MICROPROCESSADORES II
(EMA864315)

COMUNICAÇÃO SERIAL (UART)

1º SEMESTRE / 2019

Alexandro Baldassin

AULA PASSADA ...

- **Começamos a ver formas de comunicação entre o processador e dispositivos de I/O**
  - Vimos interfaces paralelas, onde um conjunto de bits é enviado/ recebido em paralelo
- **Hoje veremos comunicação serial. Prós e contras?**
  - Serial
  - Paralela

AULA PASSADA ...

- **Começamos a ver formas de comunicação entre o processador e dispositivos de I/O**
  - Vimos interfaces paralelas, onde um conjunto de bits é enviado/ recebido em paralelo

- **Hoje veremos comunicação serial. Prós e contras?**
  - Serial
    - Mais econômica para longas distâncias (relativamente poucos fios)
  - Paralela
    - Custo e complexidade de conectar múltiplos fios são maiores, além de ser mais suscetível a ruídos

AULA PASSADA ...

- **Começamos a ver formas de comunicação entre o processador e dispositivos de I/O**
  - Vimos interfaces paralelas, onde um conjunto de bits é enviado/ recebido em paralelo

- **Hoje veremos comunicação serial. Prós e contras?**
  - Serial
    - Mais econômica para longas distâncias (relativamente poucos fios)
    - Largura de banda potencialmente maior
  - Paralela
    - Custo e complexidade de conectar múltiplos fios são maiores, além de ser mais suscetível a ruídos
    - Largura de banda comparativamente menor (capacitância e indutância mútua)

LARGURA DE BANDA DE PICO

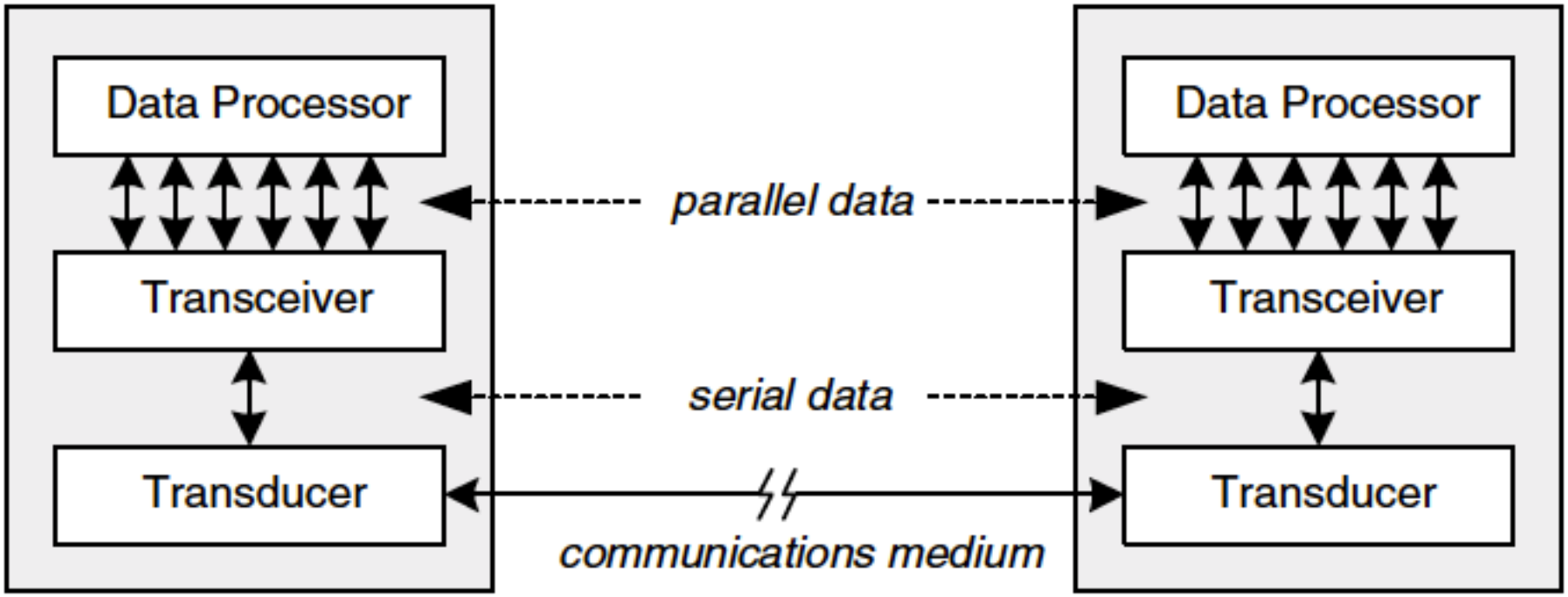
Largura de banda (pico) comparativa de alguns barramentos (em MB/s)

Firewire	USB 2.0	USB 3.0	PCI Express	SATA 2.0	PATA
100	60	625	250	300	133.5

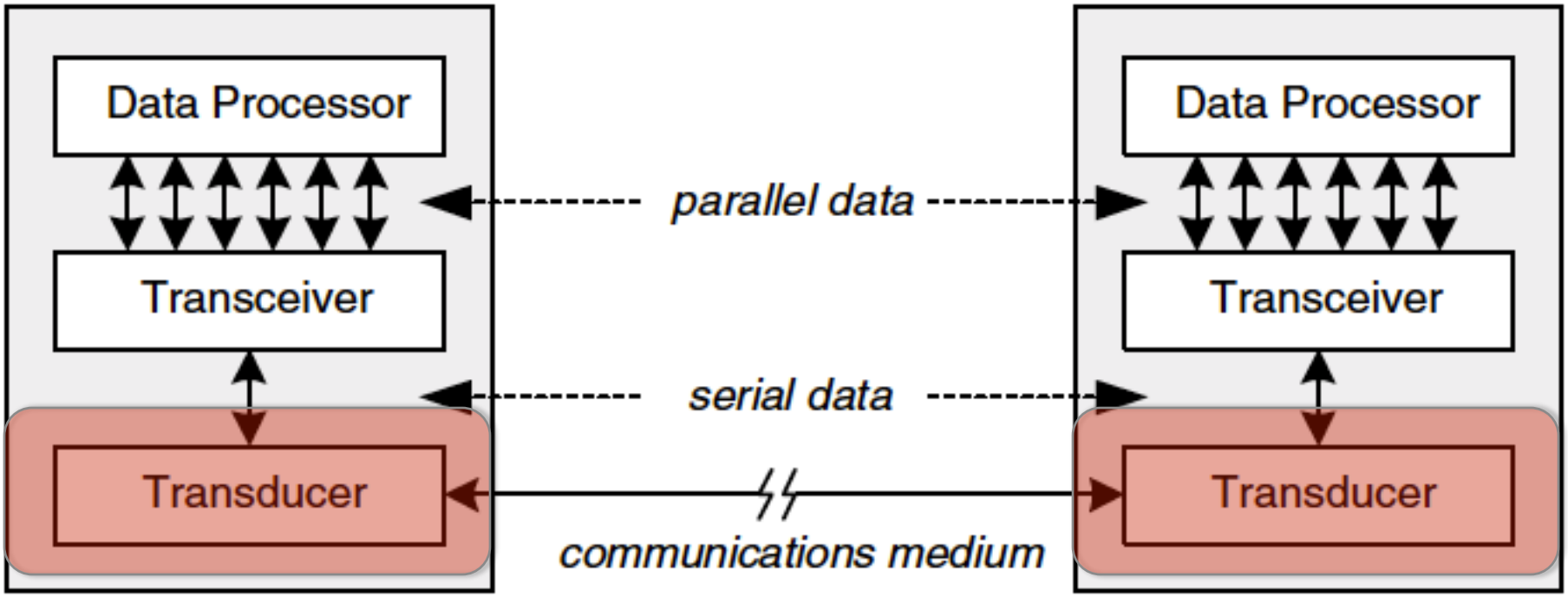
INTRODUÇÃO

- **Na interface de comunicação serial, somente um bit de informação é transmitido/recebido por vez**
- **Como os dados geralmente são processados em paralelo (por um microprocessador, por exemplo), há a necessidade de convertê-los em uma sequência de bits (e vice-versa)**
- **Um *transceptor* é o componente responsável pela conversão paralela <=> serial e pelo controle da transmissão**

COMUNICAÇÃO SERIAL



COMUNICAÇÃO SERIAL



O transdutor é responsável pela conversão entre sinais eletromagnéticos do meio de comunicação e os níveis lógicos do transceptor

UART

- **A maioria dos circuitos de comunicação serial utiliza um transceptor conhecido como UART (*Universal Asynchronous Receiver/Transmitter*)**

- **O termo *universal* refere-se ao fato do formato do dado e velocidade serem configuráveis. Os níveis elétricos são delegados a circuitos especiais externos e não fazem parte da especificação da UART**

- **Padrões de comunicação serial baseados em UART**
  - RS-232
  - RS-422

USO DA UART

Imagine que precisamos transmitir um dado do computador 1 para o computador 2

computador 1

computador 2

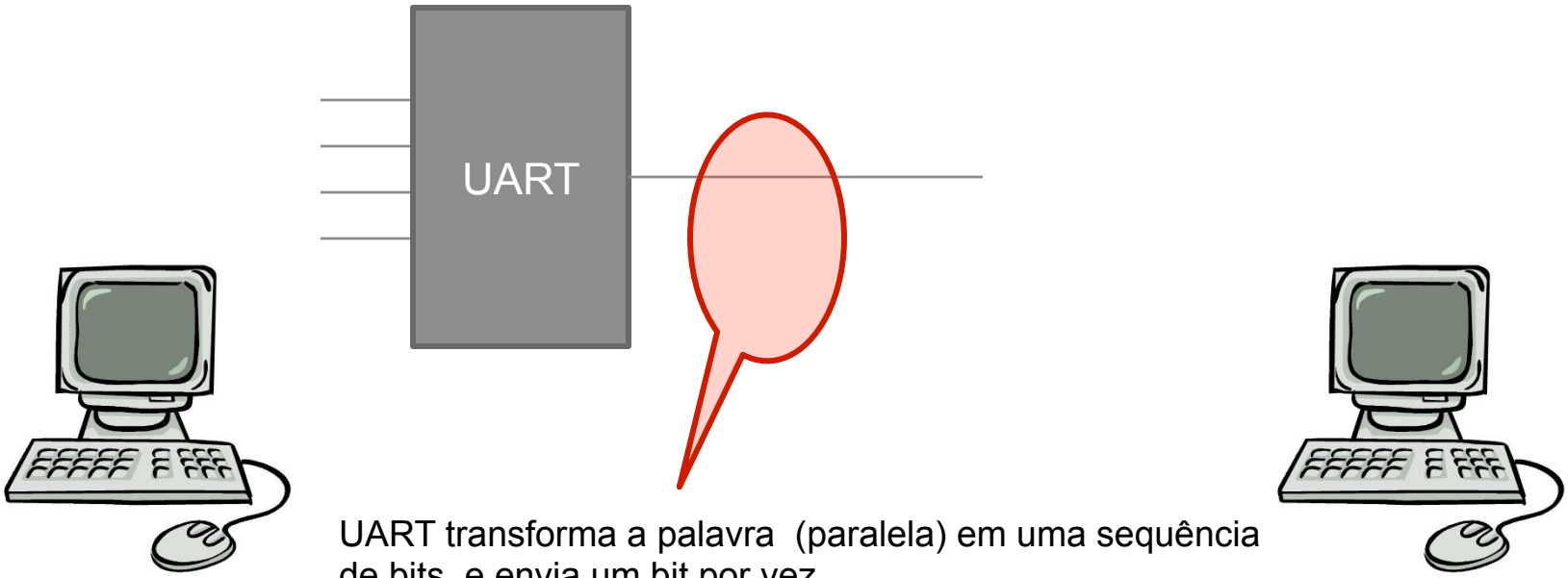
USO DA UART

Dado a ser transmitido é recebido pela UART presente na interface serial do computador 1

computador 1

computador 2

USO DA UART



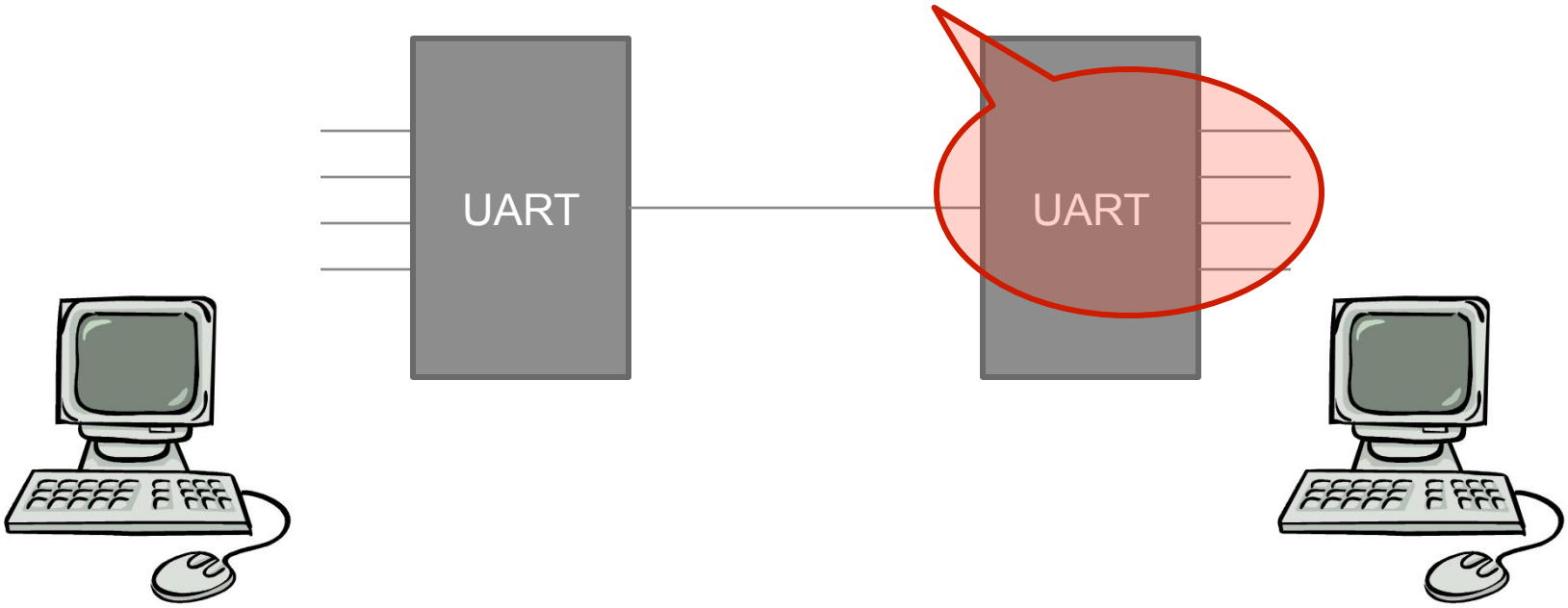
UART transforma a palavra (paralela) em uma sequência de bits e envia um bit por vez

computador 1

computador 2

USO DA UART

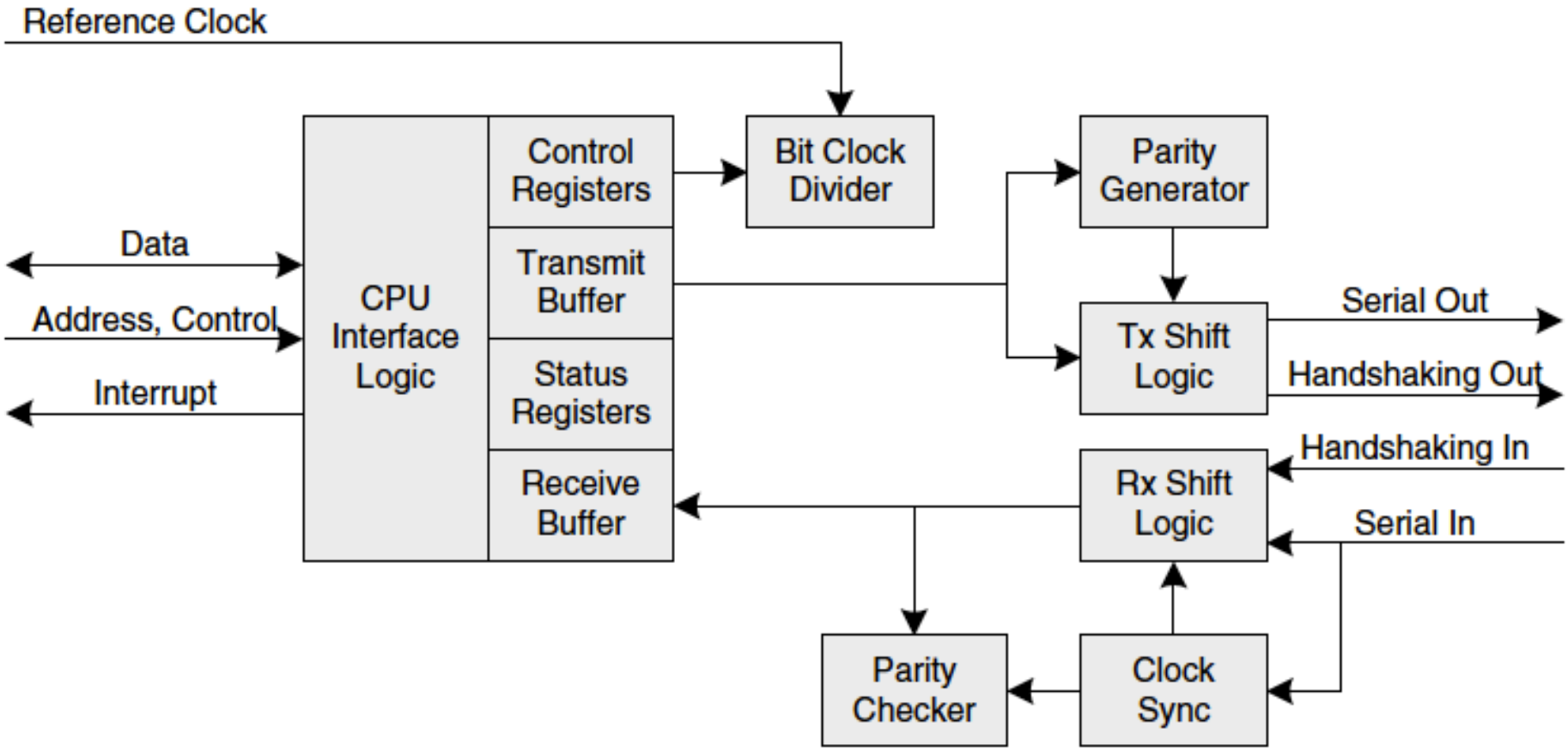
A UART do computador 2 recebe os bits individuais e os agrupa novamente em uma palavra



computador 1

computador 2

DIAGRAMA DE BLOCOS - UART

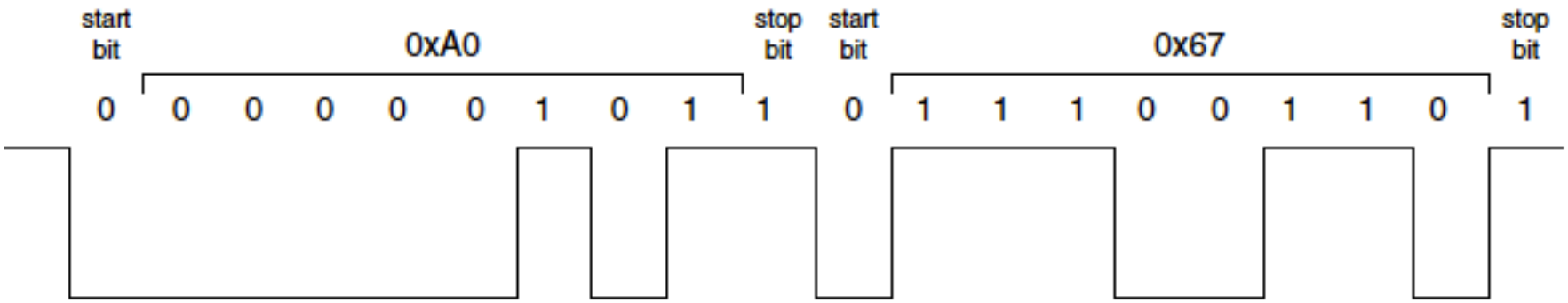


UART

- **A comunicação usada pela UART é assíncrona, ou seja, comandos precisam ser passados para sincronizar a comunicação**
  - Geralmente um bit de *start* é enviado antes dos bits referentes à palavra serem enviados
  - Um bit de *stop* é enviado ao final para informar o término do envio
  - É comum haver um bit de paridade para checar por eventuais erros na transmissão

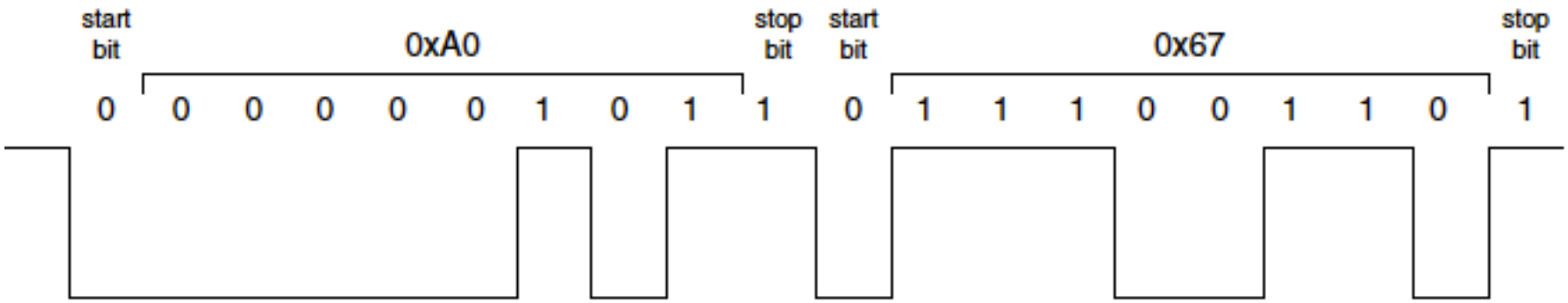
- **O tamanho da palavra a ser recebida/transmitida, assim como velocidade da transmissão, devem ser configurados e acordados entre receptor e transmissor**

EXEMPLOS DE COMUNICAÇÃO

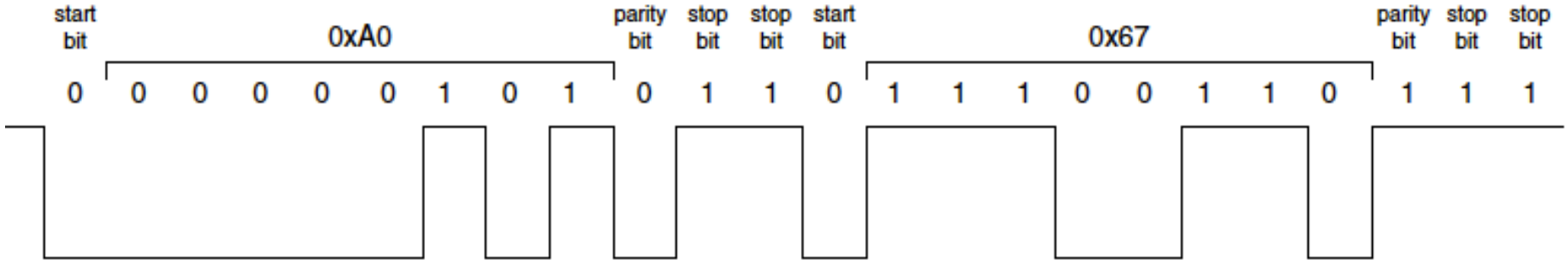


palavras de 8 bits, sem paridade, 1 bit de parada, LSB first

EXEMPLOS DE COMUNICAÇÃO



palavras de 8 bits, sem paridade, 1 bit de parada, LSB first



palavras de 8 bits, paridade par, 2 bits de parada, LSB first

UART

- **Existem diversos modelos de circuitos UART disponíveis**
  - A UART 8250 foi usado no IBM XT
- **Na aula do laboratório de hoje usaremos a UART do módulo JTAG**
  - Acesso é feito via entrada e saída mapeada em memória
    - Registrador de dados
    - Registrador de controle

COMUNICAÇÃO ENTRE PC E FPGA

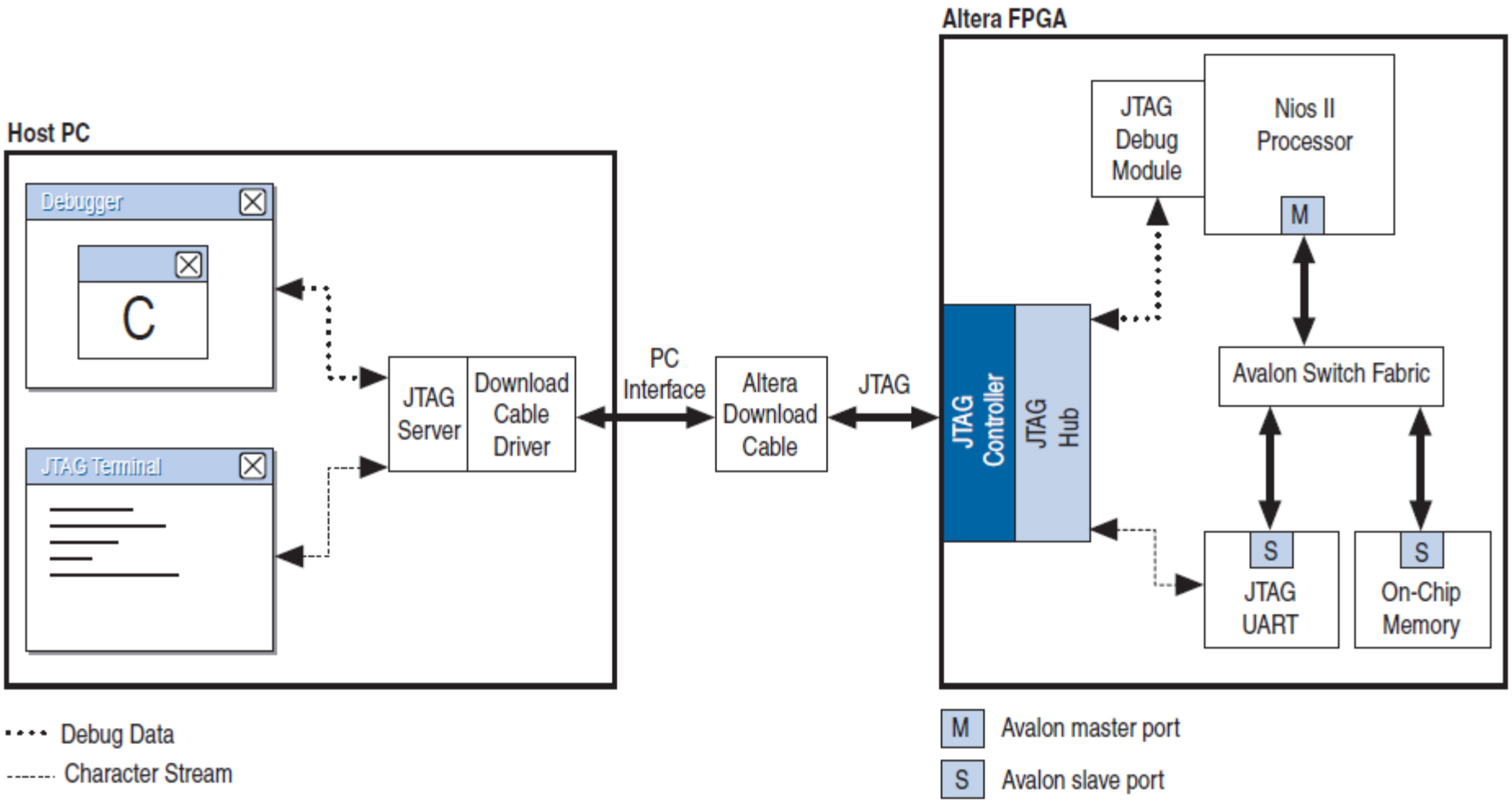
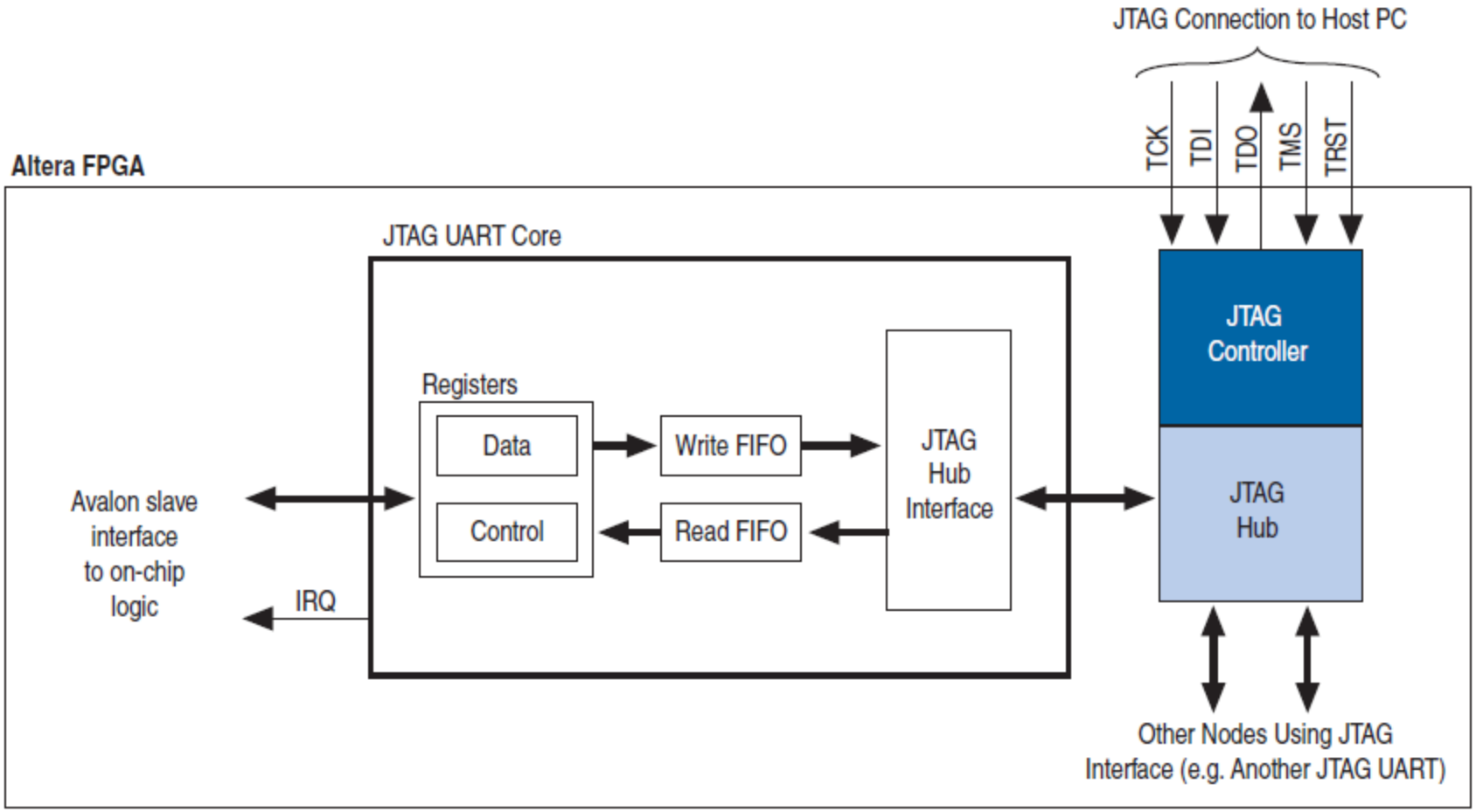


DIAGRAMA DE BLOCOS – JTAG UART



REGISTRADOR DE DADOS – JTAG UART

Offset	Register Name	R/W	Bit Description														
			31	...	16	15	14	...	11	10	9	8	7	...	2	1	0
0	data	RW	RAVAIL			RVALID	*(1)*						DATA				
1	control	RW	WSPACE			*(1)*				AC	WI	RI	*(1)*			WE	RE

(1) Reserved. Read values are undefined. Write zero.

Table 5–3. data Register Bits

Bit Number	Bit/Field Name	Read/Write/Clear	Description
0 .. 7	DATA	R/W	The value to transfer to/from the JTAG core. When writing, the DATA field is a character to be written to the write FIFO. When reading, the DATA field is a character read from the read FIFO.
15	RVALID	R	Indicates whether the DATA field is valid. If RVALID=1, then the DATA field is valid, else DATA is undefined.
16 .. 32	RAVAIL	R	The number of characters remaining in the read FIFO (after this read).

REGISTRADOR DE CONTROLE – JTAG UART

Offset	Register Name	R/W	Bit Description														
			31	…	16	15	14	…	11	10	9	8	7	…	2	1	0
0	data	RW	RAVAIL			RVALID	*(1)*						DATA				
1	control	RW	WSPACE			*(1)*				AC	WI	RI	*(1)*			WE	RE

(1) Reserved. Read values are undefined. Write zero.

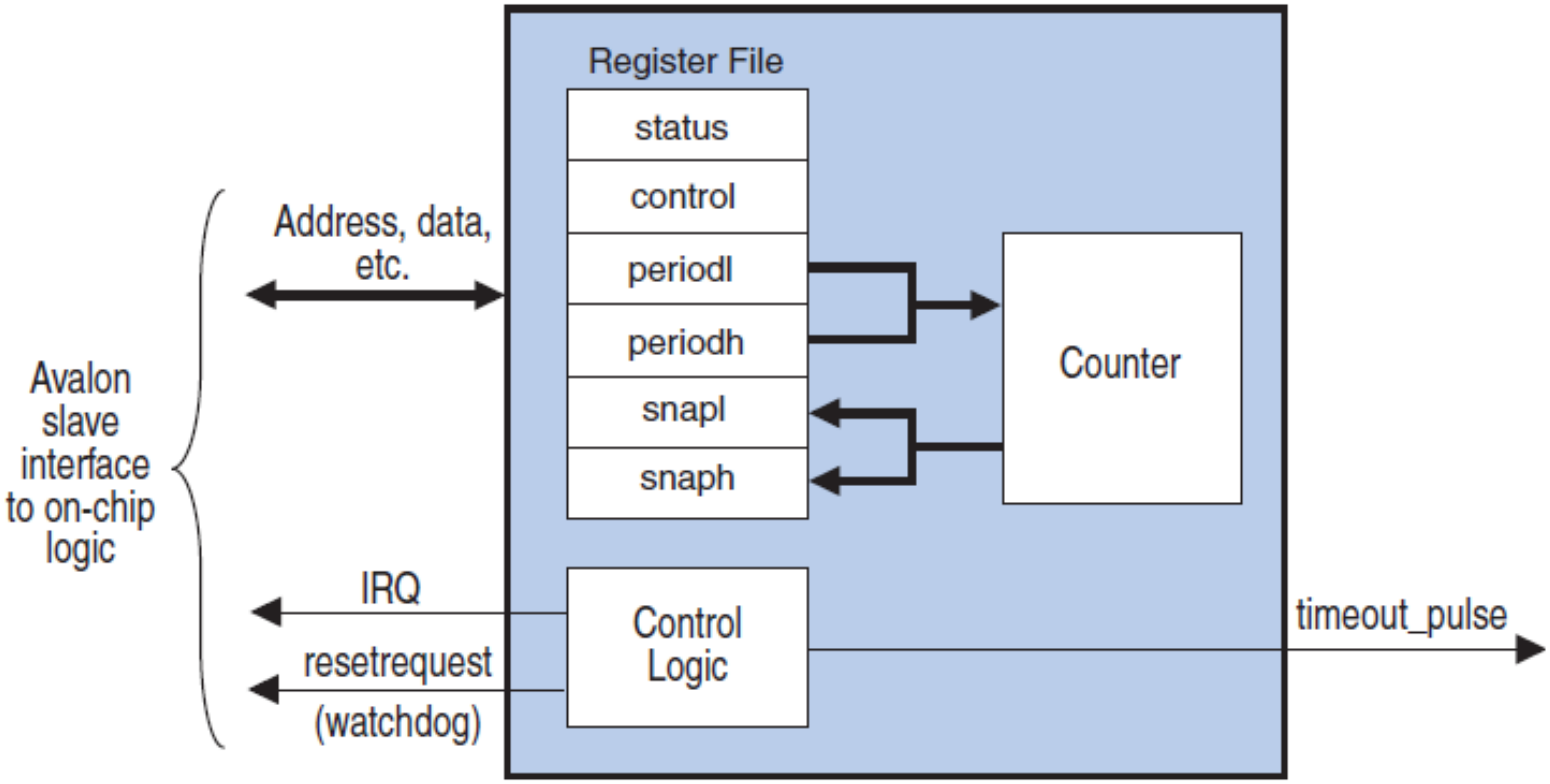
Table 5–4. control Register Bits

Bit Number	Bit/Field Name	Read/Write/Clear	Description
0	RE	R/W	Interrupt-enable bit for read interrupts
1	WE	R/W	Interrupt-enable bit for write interrupts
8	RI	R	Indicates that the read interrupt is pending
9	WI	R	Indicates that the write interrupt is pending
10	AC	R/C	Indicates that there has been JTAG activity since the bit was cleared. Writing 1 to AC clears it to 0.
16 .. 32	WSPACE	R	The number of spaces available in the write FIFO.

TEMPORIZADOR

- **Circuito que permite gerar interrupções em intervalos regulares de tempo**
- **No sistema *DE2 Basic Computer System* existe um temporizador de 32 bits que usaremos neste laboratório**
- **O acesso a este temporizador também é feito via o esquema de entrada e saída mapeada em memória**

DIAGRAMA DE BLOCOS DO TEMPORIZADOR



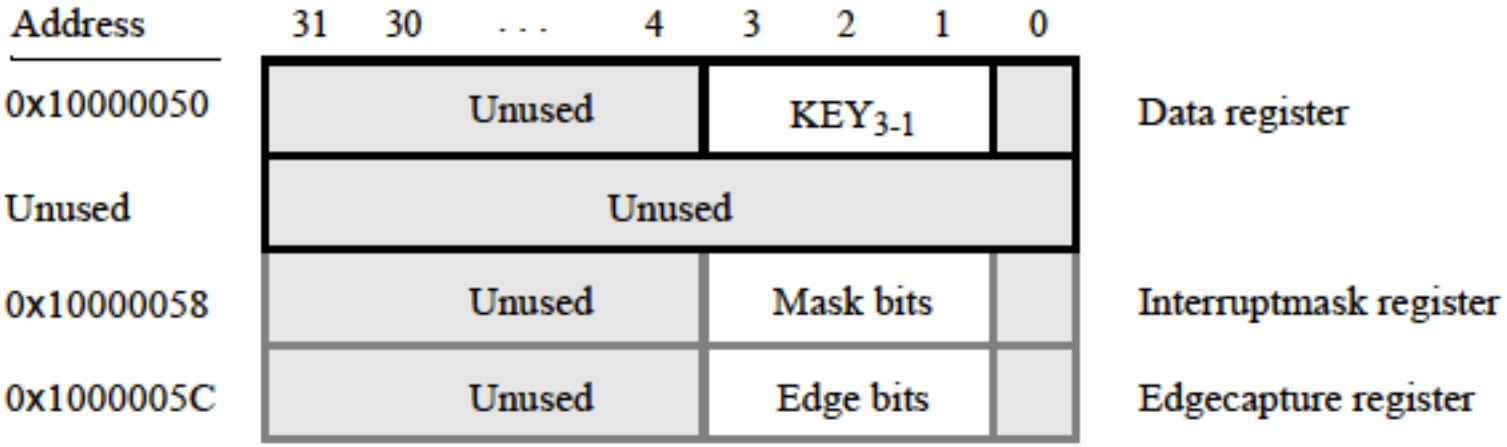
TEMPORIZADOR

- **Para usar o temporizador**
  - escrevemos nos respectivos registradores (*periodl* e *periodh*) o valor da contagem (32 bits)
  - habilitamos a interrupção (bit 0 de *control*)
  - iniciamos a contagem (bit 2 de *control*)

- **O valor da contagem é decrementado automaticamente (usando um clock de 50MHz) e, quando atingir o valor 0, uma interrupção é gerada**
  - O bit 1 de *control* indica se a contagem é reiniciada ou não

DICAS

- **Acesso aos dispositivos de I/O**



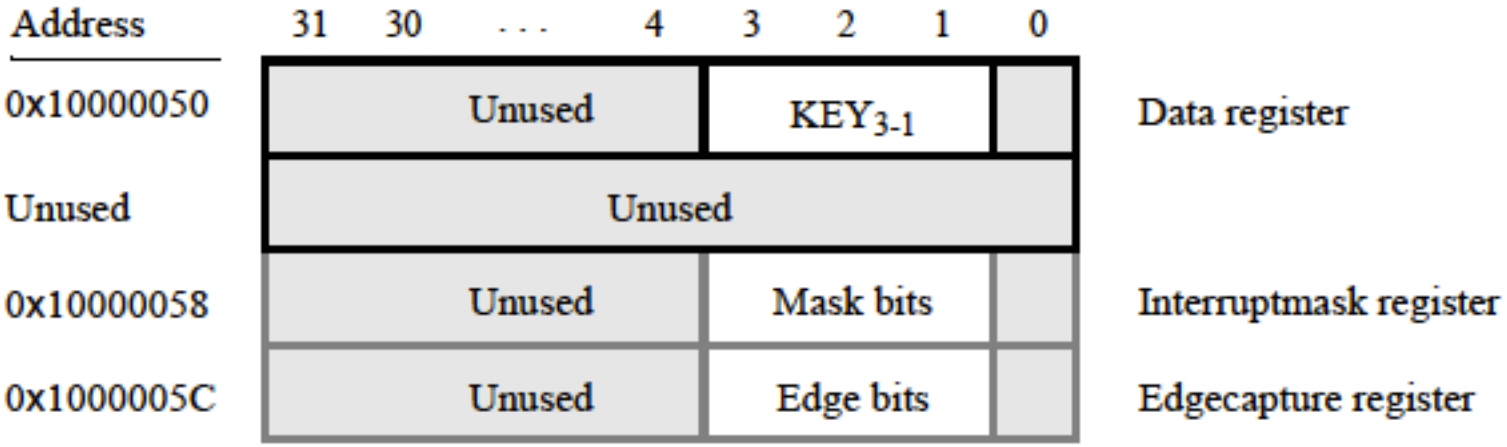

```
.equ PB_MASK, 0x10000058
.equ PB_EDGE, 0x1000005C

…
movia r2, PB_MASK
movia r3, PB_EDGE

ldwio  r10, (r2)
ldwio  r11, (r3)
```

DICAS

- **Acesso aos dispositivos de I/O**




```
.equ PB_MASK, 0x10000058
.equ PB_EDGE, 0x1000005C

…
movia r2, PB_MASK
movia r3, PB_EDGE

ldwio  r10, (r2)
ldwio  r11, (r3)
```

```
.equ PB_BASE, 0x10000050

…
movia r2, PB_BASE

ldwio  r10, 8(r2)
ldwio  r11, 12(r2)
```

DICAS

- **Tratamento de interrupção**
- **Procure usar subrotinas (inclusive salvando contexto!)**

```
.org 0x20
EXCEPTION_HANDLER:
        rdctl   et, ipending
        beq     et, r0, NOT_HW_INTERRUPT
        subi    ea, ea, 4                   # subtrai 4 de ea (instrucao deve ser reexecutada)
        andi    et, et, 02                  # certifica que interrupcao foi gerada por PB
        beq     et, r0, ERROR

        call    PUSHBUTTON_ISR

NOT_HW_INTERRUPT:           # excecao nao foi causada por hardware externo
ERROR:                      # excecao nao foi causada por PB – ignora
        eret
```

```
PUSHBUTTON_ISR:
# salva pilha
        # executa tratamento de interrupcao do pushbutton
# recupera pilha
        ret
```