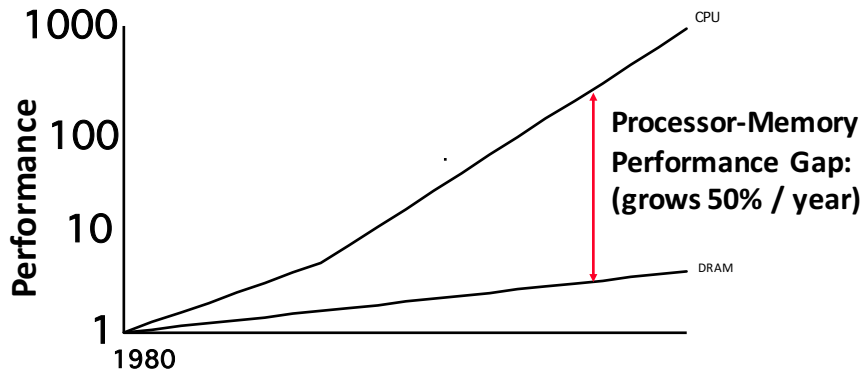
Memória Cache

Processor-DRAM Gap (latency)



Localidade das Referências

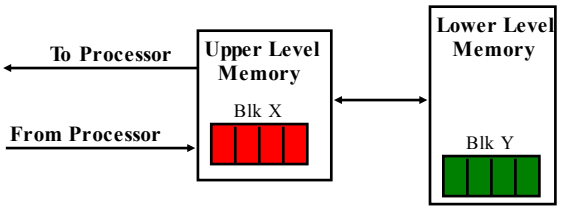
Hierarquia funciona devido ao princípio da localicade - os programas acedem a uma pequena porção do Address Space em cada instante:

- **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

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



Unidade de transferência (bloco) entre a memória central e a cache tem a dimensão de uma **linha da cache**

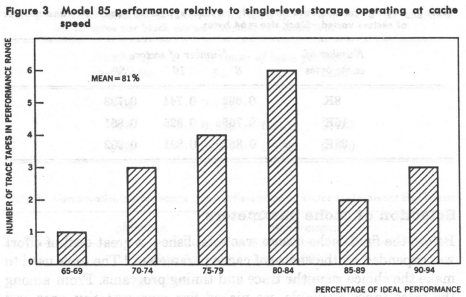
Cache Memory

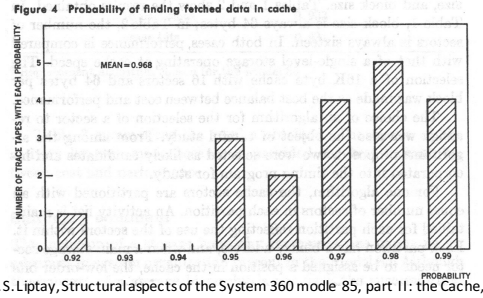
- Ideia: guardar numa memória mais rápida a informação a que o programa está a aceder – ***cache memory***
  - "*… a fast core memory of, say, 32.000 words as a slave to a slower core memory of, say, 1 million words, in such a way that in pratical cases the effective access time is nearer that of the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

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

Exemplo: Cache do IBM 360/85

- 1º computador com cache comercializado
  - Main Memory: núcleos de ferrite, 512KB a 4MB, 1.04μs cycle time
  - Cache: memória SRAM, 16 KB, 80ns cycle time
    - 64 bytes (4 palavras) por linha (1kB sectors)
    - 16-way set-associative
    - Write-through
    - Replacement policy: LRU

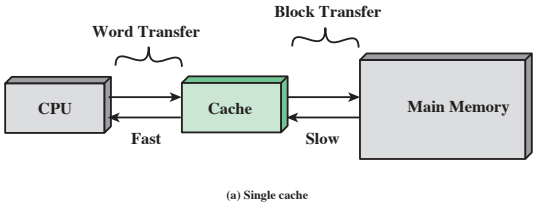




in J.S. Liptay, Structural aspects of the System 360 modle 85, part II: the Cache, IBM Systems J., 7(1), 1968

Memory performance:
Average Memory Acess Time (AMAT)
AMAT = t_{cache}+ MR_{cache}(t_{MM}+...)
t_{cache}- cache acess time
MR_{cache}- cache Miss Rate
t_{MM}- main memory access time

Cache – contem cópia parcial do conteúdo da memória para mais rápido acesso do processador aos dados



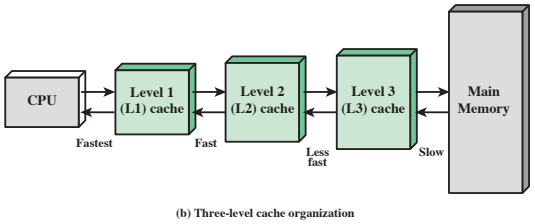
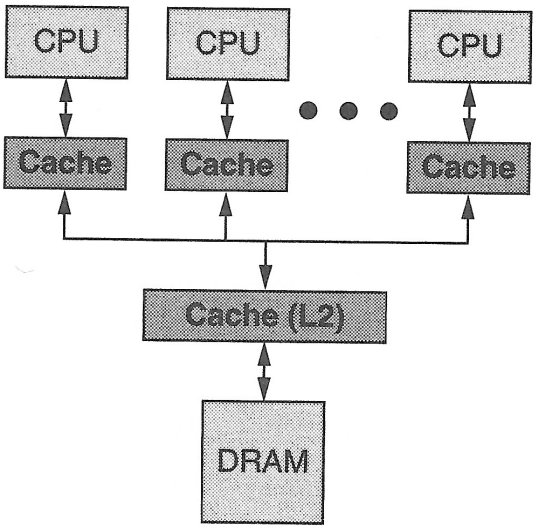


Figure 4.3 Cache and Main Memory

W.Stallings, *Computer Organization and Architecture*

Processadores atuais – vários níveis de cache
L1 < L2 < L3
$t_{L1} < t_{L2} < t_{L3}$

Processadores Multicore

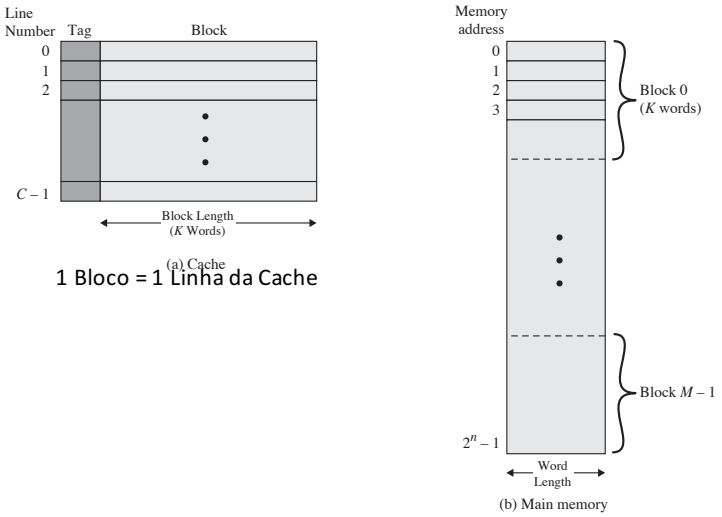


Cache L_1 – privada de cada *core* (CPU)

Cache L_2 – partilhada pelas várias *core* (CPU)

Processor	Type	Year of Introduction	L1 Cache[a]	L2 cache	L3 Cache
IBM 360/85	Mainframe	1968	16 to 32 kB	—	—
PDP-11/70	Minicomputer	1975	1 kB	—	—
VAX 11/780	Minicomputer	1978	16 kB	—	—
IBM 3033	Mainframe	1978	64 kB	—	—
IBM 3090	Mainframe	1985	128 to 256 kB	—	—
Intel 80486	PC	1989	8 kB	—	—
Pentium	PC	1993	8 kB/8 kB	256 to 512 KB	—
PowerPC 601	PC	1993	32 kB	—	—
PowerPC 620	PC	1996	32 kB/32 kB	—	—
PowerPC G4	PC/server	1999	32 kB/32 kB	256 KB to 1 MB	2 MB
IBM S/390 G6	Mainframe	1999	256 kB	8 MB	—
Pentium 4	PC/server	2000	8 kB/8 kB	256 KB	—
IBM SP	High-end server/ supercomputer	2000	64 kB/32 kB	8 MB	—
CRAY MTA[b]	Supercomputer	2000	8 kB	2 MB	—
Itanium	PC/server	2001	16 kB/16 kB	96 KB	4 MB
Itanium 2	PC/server	2002	32 kB	256 KB	6 MB
IBM POWER5	High-end server	2003	64 kB	1.9 MB	36 MB
CRAY XD-1	Supercomputer	2004	64 kB/64 kB	1MB	—
IBM POWER6	PC/server	2007	64 kB/64 kB	4 MB	32 MB
IBM z10	Mainframe	2008	64 kB/128 kB	3 MB	24-48 MB
Intel Core i7 EE 990	Workstaton/ server	2011	6 × 32 kB/32 kB	1.5 MB	12 MB
IBM zEnterprise 196	Mainframe/ Server	2011	24 × 64 kB/ 128 kB	24 × 1.5 MB	24 MB L3 192 MB L4

Cache e Memória Central



1 Bloco = 1 Linha da Cache

Figure 4.4 Cache/Main-Memory Structure

W.Stallings, *Computer Organization and Architecture*

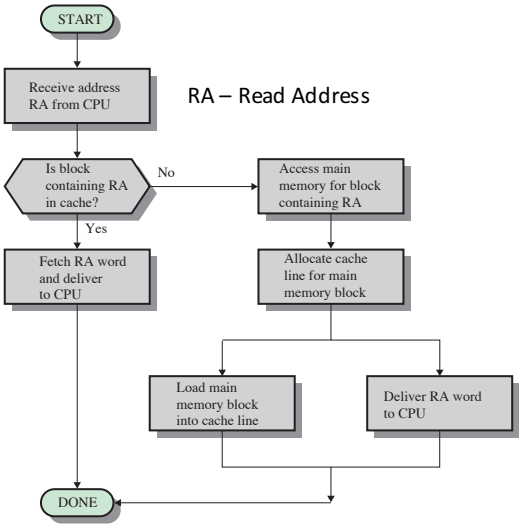
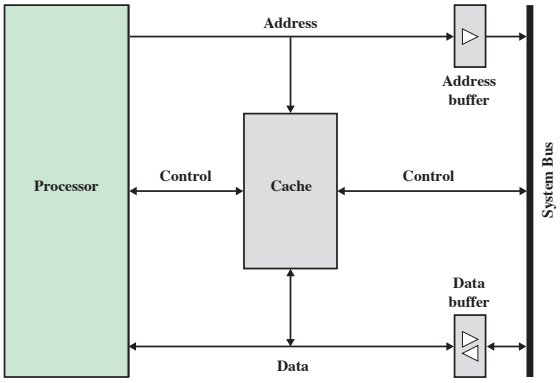


Figure 4.5 Cache Read Operation

W.Stallings, *Computer Organization and Architecture*

Acessso à Cache



Acesso à cache e à memória feitos em simultâneo

Figure 4.6 Typical Cache Organization

W.Stallings, *Computer Organization and Architecture*

Elements of Cache Design

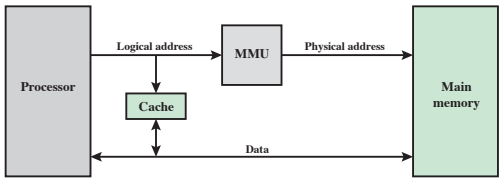
Cache Addresses	**Write Policy**
Logical	Write through
Physical	Write back
Cache Size	**Line Size**
Mapping Function	**Number of caches**
Direct	Single or two level
Associative	Unified or split
Set Associative	
Replacement Algorithm	
Least recently used (LRU)	
First in first out (FIFO)	
Least frequently used (LFU)	
Random	

1. Cache Addresses

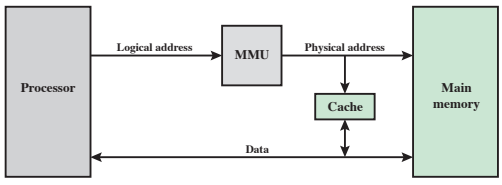
Memória Virtual

Mecanismo que permite aos programas endereçar a memória utilizando todo o espaço de endereçamento (endereços lógicos), independentemente da dimensão da memória física disponível (endereços físicos).
Assim os programas quando executados geram endereços lógicos que são traduzidos em endereços físicos pela *Memory Management Unit* (**MMU**)

MMU – componente hardware que traduz endereços lógicos em endereços físicos



(a) Logical Cache (Virtual Cache)



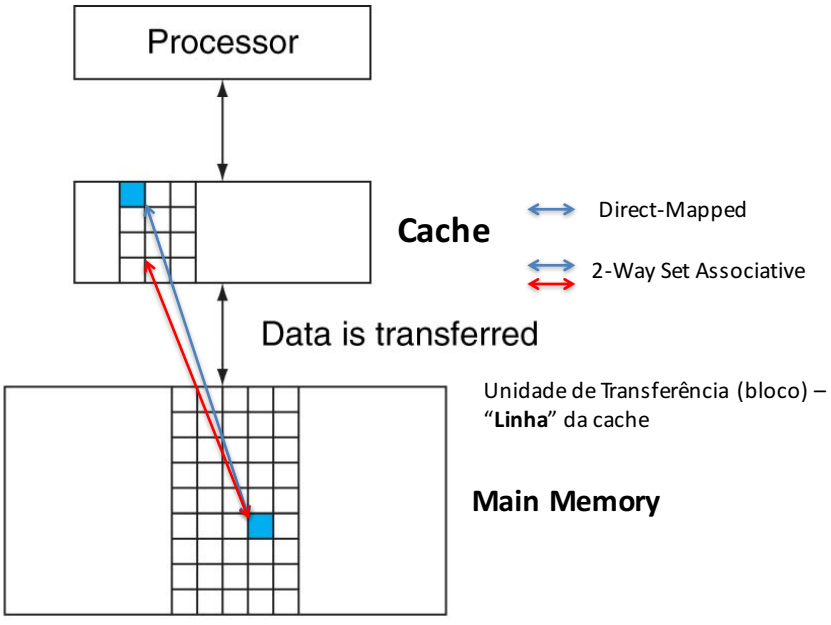
(b) Physical Cache

Figure 4.7 Logical and Physical Caches

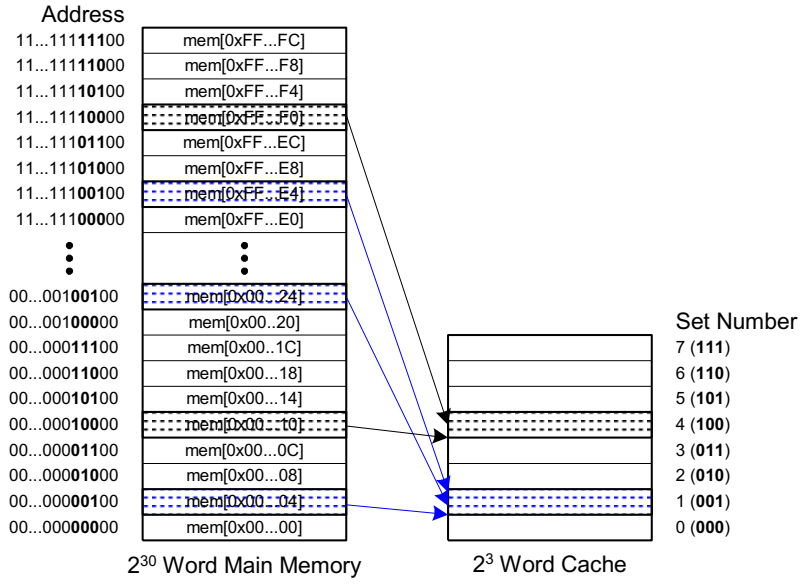
W.Stallings, *Computer Organization and Architecture*

2. Mapping Function

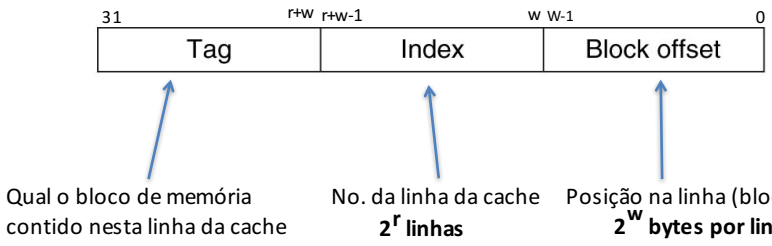
- ***Direct-Mapped*** – cada posição de memória é mapeada numa única posição na cache
- ***N-Way Set Associative*** - cada posição de memória pode ser mapeada em N posições diferentes da cache
- ***Fully Associative*** - qualquer posição de memória pode ser mapeada em qualquer posição da cache



Direct Mapped Cache

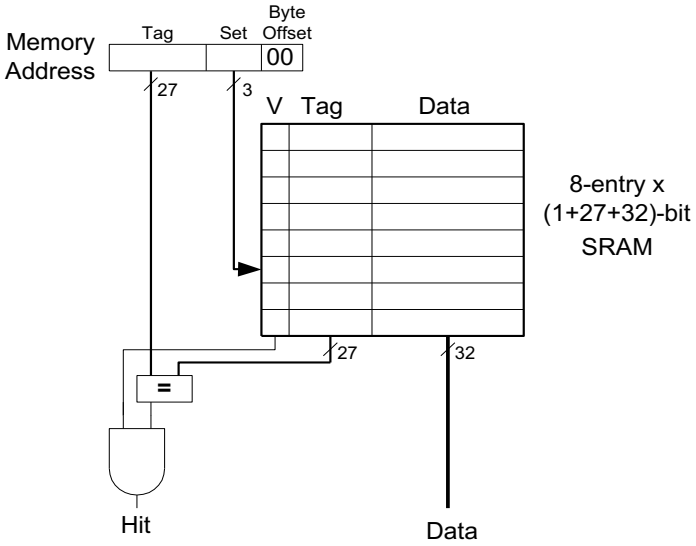


Endereçagem

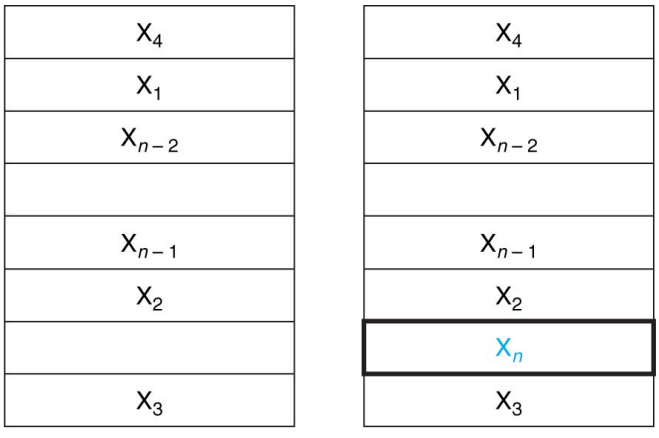


Nota: Nos slides seguintes assume-se que cada linha (bloco) da cache tem a dimensão de uma *word* (i.e. w = 2 para uma arquitetura de 32-bits em que a memória é *byte addressable*)

Direct Mapped Cache Hardware



Preenchimento da Cache: *"Demand driven"*



a. Before the reference to X_n b. After the reference to X_n

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	N		
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	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		

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	N		
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	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		

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

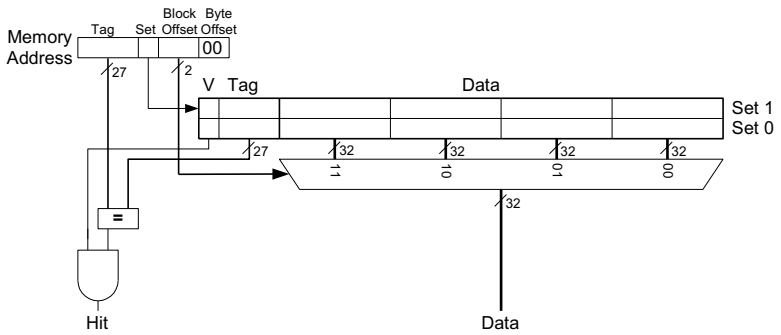
Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	10_{two}	Memory (10010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

f. After handling a miss of address (10010_{two})

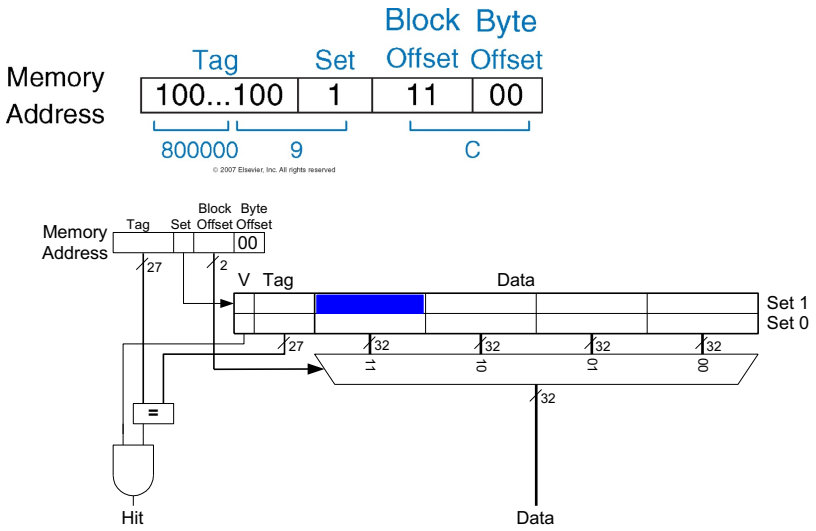
"Conflito"

Localidade Espacial?

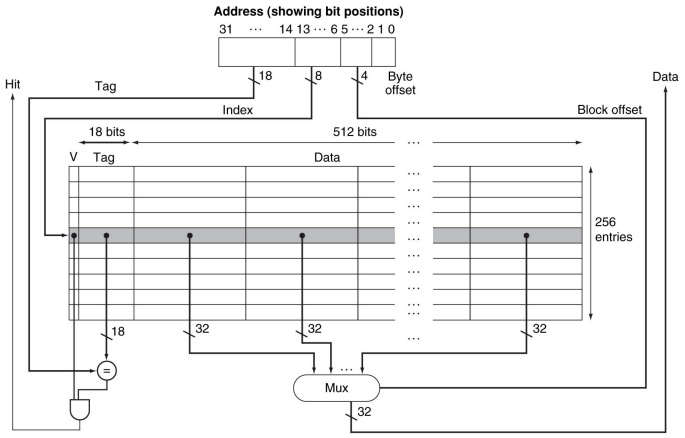
- Aumentar a dimensão dos blocos (linhas):
  - Block size, ***b* = 4 words**
  - Capacidade, *C* = 8 words
  - Direct mapped (1 block per set)
  - Number of blocks, ***B* = 2** (*C*/*b* = 8/4 = 2)



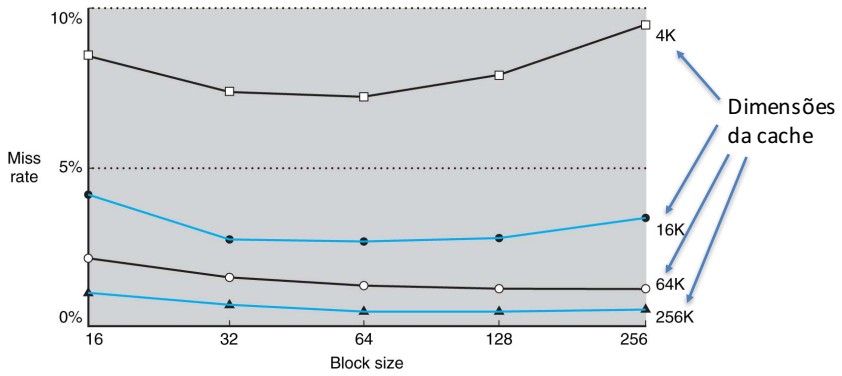
Cache com Blocos > 1 word



Tirar partido da Localidade Espacial das referências: Direct-Mapped Cache com 256 linhas de 16 palavras



Miss rate vs. block (line) size



- Blocos maiores reduzem *misses* devidos a primeiros acessos a um dado (*compulsory misses*)
- Blocos maiores aumentam *misses* devidos a conflitos (*conflict misses*)

5. Line (Block) Size

When a block of data is retrieved and placed in the cache not only the desired word but also some number of adjacent words are retrieved

As the block size increases more useful data are brought into the cache

Two specific effects come into play:

- Larger blocks reduce the number of blocks that fit into a cache
- As a block becomes larger each additional word is farther from the requested word

As the block size increases the hit ratio will at first increase because of the principle of locality

The hit ratio will begin to decrease as the block becomes bigger and the probability of using the newly fetched information becomes less than the probability of reusing the information that has to be replaced

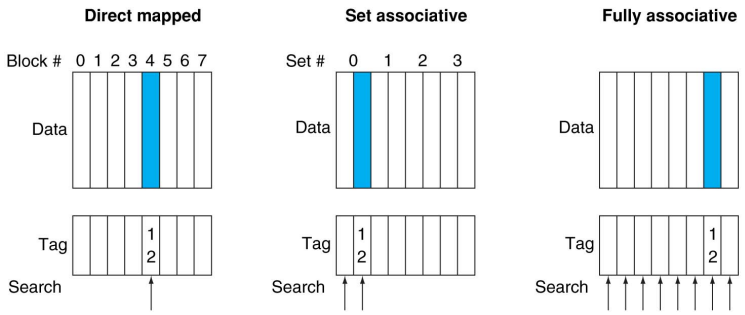
W.Stallings, *Computer Organization and Architecture*

Direct-Mapped Cache Resumo

- Address length = (s + w) bits
- No. de items endereçáveis = 2^{s+w} words ou bytes
- Block size = line size = 2^{w} words ou bytes
- No. de blocos da memória = $2^{s+w}/2^{w} = 2^{s}$
- No. de linhas da cache = m = 2^{r}
- Tag = (s – r) bits

Cache: outros tipos de mapeamento

Exemplo: Cache com 8 linhas (blocos)
Localizações possíveis do bloco 12 de memória

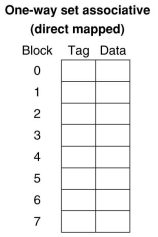


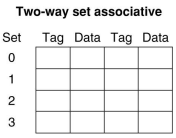
2-way set associative

Diferentes organizações da Cache

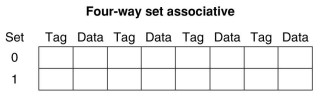
Exemplo: cache de 8 palavras

Cada bloco de memória mapeado numa única linha da cache

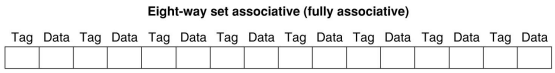




Cada bloco de memória pode ser mapeado numa de duas linhas da cache

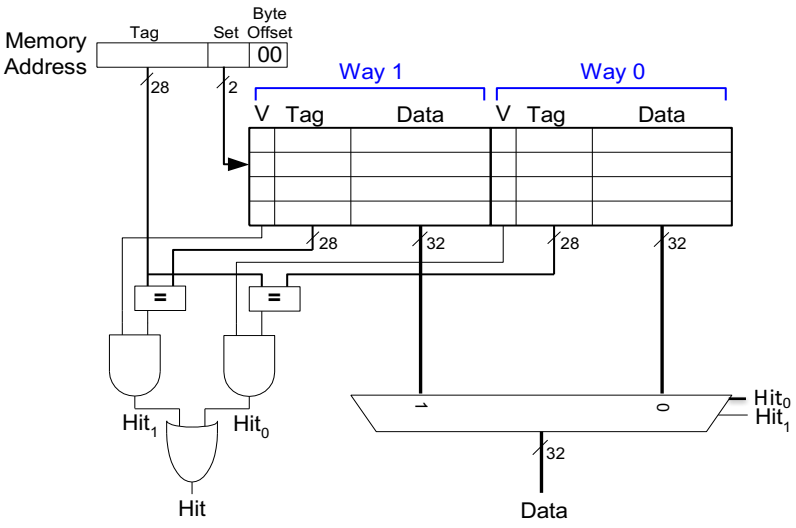


Cada bloco de memória pode ser mapeado numa de quatro linhas da cache

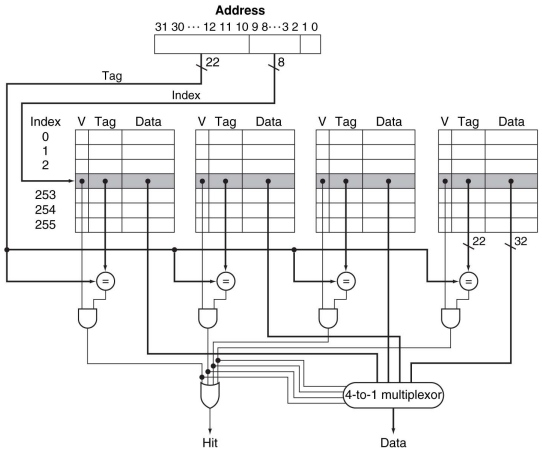


Cada bloco de memória pode ser mapeado em qualquer das linhas da cache

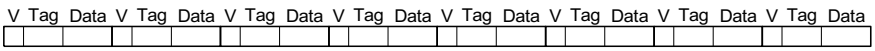
N-Way Set Associative Cache



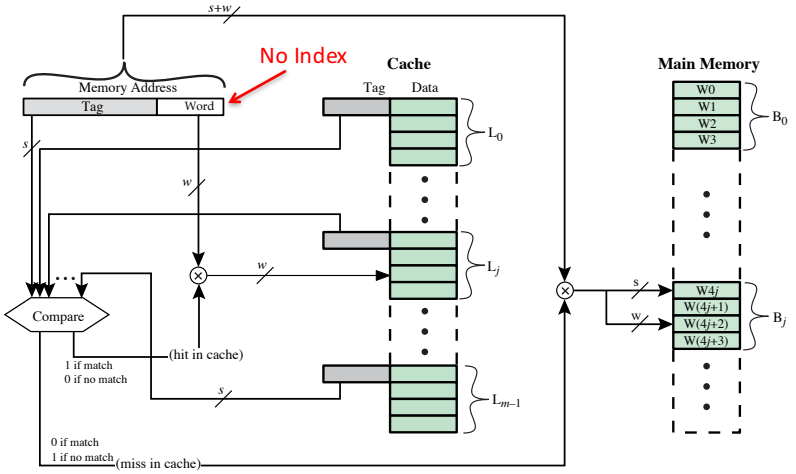
4-way set-associative
(1 word/linha, **4***256 linhas)



Fully Associative Cache



Reduz conflitos
Dispendiosa



(word-addressable memory; $2^w = 4$ words per line)

Figure 4.11 Fully Associative Cache Organization

W.Stallings, *Computer Organization and Architecture*

Resumo: 2^W word Cache

	Direct mapped	K-way Set Associative	Fully Associative
Address length	N-bits	N-bits	N-bits
Block (line) size	2^M words	2^M words	2^M words
Nº de blocos de memória	$2^N / 2^M$	$2^N / 2^M$	$2^N / 2^M$
Nº de linhas da cache	$2^W / 2^M$	$2^W / 2^M$	$2^W / 2^M$
Offset Length	M bits	M bits	M bits
Index length	(W – M) bits	(W - M – $\log_2$ k) bits	0
Tag length	(N – W) bits	(N – W + $\log_2$ k) bits	(N – M) bits

Tipos de *Misses*

- **Compulsory:** primeiro acesso a um dado
- **Capacidade:** cache demasiado pequena para armazenar os dados de interesse (*working* set)
- **Conflito:** dados acedidos mapeiam na mesma localização na cache

Miss penalty: tempo necessário para transferir um time bloco de um nível inferior da hierarquia de memória

Miss Rate: dependência do grau de associatividade

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

Para uma dimensão fixa da cache

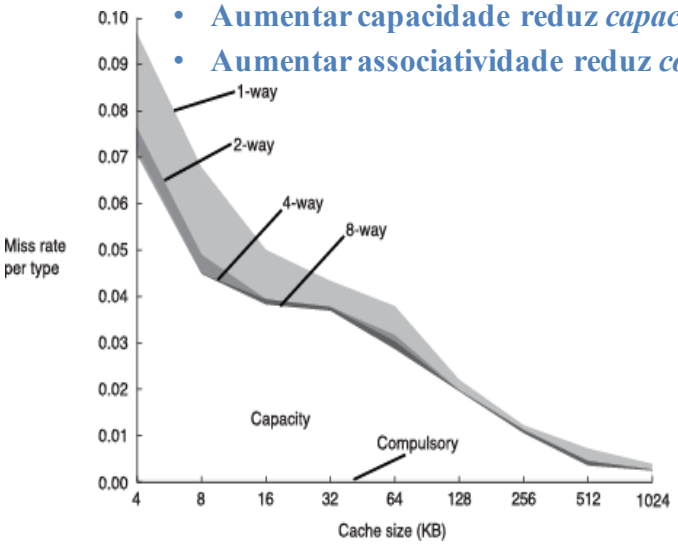
Miss rate depende de:
- Grau de associatividade
- Dimensão do bloco (linha)

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**

Miss Rate Trends

- **Aumentar capacidade reduz *capacity misses***
- **Aumentar associatividade reduz *conflict misses***



Adapted from Patterson & Hennessy, *Computer Architecture: A Quantitative Approach*, 2011

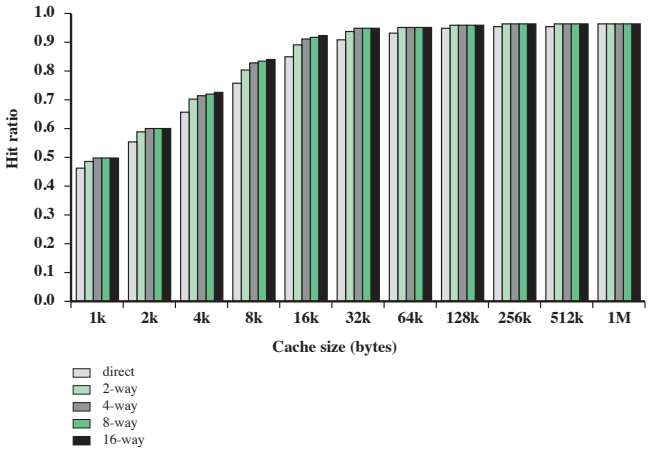


Figure 4.16 Varying Associativity over Cache Size

W.Stallings, *Computer Organization and Architecture*

3. Replacement Algorithms

Quando a cache está cheia e um novo bloco é acedido, um dos blocos na cache tem de ser substituído. Qual?
Direct-mapped cache – o bloco (único) em que o novo bloco mapeia
Associative e Set-Associative caches – necesssário *replacement algorithm* (a implementar em hardware)

Replacement algorithms:

Least recently used (LRU)
Substituir o bloco que não é acedido há mais tempo
Algoritmo mais comum e com melhores resultados

First-in-first-out (FIFO)
Substituir o bloco que está há mais tempo na cache
Fácil de implementar

Least frequently used (LFU)
Substituir o bloco que foi referenciado menos vezes
Pode ser implementado associando um contador a cada linha

4. Write Policy

When a block that is resident in the cache is to be replaced there are two cases to consider:

- If the old block in the cache has not been altered then it may be overwritten with a new block without first writing out the old block
- If at least one write operation has been performed on a word in that line of the cache then main memory must be updated by writing the line of cache out to the block of memory before bringing in the new block

There are two problems to contend with:

- More than one device may have access to main memory
- A more complex problem occurs when multiple processors are attached to the same bus and each processor has its own local cache - if a word is altered in one cache it could conceivably invalidate a word in other caches

W.Stallings, *Computer Organization and Architecture*

Alternativas: ***Write Through*** ou ***Write Back***

Write through

Todos os write (***Stores***) são feitos simultaneamente na cache e na memória central
A solução mais simples
Desvantagem: gera mais tráfico com a memória, o que pode criar um *bottleneck*

Write back

Os write (***Stores***) são feitos apenas na cache
Minimiza as operações de escrita na memória
Desvantagem: blocos da memória inválidos; consequentemente os acessos dos módulos de I/O têm de ser feitos via cache - *potential bottleneck*

6. Numero de caches: **Multilevel Caches**

Os aumentos da escala de integração tornaram possível integrar a cache no mesmo chip do processador (***on-chip cache***)

On-chip cache reduz os acessos ao bus externo e reduz o tempo de acesso
- Quando a instrução ou o dado estão na *on-chip cache*, o acesso ao bus é eliminado
- Durante esse periodo o bus está livre para suportar outras transferências

Two-level cache:
- Internal cache - level 1 (L1)
- External cache - level 2 (L2)

A vantagem de usar uma cache L2 depende das *hit rates* nas caches L1 e L2

Three-level cache (arquiteturas *Multi-Core*)
- Internal cache - level 1 (L1) – exclusiva de cada core – Icache + Dcache
- Internal cache - level 2 (L2) – partilhada pelas várias cores - unificada
- External cache – level 3 (L3)

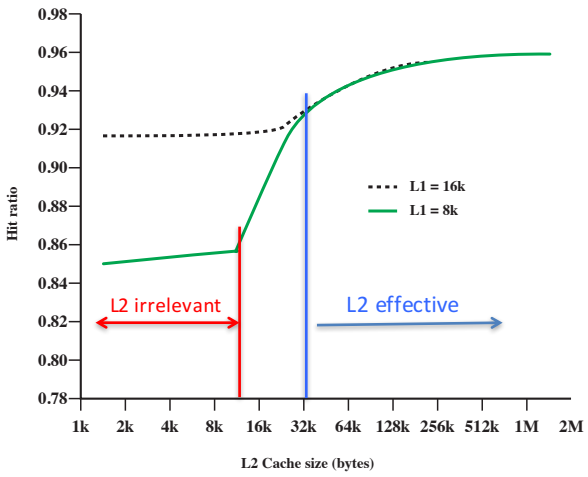


Figure 4.17 Total Hit Ratio (L1 and L2) for 8 Kbyte and 16 Kbyte L1

Unified Versus Split Caches

Split Cache – Instruction Cache + Data Cache

- Ambas existem ao mesmo nível (tipicamente L1 caches)
- Eliminam conflitos de acesso entre a instruction fetch/decode unit e a execution unit
- Importante em processadores pipelined

Vantagens de **Unified Cache**:

- Higher hit rate
  - Balanceamento automático entre instruções e dados na ocupação da cache

Tendência dominante:

- Split Caches ao nível L1 e Unified Caches nos outros níveis (L2 e L3)

Problem	Solution	Processor on which Feature First Appears
External memory slower than the system bus.	Add external cache using faster memory technology.	386
Increased processor speed results in external bus becoming a bottleneck for cache access.	Move external cache on-chip, operating at the same speed as the processor.	486
Internal cache is rather small, due to limited space on chip	Add external L2 cache using faster technology than main memory	486
Contention occurs when both the Instruction Prefetcher and the Execution Unit simultaneously require access to the cache. In that case, the Prefetcher is stalled while the Execution Unit's data access takes place.	Create separate data and instruction caches.	Pentium
Increased processor speed results in external bus becoming a bottleneck for L2 cache access.	Create separate back-side bus that runs at higher speed than the main (front-side) external bus. The BSB is dedicated to the L2 cache.	Pentium Pro
	Move L2 cache on to the processor chip.	Pentium II
Some applications deal with massive databases and must have rapid access to large amounts of data. The on-chip caches are too small.	Add external L3 cache.	Pentium III
	Move L3 cache on-chip.	Pentium 4

Cache nas CPU Intel

48

Core	Cache Type	Cache Size (kB)	Cache Line Size (words)	Associativity	Location	Write Buffer Size (words)
ARM720T	Unified	8	4	4-way	Logical	8
ARM920T	Split	16/16 D/I	8	64-way	Logical	16
ARM926EJ-S	Split	4-128/4-128 D/I	8	4-way	Logical	16
ARM1022E	Split	16/16 D/I	8	64-way	Logical	16
ARM1026EJ-S	Split	4-128/4-128 D/I	8	4-way	Logical	8
Intel StrongARM	Split	16/16 D/I	4	32-way	Logical	32
Intel Xscale	Split	32/32 D/I	8	32-way	Logical	32
ARM1136-JF-S	Split	4-64/4-64 D/I	8	4-way	Physical	32