Građa računala

Petlje i rad sa znakovima

Izvedba WHILEa

- **Primjer:** gr-v5-01.x68 – Potrebno je odrediti duljinu niza znakova (stringa). Adresa početka stringa je sadržana u 32-bitnoj varijabli `START`. Kraj stringa označen je ASCII znakom `CR` (`$0D`). Rezultat je potrebno spremiti u 16-bitnu varijablu `LENGTH`

```
char* a0 = START;
short d0 = 0;
while (*(a0++) != $0d)
  d0 += 1;
LENGTH = d0;
```

```
       MOVEA.L  START, A0
       MOVEQ #0, D0

LOOP:  CMPI.B   #$0D, (A0)+
       BEQ.S DONE

       ADDQ.W   #1, D0
       BRA   LOOP

DONE:  MOVE.W   D0, LENGTH
```

Izvedba DO-WHILEa

- **Primjer:** gr-v5-01b.x68 – nešto brža (u ovome asembleru, ne nužno i u C-u) implementacija istog programa

```
char* a0 = START;
short d0 = -1;
char d1 = $0d;
do
  d0 += 1;
while (*(a0++) != d1)
LENGTH = d0;
```

```
        MOVEA.L  START, A0
        MOVEQ #-1, D0
        MOVEQ #$0D, D1

LOOP:   ADDQ.W   #1, D0
        CMP.B (A0)+, D1
        BNE   LOOP

        MOVE.W   D0, LENGTH
```

Izvedba WHILEa

- **Primjer:** gr-v5-02.x68 – Pronađite u stringu ASCII znakova prvi znak koji nije razmak ($20). Početna adresa stringa je 32-bitna varijabla `START`. Adresu prvog nepraznog znaka spremite u varijablu `POINTER`.

```
char* a0 = START;
char d1 = ‘ ‘;
while (*(a0++) == d1) ;
a0 -= 1;
POINTER = a0;
```

```
        MOVEA.L   START, A0
        MOVEQ  #‘ ‘, D1

LOOP:   CMP.B  (A0)+, D1
        BEQ LOOP

        SUBQ.L #1, A0

        MOVE.L A0, POINTER
```

FOR s fiksnim brojem ponavljanja

- Ukoliko unaprijed znamo koliko najviše puta će nam se izvršiti neka petlja, u zavisnosti od nekog uvjeta, možemo koristiti neku od instrukcija DB?? (upitnici zamijenjuju ista slova kao i kod B?? instrukcije), koje izvrše skok u zavisnosti od stanja određenih zastavice uvjetnog registra postavljene, i brojača.
- Npr. DBRA <Dn>, <adresa> će umanjiti Dn za jedan, te će ukoliko je D0 različito od -1 izvršiti skok na adresu.
- Npr. sljedeći programčić zavrti petlju **8** puta
- for(d5 = **7**; d5 != -1; --d5)
  instrukcije;

```
        move.w 7, d5
skok:   instrukcije...
        dbra d5, skok
```

Reprezentacija stringova

- U nekim programskim jezicima (npr. C) string je null-terminated (kraj stringa označava znak s ASCII vrijednosti 0), dok je u nekim drugima (npr. Pascal, std-string) takav da prvo dolazi broj (8, 16 ili 32-bitni) – duljina stringa, a nakon njega sam sadržaj stringa

Reprezentacija stringova

- **Primjer:** gr-v5-03.x68 – Zamijenite sve početne nule sa razmacima u stringu. Početna adresa stringa je 32-bitna varijabla `START`. Prva dva bajta određuju duljinu stringa u bajtovima (niz zapravo počinje na 3. znaku)

```
char* a0 = START;
char d0='0', d1 = ' ';
short d2=*((short*)p);p += 2;
if (d2) {
  for(d2 -= 1; d2 != -1;--d2){
    if (*a0++ != d0) break;
    a0[-1] = d1;
  }
```

```
        MOVEA.L  START, A0
        MOVEQ #'0', D0
        MOVEQ #' ', D1

        MOVE.W   (A0)+, D2
        BEQ.S    DONE

        SUBQ.W   #1, D2
LOOP:   CMP.B    (A0)+, D0
        BNE.S    DONE
        MOVE.B   D1, -1(A0)
        DBRA     D2, LOOP
DONE:
```

Reprezentacija stringova

- **Primjer:** gr-v5-04.x68 – Provjerite da li su dva niza ASCII znakova jednaka. Početne adrese stringova su u varijablama `START1` i `START2`, a svaki je zapisan, tako da mu prvi znak određuje duljinu. Ako su jednaki, varijablu `MATCH` postavite na 0, inače na -1.

Primjer: gr-v5-04.x68

```
char *a0 = START1, *a1 = START2;
short d1 = -1;

char d0 = *(a0++);
if (d0 != *(a1++)) goto done;
if (d0) {
  bool uvjet;
  for(d0-=1; d0 != -1; --d0)
    uvjet=*(a0++) == *(a1++);
    if (!uvjet) goto done;
}
same:
  d1 = 0;
done:
  MATCH = d1;
```

```
    MOVEA.L   START1, A0
    MOVEA.L   START2, A1
    MOVEQ     #-1, D1

    MOVE.B (A0)+, D0
    CMP.B  (A1)+, D0
    BNE.S  DONE

    TST.B  D0
    BEQ.S  SAME

    SUBQ.W #1, D0
LOOP:
    CMPM.B (A0)+, (A1)+
    DBNE   D0, LOOP

    BNE.S  DONE

SAME:  MOVEQ  #0, D1

DONE:  MOVE.W D1, MATCH
```

Primjer: gr-v5-05.x68

- Konvertirajte vrijednost heksadecimane 8-bitne varijable DIGIT u odgovarajući znak (npr. 9 -> $39 (‘9’)). DIGIT sadrži točno jednu heksadecimalnu znamenku. Dobiveni ASCII znak spremite u varijablu CHAR

```
char d0 = DIGIT;
if (!(d0 < 10)) {
  d0 += ‘A’-’0’-10;
}
d0 += ‘0’;
CHAR = d0;
```

```
        MOVE.B   DIGIT, D0
        CMP.B #10, D0
        BLT.S ADD_0

        ADD.B #'A'-'0'-10, D0

ADD_0:
        ADD.B #'0', D0
        MOVE.B   D0, CHAR
```

Direktiva DC

- Za razliku od direktiva DS, kojom na neki način dodajemo varijable, ukoliko želimo imati niz varijabli s nekim unaprijed zadanim vrijednostima, možemo korisiti direktivu DC

```
short var1[] = {2, 4, 6, 8};
short var2[4];
```

```
    ORG $1000
VAR1 DC.W 2,4,6,8
VAR2 DS.W 4
START: MOVE.W VAR1, D0
        ...

    END START
```

Instrukcije CLR i EXT

- Instrukcija `CLR` (.B, .W ili .L) <mjesto> uništi prvih 8, 16 ili 32 bita određenog registra ili varijable.

- Instrukcija `EXT` (.W ili .L) Dn vrši sign extension registra Dn. `EXT.W` vrijednost najvažnijeg (7.) bita postavi na sve bitova 8-15, dok `EXT.L` vrijednost 15. bita postavi na sve bitove 16-31. (Dakle ako je npr D0 bio 8 bitni signed broj, nakon `EXT.W D0`, postati će 16. bitni signed broj s istom vrijednosti). Doduše u sljedećem primjeru nam ta instrukcija treba samo da bitove 8-15 postavimo na 0.

Primjer: gr-v5-06.x68

- Konvertirajte vrijednost 8-bitne varijable `DIGIT` u sedmosegmentni oblik, te ju spremite u varijablu `CODE`. Ukoliko `DIGIT` ne sadrži broj između 0 i 9, neka je rezultat 0.

```
char TABLICA[] = {
  $3F,$06,$5B,$4F,$66,
  $6D,$7D,$07,$7F,$6F};
char* a0=TABLICA;
char d1 = 0;
char d0=DIGIT;
if (d0 < 10) {
  d0 -> short
  d1 = a0[d0+0];
}
CODE = d1;
```

```
TABLICA: DC.B$3F,$06,$5B,$4F,$66,$6D,$7D,
$07,$7F,$6F

POCETAK: MOVEA.L  #TABLICA, A0
         CLR.B    D1
         MOVE.B   DIGIT, D0
         CMP.B    #9, D0
         BHI.S    DONE

         EXT.W    D0
         MOVE.B   0(A0,D0), D1

DONE:    MOVE.B   D1, CODE

         END  POCETAK
```

Instrukcija ROL, ROR, EXG

- Instrukcije ROL i ROR, rotiraju operand za dani broj bitova ulijevo/udesno. Nakon rotacije, vrijednost C zastavice jednaka je posljednjem rotiranom bitu
- Instrukcija EXG radi zamjenu sadržaja dvaju registara. Zamjena se može raditi i između podatkovnog i adresnog registra

Primjer: gr-v5-07.x68

- Konvertirajte vrijednost varijable `CHAR` s adrese $6000 iz ASCII znaka u odgovarajući decimalan broj i spremite rezultat u varijablu `DIGIT` s adrese $6001. Ako `CHAR` ne sadrži jednoznamenkasti decimalan broj u ASCII obliku onda upišite `$FF` u `DIGIT`.

Primjer: gr-v5-08.x68

- Konvertirajte broj prikazan u 16 bitnom binarnom zapisu u varijabli NUMBER s adrese $6000 u 16 ASCII znakova (ili '0' ili '1') koji predstavljaju taj broj. Spremite rezultat u niz od 16 znakova s početnom adresom $6002.