Virtual Memory

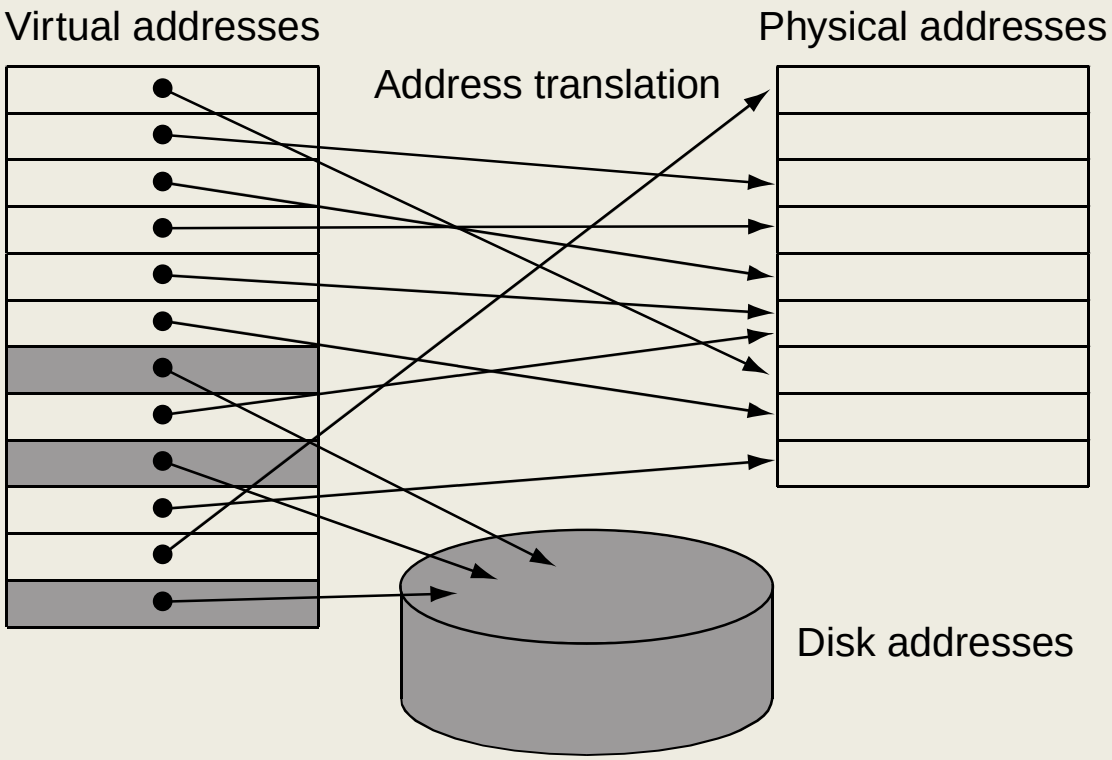
Patterson & Hennessey

Chapter 5

Virtual Memory

- Use main memory as a “cache” for secondary (disk) storage
  - Managed jointly by CPU hardware and the operating system (OS)
- Programs share main memory
  - Each gets a private virtual address space holding its frequently used code and data
  - Protected from other programs
- CPU and OS translate virtual addresses to physical addresses
  - VM “block” is called a page
  - VM translation “miss” is called a page fault

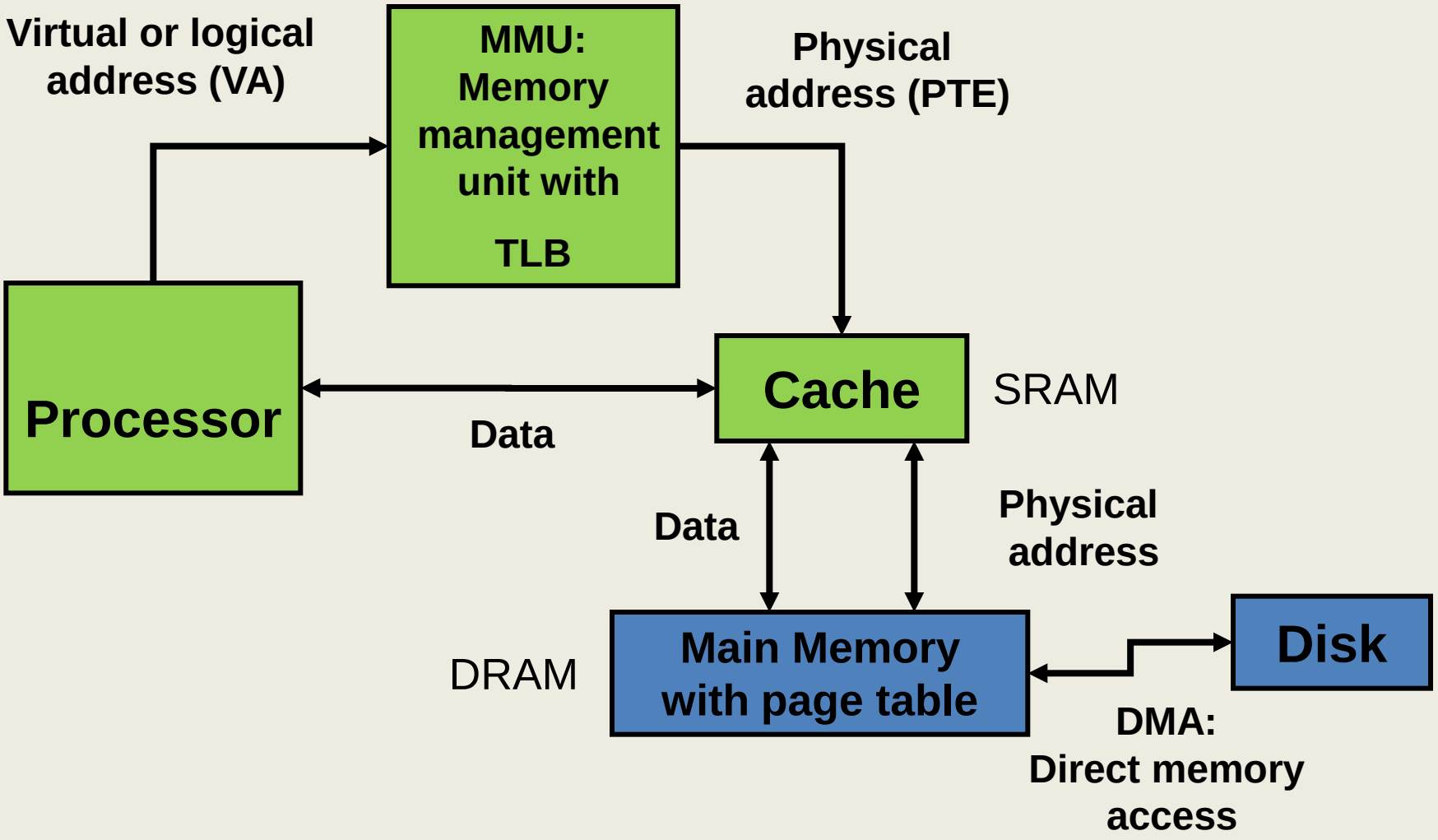
Virtual Memory Address Mapping



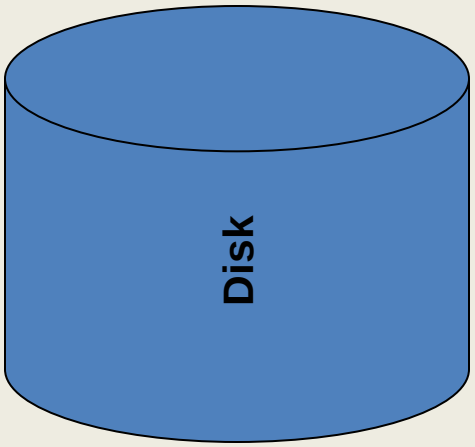
Memory Hierarchy Example

- 32-bit address (byte addressing)
- 4 GB virtual main memory (disk space)
  - Page size = 4 KB
  - Number of virtual pages = $4 \times 2^{30}/(4 \times 2^{10}) = 1M$
  - Bits for virtual page number = $\log_2(1M) = 20$
- 128 MB physical main memory
  - Page size 4 KB
  - Number of physical pages = $128 \times 2^{20}/(4 \times 2^{10}) = 32K$
  - Bits for physical page number = $\log_2(32K) = 15$
  - Page table contains 1M records specifying where each virtual page is located.

Virtual Memory System



Cache Miss and Page Fault



All data, organized in Pages (~4KB), accessed by Physical addresses

Pages

(Write-back, same as in cache)

Main Memory

Cached pages, Page table

Page fault: a required page is not found in main memory

Processor Cache MMU

Cache miss: a required block is not found in cache

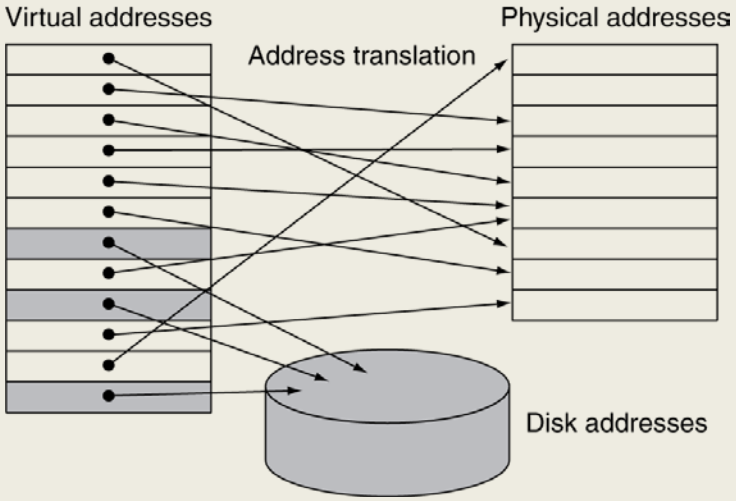
"Page fault" in virtual memory is similar to "miss" in cache.

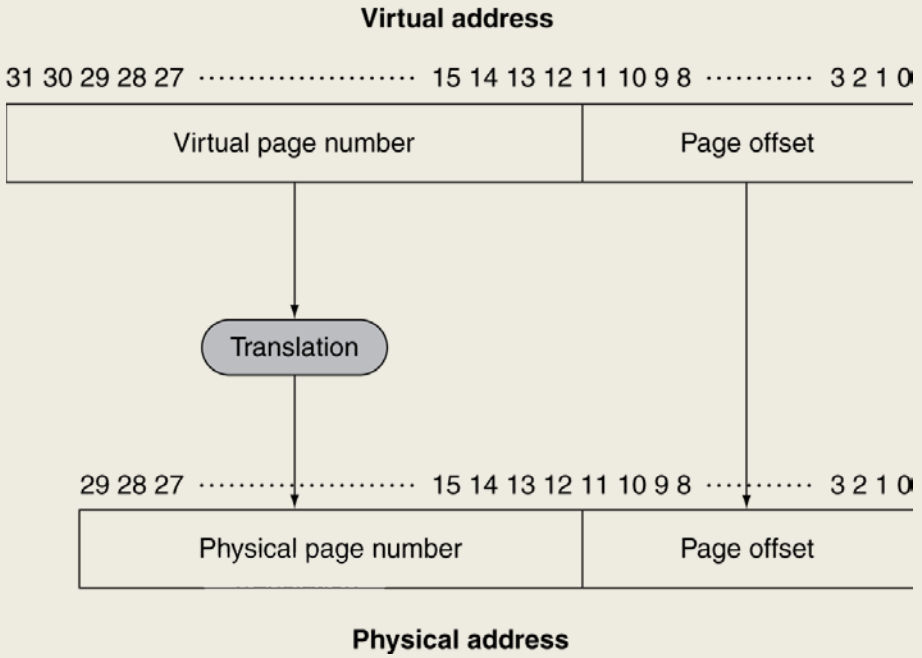
Virtual vs. Physical Address

- Processor assumes a virtual memory addressing scheme:
  - Disk is a virtual memory (large, slow)
  - A block of data is called a virtual page
  - An address is called virtual (or logical) address (VA)
- Main memory may have a different addressing scheme:
  - Physical memory consists of caches
  - Memory address is called physical address
  - MMU translates virtual address to physical address
  - Complete address translation table is large and is kept in main memory
  - MMU contains TLB (translation lookaside buffer), which keeps record of recent address translations.

Address Translation

Fixed-size pages (e.g., 4K)





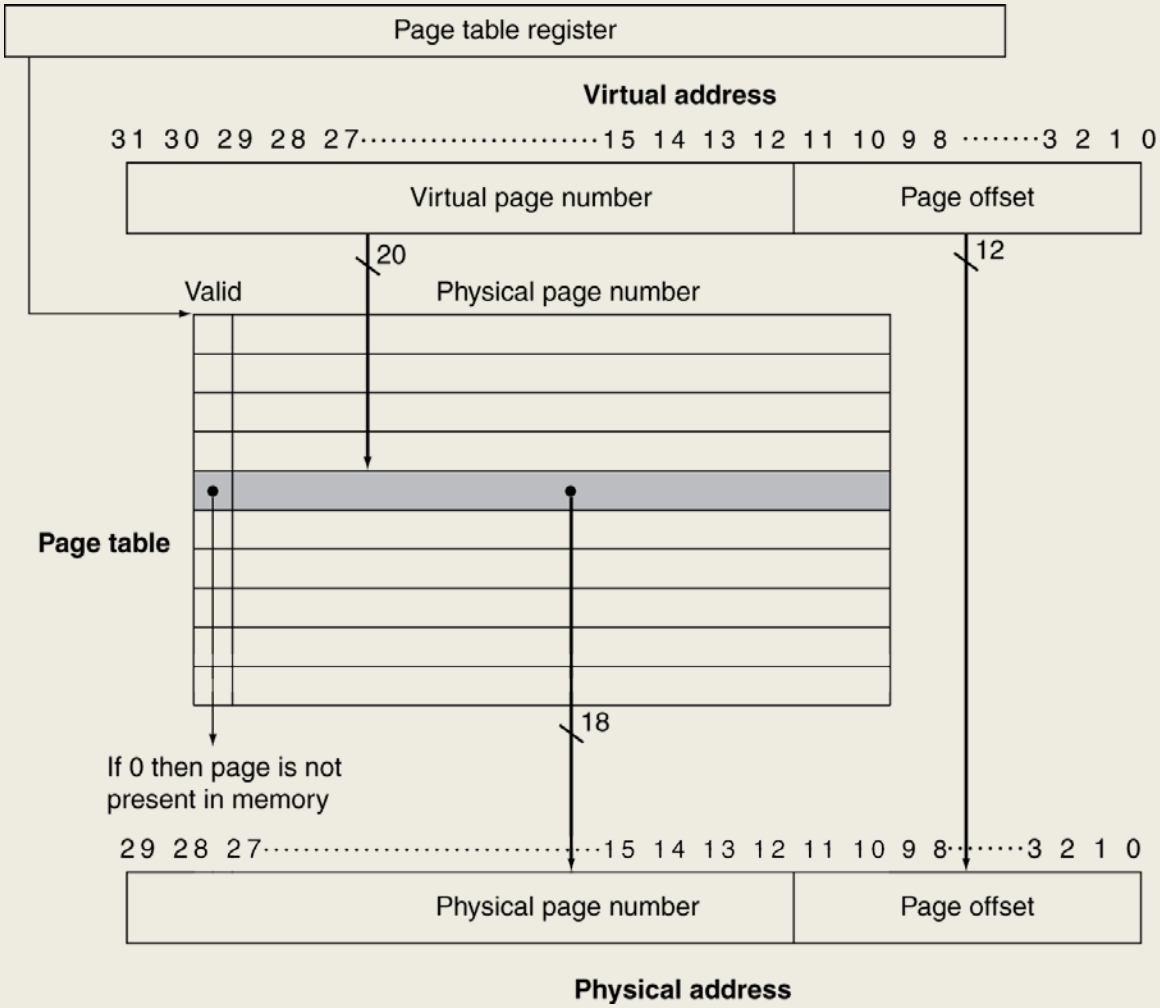
Page Fault Penalty

- On page fault, the page must be fetched from disk
  - Takes millions of clock cycles
    - huge miss penalty
  - Handled by OS code
- Try to minimize page fault rate
  - Fully associative placement
  - Smart replacement algorithms (eg. LRU)

Page Tables

- Stores placement information
  - Array of page table entries (PTE), indexed by virtual page number
  - Page table register in CPU points to page table in physical memory
- If page is present in memory
  - PTE stores the physical page number
  - Plus other status bits (referenced, dirty, …)
- If page is not present
  - PTE can refer to location in swap space on disk

Translation Using a Page Table



Page Fault Handler

- Use faulting virtual address to find PTE
- Locate page on disk
- Choose page to replace
  - If dirty, write to disk first
- Read page into memory and update page table
- Make process runnable again
  - Restart from faulting instruction

Replacement and Writes

- To reduce page fault rate, prefer least-recently used (LRU) replacement
  - Reference bit (aka use bit) in PTE set to 1 on access to page
  - Periodically cleared to 0 by OS
  - A page with reference bit = 0 has not been used recently
- Disk writes take millions of cycles
  - Transfer entire block, not individual locations
  - Write-through is impractical
  - Use write-back
  - Dirty bit in PTE set when page is written

Memory Protection

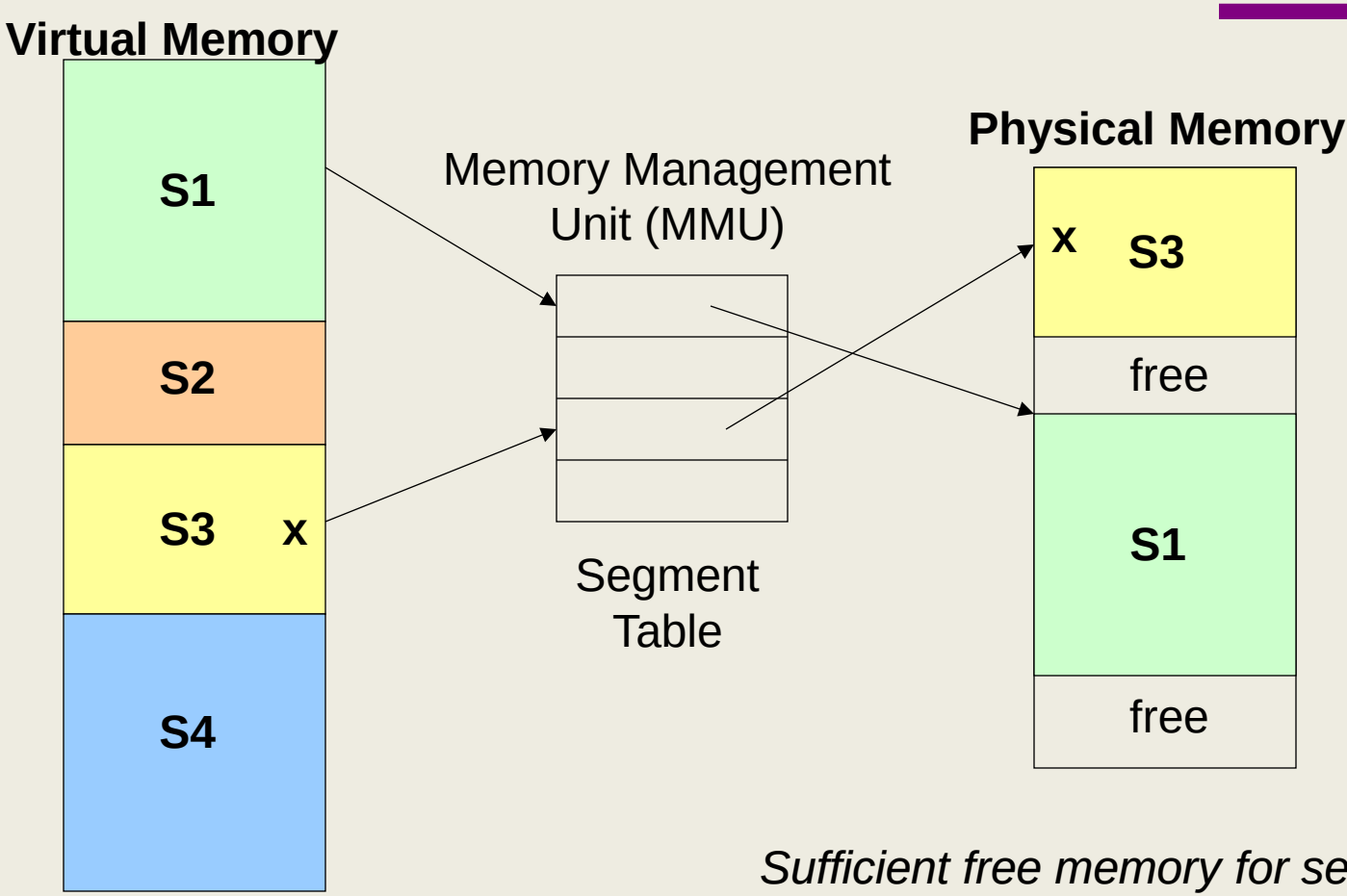
- Different tasks can share parts of their virtual address spaces
  - But need to protect against errant access
  - Requires OS assistance
- Hardware support for OS protection
  - Privileged supervisor mode (aka kernel mode)
  - Privileged instructions
  - Page tables and other state information only accessible in supervisor mode
  - System call exception (e.g., syscall in MIPS)

Segmentation vs. Paging

- Page: transparent to programmer
  - Virtual address partitioned into fixed-size pages
  - Physical memory likewise partitioned
  - Virtual pages map exactly into physical pages

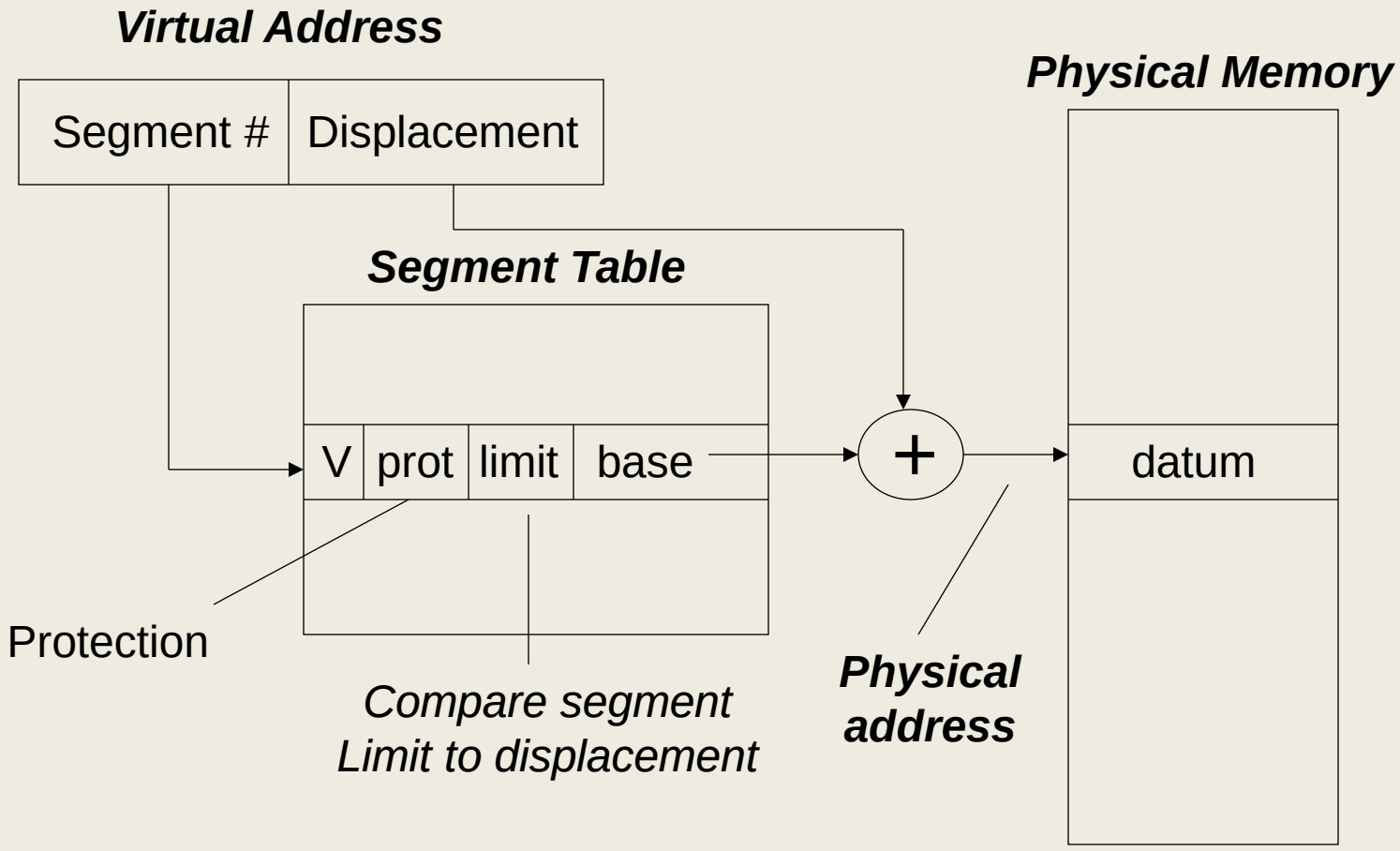
- Segment: defined by programmer/compiler
  - Compiler partitions program/data into segments of related information (code, data, etc.)
  - Segments easily protected (read-only, execute-only, etc.)
  - Segments can be of variable size
  - Memory manager must find sufficient free memory to hold each segment (fragmentation may occur)
  - Less common than paging

Segmented Memory

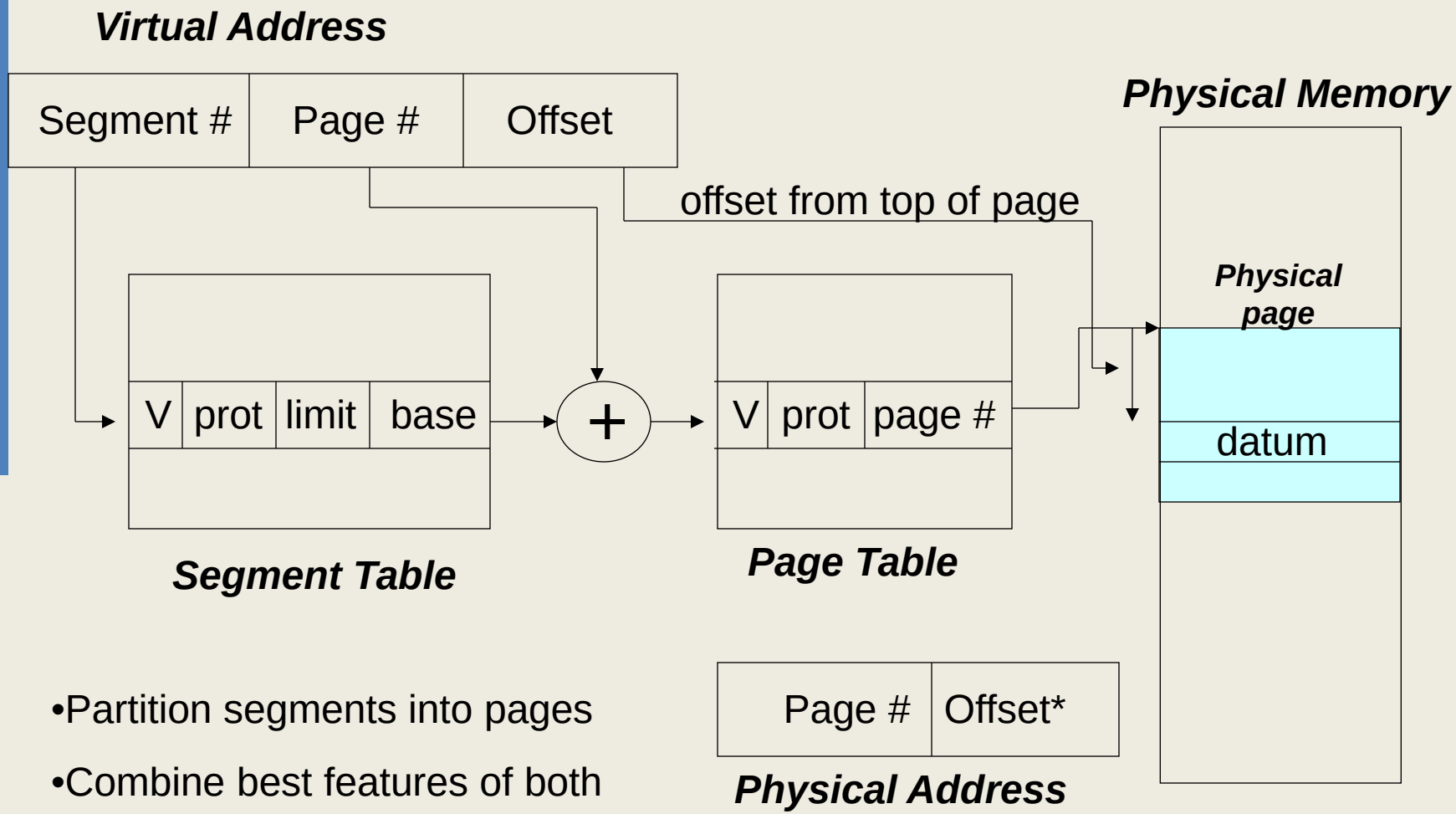


Sufficient free memory for segment S2, but fragmented, preventing loading of S2

Mapping Segmented Address



Combining Segmentation and Paging

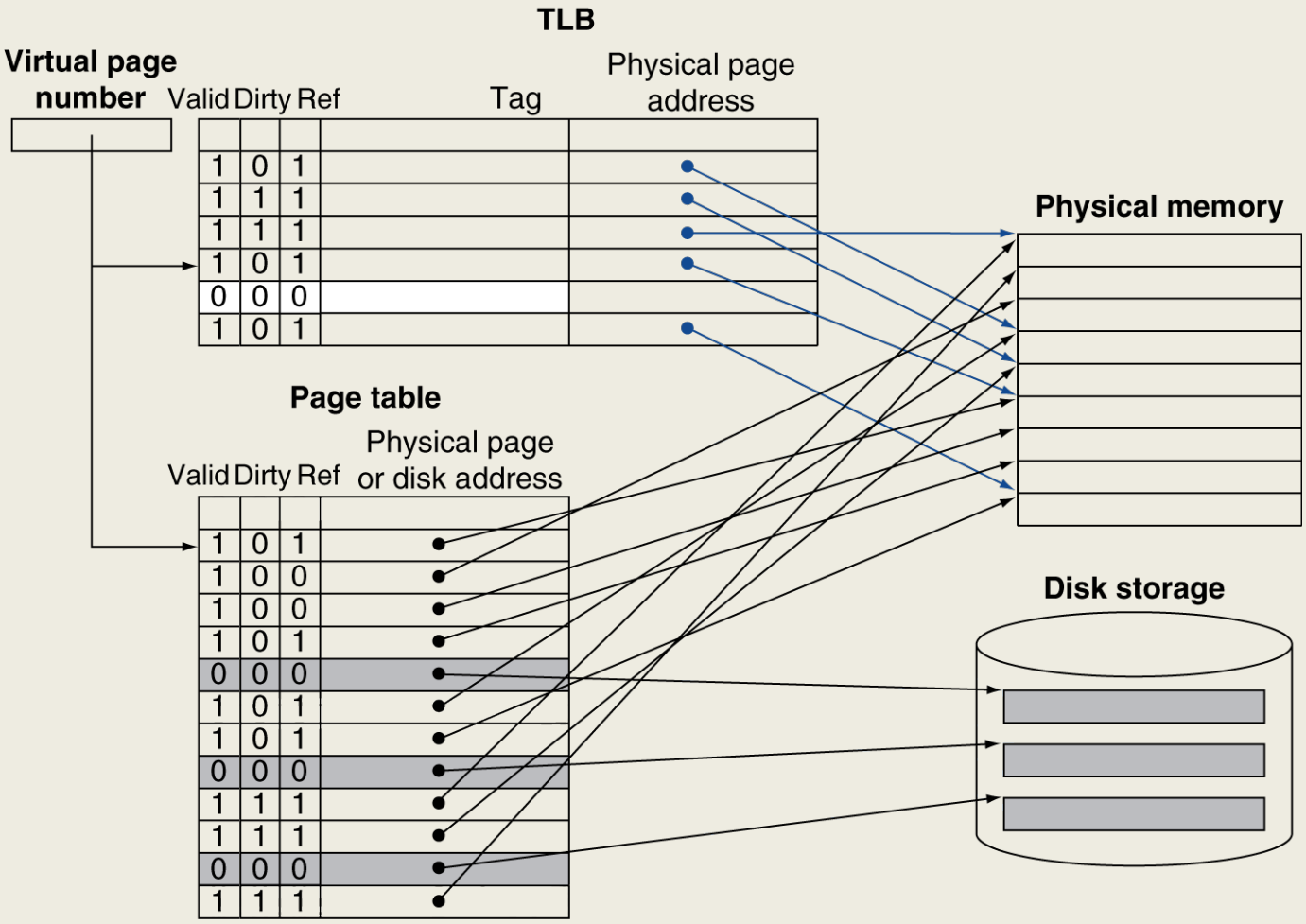


- Partition segments into pages
- Combine best features of both

Fast Translation Using a TLB

- Address translation would appear to require extra memory references
  - One to access the PTE
  - Then the actual memory access
- But access to page tables has good locality
  - So use a fast cache of PTEs within the CPU
  - Called a Translation Look-aside Buffer (TLB)
  - Typical: 16–512 PTEs, 0.5–1 cycle for hit, 10–100 cycles for miss, 0.01%–1% miss rate
  - Misses could be handled by hardware or software

Fast Translation Using a TLB



TLB Misses

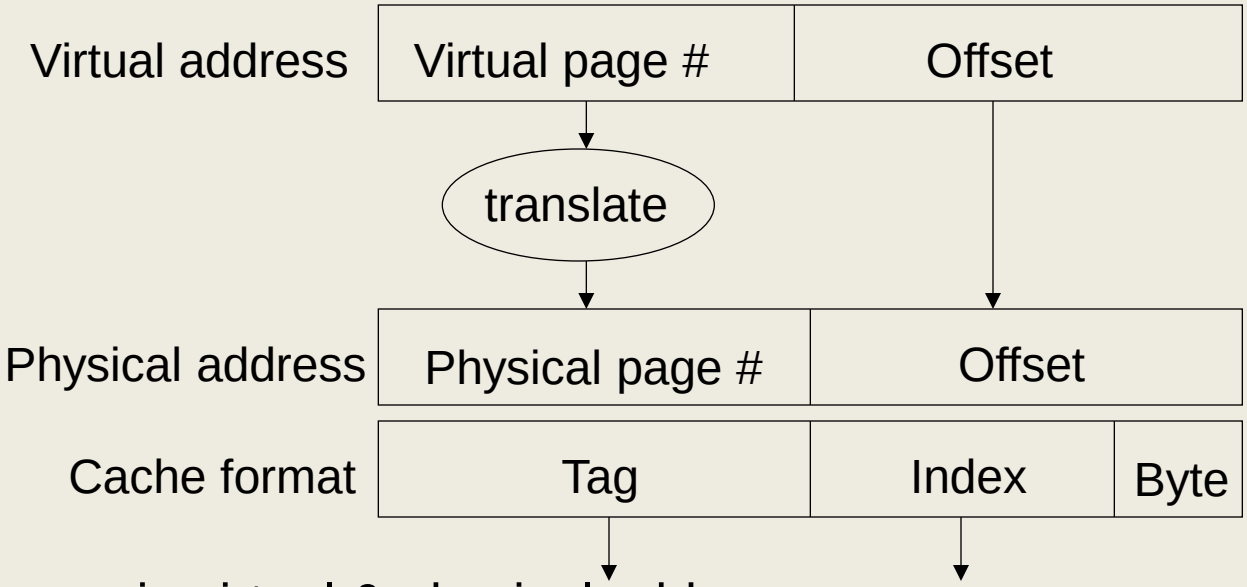
- If page is in memory
  - Load the PTE from memory and retry
  - Could be handled in hardware
    - Can get complex for more complicated page table structures
  - Or in software
    - Raise a special exception, with optimized handler
- If page is not in memory (page fault)
  - OS handles fetching the page and updating the page table
  - Then restart the faulting instruction

TLB Miss Handler

- TLB miss indicates
  - Page present, but PTE not in TLB
  - Page not present
- Must recognize TLB miss before destination register overwritten
  - Raise exception
- Handler copies PTE from memory to TLB
  - Then restarts instruction
  - If page not present, page fault will occur

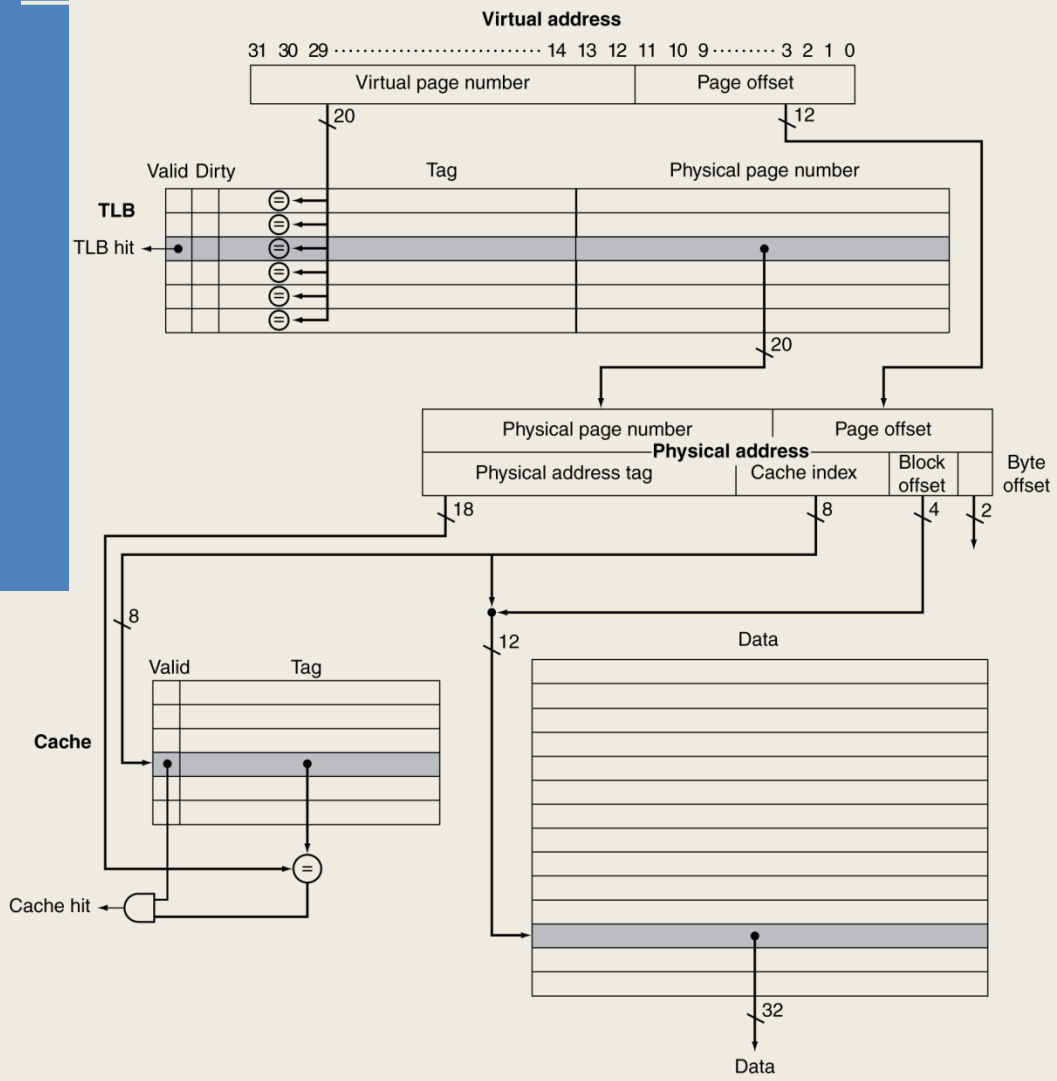
Concurrent TLB & Cache Access

- Normally access main memory cache after translating logical to physical address
- Can do these concurrently to save time



- Offset same in virtual & physical address,
- Index available for cache line access while translation occurs
- Check tag after translation and cache line access

TLB and Cache Interaction



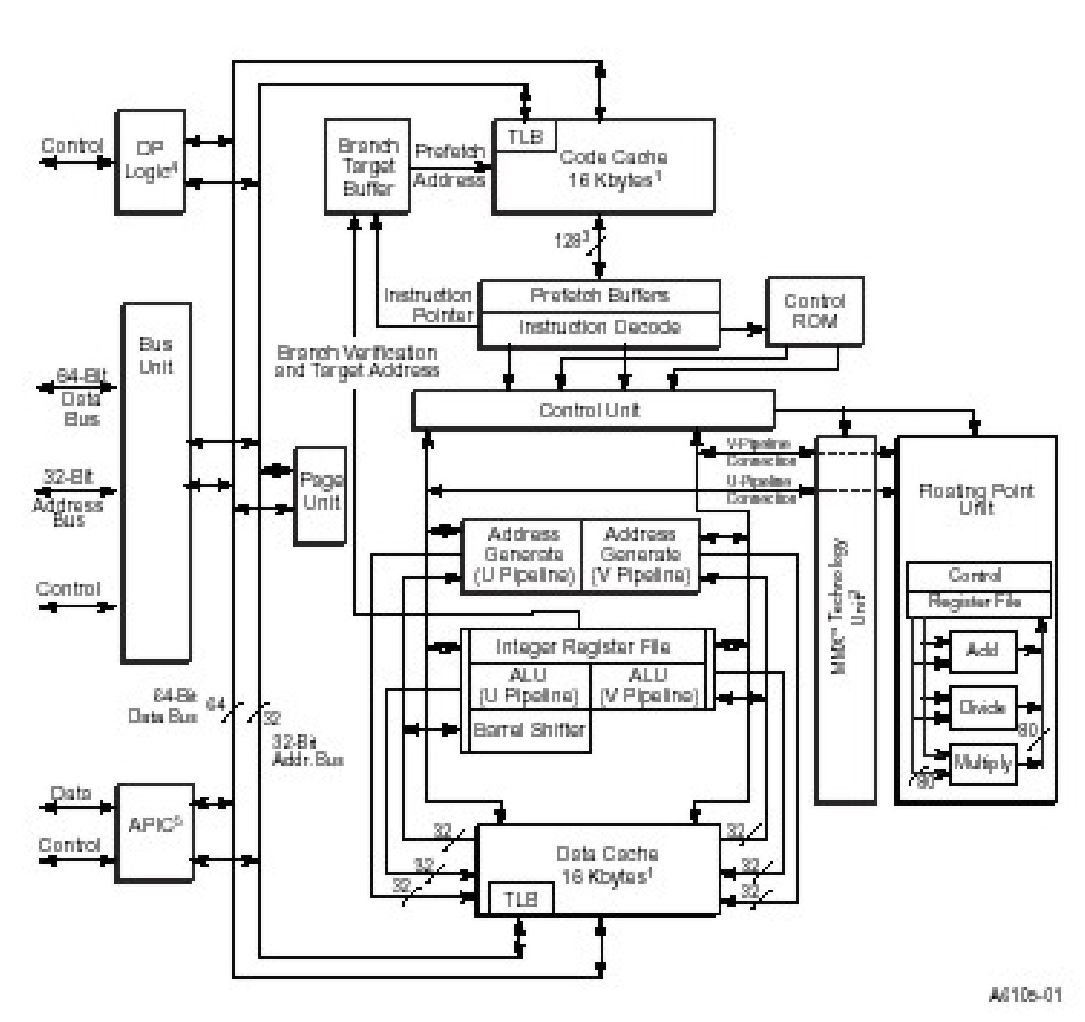
- If cache tag uses physical address
  - Need to translate before cache lookup
- Alternative: use virtual address tag
  - Complications due to aliasing
    - Different virtual addresses for shared physical address

Modern Systems

Characteristic	Intel Pentium P4	AMD Opteron
Virtual address	32 bits	48 bits
Physical address	36 bits	40 bits
Page size	4 KB, 2/4 MB	4 KB, 2/4 MB
TLB organization	1 TLB for instructions and 1 TLB for data Both are four-way set associative Both use pseudo-LRU replacement Both have 128 entries TLB misses handled in hardware	2 TLBs for instructions and 2 TLBs for data Both L1 TLBs fully associative, LRU replacement Both L2 TLBs are four-way set associativity, round-robin LRU Both L1 TLBs have 40 entries Both L2 TLBs have 512 entries TLB misses handled in hardware

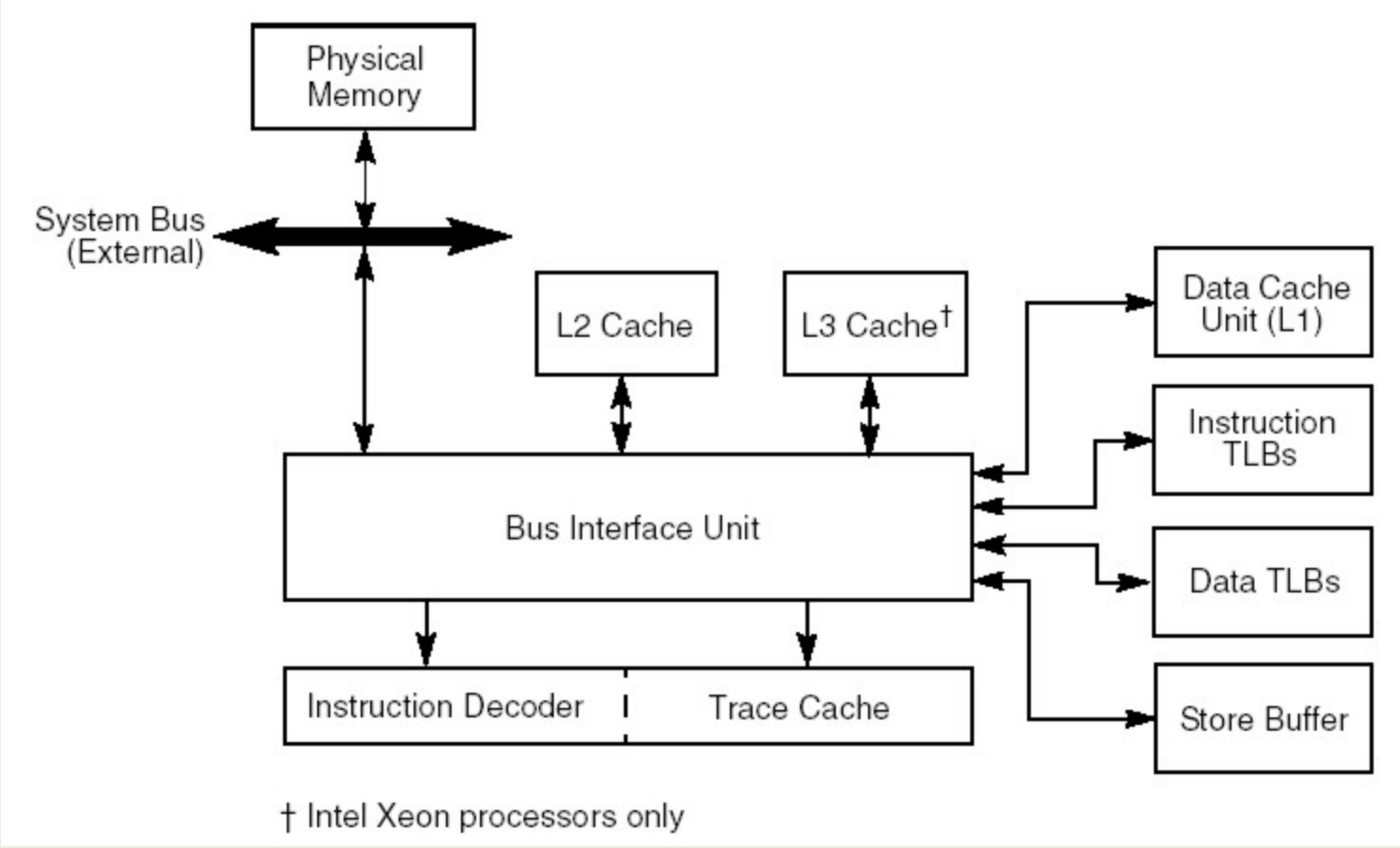
FIGURE 7.34 Address translation and TLB hardware for the Intel Pentium P4 and AMD Opteron. The word size sets the maximum size of the virtual address, but a processor need not use all bits. The physical address size is independent of word size. The P4 has one TLB for instructions and a separate identical TLB for data, while the Opteron has both an L1 TLB and an L2 TLB for instructions and identical L1 and L2 TLBs for data. Both processors provide support for large pages, which are used for things like the operating system or mapping a frame buffer. The large-page scheme avoids using a large number of entries to map a single object that is always present.

Pentium Microarchitecture



From Intel Pentium Ref. Manual

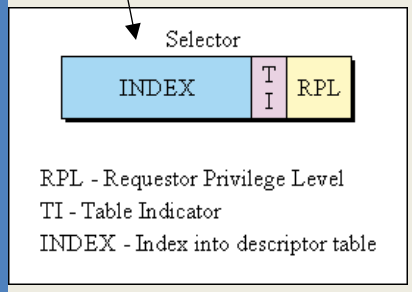
Pentium Caches

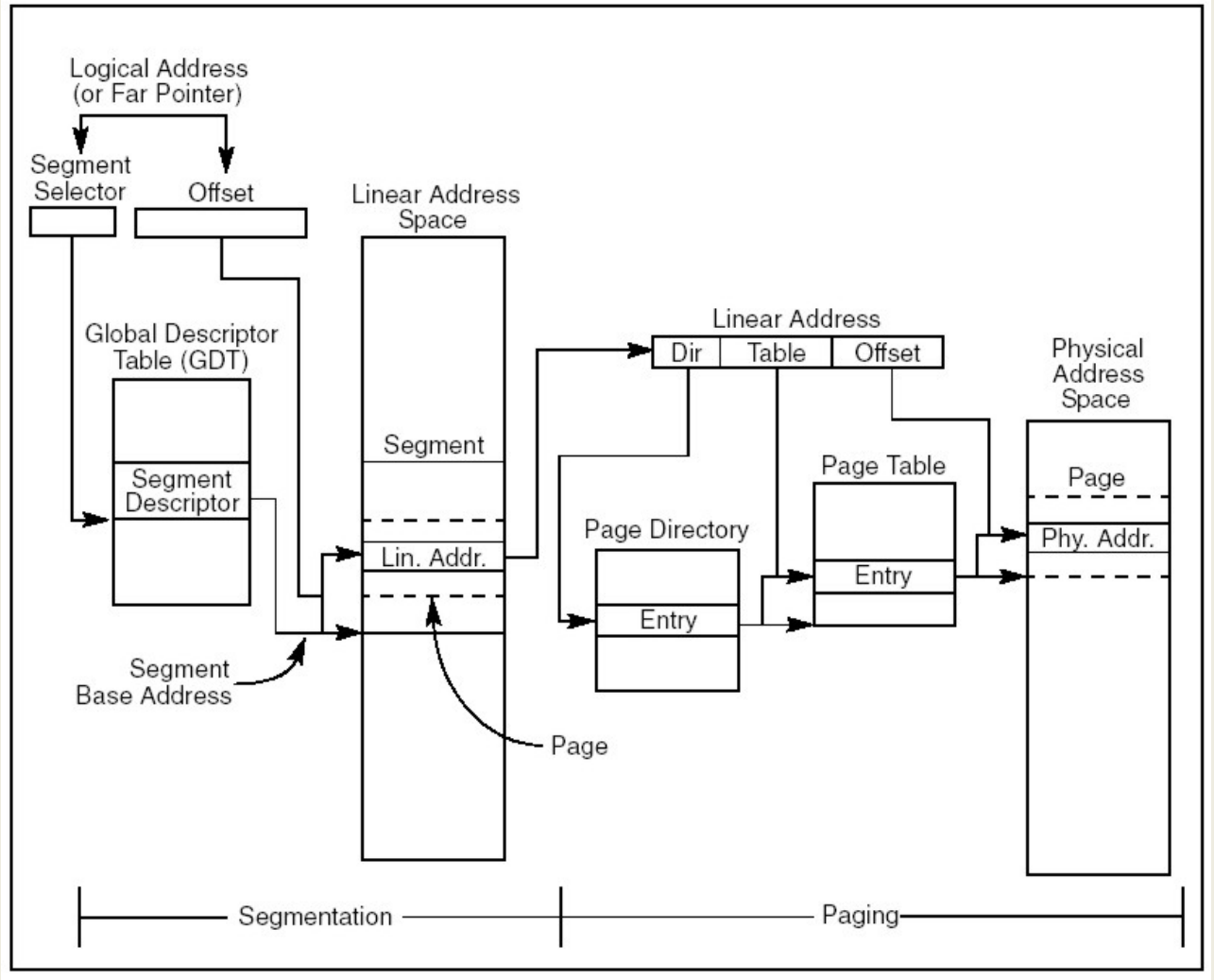


From Intel Pentium Ref. Manual

Pentium Memory Mapping

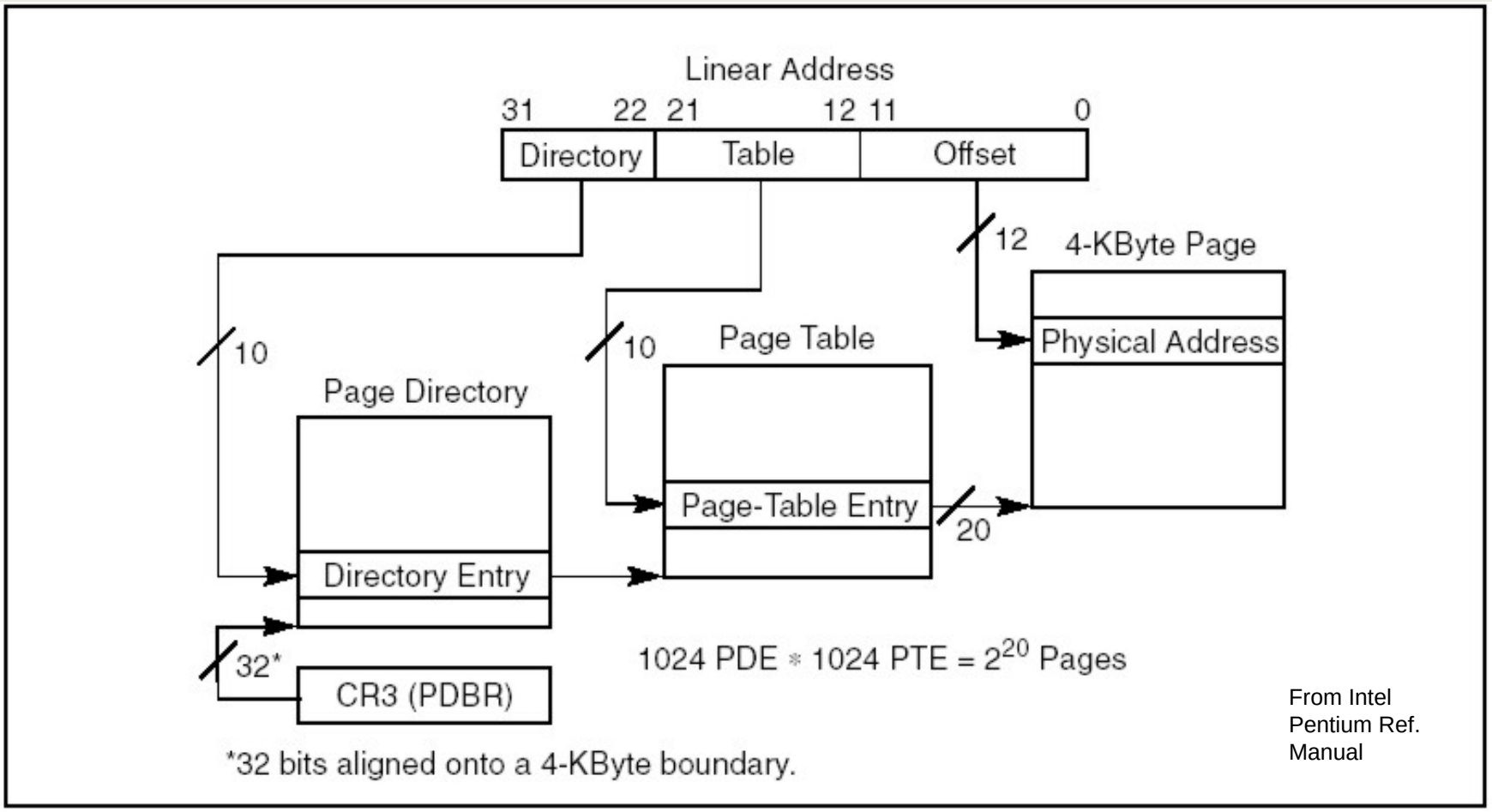
13 bit index





From Intel Pentium Ref. Manual

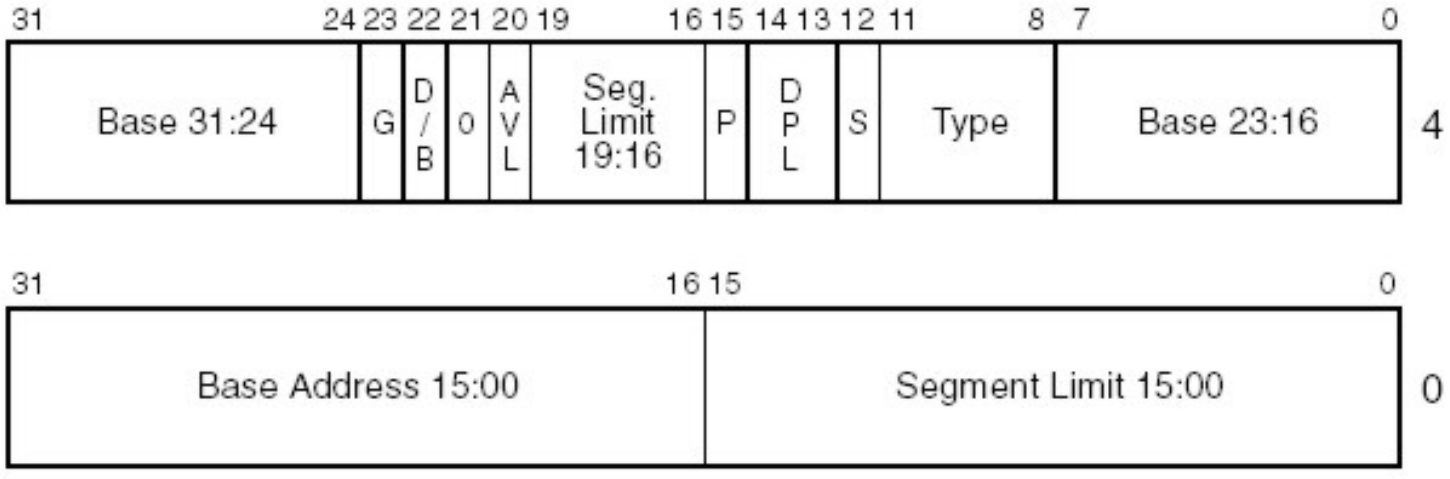
Pentium Paging Details



From Intel Pentium Ref. Manual

Figure 3-12. Linear Address Translation (4-KByte Pages)

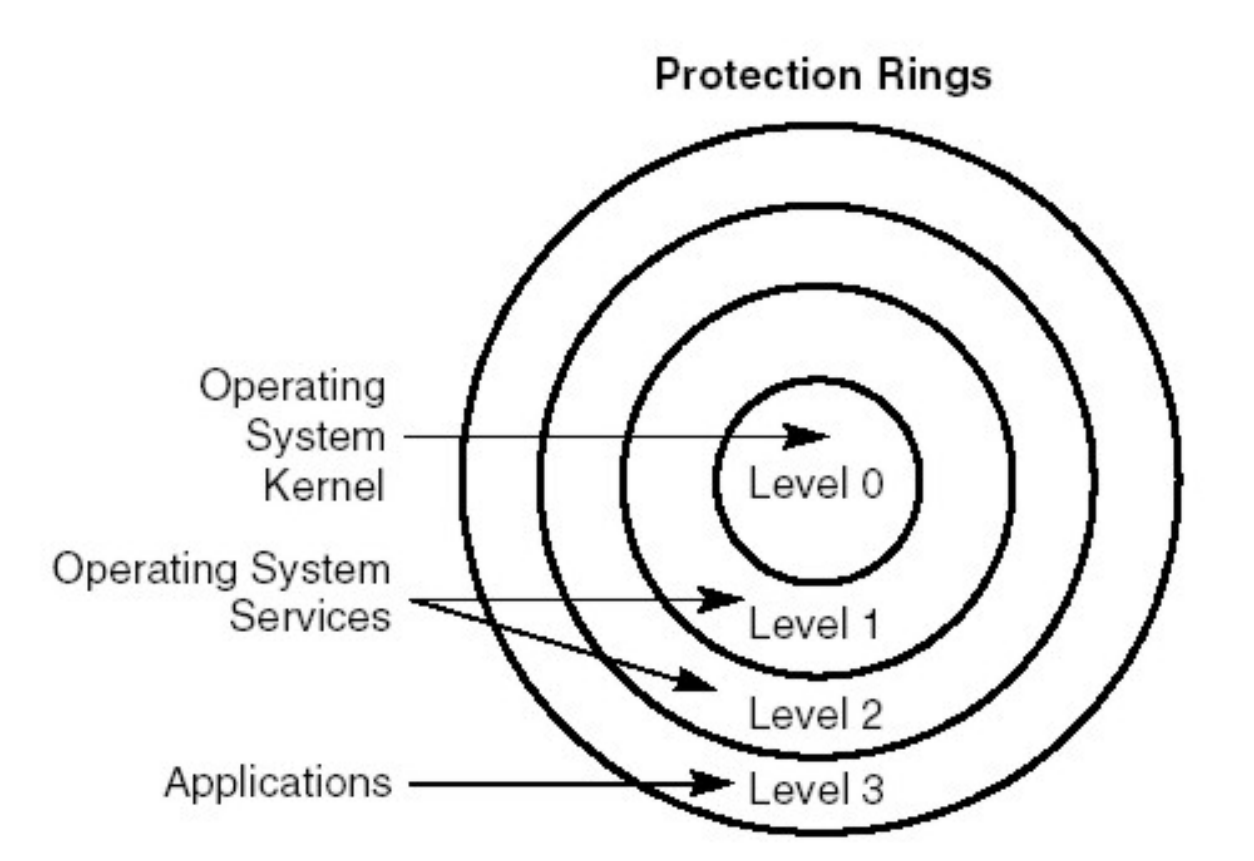
Pentium Segment Descriptor



AVL — Available for use by system software
BASE — Segment base address
D/B — Default operation size (0 = 16-bit segment; 1 = 32-bit segment)
DPL — Descriptor privilege level
G — Granularity
LIMIT — Segment Limit
P — Segment present
S — Descriptor type (0 = system; 1 = code or data)
TYPE — Segment type

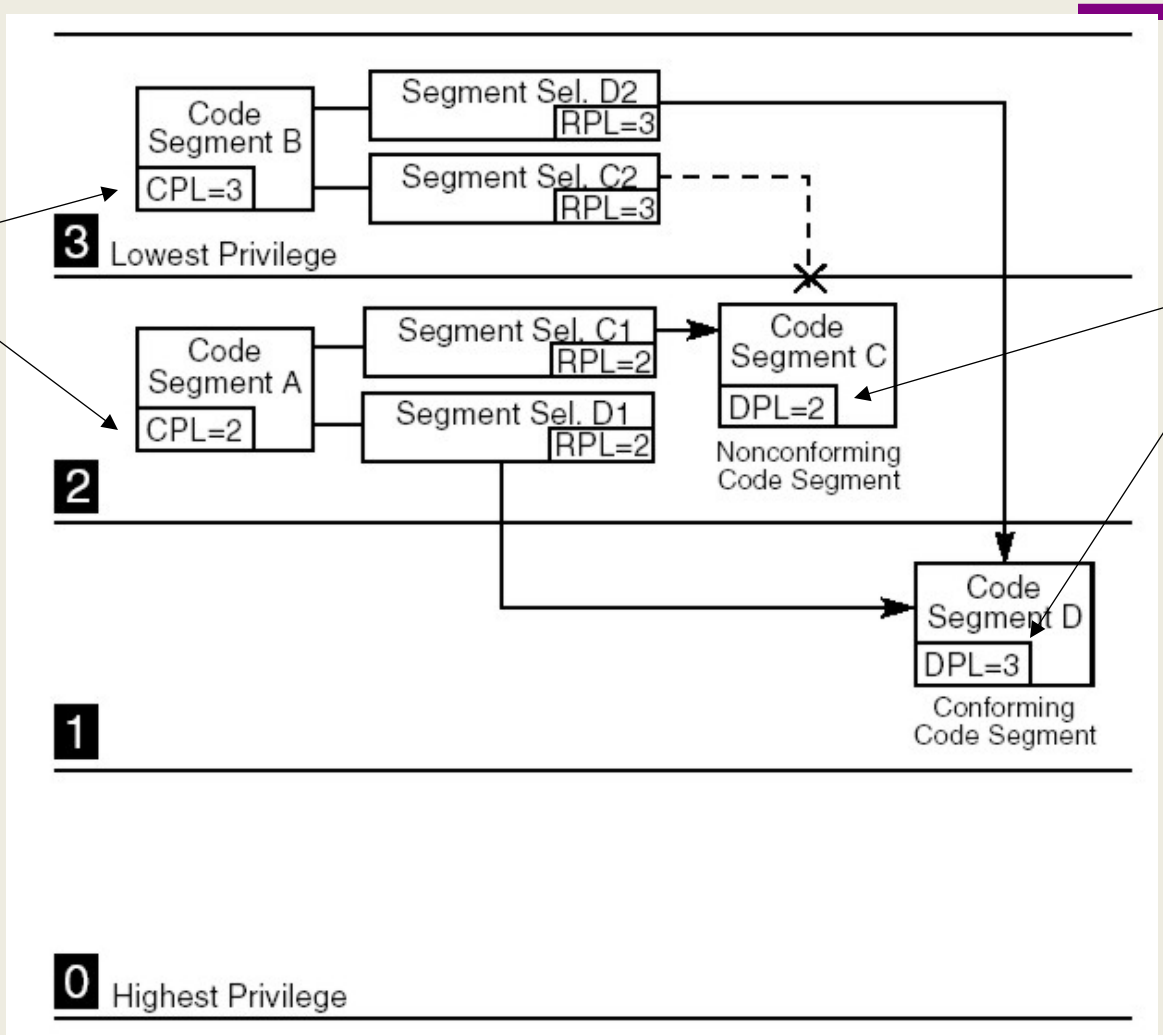
From Intel Pentium Ref. Manual

Pentium Protection (Privilege) Levels



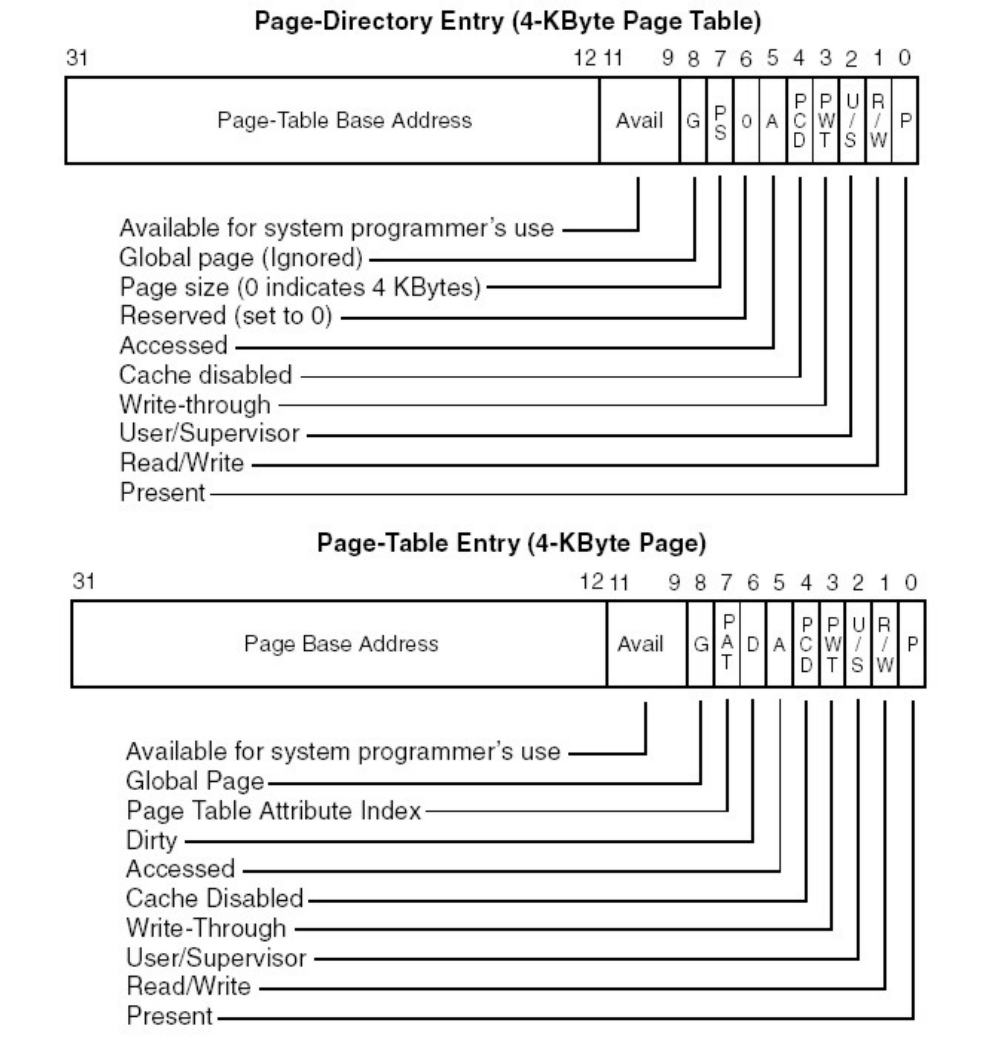
From Intel Pentium Ref. Manual

Pentium Privilege-Level Protection



From Intel Pentium Ref. Manual

Pentium Page Descriptors



From Intel Pentium Ref. Manual

Modern Systems

- Things are getting complicated!

MPU	AMD Opteron	Intrinsity FastMATH	Intel Pentium 4	Intel PXA250	Sun UltraSPARC IV
Instruction set architecture	IA-32, AMD64	MIPS32	IA-32	ARM	SPARC v9
Intended application	server	embedded	desktop	low-power embedded	server
Die size (mm^2) (2004)	193	122	217		356
Instructions issued/clock	3	2	3 RISC ops	1	4 × 2
Clock rate (2004)	2.0 GHz	2.0 GHz	3.2 GHz	0.4 GHz	1.2 GHz
Instruction cache	64 KB, 2-way set associative	16 KB, direct mapped	12000 RISC op trace cache (~96 KB)	32 KB, 32-way set associative	32 KB, 4-way set associative
Latency (clocks)	3?	4	4	1	2
Data cache	64 KB, 2-way set associative	16 KB, 1-way set associative	8 KB, 4-way set associative	32 KB, 32-way set associative	64 KB, 4-way set associative
Latency (clocks)	3	3	2	1	2
TLB entries (I/D/L2 TLB)	40/40/512/512	16	128/128	32/32	128/512
Minimum page size	4 KB	4 KB	4 KB	1 KB	8 KB
On-chip L2 cache	1024 KB, 16-way set associative	1024 KB, 4-way set associative	512 KB, 8-way set associative	—	—
Off-chip L2 cache	—	—	—	—	16 MB, 2-way set associative
Block size (L1/L2, bytes)	64	64	64/128	32	32

FIGURE 7.36 Desktop, embedded, and server microprocessors in 2004. From a memory hierarchy perspective, the primary differences between categories is the L2 cache. There is no L2 cache for the low-power embedded, a large on-chip L2 for the embedded and desktop, and 16 MB off chip for the server. The processor clock rates also vary: 0.4 GHz for low-power embedded, 1 GHz or higher for the rest. Note that UltraSPARC IV has two processors on the chip.