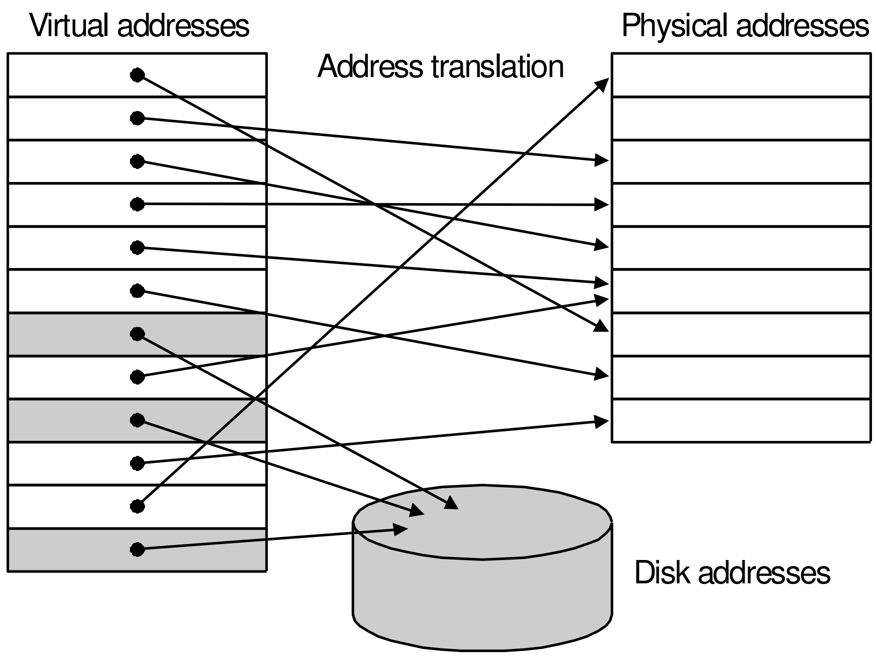
Chapter Seven
Sistemas de Memória

parte B
Memória Virtual

Virtual Memory

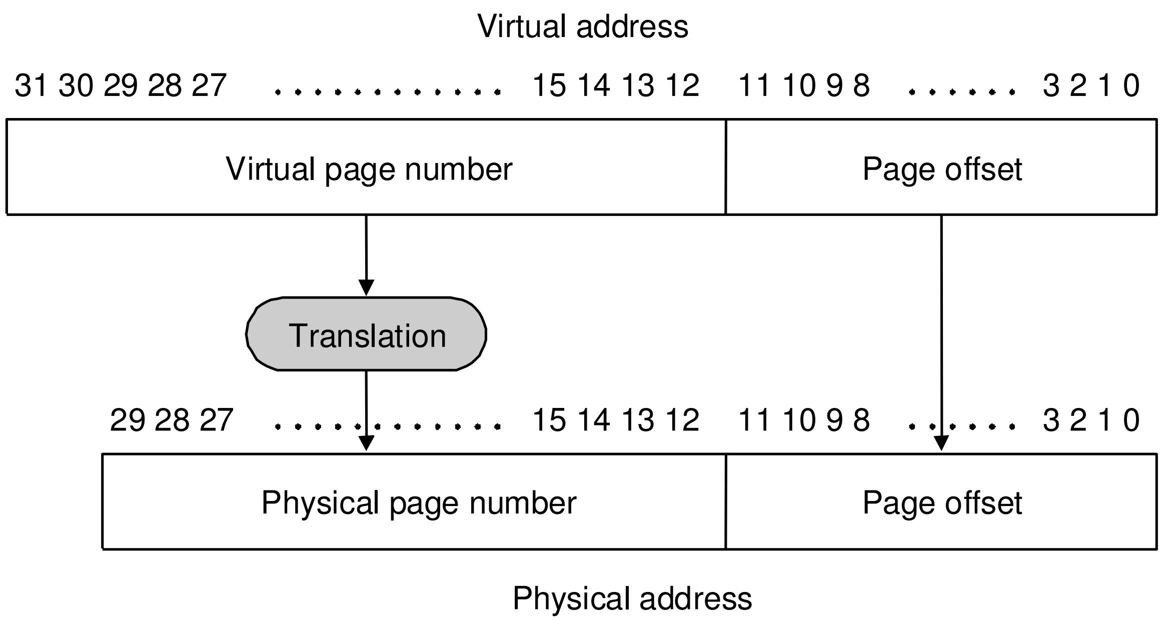
- **Main memory can act as a cache for the secondary storage (disk)**



- **Advantages:**
  - **illusion of having more physical memory (programa independente da configuração do hardware)**
  - **program relocation**
  - **protection (address space)**

Pages: virtual memory blocks

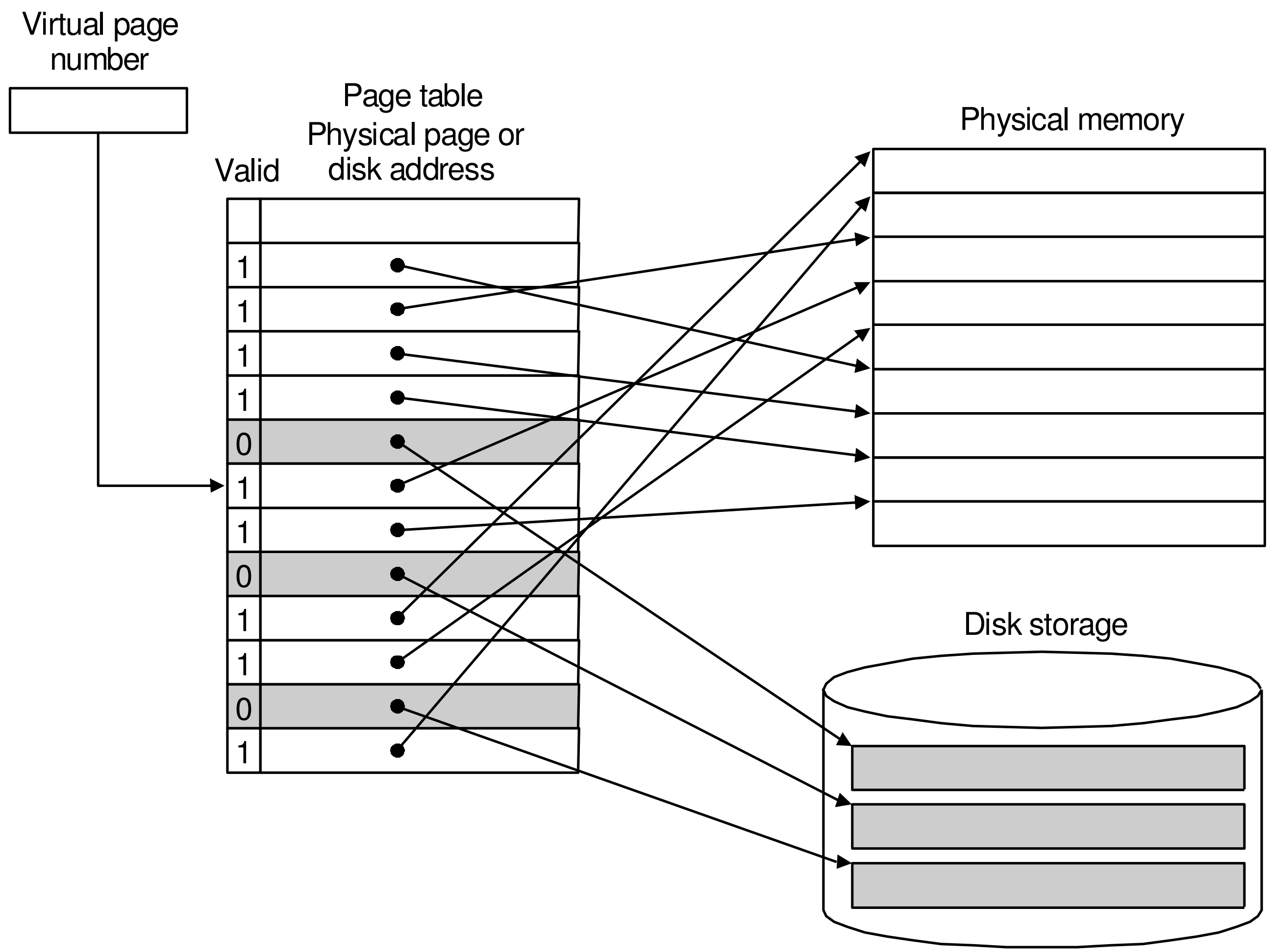
- **Page faults: the data is not in memory, retrieve it from disk**
  - **huge miss penalty, thus pages should be fairly large (e.g., 4KB)**
  - **reducing page faults is important (LRU is worth the price)**
  - **can handle the faults in software instead of hardware**
  - **using write-through is too expensive so we use write-back**



12 bits: 4 KB
VPN: 20 bits
1M page
4 GB

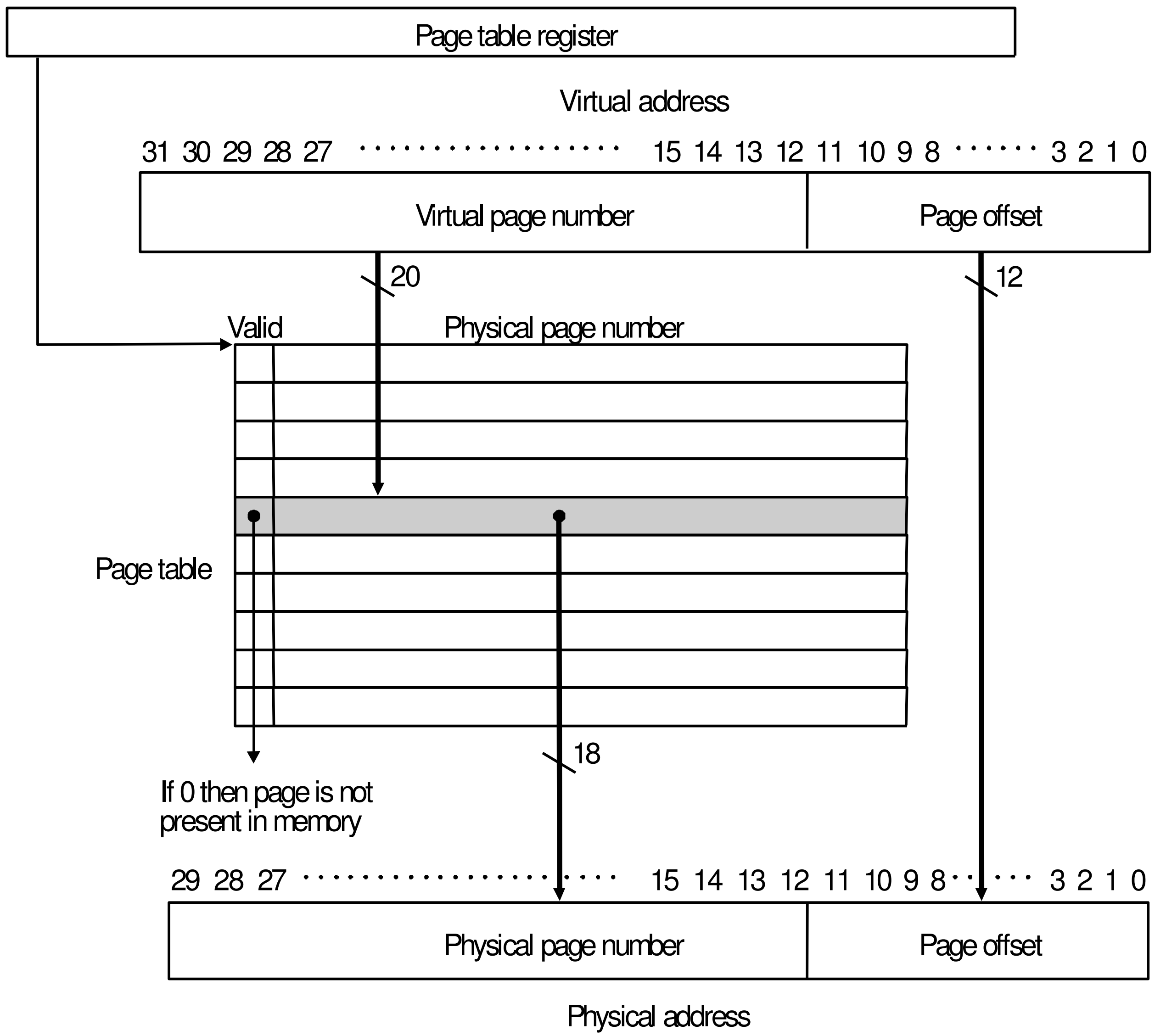
PPN: 18 bits
256 K page
1 GB

Page Tables



Virtual page number
Page table
Physical page or disk address
Valid
1
1
1
1
0
1
1
0
1
1
0
1
Physical memory
Disk storage

Page Tables



- uma PT por processo
- estado:
  - PT
  - PC
  - registradores

Política de substituição e tamanho da PT

- Se page fault (bit válido= 0)
  – sistema operacional executa a carga da página
- Para minimizar page faults, política de substituição mais usada: LRU

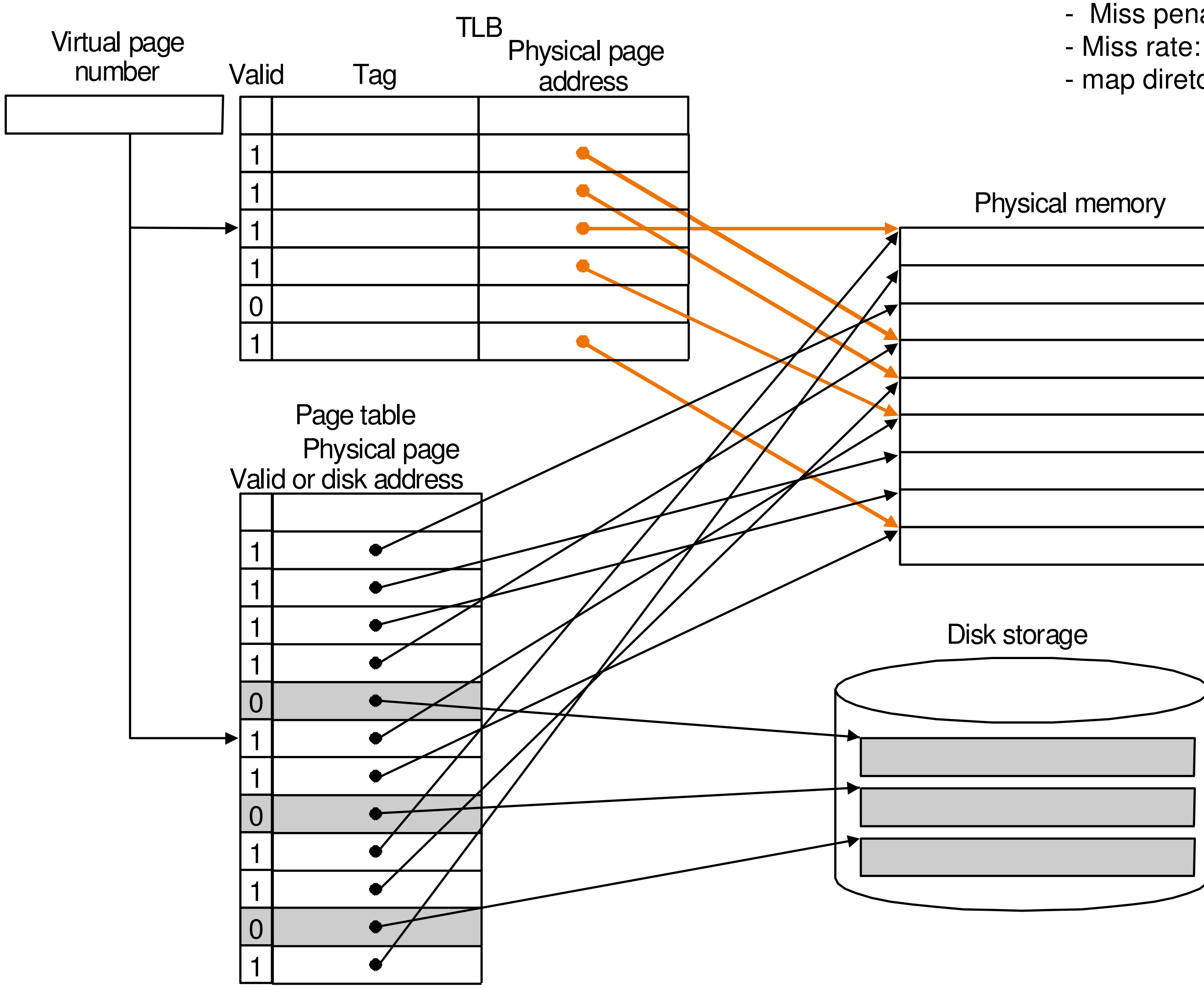
- Tamanho da PT (p/ end 32 bits, pag de 4KB, 4B / linha da PT)
  – número de linhas: $2^{32} / 2^{12} = 2^{20}$
  – tamanho da PT = 4 MB
  – 1 PT por programa ativo !!
  – para reduzir área dedicada para PT: registradores de limite superior e inferior

- PT também são paginados

TLB: translation lookaside buffer

Typical values

- TLB size: 32 - 4,096 entries
- Block size: 1 - 2 page table entries
- Hit time: 0.5 - 1 clock cycle
- Miss penalty: 10 - 30 clock cycle
- Miss rate: 0.01% - 1%
- map direto ou fully associativo

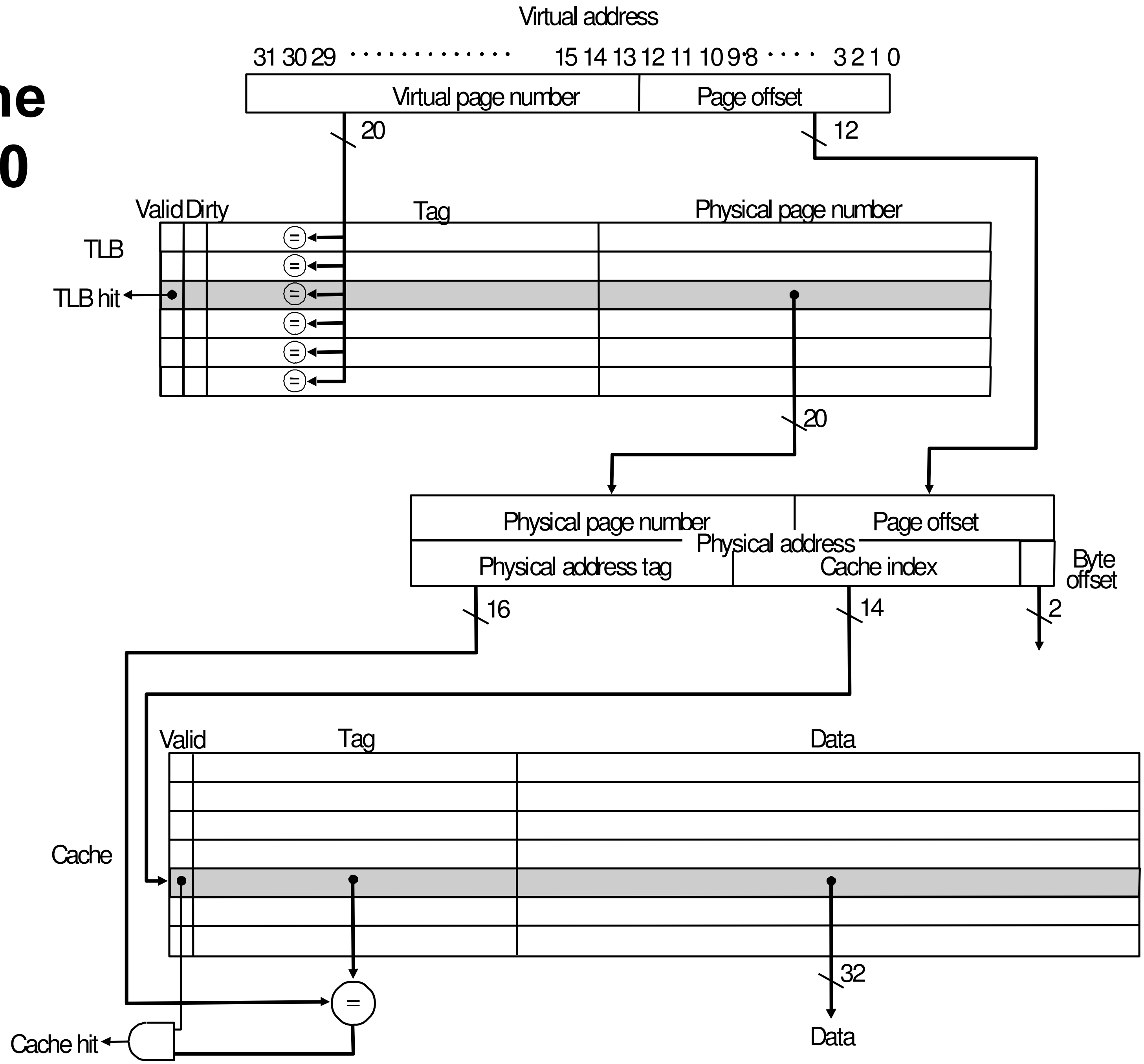


TLBs and cache DEC 3100

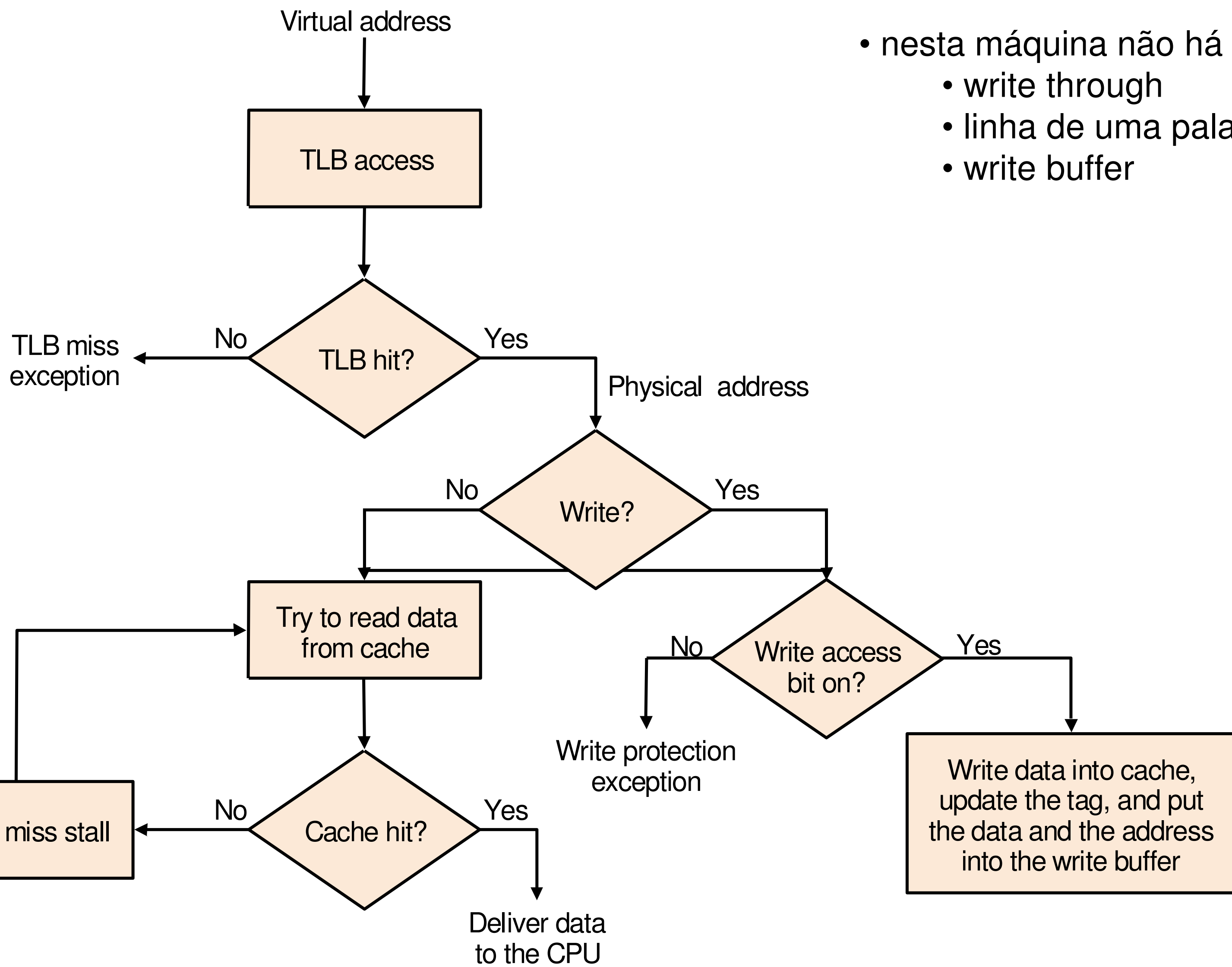
- mapeamento fully associative

- mapeamento direto

- pior caso: 3 misses TLB, PT, cache



TLBs and caches (DEC 3100)



- nesta máquina não há write hit
  - write through
  - linha de uma palavra
  - write buffer

TLB, Virtual memory and Cache (pag 595)

Cache	TLB	Virtual memory	Possible? If so, under what circumstance?
Miss	Hit	Hit	Possible, although the page table is never really checked if TLB hits.
Hit	Miss	Hit	TLB misses, but entry found in page table; after retry data is found in cache.
Miss	Miss	Hit	TLB misses, but entry found in page table; after retry data misses in cache.
Miss	Miss	Miss	TLB misses and is followed by a page fault; after retry, data must miss in cache.
Miss	Hit	Miss	Impossible: cannot have a translation in TLB if page is not present in memory.
Hit	Hit	Miss	Impossible: cannot have a translation in TLB if page is not present in memory.
Hit	Miss	Miss	Impossible: data cannot be allowed in cache if the page is not in memory.

Protection with Virtual Memory

- **Support at least two modes**
  - **user process**
  - **operating system process (***kernel, supervisor, executive***)**
- **CPU state that user process can read but not write**
  page table and TLB
  - special instructions that are only available in supervisor mode
- Mechanisms whereby the CPU can go from *user* mode to *supervisor* , and vice versa
  - user to supervisor : *system call exception*
  - supervisor to user : *return from exception (RFE)*
- OBS: page tables (operating system´s address space)

Handling Page Faults and TLB misses

- **TLB miss (*software or hardware*).**
  - **the page is present in memory, and we need only create the missing TLB entry.**
  - **the page is not present in memory, and we need to transfer control to the operating system to deal with a page fault.**
- **Page fault (*exception mechanism*).**
  - **OS saves the entire state the active process.**
  - **EPC = virtual address of the faulting page.**
  - **OS must complete three steps:**
    - look up the page table entry using the virtual address and find the location of referenced page on disk.
    - chose a physical page to replace; if the chosen page is **dirty**, it must be written out to disk before we can bring a new virtual page into this physical page.
    - Start a read to bring the referenced page from disk into the chosen physical page.

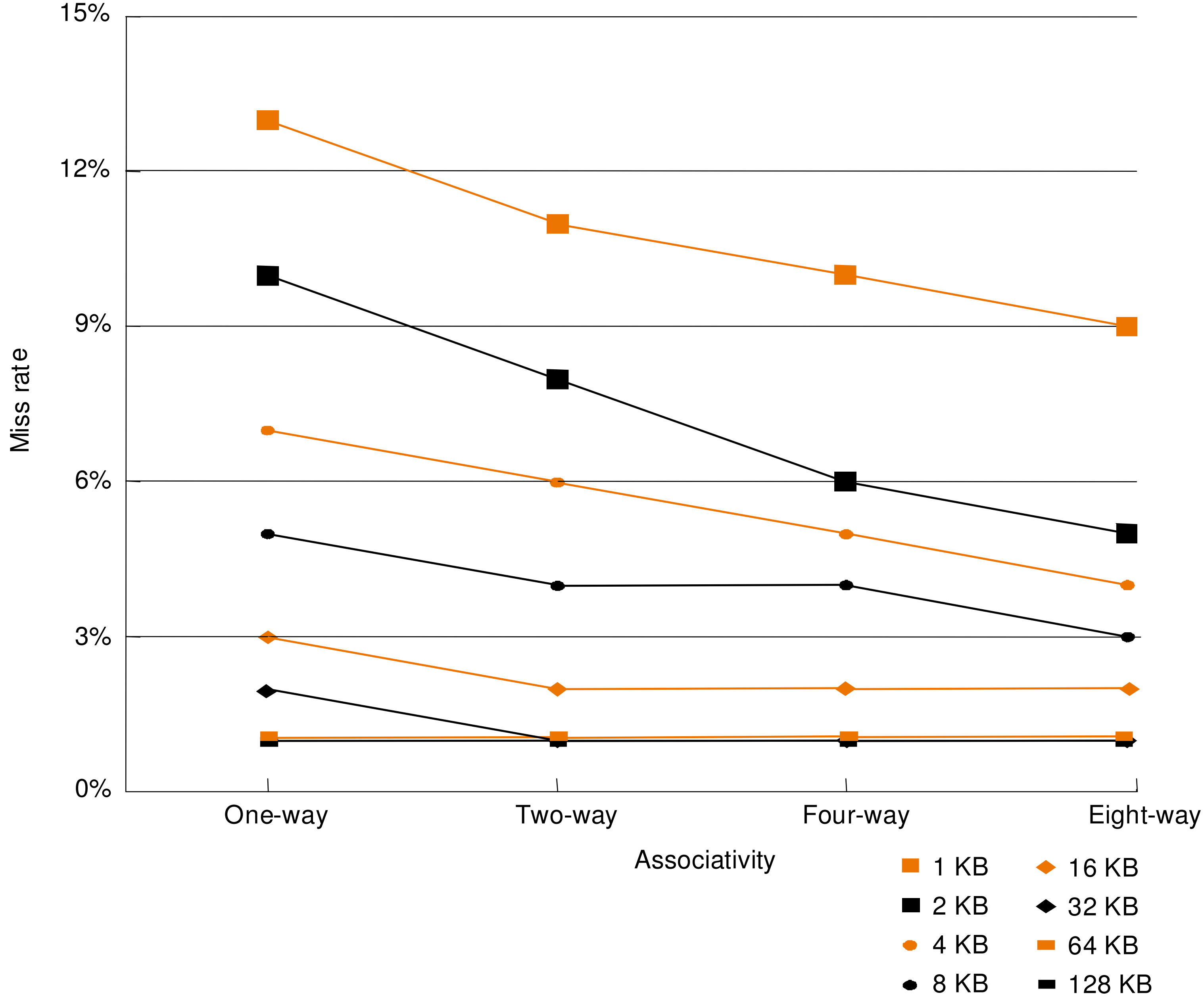
Memory Hierarchies

- **Where can a Block Be Placed?**

Scheme name	Number of sets	Block per set
Direct mapped	Number of blocks in cache	1
Set associative	$\frac{\text{Number of blocks in cache}}{\text{Associativity}}$	Associativity (typically 2 – 8)
Fully associative	1	Number of block in the cache

Feature	Typical values for cache	Typical values for page memory	Typical values for a TLB
Total size in blocks	1000 –100,000	2000 – 250,000	32 – 4,000
Total size in kilobytes	8 – 8,000	8000 – 8,000,000	0.254 – 32
Block size in bytes	16 – 256	4000 – 64,000	4 – 32
Miss penalty in clocks	10 – 100	1,000,000 – 10,000,000	10 – 100
Miss rate	0.1% -- 10%	0.00001% -- 0.0001%	0.01% -- 2%

Miss rate vs set associativity



Memory Hierarchies

- **How Is a Block Found?**

Associativity	Location method	Comparisons required
Direct mapped	Index	1
Set associative	Index the set, search among elements	Degree of associativity
Full	Search all cache entries	Size of the cache
	Separate lookup table	0

- **OBS.: In virtual memory systems**
  - **Full associativy is beneficial, since misses are very expensive**
  - **Full associativity allows software to use sophisticated replacement schemes that are designed to reduce the miss rate.**
  - **The full map can be easily indexed with no extra hardware and no searching required**
  - **The large page size means the page table size overhead is relatively small.**

Memory Hierarchies

- **Which Block Should Be Replaced on a Cache Miss?**
  - **Random : candidate blocks are randomly selected, possibly using some hardware assistance.**
  - **Least Recently Used (LRU): The block replaced is the one that has been unused for the longest time**

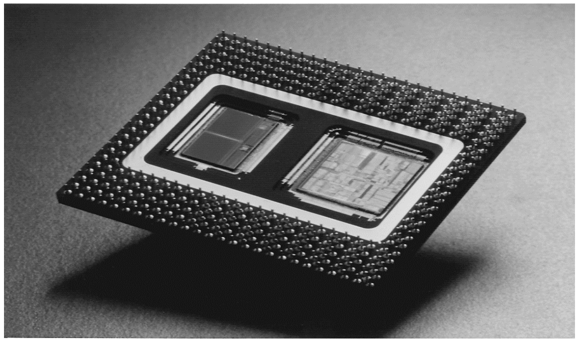
Memory Hierarchies

- **What Happens on a Write?**
  - **Write-through**
    - Misses are simpler and cheaper because they never require a block to be written back to the lower level.
    - It is easier to implement than write-back, although to be practical in a high-speed system, a write-through cache will need to use a write buffer
  - **Write-back (copy-back)**
    - Individuals words can be written by the processor at the rate that the cache, rather than the memory, can accept them.
    - Multiple writes within a block require only one write to the lower level in the hierarchy.
    - When blocks are written back, the system can make effective use of a high bandwidth transfer, since the entire block is written

Modern Systems

- **Very complicated memory systems:**

Characteristic	Intel Pentium Pro	PowerPC 604
Virtual address	32 bits	52 bits
Physical address	32 bits	32 bits
Page size	4 KB, 4 MB	4 KB, selectable, and 256 MB
TLB organization	A TLB for instructions and a TLB for data Both four-way set associative Pseudo-LRU replacement Instruction TLB: 32 entries Data TLB: 64 entries TLB misses handled in hardware	A TLB for instructions and a TLB for data Both two-way set associative LRU replacement Instruction TLB: 128 entries Data TLB: 128 entries TLB misses handled in hardware



Characteristic	Intel Pentium Pro	PowerPC 604
Cache organization	Split instruction and data caches	Split intruction and data caches
Cache size	8 KB each for instructions/data	16 KB each for instructions/data
Cache associativity	Four-way set associative	Four-way set associative
Replacement	Approximated LRU replacement	LRU replacement
Block size	32 bytes	32 bytes
Write policy	Write-back	Write-back or write-through

Some Issues

- **Processor speeds continue to increase very fast**
  — much faster than either DRAM or disk access times
- **Design challenge: dealing with this growing disparity**
- **Trends:**
  - **synchronous SRAMs (provide a burst of data)**
  - **redesign DRAM chips to provide higher bandwidth or processing**
  - **restructure code to increase locality**
  - **use prefetching (make cache visible to ISA)**

Evolução desempenho CPU vs Mem