Informe de la pràctica #1

Introducció

En aquesta primera practica de Introducció als Ordinadors, s'havien plantejat els següents objectius, segons el guió:

- Familiaritzar-se amb el simulador de la màquina rudimentària (SiMR).
- Entendre que fan les instruccions.
- Comprendre l'analogia entre llenguatge màquina i el llenguatge ensamblador.

1. Introducció teorica

En aquesta practica el que es pretenia fonamentalment era entendre i coneixer el simulador que fariem servir a la assignatura, el SiMR.

Hem vist les parts que conté la Màquina Rudimentaria (RA: acumulador, IR: registre d'estat, PC: comptador de programa, els seus 8 registres...) i la part del banc del programa, on carreguem les instruccions en ensamblador. També hem vist les instruccions que usem per a operar (**ADD, SUB, ADDI, SUBBI, ASR, BR...**) i hem aprés les directives, que no s'executen en temps de execució sino en temps de ensamblat (**.dw:** per asignar un contingut a una posició de memòria, **.rw:** per reservar n espais de memòria, **.begin:** directiva on indiquem que comença la memòria del programa i **.end:** on indiquem que acaba el programa).

Finalment, despres de veure exemples de format de programes, hem vist com compilar i executar un programa. La tasca de compilació es duu a terme amb el software POSTEN, que agafa un arxiu amb extensió *.asm* i el compila i passa a extensió *.COD*.

2. Instruccions del SiMR

En aquest apartat aprofundim i expliquem les instruccions del SiMR i de quin tipus hi hà. Podem trobar instruccions de tres tipus: instruccions aritmètico-lògiques, instruccions d'accés a memòria i instruccions de salt.

Dins les instruccions aitmètico-lògiques hi podem trobar per a sumar el contingut de dos registres (ADD), restar el contingut de dos registres (SUB), sumar o restar un numero inmediat al contingut de un registre (ADDI o SUBI) o desplaçar a la dreta o a la esquerra el contingut de un registre (ASR, que fà l'efecte de dividir entre 2 o multiplicar per 2).

A les instruccions d'accés a memòria trobem comandes per a accedir i carregar a la CPU el contingut de una adreça de memòria (LOAD) o be per a guardar el contingut de un registre a una adreça de memòria (STORE).

I finalment, a les instruccions de salt (BEQ: que salta si el bit Z esta a 1, BG: si la operació anterior dona un valor positiu, BL: si la operació anterior dona un valor negatiu i BLE o BLG: que salta si la operació anterior dona un valor igual a zero o negatiu en el primer cas o igual a zero o positiu en el segon).

3. Exercicis guiats

Hem realitzat diferents exercicis per a familiaritzarnos amb la sintàxi de les instruccions i per a entendre el procés de execució d'un programa.

En el primer exercici, hem vist una serie de instruccions i haviem de dir si eren correctes o incorrectes:

	Instruccions	Correcte/Incorrecte	Motiu
a	LOAD R1(R0), R1	incorrecte	
b	STORE R1,R1(3)	incorrecte	
c	BG 6(R1)	incorrecte	
d	ADDI R2, #11, 5(R3)	incorrecte	;ADDI R2, #11, R3
e	SUB R0,R2,R3	correcte	
f	LOAD 3(R0),R1	correcte	

En el següent exercici, hem vist una sèrie de instruccions i haviem de preveure els canvis qu es produíen en els registres i en la memòria en cada instrucció:

	Instruccions	R0	R1	R2	R3	N	Z	M[0Ch]	M[13h]	PC
a	LOAD 12(R0),R1	igual	F45Ah	igual	igual	1	0	igual	igual	+1
b	STORE R1,1(R3)	igual	igual	igual	igual	1	0	igual	igual	+1
c	BR 33	""	""	""	""	""	""	""	""	34
d	ADDI R2,#11,R3	""	""	""	R2+11	1	0	""	""	+1
e	SUB R0,R2,R3	""	""	""	R2	0	0	""	""	+1
f	ASR R1,R1	""	R1/2	""	""	0	0	""	""	+1

Finalment, hem fet un programa de prova, amb un seguit de directives i instruccions previament estudiades, per a veure el funcionament real del simulador.

```
.dw 3, 5, 2, 8
.rw 1
.begin inici
inici:
        LOAD R1(R0), R1
        STORE R1, R1(3)
        BG 6(R1)
        ADDI R2, #11, 5(R3)
        SUB R0, R2, R3
        LOAD 3(R0), R1
.end
```

En aquest petit programa, primer definim una serie de posicions de memòria amb la directiva .dw, i després guardem una posició de memòra amb la directiva .rw, tal que el nostre programa comença a la posició de memòria 5. En aquest simplement executem un seguit de instruccions per veure els canvis en els registres i la memòria, peró sense cap funció real.

Conclusions

- *S'han aprés els components i funcionament del simulador de la Màquina Rudimentaria.*
- *S'han explicat les instruccions/directives del simulador i la seva correcta sintaxi i funció.*
- *S'han realitzat exercicis proposats on hem vist i intentat preveure els estats de abans/després de les instruccions.*
- *S'ha aprés a compilar i executar un codi en ensamblador en el simulador SiMR, fent servir el POSTEN.*