COMPUTADORES DIGITAIS I

- *Disciplina: Computadores Digitais* I
- *Código:* **FEN 504.7**
- *Carga Horária: 75 horas*
- *Número de Créditos: 4*
- *Professora: Nadia Nedjah*

EMENTA E BIBLIOGRAFIA

Ementa

- *Introdução*
- *Código e Aritmética*
- *Unidade Lógica e Aritmética*
- *Unidade de Controle*
- *Memória e Armazenamento*
- *Entrada e Saída*
- *Multi-Programação*
- *Sistemas Operacionais*

Bibliografia

- *Andrew S. Tenenbaum, Organização Estruturada de Computadores, Prentice-Hall do Brasil.*
- *William Stalling, Arquitetura de Computadores, Prentice-Hall.*

AVALIAÇÃO

Avaliação

1. *Provas parciais escritas* (2)
2. *Freqüência* (75%)
3. *Média parcial computada:*

$$M_p = (2N_1 + 2N_2 + L)/5$$

4. *Se for necessário, prova final escrita* (1)
5. *Média final computada:*

$$M_p \geq 7.0 \Rightarrow M_f = M_p$$

$$4.0 \leq M_p < 7.0 \Rightarrow M_f = (M_p + N_f)/2$$

INTRODUÇÃO

Histórico

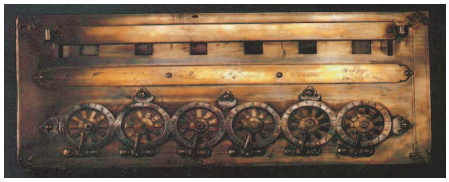
Denominação	Tecnologia	Período
Geração Zero	Mecânica	(1642 – 1945)
Primeira Geração	Válvulas	(1945 – 1955)
Segunda Geração	Transistores	(1955 – 1965)
Terceira Geração	Circuitos integrados	(1965 – 1980)
Quarta Geração	Integração de larga escala	(1980 – atual)
Quinta Geração	Circuitos quânticos	(futuro – . . .)

INTRODUÇÃO

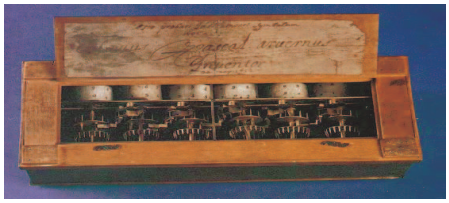
Histórico: Geração Zero

- Blaise Pascal constrói a primeira máquina de calcular somas e subtrações (1623 – 1662);

Blaise Pascal



La Pascaline

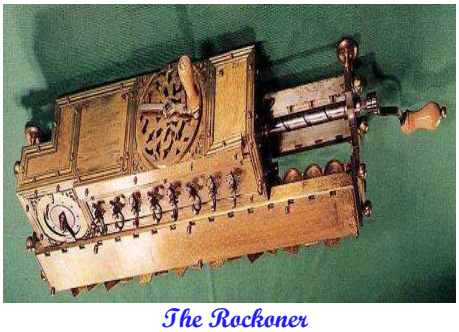


INTRODUÇÃO

Histórico: Geração Zero

- Gottfried Leibniz completa a máquina de calcular de Blaise Pascal com as operações de multiplicação e divisão (1646 – 1716);

Gottfried Leibniz



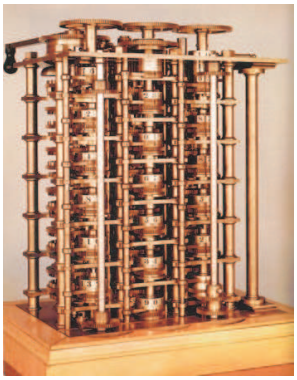
The Rockoner

INTRODUÇÃO

Histórico: Geração Zero

- *Charles Babbage constrói a máquina de diferenças que implementa o método de diferenças finitas (1792 – 1871);*

Charles Babbage

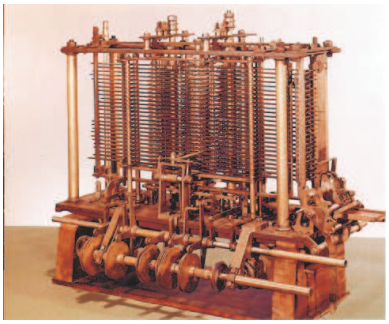


A máquina de diferenças

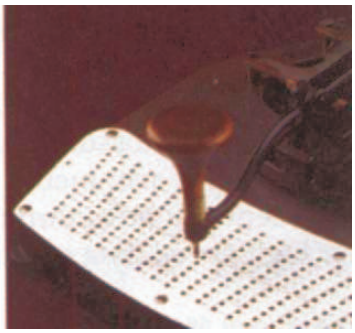
INTRODUÇÃO

Histórico: Geração Zero

- *Charles Babbage constrói uma nova máquina que possui quatro componentes: o armazenamento (memória), o engenho (unidade de cálculo), leitora de cartões perfurados (entrada) e perfurador de cartões (saída).*



A máquina analítica

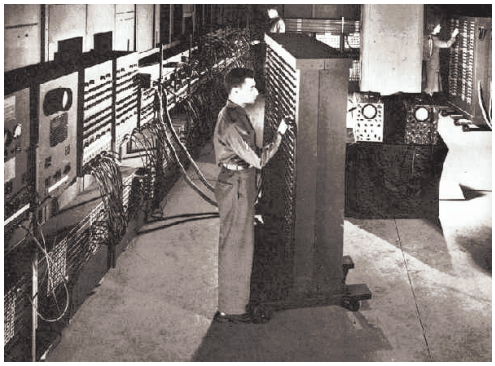


Perfurador de cartões

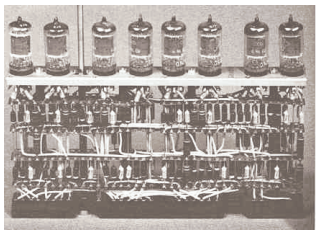
INTRODUÇÃO

Histórico: Primeira Geração

- *John Mauchley e seu aluno Presper Eckert constroem o Electronic Numerical Integrator and Computer – ENIAC. Consiste de 18.000 válvulas e 1.500 relés.*



ENIAC

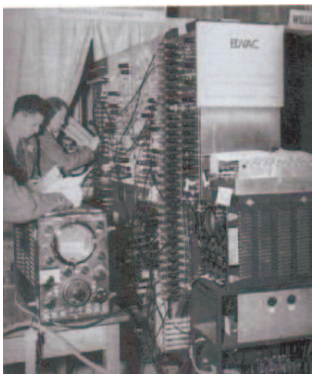


válvulas e relés

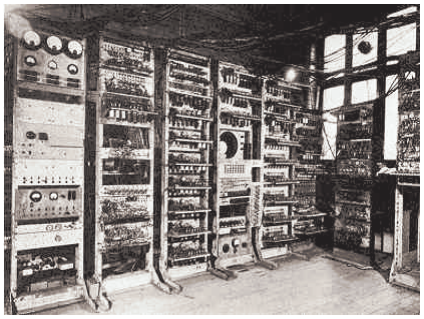
INTRODUÇÃO

Histórico: Primeira Geração

- *John von Neumann constrói o Electronic Discrete Variable Automatic Computer – EDIVAC.*



EDIVAC

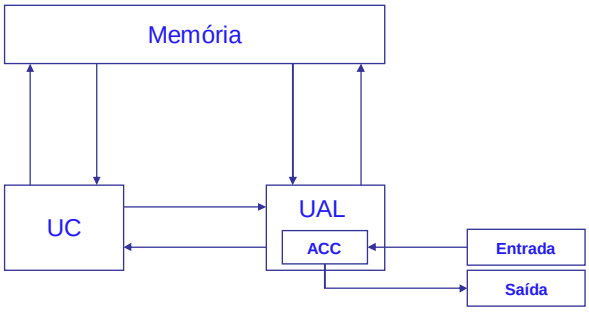


MARK I

INTRODUÇÃO

Histórico: Primeira Geração

- *Hoje em dia, EDIVAC é conhecido por máquina de von Neumann.*
- *É o primeiro computador com um programa armazenado.*
- *Possui a seguinte arquitetura, dita de von Neumann:*



INTRODUÇÃO

Histórico: Segunda Geração

- *A firma DEC (Digital Equipement Corporation) constrói o PDP-1 e a IBM o 709, todos dois baseados em transistores;*
- *Primeiro display (CRT – Cathode Ray Tube);*
- *Primeiro videogame "Guerra nas estrelas";*
- *A DEC constrói o PDP-8 e a IBM o 7090 baseados em um barramento único, diferentemente da máquina de von Neumann;*

- *A firma CDC (Control Data Corporation) constrói o 6600 cuja CPU inclui várias unidades funcionais que podem atuar em paralelo.*

INTRODUÇÃO

Histórico: Terceira Geração

- Encapsulamento de dezenas a milhares de transistores em uma única pastilha (SSI, MSI, LSI);
- Computadores menores, mais rápidos e mais baratos;
- A IBM constrói o System 360 baseado em circuitos integrados;
- Avanço no mundo de mini-computadores;
- A DEC lança o mini-computador PDP-11

INTRODUÇÃO

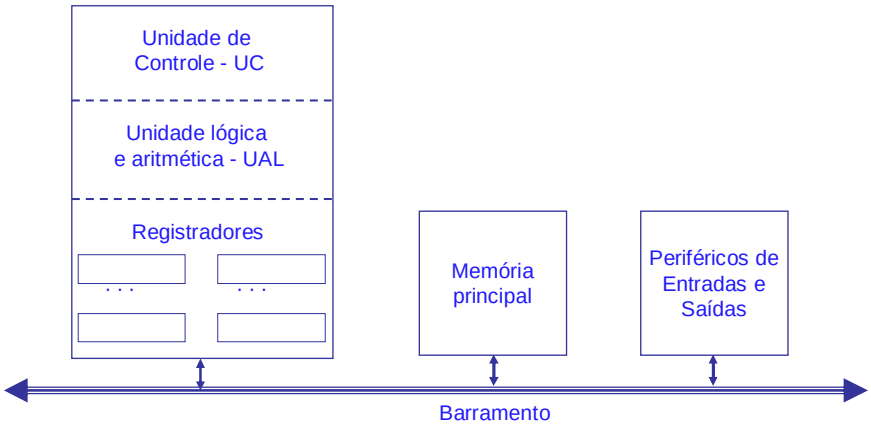
Histórico: Quarta Geração

- Ecapsulamento de milhões de transistores em uma única pastilha (VLSI);
- Avanços na indústria de micro-computadores;
- A IBM lança o micro-computador IBM OS/2;
- A Intel lança o 8080 e Zílog o Z80 que são os primeiros processadores de 8 bits numa única pastilha;
- A Intel lança o 8086 que é o primeiro processador de 16 bits em uma única pastilha;
- A Intel lança o 80386 e a Motorola o 68020 que são verdadeiros processadores de 32 bits em uma única pastilha.

INTRODUÇÃO

Organização Básica

Um computador digital consiste em um sistema interligado de processadores, memórias e dispositivos de entrada e saída.



INTRODUÇÃO

Conceitos Básicos

- *Uma instrução é um comando para uma máquina;*
- *Um programa é uma seqüência de instruções que descrevem como executar uma determinada tarefa;*
- *Uma linguagem de máquina consiste num conjunto de instruções primitivas da máquina; devem ser simples reduzindo a complexidade e custo da implementação;*
- *Uma máquina é um dispositivo capaz de executar um programa escrito na sua linguagem de máquina;*

INTRODUÇÃO

Conceitos Básicos

- *Uma linguagem de baixo nível é toda aquela cujas instruções estão intimamente ligadas às características da máquina;*
- *Uma linguagem d alto nível é toda aquela cujas instruções estão intimamente ligadas às características da aplicação;*
- *Um máquina virtual é uma máquina hipotética para uma determinada linguagem;*
- *Uma máquina define uma linguagem, assim como uma linguagem define uma máquina;*

INTRODUÇÃO

Conceitos Básicos

- *Tradução é um método pelo qual um programa escrito numa linguagem L_2 é substituído por um outro programa escrito em linguagem L_1, então executado pela máquina M_1, cuja linguagem de máquina é L_1;*
- *Compilação é um processo de tradução de uma linguagem de alto nível para uma linguagem de baixo nível;*
- *Interpretação é o método pelo qual uma máquina M_1, cuja linguagem de máquina é L_1, executa um programa escrito em linguagem L_2, instrução por instrução, através de uma seqüência de instruções equivalentes em L_1;*

INTRODUÇÃO

Conceitos Básicos

- *O hardware constitui a parte física do computador;*
- *O software consiste em programas para o computador;*
- *O firmware consiste em software embutido em dispositivos eletrônicos durante a fabricação;*
- *Hardware e software são logicamente equivalentes, já que qualquer operação efetuada pelo software pode também ser implementada pelo hardware assim como qualquer operação executada pelo hardware pode ser também simulada pelo software;*
- *A decisão entre uma implementação em hardware ou software depende de custo, velocidade, confiabilidade e freqüência de alterações;*

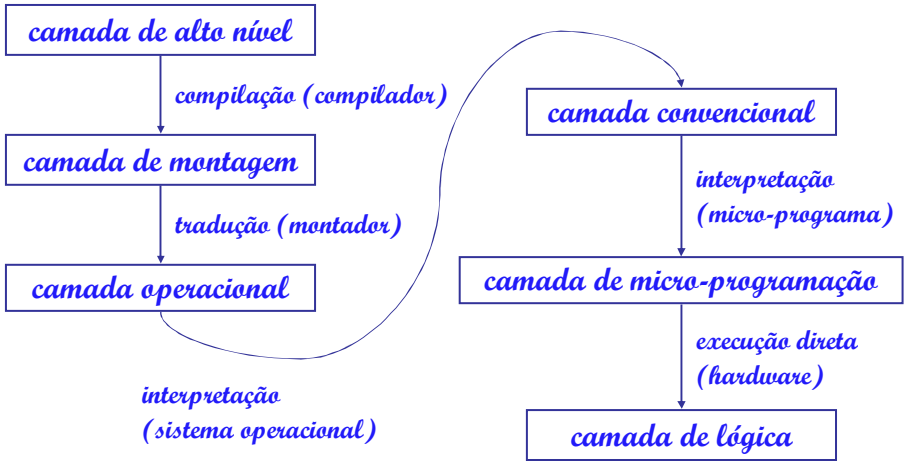
INTRODUÇÃO

Conceitos Básicos

- *O computador é um sistema complexo. A divisão em níveis permite uma melhor compreensão da sua organização e funcionamento;*
- *Um computador pode ser visto como uma hierarquia de N níveis, cada um associado a uma máquina virtual, exceto o nível mais baixo que corresponde à máquina real;*
- *Um programador de um determinado nível não precisa conhecer os níveis inferiores;*
- *Os computadores atuais são organizados em seis níveis;*

INTRODUÇÃO

Máquina Multi-Níveis



INTRODUÇÃO

Conceitos Básicos

- O $nível_0$ ou de lógica digital
  - é o hardware verdadeiro do computador e consiste nos dispositivos eletrônicos que respondem aos estímulos gerados pelas instruções da linguagem de máquina do $nível_1$;
  - não existe o conceito de programa;
  - os objetos são denominados portas.

INTRODUÇÃO

Conceitos Básicos

- O $nível_1$ ou de micro-programação
  - é o verdadeiro nível de máquina, havendo um programa denominado micro-programa, cuja função é interpretar as instruções de $nível_2$; a instrução neste nível é chamada micro-instrução;
  - as instruções deste nível são executadas diretamente pelo hardware do computador ($nível_0$).

INTRODUÇÃO

Conceitos Básicos

- O $nível_2$ ou de máquina convencional
  - é o primeiro nível de máquina virtual;
  - a linguagem de máquina deste nível é comumente denominada linguagem de máquina do computador;
  - as instruções são interpretadas pelo micro-programa;
  - em computadores que não tenham o nível de micro-programação, as instruções do nível convencional são executadas diretamente pelo hardware do computador ($nível_0$).

INTRODUÇÃO

Conceitos Básicos

- O nível$_3$ ou de sistema operacional
  - apresenta a maior parte das instruções do nível$_2$.
  - apresenta um conjunto de novas instruções que são chamadas ao sistema operacional;
  - tem organização diferente da memória principal, utilizando o conceito de memória virtual;
  - tem a capacidade de executar dois ou mais programas em paralelo, utilizando o conceito de multi-programação;
  - as novas funcionalidades são realizadas por um interpretador denominado sistema operacional em execução no nível$_2$;
  - as instruções do nível$_3$ idênticas às de nível$_2$ são executadas pelo micro-programa (nível$_1$).

INTRODUÇÃO

Conceitos Básicos

- O nível$_4$ ou de linguagem de montagem
  - apresenta as instruções do nível$_3$ na forma de mnemônicos;
  - os programas em linguagem de montagem (assembly language) são traduzidos para a linguagem de nível$_3$ e então interpretados pela máquina apropriada (nível$_2$ ou nível$_1$);
  - O programa que realiza essa tradução é denominado montador (assembler).

INTRODUÇÃO

Conceitos Básicos

- O $nível_5$ ou de linguagem orientada para problemas
  - consiste em linguagem de alto nível;
  - os programas escritos nestas linguagens são geralmente compilados para o $nível_4$ ou $nível_3$;
  - o programa que executa essa transformação é denominado compilador da linguagem de alto nível em questão.

- Os programas de $nível_2$ e $nível_3$ são sempre interpretados, enquanto os de $nível_4$ e $nível_5$ são geralmente traduzidos;
- As linguagens de $nível_1$, $nível_2$ e $nível_3$ são binárias, ao passo que as de $nível_4$ e $nível_5$ são simbólicas, contendo palavras e abreviaturas.

CÓDIGOS E ARITMÉTICA

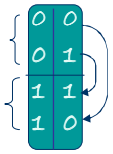
Códigos

- Um código é um conjunto de regras (ou tabela de combinações) pelo qual informações podem ser convertidos a uma representação do código e vice-versa.
- Código BCDIC ou Binary-Coded Decimal Interchange Code é formado de 4 bits
- Código ASCII ou American Standard Code for Information Interchange: é um conjunto de códigos de sete bits para símbolos alfanuméricos e de pontuação. O oitavo bit é usado como bit de paridade.
- Código EBCDIC ou Extended Binary-Coded Decimal Interchange Code é formado de 8 bits. É uma extensão do BCDIC e mais abrangente que o código ASCII mas é um código escasso.

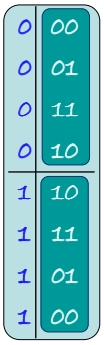
CÓDIGOS E ARITMÉTICAS

Códigos

- **Código de Gray ou** reflected binary code, **mais usado em controle.**
- **A distância de** Hamming **de dois códigos é o número de posições em que estes são diferentes**



Código de Gray de 2 bits



Código de Gray de 3 bits

Núm	Código	
0	0	000
1	0	001
2	0	011
3	0	010
4	0	110
5	0	111
6	0	101
7	0	100
8	1	100
9	1	101
10	1	111
11	1	110
12	1	010
13	1	011
14	1	001
15	1	000

Código de Gray de 4 bits

CÓDIGO E ARITMÉTICA

Sistemas de Numeração

- ***Número decimal*** **consiste na representação de um número na base 10;**

$$(213)_{10} = 2.10^2 + 1.10^1 + 3.10^0$$

- ***Número binário*** **consiste na representação de um número na base 2.**

$$(213)_{10} = 1.2^7 + 1.2^6 + 0.2^5 + 1.2^4 + 0.2^3 + 1.2^2 + 0.2^1 + 1.2^0 = (11010101)_2$$

- ***Número hexadecimal*** **consiste na representação de um número na base 16.**

$$(1293)_{10} = 5.16^2 + 0.16^1 + D.16^0 = (50D)_{16}$$

CÓDIGOS E ARITMÉTICA

Números Binários Negativos

Existem 4 sistemas de representação de números negativos no computador.

1. Sinal e magnitude: o bit mais significativo é usado para representar o sinal (0 é + e 1 é −) e os bits restantes representam a valor absoluto do número;

   +8 → 00001000 −8 → 10001000

   Sinal e magnitude complica somadores e comparadores!

2. Complemento de um: um número negativo de valor absoluto v é o complemento lógico do número positivo de mesmo valor absoluto, incluindo o bit de sinal;

   +8 → 00001000 −8 → 11110111

   Este sistema é obsoleto!

CÓDIGOS E ARITMÉTICA

Números Binários Negativos

3. Complemento de dois: é a soma da constante 1 e do complemento de um do número; Se houver um vai-um, este é desprezado;

   +8 → 00001000 −8 → 00001000 + 11111111 + 1 = 11111000

4. Excesso de 2^{n-1}: Números positivos e negativos são representados pela soma deles com 2^{n-1}.

   +8 + 128 = 136 → 10001000

   −8 + 128 = 120 → 01111000

   Observe que esse sistema é idêntico ao sistema complemento de dois com bit de sinal invertido.

CÓDIGOS E ARITMÉTICA

Operações Aritmética

```
  124
- 10
------
= 114
```

Subtração decimal

```
  01111100
- 10001010
----------
= 01110010
```

Subtração sinal magnitude

```
  01111100
+ 11110101
----------
= 101110001
+        1
----------
= 01110010
```

Subtração complemento de 1

```
  01111100
+ 11110110
----------
= X01110010
```

Subtração complemento de dois

```
  11111100
+ 01110110
----------
= X01110010
```

Subtração excesso de 2^7

CÓDIGOS E ARITMÉTICA

O complemento de dois tem <u>várias vantagens</u>:

1. *Tão fácil de determinar o sinal quanto na representação sinal magnitude;*
2. *Quase tão fácil de mudar o sinal de um número quanto na representação sinal magnitude;*
3. *Adição pode ser realizada sem se preocupar com qual dos operandos é o maior;*
4. *O número zero tem uma única representação;*
5. *Um único hardware é usado tanto para a soma de operandos com sinal quanto operandos sem sinal;*

O complemento de dois tem <u>uma desvantagem</u>:

- *Numa representação de tamanho n bits, o número -2^n pode ser representado mas o número $+2^n$ não!*

CÓDIGOS E ARITMÉTICA

Números em Ponto Fixo

- *A palavra do computador é dividida em duas parte: inteira e fracionária, não necessariamente iguais;*
- *O ponto decimal é fictício;*

$3.625_{10} = 1.2^{1} + 1.2^{0} + 1.2^{-1} + 0.2^{-2} + 1.2^{-3} = 11(.)101_{2}$

- *Para somar e subtrair em ponto fixo, a posição do ponto decimal não faz diferença.*

	3.750		00011.110		
+	6.375	+	00110.011		
=	10.125	=	01010.001		
	3.750		00011.110		
−	6.375	+	11001.101		
=	(-)2.625	=	11101.011		

$01010.001_2 = 1.2^3+1.2^1+1.2^{-3}$
$= 8+2+0.125 = 10.125$

	11101.011
−	1
=	11101.010
	10010.101

$10.101_2 = 1.2^1+1.2^{-1}+1.2^{-3}$
$= 2 + 0.5 + 0.125$
$= 2.625$

CÓDIGOS E ARITMÉTICA

Números em Ponto Fixo

- *O resultado da multiplicação e divisão em ponto fixo depende da localização do ponto decimal;*
- *Se o número de bits significativos na parte inteira do produto é maior do que o número de bits associados a parte inteira na representação, então tem um transbordamento (overflow)*
- *Se o número de bits significativos na parte fracionária do produto do que o número de bits associados a parte fracionária na representação, então tem uma perda de precisão*

	3.7500		11.11		
×	12.7500	+	1100.11		
=	47.8125	=	101111.1101		

$101111.1101_2 = 47.8125$

$T_i = 4 \Rightarrow$ *overflow*

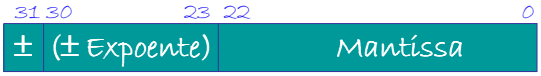
$T_f = 2 \Rightarrow$ *perda de precisão e o resultado será* 47.75

CÓDIGOS E ARITMÉTICA

Números em Ponto Flutuante

- *Os problemas de overflow e perda de precisão introduzidos pela representação em ponto fixo podem ser superado com a representação de números em ponto flutuante.*
- *Na representação em ponto flutuante, um número qualquer N na base 2 é definido com três componentes: o sinal, a mantissa M, o expoente E.*

$$N = \pm M \times 2^{\pm E}$$



Precisão simples

Precisão dupla

CÓDIGOS E ARITMÉTICA

Números em Ponto Flutuante

- *Nos dois formatos (ponto fixo ou flutuante), o expoente é representado em notação de excesso de 128 e 1024 respectivamente.*
- *O número de dígitos no expoente determina efetivamente o intervalo dos dados representados;*
- *O número de dígitos na mantissa determina a precisão do dado representado;*
- *Uma representação em ponto flutuante é normalizada se o dígito mais significativo da mantissa é diferente de zero.*
- *Representação não-normalizado:*

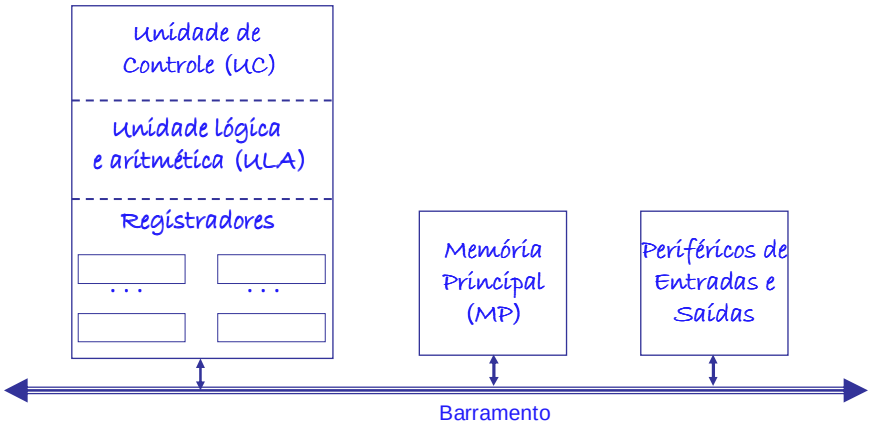
  $-0.0275 \times 2^{-19} = 1 \mid 01101101 \mid 000001110001010001 \ldots$

- *Representação normalizado:*

  $-0.0275 \times 2^{-19} = 1 \mid 01110010 \mid 1110001010001 \ldots$

ORGANIZAÇÃO BÁSICA

- *Um computador digital consiste em um sistema interligado de processadores, memórias e dispositivos de entrada e saída.*



ORGANIZAÇÃO BÁSICA

- *A UCP (Unidade Central de Processamento) tem como função executar programas armazenados na memória principal (MP), buscando as instruções, examinando-as e, então, executando uma após a outra.*
- *A UC (Unidade de Controle) é responsável pela busca das instruções da MP e sua análise.*
- *A ULA (Unidade Lógica e Aritmética) realiza operações lógicas e aritméticas.*
- *Os registradores da UCP constituem uma memória local, de alta velocidade, usada para armazenar resultados temporários, informação de controle (CP, RI, ACC)*

ORGANIZAÇÃO BÁSICA

- *A UCP executa uma instrução na seguinte seqüência:*

  1. *busca a próxima instrução;*
  2. *atualiza PC;*
  3. *determina tipo da instrução;*
  4. *determina onde estão os dados;*
  5. *busca os dados;*
  6. *executa a instrução;*
  7. *armazena resultados;*
  8. *volta ao passo 1.*

- *Esta seqüência de passos é frequentemente referida como ciclo de busca, decodificação e execução.*

ORGANIZAÇÃO BÁSICA

Classificação de Flynn

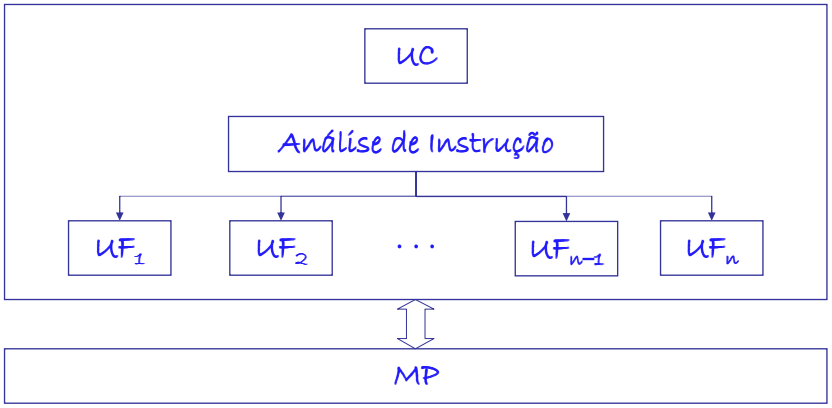
- *Limites físicos determinam até que ponto as máquinas podem ser aceleradas simplesmente aumentando a velocidade do hardware.*

- *Uma alternativa está em explorar a execução paralela de instruções, ao invés da tradicional execução seqüencial (von Neumann).*

- *As máquinas paralelas podem ser classificadas de acordo com o fluxo de instruções e de dados que elas tem.*

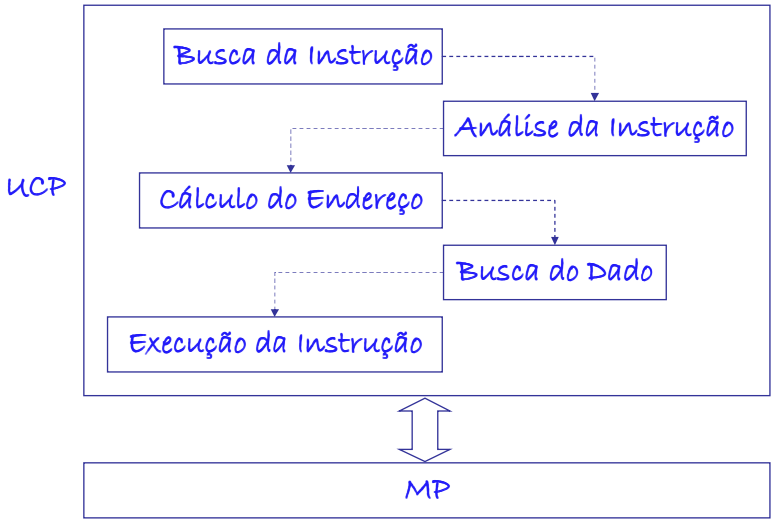
ORGANIZAÇÃO BÁSICA

- *Classe SISD – Single Instruction, Single Data (fluxo único de instruções e de dados); máquina von Neumann; algum paralelismo, buscando-se e iniciando-se a próxima instrução antes de terminar a corrente (CDC6600)*



ORGANIZAÇÃO BÁSICA

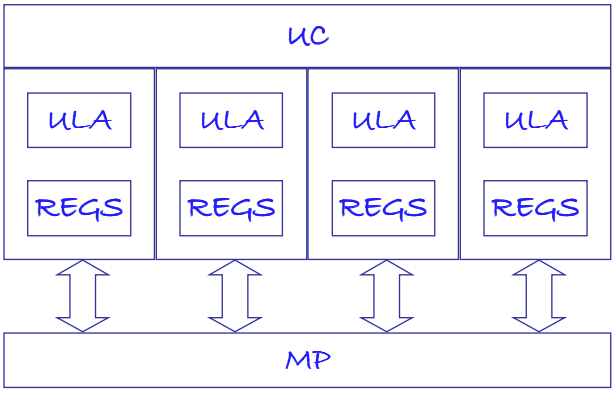
Máquina "pipeline"



ORGANIZAÇÃO BÁSICA

- *Classe SIMD - Single Instruction, Multiple Data (fluxo único de instruções e múltiplo de dados).*

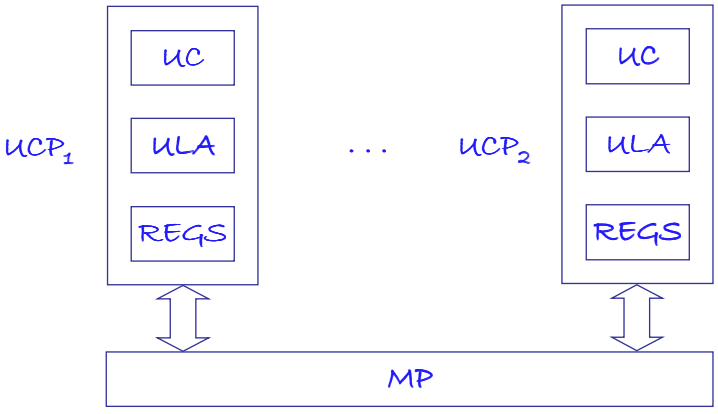
"array processor"



ORGANIZAÇÃO BÁSICA

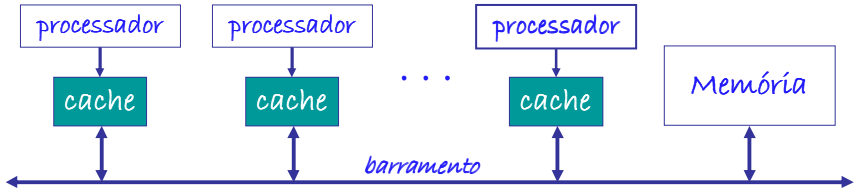
- *Classe MIMD - Multiple Instructions, Multiple Data (fluxo múltiplo de instruções e de dados); UCPs diferentes executam programas diferentes.*

sistema multiprocessador com memória compartilhada



ORGANIZAÇÃO BÁSICA

- A memória compartilhada deve possuir a propriedade de coerência.
- O problema deste esquema é que com um número pequeno de processadores (4 ou 5), o barramento fica saturado e o desempenho do sistema cai drasticamente.
- Uma solução para evitar a saturação do barramento é acrescentar cada processador de uma memória cache (memória de acesso direto e alta velocidade). A cache guarda as palavras consultadas recentemente.



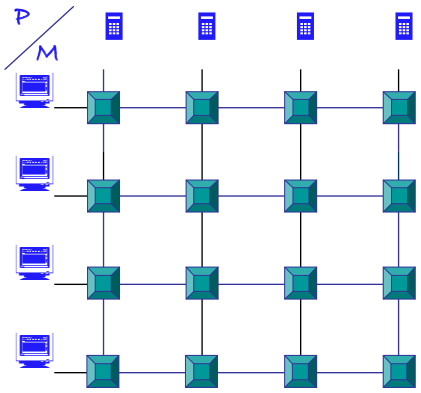
ORGANIZAÇÃO BÁSICA

- Todas as requisições à memória compartilhada passam pela memória cache.
- Se a palavra estiver na cache, ela própria responde ao processador, sem necessidade de usar o barramento.
- Análise estatística mostra que com uma memória cache de 64KByte, o grau de acerto é de 90%.
- Senão a requisição é encaminhada para a memória compartilhada
- Para conservar a propriedade de coerência, a memória cache precisa ter duas características principais:
  - write-through: modificar a memória compartilhada
  - Snoopy: monitora o barramento para se atualizar.

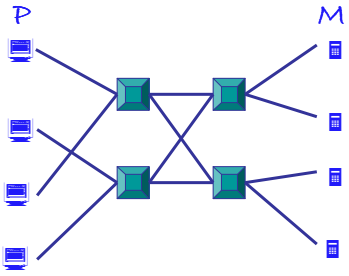
ORGANIZAÇÃO BÁSICA

- *Para um sistema multi-processador com mais de 64 processadores, não se pode empregar uma comunicação processador/memória por barramento.*
- *Uma possibilidade consiste em dividir a memória em módulos e conectá-los aos diferentes processadores do sistema.*
- *Isso pode ser implementado através de:*
  - *uma barra cruzada, permitindo um acesso rápido mas um número alto de chaves; Para conectar N processadores à N módulos de memória, são necessárias N^2 chaves*
  - *uma rede ômega, permitindo um acesso relativamente rápido com um número muito mais reduzido de chaves; Para conectar N processadores à N módulos de memória utilizando chaves de tipo k Xk, são necessárias $N/k \times \log_k N$ chaves.*

ORGANIZAÇÃO BÁSICA



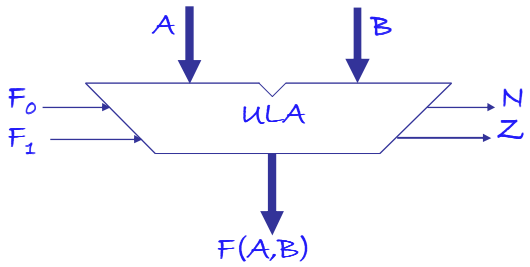
Barra cruzada



Rede ômega

UNIDADE LÓGICA E ARITMÉTICA

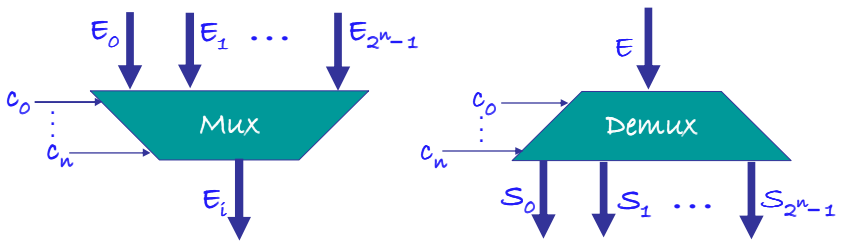
- *Uma das partes de um computador digital é a unidade lógica e aritmética (ULA);*
- *A ULA é responsável pela execução direta de somas, subtrações, funções Booleanas, comparações, etc.*
- *Em geral, a ULA é um circuito combinatório com dois sinais de entrada de dados e um sinal de saída de resultado, havendo outros sinais de controle.*



UNIDADE LÓGICA E ARITMÉTICA

Componentes Básicos

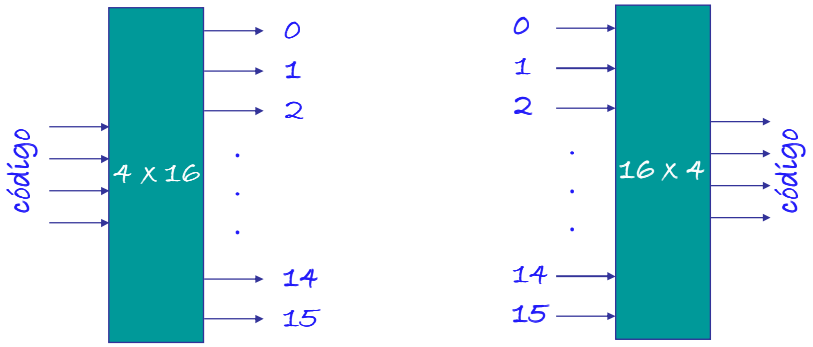
- *Um multiplexador tem 2^n entradas, uma saída da mesma largura da entrada e uma entrada de controle de n bits, que seleciona uma das entradas e a direciona para a saída.*
- *Um demultiplexador é o inverso de um multiplexador, direcionando a entrada para uma dentre 2^n saídas, de acordo com as n linhas de controle.*



UNIDADE LÓGICA E ARITMÉTICA

Componentes Básicos

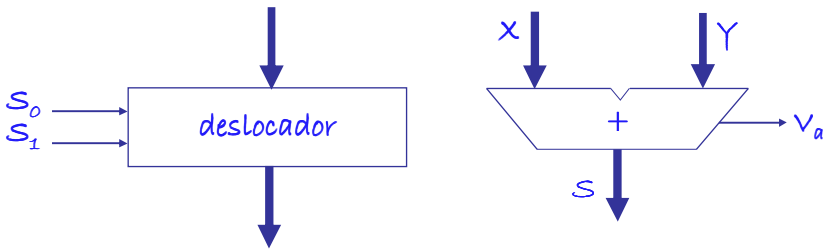
- *Um decodificador tem n linhas de entrada e 2^n linhas de saída. De acordo com o código binário da entrada, uma das linhas é ativada.*
- *Um codificador é o inverso de um decodificador, possuindo 2^n entradas e n saídas. Somente uma das entradas estará ativa.*



UNIDADE LÓGICA E ARITMÉTICA

Componentes Básicos

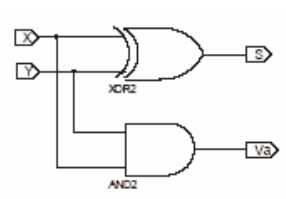
- *Um deslocador é um circuito com capacidade para deslocar à direita ou à esquerda, ou mesmo não deslocar.*
- *Um somador de n bits é um circuito que adiciona dois operandos cada um de n bits retornando a soma de n bits e o bit de "vai um".*



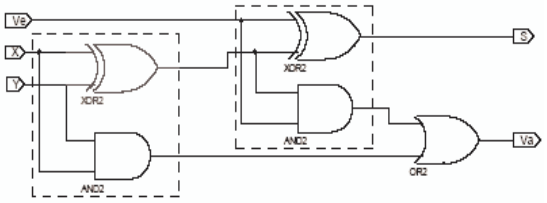
UNIDADE LÓGICA E ARITMÉTICA

Componentes básicos

- *Um meio-somador é um circuito que recebe 2 bits na entrada e fornece 2 bits na saída: o primeiro é a soma módulo 2 dos bits de entrada e o segundo é o "vai-um" ou carry-out;*
- *Um somador completo é um circuito que recebe três bits na entrada e fornece 2 bits na saída: o primeiro é a soma módulo 2 do bits de entrada e o segundo é o "vai-um"; O terceiro bit na entrada é geralmente chamado o "vem-um" ou carry-in;*



Meio-somador

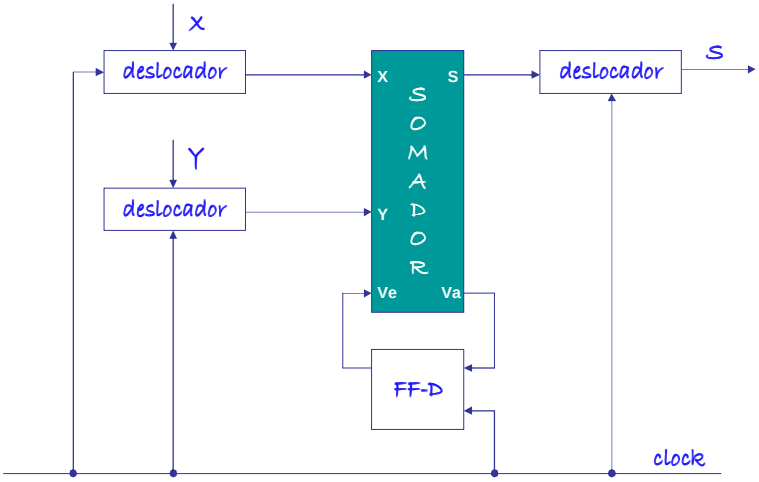


Somador completo

UNIDADE LÓGICA E ARITMÉTICA

Componentes Básicos

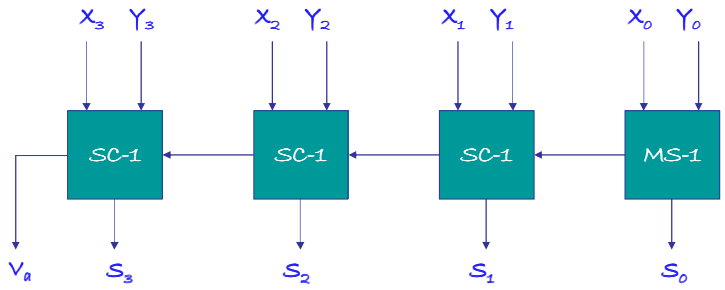
- *Somador em série: é econômico mais lento!*



UNIDADE LÓGICA E ARITMÉTICA

Componentes Básicos

- *Somador com propagação do "vai um" (ripple carry adder): é um somador paralelo, bastante usado pela sua simplicidade e economia apesar do atraso introduzido pela propagação do "vai um".*



UNIDADE LÓGICA E ARITMÉTICA

Componentes Básicos

- *Somador com "vai um" antecipado (carry lookahead adder): é um somador paralelo com previsão do "vai um".*

$$s_i = x_i \oplus y_i \oplus v_{i-1} \qquad v_i = g_i + t_i \cdot v_{i-1}$$
$$g_i = x_i \cdot y_i \qquad t_i = x_i \oplus y_i$$

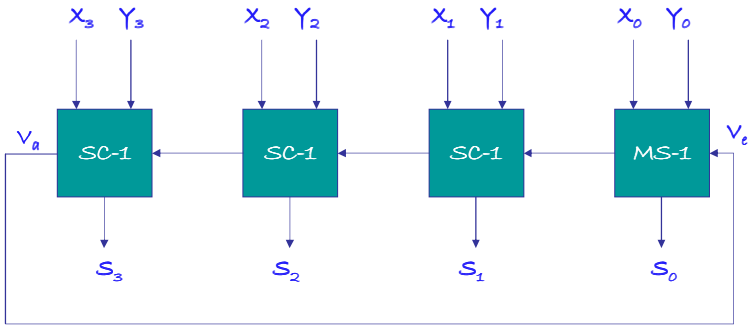
- *Chama-se g_i de geração de "vai um", pois se $g_i = 1$, com certeza há um "vai um" gerado pelos bits dos operandos no estágio i do somador.*
- *Chama-se t_i de transporte de "vai um", pois se $t_i = 1$, haverá um "vai um" se houver um "vem um". O "vem um" é gerado por um bit de um dos operandos e o "vem um".*
- *Para operandos de 4 bits, o "vai um" antecipado é calculado:*

$$v_a = g_0 + t_0 . g_1 + t_0 . t_1 . g_2 + t_0 . t_1 . t_2 . g_3 + t_0 . t_1 . t_2 . t_3 . v_e$$

UNIDADE LÓGICA E ARITMÉTICA

Componentes Básicos

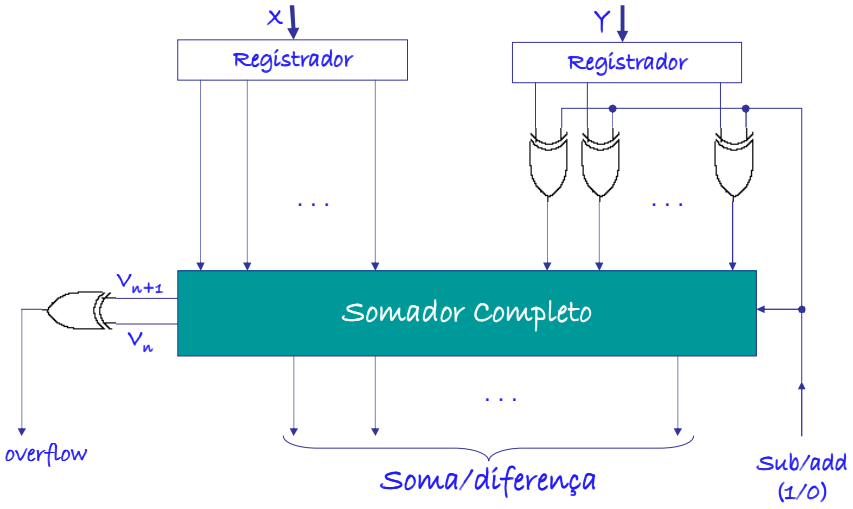
- *Somador/subtrator em complemento de 1*



UNIDADE LÓGICA E ARITMÉTICA

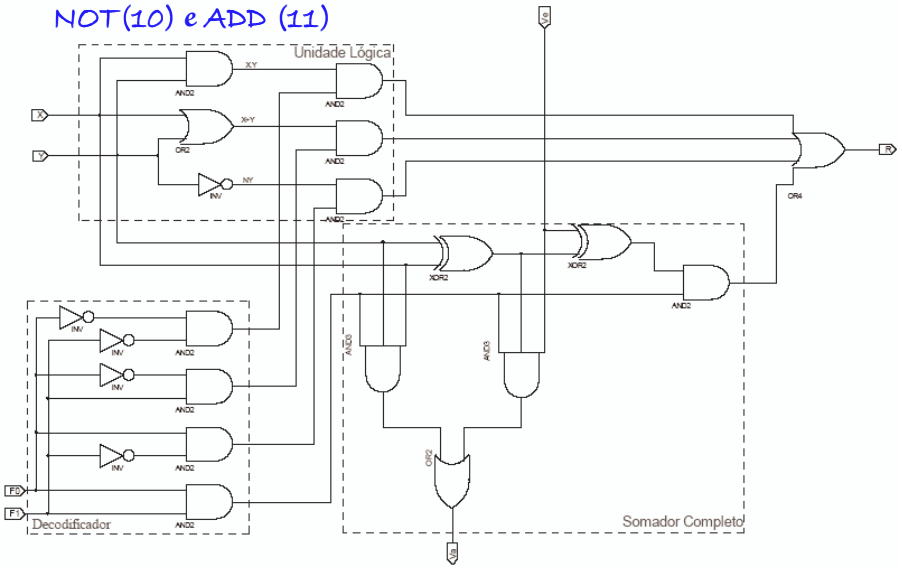
Componentes Básicos

- *Somador/Subtrator em complemento de dois*



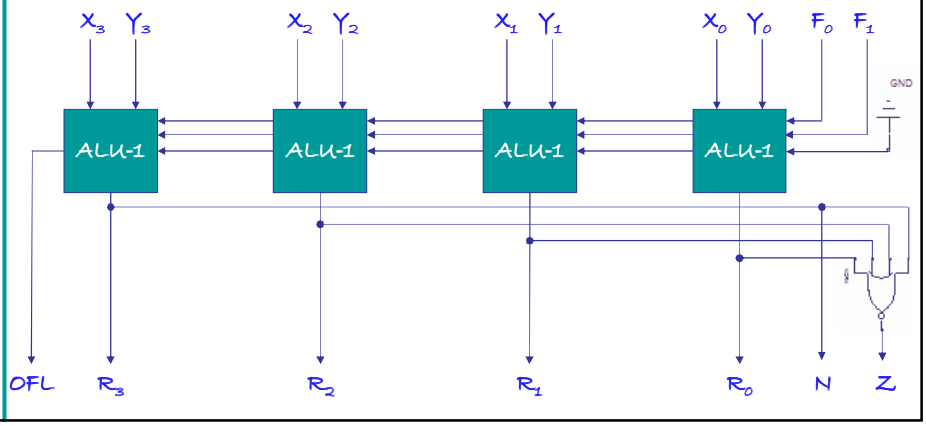
UNIDADE LÓGICA E AITMÉTICA

- *Uma ULA de 1 bit com as operações AND (00), OR(01), NOT(10) e ADD (11)*



UNIDADE LÓGICA E ARITMÉTICA

- *Uma ULA de quatro bits com as operações AND (00), OR(01), NOT(10) e ADD (11).*
- *Os bits de controle OFL indica um overflow, N indica saída da ULA negativa e Z saída da é zero.*

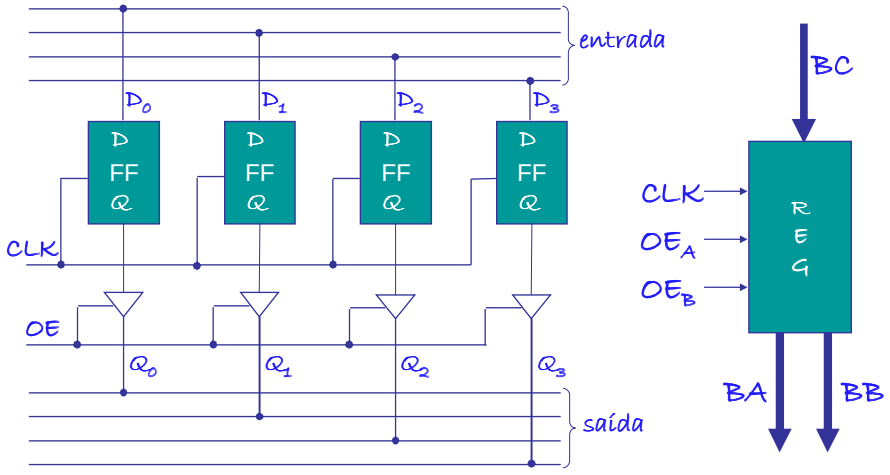


UNIDADE DE CONTROLE

- *A unidade de controle (UC) tem a função específica de executar o micro-programa cujas as instruções permitem controlar os registradores, os barramentos, a ULA, as memórias e outros componentes do hardware.*
- *O micro-programa é armazenado numa memória de controle localizada fisicamente dentro do processador;*
- *Os registradores estão localizados fisicamente dentro do processador.*
- *Um barramento é uma coleção de fios usados para transmitir sinais em paralelo. Pode ser unidirecional ou bidirecional.*

UNIDADE DE CONTROLE

- *Um barramento tri-state possui dispositivos capazes de apresentar na saída 0, 1 ou alta impedância Z; utilizados quando há muitos dispositivos ligados a um mesmo barramento.*

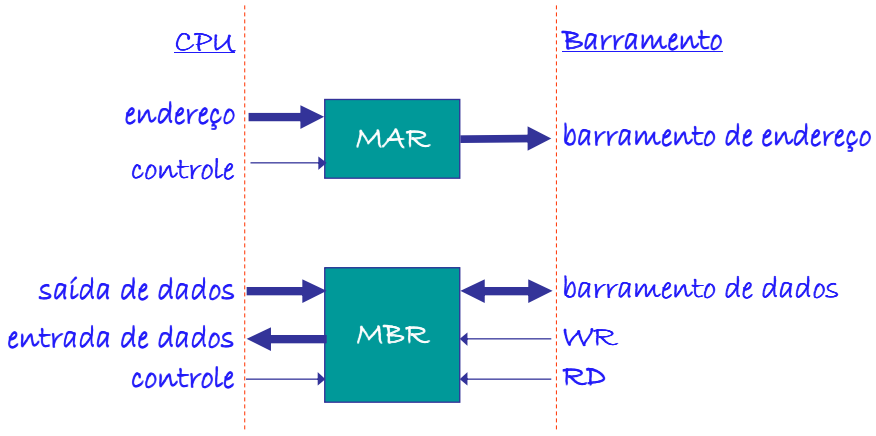


UNIDADE DE CONTROLE

- *A maioria dos computadores tem um barramento de endereços, um barramento de dados e um barramento de controle para a comunicação entre a UCP e a memória.*
- *Um acesso à memória é quase sempre consideravelmente mais demorado que o tempo necessário para executar uma única micro-instrução.*
- *O registrador MAR (Memory Access Register) é responsável pelo armazenamento do endereço da memória.*
- *O registrador MBR (Memory Buffer Rgister) é responsável pelo armazenamento do dado.*

UNIDADE DE CONTROLE

- *A linha de controle de MBR permite carregar o registrador com dado da UCP. O sinal RD (Read) permite carregar o MBR com dado do barramento. O sinal WR (Write) permite liberar o conteúdo do registrador no barramento.*



UNIDADE DE CONTROLE

- *Há 16 registradores de 16 bits que formam uma memória local, acessível apenas ao micro-programa.*
- *Os barramentos A e B alimentam uma ULA de largura de 16 bits, que executa quatro funções: A+B, A.B, A, NOT A.*
- *A ULA fornece dois bits de status, de acordo com a saída atual: N e Z.*
- *O deslocador permite deslocar um bit à esquerda, um à direita ou não deslocar.*
- *Os circuitos de latch L_A e L_B permitem manter os dados estáveis durante o ciclo de operação.*
- *O MAR é carregado a partir do latch B, em paralelo com uma operação da ULA.*

UNIDADE DE CONTROLE

UNIDADE DE CONTROLE

- *Durante uma escrita, o MBR é carregado com dado do deslocador, em paralelo ou ao invés de uma escrita na memória local.*
- *Durante uma leitura, o dado lido da memória é passado pelo lado esquerdo da ULA via um multiplexador.*
- *Para controlar as vias de dados, precisamos de 61 sinais, que correspondem ao número de bits da micro-instrução.*
- *À custa de um aumento na lógica de controle, pode-se reduzir o número de bits necessários ao controle da via de dados através da codificação dos campos da micro-instrução.*

UNIDADE DE CONTROLE

- *Um formato de micro-instrução, contendo alguns campos codificados, pode ser:*

1	2	2	2	1	1	1	1	1	4	4	4	8
AMUX	COND	ALU	SH	MBR	MAR	RD	WR	ENC	C	B	A	ADDR

AMUX : 0 ⇒ latch A; 1 ⇒ MBR

COND: 0 ⇒ não salta; 1 ⇒ salta se N = 1;
2 ⇒ salta se Z = 1; 3 ⇒ salta sempre

ALU: 0 ⇒ A+B; 1 ⇒ A.B; 2 ⇒ A; 3 ⇒ NOT A

SH: 0 ⇒ não desloca; 1 ⇒ desloca 1 bit à direita;
2 ⇒ desloca 1 bit à esquerda; 3 ⇒ não usada.

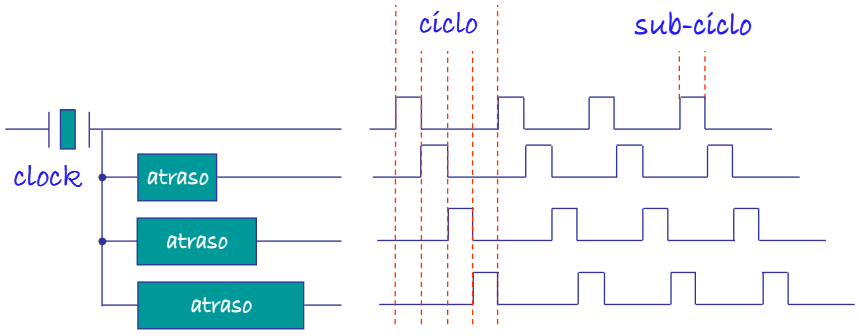
- *Para os campos de um bit: 0 ⇒ desativado e 1 ⇒ ativado.*

UNIDADE DE CONTROLE

- *Um ciclo básico consiste em colocar os valores nos barramentos A e B, armazená-los nos dois circuitos de latch, passá-los pela ULA e pelo deslocador, e armazená-los na memória local ou no MBR.*
- *Para conseguir a seqüência correta dos eventos utiliza-se o relógio de quatro ciclos:*
  1. *carregar a próxima micro-instrução no registrador de micro-instrução (μIR);*
  2. *colocar o conteúdo dos registros nos barramentos A e B, e guardá-los nos circuitos de latch A e B;*
  3. *dar tempo à ULA e ao deslocador para produzirem um resultado e carregar o MAR, se necessário;*
  4. *armazenar o valor existente no barramento C, na memória local (registradores) ou no MBR.*

UNIDADE DE CONTROLE

- *Um circuito de clock emite uma seqüência periódica de pulsos, os quais definem os ciclos da máquina. Durante cada ciclo ocorre alguma atividade básica, como a execução de uma micro-instrução.*
- *A divisão em sub-ciclos permite executar partes da micro-instrução em uma determinada ordem.*



UNIDADE DE CONTROLE

- *A memória de controle guarda as micro-instruções.*
- *O MAR da memória de controle é o μPC (micro-program counter), cuja função é apontar para a próxima micro-instrução a ser executada.*
- *O MBR da memória de controle é o μIR (micro-instruction register), cuja função é armazenar a micro-instrução atual.*
- *Para permitir saltos condicionais no micro-programa há dois campos: ADDR e COND.*
  - *ADDR: contém o endereço de uma possível micro-instrução.*
  - *COND: contém o código que determina a condição de desvio.*

UNIDADE DE CONTROLE

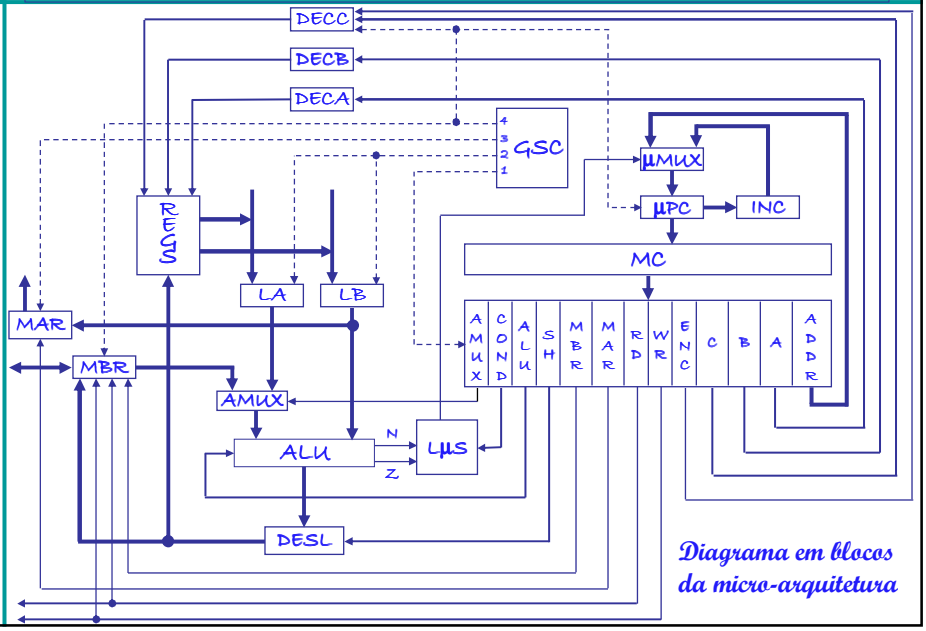


Diagrama em blocos da micro-arquitetura

UNIDADE DE CONTROLE

- *A escolha da próxima micro-instrução é determinada pela lógica de micro-seqüenciamento, durante* T_4*, quando* N *e* Z *são válidos*
- *As condições de desvio são:*
  - COND = 0 ⇒ *não salte (a próxima micro = instrução está em* MPC + 1*);*
  - COND = 1 ⇒ *desvie para* ADDR*, se* N = 1*;*
  - COND = 2 ⇒ *desvie para* ADDR*, se* Z = 1*;*
  - COND = 3 ⇒ *desvie incondicionalmente para* ADDR*.*
- *A lógica de micro-seqüenciamento combina os bits* N, Z *e os dois bits de* COND (C_1 *e* C_0)*:*

$$\mu Mux = \overline{C_1}.C_0.N + C_1.\overline{C_0}.Z + C_1.C_0 = C_0.N + C_1.Z + C_1.C_0$$

UNIDADE DE CONTROLE

UNIDADE DE CONTROLE

UNIDADE DE CONTROLE

- *A macro-arquitetura consiste em uma memória de 4096 palavras de 16 bits, registradores PC, AC, SP e endereçamento direto, indireto e local.*
- *A micro-linguagem de montagem consiste nos seguintes comandos:*
  - *atribuição:* `AC := A;`
  - *aritmética:* `PC := PC + 1;`
  - *lógica:* `A := band (IR,SMASK);`
    `B := inv (C);`
  - *deslocamento:* `TIR:= lshift (TIR);`
    `D := rshift (A + B)`
  - *desvios:* `goto 0;`
    `if N then goto 50`

UNIDADE DE CONTROLE

- *O micro-programa, para a arquitetura proposta, deve realizar a busca, decodificação e execução da instrução do programa de nível convencional de máquina.*
- *O registrador AMASK é a máscara de endereço (0x007777) usada para separar os bits de endereço do restante da instrução.*
- *O registrador SMASK é a máscara de pilha (0x000377) usada para separar a constante, associada às instruções INSP e DESP, do restante da instrução.*
- *Para realizar uma subtração utiliza-se complemento a dois*

UNIDADE DE CONTROLE

- *Em muitos computadores, a micro-arquitetura tem suporte de hardware para extrair código de operação da macro-instrução e colocá-lo diretamente no μPC.*
- *Não há instruções de E/S, utilizando E/S mapeada em memória.*

  (4092): dado a ser lido

  (4093): o bit de sinal indica dado para leitura (= 1)

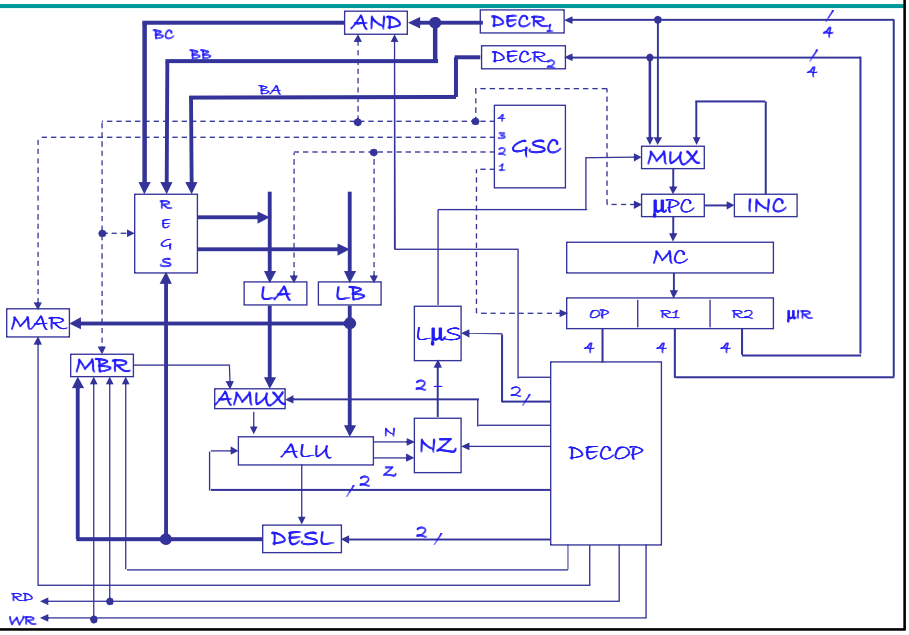
  (4094): dado a ser escrito

  (4095): o bit de sinal indica dispositivo pronto (= 1)

UNIDADE DE CONTROLE

- *Micro-programação horizontal: micro-instrução com campos muito pouco codificados.*
- *Micro-programação vertical: micro-instrução com campos mais codificados.*
- *Um micro-programa é mais vertical quanto maior for o grau de codificação da micro-instrução.*
- *Uma micro-instrução extremamente vertical poderia ter um código de operação e operandos.*

UNIDADE DE CONTROLE



UNIDADE DE CONTROLE

	OPCODE	ALU_H	ALU_L	SH_H	SH_L	NZ	AMUX	AND	MAR	MBR	RD	WR	MSL_H	MSL_L
0	ADD					1		1						
1	AND		1			1		1						
2	MOVE	1				1		1						
3	COMPL	1	1			1		1						
4	LSHIFT	1		1		1		1						
5	RSHIFT	1			1	1		1						
6	GETMBR	1				1	1	1						
7	TEST	1				1								
8	BEGRD	1							1		1			
9	BEGWR	1							1	1		1		
10	CONRD	1									1			
11	CONWR	1										1		
12														
13	NJUMP	1												1
14	ZJUMP	1											1	
15	UJUMP	1											1	1

MEMÓRIA CACHE

- *Apesar da evolução tecnológica, as CPUs continuam mais rápidas que a memória.*
- *O problema não é tecnológico, mas econômico.*

pequena quantidade de memória rápida

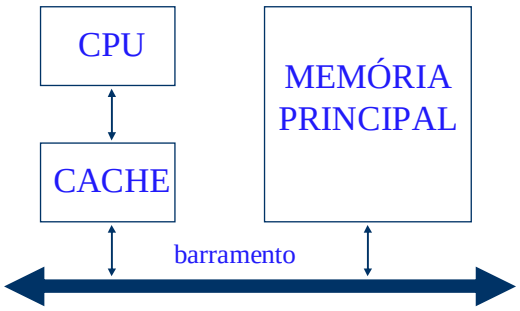
⇔

grande quantidade de memória lenta

- *A memória pequena e rápida é chamada cache e está sob o controle do microprograma*

MEMÓRIA CACHE

- *Se uma dada referência à memória é para o endereço A, é possível que a próxima referência à memória seja nas vizinhanças de A.*
- *O princípio da localidade consiste em que referências à memória, feitas em qualquer intervalo curto de tempo, tendem a usar apenas uma pequena fração da memória total.*



MEMÓRIA CACHE

- *Considerando:*

k:	*nº de vezes que uma palavra é referenciada*
c:	*tempo de acesso ao cache*
m:	*tempo de acesso à memória principal*
h:	*taxa de acerto ("hit ratio")*
$1 - h$:	*taxa de falha ("miss ratio")*

tem-se:

$$h = (k - 1) / k$$

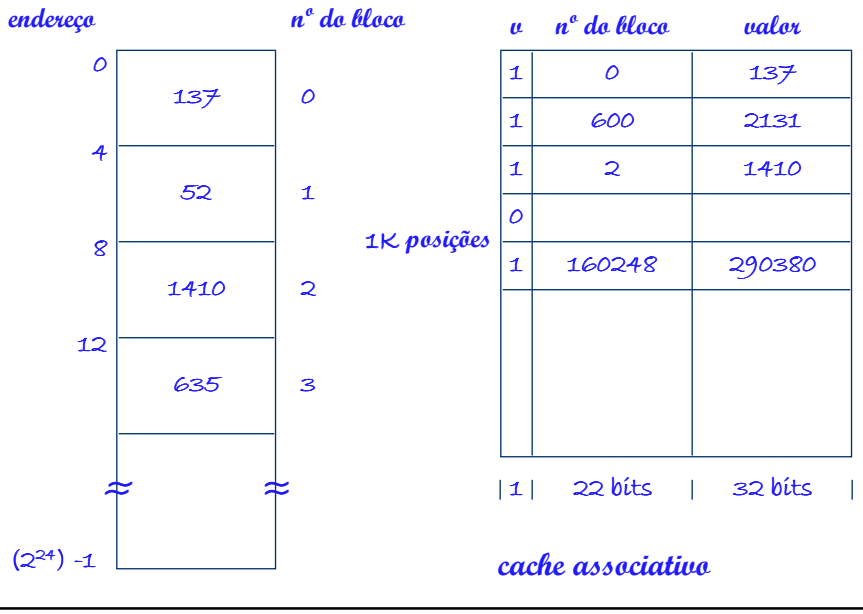
e

tempo médio de acesso $= c + (1 - h)\, m$

MEMÓRIA CACHE

- A memória, de 2^n bytes, é dividida em blocos consecutivos de b bytes, totalizando $(2^n)/b$ blocos.
- Cada bloco tem um endereço, que é um múltiplo de b, e o tamanho do bloco é, normalmente, uma potência de 2.
- O cache associativo apresenta um número de posições ("slots" ou linhas), cada uma contendo um bloco e seu número de bloco, junto com um bit dizendo se aquela posição está em uso ou não.
- A ordem das entradas é aleatória.

MEMÓRIA CACHE



MEMÓRIA CACHE

- *Se o cache estiver cheio, uma entrada antiga terá que ser descartada para deixar lugar para uma nova.*
- *Quando aparece um endereço de memória, o microprograma deve calcular o número do bloco e, então, procurar aquele número no cache.*
- *Para evitar a pesquisa linear, o cache associativo tem um hardware especial que pode comparar simultaneamente todas as posições com o número do bloco dado, ao invés de um "loop" do microprograma. Este hardware torna o cache associativo caro.*
- *No cache com mapeamento direto, cada bloco é colocado numa posição, cujo número pode corresponder, por exemplo, ao resto da divisão do número do bloco pelo número de posições.*

MEMÓRIA CACHE

posição	v	tag	valor	endereços
0	1	0	137	0, 4096, 8192, 12288, ...
1	1	600	2131	4, 4100, 8196, 12292, ...
2	1	2	1410	8, 4104, 8200, 12296, ...
3	0			
1023	0			4092, 8188, 12284, 16380, ...

cache com mapeamento direto

- *Um problema com o cache com mapeamento direto é identificar a palavra que está ocupando uma dada posição.*

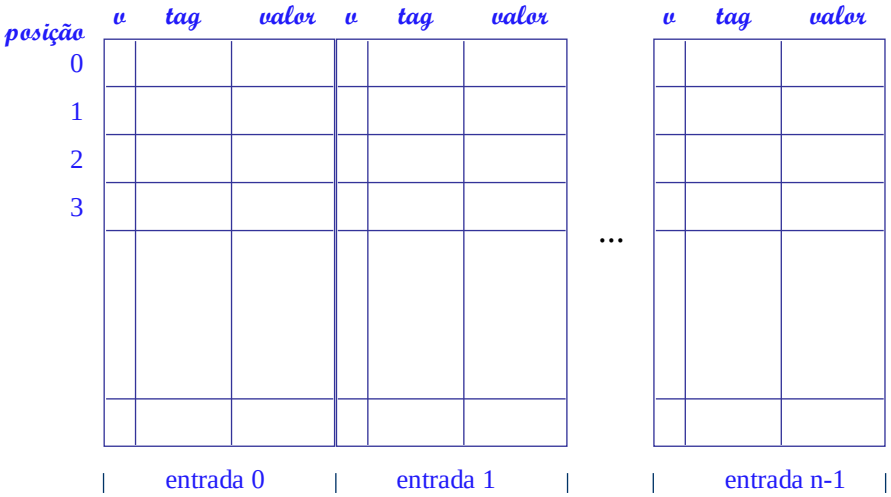
MEMÓRIA CACHE

- *A solução está em criar um campo tag no slot, que guarda a parte do endereço que não participa do endereçamento da posição.*
- *Ex: Seja a palavra no endereço 8192.*

- *Os dois bits menos significativos são 0, pois os blocos são inteiros e múltiplos do tamanho do bloco (4 bytes).*
- *O fato de que blocos múltiplos mapeiam na mesma posição pode degradar o desempenho do cache se muitas palavras que estiverem sendo usadas mapeiem na mesma posição.*

MEMÓRIA CACHE

- *No cache associativo por conjunto utiliza-se um cache de mapeamento direto com múltiplas entradas por posição.*



MEMÓRIA CACHE

- *Tanto o cache associativo quanto o de mapeamento direto são casos especiais do cache associativo por conjunto.*
- *O cache de mapeamento direto é mais simples, mais barato e tem tempo de acesso mais rápido.*
- *O cache associativo tem uma taxa de acerto maior para qualquer dado número de posições, pois a probabilidade de conflitos é mínima.*
- *Uma vantagem de se usar um tamanho de bloco com mais de uma palavra é que existe menos "overhead" na busca de um bloco de oito palavras do que na busca de oito blocos de uma palavra.*

MEMÓRIA CACHE

- *Uma desvantagem é que nem todas as palavras podem ser necessárias, de modo que algumas das buscas podem ser desperdiçadas.*
- *Uma técnica para manipular escritas é denominada write through, quando uma palavra é escrita de volta na memória imediatamente após ter sido escrita no cache (consistência de dados).*
- *Outra técnica é denominada copy back, em que a memória só é atualizada quando a entrada é expurgada do cache para permitir que outra entrada tome conta da posição (consistência de dados).*

MEMÓRIA CACHE

- *A técnica write through causa mais tráfego de barramento.*
- *A técnica de copy back pode gerar inconsistência se a CPU efetuar uma transferência entre memória e disco, enquanto a memória não tiver sido atualizada.*
- *Se a razão de leituras para escritas for muito alta, pode ser mais simples usar write through.*
- *Se houver muitas escritas, pode ser melhor usar copy back e fazer com que o microprograma expurgue todo o cache antes de uma operação de entrada/saída.*