Unit 5: Memory Subsystem

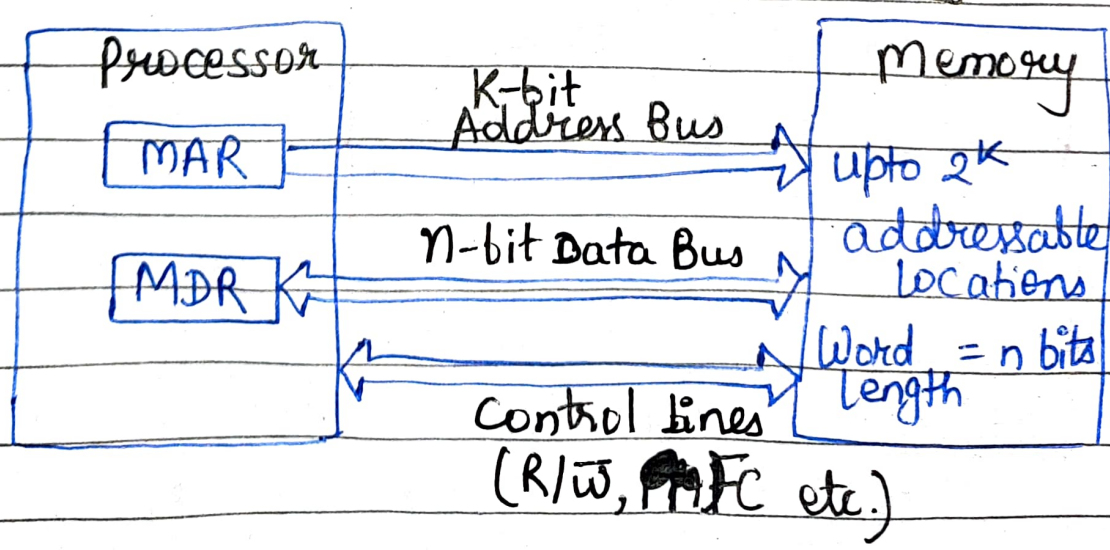
Introduction

Memory → important functional unit of a computer
- → used to store both instructions and data
- → Stores as bits (0's and 1's) usually organized in terms of bytes.

How are data stored in memory accessed?

→ Every memory location has a unique address
- → byte addressable if every byte of data has a unique address
- → word addressable if every addressed locations consits of multiple bytes, like 32 bits or 4 bytes

Connection between Processor and Memory



- Data transfer b/w memory and processor takes place through use of two processor registers
  - MAR (memory address register)
  - MDR (memory data register)

- If MAR is k bits long, MDR is n bits long → memory unit may contain upto 2^k addressable locations each storing n bits data

- During a memory cycle, n bits of data transferred b/w memory and processor over processor bus having k address lines and n data lines

- Bus also includes control lines Read / $\overline{\text{Write}}$ (R/$\overline{W}$) and other control signals like MFC (Memory function completed)

- <u>Memory Read</u>: load address of required memory location into MAR and set R/$\overline{W}$ line to 1
  ↓
  memory place data from addressed location onto data lines and set MFC=1
  ↓
  upon receipt of MFC signal, processor loads data onto data lines into MDR

- **Memory Write:** load address of location into MAR and load data into MDR
  ↓
  set R/$\overline{W}$ line to 0

If read or write operations involve consecutive address locations in main memory, then a block transfer operation can be performed in which only address sent to memory is one that identifies first location.

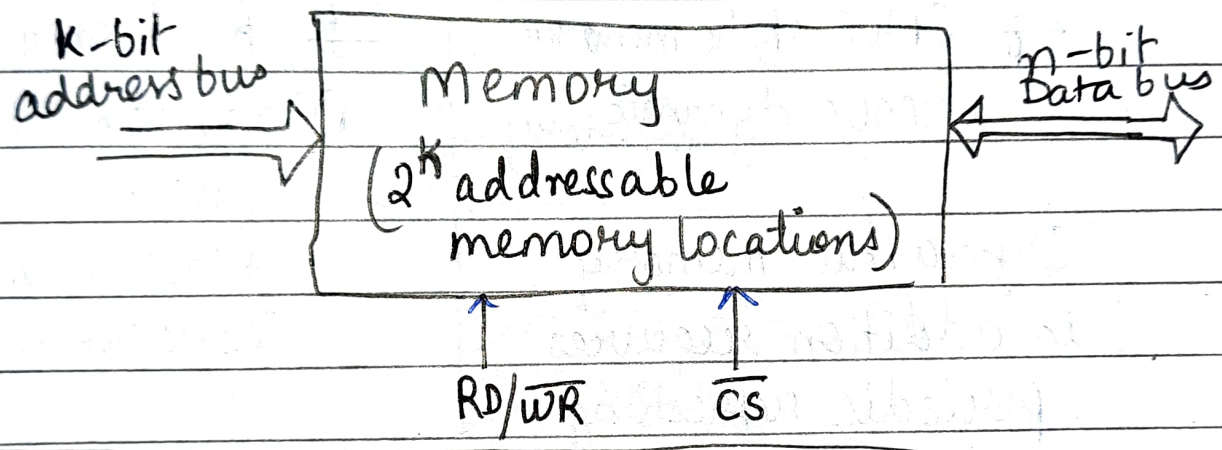
- Memory accesses may be synchronized using a clock, or they may be controlled using special signals that control transfers on bus, using bus signaling schemes.

- Memory read and write operations are controlled as input and output bus transfers respectively.

Time that elapses b/w initiation of an operation and completion of that operation (e.g time b/w Read and MFC signals) is referred to as memory access time.

- Minimum time delay required b/w initiation of two successive memory operations, (e.g. time b/w two successive Read operations) is referred to as memory cycle time.

Example memory module



$\boxed{\text{memory size is specified as } 2^K \times n}$

→ K-bit address lines
∴ max. 2^K memory locations can be accessed

→ n data lines
∴ no. of bits stored in each location is n bits

→ RD/$\overline{WR}$ = 1 (Read)
= 0 (write)

→ Chip select line ($\overline{CS}$) = 0 enable chip
↓
active low

otherwise data bus is in high impedance state.

Classification of Memory Systems

(a) Volatile vs Non volatile

Volatile	Non-volatile
stored data is lost when power is switched off	Stored data is retained even when power is switched off
e.g. CMOS static memory CMOS dynamic memory	e.g. Read only memory, Magnetic disk, CDROM/DVD, flash memory, resistive memory
Dynamic memory in addition requires periodic refreshing	

(b). Random-access vs. Direct/Sequential Access

Random-access	Sequential access	Direct or semi-random access
read/write time is independent of memory location being accessed	Stored data can only be accessed sequentially in particular order	part of access is sequential and part is random
e.g. CMOS memory (RAM and ROM)	e.g. Magnetic tape	e.g. Magnetic disk • can directly go to a track after which access will be s

(c). Read-only and Read-Write

Read-only (ROM)	Read-Write (RWM/RAM)
• Data can be read only • Data once stored in permanent or semi permanent • Data written (programmed) during manufacture or in laboratory e.g. ROM, PROM, EPROM, EEPROM	• Data can be read as well as written • Used in main/cache memory systems e.g. Static RAM (SRAM) ↳ data once written retained as long as power is on Dynamic RAM (DRAM) ↳ requires periodic refreshing even when power is on (as data stored as charge on capacitors

Design Issue of Memory System

→ Most important issue is to bridge processor-memory gap that has been widening with every passing year.

→ Advancements in memory technology are unable to cope with faster advancements in processor technology.

Important questions ?

→ How to make memory system work faster?

→ How to increase data transfer rate between CPU and memory?

→ How to address ever increasing storage needs of applications?

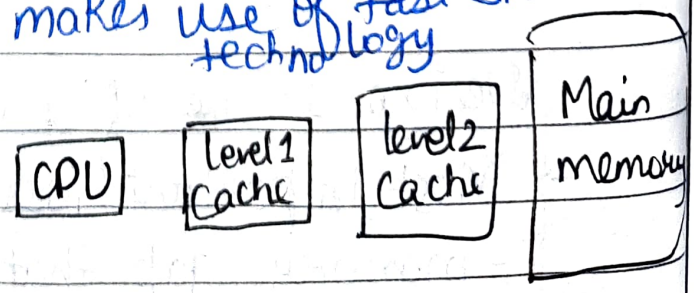
Possible Solutions

- Cache Memory: to increase effective speed of memory system

- Virtual Memory: to increase effective size of memory system

Cache Memory ?? → makes use of fast SRAM technology

→ Small fast memory inserted b/w larger slower main memory and processor

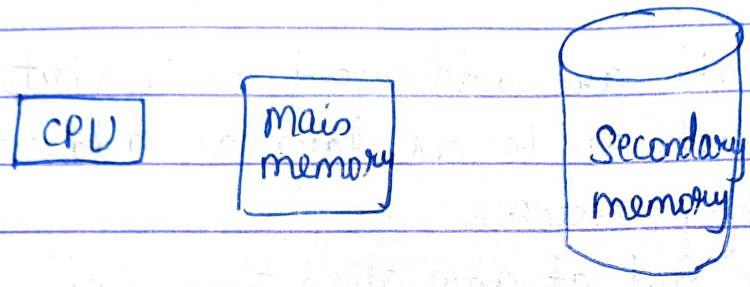


→ holds currently active segments of program & their data

Unit 5: Memory Subsystem

Virtual Memory

→ Technique used by operating system to provide an illusion of very large memory to processor



→ Important concept related to memory organization

→ Data may be stored in physical memory locations that have addresses different from those specified by program.

→ Memory control circuit translates address specified by program into an address that can be used to access physical memory.

Address generated by processor ⟶ virtual or logical address

→ Virtual address space is mapped onto physical memory where data are actually stored. Mapping function is implemented by a special memory control circuit, called memory management unit

Memory function can be changed during program execution according to system requirements.

→ Virtual memory used to increase apparent size of physical memory.

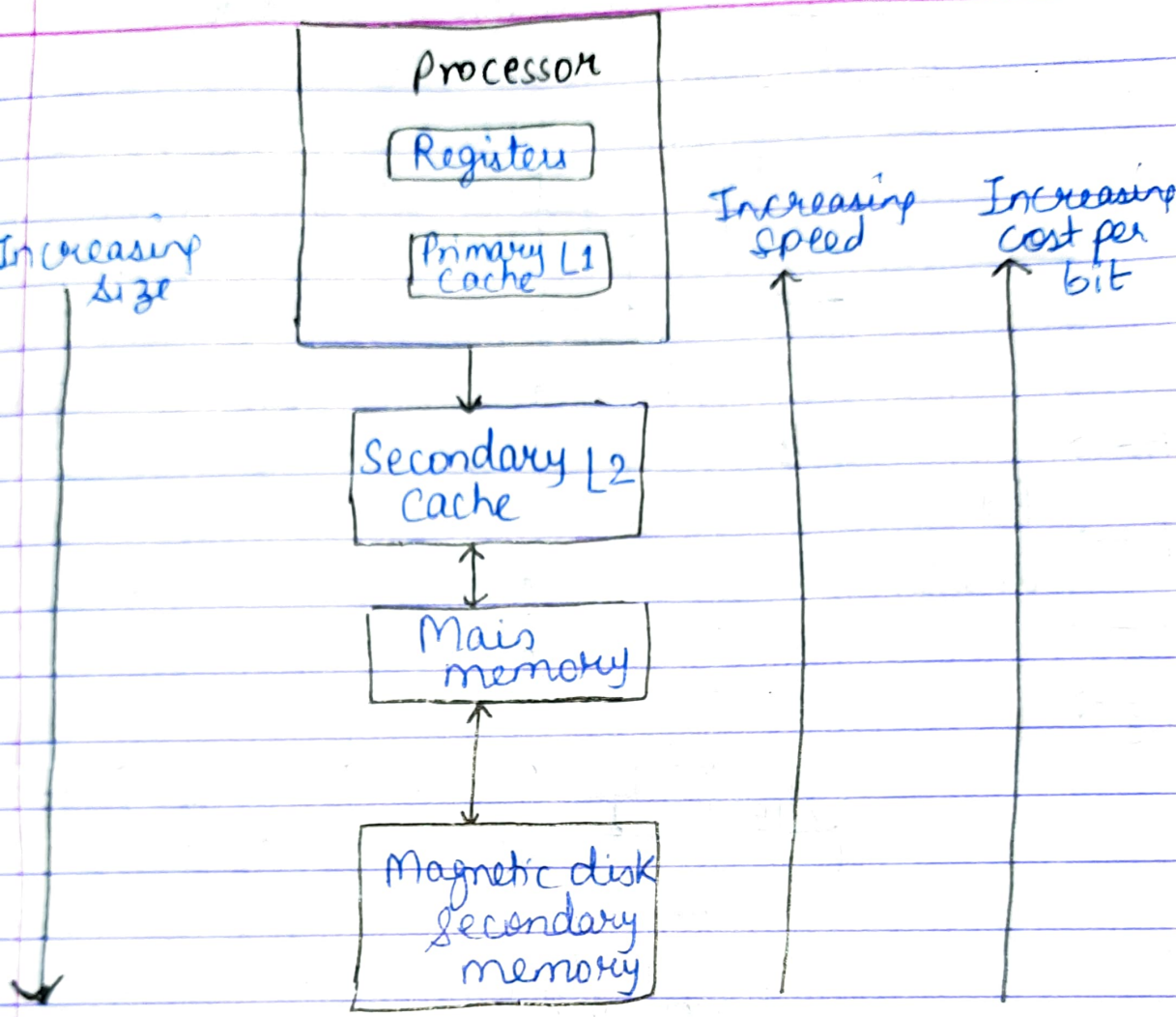
→ Data are addressed in a virtual address space that can be as large as addressing capability of processor

- But at any given time, only active portion of this space is mapped onto locations in physical memory.
- Remaining virtual addresses are mapped onto bulk storage devices used (secondary memory)

→ As active portion of virtual address space changes during program execution, memory management unit changes mapping function and transfers data b/w secondary memory and primary memory.

* <u>Memory Hierarchy</u>

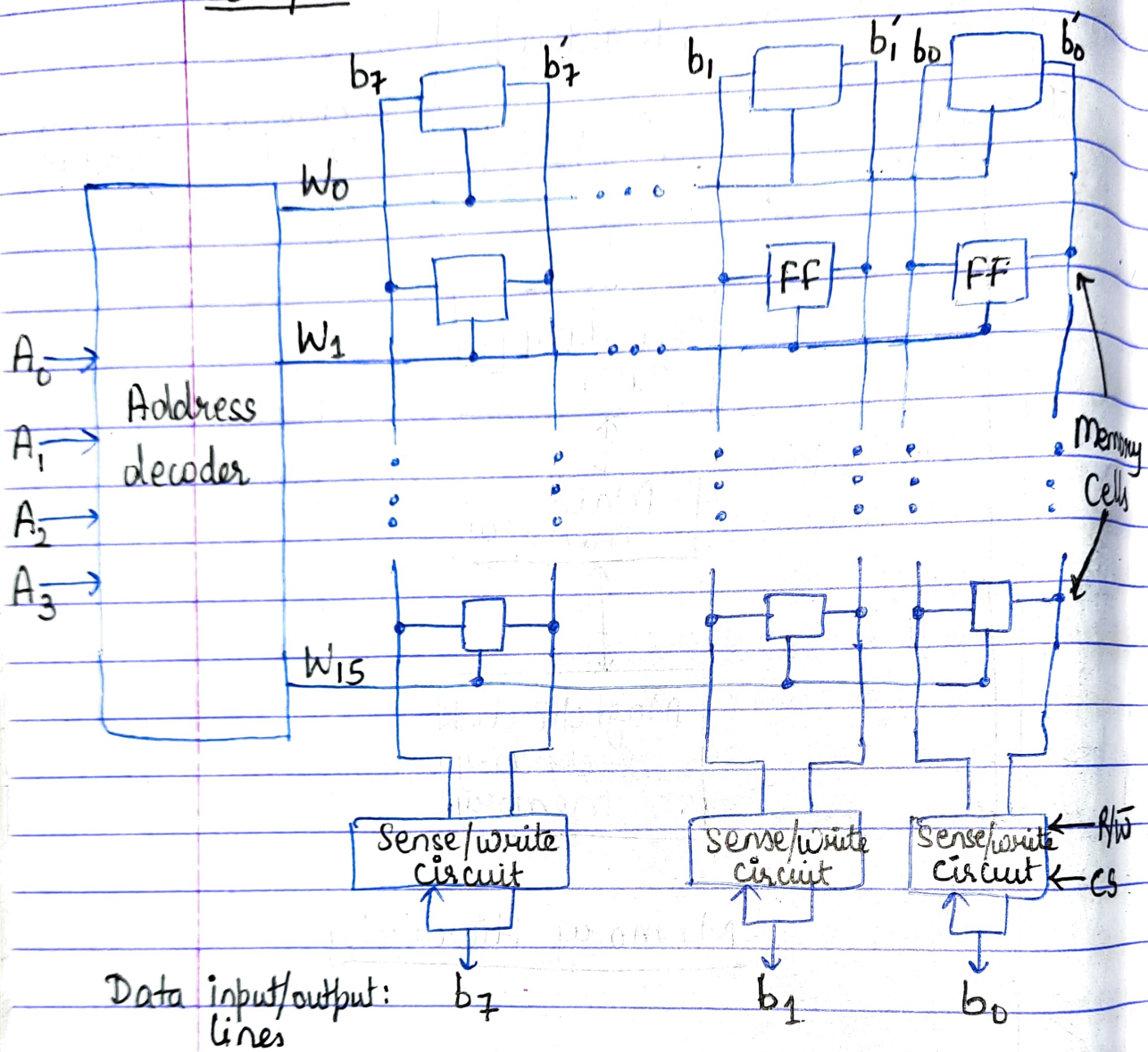
The entire computer memory can be viewed as memory hierarchy.



Memory Hierarchy

Semiconductor Memories
- → available in a wide range of speeds
- → uses VLSI (Very Large Scale Integration) technology
  - → cost ↓
- → no. of memory cells organized inside a chip

Internal Organization of Memory Chips



Organization of bit cells in a memory chip

(16x8 organization)

16 words of 8 bits each

- Memory cells organized in form of an array, each cell is capable of storing one bit of information.

- Each row of cell constitutes a memory word, and all cells of a row are connected to common word line driven by address decoder on chip

★ need of two lines for power supply & ground connections too.

→ Cells in each column are connected to a Sense/write circuit by two bit lines.

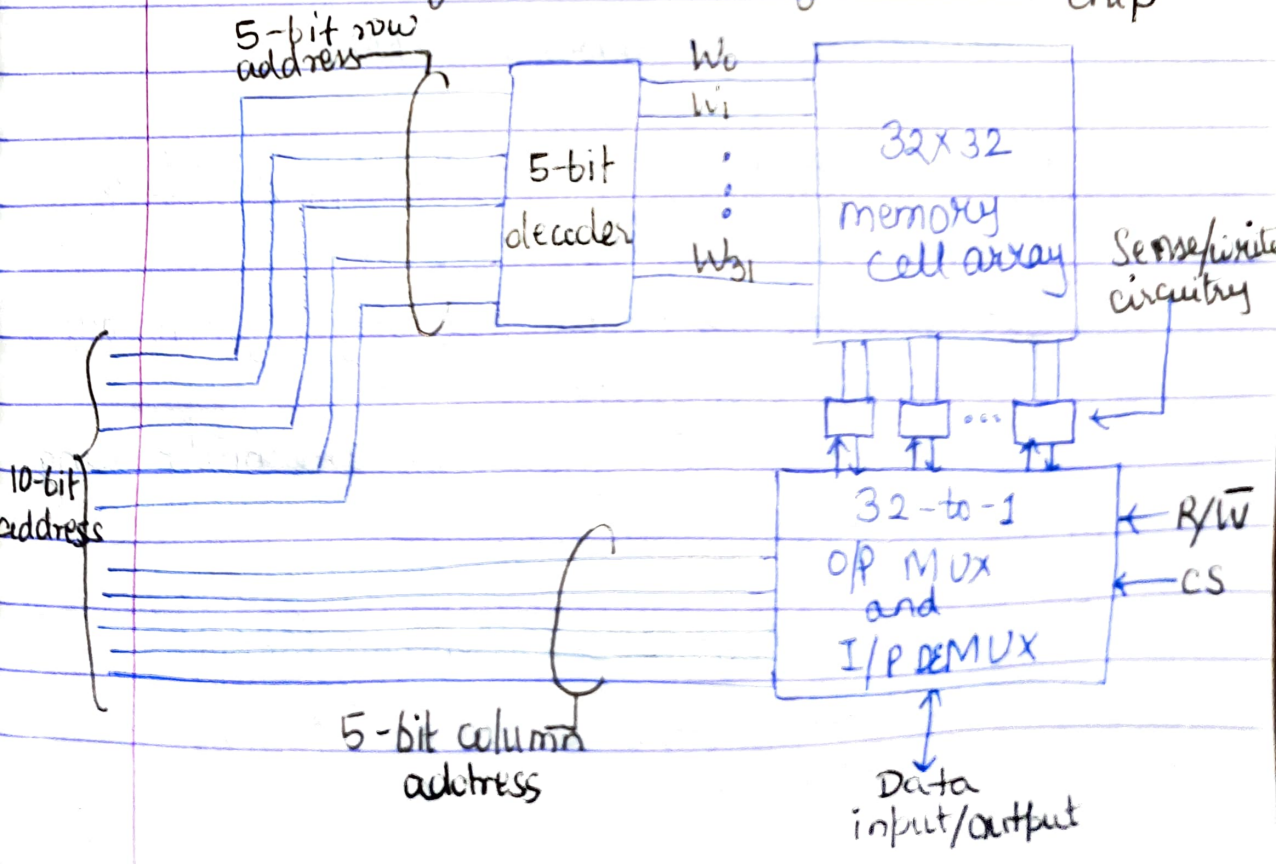
→ Sense/write circuits are connected to data input/output lines of chip.

→ During a Read operation, these circuits sense or read information stored in cells selected by a word line and transmit this information to output data lines.

→ During a Write operation, Sense/write circuits receive i/p information & store it in cells of selected word.

CS → used to select one single chip in a multichip memory system

Memory Organization of larger 1K x 1 memory chip



- Required 10-bit address is divided into two groups of 5 bits each to form row and column addresses for cell array.

- Row address decoder selects a row of 32 cells, all of which are accessed in parallel.

- However, according to column address, only one of these cells is connected to external data line by o/p MUX and i/p DEMUX

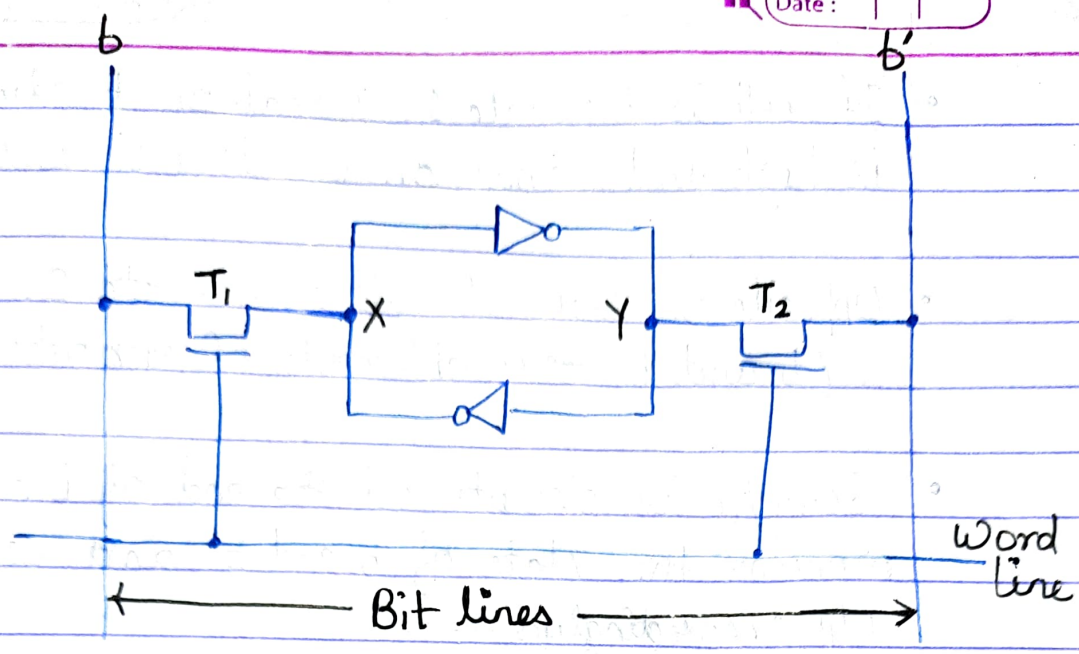
Random Access Memory (RAM)

- (a) Static RAM (SRAM)
- (b). Dynamic RAM (DRAM)
  - Asynchronous DRAM
  - Synchronous DRAM

- Vary in terms of speed, density, volatility properties, and cost.
  - → Present day main memory systems are built using DRAM
  - → Cache memory systems are built using SRAM

(a). Static Random Access Memory (SRAM)

↳ retain state as long as power is applied.



Static RAM cell

- Two cross connected inverter form a latch.
- Latch is connected to two bit lines by transistors T_1 and T_2 (acting as switches opened or closed under control of word line).

- When word line = 0, transistors off and latch retains its state

e.g. Let us assume cell is in state 1 if logic value at pt. X is 1 and at pt. Y is 0. This state is maintained as long as signal on word line is at ground level.

Read operation

To read state of SRAM cell, Word line (WL) is activated to close switches T_1 and T_2.

- If cell is in State 1, signal on bit line b is high and signal on bit line b' is low.

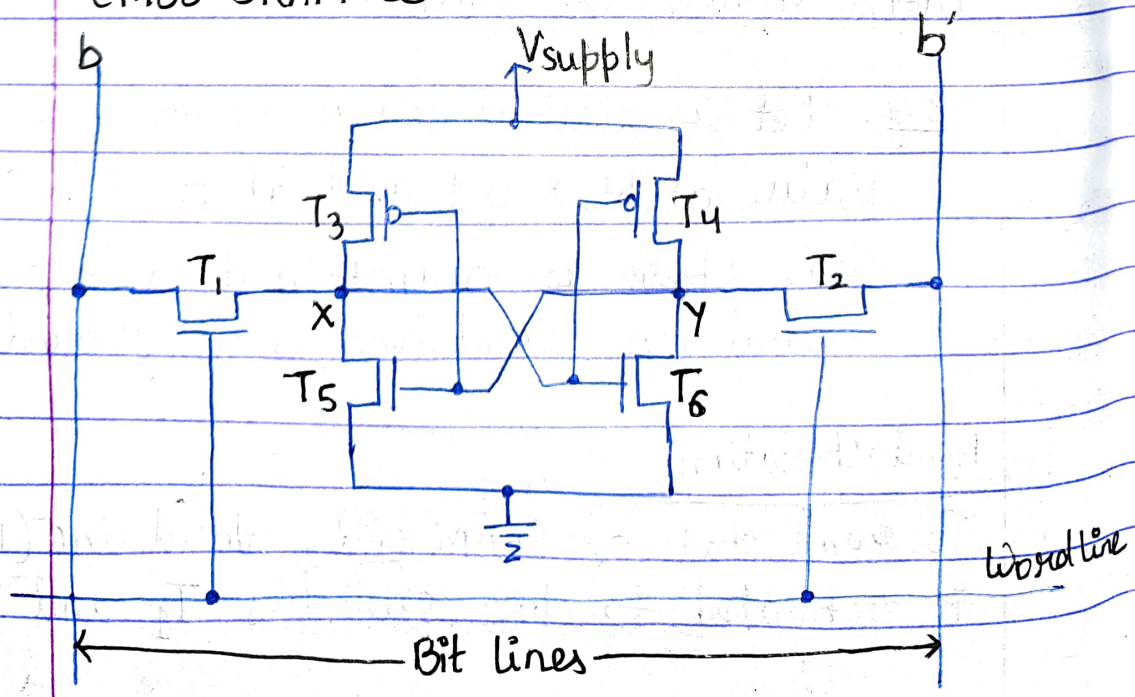
- Opposite is true if cell is in state 0. (b and b' → complements of each other)

- Sense/write circuits at the end of bit lines monitor the state of b and b' and set o/p accordingly.

Write Operation

- State of cell is set by placing appropriate value on bit line b and its complement on b' and then activating word line.
- Required signals on bit lines are generated by Sense/write circuit.

CMOS SRAM cell



Page No.
Date : | |

- Transistor pairs (T_3, T_5) and (T_4, T_6) form inverters in latch.

e.g. In state 1, voltage at pt X is maintained high by having transistors T_3 and T_6 on, while T_4 and T_5 are off

∴ if T_1 and T_2 are turned on (closed), bit lines b and b' will have high and low signals, respectively.

features → SRAMs are said to be volatile memories because their contents are lost when power is interrupte

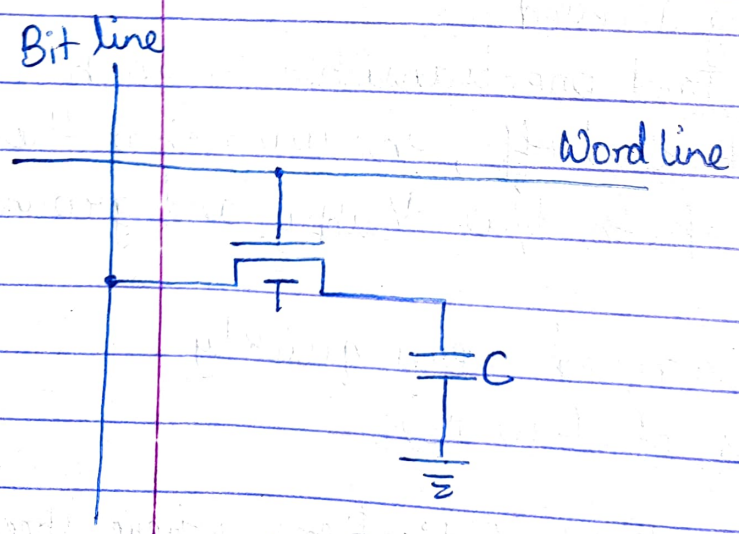
adv. * CMOS SRAMs have very low power consumption because current flows in cell only when cell is being accessed.
- otherwise, T_1, T_2 and one transistor in each inverter are turned off, ensuring that there is no active path b/w V_{supply} and ground.

* SRAMs can be accessed very quickly (access times of few ns)

SRAMs are used in applications where speed is of critical concern.

(b) Dynamic RAM (DRAM)

- SRAMs are fast but they have high cost as their cells require several transistors.
- Less expensive RAMs can be implemented if simpler cells are used.
- But such cells do not retain their state indefinitely; hence called dynamic RAMs (DRAMs).
- Information is stored in form of a charge on a capacitor.
- Since cell is required to store information for a much longer time, it contents must be periodically refreshed by restoring the capacitor charge to its full value.



Single transistor dynamic memory cell

- to store informat in cell, T is turn on and appropri voltage is applied to bit line.
- This causes a kno amount of charge to be stored in capacitor.

Page No.
Date : | |

- After transistor is turned off, capacitor begins to discharge.

- Hence information stored in cell can be retrieved correctly only if it is read before charge on capacitor drops below some threshold value.

During a Read operation, T ⇒ ON
Sense amplifier connected to bit line detects whether charge stored on capacitor is above threshold value.

If so, it drives the bit line to a full voltage that represents logic value 1, recharging capacitor to full charge corresponding to logic value 1.

If sense amplifier detects that charge on capacit is below threshold value, it pulls the bit line to ground level ensuring capacitor will have no charge, representing logic value 0.

→ Thus, reading contents of cell automatically refreshes its contents.

→ All cells in a selected row are read at the same time, which refreshes contents of entire row.

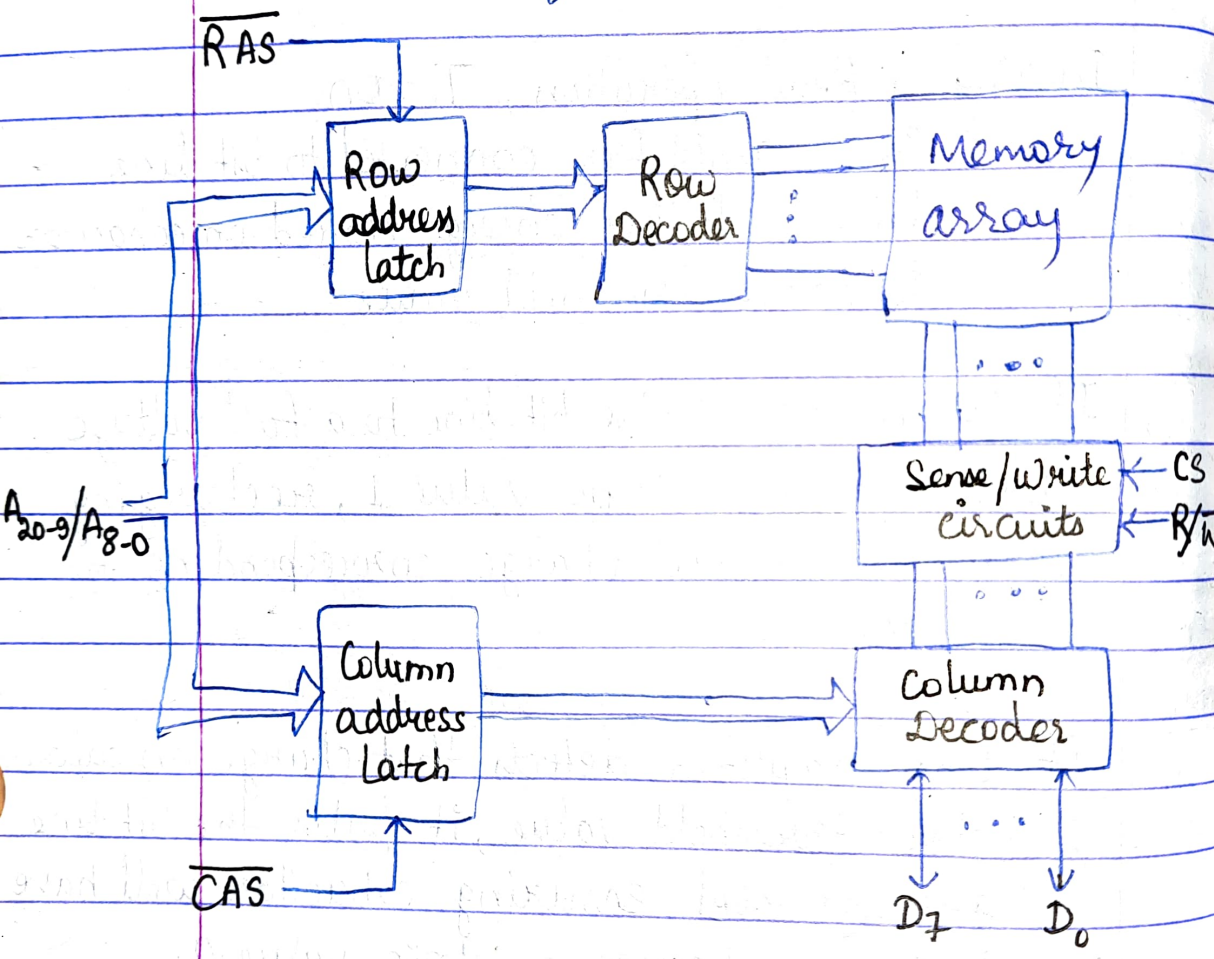
Page No.
Date : | |

① Asynchronous DRAM

↳ Timing of memory device is controlled asynchronously.

→ A specialized memory controller circuit provides necessary control signals $\overline{RAS}$ and that govern timing.

→ Processor must take into account the delay in response of memory.



Internal organization of 2M×8 dynamic memory chip

- 16-megabit DRAM chip configured as 2M×8
- cells organized in form of a 4K×4K array
- 4096 cells in each row are divided into 512 groups of 8, so that a row can store

Page No.
Date : | |

512 bytes of data.

∴ address bits needed to select a row = 12

address bits needed to specify a group of 8-bits in selected row = 9

∴ 21-bit address is needed to access a byte in this memory.

- highorder
  12-bits
  ↓
  row address
- low order
  9-bits
  ↓
  Column address

- To reduce no. of pins needed for external connections, row and column addresses are multiplexed on 12 pins

During a Read or a Write operation
↳ row address is applied first
(loaded into row address latch in response to $\overline{RAS}$ i/p of chip)

- Then a Read operation is initiated, in which all cells on the selected row are read and refreshed
- Shortly after row address is loaded, column address is applied to address pins & loaded into column address latch under control of $\overline{CAS}$ signal.

Page No.
Date : | |

→ Information in this latch is decoded and appropri group of 8 Sense/Write circuits are selected

If $R/\overline{W} = 1$ (Read operation)
↳ o/p values of selected ckts are transferre to data lines D_{7-0}

If $R/\overline{W} = 0$ (Write operation)
↳ information on D_{7-0} lines is transferred to selected circuits. This information is then used to overwrite the contents of selected cells in corresponding 8 columns

* Applying a row address causes all cells on corresponding row to be read and refreshed during both Read and Write operation.

To ensure that contents of a DRAM are maintai each row of cells must be accessed periodically. A refresh circuit usually performs this function automatically.
Many dynamic memory chips incorporate a refresh facility within chips themselves.

adv. Because of their high density and low cost, DRAMs are widely used in the memory units of computers.

Memory Interfacing and Addressing

Basic Problem

- Interfacing one or more memory modules to the processor
- Assume a single level memory at present (i.e. no cache memory)

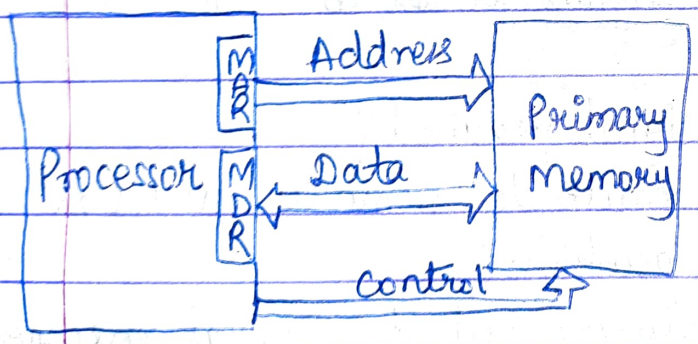
Questions to be answered

→ How processor address and data lines are connected to memory modules?

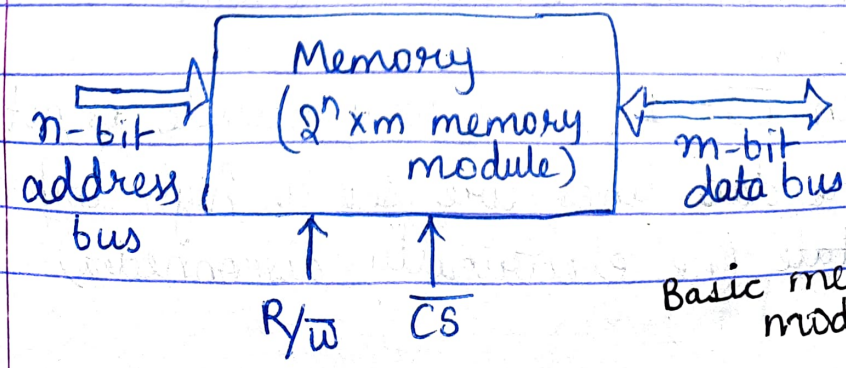
→ How are the addresses decoded?

→ How are the memory addresses distributed among the memory modules?

→ How to speed up data transfer rate between processor and memory?



processor interconnection with memory



Basic memory module

Page No.
Date : / /

Note about memory Interface Signals

- Data signals of a memory module (RAM) are typically bidirectional.
  → Some memory chips may have separate data in and data out lines

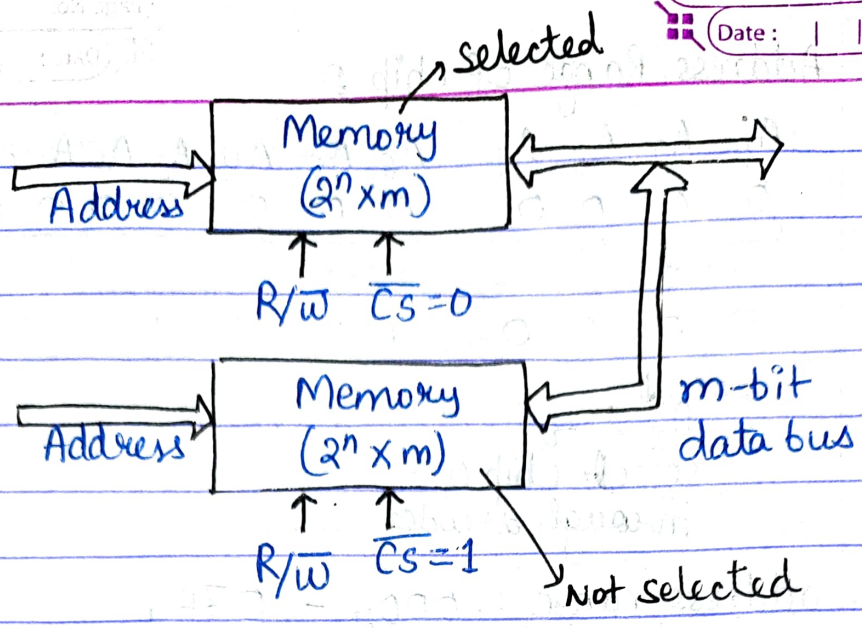
- For memory READ operation
  → Address of memory location is applied to address lines.
  → $R/\overline{W} = 1$ and $\overline{CS} = 0$
  → Data is read out through data lines after memory access time delay.

- For memory WRITE operation
  → Address of memory location is applied to address lines, and data to be written to data lines
  → $R/\overline{W}$ control signal is set to 0 and $\overline{CS} = 0$

Imp Why is $\overline{CS}$ signal required?
→ To hande multiple memory modules interfacing problem
→ Typically only one out of several memory modules is selected at a time.

when $\overline{CS} = 1$, a memory module is not selected the data lines are set to high impedance state (i.e. electrically disconnected)

e.g.



How to build larger memory from smaller memories?

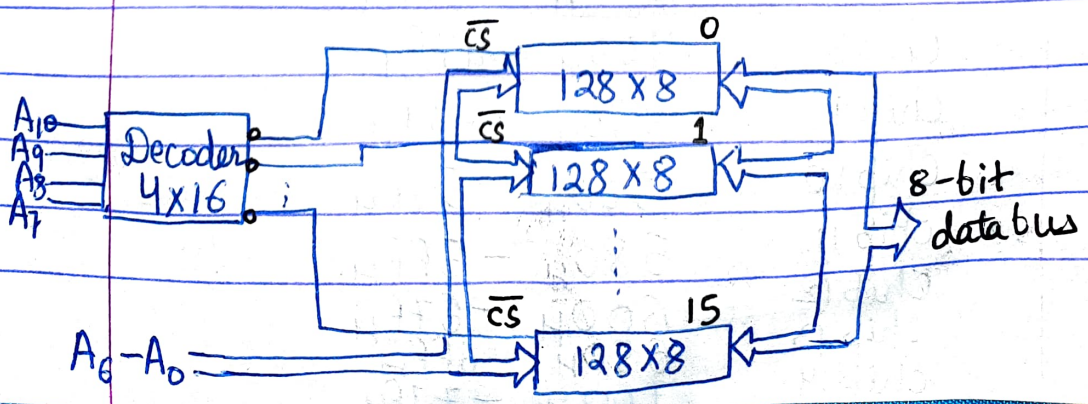
e.g. ①. How many 128 X 8 RAM chips are needed to provide a memory capacity of 2048 X 8?

→ $2048 = 2 \times 1024 = 2 \times 2^{10} = 2^{11}$

↳ 11 address bits required
$A_{10} \ldots A_1 A_0$

whereas $128 = 2^7$ → 7 address bits req.
$A_6 \ldots A_1 A_0$

No. of chips req. $= \dfrac{2048 \times 8}{128 \times 8} = 16$



Page No.
Date : | |

Address Range of chip 0

A_{10} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0

0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 1 1 1 1 1 1

fixed to select chip 0 through decoder

∴ address range : $000_H - 07F_H$

Similarly

for chip 1 : address range: A_{10} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2

0 0 0 1 0 0 0 0 0

0 0 0 1 1 1 1 1 1

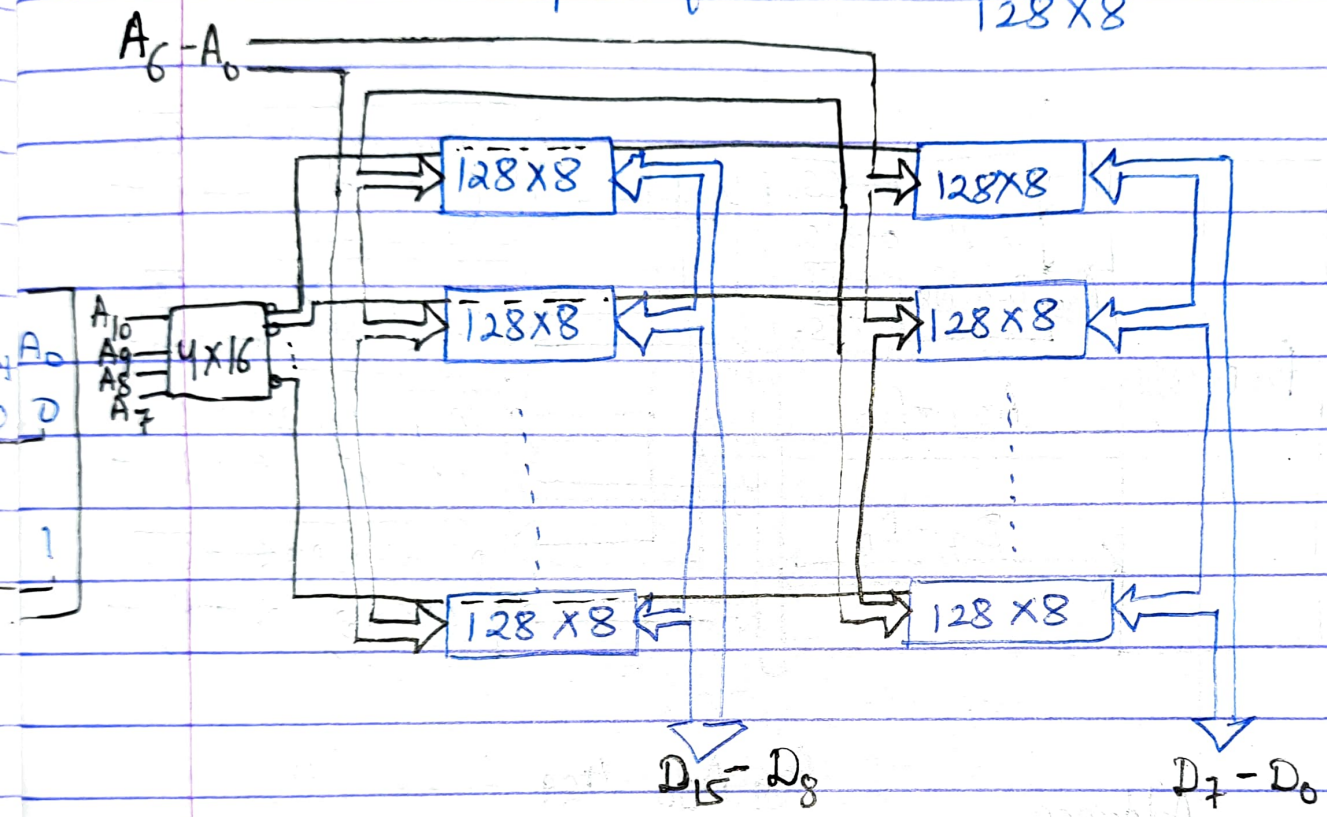
∴ $080_H - 0FF_H$

	Address Range
for chip 2	$100_H - 17F_H$
chip 3	$180_H - 1FF_H$
chip 4	$200_H - 27F_H$
chip 5	$280_H - 2FF_H$
chip 6	$300_H - 37F_H$
chip 7	$380_H - 3FF_H$
chip 8	$400_H - 47F_H$
chip 9	$480_H - 4FF_H$
chip 10	$500_H - 57F_H$
chip 11	$580_H - 5FF_H$
chip 12	$600_H - 67F_H$
chip 13	$680_H - 6FF_H$
chip 14	$700_H - 77F_H$

Chip 15 $780_H - 7FF_H$

e.g ② How many 128X8 RAM chips are required to provide a memory capacity of 2048X16?

No. of chips required $= \frac{2048 \times 16}{128 \times 8} = 32$



Example Memory Interfacing Problem

- Consider processor with 32-bit address bus and 8-bit data bus

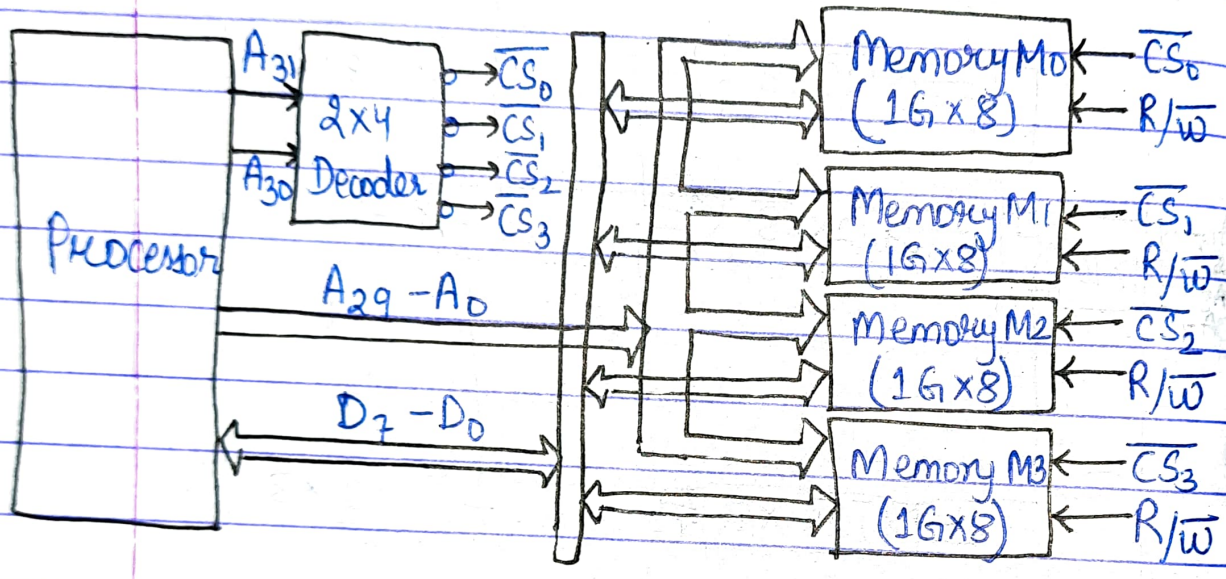
  ∴ Maximum memory that can be connected $= 2^{32} \times 8 = 4$ Gbytes

- Assume that memory chips (RAM) are available with size 1GB.

  ↳ 30 address lines
  8 data lines

∴ Lower order 30 address lines (A_{29}–A_0) are connected to memory modules.

- 4 Such chips are required to be interfaced with processor as it can access maximum 4GB memory.



	A_{31}	A_{30}	A_{29}	.....	A_0	
Address range of M_0	0	0	0	.....	0	⇒ 00000000 to 3FFFFFFF
	0	0	1	.....	1	
Address range of M_1	0	1	0	.....	0	⇒ 40000000_H to $7FFFFFFF_H$
	0	1	1	.....	1	
Address range of M_2	1	0	0	.....	0	⇒ 80000000_H to $BFFFFFFF_H$
	1	0	1	.....	1	
Address range of M_3	1	1	0	.....	0	⇒ $C0000000_H$ to $FFFFFFFF_H$
	1	1	1	.....	1	

An observation

→ Consecutive block of bytes are mapped to the same memory module.

→ In this case, access to 32 bits (4-bytes) data can be done in parallel which requires four sequential memory accesses here.

Alternate memory organization to make this possible → called memory interleaving.

Improved Memory Interface

↳ Small changes done in organization so that 32-bits of data can be fetched in a single memory access cycle.
(using concept of memory interleaving)

Main changes:

→ High order 30 address lines (A_{31} - A_2) are connected to memory modules.

→ low order two address lines (A_1 and A_0) are used to select one of the modules.

Addresses mapped to memory modules

- Module M0 : 0, 4, 8, 12, 16, 20, 24, ...
- Module M1 : 1, 5, 9, 13, 17, 21, 25, ...
- Module M2 : 2, 6, 10, 14, 18, 22, 26, ...
- Module M3 : 3, 7, 11, 15, 19, 23, 27, ...

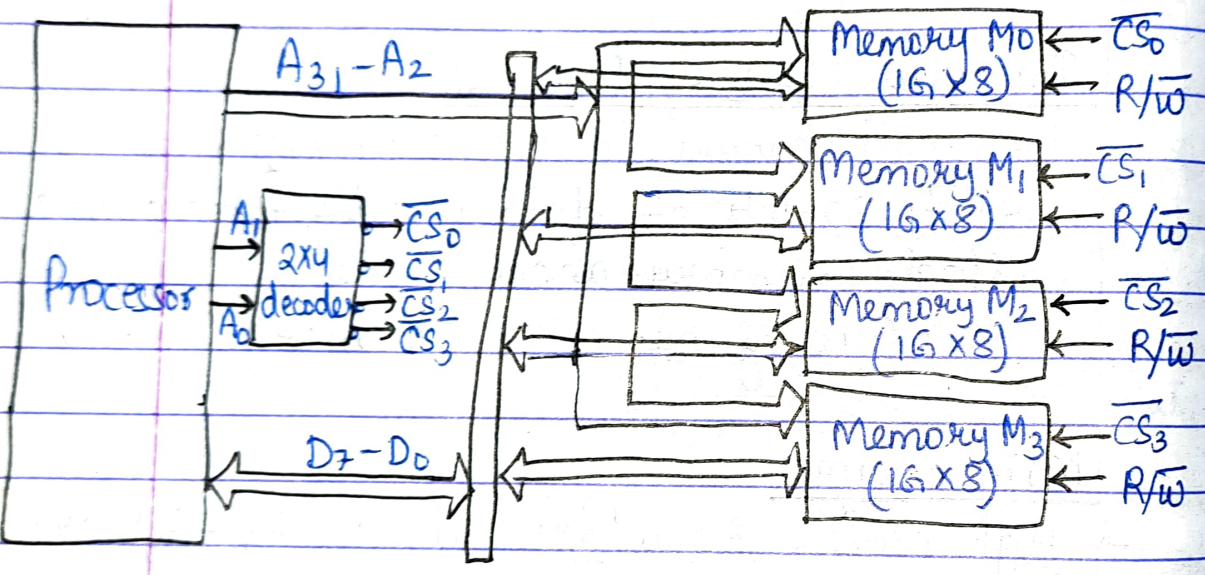
→ Memory addresses are interleaved across memory modules.

Advantages of this mapping

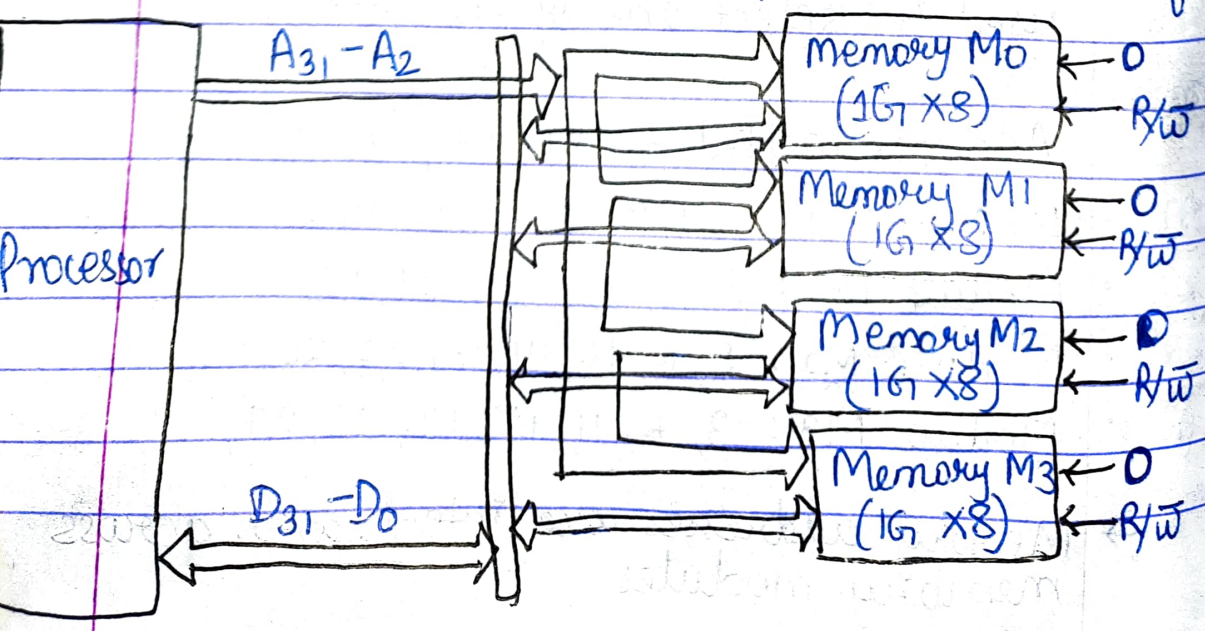
→ Consecutive addresses are mapped to consecutive modules.

→ Possible to access four consecutive words in same cycle, if all four modules are enabled simultaneously.

disadv : Still one module is selected at a time.
8 bits data transfer per cycle



Modification → Enable all four modules together
32-bit parallel data transfer



Memory Latency and Bandwidth

Memory Latency : delay from issue of a memory read request to first byte of data becoming available.

Memory Bandwidth

↳ maximum no. of bytes that can be transferred b/w processor and memory system per unit time.

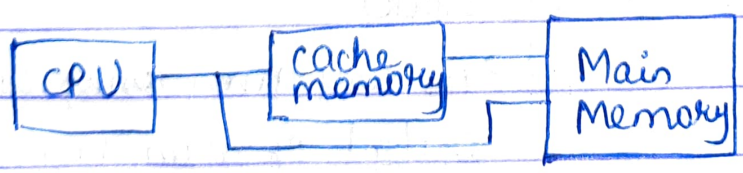
e.g. Consider a memory system that takes 20ns to service access of a single 32-bit word

- Latency $L = 20\,ns$ per 32-bit word
- Bandwidth $BW = \dfrac{32}{20 \times 10^{-9}} = 200$ Mbytes per second

Cache memory

→ Let us consider single level cache

→ memory hierarchy consists of cache memory and main memory



→ Cache memory is logically divided into blocks or lines, where every block (line) typically contains 8 to 256 bytes.

- When CPU wants to access a word in memory, a special hardware first checks whether it is present in cache memory.
  - → If so (called cache hit), the word is directly accessed from cache memory.
  - → If not, block containing the requested word is brought from main memory to cache
  - → for writes, sometimes CPU can also directly write to main memory.

- Objective is to keep commonly used blocks in cache memory.
  - ↳ result in significantly improved performance due to property of locality of reference.

Imp. Where can a block be placed in cache memory?
↳ determined by some mapping algorithm
- ↳ specifies which main memory blocks can reside in which cache memory blocks
- → At any given time, only a small subset of main memory blocks can be held in cache memory

Three common block mapping techniques
1. Direct Mapping
2. Associative Mapping
3. (N-way) Set Associative Mapping

Page No.
Date : | |

e.g. 2-level memory hierarchy

Consider 2-level cache memory / main memory hierarchy

→ cache memory has 256 blocks (lines) of 32 words each

Total cache size = 8192 words
↳ 8K words

→ Main memory is addressable by a 24-bit address

- Size of main memory $= 2^{24} = 16M$ words
- No. of 32-word blocks in main memory $= 16M/32$
  $= 512K$

ⓐ. Direct Mapping

- Each main memory block can be placed in only one block in cache.

- Mapping function is:

$$\boxed{\text{Cache Block} = (\text{Main Memory Block}) \% (\text{No. of cache blocks})}$$

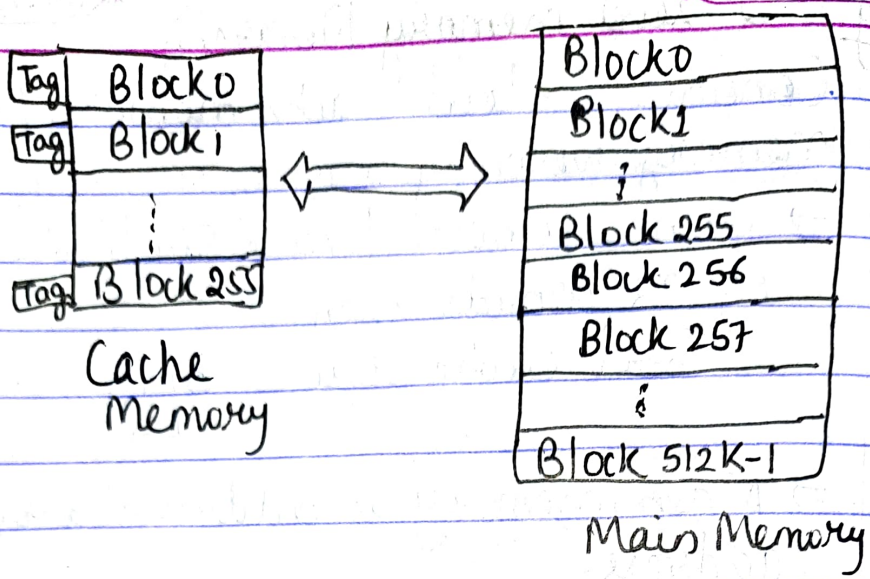
e.g. cache Block = Main memory Block % 256

- Mapping example

0 → 0, 1 → 1, 255 → 255,
256 → 0, 257 → 1, 512 → 0, 513 → 1, etc.

disadv: 0, 256, 512 .. soon can be mapped only in block 0 in cache memory.

∴ at a time all these blocks of main memory cannot be found in cache



Direct Mapping

Tag (11-bits)	Block (8-bits)	Word (5-bits)

Memory address

Tag → tell which block of main memory is mapped into a particular block of cache memory

$$\text{Tag} = \frac{\text{No. of Blocks in main memory}}{\text{No. of Blocks in cache memory}}$$

$$= \frac{512K}{256} = \frac{2^{19}}{2^{8}} = 2^{11}$$

∴ 11-bits required to represent tag.

Imp.

- Block replacement algorithm is trivial, as there is no choice.
- More than one MM block is mapped onto the

Same cache block.
→ May lead to contention even if cache is not full
→ New block will replace old block
→ May lead to poor performance if both the blocks are frequently used.

- MM address is divided into three fields:
  - TAG
  - BLOCK
  - WORD

→ When a new block is loaded into the cache, 8-bit BLOCK field determines cache block where it is to be stored.

→ High-order 11-bits are stored in a TAG register associated with cache block.

→ When accessing a memory word, corresponding TAG fields are compared.
↳ Match implies HIT

(b). <u>Associative Mapping</u>
- A MM block can potentially reside in any cache block position.
- Memory address is divided into two fields
  ↳ TAG
  WORD

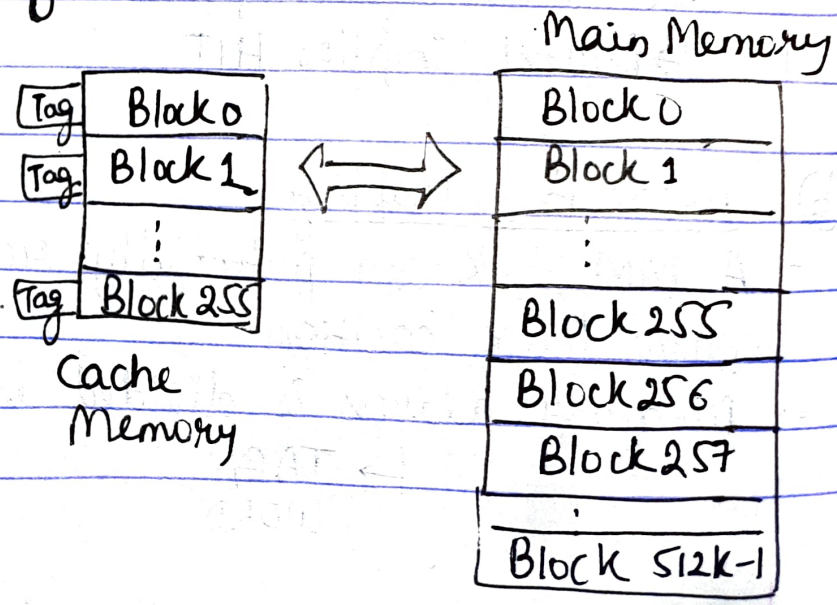
→ When a block is loaded into cache from MM, higher order 19 bits of address are stored into TAG register corresponding to cache block.

→ When accessing memory, 19-bit TAG field of address is compared with <u>all the TAG registers</u> corresponding to all cache blocks

- Requires associative memory for storing TAG values
  ↳ High cost / lack of scalability

- Because of complete freedom in block positioning, a wide range of replacement algorithms is possible.

Fully Associative Mapping



TAG	WORD
19	5

Memory Address

(c). N-way Set Associative Mapping

- A group of N consecutive blocks in cache is called a set.
- This algorithm is a balance of direct mapping and associative mapping.

→ Like direct mapping, a MM block is mapped to a set

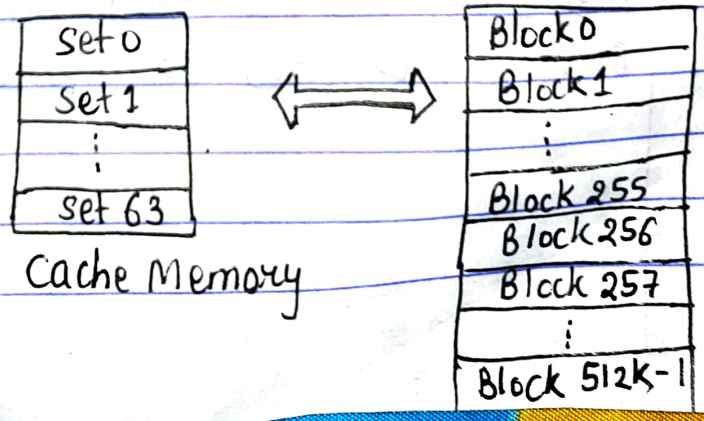
$$\text{Set Number} = (\text{MM Block Number}) \% (\text{No. of Sets in Cache})$$

→ Block can be placed anywhere within the set (there are N choices).

- The value of N is a design parameter
  - * N = 1 ⇒ same as direct mapping
  - * N = no. of cache blocks ⇒ same as associative mapping

* Typical values of N used in practice are: 2, 4 or 8

4-way set Associative Mapping



TAG	SET	WORD
13	6	5

Memory Address

Page No.
Date : | |

$$TAG = \frac{\text{Total no of Blocks in MM}}{\text{Total no. of sets in CM}} = \frac{512\,k}{64}$$

$$= \frac{2^{19}}{2^{6}} = 2^{13}$$

N = 4

- → no. of sets in cache memory = 64
- → memory blocks are mapped to a set using modulo-64 operation
- → eg. MM blocks 0, 64, 128 etc. all map to set 0, where they can occupy any of the four available positions

• MM address is divided into three fields
- → TAG
- → SET
- → WORD

→ TAG field of address must be associatively compared to TAG fields of 4 blocks of selected set.

→ This instead of requiring a single large associative memory, no. of very small associative memories are required out of which only one will be used at a time.

Page No.
Date : | |

Imp How is a block found if present in cache?

- Caches include a TAG associated with each cache bla
→ TAG of every cache block where block being requested may be present needs to be compared with TAG field of MM addres
→ All possible tags are compared in parallel, as speed is important
- Mapping Algorithms?
→ Direct mapping requires a single comparison
→ Associative mapping requires a full associative search over all TAGs corresponding to all cache blocks.
→ Set associative mapping requires a limited associated search over TAGs of only selected set.

* Use of valid bit:
→ There must be a way to know whether a cache block contains valid or garbage information.

→ A valid bit can be added to TAG, which indicates whether block contains valid data.

→ If valid bit = 0 → no need to match corresponding TAG

Page No.
Date : / /

Imp Which block should be replaced on a cache miss?

- With fully associative or set associative mapping, there can be several blocks to choose from for replacement when a miss occurs.
- Two primary strategies are used:

(a). **Random**: Candidate block is selected randomly for replacement. This simple strategy tends to spread allocation uniformly.

(b). **Least Recently Used**: The block replaced is the one that has not been used for longest period of time.

↳ Makes use of a corollary of temporal locality:

"If recently used blocks are likely to be used again, then the best candidate for replacement is the least recently used block."

To implement LRU algorithm, cache controller must track LRU block as computation proceeds.

e.g. Consider a 4-way set associative cache.

→ for tracking LRU block within set, use a 2-bit counter with every block.

- **When hit occurs**:

→ Counter of referenced block is reset to 0.

→ Counters with values originally lower than the referenced one are incremented by 1, and all others remain unchanged.

e.g.

	initial counter values		counter value
B0	01		10
B1	00		01
B2	11		11
B3	10	← now accessing B3	00

Values > 10 → 00, 01
↳ increment by 1

- <u>When miss occurs</u>

→ If set is not full, counter associated with new block loaded is set to 0, and all other counters are incremented by 1.

→ If set is full, block with counter value 3 is removed, new block put in its place, and counter set to 0. The other three counters are incremented by 1.

<u>case 1</u>: if set is not full

	counterValues	counter values when B6 has been brought
B0	10	11
B1	00	01
B2	01	10
B6		00

free

<u>case 2</u>: If set is full

	Counter values		∴ B0 is removed & B7 is brought in new counter values
~~B0~~ B7	11	← Least recently used	
B1	01		00
B2	10		10
B6	00		11
			01

- It may be verified that the counter values of occupied blocks are all distinct

Example

x	Block 0
x	Block 1
x	Block 2
x	Block 3

Initial

x	Block 0
x	Block 1
0	Block 2
x	Block 3

Miss: Block 2

0	Block 0
x	Block 1
1	Block 2
x	Block 3

Miss: Block 0

1	Block 0
x	Block 1
2	Block 2
0	Block 3

Miss: Block 3

2	Block 0
0	Block 1
3	Block 2
1	Block 3

Miss: Block 1

0	Block 0
1	Block 1
3	Block 2
2	Block 3

Hit: Block 0

1	Block 0
2	Block 1
0	Block 2
3	Block 3

Miss: Block 2

2	Block 0
3	Block 1
1	Block 2
0	Block 3

Hit: Block 3

2	Block 0
3	Block 1
0	Block 2
1	Block 3

Hit: Block 2

0	Block 0	1	Block 0	1	Block 0
3	Block 1	0	Block 1	0	Block 1
1	Block 2	2	Block 2	2	Block 2
2	Block 3	3	Block 3	3	Block 3
Hit: Block 0		Miss: Block 1		Hit: Block 1	

Types of cache Misses

① Compulsory Miss

→ On first access to a block, the block must be brought into cache.

→ Also known as cold start misses, or first reference misses

→ Can be reduced by increasing cache block size or prefetching cache blocks.

② Capacity Miss

→ Blocks may be replaced from cache because cache cannot hold all blocks needed by a program

→ Can be reduced by increasing the total cache size.

Page No.
Date : | |

3. Conflict Miss

→ In case of direct mapping or N-way set associative mapping, several blocks may be mapped to same block or set in cache.

→ May result in block replacement and hence access misses, even though all cache blocks may not be occupied.

→ Can be reduced by increasing value of N (cache associativity).

Imp. What happens on a write?

- Statistical data suggests that read operations (including instruction fetches) dominate processor cache accesses.
  → All instruction fetch operations are read.
  → Most instructions do not write to memory.

- Making common case fast:
  → Optimize cache accesses for reads.
  → But Amadahl's law reminds that for high performance designs, speed of write operations cannot be ignored.

- Common case (read operations) is relatively easy to make faster.

Page No.
Date : | |

→ A block(s) can be read at same time while TAG is being compared with block address.

→ If read is a HIT, data can be passed to CPU; if it is a MISS ignore it.

• Problems with write operations:

→ The CPU specifies size of write (b/w 1 and 8 bytes), and only that portion of a block has to be changed.
  → Implies a read-modify-write sequence of operations on the block.
  → Also, process of modifying block cannot begin until TAG is checked to see if it is a hit.

→ Thus, cache write operations take more time than cache read operation.

Cache Write Strategies

↳ cache designs can be classified based on write and memory update strategy being used:

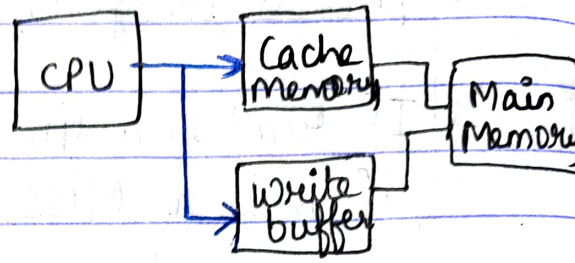
① Write through / store through (update cache & MM simultaneously)

② Write back / copy back
(write into cache for time being and later write back to MM from cache)

Page No.
Date : | |

①. Write Through Strategy

- Information is written to both cache block and main memory block.



- <u>features</u>

→ Easier to implement
→ Read misses do not result in writes to lower level (i.e. MM)
→ Lower level (i.e., MM) has most updated version of data - important for I/O operations and multiprocessor systems.
→ A write buffer is often used to reduce CPU write stall time while data is written to main memory.

Perfect Write Buffer
→ All writes are handled by write buffer; no stalling for write operations
→ for unified L1 cache

$$\frac{\text{Stall cycles}}{\text{Memory Access}} = \%\ \text{Reads} \times (1 - H_{L1}) \cdot t_{MM}$$

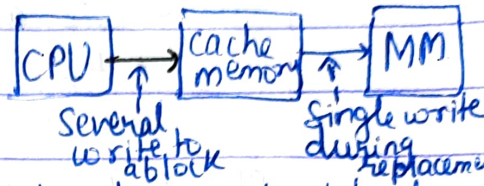
H_{L1} → hit rate of L1 cache
t_{MM} → main memory access time

Realistic Write Buffer
↳ % of write stalls are not eliminated when write buffer is full.

for unified L1 cache

$$\frac{\text{Stall cycles}}{\text{Memory Access}} = (\%\ \text{Reads} \times (1 - H_{L1}) + \%\ \text{write Stalls not eliminated}) \times t_M$$

②. Write back Strategy



- Information is written only to cache block
- A modified cache block is written to MM only when it is replaced.
- features:
  - → Write occur at speed of cache memory.
  - → Multiple writes to a cache block requires only one write to MM.
  - → Uses less memory bandwidth, makes it attractive to multiprocessors.
- Write-back cache blocks can be clean or dirty
  - → A status bit called dirty bit or modified bit is associated with each cache block, which indicates whether block was modified in the cache (0: clean, 1: dirty)

→ If status is clean, block is not written back to MM while being replaced.

Hit rate: no. of hits stated as a fraction of all attempted accesses

Miss rate: no. of misses stated as a fraction of all attempted accesses

Extra time needed to bring desired into cache

↳ miss penalty → time needed to bring data block from slower unit in memory hierarchy to a faster unit

average access time experienced by processor

$$\boxed{t_{ave} = hC + (1-h)M}$$

h → hit rate

M → miss penalty

C → time to access info in cache

Page No.
Date : | |

Cache Memories

Speed of main memory is very low in comparison with speed of modern processors.
↳ ∴ a scheme is required to reduce time needed to access necessary informatio

Speed of main memory unit is limited by electronic and packaging constraints, sol^n must be sought in a different architectural arrangement.
↳ efficient sol^n is to use a fast cache memory that essentially makes main memory appear processor to be faster than it really is.

Effectiveness of cache mechanism is based on a property of computer programs called <u>locality of reference</u>.
↳ most of execution time is spent on routines in which many instructions are executed repeatedly.

∴ Many instructions in localized areas of program are executed repeatedly during some time period and remainder of program is accessed relatively infrequently
↳ Locality of reference

locality of reference manifests itself in two ways
- temporal
- spatial

→ Temporal means that a recently executed instruction is likely to be executed again very soon.

→ Spatial aspect means that instructions in close proximity to a recently executed instructi (w.r.t. instructions' addresses) are also likely to be executed soon.

If active segments of a program can be placed in a fast cache memory, then total execution time can be reduced significantly.

Conceptually, operation of a cache memory is very simple.
↳ Memory control circuitry is designed to take advantage of property of locality of reference.

<u>Temporal aspect</u> → whenever an information item is first needed, it should be brought into cache where it will remain until it is needed again.

Page No.
Date : | |

<u>Spatial Aspect</u> → instead of fetching just one item from main memory to cache, it is useful to fetch several items that reside at adjacent addresses as well.

block → a set of contiguous address locations of some size

Cache block is also known as cache line.