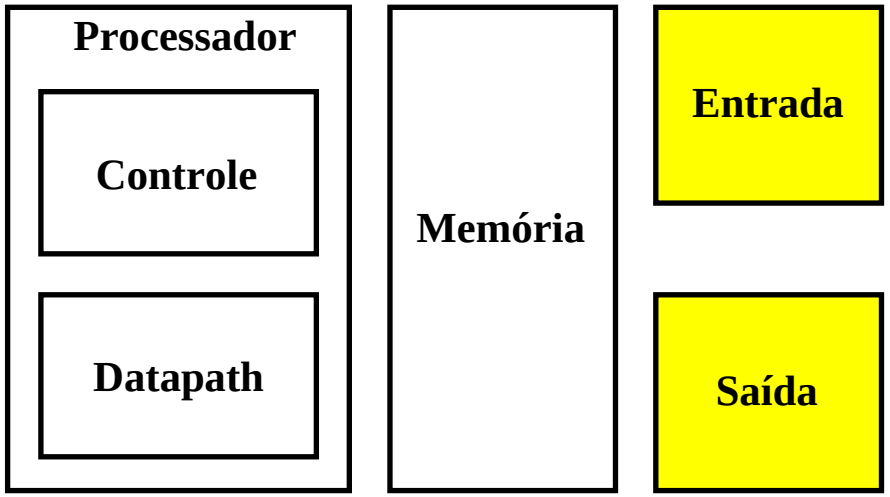
Aula 25:
Polling, Interrupção



Interação CPU e Dispositivo

- **Técnicas de monitoramento:**
  - **Consulta ou "polling"**
  - **E/S acionada por interrupção**
  - **Acesso direto à memória ("DMA")**
- **Pressupõem:**
  - **Bits do registrador de "status" (status_reg)**
    - » **Indicação de tarefa completa ("done"),**
    - » **Indicação de erro, etc.**

“Polling”

- **Requisitos**
  - **Dispositivo escreve informação no status_reg**
  - **CPU lê informação do status_reg**
- **CPU consulta periodicamente o status_reg**
  - **Para avaliar se nova operação de E/S pode ser disparada**
- **CPU tem controle total da transação**
  - **Mas executa todo o trabalho**

“Polling”: estrutura típica

```
#define status_reg 0x1000
#define data_reg 0x1001
#define done 0x0001
while (TRUE)   {
  if (get_register(status_reg) = = done)
      put_register(new_data, data_reg);
  else
      do_something_else;
  ...
}
```

“Polling”

- **Vantagem:**
  - **Método mais simples de comunicação CPU ↔ E/S.**
    - » **Implementação totalmente em SW.**
- **Desvantagem:**
  - **Pode desperdiçar tempo da CPU**
    - » **CPU é muito mais rápida que dispositivo de E/S!**
  - **CPU pode ler status_reg várias vezes**
    - » **Só para descobrir que ele não terminou a tarefa**
  - **Ao final, status_reg deve ser lido de novo**
    - » **Para verificar se transação teve sucesso**

“Polling”: uso

- **Disp. que iniciam E/S de forma autônoma**
  - **Baixa taxa de E/S**
    - » **Ex. Interfaceamento com mouse**
- **Disp. que iniciam E/S sob controle do SO**
  - **Consulta apenas quando o dispositivo está ativo**
    - » **Ex. disco, impressora**
- **Sistemas de tempo real**
  - **Taxa de E/S pré-determinada**
  - **Torna o encargo de E/S mais previsível**

E/S acionada por interrupção

- **Dispositivo de E/S usa interrupção**
  - **Para indicar que precisa de atenção**
  - **Para indicar que terminou uma tarefa**
- **Interrupção é assíncrona**
  - **Não está associada a instrução alguma**
  - **Não impede execução de instrução alguma**
    - » **Diferente de exceções como “overflow” aritmético**
  - **Unidade de controle**
    - » **Verifica se há interrupção pendente**
    - » **Antes de executar a próxima instrução**

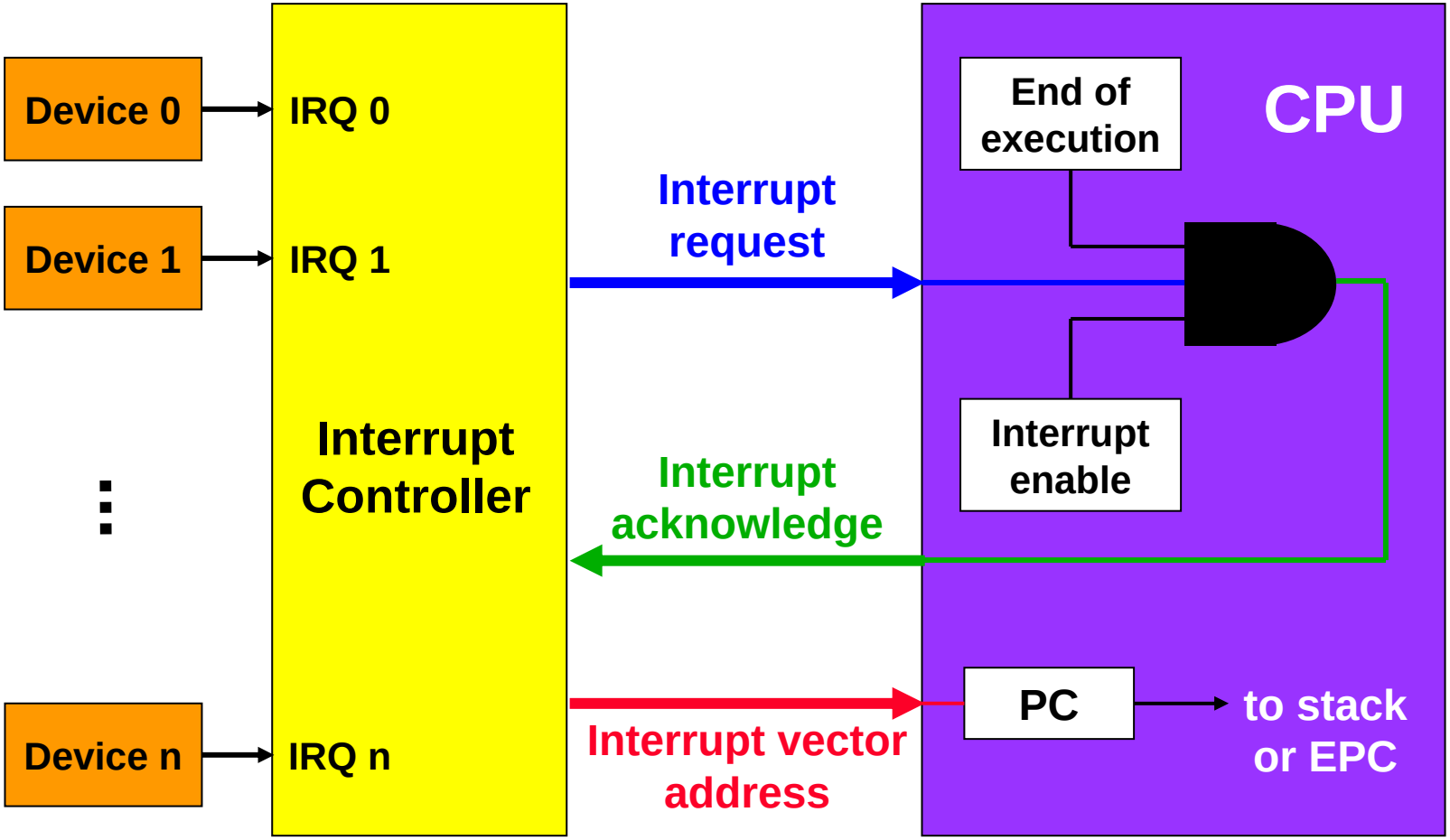
E/S acionada por interrupção

- **Requisitos:**
  - **Salvamento do contexto da CPU**
    - » **Antes do tratamento: registradores salvos na pilha**
    - » **Depois do tratamento: registradores restaurados**
  - **Identificação do dispositivo solicitante**
    - » **Interrupção vetorizada**
    - » **Registrador de causa**
  - **Priorização de interrupções**
    - » **Para capturar a urgência relativa das tarefas**

E/S acionada por interrupção

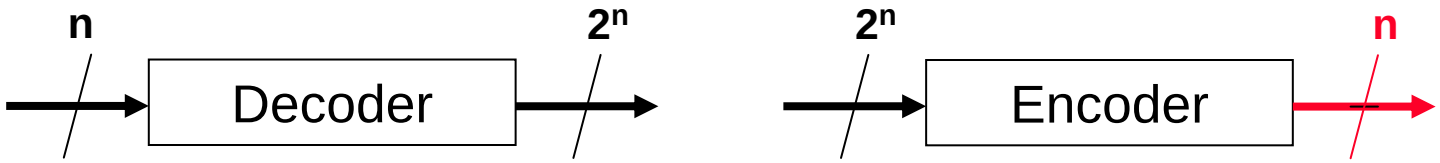
- **Alternativa 1:**
  - **Controlador de interrupção externo +**
  - **Vetor de interrupção**
    - » **Intel 8259A (82093AA) em PCs**
- **Alternativa 2:**
  - **“Controlador” de interrupção embutido +**
  - **Com registrador de causa**
    - » **coprocessor 0 do MIPS**

Processamento de interrupções



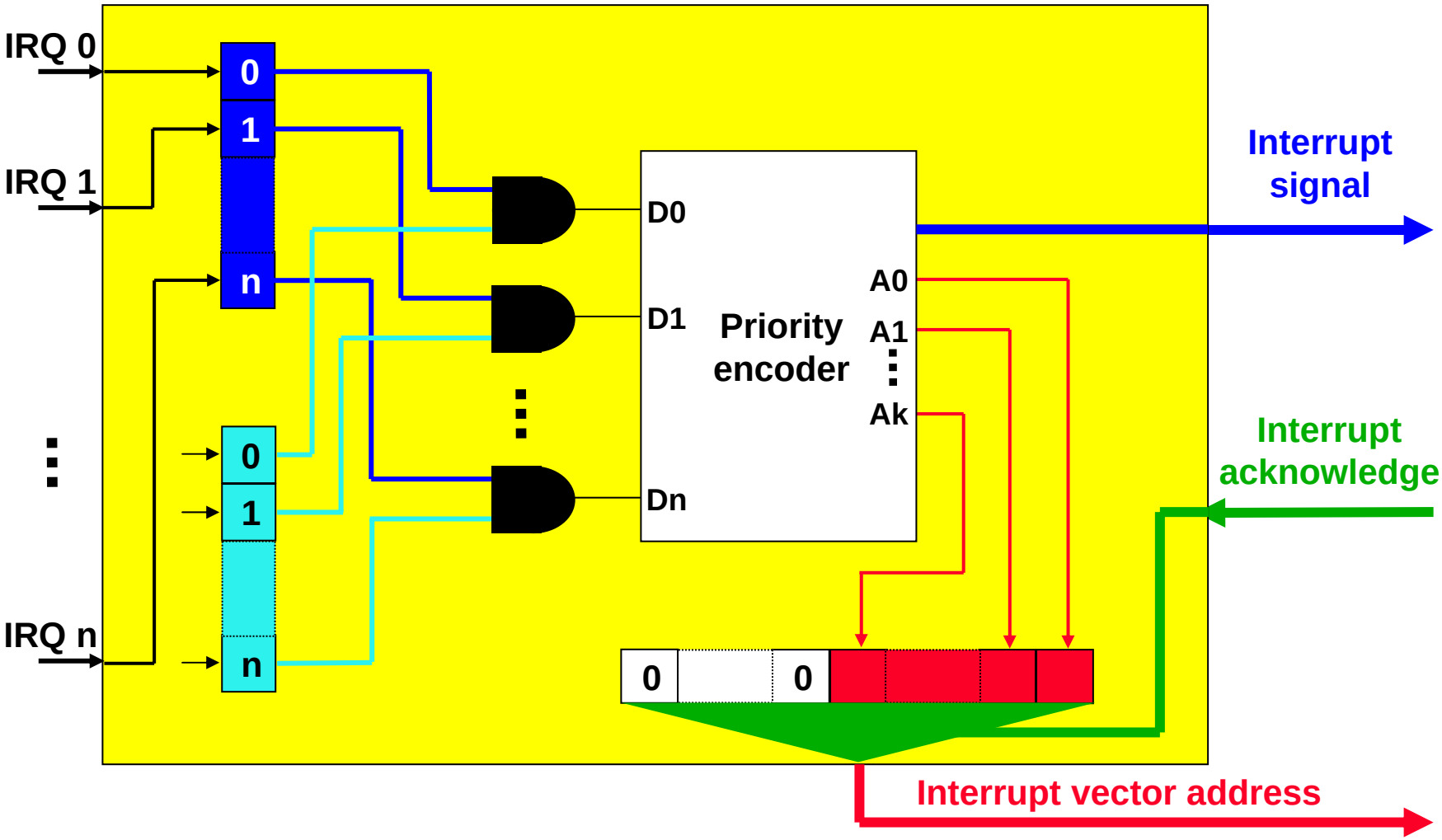
Priorização de Interrupções

- **Codificador de prioridade:**
  - **O que é mesmo um codificador (decodificador) ?**



D3	D2	D1	D0	A1	A0	V
0	0	0	0	0	0	0
X	X	X	1	0	0	1
X	X	1	0	0	1	1
X	1	0	0	1	0	1
1	0	0	0	1	1	1

Controlador de Interrupções



Interrupção: níveis de priorioridade

- **Mecanismos de interrupção**
  - **Costumam ter vários níveis de prioridade**
    - » **Ex. UNIX usa de 4 a 6 níveis**
- **Nível indica a ordem de processamento**
- **Exceções internas e interrupções de I/O**
  - **Têm prioridades associadas**
    - » **Típico: Interrupções de E/S têm mais baixa prioridade**

MIPS: priorização de interrupções

- **HW não oferece priorização**
  - **Não há codificador de prioridades**
    - » **HW trata todos os bits de interrupção da mesma forma**
- **Suporte para priorização no MIPS**
  - **Provê primitivas**
  - **Deixa o SO implementar o mecanismo**

MIPS: Coprocessor 0

- “System Control Coprocessor
  - Extensão da ISA padrão
- Controle de
  - Configuração
  - Cache
  - Unidade de gerenciamento de memória (MMU)
  - Interrupções e exceções
  - Etc.

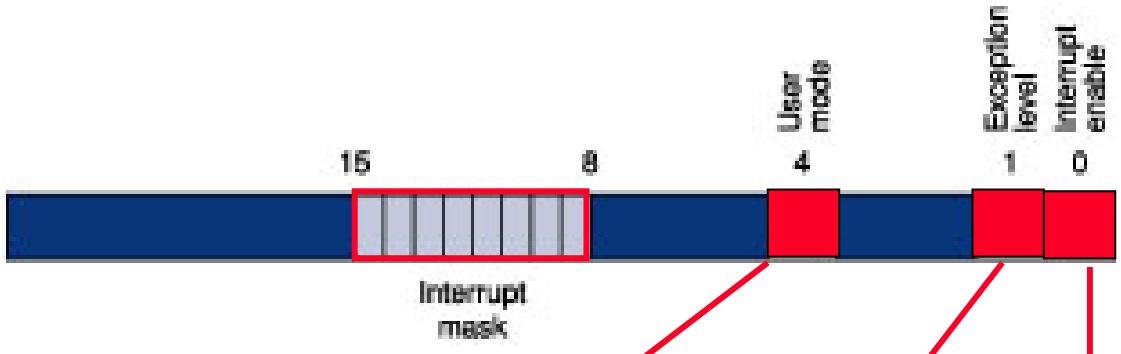
MIPS: exceções e interrupções

- **Registradores de uso específico**
  - **Cause**
    - » **O que causou essa exceção/interrupção ?**
  - **Status**
    - » **Modo privilegiado? Interrupções habilitadas ?**
- **Instruções específicas**
  - `mtc0 rs, <nn> # move to coprocessor 0`
  - `mfco rd, <nn> # move from coprocessor 0`
- **Ativando desativando bits**
  - `andi t0, t0, <zero nos bits a desativar>`
  - `ori t0, t0, <um nos bits a ativar>`

MIPS: exceções e interrupções

- **Suporte para seu tratamento**

Status



0: user

1: kernel

0: no exception

1: exception occured

(EPC não atualizado se nova exceção ocorer)

0: intr disabled

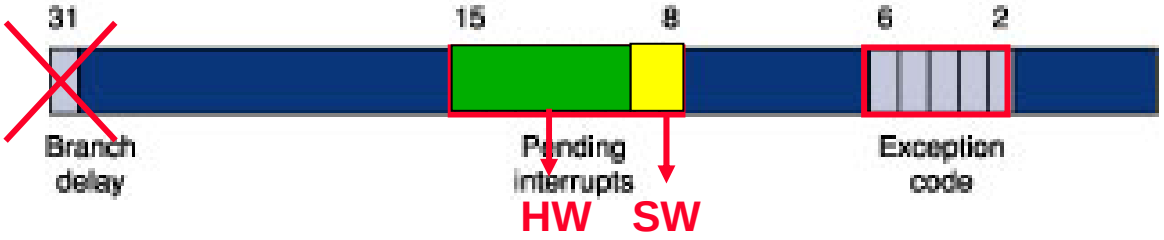
1: intr enabled

MIPS: exceções e interrupções

- **Suporte para seu tratamento**

Number	Name	Cause of exception
0	Int	Interrupt (hardware)
4	AdEL	Address error exception (load or instruction fetch)
5	AdES	Address error exception (store)
6	IBE	Bus error on instruction fetch
7	DBE	Bus error on data load or store
8	Sys	Syscall exception
9	Bp	Breakpoint exception
10	RI	Reserved intruction exception
11	CpU	Coprocessor unimplemented
12	Ov	Arithmetic overflow exception
13	Tr	Trap
15	FPE	Floating point exception

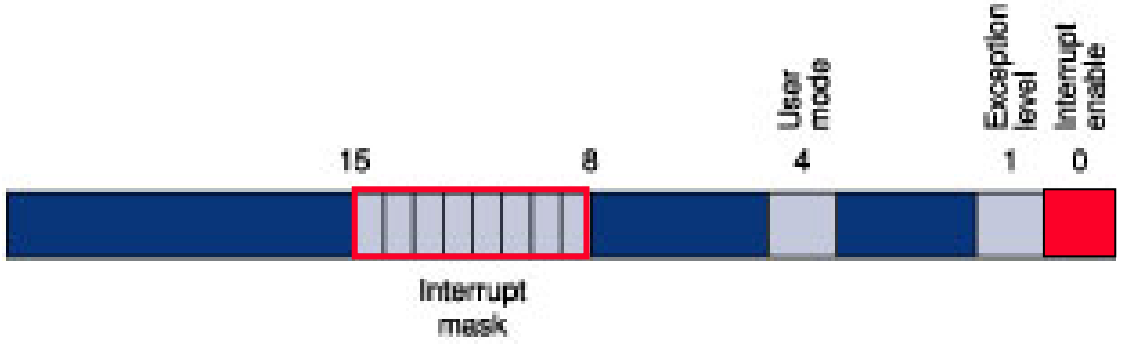
Cause



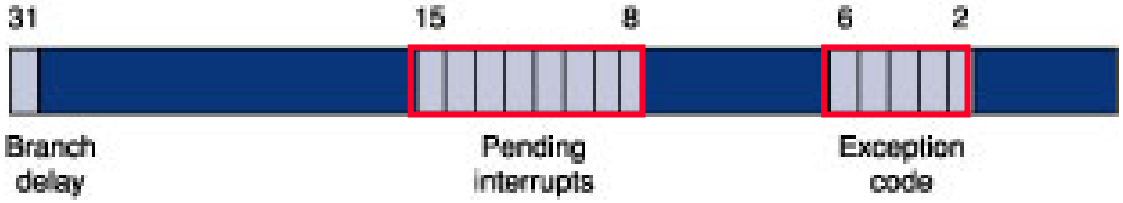
MIPS: suporte para interrupção

- **Registradores-chave**

Status



Cause



MIPS: etapas do tratamento

- **Hipótese: Exception code = 00000**
  - **Logo: IE = 1 e EXL = 0**
- **Desabilitar interrupções (EXL ← 1** [ou **IE ← 0**]**)**
- **Aplicar máscara às interrupções pendentes**
  - **Copiar Cause e Status (Usando `mfc0`)**
  - **Fazer o AND dos campos respectivos**
- **Selecionar interrupção de maior prioridade**
  - **Tipicamente, a mais à esquerda**
- **Salvar a máscara de interrupção**
  - **Por exemplo, em .kdata (ou todo o Status)**

MIPS: etapas do tratamento

- **Desabilitar interrupções menos prioritárias**
  - **Igual ou menor prioridade (alterando máscara)**
  - **Reescrevendo a máscara modificada em Status**
- **Salvar o contexto**
  - **Registradores: só os necessários para manipular a interrupção**
- **Habilitar interrupções de maior prioridade**
  - **EXL ← 0 e IE ← 1**
- **Chamar a rotina de tratamento apropriada**
  - **Servir o dispositivo de E/S**

MIPS: etapas do tratamento

Ao término do tratamento:

- **Desabilitar interrupções (EXL ← 1 [ou IE ← 0])**
- **Restaurar o campo de máscara em Status**
- **Restaurar o contexto**
- **Habilitar novas interrupções**
  - **EXL ← 0 e IE ← 1**
- **Retornar do tratador (eret)**

E/S acionada por interrupção

- **CPU lê status_reg só para discernir ...**
  - **Tarefa terminada (“Done”)**
  - **Ocorrência de erro (“Error”)**
- **Vantagem:**
  - **Mecanismo de interrupção libera a CPU**
    - » **Não é preciso consultar periodicamente os dispositivos de E/S**
    - » **CPU executa outras tarefas até ser interrompida**

E/S acionada por interrupção

- **Desvantagem**
  - **Sobrecarrega CPU c/ transferência e gerência**
    - » **Ex. Cada escrita/leitura na MP é controlada pela CPU**
- **Para grandes transferências de/para a MP ...**
  - **Por que incomodar a CPU ?**
  - **Controlador dedicado a manipular transferências**
    - » **DMA**