PONAVLJANJE
Ishodi 1 - 5

MI1

1.	Prepoznati elemente matične ploče osobnog računala	Identificirati strukturu i elemente matične ploče osobnog računala.
2.	Imenovati osnovne module i sklopove procesora	Objasniti arhitekturu, module i sklopove procesora
3.	Objasniti faze izvođenja intrukcija i stanje na sabirnicama	Prikazati na zadanom zadatku izvođenje instrukcija i stanje na sabirnicama

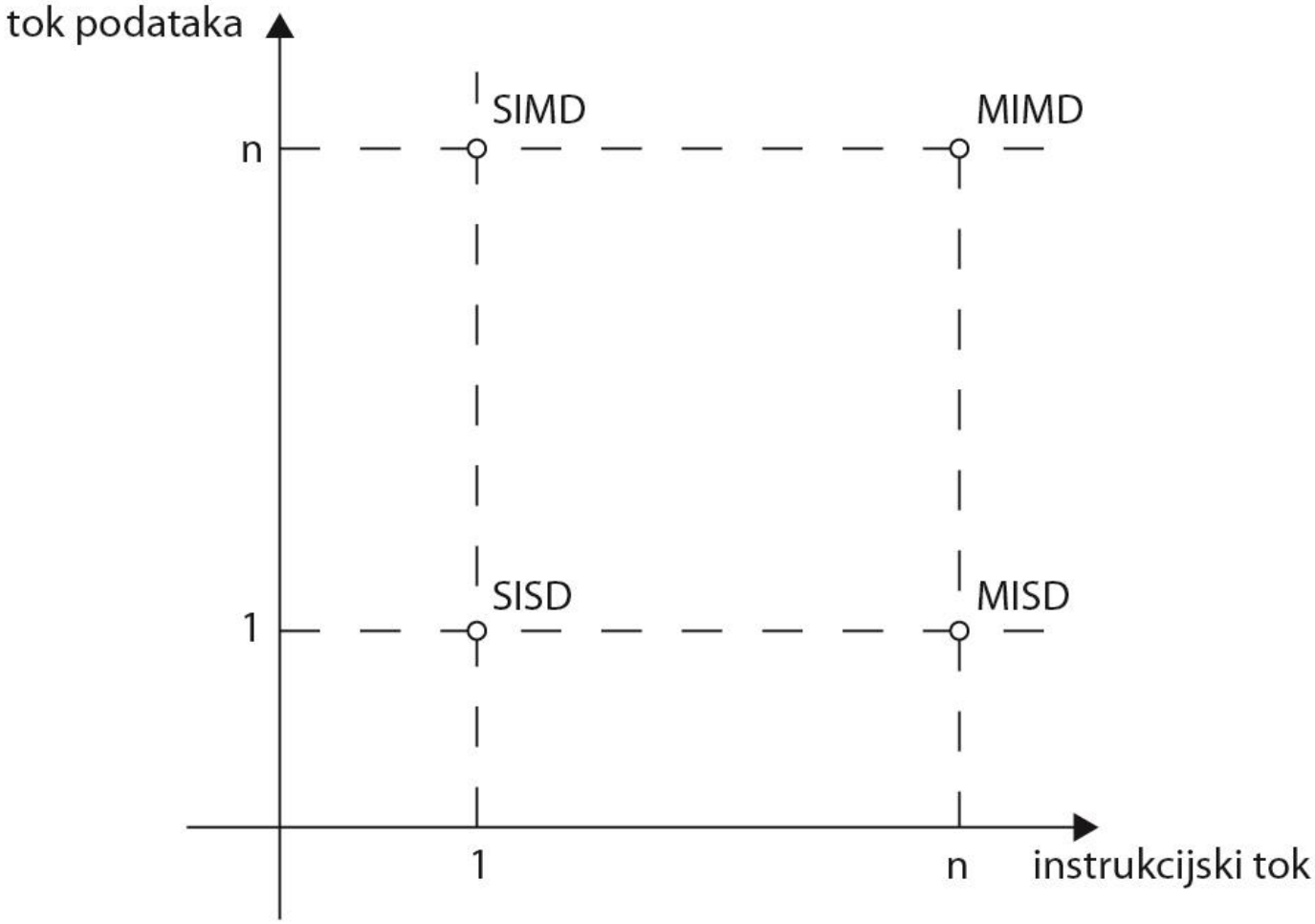
Ishod 1

1.	Prepoznati elemente matične ploče osobnog računala	Identificirati strukturu i elemente matične ploče osobnog računala.

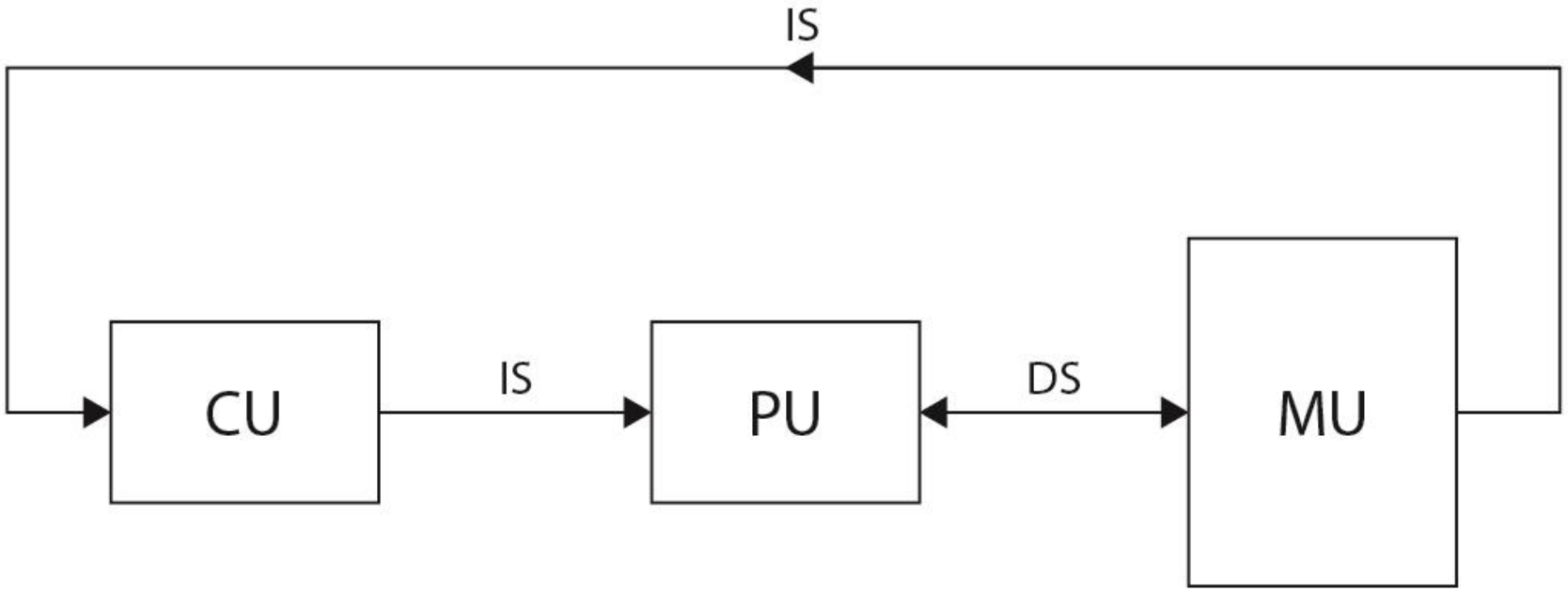
Flynnova klasifikacija

- Temelji se na:
  - instrukcijskom toku (slijedu instrukcija koje izvršava procesor)
  - toku podataka (slijedu podataka povezanim s instrukcijskim tokom)
- Postoje četiri osnovna tipa arhitekture
  - SISD
  - MISD
  - SIMD
  - MIMD

Flynnova klasifikacija



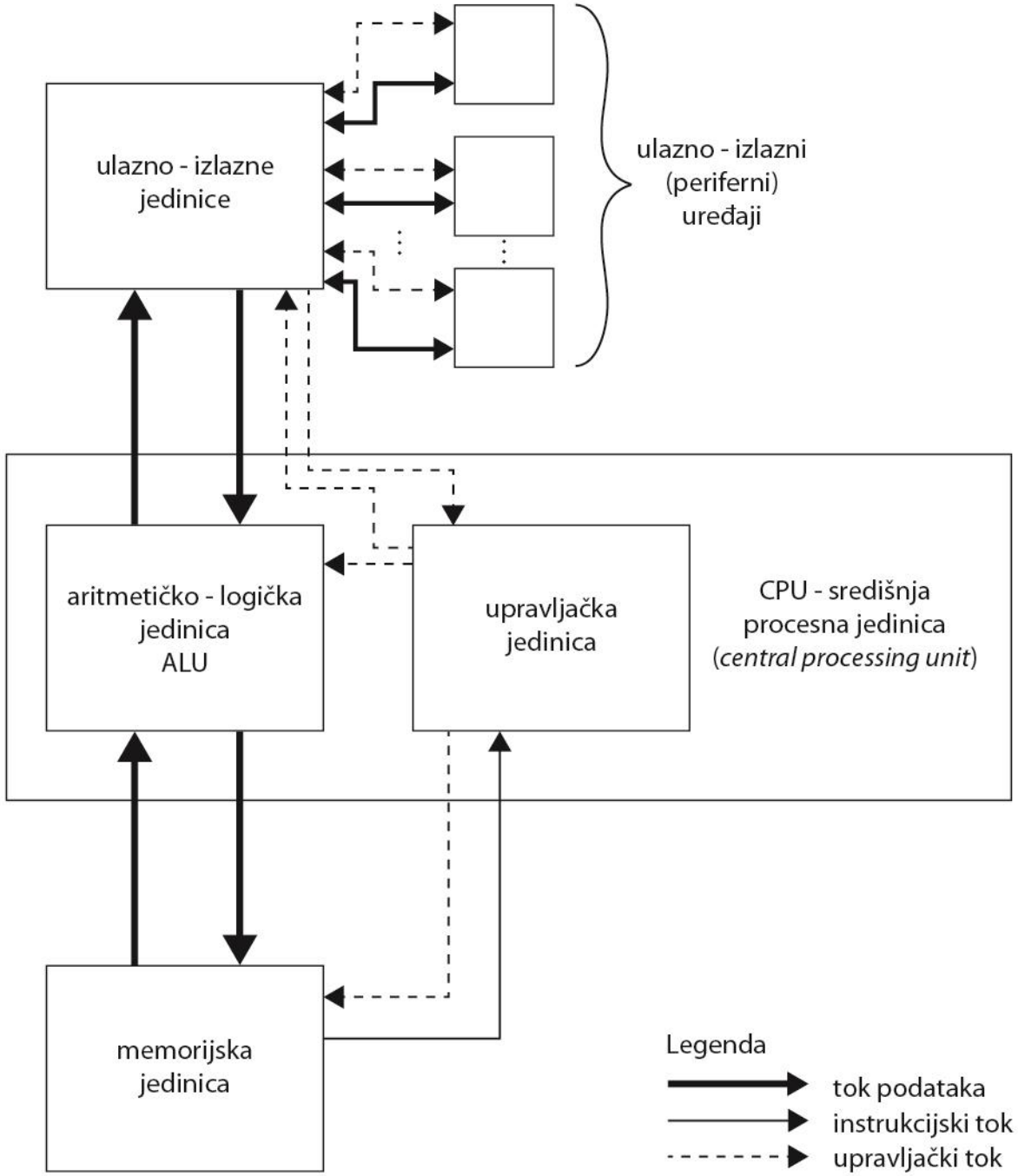
Flynnova klasifikacija – SISD



Flynnova klasifikacija - SISD

- SISD – Single Instruction Stream Single Data Stream
- Jednostruki instrukcijski tok, jednostruki tok podataka
- Arhitektura sekvencijalnih računala von Neumannova modela
- Standardna serijska računala s jednim procesorom

Von Neumannov model računala



Matična ploča

- ***Motherboard*** –osnovni je element računala
- Međusobno povezuje sve komponente osobnog računala
- Na matičnoj ploči se nalaze:
  - procesor, *chipset*, BIOS, memorija, sabirnice
  - utori sabirnica
  - svi konektori potrebni za komuniciranje s perifernim jedinicama
  - konektori za diskovne uređaje
  - konektori na stražnjoj strani ploče
  - matične ploče često uključuju mnoštvo integriranih podsustava (mreža ili modem, grafički ili zvučni podsustav)

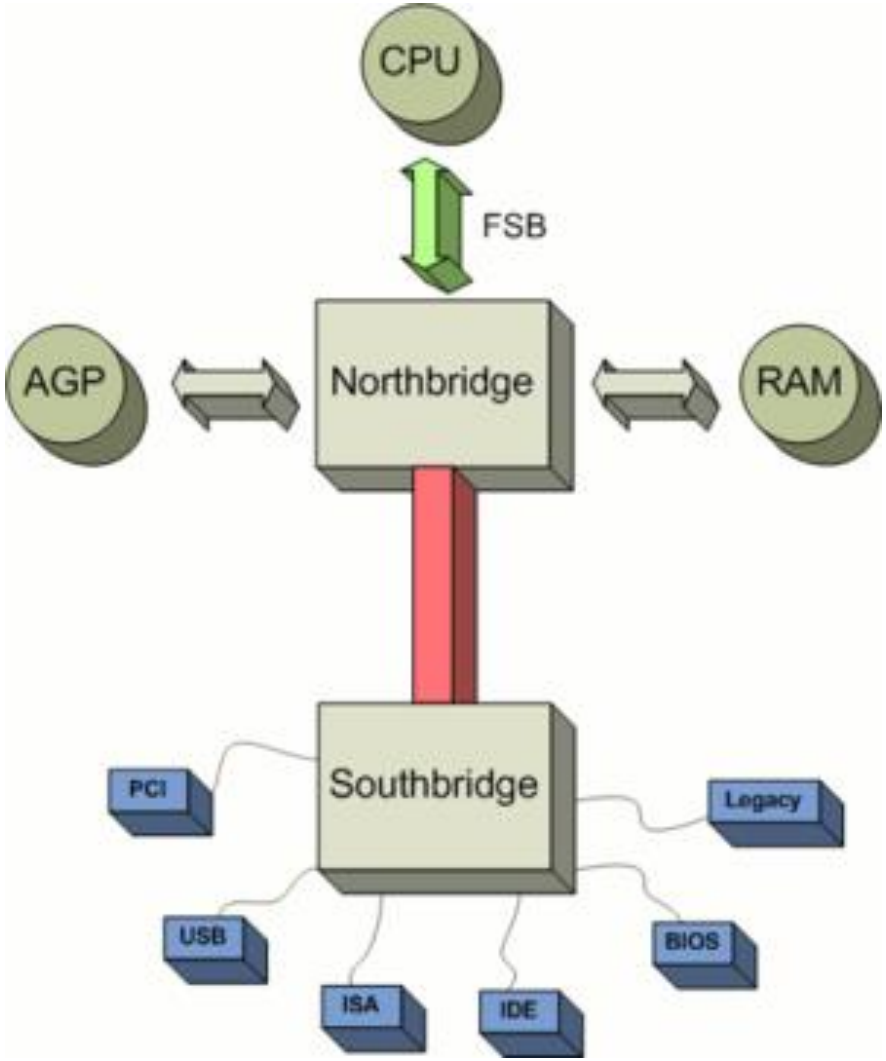
Matičnu ploču karakteriziraju parametri:

- vrste sabirnica
- arhitektura
- čipset (chipset)
- BIOS
- organizacija memorije

Matične ploče možemo podijeliti na:

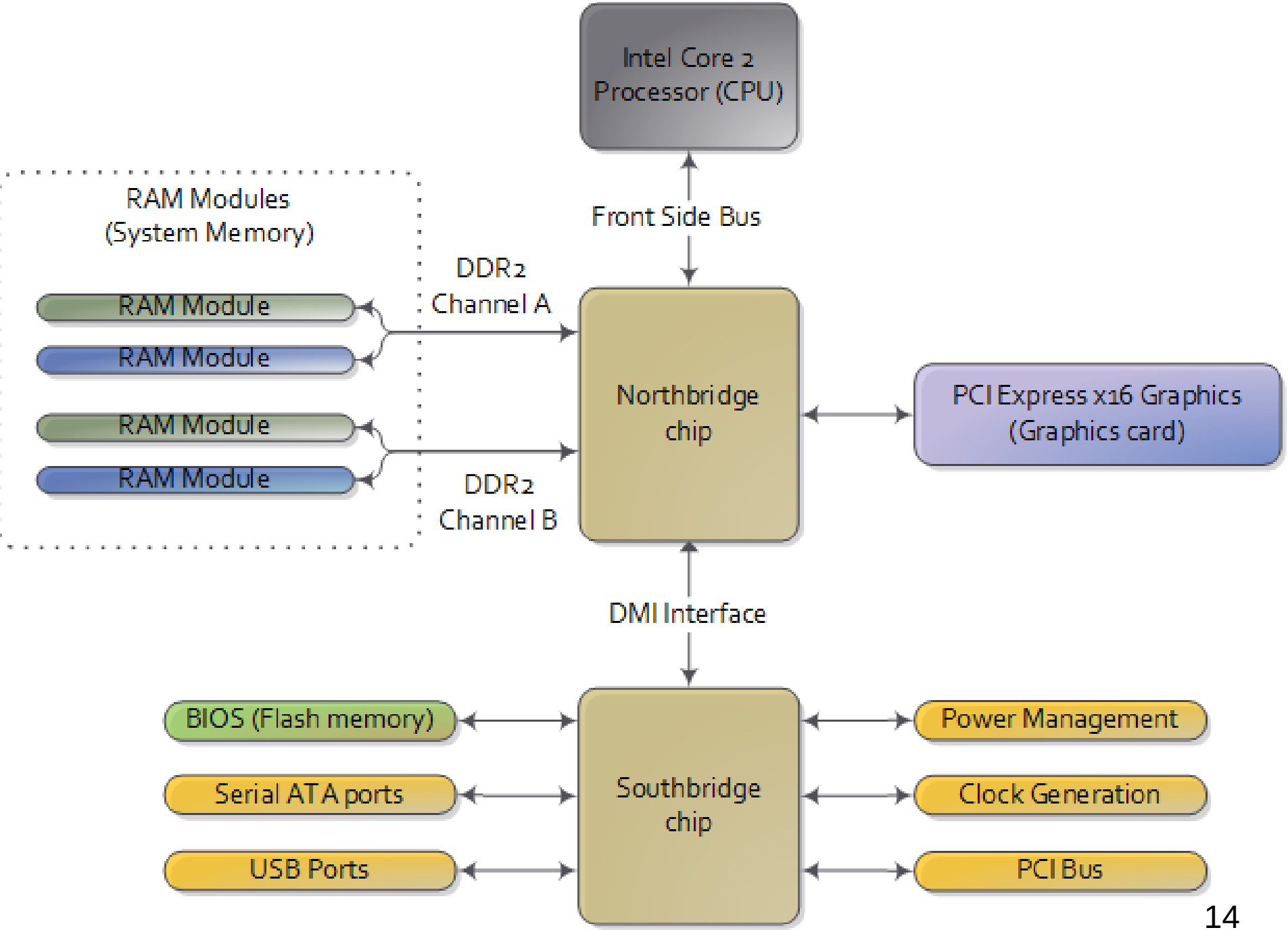
- standardne
- nestandardne
- ploče za specijalne namijene

Osnovna struktura matične ploče



Sinkrone i asinkrone matične ploče

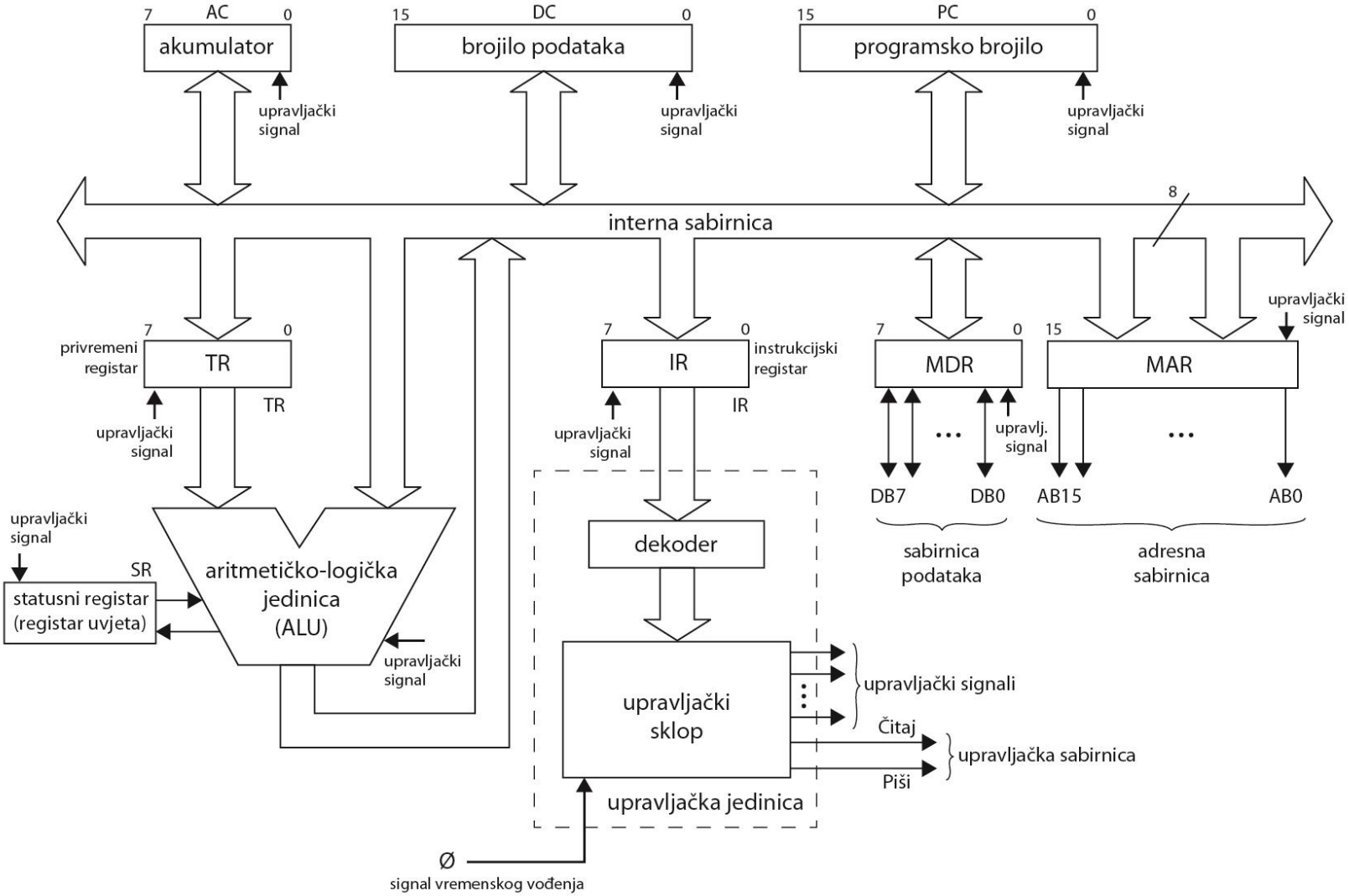
- Osnovna brzina na kojoj mikroprocesor komunicira s memorijom naziva se **memory bus speed**, odnosno **FSB** (Front Side Bus). Na većini matičnih ploča, odabirom FSB brzine, također je određena i brzina PCI sabirnice. Ove matične ploče, kod kojih brzina PCI sabirnice ovisi o odabranoj FSB brzini, nazivaju se **sinkronim matičnim pločama**.
- Neke matične ploče omogućavaju da se FSB i PCI taktovi postave neovisno. Ovakve matične ploče ne dopuštaju sinkrono povećanje takta PCI sabirnice ovisno o FSB taktu, pa se nazivaju **asinkrone matične ploče**.


Intel Core 2
Processor (CPU)
Front Side Bus
RAM Modules
(System Memory)
DDR2
Channel A
RAM Module
RAM Module
RAM Module
RAM Module
DDR2
Channel B
Northbridge
chip
PCI Express x16 Graphics
(Graphics card)
DMI Interface
BIOS (Flash memory)
Serial ATA ports
USB Ports
Southbridge
chip
Power Management
Clock Generation
PCI Bus

Ishod 2

2.	Imenovati osnovne module i sklopove procesora	Objasniti arhitekturu, module i sklopove procesora

Pojednostavljeni model 8-bitnog CISC procesora



Akumulator

- Akumulator **AC** je 8-bitni registar
- Koristi se za privremeno pohranjivanje jednog od operanda koji sudjeluju u aritmetičkoj ili logičkoj operaciji
- Rezultat operacije se najčešće pohranjuje ponovno u akumulator
- Akumulator sudjeluje i u prijenosu podataka između mikroprocesora i ostalih komponenti računala

Privremeni registar

- Privremeni registar **TR** je 8-bitni registar koji služi za privremeno pohranjivanje jednog od podataka koji sudjeluje u aritmetičko logičkoj operaciji
- Povezan je s jednim od ulaza u ALU
- Privremeni registar nije element programskog modela mikroprocesora

Programsko brojilo i brojilo podataka

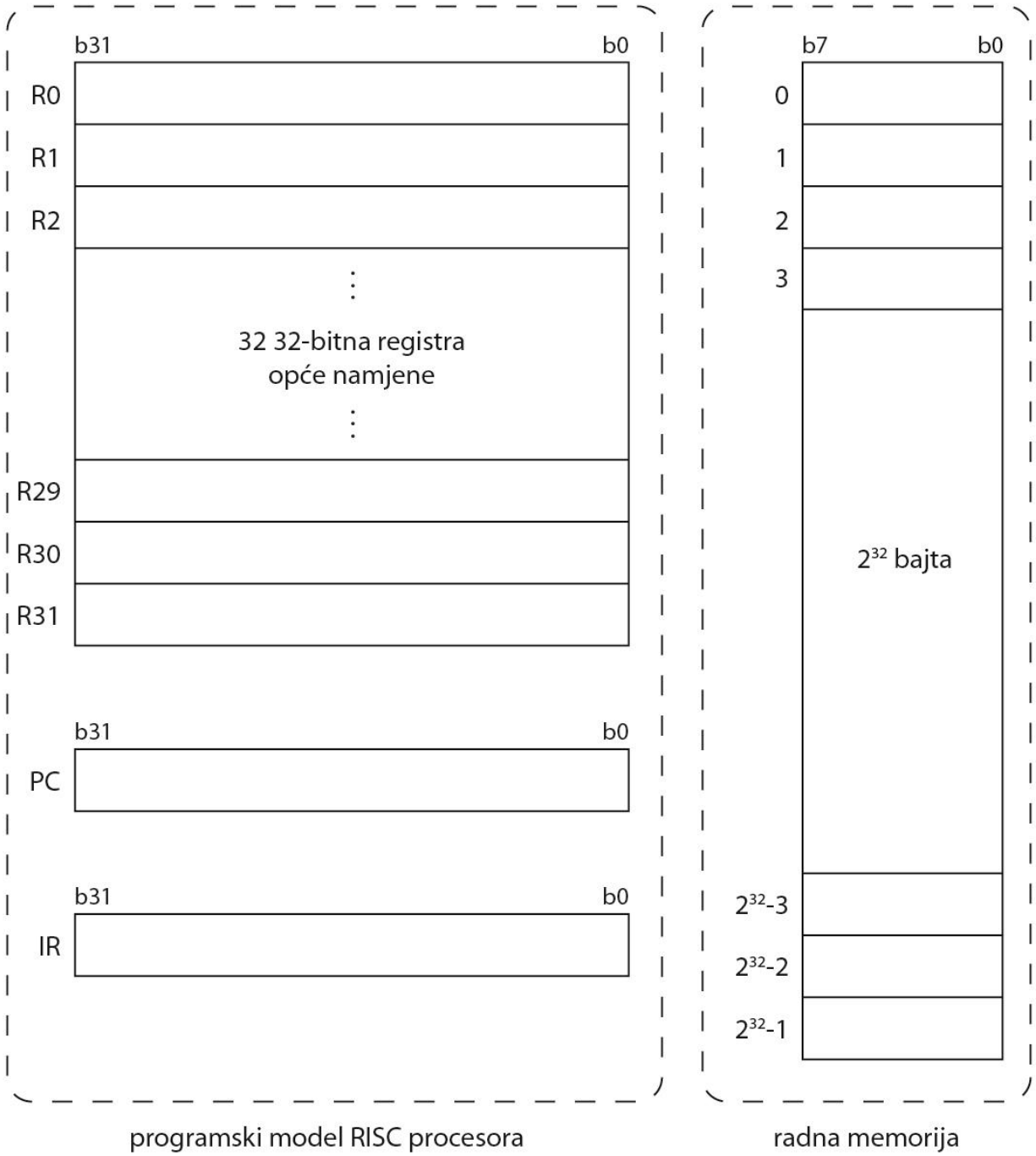
- Programsko brojilo **PC** (Program Counter) sadrži adresu sljedeće instrukcije (koja ce biti pribavljena u idućem ciklusu)
- PC je 16-bitni adresni registar

- Brojilo podataka **DC** (Data Counter) sadrži adresu memorijske lokacije na kojoj se nalazi operand
- DC je 16-bitni adresni registar

Instrukcijski registar

- U instrukcijskom registru (**IR**) pohranjen je operacijski kod instrukcije koja se upravo izvršava
- IR je 8-bitni registar
- Duljina instrukcijskog koda je jedan bajt

- Programski model RISC procesora



Model RISC procesora -opis

- Programski model procesora čine one njegove sastavnice koje su vidljive i dohvatljive programeru kada programira u zbirnom jeziku (asembleru).

Programski model

Programski model procesora sastoji se od:

- 32 32-bitna registra opće namjene (R0 – R31)
- 32-bitnog programskog brojila PC i
- 32-bitnog instrukcijskog registra IR

Desni dio slike predstavlja model memorijske jedinice (memorije) za računalo temeljeno na RISC procesoru

Neke bitne arhitektonske značajke

- procesor je ***registarsko orijentiran stroj*** – *ima 32 32-bitna registra opće namjene*
- (R0 – R31) i ima **tri** 32-bitne **interne** sabirnice:
  - sabirnicu operanda A,
  - sabirnicu operanda B i
  - sabirnicu rezultata C

Neke bitne arhitektonske značajke

- svi registri su duljine **32** bita:
  - registri opće namjene (R0 – R31)
  - instrukcijski registar IR
  - programsko brojilo PC te
  - sučelni registri – memorijski adresni registar MAR i memorijski registar podataka MDR –

Ishod 3

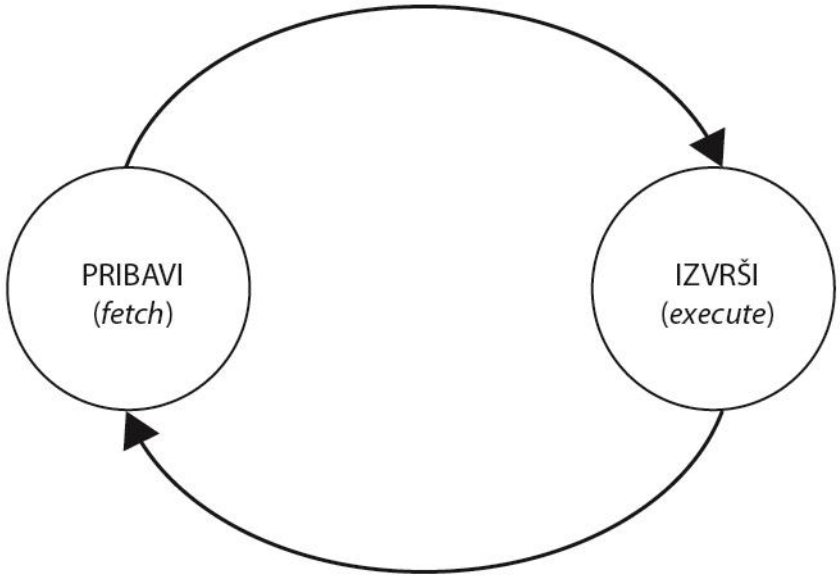
3.	Objasniti faze izvođenja intrukcija i stanje na sabirnicama	Prikazati na zadanom zadatku izvođenje instrukcija i stanje na sabirnicama

Izvršavanje instrukcija

- Instrukcije se izvršavaju u dvije faze:
  - **PRIBAVI (fetch)**
  - **IZVRŠI (execute)**
- Za vrijeme faze PRIBAVI procesor pribavlja **kod** instrukcije i **adresu** operanda

Izvođenje instrukcija

- Strojna se instrukcija u procesoru izvršava u dvije faze: PRIBAVI i IZVRŠI



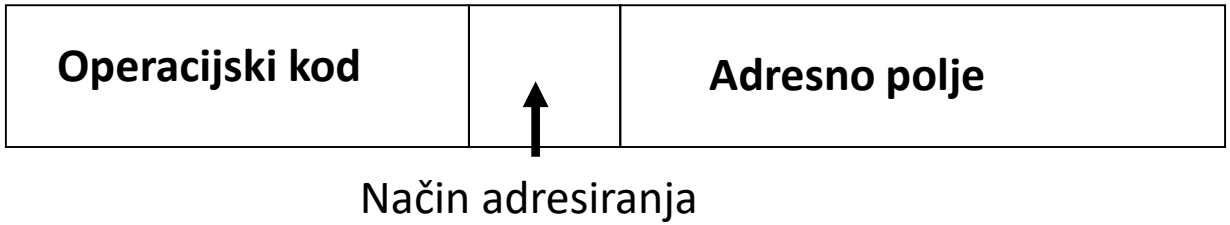
Procesor izvodi instrukcije na sljedeći način:

- ponavljati
  - dohvati iz memorije instrukciju na koju pokazuje PC
  - dekodirati instrukciju
  - PC+1
  - odrediti odakle dolaze operandi i kuda se pohranjuje rezultat
  - operande dovesti na ALU i izvesti zadanu operaciju
  - pohraniti rezultat
- do isključenja

Izvršavanje instrukcija

- Tijekom faze **IZVRŠI** upravljačka jedinica, u skladu s tumačenjem operacijskog koda, generira sljedove upravljačkih signala koji:
  - pobuđuju sklopove u aritmetičko-logičkoj jedinici
  - prijenos između registara
  - prijenos podataka između procesora i memorijske jedinice ili
  - procesora i ulazno-izlazne jedinice

Oblik instrukcijske riječi



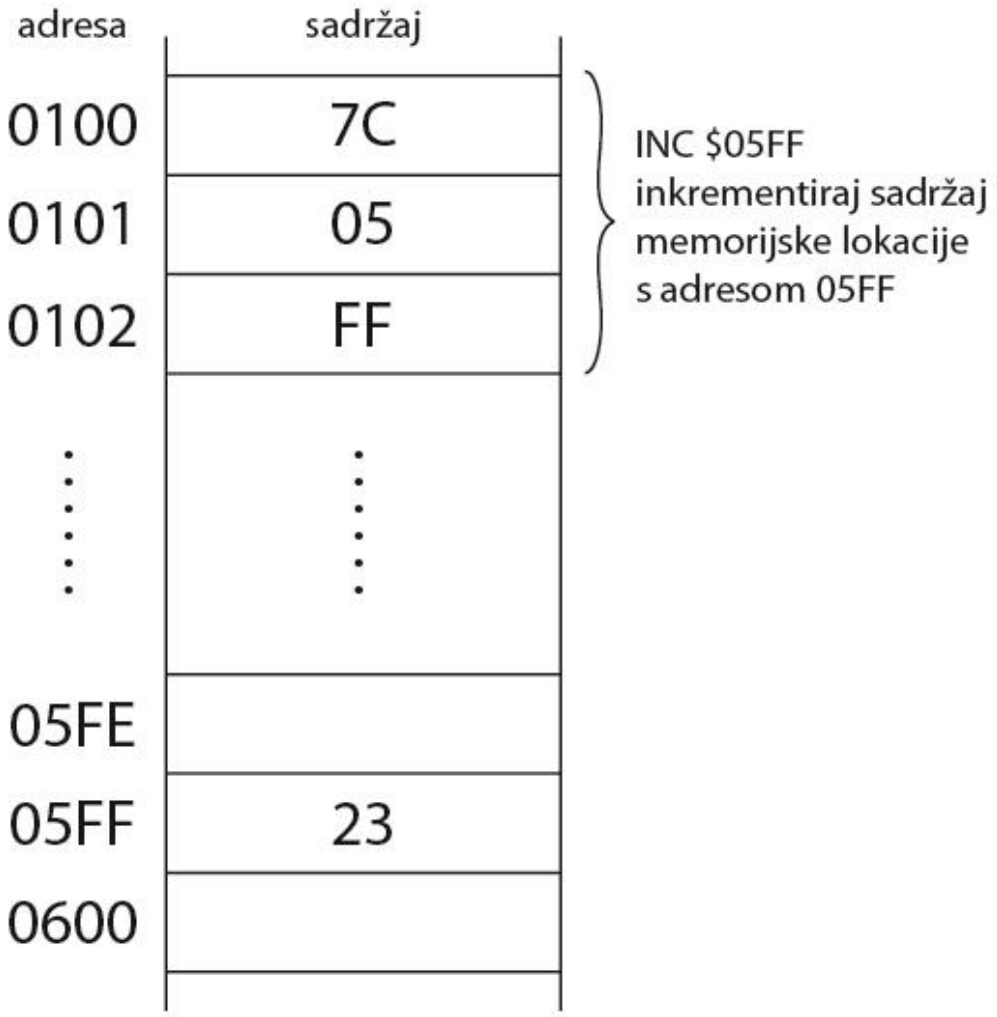
- **Operacijski kod** – određuje vrstu operacije
- Način adresiranja
- **Adresno polje** - određuje adresu operanda i rezultata

Primjer izvođenja programa

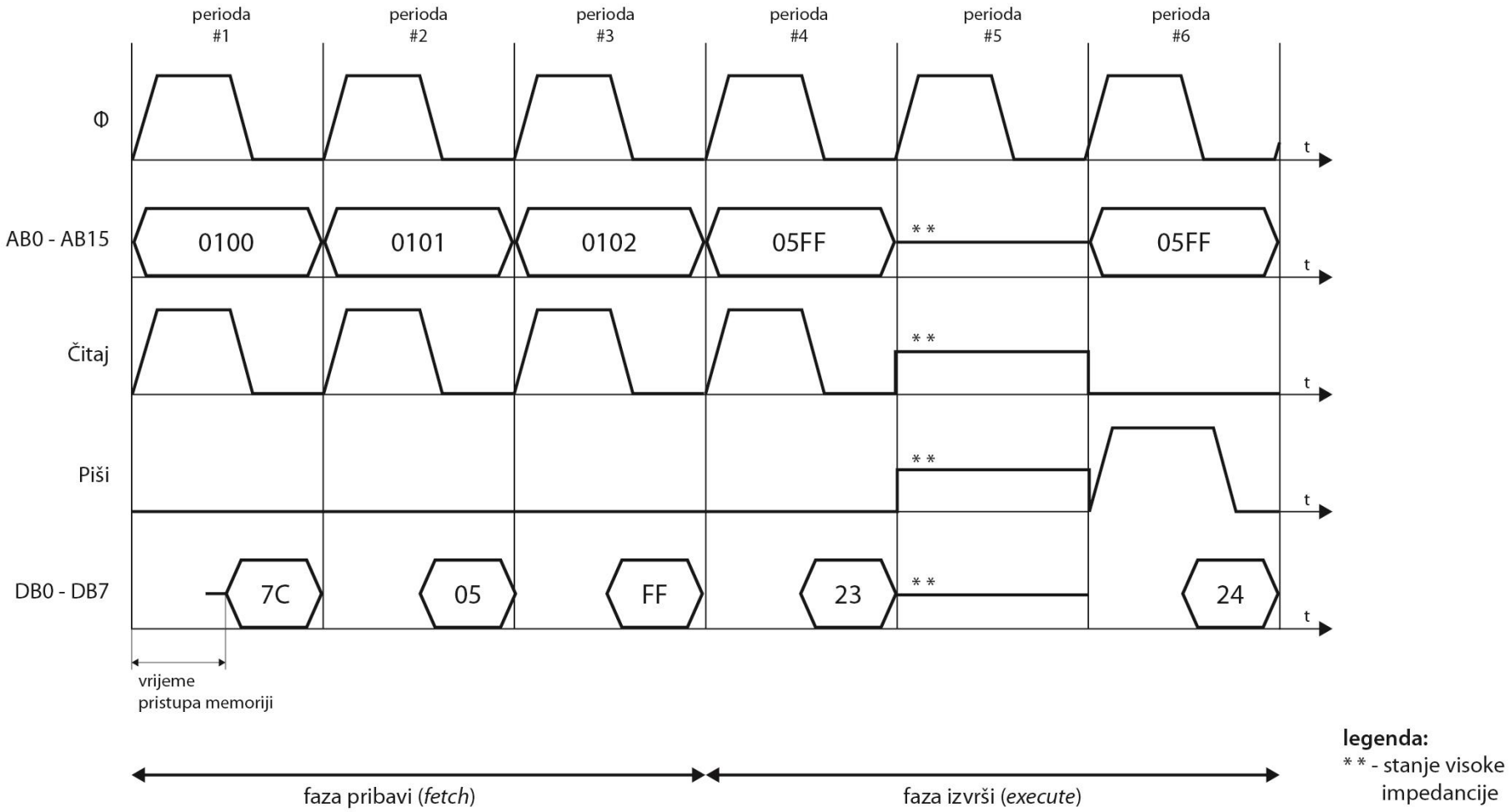
- Na primjeru izvođenja jednostavnog programa za naš model prikazat ćemo kako djeluje procesor
- Pretpostavimo da se program sastoji samo od jedne strojne instrukcije: ***INC $05FF***
- Instrukcija *INC $05FF, zapisana u skladu s pravilima za zbirni jezik, tumači se kao:*
  - *inkrementiraj* sadržaj memorijske lokacije čija je adresa 05FF (heksadekadno)
- Operacijski kod instrukcije *INC* je *7C (heksadekadno)*

Prikaz programa i podataka u memorijskoj jedinici računala

- U memoriji računala, na slijednim memorijskim lokacijama pohranjen je program



Stanje na sabirnicama



Stanje registara procesora

- programsko brojilo: (PC) = 0103,
- instrukcijski registar: (IR) = 7C,
- brojilo podataka: (DC) = 05FF,
- privremeni registar: (TR) = 24

Promijenjeni sadržaj memorijske lokacije 05FF je:
M(05FF) = 24

Zadatak

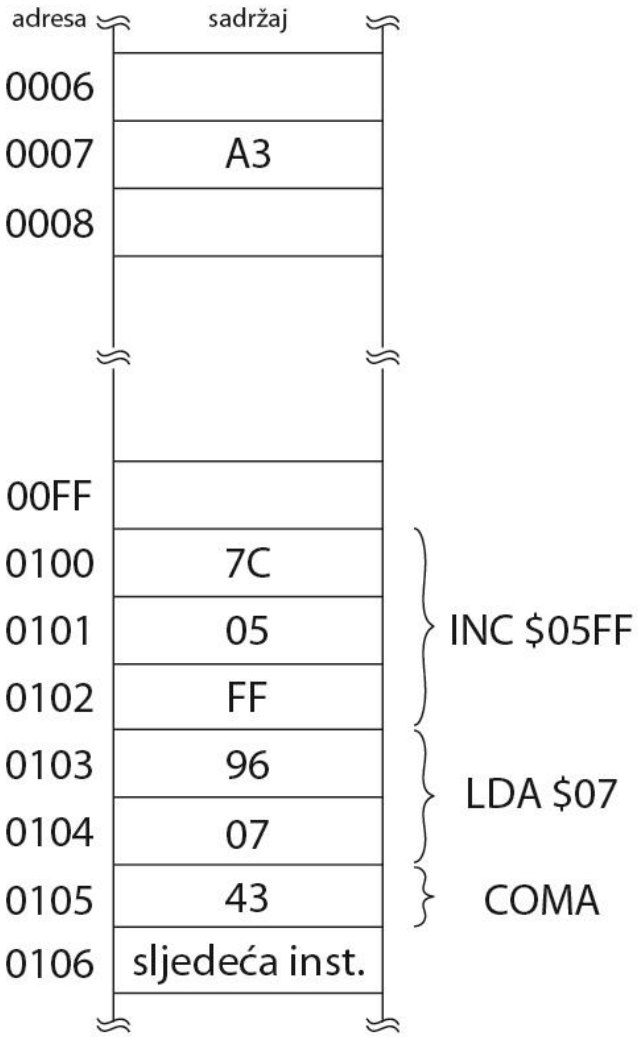
- Napisati stanje registara nakon izvođenja programa INC $3F05 u pojednostavljenom modelu procesora i napisati prikaz programa i podataka u memorijskoj jedinici računala prije i nakon izvođenja programa. Na adresi $3F05 je operand C5, a operacijski kod instrukcije *INC* je *7C.* Program je smješten u memoriji na početnoj adresi 000E

Zadatak

- Nacrtati stanje na sabirnicama za izvođenje programa INC $2CA3 i napisati prikaz programa i podataka u memorijskoj jedinici računala. Na adresi $2CA3 je operand 3F, a operacijski kod instrukcije *INC* je *7C.* Program je smješten u memoriji na početnoj adresi 0000

Primjer izvođenja programa (2)

- INC $05FF
- LDA $07
- COMA



Primjer izvođenja programa (2)

- Ako pretpostavimo da je sadržaj memorijske lokacije 0007 A3:

M(0007) = A3

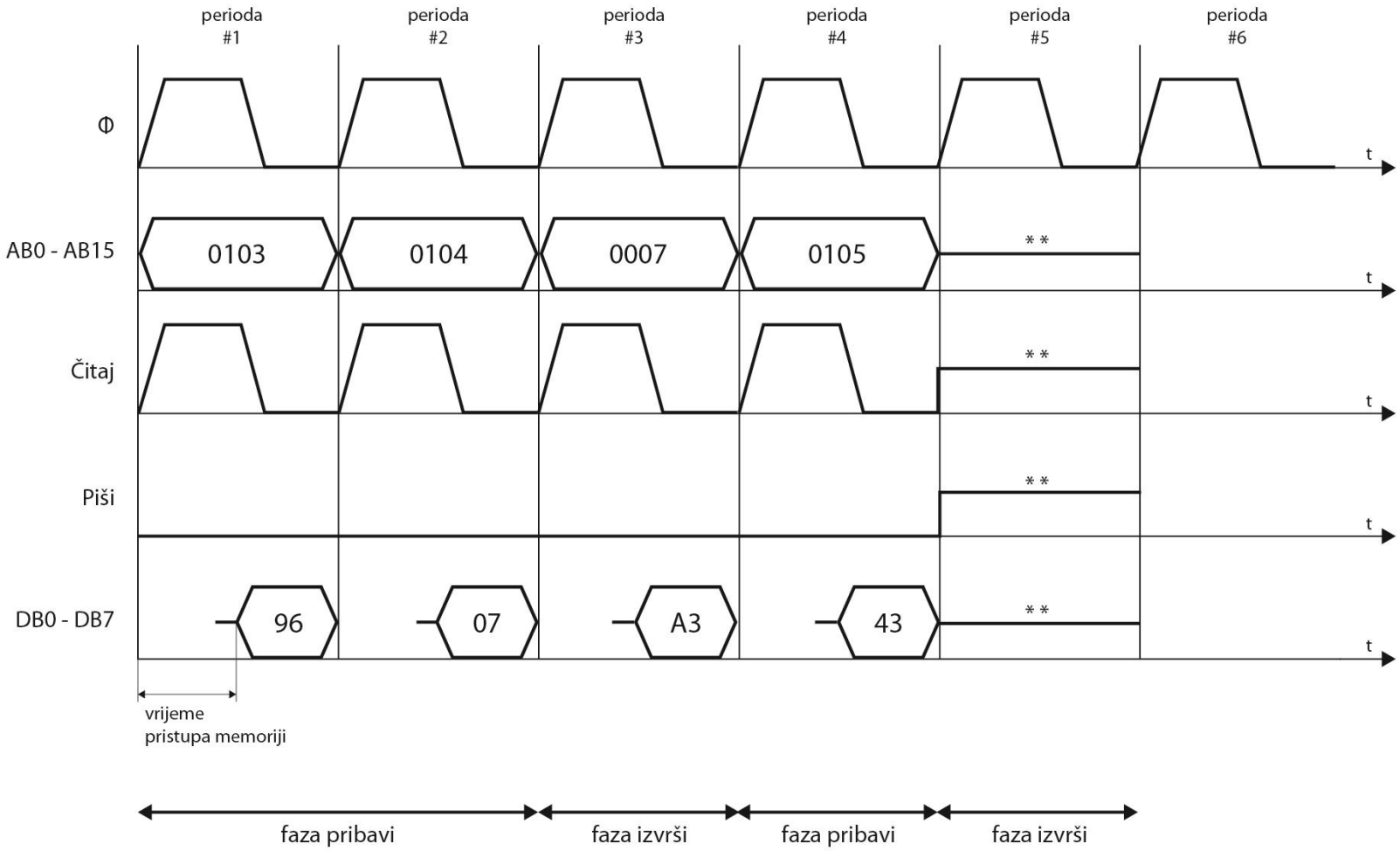
- tada posljedica izvođenja programskog odsječka:

LDA $07

COMA

- je (AC) = **5C**;
- u akumulatoru AC je pohranjen jedinični komplement operanda A3

Stanje na sabirnicama



legenda:
* * - stanje visoke impedancije

Primjer programa

- Napisati stanje registara nakon izvođenja programa LDA $5A2B u pojednostavljenom modelu procesora i napisati prikaz programa i podataka u memorijskoj jedinici računala prije i nakon izvođenja programa. Na adresi $5A2B je operand 3C, a operacijski kod instrukcije *LDA* je *69.* Program je smješten u memoriji na početnoj adresi 0010.

Ishod 4

4.	Primijeniti aritmetičke i logičke instrukcije u rješavanju asemblerskih zadataka.	Rješavati složenije asemblerske zadatke primjenom instrukcijskog seta 8051

INSTRUKCIJE ZA IZAZIVANJE UVJETNIH SKOKOVA

- **JZ rel** ;skoči ako je A=0
- **JNZ rel**
- **JC rel** ;skoči ako je C=1
- **JNC rel**

- JZ – *jump if zero*
- JNZ – *jump if not zero*
- JC – *jump if carry* – skoči ako postoji prijenos (C=1)
- JNC – jump if not carry – skoči ako ne postoji prijenos (C=0)

Instrukcije za usporedbu i skok

- Koristi se za olakšavanje programiranja usporedbi dviju vrijednosti u programu.

CJNE A,direct,rel ;usporedi i skoči ako nisu =

CJNE A, #data,rel

CJNE Rn, #data,rel

CJNE @Ri, #data,rel

Definirajte varijablu Test i dodijelite joj vrijednost 44h. U akumulator upišite vrijednost 11h. Izvršite AND operaciju nad akumulatorom i varijablom Test. Rezultat pohranite na indirektno adresiranu lokaciju DFh

```
org 0000h
test equ 44h
mov a,#11h
anl a,#test
mov r0,#0DFh
mov @r0,a
end
```

U akumulator pohraniti konstantu 20h, a u memorijsku lokaciju 40h konstantu 3. Podijeliti ove konstante te rezultat pohraniti u registar R3, a ostatak dijeljenja na lokaciju 2Ah

- org 0000h
- mov a,#20h
- mov 40h,#03h
- mov b,40h
- div ab
- mov R3,a
- mov 2Ah,b
- end

Napišite program koji će indirektnim adresiranjem na memorijske lokacije A1 i A2 upisati 2 i F. Program će brojeve zbrojiti te rezultat spremiti na lokaciju BB. Lokacija BB nije indirektno adresirana. Gdje je moguće, koristite registre po želji.

```
CLR A
MOV R0,#A1
MOV R1,#A2
MOV @R0,#2H
MOV @R1,#0FH
ADD A,#2h
ADD A,#0Fh
MOV 0BBh,a
END
```

Koristeći akumulator, lokaciju 1Fh i lokaciju 20h zbrojiti brojeve od 1 do 15. Rezultat pohraniti na lokaciji 21h.

```
clr a
mov 1Fh,#01h
mov 20h,#0Fh

zbrajanje:
add a,1Fh
inc 1Fh
djnz 20h,zbrajanje

mov 21h, a
```

Napišite program koji zbraja brojeve od A do 10. Koristite registre i petlju prema želji. Konačan rezultat pohranite u registar R4. Napišite asemblerski kod i tablično prikažite sadržaj registara za svaku iteraciju petlje.

```
CLR A
MOV R0,#0Ah
PETLJA:
ADD A,R0
INC R0
CJNE R0,#11h, PETLJA
MOV R4,A
END
```

Tablica:

A	R0
0	A
A	B
15	C
21	D
2E	E
3C	F
4B	10

Napišite program koji zbraja brojeve od 10 do 16. Napišite asemblerski kod i tablično prikažite sadržaj registara za svaku iteraciju petlje. Koristite registre i instrukciju petlje po želji.

- CLR A
- MOV R0,#10H
- PETLJA:
- ADD A,R0
- INC R0
- CJNE R0,#17H, PETLJA
- END

Tablica:

A	R0
0	10
10	11
21	12
33	13
46	14
5A	15
6F	16

Napišite program koji zbraja brojeve od A do 10. Koristite registre i petlju prema želji. Konačan rezultat pohranite u registar R4. Napišite asemblerski kod i tablično prikažite sadržaj registara za svaku iteraciju petlje.

- CLR A
- MOV R0,#0Ah
- PETLJA:
- ADD A,R0
- INC R0
- CJNE R0,#11h, PETLJA
- MOV R4,A
- END

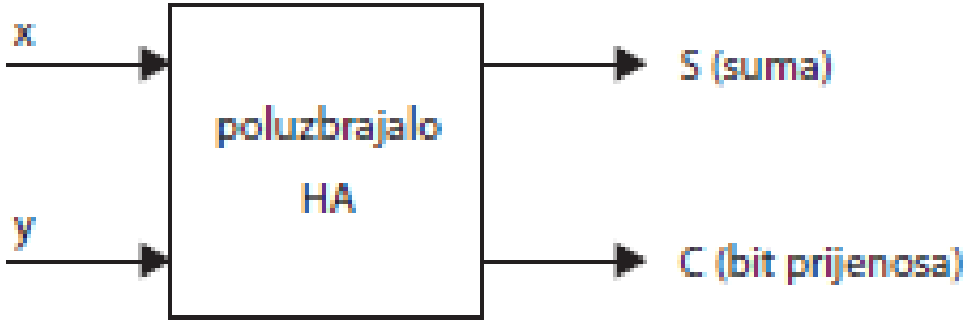
Tablica:

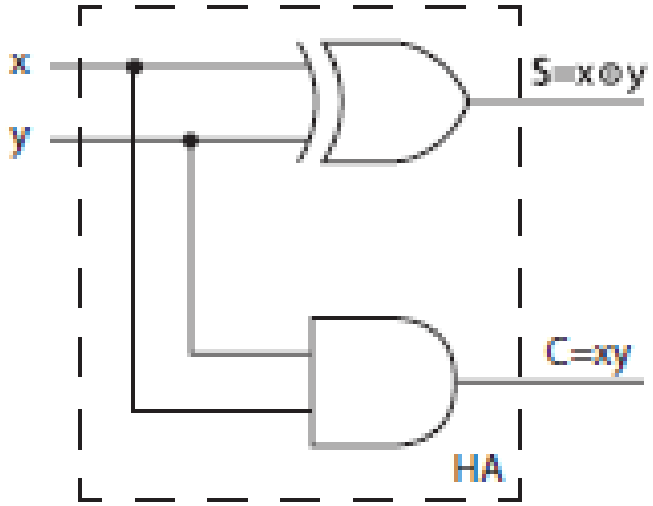
A	R0
0	A
A	B
15	C
21	D
2E	E
3C	F
4B	10

Ishod 5

5.	Opisati strukturu i osnovne elemente ALU i upravljačke jedinice	Objasniti princip rada mikroprogramirane upravljačke jedinice i ALU. Rješavati složenije asemblerske zadatke sa AL instrukcijama

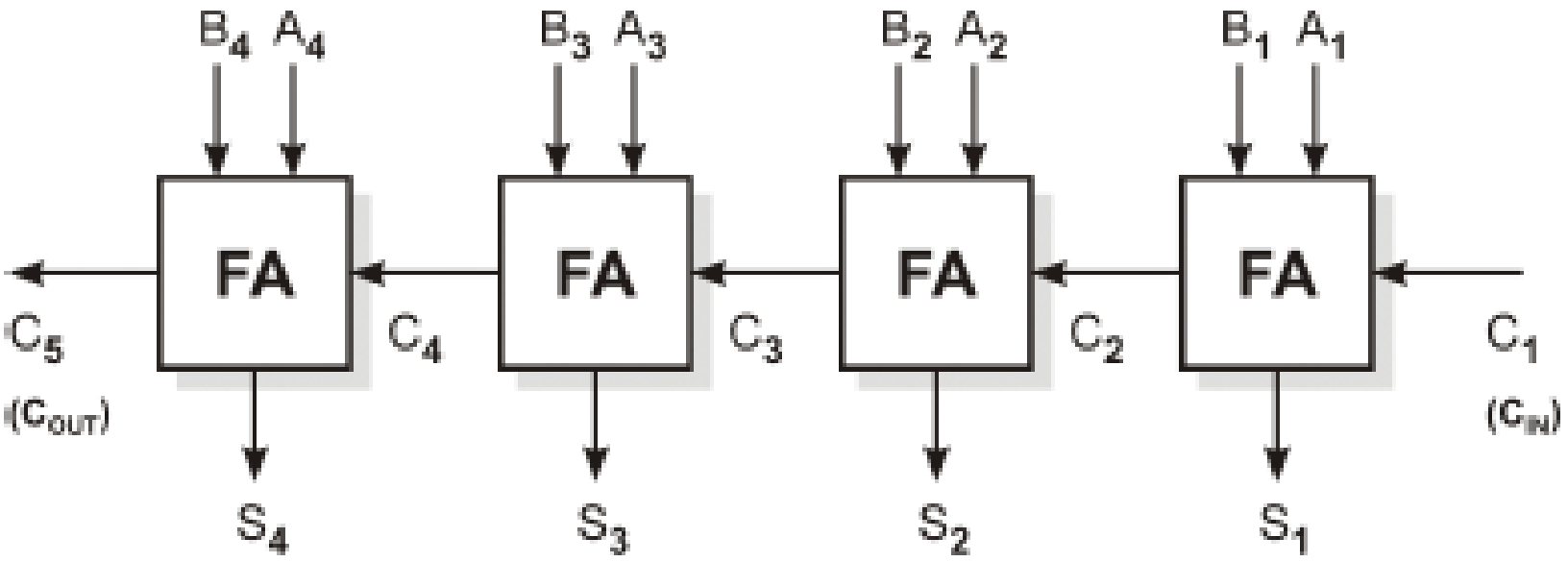
Poluzbrajalo HA





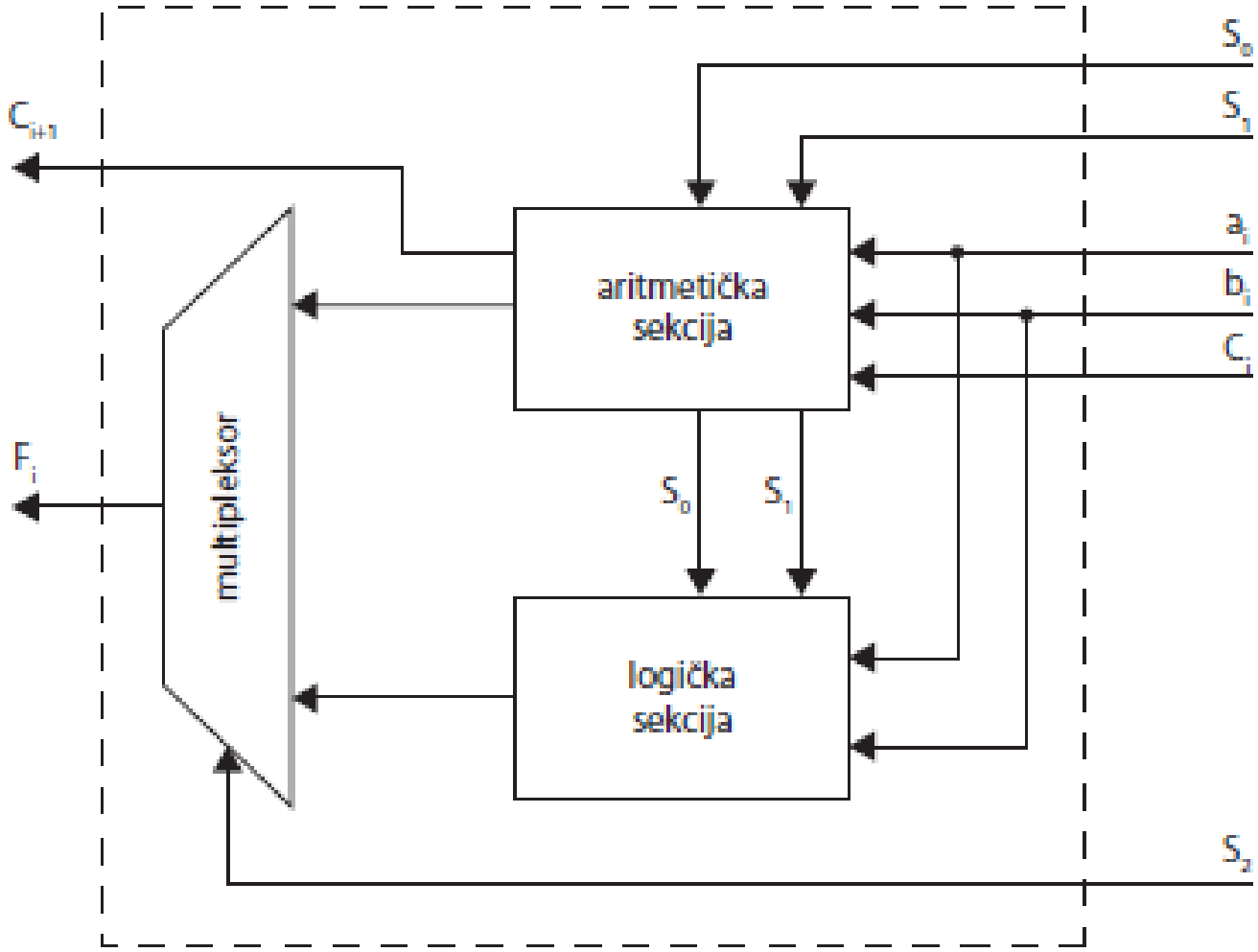
c)

Paralelno zbrajalo

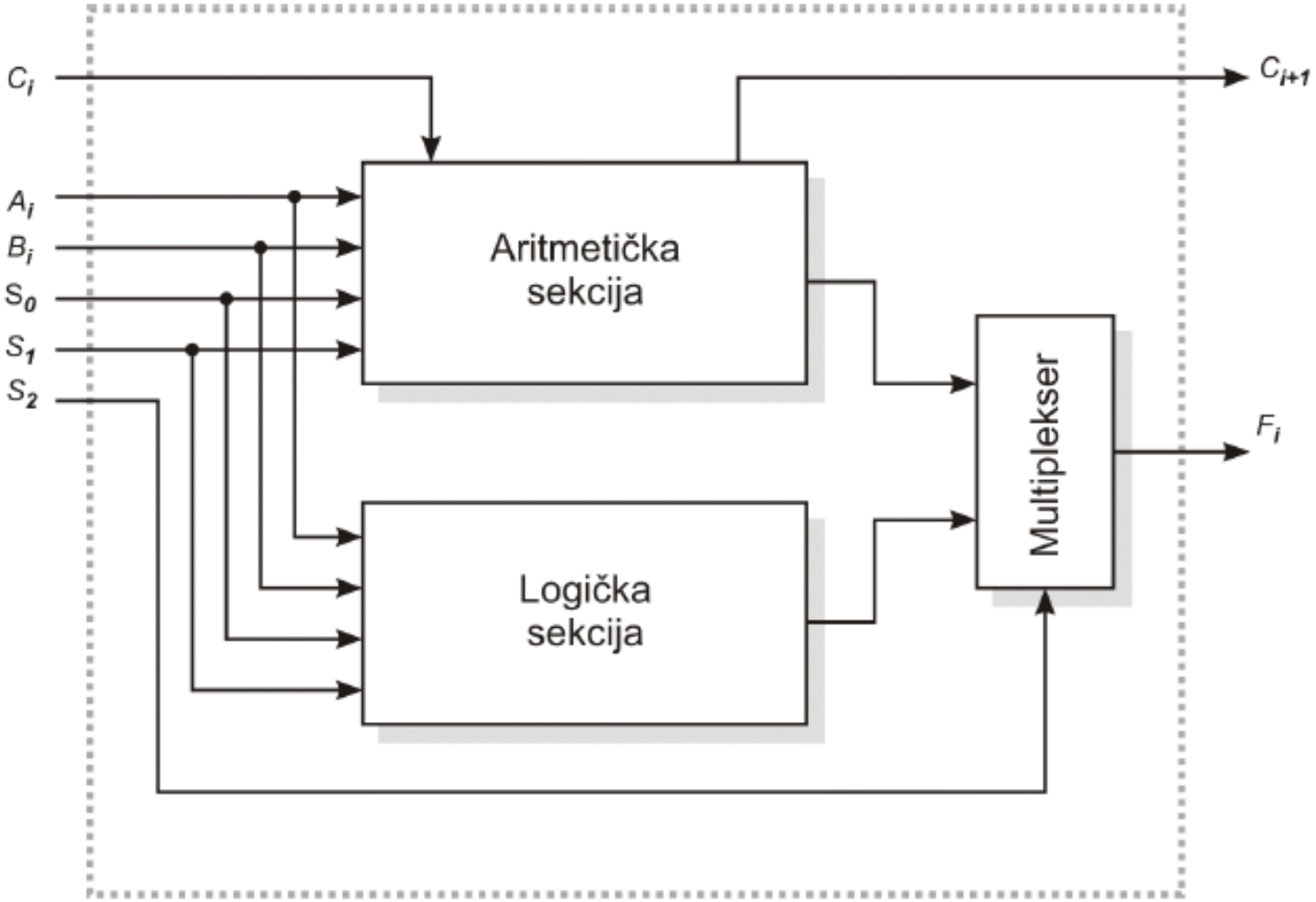


- izraz “paralelno zbrajalo” jer se podrazumijeva da će se *n zbrajanja* izvesti gotovo istodobno

i-ti stupanj ALU

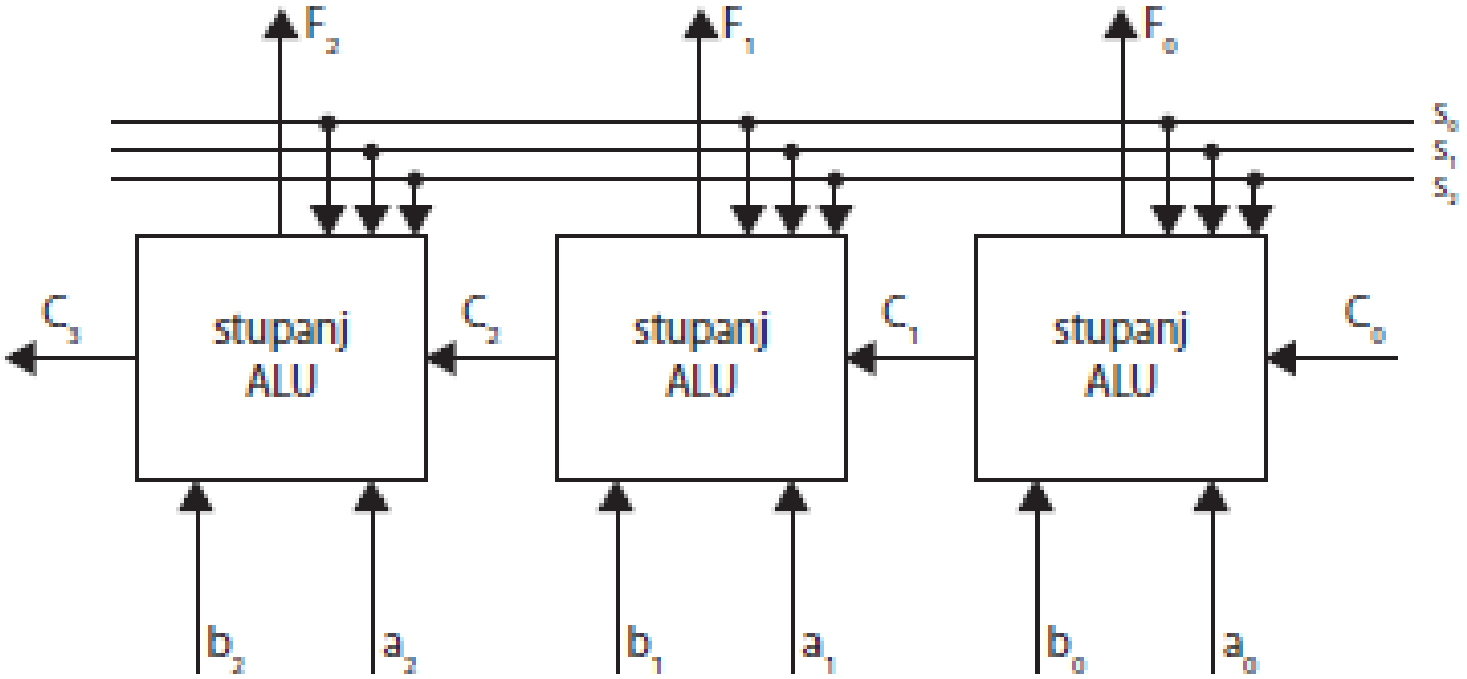


i-ti stupanj ALU

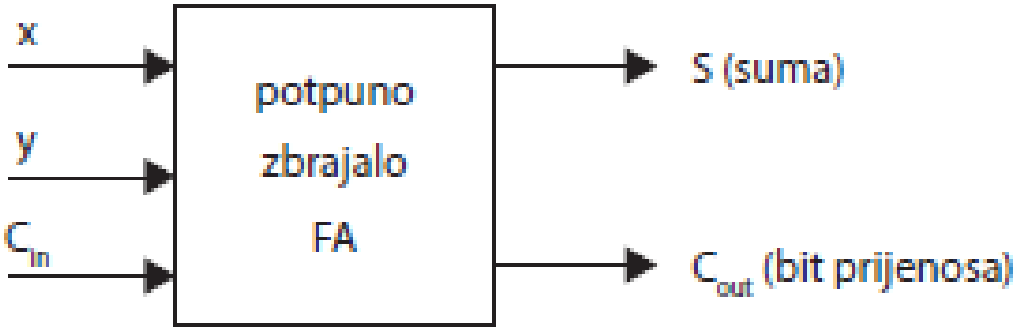


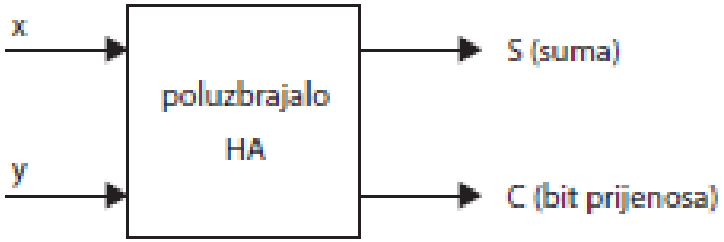
Povezivanje stupnjeva u kaskadu

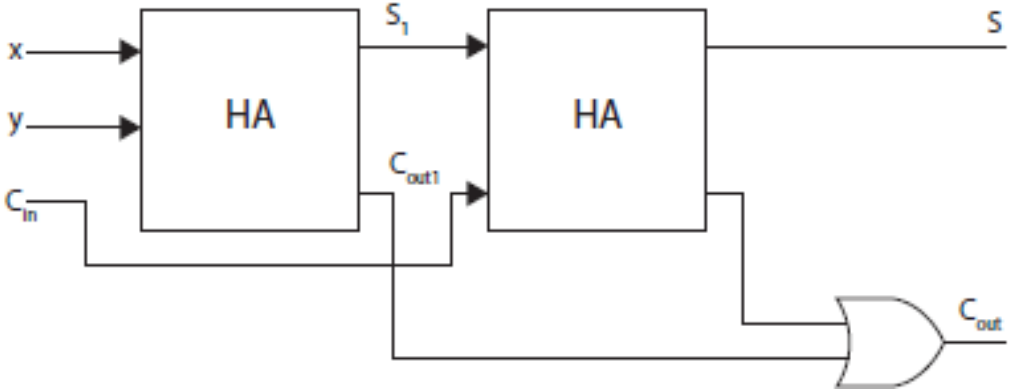
- Povezivanje stupnjeva u kaskadu ostvaruje se pomoću linija bita prijenosa Ci i Ci+1



Potpuno zbrajalo FA
(engl. full adder)







Upravljačka jedinica - pojam

- Generira upravljačke signale
- Koordinira sve aktivnosti unutar mikroprocesora
- Sinkronizira prijenos podataka i komunikaciju modula
- Pribavlja, dekodira i omogućuje izvođenje instrukcija
- Komunicira s ostalim komponentama mikroračunala preko sabirnica (adresne, podatkovne, upr.)
- Upravlja odgovorima na vanjske signale (zahtjev za prekid, zaustavljanje itd)

Upravljačka jedinica _ četiri temeljne funkcije:

1. uspostavljanje određenog stanja tijekom svakog instrukcijskog ciklusa
2. određivanje sljedećeg stanja na temelju trenutnog stanja, stanja zastavica u statusnom registru i stanja na ulaznim upravljačkim linijama procesora
3. pohranjivanje informacije koja opisuje tekuće stanje u kojem se procesor nalazi
4. generiranje upravljačkih signala za izmjenu podataka između procesora i drugih funkcijskih jedinica

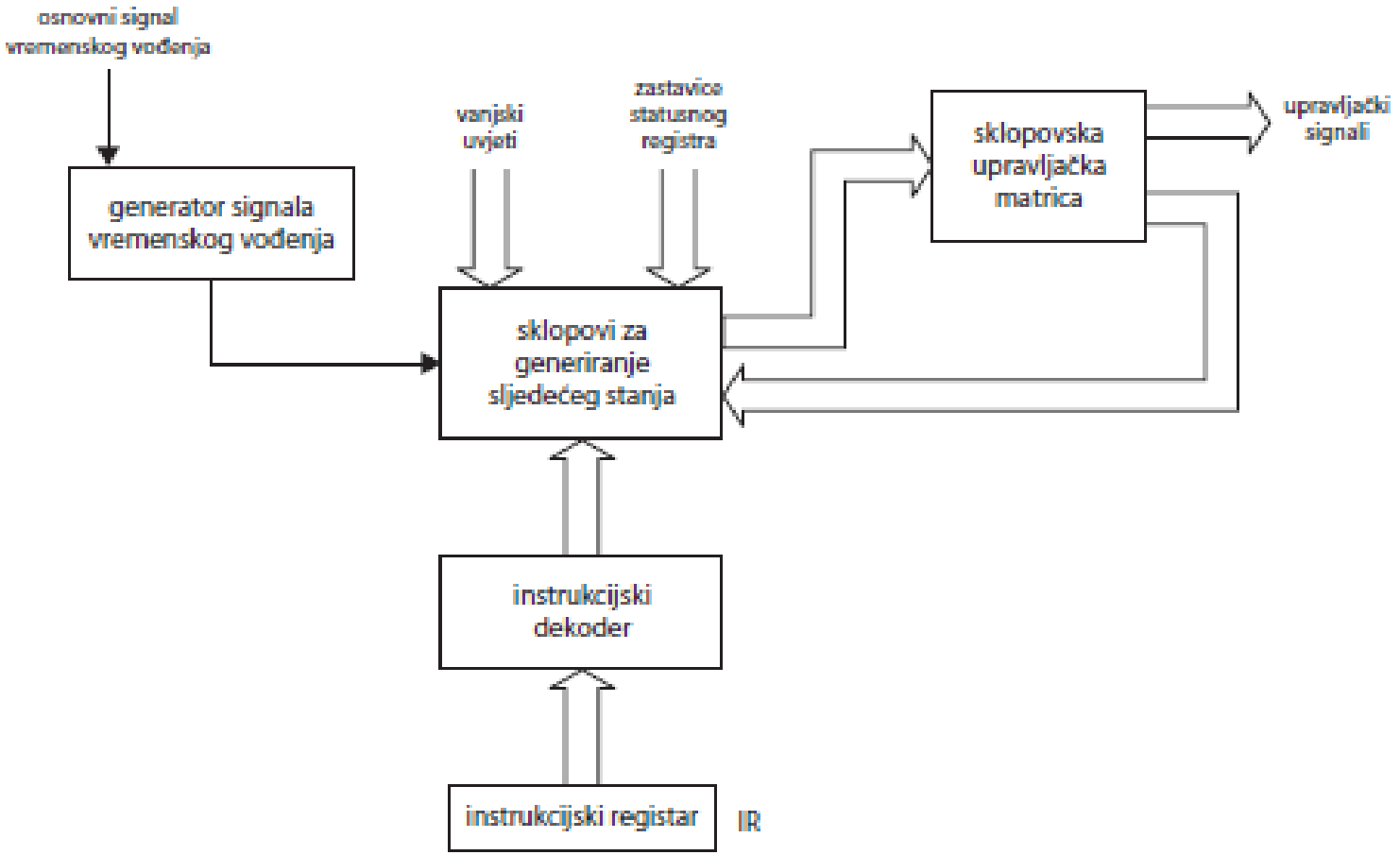
Upravljačka jedinica

- Upravljačka jedinica treba generiranim slijedom upravljačkih signala podržati:
  - Pribavljanje instrukcije
  - Dekodiranje instrukcije
  - Izvršavanje instrukcije te
  - Prijenos upravljanja s instrukcije na instrukciju u programu (engl. *instruction sequencing)*

Upravljačka jedinica –dekodiranje koda

- Instrukcijski dekoder tumači operacijski kod instrukcije i pobuđuje odgovarajuću izlaznu liniju
- *Na temelju:*
  - *pobuđene* linije
  - vanjskih uvjeta i/ili
  - stanja zastavica u statusnom registru te
  - na temelju povratnih signala iz sklopovske upravljačke matrice
- sklopovi za generiranje sljedećeg stanja pobuđuju signale koji *odgovaraju vremenskom slijedu izvođenja elementarnih operacija koje se* nazivaju **mikrooperacije**

Sklopovska upravljačka jedinica

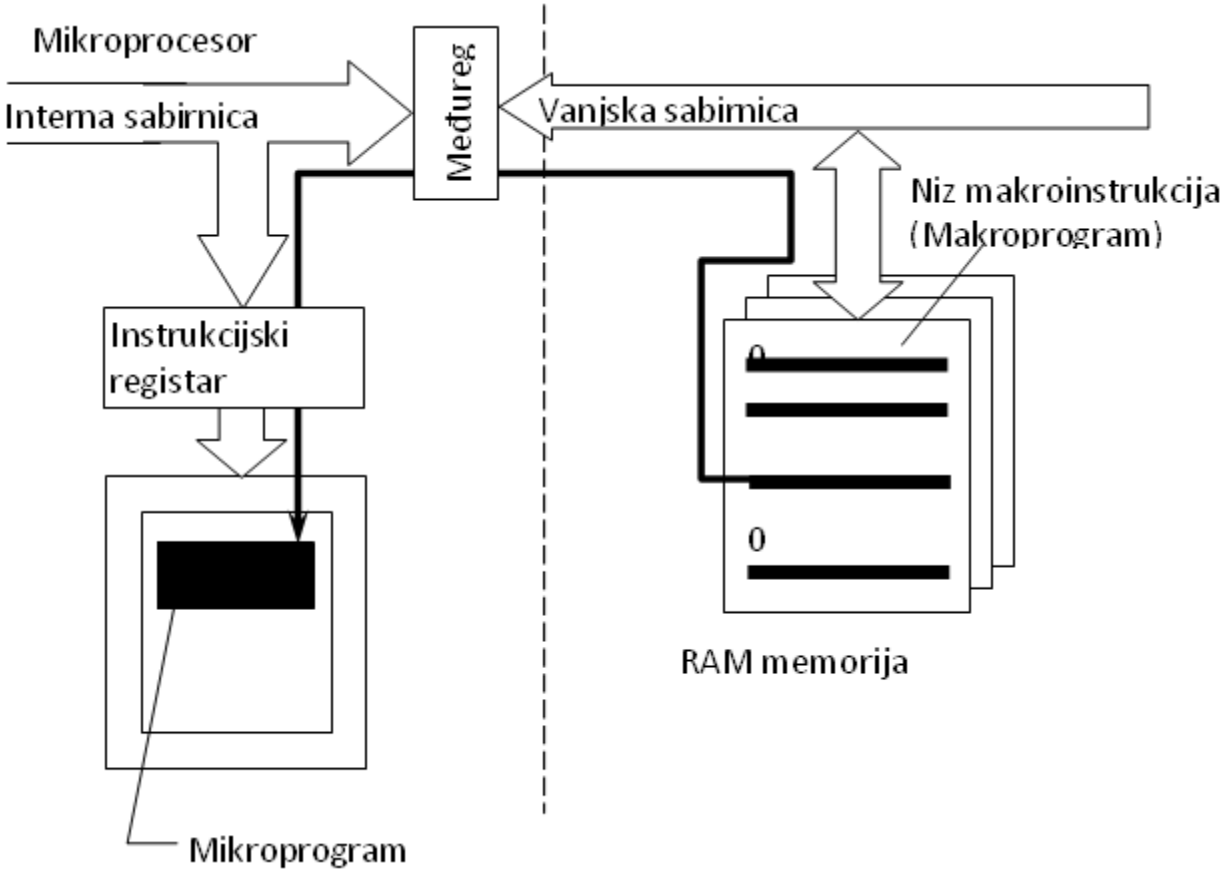


- Organizacija upravljačke jedinice ostvarena na temelju standardnog pristupa oblikovanja sekvencijalnih sklopova

Mikroprogram i makroprogram

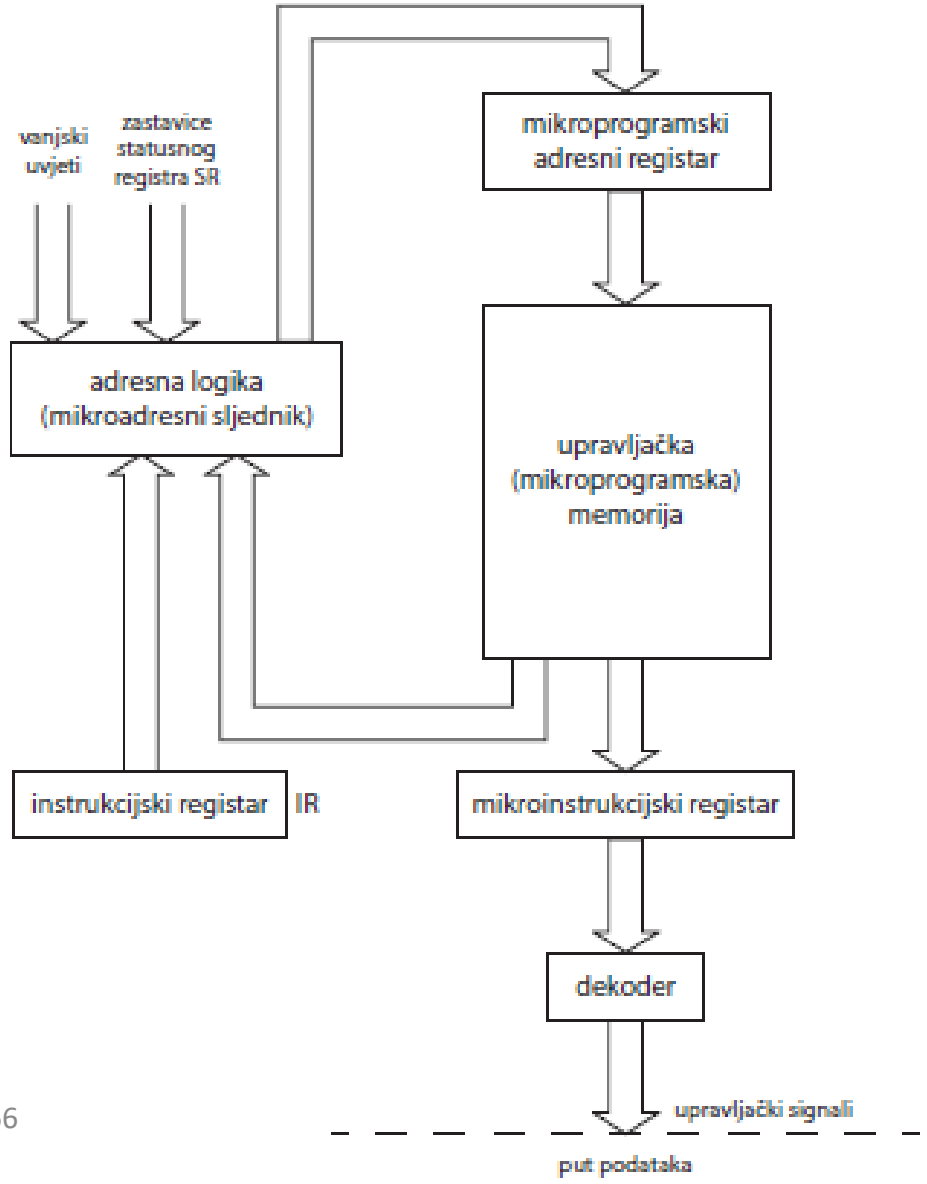
- Element iz niza upravljačkih signala naziva se ***mikroinstrukcija***
- Niz mikroinstrukcija čini ***mikroprogram***
- Mikroinstrukcija je uzrok jedne ili više osnovnih operacija na razini prijenosa podataka ili aktiviranja pojedinih sklopova
- Takva osnovna operacija naziva se ***mikrooperacija***
- Instrukcija pribavljena iz memorije (***makroinstrukcija***) uvjetuje izvođenje čitavog niza ***mikroinstrukcija***
- Niz makroinstrukcija čini ***makroprogram*** koji je smješten u RAM-u.

Makroprogram - mikroprogram



Mikroprogramski pristup

- *Mikroprogramiranje - alternativni pristup oblikovanju upravljačke jedinice procesora*



Koristeći proizvoljne registre, napišite program koji zbraja brojeve korištenjem **CJNE** instrukcije od **09h** do **10h**, te rezultat pohranjuje u registar **R1**. Nacrtajte tablicu s prikazom stanja korištenih registara u heksadekadskom formatu za svaku iteraciju **petlje**!

```
Org 0000h
Clr A
Mov R0,#09h
Petlja:
Add A,R0
Inc R0
CJNE R0,#11h, Petlja
Mov R1,A
end
```

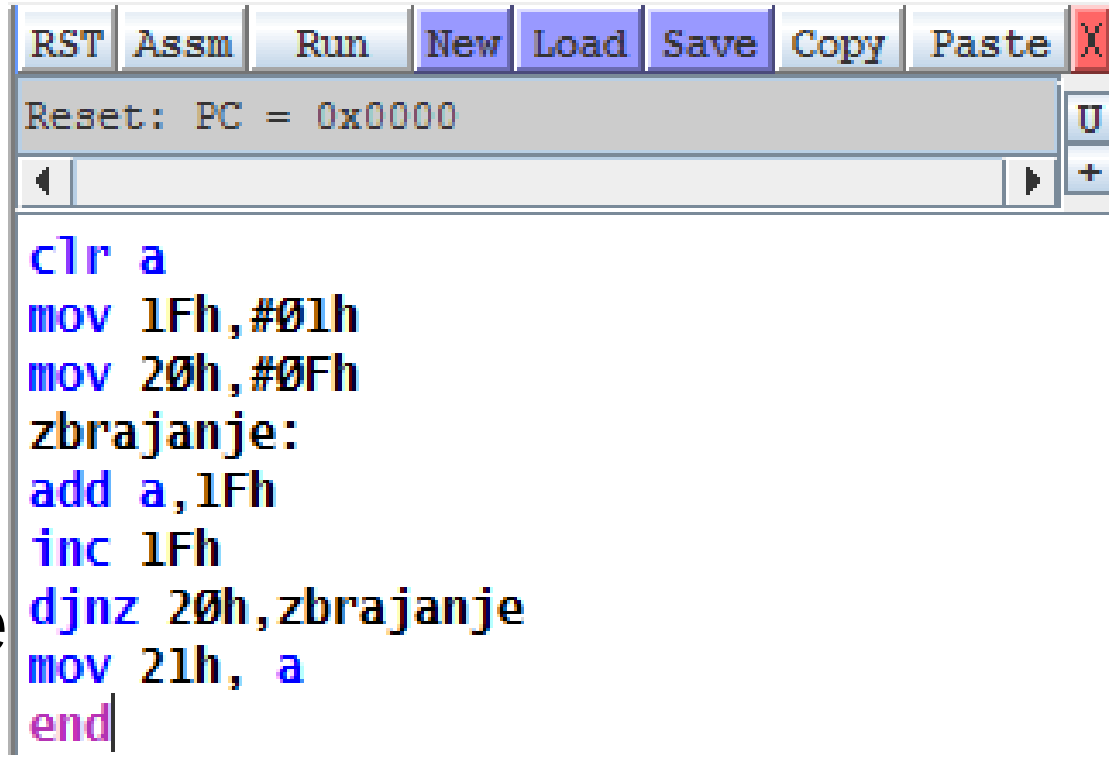
	1	2	3	4	5	6
R0	09	0A	0B	0C	0D	0E
A	09	13	1E	2A	37	45

Napišite program koji pomoću petlje računa zbroj brojeva od Ah do 10h. Za instrukciju petlje iskoristite DJNZ a registre odaberite po želji. U formi tablice prikažite sadržaj registara za svaku iteraciju petlje

```
org 0000h
mov r0,#6h //ovo je potrebno kao brojač
mov r1#10h
clr a
petlja:
add a,r1
dec r1
djnz r0,petlja
end
```

Koristeći akumulator, lokaciju 1Fh i lokaciju 20h zbrojiti brojeve od 1 do 15.
Rezultat pohraniti na lokaciji 21h

- org 0000h
- clr a
- mov 1Fh,#01h
- mov 20h,#0Fh
- zbrajanje:
- add a,1Fh
- inc 1Fh
- djnz 20h,zbrajanje
- mov 21h, a
- end



Napišite program koji pomoću petlje računa zbroj brojeva od 10h do 16h. Za instrukciju petlje iskoristite CJNE a registre odaberite po želji. U formi tablice prikažite sadržaj registara za svaku iteraciju petlje

```
org 0000h
mov r0,#10h
petlja:
add a,r0
inc r0
cjne r0,#17h,petlja
end
```

Memorija je definirana na rasponu adresa od 10 do 2A. Na svaku memorijsku lokaciju upišite brojku F. Program riješite petljom. Koristite registre i instrukciju petlje po želji

- CLR A
- MOV R0,#10H
- PETLJA:
- MOV @R0,#0FH
- INC R0
- CJNE R0,#2BH, PETLJA
- END