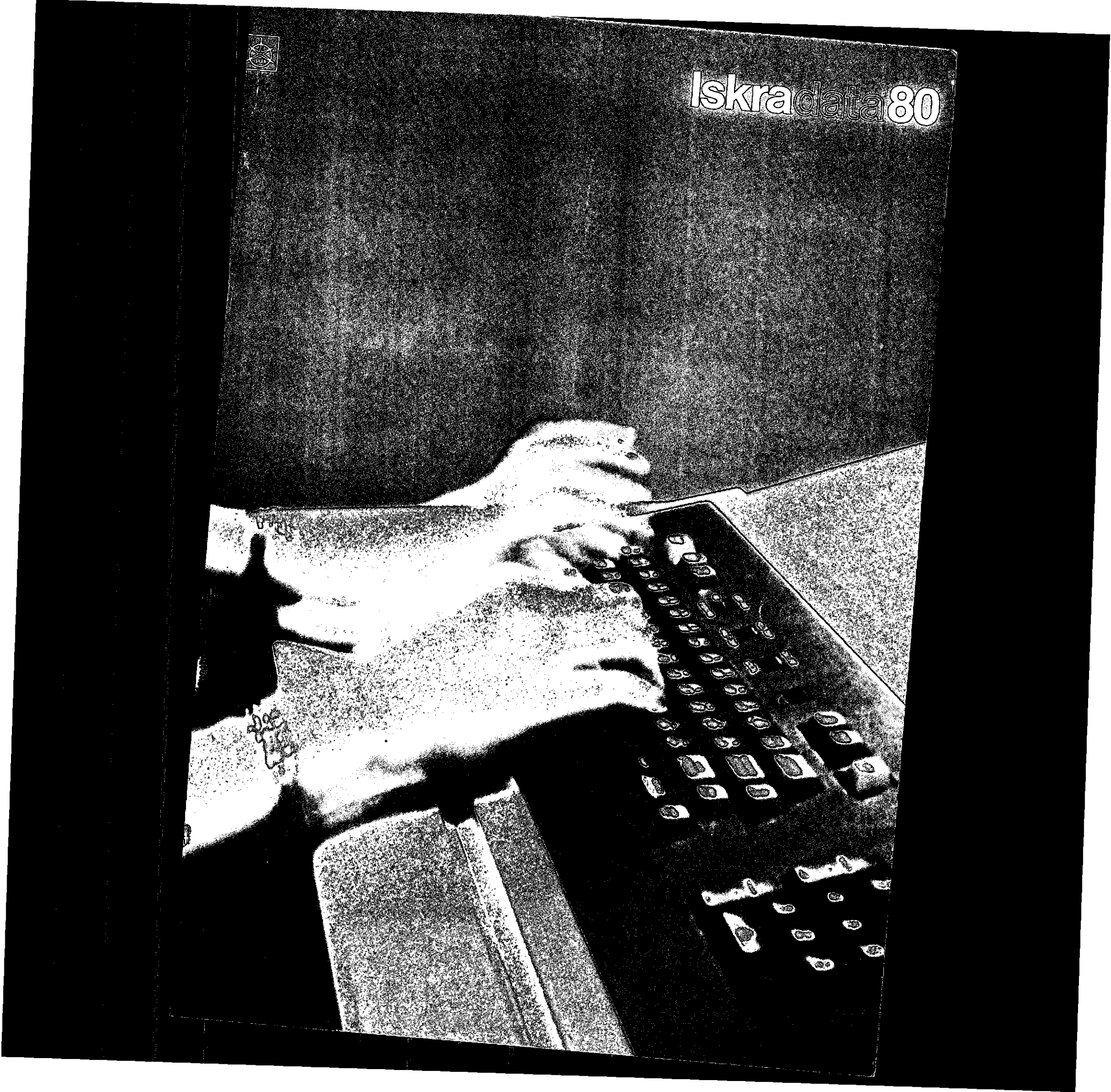
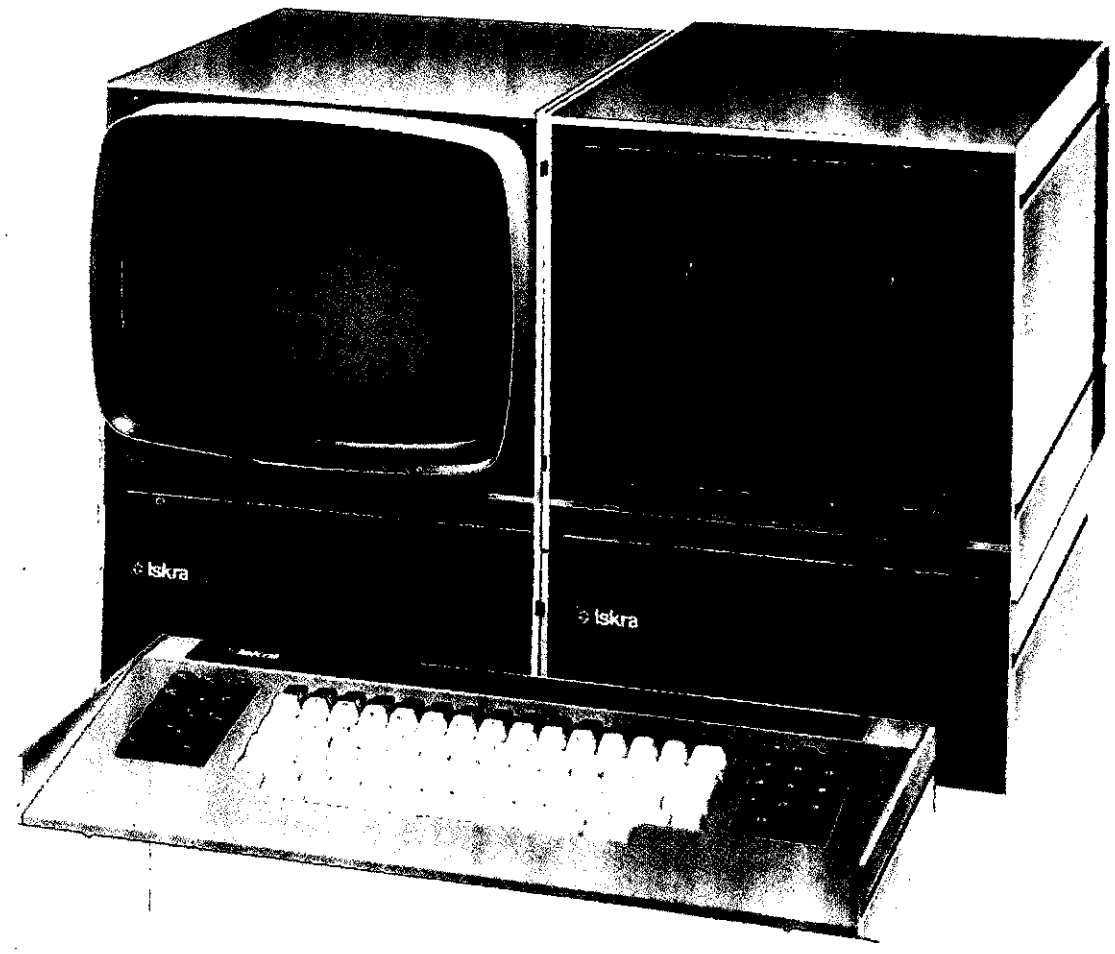
YU ISSN 0350-5596


Iskradata80

PISALNIK ISKRADATA 80

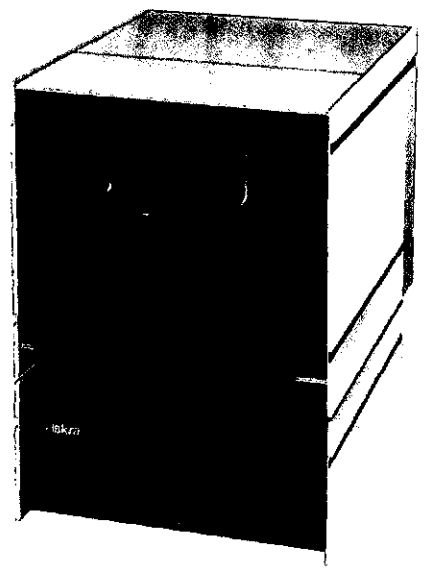


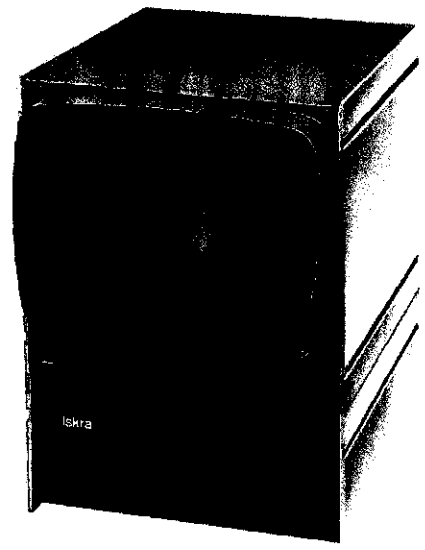
Pisalnik ISKRADATA 80 je računalnik, ki ga uporabljamo za pripravo besedila. Nadomešča nam pisalni stroj v pisarnah in drugje, ima pa številne prednosti pred navadnim pisalnim strojem. Z njim dosežemo ne samo večjo produktivnost, temveč tudi boljšo vsebinsko kvaliteto besedila in lično obliko izdelka.

Pisalnik ISKRADATA 80 je tako majhen, da ga lahko postavimo na navadno pisalno mizo, vendar imamo lahko shranjena v njem številna besedila, ki jih s pritiskom na tipko lahko takoj prikličemo na zaslon, popravljamo, sestavljamo in odtisnemo na papir. Osnovno vodilo pri izdelavi pisalnika ISKRADATA 80 je bilo, da je delo z njim preprosto in ne zahteva nobenega posebnega računalniškega ali programerskega znanja. Vsakdo se v nekaj urah lahko nauči dela s pisalnikom in tako doseže v krajšem času z manj napora boljši rezultat — vsi popravki se opravijo namreč kar na zaslonu in ko je besedilo dokončno, ga računalnik izpiše sam, brez napake. Skratka: delo s pisalnikom je pravi užitek! Stroj opravlja vsa tista opravila, ki so zamudna, naporna in utrujajoča, človek pa se lahko posveti vsebini besedila, ki ga zdaj tako na lahek način sestavlja in spreminja.

ISKRA, Industrija za telekomunikacije, elektroniko in elektromehaniko. Kranj, TOZD Računalniki, 64000 Kranj — PE Ljubljana, 61000 Ljubljana, Gruberjevo nabrežje 6. Organizacijske enote: Zagreb (041) 411-494, Beograd (011) 326-255, Skopje (091) 32-588. — Filiale: Ljubljana, Beograd, Zagreb, Sarajevo, Skopje, Rijeka, Split, Maribor, Novi Sad, Titograd, Priština, Banja Luka, Niš, Osijek.
Izdala: Iskra Commerce - - TOZD Marketing — Pravice do sprememb pridržane. — Tisk: Tiskarna Ljubljana, Ljubljana 08.80

KAKO DELAMO S PISALNIKOM?

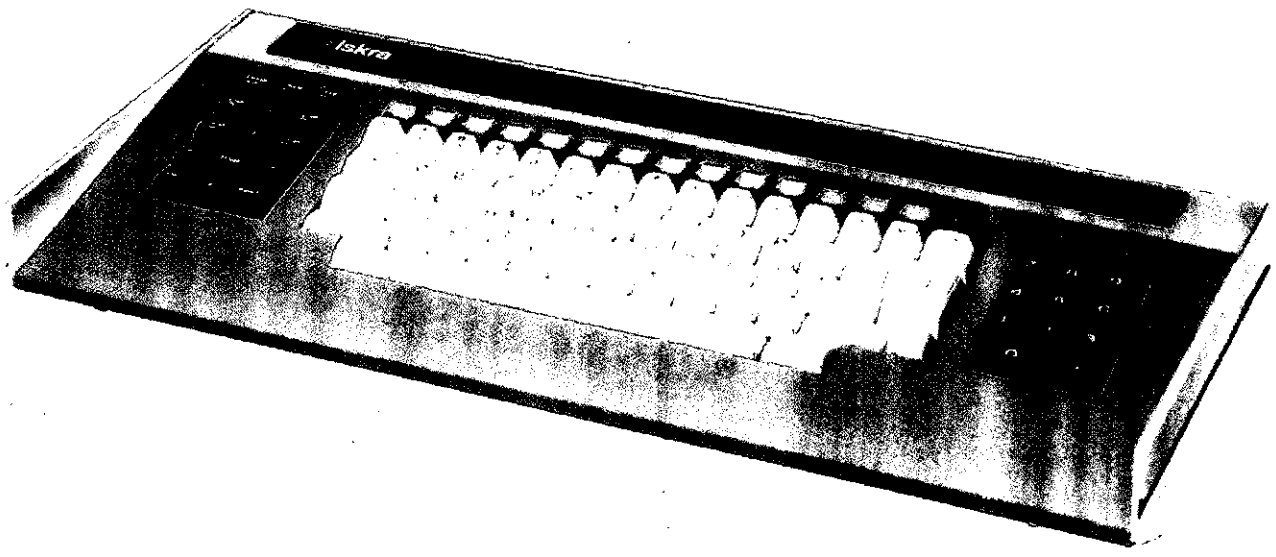




VNOS BESEDILA
se opravlja preko tastature, kjer so črke razporejene enako kot na pisalnem stroju, le da se vpisano besedilo prikazuje na zaslonu in ne na papirju. Ker pisalnik samodejno premakne besedilo v novo vrstico, ni treba paziti na poravnalni rob. S tem je prihranjenega precej časa, zlasti pri daljših besedilih.

POPRAVLJANJE
je preprosto in hitro. Ni nam več potrebno prepisati celotne strani, če želimo popraviti eno samo besedo. S pisalnikom besedilo popravljamo sproti, že ob vpisu, ali pa kasneje. S pritiskom na funkcijske tipke lahko prikličemo na zaslon katerikoli del besedila. Potem lahko brišemo, zamenjamo ali vnesemo na kateremkoli mestu znak, besedilo ali ves odstavek. Na zaslonu sproti vidimo novo, popravljeno besedilo.

OBLIKOVANJE
besedila dokončamo šele neposredno pred izpisom. S pritiskom na posebno tipko vpisano besedilo oblikujemo v poljubno obliko, ki jo tiskalnik reproducira. Z lahkoto lahko preizkusimo razne formate, od A4 stoječega, ležečega, do A3

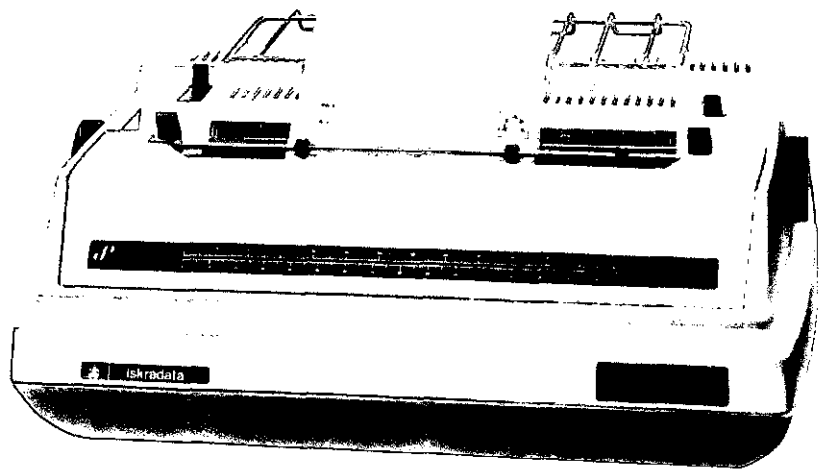


dvokolonskega formata za pomanjšano reprodukcijo, s štetjem strani, centriranjem naslovov, prostorom za slike in še marsikaj. Ta značilnost pisalnika je pomembna zlasti pri pripravi dokumentacije, kjer je oblikovanje z navadnim pisalnim strojem dokaj zamudno.

HRANJENJE
je največja prednost pisalnika. Besedilo je shranjeno na disketah in ga je mogoče vselej priklicati na zaslon. Na eno disketo lahko shranimo ca. 45 strani formata A4 tipkanega besedila. Istočasno imamo v napravi 2 disketi. Če potrebujemo več disket, jih lahko zamenjamo. Lahko si pripravimo kar knjižnico besedil, ki jih po potrebi vstavljamo v pisalnik.

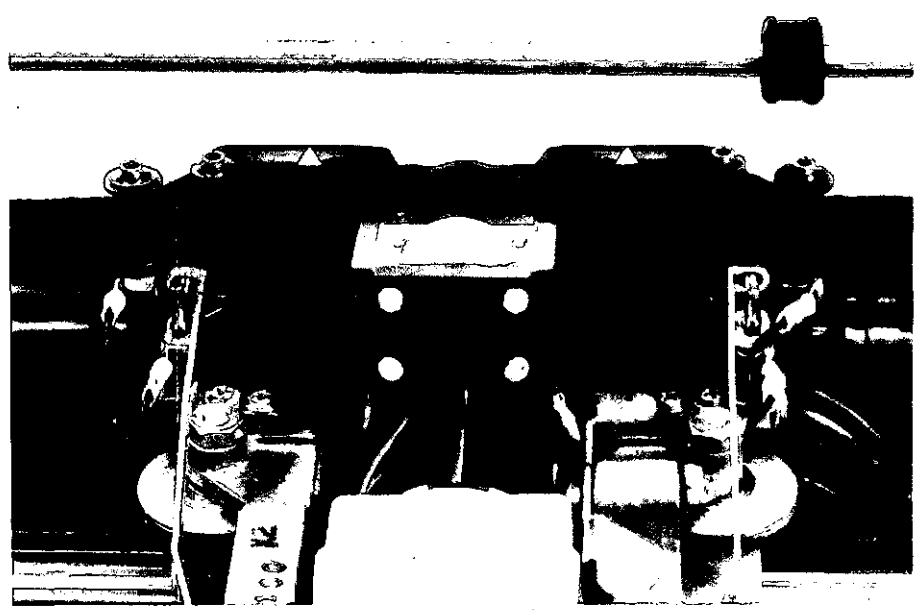
IZPIS
besedila je brezhiben. Vse napake so bile odpravljene že na zaslonu, pisalniku smo dali navodila za obliko izpisa. Poleg tega uporablja pisalnik poseben tiskalnik s krožno glavo, ki omogoča kakovost odtisa kot pri najboljših pisalnih strojih. Zato so odtisi primerni za dopise in za nadaljnje razmnoževanje. Izpis lahko ponavljamo, kolikokrat hočemo — vselej bo enak.

UPORABA
pisalnika ISKRADATA 80 je priporočljiva povsod, kjer pomeni priprava besedil ozko grlo. Kadar potrebujemo hitro izdelana dolga besedila ali večje število kratkih besedil, kjer lahko uporabimo neke standardne elemente, se pisalnik posebej obnese. Tako je na primer primeren za krožna pisma, ponudbe, pogodbe, pravilnike, sporazume. Prednosti pisalnika se pokažejo tudi pri dolgih besedilih, ki jih večkrat popravljamo, npr. zapisniki, tehnična dokumentacija, znanstvena dela. Vselej imamo kot rezultat dela: brezhiben izpis in večjo produktivnost.



TEHNIČNI PODATKI

Pisalnik ISKRADATA 80 temelji na mikroprocesorju Z 80. Zaslon s tastaturo: zaslon 1920 znakov, standardna strojepisna tastatura z numeričnimi in funkcijskimi tipkami. Zunanji pomnilnik: dvojna disketna enota, kapaciteta 2 × 256KB. Tiskalnik: D 50 s krožno glavo, izpis v obe smeri, hitrost izpisa 47 znakov v sekundi. Krožna pisalna glava s 96 znaki.



informatica

Časopis izdaja Slovensko društvo INFORMATIKA, 61000 Ljubljana, Parmova 41, Jugoslavija

Uredništvo in uprava: Informatica, Parmova 41, 61000 Ljubljana, telefon (061) 312-988, teleks: 31366 YU DELTA

Letna naročnina za delovne organizacije je 500,00 din, za redne člane 200,00 din, za študente 100,00/50,00 din, posamezne številke 100,00 din
Žiro račun št.: 50101-678-51841
Stališče uredništva se lahko razlikuje od mnenja avtorjev.

Pri financiranju revije sodeluje tudi Raziskovalna skupnost Slovenije.

Na podlagi mnenja Republiškega sekretariata za prosveto in kulturo št. 4210-44/79 z dne 1.2.1979, je časopis oproščen temeljnega davka od prometa proizvodov.

Tisk: Tiskarna KRESIJA, Ljubljana

Grafična oprema: Rasto Kirn

ČASOPIS ZA TEHNOLOGIJO RAČUNALNIŠTVA IN PROBLEME INFORMATIKE
ČASOPIS ZA RAČUNARSKU TEHNOLOGIJU I PROBLEME INFORMATIKE
SPISANIE ZA TEHNOLOGIJA NA SMETANJETO I PROBLEMI OD OBLASTA NA INFORMATIKATA

YU ISSN 0350–5596

LETNIK 5, 1981 — št. 1

VSEBINA

informatica

Published by INFORMATIKA, Slovene Society for Informatics, 61000 Ljubljana, Parmova 41, Yugoslavia

JOURNAL OF COMPUTING AND INFORMATICS

Headquarters: Informatica, Parmova 41, 61000 Ljubljana, Phone: (061) 312-988, Telex: 31366 Delta

Annual subscription rate for abroad is US $ 22 for companies, and US $ 7,5 for individuals.

Printed by: Tiskarna KRESIJA, Ljubljana

DESIGN: Rasto Kirn

YU ISSN 0350 – 5596

VOLUME 5, 1981 – No. 1

CONTENTS

K R O N I K A I N P E R S P E K T I V E

UDK: 681.3–181.4

Preteklo izdajateljsko leto (1980/81) lahko označimo kot leto poskusov, da bi se v okviru SR Slovenije dogovorili za enoten računalniški proizvodni program. Dogovori so potekali na dveh ravninah: upravljavski in tehnološki. Lahko trdimo, da so tehnologi dogovarjajočih bliže svojemu cilju: identifikaciji, klasifikaciji in ovrednotenju obstoječih in prihodnjih računalniških proizvodov Iskre, Gorenja ter Delte in njenih kooperantov.

Slovensko društvo Informatika je ustanovilo komisijo za visokošolski študij, katere prvenstvena naloga je, da v sodelovanju z ustrezno komisijo UNESCO-IFIP pripravi predlog in standard visokošolskega usmerjenega izobraževanja za področje računalništva in informatike, upoštevaje naše razmere in možnosti. Ta predlog in njegove prihodnje variacije bodo dane v širšo razpravo skupnostim strokovnih delavcev v proizvodnih, uporabniških, visokošolskih, raziskovalnih in družbenih organizacijah. Pri tem naj bi se posebej upoštevali dve dodatni izhodišči: podrobna izdelava vsebinskega dela laboratorijskih vaj predmetnika ter študijska smer računalniške proizvodnje (to dvoje ni zajeto v predlogu komisije UNESCO-IFIP). K temu bi morali dodati še izhodišča za oblikovanje učbenikov v skladu s študijskimi programi.

Slovensko društvo Informatika je organizator simpozija Informatica '81 ter simpozija in razstave računalniške tehnologije Informatica '82 na Gospodarskem razstavišču v Ljubljani (skupaj z GR in Elektrotehniško zvezo Slovenije). Informatica '81 je prehodni simpozij (ob sejmu Sodobna elektronika '81), dočim je Informatica '82 sodobna povezava proizvodnega, uporabniškega, razvojnega in raziskovalnega dela računalniške tehnologije v mednarodnem prostoru. Informatica '82 je sodoben koncept povezave proizvodnje in znanja, industrije in stroke, dela in razvoja, računalniške tehnologije in njene uporabe.

Slovensko društvo Informatika širi svojo bazo: vedno več je prostovoljnega strokovnega in organizacijskega dela, vse manj dohodkovnih odnosov posameznikov z društvom. Ureja se poslovanje društva, dopolnjujejo se zbirke podatkov o udeležencih simpozijev, o članstvu, o naročnikih časopisa Informatica.

Kaj si strokovni delavci na področju računalništva in informatike še želimo? Več svobodnega, samoupravnega dogovarjanja, postavitev prave, regularne računalniške proizvodnje, manj posegov administrativnih kadrov, ki sklepajo in odločajo pragmatično, utesnjeno, nestrokovno, brez delovnih izkušenj in praktičnega znanja. Tu vidimo vzroke težav pri dogovarjanju in organiziranju tehnološko zahtevne proizvodnje, tu vidimo vzroke stagniranja kvalitete pri izobraževanju strokovnih kadrov. Administrativno izdelujemo gore neuporabne filozofije in sporna gradiva uporabljamo za usmerjanje razvoja in izobraževanja pa tudi za zaviranje prodora kakovostnih strokovnih dejavnikov. Traktati, spisi, publikacije, elaborati, študije in psevdoznanstvena dela se neselektivno financirajo, čeprav kažejo vidno stopnjo strokovne neprizadevnosti in nesposobnosti.

Kljub navideznim in dejanskim oviram nastaja v SFRJ nova industrijska panoga, za katero je značilno kopičenje sposobnosti v razvoju, proizvodnji, prodaji, inženirinqu in strategiji poslovanja: to je tudi mikroračunalniška industrija, ki bo dovolj domača in samonikla, z minimalnim uvozom, tako da bo moč govoriti o nedevizni mikroračunalniški proizvodnji. S primernim združevanjem, usmerjanjem, produktivnostjo in ustvarjalnostjo novih kadrov bo moč doseči samostojno in neodvisno akumulacijo sredstev za nujen nadaljni razvoj mikroračunalniške industrije.

V naslednjih treh letih bo domača računalniška industrija nudila domačemu in predvsem tujemu tržišču navadne in inteligentne terminale, male poslovne sisteme, podsisteme velikih računalnikov ter šolske, laboratorijske in pisarniške sisteme. Sami bomo izdelovali tudi določene segmente računalniške periferije, preklopne usmernike ter zadovoljiv asortima kakovostne uporabniške in sistemske programske opreme.

Prihajajoče leto ni samo ustalitveno, je tudi tehnološko in organizacijsko napredujoče na področju domače računalniške proizvodnje, njenega utrjevanja in njene organizacijske strokovnosti. Tudi časopis Informatica je glasnik novih naporov in tribuna strokovnih delavcev pri uvajanju novih tehnologij, novega razvoja in raziskav ter kadrovskega izpopolnjevanja. Časopis Informatica se aktivno vključuje v nove tokove tehnološkega in proizvodnega razvoja s pomočjo vaših prispevkov, vaše zavzetosti, ustvarjalnosti in strokovnosti.

Anton P. Železnikar

INFORMATICA 1/1981

MOŽNOSTI RAZVOJA MIKRORAČUNALNIŠKE TEHNOLOGIJE V SFRJ

ANTON P. ŽELEZNIKAR,
DRAGO NOVAK

UDK: 681.3–181.4 (497.1)

SOZD ELEKTROTEHNA, DO DELTA

Članek opisuje razvoj mikroračunalniške tehnologije predvsem s stališča proizvodnje in njenih možnosti v SFRJ. Opisuje razmerja med mikro in miniračunalniško tehnologijo, kjer nakazuje možnosti prevzema funkcij današnjih minisistemov s pomočjo novih mikroprocesorjev. Kratko je prikazano stanje mikroračunalniške tehnologije v svetu s stališča sistemske arhitekture in novih operacijskih sistemov. Nakazana so izhodišča dosedanjega razvoja domače mikroračunalniške proizvodnje s preglednim proizvodnim spektrom (komponente, centralne enote, periferne enote, terminali, programska oprema). Izčrpneje je analizirana programska oprema in njeni standardi za mikroračunalnike ter zmogljivost sodobnih mikroprocesorjev v primerjavi z obstoječimi 16- in 32-bitnimi miniračunalniki. Na koncu članka so opisane možnosti domače mikroračunalniške proizvodnje s kritičnim pogledom na obstoječe stanje.

Possibilities of Microcomputer Technology Development in Yugoslavia

This article describes a microcomputer technology development from the standpoint of production and its possibilities in Yugoslavia. Some differences existing between micro and minicomputer technology are pointed out where microcomputers are taking over the market and applications from minicomputers. Briefly, the state-of-the-art of microcomputer technology is shown from the standpoint of system architecture and new operating systems. Some starting-points of domestic microcomputer production are discussed showing the wide spectrum of products (integrated components, main frames, peripherals, terminals and software). Further, an analysis of the future software and its standards for microcomputers and a comparision of coming microprocessors with existing 16- and 32-bit microcomputers is given. At the end of the article new possibilities of production and organization of domestic microcomputer industry are discussed and a critical view to this subject is given.

1. Uvod

Mikroračunalniška tehnologija nudi možnosti za samostojno računalniško proizvodnjo in uporabo tudi deželam v razvoju, še zlasti pa srednjerazvitim. Na tem področju so pričakovanja za samostojno rast, za sodelovanje s tujimi partnerji predvsem na področju trženja in s tem za pokrivanje ali celo preseganje razmerja izvoz/uvoz uresničljiva in vstop domače industrije na to področje proizvodnje in uporabe ne samo gospodarsko upravičen, marveč tudi razvojno nujen.

Računalniško proizvodnjo lahko osvajamo z različnimi strategijami, npr. začenši od čiste licenčne, prek koncepta proizvodnje izvirnih naprav (PIN = OEM) do samostojno razvitih in proizvedenih sistemov. To je industrijski (organizacijski) del koncepta. Drugo vprašanje je vsebina proizvodnje oziroma njen asortima in to vprašanje se lahko postavlja s stališča srednjerazvite družbe, njenih veljavnih odnosov, zakonitosti in možnosti.

Računalniški sistemi imajo med tehničnimi sistemi najvišjo stopnjo kompleksnosti in ta kompleksnost se odraža tudi na znanje, razvoj, proizvodnjo in trženje. Osvajanje dejanske računalniške proizvodnje je zato lahko le postopno, od manjše kompleksnosti k večji, od mikroračunalniških sistemov naprej. Tak koncept bi se tudi dobro ujemal z uporabo malih sistemov v vrsti dejavnosti gospodarstva, administracije, konstruktorstva in razvoja, kjer veliki in srednji sistemi največkrat niso gospodarni.

Druga smer, ki prihaja iz mikroračunalniškega (procesorskega) področja, je tehnološka in njeno sporočilo je, da bodo novi mikroprocesorji prevzeli sčasoma funkcije današnjih minisistemov (PDP, VAX) in poslovnih sistemov (IBM). Tehnološke zmogljivosti novih mikroprocesorjev bodo presegle zmogljivosti današnjih mini in poslovnih sistemov in z novimi perifernimi napravami (diski tipa Winchester, mehurčni pomnilniki itn.) bo moč oblikovati prave megasisteme z več centralnimi in perifernimi procesorji.

To tehnološko sporočilo nalaga, da se lotevamo načrtovanja proizvodnje mikroračunalniških sistemov z resnostjo in pozornostjo, saj je to področje v naših razmerah in pogojih dela uresničljivo in gospodarsko upravičeno. Kot že napisano, pa predstavlja tudi perspektivne možnosti za naprej, če se dovolj hitro vključujemo v domače in tuje tržišče.

Iz napisanega izhaja, da so možnosti razvoja mikroračunalniške tehnologije ter z njo povezane infrastrukture dovolj vzpodbudne, da se lotevamo resneje kot doslej razvijanja teh tehnoloških dejavnosti s pripadajočo proizvodnjo, razvojem, znanjem in prodajo.

2. Razmerja mikro in miniračunalniške tehnologije

Še pred leti je bila klasifikacija računalnikov v mikro, mini in velike računalnike dokaj jasna. Veliki računalniki so pomenili paketno obdelavo; s problemi smo prihajali v računske centre, kjer so se ti računalniki nahajali. Miniračunalniki so obvladovali vodenje procesov in manjše poslovne aplikacije. Reševali so torej probleme na licu mesta. Vedeli smo, da sodijo računalniki podjetij IBM, Borroughs, CDC itd. k velikim, računalniki podjetij DEC, Hewlett Packard, Data General itd. pa v razred miniračunalnikov. Objektivno merilo za klasifikacijo pa je bila procesna moč. Prvi mikroračunalniki so po tem kriteriju precej zaostajali za obema skupinama. Svoje področje uporabe so našli v zamenjavi in izboljšavi ožičene logike, v posameznih avtomatskih napravah in pri vodenju zelo enostavnih procesov. Odprli so tudi novo dimenzijo uporabe računalnikov, kot zasebni računalniki in računalniki za hobi. Od miniračunalnikov so se razlikovali po precej nižji ceni, manjših dimenzijah in nižji zmogljivosti. Ena od glavnih zaprek za uspešno konkurenco miniračunalnikom je bila visoka cena perifernih naprav. Šele pojav cenenih perifernih naprav, kot so pogoni za gibke diske, Winchester diski, ceneni tiskalniki itd., je omogočil mikroračunalnikom konkurenčno razmerje cena - zmogljivost.

16-bitni mikroračunalniki, ki so svoje ime ohranili zaradi svojih 8-bitnih prednikov in ker so se rodili v podjetjih, ki jih že kar identificiramo z mikroprocesorji (Intel, Motorola, Zilog), so izpolnili prej obstoječo vrzel med mini in mikroračunalniki. S svojo zmogljivostjo dosegajo in celo presegajo miniračunalnike. Meje postajajo manj ostre. Mikroračunalniki bolj in bolj izpodrivajo miniračunalnike iz področja vodenja procesov. Zelo jasen dokaz zato so napovedi, da bo stekel prvi prevajalnik za programirni jezik ADA na mikroračunalniku. Smisel delitve računalnikov na mikro in mini bo postal še bolj vprašljiv v bližji prihodnosti (1983), ko naj bi se pojavili na trgu 32-bitni mikroprocesorji, ko bo mogoče za nekaj tisoč dolarjev kupiti namizni TRS 370/135. Ustanovljena je tudi že mikroprocesorska interesna skupina "Skupina 380", ki pripravlja mikroprocesorske produkte za podjetja Intel, Motorola in IBM. Bodoči mikroračunalniki bodo torej programsko kompatibilni s serijama 370 in 380 (veliki računalniki podjetja IBM). V kratkem bomo doživeli, da nam računalniška industrija ne bo več prodajala mikro, mini in velike računalnike, temveč rešitve na osnovi visoke tehnologije, ki na trenutni stopnji razvoja še nosi ime mikroračunalniška tehnologija.

3. Stanje mikroračunalniške tehnologije v svetu

Poglejmo si nekaj osnovnih potez razvoja mikroračunalniške tehnologije v zadnjem letu. Opazna je cepitev pri arhitekturi centralnega - procesorskega oz. računalniškega integriranega vezja. En poganjek gre v smer vedno zmogljivejšega procesorja, drugi pa v smer integracije procesorja, pomnilnika in vhodno-izhodnih kanalov v eno samo integrirano vezje (single chip computer). Na področju mikroračunalniških sistemov prihaja do standardizacije na materialnem (vodilo) in programskem nivoju (operacijski sistemi, ki temeljijo na UNIXu). Pomnilniška integrirana vezja postajajo vse zmogljivejša. Pri dinamičnih pomnilnikih je poleg težnje po čim večjih obsegih opaziti še težnjo po poenostavitvi osveževanja (psevdostatični pomnilniki). Na trgu so se pojavili tudi mehurčni pomnilniki, po katerih je žejno vpila vrsta aplikacij na področju telekomunikacij in vojaških sistemov.

16-bitni mikroprocesorji, ki so bili najavljeni v letu 1978, so že dobavljivi. Stari, 8-bitni računalniki že imajo svojo domeno uporabe in nove uporabe so v polnem razmahu. Poglejmo na kratko 16-bitne procesorje. Mednje spadajo: Intel 8086, Zilog Z8000, Texas Instruments 9900, Texas Instruments 9940, Fairchild 9440, Motorola 68000, Ferranti F100L, Data General mN602, National 16016, Advanced Micro Devices 29116. Od teh so najpopularnejši 8086, Z8000 in M68000, ki so realizirani v MOS tehnologiji in jih lahko štejemo med "prave" procesorje. V MOS tehnologiji sta realizirana tudi DG mN602 in National 16016. Procesorji F 100-L, AMD 29116 in F 9440 pa bazirajo na bipolarni tehnologiji. TI9900 spada tudi še med prave procesorje, medtem ko njegov sorodnik TI9940 vsebuje 128 zlogov RAM pomnilnika in 2K zlogov ROM pomnilnika in spada v skupino mikroračunalnikov v enem integriranem vezju.

16-bitni mikroprocesorji imajo razširjeno podatkovno in naslovno vodilo. Že samo ime nam pove, da obsega podatkovno vodilo 16 bitov. V splošnem so ukazi definirani nad naslednjimi osnovnimi podatkovnimi tipi: bit, polzlog, zlog, beseda, dvojna beseda. Z razširitvijo naslovnega vodila se je povečal obseg neposredno naslovljivega pomnilnika nad standardnih 64 K zlogov. Pri 8086 je ta obseg 1M zlogov, pri Z8001 in M68000 pa 8M zlogov. Poleg razširitve sistemskih vodil prinašajo 16-bitni mikroprocesorji še celo vrsto novosti: segmentiranje, dva načina izvajanja (uporabniško, sistemsko ali nadzorno), ukaze, ki omogočajo sinhronizacijo v multiprocesorskem okolju itd. Pri nekaterih procesorjih so na voljo ukazi, namenjeni za razvoj visokih jezikov. Splošen sklep je naslednji: 16-bitni procesorji predstavljajo novo stopnjo v razvoju mikroprocesorske tehnologije, po zmogljivosti prekašajo 8-bitne procesorje (hitrost, obseg direktno naslovljivega pomnilnika, dolžina besede). Poleg tega prinašajo kvalitetne novosti, ki bodo omogočale enostavnejšo in hitrejšo realizacijo operacijskih sistemov in prevajalnikov ter enostavnejše združevanje v zmogljive večprocesorske sisteme.

Lani smo lahko zabeležili tudi pri pomnilnikih novo pridobitev - integrirano vezje s pomnilniško kapaciteto 64 K bitov. Nekaj mesecev pred tem so se pojavila na trgu enonapetostna 16 K-bitna integrirana vezja. Po napovedih naj bi šel razvoj po enakih stopinjah naprej. V trietnih intervalih naj bi se pojavila 256 K bitna in 1M bitna pomnilniška integrirana vezja. Pomnilniška vezja dobavlja na trg več proizvajalcev. Med seboj se ta vezja funkcijsko ne razlikujejo. Razlika je le v velikosti in izvedbi (maska) silicijeve ploščice. Zaradi izjemno visoke integracije postajajo pomnilniki občutljivejši na alfa delce, ki se pojavljajo pri radioaktivnem razpadu onesnaženj v materialu. Sistemski razvijalci se morajo zavedati možnosti takih napak (soft bit errors).

Že prej smo poudarili težnjo proti poenostavitvi osveževanja. Letos naj bi se na trgu pojavila vezja 4Kx8 bitov in 8Kx8 bitov z notranjim osveževanjem.

Mehurčni pomnilniki se odlikujejo po tem, da trajno hranijo informacijo in da ne vsebujejo gibljivih mehanskih delov kot nekatere druge naprave. To jim daje v mnogih uporabah prednost pred cenejšimi diskovnimi in tračnimi enotami. Trenutno je največja kapaciteta ene enote 1M bit. Čas dostopa se giblje med 10-40 ms za različne proizvajalce. Glavni proizvajalci mehurčnih pomnilnikov so podjetja Texas Instruments, Rockwell International, Intel Magnetics, Fujitsu in National Semiconductor. Poleg sa-

mih mehurčnih pomnilnikov so potrebna za pogon pomnilniških enot še periferna integrirana vezja in krmilnik. Ta dodatna vezja oblikujejo pravilne časovne poteke tokov in omogočajo odčitavanje vsebine. Krmilnik olajšuje dostop do posameznih naslovljivih blokov.

4. Dosedanji razvoj domače mikroračunalniške proizvodnje

Čeprav je mikroračunalniška proizvodnja brzkone edina računalniška proizvodnja pri nas, ki jo je v tem trenutku mogoče postaviti na čisti dinarski uvoz, se zaenkrat ni uveljavila kot redna industrijska dejavnost. Pod čistim dinarskim uvozom razumemo nedevizni uvoz tistih sestavnih delov, ki se v Jugoslaviji ne proizvajajo in katere lahko uvažamo preko maloobmejne izmenjave ter za klirinška sredstva. Na tej osnovi bi bilo danes mogoče proizvajati tudi zahtevnejše mikrosisteme, ki bi v funkcionalnih zmogljivostih dosegali standardne uvožene mikrosisteme (Cromemco, Altos, Tandy, Apple itn.).

V Sloveniji smo imeli poskuse proizvodnje domačih mikrosistemov, ki so bili v celoti razviti doma, vendar v razvoju nismo šli dosledno do konca zlasti na področju sistemske programske opreme, dočim je proizvodnja zastala zaradi meglenih in nejasnih poslovnih odločitev. To, kar se je občasno pojavljalo na tržišču, so bili laboratorijski primerki, izdelani kot industrijsko blago.

Seveda pa to ne pomeni, da v Sloveniji ne tečejo priprave za pravo proizvodnjo mikrosistemov in da ne smemo v skorajšnji prihodnosti pričakovati konkurenčen plasma več proizvodov in več proizvajalcev.

Pod domačo mikroračunalniško proizvodnjo moramo razumeti predvsem tisti tip proizvodnje, ki temelji na domačem razvoju, kjer je pretežni del proizvedene vrednosti dinarski, dočim veljajo licenčni odnosi le za tiste dele sistema, ki pogojujejo povezljivost z obstoječim mednarodnim tržiščem in s tem konkurenčnost. Nakup avtorskih pravic (licenčnina) na področju sistemske programske opreme je skorajda nujen iz dveh razlogov:

1) v primeru samostojnega razvoja bi morali razviti na določen način z nekim znanim operacijskim sistemom (OS) združljivi sistem, kar bi bilo povsem mogoče in ne bi bistveno vplivalo na ceno proizvoda;

2) kljub samostojnemu razvoju ustreznega OS izdelek ne bi imel t.i. reference, ki daje večje možnosti za plasma , za primerljivost izdelka, predvsem pa za enakovredno navajanje njegove prilagodljivosti na različne sistemske in uporabniške programske pakete.

Spremenjene poslovne razmere v Jugoslaviji bodo povzročile drugačen pristop k mikroračunalniški proizvodnji in organizaciji. Predvsem se mora ta proizvodnja oblikovati po navzgornjem načelu, izhajajoč iz tržišča in povezovanja več proizvodnih sektorjev, vključno z zasebnim. Ta nova proizvodnja se mora osposobiti tudi za ustrezno sooblikovanje tržišča z načelom, da bodo le dovolj racionalni in ceneni izdelki tisti, ki se bodo proizvajali v večjih količinah. Svoj delež bi naj dodala tudi državna administracija s podporo domači proizvodnji in vzpodbujanju lastnega razvoja, kjer bi restrikcije vplivale kvalitetno, ne pa zavirajoče. Ob tem bi morali nujno omejiti uvoz podobnih izdelkov z visoko carinsko stopnjo ter preusmerjati devizne kupce na domače izdelke.

5. Zgradba mikroračunalniških sistemov

Namen tega poglavja je prikaz kompleksnosti mikroračunalniških sistemov z navedbo in kratko analizo komponent, kriterijev in pogojev za proizvodnjo. V gospodarskih, političnih, upravljalskih, odločitvenih in administrativnih skupinah in organih največkrat ni prisotno znanje o mikroračunalniškem kompleksu, ki bi pripeljalo do ustreznih odločitev. Med temi podatki manjkajo zlasti analitični, ki kažejo stopnjo rasti, proizvodnje in porazdelitve posameznih dejavnosti oziroma komponent sistema.

V obračunskem letu 1979 (konec junija 1980) so nekatera ameriška podjetja za proizvodnjo mikroračunalnikov pridelala največje povečanje proizvodnje glede na prejšnje leto, in sicer Apple 650 %, Commodore 150 % in Tandy 131 %. Razdelitev dohodka med posameznimi komponentami ameriške računalniške industrije pa je bila tale: centralne enote 16 %, miniračunalniki 10 %, periferija in terminali 45 %, usluge in programska oprema 25 % ter ostalo 4 %. Ta razmerja so zanimiva predvsem zaradi tega, ker je moč potegniti določene vzporednice za nastajajočo mikroračunalniško industrijo.

Mikroračunalniška tehnologija obsega proizvodnjo tehle komponent in podsistemov:

1. Komponente
   - centralni mikroprocesorji
   - pomožni centralni krmilniki
     - procesor za upravljanje pomnilnika
     - razširitve centralnega procesorja
   - periferni krmilniki
     - za diske, trakove
     - za komunikacijske kanale
     - za kriptiranje itd.
   - pomnilniška vezja RAM, ROM, PROM, mehurčna itn.
   - povezovalna integrirana vezja za vodila, V/I itn.

2. Centralne naprave
   - enota s centralnim procesorjem
   - pomnilniški moduli
   - periferni moduli
   - usmerniki

3. Periferne enote
   - gibki diski
   - diski tipa Winchester (trdni diski)
   - tračne enote
   - tiskalniki
   - teleprinterji itn.

4. Terminali
   - tastature
   - navadni terminali
   - inteligenčni terminali itn.

5. Programska oprema
   - operacijski sistemi
   - prevajalniki
   - sistemi za varnost podatkov
   - diagnostični programi
   - uporabniški programi itn.

Kot vidimo, je asortima podsistemov v mikrosistemu prav tako bogat, kot ga imajo veliki sistemi ali minisistemi, le da so mikrosistemi manjši, čeprav lahko pride v posameznih primerih do stikanja (v kompleksnosti obdelave ter v zmogljivosti konfiguracij) z minisistemi. V klasifikaciji materialnih konfiguracij računalniških sistemov ločimo tri (kompleksnostne, zmogljivostne) ravnine, ki jih poimenujemo z

megaračunalniki (MR),
miniračunalniki (mR) in
mikroračunalniki (μR)

Na področju μR mora domača industrija težiti h

cilju, da postopno osvoji proizvodnjo centralnih naprav, perifernih enot (gibkih, trdnih diskov in tiskalnikov), terminalov (navadnih in inteligenčnih) ter programske opreme, predvsem uporabniške. Na sedanji stopnji razvoja ter usmerjenosti mednarodnega, zlasti razvitega tržišča, pa bo težje ali kar nemogoče osvojiti proizvodnjo konkurenčnih integriranih komponent (mikroprocesorji, periferni pomnilniki) in sistemske programske opreme, zlasti za tehnološko najnovejše računalniške (procesorske) sisteme. Procesorji in pripadajoča sistemska programska oprema določajo vnaprej (pred nami in našo ponudbo) določeno tržno usmerjenost, naravnanost in namembnost računalniških proizvodov, zato bi se domači procesorji in lastna sistemska programska oprema le težko vključevali v mednarodno tržišče, tudi če bi bili proizvodi tehnološko boljši od drugih.

6. Programska oprema za mikroračunalnike in njeni standardi

Tako kot so visoki programirni jeziki standarizirani, se postopoma standarizirajo tudi operacijski sistemi za mikroračunalnike. Standardni operacijski sistemi so določeni za

8-, 16- in 32-bitne

mikroprocesorje. K temu pa je potrebno dodati še mikroprocesorske tipe, ki oblikujejo npr. družine
(8080A, 8085, Z80)
(68000), (Z8000) itn.

Standardni operacijski sistemi so v bistvu premestljivi računalniški kodi, zaradi tega je mikroprocesorski tip bistven podatek. Ti standardi torej niso te vrste, da bi jih bilo moč uporabiti kar povprek na vseh vrstah mikroprocesorjev. Npr. standard za družino (8080A, 8085, Z80) ni uporabljiv za družino (6800), pa tudi standard za družino (Z80) vobče ne bi bil uporabljiv za družino (8080A, 8085).

Standarizacija na osnovi operacijskega sistema ima zelo široke možnosti za proizvodnjo nove sistemske in uporabniške programske opreme in s tem za njeno uveljavitev in možnosti masovne prodaje.

Vzemimo kot primer mikroračunalniški OS z imenom CP/M (Control Program/Microcomputers), ki se prodaja za procesorske družine (8080A, 8085, Z80), (8086) in (Z8000). Dejansko imamo v tem primeru tri različne operacijske sisteme, ki bi jih lahko kratko označili z

CP/M8080, CP/M8086 in CP/MZ8000

Za te operacijske sisteme obstaja nadaljna sistemska (prevajalniki, diagnostika) in aplikativna programska oprema (procesorji teksta, mali poslovni sistemi), ki bi jo lahko ustrezno označili z
S8080, S8086, SZ8000 in
A8080, A8086, AZ8000 itn.

Na ta način dobimo operacijske / sistemske/ aplikativne programske pakete, ki bi jih označili z
(CP/M, S, A) 8080,
(CP/M, S, A) 8086,
(CP/M, S, A) Z8000

itn. Vobče bi tako veljalo

⟨programski paket⟩ :: =
(⟨operacijski sistem⟩
⟨dodatna sistemska programska oprema⟩
⟨raznovrstna aplik. programska oprema⟩)
⟨tip mikroprocesorja⟩

Z razvojem in proizvodnjo nove programske opreme narašča zmogljivost oziroma moč mikroračunalniških sistemov. Dve napovedi s področja mikroračunalniške programske opreme sta izredno aktualni:

(1) Podjetje Intel bo vpeljalo jezik za sistemsko programiranje (ADA) v svoj novi 32-bitni mikroprocesor 432 in
(2) podjetje Microsoft je pripravilo različico operacijskega sistema UNIX (standard za 16-bitne procesorje, ki je bil razvit v Bellovih laboratorijih).

ADA (Ada je ime prve ženske, ki je programirala)je jezik, ki združuje lastnosti vrste prejšnjih jezikov. Jezik PL/I je imel podoben cilj, ima danes številne privržence, nima pa še take priljubljenosti, kot jo je pričakovalo podjetje IBM. Jezik ADA so razvijali 5 let ter ima tele cilje:

1) ADA naj omogoča modularno in strukturirano programiranje ter navzdoljni razvoj, ki je znan že iz PASCALa. Uvajanje strukturiranih zbirnih jezikov ter strukturiranih različic jezikov BASIC in PASCAL potrjujejo priljubljenost teh metod.

2) ADA je mogoče uporabiti za obdelavo poslovnih podatkov ter za programiranje v pročesih in v realnem času. To privede do zadostnosti enega samega programirnega jezika, kar olajšuje šolanje, vodenje programirnih projektov, vzdrževanje in dokumentacijo.

3) ADA je kombinacija splošnega, uporabniškega in sistemskega programirnega jezika. Takšen jezik omogoča pisanje uporabnostnih programov, operacijskih sistemov, prevajalnikov, komunikacijskih paketov ter interaktivnih sistemov.

4) ADA omogoča programiranje v realnem času, v interaktivnih sistemih in v paralelnih procesih.

5) ADA omogoča nekatere postopke, kot so odpravljanje napak, preizkušanje, dokumentiranje in vzdrževanje programov. Jezik naj bi imel lastnosti, ki pospešujejo in podpirajo razvoj zanesljivih in spremenljivih programov ter lahkotnost odkrivanja napak, preizkušanja in dokumentiranja.

6) Jezik mora biti popolno in nedvoumno opredeljen, tako da rabi kot standard pri njegovi implementaciji na različnih strojih.

7) Jezik naj omogoča posebne metode, ki so nujne na velikih sistemih, kot je posamično prevajanje modulov ali celotnega podsistema, kot so ravnine dostopa, ki omogočajo velikim skupinam istočasno delo **brez** motenj in kot so pripomočki za avtomatičen razvoj programske opreme.

8) Jezik naj omogoča učinkovito predstavljanje podatkovnih struktur ter operacij nad strukturami brez zahtevnih okornosti in velikih programirnih naporov. Prevajalnik naj prevzame obdelavo podrobnosti.

9) Jezik naj omogoči vpeljavo več okolij za različne uporabniške ravnine, kot so sistemsko in aplikativno programiranje, razvoj paketov, delo končnih uporabnikov, glavnih operaterjev, oddaljenih uporabnikov, učiteljev in študentov. Programi in podatki določenega okolja naj bodo delno ali polno zaščiteni pred uporabniki iz drugih okolij.

10) Jezik naj zmore opis posebnih primerov, kot je posebna oblika V/I, vključitev rutin v drugih jezikih, izjem v splošnih pravilih itn.

Doseganje naštetih ciljev seveda ni brez težav, saj je potrebno upoštevati lahkotnost uporabe, učinkovitost prevedenega (objektnega) programa, dovolj majhen obseg, prenosljivost, lahkotno realizacijo prevajalnika ter

tudi združljivost s prejšnjimi jeziki na določenih področjih uporabe. Prav zaradi tega je težko pričakovati, da bo jezik ADA predstavljal dokončno rešitev programirnih zahtev in problemov.

UNIX je operacijski sistem (z delitvijo procesorskega časa), izdelan v Bellovih laboratorijih. Ta OS je bil razvit za PDP-11 pred desetimi leti. UNIX je neke vrste PASCAL na področju operacijskih sistemov, ima veliko število uporabnikov kljub majhni podpori proizvajalcev. UNIX je tako postal obredni sistem, zlasti v akademskih in razvojnih okoljih.

Podjetje Microsoft se je zaradi tega resno lotilo razvoja sistema UNIX, ki ga je poimenovalo XENIX. Ta OS naj bi se že letos prodajal za sisteme s PDP-11, Z8000, 68000 in 8086. Ta OS pomeni

(1) razvoj popolnoma novega OS
(2) dodatno programsko opremo in pripomočke za programerja in prevajalnik - prevajalnik (YACC)
(3) odpravljanje napak v novem OS
(4) uvedbo programirnih jezikov tipa C za sistemski razvoj

Nekaj drugih razvojnih smeri sistemskega razvoja pa je:

(1) razširitev sistema CP/M na 16-bitne procesorje (8086)
(2) novi standardni zbirni jezik za mikroprocesorje
(3) izdelava novih OS realnega časa ter za več (istočasnih) nalog v mikroračunalnikih.

Seveda pa se bo mikroračunalniška tehnologija soočala z vrsto prevzetih nevšečnosti:

(1) Proizvajalci mikroračunalnikov so prevzeli jezike, operacijske sisteme in drugo programsko opremo od velikih računalniških sistemov. Tehnološki napredek je mogoč le z lastnim razvojem, vključevanjem univerz ter z združevanjem dela s proizvajalci miniračunalnikov (tak primer je tudi združitev Intel-Xerox-DEC za področje mrež).

(2) Mikroračunalniški razvojni sistemi izkazujejo še vedno prenizke zmogljivosti, predvsem z neustrezno programsko opremo. Na razpolago je premalo visokih programirnih jezikov, ni standarizacije na področju programirnih pripomočkov, kot so odkrivanje napak, dokumentacija in vzdrževanje.

(3) Ravnine strokovnega znanja pri proizvajalcih so različne. Nekateri proizvajalci govorijo o obdelavi večprocesnih nalog, istočasnosti, paralelnosti, hierarhiji dostopa, zaščiti pomnilnika, drugi se še vedno ukvarjajo z makrozbirniki, z malimi kodirnimi problemi in spremembami ukaznih zalog. Med proizvajalci so tako velike razlike v kvaliteti znanja in proizvodov.

7. Zmogljivost sodobnih mikroprocesorjev

Sodobni mikroprocesorji uspešno tekmujejo s standardnimi 16-bitnimi procesnimi računalniki. Po svoji zgradbi se med seboj precej razlikujejo. Poglejmo si za začetek strukturo registrov pri posameznih mikroprocesorjih in jo primerjajmo s strukturo registrov mini računalnika PDP-11/70. Dostopni časi za registre v procesorju samem so približno petkrat krajši od dostopnih časov za pomnilnik. Zato pomeni število registrov in način uporabe važen faktor zmogljivosti.

Procesor 8086 ima štiri 16-bitne registre, ki jih je mogoče uporabljati v aritmetičnih in logičnih operacijah. Lahko jih uporabljamo kot 16-bitne ali kot par 8-bitnih registrov. Ostale štiri 16-bitne registre lahko uporabljamo samo pri 16-bitnih operacijah in kot naslovne registre.

Z8001 vsebuje štirinajst 16-bitnih registrov. Osem jih lahko sodeluje v 8-bitnih operacijah. Po dva sosednja registra je mogoče združiti v največ štiri 32-bitne registre. Pri 32-bitnem deljenju in množenju pa združimo po 4 zaporedne registre v dva 64-bitna registra. Poleg omenjenih ima procesor Z8001 še dva kazalca v sklad in programski števec.

68000 ima po osem 32-bitnih podatkovnih in naslovnih registrov, dva 32-bitna kazalca v sklad in 32-bitni programski števec.

LSI 11/23 vsebuje devet 16-bitnih registrov, od katerih je šest popolnoma identičnih in lahko služijo kot podatkovni ali naslovni registri.

PDP 11/70 ima šestnajst 16-bitnih registrov, ki so podobno, kot pri LSI 11/23 univerzalni. Poleg tega ima še tri kazalce v sklad in programski števec.

Na sliki 1 so zbrane strukture registrov v posameznih procesorjih.

Procesor 8086 ima najrevnejšo registrsko strukturo. Z8001 lahko uporabi za shranjevanje podatkov skoraj dvakrat toliko registrov kot MC68000, zato ga prekaša v hitrosti. Po drugi strani pa ima MC68000 prave 32-bitne registre in je hitrejši od vseh ostalih procesorjev pri operacijah z 32-bitnimi besedami.

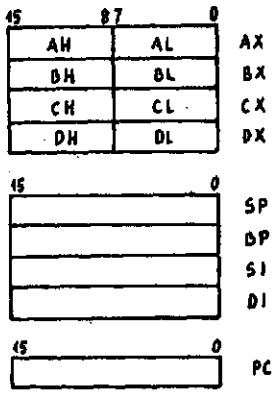
Poleg bogatejše registrske strukture se moderni mikroprocesorji razlikujejo od prejšnje generacije še po bogatejšem naboru ukazov. Za aplikacije z več procesorji so važni semaforski ukazi, ki lahko v neprekinljivem ciklu testirajo vrednost spremenljivke in ji nato dodelijo vrednost. Takšne ukaze zasledimo pri procesorjih 8086 in MC68000. Pri Z8001, LSI11/23 in PDP11/70 pa je potrebno dodatno vezje za blokiranje vodila. Ukazni nabor modernih procesorjev je prilagojen višjim programirnim jezikom. Procesorja 8086 in MC68000 imata v svojem naboru ukazov zančne ukaze. Konec zanke je mogoče definirati z logičnimi pogoji ali z iztekom števca. Z8001, LSI11/23 in PDP11/70 omogočajo le številčni pogoj. MC68000 ima ukaze, ki pri subrutinskem pozivu zasedejo del pomnilnika v skladu in ga pri vrnitvi spet sprostijo (uporabno za lokalne spremenljivke).

Način naslavljanja izvora	Z8000 (4MHz)	PDP 11/45 z 8 K
registrski	0,75	0,90
indirektni,registrski	1,75	1,88
direktni	2,25	2,78
indeksni	2,50	2,78
takojšnji	1,00	1,88

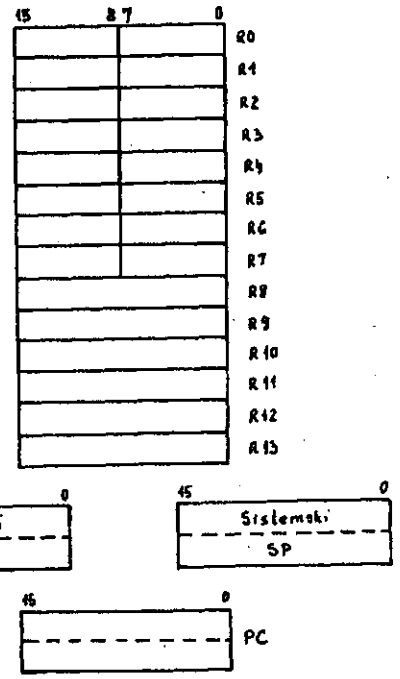
Tabela 1. Izvajalni časi za ukaz LDB R, src v mikrosekundah.

Doslej smo ugotovili, da imajo moderni mikroprocesorji dokaj bogato registrsko strukturo in da imajo v svojem naboru ukaze, ki olajšujejo multiprocesiranje in podpirajo višje programske jezike. Za konec pa si poglejmo še kako je s hitrostjo.

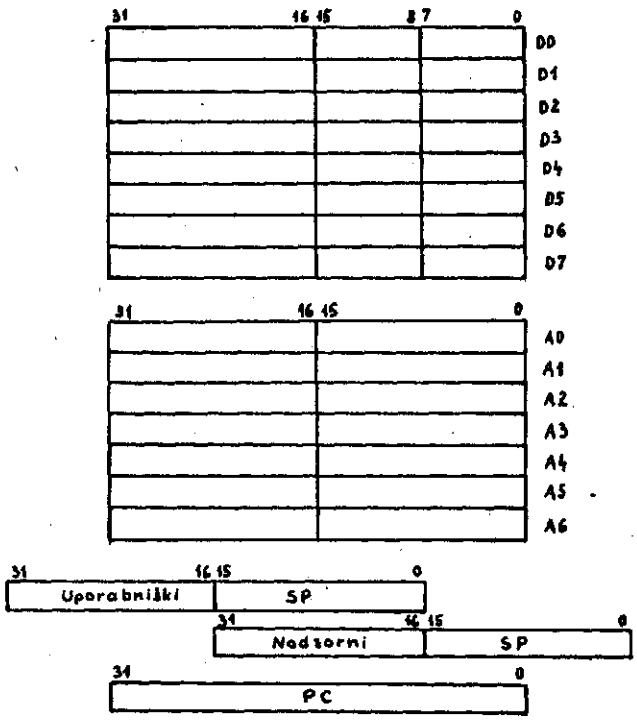
Za tipičen moderen mikroprocesor vzemimo Z8000 in ga primerjajmo s standardnim miniračunalnikom PDP 11/45. V tabeli 1 je podana primerjava hitrosti iz-



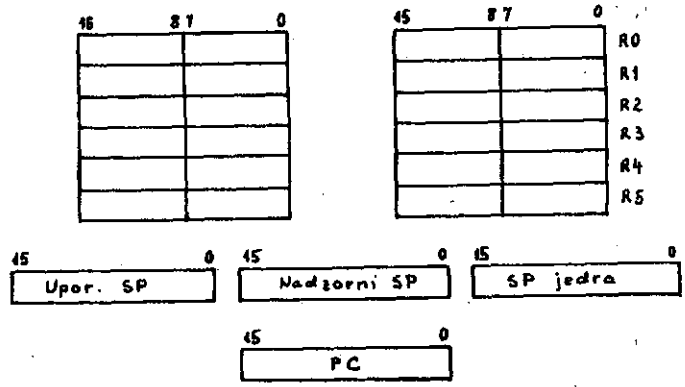
Registri procesorja 8086



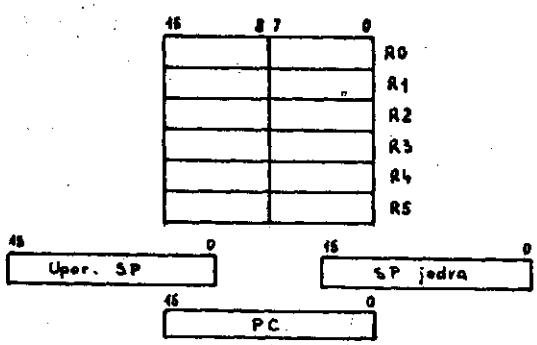
Registri procesorja Z8001



Registri procesorja MC68000



Registri PDP 11/70



Registri procesorja LSI 11/23

Slika 1.

Struktura registrov v procesorjih 8086, MC6800 , LSI 11/23, Z8001 in PDP 11/23.

Ukaz	Tip podatka	Z8000 (4 MHz) Ukazi	Zlogi	Cikli	µsec	PDP 11/45 Ukazi	Zlogi	µsec
	zlog	1	4	9	2,25	1	4	2,78
LD R, DA	beseda	1	4	9	2,25	1	4	2,78
	dvojna beseda	1	4	12	3,00	2	8	5,56
	zlog	1	4	9	2,25	2	6	3,68
ADD R, DA	beseda	1	4	9	2,25	1	4	2,78
	dvojna beseda	1	4	15	3,75	3	10	6,46
	zlog	3	8	87	21,75	2	6	6,61
MULT R,DA	beseda	1	4	70	17,50	1	4	5,56
	dvojna beseda	1	4	≈350	≈88	17	42	33,94

Tabela 2. Izvajalni časi za ukaze LD, ADD in MULT (DA - direktno naslavljanje).

vajanja ukaza LDB R, src pri različnih načinih naslavljanja. Vidimo, da je Z8000 v vseh primerih hitrejši.

Primerjajmo še nekaj ukazov pri direktnem načinu naslavljanja. Izberimo ukaze za nalaganje, seštevanje in množenje. Podatki so zbrani v tabeli 2. PDP 11/45 prekaša Z8000 pri množenju. Iz teh primerjav lahko ugotovimo, da med mikro in mini procesorji ni več prepada, ampak, da se po zmogljivosti že prekrivajo.

8. Možnosti mikroračunalniške proizvodnje v Jugoslaviji

Nekatere dosežke, možnosti in stanja v nastajajoči domači mikroračunalniški proizvodnji smo že opisali. Vprašanje o nadaljnem razvoju te proizvodnje postavimo na analizo prehojene poti in analizo novih možnosti, ki doslej niso bile upoštevane.

Preteklost je pokazala, da upravno-nestrokovni pristopi, ki so bili tipično navzdolnji (od upravnih teles k tržišču), niso bili uspešni; ugotovimo lahko celo, da so povzročili določeno

materialno in razvojno izgubo.

Ta domača računalniška izkušnja nas uči, da je potrebno z vso resnostjo upoštevati navzgornji strokovni pristop - - od tržišča k poslovnim odločitvam na področju prihodnje mikroračunalniške proizvodnje. Nakup patentov in avtorskih pravic (licenc) se očitno ne more rojevati v strokovno in tržno odmaknjenih, nekakšnih amaterskih in robustnih navdušenjih dovolj visokih vodstev, njihovih povezav in brezčasovnega taktiziranja, marveč mora izhajati iz tržne analize in inženirske ustvarjalnosti na nižjih, toda operativnih in strokovnih ravninah.

Naslednje, bistveno izhodišče prihodnje mikroračunalniške proizvodnje je ugotovitev, da obvladamo problemski prostor domače mikroračunalniške proizvodnje, in sicer:

- z lastnim razvojem
- s konkurenčnim izdelkom
- z upoštevanjem zahtev tržišča in
- z organizacijo proizvodnje

Več ali manj je tudi jasno, da lahko gradimo izvoz

- na osnovi pridobljenih izkušenj domače proizvodnje
- s kooperacijo ustreznega tujega partnerja
- s tržnimi uslugami na tujih tržiščih

Prvi in najbližji cilj je organizacija prave, regularne, dovolj kvalitetne in masovne proizvodnje mikroračunalnikov. Izkušnje nas uče, da mora biti ta pot grajena z ustreznimi koraki, od ničte proizvodnje, prek maloserijske do vse večje in rentabilnejše. Izkušnje kažejo, da je določeno prestrukturiranje izgubnih podobnih dejavnosti neučinkovito in nepriporočljivo zaradi

- nejasnih ciljev
- že oblikovanega oportunizma in
- nesposobnosti obstoječih struktur

Nova proizvodnja zahteva nove vire, materialne in kadrovske. Kot že zapisano, jih lahko oblikujemo korakoma, z manjšim tveganjem ter z večjim posluhom za tržišče in organizacijo. Nova proizvodnja mikroračunalnikov mora upoštevati svojo lastno stabilizacijo, ustrezno mora razporejati uvozne vire ter jih po možnosti reducirati na dinarske (nedevizne). Takšna organizacija proizvodnje pa zahteva:

- trdno povezavo med domačimi partnerji ter
- zanesljive odnose s tujimi partnerji na področju izmenjave proizvodov in kliringa

Določena družbena podpora nove mikroračunalniške proizvodnje bi bilo dobrodošla, saj se z njo oblikujejo večje in privlačnejše možnosti za prihodnost. Racionalna in živa proizvodnja pa bi lahko v regularnem (neintervencijskem) samoupravnem poslovnem prostoru uspevala tudi brez take podpore, s postopno rastjo kvalitete in obsega proizvodnje. Praviloma naj bi novo, lastno in samostojno mikroračunalniško proizvodnjo vzpodbujali ter ji ne odvzemali poslovnih in razvojnih možnosti z restrikcijami, tj. s spreminjanjem poslovnih pravil in z ukinjanjem normalnih delovnih pogojev. V tem naj bi bilo težišče družbene podpore, ki bi tako postala neinvesticijska, toda razvojno selektivna.

Mikroračunalniška proizvodnja temelji na

mikroračunalniški tehnologiji,

ki sodi s sociološkega vidika med t.i. transformativne tehnologije (tehnologije, ki bistveno spreminjajo način človekovega dela, razmišljanja in življenja). Uvajanje tovrstne tehnologije zahteva določene pogoje za tehnološki prodor, ki ne zajema le kopičenje materialnih, razvojnih in proizvodnih sredstev, marveč tudi kopičenje sposobnosti, tj. zajemanje, usmerjanje, produktivnost in

ustvarjalnost kadrov, od strokovnih do upravljalskih. Šele v taki mikroklimi, ki vse bolj prerašča v makroklimo, lahko računamo na vzpodbudne uspehe nove tehnologije in njene proizvodnje.

9. Sklep

Domača mikroračunalniška proizvodnja je šele v stanju svojega nastajanja, ni še niti maloserijska, nima še potrebnih tržnih analiz in izkušenj, o izvozu pa lahko zaenkrat le razmišlja. V Sloveniji obstajajo zarodki prihodnje mikroračunalniške industrije, ki se bo v naslednjih treh letih bistveno razvila na področju proizvodnje

- navadnih CRT terminalov
- inteligentnih CRT terminalov
- malih poslovnih sistemov
- enostavnejših perifernih enot
- šolskih sistemov
- podsistemov velikih računalnikov
- laboratorijskih sistemov
- pisarniških sistemov
- ohišij in montažnih enot
- navadnih in preklopnih usmernikov
- itn.

O razvoju mikroračunalniške proizvodnje v naslednjih letih ne bodo bistveno odločali nekateri nakopičeni potenciali, ki so tradicionalno obremenjeni z neproduktivnostjo, nestrokovnim vodenjem ter z utrjeno nesposobnostjo; do novih pristopov in do bistvenega prodora bo moč priti le z racionalnimi odločitvami in strokovnim organizacijskim delom: oboje pa bodo lahko nosili le dovolj sposobni in z neuspehi neobremenjeni kadri, ki bodo izpeljali razvoj, trženje in proizvodnjo po novih tirnicah.

Nova mikroračunalniška proizvodnja bo v naslednjih letih razvila tudi svoje standarde, in sicer

- materialne računalniške module
- operacijske sisteme
- uporabniške pakete

Razvoj bo šel v smer izpopolnjevanja sistemov z 8-bitnimi procesorji, začel pa se je razvoj sistemov s 16- in 32-bitnimi procesorji ter sistem z več mikroprocesorji. Prav zadnje usmeritve zagotavljajo, da bomo v nekaj letih osposobljeni za izvoz ter bomo lahko znatno dvignili tudi obseg proizvodnje.

Literatura

[1] A. P. Železnikar, Računalniška industrija: njena struktura in perspektive, Zbornik radova JUREMA 75, str. 91, Zagreb 1975.

[2] A. P. Železnikar, Razvoj računalniških sistemov, Informatika 4 (1980), št. 1, 4 - 12.

[3] A. P. Železnikar, Sodobni tokovi razvoja računalništva, Delta informator 2 (1980), št. 3, 6 - 11.

[4] I. LeMair, Microprocessors and Microcomputers, Digital Design 10 (1980), No. 12, 30.

[5] L. A. Leventhal, Microcomputing Software, Digital Design 10 (1980), No. 12, 65.

[6] D. R. McGlynn: Modern Microprocessor System Design, John Wiley Sons, New York 1980.

[7] R. Männer, B. Deluigi: 16-Bit-Processoren im Vergleich, Elektronik 30, 1981, Nr. 5, 77 - 83.

[8] P. Snigier: Minicomputers, Digital Design 10 (1980) No. 12, 26 - 28.

[9] MOS RAM: Staff Report National Semiconductor, Digital Design 10 (1980), No. 12, 35 - 36.

INFORMATICA 1/1981

PARALELNO PROCESIRANJE V IBM – SNA RAČUNALNIŠKI MREŽI

BATISTA PAVEL,
KRAIGHER MARKO

UDK: 681.3.008

RSNZ SRS

Članek obravnava nekatere vidike paralelnega procesiranja v IBM-ovem *SNA* računalniškem omrežju. Obravnavane komponente so *DOS/VSE, CICS/VS, ACF/VTAM* za računalnike razreda 370 in *ACF/NCP* za čelni računalnik 3705. Paralelno procesiranje v taki računalniški mreži obravnava *s stališča uporabnika in s sistemskega stališča.*

Some aspects of parallel processing in an IBM-SNA computer network are considered. An overwiev of DOS/VSE, CICS/VS, ACF/VTAM for IBM system 370 and ACF/NCP for IBM 3705 Communication Controller and relationships between them is given. Parallel processing in the network is presented from the user's and from the system's point of view.

1. UVOD

Čeprav se ne ujema s strogo definicijo multiprocesiranja lahko računalniško omrežje obravnavamo tudi kot neke vrste multiprocesorski sistem. *S stališča uporabnika v takem multiprocesorskem sistemu posamezni procesi ne sodelujejo med seboj zato, da bi se pospešilo delovanje posameznega procesa, ampak predvsem zaradi izmenjave informacij.*članek obravnava računalniško omrežje predvsem *s stališča* paralelnega procesiranja. Pod pojmom paralelno procesiranje v tem članku razumemo tako multiprogramiranje kot multiprocesiranje. Paralelno procesiranje je obravnavano *s stališča uporabnika in s sistemskega stališča.* Članek je omejen na prikaz *software-a*, hardware je opisan le toliko kolikor je to nujno potrebno za razumevanje. Obravnavava računalniško omrežje, ki se vklaplja v IBM-ov *SNA* koncept. V članku je obravnavano računalniško omrežje, ki ga sestavljajo trije razredi IBM-ovih računalnikov: *sistem 370* kot razred splošnonamenskih računalnikov srednje moči, čelni računalnik tipa 3705, ki kontrolira računalniško mrežo in splošen tip inteligentnega terminala(npr.:3270,3790 ali 8100).

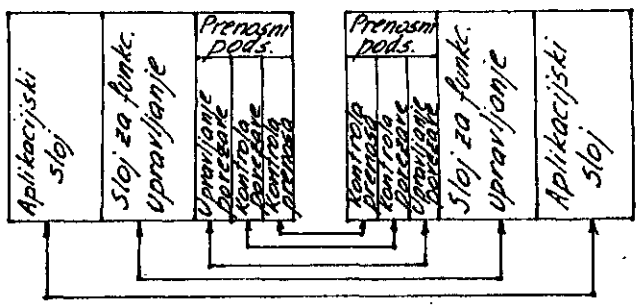
V članku najprej opisujeva posamezne programske komponente tega omrežja: *DOS/VSE, ACF/VTAM, CICS/VS* za sistem 370 in *ACF/NCP* za 3705. Seveda to še zdaleč niso vse programske komponente, ki jih je možno povezati v *SNA* koncept, opisane komponente lahko *smatramo samo* za tipične predstavnike programske opreme, ki naj ponazorijo kompleksnost pojma paralelnega procesiranja v računalniškem omrežju in različnost konceptov realizacije paralelnega procesiranja pri posameznih programskih komponentah. Posamezne programske kompongnte v članku opazujemo kot samostojne procese oziroma kot skupine koordinirano delujočih procesov.nadalje je v članku opisan princip komuniciranja in sinhroniziranja delovanja med posameznimi programskimi komponentami (npr.: relacija *CICS/VS ACF/VTAM*). Posamezne komponente in sodelovanje med njimi so opisani *s sistemskega gledišča. S stališča uporabnika* je opisano sodelovanje uporabnikovih programov, ki delujejo pod operacijskimi sistemi *CICS/VS* na različnih računalnikih razreda 370.

V članku je podan tudi kratek slovarček uporabljenih pojmov in kratic.

2. IBM-OV KONCEPT ARHITEKTURE RAČ.MREŽE – *SNA*

SNA (Systems Network Architecture) je IBM-ov koncept *arhitekture računalniške mreže. SNA* je strukturirana v tri sloje:

- aplikacijski sloj (API - Application Program Interface): obdela uporabnikove zahteve
- sloj za funkcijsko upravljanje (Function Management Layer) : posreduje informacije od enega aplikacijskega sloja do drugega
- sloj prenosnega podsistema (TSC - Transmition Subsystem Component) : skrbi za prenos podatkov



Vsak glavni računalnik (fizična enota tip 5) ima svojo kontrolno točko (*SSCP – System Services Control Point*), kateri pripadajo določene logične enote v mreži. Domeno enega glavnega glavnega računalnika sestavljajo kontrolna točka in pripadajoče logične enote. Glavni linijski protokol v *SNA* je *SDLC (Synchronous Data Link Control)*, pozna pa tudi protokole BSC (Binary Synchronous Control) in asinhrone (start-stop) protokole. V *SNA* je fizična enota vsak računalnik in vsak nadzornik terminalov v računalniški mreži.*SNA* definira pet tipov

fizičnih enot:

tip 0:ne-SNA fizične enote (asinhroni, BSC in lokalni ne-SNA terminali)

tip-1:neinteligentni terminali (starejši) npr.:3275;

tip-2:inteligentni terminali-nadzorniki npr.:3274,3276,3790;

tip-4:nadzorniki računalniške mreže npr.:3705;

tip-5:splošno namenski glavni računalniki npr.:370/3031.

Logične enote so programi v fizičnih enotah.Komuniciranje se z uporabnikovega stališča vrši med logičnimi enotami, povezanimi v sejah (session), pri čemer logične enote pri tem ne zaznajo poti povezave. Način komuniciranja se določi ob vsaki vspostavitvi seje posebej odvisno od zahtev in zmožnosti logičnih enot. Nekatere seje med logičnimi enotami so že standardizirane s tipi logičnih enot.

nekaj tipov logičnih enot:

tip-0: nestandardizirani tipi;

tip-1: 3270 printerji, ki delajo z SNA tokom podatkov

tip-2:3270 terminali, ki delajo z SNA tokom podatkov

tip-3: 3270 tiskalniki, ki delajo z 3270 tokom podatkov

tip-6: aplikacijski programi v glavnem računalniku.

3.OPERACIJSKI SISTEM - DOS/VSE

DOS/VSE (Disk Operating System/Virtual Storage Extended) je operacijski sistem za manjše računalnike razreda 370 oziroma 43xx. Omogoča paralelno izvajanje do šestnajst DOS/VSE taskov (sekvenčni procesi, katerih delovanje nadzira DOS/VSE operacijski sistem). DOS/VSE taski se izvajajo v največ dvanajstih particijah fiksne velikosti in lokacije v navideznem pomnilniku. Pri DOS/VSE operacijskem sistemu tvori naslovni prostor do 16M zlogov velik navidezni pomnilnik (učinkovito delovanje navideznega pomnilnika omogoča DAT (Dynamic Address Translation) - hardverski dodatek računalnikom razreda 370 in 43xx, ki prevaja naslove v navideznem pomnilniku v naslove v realnem pomnilniku).Particija je kontinuiran del navideznega pomnilnika - naslovnega prostora, ki je za čitanje in pisanje dostopen samo DOS/VSE taskom, ki se izvajajo v tej particiji, ostalim DOS/VSE taskom je dostopen samo za čitanje. Particijo kontrolira glavni DOS/VSE task, katerega je sprožil sam operacijski sistem na uporabnikovo zahtevo. Glavni DOS/VSE task v particiji lahko sproži DOS/VSE podtaske, ki se potem izvajajo v isti particiji. Med DOS/VSE taski znotraj ene particije ni nikakršne zaščite integritete pomnilnika. Sinhronizacija delovanja med DOS/VSE taski znotraj ene particije in uporaba skupnega dela pomnilnika poteka s pomočjo operacijskega sistema. Komuniciranje z operacijskim sistemom poteka preko posebnega tipa prekinitev (SVC supervisor call strojna instrukcija).DOS/VSE taski v različnih particijah med seboj komunicirajo na dva načina:

1. s pomočjo prekinitev preko operacijskega sistema,

2. preko dela pomnilnika, ki je za čitanje in pisanje dostopen vsem DOS/VSE taskom.

Proženje DOS/VSE podtaskov je omogočeno samo glavnemu DOS/VSE tasku v particiji.

DOS/VSE nadzira tudi delovanje kanalov. Kanali so samostojni V/I procesorji, ki delujejo asinhrono s CPE. Komuniciranje med CPE in kanali poteka preko V/I strojnih inštrukcij, v obratni smeri pa s pomočjo V/I prekinitev.

Procesor se dodeljuje DOS/VSE tasku za čas med dvema zaporednima prekinitvama na osnovi fiksno ali dinamično določenih prioritet taskov. Operacijski sistem ob vsaki prekinitvi dodeli procesor tasku z najvišjo prioriteto, ki ne čaka na izvršitev neke V/I operacije.

Iz prej navedenih dejstev sledi da je paralelno procesiranje, ki ga podpira DOS/VSE operacijski sistem vsaj za sedaj še precej omejeno - največ 16 paralelnih procesov (to pa že rešuje nova verzija, ki dopušča 256 sočasnih taskov), možnost proženja podtaskov samo do prvega nivoja, particije fiksne velikosti, proženje glavnega DOS/VSE taska samo na klasičen način preko krmilnih ukazov operacijskemu sistemu). Pomanjkljivosti izvirajo verjetno iz tega, ker je bil operacijski sistem zasnovan za paketni način izvajanja poslov na računalniku. Na prvi pogled bi iz tega lahko sklepali, da tak operacijski sistem ni primeren za vključevanje v računalniško mrežo, za zadovoljevanje 'on line' zahtev uporabnikov.

4.ACF/VTAM

ACF/VTAM (Advanced Communication Function/Virtual Telecommunications Access Method) je SNA pristopna metoda za komuniciranje v računalniški mreži. To je program, ki se izvaja pod raznimi operacijskimi sistemi npr.: pod DOS/VSE kot task v eni od particij. Realiziran ima svoj lasten multitasking, neodvisen od multitaskinga operacijskega sistema. ACF/VTAM je sestavljen iz naslednjih komponent:

- povezava med aplikacijskimi programi in ostalim delom ACF/VTAM-a (API):

v ta del ACF/VTAM-a pride aplikacijski program direktno z instrukcijo vejanja, torej se izvaja v tem primeru ACF/VTAM kot del uporabnikovega taska iz druge particije.Ta del ACF/VTAM-a analizira zahteve aplikacijskega programa in preko prekinitve preda kontrolo VTAM-ovemu upravljalcu s procesi, obenem mu posreduje naslove kontrolnih blokov, kjer je specificirana zahteva aplikacijskega programa. Prekinitev je potrebna zato,da se od tu dalje koda v ACF/VTAM-ovi particiji izvaja kot poseben DOS/VSE task paralelno z DOS/VSE taskom aplikacijskega programa. DOS/VSE task v ACF/VTAM-ovi particiji mora imeti višjo prioriteto kot katerikoli DOS/VSE task, ki uporablja VTAM pristopno metodo. (da dobi prvi kontrolo ob prekinitvi, ki jo sproži API).

- upravljalec (dispečer) ACF/VTAM-ovih procesov (PSS - Process Scheduling Services): VTAM-ov proces je zaporedje operacij, ki so potrebne za zadostitev neke zahteve.Naslovi procedur, ki izvršujejo konkretne operacije so za posamezen VTAM-ov proces shranjeni v posebnem vektorju (del navideznega pomnilnika), ki določa potek VTAM-ovega procesa.Upravljalec VTAM-ovih procesov lahko proži VTAM-ov proces ali uporabnikov exit(program v DOS/VSE tasku, ki uporablja

ACF/VTAM pristopno metodo). Upravljajalec z VTAM-ovimi procesi določa kateremu procesu bo predal kontrolo na osnovi dinamične prioritete procesov: najvišje so procesi, ki so bili prekinjeni zaradi izvršitve neke V/I zahteve, pa je bila ta zahteva izvršena, nato VTAM-ovi procesi, ki čakajo na izvajanje, nazadnje pa so uporabnikovi exiti.
- kontrolni sloj (Control Layer)- samo pri ne-SNA enotah : pregleda zahtevo,dobljeno od API, ustrezno formatizira podatke in tvori kanalni program ter preda kontrolo komunikacijskemu podsistemu. Kadar dobi kontrolo od komunikacijskega podsistema,dobljene podatke ustrezno obdela. če aplikacijski program dobljenih podatkov ni prej zahteval, jih shrani do ustrezne zahteve.

- upravljalec z vozlišči v mreži (Network Manager) - za ne SNA enote : je vmesni sloj med API in TPIOS, ki služi za povezavo med aplikacijskimi programi in terminali

- komunikacijski podsistem (TPIOS - Telecommunication Process InputOutput Services)- za ne SNA enote :VTAM-ov proces (kontrolni sloj) zahteva komuniciranje z neko fizično enoto preko komunikacijskega podsistema. Ta zahtevo obdela, jo shrani v vrsto in po vrsti preda TSC ju - prenosni podsistem(TSC): je vmesnik med operacijskim sistemom (DOS/VSE) in ACF/VTAM, za SNA enote pa vrši tudi funkcije TPIOS, upravljalca z vozlišči in kontrolnega sloja.

5.CICS/VS

CICS/VS (Customer Information Control System) je transakcijsko orientiran kontrolni program, ki deluje kot aplikacijski program v eni particiji DOS/VSE operacijskega sistema, za SNA računalniško mrežo je CICS/VS logična enota-aplikacijski program pod ACF/VTAM pristopno metodo.Ker je transakcijsko orientiran CICS/VS skuša hraniti čimveč podatkov potrebnih za delovanje sistema v tabelah v navideznem pomnilniku, te tabele pa so dostopne vsem procesom, ki se izvajajo pod CICS/VS kontrolnim programom. Da bi lahko CICS/VS v zelo kratkem času izvršil veliko število poslov (transakcij) uporablja svojo lastno obliko paralelnega procesiranja.CICS/VS taski so procesi, ki jih tvori in nadzira CICS/VS kontrolni program. Kontrolni program tvori CICS/VS taske in zanje vodi celotno upravljanje s podatkovnimi bazami, dodeljevanje pomnilnika,komuniciranje z drugimi logičnimi enotami, sinhronizira delo med CICS/VS taski in dodeljuje procesor CICS/VS taskom ob kličih kontrolnega programa (navidezne prekinitve).DOS/VSE operacijski sistem paralelnega procesiranja pod CICS/VS om ne zazna,zanj so vsi CICS/VS taski en sam DOS/VSE task. CICS/VS taski se prožijo na zahtevo drugih logičnih enot v SNA mreži (terminali, aplikacijski programi). Vsak CICS/VS task se izvaja kjerkoli v CICS/VS particiji in ne tvori kakega ločenega dela naščovnega prostora v particiji (vsak CICS/VS task ima dostop za čitanje in pisanje do celotne CICS/VS particije torej tudi do CICS/VS kontrolnih blokov in programov - to je verjetno ena od bistvenih pomanklijivosti CICS/VS kontrolnega programa (nestabilnost sistema na račun izboljšanih zmogljivosti - vsak task lahko podre celotni sistem).

CICS/VS na zahtevo zdruge logične enote ali na zahtevo CICS/VS taska proži drugi CICS/VS task, tako da najprej, če ustreznega programa še ni v navideznem pomnilniku preskrbi programu ustrezen prostor v pomnilniku, ga vloži iz knjižnice programov v pomnilnik, preskrbi programu potreben dinamični pomnilnik za podatkovne strukture, tvori za CICS/VS task ustrezne kontrolne bloke in preda tako pripravljenemu programu kontrolo.Uporabnikovi programi pod CICS/VS-om morajo biti 'ponovno uporabljivi', to pomeni da se lahko koda programa spreminja samo med dvema zaporednima klicema CICS/VS kontrolnega programa. To omogoča da se lahko ista koda uporablja v več CICS/VS taskih paralelno, pa tudi to, da programa, ki je že v navideznem pomnilniku ni potrebno več vlagati iz knjižnice v pomnilnik.

CICS/VS taski morajo obvezno uporabljati CICS/VS upravljanje s podatkovnimi bazami in CICS/VS upravljanje s pomnilnikom, vendar CICS/VS kontrolni program ne more zaznati uporabe DOS/VSE funkcij v te namene (ker CICS/VS ne upravlja z dejanskimi prekinitvami).

CICS/VS upravljanje z navideznim pomnilnikom dodeljuje pomnilnik v kontinuiranih delih npr.:programu,dinamičnemu pomnilniku, kontrolnim blokom,... . Pomnilnik dodeljuje na osnovi metode najboljšega prilagajanja (dodeli najmanjši prosti kontinuiran del navideznega pomnilnika, ki še zadovoljuje zahtevo. Tak način dodeljevanja pomnilnika je pri transakcijsko orientiranem operacijskem sistemu uporaben, ker transakcije po pravilu potrebujejo le majhen del pomnilnika in to le za kratek čas.

CICS/VS dodeljuje CICS/VS taskom procesor na osnovi prioritet posameznih CICS/VS taskov,prioritet posameznih logičnih enot, ki so sprožile zahtevo in prioritet operaterja-programa na logični enoti.Procesor se dodeljuje za čas med dvema zaporednima navideznima prekinitvama (vmes je lahko poljubno število dejanskih prekinitev).število paralelno delujočih CICS/VS taskov ni omejeno, določeno je pri startu CICS/VS kontrolnega programa, lahko se to število dinamično spreminja med izvajanjem CICS/VS-a.
CICS/VS taski med seboj lahko sodelujejo. Vsak task lahko proži drug CICS/VS task in svoje delovanje z njim tudi sinhrinizira.

6.ACF/NCP

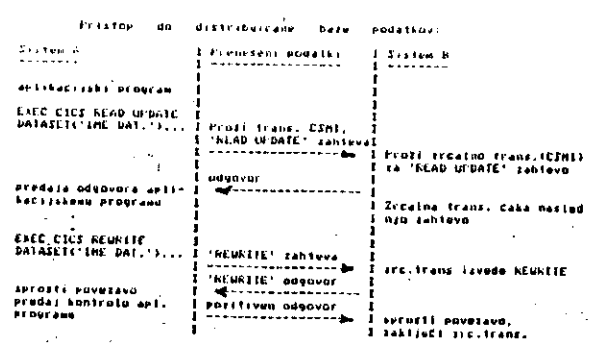
ACF/NCP (Advanced Communication Functions/Network Control program) je kontrolni program, ki teče v čelnem računalniku tipa 3705. Upravlja SNA računalniško mrežo. V to mrežo se vklapija kot fizična enota tip 4. Deluje v povezavi z glavnim računalnikom in VTAM pristopno metodo. čelni računalnik je z glavnim povezan kanalsko ali preko drugega čelnega računalnika. čelni računalnik 3705 sestavljajo: glavni procesor, kanalski priključek (na kanal glavnega računalnika), pomnilnik in pregledovatec linij. Vse tri ostale komponente imajo pristop do istega pomnilnika.
Od aplikacijskega programa VTAM prevzame zahtevo in podatke za prenos do neke logične enote. Te podatke preoblikovane shrani v svoje V/I vmesnike in s prekinitvijo signalizira DOS/VSE kontrolnemu programu zahtevo za prenos podatkov ACF/NCP-ju. DOS/VSE asinhrono štarta ustrezen kanalski program. Kanalski program s prekinitvijo signalizira ACF/NCP procesu začetek prenosa nakar kanal glavnega računalnika in kanalski priključek 3705

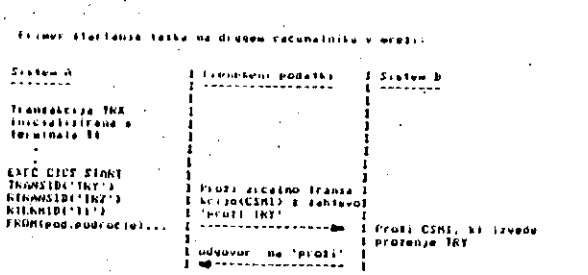
preneseta podatke iz ACF/VTAM-ovih vmesnikov (pomnilnik glavnega računalnika) v ACF/NCP-jeve vmesnike(pomnilnik 3705). Prenos se vrši asinhrono tako napram glavnemu računalniku (kanal vrši prenos) kakor napram 3705 (kanalski priključek vrši prenos). Prenos podatkov v obratni smeri vedno začne glavni računalnik s postavitvijo zahteve; prenos se lahko začne po vsakem prenosu iz glavnega računalnika v 3705 ali tako, da čelni računalnik preko kanala sporoči, da ima na voljo podatke za prenos. ACF/NCP komunicira z drugo fizično enoto v računalniški mreži preko pregledovalca linij. Pregledovalec linij stalno komunicira z fizičnimi enotami na linijah in ob tem prenaša v eni in drugi smeri podatke, ki jih najde v svojih registrih in s prekinitvijo signalizira izvršitev prenosa podatkov NCP-ju.

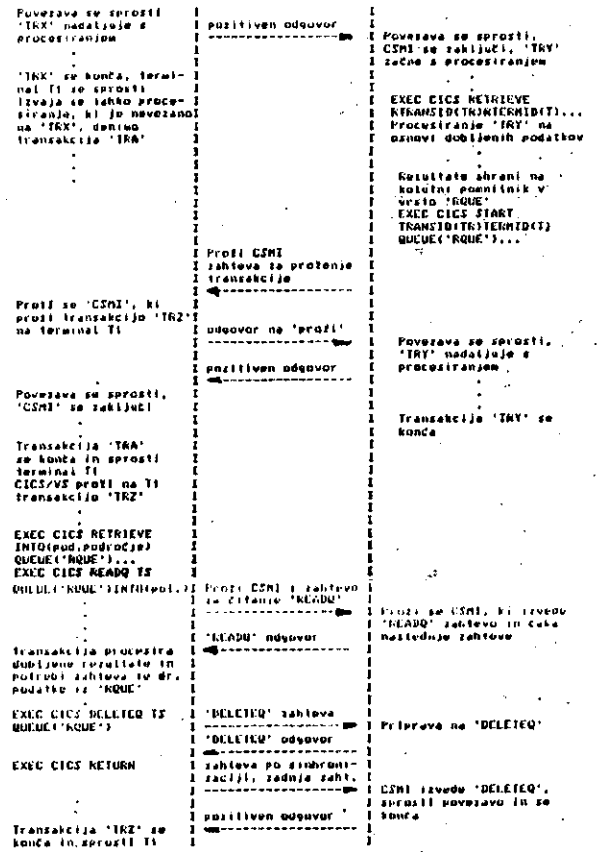
7. CICS ISC - KOMPONENTA CICS/VS-a ZA KOMUNICIRANJE S CICS/VS-om NA DRUGEM RAČUNALNIKU

CICS ISC (Inter System Communications) je komponenta CICS/VS-a, ki omogoča komuniciranje z aplikacijskim programom na drugem računalniku (s CICS/VS-om ali IMS/VS-om). Omogoča tako paralelno procesiranje ene zahteve na dveh ali več računalnikih v mreži, da posamezen aplikacijski program pod CICS/VS-om ne ve, na katerem računalniku se izvajajo taski (procesi) s katerimi sodeluje.OmogočA tudi pristop do distribuiranih podatkovnih baz na z uporabnikovega gledišča popolnoma enak način, kot če te baze podatkov sploh ne bi bile distribuirane-uporabnikov aplikacijski program sploh ne ve, da neka podatkovna baza ni na njegovem računalniku, ampak nekje drugje v mreži. To je lep prikaz dejstva, da je računalniška mreža tudi neka posebna vrsta multiprocesorskega sistema.

Primeri:







8. ZAKLJUČEK

Opisani primer paralelnega procesiranja v računalniški mreži ponazarja trditev, da je računalniška mreža v bistvu mrežni stroj - multiprocesorski sistem. Ta oblika paralelnega procesiranja se hitro razvija, spreminja in dograjuje. Zato pomeni opisani primer le trenutno stanje na tem področju pri IBM.

9. SLOVARČEK

CICS/VS - TP monitor
CICS-ISC - dodatek CICS/VS, ki omogoča povezavo z drugim CICS/VS
ACF/VTAM - mrežna pristopna metoda
SNA IBM koncept arhitekture računalniške mreže
DOS/VSE - operacijski sistem
ACF/NCP - kontrolni program za čelni računalnik
SDLC,BSC - linijski protokoli
task - sekvenčni proces
transakcija - CICS/VS task z vsem potrebnim sodelovanjem operaterja na terminalu
kanal - V/I procesor
CCW - ukaz kanalu
PSS - dispečer VTAM procesov
API - povezava aplikacijski program - VTAM
TSC - prenosni podsistem pri VTAM
SMS - upravljalec pomnilnika pri VTAM
logična enota - naslovljiv program v računalniški mreži
PIU - paket podatkov, ki se prenaša po mreži

INFORMATICA 1/1981

JEZIK PL/I IN MIKRORAČUNALNIKI II

ANTON P. ŽELEZNIKAR

UDK: 681.3.06 PL 1:181.4

SOZD ELEKTROTEHNA, DO DELTA

Članek opisuje v svojem drugem delu nadaljne lastnosti programirnega jezika PL/I-80, in sicer predvsem V/I sistem prevajalnika, prinaša pa tudi vrsto primerov, ki kažejo učinke uporabe raznovrstnih prevajalniških stikal pri generiranju zbirk in njihovih oblik. Ti primeri tudi najbolje prikazujejo nekatere lastnosti prevajalnika. Glede na vrsto zanimivih primerov PL/I programov s področja poslovnih in znanstvenih obdelav, bodo nadaljni primeri prikazani še v tretjem delu članka.

PL/I Language and Microcomputers II. This article (second part) describes some further properties of PL/I-80 dealing with I/O system of the language (and compiler) and with several examples showing the effects of various compiler switches usage when different file shapes are generated. In some way, these examples illustrate the main characteristics of the compiler. According to a large set of interestig examples of PL/I programming in the area of commercial and scientific application, further examples will be demonstrated in the third part (to appear) of the article.

5. Delovanje prevajalnega sistema

V prejšnjem delu članka smo opisali le nekatere sestavine jezika PL/I, zato bomo v nadaljevanju poskusili v primerih uporabiti tudi nekatere manjkajoče stavke.

Prevajalnik za jezik PL/I-80 je obsežen paket in ga sestavljajo tile moduli:

PLI.COM	začetni del prevajalnika
PLIØ-OVL	prvi prekrivni del
PLI1.OVL	drugi prekrivni del
PL2.OVL	tretji prekrivni del
LINK.COMP	povezovalni urejevalnik
PLILIB.IRL	knjižnica s premestljivimi rutinami
LIB.COM	knjižničar
RMAC	premeščevalni makrozbirnik

Pomen posameznih pripon pa je tale:

COM	ukazna zbirka sistema CP/M
OVL	prekritje prevajalnika jezika PL/I-80 (glej spredaj)
IRL	indeksno premestljivi kod
PLI	izvirni programi v jeziku PL/I-80
PRN	tiskalniška zbirka na disku (liste za tiskalnik na disku)
REL	premestljivi objektni kod
DAT	podatkovna zbirka

Prevajalnik za PL/I-80 potrebuje vsaj 48 k CP/M sistem ter je troprehoden. Če imamo npr. izvirni program (zbirko) IKE.PLI, ga prevedemo z direktivo

```
PLI IKE.PLI
```

Tedaj se pojavi zbirka (prevod)

```
IKE.REL
```

ki vsebuje premestljivi strojni kod programa IKE. Ta kod še ni izvršljiv in zbirke tipa REL je treba še povezati s subrutinsko knjižnico, ko uporabimo direktivo

```
LINK IKE.REL
```

Pri tem generira program LINK-80 zbirko tipa COM, torej izvršljivo zbirko

```
IKE.COM
```

na disketi, ki jo lahko uporabimo.

Prevajalnik PL/I-80 poženemo tako z ukazom

```
PLI ime_programa $ s1 ...s7
```

kjer je ime_programa ime zbirke za prevajanje, s1 ... s7 pa je seznam z največ sedmimi stikali, ki vključujejo prevajalniške opcije. Za izvedbo tega ukaza (direktive PLI) mora disketa vsebovati module PLI.COM, PLIØ.OVL, PLI1.OVL in PLI2.OVL.

Prevajalniške opcije (stikala) pa so:

B: pokaže subrutine knjižnice, ki so klicane iz uporabniškega programa;
D: pošlje tiskalniško zbirko na disk (namesto na konzolo);

I: v tretjem prehodu se lista zaporedje strojnih kodov, ki pripadajo ustreznemu PL/I stavku; stikalo I avtomatično vključi stikalo L;
K: ukine listanje parametrov in INCLUDE stavkov v prvem prehodu;
L: vključi listanje izvirnega programa;
N: oblikuje prikaz vgnezdenja posameznih blokov v prvem prehodu;
O: ukine generiranje objektne zbirke ime_programa.REL;
P: omogoči tiskanje na napravah za listanje;
S: oblikuje se simbolni seznam s pridevki v prvem prehodu.

Premeščevalni urejevalnik poveže premestljive zbirke, ki sta jih generirala prevajalnik ali modul RMAC ter vključi subrutine iz zbirke PLILIB.IRL. Navadna oblika tega ukaza je

LINK ime_programa

ki poveže zbirko ime_programa.REL, dobljeno s prevajalnikom s subrutinami iz PLIUB.IRL ter generira zbirko ime_programa.COM, ki je izvršljivi strojni kod procesorja 8080A. Če imamo več ločeno prevedenih modulov MØ, M1, ..., Mn tipa REL, jih povežemo z ukazom

LINK MØ, M1, ... , Mn

ko se oblikuje nova zbirka MØ.COM. Podoben učinek ima ukaz

LINK M = MØ, M1, ... , Mn

ko se oblikuje zbirka M.COM. Stikala ukaza LINK sledijo zbirčnim imenom, so zaprta v oglate oklepaje in ločena z vejicami. Teh stikal je 15.

Pri danem uporabniškem programu

ime_programa.PLI

se tedaj z zaporedno uporabo ukazov PLI in LINK ter z izbiro stikal oblikujejo še nadaljne zbirke, in sicer:

(1) ime_programa.REL z ukazom PLI;
(2) ime_programa.PRN, če je bilo izbrano stikalo D v ukazu PLI;
(3) ime_programa.SYM, če je bilo izbrano stikalo S v ukazu PLI;
(4) ime_programa.COM po uporabi ukaza LINK.

V nadaljnem si bomo ogledali nekaj značilnih primerov programov v jeziku PL/I-80 ter generirane zbirke prevajalnika.

6. V/I sistem za PL/I-80 prevajalnik

Že v članku (18) smo pregledno opisali možnosti vhoda/izhoda (V/I) v jeziku PL/I-80. V/I je osnovnega pomena za oblikovanje in uporabo zbirk na diskih ter za komuniciranje z ostalo periferijo (konzola, tiskalniki, teleprinterji itn.), zato si oglejmo njegove možnosti nekoliko bolj podrobno.

Sistem PL/I-80 omogoča glede na fizične enote neodvisen V/I, ki povezuje programe s CP/M zbirčnim sistemom. Parametri za takšno povezavo se določijo v OPEN stavku ter z mehanizmi GET, PUT, READ in WRITE stavkov.

6.1. OPEN stavek

OPEN stavek uporabimo s prosto izbiro, vendar se pojavi avtomatično pri dostopu v zbirko v GET, PUT, READ in WRITE stavku, ko se sam OPEN stavek ni posebej pojavil. Oblika OPEN stavka je tale:

```
OPEN FILE (z)
    STREAM RECORD
    PRINT
    INPUT OUTPUT UPDATE
    SEQUENTIAL DIRECT
    KEYED
    ENV (B (i) ) ENV (F(i)) ENV (F(i)), B (j))
    LINESIZE (i)
    PAGESIZE (i)
    TITLE (c);
```

kjer se pridevki lahko uporabijo v poljubnem zaporedju. Vrednost z označuje zbirčno konstanto ali spremenljivko, ki ima v OPEN stavku določeno ime. Vsi drugi pridevki so izbirni, vrednosti i in j pa označujeta izraze tipa FIXED BINARY. Vrednost c je znakovni izraz. Pridevki v isti vrsti so v nasprotju in če jih ne uporabimo, velja prvi. Zadnji štirje pridevki dobijo pri nevključitvi avtomatično tele vrednosti:

```
ENV (B(128))
LINESIZE (8Ø)
PAGESIZE (6Ø)
TITLE ('f.DAT')
```

STREAM zbirka vsebuje ASCII podatke spremenljive dolžine, medtem ko ima RECORD zbirka vobče čiste binarne podatke. Vrstice ASCII zbirke so določene z zaporedji parov pomik valja - pomik vrstice. PRINT pridevek se nanaša samo na STREAM zbirke.

INPUT zbirke pričakujemo v točki OPEN stavka, dočim se OUTPUT zbirke zbrišejo (če obstajajo) in oblikujejo pri OPEN stavku. UPDATE zbirka ne more imeti STREAM pridevka, lahko pa je pisana in brana. UPDATE zbirka se oblikuje, če ne obstaja.

SEQUENTIAL zbirke se berejo ali pišejo od začetka do konca, v DIRECT zbirke pa lahko vstopamo naključno. DIRECT zbirka dobi avtomatično RECORD pridevek.

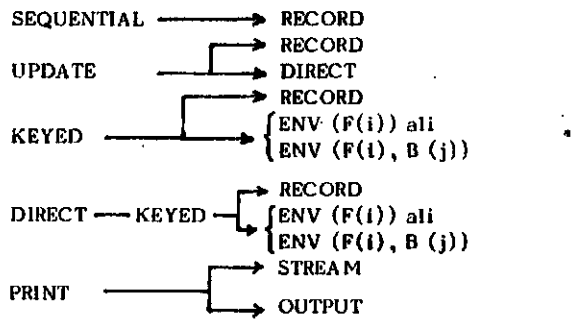
Dostop v KEYED zbirke je omogočen z uporabo ključev in RECORD pridevek se priredi avtomatično. KEYED zbirka je enostavno zbirka zapisov, ki imajo fiksno dolžino. Ključ je relativni položaj zapisa.

ENV pridevek določa zbirke z zapisi fiksne in spremenljive dolžine. Oblika ENV (B(i)) povzroči, da V/I sistem rezervira i zlogov pomnilnika, kjer se i zaokroži na naslednji mnogokratnik 128 zlogov. V tem primeru ima zbirka zapise spremenljivih dolžin in tako ne more imeti KEYED pridevka.

ENV (F(i)) oblika določa zbirko z zapisi fiksne dolžine, t.j. z i zlogi v vsakem zapisu, kjer se i zaokroži na naslednji mnogokratnik 128 zlogov. V tem primeru uporabimo KEYED pridevek.

ENV (F(i), B (j)) določa zbirko, ki vsebuje zapise fiksne dolžine i zlogov (i je zaokrožen navzgor). Uporabi se KEYED pridevek.

Pri uporabi KEYED pridevka se mora navesti dolžina zapisa in sicer z ENV (F(i)) ali z ENV (F(i), B (j)). Vse UPDATE zbirke morajo biti določene z DIRECT pridevkom, tako da je mogoče locirati posamezne zapise. Imamo tole shemo pridevkov:



Kot vidimo, se RECORD pridevek doda k SEQUENTIAL, UPDATE in KEYED zbirkam, dočim se STREAM pridevek doda k PRINT zbirkam. PRINT zbirkam se avtomatično doda OUTPUT pridevek. KEYED pridevek se doda DIRECT zbirkam, doda pa se še RECORD pridevek.

LINESIZE pridevek se nanaša samo na STREAM zbirke in določa največjo dolžino vhodne ali izhodne vrstice. PAGESIZE pridevek je povezan s STREAM OUTPUT zbirkami in določa dolžino strani.

TITLE (c) pridevek omogoča programsko povezavo med notranjim imenom zbirke in zunanjo napravo ali CP/M zbirko. Če ne navedemo imena zunanje zbirke, se priredi vrednost navedbe zbirke tipa DAT. Sicer pa se znakovni niz c izračuna v neko ime naprave: $CON (konzola), $LST (naprava za listanje), $RDR (naprava za branje) ali $PUN (naprava za luknjanje) ali v ime zbirke na disku:

d : x . y

Tu sta x in y lahko tudi $1 ali $2. Pri $1 se vstavi prvo tekoče ime iz ukazne vrstice na naslovno (TITLE) pozicijo, pri $2 pa drugo tekoče ime na ustrezno pozicijo. Ime zbirke x ne more biti prazno, x, y in d pa ne smejo vsebovati znaka '?'. Fizične V/I naprave, kot so $CON, $RDR, $PUN in $LST, se lahko odprejo kot STREAM zbirke, kjer ima $RDR pridevek INPUT, $PUN in $LST pa imata pridevek OUTPUT.

6.2. CLOSE stavek

Zaprtje zbirke dosežemo s stavkom

```
CLOSE.FILE (f);
```

kjer je f zbirčna spremenljivka ali konstanta. Vse odprte zbirke se avtomatično zaprejo na koncu programa ali pri izvedbi STOP stavka.

Zbirke, ki so bile odprte s STREAM pridevkom, so dostopne prek GET in PUT stavka, dočim so zbirke z RECORD pridevkom dostopne prek READ in WRITE stavka.

6.3. PUT LIST stavek

PUT LIST stavek ima obliko

```
PUT FILE (f)
    SKIP  (i)
    PAGE
    LIST (d);
```

kjer so elementi izbirni, toda vsaj eden mora biti naveden. Vrstni red pridevkov je poljuben, toda LIST pridevek mora biti zadnji. Tu je i celoštevilski izraz in d podatkovni seznam. PUT stavek avtomatično odpre SYSPRINT zbirko, če zbirka ni bila določena, in sicer implicitno z

```
OPEN FILE (SYSPRINT)
    PRINT ENV (B (128))
    TITLE ('$CON');
```

Podatkovni seznam d v LIST opciji ima splošno obliko

LIST (d1, d2, ..., dn)

kjer je di konstanta, skalarni izraz ali iterativna skupina; ta skupina je

(e1, e2, ..., em DO iteracija)

kjer so e1, ..., em konstante, skalarni izrazi ali iterativne skupine. T.i. "iteracija" ima obliko glave DO - skupine ter krmili ponavljanje zadevne skupine. Ekvivalent DO iteracije je tedaj

```
DO iteracija;
PUT LIST (e1, e2, ..., em);
END;
```

6.4. GET LIST stavek

PUT stavku je podoben GET stavek, ko imamo

```
GET
    FILE (f)
    SKIP  (i)
    LIST (d);
```

kjer so FILE, SKIP in LIST opcije, ki zadoščajo pogojem PUT stavka. GET stavek se uporablja za branje STREAM zbirk. Če FILE pridevek ni vključen, imamo avtomatičen OPEN stavek in je

```
OPEN FILE (SYSIN)
    STREAM ENV (b(128))
    TITLE ('$CON');
```

Pri konzolnem vstopu skozi GET stavek čaka V/I sistem na vhod iz konzole in uporabnik lahko vtipka do 80 znakov pred znakom za pomik valja. Podatkovni kosi (celote) so ločeni s presledkom ali vejico.

6.5. PUT EDIT stavek

PUT EDIT stavek je podoben PUT LIST stavku in imamo

```
PUT
    FILE (f)
    PAGE
    SKIP  (i)
    EDIT (d)  (f1);
```

kjer je f1 formatni seznam. Vsak element podatkovnega seznama d ima v f1 ustrezni formatni del, ki določa format podatkovnega elementa. Formatni elementi v f1 so lahko tile:

A	Izpiše se alfanumerično polje znakovnega podatka (njegove dolžine).
A (n)	Podobno A formatu, vendar z dolžino polja n, z zapolnitvijo presledkov na desni strani.
B	Izpiše se bitni niz, dolžina je določena z natančnostjo podatka.
B (n)	Podobno B formatu, z dolžino polja n, z zapolnitvijo presledkov na desni strani.

B1 Ekvivalentno B formatu.

B1 (n) Ekvivalentno B (n) formatu.

B2 Ekvivalentno B formatu, le da se številke pišejo s štiriško bazo (0, 1, 2, 3).

B2 (n) Ekvivalentno B (n) formatu, le da je baza štiriška.

B3 Ekvivalentno B formatu, vendar z osmiško bazo (številke od 0 do 7).

B3 (n) Ekvivalentno B (n) formatu, le da se tiskajo osmiška števila.

B4 Ekvivalentno B formatu, vendar z bazo 16 (številke 0, 1, ..., F).

B4 (n) Ekvivalentno B (n) formatu, z bazo 16.

COLUMN (n) Pomik na položaj stolpca n.

E (n) Znanstvena notacija v polju n znakov (n ⩾ 6).

E (n,m) Zapiše podatek v polje n znakov, z natančnostjo m decimalnih mest. Notacija je znanstvena.

F (n) Zapiše n številk brez ulomljenega dela, z zaokrožitvijo.

F (n,m) Zapiše n številk z m ulomljenimi številkami, z zaokrožitvijo.

LINE (n) Pomik na vrstico n.

PAGE Izda stran.

R (fmt) Določa oddaljeni format. R format je edini format v f1.

SKIP Pomik v naslednjo vrstico.

SKIP(n) Pomik za n vrstic.

TAB (n) Pomik na n-ti tab položaj, kjer so tab 8-stolpni mnogokratniki.

X (n) Vstavitev n presledkov.

Ker PUT EDIT stavka večkrat ne moremo izpisati v eni vrstici, ga lahko delimo, kot kaže tale primer:

```
PUT FILE (f) EDIT ('naslednji..', vrednost)(a,f(4));
PUT EDIT ((a(i) DO i = q To r)) (PAGE,40(3e(10,2),x(3)));
PUT EDIT (n, v, w) (r (fmt2));
```

6.6. GET EDIT stavek

GET EDIT stavek je podoben GET LIST stavku, podatki pa se berejo iz določenih polj vhodnega toka. GET LIST stavek je primeren za konzolni vhod, GET EDIT stavek pa za branje podatkov, ki so bili zapisani z drugim programom. Oblika GET EDIT stavka je

```
GET
    FILE (f)
    SKIP  (i)
    EDIT (d)  (f1);
```

Formatni seznam f1 lahko vsebuje tele elemente:

A Čitaj alfanumerično polje do pomika valja, pomika vrstice ali do konca zbirke (ni PL/I standard).

A (n) Včitaj naslednjih n znakov.

B (n) Včitaj naslednjih n znakov v obliki bitnega niza.

B1 (n) Pomen je enak kot pri B (n).

B2 (n) Podobno kot pri B1 (n), le da je baza štiriška (0, 1, 2, 3).

B3 (n) Podobno kot pri B1 (n), le da je baza osmiška (0, 1, 2, 3, 4, 5, 6, 7).

B4 (n) Podobno kot pri B1 (n), le da je baza šestnajstiška (0, 1, ..., F).

COLUMN(n) Pomik na stolpec n za vhod.

E (n) Včitaj naslednja n polja kot numerično vrednost (konstanta, celo število, število z ulomljenim delom in v znanstveni notaciji).

E (n,m) Ekvivalentno E (n), kjer se faktor m pri vhodu ne upošteva.

F (n) Ekvivalentno E (n).

F (n,m) Ekvivalentno E (n), toda z decimalno vejico, ki je za m mest pomaknjena v levo od najvišjega mesta, če v polju ni decimalne vejice.

LINE (n) Pomik na vrstico n pred branjem.

R (fmt) Določa daljinski format.

SKIP Zbriše trenutno vhodno vrstico pred branjem nadaljnih podatkovnih delov.

SKIP(n) Zbriše trenutno vhodno vrstico ter opravi pomik za n-1 vrstic pred branjem.

Pomik valja in vrstice (CR, LF) se ne upoštevata v formatih A (n), B (n), B1 (n), B2 (n), B3 (n), E (n), E (n, m), F (n) in F (n, m): včita se naslednja vrstica, kot ostanek znakov podatkovnega polja.

6.7. FORMAT stavek

Format stavek omogoča, da se seznam formatnih delov uporablja med različnimi GET EDIT in PUT EDIT stavki. Imamo formatno_ime:

```
FORMAT (f1);
```

kjer f1 označuje seznam formatnih delov. Seznam formatnih delov se potem navaja z uporabo R formata v okviru formatnih seznamov GET in PUT stavka. Npr.:

```
form1:
    FORMAT (5(X(3),4(B1(2),X(1),F(4)),
                    SKIP), SKIP (2));
    GET FILE (uslužbenec) (ure, os_dohodek)
                    (R (form1));
```

6.8. WRITE stavek

WRITE stavek se uporablja predvsem za prenos podatkov med pomnilnikom in zunanjo zbirko brez konverzije znakovnih oblik. Imamo

```
WRITE FILE (f)
      FROM (x);
```

kjer morata biti prisotna pridevka FILE in FROM, f je navedba zbirke in x je skalar ali povezan podatkovni tip. Zbirka f se pri tem odpre avtomatično z

```
OPEN FILE (f) OUTPUT SEQUENTIAL
        TITLE ('f.DAT') ENV (B(128));
```

Če je zbirka že odprta, se morajo pridevki ujemati s prejšnjim odprtjem. Tako je KEYED zbirka dovoljena (zapisi fiksne dolžine), DIRECT pa ne.

Druga oblika WRITE stavka je

```
WRITE FILE (f)
      FROM (x)
      KEYFROM (k);
```

kjer je vrstni red elementov FILE, FROM in KEYFROM poljuben. Če zbirka f še ni odprta, se odpre implicitno s stavkom

```
OPEN FILE (f) OUTPUT DIRECT ENV (F(128));
```

pred vstopom v zbirko f. DIRECT pridevek povzroči KEYED zbirko (in ta še RECORD zbirko).

KEYED pridevek povzroči dostop do zapisa s ključem k, kjer je k FIXED izraz in njegova vrednost je relativen naslov zapisa.

Posebna oblika WRITE stavka rabi za obdelavo STREAM podatkov spremenljive dolžine, če so omejeni s CR/LF zaporedjem. Pri dani zbirki f s pridevkoma STREAM in OUTPUT ter s spremenljivim znakovnim nizom "v" zapiše stavek

```
WRITE FROM (v);
```

nizno vrednost v na izhodno napravo, kot da imamo

```
WRITE FILE (SYSPRINT) FROM (v);
```

Kontrolni ASCII znaki v okviru nizne konstante so sestavljeni iz dveh znakov: "^" in ustrezne črke (CR je "^m"), dočim imamo za znak "^" zaporedje "^^".

Naj bo f zbirka, x skalar ali povezano sestavljen podatek, v spremenljivi znakovni niz in k izraz tipa FIXED BINARY. Imamo tele različne primere s pridevki:

```
WRITE FILE (f) FROM (x);
    SEQUENTIAL OUTPUT   KEYED   RECORD

WRITE FILE (f) FROM (x) KEYFROM (k);
    DIRECT OUTPUT   DIRECT UPDATE

WRITE FILE (f) FROM (v);
    STREAM OUTPUT

WRITE FROM (v);
    STREAM OUTPUT (avtomatično SYSPRINT)
```

6.9. READ stavek

READ stavek se uporablja za branje zapisov (RECORD) fiksne ali spremenljive dolžine brez znakovne pretvorbe. Podatki se prenašajo iz zunanje zbirke v podatkovne elemente pomnilnika, kjer ima zunanja zbirka binarne podatke. Imamo

```
READ FILE (f)
     INTO (x);
```

kjer je x skalar ali povezan agregat (npr. struktura, polje ali enostavna spremenljivka). FILE in INTO pridevek morata biti prisotna. Če zbirka "f" še ni bila odprta, se odpre avtomatično, kot da velja

```
OPEN FILE (f) INPUT SEQUENTIAL
     TITLE ('f.DAT') ENV (B(128));
```

Če je zbirka že bila odprta, se morajo prejšnji pridevki ujemati s pridevki v poslednjem stavku.

Kadar je bila zbirka odprta s KEYED pridevkom, imajo zapisi fiksno dolžino, kot določa ENV (F(i)) pridevek. Sicer je zapisna dolžina spremenljiva in odvisna od podatka x v INTO delu. Kadar je zapisna dolžina i večja od dolžine elementa x, se preostali zlogi zapisa ne upoštevajo (ne vpišejo v x). Če pa je zapisna dolžina n manjša od dolžine elementa x, se samo n zlogov včita v x. To velja seveda le za KEYED zbirke. Kadar zbirka ni tipa KEYED, sta dolžini zapisa in x enaki.

Ključi določene zbirke se lahko izvlečejo, ko se zbirka bere zaporedno, in sicer z obliko

```
READ FILE (f)
     INTO (x)
     KEYTO (k);
```

Učinek tega stavka je podoben onemu prejšnjega READ stavka, le da se vrednost ključa za zapis shrani v spremenljivko k s pridevkoma FIXED in BINARY. Seveda mora imeti zbirka tip KEYED. Avtomatičen OPEN stavek, ki ustreza tej obliki READ stavka, je

```
OPEN FILE (f) INPUT KEYED
    TITLE ('f.DAT') ENV (F(128));
```

Pridevek KEYED mora biti vsebovan, DIRECT pa ne, ker KEYTO izvleče ključ, toda ne povzroči branje zapisa s tem ključem. Ta oblika READ stavka se navadno uporabi, ko se zbirka prvič bere zaporedno z namenom, da se določijo ključi za kasnejši direktni dostop pri branju, čitanju ali popravljanju (UPDATE) zapisov v zbirki.

Tretja oblika READ stavka določa branje zapisa, ki ima določen ključ in sicer:

```
READ FILE (f)
     INTO (x)
     KEY (k);
```

Pripadajoči, avtomatični OPEN stavek je (če zbirka še ni odprta):

```
OPEN FILE (f)  INPUT DIRECT
    ENV (F(128))  TITLE ('f.DAT');
```

Pri odprti zbirki se morajo prvotni pridevki ujemati s pridevki v zadnjem stavku, izjeme so dopustne le pri odprtju z UPDATE pridevkom. DIRECT pridevek predpostavlja zbirko tipa KEYED.

Učinek tega READ stavka je neposreden dostop do zapisa, ki ima vrednost ključa k. Ker je zbirka tipa KEYED, je dolžina zapisa fiksna, kot določa ENV (F(i)) pridevek.

S posebno obliko READ stavka lahko obdelamo zbirke spremenljive dolžine tipa STREAM INPUT, ko uporabimo

```
READ FILE (f) INTO (v);
```

in

```
READ INTO (v);
```

kjer je "v" spremenljivi znakovni niz in f zbirka ali znakovna naprava. Če se ne uporabi FILE (f), imamo zbirko SYSIN. Če zbirka f ni odprta, se odpre avtomatično, kot da imamo stavek

```
OPEN FILE (f) PRINT
    TITLE ('f.DAT') ENV (B(128));
```

Učinek tega READ stavka je branje iz zbirke f, dokler ni dosežena dolžina elementa "v" ali ko se včita LF znak. Pri SYSIN se včita maksimalno 80 znakov, preden se avtomatično izda CR in LF znak.

Če je f ime zbirke, x skalar ali navedba sestavljenega podatka, v spremenljivi znakovni niz in k fiksni binarni ključ, imamo tele oblike s pripadajočimi pridevki:

```
READ FILE (f) INTO (x);
    SEQUENTIAL INPUT KEYED RECORD

READ FILE (f) INTO (x) KEYTO (k);
    SEQUENTIAL INPUT KEYED RECORD

READ FILE (f) INTO (x) KEY (k);
    DIRECT INPUT DIRECT UPDATE

READ FILE (f) INTO (v);
    STREAM INPUT

READ INTO (v);
    STREAM INPUT (avtomatično SYSIN)
```

V nadaljni obravnavi si oglejmo nekaj primerov.

7. Preizkus delovanja prevajalnika

V tem poglavju si bomo na kratko ogledali delovanje prevajalnika z uporabo ukazov

PLI in LINK

ter PLI stikal (opcij)

B, D, I, K, L, N, O, P in S

Pri tej uporabi se bodo pojavile zbirke naslednjih tipov:

.PLI izvirna zbirka
.REL relativna, s PLI prevedena zbirka
.PRN zbirka za tiskalnik na disku, v kateri bo upoštevana uporaba PLI stikal
.COM prevedena ukazna zbirka za procesor 8080A, z začetkom na lokaciji 100H (začetek t.im. prehodnega programskega območja oz. uporabniškega prostora)
.SYM zbirka s simbolno tabelo sistemskih identifikatorjev (pri uporabi stikala S v PLI in LINK ukazu)

Vzemimo za preizkus delovanje prevajalnika pri dovolj enostavnem PLI programu ter si oglejmo vsebine dobljenih (generiranih) zbirk. Zbirke .PLI ne bomo pisali v posebni listi, ker se pojavi tudi v zbirki tipa .PRN, kjer je lahko v odvisnosti uporabe stikal še dodatno opremljena (npr. prikaz vgnezdenja posameznih programskih blokov). V listah 1, 2, 3 in 4 si oglejmo tako zbirke tipov .PRN, .COM in .SYM pri izhodiščni zbirki DFACT.PLI.

Kot kaže lista 1, imamo PLI program za izračun faktoriala pri uporabi rekurzivne procedure FACT (vrstica 10b liste 1). Na začetku liste 1 imamo spisek uporabljenih stikal (B, D, I, N, S), torej smo imeli ukaz

PLI DFACT,PLI $BDINS

ki nam je iz zbirke DFACT.PLI generiral zbirki DFACT. .PRN in DFACT.REL.

Stikalo B omogoči proizvodnjo programske liste (druge liste v listi 1), v katero so vpisane vgrajene (sistemske) subrutine, tako da imamo pregled nad obliko generiranega programa. Iz liste 1 je razvidno, da se v uporabniški program vključujejo sistemske subrutine

?START, ?SYSPR, ?SKPOP, ?SLCTS, ?PNCOP,
?QICOP, ?PNVOP, ?QDCOP, ?QIOOP,
?SVBLK, ?DLDOP, ?RSBLK, ?QI15D, ?DMUOP

ter rekurzivna (uporabniška) subrutina FACT. S primerjavo liste 1 in liste 2 dobimo korespondenco med sistemskimi subrutinami (s prefiksi "?"), ta korespondenca pa je razvidna tudi iz liste 4.

Uporaba stikala D (glej začetek liste 1) povzroči oblikovanje zbirke .PRN na disketi.

Stikalo I povzroči mešani tekst tipa PLI in zbirnega jezika (procesorja 8080A), kar je razvidno iz druge podliste (vrstice 1 do 21, naslovi 0 do 0A5H) v listi 1.

Stikalo N opremi vrstične številke 1 do 21 s črkami a,b,c v odvisnosti od vgnezdenja posameznih blokov, stikalo S pa povzroči generiranje simbolne zbirke.

V listi 3 imamo zbirni program, ki ustreza sistemski subrutini ?QDCOP. Ta lista kaže možnost identifikacije določene vgrajene subrutine.

Končno imamo v listi 4 prikazano še vsebino zbirke DFACT.SYM, ki se dopolni z uporabo stikala S v LINK ukazu, ko imamo

LINK DFACT [S]

V listi 4 so navedeni sistemski identifikatorji, vstopne in izstopne točke ter subrutine, tudi tiste, ki so vgnezdene v klicanih sistemskih subrutinah v listi 1.

Pri preizkusu prevajalnika navedimo še nekatere lastnosti LINK ukaza oziroma LINK programa. Kot že opisano, poveže LINK ukaz uporabniške module, poveže pa tudi sistemske subrutine ter sistemske vstopne in izstop-

```
0100 F  1BA0 ?FPBNX     19F7 ?START     027B ?SYSPR
03E6 ?SKPOP     1351 ?SLCTS     01AB ?PNCOP     111D ?QICOP
01CF ?PNVOP     1081 ?QDCOP     1973 ?QIOOP     13E9 ?SVBLK
11B5 ?DLDOP     1423 ?RSBLK     12E7 ?DCRET     103E ?QI15D
11F3 ?DMUOP     1BAF ?FILAT     1BB6 ?FPB       01A5 ?PNBOP
0485 ?PNCPR     13CB ?IS22N     0280 ?SIOOP     0277 ?SYSIN
029E ?SIOPR     1BE6 ?FPBST     1BF9 SYSIN      1C1E SYSPRI
056B ?OIOOP     072F ?FPBIO     058A ?OIOPR     1306 ?BSL16
1612 ?SIGNA     03EF ?SKPPR     0791 ?GNCPR     0C75 ?WRBYT
055D ?PAGOP     130C ?NSTOP     137A ?SMVCM     1317 ?SJSVM
1364 ?SSCFS     1026 ?QB08I     0B52 ?OPNFI     1C46 ?FMTS
19C7 ?FPBOU     197F ?FPBIN     0C62 ?RDBYT     0C9B ?RDBUF
0CBE ?WRBUF     0DA7 ?CLOSE     0DD8 ?GETKY     0DFE ?SETKY
0D8B ?PATH      0005 ?BDOS      005C ?DFCB0     006C ?DFCB1
0080 ?DBUFF     14BE ?ALLOP     1554 ?FREOP     1A85 ?ADDIO
1A9C ?SUBIO     19DD ?WRCHR     0F03 ?RFSIZ     0F75 ?RRFCB
0F7A ?RWFCB     1029 ?QB16I     103A ?QI07D     139D ?ID22
13C3 ?IN20N     11DC ?DNGOP     12D3 ?DOVER     1300 ?BSL08
1319 ?SJSCM     132B ?SJSTS     134F ?SLVTS     137E ?SMCCM
13D2 ?ZEROD     13C3 ?IN20      13CB ?IS22      1E57 ?RECLS
1C4E ?CONSP     1454 ?RECOV     149E ?OFCOP     1B5B ?ERMSG
1E55 ?BEGIN     1C6F ?ONCOD     1602 ?SIGOP     1B40 ?STOPX
1E4F ?STACK     1937 ?ONCPC     189A ?ONCOP     18EF ?REVOP
1C72 ?CNCOL     1E5A ?MEMRY     1E59 ?DFDRV     0000 ?BOOT
```

Lista 4. Ta lista prikazuje seznam sistemskih identifikatorjev za sistemske subrutine, v njih vgnezdene subrutine, vstopne in izstopne točke. Za naslovom je pripisan pripadajoči identifikator. Npr. subrutini ?QDCOP iz liste 3 pripada naslov 1081H itn.

```
A_TYPE B_BFACT.PRN


PL/I-80 V1.3  COMPILATION OF: DFACT

B: Built-In Subroutine Trace
D: Disk Print
I: Interlist Source and Code
N: Nesting Level Display
S: Symbol Table List

   1 a      f:
   2 b           proc options(main);
   3 b           dcl
   4 b               i fixed;
   5 c               do i = 0 repeat(i+1);
   6 c               put skip list('Factorial(',i,')=',fact(i));
   7 c               end;
   8 b           stop;
   9 b
  10 b           fact:
  11 c               proc (i)
  12 c                   returns(fixed dec(15,0)) recursive;
  13 c               dcl
  14 c                   i fixed;
  15 c               dcl
  16 c                   f fixed dec(15,0);
  17 c                   if i = 0 then
  18 c                       return (1);
  19 c                   return (decimal(i,15) * fact(i-1));
  20 c               end fact;
  21 b           end f;
  SYMBOLS:

BLOCK AT LINE 1, AUTO STORAGE 0 BYTES
a 0000 DECIMAL BUILTIN CONSTANT
a 0000 F ENTRY PARAMETERS(0) EXTERNAL CONSTANT
BLOCK AT LINE 3, AUTO STORAGE 2 BYTES
c 0000 I FIXED BINARY(15,0) AUTOMATIC
c 0000 FACT FUNCTION PARAMETERS(1) RECURSIVE CONSTANT
c 0000 .* FIXED DECIMAL(15,0) RETURNED
c 0008 .* FIXED BINARY(15,0) PARAMETER
BLOCK AT LINE 13, AUTO STORAGE 12 BYTES, PARAMETERS I
e 0000 I FIXED BINARY(15,0) PARAMETER
e 0002 I FIXED BINARY(15,0) AUTOMATIC
e 0004 F FIXED DECIMAL(15,0) AUTOMATIC
   NO ERROR(S) IN PASS 1

   NO ERROR(S) IN PASS 2


PL/I-80 V1.3  COMPILATION OF: DFACT

   1 a 0000 f:
       0000       LXI  B,0200
       0003       CALL ?START
   2 a 0006      proc options(main);
   3 c 0006      dcl
   4 c 0006          i fixed;
   5 c 0006          do i = 0 repeat(i+1);
       0006       LXI  H,0000
       0009       SHLD I A(0014)
   6 c 000C         put skip list('Factorial(',i,')=',fact(i));
       000C       LXI  D,0252
       000F       LXI  B,0000
       0012       CALL ?SYSPR
       0015       MVI  A,01
       0017       CALL ?SKPOP
       001A       LXI  H,* S(0000)
       001D       MVI  A,0A
       001F       CALL ?SLCTS
       0022       CALL ?PNCOP
       0025       LHLD I A(0014)
       0028       MVI  A,09
       002A       CALL ?GICOP
       002D       CALL ?PNVOP
       0030       LXI  H,* S(000A)
       0033       MVI  A,02
       0035       CALL ?SLCTS
       0038       CALL ?PNCOP
       003B       LXI  H,* A(0016)
       003E       CALL 0000
       0041       MVI  A,12
       0043       MVI  B,00
       0045       CALL ?GDCOP
       0048       CALL ?PNVOP
       004B       CALL ?GIDOP
       0010       ==== 004E
       004E       LHLD I A(0014)
       0051       INX  H
       0052       SHLD I A(0014)
       0055       JMP  000C
   7 c 0058         end;
   8 c 0058      stop;
   9 c 0058
  10 c 0058      fact:
  11 c 0058          proc (i)
  12 c 0058              returns(fixed dec(15,0)) recursive;
  13 e 0058          dcl
       0058       LXI  B,0010
       005B       LXI  D,0018
       005E       CALL ?SVBLK
       0061       MOV  E,M
       0062       INX  H
       0063       MOV  D,M
       0064       XCHG
       0065       SHLD I A(0018)
       0068       LHLD I A(0018)
       006B       MOV  E,M
       006C       INX  H
       006D       MOV  D,M
       006E       XCHG
       006F       SHLD I A(001A)
  14 e 0072           i fixed;
  15 e 0072          dcl
  16 e 0072              f fixed dec(15,0);
  17 e 0072              if i = 0 then
       0072       LXI  H,I A(001A)
       0075       MOV  A,M
       0076       INX  H
```

Lista 1. Pri prevajanju zbirke DFACT (izračun faktoriala z uporabo rekurzivne funkcije) so bila uporabljena stikala B,D,I,N,S (kot je razvidno na začetku liste). Ta lista vsebuje tako izvirni program s prikazom vgnezdenja (1a do 21b), simbolno tabelo, prevod v zbirnem jeziku s subrutinami prevajalniške knjižnice (?SYSPR itd.) ter podatke o obsegu prevoda.

```
0077  ORA  M
0078  JNZ  0000
        return (1);
18 e 007B  LXI  H,* S(000C)
007E  MVI  A,0F
0080  CALL ?DLDOP
0083  CALL ?RSBLK
0086  JMP  ?DCRET
0079  ==== 0089
        return (decimal(i,15) * fact(i-1));
19 e 0089  LHLD I A(001A)
008C  DCX  H
008D  SHLD * A(0026)
0090  LXI  H,* A(0024)
0093  CALL FACT
0096  LHLD I A(001A)
0099  CALL ?QI15D
009C  CALL ?DMUDP
009F  CALL ?RSBLK
00A2  JMP  ?DCRET
20 c 00A5    end fact;
21 a 00A5  end f1

CODE SIZE = 0045
DATA AREA = 002B
FREE SYMS = 02EB
END  COMPILATION
```

Lista 1. (Nadaljevanje s prejšnje strani)

```
0100  LXI  B,0200
0103  CALL 19F7
0106  LXI  H,0000
0109  SHLD 1C88
010C  LXI  D,0252
010F  LXI  B,014E
0112  CALL 027B
0115  MVI  A,01
0117  CALL 03E6
011A  LXI  H,1C74
011D  MVI  A,0A
011F  CALL 1351
0122  CALL 01AB
0125  LHLD 1C88
0128  MVI  A,09
012A  CALL 111D
012D  CALL 01CF
0130  LXI  H,1C7E
0133  MVI  A,02
0135  CALL 1351
0138  CALL 01AB
013B  LXI  H,1C8A
013E  CALL 0158
0141  MVI  A,12
0143  MVI  B,00
0145  CALL 1081
0148  CALL 01CF
014B  CALL 1973
014E  LHLD 1C88
0151  INX  H
0152  SHLD 1C88
0155  JMP  010C
0158  LXI  B,0010
015B  LXI  D,1C8C
015E  CALL 13E9
0161  MOV  E,M
0162  INX  H
0163  MOV  D,M
0164  XCHG
0165  SHLD 1C8C
0168  LHLD 1C8C
016B  MOV  E,M
016C  INX  H
016D  MOV  D,M
016E  XCHG
016F  SHLD 1C8E
0172  LXI  H,1C8E
0175  MOV  A,M
0176  INX  H
0177  ORA  M
0178  JNZ  0189
017B  LXI  H,1C80
017E  MVI  A,0F
0180  CALL 11B5
0183  CALL 1423
0186  JMP  12E7
0189  LHLD 1C8E
018C  DCX  H
018D  SHLD 1C9A
0190  LXI  H,1C98
0193  CALL 0158
0196  LHLD 1C8E
0199  CALL 103E
019C  CALL 11F3
019F  CALL 1423
01A2  JMP  12E7
```

Lista 2. Zbirni kod za PL/I program iz liste 1, ki je bil dobljen z uporabo obratnega zbirnika iz strojnega koda, generiranega iz REL zbirke z ukazom LINK. Začetek programa je na lokaciji 100H in kot vidimo, se ta program ujema z onim iz liste 1. Določljiva je tudi korespondenca med sistemskimi subrutinami (?START, ?SYSPR, ...) in njihovimi absolutnimi naslovi v dokončno prevedenemu programu.

```
1081  MOV  C,A
1082  LXI  H,0000
1085  DAD  SP
1086  LXI  D,FFEE
1089  XCHG
108A  DAD  D
108B  SPHL
108C  PUSH B
108D  MVI  B,0A
108F  LDAX D
1090  MOV  M,A
1091  INX  D
1092  INX  H
1093  DCR  B
1094  JNZ  108F
1097  POP  B
1098  MVI  M,30
109A  INX  H
109B  MVI  M,38
109D  LXI  H,0009
10A0  DAD  SP
10A1  MOV  A,M
10A2  MVI  D,20
10A4  ORA  A
10A5  JP   10BE
10A8  MVI  D,08
10AA  LXI  H,0002
10AD  DAD  SP
10AE  STC
10AF  MVI  A,9A
10B1  CMC
10B2  SBB  M
10B3  ADI  00
10B5  DAA
10B6  MOV  M,A
10B7  INX  H
10B8  DCR  D
10B9  JNZ  10AF
10BC  MVI  D,2D
10BE  LXI  H,0002
10C1  DAD  SP
10C2  MVI  E,10
10C4  MOV  A,M
10C5  CALL 1103
10C8  JZ   10D7
10CB  MOV  A,M
10CC  RAR
10CD  RAR
10CE  RAR
10CF  RAR
10D0  INX  H
10D1  CALL 1103
10D4  JNZ  10C4
10D7  LXI  H,000A
10DA  DAD  SP
10DB  MVI  E,11
10DD  MOV  A,M
10DE  CPI  2E
10E0  JNZ  10E9
10E3  DCX  H
10E4  MVI  M,30
10E6  JMP  10F5
10E9  CPI  30
10EB  JNZ  10F5
10EE  MVI  M,20
10F0  INX  H
10F1  DCR  E
10F2  JNZ  10DD
10F5  DCX  H
10F6  MOV  M,D
10F7  MVI  A,1C
10F9  SUB  C
10FA  MOV  L,A
10FB  MVI  H,00
10FD  DAD  SP
10FE  POP  D
10FF  MOV  A,C
1100  SPHL
1101  XCHG
1102  PCHL
1103  PUSH H
1104  PUSH PSW
1105  MVI  A,11
1107  ADD  E
1108  MOV  L,A
1109  MVI  H,00
110B  DAD  SP
110C  POP  PSW
110D  ANI  0F
110F  ADI  30
1111  MOV  M,A
1112  DCX  H
1113  DCR  B
1114  JNZ  111A
1117  DCR  E
1118  MVI  M,2E
111A  POP  H
111B  DCR  E
111C  RET
```

Lista 3. Zbirni program (dobljen z obratnim zbirnikom) sistemske subrutine ?QDCOP iz liste 1 (relativni naslov 0045H, absolutni naslov 0145H) kaže, kako je mogoče identificirati vstavljene sistemske subrutine. Subrutine z vprašajem v listi 1 predstavljajo v dokončno prevedenem programu večino dobljenega strojnega koda.

ne točke z uporabniškim programom. V program tudi namesti kode posameznih subrutin in oblikuje enoten strojni program, ki je pripravljen za izvajanje. LINK tako vgradi obstoječe sistemske subrutine v uporabniški program. Posamezne opcije LINK ukaza, ki se vpišejo na koncu LINK ukaza v oglate oklepaje ter ločijo z vejicami, so tele:

A — Pomnilniško stikalo A povzroči skrčenje vmesniškega prostora, začasni podatki pa se pišejo na disk. Stikalo A se uporabi tedaj, ko se je pojavilo MEMORY OVERFLOW.

Dhhhh — Stikalo podatkovnega začetka D nastavi začetek pomnilnika skupnega in podatkovnega območja na vrednost hhhh.

Gn — Stikalo G nastavi začetni naslov programa na n, kjer je n zunanje ime z največ šestimi znaki.

```
PL/I-80 V1.3  COMPILATION OF: ACKTST

B: Built-In Subroutine Trace
D: Disk Print
I: Interlist Source and Code
N: Nesting Level Display
S: Symbol Table List

   1 a      ack:
   2 b          procedure options(main,stack(2000));
   3 b          dcl
   4 b              (m,n) fixed,
   5 b              (maxm,maxn) fixed,
   6 b              ncalls decimal(6),
   7 b              (curstack, stacksize) fixed,
   8 b              stksiz entry returns(fixed);
   9 b
  10 b          put skip list('Type max m,n: ');
  11 b          get list(maxm,maxn);
  12 c              do m = 0 to maxm;
  13 d                  do n = 0 to maxn;
  14 d                  ncalls    = 0;
  15 d                  curstack  = 0;
  16 d                  stacksize = 0;
  17 d                  put edit
  18 d                      ('Ack(',m,',',n,')=',ackermann(m,n),
  19 d                       ncalls,' Calls,',stacksize,' Stack Bytes')
  20 d                      (skip,a,2(f(2),a),f(6),f(7),a,f(4),a);
  21 d                  end;
  22 c              end;
  23 b          stop;
  24 b
  25 b          ackermann:
  26 c              procedure(m,n) returns(fixed) recursive;
  27 c              dcl
  28 c                  (m,n) fixed;
  29 c              ncalls = ncalls + 1;
  30 c              curstack = stksiz();
  31 c              if curstack > stacksize then
  32 c                  stacksize = curstack;
  33 c              if m = 0 then
  34 c                 return(n+1);
  35 c              if n = 0 then
  36 c                  return(ackermann(m-1,1));
  37 c              return(ackermann(m-1,ackermann(m,n-1)));
  38 c              end ackermann;
  39 b          end ack;
  SYMBOLS:

BLOCK AT LINE 1, AUTO STORAGE 28 BYTES
a 0000 ACK ENTRY PARAMETERS(0) EXTERNAL CONSTANT
BLOCK AT LINE 3, AUTO STORAGE 16 BYTES
c 0000 M FIXED BINARY(15,0) AUTOMATIC
c 0002 N FIXED BINARY(15,0) AUTOMATIC
c 0004 MAXM FIXED BINARY(15,0) AUTOMATIC
c 0006 MAXN FIXED BINARY(15,0) AUTOMATIC
c 0008 NCALLS FIXED DECIMAL(6,0) AUTOMATIC
c 000C CURSTACK FIXED BINARY(15,0) AUTOMATIC
c 000E STACKSIZE FIXED BINARY(15,0) AUTOMATIC
c 0000 STKSIZ FUNCTION PARAMETERS(0) EXTERNAL CONSTANT
c 0000 .* FIXED BINARY(15,0) RETURNED
c 0000 * FORMAT STATIC
c 0000 ACKERMANN FUNCTION PARAMETERS(2) RECURSIVE CONSTANT
c 0000 .* FIXED BINARY(15,0) RETURNED
c 0002 .* FIXED BINARY(15,0) PARAMETER
c 0004 .* FIXED BINARY(15,0) PARAMETER
BLOCK AT LINE 27, AUTO STORAGE 8 BYTES, PARAMETERS M, N
e 0000 M FIXED BINARY(15,0) PARAMETER
e 0004 M FIXED BINARY(15,0) AUTOMATIC
e 0002 N FIXED BINARY(15,0) PARAMETER
e 0006 N FIXED BINARY(15,0) AUTOMATIC
   NO ERROR(S) IN PASS 1

   NO ERROR(S) IN PASS 2


PL/I-80 V1.3  COMPILATION OF: ACKTST

   1 a 0000 ack:
       0000       LXI  B,07D0
       0003       CALL ?START
   2 a 0006     procedure options(main,stack(2000));
   3 c 0006     dcl
   4 c 0006         (m,n) fixed,
   5 c 0006         (maxm,maxn) fixed,
   6 c 0006         ncalls decimal(6),
   7 c 0006         (curstack, stacksize) fixed,
   8 c 0006         stksiz entry returns(fixed);
   9 c 0006
  10 c 0006     put skip list('Type max m,n: ');
       0006       LXI  D,0252
       0009       LXI  B,0000
       000C       CALL ?SYSPR
       000F       MVI  A,01
       0011       CALL ?SKPOP
       0014       LXI  H,* S(001C)
       0017       MVI  A,0E
       0019       CALL ?SLCTS
       001C       CALL ?PNCOP
       001F       CALL ?QIOOP
       000A       **** 0022
  11 c 0022     get list(maxm,maxn);
       0022       LXI  D,0290
       0025       LXI  B,0000
       0028       CALL ?SYSIN
       002B       CALL ?GNVOP
       002E       JP   0000
       0031       CALL ?QCIOP
       0034       SHLD MAXM A(0050)
       002F       **** 0037
       0037       CALL ?GNVOP
       003A       JP   0000
       003D       CALL ?QCIOP
       0040       SHLD MAXN A(0052)
       003B       **** 0043
       0043       CALL ?QIOOP
       0026       **** 0046
  12 c 0046         do m = 0 to maxm;
       0046       LXI  H,0000
       0049       SHLD M A(004C)
```

Lista 5. Primer PLI programa za Ackermannovo (rekurzivno) funkcijo z generiranim zbirnim programom, ki vključuje vrsto vgrajenih (sistemskih) subrutin.

```
      004C       LHLD MAXM A(0050)
      004F       SHLD * A(005C)
      0052       LHLD * A(005C)
      0055       XCHG
      0056       LHLD M A(004C)
      0059       CALL ?IS22N
      005C       JM   0000
 13 c 005F            do n = 0 to maxn;
      005F       LXI  H,0000
      0062       SHLD N A(004E)
      0065       LHLD MAXN A(0052)
      0068       SHLD * A(005E)
      006B       LHLD * A(005E)
      006E       XCHG
      006F       LHLD N A(004E)
      0072       CALL ?IS22N
      0075       JM   0000
 14 c 0078            ncalls    = 0;
      0078       LXI  H,* S(002A)
      007B       MVI  A,06
      007D       CALL ?DLDOP
      0080       LXI  H,NCALLS A(0054)
      0083       MVI  A,06
      0085       CALL ?DSTOP
 15 c 0088            curstack  = 0;
      0088       LXI  H,0000
      008B       SHLD CURSTACK A(0058)
 16 c 008E            stacksize = 0;
      008E       SHLD STACKSIZE A(005A)
 17 c 0091            put edit
      0091       LXI  D,0252
      0094       LXI  B,0000
      0097       CALL ?SYSPR
      009A       LXI  H,* S(0000)
      009D       CALL ?EDITF
      00A0       LXI  H,* S(002E)
      00A3       MVI  A,04
      00A5       CALL ?SLCTS
      00A8       CALL ?EDTOV
      00AB       LHLD M A(004C)
      00AE       MVI  A,09
      00B0       CALL ?QICOP
      00B3       CALL ?EDTOV
      00B6       LXI  H,* S(0032)
      00B9       MVI  A,01
      00BB       CALL ?SLCTS
      00BE       CALL ?EDTOV
      00C1       LHLD N A(004E)
      00C4       MVI  A,09
      00C6       CALL ?QICOP
      00C9       CALL ?EDTOV
      00CC       LXI  H,* S(0033)
      00CF       MVI  A,02
      00D1       CALL ?SLCTS
      00D4       CALL ?EDTOV
      00D7       LXI  H,* A(0060)
      00DA       CALL 0000
      00DD       MVI  A,09
      00DF       CALL ?QICOP
      00E2       CALL ?EDTOV
      00E5       LXI  H,NCALLS A(0054)
      00E8       MVI  A,06
      00EA       CALL ?DLDOP
      00ED       MVI  A,09
      00EF       MVI  B,00
      00F1       CALL ?QBCOP
      00F4       CALL ?EDTOV
      00F7       LXI  H,* S(0035)
      00FA       MVI  A,07
      00FC       CALL ?SLCTS
      00FF       CALL ?EDTOV
      0102       LHLD STACKSIZE A(005A)
      0105       MVI  A,09
      0107       CALL ?QICOP
      010A       CALL ?EDTOV
      010D       LXI  H,* S(003C)
      0110       MVI  A,0C
      0112       CALL ?SLCTS
      0115       CALL ?EDTOV
      0118       CALL ?QIDOP
      0095       ==== 011B
      011B       LHLD N A(004E)
      011E       INX  H
      011F       SHLD N A(004E)
      0122       JMP  006B
      0076       ==== 0125
      0125       LHLD M A(004C)
      0128       INX  H
      0129       SHLD M A(004C)
      012C       JMP  0052
      005D       ==== 012F
 18 c 012F            ('Ack(',m,',',n,')=',ackermann(m,n),
 19 c 012F             ncalls,' Calls,',stacksize,' Stack Bytes')
 20 c 012F            (skip,a,2(f(2),a),f(6),f(7),a,f(4),a);
 21 c 012F          end;
 22 c 012F       end;
 23 c 012F  stop;
      012F    CALL ?STOPX
 24 c 0132
 25 c 0132  ackermann:
 26 c 0132       procedure(m,n) returns(fixed) recursive;
 27 e 0132       dcl
      0132    LXI  B,0020
      0135    LXI  D,0064
      0138    CALL ?SVBLK
      013B    LXI  D,M A(0064)
      013E    MVI  C,04
      0140    MOV  A,M
      0141    INX  H
      0142    STAX D
      0143    INX  D
      0144    DCR  C
      0145    JNZ  0140
      0148    LHLD M A(0064)
      014B    MOV  E,M
      014C    INX  H
      014D    MOV  D,M
      014E    XCHG
      014F    SHLD M A(0068)
      0152    LHLD N A(0066)
```

Lista 5. Nadaljevanje s prejšnje strani

```
      0155          MOV  E,M
      0156          INX  H
      0157          MOV  D,M
      0158          XCHG
      0159          SHLD N A(006A)
28 e 015C             (m,n) fixed;
29 e 015C             ncalls = ncalls + 1;
      015C          LXI  H,NCALLS A(0054)
      015F          MVI  A,06
      0161          CALL ?DLDOP
      0164          LXI  H,* S(0048)
      0167          MVI  A,06
      0169          CALL ?DLDOP
      016C          CALL ?DADOP
      016F          LXI  H,NCALLS A(0054)
      0172          MVI  A,06
      0174          CALL ?DSTOP
30 e 0177             curstack = stksiz();
      0177          CALL STKSIZ
      017A          SHLD CURSTACK A(0058)
31 e 017D             if curstack > stacksize then
      017D          LHLD STACKSIZE A(005A)
      0180          XCHG
      0181          LHLD CURSTACK A(0058)
      0184          CALL ?IS22N
      0187          JP   0000
32 e 018A                stacksize = curstack;
      018A          LHLD CURSTACK A(0058)
      018D          SHLD STACKSIZE A(005A)
      0188          ==== 0190
33 e 0190             if m = 0 then
      0190          LXI  H,M A(0068)
      0193          MOV  A,M
      0194          INX  H
      0195          ORA  M
      0196          JNZ  0000
34 e 0199                return(n+1);
      0199          LHLD N A(006A)
      019C          INX  H
      019D          CALL ?RSBLK
      01A0          RET
      0197          ==== 01A1
35 e 01A1             if n = 0 then
      01A1          LXI  H,N A(006A)
      01A4          MOV  A,M
      01A5          INX  H
      01A6          ORA  M
      01A7          JNZ  0000
36 e 01AA                return(ackermann(m-1,1));
      01AA          LHLD M A(0068)
      01AD          DCX  H
      01AE          SHLD * A(0070)
      01B1          LXI  H,0001
      01B4          SHLD * A(0072)
      01B7          LXI  H,* A(006C)
      01BA          CALL ACKERMANN
      01BD          CALL ?RSBLK
      01C0          RET
      01AB          ==== 01C1
37 e 01C1             return(ackermann(m-1,ackermann(m,n-1)));
      01C1          LHLD M A(0068)
      01C4          DCX  H
      01C5          SHLD * A(0078)
      01CB          LHLD M A(0068)
      01CB          SHLD * A(007E)
      01CE          LHLD N A(006A)
      01D1          DCX  H
      01D2          SHLD * A(0080)
      01D5          LXI  H,* A(007A)
      01D8          CALL ACKERMANN
      01DB          SHLD * A(0082)
      01DE          LXI  H,* A(0074)
      01E1          CALL ACKERMANN
      01E4          CALL ?RSBLK
      01E7          RET
38 e 01E8           end ackermann;
39 a 01E8        end ack;

CODE SIZE = 01E8
DATA AREA = 0084
FREE SYMS = 0150
END  COMPILATION
```

Lista 5. Nadaljevanje s prejšnje strani

Lhhhh Stikalo L za nalagalni naslov spremeni običajni nalagalni naslov modula na vrednost hhhh.

NL Stikalo NL prepreči listanje simbolne tabele na konzoli.

NR Stikalo NR prepreči generiranje simbolne tabele v obliki zbirke.

OC To stikalo naroči pri povezovalniku (LINK) proizvodnjo COM zbirke (to velja tudi običajno).

Phhhh Stikalo programskega začetka spremeni običajni programski začetek na naslov hhhh.

Q Stikalo za simbole, ki začenjajo s vprašajem, povzroči listanje teh simbolov, ki so normalno del PL/I knjižnice.

S Iskalno stikalo S povzroči, da se predhodna zbirka obravnava kot programska knjižnica.

```
A_ACKTST

TYPE MAX M,N: 4,6

ACK( 0, 0)=      1          1 CALLS,     4 STACK BYTES
ACK( 0, 1)=      2          1 CALLS,     4 STACK BYTES
ACK( 0, 2)=      3          1 CALLS,     4 STACK BYTES
ACK( 0, 3)=      4          1 CALLS,     4 STACK BYTES
ACK( 0, 4)=      5          1 CALLS,     4 STACK BYTES
ACK( 0, 5)=      6          1 CALLS,     4 STACK BYTES
ACK( 0, 6)=      7          1 CALLS,     4 STACK BYTES
ACK( 1, 0)=      2          2 CALLS,     6 STACK BYTES
ACK( 1, 1)=      3          4 CALLS,     8 STACK BYTES
ACK( 1, 2)=      4          6 CALLS,    10 STACK BYTES
ACK( 1, 3)=      5          8 CALLS,    12 STACK BYTES
ACK( 1, 4)=      6         10 CALLS,    14 STACK BYTES
ACK( 1, 5)=      7         12 CALLS,    16 STACK BYTES
ACK( 1, 6)=      8         14 CALLS,    18 STACK BYTES
ACK( 2, 0)=      3          5 CALLS,    10 STACK BYTES
ACK( 2, 1)=      5         14 CALLS,    14 STACK BYTES
ACK( 2, 2)=      7         27 CALLS,    18 STACK BYTES
ACK( 2, 3)=      9         44 CALLS,    22 STACK BYTES
ACK( 2, 4)=     11         65 CALLS,    26 STACK BYTES
ACK( 2, 5)=     13         90 CALLS,    30 STACK BYTES
ACK( 2, 6)=     15        119 CALLS,    34 STACK BYTES
ACK( 3, 0)=      5         15 CALLS,    16 STACK BYTES
ACK( 3, 1)=     13        106 CALLS,    32 STACK BYTES
ACK( 3, 2)=     29        541 CALLS,    64 STACK BYTES
ACK( 3, 3)=     61       2432 CALLS,   128 STACK BYTES
ACK( 3, 4)=    125      10307 CALLS,   256 STACK BYTES
ACK( 3, 5)=    253      42438 CALLS,   512 STACK BYTES
ACK( 3, 6)=    509     172233 CALLS,  1024 STACK BYTES
ACK( 4, 0)=     13        107 CALLS,    34 STACK BYTES
ACK( 4, 1)=
```

Lista 6. Ta lista kaže rezultate izvajanja programa ACKTST, ki smo ga poklicali z diskovnega pogona A. Program izpiše najprej zahtevo za vhodna podatka (TYPE MAX M,N:), nakar prek konzole vtipkamo vrednosti 4,6 in CR. Program začne potem izpisovati sproti izračunane rezultate za posamezne kombinacije argumentnih vrednosti. Pri ACK(3,6) imamo vrednost funkcije 509, za izračun te vrednosti pa je bilo izvedenih 172233 rekurzivnih pozivov subrutine ACKERMANN (glej listo 5), pri čemer je znašala maksimalna zasedenost sklada 1024 zlogov. Programa ACKTST nismo končali (zadnja izračunana vrednost je bila ACK(4,Ø) = 13), ker bi izračunavanje do konca vhodne zahteve, tj. do ACK(4,6), trajalo predolgo (verjetno bi tudi prekoračili rezervirano območje sklada 2000 zlogov). Po izračunu vrednosti ACK(4,0) smo izvajanje programa prekinili. Avtomatična prekinitev izvajanja programa pri določeni prekoračitvi je prikazana v listi 7 (izvajanje programa DFACT pri prekoračitvi maksimalne dopustne fiksne vrednosti rezultata).

V naslednjem primeru si oglejmo program za izračun Ackermannove funkcije, ki je rekurzivna. Subrutina za Ackermannovo funkcijo mora razpolagati z dovolj obsežnim skladom, kamor se nalagajo subrutinski vrnitveni naslovi ter začasni (vmesni) rezultati. Ackermannova funkcija A(m,n) ima tole rekurzivno opredelitev:

```
A(m,n) =
        ČE m = Ø POTEM  n + 1  SICER
        ČE  n = Ø POTEM  A(m-1,1)  SICER
            A( m-1, A(m,n-1))
```

Kot vidimo je Ackermannova funkcija lep primer večkratne rekurzije in obseg sklada je lahko presežen pri večjih vrednostih argumentov m in n. Program na listi 5 predstavlja Ackermannovo funkcijo in včita najprej maksimalni vrednosti argumentov za m in n, za kateri bo funkcija še izračunana. Ta program bo v listi izpisal še število pozivov (ncalls, ' Calls,') in število zlogov (stacksize, ' Stack Bytes'), tako da bomo imeli pregled nad številom rekurzivnih pozivov in zasedenostjo sklada. Interakcija (izvajanje) tega programa je prikazana na listi 6.

V našem primeru smo v vrstici 2 liste 5 uporabili STACK opcijo in smo s tem povečali obseg dodeljenega pomnilnika za sklad v času izvajanja. STACK opcija velja samo skupaj z MAIN opcijo in povečuje sklad iz običajnih 512 zlogov na 2000. Vrednost STACK opcije se večkrat določi s poskusom, ker rekurzijske globine s prevajalnikom ni moč vnaprej določiti. Sporočilo "Free Space Overwrite" se pojavi pri prekoračitvi sklada med rekurzijo in program se konča zaradi premajhnega pomnilnega prostora za sklad.

Na listi 5 je prikazan tudi generirani zbirni kod za ustrezni PLI stavek. K temu kodu se dodajo v končni obliki (s povezovalnikom) še vgrajene subrutine (t i s t e, ki začenjajo z znakom "?").

Lista 6 nastane pri izvajanju programa ACKTST.COM. Ker je bila izpisana z navadnim ASCII teleprinterjem, so vse črke velike (v PLI programu liste 5 se zapisane tudi male črke, npr. v vrstici 10 itn.).

Na listi 7 je prikazana interakcija programa DFACT iz liste 1, tako da imamo popolno sliko o delovanju programov DFACT in ACKTST.

```
C_DFACT

FACTORIAL(         0 )=                    1
FACTORIAL(         1 )=                    1
FACTORIAL(         2 )=                    2
FACTORIAL(         3 )=                    6
FACTORIAL(         4 )=                   24
FACTORIAL(         5 )=                  120
FACTORIAL(         6 )=                  720
FACTORIAL(         7 )=                 5040
FACTORIAL(         8 )=                40320
FACTORIAL(         9 )=               362880
FACTORIAL(        10 )=              3628800
FACTORIAL(        11 )=             39916800
FACTORIAL(        12 )=            479001600
FACTORIAL(        13 )=           6227020800
FACTORIAL(        14 )=          87178291200
FACTORIAL(        15 )=        1307674368000
FACTORIAL(        16 )=       20922789888000
FACTORIAL(        17 )=      355687428096000
FACTORIAL(        18 )=
FIXED OVERFLOW (1)
TRACEBACK: 0007 019F 0018 0000 # 2809 6874 0355 0141
END OF EXECUTION
```

Lista 7. Ta lista kaže rezultate izvajanja programa DFACT iz liste 1, ki smo ga poklicali z diskovnega pogona C. Kot vidimo, je pri FACTORIAL(18) nastopila prekoračitev območja rezultata, ki je FIXED DEC(15,Ø) (glej vrstico 12 liste 1, kjer je definiran format rezultatne vrednosti rekurzivne funkcije FACT). Vrstica TRACEBACK kaže kot najnižjo vrednost lokacije v skladu Ø141H, kar ustreza stavku glavnega programa v listi 1, pri katerem je nastala napaka. Iz liste 2 je razvidno, da začenja kod programa DFACT na lokaciji 100H, razlika je 141-100=41H in to je relativna lokacija napake. Druga podlista liste 1 kaže, da pripada ta lokacija vrstici 6 glavnega programa, ko se izračunava FACT(1).

8. Sklep k drugemu delu

V drugem delu članka smo si ogledali V/I sistem prevajalnika, na dveh primerih pa smo pokazali zmogljivost prevajalnika PL/I-80. V nadaljevanju članka (tretji del) bomo obravnavali nekaj primerov, ki bodo povezani z uporabo, oblikovanjem in spreminjanjem zbirk. Ti primeri bodo prikazovali možnost uporabe jezika PL/I-80 pri reševanju poslovnih nalog.

Z uvedbo jezika PL/I-80 v letu 1980 so za mikroračunalniške uporabnike (z operacijskim sistemom CP/M) na voljo vsi bistveni programirni jeziki velikih sistemov. Kot že zapisano, združuje jezik PL/I-80 strukturo jezika Pascal z enostavnostjo jezikov Pascal in Basic ter zmore zapletene operacije na različnih perifernih napravah.

PL/I-80 lahko rabi tudi za razvoj sistemskih programov; je torej jezik, ki ga uporabljajo sistemski proizvajalci in sestavljavci za razvoj nove programske opreme.

Vzroki za manjši interes uporabe jezika PL/I-80 pri končnih uporabnikih tičijo v veliki izbiri različnih stavkov in njihovih kombinacij. Za priučitev jezika PL/I-80 je potrebna določena, daljša doba, če se želi razumeti zapletena jezikovna specifika. Večina programerjev uporablja le določeno podmnožico stavkov jezika PL/I-80 in tako ne izkoristi jezikovne moči in premoči nad drugimi programirnimi jeziki.

Dodatna literatura

(18) PL/I Priručnik, Intertrade, IBM šolski center, Ljubljana, 1971 (287 strani, pripravljeno za vpetje, tipkano).

Priročnik opisuje dovolj široko podmnožico jezika PL/I, iz katere so izvzete asinhrone operacije in pristopi v programsko logiko med izvajanjem programa. Delo je pisano kot priročnik za programerja. Prvi del opisuje osnovne jezikovne pojme, drugi del pa se ukvarja s sintaksnimi pravili jezika.

(19) PL/I-OS Podsetnik, Intertrade, Šolski center za obdelavo podatkov na IBM sistemih, Ljubljana, 1972 (približno 200 strani, pripravljeno za vpetje, tipkano).

Ta priročnik se nanaša na PL/I(F) prevajalnik. Opisano je še delovanje povezovalnega urejevalnika, povezovalnega nalagalnika in predprocesorja.

INFORMATICA 1/1981

ELEMENTI PARALELNEGA PROCESIRANJA V OPERACIJSKEM SISTEMU RAČUNALNIKA IBM 8100

JOŽE BARLE,
BORUT KUŠTRIN

UDK: 681.3.063

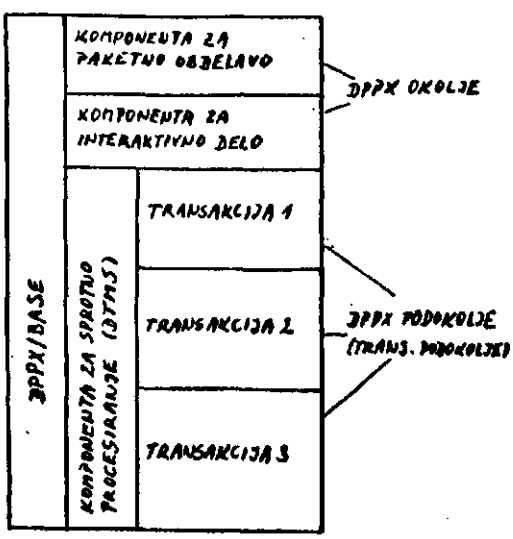
RSNZ SRS, LJUBLJANA

V članku bomo obravnavali nekatere elemente paralelnega procesiranja na računalniku IBM 8loo. S tega stališča bomo podrobneje opisali DPPX/Base kontrolni program in kot konkreten primer podali opis dela v Data Base and Transaction Management okolju (DTMS). Prikazali bomo tudi enega od možnih načinov povezave z računalnikom S/37o in sicer interaktivno povezavo aplikacije v DTMS okolju na sistemu 8loo s CICS aplikacijo na S/37o.

ELEMENTS OF PARALLEL PROCESSING IN THE OPERATING SYSTEM OF IBM 8loo COMPUTER. The article gives an overview of some elements of parallel processing in IBM 8loo computer. From this point of view DPPX/Base control program is described and as examples there are given a description of work in DTMS enviroment and a description of interactive connection of DTMS application at IBM 8loo computer with application at S/37o.

1. OPERACIJSKI SISTEM ZA DISTRIBUIRANO PROCESIRANJE - DPPX

DPPX (Distributed Processing Programming Executive) je nov komunikacijsko usmerjen multiprogramski operacijski sistem, zasnovan za splošnonamenski miniračunalnik IBM 8100. Podpira interaktivno, sprotno in paketno procesiranje ter nudi več možnosti za vključevanje v SNA (Systems Network Architecture) računalniško mrežo.



Slika 1 Primer konfiguracije DPPX sistema

DPPX ima hierarhično nivojsko strukturo. Posamezne funkcije niso duplicirane. Če en nivo vsebuje določene zmožnosti, jih po potrebi uporabljajo tudi drugi nivoji. Elementi sistema so torej v hierarhičnem medsebojnem odnosu, zasnovani pa so neodvisno drug od drugega.

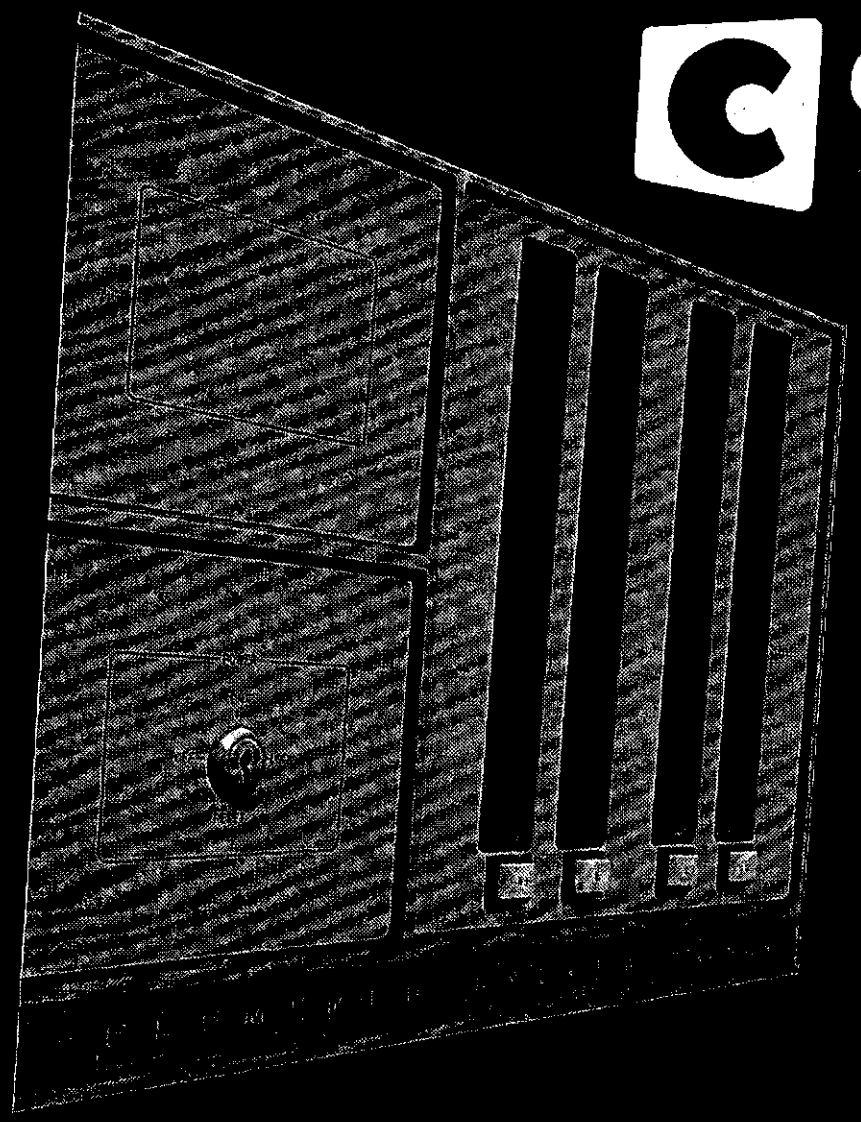
DPPX je sestavljen iz večjega števila komponent, ki lahko tečejo vsaka v svojem okolju, delo celotnega sistema pa upravlja in nadzira osnovna komponenta, imenovana DPPX/Base. Na sliki 1 je podan primer konfiguracije sistema z nekaterimi značilnimi komponentami.

2. DPPX/BASE KOMPONENTA DPPX OPERACIJSKEGA SISTEMA

DPPX/Base komponenta je osnovni del DPPXa. Njena naloga je upravljanje in nadzor celotnega sistema. Za opis načina dela DPPX/Base krmilnega programa se bomo spustili na elementarni programski nivo, pri čemer bomo posebej povdarili njegove lastnosti, ki omogočajo učinkovito multiprogramiranje.

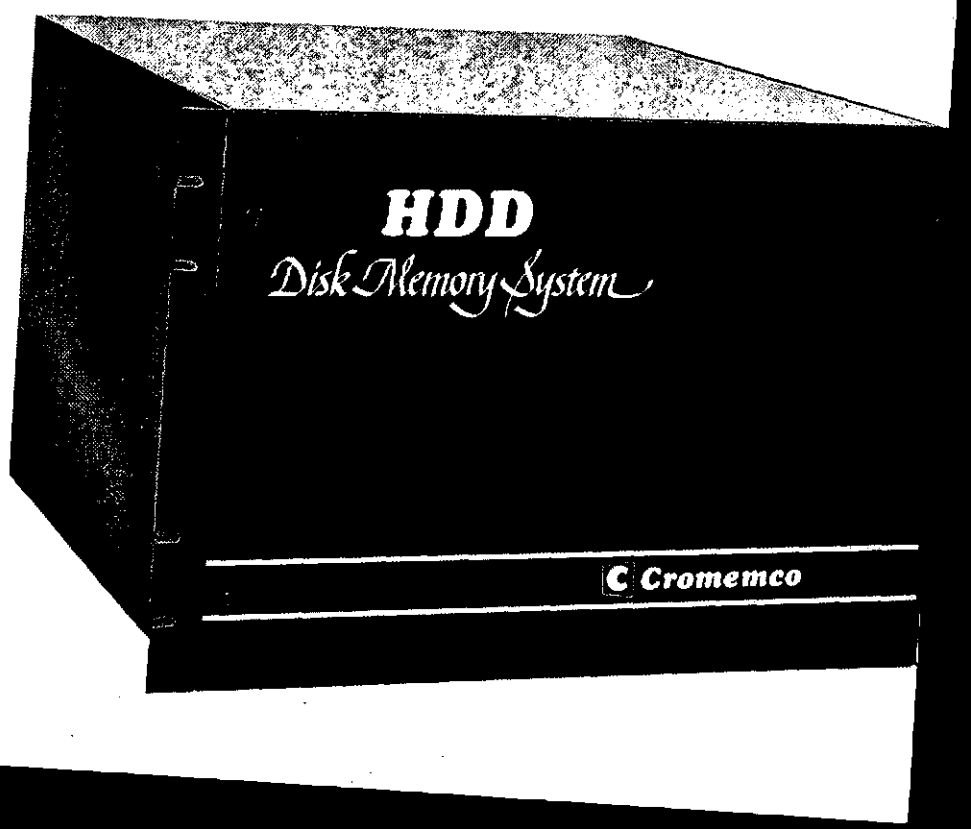
2.1. Prekinitveni nivoji:

Procesor 8100 ima osem strojnih prekinitvenih nivojev. Vsakemu programu lahko določimo na katerem od teh prekinitvenih nivojev se bo izvajal, najvišji nivo pa je rezerviran za sistemsko uporabo. Izvajanje programa se lahko prekine na osnovi V/I, programske ali strojne prekinitvene zahteve. Če ob nastopu prekinitvene zahteve procesor dela na višjem prekinitvenem nivoju od prekinitvenega nivoja, pridruženega tej prekinitveni zahtevi, materialni del shrani prekinitveno zahtevo dokler ni njej pridruženi prekinitveni nivo najvišji. Če pa nastopi prekinitev na istem prekinitvenem nivoju, procesor obdela prekinitveno zahtevo predno zapusti ta nivo.
Dejanski prenos kontrole z nivoja na nivo opravlja materialni del. Uporaba prekinitvenih nivojev omogoča dodelitev višje prioritete določenim programom. Na primer, programi za sprotno procesiranje imajo višjo prioriteto kot programi za paketno obdelavo, ki se navadno izvajajo na najnižjem prioritetnem nivoju.

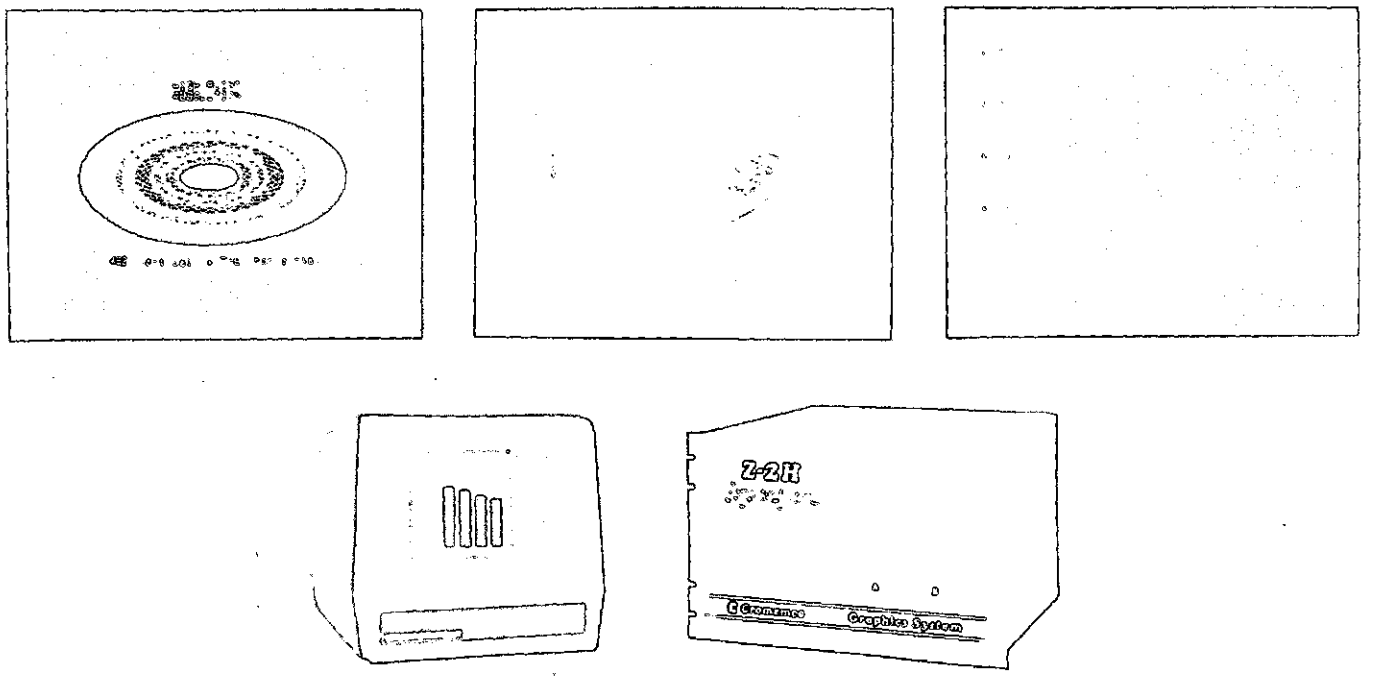


Cromemco
System Three Computer

Mikro-kompjutor za profesionalnu upotrebu

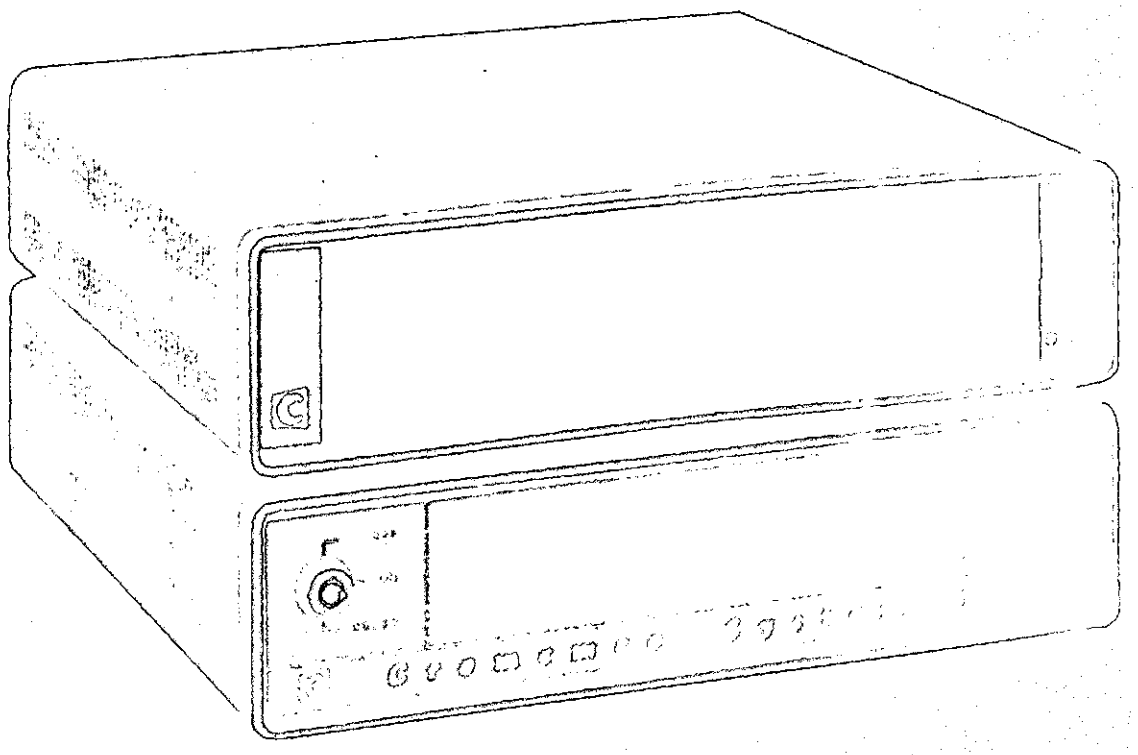


11 ili 22 Megabyte Winchester Hard disk

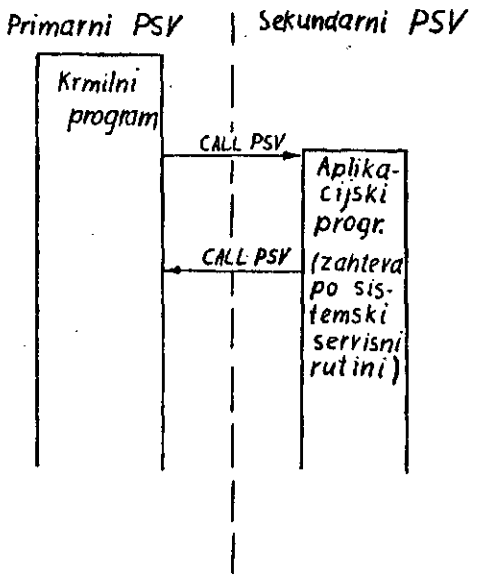


Kolor grafički sistem visoke rezolucije

System zero – namijenjen za specijalne aplikacije



2.2. Primarni in sekundarni PSV (programski statusni vektor):

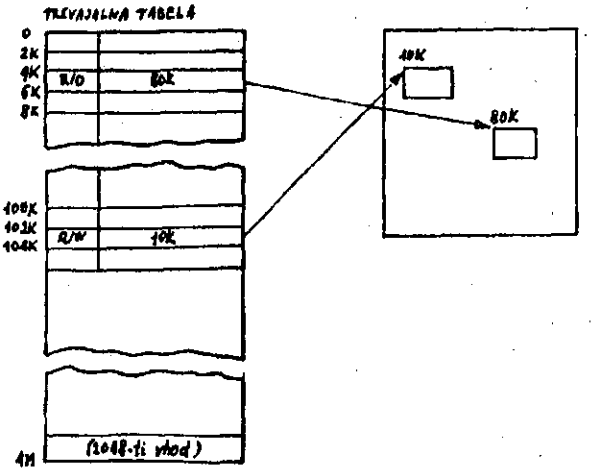


Slika 2 Sprememba načina dela

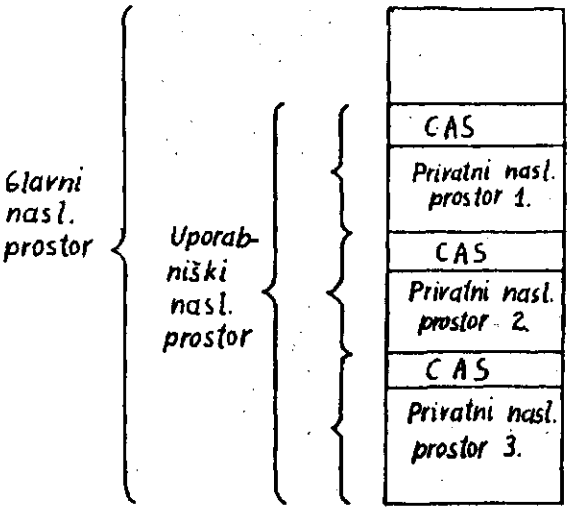
DPPX dopušča na vsakem prekinitvenem nivoju dva načina dela - nadzorni (supervisor) in aplikacijski. Vsakemu načinu je pridružen PSV, ki vsebuje podatke o naslovu naslednje instrukcije programa in o množici registrov, pridruženih programu. Vsakemu PSV je pridružen ACV (naslovni krmilni vektor - Adress Control Vector), ki vsebuje začetni naslov in dolžino naslovnega prostora okolja, v katerem se program izvaja. V splošnem lahko rečemo, da se nadzorni programi izvajajo pod kontrolo primarnega PSV, vsi ostali programi, bodisi aplikacijski ali sistemski pa pod kontrolo sekundarnega PSV.
Prehod z nadzornega načina dela na aplikacijski način omogoča CALL PSV instrukcija. Tako krmilni program z uporabo te instrukcije preda kontrolo aplikacijskemu programu, ki teče pod sekundarnim PSV, aplikacijski program pa preda kontrolo nadzornem programu z uporabo makro instrukcije, ki vsebuje CALL PSV instrukcijo.

2.3. Logični pomnilnik:

DPPX omogoča sistemskim in aplikacijskim programom naslavljanje logičnega naslovnega prostora. Logični naslovni prostor dopušča naslavljanje do 4M zlogov, maksimalna velikost realnega pomnilnika pa je 1M zlogov. Materialni mehanizem skrbi za prevajanje logičnih naslovov v realne, pri čemer uporablja prevajalno tabelo, ki vsebuje po en vpis za vsak 2K zložni blok logičnega pomnilnika. Prikaz tega stanja je na sliki 3.
Sistemu je dostopen celoten logični naslovni prostor, ki ga imenujemo glavni naslovni prostor (Master Adress Space). Kot vidimo na sliki 4, je večji del glavnega naslovnega prostora uporabniški naslovni prostor, ki se dalje deli na ločene all skupne uporabniške naslovne prostore. Vsak naslovni prostor vsebuje del, ki je skupen vsem naslovnim prostorom. To skupno naslovno področje (Common Adress Storage) omogoča dostop do ene same kopije podatkov ali programa iz več uporabnikovih naslovnih prostorov. Uporabnikov naslovni prostor vsebuje tudi zasebni naslovni prostor, ki je dostopen samo temu uporabniku.



Slika 3. Prevajalna tabela za pretvorbo logičnih v realne naslove



Slika 4. Logični spomin - razdelitev

2.4. Okolja (Environments):

Vsak uporabnik DPPX uporablja neko določeno množico računalniških virov (procesor, pomnilnik, V/I enote itd.), ki jo imenujemo okolje. Okolja so lahko vnaprej definirana all pa jih definiramo dinamično med delom sistema. Okolje kreira sistemski operater s START ukazom (vsa neinteraktivna okolja začnemo na ta način) ali pa jih vzpostavijo uporabniki z LOGON ukazom s terminala (vsak uporabnik ima na razpolago določeno okolje).
Ko je okolje vzpostavljeno, ima na razpolago določeno množico računalniških virov, ki jih bodisi deli z drugimi okolji ali pa jih ima v zasebni uporabi. Okolja lahko razdelimo na več podokolij.

2.5. Proces:

V DPPX sistemski literaturi je proces definiran kot izvršljivi program, naložen v glavni pomilnik in je pod kontrolo sekundarnega PSV-ja.

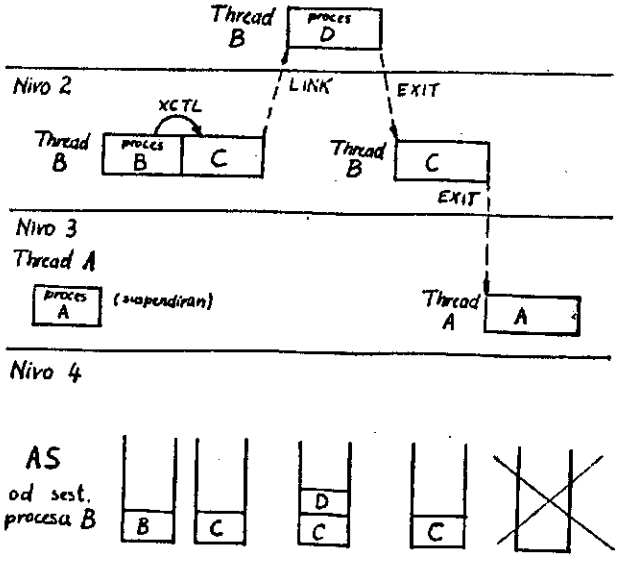
Proces je lahko:
- aktiven (run): proces se izvaja (dodeljen mu je CPE),
- pripravljen na izvajanje (ready): čaka, da se mu dodeli CPE, izpolnjeni pa so vsi pogoji za njegovo izvajanje,
- blokiran (wait): čaka, da se izpolnijo določeni pogoji za njegovo izvajanje.

2.6. Sestavljeni proces:

Razen procesa je v sistemski literaturi definiran tudi pojem nit(thread) kot določeno zaporedje izvajanja procesov pod kontrolo sekundarnega PSV. Ker gre za akcije, ki so v časovnem smislu zaporedne, lahko rečemo, da je nit tudi proces, saj popolnoma ustreza definiciji procesa pri Dijkstri (3). Pri nadaljnjem opisu bomo za nit uporabljali izraz sestavljeni proces.

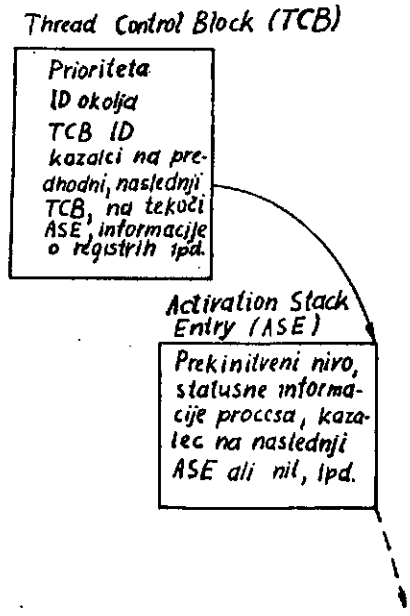
2.7. Medsebojni odnosi med prekinitvenimi nivoji, procesi in sestavljenimi procesi:

Sestavljeni procesi, ki tečejo na različnih prekinitvenih nivojih, si delijo CPE čas na podlagi prednosti nivojev (nivo 0 - najvišja prednost, nivo 7 - najnižja prednost), sestavljeni procesi na istem nivoju pa na podlagi prednosti sestavljenih procesov.
Proces se vedno izvaja samo na enem nivoju in v samo enem naslovnem prostoru, lahko pa se veže na drug proces na drugem nivoju in/ali v drugem naslovnemu prostoru.
Za vsak sestavljeni proces se formira krmilni blok (Thread Control Block - TCB), ki vsebuje informacije o tem sestavljenem procesu in kazalec, ki kaže na aktivacijski sklad procesov (AS) v tem sestavljenem procesu. Elementi AS vsebujejo informacije o procesih v sestavljenem procesu. Aktivacijski sklad je LIFO sklad.
Primer medsebojnih odnosov nivojev, procesov in sestavljenih procesov je podan na sliki 5.



Slika 5. Medsebojni odnosi med procesi in sestavljenimi procesi, aktivacijski sklad

Sestavljeni proces na nekem nivoju (sestavljeni proces A na nivoju 4) se lahko prekine (suspendira) medtem, ko dobi kontrolo sestavljeni proces na drugem, višjem prednostnem nivoju (sestavljeni proces B na nivoju 3). Proces (proces b) v sestavljenem procesu (sestavljeni proces B) lahko, ko konča z delom, prenese kontrolo na naslednji proces (XCTL), lahko pa se tudi prekine (proces c) in veže (LINK) na kakšen drug proces (proces d). Ko proženi proces (proces d) konča z delom (EXIT), dobi spet kontrolo proces, ki ga je sprožil (proces c), če seveda ni kakšnega drugega sestavljenega procesa z višjo prednostjo (npr. na nivoju 2). Ko konča z delom zadnji proces (proces c) v sestavljenem procesu, se zaključi tudi sestavljeni proces (sestavljeni proces D). Kontrolo dobi spet sestavljeni proces A, če seveda ne čaka na izvajanje kak sestavljeni proces na višjem nivoju. Sestavljeni proces (setavljeni proces A) lahko pripravi za izvajanje (READY) drug sestavljeni proces (sestavljeni proces C). Identifikatorji vsakega tekočega procesa v sestavljenem procesu se nalagajo na vrh aktivacijskega sklada. Ko proces konča z delom, se vrh sklada briše in zamenja z identifikatorji naslednjega procesa v sestavljenem procesu.
Na sliki 5 je podan tudi primer aktivacijskega sklada za sestavljeni proces B.
Na slikl 6 je prikazana povezava krmilnega bloka (TCB) in aktivacijskega sklada (AS) za sestavljeni proces. Vsak sestavljeni proces je vezan na določeno okolje. V enem okolju lahko teče več sestavljenih procesov (primer za to je DTMS okolje), ki so lahko neodvisni drug od drugega ali pa komunicirajo med seboj (multitasking oz. multithreading). Proces v sestavljenem procesu lahko proži sestavljeni proces, ki je pridružen nekemu drugemu okolju.



Slika 6. Povezava med kontrolnim blokom (TCB) sest. procesa in aktivacijskim skladom

2.8. Komunikacija in sinhronizacija med sestavljenimi procesi:

Komuniciranje med sestavljenimi procesi omogoča Queue Management (QM) komponenta krmilnega programa, ki skrbi tudi za sinhronizacijo med njimi. Pri tem se uporabljajo informacije, ki jih procesi dodajajo ali jemljejo iz vrste (queue). Vrsta je tipa FIFO, kar pomeni, da se dodaja zmeraj na konec vrste, jemlje pa iz začetka vrste.
Obstajata dva tipa vrst:
- enouporabniška vrsta (Single Server Queue) je v smislu jemanja informacij iz vrste pridružena samo enemu sestavljenemu procesu in enemu okolju, informacije pa lahko dodaja v vrsto več sestavljenih procesov, ki lahko tečejo v

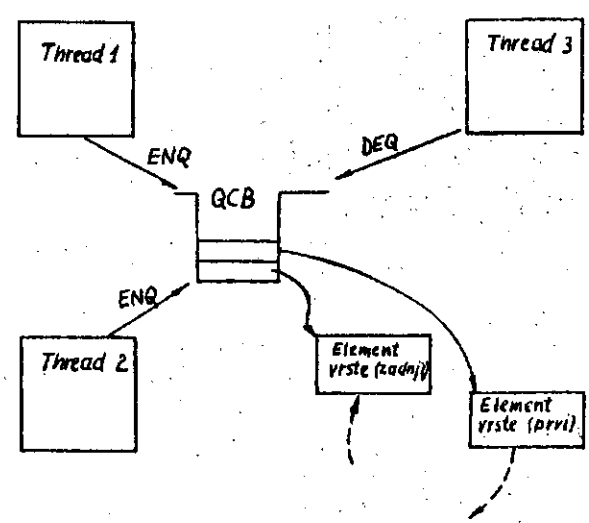
različnih okoljih. Če je vrsta prazna, lahko sestavljeni proces čaka določeno informacijo, ki naj bi jo dodal v vrsto drugi sestavljeni proces (kooperacija med sestavljenimi procesi). Na sliki 7 je primer enouporabniške vrste.
- vrsta tipa poštni nabiralnik (mailbox queue) je pridružena samo okolju. Katerikoli sestavljeni proces lahko jemlje informacije iz te vrste, seveda pri pogoju, da se izvaja v okolju, ki mu je vrsta pridružena. Ni možnosti, da bi sestavljeni proces čakal na informacijo, če je vrsta prazna. Za dodajanje informacij v vrsto ni nobenih omejitev katerikoli sestavljeni proces v kateremkoli okolju lahko vpisuje v vrsto.

Za jemanje in dodajanje informacij v vrsto procesi uporabljajo funkcije Queue Managementa, ki predstavljajo kritične dele procesov. Ko proces pokliče kakšno od teh funkcij, je onemogočena uporaba teh funkcij ostalim procesom (procesa ni možno prekiniti, dokler ne izvede te funkcije do konca).
Glavne funkcije QM so CREATEQ, ASSIGNQ, DELETEQ, SHUTDWNQ, ENQ, DEQ. Delovanje enouporabniškega (Single Server) monitorja bomo nekoliko poenostavljeno opisali v opisnem jeziku, podobnem Moduli.

```
single-server: MONITOR;
TYPE qcb=RECORD COMMENT kontrolni blok vrste;
          thread id: identifikator sest. procesa
          env. id:   identifikator okolja
          kazgla:    pointer
          kazrep:    pointer
          ostalo:    ostale informacije
          END;
TYPE tcb=RECORD COMMENT kontrolni blok
                        sest. procesa
          stanje: (pripravljen, blokiran)
          prejšnji/naslednji tcb: pointer
          prioriteta: (1:pmax)
          okolje: identifikator
          ostalo: ostale informacije o
                  sest. procesu
          END;
VAR m: sporočilo;
VAR mastm: boolean;
VAR shtdq, delq, creatq, assgnq: boolean;
VAR enable, nonempty: condition;
VAR q: vrsta;
PROCEDURE enq (m: sporočilo);
BEGIN;
  IF mastm THEN enable.wait;
  mastm:=true;
  IF shtdq OR delq THEN vrni povratno kodo;
                   ELSE append (m,q);
  mastm:=false;
  enable.signal;
  nonempty.signal;
END;
PROCEDURE deq (m:sporočilo);
BEGIN;
  IF mastm THEN enable.wait;
  mastm:=true;
  IF kazgla=nil AND kazrep=nil
  THEN IF shtdq THEN vrni povratno kodo;
                ELSE nonempty.wait;
  ELSE remove (m,q);
  mastm:=false;
  enable.signal;
END;
PROCEDURE createq;
BEGIN;
  COMMENT inicializacija qcb;
  IF mastm THEN enable.wait;
  mastm:=true;
  kazgla:=nil;
  kazrep:=nil;
  creatq:=true;
  mastm:=false;
  enable.signal;
END;
```

```
PROCEDURE assignq;
BEGIN;
  COMMENT pridružimo vrsto q (single server)
          sestavljenemu procesu
          (inicializiramo thread id v qcb);
  IF mastm THEN enable.wait;
  mastm:=true;
  assgnq:=true;
  mastm:=false;
  enable.signal;
END;
PROCEDURE shutdwnq;
BEGIN;
  COMMENT vrsto zablokiramo v smislu dodajanja;
  IF mastm THEN enable.wait;
  mastm:=true;
  shtdq:=true;
  mastm:=false;
  enable.signal;
END;
PROCEDURE deletq;
BEGIN;
  IF mastm THEN enable.wait;
  mastm:=true;
  delq:=true;
  assgnq:=false;
  creatq:=false;
  mastm:=false;
  enable.signal;
END;
COMMENT inicializacija monitorja;
  creatq:=false;
  assgnq:=false;
  shtdq:=false;
  mastm:=false;
END;
```

V zgornjem opisu imajo funkcije append in remove običajni pomen dodajanja, oziroma jemanja iz vrste. Podobno deluje tudi monitor vrste tipa poštni nabiralnik (mailbox queue), s tem da odpade monitorska procedura ASSIGNQ, ker ta vrsta ni pridružena samo enemu sestavljenemu procesu. Monitorska procedura DEQ ne more čakati na izpolnitev pogoja nonempty, v CREATEQ pa se specificira, da gre za vrsto tipa poštni nabiralnik. Ostale procedure monitorja so enake kot pri enouporabniškem monitorju.



Slika 7. Vrsta in procesi

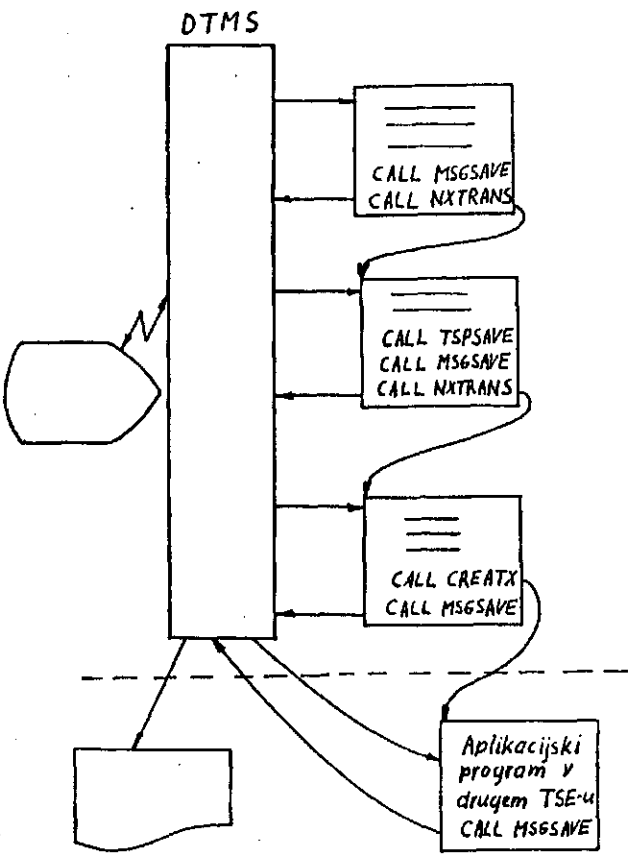
Z uporabo vrste tipa poštni nabiralnik lahko centraliziramo komunikacijo med sestavljenimi procesi. Uporabimo jo lahko na primer za medsebojno obveščanje sestavljenih procesov o tem kdaj je prost kakšen računalniški vir. En

sestavljeni proces lahko v vrsto tipa poštni nabiralnik vstavi sporočilo, da potrebuje kakšen vir in čaka v svoji enouporabniški vrsti sporočilo o tem, da je ta vir sproščen. Ko drugi sestavljeni proces sprosti ta vir, pogleda v vrsto tipa poštni nabiralnik (DEQ), če kakšen proces čaka na ta vir in pošlje (ENQ) v njegovo pridruženo enouporabniško vrsto sporočilo o tem, da je ta vir prost.

3. PRIMERI PARALELNIH PROCESOV Z UPORABNIKOVEGA STALIŠČA:

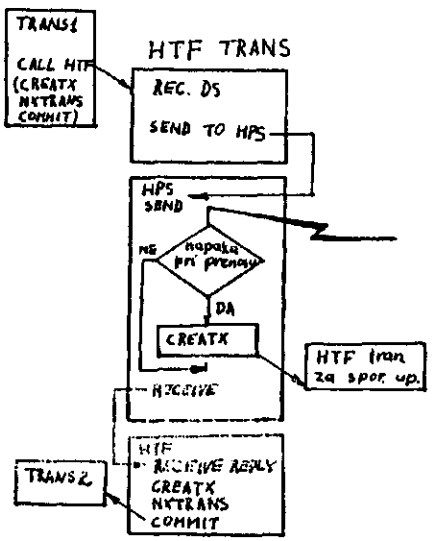
V DTMS okolju tečeta DTMS/TPM (Transaction Processing Manager) in DTMS/DBM (Data Base Manager). V DTMS okolju je vgnezdenih več podokolij (Transaction Subenvironments - TSE), v katerih se izvajajo transakcije. Navadno imajo ta podokolja različne naslovne prostore kot DTMS in tudi različne drugo od drugega. S tem so podokolja med seboj izolirana in pred nedovoljenim dostopom so zaščiteni tudi DTMS kontrolni bloki in DTMS vmesniki.
Za kontrolo dela v podokoljih skrbi kontrolni program, ki teče v vsakem od teh podokolij. Od DTMSa dobiva navodila, kateri aplikacijski program naj izvaja. Ta program nato pošlje v pomnilnik in mu preda kontrolo; kontrolo spet prevzame, ko program konča z delom in sporoči DTMSu, da je podokolje prosto.
Ko DTMS sprejme zahtevo za izvajanje transakcije, poskuša najti prosto podokolje. Če ga najde, preda zahtevo kontrolnemu programu v tem podokolju, če pa so zasedena vsa podokolja, vpiše zahtevo v čakajočo vrsto. Iz čakajoče vrste jemlje zahteve na podlagi relativne prednosti, ki jo določimo pri definiciji transakcije. Ta prednost seveda ne vpliva na transakcijo, ki se že izvaja v podokolju. Transakcija se izvaja do konca, čeprav ima nižjo relativno prednost kot čakajoča transakcija.
Za primer (slika 8) si oglejmo transakcijo v obliki segmentiranega dvogovora, ki teče v enem podokolju in proži drugo transakcijo, ki se izvaja asinhrono v nekem drugem podokolju. Pri tem uporabljamo naslednje DTMS rutine (ki jih lahko kličemo tudi iz programa, napisanega v višjem programirnem jeziku):
- MSGSAVE za pošiljanje sporočil na terminal,
- NXTRANS za predajo kontrole naslednjemu segmentu,
- TSPSAVE za predajanje podatkov naslednjemu segmentu
- CREATX za proženje asinhrone transakcije.
Oglejmo si še en zanimiv primer načina dela v DTMS okolju in sicer interaktivno povezavo DTMS aplikacije na sistemu 8100 s CICS/VS (Customer Information Control System / Virtual Storage) aplikacijo na sistemu S/370 (slika 9). Takšno povezavo omogoča HTF (Host Transaction Facility), ki je del DPPX/Base komponente operacijskega sistema DPPX.

HTF upravlja in nadzira povezavo s CICS/VS operacijskim sistemom na gostiteljskem računalniku, teče pa kot transakcija v enem ali več DTMS podokoljih, ki smo jih definirali za povezavo s CICS/VS sistemom. Uporabnikova transakcija, ki zahteva povezavo z gostiteljskim sistemom, kliče HTF Aplication Program Interface (API) in mu preda transakcijsko kodo ter vhodne podatke za CICS/VS transakcijo. API nastopa kot del uporabnikove transakcije in s CREATX postavi zahtevo za izvajanje HTF transakcije. HTF nato uporabi HPS (Host Presentation Services) komponento DPPX/Base, ki opravlja in nadzoruje dejanski prenos podatkov. Ko HPS sprejme odgovor, ga preda HTF transakciji, ta pa ga posreduje uporabnikovi transakciji, določeni za sprejem odgovora.
Vsaka transakcija v DTMS okolju je v bistvu sestavljeni proces. DTMS okolje lahko vsebuje pet podokolij, od katerih sta dve definirani za



Slika 8. Primer segmentiranega dvogovora v DTMS okolju

povezavo s CICS/VS sistemom. V takem DTMS okolju lahko simultano teče pet sestavljenih procesov, vsak v svojem podokolju. Ti sestavljeni procesi lahko komunicirajo drug z drugim, dva pa lahko komunicirata tudi s procesi v gostiteljskem sistemu. DTMS okolje je torej tipično multiprogramsko okolje.



Slika 9. Potek HTF transakcije

4. SKLEP

V članku smo obdelali le nekatere elemente paralelnega procesiranja na sistemu 8100. Kljub temu je razvidna nova zasnova tega operacijskega sistema v primerjavi s sistemi serije 360/370. DPPX je namreč že v osnovi izrazito namenjen za sprotno procesiranje, medtem ko so sistemi serije 360/370 orientirani za paketno obdelavo. Značilnosti, ki omogočajo učinkovito paralelno procesiranje, sinhronizacijo in komunikacijo med procesi so realizirane že na strojnem (osem strojnih prekinitvenih nivojev) in elementarnem programskem nivoju (mehanizmi razdeljevanja, ločeni uporabniški naslovni prostori, mehanizmi za sinhronizacijo in komunikacijo med procesi itd.)

5. SLOVSTVO

(1) S.C. Kiely: An Operating System for Distributed Processing - DPPX, IBM Systems Jour. 18 (1979), Nr.4, 507-525.
(2) C.A.R. Hoare: Monitors: An Operating System Structuring Concept, CACM,Vol. 17,n. 10,October 1974.
(3) Dijkstra E. W.:Cooperating Sequential processes, in Programming languages, ed. Genuys, Academic Press 1968.
(4) Distributed Processing Programming Executive Base(DPPX/Base) Programming: Guide to System Services, sistemska literatura, Nr. SC27-0405.

INFORMATICA 1/1981

OTKRIVANJE I POPRAVLJANJE GREŠAKA U RAČUNARSKIM MEMORIJAMA (ECC)

MILOVAN V. JEFIĆ

UDK: 681.3.001.4

SOZD ELEKTROTEHNA DO DELTA, LJUBLJANA

Pouzdan memorijski sistem se može napraviti ako upotrebimo elemente visoke sigurnosti. Pri tome moramo uzeti u obzir prisustvo statistične vjerovatnoće za pojavljivanje kvara sa obzirom na pouzdanost elemenata. To dode posebno do izraza pri velikim memorijama koje sadrže veliki broj elemenata. Da bi mogli odkriti koji od elemenata je pokvaren dodajemo sistemu posebne informacijske bitove. Kod sistema gdje se iz ekonomskih razloga zahtijeva minimalno zaustavljanje ili ako se mora obezbijediti duži period rada bez greške se primjeni u sistemu metod otkrivanja i popravljanja greške (ECC). U članku su obrađena oba navedena načina povećanja sigurnosti memorijskog sistema.

Reliable memory systems can be designed with highly reliable burned in components, but with any electronic component there is a statistical probability of its failing, regardless of its reliability. As larger memory systems became economical the statistical probability that there would be a failure of one the thousands of components within the memory system became significant. A common hardware approach to monitor the failure of the component was to add one extra data bit to the system. This "parity" could detect a single bit failure (or the failure of ann odd number of bits) and would halt the system. On large systems where the cost per hour requires minimal downtime or where long periods of error-free operation be considered. This article describes both codes.

UVOD

Svi elementi sistema nisu jednako osjetljivi i važni za osnovno djelovanje sistema. Memorija je jedan od djelova za kojeg je jako važno sigurno djelovanje. Ovdje su pregledane različite metode, a posebno ECC (Error Correction Codes) za povećavanje pouzdanosti memorijskog sistema.

SISTEMSKA POUZDANOST

Pouzdanost pojedinih elemenata je osnova pouzdanosti memorijskog sistema, koja je dana kao srednje vrijeme između dva kvara (MTBF).
MTBF je funkcija broja elemenata i broja kvarova na njima, za pojedini element je:

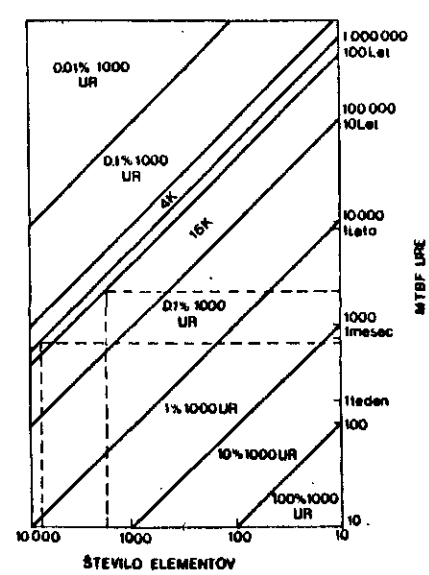
$$T_E = \frac{1}{\lambda} \qquad (1)$$

gdje je T_E = MTBF za element, a λ = broj kvarova na elementu (%/1000 sati).
Odatle je MTBF za sistem:

$$T_S = \frac{T_E}{E} \qquad (2)$$

gdje je T_S = MTBF sistema i E broj elemenata u sistemu.

Iz jednačine se može zaključiti da ako je broj elemenata veći sistemska pouzdanost pada što prikazuje takode i diagram na slici 1.



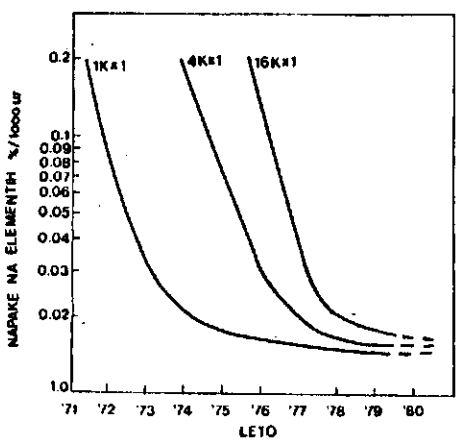
Sl. 1. Pouzdanost sistema je odvisna od broja elemenata.

Za ilustraciju uzmimo memorijski sistem kapacitete 1M riječi sa dužinom riječi 32 bita. Sistem je realiziran sa memorijskim elementima 16K X 1.

Broj potrebnih elemenata je:

$$E = \frac{1,048,576 \times 32}{16,384} = 2048$$

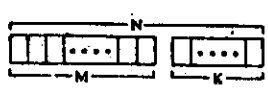
Predpostavljena greška u tom sistemu desit će se poslije 2000 sati rada prema diagramu sa sl. 1. ili poslije 2668 sati prema jednačini (2).
Za povećanje sigurnosti sistema je razvijena tehnika otkrivanja i popravljanja grešaka.



Sl.2. Greške na elementima u odvisnosti od vremena

REDUNDANDNE KODE

Redundandno kodiranje, odnosno dodatni bitovi su upotrebljeni za ustanovljavanje da li se je dogodila greška. Riječ dužine N ima M podatkovnih bitova i K kodnih. Izgled takve zakodirane riječi je:



Sl.3. Izgled zakodirane riječi

ili zapisano

$$N = M + K \qquad (3)$$

Jedna od ocjena da li je kodiranje dobro izabrano je efikasnost S, koja se izražava kao:

$$S = \frac{N}{M} = \frac{M + K}{M} \qquad (4)$$

Podatci su sadržani u M bitovima, a preostalih K bitova je za kontrolu greške.

Iz jednačine (4) ustanovimo da ako maksimiziramo efikasnost S, u slučaju S=1 ne možemo detektirati greške. Analizirajmo jednu dvo bitnu riječ. Ta može imati $2^2=4$ moguća stanja:

00
01
10
11

Sl.4. Stanja dvo bitne riječi

Sva stanja su upotrebljena kao podatci, tako da nije moguća detekcija nikakve greške. Ako sada dodamo jedan K bit na dvo bitnu riječ je moguće otkrivanje greške. Vrijednost K bita je takva da ima tako formirana riječ neparan broj "jedinica". U koliko se sada dogodi greška broj jedinica sadržanih u riječi se promjeni u paran broj.
Dvobitna riječ sa dodanim K bitom ima izgled:

00	1
01	0
10	0
11	1
M	K
N	

Sl.5. Stanje zakodirane dvo bitne riječi

Moguće jednobitne greške su:

	A	B	C	D
Zakodirano stanje	001	010	100	111
Pogrešno stanje	000 011 101	000 011 110	000 101 110	011 101 110

Sl.6. Sve moguće jednostruke greške

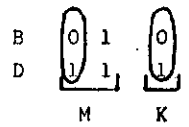
Decimalno predstavljene greške su:

Odgovarajuće stanje	1	2	4	7
Pogrešno stanje	0 3 5	0 3 6	0 5 6	3 5 6

Sl.7. Decimalno predstavljene greške

Ako analiziramo tabelu na sl.6. ustanovimo da se svako nepravilno stanje razlikuje od pravilnoga za jedan bit,

a dva nepravilna stanja se razlikuju za dva bita. Pogledajmo dva stanja sa slike 6 označena sa B i D:



Ta dva stanja se razlikuju za dva bita i tu razliku nazivamo distanca. Prema tome je definicija distance; to je broj bitova za koje se razlikuju dvije riječi. Zakodirana riječ ima minimalnu distancu dva.
Duža zakodirana riječ može imati veću distancu od dva ali nikada manju u koliko želimo otkrivati grešku.

PARNOST

Parnostni bit se generiše kao rezultat logične funkcije exkluziv-OR svih bitova u riječi i predstavljen je kao kodni bit K u riječi. Ako riječ ima M informacijskih bitova je parnostni bit C:

$$C = b_1 \oplus b_2 \oplus \ldots\ldots \oplus b_m \qquad (5)$$

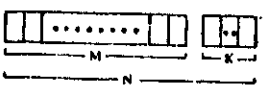
gdje je b vrijednost bita na odredenom mjestu u riječi.

Parnostni bit se kombinira sa originalnim podatkovnim bitovima i tako formiraju zakodiranu riječ.
Kada primimo tako formiranu riječ, parnostni bit se izdvoji iz riječi i zapamti. Novi parnostni bit se generiše iz M' pročitanih bitova te se uporeduje sa zapamćenim pri pisanju i tako ustanovimo da li se je dogodila jednobitna greška.

POPRAVKA GREŠKE

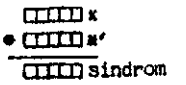
U klasičnim tekstovima o korekciji greške nademo da je minimalna distanca izmedu zakodiranih riječi tri u koliko želimo popravljati grešku. [1, 2] .

Osnovni članak iz toga područja je objavio R. W. Hamming, tako je po njemu i dobila ime Hamming-ova koda. Upotrebljavajući jednaku tehniku kao što je parnostna Hamming-ova koda generira K kodnih bitova koje dodajemo ka M podatkovnim bitovima.
U memoriji se pohrani N bitna riječ:



Sl.8. Izgled zakodirane riječi

Kada se riječ pročita iz memorije se novi komplet kodnih bitova (K') generiše iz (M') podatkovnih bitova i uporedi sa ranije dobijenim K kodnim bitovima. Komparacija se napravi sa exkluziv-OR tehnikom:



Sl.9. Generisanje sindroma

Rezultat komparacije se naziva sindromska riječ, koja sadrži informaciju da li se je dogodila greške na kojem bitu.
Ona je dakle dugačka K bitova i ima 2^k različitih vrijednosti izmedu 0 i 2^k-1. Jedna izmedu tih vrijednosti obično nula, se upotrebi za indikaciju pravilnog stanja, a preostalih 2^k-1 stanja određuje koji izmedu N bitova je pogrešan.

Područje u kojem se mora nalaziti vrijednost K je slijedeće

$$2^k-1 \geqslant N$$
$$N = M + K$$
$$2^k-1 \geqslant M + K \qquad (6)$$

Jednačina (6) daje broj K bitova potrebnih za popravku jednostrukih grešaka u riječi koja sadrži M bitova za podatke.
Za različite vrijednosti K izračunato područje M-a daje tabela:

	Jednobitna detekcija i korekcija		Jednobitna korekcija Dvobitna detekcija	
K	≤ M ≤		≤M≤	
4	4	11	1	3
5	12	26	4	10
6	27	57	11	25
7	52	120	26	56
8	121	245	57	119

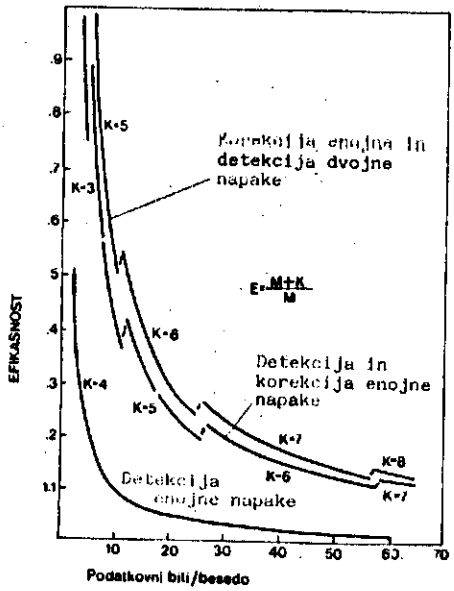
Sl. 10. Područje vrednosti M-a za korekciju jednobitne i detekciju dvobitne greške

Iz tabele sl. 10. se vidi, da za korekciju jednobitne greške u riječi sa 16 bitova trebamo pet dodatnih bitova što formira riječ sa 21 bitom.

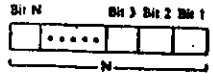
Efikasnost u tom slučaju kao funkcija podatkovnih bitova izgleda kao na sl.11.

RAZVIJANJE KODE

Sadržaj sindromske riječi je informacija potrebna za odredivanje koji bit je pogrešan. Uz potrebno dekodiranje te informacije pogrešan bit korigujemo tako da ga invertiramo.
Potrebno je napomenuti da su svi bitovi uključujući i kontrolni bit identificirani sa pozicijom u riječi.

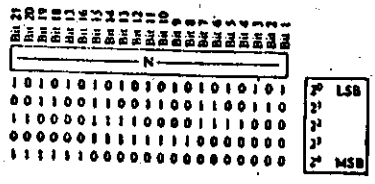


Sl.11. Efikasnost kode u zavisnosti od podatkovnih bitova



Sl. 12. Bitovi predstavljeni pozicijsko u riječi.

Organizaciju bitova u N bitni riječi prikazuje sl. 12. Primjer na sl. 13 je prikazana 16 bitna riječ. To znači za 16 informacijskih bitova M = 16 je K = 5 i N = 21. Na toj slici je riječ prikazana kao binarni ekvivalent pozicije, a gdje su lociranni M i K bitovi još nije odredeno.



Sl.13. Binarna vrijednost za poziciju bita

Sindromska riječ je razlika izmedu kontrolnih bitova dobijenih pri pisanju (primanju) i regenerisanih kontrolnih bitova pri čitanju (predaji). Pogrešan bit se identificira z binarnom vrijednostipozicije bita u sindromskoj riječi.

Sama sindromska riječ nastane s exkluziv-OR operacijom izmedu kontrolnih bitova. Sve razlike izmedu novih i starih kontrolnih bitova postave "1" u sindromskoj riječi. Za identificiranje pogrešnog bita 3 ima sindromska riječ oblik 00011 što je binarna vrijednost pozicije bitova. Obično umjesto jedinica pišemo znak "X", a nule ne pišemo, rezultat čega je tabela na sl. 14.

Bit	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	N																				
C1	X		X		X		X		X		X		X		X		X		X		X
C2			X	X			X	X			X	X			X	X			X	X	
C3	X	X					X	X	X	X					X	X	X	X			
C4							X	X	X	X	X	X	X	X							
C5	X	X	X	X	X	X															

Sl. 14. Odnos informacijskih i kontrolnih bitova

Svaki kontrolni bit može biti na bilo kojem mjestu, koji sadrži znak "X" u vrsti kao što prikazuje slika 14. Pet pozicija bitova i to 1,2,4,8 i 16 imaju samo po jedan znak "X" u koloni. Odgovarajuće kontrolne bitove postavimo upravo na ta mjesta, tako da imamo C1,C2,C4,C8 i C16.

Ako imamo sada jedan pogrešan kontrolni bit sindromska riječ će sadržavati samo jednu "1", a za slučaj da je podatkovni bit pogrešan sindromska riječ ima dvije ili više jedinica.

Podatkovni bitovi su postavljeni u polja izmedu kontrolnih bitova tako da je bit sa najmanjom vrijednosti (LSB) lociran na poziciji 3.

Sl.15 prikazuje pozicije podatkovnih i kontrolnih bitova za riječ sa 16 bitova. Svaki kontrolni bit je dobijen z exkluziv-OR operacijom izmedu podatkovnih bitova označenih z "X" na sl. 14.

$$
\begin{aligned}
C1 &= M1 \oplus M2 \oplus M5 \oplus M7 \oplus M9 \oplus M11 \oplus M12 \oplus M14 \oplus M16 \\
C2 &= M1 \oplus M3 \oplus M4 \oplus M6 \oplus M7 \oplus M10 \oplus M11 \oplus M13 \oplus M14 \\
C4 &= M2 \oplus M3 \oplus M4 \oplus M8 \oplus M9 \oplus M10 \oplus M11 \oplus M15 \oplus M16 \\
C8 &= M5 \oplus M6 \oplus M7 \oplus M8 \oplus M9 \oplus M10 \oplus M11 \qquad (7) \\
C16 &= M12 \oplus M13 \oplus M14 \oplus M15 \oplus M16
\end{aligned}
$$

16	15	14	13	12		11	10	9	8	7	6	5		4	3	2		1		
					C16								C8				C4		C2	C1
21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Sl. 15. Pozicije podatkovnih i kontrolnih bitova

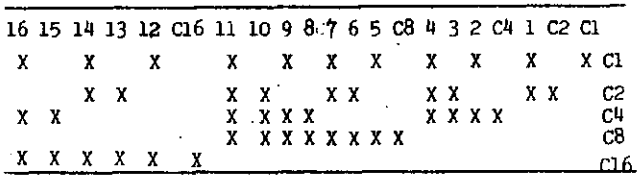
Kako Hamming-ov kod popravlja greške je najbolje objasniti na primjeru. Uzmimo 16 bitnu riječ:

0101 0000 0011 1001 (8)

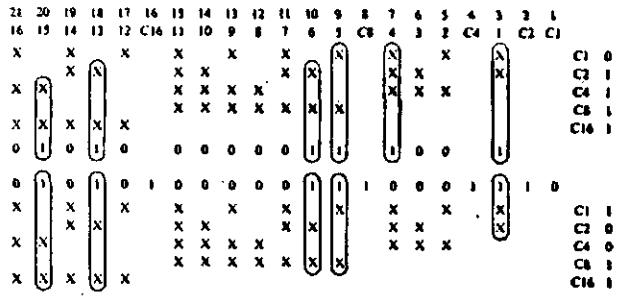
Kontrolni bitovi se generišu sa pregledom riječi za podatke u Hammingovom diagramu na sl. 16 i računanjem parnog (odd) broja znakova "X"

Jednostavan mehanizam računanja kontrolnih bitova je objašnjen na sl. 17.

Informacijske riječi su rasporedene u diagramu, a kolone u kojima se nalaze jedinice su zaokružene. Kontrolni bitovi su rezultat računanja neparnoga broja jedinica u vrsti.



Sl. 16. Hamming-ov diagram



Sl. 17. Generisanje kontrolnih bitova

Tako na primjer za C1, vrsta u kojoj se on nalazi ima zaokružena tri znaka "X" i da ostane neparan broj jedinica C1 mora imati vrijednost 0.
Ako isto napravimo za ostale kontrone bitovedobijemo zakodiranu riječ sa 21 bitom.

```
                C16                 C8          C4      C2  C1
0101    0  1  000    0011    1  100      1   1   1   0   (9)
```

Sada napravimo grešku na bitu sa pozicijom 7 odnosno informacijskom bitu 4:

```
                C16                 C8          C4      C2 C1
0101    0  1  000    011     1 000      1   1   1   0       (10)
```

Novi kontrolni bitovi generisani iz riječi rasporedene na drugom dijelu sl. 17 su:

```
C16  C8  C4  C2  C1
 1    1   0   0   1                                    (11)
```

Ako napravimo sada exkluziv-OR operaciju izmedu starih i novih kontrolnih bitova dobijemo sindromsku riječ:

```
     C16  C8  C4  C2  C1
      1   1   0   0   1    novi
⊕     1   1   1   1   0    stari
                                                      (12)
      0   0   1   1   1
```

Iz rezultata 00111 ustanovimo da je bit sa pozicijom 7 odnosno informacijski bit 4 pogrešan. Kao što vidimo Hamming-ov kod nije kompliciran, a realizacija prinese nešto problema.
Prvo broj bitova povezanih sa exkluziv-OR operacijom varira izmedu 5 i 10 tako da je prelazno vrijeme signala različito. Sistem čeka uvjek najduži prelazni period što ga usporava.

Drugi problem se javi u koliko su pogrešna dva bita tada se javlja jedan ispravan bit kao da je on pogrešan. Na primjer ako su kontrolni bitovi C1 i C2 pogrešni informacijski bit 1 će signalizirati da je pogrešan.
Zbog toga se više upotrebljava kod za popravljanje greške (ECC) poznat pod nazivom "modificiran" Hammingov kod, koji detektira sve dvostruke greške, a popravlja greške na jednome bitu.

KOREKCIJA JEDNE GREŠKE I DETEKCIJA DVOSTRUKE GREŠKE

Može se dokazati da za korekciju greške na jednom bitu i detekciju dvobitne greške je potrebna minimalna distanca četiri izmedu dvije zakodirane riječi [1,2]. Kao rezultat toga se pojavljuje zaštita svakog podatkovnog bita sa tri kontrolna tako da ima sindromska riječ uvijek neparnu vrijednost.
Sindromska riječ poprimi parno vrijednost samo u slučaju da su dva kontrolna bita pogrešna.

Broj potrebnih kontrolnih bitova za korekciju jedne greške i detekciju dvojne daje relacija:

$$2^M \leqslant \frac{2^{N-1}}{N}$$

Sindromska rieječ za popravku jedne i detekciju dvije greške je izvedena iz originalnog Hamming-ovog koda sa tim da ne pokazuje direktno poziciju bita.
Pri tom odredivanju mogu se upotrijebiti kode i empirične metode.
Sva moguća stanja kontrolnih bitova se ispišu u obliku tabele izuzimajući sva ona sa parnim brojem jedinica sl. 18.
Dalje se jednako kao pri Hammingovom kodu stanja koja imaju samo po jednu jedinicu upotrijebe za kontrolne bitove. Za riječ sa 16 bitova potrebno je 6 dodatnih bitova, sindromska riječ može zauzeti stanja kao na sl. 18.

Na sl. 18. je prikazanih samo dvadeset sindromskih riječi, jer su zbog pojednostavljivanja realizacije izostavljene riječi sa pet jedinica. Pri tome je svaki podatkovni bit zaštičen sa po tri kontrolna bita. U slučaju da imamo riječ dužine 20 moramo upotrebiti sva stanja ta kode i ona sa širinom 5 (5 jedinica).
Izbor optimalne skupine sindromskih riječi mora biti takav da pojednostavljuje realizaciju, tako pri bayt-no orijentiranim sistemima niži sindromski bitovi pokazuju koji bit je pogrešan, a viši koji bayt.

C6	C5	C4	C3	C2	C1
1	1	1	0	0	0
1	1	0	1	0	0
1	1	0	0	1	0
1	1	0	0	0	1
1	0	1	1	0	0
1	0	1	0	1	0
1	0	1	0	0	1
1	0	0	1	1	0
1	0	0	1	0	1
1	0	0	0	1	1
0	1	1	1	0	0
0	1	1	0	1	0
0	1	1	0	0	1
0	1	0	1	1	0
0	1	0	1	0	1
0	1	0	0	1	1
0	0	1	1	1	0
0	0	1	1	0	1
0	0	1	0	1	1
0	0	0	1	1	1
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	0	0	0
1	0	0	0	0	0

Sl. 18. Moguće sindromske riječi

REALIZACIJA ECC ZA 16 BITNU RIJEČ (DELTA 240,440,4780)

Memorija sa ECC pored normalne konfiguracije ima još pet dodatnih blokova:

1. Generator kontrolnih bitova pri pisanju
2. Generator kontrolnih bitova pri čitanju
3. Sindromski generator
4. Sindromski dekoder
5. Korekcija pogrešnog bita

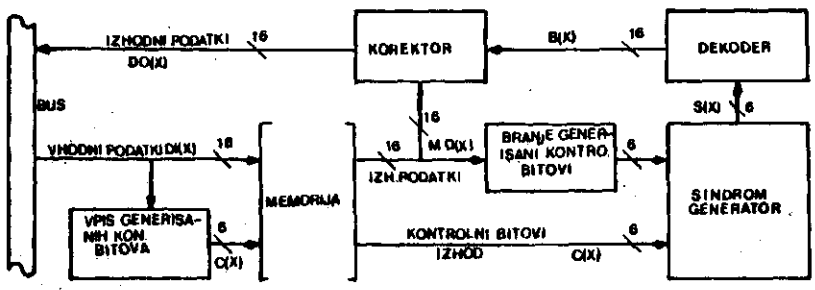
Povezani skupa kao što prikazuje sl.20 ti elementi sastavljaju bazični sistem.
Neki sistemi informaciju o otkrivenoj greški zapamte u posebnoj memoriji i iskoriste je poslije u toku održavanja za identifikaciju pokvarenog elementa. Moguće sindromske riječi bez označenih nula prikazuje sl. 19.

```
  1  1 11  1 11 111     C1
 1  1 1 1 1 1 11 11     C2
1  1  11 1  11 11 1     C3
1   111   111   111     C4
1111      111111        C5
1111111111              C6
```

Sl.19. Moguće sindromske riječi sa tri kontrolna bita

Vidimo da tabela sadrži 20 mogućih sindromskih riječi tako da svaka vrsta ima deset "1", a kolona tri "1".
Pošto je potrebno samo 16 sindromskih riječi eliminiramo četiri kolone tako da svaka vrsta ima osam "1".
Na takav način smo definirali ulaze u (74S280) parnostni generator na čijem izlazu dobijemo kontrolni bit C_n.
Ako eleminiramo po dvije kolone na početku i na kraju tabele dobijemo kao što smo prije rekli 16 kolona tako da ima svaka vrsta osam "1".

Tu je potrebno napomenuti da se generisanje kontrolnih bitova pri pisanju i čitanju vrši na isti način. Detekcija dvostruke greške se izvrši generisanjem paritete (exkluziv-OR) sa sindromskim bitovima.
U koliko isključimo sindromsku riječ 000000, nema pogreške, paran broj "1" u sindromskoj riječi detektuje dvije greške. Realizacija kompletnog ECC-a je prikazana na sl. 22.

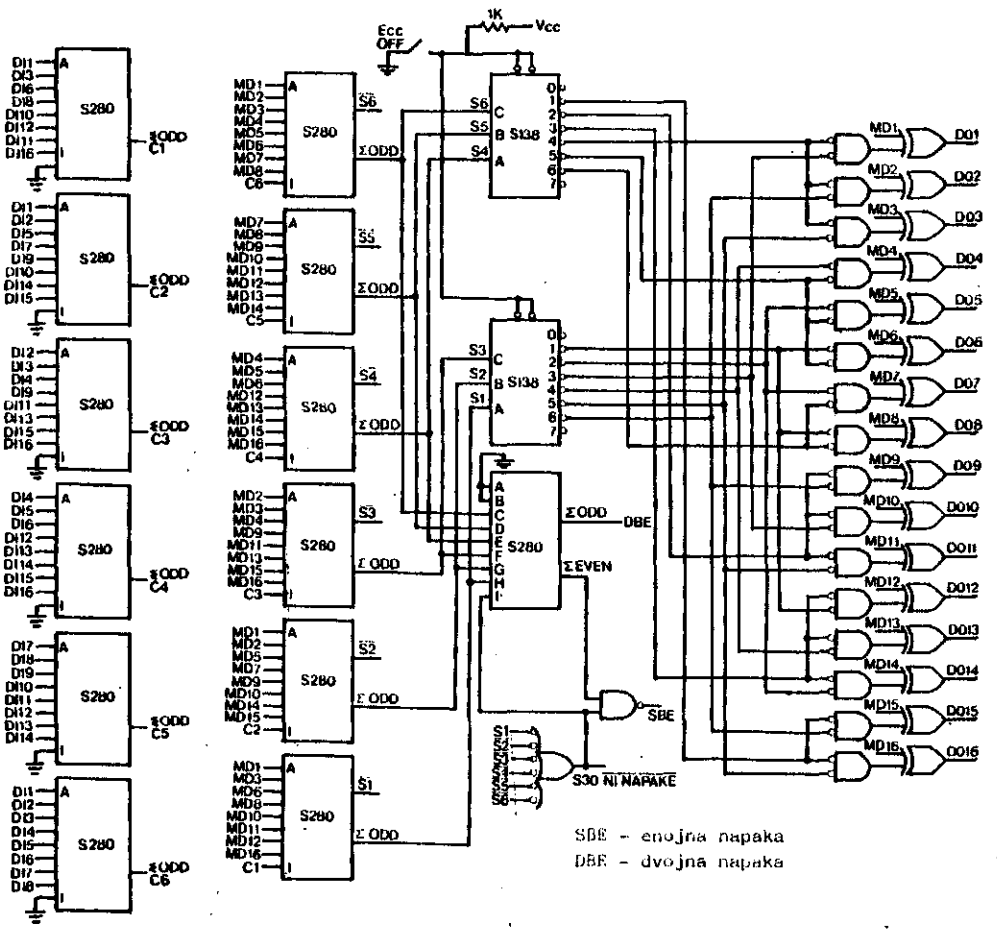


Sl. 20. Blok diagrama za ECC sistem

Podatkovni bit

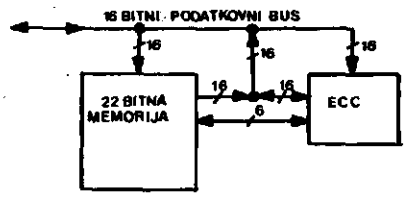
M16	M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	C1	C2	C3	C4	C5	C6	
X				X	X	X		X		X			X		X	X						C1
	X	X				X	X		X		X			X	X		X					C2
X	X		X		X		X					X	X	X				X				C3
X	X	X	X	X						X	X	X							X			C4
		X	X	X	X	X	X	X	X											X		C5
								X	X	X	X	X	X	X	X						X	C6

Sl. 21. Razvijanje kode



Sl. 22. Realizacija ECC-ja

Danas možemo dobiti navedenu šemu ECC-a realiziranu u jednom čipu što pojednostavi realizaciju kompletnog memorijskog sistema.



Sl. 23. Realizacija ECC sa jednim čipom

Neki od čipova koji se mogu dobiti trenutno na svjetskom tržištu su navedeni u tabeli na slici 24.

Proizvođač	Čip	Prelazno vrijeme
Texas Instruments	SN 74LS630 SN 74LS631	40 ns
AMD	Am 2960	50 ns
National Semiconductor	DP 8400	30 ns
Motorola	MC 68540	

Sl. 24. Pregled čipova

Svi ti ECC čipovi imaju 16 bitnu (neki i 8 bitnu) organizaciju, ako imamo 32 bitnu i dužu riječ vežemo dva ili više ECC čipova u seriju.

ZAKLJUČAK

Memorija bez zaštite ima MTBF koji je približno jednak MTBF-u elementa dijeljenog sa brojem elemenata.
Za zaštitu memorije upotrebljavamo redundandne kodove kao što je parnostni za detekciju greške i "modificiran" Hammingov kod za popravku jedne greške i detekciju dvojne greške.
Na takav način možemo dosta poboljšati MTBF sistema.

Literatura:

1. W.N. Peterson, Error - Correcting Codes, M. I.T. Press, 1972
2. R.W. Hamming, Error Detecting and Error Correcting Codes, Bell System Tehnical Journal, Vol. 26, 1950, pp 147-160
3. A Texsas Instruments Application Report, Bulletin CA-201
4. AMD, Am 2960 Error Detection and Correction Unit
5. National Semiconductor, E^2C Expandable, Advanced information, august 1980.

INFORMATICA 1/1981

PROGRAM ZA OBLIKOVANJE BESEDIL

V. MAHNIČ,
B. VILFAN

U DK: 681.39

INSTITUT JOŽEF STEFAN, ODSEK ZA UPORABNO MATEMATIKO
FAKULTETA ZA ELEKTROTEHNIKO

POVZETEK: Opisan je program, ki omogoča oblikovanje besedil v naravnem jeziku. Program poskrbi za levo in desno poravnanost besedila, tako da bodisi avtomatsko deli besede, ki presega dolžino vrstice, ali pa da v vrstico vrine potrebno število presledkov, pri čemer s posebnim algoritmom določa, med katere besede v vrstici naj bo presledek vstavljen. Poleg tega lahko uporabnik s pomočjo dokaj obsežnega nabora preoblikovalnih ukazov dodatno specificira svoje želje glede oblike izhodnega besedila (n.pr. velike/male črke, podčrtovanje in centriranje delov teksta, razmaknjeno pisanje, spreminjanje medvrstičnega razmaka, izris kazala itd.). Preoblikovan tekst lahko izpišemo na vrstičnem tiskalniku, električnem pisalnem stroju ali pa na printer/plotterju Versatec 1200A.

ABSTRACT: We describe a text formatting program whose capabilities include the left and right justification of text based on automatic word division and/or the insertion of the proper number of blanks, the latter task being accomplished by a special algorithm that determines the words between which blanks have to be inserted. Additionally, the user can specify the form of the output text (lower/upper case letters, centering or underlining, interline spacing, printing the table of contents etc.) by a quite powerfull set of formatting instructions. The formatted text may be printed on a line printer, an electric typewriter or the Versatec 1200A printer/plotter.

1. UVOD

Program za oblikovanje besedil, ki je opisan v tem sestavku, je sestavni del računalniškega programskega paketa za avtomatizacijo projektiranja, ki sa za potrebe DO ISKRA Avtomatika, TOZD Projektiranje in gradnja sistemov razvijata Odsek za uporabno matematiko Instituta Jožef Stefan in Fakulteta za elektrotehniko v Ljubljani /1,2/. Omenjeni paket se trenutno uporablja za projektiranje zavarovanja cestno-železniških prehodov in je nastal z namenom, da bi projektanta čim bolj razbremenili rutinskih postopkov v zvezi s projektiranjem in izdelavo projektne dokumentacije.

Sestavni del projektne dokumentacije je (poleg načrtov in specifikacije stroškov) tudi besedni opis vseh naprav, ki jih je potrebno vgraditi, skupaj z navodili za vgradnjo in opisom delovanja. V zvezi z izdelavo tega dela projektne dokumentacije je bilo zato treba poiskati tako rešitev, ki bi omogočala avtomatsko oblikovanje ustreznega besedila in njegov izpis na izhodni napravi, ki po svoji kvaliteti ne bi zaostajala za pisalnim strojem. V skladu s temi zahtevami je nastal program, ki je predmet tega sestavka. Seveda pa je program popolnoma splošen, tako da je uporaben za oblikovanje kakršnegakoli teksta.

Program zlasti olajša delo v slučajih, ko je treba posamezne dele besedila spremeniti, jih črtati ali na novo dodati, saj v vseh teh primerih ni treba na novo tipkati celotnega besedila, temveč vnesemo le popravke in neurejeno besedilo ponovno obdelamo s preoblikovalnim programom.

Celoten program je napisan v programskem jeziku PASCAL /3/ za računalnik CDC Cyber, prav sedaj pa je v teku prilagoditev tega programa za računalnik DELTA.

2. OPIS VHODNE DATOTEKE

Vhodna datoteka za preoblikovalni program vsebuje besedilo, ki ga želimo oblikovati, skupaj z ukazi, ki določajo način oblikovanja. Seznam ukazov je podan v dodatku k temu članku. Besedilo v vhodni kodi je znakovno zaporedje brez kakršnekoli strukture, kar ustreza n.pr. datoteki tipa TEXT v smislu programskega jezika PASCAL ali pa COMPILE datoteki UPDATE programa oziroma fortranski kodirani datoteki.

Ker računalnik CDC Cyber, za katerega je program realiziran, ponazarja znake s 6-bitno display kodo, mi pa smo želeli izpisovati večji nabor znakov (velike in male črke), smo vpeljali

- ukaza LC (lower case) in UC (upper case), s katerima definiramo obliko črk, in
- poseben kontrolni znak za napovedovanje velikih oziroma malih črk.

Če velja ukaz LC, se izhodni tekst izpiše z malimi črkami, program pa sam poskrbi za velike začetnice na začetku stavkov. Če pa velja ukaz UC, se tekst, ki sledi temu ukazu, izpiše z velikimi črkami. Obliko črk lahko z uporabo teh dveh ukazov med vhodnim besedilom poljubno spreminjamo.

Medtem ko ukaza LC in UC veljata za ves tekst, ki jima sledi (dokler pač ne naletimo na nov ukaz UC oziroma LC), se kontrolni znak za napovedovanje velikih oziroma malih črk nanaša samo na črko, ki stoji neposredno za njim. V primeru, ko velja ukaz LC, postane ta črka velika, če velja ukaz UC, pa mala. Standardna vrednost kontrolnega znaka za napovedovanje velikih oziroma malih črk je asterisk (zvezdica), lahko pa jo poljubno spreminjamo.

Poleg tega kontrolnega znaka razpoznava program še en kontrolni znak, ki služi za napovedovanje preoblikovalnih ukazov. Le-ti določajo, kako naj se tekst oblikuje, in so lahko poljubni kjerkoli med vhodnim besedilom. Vsak ukaz ima splošno obliko

kXX,...,

kjer je k kontrolni znak za napovedovanje ukazov (standardno je to procent), XX je oznaka preoblikovalnega ukaza (glej dodatek), nato pa sledi (morebiti prazno) zaporedje parametrov. Parametri so tako med seboj kot tudi od oznake ukaza ločeni z vejico.

3. OBLIKOVANJE BESEDILA

Uporabnik definira obliko strani s pomočjo ukazov za levi, desni, zgornji in spodnji rob. Program poskrbi za pravilno poravnanost vrstic z besedilom tako, da ali vstavi potrebno število presledkov ali pa da avtomatsko deli besedo, ki bi prekoračila dolžino vrstice. Od števila presledkov, ki jih je potrebno vriniti, je odvisno, katero izmed omenjenih variant program izbere. Če je to število manjše od 4, program izvrši vrivanje presledkov, v nasprotnem primeru pa deli besedo. Za deljenje besed imamo trenutno razvita dva algoritma: prvi omogoča deljenje besed v slovenskem, drugi pa v srbohrvatskem jeziku. Vrivanje presledkov poteka po posebnem algoritmu, ki skrbi, da se presledki vstavijo med daljše besede in na tistih mestih, kjer nastopajo ločila, tako da so čim bolj enakomerno razporejeni po vrstici in je videz izhodnega besedila čim lepši.

Številčenje strani poteka avtomatično. Poleg tega lahko uporabnik definira še naslov in podnaslov, ki naj se izpišeta v glavi vsake strani.

Uporabniku je dana tudi možnost, da med obdelavo 'izključi' delovanje preoblikovalnega programa. V tem primeru preoblikovalni program prepisuje vhodno besedilo na izhodno datoteko, ne da bi sa spreminjal. Pač pa preoblikovalni program še vedno razpoznava kontrolni znak za napovedovanje velikih in malih črk, medtem ko mora za krmiljenje izpisa skrbeti uporabnik sam, tako da v začetku vsake vrstice vhodnega besedila zapiše ustrezen krmilni znak ('1', '0', ' ' ali '+'). Ti znaki služijo preoblikovalnemu programu za krmiljenje izpisa in se v izhodnem tekstu ne pojavljajo. Ta način delovanja preoblikovalnega programa je zlasti uporaben za obdelavo vnaprej pripravljenih tabel in smo sa pri postopku avtomatizacije projektiranja uporabili za izpis specifikacij stroškov opreme in montaže.

Preostali ukazi omogočajo uporabniku programa, da specificira dodatne želje glede oblike izhodnega besedila. Tako lahko n.pr. podčrta, izpiše centrirano ali razmaknjeno (s presledki med črkami) del besedila, prepreči deljenje posameznih besed ali dela teksta, določa in spreminja medvrstični razmak, zahteva preskok določenega števila vrstic, puščanje praznega prostora (za naknadno vstavljanje slik), izpis v zahtevani koloni itd.

S pomočjo ukaza SC (special character) lahko uporabnik zahteva, da preoblikovalni program pusti prazen prostor za naknaden vpis (ali letrasetiranje) posebnih znakov, kot so na primer srbske črke, razni matematični simboli (integral, suma) itd., ki jih lepopisna izhodna naprava ne more izpisati. V tem primeru dobi uporabnik poleg oblikovanega teksta tudi izpis seznama posebnih znakov, ki pove, na kateri strani in v kateri vrstici izhodnega dokumenta je potrebno vpisati kak poseben znak.

Posebna odlika preoblikovalnega programa pa je avtomatsko generiranje kazala. Za razliko od podobnih programov /4-8/ nudi naš preoblikovalni program možnost, da se kazalo izpiše kjerkoli, tudi med besedilom, in ne samo na začetku ali na koncu. Pri tem program sam poskrbi, da se številčenje strani pravilno nadaljuje iz teksta pred kazalom preko kazala v tekst, ki sledi kazalu. Kazalo je torej v celoti sestavni del izhodnega besedila in ni oštevilčeno ločeno od njega, kot pri ostalih preoblikovalnih programih (v kolikor le-ti sploh nudijo možnost izdelave kazala).

4. IZPIS OBLIKOVANEGA BESEDILA

Oblikovano besedilo lahko izpišemo na vrstičnem tiskalniku, na električnem pisalnem stroju (IJS, oddelek za elektroniko) ali pa na printer/plotterju Versatec 1200A (FAGG). Izpis na vrstičnem tiskalniku služi uporabniku predvsem kot osnova za kontrolo in vnašanje morebitnih popravkov, končni izdelek pa se izpiše na eni izmed lepopisnih naprav (pisalni stroj, Versatec).

Za potrebe izpisa na Versatecu je bil izdelan poseben program, ki izhod iz preoblikovalnega programa zapiše kot zaporedje EBCDIC kod za posamezne znake, razdeli to zaporedje na logične zapise in doda ustrezno kontrolno informacijo, kot to zahteva Versatecov vektorski procesor.

5. PRIMERJAVA Z DRUGIMI PREOBLIKOVALNIMI PROGRAMI

Naš program bomo primerjali s podobnima programoma /4,5/, ki sta bila razvita pri nas, ter s programi PROSE /6/, RUNOFF /7/ in SCRIPT /8/, ki so bili razviti v tujini. Program za urejanje tekstov v naravnem jeziku /4/, programa PREPIS in PROSE ter naš program so implementirani na računalnikih firme CDC, program RUNOFF je na razpolago na računalnikih PDP oziroma DELTA, programa SCRIPT pa pri nas še nismo zasledili, vendar ga omenjamo zato, ker je gotovo eden izmed najmočnejših programov te vrste.

Vsi programi za oblikovanje besedil so si nedvomno sorodni v tem, da razpolagajo najmanj z osnovnim naborom ukazov, s pomočjo katerih lahko uporabnik specificira obliko strani v izhodnem besedilu (levi, desni, zgornji in spodnji rob oziroma število znakov v vrstici in število vrstic na eni strani). Razlikujejo pa se po načinu, kako izvajajo poravnavanje besedila v vrstici (ali samo z vrivanjem presledkov ali tudi z avtomatskim deljenjem besed) in pa po obsegu dodatnih funkcij, ki so na razpolago uporabniku (velike/male črke, avtomatska izdelava kazala, izdelava imenskega kazala, možnost izpisa na različnih lepopisnih napravah itd.).

Za primerjavo med prej navedenimi programi smo zato izbrali naslednje kriterije:

- zmožnost avtomatskega deljenja besed,
- zmožnost izpisa z velikimi in malimi črkami,
- zmožnost avtomatske izdelave kazala,
- zmožnost izdelave imenskega kazala (indeksa) in
- zmožnost izpisa oblikovanega besedila na različnih lepopisnih napravah.

Zmožnost avtomatskega deljenja besed zasledimo samo pri programih /4/, RUNOFF, SCRIPT in pri našem programu. Program PROSE lahko deli besedo le na mestih, ki jih uporabnik vnaprej označi.

Izpisovanje izhodnega besedila z velikimi in malimi črkami je možno pri našem programu in pri vseh v tujini razvitih programih.

Zmožnost avtomatske izdelave kazala nudita samo naš program in program SCRIPT.

Avtomatsko izdelavo imenskega kazala zasledimo pri programih PROSE, RUNOFF in SCRIPT, pri našem programu pa je v teku implementacija ustrezne rešitve.

Oblikovano besedilo se lahko izpiše na različnih lepopisnih napravah pri programih PROSE, RUNOFF, SCRIPT in pa pri našem programu.

Rezultati primerjave tako kažejo, da naš program prekaša oba pri nas prej razvita programa, je približno enakovreden programoma RUNOFF in PROSE, zaostaja pa (tako kot vsi ostali) za programom SCRIPT.

6. ZAKLJUČEK

Iz vsega tega je razvidno, da imamo na voljo dokaj močan sistem za oblikovanje besedil. Pri njegovem razvoju so bile uporabljene nekatere originalne rešitve (tu mislimo predvsem na postopek izdelave kazala), ki jih v podobnih programih doslej še nismo zasledili. Obenem pa so v teku nadaljnje izboljšave, ki bodo omogočile avtomatsko izdelavo imenskega kazala (indeksa) in izpisovanje opomb na koncu vsake strani. Prav tako je v teku prilagoditev tega programa za računalnik DELTA, tako da ga bo možno uporabljati tudi na računalnikih te vrste.

DODATEK: SEZNAM PREOBLIKOVALNIH UKAZOV

BM,ii, - bottom margin, spodnji rob

Nastavljanje spodnjega roba. Spodnji rob je v vrstici ii, kjer je i decimalna cifra med 0 in 9.

CA, - concatenation, konkatenacija

Predhodna in naslednja beseda se staknета, kar pomeni, da se odpravi medbesedni presledek.

CC,i,z, - control character, kontrolni znak

Kontrolni znak i (i je 1 ali 2) dobi vrednost z. Kontrolni znak 1 napoveduje velike/male črke, kontrolni znak 2 pa napoveduje ukaze. Primer: Po ukazu %CC,1,&, napovedujemo velike oziroma male črke z znakom &.

CN, - contents listing, izpis kazala

Označuje mesto, kjer naj se izpiše kazalo.

CT, - centered text, centrirano besedilo

Zaporedje %CT,...%CT, povzroči, da se besedilo, ki ga predstavljajo pike, izpiše na sredi vrstice, enako oddaljeno od obeh robov.

DV, - word division, deljenje besede

Na tem mestu se obvezno deli beseda. Tekoča vrstica se zaključi in izvrši se skok v novo vrstico.

ED,i, - editor on/off, vklop/izklop preoblikovanja

%ED,+, povzroči, da preoblikovalni program deluje normalno, %ED,-, pa izključi delovanje preoblikovalnika. V tem primeru preoblikovalni program le prepisuje vhodno besedilo na izhodno datoteko, ne spreminja pa njegove oblike (upošteva le kontrolni znak za velike/male crke in znake za krmiljenje izpisa).

FN, - footnote, opomba na dnu strani

Besedilo med ukazoma %FN,...%FN, pred-

tavlja opombo, ki naj se izpiše na koncu strani (implementacija tega ukaza je v teku).

HD, - heading, naslov

V zaporedju %HD,i,...%HD,...%HD, predstavljajo pike naslov, ki naj se izpiše v kazalu zamaknjen za i mest. Prve tri pike predstavljajo številko naslova, naslednje tri pa tekst naslova.

IE, - index entry, geslo

Beseda, ki sledi temu ukazu, je geslo v imenskem kazalu (implementacija tega ukaza je v teku).

IL,i, - inter-line spacing, medvrstični presledek

Parameter i določa razmak med vrsticami izhodnega besedila.

IT, - indivisible text, nedeljivo besedilo

%IT,...%IT, pomeni, da besede oziroma besedila, ki ga predstavljajo pike, ne smemo deliti.

LC, - lower case, male črke

Besedilo, ki sledi temu ukazu, se izpiše z malimi črkami. V tem primeru kontrolni znak 1 napoveduje velike črke. Po vsaki piki, klicaju, vprašaju in dvopičju se avtomatsko izpiše velika črka.

LM,ii, - left margin, levi rob

Levi rob zavzema ii mest (i je med 0 in 9).

MS, - message, sporočilo

Zaporedje %MS,...%MS, povzroči, da se na novi strani izpiše besedilo ..., nakar se pisalni stroj ustavi. Ta ukaz uporabljamo v primeru, ko je treba dati kako navodilo operaterju pri lepopisni izhodni napravi (n.pr. zamenjava formata papirja ipd.).

NL, - new line, nova vrstica

S tem ukazom zahtevamo, da se tekoča vrstica zaključi in se izvrši skok v novo vrstico.

PG, - new page, nova stran

Zaključi tekočo vrstico in povzroči skok na novo stran.

PR, - paragraph, nov odstavek

Zaključi tekočo vrstico in izvrši skok v nov odstavek (med odstavkoma pusti eno vrstico prazno, obstaja možnost spreminjanja zamika prve vrstice v novem odstavku).

RM,ii, - right margin, desni rob

Desni rob je na mestu ii (i je med 0 in 9).

RC,c,ii, - repeat character, ponovi znak

V izhodnem besedilu izpiše znak c ii-krat.

SC,cc,ii, - special character, posebni znak

V oblikovanem besedilu pusti ii presledkov za naknadni vpis posebnega znaka cc.

SK,ii, - skip, preskok

Povzroči preskok ii vrstic. Če na tekoči strani ni dovolj prostora, izvrši skok na novo stran.

SP, - spaced text, razmaknjeno pisanje

Besedilo med ukazoma %SP,...%SP, se izpiše razmaknjeno.

ST,zz...z, - subtitle, podnaslov

Tekst zz...z predstavlja podnaslov, ki se izpiše v glavi vsake strani.

TB,ii, - tab, tabulator

Pomik v ii-to kolono (i je med 0 in 9).

TM,ii, - top margin, gornji rob

Gornji rob je v ii-ti vrstici (i je med 0 in 9).

TT,zz...z, - title, naslov

Znakovno zaporedje zz...z predstavlja naslov, ki se izpiše v glavi vsake strani.

UC, - upper case, velike črke

Velike črke so standardne, kontrolni znak 1 napoveduje male črke.

UN, - underline, podčrtano

Znakovno zaporedje %UN,...%UN, povzroči, da se besedilo, ki ga predstavljajo pike, izpiše podčrtano.

WI,ii, - window, okno

V oblikovanem besedilu pusti (ne glede na medvrstični razmak) ii vrstic praznega prostora za naknadno vstavljanje slike. Če na tekoči strani ni dovolj prostora, nadaljuje z obdelavo besedila in izpusti zahtevan prostor ob prvem skoku na novo stran.

LITERATURA:

1. B. Vilfan, R. Kolar, V. Mahnič, M. Marušič, M. Toni: Programski paket za avtomatizacijo projektiranja, XIV. simpozij Informatica, Bled, oktober 1979

2. B. Vilfan, J. Barle, R. Kolar, V. Mahnič, M. Marušič, M. Toni: Projektiranje i izrada projektne dokumentacije s pomoću računskog stroja, IV. bosanskohercegovački simpozij iz informatike, Jahorina 1980

3. K. Jensen, N. Wirth: PASCAL User Manual and Report, Springer Verlag 1975

4. M. Martinec, I. Lajovic: Program za urejevanje tekstov v naravnem jeziku, IX. simpozij Informatica, Bled, oktober 1974

5. A. Vitek: PREPIS, RC FAGG, Ljubljana

6. J.P. Strait: PROSE Instruction Manual, University of Minnesota, 1978

7. RUNOFF Program Version M01, Revision Date 1 July 77

8. SCRIPT User's Guide, University of Kentucky, December 1978

informatica 82

PRIJAVA REFERATA/KRATKEGA REFERATA/ STROKOVNEGA POROČILA

Prijavo izpolnite s pisalnim strojem

1. Naslov referata .....
2. Razširjen povzetek (približno 1000 besed) priložite prijavi
3. Programsko področje referata (obkrožite ustrezno točko)
   1. programska oprema
   2. materialna oprema
   3. teoretični aspekti obravnavanja podatkov
   4. sistemi za upravljanje in administracijo
   5. upravljanje procesov
   6. razne aplikacije v znanosti in tehniki
   7. vzgoja in aplikacije v humanistiki
4. Razvrstitev referata (obkrožite ustrezno točko)
   1. referat — pomembnejše delo
   2. kratki referat
   3. strokovno poročilo
5. Značaj referata (obkrožite ustrezno točko)
   1. originalno teoretično delo
   2. opis konkretnega praktičnega dela
   3. pregledni referat
   4. ponovitev že znanega rezultata z novimi metodami
   5. kritična analiza
   6. opis novih hardwarskih in softwarskih proizvodov
   7. .....
6. Avtor in soavtorji: .....

   Delovna organizacija .....

   Ulica .....

   Poštna številka ..... Mesto .....

   Dežela .....

   Datum ..... Podpis .....

Prijavnica, skupaj z dvema kopijama razširjenega povzetka mora prispeti najkasneje do 1. avgust 1981 na naslov: Informatica '82 Parmova 41, 61000 Ljubljana,

PAPER/SHORT PAPER/TECHNICAL REPORT REGISTRATION

This application should be typewritten

1. Title of the Paper .....
2. Extended summary (approximately 1000 words) should be enclosed
3. Program Category of the Paper (circle appropriate choice)
   1. computer software
   2. computer hardware
   3. theoretical aspects of information processing
   4. system for management and administration
   5. process control
   6. miscellaneous scientific and engineering applications
   7. education and applications in humanities
4. Classification of the Paper (circle appropriate choice)
   1. Paper
   2. Short paper
   3. Technical report
5. Character of the Paper (circle appropriate choice)
   1. original theoretical work
   2. description of practical experience
   3. survey paper
   4. achievement of a known result by new methods
   5. critical analysis
   6. description of new hardware and software products
   7. .....
6. Author and Co-authors .....

   Organization .....
   Street .....
   Postal Code ..... City .....
   Country .....

   Date ..... Signature .....

This application form together with two copies of extended summary must reach the following address before or on August 1, 1981:
Informatica '82 Parmova 41, 61000 Ljubljana, Yugoslavia

INFORMATICA 1/1981

MEHURČNI POMNILNIKI III

B. MIHOVILOVIĆ,
J. ŠILC,
P. KOLBEZEN

UDK: 681.327.6

INSTITUT „JOŽEF STEFAN"

V članku so opisane nekatere organizacije magnetnih mehurčnih pomnilnikov kot so: dolgi premikalni register, major/minor zanke, koincidenčna izbira podatkovnega bloka, dekoderji in dinamično urejeni podatkovni bloki; njihov razvoj in primerjave med njimi. Čeprav je danes najčešče uporabljena major/minor zančna organizacija, niso ostale organizacije nič manj pomembne, saj vsebujejo mnoge dobre lastnosti. Tako recimo organizacija s koincidenčno izbiro podatkovnega bloka vnaša modularnost, skrajša čase dostopa, kakor tudi zmanjšuje dovzetnost do napak. Organizacija z dekoderji ima nekatere prednosti v primerjavi z major/minor zančno organizacijo, saj nudi možnost zbiranja pomnilniških lokacij z adresiranjem znotraj samega pomnilniškega elementa. Najkrajše čase dostopa in cikla pa doseže organizacija z dinamično urejenimi podatkovnimi bloki.

MAGNETIC BUBBLE MEMORIES - PART 3. In this paper the concepts of several bubble memory organizations such as: long shift register, major/minor loops, coincidence selection, decoders and dynamically - ordered shift registers have been reviewed; and their efficacies in achieving economy and improving performance have been compared. Both the conceptual and hardware developments for these memory organizations will be updated. While most of the hardware development is concentrated on the major/minor loops, it should be noted that the other three schemes are by no means less important. The coincident selection scheme provides array modularity, limits vulnerability to defects, and reduces access time for large - capacity chips. Decoders are more versatile than the major/minor loops since they offer address selection capability on the bubble chip. The dynamic ordering greatly reduces the access and cycle times.

1. UVOD

Današnji razvoj mehurčnih pomnilnikov poteka v več smereh:

- fizikalno-tehnološki razvoj stremi k izpopolnitvi sedanjih in uvajanju novih tehnologij, v želji za izboljšanjem osnovnih lastnosti magnetnih mehurčnih pomnilnikov, kot so večanje gostote, višanje propagacijske frekvence, poenostavljanje tehnologije izdelave čipa, uvajanju novih materialov,...
- razvoj v smeri načrtovanja in optimizacija organizacijskih struktur magnetnih mehurčnih pomnilnikov, ki poizkuša minimizirati čas dostopa, čas branja, čas bralno-pisalnega cikla,...
- sočasno poteka tudi razvoj v načrtovanju univerzalnejše in bolj integrirane podporne opreme (LSI krmilniki), kar bi omogočilo večjo kompatibilnost mehurčnih pomnilniških sistemov.

Preden spregovorimo o organizaciji magnetnega mehurčnega pomnilnika preglejmo kako se odražajo osnovne funkcije pomnilnika v magnetnem mehurčnem pomnilniku [9, 10]:

- pomnjenje: logični vrednosti 1 in 0 na določeni bitni lokaciji, sta predstavljeni s prisotnostjo oz. odsotnostjo magnetnega mehurčka na dani lokaciji,
- propagacija, dostop in organizacija: magnetni mehurček je mogoča širiti (propagirati) s pomočjo "gibljivih magnetkov", ki jih ustvarjajo z rotirajočim magnetnim poljem magneteni permalojni vzorci, katerih razporeditev določa način dostopa in organizacijo magnetnega mehurčnega pomnilnika,
- branje: za branje se izkorišča magnetorezistenčni efekt, to je spreminjanje upornosti permalojnega vzorca, ko je le-ta pod vplivom magnetnega polja, ki je posledica prehoda magnetnega mehurčka pod njim,
- brisanje: informacijo, ki je predstavljena z magnetnim mehurčkom, je mogoče zbrisati tako, da mehurček uničimo ali odstranimo z dane bitne lokacije,
- vpis: informacijo, ki je predstavljena z magnetnim mehurčkom, je mogoče vpisati tako, da generiramo nov ma-

gnetni mehurček.

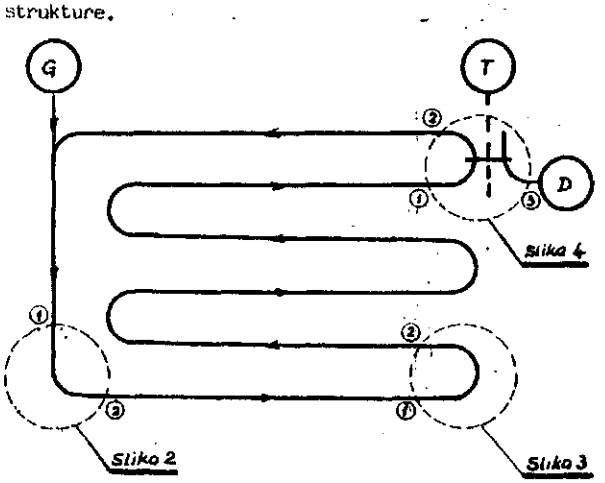
2. ORGANIZACIJA MAGNETNEGA MEHURČNEGA POMNILNIKA

Pri vseh pomnilnikih želimo organizirati shranjevanje podatkov tako, da optimiziramo število prenosnih linij, število povezav med pomnilniškim elementom in potrebno podporno opremo, kakor tudi čas dostopa in čas cikla.

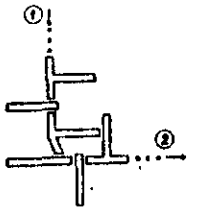
V nasprotju s koincidenčnimi pomnilniki (feritna jedra, polprevodniki, supra-prevodniki,...), kjer je osnovni mehanizem shranjevanje tokovna ali napetostna preklopna pragovna lastnost, je osnovni mehanizem uporabljen pri večini mehurčnih pomnilniških organizacij v sposobnosti voditi magnetene mehurčke od ene propaganjske sledi do druge s pomočjo ustrezno oblikovanih in magnetenih permalojnih vzorcev (prenosna vrata - transfer gate).

Vsekakor je razvoj organizacijskih struktur mehurčnih pomnilnikov rodil različne načine organizacij. Prve generacije komercialnih mehurčnih pomnilnikov (npr. 1 Kbitni Hitachijev čip, 100 Kbitni Rockwellov čip,...) so bile organizirane v obliki dolgega premikalnega registra (long shift register). Zelo kmalu pa so se pojavile zahteve po večji kapaciteti pomnilnikov, ki pa bi bili v tej organizaciji sila počasni (veliki časi dostopa in cikla). Tudi dekompozicija dolge zanke v množico med seboj ločenih krajših zank (separate short shift registers) ni dala zaželjenih rezultatov, dokler niso bile razvite učinkovitejše organizacijske strukture kot so: major/minor zanke (major/minor loops), dekoderji (decoders), koincidenčna izbira podatkovnega bloka (coincidente selection of data block) in dinamično urejeni podatkovni bloki (dynamically ordered shift registers)[1)]

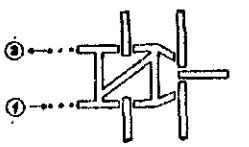
Oglejmo si nekoliko podrobneje posamezne organizacijske strukture.



slika 1.

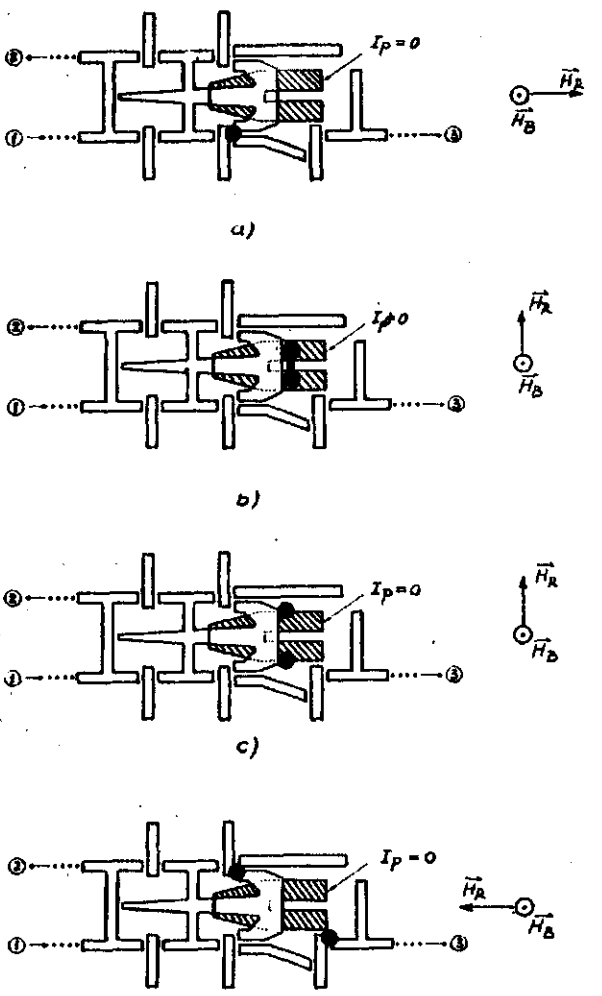


slika 2.



slika 3.

Dolgi premikalni register[1] . Takšna oblika organizacije mehurčnega pomnilnika se danes ne uporablja več, vendar jo omenjamo zato, ker vsebuje elemente, ki se pojavljajo v vseh danes uporabljenih organizacijah. Osnovna shema te organizacije je prikazana na sliki 1. Dolgo zanko oblikujejo na garnetno osnovo (nosilec mehurčkov) nanešeni permalojni propagacijski vzorci, ki v interakciji z rotirajočim magnetnim poljem povzročijo, da mehurčki neprestano krožijo v tej zanki. Spreminjanje smeri širjenja magnetnih mehurčkov je izvedeno s posebno oblikovani-



slika 4.

[1)] Naštete organizacijske strukture se uporabljajo pri tehnologijah s permalojnimi vzorci (permalloy bar technology), dočim se pri tehnologiji urejene mreže mehurčkov (bubble letice file) uporablja drugačna organizacija.

mi in nameščenimi propagacijskimi vzorci. 90° -ska sprememba smeri je prikazana na sliki 2, 180° -ska pa na sliki 3. Nedestruktivno branje informacij, ki se nahajajo v tej zanki, je mogoče s pomočjo enosmernih prenosnih podvojitvenih vrat (slika 4) in detektorja.

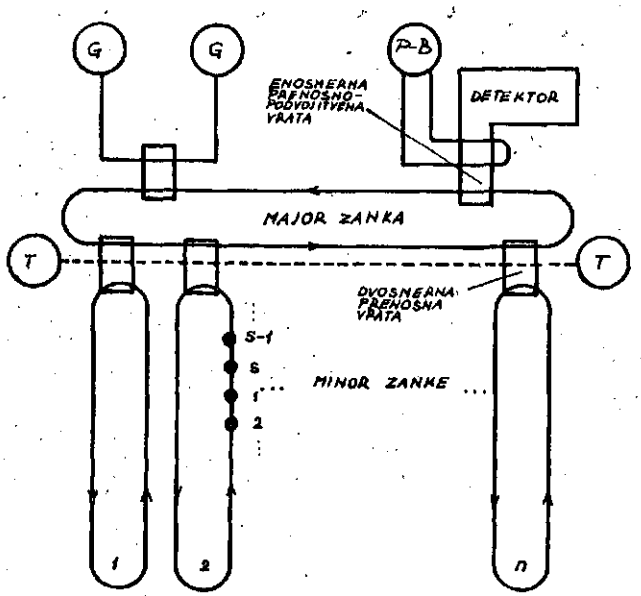
Prenos mehurčka iz zanke k detektorju omogočajo permalojni vzorci specifične oblike in pod njimi nameščena tokovna zanka. Zahteva za branje informacije pomeni prisotnost tokovnega impulza I_p (slika 4 b) v tokovni zanki, ki magnetni mehurček podvoji, "original" se vrne v zanko, "kopijo" pa sprejme detektor. Vpis nove informacije pa je mogoč preko generatorja.

Kot rečeno, ta način organizacije pomnilnika ni mogel slediti zahtevam po vse večjih kapacitetah zaradi sorazmerno dolgih časov dostopa t_a, ki so proporcionalni kapaciteti pomnilnika C in sicer:

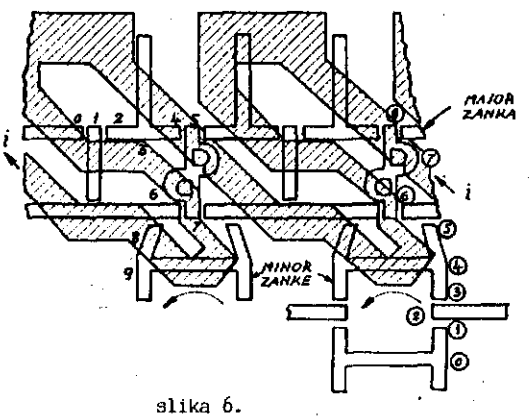
$$t_a = \frac{C}{2} T_R \qquad (1)$$

kjer je T_R perioda rotirajočega magnetnega polja H_R. Kot primer vzemimo kapaciteto mehurčnega pomnilnika 92 Kbitov in propagacijsko frekvenco 100 KHz. V tem primeru bi bil čas dostopa kar ~ o,5 sek. Seveda so takšni časi mnogo predolgi zato se takšna organizacija danes ne uporablja več.

Major/minor zanke. Pri današnjih magnetnih mehurčnih pomnilnikih najčešče zasledimo major/minor zančno organizacijo (slika 5). Informacija je shranjena v številnih paralelnih zaprtozančnih premikalnih registrih, ki jih imenujemo minor zanke (minor loop). Pod vplivom rotirajočega magnetnega polja magnetni mehurčki (nosilci informacije) sinhrono krožijo v teh zankah. Njihov položaj je v danem trenutku vselej točno določen, saj se magnetni mehurčki ob vsakem zasuku polja $\vec{H}_R$ za poln kot, premaknejo za periodo propagacijskih vzorcev p. Ta osnovni premik imenujemo korak. Operaciji vpis in branje je mogoče izvesti preko skupnega premikalnega registra, imenovanega major zanka (major loop), na katero niso vezane le vse minor zanke, temveč tudi generator (vpis informacije) in detektor (branje informacije). Povezavi detektor - major zanka in generator - major zanka sta povsem enako izvedeni kot pri organizaciji v obliki dolgega premikalnega registra (slika 4), dočim so povezave minor - major zanka izvedene s pomočjo dvosmernih prenosnih vrat brez podvajanja (slika 6). [10]

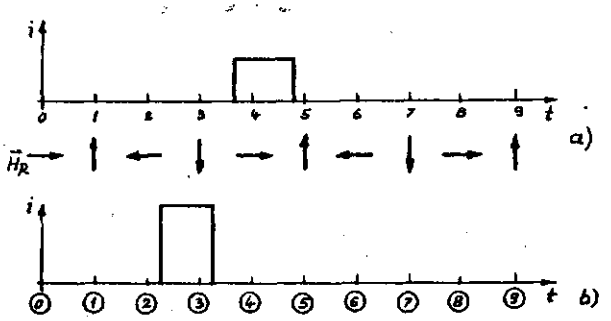


slika 5.



slika 6.

Posebnost takšnih vrat je specifičen $-sko oblikovan permalojni vzorec, dodatno pa vodimo pod major zanko spiralasto oblikovano prevodno plast, preko katere je mogoče s pomočjo tokovnega impulza izvršiti hkraten prenos iz ali v minor zanke. Opišimo najprej prenos iz major v minor zanko. Časovni potek toka i, ki je potreben za ta prenos



slika 7.

je prikazan na sliki 7 a. V času, ko je i = 0, se magnetni mehurček pod vplivom rotirajočega polja premika v major zanki. Predpostavimo, da se magnetni mehurček nahaja na poziciji 0 (slika 6) in se zaradi zasuka $\vec{H}_R$ za poln kot premakne na pozicijo 4, torej naredi en korak. Ob zasuku $\vec{H}_R$ za nadaljnih 180° bi mehurček prešel pod naslednji T vzorec, kar pa mu z ustrezno dolgim (~ $T_R/3$) in velikim (25 m A) tokovnim impulzom, ki ga pošljemo preko spiralnega vodnika, preprečimo in tako preide mehurček v pozicijo 6. Pri nadaljni rotaciji polja (i = 0) prevzame mehurček minor zanka. Prenos iz eno od minor zank v major zanko poteka povsem analogno kot v prejšnjem primeru, le da

je tu časovni potek toka i drugačen. Tokovni impulz katerega amplituda je nekoliko večja (50 mA), dolžina pa enaka, omogoči, da mehurček ob zasuku $\vec{H}_R$ za 180^0 preide iz pozicije ③ v ⑤ in ne na nasprotno stran minor zanke, kot bi bilo v primeru, če tokovnega impulza nebi bilo.

Opišimo potek osnovnih operacij in sicer: branja, brisanje in vpis informacije. Mesta magnetnih mehurčkov, ki se nahajajo na istih horizontalnih pozicijah v minorskih zankah tvorijo blok podatkov ali drugače povedano, vsak bit bloka se nahaja v drugi minor zanki. Pri branju izbranega bloka se ustrezni biti s pomočjo rotirajočega magnetnega polja sinhrono premikajo tako, da pridejo tik ob major zanko (pozicija ③ na sliki 6). Paralelni prenos vseh bitov v major zanko se izvede s pomočjo dvosmernih prenosnih vrat, tako, da v minor zankah ostanejo dotična mesta nezasedena. Bite, ki sestavljajo blok sedaj serijsko premikamo toliko časa tako, da po določenem številu korakov prvi bit bloka doseže enosmerna prenosno-podvojitvena vrata. Blok je sedaj mogoče destruktivno ali nedestruktivno brati ali pa brisati. Če želimo destruktivno branje ali brisanje, vrata magnetnih mehurčkov ne podvajajo temveč jih preko detektorja (pri brisanju detektor ni aktiviran) pošljemo v "prazno". Pri nedestruktivnem branju pa s pomočjo aktiviranih enosmernih prenosno-podvojitvenih vrat magnetne mehurčke podvojimo tako, da dobimo natančno kopijo bloka. Original, tako kot v prejšnjem primeru vodimo preko aktivnega detektorja, kopijo pa preko major zanke in dvosmernih prenosnih vrat vrnemo na prej izpraznjena mesta v minor zankah.

Če želimo vpisati novo informacijo v nezasedene lokacije v minorskih zankah, aktiviramo generator s pomočjo katerega pošljemo magnetne mehurčke v prazno major zanko. Nov blok podatkov se nato preko prenosnih vrat paralelno prenese v ustrezne lokacije minorskih zank.

Tudi pri major/minor zančni organizaciji je tako kot pri vseh ostalih organizacijah zelo pomembno, da so časi dostopa in cikla čim krajši. Za organizacijo pomnilnika, ki je shematično prikazana na sliki 5 analizirajmo čase in pogljemo v kolikšni meri nam jih je uspelo skrajšati [6]. Pomnilnik kapacitete C sestavlja n minornih zank, katerih vsaka lahko hrani s bitov informacije in pri tem velja zveza:

$$C = 2n \times \frac{s}{2} = ns \qquad (2)$$

V najneugodnejšem primeru, torej tedaj, ko je bit izbranega bloka v trenutku ko smo želeli ta blok prebrati, prešel ravno mimo dvosmernih prenosnih vrat in mora zato obkrožiti celotno minor zanko (s korakov), je čas dostopa, ki ga sestavljajo zakasnitev v minor zanki, zakasnitev med prvim prenosom in podvojitvijo (t_t) in zakasnitev detektiranja (t_d), enak:

$$t_a = s\,T_R + t_t + t_d \qquad (3)$$

oziroma:

$$t_a = \frac{C}{n}\,T_R + t_o \qquad (4)$$

kjer je T_R perioda propagacijske frekvence, to je čas v katerem mehurček naredi en korak. Kot je razvidno iz enačbe (4) večje število krajših minor zank zmanjšuje čas dostopa. Npr. 92 Kbitni (TI 0103) pomnilnik, ki ga sestavlja 157 minor zank, ki hranijo vsaka po 641 bitov, ki deluje s 100 KHz propagacijsko frekvenco in vsebuje 68 korakov od dvosemrnih prenosnih vrat do podvojitvenih vrat in 86 korakov v detektorju, ima čas dostopa v najneugodnejšem primeru $t_a = (641 + 68 + 86)\,10^{-5} = 8$ ms. To pa je kar 60-kratno skrajšanje časa dostopa glede na organizacijo v obliki dolgega premikalnega registra.

Čas potreben za branje bloka t_R, vsebuje poleg časa dostopa t_a še čas, ki je potreben za to, da se major zanka izprazni (kopija bloka se mora vrniti v minor zanke) in se glasi:

$$t_R = (\frac{C}{n} + 2n)\,T_R + t_o \qquad (5)$$

Če t_R minimiziramo glede na n, dobimo konfiguracijo pomnilnika, ki jo sestavlja $n = \sqrt{C/2}$ minor zank v katerih je po $s = \sqrt{2C}$ bitov, tako, da dobimo minimalni čas potreben za branje bloka:

$$t_{Rmin} = 2\sqrt{2C}\;T_R + t_o \qquad (6)$$

Sam pomnilnik pa ima dimenzijo 2xl ($2n \times \frac{s}{2} = 2\sqrt{\frac{C}{2}} \times \sqrt{\frac{C}{2}}$).

Čas bralno-vpisovalnega cikla t_{RW} je enak zakasnitvi v minor zanki in zakasnitvama, ki ju vnašata branje in pisanje, ter je enak:

$$t_{RW} = \frac{C}{n}\,T_R + (2n + k_1)\,T_R + (2n + k_2)\,T_R \qquad (7)$$

ali

$$t_{RW} = (\frac{C}{n} + 4n)\,T_R + t_o' \qquad (8)$$

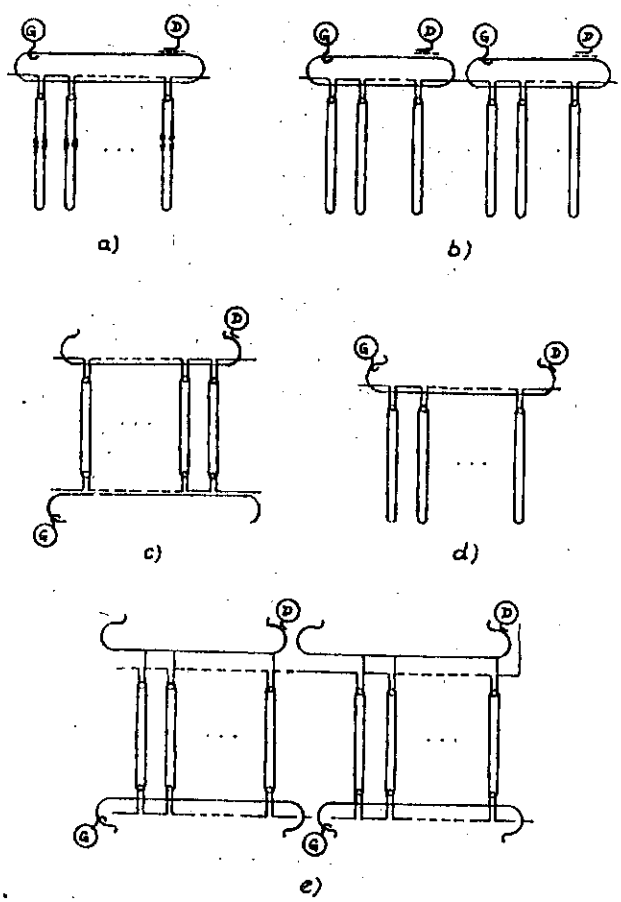
S produktom $4n\,T_R$ aproksimiramo čas potreben, da magnetni mehurček obide major zanko. Če t_{RW} minimiziramo glede na n, dobimo konfiguracijo pomnilnika, ki jo sestavlja $n = \sqrt{C}/2$ minor zank v katerih je po $s = 2\sqrt{C}$ bitov, tako, da dobimo minimalni čas bralno-vpisovalnega cikla:

$$t_{RWmin} = 4\sqrt{C}\;T_R + t_o' \qquad (9)$$

sam pomnilnik pa ima dimenzijo 1 x 1 ($2n \times \frac{1}{2} = \sqrt{C} \times \sqrt{C}$).

Vidimo, da v tej organizaciji (ena major zanka) naletimo na konflikt, saj sočasno ni možno minimizirati t_R in t_{RW},

zato so proizvajalci pristopili k različnim inačicam osnovne major/minor organizacije [1]. Nekatere od njih so prikazane na sliki 8. Slika 8 a prikazuje izvedbo kjer so uporabljene dvosmerne minor zanke, kar omogoča, da se

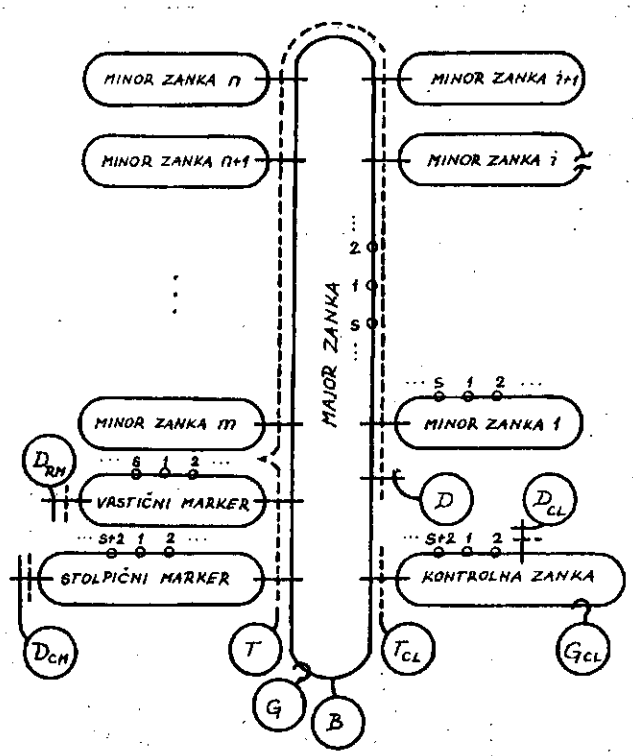


slika 8.

bit izbranega bloka približa prenosnim vratom (major zanki) po krajši poti, tako, da preide enačba (4) v obliko: $t_a = \frac{C}{2n} T_R + t_o$. Podoben efekt lahko dobimo tudi z organizacijo prikazano na sliki 8 b, saj smo z ločenim shranjevanjem lihih in sodih bitov bloka ustvarili dve podatkovni strukturi (katerih kapaciteta je C/2), ki sta vsaka zase organizirani tako kot prikazuje slika 5, le da sta povezani preko skupnega prenosa. Organizacijam na slikah 8 c, d, e je skupno to, da namesto zaključenih major zank uporabljamo odprte zanke - imenovane major trakovi in namesto dvosmernih prenosnih vrat (brez podvajanja), enosmerna oziroma dvosmerna prenosno podvojitvena vrata in jih imenujemo organizacija s podvajanjem bloka.

Ob izdelavi pomnilniškega čipa so zaradi samega tehnološkega postopka vnešene določene napake (tipično 10 napak na cm^2). Pomnilniški element je kljub temu mogoče uporabljati, če so napake takšega tipa, da so poškodovane le nekatere minor zanke, saj so že v naprej vnešene dodatne redundančne (minor) zanke, ki nadomestijo poškodovane.

Odkrivanje in nadomeščanje slabih minor zank z redundančnimi, ki je avtomatsko, je izvedeno s pomočjo specifičnih organizacij; to so v bistvu konvencionalne major/minor organizacije katerim so dodane dodatne zanke. Eno od takšnih izvedb prikazuje slika 9 in vsebujejo: s-bitno major zanko, n s-bitnih minor zank, (m - n) s-bitnih redundančnih (minor) zank, (s + 2)-bitno kontrolno zanko (control loop), s-bitni vrstični marke (row marker loop) in (s + 2)-bitni stolpični marker (column marker loop). Dodatne zanke (kontrolna, vrstični in stolpični marker) vršijo naslednji funkciji: odkrivanje defektnih minor zank in nadomeščanje le-teh z dobrimi redundančnimi ter hranjenje informacije o defektnosti minor zank, s tem pa krmiljeno branje in pisanje (kontrolna zanka); ter časovni nadzor nad branjem in pisanjem (vrstični in stolpični marker).



D_{RM} - detektor vrstičnega markerja
D_{CM} - detektor stolpičnega markerja
T - prenos major/minor in obratno
G - spontani generator
B - brisanje
T_{CL} - prenos major/kontrolna zanka
G_{CL} - generator kontrolne zanke
D_{CL} - detektor kontrolne zanke
D - detektor

Slika 9

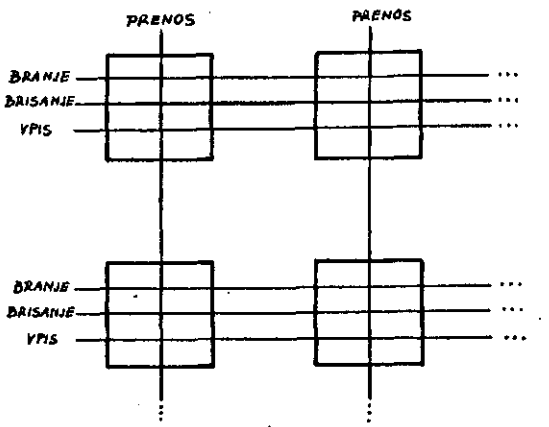
Pri začetnem testiranju pomnilniškega elementa vpišemo v vsako minor zanko na k-to mesto magnetni mehurček. Da bi si zagotovili zanesljiv vpis, po s korakih vpis ponovimo, tako, da ponovno vpišemo magnetni mehurček na k-to mesto

v vsako minor zanko. Po ponovni "zavrtitvi" minor zank prenesemo testni blok v major zanko. Vsaka morebitna napaka v minor zanki zadrži magnetni mehurček, tako, da na ustreznem mestu v testnem bloku ni prisoten magnetni mehurček. Testni blok, ki vsebuje informacijo o dobrih (mehurček prisoten) in slabih (mehurček odsoten) minor zankah, prenesemo v kontrolno zanko, kjer ga shranimo. (Ker je major zanka dolga s + 2 bitov, sta dve poziciji nezasedeni).

Pri vpisovanju informacije v minor zanke se vsak blok preko major zanke sukcesivno prenaša v minor zanke in sicer v časovnih intervalih $t_1 = s\,T_R$, $t_2 = (2s + 2)\,T_R$, $t_3 = (3s + 4)\,T_R$,..., katerih dolžina je $(s + 2)\,T_R$. Podrobneje si oglejmo dogajanje v časovnem intervalu $(0,t_1)$. V tem času generiramo n-bitni blok, $n < s$. j-ti bit tega bloka naj bi se vpisal v i-to minor zanko, za katero je, tako kot za vse ostale, v kontrolni zanki informacija o tem ali je zanka dobra ali ne. Predpostavimo, da i-ta minor zanka ni dobra, torej na i-tem mestu v kontrolni zanki ni mehurčka. Ker istočasno z generacijo j-tega bita bloka detektiramo i-ti bit kontrolne zanke, nam odsotnost i-tega mehurčka v kontrolni zanki povzroči, da se vsi predhodno generirani biti 1, 2,..., j-1 in tudi j premaknejo, tako, da bo prišel j-ti bit v i+1 minor zanko (j-1 v i+2, itd.in prvi bit v n+1, to je prvo redundančno zanko). Če je slaba zgolj i-ta minor zanka, bodo biti j+1, j+2,..., n prišli v prej namenjene minor zanke. V trenutku t_1 se bo n-bitni blok paralelno prenesel v minor zanke 1,2,..., i-1, i+1,..., n+1. V času od t_1 do t_1+2T_R pa se kontrolna zanka zavrti za dva koraka in ponovno pride v začetno stanje. S tem je vpisovanje prvega bloka končano, vse nadaljne bloke pa vpisujemo na povsem enak način. Branje poteka na podoben način, le da kontrolna zanka "krmili" detektor.

Vrstični in stolpični marker vsebujeta le po dva mehurčka, ki se nahajata na sosednjih lokacijah. V vrstičnem markerju obideta mehurčka zanko v času sT_R (to je čas, ki je potreben za vpis - branje enega bloka), dočim je ta obhod v stolpičnem markerju daljši za $2T_R$ (to je čas, ki je potreben kontrolni zanki, da se postavi v začetno stanje). S pomočjo obeh markerjev ugotavljamo začetne pozicije vsake minor zanke in mesto bitov znotraj vsakega. Kontrolna zanka nam s pomočjo dodatnega generatorja G_{CL} omogoča spreminjanje dolžine bloka l, $l \leq n$. To storimo tako, da v kontrolno zanko vnesemo l magnetnih mehurčkov.

Koincidenčna izbira podatkovnega bloka (coincidence selection of data block). Klasična major/minor zančna organizacija bi pri velikih kapacitetah zahtevala zelo dolge minor zanke in s tem dolge čase dostopa. Z razbitjem pomnilnika na več manjših enot, ki so tudi major/minor zančno organizirane s krajšimi minor zankami, smo skrajšali čase dostopa, mrtve čase in zmanjšali možnost pojava defektnih minor zank [2]. Pomnilniške enote razvrstimo v matrično obliko, tako, da so vsi elementi v stolpcu povezani s funkcijo prenos (transfer), elementi v vrstici pa s funkcijami: branje, brisanje in vpis (slika 10). Za vpis (branje, oz. brisanje) je sposobna le pomnilniška enota, ki je v presečišču aktivirane prenosne in vpiso-



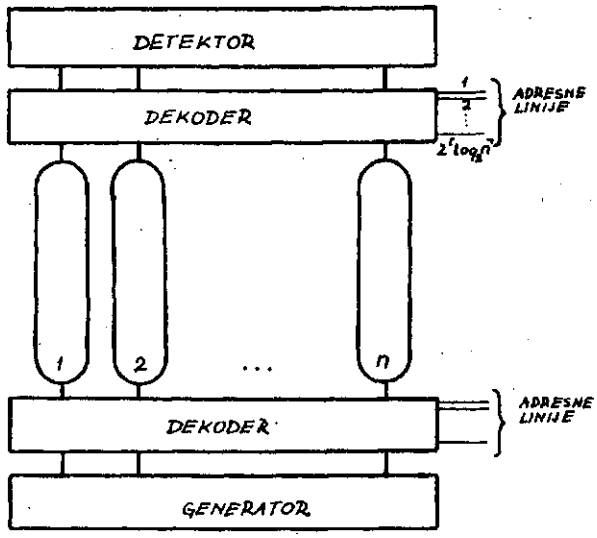
slika 10.

valne(bralne oziroma brisalne) linije, vse ostale pomnilniške enote pa niso aktivne. Takšna organizacija je nekoliko neugodna, zavoljo relativno velikega števila povezav med pomnilnikom in okolico ter bogatejše podporne opreme, kar pa vseeno ne omejuje njene uporabnosti. Zamislimo si 268 M-bitni mehurčni pomnilnik. Če bi bil organiziran kot enovita major/minor zančna struktura in bi vseboval npr. 16384 minor zank s po 16384 biti bi bil pri propagacijski frekvenci 100 KHz, čas dostopa približno 160 ms, potrebnih bi bilo 6 povezav in 4 podporna vezja. Pri organizaciji s koincidenčno izbiro podatkovnega bloka, kjer razdelimo pomnilnik na matriko 16x16, ki vsebuje 256 pomnilniških enot od katerih ima vsaka enota 1024 minor zank s po 1024 biti; bi bil čas dostopa le 10 ms, potrebnih bi bilo 81 povezav in 64 podpornih vezij.

Dekoderji (decoders). Tudi pri tej organizaciji je informacija shranjena v množici premikalnih registrov - pomnilniških zank, dodana pa je še potrebna struktura imenovana dekoder (prav tako kot pomnilniška zanka izvedena s pomočjo permalojnih vzorcev in s fotolitografskimi postopki nanešenimi prevodnimi zankami), ki je tej organizaciji dala ime (slika 11) [4, 5].

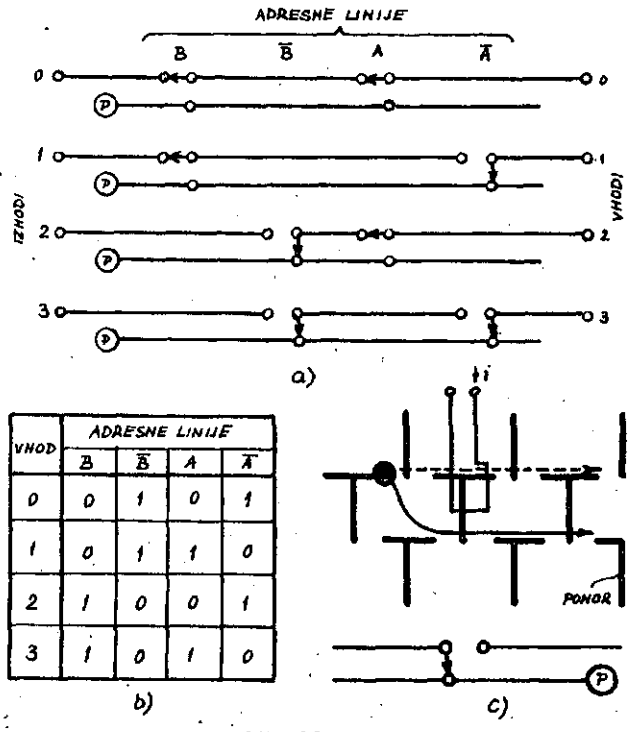
Blok informacije je shranjen v eni od s-bitnih pomnilniških zank, ki jo izberemo s pomočjo $2\lceil \log_2 n \rceil$ tih adresnih linij. Pomnilniške zanke omogočajo nedestruktivno branje, saj vsebujejo elemente, ki magnetne mehurčke podvajajo.

Če si pobliže ogledamo sam dekoder, katerega shematični prikaz in pravilnostno tabelo podaja slika 12, vidimo, da ga sestavlja n dvojnih propagacijskih poti. Prva (zgornja) pot vodi mehurčke preko $\lceil \log_2 n \rceil$ (od skupno $2\lceil \log_2 n \rceil$) stikal in jih v primeru, da so vsa stikala sklenjena pripelje od generatorja k pomnilniški zanki (vpis) ali od



slika 11.

pomnilniške zanke k detektorju (branje). Druga (spodnja) pot, pa služi kot ponor mehurčkov v primeru, da zgornja propagacijska pot ni sklenjena. Stikala so izvedena tako kot prikazuje slika 12 c. Tokovni impulz i v zanki, ki se nahaja v zgornji propagacijski poti povzroči, da magnetni mehurček preide v spodnjo propagacijsko pot kjer "izgine".



VHOD	ADRESNE LINIJE			
	B	B̄	A	Ā
0	0	1	0	1
1	0	1	1	0
2	1	0	0	1
3	1	0	1	0

slika 12.

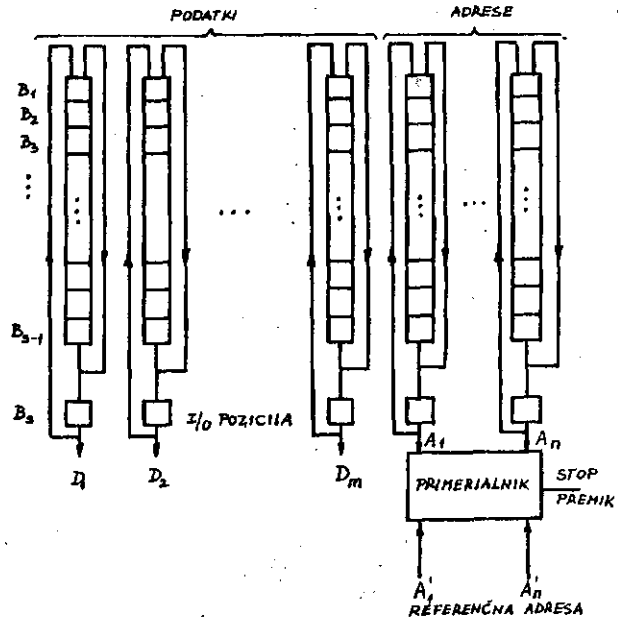
Prednost opisane organizacije je ta, da je čas dostopa t_a, ki je enak:

$$t_a = (\frac{3}{2} + 2\lceil \log_2 n \rceil)\, T_R \qquad (10)$$

mnogo krajši kot pri vseh do sedaj opisanih organizacijah; kot slabo stran pa navajamo razmeroma kompleksno podporno opremo in veliko število priključkov.

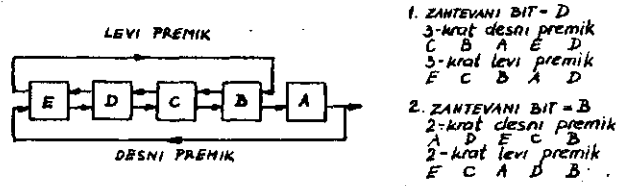
Dinamično urejeni podatkovni bloki. Za to organizacijo je značilno, da se vrstni red podatkovnih blokov, ki jih sestavlja m podatkovnih in n adresnih bitov in, ki so razporejeni vzdolž (m + n) posebno oblikovanih premikalnih registrov neprestano spreminja (slika 13). Do podatkovnih blokov, ki kot rečeno, nosijo svojo adreso, pridemo po naslednjem algoritmu [3] :

1. Adreso željenega podatkovnega bloka (referenčna adresa) pripeljemo na primerjalnik.
2. Vse premikalne registre, tako podatkovne kot adresne premikamo v desno dokler ni izpolnjen pogoj $A_i = A_i'$ za vsak i = 1,2,...n (referenčna adresa enaka adresi na I/o poziciji). Istočasno poseben števec beleži število premikov x.
3. Izbrani blok zadržimo na I/o pozicijah; in ga lahko preberemo ali prepišemo, vse ostale bloke pa premikamo za x korakov v levo.



slika 13.

Izvajanje algoritma za primer 5 bitnega registra je prikazan na sliki 14. Začetno stanje ABCD preide po izbranih bitih D in B v stanje BDACE. Premikalni registri so izvedeni tako kot pri ostalih organizacijah; magnetni mehurček pa je mogoče ujeti na I/o poziciji bodisi s posebno oblikovanimi permalojnimi vzorci in spreminjanjem smeri rotirajočega magnetnega polja $\vec{H}_R$ (slika 15 a); ali



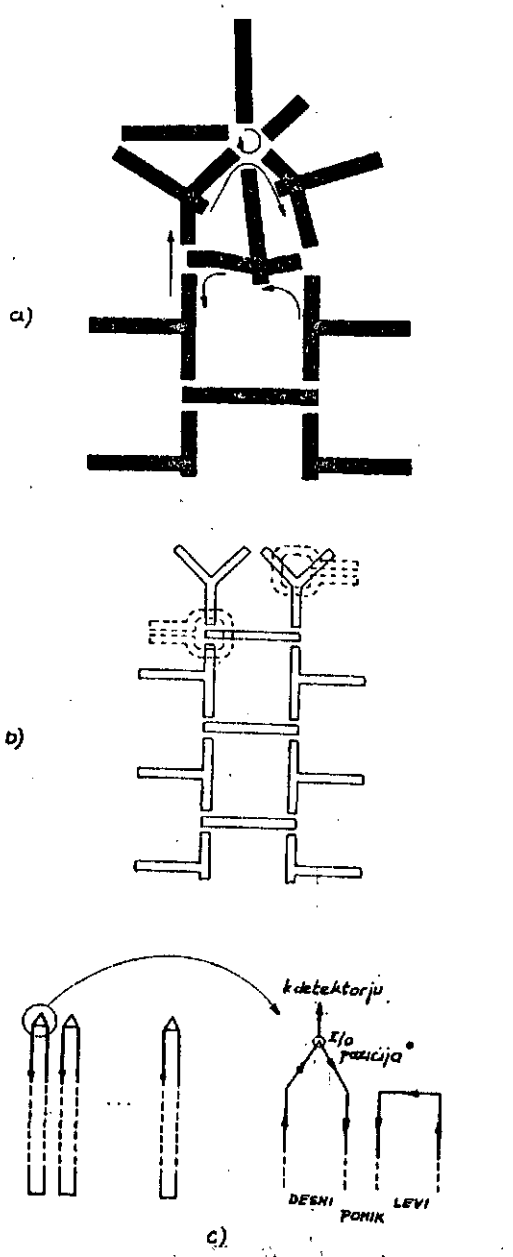
slika 14.

s prav tako posebno oblikovanimi permalojnimi vzorci in tokovnimi zankami (smer polja $\vec{H}_R$ je konstantna) (slika 15 b).

Z ustrezno strukturo te organizacije, to je velikim številom kratkih premikalnih registrov se dosežejo povprečni časi dostopa, ki so enaki:

$$\bar{t}_a \approx T_R \qquad (11)$$

kar bi pomenilo pri propagacijski frekvenci 100 KHz povprečni čas dostopa 0,01 mS.



slika 15

3. RAZMIŠLJANJE

Načrtovalca računalniških sistemov. Mehurčni pomnilniki kot masovni pomnilniki, gledano na nivoju pomnilniškega elementa in njegove organizacijske strukture, so primerljivi z diskovnimi sistemi, vendar nudijo mnogo krajše čase dostopa do podatkov, to pa že vnaša kvalitativne posega v načrtovanju celotnega računalniškega sistema. Če je mehurčni pomnilnik pridružen ostalim,od njega počasnejšim perifernim pomnilnikom, že sama razlika v časih dostopa vnaša v sistemu dodatne enote, potrebne hitre kanale, kompleksne algoritme, s tem pa je postavljena pod vprašaj prenosna funkcija sistema. Spričo tega je toliko bolj opravičljiva uporaba mehurčnih pomnilnikov in zamenjava diskovnih in ostalih počasnejših perifernih pomnilnikov z njimi; in sicer v:

- Sistemih z dodeljevanjem časa in multiprocesorskih sistemih.
  Če je centralni pomnilnik organiziran z dinamično urejenimi podatkovnimi bloki lahko hrani sistemske programe kot so: prevajalniki, zbirniki, nalagalniki in uporabniški podatki. Dodeljevanje opravil je bistveno poenostavljeno, če se opravila dodeljujejo procesorjem v centralnem pomnilniku. Manjkrat pride do konfliktov med procesorji, operacijski sistem je enostavnejši, večja je izoliranost in manjša možnost nastopa napak v podatkih.
- Samostojnih miniračunalniških sistemih z velikim pomnilnikom.
  V to skupino sodijo tako imenovani poslovni računalniki, ki ni nujno, da so hitri, pač pa zmožni izvajati veliko število programov.
- V sistemih z vmesnimi pomnilniki, virtualnimi pomnilniki in pomnilniki za shranjevanje datotek.
  Mehurčni pomnilnik je dovolj hiter (obenem pa lahko hrani veliko količino podatkov), da ga uporabimo kot vmesni pomnilnik med računalnikom in počasnejšo periferijo. Pri multi-programiranih računalnikih pa imamo opravka z dinamičnim dodeljevanjem pomnilnika in s tem se kaže potreba po načrtovanju virtualnega pomnilnika. Mehurčni pomnilniki v tej aplikaciji uspešno zadostijo potrebnim zahtevam. Kapaciteta virtualnega dela pomnilnika je lahko dovolj velika pri čemer pa so časi dostopa tako kratki, da poteka preslikava množice virtualnih adres v množico fizikalnih adres kar najhitreje. Možnost oblikovanja strukture znotraj samega pomnilniškega elementa predstavlja važno postavko pri načrtovanju arhitekture virtualnega pomnilnika, saj bistveno skrajša tabelo strani (page table) in poenostvi paginiranje. Posebno velike hitrosti dostopa pa dosežemo, če so v mehurčnih pomnilniških elementih shranjene še majhne ali srednje velike datoteke.

Načrtovalca programske opreme. V običajnih aplikacijah tretirajo načrtovalci programske opreme mehurčne pomnilnike enako kot diskovne in tračne enote. To je tudi razumljivo, saj je možno potegniti paralelo med shranjevanjem podatkov pri major/minor zančni organizaciji in shranjevanjem podatkov na diskih; kakor tudi pri organizaciji z

dolgim premikalnim registrom in magnetnimi trakovi. Vendar pa sama organizacijska struktura mehurčnih pomnilnikov nudi mnogo več, saj omogoča programirano časovno krmiljenje pomnilniškega sistema. Tako se pojavijo dodatne kvalitete, kot so: dvo ali več hitrostno delovanje pomnilnika, trenutni start-stop in dvosmerni dostop do podatkov, ki nam odpirajo možnosti snovanja specifičnih algoritmov in programov.

- Dvo (ali več) hitrostno delovanje pomnilnika (dual speed memory).
  Programirano krmiljenje hitrosti delovanja pomnilnika je kvaliteta, ki jo lahko s pridom uporabimo pri snovanju algoritmov za hitro Fourierjevo in Hadamardovo transformacijo, sortiranje podatkov, transponiranje matrik, itd.

- Trenutni start-stop (instantaneous start-stop memory). Trenutni start-stop je kvaliteta mehurčnega pomnilnika, ki omogoča simulacijo delovanja magnetnih trakov in s tem uporabe algoritmov, ki so specifični za magnetne trakove. Takšen je npr. Pavkovičev algoritem za reševanje velikih sistemov linearnih enačb. Seveda pa je izvajanje algoritmov mnogo hitrejše. Takoimenovan ciklični start-stop pomnilnik je izveden v obliki velikega števila neodvisnih premikalnih registrov (ločeni časovni nadzori) in omogoča konstruiranje zelo učinkovitih algoritmov.

- Dvosmerni dostop do podatkov (bidirectional memory). Smer pretoka podatkov v mehurčnem pomnilniku spremenimo tako, da obrnemo smer vrtenja magnetnega polja $\vec{H}_R$. Dvosmerni start-stop mehurčni pomnilniki predstavljajo pomnilniške sklade kot so: odlagalni sklad, FIFO in FILO sklad.

4. ZAKLJUČEK

Pričujoči članek je poizkušal predstaviti nekatere osnovne organizacijske strukture magnetnih mehurčnih pomnilnikov in njihov nadaljni razvoj. Razvoj organizacijskih struktur širi tudi spekter njihovih aplikacij, tako da le-ti niso uporabljivi le kot enostavni masovni pomnilniki, temveč se je področje uporabe razširilo; npr. na področje upravljalnih sistemov podatkovne baze (data base management systems), kjer mehurčni pomnilniki služijo kot nosilci podatkovne osnove (intelligent magnetic bubble memories) [7,8].

5. LITERATURA

[1] D.C.MARKHAM: Electronic Engineering, pp. 85-99, June 1979

[2] H.CHANG: Transaction on Mag., pp. 564-569, September 1972 (1972 INTERMAG Conference, Kyto, Japan)

[3] P.I.BONYHARD, T.J.NELSON: The Bell System Tech. Jour. Vol.52, No.3, pp. 307-317, March 1973

[4] H.CHANG, J.FOX, D.LU, L.L.ROSIER: IEEE Trans. on Mag. Vol. MAG-8, No.2, pp. 214-222, June 1972

[5] T.T.CHEN, T.R. OEFFINGER, I.S.GERGIS: IEEE Trans. on Mag., Vol. MAG-12, No.6, pp. 630-632, Nov. 1976

[6] H.CHANG: "Magnetic-Bubble Memory Technology", Electrical Engineering and Electronics/6, Marcel Dekker, inc., New York - Basel, 1978

[7] H.CHANG, A.NIGAM: IEEE Trans. on Mag., Vol. MAG-14, No.6, pp. 1123-1128, November 1978

[8] J.W.S.LIU, M.JINO: IEEE Trans. on Computers, Vol.C-28, No.12, pp.888-906, December 1979

[9] J.ŠILC, B.MIHOVILOVIĆ, P.KOLBEZEN: Informatica, letnik 4, 1980 - št. 4

[10] A.H.BOBECK, E.DELLA TORE: "Magnetic Bubble", North-Holland Publishing Company- Amsterdam, 1975

[11] C.K.WONG, P.C.YUE: IBM J. Res- Dev. 20, pp. 576-581, Nov. 1976

INFORMATICA 1/1981

CROMEMCO CS–3 MIKRORAČUNALO

NIKOLAJ IVANČIĆ,
BORIS KRTOLICA,
EGON ZAKRAJŠEK

UDK: 681.3–181.4 (497.1)

GRADJEVINSKI INSTITUT, ZAGREB
GRADJEVINSKI INSTITUT, ZAGREB
FAKULTET ZA NARAVOSLOVJE IN TEHNOLOGIJO, LJUBLJANA

Članak predstavlja prikaz suvremenog mikroračunala (Cromemco CS-3) sa dva važna aspekta -kao elektronički sklop i kao programsku podršku.

CROMEMCO CS-3 MICROCOMPUTER. This article is a review of a modern microcomputer (Cromemco CS-3), regarding both hardware and software properties.

1. UVOD

U okviru ovog teksta željeli smo ukratko prikazati svojstva jednog suvremenog mikrokompjutera i njegove mogućnosti. Zbog više razloga izbor je pao na Cromemco System Three. Prije svega to je računalo više puta uzastopno izabrano kao najbolje u USA (od nekoliko nezavisnih stručnih publikacija kao i udruženja korisnika računala). Uz to autori su imali priliku da više od dvije godine koriste taj sistem u veoma raznovrsnim aplikacijama. Konačno, značajan je faktori činjenica da u Zagrebu postoji generalni zastupnik Cromemca, kojem bismo se ovom prilikom željeli zahvaliti na susretljivosti.

2. OPIS SISTEMA

Cromemco CS-3 računalo je smješteno u kvadratičnu kutiju obojenu u dva tona ugodne svijetlo smedje boje koja sadrži svu sistemsku elektroniku i jedan ili dva dvostruka pogona disketa. Kućište je namjenjeno za montiranje sa strane stola ili u 19" standardnu kutiju.

Četiri vertikalna otvora na desnoj strani sistema namjenjena su za prihvat disketa od 8". Ispod svakog otvora nalazi se taster za električno izbacivanje disketa. Taster sadrži i žaruljicu koja gori kada je u toku operacija na datom disku.

Na prednjoj stranici nalazi se još samo jedna komanda -preklopnik s bravicom za uključivanje i isključivanje sistema i resetiranje mašine. Preklopnik se takodje može postaviti i u "lock" položaj, kada su sve ručne komande na prednjoj stranici onemogućene.

Pristup elektronici olakšan je time da se prednja ploča otvara kao vrata, otkrivajući tako unutrašnjost pogona za diskete i sabirnicu sa sistemskim karticama. Sabirnica je montirana u kućištu na kliznim šinama i sadrži mjesto za 21 karticu.

Jasno je da je prvenstveni cilj konstruktora CS-3 računala bio visoka performansa sistema, a ona se može postići samo s brzim komponentama. Adekvatno tome procesor je sagradjen s 8 bitnim Z-80 mikroprocesorom koji radi s taktom od 4 MHz. Uz njega u standardni sistem ugradjena je memorijska kartica s 64 K dinamičke memorije s nezavisnim osvježavanjem, s vremenom pristupa od 200 nsek,

tako da procesor može raditi punom brzinom bez ciklusa čekanja. Jedno od važnih svojstava memorijskih modula je "bank switching" koji omogućava adresiranje do 512 K memorije. Visokoj performansi sistema svakako doprinosi i PerSci 299B pogon za diskete kojega karakterizira velika brzina dosega na trag (nekoliko puta brži od pogona ostalih proizvodjača). Pogoni disketa vezani su na sistem preko 16FDC kontrolera koji omogućava zapis na obe strane diskete s dvostrukom gustoćom zapisa dajući tako 1,2 Mbyte kapaciteta po disketi. Uz sistem u standardnoj konfiguraciji nalazi se i medjusklop za priključak pisača.

Da bi se kompletirao sistem potrebno je samo priključiti terminal i pisač. Korisniku stoji na raspolaganju inteligentni ekranski terminal (modificirani Beehive DM-20) i tri pisača. Dva su Centronics matrični pisači s brzinom od 60 znakova/sek i 180 znakova/sek, te NEC-ov Spinwriter "daisy wheel" s brzinom od 55 znakova/sek.

Uz standardni sistem postoji niz opcija koje omogućavaju korisniku da izabere konfiguraciju za optimalno korištenje u željenoj primjeni (od inteligentnih podsistema, razvojnih sistema do sistema za poslovnu namjenu).

3. DODATNA OPREMA

Cromemco je poznat kao firma koja nudi veliki izbor dodatne opreme uz standardne sisteme, od periferija pa do dodatnih modula za posebne namjene. Svakako treba naglasiti da popis dodatne opreme brzo zastaruje, budući da se stalno javljaju novi moduli i to prvenstveno zahvaljujući snažnom i brojnom razvojnom timu (preko 200 ljudi), što takodje omogućava Cromemcu da prati u stopu tehnološka dostignuća, a ne rijetko biva i nosilac toga razvoja. Ukratko ćemo opisati neke od modula koji su zanimljivi radi svoje upotrebne vrijednosti i kao tehnološka rješenja.

Kolor grafički modul (SDI) je familija S-100 kartica koje se mogu smjestiti u bilo koji Cromemco kompjuter, a izlaz je na RGB TV monitor visoke rezolucije. SDI se bazno sastoji od dvije kartice. Na jednoj je logika za očitavanje memorije pomoću DMA prijenosa, a na drugoj kartici se nalazi logika za generiranje sinhro impulsa te D/A konvertori za kontroliranje boje pojedinih elemenata slike. S ovim modulom mogu se generirati i prikazivati slike s rezolucijom do 754 x 482 točke i to u 16 boja iz palete od ukupno 4096 boja. Da bi se maksimalno iskoristile prednosti SDI modula u odnosu na zajedničko korištenje memorije s procesorom (s običnom memorijom procesor je blokiran od strane DMA 45% do 95% vremena) razvijena je dinamička memorija kapaciteta 48 K s dvoja vrata te je tako zauzetost procesora smanjena na maksimalno 25% ovisno o vrsti programa koji koristi grafiku. Na ovakav način Cromemco je omogućio korisnicima svojih kompjutera da relativno jevtino dobiju kolor grafiku visoke kvalitete. Kao dokaz visokog tehnološkog potencijala Cromemca može poslužiti i činjenica da je za SDI modul dobio patent.

Jedno od vrlo zanimljivih rješenja je i Cromemcov ulazno izlazni procesor. To je u biti mikrokompjuter na jednoj pločici koji se sastoji od Z-80A mikroprocesora, 16 K memorije (RAM) te 16 K PROM memorije. Prema centralnom procesoru on se ponaša kao dvoja standardna ulazno izlazna vrata a za komuniciranje s periferijama ima tzv. C-bus koji se sastoji od Z-80 signala i nekoliko dodatnih linija za razmjenu statusa. Na C-bus se mogu priključiti razni moduli a trenutno je na raspolaganju QUADART modul s četiri

sinhrona/asinhrona serijska kanala. Univerzalnost IOP i QUADART modula očituje se u tome da se samo promjenom programa u IOP modulu može načiniti niz periferijskih kontrolera (za pogon trake, komunikacije itd.) i na taj način dobiti "inteligentnu" perifernu jedinicu i ujedno rasteretiti centralni procesor. Jasno je da na sistemsku sabirnicu može biti istovremeno priključeno nekoliko ovakovih satelitskih procesora.

Moramo ovdje spomenuti i kontroler za pogon disketa koji omogućava zapis na dvostrane diskete s dvostrukom gustoćom, a ujedno se na tom modulu nalazi i jedan serijski asinhroni kanal te monitorski program u ROM-u s ugradjenom sistemskom dijagnostikom. Pažnje je vrijedno rješenje PLL separatora podataka s diskete.

Memorijska pločica pod imenom 64KZ je dinamička memorija kapaciteta 64 K. Na njoj se nalazi i logika za selektiranje banaka, te logika za osvježavanje memorije. Za izvedbu osvježavanja memorije Cromemco je dobio patent. Treba naglasiti da je vrijeme pristupa ovoj memoriji 200 nsek te procesor radi s njom punom brzinom bez ciklusa čekanja.

4. PROGRAMSKA PODRŠKA

Osnovnu programsku podršku sistema CS-3 pretstavljaju operacioni sistemi CDOS i CROMIX.

CDOS je operacioni sistem izveden iz široko poznatog operacionog sistema CP/M (Digital Research). U suštini, to je baš CP/M operacioni sistem sa malenim proširenjima. Od periferniih jedinica taj sistem podržava diskete od 5 i 8 inča, jednostrane ili dvostrane, jednostruke ili dvostruke gustoće zapisa. Kapacitet jedne jednostrane diskete od 5 inča sa jednostrukom gustoćom zapisa iznosi 91 Kbyte, dok je kapacitet dvostrane diskete od 8 inča sa dvostrukom gustoćom zapisa 1224 Kbyte. CDOS podrzava i Winchester diskove (neizmjenjivi) sa kapacitetom od 11 Mbyte. Naravno, uz to ide i standardni asortiman perifernih jedinica kao što su terminali, printeri, čitači papirnate trake, i tako dalje.

O samoj organizaciji CDOS-a ne treba mnogo govoriti, jer je identična organizaciji CP/M-a.

CROMIX je originalan Cromemcov operacioni sistem, koji je razvijen po uzoru na UNIX (operacioni sistem na PDP-11, napravljen u Bell Laboratories Inc.). Kako je taj operacioni sistem veoma moderno koncipiran, a pretpostavljamo, da UNIX nije opće poznat, željeli bismo o CROMIX-u reći nešto više.

CROMIX je po svom karakteru "multi user - multi tasking" operacioni sistem i danas je gotovo jedini takve vrste implementiran na mikro računalima. Drugim riječima, CROMIX omogućuje istovremeni priključak više terminala (najviše šest) odnosno omogućuje paralelno izvodjenje više poslova s jednog terminala. Naravno, u ovakovim uvjetima treba imati znatno veću memoriju (po korisniku, odnosno po poslu 64 Kbyte), i treba voditi računa o tome, da je u sistemu ipak samo jedan procesor (Z-80), koji treba obaviti sav posao.

Datoteke na disketama (ili na diskovima) su organizirane u obliku drveta. To postaje naročito pogodno kod disketa većih kapaciteta jer se može informacija logički podijeliti u grane u drvu. To olakšava i zaštitu datoteka od neautoriziranog korištenja što je od posebnog značaja u multiprogramskim uvjetima.

CROMIX sadrži značajan broj pomoćnih programa za svakojake manipulacije sa datotekama (kopiranje, provjera konzistencije), za prenos datoteka CDOS -CROMIX, komunikacije izmedju pojedinih korisnika, štampanje datoteka i tako dalje. Specijalno treba istaknuti jednostavnu mogućnost "programiranja" na nivou

informatica 82

Mednarodna razstava računalniške in informacijske tehnologije

Gospodarsko razstavišče Ljubljana
10. do 14. maj 1982

Informatica '82 bo skupna sejemsko-strokovna manifestacija raziskovalnih, proizvajalnih in zastopniških delovnih organizacij, uporabnikov in strokovnih društev s področja računalništva in informatike.

Razstava Informatica je namenjena
- proizvajalcem računalniških naprav in sistemov
- uporabnikom informacijskih sistemov in
- vsem, ki uvajajo automatizacijo z uporabo računalniških sistemov.

RAZSTAVNI PROGRAM

1. Elementi naprav za obdelavo podatkov
- mikroprocesorji
- periferni procesorji
- integrirana vezja in drugi aktivni elementi
- konektorji, kabli, podnožja
- pasivni elementi

2. Enote sistemov za obdelavo podatkov
- centralne enote
- pomnilniške enote (dinamične, statične, mehanične)
- vhodno/izhodno kontrolne enote
- druge podsistemske enote
- usmerniki

3. Periferne naprave in terminali
- enote za čitanje luknjanih kartic
- enote za čitanje in luknjanje traku
- magnetnotračne enote
- enote za pogon magnetnih kaset
- diskovne enote
- pogoni za gibke diske
- optični čitalniki znakov
- čitalniki rokopisov
- znakovni, vrstični in bločni tiskalniki
- video terminali

4. Sistemi za obdelavo podatkov
- sistemi za splošno obdelavo podatkov
- pisarniški sistemi
- razvojni sistemi
- poslovni sistemi
- laboratorijski sistemi
- procesni sistemi
- vojni sistemi
- osebni in domači sistemi
- sistemi za zajemanje podatkov

5. Programska oprema

- osnovna in sistemska programska oprema
- uporabniška programska oprema za poslovne sisteme
- uporabniška programska oprema za vodenje procesov
- komunikacijska programska oprema
- uporabniška programska oprema za tehnične in znanstvene namene

6. Aplikacije računalniških sistemov

- telekomunikacijski sistemi
- vodenje elektroenergetskih sistemov
- zajemanje podatkov
- računalniška grafika
- robotika
- vodenje tehnoloških procesov
- bančni sistemi

7. Naprave za zbiranje in prenos podatkov

- modemi za prenos podatkov
- multiplekserji
- kontrolne in merilne naprave
- naprave za komutacijo

8. Oprema za proizvodnjo računalnikov

- kazalni merilni instrumenti
- elektronski merilni instrumenti
- logični analizatorji
- sistemi za načrtovanje vezij
- sistemi za izdelavo tiskanih vezij
- sistemi za testiranje tiskanih vezij
- sistemi za proizvodnjo integriranih vezij

9. Mediji za zapis podatkov

- magnetni trakovi
- magnetni diski, gibki diski
- magnetne kasete
- formularji na neskončnem traku
- papirni trak

10. Strokovna literatura

- knjige
- revije in časopisi
- uporabniška dokumentacija

Simpozij INFORMATICA 82 je mednarodni simpozij za računalniško tehnologijo in probleme informatike, ki bo v organizaciji Slovenskega društva Informatika, Elektrotehniške zveze Slovenije in Gospodarskega razstavišča v dneh od 10. do 14. maja 1982, obravnaval pereče problematike s področja računalniških znanosti, tehnologije in uporabe. Sodelovanje z mednarodnimi strokovnimi organizacijami bo zagotovilo visoko kakovost referatov in posvetov, kjer bodo sodelovali ugledni mednarodni in domači izvedenci.

Tako bosta razstava in simpozij Informatica '82 srečanje strokovnjakov, proizvajalcev, uporabnikov in drugih interesentov z računalniške informacijske panoge.

INFORMACIJE IN PRIJAVE:
Gospodarsko razstavišče Ljubljana
61000 Ljubljana, poštni predal 113, Titova 50
telefon (061) 311-022, 310-930, 311-232
telex 31 127 gr yu

pozivanja programa (kao JCL na IBM računalima, CCL na CDC računalima ili SSG na UNIVAC-u).

CROMIX dozvoljava tzv. "I/O redirection" tako, da sve ulazno izlazne jedinice tretira ekvivalentno, odakle proizlazi mogućnost, da se na primjer rezultati jednog posla, koji se normalno prikazuju na ekranu, usmjere "na ulaz" nekog drugog posla.

Kao veoma značajnu komponentu CROMIX-a treba spomenuti CDOS simulator. Taj omogućuje, da se programi, koji su napisani za CDOS operacioni sistem, koriste pod CROMIX-om. Budući je CDOS kompatibilan sa CP/M operacionim sistemom, a postoji izuzetno velik broj programa, razvijenih za CP/M, sva ta oprema postaje dostupna ne samo CDOS-u već i CROMIX-u.

Od te programske opreme spominjemo samo najznačajnije primjere. Djelomično su razvijeni kod Cromemco Inc., a djelomično kod drugih softverskih kuća pa ih je Cromemco otkupio ili su to pak programi, razvijeni potpuno nezavisno od Cromemca.

Osnovna grupa tih programa su editori, programi za unošenje programa ili drugih tekstova u kompjuter. Cromemco raspolaže sa veoma sposobnim SCREEN editorom (koji je usput korišten prilikom pripremanja ovog teksta). Od ostalih editora treba spomenuti mozda najjači postojeći -Word Star firme MicroPro, koji je istovremeno i veoma sposoban tekst procesor. Od iste firme postoji i program Data Star, veoma prikladan sistem za unošenje podataka na osnovu unapred sastavljenog formulara (Data Entry).

Efikasan makro asembler, Macro Assembler, može se koristiti za pisanje programa na nivou Z-80 procesora. Pored toga postoji niz prevodioca za sve moguće više programske jezike.

Tu treba prije svega spomenuti veoma potpun FORTRAN IV, COBOL i PASCAL. Svi su ti prevodioci veoma pristojno dokumentirani. COBOL izmedju ostalog sadrži i module koji omogućavaju jednostavno pisanje programa za unošenje podataka.

Najznačajniji produkt ove vrste je prevodilac za jezik C (Vidi B.W.Kernighan, D.M.Ritchie, The C Programming Language, Prentice Hall, 1978). C je bio na početku razvijen u Bell Laboratories pod UNIX-om, tačnije, UNIX je napisan u C-u (???). To je jezik, veoma pogodan za pisanje sistemskog softvera, a naravno, njegova se primjena nikako ne ograničava na to područje. Izražajnu snagu jezika najbolje potvrdjuje činjenica, da je i CROMIX u velikom djelu napisan u C-u. Po svojoj strukturi jezik još najviše liči na PASCAL, ali nema toliko strogo provjeravanje tipova što uveliko olakšava baš pisanje sistemskog softvera, a s druge strane olakšava i pogreške kod programiranja. U jeziku postoji niz konstrukcija koje se po svojoj prirodi dadu vrlo efikasno prevesti u mašinski kod. Posljedica je efikasno prevodjenje u smislu, da je prevod kratak i efikasan. (Samo prevodjenje je dosta sporo, ali to je djelimično i posljedica dozvoljenih konstrukcija u jeziku, koje ne dozvoljavaju prevodjenje u jednom prolazu.)

Od ostalih prevodioca mozemo spomenuti još 32K Structured Basic, 16K Extended Basic, RPG II i UCSD Pascal sistem. UCSD Pascal je sistem sam za sebe. Ima svoj operacioni sistem, svoju organizaciju datoteka i svoj editor. To mu donekle smanjuje upotrebljivost ali je to vanredno sredstvo za učenje programiranja. Zatvorenost od okoline vanredno pojednostavljuje učenje.

Značajan program u toj kategoriji je Overlay Linker. Kao Linker to je program

neophodan za slaganje programa nakon što su prevedeni asemblerom ili prevodiocem, a kao Overlay Linker omogućava jednostavno pisanje programa, za koje ne bi očekivali, da ih se može implementirati na mikro kompjuteru. Primjer takvog programa je STRESS (Structural Engineering System Solver), namjenjen gradjevinskim inženjerima za proračun statičkih konstrukcija (veličina programa je 125 Kbyte).

Od drugih produkata valja spomenuti Text Formatter, program, koji je korišten za formatiziranje ovog teksta u pristojan oblik, veoma sposoban sistem za diagnosticiranje grešaka u kompjuteru, programi za sortiranje i drugi.

Uz to treba još jednom napomenuti gotovo neiscrpivu zalihu produkata koji stoje na razpolaganju korisnicima CP/M operacionog sistema i gdje se može pronaći niz veoma korisnih programa po svačijem ukusu.

Sve u svemu, programska podrška ovog mikro komputera ne treba da se srami pred programskom podrškom "ozbiljnog" komputera od "milijun dolara". Zapravo sa istom se ozbiljnošću može ustvrditi da takav zaključak vrijedi globalno, uzevši u obzir svojstva računala kao elektroničkog sklopa i programske podrške. Naravno, takva se tvrdnja treba zasniva na odnosu performansa / cijena.

INFORMATICA 1/1981

MIKRORAČUNALNIŠKO KRMILJENJE AVTOMOBILSKEGA MOTORJA

B. ŠVAB

UDK: 681.3–181.4:621.432.3

ISKRA TEA, KRANJ

Od avtomobilskega motorja se zahteva majhna poraba goriva in čim manjše onesnaževanje okolja. Ti dve zahtevi sta si nasprotni, zato je mogoča zadovoljiva rešitev le ob natančnem krmiljenju motorja. Avtomobilski proizvajalci danes v ta namen vse bolj uporabljajo mikroračunalnike. Ta članek opisuje nekatere probleme in rešitve v zvezi s tem.

MICROCOMPUTER ENGINE CONTROL: The demands for car motor are good fuel economy and small pollution. They are contrary each other so good solution is possible only with precise engine control. Avtomakers are more and more using microcomputers for this purpose. This article describes some problems and solutions from this area.

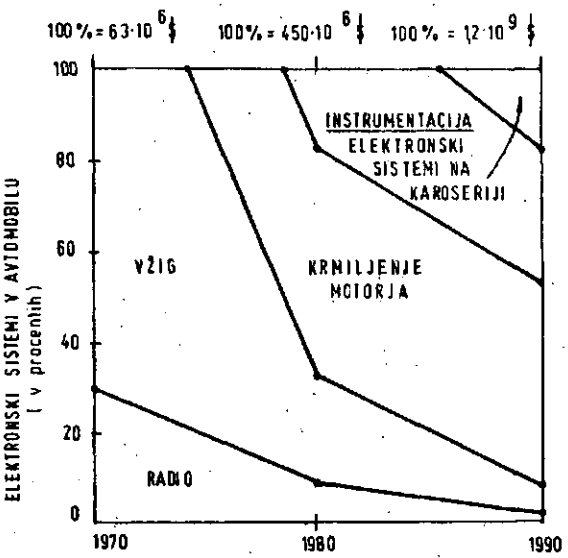
UVOD

1. UVOD

Uporaba polprevodnikov v avtomobilih ni popolnoma nov pojav. Že pred dvajsetimi leti so uporabljali diskretne tranzistorje v avtoradijih, nadaljevalo se je z usmerniki za alternatorje, elektronskimi regulatorji, napravami za vžig, krmilniki brisalcev itd. Danes pa uporaba elektronike, predvsem mikroprocesorjev in LSI vezij zaradi velike integracije funkcij na majhnem prostoru, zanesljivosti, cenenosti in nezahtevnosti vzdrževanja skokovito narašča, saj z njimi avtomobilski proizvajalci rešujejo probleme v zvezi z onesnaževanjem zraka in porabo goriva in izboljšujejo mnoge druge lastnosti. Elektronika se danes intenzivno uporablja na naslednjih področjih:

- krmiljenje motorja,
- informacijski center,
- naprave za komuniciranje,
- krmiljenje prenosa,
- naprave proti blokiranju koles,
- klimatske naprave,
- varnostni sistemi,
- testiranje in identifikacija okvar.

Delež posameznih elektronskih sistemov v celotni avtomobilski elektroniki prikazuje slika 1.



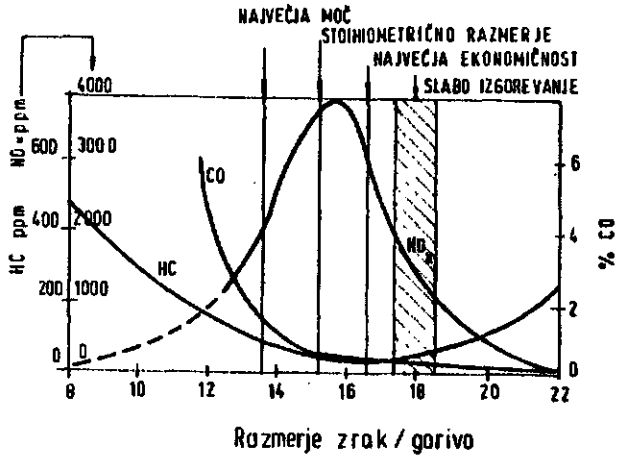
Slika 1: Porazdelitev tipa elektronskih sistemov v avtomobilu

2. ŠKODLJIVE SNOVI IN PORABA GORIVA

Avtomobilski motor krmilijo, da znižajo količino škodljivih snovi v izpušnem plinu in zmanjšajo porabo goriva. Pri manjših motorjih tem zahtevam zadostijo z raznimi mehanskimi rešitvami, pri večjih motorjih pa je to skoraj nemogoče, zato se množično uporabljajo razni dodatni sistemi, ki jih krmili elektronika.

Škodljive snovi v izpušnem plinu so dušikovi oksidi (NO_x), ogljikovodiki (HC) in ogljikov monoksid (CO). Omejitve za leto 1981 v ZDA so 0,41 g HC/miljo, 3,4 g CO/miljo in 1 g NO_x/miljo. Količina teh snovi je odvisna od razmer-

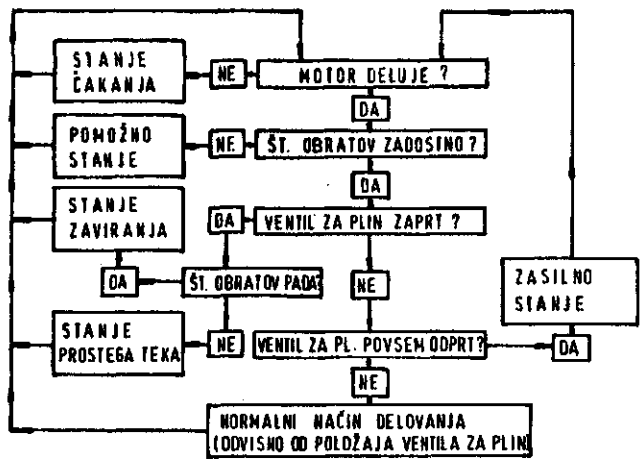
ja zraka in goriva v zmesi, ki vstopa v motor (slika 2). Najugodnejše je visoko razmerje, toda ker je tedaj poraba goriva večja, se izbere kompromisna rešitev.



Slika 2: Količina škodljivih snovi v izpušnem plinu

Težava pri odpravljanju škodljivih snovi je v tem, da postopki, ki odpravijo HC in CO, povečajo količino NO_x, ki se tvori pri visokih temperaturah (nad 3500°C).
Najpomembnejše so naslednje metode:
- katalitični pretvornik,
- uporaba revne zmesi,
- nastavitev prostega teka,
- zakasnitev iskre,
- recirkulacija izpušnih plinov,
- dodajanje zraka produktom izgorevanja.

Najbolj učinkovit je katalitični pretvornik (Three way cathalitic convertor). V posebni komori se s pomočjo katalizatorjev (rodij, platina) oksidirajo CO in HC ter reducirajo NO_x. Postopek je najbolj učinkovit tedaj, ko je razmerje zraka in goriva 15:1 (stoihiometrično razmerje). Poraba goriva se poveča za okrog 5%, količina škodljivih snovi pa se zmanjša do predpisane meje.
Porabo goriva zmanjšajo z vbrizgavanjem goriva, z optimalnim krmiljenjem posameznih režimov delovanja motorja in s krmiljenjem prenosnega razmerja.
Za krmiljenje vseh teh procesov so v preteklosti uporabljali analogne električne sisteme. Vedno strožji predpisi so zahtevali natančnejše krmiljenje, zato so uvedli mikroračunalnike, ki so dovolj hitri, natančni in prilagodljivi na različne razmere (drugi motorji, gorivo). Vsak režim delovanja motorja ima svoje lastnosti in zahteva specifičen pristop pri krmiljenju. Na sliki 3 je diagram poteka delovanja motorja, ki se upošteva pri elektronskem krmiljenju.



Slika 3: Diagram poteka delovanja avtomobilskega motorja

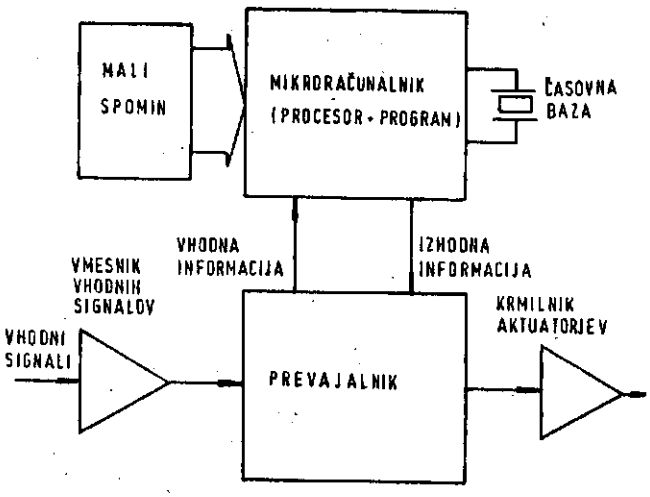
Na osnovi treh vhodnih signalov (vključenost motorja, število obratov motorja, pozicija pedala za plin) se določi eno od šestih možnih stanj motorja, za katero se privzame ustrezna programska procedura.

3. TIP MIKRORAČUNALNIKA

Ker se mikroračunalniki na avtomobilskem področju uporabljajo razmeroma kratko obdobje in zahteve še niso točno definirane, se še ni pojavil standardni mikroračunalnik. Ni še jasno, ali so primernejše obstoječe komponente, ali pa posebej za to področje razvite. Mnogo problemov se lahko reši z obstoječimi mikroračunalniki, na drugi strani pa krmiljenje motorja zahteva rutine množenja in deljenja in tu so boljši posebni mikroračunalniki. Oba pristopa sta prikazana na sliki 4.

Klasični bus orientirani mikroračunalnik dopušča več pristopov krmiljenja, posebni avtomobilski mikroračunalnik pa predstavlja optimalno rešitev glede lastnosti in cene. Glavne avtomobilske firme so v različnih fazah razvoja mikroračunalniškega krmiljenja. Nekatere uporabljajo standardne komponente, druge pa posebej za avtomobilsko področje razvite. Razvoj gre v smeri zadnjih. Za avtomobilsko industrijo razviti mikroračunalniki naj bi se uporabljali skozi daljše obdobje z eventualnimi spremembami programov v spominu. Računajo, da bo množična proizvodnja teh elementov opravičila v razvoj vložene stroške.

V prvi fazi uporabe mikroračunalnikov sta se uporabljala dva ali en sam procesor kot centralna procesna enota, ki je bila z linijami povezana z množico senzorjev in aktuatorjev. Danes, v drugi fazi gre razvoj proti porazdeljenemu procesiranju. Centralni del sicer vodi



Slika 4a: Avtomobilski mikroračunalnik

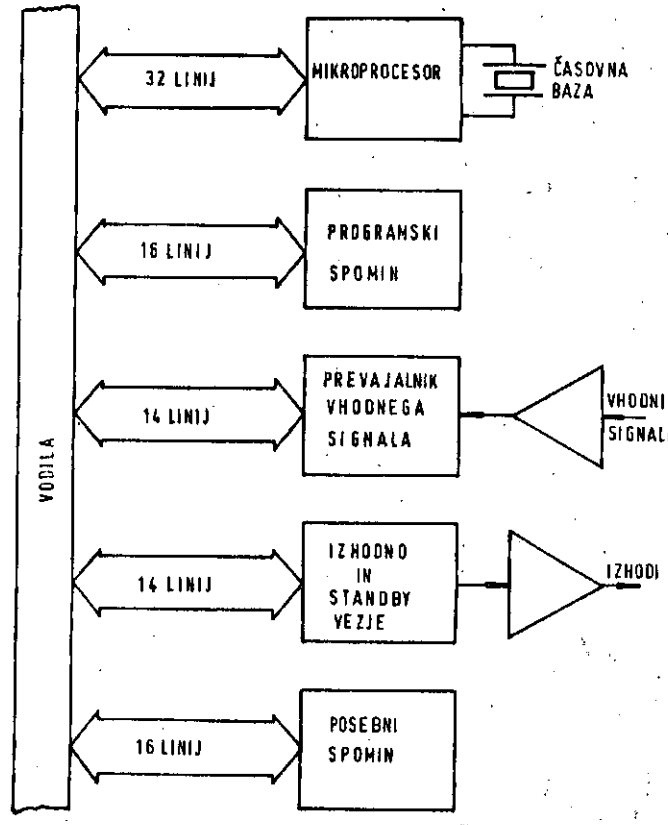
več po avtomobilu razporejenih polavtonomnih satelitskih procesorjev, ki pa so sposobni opravljati določene funkcije samostojno. Na ta način se zmanjša ožičenje in poveča zanesljivost, saj sistem deluje tudi, če je centralni del v okvari. Centralizirana arhitektura je enostavnejša in cenejša, porazdeljena pa zanesljivejša, hitrejša a tudi dražja.

4. KONCEPT KRMILNIKA MOTORJA

Zahteve za krmilnik še niso točno definirane, zato se teži k prilagodljivosti sistema na različne zahteve. Avtomobilska in elektronska industrija iščeta najboljšo rešitev s tehničnega in ekonomskega stališča. Zaradi netočnih zahtev so sistemske rešitve težke. Pri množični proizvodnji, kjer mora biti vsaka komponenta zanesljiva, preprosta in cenena, se je težko odločiti, kateri krmilnik naj se uporabi. Na sliki 5 je prikazan konceptualni pristop k izvedbi krmilnika. Pričakujejo, da se bo v prihodnosti uveljavil ne samo na avtomobilskem ampak tudi na drugih področjih.

Naloge, ki jih opravlja, so časovna nastavitev iskre, doziranje plinske zmesi in krmiljenje raznih postopkov, ki znižajo količino škodljivih snovi v izpušnem plinu. Stanje motorja določa lo vhodnih signalov v obliki napetosti ali frekvence. Signali so izpostavljeni raznim motnjam, zato so linije oklopljene, signali pa se tudi filtrirajo.

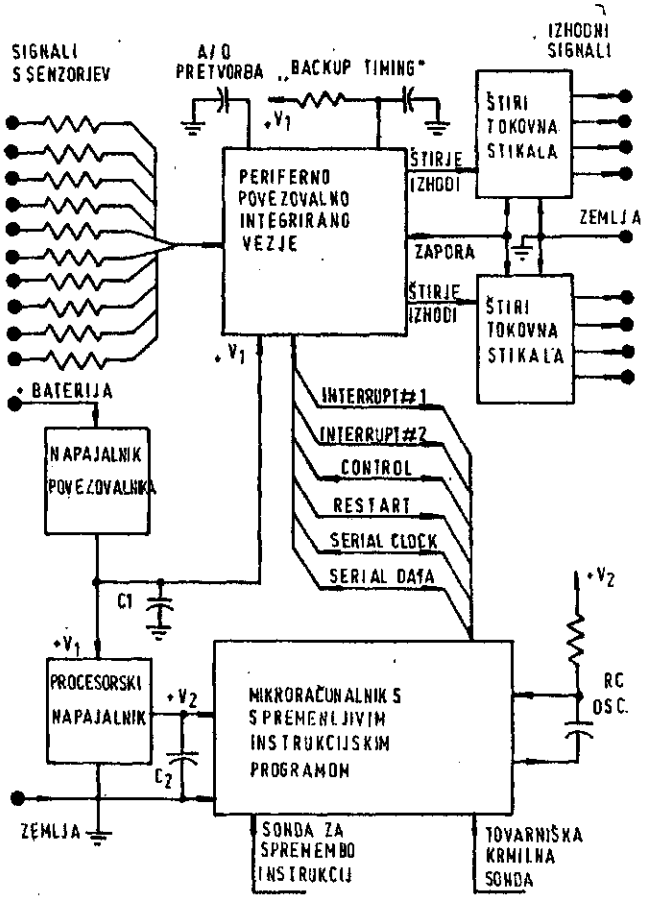
Krmilnik sestavljajo:
- vhodno prilagojevalno vezje,
- izhodna tokovna signala,
- povezovalnik,
- mikroračunalnik,
- napajanje in časovno vezje.



Slika 4b: Klasični mikroračunalnik

Elektromehanske aktuatorje krmilijo izhodna tokovna stikala, grupirana v skupine po štiri z vgrajeno zaščito proti motnjam in posebnim zapornim vezjem, ki avtomatično preklaplja izhodne aktuatorje ob posebnih stanjih (vključitev napajanja, izpad mikroračunalnika). Signali se delno obdelajo v povezovalniku (periferno povezovalno vezje). Njegova osnovna naloga je povezava med analognim realnim in digitalnim mikroračunalniškim svetom, poleg tega pa krmili osnovne funkcije motorja v primeru izpada mikroračunalnika. Vezje je v primerjavi z mikroračunalnikom mnogo bolj enostavno in eventualne spremembe tega vezja ne bi predstavljale velike ekonomske ovire. Po drugi strani pa naj bi bilo vezje univerzalno in prilagodljivo na množico obstoječih in bodočih senzorjev. Periferni povezovalnik sestavljajo naslednje enote:
- vezje za prireditev I/O signala,
- A/D pretvornik,
- vezje za povezavo z mikroračunalnikom,
- vezje za nadziranje izhodnih signalov.

Vezje povezovalnika je izdelano v digitalni in analogni tehniki. Najboljše lastnosti zagotavlja nizko impedančna bipolarna polprevodniška tehnologija.
Mikroračunalnik ima procesor s poudarkom na tistih lastnostih, ki so pomembne za krmilje-



Slika 5: Primer krmilnika

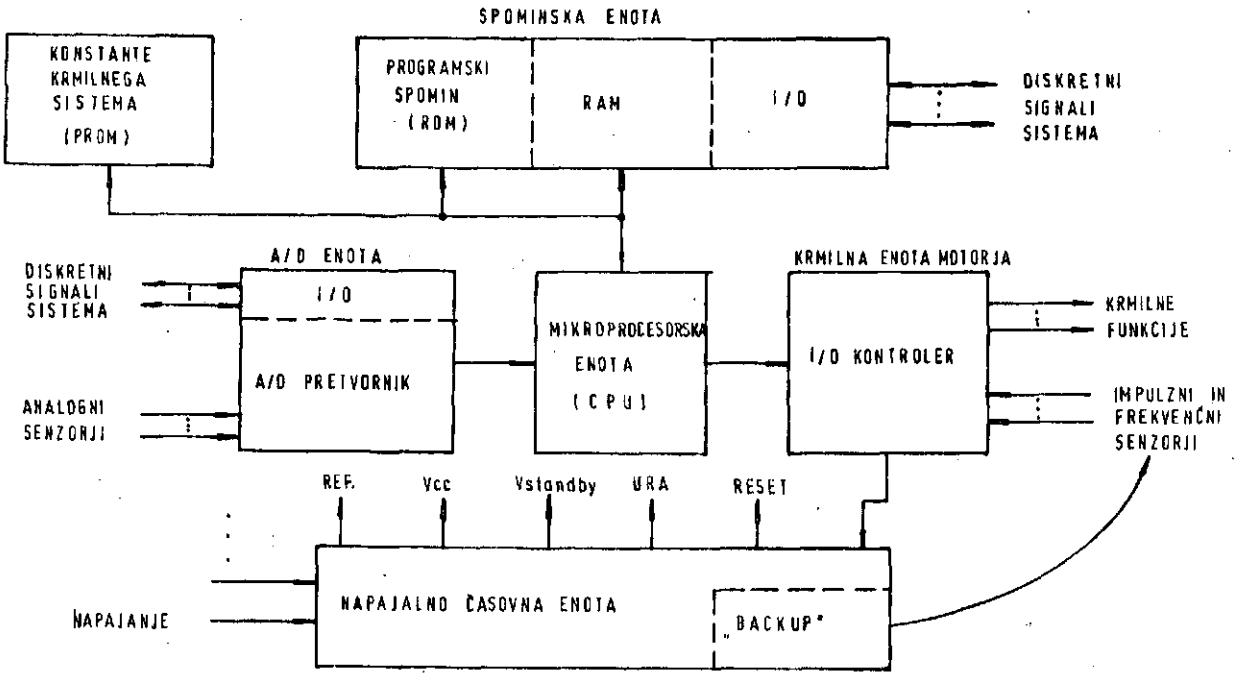
nje avtomobilskega motorja. Informacija je 8 bitna. Instrukcijski cikel traja od o,5 do 1,5 /us z dopustno toleranco 5%, zato se za uro uporablja enostavni RC oscilator. Napajalna napetost je v območju od 3,5V do 5,25V s srednjo vrednostjo pri 4,5V. Predvidena poraba mikroračunalnika je 55mA, za regulacijo napetosti se uporablja Zener dioda. Če ima tudi povezovalnik svojo napajalno napetost, je regulacija dvostopenjska, stopnji pa sta vezani serijsko tako, da padec napetosti najprej zazna povezovalnik in signalizira mikroračunalniku, da ustavi izvrševanje programa na predpisan način.
Zunanji signali pridejo na mikroračunalnik preko povezovalnika. Če je taka konfiguracija prepočasna, se nekatere signale preko dodatnega vezja poveže direktno na mikroračunalnik. Med mikroračunalnikom in povezovalnikom poteka:
- restart mikroračunalnika,
- izbira in prenos vhodnih podatkov,
- izbira in prenos izhodnih podatkov,
- povezava s tovarniškim testirnim sistemom,
- prenos instrukcijskega programa v spomin,
- več prekinitvenih signalov.

V spominu sta dve vrsti programa. V času izdelovanja in sestavljanja krmilnika je v njem testni program, v času montaže na določeno vozilo pa se v spomin vpiše instrukcijski program. Program se vpisuje preko linije s pomočjo dveh krmilnih signalov na sondah.
Opisano mikroračunalniško vezje zahteva 12 priključkov, po dva za napajanje, lokalni oscilator in reprogramiranje in šest za izmenjavo podatkov.
Zanesljivost delovanja krmilnika se poveča na tri načine:
- z rezervnimi elementi,
- s self test elementi,
- s self test programom.

Metoda z rezervnimi elementi je najdražja in je namenjena za kritične motorske funkcije (vžig, vbrizg goriva). Pri drugi metodi gre za odkrivanje napak v normalnem poteku programa. V programsko krmilno zanko se vstavi self test instrukcija. V primeru, da se program ne izvaja pravilno, se self test instrukcija ne izvrši in posebno vezje generira napako. Izhodi krmilne zanke se postavijo na nominalne nivoje, program se postavi na začetek. Če napaka ni prehodnega značaja, se uporabijo rezervni elementi. Po metodi self test programa se v primeru izpada določene komponente npr. senzorja programsko nadomesti dejanska vrednost s povprečno. Npr. če senzor hladilne tekočine odpove, temperaturo motorja določi mikroračunalnik, hkrati pa se napake signalizira na armaturni plošči.
Za testiranje se lahko uporabi tudi posebna testna oprema, v bistvu računalnik, ki se ga ob testiranju priključi na mikroračunalnik v avtomobilu.
Resen problem v razvoju avtomobilskih sistemov predstavlja pomanjkanje kvalitetnih in cenenih senzorjev. Podobni senzorji se uporabljajo v letalstvu in industriji toda za množično proizvodnjo v avtomobilski industriji so predragi. Obstaja mnenje, da se do cenenih senzorjev lahko pride le preko tesnejšega povezovanja med polprevodniško in avtomobilsko industrijo, ker ima prva dobro razvito tehnologijo množične in poceni proizvodnje elektronskih delov. V avtomobilu se uporabljajo naslednji senzorji:
- temperaturni senzorji,
- pozicijski senzorji,
- senzorji pritiska,
- pretočni senzorji,
- senzorji sestave snovi.

5. PREGLED KRMILNIKOV

Prvi veliki proizvajalec, ki je množično uporabil elektronski krmilnik, je bil General Motors. Skoraj pet milijonov vozil na bencin z letnikom 1981 bo imelo krmilnik CCC (Computer Command Control), ki ga je razvila firma Delco.



Slika 6: Sistem GMCM

Jedro sistema je 8 bitni procesor Motorola 6802 z različnim instrukcijskim programom v ROM za različne splikacije. Za leto 1982 so razvili zmogljivejši sistem GMCM (General Motors Custom Microcomputer). GMCM je sestavljen iz petih bus kompatibilnih LSI vezij (slika 6). To so mikroprocesor, krmilna logika, spomin, A/D pretvornik in napajalno časovna enota. Mikroprocesor je Motorola 6802 modificiran za opravila v zvezi s krmiljenjem motorja. Ima lo instrukcij več kot običajni M6800. To so med drugim 8 bitno množenja, 16 bitno seštevanje in odštevanje ter akumulatorske instrukcije. Poleg tega je izvršilni čas 32 izbranih instrukcij reduciran. Mikroprogramirani kontroler delno obdela vhodne signale in razbremeni mikroprocesor. Spomin ima 4 kbyt ROM in 128 byt RAM (od tega 64 byt nonvolatile) ter 8 programabilnih I/O vrat. A/D pretvornik v CMOS je v 40 pinskem ohišju, ima 16 kanalov in čas pretvorbe 300 μs. Napajalno časovna enota je v TTL, ECL in I^2L tehnologiji. Vsebuje sistemsko uro, regulacijo moči, "Backup" vezja in reset timing. Deluje v temperaturnem območju od -40°C do 85°C in pri 5V porabi 2,5W.

Raziskave pri Fordu so se začele leta 1960 in so se osredotočile najprej na analogno, potem pa na digitalno elektroniko. Prišli so do zaključka, da lahko dosežejo optimalno krmiljenje samo z interaktivnim krmilnim sistemom, ki simultano krmili več procesov. Prvi Fordov digitalni elektronski krmilni sistem (1978) je bil EEC1 (Electronic Engine Control) in je krmilil vžig in recirkulacijo izpušnih plinov. Ob odličnih lastnostih motorja so zadostili takratnim zahtevam glede onesnaževanja in porabe. Drugi sistem (1978) je bil ECU-A (Electronic Control Unit). Krmilil je razmerje zrak/gorivo in to odprto in zaprtozančno. Zaprtozančno krmiljenje poteka na osnovi odtipavanja količine kisika v izpušnem plinu, tako da je učinek pretvornika, kjer se nevtralizirajo škodljive snovi, maksimalen. Sistem EEC2 je sinteza omenjenih sistemov ob sodelovanju Toshibe, Texas Instruments in Fordovega razvojnega inštituta. Sistem sestavlja 12 bitni mikroprocesor firme Toshiba, šest LSI vezij in 120 diskretnih komponent ter je sposobnejši, enostavnejši in cenejši od predhodnikov. V letu 1980 se je pojavila izboljšana verzija EEC2-EEC3. Tokrat je v krmilniku uporabljen Motorolin mikroprocesor 67002 in štiri LSI vezja. Sistem ima 11 vhodov in 10 izhodov, 8 kbyt spomina in je hitrejši od predhodnika. Za manj zahtevne aplikacije je Ford izpopolnil ECU-A. Novi sistem ima mikroračunalnik Intel 8048. Za naslednje obdobje razvijajo EEC4 z Intelovim 16 bitnim mikroračunalnikom 8061. Chrysler uporablja CMOS mikroprocesor COSMAC 1802 firme RCA. Od vseh sistemov v ZDA je najenostavnejši in relativno najcenejši, krmili pa vžig, zaprtozančni vplinjač in recirkulacijo izpušnih plinov. V bodoče bodo uporabili hitrejši in zmogljivejši mikroprocesor 1804.

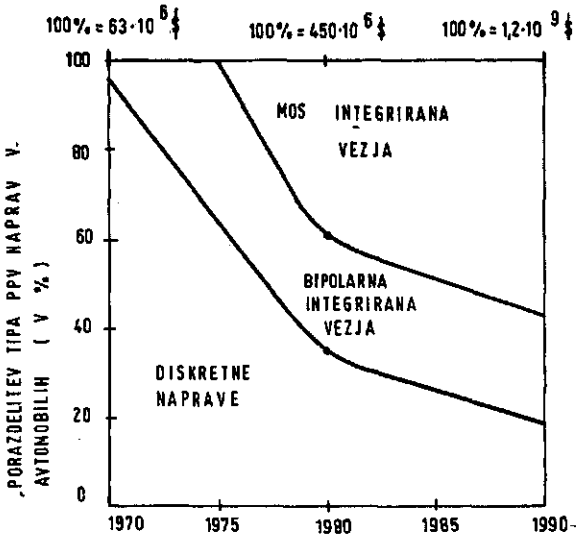
BMW je v sodelovanju s firmo BOSCH razvil digitalni elektronski sistem Motronic za krmiljenje vžiga in vbrizga goriva. Jedro sistema je mi-

kroračunalnik, ki ga sestavlja mikroprocesor COSMAC 1802 firme RCA, podatkovni in programski spomin ter I/O vezje. Senzorji zbirajo informacije o obratih motorja, poziciji ročične gredi, hitrosti in temperaturi dotekajočega zraka, temperaturi motorja in poziciji pedala za plin. Podatki o poziciji pedala za plin in odgovarjajočih obratih so shranjeni v spominu in na njihovi osnovi mikroračunalnik izračuna pravi trenutek za vžig, vbrizg goriva in količino vbrizganega goriva ter ustrezno krmili te funkcije. Sistem Motronic prihrani gorivo, izboljša štart v hladnem vremenu in poskrbi za mirno delovanje motorja.

Nissan uporablja za svoje 6 valjne motorje krmilnik ECCS (Electronic Concentrated Engine Control System), ki ima mikroprocesor 6802, dva 4 kbyt ROM, 128 byt RAM, I/O vezje (PIA), 10 in 8 bitni A/D pretvornik in 8 bitni množilnik, vse od firme Hitachi.

5. ZAKLJUČEK

Polprevodniška in avtomobilska industrija v preteklosti nista imeli mnogo skupnega, toda v zadnjem obdobju so ju skupni interesi zbližali. Kljub vse boljšemu sodelovanju obstajajo določeni problemi. Razvojni ciklus v polprevodniški industriji je približno eno leto, v avtomobilski pa od treh do pet let, tako da se investicija, ki se vloži v razvoj polprevodniškega elementa, vrne razmeroma počasi. Povpraševanje po elektronskih elementih s strani avtomobilske industrije presega ponudbo v razmerju 1,4:1, zato polprevodniška industrija veliko vlaga v razširitev kapacitet, kljub temu da je leta 1974 prišlo do hiperprodukcije. Cene elementov so danes še visoke, vendar bodo v bodoče ob večjih količinah padle. Na sliki 7 je prikazan trend razvoja elektronskih elementov za avtomobilsko industrijo.



Slika 7: Tipi elektronskih vezij za avtomobilsko industrijo

Viri:

-Gil Bassak, Microelectronic takes to the road in a big way: a special report (Electronics, 20.11.1980, str. 113-122)
-J. Gosch, Electronic system optimizes ignition and fuel injection (Electronics, 2.8.1979, str. 69-70)
-C.M. Heinen, E.W. Beckam, Balancing clean air against good mileage (IEEE Spectrum, November 1977, str. 47-50)
-G. Puckett, J. Marley, J. Gragg, Automotive electronics II: The microprocessor is in (IEEE Spectrum, November 1977, str. 37-45)
-J.G. Rivard, Microcomputers hit the road (IEEE Spectrum, November 1980, str. 44-47)

STATUT SLOVENSKEGA DRUŠTVA INFORMATIKA

UDK: 061.28.013 (497.12):681.3

I. SPLOŠNE DOLOČBE

Čl. 1

Ime društva je "Slovensko društvo INFORMATIKA" (v nadaljnem besedilu: društvo). Sedež društva je v Ljubljani, Gospodarsko razstavišče, Titova 50.

Čl. 2

Društvo je prostovoljno združenje vseh, ki so zainteresirani za razvoj in uporabo računalništva in informatike v naši deželi in jih združuje, vključuje in povezuje na profesionalni podlagi.

Čl. 3

Društvo deluje na območju SR Slovenije.

Čl. 4

Društvo je pravna oseba.

Čl. 5

Društvo ima svoj znak in pečat. Znak društva je oblikovan napis INFORMATIKA. Pečat društva je krog s premerom 3 cm, v sredini katerega je znak društva, ob robu pa napis "Slovensko društvo, Ljubljana".

Čl. 6

Društvo se včlanjuje v zvezo jugoslovanskih društev za informatiko in jugoslovansko zvezo za ETAN ob njihovi ustanovitvi.

Društvo se lahko včlani tudi v druge sorodne organizacije v SFRJ, katerih namen je pospeševanje in razvoj računalništva in informatike.

Čl. 6a

Društvo lahko ustanovi območne skupine društva zaradi učinkovitejšega delovanja povsod, kjer je za to izkazan interes članov.

Čl. 7

Društvo se lahko včlani v sorodno tujo ali mednarodno društveno organizacijo s podobnimi nameni in cilji, predpisanimi s temi pravili pod pogojem, da dejavnost te organizacije ni v nasprotju z interesi SFRJ.

Čl. 8

Dejavnost društva je zasnovana na ustavnih načelih, idejnopolitičnih izhodiščih samoupravnega socializma ter programski usmeritvi SZDL Slovenije. Društvo sooča v SZDL svoje interese z interesi drugih družbenih dejavnikov, ter se sporazumeva in dogovarja za družbene akcije, sodeluje pri sprejemanju političnih smernic, stališč in sklepov. Na lastno pobudo ali na pobudo organizacij SZDL se dogovarja o vseh aktualnih vprašanjih, še posebej o lastni programski zasnovi, kadrovski politiki, mednarodnem sodelovanju, založniški dejavnosti, politiki financiranja in drugem.

Čl. 9

Društvo v okviru svoje dejavnosti skrbi za uresničevanje družbene samozaščite v skladu z ustavo in z delovanjem vseh organiziranih socialističnih sil, za zavarovanje naše socialistične samoupravne družbe pred vsemi vrstami in oblikami dejavnosti, ki spodkopavajo, ovirajo in ogrožajo njen razvoj. Pri tem se zavzema za podružabljanje in uresničevanje zasnove ljudske obrambe ter družbene samozaščite, krepitev in razvoj varnostne kulture pri svojih članih, kar zlasti dosega s:

- preprečevanjem vsakršne dejavnosti v društvu, ki meri na spodkopavanje temeljev socialistične samoupravne demokratične, z ustavo določene ureditve ali na drugačno protiustavno spremembo družbenopolitične in ekonomske ureditve v SFRJ;
- preprečevanjem vsakršne dejavnosti v društvu, ki meri na razbijanje bratstva in enotnosti ali enakopravnosti narodov in narodnosti Jugoslavije, na spodkopavanje svoboščin in pravic človeka in občana, zajamčenih z ustavo ali na razpihovanje nacionalnega, rasnega ali verskega sovraštva ali nestrpnosti;
- preprečevanjem vsakršne dejavnosti v društvu, ki bi bila naperjena zoper neodvisnost in ozemeljsko neokrnjenost države ter obrambo socialistične ureditve;
- razvijanjem socialističnega partriotizma in varnostne kulture, s pravočasnim seznanjanjem z vsebino, metodami in oblikami javnega in podtalnega delovanja sovražnih sil, ki imajo namen spod spodkopavati družbeno ureditev in zavirati naš samoupravni socialistični razvoj.

Čl. 10

Društvo obvešča o svojem delovanju širšo in ožjo javnost.

1. Ožjo javnost obvešča:
   - z razpošiljanjem vabil in zapisnikov občnega zbora članom društva,
   - z občasnimi obvestili društva,
   - s tem, da so zapisniki vseh organov društva dostopni na vpogled članom društva;
2. Širšo javnost obvešča:
   - s tem, da so seje vseh organov društva javne in da se nanje vabijo osebe, ki izkažejo tak interes,

- z izdajo vabil, strokovnih publikacij in obvestil ter preko drugih sredstev javnega obveščanja.

Za zagotovitev javnosti dela je odgovoren predsednik izvršnega odbora.

Čl. 11

Kot strokovno-družbena organizacija ima društvo naslednje namene in cilje:

- da pomaga in pospešuje razvoj tehnologije in znanosti na področju računalništva in informatike
- da spremlja, proučuje in pomaga pri razvoju, vpeljevanju in uporabi računalniške tehnike in informatike
- da skrbi za popularizacijo in pospeševanje strokovnega usposabljanja kadrov na svojem področju na vseh nivojih
- da povezuje delo svojega članstva z delom ostalih strok
- da omogoča svojemu članstvu in javnosti seznanjanje z najnovejšimi dosežki iz svojega področja doma in v tujini
- da sodeluje s podobnimi tujimi in mednarodnimi društvenimi organizacijami pri uveljavljanju in razvoju računalništva in informatike
- da sodeluje z organi oblasti, družbeno političnimi in gospodarskimi organizacijami in ostalimi institucijami pri vprašanjih računalništva in informatike
- da sodeluje pri izdelavi zakonskih predpisov in standardov, ki se nanašajo na razvoj in uporabo računalniške tehnike in informatike v gospodarstvu in družbenih dejavnostih
- da pri svojih članih razvija in vzpodbuja profesionalno etiko, ki je v skladu s socialističnimi načeli naše družbe.

Čl. 12

Društvo uresničuje svoje namene in cilje z naslednjimi dejavnostmi:

- s prirejanjem strokovno - znanstvenih sestankov, simpozijev, seminarjev, tečajev, razstav, tekmovanj
- z izdajanjem zbornikov referatov in člankov, strokovnega časopisa, internega glasila, posebnih obvestil in strokovne literature
- s posredovanjem podatkov o aktualnih problemih in najnovejših dosežkih na področju informatike doma in v svetu javnosti preko javnega obveščanja
- s povezovanjem in vključevanjem v podobne domače, tuje in mednarodne društvene organizacije v skladu z družbenimi dogovori
- s članstvom v zvezi jugoslovanskih društev za informatiko.

Čl. 12a

Društvo uresničuje svoje dejavnosti v okviru naslednjih sekcij:

- razvoj informacijske tehnologije
- družbeni pomen informatike
- izobraževanje in informacijska kultura

Po potrebi lahko društvo ustanovi nove sekcije.
Po potrebi se lahko v okviru posameznih sekcij formirajo komisije za specifične dejavnosti. Komisije vodijo člani, ki jih pooblasti izvršni odbor.

II. ČLANSTVO

Čl. 13

Član društva lahko postane vsak državljan SFRJ, ki je zainteresiran za razvoj in uporabo računalništva in informatike v naši deželi ali dela na tem področju, sprejme ta statut, se ravna po njem ter izrazi željo postati član društva in v ta namen predloži izpolnjeno prijavnico.

Član društva lahko postane tudi tujec, če je njegovo delovanje v skladu z načeli naše družbe in nameni društva, navedenih v tem statutu.

Čl. 14

Pravice članov društva so:

- da volijo in so izvoljeni v organe društva
- da sodelujejo pri delu organov društva
- da dajejo predloge in sugestije organom društva o delu in reševanju nalog
- da imajo vpogled v delo organov društva in dajejo o njem pripombe.

Čl. 15

Dolžnosti članov društva so:

- da volijo in so voljeni v organe društva
- da sodelujejo pri delu organov društva
- da spoštujejo ta statut ter sklepe organov društva
- da z osebnim prizadevanjem in vzorom pripomorejo k uresničitvi delovnega programa društva
- da redno plačujejo članarino
- da na svojem področju dela uveljavljajo družbeno samozaščito.

Člani društva in organov društva so osebno odgovorni občnemu uboru za vestno opravljanje sprejetih nalog in funkcij.

Čl. 16

Častni član društva lahko postane oseba, ki se je posebno uveljavila z delom pri razvoju področja dejavnosti društva in sprejema to priznanje.

Častne člane imenuje občni zbor na predlog Izvršnega odbora društva.

Čl. 16a

Društvo lahko izvoli častnega predsednika, izmed članov, ki so se posebno uveljavili pri razvoju društva in njenih dejavnosti. Častni predsednik je vabljen na seje organov društva, lahko pa mu občni zbor ali izvršilni odbor poveri tudi konkretne zadolžitve.

Čl. 17

Pravice in dolžnosti članov v organih društva so častne. Za svoje delo v organih društva člani praviloma ne prejmejo plačila. Izjema so operativno-administrativne funkcije, za katere se prizna honorar po pogodbi o delu.

Čl. 18

Članstvo v društvu preneha:

- z izstopom
- s črtanjem
- z izključitvijo na podlagi odločbe disciplinskega odbora društva
- s smrtjo

Čl. 19

Član izstopi iz društva prostovoljno, kadar poda Izvršnemu odboru društva pismeno izjavo o izstopu.

Čl. 20

Član se črta iz društva, če ne plačuje članarine v dobi zadnjih treh let.

Čl. 21

Član se izključi iz društva, če grobo krši pravice in dolžnosti, naštete v 14. in 15. členu statuta, če zavestno ravna proti interesom in ciljem društva in če je pred disciplinskim odborom obsojen za nečastno dejanje.

III. ORGANIZACIJA DRUŠTVA

Čl. 22

Organi društva so:

- občni zbor
- izvršni odbor
- nadzorni odbor
- disciplinski odbor

Mandat vseh organov društva traja 2 leti.

Občni zbor

Čl. 23

Občni zbor je najvišji organ društva in voli druge njegove organe. Sestavljajo ga vsi člani društva.

Čl. 24

Občni zbor je lahko reden ali izreden. Redni občni zbor sklicuje izvršni odbor enkrat na leto. Izredni občni zbor se skliče po potrebi. Skliče ga lahko izvršni odbor na svojo pobudo, na zahtevo nadzornega odbora ali na zahtevo 1/4 članov. Izredni občni zbor sklepa samo o stvari za katero je sklican.

Izvršni odbor je dolžan sklicati izredni občni zbor najkasneje v roku enega meseca po tem, ko je prejel tako zahtevo. V nasprotnem primeru lahko skliče izredni občni zbor 1/3 članov društva.

Sklicanje občnega zbora z dnevnim redom mora biti objavljeno najmanj 10 dni pred dnevom, za katerega je sklican.

Čl. 25

Občni zbor sprejema svoje sklepe z večino glasov navzočih članov. Način glasovanja določi občni zbor. Ko se glasuje o razrešnici organov društva, ne morejo o tem glasovati člani organov društva.

Volitve organov so tajne, vkolikor občni zbor na samem zasedanju drugače ne odloči.

Čl. 26

Občni zbor je sklepčen, če je ob predvidenem začetku navzočih več kot polovica članov.

Če ob predvidenem začetku občni zbor ni sklepčen, se začetek odloži za 1 uro, nakar občni zbor veljavno sklepa, če je prisotnih vsaj 10 članov.

Čl. 27

Občni zbor odpre predsednik društva in ga vodi, dokler občni zbor ne izvoli delovnega predsedstva. Poleg tega, občni zbor izvoli še zapisnikarja in dva overovatelja zapisnika, po potrebi pa tudi volilno komisijo, kandidacijsko komisijo, in druge delovne organe.

Čl. 28

Občni zbor:

- sklepa o dnevnem redu
- razpravlja o delu in poročilih izvršnega in nadzornega odbora ter sklepa o njem
- sprejema delovni program društva
- odloča o pritožbah proti sklepom izvršnega odbora ali odločbam disciplinskega odbora
- sklepa o finančnem načrtu za prihodnje leto ter potrjuje zaključni račun za minulo leto
- sprejema, spreminja ter dopolnjuje statut ter druge splošne akte društva
- z glasovanjem neposredno voli predsednika, podpredsednika in sekretarja društva ter člane izvršnega, nadzornega in disciplinskega odbora
- odloča o prenehanju in včlanjevanju društva v druge organizacije doma ali v tujini
- imenuje delegate in delegacije, ki zastopajo društvo pri drugih organizacijah
- odloča o višini članarine ter o olajšavah, ki se priznajo članom društva ob uporabi uslug društva
- imenuje častne člane društva na predlog izvršnega odbora.

Čl. 29

O delu občnega zbora se piše zapisnik, ki ga podpišejo predsednik delovnega predsedstva in oba overovatelja zapisnika.

Izvršni odbor

Čl. 30

Izvršni odbor opravlja organizacijske, upravne, administrativne in strokovno tehnične zadeve.

Je izvršilni organ občnega zbora in opravlja zadeve, ki mu jih naloži občni zbor ter zadeve, ki po naravi spadajo v njegovo delovno področje.

Čl. 31

Izvršni odbor predstavlja društvo pri vključevanju društva v samoupravni socialistični sistem in v ta namen v imenu društva sodeluje z republiško konferenco SZDL

in ustreznimi samoupravnimi interesnimi skupnostmi na način, kakor je to določeno s pravnimi predpisi in z drugimi splošnimi akti navedenih organizacij.

Funkcijo delegacije članov društva opravljajo v prednjih primerih člani izvršnega odbora.

Čl. 32

Izvršni odbor je za svoje delo odgovoren Občnemu zboru društva.

Čl. 33

Izvršni odbor sestavljajo:
predsednik, podpredsednik in sekretar društva ter 4 izvoljeni člani odbora. Predsednik društva je istočasno predsednik izvršnega odbora.

Čl. 34

Društvo ima predsednika in 3 podpredsednike. Predsednik (v primeru njegove odsotnosti ali zadržanosti pa podpredsednik) društva posamič zastopa društvo pred državnimi organi ter drugimi organizacijami ali tretjimi osebami po navodilih izvršnega odbora. Podpredsedniki vodijo sekcije iz 12 a člena tega statuta; eden od njih pa po dogovoru nadomešča predsednika.

Čl. 35

Izvršni odbor upravlja društvo v času med dvema občnima zboroma po sklepih in programu, sprejetih na občnem zboru.

Sestaja se praviloma štirikrat na leto, po potrebi tudi bolj pogosto.

Seje izvršnega odbora sklicuje predsednik društva. Sklic seje mora biti objavljen najmanj 7 dni pred dnevom, za katerega je seja sklicana.

Čl. 36

Člane izvršnega odbora voli občni zbor za dve leti in so lahko dvakrat zaporedoma izvoljeni.

Izvršni odbor imenuje tudi predsednika in člana organizacijskih in programskih odborov posameznih strokovno znanstvenih sestankov oziroma člane uredniških odborov posameznih publikacij ali glasil društva.

Čl. 37

V okviru svojega delovnega področja iz 30. člena statuta, opravlja izvršni odbor tele zadeve:

- sklicuje občni zbor in pripravlja poročila o delu ter predloge za občni zbor
- pripravlja predloge za splošne akte društva
- pripravlja in sestavlja predlog za finančni načrt in zaključni račun
- vodi posle, ki zadevajo evidenco članov
- imenuje iz vrst članstva komisije, organizacijske in programske odbore ter uredništva
- skrbi za materialno-finančno poslovanje in sredstva društva
- neposredno skrbi za uresničevanje ciljev in nalog, ki jih določajo 10., 11. in 12. člen tega statuta
- koordinira delo sekcij, organizacijskih in programskih odborov ter uredništev
- sklepa pogodbe o delu z osebami, ki opravljajo tehnično-administrativne posle društva ter usmerja in nadzoruje njihovo delo.

Čl. 38

Izvršni odbor sprejema sklepe, če seji prisostvuje večina članov. Sklepi so sprejeti, če zanje glasuje večina navzočih članov.

Čl. 39

V primeru izpraznjenih mest lahko izvršni odbor kooptira v svoj sestav največ dva člana.

Nadzorni odbor

Čl. 40

Nadzorni odbor je sestavljen iz treh članov, ki jih izvoli občni zbor za dobo enega leta.

Nadzorni odbor izvoli iz svoje srede predsednika. Občni zbor izvoli tudi dva namestnika članov nadzornega odbora.

Čl. 41

Naloga nadzornega odbora je, da spremlja delo izvršnega odbora med dvema občnima zboroma in da opravlja stalni nadzor nad finančnim poslovanjem društva. Nadzorni odbor je odgovoren občnemu zboru in mu mora pismeno poročati ob koncu leta.

Čl. 42

Nadzorni odbor sprejema veljavne sklepe, če so prisotni trije člani in če zanje glasujeta vsaj dva člana.

Člani nadzornega odbora ne morejo biti hkrati člani izvršnega odbora, vendar se praviloma vabijo na vse seje izvršnega odbora brez pravice odločanja.

Disciplinski odbor

Čl. 43

Disciplinski odbor sestavljajo trije člani, ki jih izvoli občni zbor za dobo enega leta.

Disciplinski odbor izvoli iz svoje srede predsednika. Disciplinski odbor vodi disciplinski postopek in izreka kazni po disciplinskem pravilniku.

Za svoje delo je odgovoren občnemu zboru društva.

IV. MATERIALNO-FINANČNO POSLOVANJE DRUŠTVA

Čl. 44

Dohodki društva so:

- članarina
- dohodki od prijavnin za strokovno znanstvena srečanja

in prodaje publikacij
- dotacije in subvencije
- darila in volila
- drugi dohodki

Društvo razpolaga s finančnimi sredstvi v mejah odobrenega finančnega načrta.

Čl. 45

Premoženje društva predstavljajo vse premičnine in nepremičnine, ki so last društva in so kot take, vpisane v inventarno knjigo. S premoženjem društva upravlja izvršni odbor.

Premičnine se lahko odstopijo ali odtujijo tretjim osebam le na podlagi sklepa seje izvršnega odbora. O nakupu in odtujitvi nepremičnin odloča občni zbor društva.

Čl. 46

Materialno in finančno poslovanje mora biti v skladu z načeli, ki veljajo za društva ter z veljavnimi predpisi.

Materialna in finančna evidenca se opravlja po načelih blagajniškega in materialnega poslovanja.

Čl. 47

Finančno poslovanje se odvija prek tekočega računa pri SDK.

Blagajnik vodi finančno poslovanje društva po pravilniku o materialno-finančnem poslovanju.

Denarna sredstva društva vodi blagajnik v blagajniški knjigi. Blagajnik poroča o finančnem poslovanju izvršnemu odboru.

Čl. 48

Finančne in materialne listine podpisuje predsednik ali sekretar društva.

Odredbodajalec je predsednik društva, v njegovi odsotnosti pa podpredsednik.

Čl. 49

Delo blagajnika je javno. Vsak član društva lahko zahteva vpogled v finančno in materialno poslovanje društva.

V. KONČNE DOLOČBE

Čl. 50

Društvo preneha obstajati:

- s sklepom občnega zbora, če zanj glasuje dvotretjinska večina navzočih članov
- z odločbo pristojnega upravnega organa o prepovedi dela
- če pade število članov pod 10

Čl. 51

V primeru prenehanja društva pripadejo njegova sredstva ustrezni organizaciji ljudske tehnike SRS, ki se določi ob prenehanju ali organizaciji, ki nadaljuje dejavnost društva v SR Sloveniji.

Čl. 52

Določbe tega statuta lahko spremeni ali dopolni samo občni zbor, če za to glasuje več kot dve tretjini prisotnih članov. Sklep o spremembi ali dopolnitvi statuta stopi v veljavo, ko ga overi pristojni upravni organ za notranje zadeve.

Čl. 53

V skladu s tem statutom ima društvo naslednje splošne akte:

- disciplinski pravilnik
- pravilnik o materialno-finančnem poslovanju
- pravilnik o članarini in olajšavah za člane
- pravilnik o poslovanju komisij, organizacijskih, programskih in uredniških odborov.

Splošni akti iz prednjega odstavka morajo biti sprejeti na prvem občnem zboru po ustanovitvi društva. Pravilnike sprejema občni zbor, pripravlja pa izvršni odbor društva.

Čl. 54

Ta statut je sprejel občni zbor društva na svojem občnem zboru, dne 19.03.1981 v Ljubljani in stopi v veljavo, ko ga overi pristojni organ za notranje zadeve. S tem dnem preneha veljati statut, sprejet na ustanovnem občnem zboru, dne 8.07.1976.

Sekretar društva: Katarina Seršen, dipl.ing.

Predsednik društva: mag.Milan Mekinda, dipl.ing.

NOVICE IN ZANIMIVOSTI

Operacijski sistemi za mikroračunalnike

Mikrosistemi dobivajo čedalje bolj izpopolnjene operacijske sisteme, ki počasi skupaj z materialno opremo uspešno zamenjujejo glavne enote velikih računalniških sistemov. Sistemska programska oprema, ki jo sestavljajo

- operacijski sistem in
- uslužnostni programi

je najbolj osnovna oprema vsakega računalniškega sistema. Ta oprema nadzira delovanje in protokole vseh vhodnih in izhodnih naprav, pomnilnika, posebne materialne opreme, kot je ura realnega časa in druge sestavne enote, ki zahtevajo programsko krmiljenje. Ta oprema zagotavlja tudi nadvse pomembno povezavo med računalniškim sistemom in človeškim uporabnikom, ko omogoča dostop do diskovnih zbirk, odmev znakov na zaslonu v polnodupleksnem prenosu itn.

Uporabniška povezava s sistemom je bila že doslej temeljito raziskana in bila je napisana vrsta operacijskih sistemov, ki to povezavo optimizirajo. Pri gradnji sistemov oz. njihovem dopolnjevanju se namesto zbirnih jezikov čedalje bolj uporabljajo visoki programirni jeziki. Druga bistvena lastnost je prenosljivost programske opreme, katere cilj je npr., da bi bilo mogoče enak operacijski sistem uporabiti na vsakem računalniku.

V zadnjih letih so se na tržišču pojavili trije bistveni operacijski sistemi za mikroračunalnike, ki se približujejo gornjim zahtevam in ki predstavljajo medsebojno konkurenčne proizvode. Z njimi bo v naslednjih nekaj letih obvladan pretežni del tržišča mikroračunalniških operacijskih sistemov. Ti operacijski sistemi so

- UNIX,
- CP/M in
- OASIS

Operacijski sistem UNIX. UNIX je označitev družine operacijskih sistemov, ki so bili razviti v Bellovih laboratorijih. Prvotni UNIX je bil razvit z enim samim ciljem: oblikovati programirno okolico, ki bo ob največji prožnosti uporabniško prijazna. Tako je bil razvit operacijski sistem, ki je neobičajno lahek za priučitev ter ima učinkovite pripomočke za razvoj programov.

Uporabniška povezava s sistemom UNIX se imenuje "lupina" in deluje kot interpret ukazne vrstice, ko sprejema uporabnikove ukaze za vzdrževanje zbirk in izvajanje programov. Lupina vsebuje tudi lastnost za izmenjavo podatkov med programi, v čemer je njena najizrazitejša moč.

Že od samega začetka je bil UNIX večuporabniški operacijski sistem in takšen je ostal tudi pri uporabi na mikroračunalniških sistemih. Te njegove lastnosti in liberalizacija Bellove licenčne politike bodo povzročile, da bo UNIX postal odločilen proizvod na področju računalniških operacijskih sistemov v osemdesetih letih.

Operacijski sistem CP/M. Lahko ugotovimo, da se CP/M nahaja v drugem delu spektra operacijskih sistemov, saj je dejansko postal standard za mikroračunalniške sisteme s procesorjem 8080A. Čeprav CP/M ne omogoča oblikovanje tako zapletenih struktur, kot jih najdemo v večjih operacijskih sistemih, zagotavlja uporabniku poln dostop do gibkih in trdnih diskov ter vse tiste dodatne usluge, ki so značilne za 8- in 16-bitne sisteme. CP/M je bil razvit v podjetju Digital Research in ustanovitelj tega podjetja Gary Kildall pravi, da je tedaj občutil na tržišču hudo pomanjkanje diskovnih operacijskih sistemov za mikroračunalnike. Popularnost sistema CP/M je zelo narasla, tako da obstaja danes zanj več jezikov in aplikativnih programov kot za katerikoli drugi operacijski sistem. Čeprav nudi UNIX vrsto naprednih programskih krmilnih mehanizmov, kot so vilice (fork) in cevi (pipe), pa ima tudi CP/M vse, kar je potrebno za uporabo 8-bitnih sistemov. CP/M se tudi ustrezno vključuje v omejeni mikroračunalniški pomnilnik, saj v osnovni konfiguraciji zasede le 7 do 12 k zlogov.

Seveda pa obstaja tudi večuporabniška oblika sistema CP/M, ki se imenuje MP/M. Ta sistem razširi lastnosti sistema CP/M na več uporabnikov hkrati in omogoča dostop k zelo velikim diskovnim zbirkam na trdnih diskih. Sistema CP/M in MP/M se prodajata na licenčni osnovi za procesorja 8080A in 8086. Programe za CP/M 8080A je mogoče prevesti v kod procesorja 8086, tako da jih je moč izvajati tudi na CP/M sistemu procesorja 8086.

Najnovejši tip operacijskega sistema tipa CP/M je tudi CP/NET, ki omogoča več uporabnikom, da si delijo drage vire, kot so trdni diski in tračne enote. Sistem CP/NET je moč realizirati v obliki različnih konfiguracij, kot so krožne in zvezdne zanke, kjer je MP/M sistem mojster in CP/M sistem vajenec, z več sistemi v obliki široko razpredene mreže računalnikov. V tem primeru ima vsak uporabnik svojo lastno procesno enoto za izvajanje svojih programov, ima pa tudi dostop do skupnih zbirk in do velikih zbirk na velikih pomnilnih napravah. Ta lastnost sistema CP/NET bo uporabna v vrsti okolic, kot so npr. pisarniški informacijski centri, nadzor nad inventarjem, zalogami ter obdelava besedil.

Operacijski sistem OASIS. Primer novega večuporabniškega operacijskega sistema za mikroračunalnike je OASIS. Ta sistem je razvilo podjetje Phase One Systems (Oakland, CA) ter se uporablja v računalnikih s procesorjem Z80. OASIS vsebuje vrsto ukazov, ki jih najdemo v velikih sistemih. Tako ima obdelavo več nalog (multitasking), uporablja gibke in trdne diske, vhode realnega časa s prekinitvami ter ima vgrajen tudi integralni tiskalniški navijalnik (spooler). Vsebuje tudi pomagalne funkcije za vse svoje ukaze, tako da se lahko uporabnik priuči delovanja sistema preko zaslona.

OASIS ima učinkovite pripomočke za obdelavo zbirk in urejanje besedil. Urejevalnik je skorajda že procesor teksta in aplikativni program SCRIPT to tudi je. SCRIPT vsebuje pripomočke za izdelavo priročnikov, dokumentacije, pisem itd. V SCRIPT je vključen močnejši, splošni urejevalnik teksta, ki ima lastnosti, kot so globalne spremembe, povezovanje vrstic, iskanje teksta, makroukazi, pomikanje teksta v zunanje zbirke in druge ukaze sistema OASIS. Urejevalnik vsebuje tudi pripomočke za tiskanje in oblikovanje, kot so poravnava vrstic, oštevilčenje in opis strani v odvisnosti od sodih in lihih strani, naslavljanje poglavij in opombe, centriranje, avtomatično sestavljanje vsebinske tabele, več vrst znakovnih oblik, globalne spremenljivke, dostop do sistemskega datuma in časa ter urejevanje po imenih in naslovih. Do šestnajst uporabnikov lahko uporablja hkrati te rutine.

OASIS omogoča razvoj programov v zbirnem jeziku ter v jezikih BASIC in COBOL. Jezik COBOL je standarden

(ANSI 3.23) z nekaj dopolnili in razširitvami. BASIC vsebuje sistem za razvoj programov z urejevalnikom, interpretom in iskalnikom napak. Je tudi povezan z operacijskim sistemom pri upravljanju zbirk, spreminjanju programskih vmesnikov za pogon periferije ter pri tiskanju (navijalnik).

Razvojni sistem za zbirni jezik ima več pripomočkov, med njimi iskalnik napak v zbirnih programih in povezovalni urejevalnik. Ta urejevalnik omogoča povezavo več objektnih (prevedenih) programov, prikazuje naložitveno (pomnilniško) preslikavo ter omogoča oblikovanje absolutnega programa iz premestljivega.

Za sistem OASIS obstaja tudi vrsta aplikativnih programov. Poslovni paketi imajo programe za splošno glavno (poslovno) knjigo, plačljive in došle račune, poštni seznam, vstop naročil, razporejanje in spremljanje zalog, obdelavo besedil itd. Obstajajo pa tudi paketi za medicinsko obračunavanje, upravljanje zobarske pisarne, razporejanje časa, analizo cen itn. Zanimivi so tudi nekateri komunikacijski programi za emulacijo IBMovih bisinhronih protokolov, generiranje poročil, upravljanje podatkovnih baz v mrežah ter za splošno sistemsko obračunavanje.

OASIS se pojavlja v istih tržnih segmentih kot MP/M in XENIX (variante UNIXa podjetja Microsoft). Čas bo pokazal, kateri od treh sistemov je najboljši.

A. P. Železnikar

Ropanje programske opreme

Ropanje (piratstvo) programske opreme postaja čedalje večji problem na področju komercialnega in osebnega računalništva. Pirat je v tem primeru označen kot neupravičen ponatiskovalec programske opreme, kot kršitelj avtorske in založniške pravice, kot plagiator, ki si neupravičeno prisvaja rezultate raziskovalnega in razvojnega dela avtorja. Piratstvo na področju programske opreme ni samo nezakonito, sodi v področje kriminala (kraja) ter je z vidika veljavnih moralnih norm tudi nenravno (neetično).

Avtorji varujejo svojo programsko opremo na več načinov: pravno in tehnološko. Tehnološka zaščita je v tem, da programa ni moč kopirati oziroma je mogoča le ena kopija. Vendar ta zaščitni mehanizem ne velja za eksperta, ki obide zaporo ter kopira določen program neomejeno mnogokrat. Industrija se lahko zaščiti pred pirati tako, da daje kompleksnejše in popolnejše ponudbe (ima prednost pred piratom) ali pa piratu navidezno onemogoči kopiranje. Prednosti proizvajalca so priročniki (ki jih je teže kopirati), vzdrževanje paketov in ostale, dodatne informacije za uporabnika.

Avtor je proti piratu zaščiten tudi pravno, če je uveljavil svojo avtorsko pravico (copyright). Ker prihaja večkrat do modifikacije določenih uspešnih programskih paketov, lahko avtor v spornem primeru sproži postopek preverjanja določene programske opreme. Avtorska zaščita programske opreme je podobna zaščiti avtorskih tekstov (knjig, priročnikov). Če npr. pisatelj prepisuje odstavke različnih avtorjev ter jih zlaga s svojimi odstavki v nov roman, predstavljajo prepisani odstavki plagiate, ki s stališča avtorskih pravic niso dopustni. Prodaja in razpečavanje takega dela se na osnovi dokazov prepove, sporni avtor (plagiator) pa mora poravnati tudi odškodninske zahtevke. Podobna situacija lahko nastopi v primerih prisvajanja in prodaje programskih paketov, ko se primerjajo sporni programi glede na ukazna zaporedja avtoriziranih programov (dolžina enakih zaporedij v avtorskem piratskem paketu ne sme biti večja od določenega števila).

A. P. Železnikar

Največji na področju računalništva v tem desetletju

Do konca tega desetletja je predvideno še dodatno letno povečanje proizvodnje računalniških izdelkov, ki bo znašala 50 milijard dolarjev (k obstoječim 50 milijardam). Skupen proizvod 100 milijard dolarjev na leto naj bi si v glavnem razdelila le štiri podjetja: napovedi govorijo, da so to IBM, DEC, Xerox in AT&T. V letu 1990 naj bi DEC zaradi svoje hitre rasti dosegel velikost podjetja IBM. Vzrokov za počasnejšo rast podjetja IBM je več. Največji porast dohodka se pričakuje na področju sistemov za pisarniško avtomatizacijo. IBM naj bi se prelevil v cenenega proizvajalca s transformacijo svoje računalniške proizvodnje v informacijsko.

Prejšnja IBMova taktika je temeljila na t.i. dežniku cen ter na poudarjenem napadalnem trženju, kar je pripeljalo do stanja, ko podjetja - kupci niso mogla več izpolnjevati IBM-ovih zahtev. Tako je IBM sam preusmeril prej zanj donosno poslovanje v naročje konkurence. Zaradi tega bo IBM prisiljen povečati obseg proizvodnje in znižati cene. Že v letih 1978 do 1980 je IBM štirikrat povečal svoje proizvodne površine. Največja rast IBMa naj bi bila na področju malih in pisarniških sistemov, toda z nižjo stopnjo dobička in večjim obsegom plasmaja.

Največji IBMovi konkurenti bodo japonska podjetja, ki bodo ponujala popolno zbirko sistemov, vključno pisarniške, z lastno programsko opremo in vzdrževanjem. Ti pripomočki bodo vedno bolj inteligentni s poudarkom na znižanju osebnih izdatkov.

Xerox naj bi postal vodilno podjetje za sisteme pisarniške avtomatizacije, s tem, da bo sam proizvajal vse komponente sistemov ter bo odločilno vplival na tržišče. Podjetje AT&T ima svojo perspektivo, saj celotna informacijska industrija postaja čedalje bolj komunikacijsko usmerjena. DEC raste hitreje kot IBM; čeprav je njegova strategija v tem trenutku iluzijska, bodo DECovi mali sistemi zagotavljali njegovo stabilno rast.

A. P. Železnikar

Kaj je AFIPS ?

AFIPS (American Federation of Information Processing Societies) je organizator nacionalnih računalniških konferenc (NCC) in praznuje letos že 20. obletnico ustanovitve. AFIPS podpira vrsto aktivnosti, sestavlja pa ga 13 strokovnih združenj: ameriško združenje za informacijsko znanost, ameriško statistično združenje, združenje za računalniško lingvistiko, združenje za računalniško strojništvo, združenje za izobraževalne podatkovne sisteme, združenje za upravljanje obdelave podatkov, ameriško združenje za instrumentacijo, združenje za industrijsko in uporabno matematiko, združenje za računalniško simulacijo, IEEE računalniško združenje, združenje za prikaz podatkov, institut Charles Babbage ter mednarodna federacija za obdelavo podatkov (IFIP).

AFIPS skrbi za izmenjavo informacij med svojimi članicami ter za odnose z javnostjo, posreduje med vlado ZDA ter svojimi članicami. Razpolaga z dvema knjižnicama (v Washingtonu, D.C. in Arlingtonu, VA). AFIPS prispeva letno $ 50.000 za zgodovino in kronologijo obdelave podatkov; ta naloga se opravlja v institutu Charles Babbage. Največje dohodke ustvarja AFIPS z organizacijo NCC.

A. P. Železnikar

32 - bitni mikroprocesorji : novi podatki

Na razstavi v okviru NCC v Chicagu (4.5. - 7.5.81) je podjetje Intel razstavilo svoj novi sistem z 32-bitnim mikroprocesorjem iAPX432 (integrirana vezja 43201, 43202 in 43203). Pri tem so bili objavljeni tudi podrobnejši podatki za to procesorsko družino. Intel je pri svojem novem procesorju dokončno opustil prejšnjo arhitekturo in ukazno zalogo, tako ni programske združljivosti z mikroprocesorjem 8086 (16-bitni) in 8085 (8-bitni). Vsako od treh integriranih vezij iAPX432 ima štiri vrstice s po 16 nožicami (skupno 64 nožic). Dve vezji sestavljata splošni procesor, tretje vezje V/I procesor. Procesor iAPX432 se lahko poveže s procesorjem 8086 ter s perifernimi procesorji in pomnilnimi vezji. Intel zagotavlja zmogljivost 2 MUNS (milijon ukazov na sekundo).

Razvoj procesorja iAPC432 je trajal pet let in podjetje Intel je prispevalo (le) 25 milijonov dolarjev za ta projekt. V prvem letu proizvodnje bo Intel prodal 10 000 kompletov, proizvodnja pa bo stekla v letu 1982. Začetna cena kompleta bo 1500 dolarjev (znižanje na polovico glede na prejšnje napovedi). Intel je začel dobavljati ocenjevalne komplete v februarju 1981, ocenjevalna plošča pa ima ceno 4250 dolarjev.

Intel zatrjuje, da ima vsako od treh integriranih vezij procesorja iAPC432 približno 200 000 tranzistorjev (skupaj približno 600 000 tranzistorjev). Dve vezji delujeta kot zaporedni linijski (pipeline) par : 43201 ima ukazni dekodirnik, 43202 pa vsebuje mikroizvajalno enoto. Kot že omenjeno, je 43203 V/I procesor, ki povezuje V/I podsistem z dostopnozaščitno okolico centralnega sistema. Vsak V/I podsistem uporablja 8- ali 16-bitni mikroprocesor za neodvisno krmiljenje V/I od centralnega sistema. Procesor razpolaga z več kot 4 G zložnim naslovnim prostorom (4.10E9 zlogov) ter z virtualnim pomnilnonaslovnim prostorom 1T zlogov (10 E 12 zlogov).

Poseben zaščitni mehanizem omejuje dostop h programom. Procesor iAPX432 ima vgrajene aritmetične operacije s pomično vejico za 32, 64 in 80 bitna števila. Materialne napake se lahko ugotovijo s povezavo identičnih procesorjev iAPX432 v avtomatično preizkuševalno napravo.

Sistem uporablja prevedeni ADA kod kot računalniški (strojni) jezik. Jezikovni interpret je vsebovan v 64 k zložnem ROMu.

Intel je dal v prodajo tudi križni prevajalnik jezika ADA za procesor iAPX432. Ta prevajalnik se izvaja na sistemu DEC VAX-11/780 (Delta 3780) ter na IBM 370, njegova cena je 30.000 dolarjev. Navzdolnja materialna povezava za prevedeni kod na Intelovo ocenjevalno ploščo ima ceno 50.000 dolarjev.

Pri proizvodnji procesorja iAPX432 ima Intel dvoletno prednost pred konkurenco. Hewlett-Packard (HP) je namreč objavil razvoj svojega 32-bitnega procesorja, ki je že zgrajen in preizkušen, integriran pa bo v enem samem vezju s 450 000 tranzistorji (kar je nekaj manj, kot ima Intel v svojih treh vezjih). Ta procesor deluje s taktno frekvenco 18 MHz ter je mikroprogramiran z 9 K besedami s po 38 biti v posebnem ROMu. HP bo imel še štiri dodatna vezja: V/I krmilnik, pomnilniški krmilnik, 128 K-bitni programirljivi pomnilnik in 512 k-bitni ROM. Procesor je še v razvoju, začetek proizvodnje še ni bil določen.

Podjetje Texas Instruments (TI) je najavilo za začetek prihodnjega leta procesor z oznako 99000. Ti še ni objavil podrobnosti, vendar kaže, da gre za procesor z 32 naslovnimi biti brez 32-bitne obdelave.

IEEE (Institute of Electrical and Electronics Engineers) ima delovno skupino, ki razvija standarde za vodila mikroprocesorjev (od 8 do 32 bitov). Standard bo imel 32-bitno multipleksirano naslovno in podatkovno vodilo, ki bo združljivo z 32-, 16- in 8-bitnimi mikroračunalniki. Vodilo bo omogočalo uporabo do 32 mojstrskih in večposelnih (serijske medprocesorske povezave, presoja prekinitev) signalov. Predvidena taktna frekvenca bo lahko več kot 10 MHz.

A. P. Železnikar

SODELOVANJE VTŠ MARIBOR IN DO DELTA

Znanstveno tehnološka revolucija je tesno povezana z uvajanjem računalniške tehnologije. Informacije dobivajo iz dneva v dan vse večjo vlogo v procesu odločanja na vseh ravneh združenega dela.

Za osnovne tehniške usmeritve znanstveno tehnološke revolucije se običajno smatrajo naslednja glavna področja:

- novi viri energije
- kemizacija proizvodnje
- biologizacija proizvodnje
- osvajanje vesoljskega prostora
- avtomatizacija in računalniška tehnika.

Središče novih proizvajalnih sil, ki revolucionira vse proizvodne dejavnike, je kibernetski sistem kot priprava optimalne izbire prenašanja informacij in znanstvenega upravljanja.

Računalniška tehnologija je torej ključna tehnologija znanstveno tehnološke revolucije in zato ni čudno, da spada med tehnologije, ki uživajo posebno nacionalno pozornost v vseh državah razvitega sveta.

Naša država se je z ozirom na situacijo v svetu na področju računalništva znašla v podobnem položaju kot Evropa koncem šestdesetih let napram ZDA. Že 1967 je Jean Jacques Servan - Schreiber zapisal, da pri bodočem razvoju ne bodo pomembni niti nafta, niti tone jekla, niti dolarji, pa tudi ne sodobni stroji, temveč izključno znanje ter kreativna in organizatorska sposobnost. Od šestdesetih let dalje je v evropskem gospodarstvu elektronika dobila prvo mesto na področju znanstvenega raziskovanja ter razvoja.

Razvoj elektronske industrije pomeni razvoj produktivnosti. Računalnik pa v nobenem primeru ne smemo obravnavati izključno kot elektronsko aparaturo. Na področju elektronskega hardwara bomo v naši državi vedno zaostajali za razvitim svetom. Vendar naš cilj naj ne bi bil konkurenčnost na področju hardwara, čeprav je potrebno v celoti podpirati razvoj domače tehnologije.

Za nas ostaja bistveno, da na področju programske opreme (softwara) kot najznačajnejše komponente računalniške strategije, najdemo stik z razvitim svetom. Kvaliteta informacijskih sistemov pa je v veliki meri odvisna prav od tehnologije oz. koncepta obdelave podatkov. Kakšen koncept obdelave podatkov bomo v danem trenutku lahko realizirali, je odvisno predvsem od sposobnosti razpoložljive opreme. V SFRJ je več kot 75 % instalirane opreme starejše od 5 let, ki omogoča le paketno obdelavo podatkov. S takšno obdelavo podatkov ne moremo zagotoviti informacij, ki so potrebne za upravljanje proizvodnje pravočasno in na mestu, kjer so nam potrebne. Če želimo, da se bodo računalniki dejansko vključevali v gospodarstvo kot orodje v rokah uporabnikov za obdelavo poslovnih in tehniških informacij, potem je za proizvodne OZD sprejemljiva edino sprotna obdelava podatkov v realnem času, pri kateri lahko neposredni uporabnik interaktivno upravlja s programi in podatki.

VTŠ kot visokošolska pedagoška institucija s svojimi raziskovalnimi instituti želi na vsak način slediti sodobnim tokovom na področju računalništva, pri čemer smo oblikovali dva osnovna cilja:
1. vzgoja kadrov za potrebe OZD in sicer v rednem pedagoškem procesu in s funkcionalnim izobraževanjem v obliki seminarjev in tečajev,
2. gospodarstvu nuditi pomoč na področju programske opreme in projektiranja računalniško podprtih informacijskih sistemov.

Pri tem smo se povezali z obema slovenskima proizvajalcema aparaturne opreme tj. z ISKRO in ELEKTROTEHNO. Na VTO strojništvo smo v okviru instituta za strojništvo formirali dva centra z različno usmerjenostjo:
1. Center CETES, ki dela z Iskrino aparaturno opremo in sicer z osnovnim sistemom ISKRA DATA C18-20, je usmerjen v razvoj računalniškega projektiranja in konstruiranja. Pri tem smo se povezali še z Univerzo Imperial College iz Londona. Razen vrste programov za metode računalniškega projektiranja smo skupno razvili tudi lasten grafični zaslon, v razvoju pa je prototip ravninskega risalnika A_0.
2. Laboratorij za projektiranje informacijskih sistemov, ki dela z aparaturno opremo DO DELTA ELEKTROTEHNE z osnovnim sistemom DELTA 340/40, je usmerjen v obdelavo informacij na področju planiranja in spremljanja proizvodnje ter same tehnologije.

S tema dvema centroma podpiramo tudi osnovni izobraževalni smeri strojništva, to je:
- konstruktersko smer
- tehnološko smer.

Pri svojem razvoju tesno sodelujemo tako z ISKRO in DO DELTA, kot tudi z vrsto gospodarskih organizacij.

Cilji sodelovanja z OZD so naslednji:
1. Razvoj sprotne obdelave podatkov v realnem času, ki vključuje terminalski sistem dela z enotnim konceptom baze podatkov
2. Upravljanje s programi in skrb za vsebino podatkov mora preiti iz računskih centrov k neposrednim uporabnikom informacij. Treba se je namreč zavedati, da je centralna figura celotnega informacijskega sistema uporabnik in da mora biti celoten sistem temu tudi podrejen. S podatki in programi mora upravljati neposredni uporabnik s pomočjo terminala s svojega delovnega mesta.
3. Uporabnik mora dobivati selektivne informacije.
4. Vzgoja kadrov za potrebe OZD
5. Izdelava programske opreme in njena uporaba.

ZAKAJ PODPIRAMO NA VTŠ PROGRAM DO DELTA:

1. Smatramo, da je lahko edino nelicenčni koncept razvoja in proizvodnje računalniških sistemov za našo družbo dolgoročno sprejemljiv.
   Program DO DELTA v celoti ustreza osnovnim temeljem plana razvoja naše družbe.
2. DELTA računalniški sistemi so primerni za tehniške in poslovne informacije. Kot visokošolsko in raziskovalno institucijo tehniške smeri nas zanima predvsem uporabnost sistemov na tehniškem področju.

   DELTA sistemi so uporabni na naslednjih področjih:
   - avtomatizacija in procesna obdelava v energetiki, metalurgiji, petrokemiji, kemijski industriji, industriji cementa in v prometu
   - za vodenje proizvodnje na področju strojegradnje, obdelave kovin, tekstilne, lesne in farmacevtske industrije
   - za poslovne aplikacije v proizvodnih in neproizvodnih OZD
   - za raziskovalno dejavnost in šolstvo.
3. DO DELTA razpolaga z bogatim izborom programske opreme vključno s TOTAL sistemom za upravljanje baz podatkov, ki je najbolj razširjen tudi na računalniških sistemih drugih (tujih) proizvajalcev (IBM, CDC, NCR itd.).

Zaradi obojestranske želje po sodelovanju smo že leta 1978 podpisali samoupravni sporazum o sodelovanju za obdobje 1978 - 1982, s čimer smo opredelili skupne naloge na raziskovalnem, strokovno-tehničnem in izobraževalnem področju. Najvažnejši cilji tega sporazuma so:
- skupno izvajanje razvoja in raziskav
- medsebojno informiranje o potrebnih osnovnih raziskavah
- skupno razvijanje aplikativnega softwara za potrebe uporabnikov
- združevanje finančnih sredstev za razvoj aparaturne

in programske opreme in s tem racionalna uporaba osnovnih sredstev
- skupna uporaba tehniško informacijske dokumentacije
- skupno načrtovano izobraževanje kadrov
- opravljanje staža delavcev ene podpisnice pri drugi
- vključevanje strokovnjakov DO DELTA v študijski proces rednega in dopolnilnega izobraževanja.

Rezultat takšnega dogovora je bila instalacija računalniškega sistema DELTA 340/40 s pomnilnikom 256 Kbyte v prostorih VTŠ 1979. leta pod izredno ugodnimi finančnimi pogoji, ki nam jih je omogočila DO DELTA. S tem je bila dana osnova za sodelovanje. Ob strokovni pomoči DO DELTA je bil sistem v najkrajšem možnem času vključen v potrebe pedagoškega procesa, raziskovalnega dela ter nalog za OZD. S tem je začel razvoj našega laboratorija za projektiranje informacijskih sistemov.

V drugi polovici leta 1979 in v letu 1980 smo razvijali aplikativno programsko opremo za področje planiranja in spremljanja proizvodnje. Tozadevno so bili sklenjeni sporazumi z delovnimi organizacijami: AGIS Ptuj, STROJNA Maribor in Tovarno stikalnih naprav Maribor.

Izvršene so bile aplikacije z uporabo TOTAL sistema za upravljanje baz podatkov, in sicer za:
- tehnološke postopke
- kosovnice
- izpis proizvodne dokumentacije
- obremenitev strojev
- planiranje materiala
- obračun proizvodnih stroškov
- spremljanje naročil
- spremljanje zalog.

Razvita aplikativna programska oprema je verificirana v OZD. Na raziskovalnem področju je bil DELTA sistem vključen na področju mehanske obdelave z NC stroji, tehnoloških meritev, krmilja in regulacij in tudi na področju metod računalniškega projektiranja. Na osnovni sistem je danes na VTŠ vezanih 11 terminalov, razen tega sta terminalsko povezani s sistemom še Strojna ter TSN. V samem centru je trenutno zaposlenih 6 rednih sodelavcev, ki so delno tudi v deljenem rednem delovnem razmerju z DO DELTA. Poleg teh so v samo delo vključeni še zunanji sodelavci.

Dosedanji uspehi z računalniškim centrom DELTA so potrdili, da je bila naša odločitev pravilna, saj smo dobili izredno široko uporaben sistem, katerega uporabnost smo z vrsto programskih primerov tudi aplikativno verificirali. Pri tem je potrebno poudariti, da nam je DO DELTA nudila vso pomoč pri organiziranju in financiranju dejavnosti centra.

Zelo široka uporabnost tega računalniškega sistema je hitro privedla do tega, da so sistemske kapacitete ter prostorske težave postale omejitev nadaljnjega razvoja. Pri tem nam je DO DELTA ponovno pripravljena pomagati. Sklenjen je bil sporazum o adaptaciji prostora za ureditev prostorskih problemov centra. Celotno adaptacijo s prostorsko opremo v vrednosti 2.500.000 din je financirala DO DELTA na osnovi samoupravnega sporazuma o medsebojnem sodelovanju. S tem je DO DELTA ponovno pokazala, da je pripravljena izpolniti obveznosti po samoupravnem sporazumu in neposredno podpreti tudi izvajanje razvojno-raziskovalnega ter izobraževalnega programa VTŠ.

Dogovorili smo se prav tako za instalacijo drugega računalniškega sistema DELTA, katerega pričakujemo v kratkem. Rezultate sodelovanja z DO DELTA lahko ocenimo kot primer uspešnega sodelovanja z OZD, ki sloni na medsebojnem zaupanju in skupnih interesih.

Ob tej priložnosti v nobenem primeru ne smemo pozabiti na osnovo pri perspektivnem uvajanju računalništva v našo prakso, to je na kadre. Kadrovska problematika je predvsem v mariborskem področju izredno velika. Skupaj z DO DELTA smatramo, da je potrebno računalništvo kot predmet uvesti tudi v srednješolske programe, pri čemer je ob sedanjem uvajanju usmerjenega izobraževanja to še toliko bolj pomembno. DO DELTA in VTO strojništvo ugotavljata, da je Gimnazija Miloša Zidanška v svojem dosedanjem delu pokazala izredne kvalitete pri izobraževanju naravoslovno matematičnih predmetov in jo smatrata kot najbolj primerno za poučevanje računalništva. Ker je ta gimnazija z ozirom na formirano mrežo šol dobila naravoslovno-matematično usmeritev, je izbor v celoti pravilen. V smereh izobraževanja matematični tehnik, fizikalni tehnik ter biološko-kemijski tehnik je s fondom 245 ur (za matematičnega tehnika) oz. 70 ur vključeno tudi računalništvo. Pri tem bomo na DELTA računalniškem sistemu na VTŠ dijakom omogočili naslednje:
- uporabo terminalske učilnice
- delo na sistemu DELTA enkrat tedensko v popoldanskem času ob pedagoškem strokovnem vodstvu.

DO DELTA bo zainteresiranim predavateljem GMZ omogočila brezplačno šolanje v okviru izobraževalnega programa DELTA ter najkasneje v treh letih na GMZ uredila učilnico za pouk računalništva. Vidimo, da gre ponovno za veliko podporo DO DELTA.

PREDSTOJNIK VTO STROJNIŠTVO
doc. dr. Alojz KRIŽMAN

UVAJANJE PROGRAMIRNE TEHNIKE

Programiranje je v večini primerov dobro uporabljena množica, domiselnih prijemov in ne enotna metoda. Pri malo obsežnejši programski opremi, sistemih ali projektih pa se pokažejo pomankljivosti: slaba dokumentacija, nepoznavanje programov, spremembe se izvajajo predolgo, so drage, nezanesljive, nekvalitetne, testiranje se teško izvaja, ni več usklajenosti z zahtevam in programska oprema začne hitro umirati (še predno je dobro zaživela).

Mnogo informacij, ki so bistvene za ocenjevanje projekta, merjenje učinkovitosti programerjev in razumevanje življenskega cikla programske opreme, se običajno izgubi, npr.: zakaj je uporabljena določena odločitev, koliko časa je posamezni razvijalec porabil na različnih fazah projekta, koliko sprememb je izvedenih na vsakem modulu sistema, kako so moduli organizirani v celotni strukturi programske opreme, opis množice testnih podatkov tekom razvoja, posebne karakteristike posameznih modulov, katerim zahtevam ustrezajo moduli. Kvaliteta naj bi bila izražena z za-

nesljivostjo, možnostjo preizkušanja, učinkovitostjo, razumljivostjo in prilagodljivostjo.

Zato je v interesu razvijalskih teamov in uporabnikov, da se čim prej približamo rešitvi petih problemov: določitvi zadovoljivih zahtev, izboljšanju načina določevanja cene, doseganju pomembnega povišanja produktivnosti, izdelavi podpore za obvladovanje in doseganje preglednosti razvoja programske opreme in izdelavi popolne razvojne in uporabniške dokumentacije programske opreme.

V razvojnih OZD naj bi izdelali metodologijo programirne tehnike, prilagojeno njihovim posebnostim. Pri tem je treba upoštevati karakteristične faktorje: aplikacije, ki jih razvijajo, tipičen obseg teh aplikacij, ki vključuje tudi število razvijalcev in celotni čas razvoja, število verzij aplikacij in predvideno življensko dobo, iskušenost in spretnost razvijalcev.

Pomemben koncept poenotenja v tehniki (inženiringu) programske opreme je prav gotovo življenski cikel programske opreme, ki opisuje zaporedje faz razvoja programske opreme in njeno evolucijo . Oblikovanje metodologije inženiringa programske opreme vključuje uporavljalske pripomočke, uvajanje organizacijske strukture in izbor tehnik, ki povezujejo faze življenskega cikla programske opreme. Razvite so že številne metode in pripomočki za obravnavo različnih aspektov razvoja in evolucije programske opreme. Le malo pa jih je, ki pokrivajo vse faze življenskega cikla od začetnega koncepta sistema do modifikacij.

Metodologija naj:
- pokriva celotni življenski cikel- naj pomaga razvijalcu v vsaki fazi
- omogoča lahek prehod med fazami
- omogoča pregled nad napredovanjem projekta v številnih vmesnih točkah
- se nanaša na več tipov projektov
- se lahko osvaja
- ima neko avtomatizirano podporo.

Natančnejše razčlenjevanje pa zahteva oblikovanje podrobnih metod za razumljivo, zanesljivo in pregledno izdelavo naslednjih dokumentov:

- opis problema
- definicija zahtev (podprta s poročili analize)
- specifikacija programske opreme
- struktura programske opreme
- pretok in specifikacija podatkov
- opis programa (v nekem opisnem programskem jeziku)
- kodiranje programa (mogoče več variant)
- opis aktivnosti: za izpis, prevajanje, izvajanje...
- testiranje in testni pogoji
- dokumentacija za uporabnika (postopki vstavljanja...)
- poročilo o napakah ali težavah in zahteve za spremembe.

Na učinkovito in kvalitetnejše delo razvijalcev in teamov vpliva tudi okolje programske opreme, ki se uvaja paralelno z razumevanjem in uveljavljanjem procesov v zvezi z programsko opremo. Okolje vključuje tehnične metode, upravljalske procese, računalniško opremo in način uporabe, avtomatizirane pripomočke za razvoj programske opreme in delovno okolje. Ker je to raziskovalno področje še zelo mlado, je rigorozna definicija tvegana. Okolje mora biti zasnovano tako, da ima določeno širino za uresničevanje raznih ciljev, da je dovolj fleksibilno zaradi prilagajanja uporabniku, da je omogočeno tesno povezovanje zmogljivosti in uporabe centralnega sklada informacij. Postopek kreiranja okolja je zelo težaven ker je enakovreden dojemanju (razumevanju) osnovnih procesov programirne tehnike.

Na osnovi dosedanjih iskušenj pri posameznih fazah življenskega cikla in že izdelanih metod za posamezne faze, se naj celotna metodologija, vključno z razvojem okolja definira v okviru petletnega plana razvoja.

Neda PAPIĆ

IBM 3081 je le računalniški pritlikavec v primerjavi z novimi sistemi FUJITSU-JUMBO

S pravim pomnilniškim velikanom so presenetili Japonci računalniške strokovnjake, ko so napovedali dva nova superračunalnika. Večji model M-382 ima kar 128M zložni pomnilnik, ki je štirikrat večji od pomnilnika kompleksa IBM 3081, napovedanega v jeseni 1980. Manjši model M-380 ima glavni pomnilnik obsega 64M zlogov, s 64 V/I kanali ter kanalski pretok 96M zlogov v sekundi. Glede na doslej največji model M-200 ima M-380 zmogljivost, ki je za faktor 2,5 večja, pri M-382 pa za faktor 4,5. Oba sistema sta zračno hlajena, uporabljata pa operacijski sistem Facom OSIV/F4.

FUJITSU bo začel dobavljati nove sisteme v letu 1982, in sicer najprej na japonsko tržišče, kasneje pa tudi v Avstralijo in Evropo. Največja konkurenca med IBM in FUJITSU se kaže v Avstraliji, kjer je prodaja IBMovih sistemov bistveno nazadovala zaradi ugodnejšega razmerja cena/zmogljivost in servisnih uslug podjetja FUJITSU.

A.P.Železnikar

SEPTEMBER 1981

1. - 4. september, Brighton, England

European Conference on Electronic Design Automation (IEEE at all)

Organizator: IEEE Conference Department, Savoy Place, London, WC 2R OBC, England

8. - 11. september, Copenhagen, Danska

7th European Conference on Optical Communication

Informacije: M. Danielsen, Electromagnetic Inst. Technical University of Denmark, DK-2800 Lyngby Denmark

23. - 25. september, Firence, Italija

Computers in Cardiology

Informacije: Local Secretariat, OIC, Via G. Modena 19, 50121 Florence, Italy, tel. (055) 53-962

OKTOBER 1981

5. oktober, Ljubljana

ISEMEC 81 - VII. Seminar o uporabi mikroprocesorjev v merilni tehniki

5. - 6. oktober, Ljubljana

INFORMATICA 81 - XV. Jugoslovanski mednarodni simpozij za računalniško tehnologijo in probleme informatike

6. in 7. oktober, Ljubljana

YUTEL 81 - XV. Jugoslovanski simpozij o telekomunikacijah

7. in 8. oktober, Ljubljana

VAES 81 - II. Jugoslovanski simpozij: Vodenje in avtomatizacija elektroenergetskih sistemov

8. in 9. oktober, Ljubljana

SD 81 - XVII. Jugoslovanski simpozij o elektronskih sestavnih delih in materialih

8. in 9. oktober, Ljubljana

EP 81 - III. Jugoslovanski simpozij o elektroniki v prometu

Vse dodatne informacije dobite v pisarni Elektrotehniške zveze Slovenije
61000 Ljubljana, Titova 50, tel.: (061) 316-886

7. - 9. oktober, Nica, Francija

7th International Conference on Very Large Data Bases
Informacije: Dr. M. Edelberg, Sperry Research Centre, 100 North Road, Sudbury, MA 01776, USA

18.-22. oktober, Portsmouth, New Hampshire
ACM/MIT Conference on Functional Programming Languages and Computer Architecture
Informacije: J. Denis, MIT Laboratory for Computer Science, 545 Technology Sq., Cambridge, MA 02139

NOVEMBER 1981

1. - 4. november, Washington, USA

5th Annual Symposium on Computer Application in Medical Care
Informacije: Jan Edridge, Office of Continuing Medical Education, G. Washington Universty Medical Center, 2300 K St N.W., Washington, DC 20037

9. - 1. november, Las Angeles, California

ACM 1981 Annual Conference
Informacije: ACM, 1133 Avenue of the Americas, New York, NY 10036

MAREC 1982

9. - 11. marec, Zürich, Švica

International Zürich Seminar on Digital Communications
Informacije: Miss M. Frey, EAE, Siemens - Albis, A6 POB, CH 8047 Zürich, Switzerland

30. marec - 1. april, Metropole, Sussex, UK

CAD 82
Informacije: Alan Pipes, IPC, Science and Technology Press, PO BOX 63, Westbury House, Bury st., Guildford GU2 5BH, UK

RAZSTAVE 1981

29. - 31. julij, London, UK

Microcomputer Show

26. - 29. avgust, Coliseum, New York

5th Annual National Small Computer Show

5. - 9. oktober, Ljubljana

Sodobna elektronika

3. - 13. november, Beijing, LR Kitajska

1st US Telecommunications exhibition in China

Poročilo o letni evropski konferenci ACM, London

Glavni cilj simpozija je bil poročanje o trenutnem stanju in izmenjava idej med znanstveniki, inženerji in vodilnimi delavci o problemih tehnik in teženj v arhitekturi sistemov. Povabljeni predavatelji so bili znani strokovnjaki na določenih področjih. Od 160 prispelih referatov je bilo izbranih in predstavljenih 60. Razdeljeni so bili po teh področjih:

1. Distribuirana in odprta arhitektura
2. Multiprocesorji in mikroprogramiranje
3. Komunikacije
4. Specifikacije sistemov in zahteve
5. Metode in upravljanje
6. Zgradba podatkovnih baz
7. Sistemi, neobčutljivi na napake
8. Analiza in konstruiranje velikih sistemov
9. Vmesnik človek - stroj
10. Programska oprema in arhitektura sistemov
11. Načrtovanje jezikov
12. Arhitektura pretoka podatkov
13. Upravljanje z informacijam in sistemi.

Čeprav je bila to formalna razdelitev, je bilo mogoče skoraj vse objavljene referate razvrstiti v dve ali tri navedene skupine. Časi začetka posameznih predavanj so bili v vseh treh dvoranah enaki in bil je mogoč prehod iz skupine v skupino.

Splošnega pomena za vse prisotne so bila predavanja povabljenih predavateljev, posebno W. Turskega in P. Naura. Predavatelji so bili večinoma vodilni strokovnjaki na področju programske opreme in slušatelji so bili seznanjeni predvsem s trenutnim stanjem na tem področju v Evropi. Očitno je stopnja razvoja programske opreme dosegla nivo, ko so postali primarni taki problemi,ki nastajajo zaradi vse večjega nagibanja raziskovalnih inštitucij in univerz k uporabnikom. Pojavlja se človek kot uporabnik, naročnik in človek kot načrtovalec sistema. Ta dva uporabljata različno strokovno terminologijo, imata pa enak končni cilj - da inštaliran sistem sistem na karseda optimalen način zadovolji zahteve uporabnika. Omenjeni so problemi enega in drugega, posebna pozornost pa je posvečena razvoju in ovrednotenju metod za lažje komuniciranje med načrtovalci in uporabniki pri določevanju karakteristik sistema.

To je bila tudi tema panelnega pogovora "Formalne specifikacija, da ali ne?"

Velik poudarek je bil tudi na multiprocesorskih in distribuiranih sistemih in na predlaganih rešitvah za nekaj problemov, ki se na tem področju pojavljajo.

O stabilnosti programske opreme je govoril W. Turski v svojem referatu. On se ne strinja s splošno uporabljenim izrazom "maintenance" za vzdrževanje programske opreme, ker pravi, da je to vse prevečkrat popravljanje,dodajanje in nadgradnjo obstoječih programov, ne pa samo vzdrževanje in varovanje pred napakami. Pravi problem je stabilnost procesa spreminjanja in razširitve programov in stabilnost dinamičnega procesa. Razprava je temeljila na shemi specifikacija, program, realni model.

Faktor, ki vpliva na stabilnost razvoja programske opreme, je strukturna povezava med specifikacijo in modelom programa. Iz vsega se lahko povzamejo nekatera priporočila, ki prispevajo k stabilnosti razvoja programske opreme:

- uporaba formalne specifikacije, ki jo je moč verificirati
- formalno oblikovana, zadovoljiva povezava med programi in specifikacijami
- prepoved sprememb programa brez predhodne spremembe formalne specifikacije, ki ostaja ohranjena
- faktorizacija programov, ki ustreza eksplicitni razgradnji specifikacije.

Za uporabnika je predlagana izdelava kataloga mogočih sprememb specifikacij, ki se implementirajo relativno poceni. Predlagano je tudi domiselno oblikovanje metapodpornega sistema (meta-run-time-on line-support system--MRTOLSS) z jezikom za spremembe specifikacij in prevajalnikom, kjer uporabnik vtipka želeno spremembo specifikacije in če jo MRTOLSS spozna kot spremembo, za katero obstaja alogritemska transformacija programskega teksta, bo sprememba sprejta in nova verzija je tako generirana brez posredovanja človeka.

Na področju razvoja programske opreme je v članku "O intraktivnem stopenjskem razvoju programske opreme" B. Krämerja prikazan tudi nov pristop. Uporabljena je metoda z predikativno aktivnimi mrežami.

V referatu o "Zanesljivem programiranju sinhronizacije nalog" za sisteme v realnem času je predložena rešitev z uporabo specifikacije, zasnovane na Petrijevih mrežah. Opisani sta dve tehniki: distribuirana in centralizirana.

Na področju analize sistemov bi omenila referat P. Naura "Empirički pristop k analizi in zgradbi programov". Kot primer je uporabljen program iz numerične analize, določanje najboljše aproksimacije z najmanjšo napako katerekoli zvezne funkcije. Študija poudarja pomembnost intuitivno ocenjenih razlogov za določen pristop in opisov, ki imajo lahko več oblik, se pa izbirajo v odvisnosti od programa. Kot primer je prikazan izvleček programa, t. im. skelet.

Na tem področju sta sodelovala še dva zanimiva avtorja, in sicer M. Lehman iz Londona in C. Floyd iz Berlina.

Posebnega zanimanja vseh udeležencev je bila deležna skupina za načrtovanje jezikov. Jasno, govorili so o Adi. Za te prispevke je mogoče zanimiv utrinek, ki ga je bilo pogosto slišati:"... Trenutno vas prosimo, da tega jezika ne poskušate implementirati na vašem sistemu".

Omembe vredno je tudi to, da so vsi predavatelji, ne glede na pomembnost teme imeli točno odmerjen čas za predavanje in so se tega tudi držali (vključno 10 min za razpravo). Posebno zanimivi so bili pogovori med polurnimi odmori kjer so se oblikovale interesne skupine in izmenjavale svoja mnenja.

V avli je bila organizirana razstava knjig Založbe Prentice Hall, ki bodo izšle v letu 1981. Naslovi se precej ponavljajo, je pa nekaj takih, ki so v celoti zanimive. Seveda je bila tudi tukaj posebna pozornost namenjena jeziku Ada.

Omenim naj še zbornik, ki je lepo opremljen, v katerem so objavljeni vsi referati, razen referatov nekaterih povabljenih predavateljev.

Neda PAPIĆ

VSEBINA LETNIKA 1980

PRIJAVA REFERATA/KRATKEGA REFERATA/ STROKOVNEGA POROČILA

Prijavo izpolnite s pisalnim strojem

1. Naslov referata .....
2. Razširjen povzetek (približno 1000 besed) priložite prijavi
3. Programsko področje referata (obkrožite ustrezno točko)
   1. programska oprema
   2. materialna oprema
   3. teoretični aspekti obravnavanja podatkov
   4. sistemi za upravljanje in administracijo
   5. upravljanje procesov
   6. razne aplikacije v znanosti in tehniki
   7. vzgoja in aplikacije v humanistiki
4. Razvrstitev referata (obkrožite ustrezno točko)
   1. referat — pomembnejše delo
   2. kratki referat
   3. strokovno poročilo
5. Značaj referata (obkrožite ustrezno točko)
   1. originalno teoretično delo
   2. opis konkretnega praktičnega dela
   3. pregledni referat
   4. ponovitev že znanega rezultata z novimi metodami
   5. kritična analiza
   6. opis novih hardwarskih in softwarskih proizvodov
   7. .....
6. Avtor in soavtorji: .....

Delovna organizacija .....

Ulica .....

Poštna številka ..... Mesto .....

Dežela .....

Datum ..... Podpis .....

Prijavnica, skupaj z dvema kopijama razširjenega povzetka mora prispeti najkasneje do 1. avgust 1981 na naslov: Informatica '82 Parmova 41, 61000 Ljubljana,

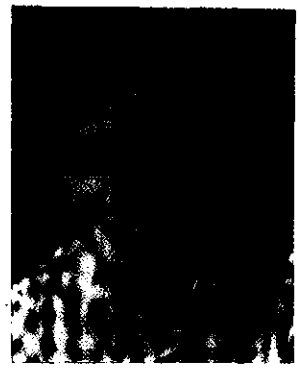
AVTORJI IN SODELAVCI

Milovan V. Jefić (1949) je diplomiral na Fakulteti za elektrotehniko na univerzi v Ljubljani, s tematiko iz področja računalništva.

Najprej je bil zaposlen na Zavodu za avtomatizacijo v Ljubljani, kjer je delal na problemih sinteze frekvence na osnovi PLL in telemehanskih sistemih. Od leta 1978 je zaposlen v DO Delta, kjer se ukvarja z razvojem elementov za DELTA računalnike.

Viljan Mahnič je diplomiral na Fakulteti za elektrotehniko Univerze Edvarda Kardelja v Ljubljani. Tu se je tudi zaposlil kot stažist raziskovalec in bil leta 1980 habilitiran za asistenta za področje računalništva in informatike. Od marca 1981 je zaposlen na Institutu Jožef Stefan, Odsek za uporabno matematiko, kjer je član skupine, ki se ukvarja z avtomatizacijo projektiranja s pomočjo računalnika. Je soavtor več referatov, ki obravnavajo omenjeno problematiko in so bili predstavljeni na raznih znanstvenih srečanjih v Jugoslaviji.

KATARINA SERŠEN. Diplomirala je leta 1979 na Fakulteti za elektrotehniko v Ljubljani, smer računalništvo in informatika, z nalogo: Statistična analiza slovenskega časopisnega besedila. Zaposlena je v Centru SRS za družbeni sistem informiranja in informatiko. Sodeluje pri razvoju informacijskih sistemov s področja družbenega sistema informiranja ter strokovno dela pri modernizaciji informacijskih sistemov državne uprave in administrativnega poslovanja. V Slovenskem društvu Informatika je tajnik društva in sodeluje pri vseh akcijah društva.



Božidar Blatnik (1953) je diplomiral 1976. leta na Fakulteti za elektrotehniko Univerze E. Kardelja v Ljubljani, smer za računalništvo in informatiko. V diplomskem delu obravnava učinkovite algoritme za probleme pretokov v grafih. Magistriral je 1978. leta na isti fakulteti z delom Problematika prenosa podatkov med računalniki v sklopu računalniške mreže. Po diplomi se je zaposlil na Institutu Jožef Stefan v Ljubljani, na odseku za avtomatiko in biokibernetiko. Ukvarjal se je z razvojem sistemske aplikativne programske opreme za mini in mikro računalnike. Sodeloval je pri razvoju merilnih instrumentov, ki jih podpirajo mikro računalniki. Bil je nosilec in sonosilec več raziskovalnih nalog, ki uvajajo mikro računalnike v procese vodenja tehnoloških procesov. Leta 1981 se je zaposlil v DO Delta v Ljubljani, kjer sodeluje pri razvoju inteligentnega terminala.

NAVODILO ZA PRIPRAVO ČLANKA

Avtorje prosimo, da pošljejo uredništvu naslov in kratek povzetek članka ter navedejo približen obseg članka (število strani A 4 formata). Uredništvo bo nato poslalo avtorjem ustrezno število formularjev z navodilom.

Članek tipkajte na priložene dvokolonske formularje. Če potrebujete dodatne formularje, lahko uporabite bel papir istih dimenzij. Pri tem pa se morate držati predpisanega formata, vendar pa ga ne vrišite na papir.

Bodite natančni pri tipkanju in temeljiti pri korigiranju. Vaš članek bo s foto postopkom pomanjšan in pripravljen za tisk brez kakršnihkoli dodatnik korektur.

Uporabljajte kvaliteten pisalni stroj. Če le tekst dopušča uporabljajte enojni presledek. Črni trak je obvezen.

Članek tipkajte v prostor obrobljen z modrimi črtami. Tipkajte do črt - ne preko njih. Odstavek ločite z dvojnim presledkom in brez zamikanja prve vrstice novega odstavka.

Prva stran članka :

a) v sredino zgornjega okvira na prvi strani napišite naslov članka z velikimi črkami;
b) v sredino pod naslov članka napišite imena avtorjev, ime podjetja, mesto, državo;
c) na označenem mestu čez oba stolpca napišite povzetek članka v jeziku, v katerem je napisan članek. Povzetek naj ne bo daljši od 10 vrst.
d) če članek ni v angleščini, ampak v katerem od jugoslovanskih jezikov izpustite 2 cm in napišite povzetek tudi v angleščini. Pred povzetkom napišite angleški naslov članka z velikimi črkami. Povzetek naj ne bo daljši od 10 vrst. Če je članek v tujem jeziku napišite povzetek tudi v enem od jugoslovanskih jezikov;
e) izpustite 2 cm in pričnite v levo kolono pisati članek.

Druga in naslednje strani članka:
Kot je označeno na formularju začnite tipkati tekst druge in naslednjih strani v zgornjem levem kotu,

Naslovi poglavij:
naslove ločuje od ostalega teksta dvojni presledek.

Če nekaterih znakov ne morete vpisati s strojem jih čitljivo vpišite s črnim črnilom ali svinčnikom. Ne uporabljajte modrega črnila, ker se z njim napisani znaki ne bodo preslikali.

Ilustracije morajo biti ostre, jasne in črno bele. Če jih vključite v tekst, se morajo skladati s predpisanim formatom. Lahko pa jih vstavite tudi na konec članka, vendar morajo v tem primeru ostati v mejah skupnega dvokolonskega formata. Vse ilustracije morate (nalepiti) vstaviti sami na ustrezno mesto.

Napake pri tipkanju se lahko popravljajo s korekcijsko folijo ali belim tušem. Napačne besede, stavke ali odstavke pa lahko ponovno natipkate na neprozoren papir in ga pazljivo nalepite na mesto napake.

V zgornjem desnem kotu izven modro označenega roba oštevilčite strani članka s svinčnikom, tako da jih je mogoče zbrisati.

Časopis INFORMATICA
Uredništvo, Parmova 41, 61000 Ljubljana

Naročam se na časopis INFORMATICA. Predplačilo bom izvršil po prejemu vaše položnice.

Cenik: letna naročnina za delovne organizacije 500,00 din, za posameznika 200,00/100,00/50,00 din

Časopis mi pošiljajte na naslov ☐ stanovanja ☐ delovne organizacije.

Priimek.....

Ime.....

Naslov stanovanja

Ulica.....

Poštna številka_____ Kraj.....

Naslov delovne organizacije

Delovna organizacija.....

.....

Ulica.....

Poštna številka_____ Kraj.....

Datum..... Podpis:

.....

INSTRUCTIONS FOR PREPARATION OF A MANUSCRIPT

Authors are invited to send in the address and short summary of their articles and indicate the opproximate size of their contributions (in terms of A 4 paper). Subsequently they will receive the outor's kits.

Type your manuscript on the enclosed two-column-format manuscript paper. If you require additional manuscript paper you can use similar-size white paper and keep the proposed format but in that case please do not draw the format limits on the paper.

Be accurate in your typing and through in your proof reading. This manuscript will be photographically reduced for reproduction without any proof reading or corrections before printing.

Use a good typewriter. If the text allows it, use single spacing. Use a black ribbon only.

Keep your copy within the blue margin lines on the paper, typing to the lines, but not beyond them. Double space between paragraphs.

First page manuscript:
a) Give title of the paper in the upper box on the first page. Use block letters.
b) Under the title give author's names, company name, city and state - all centered.
c) As it is marked, begin the abstract of the paper. Type over both the columns. The abstract should be written in the language of the paper and should not excesed 10 lines.
d) If the paper is not in English, drop 2 cm after having written the abstract in the language of the paper and write the abstract in English as well. In front of the abstract put the English title of the paper. Use block letters for the title. The lenght of the abstract should not be greater than 10 lines.
e) Drop 2 cm and begin the text of the paper in the left column.

Second and succeeding pages of the manuscript:
As it is marked on the paper, begin the text of the second and succeeding pages in the left upper corner.

Format of the subject headings:
Headings are separated from text by double spacing.

If some characters are not available on your typwriter write them legibly in black ink or with a pencil. Do not use blue ink, because it shows poorly.

Illustrations must be black and white, sharp and clear. If you incorporate your illustrations into the text keep the proposed format. Illustration can also be placed at the end of all text material provided, however, that they are kept within the margin lines of the full size two-column format. All illustrations must be placed into appropriate positons in the text by the author.

Typing errors may be corrected by using white correction paint or by retyping the word, sentence or paragraph on a piece of apaque, white paper and pasting it nearly over errors

Use pencil to number each page on the upper-right-hand corner of the manuscript, outside the blue margin lines so that the numbers may be erased.

INFORMATICA, Journal Headquarters
Parmova 41, 61000 Ljubljana, Yugoslavia

Please enter my subscription to INFORMATICA and send me the bill.

Annual subscription price: companies US $ 22, individuals US $ 7,5.

Send journal to my ☐ home address ☐ company's address.

Surname.....

Name.....

Home address

Street.....

Postal code _____ City.....

Company address

Company.....

.....

Street.....

Postal code _____ City.....

Date..... Signature

.....

CENIK OGLASOV

Ovitek - notranja stran (za letnik 1981)

2 stran	28.000 din
3 stran	21.000 din

Vmesne strani (za letnik 1981)

1/1 stran	13.000 din
1/2 strani	9.000 din

Vmesne strani za posamezno številko

1/1 stran	5.000 din
1/2 strani	3.300 din

Oglasi o potrebah po kadrih (za posamezno številko)

2.000 din

Razen oglasov v klasični obliki so zaželjene tudi krajše poslovne, strokovne in propagandne informacije in članki. Cene objave tovrstnega materiala se bodo določale sporazumno.

ADVERTIZING RATES

Cover page (for all issues of 1981)

2nd page	1300 $
3rd page	1000 $

Inside pages (for all issues of 1981)

1/1 page	790 $
1/2 page	520 $

Inside pages (individual issues)

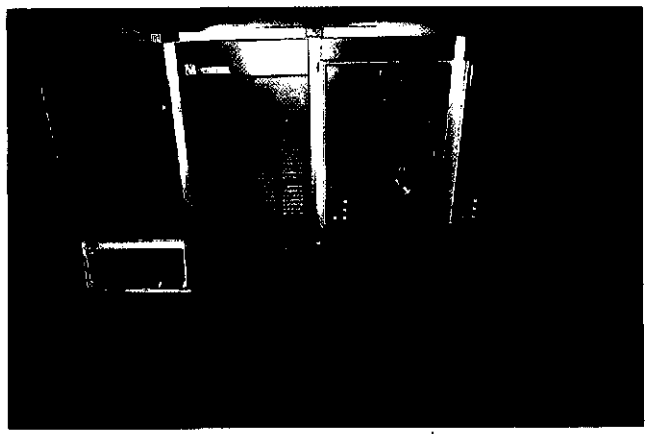
1/1 page	260 $
1/2 page	200 $

Rates for classified advertizing:

each ad	66 $

In addition to advertisment, we welcome short business or product news, notes and articles. The related charges are negotiable.


V KORAK S TEHNOLOGIJO —
V KORAK Z DELTO
delta računalniški sistemi



Delavci DO DELTA proizvajamo najpopolnejšo jugoslovansko družino računalnikov, katera obsega celotno območje od mikro računalnikov do največjih 32-bitnih sistemov.
Poseben pomen dajemo aplikacijski programski opremi. Ob izbiri segmentov je bila naša skrb posvečena povečanju produktivnosti in čimvečjemu prihranku energije ter surovin. Programski moduli za področja procesne kontrole, planiranja in upravljanja proizvodnje, ter finančnega poslovanja, predstavljajo integralen pristop v izgradnji informatizirane proizvodne delovne organizacije. Naši računalniki so narejeni tako, da niso element prestiža delovnih organizacij, ki jih kupujejo, temveč so orodje razvojnega inženirja, projektanta, delavca v skladišču in drugih. S takim načinom dela vstopa DELTA skupaj s svojimi uporabniki v informatizirano družbo prihodnosti . . .

Če želite več informacij o DELTI, pišite na naslov: ELEKTROTEHNA — DO DELTA, Služba za komuniciranje s tržiščem, Parmova 41, 61000 Ljubljana.