

Organização Básica de computadores e linguagem de montagem

Prof. Edson Borin

2º Semestre de 2015

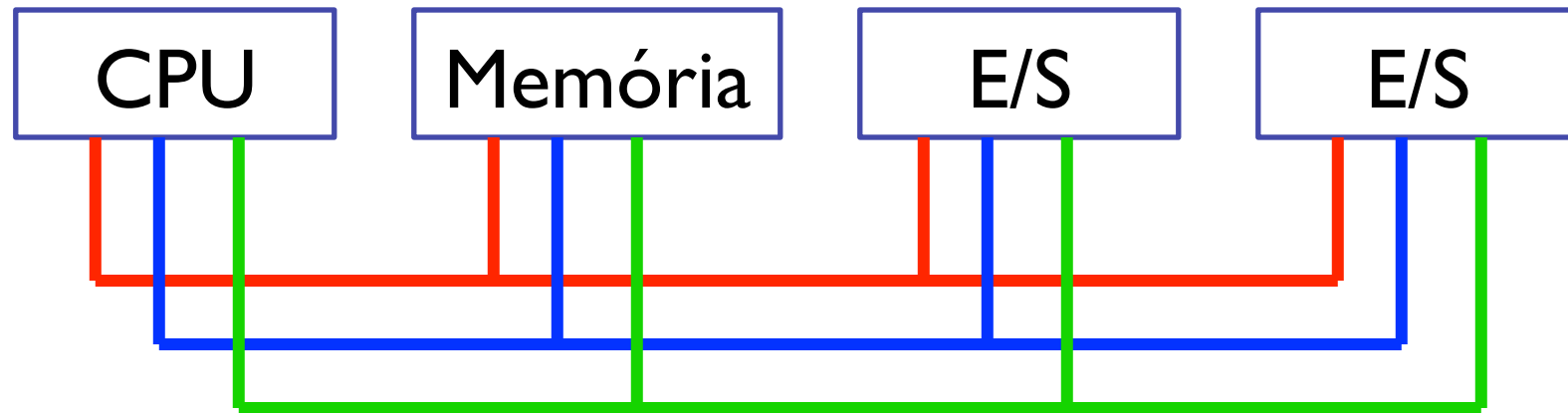


Barramentos

Barramentos

- Caminhos de comunicação entre dois ou mais dispositivos
- Diversas linhas de comunicação que podem ser classificadas em:
 - Linhas (ou barramento) de controle
 - Linhas (ou barramento) de endereço
 - Linhas (ou barramento) de dados
- Exemplos de barramento
 - PCI: desenvolvido originalmente pela Intel. Atualmente é um padrão público
 - AMBA: desenvolvido pela ARM

Barramentos

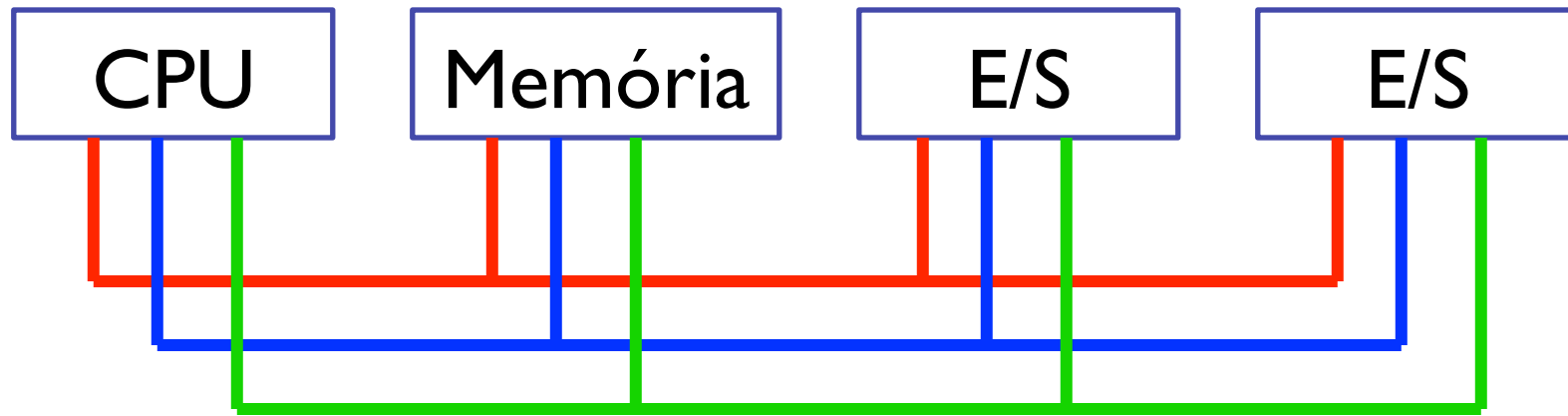


Linhas de dados

Linhas de endereço

Linhas de controle

Barramentos



Linhas de dados

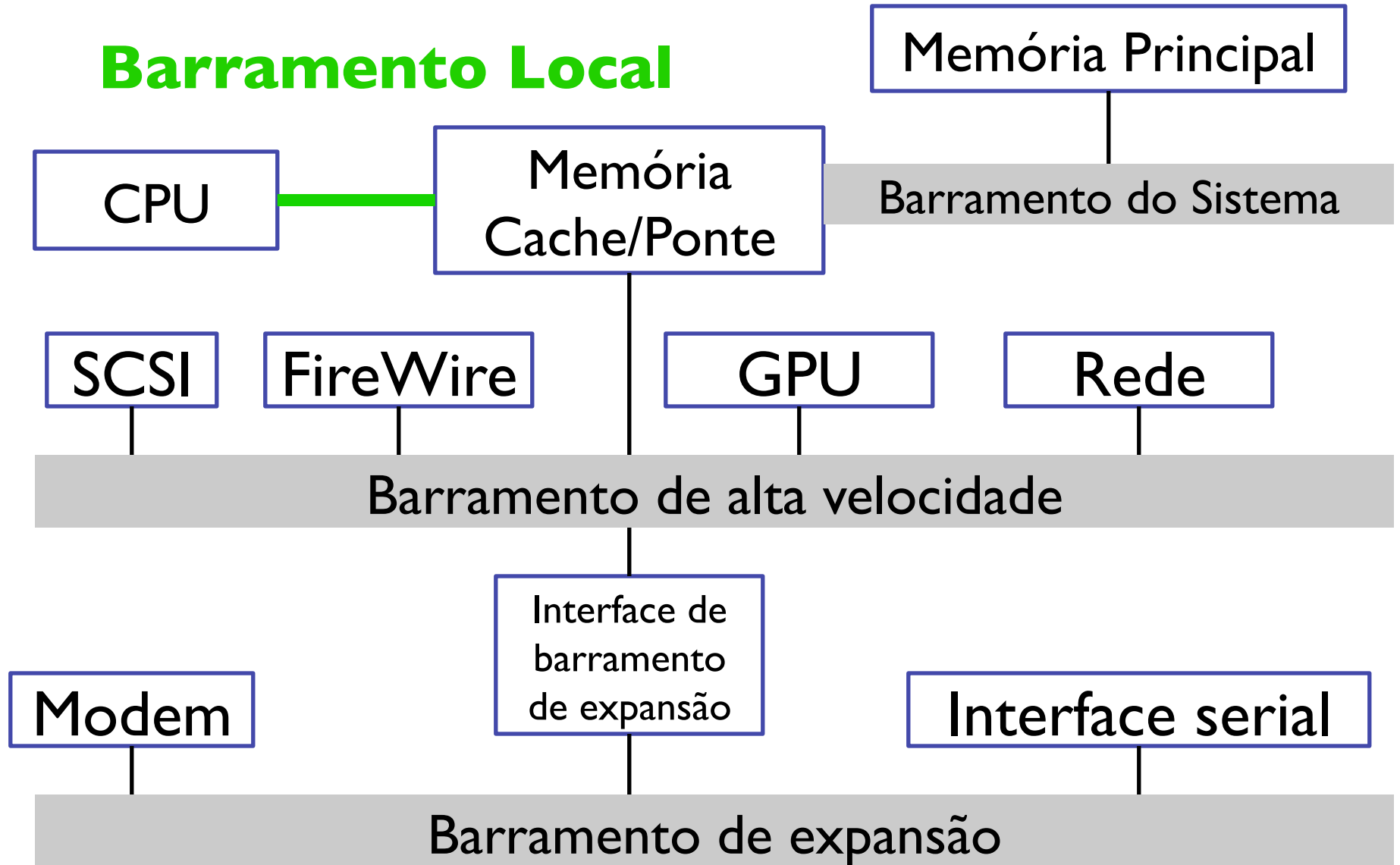
Linhas de endereço

Linhas de controle

- Todos os dispositivos compartilham o mesmo barramento
- Problema: todos têm que operar na mesma velocidade

Barramentos Modernos

Barramento Local



Barramentos

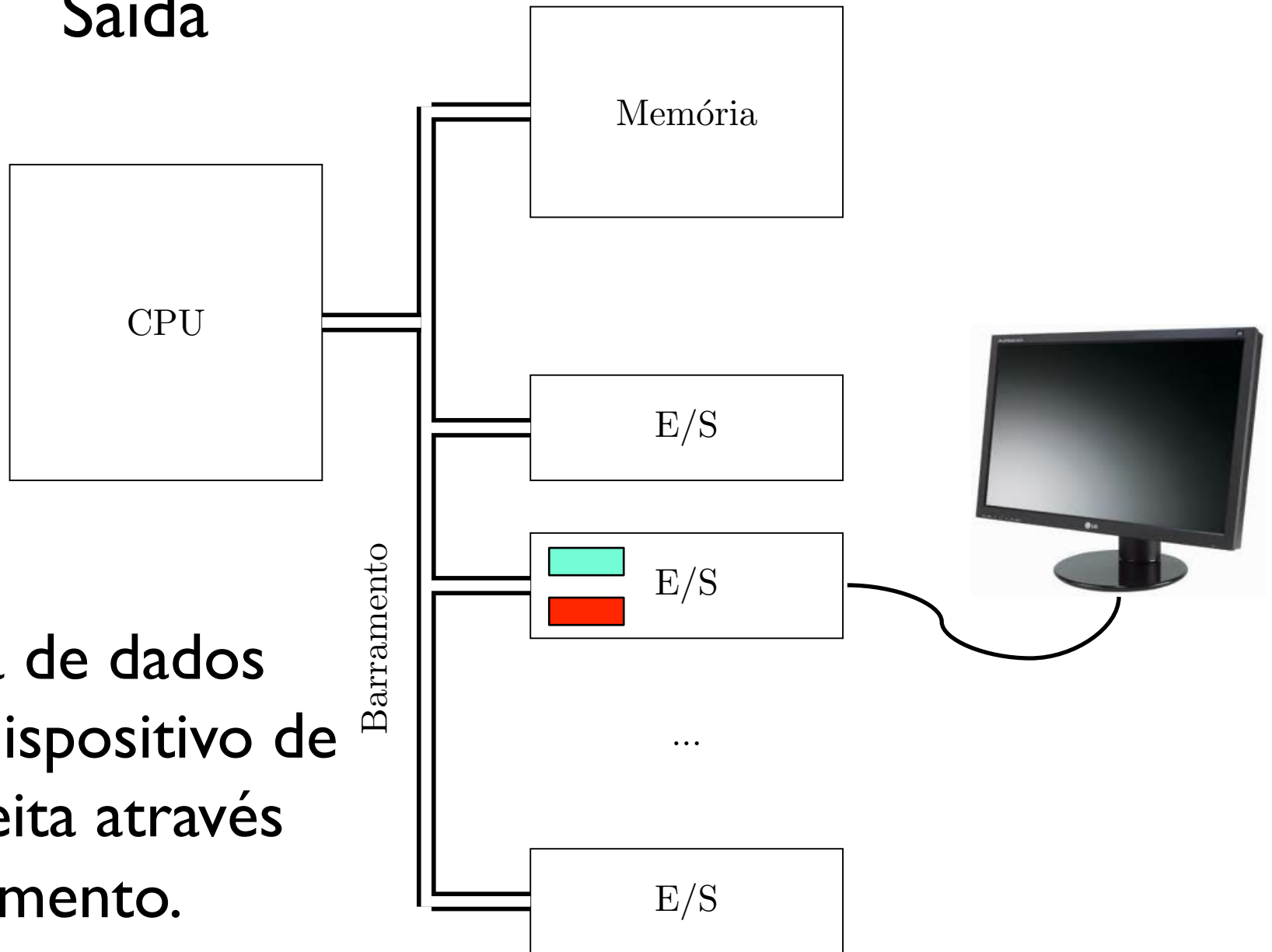
- Leitura recomendada: Capítulo 3.4 do livro do Stallings.

Entrada e Saída

Entrada e Saída

- Dispositivos de E/S (Entrada/Saída) ou I/O (Input/Output) permitem a entrada e saída de dados do processador.
- Ex: Teclado, Mouse, Monitor, Impressora, Placa de rede, disco rígido, unidade de CD-ROM.
- Como funciona?

Saída



A escrita de dados em um dispositivo de saída é feita através do barramento.

Saída

- Como o programa realiza uma saída?

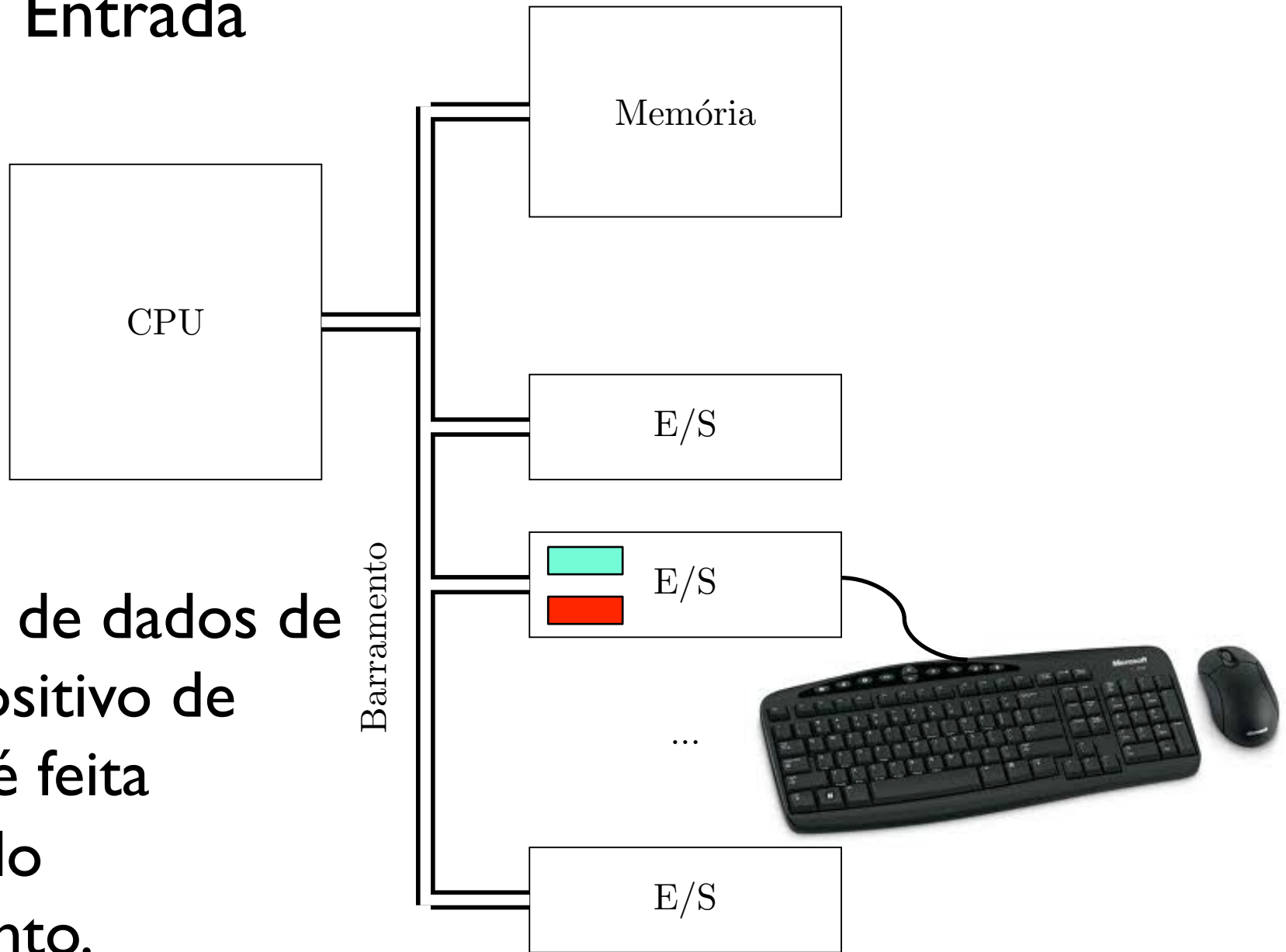
Saída

- Como o programa realiza uma saída?
- Escreve em uma *porta*, que está associada a um dispositivo de saída.

Saída

- Como o programa realiza uma saída?
- Escreve em uma *porta*, que está associada a um dispositivo de saída.
- 2 opções comuns:
 - Instrução especial para saída. Ex:
`out 0x10, r1`
 - Instrução de *store* em uma faixa de endereços reservada.
Ex:
`ldr r0, =0x80000`
`str r1, [r0]`
- Como o processador sabe se é uma saída ou acesso à memória?

Entrada



A leitura de dados de um dispositivo de entrada é feita através do barramento.

Entrada

- Como o programa realiza uma entrada?
- Lê de uma *porta*, que está associada a um dispositivo de entrada.
- 2 opções comuns:
 - Instrução especial para entrada. Ex:
`in r1, 0x10`
 - Instrução de *load* em uma faixa de endereços reservada.
Ex:
`ldr r0, =0x80000`
`ldr r1, [r0]`
- Como o processador sabe se é uma entrada ou acesso à memória?

Exemplos no ARM

- Entrada:

```
ldr    r0, =0x53FA0008
```

```
ldr    r1, [r0]
```

- Saída

```
ldr    r0, =0x53FA0000
```

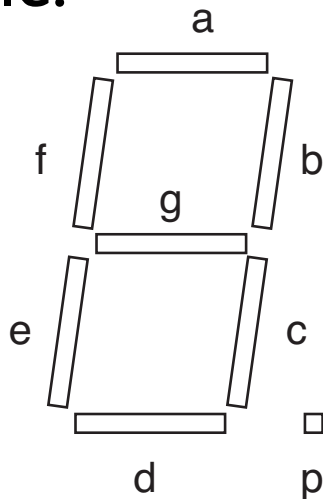
```
str    r1, [r0]
```

Exemplo - Elevador

- Dois dispositivos. 1 de entrada e 1 de saída
- Entrada:
 - sensor de andar. Conectado à porta 0x20.
 - quando acessado, responde com um *byte* indicando o andar atual (de 0 a 9)

Exemplo - Elevador

- Saída
 - mostrador digital: Conectado à porta 0x30
 - dispositivo com 7 segmentos (a,b,...,g) e um ponto luminoso que ficam ligados ou desligados de acordo com o dado no registrador de controle.
 - a saída corresponde em escrever um *byte* no registrador de controle.



7	6	5	4	3	2	1	0
p	a	b	c	d	e	f	g

Exemplo - Elevador

@ Procedimento atualiza andar

@ Lê o andar do sensor e atualiza o valor do mostrador

```
.equ  SENSOR_PORT, 0x20
```

```
.equ  DISPLAY_PORT, 0x30
```

atualiza_andar:

```
    ldr    r1, =SENSOR_PORT
```

```
    ldrb   r1, [r1]           @ lê o valor do sensor
```

```
    ldr    r0, =tab_digitos   @ converte valor para
```

```
    ldrb   r0, [r0, r1]       @ byte de controle
```

```
    ldr    r1, =DISPLAY_PORT @ escreve byte de controle
```

```
    strb   r0, [r1]           @ no mostrador
```

```
    mov    pc, lr
```

```
tab_digitos: .byte 7e,30,6d,79,33,5b,5f,70,7f,7b
```

Problemas com a abordagem anterior

- Suponha que o elevador suba 8 andares.
- Quando devemos chamar o procedimento `AtualizaAndar`?

Outro Exemplo - Teclado

- Teclado:
 - Dispositivo de Entrada
 - Duas portas: dados (0x40) e estado (0x41)
 - Dado lido representa caractere
- E se o teclado for apertado múltiplas vezes?
- Como saber se o dado que está lá já foi lido?

1	2	3
4	5	6
7	8	9
*	0	#

Outro Exemplo - Teclado

- Teclado:
 - um *bit* de estado indica se o dado atual não foi lido pelo processador ainda (READY). *Bit 0*
 - outro *bit* de estado indica se mais de um botão já foi apertado antes do processador ler o dado, ou seja, houve dado perdido (OVRN). *Bit 1*

1	2	3
4	5	6
7	8	9
*	0	#

Outro Exemplo - Teclado (*Busy waiting*)

- Rotina le_tecla
- Lê palavra de controle (end. 0x40)
- Se o dispositivo não tiver dado (*bit 0 - READY*)
 - Tenta novamente (*Busy waiting*)
- Se o dispositivo tiver dado
 - Verifica se houve perda de dado (*bit 1 - OVRN*)
 - Se houve perda de dado
 - Trata o erro
 - Senão
 - Lê dado do dispositivo (end. 0x41) e retorna

Outro Exemplo - Teclado (*Busy waiting*)

@ Procedimento lê tecla

@ Lê a o valor da tecla que foi pressionada

```
.equ    KB_DATA,    0x40
```

```
.equ    KB_STAT,    0x41
```

```
.equ    KB_READY,   0x01
```

```
.equ    KB_OVRN,    0x02
```

le_tecla:

```
    ldr    r1, =KB_STAT
```

```
    ldrb   r1, [r1]           @ lê o estado do teclado
```

```
    tst    r1, #KB_READY     @ testa se tem dado pronto
```

```
    beq    le_tecla          @ se não tiver, tenta novamente
```

```
    tst    r1, #KB_OVRN      @ perdeu dado?
```

```
    bne    lt_ovrn           @ se sim, trata erro
```

```
    ldr    r1, =KB_DATA      @ senão, lê dado
```

```
    ldrb   r0, [r1]          @ no mostrador
```

```
    mov    pc, lr
```

lt_ovrn: @ trata erro aqui

Problemas com a abordagem anterior

- Suponha que o usuário demore para apertar algo.
- O que o processador faz?

Problemas com a abordagem anterior

- Suponha que o usuário demore para apertar algo.
- O que o processador faz?
- Como melhorar?
 - Verifique o teclado de tempos em tempos e faça algum trabalho útil no intervalo entre as verificações.

Problemas com a abordagem anterior

- Suponha que o usuário demore para apertar algo.
- O que o processador faz?
- Como melhorar?
 - Verifique o teclado de tempos em tempos e faça algum trabalho útil no intervalo entre as verificações.
 - Ainda há o risco do usuário pressionar a tecla múltiplas vezes antes do programa verificar se alguma tecla foi pressionada.
 - Talvez o usuário não seja tão rápido para causar este problema, mas e se for uma placa de rede.

Problemas com a abordagem anterior

- Suponha que o usuário demore para apertar algo.
- O que o processador faz?
- Como melhorar?
 - Verifique o teclado periodicamente para algum trabalho útil no teclado.
 - Ainda há o risco de o usuário pressionar múltiplas vezes a mesma tecla antes de alguma tecla foi pressionada.
- Talvez o usuário não seja tão rápido para causar este problema, mas e se for uma placa de rede.

Interrupção: O dispositivo avisa o processador quando acontecer alguma coisa!

Interrupção

- Iniciativa de comunicação é do periférico
- Exemplo:
 - Quando o dado está disponível, o teclado “interrompe” o processador.
 - O processador pára o que está fazendo para atender o teclado
 - Após o processamento da leitura, o processador continua com o que estava fazendo.

Interrupção

- **O processador pára o que está fazendo para atender o teclado.**
- O que acontece com o programa que o processador estava executando???

Exemplo: Interrupção

@ programa faça algo útil 1000 vezes

main:

 mov r4, #1000

loop:

 bl algo_util

 sub r4, r4, #1

 cmp r4, #0

 bne loop

 ...

Exemplo: Interrupção

@ programa faça algo útil 1000 vezes

main:

mov r4, #1000

loop:

bl algo_util

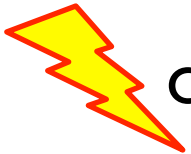
sub r4, r4, #1

cmp r4, #0

bne loop

...

Interrupção



Interrupção

- **O processador pára o que está fazendo para atender o teclado.**
- O que acontece com o programa que o processador estava executando???
- Antes de tratar a interrupção, é importante salvar todo o “contexto” do programa que está executando
 - Registradores,
 - *flags*,
 - Manter a pilha consistente...

Exemplo: Interrupção

@ programa faça algo útil 1000 vezes

main:

 mov r4, #1000

loop:

 bl algo_util

 sub r4, r4, #1

 cmp r4, #0

 bne loop

 ...

trata_interrupcao:

 @ salva contexto

 @ trata a interrupção

 @ restaura o contexto

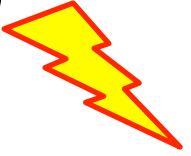
Exemplo: Interrupção

@ programa faça algo útil 1000 vezes

main:

mov r4, #1000

loop:

①  bl algo_util
sub r4, r4, #1
cmp r4, #0
bne loop
...

① Interrupção acontece

trata_interrupcao:

@ salva contexto

@ trata a interrupção

@ restaura o contexto



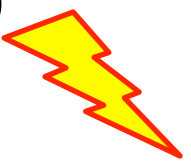

Exemplo: Interrupção

@ programa faça algo útil 1000 vezes

main:

mov r4, #1000

loop:

①  bl algo_util
sub r4, r4, #1
cmp r4, #0
②  bne loop
...

trata_interrupcao:

@ salva contexto

@ trata a interrupção

@ restaura o contexto

① Interrupção acontece

② Fluxo de controle é desviado

Exemplo: Interrupção

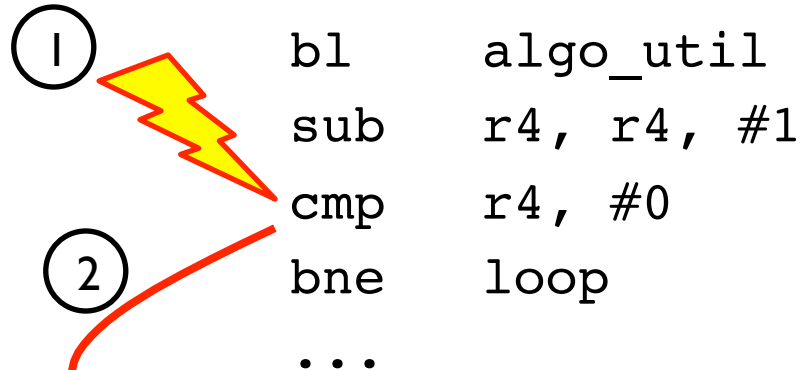
@ programa faça algo útil 1000 vezes

main:

mov r4, #1000

loop:

① bl algo_util
sub r4, r4, #1
cmp r4, #0
② bne loop
...



trata_interrupcao:

③ @ salva contexto
@ trata a interrupção
@ restaura o contexto

① Interrupção acontece

② Fluxo de controle é desviado

③ A interrupção é tratada

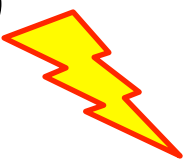
Exemplo: Interrupção

@ programa faça algo útil 1000 vezes

main:

mov r4, #1000

loop:

①  bl algo_util
sub r4, r4, #1
cmp r4, #0
② bne loop
...

trata_interrupcao:

③ @ salva contexto
@ trata a interrupção
④ @ restaura o contexto

① Interrupção acontece

② Fluxo de controle é desviado

③ A interrupção é tratada

④ Contexto é recuperado