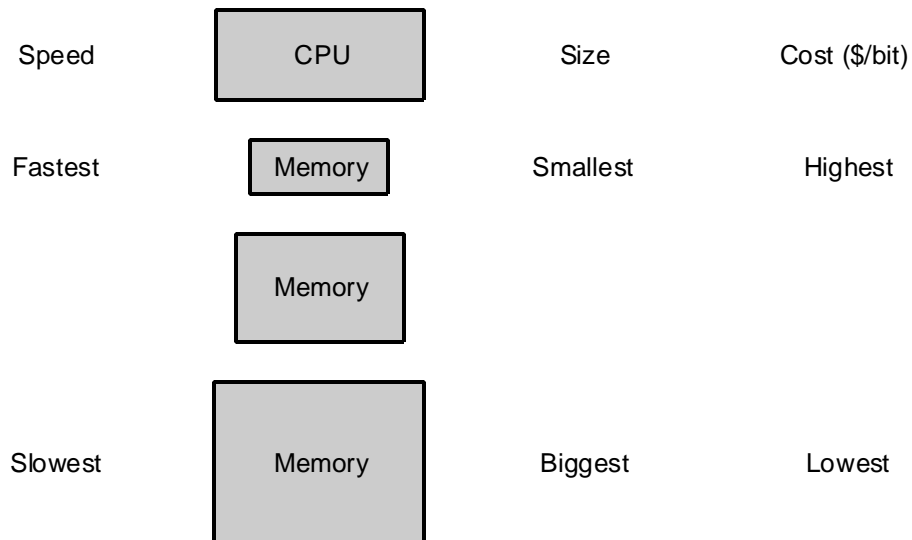


7. Sistemas de Memória – Hierarquia de Memória

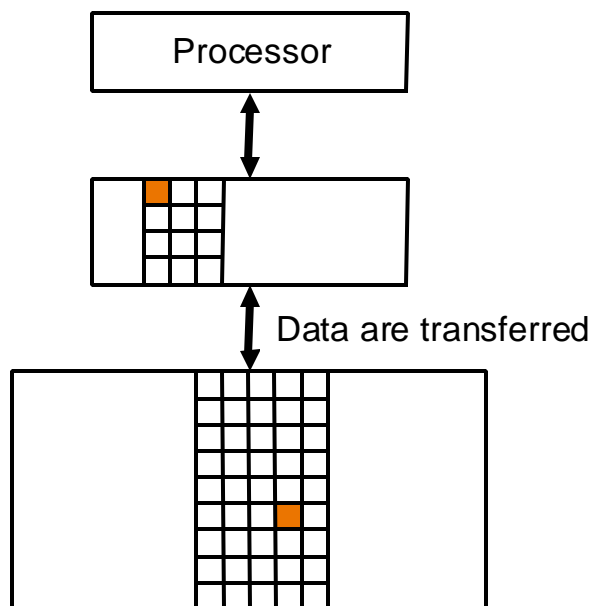
- **Princípio da localidade** → um programa acessa uma porção relativamente pequena do espaço endereçável em um instante qualquer.
 - **Localidade temporal** → Se um item é referenciado, ele tenderá a ser referenciado novamente.
Exemplo → loops (instruções e dados).
 - **Localidade Espacial** → Se um item é referenciado, itens cujos endereços são próximos a este, tenderão a ser referenciados também.
Exemplo → acesso a dados de um array.
- **Princípio da localidade** → Hierarquia de Memória
- **Hierarquia de Memória** → multi-níveis de memória com diferentes tamanhos e velocidades. As mais rápidas são as que tem maior custo de armazenamento por bit, e portanto as menores. Quanto menor a memória, mais perto do processador está localizada.
- **3 principais tecnologias usadas em hierarquia de memória:**

Tecnologia de memória	Tempo de acesso típico (1997)	Custo por Mbyte (1997)
SRAM	5-25 ns	\$100-\$250
DRAM	60-120 ns	\$5-\$10
Disco magnético	10-20 ms	\$0.10-\$0.20

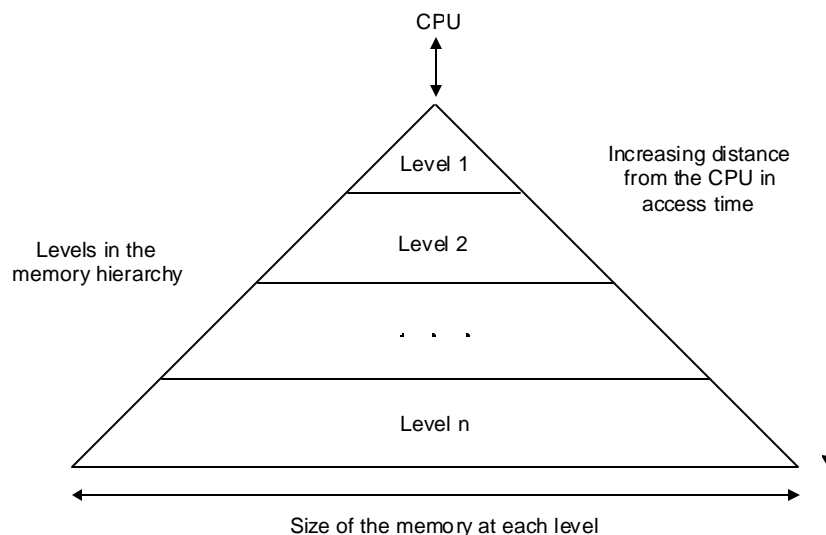
- **Estrutura básica da hierarquia de memória**



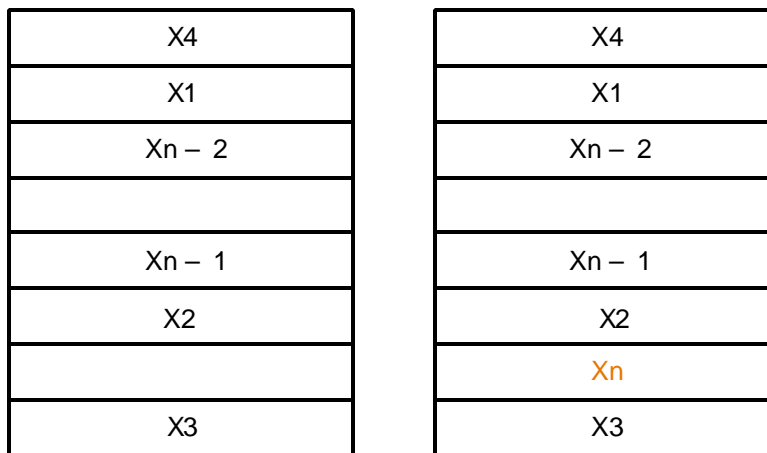
- **Relação entre os dados dos diversos níveis da hierarquia de memória**



- **Bloco** → mínima unidade de informação que pode ou não estar presente em dois níveis de hierarquia de memória.
- **Hit** → se o dado acessado aparece em algum bloco no nível superior.
- **Miss** → se o dado acessado não aparece em algum bloco do nível superior.
- **Hit ratio (hit rate)** → razão de acessos encontrados pelo número total de acessos ao nível superior.
- **Miss ratio (miss rate)** → razão de acessos não encontrados pelo número total de acessos ao nível superior → $\text{miss ratio} = 1 - \text{hit ratio}$.
- **Hit time** → tempo de acesso ao nível superior da hierarquia de memória, que inclui o tempo necessário para saber se no acesso ocorrerá um hit ou um miss.
- **Miss penalty** → tempo para recolocar um bloco no nível superior e enviá-lo ao processador, quando ocorrer um miss. O maior componente do miss penalty é o tempo de acesso ao nível imediatamente inferior da hierarquia de memória.
- **Estrutura da hierarquia de memória**



- **Memória Cache** → nível da hierarquia entre CPU e Memória Principal ou qualquer espaço de armazenamento usado para tirar vantagem da localidade de acesso.
- **Supondo uma cache simples onde um bloco corresponde a uma palavra e o processador acessa a uma palavra.**
- **Supor um bloco X_n que inicialmente não esteja na cache:**



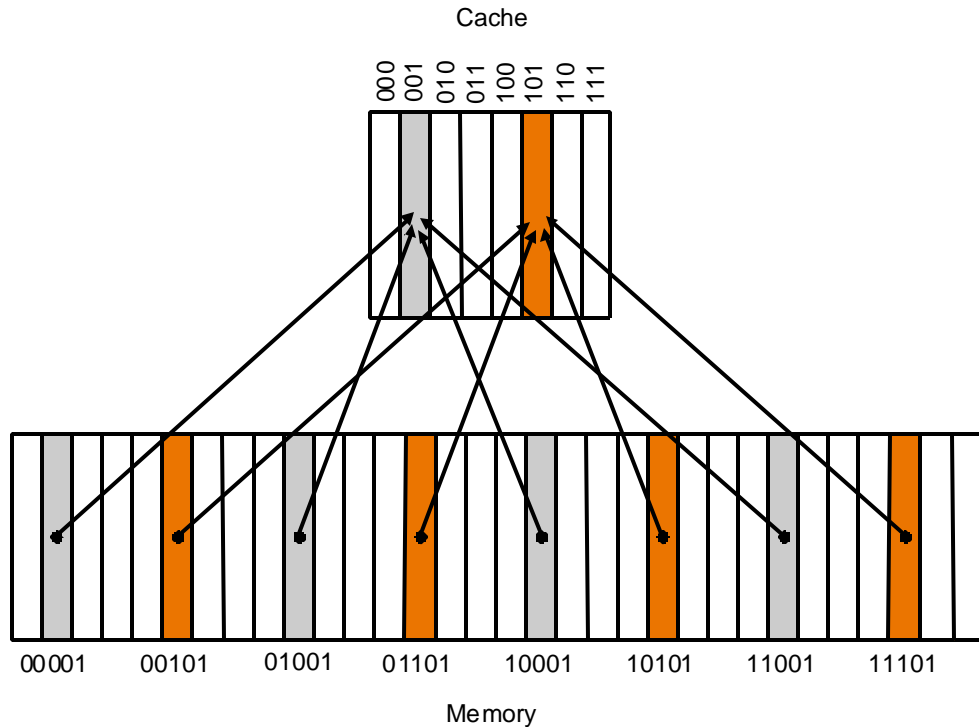
a. Before the reference to X_n

b. After the reference to X_n

- **Duas perguntas no acesso à cache:**
 - **Como saber se o dado está na cache ?**
 - **Se estiver, como encontrá-lo ?**
- **Se cada palavra tiver um lugar n cache → saberemos como encontrá-la.**

- A maneira mais simples de assinalar uma posição da cache para uma palavra de memória é através de seu endereço na memória → direct mapped →

(Endereço do bloco) mod (Número de blocos na cache)



- Como podemos ver cada localização da cache pode receber mais de uma localização da memória → como saber se o dado na cache corresponde ao dado requerido ? → adicionando um conjunto de tags à cache.
- Tags → contém a informação do endereço necessária a identificar se a palavra na cache corresponde à palavra requerida → necessita apenas da parte superior do endereço da palavra.

- **Precisamos saber também se a informação na cache é válida ou não (se ele está vazia, p. ex.) → valid bit**

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

a) estado inicial da cache após power-on

INDEX	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10	Mem(10110)
111	N		

b) Após miss do endereço (10110)

INDEX	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem(11010)
011	N		
100	N		
101	N		
110	Y	10	Mem(10110)
111	N		

c) Após miss do endereço (11010)

INDEX	V	Tag	Data
000	Y	10	Mem(10000)
001	N		
010	Y	11	Mem(11010)
011	N		
100	N		
101	N		
110	Y	10	Mem(10110)
111	N		

d) Após de miss do endereço (10000)

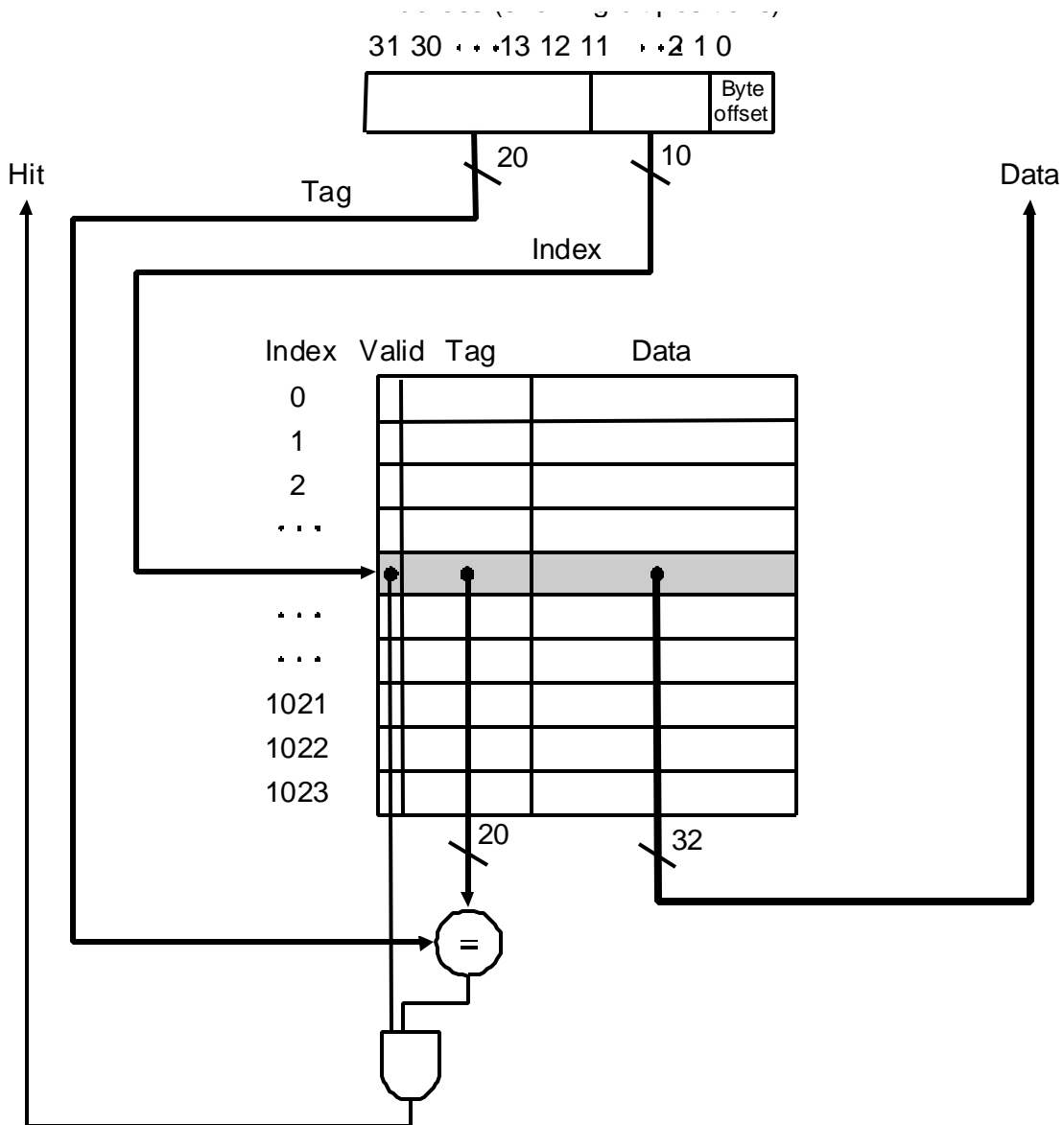
INDEX	V	Tag	Data
000	Y	10	Mem(10000)
001	N		
010	Y	11	Mem(11010)
011	Y	00	Mem(00011)
100	N		
101	N		
110	Y	10	Mem(10110)
111	N		

e) Após miss do endereço (00011)

INDEX	V	Tag	Data
000	Y	10	Mem(10000)
001	N		
010	Y	10	Mem(10010)
011	Y	00	Mem(00011)
100	N		
101	N		
110	Y	10	Mem(10110)
111	N		

f) Após de miss do endereço (10010)

- **Acessando a cache**



- **Endereço é dividido em :**

- **Cache index** → para seleccionar o bloco.
- **Campo do tag** → para comparar com o tag da cache
- **No MIPS os dois últimos bits referem-se ao byte dentro da palavra (4 bytes → 2 bits para endereçar).**

- **Número de bits necessários para uma cache é função do tamanho da cache e do tamanho do endereço (dados + tags)**
- **Número de bits de uma cache**
 - **Endereço de 32 bits, cache com mapeamento direto de 2^n words com blocos de uma palavra (4 bytes) \rightarrow tag de $32 - (n + 2)$.**
 - **2 bits usados para offset do byte e n para o índice. O número total de bits da cache $\rightarrow 2^n \times (32 + (32 - n - 2) + 1) = 2^n \times (63 - n)$.**
- **Exemplo: Quantos bits são necessários para uma cache com mapeamento direto com 64KB de dados e bloco de uma palavra, assumindo 32-bit de endereço?**

Solução:

64KB \rightarrow 16K palavras $\rightarrow 2^{14}$ palavras $\rightarrow 2^{14}$ blocos

Cada bloco tem 32 bits de dados mais o tag ($32 - 14 - 2 = 16$) mais o bit de validade

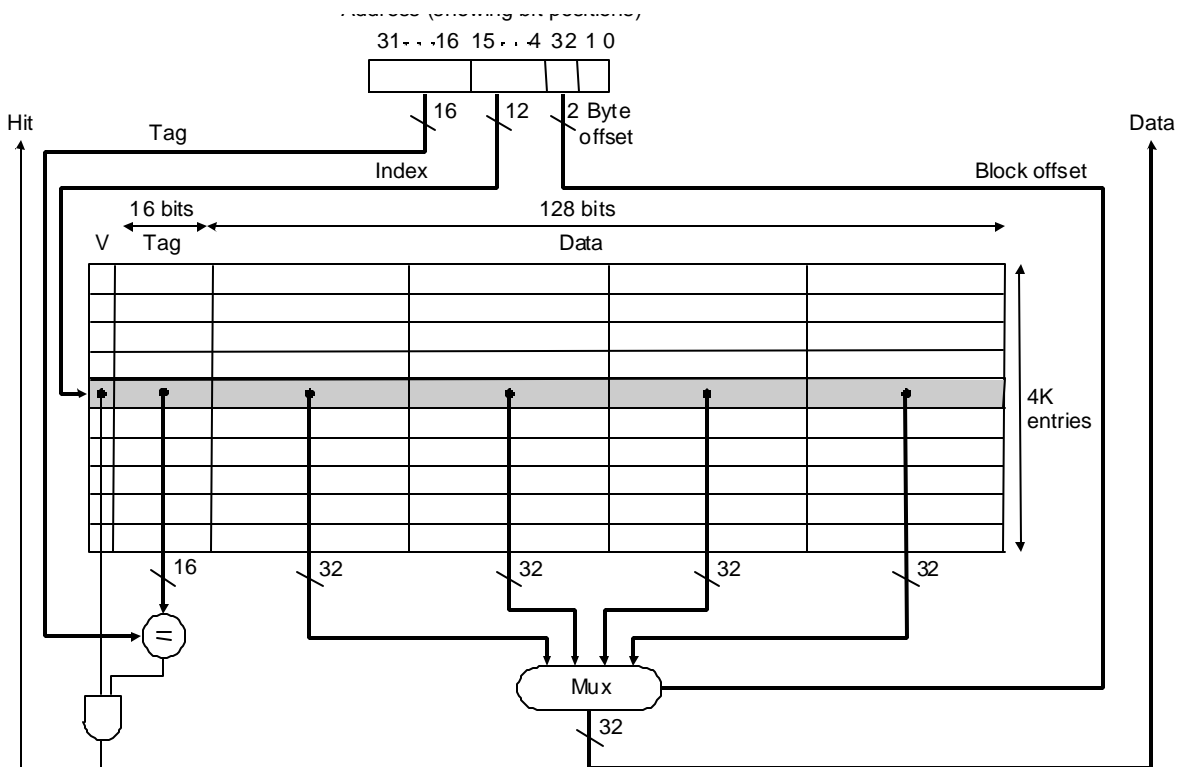
Total de bits da cache $2^{14} \times (32 + 16 + 1) = 784 \text{ Kbits} = 98 \text{ KB}$

Para esta cache, temos um overhead de 1.5, devido aos tag e aos bits de validade.

- **Tratamento de Cache Misses**
- **Quando a unidade de controle detecta um miss, ele busca o dado da memória. Se detecta um hit, continua o processamento como se nada tivesse acontecido.**
- **Mudança do datapath (cap. 5 e cap. 6) → substituir as memórias por caches e alterar o controle para quando ocorrer miss.**
- **Alteração do controle → atrasar (stall semelhante ao stall do pipeline → diferença que para todas as unidades do pipeline) a CPU, congelando o conteúdo de todos os registradores. Um controlador separado trata o miss, lendo o dado da memória.**
- **Etapas de um cache miss de uma instrução**
 - **Enviar o PC original (PC corrente – 4) para a memória**
 - **Fazer a leitura da memória e esperar o conteúdo**
 - **Escrever na cache o dado vindo da memória, escrevendo os bits mais significativos do endereço (da ULA) no campo de tag e setando o bit de validade.**
 - **Reiniciar a execução da instrução.**
- **Etapas de um cache miss de dados → stall o processador até a memória enviar o dado.**

- Escrita → na escrita de uma instrução de store → o dado tem que ser escrito na cache → valores diferentes entre cache e memória principal → inconsistência → escrever também na memória principal → write-through.
- Performance com write-through → gcc tem 13% de instruções de store. Na DECStation 3100 a CPI para store é 1.2 e gasta 10 ciclos a cada escrita → nova CPI = $1.2 + 13\% \times 10 = 2.5$ → reduz o desempenho por um fator maior que 2 → solução possível → write buffer.
- Outro esquema de atualização da memória → write back → a memória só é atualizada quando o bloco da cache que sofreu modificação for substituído por outro.
- Write miss → dado escrito na cache pelo processador → não há leitura da memória principal → atualizar tag.

- **Localidade Espacial → blocos de cache com mais de uma palavra.**
- **cache block = block address mod número de cache blocks**
- **block address = word address / número de words no bloco**



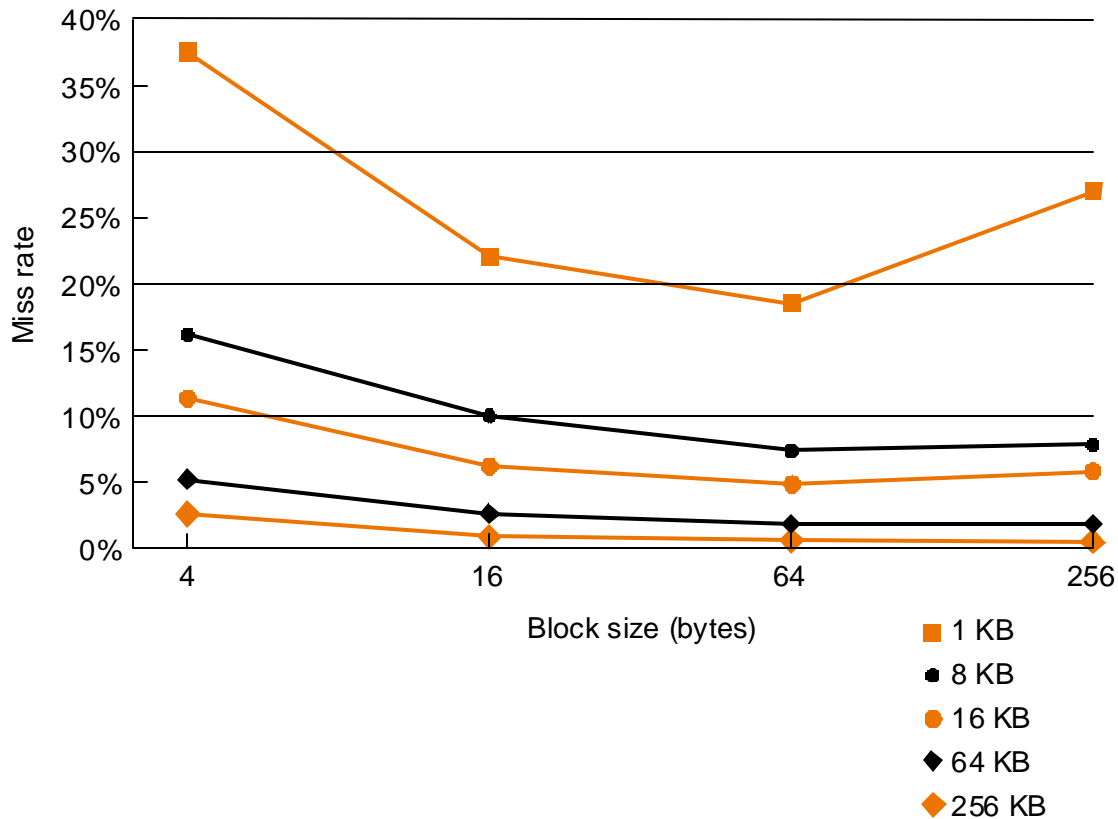
- **Exemplo de mapeamento de um endereço em uma cache de multiword block → Cache com 64 blocos de tamanho de 16 bytes. Que bloco tem o endereço 1200 ?**

Solução:

**endereço do bloco = endereço do byte / bytes por bloco =
 $1200/16 = 75$**

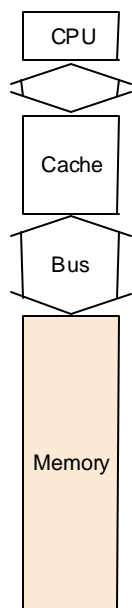
**bloco = (endereço do bloco) mod (número de blocos da cache)
 $= 75 \text{ mod } 64 = 11$**

- **Comparação de miss rate entre caches de tamanhos diferentes com blocos de tamanhos diferentes blocos.**

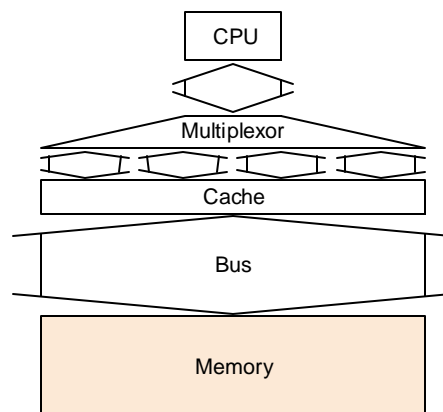


- **Sistema de memória para suportar caches**
 - **Memória principal → DRAMs → latência no fetch da primeira palavra de memória.**
 - **Redução do miss penalty → aumentar o bandwidth memória → cache → permite a utilização de grandes blocos (um dos problema de grande blocos é o tempo de transferência para a cache).**

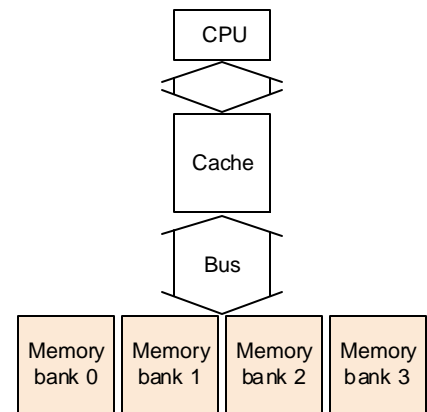
- **Diferentes organizações de memória**
- **Supor:**
 - **1 ciclo de clock para enviar o endereço**
 - **15 ciclos de clcok para cada acesso à DRAM**
 - **1 ciclo de clock para enviar uma palavra de dados.**
- **Para um cache de bloco de 4 palavras (de 32 bits) e um banco de DRAM de 1 palavra**
- **Três sistemas de memória**



a. One-word-wide memory organization



b. Wide memory organization



c. Interleaved memory organization

- a) → memória de largura de uma palavra → acessos feitos seqüencialmente → miss penalty = $1 + 4 \times 15 + 4 \times 1 = 65$ ciclos. O número de bytes transferidos por ciclo de clock em um miss = $(4 \times 4) / 65 = 0.25$
 - b) → aumento de bandwidth pelo aumento da largura de memória e barramento → acessos paralelos de palavras nos blocos → (memória de largura de 2 palavras) → miss penalty = $1 + 2 \times 15 + 2 \times 1 = 33$ ciclos. O número de bytes transferidos por ciclo de clock em um miss = $(4 \times 4) / 33 = 0.48$. (largura de 4 palavras → 17 ciclos e bandwidth de 0.96). Custo no barramento (largura) e multiplexador.
 - c) → interleaving → aumento de bandwidth pelo aumento da memória (bancos de memória). 4 bancos de memória → 15 ciclos para 4 palavras (um de cada banco) → miss penalty = $1 + 1 \times 15 + 4 \times 1 = 20$ ciclos. O número de bytes transferidos por ciclo de clock em um miss = $(4 \times 4) / 20 = 0.80$.
- Medida e melhoria de desempenho de Cache
 - Melhoria de desempenho → 2 técnicas:
 - Redução da probabilidade de de dois blocos diferentes serem alocados na mesma linha de cache.
 - Redução do miss pela adição de mais um nível de cache na hierarquia (multilevel caching).

- **CPU time = (CPU execution clock cycles + Memory-stall clock cycles) X clock cycle time**
- **Memory-stall clock cycles = Read-stall cycles + Write-stall cycles**
- **Read-stall cycles = (Reads/Program) X Read miss rate X Read miss penalty**
- **Write-stall cycles = ((Writes/Program) X Write miss rate X Write miss penalty) + Write buffer stalls**
- **Combinando leitura com escrita:**

Memory-stall clock cycles = (Memory accesses/Program) X Miss rate X Miss penalty

ou

Memory-stall clock cycles = (Instructions/Program) X (Misses/Instructions) X Miss penalty

- **Exemplo:**

Assumir uma instruction cache miss para o gcc de 2% e o data cache miss de 4 %. Se uma máquina tem um CPI de 2 sem nenhum stall de memória e o miss penalty é de 40 ciclos para todos os misses, determinar quanto mais rápido seria esta máquina se não existisse miss de cache.

Solução:

Número de miss cycles para instruções em termos do número de instruções → Instruction miss cycle = $I \times 2\% \times 40 = 0.80 \times I$

A frequência de todos os loads e stores do gcc = 36% → Número de memory miss cycles para referência de dados → Data miss cycles = $I \times 36\% \times 4\% \times 40 = 0.56 \times I$

Número total de memory-stall cycles = $0.80I + 0.56I = 1.36I$

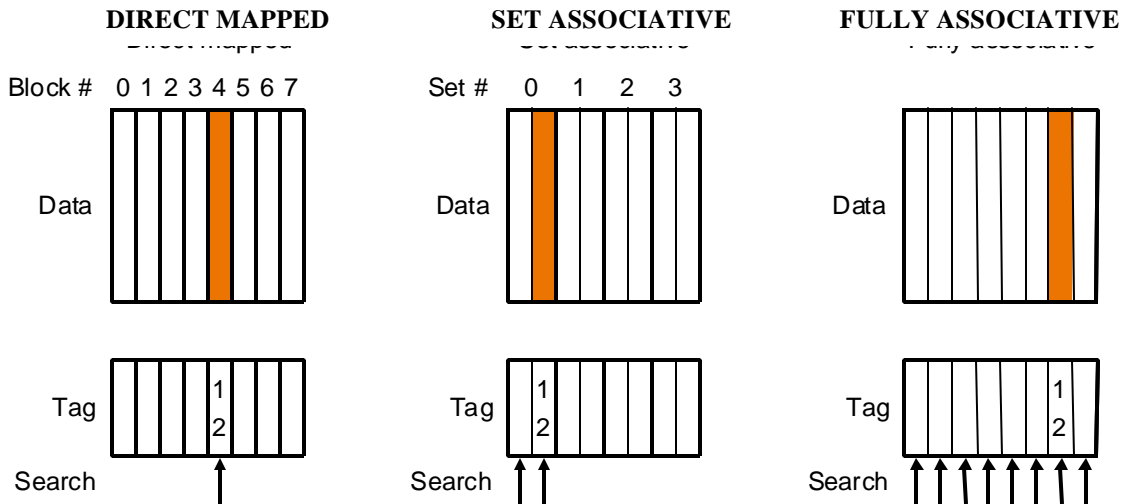
$CPI = 2 + 1.36 = 3.36$

$(CPU \text{ time with stalls} / CPU \text{ time with perfect cache}) = (I \times CPI_{\text{stall}} \times \text{Clock cycle}) / (I \times CPI_{\text{perfect}} \times \text{Clock cycle}) = CPI_{\text{stall}} / CPI_{\text{perfect}} = 3.36 / 2 = 1.68$ → o desempenho com cache perfeita é melhor 1.68.

- **O que acontece se o processador é mais rápido mais o sistema de memória permanece o mesmo ?**
- **Diminuição do CPI:**
 - **No exemplo acima suponha o CPI diminuindo de 2 para 1 sem alterar o clock rate → CPI com cache miss = $1 + 1.36 = 2.36$ → o desempenho com cache perfeita seria melhor $2.36 / 1 = 2.36$ vezes mais rápido.**
 - **O tempo gasto no memory stalls cresceria de $1.36 / 3.36 = 41\%$ para $1.36 / 2.36 = 58\%$**

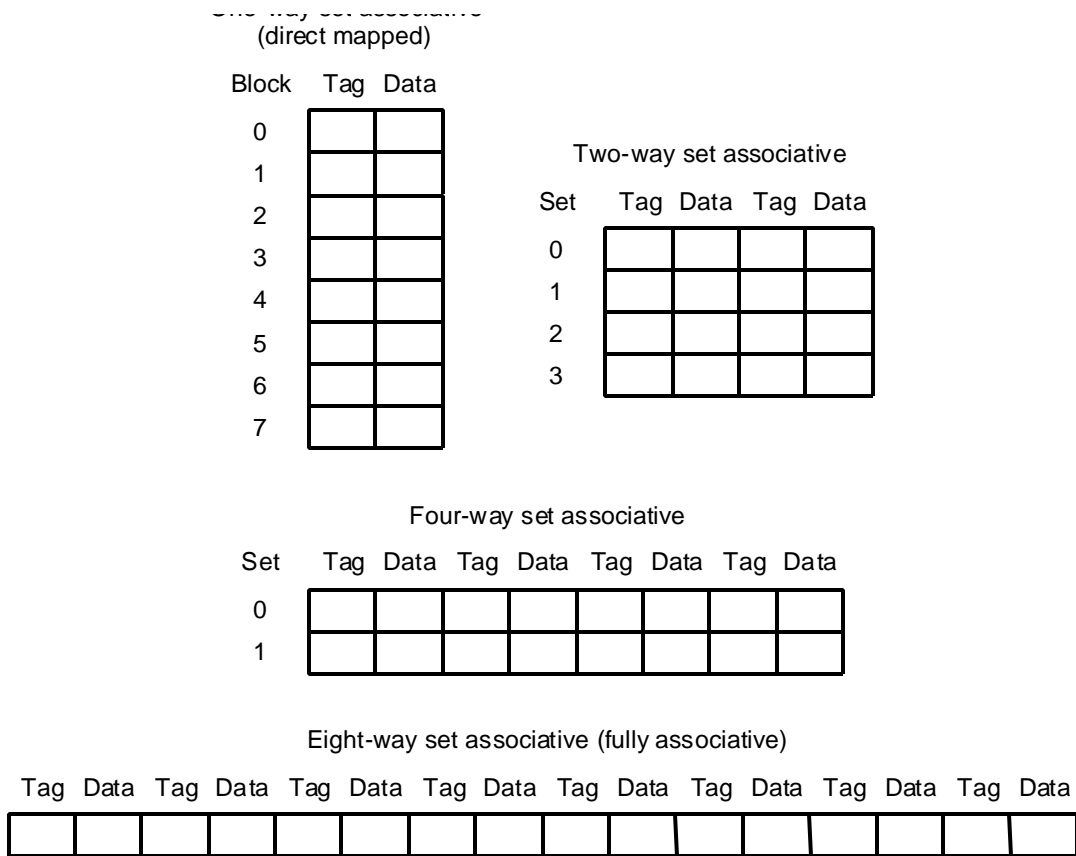
- **Aumentando o clock rate**
 - **Supor que a máquina do exemplo anterior tenha seu clock rate dobrado.**
 - **O novo miss penalty = $40 \times 2 = 80$ ciclos (sistema de memória inalterado)**
 - **Total miss cycles por instrução = $(2\% \times 80) + 36\% \times (4\% \times 80) = 2.75$**
 - **CPI com cache miss = $2 + 2.75 = 4.75$**
 - **Comparando com a máquina do exemplo anterior (CPI de 3.36)**
 - **(Performance com fast clock / Performance com slow clock) = (Execution tme com slow clock / Execution time com fast clock) = $((IC \times CPI \times Clock\ rate) / ((IC \times CPI \times (Clock\ rate)/2) = 3.36 / 4.75 \times 0.5 = 1.41$**
 - **Máquina com o dobro do clock é 1.41 vezes mais rápida.**

- **Redução de Cache Misses pela Alocação Flexível de Blocos**



- **Direct Mapped** → memory block position = (Block number) mod (Number of cache blocks) → one-way set associative.
- **Set Associative** → o set que contém o memory block = (Block number) mod (Number of sets in the cache) → 2-way ou 4-way set associative (para cache de 8 blocos).
- **Fully associative** → 8-way set associative (para cache de 8 blocos).

- **Mapeamento direto, set associativo e totalmente associativo**



- **Exemplo**

Existem 3 pequenas caches de 4 blocos de uma palavra. Uma é fully associative, a segunda two-way set associative e a terceira é direct mapped. Encontre o número de misses para cada uma, dado a seguinte seqüência de endereços de blocos: 0,8,0,6,8.

Solução:

- Direct mapped

Block address	Cache block
0	$0 \bmod 4 = 0$
6	$6 \bmod 4 = 2$
8	$8 \bmod 4 = 0$

Endereço do bloco de memória acessado	Hit or miss	Conteúdo do bloco da cache após a referência			
		0	1	2	3
0	miss	mem[0]			
8	miss	mem[8]			
0	miss	mem[0]			
6	miss	mem[0]		mem[6]	
8	miss	mem[8]		mem[6]	

- → 5 misses

- 2-way set associative

Block address	Cache set
0	$0 \bmod 2 = 0$
6	$6 \bmod 2 = 0$
8	$8 \bmod 2 = 0$

Endereço do bloco de memória acessado	Hit or miss	Conteúdo do bloco da cache após a referência			
		Set 0	Set 0	Set 1	Set 1
0	miss	mem[0]			
8	miss	mem[0]	mem[8]		
0	hit	mem[0]	mem[8]		
6	miss	mem[0]	mem[6]		
8	miss	mem[8]	mem[6]		

- → pior caso 4 misses

- Fully associative

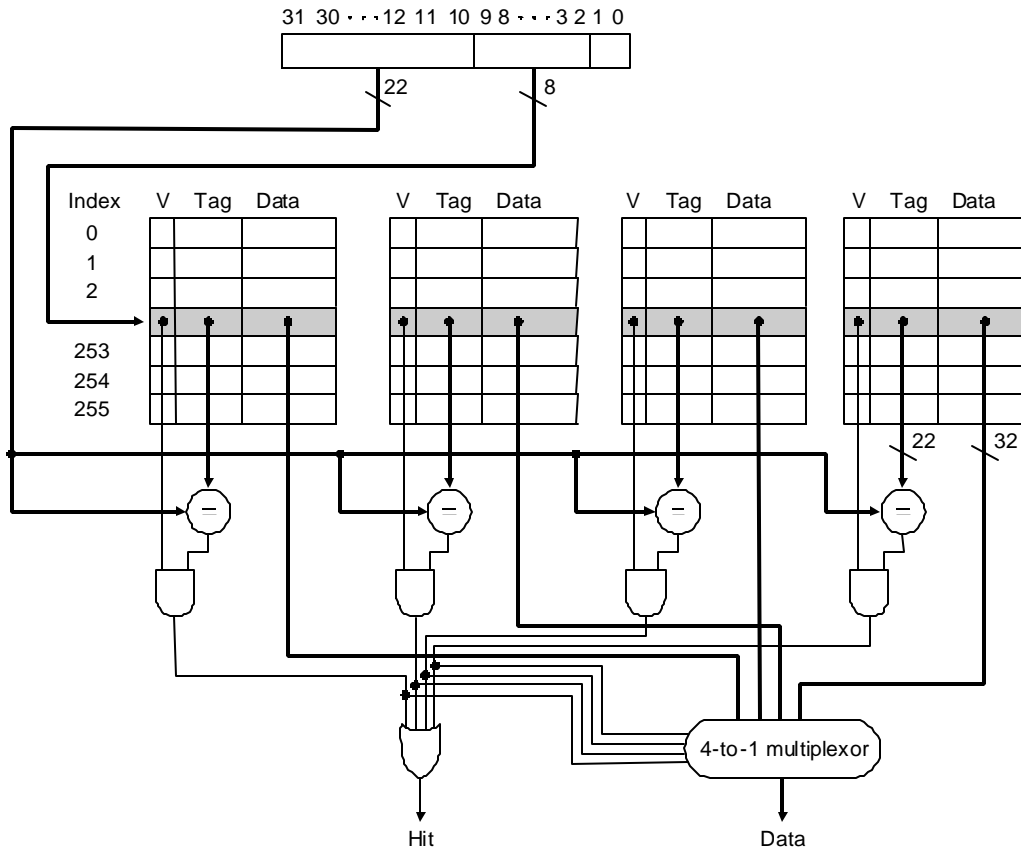
Endereço do bloco de memória acessado	Hit or miss	Conteúdo do bloco da cache após a referência			
		bloco 0	bloco 1	bloco 2	bloco 3
0	miss	mem[0]			
8	miss	mem[0]	mem[8]		
0	hit	mem[0]	mem[8]		
6	miss	mem[0]	mem[8]	mem[6]	
8	hit	mem[0]	mem[8]	mem[6]	

- → 3 misses

- **Localização de um bloco na cache**



Campos de endereço no set-associativo e mapeamento direto



- **Tamanho do Tag X Set Associatividade**

Exemplo:

Assumindo cache de 4K blocos (de uma palavra) e 32-bit de endereço, encontre o número total de sets e o número total de tag bits para caches direct mapped, 2-way set associative, 4-way set associative e fully associative.

Solução:

Direct Mapped → 4K blocos → 4K sets → 12 bits de index → $32-12=20$ bits de tag → total de bits de tag = $20 \times 4K = 80K$ bits.

2-way set associative → $4K/2 = 2K$ sets → 11 bits de index → $32 - 11 = 21$ bits de tag → total de bits de tag = $21 \times 2 \times 2K = 84K$ bits.

4-way set associative → $4K/4 = 1K$ sets → 10 bits de index → $32 - 10 = 22$ bits de tag → total de bits de tag = $22 \times 4 \times 1K = 88K$ bits.

Fully associative → 1 set de 4K blocos → tag de 32 bits → total de bits de tag = $32 \times 4K \times 1 = 128K$ bits.

- **Política de Substituição de Blocos na Cache** → mais usada LRU – Least Recently Used → o bloco que estiver mais tempo na cache sem ser usado.
- **Redução do Miss Penalty pelo uso de Multilevel Cache**
 - Segundo nível de cache é uma SRAM off-chip, que é acessada quando ocorre um miss na cache primária.
 - Exemplo

Suponha que um processador tenha como base uma CPI de 1.0, assumindo todos os hits na cahe primária e clock rate de 500 MHz. Assuma que a memória principal tenha tempo de acesso de 200 ns, incluindo todo tratamento de miss. Suponha ainda que, o miss rate por instrução na cache primária é de 5%. Quanto mais rápido será a máquina se adicionarmos uma cache secundária de 20 ns de tempo de acesso tanto para hit como para miss e que tenha uma tamanho suficiente para reduzir o miss da memória principal para 2%.

Solução:

Miss penalty da memória principal é = $200 \text{ ns} / 2 \text{ ns} = 100$ ciclos de clock

CPI efetiva para um nível de cache → Total CPI = Base CPI + Memory-stall cycles por instrução

Para máquina de 1 nível → $1 + 5\% \times 100 = 6$

Para máquina de 2 níveis:

Miss penalty de um acesso ao segundo nível = $20 \text{ ns} / 2 \text{ ns} = 10$ ciclos de clock.

Total CPI = 1 + Primary-stalls por instrução + Secondary -stalls por instrução = $1 + 5\% \times 10 + 2\% \times 100 = 3.5$.

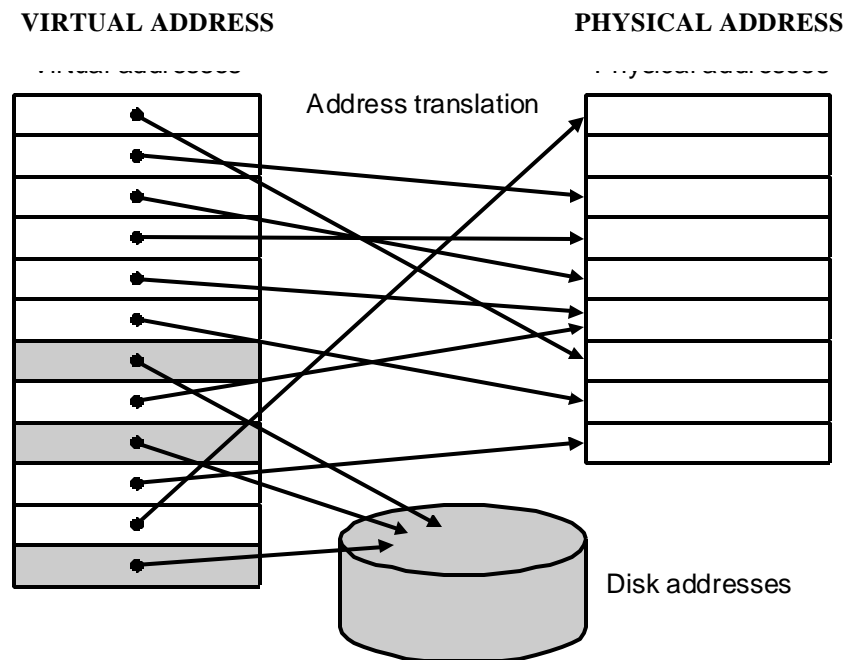
A máquina com 2 níveis de cache é $6.0 / 3.5 = 1.7$ mais rápida.

- **Memória Virtual**

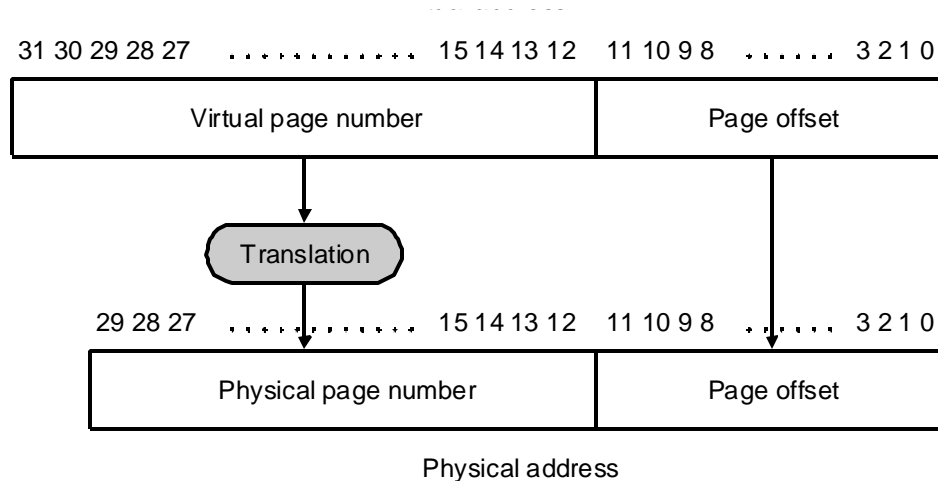
- **Motivação**

- **Permitir compartilhamento seguro e eficiente da memória entre vários programas.**
 - **Melhorar o desempenho de programas nas pequenas e limitadas memórias principais.**

- Espaço endereçável → localizações de memória acessíveis a apenas um programa.
- Overlays → módulos de um programa que são carregados na memória quando acionados.
- Memória física → memória principal
- Page → bloco de memória
- Page fault → miss page
- Endereço virtual → gerado por um processo, que será mapeado em um endereço físico.



- **Endereço virtual → composto do número da página virtual + offset na página.**



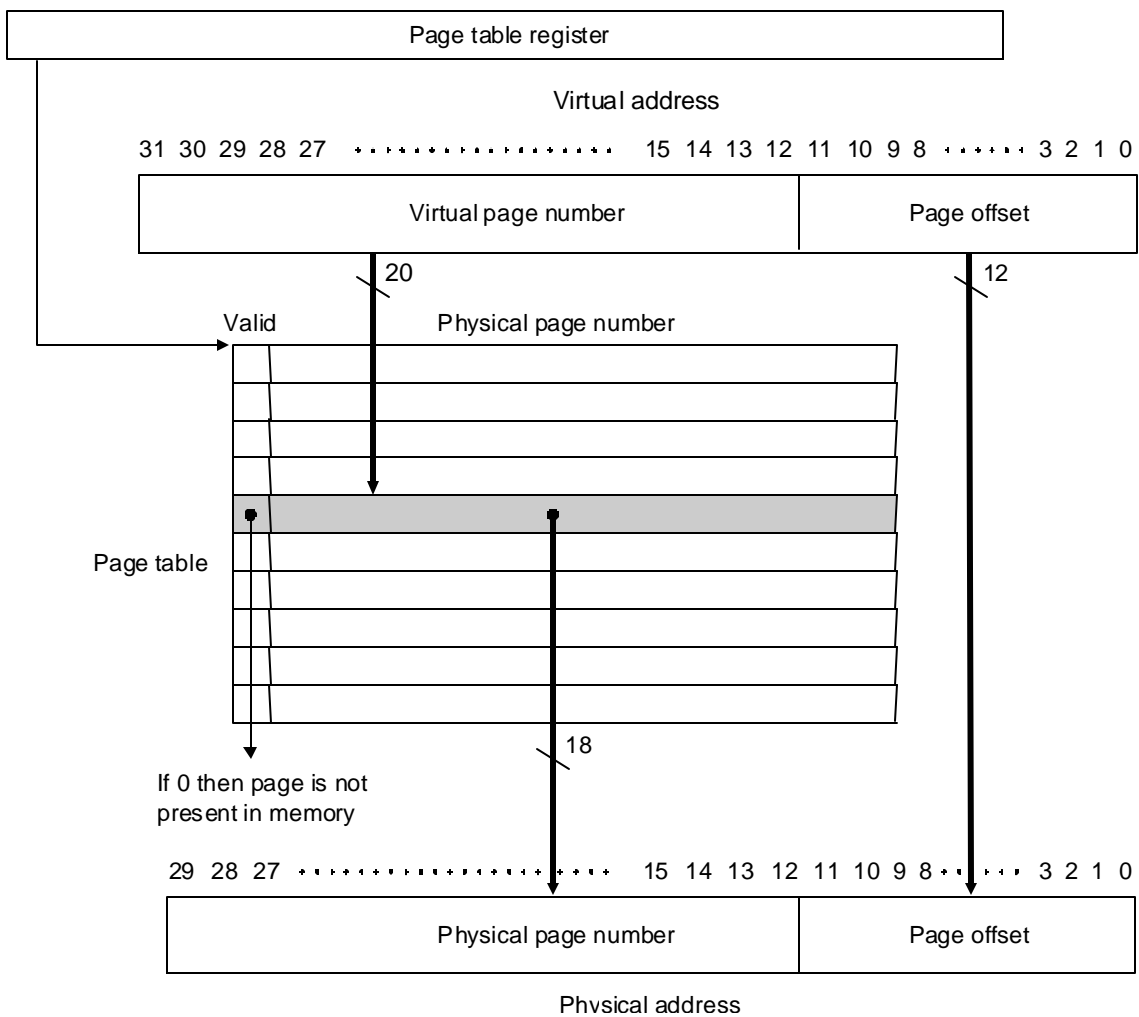
- **Premicias do projeto de memória virtual:**
 - **As páginas devem ser grandes o suficiente para amortizar o tempo (grande) de acesso em disco, quando ocorre o page fault. Atualmente, tamanhos típicos variam de 4KB a 16KB**
 - **Políticas de alocação para diminuir pages fault (fully associative).**
 - **Utilização de write-back.**

- **Memória virtual paginada**

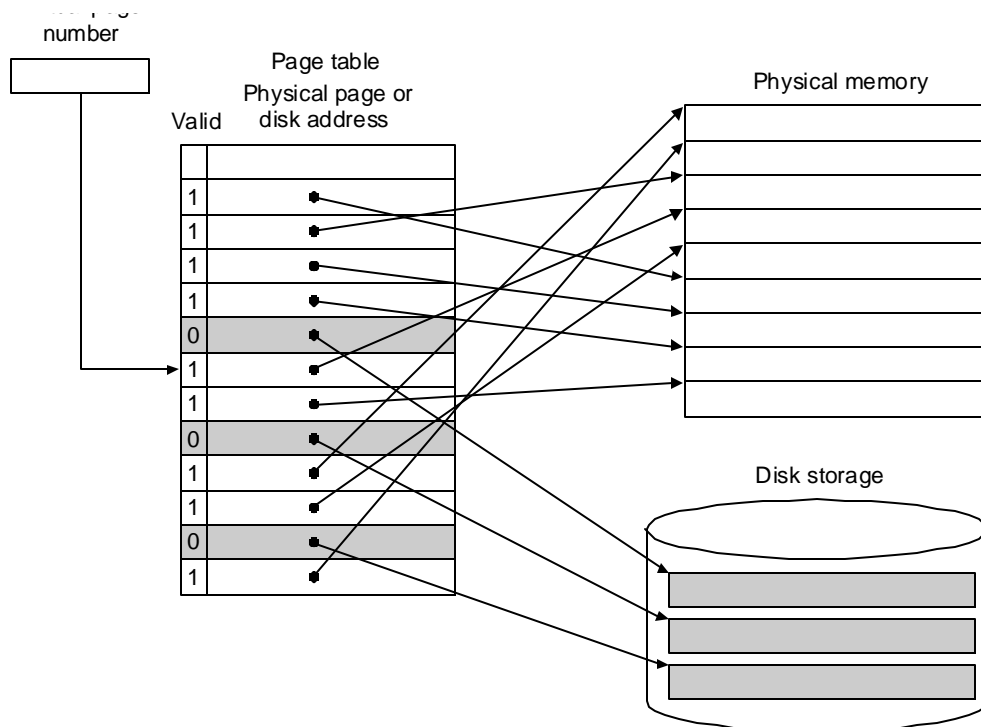
- **Page table**

- **Valid bit**

- **Page Table Register**

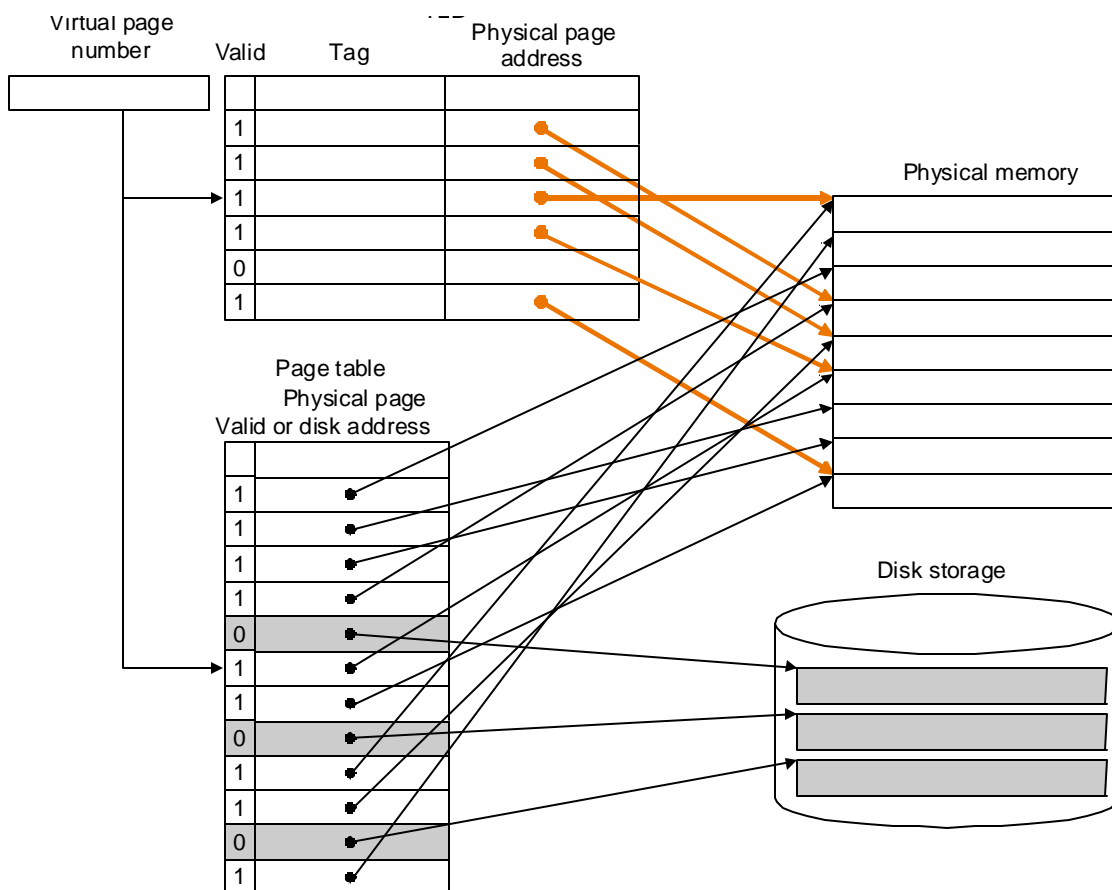


- **Page Fault**

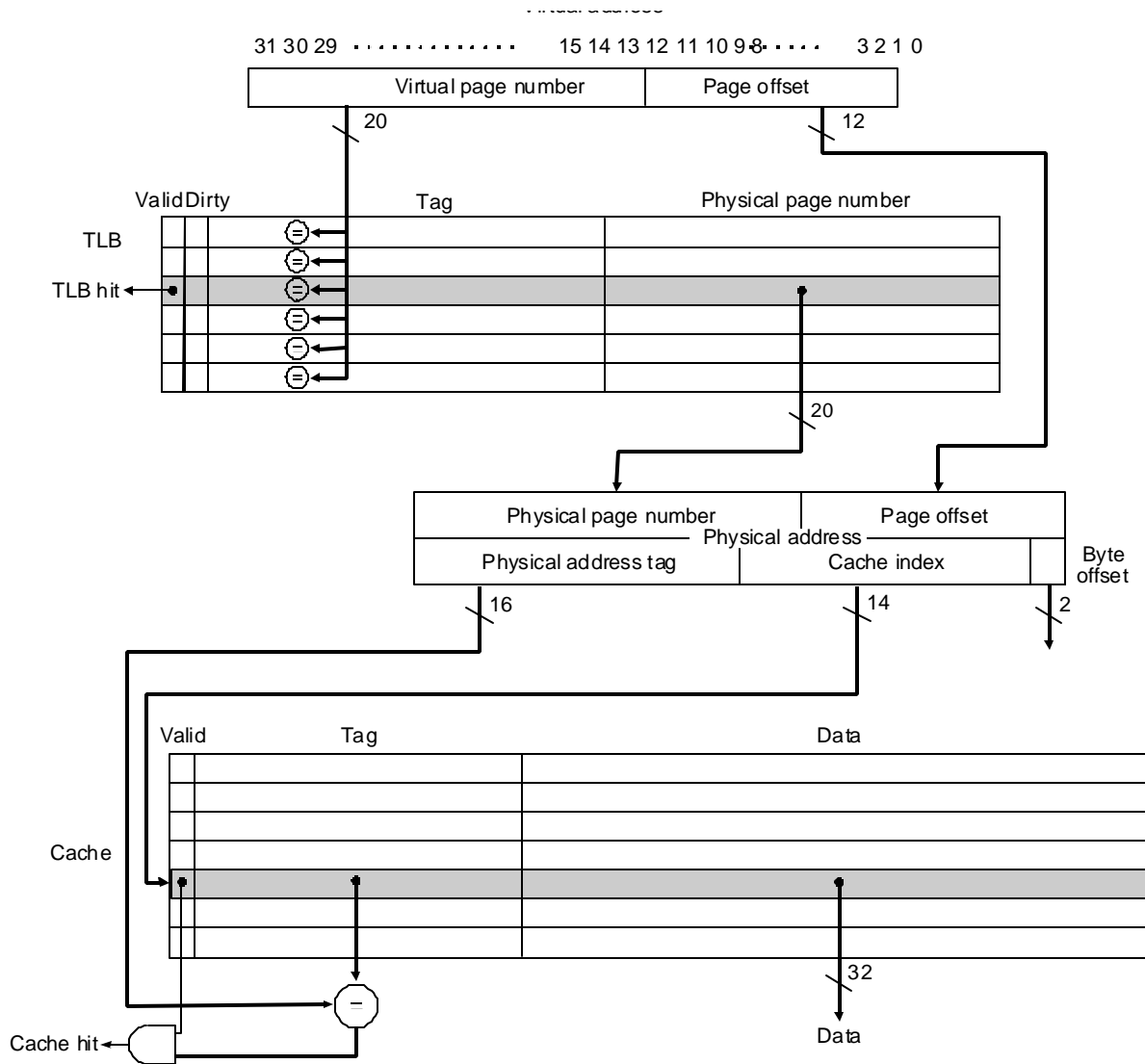


- **Translation-lookaside Buffer – TLB**

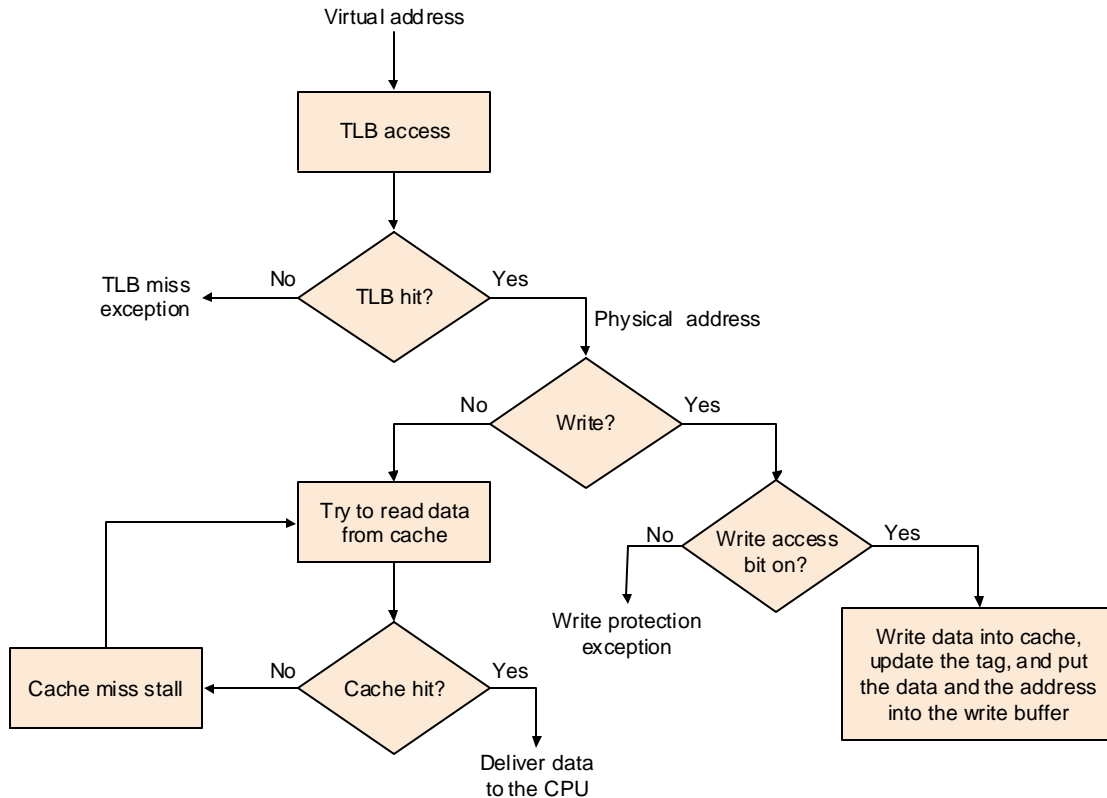
- **Pelo principio da localidade, uma página acessada provavelmente será acessada novamente.**
- **TLB é uma cache que guarda os últimos endereços das páginas acessadas.**



- **Memória virtual + TLBs + Caches**



- **Leitura ou write trough na TLB e Cache**



- **Operação em uma Hierarquia de Memória**

Exemplo

Supondo o esquema Memória Virtual + TLB + Cache visto anteriormente, uma referência de memória pode encontrar 3 diferentes tipos de misses: cache miss, TLB miss e page fault. Considere todas as possibilidades que possa ocorrer e em que condições.

Solução:

CACHE	TLB	VIRTUAL MEMORY	POSSÍVEL ? SE SIM, EM QUE CIRCUNSTÂNCIA
MISS	HIT	HIT	Sim, embora a page table nunca é checkada qdo tem hit na TLB.
HIT	MISS	HIT	Sim, TLB miss, mas hit na page table, após a consulta à page table, o dado é encontrado na cache
MISS	MISS	HIT	Sim, TLB miss, mas hit na page table, após a consulta à page table, miss na cache
MISS	MISS	MISS	TLB miss, miss na page table, miss na cache
MISS	HIT	MISS	Não, não pode ter translação na TLB se a página não está presente na memória.
HIT	HIT	MISS	Não, não pode ter translação na TLB se a página não está presente na memória
HIT	MISS	MISS	Não, dados não podem estar na cache e não na memória.