Građa računala

Kolokvij i aritmetički problemi

Kolokvij

- **ZADATAK 4.** Na lokaciji STRING nalazi se neki niz znakova čiji kraj je označen nul-znakom (vrijednosti $00). Napišite program koji upisuje broj $1 na lokaciju REZ ako se dani niz sastoji samo od znakova ‘0’ i ‘1’, a upisuje $0 inače.

Rješenje

```
program equ $1000
data    equ $6000

        org data
string: ds.l  1
rez:    ds.b  1
jedan:  equ  '1'
nula:   equ  '0'

        org program
start:
        ;; stavimo pointer u a0
        movea.l string, a0
        ;; pretpostavimo da broj zadovoljava kriterij
        move.b  #1, d1

        ;; prolazimo po svim znakovima, gledamo
        ;; zadovoljavaju li kriterij
        move.b (a0)+, d0
```

```
loop:   cmp.b #0, d0
        beq kraj

        ;; ako je jedinica, idemo na sljedeci korak
        cmp.b #jedan, d0
        beq loopend
        ;; ako je nula, isto je oke
        cmp.b #nula, d0
        beq loopend
        ;; ako smo ovdje onda nije ni nula ni jedan
        move.b  #0, d1
        bra kraj
loopend:
        move.b (a0)+, d0
        bra loop
kraj:
        move.b d1, rez
        end start
```

Kolokvij

- **ZADATAK 5.** Za prirodni broj kažemo da je “dobar” ako su mu sve znamenke jednake u heksadecimalnom zapisu. Napišite program koji prolazi po danom nizu 32-bitnih brojeva te za svaki broj određuje je li “dobar”. Program treba na lokaciju BROJ ispisati koliko je “dobrih” brojeva pronađeno u nizu.
- Adresa prvog elementa niza nalazi se na lokaciji NIZ, a duljina niza je dana na lokaciji DULJINA.

```
program   equ $1000
data      equ $6000

          org data
niz       ds.l  1
duljina   ds.l  1
rez       ds.l  1

          org program
start:
          movea.l niz, a0
          move.l  #0, d0 ; brojac "dobrih"
          move.l  duljina, d1
          ;; ako je duljina 0, skacemo na kraj
          beq kraj
          ;; pripremimo za dbra
          subq  #1, d1
          ;; sad krecemo u loop
loop1:
          move.l  (a0)+, d2 ;; u d2 je privremeni broj
          ;; cak ne trebamo loop za drugi dio, mozemo samo hardkodirati 8 puta istu stvar
          ;; ovdje ipak rjesavam s loopom :)
          move.l  #6, d7 ;; brojac
          move.l  d2, d3
          andi.l  #$f, d3 ;; uzmemo samo prvu znamenku
          lsr.l   #$4, d2 ;; maknemo prvu znamenku (nije greska ako ostane)
loop2:
          ;; u d2 imamo broj, gledamo da li su mu preostale znamenke jednake prvoj
          move.l  d2, d4 ;; zapamtimo vrijednost
          andi.l  #$f, d2 ;; uzmemo sljedecu znamenku
          cmp.l d2, d3 ;; pogledamo je li ista kao prva
          bne   loop1_end ;; ako nije, odmah mozemo na sljedeci broj
          move.l  d4, d2 ;; vratimo vrijednost
          lsr.l #$4, d2 ;; ako je, pripremimo se za sljedecu znamenku
          dbra  d7, loop2 ;; broj je 32 bitni, ponavljamo 7 puta (prva znamenka je posebna)
          ;; ako propadnemo tu, onda su sve znamenke iste
          addq.l  #1, d0
loop1_end:
          dbra  d1, loop1
kraj:
          move.l  d0, rez
          end start
```

Rješenje

- Trik:
  - možemo uočiti da broj ima sve znamenke jednake ako i samo rotacijom znamenki dobijemo ponovno isti broj
  - Neka je WXYZ broj gdje su W, X, Y i Z znamenke
  - Neka vrijedi WXYZ = XYZW
  - Slijedi: W = X, X = Y, Y = Z, Z = W iz čega slijedi da su sve znamenke iste

Instrukcija MOVE, nastavak

- MOVE <ea>, CCR
  - služi za pomak u uvjetni registar
  - uvijek pomiče riječ, ali gleda samo donji bajt
- MOVE <ea>, SR
  - pomak u statusni registar
  - pomiče riječ, utječe na cijeli registar
  - supervisor bit mora biti postavljen!
- MOVE SR, <ea>

```
START:
    MOVE #0, SR
    MOVE #0, SR
    END    START
```

Instrukcija ADDX

- Zbraja dva registra ili dvije memorijske lokacije
- Dodaje X zastavicu
  - omogućuje zbrajanje proizvoljno velikih brojeva
- Dvije varijante
  - Dn, Dn
  - -(An), -(An)
- Byte, word, longword veličine podataka

Primjeri

- **Primjer:** gr-v7-2010_01.x68

Instrukcija LEA

- Ponoviti adresiranja!

BCD brojevi

- *Binary Coded Decimal*
  - način spremanja dekadskih brojeva u 16 bitova

- Instrukcija ABCD
  - zbraja dva BCD broja
  - adresiranja - kao kod ADDX

Primjeri

Množenje brojeva

- Instrukcija MUL