Práctica 3 - Arquitectura del CPU

Organización del Computador 1

Segundo Cuatrimestre 2020

Ejercicio 1 leftShift es una rutina que decala a izquierda. Recibe como parámetros el valor a shiftear en el registro R0 y la cantidad de posiciones en el registro R1 (entendido como un número entero sin signo). El resultado final queda en el registro R0.

Se pide:

a) Escribir el *pseudocódigo* del programa leftShift.

b) Escribir leftShift en lenguaje *assembler* de ORGA1 (tener en cuenta las instrucción de retorno, `RET`).

c) Aparte de R0 y R1, ¿la rutina altera el valor de otros registros? Modifique el programa de modo que preserve el valor de todos los registros salvo R0 y R1.

Ejercicio 2 Escribir el *pseudocódigo* y el programa en *assembler* de ORGA1 que recorra un vector de números codificados en notación complemento a 2 de 16 bits de precisión y devuelva el valor máximo y mínimo de dicho vector. La posición de inicio del vector es recibida en el registro R0 y su longitud en el registro R1. En los registros R2 y R3 deben guardarse el resultado del máximo y mínimo respectivamente. La solución debe recorrer el vector una única vez.

Ejercicio 3 La arquitectura ORGA1 posee operaciones aritméticas sobre números enteros codificados en notación *complemento a 2* de *16 bits* de precisión.

El programa sumar64 realiza la suma en notación *complemento a 2* de dos números enteros de *64 bits* en esta arquitectura. En los registros R0 y R1 se indican las direcciones de cada número y en R2 se indica la posición de memoria en donde debe guardarse el resultado. Tanto los parámetros como el resultado se almacenan en orden *little-endian*.

Se pide:

a) Escribir el *pseudocódigo* del programa sumar64.

b) Escribir sumar64 en código *assembler* de ORGA1. Pista: revisar la operación ADDC de la máquina ORGA1.

Ejercicio 4 sumaVector64 es una rutina que suma los valores de un vector de n posiciones de enteros de *64 bits* codificados en notación *complemento a 2*. En R0 se recibe la cantidad de elementos que tiene el vector y en R1 la posición de memoria en donde está almacenado dicho vector. En R3 se recibe la posición de memoria en donde debe guardarse el resultado. Los elementos del vector y el resultado se almacenan en orden *little-endian*.

Suponiendo que se cuenta con el programa sumar64 descripto en el ejercicio anterior, se pide:

a) Escribir el *pseudocódigo* del programa sumaVector64.

b) Escribir sumaVector64 en código *assembler* de ORGA1 (tener en cuenta las instrucciones de invocación, `CALL`, y de retorno, `RET`).

Ejercicio 5 Traducir el siguiente programa de lenguaje C a lenguaje de la máquina ORGA1. Usar los registros R0 y R1 como contenedores de las variables `x` e `y` respectivamente;

```
1 int x = 2;
2 int y = 32;
3 x = x + y;
```

Ejercicio 6 Desensamblar el siguiente programa escrito en lenguaje de máquina ORGA1.

```
0001 1000 0000 0000 0000 0000 1111 1111
0001 1000 0100 0000 0001 0000 0000 0000
0010 1000 0010 0001
```

Ejercicio 7 Dados los siguientes valores de la memoria y del registro R0 de la arquitectura ORGA1, ¿qué valores cargan en el registro R1 las siguientes instrucciones?

a) `MOV R1, 20` (inmediato)

b) `MOV R1, [20]` (directo)

c) `MOV R1, [[20]]` (indirecto)

d) `MOV R1, R0` (registro)

e) `MOV R1, [R0]` (indirecto registro)

f) `MOV R1, [R0 + 20]` (indexado registro)

Memoria

Dirección	Contenido
0x0020	0040h
0x0030	0050h
0x0040	0060h
0x0050	0070h

Registros

Nombre	Contenido
R0	0030h

Ejercicio 8 A partir de cada uno de los siguientes vuelcos parciales de memoria y estados del procesador, realizar un seguimiento simulando ciclos de instrucción hasta encontrar una instrucción inválida (es decir, una instrucción cuyo código de operación no figure entre los del conjunto de instrucciones). Consultar la arquitectura de la máquina ORGA1 en el apunte provisto por la materia. Indicar qué celdas de memoria se modifican y el estado final del procesador.

a) *Ayuda*: la quinta instrucción a ejecutar es inválida.

PC	R0	035D	R1	034C	R2	8A6B	R3	0300	Z	0	N	0
A644	R4	E7A0	R5	A622	R6	1A2B	R7	3898	C	0	V	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
A630	2EFD	0053	0225	0000	0052	C925	F303	1A2B	5600	5B4C	A4D6	88C0	0000	A200	034B	A631
A640	FEDC	7081	A1C1	C925	223D	A639	002B	FAF0	48A7	6880	9C40	0000	0348	4AF3	A000	0340

b) *Ayuda*: la quinta instrucción a ejecutar es inválida.

PC	R0	035D	R1	034C	R2	8000	R3	A622	Z	0	N	0
2532	R4	E7A0	R5	2521	R6	1A2B	R7	FFF4	C	0	V	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
2520	F303	ABCD	2FE2	2530	8000	253E	FEDC	7081	A1C1	C925	223D	A639	002B	FAF0	48A7	6880
2530	034B	A631	5D40	5432	F205	A200	253C	2EFD	0053	0225	9200	2524	2624	2525	0000	9C40

c) *Ayuda*: la cuarta instrucción a ejecutar es inválida.

PC	R0	035F	R1	C2A3	R2	2940	R3	034F	Z	0	N	0
0358	R4	27A3	R5	0122	R6	0352	R7	0359	C	0	V	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0340	FAF0	0053	0000	8EC0	0349	6925	0340	1A2B	0350	6023	0000	0000	A000	C925	034B	5E6F
0350	88C0	FFFF	A1C1	A200	0348	0000	0225	5200	3621	0344	FBEE	2940	FFFF	563B	FFFF	F303

Ejercicio 9 Considere el siguiente programa escrito en *assembler* de la arquitectura ORGA1.

```
          JMP Inicio
  Valor:  DW 0A0A
 Inicio:  MOV R0, [Valor]
          ADD R0, 0003
          MOV [Valor], R0
 Salida:  RET
```

a) Suponiendo que el programa se ubica en la posición 0xFF0E de la memoria, indicar el valor de cada una de las etiquetas.

b) ¿Qué diferencia habría si se reemplaza la tercera instrucción por "`Inicio: MOV R0, Valor`"?

Ejercicio 10 Dado el siguiente programa para la arquitectura ORGA1 cargado en la dirección de memoria 0x0000.

```
inicio:  MOV  R1, [[once]]
         ADD  [R1], 0x479E
         CMP  R1, R2
         DW   0x0007
         DW   0xFFEF
rutina:  JVS  fin
         SUB  R1, R2
fin:     RET
once:    DW   0x000B
cuatro:  DW   0x0004
```

Si se empieza a ejecutar con los registros del procesador en su estado inicial (R0 ... R7, PC y *flags* en *0*, SP en 0xFFEF) y asumiendo que la ejecución termina al intentar ejecutar una instrucción inválida, responder:

a) ¿Cuál es la dirección de memoria denotada por cada etiqueta?

b) Realizar el seguimiento del programa indicando el estado de los registros, *flags* y valores del *stack* en cada paso. Liste en orden las instrucciones ejecutadas.

Ejercicio 11 Considere el siguiente programa escrito en *assembler* de la arquitectura ORGA1:

```
  Vector:  DW ...
           DW ...
           ...
           DW 0000
   Clave:  DW ...
Comienzo:  MOV R0, Vector
           MOV R1, [Clave]
           MOV R2, 0
   Ciclo:  MOV R3, [R0]
           CMP R3, 0
           JE Fin
           CMP R3, R1
           JE Sumo
    Sigo:  ADD R0, 1
           JMP Ciclo
    Sumo:  ADD R2, 1
           JMP Sigo
     Fin:  RET
```

a) El programa se ubica en la posición 0x0100 de la memoria. Los vectores se almacenan como valores contiguos de una palabra de tamaño. Para indicar la finalización del vector se utiliza el valor cero. Suponiendo que el vector contiene diez palabras, indicar el valor de cada una de las etiquetas.

b) Explicar qué hace el programa siguiendo su ejecución paso a paso.

c) Escribir un *pseudocódigo* que refleje el comportamiento del programa.

d) Mostrar un vector para el cual este programa no hace lo esperado.