računalniški sistemi delta®

CENTRALNA PROCESNA ENOTA
DELTA 16/BIT-SLICE

ISKRA-DELTA

RAZVOJ APARATURNE OPREME

Iskra Delta

KAZALO

KAZALO

Stran

KAZALO

Stran

1. UVOD

DELTA 16/BIT SLICE je centralna-procesna enota za sistem DELTA 340. To je emulacija CPE KD 11-EA, kar pomeni, da sta enoti neposredno zamenljivi na standardnih konektorjih v sistemskem vodilu.
Arhitektura enote DELTA 16/BIT SLICE je zasnovana tako, da omogoča presledno in hitro tvorbo mikroprogramov za izvajane poljubnih makro instrukcij. S tem namenom je v enoti DELTA 16/BIT SLICE zajeto vezje za izvedbo suspenzije, kar pomeni, da se mikroprogram lahko tudi prekine in kasneje tudi nadaljuje (izpad napetosti in ponoven vklop). Zajeto je tudi vezje za adresiranje dodatnih 48 splošno uporabnih registrov, v instrukcijskem dekoderju in v vezju za obravnavo pasti pa je rezervirano področje za operacijske kode teh nestandardnih instrukcij. Primeri takih instrukcij so iz niza "Commercial Instruction Set". To je niz instrukcij, ki tečejo na sistemih PDP 11/44 ob uporabi dodatnega CIS procesorja.
Programska in aparaturna podpora za tvorbo mikroprogramov se sestoji:

- Macro Meta Assembler (AMDASM)
- Program za vpis v mikroprogramski pomnilnik
- Vpisljiv mikroprogramski pomnilnik (WCS) s sirino do 128 bitov in globino do 2K besed

in je na voljo potencialnim uporabnikom v prostorih ISKRA DELTA, Razvoj AO, Grubarjevo nabrežje 6.

2. NAČINI NASLAVLJANJA

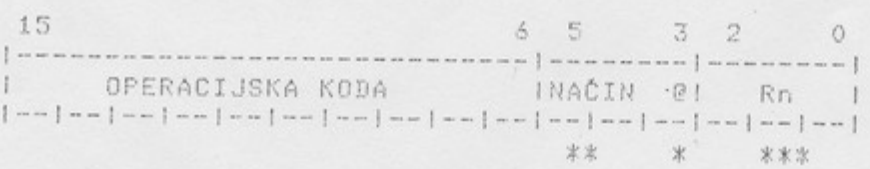
Naslavljanje pomnilnika se izvaja s pomočjo osmih registrov za splošno uporabo (General Purpose Registers).
Pri določanju naslova nekega podatka izberemo enega izmed osmih registrov in enega izmed možnih načinov naslavljanja. Vsak ukaz, ki naslavlja pomnilnik, je sestavljen iz:

-funkcije, ki naj se izvrši (instrukcijska koda)
-registra splošne uporabe, ki ga uporabimo za določanje izvornega (SOURCE) ali namembnega (DESTINATION) operanda
-načina naslavljanja, ki določa na kakšen način uporabimo izbrani register

Zelo pomembni so načini naslavljanja, ki jih programer lahko uporablja. Nabor ukazov in načinov naslavljanja, s katerimi razpolaga DELTA 16/BIT-SLICE procesor, omogoča učinkovit in fleksibilen dostop do podatkov. Glede na način naslavljanja lahko register splošne uporabe deluje kot:

-AKUMULATOR: vsebuje podatek, ki ga želimo uporabiti
-KAZALEC:vsebina registra je naslov podatka
-INDEKSNI REGISTER:vsebino registra prištejemo besedi v ukazu, tako dobimo naslov operanda. Na ta način lahko enostavno naslavljamo podatke neke tabele.

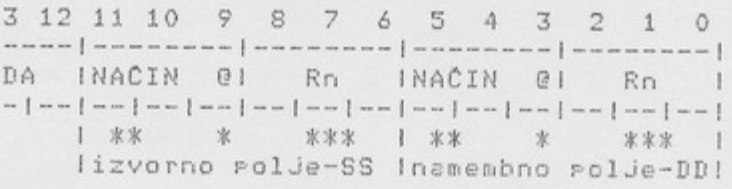
Registri R0-R5 se uporabljajo kot splošni registri. Register R6 je hardwarski kazalec na sklad, R7 pa se uporablja kot programski števec.
Načini naslavljanja omogočajo ukaze z enojnim operandom in ukaze z dvojnim operandom. Slika 2.1 prikazuje obliko instrukcijske besede za ukaze z enojnim operandom.



Slika 2.1

*---Bit, ki določa posredno ali neposredno naslavljanje
**--Določa način uporabe registra
***-Določa enega izmed registrov splošne uporabe

Obliko instrukcijske besede za ukaze z dvojnim operandom kaže slika 2.2.



Slika 2.2

Nacini neposrednega naslavljanja

-neposredno z registrom R
-neposredno prištevalni (R)+
-neposredno odstevalni -(R)
-neposredno indeksni X(R)

Pri neposrednem naslavljanju z registrom je vsebina registra uporabljena kot operand. Pri odštevalnem načinu naslavljanja se naslov, ki je v registru, najprej samodejno zmanjša za 2 in tako kaže na operand. Pri neposrednem prištevalnem načinu je na začetku izvajanja ukaza v registru naslov operanda, ki se po izvršitvi ukaza poveča in kaže na naslednjo besedo v pomnilniku. Pri indeksnem načinu nam vsota registra in vrednosti odmika X pomeni naslov operanda.
Kadar ima bit 3 izvornega ali namembnega polja vrednost "1", to označuje, da je izbran posredni način naslavljanja. Iz štirih osnovnih načinov dobimo štiri izvedene načine naslavljajna. Z znakom "@", ki ga postavimo pred navedbo operanda, ali z vstavitvijo oznake registra v oklepaje, označimo, da uporabljamo posredno naslavljanje.

Posredni načini naslavljanja so:

-posredni z registrom (R)
-posredni prištevalni @(R)+
-posredni odštevalni @-(R)
-posredni indeksni @X(R)

Načini naslavljanja s programskim števcem so:

-trenutni #n
-absolutni @#A
-relativni A
-posredni relativni @A

2.1 TABELA NAČINOV NASLAVLJANJA

Načini neposrednega naslavljanja:

nacin	bin. koda	ime	simbol	funkcija
0	000	neposredno	Rn	register vsebuje operand
2	010	pristevalno	(Rn)+	Register vsebuje naslov operanda. Vsebina se inkrementira po operaciji.
4	100	odštevalno	-(Rn)	Vsebina registra se dekrementira preden register naslovi pomnilnik.
6	110	indeksno	X(Rn)	Vrednost X,ki je shranjena v besedi,ki sledi instrukciji, se prišteje k (Rn) in dobimo vrednost operanda.

Posredni načini naslavljanja:

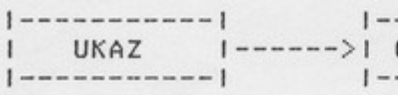
način	bin. koda	ime	simbol	funkcija
1	001	posredni z registrom	@Rn ali (Rn)	Register vsebuje naslov operanda
3	011	posredni prištevalni	@(Rn)+	Register se najprej uporabi kot kazalec na besedo,ki vsebuje naslov operanda.
5	101	posredni odštevalni	@-(Rn)	Register se dekrementira za 2 in se potem uporablja kot kazalec na besedo,ki vsebuje naslov operanda.
7	111	posredni indeksni	@X(Rn)	Vrednost X se sesteje z vrednostjo (Rn). Vsota se uporabi kot kazalec na besedo,ki vsebuje naslov operanda.

Naslavljanje s programskim števcem:

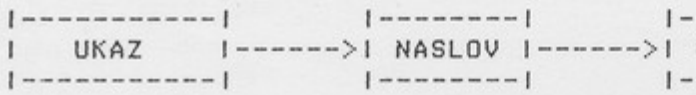
način	bin. koda	ime	simbol	funkcija
2	010	trenutni	n	Operand sledi ukazu
3	011	absolutni	@#A	Absolutni naslov operanda sledi ukazu.
6	110	relativni	A	Vrednost A, ki ji prištejemo naslov ukaza povečan za 4 je naslov operanda.
7	111	posredno relativni	@A	Vrednosti A prištejemo naslov ukaza povečan za 4 in dobimo kazalec na naslov operanda

2.2 GRAFIČNI PRIKAZ NAČINOV NASLAVLJANJA

Način 0: Registerski OPR R R vsebuje operand

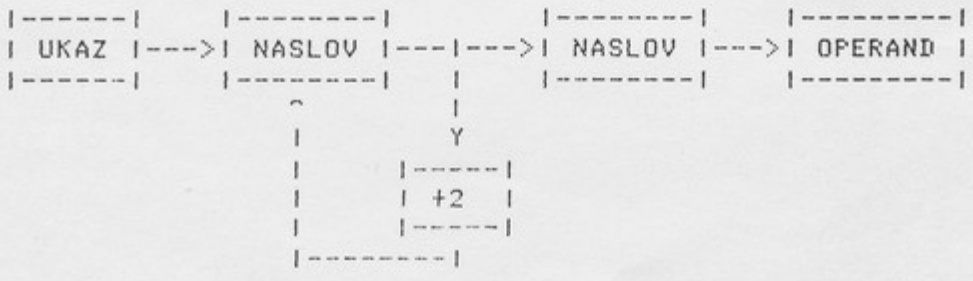


Način 1: Posredno naslavljanje z registrom
OPR (R) R vsebuje naslov operanda



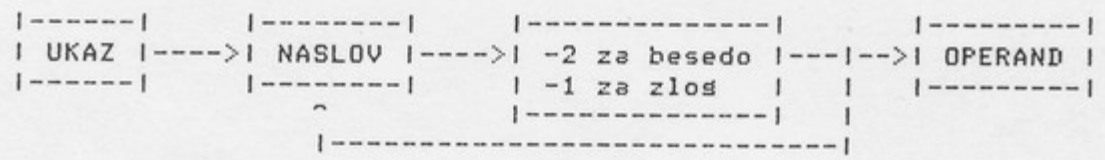
Način 2: Neposredni prištevalni način
OPR (R)+ R vsebuje naslov operanda.
Po izvršitvi ukaza se vsebina registra inkrementira.

Način 3: Posredno prištevalni način
OPR @(R)+ R vsebuje naslov naslova,
po operaciji se (R) inkrementira za 2.



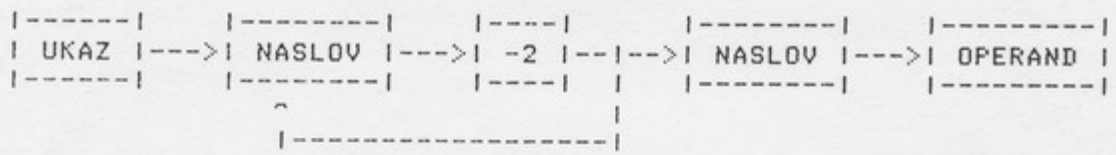
Način 4: Odstevalni način

OPR -(R) Po dekrementiranju (R), vsebuje R naslov operanda. Registra R6 in R7 se vedno dekrementirata za 2.



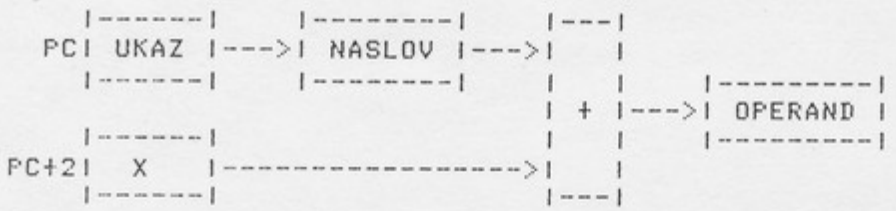
Način 5: Posredni odštevalni nacin

OPR @-(R) Po dekrementiranju (R), vsebuje R naslov naslova operanda.

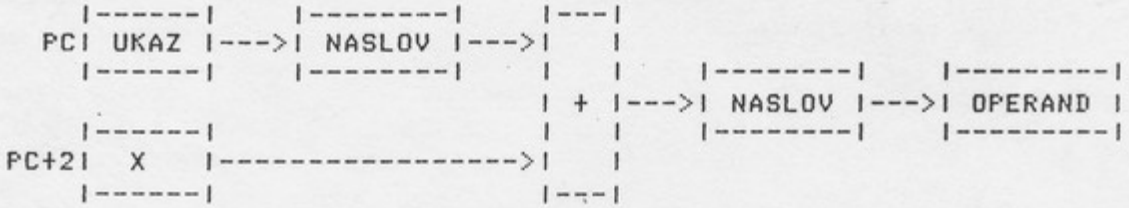


Način 6: Indeksni način

OPR X(R) (R)+X je naslov. X vsebuje beseda, ki sledi ukazu.

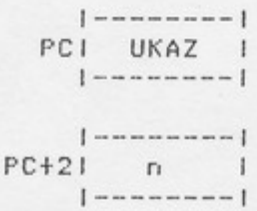


Način 7: Posredni indeksni način
OPR @X(R) (R)+X je naslov naslova operanda.X vsebuje beseda, ki sledi ukazu.



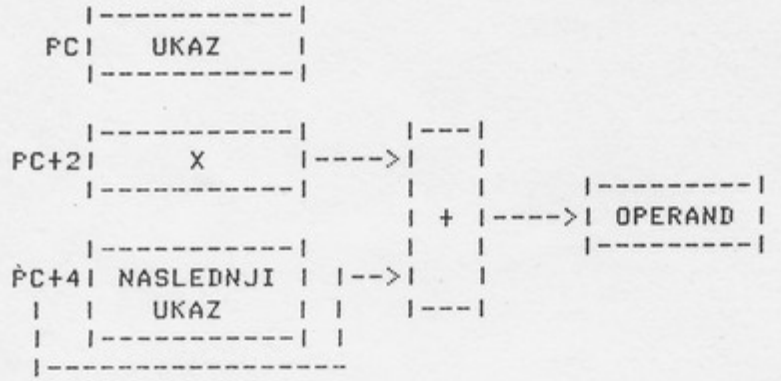
Načini naslavljanja s programskim števcem - Register = 7

Način 2: Trenutni način OPR #n Operand sledi ukazu

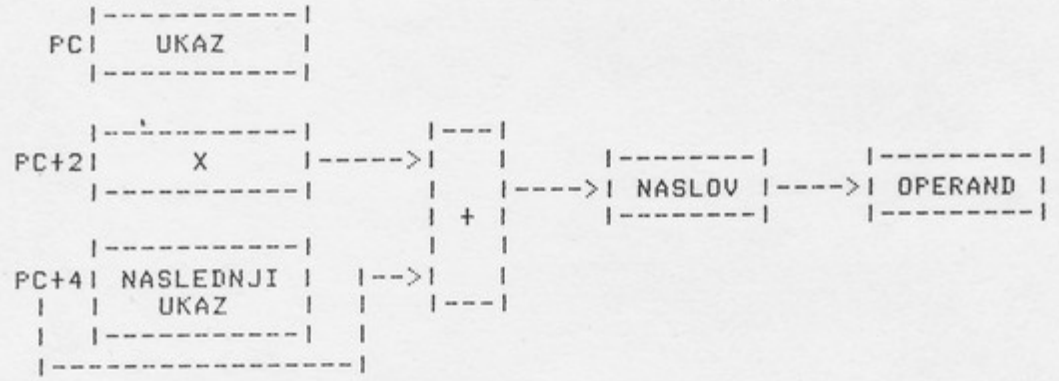


Način 3: Absolutni način OPR @#A Naslov operanda A vsebuje beseda, ki sledi ukazu.

Način 6: Relativni način OPR A Naslov operanda je PC+4+X. Nova vrednost programskega števca je PC+4.



Način 7: Posredno relativni način
OPR @A Naslov naslova operanda je PC+4+X. Nova vrednost programskega števca je PC+4.



3. NABOR INSTRUKCIJ

DELTA 16/BIT SLICE je procesor, ki v osnovnem instrukcijskem naboru emulira standardni nabor instrukcij procesorja PDP 11/34. V nadaljevanju je podan podroben opis standardnega instrukcijskega nabora.

SIMBOL	POMEN
MN	servisni ukaz (maintenance instruction)
SO	ukazi z enojnim operandom (single operand instruction)
DO	ukazi z dvojnim operandom (double operand instruction)
PC	ukazi za krmiljenje programa (program control instruction)
MS	razni ukazi (miscellaneous instructions)
CC	pogojna koda (condition code)
(x)	vsebina pomnilniške lokacije katere naslov je x
izv	izvorni naslov
nam	namembni naslov
tmp	vsebina začasnega registra
<---	postane,prenese se
(SP)+	vzeto iz HW sklada
-(SP)	dodano na HW sklad
&	logični IN
V	logični ALI
~	logični NE
R	vsebina registra
Rv1	vsebina registra, če je R lihi register
R,Rv1	32 bitno vrednost dobimo s povezavo registrov R in Rv1
M.P.I.	najvišje celo pozitivno stevilo: 077777 (beseda)
	(most positive integer) 177 (zlog)
M.N.I.	najnižje celo negativno število: 100000 (beseda)
	(most negative integer) 200 (zlog)

3.1 STANDARDNI NABOR

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
ADC ADCB prištej prenos (add carry)	SO	0055DD 1055DD	(nam) <--- (nam)+C

N: enak '1', če rezultat < 0
Z: enak '1', če rezultat = 0
V: enak '1', če je rezultat M.P.I. in če je C = 1
C: enak '1', če (nam) = -1 in C = 1

Namembnemu operandu prišteje vsebino C-bita

ADD prištej (add)	DO	06SSDD	(nam) <--- (izv)+(nam)

N: enak '1', če rezultat < 0
Z: enak '1', če rezultat = 0
V: enak '1', če rezultat operacije povzroči aritmetično prekoračitev, če sta oba operanda imela enak predznak, rezultat pa dobi nasprotni predznak.
C: enak '1', če je bil izvršen prenos iz najvišjega bita rezultata

Sešteje izvorni in namembni operand in rezultat shrani na namembnem naslovu. Prvotna vrednost na namembnem naslovu se izgubi. Vsebina izvornega polja se ne spremeni. Izvedeno je seštevanje z dvojiškim komplementom.

ASH aritmetični pomik (arithmet. shift)	DO	072RSS	R <--- R vsebina registra aritmetično pomaknjena za NN mest v desno ali v levo, pri čemer je NN = (izv) <5:0>

N: enak '1', če rezultat < 0
Z: enak '1', če rezultat = 0
V: enak '1', če se predznak registra med operacijo menja. Enako '0', če je NN = 0
C: napolnjen z zadnjim bitom, ki izpade iz registra. Enak '0', če je NN = 0

Vsebina registrov se pomakne v levo ali desno tolikokrat, kot je določeno s številcem pomikov (biti 0-5 izvornega operanda). Številec pomikov predstavlja torej nižjih šest bitov izvornega operanda. Številec lahko šteje od -32 do +31. Negativno število pomeni pomik desno, pozitivno pa pomik levo.

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
ASHC kombinirani aritmetični pomik (arithmetic shift combined)	DO	073RSS	tmp <--- R, R v 1 tmp <--- tmp pomaknjen NN krat R <--- tmp <31:16> R v 1 <--- tmp<15:00> dvojna beseda (R, R v 1) je pomaknjena NN krat levo ali desno NN = (izv) <5:0> tmp = začasna dvojna beseda

N: enak '1', če rezultat < 0
Z: enak '1', če rezultat = 0
V: enak '1', če se vrednost bita, ki označuje predznak, med pomikanjem vsaj enkrat zamenja.
C: napolnjen z najvišjim bitom pri pomiku v levo ali najnižjim bitom, pri pomiku v desno. Po izvršitvi ukaza vsebuje zadnji bit 32 bitne besede, ki je pri pomikih izpadel.

Vsebino registrov R in R v 1 skupaj smatramo, kot enojni, 32-bitni operand in se pomakne za toliko mest kot je to določeno s števcem pomikov (biti<5:0>) izvornega operanda. Zgornji biti rezultata <31:16> so v registru R, spodnji biti <15:0> pa v R v 1. Števec šteje od -32 do +31. Negativno število predstavlja pomik v desno, pozitivno pa pomik v levo. Če števec kaže vrednost 0 pomeni, da pomika ni, setirajo pa se biti pogojnega koda. Ti se vedno nastavijo glede na 32-bitni rezultat.
POMNI: 1. Pri pomiku v desno se najvišji bit (predznak) prepisuje iz leve proti desni. Pri pomiku v levo se najnižji bit polni z ničlami, C-bit se napolni z zadnjim bitom, ki je izpadel.
2. Pri pomiku v levo pride do prekoračitve(integer overflow), kadar se katerikoli bit, pomaknjen na mesto bita za predznak, razlikuje od začetne vrednosti tega bita.

ASL ASLB aritmetični pomik v levo (arithmetic shift left)	SO	0063DD 1063DD	(nam)<---(nam) premaknjen za eno mesto v levo

N: enak '1', če je najvišji bit rezultata enak '1' (rezultat < 0)
Z: enak '1', če je rezultat = 0
V: napolnjen z vrednostjo 'EXCLUSIVE OR' nad bitoma N in C
C: napolnjen z najvišjim bitom pomaknjene besede

Vsi biti operanda se pomaknejo za eno mesto v levo. Najnižji bit se napolni z '0'. C-bit se napolni z najvišjim bitom, ki izpade iz besede. Ukaz ASL predstavlja množenje operanda z vrednostjo 2 z upoštevanjem predznaka, ter označitvijo prekoračitve.

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
ASR ASRB aritmetični pomik v desno (arithmetic shift right)	SO	0062DD 1062DD	(nam) <--- (nam) pomaknjen za eno mesto v desno

N: enak '1', če rezultat < 0 (najvišji bit rezultata enak '1')
Z: enak '1', če rezultat = 0
V: napolnjen z vrednostjo 'EXCLUSIVE OR' nad bitoma N in C
C: napolnjen z najnižjim bitom pomaknjene besede

Vsi biti namembnega operanda se pomaknejo za eno mesto v desno. Bit se pri pomiku prepisuje v sosednje desne bite. C-bit se napolni z najnižjim bitom operanda. Ukaz ASR predstavlja deljenje operanda z vrednostjo 2 in upoštevanjem predznaka, ter zaokrožitvijo na minus neskončno.

BCC preskoči, če C=0 (branch if carry clear)	PC	103000 plus odmik 8 bitov	PC <--- PC + (2 x odmik) če C = 0

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Testira vrednost bita C in povzroči preskok, če je C bit enak '0'

BCS preskoči, če C=1 (branch if carry set)	PC	103400 plus odmik 8 bitov	PC <--- PC + (2 x odmik) če C = 1

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Testira vrednost bita C in povzroči preskok, če je C bit enak '1'

BEQ skok, če enako 0 (branch if equal)	PC	001400 plus odmik 8 bitov	PC <--- PC + (2 X odmik) če Z = 1

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Testira vrednost Z-bita in povzroči preskok, če je Z bit enak '1'

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
BGE preskoči, če večje ali enako (branch if greater than or equal)	PC	002000 plus odmik 8 bitov	PC <--- PC + (2 x odmik) če je (N.EXOR.V) = 0

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Povzroči preskok, kadar sta N in V bit enaka (oba '1' ali oba '0')

BGT preskoči, če je strogo večji (branch if greater than)	PC	003000 plus odmik 8 bitov	PC <--- PC + (2 x odmik) če je Z v (N.EXOR.V) = 0

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Povzroči preskok, kadar je Z bit enak '0' in sta N ter V bit enakih vrednosti. BGT zato ne povzroči preskoka, če sledi ukazu, ki sešteje dve negativni števili, četudi pride do prekoračitve. Preskok se ne izvrši tudi po ukazu CMP, ki primerja negativni izvorni in pozitivni namembni operand, četudi pride do prekoračitve. Preskok se izvrši vedno, kadar ukaz sledi CMP ukazu, ki primerja pozitivni izvorni in negativni namembni operand. Skok se ne izvrši, če je rezultat predhodnega ukaza enak 0 (brez prekoračitve).

BHI preskoči, če je višje (branch if higher)	PC	101000 plus odmik 8 bitov	PC <--- PC + (2 X odmik) če je C = 0 in Z = 0

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Povzroči preskok, če rezultat prejšnjega ukaza ni bil enak '0' in ni povzročil prenosa (C-bit ni enak '0'). To se zgodi pri izvajanju ukaza CMP, ko ima izvorni operand večje neoznačeno število, kot namembni.

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
BHIS preskoči, če je višje ali enako (branch if higher or same)	PC	103000 plus odmik 8 bitov	PC <--- PC + (2 X odmik) če je C = 0

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Testira se stanje bita C in povzroči preskok, če je ta bit enak '0'.

BIC BICB briši bit (bit clear)	DO	04SSDD 14SSDD	(nam) <--- ~(izv) & (nam)

N: enak '1', če je najvišji bit rezultata enak 1
Z: enak '1', če je rezultat enak 0
V: enak 0
C: nespremenjen

Briše tiste bite namembnega operanda, ki ustrezajo postavljenim bitom v izvornem operandu. Prvotna vrednost namembnega operanda se izgubi, izvorni operand ostane nespremenjen.

BIS BISB postavi bit (bit set)	DO	05SSDD 15SSDD	(nam) <--- (izv) V (nam)

N: enak '1', če je najvišji bit rezultata enak 1
Z: enak '1', če je rezultat = 0
V: enak 0
C: nespremenjen

Izvrši logično operacijo 'INCLUSIVE OR' med obema operandoma in vpiše rezultat v namembno polje. Biti nastavljeni v izvornem operandu se nastavijo tudi v namembnem operandu. Vsebina namembnega operanda je izgubljena.

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
BIT BITB testiranje bitov (bit test)	DO	03SSDD 13SSDD	(nam) & (izv)

N: enak "1", če je najvišji bit rezultata enak 1
Z: enak "1", če je rezultat = 0
V: enak 0
C: nespremenjen

Izvrsi logično primerjavo med obema operandoma, ter ustrezno spremeni bite posojne kode. Oba operanda ostaneta nespremenjena. S tem ukazom lahko testiramo, če so vsi odsovarjajoči biti v namembnem in izvornem polju enaki.

BLE preskoči, če je manjše ali enako (branch if less than or equal to)	PC	003400 plus odmik 8 bitov	PC <--- PC + (2 X odmik) če je Z v (N EXOR N) =1

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Povzroči preskok če je Z = 1 ali če N in V nista enaka. Zato BLE vedno povzroči preskok, kadar sledi ukazu, ki sešteje dve negativni števili ali ukazu CMP, ki primerja med negativnim izvornim operandom in pozitivnim namembnim operandom. Preskok se ne izvrši nikoli kadar BLE sledi ukazu CMP, ki primerja pozitivni izvorni in negativni namembni operand. Preskok se izvrši vedno, kadar je rezultat prejšnjega ukaza enak 0.

BLO preskoči, če je nižje (branch if lower)	PC	103400 plus odmik 8 bitov	PC <--- PC + (2 X odmik) če je C = 1

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Testira se stanje C bita in povzroči preskok, če je C bit enak "1". Z ukazom lahko testiramo izvršitev prenosa pri izvajanju predhodnega ukaza.

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
BLOS preskoči, če je manjše ali enako (branch if lower or same)	PC	101400 plus odmik 8 bitov	PC <--- PC + (2 X odmik) če je C V Z = 1

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Povzroči preskok, če je prejšnja operacija povzročila prenos ali ničelni rezultat. BLOS je komplementaren ukazu BHI. Preskok se zgodi vedno, kadar ukaz BLOS sledi ukazu primerjanja, pri čemer ima izvorni operand enako ali manjšo nepredznačeno vrednost kot namembni operand.

BLT preskoči, če je manjše (branch if less than)	PC	002400 plus odmik 8 bitov	PC <--- PC + (2 X odmik) če je N .EXOR. V = 1

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Povzroči preskok, če je "EXCLUSIVE OR" med bitoma N in V enak "1". Preskok se izvrši vedno po seštetju dveh negativnih števil, četudi pride do prekoračitve. Ravno tako se zgodi preskok po primerjanju negativnega izvornega in pozitivnega namembnega operanda (četudi pride do prekoračitve). BLT nikoli ne povzroči preskoka, če sledi ukazu, ki primerja pozitivni izvorni in negativni namembni operand. Preskok se ne izvrši tudi v primeru, če je rezultat predhodne operacije enak "0" (brez prekoračitve).

BMI preskoči, če je negativno (branch if minus)	PC	100400 plus odmik 8 bitov	PC <--- PC + (2 X odmik) če je N = 1

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Testira se stanje bita N in povzroči preskok, če je N enak "1". Z ukazom testiramo predznak (najvisji bit) rezultata iz predhodne operacije.

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
BNE preskoči, če ni enak (branch if not equal)	PC	001000 plus odmik 8 bitov	PC <--- PC + (2 X odmik) če je Z = 0

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Testira se stanje bita Z in povzroči preskok, če je Z bit enak '0'. Ukaz je komplementaren ukazu BEQ. Uporabljamo ga za testiranje enakosti po izvršitvi ukaza CMP, t.j. testiranje, če so biti nastavljeni v namembnem operandu, nastavljeni tudi v izvornem operandu (po izvršitvi ukaza BIT) ter splošno za testiranje, da rezultat prejšnjega ukaza ni enak 0.

BPL preskoči, če je pozitivno (branch if plus)	PC	100000 plus odmik 8 bitov	PC <--- PC + (2 X odmik) če je N = 0

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Testira se stanje bita N in povzroči preskok, če je N enak '0'. Ukaz BPL je komplementaren ukazu BMI.

BPT točka pasti (breakpoint trap)	PC	000003	-(SP) <--- PS -(SP) <--- PC PC <--- (14) PS <--- (16)

N: napolnjen z vrednostjo novega PS
Z: napolnjen z vrednostjo novega PS
V: napolnjen z vrednostjo novega PS
C: napolnjen z vrednostjo novega PS

Izvede postopek prekinitve z vektorjem prekinitve na naslovu 14. Ukaz uporabljamo za prekinitev programa in vstop v program za odkrivanje napak. Pri uporabi ukaza moramo biti pazljivi, kadar se program odvija pod nadzorstvom programov za odkrivanje napak (ODT,XDT). Operacijska koda ukaza je fiksna.

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
BR brezpogojni pre- skok (uncond.branch)	PC	000400 plus odmik 8 bitov	PC <--- PC + (2 X odmik)

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Omogoča način za prenos nadzora nad odvijanjem programa znotraj naslovnega področja -128 do +127 besed z enobesednim ukazom. Skok je brezpogojni.

BVC preskoči, če V=0 (branch if V=0)	PC	102000 plus odmik 8 bitov	PC <--- PC + (2 X odmik) če je V = 0

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Testira se stanje V bita in povzroči preskok, če je V bit brisan. BVC je ukaz komplementaren ukazu BVS.

BVS preskoči, če je V = 1 (branch if V set)	PC	102400 plus odmik 8 bitov	PC <--- PC + (2 X odmik) če je V = 1

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Testira stanje V bita in povzroča preskok, če je V bit enak '1'. Ukaz BVS uporabimo za odkrivanje aritmetične prekoračitve rezultata v prejšnji operaciji.

CLR CLRB briši (clear)	SO	0050DD 1050DD	(nam) <--- 0

N: brisan
Z: nastavljen na '1'
V: brisan
C: brisan

Vsebina specificirane lokacije se zamenja z ničlami.

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
C briše izbrane bite pogojnega koda (clear selected condition code bits)	CC	000240 plus maska 4 bite	PSW <3:0> <--- PSW <3:0> [~maska<3:0>]

Briše bite pogojnega koda. Izbrani biti se lahko brišejo istočasno. Briše tiste bite pogojnega koda, ki ustrezajo bitom <3:0> nastavljenim v operatorju, torej bite 0, 1, 2 in 3. Bit 4 je enak 0.

CCC briši vse bite pogojnega koda (clear all CC bits)	CC	000257	N,Z,V,C <--- 0
CLC briši C bit (clear C bit)	CC	000241	C <--- 0
CLN briši N bit (clear N bit)	CC	000250	N <--- 0
CLV briši V bit (clear V bit)	CC	000242	V <--- 0
CLZ briši Z bit (clear Z bit)	CC	000244	Z <--- 0
CMP CMPB primerjaj (compare)	DO	02SSDD 12SSDD	(izv) - (nam)

N: enak '1', če je rezultat < 0
Z: enak '1', če je rezultat = 0
V: enak '1', če nastopi aritmetična prekoračitev; to pomeni, da imata operanda nasprotni predznak, namembni operand pa enak predznak kot rezultat
C: enak '1', če je prenos iz najvišjega bita, torej kadar je (izv) + ~(nam) + 1 manj kot 2**16

Primerja izvorni in namembni operand ter nastavi bite pogojnega koda, ki jih lahko kasneje uporabimo za logično in aritmetično pogojno razvejanje. Oba operanda ostaneta nespremenjena. Po primerjavi običajno izvedemo testiranje bitov pogojnega koda in izvedemo pogojni preskok.

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
COM	SO	0051DD	(nam) <--- ~(nam)
COMB		1051DD	
komplement			
(complement)			

N: enak '1', če je najvišji bit rezultata enak 1
Z: enak '1', če je rezultat enak 0
V: enak '0'
C: enak '1'

Zamenja vsebino namembnega polja z njenim logičnim komplementom. Vsak bit, ki je enak 0 je postavljen na 1 in obratno.

DEC	SO	0053DD	(nam) <--- (nam) - 1
DECB		1053DD	
zmanjšaj			
(decrement)			

N: enak '1', če je rezultat < 0
Z: enak '1', če je rezultat = 0
V: enak '1', če je bil operand (nam) M.N.I. (največje negativno število)
C: nespremenjen

Odšteje 1 od vsebine namembnega operanda.

DIV	DO	071RSS	R, Rv1 <--- R, Rv1/(izv)
deli			
(divide)			

N: enak '1', če je kvocient < 0 ; (nedoločen, če je V=1)
Z: enak '1', če je kvocient = 0 ; (nedoločen, če je V=1)
V: enak '1', če je izvorno polje enako 0 ali, če kvocienta ne moremo predstaviti kot 16 bitni dvojiški komplement. R, Rv1 sta nedoločljiva, če je V=1 in C=0
C: enak '1', če skušamo deliti z 0

32 bitni dvojiški komplement (integer), ki je v registrih R in Rv1 se deli z izvornim operandom. Kvocient gre v register R in ostanek v Rv1. Oba sta enakega predznaka. Register R mora biti sod.

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
EMT emulatorska past (emulator trap)	PC	104000 do 104377	-(SP) <--- PS -(SP) <--- PC PC <--- (30) PS <--- (32)

N: napolnjen z vrednostjo novega PS
Z: napolnjen z vrednostjo novega PS
V: napolnjen z vrednostjo novega PS
C: napolnjen z vrednostjo novega PS

Operacijske kode od 104000 do 104377 so EMT ukazi in se jih uporablja za prenos informacij emulacijskim rutinam. Vektor pasti (trap vektor) za EMT se nahaja na naslovu 30. Novi programski števec (PC) dobi vrednost na naslovu 32. EMT in TRAP ukaza sta si po funkciji identična. Programska oprema delta racunalnikov uporablja EMT ukaz, zato se za splošno rabo priporoča ukaz TRAP.

HALT ustavi (stop)	MS	000000	

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Zaustavi procesorjeve operacije. Krmiljenje procesorja preide v roke konzoli. Prenosi na podatkovno vodilo se zaključijo. Operacije procesorja se nadaljujejo s pritiskom <CONTINUE> tipke na konzoli.

INC INCB povečaj (increment)	SO	0052DD 1052DD	(nam) <--- (nam) + 1

N: enak '1' če rezultat < 0
Z: enak '1' če rezultat = 0
V: enak '1' če je bila vrednost (nam) pred operacijo M.P.I.
C: neprizadet

Namembnemu operandu prišteje '1'

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
IOT vhodno/izhodna past (I/O trap)	PC	000004	-(SP) <--- PS -(SP) <--- PC PC <--- (20) PS <--- (22)

N: napolnjen z vrednostjo novega PS
Z: napolnjen z vrednostjo novega PS
V: napolnjen z vrednostjo novega PS
C: napolnjen z vrednostjo novega PS

Izvrši zaporedje operacij za past z vektorjem pasti na naslovu 20. Pri programski opremi, za papirni trak se ponavadi kliče VH/IZH izvajalna rutina IOT, pri operacijskih sistemih z diski pa rutina za sporočilo o napakah. V spodnjem zlogu ni prenesena nobena informacija.

JMP skoči (Jump)	PC	0001DD	PC <--- (nam)

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

JMP zagotavlja bolj sibčno izvedbo, kot so vejitveni ukazi. Skočni ukaz JMP ni omejen s +177/8 in -200/8, kot vejitveni ukazi. JMP generira novo besedo in je zaradi nje počasnejši od vejitvenih ukazov. Prenos se lahko krmili na katerokoli lokacijo v pomnilniku (nobenih omejitvenih vrednosti). Uporablja se z vsemi načini naslavljanja z izjemo načina 0. Izvajanje JMP ukaza z načinom naslavljanja 0 povzroči nedovoljeno stanje ukaza. Posredni način z registrom je dovoljen in povzroči prenos programskega krmiljenja na naslov, ki ga hrani določeni register.
POMNI: ukazi so podatki shranjeni v besedi in morajo biti dostavljeni iz sodo oštevilčenih naslovov. Ko procesor odkloni dostavo ukaza iz liho ostevilčenega naslova, to povzroči past zaradi napake v razmerju naslovov.

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
JSR skok v podprogram (jump to sub- routine)	PC	004RDD	(tmp) <--- (nam) tmp je interni začasni register procesorja -(SP) <--- reg vsebina registrov se porine na sklad procesorja reg <--- PC PC sedaj kaže na naslov pod- programa

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Pri izvajanju JSR ukaza, se stara vsebina določenega registra (povezovalni kazalec) avtomatično porine na sklad, kamor kaže R6. V register se vpiše nova povezovalna informacija. Podprogrami vgnezdeni v podprogramih do katerekoli globine, lahko kličejo preko istega povezovalnega registra. Tako ni nobene potrebe niti po preračunavanju maksimalne globine, na kateri bo klican nek določen podprogram, niti ni potrebno vsaki rutini dodati ukazov za shranjevanje in ponovno postavljanje shranjenih povezovalnih kazalcev. Ko so potem vse povezave shranjene v R6 skladu na način, ki omogoča ponovno vstopanje v podprograme, je omogočena prekinitev izvajanja podprograma in ponoven vstop v izvajanje istega podprograma s prekinitvenimi servisnimi rutinami.
Izvajanje začetnega podprograma se lahko nadaljuje, ko je zadoščeno vsem ostalim zahtevam. Ta proces se imenuje gnezdenje in lahko poteka do kateregakoli nivoja.
JSR PC (nam) je poseben primer za klice podprogramov, ki prenašajo parametre preko registrov splošne uporabe. JSR, s programskim števcem kot povezovalnim registrom, prihrani uporabo posebnega registra.
POMNI: Če je register določen s prvim registerskim operandom, v izračunu drugega operanda (namembnega) samodejno povečan ali samodejno pomanjšan (autoincrement/autodecrement), se na sklad porine spremenjena vsebina registra.
PRIMER: JSR R5,@(R5)+ povzroči prenos spremenjene vrednosti R5 na sklad.

MARK	PC	0064NN	SP <--- PC + 2 X NN PC <--- R5 R5 <--- (SP)+ NN = število parametrov

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Je del dogovorjenih standardnih povratkov iz PDP-11 podprogramov. Ukaz MARK pospeši postopke čiščenja procesorjevega sklada, pri izhodu iz podprograma. Format zbirnega ukaza je: MARK N.

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
MFPD prenesi iz prejšnjega podatkovnega prostora MFPI prenesi iz prejšnjega ukaznega prostora	MS	1065SS 0065SS	tmp <--- (izv) -(SP) <--- tmp

N: enak '1', če izvorni op. < 0
Z: enak '1', če izvorni op. = 0
V: enak 0
C: nespremenjen

Porine besedo na tekoče mesto v R6 skladu, iz naslova v prejšnjem prostoru, ki ga določa programsko stanje PS<13:12>. Izvorni naslov se izračuna s pomočjo tekočih registrov in enote za upravljanje s pomnilniškim prostorom. Ko se izvaja MFPI in sta prejšnji in tekoči način uporabniška (USER MODE), deluje MFPI enako kot MFPD.

MOV MOVB prenesi (move)	DO	01SSDD 11SSDD	(nam) <--- (izv)

N: enak '1' ce (izv) < 0
Z: enak '1' ce (izv) = 0
V: enak '0'
C: nespremenjen

Prenese izvorni operand v namembno lokacijo. Prejšnja vsebina na namembnem naslovu se izgubi. Izvorni operand ostane nespremenjen. MOVB v register (edini med zlogovnimi ukazi). Razširi 'najbolj pomemben bit nižjega zloga (razširitev znaka).

MTPD prenesi v prejšnji podatkovni prostor MTPI prenesi v prejšnji ukazni prostor (move to previous data/instruction space)	MS	1066DD 0066DD	tmp <--- SP+ (dst) <--- tmp

N: enak '1', če je izv < 0
Z: enak '1', če je izv = 0
V: enak '0'
C: nespremenjen

Ta ukaz potegne iz R6 sklada besedo, določeno s PS <15:14> in jo shrani na naslov v prejšnjem prostoru določenim s PS <13:12>. Namembni naslov se izračuna s pomočjo tekočih registrov in enote za upravljanje s pomnilniškim prostorom.

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
MUL pomnoži (multiply)	DO	070RSS	R,Rv1 <--- R X (src)

N: enak '1', če je produkt < 0
Z: enak '1', če je produkt = 0
V: enak '0'
C: enak '1', če je produkt manjši od -2**15 ali večji oziroma enak 2**15. Posojne kode so postavljene sledé na 32 bitni rezultat, čeprav je R lih.

Vsebina namembnega in izvornega registra se vzame kot celoštevilčni dvojiški komplement. Produkt se shrani v namembni register in dodatni register (če je R sod). Če je R lih se shrani samo spodnji del produkta. Zbirna sintaksa je: MUL S,R.

NEG NEGB negiraj (dvojiški komplement) (negate)	SO	0054DD 1054DD	(nam) <--- -(nam)

N: enak '1', če je rezultat < 0
Z: enak '1', če je rezultat = 0
V: enak '1', če je rezultat M.N.I.
C: enak '0', če je rezultat = 0, sicer je enak '1'

Zamenja vsebino namembnega naslova z njenim dvojiškim komplementom. Vrednost 100000 se zamenja sama s seboj.

RESET	MS	000005	

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Za 100 ms pošlje na UNIBUS signal INIT. Vse naprave se postavijo v stanje, v kakršno je bilo ob vključitvi.

ROL ROLB rotiraj v levo (rotate left)	SO	0061DD 1061DD	(nam) <--- (nam) krožno pomaknjen v levo za eno mesto

N: enak '1', če je enak '1' najvišji bit v besedi rezultata (rez<0)
Z: enak '1', če so vsi biti rezultata enaki 0
V: napolnjen z vrednostjo N .EXOR. C, kjer sta N in C postavljena ob zaključitvi operacije.
C: vsebuje najvišji bit namembnega polja

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
ROR RORB rotiraj desno (rotate right)	SO	0060DD 1060DD	(nam) <--- (nam) Krožno pomaknjen desno za eno mesto

N: enak "1", če je enak "1" najvišji bit rezultata
Z: enak "1", če so vsi biti rezultata enaki "0"
V: napolnjen z vrednostjo N .EXOR. C, kjer sta N in C postavljena ob zaključitvi operacije.
C: napolnjen z najnižjim bitom namembnega polja

Najnižji bit se naloži v C bit, prejšnja vsebina C bita pa se naloži v najvišji bit namembnega polja.

RTI vrnitev iz prekinitve (return from interrupt)	MS	000002	PC <--- (SP)+ PS <--- (SP)+

N: napolnjen iz tekočega R6 sklada
Z: napolnjen iz tekočega R6 sklada
V: napolnjen iz tekočega R6 sklada
C: napolnjen iz tekočega R6 sklada

Uporablja se za izhod iz servisnih rutin za obdelavo prekinitev ali pasti. PC in PS se obnovita iz R6 sklada. Če RTI postavi v PS T-bit, nastopi pred izvršitvijo naslednjega ukaza sledna past (TRACE TRAP). Če se RTI izvaja v nadzornem načinu (SUPERVISOR MODE), potem biti, ki označujejo prejšnji in tekoči način v obnovljeni PSW, ne smejo biti postavljeni na "KERNEL". Ko se RTI izvaja v uporabniškem načinu (USER MODE), so lahko biti, ki označujejo prejšnji in tekoči način v obnovljeni PSW, nastavljeni le na uporabniški način. RTI ne more izbrisati bita 11 v PSW, če je bil le ta postavljen.

RTS vrnitev iz podprograma (return from subroutine)	PC	00020R	PC <--- (reg) (reg) <--- (SP)+

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Naloži vsebino registra v PC in potegne vrhnji element iz R6 sklada v določeni register. Vrnitev iz podprograma, ki ne dovoljuje ponovnih vstopov, je realizirana preko istega registra, ki je bil uporabljen pri klicu. Podprogrami klicani z: JSR R5, lahko poberejo parametre z naslednjimi načini naslavljanja: (R5)+, X(R5) ali @X(R5), izhod pa je omogočen z RTS R5.

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
RTT vrnitev iz prekinitve (return from interrupt)	MS	000006	PC <--- (SP)+ PS <--- (SP)+

N: napolnjen iz tekočega R6 sklada
Z: napolnjen iz tekočega R6 sklada
V: napolnjen iz tekočega R6 sklada
C: napolnjen iz tekočega R6 sklada

Enako kot RTI ukaz se uporablja za izhod iz rutin za servisiranje pasti in prekinitev. Staro stanje PC in PS se ponovno vzpostavi iz procesorjevega sklada. Če RTI postavi na '1' T-bit v PS, to povzroči sledno past pred izvajanjem naslednjega ukaza. Razlika med RTI in RTT je v tem, da RTT zadrži sledno past, RTI pa jo dovoljuje takoj. Če v tem primeru nastopi zahteva po sledni pasti, se bo naslednji ukaz po RTT izvršil pred naslednjo T-pastjo. Ko se RTT izvaja v nadzornem načinu, biti, ki določajo prejšnji in tekoči način v ponovno postavljeni PS, ne smejo biti postavljeni na način KERNEL. Ko se RTT izvaja v uporabniškem načinu, morajo biti biti, ki določajo način v PS, postavljeni na uporabniški način. RTT ne more brisati PS <11>, če je bil le ta ravno nastavljen.

SBC SBCB odštej prenos (subtract carry)	SO	0056DD 1056DD	(nam) <--- (nam)-C

N: enak '1', če je rezultat < 0
Z: enak '1', če je rezultat = 0
V: enak '1', če je (nam) = M.N.I.
C: enak '1', če je bil (nam) = 0 in če je bil C = '1' pred izvršitvijo instrukcije

Namembnemu operandu odšteje vsebino C bita

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
S nastavi izbrane pogojne kode (set selected condition codes)	CC	000260 plus maska 4 LSB bite	PSW <3:0> <--- PSW <3:0> V V mask <3:0>
SCC postavi vse pogojne kode	CC	000277	N,Z,V,C <--- 1
SEC postavi C (set C)	CC	000261	C <--- 1
SEN postavi N (set N)	CC	000270	N <--- 1
SEV postavi V (set V)	CC	000262	V <--- 1
SEZ postavi Z (set Z)	CC	000264	Z <--- 1

Postavi bite, ki pripadajo določenim pogojnim kodam. Izbrane kombinacije teh bitov so lahko nastavljene skupaj. Spremeni se nastavitev bitov, ki ustrezajo bitom <3:0>.

SOB odštej 1 in skoči če rezultat <> 0 (subtract 1 and branch if <> 0)	PC	077R00 plus odmik 6 bitov	R <--- R-1 če rezultat <> 0 potem PC <--- PC-(2 X odmik)

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Vsebina v registru se zmanjša za 1. Če rezultat ni enak 0, se dvakratna vrednost odmika odšteje od PC, ki kaže na naslednjo besedo. Odmik nakazuje 6-bitno pozitivno število. Ta ukaz omogoča hitro in učinkovito krmiljenje zank. Zbirna sintaksa: SOB R,A; A je prenosni naslov, v primeru, da zmanjšani R ni enak '0'.
Ukaz SOB ne more izvesti skoka naprej!

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
SUB odštej (subtract)	DO	16SSDD	(nam) <--- (nam)-(izv) (nam) <--- (nam)+~(izv)+1

N: enak '1', če je rezultat < 0
Z: enak '1', če je rezultat = 0
V: enak '1' v primeru aritmetične prekoračitve rezultata. To pomeni, če sta bila operanda nasprotnih predznakov, da je predznak izvora enak predznaku rezultata.
C: postavljen, če obstaja prenos (borrow) v najvišji bit rezultata.

SWAB zamenjava zlogov (swap bytes)	SO	0003DD	zlog 1 <--- zlog 0 zlog 0 <--- zlog 1

N: enak '1', če je najvišji bit v nižjem zlogu (bit 7) rezultata enak '1'
Z: enak '1', če je nižji zlog rezultata enak '0'
V: enak '0'
C: enak '0'

V namembni besedi zamenja višji in nižji zlog. Namembni naslov mora biti beseda.

SXT razširitev pred- znaka (sign extend)	SO	0067DD	(nam) <--- 0; ce N = 0 (nam) <--- -1; ce N = 1

N: nespremenjen
Z: enak '1', ce je N-bit = 0
V: enak '0'
C: nespremenjen

Če je N-bit posojne kode '1', se v namembni operand vpiše '-1'. Če je N-bit enak '0', se v namembni operand vpiše '0'. Ukaz je uporaben v aritmetiki z večkratno natančnostjo, saj dovoljuje razširitev znaka preko več besed.

MNEMONIK UKAZA	TIP	KODA OPERACIJE	OPERACIJA
TRAP programska past	PC	104400 do 104777	-(SP) <--- PS -(SP) <--- PC PC <--- (34) PS <--- (36)

N: napolnjen z vrednostjo novega PS
Z: napolnjen z vrednostjo novega PS
V: napolnjen z vrednostjo novega PS
C: napolnjen z vrednostjo novega PS

Kode 104400 do 104777 so trap ukazi. TRAP in EMT ukaza sta si po funkciji identična. Razlika je v naslovu vektorjev prekinitve, ki je za TRAP ukaz na naslovu 34, za EMT pa na naslovu 30. Za splošno rabo se priporoča ukaz TRAP, ker ukaz EMT uporablja Delta software.

TST TSTB testiraj (test)	SO	0057DD 1057DD	tmp <--- (nam)

N: enak '1', če je rezultat < 0
Z: enak '1', če je rezultat = 0
V: enak '0'
C: enak '0'

Nastavi pogojne kode N in Z glede na vsebino namembnega naslova.

WAIT čakaj na prekini- tev (wait for interr- upt)	MS	000001	

N: nespremenjen
Z: nespremenjen
V: nespremenjen
C: nespremenjen

Procesorju je s tem omogočeno, da zapusti podatkovno vodilo in se odreče zahtevam po njem, medtem pa čaka na zunanjo prekinitev. Pri WAIT ukazu kaže PC na ukaz, ki sledi WAIT operaciji. Potem, ko prekinitev povzroči, da se PS in PC naložita na sklad, se shrani naslov ukaza, ki sledi WAIT ukazu. Izhod iz rutine za servisiranje prekinitev (izvajanje RTI ukaza) povzroči nadaljevanje prekinjenega procesa pri ukazu, ki sledi WAIT.

XOR (exclusive OR)	DO	077RDD	(nam) <--- R .EXOR. (nam)

N: enak '1', če je rezultat < 0
Z: enak '1', če je rezultat = 0
V: vedno enak '0'
C: nespremenjen
Rezultat R .EXOR. nam. operand, se shrani na namembni naslov. Vsebina registra ostane nespremenjena.

3.2 DODATEN NABOR INSTRUKCIJ

3.2.1 CIS (commercial instruction set)

Centralna procesna enota DELTA 16/BIT SLICE je aparaturno tako zasnovana, da omogoca realizacijo instrukcijskega nabora CIS. Realizacija zahteva v aparaturnem pogledu razsiritev registerskega niza ALU enote in povecanje obsega mikroprogramskega pomnilnika. Operandi CIS instrukcij so nizi znakov ali stevil, zato je potreben daljsi cas za izvedbo take instrukcije. Z namenom, da se cas od dolocene prekinitvene zahteve pa do odgovora na tako zahtevo zmanjsa, je vpeljan mehanizem suspenzije, ki omogoca regularno prekinitev izvajanja CIS instrukcije na mikro nivoju. Po servisni prekinitveni zahtevi se nadaljuje izvajanje prekinjene CIS instrukcije. Podrobnejsi opis postopka suspenzije je podan v poglavju 7.1.5
CIS nabor zahteva poleg osnovnih aritmeticnih operacij se BCD aritmetiko. Ker rezine 2903 nimajo v svojem instrukcijskem naboru BCD operacij, je za njihovo realizacijo potrebna kombinirana resitev (mikroprogramska in aparaturna). Po aparaturni strani so v ta namen dodana vrata Y59 (list A9), ki preprecijo prenose med rezinami v vseh tistih CIS mikroinstrukcijah (B3 GR * L="0"),kjer zelimo BCD aritmeticno operacijo (signal A9 CZERO L/PL/="0").
V dekodirni logiki in v enoti pasti je CIS nabor ze upostevan, njegovo aktiviranje pa omogoca signal CIS ATTACHED L, kadar se postavi v logicno niclo (s priključitvijo CIS modula). Vsi CIS ukazi spadajo v grupo 8 in se v sedanji verziji detektirajo kot pasti.

3.2.2 Uporabniski nabor instrukcij

Poleg standardnega nabora instrukcij, ki je realiziran v okviru centralne procesne enote DELTA 16/BIT-SLICE, obstajajo moznosti za realizacijo:

a) instrukcijskega nabora procesorja s plavajoco piko(FP11-A)
b) CIS nabora
c) uporabniskega nabora instrukcij

Ker ima emulator moznost mikroprogramiranja je na tej osnovi mozno realizirati dolocen uporabniski instrukcijski nabor. Vpeljava takega nabora je mozna na vec nacinov. Najenostavnejsi nacin, ki se izogne vsaki aparaturni modifikaciji (spremembe vezja, zamenjava PAL-ov ali vsebin v PROM-ih) je uporaba operacijskih kod iz CIS nabora. Uporabniskim instrukcijam pripisemo dolocene CIS operacijske kode, v PROM-e za generiranje zacetnih eksekucijskih mikroprogramskih naslovov (X26, X28 in X29) vpisemo zacetne naslove ustreznih mikroprogramskih rutin in vpisemo v mikroprogramski pomnilnik dodaten mikroprogram, ki realizira uporabnisko instrukcijo.
Na ta nacin je dana moznost uporabniku, da dolocene algoritme realizira na mikroprogramski nacin. Hkrati je omogocena prekinitev casovno daljsih mikroprogramskih sekvenc s pomocjo mehanizma suspenzije.

4. OPIS APARATURNE OPREME

4.1 BLOK SHEMA DELTA 16/BIT SLICE CPE

Prikazana na sliki 4.1

4.2 PODATKOVNA VODILA

Shema vodil na sliki 4.1a prikazuje interna vodila med posameznimi enotami procesorja in priključitev na univerzalno vodilo (Unibus).
Enosmerno vodilo SD omogoca prenos podatkov od SWAP MUX-a (zamenjava zlogov) do naslednjih ponorov:

- preko D/R vmesnika na univerzalno vodilo (16 bitov)
- statusne besede - PSW (9 bitov)
- preko BCD/CC MUX-a na Y vodilo 2904 (4 biti)
- preko DA-selektorja na DA vhod ALU rezin (16 bitov)
- preko MSI-tri stanjskih vmesnikov na Y vodilo ALU rezin (16 bitov)
- instrukcijskega registra (16 bitov)
- registrov virtualnega naslavljanja PAR, PDR, SR0

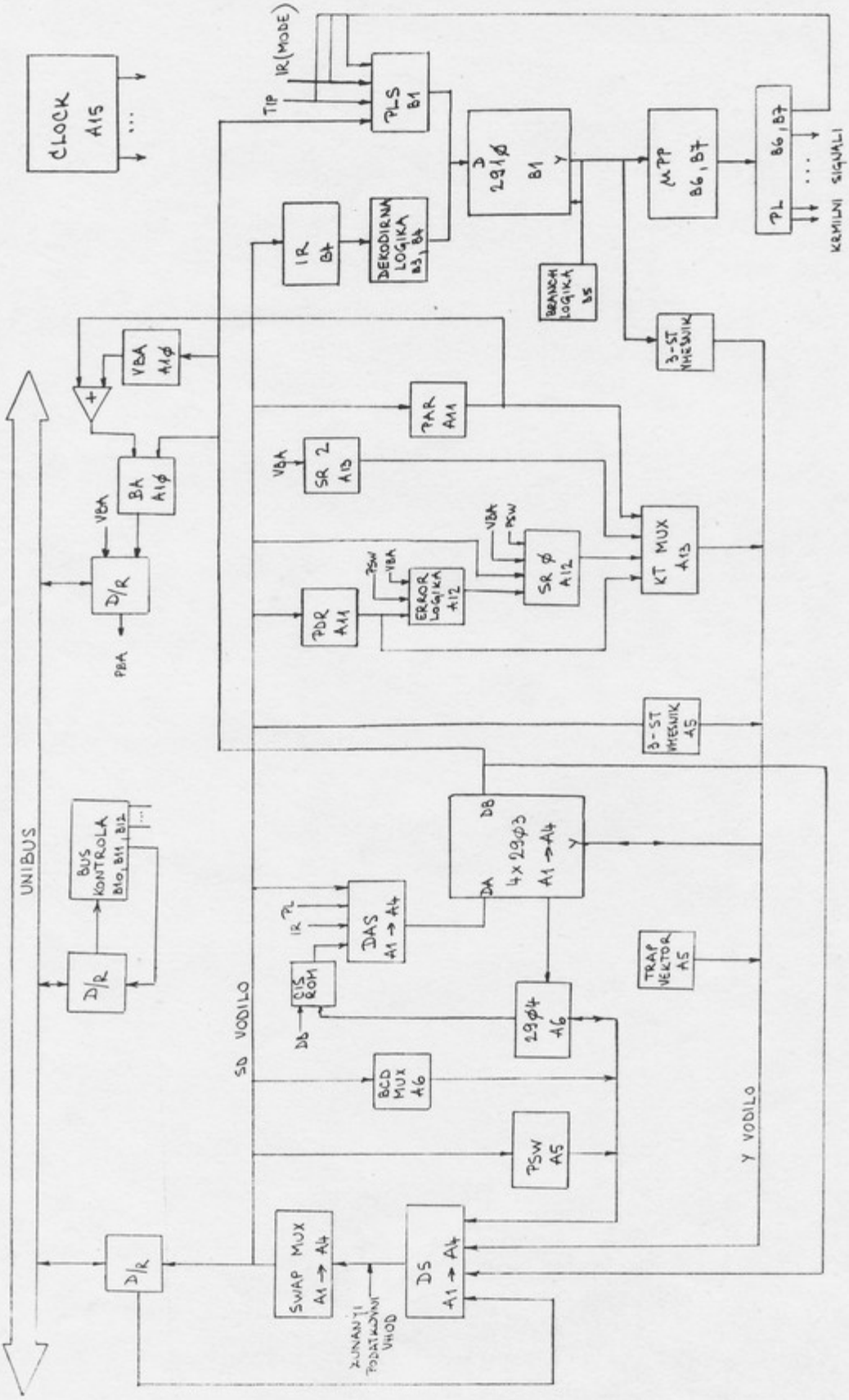
Podatkovni DS MUX izbira med naslednjimi izvori:

- vsebino iz glavnega pomnilnika preko D/R vmesnika (16 bitov)
- statusno besedo-PSW (13 bitov)
- DB vodilom (16 bitov)
- ALU Y vodilom

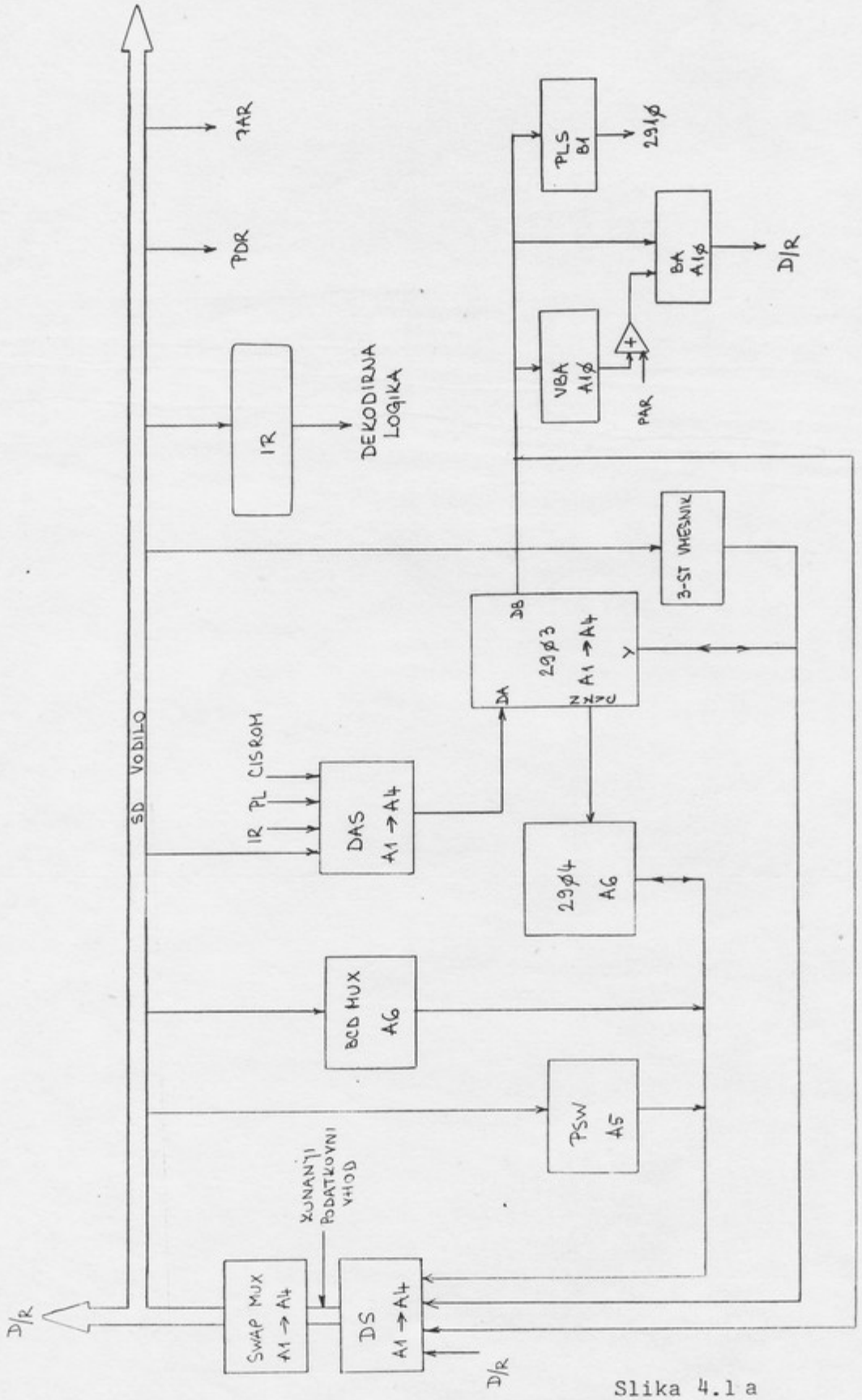
in jih posreduje izbiralniku za zamenjavo zlogov (SWAP MUX-u).
Izhod DS MUX-a je mozno krmiliti v visokoimpedancno stanje, zato je na tem mestu mozen sprejem 16 bitnih podatkov iz zunanjih modulov (procesor s plavajoco piko, CACHE pomnilnik).
V ALU generiran naslov (16 bitov) se preko DB vodila posreduje VBA registru oz. se v primeru neaktivnosti mehanizma za izracun fizicnih naslovov vpisuje neposredno v BA register, ki posreduje naslov glavnemu pomnilniku preko D/R vmesnikov. Shema vodil omogoca tudi vse funkcije konzole, ki se nanasajo na delo z ALU registerskim nizom, statusno besedo (PSW) in registri PAR, PDR, SR0, SR2.

4.2.1 Aritmeticna logicna enota (ALU-AM2903)

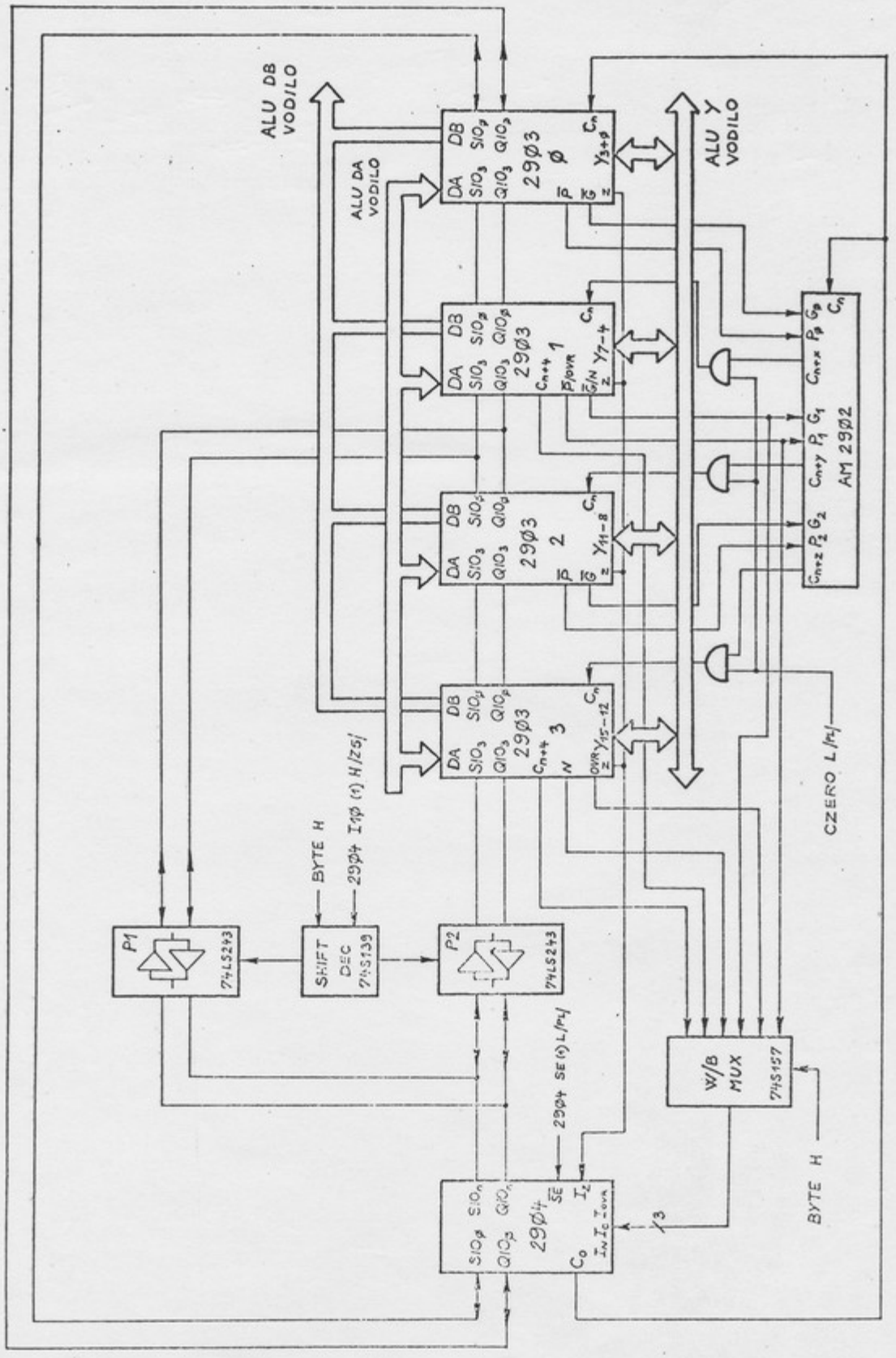
ALU (slika 4.2) je razdeljen v stiri 4-bitne rezine (A1, A2, A3 in A4) , kjer je vsaka rezina sestavljena iz 4-bitnega elementa (AM2903). Za izvedbo ALU funkcij je dodan element Look-Ahead Carry Generator (AM2902 na listu A9).



Slika 4.1


SD VODILO
D/R
SWAP MUX A1 → A4
DS A1 → A4
XUNANJI PODATKOVNI VHOD
IR PL CISLOM
DAS A1 → A4
BCD MUX A6
PSW A5
29Ø4 A6
29Ø3 A1 → A4
DB
DA
Y
3-ST VMESNIK
IR
DEKODIRNA LOGIKA
PDR
PAR
PLS B1
29Ø
VBA A1Ø
BA A1Ø
D/R

Slika 4.1 a


ALU DB VODILO
ALU DA VODILO
ALU Y VODILO
29Ø3 Ø
29Ø3 1
29Ø3 2
29Ø3 3
AM 29Ø2
P1
P2
74LS243
SHIFT DEC 74S139
BYTE H
29Ø4 IIØ (1) H/Z5/
CZERO L/H/
W/B MUX 74S157
29Ø4 SE(1) L/H/
29Ø4
3

Slika 4.2

4.2.2 ALU vhodi in izhodi

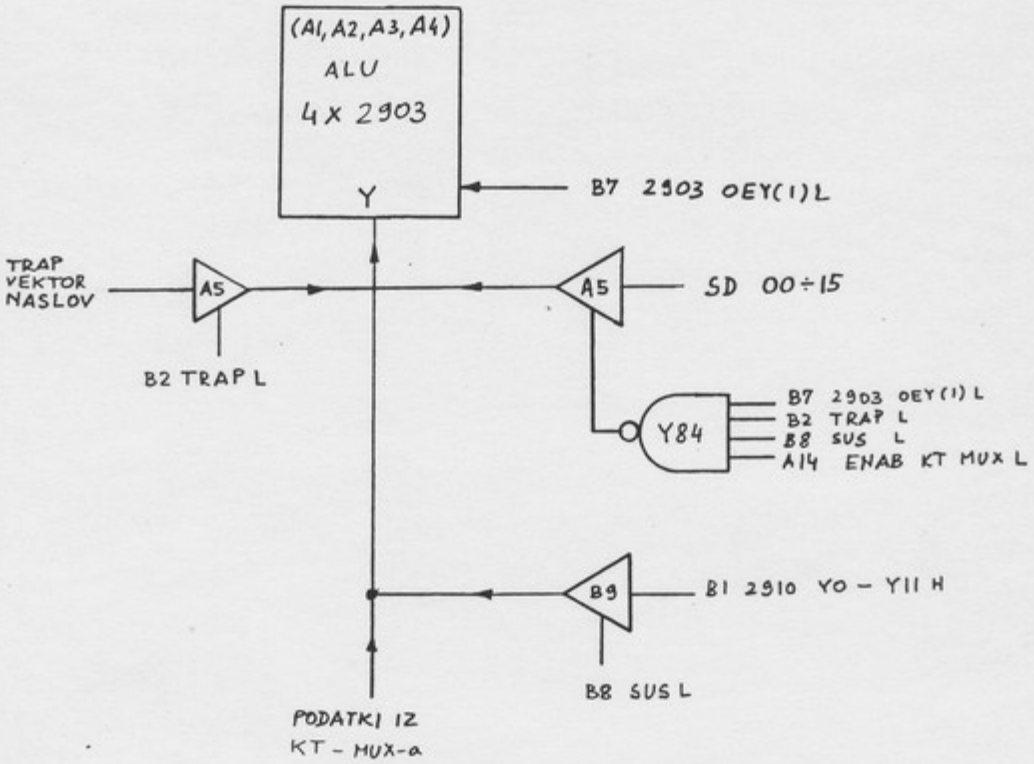
DA vhod k vsakemu ALU elementu prihaja iz DAS MUX-a (elementi so 74S253 na listih A1, A2 ter 74S257 na listih A3 in A4). DAS MUX je krmiljen mikroprogramsko s signaloma B7 DAS 1 H in B7 DAS 0 H (tabela 4.1).

Tabela 4.1 Izvorni podatki na DAS MUX-u

DAS1	DAS0	Funkcija	Opis
L	L	IMMD 00-15	Točno določen podatek za določeno mikroinstrukcijo gre na izhod DAS MUX-a
L	H	IR 00-07	8-bitni podatek iz IR gre na izhod DAS MUX-a (odmik)
H	L	SD 00-15	16-bitni podatek, ki je izbran na DS MUX-u gre na izhod DAS MUX-a
H	H	ROM 00-07	8-bitni podatek iz CIS ROM-a gre na izhod DAS MUX-a

V primeru da je ALU-ju dodan zunanji RAM (CIS instrukcije), pridejo na DA vhod še izhodi teh registrov (48 registrov). Izbiro med zunanjimi in notranjim RAM-om omogočata zgornji dve naslovni liniji A8 ADR A4 H in A8 ADR A5 H (uporabljen je dekodirni element 74S139 na poziciji E105 list A9). Pri standardnem naboru instrukcij sta ti dve naslovni liniji vedno na logični '0', ker naslavljamo vedno notranji ALU RAM.
Signal B7 2903 EA(1) L na logični '0' omogoča, da gre signal A9 ADEK I0 L v logično '0'. Takrat je kot ALU R izvorni podatek izbran naslovljeni register RAM-a.
DB vhod je uporabljen samo v primeru, ko imamo dodan zunanji RAM. Izbiro med podatki iz zunanjega ali notranjega ALU RAM-a omogočata zgornji dve naslovni liniji A7 ADR B4 H in A7 ADR B5 H (to funkcijo opravlja dekodirni element 74S139 na poziciji E108 list A9). V primeru standardnega nabora instrukcij sta ti dve naslovni liniji vedno na ničli, zato je signal A9 BDEK I0 L na logični '0', kar pomeni, da DB linije niso uporabljene kot podatkovni vhod.
DB izhod je uporabljen za posredovanje naslovov enoti za naslavljanje pomnilnika (VBA ali BA registru) ter za shranjevanje podatkov preko DS MUX-a v glavni pomnilnik. Preko DB izhoda je lahko mikroprogramskemu sekvenčniku posredovan števec iteracij (instrukciji ASH in ASHC).
Pri CIS instrukcijskemu naboru je DB izhod uporabljen za posredovanje 8-bitnega podatka CIS ROM-u in 3-bitnega podatka (TIP STRINGA) mikroprogramskemu sekvenčniku.
Na Y vhod ALU-ja prihajajo podatki iz naslednjih izvorov (slika 4.3):

- podatki, ki so izbrani z DS MUX-om (SD 00-15)
- TRAP VEKTOR naslov
- podatki iz KT MUX-a
- podatki iz izhoda enote AM2910

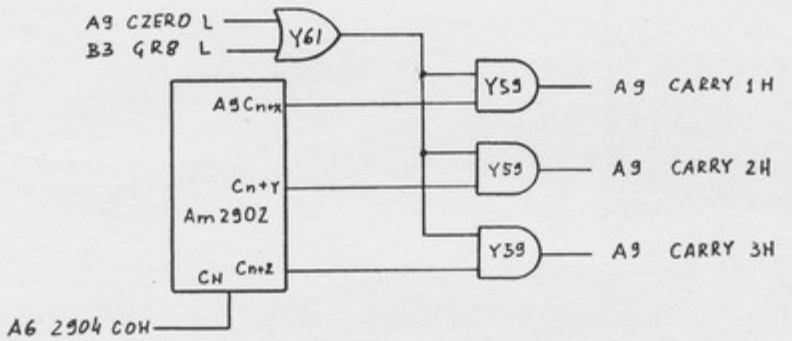


Slika 4.3

Podatki, ki so izbrani z DS MUX-om prihajajo na Y vhod ALU-ja v primeru, ko gre za dostavo naslovov in podatkov iz pomnilnika, ali pa takrat ko v mikroinstrukciji SERVICE potegnemo prekinitveni vektor naslov, če je CPU prekinjen z neko zunanjo napravo. TRAP VEKTOR naslov se zapiše v ALU register v mikroinstrukciji SERVICE takrat, ko je signal B2 TRAP L na logični "0".
Podatki iz KT MUX-a se vodijo na Y vhod v primeru, ko delamo funkcijo branja enote za delo s pomnilnikom preko konzole, ali pa če želimo delati določeno operacijo nad temi registri in podatek potegnemo v ALU.
Podatke iz izhoda enote AM2910 začasno shranjujemo v ALU registre pri dodatnem CIS naboru instrukcij, če pride do prekinitve med izvajanjem te instrukcije.
Signal B7 2903 OEY(1) L na logični "1" pove, da je Y vodilo ALU-ja uporabljeno kot vhod.
Preko Y izhoda ALU-ja se podatki shranjujejo v RAM, ali pa jih pošiljamo na Unibus v primeru, ko gre za shranjevanje podatkov v pomnilnik.
Signal B7 2903 OEY(1) L na logični "0" pove, da so Y podatkovne linije uporabljene kot izhod.

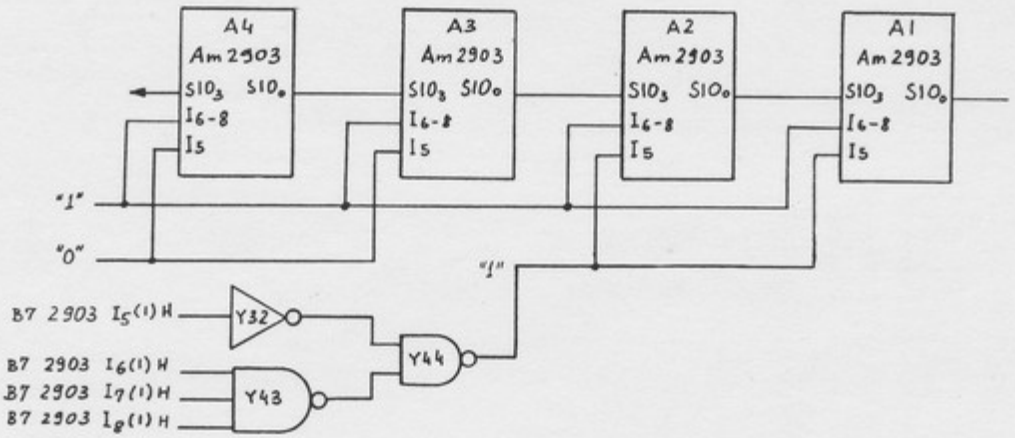
4.2.3 ALU funkcije

Rezine AM2903 imajo poleg običajnih aritmetičnih in logičnih funkcij še poseben nabor specialnih funkcij, ki omogočajo operacije deljenja, množenja in povečevanja vsebine registra za ena ali dva.
Funkcije, ki se izvršijo na ALU-ju so določene z instrukcijskimi vhodi I0 do I8 rezine AM2903 in carry-in (CN) bitom, ki ga določamo preko enote AM2904. Določitev CN bita je detaljno orisana v poglavju 4.3.
Pri izvajanju običajnih funkcij je operacija nad podatki določena z instrukcijskimi vhodi I0 do I4 [signali B7 2903 I0(1) H do B7 2903 I4(1) H], instrukcijski vhodi I5 do I8 določajo funkcije pomikanja in shranjevanja podatkov.
Pri specialnih funkcijah so instrukcijski vhodi I0 do I4 na logični "0". Operacijo, kontrolo pomikov in shranjevanje podatkov opravljajo instrukcijski vhodi I5 do I8 [signali B7 2903 I5(1) H do B7 2903 I8(1) H].
Pri izvajanju ALU funkcij sodeluje tudi enota AM 2902 (High-Speed Look-Ahead Carry Generator), ki skrbi za določitev carry bitov med posameznimi rezinami AM2903. Enota AM2902 in najmanj pomenbna rezina ALU-ja dobita začetni CN bit, ki jima ga določi enota AM2904.
Med izhodi enote AM2902 in Cn vhodi zgornjih treh rezin ALU-ja je dodana logika, ki pri CIS instrukcijskem naboru omogoči s signaloma A9 CZERO L in B3 GR8 L vsilitev logične ničle na Cn vhode (slika 4.4).



Slika 4.4

Na instrukcijski vhod I5 rezin AM2903 (list A1 in A2) je dodana logika, ki omogoča raztegnitev bita 7 v podatkovni besedi na zgornji del besede (slika 4.5)



Slika 4.5

Spodnji dve rezini (A1, A2) imata na instrukcijskih linijah I5 do I8 kontrolo za vpis z Y vodila v RAM, zgornji dve rezini (A3, A4) pa za vpis tistega podatka v RAM, ki se nahaja na SIO3 izhodu rezine AM2903 z lista A2.

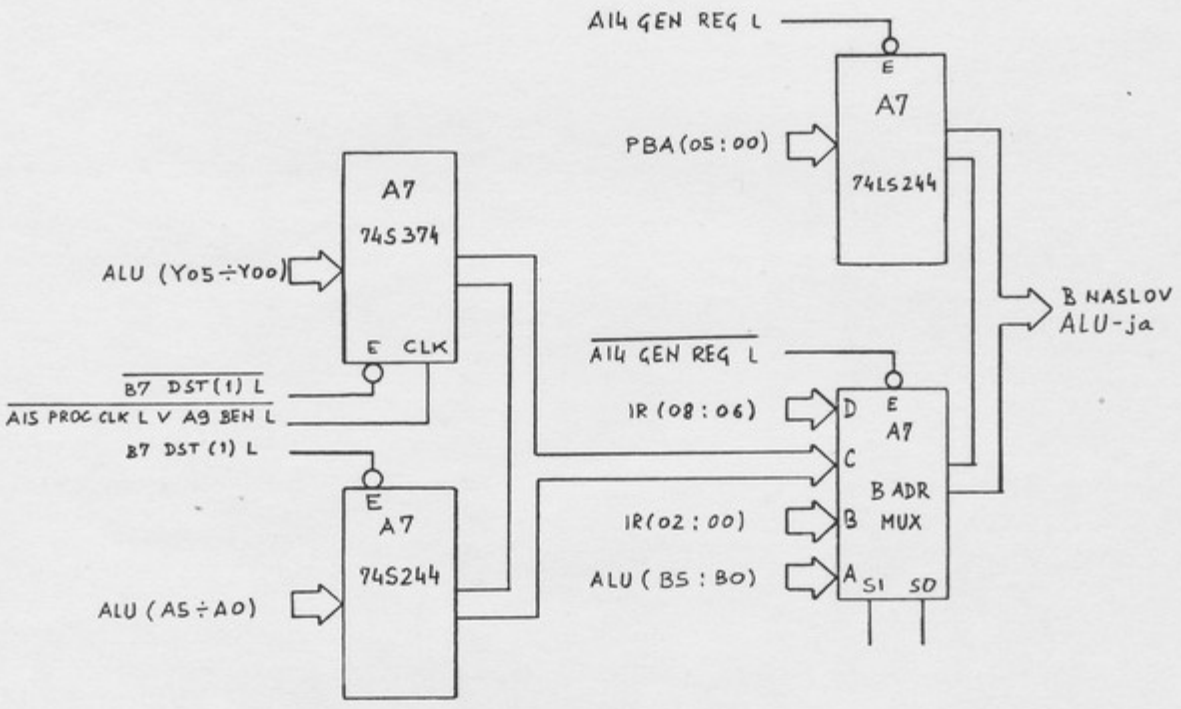
4.2.4 Naslavljanje ALU RAM-a

ALU vsebuje 16 besedni RAM z zapahoma na obeh izhodih. RAM je naslovljen z A strani in z B strani.
ALU A naslov:
Naslavljanje A dela RAM-a je izvedeno z MUX-i (74S153) na plošči A (list A8). V odvisnosti od stanja krmilnih linij B6 AS1(1) H in B6 AS0(1) H lahko izbiramo naslov za izvorni podatek med naslednjimi možnostmi (glej tabelo 4.2)

Tabela 4.2

AS1	AS0	Izvor
0	0	Mikroprogramski pomnilnik (ALU A5 : ALU A0)
0	1	Ponorno polje v IR registru (IR02 : IR00)
1	0	Neuporabljeno
1	1	Izvorno polje v IR registru (IR08 : IR06)

Pri uporabi kazalca sklada je potrebna še dodatna logika, ki omogoči spremenbo naslova R6 v R14 takrat, ko procesor dela v uporabniškem načinu. Za to pretvorbo poskrbi signal A12 MODE 01 H, ki pove v katerem načinu dela procesor.
ALU B naslov:
Naslavljanje B dela RAM-a je izvedeno z izbiralnimi elementi 74S253 (MUX) na listu A7. Na isti vhod ALU-ja je pripeljan tudi pomnilniški naslov z glavnega naslovnega vodila (PBA 05-00). Ta naslov je uporabljen pri delu s konzolo, kajti takrat poteka naslavljanje registrov R0-R15 preko visokih pomnilniških naslovov, ki so rezervirani v ta namen. Naslavljanje B dela ALU-ja prikazuje slika 4.6.



Slika 4.6

Pri delu s konzolo poteka naslavljanje registrov preko naslovov PBA(05:00). Takrat je signal A14 GEN REG L na logični '0' in je onemogočeno naslavljanje preko B ADR MUX-a.
Pri naslavljanju z B ADR MUX-om pa lahko izbiramo med naslednjimi možnostmi:

- ponorno polje v IR (IR08 : 06)
- naslavljanje preko vsebine v registru ALU-ja(ALU Y05:Y00)
- uporaba A naslovnega polja iz mikroprogramskega pomnilnika (ALU A5 : A0)
- izvorno polje v IR (IR 02: 00)
- B naslovno polje v mikroprogramskem pomnilniku (ALU B5 : B0)

Naslavljanje RAM-a je realizirano na dva načina:

- dvo-naslovni način (en izvorni in en ponorni naslov)
- tro-naslovni način (dva izvorna in en ponorni naslov)

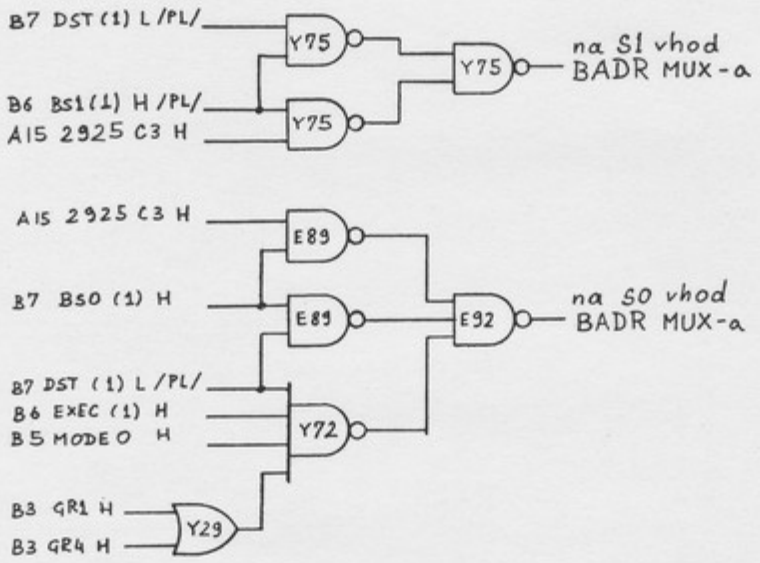
Dvo-naslovni način:
Pri tem načinu naslavljanja imamo en izvorni (A naslov) in en ponorni (B naslov) naslov, ki sta prisotna celo mikroinstrukcijo na naslovnih vhodih ALU-ja. Signal B7 DST(1) L je pri tem načinu naslavljanja na logični '1', kar pomeni, da v tem primeru ni možna uporaba naslovnega polja ALU (A5 : A0), kajti 3-stanjski element 74S244 (E83) ni zakrmiljen. Logika, ki določa izbiro med posameznimi vhodi je prikazana na sliki 4.7.
V primeru, ko gre za izvajanje instrukcij [B6 EXEC(1) H je na logični '1'] iz GRUPE 1 ali GRUPE 4 (B3 GR1 H ali B3 GR4 H na logični '1') in imajo uporabljen naslovni način 0 (B5 MODE0 H na logični '1'), potem je namesto naslova, ki je določen v mikroinstrukciji na B ADR MUX vsiljen ponorni naslov, ki je določen v instrukcijskem registru (IR 02 : 00).
Za uporabo ostalih naslovnih izvorov pa je uporabljeno krmiljenje iz mikroprogramskega pomnilnika.
ALU (Y05 : Y00) naslovni vhodi niso uporabljeni pri standardnem naboru instrukcij, ampak so dodani zaradi dodatnega nabora CIS instrukcij. Podatek, ki ga dobimo na Y izhodu ALU-ja v isti mikronstrukciji vpišemo v register (element 74S374, pozicija Y55,list A7). Ta podatek je v nadaljevanju lahko uporabljen kot naslov.
Tro-naslovni način:
Razlikuje se od dvo-naslovnega načina po tem, da je možna uporaba dveh izvornih in enega ponornega naslova. To funkcijo omogoča signal B7 DST(1) L, ki je v tem načinu naslavljanja na logični '0'. V tem primeru je možna tudi uporaba naslovnega polja ALU A5 : A0, pri tem pa onemogočena uporaba naslovnega polja ALU Y05 : Y00. Pri tronaslovnem načinu veljajo naslednje možnosti uporabe za izvorne in ponorne naslove (glej tabelo 4.3).

Tabela 4.3 Tro-naslovni način

A izv. naslov	B izv. naslov	B ponorni naslov
X	IR(08 : 06)	ALU(B5 : B0)
X	ALU(A5 : A0)	ALU(B5 : B0)
X	IR(02 : 00)	ALU(B5 : B0)

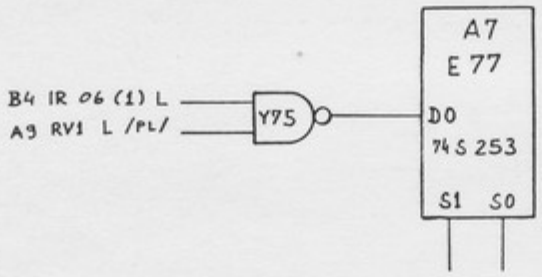
X - izbran je katerikoli od izvornih naslovov, ki so pripeljani na A naslovne linije ALU-ja

Pri tem naslovnem načinu je omogočena na B naslovu uporaba enega od treh izvornih naslovov [IR(08 : 06), ALU(A5 : A0), IR(02 : 00)] in samo ena možnost ponornega naslova [ALU(B5 : B0)]. Tro-naslovni način omogoča logika na sliki 4.7.



Slika 4.7

Spremenba naslova na B delu ALU-ja je realizirana tako, da je prvo polovico mikroinstrukcije prisoten izvorni naslov. Po prehodu signala A15 2925 C3 H iz logične "1" v logično "0 se na selektne vhode B ADR MUX-a vsilijo logične "0".
Pri instrukcijah iz GRUPE 2 (MUL, DIV, ASH, ASHC) je možno naslavljanje registrov R in RV1, kar pomeni, da je potrebno na IR06 vhodno linijo naslova dodati logiko, ki v tistih primerih, ko iz pipeline registra pride signal A9 RV1 L na logični "0" vsili logično "1" na vhod BADR MUX-a. Če ni uporabljen signal A9 RV1 L pa pride na vhod BADR MUX-a direktni IR 06(1) L signal (slika 4.8).



Slika 4.8

4.2.5 Organizacija ALU RAM-a

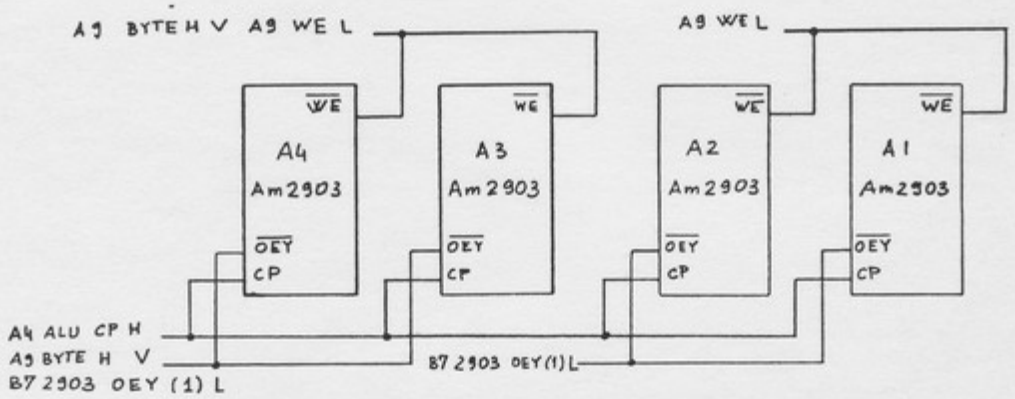
RAM je sestavni del rezine AM2903. Zagotavlja uporabo šestnajst 16-bitnih registrov v okviru celotnega ALU-ja (A1, A2, A3, A4). Njihovo uporabo pa si oglejmo v tabeli 4.4.

Tabela 4.4 Uporaba RAM registrov

Stevilka registra	Opis
R0	
R1	
R2	
R3	REGISTRI ZA ŠPLOSNO UPORABO
R4	
R5	
R6	procesorjev kazalec sklada (Kernel nacin)
R7	programski števec
R10	konstanta 0
R11	konstanta 1
R12	delovni
R13	neuporabljen
R14	delovni
R15	delovni
R16	procesorjev kazalec sklada (User nacin)
R17	delovni

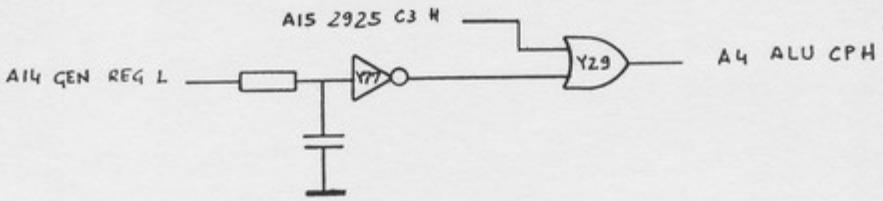
4.2.6 Branje in vpisovanje v RAM

Branje iz RAM-a rezine AM2903 se začne takoj, ko dobi veljaven naslov. Podatki iz RAM-a gredo preko zapahov, ki se zaprejo takoj, ko gre signal A4 ALU CP H v logično '0'(L). Te podatke imamo prisotne na DB izhodu ALU-ja in v ALU-ju za nadaljno operacijo nad podatki. Pri branju glavnih programskih registrov s konzole je vsebina registra, ki je bil naslovljen reko PBA naslovov prisotna na DB izhodu ALU-ja, kajti signal A14 GEN REG L na logični '0' vsili na CP signal ALU-ja logično '1', ki omogoča branje.
Vpisovanje v RAM:
Vpis v RAM ALU-ja se izvrši ob prehodu signala A4 ALU CP H v logično '1' (⎍), če je hkrati na /WE vhodu logična '0'. Omogočeno je vpisovanje besednega ali zlogovnega podatka. Vpis zlogovnega podatka se izvrši, če je signal A9 BYTE H na logični '1' (slika 4.9)



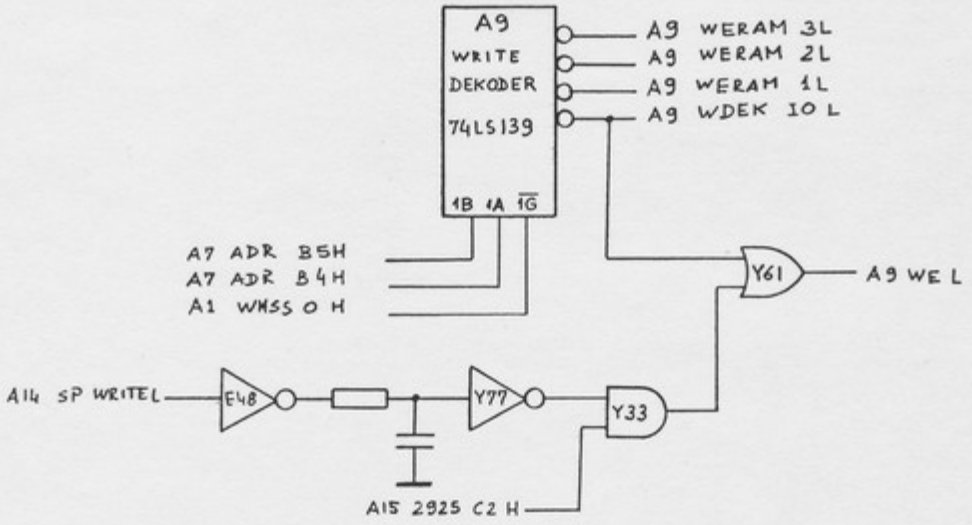
Slika 4.9

V RAM se lahko vpisujejo podatki, ki so rezultat operacije v ALU-ju [signal B7 2903 OEY(1) L na logični '0'] ali podatki, ki so prisotni na Y vodilu v primeru, ko je signal B7 2903 OEY(1) L na logični '1'.
Signal ALU CP H je v delovanju procesorja (signal A14 GEN REG L je na logični '1') enak signalu A15 2925 C3 H (slika 4.10).



Slika 4.10

Pri delu s konzolo je signal A15 2925 C3 H na logični "0" in v primeru, da gre za vpis v glavne programske registre je signal A14 GEN REG L na logični "1", zato gre signal A4 ALU CP H v logično "1"( ). Ura na ALU-ju je s tem zagotovila vpis v RAM, vendar pa je za vpis potreben še signal A9 WE L na logični "0".(slika 4.11)



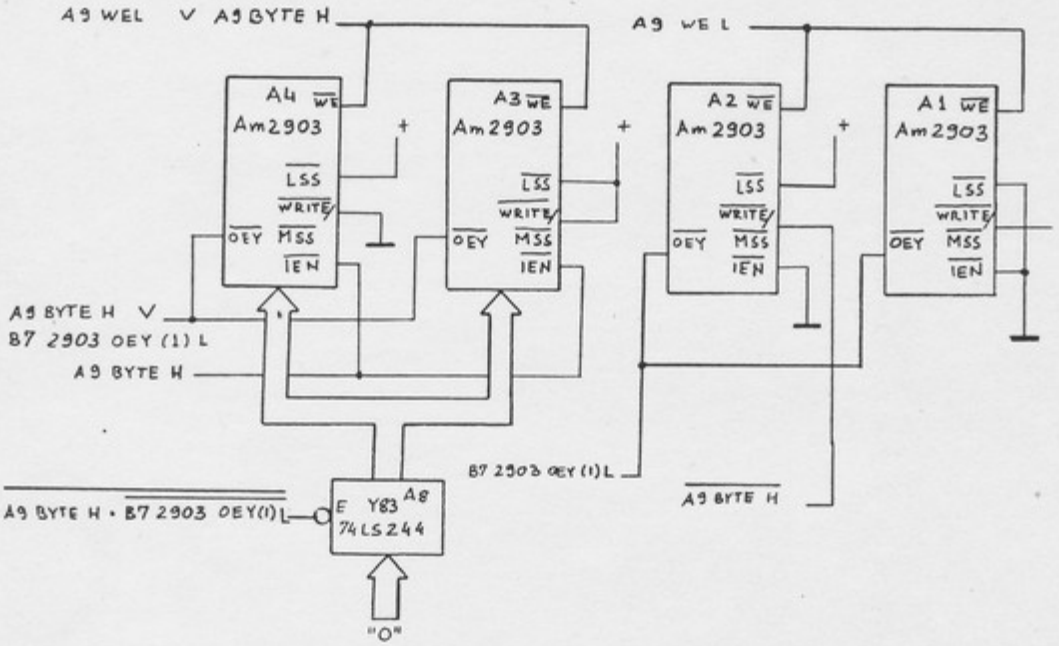
Slika 4.11

Pri delovanju procesorja je signal A14 SP WRITE L vedno na logični "1". V primeru, da želimo vpis v RAM ALU-ja so signali A7 ADR B5 H, A7 ADR B4 H in A1 WMSS O H na logični "0" in je aktiven signal A9 WDEK IO L na logični "0", kar omogoča, da je sig-

nal A9 WE L odvisen samo od signala A15 2925 C2 H. V tem primeru je možno vpisovati podatke z Y vodila v RAM ALU-ja.
Pri delu s konzolo je signal A15 2925 C2 H na logični "1" in na vpis v RAM vpliva signal A14 SP WRITE L. V mikroinstrukciji SERVICE, kjer procesor stoji je signal A1 ALU WMSS 0 H na logični "0", zato je tudi signal A9 WDEK I0 L na logični "0". Signal A14 SP WRITE L na logični "0" torej vsili signal A9 WE L na logično "0".
WRITE DECODER (slika 4.10) služi za vpisovanje v dodatni nabor RAM-ov pri izvajanju CIS instrukcij. Dodani naj bi bili trije RAM-i po 16 registrov. V kateri RAM se podatki vpisujejo, določata naslovni liniji A7 ADR B5 H in A7 ADR B4 H.

4.2.7 Zlogovne funkcije

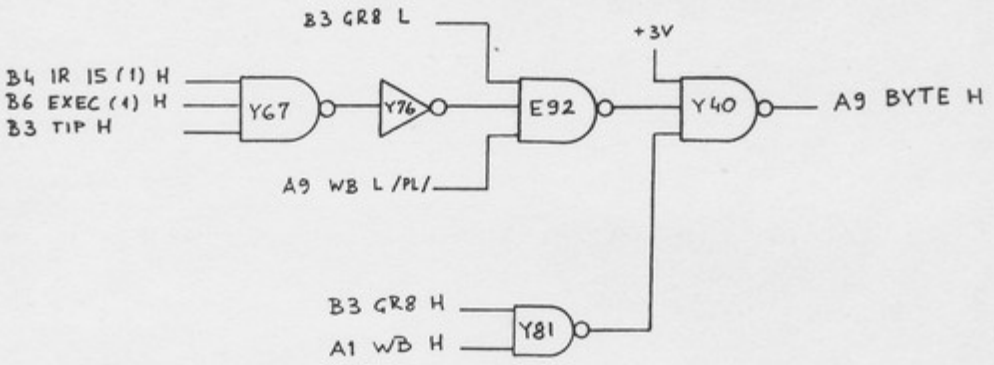
Zgradba ALU-ja, v smislu sestavljanja rezin v 16-bitno dolžino besede, omogoča delo nad besedami ali nad zlogi (slika 4.12).



Slika 4.12

V primeru, da gre za delo nad besednimi podatki, je signal A9 BYTE H na logični "0" in so aktivne vse štiri rezine (A1 do A4). Za delo nad zlogi je potrebno najprej definirati logiko, ki omogoča krmiljenje samo spodnjih dveh rezin (A1, A2). Operacije nad zlogi se v ALU-ju izvršijo v primeru, če se v IR nahaja zlo-

sovna instrukcija [signali B4 IR 15(1) H, B6 EXEC(1) H in B3 TIP H na logicni "1" in če nista aktivna signala B3 GR8 H in A1 WB H, je signal A9 BYTE H na logični "1"].
V primeru, če je v IR zlogovna instrukcija pri standardnem naboru instrukcij, lahko s signalom A9 WB L na logični "0", vsilimo besedno operacijo nad podatki (slika 4.13).



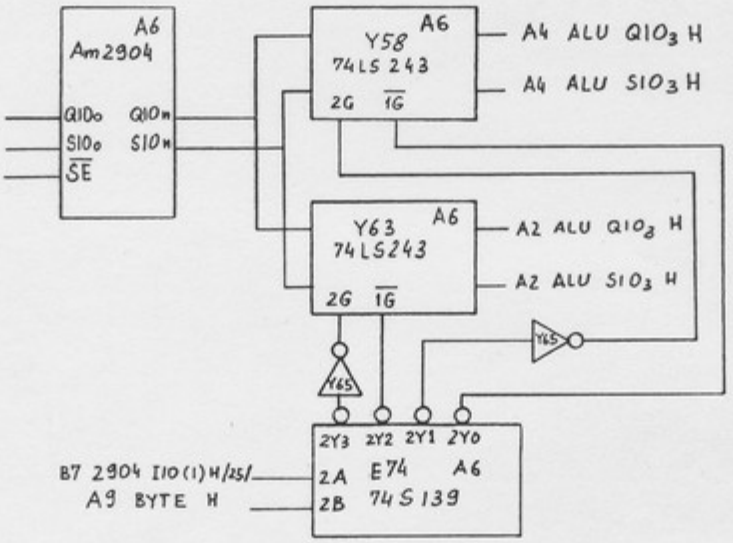
Slika 4.13

V primeru, da gre za CIS instrukcijski nabor (signal B3 GR8 H je na logični "1") lahko s signalom A1 WB H na logični "1" vsilimo zlogovno operacijo nad podatki (slika 4.13).
Če dobimo signal A9 BYTE H na logični "1" pomeni, da gre za delo v spodnjih dveh rezinah (A1, A2). Rezina A2 postane tedaj najbolj pomenbna in njeni izhodi N, V in C postanejo veljavne pogojne kode, ki se vodijo na WB MUX. Vpis rezultata se izvrši samo v spodnji zlog. Za določitev pogojne kode Z pa sledi vsilitev ničel na zgornjih 8 linij (element 74LS244, pozicija Y83, list A8), kajti detekcija rezultata glede na nič poteka nad celo besedo.

4.2.8 Pomikanje podatkov ALU-ja

V ALU-ju je omogočena funkcija pomikanja vsebine levo in desno nad besedo ali nad zlogom. Za izvedbo funkcije pomikanja je ALU-ju dodana še enota AM2904, ki skrbi za krmiljenje pomikov (slika 4.2). Za izbiro pomikov nad besedo ali nad zlogom skrbita dvosmerna prenosnika 74LS243, ki ju krmili dekodirni element 74S139, list A6.
Enota AM2904, rezine AM2903 in dvosmerna prenosnika, ki sta krmiljena z dekodirnim elementom, omogočajo logične in aritmetične pomike levo ali desno nad zlogom, besedo ali dvojno dolžino besede.
Enota AM2904 ima za krmiljenje pomikov uporabljene instrukcijske vhode I6 do I10. Na instrukcijski vhod I10 prihaja signal B7

2904 I10(1) H/W3/, ki definira smer pomikov (signal na logični "0" pomeni pomikanje desno, sicer gre za pomikanje levo). Ostali instrukcijski vhodi pa določajo ali gre za pomikanje nad besedo ali nad dvojno dolžino besede oz. določajo izvor ali ponor podatka, ki smo ga poslali v ALU, ali ga dobimo iz ALU-ja. Možni izvori podatka so: 0, 1, C statusni bit, N statusni bit, In, Iovr vhodi, Ic vhod, ali pa poskrbi za rotate funkcijo. Možni ponor pomaknjenega bita iz ALU-ja je C statusni bit.
Pri vseh pomikih mora biti signal B7 2904 SE(1) L na logični "0".
V ALU-ju skrbijo za krmiljenje pomikov instrukcijski vhodi I8 do I5, ki določajo aritmetične ali logične pomike oz. pomikanje levo ali desno.
Logika, ki skrbi za izbiro pomikanja levo ali desno oz. za pomikanje nad zlogom ali nad besedo je prikazana na sliki 4.14.

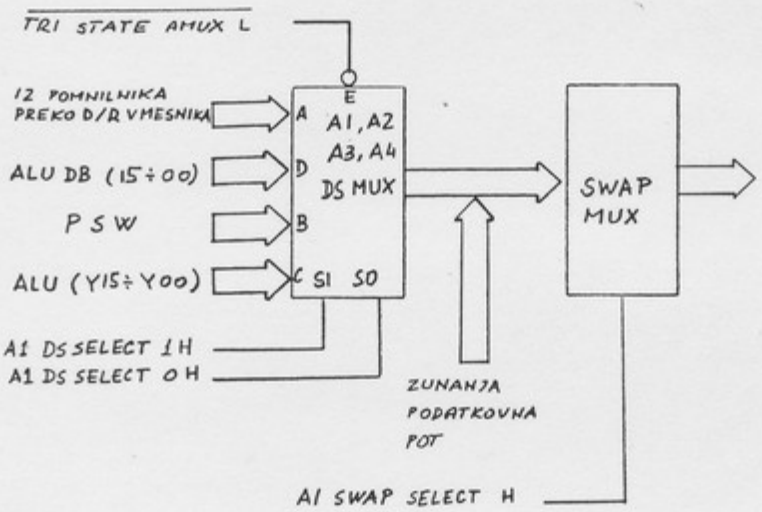


Slika 4.14

Pri pomikanju besednega podatka, je signal A9 BYTE H na logični "0" (izbran je dvosmerni prenosnik Y58). Signal B7 2904 I10(1) H/Z5/ na logični "0" pomeni pomikanje desno in dekoder E74 daje logično "0" na 2Y0 izhod, kar pomeni, da je na dvosmernem prenosniku Y58 omogočeno pomikanje desno. Signal B7 2904 I10 H /Z5/ na logični "1" pa omogoči pomikanje levo.
Pri pomikanju zlogovnega podatka je signal A9 BYTE H na logični "1". Izbran je dvosmerni prenosnik Y63. Pomikanje levo ali desno je določeno enako kot zgoraj s signalom B7 2904 I10(1) H /Z5/.

4.2.9 DS MUX in SWAP MUX

DS MUX (slika 4.15) je sestavljen iz osmih 4 na 1 3-stanjskih izbiralnikov (74S253). Podatki iz DS MUX-a gredo na vhode SWAP MUX-a, ki skrbi za zamenjavo podatkov v primeru zlogovnih instrukcij (list A1, A2, A3, A4).



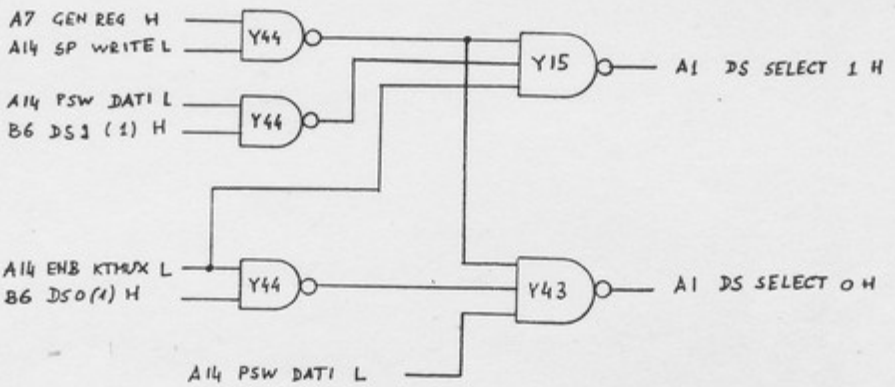
Slika 4.15

S signalom TRI STATE AMUX L na logični "0" se zakrmili DS MUX v visokoimpedančno stanje in je možno v procesor sprejeti 16-bitni podatek iz zunanjih priključenih modulov (procesor s plavajočo piko, CACHE pomnilnik).
Pri delovanju procesorja sta signala A1 DS SELECT 1 H in A1 DS SELECT 0 H zakrmiljena na več nacinov (slika 4.16):

- tako kot je zapisano v mikroprogramskem pomilniku (aktivna signala sta B6 DS 1(1) H in DS 0(1) H).
- pri naslavljanju registrov enote za naslavljanje pomnilnika(preko visokih pomnilniških naslovov) je signal A14 ENAB KT MUX L na logični "0" in je zakrmiljen ALU (Y15 - Y00) vhod DS MUX-a.
- pri naslavljanju PSW registra za branje (signal A14 PSW DATI L je na logicni "0") je zakrmiljen PSW vhod na DS MUX-u namesto, da bi podatek prišel iz pomnilnika preko D/R vmesnika.

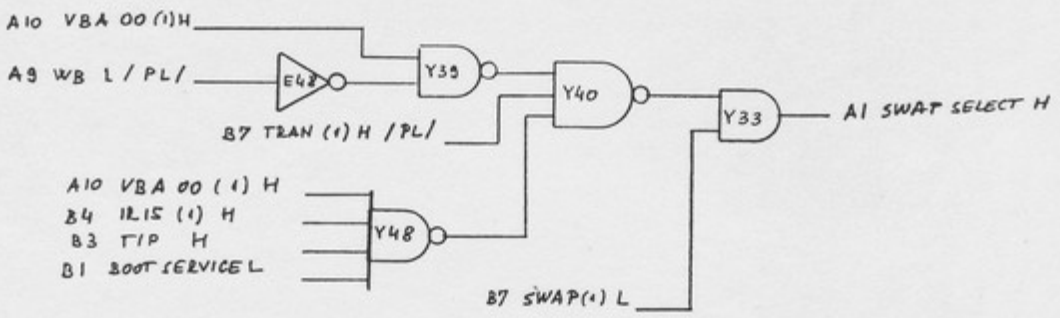
Delo s konzolo zahteva uporabo različnih podatkovnih izvorov na DS MUX-u. Iz mikroprogramskega pomnilnika je DS MUX zakrmiljen tako, da daje na izhod podatke iz pomnilnika preko D/R vmesnika, kajti procesorjeva ura stoji, v pipeline registru pa je zapisana mikroinstrukcija SERVICE. Različne možnosti krmiljenja DS MUX-a (slika 4.16) so:

- branje glavnih programskih registrov : (signal A7 GEN REG H je aktiven na logični "1" kar omogoči, da je na DS MUX-u izbran vhod ALU DB (15 - 00)).
  Vpisovanje v glavne programske registre (signal A14 SP WRITE L je na logični "0" in signal A7 GEN REG H na logični "1", kar omogoči, da je na DS MUX-u ostal izbran vhod podatkov iz pomnilnika preko D/R vmesnika)
- branje PSW registra : (signal A14 PSW DATI L na logični "0" omogoči, da je izbran PSW vhod na DS MUX-u)
  Vpisovanje v PSW register poteka preko glavnega podatkovnega vodila.
- branje registrov enote za naslavljanje pomnilnika (signal A14 ENAB KT MUX L je na logični "0", kar omogoči izbiro vhoda ALU (Y15 - Y00)).
  Vpisovanje v te registre poteka preko glavnega podatkovnega vodila.



Slika 4.16

SWAP MUX skrbi za zamenjavo zlogov pri dostavljanju ali shranjevanju podatkov v pomnilnik. Zamenjava zlogov se omogoči, če je signal B7 SWAP (1) L/PL/ na logični "0". Ta signal se nahaja v pipeline registru, torej je prepoved zamenjave pod mikroprogramsko kontrolo.
Zamenjava se zgodi ob dostavi podatkov ali shranjevanju podatka, če se izvaja zlogovna instrukcija [signali A10 VBA 00(1) H, B4 IR 15(1) H, B3 TIP H in B1 BUT SERVICE (1) H so na logični "0", potem pride do zamenjave zlogov] in je signal B7 SWAP (1) L na logični "1". Za CIS instrukcijski nabor je omogočena zamenjava zlogov, če je signal A9 WB L na logični "0" in A10 VBA 00(1) H na logični "1". Glej sliko 4.17.



Slika 4.17

V primeru dostave ali shranjevanja podatkov [B7 TRAN(1) H /PL/ je na logični "1"] pri besednih instrukcijah (B3 TIP H=0) ne sme biti aktivna SWAP logika.
V primeru dostave naslovov pri besednih ali zlogovnih instrukcijah nam mikroprogram [signal B7 SWAP (1)L /PL/=0] onemogoči zamenjavo zlogov. Pri SWAB instrukciji je signal B7 SWAP (1) L /PL/ na logični "1", signal B7 TRAN (1) H /PL/ na logicni "0",ker ni prenosa in takrat je na SWAP MUX-u omogočena zamenjava zlogov.

4.2.10 Statusna beseda procesorja (PSW)

Statusna beseda procesorja vsebuje naslednje informacije:

- o trenutnem in prejšnjem memory management načinu
- trenutno procesorjevo prioriteto
- pogojne kode sledе na prejšnjo operacijo
- procesorjevo past za testne namene (T-bit)
- o suspenziji CIS ukaza

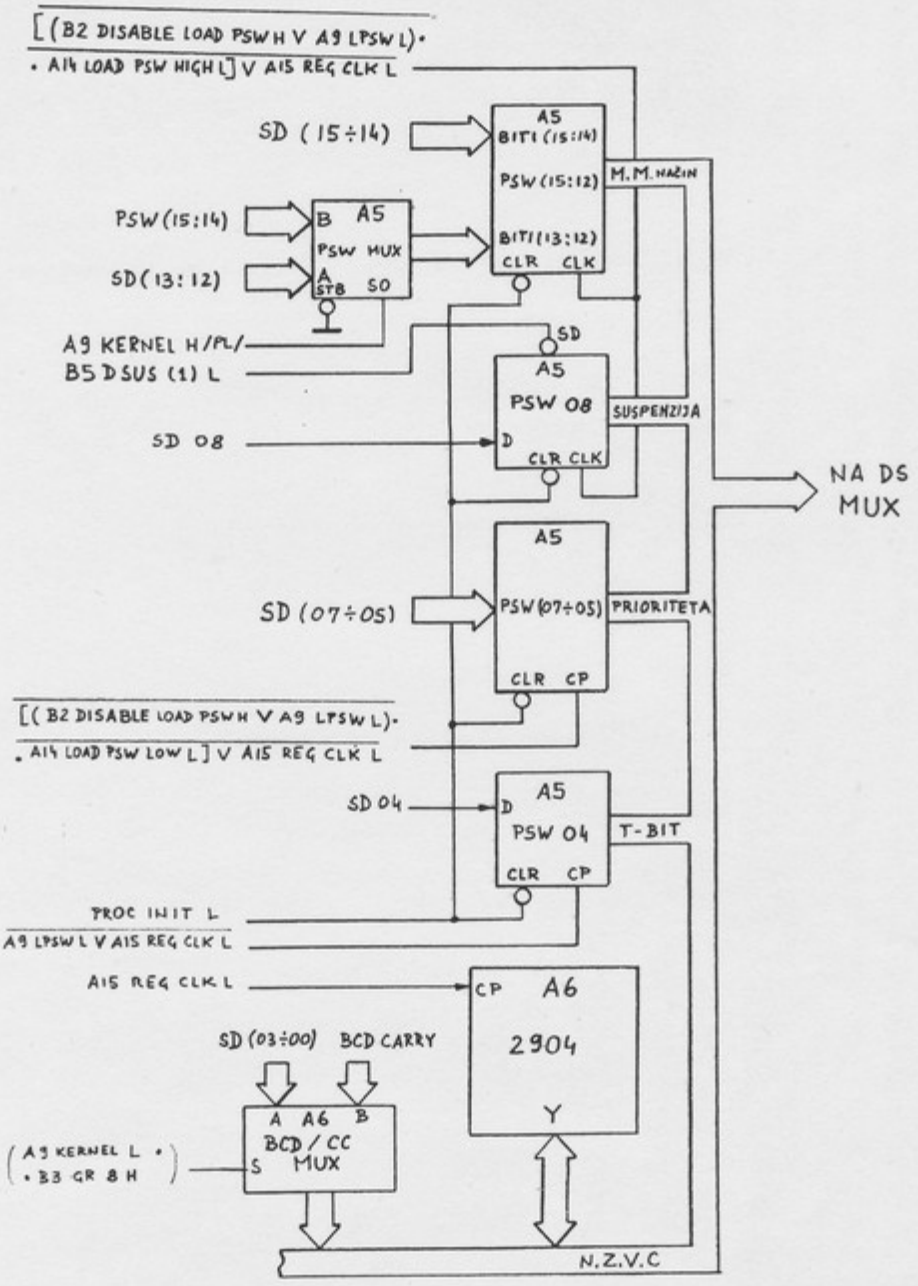
Tabela 4.5 prikazuje razporeditev in pomen posameznih bitov v PSW-ju.
PSW (slika 4.18) je 13 bitni register sestavljen iz dveh elementov (74LS175 - 4-bitne D celice), dveh ločenih D celic (74LS74) in makrostatusnega registra v enoti 2904.
Prva skupina štirih D celic (E59 na listu A5) je uporabljena za shranjevanje trenutnega in prejšnjega memory management načina. Bita 15 in 14 iz SD vodila predstavljata vhode za signale A5 PSW 15(1) H in A5 PSW 14(1) H, ki jih vodimo nazaj na izbiralnik (E53 na A5). Ta v odvisnosti od signala A9 KERNEL H /PL/ izbira PSW bita 15 in 14 ali pa signala A5 SD 13 H, A5 SD 12 H in jih daje na PSW vhode 13 in 12.
Od druge skupine štirih D celic (E62 na A5) so tri uporabljene za trenutno procesorjevo prioriteto ; na vhod teh celic so vezani biti 07 - 05 iz SD vodila. Sledita D celici 74LS74 (Y62 na A5), od katerih je ena uporabljena za procesorjevo past (T-bit,

ki ga vpisujemo s signalom A2 SD 04 H), druga pa služi za vpis informacije o suspenziji CIS ukaza. V PSW 08 celico je možen vpis signala A3 SD 08 H v primeru, ko gre za vpisovanje nove vsebine celega PSW-ja iz SD vodila. Asinhrono pa je možno vpisati enico v PSW 08 bit s signalom B5 DSUS(1) L v primeru , da je nastopila suspenzija CIS instrukcije.

Tabela 4.5 Statusna beseda procesorja

PSW bit	Ime	Uporaba
15 14	Trenutni memory management način	Vsebuje informacijo o trenutnem MM načinu
13 12	Prejšnji MM način	Vsebuje informacijo o prejšnjem MM načinu
11-09	Neuporabljeno	
08	Suspenzija	Govori o suspenziji CIS ukaza na mikro nivoju
07-05	Prioriteta	Za postavljanje prioritete procesorja
04	Trace	V primeru, da je ta bit postavljen, se izvede past slede na trace vektor
03	N	Signalizira negativen rezultat
02	Z	signalizira rezultat, ki je enak nič
01	V	Signalizira prekoračitev rezultata
00	C	Se postavi v primeru, da je operacija generirala prenos na najbolj pomenbnem bitu

Vse celice v PSW so vezane na urin impulz A15 REG CLK L. Vpis v PSW register je omogočen z mikroprogramom, ki aktivira signal A9 LPSW L /PL/ na logično "0". V primeru posega s konzole (vpis v PSW preko visokega naslova 177776) naslovni dekoder aktivira signala A14 LOAD PSW HIGH L ter A14 LOAD PSW LOW L na logično "0" kar skupaj z uro omogoči vpis nove vsebine v PSW register.


[(B2 DISABLE LOAD PSW H V A9 LPSW L)·
· A14 LOAD PSW HIGH L] V A15 REG CLK L
SD (15÷14)
A5
BIT1 (15:14)
PSW (15:12)
M.M. NAČIN
PSW (15:14)
B
A5
PSW MUX
SD (13:12)
A
STB
SO
BIT1 (13:12)
CLR
CLK
A9 KERNEL H/PL/
B5 D SUS (1) L
SD
A5
PSW 08
SUSPENZIJA
SD 08
D
CLR CLK
NA DS
MUX
A5
SD (07÷05)
PSW (07÷05)
PRIORITETA
CLR CP
[(B2 DISABLE LOAD PSW H V A9 LPSW L)·
· A14 LOAD PSW LOW L] V A15 REG CLK L
SD 04
D
A5
PSW 04
T-BIT
CLR CP
PROC INIT L
A9 LPSW L V A15 REG CLK L
A15 REG CLK L
CP
A6
2904
Y
SD (03÷00)
BCD CARRY
A
A6
B
BCD / CC
MUX
S
(A9 KERNEL L ·
· B3 GR 8 H)
N.Z.V.C

Slika 4.18

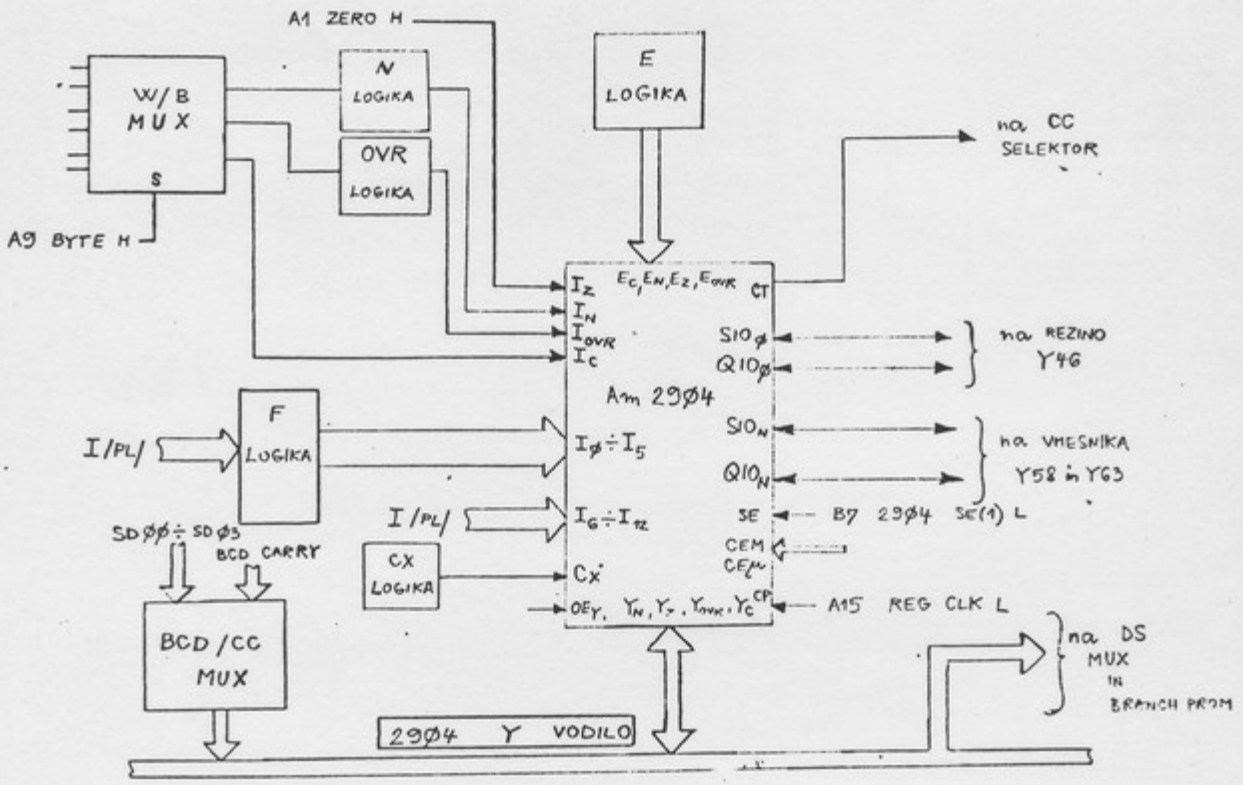
4.3 GENERIRANJE POGOJNIH KOD

Enota 2904 (E102 na A6) ima naslednje funkcije:

- makro statusni register predstavlja tisti del PSW-ja, ki shranjuje pogojne kode N, Z, V, C (podrobnejsi opis je v poglavju 4.2.10)
- sodeluje pri pogojnih vejitvah na ta nacin, da na CT izhod daje eno od pogojnih kod iz makro ali mikro statusnega registra
- generira vhodni carry za najmanj pomenbno ALU rezino in enoto 2902
- skupaj z ALU enoto omogoca pomike nad zlogom, besedo in dvojno besedo (glej poglavje 4.2.8)

4.3.1 Blok shema vhodov in izhodov enote 2904

Blok shema enote 2904 je prikazana na sliki 4.19. Dvosmerno Y vodilo je prikljuceno na BCD/CC MUX (E109 na A6) od koder dobi pogojne kode za vpis v statusni register. V okviru standardnega instrukcijskega nabora (B3 GR 8 H na logicni "0") se ob ustrezni kontroli na vhodih I0 do I5 vsebina vodila SD (signali A1 SD 00 H, A1 SD 01 H, A1 SD 02 H in A1 SD 03 H) vpise v makro statusni register. Pogojne kode iz makro statusnega registra se ob ustrezni kontroli na I vhodih posredujejo na Y vodilo, ki je spojeno na DS MUX ter BRANCH PROM. Smer prenosa V/I enote 2904 krmili signal B7 2904 OEY(1) 1 /PL/, dodatno pa signal A14 LOAD PSW L iz naslovnega dekoderja omogoci vpis pogojnih kod v makro statusni register ; pri branju vsebine PSW-ja preko visokega naslova 177776 se aktivira signal A14 PSW DATI L, ki omogoci, da se pogojne kode iz makro statusnega registra pojavijo na Y vodilu.



Slika 4.19

Funkcije F logike (slika 4.19), ki modificira mikroprogramske krmilne signale, so naslednje:

- signal B12 PROC INIT L vsili na I vhode krmiljenje, ki briše makrostatusni register
- signal A14 LOAD PSW LOW L vsili ničle na vhode I0 do I5 in s tem omogoči vpis novih pogojnih kod z Y vodila v makrostatusni register
- v primeru, da se izvaja eden izmed makroukazov, ki brišejo/postavljajo pogojne kode (pri tem se aktivirata signala B3 GR6 H na enico ter signal B3 GR6 L na ničlo) se v primeru signala B4 IR 04(1) L na "0"/"1" pojavi na I1 vhodu "1"/"0", kar skupaj z ostalimi krmilnimi vhodi zagotovi brisanje/postavljanje določenih pogojnih kod.

Polje v pipeline registru, ki generira funkcijske krmilne signale I, je dvakratno izkoriščeno. Signali B7 Z1 do B7 Z5 krmilijo I vhode po naslednji tabeli 4.6.

Tabela 4.6

B7 Z1	B7 2904 I0(1) H /Z1/	B7 2904 I6(1) H /Z1/
B7 Z2	B7 2904 I1(1) H /Z2/	B7 2904 I7(1) H /Z2/
B7 Z3	B7 2904 I1(1) H /Z3/	B7 2904 I8(1) H /Z3/
B7 Z4	B7 2904 I3(1) H /Z4/	B7 2904 I9(1) H /Z4/
B7 Z5	B7 2904 I4(1) H /Z5/	B7 2904 I10(1) H /Z5/

Ostali funkcijski krmilni signali B7 2904 I5(1) H /PL/, B7 2904 I11 (1 H /PL/ ter B7 2904 I12(1) H /PL/ pa niso prekriti.
Vloga CX logike je dvojna ; v fazi dostave (signal B6 EXEC(1) L na logični '0') pri zlogovnih makroinstrukcijah v adresnem načinu 2/4 poskrbi , da se v primeru registrov R6 ali R7 njihova vsebina poveča/zmanjša za dva. Krmilni signal B7 CX SEL(1) H /PL/ iz pipeline registra je v primeru adresnega načina 2/4 na logični '0'/'1'. V fazi eksekucije pa je možno na C0 vhod rezine Y46 s pomočjo CX izbiralnika (E112 na A6) pripeljati signal A1 ZERO H.
Enota 2904 skupaj z rezinami in dvosmernimi prenosniki (Y58 in Y63 na A6) omogoča aritmetične in logične pomike v levo ali desno nad zlogom, besedo in dvojno besedo. Signal B7 2904 I10(1) H /Z5/ definira smer pomikov. Pomike levo/desno dobimo, če je ta signal na logični enici/ničli. V primeru, da je signal A9 BYTE H na logični enici, se izvajajo pomiki nad spodnjim zlogom v rezinah. Pri vseh pomikih je potrebno imeti na logični '0' signal B7 2904 SE(1) L.
Podrobnejši opis tistega dela enote 2904, ki se tiče PSW-ja, je skupaj s pripadajočo krmilno logiko v poglavju 4.2.10.

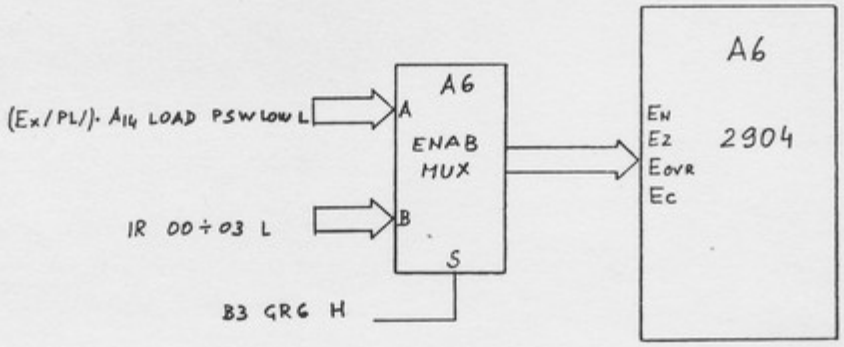
4.3.2 Pogojne kode

Vsa logika, ki je potrebna za vpis pogojnih kod v PSW je prikazana na listu A6, blok shemo pa prikazuje slika 4.18. Enota 2904 poleg ostalih funkcij nastopa tudi v vlogi PSW registra, v katerega se vpisujejo pogojne kode N, Z, V, C generirane z ALU operacijo. Za vpis pogojnih moramo na enoti 2904 definirati naslednje vhode:

- na vhodih I0, I1, I2, I3, I4 in I5 izberemo eno izmed željenih operacij vpisa (pri pogojnih kodah gre vedno za vpis v makrostatusni register)
- na enable vhodih En, Ez, Ec, Eovr s pomočjo logike CEM definiramo katere pogojne kode se lahko spremenijo (to so tiste, katerih enable vhodi so na logičnih ničlah)
- na In, Iz, Ic, Iovr vhodih pa se morajo pojaviti pogojne kode, ki ustrezajo trenutni ALU operaciji

4.3.2.1 Enable logika

Naloga logike na enable vhodih 2904 je razvidna iz slike 4.20.



Slika 4.20

Na enable vhode enote 2904 se posredujejo preko ENAB MUX-a (Y57 na A6) nespremenjeni enable signali B7 2904 EN(1) L /PL/, B7 2904 EZ(1) L /PL/ , B7 2904 EOVR(1) L /PL/ ter B7 2904 EC(1) L /PL/ iz pipeline registra v primeru, da se ne izvajajo makroukazi iz grupe 6. Ta grupa vsebuje ukaze za postavljanje in brisanje posojnih kod. Signal B3 GR6 H je na logični "1" in izbiralnik ENAB MUX (Y57 na A6) tako zakrmiljen, da se na enable vhodih 2904 pojavijo krmilni signali iz instrukcijskega registra po tabeli 4.7.

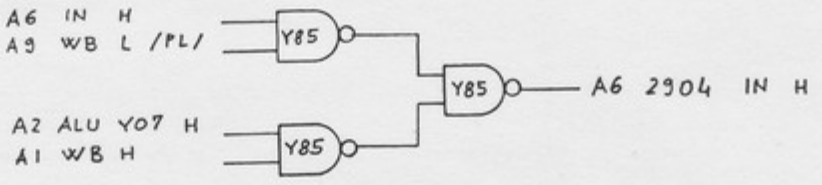
Tabela 4.7

Makro ukaz	IR03 H	IR02 H	IR01 H	IR00 H	En	Ez	Ec	Eovr
CLN, SEN	H	L	L	L	L	H	H	H
CLZ, SEZ	L	H	L	L	H	L	H	H
CLV, SEV	L	L	H	L	H	H	L	H
CLC, SEC	L	L	L	H	H	H	H	L
CCC, SCC	H	H	H	H	L	L	L	L

Vloga signala A14 LOAD PSW LOW L iz naslovnega dekoderja pa je v tem, da vsili na vse enable vhode logične "0" in tako omogoči vpis novih posojnih kod v makro statusni register.

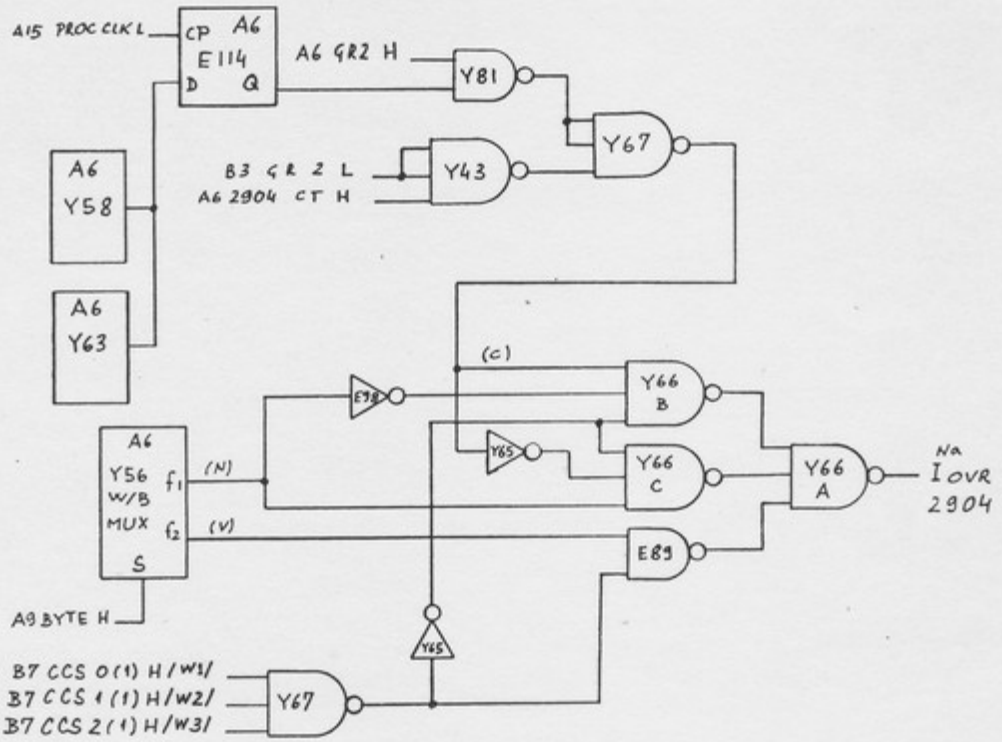
4.3.2.2 Logika na In, Iz, Iovr in Ic vhodih

Namen te logike je v tem, da na ustrezen vhod enote 2904 pripelje pravilno pogojno kodo glede na vse zahteve, ki so s tem v zvezi postavljene. V primeru da gre za zlogovno operacijo se morajo vpisati pogojne kode glede na rezultat ALU operacije nad manj pomembnim zlogom. Funkcijo izbire besednih oz. zlogovnih pogojnih kod vrši W/B MUX (Y56 na A6) v odvisnosti od signala A9 BYTE H, ki je na logični enici v primeru makroukazov nad zlogi. Signal A1 ZERO H, ki sa generirajo vse štiri rezine, je po svoji naravi izhod z odprtim kolektorjem, tako da imamo pri operacijah nad besedo ali nad zlogom, pogojno kodo Z generirano fizično z istim signalom. Tako pogojne kode Z (A1 ZERO H) ni potrebno voditi preko W/B MUX-a.
Pogojna koda carry se izbira med signaloma A2 BYTE MUXB C H (generira sa rezina Y49 na Cn+4 izhodu v primeru operacije nad zlogom) in A4 BYTE MUXA C H (generira sa rezina Y54 na carry izhodu, ko gre za besedno operacijo). Izhod W/B MUX-a A6 2904 Ic H je pripeljan direktno na Ic vhod enote 2904.
Pogojna koda N se izbira na W/B MUX-u med signaloma A2 BYTE MUXB N H, če gre za zlog ter signalom A4 BYTE MUXA N H, če gre za besedo. Signal A6 In H, ki sa dobimo na izhodu W/B MUX-a, peljemo preko N logike (slika 4.21) na In vhod enote 2904.



Slika 4.21

Naloga vezja je generiranje pravilne pogojne kode N v primeru, ko se izvaja MOVB makro ukaz. Takrat je aktiviran signal A9 WB L /PL/ na logično "0", tako da N logika na In vhod 2904 pripelje signal A2 ALU Y07 H, ki ravno predstavlja predznak zloga, nad katerim se izvaja v rezinah pomik predznaka. V vseh drugih situacijah pa dobimo na In vhodu 2904 tisto pogojno kodo, ki jo je izbral W/B MUX.
Pogojna koda V (prekoračitev) se izbere na W/B MUX-u med signaloma A2 BYTE MUXB OVR H in A4 BYTE MUXA OVR H in jo spremenimo še z OVR logiko (glej sliko 4.22), katere izhod vodimo na Iovr vhodno linijo enote 2904.



Slika 4.22

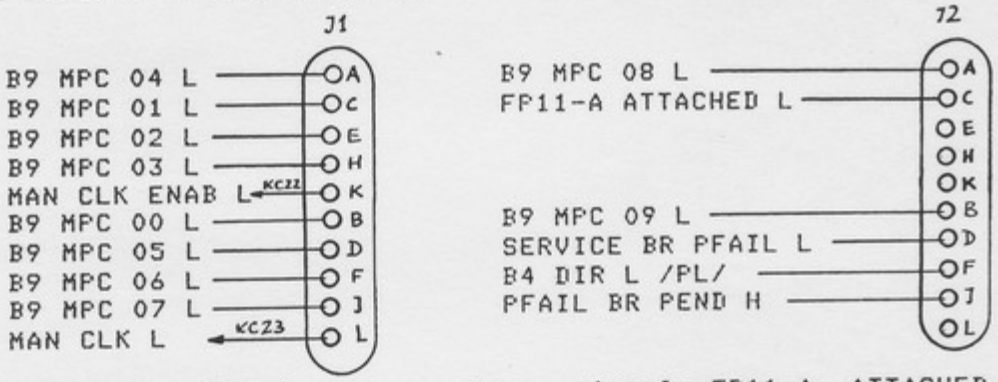
Najprej si oglejmo kako poteka najbolj običajen vpis pogojne kode V. Nespremenjeno pogojno kodo V, ki jo generira ALU enota preko W/B MUX-a vodimo na vrata E89 ter preko vrat Y66 A na Iovr vhod 2904 in sicer v vseh tistih kombinacijah, ko nimamo na CC selektorju izbrane kombinacije SHOVR. V primerih, ko želimo tako pogojno kodo V, ki je rezultat ekskluzivne OR operacije med trenutno kodo N in pogojno kodo C, pa na CC selektorju izberemo kombinacijo SHOVR. Pri tej kombinaciji so vsi signali B7 CCS 0(1) H /W1/, B7 CCS1 (1)H/W2/ in B7 CCS2(1) H/W3/ na logični "1", kar aktivira vrata Y66 B ter Y66 C, tako da izvedejo ekskluziv OR operacijo med N in C pogojno kodo. Na ta način dobimo šele pravilno pogojno kodo V, ki jo vodimo na Iovr vhod enote 2904, kjer se vpiše v statusni register. Pri tem smo z mikroprogramom poskrbeli, da se je na CT izhodu enote 2904 pojavila pogojna koda C-signal A6 2904 CT H. V primeru, da se izvajajo makroinstrukcije ASH ter ASHC (aktivirana sta signala A6 GR2 H na logični "1", signal B3 GR2 L pa na logični "0"), pa se omogoči vpliv Q izhoda D celice E114 na vrata Y81. V D celico se je ob koncu prejšnje mikroinstrukcije vpisala tista informacija , ki smo jo dobili kot posledico pomika v rezinah. Torej shranjuje D celica pogojno kodo C, OVR logika pa izvaja ekskluziv OR operacijo med N in C kodo, kar da za rezultat pravilno pogojno kodo V, za vpis v enoto 2904.

4.4 APARATURNA OPREMA ZA RAZSIRITEV

Arhitektura CPU enote DELTA 16/BIT SLICE omogoča priključitev standardne enote FP procesorja (FP 11-A), enote CACHE (KK 11-A), oz. dodatne plošče za 'Writable Control Store', ki vsebuje tudi dodaten niz registrov.

4.4.1 Priključitev FP procesne enote

V okviru CPU enote je mozna priključitev FP procesne enote preko J1 in J2 konektorja na B plošči.

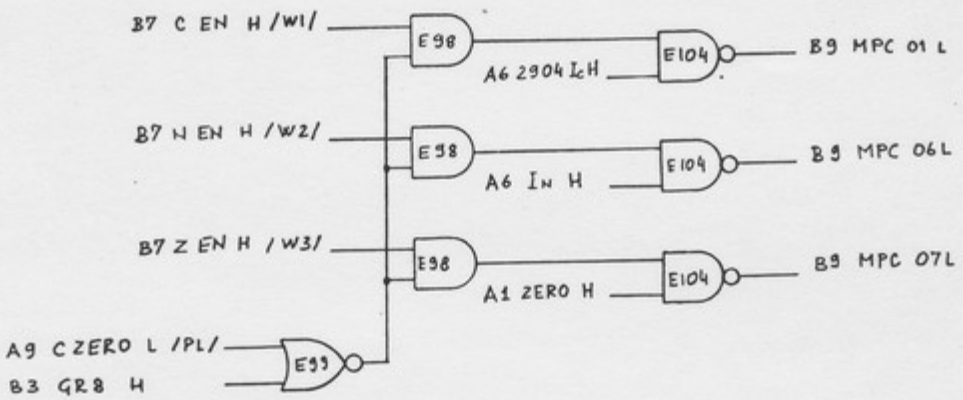


Preko teh dveh konektorjev se v primeru signala FP11-A ATTACHED L NA logični '0' določa mikroprogramska adresa naslednje mikroinstrukcije. V CPU enoti se na mikroprogramskih lokacijah od 512 do 1024 nahaja mikroprogram za izvajanje FP insrukcij, ki skrbi za dostavo in shranjevanje operandov v pomnilnik ter za vpis pogojnih kod v okviru PSW-ja, kjer v glavnem sodeluje CPU enota.
Pri generiranju mikroprogramske adrese je na adresnih linijah B9 MPC 01 L, B9 MPC 06 L IN B9 MPC 07 L dodana logika (list B9), ki omogoča skoke glede na N, C in Z bite (slika 4.23).
Signal A9 CZERO L /PL/ je pri izvajanju standardnega nabora vedno na logični '1' in tako je izločena vsilitev kakršnekoli spremembe na mikroprogramsko adreso. Isto velja tudi za CIS instrukcijski nabor, le da je takrat signal B3 GR8 H na logični '1'.
Pri izvajanju FP instrukcij sta signala A9 CZERO 1 /PL/ in B3 GR8 H na logični '0' in tako so glede na ENABLE krmilne signale B7 CEN H /W1/, B7 NEN H /W2/ in B7 ZEN H /W3/ omogočeni skoki po mikroprogramskih adresah.

4.4.2 Implementacija CIS (Commercial Instruction Set)

Pri CPU enoti je upostevana možnost izvajanja CIS instrukcij, če ji dodamo tri zunanje RAM-e po 16 registrov in dodatni mikroprogramski pomnilnik.
Logika, ki skrbi za naslavljanje in krmiljenje branja ali vpisovanja v dodani niz RAM-ov A DEKODER (E105), B DEKODER (E108), WRITE DEKODER (E108) na listu A9.
Pri A DEKODER-ju naslovni liniji A8 ADR A4 H in A8 ADR A5 H določata, iz katerega RAM-a bodo izbrani A izvorni podatki, če smo

s signalom B7 2903 EA(1) L na logični '0' izbrali izvorni podatek iz RAM-a.



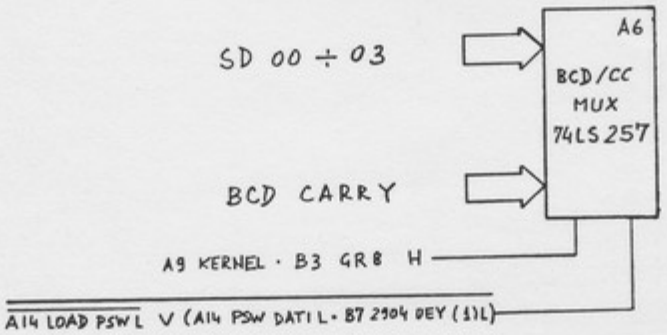
Slika 4.23

Tabela 4.8

A4	A5	Izvor podatka
0	0	ALU RAM
0	1	RAM 1
1	0	RAM 2
1	1	RAM 3

Dodatni niz treh RAM-ov po 16 registrov ni dodan obstoječi centralno procesni enoti (CPU).

Pri B DEKODER-ju naslovni liniji A7 ADR B4 H in A7 ADR B5 H določata, iz katerega RAM-a bodo izbrani B izvorni podatki po enaki tabeli kot je za A izvorni podatek.
WRITE DEKODER je krmiljen tudi z naslovnima linijama A7 ADR B4 H in A7 ADR B5 H in njegovi izhodni signali omogočajo vpis podatkov v RAM-e po isti tabeli.
Logika, ki omogoča vpis delnih CARRY bitov, ki prihajajo na Cn vhode ALU rezin v AM2904 (BCD/CC MUX) je prikazana na sliki 4.24.



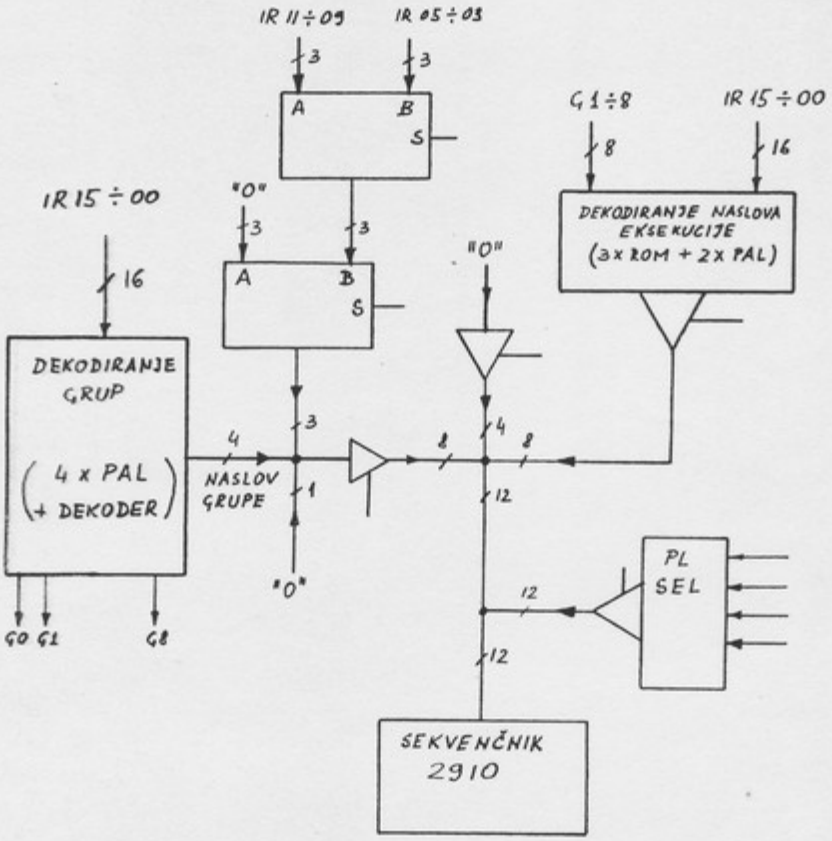
Slika 4.24

Pri izvajanju CIS instrukcij so vsi signali, razen B7 2904 OEY(1) L, na logični "1", kar pomeni, da lahko s tem signalom iz pipeline registra [B7 2904 OEY(1) L na logični "1"] omogočimo na Y vhod enote AM2904 BCD CARRY podatke.
Vsilitev ničel na Cn vhode ALU rezin je opisana v poglavju (ALU funkcije).
CIS ROM omogoča pretvorbo 8-bitnega podatka v 4-bitnega in obratno pretvorbo 4-bitnega in predznaka v 8-bitni podatek.

4.5 DEKODIRANJE INSTRUKCIJ

4.5.1 Uvod

Naloga DEKODIRNE LOGIKE je, da na osnovi vsebine v instrukcijskem registru po dostavi vsake instrukcije določi mikroprogramsko adreso, na kateri se prične izvajati instrukciji ustrezen mikroprogram.
Glede na to, da ima precej instrukcij skupne značilnosti, predvsem kar se tiče dostavljanja operandov oz. njihovih adres, smo celoten nabor instrukcij razdelili v 9 grup (8 iz standardnega nabora, eno grupo pa predstavljajo CIS instrukcije).
Zaradi grupiranja instrukcij je naloga DEKODIRNE LOGIKE dvojna: prvič, da za vsako instrukcijo določi pripadnost grupi, ter da hkrati pripravi adreso za eksekucijski del mikroprograma, ki je seveda različen za vsako instrukcijo. Vsaki grupi (razen grupam 0,3,6 in 7) ustreza mikroprogram, ki za vse instrukcije v grupi izvaja skupno dostavo operandov oz. adres operandov. V času izvajanja skupnega mikroprograma, drugi del DEKODIRNE LOGIKE pripravi eksekucijsko adreso, ki ustreza instrukciji v instrukcijskem registru.
Krmiljenje vstopne adrese v sekvenčnik 2910 je pod kontrolo mikroprograma. Ob dostavi nove instrukcije je odprta pot, ki nosi adreso grupe, ob koncu skupnega dela mikroprograma, ki ustreza grupi pa se odpre pot, ki pripelje eksekucijsko adreso instrukcije v sekvenčnik.
DEKODIRNA LOGIKA je zasnovana tako, da omogoča spremembe enostavno z reprogramiranjem njenih pomnilnih elementov PAL oz. ROM. To je pomenbno z vidika instaliranja novih specijalnih instrukcij.
Osnovno blok shemo DEKODIRNE LOGIKE prikazuje slika 4.25.



Slika 4.25

4.5.2 Instrukcijski register

Vsako makro instrukcijo, ki jo dostavimo iz glavnega pomnilnika, shranimo v instrukcijski register (IR). Vpis podatkov iz SD 00-15 vodila se izvede ob zadnji fronti urinega signala A15 PROC CLK L, če je signal A9 DIR L na logični '0'. Ta signal aktivira mikroprogram v fazi dostave makroukaza. Instrukcijski register sestavljajo naslednji elementi. Dva 4-bitna D registra 74S175 (E63 in E75), en 8-bitni D register 74S273 (E69) ter D celica 74S74 (E60). V IR registru se nahaja makroukaz od prve dostavne mikroinstrukcije dalje pa do pojava zadnje fronte signala B7 BUT SERVICE (1) H, ki sa aktivira mikroprogram na prehodu iz eksekucije v dostavo. Pri tem se brišejo vsi IR biti razen B4 IR 15(1) H, kar preprečuje dekodirni logiki, da bi spoznala HALT instrukcijo.
Če nastopi napaka na Unibus-u (signal B10 SET BE L se postavi v logično '0') in je aktiviran signal B7 DBE H na logično '1', se briše celoten instrukcijski register, kar povzroči ustavitev procesorja. Napake na Unibus vodilu, ki se pojavljajo, kadar

signal B7 DBE H ni aktiven (po vrednosti je na logični '0'), nimajo vpliva na IR. Signal B7 DBE H je aktiviran le v določenih mikroinstrukcijah prekinitvene mikrosekvence in preprečuje nastop nove pasti, preden je uspešno končana prekinitvena sekvenca, ki jo je zahtevala določena napaka. Na ta način onemogočimo dvojne napake in zagotovimo, da se v slučaju pojava napake v prenosih CPU-glavni pomnilnik v prekinitveni mikrosekvenci (sekvenca INTRA) procesor ustavi (brisanje IR).

4.5.3 Grupiranje instrukcij

GRUPA 0:

	IR[15 - 00]															
Instrukcija	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MSB			OP KODA					[MOD/REG]							LSB
BPT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
IOT	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
EMT	1	0	0	0	1	0	0	0	[x	x	x	x	x	x	x	x]
TRAP	1	0	0	0	1	0	0	1	[x	x	x	x	x	x	x	x]
											MOD		REG			
JMP	0	0	0	0	0	0	0	0	0	1	[0	0	0	x	x	x]
									REG		MOD		REG			
JSR	0	0	0	0	1	0	0	[x	x	x	0	0	0	x	x	x]
(FP11-A)	1	1	1	1	[x	x	x	x	x	x	x	x	x	x	x	x]

$$F0 = f1 \vee f2 \vee f3 \vee f4 \vee f5 \vee f6 \vee f7 \vee f8$$

GRUPA 1:

CMP(B)	x	0	1	0	[	S		S	|	D		D]
BIT(B)	x	0	1	1	[	S		S	|	D		D]
BIC(B)	x	1	0	0	[	S		S	|	D		D]
BIS(B)	x	1	0	1	[	S		S	|	D		D]
ADD	0	1	1	0	[	S		S	|	D		D]
SUB	1	1	1	0	[	S		S	|	D		D]
MOV(B)	x	0	0	1	[	S		S	|	D		D]

$$F1 = \overline{14} * 12 \vee 14 * \overline{13} \vee \overline{13} * 12 \quad \text{(stevilke pomenijo bite v IR)}$$

GRUPA 2:

MUL	0	1	1	1	0	0	0	[	R	|	S	S]
DIV	0	1	1	1	0	0	1	[	R	|	S	S]
ASH	0	1	1	1	0	1	0	[	R	|	S	S]
ASHC	0	1	1	1	0	1	1	[	R	|	S	S]

$$F2 = \overline{15} * 14 * 13 * 12 * \overline{11}$$

GRUPA 3:
========

BR	I	0	0	0	0	0	0	0	1	[	ODMIK]
BNE	I	0	0	0	0	0	0	1	0	[	-II-]
BEQ	I	0	0	0	0	0	0	1	1	[	-II-]
BGE	I	0	0	0	0	0	1	0	0	[	-II-]
BLT	I	0	0	0	0	0	1	0	1	[	-II-]
BGT	I	0	0	0	0	0	1	1	0	[	-II-]
BLE	I	0	0	0	0	0	1	1	1	[	-II-]
BPL	I	1	0	0	0	0	0	0	0	[	-II-]
BMI	I	1	0	0	0	0	0	0	1	[	-II-]
BHI	I	1	0	0	0	0	0	1	0	[	-II-]
BLOS	I	1	0	0	0	0	0	1	1	[	-II-]
BVC	I	1	0	0	0	0	1	0	0	[	-II-]
BVS	I	1	0	0	0	0	1	0	1	[	-II-]
BCC	I	1	0	0	0	0	1	1	0	[	-II-]
BCS	I	1	0	0	0	0	1	1	1	[	-II-]

$$F3 = \overline{14}*\overline{13}*\overline{12}*\overline{11}*(15 \vee 10 \vee 9 \vee 8)$$

GRUPA 4:
========

XOR	I	0	1	1	1	1	0	0	[	R		I	D	D]
SWAB	I	0	0	0	0	0	0	0	0	1	1	[	D	D]
CLR(B)	I	x	0	0	0	1	0	1	0	0	0	[	D	D]
COM(B)	I	x	0	0	0	1	0	1	0	0	1	[	D	D]
INC(B)	I	x	0	0	0	1	0	1	0	1	0	[	D	D]
DEC(B)	I	x	0	0	0	1	0	1	0	1	1	[	D	D]
NEG(B)	I	x	0	0	0	1	0	1	1	0	0	[	D	D]
ADC(B)	I	x	0	0	0	1	0	1	1	0	1	[	D	D]
SBC(B)	I	x	0	0	0	1	0	1	1	1	0	[	D	D]
TST(B)	I	x	0	0	0	1	0	1	1	1	1	[	D	D]
ROR(B)	I	x	0	0	0	1	1	0	0	0	0	[	D	D]
ROL(B)	I	x	0	0	0	1	1	0	0	0	1	[	D	D]
ASR(B)	I	x	0	0	0	1	1	0	0	1	0	[	D	D]
ASL(B)	I	x	0	0	0	1	1	0	0	1	1	[	D	D]
SXT	I	0	0	0	0	1	1	0	1	1	1	[	D	D]

$$F4 = \overline{15}*\overline{14}*\overline{13}*\overline{12}*\overline{11}*\overline{10}*\overline{9} \vee$$
$$\vee\ \overline{15}*\overline{14}*\overline{13}*\overline{12}*\overline{9}*7*6*(\overline{11}*\overline{10}*\overline{8} \vee 11*10*8) \vee$$
$$\vee\ \overline{14}*\overline{13}*\overline{12}*\overline{11}*\overline{10}*9 \vee$$
$$\vee\ \overline{14}*\overline{13}*\overline{12}*\overline{11}*\overline{10}*\overline{9}*\overline{8}$$

GRUPA 5:
========

JSR	I	0	0	0	0	1	0	0	[	R		I	0	0	0	REG	]
JMP	I	0	0	0	0	0	0	0	0	0	1	[	0	0	0	REG	]
MFPI(D)	I	x	0	0	0	1	1	0	1	0	1	[	S			S	]
MFPS	I	1	0	0	0	1	1	0	1	1	1	[	D			D	]
MTPS	I	1	0	0	0	1	1	0	1	0	0	[	S			S	]

$$F5 = \overline{15}*\overline{14}*\overline{13}*\overline{12}*\overline{11}*\overline{10}*\overline{9}*(5 \vee 4 \vee 3) \vee$$
$$\vee\ \overline{15}*\overline{14}*\overline{13}*\overline{12}*\overline{11}*\overline{10}*\overline{9}*\overline{8}*\overline{7}*\overline{6}*(5 \vee 4 \vee 3) \vee$$
$$\vee\ \overline{14}*\overline{13}*\overline{12}*\overline{11}*\overline{10}*\overline{9}*\overline{8}*(15*\overline{6} \vee 15*\overline{7} \vee 7*\overline{6})$$

GRUPA 6:
========

															N	Z	V	C	
CLC	I	0	0	0	0	0	0	0	0	1	0	1	0	[	0	0	0	1	]
CLV	I	0	0	0	0	0	0	0	0	1	0	1	0	[	0	0	1	0	]
CLZ	I	0	0	0	0	0	0	0	0	1	0	1	0	[	0	1	0	0	]
CLN	I	0	0	0	0	0	0	0	0	1	0	1	0	[	1	0	0	0	]
CCC	I	0	0	0	0	0	0	0	0	1	0	1	0	[	1	1	1	1	]
SEC	I	0	0	0	0	0	0	0	0	1	0	1	1	[	0	0	0	1	]
SEV	I	0	0	0	0	0	0	0	0	1	0	1	1	[	0	0	1	0	]
SEZ	I	0	0	0	0	0	0	0	0	1	0	1	1	[	0	1	0	0	]
SEN	I	0	0	0	0	0	0	0	0	1	0	1	1	[	1	0	0	0	]
SCC	I	0	0	0	0	0	0	0	0	1	0	1	1	[	1	1	1	1	]
NOP	I	0	0	0	0	0	0	0	0	1	0	1	x	[	0	0	0	0	]

$$F6 = \overline{15}*\overline{14}*\overline{13}*\overline{12}*\overline{11}*\overline{10}*\overline{9}*\overline{8}*\overline{7}*\overline{6}*\overline{5}$$

Opomba: Možno je hkrati setirati oz. resetirati poljubno
kombinacijo CC bitov (Npr.:CLVZ, SECVZ)

GRUPA 7:
========

HALT	I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
SOB	I	0	1	1	1	1	1	1	[	R		I		ODMIK					]
MARK	I	0	0	0	0	1	1	0	1	0	0	[	N			N			]
RTS	I	0	0	0	0	0	0	0	0	1	0	0	0	0	[	R			]
WAIT	I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1		
RTI	I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0		
RTT	I	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0		
RESET	I	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1		
MTPI(D)	I	x	0	0	0	1	1	0	1	1	0	[	D			D			]

$$F7 = \overline{15}*\overline{14}*\overline{13}*\overline{12}*\overline{11}*\overline{10}*\overline{9}*\overline{8}*\overline{6}*\overline{5}*\overline{4}*\overline{3}*(7 \vee \overline{1}*0 \vee \overline{1}*0 \vee \overline{2}*0) \vee$$
$$\vee\ \overline{15}*\overline{14}*\overline{13}*\overline{12}*\overline{11}*\overline{10}*\overline{9} \vee$$
$$\vee\ \overline{14}*\overline{13}*\overline{12}*\overline{11}*\overline{10}*\overline{9}*\overline{8}*(7*\overline{6} \vee \overline{15}*\overline{6})$$

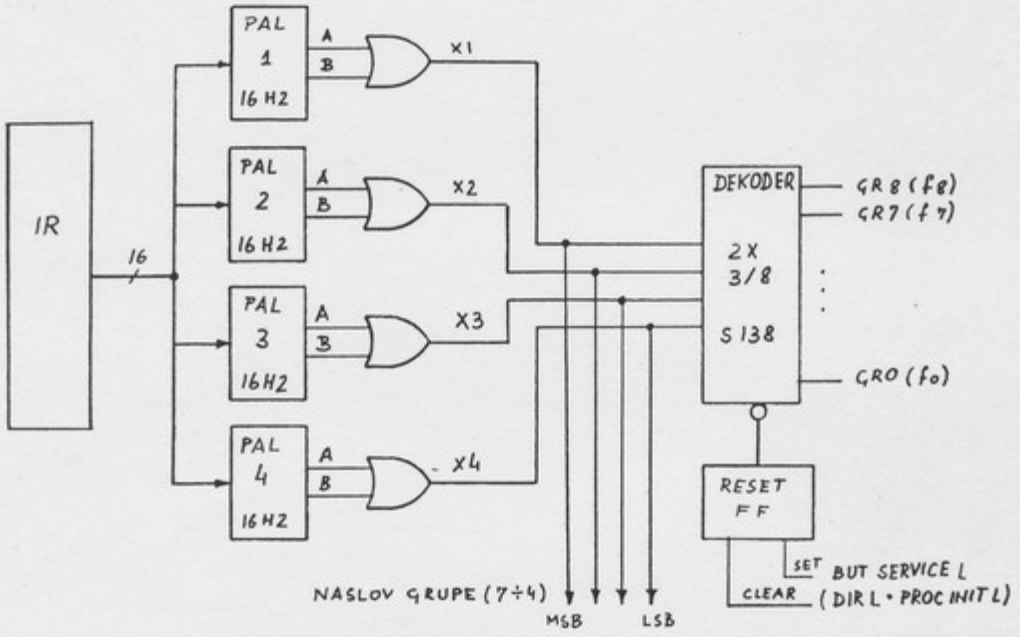
GRUPA 8 (CIS) :
==========

L2D	0	1	1	1	1	1	0	0	0	0	0	1	OC	R	]
MOVC	0	1	1	1	1	1	0	0	0	0	0	1	1	0	0	0
MOVCI	0	1	1	1	1	1	0	0	0	1	0	1	1	0	0	0
MOVRC	0	1	1	1	1	1	0	0	0	0	0	1	1	0	0	1
MOVRCI	0	1	1	1	1	1	0	0	0	1	0	1	1	0	0	1
MOVTC	0	1	1	1	1	1	0	0	0	0	0	1	1	0	1	0
MOVTCI	0	1	1	1	1	1	0	0	0	1	0	1	1	0	1	0
LOCC	0	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
LOCCI	0	1	1	1	1	1	0	0	0	1	1	0	0	0	0	0
SKPC	0	1	1	1	1	1	0	0	0	0	1	0	0	0	0	1
SKPCI	0	1	1	1	1	1	0	0	0	1	1	0	0	0	0	1
SCANC	0	1	1	1	1	1	0	0	0	0	1	0	0	0	1	0
SCANCI	0	1	1	1	1	1	0	0	0	1	1	0	0	0	1	0
SPANC	0	1	1	1	1	1	0	0	0	0	1	0	0	0	1	1
SPANCI	0	1	1	1	1	1	0	0	0	1	1	0	0	0	1	1
CMPC	0	1	1	1	1	1	0	0	0	0	1	0	0	1	0	0
CMPCI	0	1	1	1	1	1	0	0	0	1	1	0	0	1	0	0
MATC	0	1	1	1	1	1	0	0	0	0	1	0	0	1	0	1
MATCI	0	1	1	1	1	1	0	0	0	1	1	0	0	1	0	1
ADDN	0	1	1	1	1	1	0	0	0	0	1	0	1	0	0	0
ADDNI	0	1	1	1	1	1	0	0	0	1	1	0	1	0	0	0
SUBN	0	1	1	1	1	1	0	0	0	0	1	0	1	0	0	1
SUBNI	0	1	1	1	1	1	0	0	0	1	1	0	1	0	0	1
ASHN	0	1	1	1	1	1	0	0	0	0	1	0	1	1	1	0
ASHNI	0	1	1	1	1	1	0	0	0	1	1	0	1	1	1	0
CPMN	0	1	1	1	1	1	0	0	0	0	1	0	1	0	1	0
CPMNI	0	1	1	1	1	1	0	0	0	1	1	0	1	0	1	0
ADDP	0	1	1	1	1	1	0	0	0	0	1	1	1	0	0	0
ADDPI	0	1	1	1	1	1	0	0	0	1	1	1	1	0	0	0
SUBP	0	1	1	1	1	1	0	0	0	0	1	1	1	0	0	1
SUBPI	0	1	1	1	1	1	0	0	0	1	1	1	1	0	0	1
MULP	0	1	1	1	1	1	0	0	0	0	1	1	1	1	0	0
MULPI	0	1	1	1	1	1	0	0	0	1	1	1	1	1	0	0
DIVP	0	1	1	1	1	1	0	0	0	0	1	1	1	1	0	1
DIVPI	0	1	1	1	1	1	0	0	0	1	1	1	1	1	0	1
ASHP	0	1	1	1	1	1	0	0	0	0	1	1	1	1	1	0
ASHPI	0	1	1	1	1	1	0	0	0	1	1	1	1	1	1	0
CMPP	0	1	1	1	1	1	0	0	0	0	1	1	1	0	1	0
CMPPI	0	1	1	1	1	1	0	0	0	1	1	1	1	0	1	0
CVTNL	0	1	1	1	1	1	0	0	0	0	1	0	1	0	1	1
CVTNLI	0	1	1	1	1	1	0	0	0	1	1	0	1	0	1	1
CVTLN	0	1	1	1	1	1	0	0	0	0	1	0	1	1	1	1
CVTLNI	0	1	1	1	1	1	0	0	0	1	1	0	1	1	1	1
CVTPL	0	1	1	1	1	1	0	0	0	0	1	1	1	0	1	1
CVTPLI	0	1	1	1	1	1	0	0	0	1	1	1	1	0	1	1
CVTLP	0	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1
CVTLPI	0	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1
CVTNP	0	1	1	1	1	1	0	0	0	0	1	0	1	1	0	1
CVTNPI	0	1	1	1	1	1	0	0	0	1	1	0	1	1	0	1
CVTPN	0	1	1	1	1	1	0	0	0	0	1	0	1	1	0	0
CVTPNI	0	1	1	1	1	1	0	0	0	1	1	0	1	1	0	0
L3D	0	1	1	1	1	1	0	0	0	0	1	1	OC	R	]

$$F8 = \overline{15}*14*13*12*11*10*\overline{9}*\overline{8}*\overline{7}*\overline{6}*\overline{4}*\overline{3} \vee$$
$$\overline{15}*14*13*12*11*10*\overline{9}*\overline{8}*\overline{7}*\overline{5}*\overline{4}*\overline{3}*\overline{2}*(\overline{1} \vee \overline{0}) \vee$$
$$\overline{15}*14*13*12*11*10*\overline{9}*\overline{8}*\overline{7}*\overline{5}*\overline{4}*\overline{3}*(\overline{2} \vee \overline{1}) \vee$$
$$\overline{15}*14*13*12*11*10*\overline{9}*\overline{8}*\overline{7}*\overline{5}*\overline{3}$$

4.5.4 Dekodiranje skup

Blok shemo enote za dekodiranje skup v okviru DEKODIRNE LOGIKE podaja slika 4.26.



Slika 4.26

PAL vezja skupirajo instrukcije v 9 skup (poglavje 4.5.3). Ker je [log 9]=4, določajo adreso skupe štiri adresne linije (X1-X4). Le te predstavljajo vhode (7-4) pri naslavljanju mikroprogramskega pomnilnika. Adresiranje mikroprograma, skupnega za instrukcije v okviru ene skupe, je izvedeno na osnovi adrese:

```
0 0 0 0 X1 X2 X3 X4 0  MOD    =  ADR.GRUPE(11-00)
MSB                        LSB
11. . . 7  .  .  4  . . . 0
```

kjer MOD pomenijo bite (2-0) za način adresiranja. Pri tem lahko na mesto MOD bitov pripeljemo ali bite IR(11-9) ali bite IR(5-3) (glej sliko 4.25).
Adresne linije X1-X4 določajo grupe na osnovi tabele 1:

Tabela 9

	X1	X2	X3	X4	Stevilo konjunkcij
G0	0	0	0	0	4
G1	0	0	0	1	3
G2	0	0	1	0	1
G3	0	0	1	1	4
G4	0	1	0	0	5
G5	0	1	0	1	9
G6	0	1	1	0	1
G7	1	0	1	0	7
G8	1	0	0	0	6

Tabela 1 je bila zgrajena ob upoštevanju omejitev izhodov PAL vezij (16H2), ki disjunktivno povezujejo le do 8 konjuktivnih izrazov.
Adresne linije X1-X4 preko dekodirnega vezja določajo grupe tudi eksplicitno, kar potrebujemo pri dekodiranju eksekucijskih adres. Dekodirno vezje blokiramo s signalom BUT SERVICE L, odpremo pa s signalom A9 DIR(1) L /PL/.
Iz tabele 9 je sicer razvidno, katere grupe realizira posamezen PAL (npr. PAL1 pokriva G7 in G8), ni pa razvidno, kako so posamezni PAL-i programirani. To je posledica dejstva, da F5, ki opisuje G5, vsebuje več kot 8 konjuktivnih izrazov, kar pomeni, da zaseda delno tudi drugi izhod iz PAL-a. Tabela 10 zato podaja, kako so uporabljeni izhodi posameznih PAL-ov.

Tabela 10

	A(pin 15)	B(pin 16)	
PAL1	F7	F8	X1 = F7 V F8
PAL2	F5 V F6	F4 V F5	X2 = F4 V F5 V F6
PAL3	F2 V F3	F6 V F7	X3 = F2 V F3 V F6 V F7
PAL4	F3 V F5	F1 V F5	X4 = F1 V F3 V F5

V Tabeli 10 pomenijo :

$$F5 = \overline{15*14*13*12*11*10*9*8*7*6*(4 \vee 3)}$$

$$F5 = \overline{14*13*12*11*10*9*8*(15*6 \vee 15*7 \vee 7*6)} \vee \overline{15*14*13*12*11*10*9*(5 \vee 4 \vee 3)} \vee \overline{15*14*13*12*11*10*9*8*6*5}$$

$$F5 = \overline{14*13*12*11*10*9*8*(15*6 \vee 15*7 \vee 7*6)} \vee \overline{15*14*13*12*11*10*9*(5 \vee 4)}$$

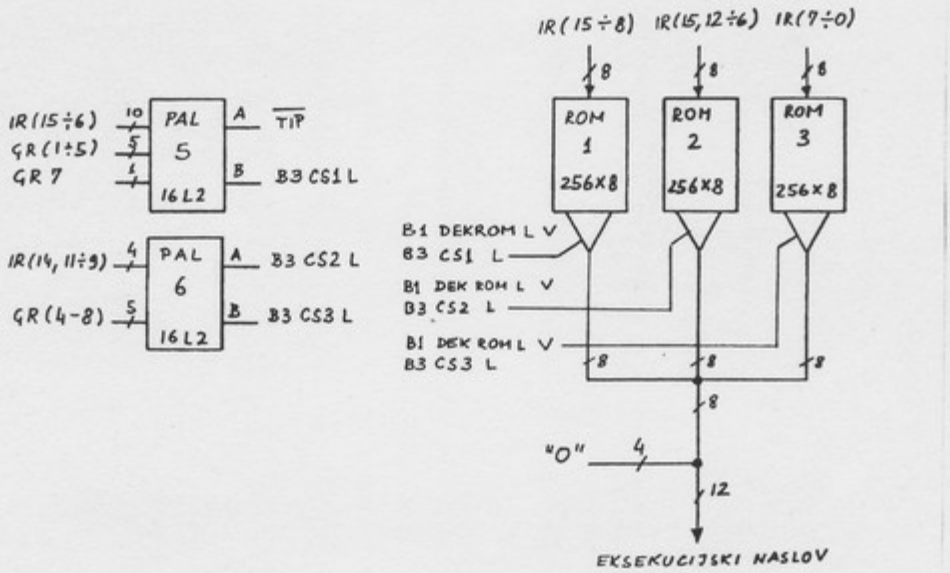
$$F5 = \overline{15*14*13*12*11*10*9*3} \vee \overline{15*14*13*12*11*10*9*8*7*6*(5 \vee 4 \vee 3)}$$

Pri tem velja:

$F5 = F5 \vee F5$ in $F5 = F5 \vee F5$

4.5.5 Dekodiranje eksekucijske adrese

Blok shemo logike za dekodiranje eksekucijskih adres podaja slika 4.27.



Slika 4.27

PAL 5 in 6 sta tipa 16L2, kar pomeni, da konjuktivne izraze združuje (Pierce). Zato so izhodi podani v negirani obliki. Vloga PAL-ov 5 in 6 je, da odpirajo ROM-e 1-3, ki hranijo eksekucijske adrese. Vedno je odprt le en ROM, kar pomeni, da vsak ROM vsebuje celotno 8-bitno eksekucijsko adreso. Instrukcije so tukaj razdeljene v tri skupine glede na to, kateri biti v IR so pomenbni s stališča operacijske kode. Tabela 11 prikazuje, kako so instrukcije porazdeljene po ROM-ih. Številke v oklepajih pomenijo število instrukcij iz grupe.

Tabela 11

ROM 1	G1(7), G2(4), G4(1), G7(1), G5(1)
ROM 2	G4(14), G5(4), G7(2)
ROM 3	G7(6), G8

Grupe 0, 3, 6 niso zajete v nobenem ROM-u, to pa zato, ker je njihova eksekucija realizirana na specifičen nacin. Instrukcije iz grupe 0 so realizirane v okviru INTRA rutine, ki izvede s pomočjo specifičnega TRAP vektorja povezavo z MACRO rutino. Instrukcije iz grupe 3 so realizirane s pomočjo BRANCH ROM vezja, instrukcije iz grupe 6 pa vsebujejo eksekucijsko mikroinstrukcijo že v okviru mikroprograma grupe 6. Vse te posebnosti so bile vpeljane zaradi hitrejše eksekucije instrukcij.
Tabela 12 podaja eksekucijske adrese mikroprogramskega pomnilnika, kjer se nahajajo začetki eksekucijskih sekvenc za posamezne instrukcije.

Tabela 12

Instrukcija	ROM 1		ROM 2		ROM 3		Pripadnost grupi
	A	D	A	D	A	D	
CMP(B)	2X AX	3B					1
BIT(B)	3X BX	3C					1
BIC(B)	4X CX	61					1
BIS(B)	5X DX	62					1
ADD	6X	6E					1
SUB	EX	6F					1
MOV	1X	81					1
MOVB	9X	BF					1
MUL	70 71	B9					2
DIV	72 73	BB					2
ASH	74 75	AA					2
ASHC	76 77	B8					2
XOR	78 79	91					4
SOB	7E 7F	86					7
JSR	08 09	8B					5
SWAB			03	69			4
CLR(B)			28 A8	37			4
COM(B)			29 A9	38			4

INC(B)			2A AA	3A			4
DEC(B)			2B AB	39			4
NEG(B)			2C AC	3D			4
ADC(B)			2D AD	3E			4
SBC(B)			2E AE	63			4
TST(B)			2F AF	3F			4
ROR(B)			30 B0	65			4
ROL(B)			31 B1	68			4
ASR(B)			32 B2	82			4
ASL(B)			33 B3	92			4
SXT			37	6B			4
JMP			01	90			5
MFPI(D)			35 B5	9A			5
MFPS			B7	9F			5
MTPS			B4	93			5
MARK			34	7B			7
MTPI(D)			36 B6	98			7
RTS					80-87	70	7
WAIT					01	75	7
RTI					02	78	7
RTT					06	78	7

RESET					05	74	7
HALT					00	BE	7

Vsebine PAL-ov 5 in 6 podajajo enačbe izhodov:

$$TIP = \overline{14}*\overline{13} \vee \overline{14}*\overline{13} \vee \overline{14}*\overline{12} \vee \overline{14}*\overline{13}*\overline{12}*\overline{11}*\overline{10}*9 \vee$$
$$\vee \overline{14}*\overline{13}*\overline{12}*\overline{11}*\overline{10}*\overline{9}*\overline{8} \vee$$
$$\vee \overline{15}*\overline{14}*\overline{13}*\overline{12}*\overline{11}*\overline{10}*\overline{9}*\overline{8}*\overline{7}*\overline{6} \vee$$
$$\vee \overline{15}*\overline{14}*\overline{13}*\overline{12}*\overline{11}*\overline{10}*\overline{9}*\overline{8}*\overline{7}*\overline{6}$$

$$CS1 = F1 \vee F2 \vee F3 \vee F4*\overline{14} \vee F7*\overline{14} \vee F5*\overline{11}*\overline{10}$$

$$CS2 = F4*\overline{14} \vee F5*\overline{11} \vee F5*\overline{10} \vee F7*\overline{10}*\overline{9}$$

$$CS3 = F6 \vee F7*\overline{11} \vee F8$$

V zgornjih izrazih pomeni:

TIP - funkcija ločuje med instrukcijami, ki so lahko besedne ali zlogovne (TIP=1) in instrukcijami, ki so samo besedne (TIP=0)

F4*14 - XOR instrukcija

F7*14 - SOB

F5*11*10 - JSR

F4*14 - XOR (vse instrukcije iz grupe 4 razen XOR)

F5(11V10) - JSR (vse instrukcije iz grupe 5 razen JSR)

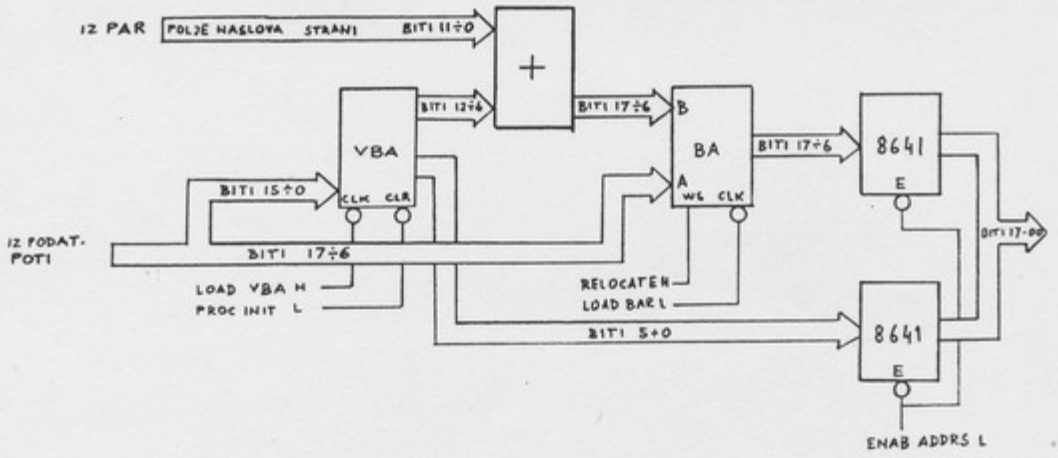
F7*10*9 - MARK, MTPI(D)

4.6 VMESNIK ZA NASLOVNE IN PODATKOVNE SIGNALE PROTI UNIBUS-u

Uporabljena so standardna integrirana vezja 8641, ki signale ojačujejo in posredujejo na vodilo, istočasno pa signale iz vodila sprejemajo in jih posredujejo procesorju. Ta vezja so na listih A1 do A4 (E20, E25, E31 in E34) in to za podatkovne signale D00 - D15, oz. na listu A10 (E18, E1, E6 in E8), kjer se posredujejo adresni signali A00 - A17.

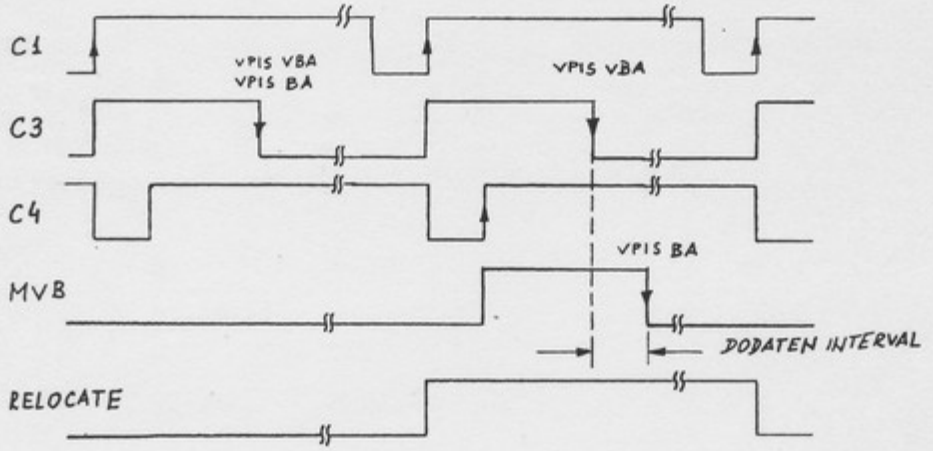
4.6.1 Tvorba naslovnih signalov

Centralno procesna enota DELTA 16/BIT SLICE vsebuje aparaturno opremo za shranjevanje v pomnilniškem prostoru (Memory management). Med izvajanjem prenosov preko UNIBUS vodila, se virtualna adresa, ki jo izračuna ALU enota, pretvori v ustrezno fizično adreso pod pogojem, da je signal A12 RELOCATE H aktiven. V tem primeru se v register BA vpiše adresa, ki je sestavljena na določen nacin (glej prilogo Upravljanje s pomnilniskim prostorom). Blok shemo vezja za tvorbo adres kaže slika 4.28. Potreben je tudi dodatni časovni interval, v katerem se pretvorba izvaja.



Slika 4.28

V primeru, da signal RELOCATE H ni aktiven, se v register BA vpiše neposredno virtualna adresa. Pretvorba ni potrebna, zato je časovni interval, ki je potreben za tak prenos, lahko ustrezno krajši. Osnovna časovna relacija med posameznimi dogodki je skicirana na sliki 4.29



Slika 4.29

4.6.2 Interno dekodiranje naslovov

Tista polovica sprejemno-oddajnih vezij 8641, ki signale sprejema, neprestano opazuje adresne signale na UNIBUS vodilu. Če je procesor v stanju RUN, kar pomeni, da HALT RQST L oz. BUS SACK L nista aktivna, se ti adresni signali lahko v internem adresnem dekoderju dekodirajo v ustrezne krmilne signale, ki služijo pri prenosih podatkov v ali iz PSW registra oz. registrov enote za upravljanje s pomnilnikom. V stanju RUN procesor ne dovoljuje vpisa v svoje registre za splošno uporabo. V primeru, ko je procesor zaustavljen (BUS SACK L je aktiven), vezje za dekodiranje dovoljuje prenos podatkov med registri za splošno uporabo in drugimi moduli, ki so prisotni na UNIBUS vodilu.

Spisek teh registrov in njihovih adres je dan spodaj.

PSW	777776	R10	777710
R0	777700	R11	777711
R1	777701	R12	777712
R2	777702	R13	777713
R3	777703	R14	777714
R4	777704	R15	777715
R5	777705	R16	777716
R6	777706	R17	777717
R7	777707		

5. KONTROLA PRENOSOV

5.1 SPLOŠNO

Vezje (list B10) nadzira statuse Unibusa, kontrolira kontrolne linije (BBSY, MSYN, C1 in C0), ki jih generira procesor, ter detektira napake na Unibusu (paritetno napako, napako zaradi lihe adrese, napako zaradi odsotnosti SSYN signala in napake enote za upravljanje s pomnilnikom).

5.1.1 Kontrolno vezje

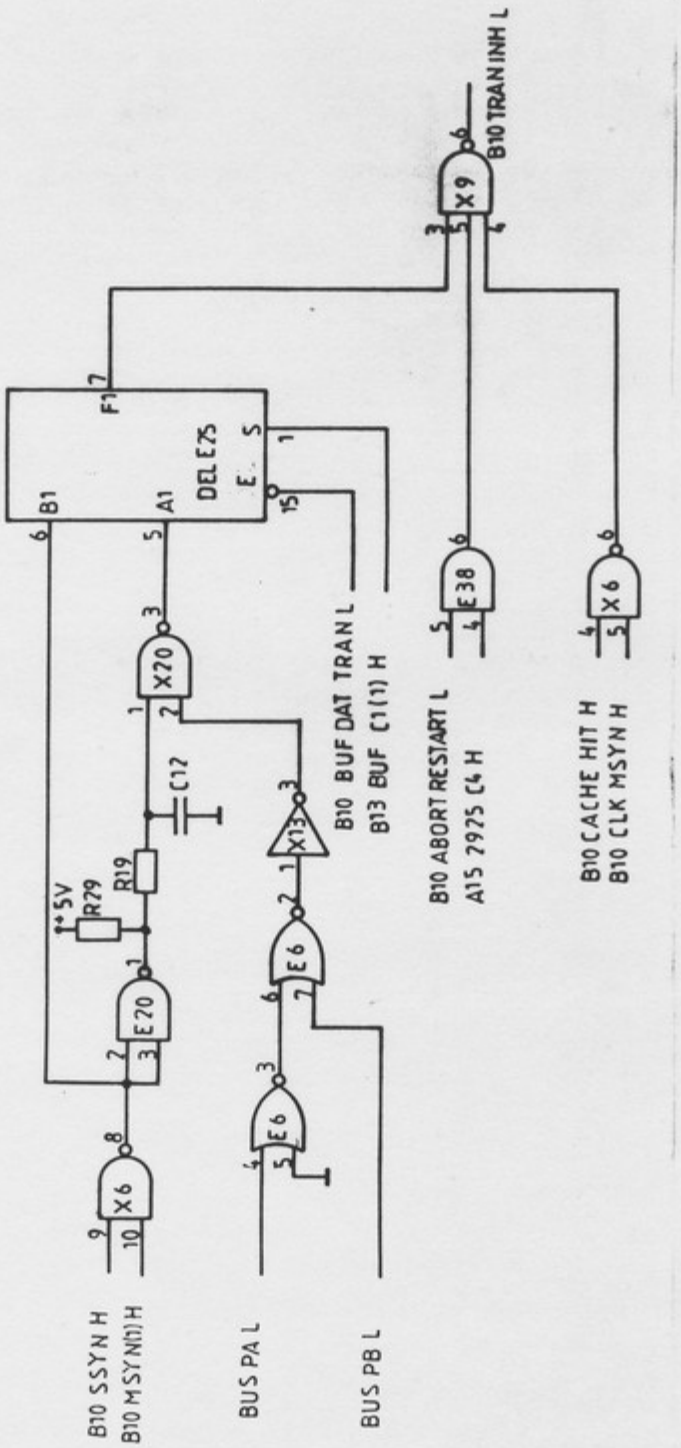
Prenos podatkov se sproži z signalom B7 TRAN (1) H/PL (slika 5.1). Ko gre urin signal A15 2925 C4 H iz stanja logične "0" v stanje logične "1", se kombinira z B10 ABORT RESTART L(običajno "1") in ustavi uro (slika 5.1), dokler se prenos ne konča. Za blokiranje ure pa morata obstajati še dva pogoja:

a) B10 CACHE HIT H ali B10 CLK MSYN H v stanju logične "0".
b) B10 SSYN H v stanju logične "0" in B10 MSYN(0) H v stanju logične "1"

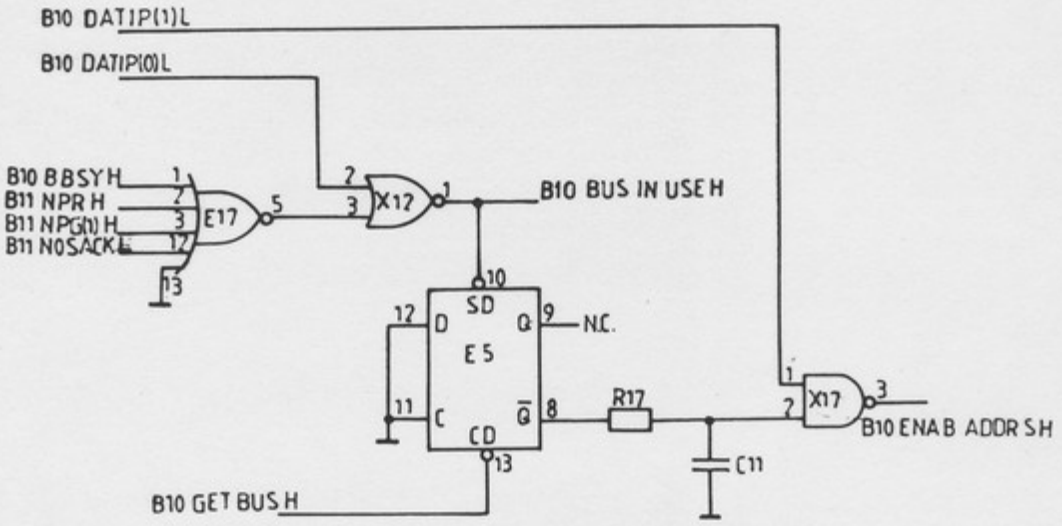
Signal CACHE HIT L je generiran samo v primeru , če imamo v sistemu CACHE pomnilnik in ce je le-ta ugotovil, da vsebuje zahtevani podatek. Deset adresnih linij se vodi na konektor J1, ki se nahaja na plošči A. Ko se aktivira B13 START TRAN H med DATI ciklom, uporabi CACHE pomnilnik sedem adresnih linij (BUS A11-BUS A17), da ugotovi, če ima podatek. Če je le-ta v CACHE-u, se postavita v aktivno stanje CACHE HIT L in TRI STATE AMUX L. Prvi povzroči, da se abortira Unibus prenos(aktivira uro) , drugi pa zapre D-S multipleksor zato, da se podatek lahko sprejme iz CACHE pomnilnika preko SWAP multipleksorja.

5.1.2 Sinhronizacija na Unibus-u

Sinhronizacijska logika na sliki 5.2 arbitrira, kdo bo kontroliral Unibus, procesor ali ena izmed perifernih enot. Logični nivo "1" na set vhodu celice E5 indicira, da je trenutno Unibus zaseden.
Vsak izmed pogojev, ki oblikujejo nivo set vhoda predstavlja stanje določenega Unibus signala.


B10 TRAN INH L
X9
F17
B1
A1
DELETS
E
S
15
1
X20
C12
B10 BUF DAT TRAN L
B13 BUF C1(1) H
X13
+5V
R79
R19
E20
E6
E6
X6
E38
X6
B10 ABORT RESTART L
A15 7925 C4 H
B10 CACHE HIT H
B10 CLK MSYN H
B10 SSYN H
B10 MSYN(1) H
BUS PA L
BUS PB L

Slika 5.1



Slika 5.2

B10 DATIP(1) L	Prisili procesor, da obdrži na Unibus-u isto adreso in da obdrži kontrolo nad Unibus-om tako dolgo , dokler se ne konča tudi DATO tip prenosa.
B10 DATIP(0) L	Usotovi DATIP tip prenosa(read/modify/write) in prevlada nad vsemi ostalimi posoji. NPR enota je sicer lahko dobila od procesorja NPG H, vendar mora počakati, da procesor sprosti Unibus.
B10 BBSY H	Ena od perifernih enot ima kontrolo nad vodilom
B11 NPR H	Periferna enota je postavila NPR L(želi dobiti kontrolo nad Unibus-om).
B11 NPG(1) H	Obstaja lahko situacija, ko je NPR enota prepoznala NPG H in je umaknila NPR L, ne da bi medtem postavila SACK ali BBSY.
B11 NO SACK L	Možno je, da je na Unibus-u v določenem trenutku prisoten samo SACK L.

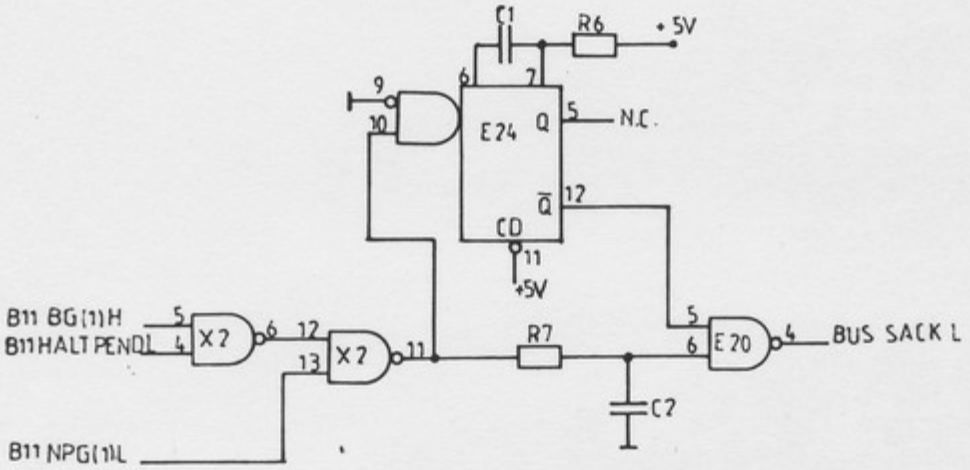
Če ne obstaja nobeden od zadnjih štirih posojev, je lahko celica E5 postavljena s signalom B10 GET BUS H, ki se aktivira , ko se pojavi signal B7 TRAN (1) H/PL. B10 GET BUS H ostane v stanju logične '1', do naslednjega prehoda signala A15 TAP 30 H. Ko je enkrat celica E5 postavljena(/Q='0'), gre v stanje logične '1' B10 ENAB ADDRS H, ki omogoči:

- Postavitev BUS BBSY L signala
- Postavitev podatka na Unibus, če je tip prenosa DATO
- Postavitev adrese , ter kontrolnih linij C0 in C1

C1	C0	tip
0	0	DATI
0	1	DATIP
1	0	DATO
1	1	DATOB

5.1.3 Vezje, ki detektira odsotnost BUS SACK L signala

Vezje na sliki 5.3 ob pojavu signala(NPG (1) L ali BG H) aktivira monostabilni multivibrator. Izhod (/Q) je konjuktivno združen z izhodom vrat X2 in generira BUS SACK L signal, če sa ni postavila periferna enota v času 22 mikrosekund od postavitve enega od GRANT signalov.

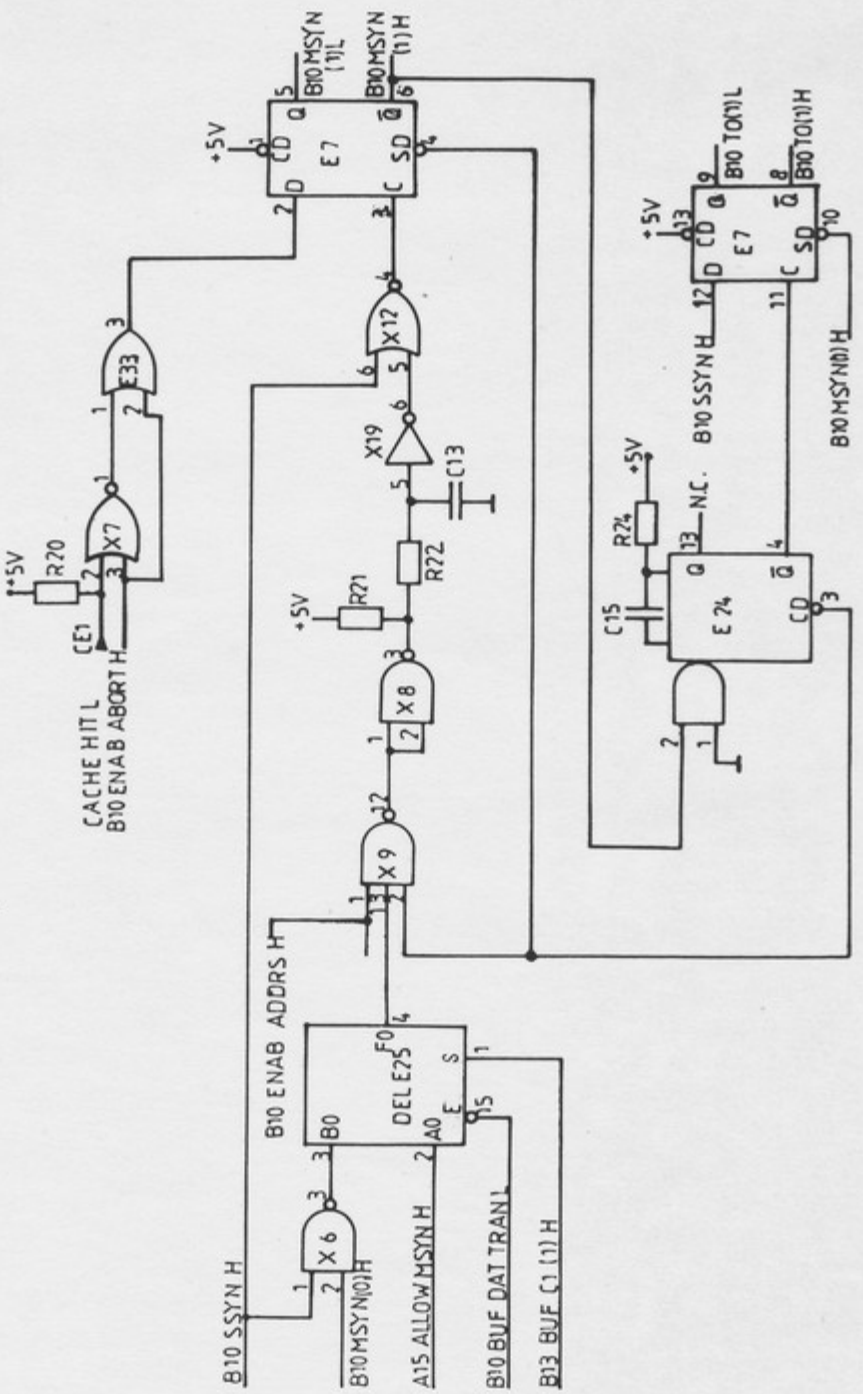


Slika 5.3

Vezje preprečuje procesorju, da bi se "obesil" v primerih, ko je GRANT linija postavljena, BUS SACK L signal pa periferna enota, ki je zahtevala kontrolo nad Unibus-om, ni aktivirala. V primeru, ko enota postavi BUS SACK L, sa to vezje ne bo aktiviralo, ker bo GRANT linija spuščena, preden preteče tako imenovani TIME-OUT monoflopa.

5.1.4 Odsotnost BUS SSYN L signala

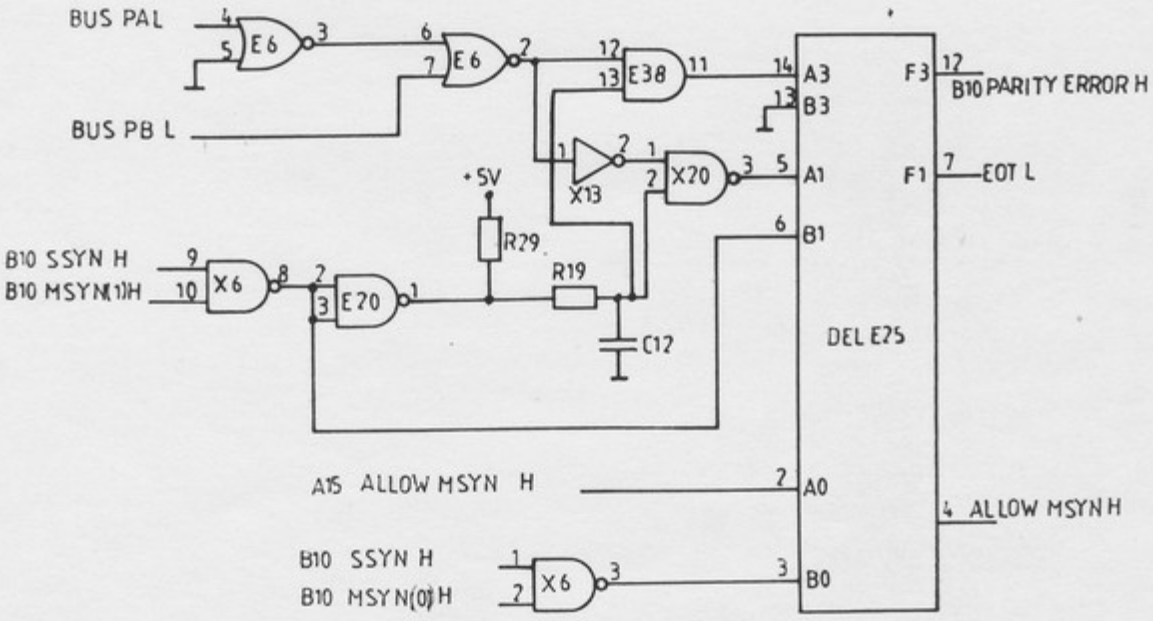
Unibus specifikacija zahteva, da se BUS MSYN L kontrolni signal ne postavi prej kot 150ns zatem, ko so bile nastavljene adresne, podatkovne in kontrolne linije. Da ustrežemo tej zahtevi, je vključeno v Unibus kontrolno logiko tudi vezje na sliki 5.4. Multipleksor E25 (slika 5.5) pomaga, vspostaviti potrebne pogoje glede na tip prenosa po Unibus-u. Izvaja naslednje funkcije:


+5V
R20
CE1
X7
E33
CACHE HIT L
B10 ENAB ABORT H
+5V
E7
B10 MSYN (1) L
B10 MSYN (1) H
X12
X19
C13
+5V
R21
R22
X8
X9
B10 ENAB ADDRS H
B10 SSYN H
X6
B10 MSYN (0) H
DELE25
B0
A0
A15 ALLOW MSYN H
B10 BUF DAT TRAN L
B13 BUF C (1) H
+5V
R24
C15
N.C.
E24
B10 SSYN H
+5V
E7
B10 TO (1) L
B10 TO (1) H
B10 MSYN (0) H

Slika 5.4

a) Onemogoči detekcijo paritetne napake med DATO prenosom
b) Generira EOT L signal, ki označuje konec prenosa
c) Zakasni postavitev BUS MSYN L, z uporabo A15 ALLOW MSYN H signala, ki ni generiran, dokler nismo naložili PBA registra

Funkcija pod točko c) se nanaša samo na DATI in DATIP prenose. Med DATO in DATOB prenosom ni adresa nikoli naložena v PBA register v istem mikroinstrukcijskem ciklu, ko je bil ta prenos zahtevan.
RC vezje omogoči prožnje MSYN celice (E7), približno 150ns zatem, ko je bila na Unibus postavljena adresa in kontrolna signala C0 in C1. Ko je enkrat ta celica postavljena, se aktivira BUS MSYN L in se sproži SSYN monostabilni multivibrator (E24). Ko adresirana periferna enota vrne BUS SSYN L, se celici E7 in E24 brišeta, ko gre B13 START TRAN H signal v logično stanje "0" v naslednji mikroinstrukciji.
Procesorjeva ura se ponovno aktivira, ko dobi B10 TRAN INH L vrednost logične "1". Če se izvaja DATI ali DATIP operacija, se podatek iz Unibus-a lahko vpiše v RAM enote Am 2903 ali v PSW ali pa v IR register.
Ob izvajanju DATO ali DATOB prenosa se podatek umakne z Unibus-a, ko adresirana periferna enota postavi BUS SSYN L signal.



Slika 5.5

5.1.5 Napake na Unibus-u

Ko se E24 monostabilni multivibrator sproži, se mora BUS SSYN L vrniti v 22 mikrosekundah. Če se v tem času ne vrne, E24 postavi celico E7, ki označuje odsotnost SSYN signala. Izhod iz te celice (B10 TO (1) H)potem generira dva signala:

- B10 ABORT RESTART L
- B10 ABORT H

Prvi povzroči ponovno aktiviranje ure, drugi pa postavi celico E30, ki označuje napako na Unibus-u, ter prisili procesor, da dostavi naslednjo mikroinstrukcijo z lokacije nič, kjer se nahaja mikroinstrukcija (SERVICE), ki omogoči logiko za prekinitve in pasti. Če je prisotna past oz. prekinitev se v R15 vpiše vektor. Vektor prekinitve dobi procesor s podatkovnih linij Unibus-a, vektor pasti pa generira logika prikazana na listu B2.

5.1.6 Paritetna napaka

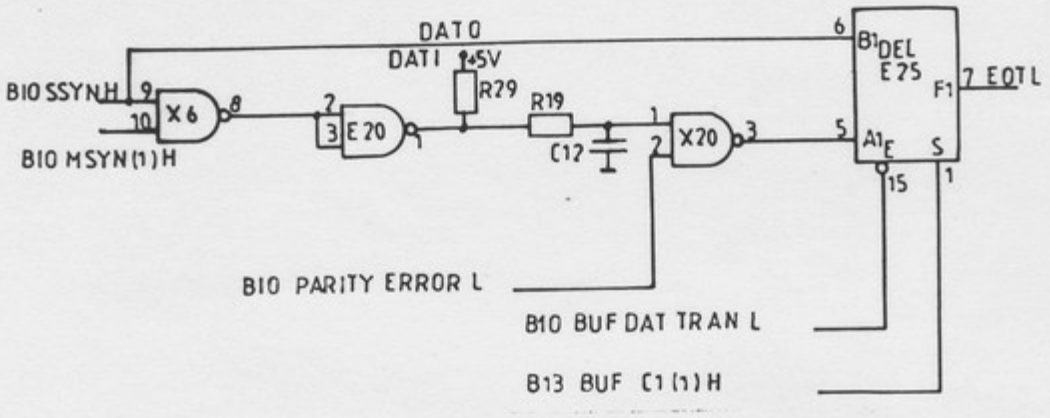
Če se prenos podatkov izvaja s pomnilnikom, ki detektira paritetne napake, se te odražajo tudi v procesorju preko Unibus linij: BUS PA L in BUS PB L

BUS PA	BUS PB	OPIS	PA	PB
1	1	ni napake	0	0
1	0	napaka pri DATI	0	1
0	1	rezervirano	1	0
0	0	rezervirano	1	1

Napaka, ki se detektira med DATI ali DATIP tipom prenosa, povzroči postavitev signala B10 ABORT H (ko procesor sprejme od pomnilnika BUS SSYN L) in celice E12, ki označuje paritetno napako.

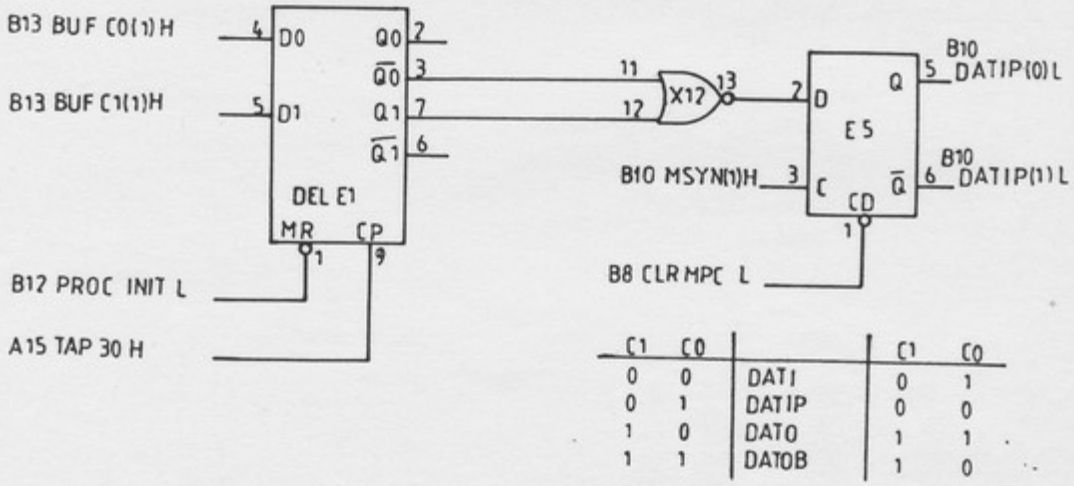
5.1.7 Konec prenosa

Vezje na sliki 5.6 detektira konec prenosa. Med izvajanjem DATI ali DATIP operacije, se generira signal EOT L približno 100ns zatem, ko je procesor sprejel SSYN H. EOT L povzroči ponovno aktiviranje ure.
Med DATO oz. DATOB pa se ura sproži takoj , ko procesor sprejme SSYN H signal.



Slika 5.6

5.1.8 DATIP tip prenosa



C1	C0		C1	C0
0	0	DATI	0	1
0	1	DATIP	0	0
1	0	DATO	1	1
1	1	DATOB	1	0

Slika 5.7

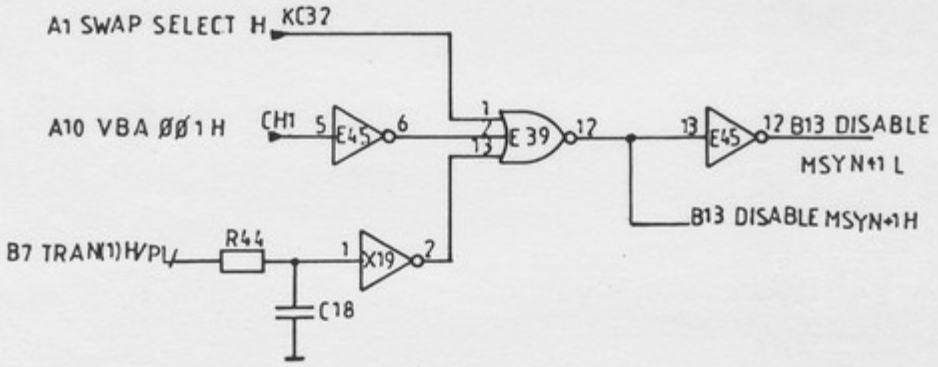
Vezje (slika 5.7) detektira poseben tip prenosa (read/modify/write) in kontrolira BUS BBSY L signal na osnovi B13 BUF C0 (1) H in B13 BUF C1 (1) H. Ko se detektira DATIP prenos, se postavi celica E5. S tem se prisili procesor, da drži BUS BBSY L, dokler se ne konča tudi DATO tip prenosa.

C1	C0	tip	C1	C0
0	0	DATI	0	1
0	1	DATIP	0	0
1	0	DATO	1	1
1	1	DATOB	1	0

5.1.9 Detekcija lihe adrese

Logika za detekcijo lihe adrese je aktivna v naslednjih primerih (slika 5.8):

- če je za dostavo adrese poslana liha adresa
- če je za dostavo ali shranjevanje podatka poslana liha adresa



Slika 5.8

Signal B13 DISABLE MSYN + 1 L je aktiven v primeru, da gre za prenos podatkov (B7 TRAN (1) H na logični "1"), najmanj pomembni bit adrese je ena (A1 VBA 00 (1) H je na logični "1") in da ni zahtevana zamenjava zlogov nad podatki, ki jih dostavljamo iz pomnilnika oz. shranjujemo v pomnilnik (A1 SWAP SELECT H na logični "0"). Signal B13 DISABLE MSYN + 1 L na logični "0" povzroči, da gre procesor v obravnavo pasti zaradi detekcije lihe adrese.

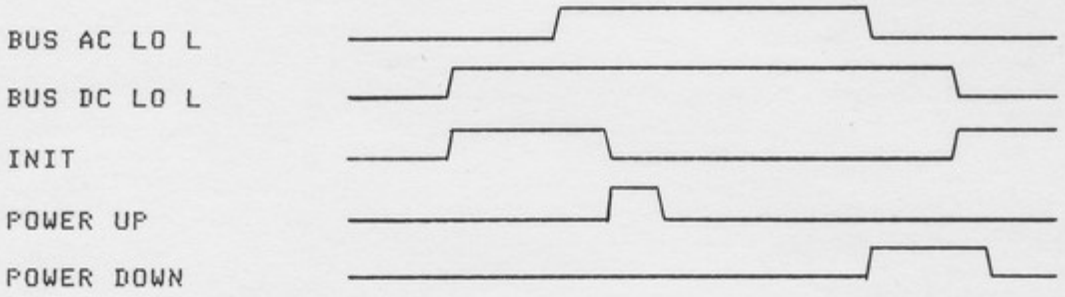
5.2 VEZJE ZA KONTROLO PROCESORJA OB IZPADU IN PONOVNEM VKLOPU NAPETOSTI

5.2.1 Splošno

Vezje služi:

a) Inicializaciji mikroprograma in krmilnega vezja za Unibus
b) Informira mikroprogram, če se pojavi napaka v napajanju
c) Prepreči procesorju, da bi se odzval na napako v napajanju 2ms po vključitvi

Delovanje vezja je odvisno od dveh signalov : AC LO in DC LO. Časovni potek na sliki 5.9 kaže, da se BUS DC LO L ne postavi pred BUS AC LO L. Če BUS DC LO L ni postavljen pomeni, da je napajanje na vsaki komponenti sistema primerno za delovanje. Enako velja za BUS AC LO L, če ta ni postavljen pomeni, da je se dovolj energije v kondenzatorjih napajalnika tako, da računalnik lahko deluje se 5ms po eventuelnem izpadu napetosti.



Slika 5.9

Ko odklopimo napajanje , BUS AC LO L opozori procesor, da je napaka v napajanju. Ko nastopi se BUS DC LO L računalniski sistem ne more vec delovati.
T1 je zakasnitev med postavitvijo BUS AC LO L signala in BUS DC LO L. Ta čas mora biti večji od 5ms. To nam omogoča , da lahko relativno hitro vklapljamo in izklapljamo sistem. Vendar je po startu sistema 2ms garantiran čas, ko ne more nastopiti past, ki označuje izpad napetosti, tudi če odklopimo omrežno napetost istočasno z vklopom.
Ko procesor zazna izpad , nastopi tako imenovani 'POWER TRAP', ki povzroči, da se tekoča vsebina PSW registra in registra R7 shrani na sklad, na lokacije, ki jih določa register R6. PSW se potem napolni z vsebino lokacije 26 , R7 pa z vsebino lokacije 24. Procesiranje se nadaljuje glede na nove vrednosti teh dveh registrov.
Ob ponovnem vklopu se v register PSW in R7 tudi vpiše vrednost z lokacije 26 in 24 (če imamo pomnilnik z dodatno baterijo) ali pa z lokacije 173026 in 173024.

5.2.2 Opis delovanja

Izpad in ponoven vklop napetosti kontrolira vezje, ki je prikazano na listu B12.
Monostabilni multivibrator E19 generira impulz dolžine 150ms brž, ko je BUS DC LO L v stanju logične "1". To je dejansko BUS INIT L signal. Po izteku 150ms se sproži naslednji multivibrator E29 (s pogojem, da je BUS DC LO L še vedno v stanju logične "1") in procesor začne izvajati mikroprogramsko rutino POWER UP (njen začetek je na lokaciji 001). E29 tudi generira 2ms impulz. Med njegovim trajanjem je BUS AC LO L ignoriran. Po izteku tega časa, lahko logično stanje "0" signala BUS AC LO L proži monostabilni multivibrator E29, ki označuje izpad napetosti in se vpiše v register E61. Ko vstopi procesor v nov servisni mikroinstrukcijski cikel, prepozna izpad napetosti in aktivira POWER DOWN past.
Če je detektirana z dekodirnim vezjem (list B3) RESET instrukcija in je procesor v Kernel načinu delovanja se sproži monostabilni multivibrator E19 , ki generira BUS INIT L impulz dolžine 100ms. Po preteku tega časa se ponovno sproži ura in procesiranje se nadaljuje.
Če pa je procesor v User načinu delovanja se RESET instrukcija dekodira kot NOP instrukcija.

5.3 ARBITRACIJA

5.3.1 Splošno

Procesor DELTA 16/BIT-SLICE odgovarja na BR in NPR na enak način kot PDP-11 procesorji. Periferna enota zahteva uporabo Unibus-a, ker želi izvesti prenos podatkov ali pa, ker želi prekiniti program, ki se trenutno izvaja.

5.3.2 BR (bus request)

Procesor ima štiri BR linije(BR4, BR5, BR6, BR7). Vsaka predstavlja svoj prioritetni nivo. Enote , ki imajo BR7 imajo najvišjo prioriteto, enote z BR4 pa najnižjo. Če npr. postavita dve enoti BR, ena na BR6 in druga na BR7, bo najprej servisirana enota, ki je na nivoju 7.
Prioriteta procesorja je določena z biti 5, 6 in 7 iz PSW registra. BG signal bo od procesorja dobila samo tista enota, ki ima višjo prioriteto , kot je trenutno vpisana v PSW. Naslednja tabela vsebuje seznam vseh prioritet :

najvišja:	HALT instrukcija
	Liha adresa
	Napaka memory management-a
	Odsotnost BUS SSYN L signala
	Paritetna napaka
	TRAP instrukcija
	TRACE past
	Prekoračitev sklada
	Izpad napetosti
	HALT s konzole
	BR7
	BR6
	BR5
	BR4
najnižja:	Dostava naslednje instrukcije

Ker lahko BRx povzroci prekinitev samo po zaključitvi trenutne instrukcije , je lahko prekinitev servisirana samo po koncu tekoče instrukcije. Naprava, ki zahteva prekinitev programa, mora ob primernem času postaviti vektor na podatkovne linije Unibus-a. Procesor najprej shrani tekočo vrednost PSW in registra R7, nakar naloži PSW in R7 z novimi vrednostmi.

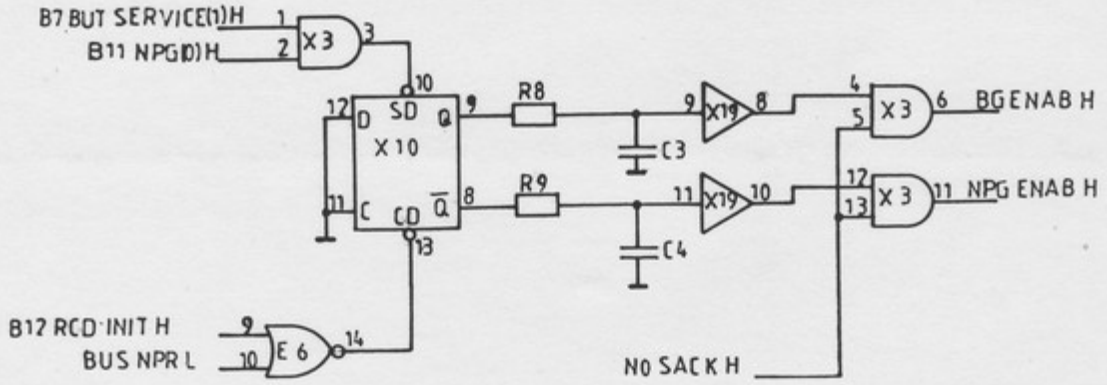
POZOR!

B11 PFAIL BR PEND H se vodi na konektor J2. Ta signal gre v stanje logicne '1', če gre v stanje logične '1' katerakoli BG linija ali , če je postavljen B11 PFAIL(1) H. Procesor s plavajoco vejico uporablja ta signal za abortiranje dolgih instrukcij, tako , da drži čakalni čas za prekinitve pod 20 mikrosekund.

Vse BR zahteve se sprejemajo direktno iz Unibus-a (E8) in se shranjujejo v register E61 na sredini vsake mikroinstrukcije (razen v mikroinstrukciji SERVICE). ROM E31 pregleduje te zahteve in jih primerja s trenutno prioriteto procesorja. Ta ROM prepoznava naslednje prioritete:

```
najvišja:    HLT RQST
             PSW 7
             BR7
             PSW 6
             BR6
             PSW 5
             BR5
             PSW 4
najnižja:    BR4
```

Če je najvišji sprejeti BR takšen , da ima višjo prioriteto kot jo ima procesor, se pripadajoči BG signal(izhod iz ROM-a) postavi v stanje logične "1". Procesorjeva ura se ustavi z B11 BG INH L. Dejanski BG se ne prenese na Unibus dokler ni postavljena celica X5. Signala BG in NPG sta kontrolirana s sinhronizacijsko logiko na sliki 5.10.



Slika 5.10

To vezje odloča ali se bo pojavil BG ali NPG, glede na to, kateri vhod (set ali reset) celice X10 se najprej deaktivira. Signal B7 BUT SERVICE (1) H bo povzročil, da bo celica X5 dobila zahtevan urin signal za vpis. To je potreben pogoj, da se akti-

vira na Unibus-u BG za enoto, ki je zahtevala vodilo z BR signalom. Ko sprejme periferna enota BG odgovori z BUS SACK L signalom. Ko procesor detektira prisotnost BUS SACK L, briše celice X10 in X5. S tem odstrani z Unibus-a BGx oz. NPG in postavi SACK RET celico X5 (urin vhod 11), ki drži uro še naprej v neaktivnem stanju. Ko enota, ki ima trenutno kontrolo nad Unibus-om sprosti vodilo, periferna enota, ki je aktivirala BR postavi BUS BBSY L , postavi vektorsko adreso na podatkovne linije , aktivira BUS INTR L in odstrani BUS SACK L. Procesor izvede deskew SACK signala, briše SACK RET celico X5, ter s tem ponovno aktivira uro, ki vpiše v register R15 ustrezen vektor. BUS INTR L signal forsira mikroprogramsko adreso 005. Na tej lokaciji se nahaja začetek mikroprogramske rutine, ki reagira na ustrezno prekinitev oz. past s tem , da shrani tekočo vrednost registra PSW in registra R7, ter jih sledе na dobljen vektor zamenja z ustreznimi novimi vrenostmi.

5.3.3 NPR(nonprocessor request)

Z NPR zahtevami se dovoljuje enotam na Unibus-u, da komunicirajo ena z drugo, z minimalno udeležbo procesorja. Funkcija procesorja pri servisiranju NPR-jev je v tem, da prepusti kontrolo nad Unibus-om na takšen način, da to ne vpliva na izvajanje instrukcije.
Ko gre reset vhod (pin 13) celice X10 v stanje logične '1', se bo pojavil NPG ENAB H signal in s tem se posredno postavi BUS NPG H, če ni prisoten BUS SACK L. Tako se da možnost DMA enoti, da prevzame kontrolo nad vodilom. Naprava, ki je zahtevala NPR, potem vrne BUS SACK L (ustavi uro), procesor odstrani BUS NPG H, enota pa počaka, da se Unibus sprosti.
Kadar gre za DMA prenos ene same besede, NPR enota umakne SACK signal (procesor lahko nadaljuje izvajanje mikroprograma) takoj, ko prevzame kontrolo nad vodilom.

5.3.4 HALT zahteva

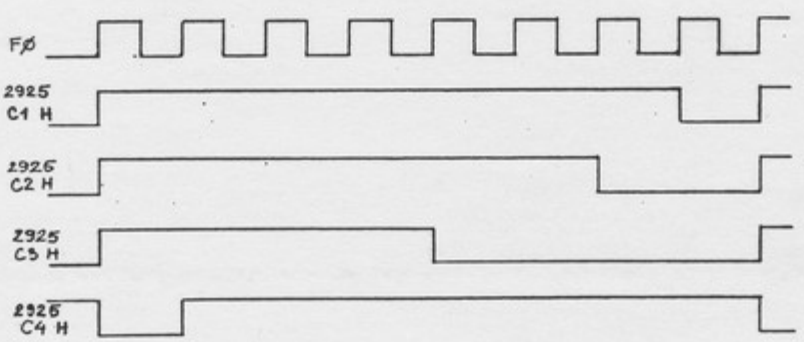
Procesor ima dejansko še dodaten prioritetni nivo, katerega stanje je odvisno od HALT/CONTINUE tipke na konzoli. Ko je detektiran HLT RQST, ga procesor prepozna, kot zahtevo za prekinitev, ko vstopi v servisno mikroinstrukcijo . Procesor ustavi uro in pošlje konzoli HLT GRANT H signal, ki povzroči, da le-ta odstrani HLT RQST, postavi BUS SACK L in BUS BBSY L, ko je vodilo prosto. Uporabnik lahko obdrži procesor v tem stanju poljubno dolgo. Pritisk na tipko CNTR-CONT povzroči, da procesor nadaljuje, kot da se ni ničesar zgodilo. Če pa je procesor že v HALT stanju, bo pritisk na HALT tipko povzročil eksekucijo ene makro instrukcije. BUS SACK L in procesor nadaljuje, kot da se ni ničesar zgodilo.

5.4 URA PROCESORJA

Vezje za generiranje vseh osnovnih urinih signalov centralno procesne enote DELTA 16/Bit-Slice je prikazano na listu A15. Osnovni gradnik urinega sistema je urin generator Am2925 (Y82 na A15), ki omogoča naslednje funkcije:

- mikroprogramsko izbiranje dolžine mikroinstrukcijskega cikla
- generira štiri različne urine signale C1, C2, C3 in C4
- lahko deluje v enem izmed osnovnih načinov: RUN, HALT, SINGLE-STEP in WAIT

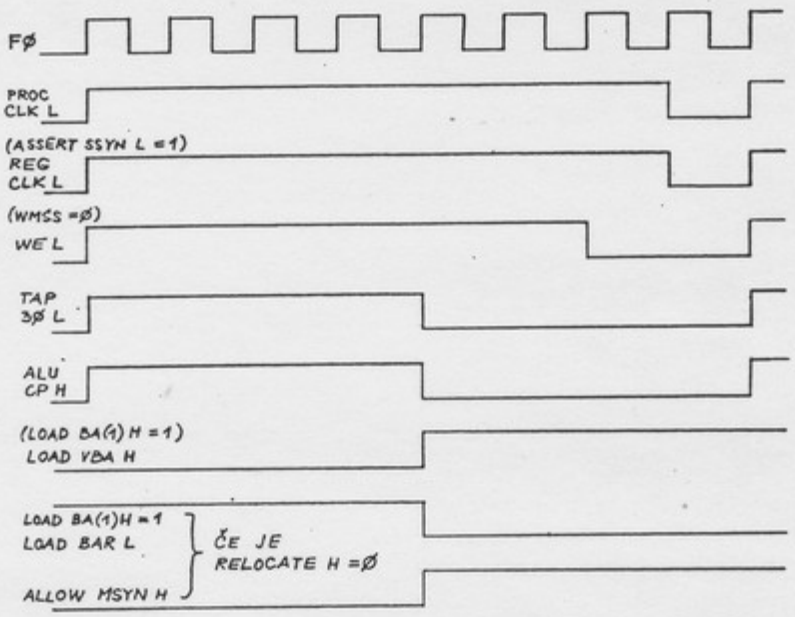
Osnovna perioda F0 urinega generatorja je določena s frekvenco nihanja oscilatorja, ki mu na X1 in X2 vhode priključimo zunanji kristal ali TTL frekvenčni izvor. Dolžina mikroinstrukcijskega cikla je mnogokratnik osnovne periode F0. Možno je izbiranje različnih dolžin(3-10 osnovnih period) cikla glede na krmilno informacijo na L1, L2, in L3 vhodih (krmilni signali iz mikroprogramskega pomnilnika B7 CLOCK L1 H, B7 CLOCK L2 H, in B7 CLOCK L3 H). Pri tem je izhodni signal C1 v logični "0" samo zadnjo, C2 zadji dve, C4 pa le prvo osnovno periodo F0 mikroinstrukcijskega cikla. Pri signalu C3 nastopi prehod iz logične "1" v "0" približno na polovici mikroinstrukcijskega cikla (glej sliko 5.11).



Slika 5.11

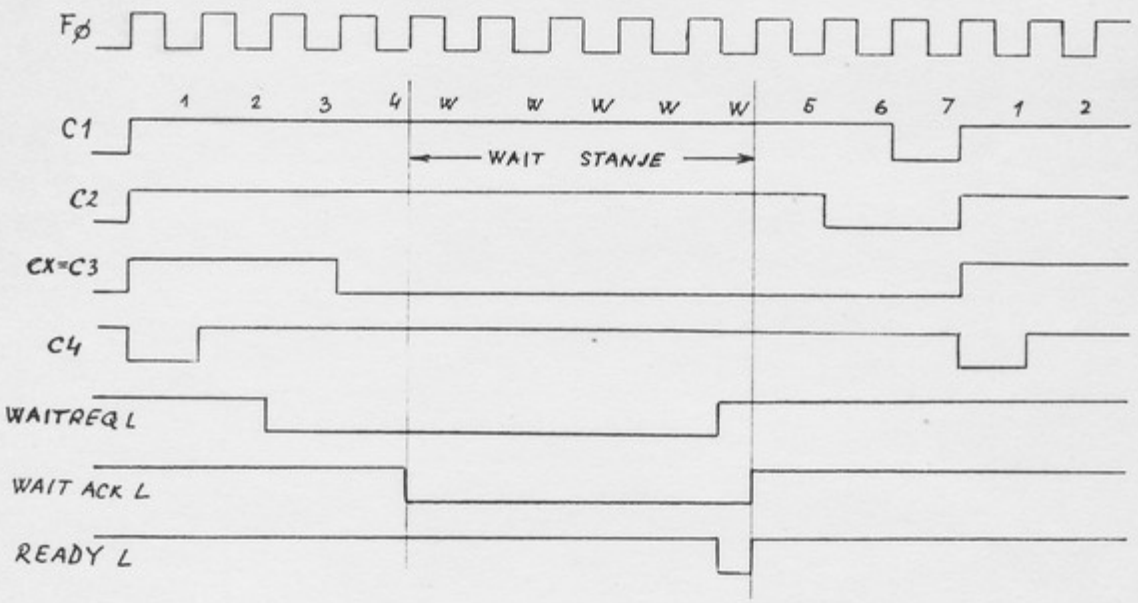
Na osnovi izhodnega signala C1 so izpeljani naslednji urini signali: A15 PROC CLK L, A15 1134 CLK L, A15 REG CLK L, A15 PROC CLK H in A15 REG CLK H. Signal A15 2925 C2 H je osnova za izvedbo signala A9 WE L, ki omogoča vpis operandov z ALU Y vodila v RAM. Signali A15 TAP 30 H, A15 TAP 30 L, A15 LOAD VBA H in A4 ALU CP H so generirani na osnovi izhodnega signala C3. Signal s svojim prehodom v logično enico proži monostabilni multivibrator (E99 na A15), ki v primeru pretvarjanja virtualnega naslova v fizičnega (signal A12 RELOCATE H je "1"), generira signala A15 LOAD BAR L(vpisovanje fizičnega naslova v BA register) in A15 ALLOW MSYN H (pri DATI prenosih sodeluje pri generiranju signala BUS MSYN L). Kadar relokacije ni (A12 RELOCATE H je "0"), se signala A15 LOAD BAR L in A15 ALLOW MSYN H lahko pojavita že ob

prehodu C3 v logicno '0', zato v tem primeru monostabilni multivibrator ni aktiviran. Funkcija multivibratorja je generiranje zakasnjenega urinega signala A15 LOAD BAR L napram signalu A15 LOAD VBA H (upoštevati je treba zakasnitev, ki nastane na seštevalnikih E9, Y5 in Y9 na A10). Časovni potek vseh osnovnih urinih signalov je prikazan na sliki 5.12.



Slika 5.12

Signala A15 PROC CLK L in A15 REG CLK L sovpadata, kadar je signal A14 ASSERT SSYN L na logični '1'. Signal A14 ASSERT SSYN L gre v logično '0', ko gre za vpis v PSW ali v registre PDR, PAR, SR0 preko visokih adres, ki jih prepozna adresni dekoder. V tem primeru se aktivira signal A15 REG CLK L oziroma A15 REG CLK H, kar omogoči vpis v določen register, čeprav je medtem ura v WAIT stanju.
Mehanizem ustavljanja ure se sproži, kadar gre eden izmed signalov B12 PROC INIT L, B11 BG INH L ali B10 TRAN INH L v logično '0'. S tem se aktivira signal B13 WAITREQ L, ki zaustavi uro (ne dovoli nikakršnih sprememb signalov C1, C2, C3 in C4) v začetku periode, ki sledi prehodu signala A15 2925 C3 H v logično '0'.
Ura je v WAIT stanju do nastopa logične '0' na sponki READY urinega generatorja 2925. Po končanem WAIT stanju se realizira še preostali del mikroinstrukcijskega cikla in sicer tako, da je dolžina celotnega cikla enaka mikroprogramsko izbranemu trajanju povečanem za čas WAIT stanja. Časovne razmere pri WAIT stanju so prikazane na sliki 5.13.



Slika 5.13

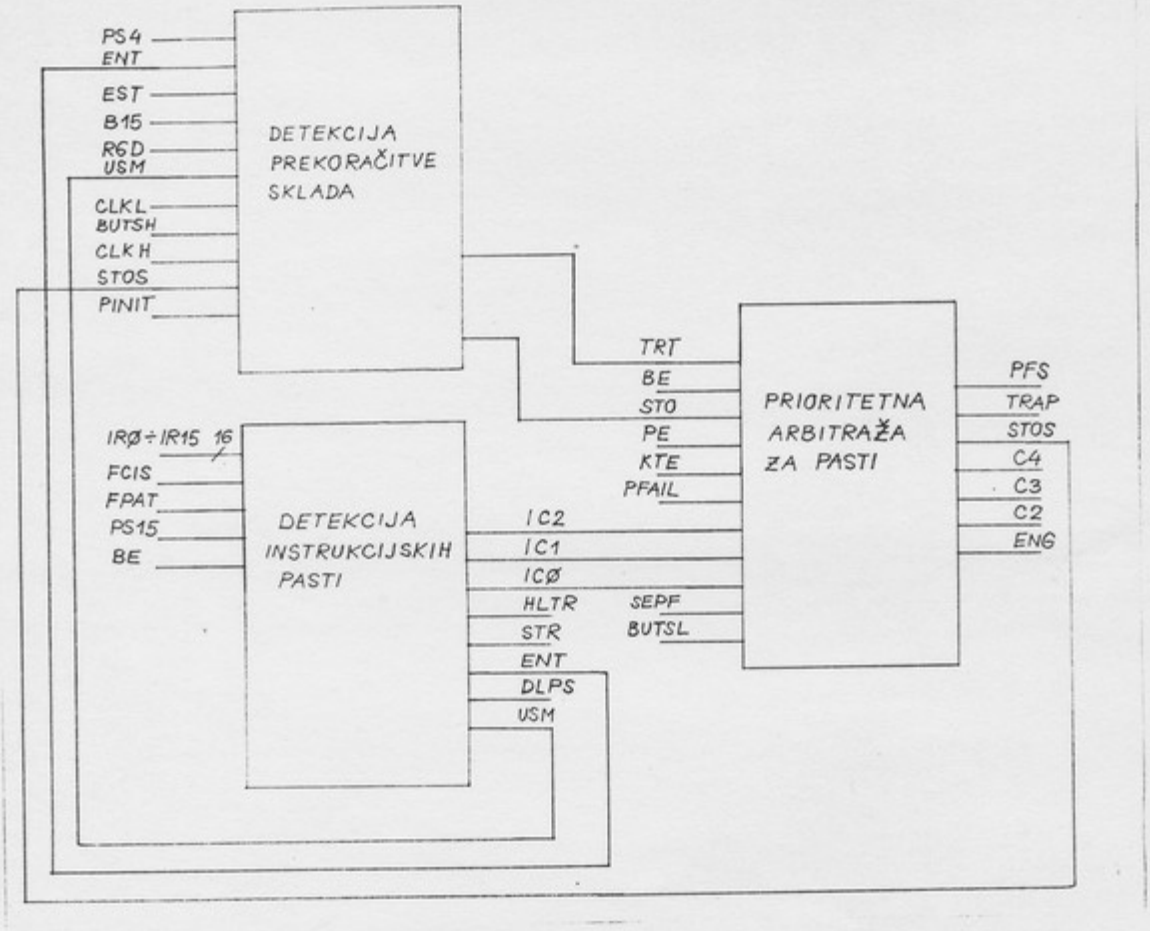
Ura se nahaja v WAIT stanju v naslednjih primerih:

- v INIT periodi pri vklopu/izklopu napajalne napetosti
- pri RESET instrukciji
- ko je aktiviran signal BUS INIT
- kadar je prisoten signal BUS SACK
- ko se izvaja instrukcija HALT(v KERNEL načinu)
- pri prenosu prekinitvenega vektorja z UNIBUS-a v CPU
- pri podatkovnih prenosih CPU - glavni pomnilnik

Urin generator omogoča v načinu SINGLE-STEP izvajanje posameznih mikroinstrukcij, kar uporabimo pri testiranju pravilnosti mikroinstrukcijskih sekvenc. Pri tem mora biti urin generator v HALT stanju.

5.5 PASTI

Ob določenih razmerah v sistemu, bolj natančno tedaj, ko se v instrukcijskem registru nahajajo določene vsebine ali tedaj, ko razna detekcijska vezja v sistemu odkrijejo prisotnost napak ali okvar posameznih enot, preide procesor avtomatsko na izvajanje uporabniško definiranih podprogramov, od katerih vsak predpisuje nadaljnje akcije v zvezi z nastalo situacijo.
Tako delovanje procesorja omogocajo mehanizmi, ki jih imenujemo pasti; implementirani pa so v enoti PASTI, katere osnovno blok shemo in signalno povezavo podaja slika 5.14.



Slika 5.14

Kot je razvidno iz slike 5.14 sestavljajo enoto PASTI 3 funkcijsko zaključene podenote, katerih vsako obravnavamo v naslednjih podpoglavjih. Detajlno logično shemo podaja list B2. Za boljšo preglednost so tako v sliki 5.14, kot tudi v celotni pričujoči dokumentaciji PASTI, uporabljene oznake signalov v skrajšanem zapisu sledе na oznake v shemi z lista B2. Korespondencno legendo teh oznak za vse signale podaja tabela 5.1.

Tabela 5.1

Oznake signalov v shemah	Oznake signalov v dokumentaciji
B4 IR 00 (1) H do	IR0 - IR15
B4 IR 15 (1) H	

B3 GR 8 H	FCIS
FP11A ATTACHED L	FPAT
B2 DMO H	DMO
B2 IR 12 - 14 = 0 L	1214
B2 USER MODE H	USM
B2 OPIS L	OPIN
A5 PSW 15 (1) L	PS15
B10 BE (1) H	BE
IC0, IC1, IC2	IC0, IC1, IC2
B2 HALT RQST L	HLTR
B2 START RESET H	STR
B2 ENAB T BIT H	ENT
B2 DISABLE LOAD PSW H	DLPS
B2 TRACET H	TRT
B2 STACK OVR H	STO
B10 PE (1) H	PE
B10 KTE (1) H	KTE
B11 PFAIL (1) H	PFAIL
FPT SERVICE BR PFAIL L	SEPF
B1 BUT SERVICE L	BUTSL
B2 PFAIL SERV H	PFS
B2 TRAP H	TRAP
B2 STOV SERV H	STOS
B2 C2 H - B2 C4 H	C2 - C4
B2 ENAB GRANTS H	ENG
INTERNI SIGNAL BREZ IMENA	STOV
A5 PSW 04 (1) H	PS4

A9 ESTOV L/PL/	I EST
A3 8 - 15 = 0 L	I 815
A7 R6 DET H	I R6D
A15 PROC CLK L	I CLKL
B7 BUT SERVICE(1) H/PL/	I BUTSH
A15 PROC CLK H	I CLKH
B12 PROC INIT L	I PINIT

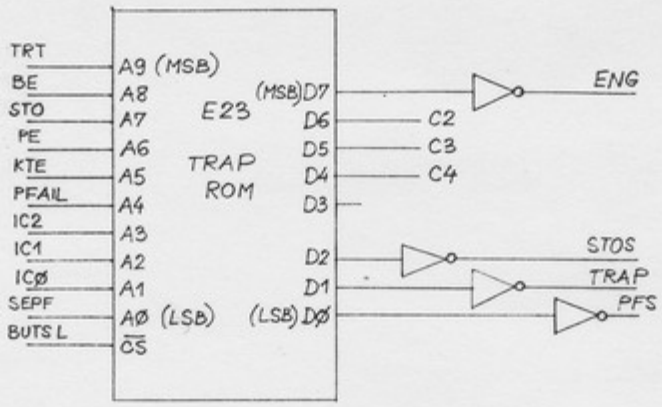
5.5.1 Prioritetna arbitraža za pasti

V mikroinstrukciji SERVICE se obravnavajo vse tedaj prisotne zahteve za prekinitve, instrukcijske pasti in v sistemu odkrite napake in sicer po takem vrstnem redu, kot ga določa prioritetna lista (tabela 5.2).

Tabela 5.2

Najvišja prioriteta	HALT INSTRUCTION
	ODD ADDRESS TIME OUT
	MEMORY MANAGEMENT ERROR
	PARITY ERROR
	INSTRUCTION TRAP
	TRACE TRAP
	STACK OVERFLOW
	POWER FAIL
	HALT FROM CONSOLE
	BUS REQUEST 7
	BUS REQUEST 6
	BUS REQUEST 5
	BUS REQUEST 4
Najnižja prioriteta	NEXT INSTRUCTION FETCH

Dodeljevanje vrstnega reda obdelav po zgornji prioritetni listi za vse, v sistemu možne zahteve za pasti in prekinitve, izvaja dekezijsko vezje v obliki bipolarnega PROM-a kapacitete 1024 x 8 na poziciji E23, ki je prikazano na sliki 5.15.
Vhodni signali IC0, IC1, IC2, ki izvirajo iz podenote DETEKCIJA INSTRUKCIJSKIH PASTI, vnašajo zahteve za vse možne instrukcijske pasti.
Vhodni signal PFAIL se generira v okviru enote BUS CONTROL (list B11) in vnaša zahtevo za past POWER FAIL.
Vhodni signal KTE izhaja iz D-celice(list B10), ki detektira napake enote za upravljanje s pomnilnikom in pomeni zahtevo za past MEMORY MANAGEMENT ERROR.



Slika 5.15

Vhodni signal PE se generira v vezju za detekcijo paritetne napake (list B10) in predstavlja zahtevo za past PARITY ERROR.
Vhodni signal STO generira podenota DETEKCIJA PREKORAČITVE SKLADA (list B2) in vnaša zahtevo za past STACK OVERFLOW.
Vhodni signal BE generira vezje, ki označuje napake na UNIBUS-u (list B10) in pomeni zahtevo za past BUS ERROR.
Vhodni signal TRT izhaja iz podenote DETEKCIJA PREKORAČITVE SKLADA (list B2) in vnaša zahtevo za past TRACE TRAP.
Vhodni signal SEPF vstopa s konektorja J2 in sa aktivira opcijski procesor FLOATING POINT. Ob aktiviranem signalu SEPF se v enoti PASTI obdelujejo le zahteve za prekinitve z vodil (BUS REQUEST) in past POWER FAIL.
Vhodni signal BUTSL vnaša informacijo, kdaj procesor izvaja servisno mikroinstrukcijo.
Rezultat obdelave posamezne zahteve za past je takoimenovana vektorska adresa, ki jo sestavljajo vrednosti izhodnih signalov iz obravnavanja vezja PROM: C2, C3 in C4 ter vhodna signala v to vezje PE in KTE po naslednjem vzorcu:

MSB LSB

| KTE | PE | KTE | C4 | C3 | C2 | 0 | 0 |

Vektorska adresa predstavlja adreso v programskem pomnilniku, s katere procesor pobira novo vsebino programskega števnika ob prehodu iz servisne mikroinstrukcije potem, ko je staro vsebino programskega števnika in PSW registra shranil na sklad. Nova vsebina za PSW register prihaja z adrese, ki sledi vektorski adresi.
Nova vsebina programskega števnika je začetna adresa uporabniškega programa za nadaljne akcije v okviru posamezne pasti.
Vektorske adrese v oktalnem zapisu za posamezne pasti so naslednje:

Tabela 5.3

VEKTORSKA ADRESA	TIP PASTI
004	ODD ADDRESS & TIMEOUT(BE) STACK OVERFLOW, ILLEGAL INSTR.
010	RESERVED INSTRUCION
014	TRACE TRAP, BREAK-POINT TRAP (BPT)
020	INPUT/OUTPUT TRAP (IOT)
024	POWER FAIL
030	EMULATOR TRAP (EMT)
034	TRAP INSTRUCTION
114	MEMORY PARITY ERROR
250	MEMORY MANAGEMENT ERROR

Poleg signalov C2, C3 in C4 vezje PROM na poziciji E23 generira še naslednje signale:

- TRAP, ki je aktiviran ob katerikoli pasti;
- ENG, ki je aktiven, kadar ni nobene zahteve za past in tedaj omogča obravnavo prekinitev (BUS REQUESTS);
- PFS, ki je aktiviran ob pasti POWER FAIL in je pogoj za brisanje zahteve za to past;
- STOS, ki je aktiviran ob pasti STACK OVERFLOW in omogoča odstranitev zahteve za to past.

Prioritetno razvrščanje zahtev za pasti in prekinitve, kot tudi rezultate obdelave, podaja naslednja pregledna tabela 5.4:

Tabela 5.4

1 TIP PASTI (TRAP)	2 VREDNOST ADRESNIH SPREMENLJIVK PROMA-a E23	3 PRIPADAJOČE ADRESNE LOKACIJE PROM-a E23 (HEX)	4 VSEBINA V PROM-u E23(HEX)
HALT INSTRUCTION	TRT=X NIŽJA PRIORITETA BE =X - " - STO=X - " - PE =X - " - KTE=X - " - PFAIL=X - " - IC2=0 IC1=0 IC0=0 SEPF=1	0X1,1X1,2X1,3X1 X=0,1,2,3,4,5,6,7 8,9,A,B,C,D,E,F	07
BUS ERRORS (ODD ADDRESS & TIMEOUT)	TRT=X NIŽJA PRIORITETA BE =1 - " - STO=X - " - PE =X - " - KTE=X - " - PFAIL=X - " - IC2=X RAZEN KOMBINACIJE: IC1=X 000 IC0=X NIŽJA PRIORITETA SEPF=1	1X3,1X5,1X7,1X9, 1XB, 1XD, 1XF 3X3,3X5,3X7,3X9, 3XB, 3XD, 3XF X=0,1,2,3,4,5,6,7 8,9,A,B,C,D,E,F	C5
MEMORY MANAGEMENT ERRORS	TRT=X NIŽJA PRIORITETA BE =0 VIŠJA PRIORITETA STO=X NIŽJA PE =X PRIORITETA KTE=1 PFAIL=X NIŽJA PRIORITETA IC2=X RAZEN KOMBINACIJE: IC1=X 000 IC0=X NIŽJA PRIORITETA SEPF=1	0X3,0X5,0X7,0X9, 0XB, 0XD, 0XF 2X3,2X5,2X7,2X9, 2XB, 2XD, 2XF X=2,3,6,7,A,B,E,F	A5
PARITY ERRORS	TRT=X NIŽJA PRIORITETA BE =0 VIŠJA PRIORITETA STO=X NIŽJA PRIORITETA PE =1 KTE=0 VIŠJA PRIORITETA IC2=X RAZEN KOMBINACIJE: IC1=X 000 IC0=X NIŽJA PRIORITETA SEPF=1	0X3,0X5,0X7,0X9, 0XB, 0XD, 0XF 2X3,2X5,2X7,2X9, 2XB, 2XD, 2XF X=4, 5, C, D	E5

Tabela 5.4 (nadaljevanje)

1 TIP PASTI (TRAP)	2 VREDNOST ADRESNIH SPREMENLJIVK PROM-a E23	3 PRIPADAJOČE ADRESNE LOKACIJE PROM-a E23(HEX)	4 VSEBINA V PROM-u E23(HEX)
ILLEGAL INSTRUCTION	TRT=X NIŽJA PRIORITETA BE =0 VIŠJA PRIORITETA STO=X NIŽJA PRIORITETA PE =0 VIŠJA KTE=0 PRIORITETA PFAIL=X NIŽJA PRIORITETA IC2=1 IC1=0 IC0=1 SEPF=1	0XB, 2XB X=0, 1, 8, 9	C5
RESERVED INSTRUCTION	TRT=X NIŽJA PRIORITETA BE =0 VIŠJA PRIORITETA STO=X NIŽJA PRIORITETA PE =0 VIŠJA KTE=0 PRIORITETA PFAIL=X NIŽJA PRIORITETA IC2=1 IC1=0 IC0=0 SEPF=1	0XD, 2XD X=0, 1, 8, 9	A5
BPT INSTRUCTION	TRT=X NIŽJA PRIORITETA BE =0 VIŠJA PRIORITETA STO=X NIŽJA PRIORITETA PE =0 VIŠJA KTE=0 PRIORITETA PFAIL=X NIŽJA PRIORITETA IC2=1 IC1=0 IC0=0 SEPF=1	0X9, 2X9 X=0, 1, 8, 9	E5
IOT INSTRUCTION	TRT=X NIŽJA PRIORITETA BE =0 VIŠJA PRIORITETA STO=X NIŽJA PRIORITETA PE =0 VIŠJA KTE=0 PRIORITETA PFAIL=X NIŽJA PRIORITETA IC2=0 IC1=1 IC0=1 SEPF=1	0X7, 2X7 X=0, 1, 8, 9	95

Tabela 5.4 (nadaljevanje)

1 TIP PASTI (TRAP)	2 VREDNOST ADRESNIH SPREMENLJIVK PROM-a E23		3 PRIPADAJOČE ADRESNE LOKACIJE PROM-a E23(HEX)	4 VSEBINA V PROM-u E23(HEX)
EMT INSTRUCTION	TRT=X BE =0 STO=X PE =0 KTE=0 PFAIL=X IC2=0 IC1=1 IC0=0 SEPF=1	NIŽJA PRIORITETA VIŠJA PRIORITETA NIŽJA PRIORITETA VIŠJA PRIORITETA NIŽJA PRIORITETA	0X5, 2X5 X=0, 1, 8, 9	B5
TRAP INSTRUCTION	TRT=X BE =0 STO=0 PE =0 KTE=0 PFAIL=X IC2=0 IC1=0 IC0=1 SEPF=1	NIŽJA PRIORITETA VIŠJA PRIORITETA NIŽJA PRIORITETA VIŠJA PRIORITETA NIŽJA PRIORITETA	0X3, 2X3 X=0, 1, 8, 9	F5
TRACE TRAP (T-BIT TRAP)	TRT=1 BE =0 STO=X PE =0 KTE=0 IC2=1 IC1=1 IC0=1 SEPF=1	VIŠJA PRIORITETA NIŽJA PRIORITETA VIŠJA PRIORITETA VIŠJA PRIORITETA - " - - " -	2XF X=0, 1, 8, 9	E5
STACK OVERFLOW	TRT=0 BE =0 STO=1 PE =0 KTE=0 PFAIL=X IC2=1 IC1=1 IC0=1 SEPF=1	VIŠJA PRIORITETA VIŠJA PRIORITETA VIŠJA PRIORITETA NIŽJA PRIORITETA VIŠJA PRIORITETA - " -	0XF X=8, 9	C1

Tabela 5.4 (nadaljevanje)

1 TIP PASTI (TRAP)	2 VREDNOST ADRESNIH SPREMENLJIVK PROM-a E23	3 PRIPADAJOČE ADRESNE LOKACIJE PROM-a E23(HEX)	4 VSEBINA V PROM-u E23(HEX)
POWER FAIL (BREZ OPCIJSKE ENOTE FP 11-A)	TRT=0 VIŠJA BE =0 PRIORITETA STO=0 - " - PE =0 - " - KTE=0 - " - PFAIL=1 IC2=1 - " - IC1=1 - " - IC0=1 - " - SEPF=1 - " -	01F	D4
POWER FAIL (Z OPCIJSKO ENOTO FP 11-A)	TRT=X * GLEJ OPOMBO BE =X - " - STO=X - " - PE =X - " - KTE=X - " - PFAIL=1 IC2=1 (INSTRUKCIJSKE IC1=1 PASTI NISO IC0=1 AKTIVIRANE) SEPF=0	0XE,1XE,2XE,3XE X=1,3,5,7,9,B,D,F	D4
BUS REQUESTS (INTERRUPTS) (Z OPCIJSKO ENOTO FP 11-A)	TRT=X * GLEJ OPOMBO BE =X - " - STO=X - " - PE =X - " - KTE=X - " - PFAIL=0 IC2=1 (INSTRUKCIJSKE IC1=1 PASTI NISO IC0=1 AKTIVIRANE) SEPF=0	0XE,1XE,2XE,3XE X=0,2,4,6,8,A,C,E	07
BUS REQUESTS (INTERRUPTS) (Z OPCIJSKO ENOTO FP 11-A)	TRT=0 VIŠJA PRIORITETA BE =0 - " - STO=0 - " - PE =0 - " - KTE=0 - " - PFAIL=0 - " - IC2=1 - " - IC1=1 - " - IC0=1 - " - SEPF=1	00F	07

* OPOMBA

5.5.2 Detekcija instrukcijskih pasti

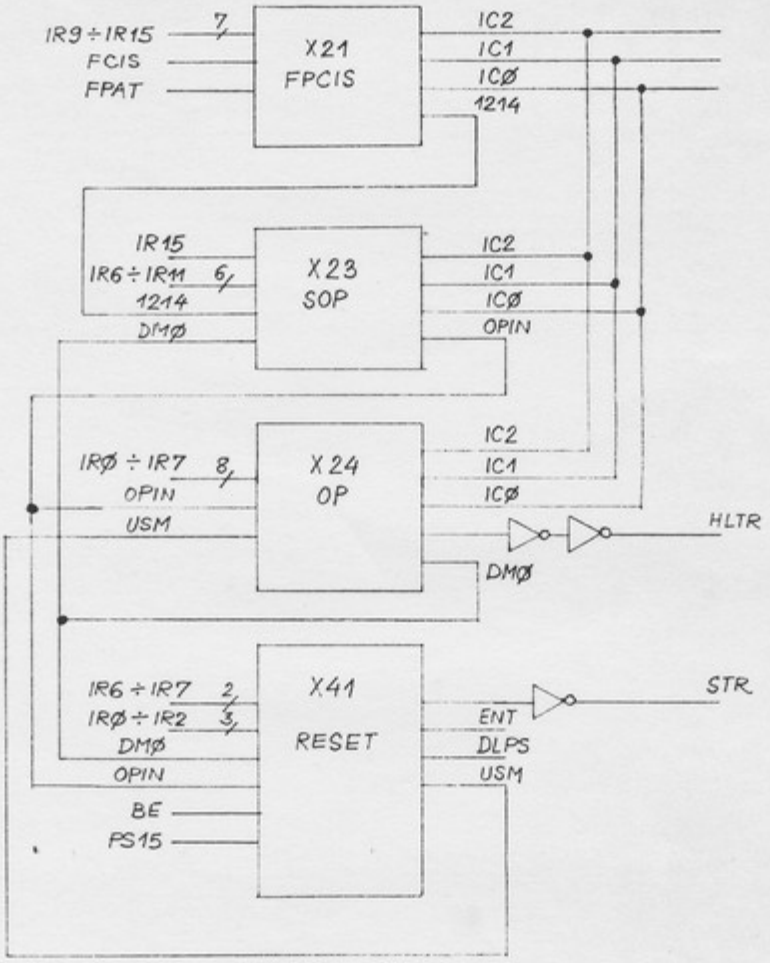
Podenota DETEKCIJA INSTRUKCIJSKIH PASTI izvaja dekodiranje vsebine instrukcijskega registra (signali IR0-IR15) in s pomočjo nekaterih drugih zunanjih signalov(FCIS, FPAT) generira signale IC0 - IC2, katerih posamezne kodne kombinacije predstavljajo zahtevo podenoti PRIORITETNA ARBITRAŽA PASTI za obdelavo različnih instrukcijskih pasti. Zahtevo po obravnavi instrukcijskih pasti povzročajo:

1. nekatere instrukcije, katerih osnovna funkcija je prehod na specifično past (instrukcije: BPT, IOT, EMT, TRAP);
2. ilegalne povezave nekaterih funkcij z določenimi adresnimi načini (instrukcija JMP v adresnem načinu MODE 0, instrukcija JSR v adresnem načinu MODE 0);
3. vse kodne kombinacije v instrukcijskem registru, ki ne predstavljajo veljavnih instrukcij iz standardnega instrukcijskega nabora osnovnega procesorja; te neuporabljene kodne kombinacije so zajete v naslednjih blokih(oktalno):

000	007	-	000 077
000	210	-	000 237
007	000	-	007 777
075	000	-	075 777
076	000	-	076 777
107	000	-	107 777
170	000	-	177 777

Katerakoli od teh kombinacij v instrukcijskem registru ima za posledico past tipa RESERVED INSTRUCTION, če sistem deluje s procesorjem brez opcij CIS(COMMERCIAL INSTRUCTION SET) in FLOATING'POINT. Prisotnost opcije CIS v sistemu izvzame iz obravnave pasti kombinacije 076 000 - 076 777; prisnost opcije FLOATING POINT pa kombinacije 170 000 - - 177 777.

Poleg tega podenota DETEKCIJA INSTRUKCIJSKIH PASTI na osnovi dekodiranja vsebine instrukcijskega registra in s pomočjo nekaterih drugih zunanjih signalov (P15, BE) generira signal HLTR za izvajanje ustavitve prekinitve (ob instrukciji HALT) ter nekatere druge krmilne signale (STR, ENT, DLPS), ki so odvisni od ustreznih instrukcij (RESET, RTT, RTI) in načina delovanja procesorja (USER/KERNEL MODE). Nadrobnejša razlaga delovanja je podana pri opisu posameznih vezij, ki sestavljajo to podenoto. Signalno povezovalno shemo podaja slika 5.16.



Slika 5.16

Podenoto DETEKCIJA INSTRUKCIJSKIH PASTI sestavljajo 4 vezja PAL 16L8. Zaradi omejene kapacitete vhodnih spremenljivk pri teh vezjih, je dekodiranje 16-bitnih vsebin instrukcijskega registra porazdeljeno na vsa 4 vezja PAL (X21, X23, X24, X41 na B2) s tem, da se v posameznih od njih generirajo pomožni signali (pomožne funkcije: 1214, OPIN, DM0), ki zgoščeno predstavljajo vrednosti posameznih segmentov instrukcijskega registra. Iz istega razloga se signali IC0 - IC2 generirajo v treh vezjih (X21, X23, X24 na B2). Izhodi iz vseh teh treh vezij so za posamezen signal vezani skupaj po načelu vodila, kar omogoča tehnološka izvedba vezja PAL 16L8 (3-stanjski izhodi). V primeru signalov IC0 - IC2 so odprti izhodi le iz enega od treh vezij, vendar nikoli iz večih.

Kodne kombinacije signalov IC0 - IC2 za posamezne instrukcijske pasti podaja tabela 5.5 :

Tabela 5.5

IC2	IC1	IC0	TIP INSTRUKCIJSKE PASTI
0	0	0	HALT INSTRUCTION
0	0	1	TRAP
0	1	1	EMT
0	1	1	IOT
1	0	0	BPT
1	0	1	ILLEGAL INSTRUCTION
1	1	0	RESERVED INSTRUCTION
1	1	1	NI INSTRUKCIJSKIH PASTI

5.5.3 Funkcije vezij PAL

5.5.3.1 Vezje FPCIS

Vezje PAL z oznako FPCIS, na poziciji X21, obdeluje naslednje kodne kombinacije vsebine instrukcijskega registra:

075 000 , 076 000 , 170 000
075 777 , 076 777 , 177 777

Za blok kombinacij 075 000 - 075 777 generira signale IC0 - IC2 tako, da predstavlja kodo za past RESERVED INSTRUCTION. Enako tudi za bloka kod 076 000 - 076 777, če ni aktiviran vhodni signal FCIS, oziroma 170 000 - 177 777, če ni aktiviran vhodni signal FPAT. Signal FCIS prihaja iz enote DEKODIRANJE INSTRUKCIJ (list B3) in je aktiviran za vse tiste in samo tiste kode iz bloka 076 000 - 076 777, ki so veljavne instrukcije iz nabora CIS.
Signal FPAT s konektorske sponke J2-C je aktiviran le takrat, kadar je v sistemu instaliran opcijski procesor FLOATING POINT.
Pri aktiviranih signalih FCIS in FPAT se bloka kod 076 000 - 076 777 oziroma 170 000 - 177 777 ne obravnavata kot pogoj za past.
V nadaljevanju podajamo pravilnostno tabelo vezja FPCIS (tabela 5.6).

Tabela 5.6

IR15	IR14	IR13	IR12	IR11	IR10	IR9	FCIS	FPAT	IC2	IC1	IC0	1214	KOMENTAR
1	1	1	1	X	X	X	X	0	1	1	1	1	FLOAT. POINT INSTR.
1	1	1	1	X	X	X	X	1	1	1	0	1	RESERVED
0	1	1	1	1	0	1	X	X	1	1	0	1	RESERVED
0	1	1	1	1	1	0	0	X	1	1	0	1	RESERVED
0	1	1	1	1	1	0	1	X	1	1	1	1	CIS
X	0	0	0	X	X	X	X	X	1	1	1	0	IR12-14= =0 L
VSE OSTALE KOMBINACIJE									1	1	1	1	

Iz te pravilnostne tabele slede naslednje logične enačbe za posamezne izhodne signale (ker so vsi izhodi v vezju PAL 16L8 negirani, podajamo negirane vrednosti izhodnih signalov ter enačbe v razviti obliki, kot se vpisujejo v vezje PAL):

$$\overline{IC0} = IR15 * IR14 * IR13 * IR12 * FPAT \vee \overline{IR15} * IR14 * IR13 * IR12 * IR11 * \overline{IR10} * \overline{IR9} \vee \overline{IR15} * IR14 * IR13 * IR12 * IR11 * IR10 * \overline{IR9} * \overline{FCIS}$$

$$\overline{IC1} = \overline{IR15} * IR15 = 0$$

$$\overline{IC2} = \overline{IR15} * IR15 = 0$$

$$\overline{1214} = \overline{IR14} * \overline{IR13} * \overline{IR12}$$

Logični izrazi krmilnih signalov za odpiranje 3-stanjskih izhodov imajo naslednjo obliko:

TSE(IC0, IC1, IC2) = 1214

TSE(1214) = VCC (stalno odprt izhod)

5.5.3.2 Vezje SOP

Vezje PAL z oznako SOP na poziciji X23 obdeluje instrukcije EMT, TRAP, JMP MODE 0 in JSR MODE 0 ter bloka kod 007 000 - 007 777 in 107 000 - 107 777. S signali IC0 - IC2 oblikuje za ta dva bloka kodo za past RESERVED INSTRUCTION, za instrukciji JMP MODE 0 in JSR MODE 0 kodo za past ILLEGAL INSTRUCTION ter ustrezni kodi za pasti, ki ju zahtevata instrukciji EMT in TRAP.

Pravilnostna tabela vezja SOP ima obliko (tabela 5.7).

Tabela 5.7

IR15	IR11	IR10	IR9	IR8	IR7	IR6	DM0	1214	IC2	IC1	IC0	OPIN	ASOP	KOMENTAR
1	1	0	0	0	X	X	X	0	0	1	0	1	1	EMT
X	1	1	1	X	X	X	X	0	1	1	0	1	1	RESERVED
1	1	0	0	1	X	X	X	0	0	0	1	1	1	TRAP
0	0	0	0	0	0	1	1	0	1	0	1	1	1	JMP/
0	1	0	0	X	X	X	1	0	1	0	1	1	1	JSR/ MODE 0
0	0	0	0	0	X	X	X	0	1	1	1	0	1	OPIN
0	0	0	0	0	X	0	X	X	1	1	1	1	0	ASOP
VSE OSTALE KOMBINACIJE									1	1	1	1	1	

Logični izrazi za posamezne izhodne signale so:

$\overline{IC0} = IR15 * IR11 * \overline{IR10} * \overline{IR9} * \overline{IR8} * \overline{1214} \vee IR11 * IR10 * IR9 * \overline{1214}$

$\overline{IC1} = \underline{IR15} * \underline{IR11} * \overline{IR10} * \overline{IR9} * \underline{IR8} * \overline{1214} \vee \overline{IR15} * \overline{IR11} * \underline{IR10} * \overline{IR9} * \overline{IR8} * \overline{IR7} * \underline{IR6} * DM0 * \overline{1214} \vee \overline{IR15} * IR11 * \overline{IR10} * \overline{IR9} * DM0 * \overline{1214}$

$\overline{IC2} = IR15 * IR11 * \overline{IR10} * \overline{IR9} * \overline{1214}$

$\overline{OPIN} = \overline{IR15} * \overline{1214} * \overline{IR11} * \overline{IR10} * \overline{IR9} * \overline{IR8}$

$\overline{ASOP} = \overline{IR15} * \overline{IR11} * \overline{IR10} * \overline{IR9} * \overline{IR8} * \overline{IR6}$

Logični izrazi za krmiljenje 3-stanjskih izhodov imajo obliko:

$TSE(IC0, IC1, IC2) = \overline{1214} * ASOP$

TSE(OPIN)= VCC STALNO ODPRT IZHOD

TSE(ASOP)= VCC STALNO ODPRT IZHOD

Signal ASOP se povratno zaključuje v samem vezju SOP in je pomozna logična funkcija za krmiljenje 3-stanjskih izhodov.

5.5.3.3 Vezje OP

Vezje PAL z oznako OP na poziciji X24 obdeluje instrukcije IOT, BPT in HALT ter bloka kod 000 007 - 000 077 in 000 210 - 000 237. S signali IC0 - IC2 oblikuje za ta dva bloka kodo za past RESERVED INSTRUCTION ter ustrezni kodi za pasti, ki ju zahtevata instrukciji IOT in BPT. Ob prisotnosti instrukcije HALT to vezje aktivira signal za ustavitev ure HLTR ter generira zahtevo za ustavitveno prekinitev (IC0 - IC2=0), če je procesor v načinu delovanja KERNEL MODE. V načinu delovanja procesorja USER MODE pa ob prisotnosti instrukcije HALT, preko signalov IC0 - IC2 zahteva obravnavo pasti RESERVED INSTRUCTION.
Evidenco o načinu delovanja procesorja predstavlja signal USM, ki se oblikuje v vezju PAL z oznako RESET (X41 na B2). Kadar je ta signal aktiviran, deluje procesor v načinu USER MODE, sicer pa v KERNEL MODE.
Pravilnostna tabela za to vezje je:

Tabela 5.8

IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0	USM	OPIN	IC2	IC1	IC0	HLTR	KOMENTAR
0	0	0	0	0	0	0	0	0	0	0	0	0	0	HALT
0	0	0	0	0	0	1	1	X	0	1	0	0	1	BPT
0	0	X	X	X	1	1	1	X	0	1	1	0	1	RESERVED
1	0	0	0	1	X	X	X	X	0	1	1	0	1	RESERVED
1	0	0	1	X	X	X	X	X	0	1	1	0	1	RESERVED
0	0	0	0	0	0	0	0	1	0	1	1	0	1	RESERVED
0	0	0	0	0	1	0	0	X	0	0	1	1	1	IOT
VSE OSTALE KOMBINACIJE										1	1	1	1	

Iz te pravilnostne tabele izhajajoči logični izrazi za izhodne signali so:

$$\overline{IC0} = \overline{IR7} * \overline{IR6} * \overline{IR5} * \overline{IR4} * \overline{IR3} * \overline{IR2} * \overline{IR1} * \overline{IR0} * \overline{OPIN} \vee$$
$$\vee \overline{IR7} * \overline{IR6} * \overline{IR5} * \overline{IR4} * \overline{IR3} * \overline{IR2} * IR1 * IR0 * \overline{OPIN} \vee$$
$$\vee \overline{IR7} * \overline{IR6} * \overline{IR2} * \overline{IR1} * IR0 * \overline{OPIN} \vee$$
$$\vee IR7 * \overline{IR6} * \overline{IR5} * \overline{IR4} * \overline{IR3} * \overline{OPIN} \vee$$
$$\vee IR7 * \overline{IR6} * \overline{IR5} * IR4 * \overline{OPIN}$$

$$\overline{IC1} = \overline{IR7} * \overline{IR6} * \overline{IR5} * \overline{IR4} * \overline{IR3} * \overline{IR2} * \overline{IR1} * \overline{IR0} *$$
$$* \overline{OPIN} * \overline{USM} \vee$$
$$\vee \overline{IR7} * \overline{IR6} * \overline{IR5} * \overline{IR4} * \overline{IR3} * \overline{IR2} * IR1 * IR0 * \overline{OPIN}$$

$\overline{IC2}= \overline{IR7} * \overline{IR6} * \overline{IR5} * \overline{IR4} * \overline{IR3} * \overline{IR2} * \overline{IR1} * \overline{IR0} *$
$* \overline{OPIN} * \overline{USM} \vee$
$\vee \overline{IR7} * \overline{IR6} * \overline{IR5} * \overline{IR4} * \overline{IR3} * \overline{IR2} * \overline{IR1} * \overline{IR0} * \overline{OPIN}$

$\overline{HLTR}= \overline{IR7} * \overline{IR6} * \overline{IR5} * \overline{IR4} * \overline{IR3} * \overline{IR2} * \overline{IR1} * \overline{IR0} *$
$* OPIN * USM \vee$

$\overline{DMO}= IR5 \vee IR4 \vee IR3$

Logični izrazi za krmiljenje 3-stanjskih izhodov so:

$TSE(IC0, IC1, IC2)= \overline{OPIN} * \overline{IR6}$

TSE(HLTR)= VCC STALNO ODPRT IZHOD

TSE(DMO) = VCC STALNO ODPRT IZHOD

5.5.3.4 Vezje RESET

Vezje PAL z oznako RESET, na poziciji X41, obdeluje instrukcije RESET, RTI in RTT. Kadar detektira instrukcijo RESET in procesor deluje v načinu KERNEL MODE, tedaj vezje aktivira signal STR, v načinu delovanja procesorja USER MODE pa obravnava instrukcijo RESET kot NOP, torej ne aktivira signal STR.
Instrukcija RTT onemogoča neposreden vpliv bita T (PS4) za aktiviranje pasti TRACE TRAP. Vezje RESET to zahtevo realizira z deaktiviranjem signala ENT ob dekodiranju instrukcije RTT. V vseh ostalih primerih je signal ENT aktiviran.
Kot zahteva strategija delovanja enote MEMORY MANAGEMENT se lahko v manj pomembne celice registra PSW vpisuje nova vrednost le iz sklada (stacka), ko deluje procesor v načinu USER MODE. Vezje RESET to zahtevo izpolni z aktiviranjem signala DLPS, kadar detektira prisotnost instrukcije RTI ali RTT. te razmere strnjeno podaja spodnja preglednica (tabela 5.9), kjer je aktivnost signala označena z 1:

Tabela 5.9

RESET KERNEL M.	RTI USER M.	RTT USER M.	RTT KERNEL M.	OSTALO	SIGNAL
1	0	0	0	0	STR
1	1	0	0	1	ENT
0	1	1	0	0	DLPS

Vezje PAL RESET izkoristimo tudi za implementacijo logičnega izraza za signal USM (USER MODE).
Pravilnostna tabela tega vezja ima obliko (tabela 5.10):

Tabela 5.10

OPIN	PS15	IR7	IR6	DM0	IR2	IR1	IR0	STR	ENT	DLPS
0	1	0	0	1	1	0	1	1	1	0
0	0	0	0	1	0	1	0	0	1	1
0	0	0	0	1	1	1	0	0	0	1
0	1	0	0	1	1	1	0	0	0	0
VSE OSTALE KOMBINACIJE								0	1	0

Na osnovi te tabele izpeljani logični izrazi za izhodne signale imajo obliko:

$$STR = \overline{OPIN} * PS15 * \overline{IR7} * \overline{IR6} * DM0 * IR2 * \overline{IR1} * IR0$$

$$\overline{ENT} = \overline{OPIN} * \overline{IR7} * \overline{IR6} * DM0 * IR2 * IR1 * \overline{IR0}$$

$$\overline{DLPS} = \overline{OPIN} \vee PS15 \vee \overline{IR1} \vee \overline{DM0} \vee DM0 * IR6 \vee \overline{PS15} * IR7 \vee DM0 * \overline{IR6} * IR0$$

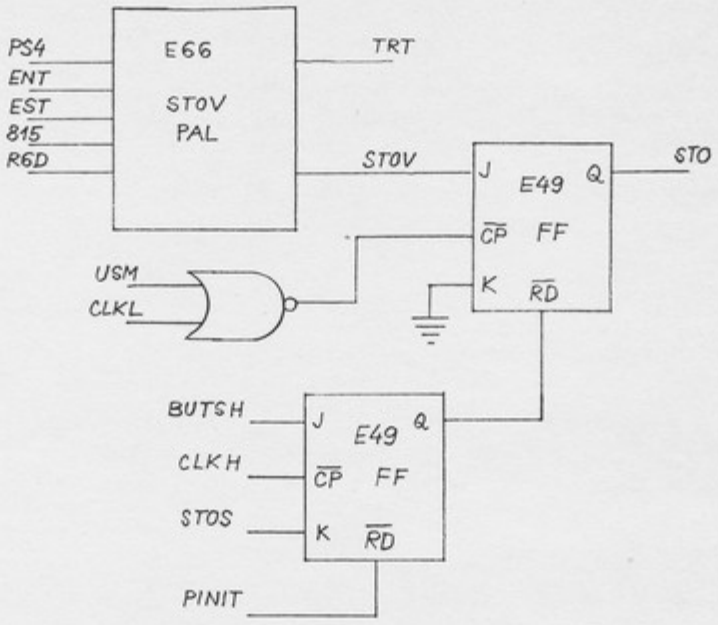
$$\overline{USM} = BE * PS15$$

Vsi 3-stanjski izhodi so krmiljeni tako, da so stalno odprti:

TSE(STR|ENT|DLPS|USM)= VCC

5.5.4 Detektiranje prekoračitve sklada

Vsak poskus procesorja, da zniža vsebino kazalca sklada (register STACK POINTER - R6) pod mejo 400(oktalno), ima za posledico prehod v past prekoračitve sklada (STACK OVERFLOW TRAP). Obdelavo situacije za to past izvaja podenota DETEKCIJA PREKORAČITVE SKLADA, katere signalno povezovalno shemo prikazuje slika 5.17.



Slika 5.17

Podenoto sestavljajo: vezje PAL16L8 z oznako STOV (E66 na B2) in dve vezji flip-flop (E49 na B2) v kaskadni vezavi.
Vezje STOV oblikuje iz signalov EST, 815 in R6D logični pogoj, STOV, za obravnavo pasti STACK OVERFLOW po naslednji logični enačbi:

$\overline{STOV} = \overline{R6D} \vee EST \vee 815$

Časovno determiniranost trajanja zahteve za past STACK OVERFLOW določa vezje dveh flip-flopov, ki daje zahtevo za past v obliki izhodnega signala STO; ta pa je aktiviran za čas trajanja servisne mikroinstrukcije (vhodni signal BUTSH), dokler se ne aktivira signal za brisanje zahteve za to past (vhodni signal STOS), ali ne izteče servisna mikroinstrukcija (vhodni signal BUTSH) ali ne pride do inicializacije procesorja (vhodni signal PINIT).
Vezje PAL z oznako STOV generira iz vhodnih signalov PS4 in ENT tudi signal TRT, ki predstavlja zahtevo za past TRACE TRAP. Logični izraz za ta signal je:

$\overline{TRT} = \overline{PS4} \vee \overline{ENT}$

6. UPRAVLJANJE S POMNILNIŠKIM PROSTOROM

6.1 SPLOSNO

Ta del opisuje MM enoto procesorja. Aparaturna oprema je načrtovana tako, da jo je mogoče uporabiti na vseh sistemih, ki imajo pomnilnik večji od 28K besed in za večuporabniške, večprogramske sisteme, kjer sta potrebna zaščita in relokacija.

6.1.1 Programiranje

Aparaturna oprema MM je bila grajena kot oprema za večprogramske sisteme. Procesor lahko deluje na dva načina: KERNEL in USER. Ko je v Kernel načinu ima program popolno kontrolo in lahko izvaja vse ukaze. Monitorji in nadzorni programi se izvajajo v tem načinu. V User načinu, program ne more izvajati ukazov, ki bi lahko:

a) Povzročili spremembo Kernel programa
b) Ustavili računalnik
c) Uporabili pomnilniški prostor, ki je dodeljen sistemu ali drugemu uporabniku
d) Inicializirali(Reset instrukcija)

V več programskih okoljih se lahko v kateremkoli trenutku v pomnilniku nahaja več uporabniških programov. Naloga nadzornega programa bi bila, da kontrolira izvajanje različnih uporabniških programov, upravlja z dodeljevanjem pomnilnika in perifernih enot ter varuje celovitost sistema s skrbno kontrolo vsakega uporabniškega programa. V multiprogramskih sistemih MM enota na določen način dodeljuje strani uporabniškemu programu (relokacija pomnilniških segmentov) in preprečuje uporabniku nepooblaščen dostop do strani izven področja, ki mu je določeno. Tako učinkovito preprečuje, da bi uporabnik, po nesreči ali namenoma uničil drug uporabniški program ali pa celo sistemske programe.
Aparaturna oprema omogoča operaciskemu sistemu, da na zahtevo dinamično dodeljuje pomnilnik, medtem ko se program izvaja. To je posebno uporabno, ko se izvajajo programi v višjem programskem jeziku, kjer se npr. polja sestavljajo med izvajanjem programa. Prevajalniku ni potrebno skrbeti za potreben prostor v pomnilniku, saj se ta dinamično določa.

6.1.2 Osnovno adresiranje

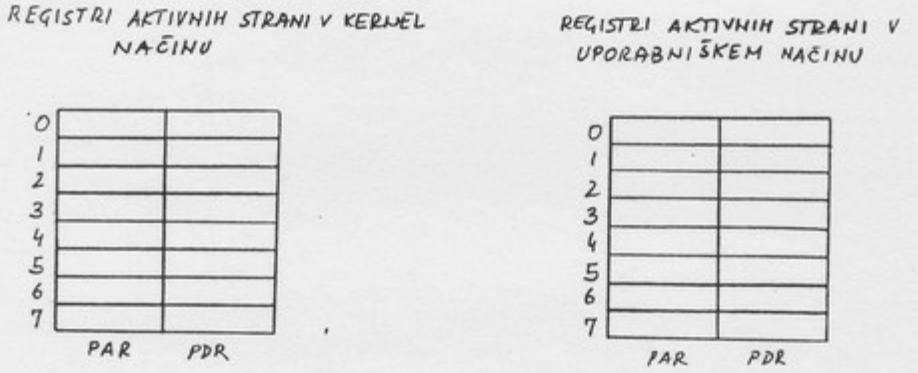
Centralna procesna enota generira 18 bitno adreso, čeprav je dolžina podatkovne besede 16 bitov. To razširitev, ki omogoča naslavljanje 128K pomnilniškega prostora, omogoča aparaturna oprema MM enote. Zgornjih 4K besed adresnega prostora je rezervirano za naslove registrov vhodno/izhodnih enot. V primerih, ko enota MM ni aktivna, se pri naslavljanju visokih adres avtomatično postavi adresna bita 16 in 17 na vrednost "1". Tako se 16 bitno naslavljanje registra z adreso 173224 avtomatično pretvori v polno 18 bitno adreso (773224).

6.1.3 Registri aktivnih strani (APR)

MM enota uporablja dva niza osmih 32-bitnih APR(slika 6.1). APR je dejansko par 16 bitnih registrov:

- Naslovni register strani(PAR)
- Opisni register strani(PDR)

Ta dva registra se vedno uporabljata kot par in vsebujeta vse potrebne podatke za vpis in relokacijo aktivne pomnilniške strani. En niz se uporablja v Kernel in drugi v User načinu. Izbira je določena glede na stanje bitov 14 in 15 v PSW registru.



Slika 6.1

Možnosti, ki jih nudi MM:

- Velikost pomnilnika	:	124K besed + 4K za V/I registre
- Naslovni prostor	:	Virtualni(16 bitov)
		Fizični(18 bitov)
- Načini delovanja	:	Kernel in User
- Kazalci sklada	:	2
- Relokacija	:	St. strani je 16(8 za vsak način)
		Dolžina strani 32 do 4096 besed
- Zaščita pomnilnika	:	Ni dostopa
		Samo branje
		Pisanje/branje

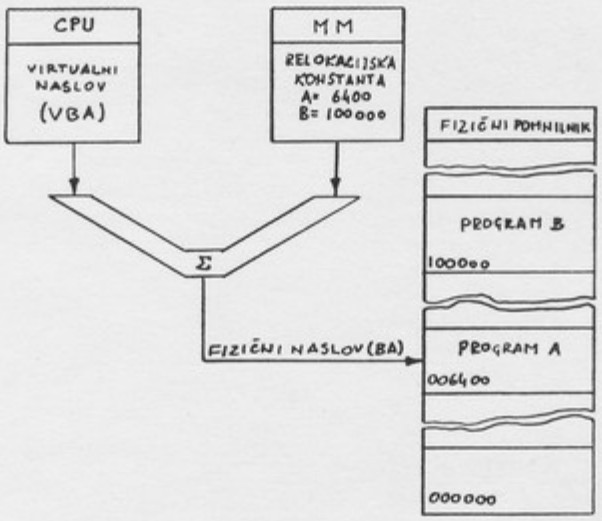
6.2 RELOKACIJA

6.2.1 Virtualno adresiranje

Ko MM enota deluje, se 16 bitni naslov ne interpretira več kot fizični naslov (BA), ampak kot virtualni naslov (VBA), ki se uporabi pri sestavljanju 18 bitne adrese. 18 bitni fizični naslov dobimo tako, da se na ustrezen način združijo virtualni naslov in vsebina PAR registra.
Ker se naslovi avtomatično relocirajo, lahko smatramo, da računalnik dela v virtualnem naslovnem prostoru. Virtualni naslovni prostor je razdeljen na 8 strani po 4K besed. Vsaka stran se relocira ločeno. Stran ima lahko tudi samo 32 besed tako, da kratke procedure ali kratek podatkovni prostor zasede le toliko prostora v pomnilniku, kot je potrebno. To je koristno pri kontrolnih sistemih v realnem času, ki vsebujejo veliko majhnih nalog. Koristno je tudi za kontrolo sklada in kontrolo vmesnih pomnilnikov.

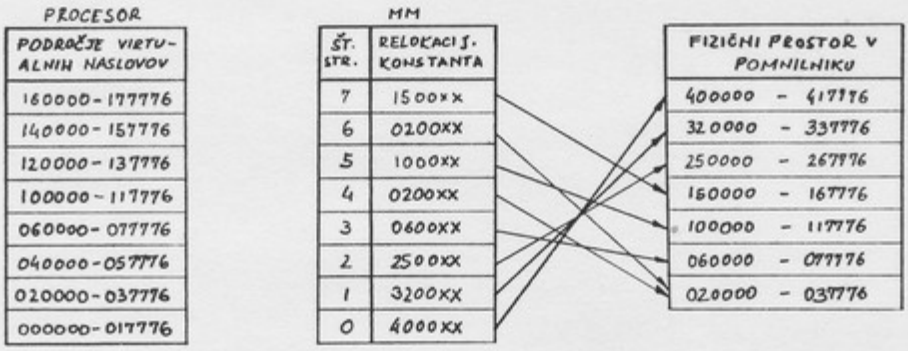
6.2.2 Relokacija programa

APR registri se uporabljajo, da se določi začetni naslov vsakega relociranega programa v pomnilniku. Slika 6.2 kaže poenostavljen primer relokacijskega koncepta(list A10)



Slika 6.2

Konstanta A relocira program A, ki se začne na naslovu 0, na fizični naslov 6400(osmiško). Če je naslednji virtualni naslov 2, bo relokacijska konstanta povzročila, da bo naslednji fizični naslov 6402. Ko se izvaja program B, se relokacijska konstanta spremeni na 100000. Program, ki se začne na virtualnem naslovu 0, je relociran tako, da je njegov fizični naslov 100000.
Program se relocira na strani, ki vsaka lahko vsebuje do 128 blokov. Velikost bloka je 32 besed. Tako je maksimalna velikost strani 4K besed (32*128). Uporaba vseh osmih APR registrov omogoča shranjevanje programov velikih do 32K besed. Vsaka od osmih strani se lahko nahaja kjerkoli v fizičnem pomnilniku, le vsaka relocirana stran se mora začeti na naslovu, ki je mnogokratnik števila 32. Strani, ki je manjša od 4K besed, je dostopen le tisti del pomnilnika, ki je strani dejansko dodeljen. Primer relokacije, ki sa kaže slika 6.3 ilustrira več podrobnosti.



Slika 6.3

Čeprav se zdi, da je program na neprekinjenem naslovnem prostoru, je 32K besed fizičnega naslovnega prostora dejansko razdrobljenih na več ločenih področij fizičnega pomnilnika. Program se lahko nalaga, dokler je preostali fizični pomnilniški prostor še dosti velik. Ni pa potrebno , da bi bil fizični pomnilniški prostor neprekinjen.
Strani se lahko relocirajo v višje ali nižje fizične naslove, glede na njihov virtualni naslov. V primeru na sliki 6.3 je stran 1 relocirana na višje področje fizičnih naslovov, kot npr. stran 4, ki je relocirana nižje, medtem ko stran 3 sploh ni relocirana, čeprav je njena relokacijsa konstanta različna od nič (virtualne in fizične adrese so nespremenjene).
Vse strani v primeru na sliki 6.3 se začnejo na mejah blokov po 32 besed.

Vsaka stran je neodvisno relocirana. Ni razloga, da dve ali več strani ne bi mogle biti relocirane na isti fizični pomnilniški prostor. Uporaba več kot enega PAR registra, je en način dostopa do istega pomnilniškega prostora (podatkov). Vsak dostop pa lahko ima različne pravice dostopa.

6.2.3 Pomnilniške enote

Blok	:	32 besed
Stran	:	1 do 128 blokov
Število strani	:	8 za vsak način
Velikost pomnilnika, ki sa lahko relociramo	:	32768 besed

6.3 ZAŠČITA

Sistem z dodeljevanjem časa izvaja več programov in pri tem dovoljuje, da je v pomnilniku več programov istočasno, ki se izvajajo eden za drugim. Dostop do teh programov in prostor v pomnilniku, ki sa zavzemajo mora biti strogo definiran in nadzorovan. Sistemi z dodeljevanjem časa imajo več vrst zaščite pomnilnika. MM enota je zasnovana tako, da lahko izvede tri tipe zaščite.

Npr.:
1. Uporabniški programi se ne smejo razširiti čez dodeljeni prostor, razen če jim to dovoli sistem.
2. Uporabnikom je treba preprečiti, da bi lahko spreminjali skupne subrotine in algoritme, ki jih lahko uporabljajo vsi uporabniki.
3. Uporabnikom je treba preprečiti, spreminjanje operacijskega sistema.

6.3.1 Prepoved dostopa

Vsaka stran ima dvobitni ključ kontrole dostopa. Ta ključ je pod programsko kontrolo. Ko je vrednost ključa 0, je stran definirana kot non-resident. Vsak poskus programa, da bi dosegel takšno stran, se konča s takojšnjim abortom. Na ta način se zaščiti pomnilnik in samo tiste strani, ki so povezane s programom imajo ključ, ki dovoljuje dostop. Ključi vseh ostalih so postavljeni na 0, kar prepreči nelegalno naslavljanje pomnilnika.

6.3.2 Pomnilniška stran z dostopom branja

Kontrolni ključ dostopa je v tem primeru postavljen na vrednost 2, kar dovoljuje branje pomnilnika na določeni strani, vendar pa takoj abortira kakeršenkoli poskus pisanja. Ta tip zaščite, lahko uporabimo na straneh, ki vsebujejo skupne podatke in podprograme, ki si jih ti programi delijo. Ta način dostopa dovoljuje, da imajo pravico do citanja določenih podatkov različni uporabniki. Vsak program ima različno definirane ključe dostopa do skupnih podatkov.
Vsak PAR register ne slede na Kernel ali User je lahko nas-

tavljen tako, da se nanaša na isto fizično stran v pomnilniku, vendar ima lahko definirane različne pravice dostopa. Npr.: User ključ dostopa je 2 (samo čitanje), Kernel ključ pa je lahko 6 (dovoljeno čitanje in pisanje).

6.3.3 Dvojnost naslovnega prostora

Obstajata dva popolnoma ločena niza PAR/PDR registrov; en niz za Kernel in en niz za User način delovanja procesorja. To omogoča sistemu še en način zaščite pomnilnika. Izbira delovanja je določena z bitoma 14 in 15 procesorjevega statusnega registra (PSW 14 in PSW 15).

PSW bit 15	bit 14	
0	0	! Kernel način delovanja
0	1	! Vsak poskus dostopa se abortira
1	0	! Vsak poskus dostopa se abortira
1	1	! User način delovanja

Uporabniški program relocira preko User APR registrov, sistemski programi pa preko Kernel APR registrov. S tem se prepreči, da bi program, ki dela v enem načinu, po nesreči naslavljal pomnilniški prostor, ki je dodeljen drugemu načinu, če so le APR pravilno postavljeni.

6.4 APR - REGISTER AKTIVNIH STRANI

MM enota uporablja dve skupini po osem APR registrov. Vsak APR je sestavljen iz naslovnega registra strani(PAR) in opisnega registra strani(PDR). Ti registri se vedno uporabljajo v paru in vsebujejo vse potrebne informacije, da se locira in opiše trenutno aktivna stran za vsak način delovanja. Bita 15 in 14 v PSW registru, ki označujeta tekoči način delovanja (v nekaterih primerih tudi bita 13 in 12, ki označujeta prejšnji način delovanja) določata, katera skupina registrov se bo uporabljala za dostop do pomnilnika. Program, ki deluje v enem načinu, ne more uporabljati APR registrov, ki so namenjeni za drug način delovanja.
Vsak PAR in PDR register za vsak način delovanja ima specifično V/I adreso(tabela 6.1).

POZOR!

Unibus enote(razen DMA in programske konzole) nimajo dostopa do PAR ali PDR registrov. Notranja dekodirna adresna logika (list A14) dovoljuje dostop do teh registrov samo procesorju.

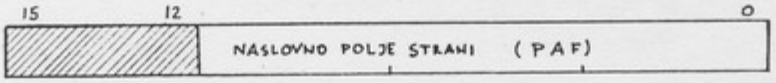
Tabela 6.1 PAR/PDR naslovi

KERNEL APR			USER APR		
St.	PAR	PDR	St.	PAR	PDR
0	772340	772300	0	777640	777600
1	772342	772302	1	777642	777602
2	772344	772304	2	777644	777604
3	772346	772306	3	777646	777606
4	772350	772310	4	777650	777610
5	772352	772312	5	777652	777612
6	772354	772314	6	777654	777614
7	772356	772316	7	777656	777614

V popolnoma zaščitenem, večprogramskem okolju, bi smelo biti samo programu, ki deluje v Kernel načinu, dovoljeno, da vpisuje na lokacije PAR in PDR z namenom, da mapira uporabniške programe. Vendar pa logika dovoljuje tudi programom v User načinu vpisovanje v te registre. Zato je naloga sistemskega inženirja, da to zaščito izvede z ustrezno programsko opremo.

6.4.1 PAR(naslovni register strani)

List A11 in slika 6.4 prikazujeta ta register, ki vsebuje 12 bitno naslovno polje strani(PAF), ki določa osnovno adreso strani.



slika 6.4

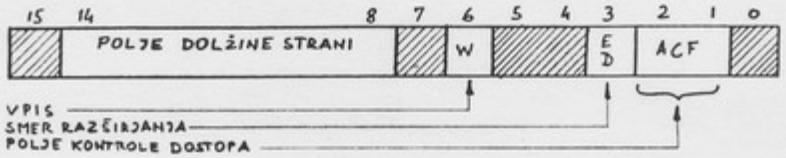
Biti 12 do 15 so neuporabljeni in so rezervirani za bodočo uporabo.

PAR se lahko interpretira bodisi kot relokacijsko konstanto ali pa kot osnovno adreso strani. Obe interpretaciji kažeta na osnovno funkcijo naslovnega registra strani pri relokaciji.

Vsebina, ki prihaja po SD vodilu, se vpisuje v PAR na naslov, ki ga izbere multiplekser E23 (list A11), če je signal A14 PAR+PDR L aktiven in to ob nastopu signala A15 REG CLK H. Zgornji biti PAR 08:11 se vpisujejo ločeno od spodnjih PAR 00:07 in sicer ob signalih A14 LOAD PAR HIGH L, oziroma ob A14 LOAD PAR LOW L. Izhodi registrov PAR se vodijo na KT MUX, multiplekser, preko katerega so dosegljivi vsi registri MM enote (list A13).

6.4.2 PDR - opisni register strani

Izvedeni so s hitrimi RAM-i (E17,E19 in E24, list A11) in vsebujejo informacije v zvezi s smerjo razširjanja strani, dolžino strani in načinu dostopa do strani (slika 6.5). Biti prihajajo preko SD vodila in sicer na adreso, ki jo izbere PAR/PDR ADRS MUX (E23 list A11), vpišejo se ob nastopu REG CLK H.

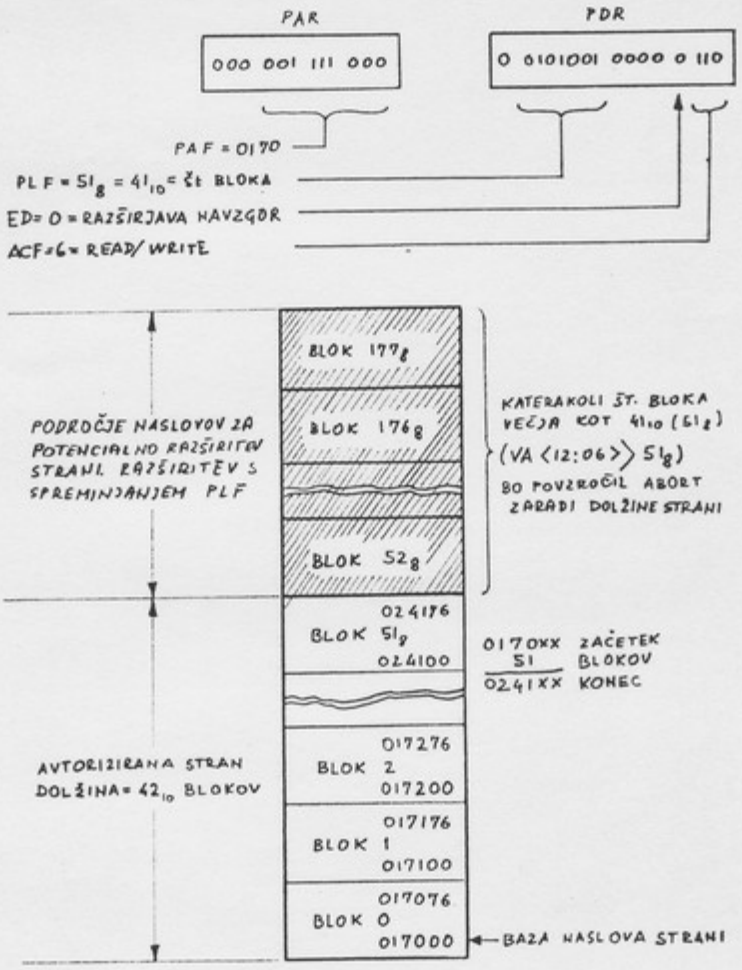


Slika 6.5

a) Polje kontrole dostopa(ACF) opisuje način dostopa za določeno stran. Dostopna koda ali ključ [A11 ACF2(1)H in A11 ACF1(1)H] odloča o tem, ali je za dotično stran dostop dovoljen, ali ni dovoljen in ga je potrebno abortirati. Dostop do pomnilnika, ki je abortiran se v trenutku prekine.
Aborte povzročijo poskusi dostopa do strani, ki jih ni, napake v zvezi z dolžino strani, nedovoljeni dostopi kot je poskus vpisa na stran, ki je samo za branje. Vsi vektorji pasti MM gredo preko lokacije 250 in se jih lahko uporabi kot pomoč pri zbiranju informacij o MM pasteh.
V zvezi s kontrolo dostopa se izraz "vpis" uporablja, da označuje poskus kateregakoli ukaza, ki spremeni vsebino adresirane besede. Slika 7 je seznam štirih ACF ključev in njihovih funkcij. ACF se vpiše v PDR pod programsko kontrolo.

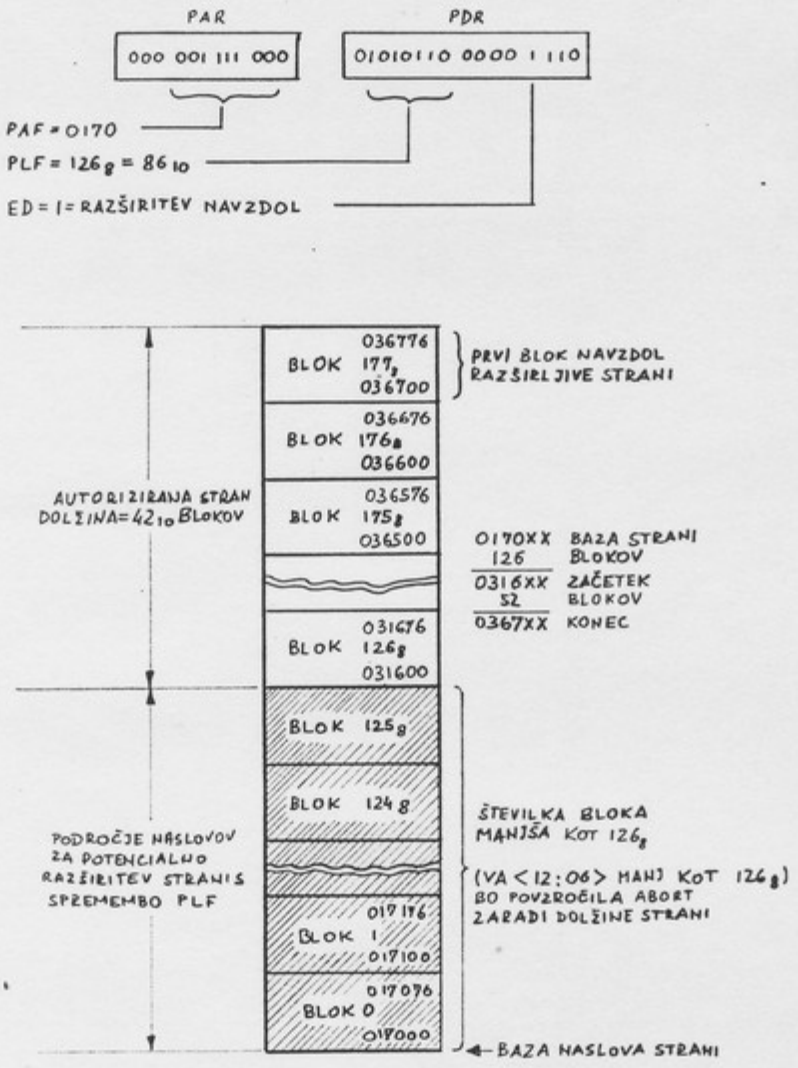
ACF	Ključ	Opis	Funkcija
00	0	Stran ne obstaja	Abortira vsak poskus dostopa
01	2	Samo beri	Abortira vsak poskus vpisa
10	4	Neuporabljeno	Abortira vsak poskus dostopa
11	6	Beri/pisi	Ne abortira

b) ED(smer razširjanja) označuje bit 3 PDR registra. Podaja smer razširjanja strani. Če je vrednost tega bita "0", potem se stran lahko širi navzgor od relativne ničle. ED=1 pa označuje razširjanje navzdol proti relativni ničli. ED bit se vpisuje pod programsko kontrolo. Ko je smer razširjanja navzgor(ED=0), se dolžina strani povečuje z dodajanjem blokov z višjim relativnim naslovom. Razširjanje navzgor običajno uporabljajo programi ali podatkovne strani za dodajanje programa ali prostora za tabele. Primer razširjanja navzgor je prikazan na sliki 6.6.



Slika 6.6

Ko je smer razširjanja navzdol(ED=1), se dolžina strani povečuje z dodajanjem blokov z nižjim relativnim naslovom. Razširjanje navzdol je običajno definirano za strani sklada, da bi se lahko dodajal prostor za sklad. Primer razširjanja strani navzdol je prikazan na sliki 6.7.



Slika 6.7

POZOR!

V zgornji zlog PDR registra se vpisuje največja dovoljena številka bloka. Bit 15 se ne uporablja, ker je lahko največja dovoljena stevilka bloka 177(osmiško).

c) Bit 6 PDR registra je W bit, ki označuje, če je bilo že kaj vpisano v stran, potem ko se je naložila v pomnilnik. W=1 potrjuje vpis. W bit se avtomatično pobriše, ko se vpisujeta PAR in PDR te strani. Ponovno sa lahko postavi le kontrolna logika

(list A11). V aplikacijah, kjer se izvaja disk-swapping in memory overlay, se lahko W bit uporablja, da se določi, katere strani v pomnilniku je uporabnik že spremenil. Te strani morajo biti shranjene v trenutni obliki. Tiste, v katere pa se ni vpisovalo (W=0), ni potrebno shraniti in se jih lahko prekrije z novimi stranmi, če je to potrebno.

d) Sedem bitno polje dolžine strani(PLF) sestoji iz bitov 08:14 PDR, ki določajo dovoljeno dolžino strani v 32 besednih blokih. PLF lahko vsebuje številke od 0 do 177(8) in tako dovoljuje dolžino strani od 1 do 128 blokov. PLF polje se vpiše v PDR signalom A14 LOAD PDR HIGH L pod programsko kontrolo.
Ko se stran razširja navzgor (ED=0), mora biti vrednost PLF za eno nižja kot je število blokov te strani. Če je število blokov 42(10) oz. 52(8), potem je vrednost PLF enaka 51(8) oz. 41(10). Blok 0 je meja strani in hkrati prvi blok strani. Komparator primerja številko bloka virtualnega naslova (VBA 12-06) s PLF, da določi, če je VBA znotraj dovoljene dolžine strani. Če je številka VBA bloka manjša ali enaka PLF, potem je VBA znotraj dovoljene dolžine strani. Če pa je številka bloka večja kot PLF, aparaturna oprema detektira napako zaradi prekoračitve dolžine strani in pošlje A12 KT FAULT L signal vezju, ki kontrolira podatkovne prenose (list B10). Tam se generira B10 ENAB ABORT H, ki povzroči abort. Ko je smer razširjanja navzgor se dolžina strani povečuje z dodajanjem blokov z višjimi relativnimi naslovi. Razširjanje navzgor se uporablja za programske ali podatkovne strani, da se lahko dodaja program ali podatki v tabelo.
Možnost razširjanja strani navzdol je posebej namenjena za strani, ki se uporabljajo kot sklad. Sklad se začne na najvišjem naslovu, ki je zanj rezerviran in se razširja navzdol proti najnižjemu naslovu, kot se podatki nalagajo na sklad. Če naj se stran razširja navzdol, mora biti PLF nastavljen tako, da dovoljuje dolžino strani (v blokih), ki se začne na najvišjem naslovu strani, kar je vedno blok 177(8). Število blokov, ki jih vpišemo v PLF dobimo tako, da izračunamo dvojiški komplement tega števila.

Maksimalna st. bloka-PLF		Zahtevana dolžina
177(8)	- 52(8) =	125(8)
127(10)	- 42(10) =	85(10)

Slika 8 kaže primer strani, ki se razširja navzdol. Dolžina strani 42 blokov je poljubno izbrana, da lahko primerjamo s primerom, ki kaže stran z razširjanjem navzgor(slika 6.6).

POZOR!

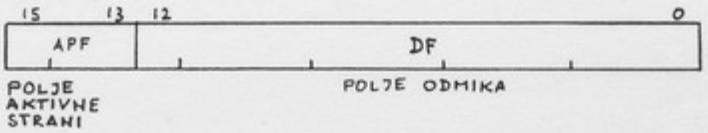
V obeh primerih je uporabljen isti PAF. S tem naj se poudari , da PAF kot začetni naslov, vedno določa najnižjo adreso strani, ne glede ali gre za stran, ki je razširljiva navzgor ali navzdol.

6.5 VIRTUALNI IN FIZIČNI NASLOV

Vezje za naslavljanje je prikazano na listu A10. Ko je MM enota omogočena (A12 RELOCATE H je postavljen), procesor preneha nalagati naslove na Unibus direktno iz ALU registrov. Namesto tega se naslovi relocirajo s pomočjo različnih konstant, ki se dobijo iz MM vezja (izbrana PAR vsebina se prišteje VBA z uporabo seštevalnikov Y23, Y26 in Y27)

6.5.1 Sestava fizičnega naslova

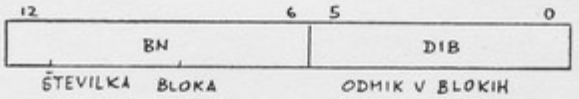
Osnovna informacija, ki je potrebna za sestavljanje fizičnega naslova (PA), pride iz virtualnega naslova (VBA), ki ga ilustrira slika 6.8.



Slika 6.8

Virtualni naslov je sestavljen iz:

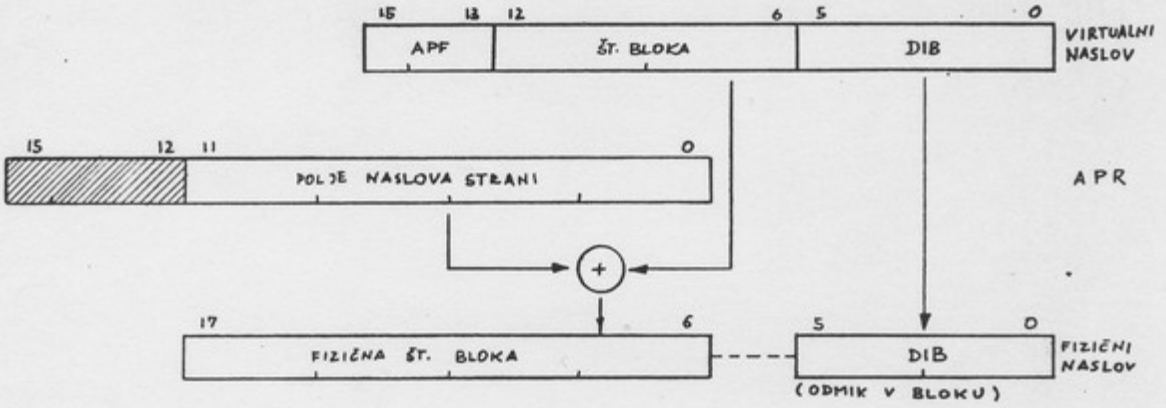
- Aktivnega polja strani(APF). To polje določa kateri od osmih APR registrov bo uporabljen za tvorbo fizičnega naslova. PAR/PDR ADDRS MUX (E23 na A11)izbere določeni PAR
- Polje odmika(DF). To 13 bitno polje vsebuje relativen naslov z ozirom na začetek strani. Dovoljena dolžina strani je 4K besed(2**13=8Kzlogov). DF se deli na dve polji, ki kaže slika 6.9. BN je številka bloka in DIB je odmik znotraj bloka.



Slika 6.9

Ostale informacije, ki so potrebne za sestavo fizičnega naslova,

pridejo iz 12 bitnega naslovnega polja strani(PAF), ki opisuje začetni naslov pomnilnika, ki ga APR opisuje. PAF je dejansko številka bloka v fizičnem pomnilniku. Npr. PAF=3 označuje začetni naslov 96 (3*32=96) besede v fizičnem pomnilniku. Formiranje fizične adrese je prikazano na sliki 6.10.



Slika 6.10

Fizični naslov se sestavlja na sledeči način:

1. Izbere se niz APR glede na trenutni način delovanja
2. Z biti 15,14 in 13 virtualnega naslova izberemo en APR
3. Naslovno polje strani (PAF) izbranega APR registra vsebuje začetni naslov trenutno aktivne strani, ki je izražen kot številka bloka v fizičnem pomnilniku
4. Številka bloka(BN) iz virtualnega naslova se prišteje k številki bloka iz PAF polja. Tako dobimo številko bloka v fizičnem pomnilniku.
5. Odmik v bloku dobimo iz DIB polja virtualnega naslova in ga pridružimo fizični številki bloka, s tem se dobi pravi 18 bitni fizični naslov.

6.5.2 Določanje programskega fizičnega naslova

Virtualni naslov omogoča naslavljanje do 32K besed v območju od 0 do 177776 (meje besed so soda oktalna števila). Trije najpomembnejši biti virtualnega naslova izbirajo APR register, ki naj se uporabi med relokacijo naslova strani. Tabela 6.2 vsebuje seznam virtualnih naslovnih območij, ki jih določa vsak APR.

Tabela 6.2

Virtualno naslovno območje	PAR/PDR
000000 - 17776	0
020000 - 37776	1
040000 - 57776	2
060000 - 77776	3
100000 - 117776	4
120000 - 137776	5
140000 - 157776	6
160000 - 177776	7

POZOR!

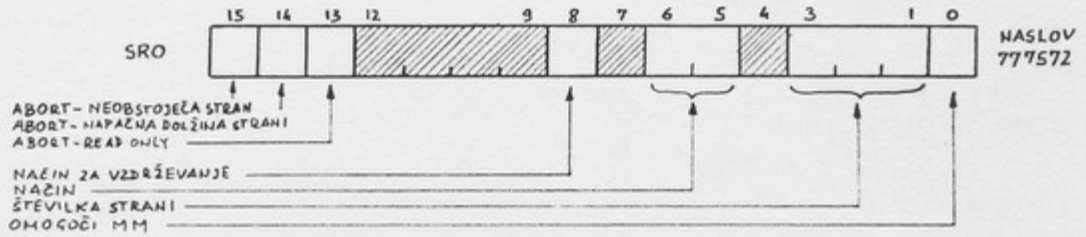
Vsaka uporaba strani, ki je je krajša od 4K besed povzroči luknje, ki ostanejo v virtualnem naslovnem prostoru.

6.6 STATUSNI REGISTRI

Aborti, ki jih generira zaščitna aparaturna oprema, gredo preko Kernel virtualne lokacije 250. Statusna registra SR0 in SR1 se uporabljata pri ugotavljanju vzroka za abort. Abort na lokacijo, ki je tudi sama neveljavna, bo povzročil nov abort. Tako mora Kernel program zagotoviti, da se virtualni naslov 250 mapira na veljavni naslov. Drugače se bo pojavila zanka, kar bo zahtevalo intervencijo s konzole.

6.6.1 Statusni register SR0

SR0 (list A12) vsebuje bite, ki označujejo napake, zaradi katerih je prišlo do aborta, bit, s katerim vključuje delovanje MM enote in druge bistvene informacije, ki jih potrebuje operacijski sistem za reševanje iz aborta oz. pasti. Format SR0 registra kaže slika 6.11. Naslov registra je 777572.



Slika 6.11

Biti 15-13 so biti abortov. Lahko smatramo, da se po prioriteti vrstijo tako, da so biti, ki so bolj desno, manj pomembni. To pomeni, da servisna rutina aborta zaradi strani, ki ne obstoja, ignorira abort zaradi napačne dolžine strani in abort zaradi prepovedanega pristopa. Servisna rutina aborta, ki ga povzroči napačna dolžina strani, pa ignorira napako zaradi napačnega dostopa.

POZOR!

Postavljen bit 15, 14 ali 13 povzroči, da logika (E32 generira A12 ERROR H) shrani vsebino bitov 1 do 6 registra SR0 in kompletno vsebino registra SR1. S temi podatki se olajša povratek iz aborta.

Adresa se relocira s tem, ko gre signal A12 RELOCATE H v visok nivo. To se doseže bodisi z vpisom enice v bit 0 registra SR0 s čimer se relocira vsaka adresa, ali pa z vpisom enice v bit 8 istega registra, s čimer se relocirajo le adrese ponorov (maintenance/destination mode). Informacije, ki so vpisane v vse ostale bite registra SR0 niso pomembne. Informacije se avtomatično vpišejo v register in so rezultat aktivnosti aparaturne opreme, ter so za kontrolo statusa MM enote. Postavljanje bitov 15-13 s programsko kontrolo ne bo povzročilo pojava pasti. Vendar pa morajo biti ti biti postavljeni na 0 potem, ko se je zgodil abort ali past, zato da lahko še naprej kontrolirajo MM enoto.

6.6.1.1 Opis posameznih bitov SR0

- Bit 15 označuje stran, ki je non-resident [A12 NR(1) H]. Postavi ga poskus dostopa do strani, katerega vrednost polja ACF je enaka 0 ali 4 ali pa zahteva za relokacijo z neveljavnim načinom v PSW registru.
- Abort zaradi poskusa dostopa do lokacije, ki je izven območja dotične strani označuje bit 14 [A12 PL(1)H].
- Bit 13 označuje poskus pisanja na stran, ki se lahko samo bere [A12 RO(1) H].
  Z istim poskusom dostopa je lahko postavljeno naenkrat več abort bitov.
- Bit 8 označuje uporabo MM enote v diagnostične namene. Za ukaze, ki se uporabljajo v začetnem diagnostičnem programu, je bit 8 postavljen tako, da je samo naslavljanje ponorov relocirano.
- Bita 5 in 6 označujeta način delovanja procesorja (User=11 ; Kernel=00) povezan s stranjo, ko se je pojavil abort.
- Biti 3,2,1 vsebujejo številko strani, ki jo naslavljamo. Strani kot tudi bloki, so oštevilčeni od '0' navzgor. Te bite uporablja rutina za povratek, da ugotovi stran, ki se je naslavljala, ko se je pojavil abort.
- Bit 0 označuje prisotnost oz. odsotnost MM enote. Ko je '1',se vsi naslovi relocirajo in jih ščiti MM enota. Ko pa je '0' je MM izključena in ni relokacije in zaščite.

6.6.2 Statusni register SR2

V statusni register SR2 (slika 6.12, list A13) se ob vsaki dostavi instrukcije naloži 16-bitni virtualni naslov. Vrednost SR2 se ne spremeni, če dostavljanje ukaza ni bilo uspešno. SR2 lahko le beremo, poskus vpisa ne bo spremenil njegove vsebine. SR2 je števec virtualnih adres. Ko se zgodi abort bodo biti 15, 14 in 13 SR0 registra zamrznili vsebino SR2. Adresa tega registra je 777576.

Slika 6.12

6.7 NAČIN DELOVANJA (KERNEL/USER MODE)

V Kernel načinu ima operacijski sistem neomejeno uporabo računalnika. Program lahko mapira uporabniške programe kamorkoli in tako neposredno varuje ključna področja (registre perifernih enot in PSW register) pred uporabniškimi programi.
V User načinu delovanja uporabniški program ne sme izvajati HALT instrukcije , ker bo v tem primeru procesor šel v past preko adrese 10. RESET instrukcija se tudi ne' sme izvesti v tem načinu delovanja. Če se pojavi tak ukaz, bo procesor izvedel NOP instrukcijo namesto RESET.
Vsak način delovanja ima svoj sklad. Prekoračitev sklada v User načinu ni mogoča, ker že sama zaščita pomnilnika skrbi za zaščito sklada.

6.8 POGOJI ZA PREKINITVE

MM enota relocira vse naslove. Ko je MM aktivna, se smatra, da so vsi vektorji pasti, prekinitev in abortov v virtualnem naslovnem prostoru Kernel načina, procesor sam v dani mikroinstrukciji zahteva Kernel način s postavitvijo signala A9 KERNEL H/PL/. Ko se pojavi prenos preko vektorja, se program nadaljuje z novo vrednostjo v programskemu števcu in novo procesorjevo statusno besedo, ki ju vsebuje 2-besedni vektor, in je relociran s Kernel APR registrom. Z relokacijo vektorjev se da sistem reševati tudi v primeru, če je prvi blok fizičnega pomnilnika v okvari.

Ko pride program v past , prekinitev ali abort, se stari PC in stara vrednost PSW registra odložita na User ali Kernel sklad, odvisno od vrednosti bitov 15 in 14 iz nove PSW besede. Ta dva bita tudi določata skupino APR registrov.

Delovanje User PSW:

PSW biti	Uporabniški RTI,RTT	Uporabniške pasti in prekinitve	PSW dostop
3 - 0	Naloži s sklada	Naloži iz vektorja	*
4	Naloži s sklada	Naloži iz vektorja	Se ne more spremeniti
7 - 5	Ne more se spremeniti	Naloži iz vektorja	*
13 - 12	Ne more se spremeniti	Prepiše PS(15,14)	*
15 - 14	Ne more se spremeniti	Naloži iz vektorja	*

*Neposredne operacije se lahko izvršijo, če je PSW mapirana v uporabniškem prostoru.

7. MIKROPROGRAMSKI KRMILNIK

7.1 APARATURNA OPREMA

Naloga krmilnika (Am2910 E72 na B1) je generiranje naslova za naslednjo mikroinstrukcijo. Mikroprogramski naslov se glede na krmilno informacijo in I vhodih 2910 izbira med mikroprogramskim števnikom, vsebino mikrospomina, vsebino ponavljalnega registra ali vsebino na D vhodu. Na D vhod so priključeni trije viri. Hkrati je aktiven le eden, ostala dva pa sta v visokoimpedančnem stanju. Mikroprogramski naslov, ki ga je generirala enota 2910, je možno spremeniti v skupni točki vezij z odprtim kolektorjem, od katerih se preko 3-stanjskega vmesnika posreduje na naslovne linije mikroprogramskega pomnilnika (MMP). Obstaja tudi mehanizem naslavljanja mikroprogramskega pomnilnika na ta način, da se naslov vsili neposredno na vhod MMP, brez sodelovanja sekvenčnika. Mehanizem je aktiviran v naslednjih primerih:

- pri vklopu napetosti (power up) se aktivira signal B12 MPC00 L na logično "0" kar ima za posledico vsilitev naslova 001 na MPP
- ob pojavu CIS suspenzije se vsili naslov 002
- kadar je pri prenosih CPU - glavni pomnilnik nastopila situacija, ki zahteva past, se s signalom B10 ABORT H na logični enici prekine izvajanje mikroprograma in se vsili naslov 000 in s tem mikroinstrukcija SERVICE, ki reagira na past

7.1.1 Pipeline selektor

Funkcija selektorja je izbira enega izmed štirih virov. Izhod je priključen neposredno na D vhod 2910 in je onemogočen le v tistih mikroinstrukcijah, pri katerih se izvaja JMAP instrukcija. Selekt vhodi PL - selektorja so krmiljeni neposredno iz pipeline registra s signaloma B6 PLS0 (1)H/PL/ in B6 PLS1 (1)H/PL/ pri čemer velja naslednja tabela:

B6 PLS1 (1)H/PL/	B6 PLS0(1)H/PL/	
0	0	PL + MODE
0	1	PL + TIP
1	0	PL
1	1	ALU NN

PL + MODE vhod uporabljamo v primerih, ko želimo večvejni skok glede na način naslavljanja v IR registru.
Vrata 74LS27 (E64 na B1) detektirajo MOV(B) instrukcijo iz grupe 1 in z logično enico na svojem izhodu omogočijo po končani dostavi izvornega operanda nadaljevanje dostave ponornega operanda v dostavni sekvenci za grupo 5. Pri vseh ostalih instrukcijah iz grupe 1 ta vrata generirajo logično ničlo tako, da sledi dos-

tavna sekvenca za grupo 4.
Za CIS nabor je rezerviran večvejni skok glede na tip stringa, ki ga definirajo biti 12,13 in 14 v deskriptorju. Le-ta se nahaja v registrih ALU-ja in kadar želimo večvejni skok glede na tip stringa posredujemo pripadajoč deskriptor na DB izhod ALU enote. Signale A4 ALU DB 12 H, A4 ALU DB 13 H in A4 ALU DB14H, ki definirajo vrsto stringa, uporabimo na PL selektorju, tako da s svojo vrednostjo določajo enega izmed osmih naslovov v večvejnem skoku.
Za brezpogojne in pogojne vejitve se celoten 12 bitni naslov iz pipeline registra posreduje preko PL selektorja (uporabljen je vhod PL) na D vhod 2910.
ALU NN vhod se v standardnem naboru uporablja za inicializacijo vsebine števca iteracij v 2910. Nova vsebina, ki se nahaja v ALU registru, se preko DB izhoda posreduje na PL selektor in preko tega na D vhod sekvenčnika ter se z instrukcijo LDCT vpiše v števec iteracij. V okviru CIS suspenzije je možno preko tega vhoda inicializirati tudi mikrosklad in mikroprogramski števnik.
Aktiviranje dekodirne logike na vhodu sekvenčnika omogoča JMAP instrukcija takoj za tem, ko se je nova makroinstrukcija že vpisala v IR register. Naslov, ki ga definira dekodirna logika je sestavljen na sledeč način:

11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	G	R	U	P	A	0	ADRESNI NAČIN	

V primeru grupe 0 dobimo na D vhod same ničle, kar omogoča začetek izvajanja programske pasti (BPT,EMT,IOT,TRAP) v okviru makroinstrukcije SERVICE.
Kadar se dekodira ena izmed vejitvenih makroinstrukcij (aktivna je grupa 3), se na D vhodu sekvenčnika pojavi naslov 030. Pri grupi 1,2,4 in 5 pa nam izbiralnika (E92 in E89 na B4) generirata ustrezen naslov glede na način naslavljanja, ki se nahaja v instrukcijskem registru. Pri grupi 1 gre za način naslavljanja izvornega operanda, ki ga definirajo signali B4 IR 11(1)H, B4 IR 10(1)H ter B4 IR 09(1)H saj je signal B3 GR 1 L na logični ničli, kar odpira A vhod izbiralnika 74S157-E92. Pri grupah 2,4 in 5 pa izbiralnik posreduje na izhod način naslavljanja, ki ga definirajo signali B4 IR 05(1)H, B4 IR 04(1)H in B4 03(1)H.
V primeru, da se v IR nahaja instrukcija iz grupe 7, se aktivira signal B1 DEK ROM L na logično "0" kar omogoča, da se 8-bitni naslov vzame iz enega izmed PROM-ov, ki tvorijo mikroprogramske naslove začetka eksekucij za posamezne makroinstrukcije.
Signal B1 DEK ROM L je na logični "0" tudi ob koncu dostavne faze v grupi 1,2,4 in 5 ter skupaj s signali B3 CS1 L, B3 CS2 L in B3 CS 1 L (od katerih je le eden na logični "0", aktivira enega izmed PROM-ov, da posreduje startni eksekucijski mikroprogramski naslov, ki pripada makroinstrukciji iz IR.

7.1.2 Pogojne vejitve

Mikroprogramski sekvenčnik opravlja pogojno vejitev v primeru, da se na I vhodih nahaja CJP instrukcija ter, da je signal B6 CCEN (1)L/PL/ na logični '0'. Vejitveni naslov v tem primeru posreduje PL selektor, vejitev pa dobimo v primeru, ko se na CC vhodu 2910 pojavi logična '0'. CC vhod krmili signal B5 CC TEST L, ki je v standardnem instrukcijskem naboru generiran na CC selektorju (E62 na B5), medtem ko ga v CIS naboru generira element 74S251 E59 na B5. Funkcija CC selektorja je izbiranje določenega testnega pogoja glede na signale B7 CCS 0(1)H/W3/ po tabeli:

W3	W2	W1	
0	0	0	PC
0	0	1	PC V SP
0	1	0	MODE 0
0	1	1	ZERO
1	0	0	CT
1	0	1	N
1	1	0	(REZERVIRANO)
1	1	1	SHOVR

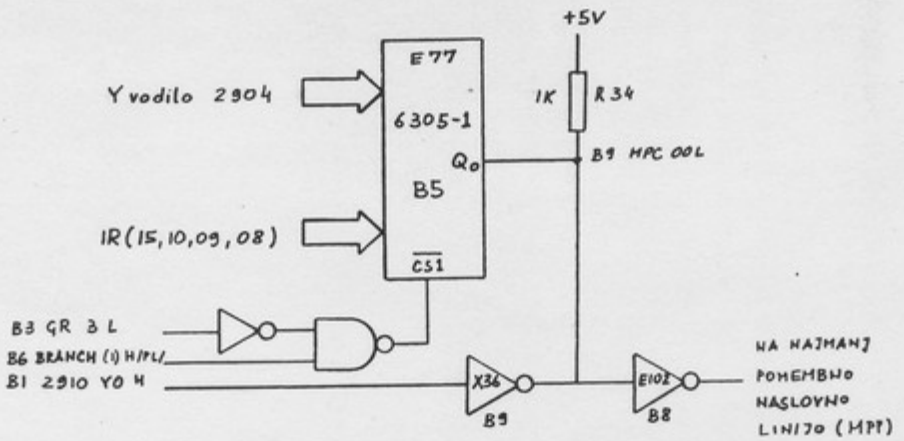
Vejitev se ne izvede (sledi izvajanje naslednje mikroinstrukcije), če je signal B5 CC TEST L na logični '1'.

7.1.3 Aparaturna pogojna vejitev

Skupna točka vezij z odprtim kolektorjem omogoča wired-or funkcijo in na ta način daje možnost za spreminjanje mikroprogramskega naslova, kar predstavlja dodatni vejitveni mehanizem za realizacijo aparaturnih pogojnih vejitev. Ta mehanizem se aktivira s signalom B6 BRANCH (1)H/PL/ na logični '0' v primeru, da se ne izvaja JMAP instrukcija na 2910, ampak sekvenčnik generira sodi vejitveni naslov z brezpogojno vejitveno funkcijo JMPBR. Kadar sta oba vhoda na vratih 74S03 E104 na B9 na logični '1' dobimo v točki s skupnim kolektorjem forsiranje logične '0', kar ima za posledico spremembo osnovnega mikroprogramskega naslova.Na ta način se je sodi naslov spremenil v lihega in s tem se izvede aparaturna pogojna vejitev. Aparaturne vejitve ni v kolikor je signal B5 CC TEST L na omenjenih vratih na logični '0'.

7.1.4 Branch PROM

Mehanizem aparaturne pogojne vejitve je uporabljen tudi pri aparaturno-mikroprogramski realizaciji vejitvenih makroinstrukcij (vse instrukcije iz grupe 3). Slika 7.1 prikazuje aparaturno vključitev BRANCH PROM-a.



Slika 7.1

Vpliv BRANCH PROM-a je dovoljen v primeru, ko gre za grupo 3 (signal B3 GR 3 L je na logični "0") in je aktiviran signal B6 BRANCH (1)H/PL/ na logično "0". Na izhodu 2910 se nahaja mikroprogramski naslov 030 in ostane nespremenjen v kolikor je izhodni signal PROM-a B9 MPC 00 L na logični "1". Kadar dobimo na izhodu PROM-a logično "0" (v makrostatusnem registru enote 2904 se nahajajo take pogojne kode, da je za dano makro vejitveno instrukcijo pogoj za vejitev izpolnjen) se naslov 030 spremeni v 031, kjer se začenja mikroprogram, ki realizira zahtevano vejitev na makro nivoju. Na naslovne linije BRANCH PROM-a so pripeljane pogojne kode iz makrostatusnega registra preko Y vodila enote 2904 ter signali B4 IR 15 (1)H, B4 IR 10 (1)H, B4 IR 09 (1)H ter B4 IR 08 (1)H, ki definirajo vrsto vejitve.
Izhodi krmilnika Y0-Y11 ne gredo neposredno na adresne vhode mikroprogramskega pomnilnika, pač pa jih vodimo preko dveh skupnih vrat, prva (X36,X33 na B9) so taka z odprtim kolektorjem, to pa zaradi tega, ker na tem mestu vstopa adresa za FP mikroprograme, ko je FP priključen, prav tako se tu bit MPC 00 L postavlja v odvisnosti od mikroprogramskega pogojnega skoka, (E107, E104 na B9). Spreminjamo lahko se B9 MPC 01 L preko vezja E104 na B9, B9 MPC 06 L in B9 MPC 07 L preko E104 na B9. Vsa ta vezja služijo za mikroprogramski skok ob testiranju pogojnih bitov C, N in Z. Za procesor FP velja, da je B9 MPC 09 L aktiven, zatorej naslovimo lokacije nad 512 v mikroprogramskem pomnilniku.
Drugi niz vrat je realiziran z X50, X52, E102, X55 na B8.

Opravlja sledeče naloge:

-Če je aktiven B8 A L, tedaj imamo odprto pot iz Y izhodov krmilnika na adresne vhode mkiroprogramskega pomnilnika.

-Če ni aktiven B8 A L, tedaj gre za slučaj vsiljevanja adres, B8 B L mora biti aktiven (E105 na B8), kar pomeni prisotnost B12 MPC 00 L (POWER UP), B10 ABORT H oziroma B5 DSUS (1) L (suspenzija) ter B12 PROC INIT H.

Vsilijo se sledeče mikroprogramske adrese "001" za POWER UP, "002" za suspenzijo in "000" za abort. Vsilitev teh različnih adres je izvršena preko vezij E99, E45 in E44 na B8.
Mikroprogramski pomnilnik obsega 512 lokacij, dolžina mikroinstrukcije je 88 bitov, pomnilniki so bipolarni PROM velikosti 512*8: E97, E28, E37 , E58, E52 na B6, E43, E88, E67, E73, E82, E103 na B7. Izhod mikroprogramskega pomnilnilka vodimo na PL register, ki ga sestavljajo vezja E114, X44, X56, X51, X49 na B6, X48, X57, X53, X56, X58 na B7. PL register menja vsebino ob prednji fronti urinega signala A15 PROC CLK L, vsebino pa brišemo ob B12 PROC INIT L.

7.1.5 Vezje suspenzije

Prekinitev instrukcije iz nabora CIS omogočimo, ko so hkrati aktivni signali:

B3 GR 8 H , $\overline{\text{A5 PSW 08 (1) L}}$, B12 INTR H in IZHOD(E90 B5).

Negirana vrednost konjunkcije le-teh in prva fronta urinega signala A15 2925 C3 L ,ki jo vodimo na CP vhod FF X43, nam postavi izhod Q, oziroma briše /Q, kar pomeni, da signal B5 DSUS (1)L postane aktiven. Izhod Q pomnilne celice asinhrono brišemo preko vrat s konjunkcijo signalov B8 SUS L in B12 PROC INIT L. Vezje E90 služi za detekcijo mikrooperacijske kode CONT na krmilniku 2910. Izhod B5 DSUS L(1) nam asinhrono postavi bit 8 v PSW, torej A5 PSW 08 (1) H. Vrednost tega FF lahko brišemo ali postavimo samo se z mikroprogramom. Signal B8 SUS L je aktiven, ko se na vhodih vrat E109 na listu B8 pojavi mikroprogramska adresa "1FX" in ko je prisotna instrukcija CONT ter signala DAS (1) H/W1/ in DAS (1) H/W2/ nista aktivna. Zakaj potrebujemo signala DAS (1) H/W1/ in /W2/? Možno je polje v mikroinstrukciji NA00-NA11 uporabiti tudi kot "immediate operand", tako da sta DAS (1) H /W1/ in DAS (1) H /W2/ aktivna. Aktiven signal B8 SUS L povzroči, da zapremo pot Y izhodov iz 2910 na mikroprogramski pomnilnik. Preko bufferjev X50 in X52 na listu B8 omogočimo, da deluje mikroprogramski pomnilnik skupaj s PL kot samostojen avtomat. Omogočena je tudi pot Y izhodom iz 2910 na ALU Y vhode preko vezij X40 in X42 na listu B9.

Potek prekinjene (suspendirane) CIS instrukcije!

Izhod pomnilne celice X43 na B5 se postavi, le če so prisotni vsi sledeči signali:

GR8 . (koda CONT na 2910). $\overline{\text{PSW(8)}}$. INTR H kjer so:

GR8 H --- grupa 8, sem sodijo CIS instrukcije

CONT --- mikrooperacija na kontrolerju 2910

$\overline{\text{PSW (8) L}}$ ---bit 8 v PSW hrani stanje instrukcije iz CIS in sicer "0" ni suspenzije, "1" je suspenzija

INTR H --- I/O ali PFAIL interrupt

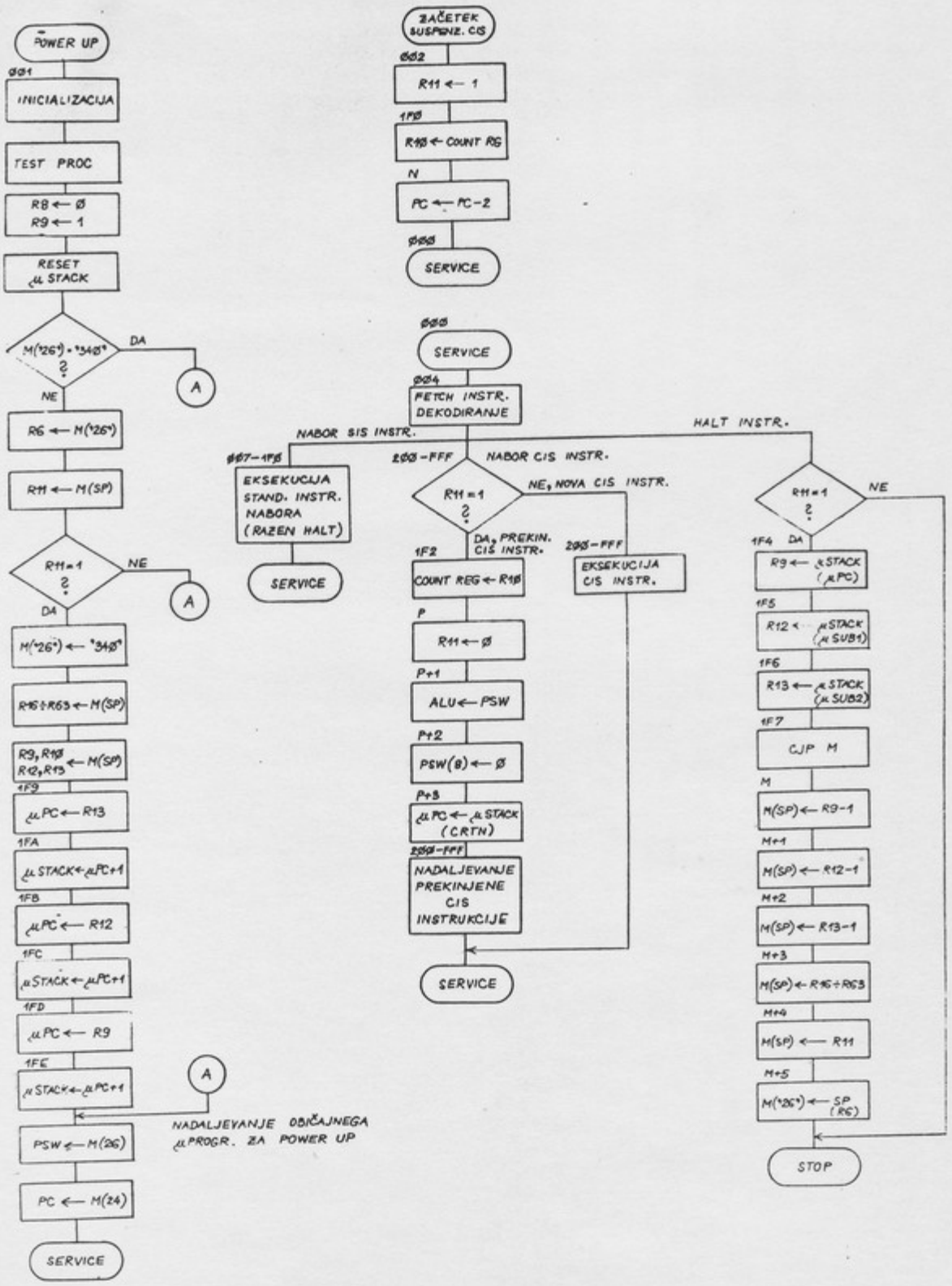
Ob detekciji suspenzije se vsili koda CJS na 2910 (glej vezje E87) ter "002" na mikroprogramski pomnilnik (vezje X50, X52 na B8 ter X42 na B9). Generirani signal D SUS (1) L istočasno zapre pot od kontrolerja na mikroprogramski pomnilnik preko vezij E102 in X55 na B8 ter omogoči izhodom iz PL registra B6 NA 00(1) H - B6 NA 11(1), da jih preko vrat X50 in X52 na B8 usmerimo na adresne vhode mikroprogramskega pomnilnika. Taka povezava ni nič drugega kot mikroprogramski avtomat, saj mora prevzeti kontrolo nad shranjevanjem mikroprogramskega števca na mikrosklad v 2910.
Takoj ko začne delovati mikroprogramski avtomat, se aktivira izhod vrat E109, ki generira signal B8 SUS L, istočasno pa nam asinhrono briše izhod FF X43 na B5.
Enostavnejša varianta suspenzije nastopi v primeru I/O interrupta, zahtevnejša pa ob PFAIL interruptu. V prvem slučaju je potrebno le shraniti vsebino COUNT REG iz 2910 v ALU in uPC na mikroprogramski sklad (uSTACK) v 2910, istočasno se postavimo R11=1. V drugem slučaju je poleg tega potrebno shraniti se celoten uSTACK iz 2910 v ALU ter nato se vse registre iz ALU-ja (64) na sklad (SP). Slednje se izvede ob zadnji instrukciji subrutine za izpad napetosti (PFAIL), to je v HALT instrukciji, ki jo s tem namenom spremenimo (glej diagram poteka).
Postopek restavriranja suspendirane instrukcije poteka v obratnem vrstnem redu. Če smo imeli opravka z navadno prekinitvijo I/O in gre za suspendirano CIS instrukcijo (ALU R11=1), tedaj vrnemo vsebino v COUNT REG v 2910 iz ALU, resetiramo PSW(8) in postavimo v register R11 vrednost nič, ter iz uSTACK-a vpišemo v mikroprogramski števec njegovo vrednost ob suspenziji.
Če je bil vzrok prekinitve izpad napetosti (PFAIL), je kriterij za restavriranje suspendirane instrukcije izpolnjen ob ponovnem vklopu napetosti (POWER UP). Po končanem mikrodiagnostičnem testu, ki ga izvedemo v mikroprogramu za POWER UP najprej posledamo lokacijo "26" v glavnem pomnilniku. Le-ta običajno hrani vrednost PSW ("340"). Vsaka drugačna vsebina pomeni, da je na tej lokaciji vrednost kazalca sklada SP, ki kaže na vrh le-tega, tja kjer je shranjen R11. V primeru suspenzije (R11=1) se v obratnem vrstnem redu prenesejo vsebine sklada (SP) v registre ALU, nato pa se iz ALU v uSTACK.

Ko procesor zaključi POWER UP mikroprogram, je pripravljen za izvajanje suspendirane instrukcije. Ob dostavi operacijske kode se najprej vprašamo, kakšen je R11. Če je R11=1, tedaj to ni običajna CIS, temveč prekinjena CIS instrukcija. To pomeni vračanje vrednosti uSTACK, COUNT REG v krmilnik 2910 iz ALU, kot zadnja sledi mikroinstrukcija, ki naloži v uPC vrednost ob suspendirani CIS. Potek prekinjanja in restavriranja instrukcije CIS vidimo na diagramu poteka.
Vlogo mikroprogramskega avtomata vidimo iz tabele:

adresa uPP	uinst. 2910	NA00-NA11	funkcija v ALU	komentar
X	CJS*	002*	-	začetek suspenzije
002	CJP	1F0	R11 1	
1F0	JPP	N	R10 Y(2910)	
1F1	-	-		
1F2	LDCT	P	R2910 R10	dostava suspendirane
P	CRTN	-		instrukcije
1F4	CRTN	1F5	R9 Y(2910)	mikroprogram HALT
1F5	CRTN	1F6	R12 Y(2910)	istrukcije
1F6	CRTN	1F7	R13 Y(2910)	shranjevanje registrov
1F7	CJP	M	-	suspendirane instrukcije
1F8	-	-	-	iz sklada v ALU
1F9	CJP	1FA	uPC R13	mikroprogram za POWER UP
1FA	CJS	1FB	uSTACK uPC+1	("001") vračanje
1FB	CJP	1FC	uPC R12	shranjenih registrov
1FC	CJS	1FD	uSTACK uPC+1	iz sklada v ALU
1FD	CJP	1FE	uPC R9	
1FE	CJS	POWER UP	uSTACK uPC+1	

*-pomeni vsilitev kode oz. podatek

Diagram poteka



7.2 MIKROPROGRAMSKA BESEDA

Polje	Dolžina polja	Komentar
NAC	4	Kontrola krmiljenja enote AM2910.
CCEN	1	Omogoči pogojne skoke na enoti AM2910.
BRANCH	1	Omogoča vpliv BRANCH ROM-a ali izhod CC selektorja na najmanj pomenbni bit mikroadrese.
DS	2	Kontrolira selekt linije DS MUX-a Podatki DS1 DS0 DBUS 0 0 PSW 0 1 YA 1 0 DB 1 1
C1 C0	2	Omogoči Unibus kontrolne linije BUS C0 L in BUS C1 L. Prenos C1(1) H C0(1) H DATI 0 0 DATIP 0 1 DATO 1 0 DATOB 1 1
NASLEDNJI NASLOV	12	12-bitni naslov naslednje mikro-instrukcije.
IMMD	16	Konstantna vrednost v mikropro-gramskem pomnilniku, ki jo lahko uporabimo v ALU-ju
PLS	2	Kontrolira selektne linije PL-S Podatki PLS 1 PLS 0 PL+MODE 0 0 PL+TIP 0 1 PL 1 0 ALU NN 1 1
MAPS	1	Loči med dostavo in eksekucijo instrukcije.
ALU A ADR	6	Omogoča adresiranje registrov ALU-ja iz mikroprogramskega pomnilnika če je tako izbran A adresni selektor.
ALU B ADR	6	Omogoča adresiranje B strani ALU-ja iz mikroprogramskega pomnilnika če je tako izbran B adresni selektor.

Polje	Dolžina polja	Komentar
A ADR S	2	Kontrolira selektne linije A ADR MUX-a Izbira AS1 AS0 A REG 0 0 IRLA 0 1 IRHA 1 1
B ADR S	2	Kontrolira selektne linije B ADR MUX-a Izbira BS1 BS0 B REG 0 0 IRLB 0 1 ABUF 1 0 IRHB 1 1
2903 OEY	1	Krmili Y ALU-ja kot vhod ali izhod.
ALU FUNKCIJA	4	Določajo ALU-funkcijo.
ALU DST	4	Določajo pomikanje in kontrolo vpisa v RAM ALU-ja.
2903 I0	1	Izbira med Q registrom in RAM-om ALU-ja.
2903 EA	1	Izbira med RAM-om in zunanjim DA vhodom ALU-ja.
DST	1	Omogoča tro-adresni način.
CX SEL	1	Omogoča vsilitev ničle ali enice na Cx vhod enote AM2904.
2904 CARRY	2	Omogoča izbiro na Co izhod enote AM2904.
2904 ENABLE BITI	8	Krmiljenje ENABLE BIT-ov enotè AM2904.
2904 I5 H	1	Signal za krmiljenje TEST-ov enote AM2904.
SHIFT	5	Krmiljenje pomikov enote AM2904.
TEST + 2904 I5	6	Krmiljenje TEST-ov enote AM2904.
START TRAN	1	Omogoča krmiljenje prenosov.
DBE	1	Detekcija dvojnih napak na vodilu
DAS	2	Krmili selektne linije DAS MUX-a Izbira DAS1 DAS0 IMMD 0 0 OFFS 0 1 SD 1 0 CIS ROM 1 1

Polje	Dolžina polja	Komentar
CCS	3	Krmili selektne linije CC-S MUX-a pri pogojnih skokih.
		Izbira CCS2 CCS1 CCS0
		PC 0 0 0
		PC V SP 0 0 1
		MODE 0 0 1 0
		ZERO 0 1 1
		CT 1 0 0
		N 1 0 1
		SHOVR 1 1 1
SWAP	1	Omogoča zamenjavo zlogov .
LOAD BA	1	Omogoča vpis adrese v BA register
DEC 1	3	Dekoder 1 daje naslednje ENABLE SIGNALE :
		R V 1 L-kontrolira kateri register bo izbran; če je R sodi register, potem je R V 1 lihi register; če R lihi register, potem je R V 1 isti register.
		ESTOV L-omogočena je logika prekoračitve kazalca sklada.
		CZERO L-vsilitev ničel na Cn vhode ALU-ja.
		PREVIOUS L-procesorju omogoči, da izvrši mikroinstrukcijo v prejšnjem MM načinu (PSW<13:12>).
		KERNEL L-procesorju vsilimo izvrševanje tekoče mikroinstrukcije KERNEL MM načinu.
		DIR L-omogočen je vpis v IR register.
		WB L-omogočeno je delo nad nižjim zlogom ALU-ja
DEC 2	2	Dekoder 2 daje naslednje ENABLE SIGNALE :
		MAINT-omogoča relokacijo v MM maintenance načinu.
		LPSW-omogoči vpis podatka v PSW register.
CLOCK	3	Izbira različne faze ure
		Izbira CLOCK L3 CLOCK L2 CLOCK L1
		CLOCK8 0 1 0
		CLOCK7 0 1 1
		CLOCK10 1 0 0
		CLOCK5 1 0 1
		CLOCK9 1 1 0
		CLOCK6 1 1 1