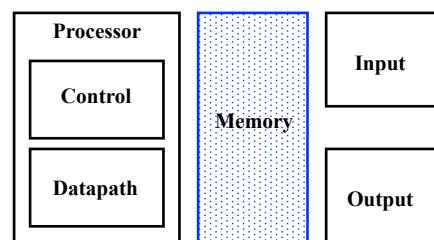


Memória

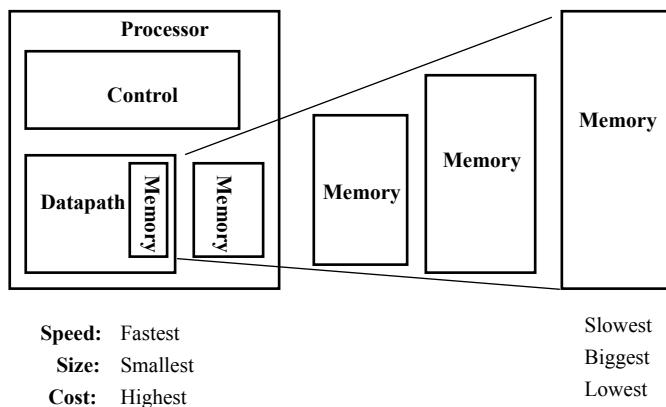
António de Brito Ferrari

ferrari@ua.pt

Memória



Hierarquia de Memória



ABF - AC1 - Memória

3

Localidade das Referências



Hierarquia funciona devido ao princípio da localidade - os programas acedem a uma pequena porção do Address Space em cada instante:

- **Localidade Temporal** – se um item é referenciado tende a sê-lo de novo em breve
- **Localidade Espacial** – se um item é referenciado items com endereços próximos tendem a sê-lo em breve

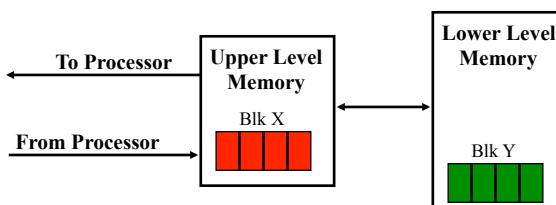
ABF - AC1 - Memória

4

Funcionamento da Hierarquia de Memória

Localidade Temporal => manter os dados acedidos mais recentemente na memória mais rápida

Localidade Espacial => mover blocos contíguos para a memória mais rápida



Hierarquia de memória

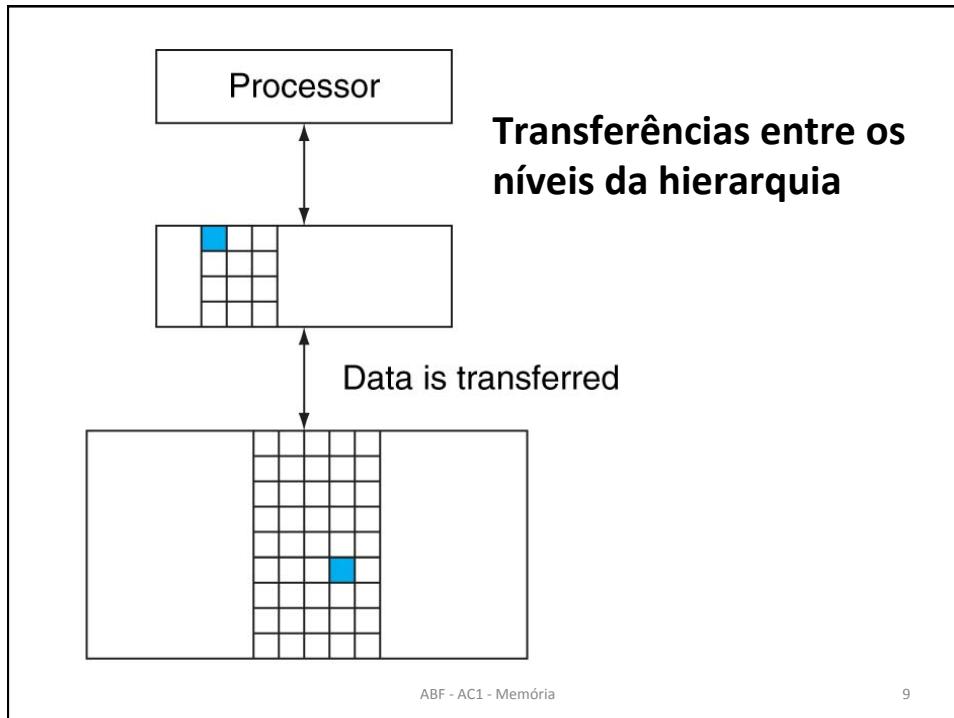
- Tirar vantagem do princípio da localidade das referências:
 - Oferecer ao utilizador tanta memória quanta a disponível com a tecnologia mais barata
 - Aceder à informação à velocidade oferecida pela tecnologia mais rápida

Memória: tipos de acesso

- Random Access:
 - Tempo de acesso idêntico para todas as posições de memória
 - **DRAM**: Dynamic Random Access Memory
 - High density, low power; mais lenta e mais barata que SRAM
 - Dynamic: periodicamente precisa de ser “refreshed”
 - **SRAM**: Static Random Access Memory
 - Low density, high power; mais rápida e mais cara que DRAM
 - Static: conteúdo mantém-se enquanto estiver ligada
- “Not-so-random” Access:
 - Tempo de acesso varia com a localização e varia também ao longo do tempo
 - Exemplos: Disk, CDROM
- Acesso Sequencial:
 - Exemplo: banda magnética (Tape)

Estrutura da Hierarquia de Memória

Speed	Processor	Size	Cost (\$/bit)	Current technology
Fastest	Memory	Smallest	Highest	SRAM
	Memory			DRAM
Slowest	Memory	Biggest	Lowest	Magnetic disk



Tipos de Memória

Preço vs. Tempo de Acesso

Memory technology	Typical access time	\$ per GiB in 2012
SRAM semiconductor memory	0.5–2.5 ns	\$500–\$1000
DRAM semiconductor memory	50–70 ns	\$10–\$20
Flash semiconductor memory	5,000–50,000 ns	\$0.75–\$1.00
Magnetic disk	5,000,000–20,000,000 ns	\$0.05–\$0.10

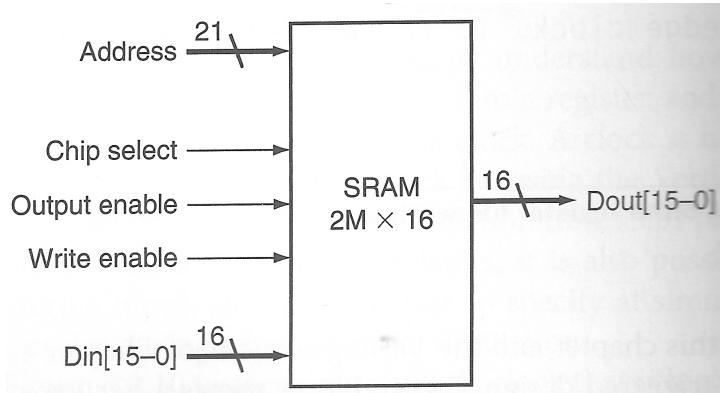
Main Memory: **DRAM** - Dynamic Random Access Memory
 Dynamic porque precisa de ser periodicamente “refreshed” (8 ms)
 Endereços divididos em duas metades (Memória - 2D matrix):
RAS or Row Access Strobe
CAS or Column Access Strobe

Cache: **SRAM** - Static Random Access Memory
 No refresh (6 transistors/bit vs. 1 transistor/bit)
Size: DRAM/SRAM - **4-8**,
Cost/Cycle time: SRAM/DRAM - **8-16**

ABF - AC1 - Memória

10

SRAM (2M X 16)



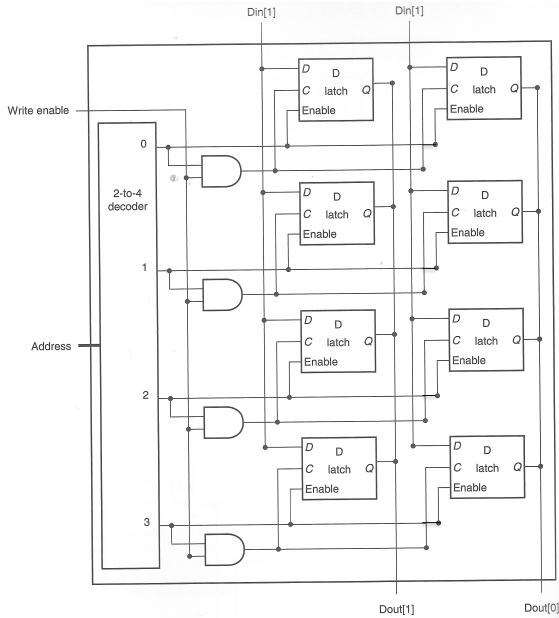
Read access time: tempo entre **OE** = true e **Address** válido e **Dout** = **Mem[Address]**

Escrita: **WE** e **CS** true – Data em **Din** escrita em **Mem[Address]**

ABF - AC1 - Memória

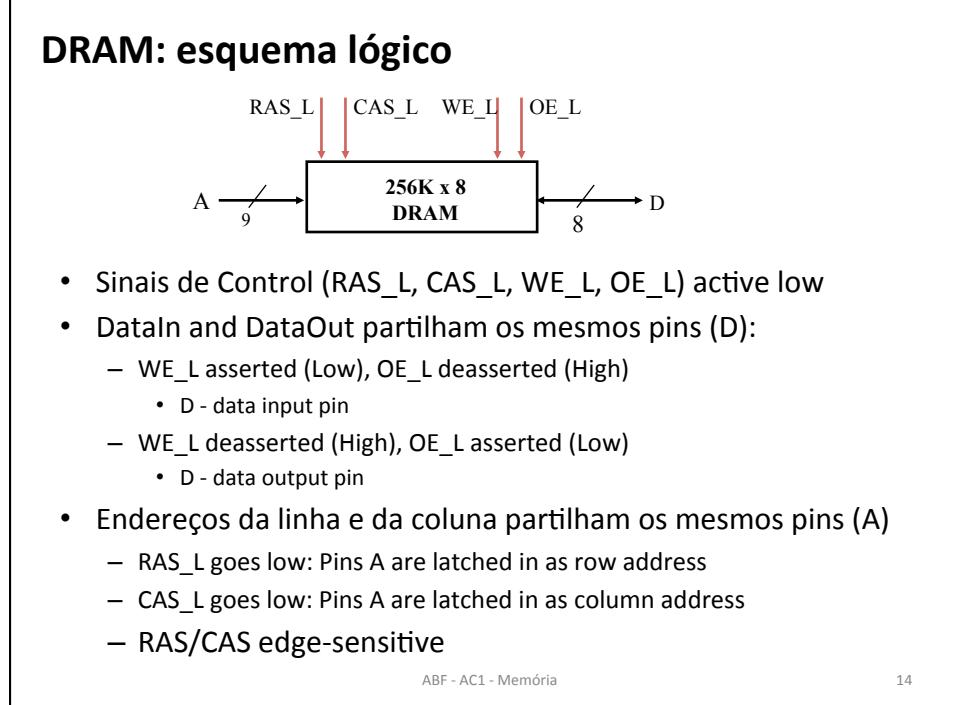
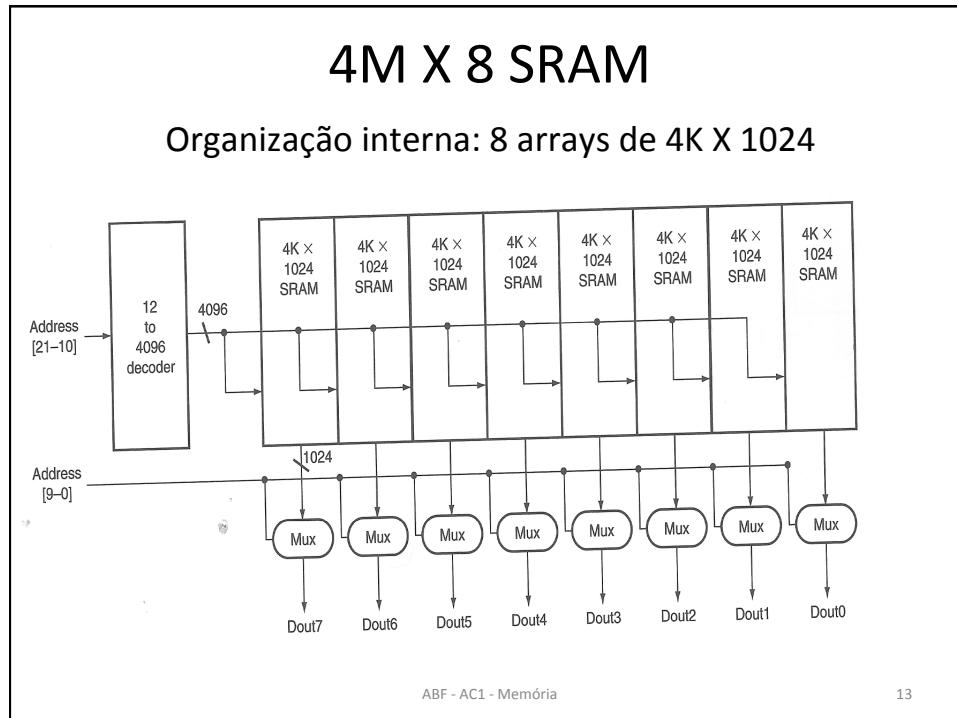
11

4X2 SRAM

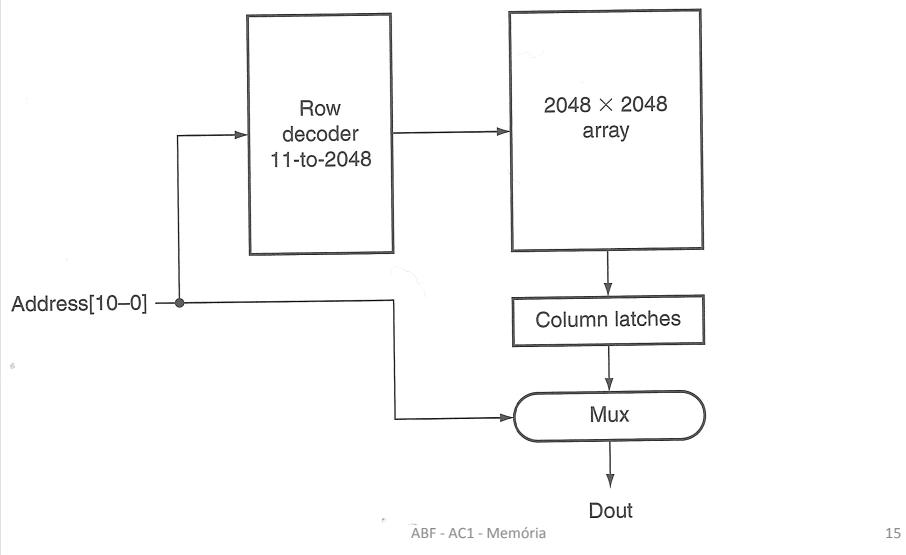


ABF - AC1 - Memória

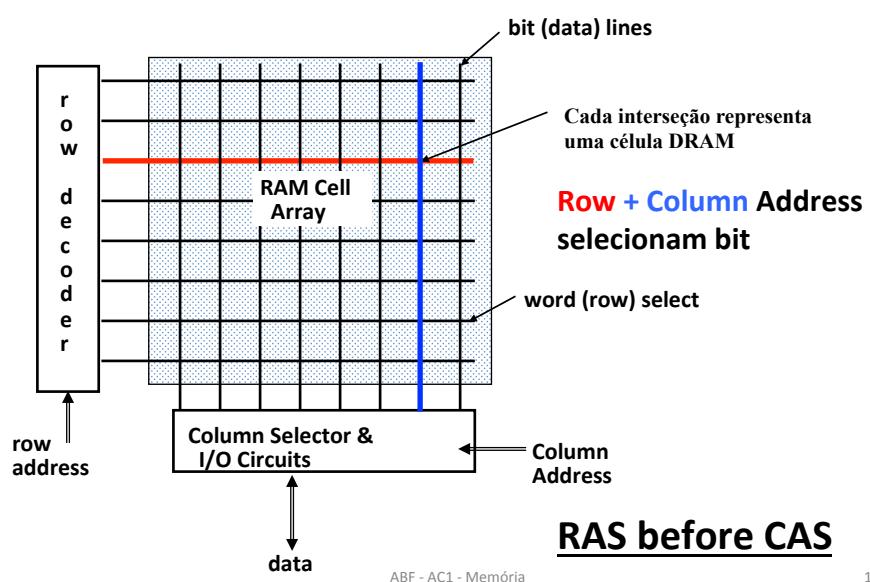
12



4M X 1 DRAM (Matriz de 2048 X 2048)

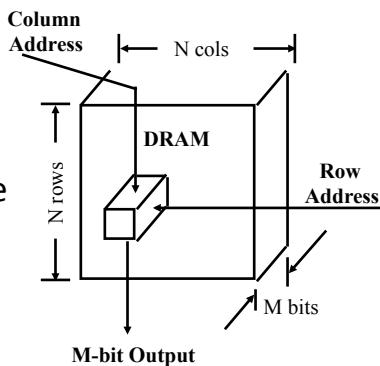


DRAM



DRAM Organization

- N rows $\times N$ column $\times M$ -bit
- Read & Write M -bit at a time
- Each M -bit access requires a RAS / CAS cycle



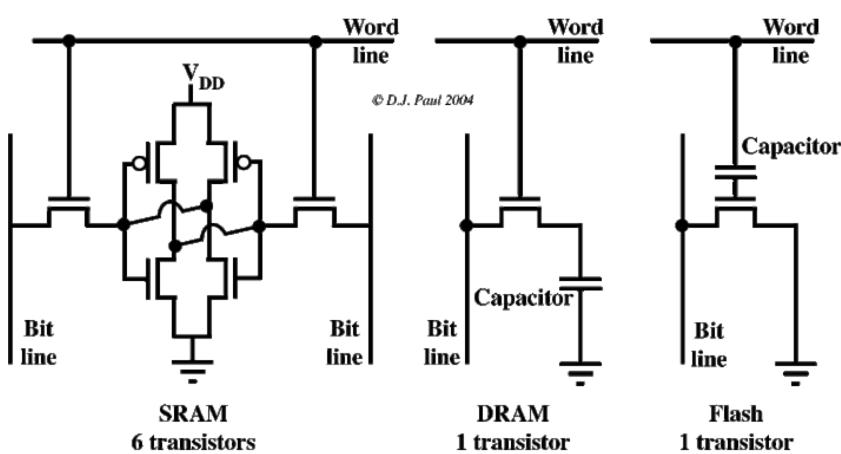
Fast Page Mode DRAM:

- $N \times M$ “register” to save a row

ABF - AC1 - Memória

17

Células de Memória



ABF - AC1 - Memória

18

Evolução da DRAM

Year introduced	Chip size	\$ per GiB	Total access time to a new row/column	Average column access time to existing row
1980	64 Kibibit	\$1,500,000	250 ns	150 ns
1983	256 Kibibit	\$500,000	185 ns	100 ns
1985	1 Mebibit	\$200,000	135 ns	40 ns
1989	4 Mebibit	\$50,000	110 ns	40 ns
1992	16 Mebibit	\$15,000	90 ns	30 ns
1996	64 Mebibit	\$10,000	60 ns	12 ns
1998	128 Mebibit	\$4,000	60 ns	10 ns
2000	256 Mebibit	\$1,000	55 ns	7 ns
2004	512 Mebibit	\$250	50 ns	5 ns
2007	1 Gibibit	\$50	45 ns	1.25 ns
2010	2 Gibibit	\$30	40 ns	1 ns
2012	4 Gibibit	\$1	35 ns	0.8 ns

ABF - AC1 - Memória

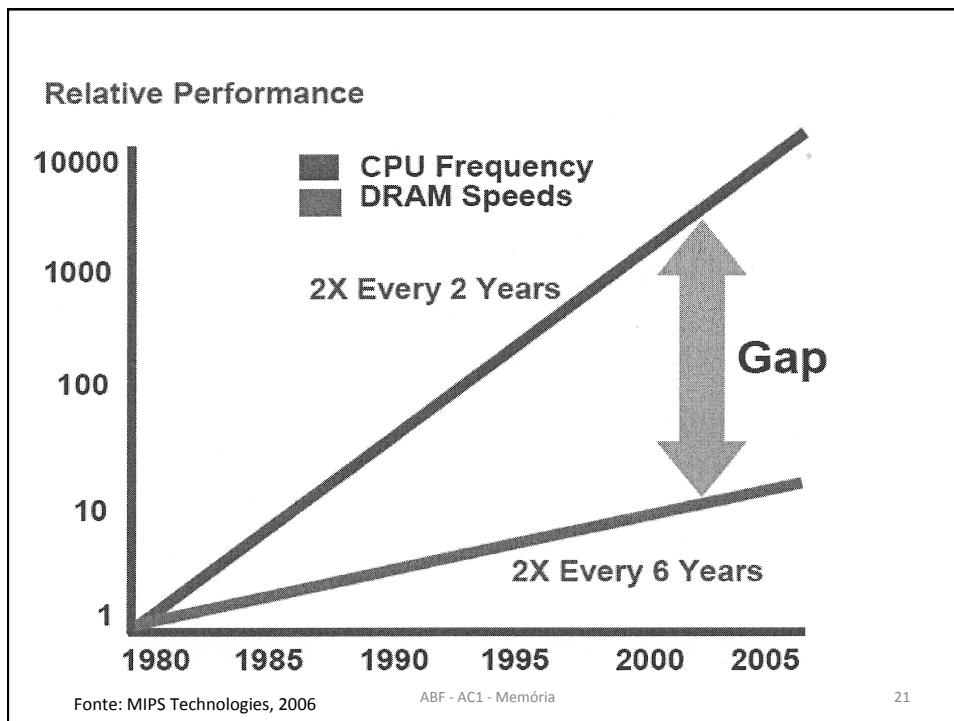
19

Evolução da tecnologia

	Capacity	Speed (latency)
Logic:	2x in 3 years	2x in 3 years
DRAM:	4x in 3 years	2x in 10 years
Disk:	4x in 3 years	2x in 10 years

ABF - AC1 - Memória

20



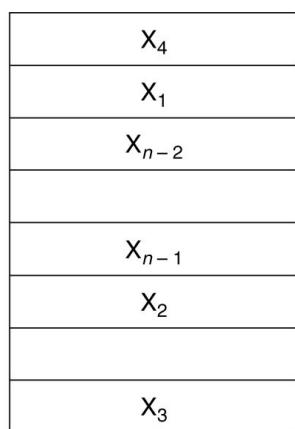
Memória Cache

Cache Memory

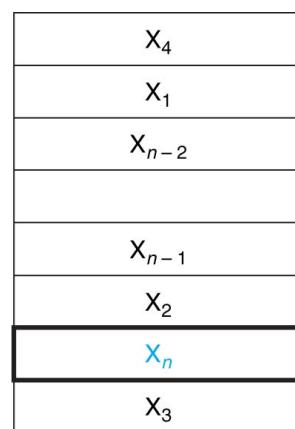
- Ideia: guardar numa memória mais rápida a informação a que o programa está a aceder – ***cache memory***
 - “... a fast core memory of, say, 32.000 words as a slave to a slower core memory of, say, 1 million words, in such a way that in practical cases the effective access time is nearer that of the fast memory than that of the slow memory.”

Maurice Wilkes, “Slave Memories and Dynamic Storage Allocation”, IEEE Tr. EC-14, **1965**

Preenchimento da Cache: “*Demand driven*”



a. Before the reference to X_n



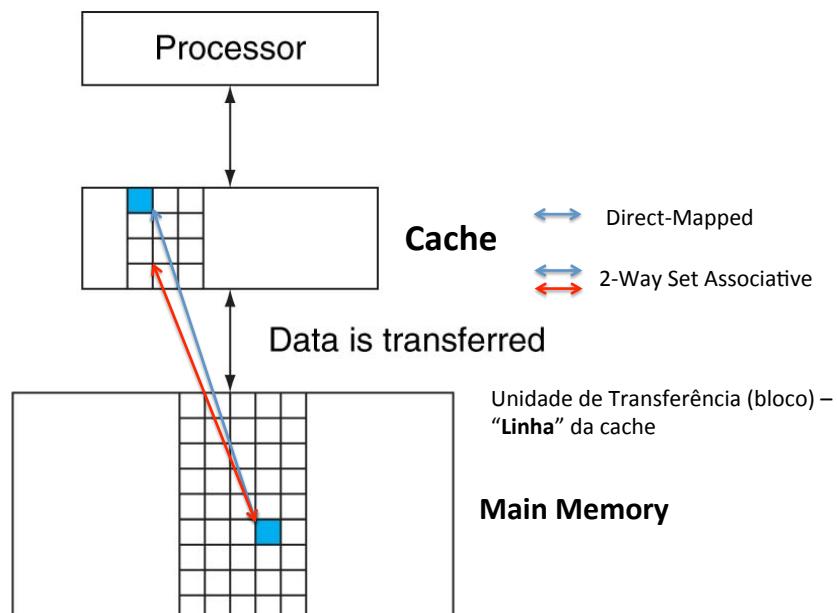
b. After the reference to X_n

Cache e Memória Central

- Cache: SRAM
 - Tempo de acesso: 2ns-4ns
- Memória Central: DRAM (SDRAM)
 - Tempo de acesso: 35ns-40ns
- **Todos os computadores atuais incluem Caches**
- Organização da cache:
 - **Direct-Mapped** – cada posição de memória mapeada numa única posição na cache
 - **N-Way Set Associative** - cada posição de memória pode ser mapeada em N posições diferentes da cache
 - **Fully Associative** - qualquer posição de memória pode ser mapeada em qualquer posição da cache

ABF - AC1 - Memória

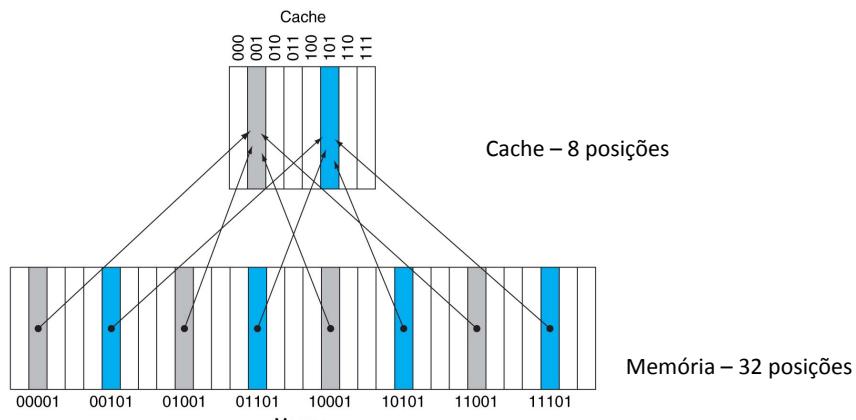
25



ABF - AC1 - Memória

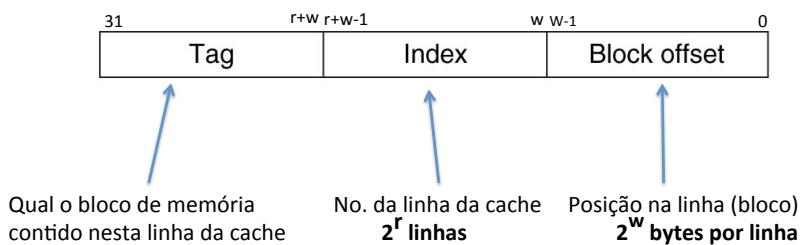
26

Direct-Mapped Cache



Duas informações necessárias: qual o bloco mapeado em cada posição **tag**
se posição da cache contem informação válida **valid bit**

Endereçagem



Nos slides seguintes assume-se que
cada linha (bloco) da cache tem a dimensão de uma *word*

Preenchimento da Cache

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

a. The initial state of the cache after power-on

Index	V	Tag	Data
000	N		
001	N		
010	Y	11 _{two}	Memory (11010 _{two})
011	N		
100	N		
101	N		
110	Y	10 _{two}	Memory (10110 _{two})
111	N		

b. After handling a miss of address (10110_{two})

Index	V	Tag	Data
000	Y	10 _{two}	Memory (10000 _{two})
001	Y	11 _{two}	Memory (11010 _{two})
010	N		
011	N		
100	N		
101	N		
110	Y	10 _{two}	Memory (10110 _{two})
111	N		

c. After handling a miss of address (11010_{two})

Index	V	Tag	Data
000	Y	10 _{two}	Memory (10000 _{two})
001	N		
010	Y	11 _{two}	Memory (11010 _{two})
011	Y	00 _{two}	Memory (00011 _{two})
100	N		
101	N		
110	Y	10 _{two}	Memory (10110 _{two})
111	N		

d. After handling a miss of address (10000_{two})

Index	V	Tag	Data
000	Y	10 _{two}	Memory (10000 _{two})
001	N		
010	Y	10 _{two}	Memory (10010 _{two})
011	Y	00 _{two}	Memory (00011 _{two})
100	N		
101	N		
110	Y	10 _{two}	Memory (10110 _{two})
111	N		

e. After handling a miss of address (00011_{two})

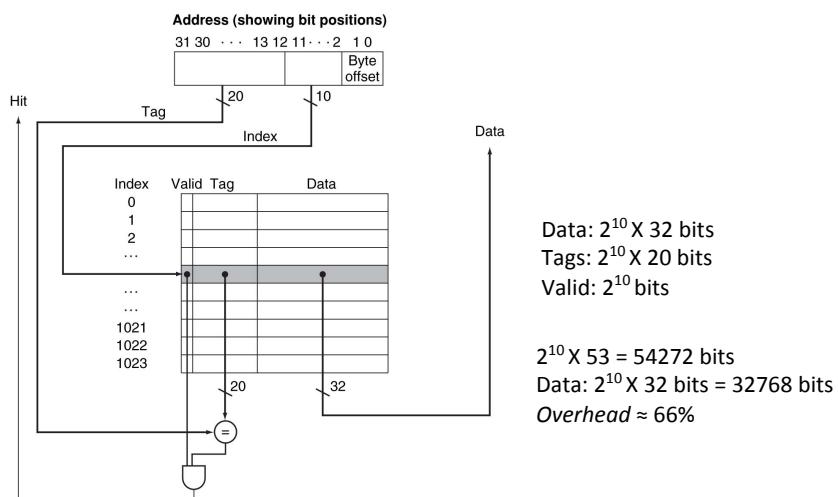
Index	V	Tag	Data
000	Y	10 _{two}	Memory (10000 _{two})
001	N		
010	Y	10 _{two}	Memory (10010 _{two})
011	Y	10 _{two}	Memory (10110 _{two})
100	N		
101	N		
110	Y	10 _{two}	Memory (10110 _{two})
111	N		

f. After handling a miss of address (10001_{two})

29

“Conflito”

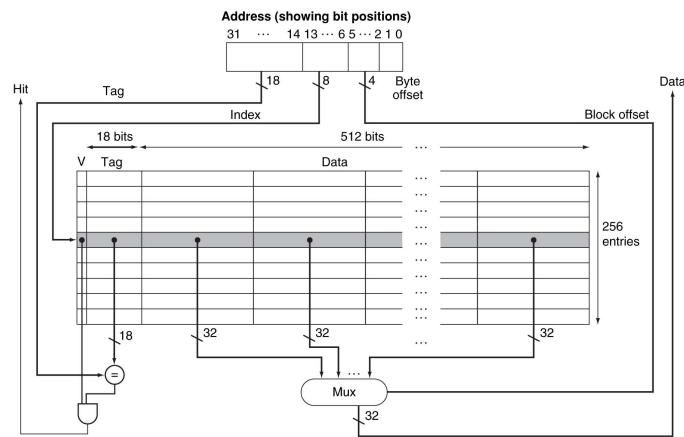
1k Word Direct Mapped Cache



ABF - AC1 - Memória

30

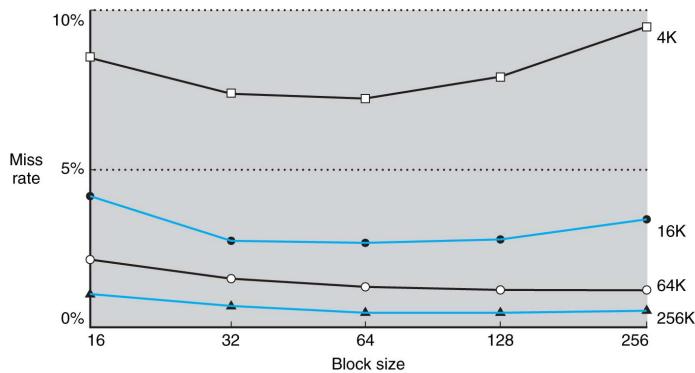
Tirar partido da Localidade Espacial das referências: Direct-Mapped Cache com 16 palavras por linha



ABF - AC1 - Memória

31

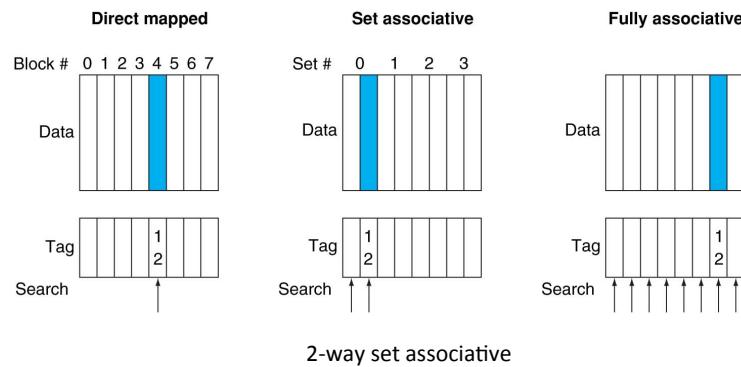
Miss rate vs. block (line) size



ABF - AC1 - Memória

32

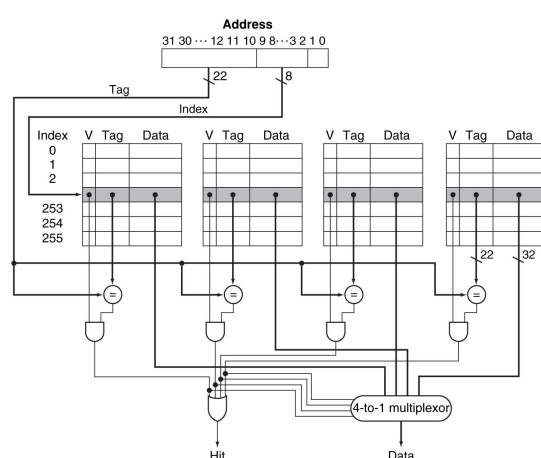
Outras formas de organização da cache



ABF - AC1 - Memória

33

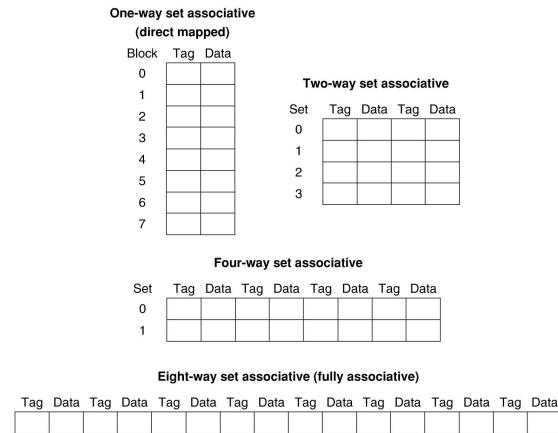
4-way set-associative



ABF - AC1 - Memória

34

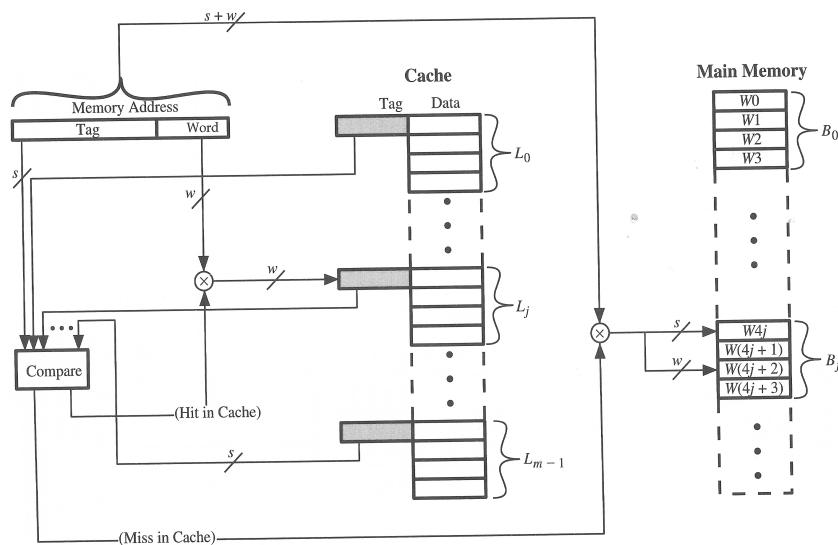
Diferentes organizações da Cache



ABF - AC1 - Memória

35

Fully_Associative Cache



ABF - AC1 - Memória

36

Memória Associativa (CAM)

- CAM – Content-Addressable Memory
 - Acesso à memória através da indicação do conteúdo da informação
 - Memória convencional (SRAM, DRAM, Flash) – acesso através da indicação da posição (endereço) da informação
- Acesso através do conteúdo – outras aplicações:
 - Por exemplo pesquisa de informação numa base de dados
- A memória humana funciona também de forma associativa – lembramo-nos de uma determinada informação, não da sua “posição” – CAM é usada na implementação de certo tipo de redes neurais (p.ex. nos “Mapas de Kohonen”).

Miss Rate: dependência do grau de associatividade

Associativity	Data miss rate
1	10.3%
2	8.6%
4	8.3%
8	8.1%

Miss rate depende de:

- Grau de associatividade
- Dimensão do bloco (linha)

Custo é função de:

- Dimensão da Cache
- Grau de associatividade

Grau de associatividade influencia o custo:

- Para igual dimensão da cache aumentar o grau de associatividade implica mais memória associativa (CAM – Content-Addressable Memory)
- ◆ Memória Associativa é **cara**

Fim da matéria de AC I em 2013/14

REVISÕES E DÚVIDAS