Trabajo práctico N° 9

Programación de la MCBE, traducción y ejecución

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

Objetivo: Fortalecer los conceptos vistos sobre la MCBE, en particular el vínculo entre lenguaje ensamblador y código máquina.

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`

Programación en lenguaje ensamblador MCBE

1. Sea un programa que lee un número desde la entrada, y muestra por pantalla el doble de ese número.

   a) Escriba el programa indicado en lenguaje ensamblador *MCBE*.

   b) Traduzca el programa desarrollado en el punto anterior a código máquina.

   c) Realice la traza del del código creado en el paso anterior.

   d) Modifique el programa en ensamblador para que se imprima el cuádruple del número ingresado, y traduzca el nuevo programa a código máquina. ¿Cuántas lineas se modificaron en lenguaje ensamblador? ¿Cuántas lineas se vieron modificadas en el lenguaje máquina?

2. Sea un programa que lee dos números desde la entrada e imprime el primer número la cantidad de veces indicada por el segundo.

   a) Escriba el programa indicado en lenguaje ensamblador *MCBE*.

   b) Traduzca el programa desarrollado en el punto anterior a código máquina.

   c) Realice la traza del código creado en el paso anterior.

3. Sea un programa que lee dos números desde la entrada y muestra por pantalla el resultado de su multiplicación.

   a) Escriba el programa indicado en lenguaje ensamblador *MCBE*.

   b) Traduzca el programa desarrollado en el punto anterior a código máquina.

   c) Realice la traza del código creado en el paso anterior.

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.