Građa računala

Ponavljanje brojevnih sustava
i uvodni programi

Brojevni sustavi (ponavljanje)

- Pri računanju koristimo dekadsku bazu:

$$123 = 1 * 10^2 + 2 * 10^1 + 3 * 10^0$$

- Broj ne ovisi o zapisu u bazi – broj možemo zamisliti kao udaljenost od nule (lijevo ili desno, ovisno o predznaku)
  - zapis generalno nije jedinstven: 2/4 = 4/8 = 0.5
  - Uočite: zapis se mijenja, broj ne!
- Ovakav zapis jedan je od beskonačno mnogo mogućih (npr. rimski numerali)

Brojevni sustavi (ponavljanje)

- Znamenka
  - koeficijent u polinomu kojem je jedina varijabla baza b

$$(123)_b = 1b^2 + 2b^1 + 3b^0$$

- Baza
  - najveća potencija baze b koja se javlja određuje broj znamenaka broja
  - određuje točku u kojoj evaluiramo polinom
- Uočimo: $(123)_{10} \neq (123)_{15}$
  - radi se o potpuno drugom broju (polinom smo evaluirali u drugim točkama)

Promjena baze

- Promjena baze u dekadsku
  - Izračunavanje polinoma (Hornerova shema je najbrža)

$$(a_n a_{n-1} \ldots a_0)_b = a_n b^n + a_{n-1} b^{n-1} + \ldots + a_0$$

- Promjena baze iz dekadske:
  - Dijelimo bazom i gledamo ostatak, ostatak čitamo unazad, ponavljamo dok ostatak nije jednak nuli

$$a_n a_{n-1} \ldots a_0 \div b = r_1 + o_1$$
$$r_1 \div b = r_2 + o_2 \longrightarrow Broj\ je : (o_k o_{k-1} \ldots o_1)$$
$$\ldots$$
$$r_{k-1} \div b = r_k + o_k$$

Promjena baze (direktno)

- Ako je jedna baza potencija druge, možemo promjenu raditi na razini znamenki
  - Generalni slučaj – podsjetiti se s Prog1!
- Binarni ↔ Heksadecimalni
  - Jedna znamenka heksadecimalne baze odgovara točno 4 znamenke binarne baze

$0000_2 = 0_{16}$	$0100_2 = 4_{16}$	$1000_2 = 8_{16}$	$1100_2 = C_{16}$
$0001_2 = 1_{16}$	$0101_2 = 5_{16}$	$1001_2 = 9_{16}$	$1101_2 = D_{16}$
$0010_2 = 2_{16}$	$0110_2 = 6_{16}$	$1010_2 = A_{16}$	$1110_2 = E_{16}$
$0011_2 = 3_{16}$	$0111_2 = 7_{16}$	$1011_2 = B_{16}$	$1111_2 = F_{16}$

Decimalni brojevi

- Princip je isti:

$$0.321 = 3 * 10^{-1} + 2 * 10^{-2} + 1 * 10^{-3}$$

- "Razlika"
  - Dijeljenje s recipročnim vrijednostima je množenje

Za binarne brojeve gledamo decimalni dio:

- Množimo s 2
- Gledamo ostatak dijeljenja s 2
- Decimalni dio jednak 0 – kraj
- Čitamo odozgo prema dolje

$$0.125 * 2 = 0.25$$
$$0.25 * 2 = 0.5$$
$$0.5 * 2 = 1.0$$
$$Rezultat : 0.001$$

Zapis (cijelih) brojeva u računalu

- Svi brojevi su nizevi bitova
  - 1, 2 ili 4 bajta
  - Big endian
- *Signed* vs *unsigned*
  - Razlika – sign bit, krajnji lijevi bit
  - (`.B`) – 8. bit, (`.W`) – 16. bit, (`.L`) – 32. bit
- Negativni brojevi ne postoje na razini stroja
  - Oduzimanje = zbrajanje dvojnog komplementa u modularnoj aritmetici
- Overflow – kada se promijeni *sign* bit
- Nakon najvećeg pozitivnog broja slijedi najveći negativni

Dvojni komplement

- 32 bitni broj: `0010 0010 1110 1111`
- Komplement: `1101 1101 0001 0000`
- Dodamo jedan

  `1101 1101 0001 0001 = "-0010 0010 1110 1111"`
- Operacija je sama sebi inverz
- Primjer s 4 bita:

  `7 = 0111` (pozitivni broj)

  `-7 = 1000 + 1 = 1001 = 9` (njegov komplement – "negativan" broj)
- Modularna aritmetika:

  `10 + 9 = 19 mod (2^4) = 3`

  `10 - 7 = 3`

Direktiva `DS`

- Služi za zauzimanje mjesta
  - Pridružuje adresu nekoj labeli
  - Instrukcije ili direktive koje slijede smještaju se u adrese nakon zauzetih
- Prima jedan parametar – broj (bez #)
  - Određuje broj blokova koji se zauzimaju
  - Veličina bloka ovisi o sufiksu koji je `.B`, `.W` ili `.L` (kao kod instrukcija)

Direktiva DS

- Iako VAR1 ima samo jednu riječ, možemo pristupiti long-wordu koji je na toj adresi!
- Zapamtite: DS samo pridružuje adresu labeli i ne "zna" ništa o veličini podatka koji se tamo nalazi

Primjer: gr-v3_01.x68

```
        ORG     $1000

VAR1    DS.W    1

VAR2    DS.W    4

START:  MOVE.L  VAR1, D0

        SIMHALT

        END     START
```

Pitanja:

1)Koja je adresa labele START (tj. na kojoj adresi se nalazi MOVE.L)?

2)Koja je adresa VAR2?

3)Kako bismo pristupili četvrtoj riječi varijable VAR2?

Direktiva DS

4) Je li ovaj program sintaktički ispravan?

   Ako da, što radi?

```
        ORG     $1000
VAR1    DS.B    1
VAR2    DS.B    8
START:  MOVE.L  #VAR1, VAR2
        SIMHALT
        END     START
```

Dodatno:

5) Kako bismo, za varijable zadane kao gore, četvrtu **riječ** varijable `VAR2` prekopirali na adresu `$6000`? Pripazite, `VAR1` je veličine jednog bajta pa je `VAR2` na neparnoj adresi.

Logičke operacije nad bitovima

- Instrukcija `NOT`
  - Radi bitovni komplement
  - Postavlja N i Z zastavice
- Instrukcija `AND`, `OR`, `EOR`
  - Binarne operacije
  - Adresiranje – jedan operand je uvijek podatkovni registar
- Rad s bitovima
  - `NOT` – negira svaki bit
  - `AND` – možemo podesiti određene bitove na 0
  - `OR` – možemo podesiti određene bitove na 1

Instrukcije `LSL` i `LSR`

- Logical Shift Left/Right
  - Pomiču bitove za dani broj mjesta
  - Zastavice – `C` se postavlja na bit koji “izbacujemo”
  - Bit koji se ubacuje uvijek je 0
- Ekvivalentni množenju/dijeljenju s 2 (zašto?)
- U kombinaciji s logičkim operatorima možemo dohvatiti pojedinačne bitove

Maskiranje bitova

- Bitove maskiramo kada
  - nisu bitni (gledamo određeni segment bitova)
  - moraju imati specifičnu vrijednost za izvršavanje operacije
- Pomoću maskiranja možemo raditi sa segmentima manjim od bajta
  - npr. bajt podijelimo na dva dijela, svaki spremimo posebno
  - Veća fleksibilnost ali gubimo podršku procesora za instrukcije koje koriste uvjetni registar

Logičke operacije i maskiranje bitova

- **Primjeri:** gr-v3_02.x68, gr-v3_04.x68, gr-v3_05.x68
- **Pitanja i zadaci**:

1) Kako dohvatiti najznačajniji/najmanje značajan bit registra?
2) Napišite program koji iz 32 bitne varijable na adresi $1000 vadi bitove 4-12 (bitove brojite od nule) i sprema ih na adresu $2000.
3) Rastavite 32bitnu varijablu na adresi $1000 na pojedinačne bajtove, spremite ih redom u registre D4-D7
4) Na adresi $2000 nalazi se 16 bitna varijabla VAR. Na adresi neposljedno nakon nalazi se 8 bitna varijabla NUM u kojoj piše broj n<16. Na adresu $6000 spremite vrijednost dobivenu bitovnim AND-om varijable VAR i broja 2^n (koje su moguće vrijednosti?).

Logičke operacije i maskiranje bitova

5) Napišite program koji radi bitovno *isključivo ili* sadržaja dviju registara (bez korištenja `EOR` instrukcije).

Dodatno*:

6) Napišite program koji zbraja dvije 4 bitne varijable korištenjem logičkih operatora, bez korištenja `ADD` ili `SUB`.

Instrukcije `ADD` i `SUB`

- Jedan od operanada je uvijek podatkovni registar
- `SUB = ADD` s dvojnim komplementom
  - Razlike u postavljenim zastavicama
- Zastavice
  - Sve se postavljaju
  - N – ako je rezultat negativan broj
  - Z – ako je rezultat nula
  - V – ako se postavio najznačajniji bit
  - C – ako rezultat ne stane u dani spremnik/ako je “posuđen” najznačajniji bit

Instrukcije `ADD` i `SUB`

- **Primjer:** gr-v3_03.x68
- **Pitanja i zadaci:**

1)Hoće li se postaviti V bit ako zbrojimo 130 i 1? Hoće li se postaviti N bit?

2)Hoće li se postaviti V bit ako zbrojimo 127 i 1?

3)Hoće li se postaviti C bit ako zbrojimo 200 + 200 kao bajtove? Što ako ih zbrojimo kao riječi?

4)Demonstrirajte u programu da je dodavanje negativnog broja ekvivalentno oduzimanju njegovog dvojnog komplementa (ne gledajući zastavice).

Pitanja i zadaci

5) Napišite program koji zbraja dvije četverobitne vrijednosti i sprema “jedan dalje” u registar D0.

Dodatno:

6) Napisati program koji čita dva 32 bitna operanda iz memorije. Brojeve treba zbrojiti i zapisati u memoriju tako da je prvi bit rezultata spremljen na treći bit adrese $6000 (bitovi koji ne pripadaju rezultatu neka budu 0).