Unidade II

3 ORGANIZAÇÃO DO COMPUTADOR

3.1 Processadores

O processador (figura 30) é o 'cérebro' do computador e é também conhecido como CPU (*Central Processing Unit*), ou Unidade Central de Processamento. Utiliza a linguagem de máquina, binário 0 e 1, nos seus cálculos.

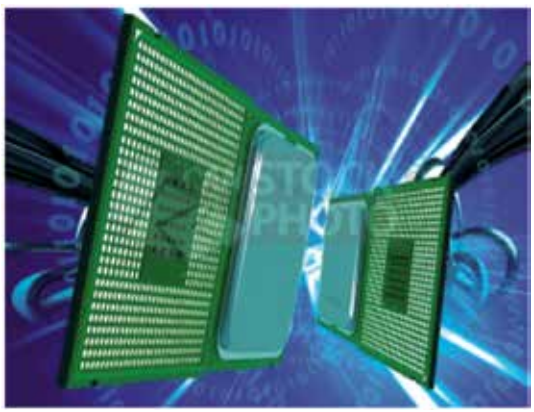


Figura 30 - Modelo de processador de microcomputador [Processador]

Basicamente, o processador é o responsável por executar os programas que ficam carregados na memória principal. Ele executa as instruções ou tarefas dos programas uma a uma.

As principais funções do processador, de acordo com Stallings, são:

1) Busca de instrução: lê uma instrução da memória;

2) Interpretação da instrução: decodificação da instrução;

3) Busca de dados: uma instrução pode necessitar de diferentes dados da memória ou de E/S;

4) Processamento de dados: execução aritmética ou lógica sobre os dados;

5) Escrita de dados: os resultados podem necessitar escrever dados na memória ou em E/S.

Para executar as tarefas, o processador utiliza partes distintas, como Unidade de Controle, Unidade Lógica Aritmética (ALU) (figura 31) e registradores.

Lembrete

ALU é abreviação em inglês de *Arithmetic Logic Unit*, também conhecida como ULA (Unidade Lógica Aritmética).

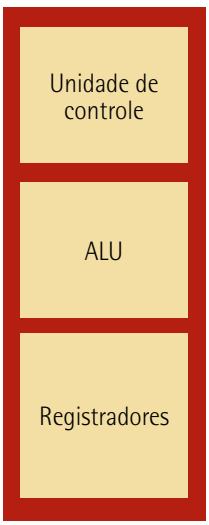


Figura 31 - ALU / ULA

Fonte: TANENBAUM, 2007a.

- A unidade de controle busca as informações na memória principal.
- A unidade lógica e aritmética realiza cálculos e comparações entre valores.
- Os registradores compõem uma memória de alta velocidade (interna à CPU) utilizada para armazenar resultados temporários e para o controle do fluxo de informações.

Os registradores no processador desempenham dois papéis:

- Registradores visíveis ao usuário: possibilitam que o programador de linguagem de máquina minimize as referências à memória.
- Registradores de controle e estado: usados pela unidade de controle para controlar a operação do processador.

Observação

Existem diferentes tipos de registradores, isto é, com funções distintas; entretanto, normalmente, todos os registradores têm o mesmo tamanho.

Os registradores mais conhecidos são:

- Contador de Programa: que indica a próxima instrução a ser executada.
- Registrador de Instrução: que contém a instrução que está sendo executada, ou seja, a informação captada da memória cache, principal, E/S, etc.
- Registradores de dados: são utilizados apenas para armazenar dados, não sendo empregados no cálculo de endereços.
- Registradores de endereços: recebem o endereço de um dado objeto. Esse registrador oferece aos programadores a possibilidade da utilização de ponteiros.

Lembrete

Ponteiros são variáveis contendo um endereço no programa.

Registrador (figura 32) é o elemento superior no nível de hierarquia de memória, por possuir a maior velocidade de transferência dentro do sistema, estando acima da memória cache, da memória principal e da memória secundária. Estudaremos essas estruturas mais adiante. O registrador possui a menor capacidade de armazenamento e o maior custo, é o tipo mais caro de memória.

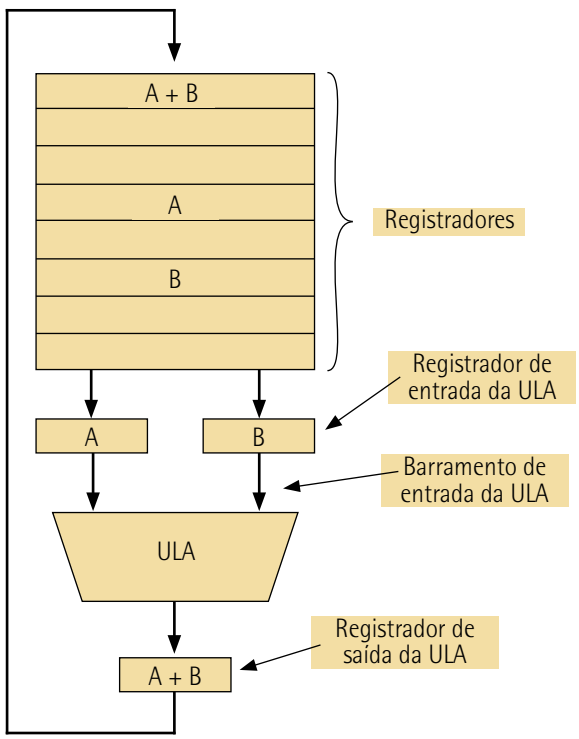


Figura 32 - Registradores

Fonte: TANENBAUM, 2007a.

Nessa figura, temos a representação gráfica dos registradores de entrada que recebem os valores hipotéticos A e B que serão somados na ALU e enviados ao registrador de saída. Os resultados são armazenados em registradores para novos cálculos e posteriormente podem ser armazenados (escritos) em memória. A comunicação entre os registradores e a ALU acontecem por meio de barramentos internos da CPU. A seta à esquerda indica o caminho dos dados; e, quanto mais rápido for esse processo, maior o desempenho do processador e do sistema computacional como um todo.

Resumidamente, uma operação normal do caminho de dados consiste em selecionar o conteúdo de um ou dois registradores, submetê-lo à ALU e movimentar o resultado para outro registrador.

3.1.1 Instruções de máquina

As instruções de máquina são os pontos centrais na arquitetura de um processador. Elas representam um conjunto de instruções que determinam a operação do processador, além de influenciar as operações realizadas pela unidade lógica e aritmética e por registradores.

Os principais tipos de instruções são:

- Instruções aritméticas e lógicas: realizam operações aritméticas sobre números inteiros (adição e subtração) e operações lógicas bit-a-bit (AND, OR).
- Instruções de movimentação de dados: transferem dados entre registradores ou entre registradores e a memória principal.
- Instruções de transferência de controle: transferem a execução para uma determinada instrução dentro do código do programa.

Várias arquiteturas de processadores oferecem outras categorias de instrução, voltadas para operações especializadas: instruções de ponto flutuante; instruções decimais; instrução de manipulação de bits; instrução de manipulação de strings.

Os elementos de uma instrução são:

- Código da Operação (Opcode): especifica a operação a ser realizada.
- Referência ao operando fonte: operandos que são entradas para a operação.
- Referência ao operando destino: operandos que expressam o resultado.
- Referência à próxima instrução: informa onde se deve buscar a nova instrução.

A localização dos operandos pode se dar das seguintes formas:

- Memória principal ou virtual: descoberto a partir do endereço de memória designado na instrução.

- Registrador do processador: descoberto a partir da referência ao registrador designado na instrução.
- Imediato: o valor do operando está descrito no campo da instrução.
- Dispositivo de E/S: instrução especifica o módulo e o dispositivo de E/S.

A instrução é representada por uma sequência de bits (números binários 0 e 1). Ela é dividida em campos, correspondentes aos elementos constituintes da instrução, sendo uma prática comum a utilização de representações simbólicas das instruções.

O formato de instruções pode seguir as seguintes maneiras:

OPCODE

OPCODE + OPERANDO

OPCODE + OPERANDO1 + OPERANDO 2

Os opcodes são representados por abreviações, chamadas de mnemônicos, que indicam a operação.

ADD: Adiciona

SUB: Subtrai

MUL: Multiplica

DIV: Divide

LOAD: Carrega dados da memória

STOR: Armazena dados na memória

A seguir é possível ver o exemplo de uma instrução:

ADD R, X

Opcode: ADD

Endereços dos Operandos: R e X

Esta instrução pode significar somar o valor contido no local de dados X com o conteúdo do registrador R.

Nesta instrução a operação é executado sobre o conteúdo de um local e não sobre o endereço.

Os principais tipos de operandos são: endereços; números; caracteres; dados lógicos.

Os tipos mais gerais de opcodes são: transferência de dados; aritmética; lógica; conversão; E/S; transferência de controle.

Para a operação de transferência de dados é necessário:

- Local dos operandos de origem e destino.
- Extensão dos dados a serem transferidos.
- Modo de endereçamento para cada operando.

Por exemplo:

MOVE – transfere palavra da origem para o destino

SET – Transfere 1s para o destino

As operações aritméticas da CPU envolvem adição, subtração, multiplicação e divisão. Outros exemplos são:

ABSOLUTE – apanha o valor absoluto do operando

NEGATE – inverte o sinal do operando

INCREMENT – soma 1 ao operando

DECREMENT – subtrai 1 do operando

Operações lógicas utilizam a manipulação de bits (números binário 0 e 1). Exemplos:

OR – Operação "OU"

AND – Operação "E"

XOR – Operação "OU" exclusivo

NOT – Inversão de BIT

As arquiteturas de instruções podem ser divididas em:

- Arquitetura memória-memória: usam três operandos e todos podem estar na memória.

- Arquitetura registrador-memória: usam dois operandos, sendo que apenas um deles pode residir na memória.

- Arquitetura registrador-registrador: usam três operandos, todos em registradores.

De acordo com Tanenbaum, a CPU executa as instruções através das seguintes etapas:

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 da instrução;

4) Se a instrução necessitar de uma palavra da memória, determinar onde essa palavra está.

5) Trazer a palavra para dentro de um registrador da CPU, se necessário.

6) Executar a instrução.

7) Voltar à etapa 1 para iniciar a execução da instrução seguinte.

Essa sequência é chamada ciclo **buscar-decodificar-executar** e é fundamental para a operação de todos os computadores.

Lembrete

Que o programa é armazenado na memória na forma de instruções que o processador consiga interpretar, ou seja, o programa quando escrito em linguagem de alto nível, sofre vários níveis de tradução antes de ser executado diretamente pelo processador.

3.1.2 *Clock* interno

O ***Clock*** do processador ou interno é um *chip* que emite sinais elétricos através de um cristal de quartzo e realiza a sincronização entre as instruções a serem processadas. Quando as tarefas são executadas, ocorre o pulso de *Clock*. As oscilações são mensuradas em *hertz* (Hz), unidade padrão de medidas de frequência, que indica o número de ciclos que ocorre em certo tempo, no caso, segundos. Desse modo, se um processador trabalha com 500 MHz, por exemplo, significa que é capaz de lidar com 500 milhões de operações de ciclos de *Clock* por segundo. Quanto mais *Clocks*, mais rápido o processador; entretanto, existem limites.

Lembrete

Existe outro tipo de *Clock*, externo à CPU.

Saiba mais

<http://www.dca.ufrn.br/~pablo/FTP/arq_de_comp/apostilha/capitulo2.pdf>

3.2 RISC *versus* CISC

Com o avanço da tecnologia, no final da década de 1970 já existia uma deficiência entre o que as máquinas podiam fazer e o que as linguagens de alto nível exigiam.

A ideia que prevalecia era desenvolver processadores cada vez mais poderosos no intuito de diminuir ou extinguir a lacuna existente.

Entretanto, alguns profissionais resolveram nadar contra a corrente e passaram a pensar no desenvolvimento de máquinas mais simples. Em 1980, um grupo em Berkeley, liderado por David Patterson e Carlo Séquin, criou *chips* de CPU e batizou com o termo RISC (*Reduced Instruction Set Computer*), ou computador com conjunto de instruções reduzidas, em comparação com CISC (*Complex Instruction Set Computer*), ou computador com conjunto de instruções complexas.

Eles acreditavam que, embora fossem necessárias mais instruções para o RISC em comparação com o CISC, se as instruções do RISC fossem mais rápidas, este teria um melhor desempenho.

A figura 32 representa uma estrutura de processamento com tecnologia RISC.

Apesar da vantagem da tecnologia RISC, a tecnologia CISC não deixou de existir; primeiro, porque existiam bilhões de dólares investidos em *softwares* para a arquitetura CISC (compatibilidade), e segundo, porque a Intel, a partir do 486, desenvolveu uma tecnologia híbrida, ou seja, um núcleo com RISC que executa as instruções mais simples (as mais comuns) e CISC para as mais complicadas. O resultado é que instruções comuns são rápidas e as complexas, lentas.

Quase três décadas após o lançamento da tecnologia RISC, aprendemos que novas tecnologias de impacto podem surgir a todo momento, tornando-se difícil e às vezes inviável sua implementação imediata; desse modo, nada mais coerente do que delimitar princípios de projeto, visando minimizar esses riscos.

Os princípios de projeto também visam maximizar o desempenho de sistemas computacionais. A seguir, mencionamos os mais relevantes.

3.3 Princípios de projeto para computadores modernos

3.3.1 Todas as instruções são executadas diretamente pelo *hardware*

Atualmente a recomendação é que as instruções correspondam diretamente a elementos de *hardware*, pois antigamente era comum em estruturas computacionais adicionar uma camada de *software* entre o *hardware* e os demais programas, providenciando um novo nível de interpretação, o microcódigo. O microcódigo oferece a possibilidade de incluir ou alterar instruções sem alterar o *hardware*; mas, apesar de oferecer mais flexibilidade, não é considerado uma boa prática, pois o nível adicional de interpretação acarreta perda de desempenho, que não compensa às novas instruções que possam ser adicionadas.

3.3.2 Maximize a taxa de execução de instruções

Uma das formas de maximizar o desempenho do computador é aumentar a taxa de execuções de instruções por segundo, através do aumento da velocidade de relógio (*Clock*), ou através do paralelismo, como veremos adiante.

3.3.3 As instruções devem ser fáceis de decodificar

Quanto mais simples ou menos formatos alternativos tiverem as opções de execução de uma instrução, melhor; pois, se a instrução tiver poucos formatos, o tempo de identificação da instrução será reduzido logo, e o tempo de processamento será otimizado.

3.3.4 Somente *Load* e *Store* devem referenciar a memória

O acesso à memória principal é uma operação mais demorada. Por esse motivo, é recomendado que apenas as instruções *Load* (leitura) e *Store* (gravação) tenham acesso à memória.

3.3.5 Providencie muitos registradores

Quando o processador não tem um registrador disponível para armazenar um valor resultante, ele transfere esse valor para a memória principal. Sabemos que a transferência de dados entre o processador e a memória principal é um processo mais lento do que a movimentação de dados dentro do processador; portanto, quanto mais registradores possíveis, melhor.

4 PARALELISMO (*PIPELINE*)

4.1 Paralelismo no nível de instrução

Conforme mencionamos, um dos caminhos para se maximizar a *performance* do computador é aumentar a taxa de execuções de instruções por segundo, aumentando a velocidade de relógio do computador. Outra forma é fazer com que o processador realize atividades em paralelo.

O *Pipeline* ou Paralelismo é a técnica de dividir a execução da instrução em várias partes, e cada uma será manipulada por uma parte específica do processador. A figura 33, extraída da obra de Tanenbaum, exemplifica a questão.

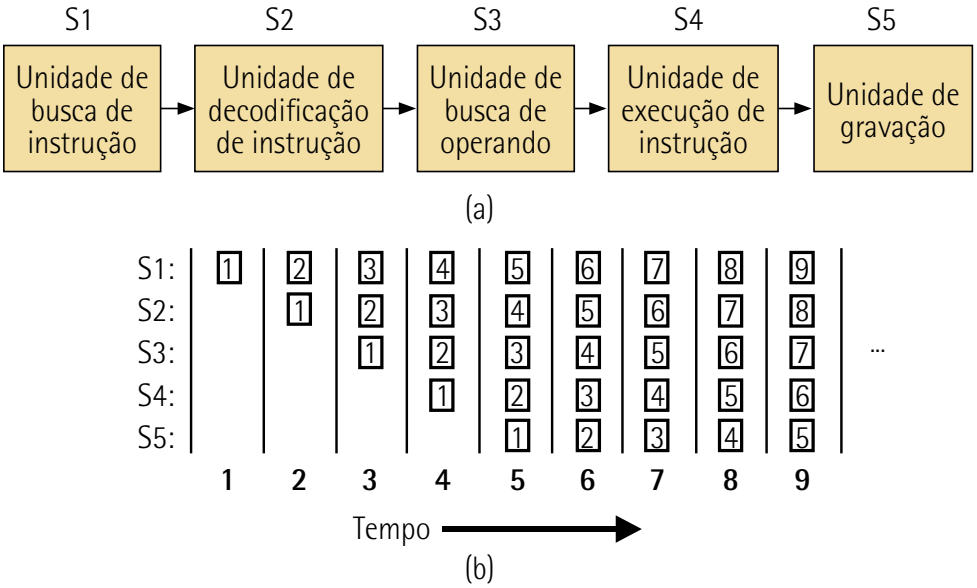


Figura 33 - Modelo de Pipeline

Fonte: TANENBAUM, 2007a.

Nesse exemplo, temos cinco estágios ou unidades S1 (busca de instrução), S2 (decodificação de instrução), S3 (busca de operando), S4 (execução) e S5 (gravação). Nove ciclos de relógio (*time*) e nove instruções (quadradinhos).

No primeiro ciclo, o processador busca a instrução 1, na unidade de busca de instrução; no segundo ciclo, enquanto realiza a decodificação da instrução 1, na unidade de decodificação, busca a instrução 2 na unidade de busca de instrução, e assim sucessivamente, como uma linha de montagem.

A questão é que nem todas as instruções demandam o mesmo tempo para serem executadas, pois nem todas as unidades trabalham na mesma velocidade; a unidade de execução, por exemplo, com instruções do tipo *LOAD* e *STORE*, que acessam a memória principal, são mais demoradas. A solução são as arquiteturas superescalares, como veremos a seguir.

4.2 Arquiteturas superescalares

A Idea da arquitetura superescalar é manter na unidade de execução mais de uma ALU, uma unidade para *LOAD*, uma para *STORE* e uma de ponto flutuante ex.1,35789. Desse modo, ocorre uma compensação, pois as demais unidades são mais rápidas. A figura 34 exemplifica a questão.

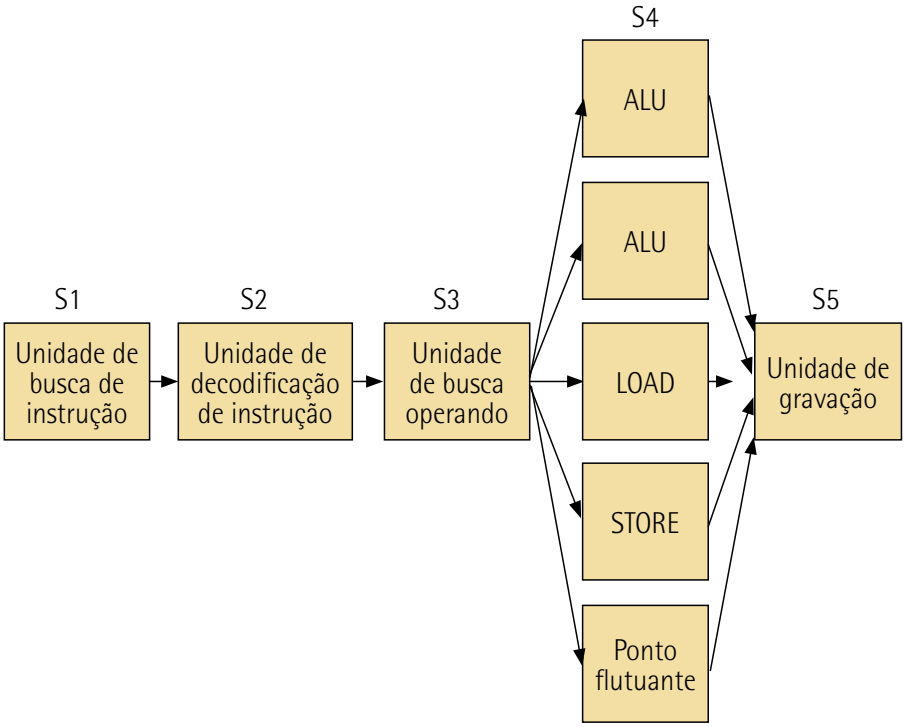


Figura 34 - Arquitetura superescalar

Fonte: TANENBAUM, 2007a.

Lembrete

Esse tipo de paralelismo é implementado dentro do processador, ou seja, não depende de mais de um processador. Nesse tipo de *design*, deve ser tomado o cuidado para que as instruções sejam executadas na ordem correta.

Saiba mais

<http://rossano.pro.br/fatec/cursos/sistcomp/apostilas/organizacao-computadores-processadores.pdf>

4.3 Paralelismo no nível do processador

Por mais que consigamos aumentar o desempenho da CPU, maximizando as taxas de execução (*Clock*) e implementando paralelismo no nível de instrução, as limitações são uma constante, pois aumentar taxas de execução não pode ultrapassar a velocidade da luz. Outro problema é a

dissipação do calor, cada vez maior em relação às execuções. O paralelismo no nível de instrução, como vimos, ajuda, mas não é suficiente, pois exige uma grande dependência entre as instruções. A solução é implementar paralelismo no nível do processador, ou seja, projetar computadores com várias CPUs. Vejamos a seguir.

4.3.1 Computadores matriciais

Um computador matricial é aquele com grande número de processadores que executam o mesmo conjunto de instruções em conjuntos diferentes de dados. Esse tipo de arranjo encontra uso em aplicações científicas, em que um grande conjunto de dados deve ser submetido a um mesmo conjunto de fórmulas. Ele não é comum em computadores comerciais, mas muitos supercomputadores (especialmente os fabricados pela *Cray Computing*) usam um tipo especial de arranjo desse tipo, chamado processador vetorial.

4.4 Multiprocessadores

Uma forma simples de aumentar a capacidade de processamento de um computador é adicionar outro processador. Essa é a ideia do multiprocessador, mas isso não é tão simples. Sempre que fazemos algum tipo de paralelismo, deve haver alguma figura fazendo a arbitragem das tarefas que serão executadas por cada processador. Além disso, no multiprocessador, todas as CPUs compartilham uma mesma quantidade de memória.

O acesso a essa memória deve ser controlado também, para que um processador não tente utilizar um espaço que esteja sendo usado por outro. Normalmente essa função é desempenhada pelo sistema operacional.

A maioria dos sistemas operacionais de mercado atualmente tem alguma capacidade de multiprocessamento.

4.5 Multicomputadores

A diferença entre o multicomputador e o multiprocessador é que no multicomputador cada CPU tem uma memória local que só é acessada por aquele processador.

No multicomputador, pode ou não haver uma quantidade de memória compartilhada entre os processadores. Um dos problemas desses tipos de arranjo é conectar todos os processadores à memória. Por esse motivo, muitos projetistas abandonam a ideia da memória compartilhada e criam computadores nos quais há apenas processadores com memórias privadas. Esse tipo de arranjo normalmente é chamado de fracamente acoplado, em contraste com o arranjo fortemente acoplado, que é o multiprocessador.

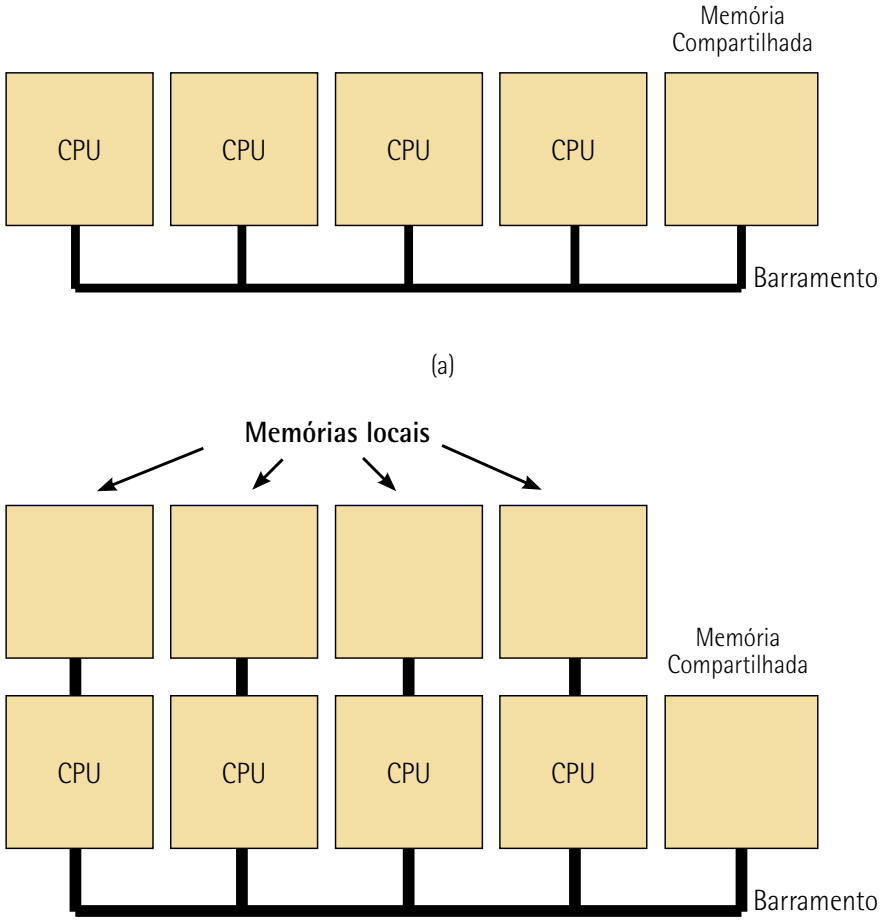


Figura 35 - **(a)** Multiprocessador de barramento único. (b) Multicomputador com memórias locais

Fonte: TANENBAUM, 2007a.

Resumo

Nesta unidade, abordamos temas como a organização do computador, o que são processadores, *Clock*, Arquitetura RISC *versus* CISC, entre outros.

Conhecemos essas arquiteturas e as superescalares, vimos a arquitetura do multiprocessador e de multicomputadores. Entendemos um pouco mais sobre discos magnéticos e periféricos.

Exercícios

Questão 1

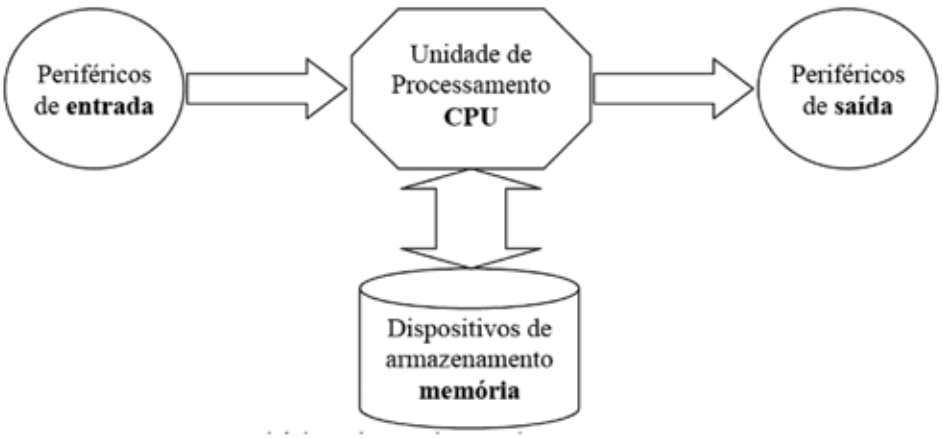


Figura – Organização dos sistemas de computação

Em relação à Arquitetura dos Computadores, analise as afirmativas:

I. A função da CPU é executar programas armazenados no disco rígido, buscando suas instruções, examinando-as e executando-as, uma após a outra.

II. A CPU possui uma memória interna, de alta velocidade, usada para armazenar resultados temporários e certas informações de controle.

III. Um computador é uma máquina composta de um conjunto de partes eletrônicas e eletromecânicas, com capacidade de coletar, armazenar e manipular dados, além de fornecer informações.

IV. O *hardware* do computador é tudo aquilo que o compõe fisicamente. Constituí-se em *hardware* o próprio sistema operacional do computador e outros programas do computador.

V. O computador é uma máquina programável capaz de processar informações com grande rapidez.

VI. Os computadores digitais são totalmente binários, isto é, trabalham apenas com dois valores, tornando assim simples o emprego da lógica booleana (Verdadeiro/Falso, Aberto/Fechado, ...) tanto na construção de componentes quanto como base para a escrita de programas.

É correto o que consta em:

A. I, II, IV e VI, apenas.

B. II, III, IV e V, apenas.

C. I, III, IV e VI, apenas.

D. II, III, V e VI, apenas.

E. I, II, III, IV, V e VI.

Resposta correta: Alternativa D.

Análise das afirmativas

I) Afirmativa incorreta.

Justificativa: A CPU executa programas armazenados na memória RAM.

II) Afirmativa correta.

Justificativa: A CPU utiliza a memória cache para armazenar resultados temporários. A memória cache é mais rápida que a memória RAM.

III) Afirmativa correta.

Justificativa: Os computadores executam quatro funções básicas: Entrada, Processamento, Armazenamento/recuperação de dados e Saída.

IV) Afirmativa incorreta.

Justificativa: Sistema operacional é um *software*.

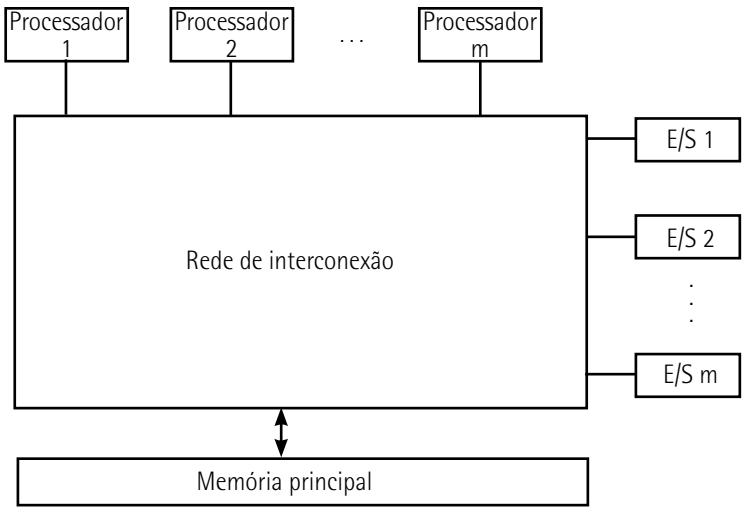
V) Afirmativa correta.

Justificativa: O computador sozinho é um juntado de dispositivos eletrônicos sem nenhuma função. Pare que um computador trabalhe é necessário à inserção de informações (Entrada), seguindo as instruções fornecidas pelos programas, o computador processa os dados originários da entrada. A saída costuma ser armazenada para posterior utilização.

VI) Afirmativa correta.

Justificativa: os computadores seguem a lógica clássica que é constituída por 0 e 1.

Questão 2. (IFTO adaptada) Considere o diagrama de blocos geral de um multiprocessador fortemente acoplado mostrado a seguir:



Arquitetura e organização de computadores

Analise a veracidade das afirmações abaixo e indique a alternativa correta:

I) Neste esquema, cada processador é autocontido, possuindo unidade de controle, ULA, registradores e memória cache.

II) Mais de um processador pode participar da execução de uma aplicação.

III) A memória pode ser organizada de modo a permitir acessos simultâneos a blocos separados de memória por diferentes processadores.

IV) Cada processador pode possuir recursos próprios, não compartilhados, de memória e dispositivos de E/S.

A. Os itens I e II são falsos.

B. O item III é falso.

C. O item IV é falso.

D. Todos os itens são verdadeiros.

E. Todos os itens são falsos.

Resolução desta questão na Plataforma.