SISTEMAS OPERATIVOS

Lenguaje ensamblador en arquitecturas 8086/8088 parte 2

Elaborado por: Ukranio Coronilla

Ejercicio 1
Agregue al final de los tres datos definidos en el segmento de datos (líneas 5, 6 y 7) los Siguientes datos:

```
FLD1DB DB ?
FLD2DB DB 32
FLD3DB DB 20H
FLD4DB DB 01011001B
FLD5DB DB 10 DUP(0)
FLD6DB DB 'SISTEMAS OPERATIVOS'
FLD7DB DB '32654'
FLD8DB DB 01, 'ENERO', 02, 'FEBRERO', 03, 'MARZO'
```

Escriba como quedarían almacenados en memoria (hexadecimal) estos datos y después compruebe sus resultados utilizando turbo debugger.

Ejercicio 2
Añada al final de los datos anteriores los siguientes:

```
FLD1DW DW 0FFF0H
FLD2DW DW 01011001B
FLD3DW DW FLD7DB
FLD4DW DW 3, 4, 7, 8, 9
FLD5DW DW 5 DUP(0)
FLD1DD DD ?
FLD2DD DD 32574
FLD3DD DD 14, 49
FLD4DD DD FLD3DB - FLD2DB
FLD5DD DD 'PC'
FLD1DQ DQ ?
FLD2DQ DQ 04D47H
FLD3DQ DQ 32572
```

Escriba como quedarían almacenados en memoria (hexadecimal) estos datos y después compruebe sus resultados utilizando turbo debugger (Observe que FLD3DW contiene la dirección de memoria donde se almacena el dato FLD7DB definido con anterioridad). ¿Qué valor almacena FLD4DD? Explique qué sucede en este caso. ¿Qué sucede si se intenta inicializar un dato DW con 65536? ¿Porque? ¿Cuál es el máximo valor permitido?

En la línea 18 del programa PROG1_1.ASM, se solicita una interrupción INT 21H. La instrucción INT interrumpe el procesamiento y accede a la tabla de servicios de interrupción para determinar la

dirección de la rutina solicitada. Con dicha dirección se transfiere el control al DOS o al BIOS (dependiendo del número de interrupción) para ejecutar la rutina, y finalmente regresa a la instrucción posterior a INT.
Véase: https://es.wikipedia.org/wiki/Interrupci%C3%B3n

INT 21H es una interrupción de DOS que antes de ejecutarse, checa el valor almacenado en el registro AH cuyo valor especifica la acción que se va a realizar. En este caso, en la línea 17 le estamos asignando el valor 0x4C al registro AH, este valor corresponde a una petición para terminar de ejecutar un programa.

Finalmente el segmento de pila se define en las líneas 1, 2 y 3. Hemos definido una pila de 32 datos inicializados todos a 16, donde cada elemento de la pila es de 2 bytes. El registro SS contiene la dirección de inicio de la pila y SP contiene el desplazamiento que apunta al elemento tope de la pila. Verifique esto en la ventana segmento de pila.

Otros registros importantes del CPU que aparecen en la ventana registros son:

BP Facilita la referencia de parámetros, los cuales son datos y direcciones transmitidas vía la pila.
CS Almacena la dirección de inicio del segmento de código.
DS Almacena la dirección de inicio del segmento de datos.
SS Almacena la dirección de inicio del segmento de pila.
ES Almacena la dirección de inicio del segmento extra.
SI Se asocia con DS y se usa para operaciones con cadenas de caracteres.
DI Se asocia con ES y se usa para operaciones con cadenas de caracteres.

Ahora vayamos a la ventana correspondiente al segmento de pila. Inicie la ejecución paso a paso (F7), observe que alguno(os) valor(es) en la pila se van modificando, vea que algunos de los datos almacenados son los registros. Esta pila se utiliza por las funciones que son llamadas en el programa principal. Si el tamaño de la pila es muy pequeño, la ejecución del programa puede sufrir una detención total. Se recomiendan 32 elementos DW en la pila al menos para aplicaciones típicas.

Para observar un manejo sencillo de la pila, cree el ejecutable del siguiente programa en C (PROG2_1.C). Para crear el ejecutable baje el compilador Turbo C disponible en la computadora del profesor como TC.zip

```
1 #include <stdio.h>
2 void main(void)
3 {
4 int a=20000, b=40000, c;
5 c = a + b;
6 }
```

Abra el programa ejecutable con turbo debugger, aparecerá el código fuente del programa (Es importante que los archivos PROG2_1.C PROG2_1.EXE asociados se encuentren en la misma carpeta de turbo debugger). Para ver el código ensamblador, en el menú VIEW elija CPU. Observe que cada línea de código en C dentro de la función principal, está identificada con su número de línea y se forma por una o más instrucciones en lenguaje ensamblador.
Por ejemplo la línea 4 se describe así:

```
#PROG2_1 4#: int a==20000, b=40000, c;
```

Y se codifica en lenguaje ensamblador como sigue:

```
MOV WORD PTR [BP-02], 4E20
MOV WORD PTR [BP-04], 9C40
```

(Estas instrucciones pueden variar un poco respecto a su sistema) Observe que la primera instrucción a ejecutarse es:

```
PUSH BP
```

La instrucción PUSH disminuye el SP hacia la siguiente palabra de la pila y coloca un valor ahí. La instrucción POP obtiene la palabra almacenada en el tope de la pila e incrementa SP hacia la siguiente palabra. La instrucción WORD PTR únicamente restringe un dato a una longitud de 2 bytes. La instrucción SUB, resta el segundo parámetro del primero y el resultado se almacena en el primer parámetro.

Ejercicio 3

Escriba los siete bytes que se ubican en el tope de la pila y ejecute con F7 paso a paso hasta llegar a la última línea de nuestro código fuente. Observe los cambios que se suceden en los registros y en la pila. ¿Qué utilidad tiene la pila y para qué sirve SP?

Finalmente en la ventana de registros se exhiben 8 de los 16 bits de registros de banderas e indican los casos de: desbordamiento, resultados de comparación, paridad, acarreo, etc...