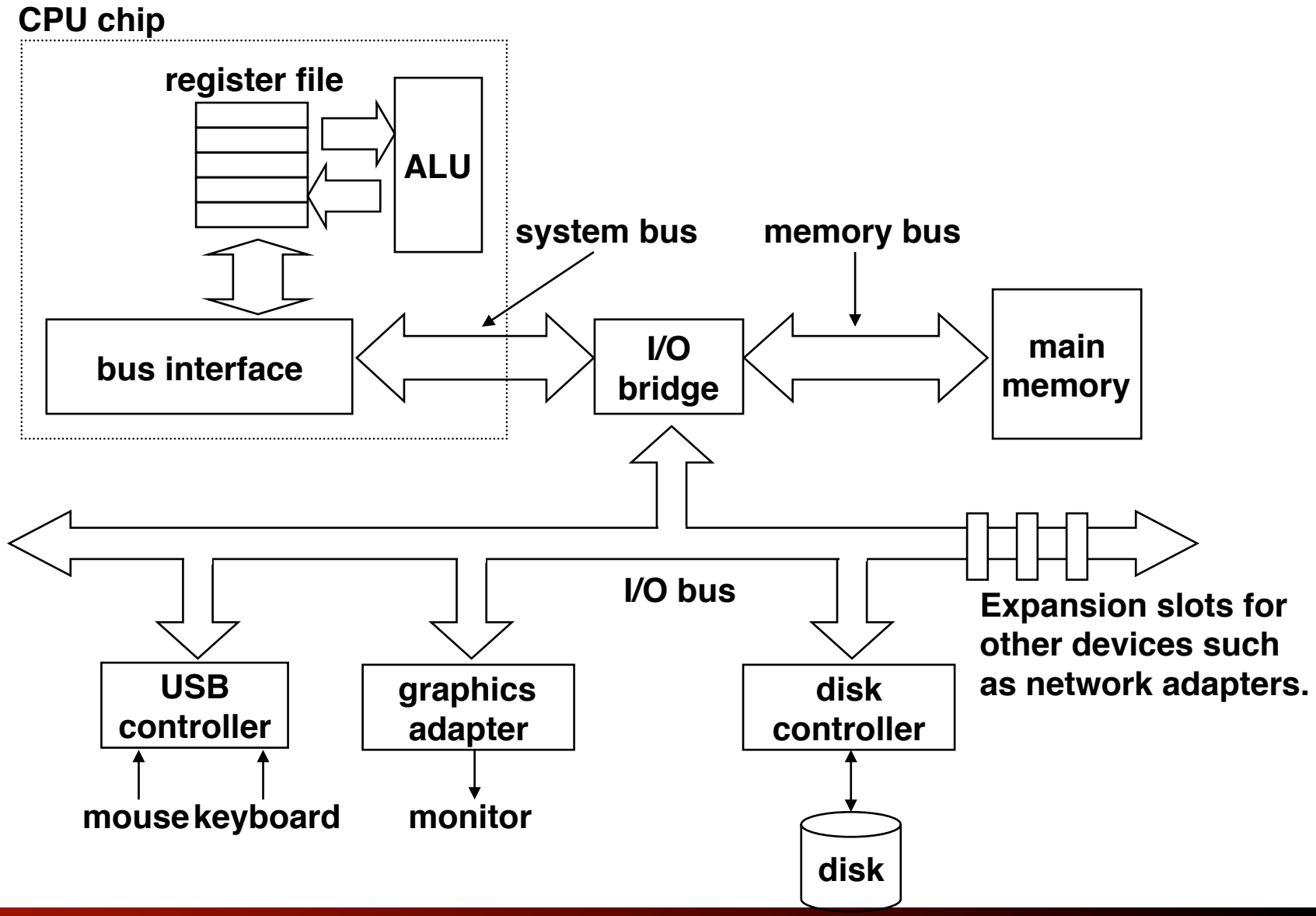
Sistema de Entradas-Saídas
Acesso aos dispositivos

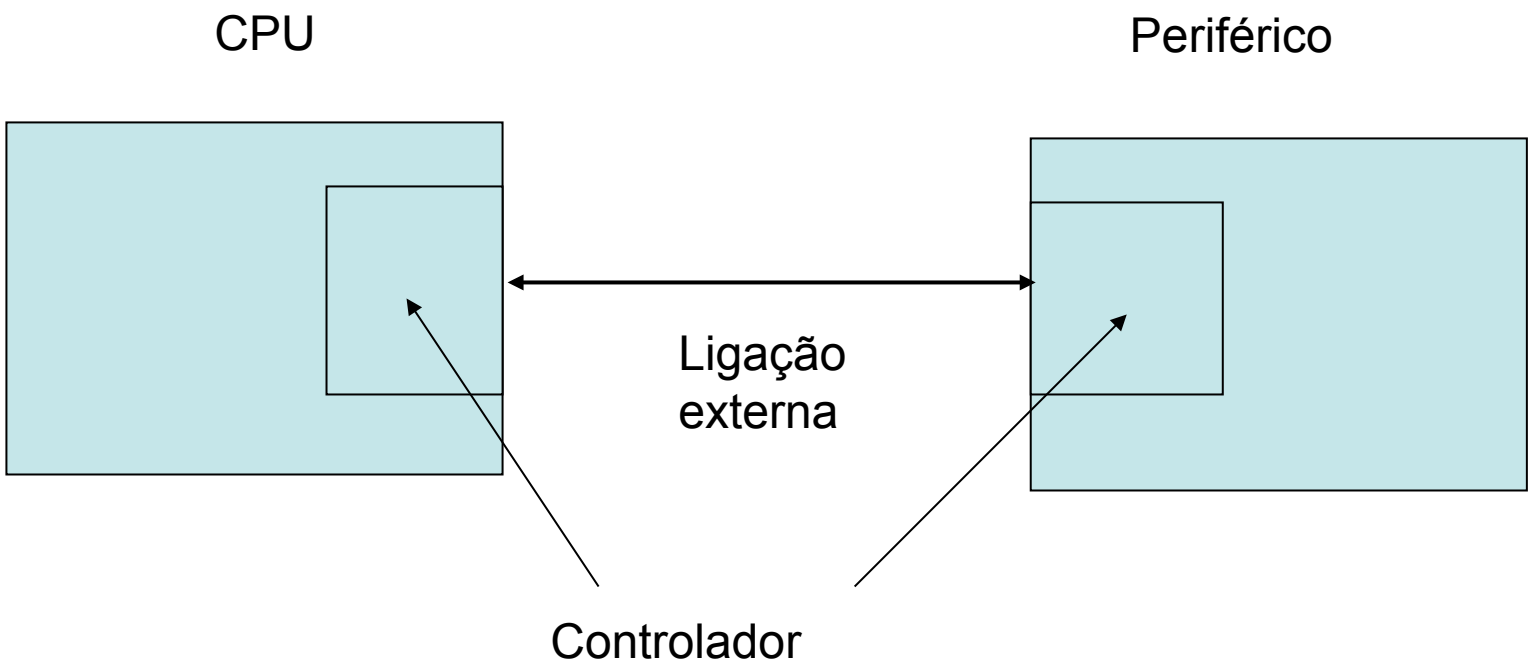
Secções 19.1 a 19.4

S.P. Dandamudi, *Fundamentals of Computer Organization and Design*, Ed. Springer, 2003

Hardware típico



Interligação CPU-Periférico



- Para transferir dados entre o CPU e o periférico
  - ✓ Quem inicia a transferência?
  - ✓ Que protocolos (nível eléctrico, ritmo, ...)

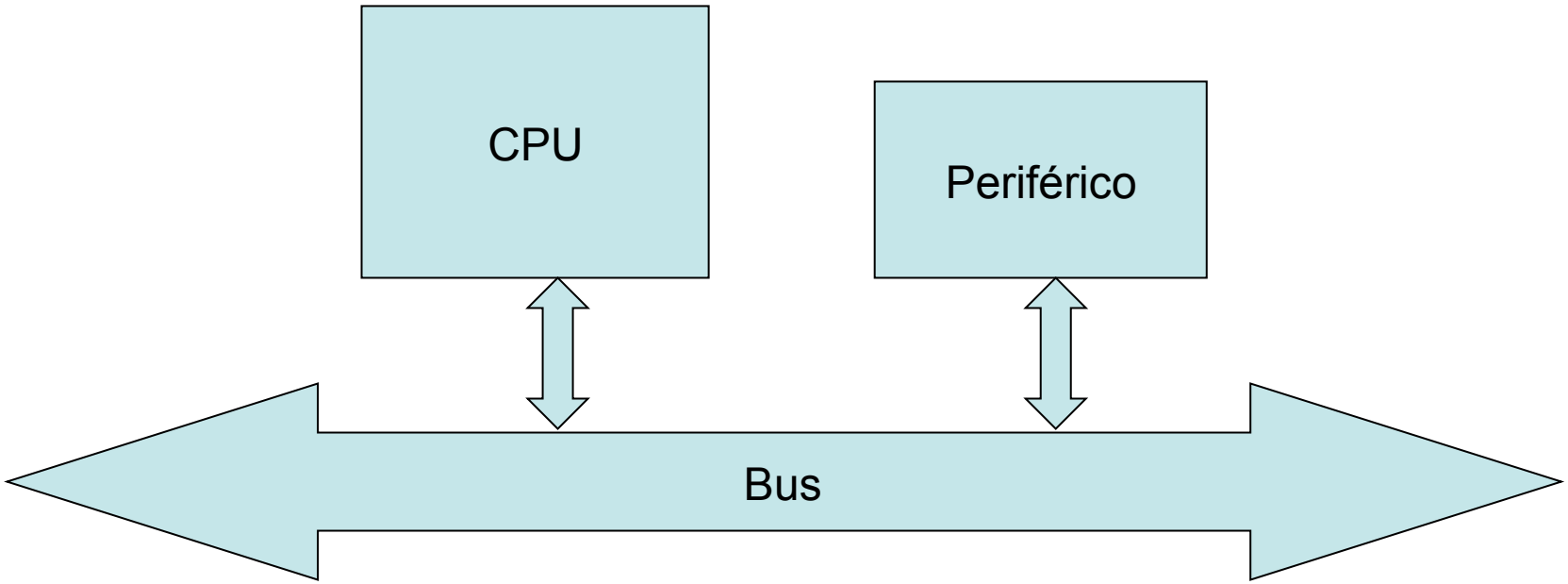
Visão do CPU em relação às entradas/saídas

- O CPU não faz acesso aos periféricos directamente
- Usa uma interface programável para passar pedidos ao controlador de interface e para receber as repostas a esses pedidos
- O controlador de interface converte esses pedidos em sinais eléctricos
- Esses sinais são interpretados pelo controlador do lado do periférico
- Este é que interage directamente com o dispositivo físico.

Bus (barramento)

- Um “bus” é um mecanismo de comunicação digital que permite a duas ou mais unidades funcionais transferir dados e sinais de controle
- Um computador pode conter vários “buses” que estão optimizados para um determinado objectivo
  - ✓ “bus” de memória: liga o CPU ao subsistema de memória
  - ✓ “bus” de entrada/saída (i/o bus) interliga o CPU a um conjunto de dispositivos de entrada/saída

Bus (barramento)



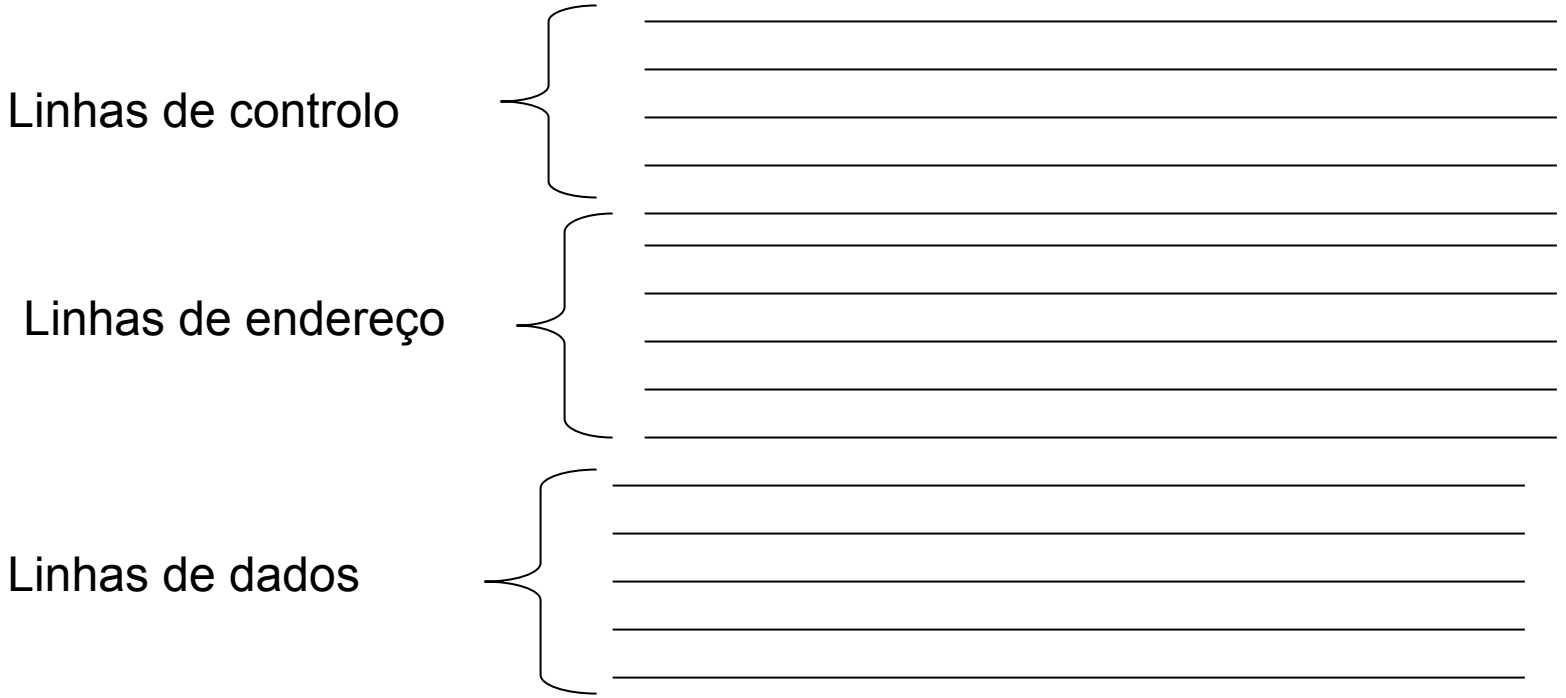
- Um “bus” suporta as ligações entre o CPU e os periféricos referidas anteriormente

Características do “bus”

- Transferência de dados em paralelo
  - ✓ Pode transferir múltiplos bits em simultâneo
  - ✓ Largura típica: 32 ou 64 bits
- O “bus” é normalmente passivo
  - ✓ Não contém muita lógica própria; pode ser visto como um mero conjunto de fios (linhas)
    - ✓ Fios numa placa (board) de circuito impresso
    - ✓ Fichas (sockets) no motherboard permitem inserir e remover dispositivos facilmente
  - ✓ São os dispositivos a ele ligados que suportam a comunicação

Organização do “bus”

- Bus está separado funcionalmente em 3 partes
  - ✓ Controlo (leitura, escrita, interrupções)
  - ✓ Especificação da localização do endereço
  - ✓ Dados a transferir



Acesso ao “bus”

- O “bus” só suporta duas operações
  - ✓ Leitura
  - ✓ Escrita
- O modo de acesso é o mesmo da memória
- Todas as operações de E/S são efectuadas efectuando leituras e escritas no “bus”

Acesso em leitura

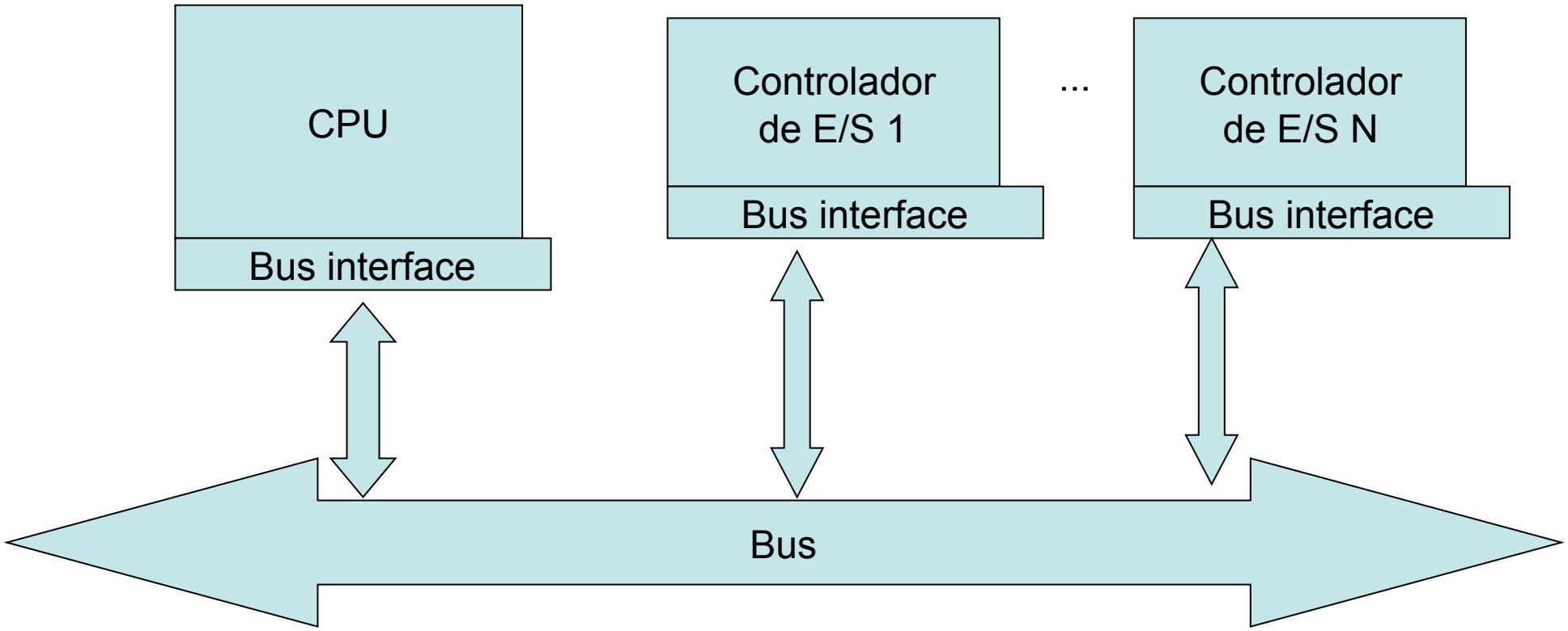
- Usa as linhas de controlo para obter acesso ao “bus”
- Coloca um endereço nas linhas de endereço
- Usa as linhas de controlo para requerer uma operação de leitura
- Testa as linhas de controlo para verificar se a transferência já se concluiu
- Lê um valor das linhas de dados
- Posiciona as linhas de controlo para deixar outra entidade fazer acesso ao “bus”

Acesso em escrita

- Usa as linhas de controlo para obter acesso ao “bus”
- Coloca um endereço nas linhas de endereço
- Coloca um valor nas linhas de dados
- Usa as linhas de controlo para indicar uma operação de escrita
- Testa as linhas de controlo para verificar se a transferência já se concluiu
- Posiciona as linhas de controlo para deixar outra entidade fazer acesso ao “bus”

Endereçamento no “bus”

- Um “bus” define um espaço de endereçamento



Endereçamento no “bus”

- A interface de bus implementa o protocolo do “bus” e trata de todas as transferências
- Usa as linhas de controlo para fazer acesso ao bus
- Posiciona as linhas de dados e endereços
- Embora receba todos os pedidos que passam no “bus”, *a interface só responde aos que contêm os endereços para os quais a interface foi configurada*

Ponto de vista do CPU

- A interface do bus fornece uma interface de programação, que tem apenas duas operações:
  - ✓ Ler e escrever
- Quando há uma referência à memória o CPU deixa tudo a cargo da interface do bus:
  - ✓ Leitura – o CPU pede à interface para efectuar a leitura
  - ✓ Escrita – idem
- Do ponto de vista do programador, a interface de bus é invisível: ela define um **espaço de endereçamento, em que cada controlador corresponde a uma faixa de endereços distinta**

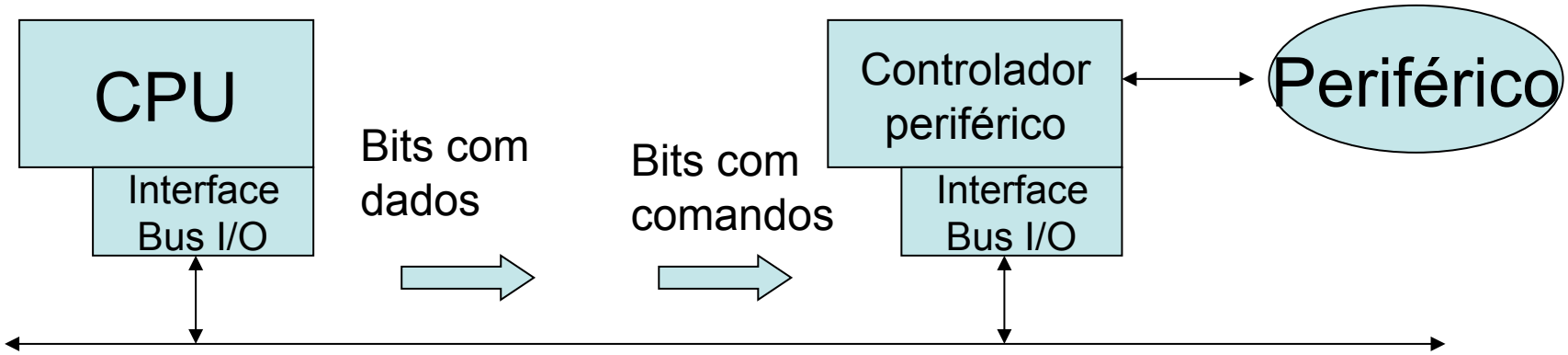
Ponto de vista do controlador do dispositivo

Seja R o conjunto de endereços associdо à interface

```
while(1){
   monitorar o bus até aparecer um pedido
   if( pedido refere um endereço pertencente a R )
       responder ao pedido
   else
       ignorar o pedido
}
```

Como é que operações de leitura e escrita manipulam periféricos ?

- O bus apenas fornece um meio de passar bits de uma entidade para outra
- Os bits que passam no bus podem ser:
  - ✓ Dados que estão a ser transferidos
  - ✓ Representar operações de controlo sobre o periférico; uma determinada configuração de bits é interpretada pela interface hardware



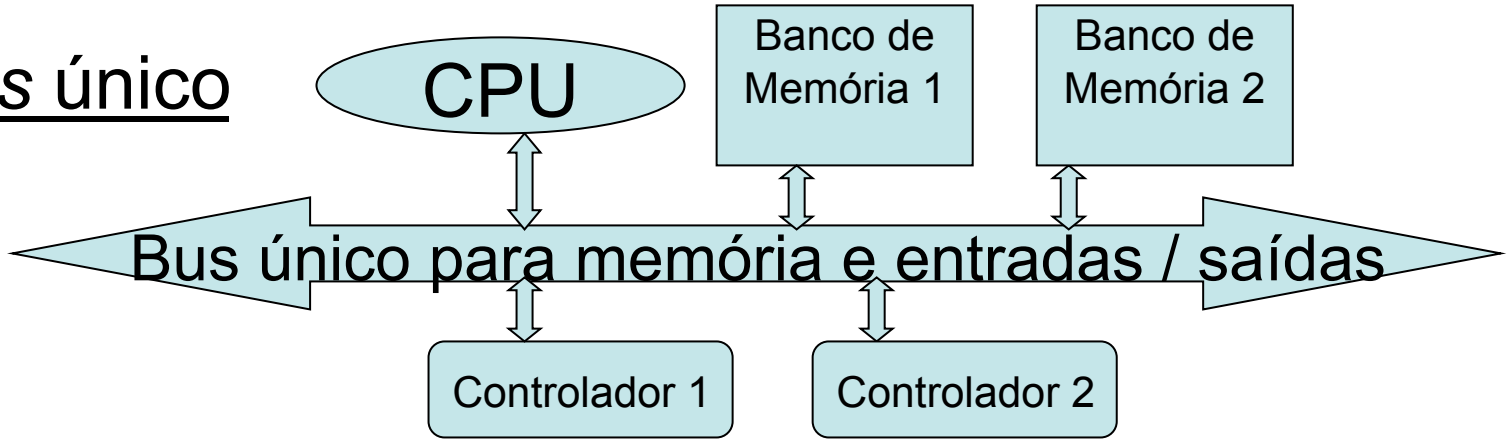
Exemplo: um “display” de luzes

- Periférico faz as seguintes acções:
  - ✓ Ligar/desligar o display
  - ✓ Mudar o brilho
  - ✓ Luz nº i on e off
- Controlador ocupa endereços de 100 a 103; bus tem 16 bits de dados

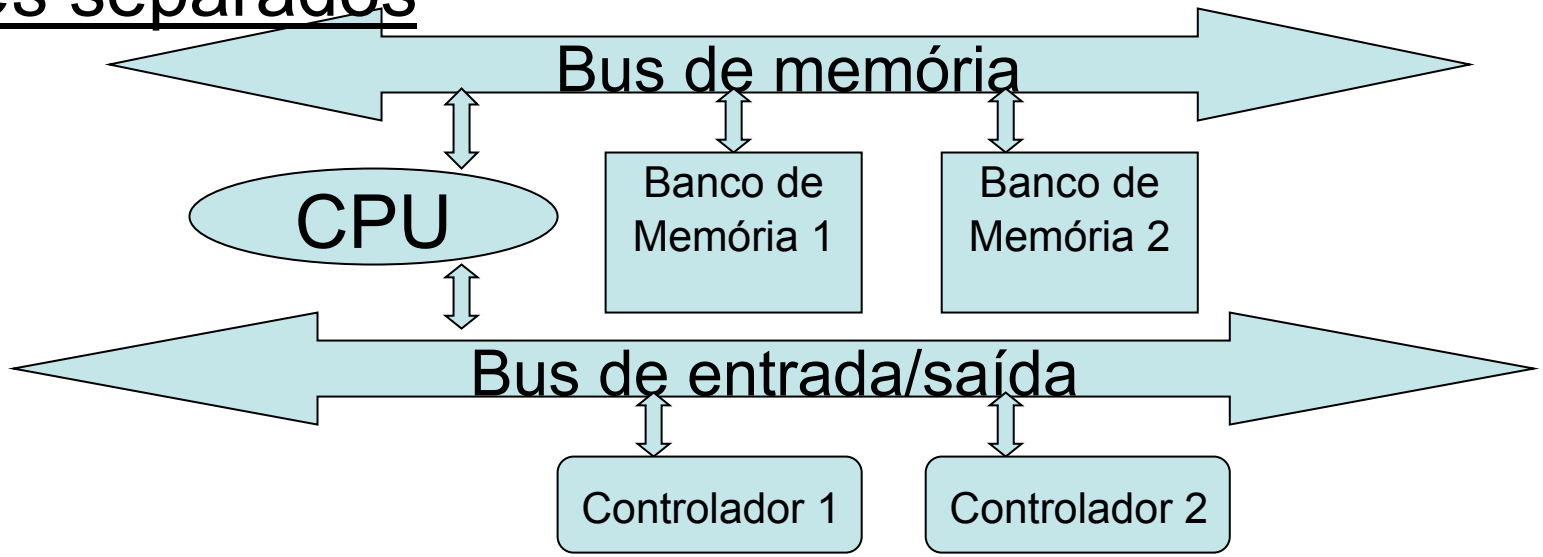
Endereço	Operação	Acções associadas
100	Escrita	Dados a zero desligam o display; dados não nulos ligam o display
101	Leitura	Retorna 0 se o display está desligado; retorna um valor não nulo se está ligado
102	Escrita	Muda o brilho. Os 4 bits menos segnificativos especificam o brilho de 0 (menor) a 15 (maior)
103	Escrita	“Bitmap” das luzes 0 a 15: bit a 1 luz acesa, bit a 0 luz apagada

Buses de memória e E/S unificados ou separados

Bus único



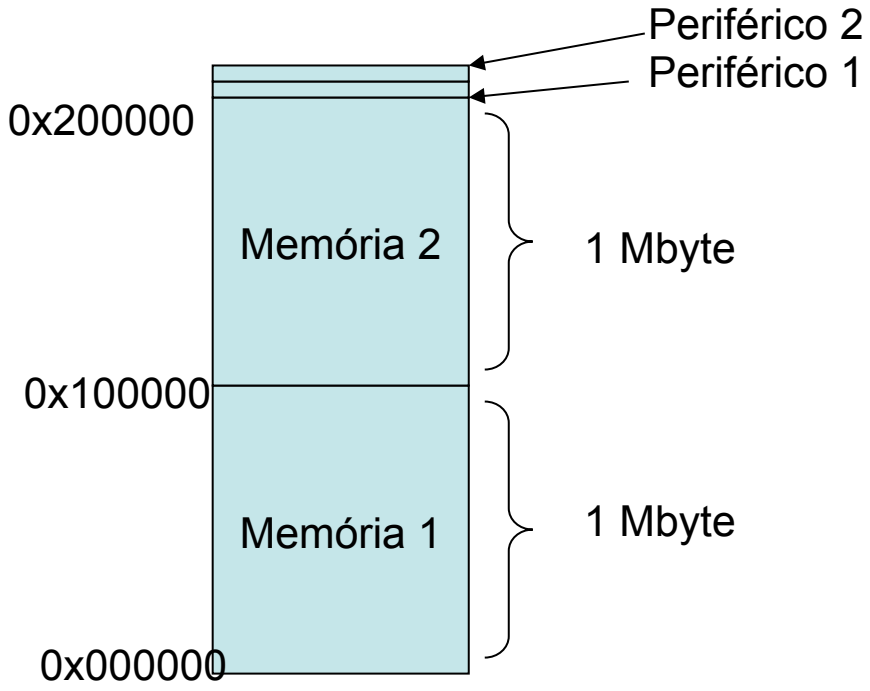
Buses separados



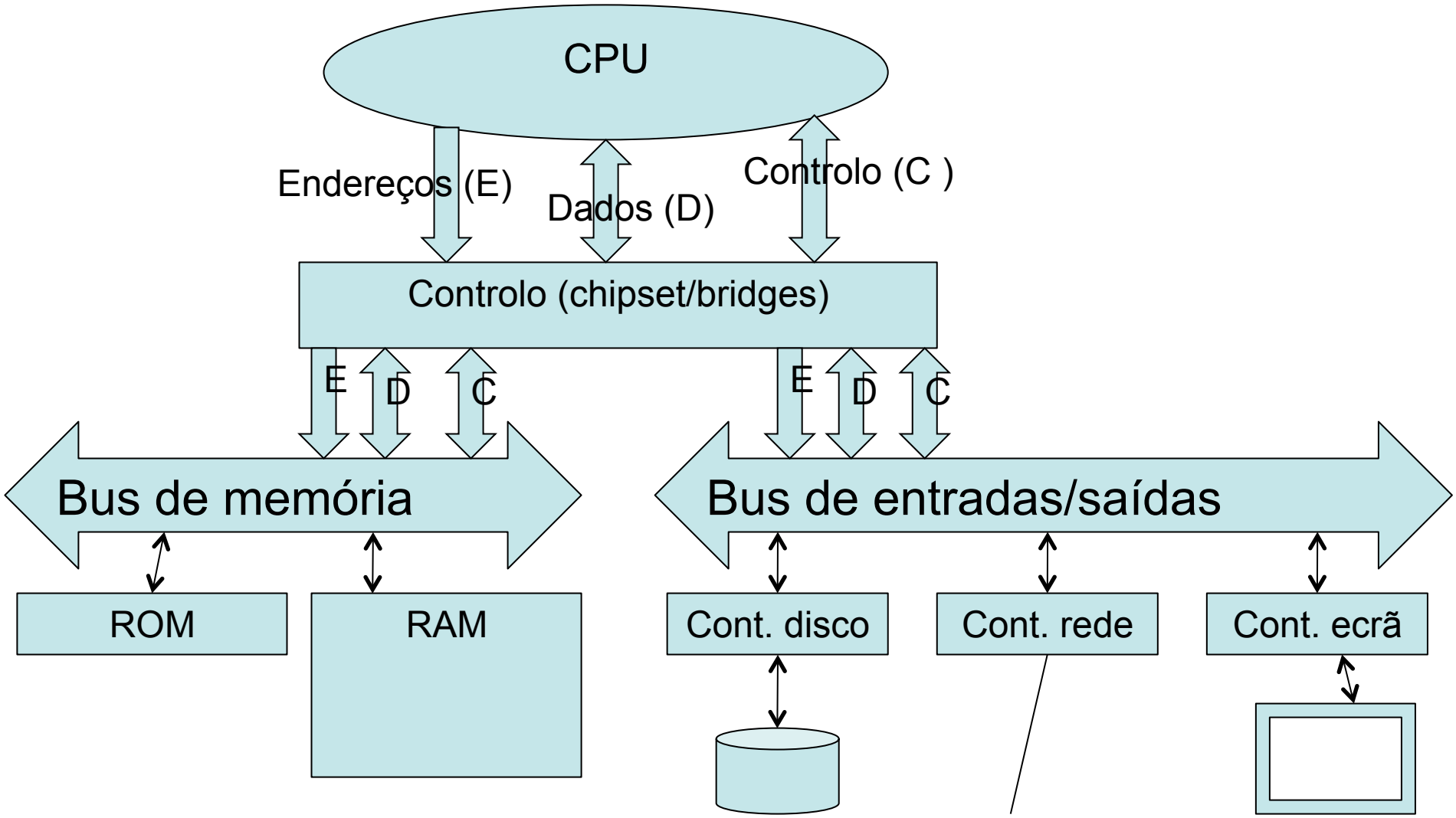
Bus único

- Espaço de endereços único
- Cada banco de memória é configurado para uma faixa de endereços que não tem intersecção com a faixa de endereços de nenhum outro banco
- Cada controlador é configurado para responder a uma faixa de endereços a que nenhum outro controlador responde
- As faixas de endereços dos bancos de memória e dos controladores também são disjuntos

Entidade	Faixa de endereços
Memória 1	0x000000-0x0FFFFF
Memória 2	0x100000-0x1FFFFF
Periférico 1	0x200000-0x20000B
Periférico 2	0x20000C-0x200017



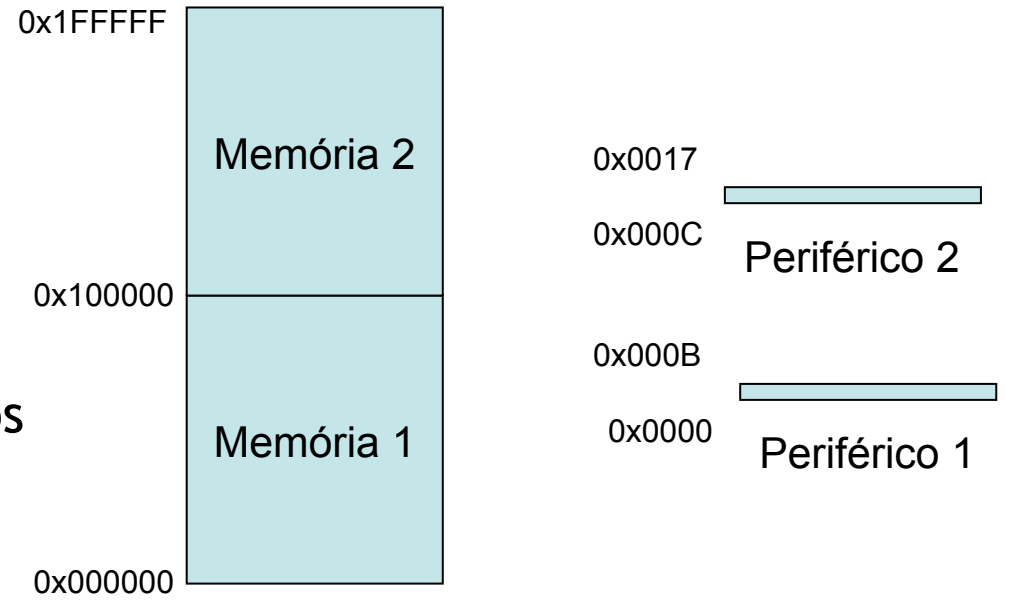
Espaços de endereçamento separados para RAM e E/S



“Buses” separados

- Dois espaços de endereçamento separados
  - ✓ Um para a memória
  - ✓ Um para os controladores dos dispositivos
- Cada banco de memória é configurado para uma faixa de endereços que não tem intersecção com a faixa de endereços de nenhum outro banco
- Cada controlador é configurado para responder a uma faixa de endereços a que nenhum outro controlador responde
- As faixas de endereços dos bancos de memória e dos controladores não precisam de ser disjuntos

Entidade	Faixa de endereços
Memória 1	0x000000-0x0FFFFF
Memória 2	0x100000-0x1FFFFF
Periférico 1	0x0000-0x000B
Periférico 2	0x000C-0x0017



Bus separado vs. bus único (1)

- Bus separado
  - ✓ Há instruções máquina dedicadas a fazer acesso ao bus de entrada/saída
  - ✓ Exemplo da família 80x86/Pentium
    - ✓ IN endereço (leitura para o registo EAX)
    - ✓ OUT endereço (escrita a partir do conteúdo do EAX)
- Exemplo – display: ligar a lâmpada 1

```
mov eax, 0x00000001
out 103
```

Bus separado vs. bus único (2)

- Bus único
  - ✓ Não há instruções máquina especiais para acesso aos controladores. São usadas LOAD e STORE para endereços que pertencem á faixa de endereços de entrada/saída
- Exemplo – display: ligar a lâmpada 1

```
mov ebx, 103
mov [ebx], 0x00000001
```

Programação dos controladores de dispositivos de E/S

- Controladores podem ser mais ou menos autónomos em relação ao CPU
  - ✓ Controladores com pouca inteligência precisam de grande interacção com o CPU
  - ✓ Controladores com grande inteligência só necessitam de atenção do CPU no início e fim da transferência
- Sincronização entre o CPU e os controladores
  - ✓ Periféricos são muito mais lentos do que o CPU
  - ✓ O CPU tem de interactuar com o periférico para saber se este já está disponível para receber novo comando, se já tem o dado pretendido ...

Programação por espera activa (polling)

- O CPU interroga repetidamente o controlador do periférico para saber se a operação anteriormente especificada já terminou
- Exemplo – interacção com uma impressora
  - ✓ Mandar avançar o papel
  - ✓ Esperar fazendo polling até que o papel avance
  - ✓ Mover a cabeça de impressão para o início da linha
  - ✓ Esperar fazendo polling até que a cabeça esteja posicionada
  - ✓ Especificar um carácter para imprimir
  - ✓ Esperar que o carácter escolhido esteja em frente do martelo
  - ✓ Provocar o impacto do martelo no carácter
  - ✓ Esperar que esta operação termine

Código para espera activa (polling)

- A espera activa é feita lendo um endereço do controlador; quando o CPU lê este endereço o controlador responde com um conjunto de bits que representam o seu *estado*
- Aplicação ao caso da impressora estúpida:
  - ✓ O controlador usa os endereços:
    - ✓ 0x40 (escrita) – a escrita de um valor dif de 0 avança o papel
    - ✓ 0x41 (escrita) – a escrita de um valor dif de 0 move a cabeça para o início da linha
    - ✓ 0x42 (escrita) – caracter ASCII a imprimir
    - ✓ 0x43 (escrita) - a escrita de um valor dif de 0 faz o martelo percutir
    - ✓ 0x40 (leitura) – estado; um valor dif de 0 indica que o comando anterior ainda não terminou

Interacção com a impressora

```
        out [40], 0x01          ; iniciar avanço do papel
L1:     in  al, [40]  ; lê o estado da interface
        test al,0xFF
        jnz  L1                 ; espera que o estado seja 0
;
        out [41], 0x01          ; iniciar retorno da cabeça
L2:     in  al, [40]  ; lê o estado da interface
        test al,0xFF
        jnz  L2                 ; espera que o estado seja 0
;
        out [42], ´C´           ; escolher o caracter ´C´
L3:     in  al, [40]  ; lê o estado da interface
        test al,0xFF
        jnz  L3                 ; espera que o estado seja 0
;
        out [43], 0x01          ; mandar o martelo percutir
L4:     in  al, [40]  ; lê o estado da interface
        test al,0xFF
        jnz  L4                 ; espera que o estado seja 0
```

Registos de uma interface (1)

- Uma interface ocupa uma série de endereços (normalmente consecutivos) no espaço de endereçamento
  - ✓ Registo(s) de dados (leitura): onde se lêm os dados que vêm do exterior
  - ✓ Registo(s) de dados (escrita): onde se escrevem os dados a enviar para o exterior
  - ✓ Registo de comando (escrita): onde se dão comandos à interface
  - ✓ Registo de estado (leitura): conjunto de bits que dão informação sobre o estado do controlador: livre/ocupado, transferência com/sem erro , ...

Registos de uma interface (2)

- Muitas vezes, o registo de comando e de estado ocupam o mesmo endereço
  - ✓ Uma escrita no endereço controla o periférico
  - ✓ Uma leitura do mesmo endereço retorna o estado do controlador
- Muitas vezes, uma operação de leitura, implica implicitamente uma ordem para ler mais um valor. Exemplo:
  - ✓ Quando um rato se move, há um registo de dados em que fica guardado o movimento relativo em relação à última posição
  - ✓ Quando o CPU lê esse registo de dados, desencadeia automaticamente um novo processo de medição de deslocamento, cujo resultado virá para o registo de dados.

Inconvenientes do uso do “polling”

- Os controladores que só suportam espera activa são muito simples do ponto de vista hardware, mas implicam um grande desperdício de tempo de CPU
- Os dispositivos de E/S são muito lentos e o CPU vai passar grande parte do tempo nos ciclos do exemplo anterior
- O tempo de espera é fixo (depende do periférico) e independente da velocidade do CPU
- Enquanto se espera há potencial para o CPU executar muitas instruções

O mecanismo de interrupções aumenta a taxa de uso do CPU

- A invenção do mecanismo de interrupções (~1960) permitiu resolver a questão da desadequação das velocidades do CPU e dos periféricos
- Permite que o CPU continue a efectuar computações enquanto espera que a transferência de dados acabe
- O controlador actua autonomamente depois do CPU iniciar a operação de transferência; quando esta termina o controlador envia uma interrupção ao CPU.