Arquitetura de Computadores: por trás de como seu programa funciona

O código de máquina

Uma **Linguagem de Máquina** é a linguagem de mais baixo nível de entendimento pelo ser humano, e a única entendida pelo processador (UCP). É constituída inteiramente de números, o que torna praticamente impossível entendê-la diretamente.

As **Linguagens de Programação de Alto Nível** são as linguagens de programação que possuem uma estrutura e palavras-chave que são mais próximas da linguagem humana. Tornando os programas mais fáceis de serem lidos e escritos. Esta é a sua principal vantagem sobre as linguagens de nível mais baixo. Os programas escritos nessas linguagens são convertidos para a linguagem de baixo nível através de um programa denominado compilador ou de um interpretador. Exemplos: JavaScript, PHP, Java.

- *Pontos Positivos*

- Baixo custo operacional;
- Facilidade de aprendizagem e entendimento;
- Manutenção simplificada;
- Produtividade.

- *Pontos Negativos*

- Maior ocupação de espaço em comparação às Linguagens de Baixo Nível;
- Maior tempo de processamento.

As **Linguagens de Programação de baixo nível** são aquelas que estão mais próximas da **Linguagem de Máquina**. Estão voltadas ao dispositivo e bem mais próximas da simbologia de máquina, possuindo, assim, o objetivo de se comunicar com o computador de maneira rápida e eficaz. Exemplo: Assembly.

- *Pontos Positivos*

- Melhor aproveitamento da arquitetura do computador;
- Tempo de processamento mais rápido em comparação às Linguagens de Alto Nível.

- *Pontos Negativos*

- Difícil compreensão e demorada entendimento da sintaxe;
- Necessidade de conhecer profundamente o hardware da máquina.

Bits e Bytes

Em um computador, cada um desses 0s e 1s é chamado de **bit**, que vem do inglês Binary Digit ou dígito binário. Essa é a menor unidade possível para o armazenamento de informação. Como bit é uma unidade muito pequena, costuma-se trabalhar com grupos de 8 bits, essa quantidade agrupada de bits é chamada de byte. Por convenção, quando medimos o armazenamento em um computador usamos o byte.

Para representar essas unidades, utilizamos b (“b” minúsculo) para o bit e B (“B” maiúsculo) para byte. Assim, **1B = 8b**. A diferença entre maiúsculas e minúsculas pode passar despercebida, mas é algo bem importante!

Compiladores Vs Interpretadores

Antes de entender os conceitos de *Compilador* e *Interpretador* é necessário entender a função de um tradutor, que é aceitar um conjunto de instruções escritas em uma linguagem de programação de alto nível - que é independente da máquina - e fazer com que as atividades especificadas por estas instruções sejam executadas pelo computador.

Um **Compilador** é um traduto que transforma automaticamente uma linguagem de programação de alto nível para alguma outra forma que viabilize sua execução. O código executável gerado pelo compilador é dependente do sistema operacional e da linguagem de máquina para o qual o código fonte foi traduzido. A enorme variedade de compiladores existentes é bem vinda, visto que existem milhares de linguagens fonte, e as linguagens alvo são também muito variadas. O processo de compilação pode ser dividido em duas etapas:

- *Análise:* parte o programa fonte em peças constituintes e cria uma representação intermédia do programa fonte;
  - *Análise Léxica (ou Linear):* a cadeia de caracteres que forma a estrutura do programa fonte é lida da esquerda para a direita e agrupada em tokens que são sequências de caracteres com sentido coletivo. A sua função básica é o reconhecimento e a classificação das estruturas elementares ou classes sintáticas das linguagens.
  - *Análise Sintática (ou Hierárquica):* caracteres e tokens são agrupados hierarquicamente em coleções aninhadas com sentido coletivo. Verifica se a estrutura geral do texto ou programa fonte está correta;
  - *Análise Semântica:* são executadas certas paragens para assegurar que os componentes de um programa são juntamente ajustados em sentido absoluto. Verifica se o programa fonte tem erros semânticos e reúne informação dos tipos para a fase de gerador de código subsequente.
- *Síntese:* constrói o desejado programa alvo (código de máquina) a partir da representação intermédia.

O funcionamento dos **Interpretadores** é bem similar ao dos compiladores. O interpretador traduz o código linha a linha, o que significa que o código-fonte não é totalmente traduzido antes de ser executado. Além disso, não existem fases distintas nem se produz código intermediário.

O **Compilador Just In Time** traduz inicialmente para uma linguagem intermediária (bytecode). Então compila a linguagem intermediária dos subprogramas em código de máquina nativa quando eles são chamados (tempo de execução) para executar na máquina nativa. Este código é mantido para chamada subsequentes.

	Vantagens	Desvantagens
Compiladores	Execução mais rápida • Permite estruturas de programação mais completas • Permite a optimização do código fonte	Várias etapas de tradução • Programa final é maior, necessitando mais memória para a sua execução • Processo de correção de erros e depuração é mais demorado
Interpretadores	Depuração do programa é mais simples • Consome menos memória • Resultado imediato do programa ou rotina desenvolvida	Execução do programa é mais lenta • Estruturas de dados demasiado simples • Necessário fornecer o programa fonte ao utilizador

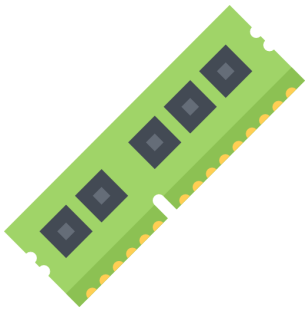
⭐ Questão 1

William terminou de programar o código fonte de um jogo e, para executá-lo, precisou passar esse código por um tradutor. Ele observou que após esse processo foi gerado um novo arquivo que pode ser executado diretamente pelo computador. Sobre esse processo de tradução, marque as alternativas corretas:

Respostas:

- O novo arquivo gerado é o jogo em código de máquina.
- Como o código foi traduzido de uma vez só para um executável, o tradutor usado foi um compilador.

🔹 Memória RAM



A memória **Random Access Memory** permite a leitura e a escrita de arquivos, ou seja, sua função é possibilitar que o processador tenha acesso imediato aos dados que deseja, contribuindo para uma maior rapidez e capacidade de resposta das solicitações. Ao executar um app ou game, ele não é carregado na RAM. Enquanto os dados de uma aplicativo ainda estiverem na memória, é possível voltar de onde parou sem ter que carregá-lo novamente. É pois isso que a memória RAM é importante para multitarefas. Os aplicativos carregados ficam lá até que sua RAM fique cheia e precise limpar alguns dados para conseguir abrir espaço para outros. Ao contrário da memória de armazenamento, ela não guarda o conteúdo de forma permanente, já que é usada para processar os dados que estão sendo usados naquele momento.

Memória Secundária

Atualmente, existem 3 tipos de unidade de armazenamento de dados que podem ser instalados de forma permanente no computador: discos rígidos, unidades SSD e unidades ópticas. É importante não confundir armazenamento com memória. Apesar de módulos de memória (RAM) e unidades de armazenmento como discos rígidos e SSDs serem rotulados com a capacidade de bytes que conseguem armazenar, o princípio de funcionamento e a finalidade são completamente diferentes. Apesar de unidades SSD utilizarem chips de memória do tipo falsh em seu interior (que são diferentes dos chips usados em módulos de memória), elas são classificadas como "dispositivos de armazenamento" e não como "memórias".

Hard Disk Drive (HDD)

É o dispositivo de armazenamento de dados mais utilizado nos computadores. Esse tipo de equipamento de dados mais utilizado nos computadores. Esse tipo de equipamento guarda desde os arquivos pessoais até informações utilizadas exclusivamente pelo sistema operacional.

Solid State Drive (SSD)

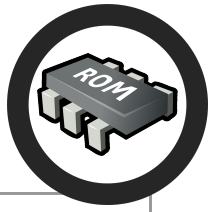
É um componente de Hardware que substitui o HD (Hard Disk) como unidade de armazenamento de dados. Sendo muito mais rápido, o SSD não possui discos físicos ou agulhas magnéticas, sendo capaz de acessar dados em uma fração de segundo e tornar o computador mais ágil para abrir programas e executar tarefas.

Como não possuem partes móveis, os SSDs resolvem a limitação de velocidade dos HDs, que utilizam cabeças de leitura e gravação para fazer operações em discos magnéticos.

HDD	SSD
mais lento	mais rápido
vida útil maior	vida útil menor
mais barato	mais caro
usabilidade mecânica (movimentação de partes)	não-mecânico (flash)
mais fragilidade	mais resistente
melhor para armazenar conteúdos como filmes, fotos e documentos	melhor para armazenar operações do sistema, jogos e arquivos usados com frequência

Memória ROM

A memória **Read-Only Memory** é um tipo de memória não-volátil com baixo armazenamento, bem diferente da RAM ou HDD/SSD. Uma ROM possui vários modelos, mas todos são projetados para serem lidos, ou seja, não é esperado que o usuário escreva informações nessa memória.



> Um UDD ou SSD não são ROMs!

Dessa forma, quando o computador é inicializado, as primeiras informações que ele irá buscar estarão na ROM, por exemplo, a BIOS (Basic Input/Output System). Elas vão ajudá-lo a identificar os dispositivos conectados, como o HD, e carregar o sistema operacional.

⭐ Questão 2

Letícia quer investir em um novo computador. Ela usualmente possui poucos arquivos e programas instalados, mas usa muitos deles todos de uma vez. Pensando nessa situação, quais são conclusões corretas para a escolha do novo computador?

Respostas:

- Como Letícia usa vários programas ao mesmo tempo, investir em uma memória RAM com mais capacidade é importante.
- Mesmo com poucos programas, ainda é possível abrir eles mais de uma vez ao mesmo tempo. Por exemplo, podemos abrir várias janelas de um mesmo navegador ou o mesmo editor de texto para documentos diferentes. Dessa forma, o consumo de RAM será bem alto.

🔹 Processador

A **Central Processing Unit (CPU)**, também chamada de **Processador**, é responsável por executar instruções que permitem manipular os dados na memória do computador através de operações aritméticas e lógicas, como somas, subtrações, comparações entre números, etc. A CPU lê as instruções da memória do computador e pula de uma instrução para a seguinte. Também há instruções que podem mudar qual será a próxima instrução a ser lida pelo processador, tais instruções permitem alterar o fluxo do processamento.

- **Registradores:** armazenam pequenas quantidades de informação necessárias durante a execução, como a instrução recebida, a posição da RAM onde foi buscada e os valores intermediários das operações.
- **Unidade de Controle:** decodifica as instruções recebidas, prepara a ULA para realizar a operação e o ponteiro da RAM de onde se extrairá o valor a ser usado na operação.
- **Unidade Lógica Aritmética:** executa as intruções decodificadas com o apoio dos registradores e da unidade de controle.
- **Clock:** determina o ritmo/velocidade das iterações { 'busca' → 'decodifica' → 'executa' }.

⭐ Questão 3

Um processador é composto de várias partes com funções bem específicas. O que é correto dizer sobre o funcionamento do processador?

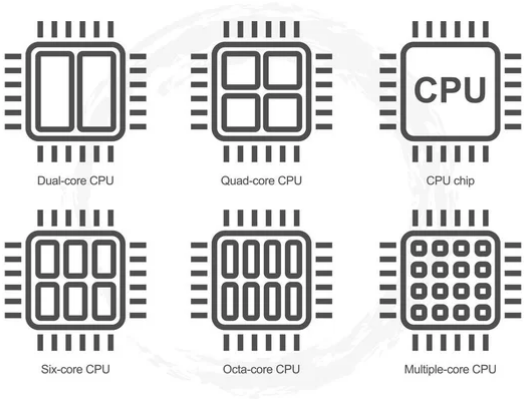
Respostas:

- As instruções são processadas num ciclo de três passos chamados buscar, decodificar e executar.
- O clock define a velocidade com que as instruções são executadas. Ou seja, quanto maior a velocidade do clock, mais rápido um programa será executado.
- Antes de serem executadas, as instruções precisam ser decodificadas pela Unidade de Controle (UC).

Processadores Modernos e Lei de Moore

O projeto do processador vem recebendo constantes melhorias visando maior desempenho: pipelining de instruções, processador superescalar, multicore e etc. **Pipelining** se aseemelha a uma linha de montagem (assembly line), sendo que a execução de uma tarefa completa é dividida em estágios. Há uma estação separada para a execução de cada estágio.

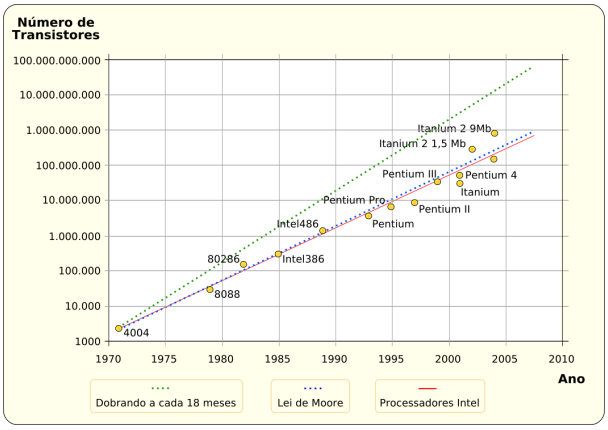
O pipelining possibilita a execução de diferentes estágios de várias tarefas ao mesmo tempo. Quando uma estação termina de executar o estágio de uma tarefa, ela passa a executar o mesmo estágio, mas da tarefa seguinte (Anteriormente: 'busca' → 'decodificação' → 'execução' Agora: quando uma busca termina, já começa outra, sem aguardar o ciclo acabar). A primeira tarefa leva o tempo normal para ser concluída. Mas a partir daí, uma nova tarefa é concluída logo após o seu último estágio.



Lei de Moore

Em 1965, Gordon Moore previu que o número de componentes de um circuito integrado (transistores) dobraria a cada período de 18 meses, resultando num processador duas vezes mais rápido. Isso originou a chamada Lei de Moore. Essa lei virou uma medida econômica das empresas e é responsável pelo crescimento exponencial de performance da tecnologia que vemos nas últimas décadas.

Entretanto, nos últimos anos, os fabricantes de processadores estão encontrando limites físicos para manter esse padrão e o crescimento está cada vez diminuindo mais. Por isso, pesquisadores procuram outras soluções para manter o crescimento da performance de um processador, mas sem precisar aumentar o número de transistores ou a velocidade de clock.



🔹 Dispositivos de Entrada e Saída (I/O)

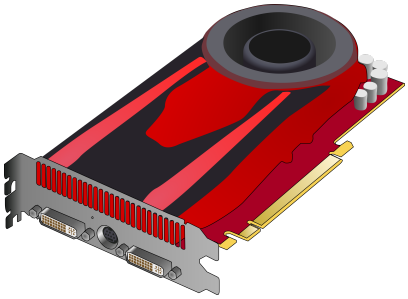
Os dispositivos de entrada são: teclado, mouses, scanners, câmeras de vídeo, microfones e etc. As funções desses dispositivos são coletar informações e introduzir as informações na máquina, converter informações do homem para a máquina e vice-versa, e recuperar informações dos dispositivos de armazenamento.

Já os dispositivos de saída são: impressoras, monitores de vídeo, saídas de áudio e etc. As funções desses dispositivos são exibir ou imprimir os resultados do processamento, ou ainda controlar dispositivos externos.

A UCP não se comunica diretamente com cada dispositivo de I/O e sim com interfaces, de forma a compatibilizar as diferenças características. O processo de comunicação (ou protocolo) é feito através de transferência de informações de controle, endereços e dados propriamente ditos. Inicialmente, a UCP interroga o dispositivo, enviando o endereço do dispositivo e um sinal dizendo se quer mandar ou receber dados através da interface. O periférico, reconhecendo seu endereço, responde quando está pronto para receber (ou enviar) dados.

A UCP então transfere (ou recebe) os dados através da interface, e o dispositivo responde confirmando que recebeu (ou transferiu) os dados (acknowledge ou ACK) ou que não recebeu os dados, neste caso solicitando retransmissçao (not-acknowledge ou NAK). A compatibilização de velocidades é feita geralmente por programa, usando memórias temporárias na interface chamadas "buffers" que armazenam as informações conforme vão chegando na UCP e as libera para o dispositivo à medida que este as pode receber.

Placa de Vídeo



Uma Graphics Processing Unit (GPU) é a unidade responsável pela renderização dos pixels na tela do computador e está presente em todos os computadores pessoais. A GPU pode variar de "gráficos integrados" simples, que fazem parte da placa-mãe ou do processador, a placas de expansão maiores e mais poderosas.

Essas placas de expansão - geralmente chamadas de placas dedicadas - são capazes de executar tarefas mais poderosas do que as placas gráficas integradas, como gráficos de jogos, renderização acelerada de vídeo ou até mesmo trabalhos não-gráficos, como mineração de bitcoin. O uso de uma placa gráfica dedicada custa mais consumo de energia, mais calor e mais espaço no computador, e é por isso que raramente elas são utilizadas em ultrabooks e outros notebooks finos.

⭐ Questão 4

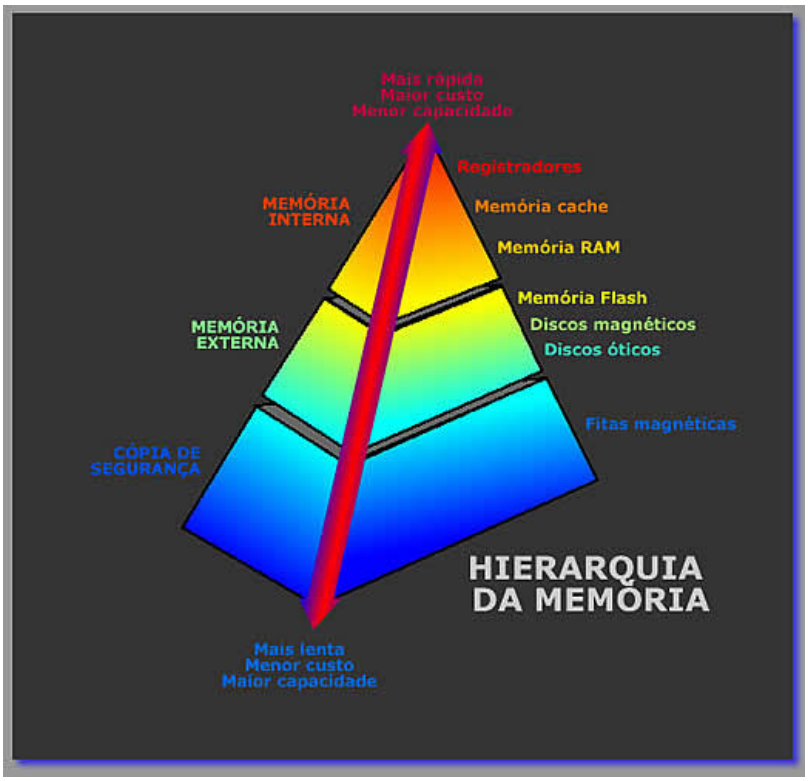
Nesta aula, vimos diferentes tipos de dispositivos de entrada e saída e o funcionamento de alguns deles. O que é correto dizer sobre esses dispositivos?

Respostas:

- Os drivers de dispositivo permitem escrevermos o mesmo código para diversos dispositivos diferentes.
- Como telas atuais possuem milhões de pixels, computadores pessoais precisam de placas de vídeo (GPUs) para renderizar eles a tempo para quem está usando.

Memória Cache e Hierarquia de Memória

Há vários tipos de memórias, cada uma com características distintas em relação ao custo (preço por bit), capacidade de armazenamento e velocidade de acesso. A memória ideal seria aquela que é mais barata, de grande capacidade e de rápido acesso. Porém, a memória rápida é custosa e de pequena capacidade, ao passo que a memória de grande capacidade apresenta baixa velocidade de acesso.



Um dado (ou instrução) no disco pode ter uma cópia na memória e no processador (num registrador ou na memória cache). Quando o processador precisa de um dado, ele pode já estar na cache. Se não, tem que buscar na memória. Quando um dado é acessado na memória, um bloco inteiro contendo o dado é trazido à memória cache. Blocos vizinhos podem também ser trazidos à memória cache para possível uso futuro. Na próxima vez o dado (ou algum dado vizinho) é usado, já está no cache cujo acesso é rápido.

Fenômenos de Localidade

- Localidade Temporal: um dado ou instrução acessado recentemente tem maior probabilidade de ser acessado novamente, do que um dado ou instrução acessado há mais tempo.
- Localidade Espacial: se um dado ou instrução é acessado recentemente, há uma probabilidade grande de acesso a dados ou instruções próximas.

⭐ Questão 5

Nesta aula, vimos que o computador acessa dados a partir de diferentes tipos de memória, cada uma com sua particularidade. Por que precisamos desses vários tipos de memória?

Respostas:

- Como o princípio da localidade diz que nós usamos a mesma porção de dados durante um período de tempo, é importante ter memórias rápidas para poucos dados e lentas para muitos dados.
- Memórias rápidas são muito caras, então não é viável usá-las para armazenar uma grande quantidade de dados.

🔹 Tipos de Codificação

UTF-8

Essa codificação representa os caracteres em pedaços de 8 bits e sua grande vantagem é manter os textos codificados apenas em ASCII (a grande maioria da Web no passado) intactos. Ele tem um tamanho variável e cada caractere pode ocupar 8, 16, 24 ou 32 bits. Essa é a codificação mais comum na web hoje.

UTF-16

Essa codificação representa os caracteres em pedaços de 16 bits. Isso significa que a codificação não é mais compatível com ASCII e ocupa o dobro de memória em textos que possuem apenas caracteres da língua inglesa. Ele tem um tamanho variável e cada caractere pode ocupar 16 ou 32 bits.

Sua grande vantagem é ocupar menos espaço quando o texto possui muitos caracteres asiáticos (UTF-8 usaria 3 bytes por caractere e UTF-16 apenas 2). Mesmo assim, essa vantagem é bem questionada (https://utf8everywhere.org/) e muitas pessoas não recomendam o seu uso em muitos casos. Essa é a codificação usada em sistemas Windows.

UTF-32

Essa codificação representa os caracteres em pedaços de 32 bits. Ela não é compatível com ASCII e ocupa 4 vezes mais espaço em textos que só utilizavam ASCII. Mesmo assim, diferentemente do UTF-8 e UTF-16, essa codificação tem um tamanho fixo e essa é sua grande vantagem. Como cada caractere ocupa a mesma quantidade de bits, é fácil saber em qual posição cada caractere está em um texto (é só pegar o índice do caractere e multiplicar por 32).

⭐ Questão 6

Stefany vai fazer aniversário e, para isso, ela decidiu criar um programa em JavaScript que mexe com idades. Primeiro, ela fez uma função calculaProximaIdade(), que recebe a idade que ela tem agora e imprime quantos anos ela terá depois do aniversário. Em seguida, ela criou a função calculaProximasIdades(), que recebe a lista de idades dela e de seus amigos e devolve quantos anos todos terão ao final do ano. Por fim, ela fez uma função calculaIdadesDaqui5Anos, que recebe a mesma lista de antes mas devolve as idades que todos terão daqui cinco anos. O programa ficou da seguinte forma:

```
function calculaProximaIdade(idade) {
    idade += 1;
    console.log(idade);
}

function calculaProximasIdades(idades) {
    for (let i = 0; i < idades.length; i += 1) {
        idades[i] += 1;
    }
    console.log(idades);
}

function calculaIdadesDaqui5Anos(idades) {
     for (let i = 0; i < idades.length; i += 1) {
        idades[i] += 5;
    }
    console.log(idades);
}

const idadeStefany = 21;
calculaProximaIdade(idadeStefany);

const idadesAmigos = [idadeStefany, 20, 23, 18, 7];
calculaProximasIdades(idadesAmigos);

calculaIdadesDaqui5Anos(idadesAmigos);
```

O programa de Stefany se comporta como o esperado?

> Orientação da Alura: Listas são passadas por referência para funções. Dessa forma, se modificamos seu conteúdo dentro da função, a lista é modificada. Por outro lado, isso não acontece com número, que são passados por valor.