T-19
1 de Junho de 2010

Sumário:

Acesso directo à memória (DMA).

Processadores de entrada/saída

Interrupções por software

Excepções. Single step

Bibliografia:

S. Dandamudi, *Fundamentals of Computer Organization and Design*, Springer 2003, 19.4.2, 20.2, 20.4, 20.5

Vantagens das interrupções

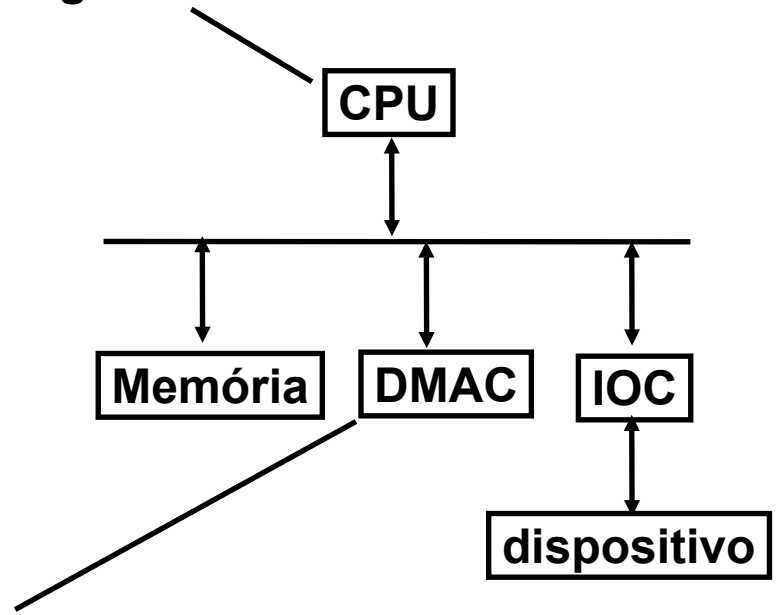
- O uso de interrupções permite sobrepor (overlap) a computação e a realização de entradas/saídas
- Permitem adaptar a velocidade relativa de CPUs e periféricos
- Um computador que usa interrupções permite programar mais facilmente as operações de entrada/saída e tem mais desempenho do que um que não tem

Acesso directo à memória
Direct Memory Access (DMA)

- A programação de E/S ainda requer uma intervenção activa do CPU
  - Ritmo de transferência limitado
  - CPU precisa de intervir muitas vezes
- A resposta a estes problemas é o Acesso Directo à Memória (Direct Memory Access)
- Requer hardware extra no “bus” de E/S
- O controlador com DMA também pode “tomar conta” do bus

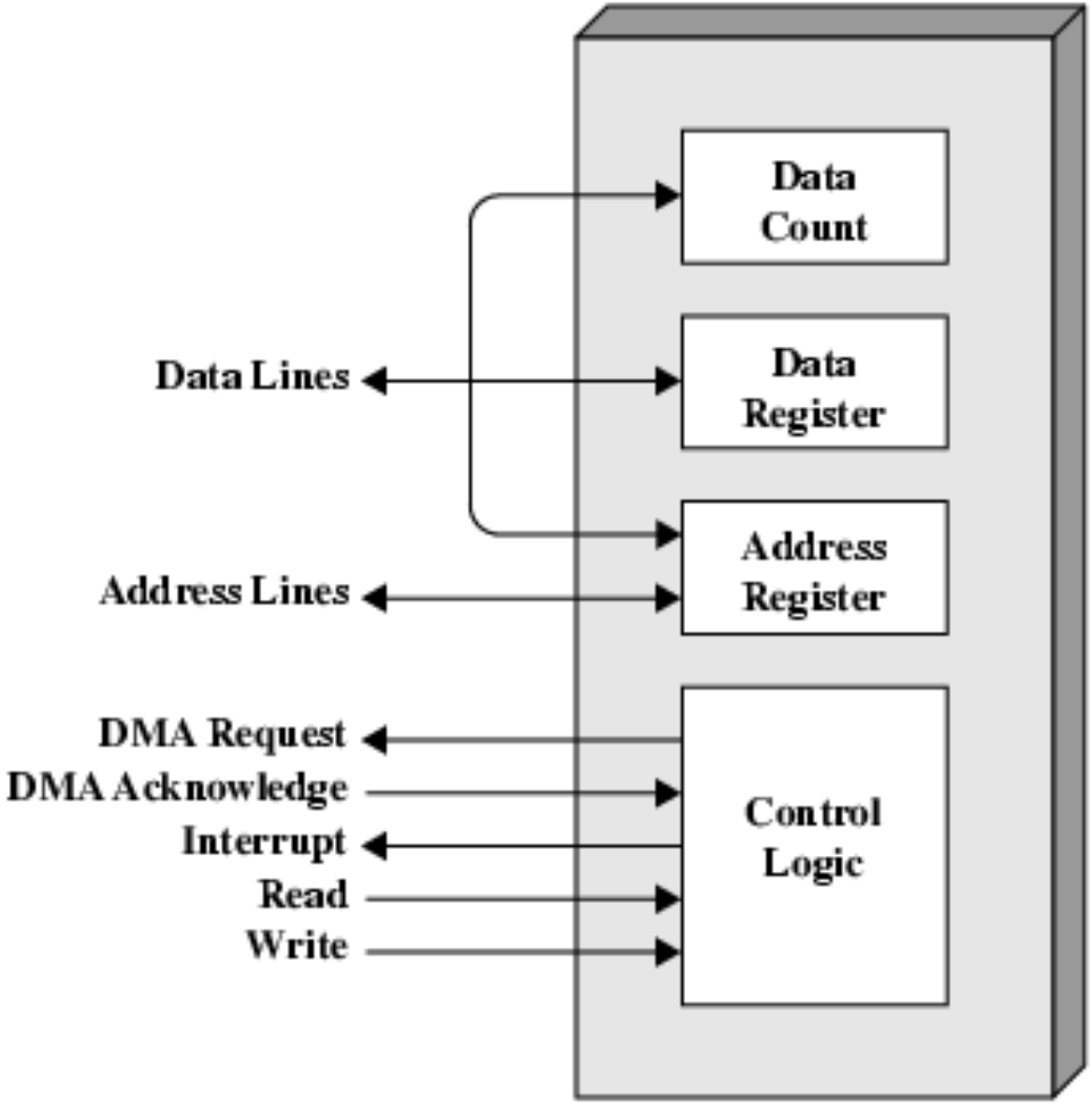
Direct Memory Access

O CPU envia a direcção, end. inicial e nº de bytes ao controlador de DMA (DMAC). A seguir envia "start".



DMAC fornece sinais de “handshake” para o controlador do periférico e sinais de “handshake” e endereços à memória central

Módulo para realizacão de DMA



Operação do DMA

- O CPU informa o controlador de DMA sobre a transferência:
  - Leitura ou escrita
  - Endereço do dispositivo
  - Endereço do bloco de memória central onde estão / para onde vão os dados a transferir
  - Número de bytes a transferir
- O CPU continua com o processamento
- O controlador de DMA trata da transferência
- O controlador de DMA envia uma interrupção quando termina

O “roubo” do bus (cycle stealing) pelo controlador de DMA

- O controlador de DMA ocupa o “bus” durante um ciclo; durante esse ciclo transfere uma palavra (largura do “bus” de E/S)
- Não é uma interrupção
  - O CPU não interrompe o que está a fazer (não há chamada de rotina de interrupção)
- O CPU espera pelo acesso ao bus
  - i.e. quando faz “fetch” ou lê ou escreve um dado
  - Só na RAM; na cache não há problema
- Atrasa o CPU mas muito menos do que se fosse ele a fazer a transferência de dados do controlador

Direct Memory Access

Sem DMA

Tempo para fazer 1000 transfs a 1 msec cada:

1 preparação de DMA @ 50 µsec
Transferir 1000 bytes @ 1000 µsec
1 interrupção @ 2 µsec
1 rotina de interrupção @ 48 µsec

.0011 segs de tempo de CPU

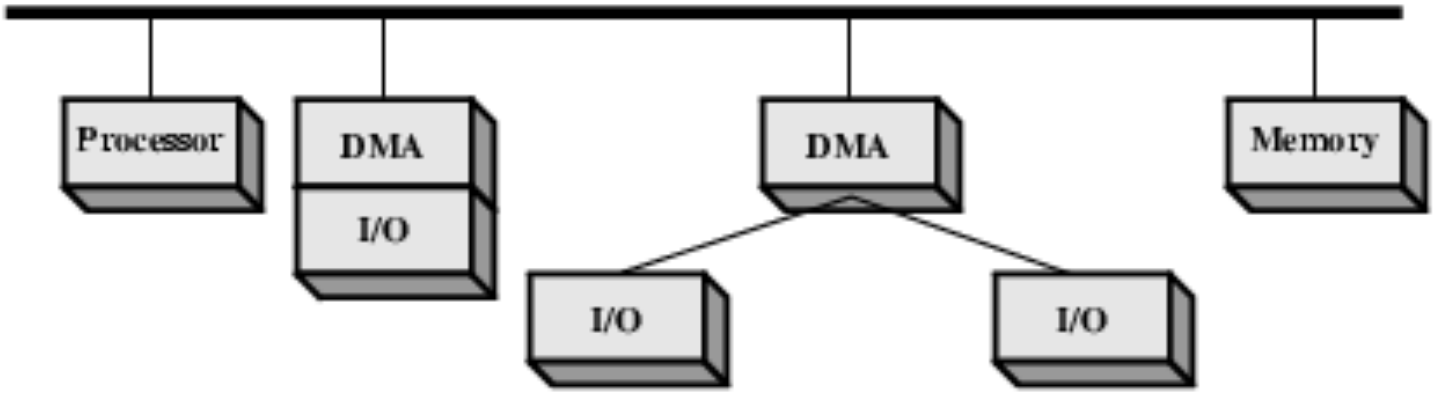
Com DMA

Tempo para fazer 1000 transfs a 1 msec cada:

1 preparação de DMA @ 50 µsec
1 interrupção @ 2 µsec
1 rotina de interrupção @ 48 µsec

.0001 segs de tempo de CPU

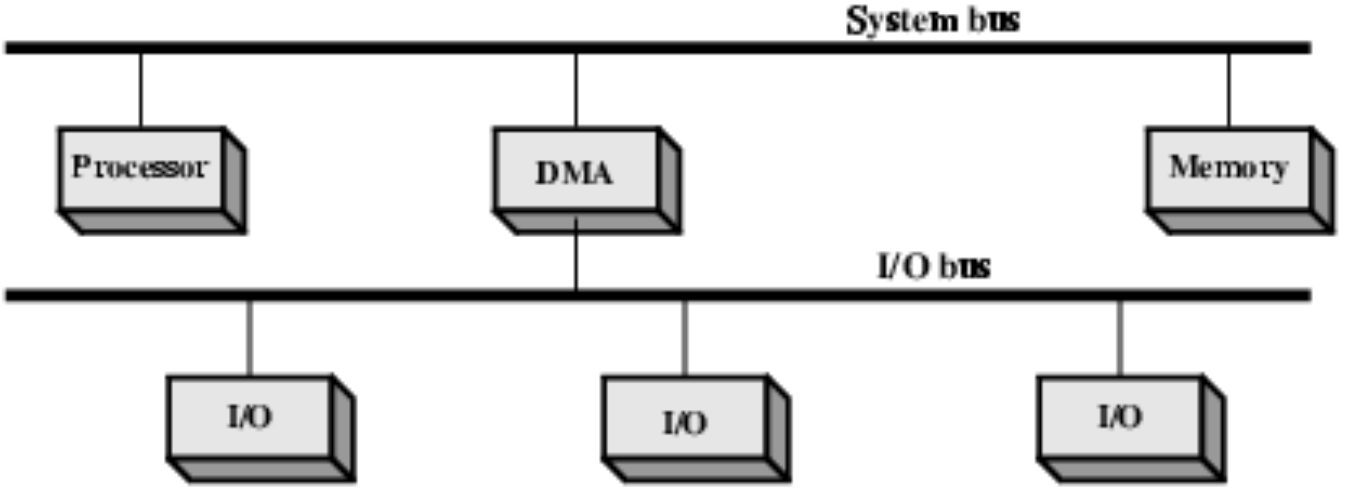
Configurações de DMA (1)



(b) Single-bus, Integrated DMA-I/O

- Bus único, Controlador de DMA integrado no controlador de periférico
- Controlador pode suportar mais de um dispositivo
- Cada transferência só usa o bus uma vez
  - DMA para a memória, CPU é travado uma vez

Configuração de DMA (2)



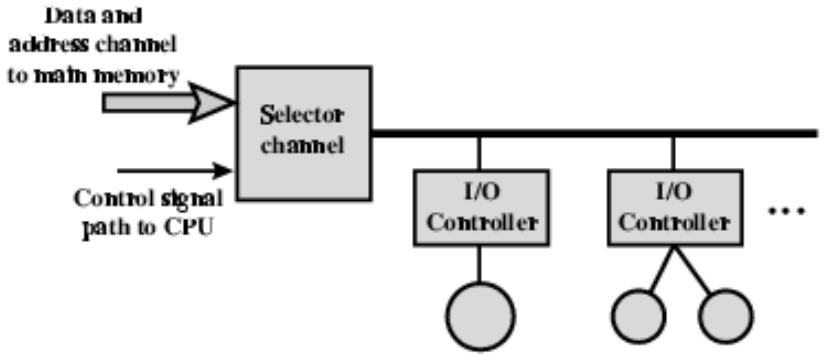
(c) I/O bus

- Controlador de DMA partilhado por vários controladores de periféricos
- Cada transferência usa o bus uma vez
  - DMA para a memória; CPU é travado uma vez

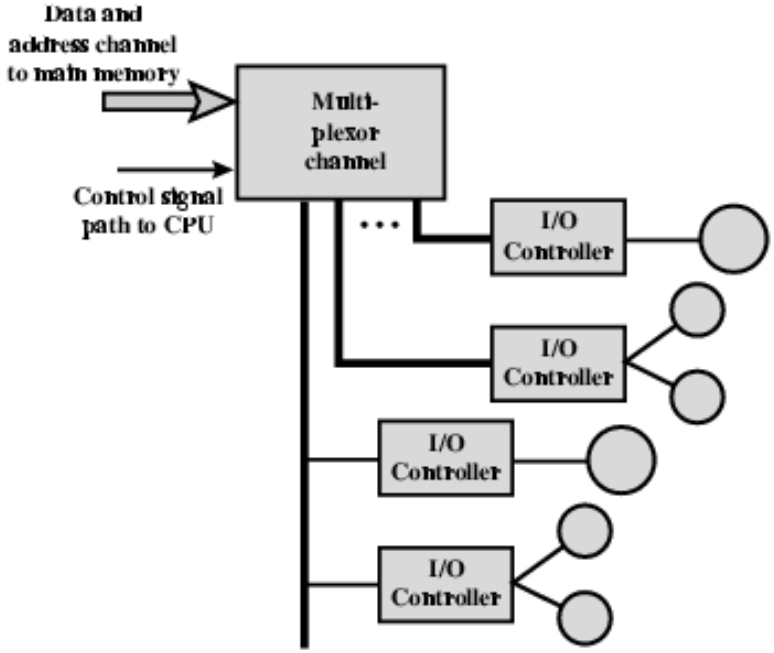
Processadores de I/O ou processadores de canal

- Os dispositivos de E/S são cada vez mais sofisticados; e.g. controladores gráficos 3D
- CPU programa o controlador de E/S para fazer a transferência
- O controlador de E/S faz a transferência completa
- Melhora a velocidade
  - Retira carga ao CPU
  - Um processador dedicado tem potencial para ser mais rápido

Arquitectura de canais de E/S



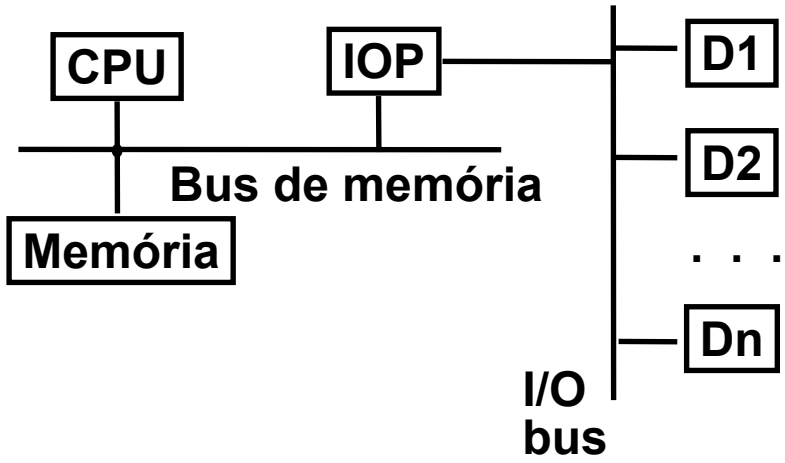
(a) Selector

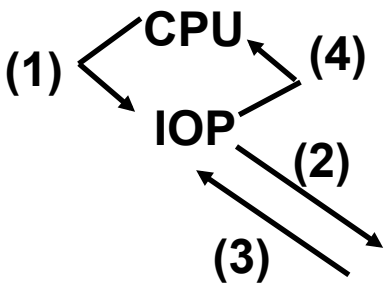


(b) Multiplexor

Exemplo:
"mainframes" IBM

Processadores de E/S (IOP)





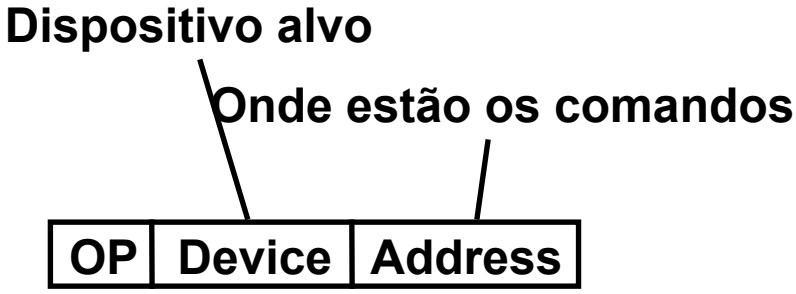
Envia instrução ao IOP

Interrompe quando acaba

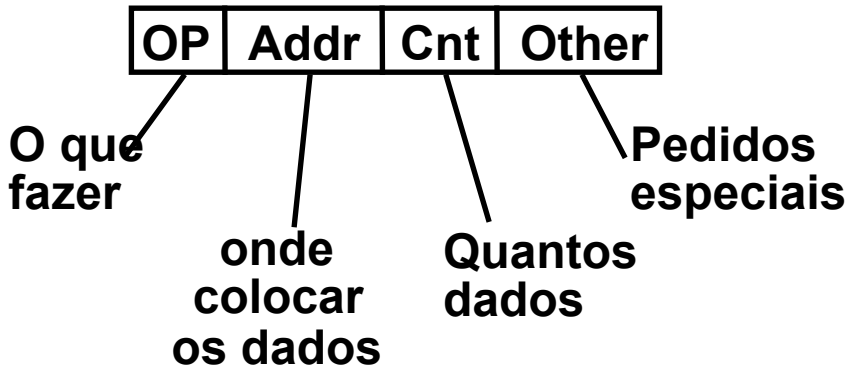
memória

Transferência de/para a memória são controladas directamente pelo IOP.

O IOP "rouba" ciclos de memória



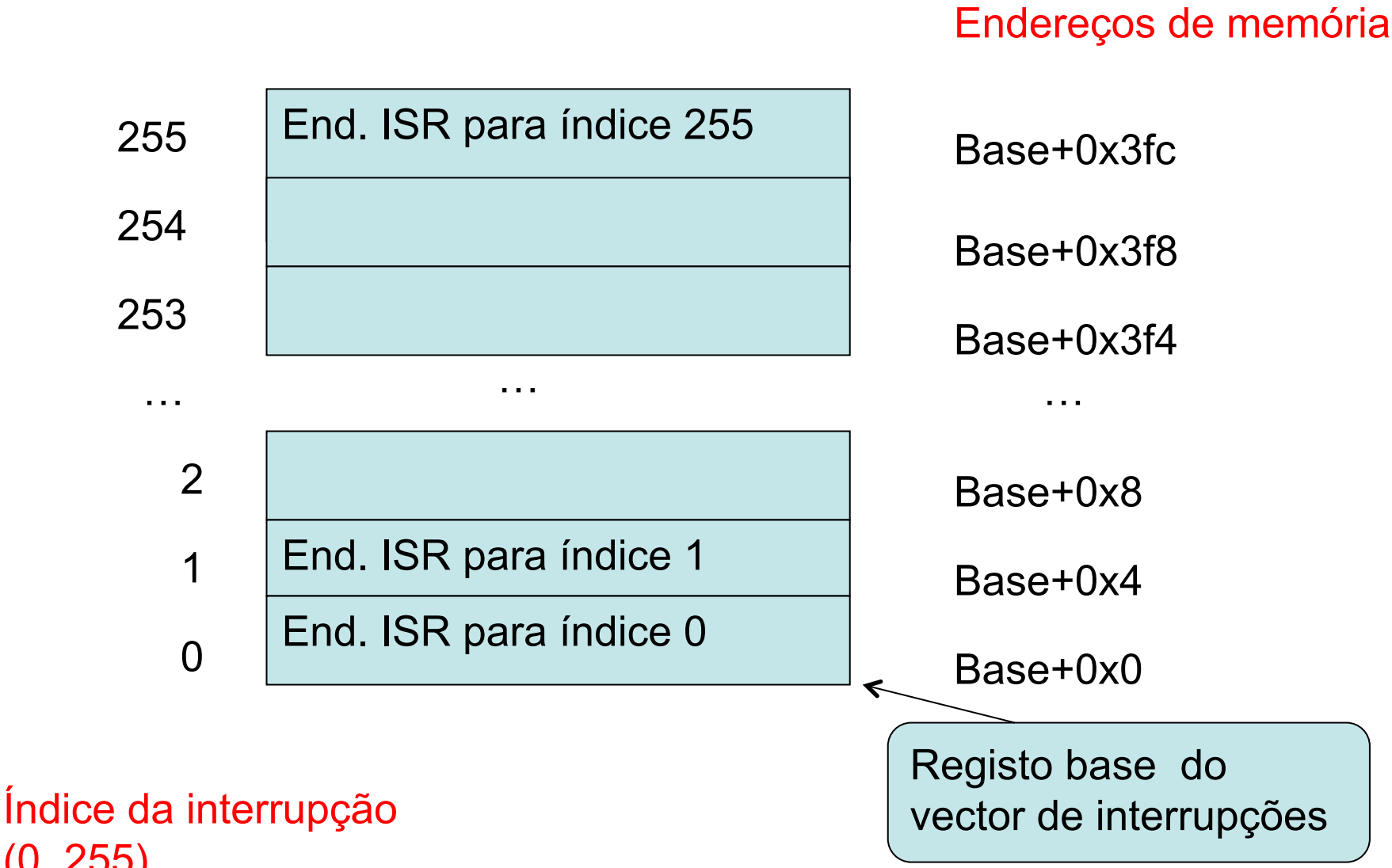
Procurar comandos em memória



Taxonomia das Interrupções

- Interrupções por hardware
  - Exterior do CPU
  - Mascaráveis e não mascaráveis
- Interrupções software
  - Para provocar deliberadamente uma troca de ambiente de execução (por ex. chamar o sistema)
- Excepções
  - Ligadas a algo que aconteceu na instrução máquina corrente

Vector de interrupções



Interrupções por software

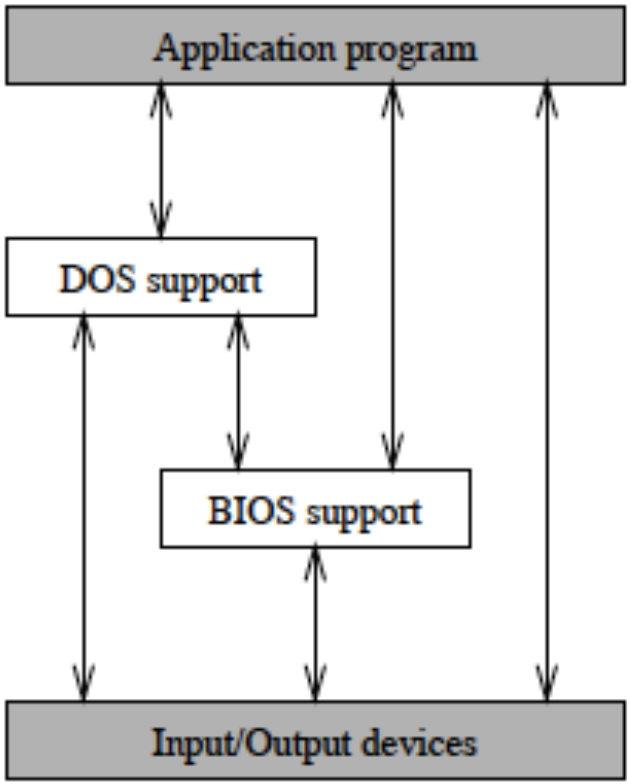
- Desencadeada pela execução da instrução máquina

  `int  nº-interrupção`

  nº interrupção é um inteirro de 0 a 255 (Pentium)
- Cada tipo de interrupção pode ser usado para invocar diferentes tipo de serviços.
- Exemplo as chamadas ao sistema são invocadas executando int *21H*
  - Há mais de 80 pedidos de serviço diferente
  - O registo AH é usado para identificar o tipo de serviço

Exemplo DOS: Ler teclado

- Pode ser usado o sistema (**`int 21H`**) ou a BIOS (**`int 10H`**)



Chamadas ao sistema no MS-DOS

Vector de interrupções da arquitectura x86

```
; imprimir uma mensagem na consola

MSG DB ´Hello, world . $´
...
mov dx, offset MSG
mov ah,9                    ; código da função
int  21h
```

Salta para o endereço definido na entrada 21H do vector de interrupções. Mantém no mesmo modo de funcionamento. Não há protecção do sistema.

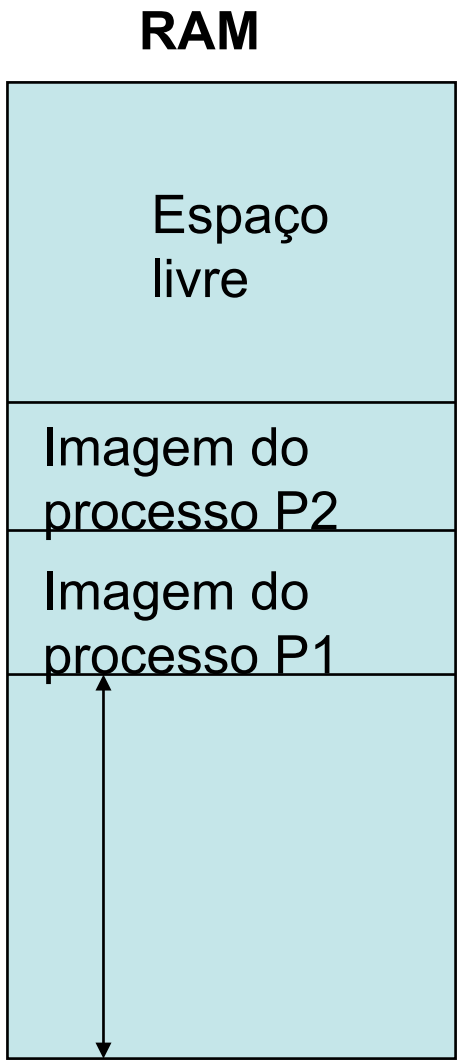
Chamadas ao sistema

- Exemplo típico (UNIX, POSIX)

```
#include <errno.h>
…
retval = read ( fd, buffer, nbytes);
if (retval == -1) switch(errno){
                        case EIO:  printf( … ); break;
                        case EBADF: printf( … ); break;
                        …
                  }
```

- Sistema de execução da linguagem (neste caso libc …)
  - Prepara um conjunto de parâmetros no stack ou em registos do CPU
    - Código da operação (registo) Restantes parâmetros (stack)
    - Invoca os serviços do sistema
    - Devolve os resultados

Organização



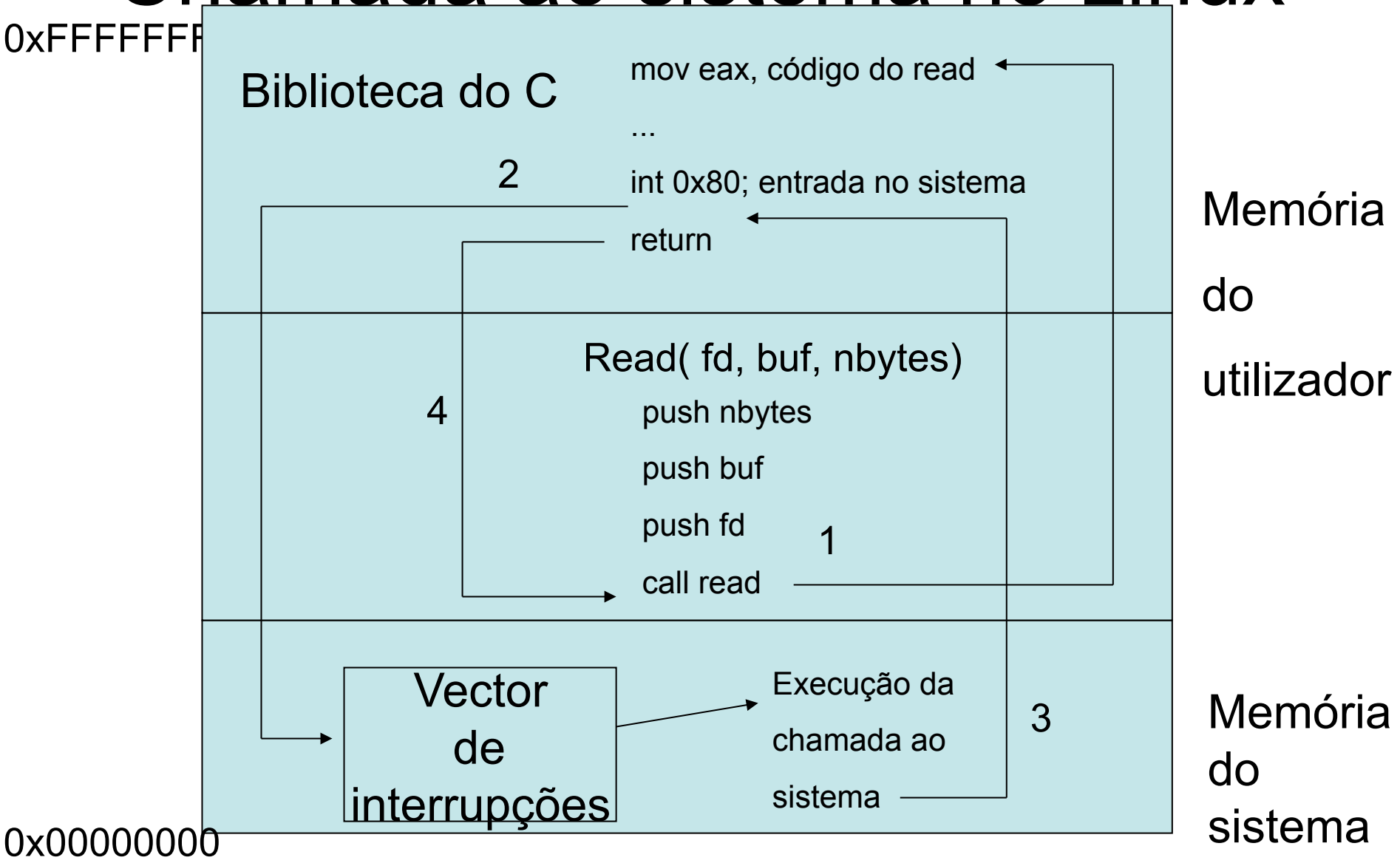
RAM disponível para imagens de processos.
Nesta zona o CPU corre em modo utilizador; quando o PC está nesta zona só está acessível a RAM reservada para o processo

RAM ocupada pelo SO;
Quando o PC está nesta zona o CPU está em modo supervisor
Quando o PC está nesta zona está acessível todo o código do sistema

Chamada ao sistema como subrotina

- Não há troca de contexto – a chamada ao sistema é executada pelo processo utilizador
- O kernel :
  - Muda para modo supervidor
  - Troca para um “stack” usado só nas chamadas
  - Verifica os parâmetros
  - Executa a chamada
  - Troca o “stack” para o habitual
  - Muda para modo utilizador
  - retorna

Chamada ao sistema no Linux



Chamada ao sistema read (fd, buf, nbytes)

Chamadas ao sistema nas versões clássicas do Unix (ex: Linux)

- UNIX – desde a versão original que não existe grande estruturação. O UNIX pode ser dividido em duas partes:
  - Programas de sistema
  - Núcleo (kernel)
    - Tudo o que está abaixo da “system-call interface” e acima do hardware
    - Suporta o sistema de ficheiros, escalonamento do CPU, e a gestão de memória

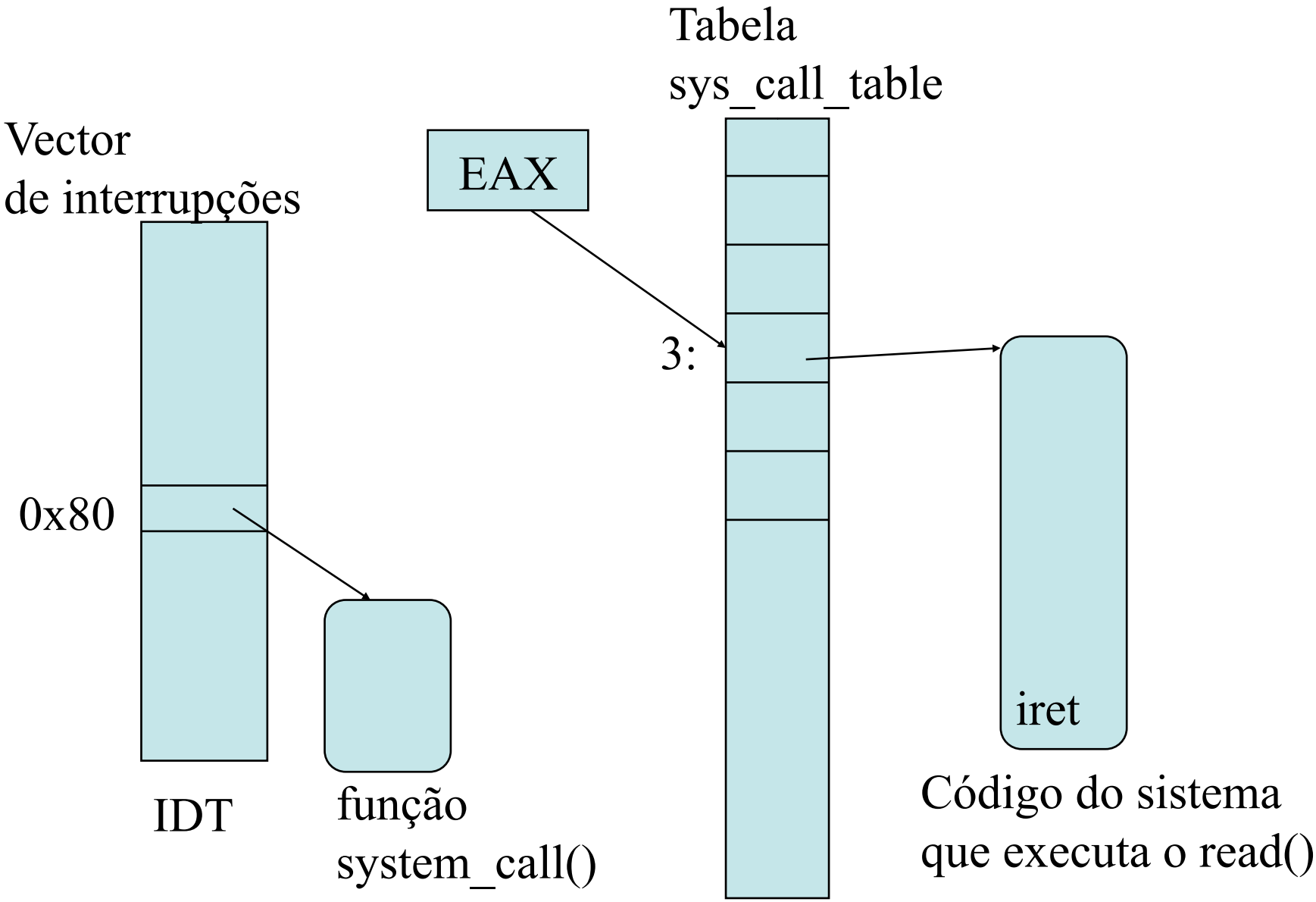
Chamadas ao sistema no Linux

- Biblioteca standard do C na arquitectura x86

```
        mov eax, código da chamada
        mov ebx, 1o. Parâmetro
        ...
        int 0x80
Cont:
```

- O registo IDTR aponta para a base do vector de interrupções em memória (256 entradas de 8 bytes)
- CPU em modo supervisor; PC ← end. Handler de chamadas ao sistema no kernel (*system_call()* : ficheiro **entry.S**)
- A tabela *sys_call_table* é indexada pelo registo EAX e contem os endereços de cada uma das rotinas que suportam as chamadas ao sistema
- A rotina executada termina com a instrução *iret* que volta a colocar o CPU em modo utilizador e a execução continua no ponto a seguir à invocação

Chamada ao sistema read (eax=3)



Excepções

- Vários tipos de excepções
  - Dependendo da forma como são reportadas
    - Durante a instrução
    - No final da instrução
  - A instrução que provocou a excepção pode ou não ser reiniciada

Excepções (cont.)

- Com reinício da instrução
  - Gerada antes da instrução terminar
  - Exemplos:
    - Erro na divisão (detectada em **`div/idiv`**)
    - Página não presente em memória
- Sem reinício da instrução
  - Gerado só quando a instrução termina

Entradas nos vector de interrupções dedicadas

- Exemplo Pentium (dedicated interrupts)
- São as entradas 0 a 4

Índice da interrupção	Objectivo
0	Divide error
1	Single-step
2	Nonmaskable interrupt (MNI)
3	Breakpoint
4	Overflow

Interrupções dedicadas (cont)

- Interrupção por erro na divisão
  - O CPU gera uma interrupção tipo 0 se o quociente resultado das instruções *div/idiv* é maior do que a capacidade do destino
- Single-Step Interrupt (Passo a Passo)
  - Útil em depuração (debugging)
  - Para o passo a passo actuar , a *flag* Trap (TF) deve estar a 1
  - O CPU gera uma interrupção tipo 1 após cada instrução se TF estiver a 1
  - A rotina de serviço (ISR) da interrupção tipo 1 pode ser usada para mostrar o estado do sistema ao utilizador

Interrupções dedicadas (cont)

- Breakpoint Interrupt
  - Útil em depuração (debugging)
  - O CPU gera uma interrupção tipo 3
  - Há uma instrução especial **`int 3`** que tem só um byte (opcode 0xCC)
- Overflow Interrupt
  - Há duas formas de gerar esta interrupção (tipo 4)
    - **`int 4`** :gera incondicionalmente uma interrupção tipo 4
    - **`into`** :interrupção gerada só se OF (overflow flag) está a 1