Trabajo práctico N° 7

Programación de la MCBE

Introducción a la computación
Departamento de Ingeniería de Computadoras
Facultad de Informática - Universidad Nacional del Comahue

Objetivo: Comprender cómo realizar una traza de un programa en lenguaje ensamblador, y el proceso de ensamblado.

Recursos bibliográfico:

- *Andrew S. Tanenbaum.* Organización de computadoras: un enfoque estructurado. Cuarta edición, editorial Pearson Educación, 2000. ISBN 970-170-399-5.

Lectura propuesta:

- Apunte de la materia. *Capítulo 7: El software. Sección 7.1 y 7.2* . Disponible en: `https://ic.fi.uncoma.edu.ar/uploads/misc/apunte_ic_2026.pdf`

Modelo Computacional Binario Elemental (MCBE)

1. Con respecto a la memoria de la *MCBE*, indique:

   a) Cantidad de celdas de memoria.

   b) Tamaño total en *bytes* de la memoria.

   c) ¿A que direcciones de memoria hacen referencia las etiquetas *IN* y *OUT*? ¿Qué funcion cumple cada una?

2. Para los programas que se describen a continuación realice una traza del programa e intente identificar que tarea realiza el programa. Busque una explicación de alto nivel (por ejemplo: *el programa pide ingresar un número y lo multiplica por 3*, o *el programa imprime los números del 5 al 1*).

   a)

#	Rotulo	Mnemónico/Dato	Argumento
0		LD	IN
1		SUB	DATO
2		SUB	DATO
3		SUB	DATO
4		ST	OUT
5		HLT	
6	DATO:	2	

b)

#	Rotulo	Mnemónico/Dato	Argumento
0	SALTO2:	LD	DATO
1		JZ	SALTO1
2		LD	DATO2
3		ST	OUT
4		LD	DATO
5		SUB	UNO
6		ST	DATO
7		JMP	SALTO2
8	SALTO1:	HLT	
9	DATO:	5	
10	UNO:	1	
11	DATO2:	8	

3. Para los programas que se describen a continuación:

   a) Realice una traza del programa e identifique la tarea que realiza el programa.

   b) Completar la columna de Contenido binario con la codificación en *Lenguaje Máquina* de cada instrucción.

1)

#	Contenido Binario	Rotulo	Mnemónico/Dato	Argumento
0			LD	IN
1			ST	DATO
2			LD	IN
3			ADD	DATO
4			ST	OUT
5			HLT	
6		DATO:	0	

2)

#	Contenido Binario	Rotulo	Mnemónico/Dato	Argumento
0			LD	IN
1			SUB	DATO1
2			ST	OUT
3			ADD	DATO2
4			HLT	
5		DATO1:	4	
6		DATO2:	9	

3)

#	Contenido Binario	Rotulo	Mnemónico/Dato	Argumento
0			LD	IN
1			ST	DATO
2			ADD	DATO
3			ADD	DATO
4			ADD	DATO
5			ADD	DATO
6			SUB	DATO2
7			ST	OUT
8			HLT	
9		DATO:	0	
10		DATO2:	3	

4. Dado el siguiente programa:

#	Rotulo	Mnemónico/Dato	Argumento	Contenido Binario
0		LD	IN	0101 1110
1	SIGUE:	JZ	FIN	1110 0100
2		ST	OUT	0111 1111
3		SUB	UNO	1010 0110
4		JMP	SIGUE	1101 1101
5	FIN:	HLP		0010 0000
6	UNO:	1		0000 0001

   a) ¿Qué tarea realiza el programa?

   b) ¿Qué modificación sería necesaria realizarle al código ensamblador para que el programa imprima cada número dos veces? ¿Se modifican la misma cantidad de celdas en el programa en código maquina?

5. Dado el siguiente programa:

#	Rotulo	Mnemónico/Dato	Argumento
0		LD	IN
1		ST	NUM
2	SIGUE:	LD	CANT
3		JZ	IMPRIME
4		SUB	UNO
5		ST	CANT
6		LD	NUM
7		SUB	DOS
8		ST	NUM
9		JMP	SIGUE
10	IMPRIME:	LD	NUM
11		ST	OUT
12	FIN:	HLT	
13	TEST:	-64	
14	NUM:	0	
15	CANT:	5	
16	UNO:	1	
17	DOS:	2	

   a) ¿Qué tarea realiza el programa?

b) ¿Cual sera el comportamiento del programa si se cambia la instrucción en la dirección nueve por **JMP TEST**?

6. Analice y responda las siguientes preguntas, justificando su respuesta:

   a) Supongamos que hemos almacenado en la posición 14 un dato numérico que representa la edad de una persona. ¿Qué pasa si en algún momento de la ejecución el *PC* contiene el número 14? ¿Qué pasará si esa persona tiene 33 años? ¿Qué pasará si tiene 65?

   b) ¿Qué pasa si el programa no contiene una instrucción *HLT*?

   c) Un programa ¿Puede modificarse a sí mismo? ¿Esto es útil? ¿Conveniente? ¿Peligroso?

   d) ¿Podría aumentarse la capacidad de memoria de la *MCBE*? ¿Esto requeriría algún cambio adicional a la máquina?

   e) ¿A qué distancia máxima puede *saltar* el control del programa en MCBE?

Conjunto de instrucciones de la MCBE

1. ¿Cómo se podría aumentar la cantidad de instrucciones diferentes de el *MCBE*? ¿Esto tendría algún efecto sobre la longitud de los programas que puede ejecutar la máquina?

2. ¿Puede el *MCBE* encontrar una instrucción que no sea capaz de decodificar?

Anexo

Op.	Código de operación *3 bits*	Operando *5 bits*	Descripción
`LD`	010	*dirección*	**Memoria → Acumulador**. Copia un byte desde la dirección de memoria al acumulador.
`ST`	011	*dirección*	**Acumulador → Memoria**. Copia el contenido del acumulador en esa dirección de memoria.
`ADD`	100	*dirección*	**Suma**. El contenido de la dirección se suma al acumulador, y el resultado se almacena en el acumulador.
`SUB`	101	*dirección*	**Resta**. El contenido de la dirección se resta al acumulador, y el resultado se almacena en el acumulador.
`JMP`	110	*desplazamiento*	**Salto incondicional**. Se suma (en complemento a 2) el desplazamiento al **PC**.
`JZ`	111	*desplazamiento*	**Salto condicional**. Si el acumulador es cero, se suma (en complemento a 2) el desplazamiento al **PC**, en caso contrario el **PC** se incrementa en uno.
`HLT`	001	*(sin uso)*	**Detiene la maquina**. No se ejecutan nuevas instrucciones. Los registros y la memoria quedan con el último valor que tenían.
`NOP`	000	*(sin uso)*	**No operación**. No tiene ningún efecto sobre el acumulador ni memoria. El **PC** se incremente en uno.

Memoria: consta de 32 posiciones de 8 bits. Las direcciones 0 a 29 corresponden a direcciones que pueden ser escritas y leídas. La dirección 30 es de **sólo lectura**, permite leer datos del dispositivo de entrada, por ejemplo un teclado. La dirección 31 es de **sólo escritura**, permite escribir datos en el dispositivo de salida, por ejemplo en una pantalla o una impresora.

Registro PC: registro de 8 bits, contiene la dirección de la próxima instrucción a ejecutar. Se inicializa en cero.

Registro IR: registro 8 bits donde se guarda la instrucción que se esta decodificando o ejecutando.

Registro acumulador: registro de 8 bits donde se almacena un número entero representado en *complemento a 2*.

Etiquetas predefinidas:

IN: dirección 30, entrada, dirección de solo lectura.

OUT: dirección 31, salida, dirección de solo escritura.

Instrucciones: de 8 bits, los 3 bits más significativos almacenan el código de operación, y los 5 menos significativos almacenan el operando.