

Organización de Computadoras 2009

Práctica 8 – Programando

Objetivos de la práctica: que el alumno

- *Realice el diseño de programas utilizando instrucciones del MSX88.*
- *Comprenda la utilidad y funcionamiento de las subrutinas.*

Bibliografía:

- *Apunte 4 de la cátedra, “Lenguaje Assembler”.*
- *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) Escribir un programa que sume dos números de 16 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 2 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 8 bits menos significativos de cada número deben ser en BSS.

2) Escribir un programa que efectúe la suma de dos vectores de 6 elementos cada uno (donde cada elemento es un número de 16 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.

3) 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?

Organización de Computadoras 2009

Explicar detalladamente:

- Todas las acciones que tienen lugar al ejecutarse la instrucción CALL SUB1.
- ¿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?

4) El siguiente programa es otra forma de implementación de la tarea del punto anterior (ejercicio3). 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
        ---
```

Explicar detalladamente:

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

5) Escribir un programa que sume 2 vectores de 6 elementos (similar al realizado en el ejercicio 2), de modo tal que utilice una subrutina que sume números de 16 bits (similar al programa escrito en ejercicio 1).

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.