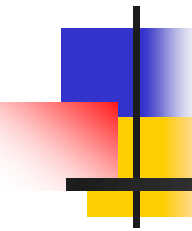


Sistemas de Procesamiento de Datos

Assembler (parte 5)

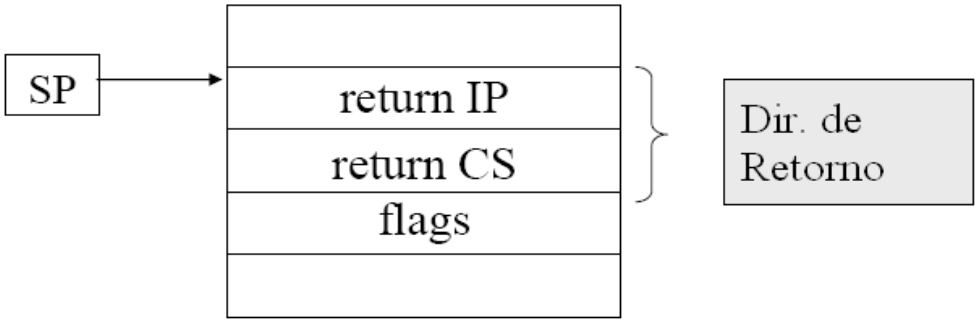
Profesor: Fabio Bruschetti

Aydte: Pedro Iriso

Ver 2019

Rutinas ISR (atención a interrupción)

- Instrucción CALL y las Interrupciones
  - Semantica del CALL
    - NEAR CALL *destino* — PUSH IP; IP**:=***destino*
    - FAR CALL *destino* — PUSH CS; PUSH IP; CS:IP**:=***destino*
  - Semantica de una Interrupción
    - Se genera un stack frame distinto al de las rutinas
    - Es el hardware quien escribe los siguientes valores en el stack frame
      - PUSH registro de FLAGS
      - bit IF = 0 en el registro FLAGS
      - bit TF = 0 en el registro FLAGS
      - PUSH CS
      - PUSH IP
      - CS := [0:n*4+2]
      - IP := [0:n*4]



Rutinas ISR (atención a interrupción)

- Regresando de una ISR
  - La instrucción RET utilizada para volver de un CALL no funciona
  - Se debe utilizar la instrucción IRET
    - Saca la dirección de retorno de 32 bits (CS:IP)
    - Saca los flags

Restaura los flags a los valores anteriores a limpiar IF=0 y TF=0

Ejemplo :

```
isr PROC FAR
   IRET
isr ENDP
```

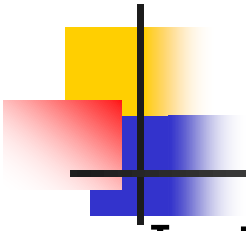
- Instalando un ISR
  - Las interrupciones solo funcionan si la dirección de la ISR han sido cargadas previamente en el vector correcto.

Rutinas ISR (atención a interrupción)

- Instalación dinámica de una ISR (primera aproximación)

```
miint_type    EQU     40          ; Se instala como vector
mivector      EQU     miint_type* 4
.code
miisrPROC FAR
    IRET
miisrENDP

main PROC
    MOV       AX, 0
    MOV       ES, AX              ; ES apunta a la tabla de vectores
    MOV       ES:[mivector], OFFSET miisr
    MOV       ES:[mivector+2], @code
    ...
    STI                           ; Ahora pueden ocurrir las interrupciones
```



Rutinas ISR (atención a interrupción)

- Instalación dinámica de una ISR
  - Cuando se instala una ISR, se está sobrescribiendo el valor anterior insertado en el arranque del sistema operativo.
  - Cuando se instalan vectores, primero se debe salvar el contenido existente del vector
    - Dichos valores se debe reponer cuando el programa termine
    - Si no se salvan y restablecen los valores, el S.O puede no funcionar correctamente.

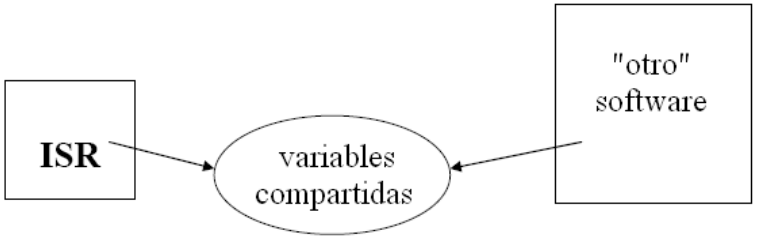
Rutinas ISR (atención a interrupción)

- Instalación dinámica de una ISR (más robusto)

```
.data
vector_viejo_offset             dw      ?
vector_viejo_segmento           dw      ?
.code
...
main PROC
    MOV     AX, 0
    MOV     ES, AX                      ; ES apunta a la tabla de vectores
    MOV     AX, ES:[mivector]
    MOV     vector_viejo_offset, AX
    MOV     AX, ES:[mivector+2]
    MOV     vector_viejo_segmento, AX
    MOV     ES:[mivector] , OFFSET miisr
    MOV     ES:[mivector+2] , @code
    ...
```

Interrupciones x Hardware

- Programación de las ISR
  - No puede pasarse parámetros a las ISR que atienden interrupciones x hardware
    - ***Compartir variables de Estado persistentes***
  - Interferencia:
    - Las ISR interfieren con el acceso a las variables compartidas
    - Cuando se accede a una variable compartida con una rutina ISR se puede tener:
      - que la ISR interrumpa en el medio de un acceso
      - que la ISR modifique el valor
      - Resultado: valor corrupto
    - Ejemplo
      - Programa principal quiere leer la variable **count_high**
      - ISR interrumpe al programa principal modifica **count_high** y **count_low**
      - ISR termina y el programa principal continúa leyendo la variable **count_low**
      - **Resultado: el progrma principal obtiene un valor corrupto de count**



Interrupciones x Hardware

- Regiones Críticas
  - Se trata de una sección de código donde se puede dar potenciales "interferencias"
  - La existencia de una región crítica no implica que existirá interferencia sino que en ese lugar existe una posibilidad / probabilidad. Por eso es potencial
  - Se debe poder reconocer las regiones críticas y prover protección a la solución durante el diseño.
- Protección de regiones críticas
  - Prevenir la interrupción de las regiones críticas
    - O sea eliminar la ocurrencia de interferencias
  - El software tiene una abilidad limitada de controlar ocurrencias de interrupciones
    - Puede prevenirlas enmascarando interrupciones
    - Se enmascaran usando CLI / STI
      - Prevenir todas las interrupciones --> overkill ☹
      - Mantener las regiones críticas cortas. ☺

Interrupciones x Hardware

- Protección de regiones críticas
  - Enmascarando interrupciones en el controlador de interrupciones mediante el registro mascara. Recordar que el controlador de interrupciones comparte recursos y puede inhibir algunas interrupciones, pero aún permitir a otras ocurrir.
  - Enmascarando interrupciones en los dispositivos ocurre lo mismo que en el caso anterior, sin involucrar modificaciones al controlador de interrupciones
- Posibles regiones críticas
  - Instalando la rutina ISR
    - Se modifica la tabla de vectores (es un recurso compartido)
  - Las rutinas ISR y el resguardo de registros
    - Las rutinas deben salvar todos los registros a usar y restablecerlos dado que los programas interrumpidos no pueden salvar registros antes de una interrupción.
  - Las rutinas ISR y el Fin de Interrupción EOI
    - ISR deben enviar el EOI al controlador de interrupciones, si no lo envia o envían más de un EOI provocan comportamientos no deseados

Interrupciones x Hardware

- Habilitando interrupciones
  - Es deseable que las interrupciones existan y sean generadas para que se ejecuten las rutinas ISR
  - Si IF=0 entonce NINGUNA interrupción ocurrirá
  - La instrucción IRET es ejecutada por el hardware ISR, IF será puesta en 1 cuando los FLAGS se restauren
  - Las interrupciones de prioridad menor o igual no serán enviadas por el controlador de interrupciones hasta que un EOI sea recibido.
- Errores comunes
  - Olvidarse de salvar/restaurar TODOS los registros que modifique la ISR
  - Olvidarse de inicializar DS antes de acceder a las variables
  - Olvidarse de enviar un EOI al controlador de interrupciones
  - Olvidarse de habilitar las interrupciones
    - STI en el controlador de interrupciones y/o en los dispositivos

Interrupciones x Software

- Una Interrupción por software es una llamada especial a un procedimiento previamente definido como parte del S.O., también conocido como TRAP o System Call
  - Implementado usando el mecanismo de hardware de interrupciones ISR
- Ejemplos:
  - Funciones de DOS: (Int 21H)
    - Imprimir un mensaje: DS:DX = dir. Mensaje; AH = 9; int 21h
    - Salir AH = 4C; int 21h
    - Leer un carácter AH = 1; int 21h --> AL =caracter ASCII
    - Imprimir un carácter DL = caracter ASCII ; AH = 2; int 21h
    - Salida a Impresora DL = caracter ASCII ; AH = 5; int 21h
  - Servicios del BIOS
    - Display (Int 10H)
    - Funciones de E/S a discos (Int 13H)
    - Teclado (Int 16H) [Lectura AH=0]
    - Impresoras (Int 17H) para funciones de impresora

Interrupciones x Software

- Subrutinas vs Interrupciones x software
  - Las subrutinas son parte del programa
    - desarrolladas, compiladas, linkeadas y cargadas
      - Durante la compilación, el compilador determina la dirección destino
      - Durante la linkedición, son juntadas con otros software
      - Durante la carga, los valores de segmentos son inicializados (distintos cada vez)
    - Se debe regenerar la aplicación completa si se modifica parte de programa o librería
  - Los Traps son como subrutinas
    - Se transfiere el control a una actividad encapsulada terminando con un regreso al punto de invocación
  - Los Traps son distintos a las subrutinas
    - No son parte del programa aplicación
    - No son parte del desarrollo de la aplicación
      - El S.O. no es una librería
      - El código OBJ no se linkedita con los códigos OBJ del SO.
  - El S.O. Es un residente permanente en memoria mientras que una aplicación es un residente transitorio en memoria

Interrupciones x Software

- Invocación de las funciones del S.O.
  - Primera aproximación: Localizar cada procedimiento del SO en ubicaciones globales reservadas
    - Las ubicaciones son publicadas como parte del S.O.
    - Modo de direccionamieto directo a memoria
    - Problema cuando una función/procedimieno del S.O. es revisada/modificada.
  - Segunda aproximación: Ubicar un arreglo de direcciones en una ubicación global reservada conteniendo las direcciones de los procedimientos del SO
    - Modo de direccionamieto indirecto a memoria
    - El SO pone punteros a sus procedimientos en variables cuando se carga el SO.
    - El desarrollo de programas asumen que estas variables globales existen
- Instrucción INT
  - Las rutinas por software se basan en la aproximación de punteros almacenados, ofreciendo "invocación dinámica de subrutinas" por la instalación de nuevos punteros.
    - Subrutinas son identificadas por su label, las interrupciones por su tipo
    - Subrutinas son llamadas por CALL , las instrucciones por INT

Interrupciones

- Interrupciones por software: invocadas por la instrucción INT
- Interrupciones por hardware: invocadas por eventos externos de E/S
- Ambos usan la misma semantica:
  - Push el registro de FLAGS
  - Limpiar IF y TF bits en los FLAGS
  - Push CS e IP
  - Trae nuevo CS:IP de la tabla de interrupciones: (0:n*4+2):(0:n*4)
- Diferencia entre subrutinas e interrupciones x software

```
; subroutine initialization
        none
        . . .
; call set up
        push arguments
        CALL    subr
        ADD     SP, 2*numArgs
        . . .
subr:
        standard entry code
        access param's  [ BP + 4+ ]
        standard exit code
        RET
```

n=0..255

```
; ISR initialization
        install address at 0:4*n
        . . .
; call set up
        push arguments
        INT     n
        ADD     SP, 2*numArgs
        . . .
subr:
        standard entry code
        access param's  [ BP + 8+ ]
        standard exit code
        IRET
```

Interrupciones

- Como puede manejar varios servicios una rutina ISR?
  - El tipo de servicio es pasado como parámetro
  - Los parámetros son pasados por registros, no por el stack.
  - El valor de retorno, depende del parámetro de código de servicio.
- Tabla de Vectores en Intel 8086

Interrupt Types	**Descriptions**
0 … 1F	Reserved by Intel Includes : BIOS 10-16h
20h	Terminate a COM program
21-24h	DOS Functions
33h	Mouse Functions
60-6Bh	Available for Applications
80-F0h	Reserved
F1-FFh	Available for Applications

Interrupciones

- Formas de cambiar el vector de interrupciones
  - Por DOS
    - MOV AH,25h ; servicio para cambiar vector
    - MOV AL,vector ; entre 0 y 255
    - LEA DX,rutina ; DS:DX nueva rutina de gestión
    - INT 21h ; llamar al DOS
  - Por ASSEMBLER
    - MOV BL,vector*4 ; vector a cambiar en BL
    - MOV BH,0 ; ahora en BX
    - MOV AX,0
    - PUSH DS ; preservar DS
    - MOV DS,AX ; apuntar al segmento 0000
    - LEA DX,rutina ; CS:DX nueva rutina de gestión
    - CLI ; evitar posible interrupción
    - MOV [BX],DX ; cambiar vector (offset)
    - MOV [BX+2],CS ; cambiar vector (segmento)
    - STI ; permitir interrupciones
    - POP DS ; restaurar DS