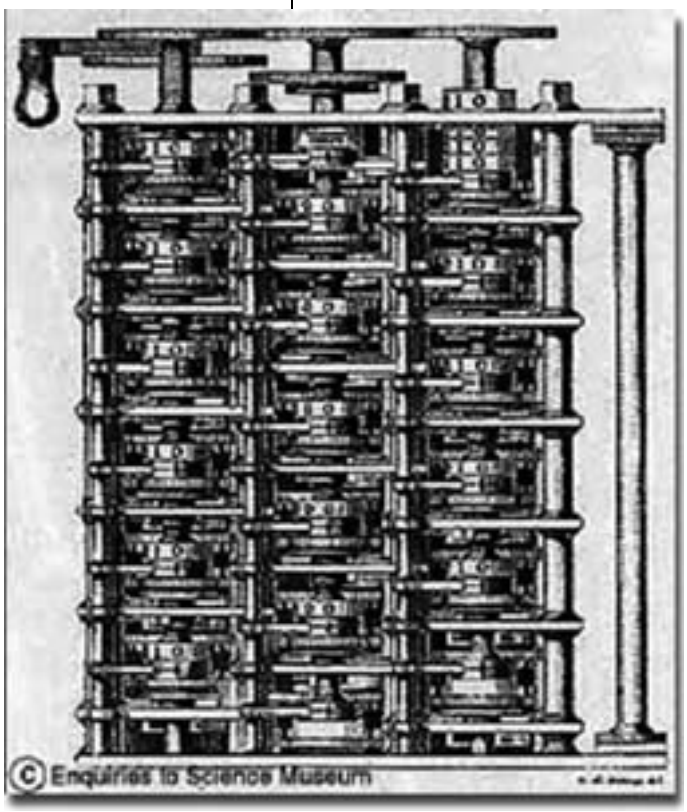
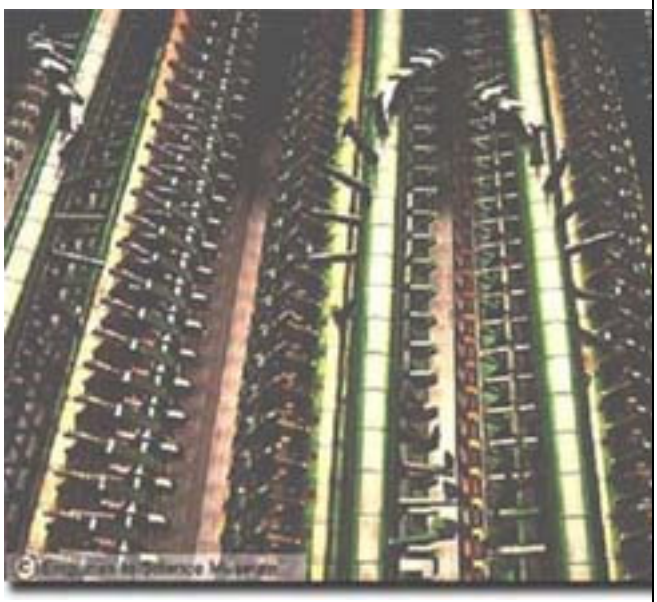
UVOD

Kratki pregled bitnih događaja koji su doveli do razvoja mikroprocesorske tehnologije:

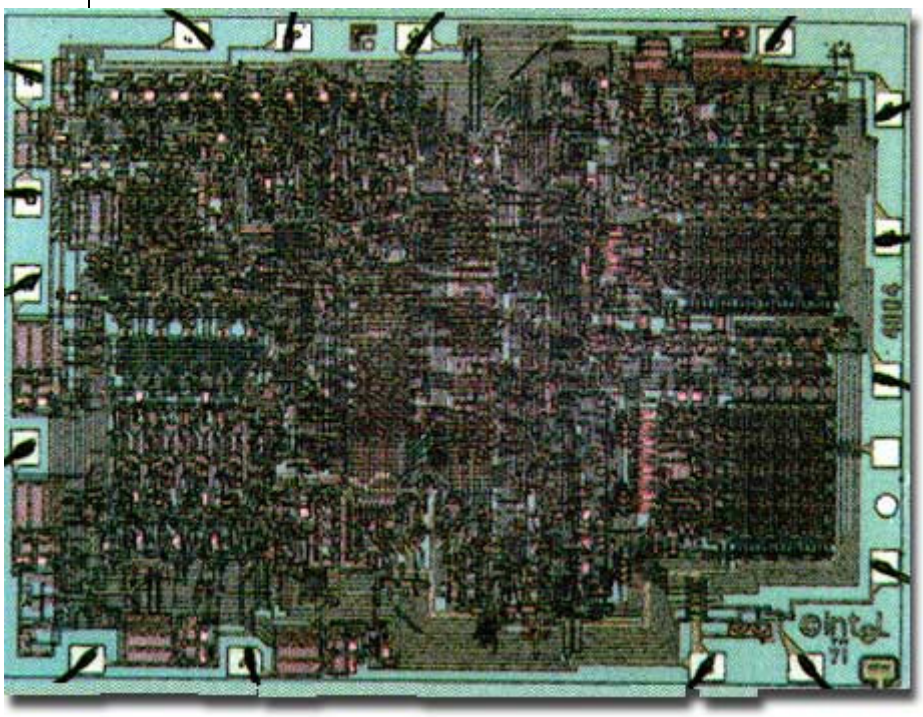
3000 p.n.e.	Abacus – Kina
1642	Blaise Pascal (francuski matematičar i filozof) izumio je mehanički stroj za zbrajanje – do 8 dekadskih znamenaka. Zbog svoje ograničenosti nije bio u široj primjeni.
1822 – 1833	Charles Babbage – Diferencijalni stroj (1822) Nikada nije realiziran, ali je postavio temelje za sljedeći, također nedovršen projekt analitičkog stroja (oko 1833). Ovi projekti imali su sve komponente suvremenih računala iako su zamišljeni kao potpuno mehaničke naprave.





1939	Britanska tajna služba angažira tim matematičara i drugih znanstvenika kako bi razvili stroj za razbijanje Njemačke ENIGMA-e. Rezultat je COLOSSUS, prvo električno digitalno računalo (1943).
1941	Njemački inženjer Konrad Zuse između 1938 i 1941 razvija tri računska stroja. Posljednji Z-3 može se smatrati prvim potpuno upotrebljivim digitalnim računalom. Koristio je binarni brojevni sustav i mogao je izvoditi operacije s brojevim u floating-point zapisu.
1944	Howard Aiken - Mark I – prvo elektroničko računalo zasnovano na relejima (dužina 17 m, visina skoro 3 m). Ovo je prvo konkretno ostvarenje Babbageovog projekta analitičkog stroja.

1946	University of Pennsylvania u suradnji s Vladom SAD-a razvijaju ENIAC (Electronic Numerical Integrator And Computer). Računanje je ubrzano oko 1000 puta u odnosu na Mark I zbog upotrebe elektronskih cijevi. Programirao se mjenjanjem ožičenja.
	
1948	EDVAC (The Electronic Discrete Variable Automatic Computer) – kompjuter koji je, osim pohrane podataka i davanja rezultata mogao pohranjivati i programe – John von Neuman.
1948	Tranzistor – Bell Laboratories – poluvodički elementi zamjenjuju elektronske cijevi, time smanjivši glomaznost dotadašnjih računala.
1951	UNIVAC (UNIVersal Automatic Computer) – Presper Eckert i John Mauchley, prvo komercijalno računalo plaćeno više od 1 milijun dolara.
1952	Pojavljuje se koncept integriranog kruga – Geoffrey

	Dummer
1964	Integrirani sklopovi niskog stupnja integracije – SSI (logička vrata na jednom chipu)
1968	Srednji stupanj intgracije – MSI (registri na jednom chipu)
1971	Visoki stupanj integracije – LSI (1kbitna memorija i UART)
1971	Intel 4004 – prvi mikroprocesor, razvijen kao kalkulatorski chip 
1972	Intel 8008 (8 bitni)
1973	Intel 8080 (8 bitni)
1974	Motorola 6800 (8 bitni)
1974	Broj mikroprocesora prestigao je ukupan broj mini, srednjih i velikih računala u svijetu.

1975	Altair – prvo osobno računalo proizvedeno u Micro Instrumentation and Telemetry Company (MITS). Zasnovano je na mikroprocesoru Intel 8080, a isporučivalo se u "kitu". Programirao se prekidačima, a kao izlaznu jedinicu imao je lampice.
1976	Zilog Z80 (8 bitni)
1978	Intel 8086 (16 bitni)
1979	Motorola MC68000 (32/16 bitni)
1982	Intel 80286 (16 bitni)
1984	Motorola MC68020 (32 bitni)
1993	Pentium generacija mikroprocesora (3.1 milijun tranzistora)

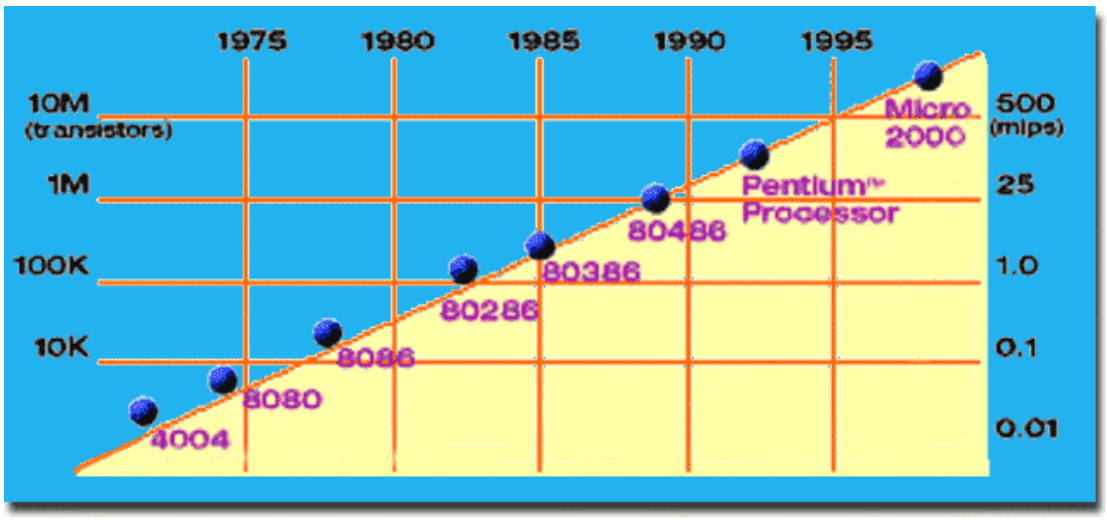
Što je to mikroprocesor?

- Mikroprocesor je složeni programski upravljivi sklop koji pribavlja, dekodira i izvršava instrukcije.
- Sastoji se od sklopova za rukovanje podacima i upravljačkih sklopova.
- Sklopovi za rukovanje podacima sastoje se od:
  - ALU
  - akumulatora
  - registara opće namjene
  - registara uvjeta ili status-registara
  - adresnih registara
- Upravljački sklopovi:
  - programsko brojilo
  - instrukcijski registar
  - sklopovi za dekodiranje
  - sklopovi za vremensko vođenje

- U početku se koristio u terminalima, kalkulatorima i komunikacijskim uređajima, zatim je postao centralna komponenta osobnih računala, a danas ga nalazimo u gotovo svim segmentima života (mikrovalne i ostale pećnice, televizori, glazbene linije i sl.)

Mooreov zakon

- Formuliran je 1964 godine
- G.E.Moore, direktor istraživanja tvrtke Fairchild Semiconuctor
- Glasi: Broj komponenti na integriranim krugovima svake godine se udvostručava



Mooreov zakon na djelu – Intel

Binarni brojevni sustav

- Zašto binarni brojevni sustav za računala?

  cijena ~ broju elektroničkih elemenata

  broj elektroničkih elemenata ~ baza sustava * broj pozicija

  cijena – c; baza – q; broj pozicija – p

  c = q * p

  najveći broj u bazi q s p pozicija N<q^p

  ln N < p * ln q

  p>=ln N/ln q

  c>=q * ln N/ln q

  c ima minimum za q=e (2,718)

Zbog jednostavnosti izvedbe uzima se q=2

- Reprezentacija brojeva u binarnom i decimalnom brojevnom sustavu (konverzije)

  Primjer:

  437->110110101

  437:2=218:2=109:2=54:2=27:2=13:2=6:2=3:2=1:2=0

  1 0 1 0 1 1 0 1 1

  110110101->437

  1*256+1*128+0*64+1*32+1*16+0*8+1*4+0*2+1*1=437

- Kratki pregled aritmetičkih i logičkih operacija

NOT		AND			OR			XOR		
0	1	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	1	0	1	1
		1	0	0	1	0	1	1	0	1
		1	1	1	1	1	1	1	1	0

A	B	ADD	C	SUB	C
0	0	0	0	0	0
0	1	1	0	1	1
1	0	1	0	1	0
1	1	0	1	0	0

TIPOVI ARHITEKTURA

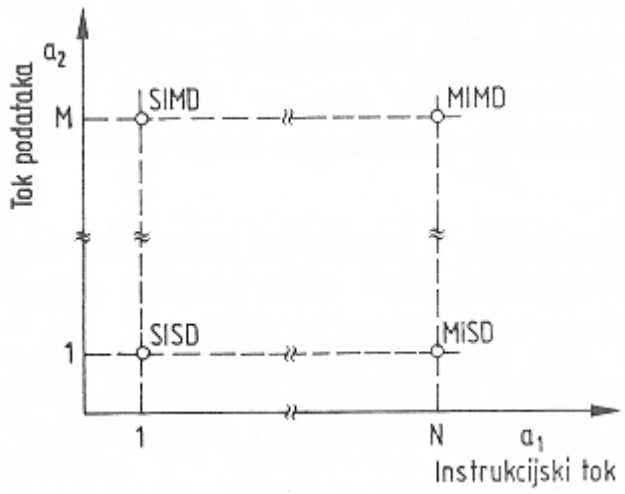
- Postoji mnogo različitih klasifikacija arhitektura računala.
- Klasifikacija prema načinu izvršavanja instrukcija:
  - ♦ računala s upravljačkim tokom (engl. control-flow)
    - von Neumannova arhitektura
    - program se izvršava točno prema uređenju instrukcija u programu
  - ♦ računala upravljana tokom podataka (engl. data-flow)
    - ova arhitektura omogućuje izvršavanje instrukcija kada su raspoloživi svi operandi potrebni za instrukciju
  - ♦ računala upravljana zahtjevom (engl. demand driven)
    - potreba za rezultatom pokreće izvršavanje instrukcije koja će taj rezultat generirati.

Primjer:

Pretpostavimo da je a=2, c=4 i d=7 te da imamo instrukciju u višem programskom jeziku: z=(a*5)+(c*d)

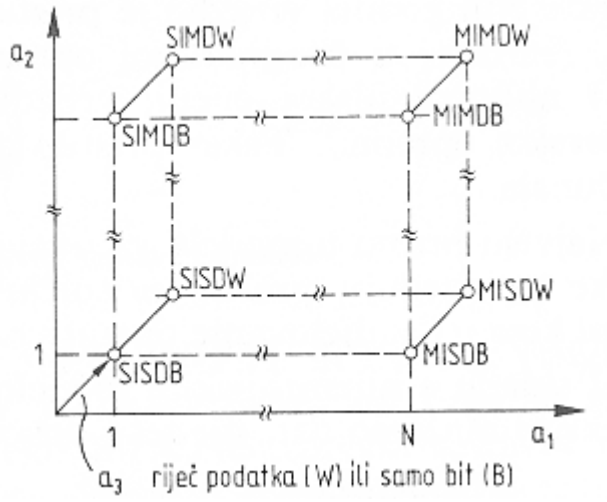
- arhitektura s kontrolnim tokom:
  - Instrukcije se odvijaju sekvencijalno
  - i1:(* a 5 z1)
  - i2:(* c d z2)
  - i3:(+ z1 z2 z)
- arhitektura upravljana tokom podataka
  - instrukcija će se obaviti kad se popune prazni argumenti
  - i1:(* () 5 i3/1)
  - i2:(* () () i3/2)
  - i3:(+ () () z/1)
  - z:()
- arhitektura upravljana zahtjevom
  - i1:(* a 5)
  - i2:(* c d)
  - z:(+ i1 i2)
  - program se pokreće upitom kojim se traži izračunavanje z-a i onda se obavlja redukcija.

M.J.Flynn, 1972 klasificira arhitekture prema paralelizmu u instrukcijskom toku i toku podataka.



Flynnova klasifikacija arhitektura

Handler, 1974 proširuje Flynnovu klasifikaciju arhitektura novom dimenzijom koja govori o tome da li su podaci bit-orijentirani ili su word-orijentirani.



Handlerovo proširenje Flynnove klasifikacije

- SISD – Single Instruction stream Single Data stream
  - ♦ uobičajeni naziv je von Neumannova arhitektura ili von Neumannova računala.

- MISD – Multiple Instruction stream Single Data stream
  - ♦ protočna računala – pipeline

  Primjer:

  dohvati instrukciju – 20ns

  dekodiraj instrukciju – 40ns

  dohvati operande – 60ns

  izvrši instrukciju - 50ns

  Izvršavanje instrukcija u neprotočnoj strukturi bi trajalo 170ns po instrukciji, a u protočnoj je efekt kao da izvršavanje traje 60ns.

  - ♦ posebna vrsta: sistolička polja

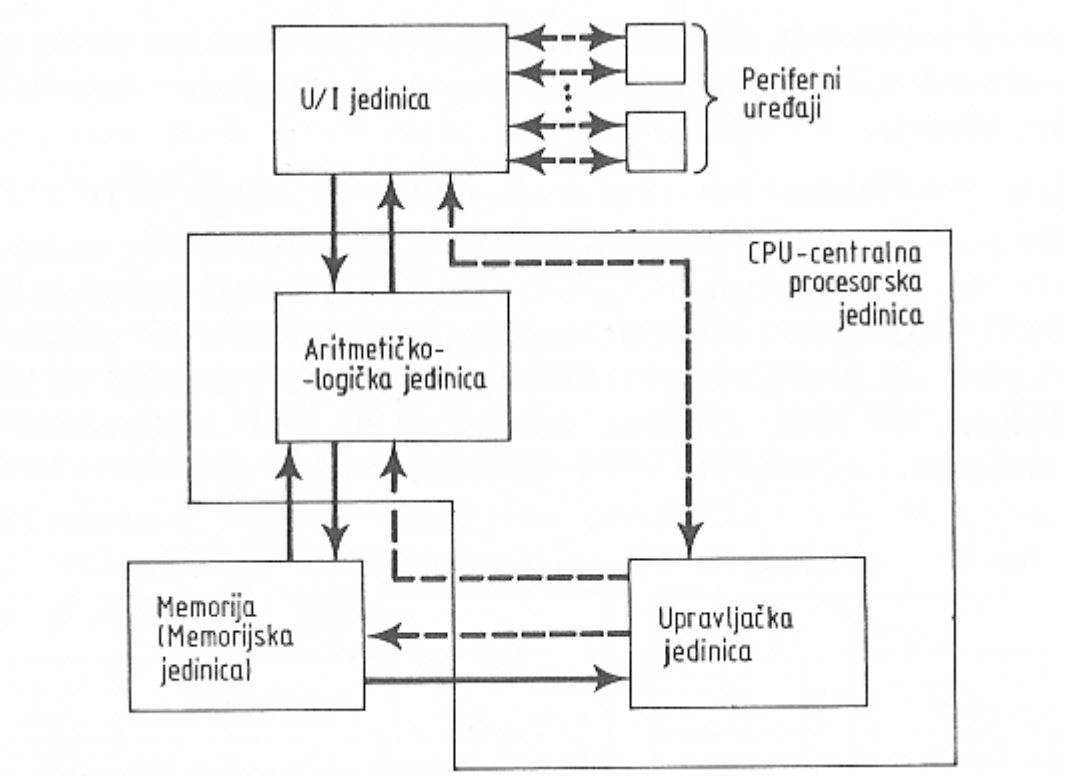
- SIMD – Single Instruction stream Multiple Data stream
  - ♦ Matrična računala
  - ♦ Nekoliko ALU-a rade pod nadzorom jedne upravljačke jedinice i izvršavaju iste instrukcije nad različitim podacima

- MIMD – Multiple Instruction stream Multiple Data stream
  - ♦ multiprocesorski sustavi
  - ♦ problemi koji se rješavaju vezani su uglavnom uz ubrzanja međuprocesorske komunikacije i komunikacije procesora s memorijom.

- Klasifikacija prema strukturi instrukcijskog skupa
  - ♦ RISC – Reduced Instruction Set Computer
  - ♦ CISC – Complex Instruction Set Computer

von Neumannova arhitektura

- predmet detaljnog izučavanja u ovom kolegiju
- A.W. Burks, H.H. Goldstine i J. von Neumann: "Uvodna rasprava o logičkom oblikovanju elektroničkog računalskog uređaja", 1946
  - ♦ opća namjena i potpuno automatsko izvođenje programa
  - ♦ osim podataka za računanje računalo mora pohranjivati i međurezultate
  - ♦ mora imati sposobnost pohrane programa
  - ♦ instrukcije su svedene na numerički kod tako da se instrukcije i podaci mogu pohranjivati na jedak način u memoriji.
  - ♦ mora imati jedinicu koja služi za obavljanje aritmetičkih operacija
  - ♦ mora imati jedinicu koja "razumije" instrukcije (upravljačka jedinica)
  - ♦ mora imati mogućnost komunikacije s vanjskim svijetom (U/I)



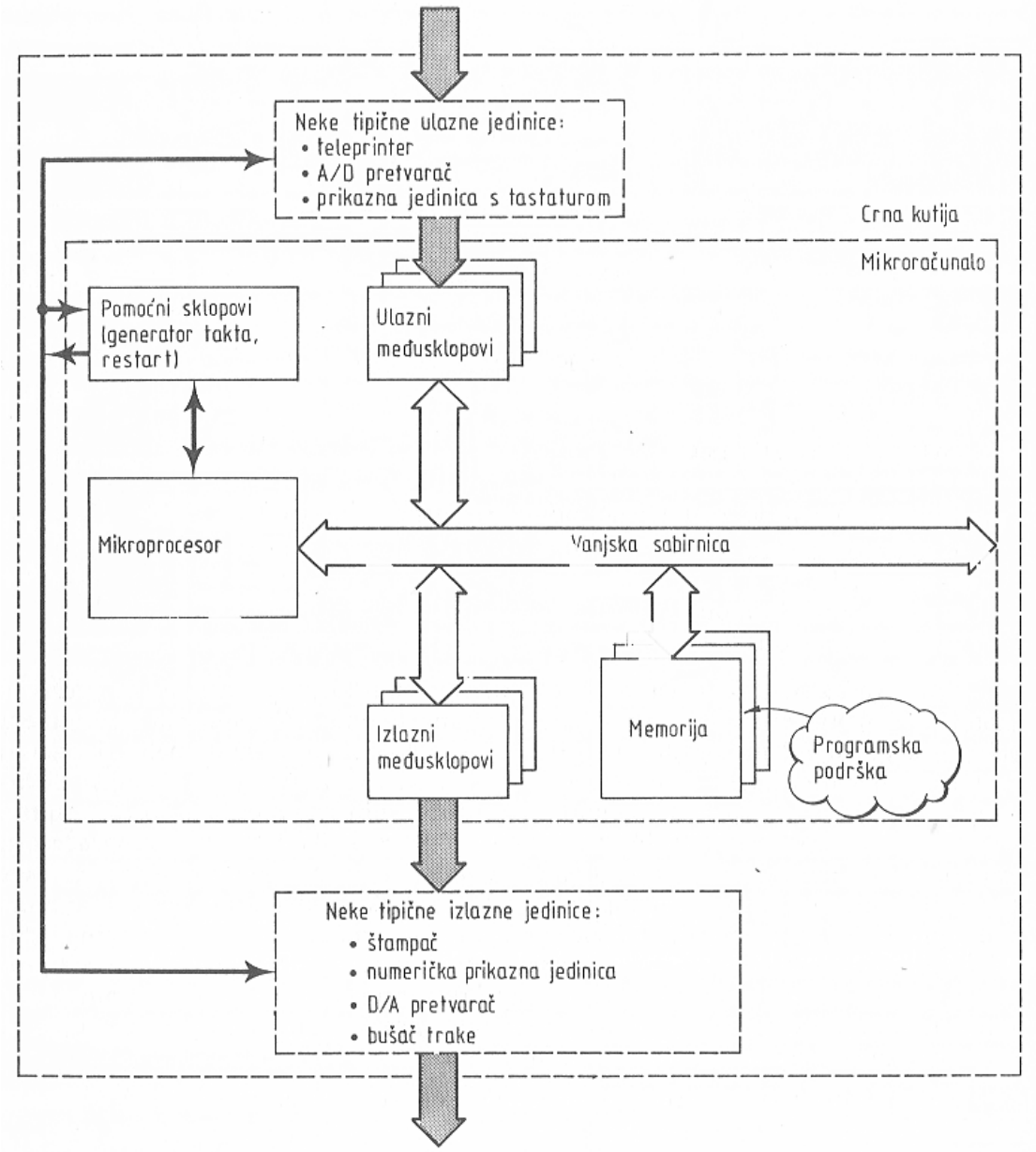
Model von Neumannove arhitekture

- nekoliko osnovnih svojstava von Neumannova računala:
  - ♦ binarni brojevni sustav
  - ♦ operandi duljine 40 bita (12 decimalnih mjesta)

- Faze pri izvršavanju instrukcija
  - ♦ PRIBAVI
    - iz memorije se čita instrukcija koja je na redu za čitanje
    - uvećava se PC tako da pokazuje na instrukciju koja neposredno slijedi
    - dekodiranje
  - ♦ IZVRŠI
    - generira se niz kontrolnih signala kako bi se izvršila instrukcija

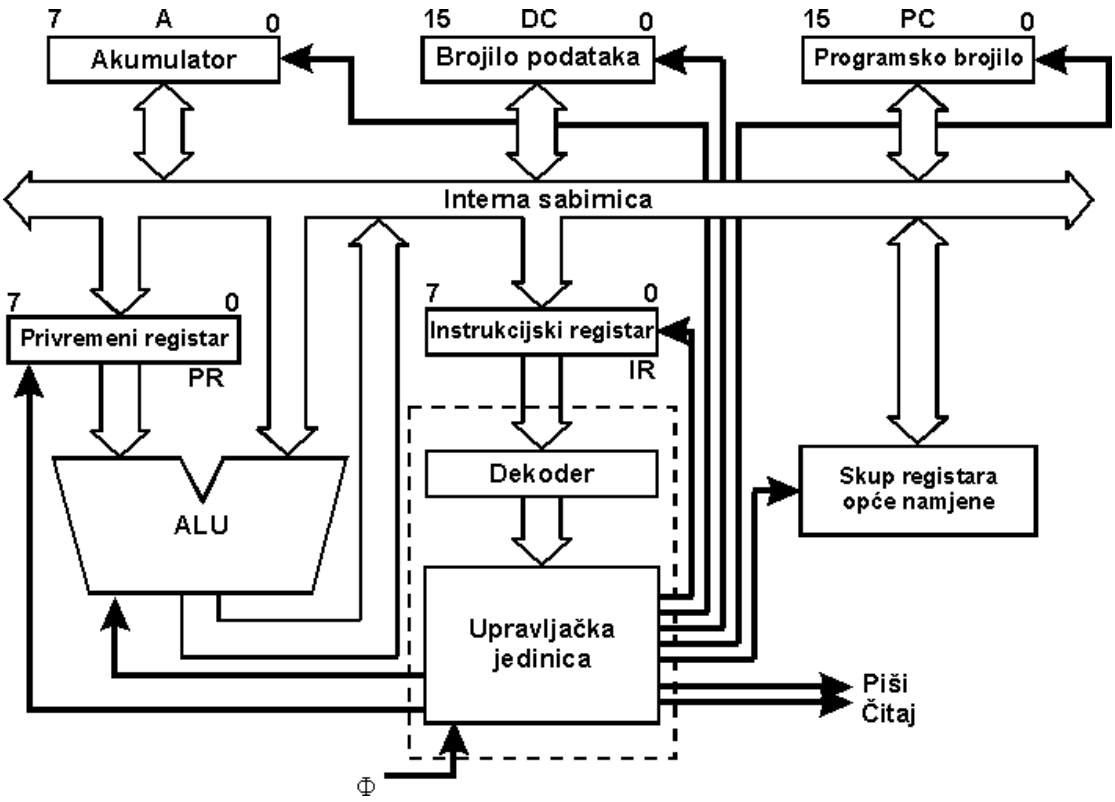
– Harvardska arhitektura – jedan od pokušaja da se zaobiđe von Neumannovo usko grlo (razdvojena je memorija za programe i podatke).

MIKRORAČUNALO



– mikroračunalo se sastoji od četiri osnovna bloka:
  - ♦ mikroprocesor
  - ♦ memorija
  - ♦ ulazni međusklop
  - ♦ izlazni međusklop

MODEL MIKROPROCESORA



Pojednostavljeni model mikroprocesora

Jednostavni model mikroprocesora sastoji se od:

- Akumulatora – 8-bitni registar koji sudjeluje u svim aritmetičko logičkim operacijama, služi za pohranu jednog od operanada
- Upravljačkog dijela koji se sastoji od:
  - ♦ Instrukcijskog dijela
  - ♦ Dekodera
  - ♦ Upravljačke jedinice
- Brojila podataka – 16-bitni registar koji služi za dohvat operanada iz memorije

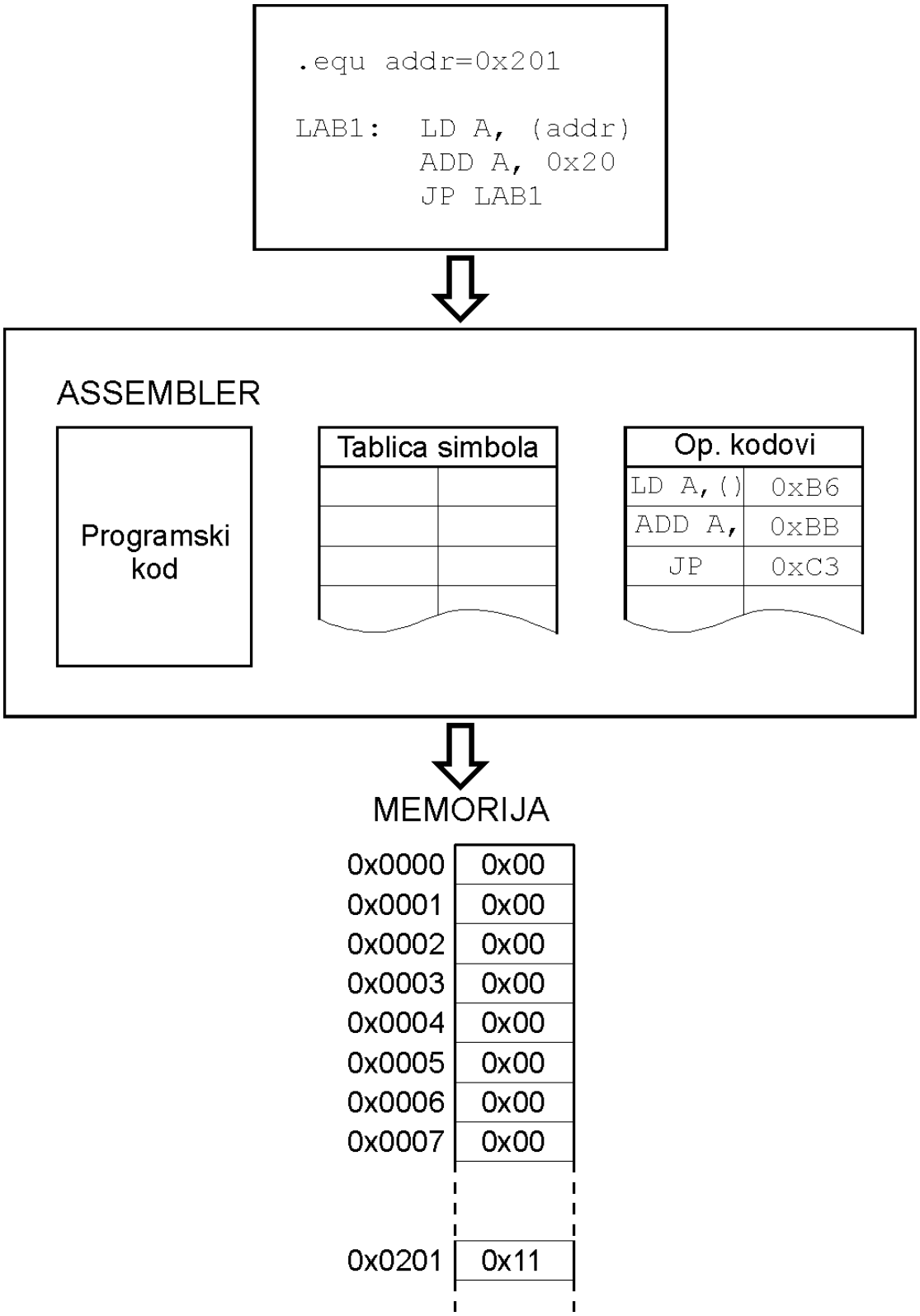
- Programskog brojila – 16-bitni registar koji služi za dohvat instrukcija iz memorije
- ALU – aritmetičko logička jedinica
- Skupa registara opće namjene – skup 8-bitnih registara koji služe za privremenu pohranu podataka
- Interne sabirnice koja preko odgovarajućeg međusklopa (koji se nalazi u mikroprocesoru) se nastavlja na vanjsku sabirnicu (nožice mikroprocesora)

Primjer:

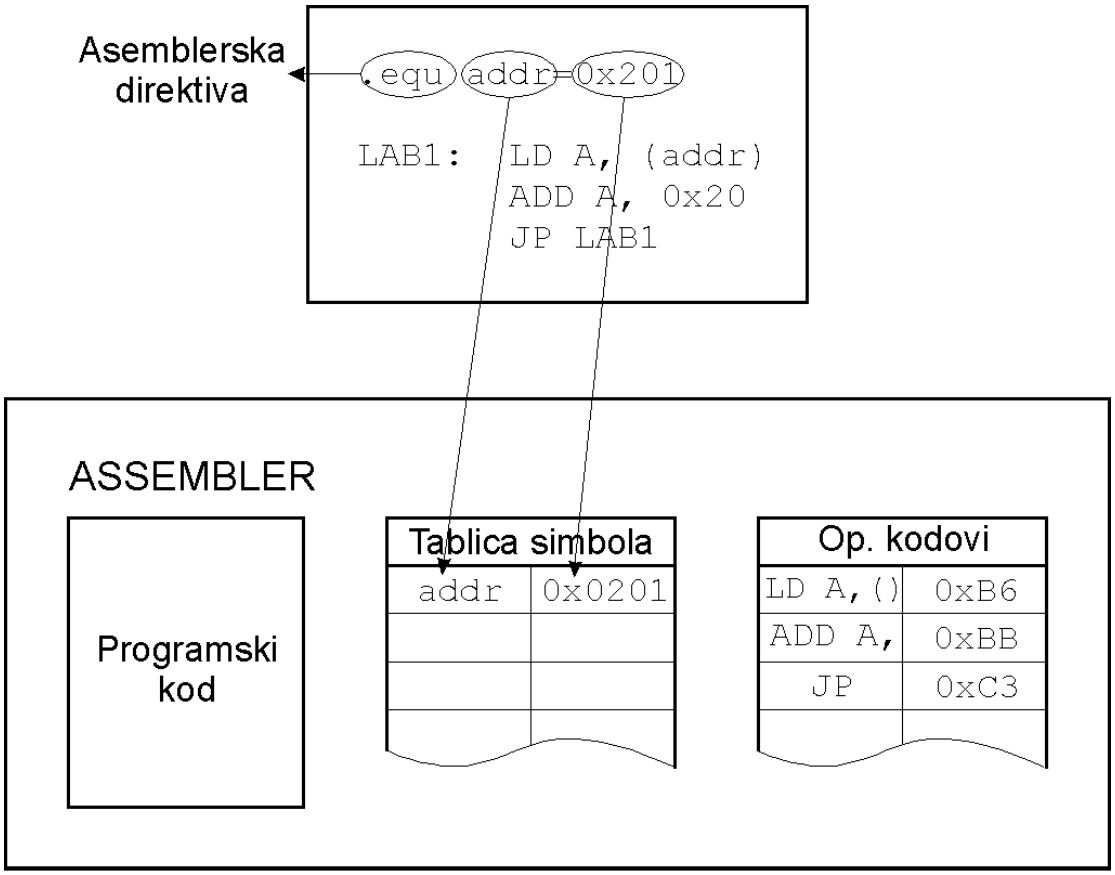
Prevesti (asemblirati) sljedeći primjer i pokazati njegovo izvršavanje na pojednostavljenom modelu mikroprocesora.

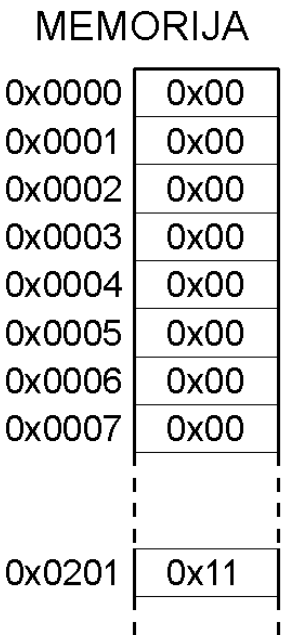
```
.equ addr=0x201
LAB1:   LD A, (addr)
ADD A, 0x20
JP LAB1
```

Početak

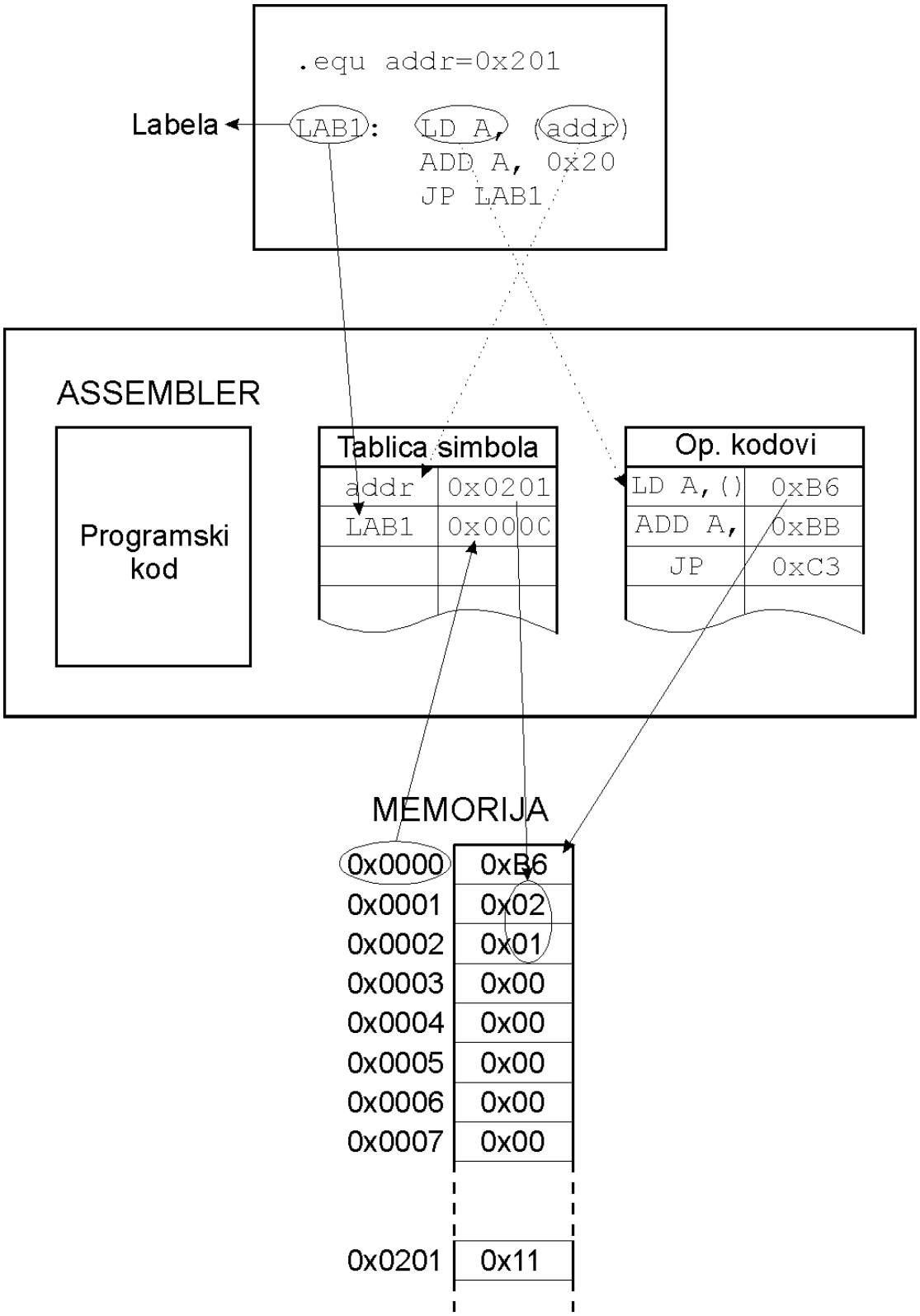

.equ addr=0x201
LAB1: LD A, (addr)
ADD A, 0x20
JP LAB1
ASSEMBLER
Programski kod
Tablica simbola
Op. kodovi
LD A,()
0xB6
ADD A,
0xBB
JP
0xC3
MEMORIJA
0x0000
0x00
0x0001
0x00
0x0002
0x00
0x0003
0x00
0x0004
0x00
0x0005
0x00
0x0006
0x00
0x0007
0x00
0x0201
0x11

Asembliranje prvog retka koda

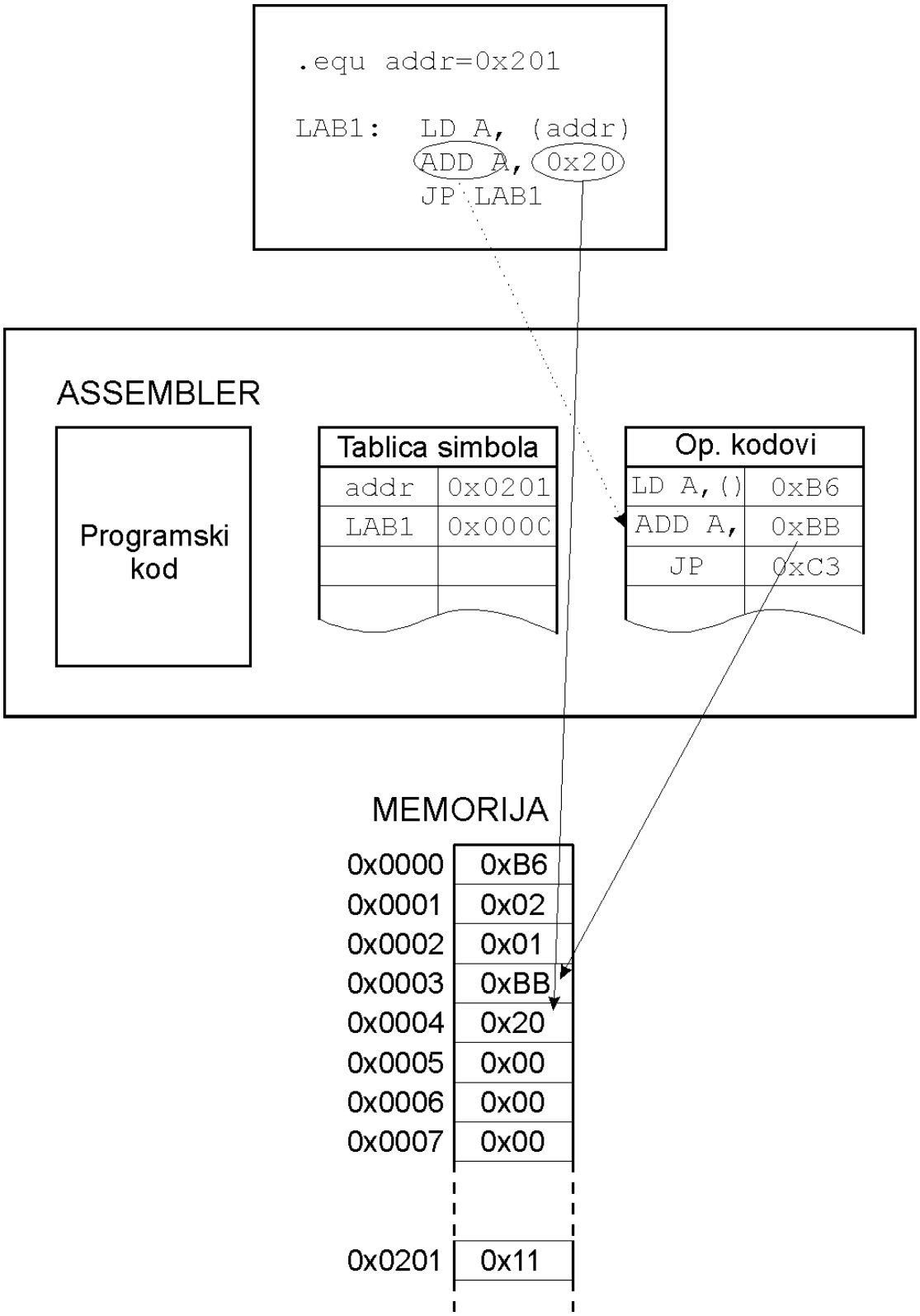




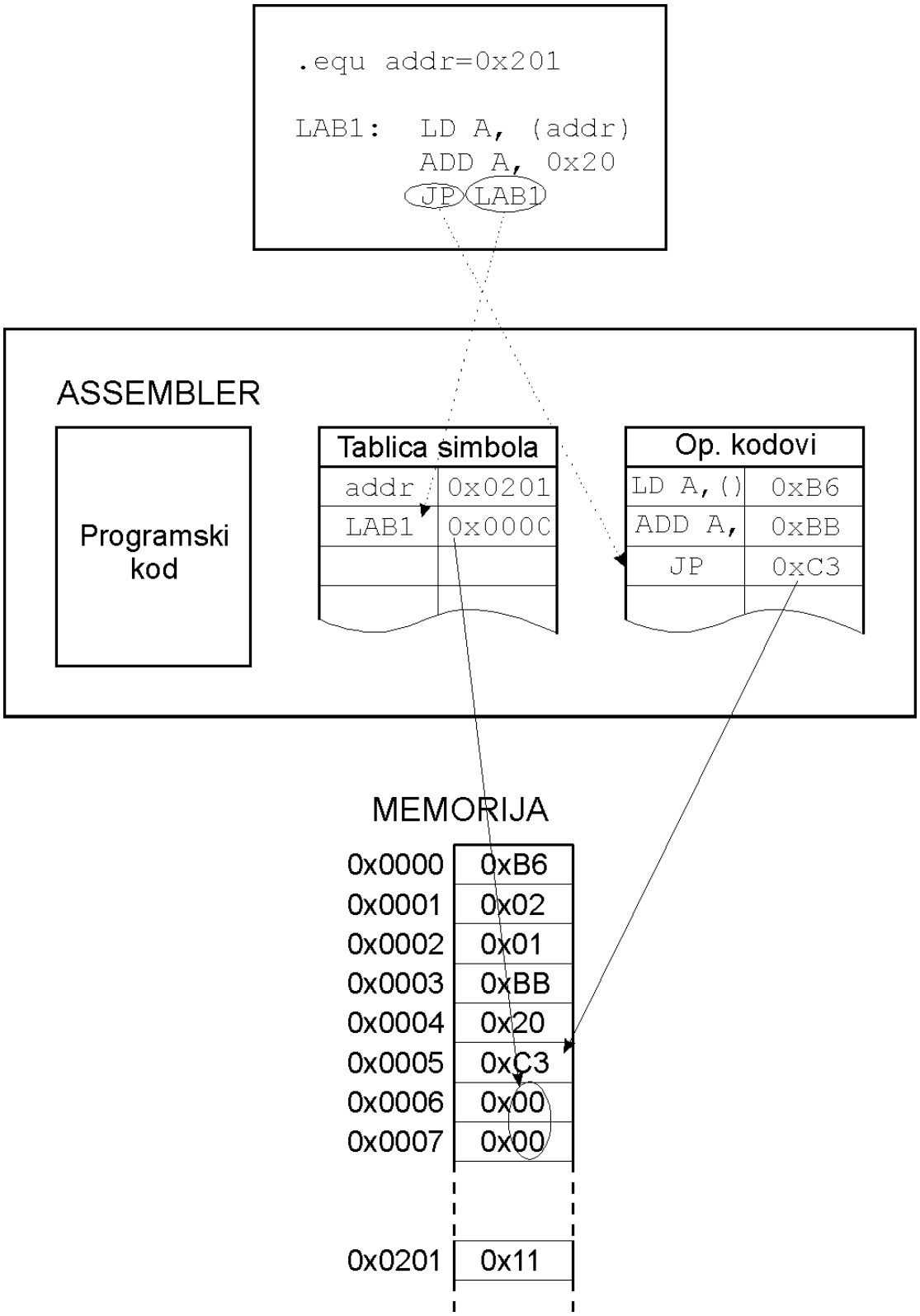
Asembliranje drugog retka koda



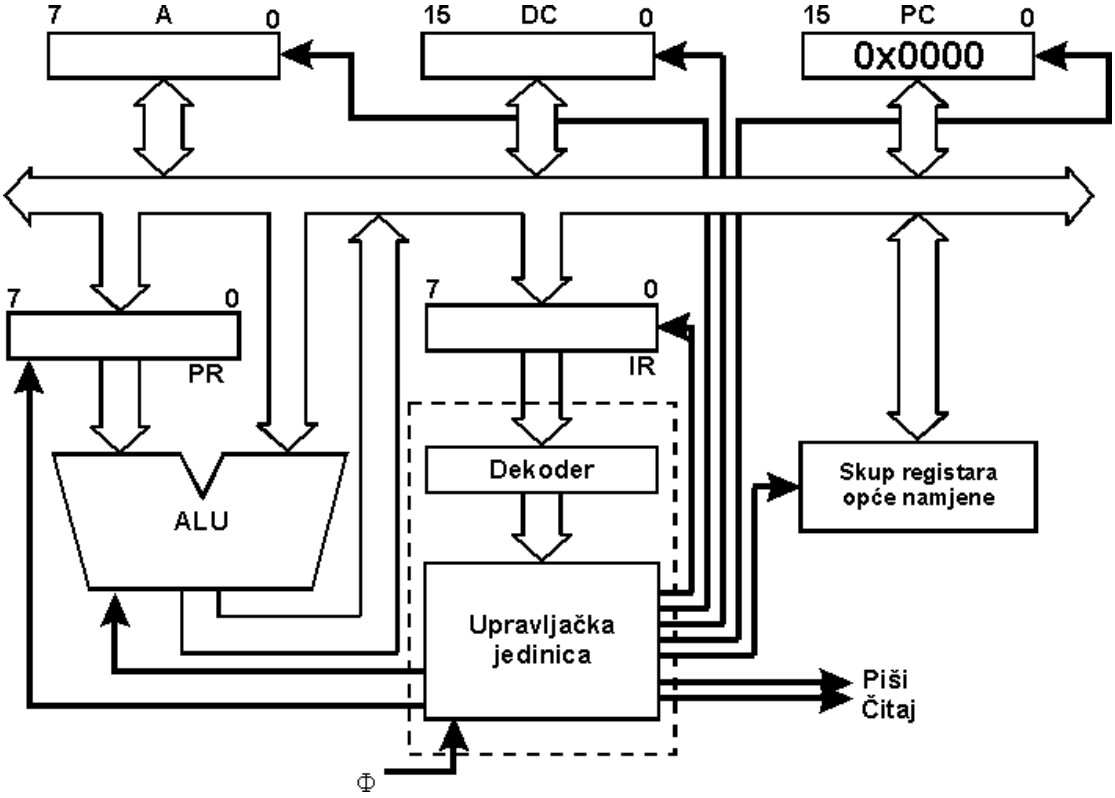
Asembliranje trećeg retka koda



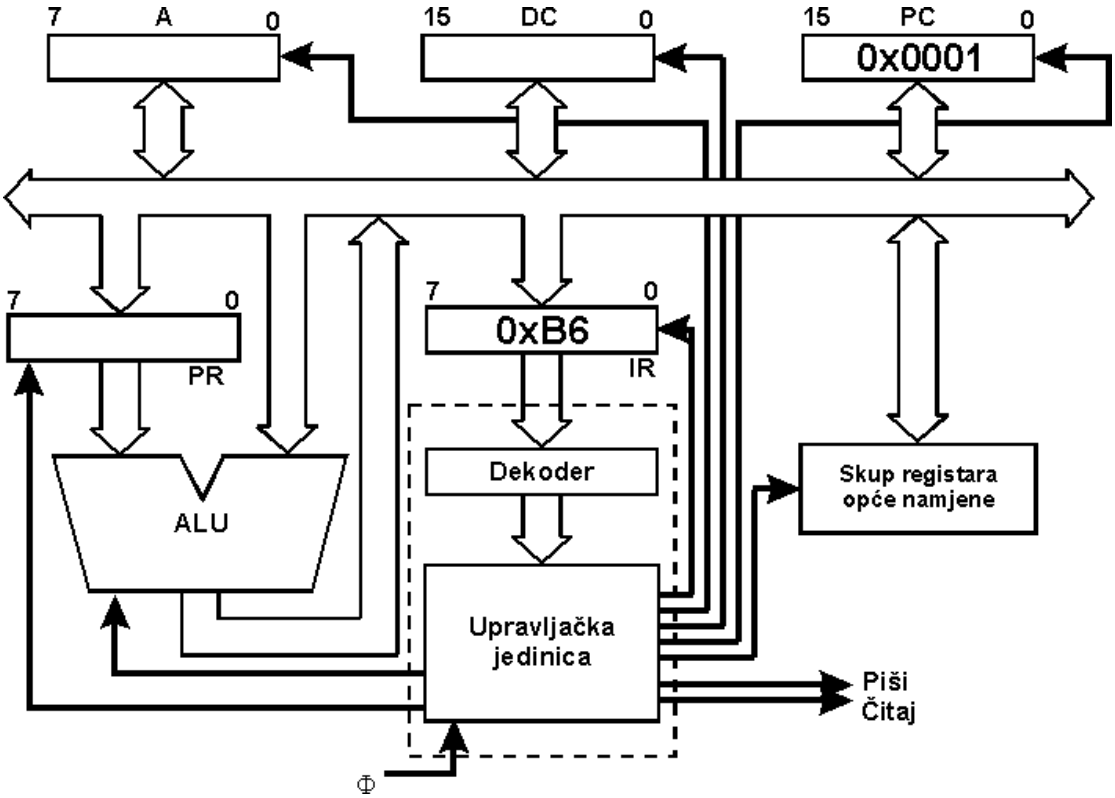
Asembliranje četvrtog retka koda



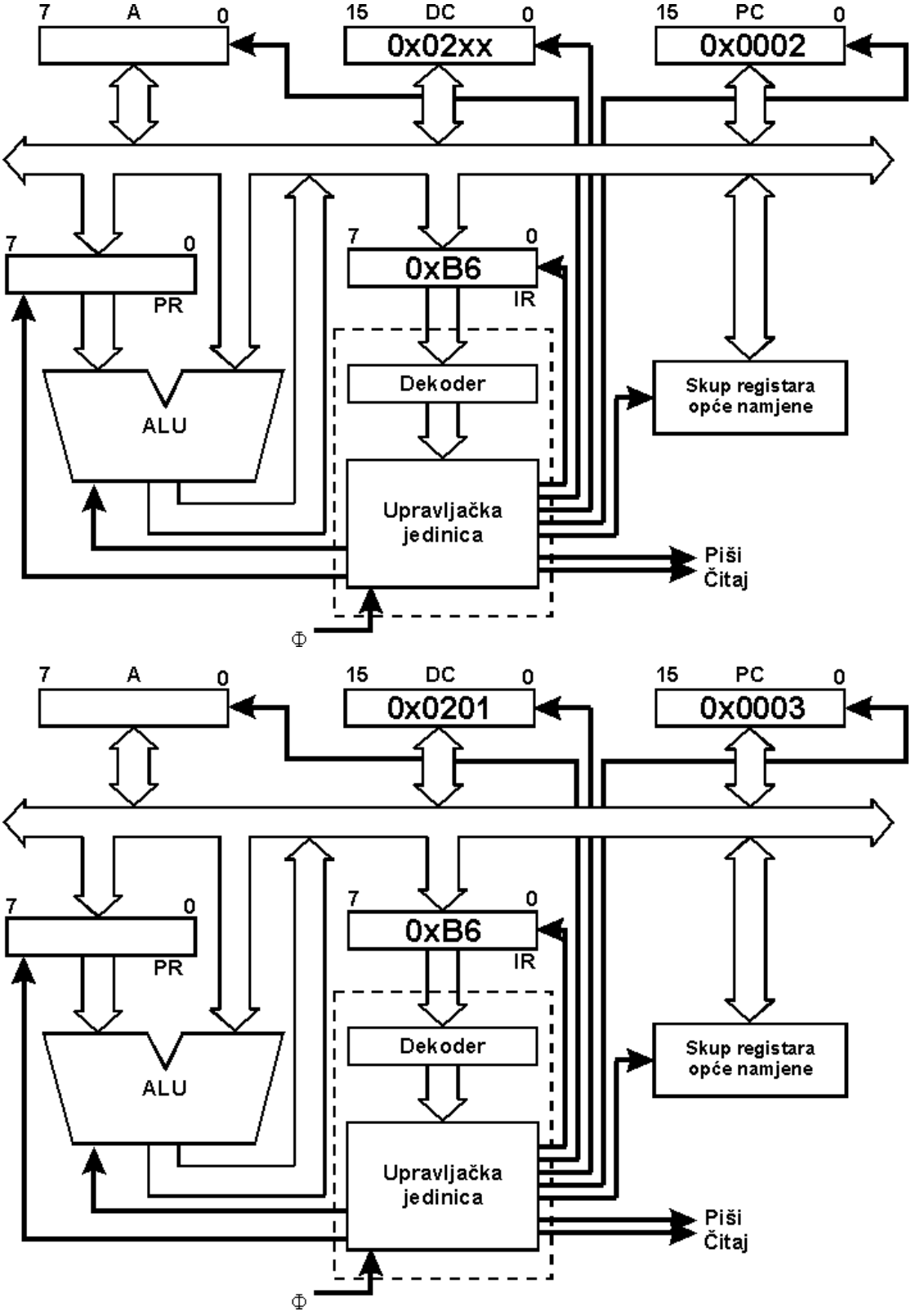
Početno stanje mikroprocesora



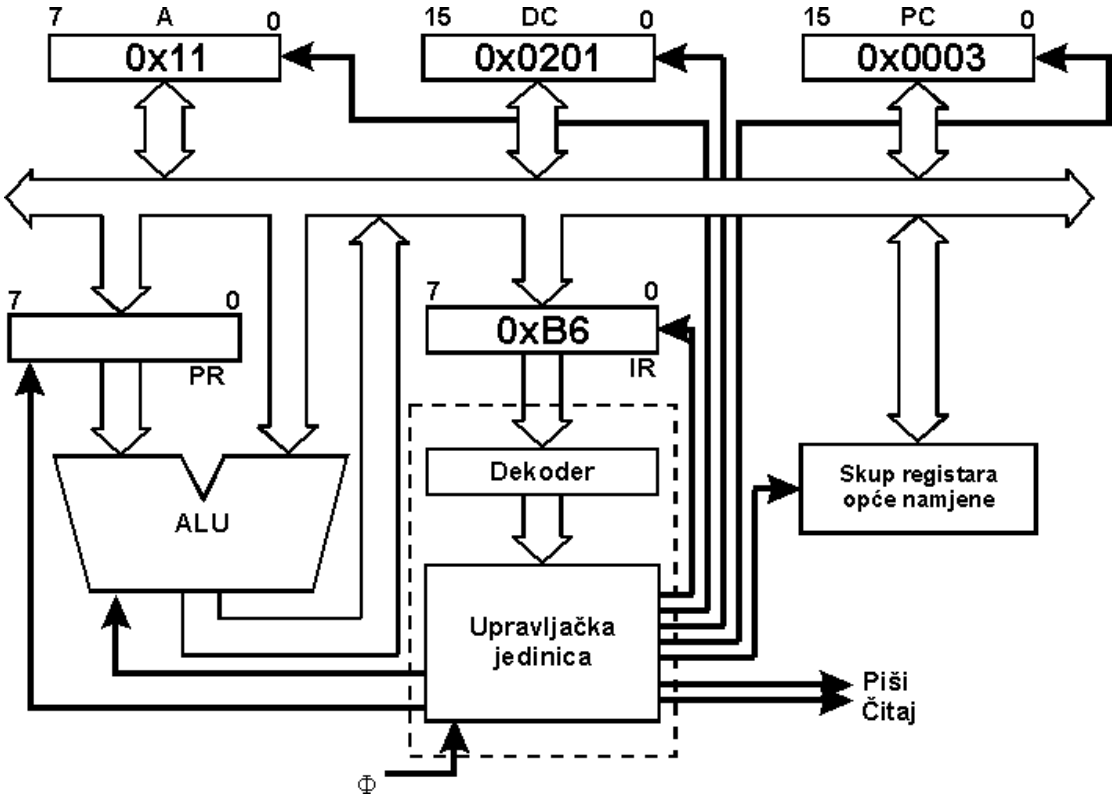
LD A, (0x0201) – dohvat instrukcije



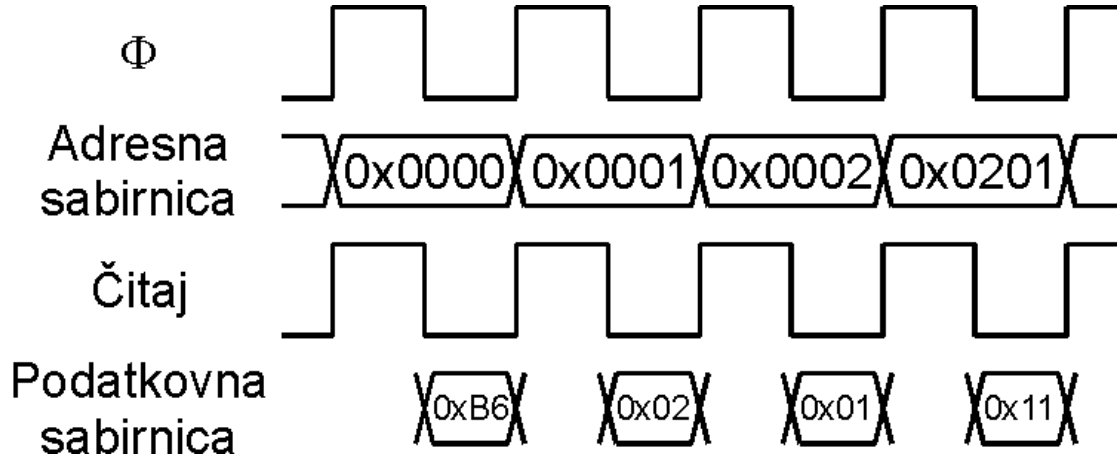
LD A, (0x0201) – dohvat operanda



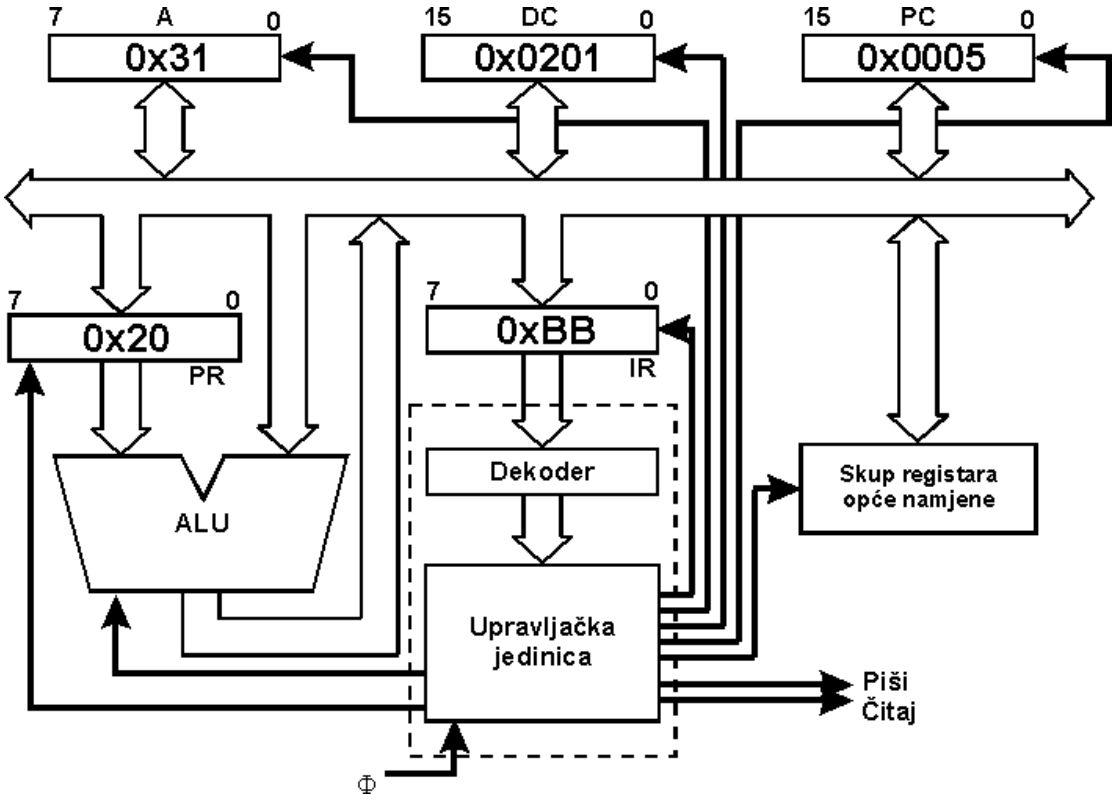
LD A, (0x0201) – izvrši



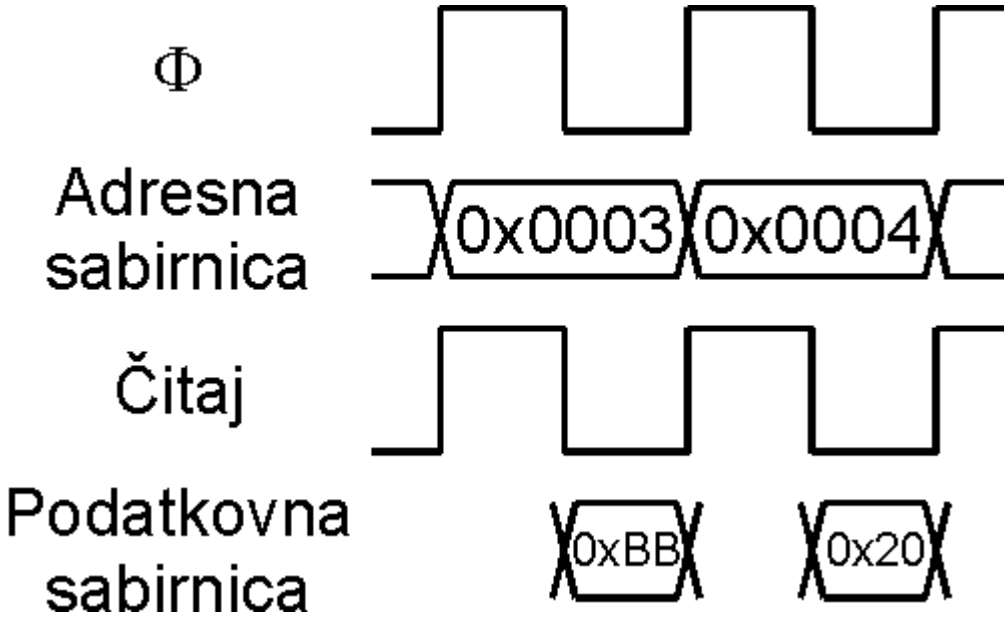
Dijagram stanja na sabirnicama za instrukciju LD A, (0x0201)



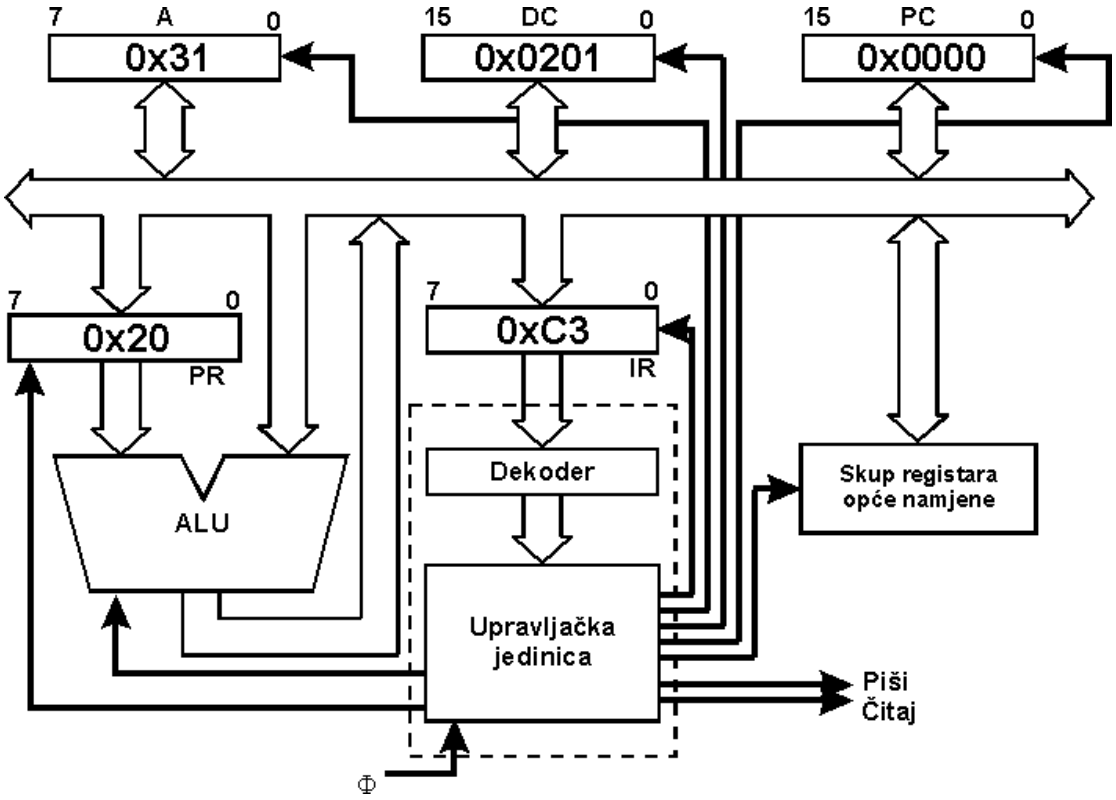
ADD A, 0x20



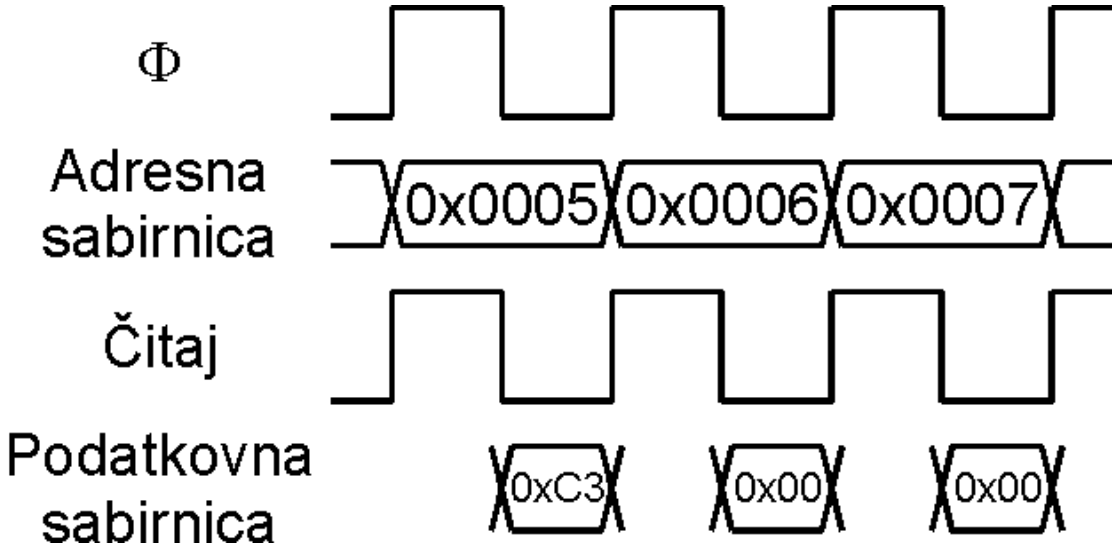
Dijagram stanja na sabirnicama za instrukciju ADD A, 0x20



JP 0x0000



Dijagram stanja na sabirnicama za instrukciju JP 0x0000



ADRESNI NAČINI (MODOVI)

Način na koji upravljačka jedinica pribavlja instrukcije i operande određen je adresnim načinom (address mode, načinom adresiranja, adresnim modom).

Postojanje više različitih adresnih načina omogućuje efikasnije rukovanje različitim strukturama podataka i pisanje efikasnijeg koda.

Adresni načini zastupljeni u današnjim mikroprocesorima

- **PROGRAMSKI**

  Programski adresni način opisuje na koji način se procesoru može "kazati" da skoči na neku instrukciju i od nje nastavi dalje izvršavanje programa,

  - DIREKTNI

    JMP xxxx i CALL xxxx – adresa je navedena direktno u sklopu instrukcije

  - INDIREKTNI

    IJMP, ICALL – adresa se nalazi u nekom od registara (za AVR ujedno i primjer implicitnog adresiranja jer nema operanda već se zna da se adresa nalazi u Z registru)

- RELATIVNI (ODNOSNI)

  RJMP xxxx, RCALL xxxx – adresa se izračuna tako da se na trenutnu vrijednost registra PC doda vrijednost operanda (xxxx može biti i pozitivan, ali i negativan broj)

– **PODATKOVNI**

Podatkovni adresni način opisuje na koji način se procesoru "kazuje" gdje se nalazi operand za operaciju zadanu instrukcijom (LDI, ADD...).

- USPUTNO (IMMIDIATE)

  LDI r16, xxxx – operand je u sklopu instrukcije

- REGISTARSKO DIREKTNO

  Instrukcije koje adresiraju samo registre. Kod AVR-a imamo instrukcije koje uključuju 1 operand (INC Rd) ili 2 operanda (ADD Rd, Rr).

- IMPLICITNO (UKLJUČNO)

  Instrukcije koje "nemaju" argumente tj. argumenti se podrazumijevaju (LPM gdje se podrazumijeva da je r0<-(Z)).

- DIREKTNO

  LDS Rd, xxxx – Vrijednost operanda se nalazi na adresi koja je direktno adresirana vrijednošću xxxx

- INDIREKTNO

  LD Rd, X – Vrijednost operanda se nalazi na adresi koja se nalazi u registru.

- INDIREKTNO S POMAKOM

  LDD Rd, Y+q – Vrijednost operanda se nalazi na adresi koja se dobije tako da se sadržaj koji se nalazi u registru uveća za pomak.

- INDIREKTNO S PRETHODNIM UMANJENJEM ADRESE

  LD Rd, -Y – Vrijednost koja se nalazi u registru se prvo umanji za jedan (dekrementira; predekrementno indirektno adresiranje), a zatim se tom vrijednošću adresira memorija i dohvaća operand.

- INDIREKTNO S NAKNADNIM UVEĆANJEM ADRESE

  LD Rd, Y+ - Vrijednost operanda se nalazi na adresi koja se nalazi u registru. Nakon dohvata operanda vrijednost u

adresnom registru (Y) se uveća za jedan (inkrementira; postinkrementno indirektno adresiranje).

- ADRESIRANJE KAZALOM STOGA (STACK POINTER)
  PUSH Rr; POP Rd – Sadržaj registra se pohranjuje na stog odnosno vrijednost s vrha stoga se sprema u registar.

UPOTREBA STOGA (STACK)

- Stog je LIFO struktura (Last In First Out).
- Gotovo svi suvremeni mikroprocesori koriste stog
- Vrh stoga je određen registrom koji se zove pokazivač stoga (stack pointer; SP, SPH, SPL)
- Stog se obično puni tj. "raste" od viših adresa prema nižima

Mikroprocesor koristi stog:

- za pohranu povratne adrese pri pozivu potprograma
- za pohranu povratne adrese pri obradi prekida (poneki i za pohranu sadržaja radnih registara)

Programer u asembleru koristi stog:

- za privremenu pohranu sadržaja nekog registra

Operativni sustav koristi stog za:

- za pohranu minimalnog konteksta

Viši programski jezici (C) koriste stog:

- za prijenos parametara između podprograma
- za smještaj lokalnih varijabli

ARITMETIČKO LOGIČKA JEDINICA (ALU)

- Izvorni naziv: Arithmetic Logical Unit (ALU)
- ALU je zadužena za obavljanje operacija nad cijelim brojevima, dok se rad s realnim brojevima rješava posebnom jedinicom (FPU) ili programski.
- Aritmetičke operacije koje obavlja ALU:
  - ♦ zbrajanje
  - ♦ oduzimanje
  - ♦ zbrajanje s bitom prijenosa (carry)
  - ♦ oduzimanje s bitom prijenosa (carry)
  - ♦ uvećavanje za 1 (inkrement)
  - ♦ umanjivanje za 1 (dekrement)
  - ♦ (množenje)
  - ♦ (cjelobrojno dijeljenje)
- Logičke operacije koje obavlja ALU:
  - ♦ AND
  - ♦ OR
  - ♦ XOR (EOR)
  - ♦ NOT

- ♦ operacije nad bitovima (posmak, rotiranje, postavljanje, brisanje i testiranje stanja nekog bita, eksplicitno upravljanje zastavicama (flags))

– ALU se obično gradi na način da se izgradi aritmetički dio a zatim se on nadopuni i za obavljanje logičkih operacija.

Osnovni građevni element ALU-a je **potpuno zbrajalo**. To je **kombinacijski** sklop koji je u potpunosti opisan sljedećom tablicom istine:

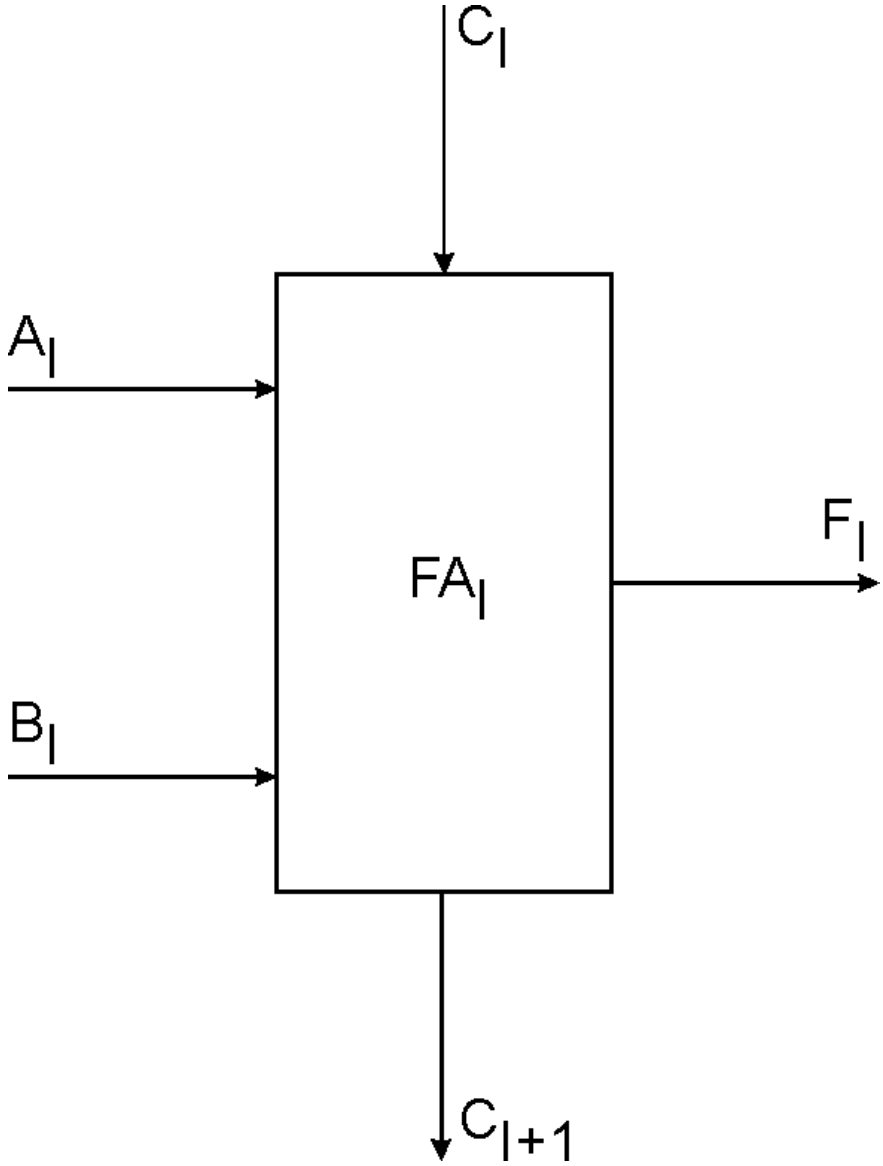
A	B	C_{in}	F	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Potpuno zbrajalo se može prikazati sljedećim logičkim funkcijama:

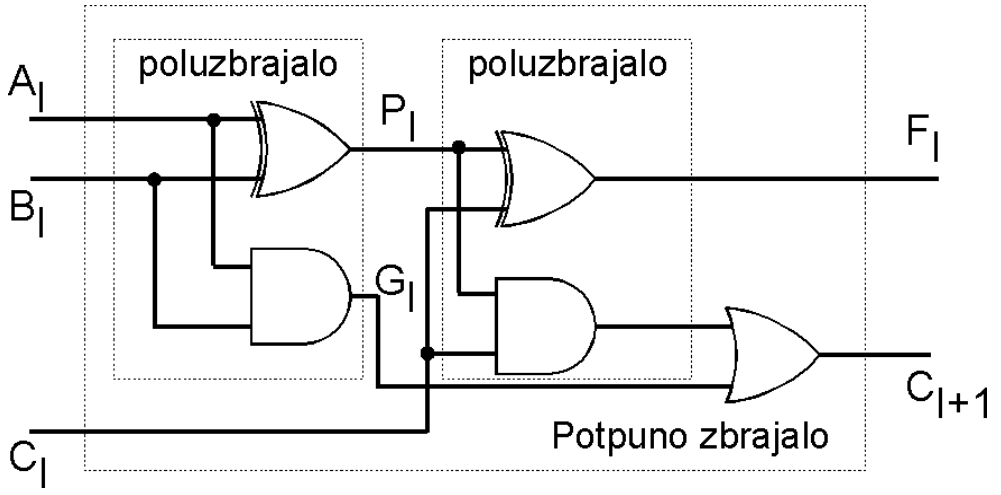
$F=A \text{ xor } B \text{ xor } C_{in}$

$C_{out}=AB + AC_{in} + BC_{in}$

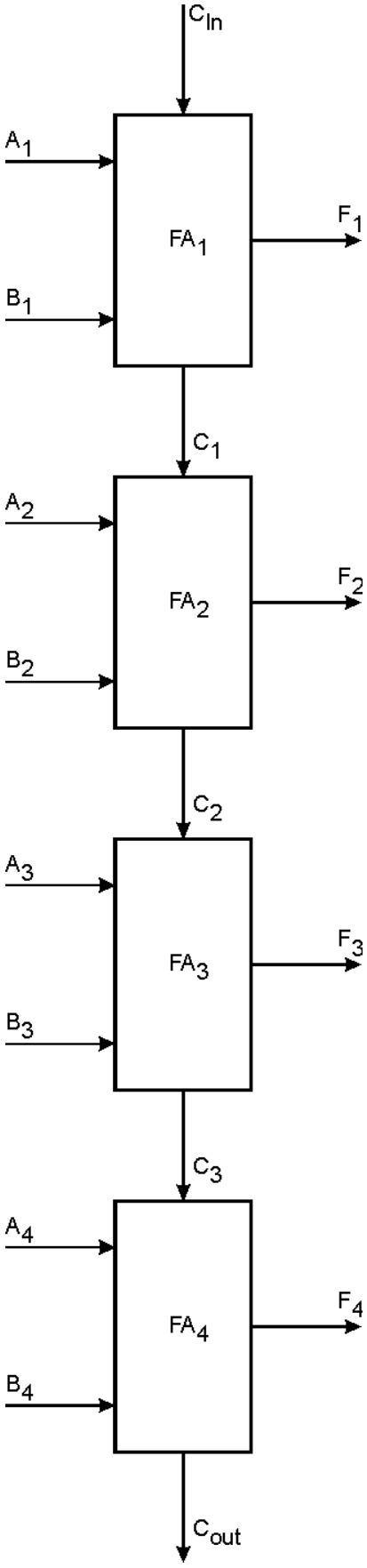
Simbol koji se koristi za potpuno zbrajalo:



Realizacija potpunog zbrajala:



Zbrajalo za više bitova dobiva se kaskadiranjem potpunih zbrajala:



- Ovako realiziran sklop za zbrajanje je spor.
- Razlog tome je što se bit prijenosa (carry) širi (propagira) kroz nekoliko nivoa logičkih vratiju. (za n bitova 3n vratiju)
- Rješenje je sklop za predviđanje bita prijenosa (carry look ahead)
  Cilj je konstruirati sklop koji znatno brže i istovremeno daje bitove prijenosa za sva zbrajala. To se može postići ako iz potpunog zbrajala izdvojimo signale generiranja bita prijenosa G_I i širenja bita prijenosa P_I:

  $G_I=A_IB_I$

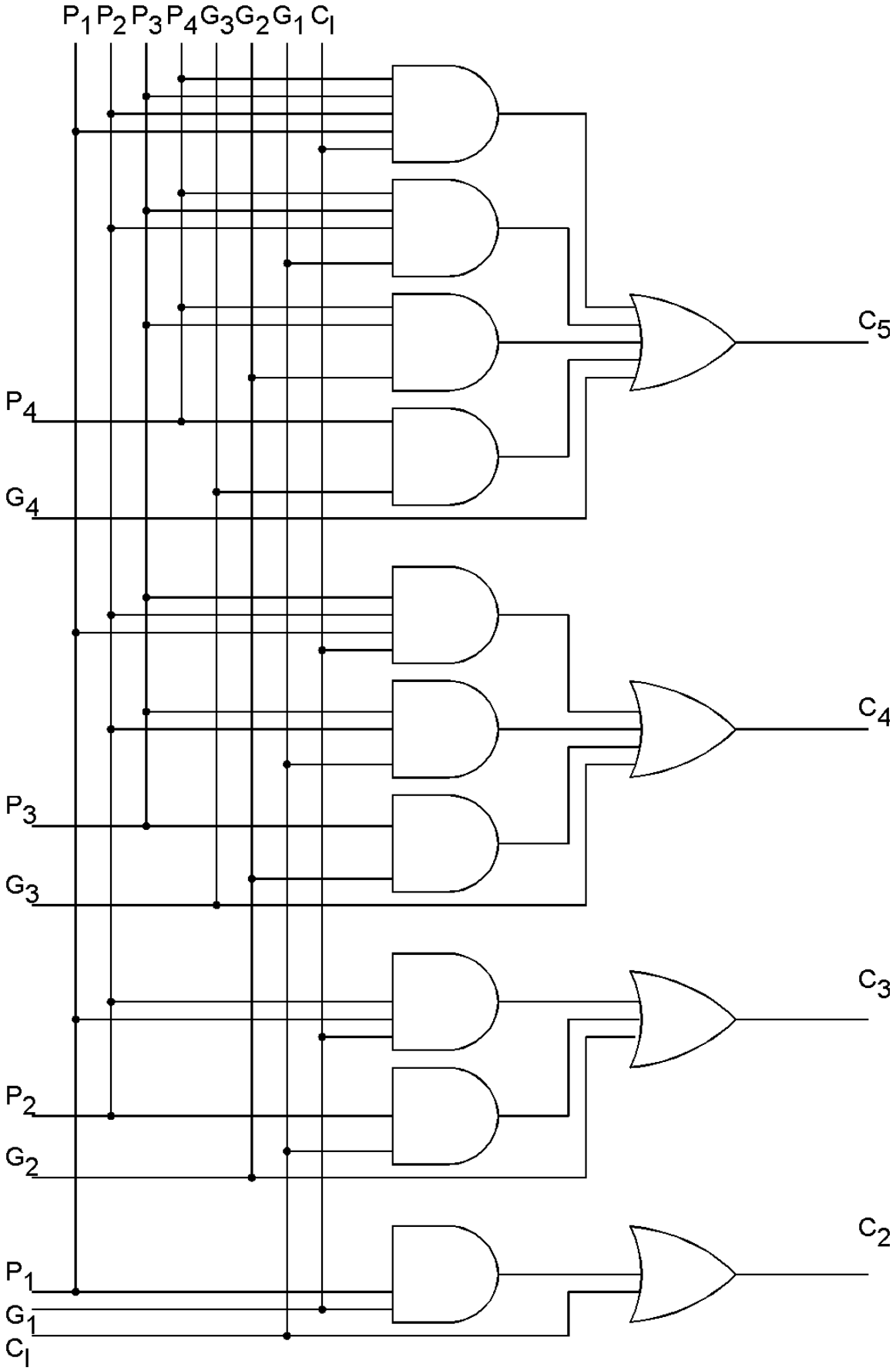
  $P_I=A_I \text{ xor } B_I$

  $C_{I+1}=G_I+P_IC_I$

  $C_{I+2}=G_{I+1}+P_{I+1}C_{I+1}=G_{I+1}+P_{I+1}(G_I+P_IC_I)=G_{I+1}+P_{I+1}G_I+P_{I+1}P_IC_I$

  Zahvaljujući tehnologiji koja nam omogućava izvođenje logičkih vratiju s više ulaza svaki bit prijenosa (carry) se generira prolaskom kroz točno dva nivoa logičkih vrata (jedan AND i jedan OR).

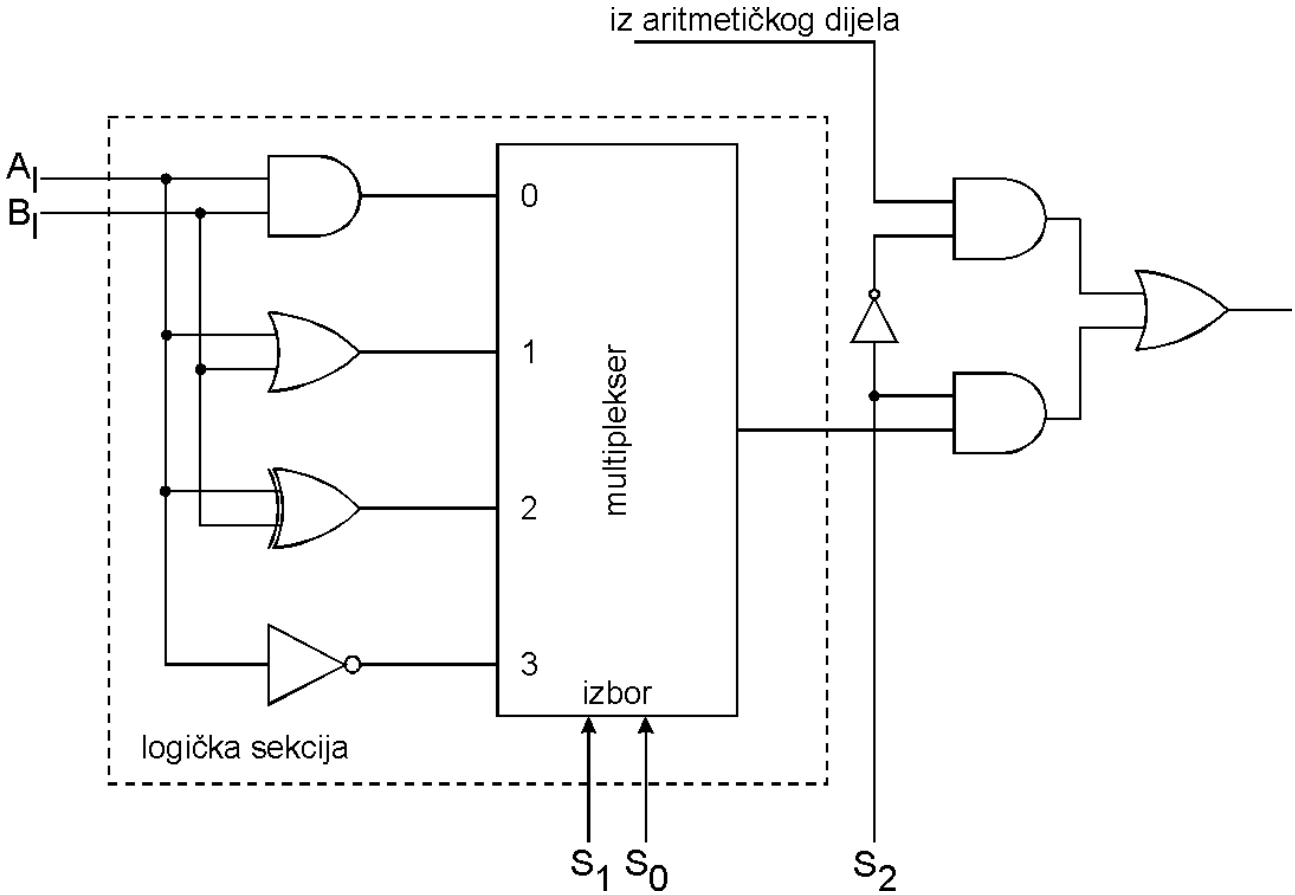
Sklop za predviđanje bita prijenosa (carry look ahead):



Koristeći zbrajalo lako se realizira još nekoliko osnovnih aritmetičkih operacija, računanjem u sistemu dvojnog komplementa:

- oduzimanje – A+not(B)+C_{In} gdje je C_{In}=1
- prijenos – B=0, C_{In}=0
- inkrementiranje – B=0, C_{In}=1
- dekrementiranje – B=1, C_{In}=0

Iako se u praksi aritmetička sekcija dopunjava i adaptira kako bi mogla izvoditi i logičke operacije (zbog uštede u broju logičkih vratiju, odnosno mjesta na chipu), načelno logička sekcija ALU-a se izvodi ovako:



Osim sklopa za zbrajanje, neke naprednije ALU imaju i sklopove za množenje i cjelobrojno dijeljenje. Treba imati na umu da kad se govori o ALU gotovo uvijek se misli na rad s CIJELIM brojevima, dok se operacije s realnim brojevima (tj. njihovim aproksimacijama) odvijaju u zasebnom dijelu procesora koji je obično toliko složen da ga se naziva matematičkim koprocesorom, koprocesorom za rad s brojevima u pomičnom zarezu ili floating point unit (FPU).

Osim sklopova koji tvore aritmetičku i logičku sekciju ALU-a važan dio su i sklopovi za posmak (shift).

Najčešće operacije koje se izvode sklopovima za posmak su:

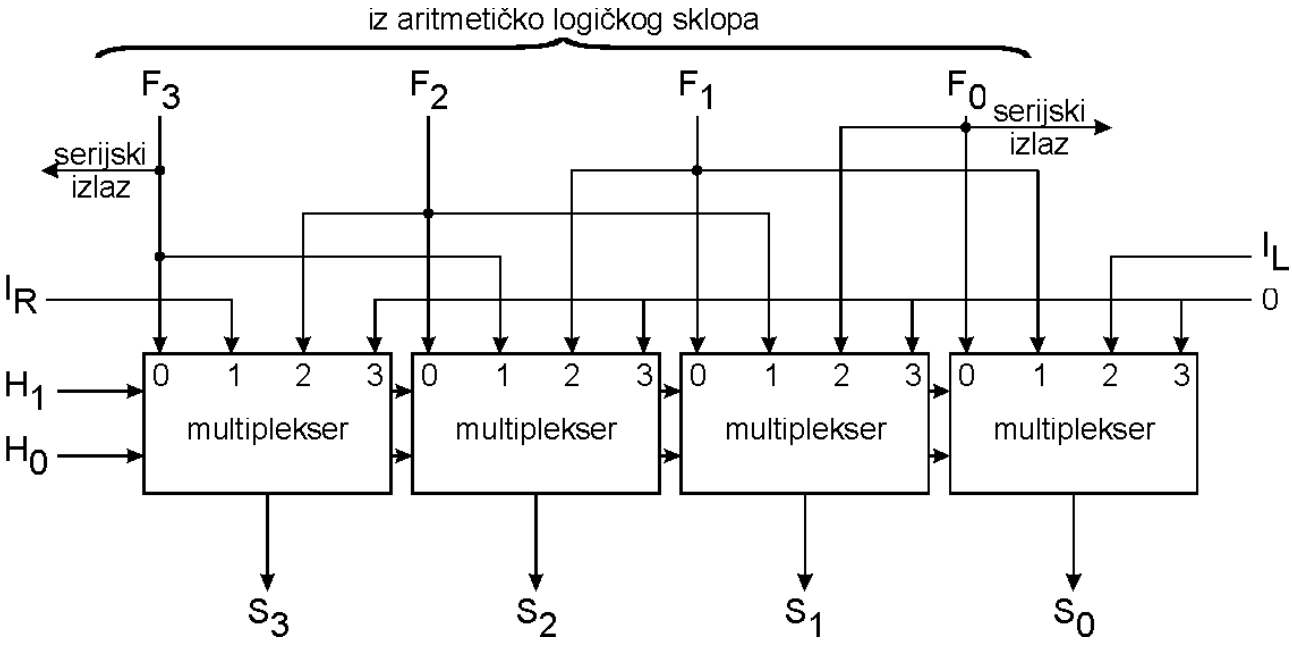
- logički posmak u lijevo ili u desno – 0 ulazi na LSB, odnosno na MSB
- aritmetički posmak u lijevo ili u desno – za razliku od logičkog ovaj čuva predznak broja u dvojnom komplementu, tj. MSB ostaje kakav je bio i prije posmaka
- rotacija u lijevo ili u desno – C ulazi na LSB, odnosno MSB, a MSB odnosno LSB izlaze u C

Sklopovi za posmak mogu biti realizirani:

- u vidu shift registara
- kombinacijskim sklopovljem – ovaj način izvedbe se češće prakticira jer je rad sklopa brži.

Sklop za posmak se spaja na izlaz iz aritmetičko logičke sekcije.

Primjer realizacije sklopa za posmak kombinacijskim sklopovima (multiplekserima):



Ovakav sklop je sposoban posmaknuti sadržaj ulaza samo za jedno mjesto u lijevo ili u desno. Ako je potrebno napraviti posmak za dva ili više mjesta tada je potrebno upotrijebiti instrukciju za posmak dva ili više puta tj. potrebno je dva ili više ciklusa izvršavanja instrukcije.

U suvremenim mikroprocesorima postoje sklopovi za posmak koji su u stanju samo jednom instrukcijom napraviti posmak za dva ili više mjesta. Takvi sklopovi se uobičajeno nazivaju bačvasti sklopovi za posmak tj. barrel shifter.

UPRAVLJAČKA JEDINICA

Upravljačka jedinica pribavlja, dekodira i upravlja izvođenjem instrukcija. Preko adresne, podatkovne i kontrolne sabirnice komunicira sa svim komponentama mikroprocesora i mikroračunala.

Rad mikroprocesora se dijeli u dvije faze:

- PRIBAVI – tijekom koje upravljačka jedinica šalje upravljačke signale potrebne za dohvat instrukcije iz memorije
- IZVRŠI – tijekom koje upravljačka jedinica šalje signale potrebne za obavljanje instrukcije

Elementarna operacija kojom se generira jedan ili više upravljačkih signala potrebnih za prijenos podataka ili aktiviranje pojedinog sklopa naziva se *mikrooperacija*.

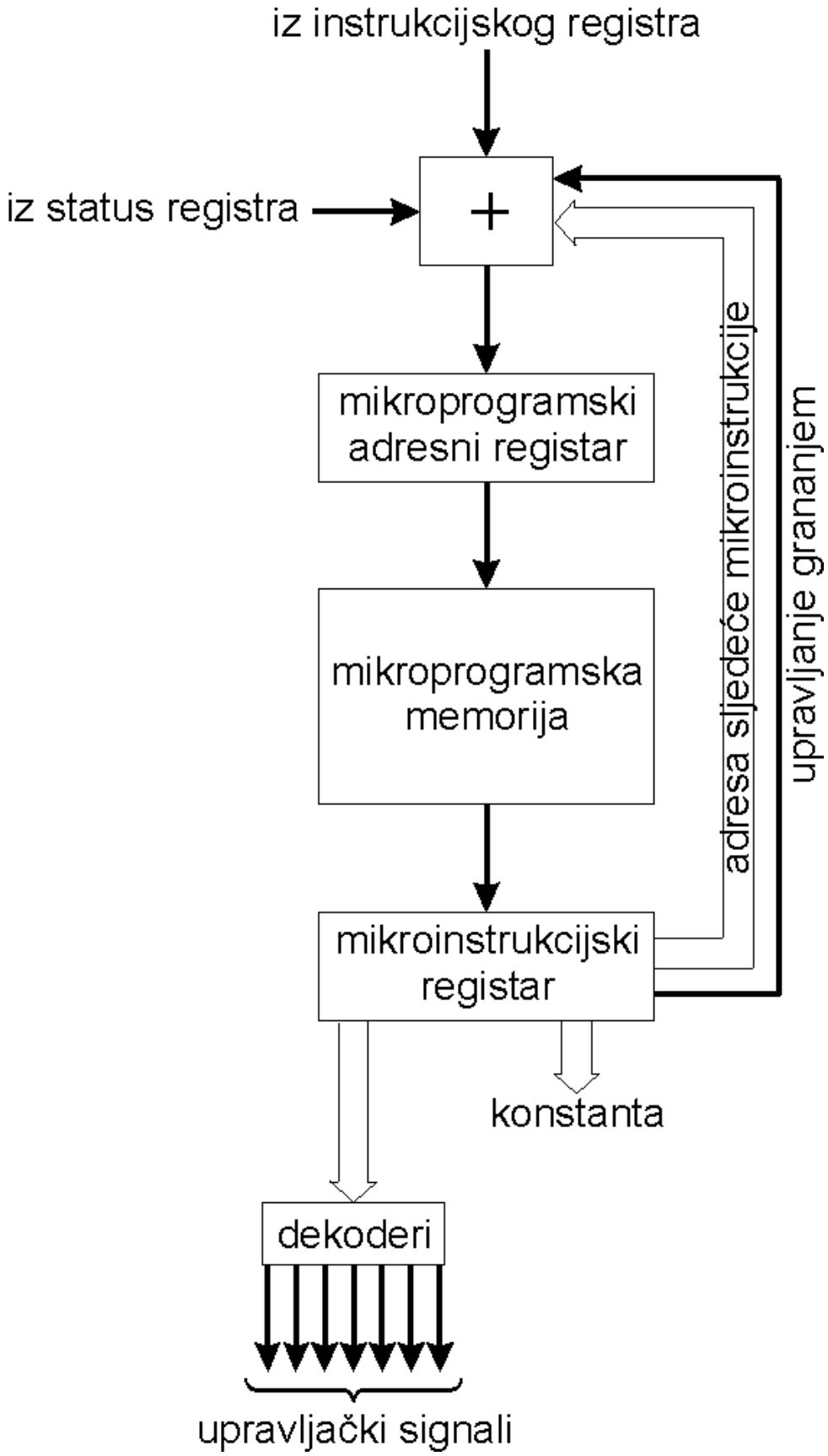
Primjeri mikrooperacija:

- prijenos podataka između registara
- aktiviranje ALU-a za obavljanje pojedine operacije
- aktiviranje sklopa za posmak

Najčešći način na koji se danas realiziraju upravljačke jedinice jest mikroprogramiranje.

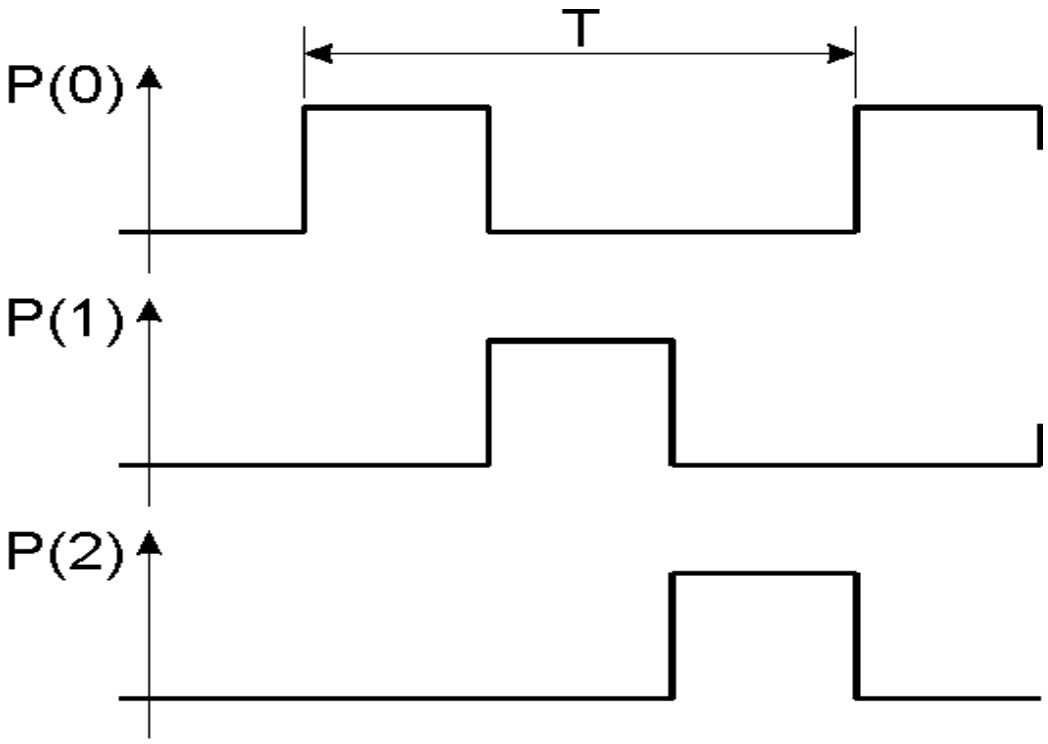
Važnost upravljačke jedinice vidljiva je i u tome što na suvremenim procesorima zauzima i preko 80% prostora na siliciju.

Način realizacije mikroprogramirane upravljačke jedinice:



Svojstva ovako realizirane mikroprogramirane upravljačke jedinice:

- mikrooperacije su opisane mikroinstrukcijama koje su zapisane u mikroprogramskoj memoriji
- mikroinstrukcija se sastoji od upravljačkog dijela, konstante koja se može koristiti za razne svrhe tijekom izvođenja instrukcije, dijela koji upravlja grananjem i dijela koji govori o tome na kojoj adresi se nalazi sljedeća mikroinstrukcija
- duljina mikroprogramske riječi NEMA veze s duljinom riječi mikroprocesora
- za realizaciju je potreban trofazni generator signala vremenskog vođenja:
  - P(0) – izvođenje mikroinstrukcije
  - P(1) – prijenos adrese u mikroprogramski adresni registar
  - P(2) – smještanje mikroinstrukcije u mikroinstrukcijski registar



– mikroprogramska memorija je u većini slučajeva skrivena od korisnika, mada ponekad može biti i dostupna korisniku; tada govorimo o mikroprogramirljivim mikroprocesorima – primjer: većina bit-slice procesora.

Primjer:

Upravljačka jedinica realizirana mikroprogramiranjem za pojednostavljeni model mikroprocesora. Operacijski kod instrukcije sastoji se od 6 bitova, tj. moguće je iskodirati 64 različite instrukcije.

Mikrooperacije:

A

R, R', W	čitanje, čitanje komplementa, pisanje

DC

R, R_L, R_H, W, W_L, W_H	čitanje (16 bita), čitanje (nižih 8 bita), čitanje (viših 8 bita), upis (16 bita), upis (nižih 8 bita), upis (viših 8 bita)

PC

R, R_L, R_H, W, W_L, W_H, INC	čitanje (16 bita), čitanje (nižih 8 bita), čitanje (viših 8 bita), upis (16 bita), upis (nižih 8 bita), upis (viših 8 bita), uvećanje za jedan

PR

R, R', W	čitanje, čitanje komplementa, pisanje

MAR

R, W	čitanje, pisanje

MDR

R, W	čitanje, pisanje

M

R, W	čitanje, pisanje

ALU

ADD, ADC, SUB	zbrajanje, zbrajanje s prijenosom

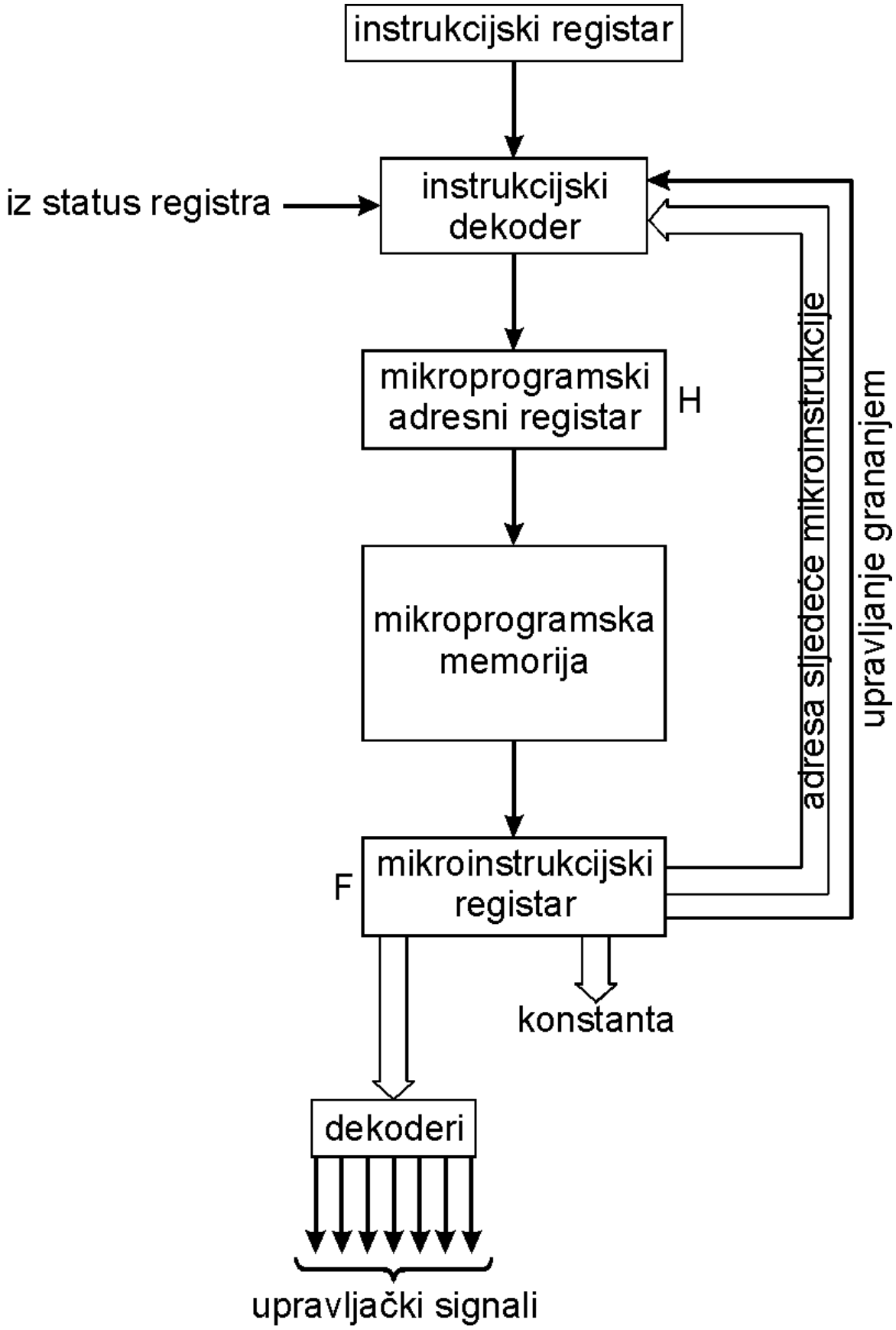
Pojednostavljenja:

- interna podatkovna sabirnica je 16 bitna, s tim da 8 bitni registri (A i PR) koriste nižih 8 bita, a 16 bitni registri su u stanju na sabirnicu staviti svih 16 bita (R, W) ili na nižih 8 bita staviti gornjih ili donjih 8 bita (R_H, W_H, R_L, W_L).
- ne vodimo računa o ograničenjima po pitanju veličine mikroprogramske memorije.

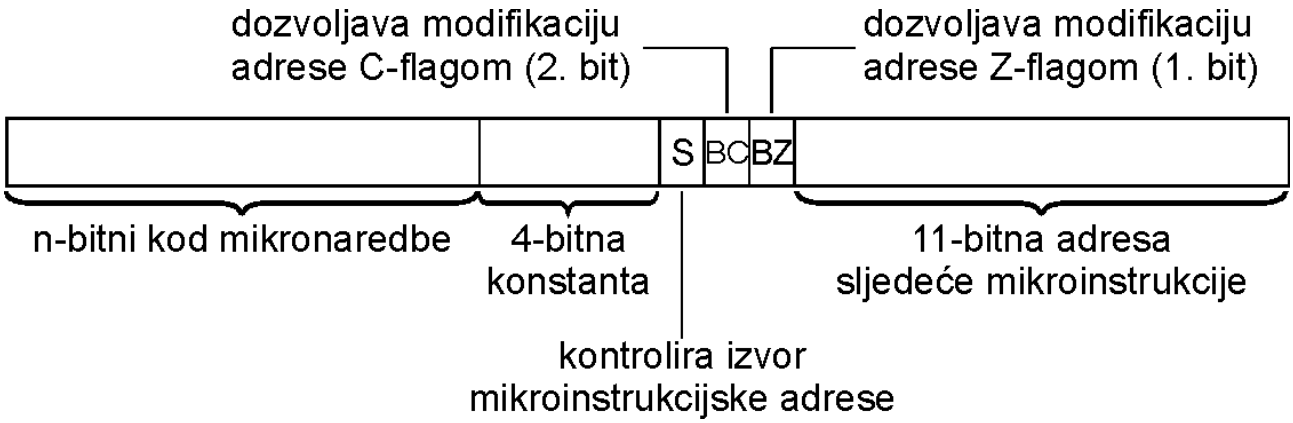
Dodatak na postojeći model:

- dodani su memorijski adresni registar (MAR) i memorijski podatkovni registar (MDR) koji služe kao sučelje prema vanjskim sabirnicama.

Struktura upravljačke jedinice:

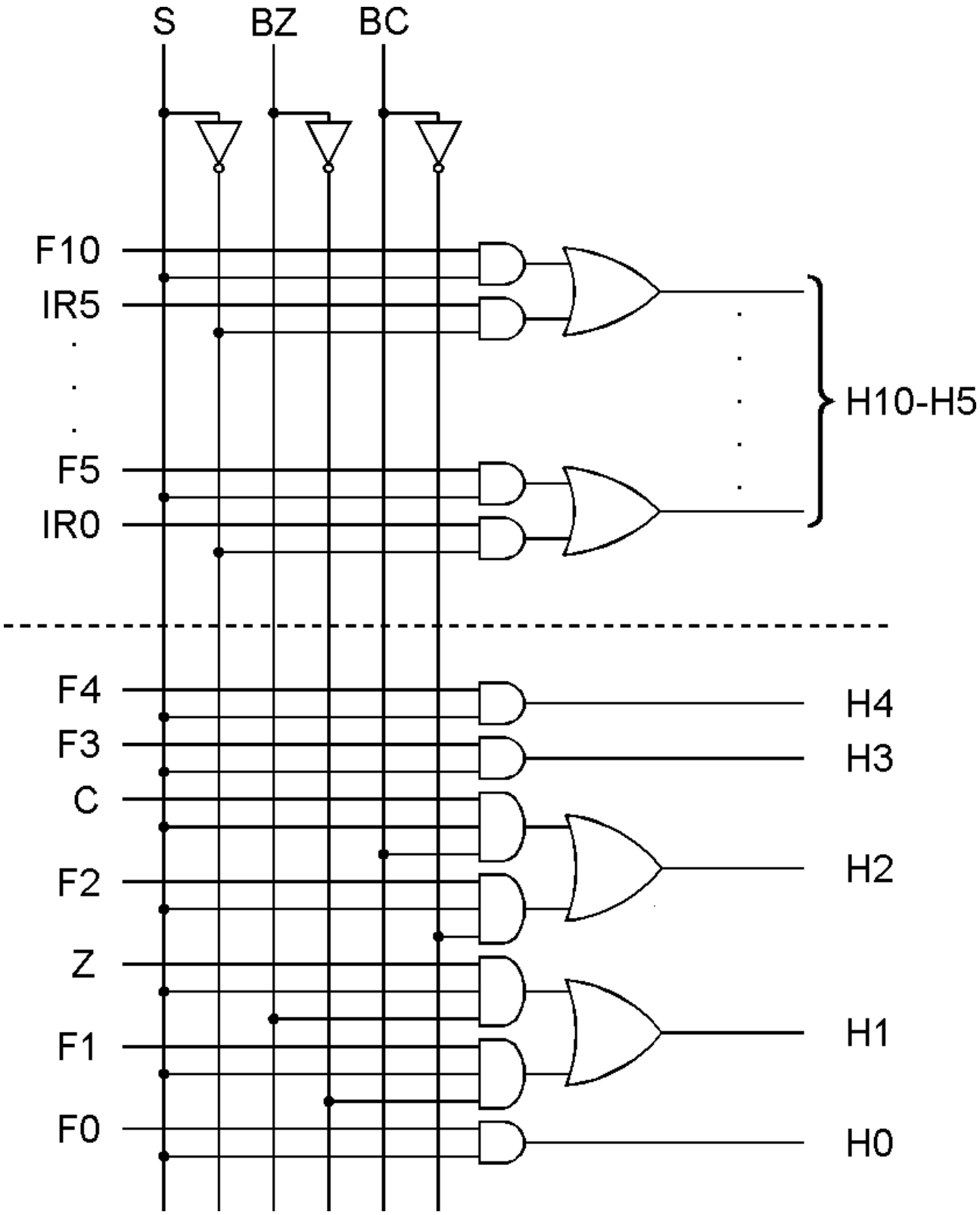


Struktura mikroinstrukcije:



- Kada je S=0 tada je viših 6 od 11 bitova mikroinstrukcijske adrese direktno preuzeto iz IR registra, a nižih 5 je 0.
- BC i BZ su bitovi koji kontroliraju grananja unutar mikroprograma na način da dozvoljavaju prepisivanje indikatora C i Z u 2 odnosno 1 bit mikroprogramske adrese.

Instrukcijski dekoder:



Realizacija instrukcija CP i BREQ mikroprogramiranjem:

CP 0xHHLL – uspoređuje sadržaj akumulatora sa podatkom koji se nalazi u memoriji na adresi 0xHHLL. Neka je op kod instrukcije CP 0x0F. Tada se ona asemblira u niz: 0x0F 0xLL 0xHH

BREQ 0xHHLL – ako je postavljen indikator Z (Z=1) tada nastavlja s izvršavanjem programa od adrese 0xHHLL, a ako nije tada nastavlja s izvršavanjem od sljedeće instrukcije. Neka je op kod instrukcije BREQ 0x1A. Tada se ona asemblira u niz: 0x1A 0xLL 0xHH.

Mikroprogram za dohvat op koda:

Adresa	**Mikrooperacije**	**S**	**BC**	**BZ**	**Sljedeća**
0x7F8	PC(R), MAR(W)	1	0	0	0x7F9
0x7F9	MAR(R), M(R), MDR(W)	1	0	0	0x7FA
0x7FA	MDR(R), IR(W)	1	0	0	0x7FB
0x7FB	PC(INC)	1	0	0	0x7FC
0x7FC	IR(R)	0	0	0	0x000

Mikroprogram za instrukciju CP:

Adresa	**Mikrooperacije**	**S**	**BC**	**BZ**	**Sljedeća**
0x1E0	PC(R), MAR(W)	1	0	0	0x1E1
0x1E1	MAR(R), M(R), MDR(W)	1	0	0	0x1E2
0x1E2	MDR(R), DC(W_L)	1	0	0	0x1E3
0x1E3	PC(INC)	1	0	0	0x1E4
0x1E4	PC(R), MAR(W)	1	0	0	0x1E5
0x1E5	MAR(R), M(R), MDR(W)	1	0	0	0x1E6
0x1E6	MDR(R), DC(W_H)	1	0	0	0x1E7
0x1E7	PC(INC)	1	0	0	0x1E8
0x1E8	DC(R), MAR(W)	1	0	0	0x1E9
0x1E9	MAR(R), M(R), MDR(W)	1	0	0	0x1EA
0x1EA	MDR(R), PR(W)	1	0	0	0x1EB
0x1EB	A(R), PR(R), ALU(SUB)	1	0	0	0x7F8

Mikroprogram za instrukciju BREQ:

Adresa	**Mikrooperacije**	**S**	**BC**	**BZ**	**Sljedeća**
0x340	PC(R), MAR(W)	1	0	1	0x341 0x343
0x341	PC(INC)	1	0	0	0x342
0x342	PC(INC)	1	0	0	0x7F8
0x343	MAR(R), M(R), MDR(W)	1	0	0	0x344
0x344	MDR(R), DC(W_L)	1	0	0	0x345
0x345	PC(INC)	1	0	0	0x346
0x346	PC(R), MAR(W)	1	0	0	0x347
0x347	MAR(R), M(R), MDR(W)	1	0	0	0x348
0x348	MDR(R), DC(W_H)	1	0	0	0x349
0x349	DC(R), PC(W)	1	0	0	0x7F8

Napomena: Tijekom dizajniranja mikroprograma uočavaju se određene sekvence mikrooperacija koje su zajedničke za više instrukcija. Kako bi se izbjeglo njihovo ponavljanje (zauzeće mikroprogramske memorije!) upravljačka jedinica se izvodi u dva nivoa:

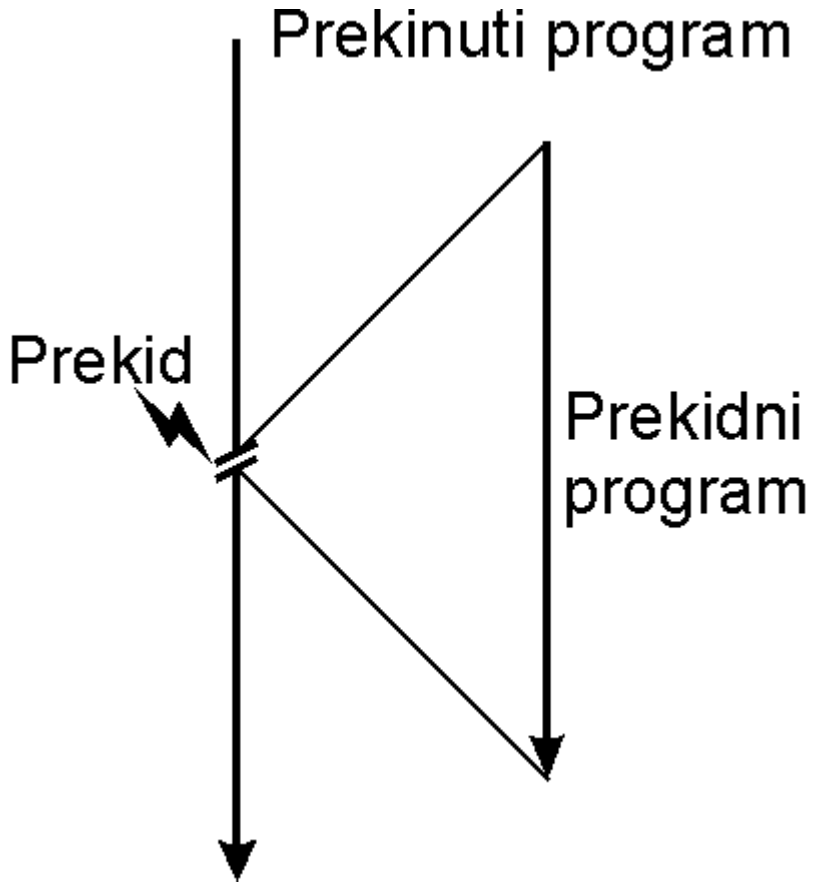
- mikroinstrukcijski nivo
- nanoinstrukcijski nivo

Prekidi i prekidni sustav

- Prekidi (eng. interrupt) i prekidni sustavi mikroprocesora omogućavaju da mikroprocesor brzo i efikasno odgovori na zahtjeve vanjskih uređaja.
- Proširivanjem prekidne logike došlo se do iznimaka (eng. exception) koje kod suvremenih procesora objedinjavaju:
  - ♦ prekide nastale od strane vanjskih (perifernih) uređaja
  - ♦ prekide nastale zbog grešaka pri obavljanju instrukcija:
    - greška pri adresiranju (pokušaj adresiranja memorijske lokacije koja ne postoji, pokušaj adresiranja memorijske lokacije koju program koji se izvodi ne smije adresirati i sl.)
    - greška zbog dijeljenja s nulom
    - greška zbog pokušaja izvođenja privilegirane instrukcije u korisničkom načinu
  - ♦ prekide izazvane instrukcijom (instrukcijama) koja je namijenjena prebacivanju procesora u stanje identično stanju obrade vanjskog prekida (primjer: instrukcije koje služe za pokretanje podprograma koji se obavljaju u nadglednom načinu - podprogrami operativnog sustava).

- Mikroprocesor ima jednu ili više kontrolnih linija preko kojih prima zahtjeve za prekid
- Prekidi mogu biti:
  - ♦ maskirajući
  - ♦ nemaskirajući

Shematski prikaz prekida:



Postupak obrade prekida od strane mikroprocesora:

- Periferni uređaji šalju mikroprocesoru **zahtjev za prekid** (važno je uočiti da je ovaj događaj asinkron i da se može dogoditi u bilo kojem trenutku)
- Ako mikroprocesor prihvaća prekid (prihvaća ga uvijek ako prekid nije maskiran) **na kraju izvršenja tekuće instrukcije** odgovara **signalom potvrde prekida** i **onemogućava dalje prekide** (iste ili niže razine).
- **Pohranjuje** se sadržaj **PC**-a i pokreće se **prekidni program**. Ako mikroprocesor to omogućava automatski se pohranjuje i sadržaj svih radnih i statusnih registara. Pohrana se obavlja na stogu.

- Na početku prekidnog programa, ako to mikroprocesor automatski ne obavlja, potrebno je pohraniti sadržaj svih radnih i statusnih registara.
- Na kraju prekidnog programa obnavlja se sadržaj svih radnih i statusnih registara, **omogućuju se naredni prekidi** (tek od iduće instrukcije) i vrši se **povratak u prekinuti program**.

U slučaju da imamo više U/I uređaja nego prekidnih linija, tada je potrebno odrediti koji U/I uređaj je prouzročio prekid. To se radi na jedan od dva načina:

- metodom prozivanja prekida – mikroprocesor proziva jedan po jedan uređaj pokušavajući ustanoviti koji od njih je zahtijevao prekid
- metodom vektorskog prekida – uređaj koji je izazvao prekid na neki način postavlja svoju identifikaciju koja omogućava direktan poziv prekidnog programa namjenjenog obradi baš njegovog zahtjeva.

Pri analizi prekidnog sustava mikroprocesora potrebno se rukovoditi sljedećim pitanjima:

- Koliko prekidnih linija posjeduje mikroprocesor?
- Kakav je odaziv mikroprocesora na prekid (sabirnički ciklus potvrde prekida)?
- Na koji način mikroprocesor određuje uzročnika prekida ako ima više izvora nego prekidnih linija?
- Da li postoji prioritetna struktura prekida (prekidi u više nivoa prioriteta)?
- Kako i kada se prekid omogućuje ili onemogućuje?

MEMORIJA

Osnovna podjela memorija obzirom na mogućnost promjene njihovog sadržaja:

- memorija sa slobodnim pristupom
  - ♦ RAM – eng. Random Access Memory
- ispisna memorija
  - ♦ ROM – eng. Read Only Memory
- uglavnom ispisna memorija
  - ♦ PROM – eng. Programmable Read Only Memory
  - ♦ EPROM – eng. Erasable Programmable Read Only Memory
  - ♦ EEPROM – eng Electricaly Erasable Programmable Read Only Memory

RAM je oznaka kojom se uglavnom označava glavna ili radna memorija mikroračunala. Slobodni pristup (eng. random access) znači da se svakoj adresi može direktno pristupiti, za razliku od memorija koje imaju slijedni (eng. sequential) pristup. U ovoj memoriji se nalazi većina programa koji mikroprocesor izvršava, te međurezultati i rezultati obrada.

ROM je oznaka za memoriju kojoj se jednako kao i RAM memoriji može pristupati direktno (vrijeme pristupa ne zavisi od adrese kojoj se pristupa), ali se sadržaj s pojedine lokacije može samo čitati.

Skupina **uglavnom ispisnih memorija** predstavlja memorije kojima je namjena vrlo slična ROM memoriji, ali se daju programirati (jednokratno PROM ili višekratno EPROM, EEPROM) od strane korisnika.

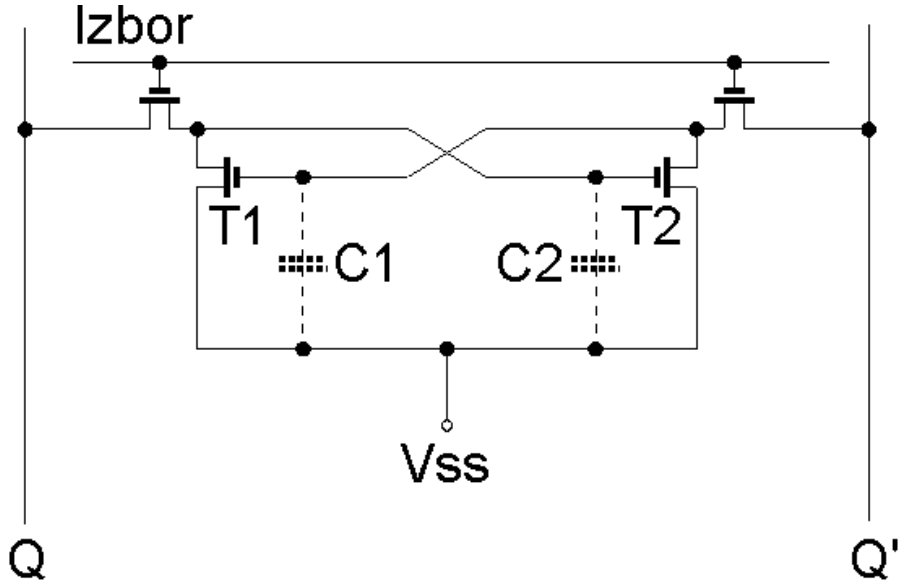
Obzirom na izvedbu RAM memoriju dijelimo na:

- Dinamički RAM (DRAM)
- Statički RAM (SRAM)

Usporedba svojstava DRAM-a i SRAM-a:

- DRAM “pamti” bit na osnovi kapacitivnosti tj. naboja u kondenzatoru, a SRAM “pamti” stanje u bistabilu.
- DRAM za pamćenje jednog bita treba manje tranzistora od SRAM-a, te je stoga jeftiniji i danas najčešće korišten za glavnu radnu memoriju u suvremenim mikroračunalima.
- DRAM zahtjeva “osvježavanje” što ga čini sporijim od SRAM-a, a zahtjeva i dodatno sklopovlje za osvježavanje.
- Informacija pohranjena u DRAM-u je vrlo “osjetljiva i slaba” jer je zasnovana na vrlo malom naboju koji je pohranjen u kondezator vrlo malene kapacitivnosti.

Osnovna građevna jedinica DRAM memorije:



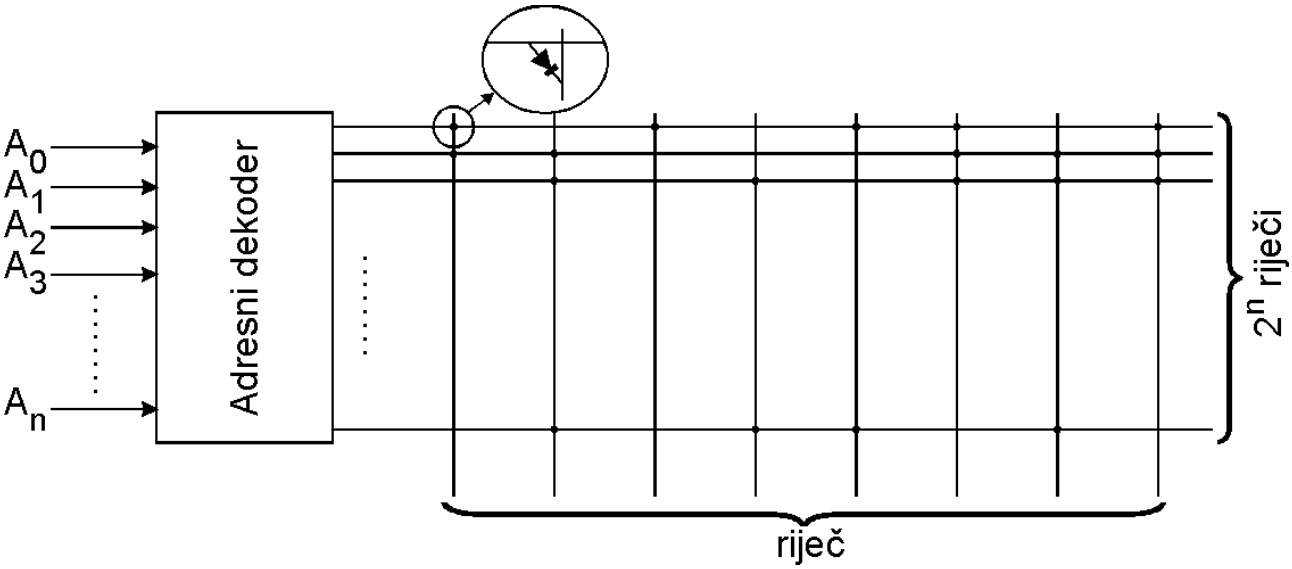
Pisanje:

Q → 1, Q' → 0 } T2 → zatvoren, T1 → otvoren } C1 → nabija se, C2 → izbija se

Čitanje:

C1 → nabijen, C2 → izbijen } T1 → otvoren, T2 → zatvoren } Q → Vss, Q' → C1(vrt. nula)

ROM memorije se uglavnom izvode kao diodna matrica, kao što je to prikazano slikom:



Dok se kod uglavnom ispisnih memorija koriste druge tehnike za realizaciju prospojnih elemenata npr:

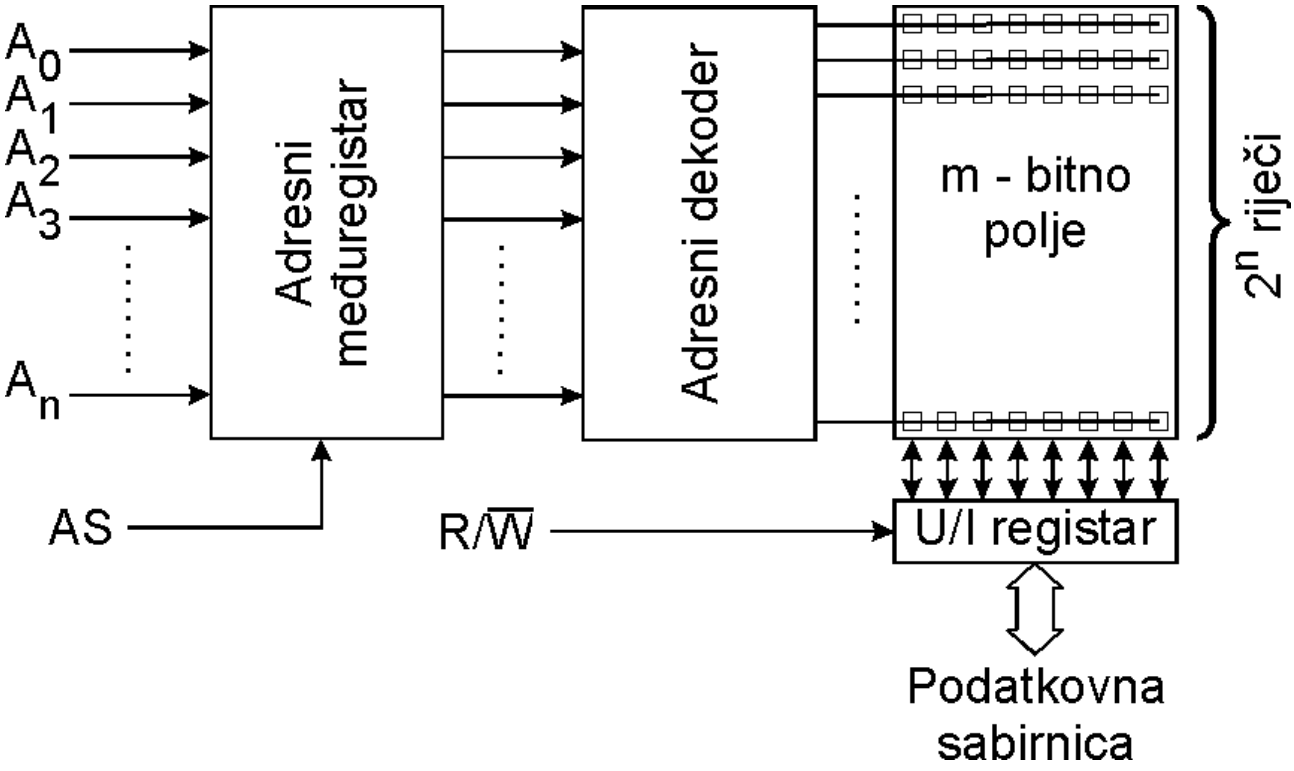
- PROM – koristi pregorljive veze (fusible links)
- EPROM i EEPROM – posebne elemente koji se mogu prebaciti iz provodljivog u neprovodljivo stanje i obratno te to svoje stanje zadržavaju bez obzira na napajanje.

Organizacija memorije

Obzirom na način na koji se odabiru pojedine memorijske ćelije (jedna ćelija sadrži jedan bit) memorije dijelimo na:

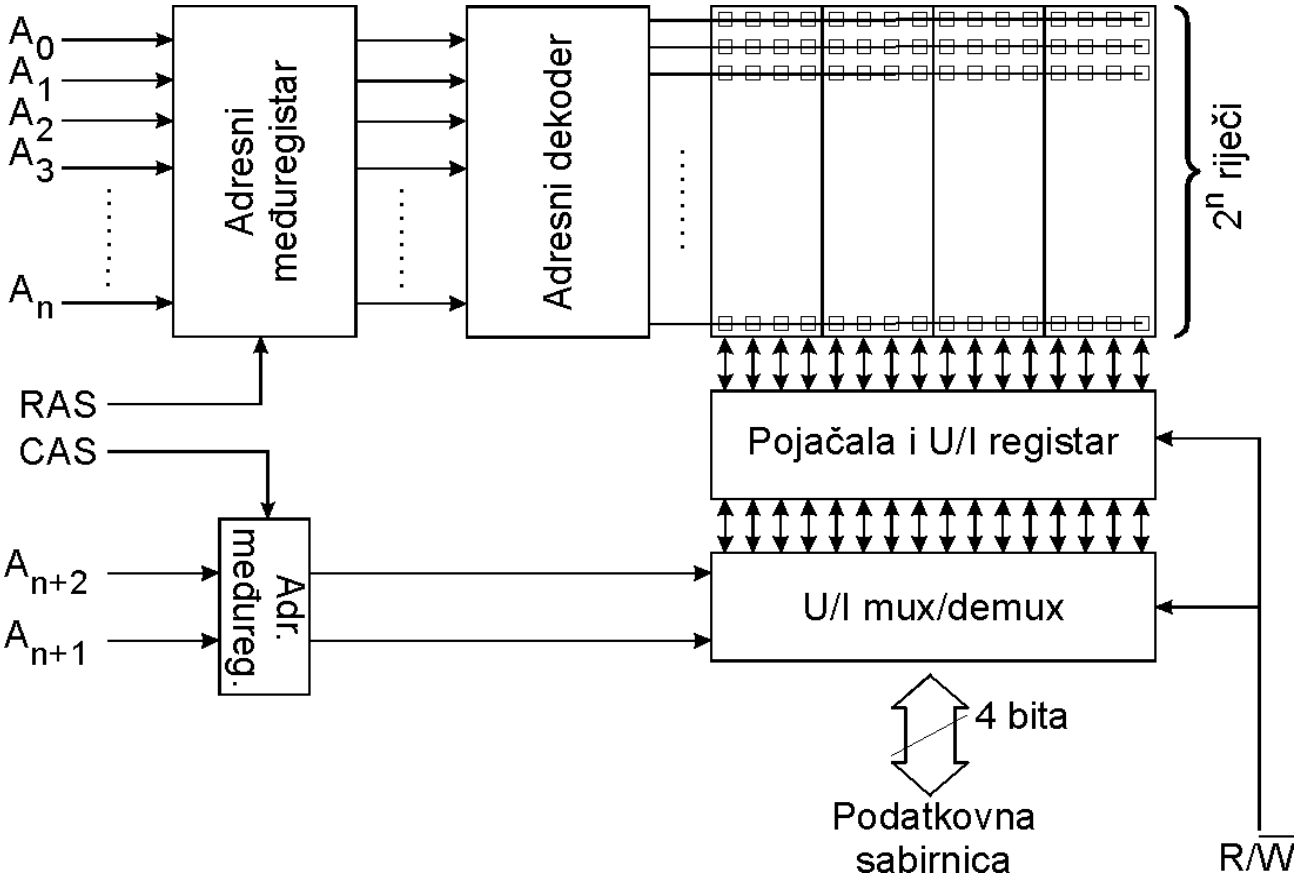
- s linearnim adresiranjem (2D memorije)
- s koincidentnim adresiranjem (2 1/2D memorije)

Memorija s linearnim adresiranjem:



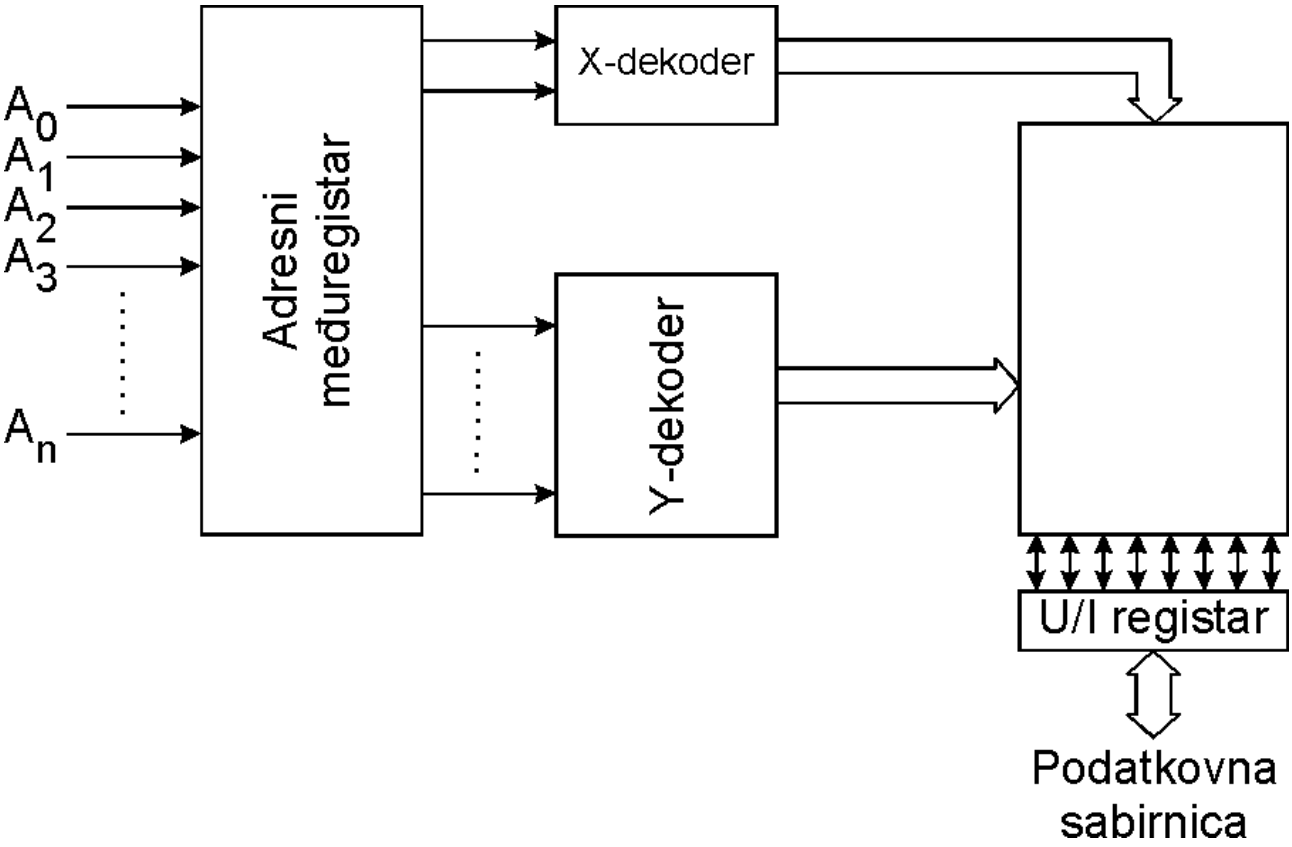
– AS (Address Strobe) je signal koji govori da je signal na adresnoj sabirnici postavljen i da je stabilan i spreman za uporabu.

Memorija s koincidentnim adresiranjem:

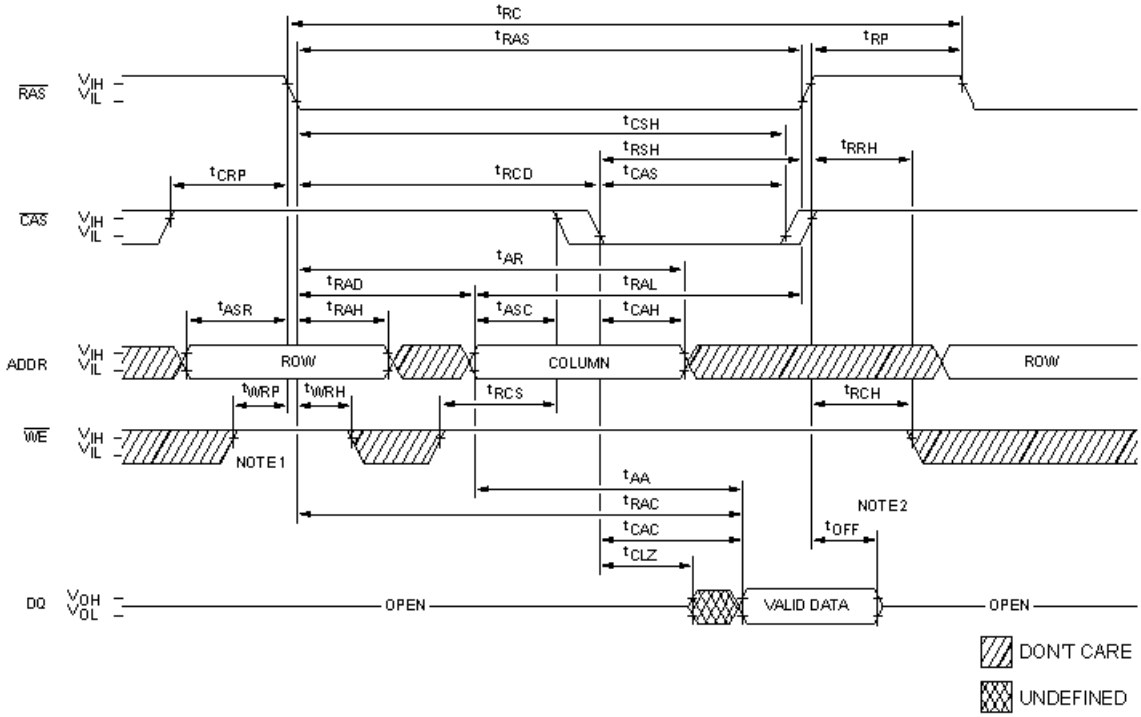


- Kako je adresa podijeljena u dva dijela (tj. adresu kolone i adresu retka) tako su potrebna i dva signala koji zamjenjuju AS signal. To su RAS (Row Address Strobe) koji kazuje da je postavljena adresa retka i CAS (Column Address Strobe) koji kazuje da je postavljena adresa kolone.
- Često se koriste iste linije na sabirnici za prijenos adrese retka i kolone. To se radi na način da se prvo postavi jedna adresa, a zatim druga, a signali RAS i CAS kazuju koja je adresa na sabirnici.

Memorije s koincidentnim adresiranjem često se prikazuju shematski ovako:



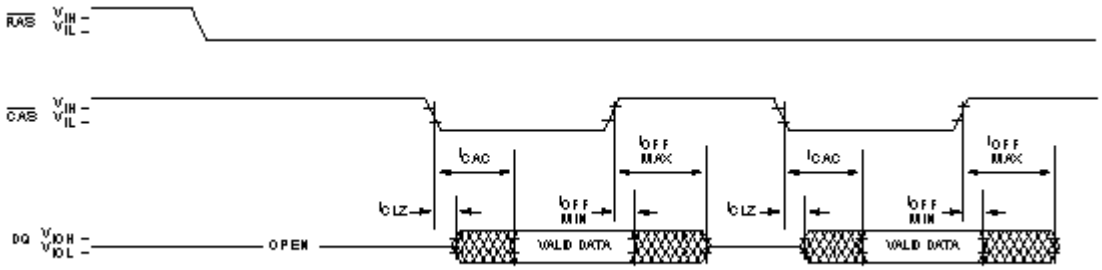
Većina memorijskih chipova imaju U/I registre izvedene kao registre s 3 stanja i nožice za odabir chipa, što znači da se chip, kada nije adresiran, povlači sa podatkovne sabirnice.



Dijagram tipičnog ciklusa čitanja DRAM memorije

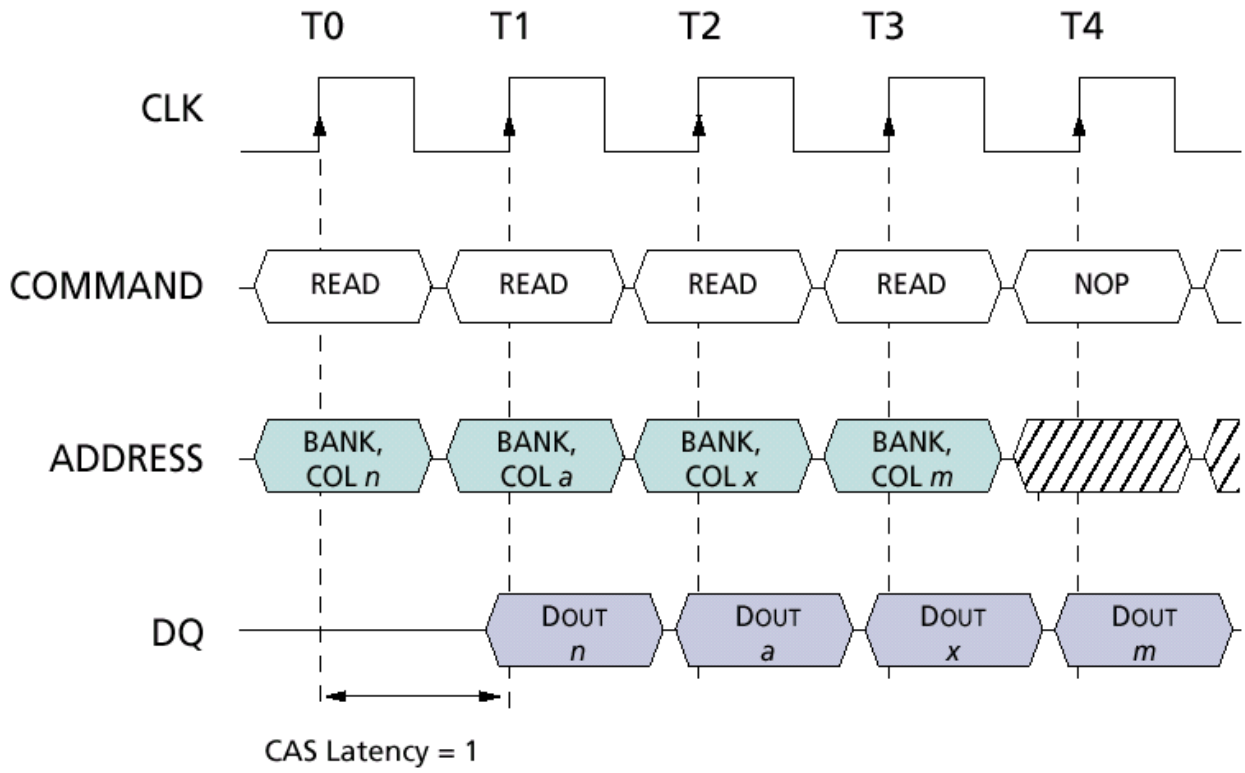
Naprednije vrste DRAM memorije koje se koriste u današnjim mikroračunalima uglavnom postižu veće brzine primjenom nekih od tehnika za smanjenje **prosječnog** vremena pristupa:

– FPMDRAM (Fast Paging Mode DRAM) – omogućava mijenjanje adrese kolone dok se adresa retka podrazumjeva iz prethodnog pristupa.



Čitanje sadržaja dvaju uzastopnih memorijskih adresa

– EDO DRAM (Enhanced Data Output DRAM) – poboljšanje FPDRAM-a
– SDRAM (Synchronious DRAM) – DRAM koji postiže veće brzine zahvaljujući sinkroniziranju s procesorom.



SABIRNICE

- Unutarnja (interna) sabirnica

  Sabirnica koja povezuje elemente mikroprocesora (registre, ALU...) i nalazi se unutar mikroprocesora tj. na samom chipu.
- Vanjska (eksterna) sabirnica

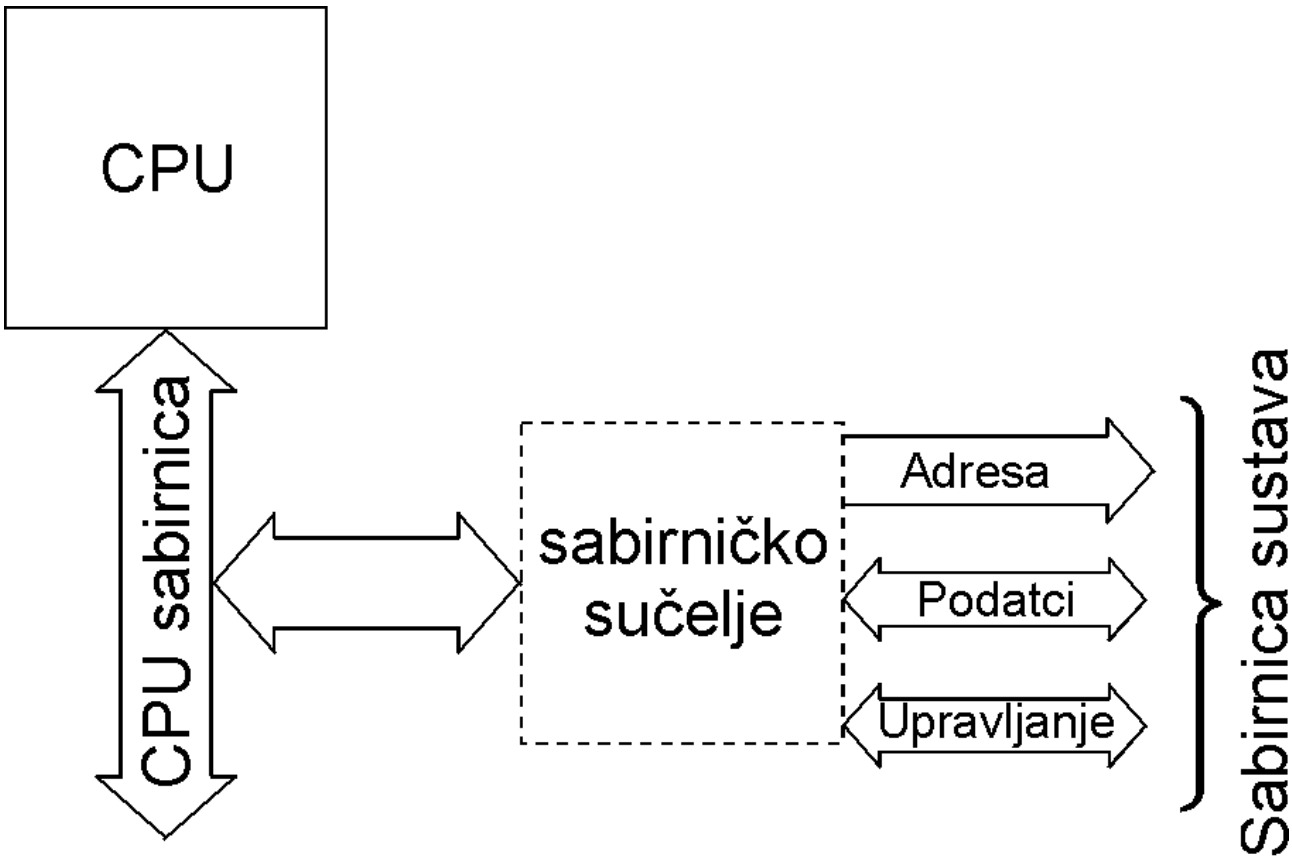
  Sabirnica na koju se povezuju ostali elementi mikroračunala, a spaja se na internu sabirnicu preko nožica mikroprocesora.

- Vanjska sabirnica se dijeli na:
  - ♦ sabirnicu centralne procesorske jedinice – sabirnica koja je direktno spojena na mikroprocesor tj. linije koje direktno izlaze iz mikroprocesora. Ova sabirnica je često vremenski multipleksirana.
  - ♦ sabirnicu sustava – sabirnica koja povezuje većinu elemenata u sustavu. Linije sabirnice sustava su dobivene demultipleksiranjem sabirnice CPU-a tj. CPU sabirnica je povezana na sabirničko sučelje, a ono generira signale na sabirnici sustava.

Sabirnica CPU-a i sabirnica sustava kod jednostavnijih mikroprocesora i mikroračunalnih sustava može biti jedna te ista sabirnica.

Obzirom na namjenu linije koje tvore sabirnicu se dijele u tri skupine:

- adresni dio, ili adresna sabirnica (eng. address bus)
- podatkovni dio, ili podatkovna sabirnica (eng. data bus)
- upravljački dio ili upravljačka sabirnica (eng. control bus)



Odnos CPU sabirnice i sabirnice sustava

- Svi sklopovi koji se prikapčaju na sabirnicu moraju biti opremljeni izlazima koji se mogu postaviti u visokoimpedantno stanje.
- Obzirom na električka svojstva sabirnice (otpor, parazitna kapacitivnost i induktivitet) za prijenos podataka po sabirnici koriste se sabirnička pojačala (eng. bus driver).

U jednom trenutku preko sabirnice mogu komunicirati samo dva uređaja prikopčana na sabirnicu. Svi ostali uređaji moraju biti efektivno električki odspojeni od sabirnice i nalaziti se u visokoimpedantnom stanju.

Sklopovi koji se prikapčaju na sabirnicu po svojoj funkciji mogu biti:

- vodeći moduli (eng. bus master)
  - ♦ trajno vodeći moduli (npr. CPU)
  - ♦ privremeno vodeći moduli (npr. DMA)
- prateći moduli (eng. bus slave)

Vodeći modul upravlja sabirnicom i odgovoran je za sve sabirničke aktivnosti. Generira sve signale potrebne za adresiranje, izbor pratećih modula i prijenos podataka.

Prateći moduli "osluškuju" kontrolnu i adresnu sabirnicu i kad ih vodeći modul adresira tada u skladu sa sabirničkim protokolom odgovaraju i učestvuju u prijenosu podataka.

Sabirničke aktivnosti su podijeljene u sabirničke cikluse. Najčešći ciklusi su:

- ciklus prijenosa podataka
- ciklus potvrde prekida

Prema načinu djelovanja sabirnice se dijele na:

- asinkrone
  - ♦ događaji na njoj su uvelike nezavisni od signala vremenskog vođenja
  - ♦ događaji se sinkroniziraju koristeći dogovoreno "rukovanje" (eng. handshake) signalima potvrde (eng. acknowledge)
  - ♦ brzina rada prilagođena je svakom individualnom modulu

- sinkrone
  - ♦ događaji na njoj u potpunosti ovise o signalu vremenskog vođenja
  - ♦ događaji se sinkroniziraju na osnovi broja ciklusa signala vremenskog vođenja
  - ♦ brzina rada prilagođena je najsporijem modulu koji se nalazi na sabirnici tj. propisano je najdulje vrijeme unutar kojeg svaki modul koji se prikapča na sabirnicu mora biti sposoban obaviti pojedinu operaciju

U/I OPERACIJE

Prijenos podataka između mikroračunala i vanjske logike (drugo računalo, razni uređaji za nadzor i upravljanje procesima, printer...) obavlja se korištenjem U/I međusklopova (eng. I/O interface).

Uloga U/I sklopova:

- međupohranjivanje (eng. buffering)
- dekodiranje adrese ili izbor uređaja
- dekodiranje komandi
- vremensko vođenje i upravljanje

Adresiranje i izbor U/I sklopova može biti:

- izdvojeno
  - ♦ postoje posebne instrukcije za prijenos podataka iz/u U/I sklop
  - ♦ u kontrolnoj sabirnici nalazi se linija (linije) koje govore o tome da li je na adresnoj sabirnici adresa memorijske lokacije ili U/I sklopa
- memorijskim preslikavanjem
  - ♦ ne postoje posebne instrukcije za prijenos podataka iz/u U/I sklop, tj. mikroprocesor registre U/I sklopa tretira kao memorijske lokacije u svom memorijskom prostoru. Shodno tome, na tim adresama ne može se nalaziti memorijski prostor
  - ♦ ne postoje posebne kontrolne linije koje govore o tome da li se adresira U/I međusklop ili memorija

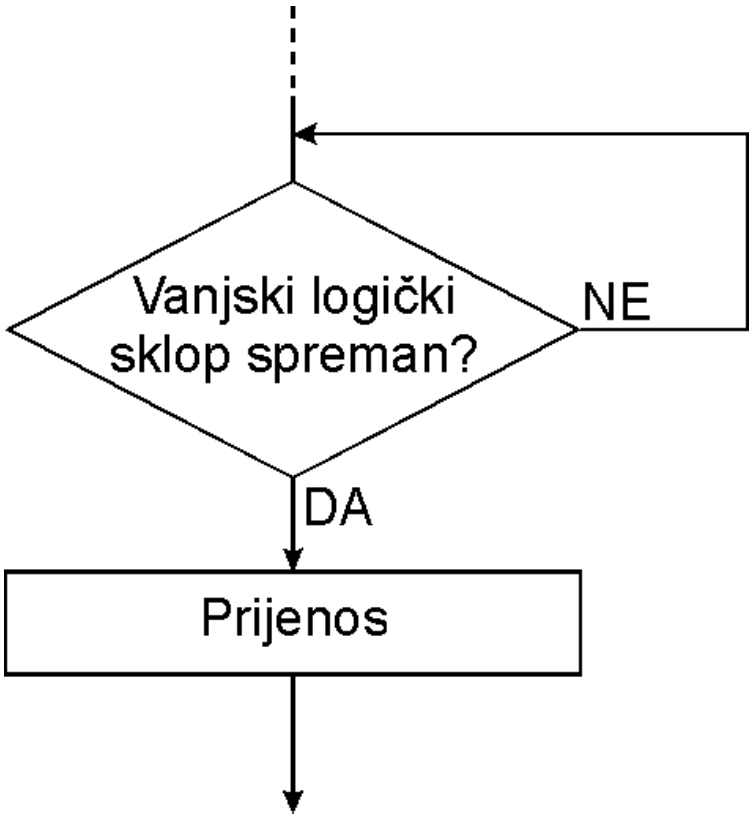
Način izmjene podataka između U/I sklopa i mikroračunala može se obavljati na jedan od tri načina:

- programirani prijenos
- prekidni prijenos
- prijenos izravnim pristupom memoriji (DMA, eng. Direct Memory Access)

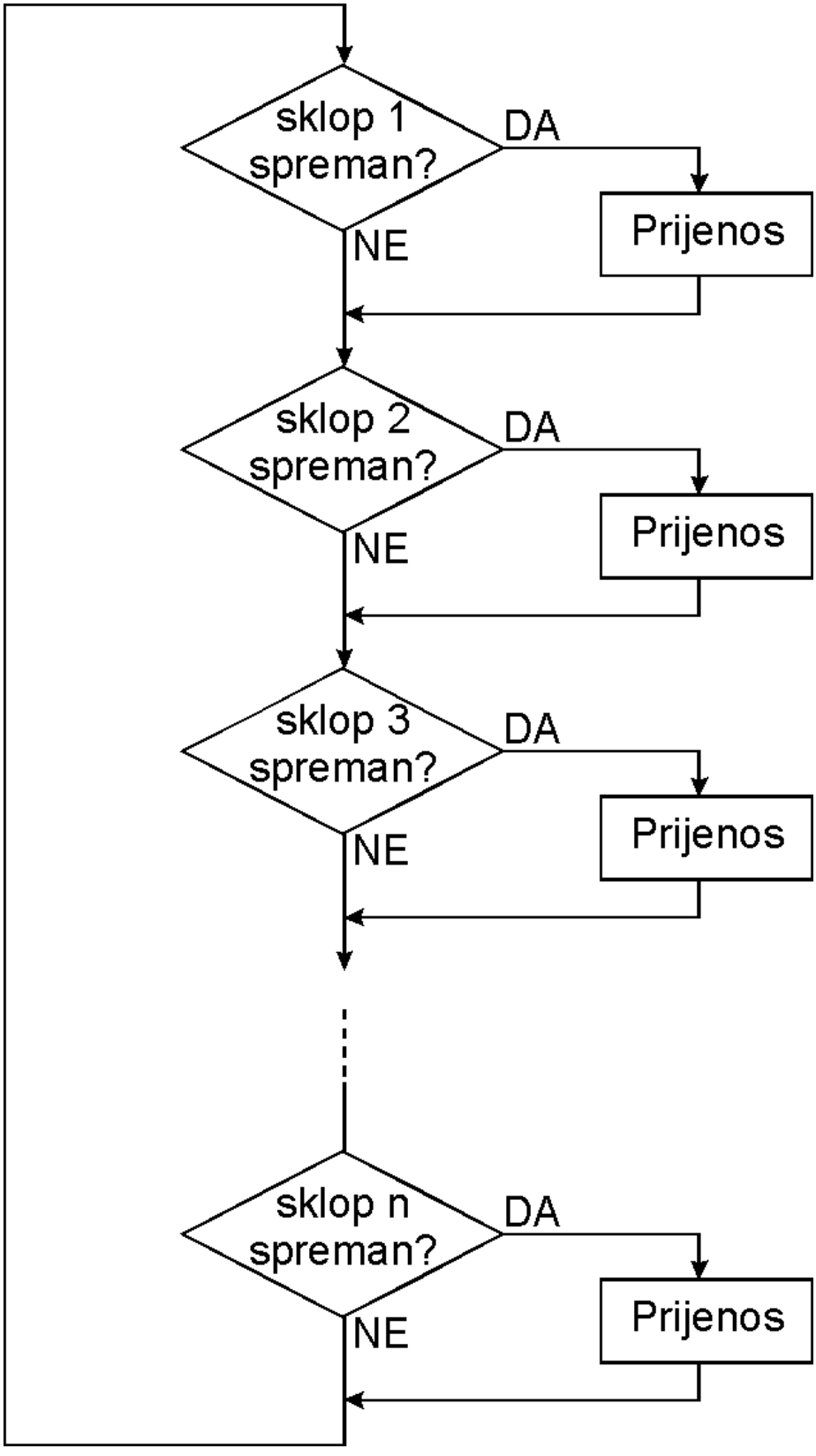
Programirani prijenos:

- procesor obavlja sav posao oko prijenosa podataka između vanjskog logičkog sklopa i ostatka mikroračunala
- prijenos se obavlja u "glavnom" programu
- može biti bezuvjetan (gotovo se nikada ne koristi) ili uvjetan

Uvjetni prijenos podataka:



Ako postoji više U/I uređaja za koje je potrebno obavljati prijenos tada se koristi “prozivanje” (eng. polling):



Prekidni prijenos:

- procesor također obavlja sav posao oko prijenosa podataka između vanjskog logičkog sklopa i ostatka mikroračunala
- prijenos se obavlja u prekidnom programu, dok "glavni" program ne mora ni biti svjestan prijenosa
- ako na jednoj liniji za zahtijevanje prekida imamo više od jednog U/I sklopa tada se koriste metode:
  - ♦ vektorskog prekida
  - ♦ prozivanja

Prijenos izravnim pristupom memoriji (DMA):

- prijenos podataka između memorije i vanjskog logičkog sklopa NE OBAVLJA se pod vodstvom procesora već pod vodstvom posebnog sklopa DMA.
- ovo je najbrži način za obavljanje prijenosa podataka između periferije i memorije
- za vrijeme prijenosa podataka DMA sklop postaje privremeno vodeći modul na sabirnici što se ostvaruje jednom od metoda:
  - ♦ zaustavljanje procesora

    nakon što završi s izvršavanjem tekuće instrukcije mikroprocesor predaje sabirnicu DMA sklopu i ostaje zaustavljen za cijelo vrijeme trajanja prijenosa
  - ♦ krađa ciklusa

    u bilo kojem trenutku (čak i tijekom izvođena instrukcije) DMA može blokirati signal vremenskog vođenja koji ide prema

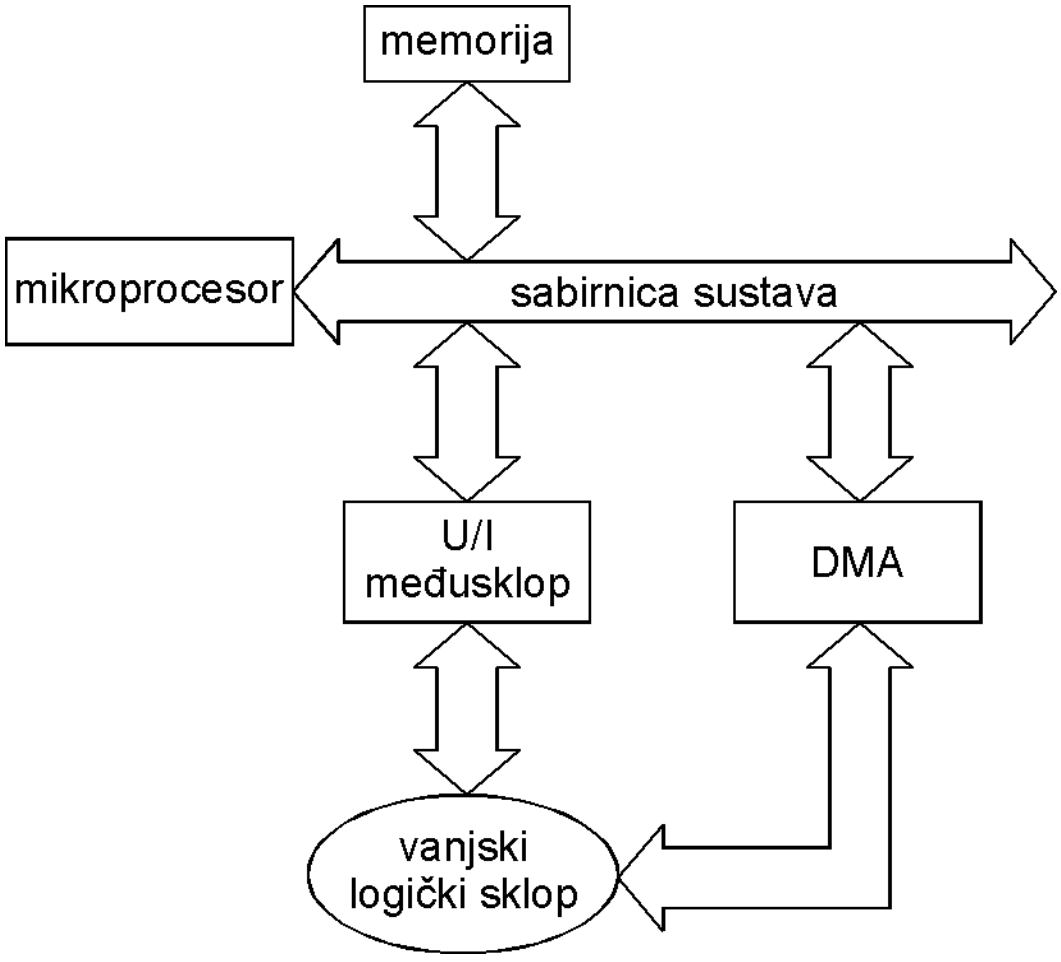
procesoru i tako ga efektivno zaustaviti na jedno kraće vrijeme potrebno da obavi prijenos podataka

- kombinacija zaustavljanja procesora i krađa ciklusa

  procesor se zaustavlja, ali samo za vrijeme potrebno za prijenos jednog bajta

- multipleksiranje operacija CPU/DMA

  ova metoda je moguća samo kada se upotrebljava memorija koja je dvostruko brža od procesora; tada je moguće za vrijeme jedne periode signala vremenskog vođenja obaviti dva pristupa u memoriju (čitanja/pisanja) tj. jedan prijenos može obaviti procesor, a drugi DMA



Shematski prikaz izravnog pristupa memoriji (DMA)

FP ARITMETIKA

Za računanje s realnim brojevima u digitalnim računalima se upotrebljava njihova **aproksimacija** u vidu brojevnog zapisa tehnikom pomičnog zareza (eng. floating point).

Starija mikroračunala su za računanje s ovim zapisom koristila odgovarajuće podprograme, a suvremeniji mikroprocesori posjeduju specijalizirani dio namijenjen obavljanju aritmetičkih operacija nad brojevima zapisanim u pomičnom zarezu.

Reprezentacija realnih brojeva:

- u decimalnom sustavu

  $91.25 = 9*10^1+1*10^0+2*10^{-1}+5*10^{-2}=9,125*10^1$

- u binarnom sustavu

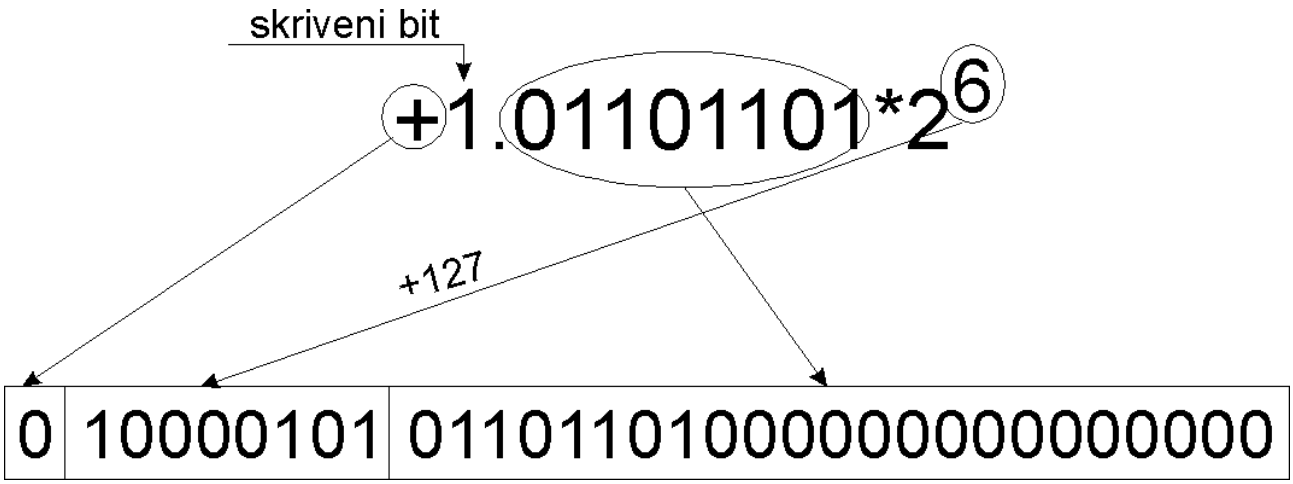
  $1011011.01=1*2^6+0*2^5+1*2^4+1*2^3+0*2^2+1*2^1+1*2^0+0*2^{-1}+1*2^{-2}$

  $=1.01101101*2^6$

Standard IEEE 754 propisuje način zapisa realnih brojeva u pomičnom zarezu:

- s jednostrukom preciznošću
  - ♦ 1 bit predznaka – +=0; -=1
  - ♦ 8 bita za eksponent
  - ♦ 23 bita za mantisu

– s dvostrukom preciznošću
  - ♦ 1 bit predznaka - +=0; -=1
  - ♦ 11 bitova za eksponenet
  - ♦ 52 bita za mantisu



Način zapisa realnog broja u IEEE 754 standardu (jednostruka preciznost)

Specijalni slučajevi:

Jednostruka preciznost		Dvostruka preciznost		Predstavlja
Eksponent	Mantisa	Eksponent	Mantisa	
0	0	0	0	Nula
0	nije nula	0	nije nula	denormalizirani broj
255	0	2047	0	beskonačno
255	nije nula	2047	nije nula	NaN (nije broj)

Denormalizirani broj:

- kada je eksponent 0 tada to ne predstavlja $1.xxxxx*2^{-127}$ već se radi o broju $0.xxxxx*2^{-126}$ što predstavlja denormalizaciju u odnosu na normalni način zapisa u FP-u.

Pretvorba decimalnog broja u FP zapis:

- pretvoriti cijeli dio
- pretvoriti decimalni dio
- normalizirati

Primjer:

48,640625

pretvorba cijelog dijela:	pretvorba decimalnog dijela:
48:2=24 0	.640625*2= 1.28125
24:2=12 0	.28125*2= 0.5625
12:2=6 0	.5625*2=1.125
6:2=3 0	.125*2=0.25
3:2=1 1	.25*2=0.5
1:2=0 1	.5*2=1
$(48)_{10}=(110000)_2$	$(.640625)_{10}=(.101001)_2$

$(48.640625)_{10}=(110000.101001)_2$

normalizacija:

$1.10000101001*2^5$

zapis u FP-u:

0 10000100 10000101001000000000000

Operacije nad FP brojevima:

- zbrajanje i oduzimanje:
  - ♦ dovesti brojeve na zajednički eksponent posmicanjem mantise broja koji ima manji eksponent u desno
  - ♦ zbrojiti/oduzeti mantise
  - ♦ provesti normalizaciju
- množenje i dijeljenje:
  - ♦ denormalizirati
  - ♦ pomnožiti/podijeliti mantise
  - ♦ zbrojiti/oduzeti eksponente
  - ♦ normalizirati

Primjer:

$1101.101=1.101101*2^3=13.625$

$100.11=1.0011*2^2=4.75$

Zbrajanje:

zajednički eksponent je 3

```
  101101
 +100110
  1010011
```

potrebno je provesti normalizaciju pa je mantisa rezultata: 0010011, a eksponent: 4, te nam rezultat glasi:

$1.0010011*2^4=10010.011=18.375$

Množenje:

1101101*10011=100000010111

zbroj eksponenata nakon denormalizacije: 4+3=7

rezultat:

$0.100000010111*2^7=1.00000010111*2^6$

$(1.00000010111*2^6)_2=(64.71875)_{10}$

Dijeljenje:

- Ako je djelitelj 0 a djeljenik različit od nule tada je rezultat plus/minus beskonačno
- Ako je i djelitelj i dijeljenik jednak 0 tada je rezultat NaN

Mantisa je 8 bitna, djeljenika proširujemo do 16 bita pri dijeljenju

$0.11011010*2^4:0.10011000*2^3$

mantisa djeljenika:

1101101000000000

mantisa djelitelja

1101101000000000:10011000=101101111

$1.01101111*2^{(4-3)}= 1.01101111*2^1$

$(1.0110111*2^1)_2=(2.8671875)_{10}$

Voditi računa o eventualno potrebnoj normalizaciji.