Aula 9 - Estudo Avançado de Assembly
Arquitetura de Computadores I

Prof. MSC. Wagner Guimarães Al-Alam

Universidade Federal do Ceará
Campus de Quixadá

2020-1

Interrupt Handling

- Visão Geral
- Interrupções de Hardware
- Instruções de Controle de Interrupção
- Escrevendo um Manipulador de Interrupções Customizado
- Programas Residentes após o Término

Introdução

Definição

Interrupção é um mecanismo onde o controle de fluxo de um programa pode ser alterado.

- Vimos outros 2 mecanismos que possibilitam a transferência de controle:
  - Saltos: provêm um meio de transferir o controle;
  - Procedimentos: além do salto, possuem um mecanismo de retorno ao ponto onde o procedimento foi chamado.

Introdução

- Procedimentos
  - Executados a partir de uma chamada de software
  - Salta para um trecho de código do segmento de código do próprio programa que chamou e executa
  - Após concluir o procedimento, a execução retorna à instrução subsequente à instrução CALL que iniciou a execução do procedimento.
  - Existe somente durante a execução do programa principal
- Interrupções
  - Executadas a partir de software ou de hardware
  - Interrompe um programa qualquer em execução
  - Salta para um trecho de código de um programa residente na memória e executa
  - Após concluir, a execução retorna à mesma instrução do programa que foi interrompido.

Introdução

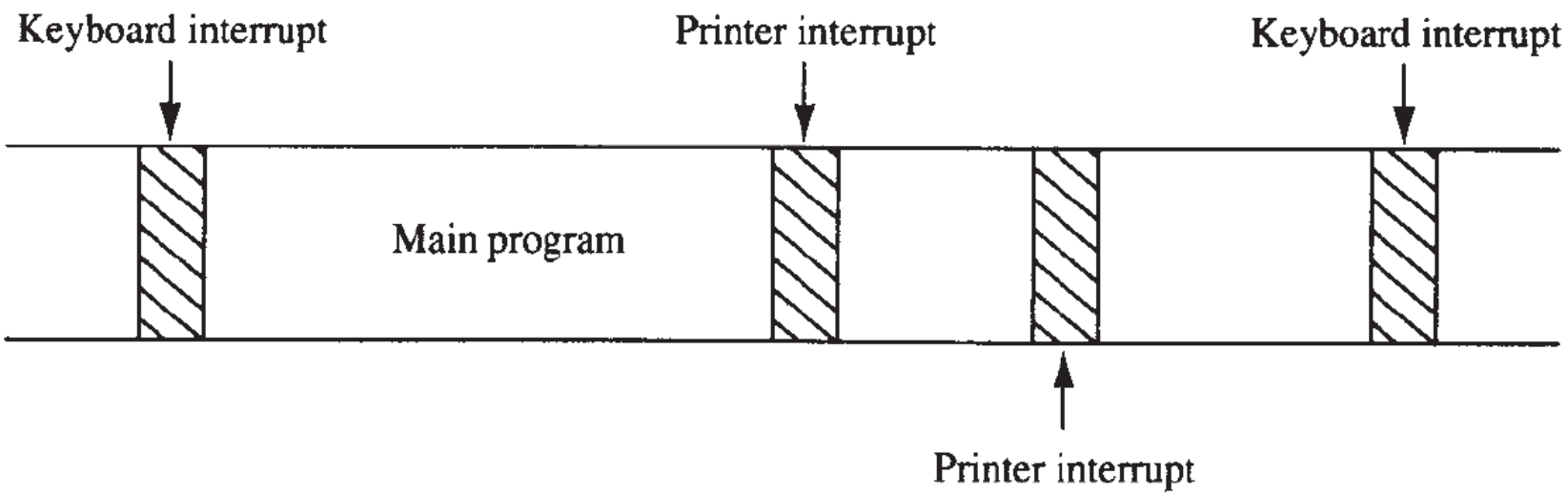
- Tratador de Interrupções (interrrupt service routine - ISR) - realiza tarefas comuns de I/O
  - pode ser chamado como funções
  - póde ser ativado por eventos de hardware
- Exemplos:
  - tratador de saída de vídeo
  - tratador de erro crítico
  - tratador de teclado
  - tratador de divisão por zero
  - tratador de Ctrl-Break
  - entrada e saída por porta serial

Note

Interrupções são particularmente úteis quando estão fazendo interfaceamento de dispositivos de I/O que preveem e requerem dados a uma baixa taxa de transferência.

Exemplo - Teclado

Supondo que uma pessoa digita uma tecla por segundo, seria um desperdício de tempo o processador esperar a digitação, enquanto há uma fila de instruções para executar no CPU.



A figura ilustra um cenário em que, devido à existência de interrupções, o CPU executa processamento enquanto dispositivos lentos de I/O executam suas tarefas.

Interrupções Dedicadas no Microprocessador

TYPE 0 Erro de divisão, ocorre em caso de overflow na divisão ou resultado de divisão por zero.

TYPE 1 Single-step ou trap ocorre após a execução de cada instrução se a flag de trap (TF) está ativada. Ao aceitar a interrupção, TF é limpa e o procedimento volta a executar na velocidade máxima.

TYPE 2 A interrupção non-maskable ocorre quando é colocado 1 no pino de entrada NMI. Ser non-maskable significa que não pode ser desabilitada.

TYPE 3 A instrução especial de 1 byte (INT 3) que usa esse vetor para acessar seu processamento do serviço de interrupção. A instrução INT 3 é frequentemente usada para armazenar um ponto de parada no processo de debugging do programa.

TYPE 4 Overflow é um ISR especial usando com a instrução **INTO**. A instrução INTO interrompe o programa se uma condição de overflow existe, conforme refletido pela flag de overflow (OF).

TYPE 5 A instrução **BOUND** compara um registrador com os limites armazenados na memória. Se o conteúdo do registrador é maior ou igual à primeira palavra na memória e menor ou igual à segunda palavra, não ocorre interrupção. Se o conteúdo do registrador está fora do limite (out of bounds), uma interrupção do tipo 5 ocorre.

Interrupções Dedicadas no Microprocessador

TYPE 6 Uma interrupção de opcode inválido ocorre sempre que um opcode indefinido é encontrado no programa.

TYPE 7 Uma interrupção de coprocessador não disponível ocorre quando um coprocessador não é encontrado no sistema.

TYPE 8 Uma interrupção de dupla falta é ativada sempre que duas interrupções separadas ocorrerem durante a mesma instrução.

TYPE 9 Exceder o segmento do coprocessador ocorre se um operando de memória da instrução ESC (coprocessor opcode) exceder o endereço FFFFH no modo real.

TYPE 10 Interrupção de estado de segmento inválido (modo protegido).

TYPE 11 A interrupção de segmento não presente ocorre no modo protegido quando o bit P está limpo ($P = 0$) no descritor, indicando que o segmento não está presente na memória ou não é válido.

Interrupções Dedicadas no Microprocessador

TYPE 12 A interrupção de pilha não presente ocorre no modo protegido quando o bit P está limpo (P = 0) no descritor, indicando que a pilha não está presente na memória ou o limite do segmento de pilha foi excedido.

TYPE 13 Uma falta geral de proteção ocorre para maior parte das violações no modo protegido do sistema 80286–Core2.

TYPE 14 Interrupção de page fault ocorre para qualquer página que falte na memória.

TYPE 16 Erro no coprocessador.

TYPE 17 Checagem de alinhamento indica que words e doublewords estão em uma posição de memória ímpar (word) ou por localização incorreta no caso de (doubleword).

TYPE 18 Checagem de máquina ativa a interrupção do sistema de gerenciamento de memória no microprocessador Pentium–Core2.

Operações

- Chamadas
  - INT N: Chama uma interrupção de software do vetor de interrupções.
  - INT 3: Usada como ponto de parada (breakpoint em deputradores), útil em depuração
- Condicionais
  - Bound: BOUND AX,DATA verifica se um registrador (AX) está dentro do limite definido por duas palavras DATA e DATA+1
  - INTO: Verifica se a flag de overflow está ativa (OF = 1) e gera uma interrupção 4. Se OF=0, nada ocorre e segue para a próxima operação.
- Retorno
  - IRET: Semelhante ao RET pois também retira o endereço de retorno da pilha. Retira 6 bytes da pilha: dois para IP, dois para CS, e dois para as flags. Para 32 bits existe o IRETD, e para 64 bits o IRETQ.

Tipos de Interrupção

- As interrupções podem ser geradas por exceção, software ou por hardware.

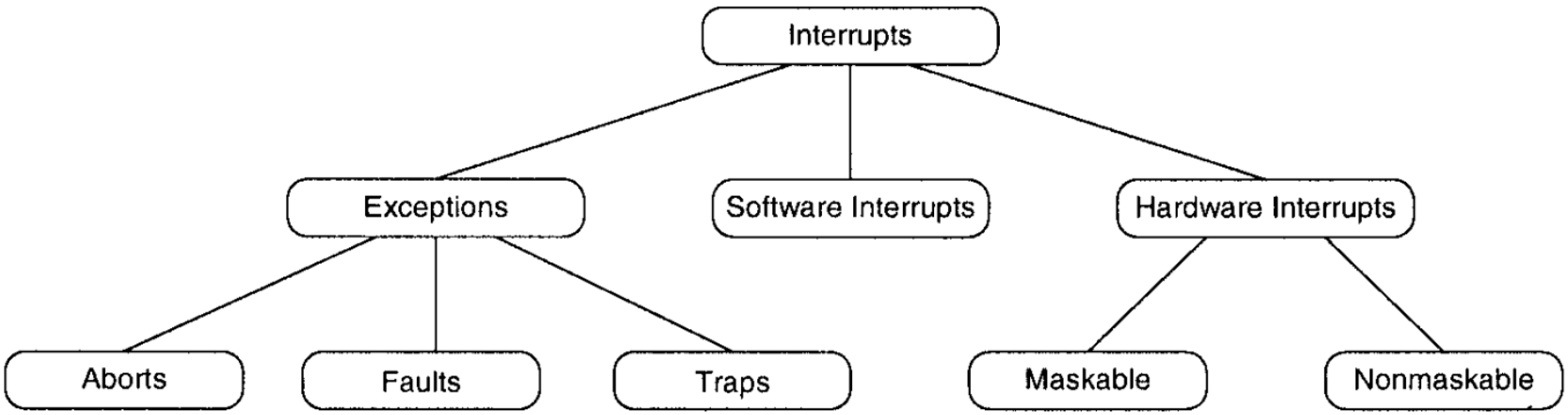


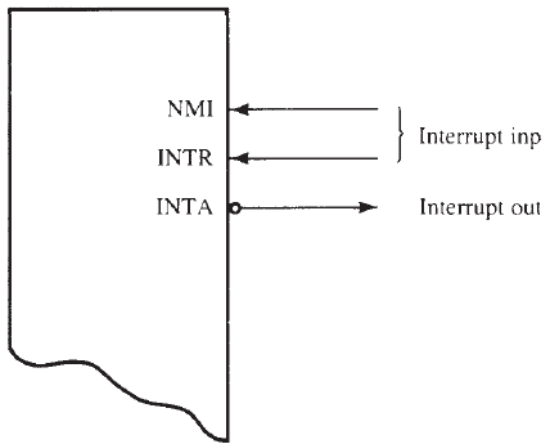
Figure 20.1 A taxonomy of the IA-32 interrupts.

Interrupções de Hardware

- Geradas pelo controlador programável de interrupção (Programmable Interrupt Contoller - PIC) Intel 8259
  - em resposta a um sinal de hardware
- Níveis de requisição de interrupções (Interrupt Request Levels - IRQ)
  - escalonador de interrupção baseado em prioridade
  - intermedia requisições de interrupções simultâneas
  - previne interrupções de baixa prioridade de interromper um interrupção de alta prioridade

Pinos de Interrupção

FIGURE 12–5 The interrupt pins on all versions of the Intel microprocessor.



- INTR - Interrupt Request
- INTA - interrupt acknowledge
- NMI - Non-maskable interrupt

Atribuições comuns de IRQ

IRQ	Interrupt Number	Description
0	8	System timer (18.2 times/second)
1	9	Keyboard
2	0Ah	Programmable Interrupt Controller
3	0Bh	COM2 (serial port 2)
4	0Ch	COM1 (serial port 1)
5	0Dh	LPT2 (parallel port 2)
6	0Eh	Floppy disk controller
7	0Fh	LPT1 (parallel port 1)
8	70h	CMOS real-time clock
9	71h	(Redirected to INT 0Ah)
10	72h	(Available) sound card
11	73h	(Available) SCSI card
12	74h	PS/2 mouse
13	75h	Math coprocessor
14	76h	Hard disk controller
15	77h	(Available)

Exceções

- Faults:
  - Ocorre limitados à antes do inicio da execução da instrução.
  - Instrução pode ser reiniciada. Ex: Falha de falta de segmento na memória
- Traps:
  - Ocorre limitados à logo após a execução da instrução.
  - A instrução não é reiniciada. Ex: Interrupções de depuração
- Aborts:
  - São exceções que reportam erro severo e o programa é encerrado
  - Erro na verificação de bit de paridade de leitura da memória

Modo Real x Modo Protegido

- Interrupções podem ser usadas no modo real ou no modo protegido:
- **Modo Real:**
  - Primeiras 1000 posições da memória guardam a tabela de vetores de interrupções (IVT)
- **Modo Protegido:**
  - Ao invés das posições do vetor de interrupções, é endereçado um descritor de interrupção, chamando-se assim de Tabela de Descritores de Interrupção (IDT)

Tabela de Vetores de Interrupção

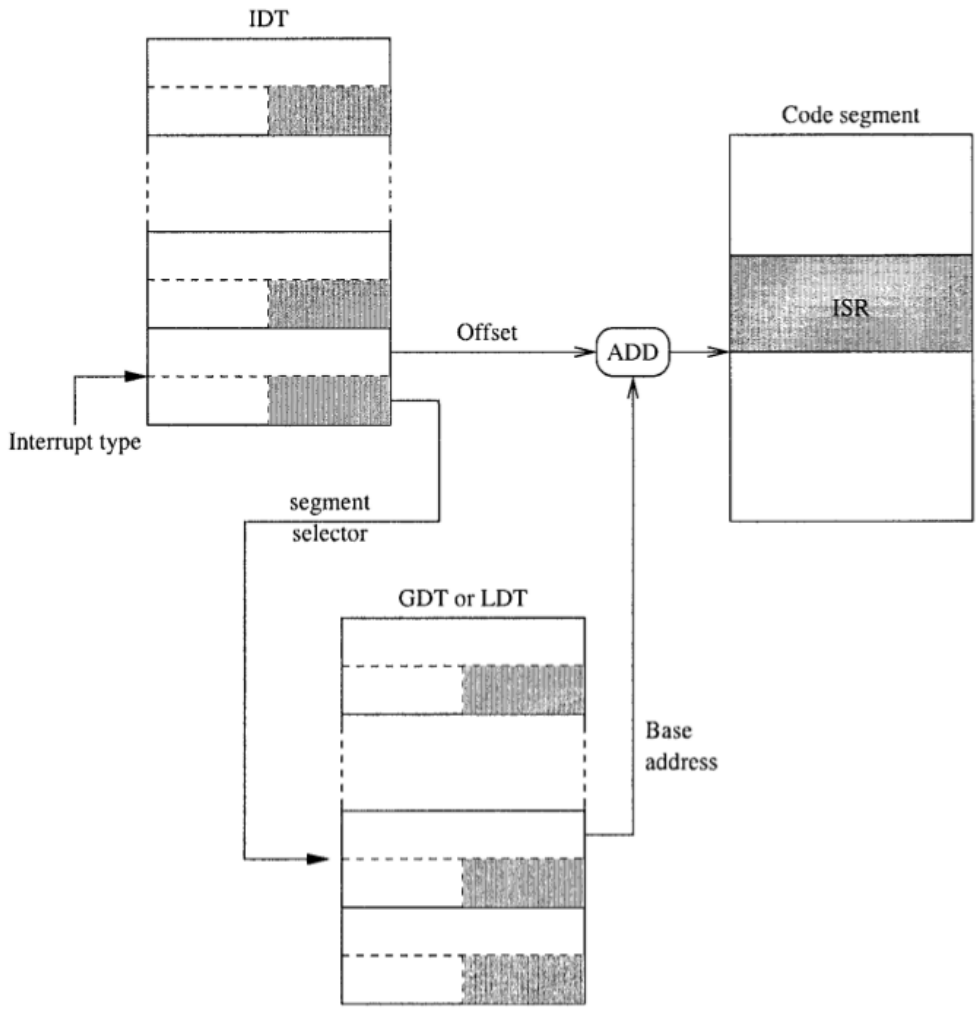
Interrupt Vector Table (IVT)

- Tabela armazenada em memória qu mantém o endereço do manipulador de interrupção
- Localizada nos primeiros 1024 bytes da memória (000000H–0003FFH) ou Armazenada no Registrador IDTR (48 bits - 32 base e 16 limite)
- Cada entrada contém um endereço de 32-bit segment/offset que aponta para uma rotina do serviço de interrupção
- Offset = interruptNumber * 4
- Exemplo:

Interrupt Number	Offset	Interrupt Vectors
00-03	0000	02C1:5186 0070:0C67 0DAD:2C1B 0070:0C67
04-07	0010	0070:0C67 F000:FF54 F000:837B F000:837B
08-0B	0020	0D70:022C 0DAD:2BAD 0070:0325 0070:039F
0C-0F	0030	0070:0419 0070:0493 0070:050D 0070:0C67
10-13	0040	C000:0CD7 F000:F84D F000:F841 0070:237D

Invocação de Interrução no Modo Protegido

Interrupt Descriptor Table - IDT



Instruções de Controle de Interrupção

- STI – set interrupt flag
  - ativa interrupções externas
  - sempre é executada no início do tratador de interrupções
- CLI – clear interrupt flag
  - desabilita interrupções externas
  - usada antes de uma seção crítica de código que não podem ser interrompidas
  - suspende o mecanismo de temporização do escalonador do SO

A Operação de Interrupções em Modo Real

Quando o microprocessador completa a execução da instrução corrente, verifica as interrupções na seguinte ordem: (1) instruction executions, (2) single-step, (3) NMI, (4) coprocessor segment overrun, (5) INTR, and (6) INT. Se uma ou mais condições está presente, a seguinte sequência de eventos ocorre:

1. O conteúdo do registrador de flags é empilhado na pilha de execução.
2. Ambas flags IF(interrupt flag) e TF(trap flag) são limpas. Isso desabilita o pino INTR e as funcionalidades de trap ou passo simples.
3. O conteúdo do registrador de segmento é empilhado na pilha de execução.
4. O conteúdo do ponteiro de instrução (IP) é empilhado na pilha de execução.
5. O conteúdo do vetor de interrupção é buscado e então são aplicado aos registradores IP e CS de modo que a próxima instrução execute o procedimento do serviço de interrupção endereçado pelo vetor de interrupção.

Escrevendo um Tratador de Interrupção Personalizado

- Motivação
  - Mudar o comportamento de um tratador pré-existente
  - Corrigir um bug no tratador existente
  - Melhorar a segurança do sistema desabilitando alguns comandos de teclado
- O que está envolvido
  - Escrever um novo tratador
  - Carregar na memória
  - Sobrescrever a entrada na tabela do vetor de interrupção
  - Encadear ao tratador de interrupção existente(usualmente)

Pegar o Vetor de Interrupção

- INT 21h Function 35h – Pegar o vetor de interrupção
  - retorna o offset do endereço do tratador no segmento em ES:BX

```
1  .data
2  int9Save LABEL WORD
3  DWORD ?              ; store old INT 9 address here
4  .code
5  mov ah,35h           ; get interrupt vector
6  mov al,9             ; for INT 9
7  int 21h              ; call MS-DOS
8  mov int9Save,BX      ; store the offset
9  mov [int9Save+2],ES  ; store the segment
```

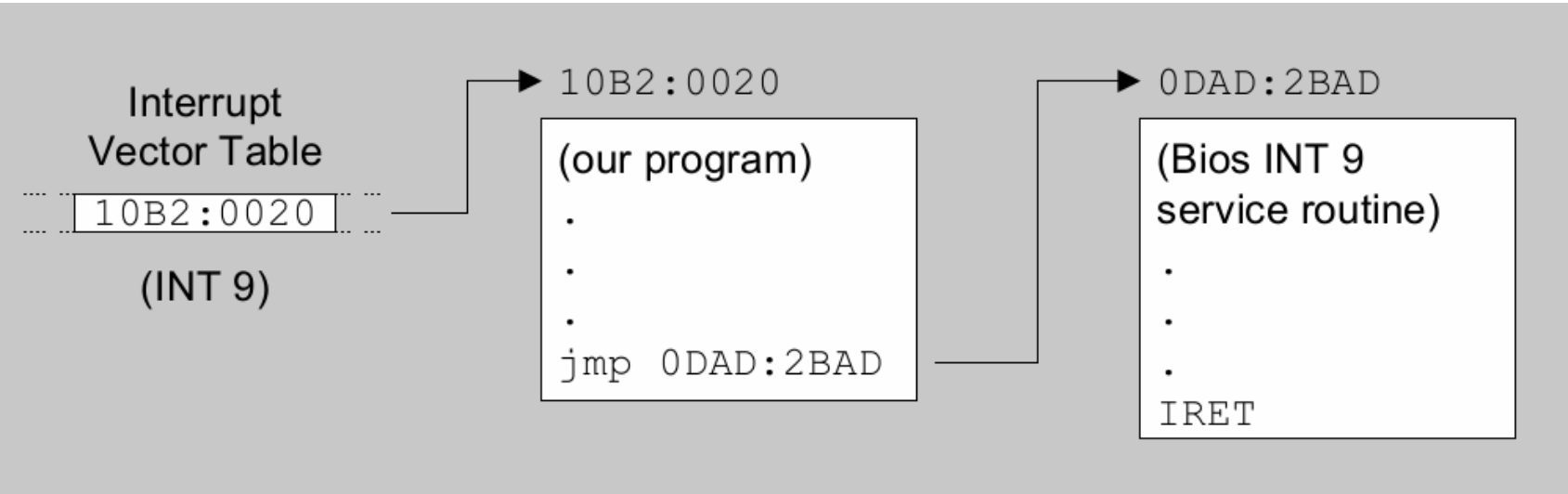
Definir o Vetor de Interrupção

- INT 21h Function 25h – Definir o vetor de interrupção
  - instala um novo tratador de interrupção apontado por DS:DX

```
1      mov ax,SEG kybd_rtn          ;keyboard handler
2      mov ds,ax                    ;segment
3      mov dx,OFFSET kybd_rtn       ;offset
4      mov ah,25h                   ;set Interrupt vector
5      mov al,9h                    ;for INT 9h
6      int 21h
7      .
8      .
9      kybd_rtn PROC                ;(new handler begins here)
```

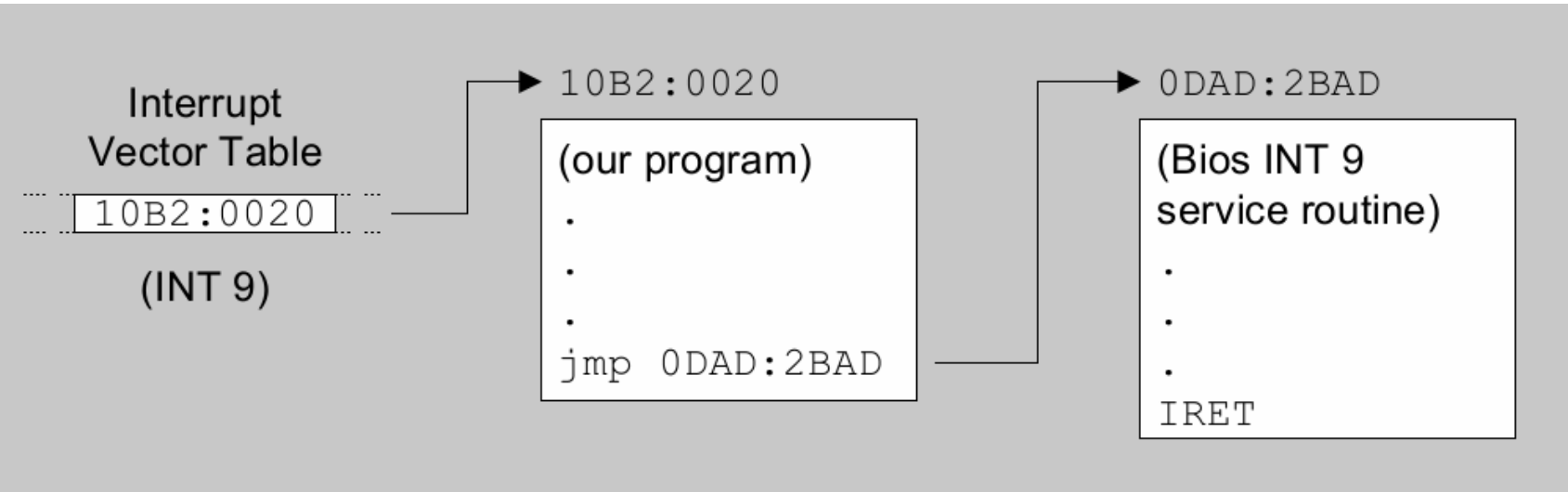
Passos no Processamento de Teclado

1. Tecla pressionada, byte enviado por hardware para a porta do teclado
2. O controlado 8259 interrompe a CPU, passando seu número de interrupção
3. a CPU encontra a entrada 9h da tabela do tratador de interrupção e salta a CPU para o endereço encontrado



Passos no Processamento de Teclado

4. Nosso tratador executa, interceptando o byte enviado pelo teclado
5. Nosso tratador salta para o tratador "INT 9" regular
6. O tratador "INT 9h" finaliza e retorna
7. O sistema continua o processamento normal



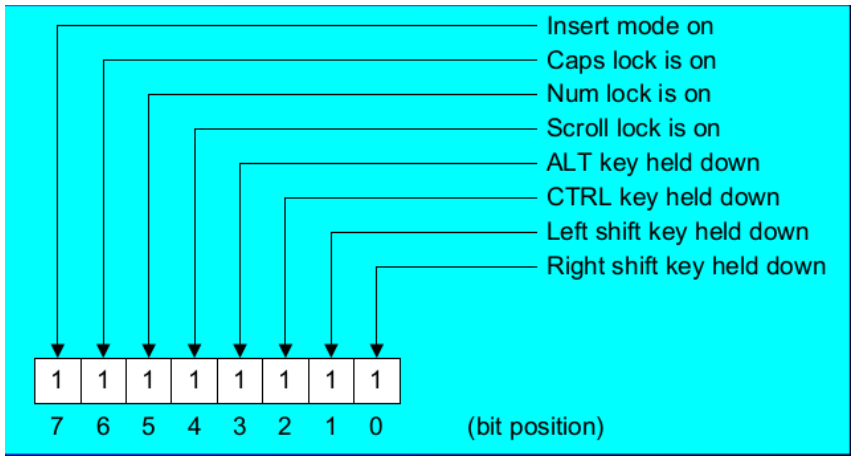
Programas que Terminam e Continuam Residente em Memória

- (terminate and stay resident - TSR): Instalado na memória, continua residente até ser removido por:
  - um programa de remoção
  - reinicialização
- Exemplo de teclado
  - substitui o vetor "INT 9"apontando para o nosso próprio tratador
  - verifica ou filtra certas combinações de pressionamento de teclas usando nosso tratador
  - encadeia no final o tratador "INT 9"existente para o processamento normal do teclado

Programa de Não Reiniciar (1 / 5)

- Inspeciona cada tecla que chega
- Se a tecla Del é recebida:
  - verifica as teclas Ctrl e Alt
  - permite o reset do sistema somente se a tecla Shift da direita também está pressionada

O byte de status do teclado indica o estado corrente das teclas especiais:



Programa de Não Reiniciar (2 / 5)

- Visualiza o código fonte
- Programa residente inicia com:

```
1     int9_handler PROC FAR
2     sti                 ;enable hardware interrupts
3     pushf               ;save regs & flags
4     push es
5     push ax
6     push di
```

Programa de Não Reiniciar (3 / 5)

- Localiza o byte de flag de teclado e copia para AH:

```
1     L1:
2     mov ax,40h              ; DOS data segment is at 40h
3     mov es,ax
4     mov di,17h              ; location of keyboard flag
5     mov ah,es:[di]          ; copy keyboard flag into AH
```

- Verifica para ver se as teclas Ctrl e Alt estão pressionadas:

```
1     L2:
2     test ah,ctrl_key        ; Ctrl key held down?
3     jz L5                   ; no: exit
4     test ah,alt_key         ; ALT key held down?
5     jz L5                   ; no: exit
```

Programa de Não Reiniciar (4 / 5)

- Testa as teclas Del e Shift da direita:

```
1    L3:
2    in al,kybd_port        ; read keyboard port
3    cmp al,del_key         ; Del key pressed?
4    jne L5                 ; no: exit
5    test ah,rt_shift       ; right shift key pressed?
6    jnz L5                 ; yes: allow system reset
```

- Desabilita a tecla Ctrl e escreve o byte de flag do teclado de volta na memória:

```
1    L4: and ah,NOT ctrl_key    ; turn off bit for CTRL
2    mov es:[di],ah             ; store keyboard_flag
```

Programa de Não Reiniciar (5 / 5)

- Desempilha as flags e registradores e executa o salto longo para a rotina INT 9h existente na BIOS:

```
1    L5: pop di                    ; restore regs & flags
2    pop ax
3    pop es
4    popf
5    jmp cs:[old_interrupt9]       ; jump to INT 9 routine
```

Exemplo Acesso a Arquivos Usando Interrupção

- Acesso a disco é semelhante ao acesso a teclado e display
- Quando usamos um SO, temos o auxílio de um Sistema de Arquivos, que gerencia o baixo nível do acesso e proporciona interrupções de software para propiciar o acesso aos arquivos de maneira fácil e padronizada.
- Descritor de arquivo Aberto
  - Quando um arquivo é aberto ou criado, a partir de seu nome, o descritor de arquio de 16 bits é retornado
  - Chamadas de Sistema
    - open
    - create
- Ponteiro de Arquivo Aberto
  - Associado a cada arquivo aberto.
  - Offset da posição corrente relativo ao início do arquivo, em bytes.
  - Quando arquivo é aberto, seu valor é zero.

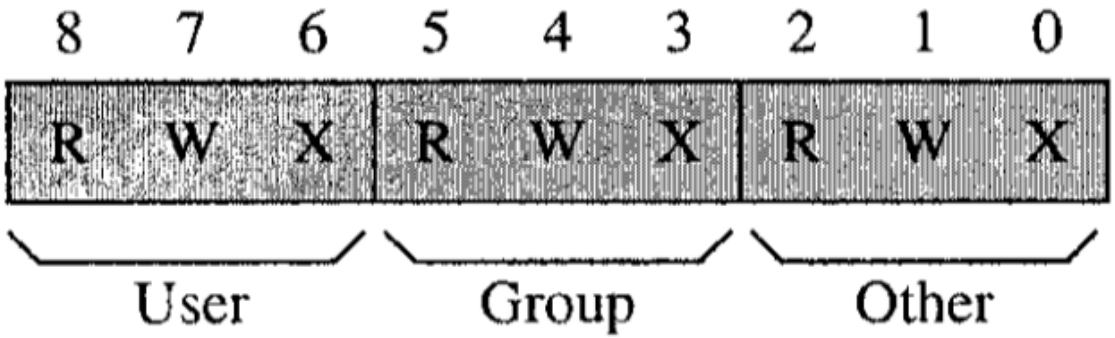
Chamadas de Sistema de Arquivos

- Criar um arquivo novo: (system call 8)
- Abrir um arquivo existente: (system call 5)
- Ler dado: (system call 3)
- Escrever dado: (system call 4)
- Fechar o arquivo: (system call 6)
- Atualizar o ponteiro de um arquivo aberto: (system call 19)

Novo Arquivo

System call 8 — Create and open a file

Inputs:	EAX	=	8
	EBX	=	file name
	ECX	=	file permissions
Returns:	EAX	=	file descriptor
Error:	EAX	=	error code



Um erro é representado por um inteiro negativo em EAX

Abrir Arquivo

System call 5 — Open a file

Inputs:	EAX	=	5
	EBX	=	file name
	ECX	=	file access mode
	EDX	=	file permissions
Returns:	EAX	=	file descriptor
Error:	EAX	=	error code

Modo de acesso do arquivo

- read-only (0)
- write-only (1)
- read-write (2)

Ler Dado

System call 3 — Read from a file

Inputs:	EAX	=	3
	EBX	=	file descriptor
	ECX	=	pointer to input buffer
	EDX	=	buffer size (maximum number of bytes to read)
Returns:	EAX	=	number of bytes read
Error:	EAX	=	error code

Escrever Dado

System call 4 — Write to a file

Inputs:	EAX	=	4
	EBX	=	file descriptor
	ECX	=	pointer to output buffer
	EDX	=	buffer size (number bytes to write)
Returns:	EAX	=	number of bytes written
Error:	EAX	=	error code

Fechar o Arquivo

System call 6 — Close a file

Inputs:	EAX	=	6
	EBX	=	file descriptor
Returns:	EAX	=	—
Error:	EAX	=	error code

Atualizando o Ponteiro de um Arquivo Aberto

System call 19 — lseek (Updates file pointer)

Inputs:	EAX	=	19
	EBX	=	file descriptor
	ECX	=	offset
	EDX	=	whence
Returns:	EAX	=	byte offset from the beginning of file
Error:	EAX	=	error code

Whence:

- Início do arquivo (0)
- Posição corrente (1)
- Final do arquivo (2)