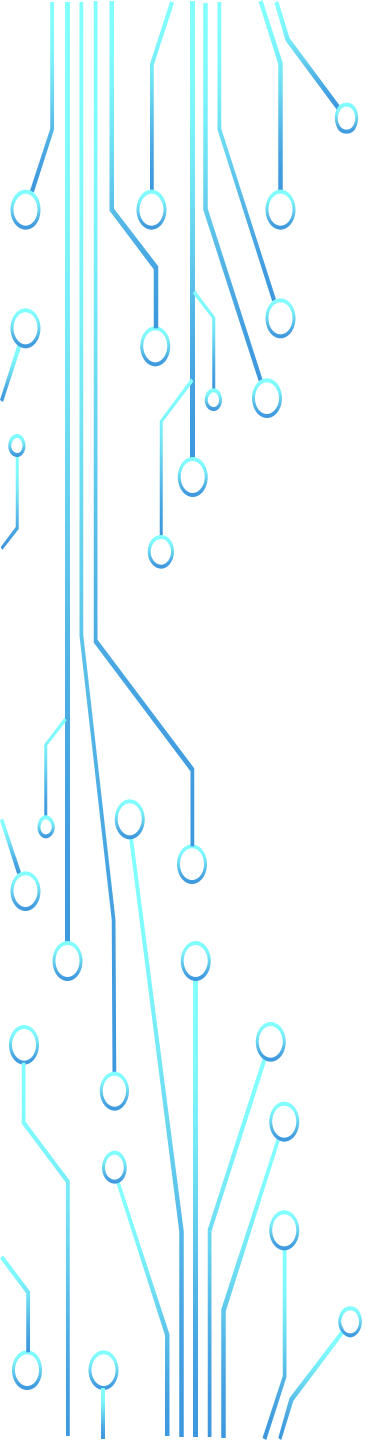


LENGUAJE ENSAMBLADOR PASAJE DE PARÁMETROS A SUBRUTINAS

FORMAS DE PASAR PARÁMETROS A SUBRUTINAS

Hay 3 formas generales de pasar parámetros, a través de:

- registros de la CPU
- zona de memoria
- pila

PASAJE DE PARÁMETROS EN LOS **REGISTROS**

- Los módulos utilizan ciertos registros para comunicarse. Todos los demás registros han de permanecer inalterados, por lo cual, si son empleados internamente, han de ser preservados al principio del módulo y restaurados al final.

Este es el método empleado por el DOS y la BIOS en la mayoría de las ocasiones para comunicarse con quien los llama.

PASAJE DE PARÁMETROS EN LOS **REGISTROS**

- Los registros serán preservados preferiblemente en la pila (con PUSH) y recuperados de la misma (con POP en orden inverso); de esta manera, los módulos son **reentrantes** y pueden ser llamados de manera múltiple soportando, entre otras características, la recursividad (sin embargo, se requerirá también que las variables locales se generen sobre la pila).

PASAJE DE PARÁMETROS A TRAVÉS DE UN ÁREA COMÚN DE **MEMORIA**

- Se utiliza una zona de memoria para la comunicación.

Este tipo de módulos no son reentrantes y hasta que no acaben de procesar una llamada no se les debe llamar de nuevo en medio del procesamiento

PASAJE DE PARÁMETROS POR **PILA**

- Los parámetros son apilados antes de llamar al módulo que los va a recoger. Este debe conocer el número y tamaño de los mismos, para equilibrar el puntero de pila al final antes de retornar (método de los compiladores de lenguaje Pascal) o en caso contrario el programa que llama deberá encargarse de esta operación (lenguaje C).

La ventaja del paso de parámetros por la pila es el prácticamente ilimitado número de parámetros admitido, de cómodo acceso, y que los módulos siguen siendo reentrantes.

EJERCICIO 1

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

1) Sin hacer llamados a subrutinas, **resolviendo el problema desde el programa principal;**

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

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

SOLUCIÓN EJERCICIO 1.1

1.1)

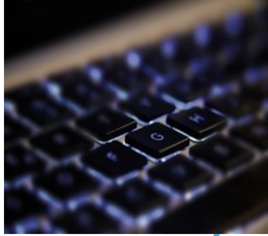
```
        ; Memoria de Datos
        ORG 1000H
NUM1    DB    5H
NUM2    DB    3H
        ; Memoria de Instrucciones
        ORG 2000H
        MOV   AL, NUM1
        CMP   AL, 0
        JZ    FIN
        MOV   AH, 0
        MOV   DX, 0
        MOV   CL, NUM2
LOOP:   CMP   CL, 0
        JZ    FIN
  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

SOLUCIÓN EJERCICIO1.2

1.2)

```
        ; 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
        MOV   AH, 0
        MOV   DX, 0
LAZO:   ADD   DX, AX
        DEC   CL
        JNZ   LAZO
FIN:    RET

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



1.3)

```
      ;
      OI
NUM1  DW
NUM2  DW     3H
```



```
      ; Memoria de Instrucciones
      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
      RET
```



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

EJERCICIO 2

Escribir un programa que calcule el producto entre dos números sin signo almacenados en la memoria del microprocesador llamando a una subrutina MUL, pero en este caso **pasando los parámetros por valor y por referencia a través de la pila**.

SOLUCIÓN EJERCICIO 2

```
        ORG   1000H; Memoria de datos
NUM1    DW    5H
NUM2    DW    3H
RES     DW    ?

        ORG   3000H ; Subrutina MUL
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
```

```
ORG   2000H ; Programa principal
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
```

¿Por qué se realizan estas operaciones?

DIRECCIÓN DE MEMORIA	CONTENIDO
7FF0H	DL
	DH
7FF2H	AL
	AH
7FF4H	CL
	CH
7FF6H	BL
	BH
7FF8H	IP RET. L
	IP RET. H
7FFAH	DIR. RES L
	DIR. RES H
7FFCH	NUM2 L
	NUM2 H
7FFEH	NUM1 L
	NUM1 H
8000H	——

¿Cuándo aparece esto?

POSIBLES PASOS GENERALES PARA ESCRIBIR UNA SUBRUTINA

1. Salvar el estado de BX
2. Salvar el estado de SP (SP=BX)
3. Reservar espacio ara daos locales (opcional)
4. Salvar valores de otros registros (opcional)
5. Acceder a parámetros
6. Escribir sentencias a ejecutar
7. Retornar parámetro (opcional)
8. Regresar correctamente de la subrutina

¿CÓMO SE ESCRIBE EN GENERAL LA SALIDA DE UNA SUBRUTINA?

1. Los registros salvados en la pila deben ser descargados en orden inverso
2. Si se reservó espacio para variables locales, se debe reponer SP con el valor de BX que no cambió durante la subrutina
3. Reponer BX
4. Volver al programa que llamó a la subrutina con RET

ANIDAMIENTOS DE SUBRUTINAS

```
            ORG 1000H
rutina1:    NEG     AX
            PUSH    AX
            CALL    rutina2
            POP     AX
            RET
            ORG 1020H
rutina2:    INC     AX
            RET
```

```
            ORG 2000H
PPIO:       MOV     BX, 0
            MOV     AX, dato
            PUSH    AX
            CALL    rutina1
            POP     BX
            HLT
            ORG 3000H
dato:       DB      55H
            END
```

EJEMPLO PARA EL SIMULADOR MSX88

```
        ORG 1000H
NUM1    DW  5H
NUM2    DW  3H
RES     DW  ?

        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

        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
```

EJERCICIO 3

- **Realizar una rutina que dado un vector de byte ya cargado, busque el final del mismo (byte = 0h) y termine dejando en BX la longitud del mismo.**

```
        org 1000h
        vec db "abcdefghijklmnopqrstuvwxyz",0 ; cargar un vector con constantes
        org 2000h
        mov bx,OFFSET vec      ; en bx tenemos la posición dentro del vector
otro: mov AL, [bx] ; cargamos en AL el elemento del vector indicado en BX
        inc bx
        cmp AL,0       ; comparamos  para ver el fin del vector
         jz fin        ; si la comparación es 0 salimos del programa
         jmp otro      ; buscamos otro elemento
fin:
         ret ; volver al programa
```

EJERCICIOS

- Realizar un programa que llame a la subrutina del ejercicio 3.

- Repaso

ESCRIBIR EL CÓDIGO EN PASCAL PARA EL SIGUIENTE CÓDIGO ASSEMBLER

```
        MOV AL, 0
        MOV CL, 10
Iterar: ADD AL, AL
        DEC CL
        CMP CL, 1
        JNZ Iterar
Fin:    HLT
```

¿Es un repeat….until?
¿Puede ser un for?
¿Por qué?

ESCRIBIR EL CÓDIGO EN PASCAL PARA EL SIGUIENTE CÓDIGO ASSEMBLER

```
        MOV AL, 0
        MOV CL, 10
Iterar: CMP CL, 1
        JZ Fin
        ADD AL, AL
        DEC CL
        JMP Iterar
Fin:    HLT
```

Escribir el código en Pascal para el siguiente código assembler

```
            MOV AL, 0
            MOV CL, 1
Iterar:     CMP CL, 10
            JZ Fin
            ADD AL, AL
            INC CL
            JMP Iterar
Fin:        HLT
```

¿Es un while?
¿Puede ser un for?
¿Por qué?

PASAR EL SIGUIENTE CÓDIGO EN EL SIMULADOR MSX88

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

        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 hace este código?

¿Qué modificaciones debe realizar para el resultado se almacene en TOTAL?

¿Conviene usar la variable Total en vez de registros de la CPU para acumular el total?

CALCULAR EL MÁXIMO ENTRE DOS NÚMEROS

```
                org2000h

                mov AX, 5

                mov BX, 10

                sub AX,BX

                js A_MENOR_B

                mov CX,AX    ;AX-BX mayor a 0

                jmp FIN

A_MENOR_B:      mov CX,BX

FIN:            HLT
```

¿Qué valor queda en CX independientemente del los valores de AX y BX?