Laboratorio de Sistemas Basados en Microprocesadores

Práctica 2a: Matrices y Vectores

En esta práctica, vamos a trabajar con pequeñas matrices. Se pretende elaborar un programa que permita recorrer una serie de matrices de tamaño 1x4 y validar su contenido. El alumno deberá realizar el correspondiente análisis de los requisitos expuestos y realizar su implementación. En el desarrollo de esta práctica, será necesario emplear una buena parte del juego de instrucciones del 8086.

Programa 1: pract2a.asm

Desarrolle un programa en lenguaje ensamblador que compruebe que una serie de vectores, con valores por defecto inicializados en el segmento de datos, cumple las siguientes reglas:

- Regla 1: Todos los elementos de este vector pertenecen al conjunto [1..4] (números del 1 al 4)
- Regla 2: En caso de que alguno de estos elementos no pertenezca a este conjunto, deberá imprimirse el mensaje “FUERA DEL CONJUNTO: 1,2,3,4”
- Regla 3: En caso de que todos los elementos pertenezcan a este conjunto deberá imprimirse el mensaje “CORRECTO”

Es decir, si tenemos la siguiente definición de vectores:

```
vector1 db 1,2,2,4

vector2 db 4,2,5,1

vector3 db 3,2,4,1
```

La salida de nuestro programa será la siguiente:

```
C:\PR2\P2>PRACT2A.EXE
CORRECTO
FUERA DEL CONJUNTO: 1,2,3,4
CORRECTO
```

Notas:

- Para acceder a los números de la matriz se ha de utilizar un direccionamiento base-indexado con desplazamiento fijo.
- Para la resolución de este ejercicio, recomendamos utilizar la plantilla *pract2a.asm* que encontraréis en el ANEXO II.

Programa 2: pract2b.asm

Desarrolle un programa en lenguaje ensamblador que compruebe que una serie de vectores, con valores por defecto inicializados en el segmento de datos, cumple las siguientes reglas:

- Regla 4: No existen elementos repetidos dentro de una misma matriz
- Regla 5: En caso de que alguno de estos elementos esté repetido, deberá imprimirse el mensaje "REPETICION"
- Regla 6: En caso de que no haya elementos repetidos deberá imprimirse el mensaje "CORRECTO"

Es decir, si tenemos la siguiente definición de vectores:

```
vector1 db 1,2,2,4

vector2 db 4,2,5,1

vector3 db 3,2,4,1
```

La salida de nuestro programa será la siguiente:

```
C:\PR2\P2>PRACT2b.EXE
REPETICION
CORRECTO
CORRECTO
```

Notas:

- Para acceder a los números de la matriz se ha de utilizar un direccionamiento base-indexado con desplazamiento fijo.
- Para la resolución de este ejercicio, recomendamos utilizar la plantilla *pract2a.asm* que encontraréis en el ANEXO II.

Programa 3: pract2c.asm

Desarrolle un programa en lenguaje ensamblador que, utilizando lo aprendido en los dos apartados anteriores, compruebe que una serie de vectores, con valores por defecto inicializados en el segmento de datos, cumple las siguientes reglas:

- Regla 1: Todos los elementos de este vector pertenecen al conjunto [1..4] (números del 1 al 4)
- Regla 2: En caso de que alguno de estos elementos no pertenezca a este conjunto, deberá imprimirse el mensaje "FUERA DEL CONJUNTO: 1,2,3,4"
- Regla 4: No existen elementos repetidos dentro de una misma matriz
- Regla 5: En caso de que alguno de estos elementos esté repetido, deberá imprimirse el mensaje "REPETICION"
- Regla 7: En caso de que se cumplan las Reglas 1 y 4, deberá imprimirse el mensaje "CORRECTO"

Además, se deberá implementar una subrutina auxiliar que convierta un número entero en una cadena de caracteres ASCII que represente los resultados como números en notación **decimal.**

Por ejemplo: el número 100h deberá ser convertido a la cadena de caracteres ASCII "256" (32h, 35h, 36h).

Esta subrutina será utilizada para imprimir los vectores junto al resultado obtenido al ser procesado.

Es decir, si tenemos la siguiente definición de vectores:

```
vector1 db 1,2,2,4

vector2 db 4,2,5,1

vector3 db 3,2,4,1
```

La salida de nuestro programa será la siguiente:

```
C:\PR2\P2>PRACT2C.EXE
 1 2 2 4 REPETICION
 4 2 5 1 FUERA DEL CONJUNTO: 1,2,3,4
 3 2 4 1 CORRECTO
```

Notas:

- Para acceder a los números de la matriz se ha de utilizar un direccionamiento base-indexado con desplazamiento fijo.
- La subrutina que convierte de binario a ASCII debe recibir el número a convertir en el registro AX, y retornar en DX:BX la dirección de memoria a partir de la que se encuentra la cadena de caracteres ASCII (En DX el segmento y en BX el desplazamiento). La cadena de caracteres debe estar terminada con '$', a fin de facilitar la impresión mediante la función 9 de la INT 21h del MS-DOS (ver anexo de la práctica y alumno.asm de la práctica 0).

ENTREGA DE LA PRÁCTICA:

Ejercicio 1

Se deberán entregar un zip con el fichero fuente (*pract2a.asm)* y el makefile.

Ejercicios 2 y 3

Se deberán entregar un zip incluyendo los ficheros fuentes (*pract2b.asm, pract2c.asm)* y el makefile.

Para todos los ejercicios, el código generado deberá estar correctamente tabulado y comentado. La falta de comentarios, o la baja calidad de éstos, será calificada negativamente.

Las fechas límite para la subida de los archivos son:

Ejercicio 1:

Grupo del Jueves: 25 de Marzo a las 20.15
Grupos del Viernes: 26 de Marzo a las 19:15h

Ejercicios 2 y 3:

Grupos del Jueves: 7 de Abril a las 23:55
Grupos del Viernes: 8 de Abril a las 23:55h

Anexo I: Funciones del sistema operativo para imprimir texto y finalizar la ejecución.

El sistema operativo MS-DOS facilita la mayor parte de sus servicios a través de la interrupción 21h. Antes de que nuestro programa invoque la llamada éste mediante la instrucción INT 21H, se debe cargar en el registro AH el número de la función solicitada. Además, cada función puede requerir de otros parámetros de entrada cuyos valores deberán almacenarse en otra serie de registros. A continuación, se detallan tres funciones de uso habitual en los programas que se desarrollan en el laboratorio.

Función 2H: Envío de un código ASCII al periférico pantalla

Descripción: Imprime un único carácter por pantalla.
Parámetros de entrada: AH = 2h.
DL = código ASCII del carácter que se imprimirá en pantalla.
Parámetros de salida: Ninguno.
Registros afectados: Ninguno.

Ejemplo:

```
mov ah, 2        ; Número de función = 2
mov dl, 'A'      ; Se desea imprimir la letra A
int 21h          ; Interrupción software al sistema operativo
```

Función 9H: Envío de una cadena de caracteres ASCII al periférico pantalla

Descripción: Impresión de una cadena de caracteres que termina con el carácter ASCII=$ por pantalla.
Parámetros de entrada: AH = 9h.
DX = Desplazamiento en memoria desde el origen del segmento de datos donde se ubica en memoria el primer carácter de la cadena.
Parámetros de salida: Ninguno.
Registros afectados: Ninguno.
Ejemplo:

```
.DATA
Texto DB "Hello world",13,10,'$' ; string terminado con los caracteres CR,LF y'$'
. CODE
.....
; Si DS es el segmento donde está el texto a imprimir:
mov dx, offset Texto      ; DX : offset al inicio del texto a imprimir
mov ah, 9                 ; Número de función = 9 (imprimir string)
int 21h                   ; Ejecuta el servicio del sistema operativo
```

Función 0AH: Lectura de caracteres introducidos por el periférico teclado

Descripción: Captura los caracteres introducidos por el teclado, los imprime por pantalla a modo de eco local (local echo) y se los transfiere posteriormente a la aplicación. La función finaliza cuando el usuario introduce el ASCII 13 (Retorno de carro). El sistema operativo permitirá la entrada de caracteres siempre que no se supere el máximo permitido definido en uno de los parámetros de entrada. En el caso contrario, estos no se mostrarán por pantalla ni se almacenarán.

Parámetros de entrada: DX = Desplazamiento en memoria desde el origen del segmento de datos donde se ubica el espacio reservado para capturar los caracteres. En el primer byte de dicho espacio reservado se ha de almacenar el número de caracteres que se desea capturar.

Parámetros de salida: En el segundo byte de la zona de memoria apuntada por DX el sistema operativo almacena el número de caracteres almacenados. A partir del tercer byte, el sistema operativo almacena los caracteres capturados.

Registros afectados: Ninguno.

Ejemplo:

```
MOV AH,0AH                ;Función captura de teclado
MOV DX,OFFSET NOMBRE      ;Area de memoria reservada = etiqueta NOMBRE
MOV NOMBRE[0],60          ;Lectura de caracteres máxima=60
INT 21H
```

En NOMBRE[1] el sistema operativo almacenará el número de caracteres registrados.

Función 4C00H: Fin de programa

Descripción: Esta función indica al sistema operativo que el proceso(programa) ha finalizado correctamente.

Parámetros de entrada: Ninguno.

Parámetros de salida: Ninguno.

Registros afectados: Ninguno.

Ejemplo:

```
mov ax, 4C00h    ; Fin de programa
int 21h          ; Ejecuta el servicio del sistema operativo
```

pract2a.asm

```
; Autor: XXXX
; Grupo XXXX
; Práctica 2, ejercicio a

; DEFINICION DEL SEGMENTO DE DATOS
DATOS SEGMENT
	; DEFINICION DE LOS VECTORES
	vector1 db 1,2,2,4
	vector2 db 4,2,5,1
	vector3 db 3,2,4,1
	errorNumDif DB "FUERA DEL CONJUNTO: 1,2,3,4", 13, 10, '$'
	salidaCorrecta DB "CORRECTO", 13, 10, '$'

DATOS ENDS
;*************************************************************************
; DEFINICION DEL SEGMENTO DE PILA
PILA SEGMENT STACK "STACK"
	DB 40H DUP (0) ;ejemplo de inicialización, 64 bytes inicializados a 0
PILA ENDS
;*************************************************************************
; DEFINICION DEL SEGMENTO EXTRA
EXTRA SEGMENT
	RESULT DW 0,0 ;ejemplo de inicialización. 2 PALABRAS (4 BYTES)
EXTRA ENDS
;*************************************************************************
; DEFINICION DEL SEGMENTO DE CODIGO
CODE SEGMENT
ASSUME CS: CODE, DS: DATOS, ES: EXTRA, SS: PILA
; COMIENZO DEL PROCEDIMIENTO PRINCIPAL
INICIO PROC
```

```
; INICIALIZA LOS REGISTROS DE SEGMENTO CON SU VALOR
        MOV AX, DATOS
        MOV DS, AX
        MOV AX, PILA
        MOV SS, AX
        MOV AX, EXTRA
        MOV ES, AX
        MOV SP, 64 ; CARGA EL PUNTERO DE PILA CON EL VALOR MAS ALTO
        ; FIN DE LAS INICIALIZACIONES
        ; COMIENZO DEL PROGRAMA


INICIO ENDP


;ESPACIO PARA SUBRUTINAS


; FIN DEL SEGMENTO DE CODIGO
CODE ENDS
; FIN DEL PROGRAMA INDICANDO DONDE COMIENZA LA EJECUCION
END INICIO
```