

Capítulo 1 - Conceitos Básicos

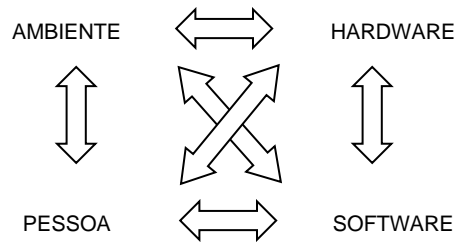
Objetivos

- Introduzir os conceitos básicos de processamento de dados;
- Apresentar um modelo e componentes lógicos de um computador;
- Mostrar o princípio de funcionamento de um computador.

Tipos de computador

- Computadores digitais
 - representam informações através de grandezas discretas;
 - maior precisão e maior capacidade de armazenamento de dados.
- Computadores analógicos
 - representam informações através de grandezas contínuas;
 - maior velocidade.
- Computadores híbridos
 - reúnem as características dos dois tipos acima.

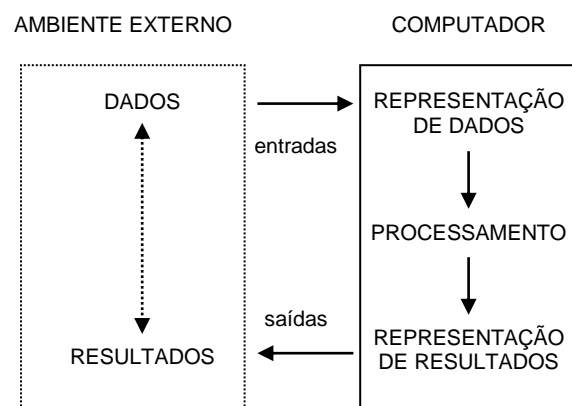
Sistema computacional



- **HARDWARE:** conjunto de componentes eletro-eletrônicos e/ou eletro-mecânicos com os quais são construídos os computadores e seus periféricos;
- **SOFTWARE:** conjunto de programas e sua documentação.

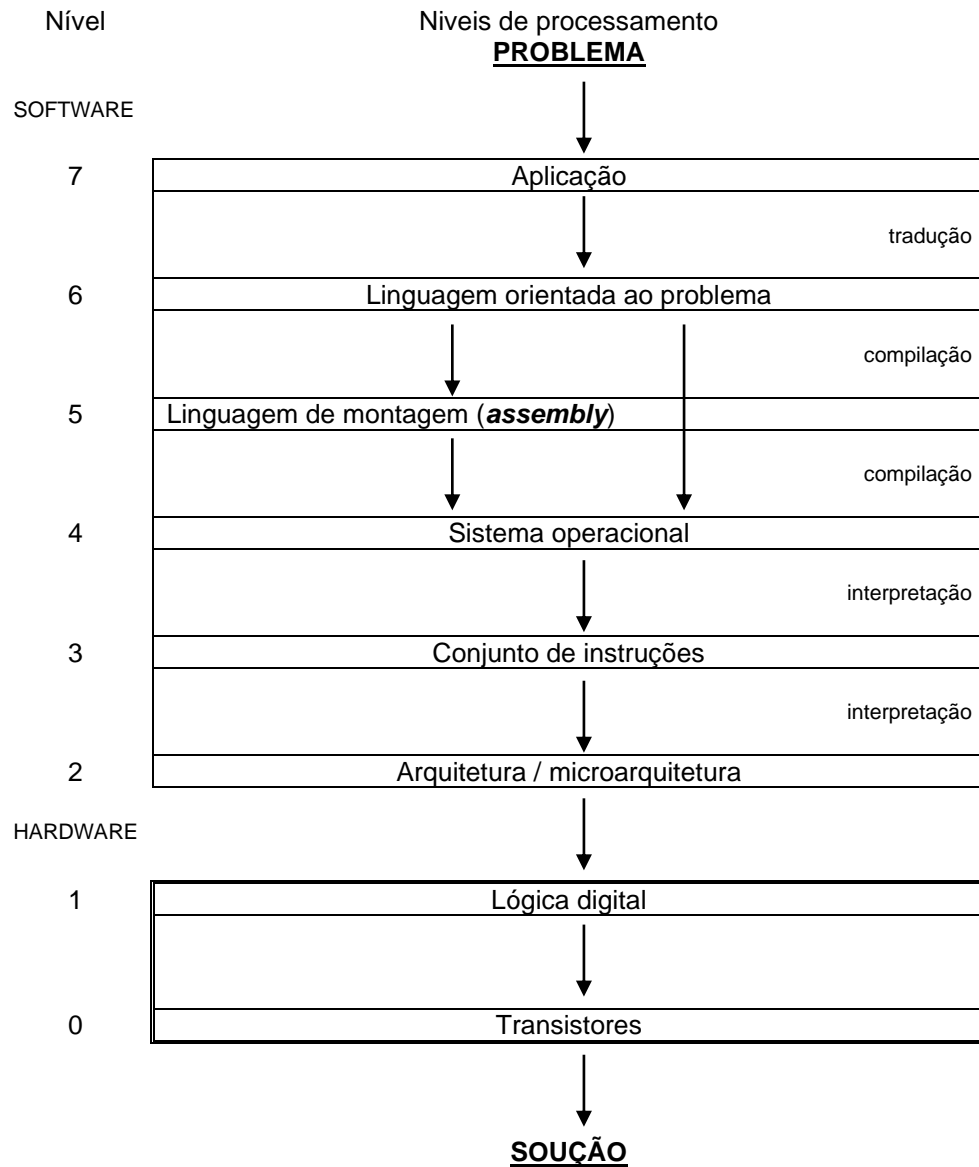
Processamento de dados

Os conceitos básicos de processamento de dados são os de codificar, armazenar e transformar de informação.



Modelo para o processamento de dados

O processamento de dados pode ser organizado em níveis para melhor se situar os componentes e ações envolvidas.



Funcionamento do computador

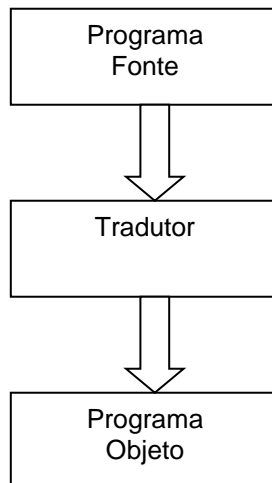
O computador, seguindo instruções colocadas em sua memória, pode receber dados (por meio de unidades de entrada, ligadas às interfaces de entrada); armazená-los em sua memória principal; transformá-los através de operações aritméticas e decisões lógicas, e colocar os resultados obtidos em unidades de saída de dados (ligadas às interfaces de saída).

As ligações entre unidades são feitas através das vias (ou *barramento*), por onde passam as informações relativas ao conteúdo (*dados*) ou de sua localização (*endereços*); e da via de controle, por onde passam as informações relativas ao comando das unidades.

A sequência de instruções colocada na memória do computador que serve para manipular os dados é denominada *programa*, que é a tradução de um algoritmo mediante uso de uma linguagem de programação.

Conceitos importantes

- *BIT* (“*Binary digiT*” - dígito binário)
 - unidade de informação tem somente pode receber os valores "0" ou "1";
- *BYTE* (“*Binary Term*” - termo binário)
 - conjunto de bits que serve para representar os números, as letras, os sinais de pontuação etc.;
- Palavra (*WORD*)
 - conjunto de bytes que pode ser tratado como uma unidade
- Algoritmo
 - conjunto de instruções que atendem a um objetivo definido
- Linguagens de programação
 - Linguagem de máquina
 - Linguagens simbólicas
 - Linguagem de montagem (ASSEMBLY – ou de máquina)
 - Linguagem algorítmica (programador)
- Tradutor (compilador/interpretador)
 - programa que traduz outro, escrito em uma linguagem de programação (*programa fonte*), para um terceiro, escrito em linguagem de máquina, ou outra qualquer (*programa objeto*)



- o processo de tradução pode ser feito por:

- **COMPILAÇÃO:**
cada comando é convertido em uma série de instruções em linguagem da máquina-objetivo, em uma série de etapas. Executa a decodificação uma só vez para cada comando, mas tem que guardar todo o código gerado. Pode otimizar o código gerado; e tem depuração mais complexa
- **INTERPRETAÇÃO:**
cada comando da linguagem é diretamente executado, produzindo um resultado imediato. Economiza memória, mas a rapidez de execução é comprometida; e tem depuração mais simples

Arquiteturas de computadores

As arquiteturas típicas de computadores podem ser agrupadas em três tipos:

- arquiteturas sequenciais
- arquiteturas com paralelismo em baixo nível
- arquiteturas paralelas

Classificação segundo Flynn para fluxos de dados e instruções:

- SISD (**S**ingle **I**nstruction **S**ingle **D**ata stream)
- SIMD (**S**ingle **I**nstruction **M**ultiple **D**ata stream)
- MISD (**M**ultiple **I**nstruction **S**ingle **D**ata stream)
- MIMD (**M**ultiple **I**nstruction **M**ultiple **D**ata stream)
 - com memória compartilhada
 - programação mais simples com compartilhamento de recursos
 - limitações (barramento/memória cache)
 - com memória distribuída
 - programação mais complexa (múltiplos usos/usuários)
 - limitações devido às necessidades de comunicação entre componentes

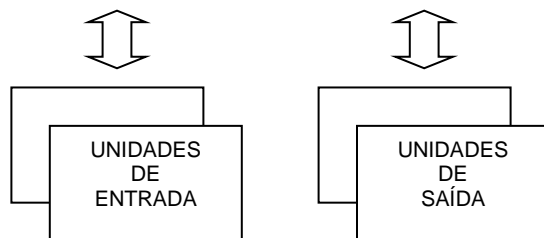
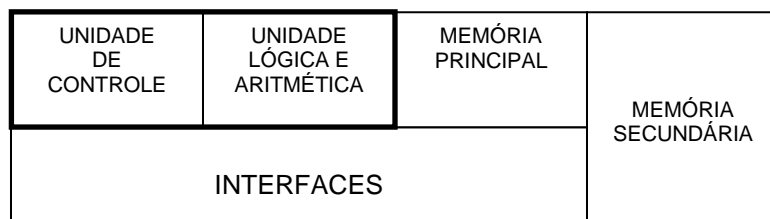
Separação segundo o conjunto de instruções:

CISC	RISC
- conjunto de instruções mais complexo	- conjunto de instruções mais simples
- conjunto de instruções mais numeroso	- conjunto de instruções reduzido
- instruções mais longas (mais bits)	- instruções menores (menos bits)
- requer mais ciclos de <i>clock</i>	- requer menos ciclos de <i>clock</i> (mais rápido)
- menos ortogonal (instruções específicas e pouco usadas)	- mais ortogonal (mais potência e maior flexibilidade)

Modelo de computador de arquitetura sequencial

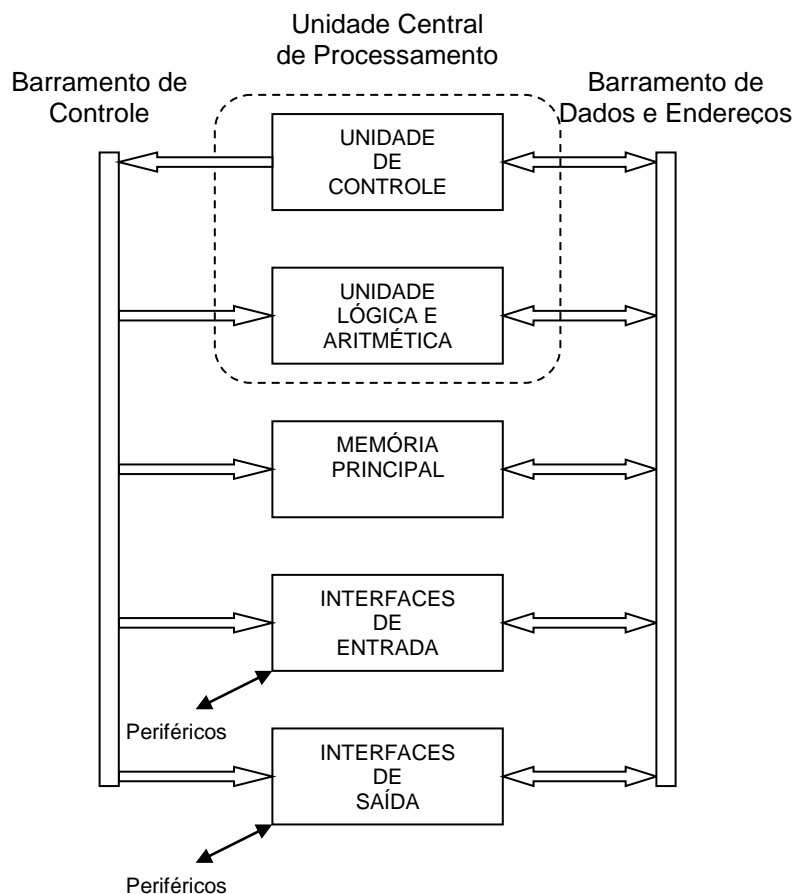
MODELO DE COMPUTADOR

Unidade Central de Processamento



PERIFÉRICOS

As unidades funcionais se comunicam através de vias (ou *barramentos*) de dados e endereços, de controle e pelas interfaces de entrada ou saída com componentes periféricos.



Unidades de entrada e saída

- Unidade de fita magnética (DAT)
- Unidade de disco magnético (rígido ou flexível)
- Unidade de disco ótico (CD-ROM, CD-RW, DVD-ROM, DVD-RW, Blue-Ray)
- Leitora e perfuradora de cartões
- Leitora e perfuradora de fita de papel
- Terminal de apresentação visual ("vídeo")
- Leitora ótica e leitora de caracteres magnéticos
- Impressora
- Tabletes e mesa digitalizadora
- Manipuladores (*gamepad, joystick, touchpad, mouse, trackball, pen*)
- Unidade de fotocomposição
- Unidade de captura e processamento de som/voz
- Unidade de captura e processamento de imagens
- Unidade de captura de movimentos
- Sensores
- Atuadores

Unidade aritmética e lógica

É a unidade encarregada de realizar operações aritméticas e lógicas elementares.

Unidade de controle

É a unidade encarregada de coordenar os diversos componentes.

Memória principal

É a unidade encarregada de guardar os dados recebidos das unidades de entrada para imediato processamento. Um dado a ser processado pelo computador pode ser colocado na memória na hora de se executar as ações de transformação e, também, um resultado de uma transformação pode ser armazenado antes que passe para as unidades de saída.

A memória é considerada um meio temporário de armazenamento de dados, que permanecem ali durante o tempo em que estiverem sendo processados.

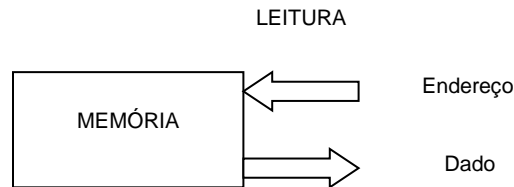
Tipos de memória

- RAM (*Random Access Memory*)
 - leitura e gravação
- ROM (*Read-Only Memory*)
 - apenas leitura
- PROM (*Programmable Read-Only Memory*)
 - ROM programável pelo usuário
- EPROM (*Erasable Programmable Read-Only Memory*)
 - PROM apagável por luz ultra-violeta
- EAPROM (*Electrically Alterable Read-Only Memory*)
 - PROM apagável eletricamente

Transferência de dados

Em qualquer computador dados são transferidos entre a unidade de armazenamento (*memória*) e as outras unidades. O esquema de seleção de dados a serem transferidos para (ou da) memória é conhecido como *endereçamento*.

Se for desejado saber um conteúdo de memória, o processador coloca o endereço correspondente no *barramento* de endereços e a memória responde colocando no *barramento* de dados uma cópia da palavra contida naquela posição.

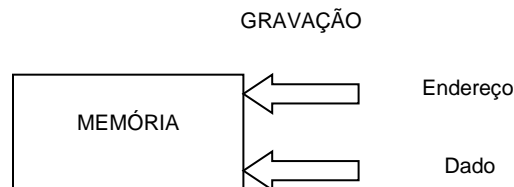


Ou de forma reduzida para se obter cópia de um conteúdo de memória em certo endereço:

dado <- MEMÓRIA [endereço]

Se for desejado guardar um conteúdo na memória, o processador coloca o endereço correspondente no *barramento* de endereços e o dado no *barramento* de dados.

Vários sinais de controle são usados para controlar a direção e a temporização das transferências.



Ou de forma reduzida para se obter cópia de um conteúdo de memória em certo endereço:

MEMÓRIA [endereço] <- dado

Memória secundária

A memória secundária pode ser composta por vários tipos de dispositivos capazes de ampliar a capacidade de armazenamento da memória principal. Essas memórias auxiliares podem armazenar grandes quantidades de dados e programas, permitindo que sejam solicitados diretamente pela memória principal quando necessários.

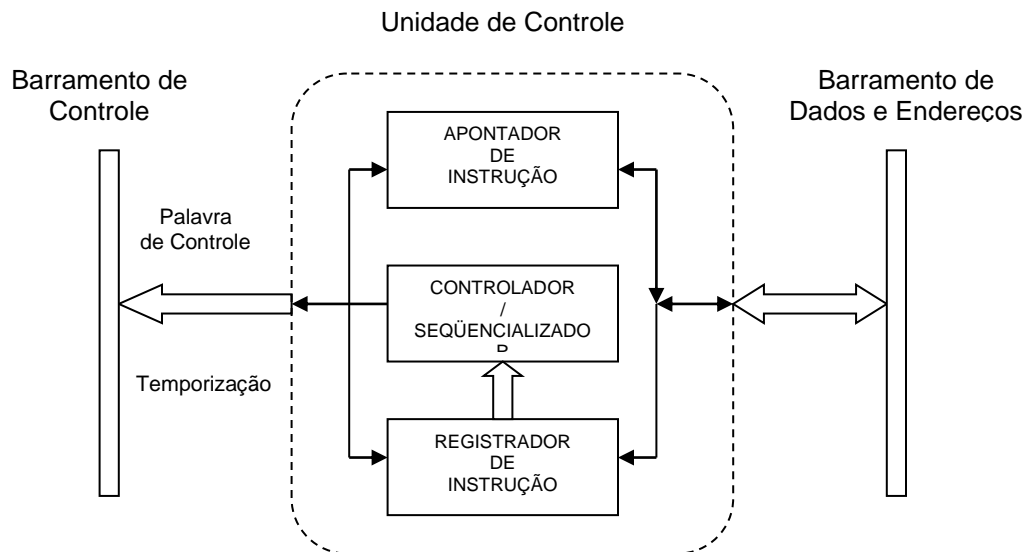
Outra função da memória secundária é oferecer expansão virtual da memória principal.

Componentes básicos

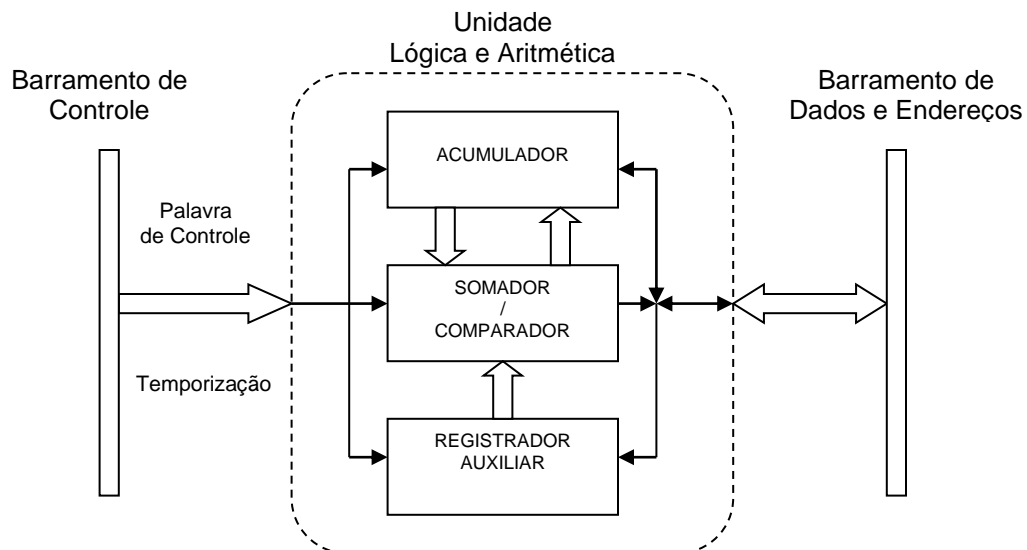
Os registradores são os blocos básicos com os quais são construídos os computadores. Qualquer computador pode ser entendido como um conjunto de registradores e linhas de comunicação (vias ou barramentos) entre eles.

Exemplos:

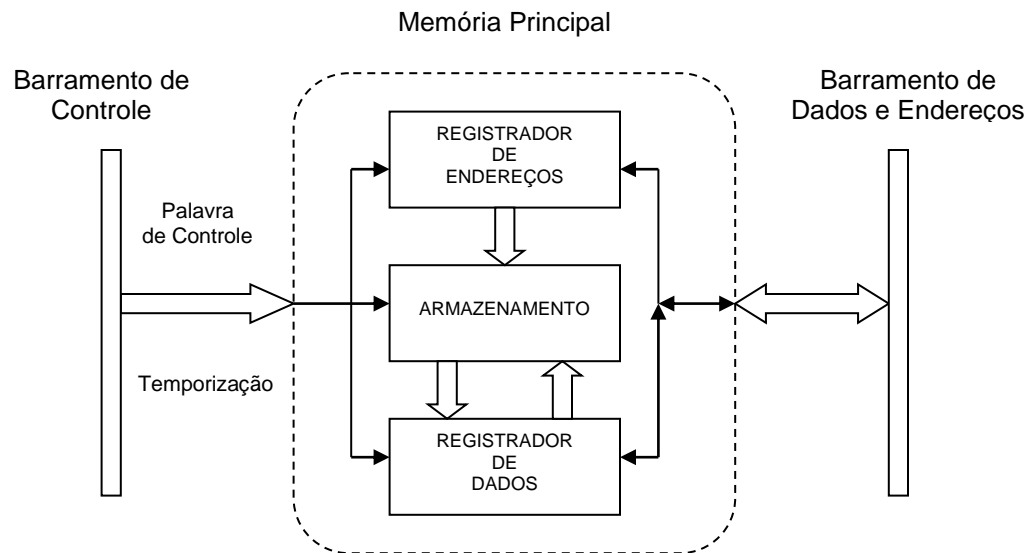
- para controle:
 - apontador de instrução (AI / PC - *Program Counter*)
serve para guardar o endereço da instrução que vai ser executada;
 - registrador de instrução (RI / IR - *Instruction Register*)
serve para guardar uma cópia da instrução que vai ser executada



- para operações:
 - registrador principal (acumulador)
 - registrador auxiliar



- para armazenamento e organização do acesso à memória:
 - registrador de endereço (REM / NR - *Index Register*)
 - registrador de dados (RDM / DR - *Data Register*)



- Instruções

Tipos de instrução:

- carga e armazenamento
- lógico-aritméticas
- teste e desvio

- Instruções

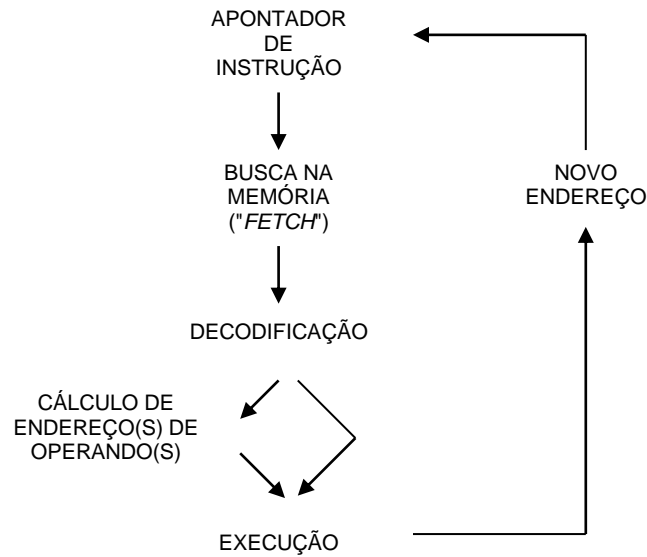
Tipos de instrução:

- carga e armazenamento
- lógico-aritméticas
- teste e desvio

Formatos de instrução

código da operação			
código da operação	endereço de	operando	(ou instrução)
código da operação	endereço de operando	endereço de operando	(ou instrução)
código da operação	endereço de operando	endereço de operando	endereço do resultado (ou instrução)

Ciclo básico de execução de uma instrução:



Ciclo de memória:

Tempo requerido para uma operação elementar.

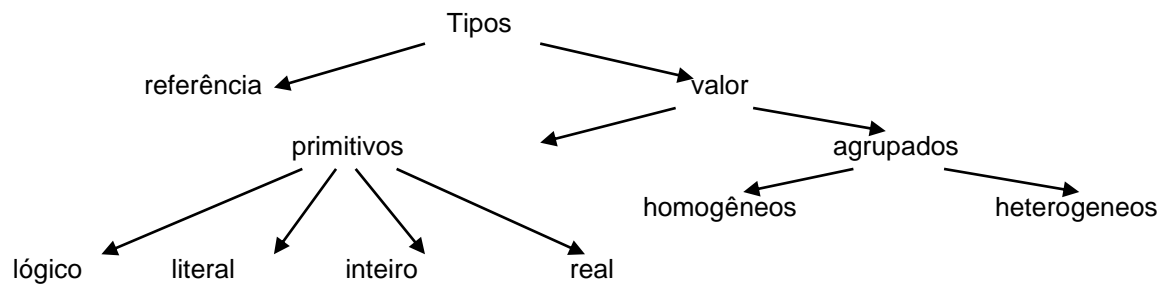
- Tamanho de palavra
É o número de bits de informação (instrução ou dado) que um registrador ou posição de memória é capaz de armazenar.

Exemplos:

Computadores com palavras de:

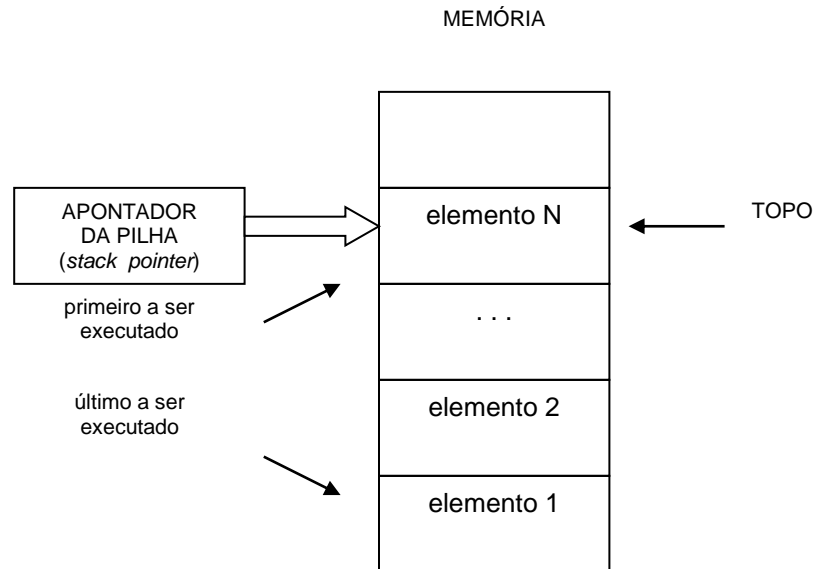
- 08 bits - baseados no microprocessador Zilog Z80
- 16 bits - baseados no microprocessador Intel 8086
- 32 bits - baseados no microprocessador Intel 80386
- 64 bits - baseados no microprocessador Intel Itanium
- 52 bits - UNISYS B6910

- Tipos de operandos (dados) e operações
Os operandos podem ter tamanhos variando de 1 bit a vários bytes, e representar valores de diversas naturezas segundo uma codificação. Podem ser simples ou agrupados. Podem conter os valores diretamente ou serem referências para onde esses estão guardados na memória.



- Pilha

A pilha em um computador é uma parte da memória usada também por subrotinas e para tratar interrupções. O conceito é simples: a cada vez que um dado é colocado na pilha (*push*), ele é posto em cima de outros já armazenados. Só o topo da pilha é acessível a cada momento e pode ser retirado (*pop*). Quando isto acontece, o dado é removido da pilha, e o dado imediatamente abaixo desse seerá indicado como o topo. A vantagem da pilha é o acesso rápido ao conteúdo.



- Capacidade de endereçamento

Refere-se a quantidade de memória que o computador é capaz de endereçar.

Exemplo:

Se um apontador de instrução tem 16 bits, nenhum programa pode usar mais de 65536 posições de uma só vez.

- Velocidade

Depende de dois fatores: número de ciclos gastos em uma instrução e da velocidade de operação geral do computador.

Exemplo:

Se um computador opera a 1 Ghz, o tempo de um ciclo de instrução é 1 nanossegundo.

- Programação

Exemplo:

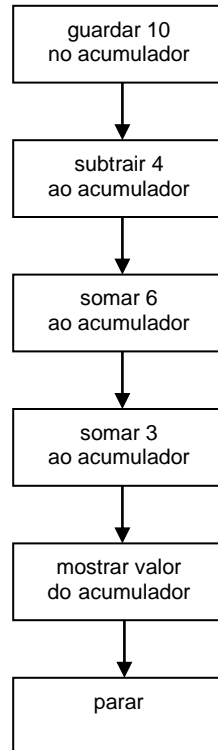
Supor para o modelo de computador descrito acima as seguintes instruções:

Mnemônico	Código	Significado
nop	0000 0000	passar adiante, não fazer nada
out	0001 0000	copiar o conteúdo do acumulador para o registrador de saída
inc	0010 0000	incrementar de uma unidade o valor no acumulador
dec	0011 0000	decrementar de uma unidade o valor no acumulador
load [dd]	0100 xxxx	carregar o acumulador com o conteúdo da memória na posição xxxx (equivalente binário a [dd])
store dd	0101 xxxx	carregar o conteúdo do acumulador na memória na posição xxxx
add [dd]	0110 xxxx	carregar o registrador B com o conteúdo da memória na posição xxxx, e somá-lo ao acumulador
sub [dd]	0111 xxxx	carregar o registrador B com o conteúdo da memória na posição xxxx, e subtraí-lo do acumulador
not	1000 0000	complementar o conteúdo do acumulador
or	1001 0000	disjunção do acumulador com o conteúdo da memória na posição xxxx
and	1010 0000	conjunção do acumulador com o conteúdo da memória na posição xxxx
xor	1011 0000	disjunção exclusiva do acumulador com o conteúdo da memória na posição xxxx
jump dd	1100 xxxx	carregar o apontador de instrução como endereço da posição xxxx e executar a instrução que ali estiver
jumpS dd	1101 xxxx	testar se o sinal do acumulador é negativo; se for, saltar para a instrução na posição xxxx; senão, executar a próxima instrução
jumpZ dd	1110 xxxx	testar se o acumulador é igual a zero; se for, saltar para a instrução na posição xxxx; senão, executar a próxima instrução
halt	1111 0000	parar a execução

Aplicação 1:

Escrever um programa capaz de calcular: $10 - 4 + 6 + 3$.

Ações a serem processadas



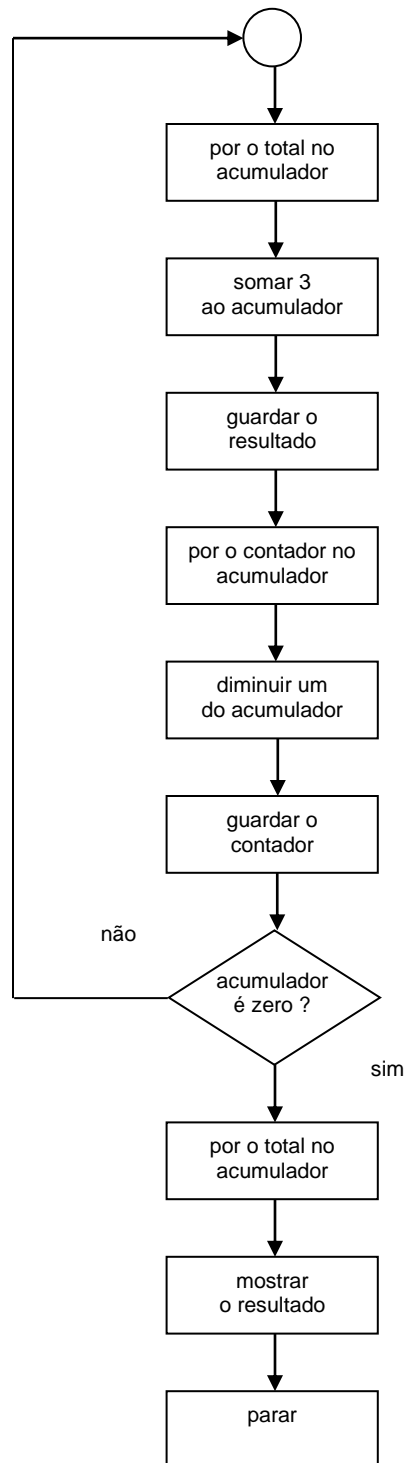
Mapeamento das ações na memória:

	Endereços	Conteúdo	Significado	Descrição
00	0000	0100 1100	load [12]	pegar (10) (valor em [12])
01	0001	0111 1101	sub [13]	subtrair (4) (valor em [13])
02	0010	0110 1110	add [14]	somar (6) (valor em [14])
03	0011	0110 1111	add [15]	somar (3) (valor em [15])
04	0100	0001 0000	out	mostrar resultado
05	0101	1111 0000	halt	parar
06	0110	0000 0000	nop	
07	0111	0000 0000	nop	
08	1000	0000 0000	nop	
09	1001	0000 0000	nop	
10	1010	0000 0000	nop	
11	1011	0000 0000	nop	
12	1100	0000 1010	[10]	valor = (10)
13	1101	0000 0100	[04]	valor = (04)
14	1110	0000 0110	[06]	valor = (06)
15	1111	0000 0011	[03]	valor = (03)

Aplicação 2:

Escrever um programa capaz de calcular: 3×5 .

Ações a serem processadas



Mapeamento das ações na memória:

	Endereços	Conteúdo	Significado	Descrição
00	0000	0100 1111	load [15]	pegar o total em [15]
01	0001	0110 1100	add [12]	somar (3)
02	0010	0101 1111	store 15	guardar resultado em [15]
03	0011	0100 1101	load [13]	pegar contador em [13]
04	0100	0111 1110	sub [14]	subtrair (1)
05	0101	0101 1101	store 13	guardar contador em [13]
06	0110	1110 1000	jumpZ 08	se for zero, ir para [08]
07	0111	1100 1001	jump 00	se não for, voltar a [00]
08	1000	0100 1111	load [15]	pegar o total em [15]
09	1001	0001 0000	out	mostrar resultado
10	1010	1111 0000	halt	parar
11	1011	0000 0000	nop	
12	1100	0000 0011	(03)	parcela a somar
13	1101	0000 0101	(05)	vezes a somar
14	1110	0000 0001	(01)	unidade
15	1111	0000 0000	(00)	total (valor inicial)

Modelos de arquiteturas de computadores

Há três modelos de arquiteturas que se destacam:

- *o de von Neumann (ou de Princeton)*, com uma unidade central de processamento (CPU) e uma unidade de armazenamento (memória) para dados e instruções; usado nos primeiros computadores como o Mark I, ENIAC, EDSAC e ENIAC;
- *o de Harvard*, com unidades de armazenamento separadas para dados e instruções; geralmente empregado em processadores de sinais (DSP) e microcontroladores (PIC);
- *o de Harvard Modificado* com uma só unidade de armazenamento, mas com a capacidade da CPU acessar concorrentemente dois ou mais barramentos de dados. Inclui dispositivos separados (caches) para acessar dados (como no modelo de von Neumann) e instruções (como no modelo de Harvard). Empregado nas famílias de processadores x86 e ARM.

Histórico dos principais microprocessadores

A partir do início dos anos 1960, fabricantes de componentes eletrônicos começaram a colocar vários transistores em pastilhas de silício com 1/4" de área (*circuito integrado*), aumentando progressivamente o número de elementos por pastilha (circuitos integrados em larga escala). Esses circuitos foram os modelos para a maioria dos microprocessadores modernos.

Em 1971, a INTEL Co. apresentou o primeiro microprocessador de 4 bits - o 4004. Em 1973, foi lançado o de 8 bits - o 8008. Em seguida apareceram no mercado microprocessadores como o INTEL 8080 e INTEL 8085, o ZILOG Z-80 e o MOTOROLA 6800.

Essas linhas de produtos evoluíram naturalmente para os microprocessadores de 16 bits como: os INTEL 8088, 8086, 80186, 80286, o ZILOG Z-8000 e o MOTOROLA 68000.

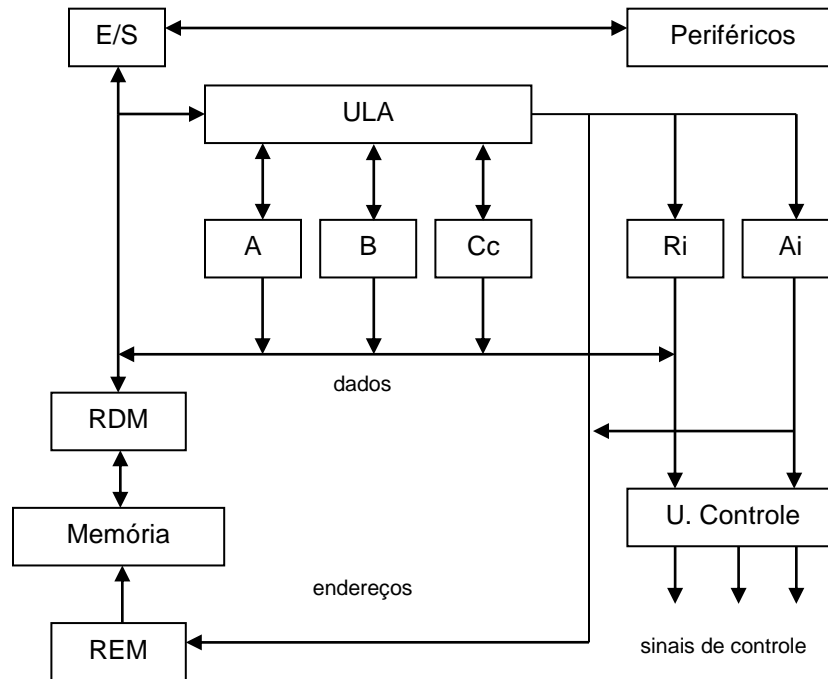
Em seguida surgiram os microprocessadores de 32 bits: os INTEL 80386, 80486 e Pentium, o INTEL i860, e a família MOTOROLA 680xx; e os de 64 bits, como a família Alpha da DEC, Ultra da Sun, Itanium da Intel e Opteron da AMD.

Nos últimos anos, têm sido introduzidos microprocessadores, cada vez mais rápidos e mais complexos que seus predecessores, bem como arranjos de multiprocessadores: Intel Dual-Core, Core Duo, Quad Core; e AMD Turion e Fusion.

Isso se reflete diretamente na capacidade de endereçamento de memória, na rapidez de execução e na complexidade de funções desempenhadas.

Organização básica de um computador digital (*von Neumann*):

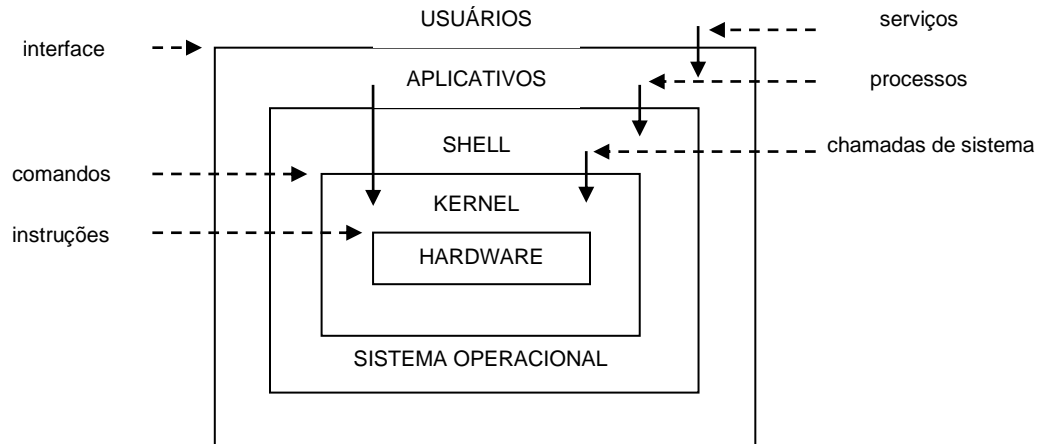
Usa a máquina para guardar dados e os códigos de uma sequência de operações (*instruções*) que atuam sobre dados que representam a solução algorítmica do problema (*programa*).



A	Acumulador ou registrador A Dedicado às tarefas de cálculo e manipulação de dados.
B	Registrador B Auxiliar para operações com dois operandos.
Ai	Apontador de instrução Guarda o endereço da posição de memória que contém o código da próxima instrução a ser executada. Após a execução dessa, ele passará a indicar a próxima.
Ri	Registrador de instrução Guarda uma cópia da instrução a ser decodificada e interpretada pela Unidade de Controle.
Cc	Códigos de condição (ZERO, NEGATIVO, CARRY, OVERFLOW) Registra ocorrências excepcionais (<i>flags</i> ou <i>status</i>).
REM	Registrador de Endereços da Memória Guarda o endereço da palavra a ser manipulada.
RDM	Registrador de Dados da Memória Guarda cópia do conteúdo de um endereço da memória.

Sistemas operacionais

Sistema operacional é o programa (ou a coleção de programas) responsável pela supervisão dos processos computacionais em progresso em um computador digital de uso geral.



Funções:

- tornar a comunicação do homem com a máquina (interface) mais natural e inteligível, apresentando ao usuário uma máquina mais flexível para se utilizar/programar;
- possibilitar o uso eficiente e controlado dos seus diversos recursos. É o principal responsável pela alocação e controle dos recursos físicos (*hardware*), de modo a homogeneizar e compatibilizar as diferentes velocidades de operação, permitindo ao computador funcionar na sua capacidade máxima, ou próximo a ela;
- possibilitar o uso compartilhado e protegido dos diversos recursos, ainda assim eficiente, e que os usuários possam se beneficiar do trabalho conjunto de outros, e cooperando entre si na execução de projetos complexos.

A sua comunicação com o usuário se faz através de uma *linguagem de controle* - declarações ou comandos simples (*shell*) - que levam o sistema operacional a executar uma ação específica.

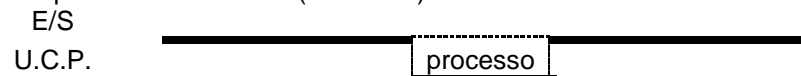
Tipos de sistemas operacionais:

- monolíticos
- em camadas
- micronúcleo (*microkernel*)
- cliente-servidor
- máquina virtual
- exonúcleo (máquinas virtuais separadas por recursos)

Limitado pela unidade central de processamento (*CPU-bound*)



Limitado por entradas/saídas (*IO-bound*)



Classificação

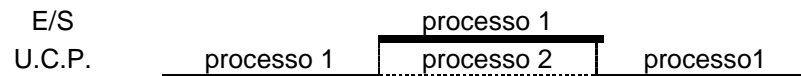
1. Sistemas com monoprogramação (monotarefa):

Possuem apenas uma unidade central de processamento ativa todo o tempo (monoprocessamento), mas pode ficar livre quando há operações de entrada/saída. Permitem a execução de apenas um processo/tarefa de cada vez por usuário (monousuário).



2. Sistema com multiprogramação (multitarefa/*multitasking*):

Geralmente com uma unidade central de processamento ativa ao mesmo tempo. Permitem a execução de dois, ou mais processos (tarefas), concorrentemente (multiprocessamento), de um ou mais usuários (multiusuário). Realizam uma única tarefa, de cada vez, em termos de processamento.



2.1 Sistemas do tipo lote (*batch*):

Neste sistema, as tarefas (atividades computacionais solicitadas de uma só vez por um usuário externo, através de uma sequência de comandos) são agrupadas fisicamente e processadas sequencialmente. Têm por objetivo maximizar o número de tarefas processadas por unidade de tempo, e minimizar o tempo médio de espera para execução de cada tarefa.

2.2 Sistemas de tempo compartilhado (*time-sharing*):

Neste sistema, há várias tarefas ativas, que devido ao tempo relativamente longo entre comandos (*think time*), são atendidas por intervalos de tempo definidos, suspensas, e colocadas à espera de um novo ciclo, ou ao atendimento de prioridades. Devem ter um tempo de resposta por comando dentro de limites aceitáveis.

2.3 Sistemas de tempo real (*real-time*):

Semelhantes ao de tempo repartido, mas com um intervalo de tempo prefixado para a resposta, após o qual poderá haver perda de informação, ou operação incorreta, e até catastrófica sobre o objeto monitorado.

2.4 *Smartcards* / *embedded* (embutidos)

3. Sistemas com múltiplos processadores

Possuem mais de uma unidade central de processamento ativa ao mesmo tempo. Podem atender vários usuários simultaneamente.

3.1 Fracamente acoplados (*loosely coupled*)

Sistemas interconectados de tempo uniforme (SMP) ou de tempo variável (NUMA).

3.2 Fortemente acoplados (*tightly coupled*)

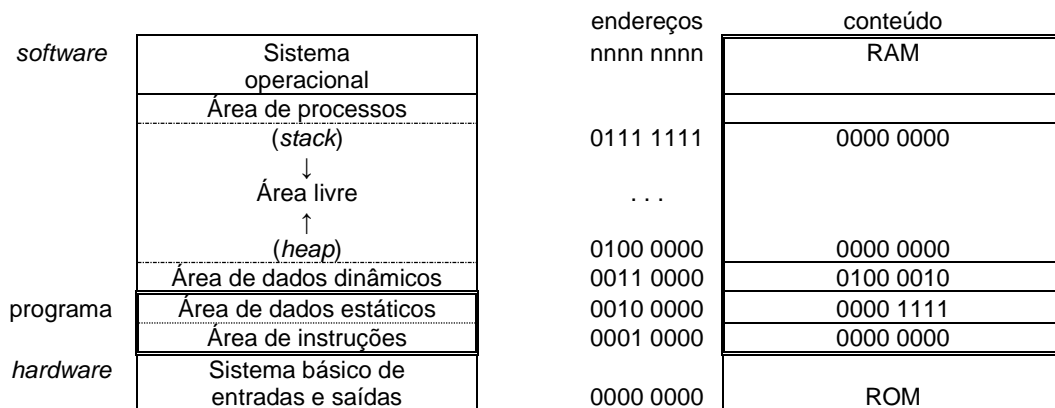
Sistemas interconectados em *clusters*, em rede ou distribuídos (nuvem).

Modelo de memória com sistema operacional e espaço de trabalho

O modelo abaixo procura situar uma possível localização relativa de elementos de um sistema operacional e o espaço de trabalho de um programa, incluindo suas áreas de dados estáticos e dinâmicos, bem como a área livre cujo uso poderá ser compartilhado por ambos. Sobre essa área livre poderão atuar estruturas gerenciadoras de espaço chamadas de **stack** e **heap**.

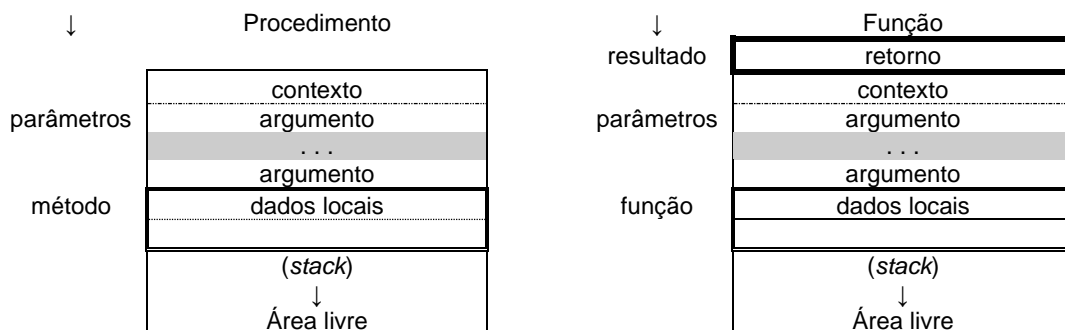
Parte da memória poderá ser apenas para leitura de dados (ROM), e outra parte poderá servir para se escrever e ler dados (RAM). Quer sejam dados ou instruções, as representações desses conteúdos serão feitas por valores em binário associados, cada um, a endereços. A quantidade de bits usados nessas representações poderá variar de acordo com a arquitetura do sistema computacional.

Modelo de memória.



Para o acionamento de módulos (procedimentos e funções), uma porção da área livre (*stack*) é usada para se guardar os contextos das chamadas, os argumentos (parâmetros), e os resultados (retornos) das funções.

Acionamento de módulos.



Para a alocação dinâmica, uma porção da área livre (*heap*) é reservada para guardar dados/objetos.

Modelos de representações de dados na memória

Exemplo de modelo físico de dados simples na memória.

endereço hexadecimal	endereço binário (&)	memória	representação interna	valor
0x0C	0000 1100			
0x0B	0000 1011			
0x0A	0000 1010	1111 1111	byte max = 0b1111 1111;	max = 255
0x09	0000 1001	0100 0001	literal símbolo = 0x00 41;	símbolo = 'A'
0x08	0000 1000	0000 0000	&símbolo = 0x08;	(endereço do símbolo)
0x07	0000 0111	0000 0011	inteiro y = 0x0000 000A;	y = 10;
0x06	0000 0110	0000 0000		
0x05	0000 0101	0000 0000		
0x04	0000 0100	0 000 0000	&y = 0b0000 0100;	(endereço de y)
0x03	0000 0011	0000 0101	inteiro x = 0x0000 0005;	x = 5
0x02	0000 0010	0000 0000		
0x01	0000 0001	0000 0000		
0x00	0000 0000	0 000 0000	&x = 0b0000 0000;	(endereço de x)


Exemplo de modelo físico de dados agrupados na memória.

endereço binário	memória	arranjo inteiro v[3];
0000 1100		
0000 1011	0000 0001	v[2] = 1;
0000 1010	0000 0000	
0000 1001	0000 0000	endereço do indexado (referência indexada)
0000 1000	0 000 0000	&(v[2]) = &(v[0])+2*tamanho(inteiro); // 2*4 bytes
0000 0111	0000 0011	v[1] = 3;
0000 0110	0000 0000	
0000 0101	0000 0000	endereço do indexado (referência indexada)
0000 0100	0 000 0000	&(v[1]) = &(v[0])+1*tamanho(inteiro); // 1*4 bytes
0000 0011	0000 0101	v[0] = 5;
0000 0010	0000 0000	
0000 0001	0000 0000	endereço de base (referência primária)
0000 0000	0 000 0000	&(v[0]) = 0x00;

Exemplo de modelo físico de objeto (referência para outros dados/métodos) na memória.

endereço (referência)	memória	dados
		atributo 3
		atributo 2
base dos atributos		atributo 1

		método 1
base dos métodos		construtor padrão
		endereço base do descritor de métodos
base do objeto		endereço base do descritor de atributos



Exercícios propostos

1. O que significa *endereçamento de memória* ?
2. Para que serve um *registrador* ?
3. Quais os dois principais tipos de memória ?
4. Relacionar os itens abaixo:

(a) PROGRAMA	() unidade de informação
(b) PALAVRA	() unidade de representação
(c) BIT	() unidade de tratamento interno
(d) DADO	() conjunto de instruções
(e) BYTE	() conjunto de informações
5. Diferenciar *compilador* de *interpretador*.
6. Qual o tamanho de uma memória com 16 bits de endereço ?
7. Qual a velocidade de processamento de um computador que trabalha a 2 GHz ?
8. Descrever o que acontece na fase de busca de instrução.
9. Escrever um programa capaz de calcular: $3 \times 5 - 4$
10. Citar as principais funções de sistemas operacionais.
11. Relacionar, pelo menos, 03 sistemas operacionais atuais.
12. Procurar definições para:

CONNECTIVITY	DOWNSIZING
OUTSOURCING	RIGHTSIZING
DBMS	RESIZING
LAN	WAN
TOP-DOWN	BOTTOM-UP