Computer Architecture and Organization

Course Code CSE2003

Subject Guide: Dr. Aprna Tripathi

Course Objectives:

- The objectives of this course are:
  - to provide basic concepts of computer architecture and organization
  - to impart the knowledge of implementation of arithmetic operations in the computer.
  - to develop a deeper understanding of the hardware environment upon which all processing are carried out.
  - to provide knowledge about internals of memory system, interfacing techniques and subsystem devices.

Course Outcomes:

- A student who successfully fulfills the course requirements will be able to:

1. Identify and explain the building blocks of computer.
2. Recognize addressing modes, and data/instruction formats.
3. Perform the arithmetic operations using various algorithms and number systems.
4. Design the single cycle data path for an instruction format for a given architecture.
5. Compare various cache memory mapping techniques.
6. Explain memory control, direct memory access, interrupts, and memory organization.

Student Outcomes (SO):

a) An ability to apply the knowledge of mathematics, science and computing appropriate to the discipline

b) An ability to analyze a problem, identify and define the computing requirements appropriate to its solution.

c) An ability to design, implement and evaluate a system / computer-based system, process, component or program to meet desired needs

Assessment Pattern:

- At least one assignment for each Module
- At least one tutorial per module
- At least four quiz
- Quiz can be conducted before announcement
- In final assessment
  - Tutorial- 10
  - Assignment- 10
  - Quiz – 10
  - Group Activity – 05

Text Books:

The main text books are:

1. Computer Organization Authors: Carl Hamacher, Zvonko Vranesic and Safwat Zaky Publisher: McGraw Hill
2. Computer Organization and Architecture: Designing for Performance Authors: William Stallings Publisher: Prentice-Hall India

Reference Books:

1. David A. Patterson and John L. Hennessy “Computer Organization and Design-The Hardware/Software Interface”, Morgan Kaufmann, 5th edition, 2011.
2. James P Hayes, “Computer Architecture and Organization”, Mc Graw Hill, 3rd Edition, 2012, ISBN:9781259028564.

Module-2
(Memory Systems Hierarchy)

- In this Module, we have lectures on
  - Main memory organization
  - Types of Main memory : SRAM , DRAM and its characteristics and performance – latency –cycle time -bandwidth- memory interleaving
  - Cache memory: Address mapping-line size-replacement and policies-coherence
  - Virtual memory: Paging (Single level and multi level) – Page Table Mapping –
  - TLB Reliability of memory systems
  - Error detecting and error correcting systems.
  - Tutorial on design aspects of memory system

Lecture: Key Characteristics of Memories

Introduction to Memory

- Charles Babbage started Difference engine in1821 but failed its test in 1833, Why?

  Due to unavailability of Memory

- What is Memory?
- A single separate storage structure that holds information in the form of bits called as Memory
- The binary information may be instructions and data

https://youtu.be/XSkGY6LchJs

- Stored program concept was introduced with the advent of vacuum tubes by John Von Neumann 1940
- EDVAC (*Electronic Discrete Variable Automatic Computer*)

Key Characteristics of Memories

Location	Physical Characteristics	Unit of Transfer	Physical Type	Capacity	Performance	Access Methods
• CPU • Internal (Main) • External (Secondary)	• Erasable / non erasable • Volatile/ non-volatile	• Word • Block	• Semiconductor • Magnetic Surface • Optical	• Word Size • Number of Words	• Access Time • Cycle Time • Transfer Rate	• Sequential Access • Direct Access • Random Access • Associative Access

Location

With most visible aspect of memory, the memories are located in three areas of the computer.

CPU

One is in the form of CPU registers, which are used by CPU as its local memory.

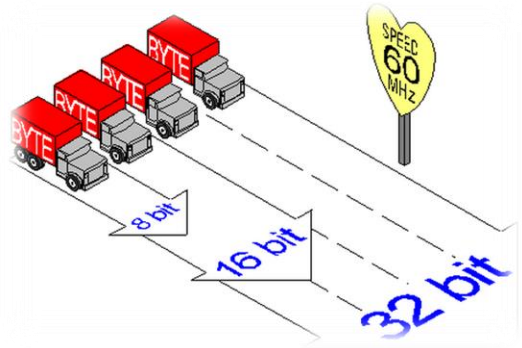
Internal(main)

The internal memory is often equated with main memory.

External (secondary)

The external memory consists of peripheral device, such as disk and tape that are accessible to CPU via I/O controllers.

Capacity



Word size

The capacity of the internal memory is typically expressed in terms of bytes or words.

Number of words

This capacity also depends on number of word in memory.

Unit of Transfer

For the internal memory, the unit of transfer is equal to the number of data lines into and out of the main memory module (word Length).

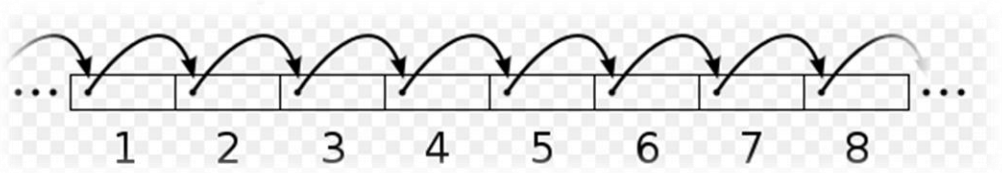
Word

The common word lengths are 8,16,1nd 32 bits.

Block

For external memory, the data often transferred in much longer units than a word, and these are referred to as Blocks.

Access methods



Sequential access

sequential access means that a group of elements is accessed in a predetermined, ordered sequence.

e.g. data in a memory array or a disk file or on a tape

Memory is organized into units of data, called Records. Access must be made in a specific linear sequence

In data structures, a data structure is said to have sequential access if one can only visit the values it contains in one particular order.

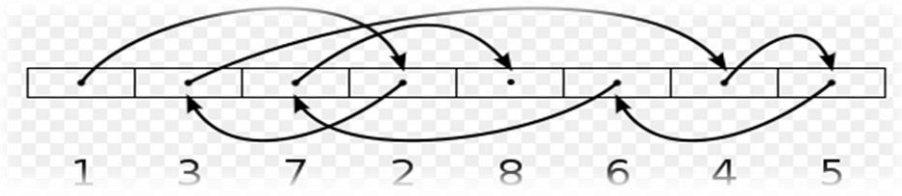
The canonical example is the linked list.

Direct access

Access is accomplished by general access to reach a general vicinity plus sequential searching, counting, waiting to reach the final location.

Again access time is variable.

Access methods



Random access

Each addressable location in the memory has unique

The time to access a given location is independent of the sequences of prior access and is constant.

Thus any location can be selected at random and directly addressed and accessed .

The main memory systems are a random access.

Associative access

word is retrieved based on a portion of its contents rather than its address

This is Random access type that enables one to make a comparison of desired bit locations within a word for specific match

Example: Cache memories

Performance

There are three parameters are used to measure the performance of the computer.

Access time

The time required to read/write the data from /into desired record is called Access time.

The access time is depends on amount of data to be read/write in the desired record .

If the amount data is uniform for all records then the access time is same for all records.

Latency

The time it takes to access a particular location

Cycle time

Is the combination access time plus time required to access particular location.

For Random access method ,this memory cycle time is same for all records

The sequential access and direct access ,the memory cycle time is different

Transfer rate/Throughput

The rate at which the data can be transferred into or out of a memory unit.

For random access memory it is equal to 1/cycle time

For non random access memory the following relationship holds:

$$Tn = Ta + (N/R)$$

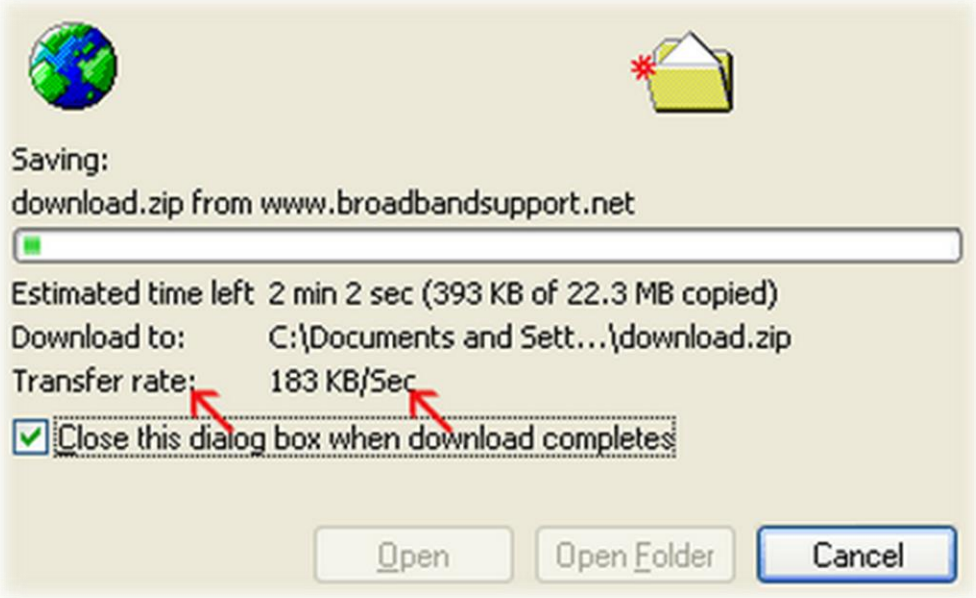
Where

Tn = Average time to read or write N bits

Ta =Average access time

N = Number of bits

R = Transfer rate ,in bits per second(BPS)



Physical type

Semiconductor

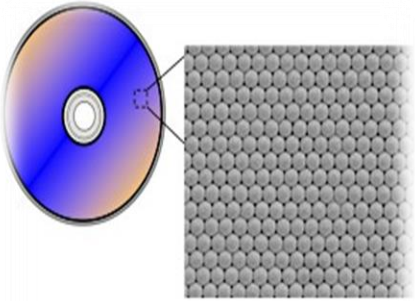
Semiconductor memory uses semiconductor-based integrated circuits to store information.

Magnetic surface

Magnetic storage uses different patterns of magnetization on a magnetically coated surface to store information.

Example:
Magnetic disk, Floppy disk (used for off-line storage)
Hard disk drive (used for secondary storage)

Optical



The typical optical disc, stores information in deformities on the surface of a circular disc
and reads this information by illuminating the surface with a laser diode and observing the reflection.

Physical characteristics

Erasable/non erasable

Erase the stored information by writing new information

Magnetic storage is erasable

Unable to erase the stored information

CD-R, Flash Memorories

Volatile/non volatile

Non volatile memory Will retain the stored information even if it is not constantly supplied with electric power.

Volatile Memory Requires constant power to maintain the stored information.

Magnetic storage is *non-volatile*.

Both *volatile* and *non-volatile* forms of semiconductor memory exist.

A type of non-volatile semiconductor memory known as flash memory

A type of volatile semiconductor memory is random access memory

Memory Organization (1/6)

- Recall:
  - Information is stored in the memory as a collection of bits.
  - Collection of bits are stored or retrieved simultaneously is called a word.
  - Number of bits in a word is called word length.
  - Word length can be 16 to 64 bits.
- Another collection which is more basic than a word:
  - Collection of 8 bits known as a “byte”
- Bytes are grouped into words, word length can also be expressed as a number of bytes instead of the number of bits:
  - Word length of 16 bits, is equivalent to word length of 2 bytes.
- Words may be 2 bytes (older architectures), 4 bytes (current architectures), or 8+ bytes (modern architectures).

Exercise

Find any five computer architecture and fill their details as per the following table

Year	Computer architecture	Word size *w*	Integer sizes	Floatingpoint sizes	Instruction sizes

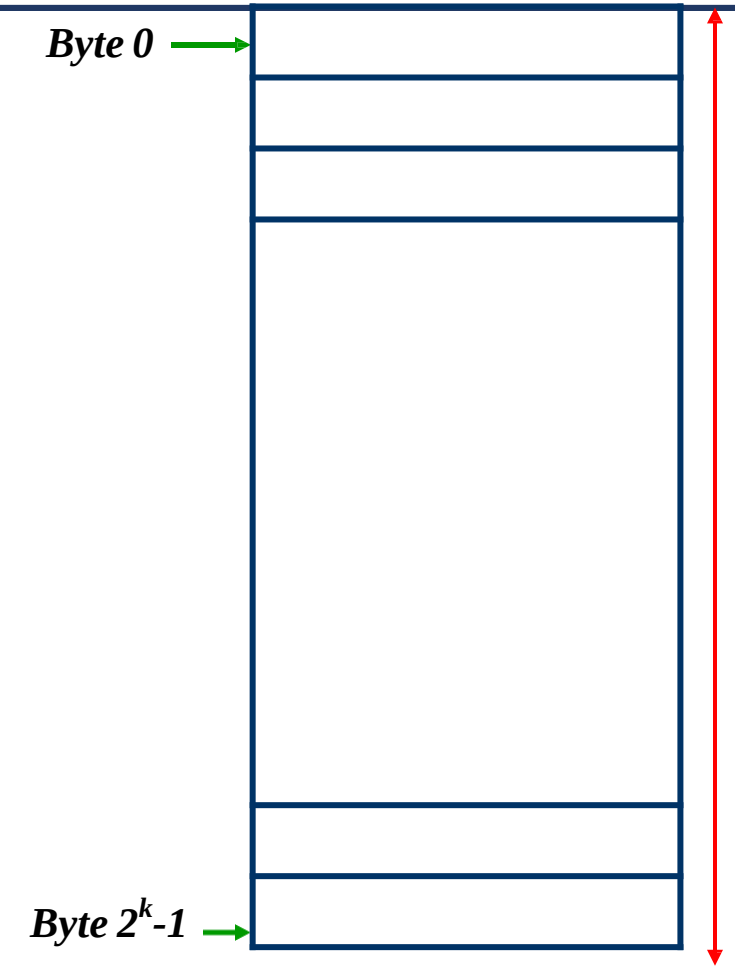
Memory Organization (2/6)

- Accessing the memory to obtain information requires specifying the "address" of the memory location.
- Recall that a memory has a sequence **of bits:**
  - Assigning addresses to each bit is impractical and unnecessary.
  - Typically, addresses are assigned to a single byte.
  - "Byte addressable memory"
- Suppose k bits are used to hold the address of a memory location:

Size of the memory in bytes is given by: 2^k
where k is the number of bits used to hold a memory address.
E.g., for a 16-bit address, size of the memory is 2^16= 65536 bytes

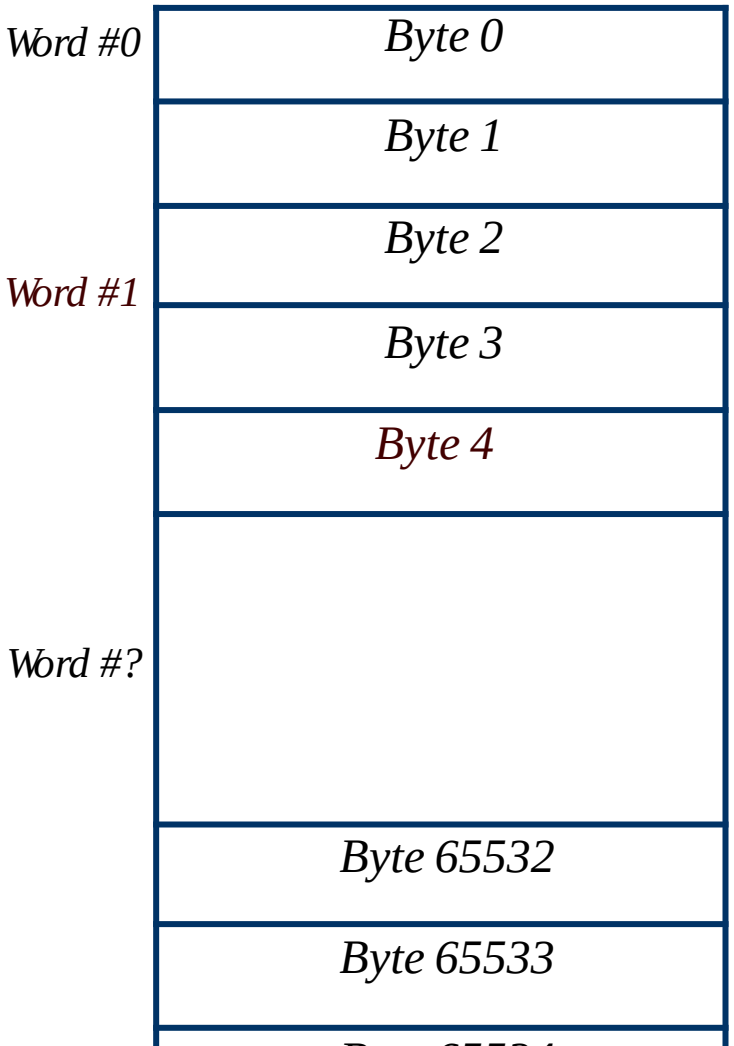
What is the size of the memory for a 24-bit address?

Memory Organization (3/6)



- **Memory is viewed as a sequence of bytes**.
  - Address of the first byte is 0
  - Address of the last byte is $2^k - 1$.
  - Where k is the number of bits used to hold memory address.
- E.g. when k = 16,
  - Address of the first byte is 0
  - Address of the last byte is 65535

Memory Organization (4/6)



Consider a memory organization:

- **16-bit** memory addresses

→ Size of the memory is = 2^16 =65536 B

→ [Byte 0..Byte 65535]

- Word length is **4 bytes**

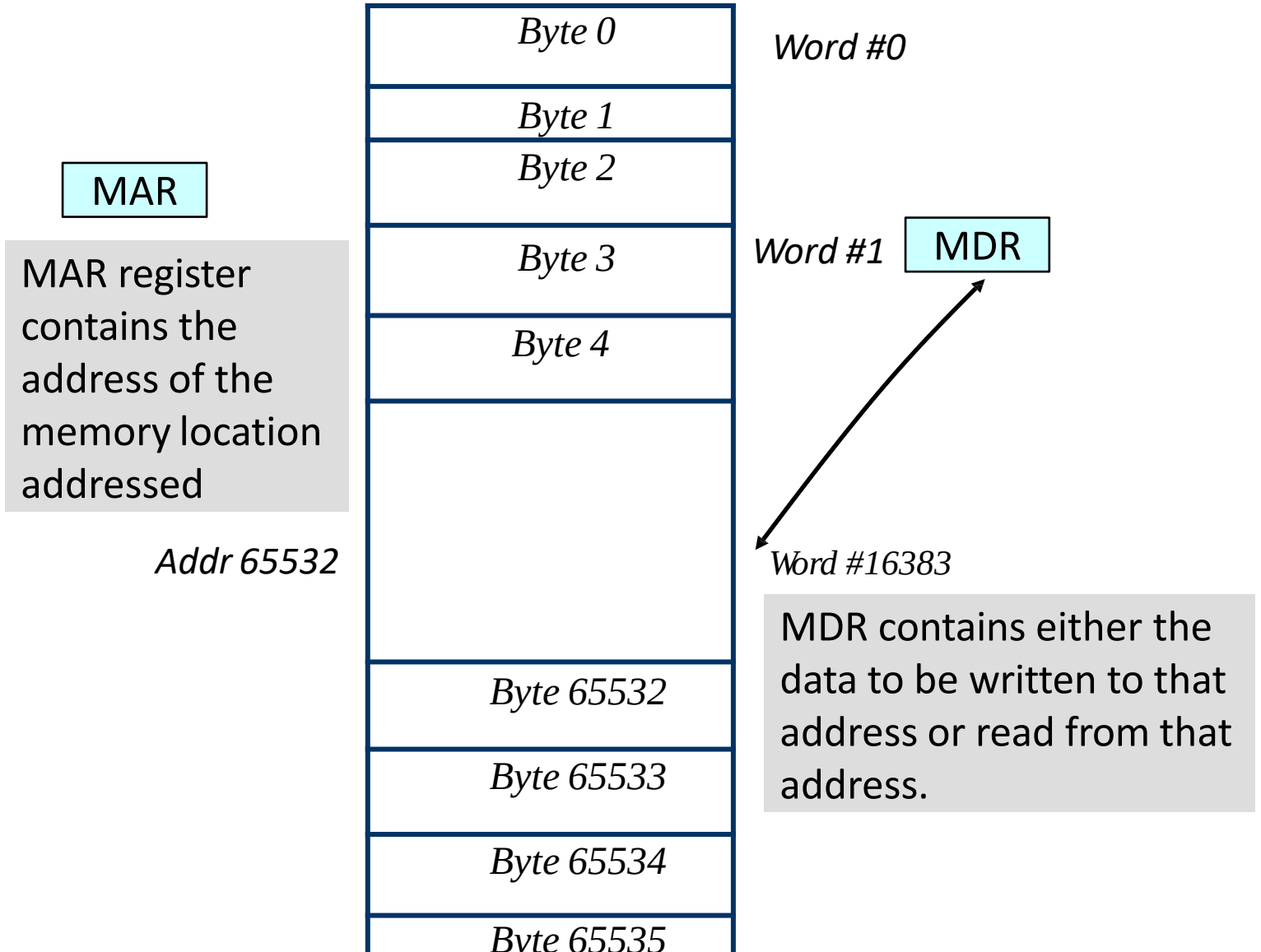
→**Number of words =** $\frac{\textbf{\textit{Memory size(bytes)}}}{\textbf{\textit{Word length(bytes)}}}$ **= ?**

=16384 words

- Word #0 starts at Byte #0.
- Word #1 starts at Byte #4.

→Last word (Word #?) starts at Byte#?

Memory Organization (5/6)



Memory Organization (6/6)

- Memory read or load:
  - Place address of the memory location to be read from, into MAR.
  - Issue a Memory_read command to the memory.
  - Data read from the memory is placed into MDR automatically (by control logic).

- Memory write or store:
  - Place address of the memory location to be written to into MAR.
  - Place data to be written into MDR.
  - Issue Memory_write command to the memory.
  - Data in MDR is written to the memory automatically (by control logic).

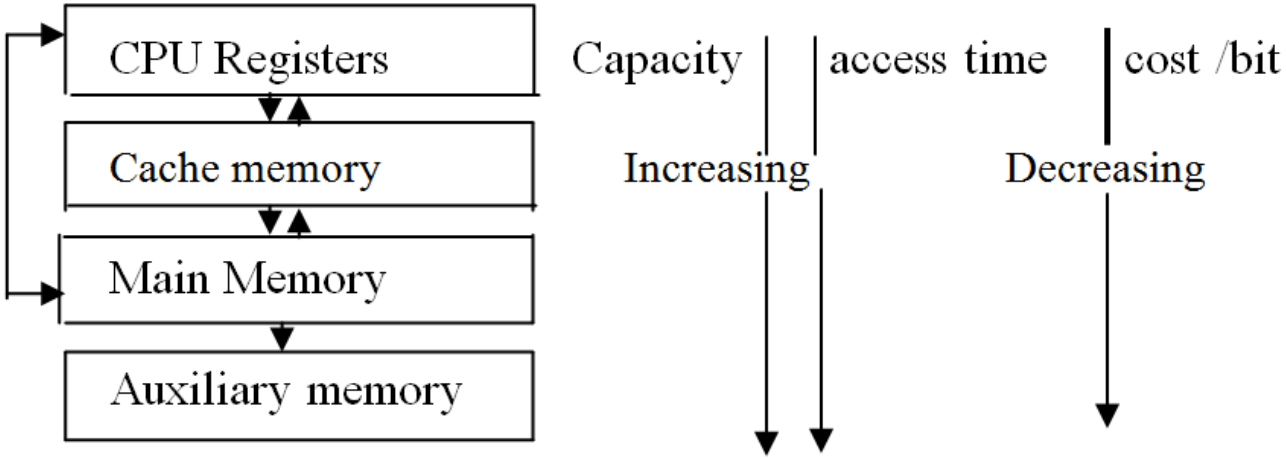
Memory Hierarchy (1/3)

- The memory unit is then an essential component in any digital computer since it is needed for **storing programs and data.**
- The memory unit that communicates directly with the CPU is called the **main memory**.
- Devices that provide backup storage are called **auxiliary memory (Magnetic disks and tapes).**
- Only programs and data currently needed by the processor reside in main memory.
- There is a special very – thigh – speed memory called **cache** employed between the CPU & main memory to **compensate for the mismatch in operating speeds.**

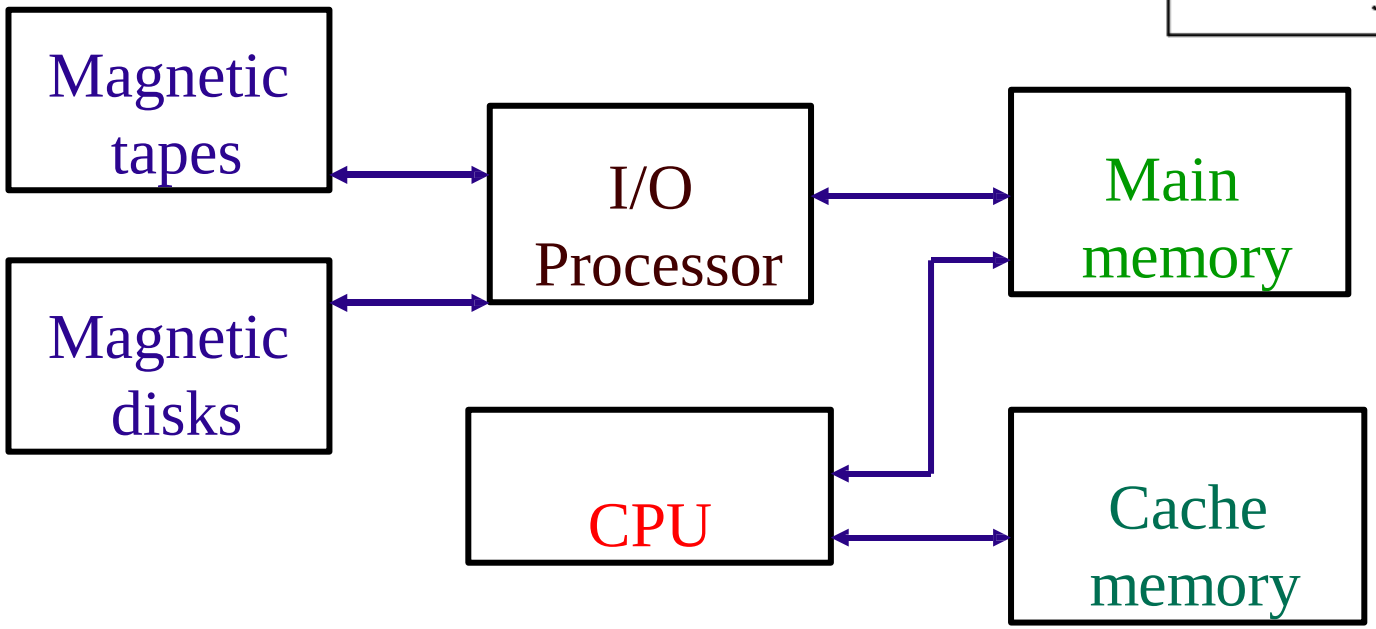
Memory Hierarchy (2/3)

- **Cache memory** is sometimes used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate (extremely fast), Whose access time is close to processor logic clock cycle time.
- It used for storing **segments of programs** currently being executed in the CPU and **temporary data frequently needed.**
- The cache organization is then concerned with the **transfer of information between a main memory and CPU.**
- The part of the computer system that supervises the flow of information between auxiliary memory and main memory is called **the memory management system.**

Memory Hierarchy (3/3)



Auxiliary memory
(Secondary)



There are different types of organization of $N^1 \times W^1$ –memory using N x W –bit chips

Case 1: If $N' > N$ & $W' = W$

Increase number of words by the factor of $p = \left\lceil \frac{N'}{N} \right\rceil$

Case 2: If $N' = N$ & $W' > W$

Increase the word size of a Memory by a factor of $q = \left\lceil \frac{W'}{W} \right\rceil$

Case 3: If $N' > N$ & $W' > W$

Increase number of words by the factor of p &
Increase the word size of a Memory by a factor of q

How many 1024x 8 RAM chips are needed to provide a memory capacity of 2048 x 8?

Available Chip size = N x W = 1024 x 8

Available memory size = N' x W' = **2048 x 8**

$$p = \left\lceil \frac{N'}{N} \right\rceil = \left\lceil \frac{2048}{1024} \right\rceil = 2$$

Where $N' \geq N$

$$q = \left\lceil \frac{W'}{W} \right\rceil = \left\lceil \frac{8}{8} \right\rceil = 1$$

Where $W' \geq W$

p * q, N x W Chips are needed for **N' x W'** memory size

2 *1 =2, 1024 x 8 RAM Chips are needed for **2048x 8** memory size

How many 1024x 4 RAM chips are needed to provide a memory capacity of 1024 x 8?

Available Chip size = N x W = 1024 x 4

Available memory size = $N^{/}$ x $W^{/}$ = **1024 x 8**

$$p = \left\lceil \frac{N^{/}}{N} \right\rceil = \left\lceil \frac{1024}{1024} \right\rceil = 1$$

Where $N^{/} \geq N$

$$q = \left\lceil \frac{W^{/}}{W} \right\rceil = \left\lceil \frac{8}{4} \right\rceil = 2$$

Where $W^{/} \geq W$

p * q, N x W Chips are needed for **$N^{/}$ x $W^{/}$** memory size

1 * 2 =2, 1024 x 4 RAM Chips are needed for **1024x 8** memory size

How many 1024x 4 RAM chips are needed to provide a memory capacity of 2048 x 8?

Available Chip size = N x W = 1024 x 4

Available memory size = N' x W' = **2048 x 8**

$$p = \left\lceil \frac{N'}{N} \right\rceil = \left\lceil \frac{2048}{1024} \right\rceil = 2$$

Where $N' \geq N$

$$q = \left\lceil \frac{W'}{W} \right\rceil = \left\lceil \frac{8}{4} \right\rceil = 2$$

Where $W' \geq W$

p * q, N x W Chips are needed for **N' x W'** memory size

2 * 2 =4, 1024 x 4 RAM Chips are needed for **2048x 8** memory size

There are different types of organization of $N^1 \times W^1$ –memory using N x W –bit chips

How many 1024x 8 RAM chips are needed to provide a memory capacity of 2048 x 8?

Case 1: If $N' > N$ & $W' = W$

Increase number of words by the factor of $p = \left\lceil \frac{N'}{N} \right\rceil$

How many 1024x 4 RAM chips are needed to provide a memory capacity of 2048 x 8?

Case 2:

If $N' = N$ & $W' > W$

Increase the word size of a Memory by a factor of $q = \left\lceil \frac{W'}{W} \right\rceil$

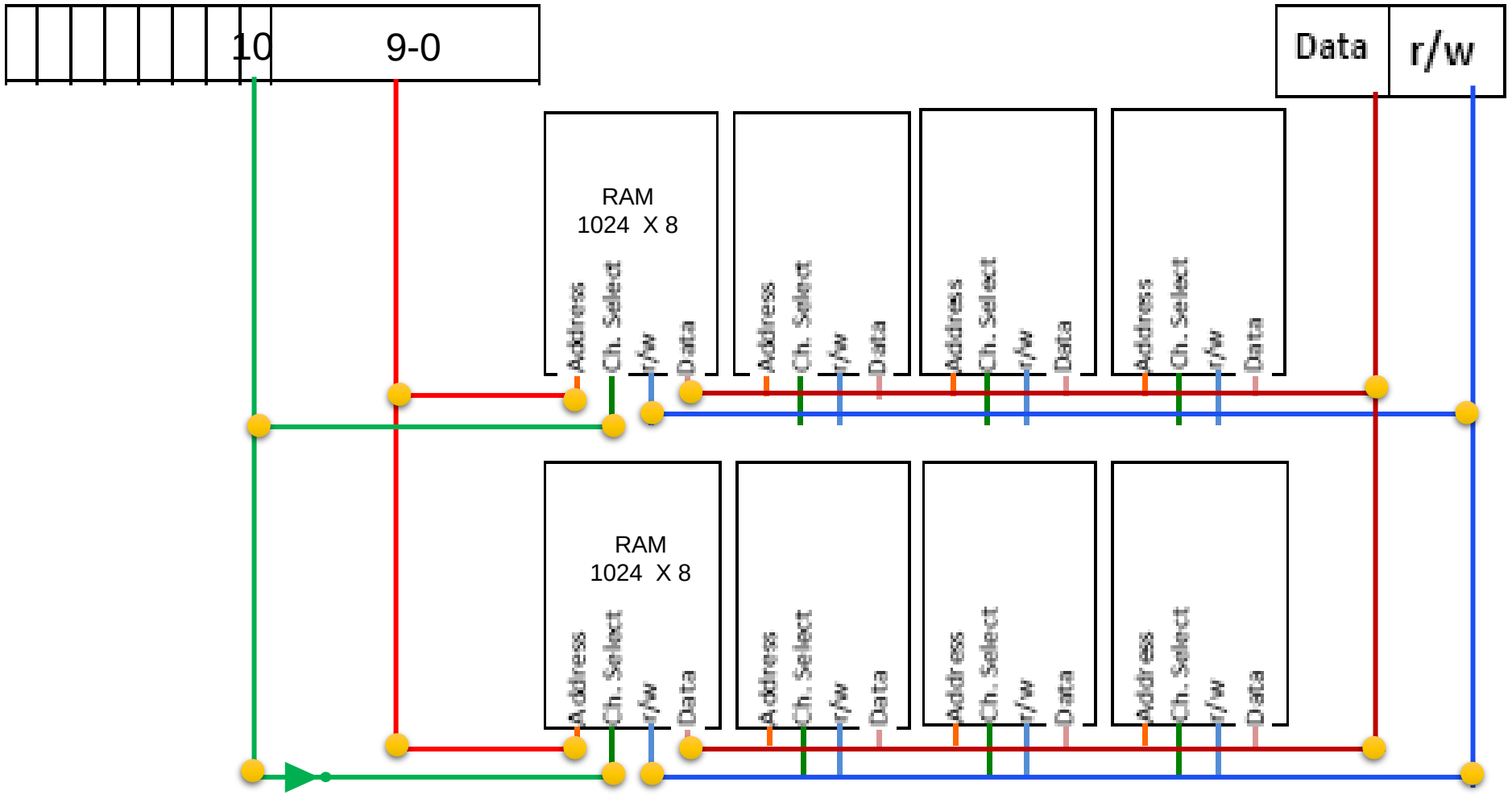
How many 1024x 4 RAM chips are needed to provide a memory capacity of 2048 x 8?

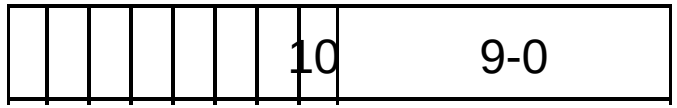
Case 3: If $N' > N$ & $W' > W$

Increase number of words by the factor of p &
Increase the word size of a Memory by a factor of q

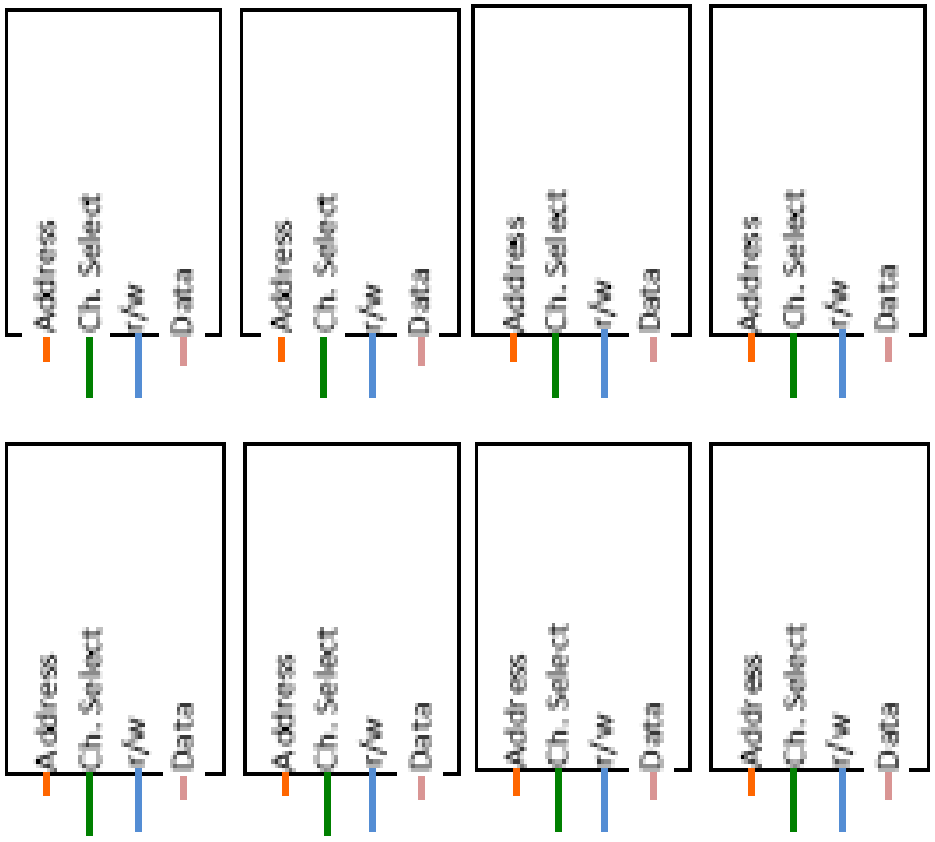
S.NO	Memory	N x W	N^1 x W^1	p	q	P * q	x	y	z	Total
1	RAM	1024 x 8	2048 x 8	2	1	2	10	1	0	11
2										
3										
4										

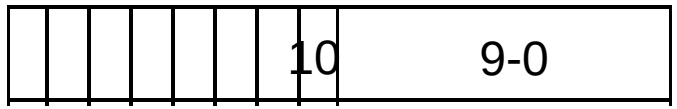
Component	Hexadecimal address		Address bus														
	From	To	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAM1	000	07F							**0**	x	x	x	x	x	x	x	x	x
RAM2	080	100							**1**	x	x	x	x	x	x	x	x	x

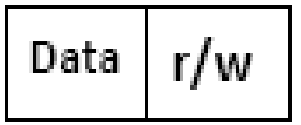

10
9-0
Data
r/w
RAM
1024 X 8
Address
Ch. Select
r/w
Data
RAM
1024 X 8
Address
Ch. Select
r/w
Data

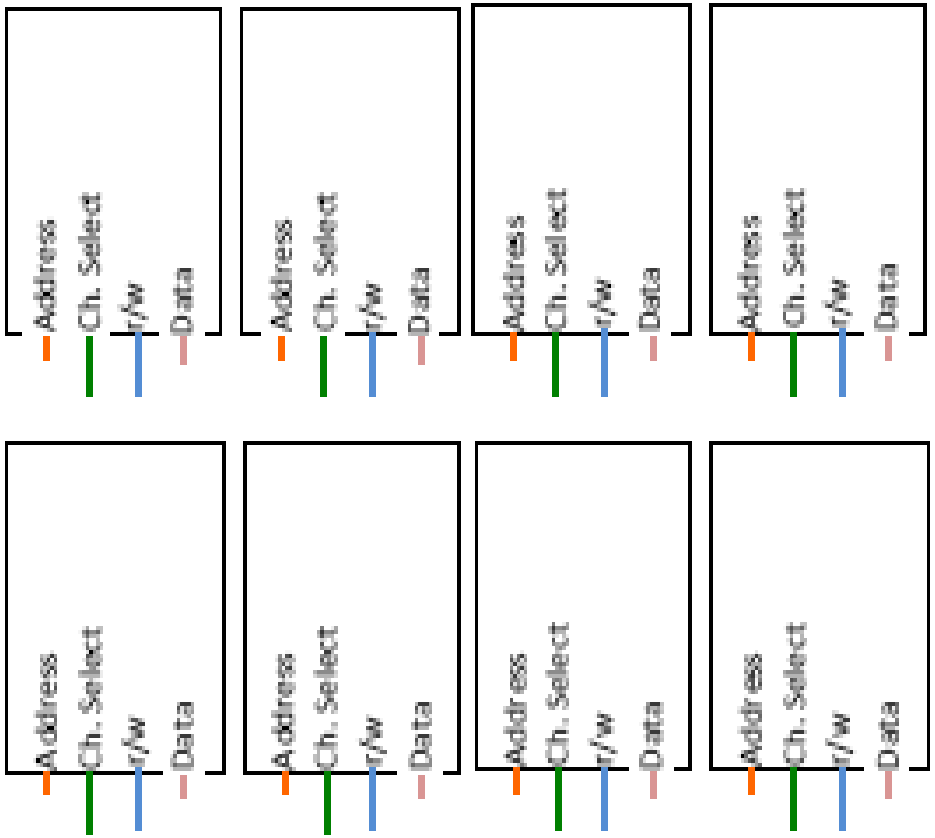

10
9-0


Data
r/w


Address
Ch. Select
r/w
Data
Address
Ch. Select
r/w
Data
Address
Ch. Select
r/w
Data
Address
Ch. Select
r/w
Data
Address
Ch. Select
r/w
Data
Address
Ch. Select
r/w
Data
Address
Ch. Select
r/w
Data
Address
Ch. Select
r/w
Data


10
9-0


Data
r/w


Address
Ch. Select
r/w
Data

Exercise

- How many 128 x 8 RAM chips are needed to provide a memory capacity of 2048 x 8?

How many 128 x 8 RAM chips are needed to provide a memory capacity of 2048 x 8?

Available Chip size = N x W = 128 x 8

Available memory size = N' x W' = **2048 x 8**

$$p = \left\lceil \frac{N'}{N} \right\rceil = \left\lceil \frac{2048}{128} \right\rceil = 16$$

Where $N' \geq N$

$$q = \left\lceil \frac{W'}{W} \right\rceil = \left\lceil \frac{8}{8} \right\rceil = 1$$

Where $W' \geq W$

p * q, N x W Chips are needed for **N' x W'** memory size

16 *1 =16,128 x 8 RAM Chips are needed for **2048x 8** memory size

Main Memory (1/4)

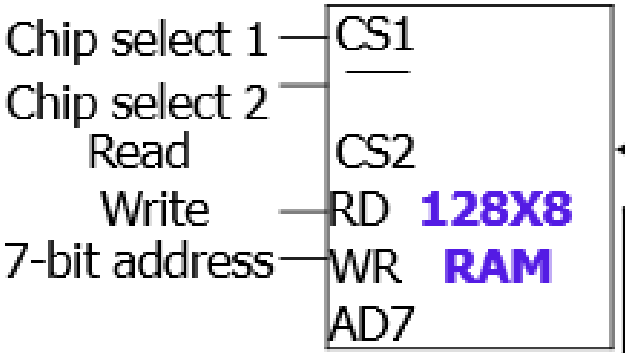
- The main 2 types of main memory are:
  - RAM which is to used to refer to a read/write random-access memory
  - ROM which is to used to refer to a read only random-access memory.
- Integrated circuit RAM chips are available in two possible operating modes, static and dynamic .
- The static RAM consists essentially of internal flip–flops that store the binary information, the stored information remains valid as long as power is applied to the unit.
- The dynamic RAM stores the binary information in the form of electric charges that are applied to capacitors.

Main Memory (2/4)

- The stored charge on the capacitors tend to discharge with time and the capacitors must be periodically recharged by refreshing the dynamic memory.
- The dynamic RAM offers reduced power consumption and larger storage capacity in single memory chip.
- But the static RAM is easier to use and has shorter read and write cycles.
- ROM is used for storing programs that are permanently resident in the computer.
- The ROM portion of main memory is needed for storing an initial program called a bootstrap loader.
- The bootstrap loader is a program whose function is the start the computer software operating when power is turned on.

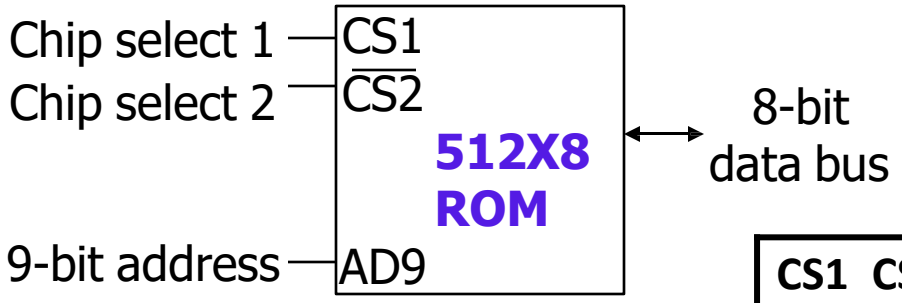
Main Memory (3/4)

- RAM and ROM chips are available in a variety of sizes.
- For example of a 1024 x 8 memory can be constructed with 128 x 8 RAM chips and 512 x 8 ROM chips.



CS1	$\overline{CS2}$	RD	WR	Memory function	State of data bus
0	0	x	x	Inhibit	High-impedance
0	1	x	x	Inhibit	High-impedance
1	0	0	0	Inhibit	High-impedance
1	0	0	1	Write	Input data to RAM
1	0	1	x	Read	Output data from RAM
1	1	x	x	Inhibit	High-impedance

Main Memory (4/4)



CS1	CS2	Memory function	State of data bus
0	0	Inhibit	High-impedance
0	1	Inhibit	High-impedance
1	0	Read	Output data from ROM
1	1	Inhibit	High-impedance

- The data bus can only be in an output mode.
- When chip is enabled by the 2 select inputs, the byte selected by the address lines appears on the data bus

Main Memory: Memory address map (1/4)

Memory address map

- The addressing of memory can be established by means of a **table , called a memory address map,** that specifies the **memory address assigned to each chip.**

- For example, assume that a computer system needs 512 bytes of RAM and 512 bytes of ROM.

- The **memory address map** for this configuration is shown in table following table.

Main Memory: Memory address map (2/4)

Memory Address Map for Microprocomputer

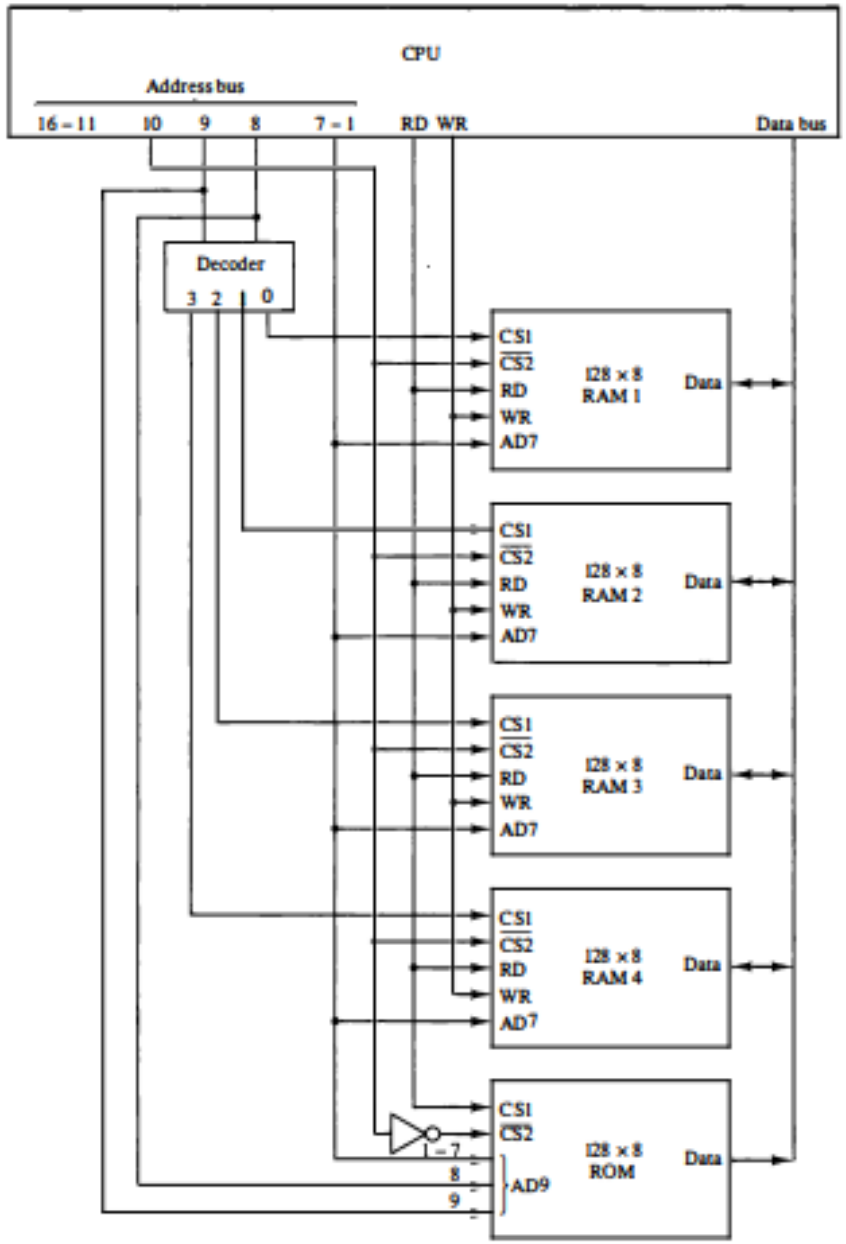
Component	Hexadecimal address	Address bus									
		10	9	8	7	6	5	4	3	2	1
RAM 1	0000-007F	0	0	0	X	X	X	X	X	X	X
RAM 2	0080-00FF	0	0	1	X	X	X	X	X	X	X
RAM 3	0100-017F	0	1	0	X	X	X	X	X	X	X
RAM 4	0180-01FF	0	1	1	X	X	X	X	X	X	X
ROM	0200-03FF	1	X	X	X	X	X	X	X	X	X

Main Memory: Memory address map (3/4)

- Although there are 16 lines in the address bus, the table shows only 10 lines because the other 6 are not used in this example and are assumed to be zero.
- The small x's under the address bus lines designate those lines that must be connected to the address inputs in each chip.
- The RAM chips have 128 bytes and need 7 address lines.
- The ROM chip has 512 bytes and needs 9 address lines.
- The x's are always assigned to the low – order bus lines:
  - **lines 1 through 7 for the RAM**
  - **and lines 1 through 9 for the ROM.**
- The distinction between a RAM and ROM address is done with another bus line. This can be done by choosing line 10 for this purpose.

Main Memory :
Memory address map (4/4)

Memory address map & Connection to CPU



Main Memory Practice
Example -1

- How many chips are necessary to implement a 4 MBytes memory:
- 1) using 64 Kbit SRAM;
- 2) using 1Mbit DRAM;
- 3) 64 KBytes using 64 Kbit SRAM and the rest using 1Mbit DRAM.

Solution

The number of chips is computed as:

$$\frac{\text{Memory capacity (expressed in bits)}}{\text{Chip capacity (expressed in bits)}}$$

or as

$$\frac{\text{Memory capacity (expressed in bytes)}}{\text{Chip capacity (expressed in bytes)}}$$

1)$n1 = 2^{22}/2^{13} = 512$ chips.(using the second formula)

2)$n2 = 2^{22}/2^{17} = 32$ chips.

3)$n3 = 2^{16}/2^{13} + \text{floor}((2^{22}-2^{16})/2^{17}) = 8 + 32$(SRAM + DRAM)

Cache Memory
(1/5)

- Analysis of a large number of typical programs has shown that the references to memory at any given interval of time tent to be confined within a few localized areas in memory.

- This phenomenon is known as the **property of locality of reference.**

- If the active portions of the program and data are placed in a fast small memory, the average memory access time can be reduced, thus reducing the total execution time of the program.

Cache Memory (2/5)

- The performance of cache memory is frequently measured in terms of a quantity called **hit ratio**.
- When the CPU refers to memory and finds the word in cache, it is said to produce **a hit.**
- If the word is not found in cache , it is in main memory and it counts as **a miss .**

- **No. of hits = number of successful cache memory references.**
- **Misses = number of unsuccessful cache memory references.**

Cache Memory (3/5)

- The ratio of the number of hits divided by the total CPU references to memory (hits plus misses) is the hit ratio,

  [Hit ratio = hits/ (hits + misses)]

- Average access time for a memory with a single-level cache formula. ***A*** **= *C* + (1-*H*)*M***
  - ***A*** is the average access time of the system
  - ***M*** is the access time of the main memory
  - ***C*** is the access time of the cache
  - ***H*** is the percent of accesses that are satisfied from the cache.
- Example, a computer with cache access time of 100 ns, a main memory access time of 1000 ns , and a hit ratio of 0.9, produces an average access time of 200 ns.

Exercise

Cache access time Tc = 100 ns
Memory access time Tm = 500 ns
If the effective access time is 10% greater than the cache access time, what is the hit ratio H?
(A) 89%
(B) 91%
(C) 98%
(D) 95%

Exercise

Cache access time Tc = 100 ns
Memory access time Tm = 500 ns
If the effective access time is 10% greater than the cache access time, what is the hit ratio H?
(A) 89%
(B) 91%
(C) 98%
(D) 95%

effective access time =cache hit ratio*cache access time+ cache miss ratio *(cache access time +main memory access time)
effective access time =10% greater the cache access time ==>110
let cache hit ratio is h
110 = h*100ns +(1-h) (100+500)
110= 100h+600-600h
500h= 490
h= 490/500= .98 = 98%

Exercise-2

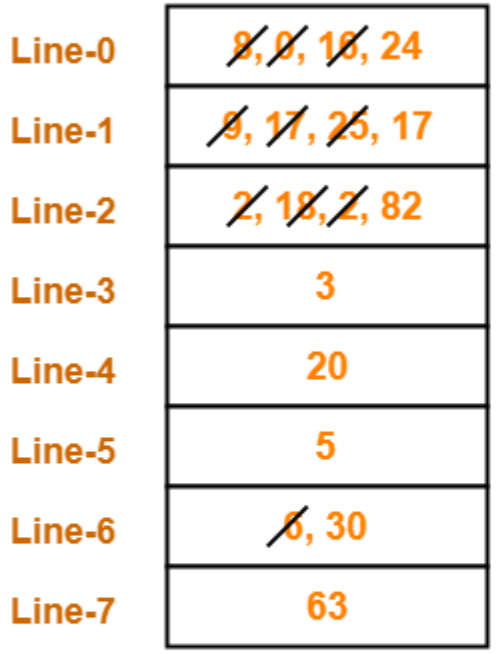
- Consider a direct mapped cache with 8 cache blocks (0-7). If the memory block requests are in the order-
- 3, 5, 2, 8, 0, 6, 3, 9, 16, 20, 17, 25, 18, 30, 24, 2, 63, 5, 82, 17, 24
- Which of the following memory blocks will not be in the cache at the end of the sequence?
- 1. 3
- 2. 18
- 3. 20
- 4. 30
- Also, calculate the hit ratio and miss ratio.

Solution

- We have,
- There are 8 blocks in cache memory numbered from 0 to 7.
- In direct mapping, a particular block of main memory is mapped to a particular line of cache memory.
- The line number is given by-
- Cache line number = Block address modulo Number of lines in cache
-
- For the given sequence-
- Requests for memory blocks are generated one by one.
- The line number of the block is calculated using the above relation.
- Then, the block is placed in that particular line.
- If already there exists another block in that line, then it is replaced.

Solution

- Thus,
- Out of given options, only block-18 i
- Option-(B) is correct.
- Hit ratio = 3 / 20
- Miss ratio = 17 / 20



Cache Memory

Blocks requests in order
Calculation of line number

3 % 8 = 3	(Miss)
5 % 8 = 5	(Miss)
2 % 8 = 2	(Miss)
8 % 8 = 0	(Miss)
0 % 8 = 0	(Miss)
6 % 8 = 6	(Miss)
3 % 8 = 3	(Hit)
9 % 8 = 1	(Miss)
16 % 8 = 0	(Miss)
20 % 8 = 4	(Miss)
17 % 8 = 1	(Miss)
25 % 8 = 1	(Miss)
18 % 8 = 2	(Miss)
30 % 8 = 6	(Miss)
24 % 8 = 0	(Miss)
2 % 8 = 2	(Miss)
63 % 8 = 7	(Miss)
5 % 8 = 5	(Hit)
82 % 8 = 2	(Miss)
17 % 8 = 1	(Miss)
24 % 8 = 0	(Hit)

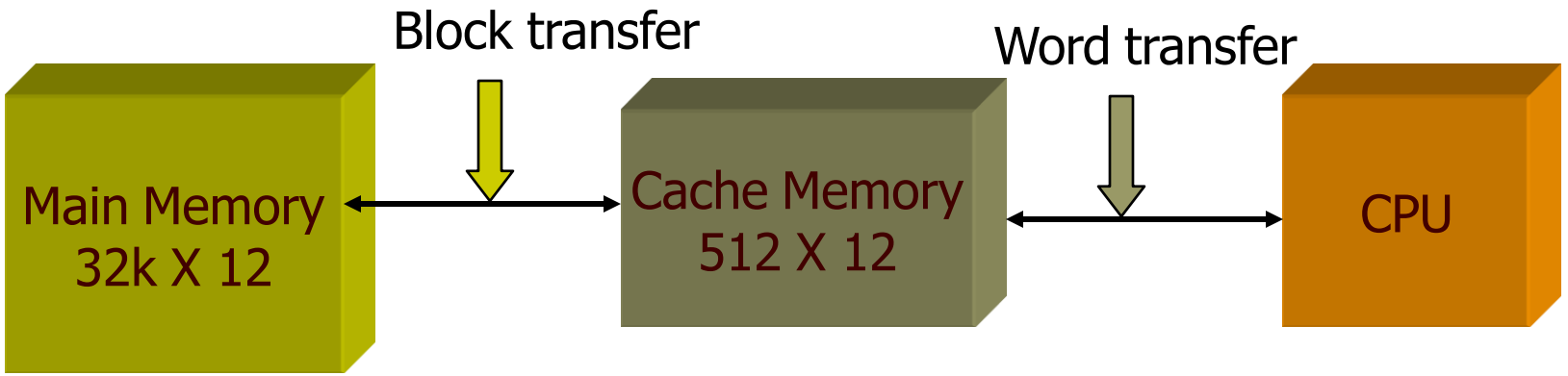
3, 5, 2, 8, 0, 6, 3, 9, 16, 20, 17, 25, 18, 30, 24, 2, 63, 5, 82, 17, 24

Cache Memory (4/5)

- The transformation of data from main memory to cache memory is referred to as **a mapping process**.
- Three types of mapping procedures are of practical interest when considering the organization of cache memory.
  - Direct mapping.
  - Associative mapping.
  - Set-associative mapping.

Cache Memory (5/5)

- Small amount of fast memory
- Sits between normal main memory and CPU
- May be located on CPU chip or module
- For every word stored in cache, there is a duplicate copy in main memory.
- To illustrate these 3 mapping procedures, a specific example of a memory organization will be used as shown.



Example of Cache Memory

Exercise

How many address lines and data lines are required to provide a memory capacity of 16K×16?

1.10,4

2.16,16

3.14,16

4.4,16

Solution

ROM memory size = 2^m x n

m = no. of address lines

n = no. of data lines

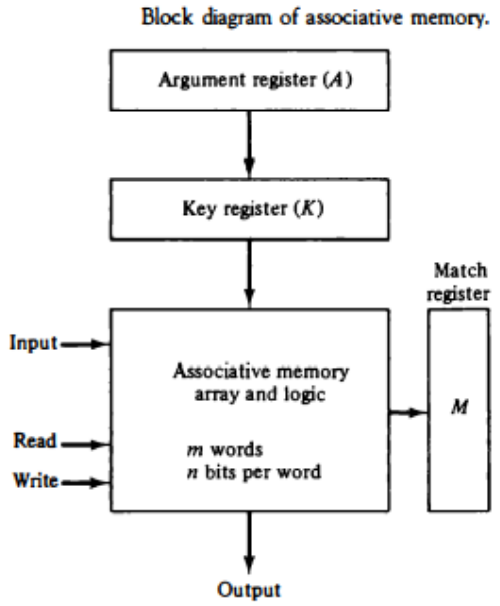
Given, 16K x 16 = 2^{14} x 16

Address lines = 14

Data lines = 16

Cache Memory : Associative Mapping (1/2)

- The fastest and most flexible cache organization uses an
  - **Associative Mapping**, as illustrated in the following diagram.
  - Associative mapping cache (all numbers in octal)
    - **CPU address (15 bits)**



Argument register

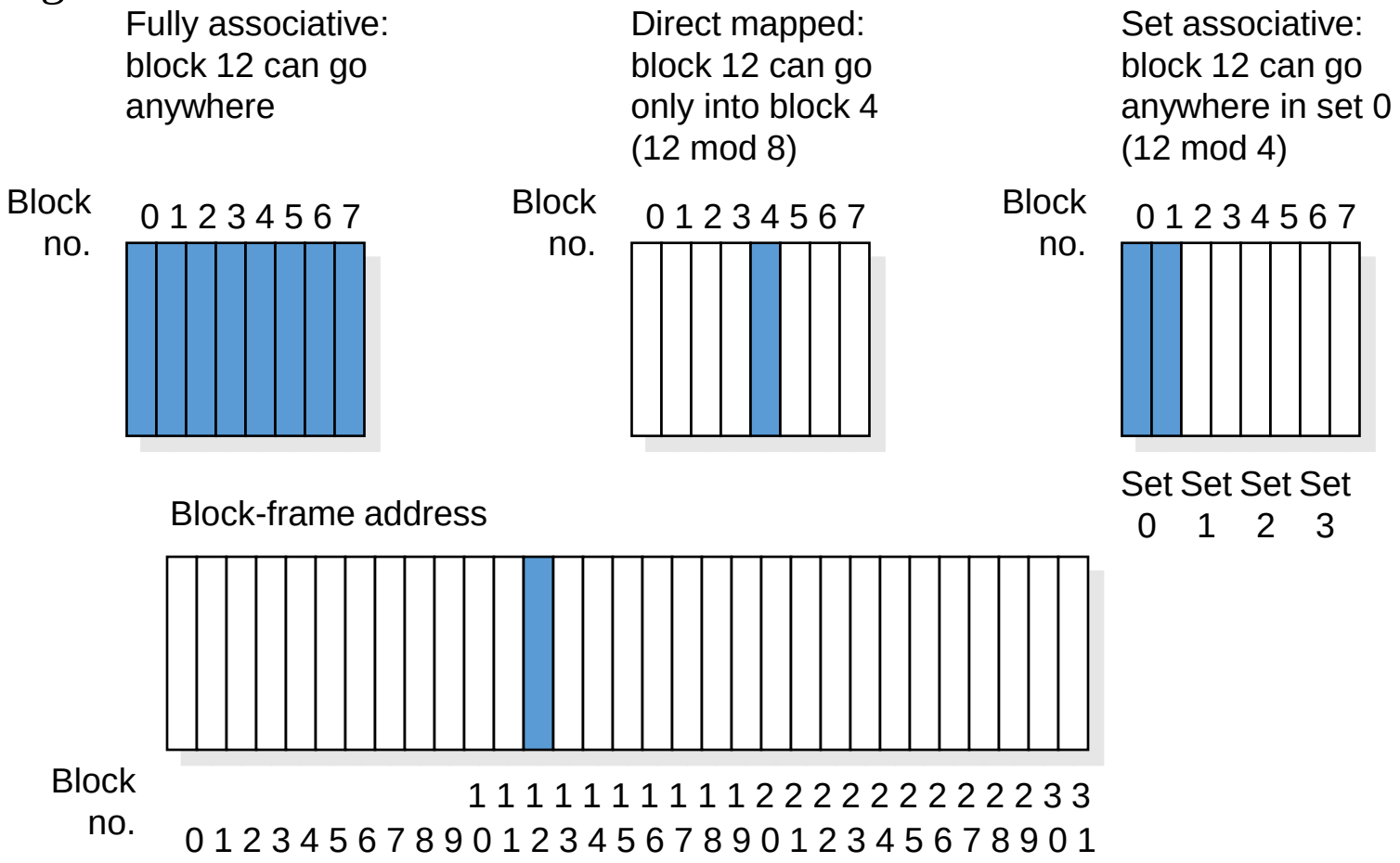
←15-bit Address→	←12-bit Data→
01000	3450
02777	6710
22345	1234

Cache Memory: Associative Mapping (2/2)

- The **associative memory** stores both the address and content (data) of the memory word.
- This permits any location in cache to store any word from main memory.
- A CPU address of **15** bits is placed in the **argument register** and the **associative memory** is searched for a matching address.
- If the address is found , the corresponding 12- bit data is read and sent to the CU.
- If no match occurs, the main memory is accessed for the word.
- The **address-data pair** is then transferred to the **associative cache** memory based on a **replacement algorithm**.

Associative Caches

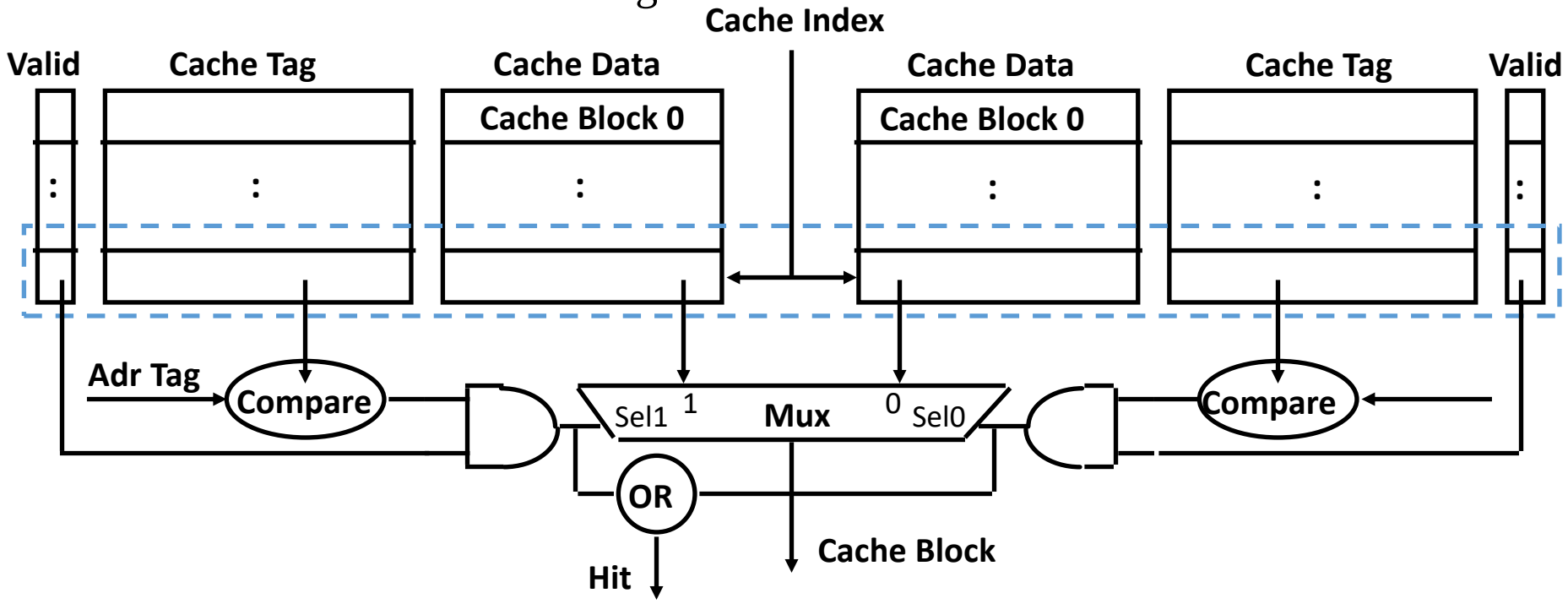
- Block 12 placed in 8 block cache:
  - Fully associative, direct mapped, 2-way set associative
  - S.A. Mapping = Block Number Modulo Number Sets



Set Associative Cache

- N-way set associative: N entries for each Cache Index
  - N direct mapped caches operates in parallel
- Example: Two-way set associative cache
  - Cache Index selects a “set” from the cache
  - The two tags in the set are compared to the input in parallel
  - Data is selected based on the tag result

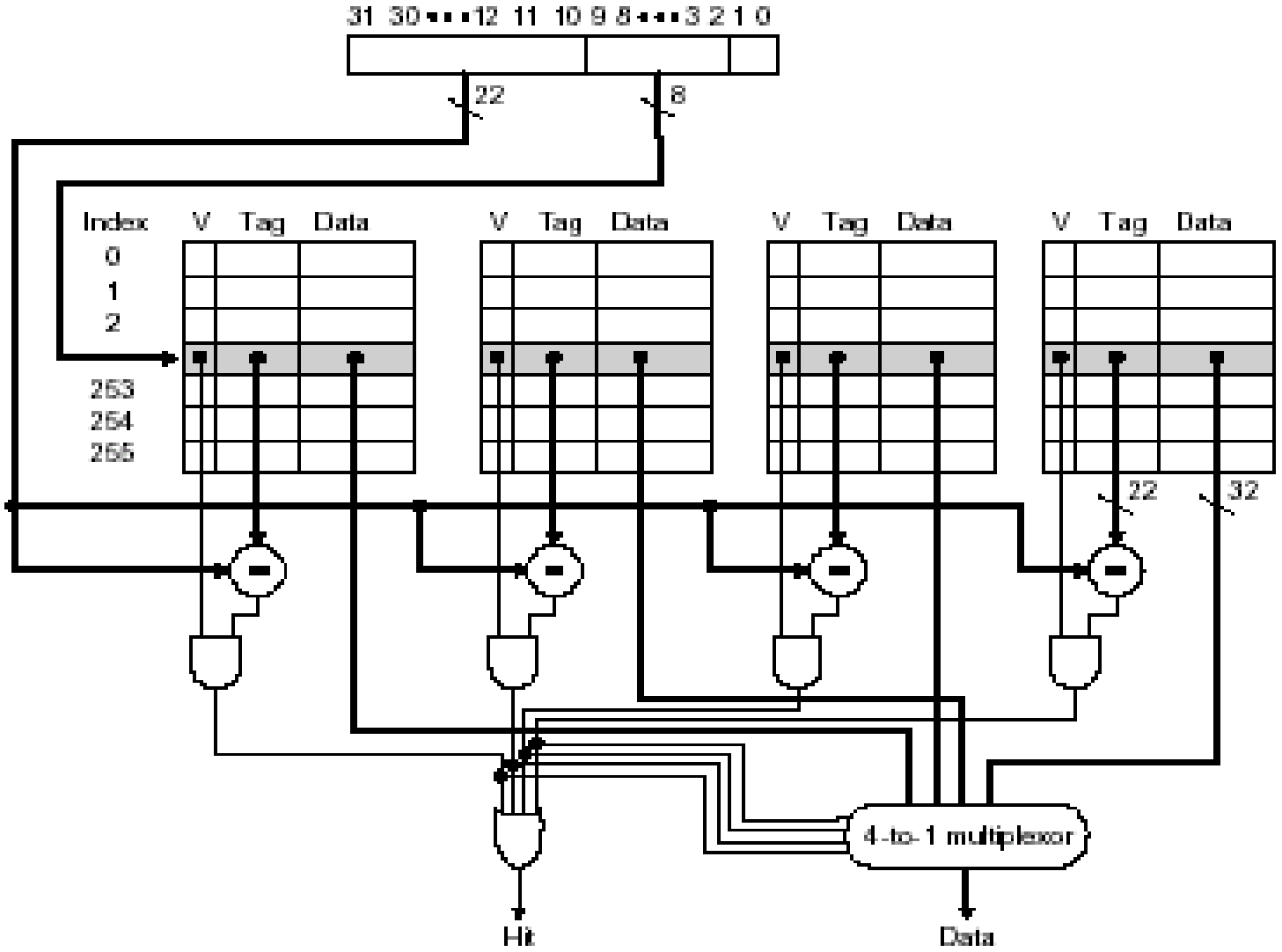
Each data word is stored together with its tag and the number of tag-data items in one word of cache is said to form a set.



2-ways Set Associative Mapping

Index	Tag	Data	Tag	Data
000	0 1	3 4 5 0	0 2	5 6 7 0
777	0 2	6 7 1 0	0 0	2 3 4 0

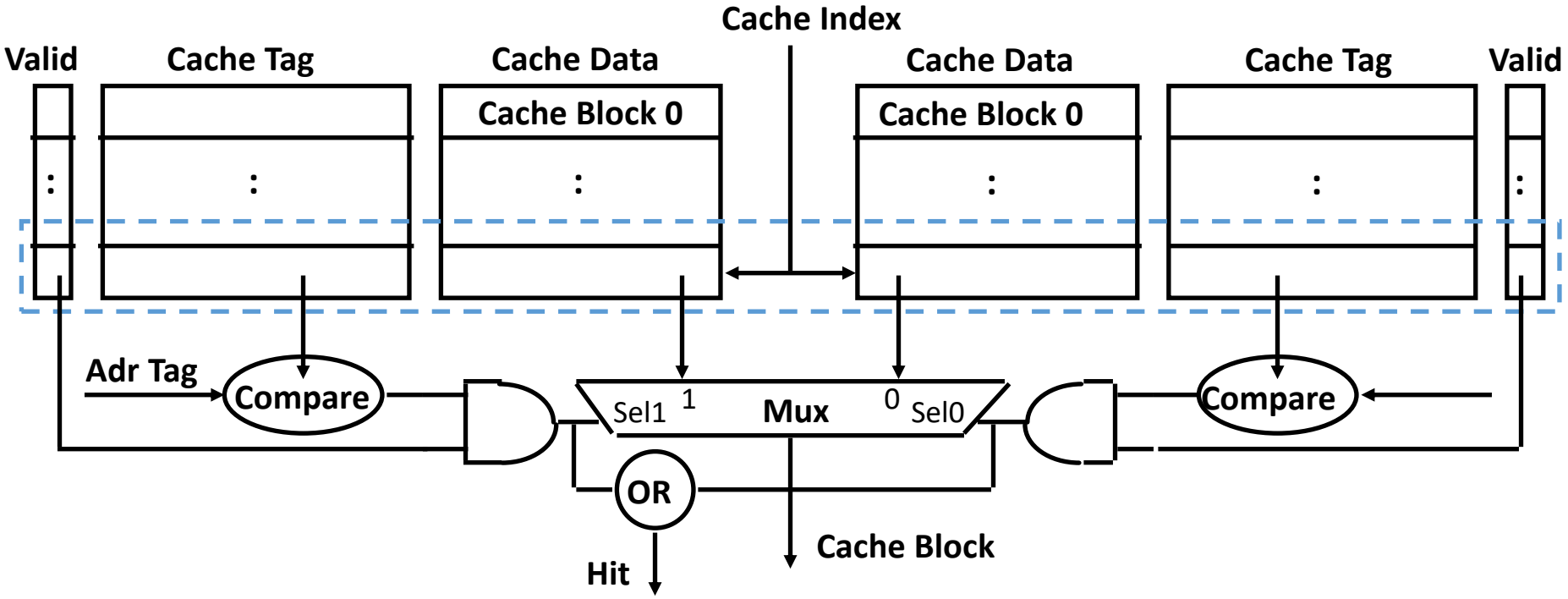
Example: 4-way set associative Cache



What is the cache size in this case ?

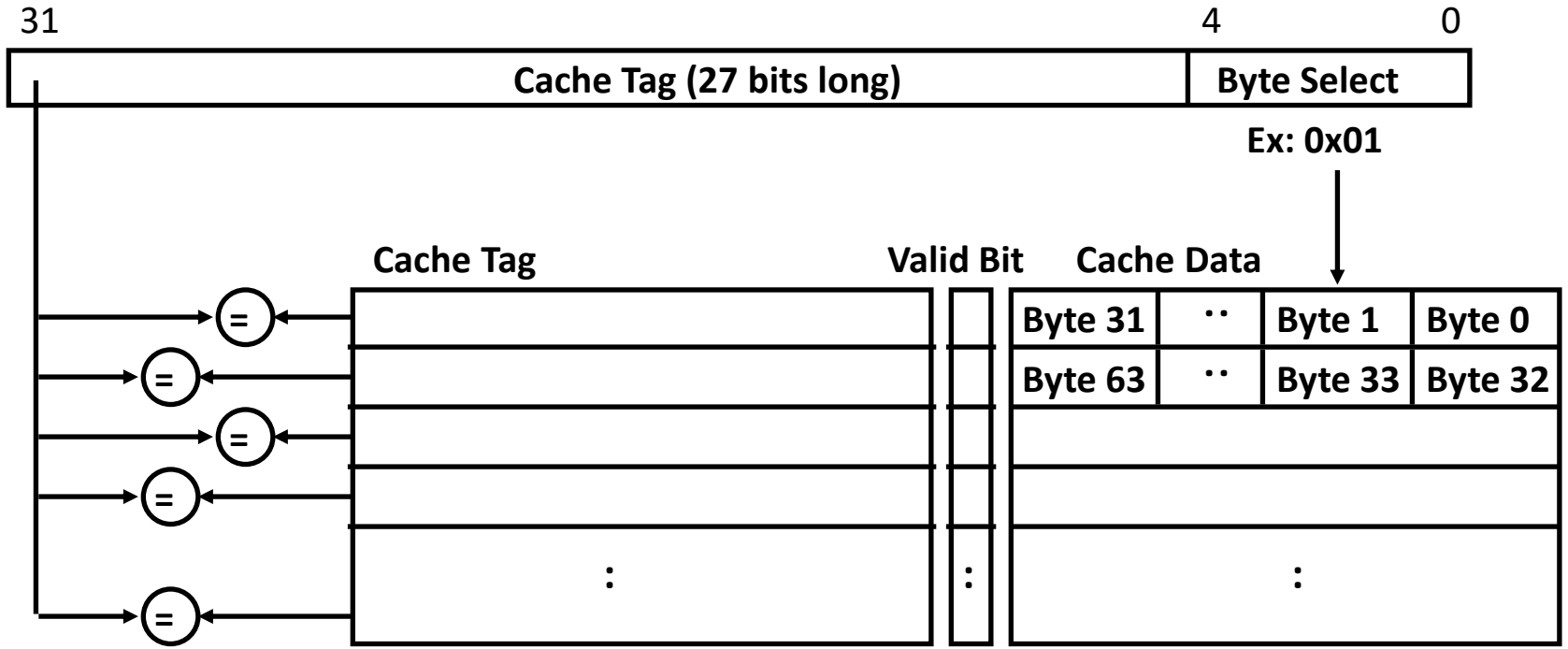
Disadvantages of Set Associative Cache

- N-way Set Associative Cache versus Direct Mapped Cache:
  - N comparators vs. 1
  - Extra MUX delay for the data
  - Data comes AFTER Hit/Miss decision and set selection
- In a direct mapped cache, Cache Block is available BEFORE Hit/Miss:



Fully Associative Cache

- Fully Associative Cache
  - Forget about the Cache Index
  - Compare the Cache Tags of all cache entries in parallel
  - Example: Block Size = 32 B blocks, we need N 27-bit comparators
- By definition: Conflict Miss = 0 for a fully associative cache



Exercise based on set associative mapping

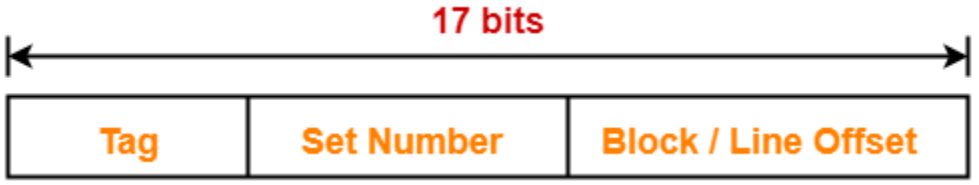
- Consider a 2-way set associative mapped cache of size 16 KB with block size 256 bytes. The size of main memory is 128 KB. Find-
  - Number of bits in tag
  - Tag directory size

Solution

- Given-
- Set size = 2
- Cache memory size = 16 KB
- Block size = Frame size = Line size = 256 bytes
- Main memory size = 128 KB

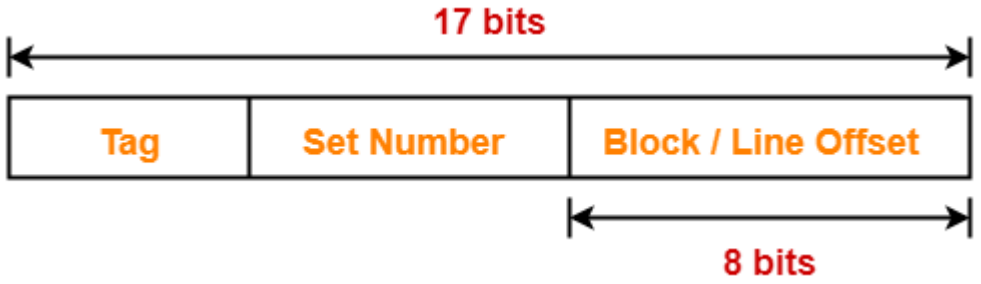
Solution

Number of Bits in Physical Address-



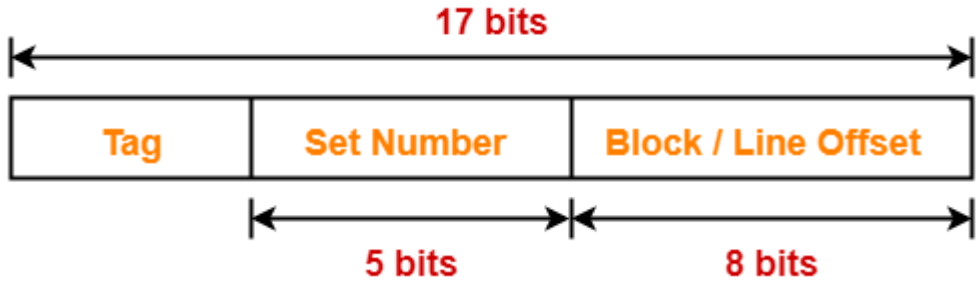
We have,
Size of main memory = 128 KB = 2^{17} bytes
Thus, Number of bits in physical address = 17 bits

- **Number of Bits in Block Offset-**
- We have,
- Block size = 256 bytes = 2^8 bytes
- Thus, Number of bits in block offset = 8 bits



Solution

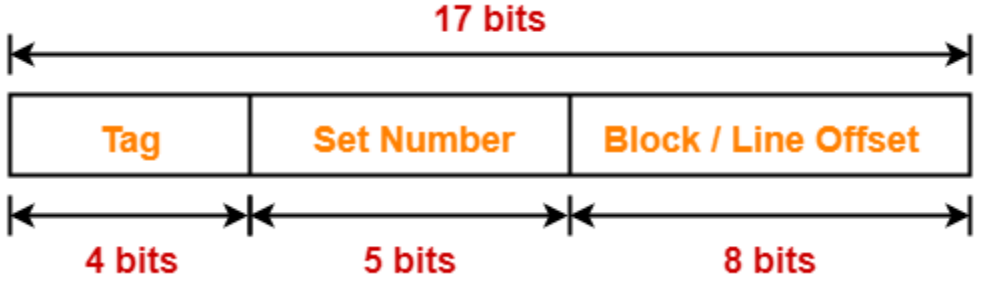
- **Number of Lines in Cache-**
- Total number of lines in cache
- = Cache size / Line size = 16 KB / 256 bytes = 2^{14} bytes / 2^{8} bytes = 64 lines
- Thus, Number of lines in cache = 64 lines



- **Number of Sets in Cache-**
- Total number of sets in cache
- = Total number of lines in cache / Set size = 64 / 2 = 32 sets = 2^{5} sets
- Thus, Number of bits in set number = 5 bits

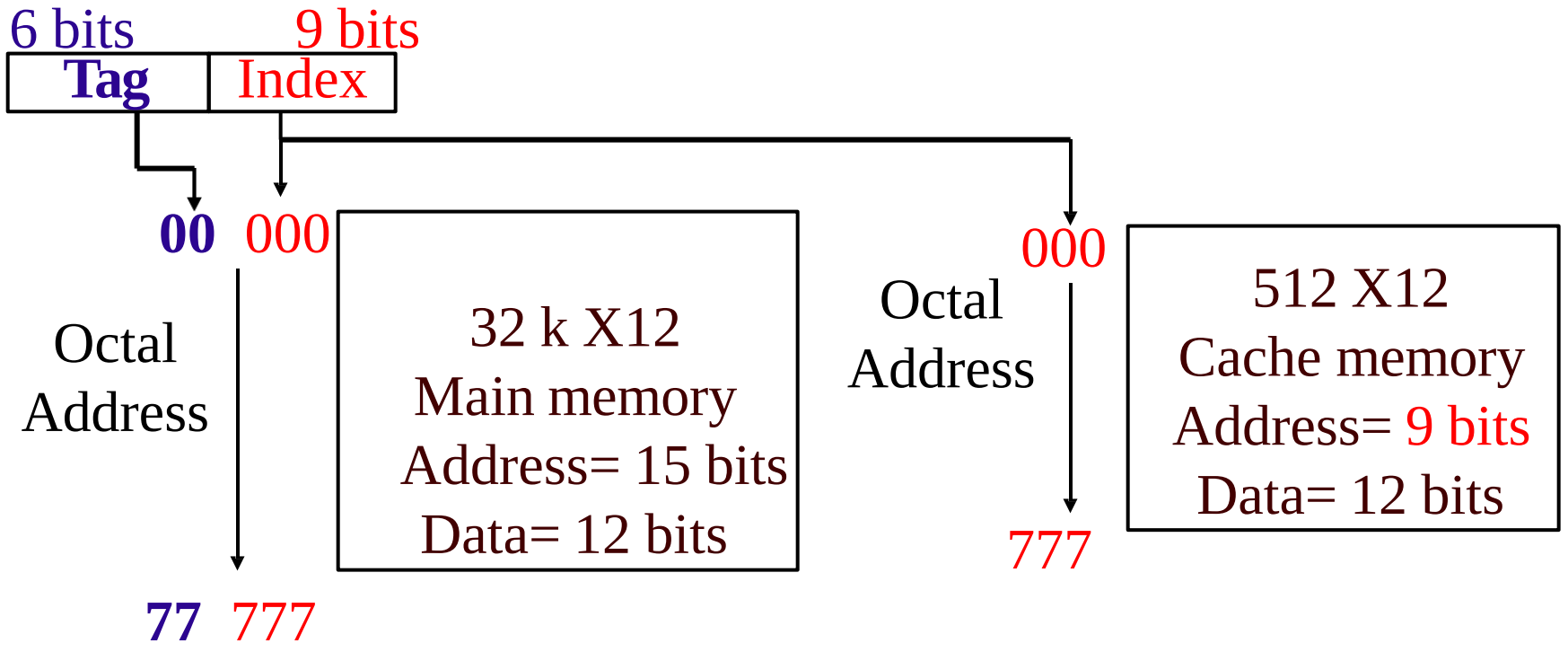
Solution

- **Number of Bits in Tag-**
- Number of bits in tag
- = Number of bits in physical address – (Number of bits in set number + Number of bits in block offset)
- = 17 bits – (5 bits + 8 bits) = 17 bits – 13 bits = 4 bits
- Thus, Number of bits in tag = 4 bits
- **Tag Directory Size-**
- Tag directory size
- = Number of tags x Tag size
- = Number of lines in cache x Number of bits in tag = 64 x 4 bits = 256 bits = 32 bytes
- Thus, size of tag directory = 32 bytes



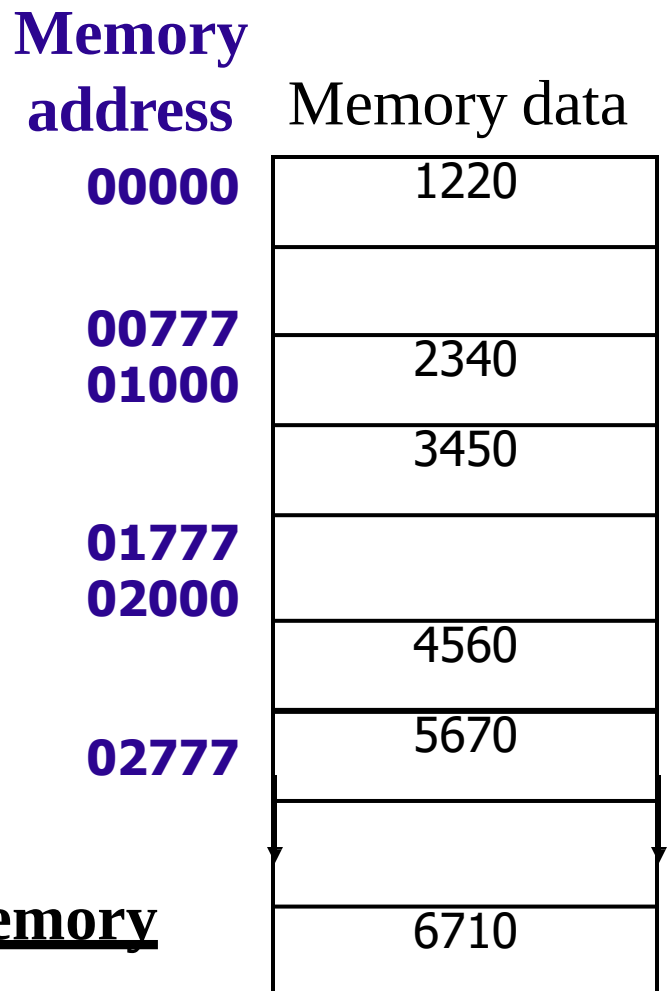
Cache Memory: Direct mapping (1/5)

- Associative memories are expensive compared to random access memories, which can be used for the cache, because of the added logic associated with each cell.
- This organization is shown in the following diagram:

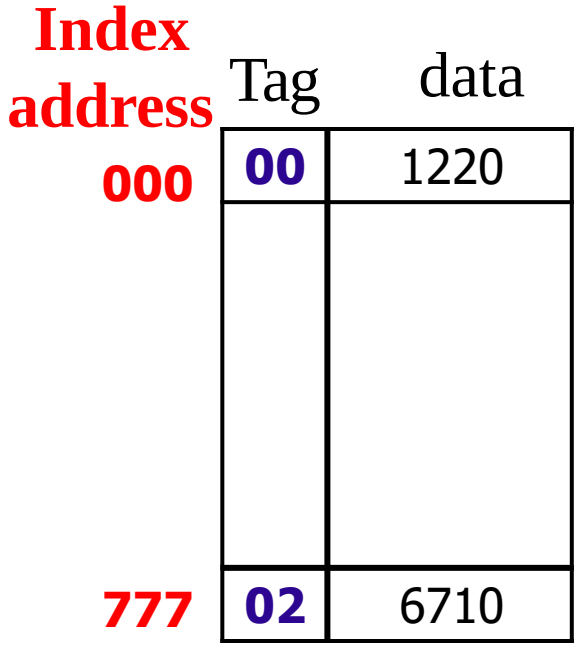


Cache Memory: Direct mapping (2/5)

1. Direct mapping cache Organization: Block size of one word (simplest cache)

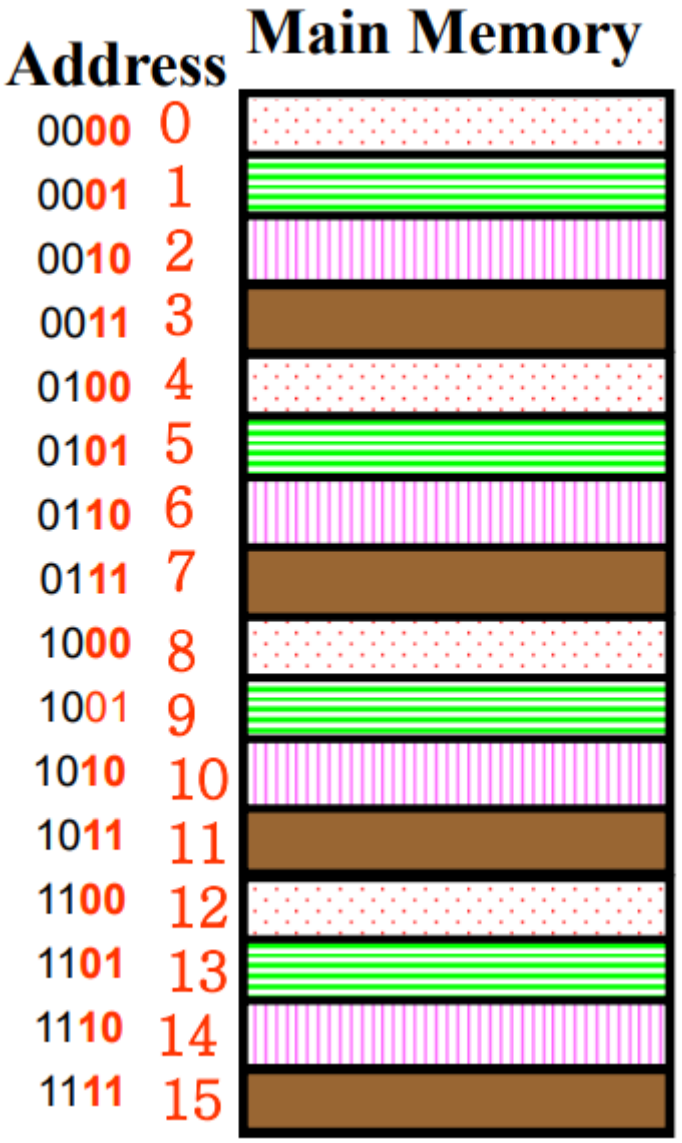


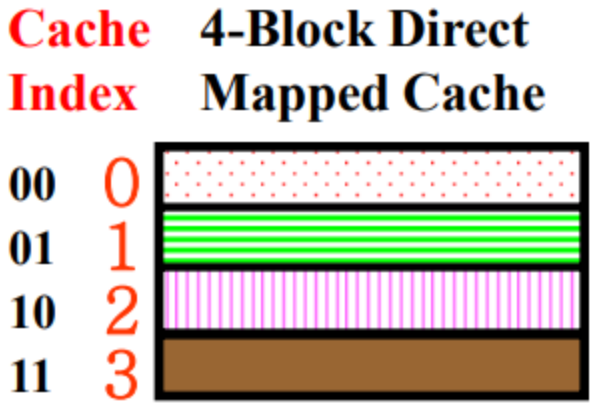
Main memory



Cache memory

Cache Memory: Direct mapping



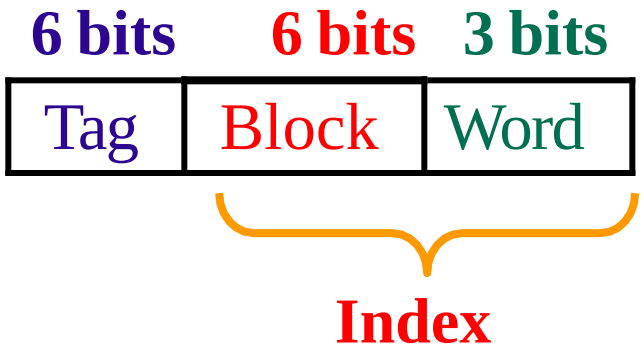


- Index determines block in cache
- If number of cache blocks is power of 2, then cache index is just the lower n bits of memory address

Cache Memory: Direct mapping (3/5)

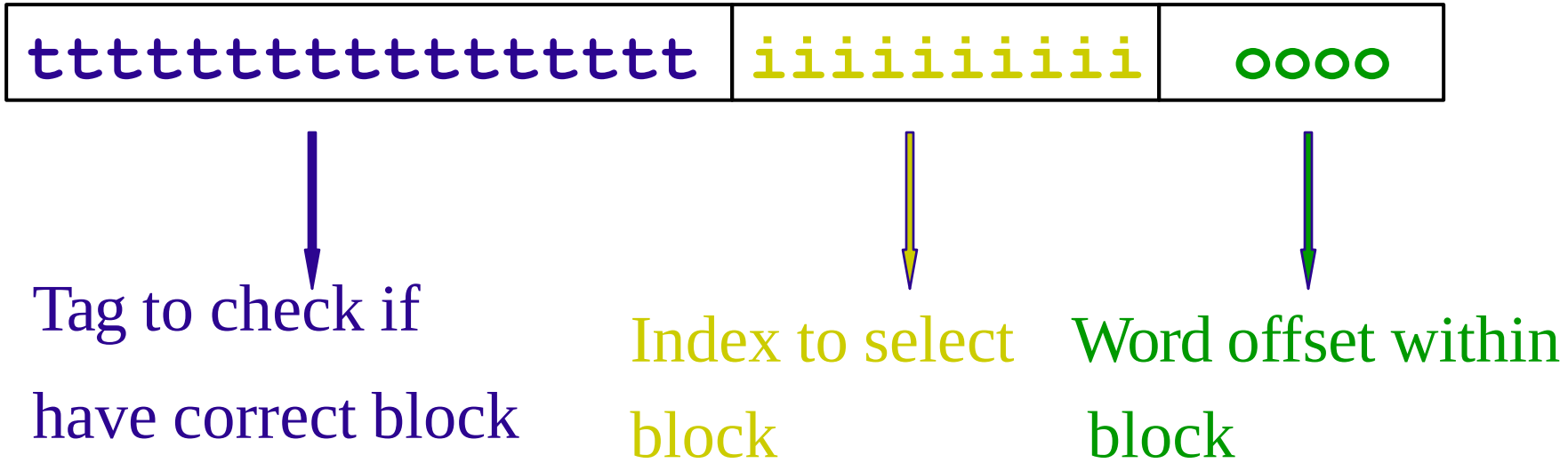
2. Direct mapping cache Organization: Block size of eight words

	Index	Tag	Data
Block 0	000	01	3450
	007	01	6578
Block 1	010		
	017		
Block 63	770	02	9989
	777	02	6710



Cache Memory: Direct mapping (3/5)

- If **block size > 1 word**, rightmost bits of index are really the **offset** of a word (possibly byte number) within the indexed block



Cache Memory: Direct mapping (4/5)

- The CPU address of **15** bits is divided into two fields.
- The **9** least significant bits constitutes the index field and the remaining **6** bits form the tag field.
- **The number of bits in the index field = the number of address bits required to access the cache memory.**
- In the general case , there are **2^*k* words in cache memory** and **2^*n* words in main memory** . The ***n*-bits** memory address is divided into two fields:
  - ***k* bits for the index to access the cache,**
  - ***n*-*k* bits for the tag field.**
- The direct mapping cache organization uses:
  - **The *n*-bit address to access the main memory,**
  - **And the k-bit index to access the cache.**

Cache Memory: Direct mapping (5/5)

- Each word in cache consists of the data word and its associated tag.
- When the CPU generates a memory request:
  - The index field is used for the address to access the cache.
  - The tag field of the CPU address is compared with the tag in the word read from the cache. If the two tags, **match** , there is a **hit** and the desired data word is in cache.
  - If there is **no match, there is a miss** and the required word is read from main memory.
  - It is then stored in the cache together with the new tag, replacing the previous value.

Disadvantage of direct mapping

- Two words with the same index in their address but with different tag values cannot reside in cache memory at the same time.

Direct Mapped Cache Example

- Direct mapped – each memory block can occupy one and only one cache block
- Example:
  - Cache block size: 16 words
  - Memory = 64K (4K blocks)
  - Cache = 2K (128 blocks)
- Memory block *n* occupies cache block (*n* mod 128)
- Consider address $2EF4

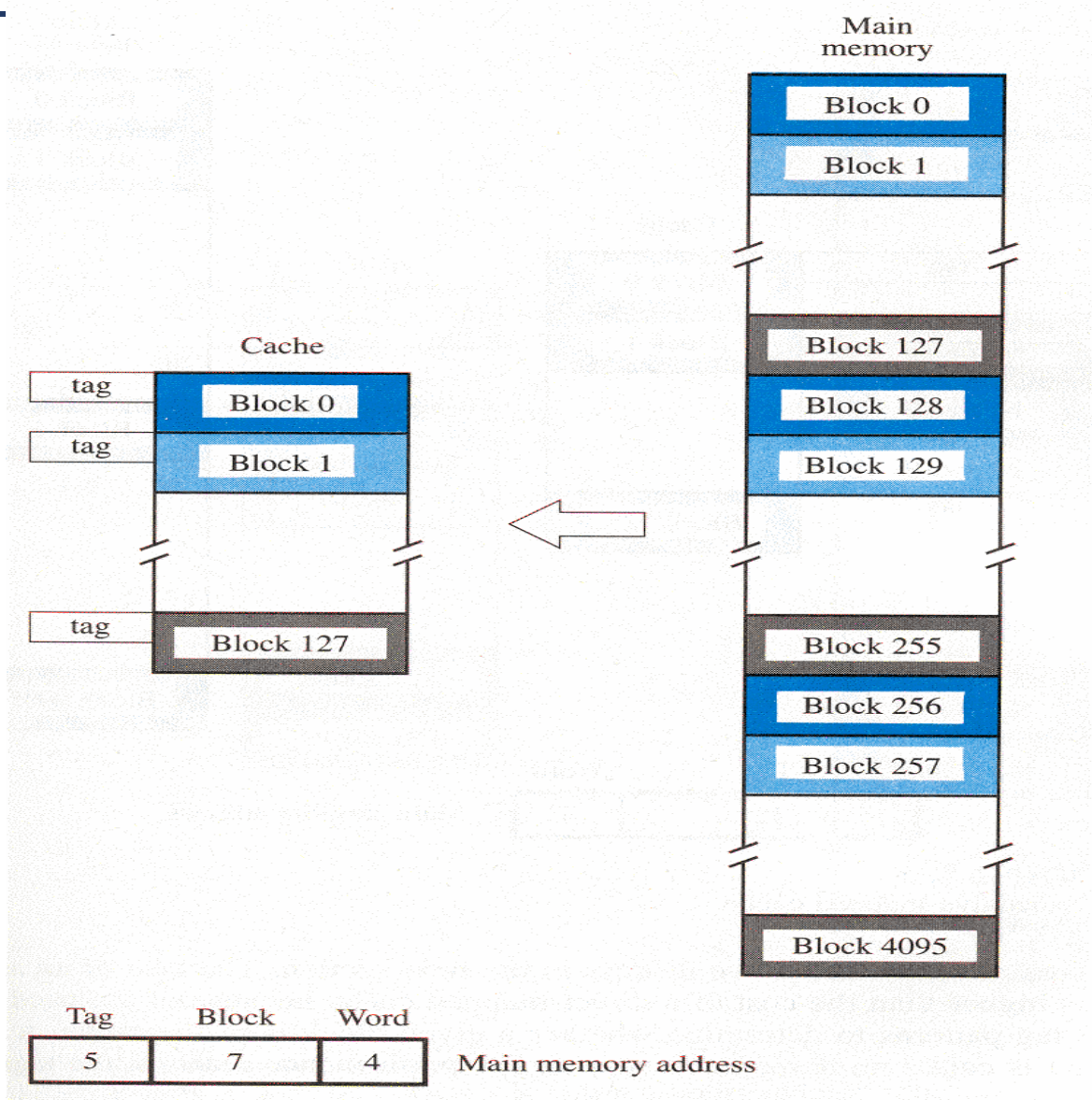
  `0010111101111 0100`

  block: $2EF = 751 word: 4
- Cache:

  `00101 1101111 0100`

  tag: 5 block: 111 word: 4

Direct Mapped Cache Example



Exercise

- A 4-way set associative cache memory unit with a capacity of 16 KB is built using a block size of 8 words. The word length is 32 bits. The size of the physical address space is 4 GB. The number of bits for the TAG field is _____.

Exercise

- A 4-way set associative cache memory unit with a capacity of 16 KB is built using a block size of 8 words. The word length is 32 bits. The size of the physical address space is 4 GB. The number of bits for the TAG field is _____.
- Given-
- Set size = 4 lines
- Cache memory size = 16 KB
- Block size = 8 words
- 1 word = 32 bits = 4 bytes
- Main memory size = 4 GB

- **<u>Number of Bits in Physical Address-</u>**
- We have,
- Main memory size
- = 4 GB
- = 2^{32} bytes
- Thus, Number of bits in physical address = 32 bits

- **<u>Number of Bits in Block Offset-</u>**
- We have,
- Block size
- = 8 words
- = 8 x 4 bytes
- = 32 bytes
- = 2^5 bytes
- Thus, Number of bits in block offset = 5 bits

- **<u>Number of Blocks in Cache-</u>**
- Number of lines in cache
- = Cache size / Block size
- = 16 KB / 32 bytes
- = 2^{14} bytes / 2^{5} bytes
- = 2^{9} blocks
- = 512 blocks
- Thus, Number of lines in cache = 512 blocks

- **Number of Sets in Cache-**
- Number of sets in cache
- = Number of blocks in cache / Set size
- = 512 lines / 4 lines
- = 2^9 lines / 2^2 lines
- = 2^7 sets
- Thus, Number of bits in set number = 7 bits

- **Number of Bits in Tag-**
- Number of bits in tag
- = Number of bits in physical address – (Number of bits in set number + Number of bits in block offset)
- = 32 bits – (7 bits + 5 bits)
- = 32 bits – 12 bits
- = 20 bits
- Thus, number of bits in tag = 20 bits

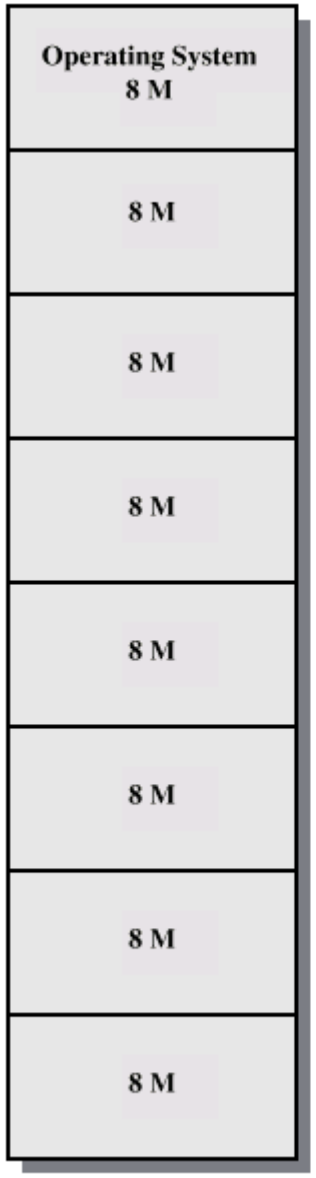
Memory Management

- Uni-program
  - Memory split into two
  - One for Operating System (monitor)
  - One for currently executing program
- Multi-program
  - “User” part is sub-divided and shared among active processes

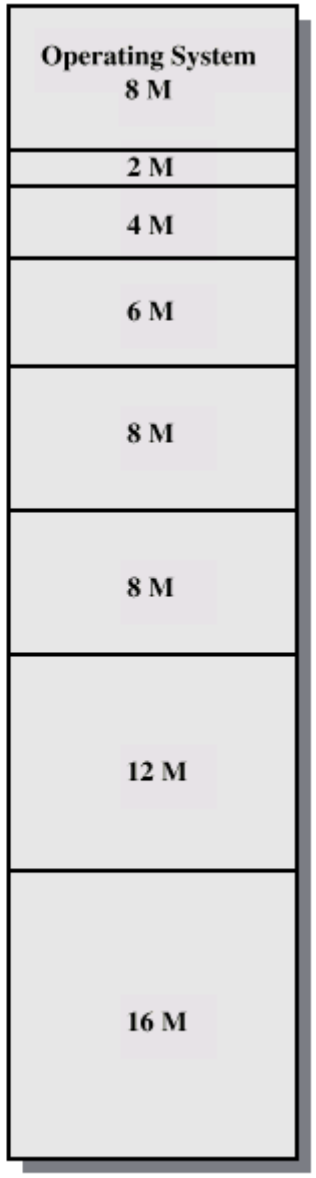
Partitioning

- Splitting memory into sections to allocate to processes (including Operating System)
- Fixed-sized partitions
  - May not be equal size
  - Process is fitted into smallest hole that will take it (best fit)
  - Some wasted memory
  - Leads to variable sized partitions

Fixed Partitioning



(a) Equal-size partitions



(b) Unequal-size partitions

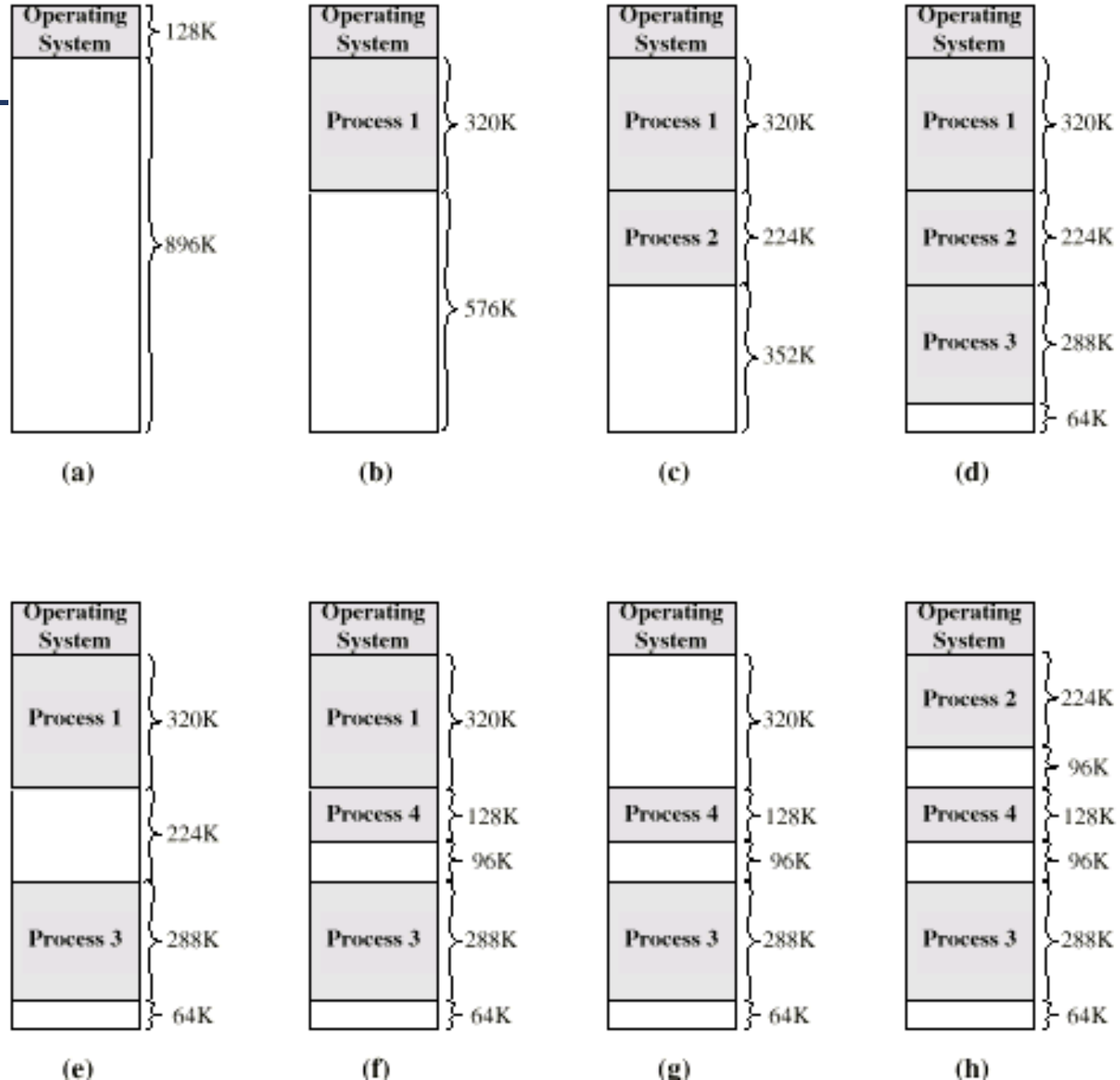
Variable Sized Partitions (1)

- Allocate exactly the required memory to a process
- This leads to a hole at the end of memory, too small to use
  - Only one small hole - less waste
- When all processes are blocked, swap out a process and bring in another
- New process may be smaller than swapped out process
- Another hole

Variable Sized Partitions (2)

- Eventually have lots of holes (fragmentation)
- Solutions:
  - Coalesce - Join adjacent holes into one large hole
  - Compaction - From time to time go through memory and move all hole into one free block (c.f. disk de-fragmentation)

Effect of Dynamic Partitioning



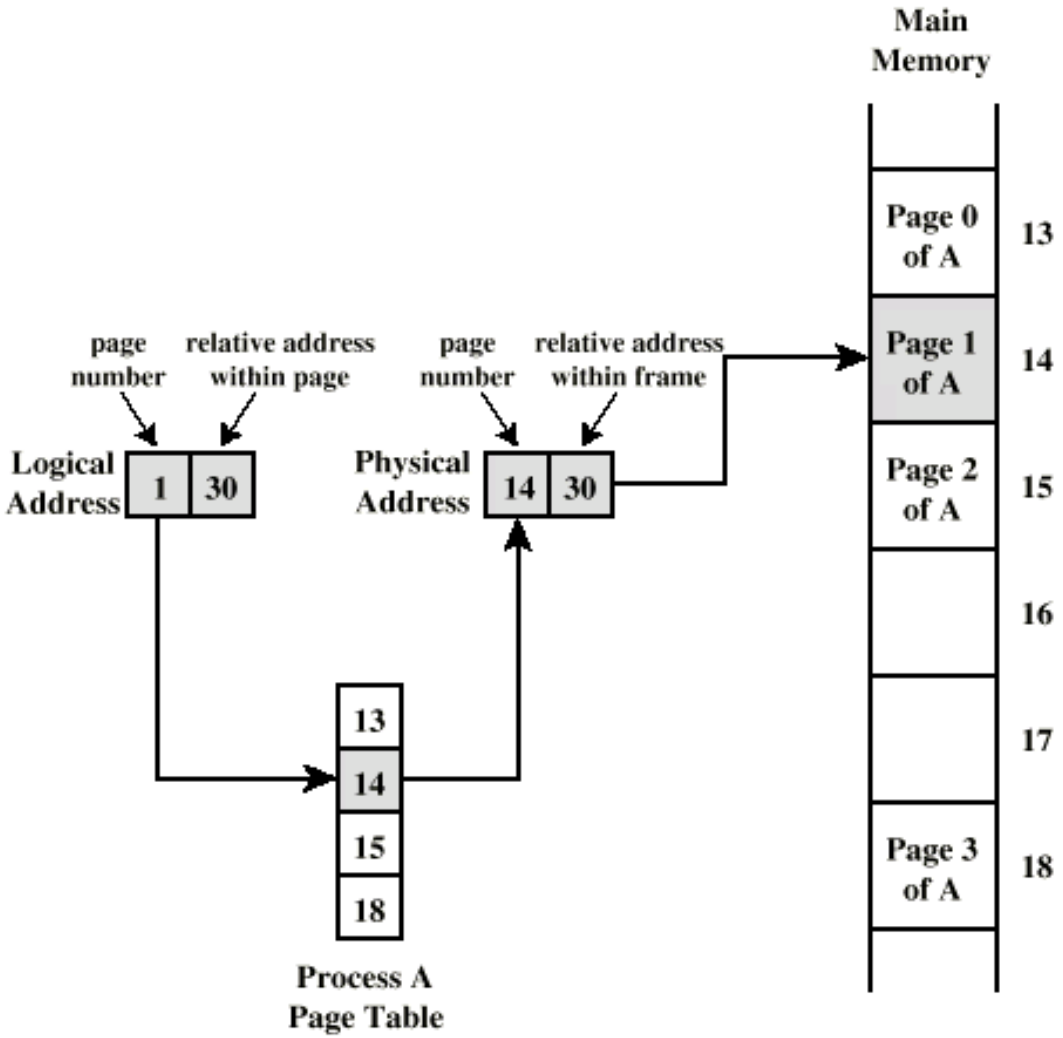
Relocation

- No guarantee that process will load into the same place in memory
- Instructions contain addresses
  - Locations of data
  - Addresses for instructions (branching)
- Logical address - relative to beginning of program
- Physical address - actual location in memory (this time)
- Automatic conversion using base address

Paging

- Split memory into equal sized, small chunks -page frames
- Split programs (processes) into equal sized small chunks - pages
- Allocate the required number page frames to a process
- Operating System maintains list of free frames
- A process does not require contiguous page frames
- Use page table to keep track

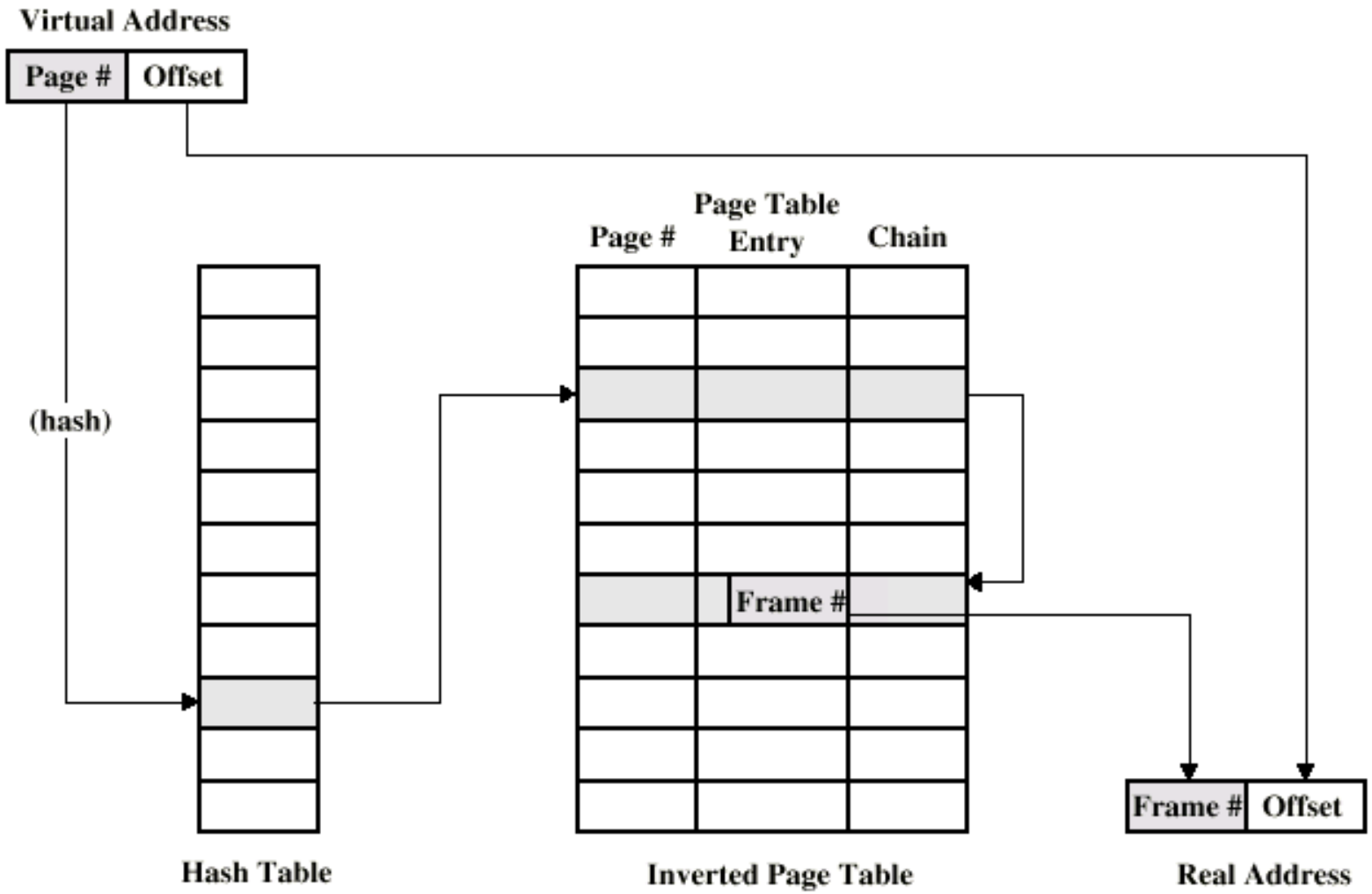
Logical and Physical Addresses - Paging



Virtual Memory

- Demand paging
  - Do not require all pages of a process in memory
  - Bring in pages as required
- Page fault
  - Required page is not in memory
  - Operating System must swap in required page
  - May need to swap out a page to make space
  - Select page to throw out based on recent history

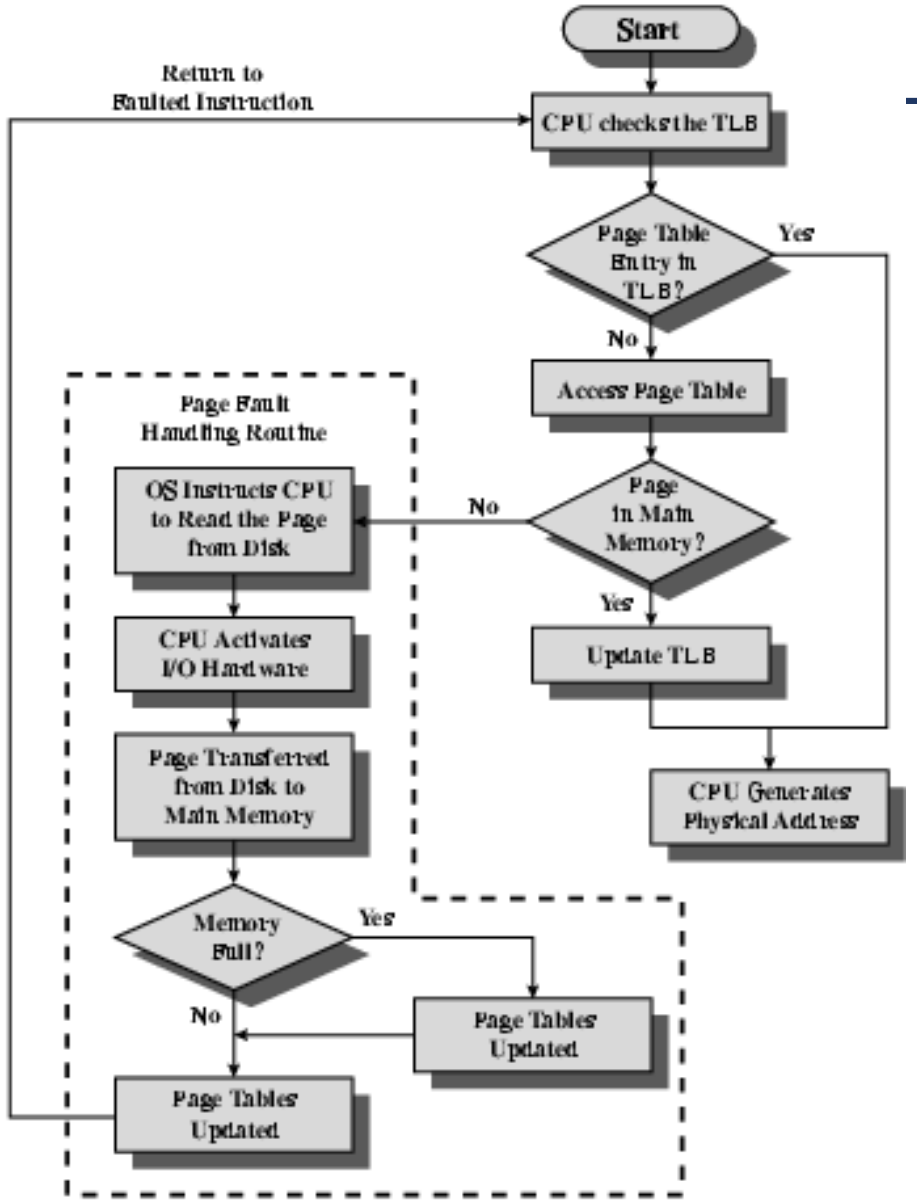
Inverted Page Table Structure



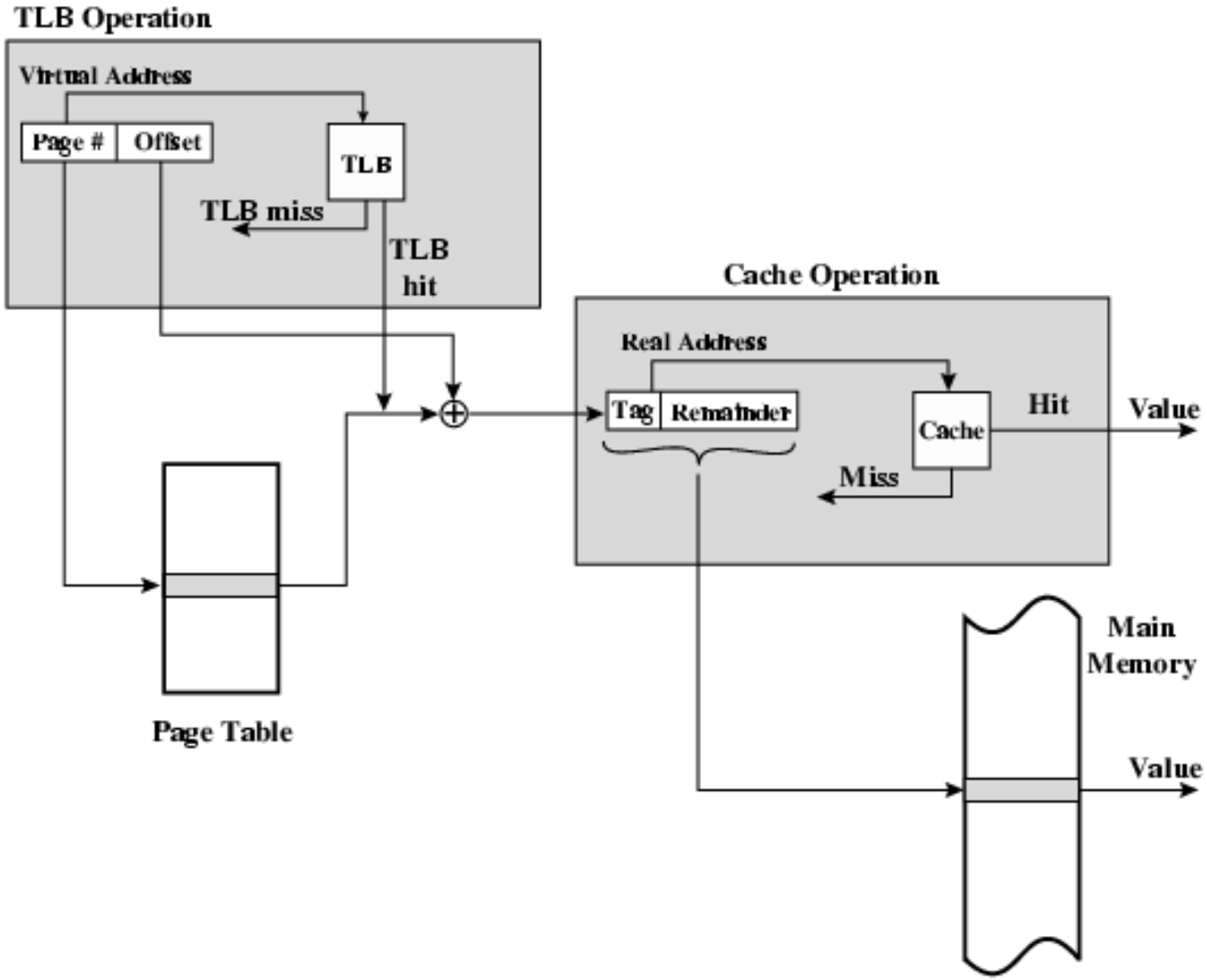
Translation Lookaside Buffer

- Every virtual memory reference causes two physical memory access
  - Fetch page table entry
  - Fetch data
- Use special cache for page table
  - TLB

TLB Operation



TLB and Cache Operation



Segmentation

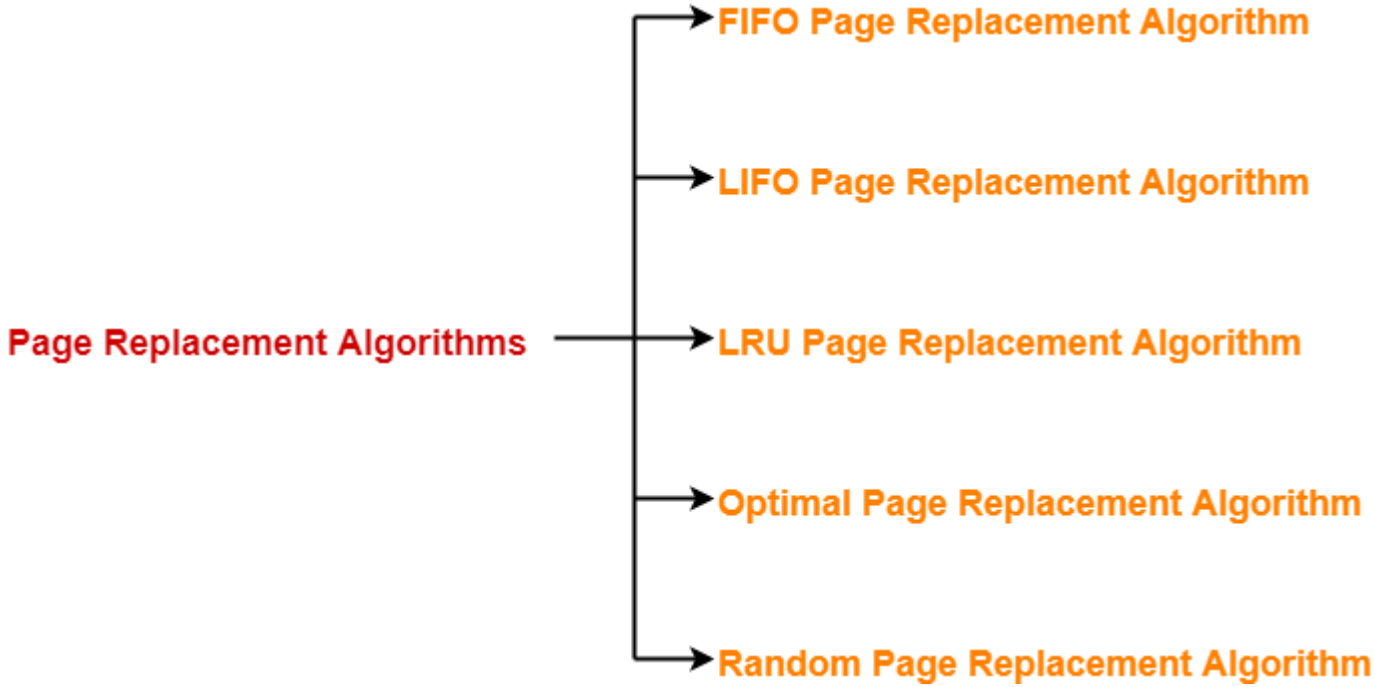
- Paging is not (usually) visible to the programmer
- Segmentation is visible to the programmer
- Usually different segments allocated to program and data
- May be a number of program and data segments

Advantages of Segmentation

- Simplifies handling of growing data structures
- Allows programs to be altered and recompiled independently, without re-linking and re-loading
- Lends itself to sharing among processes
- Lends itself to protection
- Some systems combine segmentation with paging

Page Replacement Algorithms

Page replacement algorithms help to decide which page must be swapped out from the main memory to create a room for the incoming page.



Page Replacement Algorithms

- **FIFO Page Replacement Algorithm-**
  - As the name suggests, this algorithm works on the principle of "**First in First out**".
  - It replaces the oldest page that has been present in the main memory for the longest time.
  - It is implemented by keeping track of all the pages in a queue.
- **LIFO Page Replacement Algorithm-**
  - As the name suggests, this algorithm works on the principle of "**Last in First out**".
  - It replaces the newest page that arrived at last in the main memory.
  - It is implemented by keeping track of all the pages in a stack.

Page Replacement Algorithms

- **LRU Page Replacement Algorithm-**
  - As the name suggests, this algorithm works on the principle of "**Least Recently Used**".
  - It replaces the page that has not been referred by the CPU for the longest time.
- **Optimal Page Replacement Algorithm-**
  - This algorithm replaces the page that will not be referred by the CPU in future for the longest time.
  - It is practically impossible to implement this algorithm.
  - This is because the pages that will not be used in future for the longest time can not be predicted.
  - However, it is the best known algorithm and gives the least number of page faults.
  - Hence, it is used as a performance measure criterion for other algorithms.

Page Replacement Algorithms

- **Random Page Replacement Algorithm-**
  - As the name suggests, this algorithm randomly replaces any page.
  - So, this algorithm may behave like any other algorithm like FIFO, LIFO, LRU, Optimal etc.

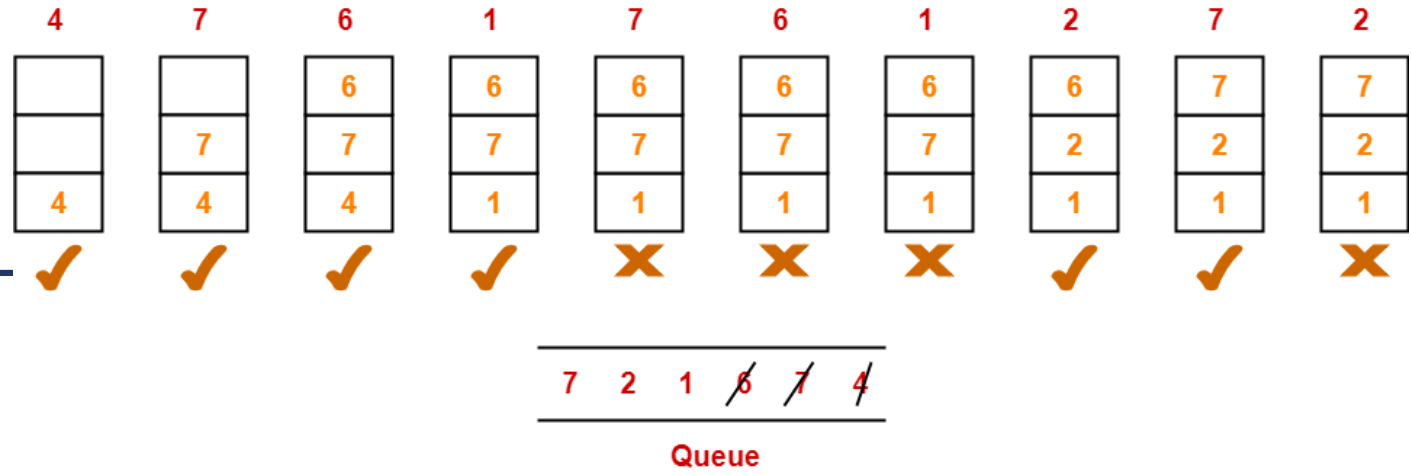
Page Replacement Algorithms Based Exercise

- A system uses 3 page frames for storing process pages in main memory. It uses the First in First out (FIFO) page replacement policy. Assume that all the page frames are initially empty. What is the total number of page faults that will occur while processing the page reference string given below-

  4 , 7, 6, 1, 7, 6, 1, 2, 7, 2

- Also calculate the hit ratio and miss ratio.

Solution



- Total number of page faults occurred = 6

Calculating Hit ratio-

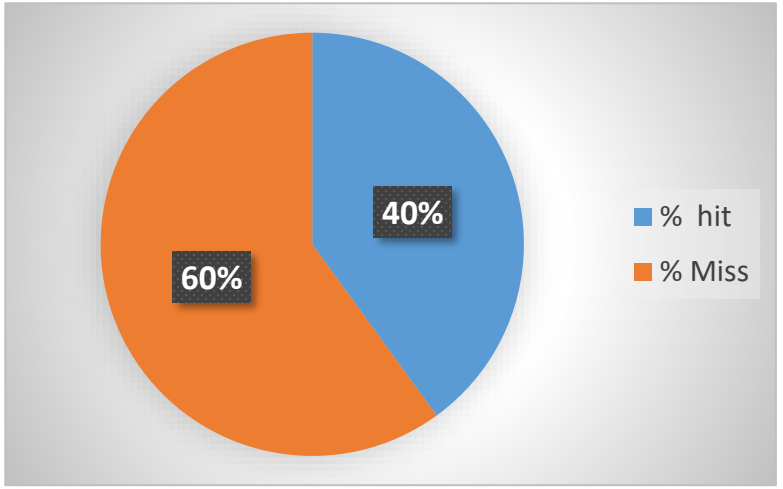
- Total number of page hits
- = Total number of references – Total number of page misses or page faults
- = 10 – 6 = 4

Hit ratio

- = Total number of page hits / Total number of references
- = 4 / 10 = 0.4 or 40%

Miss ratio

- = 1 – Hit ratio
- = 1 – 0.4 = 0.6 or 60%



Virtual Memory Concept Practice Exercise-1

- Calculate the size of memory if its address consists of 24 bits and the memory is 4-byte addressable.

Solution-

- We have-
- Number of locations possible with 24 bits = 2^{24} locations
- It is given that the size of one location = 4 bytes
- Thus, Size of memory
- = 2^{24} x 4 bytes
- = 2^{24} x 2^{2} bytes
- =2^{26} bytes
- = 64 MB

Exercise-2

- A 32 bit address system, uses a paged virtual memory; the page size is 2 KBytes.
- What is the virtual page and the offset in the page for the virtual address 0x00030f40(in hexadecimal)?

Solution

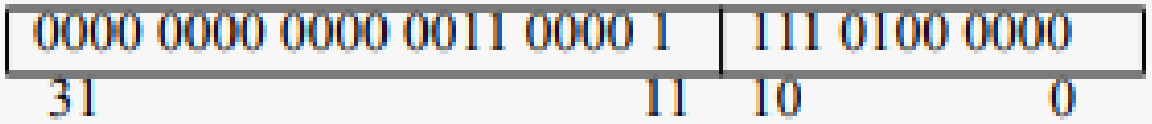
For a page size of N Bytes the number of bits in the offset field is $\log_2 N$. In the case of a 2 KBytes page there are:

$$\log_2 2^{11} = 11 \text{ bits}$$

Therefore the number of bits for the page number is:

$$32 - 11 = 21$$

which means a total of $2^{21} = 2$ Mpages. The binary representation of the address is:



virtual address 0x00030f40?

The given virtual address identifies

The virtual page number 0x61 = 97_{10};

The offset inside the page is 0x740 = 1856_{10}.

Exercise-3

- A 32 bit address system, has a 4 MBytes main memory.
- The page size is 1 KByte.
- What is the size of the page table?

Solution

The page table is addressed with the page number field in the virtual address. Therefore the number of lines in the page table equals the number of virtual pages. The number of virtual pages is:

$$\text{virtual\_pages} = \frac{\text{address\_space [Bytes]}}{\text{page\_size [Bytes]}}$$

$$\text{virtual\_pages} = \frac{2^{32}}{2^{10}} = 2^{22} = 4 \text{ Mpages}$$

The width of each line in the page table equals the width of the page number field in the virtual address. If the number of virtual pages is 2^{22}, then the line width is 22 bits. It results that the size of the page table is:

page_table_size = number_of_entries * line_size

page_table_size = 2^{22}*22 bits = 11.5 Mbyte!!

Exercise-4

- In a virtual memory system, size of virtual address is 32-bit, size of physical address is 30-bit, page size is 4 Kbyte and size of each page table entry is 32-bit.
- The main memory is byte addressable.
- Which one of the following is the maximum number of bits that can be used for storing protection and other information in each page table entry?

Solution

- Given-
- Number of bits in virtual address = 32 bits
- Number of bits in physical address = 30 bits
- Page size = 4 KB
- Page table entry size = 32 bits

Solution

- **Size of Main Memory-**
- Number of bits in physical address = 30 bits
- Thus, Size of main memory
- = 2^{30} B = 1 GB
- **Number of Frames in Main Memory-**
- Number of frames in main memory = Size of main memory / Frame size
- = 1 GB / 4 KB = 2^{30} B / 2^{12} B = 2^{18}
- Thus, Number of bits in frame number = 18 bits

- **Number of Bits used for Storing other Information-**
- Maximum number of bits that can be used for storing protection and other information
- = Page table entry size – Number of bits in frame number
- = 32 bits – 18 bits = 14 bits

Question

If an instruction takes i microseconds and a page fault takes an additional j microseconds, the effective instruction time if on the average a page fault occurs every k instruction is _____.

Solution

Given-

Page fault service time = j µsec
Average memory access time = i µsec
One page fault occurs every k instruction

Page Fault Rate-

- It is given that one page fault occurs every k instruction.
- Thus, Page fault rate = 1 / k

Effective Access Time With Page Fault-

- It is given that effective memory access time without page fault = i µsec
- So, Effective access time with page fault
- = (1 / k) x { i µsec + j µsec } + (1 – 1 / k) x { i µsec }
- = j / k µsec + i µsec
- = i + j / k µsec

Question

Effective Access Time With Page Fault-

- It is given that effective memory access time without page fault = i µsec
- So, Effective access time with page fault
- = (1 / k) x { i µsec + j µsec } + (1 – 1 / k) x { i µsec }
- = j / k µsec + i µsec
- = i + j / k µsec

References

Text Book

- M. M. Mano, Computer System Architecture, Prentice-Hall,2004