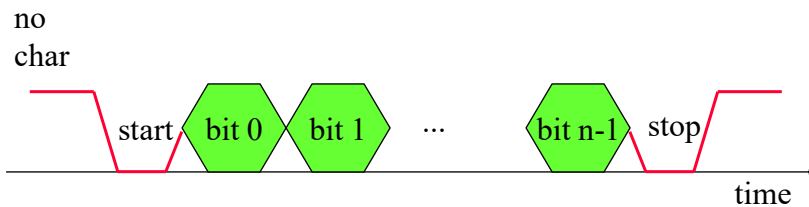


Comunicação em série

- z Carateres são transmitidos separadamente
- z Todos começam com um bit "0" e terminam com um bit "1":



5

5

Parâmetros de comunicação em série

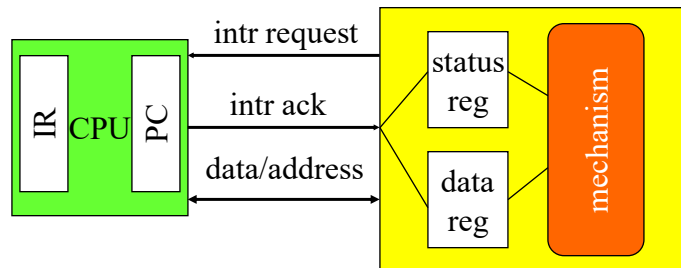
- z *Baud (bit) rate* (nº mudanças de sinal por segundo) (velocidade da comunicação)
- z Número de bits por caractere.
- z Com ou sem paridade (verificação de erros)
- z Paridade par ou ímpar.
- z Comprimento do bit de paragem (1, 1.5, 2 bits). (sinaliza o fim da comunicação)

6

6

Interface de Interrupções I/O

- z As interrupções permitem que um dispositivo altere o fluxo de controlo do CPU
- y Provocam a chamada de uma sub-rotina (*ISR*) para gerir o dispositivo



ISR = interrupt service routine

11

11

Comportamento de Interrupção

- z Baseado no mecanismo de chamada de uma sub-rotina
- z A interrupção força a próxima instrução a ser uma chamada a uma sub-rotina para uma localização predeterminada
- y O endereço de retorno é gravado para permitir a retoma ao *foreground program*.

12

12

Interface físico interrupção

- z O *CPU* e o dispositivo estão conectados pelo barramento do *CPU*
- z Negociação do *CPU* e dispositivo (*handshake*):
 - y dispositivo alega pedido de interrupção;
 - y *CPU* reconhece a interrupção (*acknowledge*) quando puder lidar com ela.

13

13

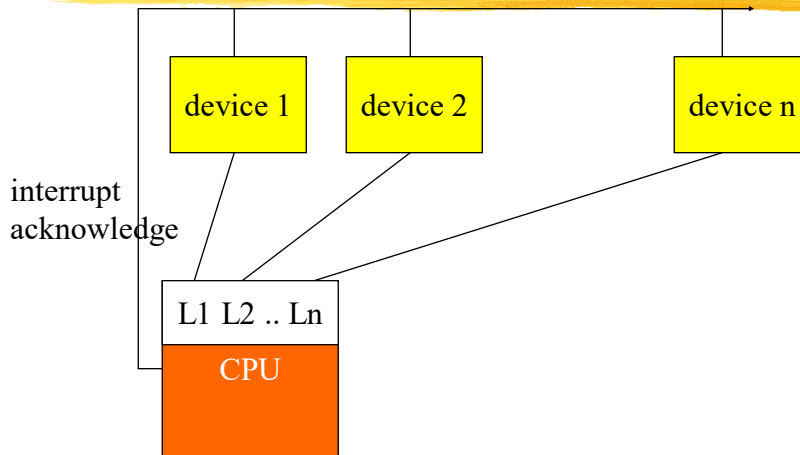
Prioridades e vetores

- z Dois mecanismos permitem tornar as interrupções mais específicas:
 - y **Prioridades** determinam quais as interrupções que acedem ao *CPU* mais rapidamente
 - y **Vetores** determinam que código é chamado para cada tipo de interrupção
- z Os mecanismos são ortogonais: a maioria do *CPUs* fornecem ambos

14

14

Interrupções prioritárias



15

15

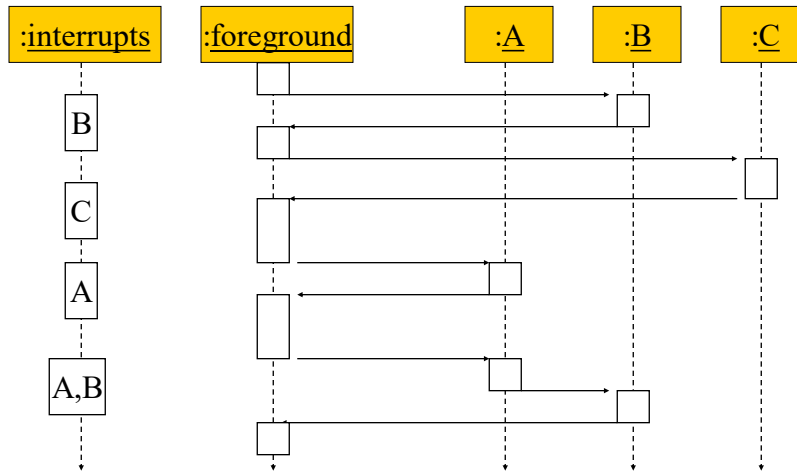
Prioridades de Interrupções

- z **Masking**: interrupções com prioridade inferior à interrupção atual não são reconhecidas enquanto a interrupção não estiver completa
- z **Non-maskable interrupt (NMI)**: prioridade de maior valor, nunca é mascarada.
 - y Frequentemente utilizada para *power-down*

16

16

Exemplo: //O Prioritários

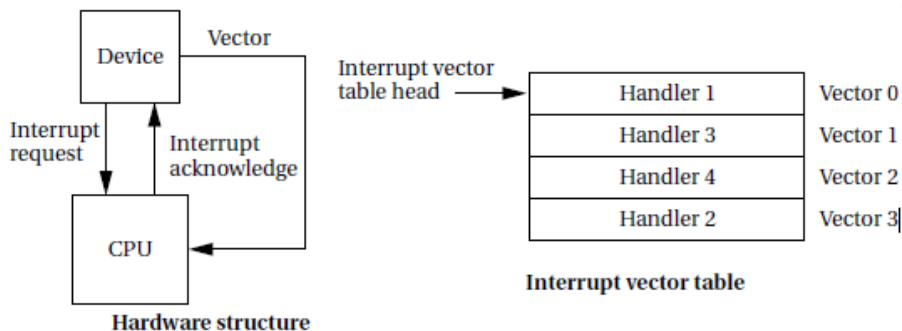


17

17

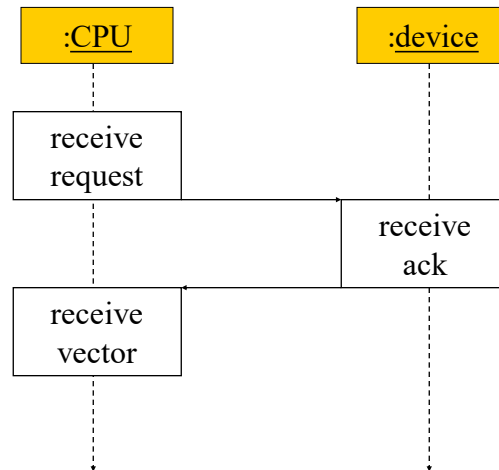
Vectores de Interrupt

- z Permite que diferentes dispositivos sejam manipulados por códigos diferentes.
- z Tabela de vetores de Interrupts :



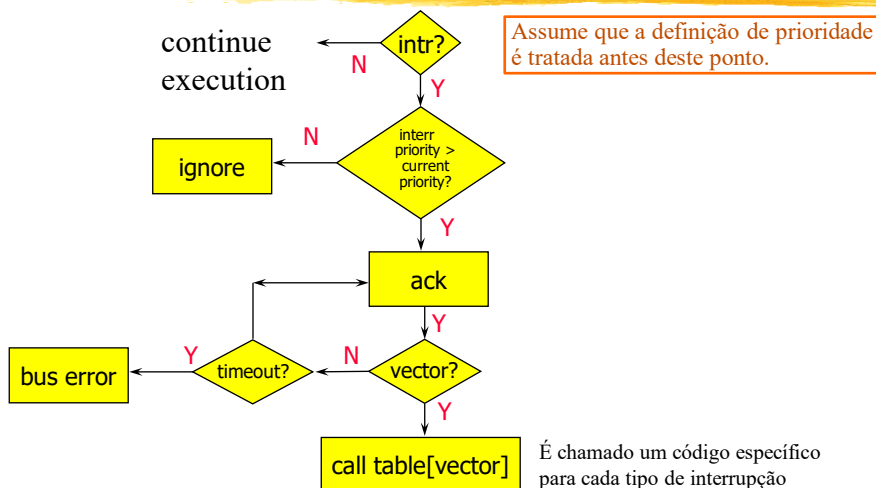
18

Atribuição de vetores de Interrupts



19

Mecanismo de interrupções genérico



20

20

Sequência de Interrupção

1. CPU afirma o pedido (*ACK*)
2. Dispositivo envia o vetor
3. CPU chama o *handler*
4. Software processa o pedido
5. CPU restaura o estado ao programa *foreground*

21

21

Fontes da sobrecarga do interrupt

- z Tempo de execução do *Handler*.
- z Sobrecarga do mecanismo do *interrupt*.
- z *Save* e *restore* dos registos
- z Penalizações relacionadas com *pipeline*.
- z Penalizações relacionadas com *cache*.

22

Interrupções do ARM

- z ARM7 suporta 2 tipos de interrupções:
 - y *Fast interrupt requests (FIQs)* – Interrupts mais rápidos, permitem atalhar algumas etapas do *handler*
 - y *Interrupt requests (IRQs)*
- z A tabela de interrupções inicia-se na localização 0

23

23

Procedimento de interrupções do ARM

- z Ações do CPU:
 - y Salva o PC. Copia CPSR para o SPSR.
 - y Força os bits no CPSR a guardar a interrupção
 - y Força o PC para o vetor
- z Responsabilidades do *handler* :
 - y Restaura o PC correto.
 - y Restaura CPSR do SPSR.
 - y Limpa (*interrupt disable flags*).

SPSR - saved program status register
CPSR - Current program status register
PC - program counter

24

24

Latência das interrupções do *ARM*

- z O pior caso de latência para resposta a uma interrupção é de 27 ciclos:
 - y 2 ciclos para sincronizar o pedido externo
 - y Até 20 ciclos para completar a instrução em curso
 - y 3 ciclos para abortar dados
 - y 2 ciclos para entrar no estado de *handling* da interrupção

25

25

Modo Supervisor

- z Proporciona uma forma de proteção de memória entre programas
 - y Impede a corrupção de memória
- z Necessidade de **modo supervisor** para gerir os vários programas

26

26

Modo Supervisor do ARM

- z Uso da instrução *SWI* (*software interrupt*) para entrada no modo supervisor, ex:

```
SWI CODE_1
```

- z Define *PC* em 0x08
- z O argumento para o *SWI* é passado com o código do modo supervisor
- z Salva *CPSR* no *SPSR*

27

27

Exceção

- z **Exceção**: erro detetado internamente
- z As exceções são síncronas com instruções mas imprevisíveis
- z O mecanismo de exceções está no topo do mecanismo de interrupções
- z As exceções são habitualmente priorizadas e vetorizadas
- z Um exemplo simples é a divisão por zero

28

28

Trap

- z **Trap** (interrupção de software): exceção gerada por uma instrução/processo
 - y Exemplos: Chamada do modo supervisor, acesso à memória inválido, etc.
- z O ARM utiliza a instrução *SWI* para as **traps**

29

29

Co-processor

- z **Co-processor**: unidade de funcionamento invocada por uma instrução
 - y Ex: co-processor de vírgula flutuante, *ALU*
- z O ARM permite até 16 co-processadores
 - y O co-processor de vírgula flutuante utiliza as unidades 1, 2

30

30

Sistemas Embebidos



Ficha de trabalho nº3