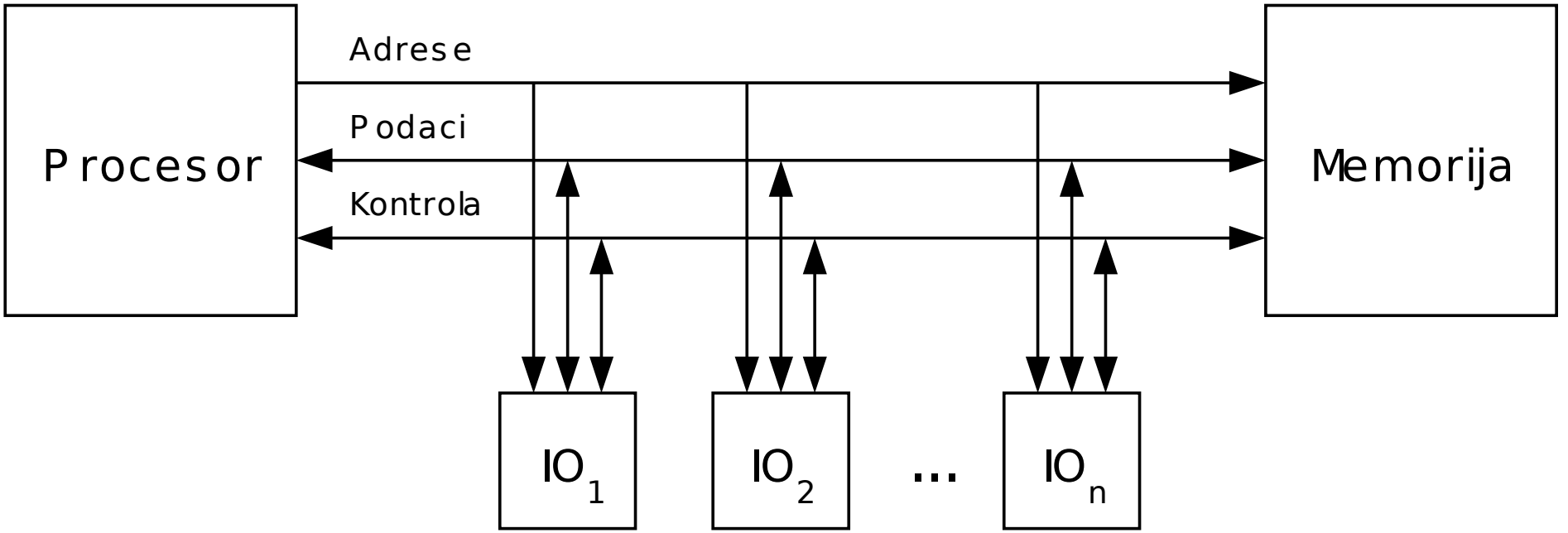
Arhitektura računara

dr.sc. Amer Hasanović

Osnovni pojmovi

- Računar
  - Uređaj koji izvršava proizvoljno dugu sekvencu instrukcija, tj. programe.
- Instrukcija
  - Jednostavna operacija, obično logičkog ili aritmetičkog karaktera.
- Procesor
  - Komponenta računara u kojoj se izvršava svaka instrukcija.
- Memorija
  - Komponenta računara koja pohranjuje:
    - instrukcije,
    - podatke koji se koriste u instrukcijama.

Pojednostavljen dijagram računara



Tipovi procesora

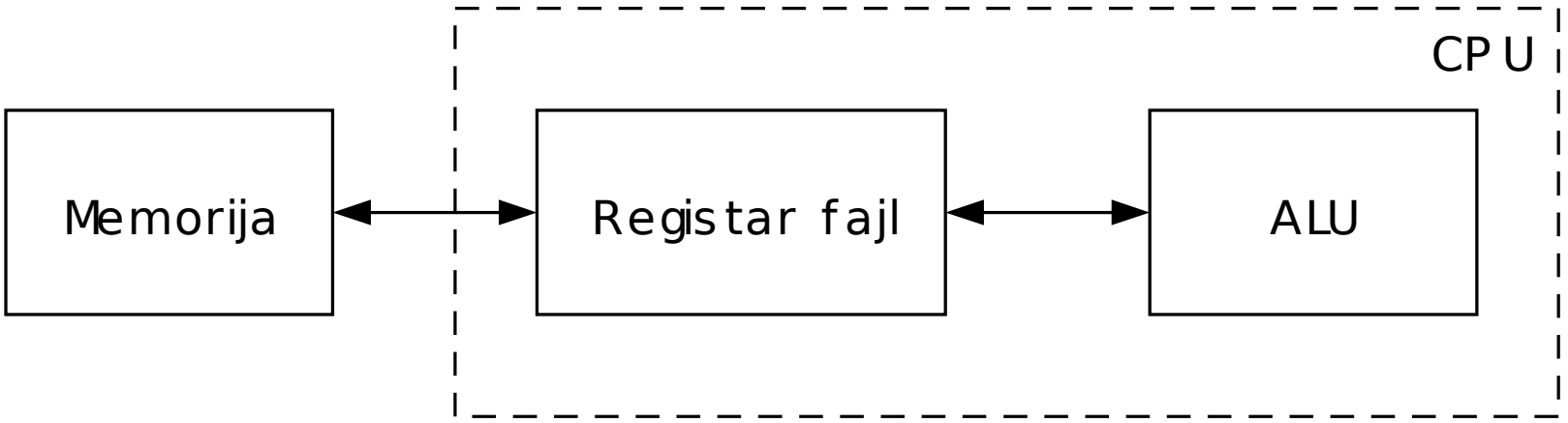
- RISC - Reduced Instruction Set Computer
  - Jednostavni procesor koji je specijaliziran da izvršava mali skup pažljivo odabranih instrukcija. (npr. ARM, **MIPS** itd.)
- CISC - Complex Instruction Set Computer
  - Kompleksan procesor koji direktno u hardveru implementira veliki broj instrukcija različitih namjena (npr. Intel itd.)

Princip rada procesora

- Bez obzira na tip procesora, od trenutka napajanja procesori izvršavaju instrukcije u petlji koja se sastoji od sljedećih koraka:

  1) Preuzimanje instrukcije iz memorije.
  2) Dekodiranje preuzete instrukcije.
  3) Izvršenje dekodirane instrukcije.
  4) Prelazak na izvršenje sljedeće instrukcije.

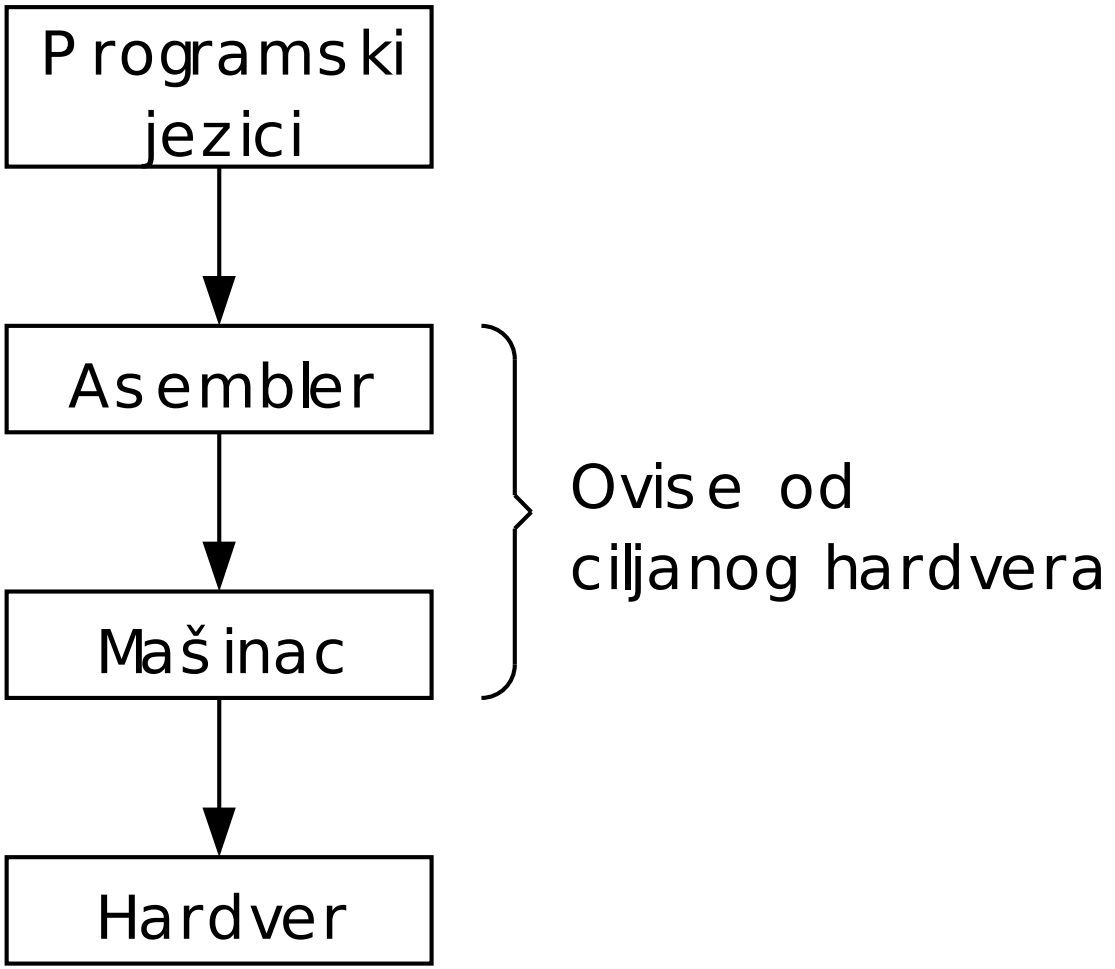
Load-Store arhitektura



- **Registar fajl** – sastavljen od registara, uređaja za privremenu pohranu podataka direktno unutar samog procesora. Registar fajl u MIPS procesoru ima 32 registra, a svaki registar putem flip-flopova može da pohrani 32 bita informacije.
- **ALU** – putem osnovnih logičkih kola, implementira operacije svih definiranih instrukcija iz domena date procesorske arhitekture (ISA).

Kreiranje programa

- Programiranje je proces formulacije konkretne sekvence instrukcija, tj. programa.
  - instrukcije se formulišu u binarnom formatu, tzv. **mašinac**, koji je razumljiv procesoru**.**
  - mašinac nije pogodan za direktnu manipulaciju od strane programera, pa se programi pišu u raznim programskim jezicima u tekstualnoj formi koja je razumljiva programerima.
  - program napisan u nekom programskom jeziku se tranformiše u mašinac u procesu **kompajliranja**.


Programski jezici
Asembler
Mašinac
Hardver
Ovise od ciljanog hardvera

Reprezentacije programa

Primjer C funkcija:

```
int foo(int a, int b) {
  return (a + b) / 8;
}
```

Asembler nakon kompajliranja:

MIPS:

```
foo:
  addu  $1, $5, $4
  sra   $2, $1, 31
  srl   $2, $2, 29
  addu  $1, $1, $2
  jr    $ra
  sra   $2, $1, 3
```

X86:

```
foo:
  movl  8(%esp), %ecx
  addl  4(%esp), %ecx
  movl  %ecx, %eax
  sarl  $31, %eax
  shrl  $29, %eax
  addl  %ecx, %eax
  sarl  $3, %eax
  retl
```

Mašinac nakon asembliranja:

MIPS:

```
 0:  00a40821
 4:  000117c3
 8:  00021742
 c:  00220821
10:  03e00008
14:  000110c3
```

X86:

```
 0:  8b4c2408
 4:  034c2404
 8:  89c8
 a:  c1f81f
 d:  c1e81d
10:  01c8
12:  c1f803
15:  c3
```

Elementi asembler koda

- Asembler datoteke mogu imati tri tipa elemenata:
  - **direktive** – počinju karakterom . i predstavljaju informacije ili upute koje asembler koristi u procesu asembliranja.
  - **oznake** – završavaju se karakterom : i asociraju pozicije (lokacije) u asembler kodu sa proizvoljnim imenima.
  - **instrukcije** – ne spadaju niti u jednu od prethodnih kategorija, a predstavljaju instrukcije određenog procesora koje će da čine program.

Primjer

- C – datoteka:

```
int main() {
  return 0;
}
```

- Asembler reprezentacija:

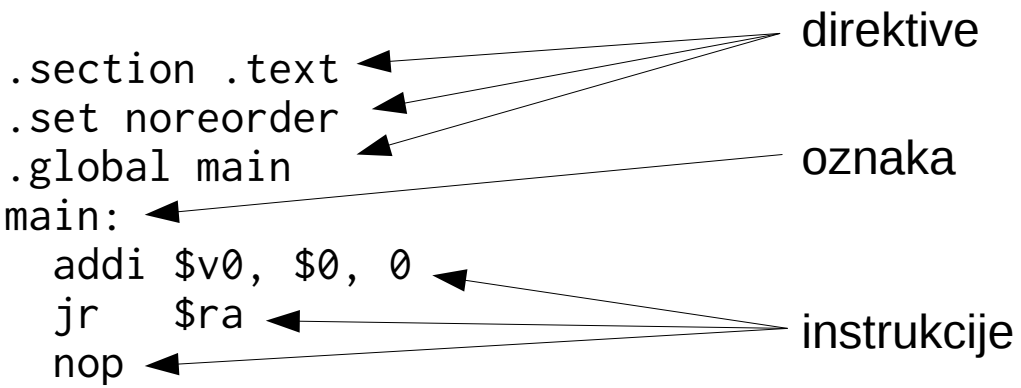
```
.section .text
.set noreorder
.global main
main:
  addi $v0, $0, 0
  jr   $ra
  nop
```

Primjer

- C – datoteka:

```
int main() {
  return 0;
}
```

- Asembler reprezentacija:



Izvršne datoteke

- Programi se distribuiraju u obliku izvršnih datotka u različitim binarnim formatima povezanim sa konkretnim operativnim sistema, npr.:
  - ELF – za Unix sisteme,
  - PE – za Microsoft Windows OS.
- Izvršna datoteka se sastoji od **sekcija,** u kojim se nalaze različiti strukturni elementi programa koji su neophodni za njegovo normalno funkcionisanje.
- Određene sekcije iz izvršnog fajla prilikom izvršavanja programa kopiraju se u memoriju, nakon čega se naređuje procesoru da počne izvršavati prvu instrukciju iz upravo kopiranog segmenta u memoriji, i to sa lokacije gdje se nalazi ulazna tačka u program, tzv. **entry point**.

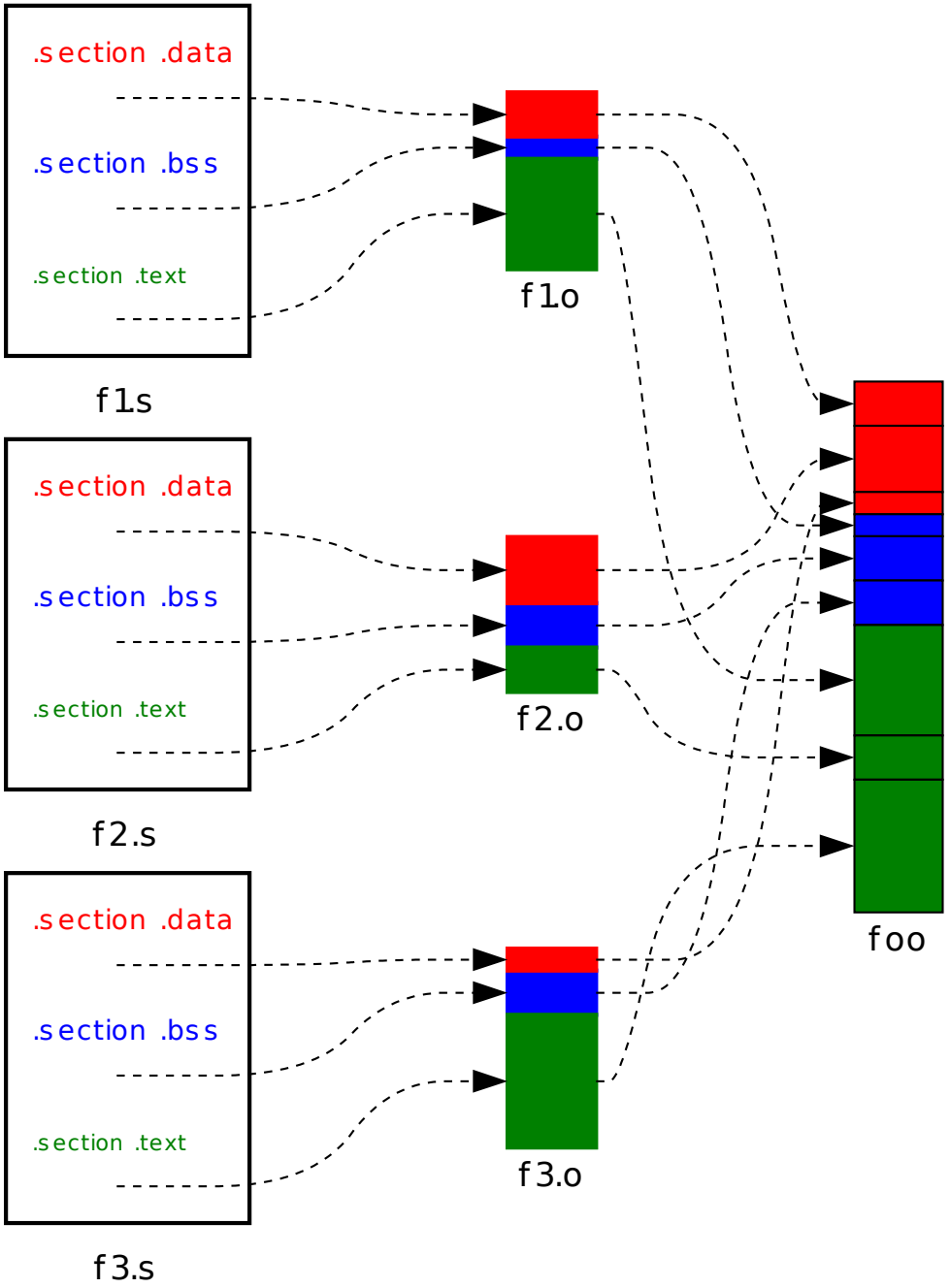
Sekcije asembler i objektnih fajlova

- Izvršna datoteka nastaje uvezivanjem (linkanjem) proizvoljnog broja **objektnih** fajlova.
  - sekcije prisutne u objektnim fajlovima kopiraju se u izvršni fajl.
- Pojedinačni objektni fajl nastaje asembliranjem pojedninačnih asembler fajlova.
  - sekcije koje nastaju u objektim fajlovima korespondiraju sekcijama u asembler fajlovima, a koje se definiraju putem **.section** asembler direktive.
  - simboli (tj. oznake) defnirani u jednom asembler fajlu mogu se eksportovati za korištenje u drugom fajlu putem instrukcije **.globl** ili .global

Sekcije ELF fajlova

- **text** – sekcija u kojoj se nalaze mašinske instrukcije nastale asembliranjem. Sekcija često nosi i oznaku “code” sekcije.
- **data** – sekcija u kojoj se nalaze vrijednosti inicijaliziranih globalnih varijabli.
- **bss** – sekcija rezervirana za neinicijalizirane globalne varijable.
- **rodata** – sekcija u kojoj se nalaze konstantne vrijednosti koje se koriste u programu.

Primjer asembliranja i uvezivanja



Asembler instrukcije

- Instrukcije imaju ključnu ulogu u asembler fajlovima, jer se putem istih na nivou asemblera zapravo i kreiraju programi.
- Gotovo striktno se nalaze u *text* sekcijama asembler datoteka.
- Za razliku od ostalih elemenata asembler koda, instrukcije su vezane za konkretni procesor.
- MIPS instrukcije u asembler fajlovima formatiraju se na više na čina. Najčešći je sljedeći format:
  - `ime_instrukcije operand1, operand2, operand3`

- Operandi u instrukcijama mogu biti: **brojevi**, **registri** ili **oznake**.
- Brojevi ukoliko se koriste u instrukcijama, pojavljuju se obično kao treći operand. Brojevi mogu biti zadani u decimalnom ili heksadecimalnom formatu, sa ili bez predznaka, odnosno na isti način kao u programskom jeziku C.
- Registri u instrukcijama počinju karakterom $ i mogu biti referencirani simboličkim imenima, npr., ili njihovim indeksom u registar fajlu MIPS procesora.
  - MIPS procesor u registar fajlu ima 32 registra koji se mogu koristiti kao operandi u instrukcijama.
  - Dok traje izvršenje određene instrukcije poseban MIPS registar označen kao **pc**, koji se ne nalazi u registar fajlu, uvijek sadrži adresu naredne instrukcije za izvršenje u programskoj sekvenci. Nusprodukt izvršenja svake instrukcije je promjena vrijednosti ovog registra.

Tabela registara MIPS procesora

Indeks	Simboličko ime	Opis
0	zero	vrijednost 0
1	$at	za asembler upotrebu
2-3	$v0-$v1	povratne vrijednosti funkcija
4-7	$a0-$a3	argumenti funkcija
8-15	$t0-$t7	privremene vrijednosti
16-23	$s0-$s7	snimljene vrijednosti
24-25	$t8-$t9	privremene vrijednosti
26-27	$k0-$k1	rezervirano za OS
28	$gp	globalni pointer
29	$sp	stek pointer
30	$fp	frejm pointer
31	$ra	povratna adresa