ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES I

Aula 03 – Processadores

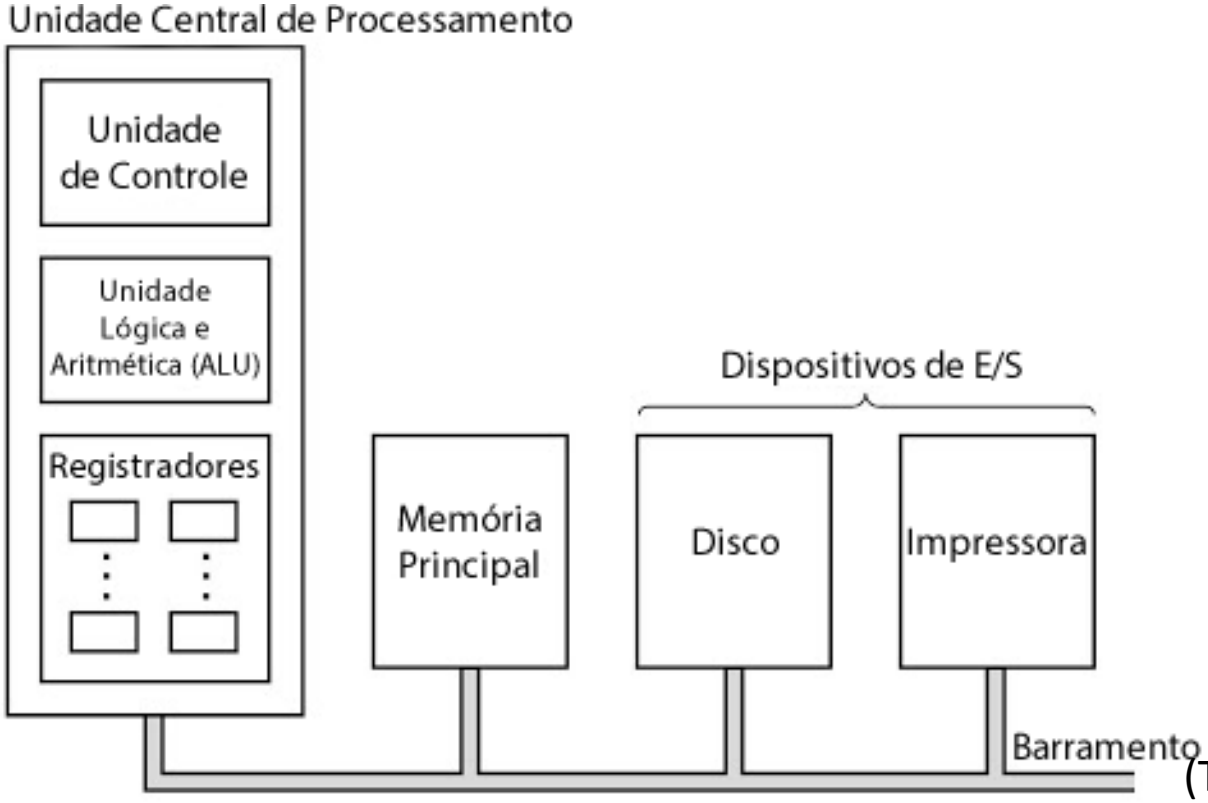
Prof. Dr. Guilherme Pina Cardim

guilhermecardim@fai.com.br

05 de março de 2020

Processadores

- Um computador digital consiste em um sistema interconectado de processadores, memórias e dispositivos de entrada e saída.



(Tanenbaum, 2007)

Processadores

- A CPU (*Central Processing Unit* – Unidade Central de Processamento) é considerada como o cérebro do computador;

- Possui a função de executar programas armazenados na memória principal realizando a busca, a análise e a execução de cada instrução armazenada;
  - Cada tarefa (busca, análise e execução de instruções) é realizada por unidades distintas.

Processadores

- As instruções são buscadas na memória principal pela **Unidade de Controle**;
- A **Unidade Lógica Aritmética** (ALU) é responsável por efetuar as operações, tais como adição e AND booleano;
- Na CPU também é encontrada uma pequena memória de alta velocidade utilizada para armazenar resultados temporários. Essa memória é composta por **registradores**.

Registradores

- **Contador de Programa** (PC – *Program Counter*): é considerado muitas vezes como o registrador mais importante em uma CPU. Ele indica qual é a próxima instrução a ser executada;

- **Registrador de Instrução** (IR – *Instruction Register*): armazena a instrução do programa que está sendo executada no momento.

Instruções

- As instruções de um programa são normalmente divididas em dois tipos:

Instruções registrador-memória

Permitem que palavras sejam buscadas em registradores para serem utilizadas como entradas da ALU

Instruções registrador-registrador

Permitem que palavras sejam buscadas em registradores para serem utilizadas como entradas da ALU

(Tanenbaum, 2007)

Instruções

- Uma instrução é executada na CPU em 7 etapas conhecidas como o ciclo **buscar-decodificar-executar**:

1. Trazer a próxima instrução da memória até o registrador;
2. Alterar o contador de programa para indicar a próxima instrução;
3. Determinar o tipo de instrução trazida;
4. Se a instrução usar uma palavra na memória, determinar onde essa palavra está;
5. Trazer a palavra para dentro do registrador da CPU, se necessário;
6. Executar a instrução;
7. Voltar a etapa 1 para iniciar a execução da instrução seguinte.

(Tanenbaum, 2007)

Instruções

```
public class Interp {
  static int PC;                          // contador de programa contém endereço da próxima instr
  static int AC;                          // o acumulador, um registrador para efetuar aritmética
  static int instr;                       // um registrador para conter a instrução corrente
  static int instr_type;                  // o tipo da instrução (opcode)
  static int data_loc;                    // o endereço dos dados, ou -1 se nenhum
  static int data;                        // contém o operando corrente
  static boolean run_bit = true;          // um bit que pode ser desligado para parar a máquina

  public static void interpret(int memória[ ], int starting_ address) {
    // Esse procedimento interpreta programas para uma máquina simples com instruções que têm
    // um operando da memória. A máquina tem um registrador AC (acumulador), usado para
    // aritmética. A instrução ADD soma um inteiro na memória do AC, por exemplo.
    // O interpretador continua funcionando até o bit de funcionamento ser desligado pela instrução HALT.
    // O estado de um processo que roda nessa máquina consiste em memória,
    // contador de programa, bit de funcionamento e AC. Os parâmetros de entrada consistem
    // na imagem da memória e no endereço inicial.

    PC = starting_address;
    while (run_bit) {
      instr = memory[PC];                 // busque a próxima instrução e armazena em instr
      PC = PC + 1;                        // incremente contador de programa
      instr_type = get_instr_type(instr); // determine tipo da instrução
      data_loc = find_data(instr, instr_type); // localize dados (-1 se nenhum)
      if (data_loc >= 0)                  // se data_loc é -1, não há nenhum operando
        data = memory[data_loc];          // busque os dados
      execute(instr_type, data);          // execute instrução
      }
    }
    private static int get_instr_type(int addr) { ... }
    private static int find_data(int instr, int type) { ... }
    private static void execute(int type, int data) { ... }
```

(Tanenbaum, 2007)

Interpretador

- O fato de ser possível escrever um programa que imite / simule a função de uma CPU, mostra que um programa pode ser executado por intermédio de outro programa;
- Um programa pode ser executado fazendo com que um segundo programa busque, examine e execute suas instruções;

Interpretador

Programa que busca, examina e executa as instruções de outro programa.

Interpretador

- Dada uma linguagem de máquina L para um novo processador desenvolvido, é possível:
  - Construir um processador de hardware para executar programas diretamente na linguagem L;
  - Desenvolver um interpretador para interpretar programas na linguagem L
    - Deve desenvolver uma máquina em hardware capaz de executar o interpretador;
  - Construções híbridas também são possíveis.

Evolução do Hardware x Software

- A evolução do hardware dos computadores aconteceu de forma rápida;

- Ao mesmo tempo, as linguagens de programação se tornavam de alto nível possuindo funções mais complexas que facilitavam a programação;

- Resultou no seguinte questionamento:
  - É mais eficiente implementar um algoritmo complexo em software ou realizar a implementação em hardware do mesmo?

Evolução do Hardware x Software

- É mais eficiente implementar um algoritmo complexo em software ou realizar a implementação em hardware do mesmo?

- Uma implementação realizada diretamente em hardware é normalmente mais rápida;

- Programas são executados mais rápidos se tarefas complexas são implementadas diretamente em hardware.

Arquitetura CISC

CISC

Computador com conjunto de instruções complexo (*Complex Instruction Set Computer*).

- Os processadores se tornaram cada vez mais complexos:
  - Grande número de instruções;
  - Instruções complexas;
  - Inúmeros modos de endereçamento;
  - Implementação em hardware de comandos de linguagens de alto nível;
  - Instruções com tamanhos e tempos de execução variados;
  - Circuitos Complexos.

Arquitetura CISC

- Foi desenvolvida na tentativa de diminuir a diferença entre o que as máquina podiam executar com o que as linguagens de programação de alto nível demandavam;

- As instruções complexas nas arquiteturas CISC só são possíveis graças ao uso do interpretador.

Arquitetura RISC

- A interpretação e execução das instruções mais complexas de um computador CISC afetam o desempenho?

- No final da década de 1970, alguns pesquisadores acreditavam que as instruções complexas afetavam o desempenho:
  - Para eles, era melhor possuir uma máquina mais simples e "dez" vezes mais rápida, por não ser interpretada, mesmo se essa máquina necessitasse de "cinco" instruções para executar o que era feito por "uma" instrução na máquina CISC.

Arquitetura RISC

RISC

Computador com conjunto de instruções reduzido (*Reduced Instruction Set Computer*).

- Instruções mais simples;
  - Não necessariamente menos instruções;
- Instruções com formato uniforme;
  - Tamanho fixo facilita a decodificação das instruções;
- Poucos modos de endereçamento;
- Maior número de registradores.

Arquitetura RISC

- Com instruções mais simples, as arquiteturas RISC se distanciam das linguagens de programação de alto nível;
- Implicam em mais trabalho para os compiladores durante a tradução e otimização do código;
- O termo reduzido não se refere ao número de instruções, porém tal fato acaba sendo consequência em diversos processadores RISC.

Característica	CISC			RISC	
Modelo	IBM 370/168	VAX 11/780	Intel 80486	SPARC	MIPS R4000
Ano	1973	1978	1989	1987	1991
Instruções	208	303	235	69	94
Registradores	16	16	8	40-520	32

(Passos, 2017)

CISC vs RISC

Característica	CISC	RISC
Arquitetura	Registrador-Memória	Registrador-Registrador
Tipos de Dados	Muita variedade	Pouca variedade
Formato de Instruções	Instruções com muitos endereços	Instruções com poucos endereços
Modo de Endereçamento	Muita variedade	Pouca variedade
Acesso aos Dados	Via memória	Via registradores

Trabalho – CISC vs RISC

- Realizar uma pesquisa sobre os dois tipos de arquiteturas, CISC e RISC, contendo:
  - Dados históricos;
  - Principais características;
  - Principais diferenças;
  - Exemplos atuais equipamentos com ambas arquiteturas;
  - Suas conclusões.
- Fazer um relatório (normas ABNT);
- Entrega: 26/03/2020 ou 02/04/2020
- Impresso ou por email: guilhermecardim@fai.com.br

Material Referência

- CIPOLI, Pedro. **O que é a Lei de Moore?** Canaltech, ~2014.
- FERNANDES, Carlos. **Aula 01 – Arquitetura de Computadores**. IFPE, 2020.
- GONÇALVES, Marcelo M. **Arquitetura de computadores**. CEFETPR, 2006.
- PARHAMI, Behrooz. **Arquitetura de Computadores: de microprocessadores a supercomputadores**. São Paulo: McGraw-Hill, 2007.
- PASSOS, Diego. **Aula 26: Arquiteturas RISC vs. CISC**. UFF: 2017.
- TANENBAUM, Andrew S. **Organização estruturada de computadores**. 5.ed. São Paulo: Pearson Prentice Hall, 2007.
- TEIXEIRA, Tony; TORCATO, Francisco. **A evolução dos computadores**. CFPIC, 2013.