TEORÍA PRÁCTICA 4

ESTRUCTURA DE UN PROGRAMA EN ENSAMBLADOR. EJEMPLOS SENCILLOS DE PROGRAMAS EN ENSAMBLADOR

OBJETIVOS:

1. ***Conocer la función y uso básico de la pila: PUSH y POP***
2. ***Realizar una serie de algoritmos sencillos que faciliten la comprensión y el manejo de las operaciones y saltos vistos hasta el momento.***
3. ***Introducir al alumno en la definición de procedimientos que faciliten la labor de programación y la legibilidad del código fuente***
4. ***Manejo básico de E/S***

CONTENIDOS:

1. ***Función y uso básico de la pila: PUSH y POP***
2. ***Definición de Procedimientos: PROC***
3. ***Uso de interrupciones software para la E/S de información a través de teclado y pantalla.***

EJEMPLOS:

1. ***Procedimiento que transforma un número de un byte introducido por teclado en su valor binario.***
2. ***Procedimiento que suma dos números. Los números se piden en el programa y se introducen por teclado.***
3. ***Procedimiento que calcula el máximo común divisor de dos números (introducidos por teclado) de tipo byte***

TEORÍA PRÁCTICA 4

1. Función y uso básico de la pila: PUSH y POP

La pila permite almacenar (apilar) y extraer (desapilar) valores en memoria (memoria reservada para la pila en la definición del segmento de pila) de una forma sencilla y cómoda sin tener que acceder a zonas de memoria directamente con lo que tendríamos que mantener indices y sin tener que definir previamente variables.

Las operaciones más básicas con la pila son:

· POP (extraer de la pila)

Transfiere el elemento palabra que se encuentra en la cima de la pila (apuntado por SP) al operando destino que ha de ser tipo palabra, e incrementa en dos el registro SP.

Sintaxis: POP destino

Indicadores:	OF	DF	IF	TF	SF	ZF	AF	PF	CF
	-	-	-	-	-	-	-	-	-

Ejemplos:

```
POP ax
POP variable1
```

TEORÍA PRÁCTICA 4

1. Función y uso básico de la pila: PUSH y POP

· PUSH (introduce en la pila)

Decrementa el puntero de pila (SP) en 2 y luego transfiere la palabra especificada en el operando fuente a la cima de la pila.

Sintaxis: PUSH origen

Indicadores:	OF	DF	IF	TF	SF	ZF	AF	PF	CF
	-	-	-	-	-	-	-	-	-

Ejemplo:

```
PUSH AX
```

2. Definición de procedimientos

¿Por qué?

Los procedimientos y en general la modularización facilita la labor de programación y la legibilidad del código fuente.

La programación por medio de procedimientos facilita así mismo la abstracción sobre determinadas funciones no implementadas, agiliza los desarrollos por medio de la reutilización de procedimientos y permite una mayor flexibilidad.

También mejora la ejecución y optimiza el tamaño de la aplicaciones que hacen un uso continuado de una determinada funcionalidad, ya que esta solo se implementa una vez y se puede invocar tantas veces como sea necesario.

Así mismo en grandes proyectos con amplios grupos de trabajo, permite definir claramente la tarea de cada miembro del equipo sin que esta se inmiscuya o influya en el trabajo de sus compañeros.

2. Definición de procedimientos

¿Donde y como definirlos?

Los procedimientos se definen, al igual que el procedimiento principal (p.e. START) dentro del segmento de código y se define de manera similar a como lo hemos hecho para el procedimiento principal.

Se pueden definir antes o después del procedimiento principal, pero nunca dentro de él.

El formato básico sería:

```
RESTAR PROC NEAR
 ;Aquí iría el código
...
RET
RESTAR ENDP
```

Para lograr que los procedimientos sean independientes del programa, que es lo que queremos lograr para poderlos reutilizar, utilizaremos las llamadas a la pila comentadas con el fin de no perder los valores de los registros. Para comprender esto pongamos un ejemplo:

Procedimiento para mostrar una cadena de caracteres por pantalla y realizar un programa que haga uso de esta llamada

Solución: ***(marcado solo el código añadido sobre el esqueleto propuesto por Emu8086)***

2. Definición de procedimientos

Solución incorrecta	Solución correcta
TITLE 8086 Code Template (for EXE file) ; AUTHOR emu8086 ; DATE ? ; VERSION 1.00 ; FILE ?.ASM ; 8086 Code Template ; Directive to make EXE output: #MAKE_EXE# DSEG SEGMENT 'DATA' cadena db "Hola a todos$" DSEG ENDS SSEG SEGMENT STACK 'STACK' DW 100h DUP(?) SSEG ENDS CSEG SEGMENT 'CODE'	TITLE 8086 Code Template (for EXE file) ; AUTHOR emu8086 ; DATE ? ; VERSION 1.00 ; FILE ?.ASM ; 8086 Code Template ; Directive to make EXE output: #MAKE_EXE# DSEG SEGMENT 'DATA' cadena db "Hola a todos$" DSEG ENDS SSEG SEGMENT STACK 'STACK' DW 100h DUP(?) SSEG ENDS CSEG SEGMENT 'CODE'

2. Definición de procedimientos

Solución incorrecta	Solución correcta

START PROC FAR
; Store return address to OS:
PUSH DS
MOV AX, 0
PUSH AX
; set segment registers:
MOV AX, DSEG
MOV DS, AX
MOV ES, AX
MOV AH,3
LEA DX, cadena
CALL MOSTRAR
DEC AH
RET
START ENDP
 	
MOSTRAR PROC NEAR
MOV AH,9
INT 21h
RET
MOSTRAR END
CSEG ENDS
END START ; set entry point.
	START PROC FAR
	; Store return address to OS:
	PUSH DS
	MOV AX, 0
	PUSH AX
	; set segment registers:
	MOV AX, DSEG
	MOV DS, AX
	MOV ES, AX
	MOV AH,3
	LEA DX, cadena
	CALL MOSTRAR
	DEC AH
	RET
	START ENDP
	MOSTRAR PROC NEAR
	PUSH AX
	MOV AH,9
	INT 21h
	POP AX
	RET
	MOSTRAR ENDP
	CSEG ENDS
	END START ; set entry point.

2. Definición de procedimientos

La diferencia entre una y otra solución estriba en el uso de la pila.

El procedimiento incorrecto funciona de la siguiente manera:

1) **Inicializa la entrada y los segmentos de registro**
2) **Mueve a AH la constante 3**
3) **Carga en DX la dirección de comienzo de la variable cadena**
4) **Llama a MOSTRAR para sacar por pantalla la variable 'cadena'. Dentro de este procedimiento, AH cambia de valor para contener el valor de la función a ejecutar (9) al llamar a la interrupción 21h**
5) **Decrementar el valor de AH**

Lo que hay que notar, es que el procedimiento MOSTRAR hace uso en la implementación del registro AH (MOV AH, 9) y por tanto del registro AX del que forma parte. Esto hace que el registro cambie de valor dentro de la función, lo que provocaría que al finalizar el programa principal AH contenga el valor 8 en vez del valor 2.

2. Definición de procedimientos

Para solucionar este tipo de problema utilizamos la pila. Lo que hacemos de forma general, aunque puede variar al utilizar los registros o la pila como variables de E/S, es apilar todos los registros que se usan como destino dentro del procedimiento (aparecen a la parte izquierda de las instrucciones con dos operandos) para después desafilarlos en orden inverso.

Para el ejemplo que nos ocupa, guardamos (apilamos) el valor del registro AX en la pila (solo AX ya que es el único registro que usa el procedimiento), realizamos las operaciones requeridas y antes de salir (RET) restauramos (desafilamos) el valor inicial del registro AX.

¿Cómo invocarlos?

La instrucción para invocar un procedimiento es ***CALL*** seguido del nombre del procedimiento.

Para el ejemplo concreto anterior, podríamos realizar la llamada al procedimiento como si se tratase de una instrucción más dentro por ejemplo del procedimiento principal (START) de la siguiente manera:

```
CALL MOSTRAR
```

2. Definición de procedimientos

¿Cómo pasarles parámetros?

Los parámetros de E/S en los procedimientos nos permiten especificar los datos con los que va a funcionar el procedimiento, es decir, si queremos sumar dos números qué números vamos a sumar (donde se encuentran los parámetros de entrada) y tras obtener el resultado donde va a quedar este almacenado (parámetro de salida).

Las opciones más usuales a la hora de pasar parámetros a los procedimientos son:

1. ***Por medio de variables o utilizando directamente la memoria principal***
2. ***Por medio de registros***

Ejemplos:

1. Por medio de variables o utilizando directamente la memoria principal: Procedimiento que suma de dos números con variables como parámetro de E/S.

Este procedimiento suma dos números contenidos en las variables A y B y devuelve el resultado en una tercera variable llamada C.

TEORÍA PRÁCTICA 4

2. Definición de procedimientos

```
...
DSEG    SEGMENT 'DATA'

    A dw 0
    B dw 0
    C dw 0

DSEG    ENDS
...

    MOV A, 3
    MOV B, 4
    CALL SUMAR

    RET

START   ENDP
...

SUMAR PROC NEAR
    PUSH AX
    MOV AX, A
    ADD AX, B
    MOV C, AX
    POP AX
    RET
SUMAR ENDP
...
```

2. Definición de procedimientos

Problema: Siempre que queramos utilizar este procedimiento deberemos definir tres variables adicionales con esos nombres… ¿y si tenemos 2 procedimientos que usan una misma variable?

1. Por medio de registros: **Procedimiento que resta dos números con registros como parámetros de entrada**

Este procedimiento resta dos números contenidos en BX y CX y devuelve el resultado en DX.

```
; Ahora el segmento de datos podría estar vacío

        …
        MOV BX, 4
        MOV CX, 3
        CALL RESTAR


        RET
        START   ENDP
```

2. Definición de procedimientos

```
RESTAR PROC NEAR
PUSH AX
MOV AX, BX
SUB AX, CX
 MOV DX, AX
POP AX
RET
RESTAR ENDP
...
```

Conviene hacer notar que aunque DX es destino de instrucción dentro del procedimiento (MOV DX, AX), como es también parámetro de salida, no sería correcto apilarlo y desapilarlo, ya que al desapilarlo (POP) reescribiríamos su contenido inicial y ya no almacenaría el resultado obtenido.

TEORÍA PRÁCTICA 4

3. Uso de interrupciones software para la E/S de información a través de teclado y pantalla.

Para realizar operaciones de E/S de información utilizaremos una interrupción software. Es una llamada a una rutina que gestiona la operación especifica de E/S.

La interrupción software es: ***INT 21h***

Esta interrupción sirve para hacer múltiples operaciones de E/S y para especificar la función exacta que deseas realizar se utiliza el registro AH.

Por ejemplo si deseamos sacar un mensaje por pantalla, esto son los pasos:

1. **Definir en el segmento de datos la variable de memoria que guarda el mensaje:**

Mensaje DB 'ESTO ES UN EJEMPLO $' (la cadena de caracteres debe acabar en $)

2. **Dentro del segmento de código pondremos las siguientes instrucciones:**

   MOV DX, OFFSET MENSAJE (DX debe guarda la dirección de memoria donde esta el mensaje guardado
   MOV AH, 9 (función para enviar una cadena de caracteres a pantalla)
   INT 21H

TEORÍA PRÁCTICA 4

3. Uso de interrupciones software para la E/S de información a través de teclado y pantalla.

Para introducir un valor numérico a través de teclado haremos lo siguiente:

1. Definir en el segmento de datos la variable de memoria que contendrá el número leído de teclado:

numero DB 0 (podemos ponerlo a valor inicial cero o cualquier otro valor)

2. Dentro del segmento de código:

```
MOV AH, 1          (función de lectura de teclado)
INT 21H            (espera a que se introduzca el número que se guarda en AL)
MOV numero,AL (guardo en la variable numero el valor leído)
```