Informe de la pràctica 1

Introducción

En este informe de la 1ª de la práctica de Introducció als Ordinadors encontraremos los objetivos, los ejercicios propuestos a clase y las conclusiones:

- Familiarizarse con el simulador de la máquina rudimentaria (SiMR) y el simulador Ripes
- Entender qué hacen las instrucciones
- Comprender la analogía entre lenguaje máquina y lenguaje ensamblador

Exercicis guiats

1. Indiqueu quines de les següents instruccions són incorrectes segons les especificacions de la Màquina Rudimentària (MR), expliqueu el perquè en cada cas:

	Instruccions	Correcte/ Incorrecte	Motiu
a	LOAD R1(R0), R1	Incorrecto	La dirección base de la dirección de memoria tiene que ser un número entre 0 y 255, no un registro. Sería por ejemplo LOAD 0(R0), R1
b	STORE R1,R1(3)	Incorrecto	La dirección de destino es una dirección de memoria con la forma de (número)Registro STORE R1, (3)R1
c	BG 6(R1)	Incorrecto	Las instrucciones de salto solo aceptan un número, no direcciones de memoria. BG 6
d	ADDI R2, #11, 5(R3)	Incorrecto	Este tipo de instrucción aritmético-lógica nunc van directamente a la memoria principal, debería haber un registro en la última comanda (tenemos un registro fuente y un inmediato y se guardar en un registro no en una dirección de memoria) ADDI R1, #11, R3
e	SUB R0,R2,R3	Correcto	Suma R0 y R2 y el resultado se guarda en

			R3
f	LOAD 3(R0),R1	Correcto	Carga la posición de memoria 03h en R1

2. Suposeu que la màquina rudimentària té els següents valors en alguns registres i posicions de memòria (recordeu que amb la lletra "h" s'indica que el número està en format hexadecimal):

-Registres: R0=0000h, R1=0002h, R2=A5E3h, R3=F412h

-Bits de condició (recordeu que N és el bit de Negatiu, Z és el bit de resultat igual a zero): N=0, Z=1

-Memòria: M[0Ch]=F45Ah i M[0Dh]=0033h

Indiqueu què fa cadascuna de les instruccions següents, cal explicitar totes les alteracions que es produeixen a la memòria, bits de condició, registres i valor final. Per cada una de les instruccions sempre partiu de les condicions inicials descrites més a dalt.

	Instruccions	R0	R1	R2	R3	N	Z	M[0Ch]	M[13h]	PC
a	LOAD 12(R0),R1	0000h	F45Ah	A5E3h	F412h	1	0	F45Ah	0033h	2
b	STORE R1,1(R3)	0000h	F45Ah	A5E3h	F413h	0	0	F45Ah	F45Ah	3
c	BR 33	0000h	F45Ah	A5E3h	F413h	0	0	F45Ah	F45Ah	33
d	ADDI R2,#11,R3	0000h	F45Ah	A5E3h	A5EEh	1	0	F45Ah	F45Ah	34
e	SUB R0,R2,R3	0000h	F45Ah	A5E3h	5A1Dh	0	0	F45Ah	F45Ah	35
f	ASR R1,R1	0000h	7A3Dh	A5E3h	5A1Dh	0	0	F45Ah	F45Ah	36

Breu descripció del que fa cada instrucció:

a. Carga el contenido de la dirección de memoria 12(R0) = C(R0), es decir, 0Ch + 0 = 12 + 0 = 12 = 0Ch, al registro R1.

b. Guarda el contenido del registro R1 en la dirección de memoria 1(R3) obteniendo F413h.

c. Salto incondicional. Ponemos el PC, contador del programa, a 33.

d. Suma el contenido del registro R2 y el inmediato decimal 11 (A5E3h + 11 = A5EEh) para después guardar el resultado en el registro R3.

e. Resta los registros R0 y R2 y guarda el resultado, un negativo (0000h – A5E3h = 5A1Dh), en R3.

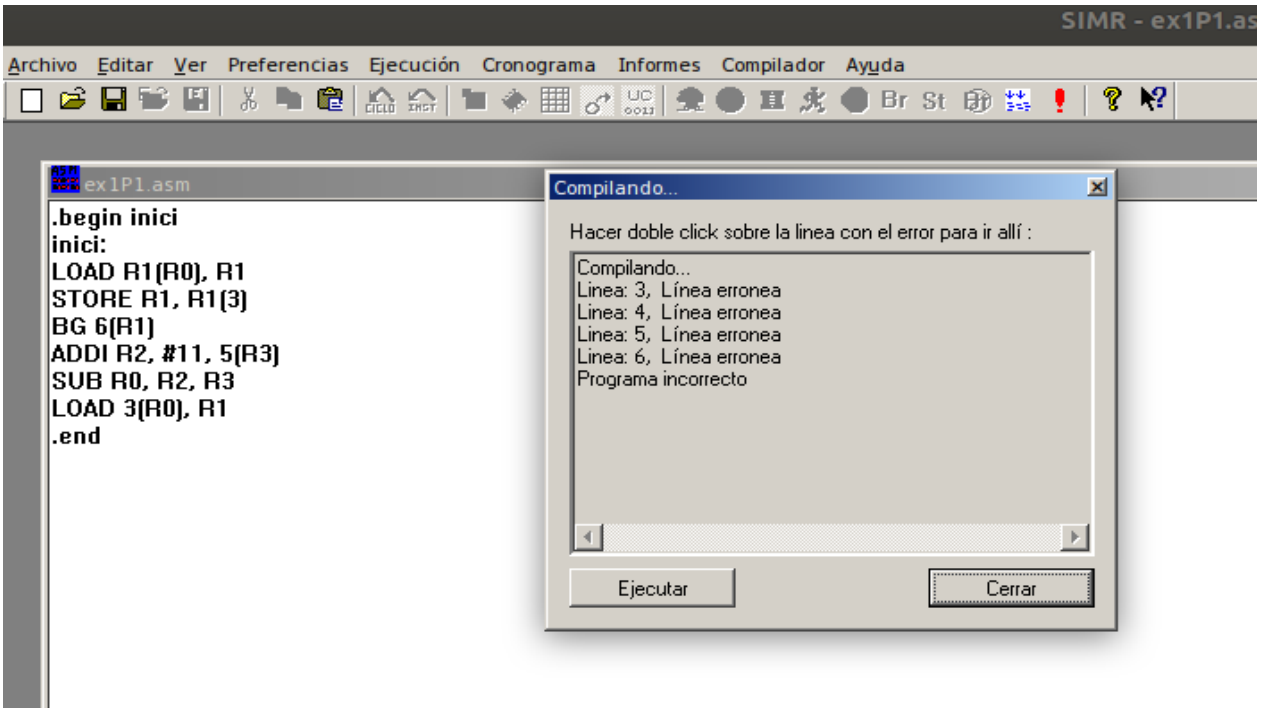
f. Desplazamiento hacia la derecha de un bit del contenido del registro R1 y lo guarda en el mismo registro.

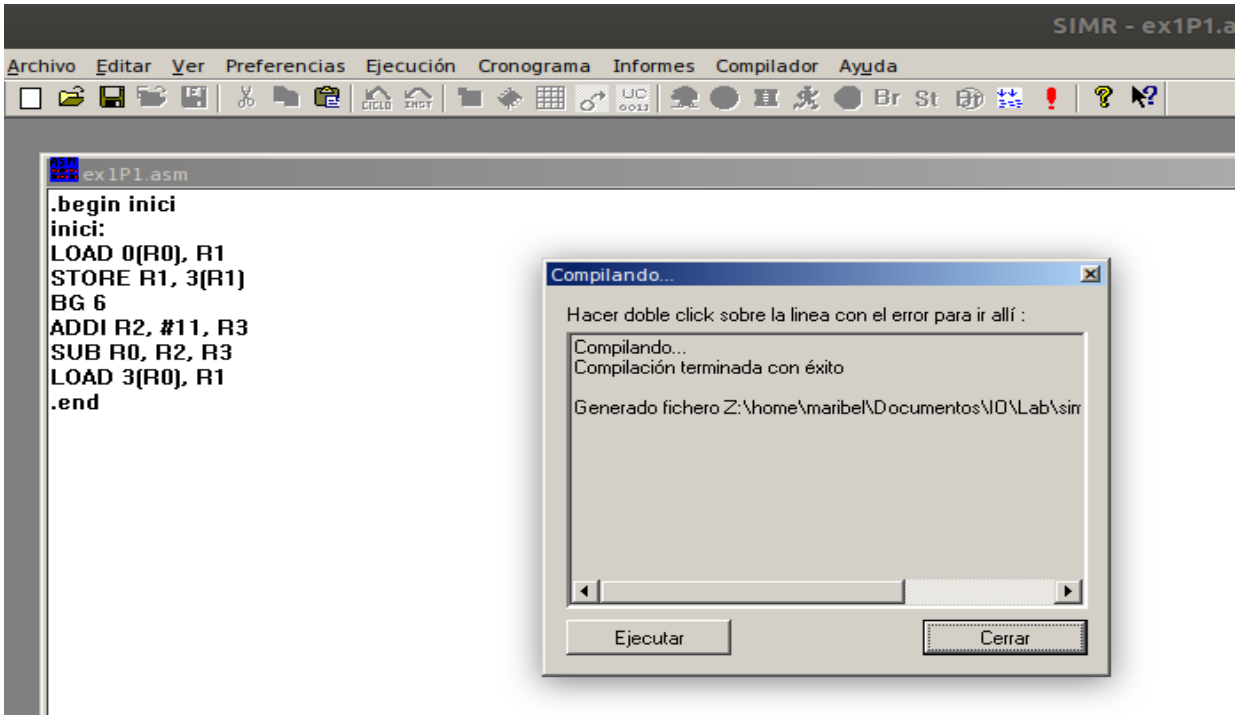
3. Escriviu en llenguatge màquina (segons les especificacions de la MR) el següent programa, feu-ho en hexadecimal i binari:

@	M[@]	LLENGUATGE MÀQUINA	LLENG. MAQ. (hex)
05 h	SUB R1,R1,R1	11 001 001 001 00 101	C925h
06 h	ADD R0,R0,R2	11 010 000 000 00 100	D004h
07 h	SUBI R1, #4, R0	11 000 001 00100 001	C121h
08 h	BEQ 13	10 001 000 00001101	880Dh
09 h	LOAD 0(R1),R3	00 011 001 00000000	1900h
0A h	ADD R3,R2,R2	11 010 011 010 00 100	D344h
0B h	ADDI R1,#1,R1	11 001 001 00001 000	C908h
0C h	BR 07	10 000 000 00000111	8007h
0D h	STORE R2, 4(R0)	01 010 000 00000100	5004h

Realització Pràctica Guiada

1. Escriure les instruccions de l'apartat 1 de l'estudi previ en un programa al SiMR (no cal que implementi cap funcionalitat) i intenteu compilar-lo. Podreu comprovar com les instruccions incorrectes us donen error alhora de compilar. Feu les modificacions adequades per a que no us doni cap error en la compilació (recordeu que no es demana cap funcionalitat).





2. Pel programa de l'apartat 3 de l'estudi guiat, les posicions de memòria que van des de la 00h a la
03h, ambdues incloses, contenen una seqüència del quatre números 3, 5, 2, 8, emmagatzemats de
forma consecutiva. Aquesta seqüència de valors no està especificada al programa anterior. La
posició 00h conte el primer element, la posició 01h el segon element, etc.. El programa descrit
anteriorment es troba emmagatzemat a partir de la posició de memòria 05h.

Partint d'aquesta descripció responeu a les següents preguntes:
a) En quina posició de memòria escriu aquest programa el resultat?
En la posición 04h + 00h = 04h

b) Què fa el programa?
Primero pone a 0 los registros R1 y R2. Luego resta 04h al contenido del registro R1, que es 0, para guardarlo en el registro R0 (no se modifica nada).
Si en la operación anterior (SUBI, operación aritmético-lógica) se activa el bit de condición Z en los registros, saltaremos a la posición de memoria 13, es decir, 0Ch en hexadecimal. En caso contrario, no saltará. Lo siguiente será cargar el contenido de la dirección de memoria 0(R1) al registro R3 y sumar los contenidos de R3 y R2, que se guardarán en R2.
Volvemos a sumar otra vez. En este caso será el contenido de R1 más 01h y se guardará en el mismo registro. Ahora saltamos a la posición 7 y, por último guardamos el contenido de R2 en la posición de memoria 04h.

c) Quina sentència de control de flux (en llenguatge d'alt nivell) implementa el programa?
Tenemos un control SUBI R1, #4, R0 y BEQ 13, es decir, tenemos un bucle que hace una iteración que se repite 4 veces, hasta salir de dicho bucle. En lenguaje de alto nivel sería equivalente a un bucle de la forma: while variable <= 4 o for (variable = 0; variables <= 4; variable++).

d) Quina és la funció d'R1 al programa? I la d'R0?
La función de R1 es ser un contador y un control, es decir, nos ayuda a iterar sobre las posiciones de memoria que necesitamos para sumar. En cuanto a la función de control, sirve para terminar el bucle.

La función de R0 es inicializar valores y acceder a posiciones de memoria, ya que es invariable en 0.

Recordeu que aquest és el programa:
@ M[@]
05h SUB R1,R1,R1

06h ADD R0,R0,R2
07h SUBI R1, #4,R0
08h BEQ 13
09h LOAD 0(R1),R3
0Ah ADD R3,R2,R2
0Bh ADDI R1,#1,R1
0Ch BR 07
0Dh STORE R2,4(R0)

3. Realitzeu les següents accions sobre el Simulador de la màquina rudimentària (SiMR) i
expliqueu-ne els resultats:
e) Arranqueu el simulador de la MR y carregueu el programa pract1.asm. Compileu-lo. Carregueu
ara el programa pract1.cod. Estabiu una relació entre el codi assemblador del programa presentat
pel simulador i el de l'apartat anterior.
Los 2 programas son similares. En ambos se suman todos los elementos a través de un bucle, dando como resultado dicha suma.

f) Observeu que inicialment PC=05h (el PC apunta a la primera instrucció executable del
programa). Esbrineu com ho fa.
La primera instrucción ejecutable a la que apunta el PC es 05h porque considera que las 5 primeras con de la forma: LOAD 0X(R0), R0 para X=0,1,2,3,4. Por las propiedades el registro R0 (se mantiene siempre a cero), la primera instrucción ejecutable será la 05h.

g) Executeu el programa, instrucció per instrucció, observant el resultat d'executar cada una de les instruccions:

-Abans d'executar cada instrucció prediu les variacions que es produiran al banc de registres i a la memòria.

-Comproveu que el resultar de l'execució del programa coincideix amb el que havíeu previst.

-Per cada instruccions anoteu tots els canvis.

SUB R1,R1,R1

Esta instrucción resta el contenido de R1 a R1 y lo guarda en el mismo registro, quedando a 0 però como ya estaba a cero no varía.

ADD R0,R0,R2

Esta instrucción suma el contenido de R0 a R0 y lo guarda en R2, però como en el anterior, al sumar 0 y 0 da 0 y se guarda en un registro que inicialmente ya estaba a 0 (inicialización a 0).

SUBI R1,#4,R0

Esta instrucción resta el inmediato 4 al contenido de R1 (que es cero) y lo guarda R0. Como R0 siempre se mantiene a cero no se detectan alguna variación en el banco de registros. Sin embargo, como hemos empezado un bucle si notamos cambios en el Registro de Aritmético (pasa de 0000, a 0001 y así hasta llegar a 0004, que es cuando se alcanza la condición de fin)

BEQ 13

Esta instrucción nos indica la condición de que el valor del último resultado es igual a 0, es decir, sufrirá un cambio cuando el bit de cero del Registro de Estado se ponga a 1 y saltará a la condición de fin de bucle.

LOAD 0(R1),R3

Esta instrucción carga una posición de memoria a un registro, siendo la primera posición 00h, luego 01h y así hasta 03h.

ADD R3,R2,R2

Esta instrucción suma el contenido de R3 a R2 y lo guarda en R2, produciendo cambios en el Registro Aritmético y en el Registro de Instrucciones.

ADDI R1,#1,R1

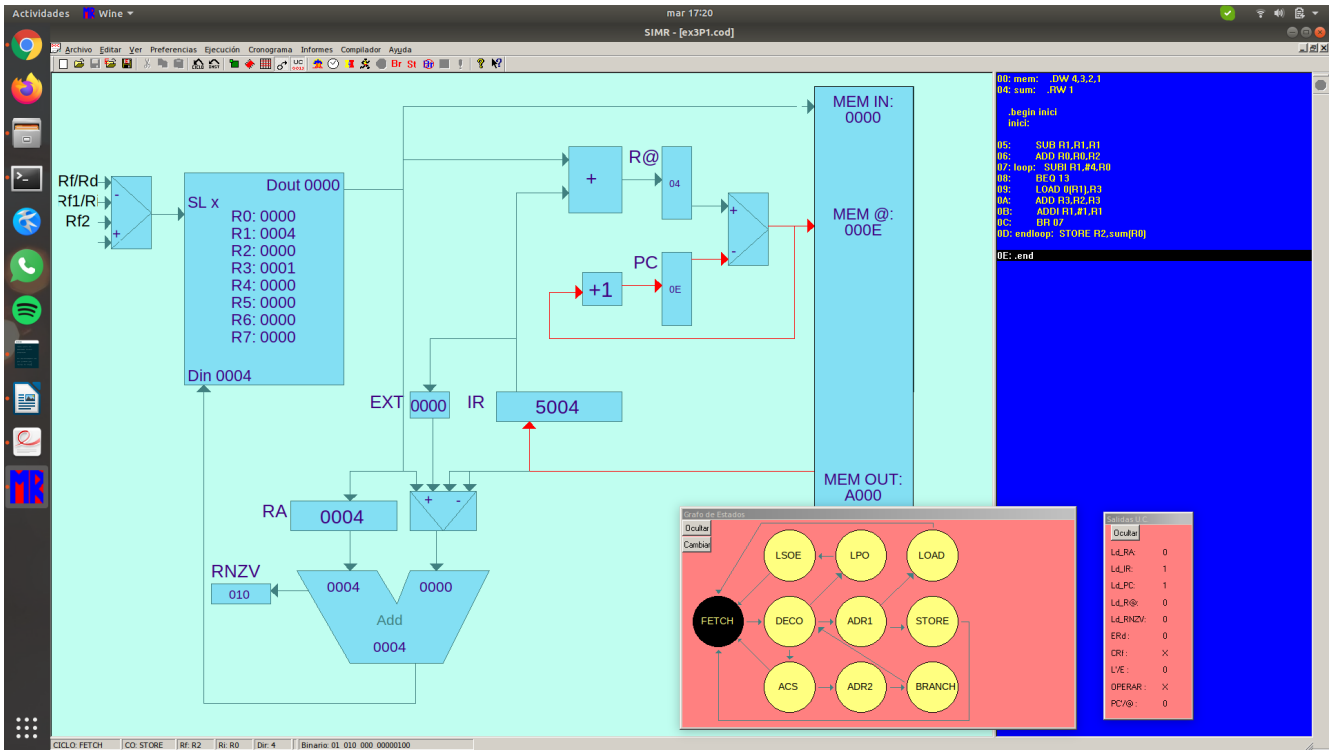
Esta instrucción incrementa en 1 el contenido de R1 y lo guarda en el mismo registro, es decir, inicialmente está a 0000 y va aumentando a 0001, 0002, 003 y por último 0004.

BR 07

Esta instrucción realiza un salto incondicional a la posición de memoria 07h.

STORE R2,4(R0)

Una vez acabado el bucle, esta instrucción guarda el contenido de R2 en la posición de memoria 4(R0), es decir 04h + 00h = 04h y queda guardo 010 = 0Ah.



Treball a fer a casa

1.- Obre l'editor de text predefinit i escriu el següent programa:

2.- Guarda el programa amb el nom Practica1_riscv.asm (sobre tot és important l'extensió). Obre el simulador Ripes. Posiciona a la part de l'editor.

3.- Clica File i selecciona l'opció Load Assembly File. Veureu el vostre codi a la part esquerra de la

pantalla (source code) i el codi desensamblat a la part dreta.

4.- Clica a la part del processador

5.- Executa el programa pas a pas. El programa que s'executarà és el desensamblat. El que tens a la figura superior. La primera instrucció el que fa es incrementar el registre PC per carregar la instrucció des de la memòria de programa. El processador amb el que esteu simulant és un processador amb 5 pipelines. Això vol dir que, en un cicle de rellotge està executant 5 fases de 5 instruccions diferents. Analitza el comportament del programa.

- Què fa? Reescriu el codi (sempre que es pugui) perquè s'executi en el SiMR.

  Carga primero unas direcciones de memoria para luego guardar el inmediato que corresponda (0,1,4,etc). Después entra en un bucle en el que compara los registro a0 y a1, restando en una unidad el contenido de a1 mientras no sean iguales. Una vez lo sean, salta fuera del bucle a un load word en a3.

- Quines són les directives utilitzades en Ripes? Compara-les amb les de SiMR.

  Los dataByteX y dataX corresponden en SIMR a array: .DW a,b,c,d (siendo a,b,c,d las variables que guardaríamos en cada data).

  lw y lb corresponden a la instrucción LOAD en SIMR, aunque en SIMR cargamos una posición de memoria a un registro y en Ripes primero va el registro y luego la memoria.

  sw corresponde a la instrucción STORE en SIMR y son bastante similares: guardamos el contenido de un registro a una posición de memoria.

  beq presenta la misma nomenclatura en ambos simuladores, como la mayoría de instrucciones de salto. Sin embargo, en Ripes se especifica los registros a comparar y el salto y en SIMR, solo la posición de memoria a donde saltar si se cumple la condición.

  sub también presenta la misma nomenclatura y, además, la misma sintaxis, igual que el loop o el end. Aún así en Ripes hay un main, mientras que en SIMR hacemos un .begin inici.

- Troba les similituds i diferències entre les instruccions emprades en SiMR i les instruccions emprades en Ripes. Feu els diferents programes implementats en SiMR amb Ripes.

_____Además de las diferencias y similitudes comentadas en el apartado anterior, el PC funciona de forma distinta en los 2 simuladores: en SIMR se actualiza automáticamente en la máquina y en Ripes utiliza el auipc (con un rd y un offset).

En cuanto a las instrucciones LOAD y STORE de SIMR, en Ripes se subdividen en varios tipos, según si cargan/guardan un byte(8 bits), half(16 bits), word (32 bits), double (64 bits) etc.

Conclusiones

En este apartado enumeramos las principales aptitudes y conocimientos aprendidos con esta práctica:

- *Introducción al simulador de la máquina rudimentaria SIMR.*
- *Introducción al lenguaje ensamblador y al código máquina*
- *Introducción al simulador Ripes.*