Introdução a computação

Noções Fundamentais de Computação

Alexandre L. M. Levada

Sumário

- Breve Histórico

- Conceituação de Arquitetura
  - Modelo de Von Neumann
    - CPU
    - Memória
    - E/S
  - Aspectos de Sistemas Operacionais
    - Hardware x Software

- Aspectos de Linguagens de Programação
  - Compiladores
  - Linguagens de Alto Nível
  - Linguagens de Máquina
  - Vantagens e Desvantagens

Histórico

A história da computação pode ser dividida em 3 grandes eras.

- A era dos dispositivos mecânicos (500 a.C. - 1880)
- A era dos dispositivos eletromecânicos (1880 - 1930)
- A era dos componentes eletrônicos (1930 -)
  - Primeira geração: Computadores a Válvula
  - Segunda Geração: Computadores Transistorizados
  - Terceira Geração: Computadores com Circuitos Integrados
  - Quarta Geração: Microcomputadores

Apenas na última grande era o padrão de arquitetura de computadores como conhecemos hoje foi desenvolvido.

- Antes, dados e programas eram armazenados externamente
- Exemplo: Cartões perfurados
- Principais desvantagens:
  - Pouco confiável
  - Lento e ineficaz
  - Organização ficava por conta do operador
  - Interface pobre

O Modelo de Von Neumann

Em 1946, John Von Neumann, um matemático húngaro, revolucionou a computação ao postular 5 princípios básicos, que nortearam o desenvolvimento de novos computadores.

- Princípios Básicos
  - Execução completamente eletrônica
  - Sistema de numeração padronizado e binário
  - Uma memória interna (armazenar dados)
  - Um programa armazenado (para acessar e manipular os dados)
  - Universalidade (uma máquina que pudesse realizar mais que uma única tarefa, ou seja, que pudesse ser programada)

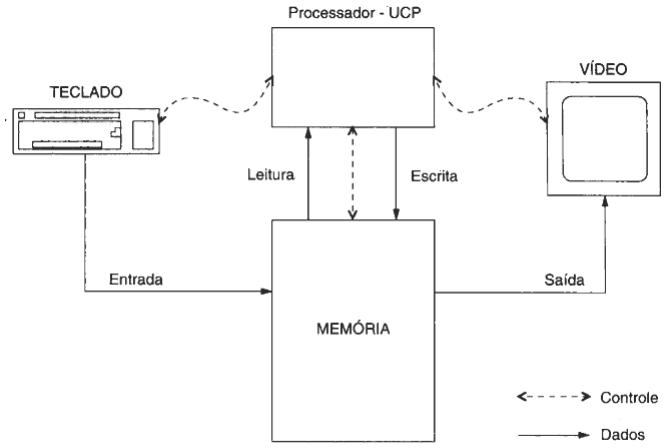
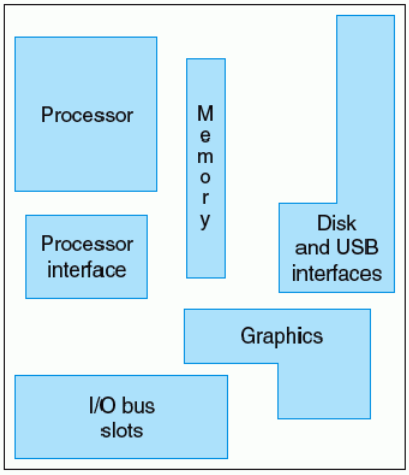


Figura 2.2 Componentes de um sistema de computação.

Organização de um computador



Sistemas de Numeração

Aspecto fundamental para o sucesso da computação.

Base Decimal: Algarismos (de 0 a 9) assumem um valor diferente (múltiplo de 10) dependendo da posição.
$2562_{10} = 2 \times 10^3 + 5 \times 10^2 + 6 \times 10^1 + 2 \times 10^0$

Base Binária: Algarismos (0 e 1) assumem um valor diferente (múltiplo de 2) dependendo da posição.
$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

- Conversão binário para decimal
- Conversão decimal para binário

Base Hexadecimal: Algarismos (de 0 a F) assumem um valor diferente (múltiplo de 16) dependendo da posição.
$1A7B_{16} = 1 \times 16^3 + A \times 16^2 + 7 \times 16^1 + B \times 16^0$

$= 1 \times 16^3 + 10 \times 16^2 + 7 \times 16^1 + 11 \times 16^0$

$= 4096 + 2560 + 112 + 11 = 6779_{10}$

Conversão de Binário para Hexadecimal: Basta, da direita para a esquerda, agrupar 4 bits e substituir pelo respectivo valor hexadecimal
$101111011101_2 = BDD_{16}$ pois

$1101_2 = D_{16}$, $1101_2 = D_{16}$ e $1011_2 = B_{16}$

Tabela com representação numéricas em diferentes bases

Base 2	Base 8	Base 10	Base 16
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F
10000	20	16	10
10001	21	17	11

Organização da Informação no Computador

Definições básicas:

- Bit: é a unidade básica da informação em um computador. Um bit pode assumir apenas um de dois valores: 0 ou 1
- Byte: é um conjunto de 8 bits, por exemplo, 01101110.
- Palavra: é a menor subdivisão da memória de um computador, sendo que cada palavra possui um endereço de memória. O tamanho da palavra é determinado pelo número de bits (i.e., 16, 32, 64 bits).
- Memória: é onde são armazenados os dados e as instruções de um programa que está sendo executado.

A memória de um computador compara-se a um armário com várias gavetas. Cada uma possui uma posição específica dentro do armário (endereço único), o que permite rápida localização. Cada gaveta armazena alguma coisa (dados ou instruções). Há gavetas para dados e há gavetas com instruções (como por exemplo, *mova o conteúdo da gaveta X para a gaveta Y*). O tamanho máximo do dado ou da instrução que cada gaveta pode armazenar é definido pelo **tamanho da palavra**.

Endereço binário	Palavra			
→	00000	00001	00010	00011
	00101010	01001100	00101110	01101011
	00100	00101	00110	00111
	01001110	00101100	10001101	01001100
	01000	01001	01010	01011
	11000010	11100000	00000011	00000100
	01100	01101	01110	01111
	00000000	00000001	00000000	00000000

Endereço da Instrução (em binário)	Código de Operação	Operando
00000	001	01010
00001	010	01100
00010	001	01110
00011	011	01011
00100	010	01110
00101	001	01100
00110	100	01101
00111	010	01100
01000	110	00010
01001	111	00000

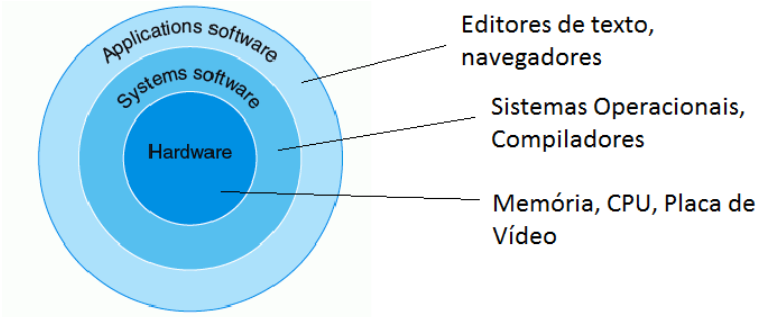
Interface Hardware/Software

- Como gerenciar tudo isso ?
- Como programar nesse esquema de gavetas ?
- Precisamos escrever tudo com 0's e 1's ?
- Precisamos saber o local exato de cada dado na memória ?

Apesar do hardware executar apenas instruções simples de baixo nível, dois componentes são fundamentais para o desenvolvimento de aplicações:

- Sistemas Operacionais
- Compliadores

Modelos de camadas hierárquicas



O Sistema Operacional

É um tipo de software vital pois trata-se da principal interface entre o hardware e o software. (Windows, Linux, Mac OS,...)

Um S.O. realiza duas funções básicas:

- Atuar como uma máquina virtual
  - Encapsular todo hardware do programador de modo a oferecer uma interface simples e acessível aos recursos da máquina
- Atuar como um gerenciador de recursos
  - Compartilhamento de recursos em tempo e espaço
  - Memória é limitada mas diversos programas precisam ser executados
  - Mediador de conflitos por recursos

Linguagens de Programação de Alto Nível

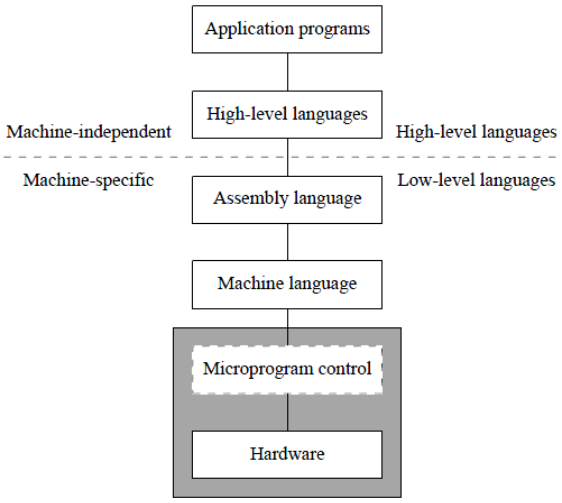
Desempenham um papel fundamental na programação de computadores, pois trazem uma série de benefícios, como:

- Desenvolvimento de programas torna-se muito mais rápido (codificação mais fácil)
- Facilita a manutenção do software
- Aumentam a portabilidade (independência do código de máquina)

Mas, se programas são simples conjuntos de instruções para o computador, então porque simplesmente não utilizamos o português ou inglês para escrevê-los?

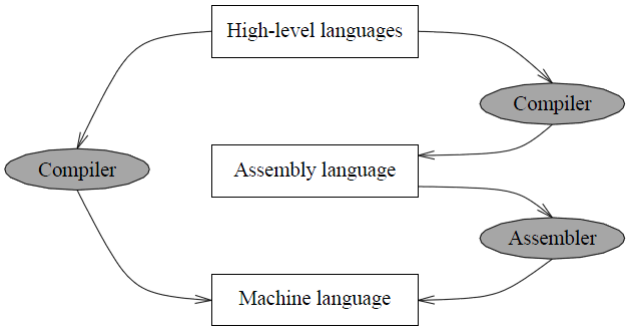
Falta de rigidez sintática e semântica: Ambiguidade!

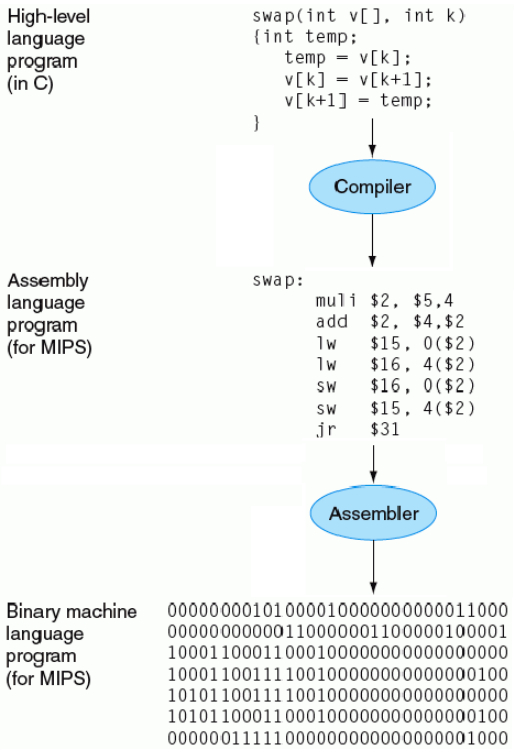
Solução: Linguagens Formais



O Compilador

É o programa que faz a tradução de uma linguagem de alto nível (C/C++, Pascal, Java) para o código de máquina ou assembly (linguagem de baixo nível dependente do processador e que define mneumônicos para os códigos binários)





A Linguagem Python

- Trata-se de uma linguagem de alto nível independente de plataforma (alta portabilidade)
- Programas escritos em Python rodam em qualquer S.O. (windows, linux, MacOS,...). Porque ?
- A linguagem é interpretada e não compilada.
- Vantagem: o mesmo código roda em qualquer sistema operacional!
- Desvantagem: execução mais lenta.

Porque é mais lento? O interpretador Python traduz a linguagem de alto nível para código de máquina antes da execução de cada script em tempo real, ou seja a tradução é feita em tempo real. Não há geração de um arquivo executável binário empacotando o código de máquina.

Referências

MONTEIRO, M. A. **Introdução à Organização de Computadores**, LTC Editora, 4 ed., 2002.

DANDAMUDI, S. P. **Fundamentals of Computer Organization and Design**, Springer, 2003.

PATTERSON, D. A., HENNESSY, J. L. **Computer Organization and Design**, Morgan Kaufmann, 3 ed., 2005.

TANENBAUM, A., **Sistemas Operacionais Modernos**, Prentice-Hall, 2 ed., 2003.

MEDINA, M., FERTIG, C. **Algoritmos e Programação: Teoria e Prática**, Novatec, 2 ed., 2006.

No mundo só existem 10 tipos de pessoas, as que conhecem binário e as que não conhecem.