Građa računala

Grananja programa

Bezuvjetni skok

- Ukoliko želimo nastaviti izvršavati program od neke druge lokacije u memoriji, možemo koristiti instrukciju JMP ili BRA. Obje instrukcije primaju jedan parametar – adresu instrukcije od koje nastavljamo izvršavanje
- JMP prima stvarnu adresu (ili registar) i nastavlja izvršavanje programa od toga mjesta, dakle kao da smo napisali MOVE.L adresa, PC
- BRA prima relativnu adresu (8 ili 16 biti broj). Možemo navesti veličinu broja, ali i ne trebamo, kompajler će sam odrediti stane li mu udaljenost za skok u 8 ili 16 bita. Adresa se shvaća kao broj sa predznakom, te se trenutni PC uveća (umanji) za dani parametar
- BRA.S – kratki skok – adresa je 8 bitna
- BRA.L – dugi skok – adresa je 16 bitna

```
SKOK:
    ...
    JMP SKOK
    BRA SKOK
```

Grananje izvršavanja programa

- Ukoliko želimo izvršiti skok na neku drugu instrukciju u zavisnosti od nekog uvjeta, možemo koristiti neku od instrukcija B?? (**B**ranch) koje izvrše skok ovisno o stanju uvjetnog registra
- Npr. BCS <adresa> će ukoliko je **C**arry flag postavljena (**S**et) nastaviti izvršavati program od mjesta <adresa>, a ako nije, program će se nastaviti izvršavati od sljedeće instrukcije u programu

```
SKOK:
   MOVE.B #$F0, D0
   ADD.B #$20, D0
   BCS SKOK
```

- Što će se dogoditi u sljedećem programu, a što ako ADD.B zamijenimo sa ADD.W?

Ostale B?? instrukcije

- BCS skače ako je **C**arry bit postavljen (**S**et)
- BCC skače ako **C**arry bit nije postavljen (**C**lear)
- BEQ skače ako je Zerro bit postavljen (**EQ**ual)
- BNE skače ako Zerro bit nije postavljen (**N**ot **E**qual)
- BGE skače ako N i V zastavice imaju iste vrijednosti (**G**reater of **E**qual)
- BGT skače ako su N i V zastavice iste, a Z nije postavljena (**G**reater **T**han)
- BHI skače ako C i Z nisu postavljene (**HI**gher than)
- BLE skače ako su N i V različite i Z je postavljena (**L**ess or **E**qual)
- BLS skače ako je C ili Z postavljena (**L**ower or **S**ame)
- BLT skače ako su N i V različite (**L**ess **T**han)
- BMI skače ako je N postavljena (**MI**nus)
- BPL skače ako N nije postavljena (**PL**us)
- BVC skače ako V nije postavljena (**V C**lear)
- BVS skače ako je V postavljena (**V S**et)
- BRA skače uvijek (**BR**anch **A**lways)

Grananje izvršavanja programa

- Da bismo dobili dojam odakle dolaze imena spomenimo i instrukciju CMP
  - uspoređuje dva parametra (nakon nje, zastavice će biti postavljene jednako kao i kod SUB instrukcije, samo što će drugi parametar ostati nepromijenjen)
- Na primjer, ukoliko oba parametra imaju istu vrijednost, rezultat oduzimanja bi bio 0, pa je zbog toga Z bit zastavice postavljen, pa od toga dolazi ime BEQ (**B**ranch if **Eq**ual)

Grananje izvršavanja programa

- Budući da nemamo instrukcije slične drugim jezicima (if…then… else), npr. C-ovski kôd
  ```
  if (uvjet) then { neke_instrukcije }
  ```
  se ustvari shvati kao
  ```
  if (!uvjet) goto kraj_prvog_ifa;
  neke_instrukcije;
  kraj_prvog_ifa:
  ```
- `if (uvjet) then {inst1} else {instr2}`
  se shvaća kao
  ```
  if (!uvjet) goto else_prvog_ifa;
  instr1; goto kraj_prvog_ifa;
  else_prvog_ifa:
  instr2;
  kraj_prvog_ifa:
  ```

Grananje programa – provjera jednakosti

- `if (d0==d1) d2=1; else d2=0;`

```
  CMP.B D0, D1
  BNE razliciti
  MOVE.B #1, D2 ; isti su
  BRA kraj_prvog_ifa
razliciti:
  MOVE.B #0, D2 ; razliciti su
kraj_prvog_ifa:
```

```
  MOVE.B #$F0, D0
  MOVE.B #$F0, D1

  CMP.B D0, D1
  BEQ isti
  MOVE.B #0, D2 ; razliciti su
  BRA kraj_prvog_ifa
isti:
  MOVE.B #1, D2 ; isti su
kraj_prvog_ifa:
```

Grananje programa – petlje

- Uočimo da instrukcije za grananje mogu skočiti na proizvoljnu memoriju
  - Koristimo ih za realizaciju petlji
- Neki od registara je brojač
  - Brojimo prema nuli (SUB, BEQ)
  - Brojimo do nekog broja (ADD, CMP, BEQ)

Primjer: gr-v4-01.x68

Napomena: SUBQ zauzima manje memorije, jer kao prvi parametar prima samo konstantu do 3 bita

Grananje programa – ostale usporedbe

- Kod ostalih usporedbi (<, <=, >, >=) treba biti oprezan kakvi su nam parametri (da li ih shvaćamo kao signed ili kao unsigned brojeve). Nakon instrukcije CMP D0, D1, želimo provjeriti koji je manji:
  - ukoliko su unsigned i D1 je manji od D0, Carry bit će biti postavljen, pa za skok koristimo BCS
  - Ukoliko su signed i D1 je manji, dobit ćemo ili promjenu predznaka (V bit) ili negativan broj (N bit), pa tada koristimo BLT

Grananje programa – ostale usporedbe

- Kod ostalih usporedbi (<, <=, >, >=) treba biti oprezan kakvi su nam parametri (da li ih shvaćamo kao signed ili kao unsigned brojeve). Nakon instrukcije CMP D0, D1, koristimo sljedeće instrukcije za provjeru koji je bio veći:

CMP D0, D1	unsigned	signed
D1 < D0	BCS	BLT
D1 <= D0	BLS	BLE
D1 == D0	BEQ	BEQ
D1 != D0	BNE	BNE
D1 > D0	BHI	BGT
D1 >= D0	BCC, BPL, BVC	BGE, BMI, BVS

Instrukcija MOVEM

- Instrukcija MOVEM služi za brzi transfer više vrijednosti iz memorije u registre, ili obratno. Ona ne radi sa 8 bitnim brojevima, a u 16 bitnom obliku izmijeni svih 32 bita registra.
- Kao jedan parametar navodimo listu registara, a kao drugi memorijsku lokaciju odakle/gdje će se podaci izmijeniti. Npr.
  - MOVEM D0-D2, $1000 sprema redom vrijednosti od D0, D1 i D2 na lokacije $1000, $1002, i $1004
- MOVEM.L $2000,D0-D2/A1-A3/D5/D7 čita vrijednosti iz memorije u registre (zapisani su kao bitfield, pa redoslijed kojim ih pišemo nema veze), i to registara $2000->D0, $2004->D1, $2008->D2, $200C→D5…

Primjer: gr-v3_06.x68

Grananje programa – ostale usporedbe

Primjeri:

- gr-v4-02.x68 i gr-v4-03.x68
- gr-v4-05.x68

Testiranje jedne vrijednosti

- Za razliku od instrukcije CMP, koja postavlja zastavice uvjetnog registra kao da smo napravili instrukciju SUB, ukoliko želimo samo provjeriti stanje neke vrijednosti (npr. slično instrukciji MOVE, ali bez promjene vrijednosti), možemo koristiti instrukciju TST (ona prima jedan parametar, pa provjeri je li jednak nuli ili negativan)
- **Primjer:** gr-v4-04.x68