Universidade Federal da Fronteira Sul
Curso de Ciência da Computação
Campus Chapecó

GEX 612 – Organização de Computadores

Subsistema de Memória

Prof. Luciano L. Caimi
lcaimi@uffs.edu.br

Roteiro

1. Hierarquia de Memória
   - Introdução; Operações; Características e métricas
2. Memória Principal
   - Tecnologia, funcionamento; evolução
3. Memória Cache
   - Localidade; Políticas de mapeamento, substituição e escrita;
4. Memória Virtual
   - Conceito; Funcionamento; Implementações

Introdução

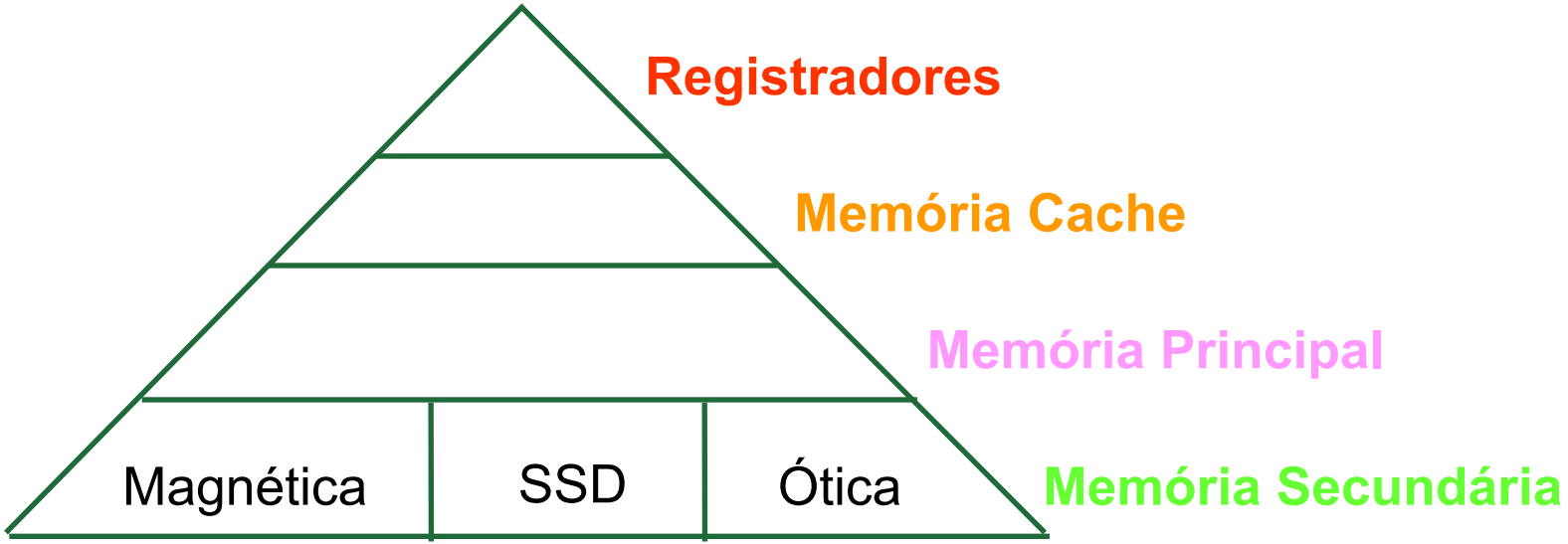
- A memória é responsável pelo armazenamento dos programas e informações a serem manipulados
- Realiza duas operações básicas: leitura e escrita de dados.
- Para escrever ou ler a informação precisamos informar o lugar preciso onde a informação será ou está armazenada: o endereço
- Assim cada informação 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

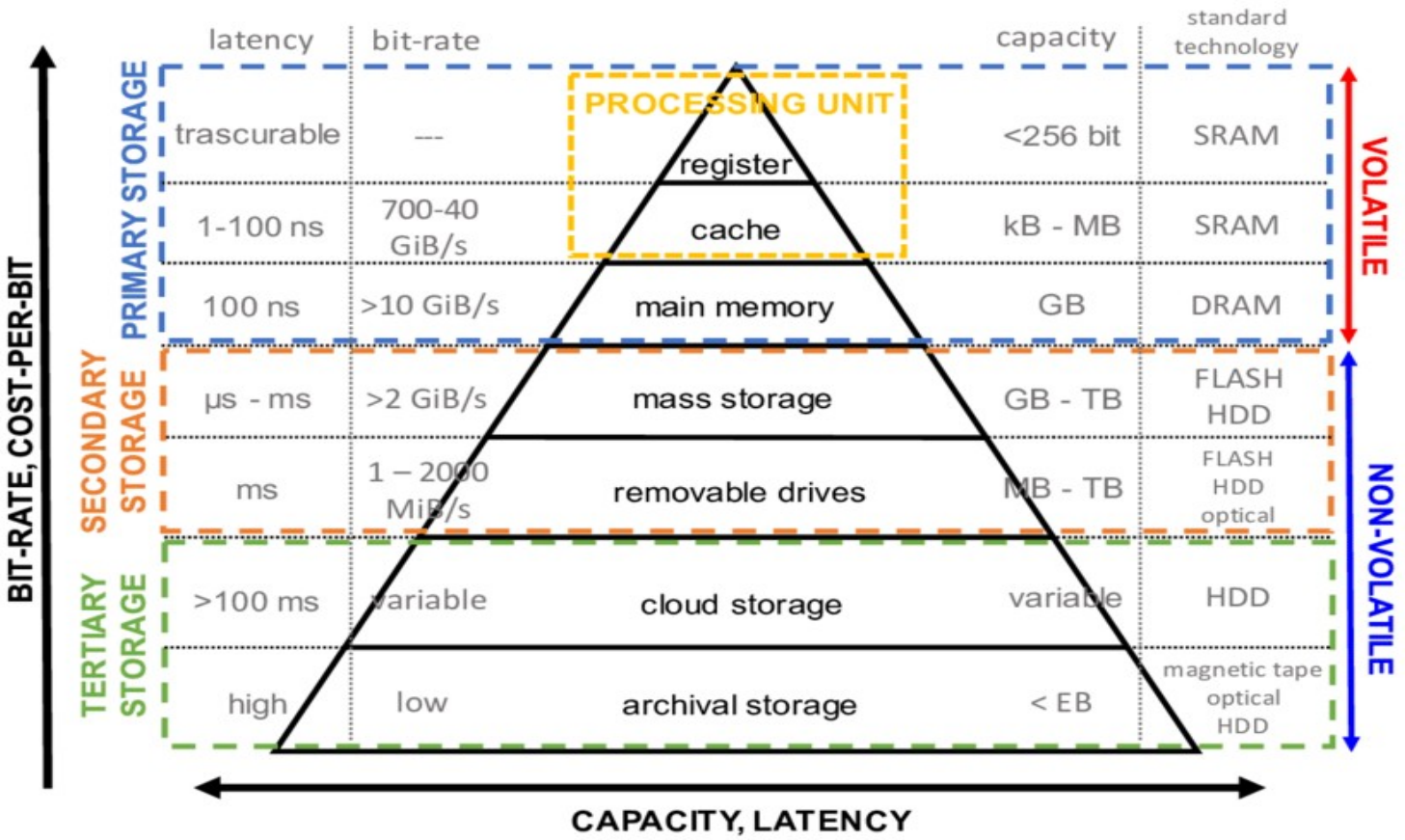
Hierarquia de Memória



Características:

- Tempo de acesso
- Tempo de ciclo
- Capacidade
- Custo
- Frequência
- Temporariedade
- Tecnologia de Fabricação
- Volatilidade
- Latência

Hierarquia de Memória



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
- Frequência: frequência 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

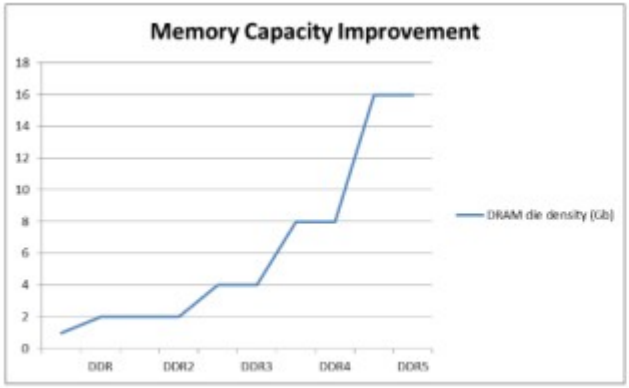
Hierarquia de Memória

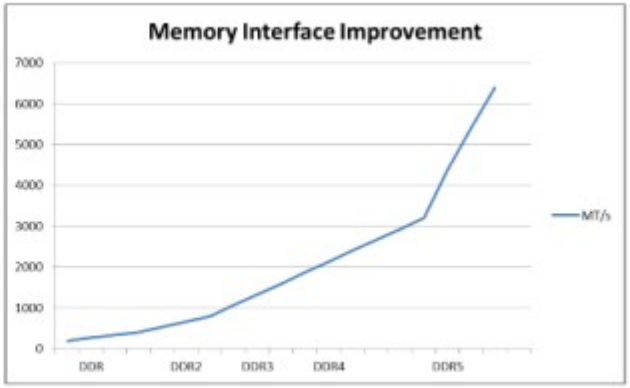
- Latência: tempo decorrido para finalizar uma operação

Nível da Memória	Latência de Acesso	Analogia 1	Analogia 2
Registradores	1 ciclo (~0,3 ns)	1 segundo	Seu cérebro
Cache L1	3-4 ciclos (~1,2 ns)	3 segundos	Uma sala
Cache L2	12-14 ciclos (~2,8 ns)	9 segundos	Um andar do andar
Cache L3	12.9 ns	43 segundos	Um prédio
Memória Principal	120 ns	6 minutos	O Campus
SSD	50-150 µs	2-6 dias	
HDD	1-10 ms	1-12 meses	
MP servidor remoto	~100 ms	1 século	
Armazenamento Ótico	segundos	milênios	

Hierarquia de Memória

Evolução



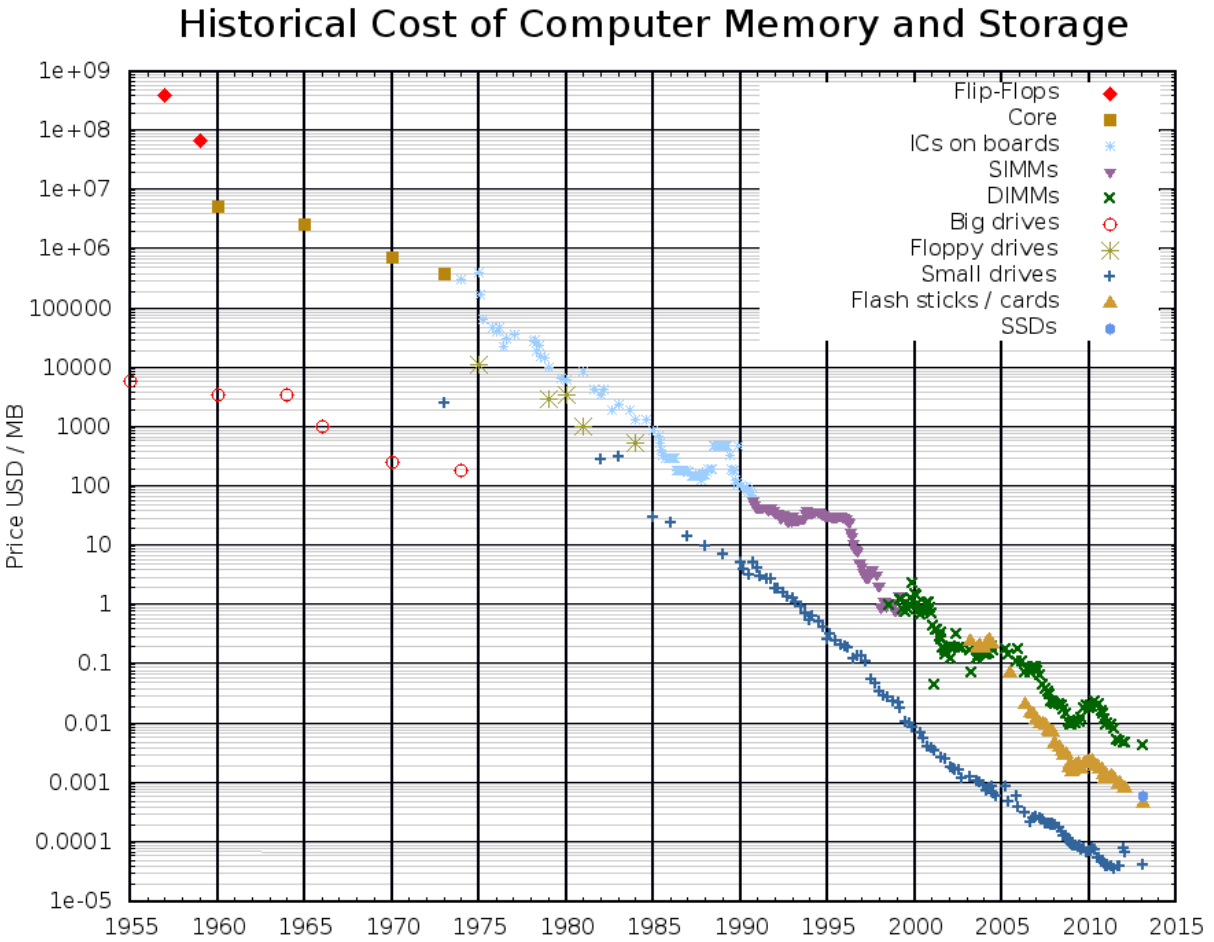


Metric	DRAM				HDD				SATAFlash SSD	
	1987	1997	2007	2018	1987	1997	2007	2018	2007	2018
Unit price($)	5k	15k	48	80	30k	2k	80	49	1k	415
Unit capacity	1MB	1GB	1GB	16GB	180MB	9GB	250GB	2TB	32GB	800GB
$/MB	5k	14.6	0.05	0.005	83.33	0.22	0.0003	0.00002	0.03	0.0005
Random IOPS	–	–	–	–	5	64	83	200	6.2k	67k (r)/20k (w)
Sequential b/w (MB/s)	–	–	–	–	1	10	300	200	66	500 (r)/460 (w)

https://cacm.acm.org/magazines/2019/11/240388-the-five-minute-rule-30-years-later-and-its-impact-on-the-storage-hierarchy/fulltext

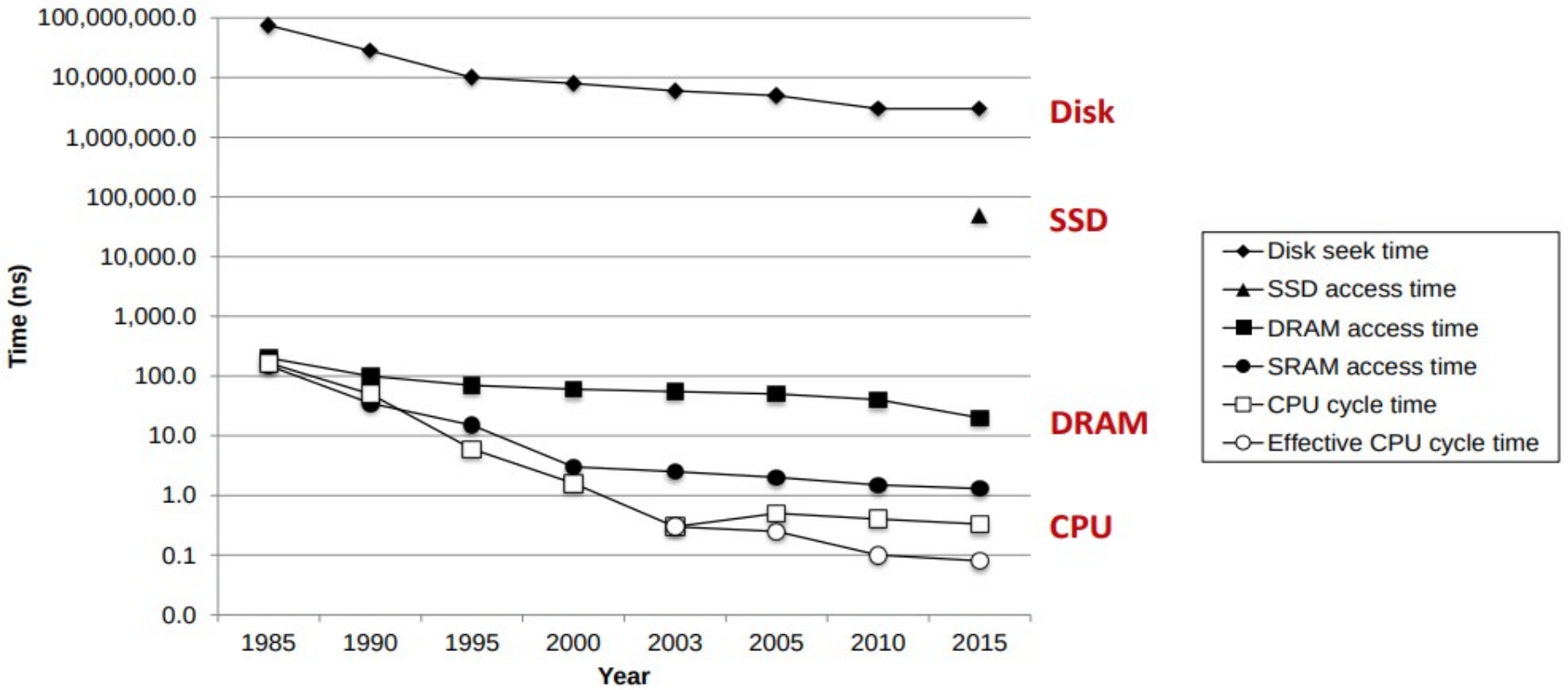
Hierarquia de Memória

Evolução



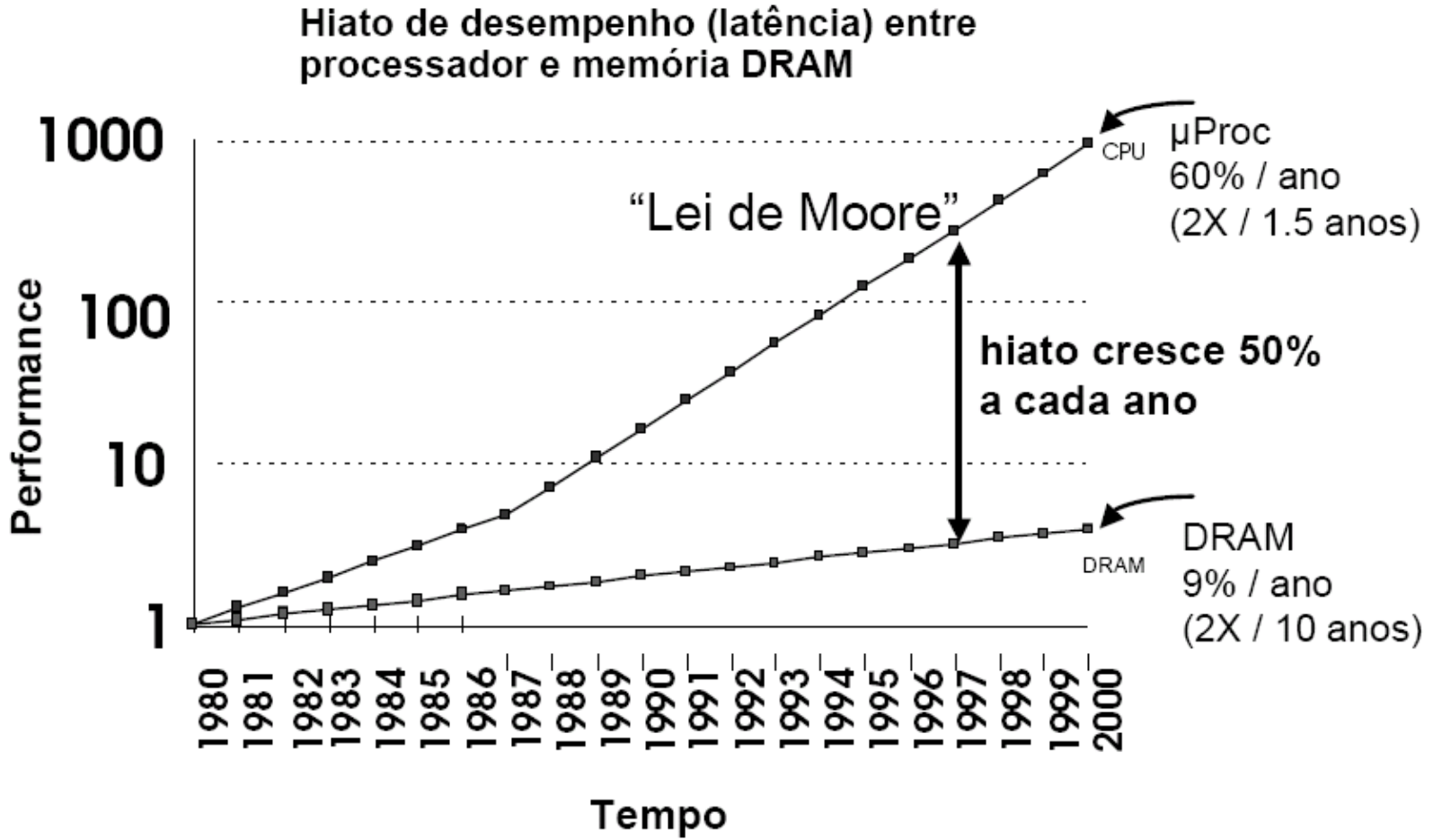
Hierarquia de Memória

Evolução



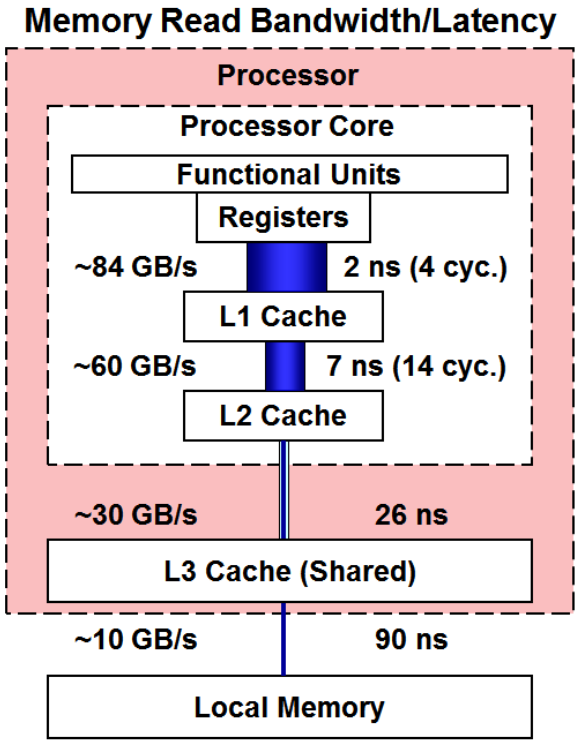
Hierarquia de Memória

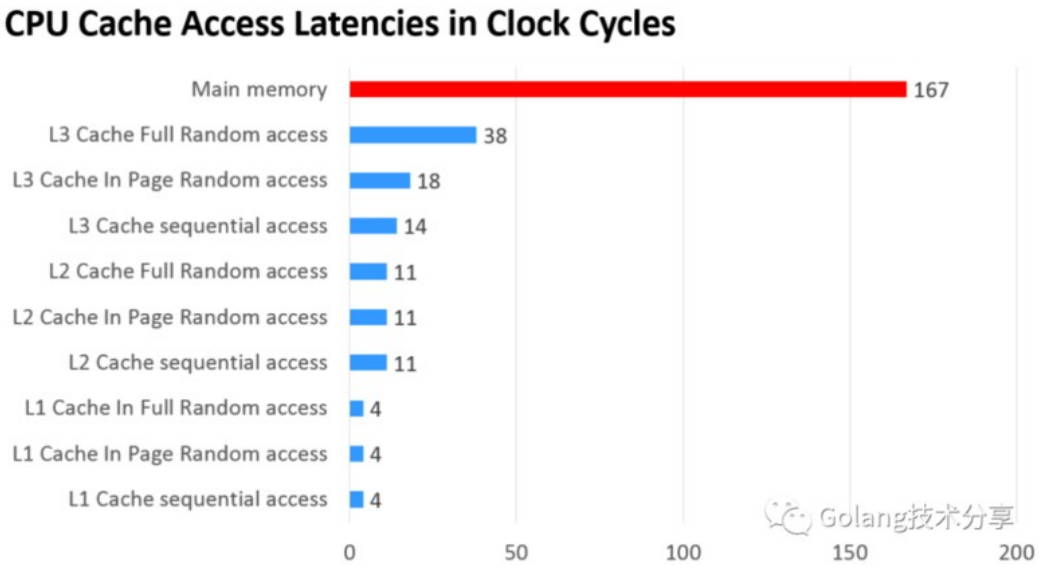
Evolução



Hierarquia de Memória

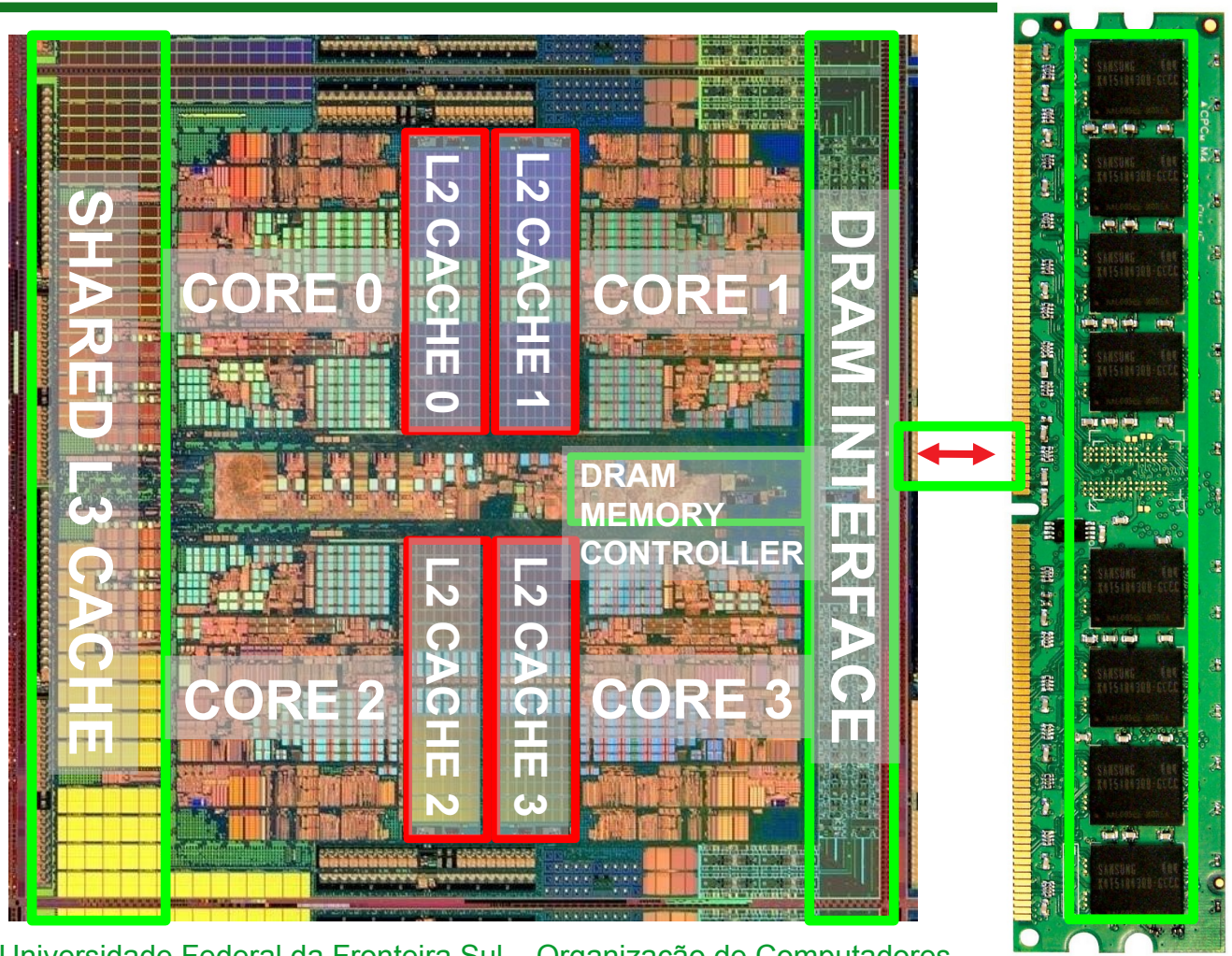
Evolução



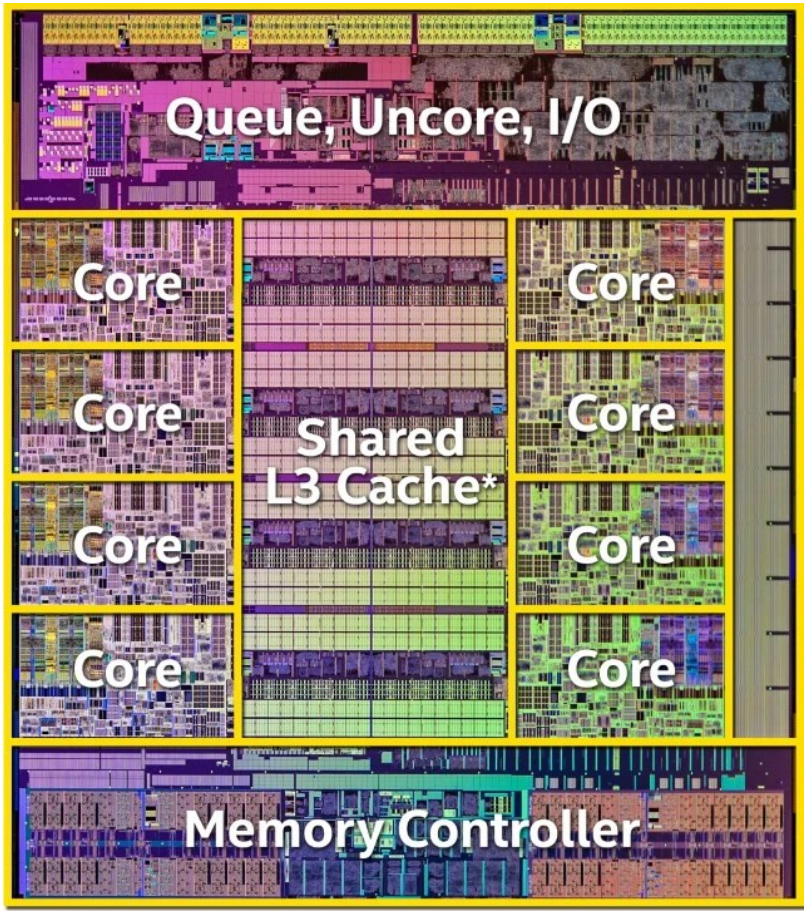


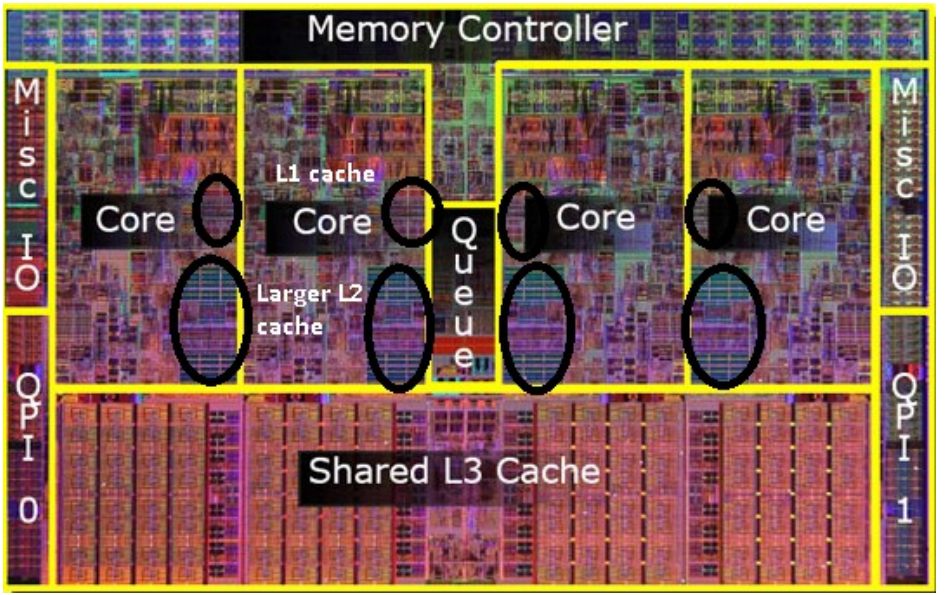
Hierarquia de Memória

Processador Intel I5



Hierarquia de Memória





Hierarquia de Memória

32KB L1 Instruction/core

32KB L1 Dada/core

3 ciclos de latência

256KB Unificada L2/core

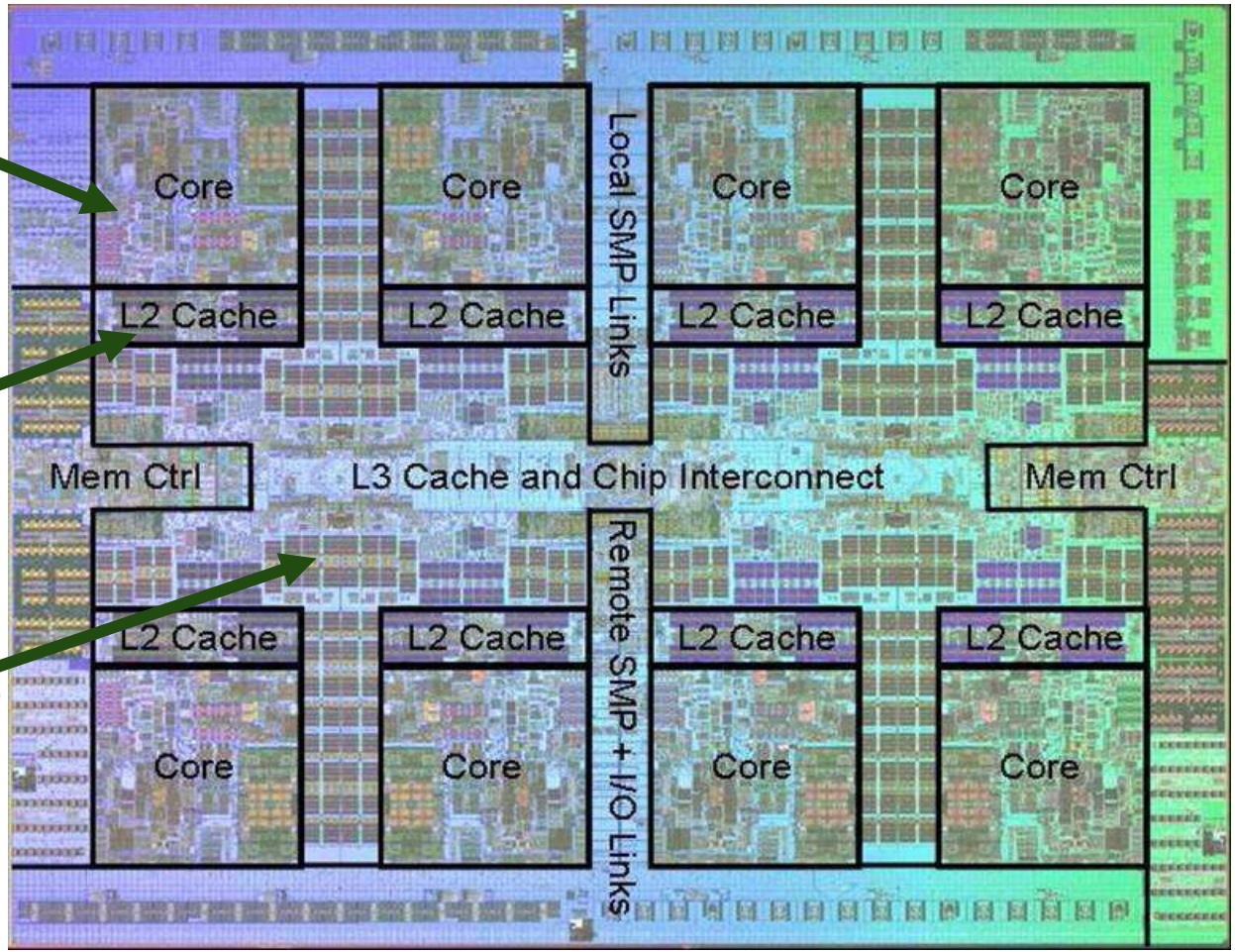
8 ciclos de latência

Processador IBM Power 7

32MB Compartilhamento Unificado L3

Embedded DRAM (eDRAM)

25 ciclos de latência



Hierarquia de Memória

Como a hierarquia é gerenciada?

- **registradores ↔ memória**

  → pelo compilador

- **cache ↔ memória principal**

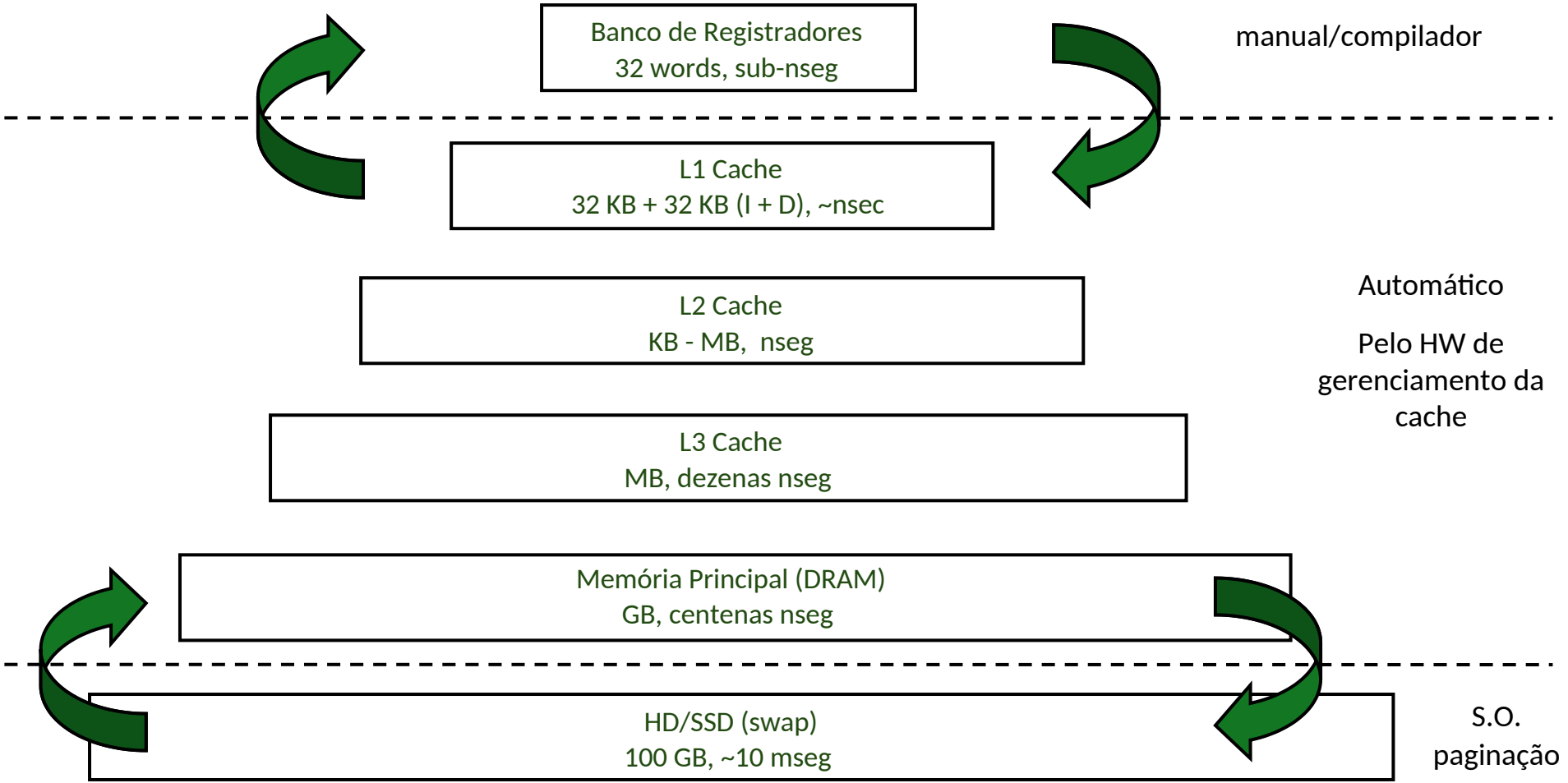
  → pelo hardware

- **memória principal ↔ HD/SSD**

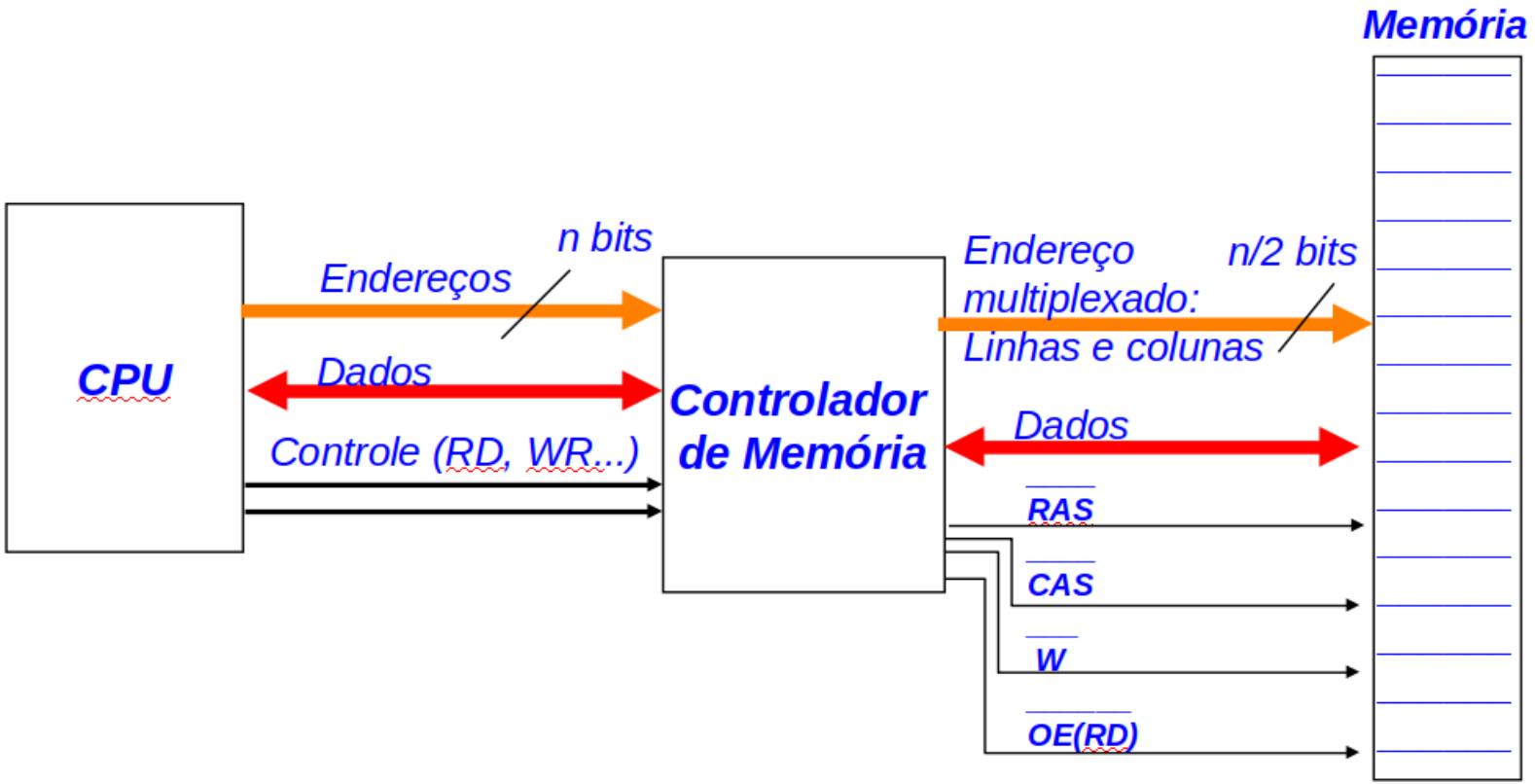
  → pelo hardware e pelo sistema operacional (memória virtual)

  → pelo programador e pelo usuário (arquivos)

Hierarquia de Memória



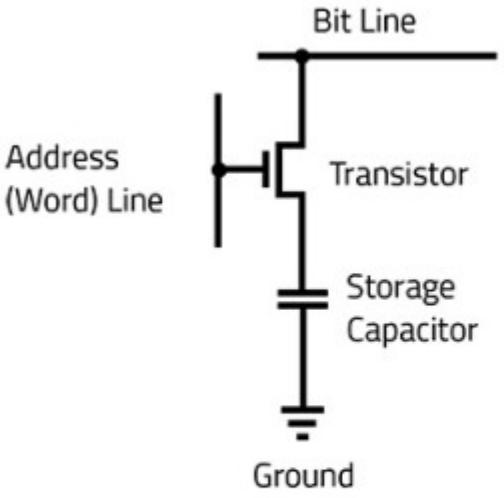
Memória Principal



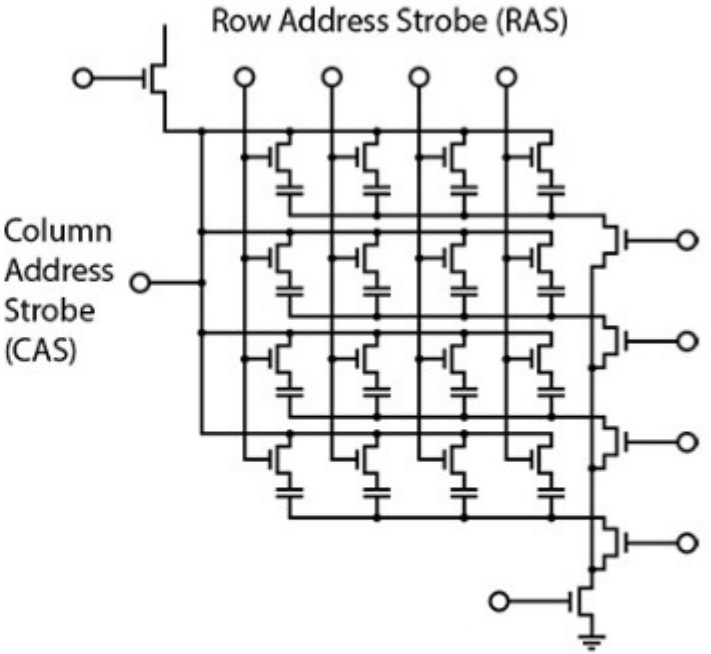
Memória Principal

- É a memória básica do sistema a partir de onde os programas são executados
- 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

Memória Principal

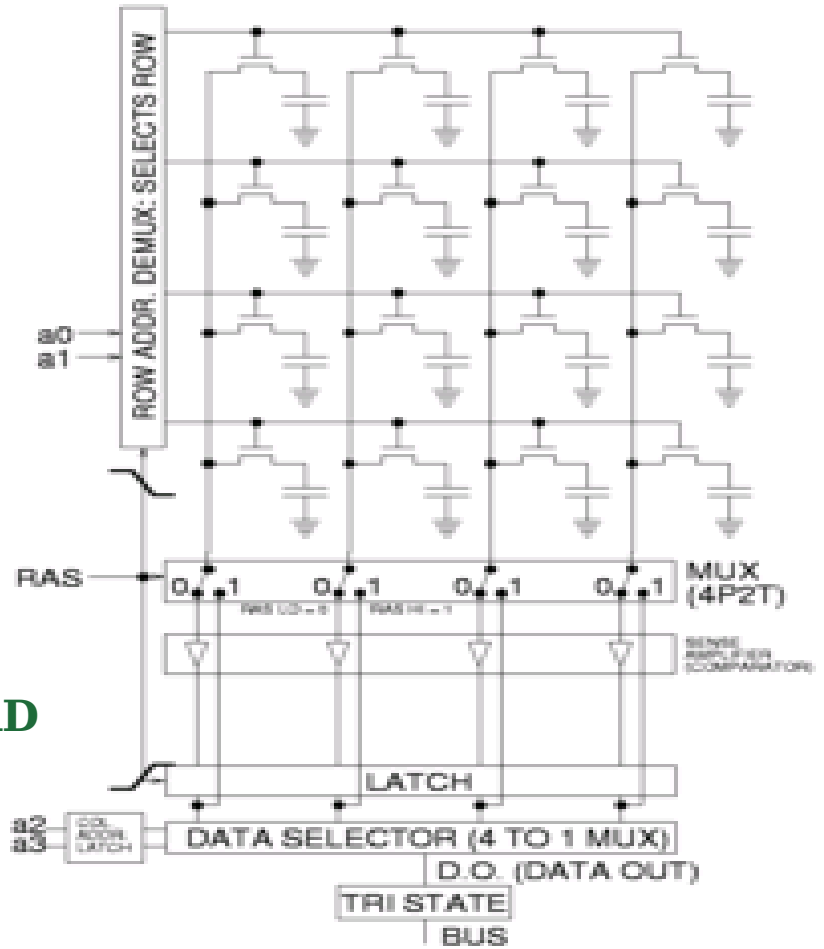


Single Memory Cell

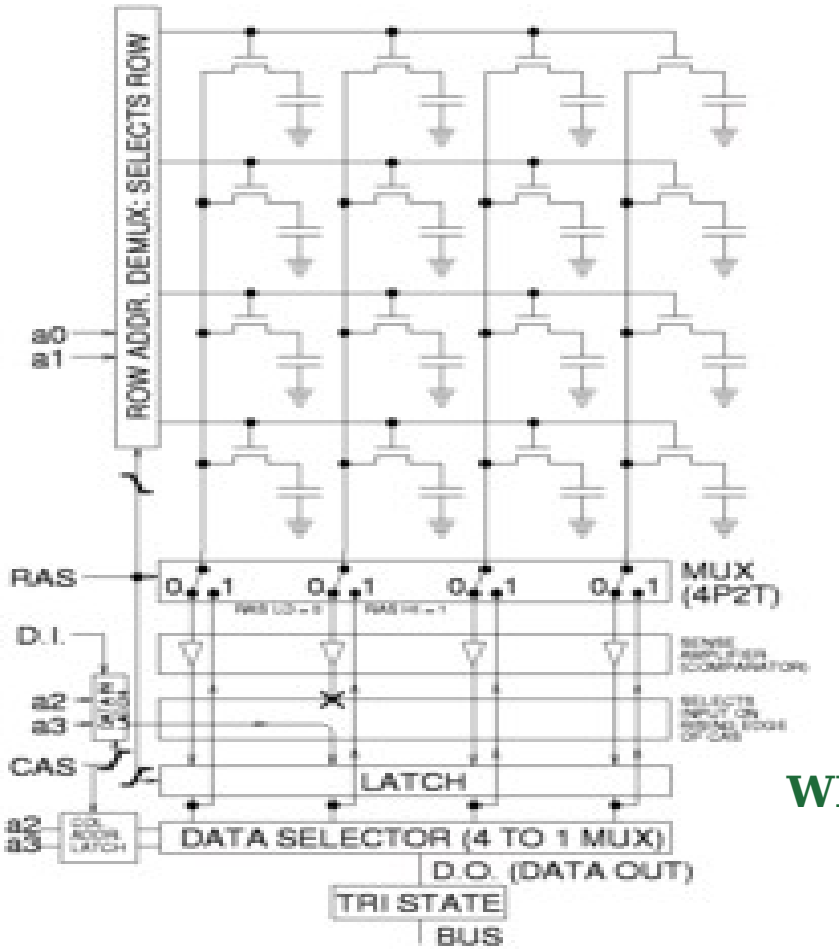


Memory Cell Array

Memória Principal

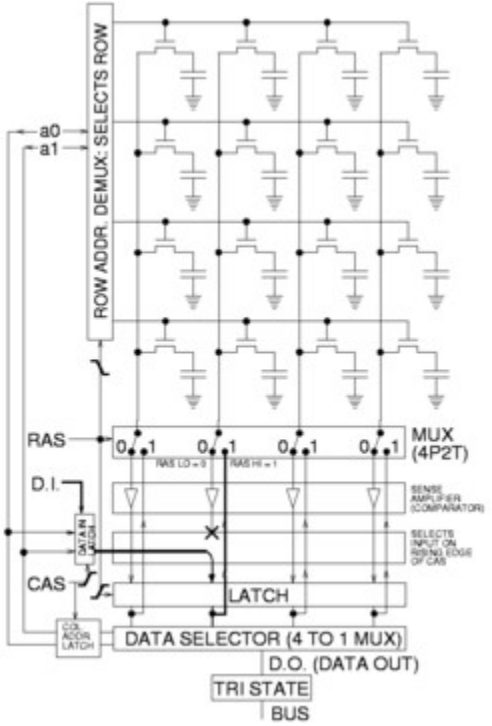


READ

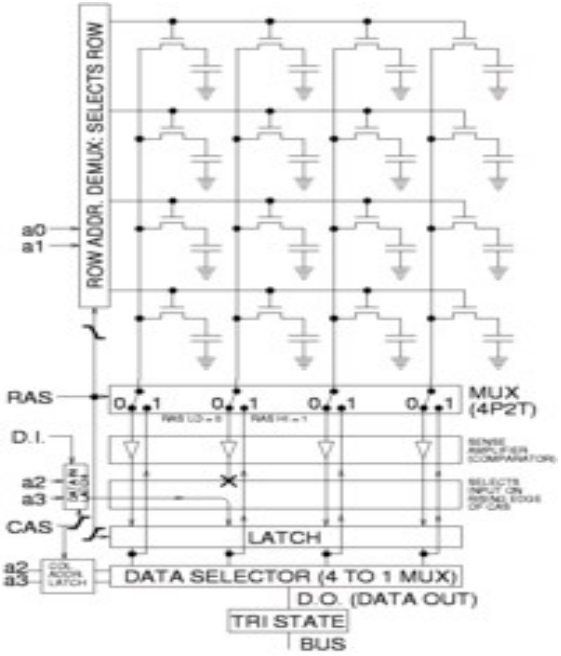


WRITE

Memória Principal

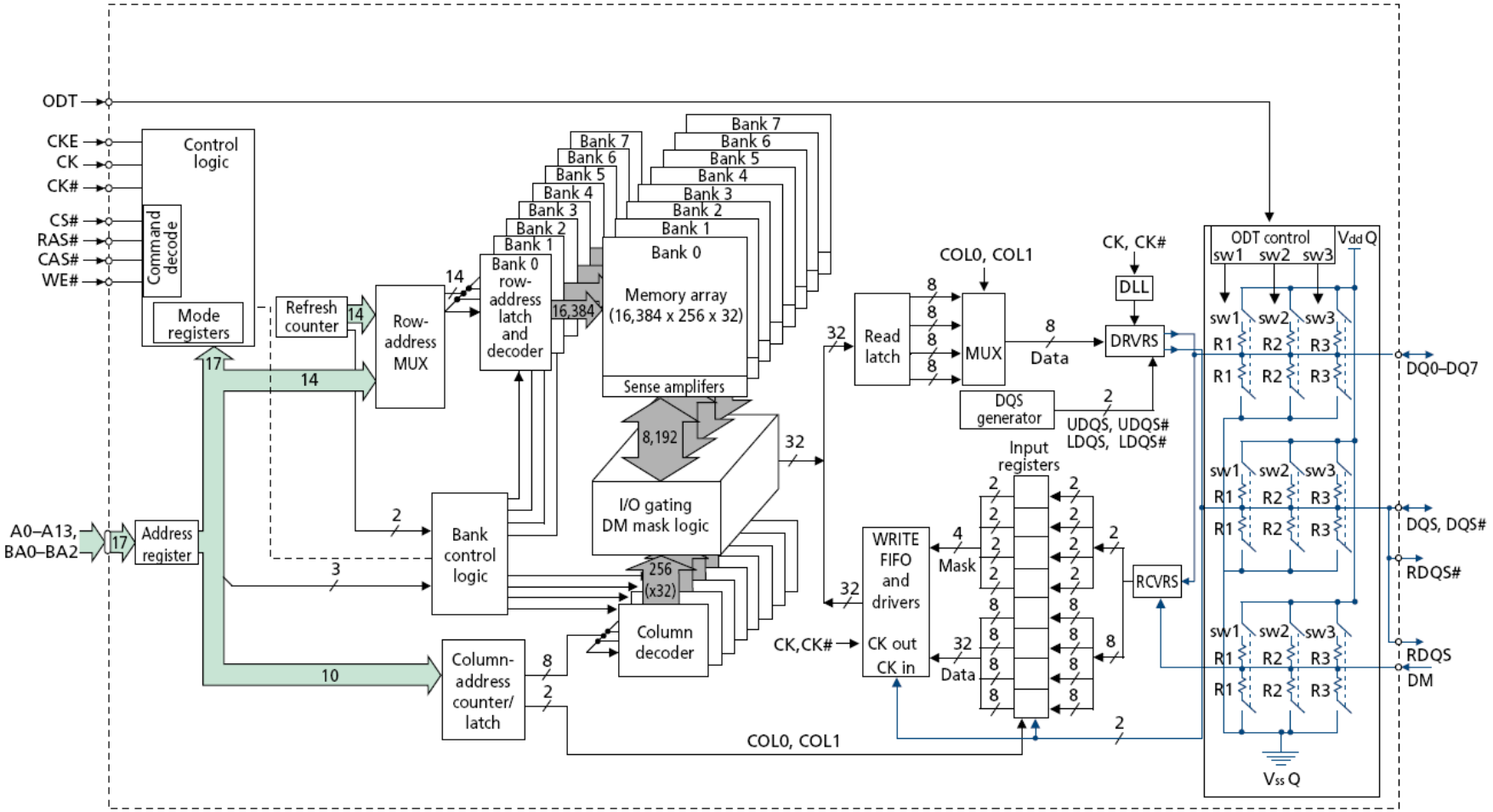


READ



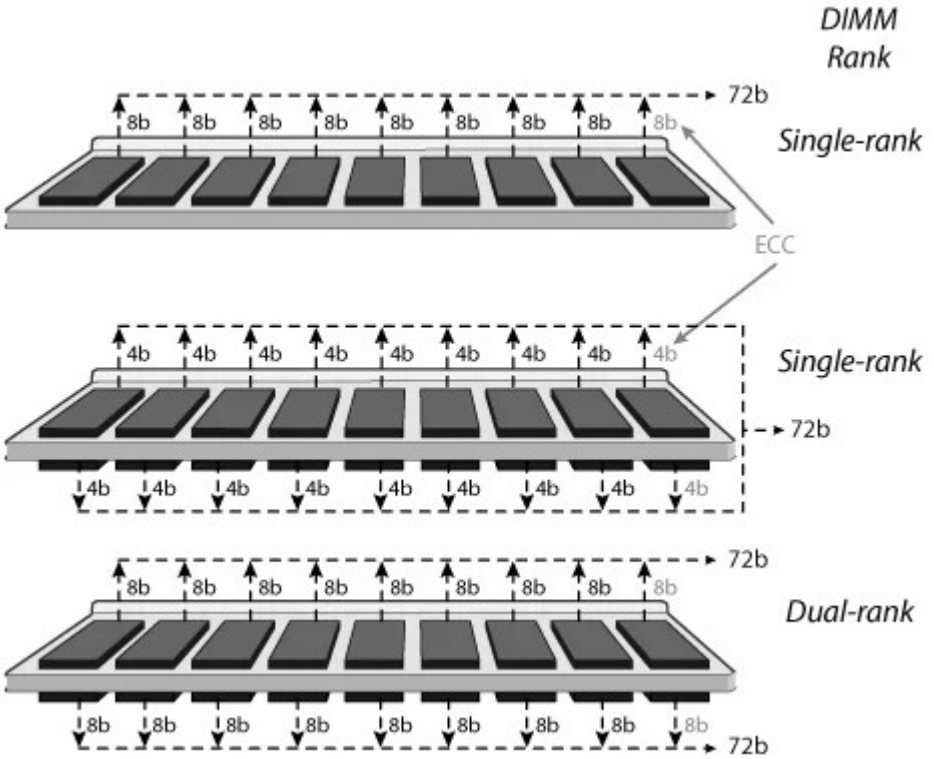
WRITE

Memória Principal



Rank da memória

- Diz respeito ao arranjo da quantidade de chips e a respectiva quantidade de bits de dados fornecidos por cada chip durante uma operação (leitura ou escrita) na memória principal



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 clock do barramento, chamados neste caso de "wait states": 8-6-6-6

  ou seja, o primeiro acesso precisou de 8 ciclos de clock espera até o dado ser entregue e os demais acessos 6 ciclos de clock de espera cada um

Memória Principal

Largura de Banda Máxima Teórica ou

Taxa de Transferência Máxima Teórica

LBMT = Frequência do Barramento * Largura do Barramento

Exemplo:

Frequência de Barramento = 33 MHz

Largura do Barramento = 32 bits

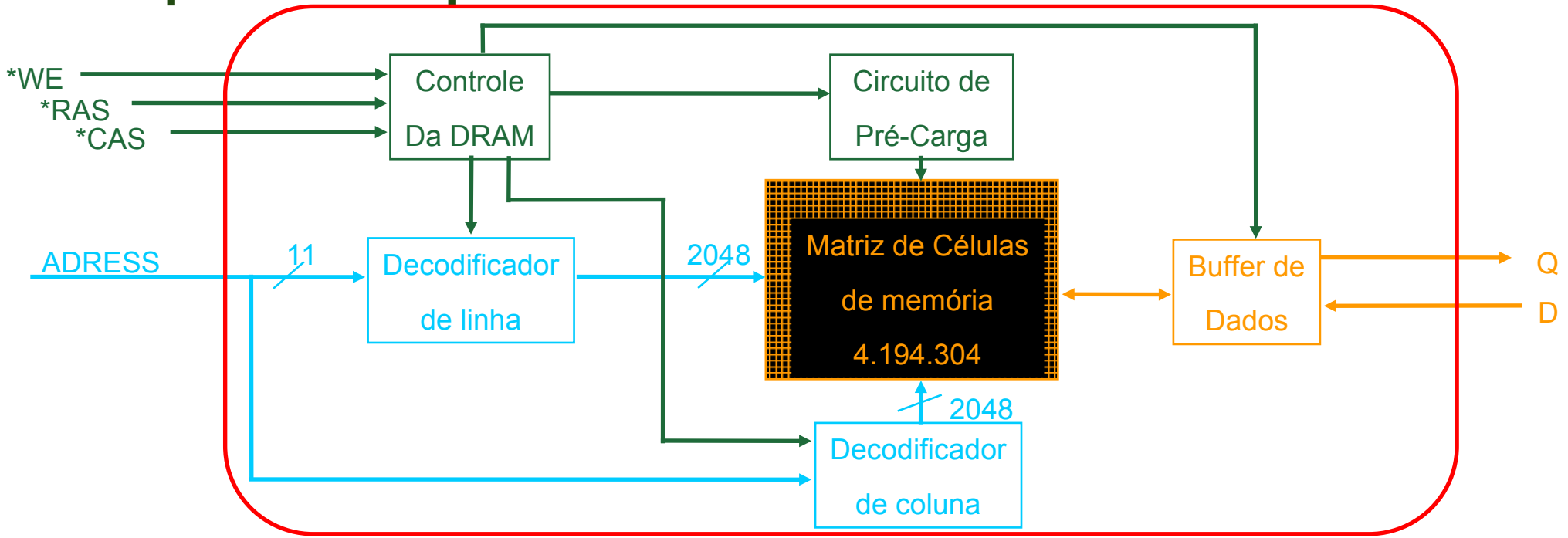
LBMT = $33x10^6$ ciclos/segundo * 32 bits/ciclo

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

LBMT = 1056 Mbits/segundo = 132 MBytes/segundo

Memória Principal

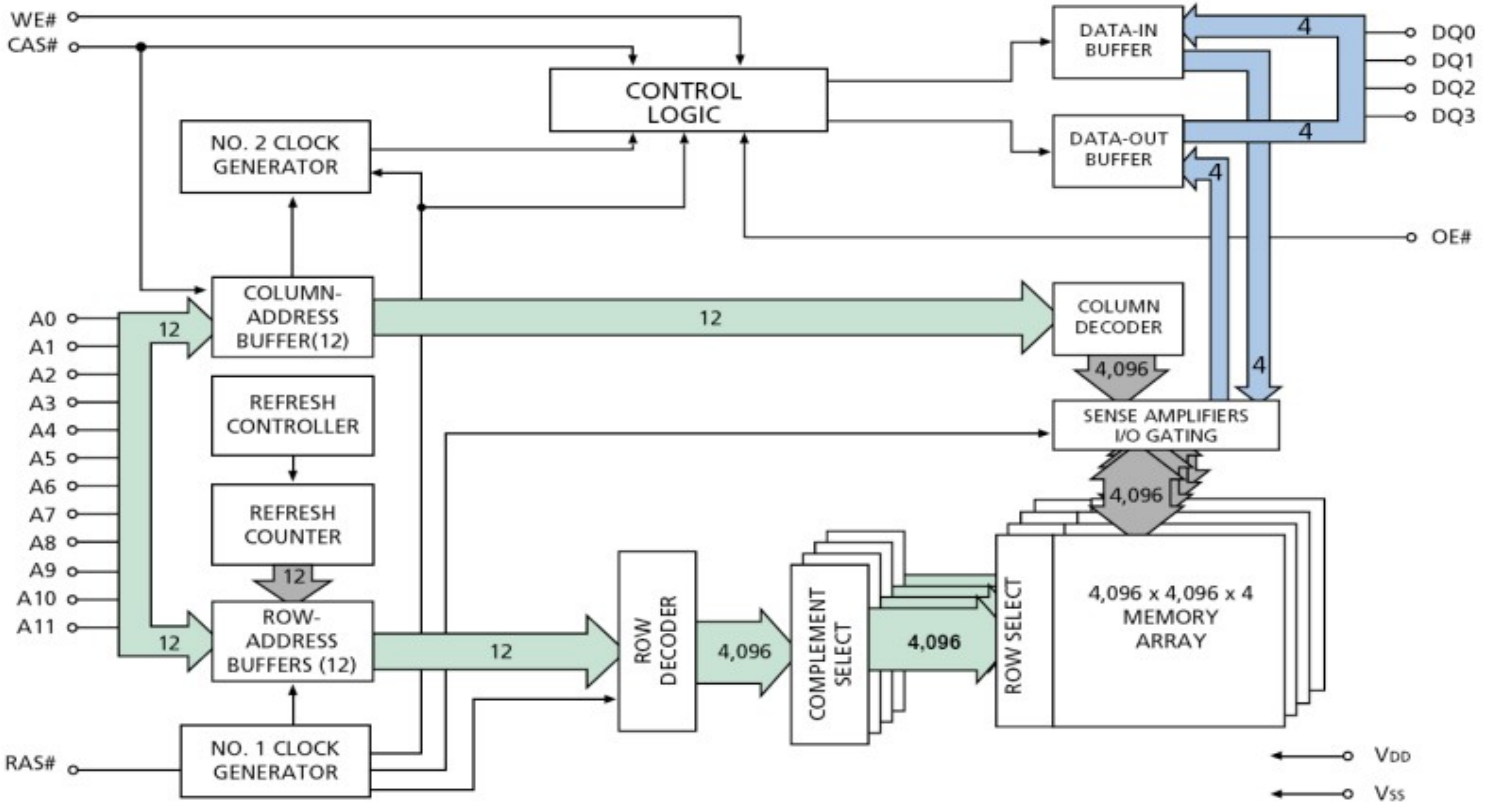
- **Esquema simplificado de uma DRAM**



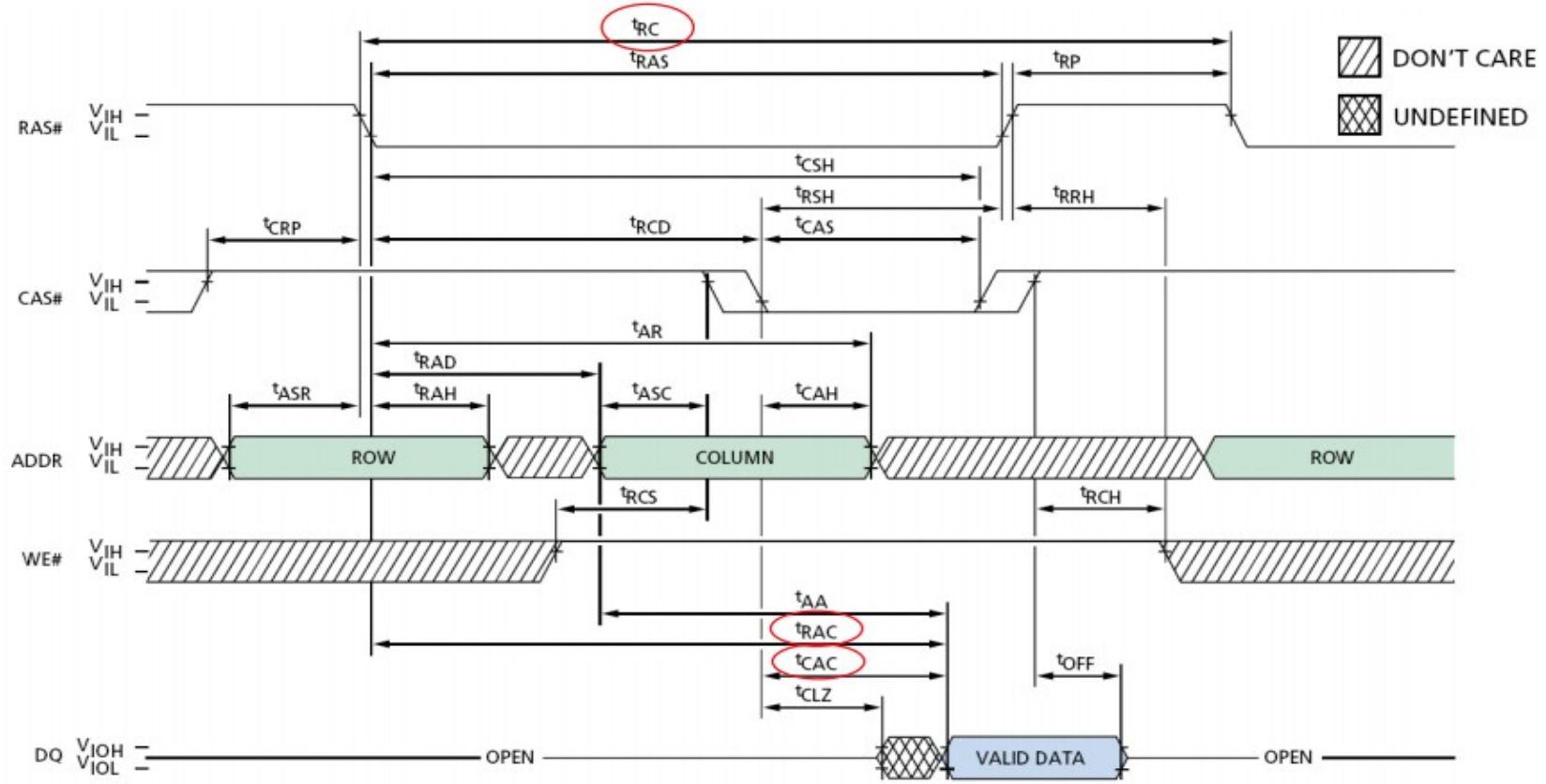
1 bit por acesso; 4Mbits de dados armazenados; 22 bits de endereço; endereço multiplexado; RAS – Row Address Strobe; CAS – Column Address Strobe; WE – Write Enable;

Memória Principal

- Micron MT4LC16M4T8 (16M x 4bit)



Memória Principal



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

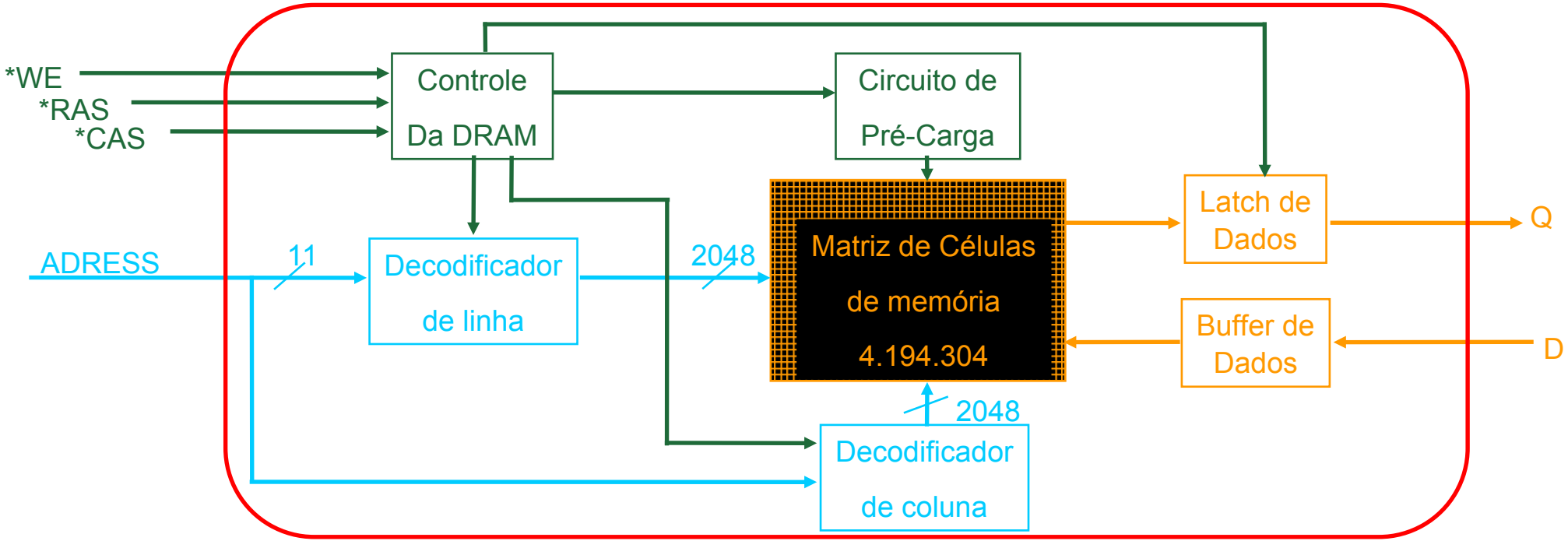
Assim a LB_efetiva é de 4/11 da LBMT ou seja:

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

LB_efetiva = 192 MBytes/seg

Memória Principal

- **EDO-RAM**



Burst Extended Data Output – BEDO-RAM

- Implementa um contador que gera os endereços subsequentes 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

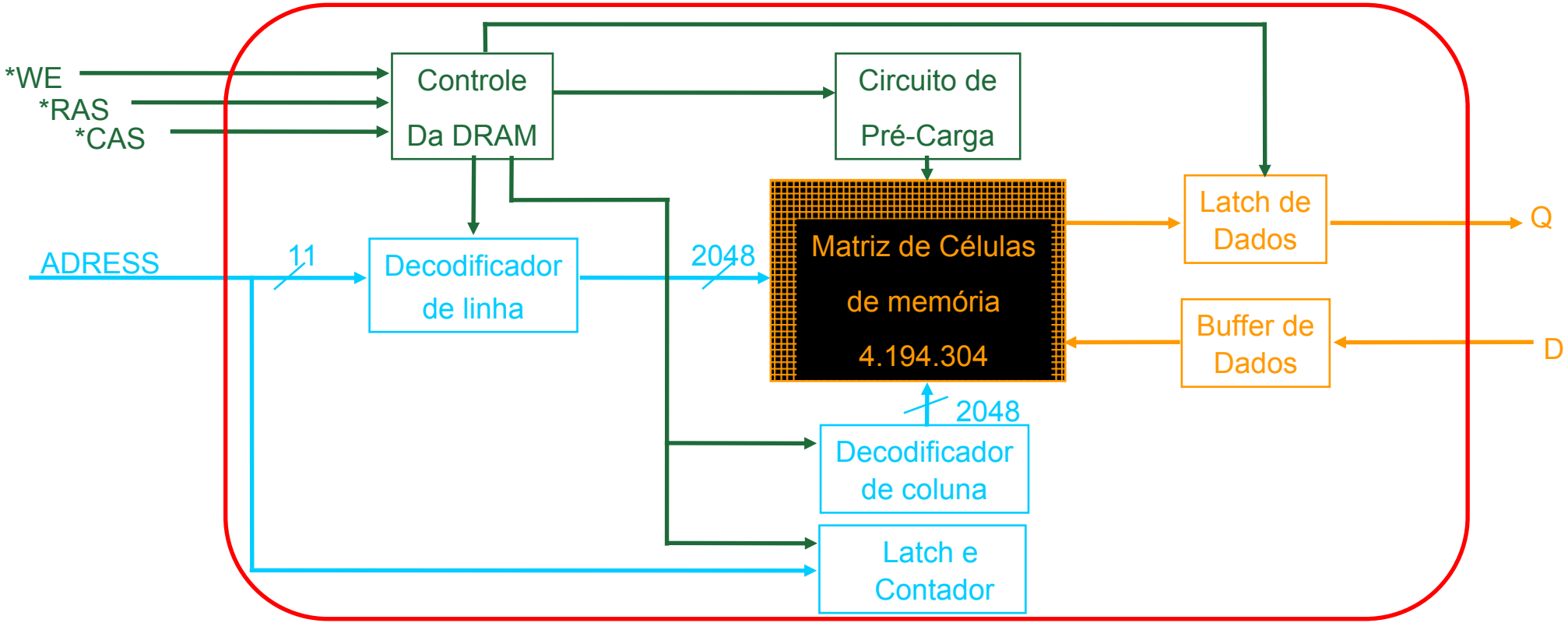
Assim a LB_efetiva é de 4/8 da LBMT ou seja:

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

LB_efetiva = 400 MBytes/seg

Memória Principal

- **BEDO-RAM**



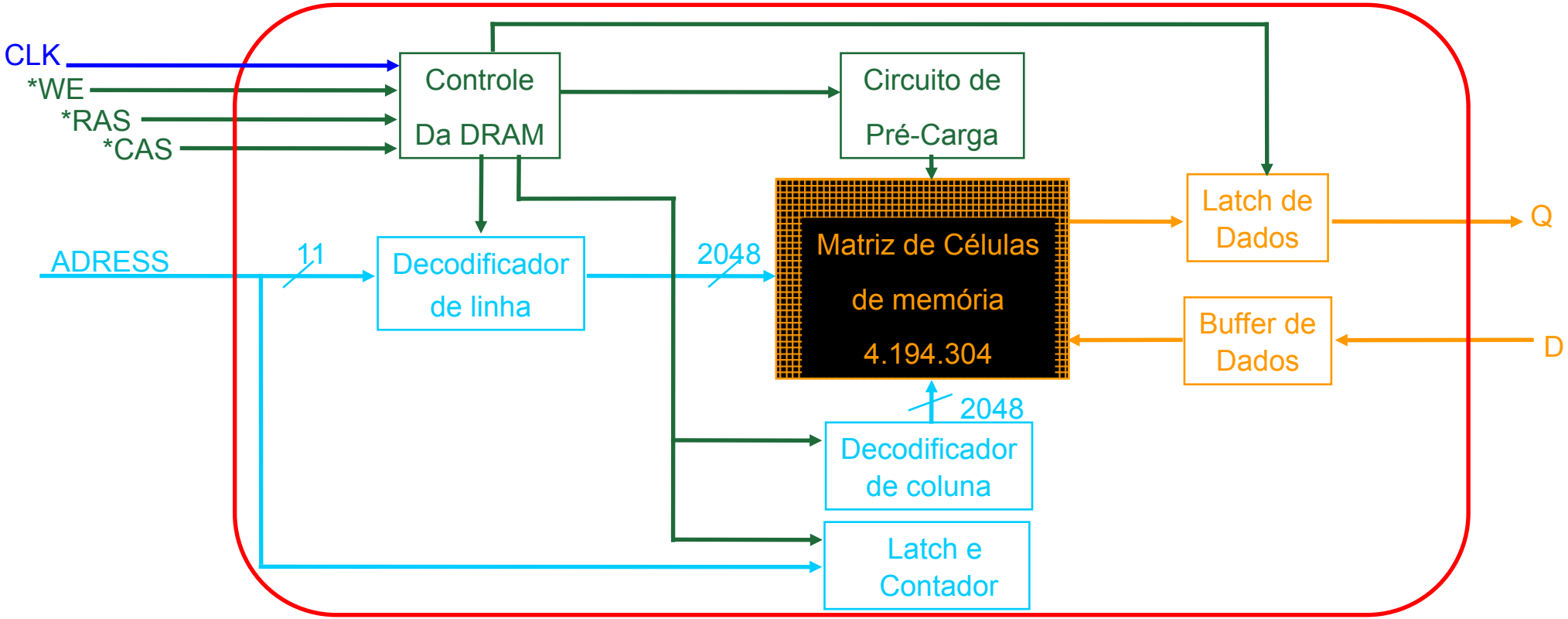
Memória Principal

Syncronous Dinamic RAM – SDRAM

- Um sinal de clock sincroniza a memória e seu controlador. Não são necessários sinais de handshake subsequentes pois o clock sincroniza as operações
- "wait-states": 5-1-1-1
- Vantagens: opera em frequências 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 quanto a de descida é utilizada

  Assim na mesma frequência de clock a taxa de transferência é o dobro
- A especificação mostra a frequencia aparente e não a real, ou seja, uma DDR 400 tem frequencia real de 200MHz
- Frequências de 133, 166 e 200MHz
- Alimentação é de 2,5 Volts e 184 pinos
- Terminação resistiva na placa-mãe

Memória Principal

Double Data Rate 2 SDRAM – DDR2 SDRAM

- Mesmo princípio de funcionamento da DDR
- Construídas nas frequências 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
- Frequências 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

Double Data Rate 4 SDRAM – DDR4 SDRAM

- Frequências até 1600 MHz
- Buffer interno com o mesmo tamanho das DDR3
- Alimentação é de 1,2 Volts
- 288 pinos (contra 240 na DDR3)
- Wait-States: 20-15-15-15 → DDR4-2666
  24-15-15-15 →DDR4-3200

Memória Principal

Double Data Rate 5 SDRAM – DDR5 SDRAM

- Permite dobrar a largura de banda em comparação a DDR4
- Alimentação é de 1,1 Volts
- Latências similares a DDR4
- Canais A e B integrados no próprio módulo

Memória Principal

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

Disponível nas tecnologias DDR SDRAM em diante

Integrado no próprio módulo nas DDR5

Name		Release year	Chip			Bus			Voltage (V)	Pins		
Gen	Standard		Clock rate (MHz)	Cycle time (ns)	Pre-fetch	Clock rate (MHz)	Transfer rate (MT/s)	Bandwidth (MB/s)		DIMM	SO-DIMM	Micro-DIMM
DDR	DDR-200	1998	100	10	2n	100	200	1600	2.5	184	200	172
	DDR-266		133	7.5		133	266	2133⅓				
	DDR-333		166⅔	6		166⅔	333	2666⅔				
	DDR-400		200	5		200	400	3200	2.6			
DDR2	DDR2-400	2003	100	10	4n	200	400	3200	1.8	240	200	214
	DDR2-533		133⅓	7.5		266⅔	533⅓	4266⅔				
	DDR2-667		166⅔	6		333⅓	666⅔	5333⅓				
	DDR2-800		200	5		400	800	6400				
	DDR2-1066		266⅔	3.75		533⅓	1066⅔	8533⅓				
DDR3	DDR3-800	2007	100	10	8n	400	800	6400	1.5/1.35	240	204	214
	DDR3-1066		133⅓	7.5		533⅓	1066⅔	8533⅓				
	DDR3-1333		166⅔	6		666⅔	1333⅓	10666⅔				
	DDR3-1600		200	5		800	1600	12800				
	DDR3-1866		233⅓	4.29		933⅓	1866⅔	14933⅓				
	DDR3-2133		266⅔	3.75		1066⅔	2133⅓	17066⅔				
DDR4	DDR4-1600	2014	200	5	8n	800	1600	12800	1.2/1.05	288	260	-
	DDR4-1866		233⅓	4.29		933⅓	1866⅔	14933⅓				
	DDR4-2133		266⅔	3.75		1066⅔	2133⅓	17066⅔				
	DDR4-2400		300	3⅓		1200	2400	19200				
	DDR4-2666		333⅓	3		1333⅓	2666⅔	21333⅓				
	DDR4-2933		366⅔	2.73		1466⅔	2933⅓	23466⅔				
	DDR4-3200		400	2.5		1600	3200	25600				
DDR5	DDR5-3200	2020	200	5	16n	1600	3200	25600	1.1	288		
	DDR5-3600		225	4.44		1800	3600	28800				
	DDR5-4000		250	4		2000	4000	32000				
	DDR5-4800		300	3⅓		2400	4800	38400				
	DDR5-5000		312½	3.2		2500	5000	40000				
	DDR5-5120		320	3⅛		2560	5120	40960				
	DDR5-5333		333⅓	3		2666⅔	5333⅓	42666⅔				
	DDR5-5600		350	2.86		2800	5600	44800				
	DDR5-6400		400	2.5		3200	6400	51200				
	DDR5-7200		450	2.22		3600	7200	57600				

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...

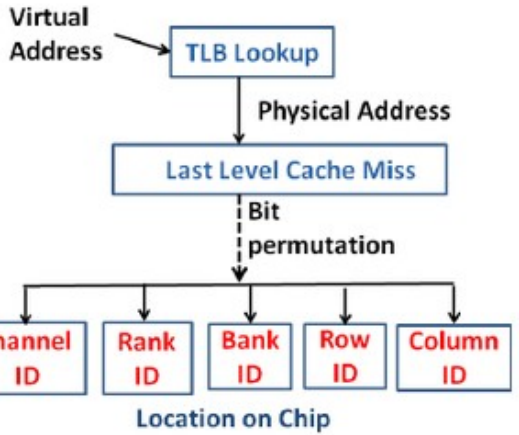
- **RAS Precharge (tRP)**

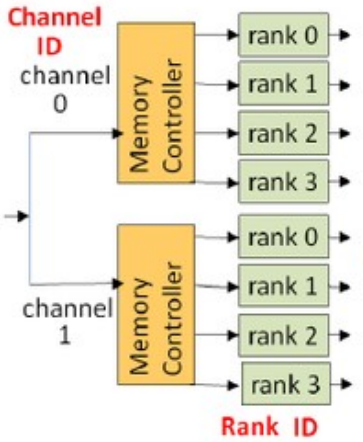
Após o dado ter sido entregue pela memória, um comando chamado Precharge precisa ser executado para desativar a linha da memória que estava sendo usada e para permitir que uma nova linha seja ativada tRP é o tempo entre o comando Precharge e o próximo comando “Active” (que inicia a leitura)

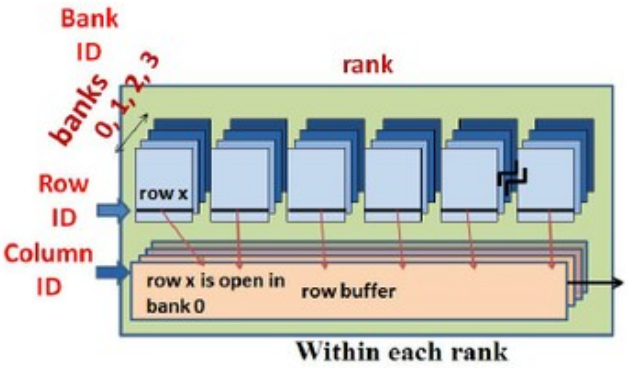
- **Command Rate (CMD)**

Tempo 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 respectivamente)

Memória Principal







Memory options part number decoder

Short Name

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
HPE				ggg	GB					s	R	x	ff			PC4	-	wwww	a	-	ppp	m		eeeee			Kit												

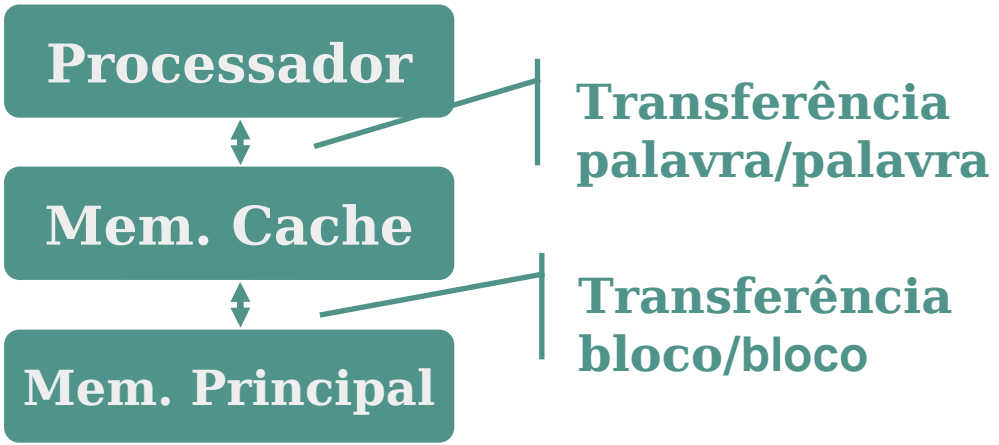
- HPE indicates it's HPE approved memory being sold through an HPE channel or direct from HPE.
- ggg indicates module capacity: 8 = 8GB; 16 = 16GB; 32 = 32GB; 64 = 64GB; 128 = 128GB
- s: number of Ranks: 1 = one rank; 2 = two ranks; 4 = four ranks; 8 = eight ranks
- R=Rank(s)
- ff = data bit width: 4 = 4 bit; 8 = 8 bit; 16 =16 bit
- PC4- is the memory type: PC4 = DDR4
- wwww is the module speed (MT/s): 2133 = 2133 MT/s; 2400 = 2400 MT/s; 2666 = 2666 MT/s; 2933 = 2933 MT/s; 3200 =3200 MT/s
- a is Speed Grade CAS latency (column address strobe): P = CAS 15-15-15; T = CAS 17-17-17; U = CAS 20-18-18; V = CAS 19-19-19; V = CAS 22-19-19
- ppp is DRAM package type: Blank = Non-3DS (Mono, SDP, DDP or QDP); 3DS = 3 Dimensional Stacked
- m is the module type: E = Unbuffered ECC (UDIMM); L = Load Reduced DIMM; R = Registered DIMM
- eeeee is a special descriptor: Smart = SmartMemory; STND = Standard Memory; Blank = SGI SKU

Example: HPE 128GB 8Rx4 PC4-2666V-3DSL Smart Kit

For example, HPE 128GB 8Rx4 PC4-2666V-2HTSVL Smart Kit indicates an HPE SmartMemory DIMM with a 128GB capacity, Octal rank, a data width of 4, memory type of DDR4, 2666 Load Reduced 22-19-19 latency, and an HPE kit.

Memória Principal

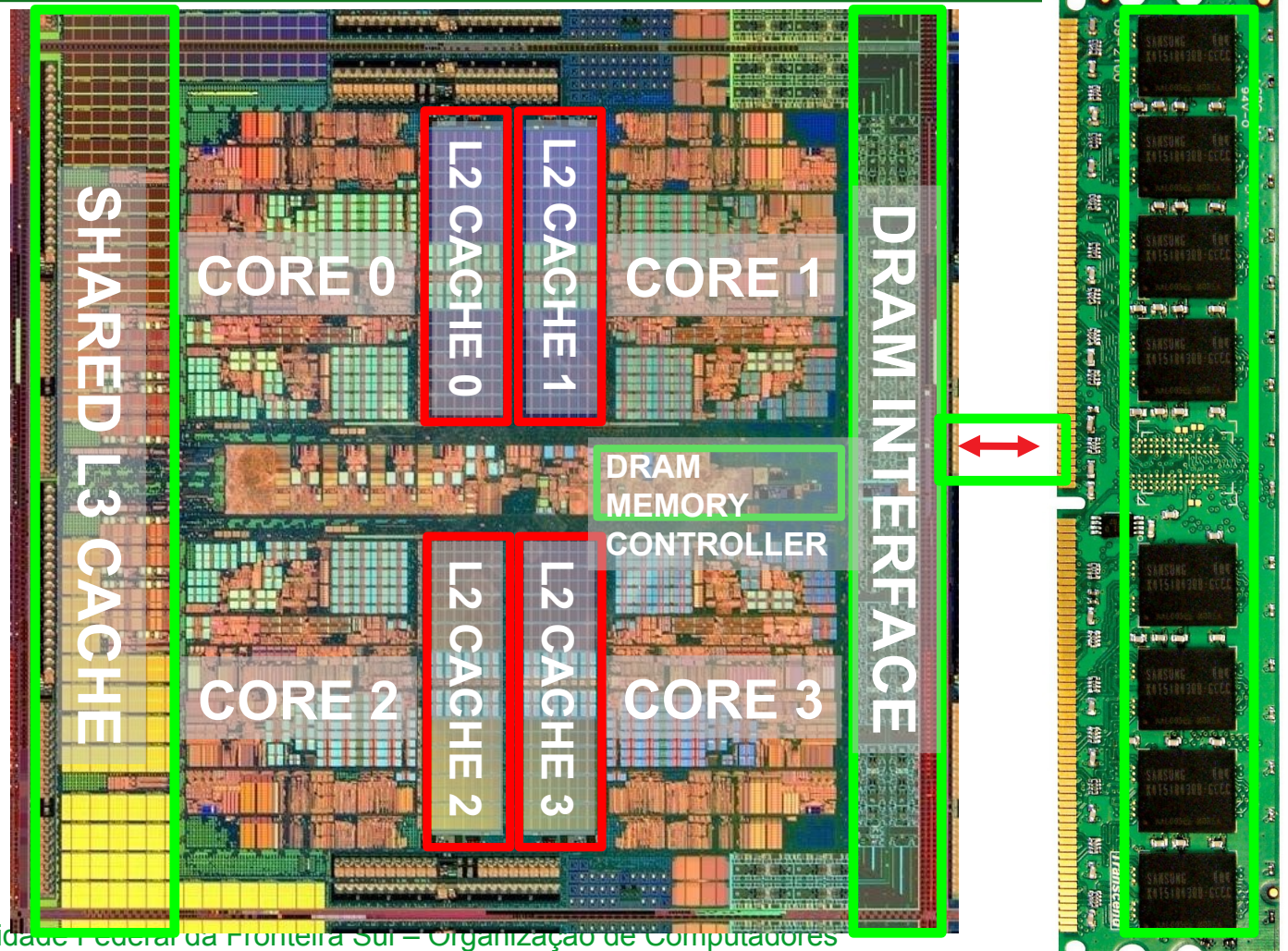
- Memória estática (construída a partir de Flip-Flops) disposta logicamente entre a CPU e a Memória Principal



- Capacidade de armazenamento menor que a MP mas tempo de acesso melhor (mais rápida)

Hierarquia de Memória

Processador Intel I5



Memória Cache

- Programas de computador são essencialmente sequenciais (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 adjacente, dado a natureza sequencial dos programas

Localidade Temporal

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

```
67  INICIO_LACO:
68                  beq     t2, t3, FIM_LACO
69                  add     a0, zero, t1            # carrega limite para % (resto da divisão)
70                  jal     PSEUDO_RAND
71                  add     t4, zero, a0            # pega linha sorteada e coloca em t4
72                  add     a0, zero, t1            # carrega limite para % (resto da divisão)
73                  jal     PSEUDO_RAND
74                  add     t5, zero, a0            # pega coluna sorteada e coloca em t5
75
76  LE_POSICAO:
77                  mul     t4, t4, t1
78                  add     t4, t4, t5              # calcula (L * tam) + C
79                  add     t4, t4, t4              # multiplica por 2
80                  add     t4, t4, t4              # multiplica por 4
81                  add     t4, t4, t0              # calcula Base + deslocamento
82                  lw      t5, 0(t4)               # Le posicao de memoria LxC
83  VERIFICA_BOMBA:
84                  addi    t6, zero, 9             # se posição sorteada já possui bomba
85                  beq     t5, t6, PULA_ATRIB      # pula atribuição
86                  sw      t6, 0(t4)               # senão coloca 9 (bomba) na posição
87                  addi    t3, t3, 1               # incrementa quantidade de bombas sorteadas
88  PULA_ATRIB:
89                  j       INICIO_LACO
```

Memória Cache

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

Localidade Espacial

Mover blocos de palavras contíguas 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

- **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% na busca de instruções resultaria na adição de 0.5 ciclos ao CPI médio

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

- 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:

$$\text{Número de células} = B * K$$

ou $$\text{Número de células} = 2^E$$

ou $$2^E = B * K$$

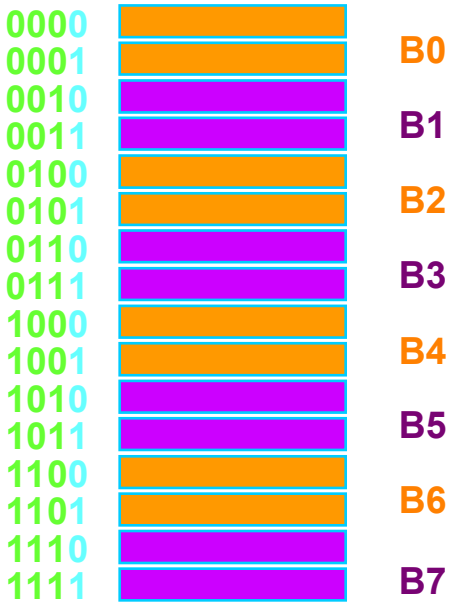
Memória Cache

Número de células = 16

B = 8 ← 3 bits

K = 2 ← 1 bit

E = 4 bits



$2^N = B$

$2^N = K$

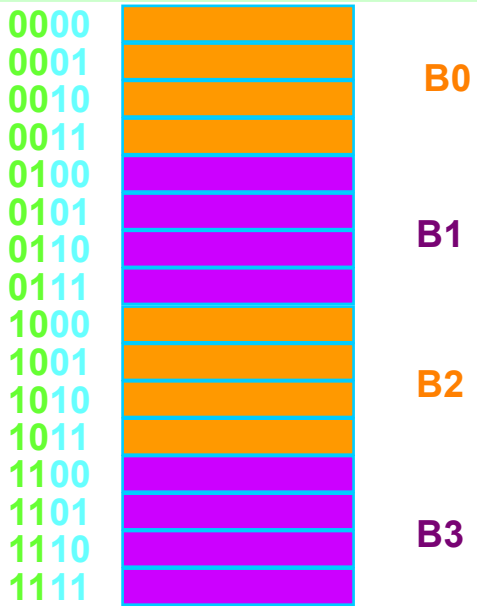
N = Número de Bits

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:

$$\text{Número do Bloco} = \frac{\text{Endereço}}{K} \quad (\text{parte inteira})$$

$$\text{Deslocamento do Bloco} = \text{Endereço} \ \ \text{MOD} \ \ K$$

Por exemplo, considerando:

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:

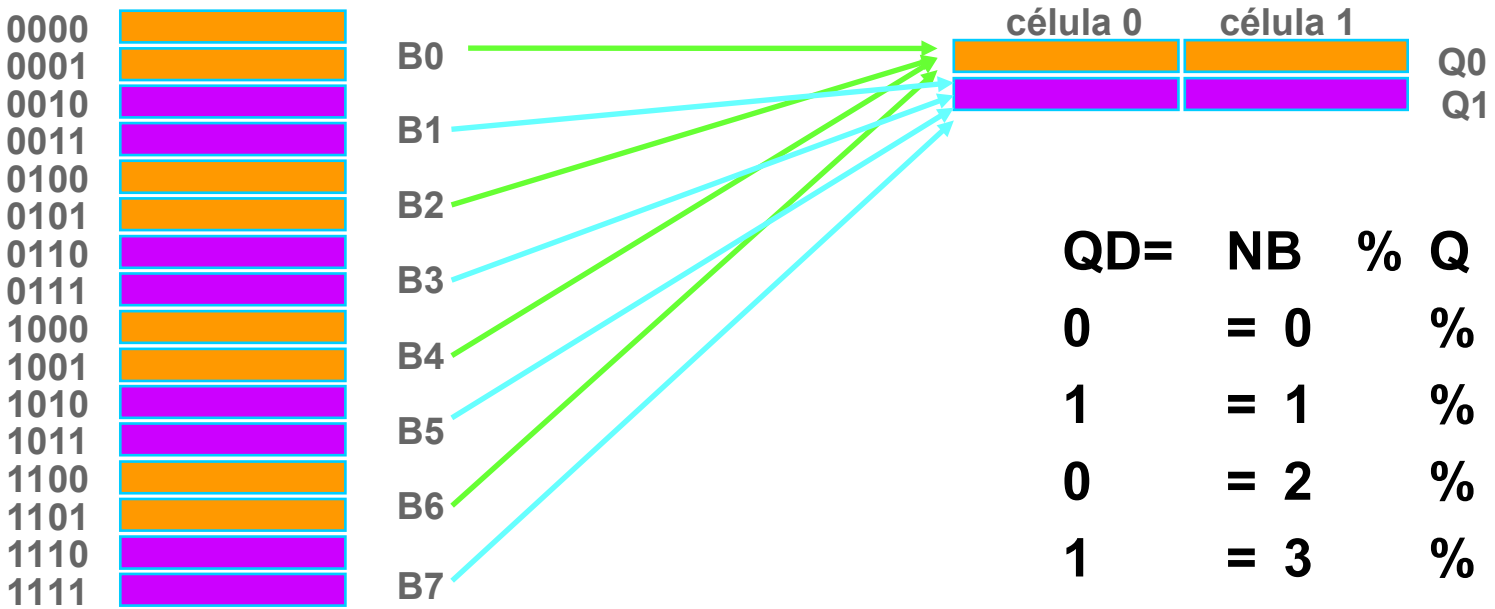
$$QD = NB \% Q$$

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

Mapeamento Direto

Memória Cache

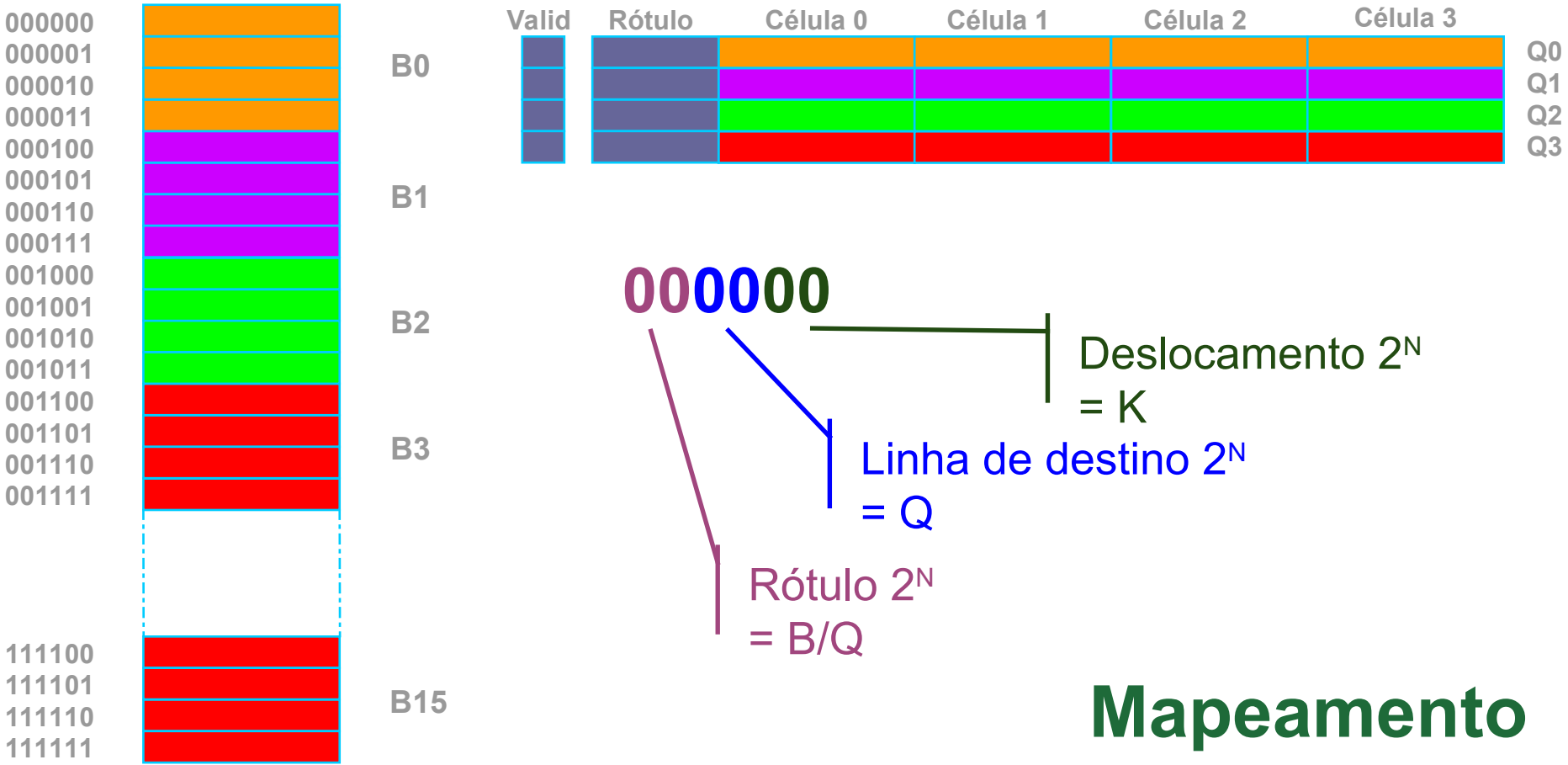
18) Com base na figura abaixo que descreve um sistema MP/Cache e sabendo que se utiliza mapeamento direto, responda, justificando todas as questões:

21	000000000
32	000000001
E8	000000010
	...
F2	111111111

Rótulo	Cel 0	Cel 1	Cel 2	Cel 3	
01	...				Q0
01	...				Q1
...					.
...					.
...					.
11	...				Q29
00	...				Q30
11	...				Q31

a) Qual a divisão do endereço deste sistema?
b) A qual linha da memória cache está destinado o endereço da MP 101110110?
c) Qual o endereço de memória que encontra-se na 1ª célula da linha Q30 M. cache?
d) Qual a quantidade de blocos de memória que o sistema possui? Justifique.
e) Qual o tamanho da memória cache e da memória principal em bytes? Justifique.

Memória Cache



Mapeamento Direto

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

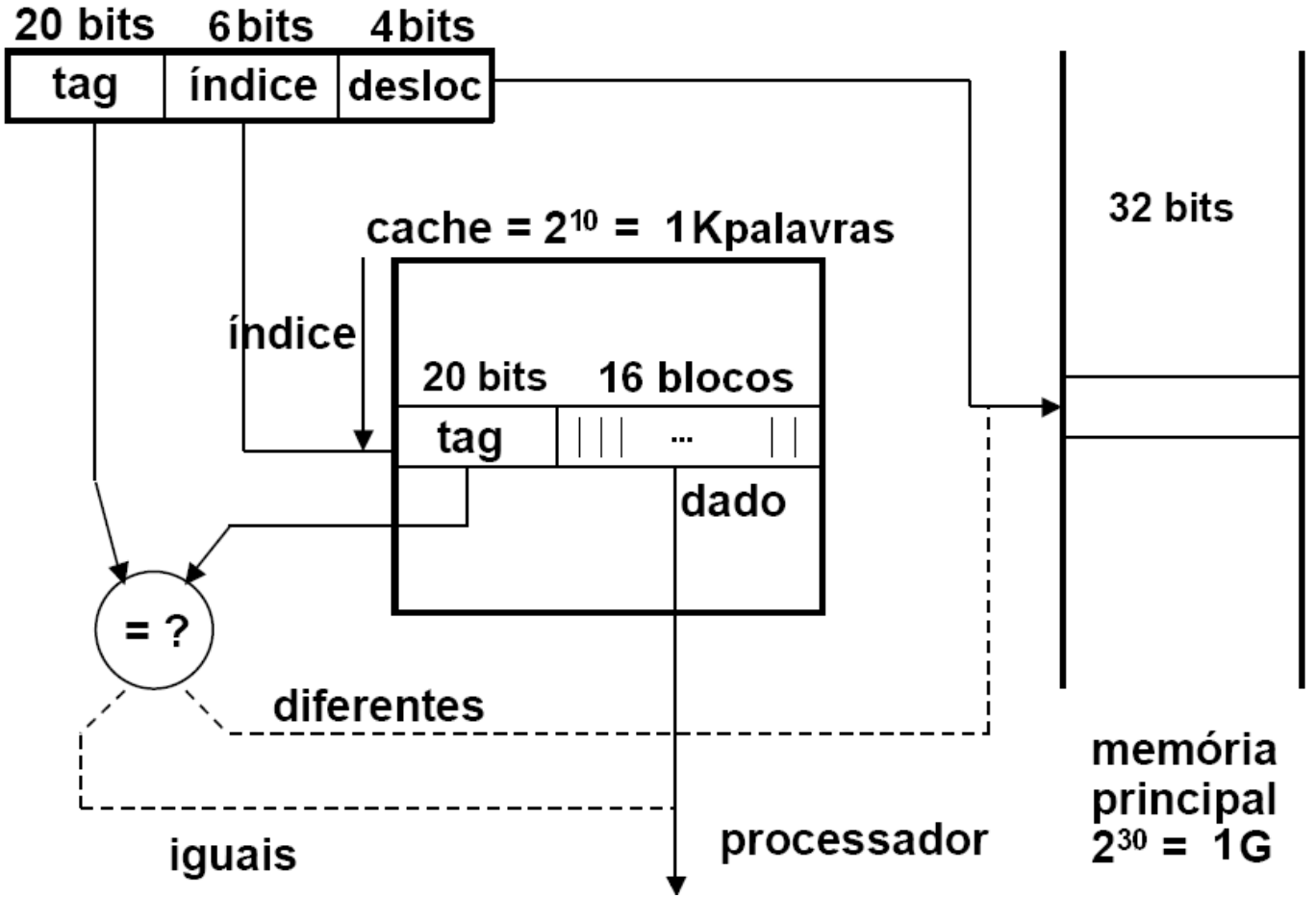
3. O controlador de cache examina por comparação o valor do rótulo do quadro 1 da cache com os 2 bits mais significativos do endereço solicitado ($10XXXX_2$). Caso sejam iguais realiza o passo 4, senão o passo 5;

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

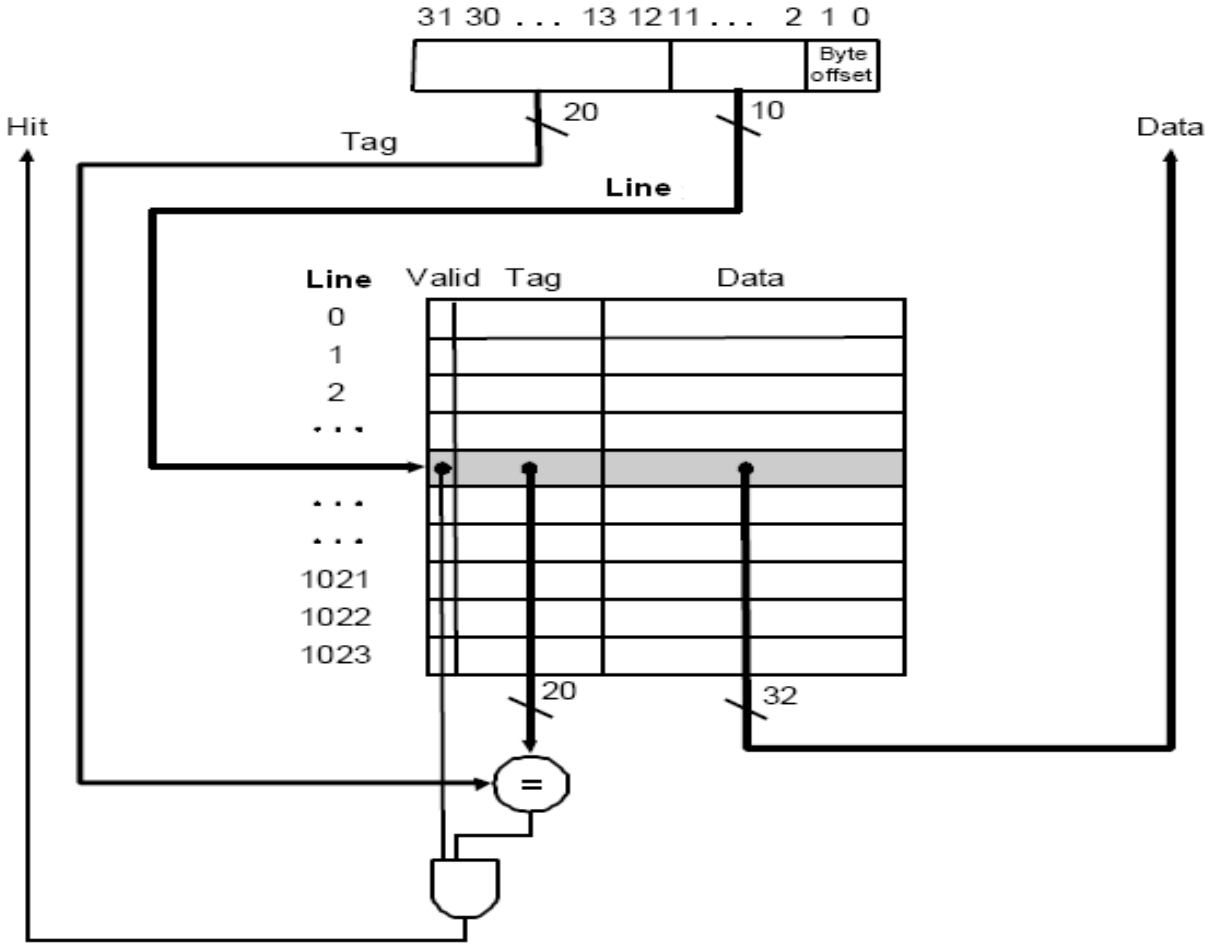
5. Se no passo 3 a comparação resulta negativa (ou seja, o bloco desejado não se encontra no quadro 1), o mesmo será transferido da MP para a cache, substituindo o bloco atual. Para esta última tarefa são utilizados os 4 bits mais significativos do endereço solicitado ($1001XX_2$) correspondentes ao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

Memória Cache

Mapeamento Direto



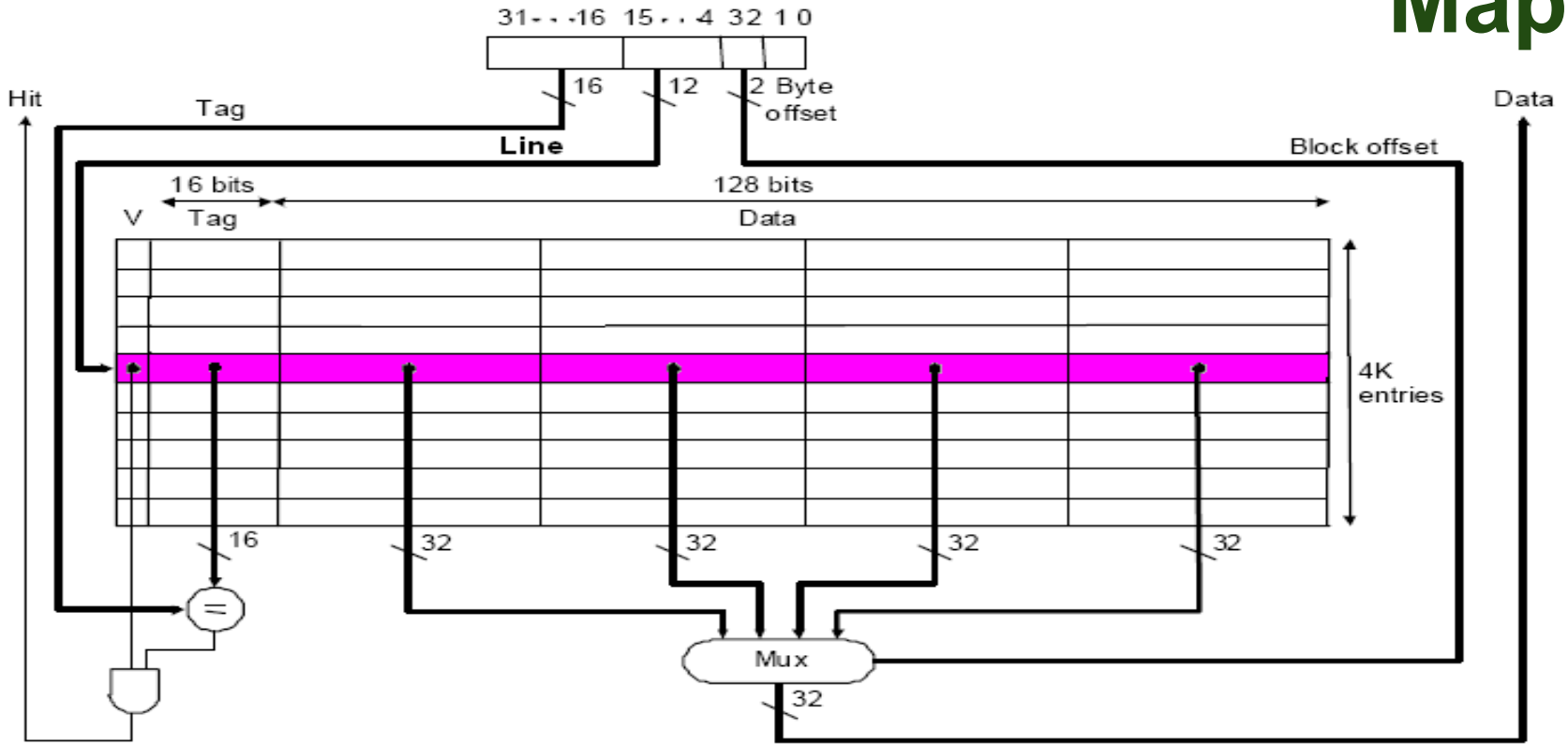
Memória Cache



Mapeamento Direto

Memória Cache

Mapeamento Direto



Mapeamento Direto

Conclusões:

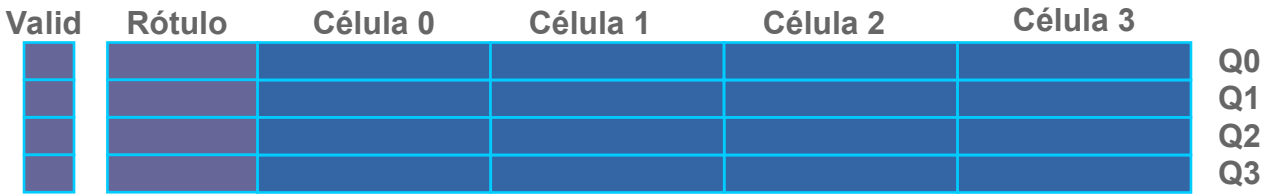
- Método simples e de baixo custo de implementação (um multiplexador e um comparador);
- Blocos que disputam o mesmo quadro/linha não podem estar na cache concomitantemente;
- Overhead baixo:

  Overhead = Nº Bits política / Nº Bits Dados

Memória Cache

Mapeamento Associativo

- Qualquer bloco pode estar em qualquer linha da cache



- O valor presente no campo rótulo da cache é o próprio número do bloco
- Quando todas as linhas estiverem ocupadas e houver o acesso a um endereço que não está presente na cache, deve-se aplicar uma política de substituição para definir qual linha será utilizada para armazenar o bloco solicitado

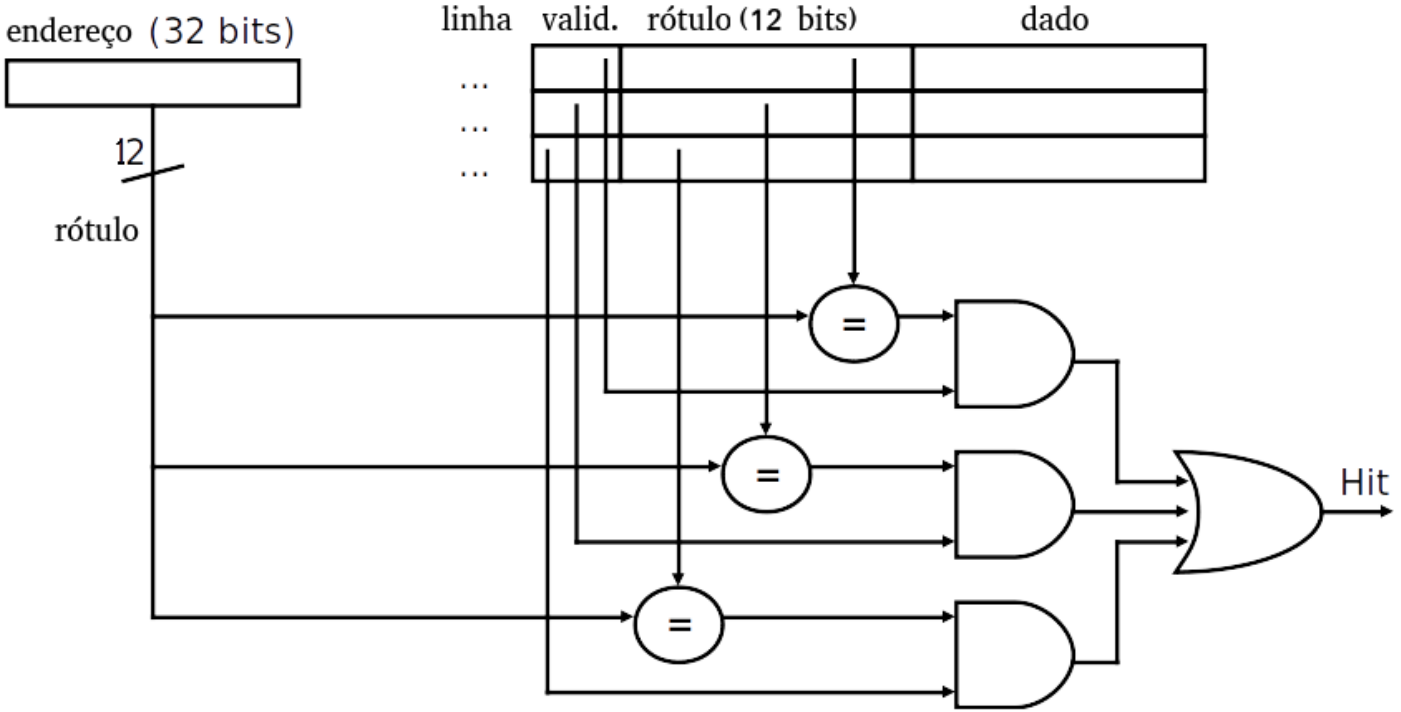
https://www.scss.tcd.ie/Jeremy.Jones/VivioJS/caches/cache.htm

Memória Cache

Políticas de substituição

- **FIFO – First In First Out**: como uma fila, substitui o bloco que entrou primeiro e assim sucessivamente
- **LRU – Least Recently Used**: substitui o bloco presente na linha que não é utilizada a mais tempo
- **LFU – Least Frequently Used**: substitui o bloco presente na linha que tem menos acessos
- **Aleatória** – escolhe aleatoriamente a linha que será utilizada para armazenar o bloco solicitado

Memória Cache



Mapeamento Associativo

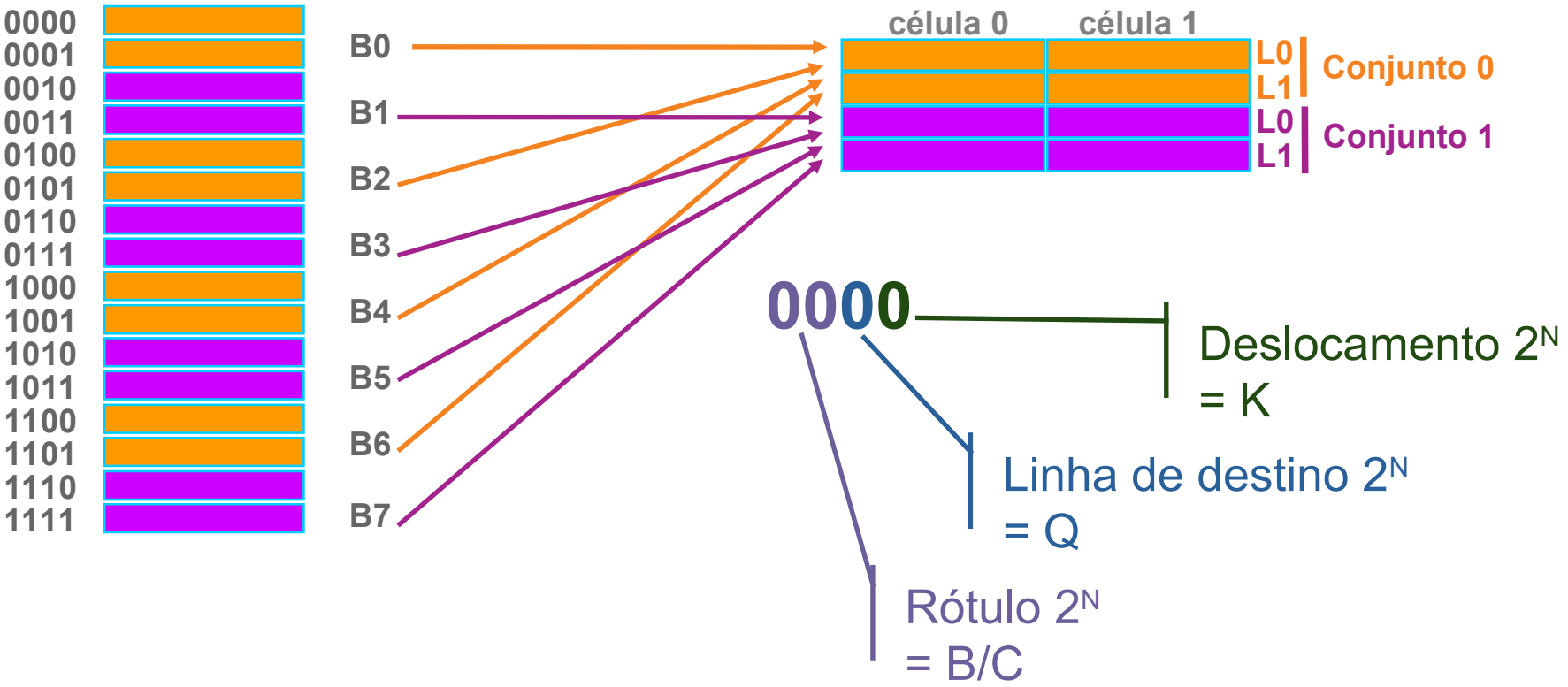
Memória Cache

Mapeamento Associativo por Conjuntos

- Como no mapeamento direto, cada bloco é pré destinado a um determinado conjunto
- Como no mapeamento associativo, dentro do conjunto o bloco pode ser colocado em qualquer das linhas

Memória Cache

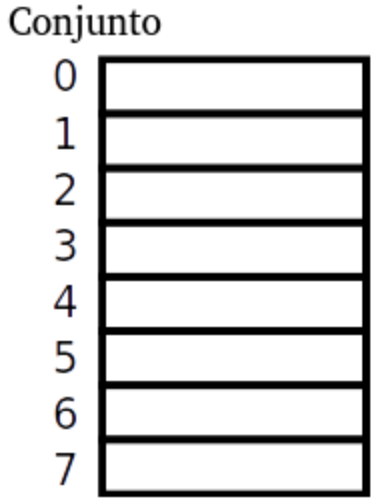
Mapeamento Associativo por Conjunto



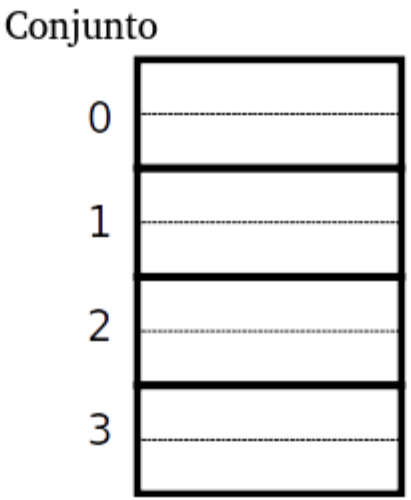
Memória Cache

Mapeamento Associativo por Conjuntos

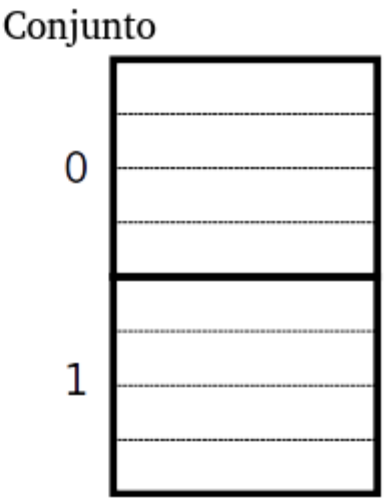
1-way associativity
8 sets, 1 block each



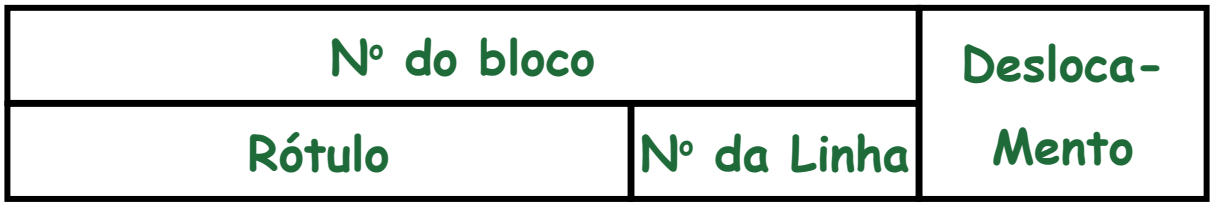
2-way associativity
4 sets, 2 blocks each



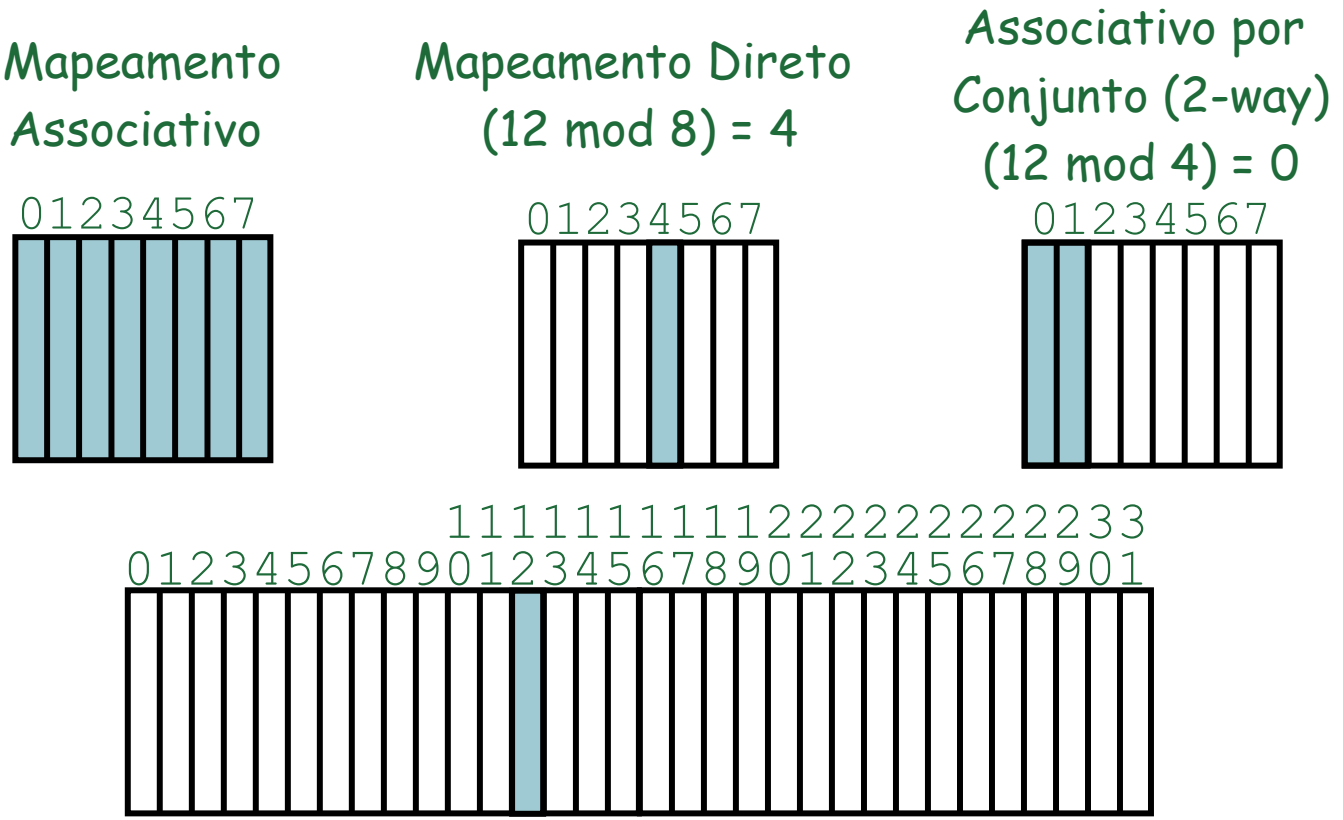
4-way associativity
2 sets, 4 blocks each



Memória Cache



Memória Cache



Virtual Memory

Protection via virtual memory

Keeps processes in their own memory space

Role of architecture:

Provide user mode and supervisor mode

Protect certain aspects of CPU state

Provide mechanisms for switching between user mode and supervisor mode

Provide mechanisms to limit memory accesses

Provide TLB to translate addresses

Virtual Memory

Supports isolation and security

Sharing a computer among many unrelated users

Enabled by raw speed of processors, making the overhead more acceptable

Allows different ISAs and operating systems to be presented to user programs

"System Virtual Machines"

SVM software is called "virtual machine monitor" or "hypervisor"

Individual virtual machines run under the monitor are called "guest VMs"

Virtual Memory

Each guest OS maintains its own set of page tables

VMM adds a level of memory between physical and virtual memory called “real memory”

VMM maintains shadow page table that maps guest virtual addresses to physical addresses

- **Requires VMM to detect guest’s changes to its own page table**
- **Occurs naturally if accessing the page table pointer is a privileged operation**