Arhitektura računara

dr.sc. Amer Hasanović

Pregled

- Disasembliranje
- Množenje, dijeljenje i pseudo-instrukcije
- Realni brojevi
  - reprezentacije
  - MIPS tretman realnih brojeva

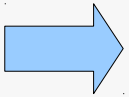
Disasembliranje

- Proces dekodiranja mašinskih instrukcija u asembli kod:
  - konvertirati mašinske instrukcije u binarni format
  - na osnovu *opcode* polja identificirati format instrukcije
    - `0` za `R`
    - `2` ili `3` za `J`
    - ostale vrijednosti za `I`
  - odrediti ostala polja u svakoj instrukciji
  - identificirati vrijednosti operanada npr. registri, adrese, numeričke konstante
  - reprezentirati svaku instrukciju sa MIPS asembli kodom

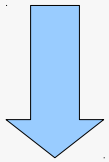
Primjer disasembliranje

- Neka je dat binarni kod u kombinaciji sa adresama na kojim je učitan mašinski kod:

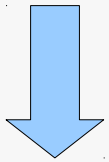
```
Adresa      Kod
0x00400000  0x12740002
0x00400004  0x02328022
0x00400008  0x08100004
0x0040000c  0x02328020
```



```
Adresa  Kod
4194304 00010010011101000000000000000010
4194308 00000010001100101000000000100010
4194312 00001000000100000000000000000100
4194316 00000010001100101000000000100000
```



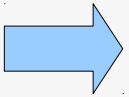
```
000100 10011101000000000000000010
000000 10001100101000000000100010
000010 00000100000000000000000100
000000 10001100101000000000100000
```



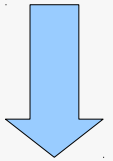
```
000100 10011 10100 0000000000000010
000000 10001 10010 10000 00000 100010
000010 00000100000000000000000100
000000 10001 10010 10000 00000 100000
```

Primjer disasembliranje

```
000100 10011 10100 0000000000000010
000000 10001 10010 10000 00000 100010
000010 00000100000000000000000100
000000 10001 10010 10000 00000 100000
```

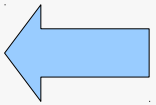


```
4 19 20 2
0 17 18 16 0 34
2 1048580
0 17 18 16 0 32
```



```
4 * 1048580 = 4194320
```

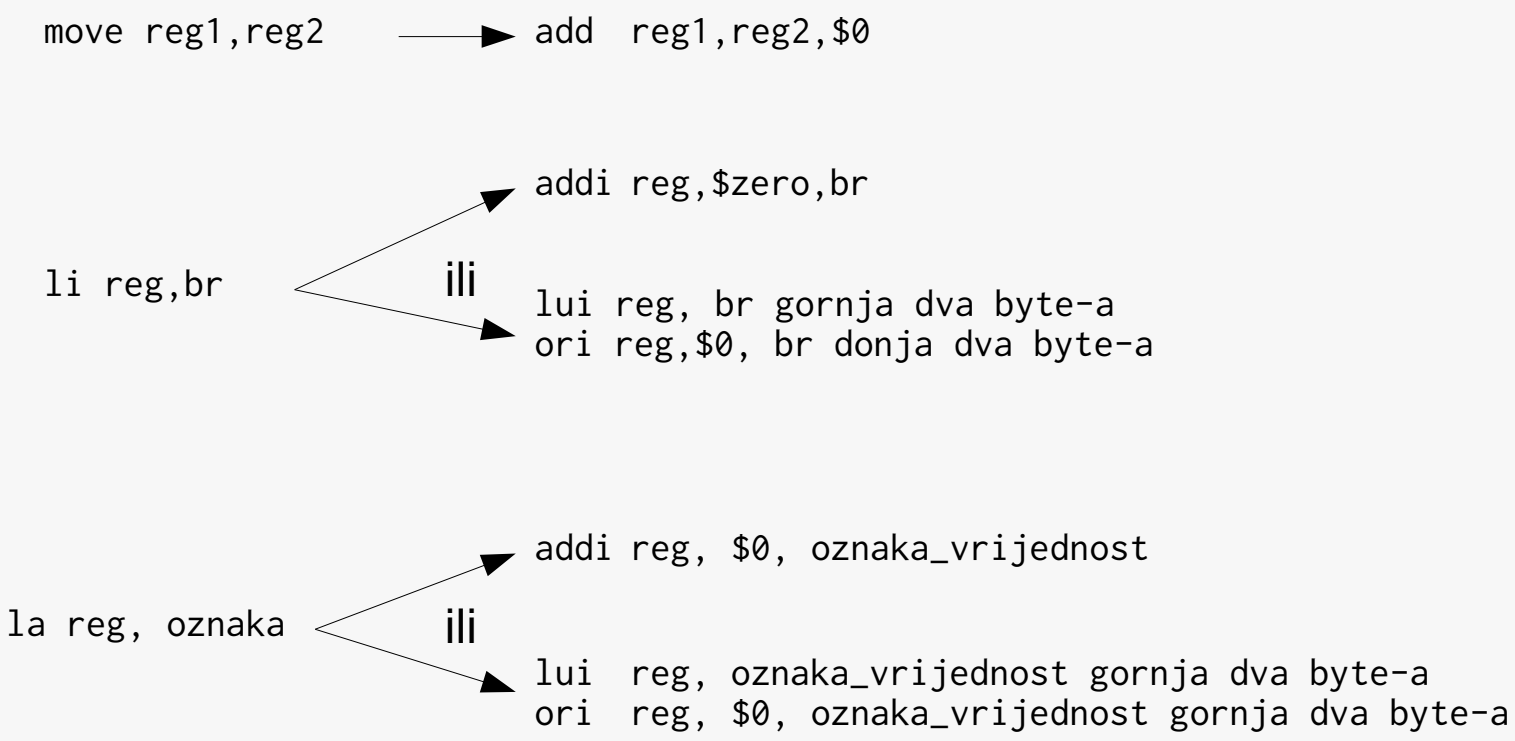
```
4194304 beq $s3 $s4 2
4194308 sub $s1 $s2 $s0
4194312 j 1048580
4194316 add $s1 $s2 $s0
```



```
    beq $s3, $s4, L1
    sub $s0, $s1, $s2
    j L2
L1: add $s0, $s1, $s2
L2:
```

Pseudo-instrukcije

- MIPS asembler podržava pseudo-instrukcije koje daju iluziju postojanja ekspresivnijeg skupa MIPS instrukcija
  - Ove instrukcije prevode se u jednu ili više MIPS instrukcija



Pseudo-instrukcije

- Pri generiranju pravih MIPS instrukcija na osnovu jedne pseudo-instrukcije asembler nekada treba registar.
  - registar `$at` (ili `$1`) rezerviran je za ovu namjenu
- U pseudo-instrukcije spadaju i neke instrukcije sa pogrešno unesnim tipom operanda:

```
add $t0,$0,52000   →   lui $1,$0,0
                       ori $1,$1,52000
                       add $8,$0,$1
```

- Često su u upotrebi termini:
  - MAL (MIPS Assembler Language) jezik koji uključuje pseudo-instrukcije
  - TAL (True Assembler Language) jezik koji uključuje samo instrukcije koje imaju binarnu reprezentaciju

Množenje i dijeljenje

- Za množenje i dijeljenje MIPS koristi registre `HI` i `LO`
  - Rezultat cijelobrojnog moženja podijeljen je na gornja 32 bita koji su u registru `HI` i donja 32 bita koji su u registru `LO`
  - Rezultat cijelobrojnog dijeljenja snima se u registar `LO` a ostatak u registar `HI`
  - Za prenos rezultata u generalne registre koristi se instrukcija `mflo` ili `mfhi`
- Instrukcije za množenje `mult`, `multu` npr:
  - `mult $t0,$t1`
- Instrukcije za dijeljenje `div`, `divu` npr:

```
li $t0,5
li $t1,3
div $t0,$t1
mfhi $s0
```

Realni brojevi reprezentacija

- Cijeli brojevi reprezentiraju se u binarnom obliku korištenjem dvojnog komplementa
- Realni brojevi u binarnom obliku zapisuju se u notaciji sa pomičnim zarezom spram IEEE 754 standarda i to u obliku:

- Polja *s*, *e* i *f* služe za konstrukciju broja čija se vrijednost iskazuje kao:

$$\pm \text{ mantisa} * 2^{\text{eksponent}}$$

- Postoje dvije varijante notacije:
  - *jednostruka preciznost*: broj je 32 bit, polje *e* je 8 a *f* 23 bita
  - *dvostruka preciznost*: broj je 64 bit, polje *e* je 11 a *f* 52 bita
  - u oba slučaja *s* polje predstavlja predznak i to 0 za + 1 za -

Mantisa

- Normalizirana forma bilo decimalnog ili binarnog zapisa podrazumjeva tačno jednu nenultu cifru sa lijeve strane zareza
  - Npr $0.238*10^3 = 2.38*10^2$
- Mantisa realnog broja je u normaliziranoj formi sa implicitnom cifrom 1 sa lijeve strane binarnog zareza
  - tj mantisa=1 + f
  - npr za f=01101 mantisa je 1.01101
  - Implicitno za mantisu je rezervirano 24 bit-a

Eksponent

- Polje *e* u zapisu predstavlja eksponent pomaknut (biased) i to:
  - za vrijednost 127 slučaj jednostruke preciznosti
  - za vrijednost 1023 slučaj dvostruke preciznosti
- Primjeri:
  - za eksponent 4, *e* polje je 127+4 ili 100000011
  - za *e* polje 01011101, eksponent je 93-127=-34

IEEE konverzija

- Konverzija u decimalnu vrijednost iz IEEE formata računa se na osnovu:
  - $(1 - 2s) * (1 + f) * 2^{e\text{-pomak}}$
  - e, f, s konveritrani u decimalnu formu
- Specijalni slučajevi:
  - Za e i f 0, broj je 0
  - Za e 255 i f 0, broj je beskonačan
  - Za e 255 i f različit od 0, broj je NAN

Primjeri konverzije

- Neka je dat broj 18.625
  - (16+2).(0.5*0.125)
  - $(2^4+2^1).(2^{-1}+2^{-3})$
  - 10010.101
  - $1.0010101*2^4$

0	10000011	00101010000000000000000

- Neka je dat broj 1 10000001 11000000000000000000000
  - s=1 riječ je o negativnom broju
  - e=10000001 ili 129 → eksponent je 129-127=2
  - 1.11_2*2^2 tj
  - 111_2*2^0 ili 7.0

MIPS i realni brojevi

- Za rad sa realnim brojevima MIPS koristi separatni skup od 32 registra označenih sa `$f0`, `$f1`,... `$f31`
  - Prve implementacije MIPS procesora koristile 32 bitne fp registre na koprocesoru
  - Moguć rad sa jednostrukom i dvostrukom preciznošću.
    - Za dvostruku preciznost koristi se par registara
- Sve operacije na realnim brojevima izvode se striktno na fp registrima i to `s` za jednostruku preciznost i `d` za dvostruku preciznost
  - `add.s, div.s, mult.s, sub.s`
  - `add.d, div.d, mult.d, sub.d`

fp registri i prenos podataka

- Za pristup memoriji koriste se slično kao `lw` i `sw`
  - `lwc1, swc1`
  - Destinacija je neki od fp registara
  - Adresni registar je neki od generalnih registara
- Za prenos podataka (bez konverzije):
  - `mtc1` reg,fpreg
  - `mfc1` reg,fpreg
  - `mov.s` i `mov.d` izvor i destinacija fp registri

fp registri i konverzija podataka

- Za konverziju podataka koriste se striktno fp registri
- Operacija za konverziju je u obliku:
  - `cvt.x.y fd,fs`
  - `fd,fs` destinacijski I izvorni registri
  - `x` destinacijski tip, `y` izvorni tip
  - Tipovi:
    - `s` single fp, `d` double fp, `w` 32 bit cijeli broj
- Eksplicitna konverzija fp → cijeli broj može se izvesti putem:
  - `trunc.x.y, round.x.y, ceil.x.y, floor.x.y`

fp registri poređenje i grananje

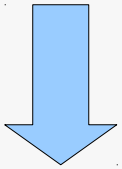
- Za poređenje realnih brojeva koristi se operacija:
  - `c.uslov.d fs1,fs2`
  - Uslov predstavlja tip poređenja npr `eq` → jednako, `lt` → manje, `gt` → veće, `le` → manje ili jednako itd...
  - Stanje bita CC u FP statusno/kontrolnom registru FCSR mijenja se u ovisnosti od rezultata operacije poređenja
- Na osnovu stanja statusnog bita CC vrši se grananje i to u oblicima:
  - `bc1t oznaka # za tacno`
  - `bc1f oznaka # za netacno`

fp registri i funkcije

- Tretman fp registara u funcijama:
  - Povratne vrijednosti: `$f0`, `$f2`
  - Prezervirani registri: `$f20` do `$f30`
  - Privremeni registri: `$f4` do `$f10`, `$f16` i `$f18`
- Proslijeđivanje vrijednosti:
  - Samo za slučaj da je prvi argument tip fp, koriste se `$f12` i `$f14` za prva dva fp argumenta a ostatak na stack-u
  - U suprotnom fp argumenti proslijeđuju se na stack-u

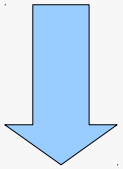
Primjeri C funkcija

```
double f1(float *p)
{
  return *p;
}
```



```
f1:
   lwc1 $f0,0($a0)
   cvt.d.s $f0,$f0
   jr $ra
```

```
int f2(double a, int b)
{
  return a+b;
}
```



```
f2:
   mtc1 $a2,$f4
   cvt.d.w $f4,$f4
   add.d $f4,$f4,$f12
   trunc.w.d $f4,$f4
   mfc1 $v0,$f4
   jr $ra
```