Informe de la pràctica #2

Introducció

En aquesta segona practica, com a la primera, s'han fet exercicis guiats pel professor per tal de assolir els seguents objectius:

- Familiaritzar-se amb el funcionament dels registres de la CPU, usant el simulador SiMR.
- Aprendre a localitzar i diferenciar els registres de proposit general, i els registres de proposit específic.

Exercicis Guiats

En el primer exercici es demana que es carregui un programa ja codificat al SiMR, i veure els resultats. Aquest programa consta de unes linies de codi bàsic que introdueïx un “loop”, es a dir, un bucle que es repeteix fins que es dona una condició, en aquest cas, que un comptador arribi a zero.

El programa en qüestió es aquest:

```
Dades: .dw 4, 5, 6, 7
.begin inici
inici:
       SUB R7, R7, R7
       LOAD Dades(R0), R1
       ADDI R7, #1, R7
       LOAD 0(R7), R2
       ADDI R7, #1, R7
       LOAD 0(R7), R3
       ADDI R7, #1, R7
       LOAD 0(R7), R4
loop:
       ADD R1, R2, R5
       ADD R3, R4, R6
       SUBI R3, #1, R3
       BG loop
.end
```

Els resultats que queden als registres al final del programa, quan surt del “loop” son:

R0:0000 *R1: 0004* *R2: 0005* *R3: 0000*

R4: 0007 *R5: 0009* *R6: 0008* *R7: 0003*

Al segon exercici es demana fer un programa similar al del exercici 1, amb un registre que es decrementa fins a arribar a 0, pero que faci altres operacions.

El codi realitzat es el següent:

```
dada1: .dw 3083
dada2: .dw 17
.begin inici
inici:
       LOAD dada1(R0), R1
       LOAD dada2(R0), R2
loop:
       ADD R7, R1, R7
       SUBBI R2, #1, R2
       BG loop
.end
```

Al final de la execució del programa, el cual s'executa tants cops com gran sigui R2 (perque el bucle es basa en decreixer R2 en 1 per cada iteracio, osigui 17), el resultat que queda als registres és:

R1: 0C0B *R2: 0000* *R3: CCBB*

Per últim, el tercer exercici demana calcular un algorisme que faci el següent: Donades dues entrades A i B, fer la comparació. Si A>B, calculem la suma. Si A<B fem la diferència (B-A) i si son iguals sortim del algorisme.

El codi realitzat és el següent:

```
A: .dw 50
B: .dw 100
.begin inici
inici:
       LOAD A(R0), R1
       LOAD B(R0), R2
loop:
       SUB R2, R1, R0
       BG 9
       BEQ 10
       ADD R1, R2, R3
       BR 10
       SUB R2, R1, R3
.end
```

Aquest programa un cop carrega les dades al registre i entra al “loop” fa una comparació, es a dir, resta R2-R1 i ho guarda a R0, que es inmodificable, aixi simplement podem treballar amb els bits de zero i negatiu de la operació anterior. En cas de que el resultat sigui positiu sabrem que R2>R1, per tant volem la resta i fem un BG(posicio) on es troba la resta. Si el resultat dona zero, ens interessa sortir del programa, per això fem un BEQ(posicio) ón posició és el final del programa. En la resta de casos, sabem que R2<R1, per tant deixem que passi fins a la suma i acabi el programa amb un salt incondicional al final de programa amb BR(posicio).

Si al final volem guardar el resultat de R3, fem una comanda STORE amb la posició de memòria a on volem guardar, en aquest cas 22h, que equival a 34 en decimal, per tant afegim la següent comanda abans del final de programa:

STORE R3, 34(R0)

Conclusions

En aquesta practica hem utilitzat i après els bucles i loops que permet el ensamblador, aixi com hem aprofundit en com funcionen els salts, tant condicionals com incondicionals.