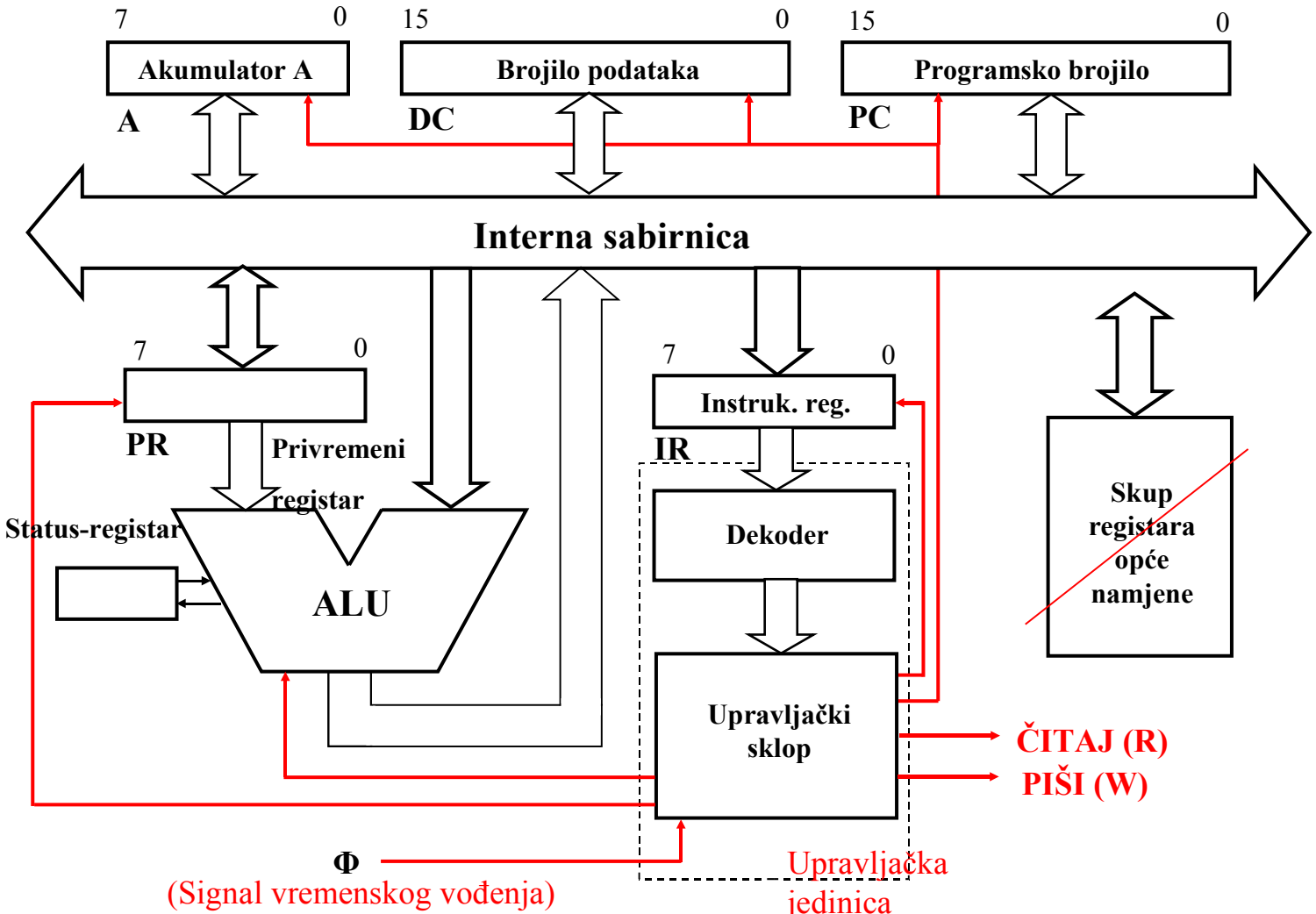
3. Pojednostavnjeni modeli CISC I RISC procesora

3.1. Model procesora CISC

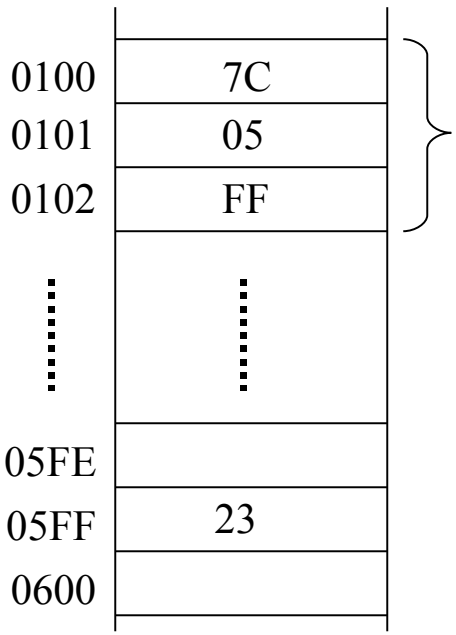


Komponente modela

- Akumulator A
- Programsko brojilo PC
- Instrukcijski registar IR
- Brojilo podataka DC
- Privremeni registar PR
- Status-registar (Registar stanja)
- ALU
- (Skup registara opće namjene)
- Interna sabirnica
- Upravljačka jedinica

3.2. Primjer izvođenja programa

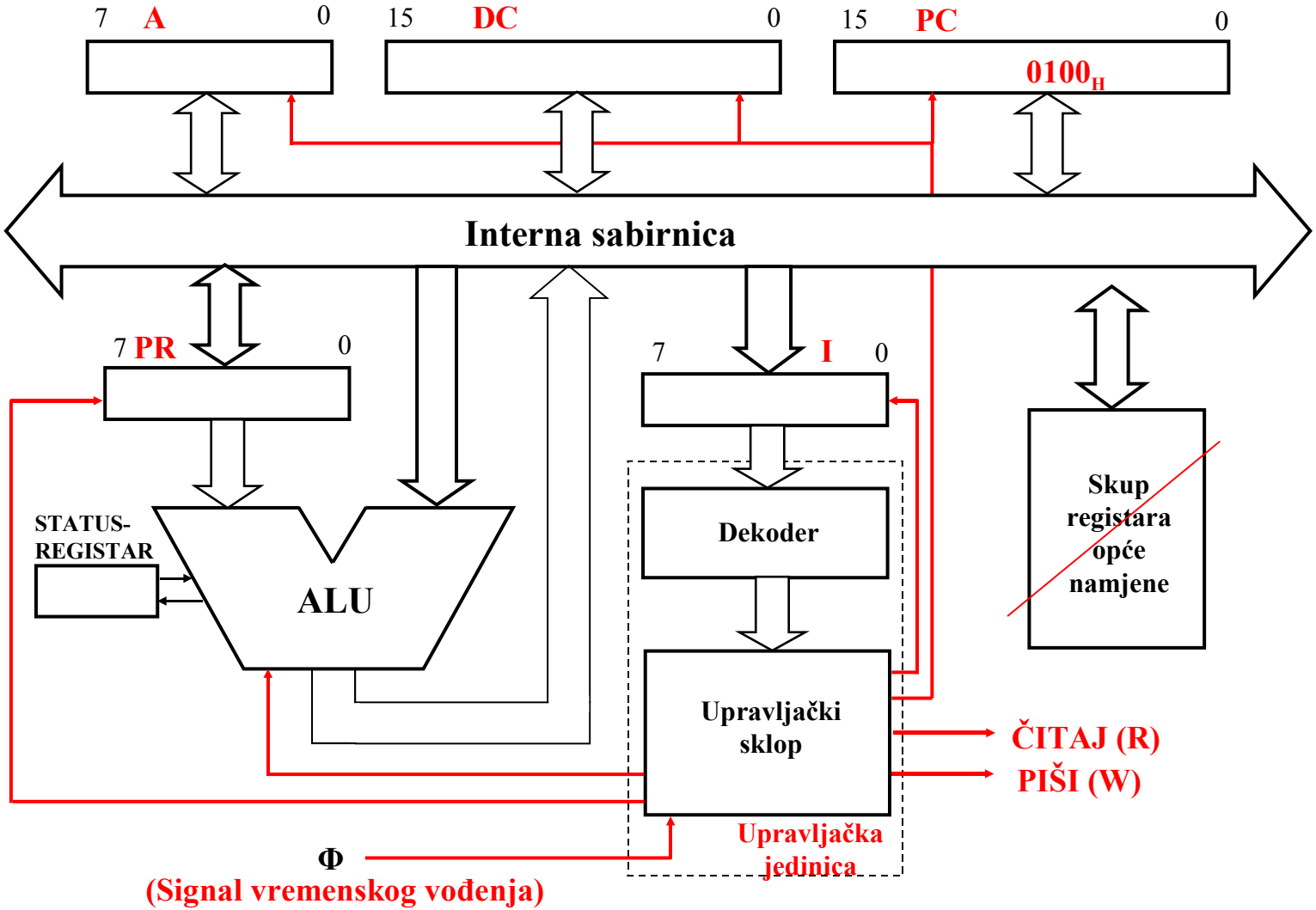
- Program samo od jedne instrukcije INC $05FF



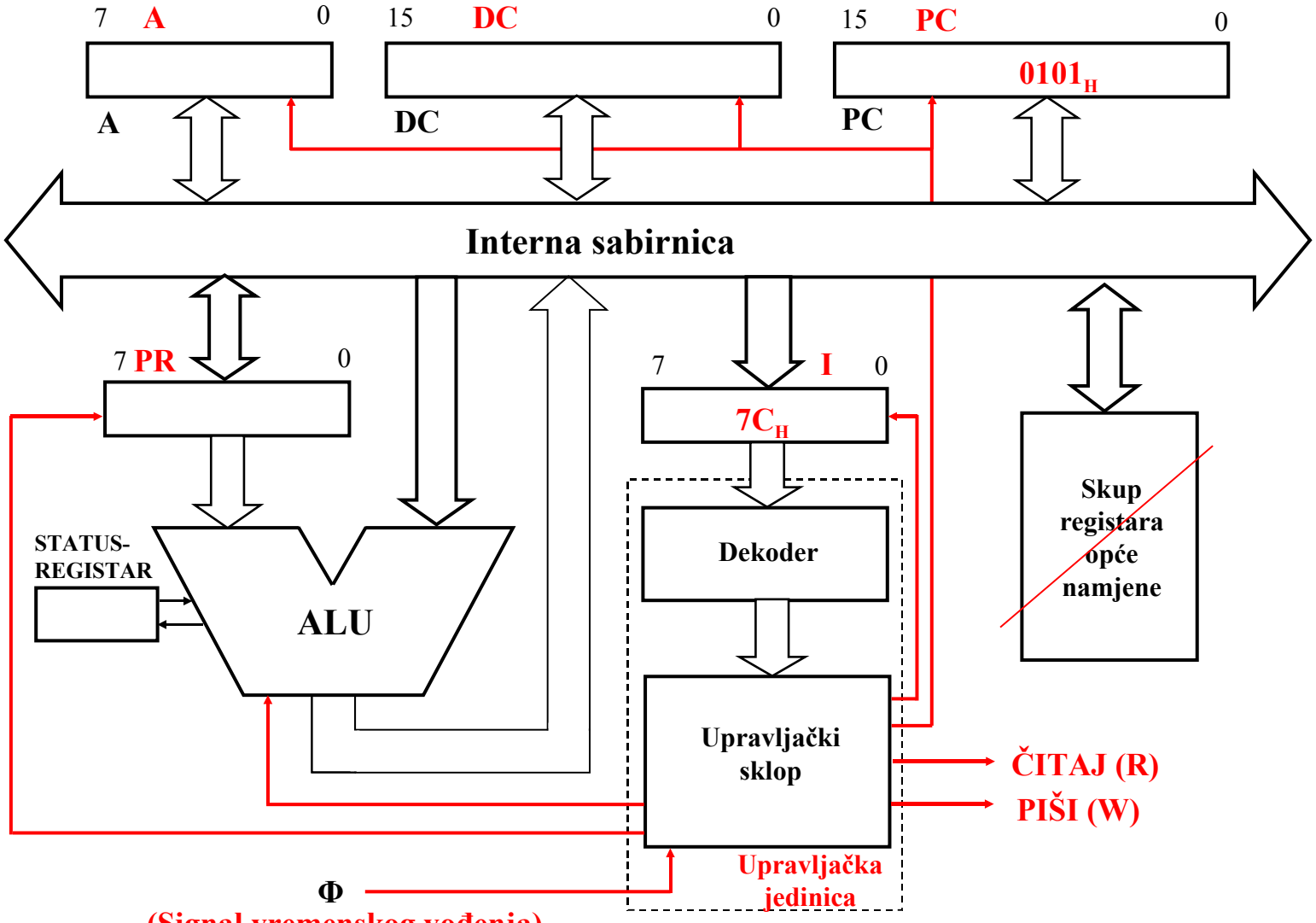
Inkrementiraj sadržaj memorijske lokacije s adresom 05FF

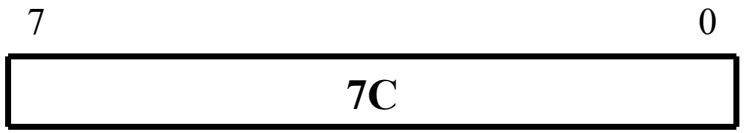
7C - operacijski kod
23 – podatak (operand) pohranjen u memoriji

Početni uvjeti



Stanje nakon pribavljanja operacijskog koda instrukcije





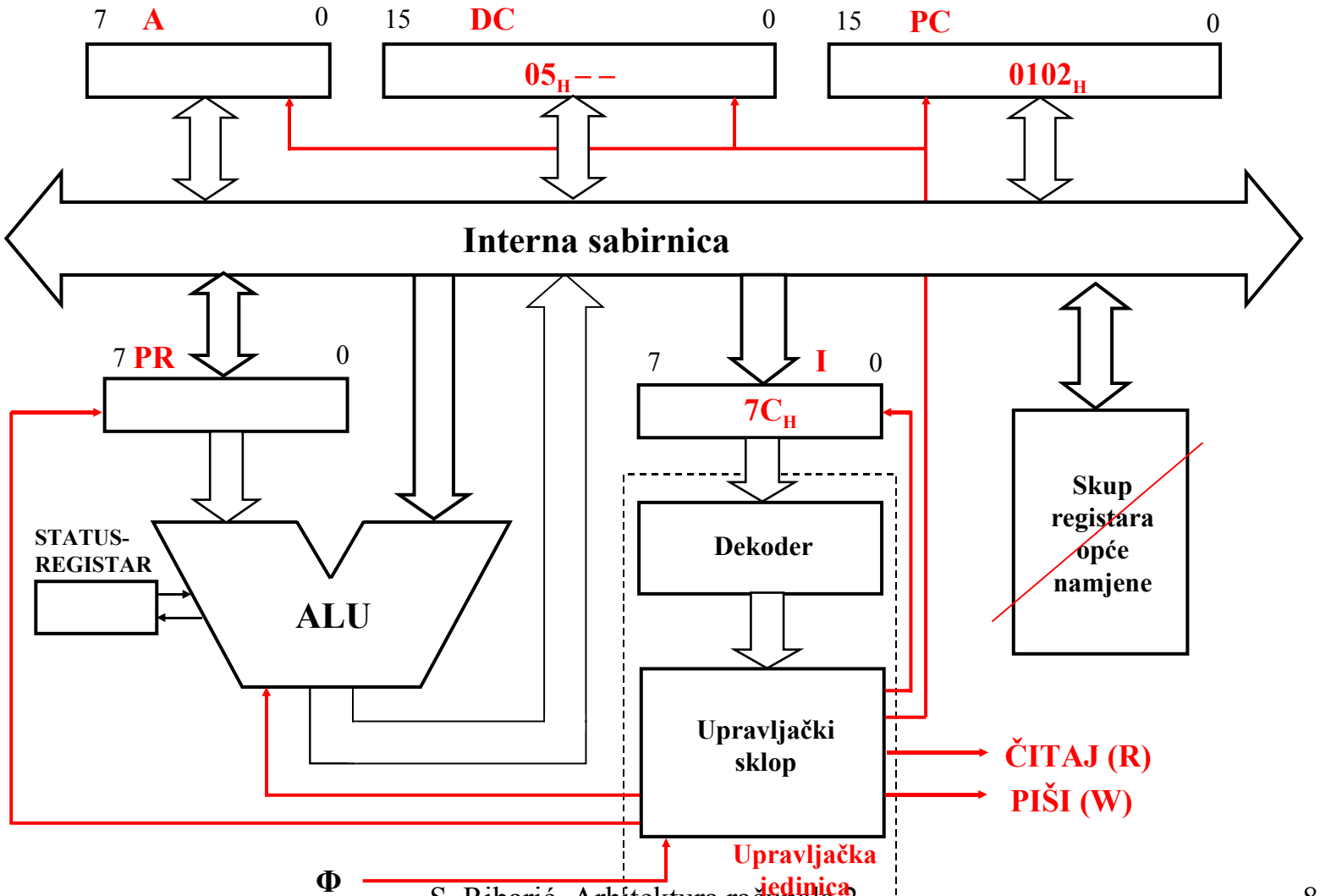
Operacijski kod

7C = 01111100

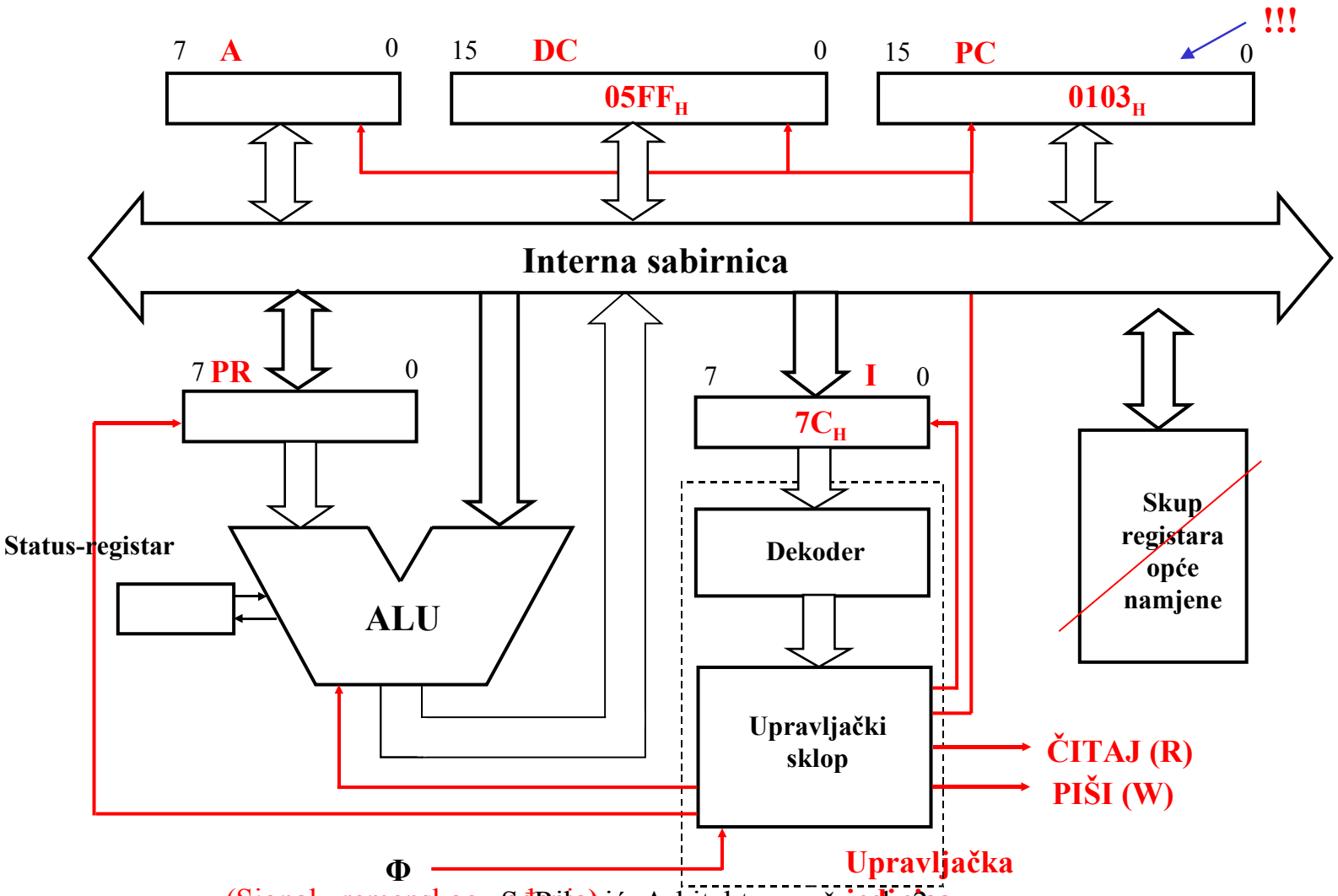
se tumači kao: Povećaj za 1 vrijednost operanda čija je adresa sadržana u dva bajta koja slijede ovom operacijskom kodu.

Stanje nakon pribavljanja značajnijeg bajta adrese operanda

Pazi: Faza Pribavi još uvijek traje!



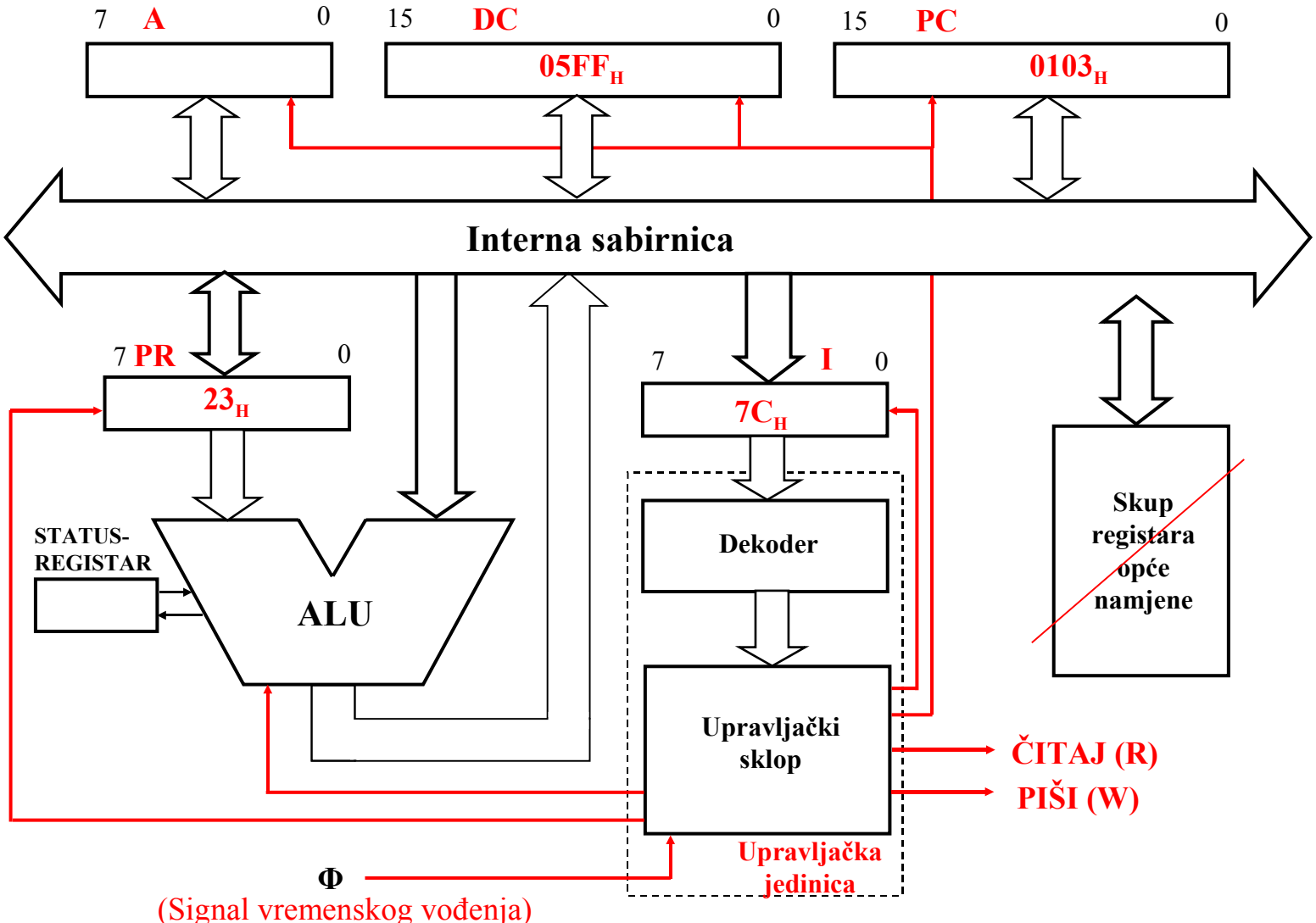
Stanje nakon pribavljanja manje značajnijeg bajta adrese operanda



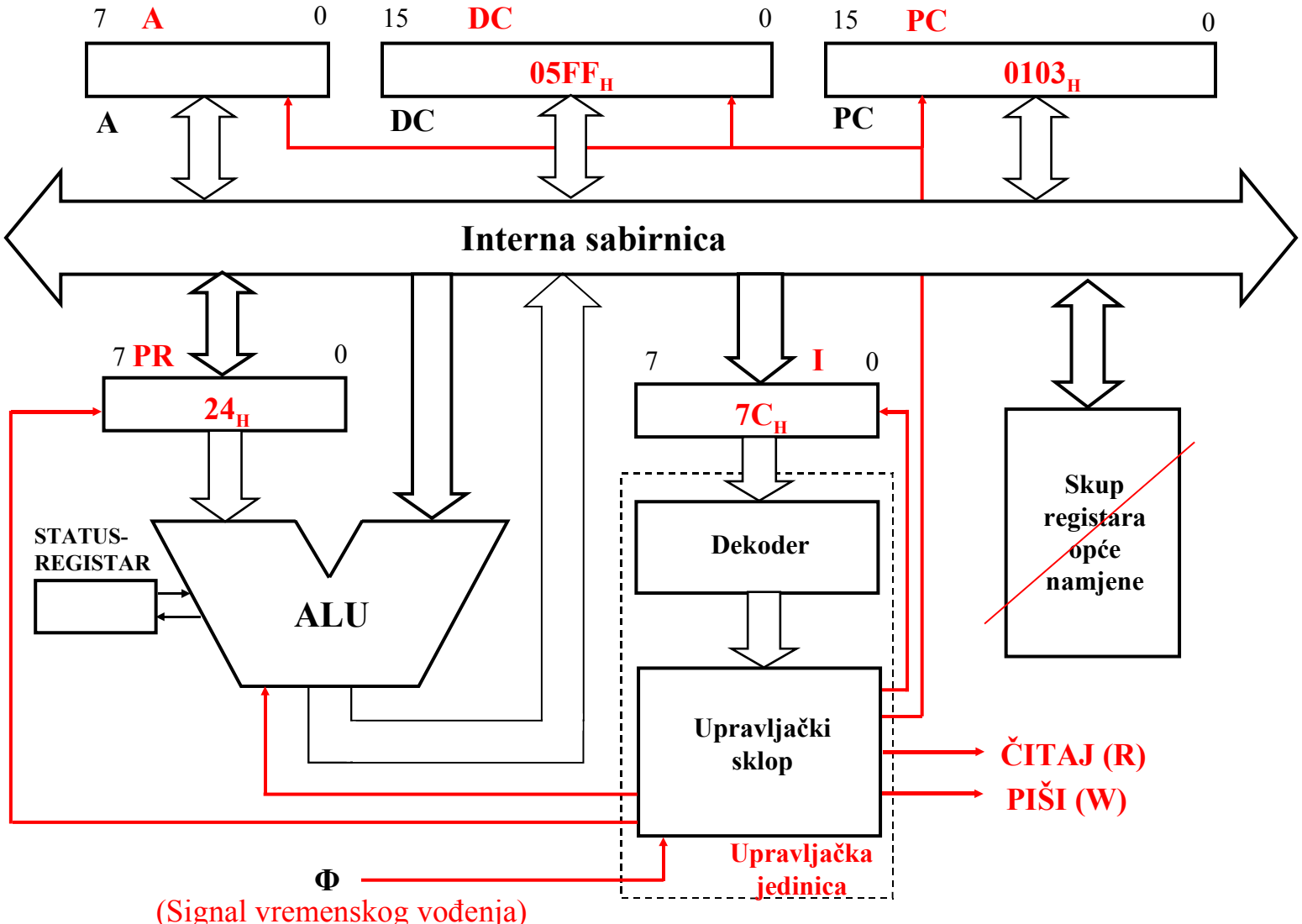
Faza Pribavi je završena!

Jednoadresni format instrukcije – kao i kod von Neumannovog računala

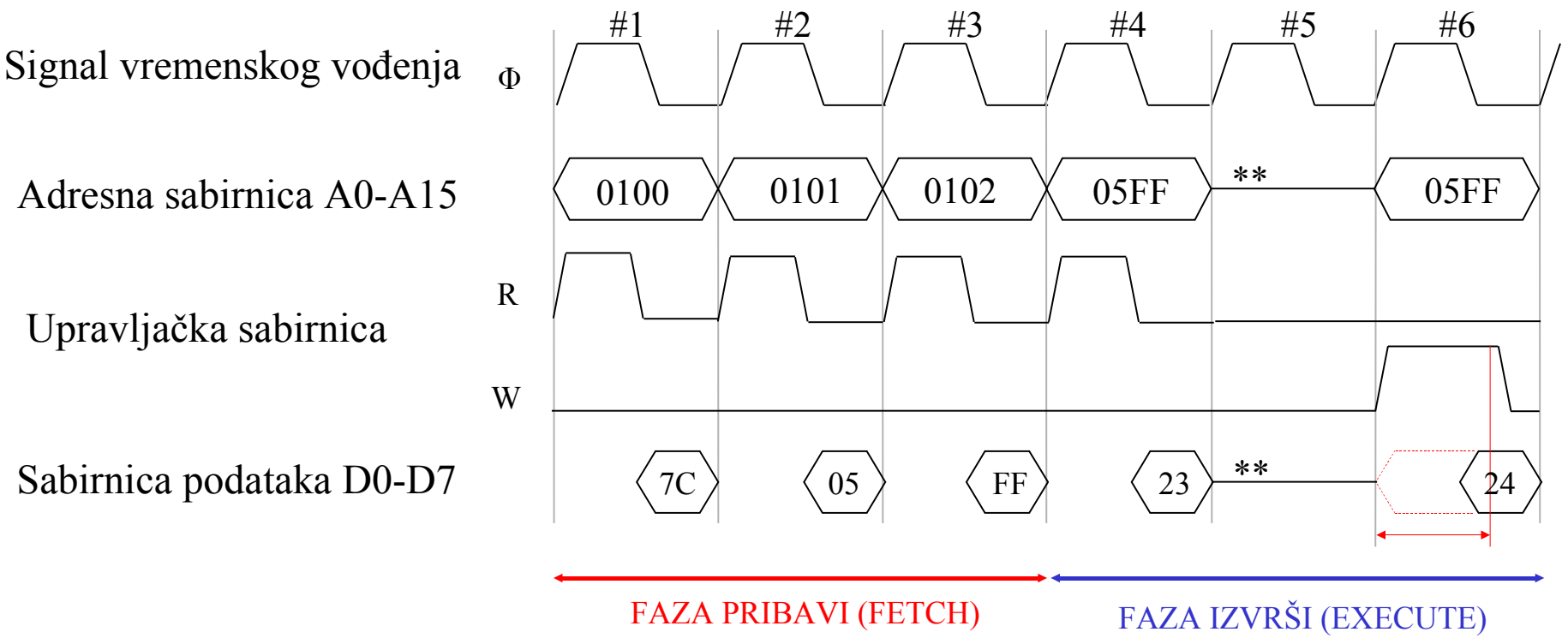
Stanje nakon dohvata operanda (faza IZVRŠI)



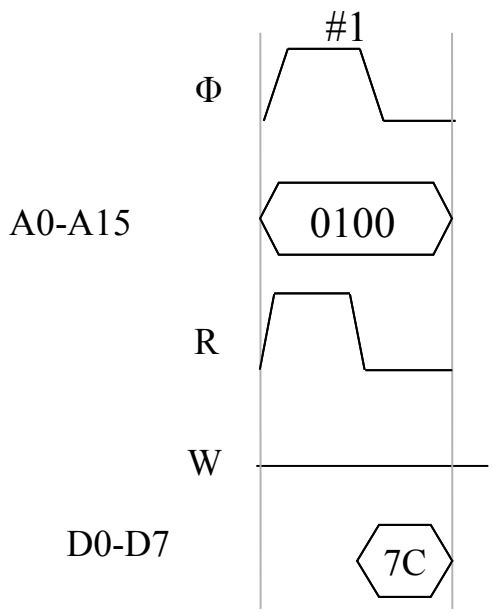
Stanje nakon povećanja operanda za jedan (faza IZVRŠI)



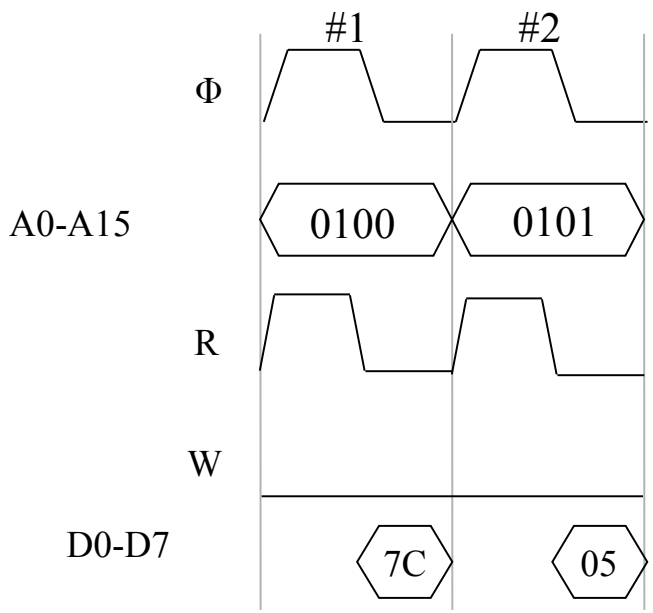
3.3. Stanje na vanjskim sabirnicama



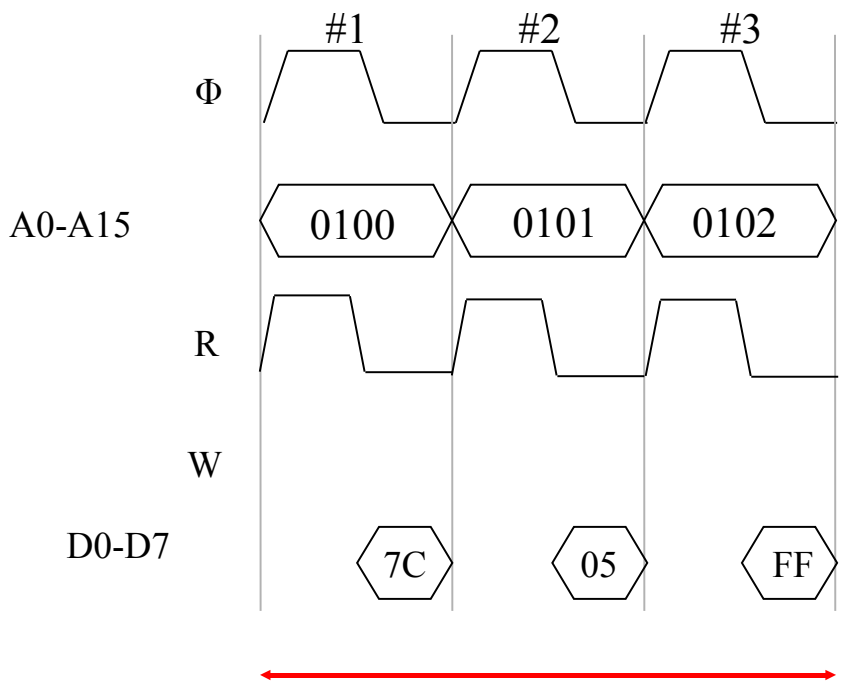
Prva perioda signala vremenskog vođenja



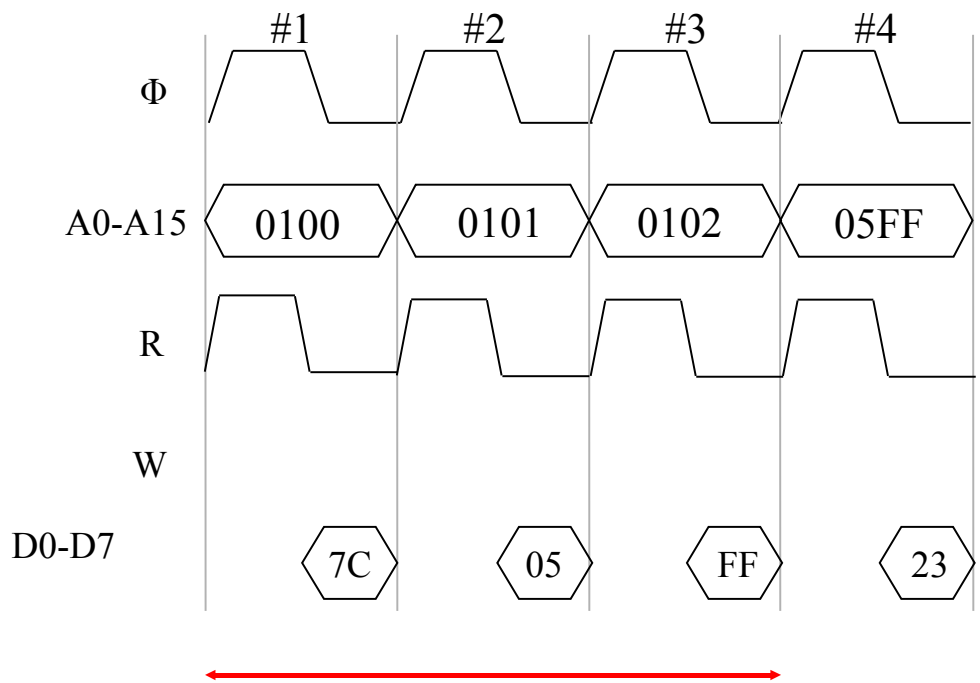
Druga perioda signala vremenskog vođenja



Treća perioda signala vremenskog vođenja



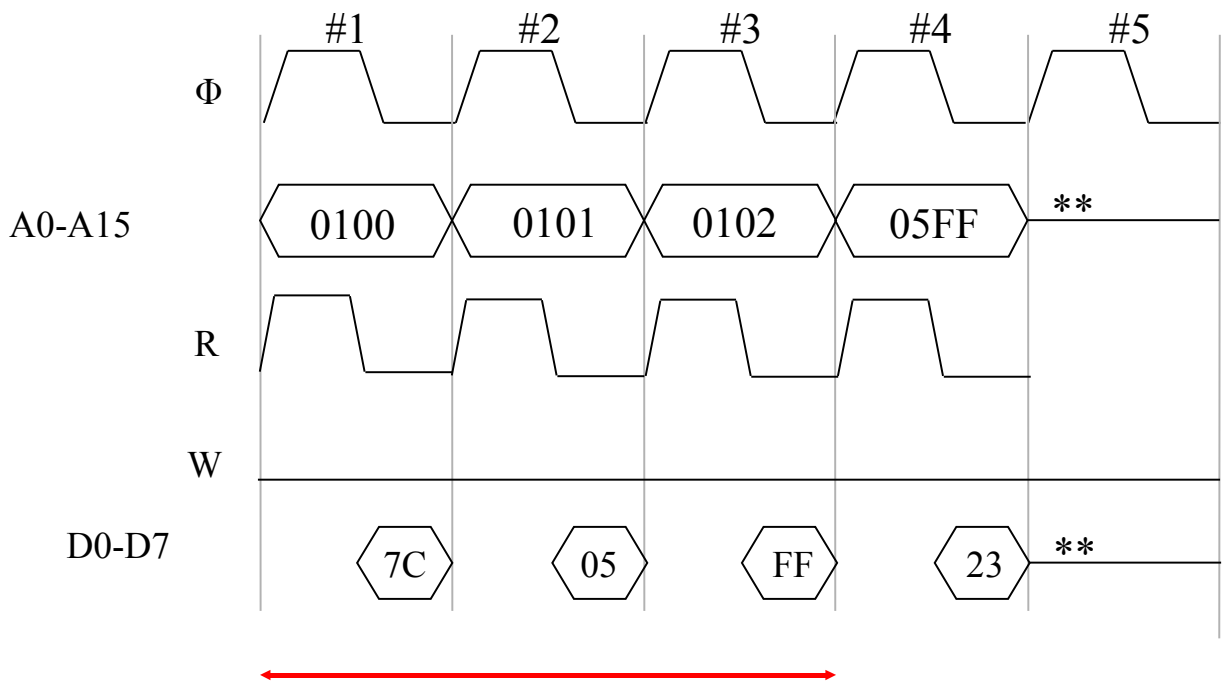
Četvrta perioda signala vremenskog vođenja



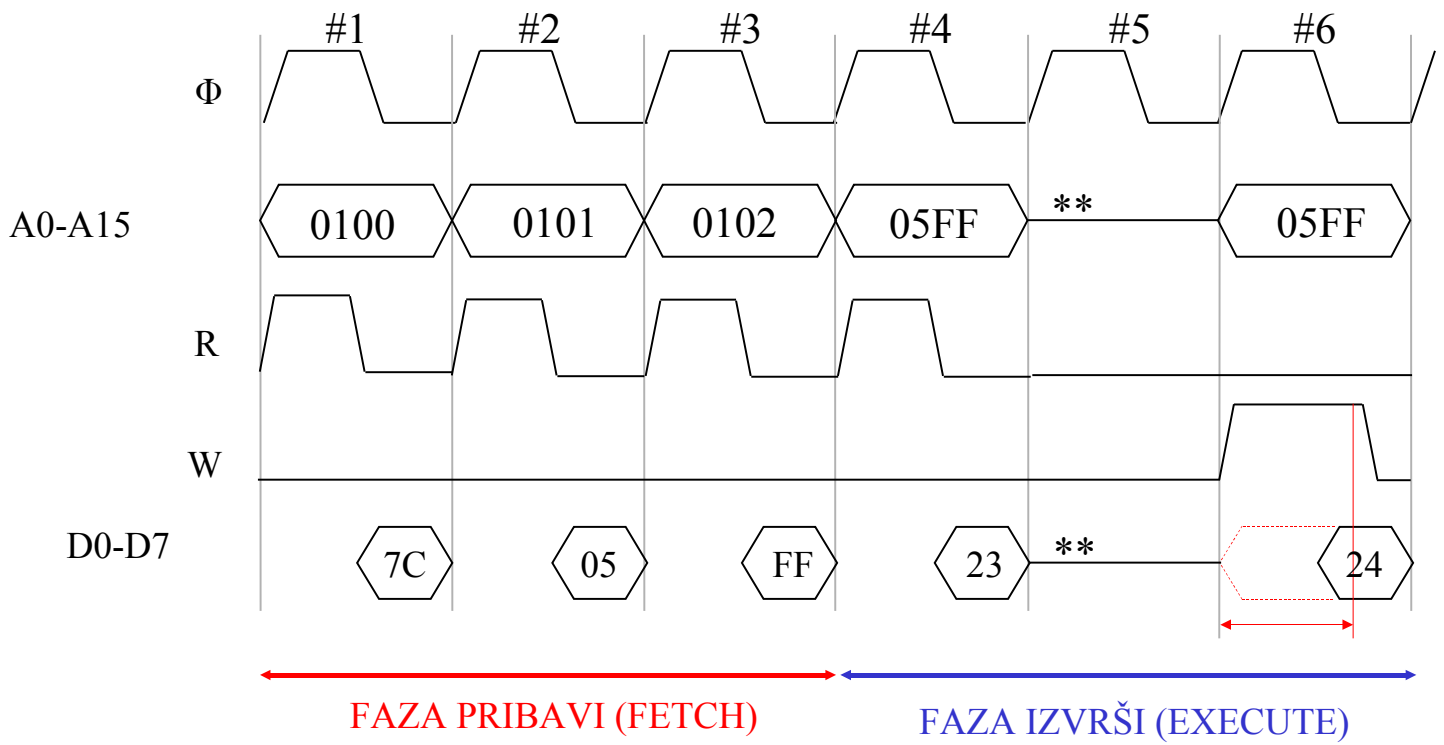
Peta perioda signala vremenskog vođenja

Pozor:

** - Označava stanje visoke impedancije /treće stanje/



Šesta perioda vremenskog vođenja



Motorola MC 6800

(izravni prošireni način adresiranja)

	OP	~	#
INC	7C	6	3

Zahtijeva 6 periode signala vremenskog vođenja!

Naš model obavlja ovu instrukciju također za 6 perioda!

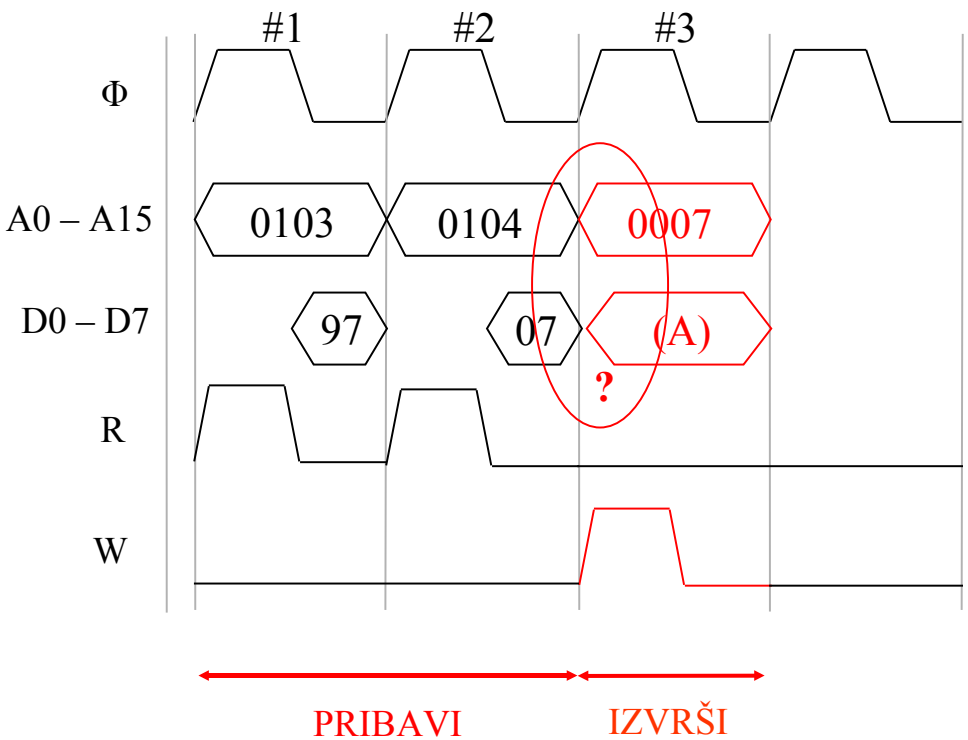
Zadatak 1.

Za računalo temeljeno na pojednostavljenom modelu procesora nacrtati stanje na sabirnicama za instrukciju

STA $07

(pohrani sadržaj akumulatora A na memorijsku lokaciju 0007 – izravni kratki način adresiranja;adresiranje nulte stranice).
Operacijski kod instrukcije je 97(heksadekadno) a instrukcija je pohranjena u memoriji na lokacijama 0103 i 0104 (heksadekadno).
Odrediti potreban broj perioda signala vremenskog vođenja i usporediti ga s onim koji se zahtijeva za MC 6800 (4 periode).

Stanje na sabirnicama za instrukciju STA $07



Motorola MC 6800

(izravni način adresiranja)

	OP	~	#
STA A	97	4	2

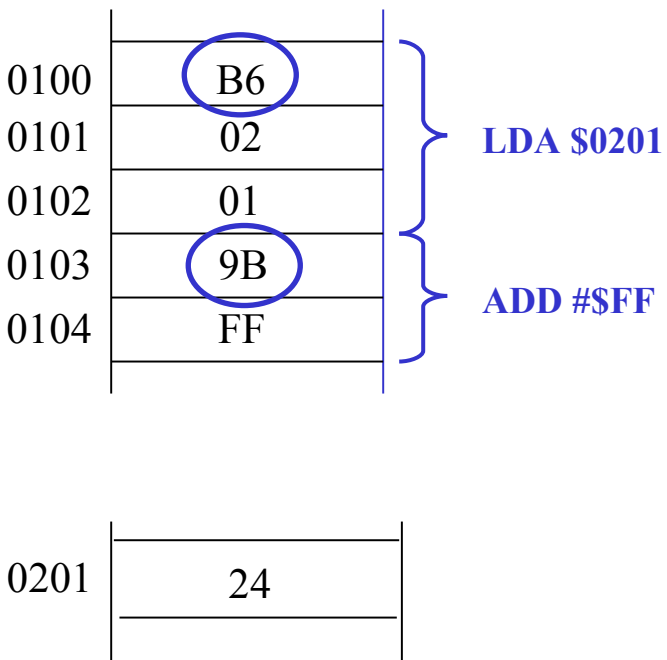
Zahtijeva 4 periode signala vremenskog vođenja!

Naš model obavlja ovu instrukciju tijekom 3 periode?!

Razlozi?

Zadatak 2.

Nacrtati stanje na sabirnicama za programski odsječak prikazan na slici. Odrediti stanja registara na početku, tijekom i nakon izvođenja programskog odsječka.



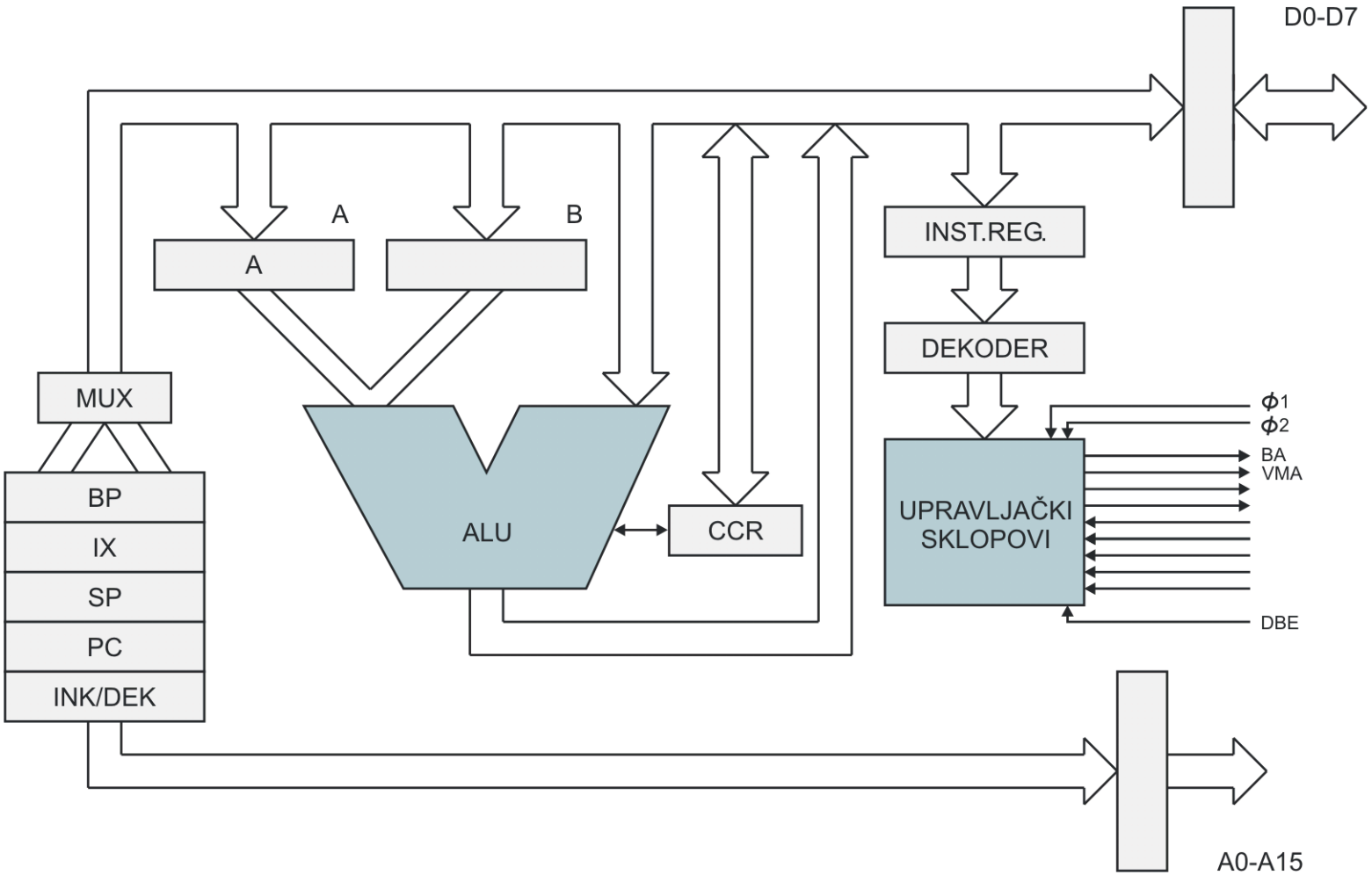
Zadatak 3.

Strojnu instrukciju *inc $A0A0* nadomjestiti programskim odsječkom (tri instrukcije) za registarsko orijentiran procesor koji operand (A0A0) = 0F prvo smješta u jedan od registara (r1), zatim sadržaj registra povećava za jedan te, napokon, tako uvećan sadržaj registra pohranjuje na memorijsku lokaciju A0A0. Za tako napisani programski odsječak nacrtajte stanje na sabirnicama tijekom njegovog izvođenja. Odredite broj potrebnih perioda signala vremenskog vođenja i usporedite ga s onim koji je potreban za instrukciju *inc $A0A0.*
(Potrebne podatke: operacijske kodove instrukcija, početnu adresu programskog odsječka te vrijednost operanda izaberite sami.)

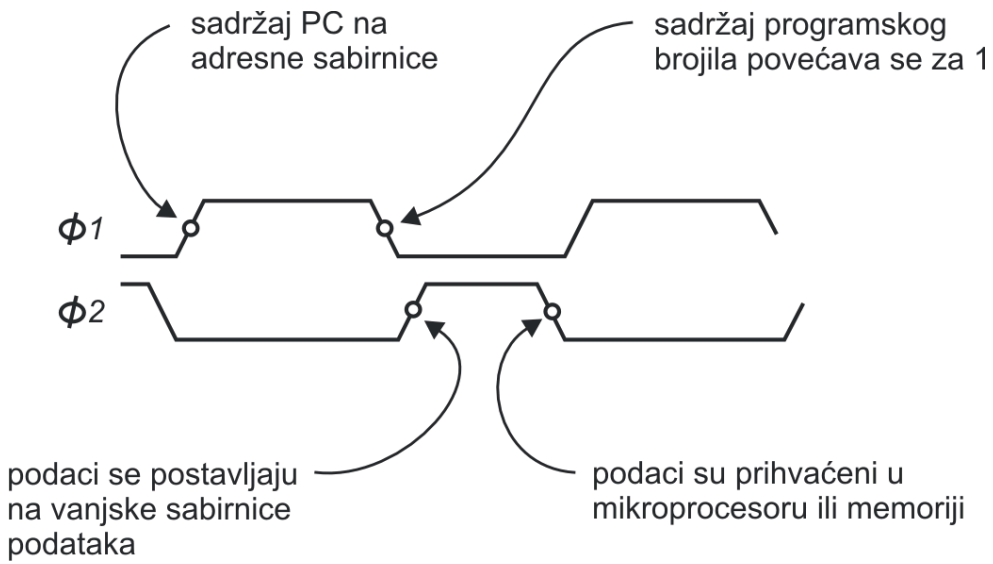
Zadatak 4.

Pročitati poglavlje “3. Pojednostavnjeni model mikroprocesora” u knjizi S. Ribarić, “Naprednije arhitekture mikroprocesora”, Element, Zagreb, 2002, pp. 25 – 39.

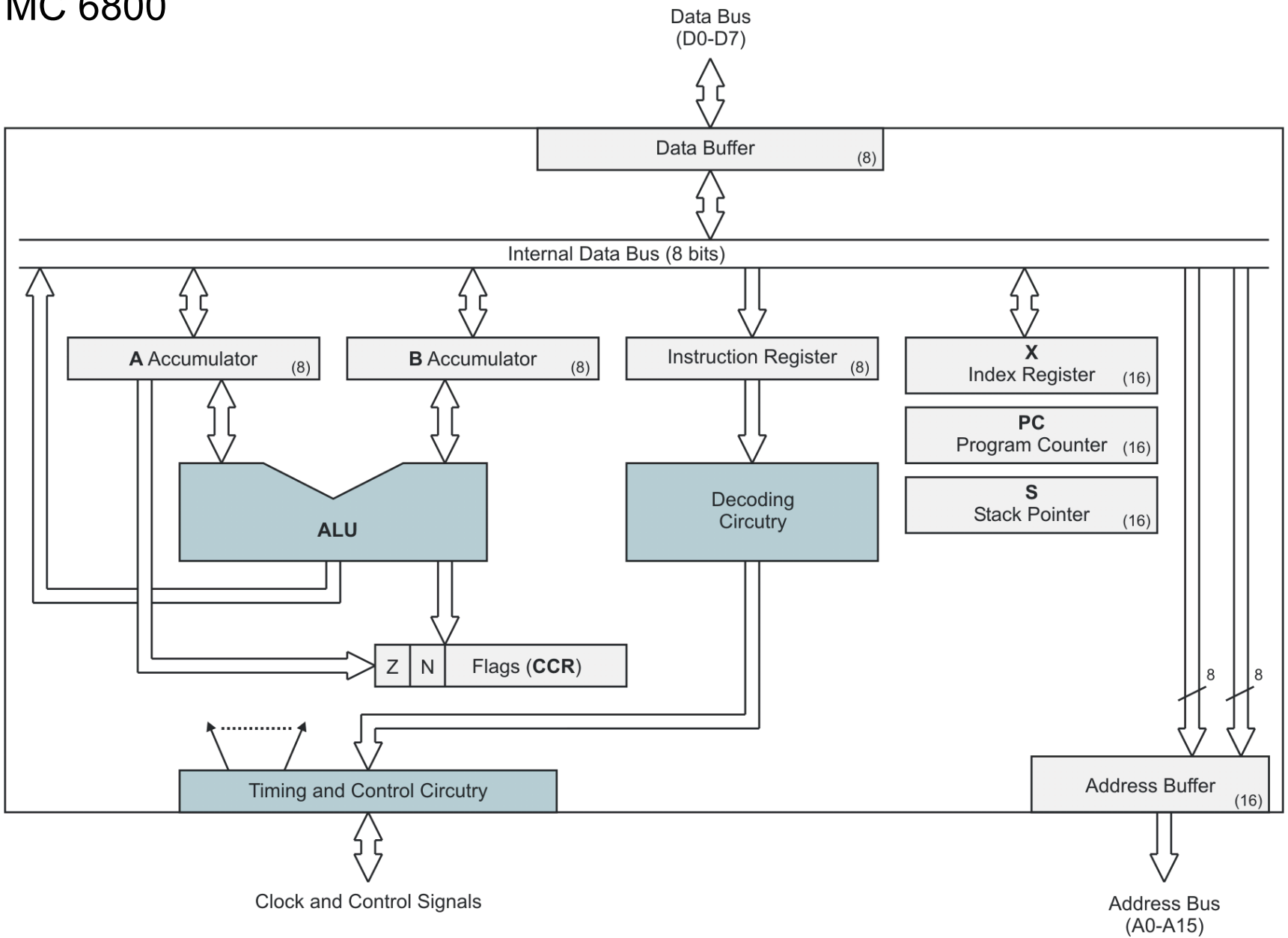
Primjer: Model 8-bitnog procesora MC 6800



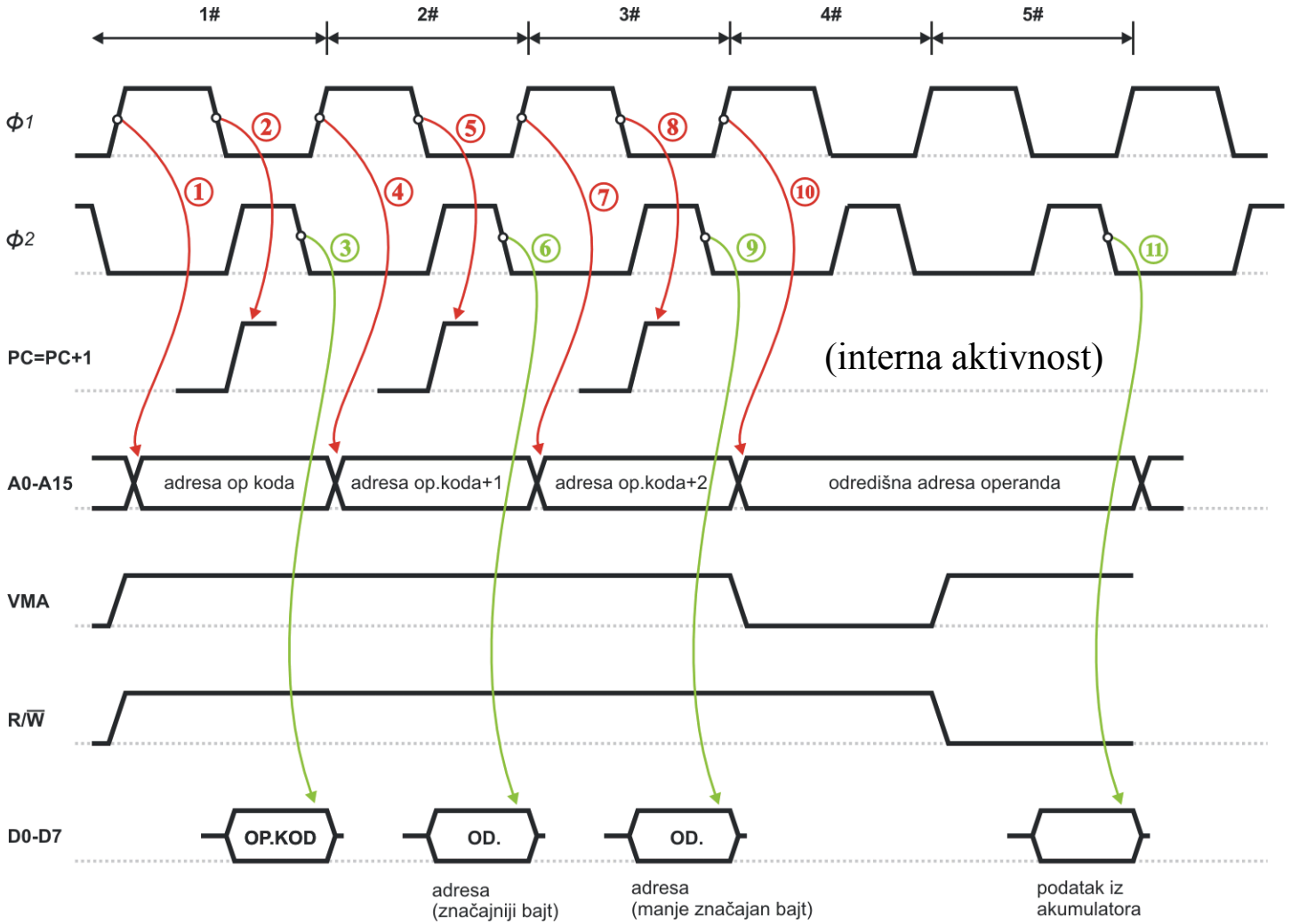
Signali vremenskog vođenja Φ 1 i Φ2



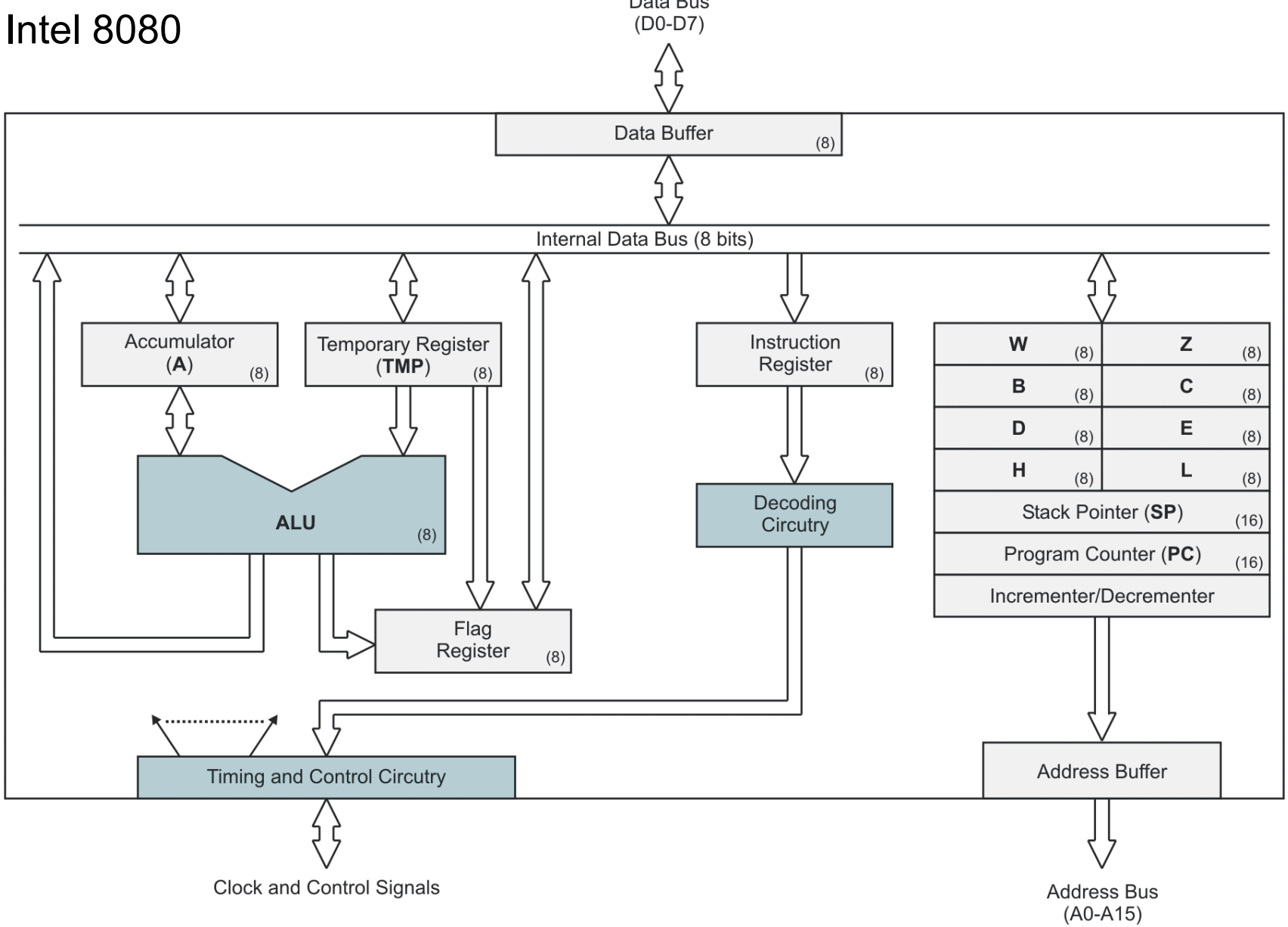
MC 6800


Data Bus
(D0-D7)
Data Buffer
(8)
Internal Data Bus (8 bits)
A Accumulator
(8)
B Accumulator
(8)
Instruction Register
(8)
X
Index Register
(16)
PC
Program Counter
(16)
S
Stack Pointer
(16)
ALU
Decoding
Circutry
Z
N
Flags (CCR)
8
8
Timing and Control Circutry
Address Buffer
(16)
Clock and Control Signals
Address Bus
(A0-A15)

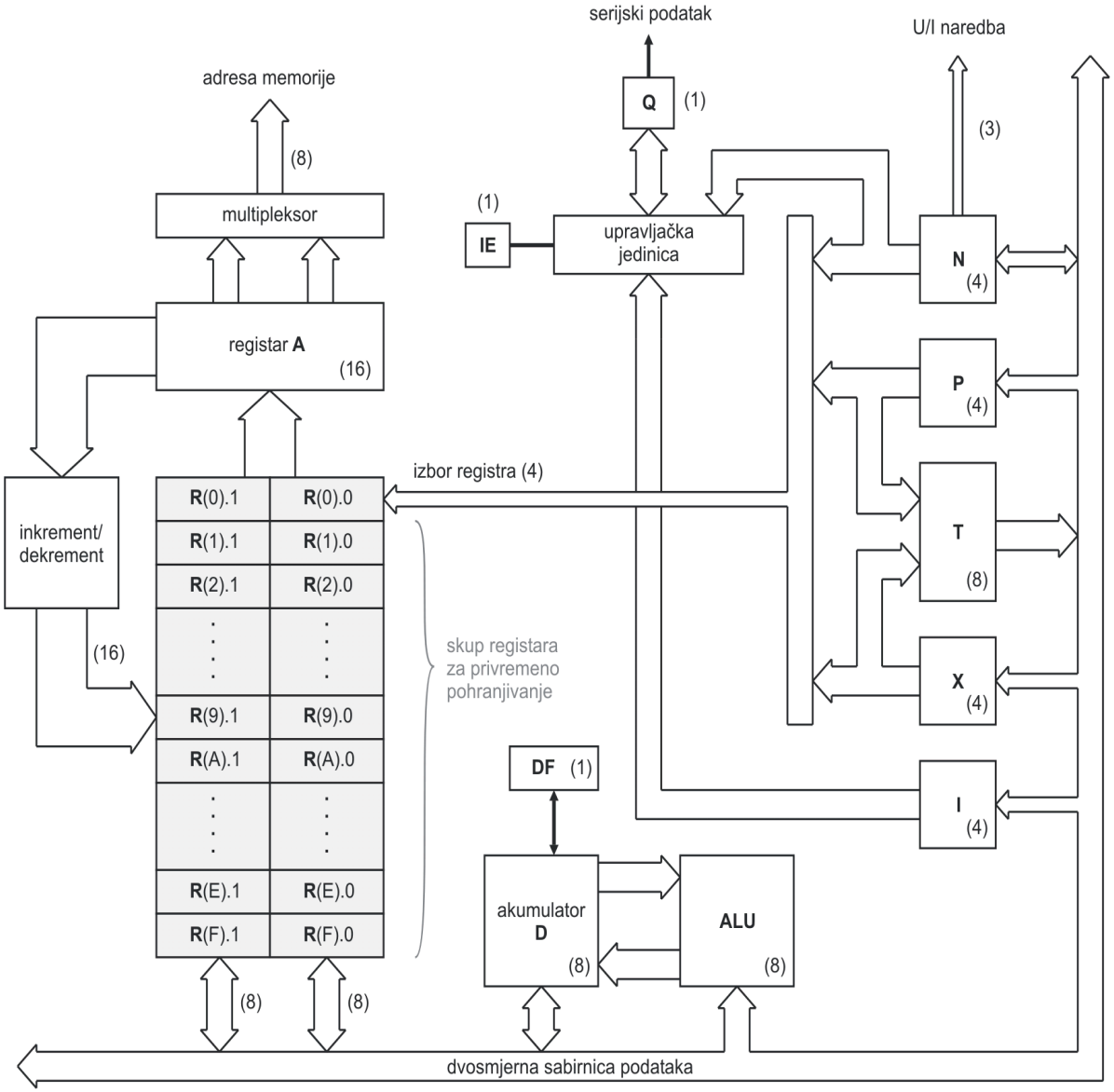
Stanje na sabirnicama za STA A $010F (MC 6800)



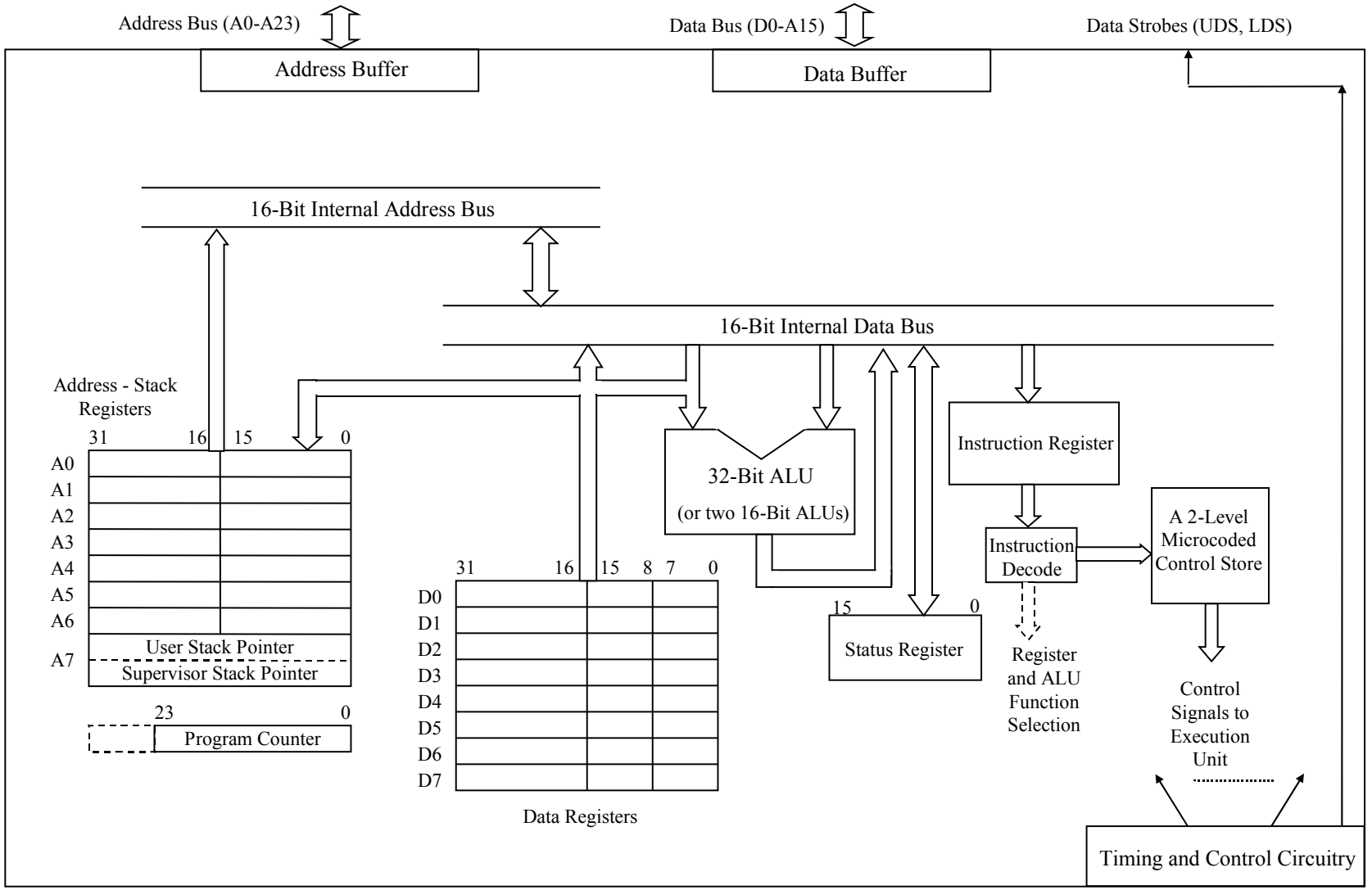
Intel 8080


Data Bus
(D0-D7)
Data Buffer
(8)
Internal Data Bus (8 bits)
Accumulator
(A)
(8)
Temporary Register
(TMP)
(8)
Instruction
Register
(8)
W
(8)
Z
(8)
B
(8)
C
(8)
D
(8)
E
(8)
H
(8)
L
(8)
Stack Pointer (SP)
(16)
Program Counter (PC)
(16)
Incrementer/Decrementer
ALU
(8)
Decoding
Circuitry
Flag
Register
(8)
Timing and Control Circuitry
Address Buffer
Clock and Control Signals
Address Bus
(A0-A15)

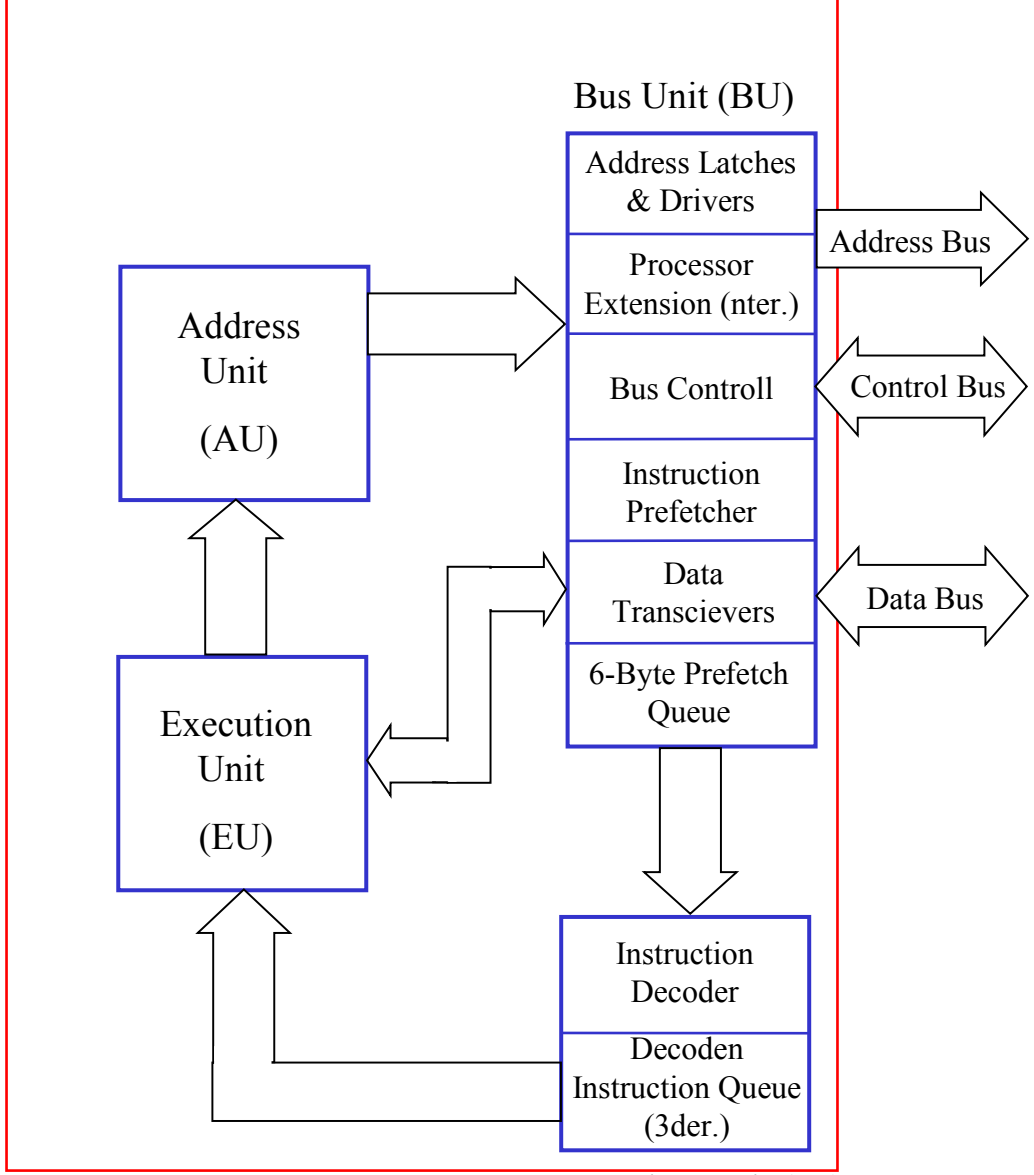
RCA 1802


serijski podatak
U/I naredba
adresa memorije
Q
(1)
(3)
(8)
multipleksor
(1)
IE
upravljačka jedinica
N
(4)
registar A
(16)
P
(4)
izbor registra (4)
inkrement/ dekrement
R(0).1
R(0).0
T
(8)
R(1).1
R(1).0
R(2).1
R(2).0
(16)
skup registara za privremeno pohranjivanje
X
(4)
R(9).1
R(9).0
R(A).1
R(A).0
DF (1)
I
(4)
akumulator D
(8)
ALU
(8)
R(E).1
R(E).0
R(F).1
R(F).0
(8)
(8)
dvosmjerna sabirnica podataka

MC 68000 – registarsko orijentiran procesor



Intel 80286


Bus Unit (BU)
Address Latches & Drivers
Address Bus
Processor Extension (nter.)
Address Unit (AU)
Bus Controll
Control Bus
Instruction Prefetcher
Data Transcievers
Data Bus
6-Byte Prefetch Queue
Execution Unit (EU)
Instruction Decoder
Decoden Instruction Queue (3der.)
Instruction Unit (IU)

Intel 80286 i 80386

CLOCK input

Frekvencija signala na ulazu CLOCK interno se dijeli s 2:

fPCLOCK(ProcessorCLOCK) = ½ fCLOCK

Intel 486 DX

fPCLOCK(ProcessorCLOCK) = fCLOCK

Proizvođači se obično referenciraju na PCLOCK

Intel 486 DX4

fPCLOCK(ProcessorCLOCK) = 2fCLOCK
3fCLOCK

Izvor: T. Shanley, D. Anderson, ISA System Architecture, Addison-Wesley, 1995.

3.4. Sabirnički ciklus

Kada procesor izvodi operaciju ČITANJA ili PISANJA, započinje slijed događaja koji se naziva SABIRNIČKI CIKLUS (Bus Cycle).

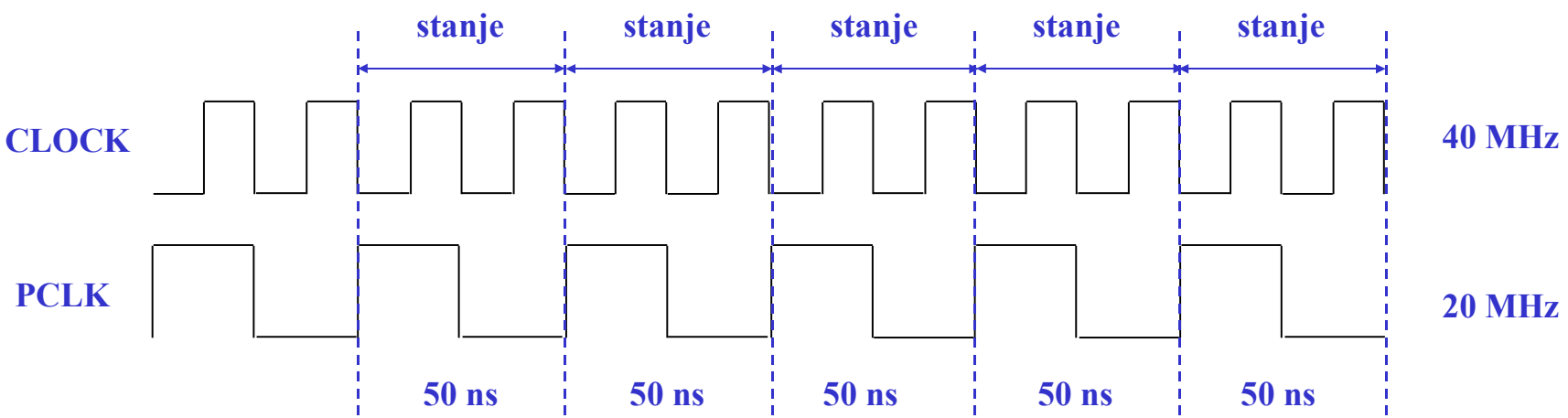
Tijekom sabirničkog ciklusa:

- procesor postavlja adresu na adresnu sabirnicu
- generira upravljačke signale i postavlja ih na upravljačku sabirnicu
- označava vrstu prijenosa
- prenose se podaci između ciljne lokacije i procesora

x86 procesori imaju podsustav – sabirničku jedinicu (bus unit) koja podržava sabirnički ciklus

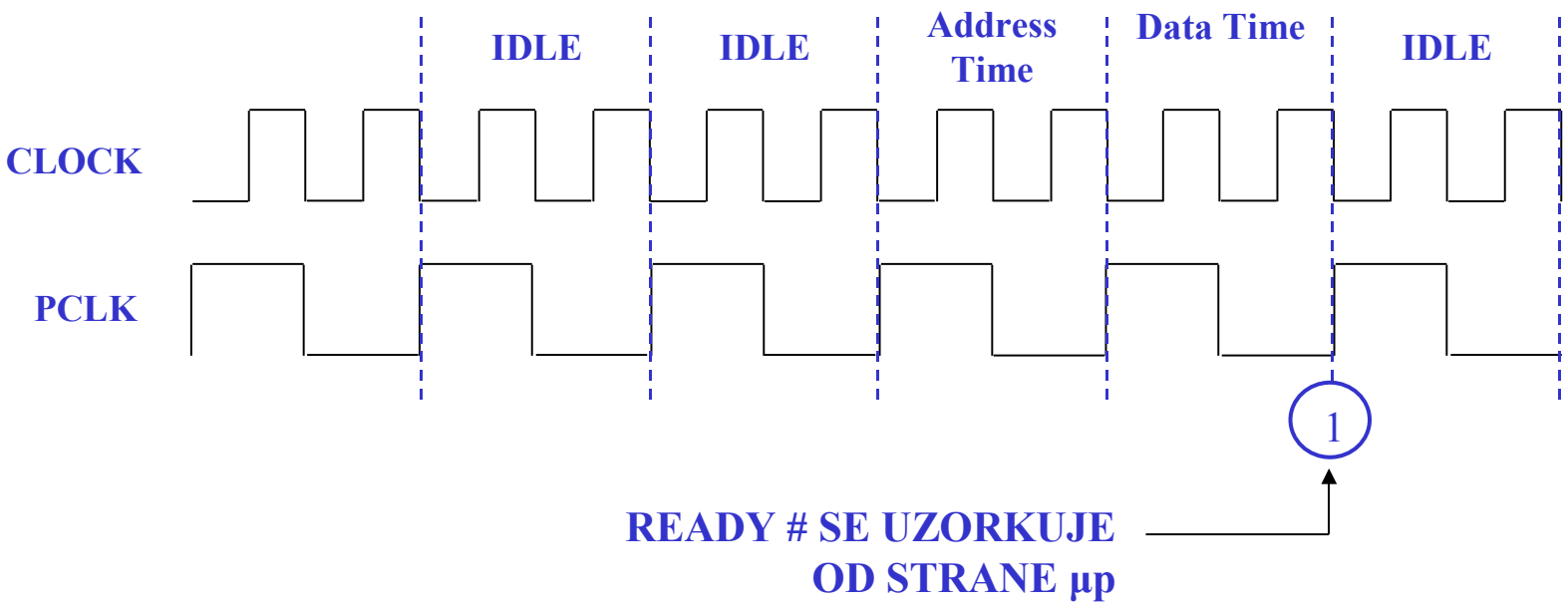
Sabirnička jedinica – stroj stanja

Vrijeme trajanja stanja ⟶ perioda PCLOCK-a



Stanja:

- Vrijeme neaktivnosti (Idle)
- Vrijeme adresiranja (Address Time)
- Vrijeme podataka (Data Time)



Vrijeme adresiranja:

- Sabirnička jedinica započinje sabirnički ciklus
iz stanja IDLE prelazi u stanje Vrijeme adresiranja;

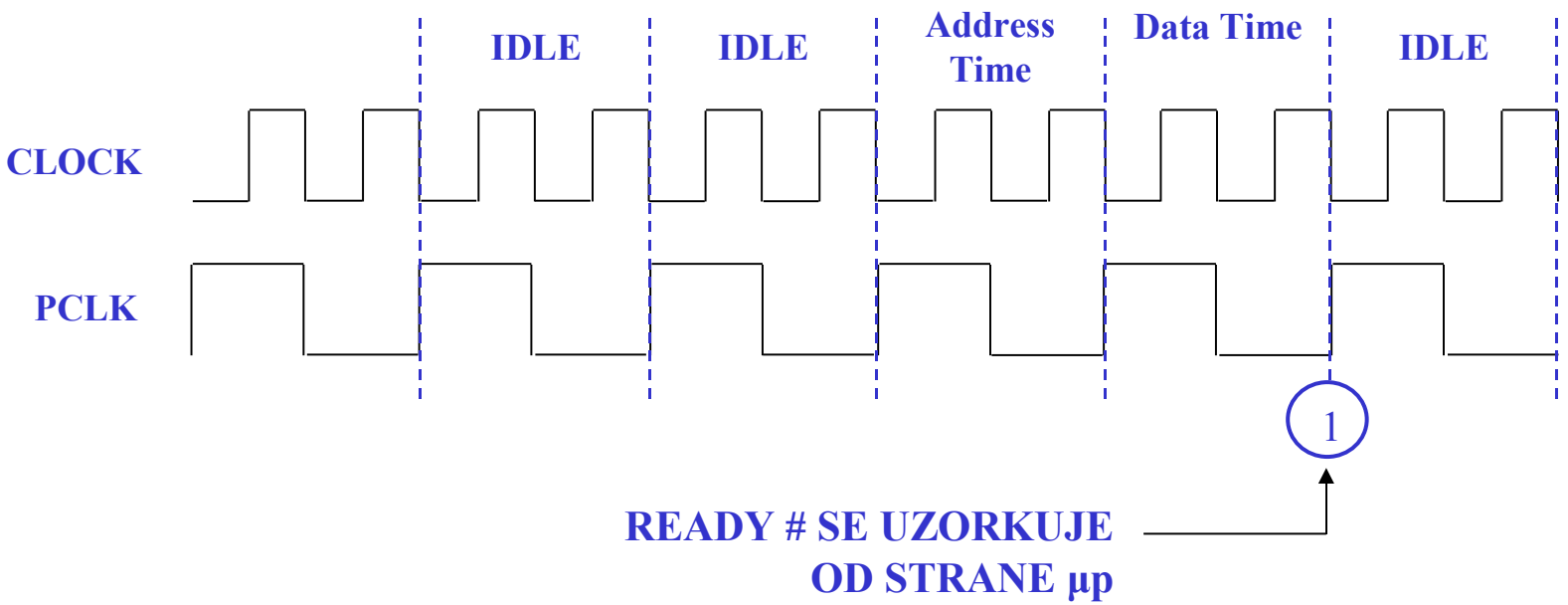
- Tijekom tog stanja (traje jednu periodu PCLOCK-a)
sabirnička jedinica postavlja adresu na adresnu sabirnicu a na upravljačku sabirnicu postavlja informaciju o vrsti sabirničkog ciklusa;

Vrijeme podataka:

- Iz stanja Vremena adresiranja sabirnička jedinica prelazi u stanje Vrijeme podataka;

-Tijekom tog stanja (traje jednu periodu PCLOCK-a) sabirnička jedinica obavlja prijenos podataka;
- Na kraju tog stanja, sinkrono sa signalom PCLOCK, procesor ispituje stanje ulaza READY#;
- Ako je READY# nisko, tj. logičko “0” sabirnička jedinica završava sabirnički ciklus;

0 – Wait-state bus cycle (sabirnički ciklus sa nula stanja čekanja)



Najbrži sabirnički ciklus x86 procesora traje dvije periode PCLOCK-a

ISA sabirnica (Industry Standard Architecture)

- Proširenje IBM PC sabirnice /temelji se na Intel 8088 sustavima;
- 62 signalne linije (20 linija za adresiranje, 8 linija podataka,
  te upravljačke linije (npr., Memory read, Memory Write,
  I/O read, I/O write, linije za zahtijevanje prekida, potvrdu prekida,
  te linije za DMA);
- Frekvencija (PCLOCK-a) 8.33MHz /brzina prijenosa 4.166MB/s
  ili 8.33 MB/s (za 16 bitnu sabirnicu podataka);

EISA (Extended ISA)

- 32-bitna adresna sabirnica
- 8-,16- ili 32-bitni prijenos podataka
- Brzina prijenosa 33 MB/s

PCI (Peripheral Component Interconnect Bus)

Potreba za većom propusnosti sabirnice!

Primjer:

Slika /zaslon/ rezolucije 1024 x 768 slikovnih elemenata (pixel) –true color (3 bajta/pixel)

1 zaslon = 2.25 MB podataka

Slika u pokretu: 30 zaslona/s.

Potrebna brzina: 67.5 MB/s

Prikaz video zapisa sa čvrstog diska, CD-ROM-a ili DVD-a
podaci se s diska preko sabirnice prenose u memoriju i tek tada, da bi se prikazali, putuju preko sabirnice prema grafičkoj kartici

Potrebna brzina prijenosa 135 MB/s!

/to je samo za video – ne uzimajući u obzir potrebe CPU-a i drugih uređaja priključenih na sabirnicu/

Intel, 1990. godine uvodi PCI sabirnicu za računala temeljena na njihovim procesorima Pentium;

Sun – Ultra SPARC II

Izvorno: PCI prenosi 32 bita/perioda i radi na 33 MHz (perioda 30 ns)

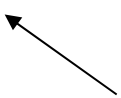
- Brzina prijenosa 133 MB/s;

1993. godine: PCI 2.0
1995. godine: PCI 2.1

PCI 2.2

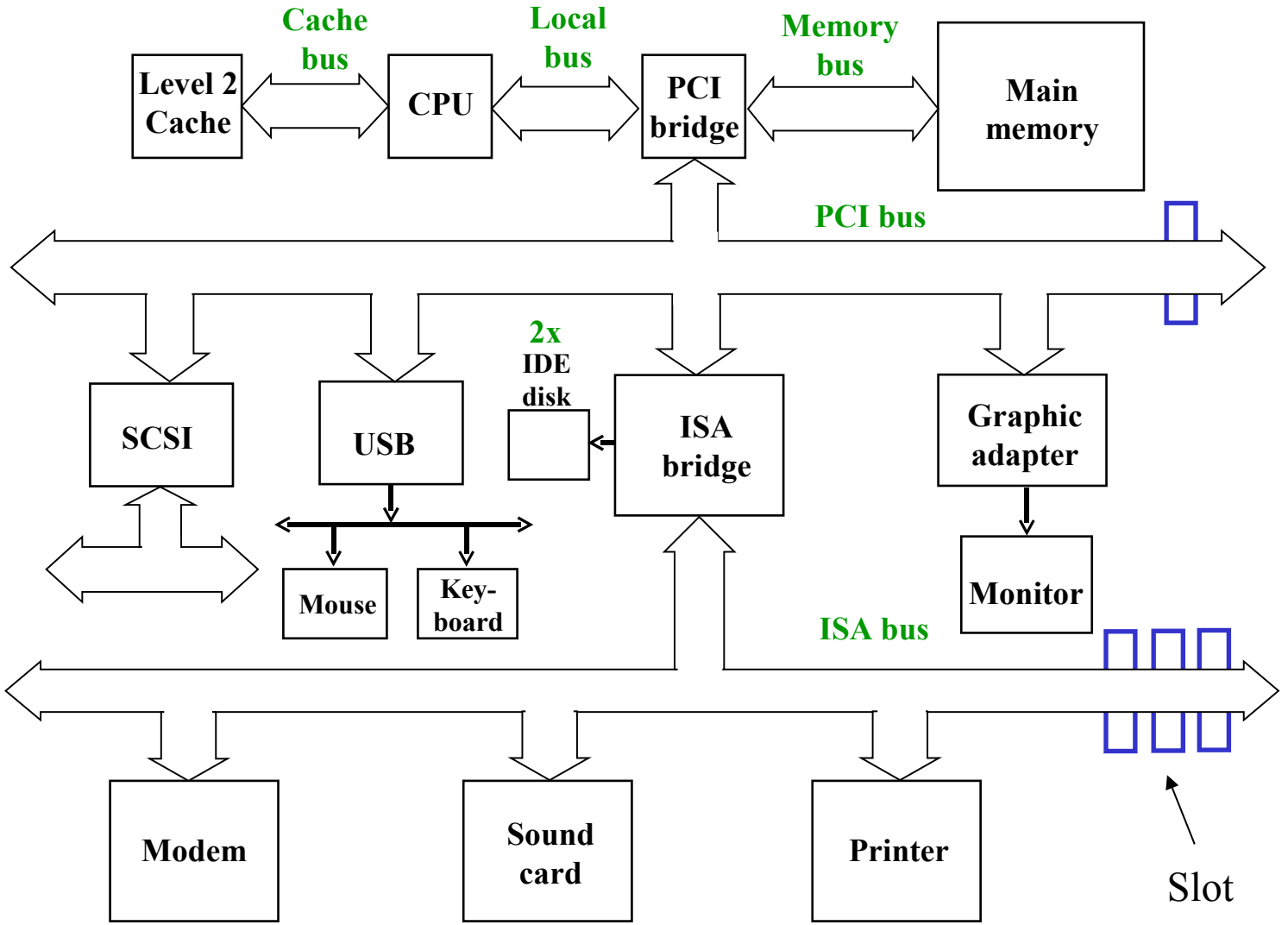
- 66 MHz i rukuje 64-bitnim prijenosom

- Brzina prijenosa 528 MB/s



Još uvijek nije dovoljna brzina prijenosa (npr., za memorijsku sabirnicu);
Brzina nije kompatibilna s ISA/EISA modulima;

Intel ponudio rješenje:



IDE - Integrated Drive Electronics

EIDE – Extended IDE (adresna shema LBA /Logical Block Address; 0 do 224 – 1 broj sektora; kapacitet diskova ≥ 528 MB/)

USB - Universal Serial Bus

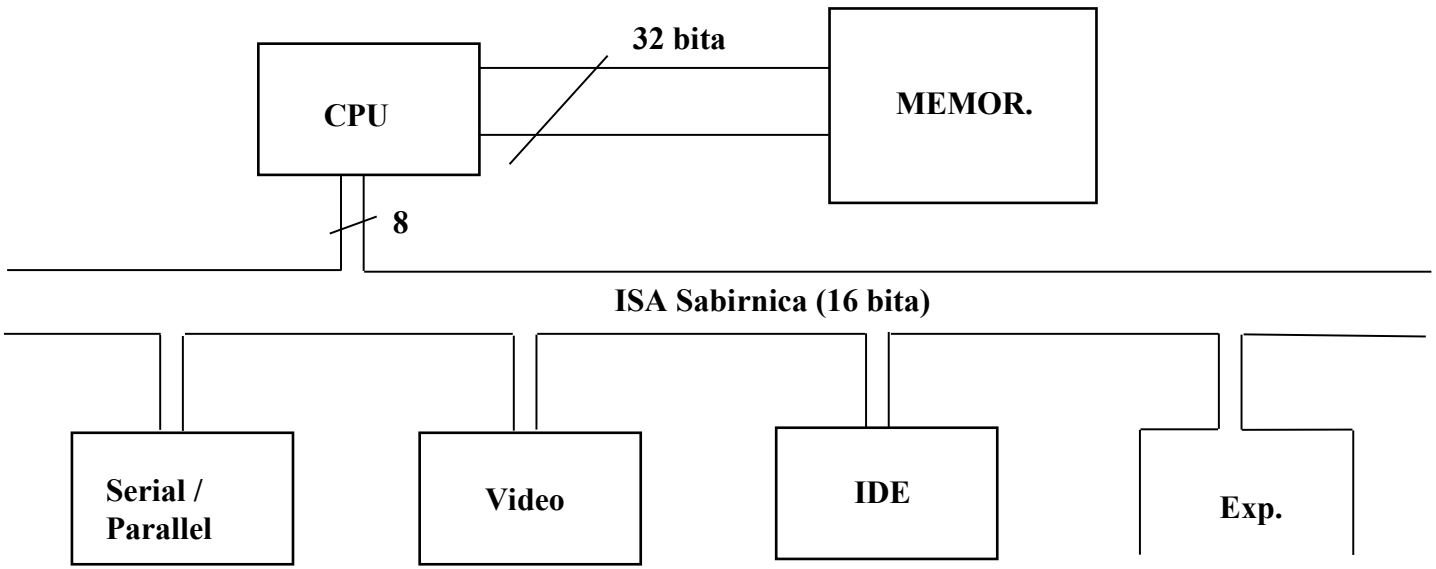
USB male brzine: 1.5 Mbita/s (USB 1.0)
USB srednje brzine: 12 Mbita/s (USB 1.0)
USB velike brzine: 480 Mbita/s (USB 2.0)

SCSI - Small Computer System Interface

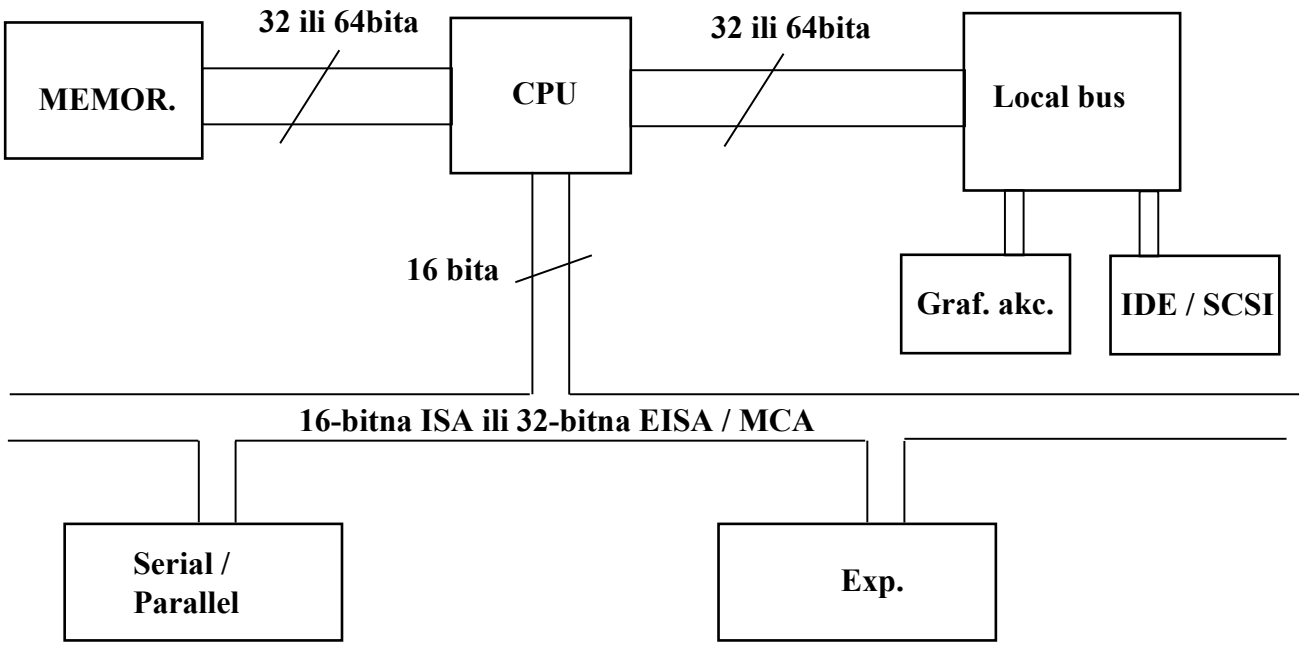
Neke značajke SCSI

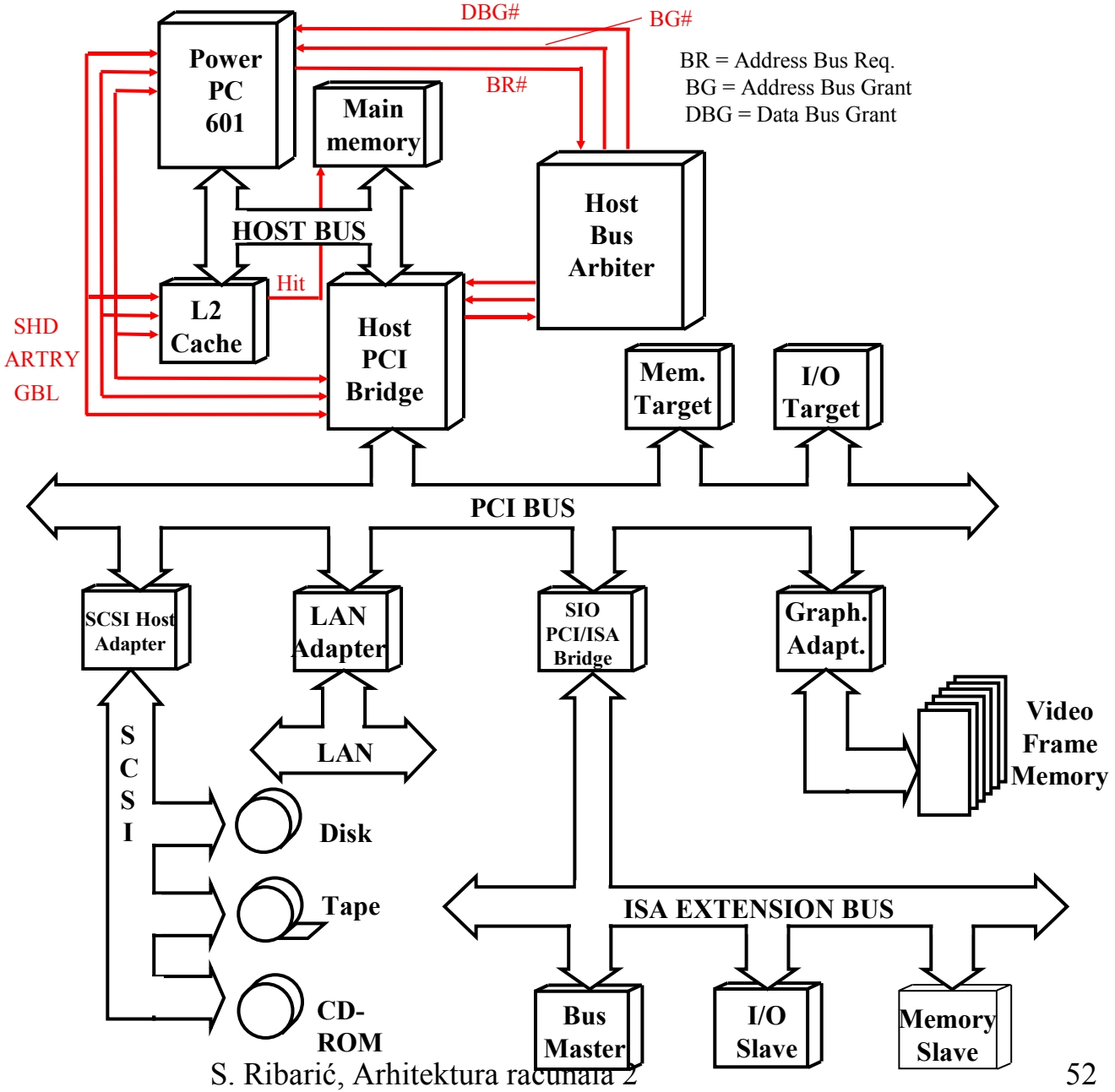
Naziv	Broj bitova	Frekvencija sabirnice(MHz)	MB/s
SCSI-1	8	5	5
Fast SCSI	8	10	10
Wide Fast SCSI	16	10	20
Ultra SCSI	8	20	20
Wide Ultra SCSI	16	20	40
Ultra2 SCSI	8	40	40
Wide Ultra2 SCSI	16	40	80

PC - 1986. godina



PC – 1993. godine

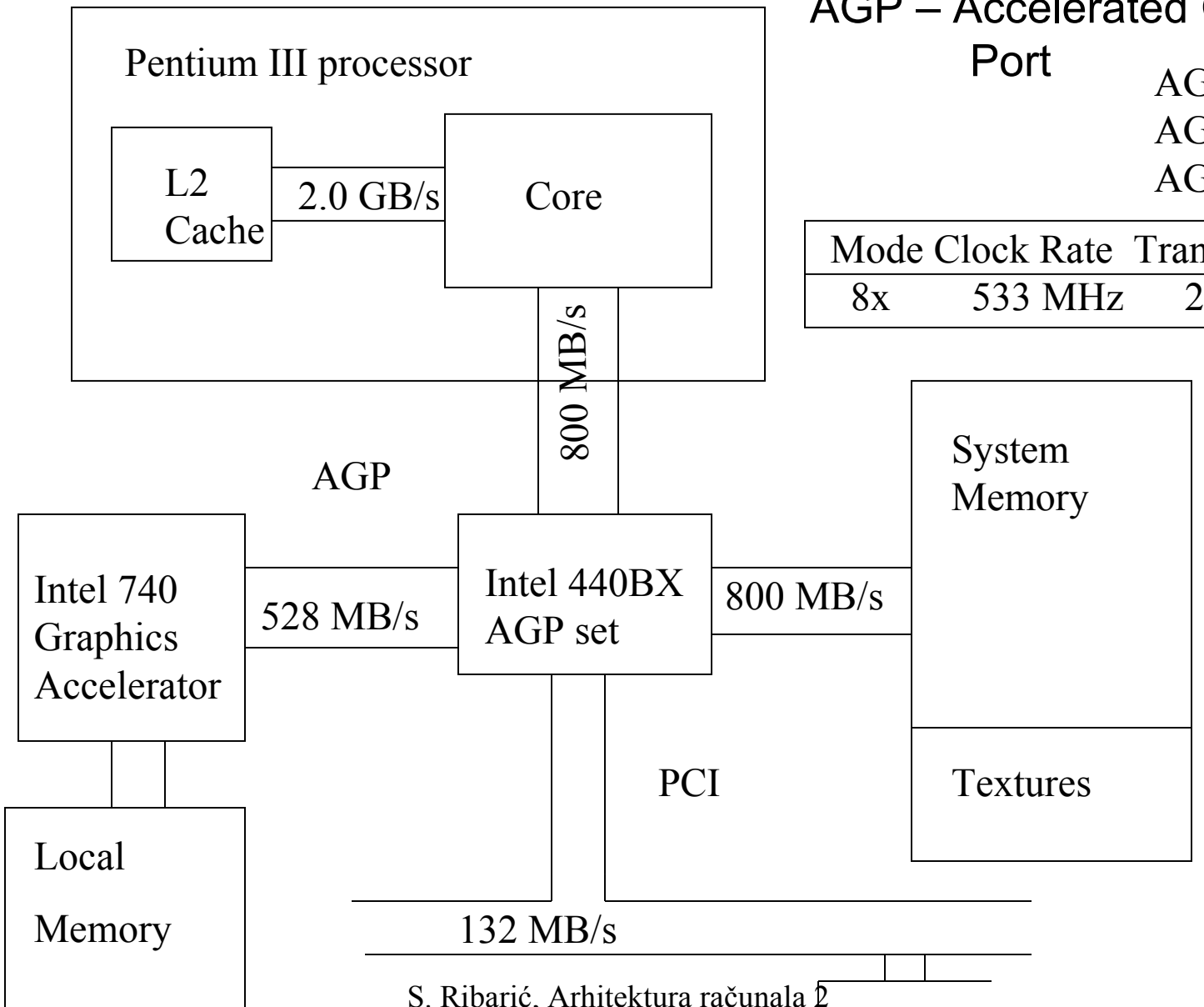



DBG#
BG#
BR#
BR = Address Bus Req.
BG = Address Bus Grant
DBG = Data Bus Grant
Power PC 601
Main memory
HOST BUS
Host Bus Arbiter
Hit
L2 Cache
Host PCI Bridge
SHD
ARTRY
GBL
Mem. Target
I/O Target
PCI BUS
SCSI Host Adapter
LAN Adapter
SIO PCI/ISA Bridge
Graph. Adapt.
Video Frame Memory
LAN
S C S I
Disk
Tape
CD-ROM
ISA EXTENSION BUS
Bus Master
I/O Slave
Memory Slave

AGP – Accelerated Graphic Port

AGP 1.0
AGP 2.0
AGP Pro

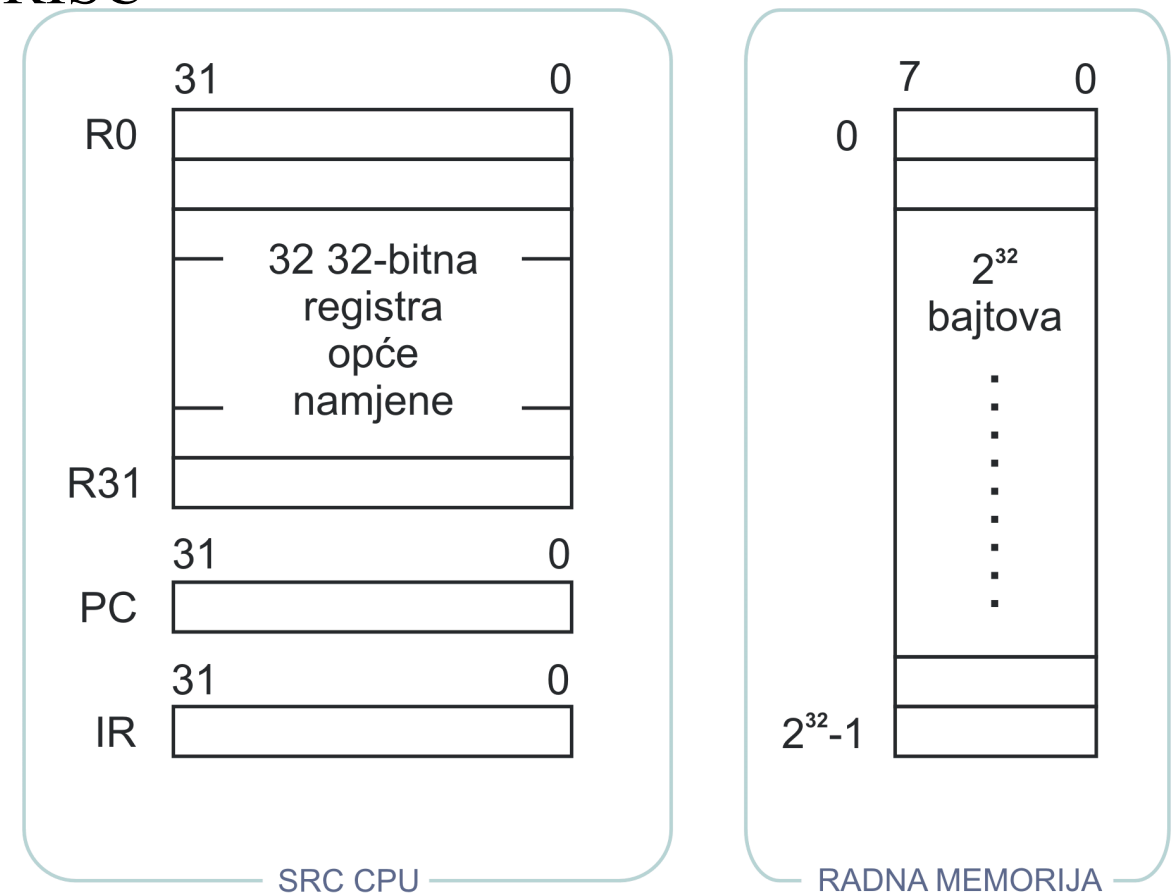
Mode	Clock Rate	Transfer Rate
8x	533 MHz	2.133 GB/s



3.5. Komponente modela (S)RISC

SRISC – Simple RISC

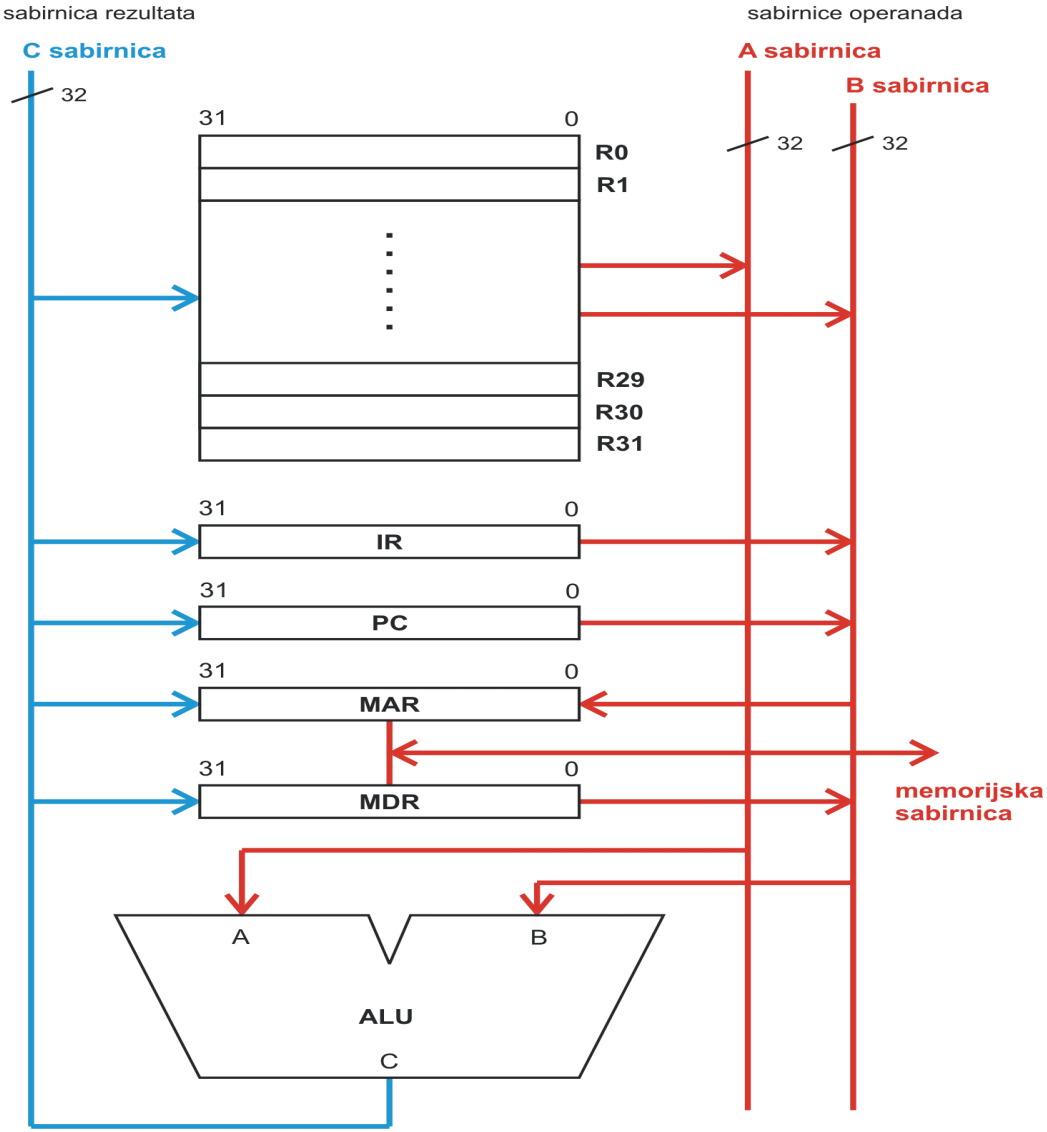
- Programski model



Značajke procesora:

- Memorija bajtno organizirana – dohvat samo 32-bitne riječi
- *load/store* arhitektura
- Big – Endian Byte Ordering

Trosabirnički SRISC



3.6. ISA-model procesora (S)RISC

ISA – Instruction Set Architecture

SRISC ima osam različitih formata instrukcija:

1. format

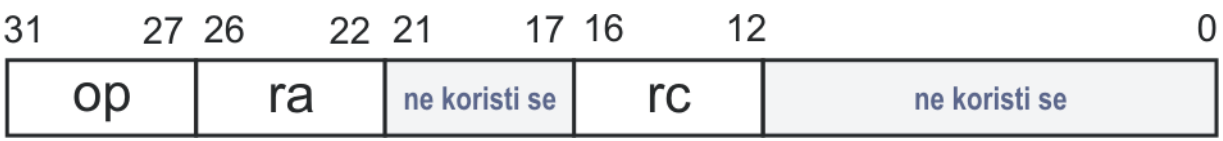
Instrukcije: *ld, st, la, addi, andi, ori*

2. format

Instrukcije: *ldr, str, lar*

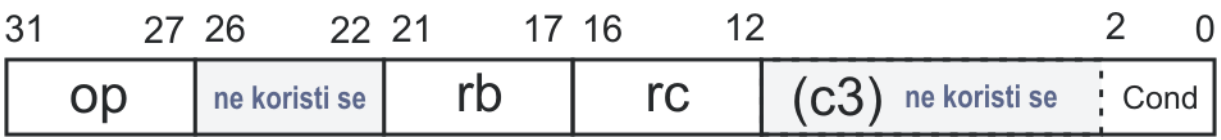
3. format

Instrukcije: *neg, not*



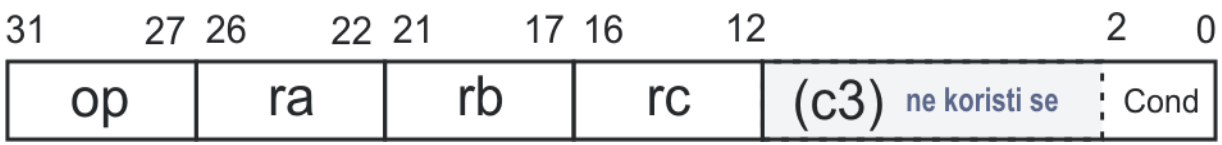
4. format

Instrukcija: *br*



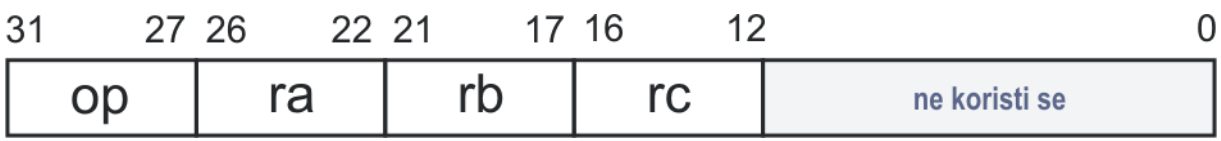
5. format

Instrukcija: *brl*



6. format

Instrukcije: *add, sub, and, or*



Troadresni format instrukcije!

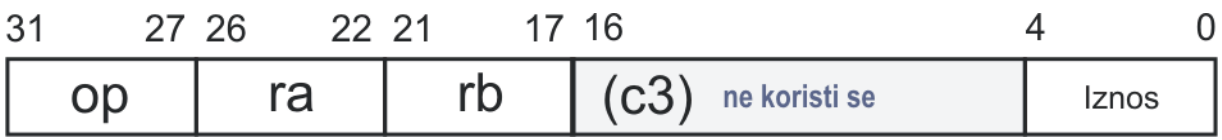
7. format

Instrukcije: *nop, stop*

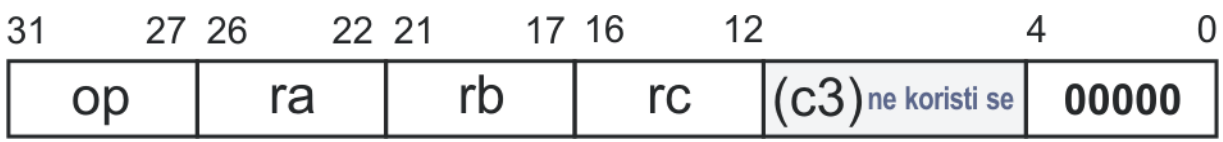
8. format

Instrukcije: *shr, shl*

a)



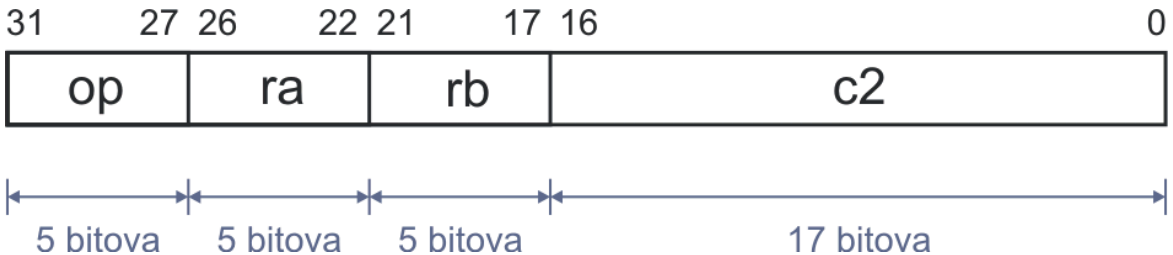
b)



Instrukcije za pristup memoriji (load i store tip instrukcija)

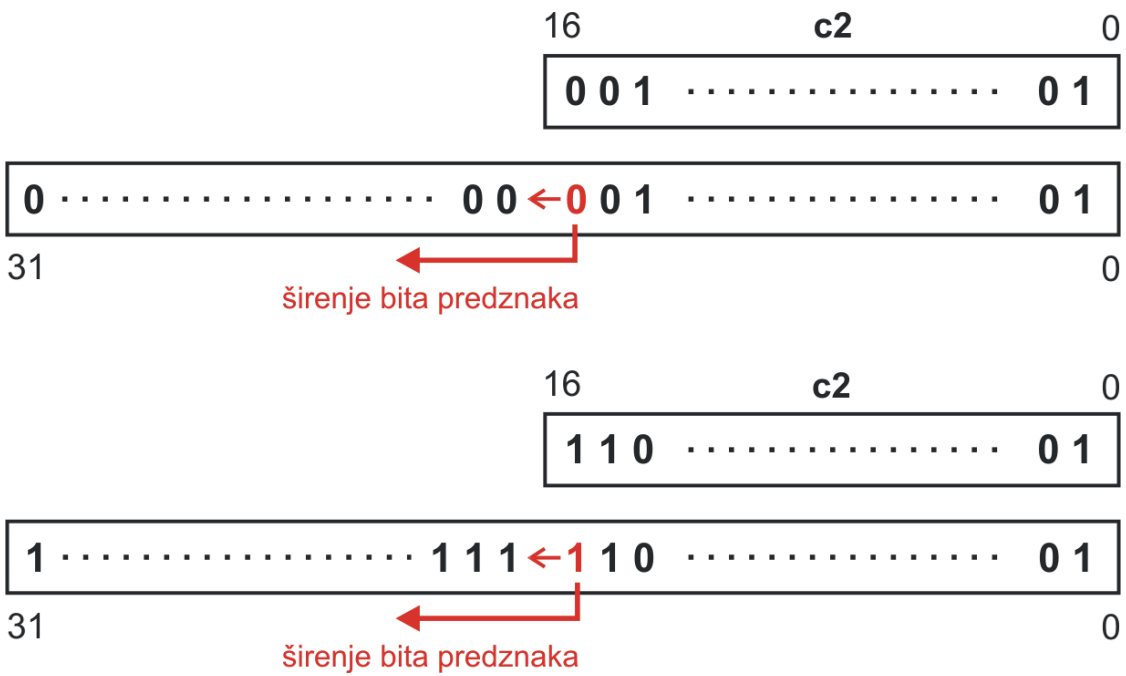
- *load* i *store* **jedine** su instrukcije za dohvat i pohranu operanada iz/u memorije:
  4 *load* instrukcije: *ld, ldr, la, lar*
  2 *store* instrukcije: *st, str*

load: /mnemonik ld/



- puni registar određen 5-bitnim poljem *ra* (1. format instrukcije)
- adresa izvorišta određena je 17-bitnom vrijednosti u polju *c2*
- polje *rb* ima dvostruku ulogu:
  1. Ako je *rb* = 0 (00000_B) tada (*rb* = 0) služi kao "signal" upravljačkoj jedinci da je adresa memorijske lokacije (izvorišta) vrijednost *c2* koja se pretvara u 32-bitnu i to **širenjem bita predznaka (engl. sign-extended)**

Širenje bita predznaka



2. Ako je *rb* $\neq$ *0* tada se adresa memorije oblikuje kao *c2* + R[*rb*] (based, displacement addressing mode)

POZOR: zbrajanje *c2* + R[*rb*] obavlja se TIJEKOM izvođenja instrukcije (engl. run time)

Instrukcije koje koriste 1. format instrukcije:

ld ra, c2 ; izravno (direktno) adresiranje: R[*ra*] = M[*c2*]

ld ra, c2(rb) ; indeksno adresiranje (*rb* $\neq$ *0*): R[*ra*] = M[*c2* +R[*rb*]]

store: /mnemonik st/

Instrukcija *st* obavlja “obrnutu” (u odnosu na *ld*) operaciju:

1. Ako *rb* = 0 (00000B) tada se R[*ra*] pohranjuje na memorijskoj lokaciji određenu adresom *c2*

2. Ako je *rb* $\neq$ *0* tada se R[*ra*] pohranjuje na memorijskoj lokaciji s adresom *c2* + R[*rb*]

Instrukcije koje koriste 1. format instrukcije:

st ra, c2 ; izravno adresiranje M[*c2*] = R[*ra*]

st ra, c2(rb) ; indeksno adresiranje (*rb* $\neq$ 0): M[*c2* + R[*rb*]] = R[*ra*]

Instrukcije koje koriste 1. format instrukcije (nastavak):

la – load address

računa adresu operanda (kao u prethodnim slučajevima ali umjesto dohvata operanda pohranjuje se izračunata adresa u R[*ra*]

la ra, c2 ; napuni R[*ra*] s adresnim pomaknućem: R[*ra*] = *c2*

la ra, c2(rb) ; napuni R[*ra*] s adresnim pomaknućem: R[*ra*] = *c2* + R[*rb*]

la se koristi za sintezu složenijih načina adresiranja!

Zadatak 5.

Na slici je prikazan format instrukcije *la* (*la – load address*) za procesor SRISC.
Odredite promjene sadržaja registara procesora za instrukciju

la r7,$100FF

Isto to napravite za instrukciju

la r7,$100FF(r4).

Napomena: Početni sadržaj registra `R[r4]` neka je 12_H.

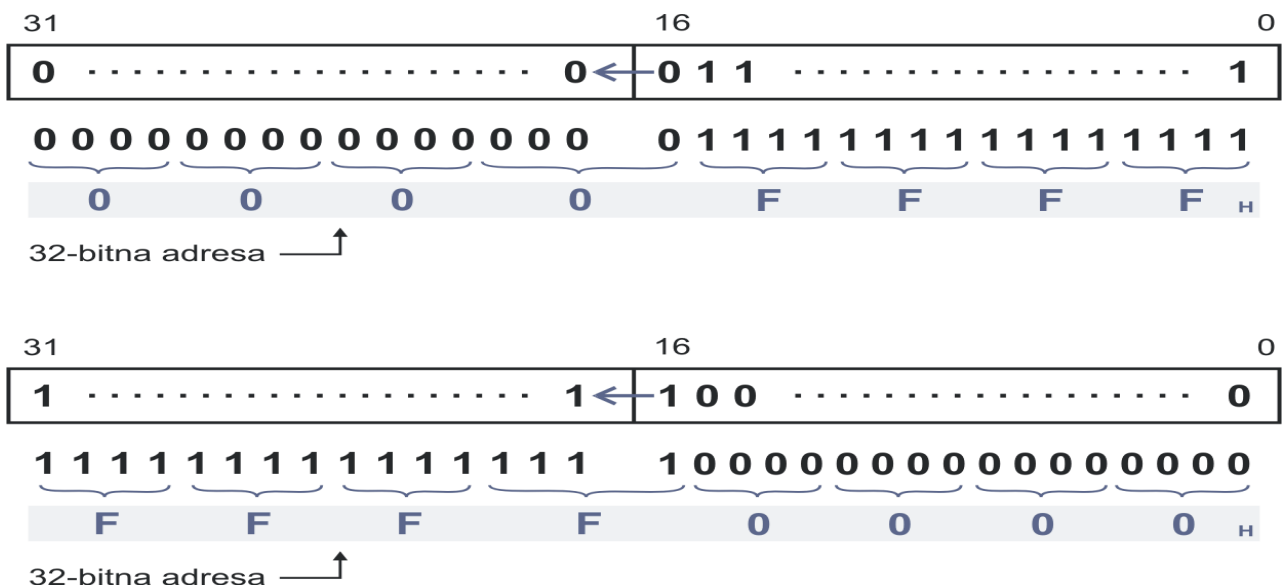
Ograničenje izravnog adresiranja

Pazi: *c2* je 17-bitna konstanta

Operandi se nalaze:

1) Na prvih 2^{16} bajtova:

2) Na posljednjih 216 bajtova:



Adresni prostori:

00000000 - 0000FFFF	1. prostor
FFFF0000 – FFFFFFF	2. prostor

Kako ostvariti pristup operandima drugdje u memoriji?

Za pristup operandima drugdje u memorijskom prostoru:

Adresiranje pomaknućem: R[*rb*] – baza
c2 – pomaknuće

efektivna adresa = R[*rb*] + *c2*

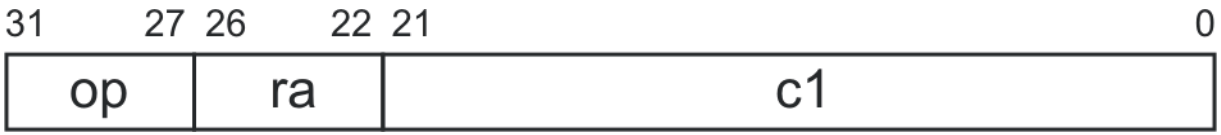
Indirektno adresiranje: uporaba R[*rb*] pri *c2* = 0

efektivna adresa = R[*rb*] + 0

POZOR: U postupku računanja (zbrajanja) adrese prvo se 17-bitno pomaknuće treba pretvoriti u 32-bitni broj

Instrukcije: *ldr, str, lar*

Koriste 2. format instrukcije



ldr ra, c1 ; napuni registar R[*ra*] = M [PC + *c1*]
; relativni način adresiranja

str ra, c1 ; pohrani M [PC + *c1*] = R[*ra*]

lar ra, c1 ; napuni relativnu adresu: R[*ra*] = PC + *C1*

POZOR: Efektivna se adresa
oblikuje tijekom izvođenja instrukcije (run-time addition)

Premjestive instrukcije (engl. Relocatable instructions)

- relativne adrese (u odnosu na PC) čine instrukcije premjestivim – cijeli se programski moduli mogu premještati bez mijenjanja vrijednosti pomaknuća

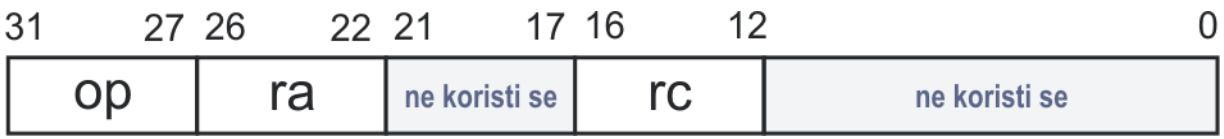
- *c1* je duljine 22 bita – može se specificirati adresa u prostoru od $\pm 2^{21}$ u odnosu na tekuću instrukciju

Primjeri instrukcija *load* i *store*

Instrukcija	op	ra	rb	c1	Komentar	Način adresiranja
`ld r1,32`	1	1	0	32	`R[1]←M[32]`	Izravno
`ld r22,24(r4)`	1	22	4	24	`R[22]←M[24+R[4]]`	Pomaknuće
`st r4,0(r9)`	3	4	9	0	`M[R[9]]←R[4]`	Reg.Indirektno
`la r7,32`	5	7	0	32	`R[7]←32`	Usputno
`ldr r12,-48`	2	12	-	-48	`R[12]←M[PC-48]`	Relativno
`lar r3,0`	6	3	-	0	`R[3]←PC`	Registar

- Instrukcije s jednim operandom

- Koriste 3. format instrukcije:

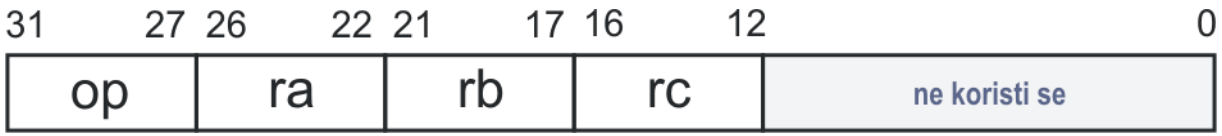


neg ra, rc ; R[*ra*] = - R[*rc*] / dvojni komplement/

not ra, rc ; R[*ra*] = $\overline{\text{R[}rc\text{]}}$ /jedinični komplement/

- ALU instrukcije (binarne operacije)

- koriste 6. format instrukcije (troadresni!):



add ra, rb, rc ; R[*ra*] = R[*rb*] + R[*rc*]

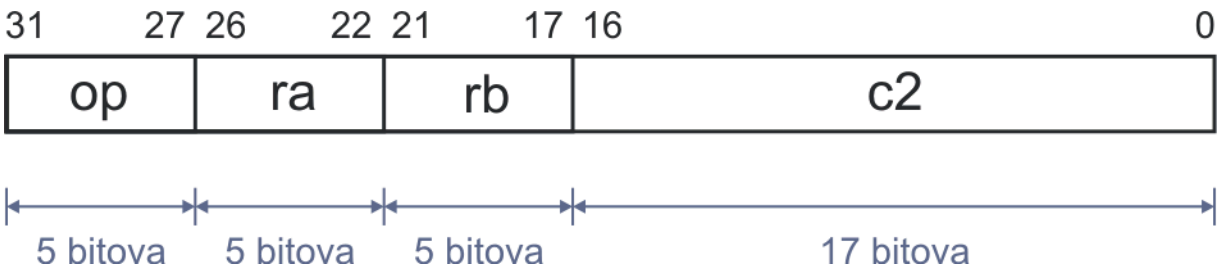
sub ra, rb, rc ; R[*ra*] = R[*rb*] - R[*rc*]

and ra, rb, rc ; R[*ra*] = R[*rb*] $\wedge$ R[*rc*]

or ra, rb, rc ; R[*ra*] = R[*rb*] $\vee$ R[*rc*]

• ALU instrukcije (usputno adresiranje)

- koriste 1. format instrukcije:



addi ra, rb, c2 ; R[*ra*] = R[*rb*] + *c2* : *c2 – usputna konstanta*

andi ra, rb, c2 ; R[*ra*] = R[*rb*] $\wedge$ *c2*

ori ra, rb, c2 ; R[*ra*] = R[*rb*] $\vee$ *c2*

Zadatak 6.

Za instrukciju

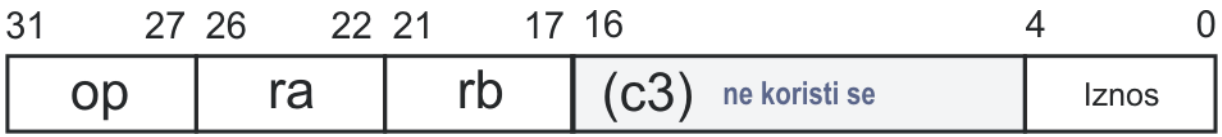
addi r3, r4, ,$100F0

odredite sadržaje registara *r3* i *r4* nakon njena izvođenja.
Sadržaj registra *r4* je $0010000F

Pozor: 17-bitna konstanta koja se nalazi u polju konstante c2 se operacijom širenja bita predznaka pretvara u 32-bitnu vrijednost prije negoli se obavi aritmetička operacija.

• Posmačne instrukcije

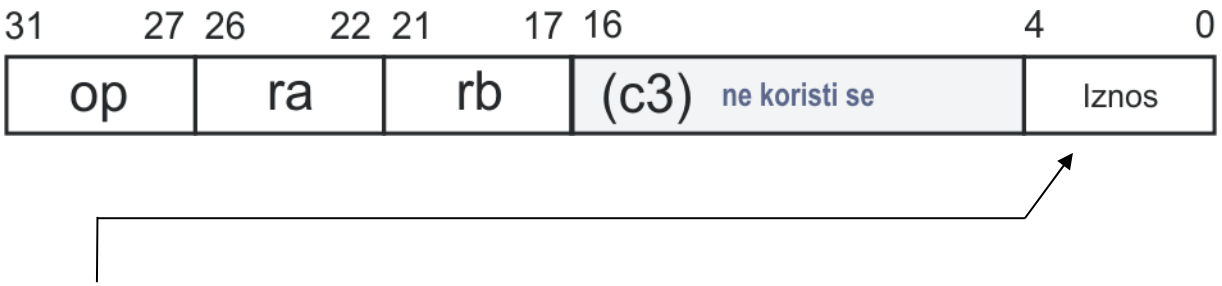
- koriste 8a) format instrukcije:



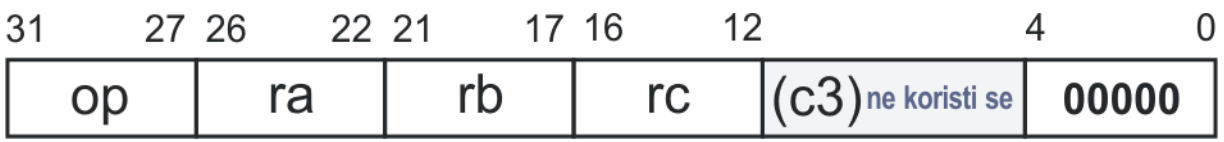
Instrukcije posmiču operand iz R[*rb*] (udesno, ulijevo ili kružno) i smještavaju rezultat u R[*ra*].

Broj posmaknutih mjesta određen je 5-bitnim nepredznačenim brojem

5-bitna vrijednost ⟶ iznos posmaka od 0 do 31



Ako je ta vrijednost 0 onda se kao iznos posmaka uzima 5 LSb registra R[*rc*] (format 8b))

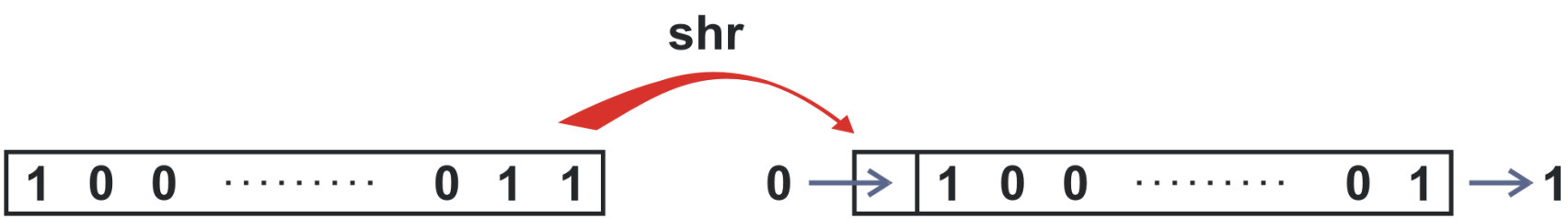


- Razlikujemo dvije vrste posmaka:

1) logički posmak
2) aritmetički posmak

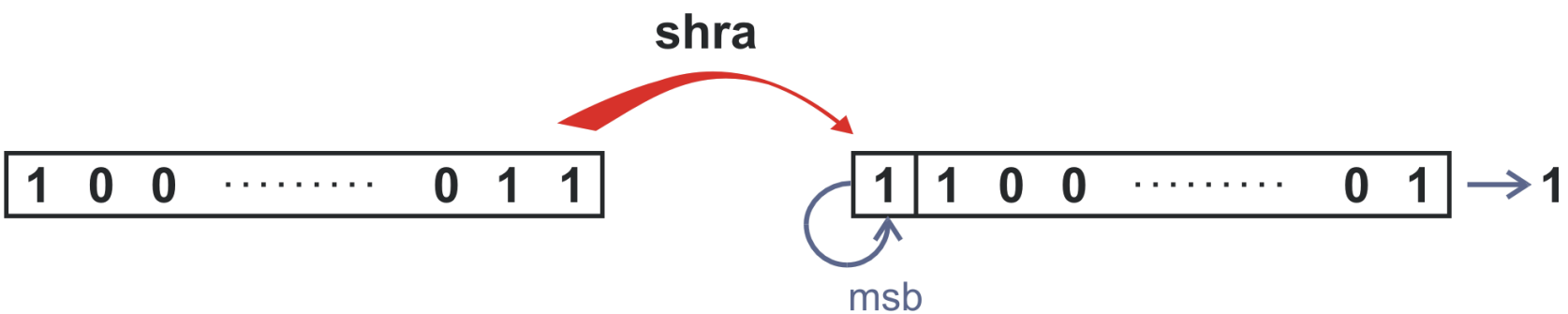
Primjer:

shr logički posmak udesno



Primjer:

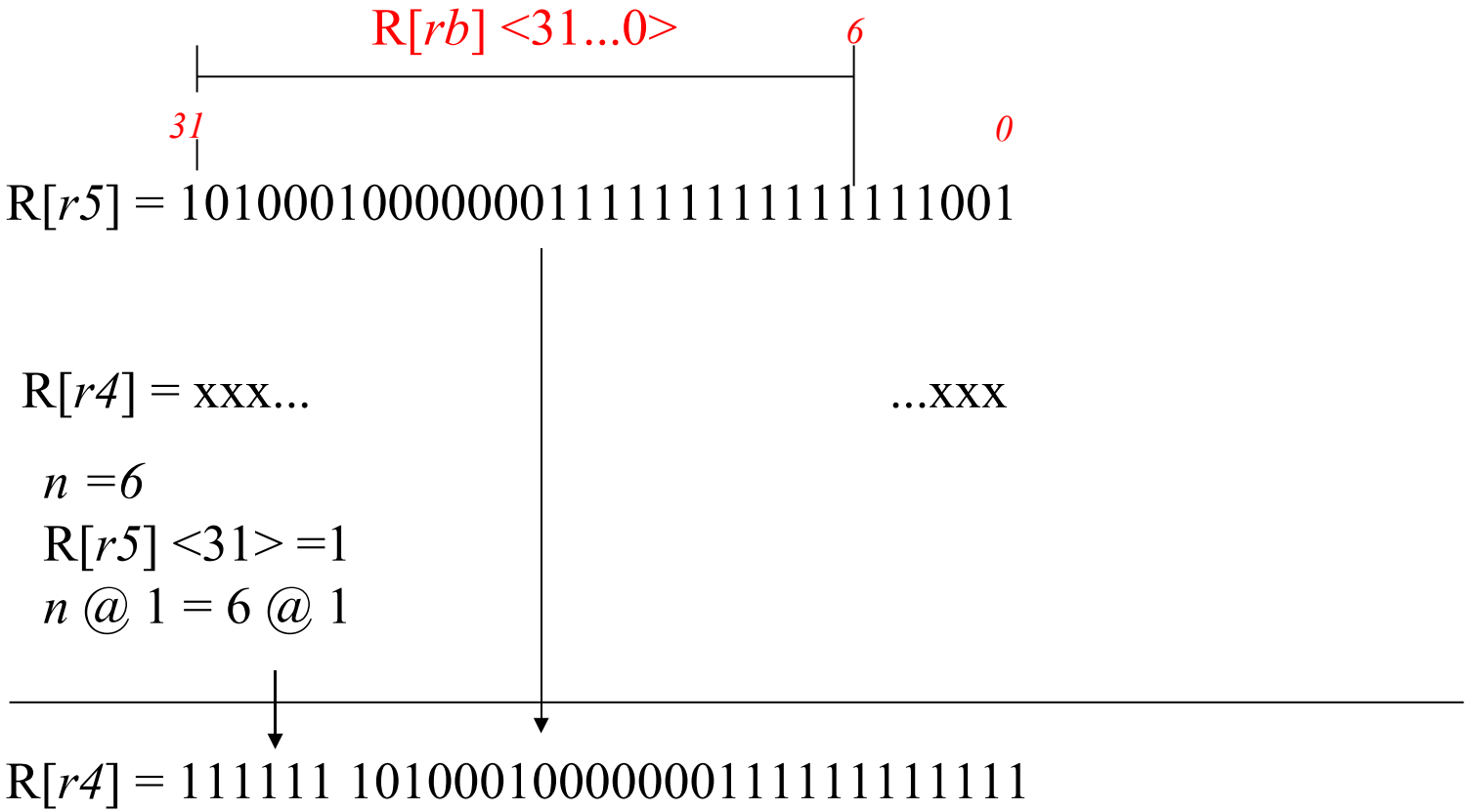
shra aritmetički posmak udesno



Ohranjuje se bit predznaka

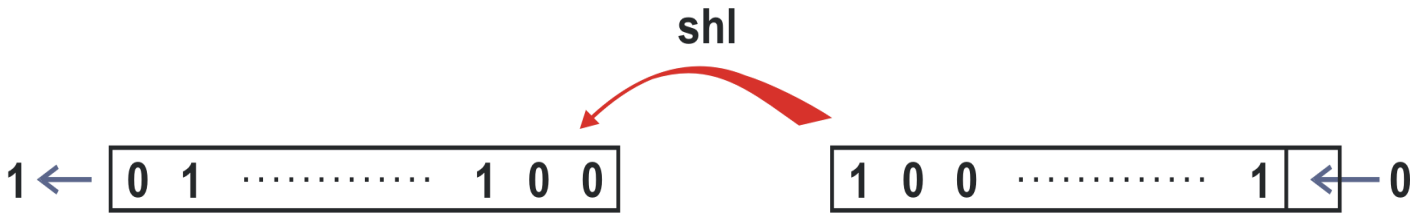
shra (:=op = 27) ⟶ R[*ra*] <31...0> ⟵ (*n*@R[*rb*] <31>) # R[*rb*]<31...n>

shra r4, r5, 6



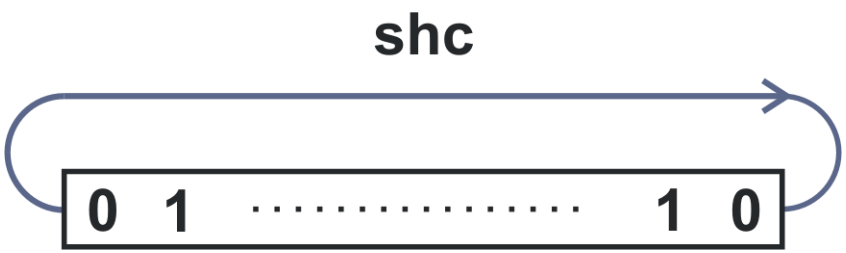
Posmak ulijevo

shl



Kružni posmak

shc



Posmačne instrukcije – primjeri:

shr ra, rb, rc ; posmakni R[*rb*] (za broj mjesta koji je određen s R[*rc*]) i pohrani ga u R[*ra*]

shr ra, rb, count ; posmakni R[*ra*] udesno i pohrani ga u R[*ra*] (za broj mjesta određen u 0-4 polju c3)

shra ra, rb, rc

shl ra, rb, rc

shl ra, rb, count

shc ra, rb, rc

shc ra, rb, count

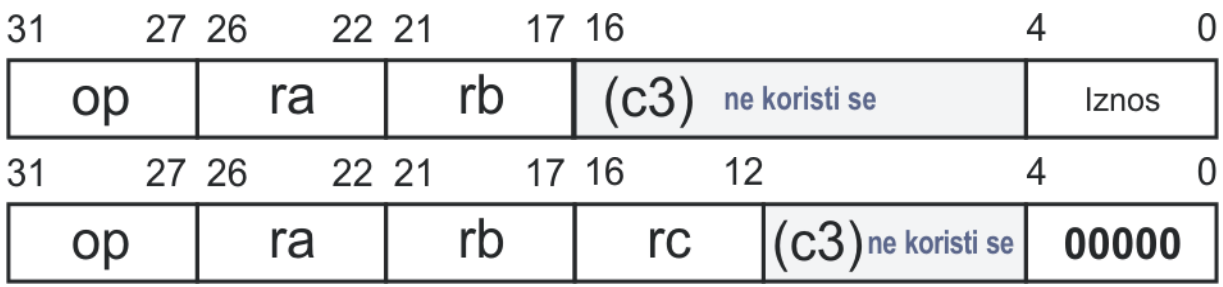
*A*ko je *count* polje u instrukciji 0 (format 8b) tada se za broj mjesta *posmaka* uzima iz registra koji je određen bitovima instrukcije na poziciji 12 - 16

Zadatak 7.

Za instrukciju

shl *r3*, *r5*, *r7*

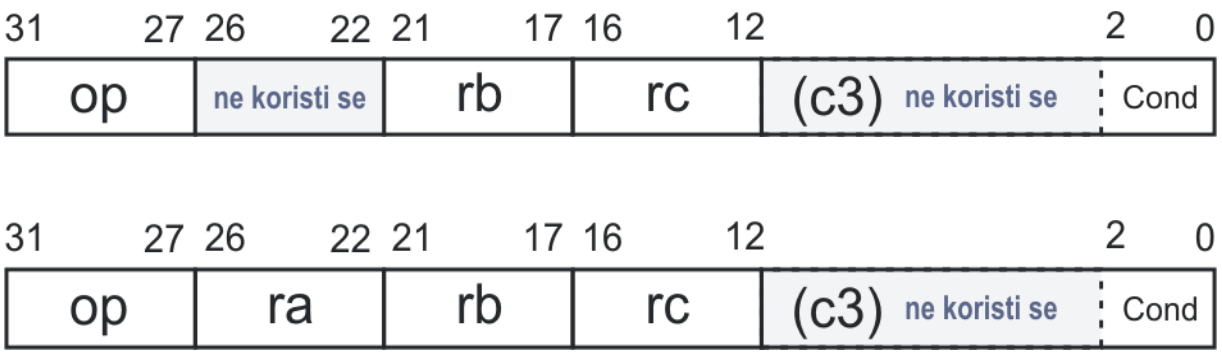
koja ima operacijski kod $28 odredite binarne sadržaje pojedinih polja u odgovarajućem formatu instrukcije (slika).



Odredite sadržaje registara nakon izvođenja instrukcije ako su inicijalni sadržaji registara bili sljedeći (r3) = $00AABBCC, (r5) = $000010A0 i (r7) = $FF00AA05.

br i *brl*

Instrukcije koriste 4. i 5. format:



Instrukcija *br* se izvodi tako da se mijenja sadržaj PC s ciljnom adresom

Instrukcija *brl* (branch&link) se izvodi tako da kopira PC u povezni (engl. linkage) registar i to prije grananja

Povezni registar dopušta povratak iz potprograma i rabi se za implementaciju procedura i funkcija u HLL

Pozor: PC se kopira u povezni registar bez obzira da li će se grananje izvesti (ili ne)

Instrukcije grananja (uvjeti!):

zbirni jezik	*c3*<2...0>	Uvjeti grananja
brnv, brlnv	*0*	*Nikada*
br, brl	*1*	*Bezuvjetno*
brzr, brlzr	*2*	*Ako je* R[*rc*] = 0
brnz, brlnz	*3*	*Ako je* R[*rc*] ≠ 0
brpl, brlpl	*4*	*Ako je* R[*rc*]<31> = 0
brmi, brlmi	*5*	*Ako je* R[*rc*] < 0

Primjer:

brl ra, rb,rc, c3 ; R[*ra*] ← PC i granaj na ciljnu
; adresu određenu s R[*rb*]
; ako sadržaj registra *rc*
; zadovoljava uvjet *c3*

ili

brlzr r7, r5, r1 ; R[*7*] ← PC i ako je R[1] = 0
; tada PC ← R[5]

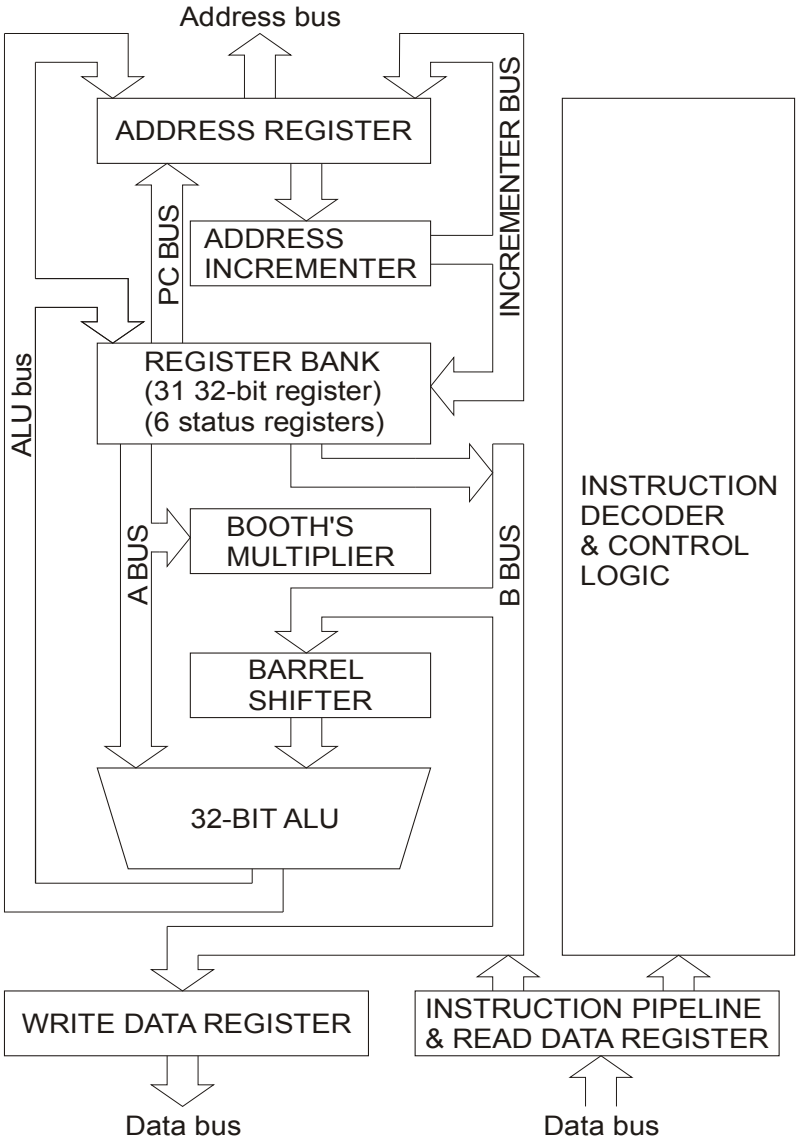
Mješovite instrukcije

nop ; ne čini ništa

stop ; zaustavi stroj

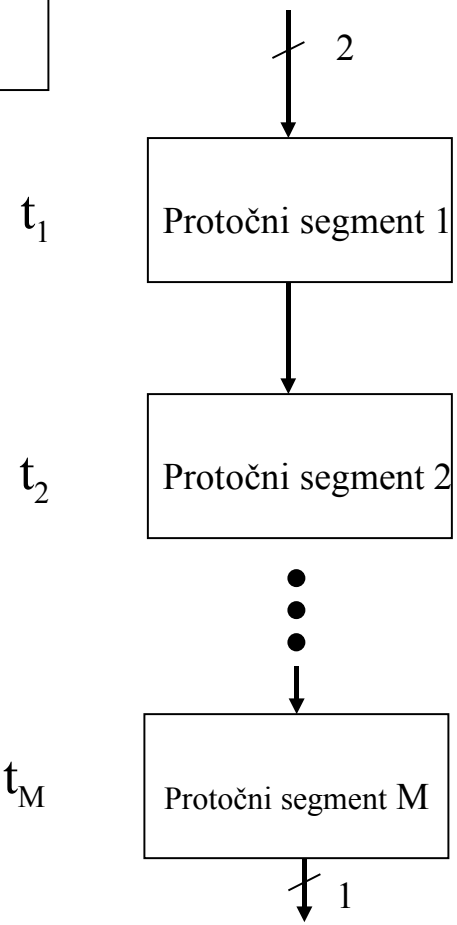
nop ??? ⟶ *važna instrukcija u protočnoj izvedbi procesora!!!*

ARM 6 procesor - Acorn RISC Machine



3.7. Model protočne strukture

Model protočne strukture dubine M



- N >> M parova operanada
- Vrijeme obrade u protočnom segmentu: $t_1 = t_2 = ... = t_M = t_S$

Prvi rezultat: $M \cdot t_S$

Svaki slijedeći rezultat nakon t_s

Vrijeme obrade N parova operanada:

$T_N = Mt_S + (N - 1)\, t_S$

Vrijeme obrade jednog para operanada:

$T_{ef} = T_N / N$

$T_{ef} = (Mt_S + (N - 1)\, t_S) / N$

Vrijeme obrade para operanda:

$$T_{ef} = (Mt_S + (N - 1)\, t_S) / N$$

$$N >> M;\ N \rightarrow \infty$$

$$T_{\infty} = \lim_{N \to \infty} ((Mt_S + (N - 1)\, t_S) / N)$$

Vrijeme obrade jednog para operanada:

$$T_{\infty} = t_S$$

!!!

Protočnost – značajka RISC i CISC procesora!

Zadatak 8.

Za programski odsječak:

```
#define Cost 125
if (X<0) X = -X;
```

napišite program u zbirnom jeziku (asembleru) za SRISC.

Uputa: Neka izvršni kod bude pohranjen u memoriji na početnoj adresi $5000. Za definiranje vrijednosti konstante Cost, za rezervaciju jedne riječi za varijablu X (koja će biti pohranjena na adresi $1000) te za definiranje početka izvršnog koda programa koristiti pseudo asemblerske naredbe: equ, org, dw (define word), a za ispitivanje varijable X i njenu (eventualnu) promjenu registre r0 i r1.

Rješenje:

```
                Cost    equ     125     ; def. vrijednost konstante
                        org     $1000   ; sljedeća riječ će se pohr. na
1000
                X:      dw      1       ; rezerviraj 1 riječ za varijablu X
                        org     $5000   ; početak programa
                        lar r0, Over    ; def. odredišnu adresu grananja
                        ld r1, X        ; pohrani vrijednost X u r1
                        brpl r0, r1     ; granaj na Over ako je r1>= 1
                        neg r1, r1      ; promijeni predznak varijable X
                Over:   ...
```

Zadatak 9.

Napišite SRISC programski kod za sljedeće naredbe u višem programskom jeziku:

6. if (a = = 0) a = a +1; else a = a –1
   i
8. for (i = 1; i <10; i++)
   ndigit[i] = 0;
   /pretpostavite da je ndigit[10] određeno odgovarajućom pseudo asemblerskom naredbom/

Zadatak 10.

Pročitati poglavlje “0. Računala s reduciranim skupom instrukcija” u knjizi S. Ribarić, “Arhitektura računala RISC i CISC”, Školska knjiga, Zagreb, 1996, pp. 1 – 29.