Programiranje

TEME

- ✓ Mašinski jezik
- ✓ Asemblerski jezici
- ✓ Viši programski jezici

Mašinski jezik

Mašinski jezik je kolekcija mašinskih instrukcija predstavljenih u binarnom obliku, tj. u vidu nizova 0 i 1.

- Procesor razume i izvršava samo mašinski kod.
- Programi pisani u mašinskom jeziku se ne prevode, jer su već u obliku prilagođenom datom hardveru, odnosno arhitekturi sistema.
- Programi pisani u drugim jezicima, da bi se izvršili, moraju se dovesti na nivo mašinskog koda, što se postiže raznim prevodiocima.

Mašinski jezik

- Da bi softver mogao da radi, odgovarajući mašinski kod mora da se smesti u memoriju.
- Pisanje i tumačenje mašinskog koda je teško i podložno greškama.
- Danas se ne programira u mašinskom jeziku.

Primer 1

Mašinski kod

```
00100101 11010011
00100100 11010100
10001010 01001001 11110000
01000100 01010100
01001000 10100111 10100011
11100101 10101011 00000010
00101001
11010101
11010100 10101000
10010001 01000100
```

Asemblerski jezik

Asemblerski jezik predstavlja simboličku reprezentaciju mašinskog jezika.

- Asemblerski jezik je uveden da bi se olakšalo pisanje programa.
- Asemblerski program se sastoji od niza instrukcija (iskaza) predstavljenih mnemonicima i simboličkim adresama koje čovek bolje razume i njima može lakše da upravlja.
- Asemblerski jezik je hardverski zavisan, što znači da svaki tip procesora ima svoj asemblerski jezik.

Primer 2

Asemblerski kod

```
        ST 1,[801]
        ST 0,[802]
TOP:    BEQ [802],10,BOT
        INCR [802]
        MUL [801],2,[803]
        ST [803],[801]
        JMP   TOP
BOT:    LD A,[801]
        CALL PRINT
```

Asembler

Asembler je program koji prevodi kod napisan na asemblerskom jeziku u odgovarajući mašinski kod. Svaka instrukcija u asemblerskom jeziku se prevodi u odgovarajuću mašinsku instrukciju.

Asembler:

- zamenjuje simboličke adrese numeričkim
- simboličke kodove operacije mašinskim kodovima
- rezerviše mesto za instrukcije i podatke u memoriji
- prevodi konstante u mašinski oblik...

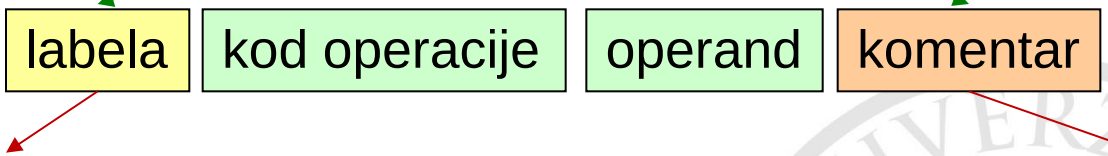
- Asemblerski jezik može da sadrži i iskaze koji se ne izvršavaju, već služe kao komande asembleru pri generisanju mašinskog koda.

Asemblerski program

Asemblerski program se sastoji od sekvence asemblerskih instrukcija.

Asemblerska instrukcija

opciona polja



- služi za simboličko imenovanje memorijskih adresa ili podataka
- identifikator koji se može koristiti za skok na liniju sa lebelom

- pisanje objašnjenja

Primer 3

```
START   LD      X       / kopiraj sadrzaj lokacije X u akumulator
        BRA     START   / idi na iskaz sa labelom START
```

Asemblerski program

Direktive ili pseudo-operacije su komande koje su razumljive asembleru i utiču na način prevođenja asemblerskog koda u mašinski.

- Direktive omogućavaju da se isti program asemblira na različite načine zavisno od parametara koje zadaje programer.
- Direktive nisu asemblerske instrukcije i ne generišu nikakav mašinski kod.

Primer 4

Rezervacija i inicijalizacija prostora za smeštaj promenljivih:

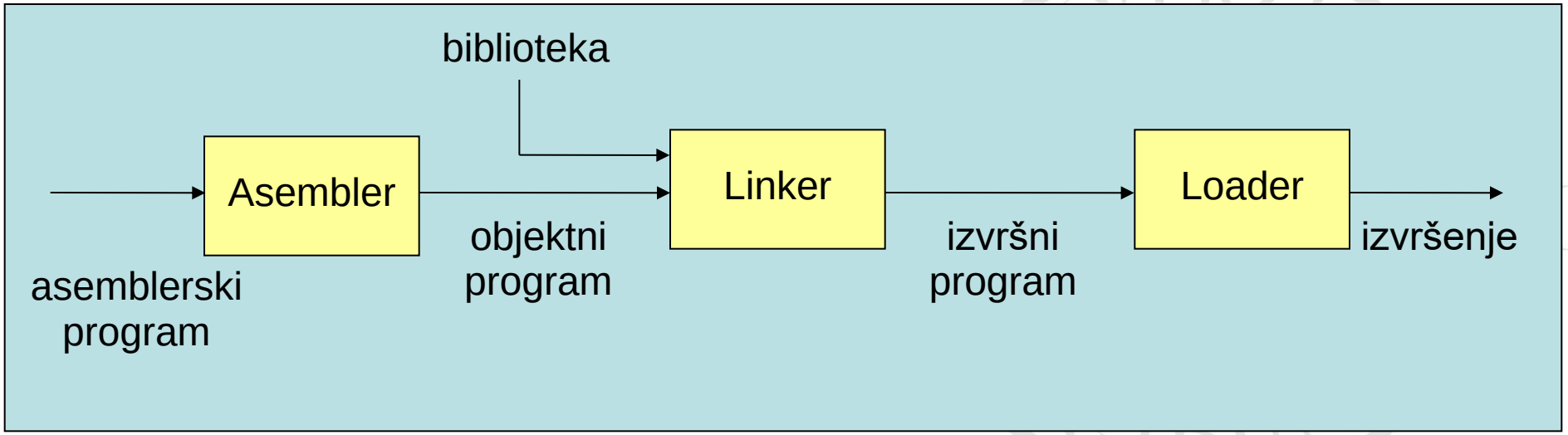
→ direktiva W rezerviše 16-bitnu reč u memoriji za lebelu X i inicijalizuje je na 120

```
X    W    120 \ rezervise rec i inicijalizuje je na 120
```

Asemblerski program

Da bi se izvršio asemblerski program, potrebni su:

- asembler – prevodi asemblerski kod u mašinski
- linker – povezuje kod sa bibliotekama i drugim programima
- punilac (*loader*) – smešta izvršni kod u memoriju i startuje izvršenje



Viši programski jezici

- Imaju viši nivo apstrakcije u odnosu na asemblerski jezik.
- Umesto sa registrima, memorijskim lokacijama, stekovima, ovi jezici rade sa varijablama, nizovima, objektima, petljama, funkcijama, potprogramima i dr.
- Primeri: *C*, *C++*, *Java*, *Visual Basic*, *Pearl*, *PHP*, *Python*, ...

Primer 5

C kod

```
class Triangle {
  ...
  float surface()
    return b*h/2;
  }
```

- Da bi se kod pisan na višem programskom jeziku preveo u izvršni kod (mašinski, objektni) koriste se prevodioci (*compilers*).
- Linkeri (*linkers*) služe da od više modula sa objektnim kodom proizvedu modul za unos u memoriju (*load module*).

Prednosti

Asemblerskog jezika

- ✓ pojašnjava proces izvršavanja instrukcija
- ✓ pokazuje kako se podaci čuvaju u memoriji
- ✓ pokazuje interakciju programa sa OS, procesorom, I/O jedinicama,...
- ✓ olakšava programiranje na višim programskim jezicima, jer programeri znaju šta se dešava u računaru

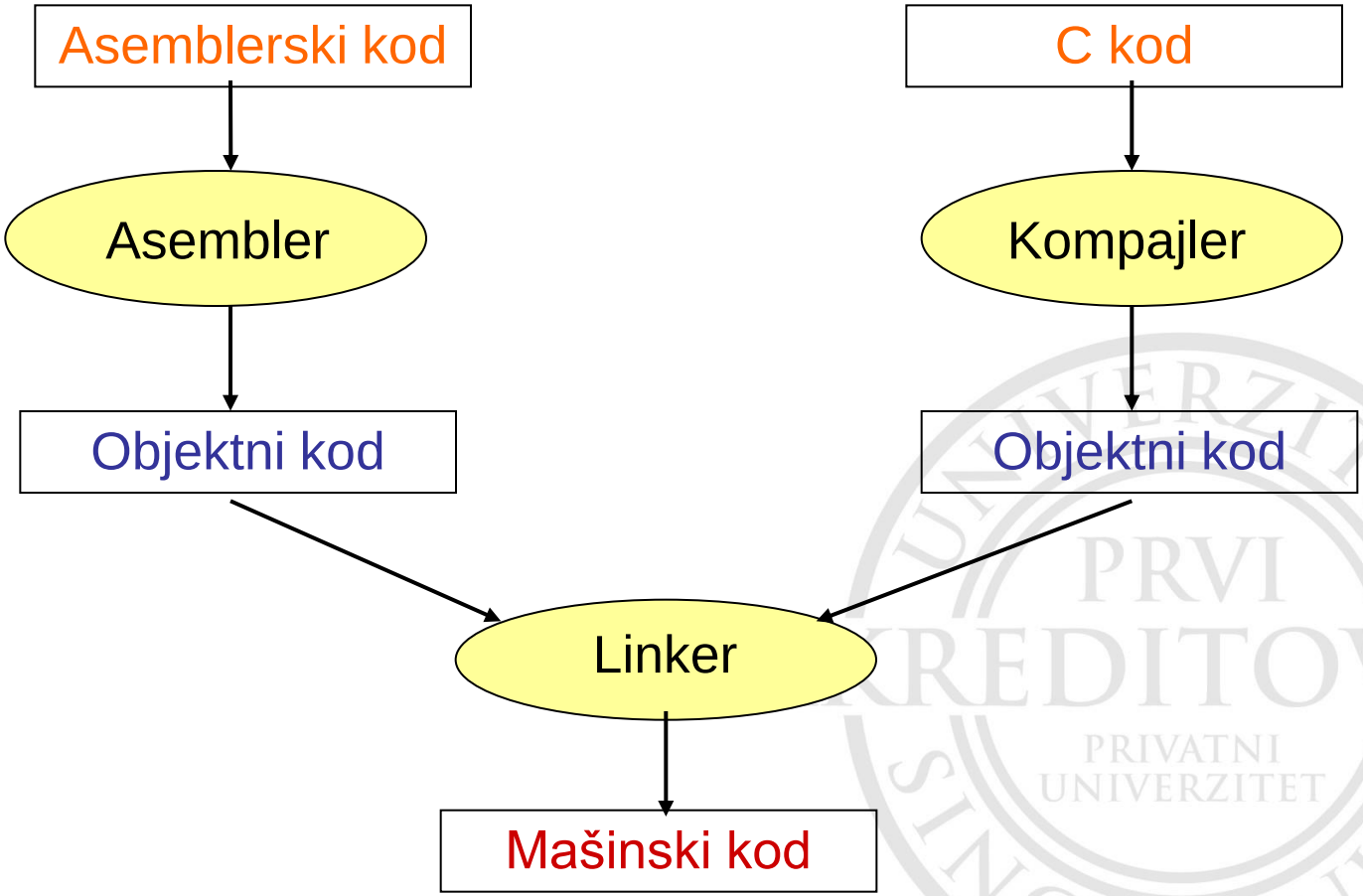
Viših programskih jezika

- ✓ imaju veću ekspresivnost i konciznost
- ✓ zahtevaju manje vremena za razvoj softvera
- ✓ omogućavaju lakše debagovanje i verifikaciju koda
- ✓ omogućavaju lakše održavanje koda
- ✓ veća je mogućnost prenosivosti koda
- ✓ pružaju veću pouzdanost i sigurnost

Primena asemblerskog jezika

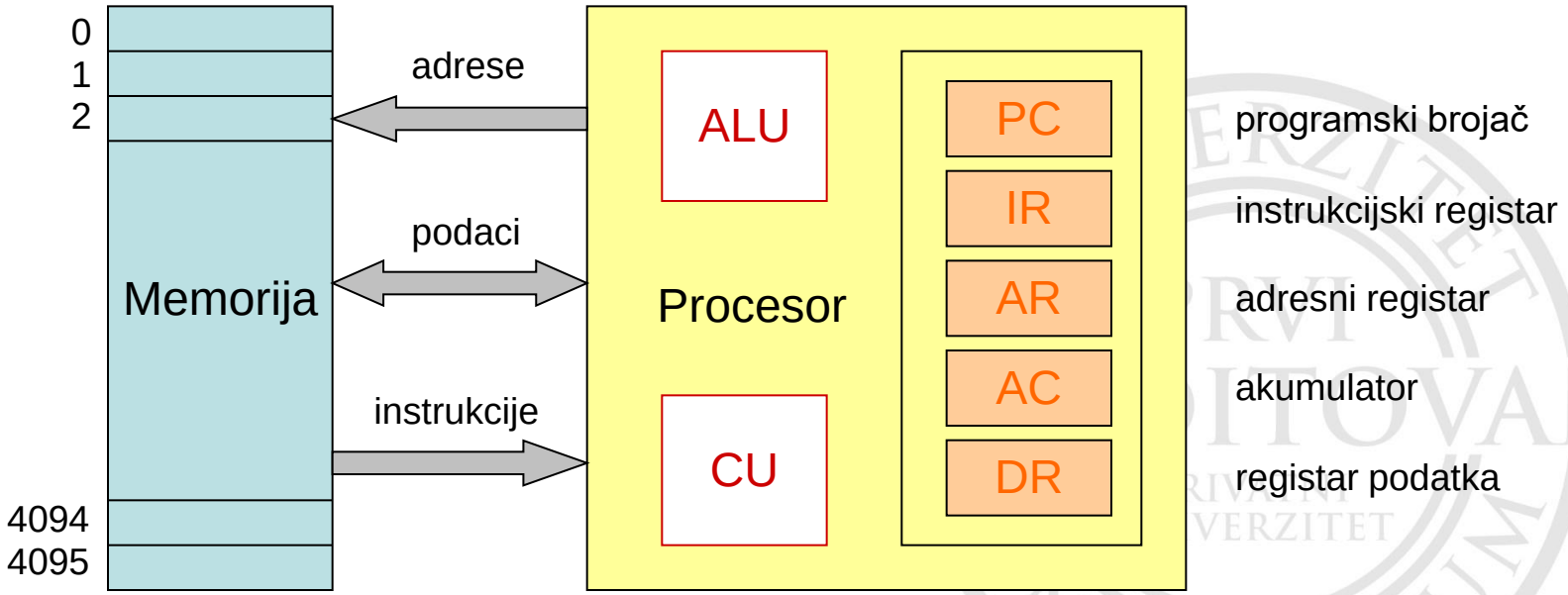
- Iako programiranje u višim programskim jezicima ima svoje prednosti, u nekim slučajevima je neohodno korišćenje asemblerskog jezika.
- Programiranje u asemblerskom jeziku može rezultovati mašinskim kodom koji je mnogo kraći i brži od koda nastalog primenom viših programskih jezika.
- Primeri primene: sistemski programi (drajveri), *embedded* sistemi, aplikacije sa vrlo ograničenim resursima,...

Generisanje izvršnog koda



Ilustrativni primer

Neka su na raspolaganju jednostavan hipotetički procesor koji ima 5 16-bitnih registara i 11 16-bitnih instrukcija u instrukcijskom setu i memorija od 4096 8-bitnih reči.



Ilustrativni primer

Instrukcijski set

Kod op.	Mnemonik	Operand	Funkcija	Tip instrukcije
0000	STOP		zaustavljanje izvršenja	
0001	LD	*adr*	M → AC	instrukcije prenosa
0010	ST	*adr*	AC → M	
0011	MOVAC		AC → DR	
0100	MOV		DR → AC	
0101	ADD		AC + DR → AC	aritmetičke i logičke instrukcije
0110	SUB		AC - DR → AC	
0111	AND		AND(AC, DR) → AC	
1000	NOT		NOT(AC) → AC	
1001	BRA	*adr*	skok na instr. na *adr*	instrukcije skoka
1010	BZ	*adr*	skok na instr. na *adr* ako AC = 0	

Ilustrativni primer

Zadatak:

a) Napisati program na mašinskom jeziku koji sadržaj mem.lok. na adresi 12 dodaje sadržaju iz mem.lok. na adresi 14 i rezultat smešta u lokaciju sa adresom 16.

b) Napisati odgovarajući program i na asemblerskom jeziku.

Inicijalne vrednosti u memoriji su:

mem.lok. sa adresom 12:	350
mem.lok. sa adresom 14:	96
mem.lok. sa adresom 16:	0

Univerzitet Singidunum
www.singidunum.ac.rs

Ilustrativni primer

Rešenje:

a) Program na mašinskom jeziku

Adresa mem.lok.	Instrukcija (program)	Opis
0000 0000 0000	0001 0000 0000 1100	M(12) → AC
0000 0000 0010	0011 0000 0000 0000	AC → DR
0000 0000 0100	0001 0000 0000 1110	M(14) → AC
0000 0000 0110	0101 0000 0000 0000	AC + DR → AC
0000 0000 1000	0010 0000 0001 0000	AC → M(16)
0000 0000 1010	0000 0000 0000 0000	kraj
0000 0000 1100	0000 0001 0101 1110	podatak 350
0000 0000 1110	0000 0000 0110 0000	podatak 96
0000 0001 0000	0000 0000 0000 0000	podatak 0

KO	Mnemonik
0000	STOP
0001	LD
0010	ST
0011	MOVAC
0100	MOV
0101	ADD
0110	SUB
0111	AND
1000	NOT
1001	BRA
1010	BZ

Ovaj program je težak za razumevanje i debagovanje.

Ilustrativni primer

Rešenje:

b) Program na asemblerskom jeziku

```
        LD X            \ AC ← X
        MOVAC           \ DR ← AC
        LD Y            \ AC ← Y
        ADD             \ AC ← AC + DR
        ST Z            \ Z← AC
        STOP
X       W       350     \ rezervise rec inicijalizovanu na 350
Y       W       96      \ rezervise rec inicijalizovanu na 96
Z       W       0       \ rezervise rec inicijalizovanu na 0
```

Ovaj program je jednostavniji za razumevanje i debagovanje.

Šta smo naučili?