SIMULACIÓN DE MULTITAREA (TIEMPO COMPARTIDO)

Temas:

1. Interrupciones externas (de hardware)

   INT 08H : Cronómetro
   INT 09H : Teclado

2. Uso de la pila (stack) en las llamadas a procedimientos

   Registros SS, BP, SP

Programa principal:

1. Inicializar área de datos
2. Indicar el modo de video a utilizar (texto o gráfico), por medio de la interrupción correspondiente.
3. Preparar tantas pilas (stacks) como procedimientos quieran ejecutarse simultáneamente
4. Obtener las direcciones de las interrupciones de cronómetro y teclado y programar las direcciones nuevas en la tabla de interrupciones
5. Programar las nuevas interrupciones de:
   a. Cronómetro
   b. Teclado
6. Llamar a proceso 1, en el cual se entra a la simulación
   a. es el único proceso que se llama con CALL
   b. Este proceso debe tener el control de la salida del ciclo infinito, con ayuda de la nueva interrupción de teclado
   c. RECOMENDACIÓN IMPORTANTE: los procesos que se corren como multitasking no deben tener CALL a los mismos procedimientos, pues pueden perderse los parámetros, tampoco deben usar las mismas variables, y si desean hacerlo, ver más adelante como debe hacerse push y pop de estas variables.
7. Regresar a las interrupciones normales de:
   a. Cronómetro
   b. Teclado

1. Inicializar área de datos

```
RelojViejo      dd    ?          ; guarda direcciones antiguas para
TecladoViejo    dd    ?          ; ser restauradas
VectorBP        dw    4 dup(?)   ; una posición por proceso (4)
VectorSP        dw    4 dup(?)   ; son vectores de control circulares
Contador        db    0          ; contador de procesos
Teclazo         db    ?          ; bandera para terminar
```

2. Preparación de stacks

a. El tamaño total del segmento de stack (SS) es de 64K, dividirlo en partes iguales:

- Proceso 1: guardar SP y BP actuales
- Proceso 2: 1000H (BP y SP iniciales)
- Proceso 3: 2000H(BP y SP iniciales)
- Proceso 4: 3000H (BP y SP iniciales)

```
VectorBP    DW   4      DUP (?)
VectorSP    DW   4      DUP (?)
; por ejemplo, para el proceso 2
mov [vectorBP + 02], 1000H
mov [vectorSP + 02], 1000H
```

b. Guardar en variables temporales los registros SP, BP (se van a cambiar)

c. Para cada proceso vamos a armar las pilas artificiales de la siguiente forma:

- Actualizar BP y SP con los valores de las pilas (stacks) definidos en el paso a
- Inicializar AX, BX con el inicio del procedimiento, que es el mismo nombre con que lo invoca la instrucción CALL, utilizando los operadores SEG y OFFSET
  - Operador SEG: regresa la dirección del segmento en el que una variable o etiqueta especificada es colocada
    - Ej: ejemplo proc near
      - …
      - ejemplo endp
      - mov ax, seg ejemplo
  - Operador OFFSET: regresa la dirección de desplazamiento, es decir la dirección relativa dentro del segmento de datos, dentro de una variable o etiqueta
    - Ej: mov bx, offset ejemplo
  - Los pasos anteriores (operador SEG, operador OFFSET) pueden reemplazarse por las instrucciones, que evitan el paso intermedio por los registros AX,BX:
  - PUSH SEG EJEMPLO
  - PUSH OFFSET EJEMPLO

- Hacer push de estos registros:
  - Registro de banderas
  - Registro que contiene el segmento (seg)
  - Registro que contiene el desplazamiento (offset)
  - Hacer push de todos los registros (el correspondiente pop estará en la nueva interrupción de cronómetro)
  - A continuación todos los valores que se esperen como parámetros. Si ud. hace más push aquí, debe tener sus pop correspondientes en el proceso que está programando. Esto se recomienda en el caso que los procesos que corren simultáneamente estén compartiendo variables.

d. Guardar en los vectores de control (búfers circulares), el valor de SP para ese proceso. El valor de BP está siempre fijo, SP es el que va moviéndose

e. Recuperar SP y BP originales

3. Obtener direcciones de interrupciones normales para cronómetro y teclado y programar direcciones de nuevas interrupciones

08H: 20H Sistema de cronómetro (20H a 23H)
09H: 24H Interrupción de teclado (24H a 27H)

a. Cronómetro:

- Guardar registro DS pues va a modificarse
- Llamar a la función 35h interrupción 21h: obtener dirección de la interrupción:

```
mov  ah, 35h
mov  al, 08h
int  21h
```

- La interrupción regresa la dirección en ES:BX

```
RelojViejo  dd   ?
RelojNuevodd   ?
.....
mov  [word ptr RelojViejo+2], ES
mov  [word ptr RelojViejo], BX
```

- Colocar la dirección de la nueva interrupción en DX

```
push DS
mov AX, seg RelojNuevo
mov DS, AX
mov DX, offset RelojNuevo
```

- Llamar a la función 25h interrupción 21h (establecer dirección de la interrupción)

```
mov ah, 25h
mov al, 08h
int 21h
pop DS
```

Hacer lo mismo para el teclado

4. Programar nuevas interrupciones

Las nuevas interrupciones deben siempre comenzar con la instrucción:

CLI: Limpia bandera de interrupción, para deshabilitar interrupción externa enmascarable

Y terminar con la instrucción:

STI: Pone en uno la bandera de interrupción, para permitir interrupciones externas enmascarables

Además el retorno debe hacerse con la instrucción:

IRET: Proporciona un regreso lejano de una rutina de interrupción

Lo que hace el IRET es:

1. Saca de la pila la palabra de la parte superior al IP
2. Incrementa en dos el SP
3. Saca de la pila la parte superior al CS
4. Incrementa en dos el SP
5. Saca de la pila el registro de banderas

La nueva interrupción INT 08H (cronómetro):

- Debe hacer el cambio de proceso cada vez que ocurra la interrupción externa del cronómetro. Cuando se detecte un carácter en el teclado no se hará el cambio, sino que vuelve al proceso 1, el cual se encarga de salir del programa.

- El cambio de proceso puede controlarse por medio de un vector que contiene los BPs y SPs de cada proceso (recordar que son DW). Ese vector se creó en la preparación de stacks. Los pasos a seguir son:

  - cli ; Se asegura que no se interrumpa este procedimiento.
  - Push del registro de banderas
  - Llamar al reloj "viejo" (call dword ptr relojviejo), para que haga sus funciones normales
  - Hacer push de todos los registros
  - Mover un contador de procesos a SI (se inicializó a 0)
  - Guardar SP en el vectorSP del proceso actual que se está atendiendo:

```
mov [offset vectorSP + SI], SP
```

- Incrementar en 2 contador de procesos
- Mover contador de procesos a SI
- Cambiar el stack al nuevo proceso: con las nuevas posiciones del vector

```
mov SP, [offset vectorSP + SI]
mov BP, [offset vectorBP + SI]
```

- Hacer pop de los registros
- sti ; Enciende la bandera de interrupciones.
- iret ; retorno de interrupción

Debe verificarse cuando llegue al último proceso para que regrese al primero (búfer circular).

La nueva interrupción INT 09H (teclado):

La nueva interrupción:

```
Call   dword ptr TecladoViejo
Mov   teclazo, ------      ; variable de control para indicar que
                           ; se presionó cualquier tecla
```

Esta bandera nos va a controlar la salida del ciclo infinito de procesos.

Sugerencia: La lectura del teclado sin interrupciones puede hacerse utilizando la instrucción IN.

Referencias

- Ing. Luis Barrera
- Ing. Alejandro Gutiérrez
- Notas de clase de Martha Ligia Naranjo
- Auxiliares del curso desde 2002