Organización de Computadoras 2025

Práctica 4

Assembly, Instrucciones y Simulador MSX88

Objetivos de la práctica: que el alumno domine

- ***Las instrucciones básicas del lenguaje assembly del simulador MSX88.***
- ***Los diferentes modos de direccionamiento.***
- ***El diseño de programas utilizando instrucciones de salto condicional.***
- ***Realice el diseño de programas utilizando instrucciones del MSX88.***
- ***Comprenda la utilidad y funcionamiento de las subrutinas.***

Bibliografía:

- ***Apunte 4 de la cátedra, "Lenguaje Assembly".***
- ***Manual del simulador MSX88.***
- ***Set de Instrucciones de MSX88.***

---

1) Dada la siguiente definición de datos y el código: **`F = [(A+B)/C]-D`**

nombre	tamaño	valor
A:	1 byte	6
B:	1 byte	4
C:	1 byte	2
D:	1 byte	1
F:	1 byte	?

Suponiendo que se poseen las instrucciones necesarias en cada caso, escribir el programa que implemente el código anterior utilizando máquinas de 1, 2 ó 3 direcciones.

Maq. de 1 dirección	Maq. de 2 direcciones	Maq. de 3 direcciones

2) Suponga que cada código de operación ocupa 6 bits y las direcciones son de 10 bits. Analice las soluciones implementadas en el ejercicio anterior y complete la siguiente tabla:

	M. de 1 dirección	M. de 2 direcciones	M. de 3 direcciones
Tamaño del programa en memoria (cod.operación + operandos)			
Cantidad de accesos a memoria (instrucciones + operandos)			

3) Dado el siguiente código: **F = ((A - B)*C) + (D/E)**;

a) Implemente el código utilizando máquinas de 1, 2 y 3 direcciones.
b) Realice una tabla de comparación similar a la del ejercicio 2.
c) ¿Cuál máquina elegiría haciendo un balance de la cantidad de instrucciones, el espacio en memoria ocupado y el tiempo de ejecución (1 acceso a memoria = 1 ms)? ¿Es ésta una conclusión general?

Para cada programa propuesto en los siguientes ejercicios, deberá editar el archivo fuente con extensión asm (ej: ejer1.asm), luego ensamblarlo usando asm88.exe (comando: asm88 ejer1.asm) y enlazarlo con link88.exe (comando: link88 ejer1.o). Cada archivo obtenido con extensión eje (ej: ejer1.eje) deberá ser cargado y ejecutado en el simulador MSX88.

4) El siguiente programa utiliza una **instrucción de transferencia de datos** (instrucción MOV) con diferentes modos de direccionamiento para referenciar sus operandos. Ejecutar y analizar el funcionamiento de cada instrucción en el Simulador MSX88 observando el flujo de información a través del BUS DE DATOS, el BUS DE DIRECCIONES, el BUS DE CONTROL, el contenido de REGISTROS, de posiciones de MEMORIA, operaciones en la ALU, etc.

```
        ORG 1000h
NUM0    DB  0CAh
NUM1    DB  0
NUM2    DW  ?
NUM3    DW  0ABCDh
NUM4    DW  ?
```

```
ORG 2000H
MOV BL, NUM0
MOV BH, 0FFh
MOV CH, BL
MOV AX, BX
MOV NUM1, AL
MOV NUM2, 1234h
MOV BX, OFFSET NUM3
MOV DL, [BX]
MOV AX, [BX]
MOV BX, 1006h
MOV WORD PTR [BX], 0CDEFh
HLT
END
```

Cuestionario:

a) Explicar detalladamente qué hace cada instrucción MOV del programa anterior, en función de sus operandos y su modo de direccionamiento.
b) Confeccionar una tabla que contenga todas las instrucciones MOV anteriores, el modo de direccionamiento y el contenido final del operando destino de cada una de ellas.
c) Notar que durante la ejecución de algunas instrucciones MOV aparece en la pantalla del simulador un registro temporal denominado "**ri**", en ocasiones acompañado por otro registro temporal denominado "**id**". Explicar con detalle que función cumplen estos registros.

5) El siguiente programa utiliza diferentes **instrucciones de procesamiento de datos** (instrucciones aritméticas y lógicas). Analice el comportamiento de ellas y ejecute el programa en el MSX88.

```
      ORG 1000H
NUM0  DB  80h
NUM1  DB  200
NUM2  DB  -1
BYTE0 DB  01111111B
BYTE1 DB  10101010B
```

```
ORG  2000H
MOV  AL, NUM0
ADD  AL, AL
INC  NUM1
MOV  BH, NUM1
MOV  BL, BH
DEC  BL
SUB  BL, BH
MOV  CH, BYTE1
AND  CH, BYTE0
NOT  BYTE0
OR   CH, BYTE0
XOR  CH, 11111111B
HLT
END
```

Cuestionario:

1) ¿Cuál es el estado de los FLAGS después de la ejecución de las instrucciones ADD y SUB del programa anterior? Justificar el estado (1 ó 0) de cada uno de ellos. ¿Dan alguna indicación acerca de la correctitud de los resultados?
2) ¿Qué cadenas binarias representan a NUM1 y NUM2 en la memoria del simulador? ¿En qué sistemas binarios están expresados estos valores?
3) Confeccionar una tabla que indique para cada operación aritmética ó lógica del programa, el valor de sus operandos, en qué registro o dirección de memoria se almacenan y el resultado de cada operación.

6) El siguiente programa implementa un contador utilizando una **instrucción de transferencia de control**. Analice el funcionamiento de cada instrucción y en particular las del lazo repetitivo que provoca la cuenta.

```
      ORG 1000H
INI   DB  0
FIN   DB  15
```

```
      ORG  2000H
      MOV  AL, INI
      MOV  AH, FIN
SUMA: INC  AL
      CMP  AL, AH
      JNZ  SUM
      HLT
      END
```

Cuestionario:

a) ¿Cuántas veces se ejecuta el lazo? ¿De qué variables depende esto en el caso general?
b) Analice y ejecute el programa reemplazando la instrucción de salto condicional JNZ por las siguientes, indicando en cada caso el contenido final del registro AL:
   1º) JS
   2º) JZ
   3º) JMP

7) Escribir un programa en lenguaje assembly del MSX88 que implemente la sentencia condicional de un lenguaje de alto nivel IF **A** < **B** THEN **C** = **A** ELSE **C** = **B**. Considerar que las variables de la sentencia están almacenadas en los registros internos de la CPU del siguiente modo **A** en AL, **B** en BL y **C** en CL.
Determine las modificaciones que debería hacer al programa si la condición de la sentencia IF fuera:
   a) A ≤ B
   b) A = B

8) El siguiente programa suma todos los elementos de una tabla almacenada a partir de la dirección 1000H de la memoria del simulador. Analice el funcionamiento y determine el resultado de la suma. Comprobar resultado en el MSX88.

```
       ORG 1000H
TABLA  DB  2,4,6,8,10,12,14,16,18,20
FIN    DB  ?
TOTAL  DB  ?
MAX    DB  13
```

```
       ORG 2000H
       MOV AL, 0
       MOV CL, OFFSET FIN - OFFSET TABLA
       MOV BX, OFFSET TABLA
SUMA:  ADD AL, [BX]
       INC BX
       DEC CL
       JNZ SUMA
       HLT
       END
```

¿Qué modificaciones deberá hacer en el programa para que el mismo almacene el resultado de la suma en la celda etiquetada TOTAL?

9) Escribir un programa que, utilizando las mismas variables y datos que el programa del punto anterior (TABLA, FIN, TOTAL, MAX), determine cuántos de los elementos de TABLA son menores o iguales que MAX. Dicha cantidad debe almacenarse en la celda TOTAL.

10) Analizar el funcionamiento del siguiente programa.

```
       ORG 2000H
       MOV AX, 1
       MOV BX, 1000h
CARGA: MOV [BX], AX
       ADD BX, 2
       ADD AX, AX
       CMP AX, 200
       JS  CARGA
       HLT
       END
```

Cuestionario:

a) El programa genera una tabla. ¿Cómo están relacionados sus elementos entre sí?
b) ¿A partir de qué dirección de memoria se crea la tabla? ¿Cuál es la longitud de cada uno de sus elementos (medida en bits)?
c) ¿Cuántos elementos tiene la tabla una vez finalizada la ejecución del programa? ¿De qué depende esta cantidad?

11) Escribir un programa que genere una tabla a partir de la dirección de memoria almacenada en la celda DIR con los múltiplos de 5 desde cero hasta MAX.

12) Escribir un programa que, dado un número X, genere un arreglo con todos los resultados que se obtienen hasta llegar a 0, aplicando la siguiente fórmula: si X es par, se le resta 7; si es impar, se le suma 5, y al resultado se le aplica nuevamente la misma fórmula. Ej: si X = 3 entonces el arreglo tendrá: 8, 1, 6, -1, 4, -3, 2, -5, 0.

13) Dada la frase "Organización y la Computación", almacenada en la memoria, escriba un programa que determine cuantas letras ‘a’ seguidas de ‘c’ hay en ella.

14) Escribir un programa que sume dos números representados en Ca2 de 32 bits almacenados en memoria de datos y etiquetados NUM1 y NUM2 y guarde el resultado en RESUL (en este caso cada dato y el resultado ocuparán 4 celdas consecutivas de memoria). Verifique el resultado final y almacene 0FFH en la celda BIEN en caso de ser correcto o en otra MAL en caso de no serlo. Recordar que el MSX88 trabaja con números en Ca2 pero tener en cuenta que las operaciones con los 16 bits menos significativos de cada número deben realizarse en BSS.

15) Escribir un programa que efectúe la suma de dos vectores de 6 elementos cada uno (donde cada elemento es un número de 32 bits) almacenados en memoria de datos y etiquetados TAB1 y TAB2 y guarde el resultado en TAB3. Suponer en primera instancia que no existirán errores de tipo aritmético (ni carry ni overflow), luego analizar y definir los cambios y agregados necesarios que deberían realizarse al programa para tenerlos en cuenta.

16) Los siguientes programas realizan la misma tarea, en uno de ellos se utiliza una **instrucción de transferencia de control con retorno**. Analícelos y compruebe la equivalencia funcional.

```
      ; Memoria de Datos
      ORG 1000H
NUM1  DB    5H
NUM2  DB    3H

      ; Memoria de Instrucciones
      ORG 2000H
      MOV   AL, NUM1
      CMP   AL, 0
      JZ    FIN
      MOV   AH, 0
      MOV   DX, 0
      MOV   CL, NUM2
LOOP: CMP   CL, 0
      JZ    FIN
      ADD   DX, AX
      DEC   CL
      JMP   LOOP
FIN:  HLT
      END
```

```
      ; Memoria de Datos
      ORG 1000H
NUM1  DB    5H
NUM2  DB    3H

      ; Memoria de Instrucciones
      ORG 3000H   ; Subrutina SUB1
SUB1: CMP   AL, 0
      JZ    FIN
      CMP   CL, 0
      JZ    FIN
      MOV   AH, 0
      MOV   DX, 0
LAZO: ADD   DX, AX
      DEC   CX
      JNZ   LAZO
FIN:  RET

      ORG 2000H   ; Programa principal
      MOV   AL, NUM1
      MOV   CL, NUM2
      CALL  SUB1
      HLT
      END
```

Responder:

1) ¿Cuál es la tarea realizada por ambos programas?
2) ¿Dónde queda almacenado el resultado?
3) ¿Cuál programa realiza la tarea más rápido? ¿El tiempo de ejecución de la tarea depende de los valores almacenados en NUM1, en NUM2, en ambos lugares o en ninguno?

Explicar detalladamente:

a) Todas las acciones que tienen lugar al ejecutarse la instrucción CALL SUB1.
b) ¿Qué operación se realiza con la instrucción RET?, ¿cómo sabe la CPU a qué dirección de memoria debe retornar desde la subrutina al programa principal?

17) El siguiente programa es otra forma de implementación de la tarea del punto anterior (ejercicio 16). Analizar y establecer las diferencias con las anteriores, en particular las relacionadas a la forma de 'proveer' los operandos a las subrutinas.

```
      ; Memoria de datos
      ORG 1000H
NUM1  DW    5H    ; NUM1 y NUM2 deben ser mayores que cero
NUM2  DW    3H

      ; Memoria de Instrucciones
      ORG 3000H   ; Subrutina SUB2
SUB2: MOV   DX, 0
LAZO: MOV   BX, AX
      ADD   DX, [BX]
      PUSH  DX
      MOV   BX, CX
      MOV   DX, [BX]
      DEC   DX
      MOV   [BX], DX
      POP   DX
      JNZ   LAZO
      RET

      ORG 2000H   ; Programa principal
      MOV   AX, OFFSET NUM1
      MOV   CX, OFFSET NUM2
      CALL  SUB2
      HLT
      END
```

Explicar detalladamente:

a) Todas las acciones que tienen lugar al ejecutarse las instrucciones PUSH DX y POP DX.
b) Cuáles son los dos usos que tiene el registro DX en la subrutina SUB2.

18) Escribir un programa que sume 2 vectores de 6 elementos (similar al realizado en el ejercicio 15), de modo tal que utilice una subrutina que sume números de 32 bits (similar al programa escrito en ejercicio 14).

19) Escriba una subrutina que reciba la mantisa entera en BSS y el exponente en BSS de un número en los registros AH y AL respectivamente y devuelva, en ellos, una representación equivalente del mismo pero con el exponente disminuido en 1 y la mantisa ajustada. De no ser posible el ajuste, BL debe contener 0FFH en vez de 00H en el retorno.

20) Escriba una subrutina que reciba como parámetro un número en el formato IEEE 754 de simple precisión y analice/verifique las características del mismo devolviendo en el registro CL un valor igual a 0 si el número está sin normalizar, 1 en caso de ser +/- infinito, 2 si es un NAN, 3 si es un +/- cero y 4 si es un número normalizado. La subrutina recibe en AX la parte alta del número y en BX la parte baja.

21) Modifique la subrutina del ejercicio 19 para el caso en que la mantisa y el exponente estén representados en BCS.

Datos útiles:

- Las subrutinas siempre se escriben antes que el programa principal, aunque su dirección de comienzo sea más alta.
- Las etiquetas de subrutinas y bucles van seguidas de dos puntos (:).
- Los operandos en hexadecimal terminan en H y los que comienzan con una letra van precedidos por un cero (0) para no ser confundidos con etiquetas (por ejemplo, 0A4H en lugar de A4H).
- Se pueden incluir comentarios en los programas, anteponiendo siempre un punto y coma (;).
- El direccionamiento indirecto solo está implementado con el registro BX.
- Cada celda de memoria almacena un byte. Los datos de dos bytes (words) se almacenan de la siguiente manera: primero la parte baja (byte menos significativo) y luego la parte alta. Esto se corresponde con la idea de que la parte baja del dato se almacena en la dirección más baja y la parte alta, en la dirección más alta.