Predmet: Arhitektura računara
Profesor: redovni profesor dr Dušan Regodić, dipl. inž.

Programiranje

1. MAŠINSKI JEZIK
2. ASEMBLERSKI JEZIK
3. VIŠI PROGRAMSKI JEZICI

Mašinski jezik

1. **Mašinski jezik** je kolekcija mašinskih instrukcija predstavljenih u **binarnom obliku**, tj. u vidu nizova 0 i 1.
2. Procesor može da razume i izvršava samo mašinski kod.
3. Programi pisani u mašinskom jeziku se ne prevode, jer su već u obliku prilagođenom datom hardveru, odnosno arhitekturi sistema.
4. Programi pisani u drugim jezicima, da bi se izvršili, moraju se dovesti na nivo mašinskog koda, što se postiže raznim prevodiocima.
5. Da bi softver mogao da radi, odgovarajući mašinski kod mora da se smesti u memoriju.
6. Pisanje i tumačenje mašinskog koda je teško i podložno greškama.
7. Danas se retko programira u mašinskom jeziku.

Asemblerski jezik

1. Da bi se olakšalo pisanje programa, uveden je **asemblerski jezik** koji predstavlja **simboličku reprezentaciju mašinskog** jezika.

2. Asemblerski program se sastoji od niza instrukcija (iskaza) predstavljenih **mnemonicima i simboličkim adresama** koje čovek **bolje razume i njima može lakše da upravlja**.

3. Asemblerski jezik je **hardverski zavisan, tj. svaki tip procesora ima svoj asemblerski jezik.**

Asembler

Asembler je program koji prevodi kod pisan na asemblerskom jeziku u odgovarajući mašinski kod, tj. svaka instrukcija u asemblerskom jeziku se prevodi u 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.

Viši programski jezici

- Imaju **viši nivo apstrakcije** u odnosu na asemblerski jezik.

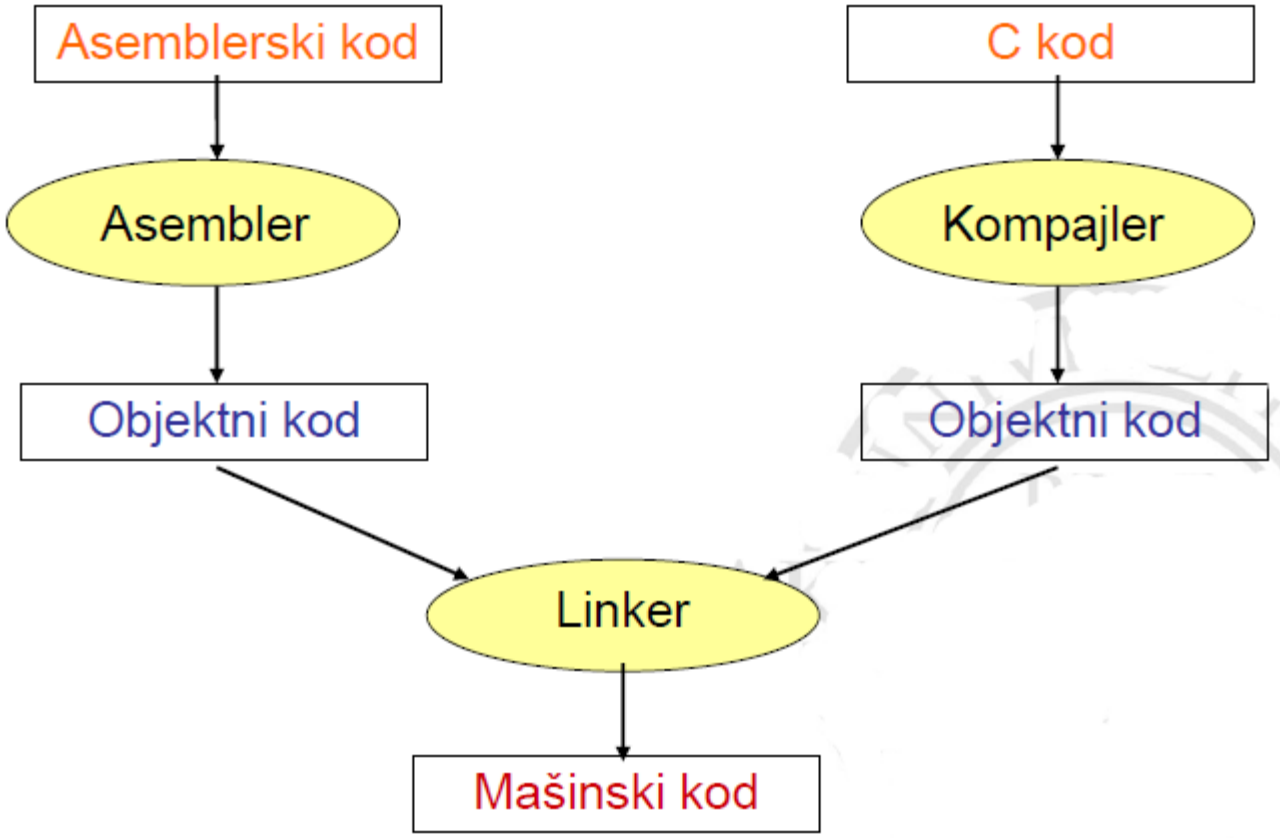
- Umesto sa registrima, memorijskim lokacijama, stekovima, ovi jezici rade sa varijablama, nizovima, objektima, složenim aritmetičkim i logičkim izrazima, petljama, funkcijama, potprogramima i dr.

- **Primeri:** *C*, *C++*, *Java*, *Visual Basic*, *Pearl*, *PHP*, *Python*, ...

- Da bi se kod pisan na višem programskom jeziku preveo u izvršni, mašinski, tj. objektni kod koriste se prevodioci (*compilers*).

- **Linkeri (*linkers*)** služe da od više modula sa objektnim kodom proizvedu modul za unos u memoriju (*load module*).

Generisanje izvršnog koda



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 (kompajleri, drajveri), *embedded* sistemi, prenosive aplikacije, aplikacije sa vrlo ograničenim resursima,...

Prednosti asemblerskog jezika

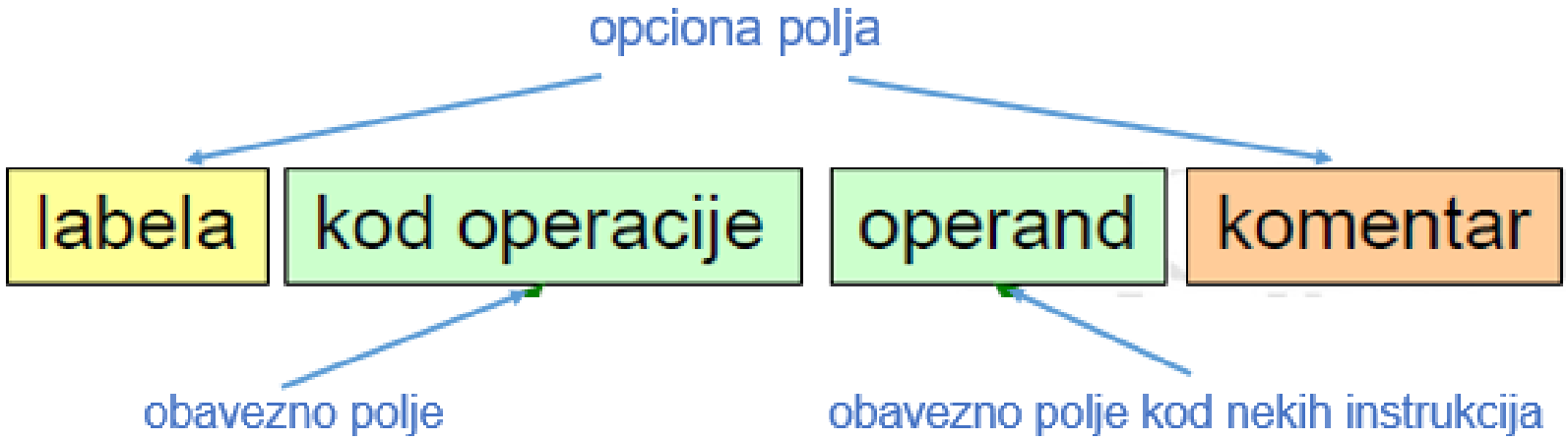
- pojašnjava proces izvršavanja instrukcija
- pokazuje kako se podaci čuvaju u memoriji
- pokazuje interakciju programa sa OS, procesorom, I/O jedinicama,...
- olakšava programiranje na višim programskim jezicima, jer programeri znaju šta se dešava u računaru

Prednosti 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

Asemblerski program (1)

- **asemblerski program** se sastoji od sekvence asemblerskih iskaza
- svaki iskaz se piše u posebnoj liniji
- svaka linija sadrži četiri polja



Asemblerski program (2)

- **labela**
  - služi za simboličko imenovanje memorijskih adresa ili podataka
  - identifikator koji se može koristiti u drugim linijama koda za skok na liniju sa lebelom
- **kod operacije**
  - skraćenica za operaciju
- **operand**
  - sadrži podatak ili dodatnu informaciju za kod operacije
- **komentar**
  - služi za pisanje objašnjenja

Primer:

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

Asemblerski program (3)

- **Direktive ili pseudo-operacije** su komande koje su razumljive asembleru i utiču na njegov način rada prilikom 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 upotrebe:** za rezervaciju i inicijalizaciju prostora za smeštaj promenljivih ili za kontrolu smeštaja programskog koda

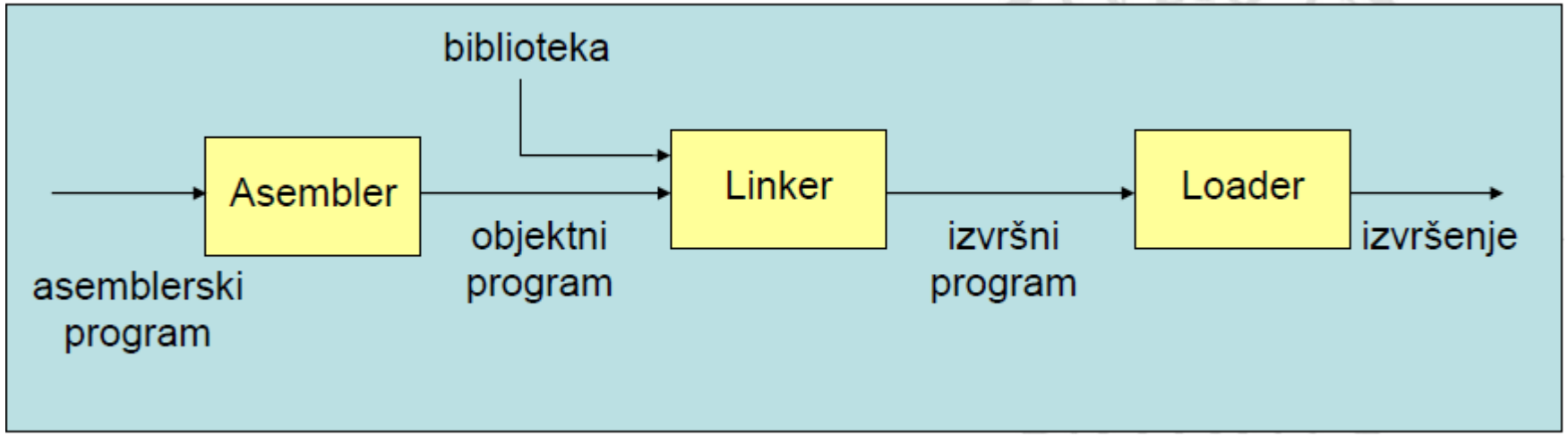
→ direktiva W rezerviše 16-bitnu reč u memoriji za lebelu X i inicijalizuje je na 120

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

Asemblerski program (4)

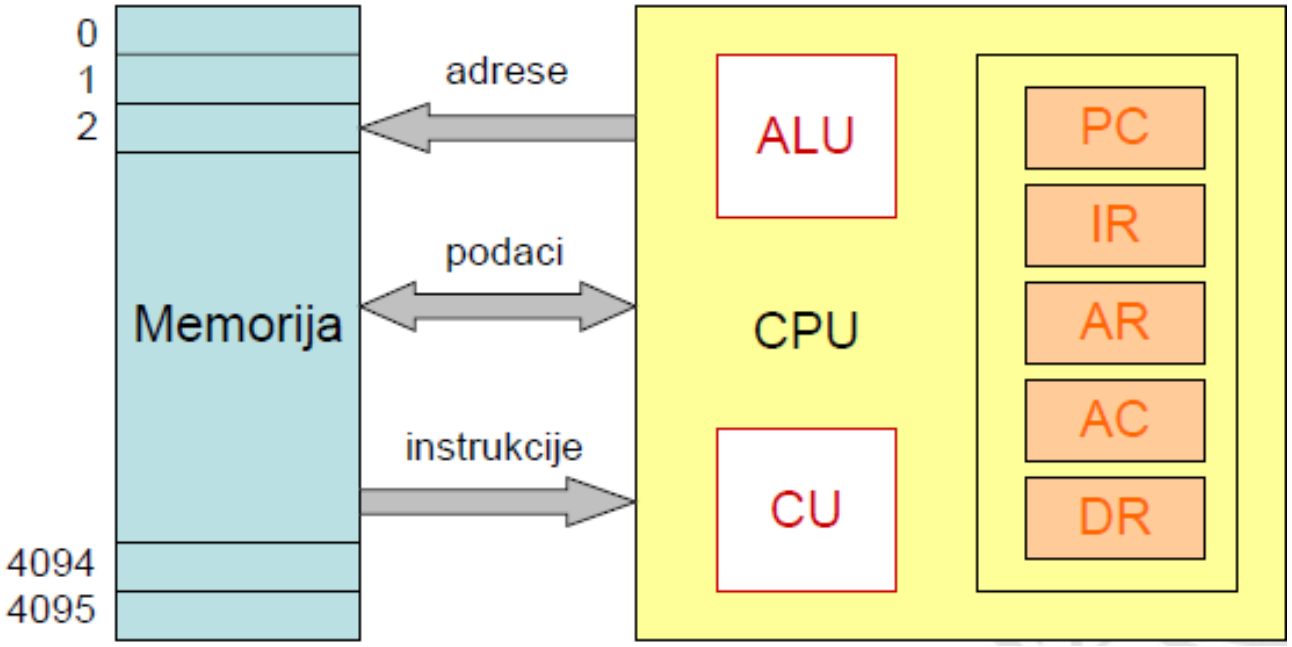
Da bi se izvršio asemblerski program, potrebni su:

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



Ilustrativni primer (1)

❖Programiranje će biti ilustrovano na primeru jednostavnog hipotetičkog procesora koji ima: 5 16-bitnih registara, 11 16-bitnih instrukcija u instrukcijskom setu i memoriju od 4096 8-bitnih reči.



Ilustrativni primer (2)

Procesorski registri

Registar	Oznaka
programski brojač	PC
instrukcijski registar	IR
adresni registar	AR
akumulator	AC
registar podatka	DR

Ilustrativni primer (3)

Instrukcijski set

KO	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	

Zadatak (1)

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

Zadatak (2)

Program na mašinskom jeziku

Adresa mem.lok.	Instrukcija	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.

Zadatak (3)

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.

HVALA VAM NA PAŽNJI

Redovni profesor dr Dušan Regodić, dipl.inž.