HARDWARE E ORGANIZAÇÃO DE COMPUTADORES

SUMÁRIO

Capítulo I: Fundamentos iniciais

1 - Conceito de computador

1.1 O que é um computador?

Um computador digital é uma máquina que pode realizar tarefas, atender pedidos e resolver problemas para as pessoas, através da execução de instruções que lhe são dadas. Uma sequência de instruções descrevendo como se deve fazer qualquer uma dessas ações é chamada de programa ou algoritmo. O computador manipula diversos tipos de informações, como fotos, vídeos, arquivos de texto, músicas, desenhos, planilhas, e muito mais.

1.2 A evolução dos computadores

O computador, assim como outras tecnologias, passou por um processo evolutivo ao longo do tempo. Para entender como chegamos ao nível de desenvolvimento dos dias de hoje, precisamos voltar no tempo.

Uma das premissas da criação do computador é a de execução de tarefas, como por exemplo, contar números. Se nós analisarmos a palavra “computar” da sua forma mais básica, entendemos que é a realização de uma contagem ou operação matemática. A primeira ferramenta de computação foi o ábaco que permitia a realização de contagem, operações como adição e subtração e armazenamento de valores. A partir do ábaco surgiram diversas máquinas com premissas de automatização, computação e execução de tarefas. No século XVII, Blaise Pascal inventou a Pascalina, uma das primeiras calculadoras mecânicas, seguida pela máquina de Leibniz, uma calculadora mecânica que permitia também a multiplicação e divisão.

No início do século XIX, Charles Babbage concebeu a "Máquina Analítica", um conceito revolucionário que incorporava ideias de programação por cartões perfurados, semelhante aos usados em teares mecânicos da época. Ada Lovelace, trabalhando com Babbage, elaborou os primeiros algoritmos para essa máquina, sendo reconhecida como a primeira programadora da história.

Essa fase marcou o nascimento dos conceitos fundamentais da computação moderna, mesmo que a tecnologia da época ainda não permitisse a construção completa dessas máquinas visionárias. Com o avanço da eletricidade e da engenharia no século XX, novas invenções começaram a tornar realidade aquilo que antes era apenas conceitual. Durante a Segunda Guerra Mundial, a necessidade de cálculos rápidos e precisos impulsionou o desenvolvimento de computadores eletromecânicos e eletrônicos, como o Z3 de Konrad

Zuse e o ENIAC, considerado o primeiro computador eletrônico de propósito geral. Esses marcos deram início à chamada Primeira Geração de Computadores, caracterizada pelo uso de válvulas eletrônicas, grandes dimensões físicas e alto consumo de energia. A partir daí, a evolução dos computadores se acelerou com o surgimento dos transistores, circuitos integrados e microprocessadores, tornando os computadores cada vez menores, mais rápidos, acessíveis e poderosos — culminando nos dispositivos modernos que utilizamos hoje.

1.3 Tipos de computadores

Os computadores que são utilizados pelo público comum são, na verdade, chamados de microcomputadores. Entre esses microcomputadores, podemos dividi-los em “desktop” (computadores de mesa) e “laptop” (computadores portáteis, também conhecidos como notebooks).

O desktop possui algumas variações como, por exemplo, o desktop “all in one” que possui o gabinete integrado ao próprio monitor. Os laptops também possuem diversas variações, como netbooks (um computador pequeno, pouco veloz, e criado basicamente para acesso à internet), ultrabooks (um computador com dimensões semelhantes às de um notebook, ultrafinos e baterias que duram por mais tempo), e o próprio tablet (um computador pequeno com tecnologia touchscreen).

2 - Linguagem do computador (Sistema binário)

2.1 Linguagem do computador vs. Linguagem humana

Existe uma grande diferença entre o que é conveniente para as pessoas e o que é conveniente para computadores. As pessoas querem fazer X, mas os computadores só podem fazer Y, o que dá origem a um problema.

Nós seres humanos enxergamos um tipo de dado, seja ele uma foto, vídeo ou arquivo de texto, enquanto que o computador enxerga apenas os números “0” e “1”. Os circuitos eletrônicos de cada computador podem reconhecer e executar diretamente um conjunto limitado de instruções simples, para o qual todos os programas devem ser convertidos antes que possam ser executados. Essas instruções básicas raramente são muito mais complicadas do que:

- Some dois números;
- Verifique se um número é zero;
- Copie dados de uma parte da memória do computador para outra.

Essa linguagem é chamada de linguagem de máquina ou binária, e é a linguagem de mais baixo nível, enquanto que a linguagem humana é a de mais alto nível.

2.2 Linguagem de Máquina e Código Binário

Existem unidades de grandeza que são imprescindíveis para o entendimento da linguagem de máquina. A primeira delas é o bit (dígito binário) que representa um único “0” ou um único “1”. Podemos compreender também que um bit representa um único pulso elétrico (o bit 1), ou a ausência desse pulso (o bit 0).

Digamos que exista um projeto de sistema de cadastro para estudantes, onde uma das perguntas seja “É a primeira matrícula deste aluno?”. Para responder isso, podemos usar a opção “SIM” e atribuir a ela o valor do bit “1”, representando a presença do impulso elétrico para informar ao computador que é a primeira vez que este aluno se matricula nesta instituição.

Para a opção “NÃO” atribui-se o valor do bit “0”, representando portanto a ausência do impulso elétrico, o que informa ao computador que esta não é a primeira vez que este aluno realiza a matrícula na escola.

A segunda unidade é o byte (termo binário) que representa um conjunto formado necessariamente por oito bits. Enquanto que um bit pode ser representado por apenas 2 opções, um byte pode ter 256 variações, pois se cada bit deste byte tem 2 opções, então temos 2^8, que é igual a 256.

Vamos considerar que nesse sistema de cadastro de estudantes existe uma lacuna a ser preenchida com o nome do aluno. Como podemos representar um nome como “Alexandre Borges Carvalho” para um computador através de impulsos elétricos se os bits tem apenas duas opções? Através dos bytes. Podemos estabelecer que a letra A é um byte cujo valor é “00001010”, que o valor do byte da letra L é “10000010” e assim por diante. Esse aumento na possibilidade de representações de informações é o que possibilita que os computadores realizem tarefas extremamente complexas.

Numa forma de padronizar como os caracteres podem ser representados pela linguagem binária, criou-se o “ASCII” (American Standard Code for Information Interchange) que em português significa “Código Padrão Americano para Intercâmbio de Informação”. Esse código contém letras, algarismos e sinais de pontuação e de controle, através de um sinal codificado em forma de código binário (cadeias de bits formada por vários 0 e 1). Com isso fica determinado que a letra “A” é representada pelo byte “0100 0001”.

Além do ASCII existem outros códigos de representação, como o “Unicode” por exemplo que consegue representar ainda mais caracteres que o ASCII já que um caractere no ASCII é representado por apenas um byte, o que permite apenas 256 possibilidades, enquanto que o Unicode representa um caractere através de 2 bytes, permitindo 65.536 possibilidades.

Por utilizar mais bytes para representações de caracteres, o Unicode exige uma capacidade maior de processamento. Se quiséssemos representar a palavra “casa” através do ASCII gastaríamos apenas 4 bytes, enquanto que através do Unicode gastaríamos 8 bytes.

Capítulo II: Hardware

3 - Os componentes básicos do hardware

A arquitetura básica de qualquer computador completo, seja um PC, um Macintosh ou um computador de grande porte, é formada por cinco componentes básicos de hardware (toda a parte física do computador):

- Placa-mãe
- Processador (CPU);
- Memória Principal;
- Memória Secundária (HD/SSD);
- Periféricos de entrada e saída;

Aliados a esses componentes também temos:

- Placa de vídeo
- Fonte de alimentação
- Placa de rede
- Gabinete

É importante ressaltar que para que um computador possa funcionar ele só precisa apenas, na sua forma essencial, de uma placa-mãe, CPU e Memória Principal (Memória RAM e Memória ROM).

4 - Placa-mãe

4.1 Conceito de placa-mãe

Para analisar um hardware, devemos começar pelo princípio, que é a placa-mãe. Conhecida também como motherboard, mainboard ou MOBO, a placa-mãe é o principal componente de um computador. É nessa placa de circuitos que todos os outros componentes são conectados, e sua função é gerenciar a transmissão de dados entre os diversos dispositivos de hardware do sistema.

4.2 Chipset

O *Chipset* é o componente da placa-mãe responsável por fazer o controle de transferência de dados de todos os componentes do computador, ou seja, é responsável por organizar a comunicação entre os componentes do computador. Ele é dividido em dois:

- Chipset sul: Gerencia os dados dos periféricos mais lentos do computador.
- Chipset norte: Gerencia os dados dos periféricos mais rápidos do computador.

Vale ressaltar que, em placas-mãe mais modernas, o Chipset Norte (Northbridge) foi praticamente eliminado, pois suas funções foram incorporadas ao próprio processador. Hoje, a maioria das placas-mãe usa apenas o Chipset Sul (Southbridge), que gerencia comunicação com periféricos, armazenamento e portas USB. A placa-mãe possui várias conexões diferentes, cada uma específica para determinados componentes de hardware.

4.3 Barramentos

Os barramentos são as vias de comunicação dentro do computador que interligam os diversos componentes. São os caminhos por onde as informações trafegam entre a CPU, memória e dispositivos periféricos. Existem diferentes tipos de barramentos, classificados de acordo com sua função e estrutura.

4.4 Chipsets e barramentos

Os *chipsets* são conjuntos de circuitos integrados fundamentais para a arquitetura de um computador, responsáveis por gerenciar a comunicação entre o processador, a memória, o armazenamento e outros periféricos.

Eles desempenham um papel crucial na coordenação do fluxo de dados entre diferentes componentes do sistema, utilizando barramentos para facilitar essa comunicação.

Barramentos, como PCIe, USB e SATA, servem como os meios pelos quais os dados são transferidos entre os dispositivos conectados, enquanto o chipset atua como um intermediário que regula e controla esses fluxos.

Com o avanço das tecnologias, muitos chipsets modernos integraram funcionalidades antes distribuídas entre várias unidades, permitindo uma comunicação mais eficiente e direta entre o processador e os dispositivos, como memória e placas de expansão, melhorando o desempenho geral do sistema. Os chipsets são integrados à placa-mãe.

4.5 Conectores

Podemos considerar que os barramentos de um microcomputador são como uma grande mansão. Essa mansão precisa ter pontos de acesso para as pessoas transitarem por dentro da mansão. Cada porta interna leva as pessoas a diferentes cômodos, e cada porta externa também pode levar as pessoas a diferentes entradas da casa. Os conectores são como as portas dessa mansão.

Os conectores são divididos em dois tipos, que são o slot e a porta. A principal diferença entre slot e porta está na função e no local onde são usados dentro do computador.

- Slot: É um conector interno localizado na placa-mãe, projetado para receber placas de expansão como placas de vídeo, som e rede. Exemplos comuns incluem os slots PCI, PCI Express (PCIe) e RAM (memória).
- Porta: É um conector externo, geralmente localizado na parte traseira ou frontal do gabinete, permitindo a conexão de periféricos como teclados, mouses, impressoras e dispositivos USB. Exemplos incluem as portas USB, HDMI, Ethernet e áudio.

Em resumo, slots são usados para expandir os recursos internos do computador, enquanto portas servem para conectar dispositivos externos.

Socket:

Na placa-mãe, temos o conector interno (slot) onde colocamos o processador, que é chamado de *Socket* (soquete). Os sockets tem diferenças entre modelos. Para saber o que é compatível com a placa-mãe, deve-se consultar seu manual.

CPU_FAN:

É o conector interno (slot) destinado para o cooler, que serve para resfriar a placa e o processador, evitando que superaqueça. Alguns modelos avançados de placas-mãe

possuem conectores adicionais como CHA_FAN (para ventoinhas do gabinete) e AIO_PUMP (para sistemas de refrigeração líquida).

Slot de RAM:

A placa mãe também contém os slots de memória, conectores internos destinados para a memória RAM.

Conectores SATA:

Os conectores *SATA*, ou *Serial-ATA* são responsáveis pela conexão dos HDs e SSDs. Os conectores SATA podem se referir tanto a um slot quanto a uma porta, dependendo do contexto:

- Porta SATA: Na placa-mãe do computador, existem portas SATA que permitem a conexão de dispositivos de armazenamento, como discos rígidos (HDDs) e unidades de estado sólido (SSDs). Essas portas são conectores físicos que recebem os cabos SATA de dados e de energia.
- Slot SATA: Embora o termo "slot" seja mais comumente associado a slots PCIe ou slots de memória RAM, ele também pode ser usado para descrever o local onde um dispositivo SATA é conectado na placa-mãe. Em alguns casos, principalmente quando se trata de SSDs no formato M.2 que usam interface SATA, pode-se referir ao local de encaixe desse SSD como um slot.

5 - CPU

5.1 Conceito de CPU

Também conhecida como *CPU* (*Central Processing Unit*, ou Unidade Central de Processamento) ou *UCP*, a CPU é o componente de hardware responsável por todo o processamento lógico e aritmético realizado pelo computador durante a execução de um programa. Podemos então tratar a CPU como o cérebro do computador, já que tudo o que o computador faz passa, de alguma forma, pela CPU.

Atualmente existem processadores com mais de um núcleo ou “core”, que são chamados de MultiCore (Multinúcleos). Isso significa que dentro de cada processador existem diversos circuitos centrais de processamento ao invés de apenas um só. É como se fossem diversos cérebros dentro de uma única cabeça. Dentro do processador estão os seguintes componentes:

- ULA (Unidade Lógica e Aritmética): Responsável por executar operações matemáticas e lógicas, como somas, subtrações e comparações de valores.
- UC (Unidade de Controle): Decodifica e gerencia a execução das instruções, coordenando o fluxo de dados e garantindo que os componentes da CPU operem de forma sincronizada.
- Registradores: Pequenas memórias ultra rápidas que armazenam temporariamente dados e instruções em execução.
- Memória Cache: Memória de acesso rápido usada para armazenar dados frequentemente acessados pela CPU.

6 - Memória

A memória é todo componente que é capaz de armazenar informações. Existem diversos tipos de memórias, mas as mais conhecidas num contexto de hardware são a *RAM* e *ROM*, e o *HD* e *SSD*.

6.1 Memória Principal (RAM “Rapid Access Memory” e ROM “Read Only Memory”)

A memória principal é composta pela RAM e ROM. A RAM é uma memória que consegue guardar informações apenas enquanto estamos trabalhando com elas. As principais características deste tipo são:

- É volátil. Isso significa que quando o computador é desligado, todo o conteúdo que está na memória RAM é perdido.
- É diretamente endereçável pelo processador, ou seja, lida diretamente com a CPU.
- Usada para armazenar programas em execução e dados do utilizador.

A memória RAM é composta por circuitos eletrônicos que conseguem guardar informações apenas quando há energia elétrica. Por isso que quando desligamos a máquina as informações que estavam armazenadas por lá são perdidas.

Essa memória é usada pelo processador para armazenar os dados que estão sendo processados, funcionando como uma espécie de mesa de trabalho. A quantidade de memória RAM disponível, determina quais atividades o processador poderá executar. Um engenheiro não pode desenhar a planta de um edifício sobre uma carteira de escola. Caso a quantidade de memória RAM disponível seja insuficiente, o computador não será capaz de rodar aplicativos mais complexos. A memória RAM é capaz de responder às solicitações do processador numa velocidade muito alta.

A memória RAM é essencial para o funcionamento de um computador já que ela, juntamente com a memória ROM (Read Only Memory) atuam na função de memória principal. Quando um computador está desligado, a memória RAM está vazia. Então como que o computador vai funcionar? Através da utilização em seus instantes iniciais da ROM, uma memória onde os dados não são perdidos (ou seja , não é volátil). As memórias ROM são eletrônicas como a RAM, mas não perdem seu conteúdo após o desligamento do computador. O conteúdo também não pode ser modificado, e por isso que seu nome é “Read Only”(apenas leitura).

O conteúdo da memória ROM é gravado na fábrica, e portanto, não pode ser alterado e nem perdido quando o computador é desligado. Dentro da memória ROM está o *BIOS* (Basic Input/Output System) que é o sistema básico de entrada e saída. Ele é um firmware (programa que controla o funcionamento dos componentes de hardware) não volátil que inicializa o hardware do computador. Ele inicializa o sistema e reconhece os dispositivos conectados ao computador. Além disso, o BIOS atua como uma interface entre o hardware e o sistema operacional.

A memória RAM é vendida em placas, chamadas de “pentes de memória” (o nome oficial é módulo de memória). Existem dois grandes tipos de memória RAM, a estática e a dinâmica. A estática é mais rápida, e o cache e registradores são desse tipo. A memória principal, memória de vídeo e cache de disco (nos HD’s) são dinâmicas.

6.2 Memórias Secundárias (Memórias de massa ou permanentes)

São as memórias que conseguem guardar informações por muito tempo (mesmo sem estarem ligadas à energia elétrica). Existem vários formatos e capacidades, mas iremos falar apenas das mais populares.

Uma das memórias secundárias mais conhecidas é a do disco rígido, conhecido como HD (hard-disk). O HD é a maior memória do computador e possui partes móveis. Para compreender a diferença entre a memória RAM e o HD, você pode imaginar uma lousa e uma estante cheia de livros com vários problemas a serem resolvidos. Depois de ler nos livros (HD) os problemas a serem resolvidos, o processador usaria a lousa (a memória RAM) para resolvê-los. Assim que um problema é resolvido, o resultado é anotado no livro e a lousa é apagada para que um novo problema possa ser resolvido.

O HD é um componente magnético, formado por discos metálicos magnetizáveis. Acoplado a eles está um braço que magnetiza os discos/pratos enquanto eles rotacionam. Os dados (ou seja, os 0 e 1) são magnetizados então nos ímãs dos pratos que compõem o disco rígido. Por conta desse processo que esta é uma memória magnética. O disco rígido não é essencial para um computador, mas a sua ausência limita muito as suas ações.

Uma alternativa para memória secundária é o SSD (Solid State Drive) que é uma tecnologia de armazenamento considerada como a evolução do HD. O SSD não possui partes móveis e é construído em torno de um circuito integrado semicondutor o qual é responsável pelo

armazenamento, diferentemente dos sistemas magnéticos do HD. Os SSDs são muito mais rápidos que os HDs pois são dispositivos do tipo memória flash, tecnologia semelhante aos pen-drives. Os SSDs não possuem discos rígidos iguais ao HD, e sim chips onde são armazenados os dados de forma eletrônica. O HD é mais lento por causa do tempo que se leva para fazer uma leitura nos discos. Existem também algumas memórias secundárias móveis como disquetes, CD-ROM, pen-drive entre outros.

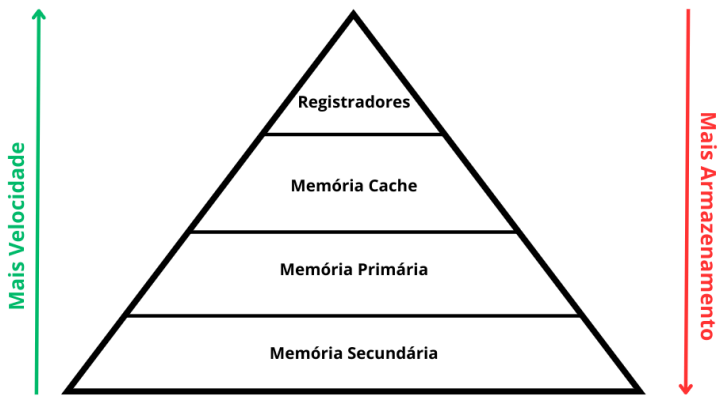
6.3 Memória Virtual

A memória virtual não é um componente físico, mas sim um recurso utilizado quando a memória RAM está esgotada (ou seja, quando não há mais espaço para ser utilizado na memória RAM). Os sistemas operacionais atuais permitem usar o HD e o SSD para gravar dados caso isso aconteça. Esse recurso é chamado de memória virtual.

Ao utilizar essa técnica, mesmo que a memória RAM esteja completamente ocupada, o programa será executado, porém muito lentamente. Isso porque, como já foi dito, a CPU se comunica diretamente com a memória RAM e não com o HD/SSD. Quando o processador necessita acessar um espaço emprestado pelo HD/SSD, ele tem de passar pelo chipset (responsável por fazer o controle de transferência de dados de todos os componentes do computador) e por esse ser um caminho mais longo, ele promove a lentidão.

6.4 Hierarquia de memória

É uma forma de organização das memórias presentes em um computador baseando-se em determinados aspectos, como velocidade, capacidade e custo. Essa hierarquia geralmente é representada através de uma pirâmide, onde os dispositivos mais velozes, e consequentemente mais caros, estão no topo. Na base estão as memórias mais lentas, porém mais baratas.



6.5 RAID

RAID (Redundant Array of Independent Disks, ou Conjunto Redundante de Discos Independentes) é uma tecnologia utilizada para aumentar o desempenho, a confiabilidade e a segurança dos dados em sistemas de armazenamento. Ela funciona através da combinação de múltiplos discos rígidos ou SSDs para formar um único sistema de armazenamento, melhorando a tolerância a falhas e, em alguns casos, aumentando a velocidade de leitura e escrita.

O princípio fundamental do RAID é a redundância, ou seja, a gravação dos dados de forma distribuída entre dois ou mais discos, reduzindo o risco de perda de informações em caso de falha de um dos dispositivos. Dependendo do tipo de RAID implementado, os benefícios podem incluir aumento da velocidade de acesso, proteção contra falhas de hardware e otimização do espaço de armazenamento disponível. Os sistemas RAID podem ser implementados de duas formas:

- RAID por hardware: Utiliza um controlador dedicado, geralmente embutido na placa-mãe ou em uma controladora RAID independente. Esse método oferece melhor desempenho, pois o processamento é feito diretamente pelo hardware, sem sobrecarregar a CPU do sistema.
- RAID por software: Realizado pelo sistema operacional sem a necessidade de um controlador dedicado. Apesar de ser uma opção mais acessível, pode impactar o desempenho, pois consome recursos do processador da máquina.

Níveis de RAID:

Existem diferentes níveis de RAID, cada um com suas características específicas:

- RAID 0 (Striping): Distribui os dados entre dois ou mais discos para aumentar a velocidade de leitura e escrita. No entanto, não possui redundância; se um dos discos falhar, todos os dados serão perdidos.
- RAID 1 (Mirroring): Cria uma cópia idêntica dos dados em dois ou mais discos, garantindo alta redundância e segurança, mas sem otimização de desempenho ou espaço de armazenamento.
- RAID 5: Utiliza no mínimo três discos e distribui os dados junto com informações de paridade, permitindo que o sistema continue funcionando mesmo com a falha de um disco. Oferece um equilíbrio entre desempenho, espaço e segurança.
- RAID 6: Semelhante ao RAID 5, mas com uma camada extra de paridade, permitindo que até dois discos falhem sem perda de dados.

- RAID 10 (ou RAID 1+0): Combina RAID 1 e RAID 0, oferecendo alta velocidade e redundância, mas exigindo pelo menos quatro discos.

Além desses, existem outros níveis e variações, cada um projetado para atender diferentes necessidades de armazenamento e segurança. A escolha do tipo de RAID ideal depende do cenário de uso, do orçamento e da importância da proteção dos dados.

7 - Periféricos

7.1 Periféricos Usuário-CPU

Para permitir a comunicação entre o usuário e o processador, que é o cérebro do computador, temos os periféricos de entrada e saída, ou I/O “input/output”. Estes são os olhos, ouvidos e boca do processador, por onde ele recebe e transmite informações. Os periféricos de entrada e saída são divididos da seguinte forma:

- Entrada: Recebe dados do usuário. Nesta categoria podemos enquadrar o teclado, mouse, microfone, etc.
- Saída: Exibe dados para o usuário. Temos, por exemplo, o monitor, impressoras, caixas de som, etc.
- Entrada e Saída: São os periféricos híbridos, que recebem e exibem dados. O maior exemplo é o monitor touchscreen.

7.2 Periférico CPU-Componentes Hardware

Existem outros tipos de periféricos que estão destinados a fazer a comunicação entre o processador e os demais componentes internos do micro, como a memória RAM e o disco rígido. Os dispositivos que fazem parte desta categoria estão dispostos basicamente na placa mãe, e incluem controladores de discos, controladores de memória, etc.

Placa de vídeo:

É o componente responsável por exibir os resultados do que é feito pelo computador na tela do monitor. Existem placas “onboard” e “offboard”.

- Onboard: São placas que já vêm, por padrão, soldadas na placa-mãe. Precisam ter o processamento feito pela CPU do computador.
- Offboard: São placas expansivas em que podemos instalar no computador através de uma placa de expansão. Essas placas possuem processamento e memória próprias, fazendo com que o processamento gráfico tenha um desempenho melhor.

A placa de vídeo pode ser considerada um dispositivo periférico de saída.

Placa de rede:

É um componente que ganhou muita importância ao longo dos anos. Por padrão, qualquer placa-mãe virá com uma conexão de rede do tipo ethernet (rede cabeada). Em notebooks também se faz presente a placa de rede wireless (sem fio). A placa de rede é um dispositivo periférico de entrada e saída.

8 - Outros componentes de Hardware

8.1 Fonte de alimentação

É responsável por fornecer energia ao computador, transformando a corrente alternada que vem da tomada em corrente contínua, usada pelos computadores, que podem ser -5, 5, -12 ou 12 volts. A fonte deve fornecer energia suficiente para todos os componentes de um computador para que estes funcionem corretamente e não danifiquem.

8.2 Gabinete

A peça do computador desktop onde serão instalados todos os componentes, começando pela placa-mãe que é fixada na lateral do gabinete através de parafusos. Assim os componentes restantes são conectados na placa-mãe.

Capítulo III: Como um computador funciona

9 - Instruções e Ciclo de Instrução

9.1 Instruções e Programas

Já ficou estabelecido que um computador é uma máquina responsável por solucionar problemas e executar tarefas, e a maneira que ele faz isso é obedecendo ordens e comandos que estão na forma de instruções. O conjunto dessas instruções é o que nós chamamos de algoritmo, programa, ou software.

Como já havia sido discutido antes, a linguagem humana é muito diferente da linguagem de máquinas. Enquanto trabalhamos com ideias, imagens, falas, escritas, os microcomputadores só lidam com a linguagem binária. Então, essas instruções que estamos passando para um computador devem ser convertidas na linguagem binária, transformando-se em instruções elétricas para que a CPU obedeça.

9.2 Funcionamento e Ciclo de Instrução

Todos os programas que estão em um computador, e não estão sendo utilizados no momento, são armazenados no HD/SSD, ou seja, na memória secundária (aquela onde os dados não são perdidos por falta de uso). Quando abrimos um programa que estava fechado, uma “cópia” dele vai para a RAM, ou seja, a memória principal (aquela onde os dados são armazenados apenas quando estão em uso). Os dados originais do programa não saem da memória secundária sem estarem em uma cópia, pois caso contrário, correria-se o risco de perder os dados quando eles fossem alocados para RAM, num eventual desligamento da máquina.

O programa só começa a ser executado quando ele está na memória RAM, pois a CPU (que é o cérebro do computador, responsável por todas as decisões lógicas e aritméticas) não consegue buscar instruções de forma direta no HD/SSD. A CPU precisa acessar a memória RAM para buscar as informações e armazená-las em seus registradores, componentes internos de alta velocidade usados para guardar temporariamente dados e instruções durante o processamento.

Nesse processo, o contador de programa (PC) armazena o endereço da próxima instrução a ser executada, enquanto o registrador de instrução (IR) guarda a instrução atualmente sendo processada. Quando uma instrução precisa ser executada, ela passa por esses registradores e, se necessário, a ULA (Unidade Lógica e Aritmética) realiza operações

matemáticas ou lógicas, armazenando os resultados em registradores como o acumulador (ACC).

Assim que a CPU tiver acesso às instruções, ela vai lê-las, decodificá-las e executá-las uma por uma, fazendo com que o programa funcione. Esse ciclo é conhecido como *Ciclo de Instrução*.

9.3 Formatos de Instrução

Os formatos de instrução se referem à estrutura das instruções. Essas instruções consistem em um opcode (que especifica a operação a ser realizada) e endereços (que indicam os operandos e onde os resultados vão). Os tipos de formatos de instrução são:

- Sem endereço: Só contém o opcode.
- Um endereço: Contém o opcode e um endereço.
- Dois endereços: Contém o opcode e dois endereços.
- Três endereços: Contém o opcode e três endereços.

Comprimento das Instruções:

Algumas máquinas têm instruções de comprimento fixo, enquanto outras têm comprimentos variáveis. Instruções de comprimento fixo são mais fáceis de decodificar, mas podem desperdiçar espaço.

9.4 Tipos de Instrução

Os tipos de instrução em um processador se referem às categorias de operações que ele pode executar, cada uma com um objetivo específico no processamento de dados e controle de fluxo de execução de programas. Em um conjunto de instruções (chamado de ISA, ou Arquitetura Conjunto de Instruções, na sigla em inglês), as instruções são classificadas em tipos conforme a operação que realizam. Essas instruções formam a base para a execução de programas em qualquer computador.

Instruções Aritméticas:

Essas instruções realizam operações matemáticas, como adição, subtração, multiplicação e divisão. Elas lidam com operações diádicas, que são operações realizadas entre dois operandos. Exemplos de operações diádicas incluem:

- ADD: Adição de dois valores.
- SUB: Subtração de dois valores.
- MUL: Multiplicação de dois valores.

- DIV: Divisão de dois valores.

Essas operações são fundamentais para realizar cálculos em qualquer programa. São amplamente utilizadas em algoritmos numéricos, gráficos, criptografia, entre outros.

Instruções Lógicas:
Essas instruções operam sobre os bits de uma palavra e podem ser classificadas em operações monádicas e diádicas. Entre as operações monádicas (operando com um único operando) temos:

- NOT: Negação lógica (inverte os bits do operando).

Já entre as diádicas (operando com dois operandos):

- AND: Operação lógica "E" entre dois operandos.
- OR: Operação lógica "OU" entre dois operandos.
- XOR: Operação lógica "OU exclusivo" entre dois operandos.

Essas operações são essenciais para manipulação de dados em nível de bit, como em algoritmos de criptografia, controle de fluxo e manipulação de strings.

Instruções de Memória:
Essas instruções movem dados entre a memória e os registradores. O acesso aos registradores é mais rápido que o da memória principal, o que torna essas instruções vitais para o gerenciamento eficiente de dados. As instruções de memória em si podem envolver operações diádicas, já que elas geralmente precisam de dois operandos: um para a localização da memória e outro para o registrador ou o valor a ser movido. Exemplos de instruções de memória incluem:

- MOV: Mover dados de um local para outro (registrador para registrador ou memória para registrador).
- LOAD: Carregar um valor da memória para um registrador.
- STORE: Armazenar um valor de um registrador na memória.

Essas instruções são essenciais para o funcionamento do processador e para acessar dados de maneira eficiente.

Instruções de Entrada/Saída (E/S):
Essas instruções lidam com a movimentação de dados entre o processador e os dispositivos periféricos, como teclados, impressoras ou discos rígidos. Elas envolvem

tipicamente operações diádicas, já que manipulam dois elementos: o dispositivo de entrada/saída e os dados que estão sendo transferidos.
Exemplos incluem:

- IN: Ler dados de um dispositivo de entrada.
- OUT: Enviar dados para um dispositivo de saída.
- PUSH: Inserir dados na pilha (e.g., salvando registros de retorno em chamadas de função).
- POP: Remover dados da pilha (e.g., restaurando valores de retorno).

Essas instruções são vitais para a interação do processador com o mundo exterior, permitindo a comunicação entre o sistema e o usuário ou outros sistemas.

Instruções de Teste e Desvio:

Essas instruções comparam operandos e, com base no resultado da comparação, controlam o fluxo de execução do programa, realizando desvios condicionais. As instruções de teste são geralmente monádicas, porque operam em apenas um operando, enquanto as instruções de desvio são diádicas, pois dependem de uma comparação entre dois valores. Entre as operações monádicas estão:

- CMP: Compara dois valores (geralmente no contexto de definir flags ou condições de desvio).

E as diádicas:

- BEQ: Desvia se os valores comparados forem iguais.
- BNE: Desvia se os valores comparados forem diferentes.
- JMP: Desvia incondicionalmente para um novo endereço de memória.

Essas instruções são essenciais para o controle de fluxo do programa, implementando loops, condições e verificações.

10 - Memória Cache

10.1 O que é Memória Cache

É importante ressaltar que existe um personagem intermediário nesse Ciclo de Instrução, chamado de *Memória Cache*. A cache está localizada dentro da CPU e serve como intermediária entre a sua comunicação com a RAM. A memória cache é uma RAM Estática, e é muito mais rápida que a RAM principal. A cache guarda informações presentes na RAM que possam ser utilizadas novamente pela CPU, reduzindo assim o tempo que seria necessário para buscar essas informações. A comunicação entre a CPU e a cache é muito mais rápida que a comunicação entre a CPU e a RAM, pois a cache já está inserida dentro da CPU, e não precisaria percorrer todo o caminho da placa-mãe como a CPU faz quando quer acessar a memória RAM. Esse tipo de memória está presente também nos navegadores de internet e nos servidores, mas eles não são a mesma coisa.

Podemos concluir então que a memória cache guarda as instruções/informações que são mais usadas, a fim de agilizar processos repetitivos. Quem define quais dados são mais frequentemente utilizados é a própria CPU. A memória cache possui três níveis.

L1 (Nível 1 ou primária):
É a menor e mais rápida, armazenada diretamente dentro dos núcleos do processador. Geralmente varia entre 16 KB e 128 KB por núcleo e é dedicada exclusivamente a cada um deles. Sua velocidade extremamente alta permite que a CPU acesse os dados praticamente sem demora.

L2 (Nível 2 ou secundária):
Maior que a L1, mas um pouco mais lenta. Pode ser dedicada a cada núcleo ou compartilhada entre alguns núcleos. Seu tamanho varia entre 256 KB e alguns megabytes, e seu papel é armazenar dados que não couberam na L1, mantendo um equilíbrio entre capacidade e velocidade.

L3 (Nível 3 ou terciária):
Geralmente compartilhada entre todos os núcleos do processador, possui maior capacidade (de alguns megabytes até dezenas de megabytes) e é mais lenta que a L2. Sua função é evitar acessos frequentes à RAM, armazenando um conjunto maior de dados que podem ser usados por múltiplos núcleos da CPU.

10.2 Funcionamento da Memória Cache

A memória cache opera com base nos princípios de localidade temporal e localidade espacial:

- Localidade temporal: Dados acessados recentemente têm maior probabilidade de serem acessados novamente em breve.
- Localidade espacial: Dados próximos a um dado acessado recentemente têm maior probabilidade de serem utilizados.

Quando a CPU precisa de uma informação, ela primeiro verifica a cache L1. Se os dados não estiverem lá, ela verifica a cache L2, depois a L3, e por fim, acessa a memória RAM. Esse processo reduz significativamente o tempo de espera da CPU.

Além disso, a cache segue políticas de substituição para gerenciar quais dados devem ser mantidos ou removidos, como LRU (Least Recently Used), que descarta os dados menos utilizados recentemente para liberar espaço. A memória cache não é essencial para o funcionamento de um computador, mas a sua presença é de grande ajuda para otimização de desempenho.

11 - BIOS e Firmware

11.1 BIOS

Quando o computador está desligado, a memória RAM está vazia. Então como que a CPU vai adquirir as primeiras instruções necessárias? Através da BIOS que está presente na memória ROM. A BIOS executa um processo chamado POST (Power-On Self Test), verificando o hardware básico e, em seguida, procura um dispositivo de inicialização (HD, SSD, pendrive, etc.) para carregar o sistema operacional na RAM e permitir que a CPU comece a executá-lo. O processo de inicialização de um computador pode ser descrito da seguinte forma:

1. Ligamento do computador – Ao pressionar o botão de ligar, a fonte de alimentação energiza os componentes, incluindo a CPU.
2. Execução da BIOS/UEFI – A CPU busca na memória ROM (ou memória flash) as primeiras instruções da BIOS/UEFI.
3. POST (Power-On Self Test) – A BIOS executa o POST para verificar se componentes essenciais (RAM, teclado, placa de vídeo, etc.) estão funcionando corretamente.
4. Busca por um dispositivo de boot – Após o POST, a BIOS procura um HD, SSD, pendrive ou outro dispositivo bootável (qualquer dispositivo de armazenamento que contenha um sistema operacional completo e funcional) que contenha um sistema operacional.
5. Carregamento do sistema operacional – A BIOS/UEFI encontra o bootloader (um programa que inicializa o sistema operacional de um dispositivo eletrônico como GRUB no Linux ou Windows Boot Manager no Windows) e o carrega na RAM.
6. Execução do bootloader pela CPU – O bootloader então carrega o núcleo do sistema operacional na RAM.
7. Sistema operacional assume o controle – A partir desse momento, o SO gerencia o hardware e executa os programas conforme necessário.

Cada passo envolve leitura, cópia e execução de instruções, sempre coordenadas entre a CPU, RAM e dispositivos de armazenamento.

O conteúdo da ROM é gravado na fábrica e não pode ser alterado. No computador, é comum o uso de um pequeno chip de memória ROM, na placa-mãe, que guarda o BIOS. Mesmo que você consiga alterar a BIOS do seu computador, isso não quer dizer que você conseguiu alterar o conteúdo da memória ROM. Os programas gravados em memória

ROM são chamados de firmware. Hoje em dia as placas-mães não utilizam mais memória ROM puramente, e sim memórias do tipo flash que são consideradas uma evolução da memória ROM, permitindo que a BIOS possa ser alterada.

11.2 Firmware

Firmware é um tipo de software que está integrado no hardware de um dispositivo. Ele controla a funcionalidade do dispositivo, permitindo que o sistema operacional e os aplicativos funcionem corretamente. Podemos listar as seguintes características dos firmwares:

- É uma camada intermediária entre o hardware e o software
- É geralmente armazenado em memória não volátil, como ROM ou flash
- Não é projetado para ser modificado ou atualizado frequentemente
- É responsável por gerenciar as funções básicas do dispositivo
- Permite que o hardware interaja com o sistema operacional e os aplicativos

O firmware é fundamental para o funcionamento do dispositivo, pois atua como uma ponte entre o hardware e o software. Por isso, é importante mantê-lo atualizado para se proteger de malwares.

12 - Funcionamento e tipos de Barramento

12.1 Conceito de Barramento

Os barramentos são as vias de comunicação dentro do computador que interligam os diversos componentes. São os caminhos por onde as informações trafegam entre a CPU, memória e dispositivos periféricos. Existem diferentes tipos de barramentos, classificados de acordo com sua função e estrutura.

Caminho compartilhado x caminho dedicado:

Os barramentos foram feitos com o intuito de simplificar a construção de computadores. Isso se dá porque um barramento serve como um caminho compartilhado entre os diversos componentes do computador, ao invés de utilizar caminhos dedicados (destinados para a comunicação somente entre componentes específicos.

Gargalo:

Entretanto, essa solução traz um ponto negativo. Por ser um caminho compartilhado, o barramento não deve ser usado por mais do que uma dupla de componentes. Imagine uma situação onde a CPU esteja enviando sinais elétricos para a memória RAM, criando assim um tráfego de informações no caminho compartilhado do barramento. Enquanto essa comunicação estiver sendo feita, os outros componentes não poderão utilizar esse mesmo caminho. Esse problema é chamado de “gargalo”.

12.2 Barramento de sistema

O barramento de sistema é o principal barramento do computador, essencial para seu funcionamento. Ele liga a CPU com a memória principal e outros componentes fundamentais. Tradicionalmente, o barramento de sistema era subdividido em três partes:

- Barramento de dados: Transporta os dados entre a CPU e a memória
- Barramento de endereços: Indica os locais na memória onde os dados devem ser lidos ou gravados
- Barramento de controle: Coordena as operações de leitura e escrita.

Nas arquiteturas modernas os barramentos tradicionais foram substituídos por interconexões de alta velocidade como HyperTransport e Ultra Path Interconnect, que oferecem um maior desempenho.

12.3 Barramentos de expansão

Barramentos secundários que permitem a conexão dos periféricos e componentes adicionais ao computador. Eles podem ser classificados conforme sua localização:

- Externos: Ligam periféricos fora do gabinete
- Internos: Ligam componentes dentro do gabinete

Eles também podem ser classificados de acordo com o modo de transmissão de dados:

- Seriais: Transmitem os dados sequencialmente, um bit por vez. Esse tipo de barramento passou a ser mais utilizado devido à sua maior eficiência e capacidade de transferência em altas velocidades.
- Paralelos: Transmitem vários bits ao mesmo tempo, através de várias linhas. Era muito usado nos tempos passados, mas hoje são menos comuns em conexões externas. Entretanto, ainda são utilizados em algumas aplicações internas, como na comunicação entre módulos de memória RAM.

USB:

Um dos barramentos mais conhecidos é o *USB*. É um barramento de expansão serial e externo, que pode ligar (teoricamente) até 127 equipamentos ao mesmo tempo. Ele possui a habilidade conhecida como *Hot Swap*, que permite a conexão e desconexão de dispositivos sem precisar desligar o computador. Outro atributo neste barramento é o *Plug and Play*, que é a detecção automática de dispositivos conectados. As versões do barramento USB (Universal Serial Bus) evoluíram ao longo do tempo para oferecer maior velocidade e eficiência na transferência de dados e na alimentação de dispositivos. O USB 1.0 e 1.1, lançados na década de 1990, ofereciam velocidades de até 12 Mbps. O USB 2.0, introduzido no ano 2000, aumentou significativamente essa taxa para 480 Mbps. Com o USB 3.0, lançado em 2008, a velocidade subiu para 5 Gbps, sendo aprimorado pelo USB 3.1 (10 Gbps) e pelo USB 3.2 (até 20 Gbps). Atualmente, o USB4, baseado na tecnologia Thunderbolt 3, pode atingir velocidades de 40 Gbps, proporcionando maior desempenho para dispositivos de alta demanda, como monitores e armazenamento externo. Além da velocidade, cada nova versão trouxe melhorias na eficiência energética e na compatibilidade com dispositivos anteriores.

PCI Express:

O PCI Express (PCIe) é um barramento de expansão serial e interno usado para conectar placas de expansão em computadores e servidores. Ele substituiu o antigo PCI, oferecendo maior largura de banda e eficiência. Atualmente, o PCIe é amplamente utilizado para

conectar dispositivos como placas de vídeo, placas de rede e placas de captura de vídeo. Algumas placas de som ainda utilizam PCIe, mas muitas migraram para conexões USB ou interfaces profissionais dedicadas. Modems internos PCIe são raros, pois a maioria dos dispositivos de internet utilizam conexões externas, como Wi-Fi ou USB. O PCIe opera em diferentes configurações de largura de banda, como x1, x4, x8 e x16, que determinam a quantidade de dados que podem ser transferidos simultaneamente, sendo o x16 o mais comum para placas de vídeo.

SATA:

O SATA (Serial ATA) é um barramento de expansão serial e interno utilizado para conectar discos rígidos (HDDs), SSDs SATA e unidades ópticas como DVD, CD e Blu-ray. Ele substituiu o antigo padrão PATA, oferecendo maior velocidade e cabos mais finos. Embora o SATA tenha suporte para Hot Swap, permitindo a conexão e desconexão de discos sem desligar o computador, esse recurso depende da placa-mãe e do sistema operacional, sendo mais comum em servidores e sistemas com suporte adequado.

12.4 Largura de barramento

A largura de um barramento, em termos de computação, refere-se à quantidade de dados que podem ser transmitidos simultaneamente entre componentes de um computador. Ela é medida em bits, e quanto maior a largura do barramento, mais dados podem ser transferidos por vez, resultando em maior velocidade e desempenho.

13 - Arquitetura de Von Neumann e Arquitetura de Harvard

13.1 Conceito de arquitetura de computadores

A arquitetura de um computador descreve a organização e funcionamento dos componentes de seu sistema. Existem diversos tipos de arquiteturas, porém duas delas são as mais populares.

13.2 Arquitetura de Von Neumann

A arquitetura de Von Neumann baseia-se em cinco componentes básicos de um computador. São eles:

- Unidade de entrada: Responsável por obter dados e instruções para o funcionamento do sistema computacional.
- Unidade de Controle (UC): Gerencia o processamento de dados.
- Unidade Lógica e Aritmética (ULA): A unidade que realiza o processamento de dados.
- Unidade de memória: Armazena os dados processados.
- Unidade de saída: Exibe os resultados.

Em primeiro lugar, o computador recebe os dados através da Unidade de Entrada, e a partir de seu processamento, gera informações que serão úteis para a resolução de um problema ou tarefa específica.

No segundo passo, para que o processamento seja feito da forma correta é necessário que haja um gerenciamento da Unidade de Controle. A UC vai gerenciar os recursos do computador e coordenar o funcionamento de suas partes.

O terceiro passo refere-se a ULA. O componente que vai de fato realizar o processamento é a Unidade Lógica e Aritmética. Ela é responsável por fazer operações como soma, subtração, divisão, multiplicação, além de operações relacionais e lógicas.

O quarto passo é referente ao armazenamento. Os dados que estão sendo processados (ou parte deles) são armazenados na Unidade de Memória.

Por fim, esses dados processados, que agora são informações, devem ser disponibilizados pelo usuário a partir da Unidade de Saída.

Toda essa lógica foi trabalhada nos capítulos passados através dos conceitos dos componentes de hardware. Os periféricos de entrada representam a Unidade de Entrada, a CPU representa tanto a UC quanto a ULA, A memória RAM e HD/SSD representam a Unidade de Memória, e os periféricos de saída representam a Unidade de Saída.

No modelo de Von Neumann, os barramentos (as conexões entre os componentes destinados para tráfego de dados) utilizam de um grande caminho compartilhado. Como já havia sido explicado anteriormente, isso facilita o desenvolvimento de um computador, mas cria um problema chamado de *Gargalo de Von Neumann*, pois a utilização de um único caminho impede que vários componentes possam usá-lo ao mesmo tempo, limitando a velocidade de processamento.

A compreensão de como um programa é executado tem ligação direta com o entendimento do modelo de Von Neumann. Como já foi explicado, é feita uma cópia do conjunto de instruções necessárias para a execução de um programa que estava armazenado na memória. Esta cópia, por sua vez, é armazenada nos registradores do processador. As instruções desta cópia serão então endereçadas, uma de cada vez, para o processador (ou melhor, a ULA). Agora, uma instrução será lida, decodificada e executada, passando assim para a próxima. Os registradores vão guardando os resultados dessas instruções. Essa lógica sequencial de execução é uma característica desta arquitetura.

É importante ressaltar que as instruções e dados ficam armazenados na mesma memória, que é a principal (RAM). Isso significa que a memória armazena tanto o código do programa quanto os dados que ele manipula, sem distinção física entre os dois. Esse fato contribui para a ocorrência do gargalo de Neumann.

Podemos elencar então como principais características desta arquitetura as seguintes:

- Memória única: Instruções e dados compartilham o mesmo espaço de memória.
- Barramento único: Existe apenas um caminho para transferência de dados e instruções entre a CPU e a memória.
- Execução sequencial: As instruções são buscadas, decodificadas e executadas uma de cada vez.
- Flexibilidade: A mesma memória pode armazenar tanto código quanto dados, permitindo programas auto modificáveis.

Vantagens da Arquitetura Von Neumann:

- Maior flexibilidade no armazenamento de dados e programas.
- Facilidade na implementação e custo reduzido.
- Utilização eficiente da memória.

Desvantagens da Arquitetura Von Neumann:

- Gargalo de Von Neumann: O uso de um único barramento para dados e instruções limita a velocidade de processamento.
- Acesso mais lento à memória, pois instruções e dados competem pelo mesmo canal.

13.3 Arquitetura de Harvard

A arquitetura de Harvard é um modelo de arquitetura de computador que separa as memórias de dados e instruções. Elas não ficarão mais armazenadas no mesmo local, permitindo que o processador acesse simultaneamente instruções e dados, o que aumenta a eficiência do processamento.

Uma outra mudança refere-se à separação de barramentos. Ao invés de um único caminho compartilhado, a arquitetura de Harvard apresenta uma independência de barramentos, ou seja, caminhos separados para a transferência de dados e instruções. Podemos então listar como características principais desse modelo:

- Memória separada: Dados e instruções são armazenados em memórias distintas.
- Barramentos independentes: Existem caminhos separados para transferência de dados e instruções.
- Maior eficiência: A CPU pode buscar uma instrução ao mesmo tempo que acessa um dado.

Vantagens da Arquitetura Harvard:

- Maior desempenho, pois elimina o gargalo de Von Neumann.
- Permite pipelines (série de etapas ou estágios através dos quais dados ou instruções são processados) mais eficientes, aumentando a velocidade de execução.
- Maior segurança, pois dados e instruções não se misturam.

Desvantagens da Arquitetura Harvard:

- Maior complexidade e custo de implementação.

- Menos flexibilidade, pois os tamanhos das memórias de dados e instruções são fixos.

14 - Arquitetura de CPU

14.1 Instruções e Clock

Nós já vimos que um programa é um conjunto de instruções, que por sua vez, através do ciclo de instruções, serão endereçadas, lidas, decodificadas e executadas pela CPU do computador.

Para um computador executar uma instrução, quanto mais complexa ela for, mais tempo será necessário. Para medir esse tempo, deve-se levar em consideração a quantidade de instruções que foram executadas no intervalo de um segundo, que é medida por Hertz. Ou seja, se um processador consegue executar 3 instruções por segundo, ele possui uma velocidade de 3 Hertz por segundo, ou 3 Hz. Esse ciclo de instruções que acontece no período de um segundo é o que chamamos de *Ciclo de Relógio* ou *Clock*.

Diante desses fatos, é necessário entender como os tipos de arquiteturas dos processadores influenciam na forma como eles lidam com essas instruções.

14.2 CISC (Complex Instruction Set Computer)

O CISC é uma arquitetura projetada para um computador com um conjunto de instruções muito amplo e complexo. Com isso, o processador é capaz de executar centenas de instruções complexas diferentes, através das instruções que estão gravadas nele mesmo via micro programação.

Cada instrução de um programa é representada por múltiplas instruções de Hardware. Portanto, um processador com arquitetura CISC consegue comportar mais instruções. Por isso, a execução dessas instruções também fica mais lenta.

Vantagens do CISC:

- Menos Instruções: Programas escritos em CISC tendem a ser mais compactos, pois podem realizar operações complexas com menos linhas de código.
- Eficiência de Memória: Ao usar instruções mais complexas, há uma redução na quantidade de memória necessária para armazenar o código.

Desvantagens do CISC:

- Complexidade do Hardware: A implementação de um conjunto de instruções complexo requer hardware mais sofisticado, o que pode aumentar o custo e a dificuldade de desenvolvimento.
- Desempenho Variável: Algumas instruções podem levar mais tempo para serem executadas, resultando em um desempenho inconsistente.

14.3 RISC (Reduced Instruction Set Computer)

O RISC é uma arquitetura projetada para um computador com um conjunto reduzido de instruções simples. O design RISC busca aumentar a eficiência através da execução de instruções simples que podem ser realizadas em um único ciclo de clock. Processadores com essa tecnologia não possuem microprogramação, e cada instrução do programa é executada diretamente pelo hardware. A falta de microprogramação implica em uma maior simplicidade, ou seja, essa arquitetura comporta menos instruções, e portanto também é mais rápida. O foco está em maximizar a velocidade de execução e simplificar a implementação do hardware.

Vantagens do RISC:

- Desempenho Elevado: Instruções simples podem ser executadas rapidamente, permitindo um desempenho consistente e previsível.
- Simplicidade do Hardware: O hardware é geralmente mais simples e menos caro de desenvolver e fabricar, uma vez que o conjunto de instruções é menor.

Desvantagens do RISC:

- Mais Instruções: Programas em RISC podem ser mais longos, pois muitas operações complexas requerem várias instruções simples para serem realizadas.
- Eficiência de Memória: A necessidade de mais linhas de código pode resultar em um maior uso de memória.

14.4 Arquitetura x86 e x64

Arquitetura x86:

Esta é uma das mais tradicionais arquiteturas de processadores. Suas principais características são a adoção do modelo CISC (permitindo instruções mais complexas), e a retrocompatibilidade, ou seja, ela suporta softwares de versões anteriores.

Atualmente é possível encontrar processadores x86 em sistemas embarcados antigos, visto que foram substituídos por uma nova versão desse modelo. Além disso, os registradores desse tipo de CPU possuem uma largura de 32 bits. Isso significa que o tamanho máximo de dados que poderão ser processados em uma única instrução é de 32 bits.

Arquitetura x64:

É uma extensão da x86, que suporta o processamento de 64 bits. Isso significa que o tamanho máximo de dados que poderão ser processados em uma única instrução é de 64 bits. A arquitetura x64 suporta a execução de aplicativos tanto em 32 bits quanto em 64 bits. Além disso, este modelo permite o uso de mais memória RAM, fator que contribui para uma maior eficiência no processamento de dados. Hoje a maioria dos computadores pessoais possuem esse tipo de processador

14.5 Arquitetura ARM

ARM (Advanced RISC Machine) é baseada na filosofia RISC (Reduced Instruction Set Computing), utilizando um conjunto de instruções mais enxuto e eficiente. É amplamente adotada em dispositivos móveis, tablets e sistemas embarcados.

14.6 Registradores

Os registradores são um tipo de memória extremamente rápida localizada dentro da CPU e podem ser divididos em diferentes categorias, incluindo os registradores de armazenamento e os registradores de instrução.

Registradores gerais:

Os registradores de armazenamento são usados para guardar dados temporários que a CPU precisa durante o processamento. Um exemplo clássico é o Acumulador (ACC), que armazena resultados intermediários de operações.

Registradores de instrução:

O registrador de instrução (IR - Instruction Register), por sua vez, tem a função de armazenar a instrução atual que está sendo decodificada e executada pela CPU. Ele recebe a instrução diretamente do barramento de memória e a mantém enquanto a CPU a processa.

Registradores especiais:

Existem diversos tipos de registradores especiais. Um deles é o Contador de Programa (PC - Program Counter), que armazena o endereço da próxima instrução a ser buscada da memória RAM. Diferente do IR, o PC não armazena a instrução em si, apenas o local onde ela está armazenada, permitindo que a CPU saiba qual deve ser o próximo passo na execução do programa.

Registradores de propósito específico:
Estes registradores são projetados para funções específicas e podem incluir:

- MAR (Memory Address Register): Armazena o endereço da memória que a CPU deve acessar
- MDR (Memory Data Register): Armazena os dados que estão sendo transferidos para ou da memória.
- MBR (Memory Buffer Register): Um registrador que atua como um buffer entre a CPU e a memória, armazenando dados temporariamente durante a transferência.

14.7 Unidade de Ponto Flutuante (FPU)

A computação de ponto flutuante é essencial para operações matemáticas complexas, como cálculos científicos, modelagem tridimensional e inteligência artificial. Diferente da aritmética inteira, que lida apenas com números inteiros, a aritmética de ponto flutuante permite representar números fracionários e de grande magnitude com maior precisão.

A Unidade de Ponto Flutuante (FPU) é um componente do processador projetado especificamente para realizar cálculos envolvendo números em ponto flutuante. Originalmente, a FPU existia como um coprocessador separado, mas atualmente está integrada diretamente na CPU. As principais funções da FPU são:

- Realiza operações matemáticas como adição, subtração, multiplicação e divisão de números de ponto flutuante.
- Manipula dados de alta precisão, como números de 32 bits (precisão simples) e 64 bits (precisão dupla).
- Utiliza instruções SIMD (Single Instruction, Multiple Data) para processamento vetorial eficiente, especialmente em aplicações de gráficos e aprendizado de máquina.

15 - Computação paralela

15.1 Instruções e Tarefas

Nós já vimos que um programa é um conjunto de instruções, apesar desta ideia ser uma simplificação. O que acontece, na realidade, é que as instruções podem ser agrupadas em uma único conjunto com um propósito específico, chamado de *tarefa*.

Uma tarefa como “carregar uma página da web” é composta por diversas instruções que vão permitir a sua realização. Portanto, um programa é um conjunto de tarefas, que são conjuntos de instruções. Antigamente, os sistemas operacionais suportavam a execução de apenas uma tarefa por vez. Ou seja, apenas após o fim da execução da primeira tarefa que a segunda poderia ser carregada na memória e executada.

15.2 Programa, processo e thread

Um processo é uma instância de um programa em execução e um conceito fundamental em qualquer sistema operacional. Quando um programa é executado, o sistema operacional cria um processo que possui seu próprio espaço de endereçamento, uma lista de posições de memória que define onde ele pode ler ou gravar informações. Cada processo é composto por:

- Código do programa (conjunto de instruções a serem executadas)
- Dados e variáveis utilizadas durante a execução
- Recursos do sistema, como acesso à memória e ao processador
- Threads, que representam fluxos de execução dentro do processo

Além disso, um único programa pode gerar múltiplos processos, dependendo do sistema operacional e da forma como foi projetado.

As *threads* são as menores unidades de execução em um processo. Elas são responsáveis por executar as tarefas. Um processo deve conter ao menos uma thread para ser executado, mas ele pode conter várias threads, o que permite a realização de várias tarefas de forma paralela.

15.3 Paralelismo

Foi com a necessidade de se executar mais de uma tarefa por vez que a computação paralela se desenvolveu. Nela as tarefas são divididas em partes menores e executadas de forma simultânea, ao invés de esperar a conclusão de uma tarefa para executar outra.

As subtarefas são independentes e distribuídas entre múltiplos processadores ou núcleos de um único processador. Cada processador/núcleo executa sua subtarefa ao mesmo tempo. Neste processo está presente a sincronização, garantido que as sub tarefas sejam executadas na ordem correta e que os resultados sejam devidamente combinados.

Existem dois tipos de paralelismo:

- Paralelismo de Dados: Os dados são divididos em partes, e cada processador ou núcleo processa uma parte diferente dos dados.
- Paralelismo de Tarefas: A tarefa é dividida em subtarefas diferentes, e cada processador ou núcleo executa uma subtarefa diferente.

15.4 Computação Paralela em GPUs

As GPUs (Graphics Processing Units) são projetadas especificamente para a computação paralela em larga escala. Diferente das CPUs (Central Processing Units), que possuem poucos núcleos altamente otimizados para tarefas sequenciais complexas, as GPUs possuem milhares de núcleos menores e mais eficientes para a execução simultânea de múltiplas operações em grande quantidade de dados. As principais características das GPUs para computação paralela são:

- Arquitetura Massivamente Paralela: Enquanto uma CPU pode ter entre 4 e 64 núcleos, uma GPU moderna pode ter milhares de núcleos, permitindo a execução de milhares de threads simultaneamente.
- Execução SIMD (Single Instruction, Multiple Data): As GPUs aplicam a mesma instrução a diferentes blocos de dados ao mesmo tempo, tornando-as ideais para computação científica, aprendizado de máquina e processamento gráfico.
- Uso em Cibersegurança: GPUs são amplamente utilizadas para acelerar ataques de força bruta e mineração de hashes criptográficos, mas também são empregadas para defesa, como na análise de grandes volumes de dados para detecção de anomalias em redes.

O uso de GPUs na computação paralela é um dos maiores avanços na otimização de tarefas que exigem alto desempenho, tornando-as indispensáveis para várias áreas da computação moderna.

16 - Nível de abstração

16.1 Conceito

Na computação, um nível de abstração é uma forma de ocultar os detalhes internos de um subsistema, permitindo que diferentes partes de um sistema interajam sem precisar conhecer sua implementação exata. Isso facilita a interoperabilidade, a modularidade e a independência da plataforma, permitindo que softwares e hardwares evoluam separadamente.

Na arquitetura de computadores, os níveis de abstração organizam a complexidade do sistema, separando as camadas de software e hardware. Isso possibilita que programadores desenvolvam aplicações sem precisar conhecer a lógica interna dos processadores e que engenheiros de hardware criem novos chips sem afetar o funcionamento dos programas existentes.

16.2 Principais Níveis de Abstração

A arquitetura de um computador pode ser dividida em múltiplas camadas de abstração, que se comunicam através de interfaces bem definidas. Cada nível esconde detalhes internos do nível inferior, permitindo que sistemas complexos sejam construídos de forma organizada.

Nível do Software de Aplicação:
Representa os programas que os usuários utilizam, como navegadores, editores de texto e jogos. Esses programas são escritos em linguagens de alto nível (Python, Java, C++), sem precisar conhecer detalhes da arquitetura do processador ou da memória.

Nível do Sistema Operacional:
Atua como intermediário entre os programas e o hardware, gerenciando recursos como memória, CPU, entrada e saída de dados. Implementa conceitos como abstração de arquivos, processos e drivers de dispositivos, permitindo que um mesmo software funcione em diferentes computadores.

Nível da Arquitetura do Conjunto de Instruções (ISA - Instruction Set Architecture):
Define como o software interage com o processador, incluindo os tipos de instruções suportadas, registradores e modos de endereçamento. Diferentes arquiteturas (como x86,

ARM e RISC-V) possuem conjuntos de instruções distintos, mas o software pode ser compilado para cada uma delas sem precisar reescrevê-lo do zero.

Nível da Microarquitetura:
Especifica como as instruções do ISA são executadas internamente pelo processador, incluindo unidades funcionais como ALU, registradores, barramentos e pipeline. Esse nível é invisível para o programador comum, mas essencial para otimizar a eficiência dos processadores modernos.

Nível Físico (Hardware):
Representa os componentes físicos que compõem o computador, como transistores, portas lógicas, memória e circuitos eletrônicos. Esses elementos são organizados para formar processadores, chips de memória, placas-mãe e outros dispositivos que executam os programas.

16.3 Importância da Abstração na Computação

Os níveis de abstração permitem que sistemas computacionais sejam desenvolvidos de forma modular e flexível. Isso traz várias vantagens:

- Interoperabilidade: Softwares podem rodar em diferentes hardwares sem modificações significativas.
- Independência da plataforma: Um sistema operacional pode rodar em vários processadores, desde que exista uma camada de abstração compatível.
- Facilidade de desenvolvimento: Programadores não precisam lidar com circuitos eletrônicos ao escrever um software, pois o sistema operacional e o compilador fazem essa ponte.
- Escalabilidade: Melhorias podem ser feitas em um nível sem afetar os outros, permitindo a evolução da computação sem exigir reescritas completas dos sistemas.

17 - Endereços, endereçamento e modelos de memória

17.1 Endereços

Em resumo ele se refere ao processo de atribuir identificadores únicos, chamados de endereços, a locais específicos na memória de um computador. Esses endereços são utilizados para acessar e manipular dados armazenados na memória.

Em computadores modernos com endereçamento por byte, cada endereço representa um byte distinto de armazenamento. Dados maiores que um byte podem residir em múltiplos bytes, ocupando uma sequência de bytes consecutivos.

17.2 Endereçamento

O endereçamento é um conceito que descreve como os operandos (dado ou valor sobre o qual uma operação é realizada em um programa de computador) são especificados nas instruções, ou seja, como o processador encontra os dados necessários para executar uma instrução.

Existem diversos modos de endereçamento, que determinam como o endereço dos dados é interpretado. Cada modo de endereçamento tem vantagens e desvantagens, dependendo do tipo de dado ou operação, influenciando a eficiência e a complexidade das instruções.

Endereçamento Imediato:
O operando é especificado diretamente na instrução. Não há necessidade de acessar a memória, tornando-o rápido, mas limitado a valores constantes pequenos.

Endereçamento Direto:
O endereço da memória é especificado diretamente na instrução. A localização na memória é fixa, sendo útil para variáveis globais cujo endereço é conhecido durante a compilação.

Endereçamento de Registrador:
Especifica um registrador diretamente. É o modo mais rápido, pois os registradores têm acesso muito rápido. Arquiteturas como ARM utilizam esse modo amplamente.

Endereçamento Indireto de Registrador:
O endereço de memória não está na instrução, mas em um registrador. Isso permite a utilização de diferentes endereços a cada execução da instrução, como em laços que percorrem um vetor.

Endereçamento Indexado:
Permite acessar palavras de memória com base no valor de um registrador somado a um deslocamento constante. Usado para acessar elementos de arrays e variáveis locais.

Endereçamento de Base Indexado:
Semelhante ao endereçamento indexado, mas usa dois registradores, um para a base e outro para o índice, além de um possível deslocamento. Esse método é útil em operações com arrays em várias máquinas.

Endereçamento de Pilha:
Utiliza a pilha para armazenamento temporário de dados, permitindo que instruções funcionem sem especificar endereços diretamente. Isso reduz o tamanho das instruções e é eficiente para operações como chamadas de função.

17.3 Modelos de Memória

Nos computadores, a memória é comumente organizada em células de endereços consecutivos, geralmente de 8 bits (byte). O byte é comum porque os caracteres ASCII usam 7 bits, se ajustando perfeitamente a essa estrutura. Em muitas arquiteturas, os bytes são agrupados em palavras de 4 (32 bits) ou 8 bytes (64 bits). Para que o computador acesse a memória de maneira mais eficiente, as palavras precisam começar em endereços específicos, como múltiplos de 4 ou 8, o que é chamado de “alinhamento de memória”.

Em muitos sistemas, há um único espaço de endereços para toda a memória (que vai de 0 até um valor máximo). Porém, em alguns sistemas, os endereços de instruções (como comandos de programação) e dados são separados, o que tem algumas vantagens: torna o sistema mais seguro e permite mais espaço para armazenar tanto programas quanto dados.

A “semântica da memória” é como as instruções de leitura e escrita na memória se comportam. Em sistemas mais complexos, as instruções podem ser reordenadas (executadas em uma ordem diferente da esperada), o que pode causar problemas, especialmente em sistemas com múltiplos processadores. Para resolver isso, os sistemas

podem garantir que todas as operações de memória ocorram em uma ordem específica, o que pode afetar o desempenho.

Em resumo, a organização da memória e o controle sobre suas operações envolvem desafios relacionados ao desempenho e à consistência, com técnicas variadas para atender a diferentes necessidades de sistemas e arquiteturas.

18 - Fluxo de controle

18.1 Conceito

O fluxo de controle de um programa se refere à sequência em que as instruções são executadas durante sua execução. Em um fluxo de controle linear, as instruções são executadas em ordem sequencial, ou seja, uma após a outra, em locais consecutivos de memória. No entanto, é comum que o fluxo de controle seja alterado por diferentes estruturas ou mecanismos presentes no programa. Essas alterações permitem que o programa reaja a condições específicas, execute tarefas em paralelo ou trate eventos inesperados.

A alteração do fluxo de controle pode ser causada por vários fatores, como chamadas de procedimento, corrotinas, exceções e interrupções, que têm a capacidade de interromper o processo atual e desviar a execução para outra parte do código, retornando posteriormente ao fluxo original.

18.2 Chamadas de procedimento

Uma chamada de procedimento é um mecanismo que permite que o programa interrompa sua execução no ponto em que está e transfira o controle para uma função ou procedimento específico. O procedimento executa suas instruções e, ao ser concluído, o controle retorna ao ponto de onde o procedimento foi chamado, continuando a execução do programa a partir daí. Esse mecanismo de alteração do fluxo de controle é fundamental para a modularização e reutilização de código.

As chamadas de procedimento são úteis para dividir um programa em partes menores e mais manejáveis, facilitando a leitura e manutenção do código. Em linguagens de programação, essas chamadas são realizadas por meio de palavras-chave como call ou invoke, e o retorno ao ponto original é feito com o comando return.

18.3 Corrotinas

Corrotinas são uma forma mais flexível de manipulação do fluxo de controle, permitindo que a execução do programa seja intercalada entre diferentes partes do código, sem a necessidade de retornar diretamente ao ponto inicial. Enquanto procedimentos tradicionais exigem um retorno explícito, as corrotinas podem "pausar" sua execução e retornar posteriormente, no momento adequado.

As corrotinas são frequentemente utilizadas para simular processos paralelos ou concorrentes, permitindo que diferentes tarefas sejam executadas de forma cooperativa. Em sistemas onde a execução paralela não é completamente suportada, as corrotinas podem ser usadas para dividir o tempo de execução entre várias tarefas, tornando o programa mais eficiente e responsivo.

18.4 Exceções e interrupções

Exceções e interrupções são eventos que causam alterações no fluxo de controle quando ocorrem condições especiais ou anormais.

Exceções:

São geradas por eventos inesperados durante a execução do programa, como erros de lógica ou operações inválidas (exemplo: divisão por zero). Quando uma exceção ocorre, o fluxo de controle é desviado para uma rotina de tratamento de exceções. Caso a exceção não seja tratada, o programa pode ser encerrado. O tratamento de exceções é uma técnica fundamental para aumentar a robustez de um programa, garantindo que ele consiga lidar com erros de maneira controlada.

Interrupções:

Diferentemente das exceções, que são geradas pelo próprio programa, as interrupções são sinais enviados pelo hardware ou pelo sistema operacional para chamar a atenção do processador. Por exemplo, a chegada de dados em uma porta de entrada ou a necessidade de realizar uma tarefa de manutenção pode gerar uma interrupção. Quando uma interrupção ocorre, o fluxo de controle do programa em execução é interrompido, e o processador passa a executar um código específico para lidar com a interrupção. Após o processamento da interrupção, o controle retorna ao ponto do programa que foi interrompido.