Subsistema de Memória

Luciano L. Caimi

lcaimi@uffs.edu.br

Roteiro

1) Introdução
2) Hierarquia de Memória
3) Memória Principal
4) Memória Cache
5) Memória Virtual

Introdução

- Elemento de armazenamento das informações e programas a serem manipulados.
- Realiza duas operações básicas: leitura e escrita de dados.
- Para escrever ou ler a informação precisamos informar o lugar preciso onde a informação será ou está armazenada: o endereço.

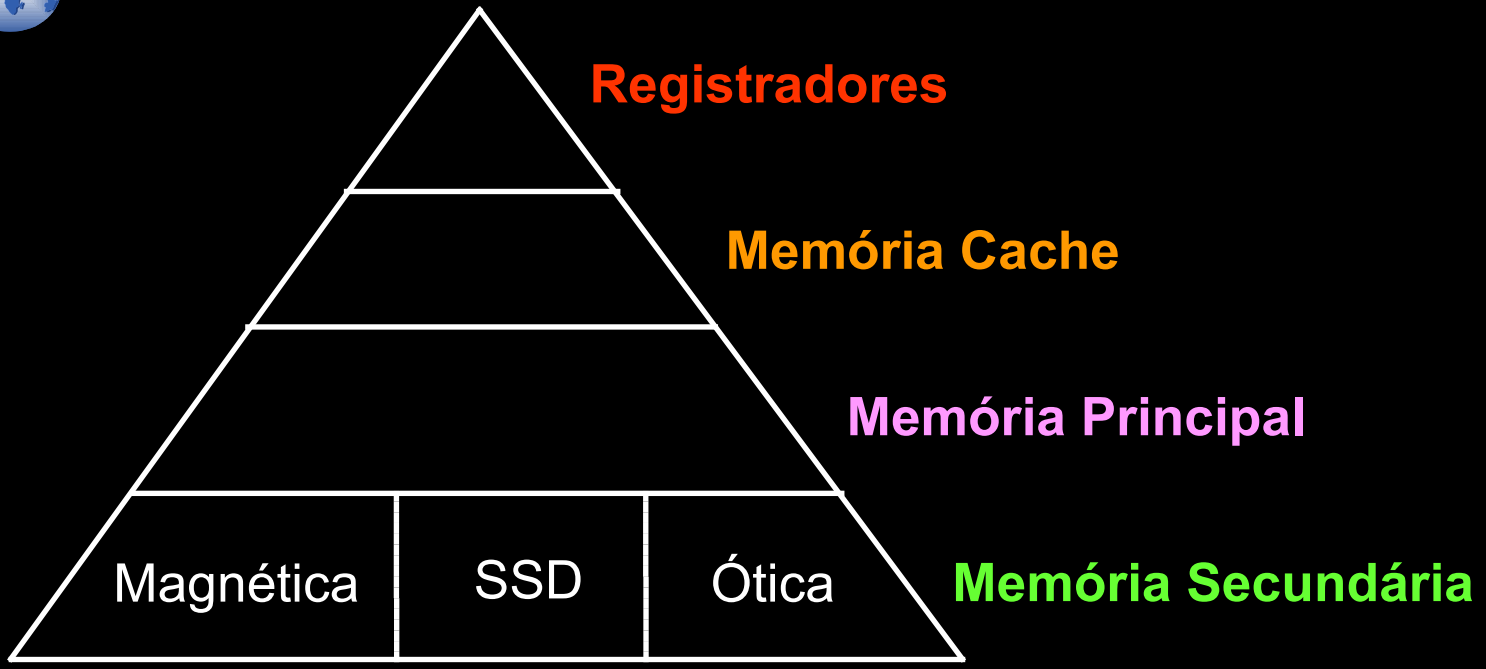
  Assim cada informação tem um local onde o dado está armazenado e um endereço que o localiza.

Introdução

- **Devido a grande variedade de características desejadas e dos diferentes tipos de memória não é possível implementar um sistema de computação com uma única memória.**

- **A memória consiste em um subsistema hierarquizado e estruturado que atende as características necessárias em diferentes níveis do sistema.**

Hierarquia de Memória



- **Características:**

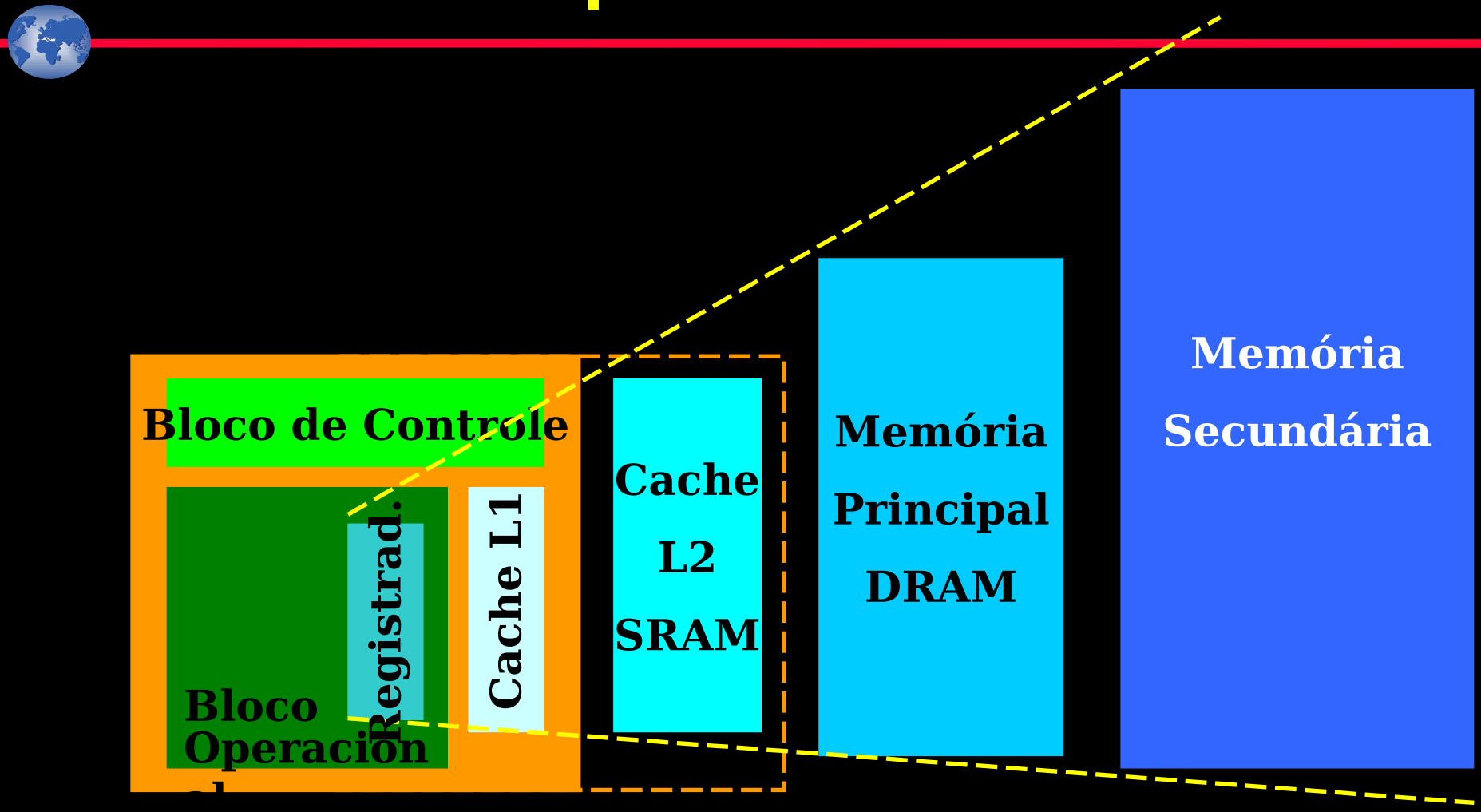
- Tempo de acesso
- Tempo de ciclo
- Capacidade
- Custo
- Freqüência
- Temporariedade
- Tecnologia de Fabricação
- Volatilidade

Hierarquia de Memória

- **Tempo de acesso:** tempo decorrido desde a colocação do endereço até o dado ser disponibilizado;
- **Tempo de ciclo:** tempo decorrido entre duas operações sucessivas de acesso a memória;
- **Freqüência:** freqüência de operação do barramento local (FSB);
- **Capacidade:** quantidade de dados armazenados em bits, bytes, Kbytes, MBytes, GBytes, etc;
- **Temporariedade**: tempo de permanência do dado na memória;
- **Custo:** preço por byte armazenado;
- **Volatilidade:** voláteis – perdem os dados quando acaba a energia

  não-voláteis – não perdem os dados quando acaba a energia. Exemplos: memórias magnéticas, óticas, ROM, PROM, EPROM, EEPROM, Flash

Hierarquia de Memória



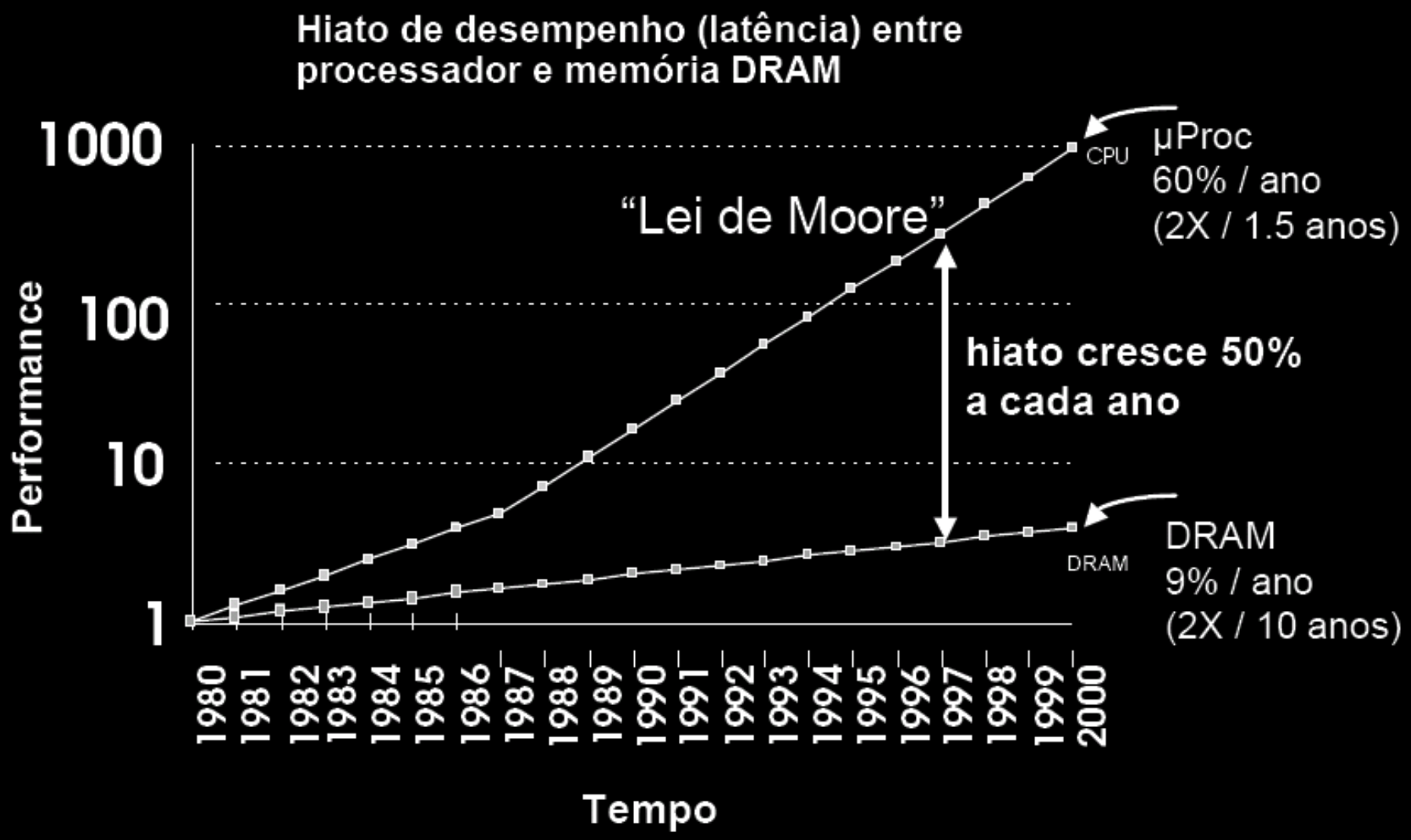
Hierarquia de Memória

3) Performance

	Capacidade	Velocidade
Lógica	2x em 3 anos	2x em 3 anos
DRAM	4x em 3 anos	2x em 10 anos
Disco	4x em 3 anos	2x em 10 anos

DRAM		
Ano	Capacidade	Tempo de Acesso
1980	64 K	250 ns
1983	256 K	220 ns
1986	1 M	190 ns
1989	4 M	165 ns
1992	16 M	145 ns
1995	64 M	120 ns

Hierarquia de Memória



Introdução

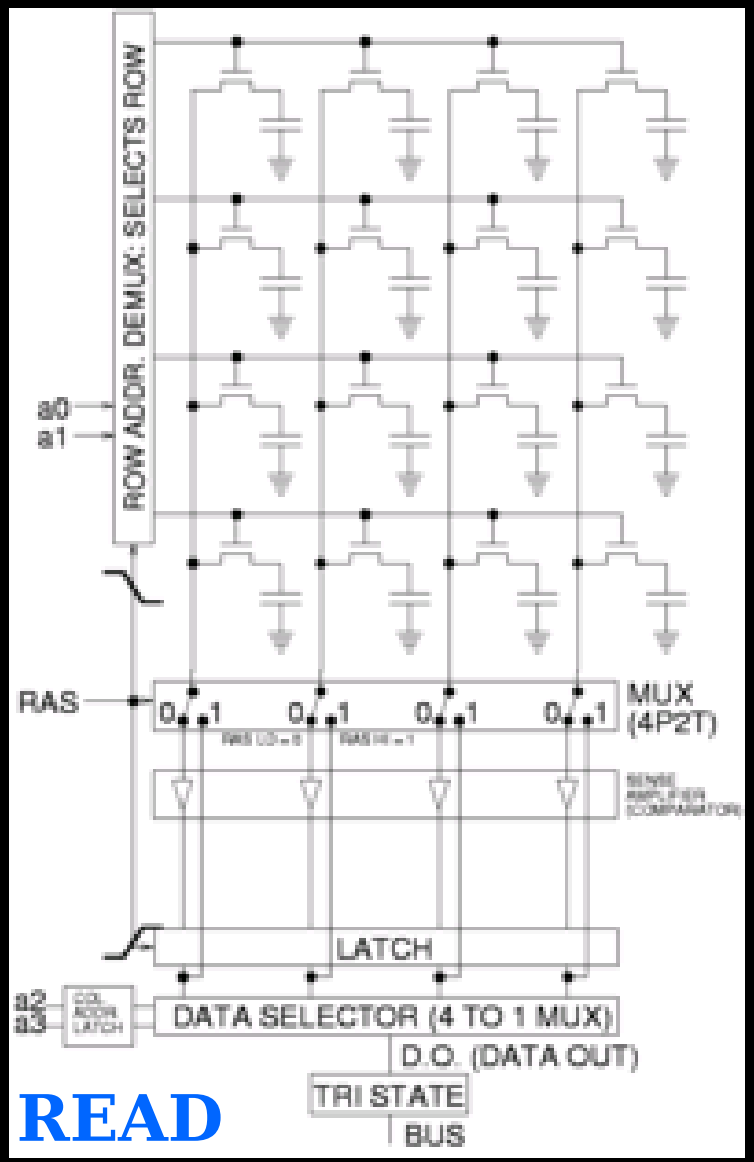
Como a hierarquia é gerenciada?

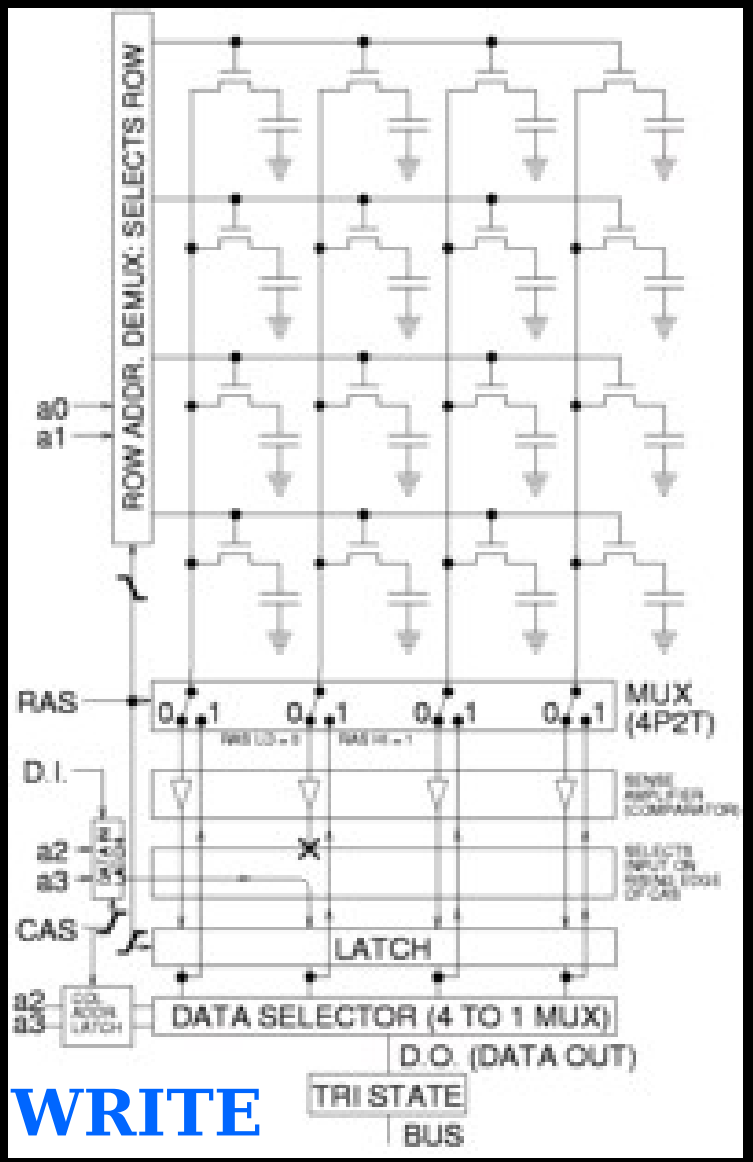
- Registradores ↔ memória
  - pelo compilador
- cache ↔ memória principal
  - pelo hardware
- memória principal ↔ disco
  - pelo hardware e pelo sistema operacional (memória virtual)
  - pelo programador (arquivos)

Memória Principal

- É a memória básica do sistema a partir de onde os programas são executados.
- Atualmente é construída com tecnologia DRAM, onde cada bit é armazenado em um capacitor construído em semicondutor.
- Tempo de retenção da ordem de 50 milisegundos
- Três operações: leitura, escrita, refresh.

DRAM


ROW ADDR. DEMUX: SELECTS ROW
a0
a1
RAS
MUX
(4P2T)
LATCH
a2
a3
DATA SELECTOR (4 TO 1 MUX)
D.O. (DATA OUT)
TRI STATE
BUS
READ


ROW ADDR. DEMUX: SELECTS ROW
a0
a1
RAS
MUX
(4P2T)
D.I.
a2
a3
CAS
LATCH
DATA SELECTOR (4 TO 1 MUX)
D.O. (DATA OUT)
TRI STATE
BUS
WRITE

Memória Principal

- Modo "Burst" - Rajada

  Cada leitura na memória faz 4 acessos consecutivos a memória.

  O primeiro acesso é mais lento e os três acessos seguintes são mais rápidos.

  O tempo para obter o dado é contado em ciclos de barramento, chamados neste caso de "wait states": 8-6-6-6

  ou seja, o primeiro acesso precisou de 8 ciclos de espera e os demais 6 cada um.

Memória Principal

Largura de Banda Máxima Teórica ou
Taxa de Transferência Máxima Teórica

LBMT = Freqüência do Barramento * Largura do Barramento

Considerando:

Frequência de Barramento = 33 MHz

Largura do Barramento = 32 bits

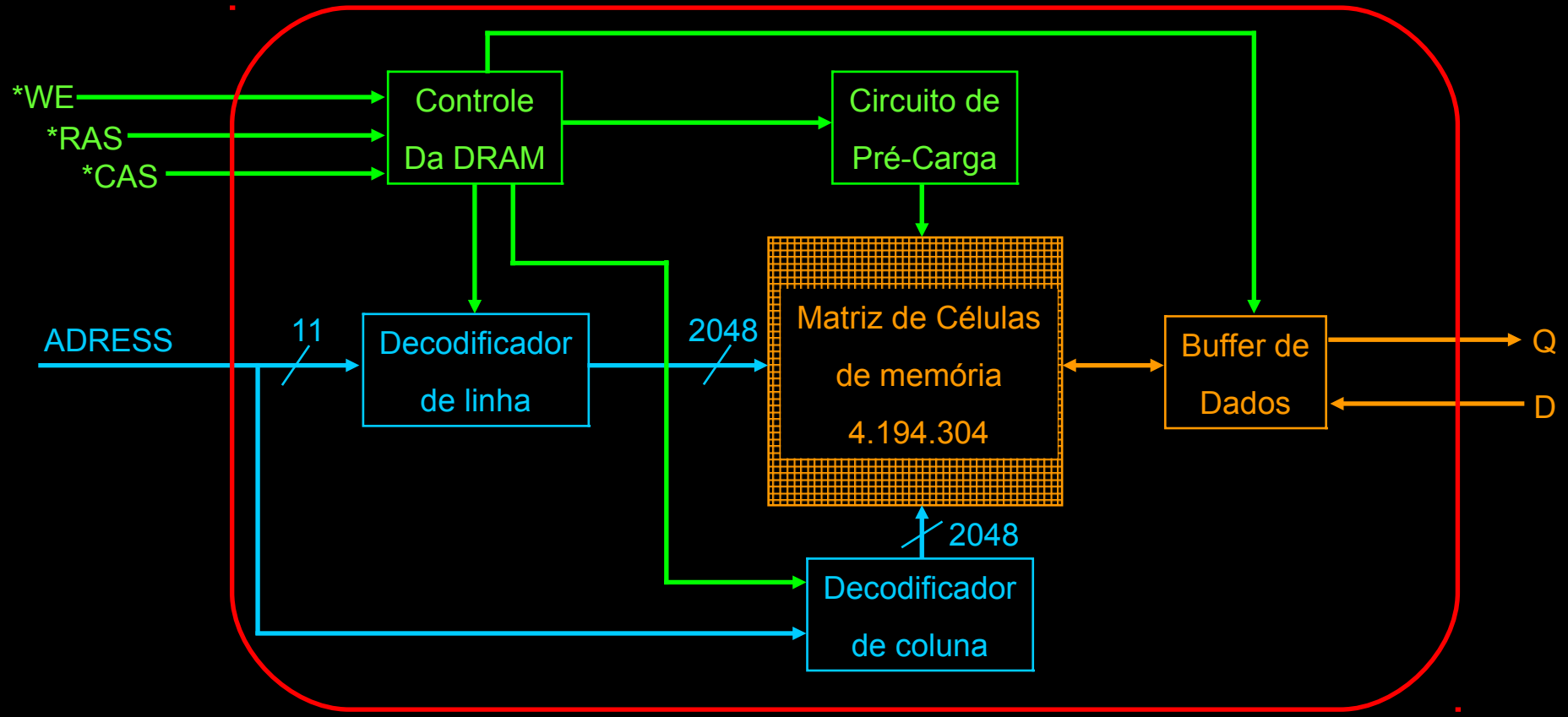
LBMT = $33x10^6$ ciclos/segundo * 32 bits/ciclo

LBMT = 33 * 32 * 10^6 bits/segundo

LBMT = 1056 Mbits/segundo = 132 MBytes/segundo

Memória Principal

- Esquema simplificado de uma DRAM



1 bit por acesso; 4Mbits de dados armazenados; 22 bits de endereço; endereço multiplexado; RAS – Row Address Strobe; CAS – Column Address Strobe; WE – Write Enable;

Memória Principal

- **Fast Page Mode – FPM**

  - Espera que **todos endereços estejam na mesma linha**. Somente reenvia o endereço da coluna;
  - **"wait-states": 5-3-3-3**
  - **LBMT** = 66MHz * 64 bits = 528MBytes/seg
  - No caso da LBMT transfere-se 32 bytes em 4 ciclos de clock
  - No caso da memória FPM p/ transferir 32 bytes gastam-se 5+3+3+3 = 14 ciclos de clock
  - Assim a LB_efetiva é de 4/14 da LBMT ou seja:

    LB_efetiva = 528 MBytes/seg * (4/14)

    LB_efetiva = 150,85 MBytes/seg

Memória Principal

- **Extended Data Output – EDO-RAM**

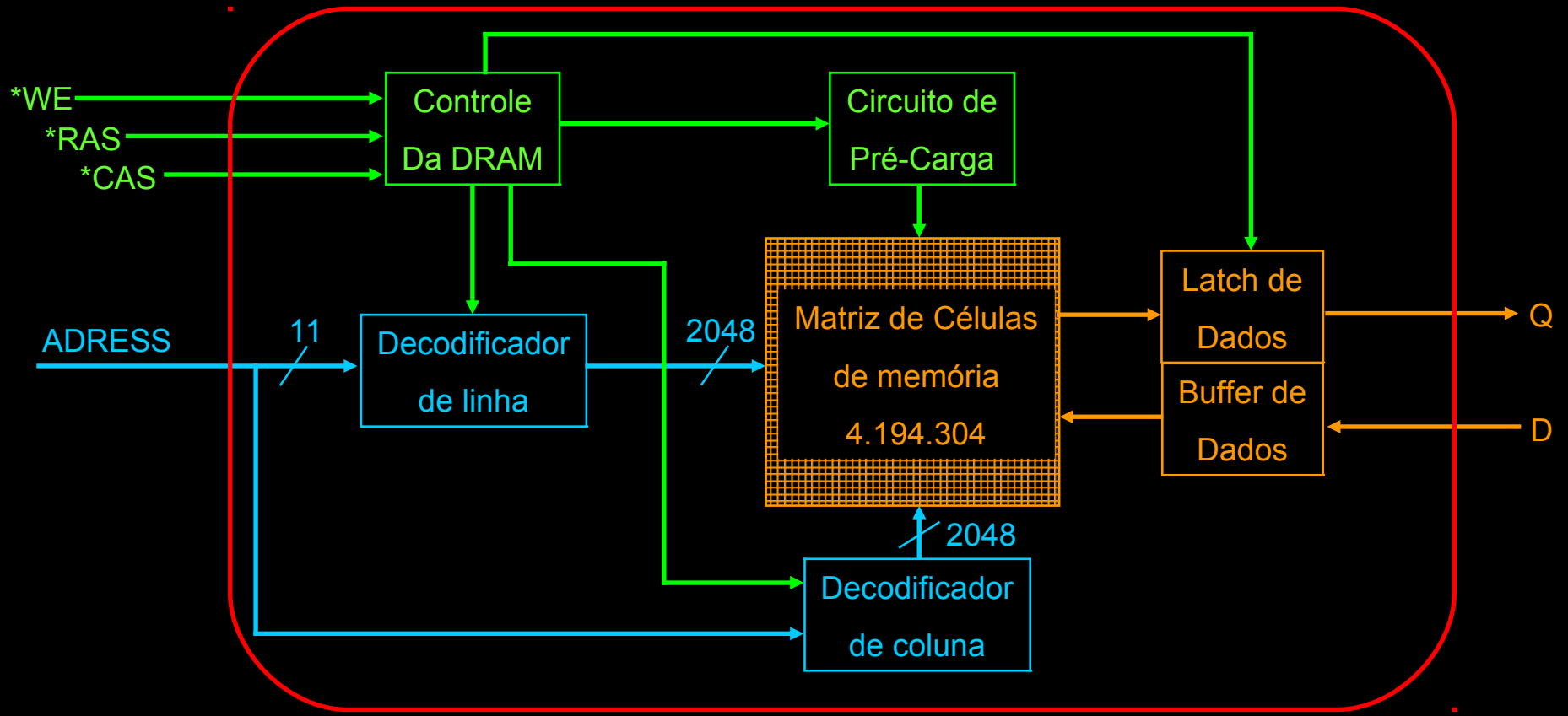
  - Adiciona uma nova latch na saída de dados, liberando o *CAS para enviar um novo endereço de coluna.
  - "wait-states": 5-2-2-2
  - LBMT = 66MHz * 64 bits = 528MBytes/seg
  - A LBMT transfere 32 bytes em 4 ciclos de clock
  - No caso da memória FPM p/ transferir 32 bytes gastam-se 5+2+2+2 = 11 ciclos de clock
  - Assim a LB_efetiva é de 4/11 da LBMT ou seja:

    LB_efetiva = 528 MBytes/seg * (4/11)

    LB_efetiva = 192 MBytes/seg

Memória Principal

- EDO-RAM



Memória Principal

- **Burst Extended Data Output – BEDO-RAM**

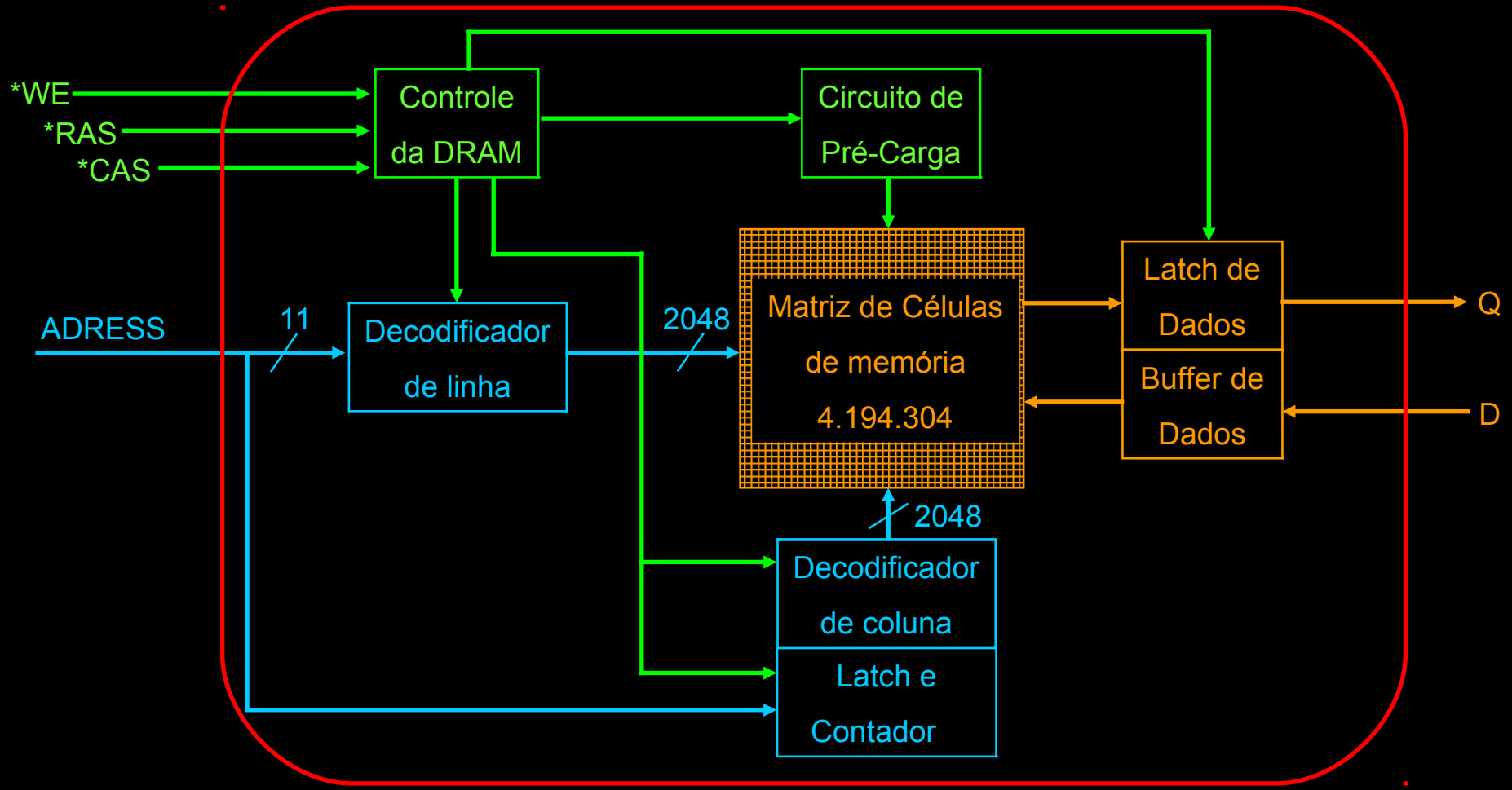
  - Implementa um contador que gera os endereços subseqüentes de coluna. Apenas handshake de CAS.
  - "wait-states": 5-1-1-1
  - LBMT = 100MHz * 64 bits = 800MBytes/seg
  - A LBMT transfere 32 bytes em 4 ciclos de clock
  - No caso da memória FPM p/ transferir 32 bytes gastam-se 5+1+1+1 = 8 ciclos de clock
  - Assim a LB_efetiva é de 4/8 da LBMT ou seja:

    LB_efetiva = 800 MBytes/seg * (4/8)

    LB_efetiva = 400 MBytes/seg

Memória Principal

- **BEDO-RAM**



Memória Principal

- **Syncronous Dinamic RAM – SDRAM**

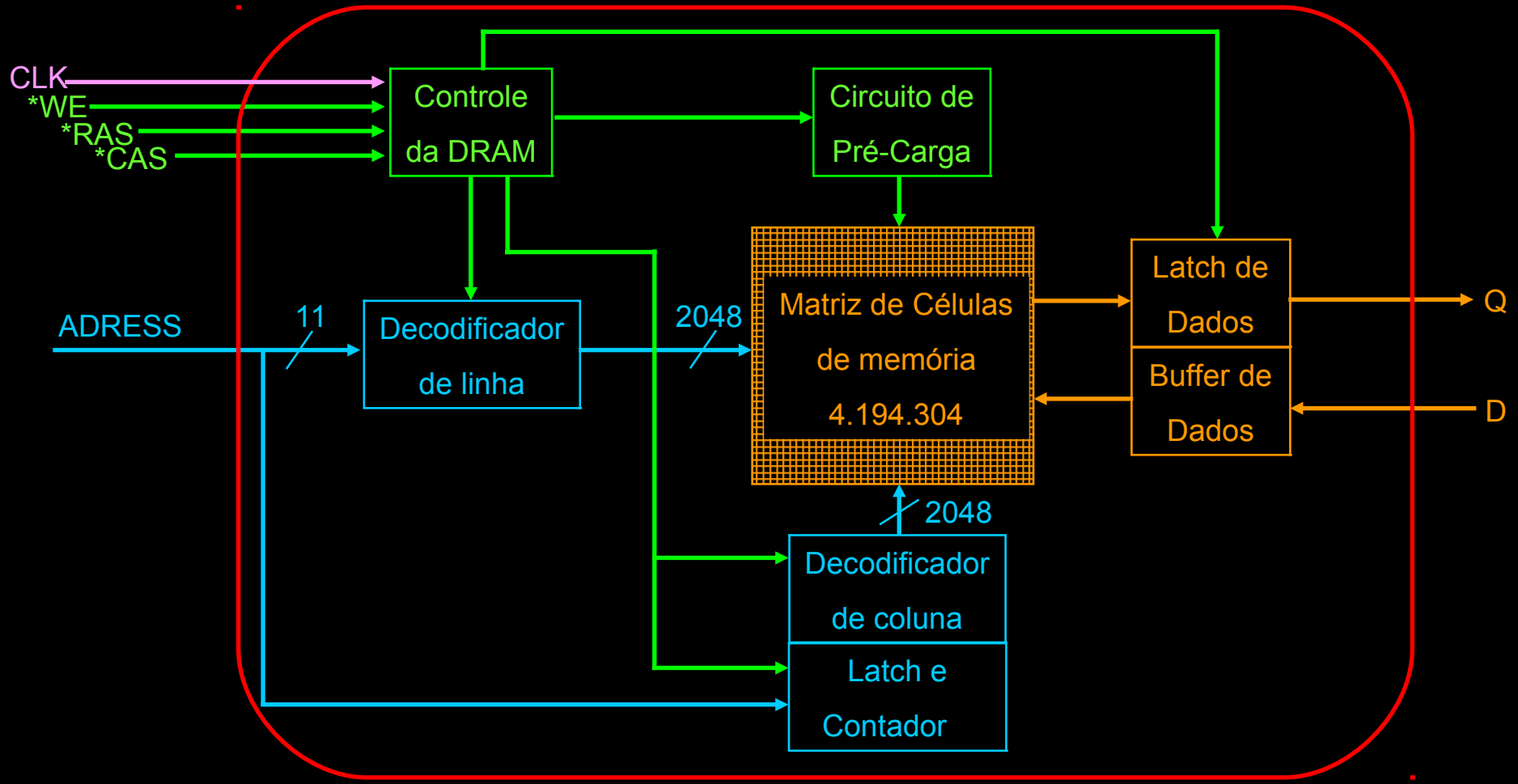
  - Um sinal de clock sincroniza a memória e seu controlador. Não são necessários sinais de handshake subseqüentes pois o clock sincroniza as operações.

  - "wait-states": 5-1-1-1

  - Vantagens: opera em freqüências maiores que a BEDO RAM como 133 MHz, 200 MHz, etc;

Memória Principal

- SDRAM



Memória Principal

- **Double Data Rate SDRAM – DDR SDRAM**

  - É capaz de realizar duas operações internas por ciclo de clock, ou seja, tanto a borda de subida quando a de descida é utilizada.

  - Assim na mesma frequência de clock a taxa de transferência é o dobro;

  - A especificação mostra a freqüência aparente e não a real, ou seja, uma DDR 400 tem freqüência real de 200MHz;

  - Freqüências de 133, 166 e 200MHz;

  - Alimentação é de 2,5 Volts e 184 pinos;

  - Terminação resistiva na placa-mãe;

  - Wait-States:

Memória Principal

- **Double Data Rate 2 SDRAM – DDR2 SDRAM**

- Mesmo princípio de funcionamento da DDR

- Construídas nas freqüências de 200, 266, 333, 400 e 533MHz;

- Possuem um número de ciclos de wait-state maior que as DDR

- Alimentação é de 1,8 Volts e 240 pinos;

- Mesmo tamanho físico que as DDR;

- Possuem a terminação resistiva no próprio módulo de memória;

- Wait-States: 15-5-5-5

Memória Principal

- **Double Data Rate 3 SDRAM – DDR3 SDRAM**

- Mesmo princípio de funcionamento da DDR2

- Freqüências entre 400 e 1066 MHz;

- Buffer interno com o dobro do tamanho;

- Alimentação é de 1,5 Volts;

- Mesmo tamanho físico e quantidade de pinos que as DDR2;

- Wait-States: 20-7-7-7 -> DDR3-1066

24-8-8-8 -> DDR3-1333

Memória Principal

Dual Channel – Utiliza dois canais independentes de acesso a memória (hardware duplicado) dobrando a taxa de transferência;

Disponível nas tecnologias DDR SDRAM e DDR2 SDRAM;

Memória Principal

Memória	Tecnologia	Clock Aparente	Clock Real	Taxa de Transferência Máxima
PC66	SDRAM	66 MHz	66 MHz	533 MB/s
PC100	SDRAM	100 MHz	100 MHz	800 MB/s
PC133	SDRAM	133 MHz	133 MHz	1.066 MB/s
DDR200	DDR-SDRAM	200 MHz	100 MHz	1.600 MB/s
DDR266	DDR-SDRAM	266 MHz	133 MHz	2.100 MB/s
DDR333	DDR-SDRAM	333 MHz	166 MHz	2.700 MB/s
DDR400	DDR-SDRAM	400 MHz	200 MHz	3.200 MB/s
DDR2-400	DDR2-SDRAM	400 MHz	200 MHz	3.200 MB/s
DDR2-533	DDR2-SDRAM	533 MHz	266 MHz	4.264 MB/s
DDR2-667	DDR2-SDRAM	667 MHz	333 MHz	5.336 MB/s
DDR2-800	DDR2-SDRAM	800 MHz	400 MHz	6.400 MB/s

Memória Principal

Standard name	Memory clock (MHz)	Cycle time (ns)	I/O bus clock (MHz)	Data rate (MT/s)	Module name	Peak transfer rate (MB/s)	Timings (CL-tRCD-tRP)	CAS latency (ns)
DDR3-800D DDR3-800E	100	10	400	800	PC3-6400	6400	5-5-5 6-6-6	12 ½ 15
DDR3-1066E DDR3-1066F DDR3-1066G	133⅓	7 ½	533⅓	1066⅔	PC3-8500	8533⅓	6-6-6 7-7-7 8-8-8	11 ¼ 13 ⅛ 15
DDR3-1333F* DDR3-1333G DDR3-1333H DDR3-1333J*	166⅔	6	666⅔	1333⅓	PC3-10600	10666⅔	7-7-7 8-8-8 9-9-9 10-10-10	10 ½ 12 13 ½ 15
DDR3-1600G* DDR3-1600H DDR3-1600J DDR3-1600K	200	5	800	1600	PC3-12800	12800	8-8-8 9-9-9 10-10-10 11-11-11	10 11 ¼ 12 ½ 13 ¾
DDR3-1866J* DDR3-1866K DDR3-1866L DDR3-1866M*	233⅓	4 ²⁄₇	933⅓	1866⅔	PC3-14900	14933⅓	10-10-10 11-11-11 12-12-12 13-13-13	10 ⁵⁄₇ 11 ¹¹⁄₁₄ 12 ⁶⁄₇ 13 ¹³⁄₁₄
DDR3-2133K* DDR3-2133L DDR3-2133M DDR3-2133N*	266⅔	3 ¾	1066⅔	2133⅓	PC3-17000	17066⅔	11-11-11 12-12-12 13-13-13 14-14-14	10 ⁵⁄₁₆ 11 ¼ 12 ³⁄₁₆ 13 ⅛

Memória Principal

- **Temporizações**

Além dos ciclos de wait-states existem outros valores relativos a temporização da memória que são significativos no que diz respeito a sua performance:

- CAS Latency (CL)

  indica a quantidade de pulsos de clock que a memória leva para retornar um dado solicitado;

- RAS to CAS Delay (tRCD)

  indica a quantidade de pulsos de clock que precisa ser respeitado entre a ativação da linha de RAS e a ativação da linha de CAS;

- Active to Precharge Delay (tRAS)

  este parâmetro limita quando a memória pode iniciar a leitura (ou escrita) em uma linha diferente;

Memória Principal

Temporizações...

- **RAS Precharge (tRP)**

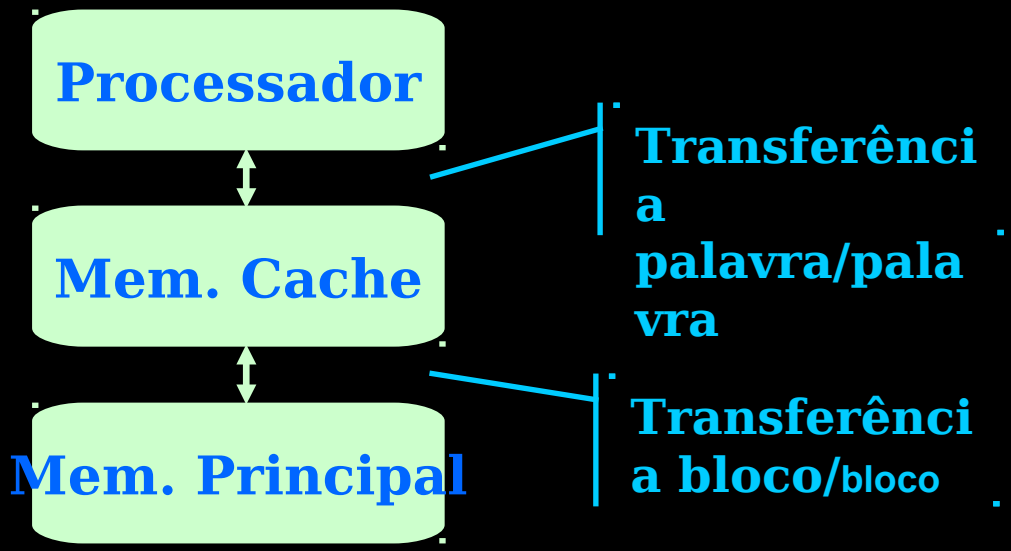
Após o dado ter sido entregue pela memória, um comando chamado Precharge precisa ser executado para desativar a linha da memória que estava sendo usada e para permitir que uma nova linha seja ativada **tRP** é o tempo entre o comando Precharge e o próximo comando "Active" (que inicia a leitura);

- **Command Rate (CMD)**

Tempo demorado entre o chip de memória ter sido ativado (através do seu pino CS – Chip Select) e qualquer comando poder ser enviado para a memória. Este parâmetro leva a letra "T" e normalmente possui o valor T1 ou T2 (1 ou 2 pulsos de clock respectiv.).

Memória Cache

- Memória estática (construída a partir de Flip-Flops) disposta logicamente entre a CPU e a Memória Principal.



- Tempo de acesso entre 2 e 5 ns
- Capacidade de armazenamento pior que a MP (menor quantidade) mas tempo de acesso melhor (mais rápida);

Memória Cache

- **Funcionamento Básico:**

  1) O processador solicita um determinado endereço;

  2) O controlador da memória cache verifica se o endereço está armazenado na cache;

  2.1) Caso sim: ocorre um **acerto** (**hit**) - a cache entrega para o processador a palavra solicitada;

  2.2) Caso não: ocorre uma **falta** (**miss**) – o controlador da cache acessa o nível inferior da hierarquia;

  2.2.1) o bloco onde o endereço solicitado se encontra é carregado na cache;

  2.2.2) a cache entrega a palavra a palavra solicitada ao processador

Memória Cache

- Programas de computador são essencialmente sequênciais (instruções são armazenadas e executadas uma após a outra) e...
- Programas utilizam de maneira extensiva de repetições (laços for, while, repeat...) fazendo com que trechos de código sejam repetidamente executados.
- Estas características fazem parte do conceito de **localidade**, princípio básico de funcionamento das memórias cache.

Memória Cache

- **Dois tipos de localidade:**

  Localidade Espacial

  Se um determinado posição de memória é acessada há uma grande tendência de que o próximo acesso seja em um endereço adjascente, dado a natureza sequencial dos programas.

  Localidade Temporal

  Posições de memória, uma vez acessadas, tendem a ser acessadas novamento no futuro. Basta que a innstrução ou dado esteja em um laço do programa.

Memória Cache

- **Como os princípios de localidade espacial e temporal são utilizados na implementação da cache:**

  Localidade Espacial

  Mover blocos de palavras contiguas para níveis de memória mais próximos do processador.

  Localidade Temporal

  Manter itens de dados mais recentemente acessados nos níveis de hierarquia mais próximos do processador.

Memória Cache

Localidade Temporal

- usualmente encontrada em laços de instruções e acessos a pilhas de dados e variáveis;
- é essencial para a eficiência da memória cache;
- se uma referência é repetida N vezes durante um espaço de programa, após a primeira referência ela estará encontrada na cache;

  Tc = tempo de acesso

  Tm = tempo de acesso a

  Tce = tempo efetivo

Conclusão: uma vez que o dado se encontra na cache, quanto maior o número de acessos, menor o tempo médio de acesso

Se:

Tc = 1 ns;

Tm = 20 ns

então:

N = 10 ⇒ Tce = 3 ns

N = 100 ⇒ Tce = 1,2 ns

Memória Cache

Localidade Espacial

- A memória principal é entrelaçada (1 byte ou 1 palavra por módulo);
- Um quadro (ou linha) é transferida num único acesso a MP
- Há o casamento entre o tempo de acesso da MP e da M. Cache

  Tc = tempo de acesso à cache

  Tm = tempo de acesso à memória principal

  M = Número de módulos da memória principal

Memória Cache

Localidade Espacial

- Tempo médio de acesso a um byte na primeira referência:

- Se cada byte é referenciado N vezes na cache então o tempo efetivo (médio) de acesso a cache Tce a cada byte é:

Se:

Tc = 1 ns;

Tm = 20 ns

então:

N = 10 ⇒ Tce = 1,1 ns

N = 100 ⇒ Tce = 1,01 ns

Memória Cache

Desempenho

- Qual o impacto da taxa de acertos (hit ratio) no tempo de efetivo de acesso?

  Tc = tempo de acesso à cache

  Tm = tempo de acesso à memória principal

  Tce = tempo efetivo de acesso à cache

Se:

Tc = 1 ns;

Tm = 20 ns

então:

h = 0,85 ⇒ Tce = 4 ns

h = 0,95 ⇒ Tce = 2 ns

h = 0,99 ⇒ Tce = 1,2 ns

h = 1,00 ⇒ Tce = 1 ns

Memória Cache

- Tempo gasto com um *cache miss*, em número de instruções executadas

1º Alpha 340 ns / 5.0 ns = 68 clks x 2 instr. ou 136 instruções
2º Alpha 266 ns / 3.3 ns = 80 clks x 4 instr. ou 320 instruções
3º Alpha 180 ns / 1.7 ns = 108 clks x 6 instr. ou 648 instruções

1/2 X latência x 3 X freqüência clock x 3 X instruções/clock ⇒ ≈ 5X

Memória Cache

- Supondo um processador que executa um programa com:
  - CPI = 1.1
  - 50% aritm/lógica, 30% load/store, 20% desvios
- Supondo que 10% das operações de acesso a dados na memória sejam *misses* e resultem numa penalidade de 50 ciclos

CPI = CPI ideal + nº médio de stalls (bolhas) por instrução

= 1.1 ciclos + 0.30 acessos à memória / instrução

x 0.10 misses / acesso x 50 ciclos / miss

= 1.1 ciclos + 1.5 ciclos

= 2. 6

- 58 % do tempo o processador está parado esperando pela memória!
- um miss ratio de 1% no fetch de instruções resultaria na adição de 0.5 ciclos ao CPI médio

Memória Cache

- **Fundamentos**
  - Número de células da memória principal acessíveis ao processador é dado por 2^E onde E é o número de bits de endereço. Por exemplo, se E = 8 bits podemos ter até $2^8 = 256$ células de memória.
  - A memória principal é dividida em B blocos lógicos, cada um deles com K células de memória.
  - Considerando um sistema com E = 4 teremos $2^4 = 16$ células de memória;
  - Considerando B = 8, ou seja, a memória está dividida em 8 blocos lógicos, cada bloco terá 2 células de memória, isto é, K = 2;
  - Assim:

    Número de células = B * K

    ou Número de células = 2^E

    ou $2^E = B * K$

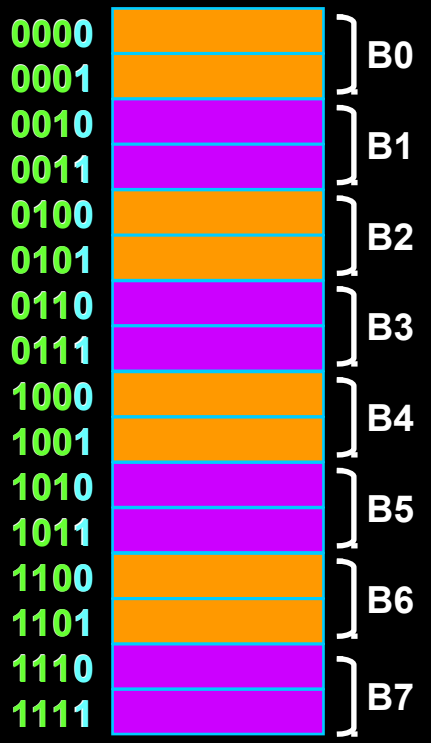
Memória Cache

Número de células = 16

B = 8 ← 3 bits

K = 2 ← 1 bit

E = 4 bits



$2^N = B$

$2^N = K$

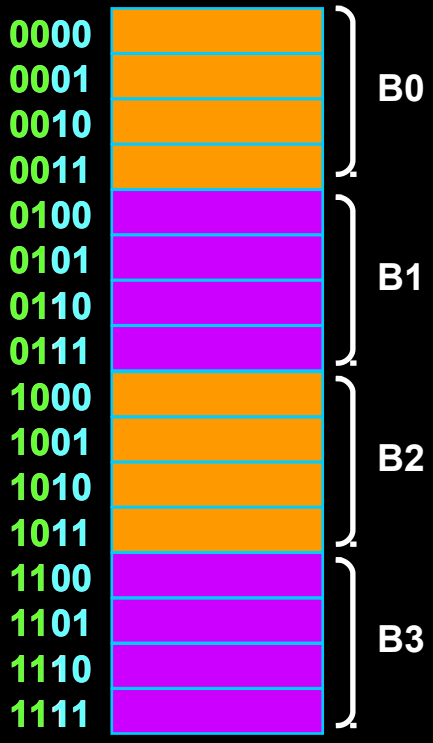
N = Número de Bits

Número de células = 16

B = 4 ← 2 bits

K = 4 ← 2 bits

E = 4 bits



Memória Cache

- Além de trabalhar com binário como anteriormente, podemos definir equações para decimal;
- Assim:

- Por exemplo:

  para B = 8 e K = 2

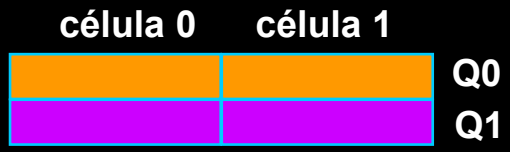
  Endereço = 13

  Número do bloco = 13 / 2 = 6

  Deslocamento no bloco = 13 MOD 2 = 1

Memória Cache

- A memória cache, por sua vez é dividida em linhas (denominadas quadros), que possuem o mesmo tamanho de um bloco (tamanho K);



- Cada vez que ocorre uma falta o controlador da cache irá transferir um bloco inteiro para uma linha (quadro) da cache;
- Algumas questões surgem:

1) Como determinar qual bloco se encontra em um determinado quadro em um instante de tempo qualquer?

2) Qual dado deve sair quando a cache está cheia e uma falta no acesso ocorre?

3) O que fazer quando uma escrita ocorre (coerência X performance)?

Memória Cache

Políticas de Mapeamento

- Mapeamento Direto
- Mapeamento Associativo
- Mapeamento Associativo por Conjuntos

- Definem a maneira como as informações são armazenadas e localizadas na cache;

Memória Cache

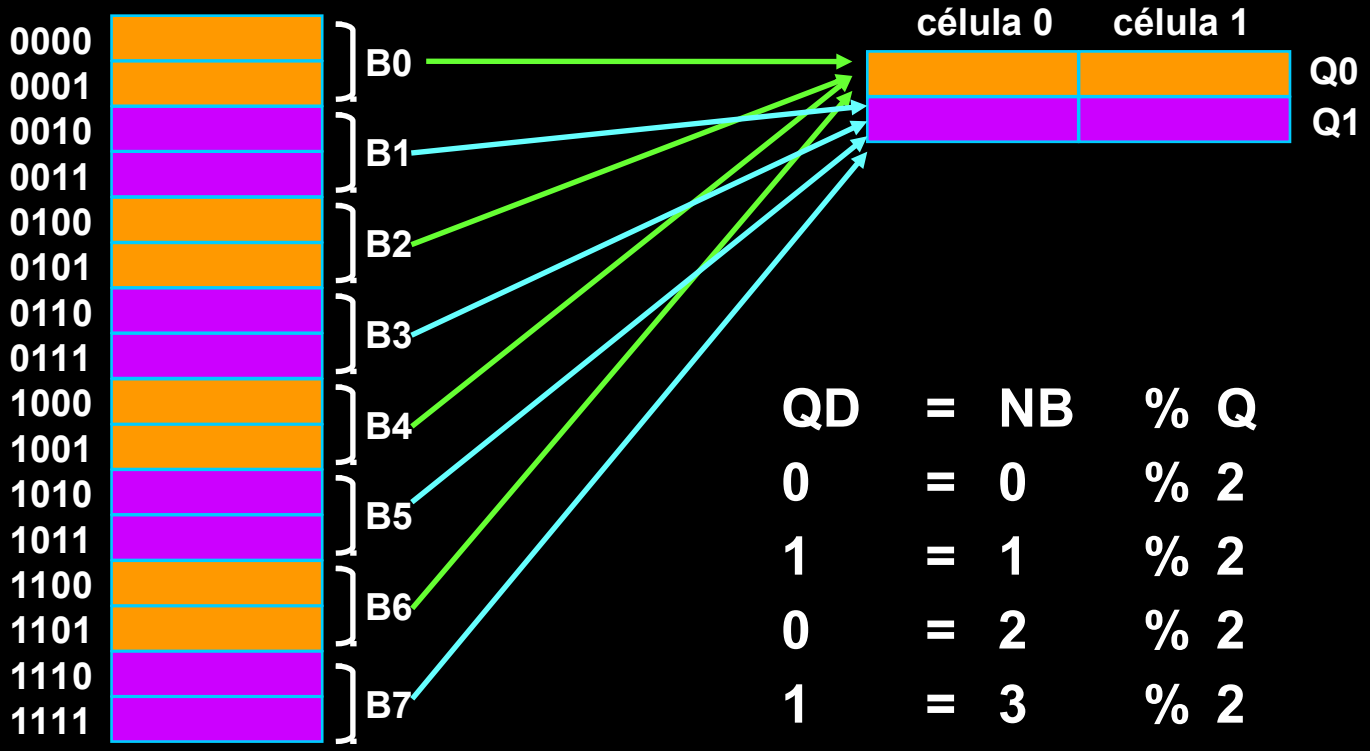
Mapeamento Direto

- Nesta política o quadro em que cada bloco será armazenado é pré-definido:

  QD = NB % Q

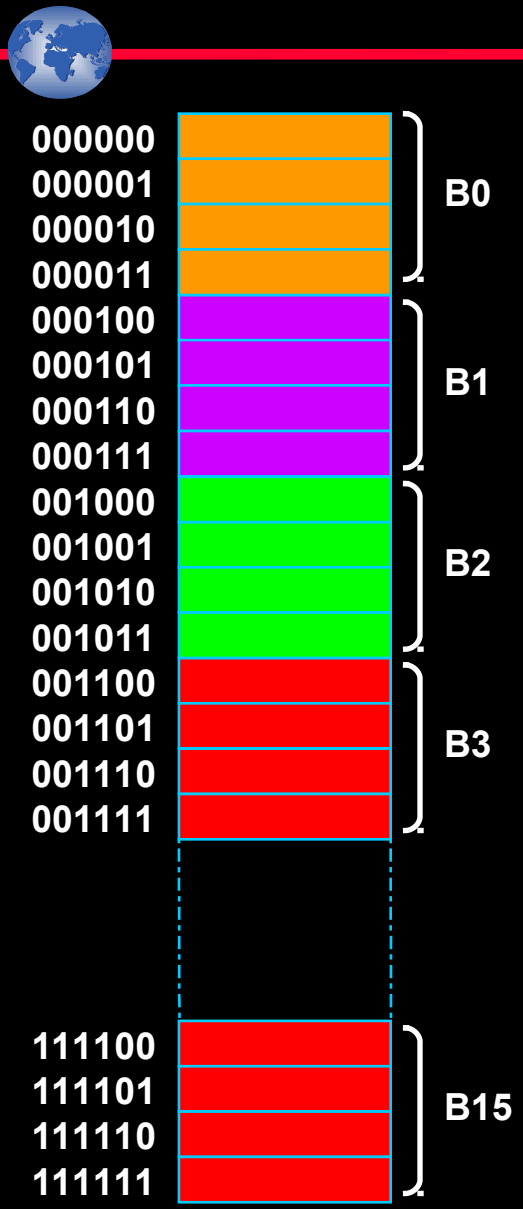
  - QD: Quadro de destino
  - NB: Número do Bloco
  - Q: Quantidade de Quadros

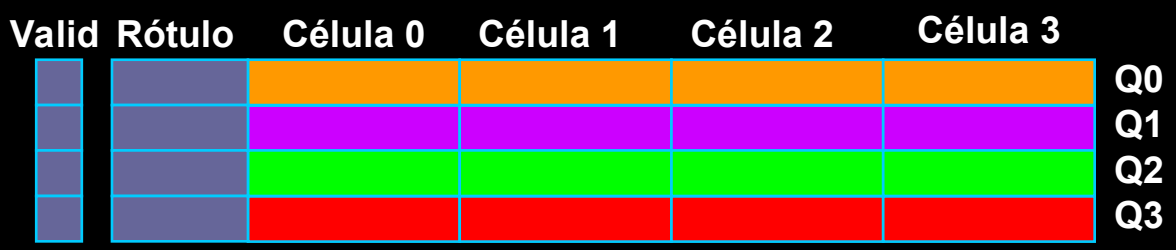
Memória Cache

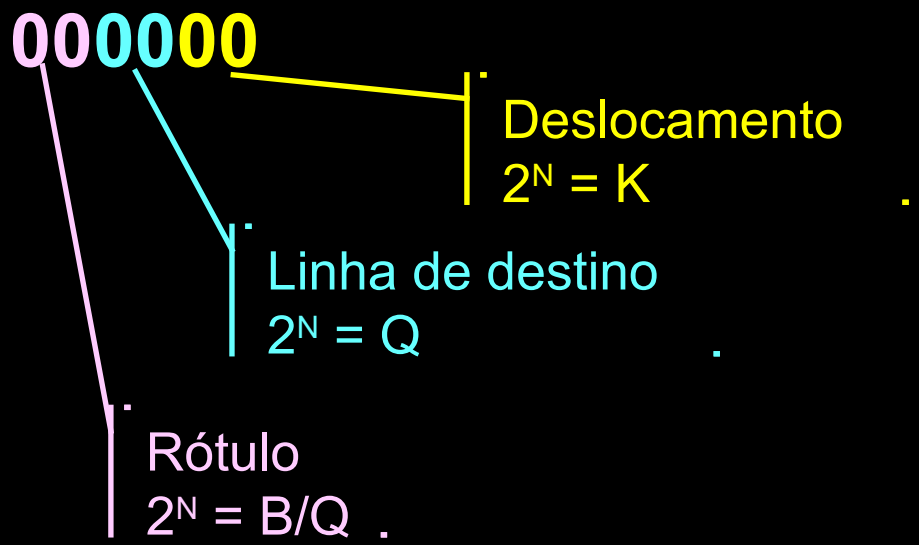


QD	=	NB	%	Q
0	=	0	%	2
1	=	1	%	2
0	=	2	%	2
1	=	3	%	2
0	=	4	%	2
1	=	5	%	2
0	=	6	%	2
1	=	7	%	2

Memória Cache







Memória Cache

Mapeamento Direto

- **Processo completo de leitura na cache com mapeamento direto:**

1. CPU apresenta endereço de 6 bits ao circuito de controle da cache;
   endereço solicitado = 100110_2

2. Os 2 bits centrais são examinados para determinar o quadro de destino:
   $XX01XX_2 \Rightarrow$ Quadro 1.
   Resta verificar se o bloco solicitado encontra-se no quadro 1

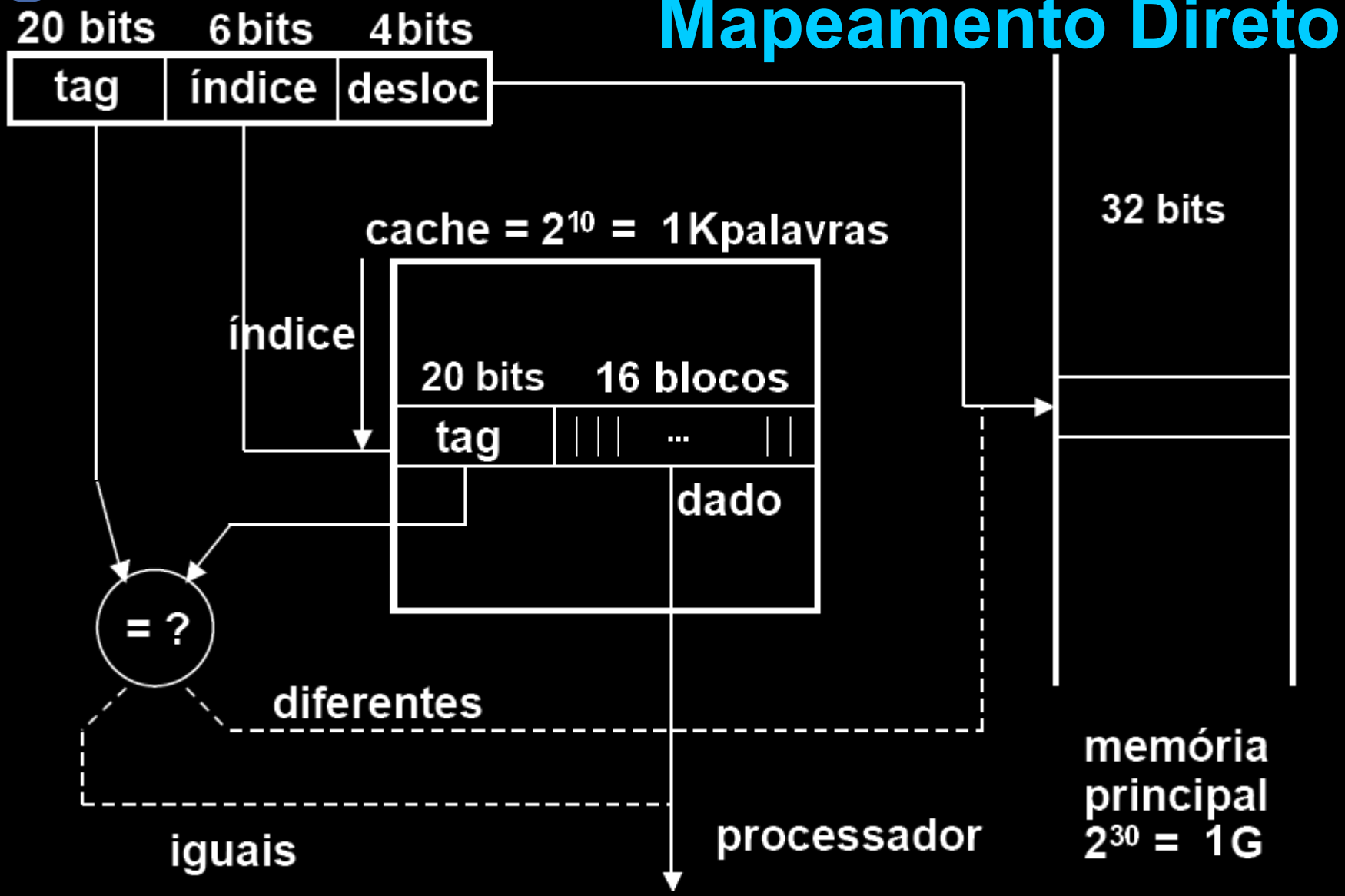
3. O controlador de cache examina por comparação o valor do rótulo do quadro 1 da cache com os 2 bits mais significativos do endereço solicitado ($10XXXX_2$). Caso sejam iguais realiza o passo 4, senão o passo 5;

4. Caso sejam iguais os rótulos a célula de deslocamento 2 - $XXXX10_2$) do quadro 1 tem seu conteúdo transferido para a CPU;

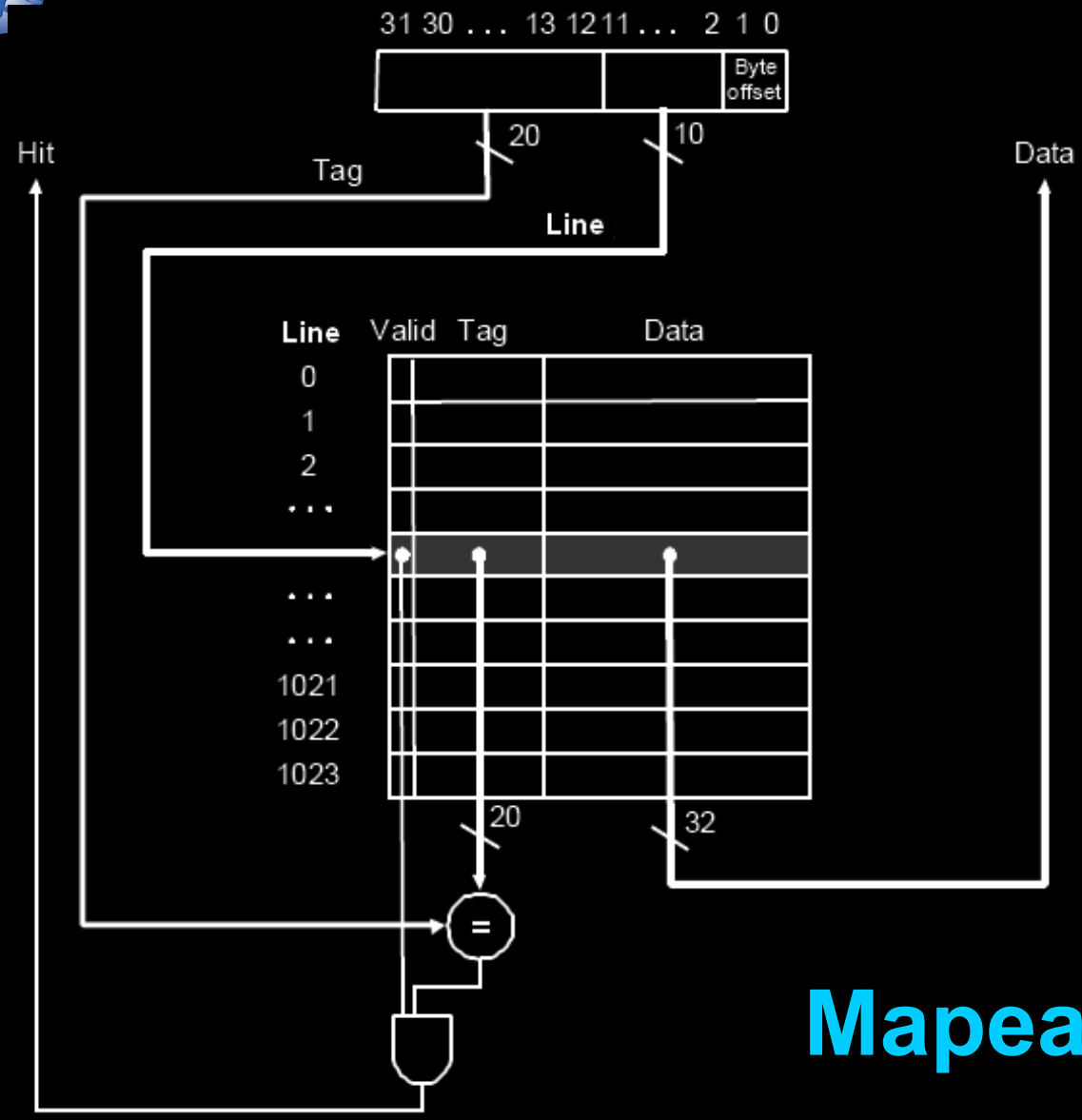
5. Se no passo 3 a comparação resulta negativa (ou seja, o bloco desejado não se encontra no quadro 1), o mesmo será transferido da MP para a cache, substituindo o bloco atual. Para esta última tarefa são utilizados os 4 bits mais significativos do endereço solicitado ($1001XX_2$) correspondentes ao número do bloco desejado.

Memória Cache

Mapeamento Direto



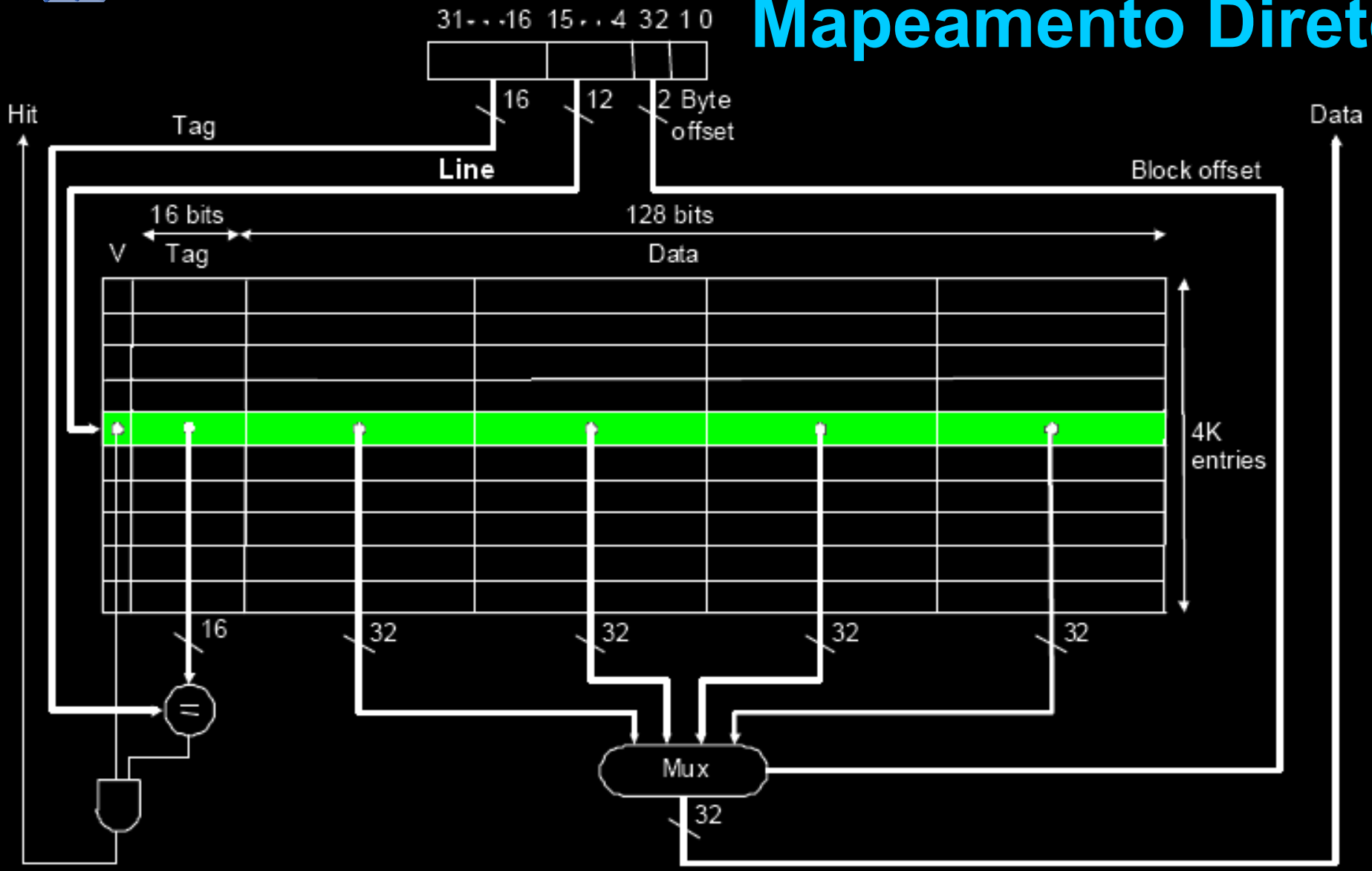
Memória Cache



Mapeamento Direto

Memória Cache

Mapeamento Direto



Memória Cache

Mapeamento Direto

Conclusões:

1) Método simples e de baixo custo de implementação (um multiplexador e um comparador);

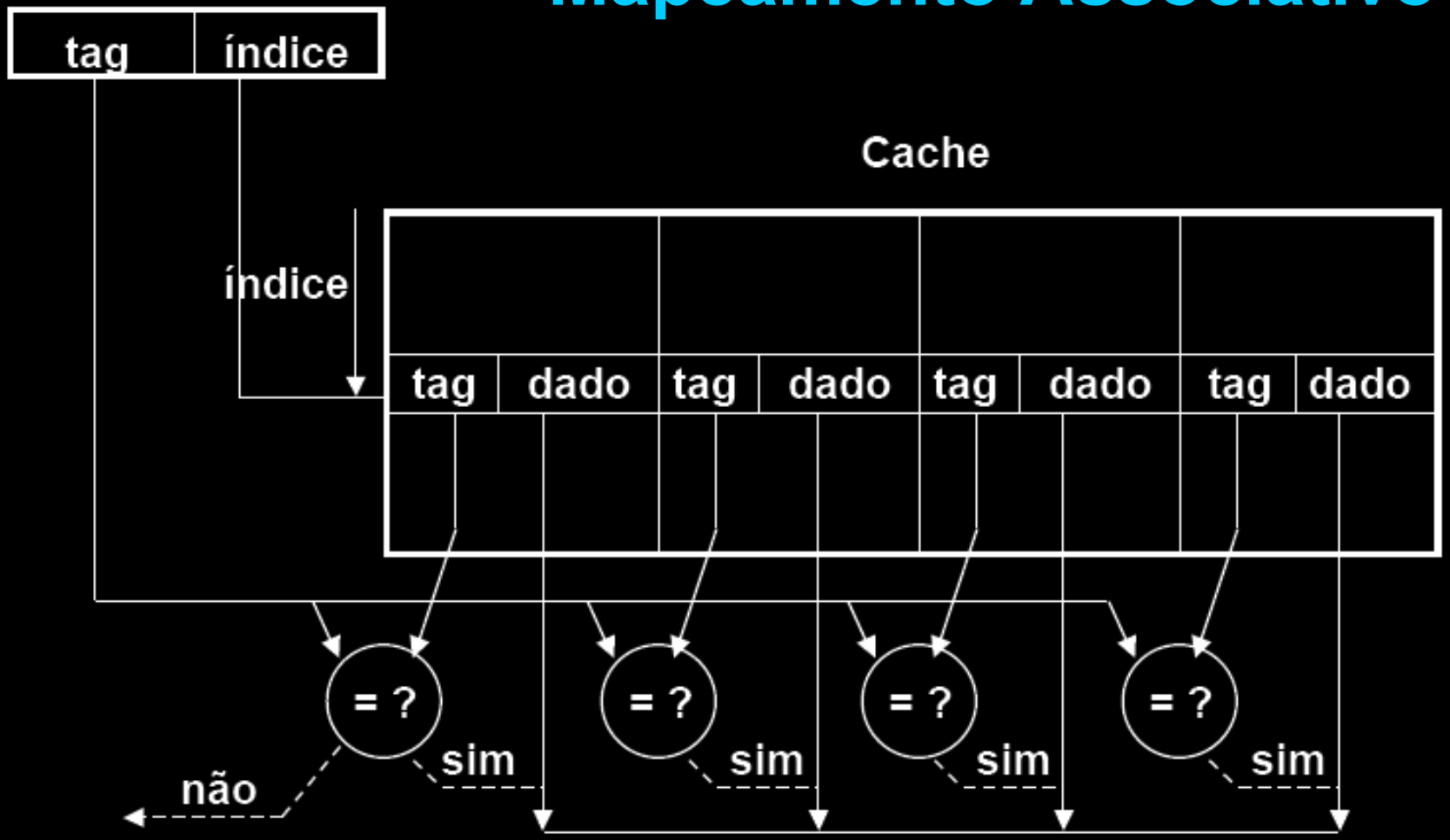
2) Blocos que disputam o mesmo quadro/linha não podem estar na cache concomitantemente;

3) Overhead baixo:

Overhead = Nº Bits política / Nº Bits Dados

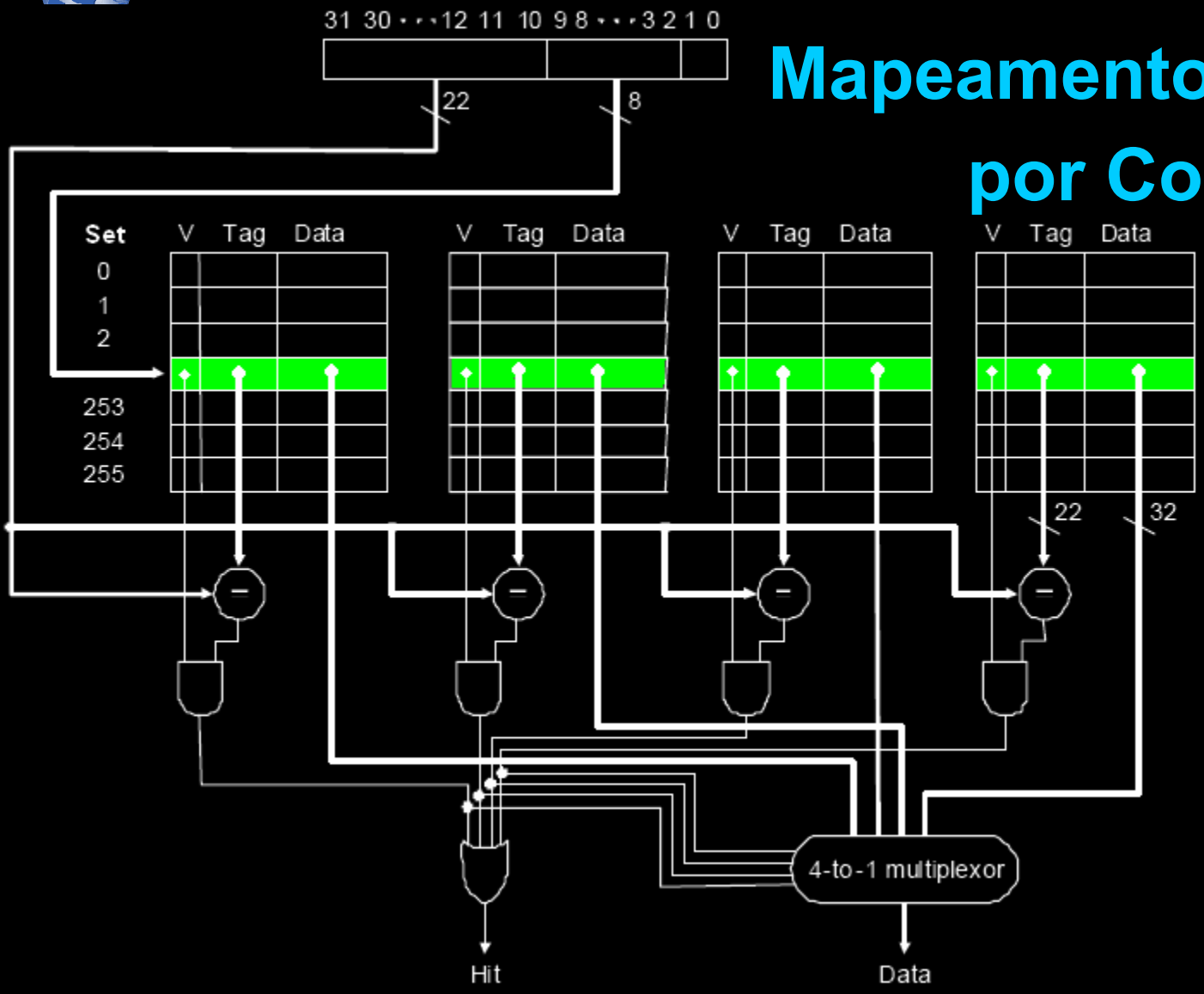
Memória Cache

Mapeamento Associativo



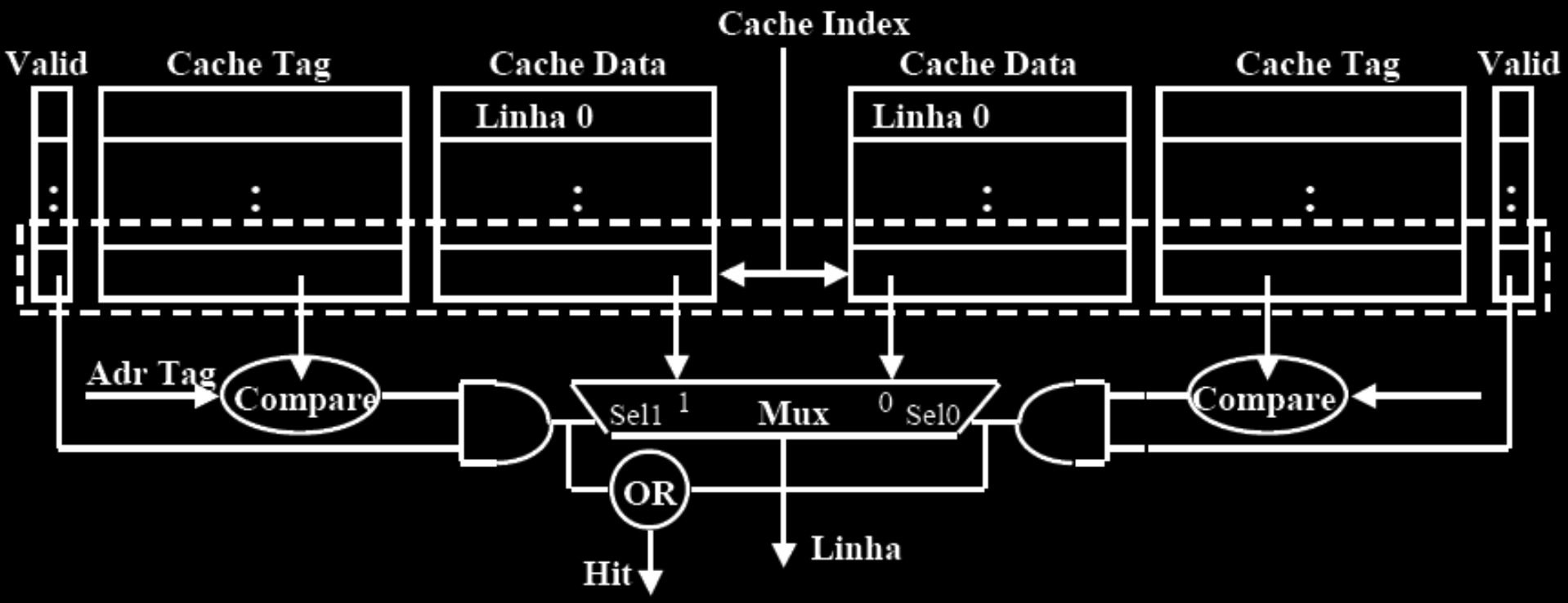
Memória Cache

Mapeamento Associativo por Conjuntos



Memória Cache

Mapeamento Associativo por Conjuntos



Memória Virtual

Mapeamento Direto

Conclusões:

1) Método simples e de baixo custo de implementação (um multiplexador e um comparador);

2) Blocos que disputam o mesmo quadro/linha não podem estar na cache concomitantemente;

3) Overhead baixo:

Overhead = Nº Bits política / Nº Bits Dados