William Stallings
Computer Organization and Architecture
7th Edition

Chapter 5

Memory

Syllabus

Characteristics of memory system and hierarchy, main memory ,ROM, Types of ROM, RAM, SRAM, DRAM, Flash memory, High speed memories

Cache Memory Organization: Address mapping, Replacement Algorithms, Cache Coherence, MESI protocol, Interleaved and associative memories, virtual memory, main memory allocation, segmentation paging, secondary storage ,RAID levels

Characteristics of Memory

- Location
- Capacity
- Unit of transfer
- Access method
- Performance(SRAM,DRAM)
- Physical type
- Physical characteristics
- Organisation
  - —Direct Mapping,Associative Mapping

Location

- CPU
- Internal
- External

Capacity

- Word size
  - —The natural unit of organisation
- Number of words
  - —or Bytes

Unit of Transfer

- Internal
  - Usually governed by data bus width
- External
  - Usually a block which is much larger than a word
- Addressable unit
  - Smallest location which can be uniquely addressed

Access Methods (1)

- **Sequential**
  - —Start at the beginning and read through in order
  - —Access time depends on location of data and previous location
  - —e.g. tape

- **Direct**
  - —Individual blocks have unique address
  - —Access is by jumping to vicinity plus sequential search
  - —Access time depends on location and previous location
  - —e.g. disk

Access Methods (2)

- **Random**
  - —Individual addresses identify locations exactly
  - —Access time is independent of location or previous access
  - —e.g. RAM
- **Associative**
  - —Data is located by a comparison with contents of a portion of the store
  - —Access time is independent of location or previous access
  - —e.g. cache

Performance

- Access time
  - —Time between requesting for operation and the time it is made available at the required location
- Memory Cycle time
  - —Minimum time elapsed between two consecutive read requests
- Transfer Rate
  - —Rate at which data can be moved

Physical Types

- Semiconductor
  - RAM
- Magnetic
  - Disk & Tape
- Optical
  - CD & DVD

Physical Characteristics

- Decay
- Volatility
- Erasable

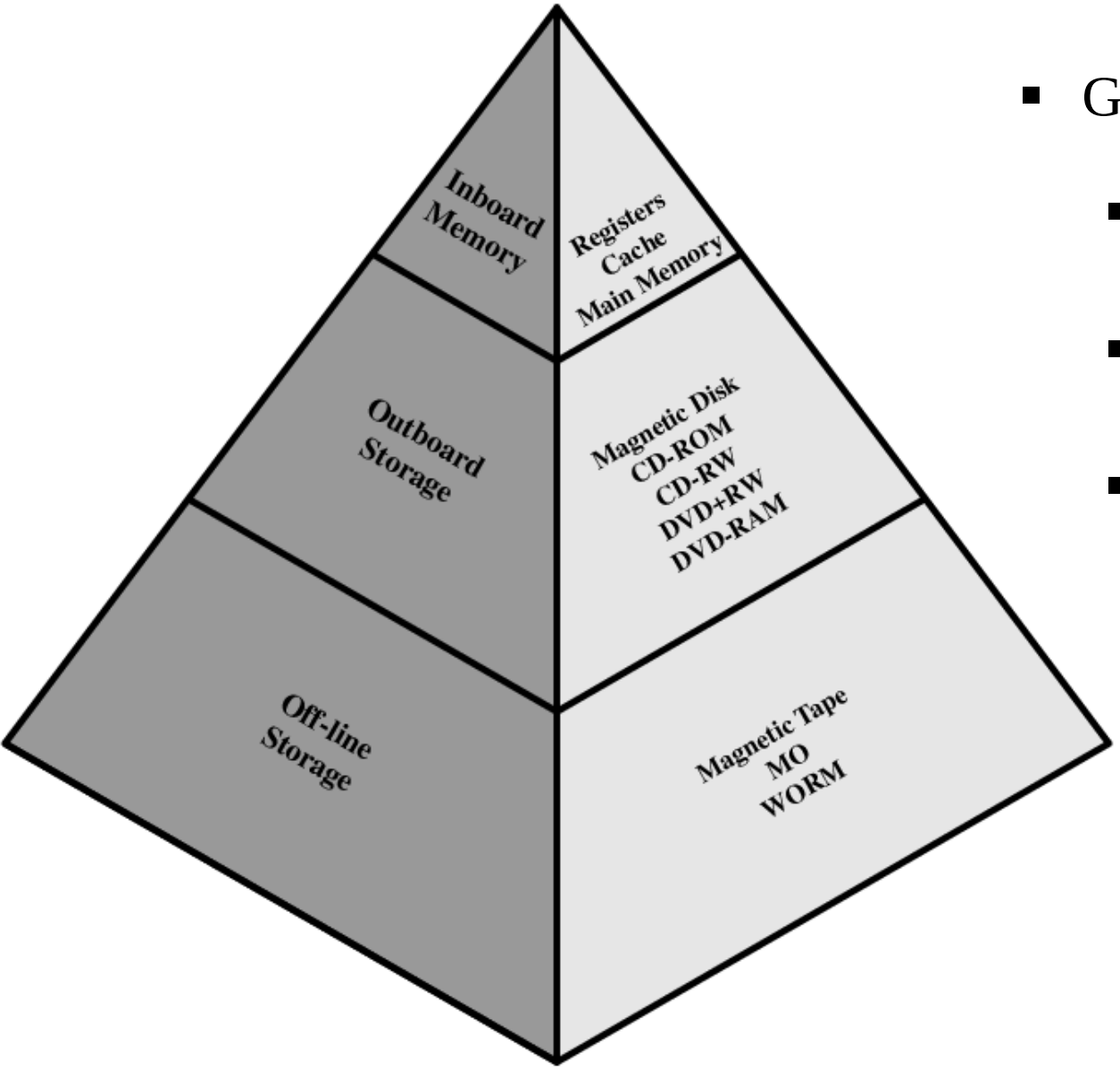
Organisation

- Physical arrangement of bits into words
- Not always obvious
- e.g. interleaved

Memory Hierarchy

- Registers
  - In CPU
- Internal or Main memory
  - May include one or more levels of cache
  - "RAM"
- External memory
  - Backing store

Memory Hierarchy - Diagram

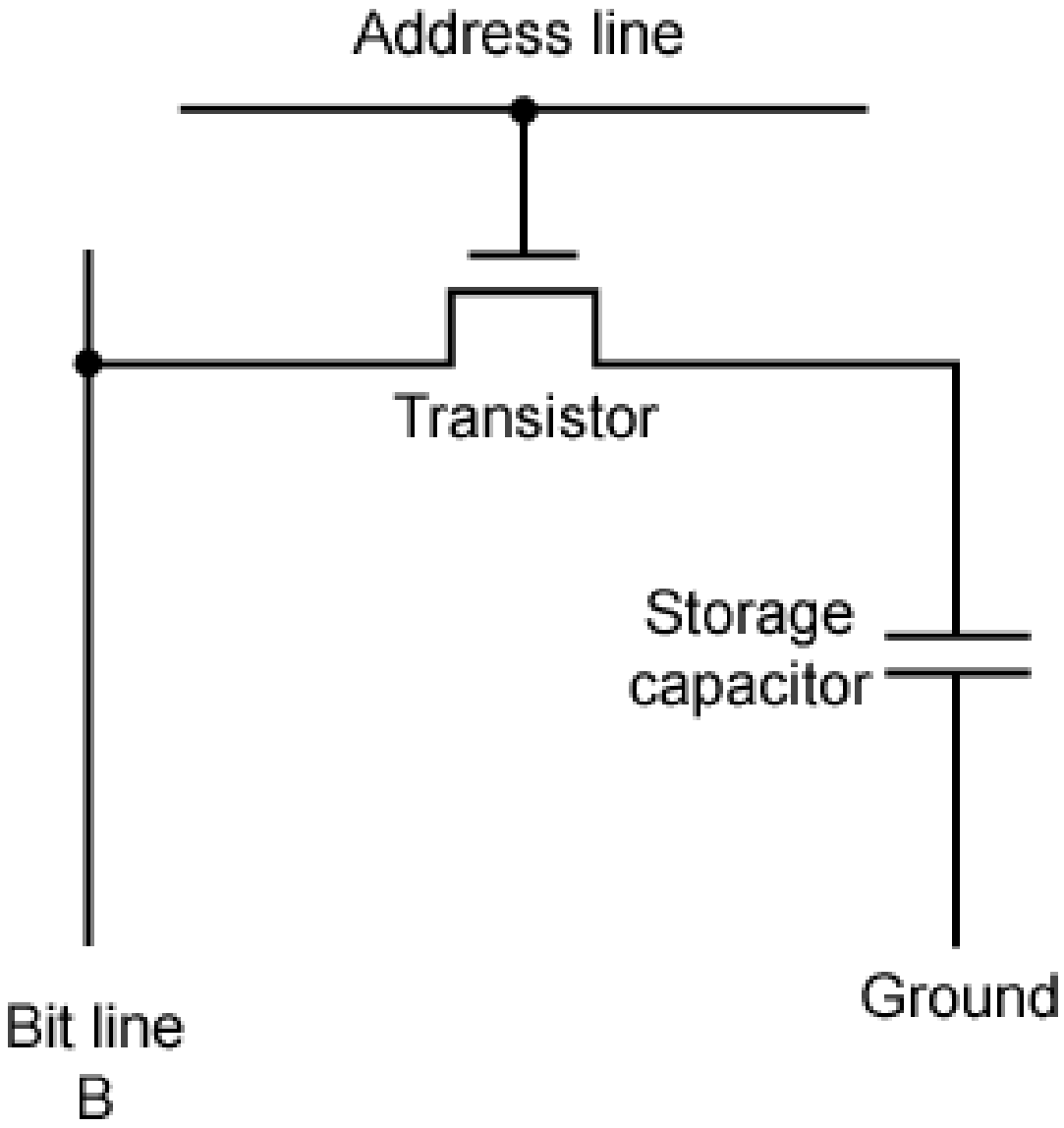


- Going down the hierarchy
  - Decreasing **Cost**
  - Increase **Capacity**
  - Increase **Access Time**

RAM

- RAM
  - —random access
  - —Read/Write
  - —Volatile
  - —Temporary storage
  - —Static or dynamic

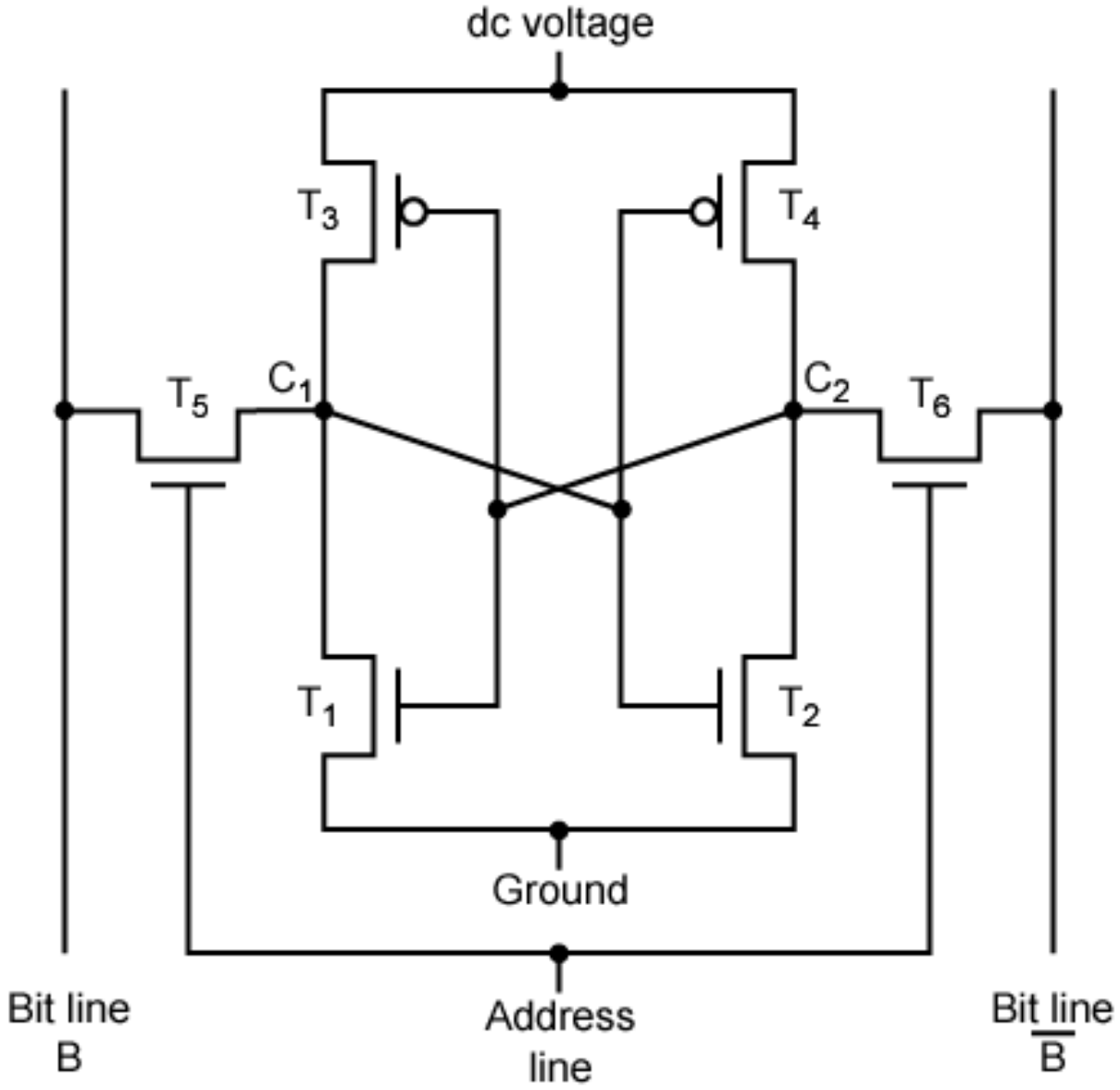
Dynamic RAM Structure



Dynamic RAM

- Bits stored as charge in capacitors
- Charges leak
- Need **refreshing** even when powered
- Simpler construction
- Smaller per bit
- Less expensive
- Need refresh circuits
- Slower
- Main memory
- Essentially analog
  - —Level of charge determines value

Static RAM Structure

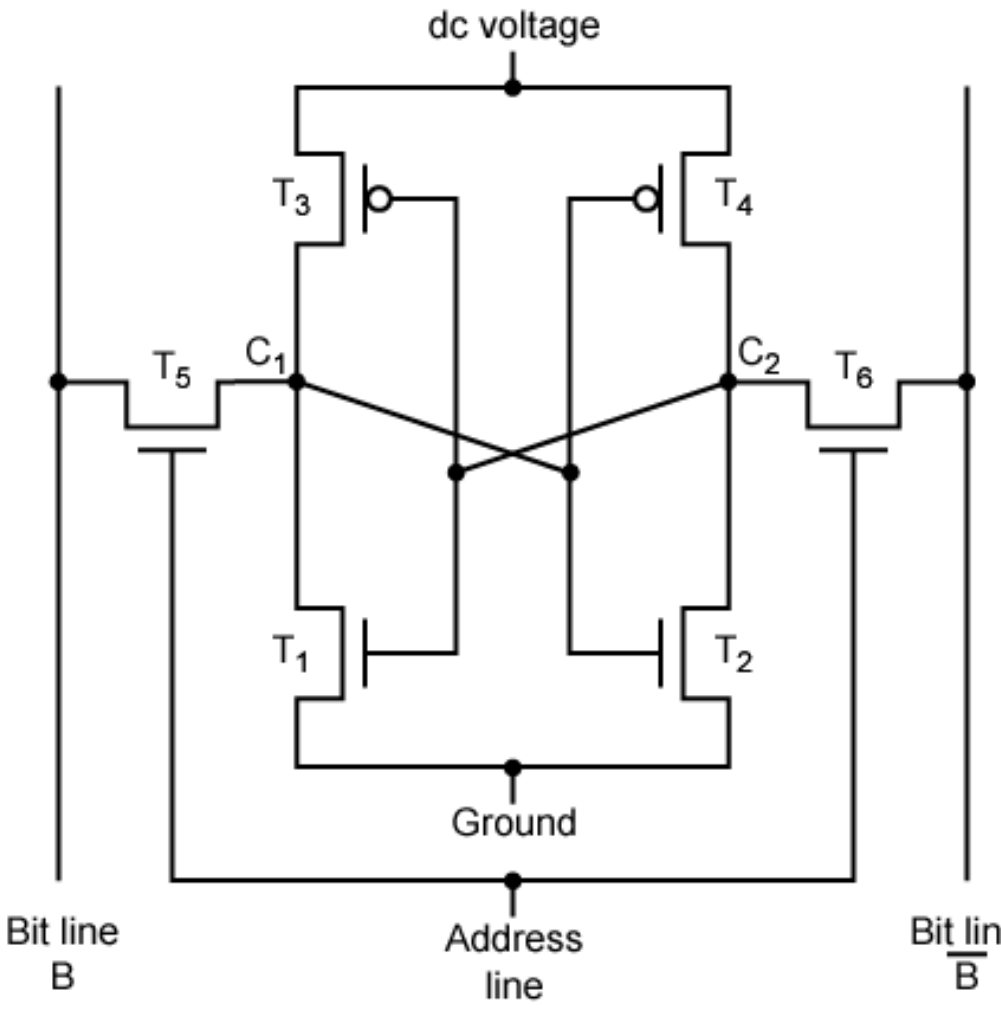


Static RAM

- Bits stored as on/off switches
- No charges to leak
- No refreshing needed when powered
- More complex construction
- Larger per bit
- More expensive
- Does not need refresh circuits
- Faster
- Cache
- Digital
  - —Uses flip-flops

Static RAM Operation

- Transistor arrangement gives stable logic state
- State 1
  - C_1 high, C_2 low
  - T_1 T_4 off, T_2 T_3 on
- State 0
  - C_2 high, C_1 low
  - T_2 T_3 off, T_1 T_4 on



SRAM v DRAM

- Both volatile
  - Power needed to preserve data
- Dynamic cell
  - Simpler to build, smaller
  - More dense
  - Less expensive
  - Needs refresh
  - Larger memory units
- Static
  - Faster
  - Cache

Read Only Memory (ROM)

- Permanent storage
  - —Nonvolatile
  - —Can read a ROM but cant write new data into it
    e.g.Library subroutines, system programs, function tables
- TYPES OF ROM
  - —PROM
  - —EPROM
  - —EEPROM

Types of ROM

- Written during manufacture
  - Very expensive for small runs
- Programmable ("once")
  - **PROM**
  - Small amount of data to be written
  - Less expensive
  - Non volatile, written only once
  - Writing performed electrically at the time of chip fabrication

Read “mostly”

- Erasable Programmable (**EPROM**)
  - Erased by UV
- Electrically Erasable (**EEPROM**)
  - Takes much longer to write than read
- Flash memory
  - Erase whole memory electrically

EPROM

- Read and written electrically
- All storage cells should be erased electrically to initial state by exposure to UV radiation
- Can be altered multiple times and holds data virtually indefinitely
- More expensive than PROM

EEPROM

- Can be written anytime without erasing prior contents
- Write operation takes longer than read
- More expensive than EPROM, less dense

Types of ROM

1. Programmable Read Only Memory (PROM)

- Empty of data when manufactured
- May be permanently programmed by the user

2. Erasable Programmable Read Only Memory (EPROM)

- Can be programmed, erased and reprogrammed
- The EPROM chip has a small window on top allowing it to be erased by shining ultra-violet light on it
- Before write operation ,all storage cells must be erased to the same initial state by exposing it to UV light
- After reprogramming the window is covered to prevent new contents being erased
- Access time is around 45 – 90 nanoseconds

Types of ROM

3. Electrically Erasable Programmable Read Only Memory (EEPROM)

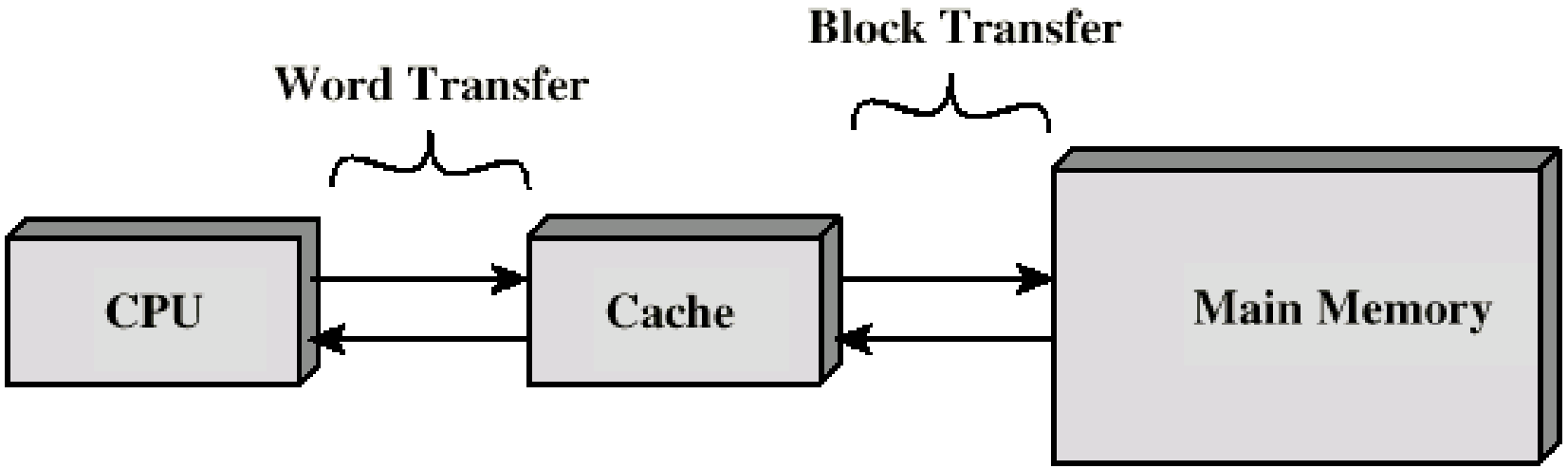
- Reprogrammed electrically **without** using ultraviolet light
- Must be removed from the computer and placed in a special machine to do this
- Written into at any time without erasing prior contents
- Access times between 45 and 200 nanoseconds

4. Flash ROM

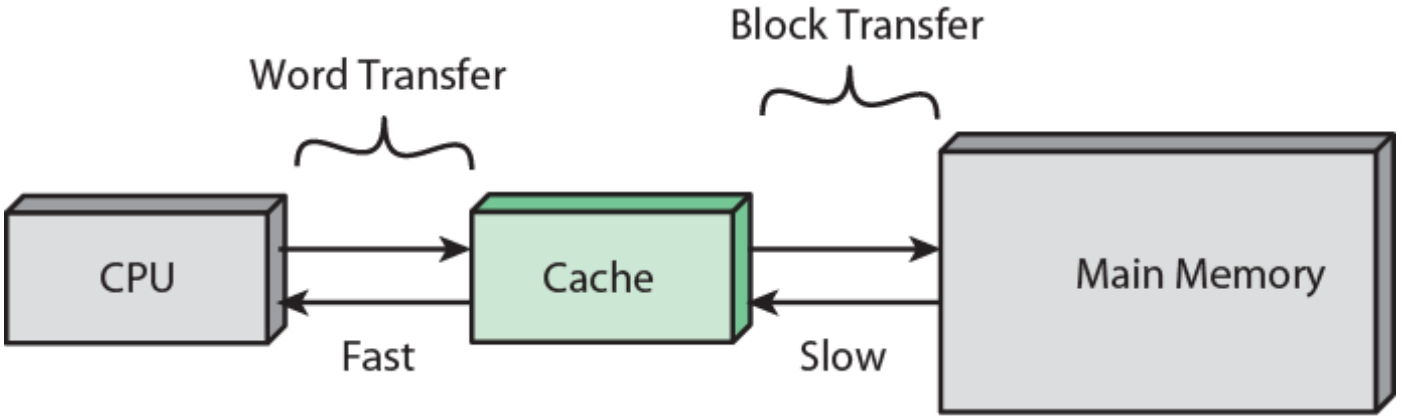
- Similar to EEPROM but speed is different.
- However, can be reprogrammed while still in the computer
- Easier to upgrade programs stored in Flash ROM
- An entire flash memory can be erased in few nanoseconds
- Used to store programs in devices e.g. modems
- Access time is around 45 – 90 nanoseconds

Cache

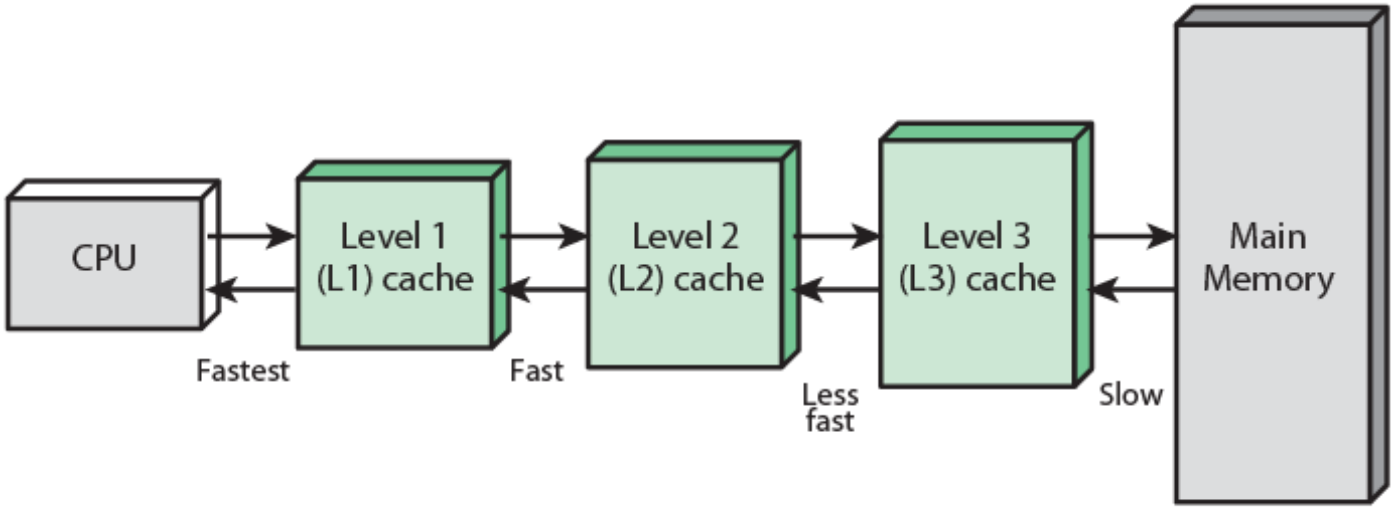
- Small amount of fast memory
- Sits between normal main memory and CPU
- May be located on CPU chip or module



Cache and Main Memory



(a) Single cache

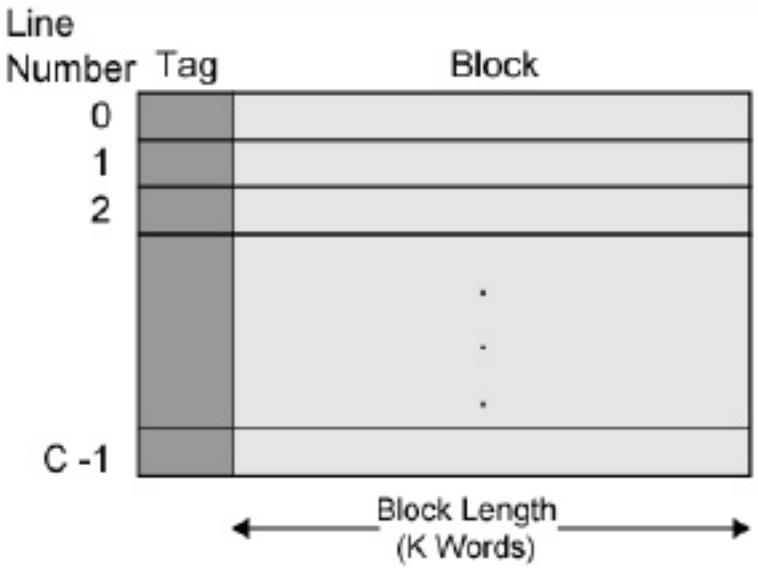


(b) Three-level cache organization

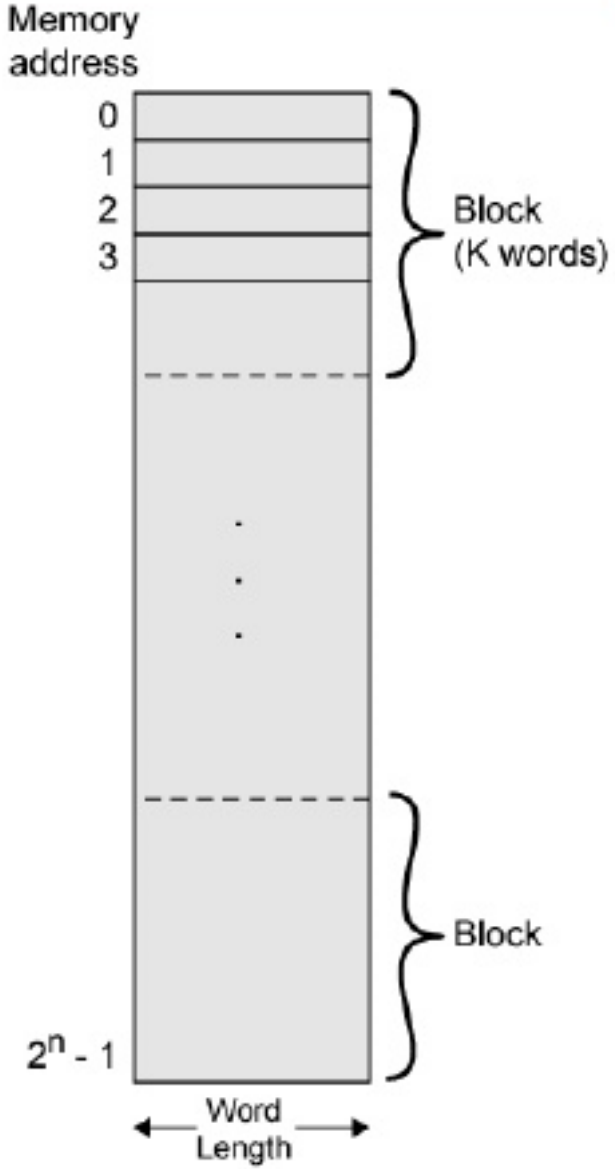
CACHE

- Cache memory is a high speed memory kept in between processor and RAM to increase the data execution speed. It is kept near to the processor.

Cache/Main Memory Structure



(a) Cache



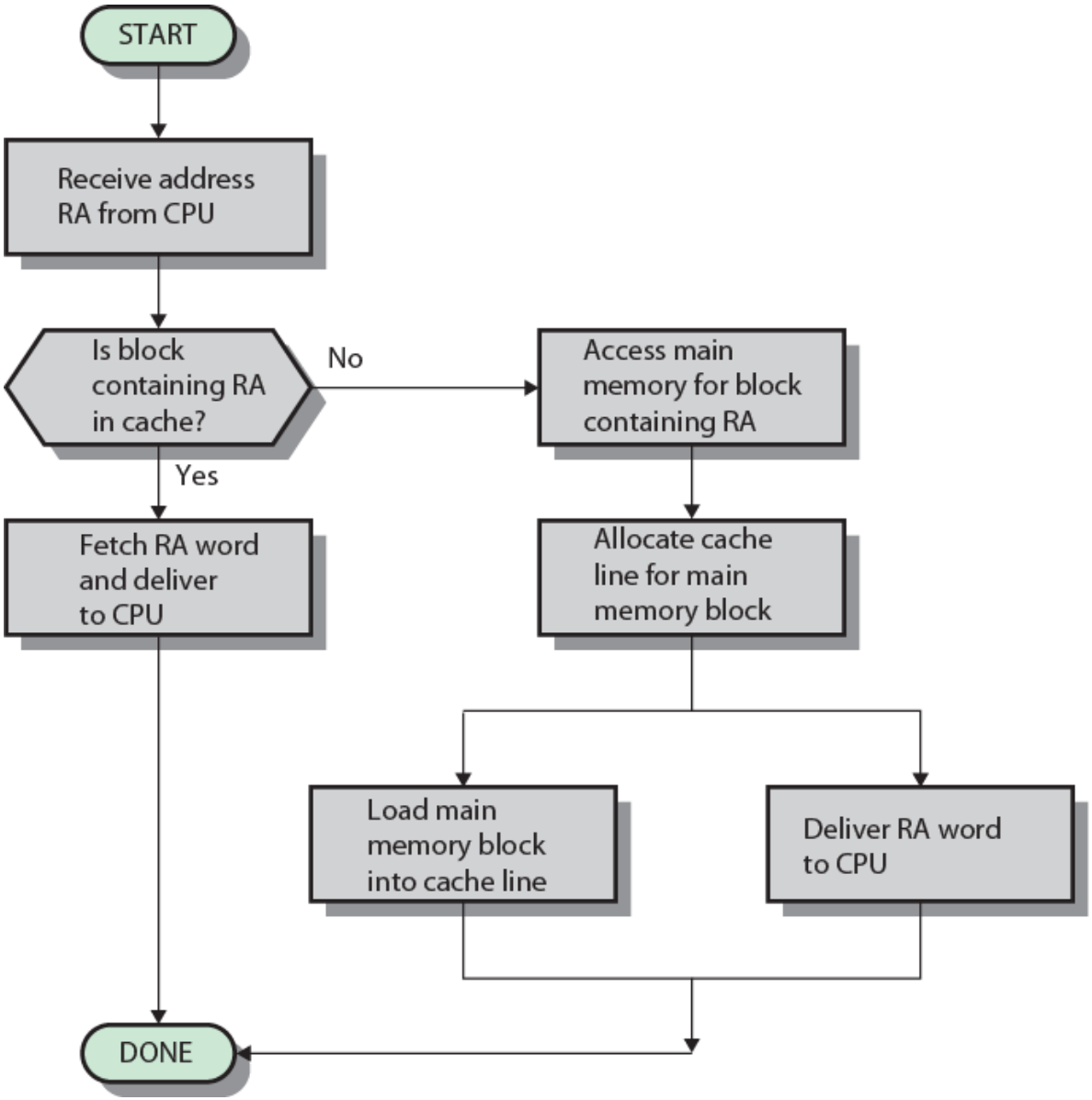
(b) Main memory

TAG - A unique identifier for a group of data. Because different regions of memory may be mapped into a block, the tag is used to differentiate between them.

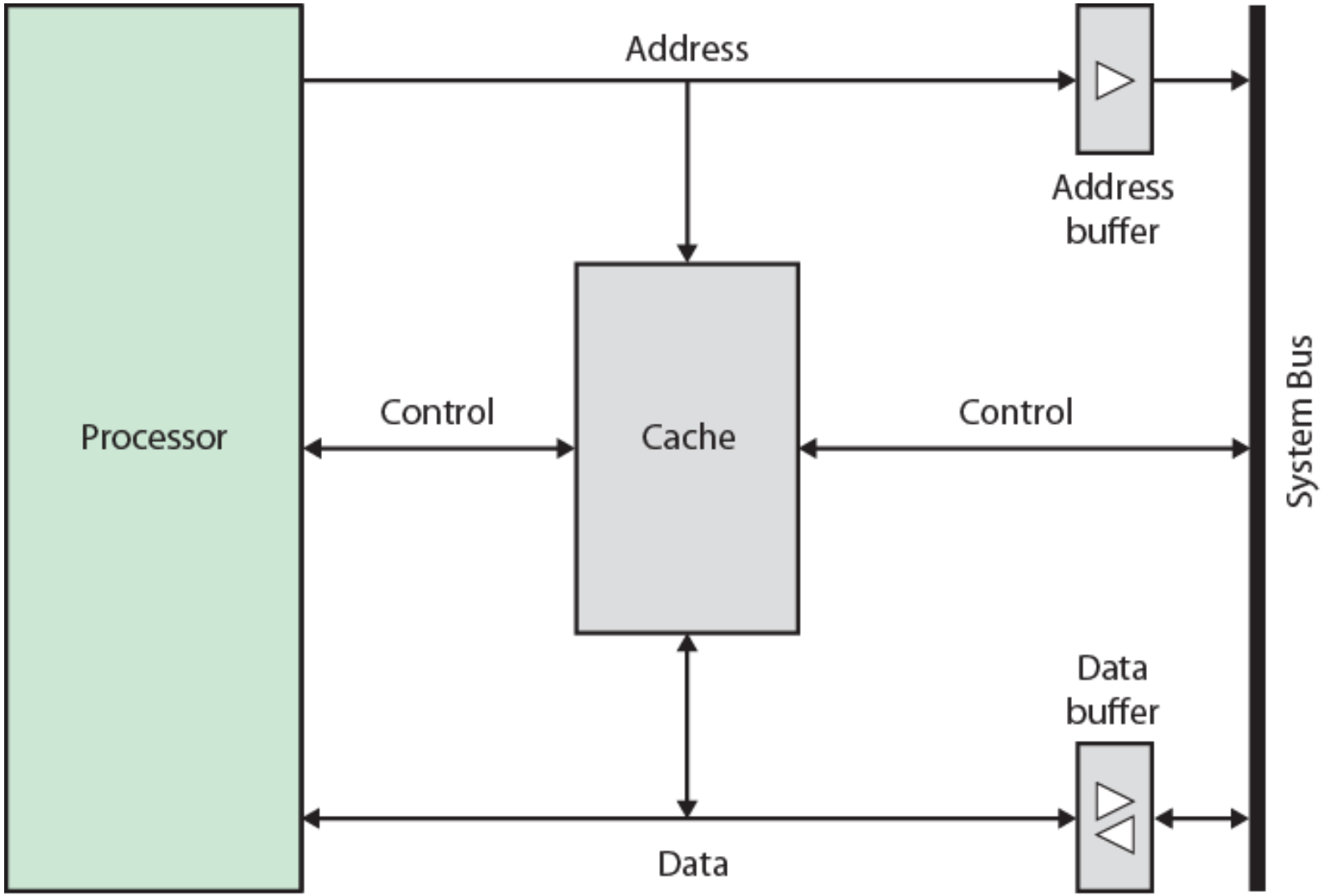
Cache operation – overview

- CPU requests contents of memory location
- Check cache for this data
- If present, get from cache (fast)
- If not present, read required block from main memory to cache
- Then deliver from cache to CPU
- Cache includes tags to identify which block of main memory is in each cache slot

Cache Read Operation - Flowchart



Typical Cache Organization



Locality of Reference

- During the course of the execution of a program, memory references tend to cluster
- e.g. loops
- 3 types
- Temporal: recently referenced inst or data

  e.g loops
- Spatial: whose addresses are near one another

  e.g searching and sorting algo
- Sequential: sequential order

Cache Design

- Cache size
- Replacement Algorithm
- Mapping Function
- Write Policy
- Block Size
- Number of Caches

Size does matter

- Cost
  - More cache is expensive
- Speed
  - More cache is faster (up to a point)
  - Checking cache for data takes time

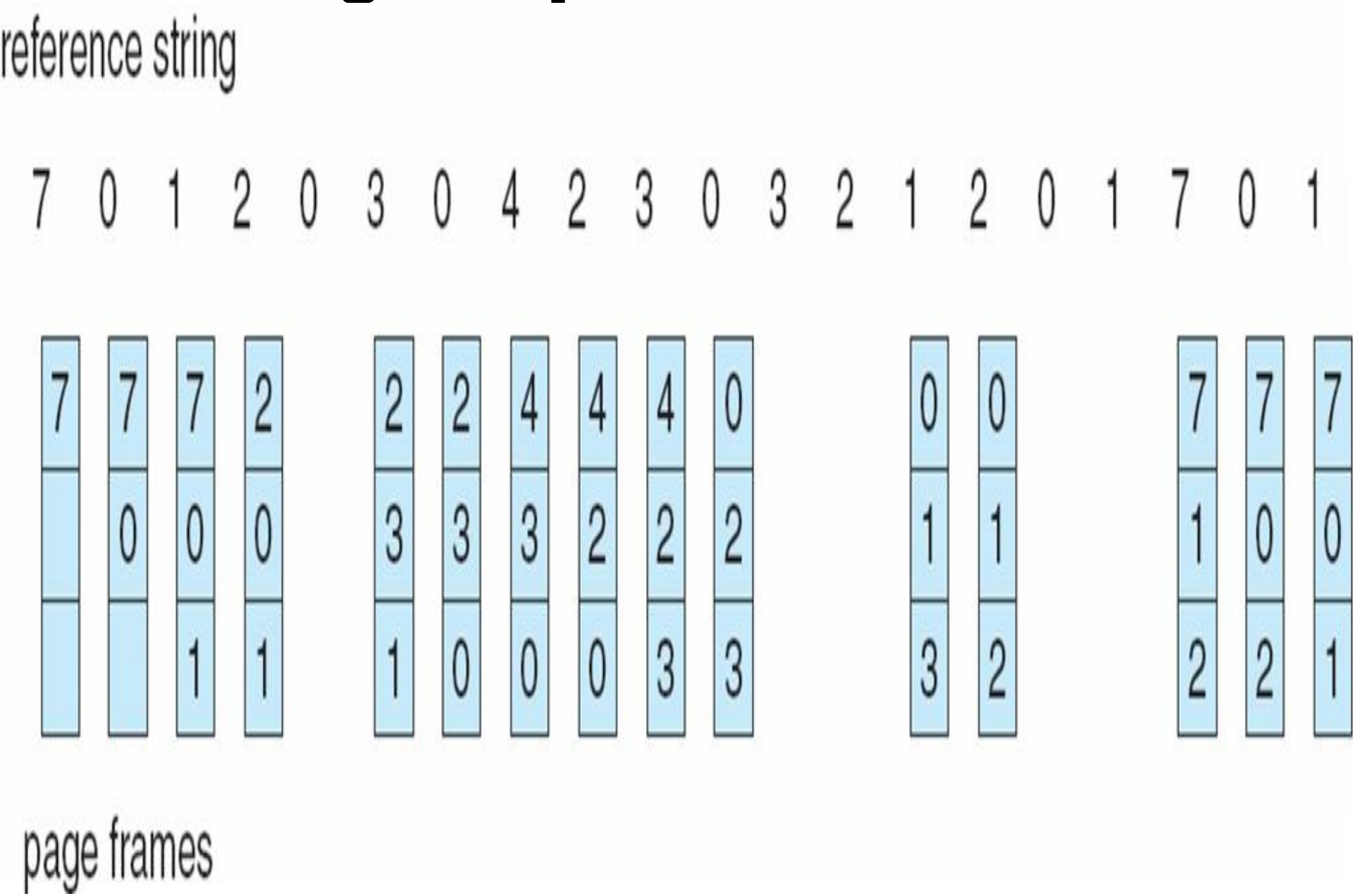
Replacement Algorithms

- Hardware implemented algorithm (speed)
- Least Recently used (LRU)
  - Pick the slot that hasn't been used in the longest time.
- First in first out (FIFO)
  - replace block that has come into cache first
- Random
- OPT-Optimal(Future)

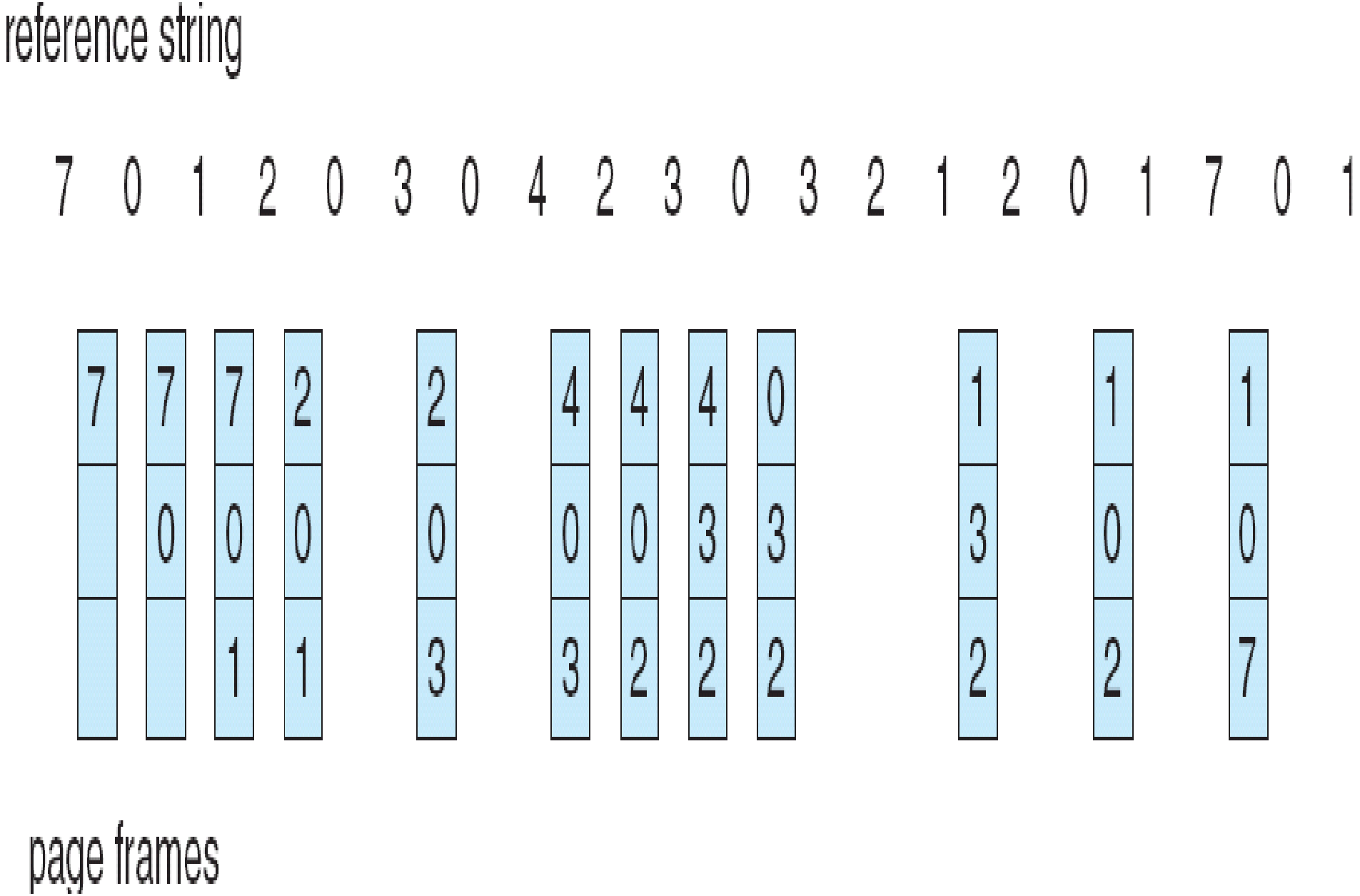
FIFO,LRU,OPT

- 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

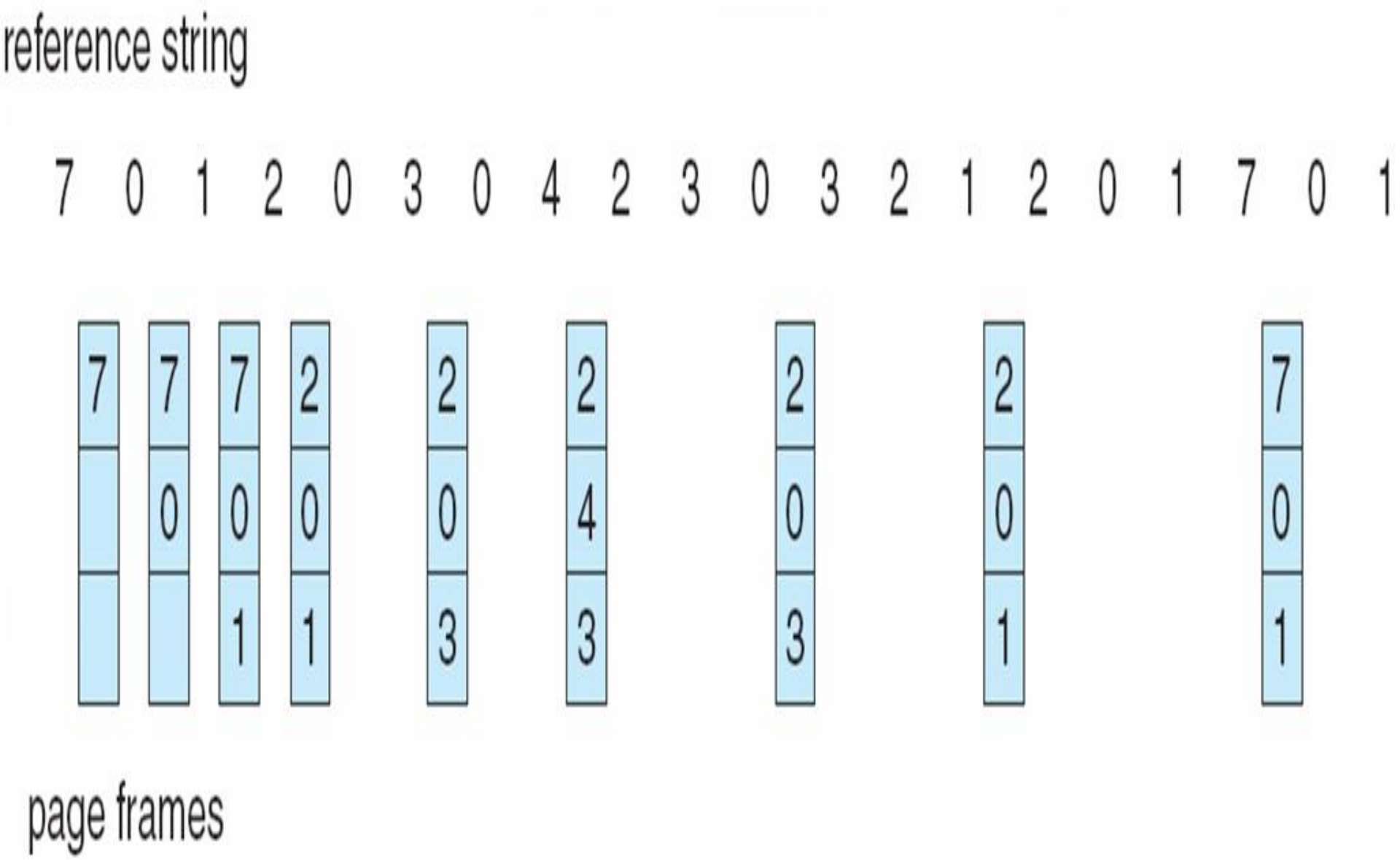
FIFO Page Replacement



LRU Page Replacement



Optimal Page Replacement



2,3,2,1,5,2,4,5,3,2,5,2

6,0,12,0,30,4,2,30,32,1,20,15

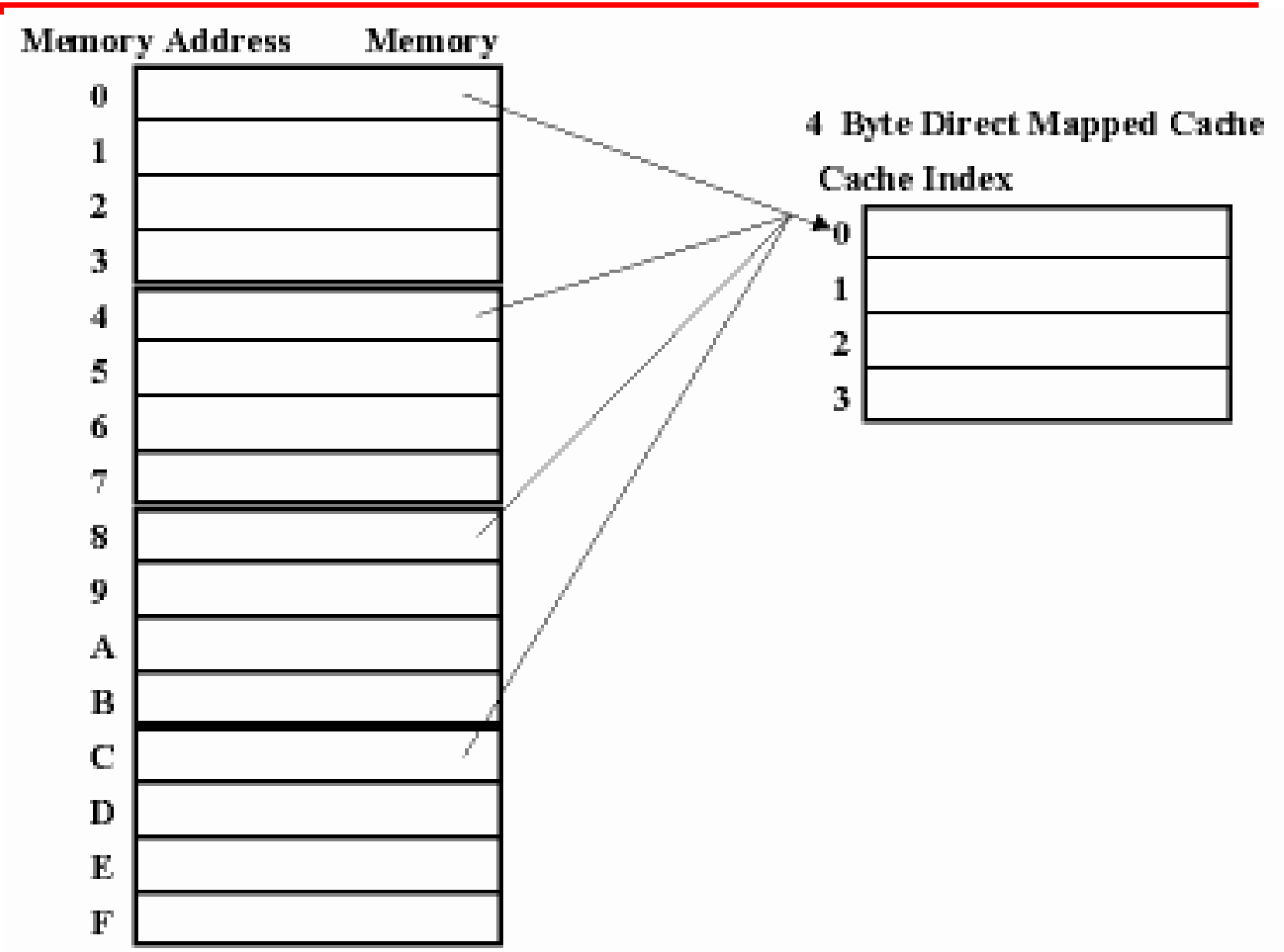
MAPPING TECHNIQUES

- There are fewer lines than main memory blocks
- An algorithm is needed for mapping MM blocks into cache lines.
- A means is needed for determining which MM block currently occupies a cache line.

MAPPING TECHNIQUES

- DIRECT MAPPING
- ASSOCIATIVE MAPPING
  - —FULLY ASSOCIATIVE MAPPING
  - —SET ASSOCIATIVE MAPPING
    - **– 2-WAY SET ASSOCIATIVE MAPPING**

DIRECT MAPPING CONCEPT



DIRECT MAPPING

- Cache- 128 blocks of 16 words each
  - —128 * 16 = 4096 **approx 4 KB**

- Main Memory – 4K blocks of 16 words each
  - —4K * 16 = 64000 **approx 64 KB**

- **TOTAL ADDRESS SIZE 16 bit**

Tag	Block	Word
5(16-(7+4))	**7(2^7)**	**4 (2^4)**

Direct Mapping
Cache Line Table

Cache line	Main Memory blocks held
0	0, m, 2m, 3m…2s-m
1	1,m+1, 2m+1…2s-m+1
m-1	m-1, 2m-1,3m-1…2s-1

$$I = J \bmod M$$

I=cache line no

J=Block no of main memory

M=total no of slots in cache

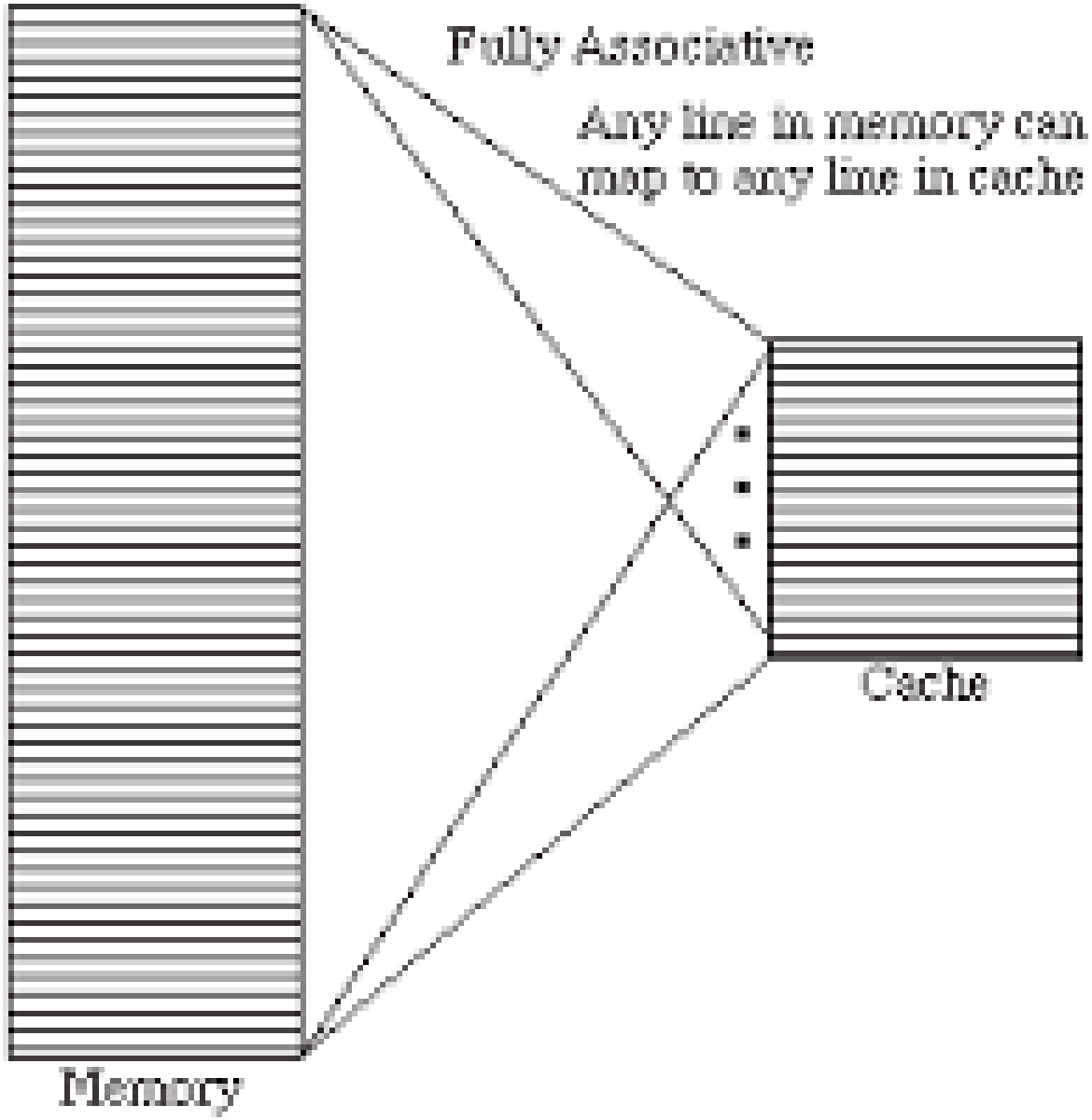
Direct Mapping pros & cons

- Simple
- Inexpensive
- Fixed location for given block
  - If a program accesses 2 blocks that map to the same line repeatedly, cache misses are very high

Associative Mapping

- A main memory block can load into any line of cache
- Memory address is interpreted as tag and word
- Tag uniquely identifies block of memory
- Every line’s tag is examined for a match
- Cache searching gets expensive

FULLY ASSOCIATIVE MAPPING



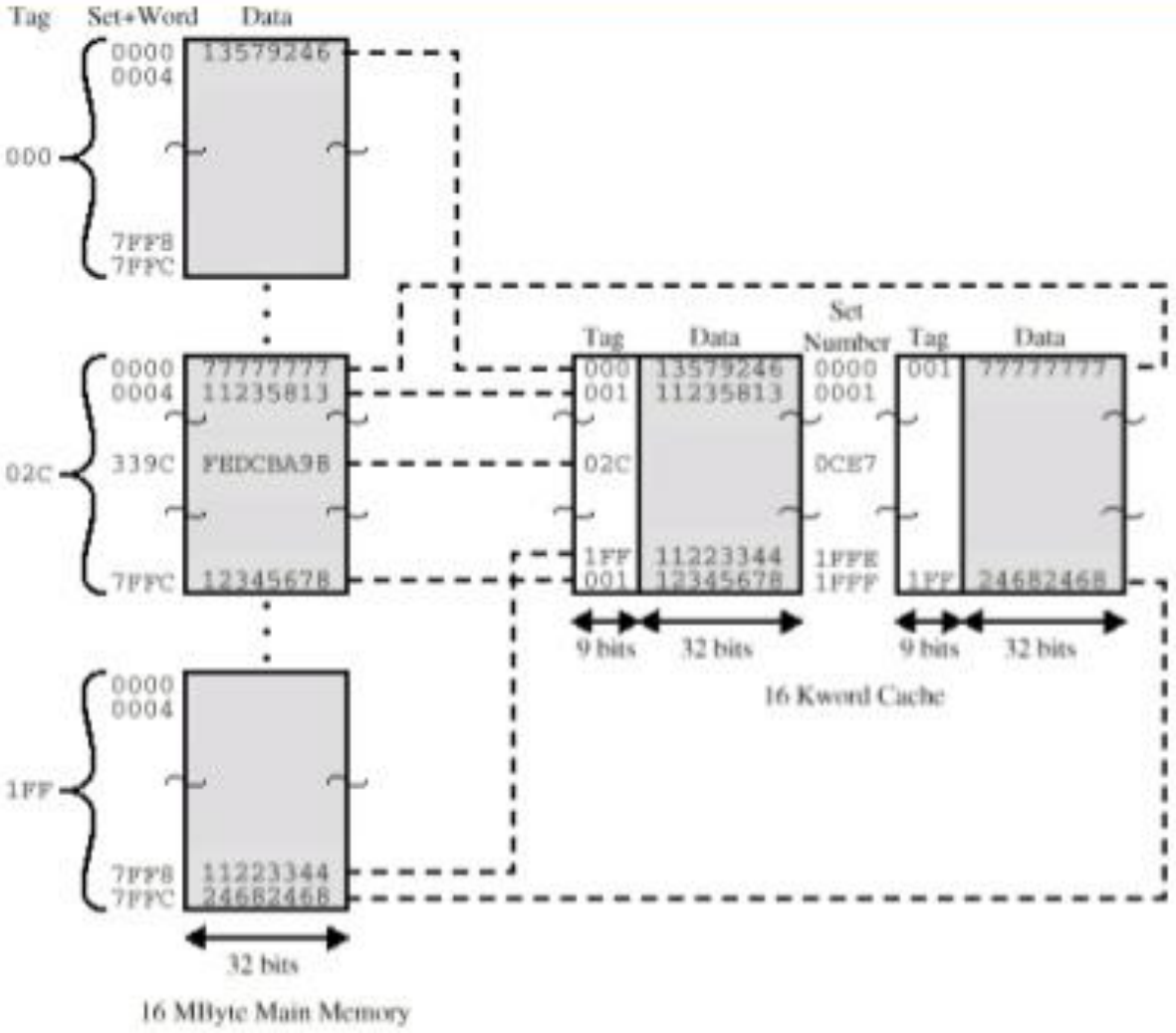
ASSOCIATIVE MAPPING

- Cache- 128 blocks of 16 words each
- 128 * 16 = 2048 **approx 2 KB**

- Main Memory – 4K blocks of 16 words each
- 4K * 16 = 64000 **approx 64 KB**
- **TOTAL ADDRESS SIZE 16 bit**

Tag	Word
12(16-4)	4 (2^4)

Two Way Set Associative Mapping Example



SET ASSOCIATIVE MAPPING(2-way)

- Cache- 128 blocks of 16 words each
- 128 * 16 = 2048 **approx 2 KB**
- Set divided into 2 (128 /2) =64
- Main Memory – 4K blocks of 16 words each
- 4K * 16 = 64000 **approx 64 KB**

Tag	Set	Word
6(16-(4+6))	**6(2^6)**	**4 (2^4)**

Set Associative Mapping

- Cache is divided into a number of sets
- Each set contains a number of lines
- A given block maps to any line in a given set
  - —e.g. Block B can be in any line of set i
- e.g. 2 lines per set
  - —2 way associative mapping
  - —A given block can be in one of 2 lines in only one set

(1) Tag size $= \log_2 \frac{\text{No. of blocks in mm}}{\text{No. of sets in cache}}$

(2) No. of lines in each cache set
$= \frac{\text{cache size}}{\text{No. of ways.}}$
set size $= \log_2$ (No. of ~~lines~~ in cache)

(3) word length $= \log_2$ (No. of words in a block)

(4) size of the main memory =
No. of blocks * size of each block.

Problem statement

Consider a cache consisting of 256 blocks of 16 words each for a total of 4096(4KB) words and assume that the main memory is addressable by a 16 bit address and it consists of 4KB blocks. How many bits are there in each of the TAG,BLOCK/SET and WORD field for 2-way set associative technique?

∴ No. of bits req^d to address block
$= \log_2(4k) \Rightarrow 12$ bits

No. of lines in each cache set
$\Rightarrow \frac{256}{2} = 128 \Rightarrow$ ~~7~~

∴ No. of bits req^d to address a set $= \log_2 128 \Rightarrow \log_2 2^7 = 7$ bits

word length $= \log_2 16 = 4$ bits

Tag size $= \log_2 \frac{\text{No. of block in mm}}{\text{No. of sets in cache}}$

$= \log_2 \frac{4K}{128}$

$= \log_2 2^5 \Rightarrow 5$ bits.

memory address

5	7	4

Problem statement

A block set associative cache memory consists of 128 blocks divided into four block sets. The main memory consists of 16,384 blocks and each block contains 256 eight bit words.

i) How many bits are required for addressing the main memory?

ii) How many bits are needed to represent the TAG, SET and WORD fields?

i. 2^22 ii.9 5 8

Problem statement

A block set associative cache memory consists of 64 blocks divided into four block sets. The main memory consists of 4096 blocks and each block contains 128 words of 16 bits length.

i) How many bits are there in main memory?

ii) How many bits are needed to represent the TAG, SET and WORD fields?

i. 2^23 ii. 8 4 7

No of Caches: L1 L2 L3 CACHE

- **L1-cache** is the fastest cache and it usually comes within the processor chip itself.
- The L1 cache typically ranges in size from 8KB to 64KB and uses the high-speed SRAM (static RAM) instead of the slower and cheaper DRAM (dynamic RAM) used for main memory.
- The Intel Celeron processor uses two separate 16KB L1 caches, one for the instructions and one for the data.

L1 L2 L3 CACHE

- **L2 cache** comes between L1 and RAM

  —(**Processor-L1-L2-RAM**) and is bigger than the primary cache (typically 64KB to 4MB).

- **L3 cache** is not found nowadays as its function is replaced by L2 cache. L3 caches are found on the motherboard rather than the processor. It is kept between RAM and L2 cache.

Write Policy

- Must not overwrite a cache block unless main memory is up to date
- Multiple CPUs may have individual caches
- I/O may address main memory directly

Write through

- All writes go to main memory as well as cache
- Multiple CPUs can monitor main memory traffic to keep local (to CPU) cache up to date
- Lots of traffic
- Slows down writes

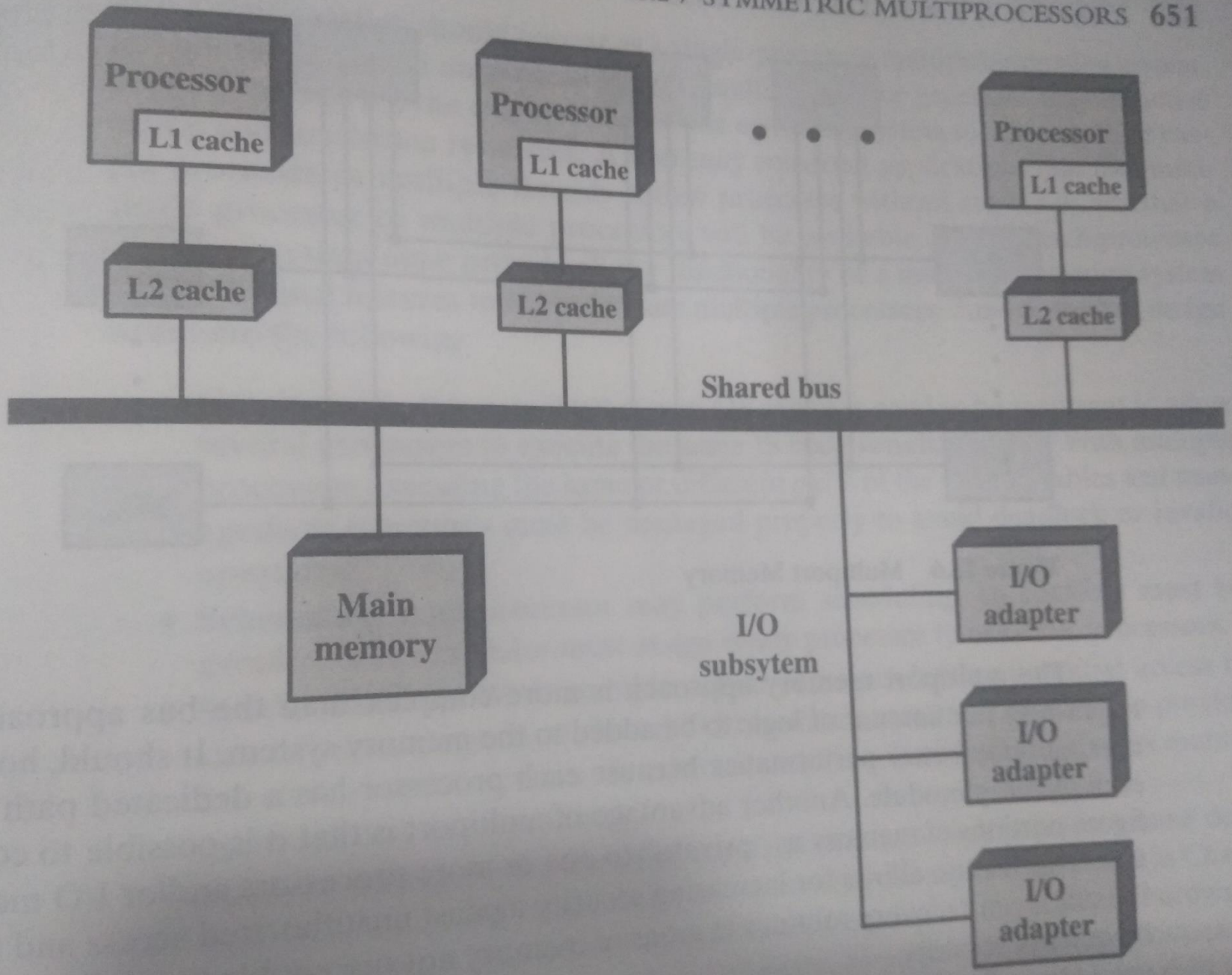
- Remember bogus write through caches!

Write back

- Updates initially made in cache only
- Update bit for cache slot is set when update occurs
- If block is to be replaced, write to main memory only if update bit is set
- Other caches get out of sync
- I/O must access main memory through cache
- N.B. 15% of memory references are writes

Buffered Write through

- Write accesses are buffered
- Read access- cache hit , can be performed simultaneously
- Two consecutive write operations-cache miss ,requires processor to wait


Processor
L1 cache
L2 cache
Processor
L1 cache
L2 cache
Processor
L1 cache
L2 cache
Shared bus
Main memory
I/O subsytem
I/O adapter
I/O adapter
I/O adapter

Cache Coherence

- Problem - multiple copies of same data in different caches
- Can result in an **inconsistent** view of memory
- Write through can also give problems unless caches monitor memory traffic
- Write back policy can lead to inconsistency

Software Solutions

- Compiler and operating system deal with problem
- Overhead transferred to compile time
- Complier marks data likely to be changed and OS prevents such data from being cached
- Design complexity transferred from hardware to software

Hardware Solution

- Cache coherence protocols
- Dynamic recognition of potential problems
- Run time
- More efficient use of cache
- Transparent to programmer
- Snoopy protocols(to maintain cache consistency)

Snoopy Protocols

- Distribute cache coherence responsibility among **cache controllers**
- Cache recognizes that a line is shared
- Updates announced to other caches
- Suited to bus based multiprocessor
- Increases bus traffic

Write Update

- Multiple readers and writers
- Updated word is distributed to all other processors
- Some systems use an adaptive mixture of both solutions
- Broadcast new data block to all other

Write Invalidate

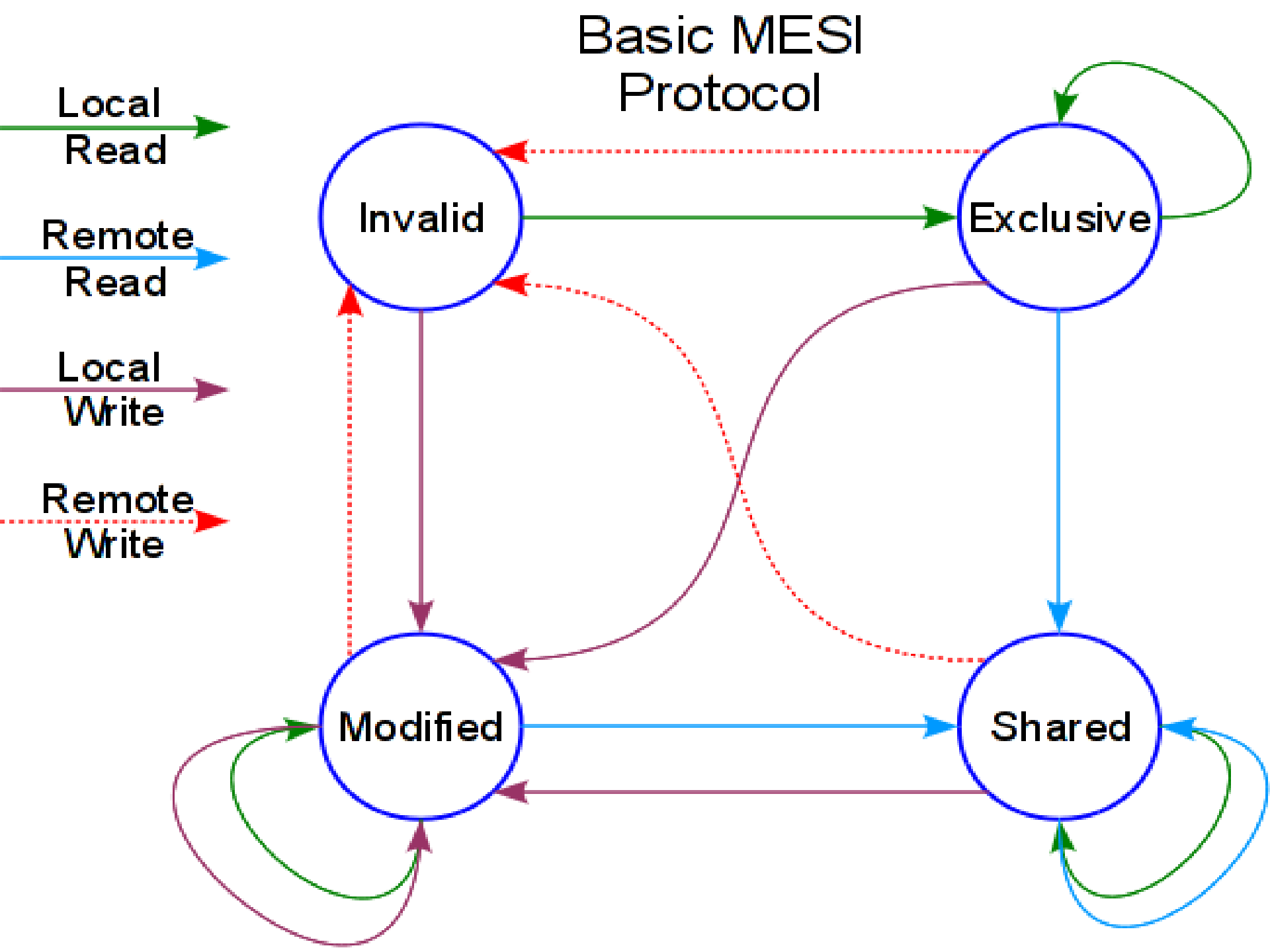
- Multiple readers, one writer
- When a write is required, all other caches of the line are invalidated
- Writing processor then has **exclusive access** until line required by another processor
- State of every line is marked as modified, exclusive, shared or invalid
- MESI protocol

MESI Protocol

- Processors providing cache coherence commonly implement a MESI protocol - where the letters of the acronym represent the four states that a cache line may be in:
- Modified
- Exclusive
- Shared
- Invalid

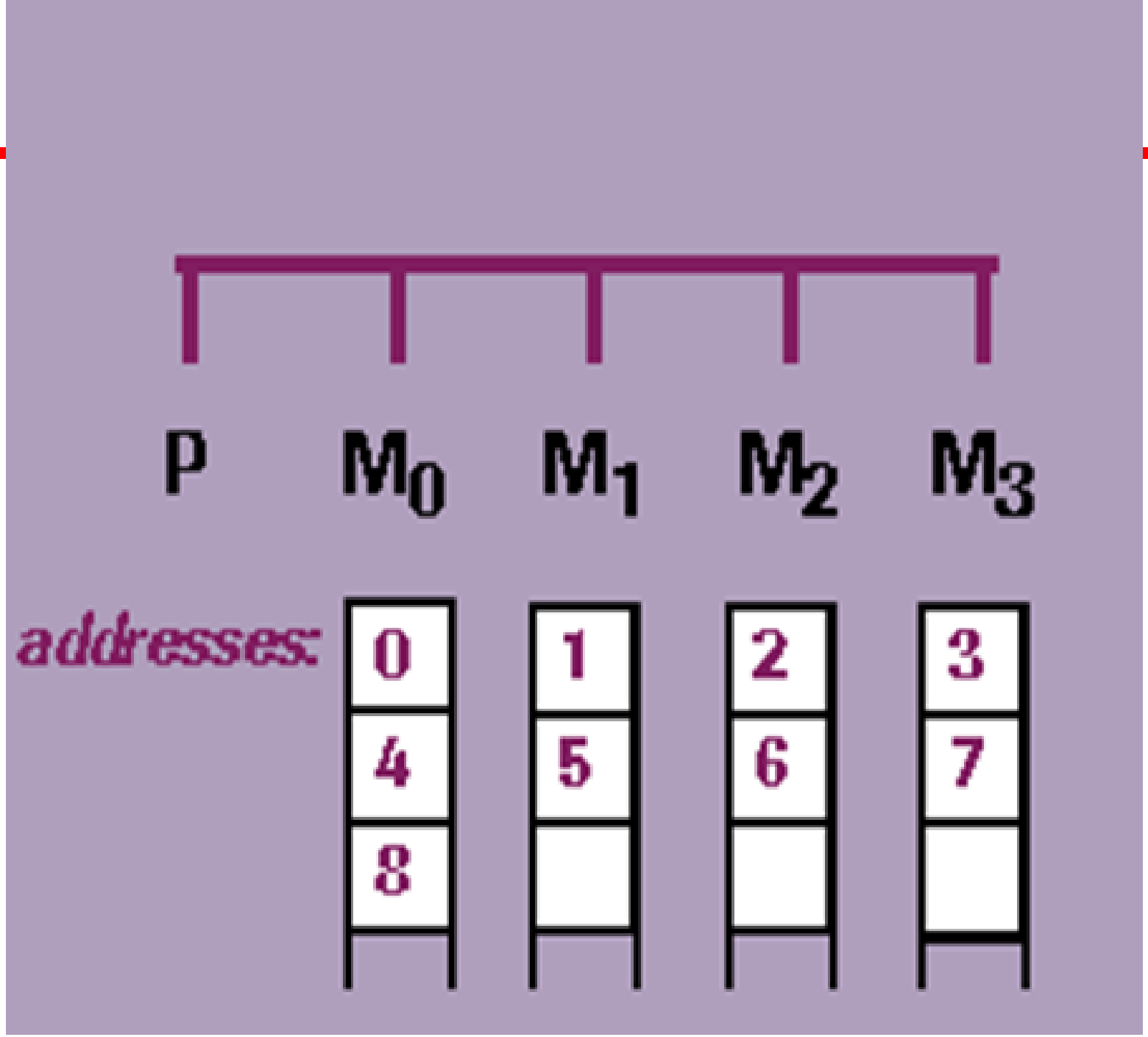
MESI protocol

- **I**nvalid: This cache line is not valid
- **E**xclusive: This cache has the only copy of the data. The memory is valid.
- **S**hared: More than one cache is holding a copy of this line. The memory copy is valid.
- **M**odified: The line has been modified. The memory copy is invalid.

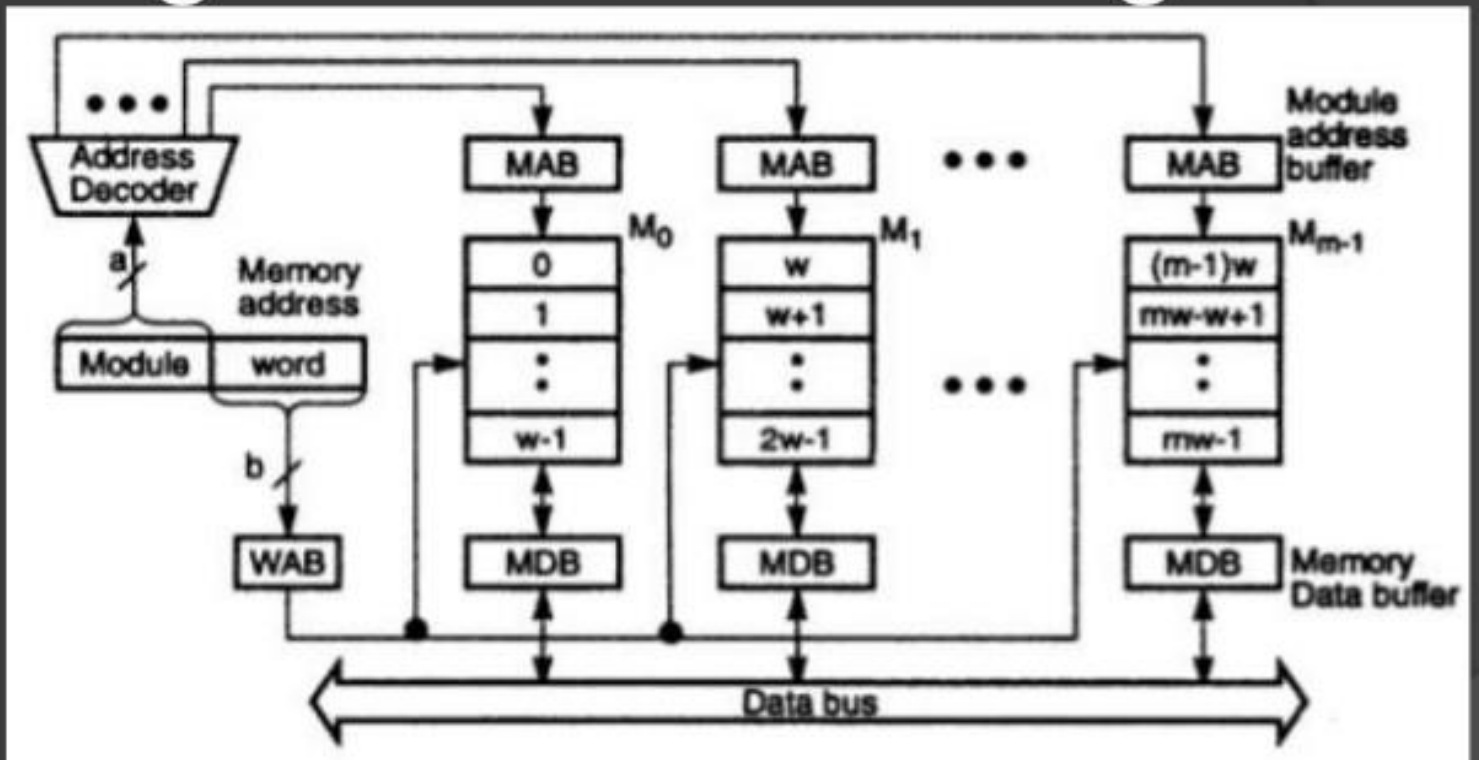

Basic MESI Protocol
Local Read
Remote Read
Local Write
Remote Write
Invalid
Exclusive
Modified
Shared

High Speed memories:Interleaved And Associative Memory

- **Interleaved memory** is a design made to compensate for the relatively slow speed of dynamic random-access memory (DRAM)
- **Spreads memory** addresses evenly across
- Contiguous memory reads and writes
- Resulting in **higher memory throughputs** due to reduced waiting.


P
M0
M1
M2
M3
addresses:
0
4
8
1
5
2
6
3
7

High order interleaving



(b) High-order m-way interleaving

15 Two interleaved memory organizations with $m = 2^a$ modules a

Interleaved And Associative Memory

- Content-addressed or associative memory-memory is **accessed by its content** (as opposed to an explicit address).
- **Reference clues** are "associated" with actual memory contents until a desirable match (or set of matches) is found.
- Humans retrieve information best when it can be linked to other related information.

Interleaved And Associative Memory

memory can be viewed as having the two field format : called the key. Items s

KEY, DATA

KEY is the stored address and DATA is the information to be acces

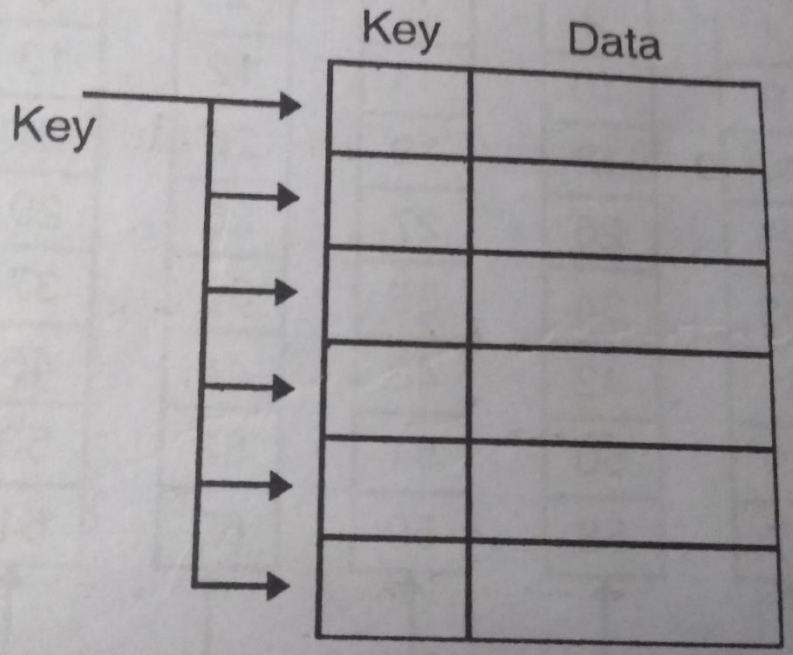
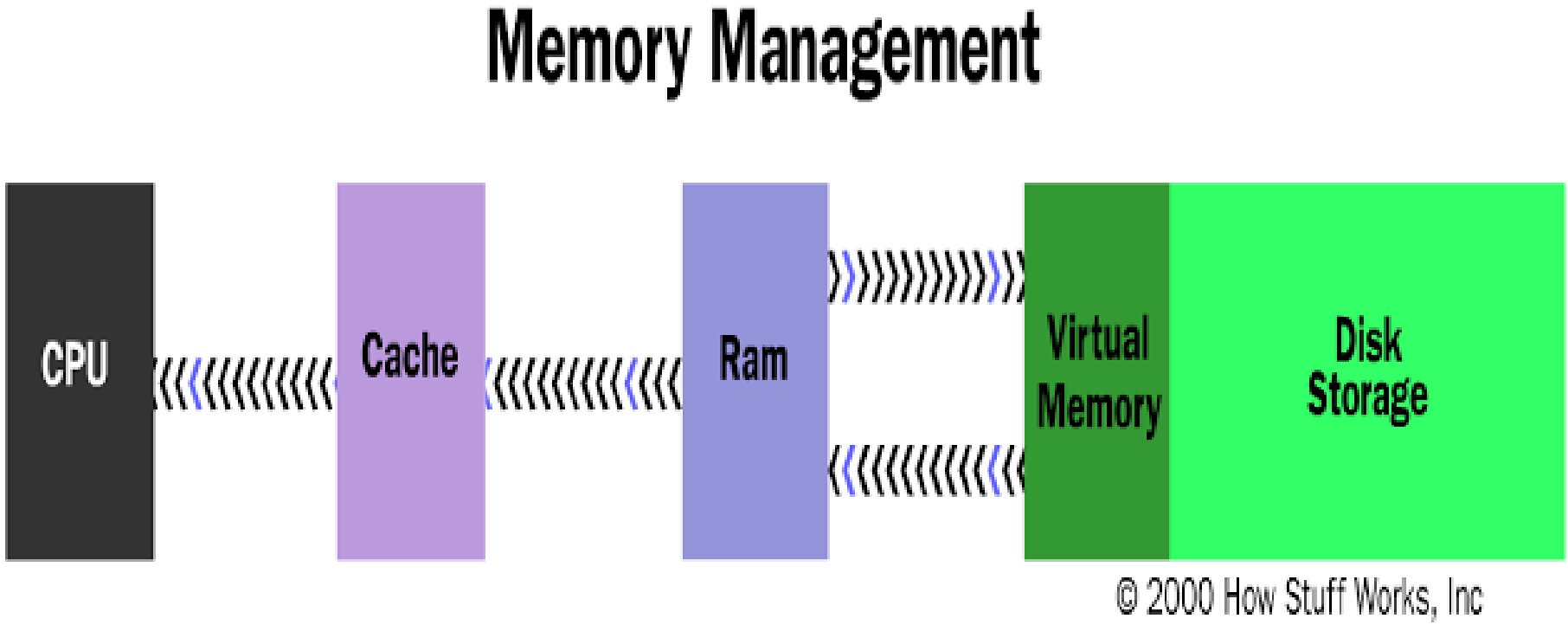
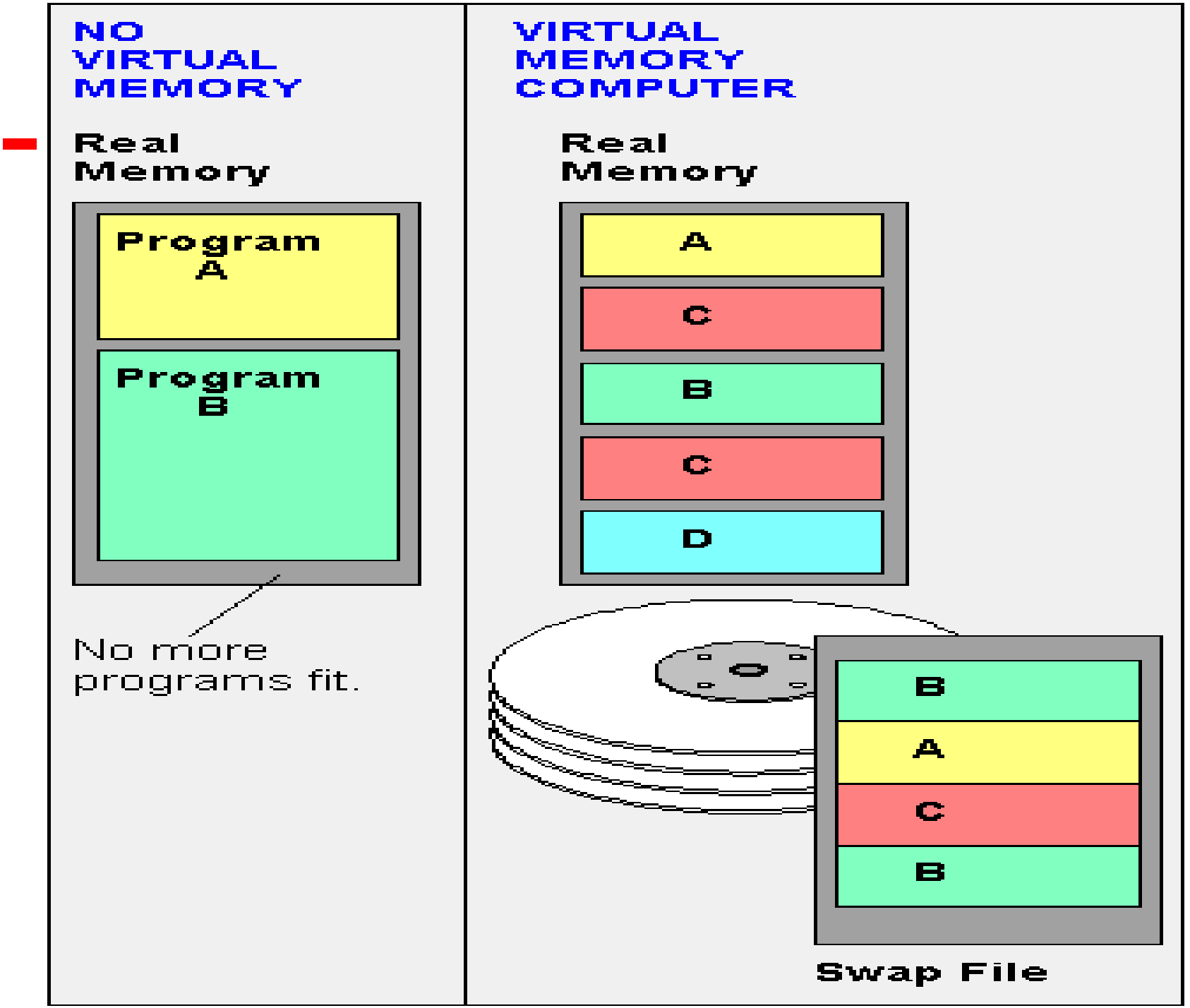


Fig. 4.39 : A key to be searched is matched simultaneously

ve searching is based on simultaneous matching of the key to

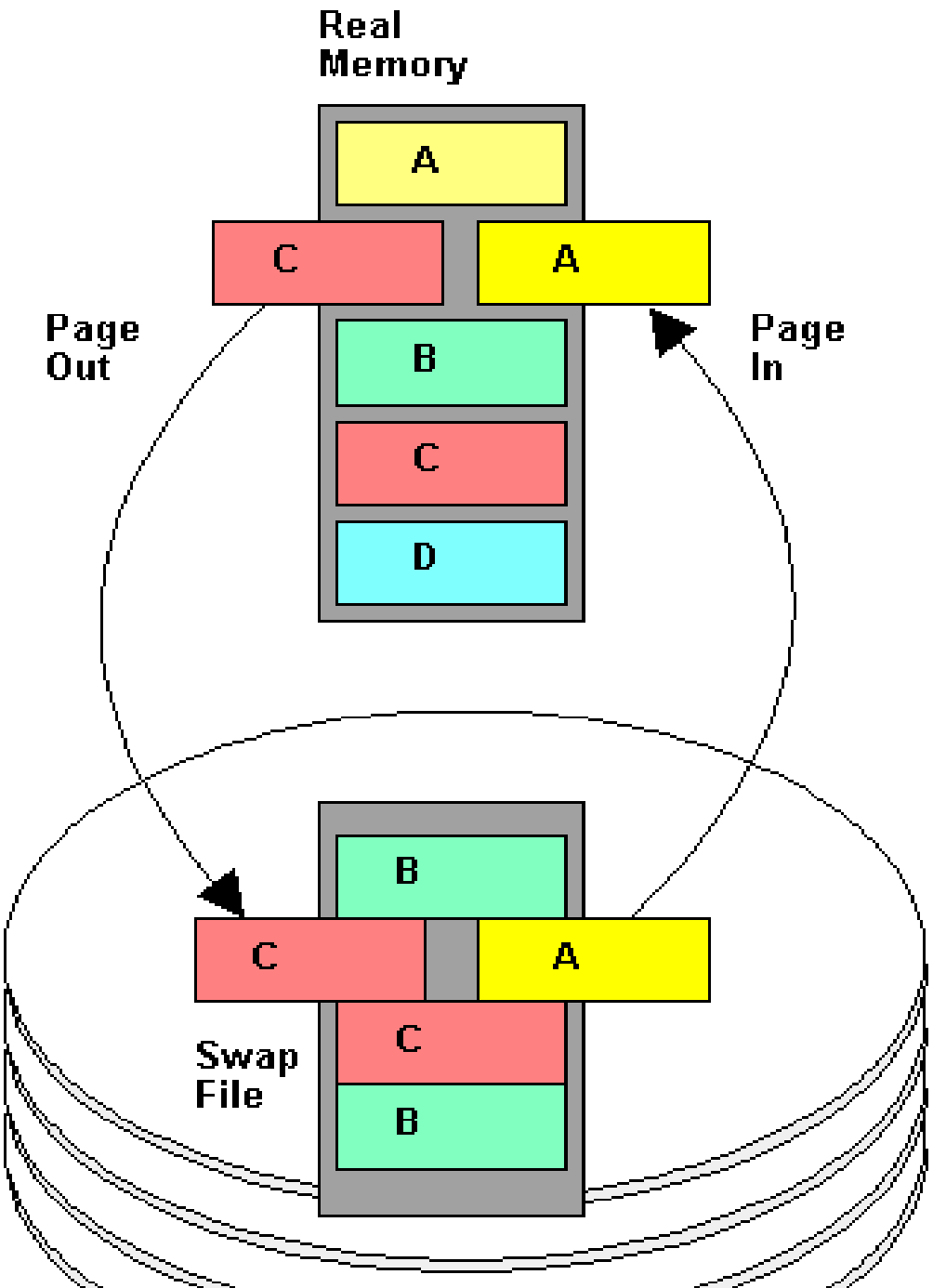
Virtual Memory




NO VIRTUAL MEMORY
Real Memory
Program A
Program B
No more programs fit.
VIRTUAL MEMORY COMPUTER
Real Memory
A
C
B
C
D
B
A
C
B
Swap File

- Virtual memory

Allows more programs to be opened simultaneously by using the hard disk as temporary storage of memory pages.



VIRTUAL MEMORY

- Most computers today have something like 32 or 64MB of RAM available for the CPU to use
- Not enough to run all of the programs that most users expect to run at once.
- Email program,a Web browser and word processor into RAM simultaneously
- Look at RAM for areas that have not been used recently and copy them onto the hard disk

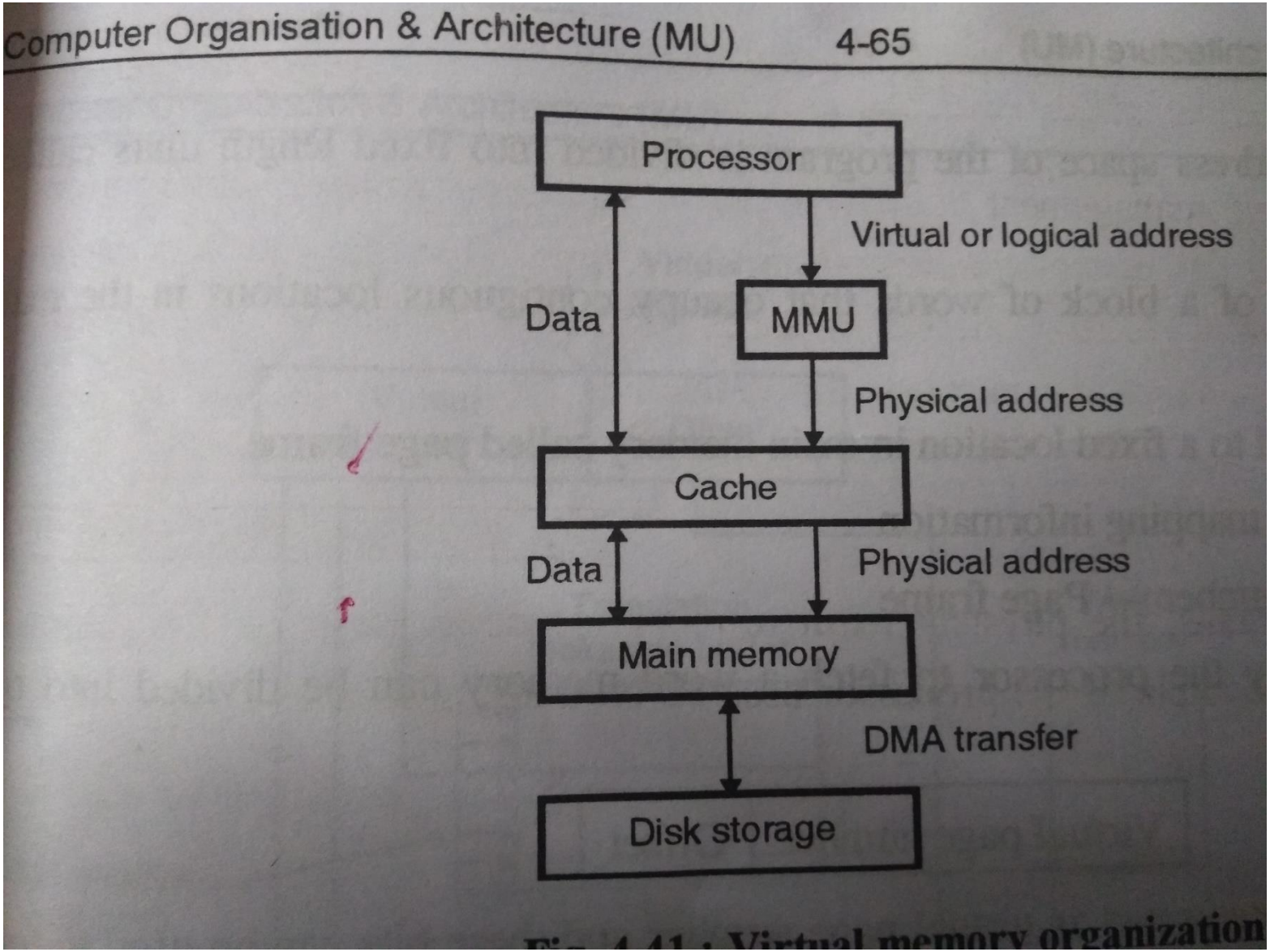
VIRTUAL MEMORY

- This frees up space in RAM to load the new application.
- **Copying** happens automatically(feels like unlimited RAM space even though it only has 32 megabytes installed.
- Hard disk space is so much cheaper than RAM chips, thus has a economic benefit.

VIRTUAL MEMORY

- Read/write speed of a hard drive is much slower than RAM, and the technology of a hard drive is not geared toward accessing small pieces of data at a time.
- **Operating system** has to constantly swap information back and forth between RAM and the hard disk. This is called **thrashing**, and it can make your computer feel incredibly slow.

Address translation



PAGING

- Primary memory is divided into small equal sized partitions (256, 512, 1K) called page frames.
- Process are divided into **same sized blocks** called **pages.**
- Only bring in the pages you are referencing and keep those you have recently referenced.
- Need a page table to this management.

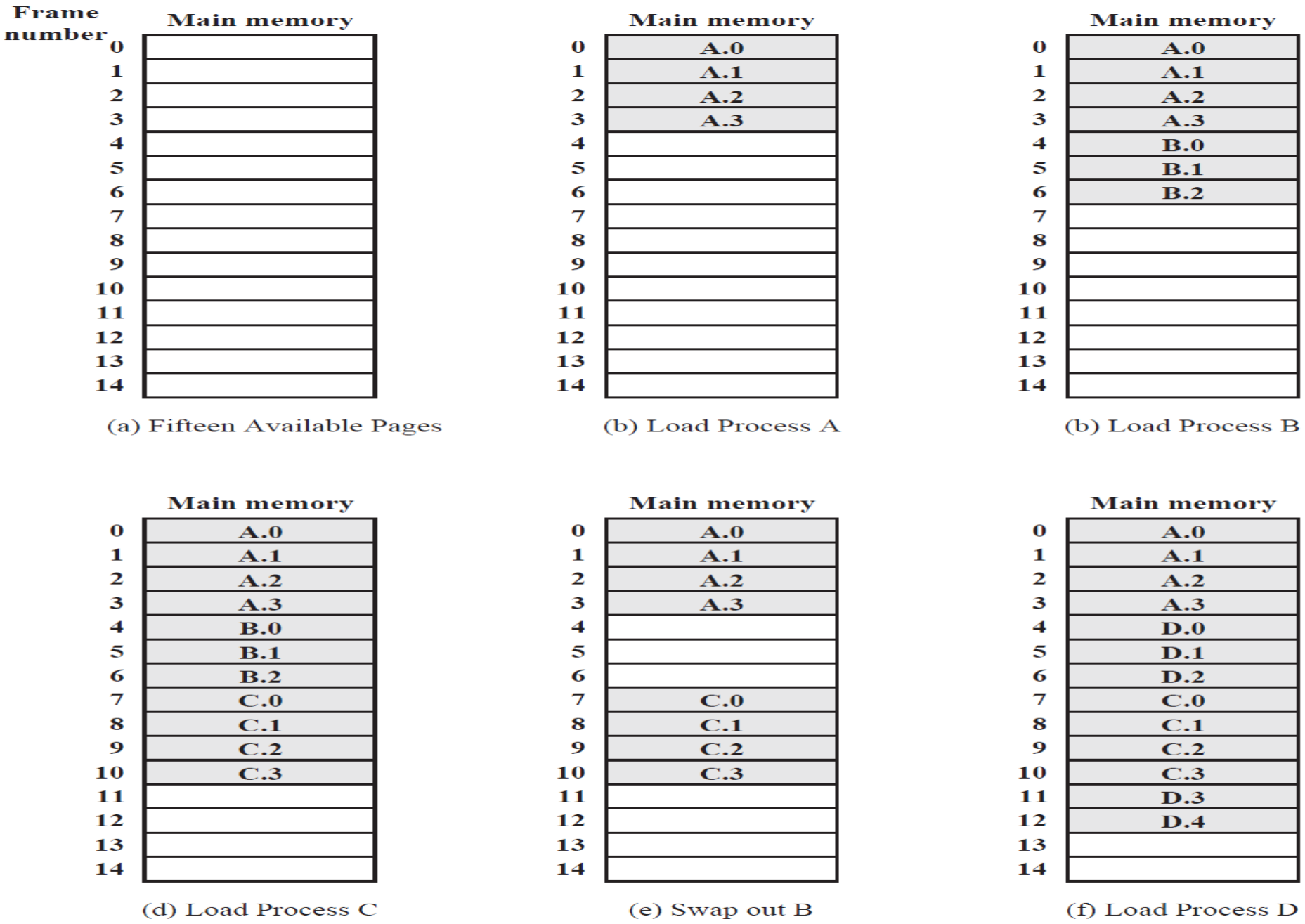


Figure 7.9 Assignment of Process Pages to Free Frames

Paging

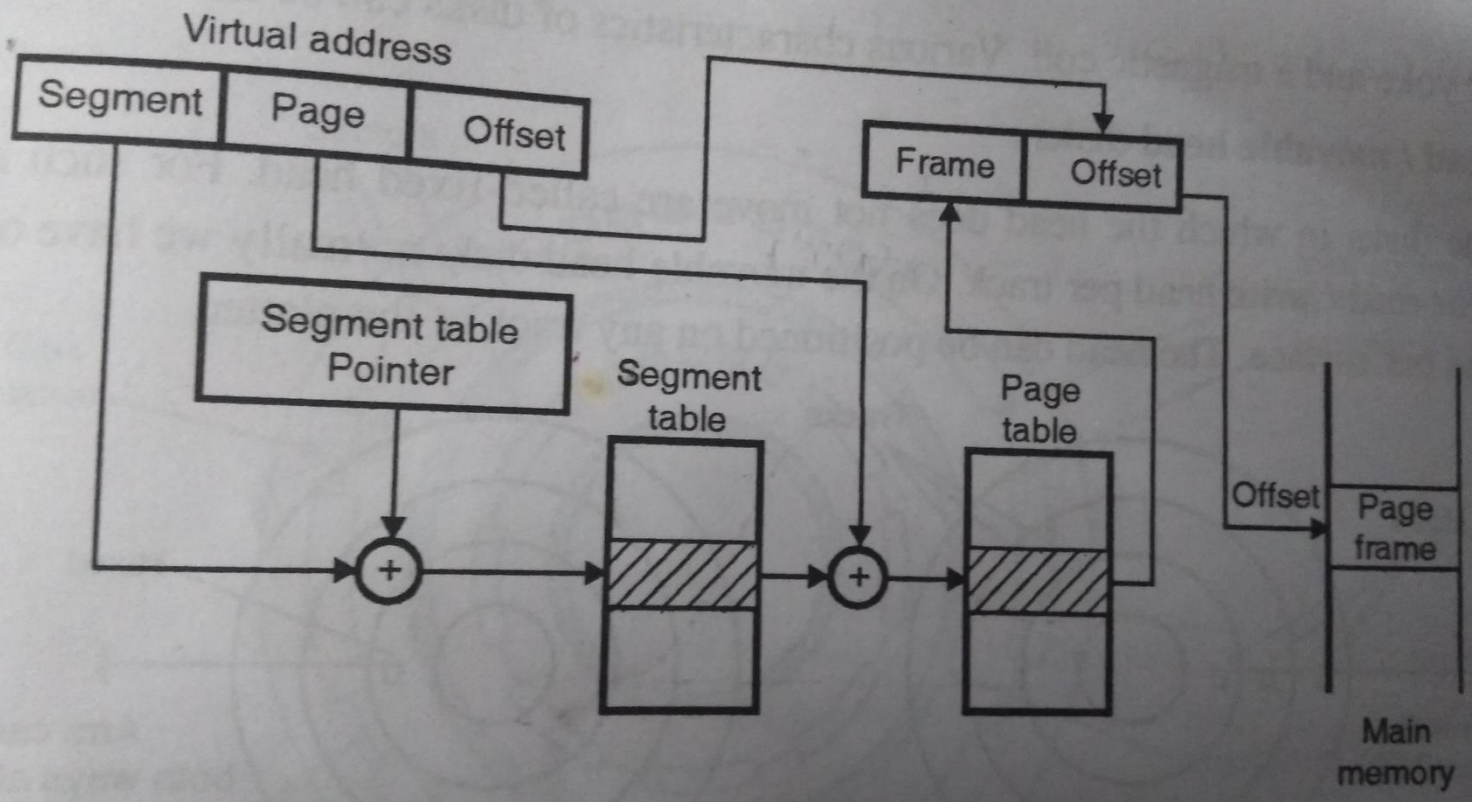


Fig. 4.50 : Address translation in a segmentation / paging system

With each process, there is a segment table and a number of page map tables.

Paging-TLB

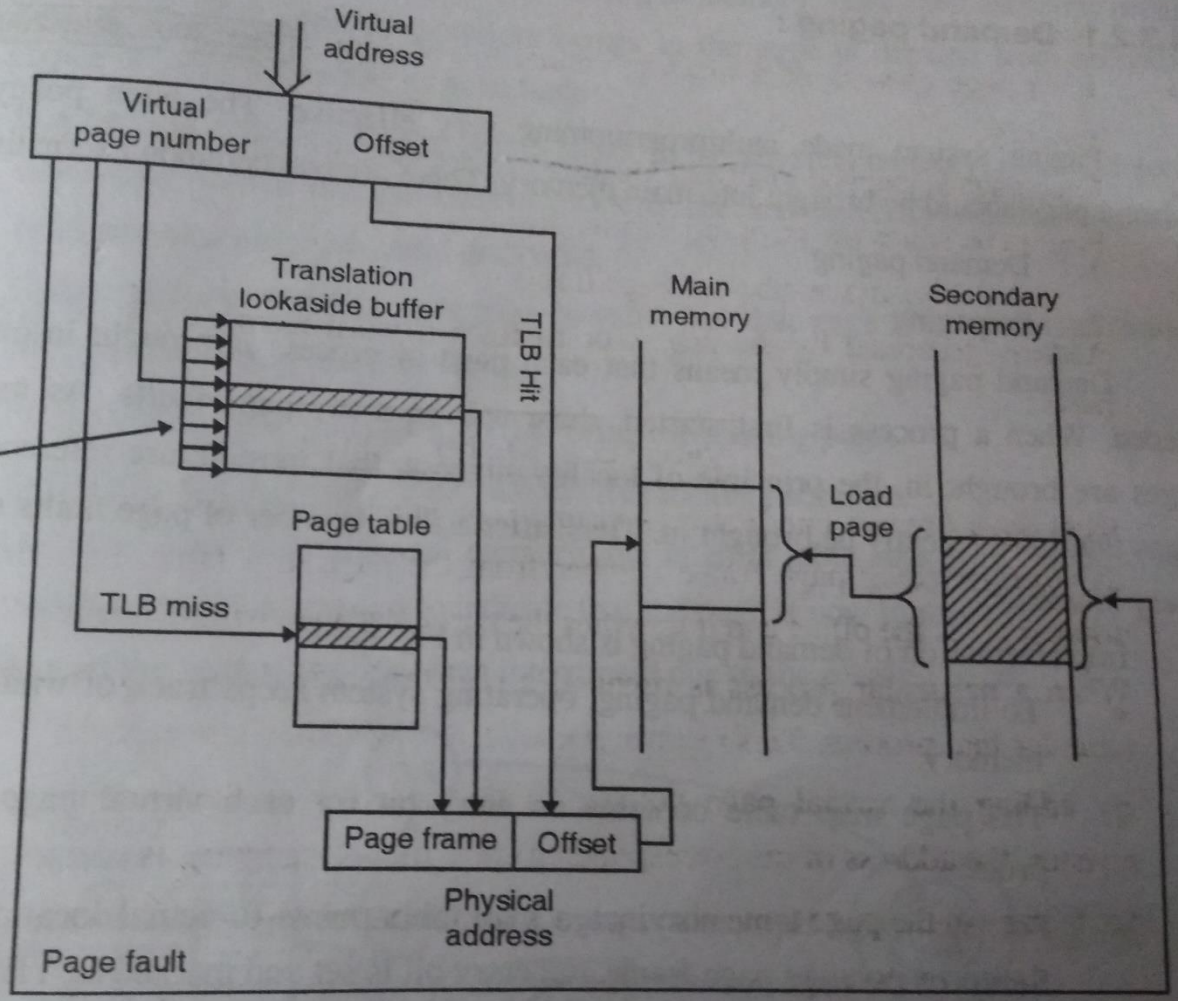


Fig. 4.43 : Use of a Translation Lookaside Buffer

The organization of the resulting paging hardware is shown in Fig. 4.43. In the abo

SEGMENTATION

- Paging suffers from **internal fragmentation**.
- Segmentation maps segments representing data structures, modules, etc. into **variable partitions.** Not all segments of a process again are loaded at a time, nor are they in contiguous memory blocks.
- We need a segment table very much like a page table.

SEGMENTATION

- Properties of a program such as scope of a variable and access rights are naturally specified by segment. These properties can be used to protect a program from unauthorized access.
- Segments can grow dynamically. Certain segment types-stacks and queues grow during run time.

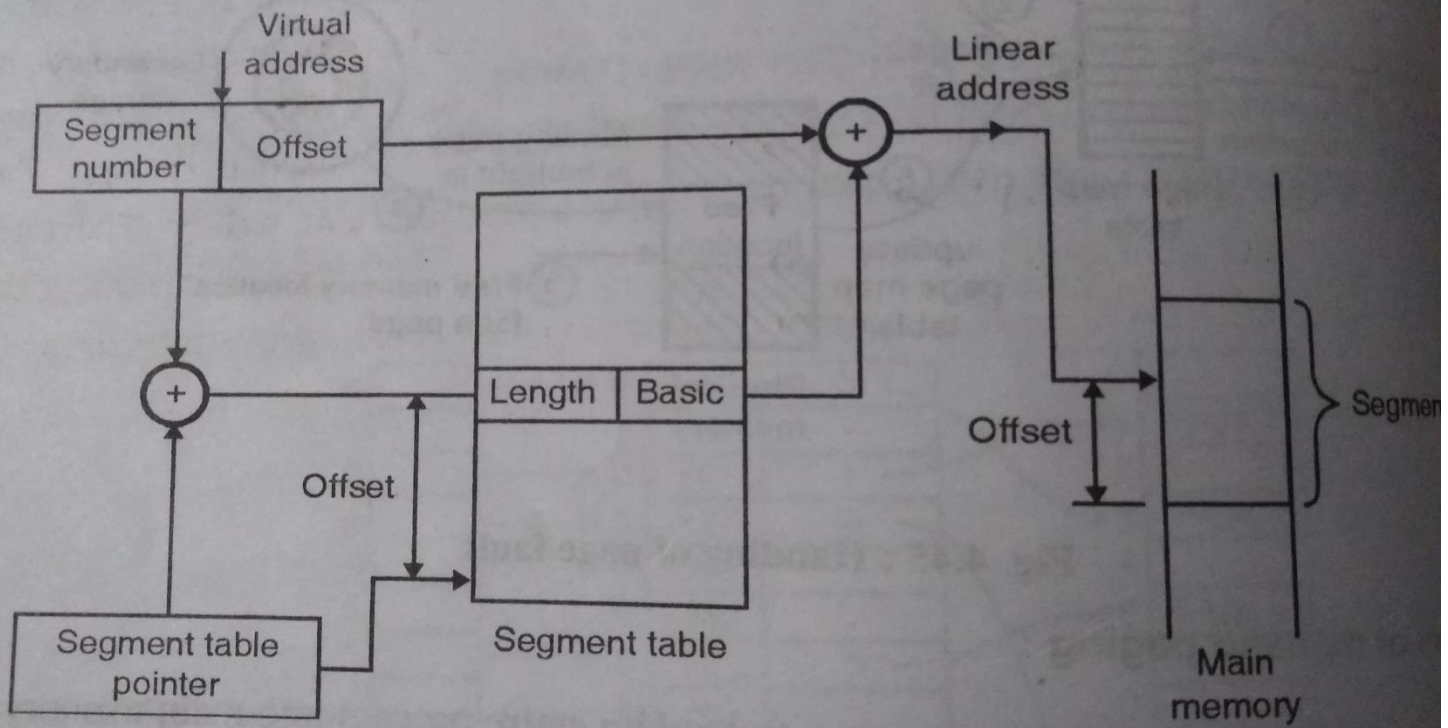


Fig. 4.46 : Address translation in a segmentation system

A typical segmentation system is shown in Fig. 4.46

Main Memory Allocation

- Memory is divided into set of contiguous locations called regions/segments/pages
- Store blocks of data
- Placement of blocks of information in memory is called **Memory Allocation**
- **Memory Management Systems** keeps information in a table containing available and free slots

Secondary Storage

- Magnetic disks
- Floppy disks
- Magnetic Tape
- RAID
- Optical Memory
- CD-ROM
- DVD

RAID

- Redundant Array of Independent Disks
- Redundant Array of Inexpensive Disks
- 6 levels in common use
- Not a hierarchy
- Set of physical disks viewed as single logical drive by O/S
- Data distributed across physical drives
- Can use redundant capacity to store parity information

RAID 0

- No redundancy
- Data striped across all disks
- Round Robin striping
- Increase speed
  - Multiple data requests probably not on same disk
  - Disks seek in parallel
  - A set of data is likely to be striped across multiple disks

RAID 1

- Mirrored Disks
- Data is striped across disks
- 2 copies of each stripe on separate disks
- Read from either
- Write to both
- Recovery is simple
  - Swap faulty disk & re-mirror
  - No down time
- Expensive

RAID 2

- Disks are synchronized
- Very small stripes
  - —Often single byte/word
- Error correction calculated across corresponding bits on disks
- Multiple parity disks store Hamming code error correction in corresponding positions
- Lots of redundancy
  - —Expensive
  - —Not used

RAID 3

- Similar to RAID 2
- Only one redundant disk, no matter how large the array
- Simple parity bit for each set of corresponding bits
- Data on failed drive can be reconstructed from surviving data and parity info
- Very high transfer rates

RAID 4

- Each disk operates independently
- Good for high I/O request rate
- Large stripes
- Bit by bit parity calculated across stripes on each disk
- Parity stored on parity disk

RAID 5

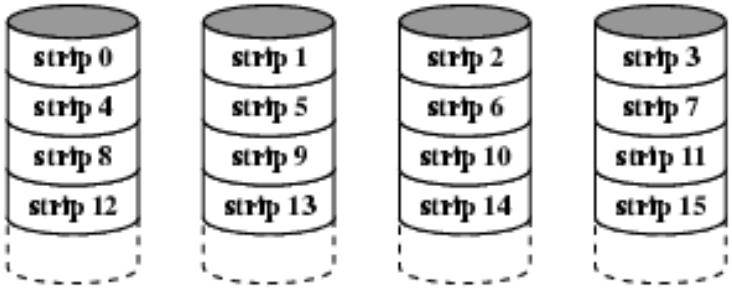
- Like RAID 4
- Parity striped across all disks
- Round robin allocation for parity stripe
- Avoids RAID 4 bottleneck at parity disk
- Commonly used in network servers

- N.B. DOES NOT MEAN 5 DISKS!!!!!

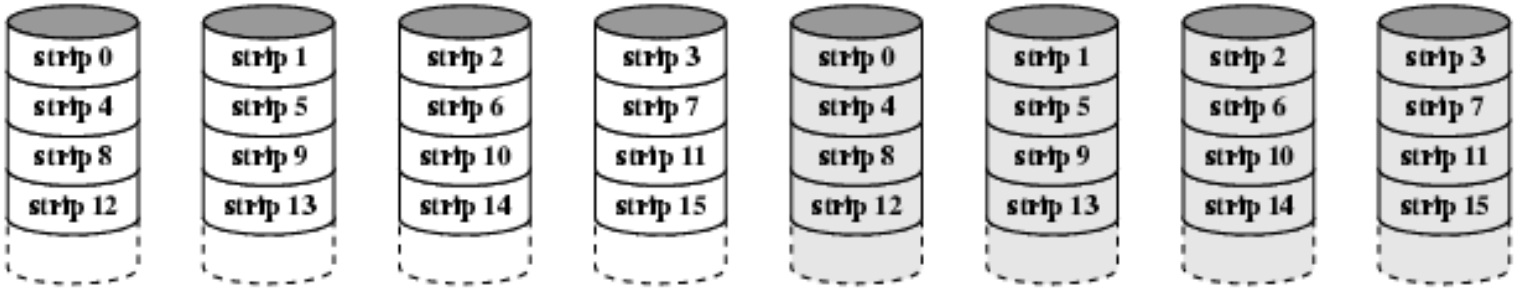
RAID 6

- Two parity calculations
- Stored in separate blocks on different disks
- User requirement of N disks needs N+2
- High data availability
  - Three disks need to fail for data loss
  - Significant write penalty

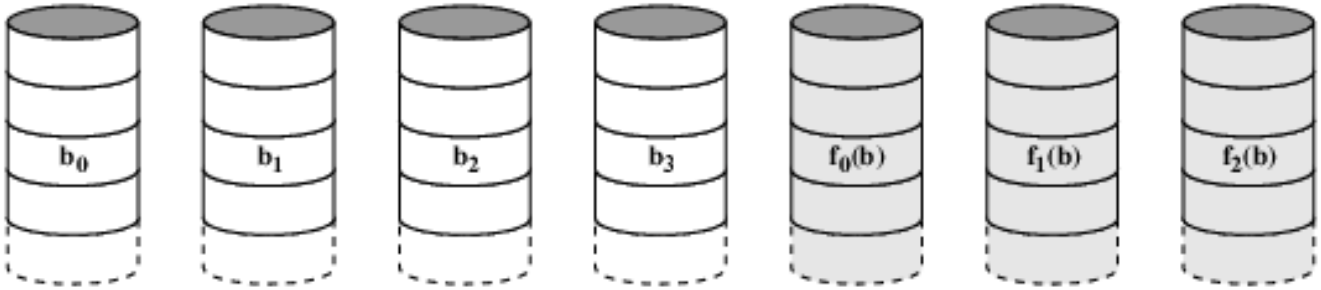
RAID 0, 1, 2



(a) RAID 0 (non-redundant)

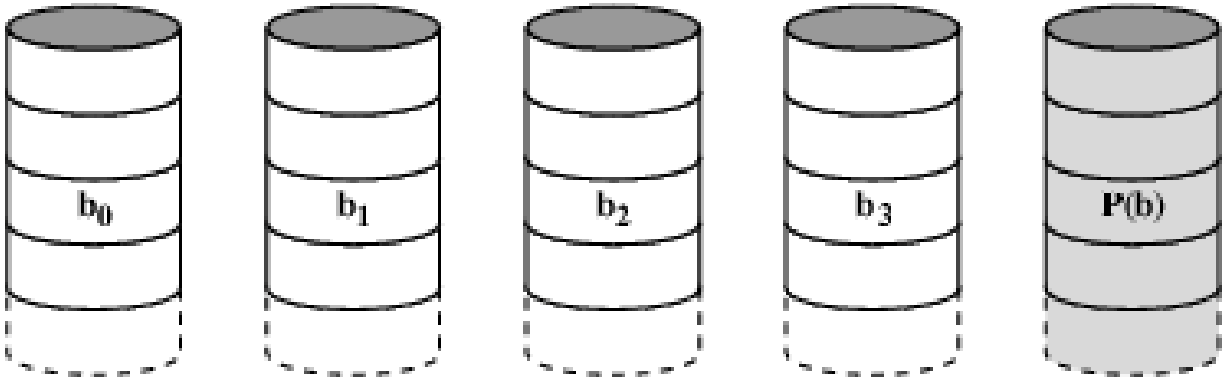


(b) RAID 1 (mirrored)

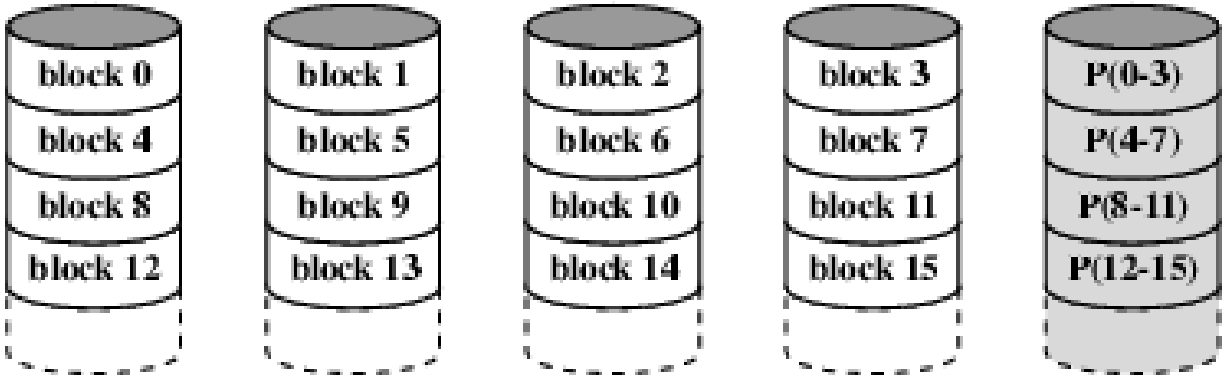


(c) RAID 2 (redundancy through Hamming code)

RAID 3 & 4

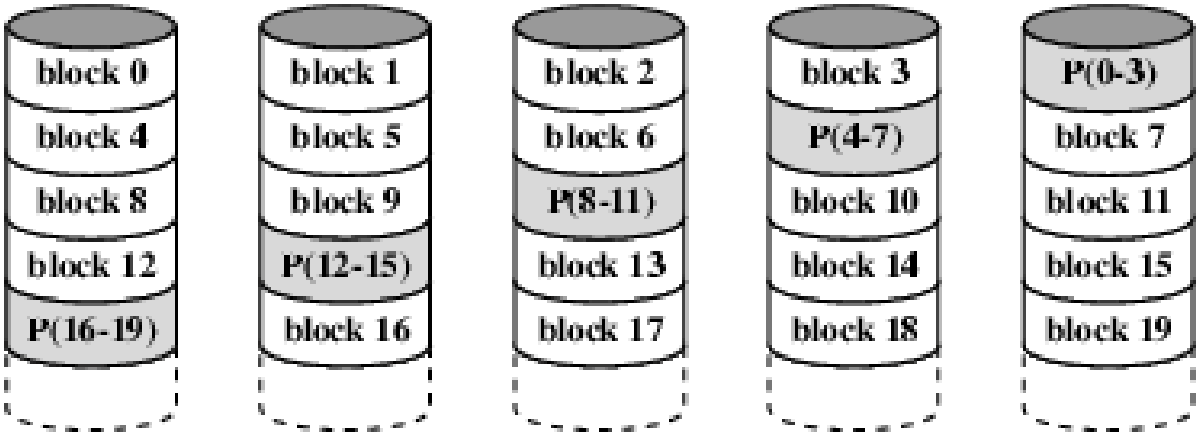


(d) RAID 3 (bit-interleaved parity)

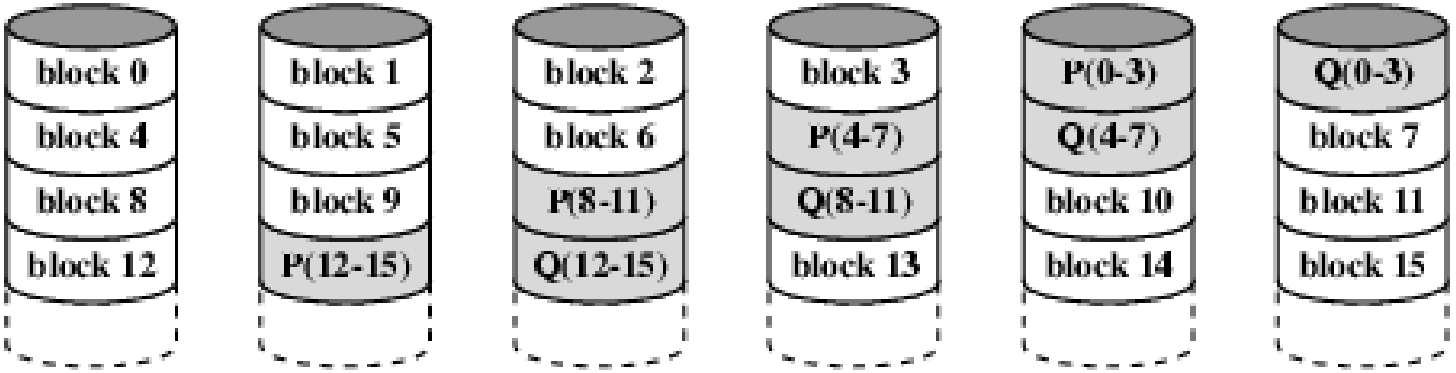


(e) RAID 4 (block-level parity)

RAID 5 & 6



(f) RAID 5 (block-level distributed parity)



(g) RAID 6 (dual redundancy)