Aula AOC - 28/09

segunda-feira, 28 de setembro de 2020 08:59

Problemas da E/S

- Dispositivos externos, geralmente, não estão conectados diretamente na estrutura de barramento do computador
  - Uma ampla variedade de dispositivos requerem diferentes lógicas de operação.
  - Diversidade de *interfaces* – impraticável para a CPU "*saber como*" controlar cada dispositivo.
  - Disparidade entre as taxas de transferências de dados.
  - Diferentes representações de dados.
- Todos mais lentos do que CPU e RAM
- Necessidade de **módulos entrada/saída (I/O)**
  - Intermediários na comunicação entre a CPU, RAM e os periféricos

Módulo de Entrada/Saída

- Provê uma interface padronizada para a CPU e o barramento.
- Moldada para um dispositivo de E/S específico e a seus requisitos de interface.
- Delega a CPU o gerenciamento dos dispositivos de E/S.
- A interface consiste em sinais de:
  - **Control**
  - **Status**
  - **Dados**
- Interface entre a CPU e a memória.
- Interface para um ou mais periféricos.

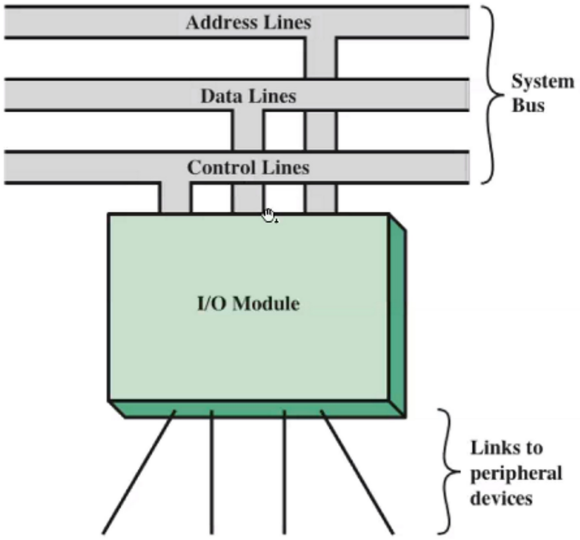


Figure 7.1 Generic Model of an I/O Module

- Interpretação humana
  - vídeo, impressora, teclado
- Operação pela máquina
  - Discos magnéticos e ópticos
  - Monitoração e controle
- Comunicação
  - Modem
  - Placa de rede ("Network Interface Card" – NIC)

Funções do Módulo de E/S

Controle e temporização (*Control & timming*)
- Coordenação do fluxo de dados

Comunicação com a CPU
- Decodificar comandos;
- Enviar e receber dados
- Informar o estado dos periféricos
- Reconhecimento de endereços

Comunicação com o dispositivo periférico
- Emitir comandos; Enviar e receber dados
- Receber informação do estado dos periféricos

Bufferização de dados (*data buffering*)
- Adequação a diferentes taxas de transferência

Deteção de erros
- Funcionamento incorreto dos periféricos
- Transmissão de dados (bit de paridade)

Etapas da operação de E/S

1. A CPU verifica o status do módulo de E/S
2. O módulo de E/S retorna o status
3. Se pronto (*ready*), a CPU requisita a transferência de dados
4. O módulo de E/S obtém os dados do dispositivo
5. O módulo de E/S transfere os dados para a CPU
6. Variações para a saída, DMA, etc.

Decisões de um módulo de E/S

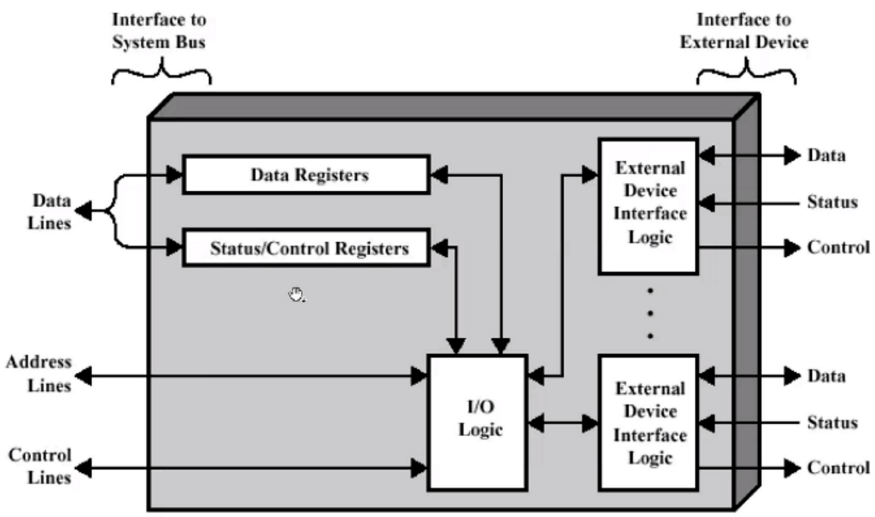
Esconder e/ou revelar as propriedades do dispositivo para a CPU.

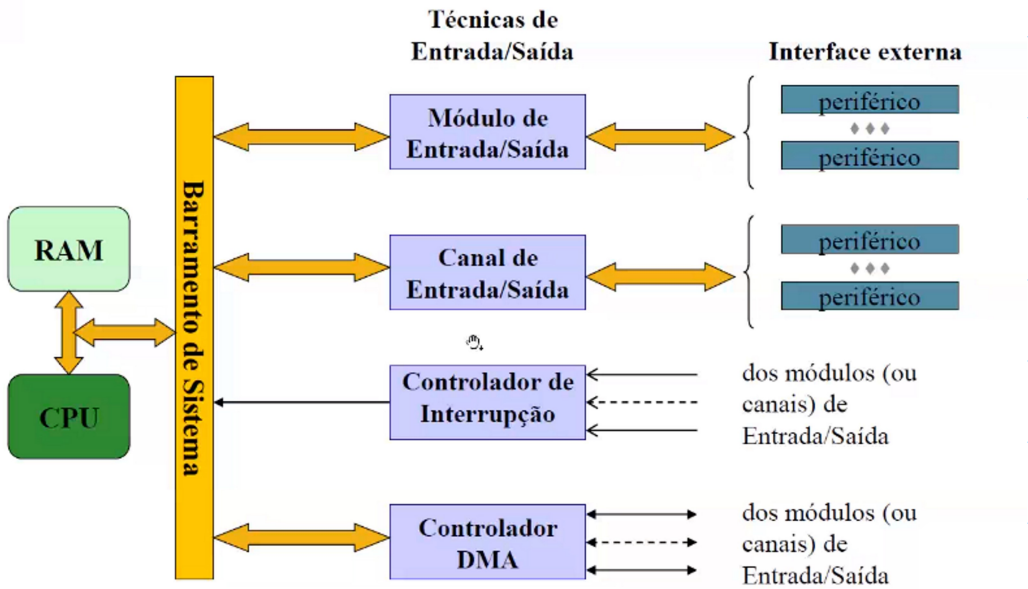
Suportar simples/múltiplos dispositivo(s).

Controlar as funções de um dispositivo ou delegar para a CPU.

Também são decisões do *Sistema Operacional*

- Por exemplo, *Unix* trata tudo, se possível, como arquivo.





Técnicas para Operações de Entrada/Saída

- Programmed I/O
- Interrupt driven I/O
- Direct Memory Access (DMA)

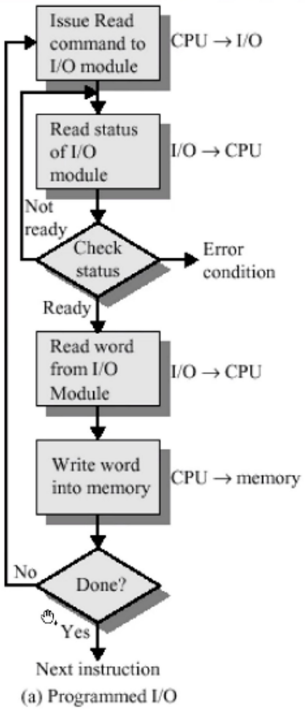
Programmed I/O

1. Operação de E/S onde a CPU envia um comando de E/S para o módulo de E/S.
2. A CPU tem controle direto da operação.
    1. Sensing *status*
    2. Read/write commands
    3. Transferring data
3. A CPU espera até que a operação de E/S seja finalizada, antes de realizar outras tarefas.
4. A finalização é indicada pela mudança dos bits de *status* do módulo de E/S.
5. A CPU deve, periodicamente, consultar (*polling*) o módulo para verificar seu *status.*
6. Isto, consome tempo de CPU.

I/O Commands

- A CPU fornece os endereços.
    - Identifica o módulo (e dispositivo, se mais do que um por módulo)
- A CPU emite os comandos
    - Controle – indicar ao módulo o que fazer
        - ✓ e.g. spin up disk
    - Teste – verificar o *status*
        - ✓ e.g. power? Error?
    - Leitura/Escrita
        - ✓ O módulo transfere, via **buffer**, dados de/para o dispositivo
- Endereçamento de dispositivos de entrada/saída
    - Nas entradas/ saídas programadas a transferência de dados consiste, muito provavelmente, num acesso à memória (ponto de vista da CPU)
    - Cada dispositivo possui um identificador único.
    - As instruções da CPU contêm um identificador (endereço).

I/O Operations



(a) Programmed I/O

Endereçando Dispositivos de E/S

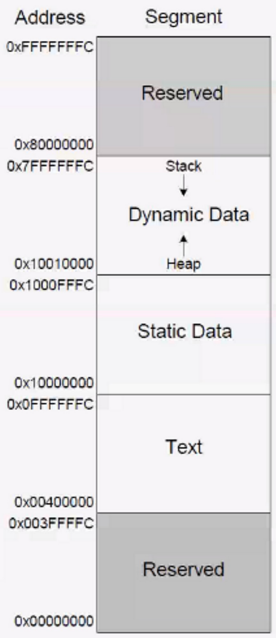
- O modo programado (***programmed I/O***) de transferência de dados é muito parecido com o acesso a memória (do ponto de vista da CPU).
- A cada dispositivo é atribuído um identificador único.
- Os comandos da CPU carregam este identificador (endereços).

Espaço de Endereçamento Lógico da Memória

- Os endereços mostrados são convenções de software (não faz parte da arquitetura MIPS)

- **Text segment**: local do código da aplicação
  - O tamanho do segment de texto é de 256MB.
- **Static and global data segment** local dos dados globais estáticos e constantes da aplicação.
  - O tamanho do segmento de dados global é de 64KB.
- **Dynamic data segment** contempla stack e heap
  - Os dados nesse segmento são dinamicamente alocados e desalocados durante a execução do programa.
  - **Stack** é usada para
    - Salvar e restaurar registradores usados em procedimentos.
    - Armazenar variáveis locais de funções ou procedimentos.
  - **Heap** armazena dados alocados dinamicamente durante a execução.
    - Aloca espaço no heap com **malloc()** e libera com **free()** in C.

- Segmentos reservados são usados pelos serviços do Sistema Operacional e rotinas de tratamento de interrupções.



I/O Mapping

Memory mapped I/O

- Dispositivos e memória compartilham um espaço de endereçamento.
- I/O parece como uma memória de leitura/escrita.
- Nenhum comando especial para I/O
  - ✓ Um grande conjunto de comandos de acesso a memória estão disponíveis.

Isolated I/O

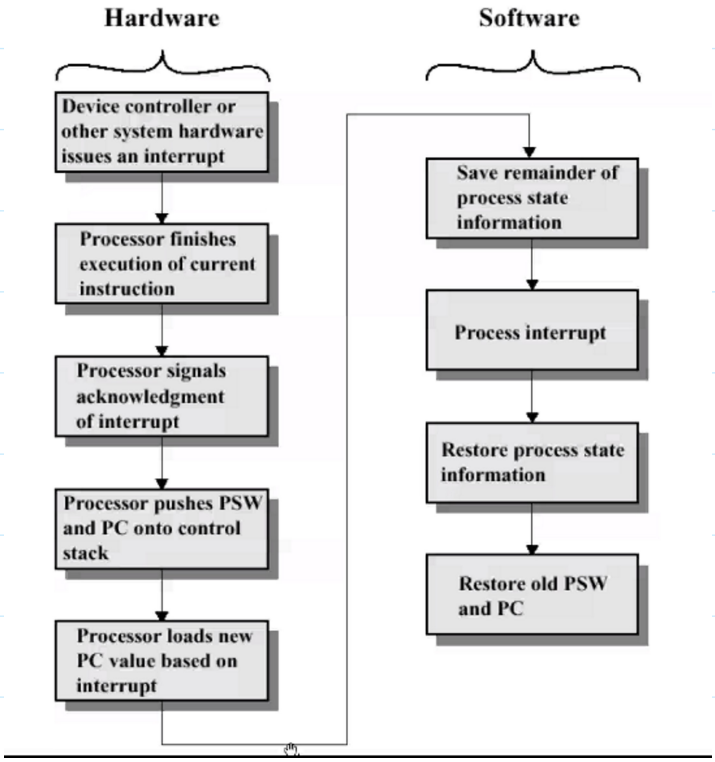
- Espaço de endereçamento separado
- Necessidade de linhas de seleção para I/O ou memória
- Comandos especiais para I/O
  - ✓ Conjunto limitado

Programmed I/O - Resumo

- Devido a diferença de velocidade entre a CPU e os dispositivos periféricos (ordens de magnitude), **programmed I/O** consome uma enorme quantidade do potencial de processamento da CPU.
  - Muito ineficiente
  - A CPU trabalha na velocidade do periférico.
- Vantagens
  - Implementação Simples
  - Requer pouquíssimo software especial ou hardware.

Interrupt Driven I/O

- Técnica para reduzir o tempo gasto nas operações de E/S.
  - A CPU fornece comandos de E/S para o módulo
  - A CPU continua com suas outras tarefas enquanto o módulo realiza a sua operação.
  - O módulo sinaliza à CPU quando a operação de E/S for finalizada (a interrupção)
  - A CPU responde a interrupção, executando uma rotina de serviço de interrupção (**isr**), logo após, continua a tarefa que estava executando durante este evento.
- Contorna os problemas de espera da CPU.
  - A CPU não precisa realizar uma verificação repetitiva do *status* do dispositivo.
  - O próprio módulo de E/S anuncia quando pronto (via interrupção).



Identificando o módulo que interrompe (1)

- Cada módulo possui uma linha de interrupção diferente.
  - PC
  - Limita o número de dispositivos.
- ***Software polling***
  - CPU consulta cada módulo em uma sequência circular
  - Lento.

Identificando o módulo que interrompe (2)

- ***Daisy Chain*** ou ***Hardware poll***
  - O recebimento da interrupção (*interrupt acknowledge*) é enviado cadeia abaixo.
  - O módulo responsável coloca o vetor de identificação no barramento.
  - A CPU usa o vetor para identificar a rotina de tratamento.
- ***Bus Mastering***
  - O módulo deve requerer o barramento antes que ele possa gerar a interrupção.
  - Por exemplo, PCI & SCSI

Direct Memory Access

- ***Interrupt driven*** e ***programmed I/O*** requerem intervenção ativa da CPU.
  - A taxa de transferência é limitada
  - A CPU fica aprisionada

- Para contornar estes problemas a técnica de **DMA** surge como uma solução.
- **DMA** Function:
  - Módulo adicional (*hardware*) no barramento.
  - O *DMA controller* liberta a CPU da lenta tarefa das operações de E/S.