MÓDULO ASSEMBLER

Pila del procesador y Subrutinas

Autores:
Alejandro Héctor Gonzalez
Silvana Lis Gallo
Junio 2021

RESUMEN

En esta clase se trabaja el concepto de PILA del procesador y su relación con el llamado a subrutinas
Se explican los tipos de pasaje de parámetro en bajo y nivel y se desarrolla el pasaje de parámetros por valor y por referencia vía registro

Palabras clave

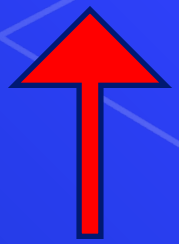
pila, subrutina, pasaje de parámetros, registros

Pila - Características

Cuando nos referimos a **“pila”**, nos referimos a un segmento de memoria donde se implementa una estructura **LIFO** (Last Input First Output), donde el último elemento en entrar es el primero en salir.

La pila se implementa sobre la misma memoria que la del programa.

Los elementos apilados tienen **16 bits (ni mas, ni menos).**

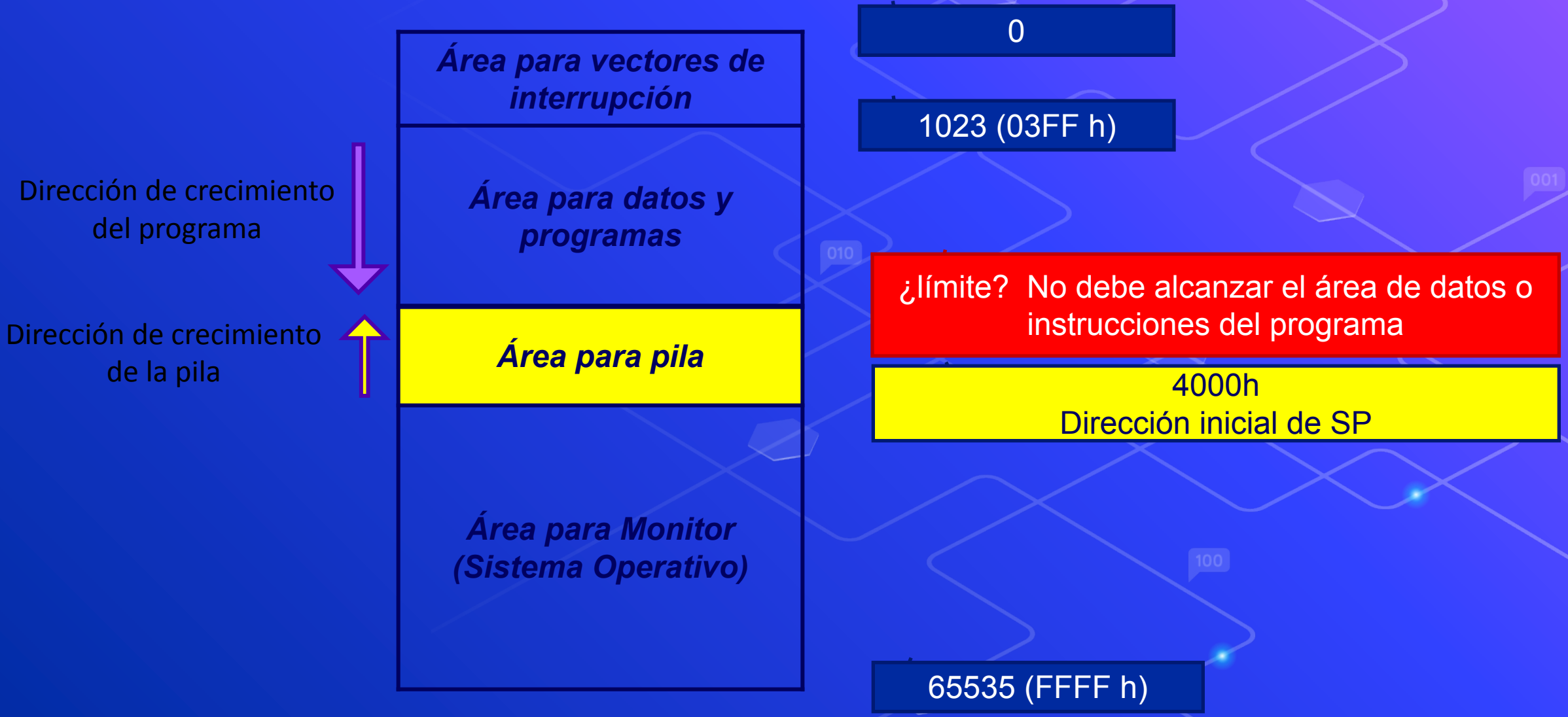


Pila - Características

Cuenta con un registro de 16 bits para implementar la pila. Se denomina **SP** (Stack Pointer o Puntero de Pila) y apunta siempre al último elemento apilado.

Tiene 2 operaciones: **Push** (Apilar) y **Pop** (Desapilar)

Pila - Memoria



Pila del MSX 88 – Operaciones

Operaciones de pila (seudo-código):

Push {registro 16 bits}:

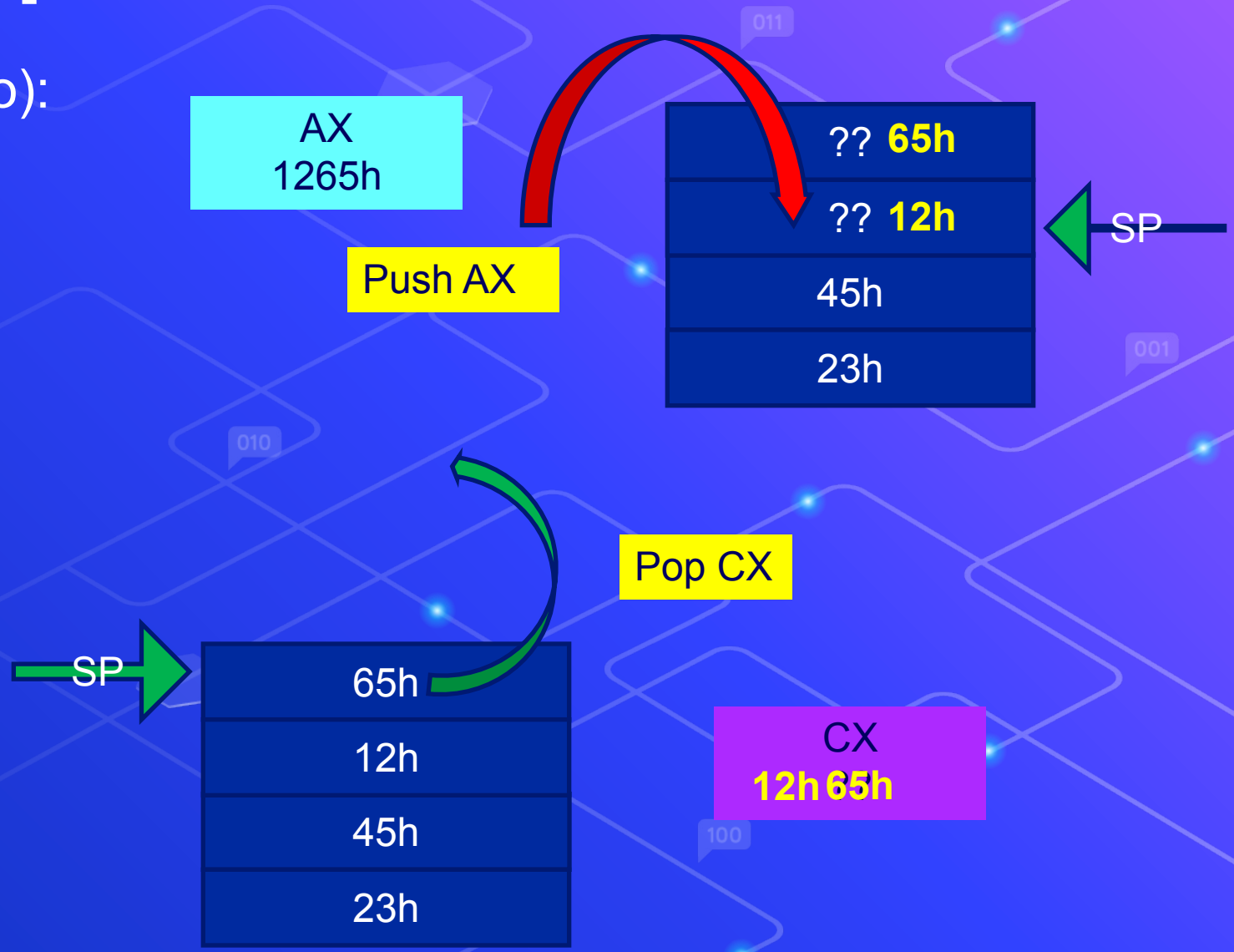
SP SP – 2

[SP] {Reg. 16 bits}

Pop {registro de 16 bits}:

{Reg. 16 bits} [SP]

SP SP + 2



Pila del MSX 88 – Operaciones

Ejemplos:

PUSH AX

POP BX

PUSH CL

POP MiVar

CL es de 8 bits NO!

No se puede, no compila!!!!, ¿Porqué?

Es una dirección de memoria!!

Ejemplo de PUSH y POP

```
        org   1000h
datos   db    55h,33h, 44h, 22h
        org   2000h
        mov   BX,1000h
        mov   AX, [BX]
        push  AX
        mov   BX, 1002h
        mov   CX, [BX]
        push  CX
        pop   AX
        pop   CX
        hlt
        end
```

Memoria de Datos	
1000	55
1001	33
1002	44
1003	22

datos
datos
datos
datos

Pila	
3FFC	44
3FFD	22
3FFE	55
3FFF	33

Contenido de CX apilado

Contenido de AX apilado

Estado final de

AX | 22 | 44 |

CX | 33 | 55 |

Subrutinas

Subrutinas - Ventajas

Evitan la repetición de código.

Facilitan la reutilización de código.

Permiten modularizar las soluciones de los programas.

Facilitan la lectura y comprensión del programa porque ocultan los detalles de implementación.

Limita la posibilidad de cometer errores.

Permite independencia de variables si se utilizan parámetros.

Subrutinas - Instrucciones

Como invocar a una subrutina (seudo-código):

CALL *{Dir. Subrutina}*

SP ← SP – 2

[SP] ← {Dir. Retorno = instr. Sig. al CALL}

IP ← *{Dir. Subrutina}*

Pone en la pila la dirección de retorno

Asigna dir. rutina a IP

Como regresar de una subrutina (seudo-código):

RET

IP ← [SP]

SP ← SP + 2

Asigna dirección de retorno a IP

Desapila

Muy importante recordar el movimiento de la pila y el SP para el pasaje de parámetros

Subrutinas - Ejemplo

Subrutina para sumar

```
ORG 3000H
MI_SUMA:    ADD BX, AX
            RET
OTRA_RUTINA: .....
```

```
            ORG 2000H ; prog principal
            MOV AX, 5
            MOV BX, 10
            CALL MI_SUMA
            ... ; esta es la dirección de retorno
```

MEM
2000h
2003h
2006h
2009h

Al ejecutar CALL se asigna:
IP = 3000h
[SP] = 2009h

Ejemplo con subrutina

```
            org    1000h
 dato       db      55h
            org    3000h
subrutina:   neg    AX
            ret
            org    2000h
            mov   BX, 1212h
            mov   AX, dato
            call subrutina
            push  AX
            pop   BX
            hlt
            end
```

Probar en VonSim

¿Qué valor final queda en AX y porque?

¿Qué valor final queda en BX y porqué?

Subrutinas y Pasaje de parámetros

Subrutinas - Parámetros

Los parámetros nos permiten intercambiar datos con subrutinas.

Ofrecen independencia entre las variables del programa y las subrutinas. En otras palabras las subrutinas no quedan ligadas a variables del programa.

Existen dos clasificaciones (superpuestas) del pasaje de parámetros:

1) Por valor y por referencia.
2) Por registro o por pila.

Subrutinas - Parámetros

Por Registro y Valor

Tamaño: 8 o 16 bits

Pasaje: cargar registros, llamar subrutina.

Uso: usar directamente desde la subrutina los registros para acceder a los valores.

```
Ejemplo:        MOV AL, MiVar1
                CALL Subrutina
                ...
Subrutina:      ADD AL, 1
                …
                RET
```

Subrutinas - Parámetros

Por Registro y Referencia

Tamaño: solo 16 bits (tamaño de una dirección de memoria)

Pasaje: cargar registros usando OFFSET, llamar subrutina.

Uso: desde la subrutina direccionar indirectamente con BX para acceder a los valores.

```
Ejemplo:        MOV AX, OFFSET MiVar1
                CALL Subrutina
                ...
Subrutina:      MOV BX, AX
                OR [BX], CX; cambia la celda apuntada por BX
                …
                RET
```

Parámetros – Ejemplos

Escribir un programa que calcule el producto entre dos números sin signo almacenados en la memoria del microprocesador:

Ejemplo 1) Llamando a una subrutina MUL para efectuar la operación, pasando los parámetros **por valor** desde el programa principal **a través de registros**

Ejemplo 2) Llamando a una subrutina MUL, pasando los parámetros **por referencia** desde el programa principal **a través de registros**.

Parámetros – Ejemplo 1- Parámetros por valor vía registros

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

; Instrucciones
; Subrutina MUL
; entrada AL y CL
; resultado en DX
ORG 3000H
MUL: MOV DX, 0
     CMP AL, 0
     JZ FIN
     CMP CL, 0
     JZ FIN
     MOV AH, 0
LAZO: ADD DX, AX
     DEC CL
     JNZ LAZO
FIN: RET
 ORG 2000H ; Programa
```

•. Inicializo DX en cero por si la operación da 0

• **La multiplicación se resuelve haciendo sumas**.
• Se usa 16 bits (DX) porque el resultado puede no entrar en 8 bits (ej: 255 x 255 = 65535).

• Esto es por si AL o CL son cero (el resultado es 0)

•Es para que AX tenga un valor válido en 16 bits. Como no hay instrucciones que sumen registros de 16 bits con 8 bits.

Parámetros – Ejemplo 2 - Parámetros por referencia vía registros

```
; Mem. de datos
ORG 1000H
NUM1 DW 5H
NUM2 DW 3H
; NUM1 y NUM2 > 0
ORG 3000H ; Subrutina MUL
MUL : 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
 MOV AX, OFFSET NUM1
 MOV CX, OFFSET NUM2
 CALL MUL
 HLT
```

; resultado de 16 bits en DX
; recupera direccion de NUM1 (1000H)
; recupera valor de NUM1 (5H) y suma a DX
; guarda resultado parcial en pila
; recupera direccion de NUM2 (1002H)
; recupera valor de NUM2 (3H, 2H, 1H)
; decrementa valor recuperado de NUM2
; asigna NUM2 (queda modificado en memoria)
; recupera resultado desde la pila
; flag Z=0 vuelve a L8.
Nota Z se modificó en L13. L14 y L15 no modifican Z.