PRACTICA 1

Assembly, Instrucciones, Programas, Subrutinas y Simulador MSX88

Objetivos: que el alumno

- *Domine las instrucciones básicas del lenguaje assembly del MSX88.*
- *Utilice los diferentes modos de direccionamiento.*
- *Realice el diseño de programas utilizando instrucciones del MSX88.*
- *Comprenda la utilidad y funcionamiento de las subrutinas.*
- *Ejecute y verifique los resultados, analizando el flujo de información entre los distintos componentes del sistema.*

Herramientas y Bibliografía

- *Simulador MSX88*
- *Manual del simulador MSX88.*
- *Set de Instrucciones de MSX88.*

Para cada programa propuesto, 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.

1) 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], 1006H
        HLT
        END
```

Cuestionario:

1a) Explicar detalladamente qué hace cada instrucción MOV del programa anterior, en función de sus operandos y su modo de direccionamiento.

1b) 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.

1c) 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.

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

2a) ¿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?

2b) ¿Qué cadenas binarias representan a NUM1 y NUM2 en la memoria del simulador? ¿En qué sistemas binarios están expresados estos valores?

2c) 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.

3) 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   SUMA
        HLT
        END
```

Cuestionario:

3a) ¿Cuántas veces se ejecuta el lazo? ¿De qué variables depende esto en el caso general? Se ejecuta 15 veces. Depende de INI y de FIN

3b) 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:

1°) JS
2°) JZ
3°) JMP

4) 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

5) 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 el resultado en el MSX88.

```
        ORG   1000H
```

```
TABLA DB    DUP(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?

6) 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.

7) Multiplicación de números sin signo. Escribir un programa que calcule el producto entre dos números sin signo almacenados en la memoria del microprocesador:
7.1) sin hacer llamados a subrutinas, resolviendo el problema desde el programa principal;
7.2) llamando a una subrutina MUL para efectuar la operación, pasando los parámetros por valor desde el programa principal a través de registros;
7.3) llamando a una subrutina MUL, pasando los parámetros por referencia desde el programa principal a través de registros;
7.4) llamando a una subrutina MUL, pasando los parámetros por valor y por referencia a través de la pila.

```
7.1) ; Memoria de Datos
          ORG 1000H
NUM1  DB    5H
NUM2  DB    3H
; Memoria de Instrucciones
          ORG 2000H
          MOV   DX, 0
          MOV   AH, 0
          MOV   AL, NUM1
          CMP   AL, 0
          JZ    FIN
          MOV   CL, NUM2
LOOP:     CMP   CL, 0
          JZ    FIN
          ADD   DX, AX
          DEC   CL
          JMP   LOOP
FIN:      HLT
          END
```

CMP --> modifica flags

preg de examen -> q banderas se ven afectadas segun el programa q nos den

```
7.2) ; Memoria de Datos
          ORG 1000H
NUM1  DB    5H
NUM2  DB    3H
; Memoria de Instrucciones
          ORG 3000H       ; Subrutina MUL
MUL:      CMP   AL, 0
          JZ    FIN
          CMP   CL, 0
          JZ    FIN
LAZO:     ADD   DX, AX
          DEC   CX
          JNZ   LAZO
FIN:      RET

          ORG 2000H       ; Programa principal
          MOV   AL, NUM1
          MOV   CL, NUM2
          MOV   DX, 0
          MOV   AH, 0
          CALL  MUL
          HLT
          END
```

7.3)
```
; 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 MUL
MUL:    MOV     BX, AX
        ADD     DX, [BX]
        PUSH    DX
        MOV     BX, CX
        MOV     DX, [BX]
        DEC     DX
        MOV     [BX], DX
        POP     DX
        JNZ     MUL
        RET

        ORG 2000H       ; Programa principal
        MOV     AX, OFFSET NUM1
        MOV     CX, OFFSET NUM2
        MOV     DX, 0
        CALL    MUL
        HLT
        END
```

7.4)
```
; Memoria de datos
        ORG 1000H
NUM1    DW      5H
NUM2    DW      3H
RES     DW      ?
; Subrutina MUL
        ORG     3000H
MUL:    PUSH    BX
        MOV     BX, SP
        PUSH    CX
        PUSH    AX
        PUSH    DX
        ADD     BX, 6
        MOV     CX, [BX]
        ADD     BX, 2
        MOV     AX, [BX]
SUMA:   ADD     DX, AX
        DEC     CX
        JNZ     SUMA
        SUB     BX, 4
        MOV     AX, [BX]
        MOV     BX, AX
        MOV     [BX], DX
        POP     DX
        POP     AX
        POP     CX
        POP     BX
        RET
; Programa principal
        ORG     2000H
        MOV     AX, NUM1
        PUSH    AX
        MOV     AX, NUM2
        PUSH    AX
        MOV     AX, OFFSET RES
        PUSH    AX
        MOV     DX, 0
        CALL    MUL
        POP     AX
        POP     AX
        POP     AX
        HLT
        END
```

Analizar el diagrama esquemático de la pila y verificar con el simulador:

DIRECCIÓN DE MEMORIA	CONTENIDO
7FF0	DL
	DH
7FF2	AL
	AH
7FF4	CL
	CH
7FF6	BL
	BH
7FF8	IP L
	IP H
7FFA	DIR. RES L
	DIR. RES H
7FFC	NUM2 L
	NUM2 H
7FFE	NUM1 L
	NUM1 H
8000	-

Explicar detalladamente:

a) Todas las acciones que tienen lugar al ejecutarse la instrucción CALL MUL.

b) Todas las acciones que tienen lugar al ejecutarse las instrucciones PUSH y POP.
c) ¿Qué operación se realiza con la instrucción RET?

8) Escribir una subrutina ROTARIZ que haga una rotación hacia la izquierda de los bits de un byte almacenado en la memoria del microprocesador. Dicho byte y el número de posiciones a rotar deben pasarse por valor desde el programa principal a la subrutina a través de registros.

9) Escribir una subrutina CONCAR que cuente el número de caracteres de una cadena de caracteres terminada en cero (00H) almacenada en la memoria del microprocesador. La cadena se pasa a la subrutina por referencia vía registro.

10) Escribir una subrutina SWAP que intercambie dos datos de 16 bits almacenados en memoria. Los parámetros deben ser pasados por referencia desde el programa principal a través de la pila.

11) Modificar la subrutina del ejercicio 9 para que cuente la cantidad de veces que se repite un dado caracter en una cadena. Además, la subrutina debe cambiar el caracter especificado por una "X". El caracter a buscar se debe pasar por valor mientras que la cadena a analizar por referencia, todo a través de registros.

12) Usando la subrutina ROTARIZ del ejercicio 8, escriba una subrutina ROTARDER que haga una rotación hacia la derecha de un byte almacenado en la memoria del microprocesador. Dicho byte y el número de posiciones a rotar deben pasarse por valor desde el programa principal a la subrutina a través de registros.

13) Escriba la subrutina ES_VOCAL, que determina si un caracter es vocal o no. La rutina debe recibir el caracter por valor, y debe retornar, vía registro, el valor 0FFH si el caracter es una vocal, o 00H en caso contrario.

14) Usando la subrutina del ejercicio anterior (ejercicio 13) escribir la subrutina VOCALES, que recibe una cadena por referencia, y devuelve, en un registro, la cantidad de vocales que tiene esa cadena.