

KAZALO	str.
1. Pregled lastnosti in možnosti uporabe	3
2. Arhitektura EMZ 1001	5
2.1. Pomnilnik ROM	
2.2. Pomnilnik RAM	
2.3. Akumulator in ALU	
2.4. Vhodi	
2.5. Izhodi	
2.6. TTL kompatibilnost	
3. Posebni načini delovanja	13
3.1. Zunanji pomnilnik ROM	
3.2. Preskusni način	
3.3. Časovnik in ukaz EUR	
3.4. Posebni vhodi	
3.5. Posebni izhodi	
4. Tehnični podatki	17
4.1. Okrov in izvodi	
4.2. Električne specifikacije	
4.3. Kodiranje ukazov	
4.4. Nabor ukazov	
5. Razvijanje aplikativnih programov	26

1. Pregled lastnosti

Mikroračunalnik Iskra EMZ 1001 je zaključeni mikroračunalnik na eni sami ploščici silicija. Z veliko gostoto integriranih elementov prinaša vse prednosti krmiljenja z mikroračunalnikom pri minimalnih stroških opreme. Njegova notranja zgradba je usmerjena zlasti v povezavo s tastaturami in izpisi na segmentiranih svetlečih díalah (LED). Ima zelo bogato organizacijo vhodov in izhodov ter nabor ukazov, ki je optimiran glede na predvideno uporabo v sistemih z nadzornimi ali aritmetičnimi funkcijami.

EMZ 1001 ima vgrajen **pomnilnik ROM** s 1024 ukazi. Kadar je to potrebno, lahko dodajamo tudi zunanje pomnilnike do končne kapacitete 8192 ukazov. Vgrajeni programski števec s svojo vsebino kaže na ukaz, ki se trenutno izvršuje. Po izvršenem ukazu se vrednost števca spremeni in kaže na naslednji ukaz iz programa, ki se bo izvršil. **Odlagalnik** shranjuje povratne naslove subrutin, tako da se te lahko pravilno vračajo. V podatkovnem **pomnilniku RAM** shranujemo podatke, ki jih potrebujemo med izvajanjem programa. Na voljo je 64 celic, od katerih vsaka vsebuje 4 bite. V njih hranimo numerične podatke ali podatke o posameznih odločitvah. **Dostop** do posamezne celice pomnilnika RAM omogočata registera BU in BL. Register E lahko uporabljamo kot splošni register ali kot omejevalnik indeksiranja v pomnilniku RAM.

Aritmetična enota izvaja logične ali aritmetične operacije. Pri tem uporablja register A (akumulator) in register C (indikator prepolnitve). Na voljo sta tudi dve samostojni kretnici, kateri lahko programsko nastavljamo in sprašujemo.

Nadzorna logika usmerja delovanje celotnega sistema. Vsebuje tudi 3 vhode in 3 izhode, ki omogočajo povezave z zunanjimi elementi pri različnih načinih delovanja.

Oscilator generira vse potrebne časovne impulze osnovne frekvence. Vhod KREF prenaša analogno napetost, ki služi kot referenčna vrednost za vhode K. Vhodi K in I so namenjeni prenašanju zunanjih odločitev in jih preskusimo s posameznimi ukazi. Vhod I8 je povezan tudi s časovnikom.

Vseh 8 izvodov **vodila D** lahko prenaša podatke v dveh smereh. Izvodi **vodila A** prenašajo izhodne podatke, s katerimi običajno izbiramo tipke, posamezne pozicije v izpisu in podobno.

Možnosti uporabe

Mikroračunalnik EMZ 1001 je namenjen uporabi v veliko-serijskih izdelkih, ki potrebujejo sposobno nadzorno enoto na majhnem prostoru in za nizko ceno. Najprimernejši je za uporabo v izdelkih, pri katerih srečamo naslednje zahteve:

- vgrajena ura, koledar in s tem povezane časovne odločitve
- sinhronizacija z omrežno frekvenco
- izpis segmentiranih številčnih prikazov
- vhodi iz tastature (ohmski ali kapacitivni)
- aritmetične operacije
- eno samo napajanje
- možnosti razširjanja in preskušanja
- krmiljenje za »triac«.

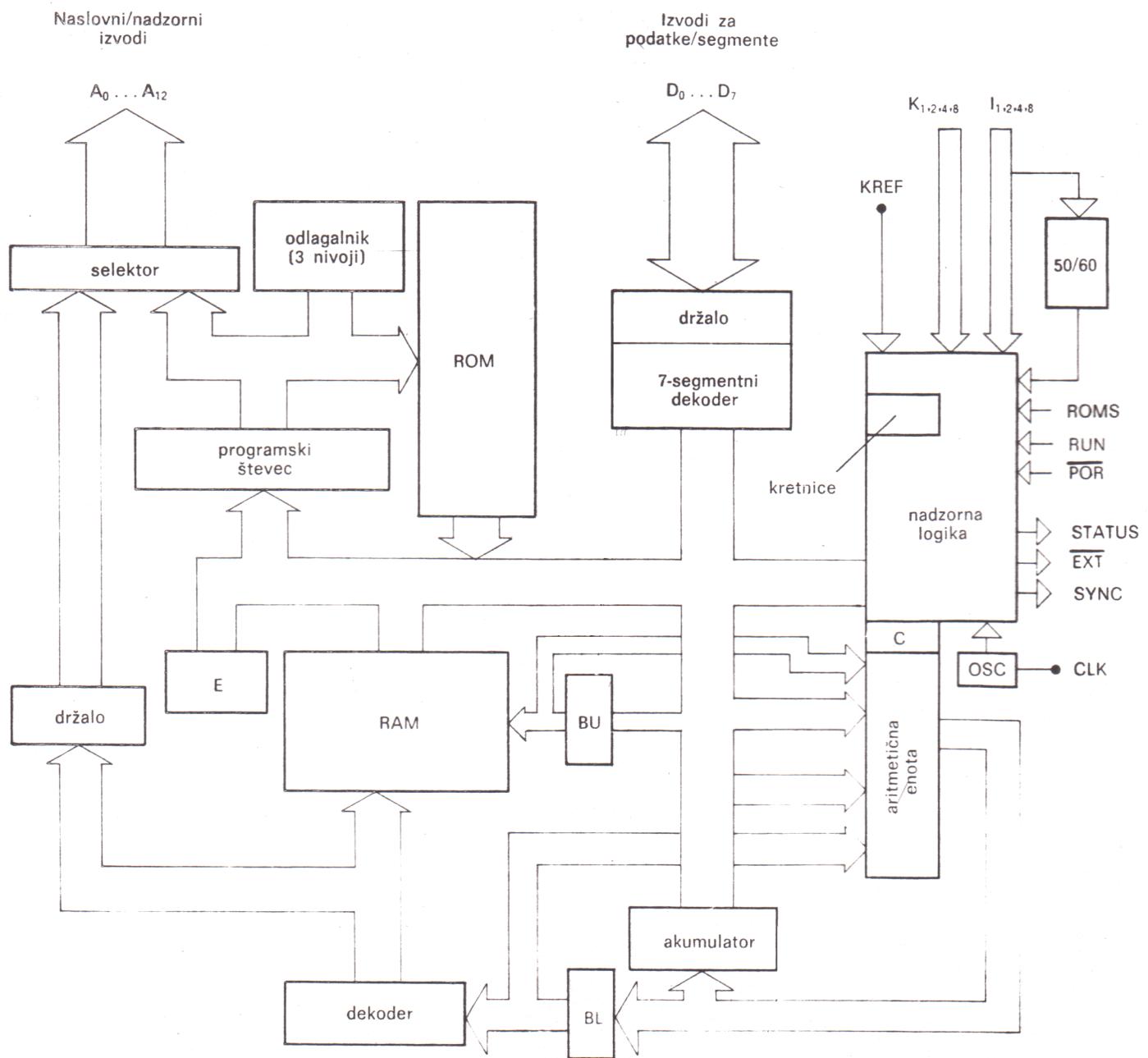
Uporaba mikroračunalnika EMZ 1001 bo prinesla ekonomske in tehnične prednosti zlasti na področju naslednjih izdelkov:

- gospodinjski avtomati
- avtomati za hrano in pijačo

- elektronske tehnice
- elektronske igre in igralni avtomati
- laboratorijski instrumenti
- blagajne
- instrumenti v vozilih
- programirani kalkulatorji
- enote za zbiranje vzorčnih podatkov
- registratorji dogodkov
- preskusni instrumenti
- telekomandne aparature
- elementarna obdelava podatkov.

Mikroračunalnik Iskra EMZ 1001

EMZ 1001 Poenostavljeni blokovni diagram



2.1. Pomenilnik ROM

Naslovni prostor programskega pomenilnika je razdeljen na 8 bank, od katerih vsaka vsebuje 1 K besed. Vsaka banka v pomenilniku ROM je razdeljena na 16 strani in vsaka stran vsebuje 64 ukazov. Programski števec je dolg 13 bitov: spodnjih 6 bitov naslavlja lokacije v okviru ene strani, naslednji 4 biti določajo stran, zadnji trije biti z največjo težo pa določajo banko. Banka 0 je tisti del pomenilnika, ki se nahaja že na sami ploščici (1 K). Ob priključitvi napajanja programski števec starta v banki 0, strani 0, lokaciji 0. Pripravljala regrista PPR in PBR nimata določene vrednosti.

Po izvršitvi vsakega ukaza se programski števec poveča za 1 in tako adresira naslednji ukaz v programu. Izjemo predstavljajo edino skočni ukazi (JMPx, JMSx, RT in RTS). Ukaz JMP skoči na naslov x na tisti strani, kjer je ta ukaz zapisan; ukaz JMSx skoči na naslov x na zadnji strani (stran 15) banke, v kateri je zapisan. Naslove na drugih straneh dosežemo tako, da neposredno pred ukazom JMP oz. JMS zapišemo ukaz za pripravo strani (ukaz PPx, kjer je x številka strani). Ukaz za klicanje subrutin (JMSx) shrani v odlagalniku naslov (4 bite za stran in 6 bitov za lokacijo) ukaza, ki sledi ukazu JMSx. Ko se izvrši ukaz za vračanje (RT ali RTS), se ta naslov prenese iz odlagalnika nazaj v programski števec. Odlagalnik ima globino za 3 nivoje subrutin, tj. hrani lahko tri različne povratne naslove hkrati.

Kadar je ukaz JMPx zapisan na zadnji lokaciji (naslov 63) na neki strani, mora biti pred njim vedno ukaz PPx.

Primeri:

Za skok na naslov LL na isti strani, kjer je zapisan ukaz, uporabimo:

JMP LL.

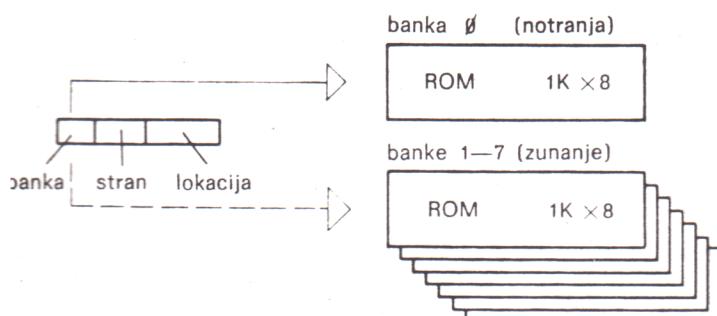
Za skok na naslov LP, ki se nahaja na neki drugi strani, uporabimo:

PP LP/64 ; pripravi naslov strani
JMP LP ; skok

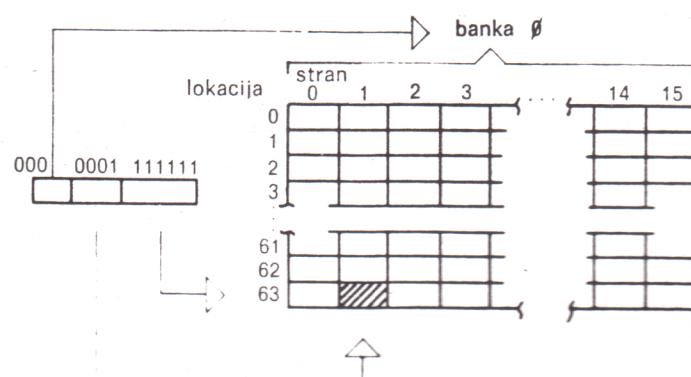
Za skok na naslov LB, ki se nahaja v neki drugi banki, uporabimo:

PP LB/64 ; pripravi stran
PP LB/1024 ; pripravi banko
JMP LB ; skok

sl. 2.1. Naslavljjanje pomenilnih bank



Naslavljanje strani in lokacij znotraj pomenilne banke



2.2. Pomnilnik RAM, notranje vodilo podatkov in kretnice

256 bitov pomnilnika RAM je namenjenih za hranjenje vmesnih rezultatov programa. Pomnilnik je razdeljen na 4 strani, od katerih ima vsaka 16 lokacij. Besede v posameznih lokacijah so dolge 4 bite. Pomnilnik RAM naslavljamo z registroma BL (4 biti) in BU (2 bita). Register BL naslavlja 16 besed v okviru ene strani. Ta register lahko uporabljamo tudi kot splošni register in ga spremiščamo z aritmetičnimi ukazi. Na voljo so ukazi za polnjenje registra BL, izmenjavo z akumulatorjem, nastavitev na same 0, nastavitev na same 1, povečanje za 1 in zmanjšanje za 1. Prek dekoderja se register BL uporablja tudi za formiranje izhodov na vodilu A.

Preostala dva bita naslova v pomnilniku RAM sta določena v registru BU. Operacije v registru BU se izvršijo hkrati z nekaterimi operacijami v drugih registrih. Tako imamo npr. nekaj ukazov (glej poglavje 4.4.), ki poleg osnovne operacije spremenijo tudi register BU v skladu z enačbo

$$\text{novi BU} = \text{stari BU} \oplus Y.$$

Pri tem se Y skriva v argumentu ukaza, \oplus pa pomeni seštevanje po modulu 2 na vsakem bitu posebej:

argument	stara vrednost BU			
	0	1	2	3
0	0	1	2	3
Y	1	0	3	2
2	2	3	0	1
3	3	2	1	0

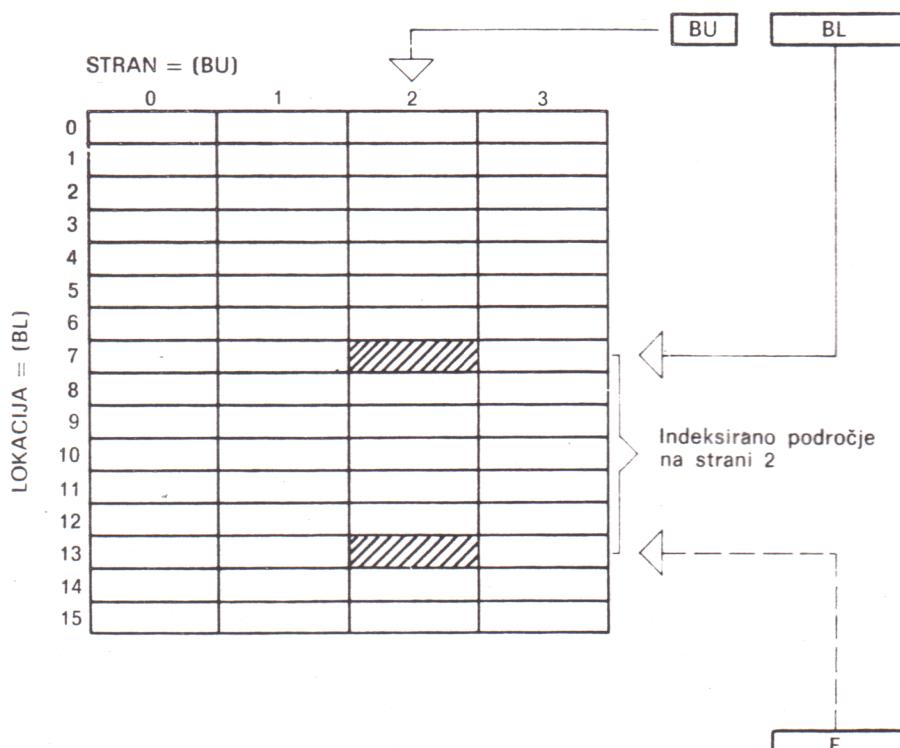
Opomba:

kadar je Y = 0, se BU ne spremeni

V notranosti mikroracunalnika se podatki prenašajo po vodilih R in L. Vsako prenaša po 4 bite, tako da z njima lahko ob istem času prenašamo ukaze z 8 biti. Vhod za pomnilnik RAM se jemlje iz vodila L, izhod pa gre na vodilo R.

V pomnilniku RAM lahko poleg besednih operacij s 4 biti izvajamo tudi operacije nad vsakim posameznim bitom (STM, RSM, SZM). Pri naslavljjanju pomnilnika RAM ima določeno vlogo tudi register E. Služi lahko kot splošni register (tako kot register BL), poleg tega pa je z njim možno nastaviti in nadzorovati vsebine registra BL in s tem indeksiranje v pomnilniku RAM. Register E se polni prek vodila L in je povezan z vodilom R.

Kadar v pomnilniku RAM ne želimo uporabljati posameznih bitov, lahko uporabimo direktno naslovljivki kretnici F1 in F2, ki sta neodvisni od vseh ostalih registrov.



sl. 2.2. Adresiranje pomnilnika RAM

2.3. Akumulator in aritmetično-logična enota

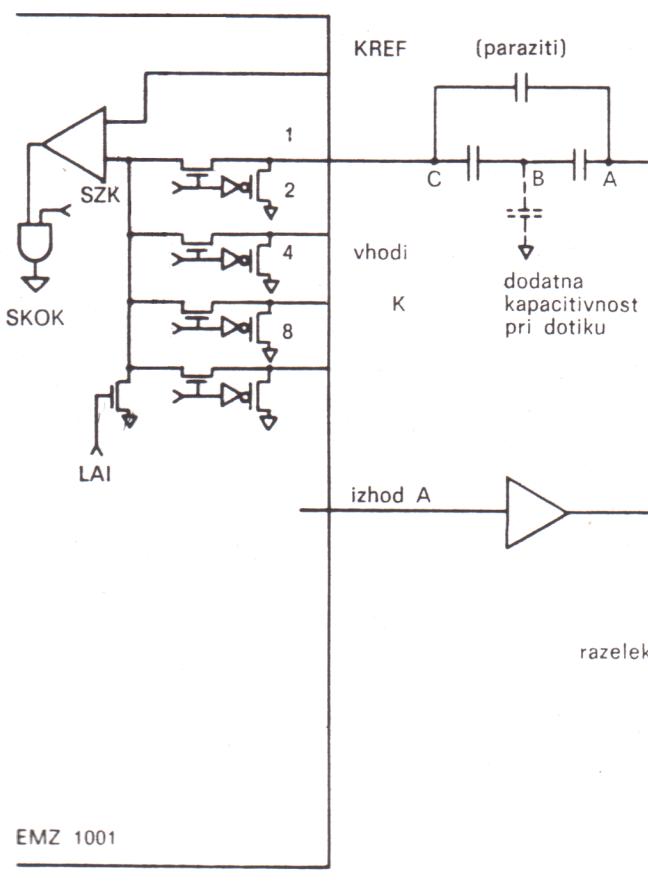
Logične in aritmetične operacije, kot so seštevanje dveh števil s 4 biti, komplementiranje, štetje navzgor, štetje navzdol, primerjanje in konjunkcija, izvajamo v hitrem paralelnem seštevalniku s 4 biti in registrom prepолнитељем C. Kot vidimo na blokovnem diagramu, prihajajo podatki za operacije iz različnih delov vezja glede na to, kateri ukazi se izvajajo. Rezultati se odlažajo v akumulator ACC ali v register BL. Register C lahko uporabljamo tudi samostojno brez operacij v aritmetični enoti.

Akumulator je glavni delovni register v klasičnem smislu, uporabljamo ga za izvor in ponor podatkov pri notranjih operacijah in pri I/O posegih. Podatke sprejema iz vodila R ali neposredno iz aritmetične enote, izhod pa posreduje na vodilo L.

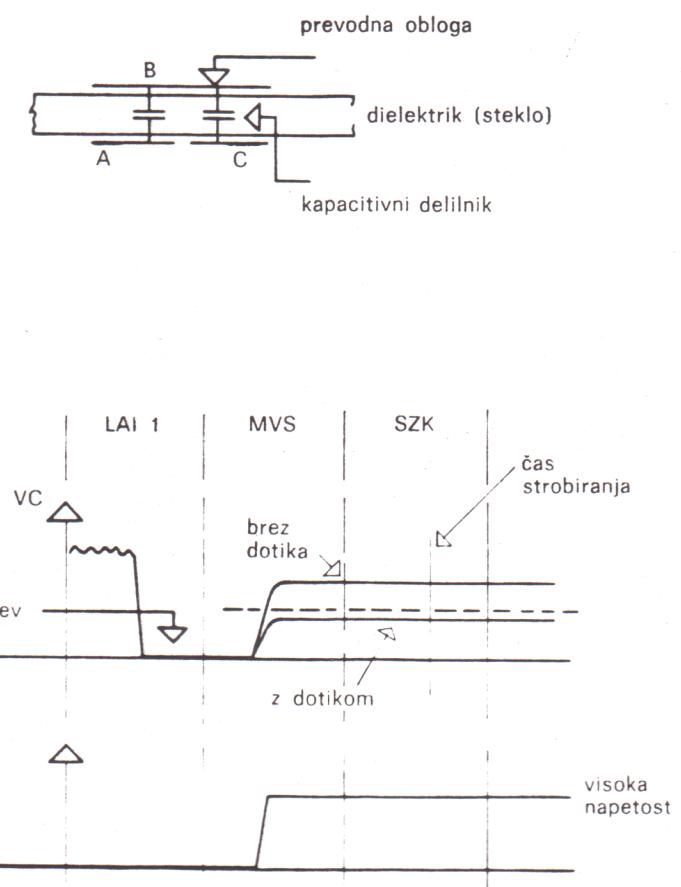
2.4. Vhodi (K, I in D)

Štirje vhodi K in štirje vhodi I ne prenašajo podatkov v notranje registre procesorja, pač pa jih lahko uporabimo za krmiljenje programa prek skočnih ukazov SZK in SIZ. Vhodi se izberejo prek selektorja, ki ga napolni zadnji izvršeni ukaz LAI X. Vsak bit v argumentu X, ki je v stanju »1«, odpre prostot pot istoležnemu vhodu I ali K. Poleg vsakega posameznega vhoda ($X = 1, 2, 4, 8$) lahko izberemo tudi več vhodov hkrati (npr. $X = 7$ izbere vhode 1, 2 in 4).

Vhod I 8 je povezan s časovnikom — števcem omrežza prenos vhodnih podatkov postavimo vodilo D v nevenco 1 Hz in ga ločeno preskušamo prek ukaza SOS. Vhodi K se lahko uporabijo za priključevanje kapacitivnih tipk. Vsak od vhodov K, ki ni izbran, se raz elektri proti masi skozi upornost okoli $40 \text{ k}\Omega$. Kadar je vhod izbran, ima zelo veliko upornost in je povezan z diferenčnim ojačevalnikom (glej sliko 2.3.).



sl. 2.3. Primer priključitve kapacitivne tipke



Vodilo D kot vodilo vhodnih podatkov

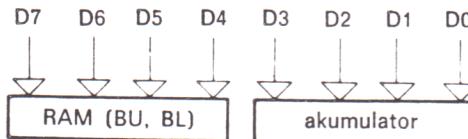
Vodilo D ima tri možna stanja: logični stanji »1« in »0«, ter neutralno stanje velike upornosti (»float«). Običajno za prenos vhodnih podatkov postavimo vodilo D v neutralno stanje, tako da prek njega zunanje enote lahko posredujejo svoja logična stanja. Zunano enoto izberemo s posebnim ukazom (MVS), ki avtomatično postavi vodilo D v neutralni položaj.

Vodilo D je v neutralnem položaju tudi vedno po vklopu napajanja, kadar je izvod RUN vezan na maso in v času T 13 multipleksnega delovanja (glej poglavje 3.1.). Neutralni položaj vodila D je moč regulirati tudi pod vplivom programa z naslednjimi posegi:

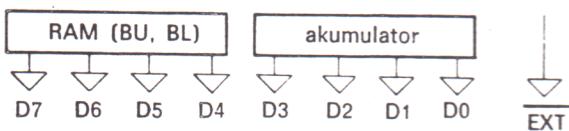
prehod v neutralni položaj : PSL pri BL = 14
(FLOAT)

konec neutralnega položaja : PSH pri BL = 14
(EXIT FLOAT)

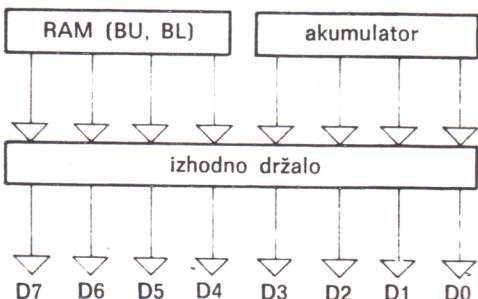
Vhod podatkov programiramo z ukazom INP. Ta ukaz prenese vseh 8 bitov istočasno, bite D0...D3 zapiše v akumulator, bite D4...D7 zapiše v pomnilnik RAM na tisto lokacijo, ki je trenutno določena z registrom BU in BL.



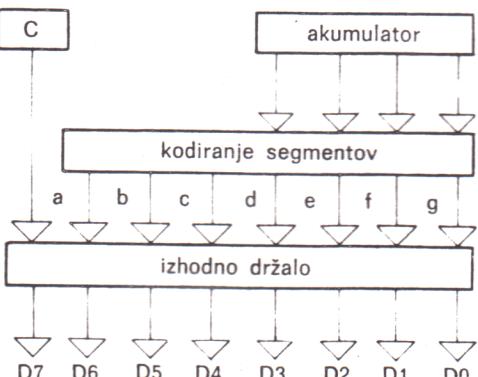
Vhod podatkov pri ukazu INP



Izhod podatkov pri ukazu OUT



Izhod podatkov pri ukazu DISB



Izhod podatkov pri ukazu DISN

2.5. Izhodi (A in D)

Vodilo D kot vodilo izhodnih podatkov

Vodilo D je s svojimi osmimi biti namenjeno predvsem za prenose I/O med standardnimi vezji z 8 biti ter za krmiljenje segmentov pri izpisu rezultatov. Izhodi na vodilu D so vedno vzporedni za vseh 8 bitov.

Z ukazom OUT prenesemo vsebino akumulatorja in pomnilnika RAM na vodilo D, istočasno pa se na izvodu EXT pojavi strobirni impulz, s katerim zagotovimo sprejem podatkov v pravem trenutku. Po izvršenem ukazu OUT se vodilo D vrne v prejšnje stanje.

Za krmiljenje segmentov izpisa se poslužujemo ukazov DISN (display number) ali DISB (display binary). V obeh primerih se izhodni podatki shranijo v držalu D in so od takrat naprej stalno prisotni na vodilu D, dokler s kakim ukazom ne povzročimo prehoda vodila v neutralno stanje.

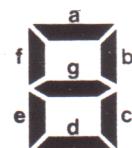
Ukaz DISN pretvori vsebino akumulatorja (4 bit) v 7-bitno segmentno kodo. Osmi bit se prenese iz registra C in ga običajno uporabimo za krmiljenje decimalne pike. Ukaz DISB prenese vsebino akumulatorja in vsebino pomnilnika RAM skupaj, brez segmentnega kodiranja. Primeren je za izpis poljubnih binarnih vzorcev.

Za izhode na držalu D (ukaza DISB in DISN) lahko programsko sprememjamo polarizacijo izhodov. Normalna polarizacija, kakršna se vzpostavi po vklopu napajanja, prenaša podatke kakor so zapisani v ACC in M, oziroma kakor jih prikazuje sl. 2.5. Polarizacijo lahko invertiramo s pomočjo ukaza EUR (glej poglavje 3.3.).

Opozorilo: spremembe v polarizaciji nimajo nobenega vpliva na izhode z ukazom OUT.

Kodiranje segmentov

sl. 2.5. Kodiranje pri ukazu DISN (normalna polarizacija)

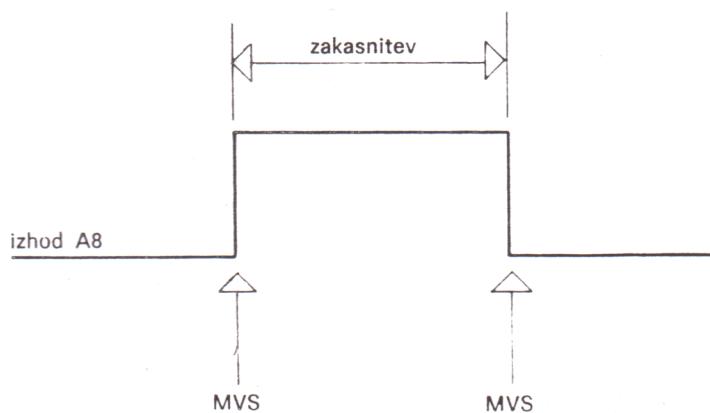


Vsebina akumulatorja	a D6	b D5	c D4	d D3	e D2	f D1	g D0	Izpis
0	1	1	1	1	1	1	0	0
1	0	1	1	0	0	0	0	1
2	1	1	0	1	1	0	1	2
3	1	1	1	1	0	0	1	3
4	0	1	1	0	0	1	1	4
5	1	0	1	1	0	1	1	5
6	1	0	1	1	1	1	1	6
7	1	1	1	0	0	0	0	7
8	1	1	1	1	1	1	1	8
9	1	1	1	1	0	1	1	9
10	1	1	1	0	1	1	1	A
11	0	0	1	1	1	1	1	b
12	1	0	0	1	1	1	0	C
13	0	1	1	1	1	0	1	d
14	1	0	0	1	1	1	1	E
15	1	0	0	0	1	1	1	F

Izhodi na vodilu A

Izhodi na vodilu A so namenjeni za strobiranje izpisa, za strobiranje tipk in splošen nadzor nad periferijo. Izvor podatkov za vodilo A je dekodirani register BL. Tega registra ne uporabljamo le za naslavljjanje v pomnilniku RAM, temveč tudi za generiranje stanj na vodilu A. Ukaza PSH in PSL nastavlja posamezne bite v izhodnem držalu na vrednosti »0« ali »1«. Dekodirana vrednost registra BL pove, kateri bit se nastavlja. Na vodilo A ne moremo neposredno pošiljati binarne vsebine delovnih registrov, pač pa moramo posebej nastaviti vsak bit v izhodnem držalu. Prenos iz registra »master-slave« na vodilo A se izvrši z ukazom MVS za vseh 13 bitov istočasno. Spremembe, ki smo jih individualno nastavljali za vsak bit posebej, se torej prenesajo na izhod vzporedno z enim samim ukazom. Ukaz MVS povzroči tudi prehod vodila D v neutralno stanje. Če ne potrebujemo večjega števila številk v izpisu, lahko krmilimo segmente in katode LED izpisa neposredno iz vodil A in D. V tem primeru vodilo D predstavlja tokovni izvor za tiste segmente, ki so prižgani, vodilo A pa uporabimo kot ponor toka na katodah vsake posamezne številke. Za to funkcijo so zlasti primerni izvodi A0...A3, ki lahko požirajo tudi do 25 mA toka. Če uporabljamo izpis s skupno anodo ali zunanje ojačevalnike, lahko polariteto linij A in D spremenimo s programskimi posegi (glej odstavek 3.3. in opis ukaza EUR).

sl. 2.6. Primer za generiranje impulza na izhodu A8



LAI 8 ; izberi izhod 8

XAB

PSH ; določi »1«

MVS ; prenos za »1«

PSL ; določi »0«

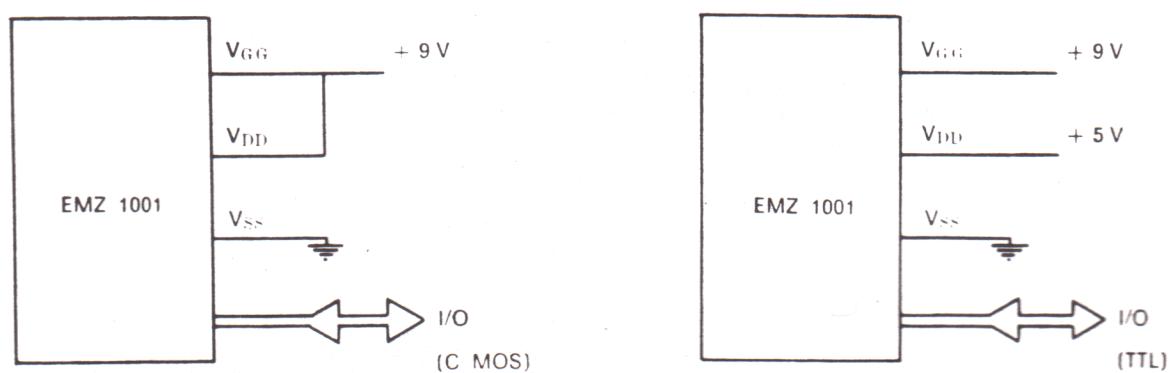
zakasnitev
(drugi ukazi)

MVS ; prenos za »0«

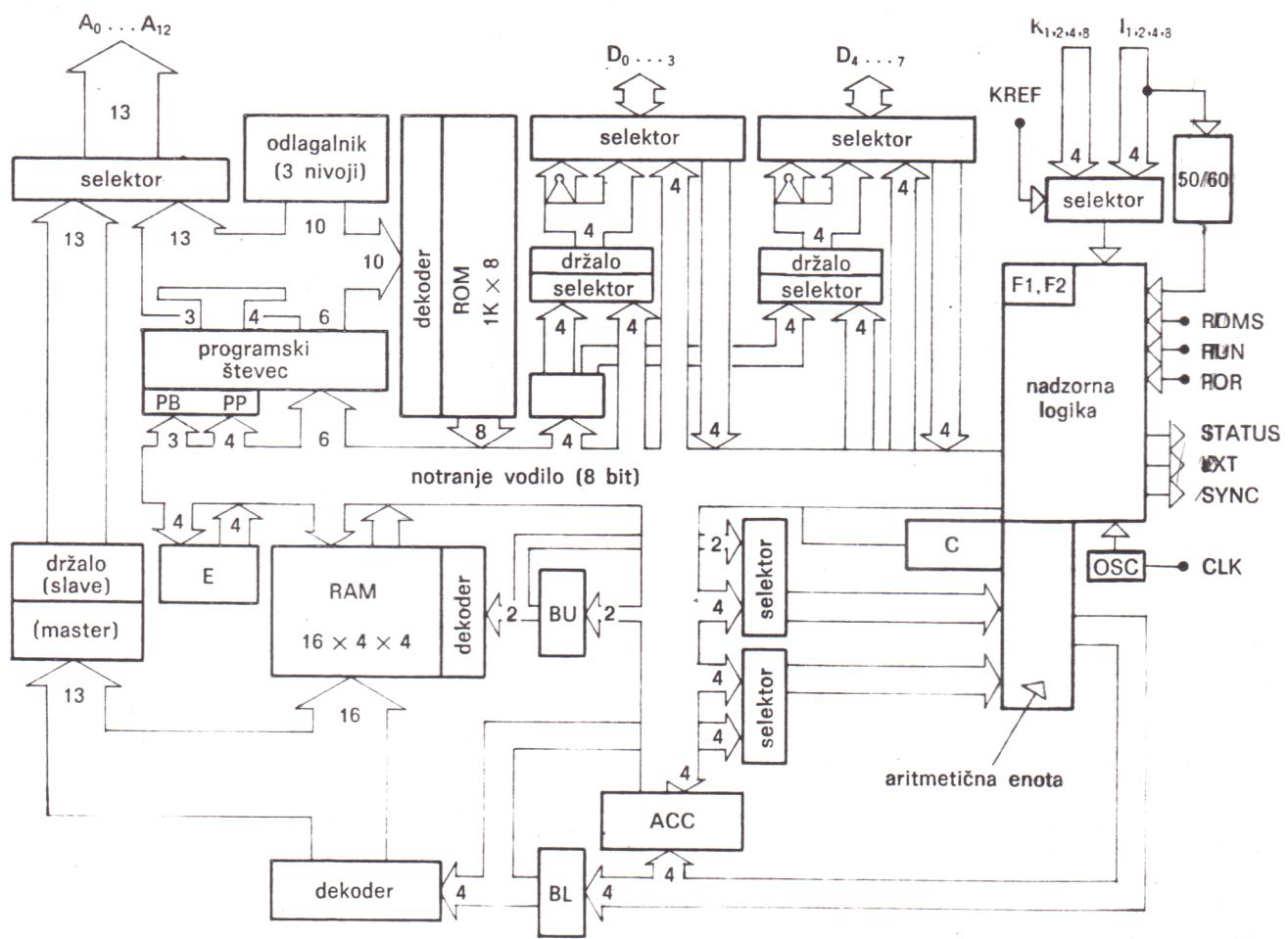
2.6. TTL kompatibilnost

Napajalna napetost mikroračunalnika je + 9 V, kar doča tudi nivoje vhodnih in izhodnih signalov. Če je mikroračunalnik edino vezje v sistemu, to ne predstavlja nobene težave. Povezava z elementi iz družine CMOS prav tako ne predstavlja problema. Če pa želimo skupaj z mikroračunalnikom EMZ 1001 uporabiti TTL kompatibilna vezja, smo primorani znižati izhodne nivoje signalov in omogočiti predpisane minimalne tokovne zmogljivosti. Za tak primer je predviden izvod V_{DD} , ki služi za določanje nivoja izhodnih signalov. Na sliki 2.7. sta prikazana oba načina napajanja mikroračunalnika.

sl. 2.7. Dva načina napajanja



sl. 2.8. Podrobni blokovni diagram Iskra EMZ 1001



3. Posebni načini delovanja EMZ 1001

3.1. Zunanji pomnilnik ROM

Kadar potrebujemo dodatni pomnilnik ROM, lahko notranjemu pomnilniku dodamo do 7 K zlogov zunanjega pomnilnika (banke 1 ... 7). EMZ 1001 pri razširitvi pomnilnika deluje v multipleksnem načinu tako, da stalno preklaplja vodila A in D nad pomnilnik in med I/O funkcije. V prvi polovici cikla (časa T1 in T3) se na vodilu A pojavi programski naslov, vodilo D pa je nevtralnem stanju in lahko prenaša ukaz iz priključenega pomnilnika. V drugi polovici cikla (časa T5 in T7) se vodili A in D vrneta v normalni način delovanja in ukaz se izvrši do konca. Takšno multipleksno delovanje nastane v naslednjih treh primerih:

1. Izvod ROMS priključimo na signal SYNC. V tem primeru deluje samo zunanji pomnilnik (tudi banka 0 je zunanj).
2. Izvod ROMS je povezan z VSS (masa) in programski naslov postane večji od 1023. V tem primeru je banka 0 v notranjem pomnilniku ROM.
3. Izvršen je ukaz PSH pri vrednosti BL = 13. Pomnilnik je določen z izvodom ROMS (gl. 3.4.).

Signal na izvodu SYNC služi kot selektivni impulz za vklop zunanjega pomnilnika. Ta signal je podrobneje prikazan na sl. 4.1.

Opozorimo na tem mestu na to, da za običajno uporabo mikroračunalnika EMZ 1001 ne predvidevamo zunanjega pomnilnika, temveč samo notranjega. V tem primeru naj vezje ne deluje v multipleksnem, temveč v statičnem načinu brez preklapljanja vodil A in D, kot je opisano v poglavju II. Statični način nastopi vedno po vklopu napajanja. Iz multipleksnega načina pridemo nazaj v statičnega, če izvršimo ukaz PSL pri vrednosti BL = 13.

3.2. Preskusni način delovanja

Možnost priključevanja zunanjih pomnilnikov omogoča pri mikroračunalniku EMZ 1001 razmeroma enostavno preskušanje. Vsebino notranjega pomnilnika in delovanje ostalega vezja preskušamo ločeno.

Če povežemo izvod ROMS z invertiranim signalom SYNC, to prepreči izvajanje vseh ukazov in mikroračunalnik deluje v preskusnem načinu. V prvi polovici vsakega cikla (T13) pride na vodilo D ukaz iz notranjega pomnilnika ROM. Zaradi zahtev notranje logike se ta podatek rotira za 4 mesta v desno, kar moramo upoštevati pri gradnji preskusne aparature. V drugi polovici cikla (T57) se programski števec poveča za 1, vodilo D pa zavzame vrednost FF (same »1«). Ker po vklopu napajanja dobi programski števec začetno vrednost 0, lahko pregledamo celotno vsebino notranjega pomnilnika v 1024 ciklih preskusnega delovanja. Ostalo vezje preskusimo s posebnimi preskusnimi programi neodvisno od vsebine notranjega pomnilnika. Kot smo že omenili, je to možno narediti z zunanjim pomnilnikom, če povečamo izvod ROMS s signalom SYNC.

3.3. Časovnik in ukaz EUR

EMZ ima vgrajen časovnik, s katerim lahko omrežno frekvenco 50 ali 60 Hz spremojamo v sekundne impulze. Vhod časovnika je priključen na sponko 18. Signal potuje skozi Schmittov prožilnik in digitalni filter na števec, ki šteje po modulu 50 ali 60. Digitalni filter prepušča samo impulze, ki so daljši od štirih osnovnih ciklov (ca. 18 µs). Izvod števca se postavi v stanje »1« enkrat v sekundi.

Sekundni izvod števca preskusimo z ukazom SOS. Kadar je števec v stanju »1«, dobimo preskok, hkrati pa se stanje vrne na »0«. Števec šteje stalno, ne glede na stanje sekundnega izhoda.

Modul štetja nastavimo programsko z ukazom EUR. Ta ukaz poleg tega služi tudi pri nastavljanju polaritete držala D.

Po vklopu napajanja dobimo normalno polaritetno in modul štetja za 60 Hz.

Ukaz EUR deluje samo v povezavi z vsebinou akumulatorja:

ACC Operacija z EUR

0XX0	60 Hz; invertirani D
0XX1	60 Hz; normalni D
1XX0	50 Hz; invertirani D
1XX1	50 Hz; normalni D

3.4. Posebni vhodi

KREF:

Izvod KREF je priključen na drugi vhod diferenčnega ojačevalnika za signale K1, K2, K4 in K8. Vhod K se nato lahko preskusijo na neko referenčno napetost, ki jo priključimo na KREF.

RUN:

Izvod RUN uporabimo, kadar želimo ustaviti delo procesorja. Služi predvsem za nadzor v koračnem delovanju (single step).

Vezava izvoda RUN

Operacija

»1« (VGG) (notranje povezan prek upora)	normalno izvajanje programa
»0« (VSS)	stanje WAIT, linije D v nevtralnem položaju

ROMS:

Izvod ROMS določa vrsto pomnilnika ROM in prekusno delovanje.

Vezava izvoda ROMS

Operacija

»1« (VGG) (notranje povezan prek upora)	notranji ROM (izključno)
»0« (VSS)	notranji ROM za banko zunanji ROM za banke 1 ...)

SYNC

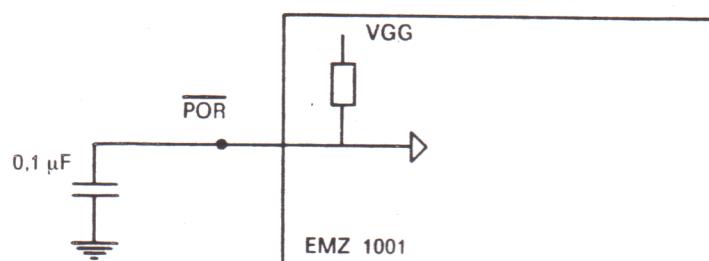
Operacija

SYNC

Operacija

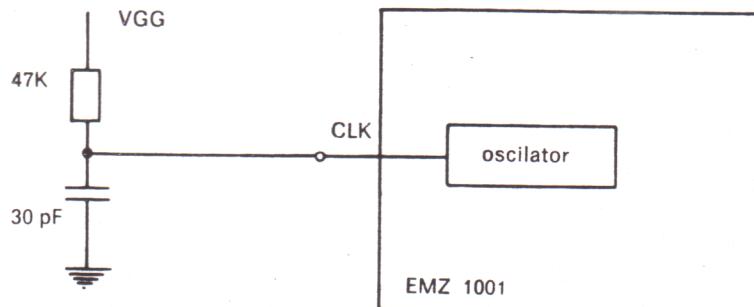
POR:

Izvod POR služi za brisanje ob vklopu napajanja. Običajno ga povežemo s kondenzatorjem.



CLK:

Na izvod CLK priključimo RC konstanto, ki določa osnovno frekvenco oscilatorja.



Stabilnost frekvence $\pm 10\%$

3.5. Posebni izhodi

EXT:

V času T7 pri ukazu OUT dobimo na tem izvodu negativni impulz. Služi nam za strobiranje podatkov iz vodila D. (glej pogl. 2.5.).

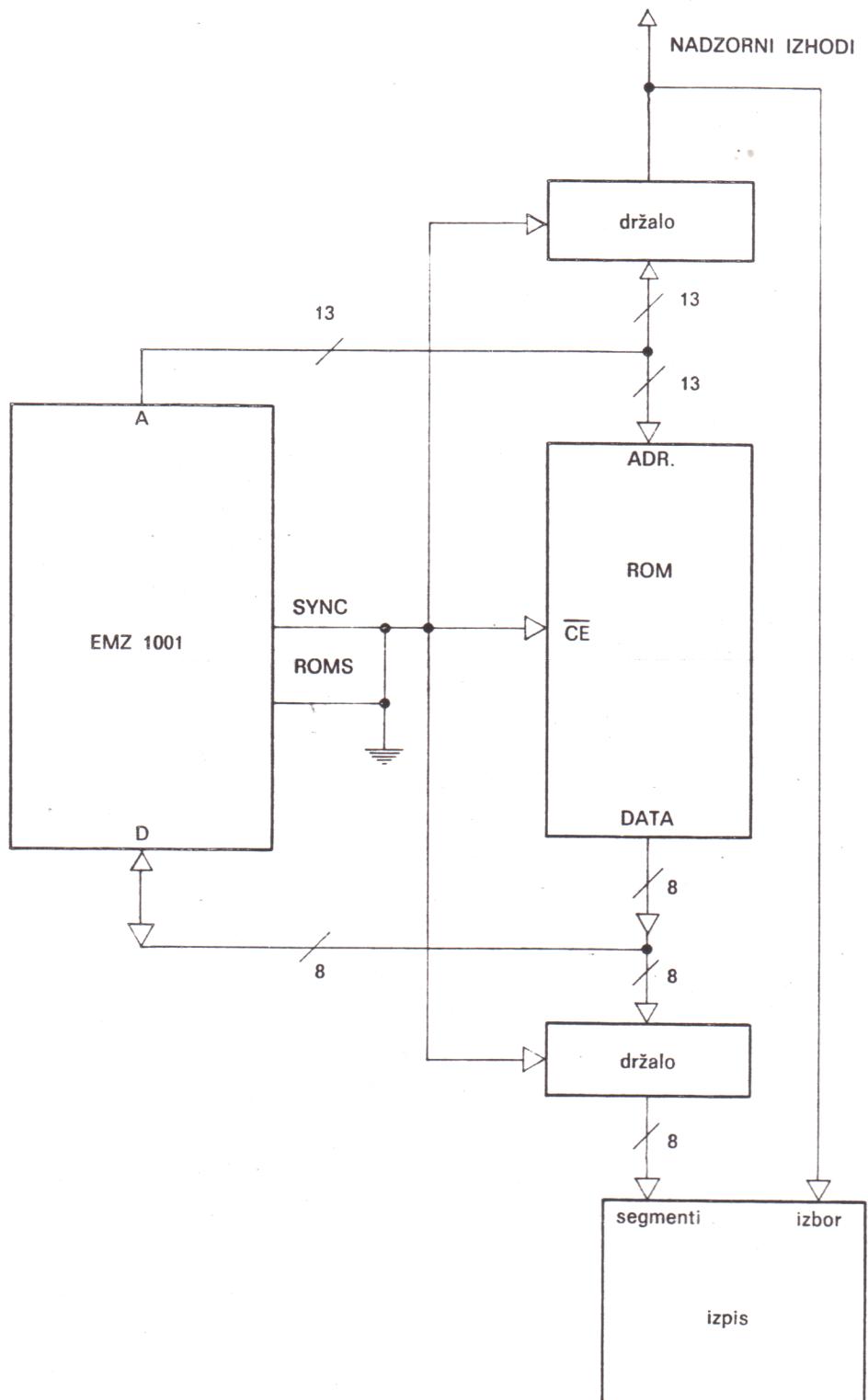
SYNC:

Na izvodu SYNC je stalno prisoten signal pravokotne oblike, ki ga uporabimo za vključevanje zunanjega pomnilnika ROM in za podobne sinhronizacijske probleme. Signal SYNC je v stanju »0« v prvi polovici cikla (T1, T3) in v stanju »1« v drugi polovici cikla (T5, T7).

STATUS:

Izvod STATUS posreduje navzven informacije o stanju mikroračunalnika. V vsaki časovni enoti je prisotna druga informacija.

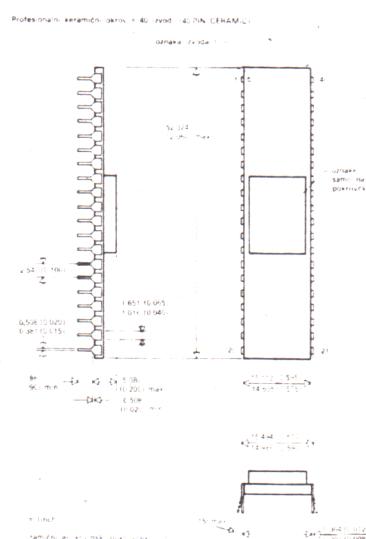
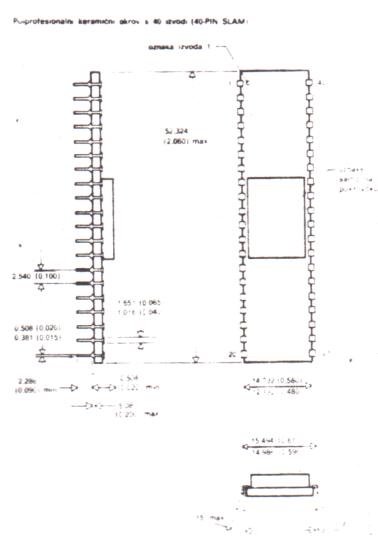
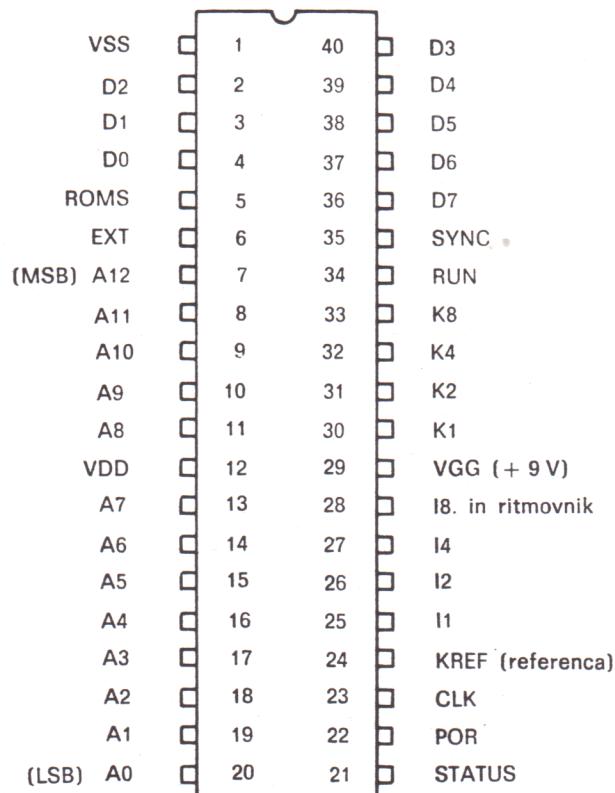
Čas v ciklu	Posredovana informacija	
	STATUS = »1«	STATUS = »0«
T1	vodilo D v nevtralnem stanju	vodilo D ni v nevtralnem stanju
T3	BL = 13	BL ≠ 13
T5	C = 1	C = 0
T7	naslednji ukaz se preskoči	naslednji ukaz se ne preskoči



sl. 3.1. Priključitev zunanjega pomnilnika ROM

4. Tehnični podatki

4.1. Okrov in izvodi



Opis izvodov

Št. izvoda	Ime	Opis
1	VSS	Napajanje — negativni priključek. Običajno ozemljen.
29	VGG	Napajanje — pozitivni priključek. Povezan na + 9 V.
12	VDD	Napajanje izhodov. Običajno povezan na V _{GG} ali na + 5 V.
4—2 40—36	D0 ... D7	Vodilo D. Izvod z ukazi OUT, DISB, DISN. Vhod z ukazom INP. Nevtralno stanje po ukazu MVS, po vklopu napajanja, če RUN = 0 ali če izvršimo PSL z BL = 14. Možnost invertiranja podatkov za ukaza DISB in DISN.
20—13 11—7	A0 ... A12	Vodilo A. Posreduje izhode ukazov PSH, PSL pogojene z MVS. V multipleksnem načinu prenaša tudi programske naslove.
5	ROMS	Izbira programskega pomnilnika.
6	EXT	Sinhronizacijski impulz za prevzem podatkov iz vodila D pri ukazu OUT.
35	SYNC	Osnovni sinhronizacijski signal pravokotne oblike, ki je stalno prisoten.
34	RUN	Nadzor nad izvajanjem programa. »1« za stanje »program teče« in »0« za stanje »stoji«.
22	POR	Inicializacija po vklopu napajanja. Dokler ta izvod držimo v logični »0«, je vezje v stanju brisanja.
23	CLK	Vhod oscilatorja. Običajno priključen na RC člen, možna pa je tudi povezava s kristalom.
21	S	Statusni izhod. Posreduje informacije o notranjih stanjih procesorja.
24	KREF	Referenčni vhod za vhode K. Običajno povezan + 2,7 V prek zunanjega uporavnega delilnika.
30—33	K1, K2, K4, K8	Vhodi K, katere lahko preskusimo z ukazom ŠZK. Želeni vhod izberemo z ukazom LAI. Neizbrani vhodi so ozemljeni.
25—28	I1, I2, I4, I8	Vhodi I, katere lahko preskusimo z ukazom SZI. Želeni vhod izberemo z ukazom LAI. Vsi vhodi so povezani na VGG prek upora ca. 400 kΩ. I8 je povezan tudi s časovnikom.

4.2. Električne specifikacije

Absolutne omejitve: (Vse napetosti navedene glede na VSS)

Temperatura skladiščenja	— 55 do + 125 ⁰ C
Delovna temperatura	0 do + 70 ⁰ C
Maks. pozitivna napetost	+ 18 V
Maks. negativna napetost	— 0,3 V
Napajalni tok (opomba 1)	75 mA
Poraba moči	0,5 W

Karakteristične vrednosti:

(VSS = 0V, VGG = 9 ± 0,5 V, VDD = 5 V*, (T_A = 25 °C)
(pozitivna logika)

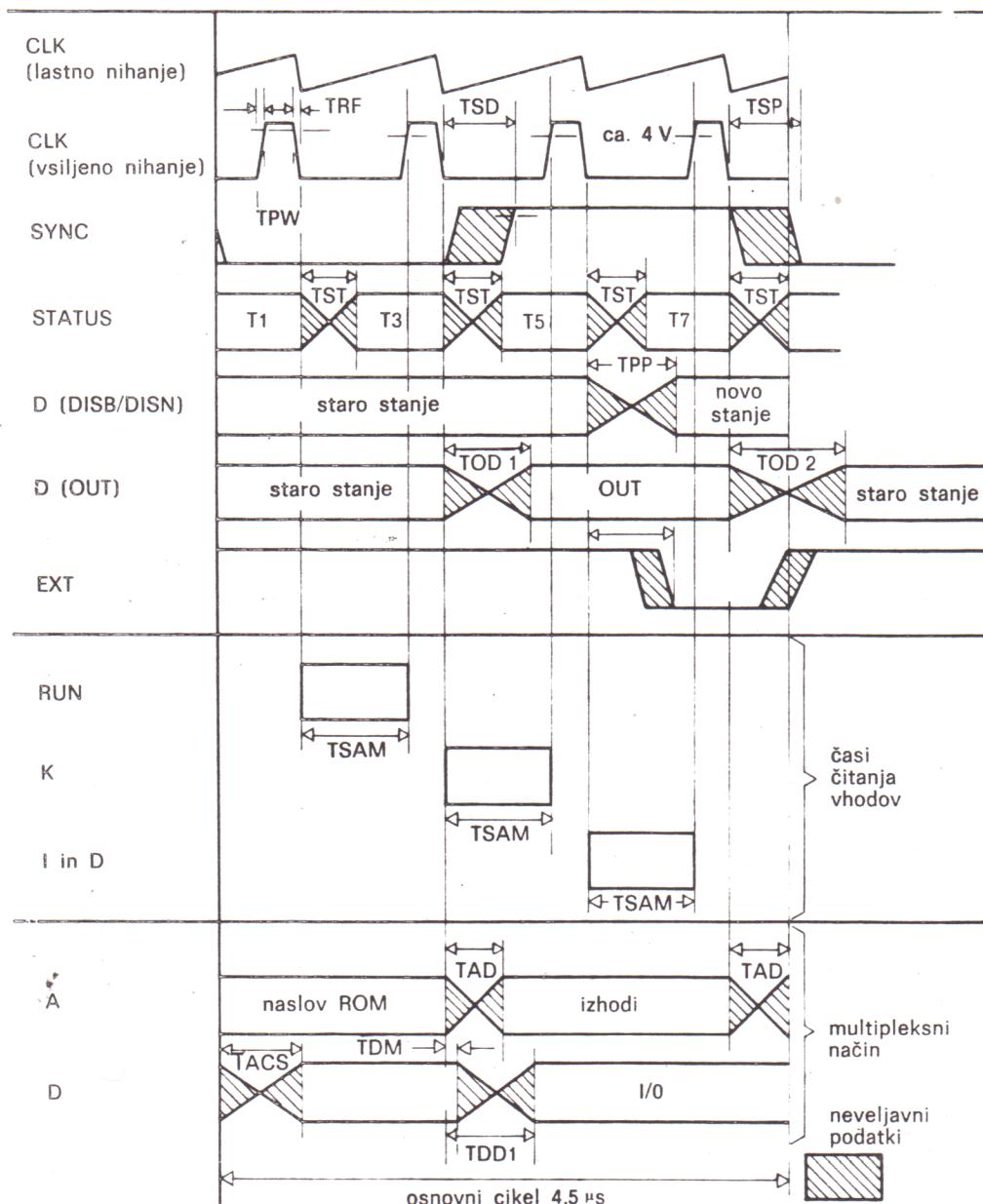
Parametri	Pogoji	Min.	Maks.	Enota
Vhodi K1, K2, K4, K8				
logična »0«		0	KREF 0,5 V	V
logična »1«		KREF + 0,5 V	VGG	V
Vhod KREF		0,28 VGG	0,32 VGG	
Vhodi I1, I2, I4, I8	(opomba 2)			
POR in časovnik				
logična »0«		0	0,5	V
logična »1«		5,3	VGG	V
Vhodi ROMS, RUN				
logična »0«		0	0,8	V
logična »1«		3,5	VGG	
Vhodi D0 ... D7				
logična »0«		0	0,8	V
logična »1«	{ program podatki }	4,5 3,5	VGG	
Izhodi A0 ... A3				
logična »0«	I = + 25 mA	0	0,8	V
logična »1«	I = — 5 mA	3,5	VDD*	V
Izhodi A4 ... A12				
EXT, SYNC in S				
logična »0«	I = + 5 mA	0	0,6	V
logična »1«	I = — 5 mA	3,5	VDD*	V
Izhodi D0 ... D7				
logična »0«	I = + 12 mA	0	1	V
logična »1«	I = — 5 mA	3,5	VDD*	V
Napajalni tok IGG	tip 28	50		
Napajalni tok IDD		odvisen od bremen*		mA

Opomba 1: Odvisen od bremen. Skupna poraba na izhodih naj ne bo večja od 150 mW.

Opomba 2: Vsak od teh vhodov ima znotraj vezja upor (ca. 100 kΩ) povezan z VGG. Signal gre na Schmittov prožilnik s histerezo okoli 1 V (0 → 1 pri ≈ 4 V, 1 → 0 pri ≈ 3 V).

* VDD lahko povežemo z VGG, če želimo eno samo napajanje.

sl. 4.1. Časovni diagram



Simbol	Min.	Tip	Maks.	Enota
TPW	250	300	350	ns
TRF	25	50	75	ns
TSD			550	ns
TST			425	ns
TDD			725	ns
TOD1			600	ns
TOD2			925	ns
TED	200		550	ns
TSAM		700		ns
TAD*			425	ns
TACS*			700	ns
TDH*	150			ns
TDD1*			600	ns

* = multipleksni način

4.3. Kodiranje ukazov

Mnemonika	Cikli	Binarna koda	Heksadecimalna koda
NOP	1	00	00000000
RT	1	02	00000010
RTS	1 + n	03	00000011
PSH	1	04	00000100
PSL	1	05	00000101
AND	1	06	00000110
SOS	1 + n	07	00000111
SBE	1 + n	08	00001000
SZC	1 + n	09	00001001
STC	1	0A	00001010
RSC	1	0B	00001011
LAE	1	0C	00001100
XAE	1	0D	00001101
INP	1	0E	00001110
EUR	1	0F	00001111
CMA	1	10	00010000
XABU	1	11	00010001
LAB	1	12	00010010
XAB	1	13	00010011
ADCS	1 + n	14	00010100
XOR	1	15	00010101
ADD	1	16	00010110
SAM	1 + n	17	00010111
DISB	1	18	00011000
MVS	1	19	00011001
OUT	1	1A	00011010
DISN	1	1B	00011011
SZM B	1 + n	1C do 1F	000111XX
STM B	1	20 do 23	001000XX
RSM B	1	24 do 27	001001XX
SZK	1 + n	28	00101000
SZI	1 + n	29	00101001
RF1	1	2A	00101010
SF1	1	2B	00101011
RF2	1	2C	00101100
SF2	1	2D	00101101
TF1	1 + n	2E	00101110
TF2	1 + n	2F	00101111
XCI Y*	1 + n	30 do 33	001100XX
XCD Y*	1 + n	34 do 37	001101XX
XC Y*	1	38 do 3B	001110XX
LAM Y*	1	3C do 3F	001111XX
LBZ Y	1	40 do 43	010000XX
LBF Y	1	44 do 47	010001XX
LBE Y	1	48 do 4B	010010XX
LBEP Y	1	4C do 4F	010011XX
ADIS X	1 + n	50 do 5F	0101XXXX
PP X*	1	60 do 6F	0110XXXX
LAI X	1	70 do 7F	0111XXXX
JMS X	2	80 do 8F	10XXXXXX
JMP X	1	C0 do FF	11XXXXXX

* Argumenti morajo biti invertirani, zbirni prevajalnik AP jih avtomatično invertira.
n = število preskočenih ukazov

Opomba: Koda 00000001 je rezervirana za uporabo v podpornih programih razvojnega sistema.

4.4. Nabor ukazov EMZ 1001

Medregisterski ukazi

Mnemonika	Operand	Opis	Predstavitev
LAB		Napolni ACC z vsebino BL	$BL \rightarrow ACC$
LAE		Napolni ACC z vsebino E	$E \rightarrow ACC$
LAI	X	Napolni ACC s konstanto* Izberi vhode K in vhode I	$X \rightarrow ACC$ $0 \leq X \leq 15$ $X \rightarrow \text{vhodna maska}$
LBE	Y	Napolni BL z vsebino E, BU pa s konstanto Y	$E \rightarrow BL$ $Y \rightarrow BU$ $0 \leq Y \leq 3$
LBEP	Y	Napolni BL z vsebino E, povečano za 1, BU pa s konstanto Y*	$E + 1 \rightarrow BL$ $Y \rightarrow BU$ $0 \leq Y \leq 3$
LBF	Y	Napolni BL z 1111, BU pa s konstanto Y*	$15 \rightarrow BL$ $Y \rightarrow BU$ $0 \leq Y \leq 3$
LBZ	Y	Napolni BL z 0000, BU pa s konstanto Y*	$0 \rightarrow BL$ $Y \rightarrow BU$ $0 \leq Y \leq 3$
XAB		Zamenjamo vsebini ACC in BL	$BL \leftrightarrow ACC$
XABU		Zamenjamo vsebini ACC (biti 0, 1) in BU. ACC (2, 3) se ne spremeni	$BU \leftrightarrow ACC \{0, 1\}$
XAE		Zamenjaj vsebini ACC in E	$E \leftrightarrow ACC$
STC		Napolni C z »1«	$1 \rightarrow C$
RSC		Napolni C z »0«	$0 \rightarrow C$
SFI		Napolni kretnico F1 z »1«	$1 \rightarrow F1$
RFI		Napolni kretnico F1 z »0«	$0 \rightarrow F1$
SF2		Napolni kretnico F2 z »1«	$1 \rightarrow F2$
RF2		Napolni kretnico F2 z »0«	$0 \rightarrow F2$

* Prvi ukaz po vklopu napajanja ne sme biti tipa LB ali LAI. Kadar program vsebuje zaporedje ukazov LAI ali (LB-), se v tem zaporedju vrši samo prvi ukaz.

Ukazi za delo s pomnilnikom RAM

V vseh navedenih ukazih uporabljamo kratico M za označitev tiste lokacije v pomnilniku RAM, ki jo določata registra BU in BL.

Mnemonika	Operand	Opis	Predstavitev
LAM	Y	Napolni ACC z M, nato spremeni BU	$M \rightarrow ACC$ $BU \oplus Y \rightarrow BU$
XC	Y	Zamenjaj vsebini ACC in M in nato spremeni BU	$M \leftarrow ACC$ $BU \oplus Y \rightarrow BU$
XCI	Y	Zamenjaj vsebini ACC in M, nato povečaj BL za 1 in spremeni BU. Preskoči, če po povečanju velja $BL = 0^*$	$M \leftarrow ACC$ $BL + 1 \rightarrow BL$ $BU \oplus Y \rightarrow BU$ IF $BL = 0$ SKIP *
XCD	Y	Zamenjaj vsebini ACC in M, nato zmanjšaj BL za 1 in spremeni BU. Preskoči, če po zmanjšanju velja $BL = 15^*$	$M \leftarrow ACC$ $BL - 1 \rightarrow BL$ $BU \oplus Y \rightarrow BU$ IF $BL = 15$ SKIP *
STM	B	Napolni izbrani bit v celici M z »1«	$1 \rightarrow M \text{ (bit B)}$ $0 \leq B \leq 3$
RSM	B	Napolni izbrani bit v celici M z »0«	$0 \rightarrow M \text{ (bit B)}$ $0 \leq B \leq 3$

Spremembe registra BU potekajo z operacijo XOR:

	stari BU			
	0	1	2	3
0	0	1	2	3
1	1	0	3	2
novi BU	2	2	3	0
	3	3	2	1
				Y

Posamezni biti v celici M so izbrani takole:

MSB	LSB
B = 3	B = 2 B = 1 B = 0

Aritmetični in logični ukazi

V vseh navedenih ukazih uporabljamo kratico M za označitev tiste lokacije v pomnilniku RAM, ki jo določata registra BU in BL.

Mnemonika	Operand	Opis	Predstavitev
ADD		Seštej ACC in M. C se ne spremeni.	$M + ACC \rightarrow ACC$
ADCS	X	Seštej ACC, M in C. Preskoči, če je vsota manjša od 16.*	$M + ACC + C \rightarrow ACC, C$ IF $SUM \leq 15$ SKIP
ADIS	X	Seštej ACC in konstanto X. C se ne spremeni. Preskoči, če je vsota manjša od 16.*	$X + ACC \rightarrow ACC$ IF $SUM \leq 15$ SKIP
AND		Logični IN med ACC in M.	$M \& ACC \rightarrow ACC$
XOR		Logični eksluzivni ALI med ACC in M.	$M \oplus ACC \rightarrow ACC$
CMA		Logični eniški komplement ACC.	$15 - ACC \rightarrow ACC$

* Kadar pride do preskoka, se preskoči prvi naslednji ukaz. Če preskočimo ukaz PP, se preskoči še naslednji ukaz.

Ukazi za preskoke*

Mnemonika	Operand	Opis	Predstavitev
SAM		Preskoči, če ACC = M	IF ACC = M SKIP
SZM	B	Preskoči, če M (bit B) = 0	IF M (bit B) = 0 SKIP $0 \leq B \leq 3$
SBE		Preskoči, če BL = E	IF BL = E SKIP
SZC		Preskoči če C = 0	IF C = 0 SKIP
SOS		Preskoči, če je sekundni izhod časovnika enak 1. Po preskoku vrni izhod časovnika na 0.	IF SF = 1 SKIP $0 \rightarrow SF$
SZK		Preskoči, če je vhod K enak 0. Vhodi K so izbrani z argumentom zadnjega izvršenega ukaza LAI. Neizbrani vhodi so zvezani proti masi. Za preskok je potrebno, da so izbrani vhodi K \leq KREF.	IF $K_{8,4,2,1} \leq KREF$ SKIP
SZI		Preskoči, če je vhod I enak 0. Vhodi I so izbrani z argumenta zadnjega izvršenega ukaza LAI.	IF $I_{8,4,2,1} = 0$ SKIP
TF1		Preskoči, če je kretnica F1 = 1	IF F1 = 1 SKIP
TF2		Preskoči, če je kretnica F2 = 1	IF F2 = 1 SKIP

* Kadar pride do preskoka, se preskoči prvi naslednji ukaz. Ce preskočimo ukaz PP, se preskoči še naslednji ukaz.

Ukazi za usmerjanje programa

Mnemonika	Operand	Opis	Predstavitev
PP	Y	Pripravi naslednjo stran če predhodni ukaz ni bil PP. Pripravi naslednjo banko, če je predhodni ukaz bil PP.	$Y \rightarrow (\text{PPR/PBR})$
JMP	X	Skoči na lokacijo X na isti strani, kjer je napisan ukaz. Če je predhodni ukaz bil PP, spremeni registra za stran in banko z vsebino pripravljenih registrov. Uporabljajo se naslednje 3 sekvence: <ol style="list-style-type: none"> 1. Skok na isti strani JMP LL 2. Skok na neko stran v isti banki PP LP/64 JMP LP 3. Skok v drugo banko PP LB/64 PP LB/1024 JMP LB 	$X \rightarrow \text{LR}$ $0 \leq X \leq 63$ $X \rightarrow \text{LR}$ $\text{PPR} \rightarrow \text{PR}$ $\text{PBR} \rightarrow \text{BR}$
JMS	X	Skoči na lokacijo X na strani 15. Shrani PR + 1 in LR + 1 v odlagalnik. Če je predhodni ukaz bil PP, spremeni registra za stran in banko z vsebino pripravljenih registrov. Uporabljata se dve sekvenci: <ol style="list-style-type: none"> 1. Klic subroutine na strani 15 JMS LS 2. Klic subroutine na poljubni strani v isti banki PP LP/64 JMS LP 	$\text{LR} + 1 \rightarrow \text{L STACK}$ $\text{PR} \rightarrow \text{P STACK}$ $X \rightarrow \text{LR}$ $15 \rightarrow \text{PR}$ $\text{LR} + 1 \rightarrow \text{L STACK}$ $\text{PR} \rightarrow \text{P STACK}$ $X \rightarrow \text{LR}$ $\text{PPR} \rightarrow \text{PR}$ $\text{PBR} \rightarrow \text{BR}$
RT		Vračanje iz subroutine	$\text{L STACK} \rightarrow \text{LR}$ $\text{P STACK} \rightarrow \text{PR}$
RTS		Vračanje iz subroutine in preskok prvega ukaza v glavnem programu.*	$\text{L STACK} \rightarrow \text{LR}$ $\text{P STACK} \rightarrow \text{PR}$ SKIP
NOP		Neoperativna koda.	

Kadar pride do preskoka (tudi pri ukazih XCI, XCD, ADCS in ADIS), se preskoči prvi naslednji ukaz. Če preskočimo ukaz PP, se preskoči še naslednji ukaz.

Primer:

```

=====
A = M
SAM
PP  LB/64    se preskoči
PP  LB/1024   se preskoči
JMP LB       se preskoči
ADD          se izvrši
=====

```

Opozorila:

1. Register BR se ne shranjuje pri klicanju subroutine.
2. Če je ukaz JMPX zapisan na zadnji lokaciji neke strani, bo skočil že na naslednjo stran, ne pa na lastno stran. Temu se lahko izognemo, če ga opremimo s predhodnim ukazom PP X/64.

Ukazi za vhode in izhode

5. Razvijanje aplikativnih programov

Programska oprema za posamezne aplikacije mikro- računalnika EMZ 1001 se pripravlja na podpornem računalniškem sistemu, ki omogoča urejanje in shranjevanje programov, prevajanje in preverjanje (simulacija). Prevedeni program se v fazi preizkušnje zapiše v električno programirani pomnilnik PROM, katerega lahko tudi večkrat zbrišemo in popravimo vsebino s ponovnim programiranjem.

Zaradi možnosti multipleksnega delovanja nudi mikrorračunalnik EMZ 1001 veliko prednost pri preizkušanju programov za prototip. V ta namen je zgrajena posebna enota — statični emulator — s katero preverimo lastnosti programa tako, kot da bi bil program že zapisan v notranjem pomnilniku mikrorračunalnika. Statični emulator vsebuje mikrorračunalnik EMZ 1001, pomnilnik

PROM z uporabnikovim programom, pomnilni vmesnik in 40-žilni kabel s priključkom v obliki mikroračunalnika. Za preverjanje programa zadostuje, da priključek emulטורja spojimo z vznožkom, v katerem bo po končanem razvoju mikroračunalnik EMZ 1001 z notranjim uporabniškim programom.

Na osnovi primerno podanih specifikacij bo Iskra — Mikroelektronika lahko pogodbeno prevzela izdelavo programov in prototipov za posamezne porabnike.

Tolmač manj znanih pojmov

- Akumulator** (accumulator): Osrednji delovni register, ki posreduje informacije pri izvrševanju ukazov.
- Aritmetična enota** (ALU): Enota, v kateri se izvajajo aritmetične in logične operacije.
- Beseda** (word): Skupina bitov.
- Bit** (bit): Osnovna informacijska enota. Zavzame lahko dve stanji, ki jih označujemo z »0« in »1«.
- Banka** (bank): Pomnilna enota z določenim številom besed.
- Celica**: Beseda v pomnilniku RAM, ki je določena s svojim naslovom.
- Časovnik** (timer): Generator časovnih funkcij.
- Dekodiranje** (decoding): Razpoznavanje informacije.
- Držalo** (latch): Register, ki služi za zadrževanje informacije.
- Indeksiranje** (indexing): Vrsta naslavljanja, pri katerem naslov povečujemo ali zmanjšujemo z neko konstanto (v našem primeru z 1).
- Incializacija**: Nastavljanje začetnih pogojev ob vklopu napajanja.
- I/O**: Prenosi informacij med računalnikom in zunanjimi enotami.
- Kodiranje** (coding): Šifriranje informacije v neko obliko binarnega zapisa.
- Kompatibilnost TTL** (TTL compatibility): Sposobnost združevanja z vezji iz družine TTL.
- Kretnica** (flag): Majhen register (v našem primeru 1 bit), ki služi za pomoč pri uravnavanju programskega teka.
- Lokacija** (location): Beseda v pomnilniku, ki je določena s svojim naslovom.
- Odlagalnik** (stack): Posebno organizirani pomnilnik vrste RAM, iz katerega se informacije berejo na osnovi vrstnega reda, v katerem smo jih zapisali.
- Procesna enota** (CPU): Del računalniškega sistema, ki skrbi za izvrševanje posameznih ukazov.
- Programski števec (program counter)**: Števec, ki šteje programske korake in s tem ustvarja naslove ukazov v pomnilniku ROM.
- RAM** (RAM): Pomnilnik, v katerega lahko vpisujemo in beremo zapisane informacije.
- Register** (register): Majhen pomnilnik vrste RAM, ki vsebuje eno samo besedo informacije.
- ROM** (ROM): Pomnilnik s stalno zapisano vsebino, ki jo lahko le beremo, ne moremo pa je spremnijati.
- Selektor** (selector): Vezje, ki lahko izbira informacijo med priključenimi vodili in jo prenese na svoj izhod.
- Strobiranje** (strobing): Otipavanje informacije v nekem danem trenutku.
- Subrutina** (subroutine): Zaključena programska enota, katero kličemo iz drugih delov programa.
- Vodilo** (bus): Skupina vodnikov in krmilnih stopenj, ki skrbijo za prenos informacije med posameznimi enotami.
- Zlog** (byte): Beseda z dolžino 8 bitov.