5

Memory-system Architecture

Memory-system

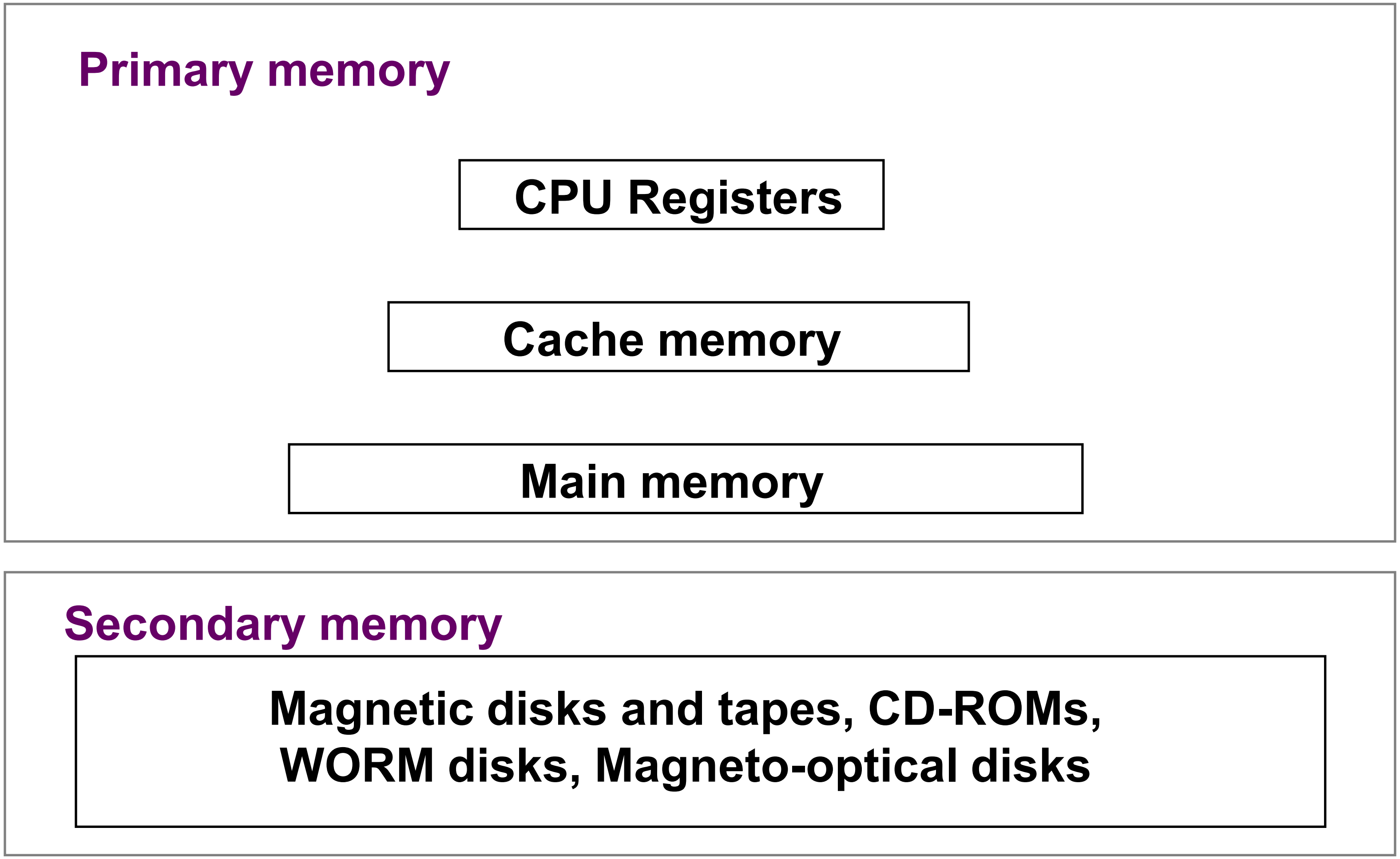
Cache Memory

Virtual Memory

Memory Design Issues

Memory-system

Memory hierarchy (Multilevel storage system)



Primary memory

CPU Registers

Cache memory

Main memory

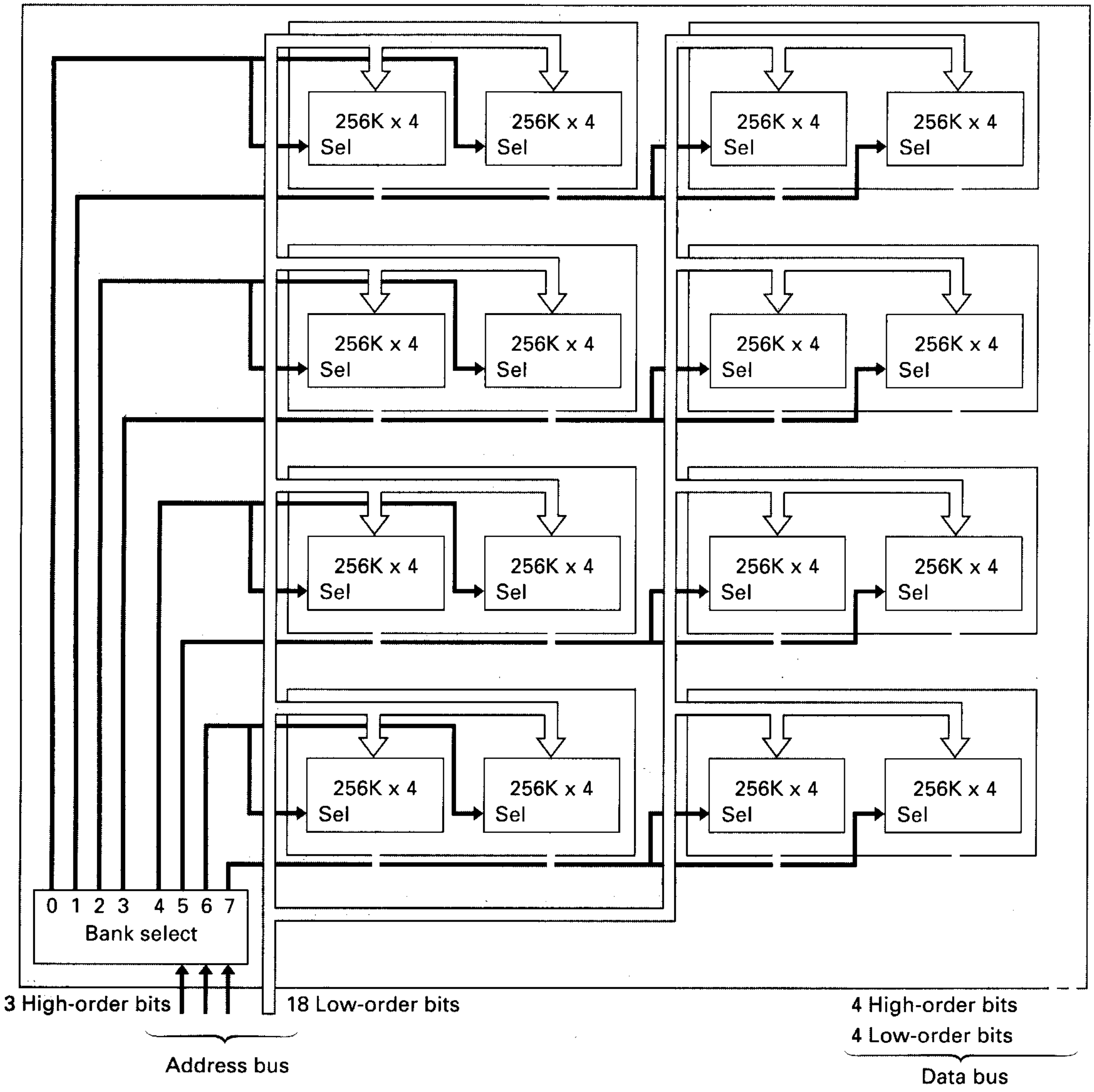
Secondary memory

Magnetic disks and tapes, CD-ROMs,
WORM disks, Magneto-optical disks

Memory organization

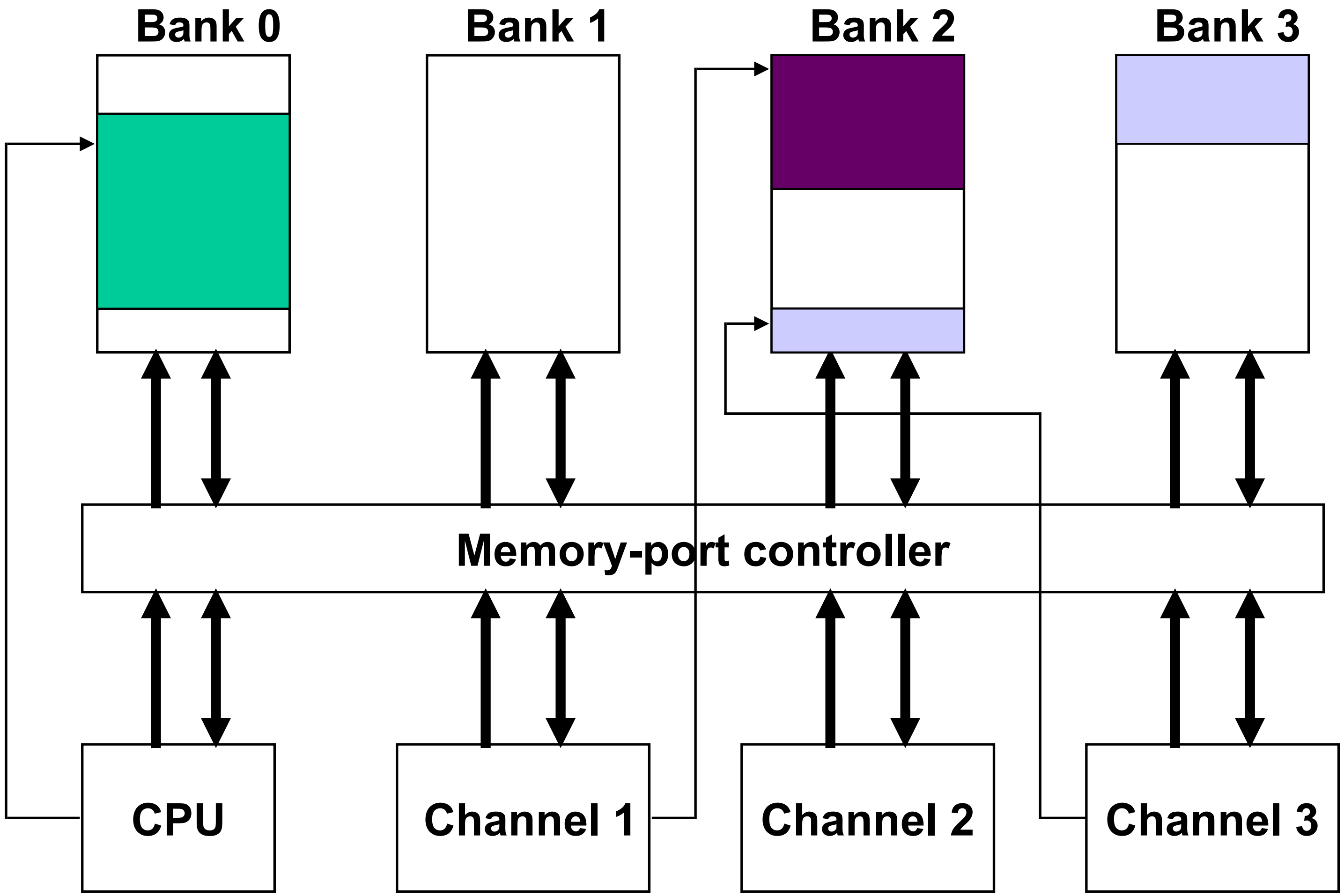
High order interleave

2 M byte memory assembled using 16 256K x 4 bits RAM chip



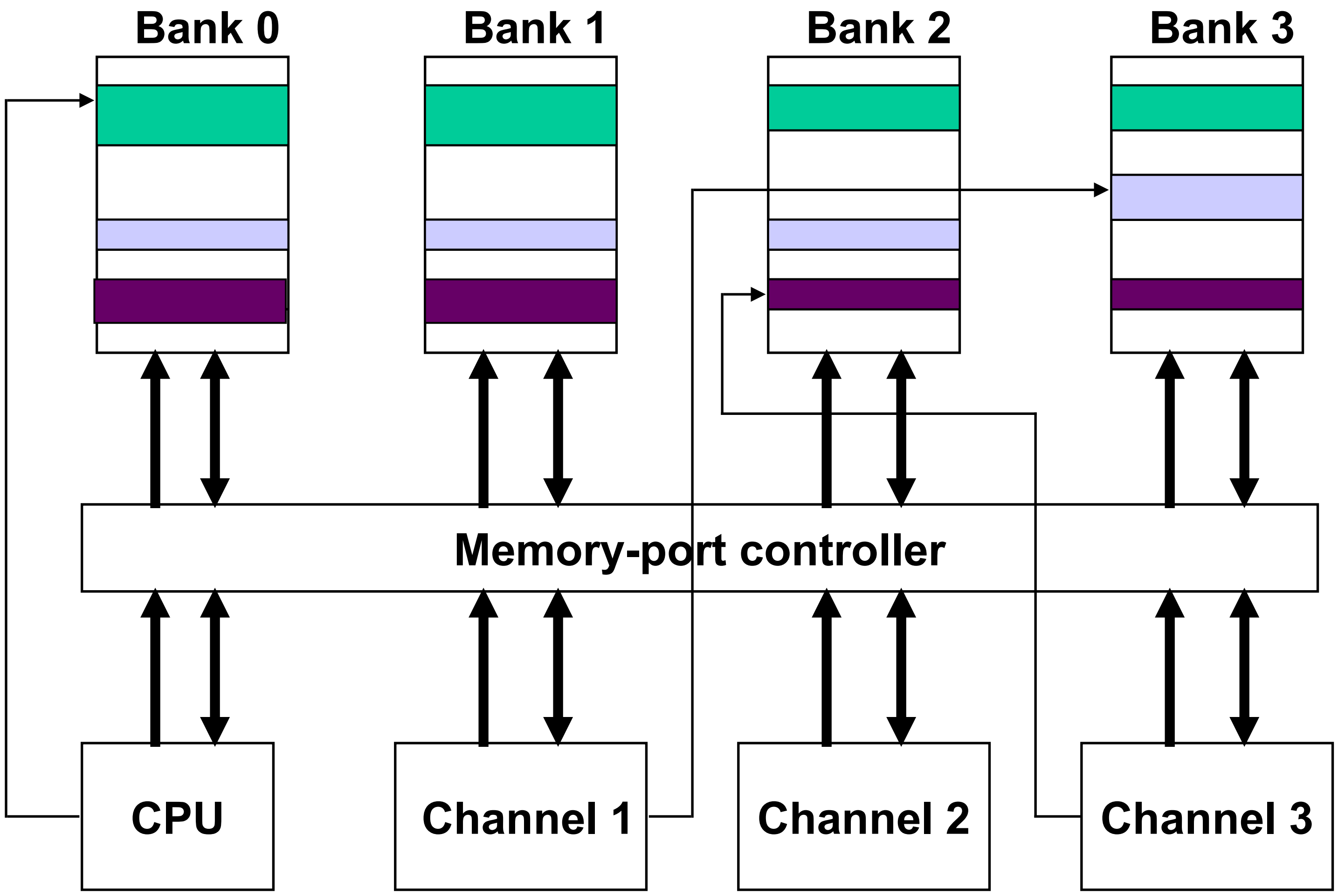
Memory-system Technology and Cost

High-order interleave: A few most significant bits in the address are used to select banks.



Memory-system Technology and Cost

Low-order interleave: A few least significant bits in the address are used to select banks.



Type of memory

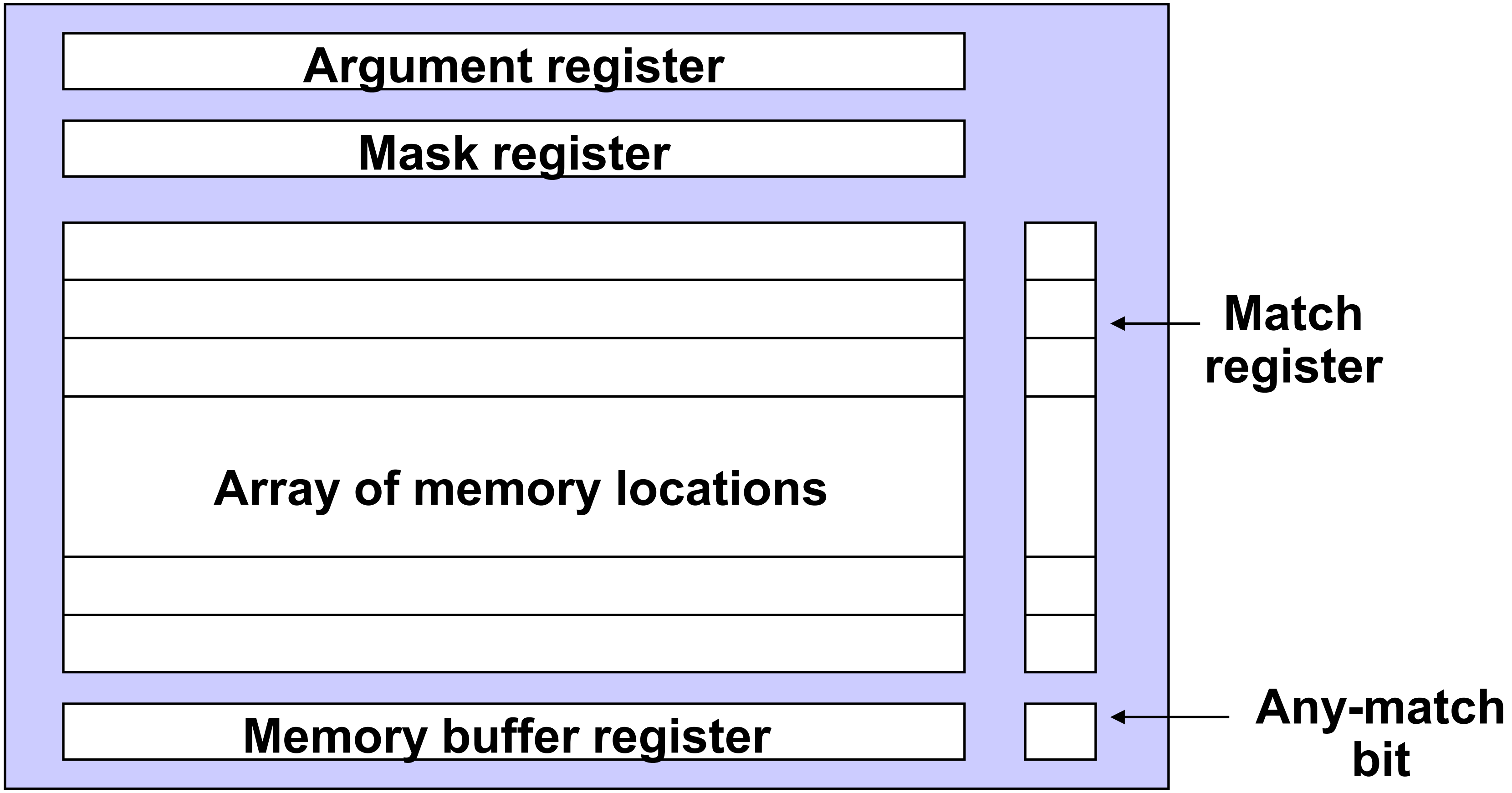
- Non volatile
  - ROM: read-only memory
  - PROM: programmable read-only memory
  - EPROM: erasable programmable read-only memory
  - EEPROM: electrically erasable programmable read-only memory
  - Flash memory
    - High density NAND type: must be programmed and read in blocks
    - NOR type allows a single machine word (byte) to be written and/or read independently

Type of memory

- Volatile: Read/Write Memory: Random-Access Memory (RAM)
  - Static RAM (SRAM)
    - Composed of flip-flops - nondestructive read.
    - Fastest and most expensive.
  - Dynamic RAM (DRAM)
    - Composed of capacitors - refresh circuits needed.
    - Destructive read (Cycle time = read cycle + restore cycle).
    - Slower and less expensive.
  - Synchronous dynamic random access memory (SDRAM):
    - DRAM that is synchronized with the system bus
  - DDR SDRAM: high potential bandwidth because each internal read is actually a row of many thousands of bits
    - Reads or writes two words of data per clock cycle
  - DDR2, DDR3 – 4 and 8 read/write

Associative Memory (Content-addressable memory)

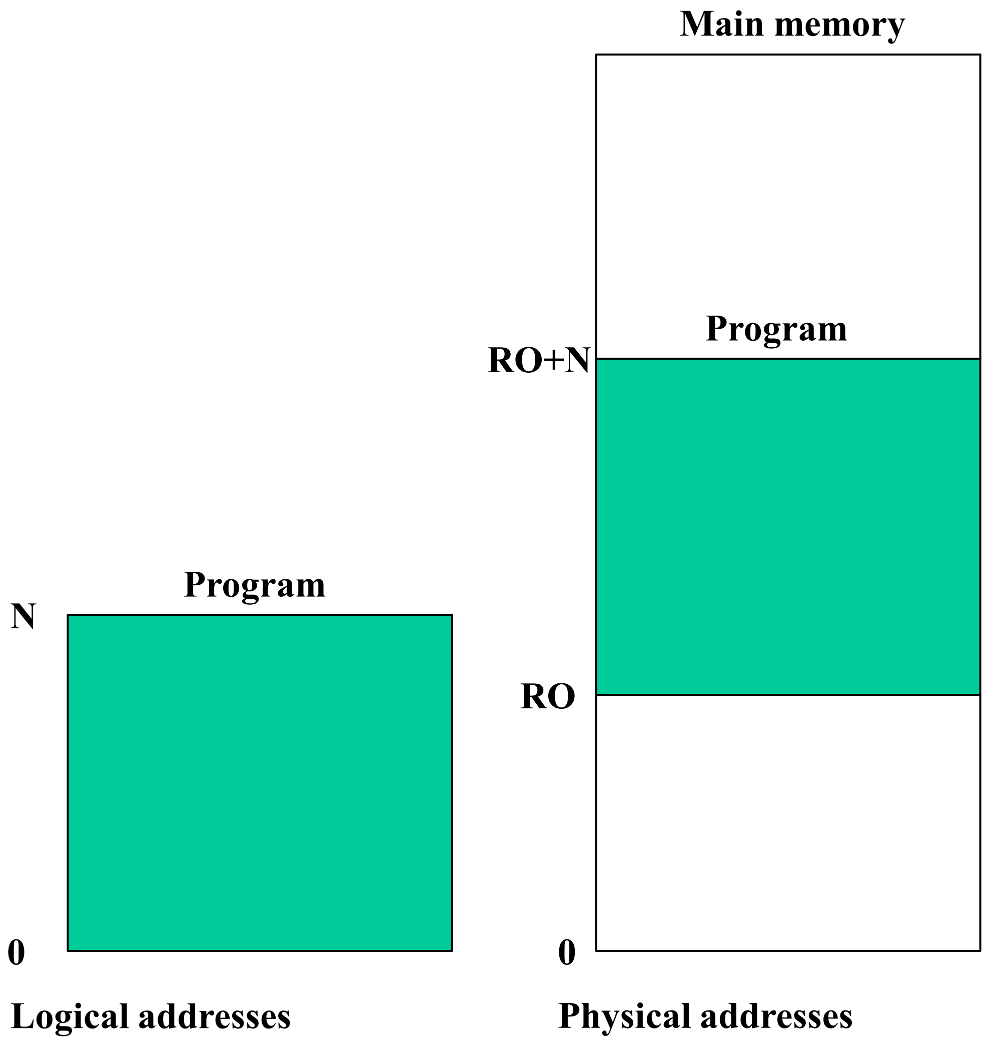
- The contents of part of the memory words are used to select the cells being read or written.



Sequential-Access Memory

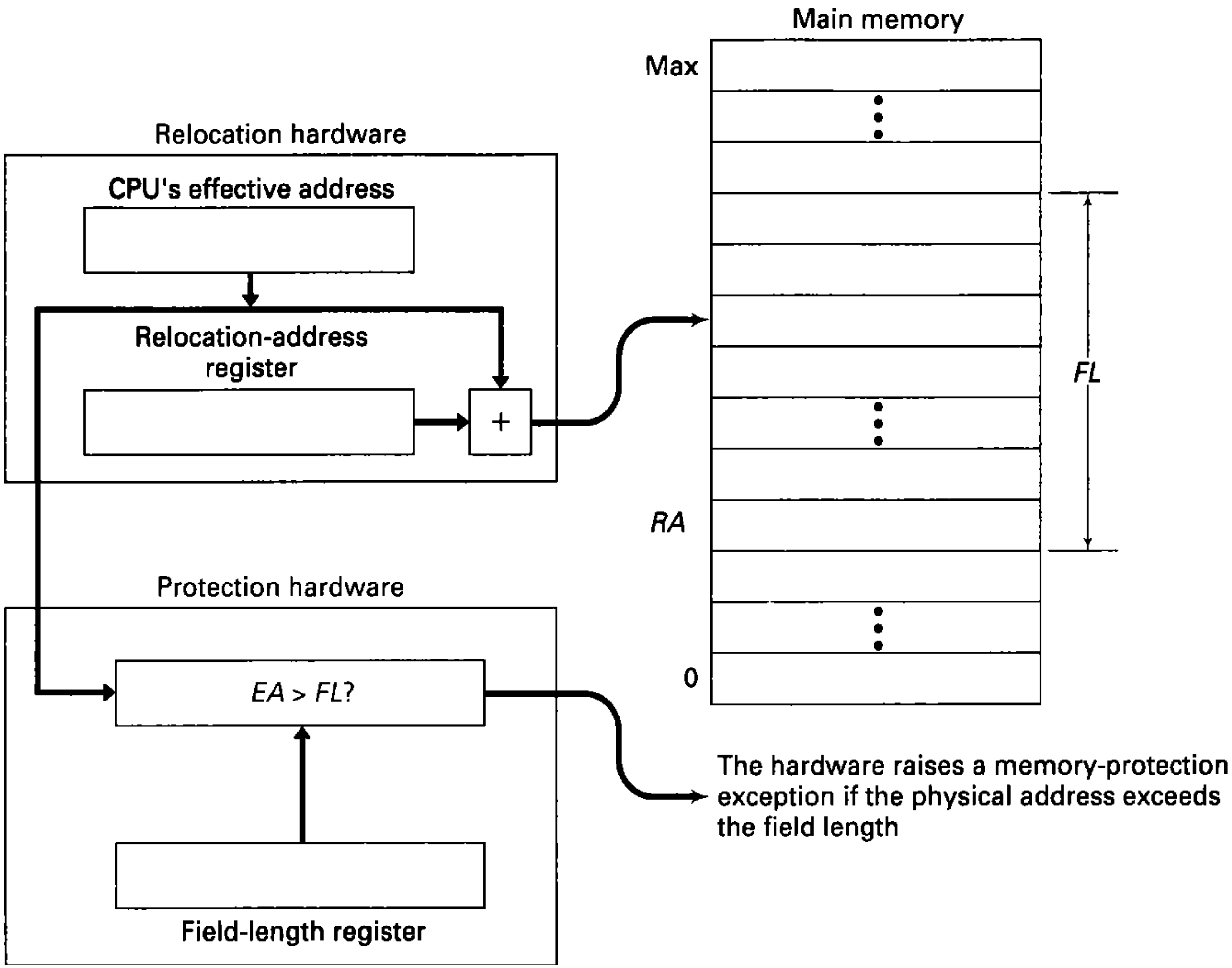
- Magnetic tapes:
  - Data are accessed by specifying its offset from current position of the read/write head.
  - Magnetic tapes can be used as Archival Memory, which is a nonvolatile memory for holding a lot of data at very little cost for a very long time.
- Other common archival memories are magnetic disks and optical disks. Some optical disks can only be written once and are called WORM (write once, read many times) memories.

Main memory


Main memory
Program
RO+N
Program
N
RO
0
0
Logical addresses
Physical addresses

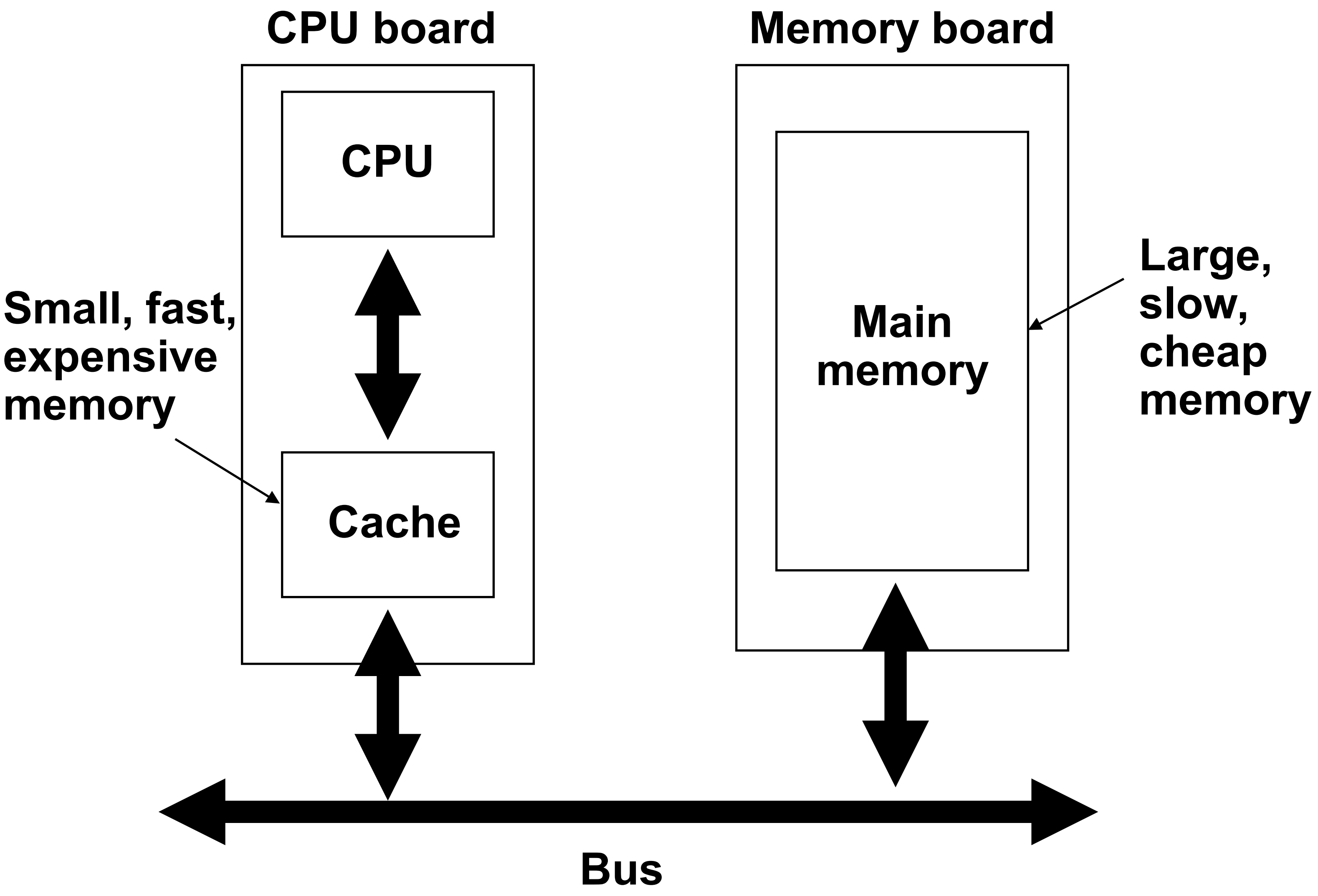
Relocation and protection hardware

The hardware adds the relocation address to the effective address (EA) to get the physical address of the reference



The hardware raises a memory-protection exception if the physical address exceeds the field length

Cache Memory



Cache Memory

- Why can cache be used to improve the performance of a computer ?
  - Principle of locality of reference:
    - The observation that the instructions and data used sequentially by a program tend to be clustered in memory.
    - Programs tend to execute instructions in sequence and hence in nearby memory locations.
    - Programs often have loops in which a group of nearby instructions is executed repeatedly.
    - Most compilers store arrays in blocks of adjacent memory locations and programs frequently access array elements in sequence.
    - Compilers often place unrelated data items in data segments. Local variables are often placed in stacks.

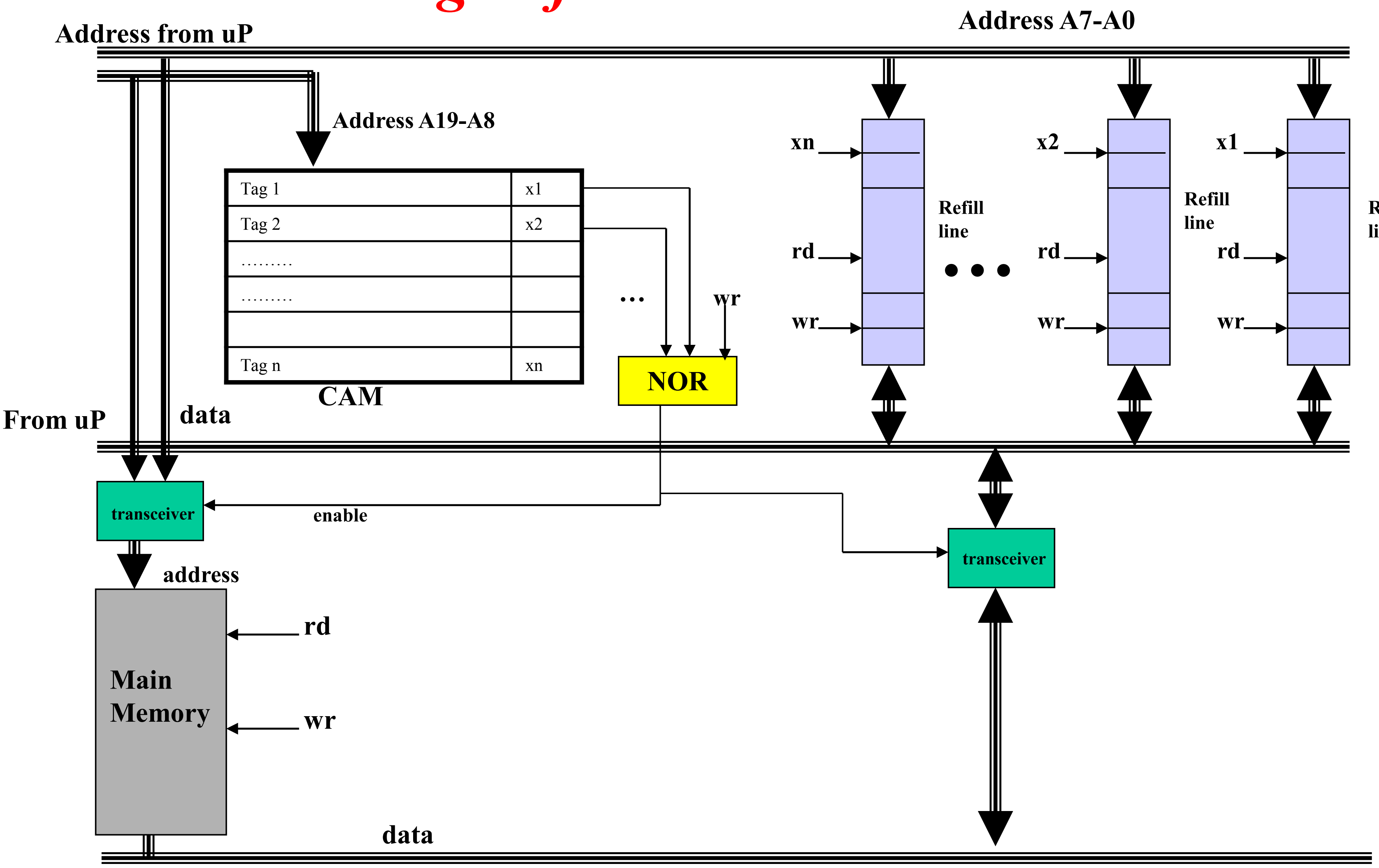
How does a cache work?

- When the CPU initiates a memory access, a physical address is sent to the cache.
- The cache compares the physical address with all (or some) of its address tags to see if it holds a copy of the datum.
- If the operation is a read access
  - and the cache holds the datum, it is a cache hit (also called a read hit in the case of a read operation) and the datum is read from the cache.
  - If it is a cache miss (a read miss), the cache passes the address to the main-memory to read the datum.
- When the datum arrives from the main memory, both the CPU and the cache receive a copy. The cache stores its copy with the appropriate address tag.
- While the CPU works, the cache concurrently reads additional data from nearby main-memory cells and stores them in the cache.

Why can caches work so well?

- The speed of cache is faster than that of main memory and the utility of cache increases with the ratio of CPU speed to main memory speed.
- In a read miss, the cache works in parallel with the CPU, that is, it loads additional words from the main memory while the CPU is executing the instruction. The new data becomes immediately available for the CPU at cache speed.
- Because of the principle of locality of reference, the CPU is likely to request these new data soon.

Design of a Cache Controller

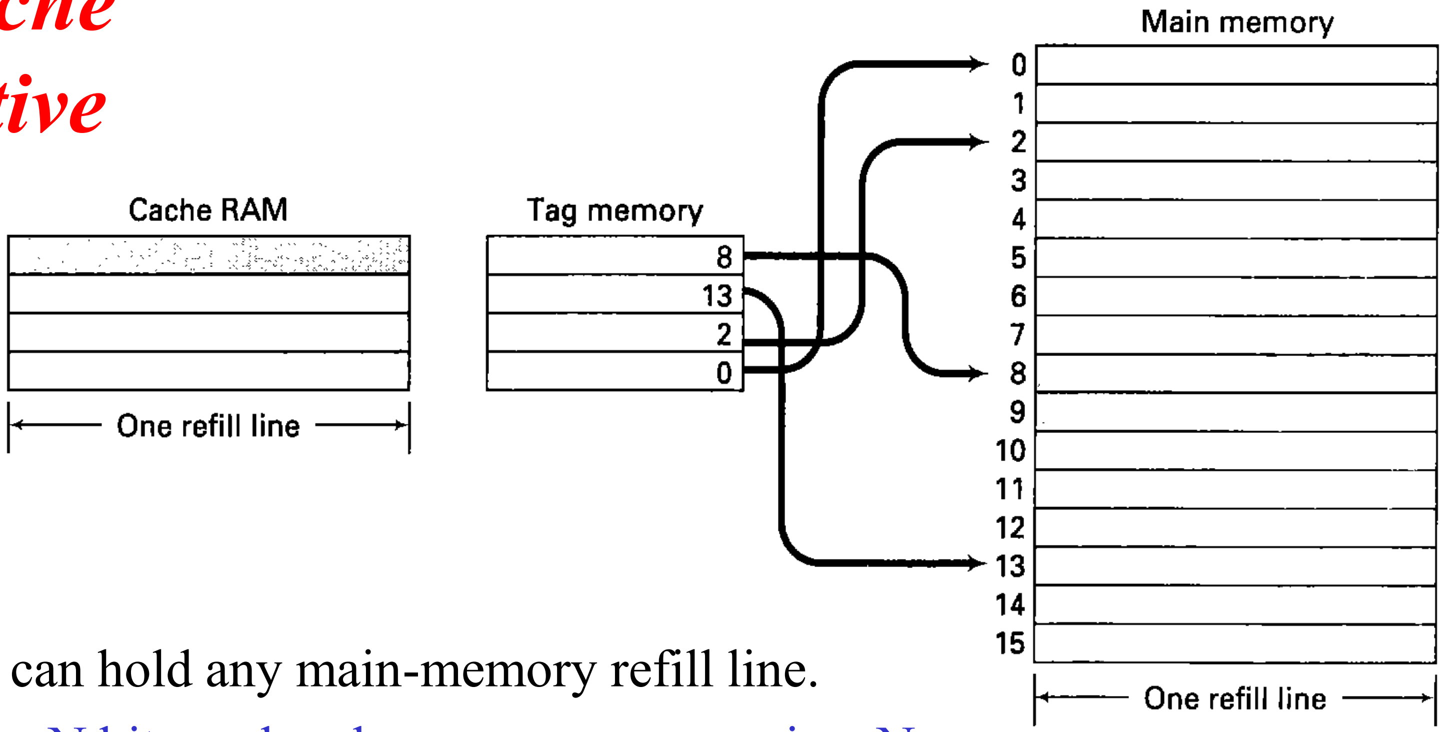


Cache Structure and Organization

- Tag subsystem: Holding the addresses and determines whether there is a match for a requested datum
  - Memory subsystem: Holding the data
  - Refill lines: A unit to divide main memory and cache, usually containing 4 to 64 bytes

- Four common cache organizations:
  - Associative Cache
  - Direct-mapped Cache
  - Set-associative Cache
  - Sector-mapped Cache

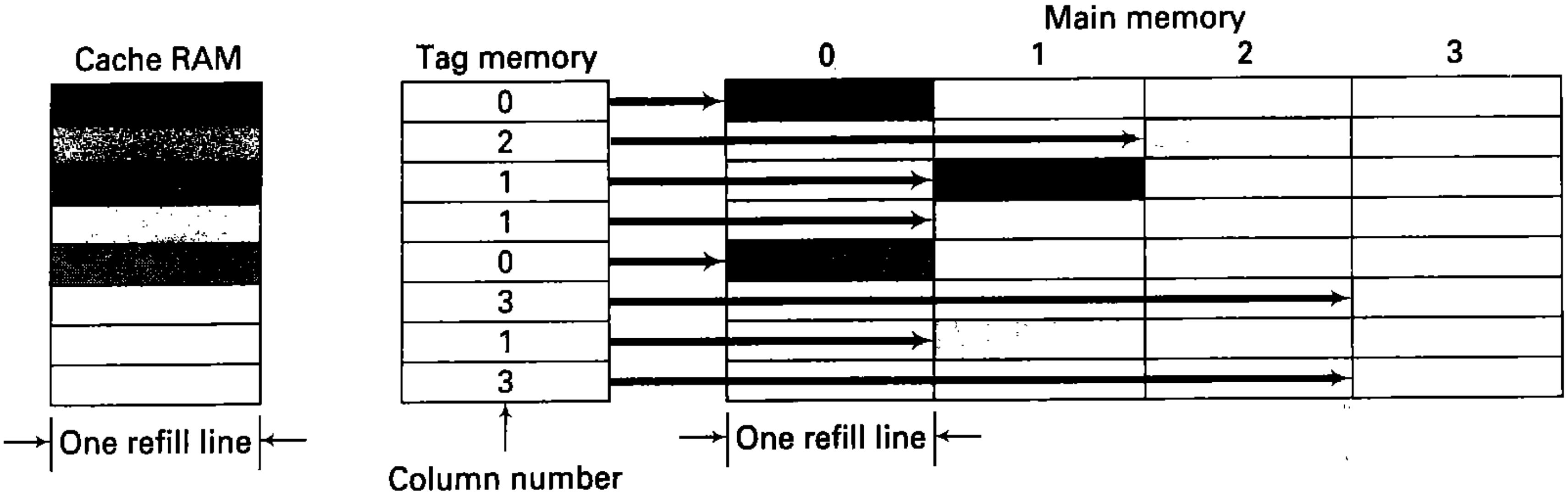
Associative Cache (Fully associative cache)



- Any cache refill line can hold any main-memory refill line.
- Address tags have $\log_2 N$ bits each, where memory comprises N refill lines.
- Number of comparisons: Depending on the number of refill lines of the cache.
- All comparisons occur simultaneously.
- The comparison hardware is expensive and slow for a large cache.
- High hit rate can be expected.

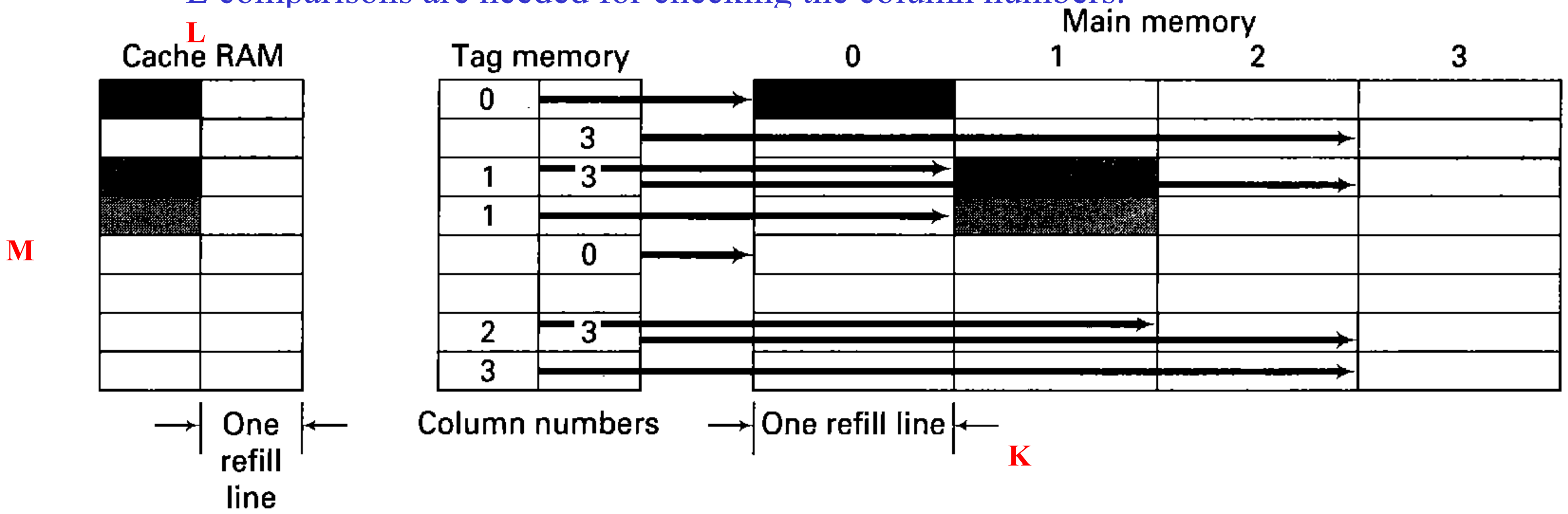
Direct-mapped Cache

- The cache has M refill lines.
- The main memory is partitioned into K columns of M refill lines per column.
- The Jth refill line in the cache can only hold the Jth refill line from any column of the main memory.
- Due to this direct mapping, the part of the address for a refill line within a column can be used to locate the refill line in the cache.
- The address tag ($\log_2 K$ bits) holds the column number of the current refill line.
- One comparison is needed for checking the column number.
- It is fast and easy to implement.

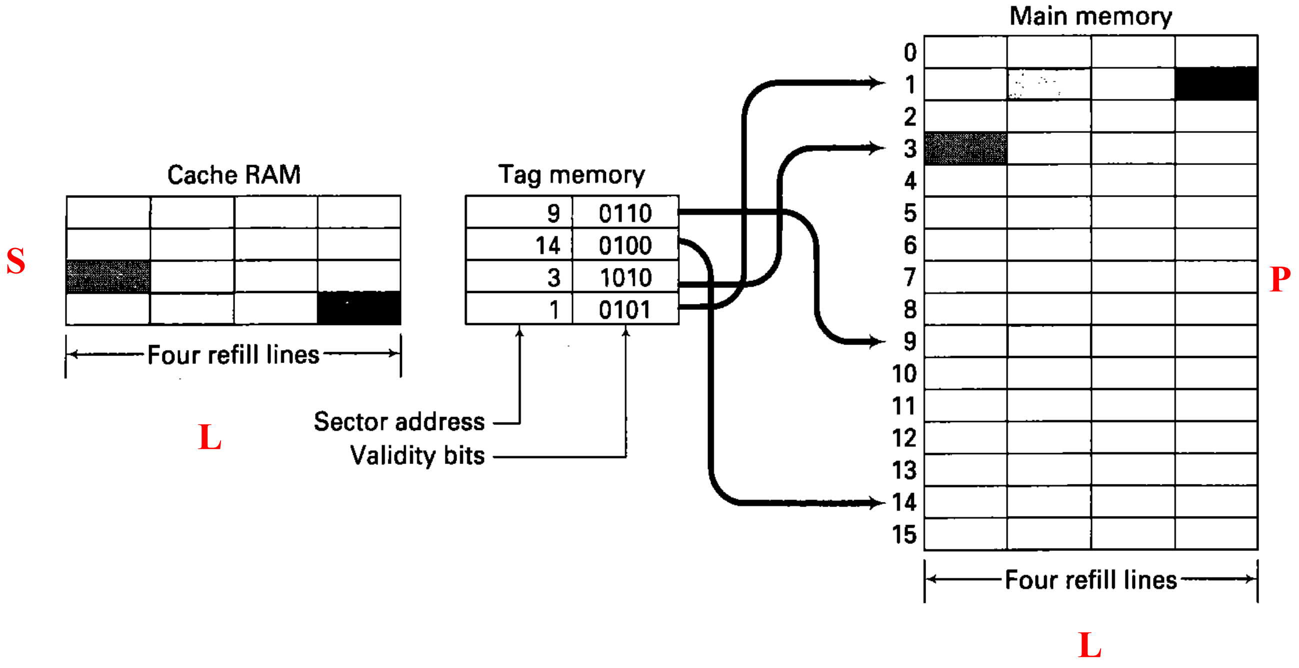


Set-associative Cache (L-way set-associative cache)

- This is a multiple-column, direct mapping organization.
- The cache has L columns of M refill lines per column.
- Each row of the cache comprises L tags and refill lines.
- The main memory is partitioned into K columns of M refill lines per column.
- The Jth refill line of any column in the cache can hold the Jth refill line from any column of the main memory.
- Due to this direct mapping, the part of the address for a refill line within a column can be used to locate the row in the cache.
- Each address tag ($\log_2 K$ bits) holds the column number of the current refill line.
- L comparisons are needed for checking the column numbers.



Sector-mapped Cache



- The cache has S rows (sectors) of L refill lines per sector.
- Each row (sector) of the cache comprises one tag, L validity bits and L refill lines.
- The main memory is partitioned into P sectors of L refill lines per sector.
- Any cache sector can hold any main-memory sector.
- Address tags have $\log_2 P$ bits each, where memory comprises P sectors.
- The validity bits, one for each refill line, indicate if the refill lines are present in the cache.
- Refill lines always preserve their ordering within sectors.
- Number of comparisons = S.

Cache Performance

- The apparent access time (T) of the memory is the weighted average of the cache access time (Tc) and the main-memory access time (Tm).
- Let h represent hit rate. Then miss rate = (1 - h).
- $T = Tc + (1 - h)Tm$
- Cache hit rates exceeding 90% are common for current cache memories.
- Many factors affect the hit rate of the cache, such as the size of the cache, the replacement strategy, the length of the refill line, the workload on the computer, the operating system strategies, the compilers, and the structure of the cache.

Cache coherence problem and cache consistency protocols

- In the case of a write hit, if only the value in the cache is updated, the cache value is then different from the value in the main memory. If cache coherence should be maintained, the datum should also be written to the main memory. Depending on when to do this, we can distinguish different cache consistency protocols.

- Write-through cache
  - When the cache is changed, the datum is also written to the main memory.

- Write-back cache
  - A write-back cache does not update the main memory whenever the cache is changed. The main memory is only updated when the refill line is purged from the cache to allow another refill line to take over its place.

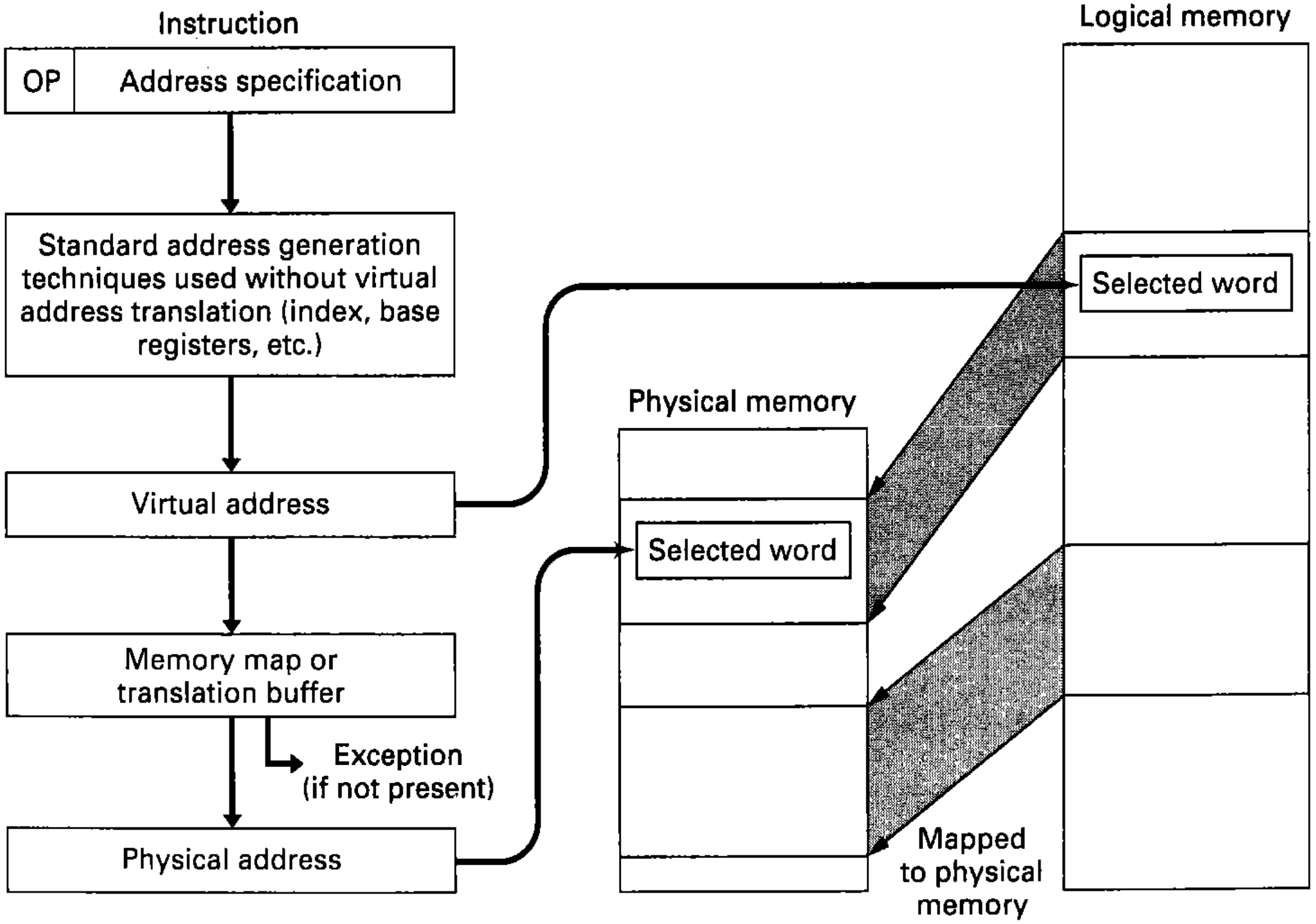
Specialized Caches

- The access pattern of data is different from that of the instructions. Using separate caches allows the designer to put each one close to its user for fast response, and tailor the length of refill line and replacement strategies.

- Instruction cache
  - Only read operations are allowed.
  - It performs better with large refill lines than those optimal for data caches.
  - Direct-mapped cache can be used to implement it.

- Data cache
  - Two-way or four-way set associative caches with refill lines of about 64 bytes tend to be the best for data caches.

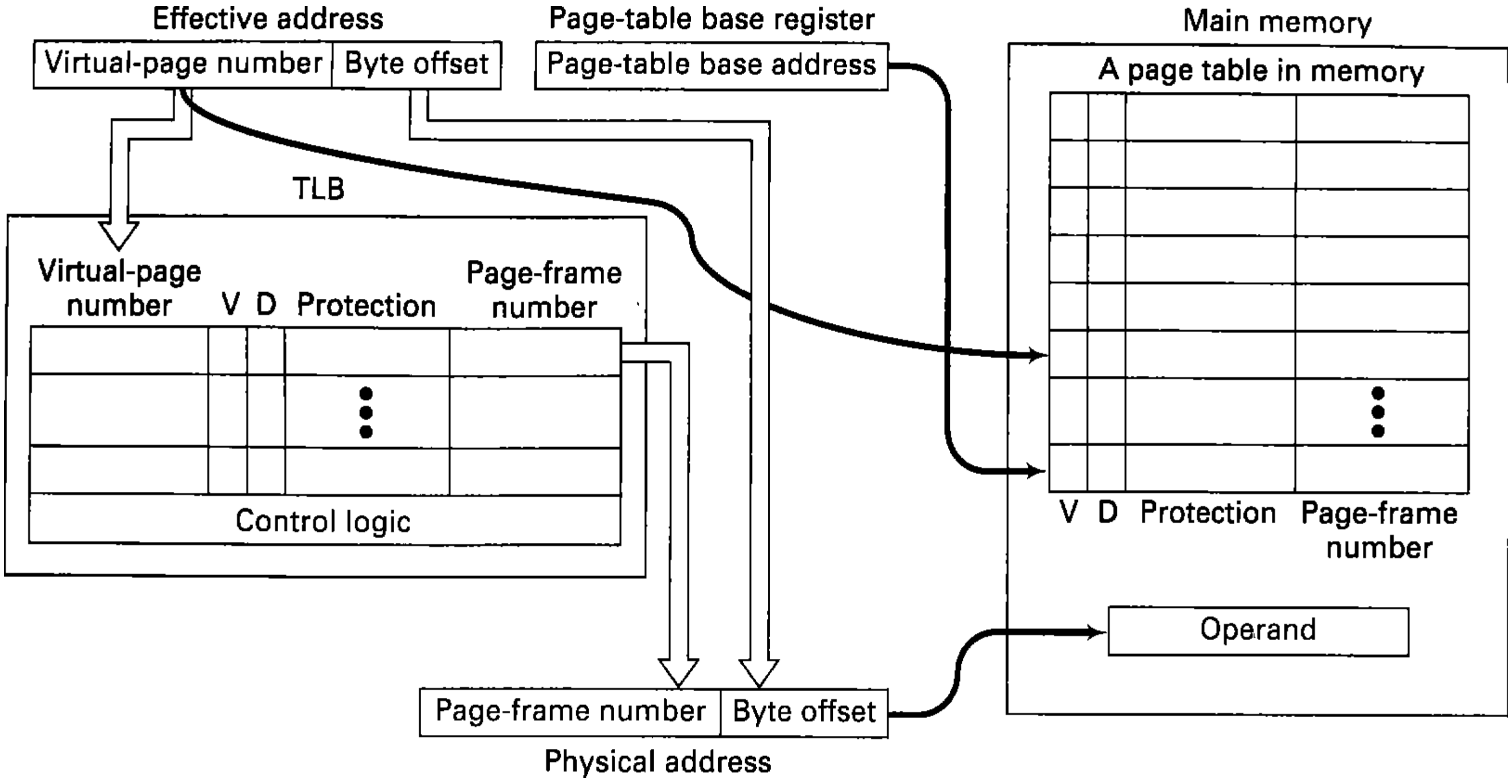
Virtual memory system


Instruction
OP
Address specification
Standard address generation techniques used without virtual address translation (index, base registers, etc.)
Virtual address
Memory map or translation buffer
Exception (if not present)
Physical address
Physical memory
Selected word
Logical memory
Selected word
Mapped to physical memory

Paging

- Logical address space is divided into pages of fixed size.
- Main memory (physical address space) is divided into fixed-size page frames.
- In a paging system, the virtual-memory hardware divides logical addresses into two parts, a virtual-page number (the high-order bits) and a word offset (the low-order bits) within the page.
- Components of a paging system:
  - Page table
  - Page-table base register
  - Translation lookaside buffer (TLB)

How to access a page?



- 1. The CPU sends an effective address to the TLB.
- 2. If the TLB holds an entry for the page, it will produce the page-frame number.
- 3. If the TLB has no entry, the hardware consults the page table in the main memory by using the page number as an offset into the page table.
- 4. If the validity bit indicates the page is in memory, the hardware uses the page-frame number to access the memory and simultaneously copies the page-table entry into the TLB.
- 5. Otherwise the hardware initiates a trap called a page fault.
- 6. Then the operating system loads the demanded page in memory and update the page table.

Paging

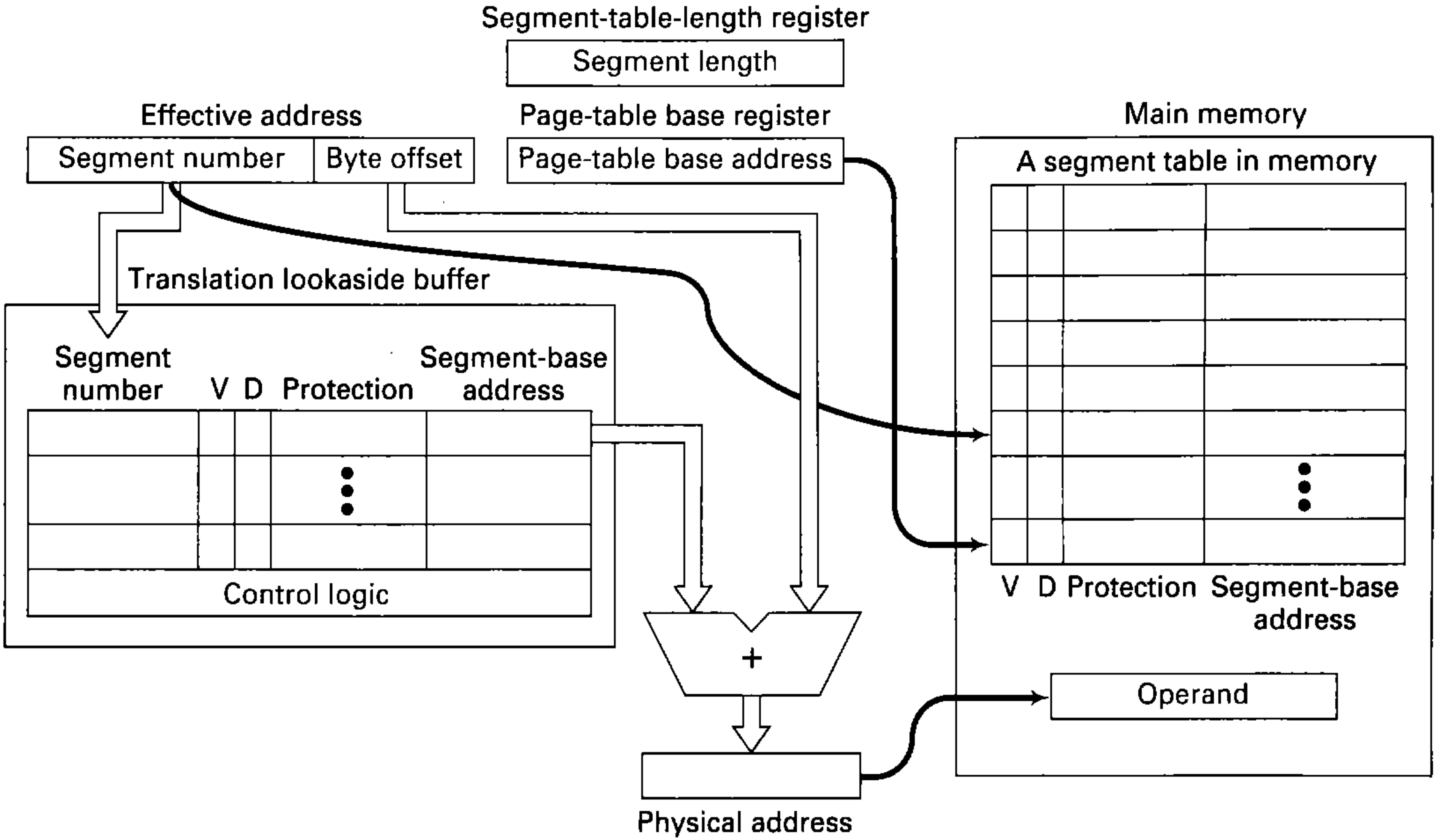
- Delayed page fault:
  - A page fault which occurs during the middle of an instruction execution. Example: MOVE STRING instruction.

- How to handle it?
  - 1. The hardware checks for the validity of every page that the instruction will use before starting execution of the instruction.
  - 2. Roll back: The hardware records enough information during the execution of the instruction so that if a page fault occurs, it can restore the system to the state that existed prior to instruction execution. After processing the page fault, the hardware can restart the instruction at its beginning.
  - 3. The hardware can interrupt the instruction in the middle of execution.

Segmentation

- Logical address space is divided into segments of arbitrary size.
- Main memory is treated as a single block by the processor.
- Types of segments: Code segments, data segments, ...
- Memory protection can be provided easily.
- Logical addresses have two parts: a segment number and a byte offset.
- Components of a segmentation system:
  - Segment tables
  - Segment-table base register
  - Segment-table-length register
  - Translation lookaside buffer (TLB)
  - Adder

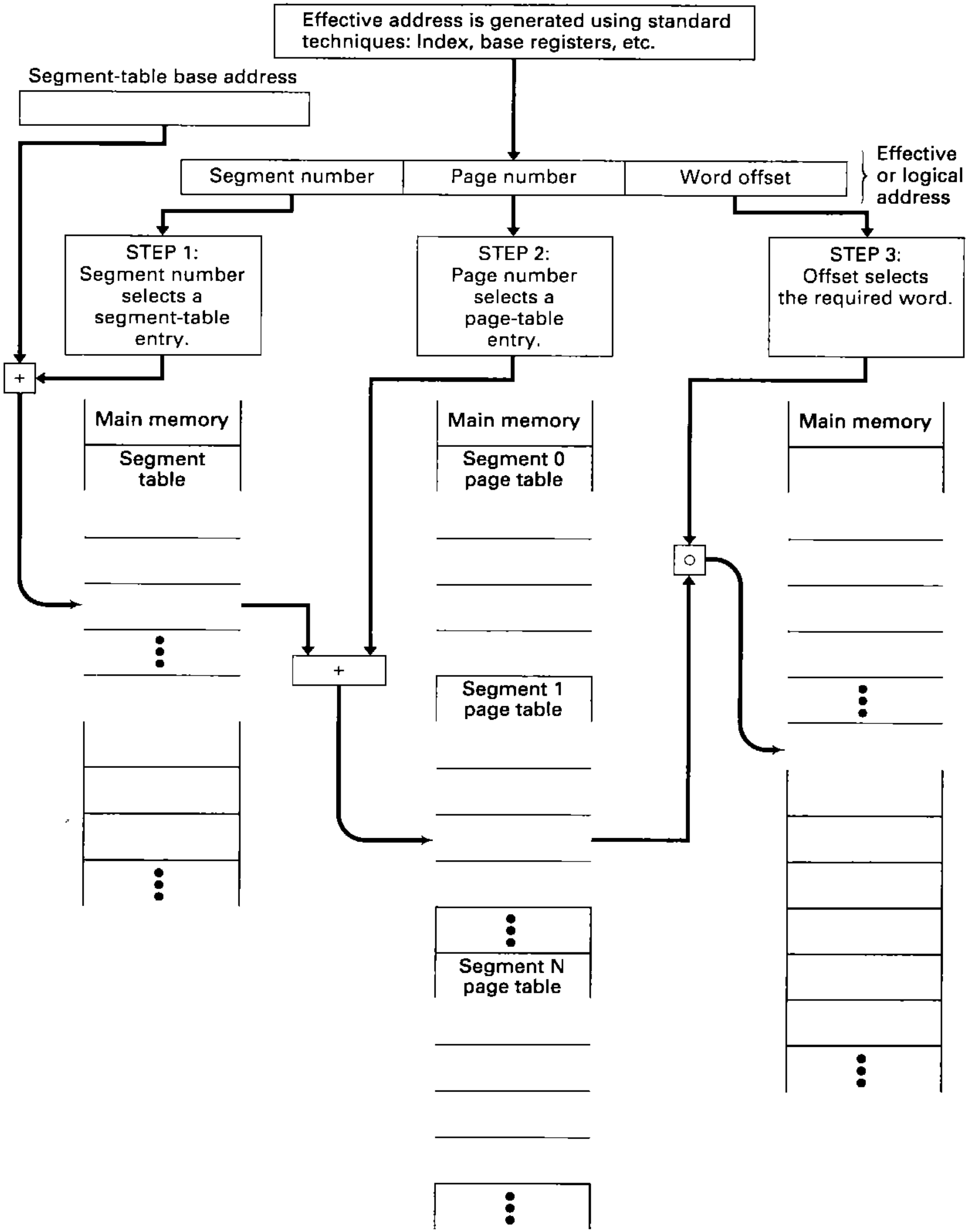
Segmentation: procedure



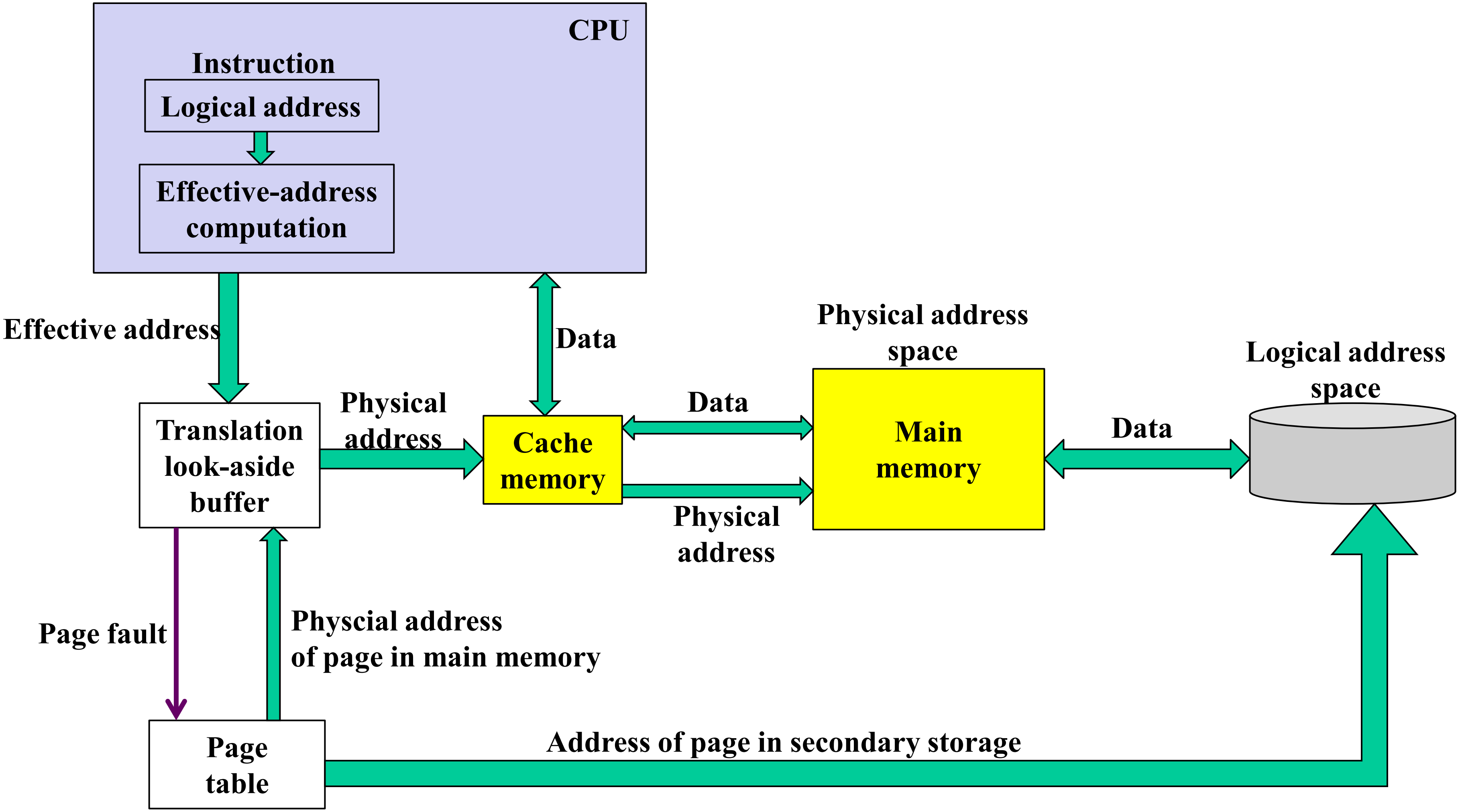
Segmentation with paging

- Logical address space is divided into segments.
- Each segment is divided into pages.
- Main memory is divided into page frames.
- The hardware divides the logical address into a segment number, page number, and word offset.
- Components of the page-segmented system:
  - Segment table
  - Segment-table base register
  - Page tables for segments
  - Adder

Paged segmentation


Effective address is generated using standard techniques: Index, base registers, etc.
Segment-table base address
Segment number
Page number
Word offset
Effective or logical address
STEP 1: Segment number selects a segment-table entry.
STEP 2: Page number selects a page-table entry.
STEP 3: Offset selects the required word.
+
Main memory
Segment table
+
Main memory
Segment 0 page table
Segment 1 page table
Segment N page table
○
Main memory

Cache plus Virtual Memory


CPU
Instruction
Logical address
Effective-address computation
Effective address
Data
Translation look-aside buffer
Physical address
Cache memory
Data
Physical address
Physical address space
Main memory
Data
Logical address space
Page fault
Physcial address of page in main memory
Page table
Address of page in secondary storage