PAGE No.
DATE / /

INTRODUCTION TO COMPUTER ORGANIZATION AND ARCHITECTURE

BASIC FRAMEWORK

* ARCHITECTURE → instruction set, → number of bits, → I/O mechanism
  → attributes of system visible to the programmer.
  → attributes that have direct impact on the logical execution of the program.

HOW IT SHOULD BE USED.

* ORGANIZATION → control signals, → interface between the comp. & peripheral.
  → operational units and their interconnection that realize the architectural specification.

* Architectural design issue is ~~when a co~~ whether a computer will have a multiply instruction or repeated addition unit.

* Organizational design issue is when it has to be decided if a ~~single~~ we require a multiplication unit or a an addition unit for repeated addition to perform multiplication.

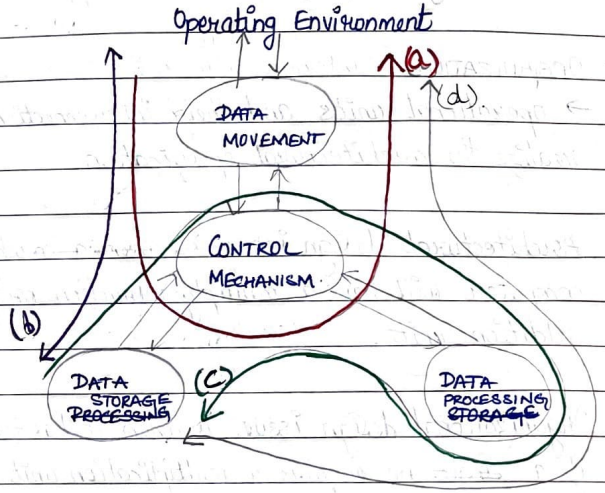
Organization decision is based on
→ anticipated frequency of use
→ relative speed
→ cost and physical size.

* FUNCTIONS OF A COMPUTER:
  → DATA PROCESSING.
  Data recieved has to be processed for further use (sorting, searching, etc)

* The comp. must store the data temporarily that have to be worked at the given movement.

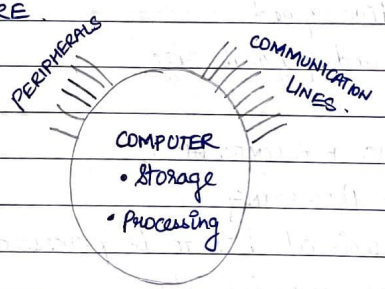
PAGE No. / DATE / /

→ DATA STORAGE.
To store data.
→ DATA MOVEMENT.
To process data ~~fast~~ from any given location.
→ CONTROL
To control above 3 functions.



(a) movement of data
(b) data storage.
(c) processing of stored data
(d) taking new data and storing or working on stored data.

* STRUCTURE.



TOP - LEVEL STRUCTURE.

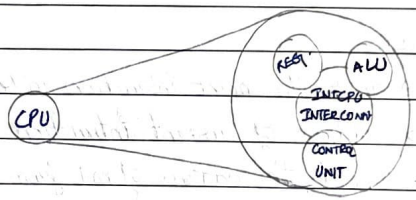
Von - Neuman model.
* Common memory for instructions & program data & a single bus to access the data.
Harvard Architecture - multiple buses to access data.

PAGE No. / DATE / /

→ Comp. is an entity that interacts in some fashion with its external environment.
→ All linkages to the external environment can be classified as peripheral devices or communication lines.

1. CPU → CONTROL UNIT, ALU, REGISTERS.
2. Main Memory
3. I/O Devices.
4. System interconnections.

} Further division of the structure.



Control unit further consists of.
1. Control Memory
2. Control Unit Register & decoders.
3. Sequencing logic

* GENERAL SYSTEM ARCHITECTURE.

STORE PROGRAM CONTROL CONCEPT

FLYNN'S CLASSIFICATION.

SHORT NOTE → VON-NUMAN MODEL.
→ Type of Architecture
→ General information.
→ Features
→ DIAGRAM.
→ Explanation of registers and other parts.

PAGE No.
DATE / /

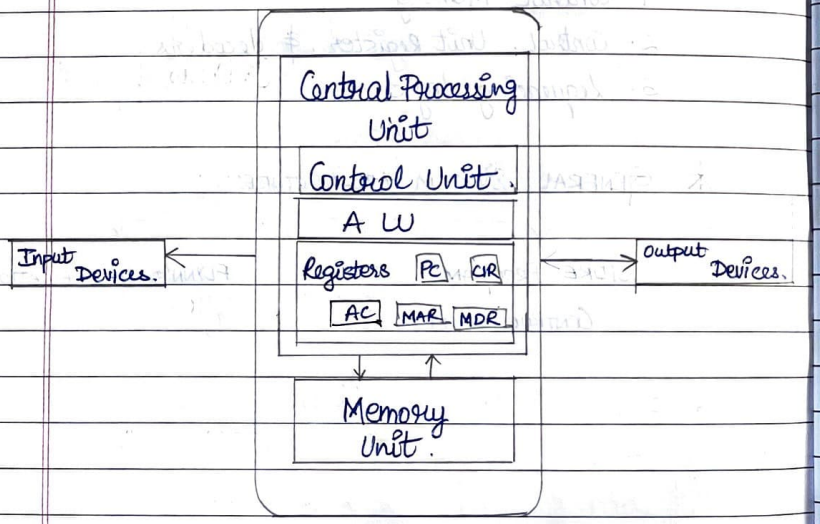
→ storage of instruction in computer memory to enable it to perform a variety of tasks.

* VON-NEUMANN MODEL.

FEATURES
→ Uses a single processor.
→ Uses one memory.
→ Uses fetch-decode-execute approach.

It uses binary instructions.

REGISTERS :
PC → gives the next instruction to be executed address.
CIR → address of current instruction to be executed.
MAR → holds the address of m.l. from where data is accessed.
MDR → stores data that is transfered.
AC → holds intermediate arithematic & logic operation.

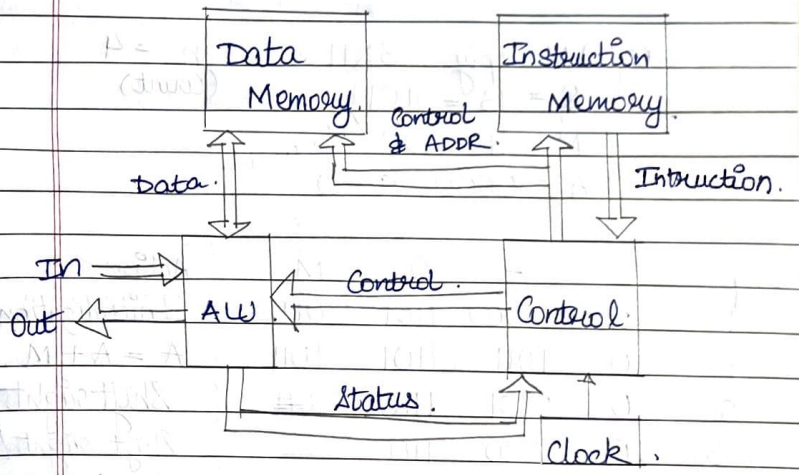


2's complement / 0/8/16/FD 1's complement / 0/7/15/D.
↓ RADIX complement ↓ REDUCED RADIX compliment

PAGE No.
DATE / /

* HARVARD ARCHITECTURE.

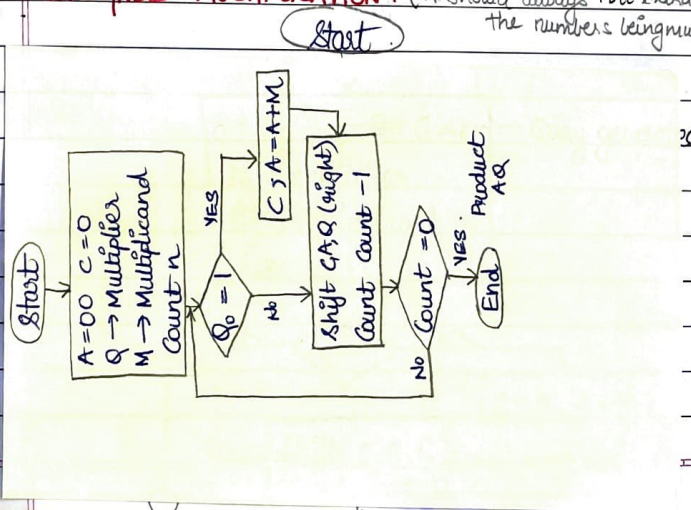
Seperate memory for data and instructions (i.e. stored in different ports).
Multiple buses ~~to~~ for fetching data.



* DATA REPRESENTATION AND ARTHIMETIC ALGORITHMS.

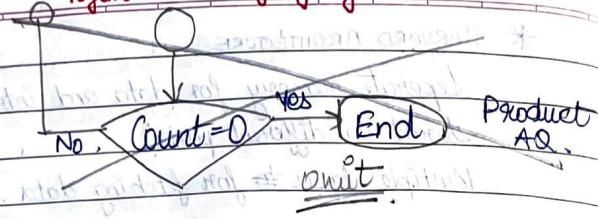
Binary, Octal ~~and~~ Arithematics.

SHIFT AND ADD METHOD ← UNSIGNED MULTIPUCATION. (A should always 1 bit extra than the numbers being multiplied)



13 × 11
Q ↗ ↖ M.

* C, A & Q should be considered together while shifting right



eg 1) Multiply 13 × 11. $n = 4$ (Count)

$Q = 13 = 1101$

$M = 11 = 1011$.

$A = 0000$ $C = 0$

Step	C	A	Q	M	Action.
1	0	0000	1101	1011	Initialization.
2.	0	1011	1101	1011	A = A + M.
	0	0101	1110		Shift right / n-1 (3)
3.	0	0010	1111		Shift right / n-1 (2)
4.	0	1101	1111	1011	A = A + M.
	0	0110	1111		Shift right / n-1 (1).
5.	1	0001	1111	1011	A = A + M.
	0	1000	1111		Shift right / n-1 (0).

$(143)_{10}$.

∴ (1101) × (1011) = (10001111)

13 × 11 = 143.

2) Multiply 5 × 13. $n = 4$

$Q = 5 = 0101$

$M = 13 = 1101$

$A = 0000$ $C = 0$.

1111
0001
10000

Steps	C	A	Q	M	n	Action.
1.	0	0000	0101	1101	4	Initialization
2.	0	1101	0101	1101	4	A = A + M
	0	0110	1010		3	Shift right
3.	0	0011	0101		2	Shift right
4	0	0000	0101			A = A + M.
	0	1000	0010		1	Shift right
5.	0	0100	0001		0	Shift right

65.

eg. 3) Multiply 9 × 15 $n = 4$.

$Q = 9 = 1001$

$M = 15 = 1111$

$A = 0000$ $C = 0$

Steps	C	A	Q	M	n	Action.
1	0	0000	1001	1111	4	Initialization
2.	0	1111	1001	1111		A = A + M
	0	0111	1100		3	Shift right
3.	0	0011	1110		2	Shift right
4.	0	0001	1111		1	Shift right.
5.	1	0000	1111			A = A + M
	0	1000	0111		0	Shift right

135.

SIGNED MULTIPLICATION → BOOT'S ALGORITHM.

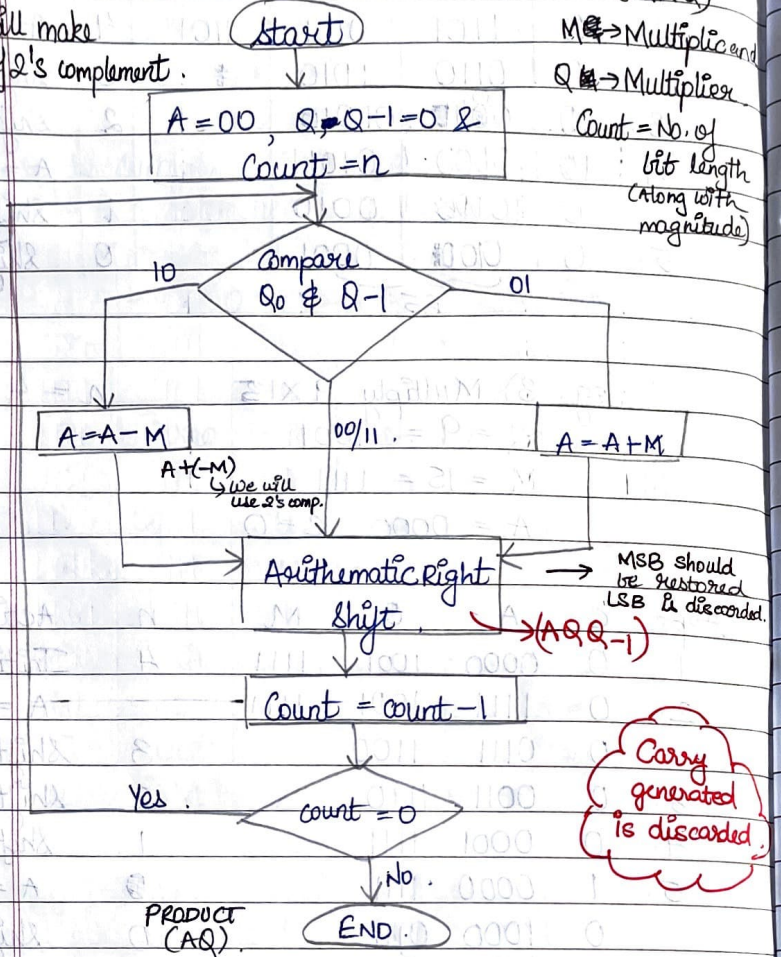
(M X Q)

M → Multiplicand

Q → Multiplier

Count = No. of bit length (along with magnitude)

We will make use of 2's complement.



Carry generated is discarded.

eg. 1) M = 7 Q = -3.

M = 0111.

1's complement of M ⇒ 1000

2's complement of M ⇒ 1001. → -M

While finding 1's & 2's comp. for the signed no. signed bit (MSB) is not changed. (when you count it's magnitude only changed).

Q = 1011.

1's complement of Q ⇒ 1100.

2's complement of Q ⇒ 1101.

Steps.	A	Q	Q-1	n	Action.
1.	0000	1101	0	4	Initialization.
2.	1001	1101	0		A = A - M.
	1100	1110	1	3.	ARS.
3.	0011	1110	1		A = A + M.
	0001	1111	0	2	ARS.
4.	1010	1111	0		A = A - M.
	1101	0111	1	1	ARS.
5.	1110	1011	1	0.	ARS.

11101011.

If the MSB is 1, then the answer is not in true form. For true will find its 2's complement and the answer will be negative.

If the MSB is 0, the answer is in true form.

Ans ⇒ 11101011.

1's complement ⇒ 00010100

2's complement ⇒ 00010101.

∴ Ans ⇒ -21.

eg 2. M = −7 , Q = −3.

−M ⇒ 1111

1's complement → 1000.

2's complement → 1001. = +M

−Q ⇒ 1011

1's complement → 1100 omit.

2's complement → 1101. = +Q.

Steps.	A	Q	Q-1	n	Action.
1.	0000	1101	0	4	Initialization.
2.	1111	1101	0		A = A − M.
	1111	1110	1	3	ARS.
3.	1000	1110	1		A = A + M.
	1100	0111	0	2	ARS.
4.	1011	0111	0		A = A − M.
	1101	1011	1	1	ARS.
5.	1010	1101	1	0	

eg. 2) M = −7 , Q = −3.

M = 1111.

1's complement ⇒ 1000

2's complement ⇒ 1001. → M.

1's complement ⇒ 0110

2's complement ⇒ 0111 ⇒ −M.

Q = 1011

1's complement ⇒ 1100

2's complement ⇒ 1101 ⇒ Q.

0011 + 1001 = 1100

0110 + 0111 = 1101

Steps.	A	Q	Q-1	n	Action.
1.	0000	1101	0	4	Initialization.
2.	0111	1101	0		A = A − M.
	0011	1110	1	3	ARS.
3.	1100	1110	1		A = A + M
	1110	0111	0	2	ARS.
4.	0101	0111	0		A = A − M
	0010	1011	1	1	ARS.
5	0001	0101	1	0	ARS.

00010101.

∴ Ans = $(10101)_2$ = 21.

eg. 3) M = −7 Q = 3.

M = 1111 M = 1001 −M = 0111.

Q = 0011.

Steps	A	Q	Q-1	n	Action.
1.	0000	0011	0	4	Initialization.
	+0111				
2.	0111	0011	0		A = A − M
	0011	1001	1	3.	ASR.
3	0001	1100	1	2	ASR.
	+1001				
4.	1010	1100	1		A = A + M.
	1101	0110	0	1	ASR.
5.	1110	1011	0	0	ASR.

Ans = 2's complement of $(11101011)_2$

= $(00010101)_2$

= −21.

Q. Perform boot multiplication for -11×13.

A. $M = -11 = 11011$.

$M = 10101 \quad -M = 01011$.

$Q = 01101$.

Steps	A	Q	Q-1	n	Action
1	00000	01101	0	5	Initialization.
	+01011				
2.	01011	01101	0		A=A-M.
	00101	10110	1	4.	ASR.
	+10101				
3.	11010	10110	1		A=A+M.
	11101	01011	0	3	ASR.
	+01001				
4	00100	01011	0		A=A-M.
	00010	00101	1	2	ASR.
5.	00001	00010	1	1	ASR.
	+10101				
6.	10111	00010	1		A=A+M.
	11011	10001	0	0	ASR.

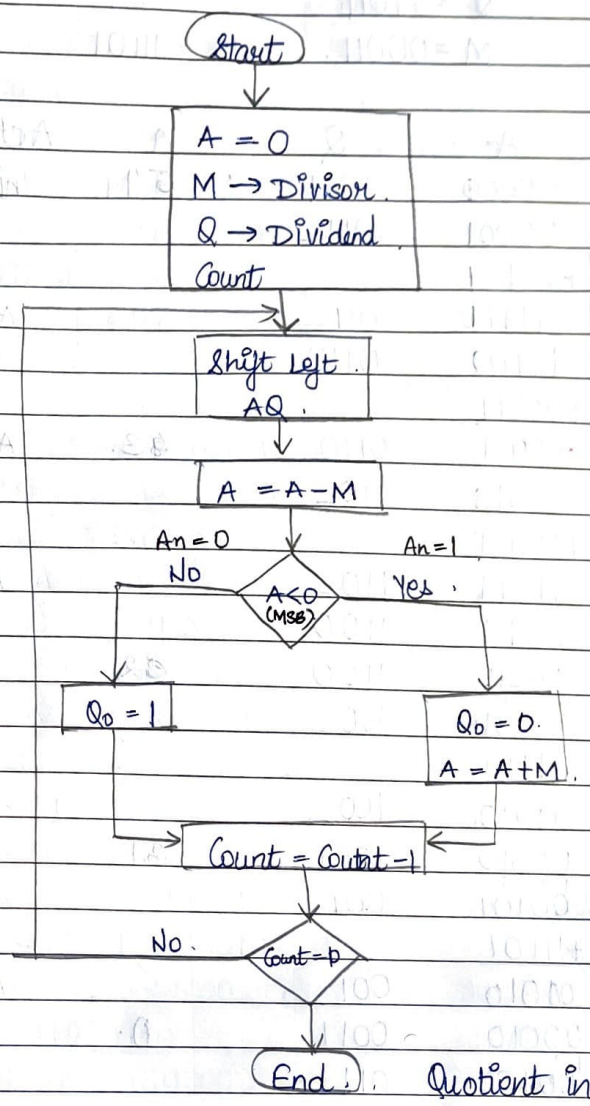
Ans = 2's complement $(1101110001)_2$

$= (0010001111)_2$

$= -143$

* If Dividend & Divisor are of length n then A will (n+1) length.
* When shifting left, LSB should remain blank.

RESTORING DIVISION.



Quotient in Q
Remainder in A.

n → length of Q.

eg. 11/3.

Q = 1011

M = 00011. −M = 11101.

Steps	A	Q	n	Action
1.	00000	1011	4	Initialization
2.	00001	011_		SL.
	+11101			
	11110	011_		A = A−M.
	11110	0110		$Q_0 \rightarrow 0$
	+00011			
	00001	0110	3.	A = A+M.
3.	00010	110_		SL
	+11101			
	11111	110_		A = A−M.
	11111	1100		$Q_0 \rightarrow 0$
	00010	1100	2	A = A+M.
4.	00101	100_		SL.
	+11101			
	00010	100_		A = A−M.
	00010	1001	1	$Q_0 \rightarrow 1$
5.	00101	001_		SL
	+11101			
	00010	001_		A = A−M.
	00010	0011	0.	$Q_0 \rightarrow 1$
6.	00100	011_		SL
	+11101			
	00001	011_		A = A−M
	00001	0111	0.	$Q_0 \rightarrow 1$.

ORIGINAL VALUE RESTORED

Ans: 3

eg. 17/5.

Q = 10001 M = 000101 −M = 111011.

Steps.	A	Q	n	Action.
1.	000000	10001	5.	Initialization.
2.	000001	0001_		SL
	+111011			
	111100	0001_		A = A−M.
	111100	00010		$Q_0 \rightarrow 0$.
	+000101	00010	4	A = A+M.
3.	000010	0010_		SL
	+111011			
	111101	0010_		A = A−M.
	111101	00100		SL
	000010	00100	3.	A = A+M.
4.	000100	0100_		SL
	+111011			
	111111	0100_		A = A−M
	111111	01000		$Q_0 \rightarrow 0$
	000100	01000	2	A = A+M
5.	001000	1000_		SL
	+111011			
	000011	1000_		A = A−M.
	000011	10001	1	$Q_0 \rightarrow 1$.
6.	000111	0001_		SL
	+111011			
	000010	0001_		A = A−M
	000010	00011	0.	$Q_0 \rightarrow 1$.

A → $(00010)_2$ = $(2)_{10}$

Q → $(0011)_2$ = $(3)_{10}$.

A → $(000010)_2$ = $(2)_{10}$.
Q → $(000011)_2$ = $(3)_{10}$.

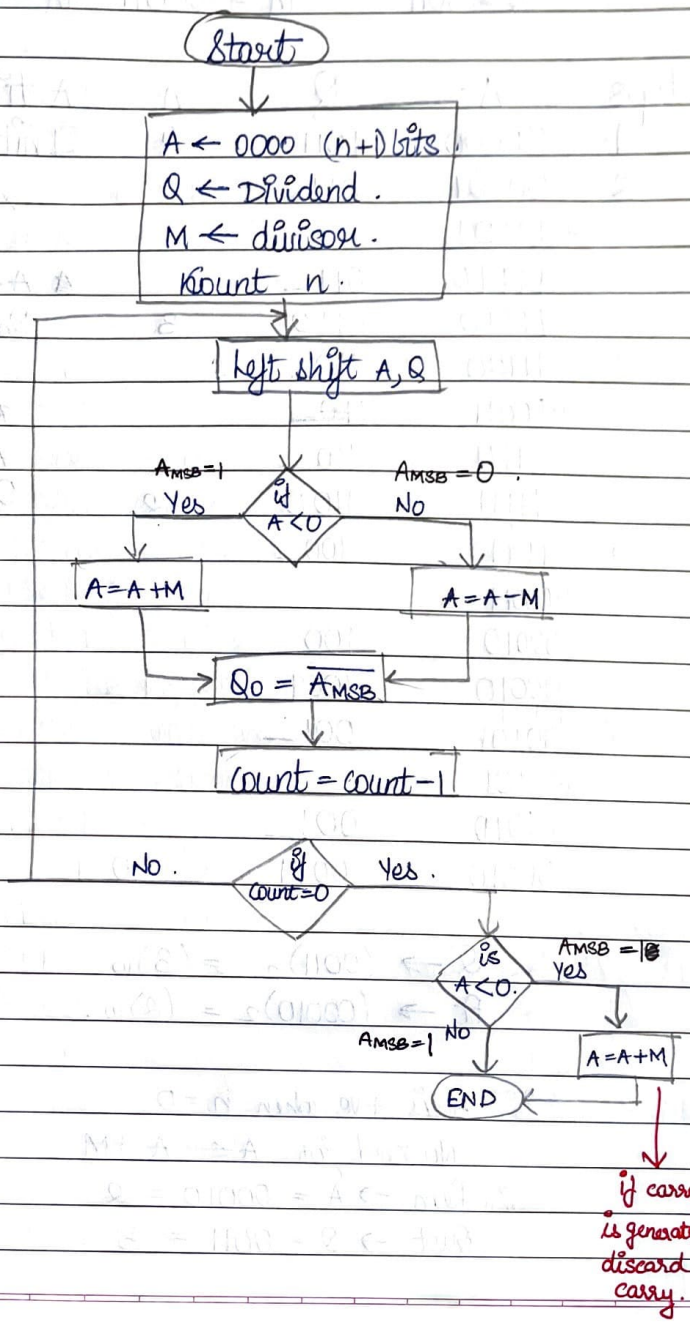
eg. 13/5.

Q → 1101.
M → 00101 −M → 11011

Steps.	A	Q	n	Action.
1.	00000	1101	4	Initialization
2.	00001	101_		SL
	+11011			
	11100	101_		A−M.
	11100	1010		$Q_0 \leftarrow 0$.
	00001	1010	3.	A+M.
3.	00011	010_		SL.
	+11011			
	11110	010_		A−M.
	11100	0100		$Q_0 \leftarrow 0$.
	00011	0100	2.	A+M
4.	00110	100_		SL.
	+11011			
	00001	100_		A−M.
	00001	1001	1	$Q_0 \leftarrow 1$
5.	00011	001_		SL.
	+11011			
	11110	001_		A−M.
	11110	0010		$Q_0 \leftarrow 0$.
	00011	0010	0	A+M.

∴ Q → $(0010)_2$ = $(2)_{10}$
A → $(0011)_2$ = $(3)_{10}$.

NON RESTORING DIVISION



eg. 11/3.

Q = 1011 M = 00011 −M = 11101.

Steps.	A	Q	n	Action.
1.	00000	1011	4	Initialization.
2.	00001	011_		SL.
	+11101			
	11110	011_		A−M.
	11110	0110	3.	$Q_0 \leftarrow \overline{A_{MSB}}$
3.	11100	110_		SL
	+00011			
	11111	110_		A+M.
	11111	1100	2.	$Q_0 \leftarrow \overline{A_{MSB}}$
4.	11111	100_		SL.
	+00011			
	00010	100_		A+M.
	00010	1001	1	$Q_0 \leftarrow \overline{A_{MSB}}$
5.	00101	001_		SL.
	+11101			
	00010	001_		A−M.
	00010	0011	0	$Q_0 \leftarrow \overline{A_{MSB}}$

∴ Q → $(0011)_2 = (3)_{10}$

A → $(00010)_2 = (2)_{10}$.

∴ A is +ve when n=0,

No need for A ← A+M.

∴ Rem → A = 00010 = 2

Quot → Q = 0011 = 3.

eg. 7/3

Q = 111 M = 0011 −M = 1101.

Steps	A	Q	n	Action.
1.	0000	111	3	Initialization
2.	0001	11_		SL.
	+1101			
	1110	11_		A = A−M.
	1110	110	2.	$Q_0 \leftarrow \overline{A_{MSB}}$
3.	1101	10_		SL.
	+0011			
	0000	10_		A+M
	0000	101	1	$Q_0 \leftarrow \overline{A_{MSB}}$
4.	0001	01_		SL
	+1101			
	1110	01_		A−M
	1110	010	0.	$Q_0 \leftarrow \overline{A_{MSB}}$
5.	1110			A<0.
	+0011			
	0001	010		A+M

Remainder → A = $(0001)_2 = (1)_{10}$

Quotient → Q = $(010)_2 = (2)_{10}$.

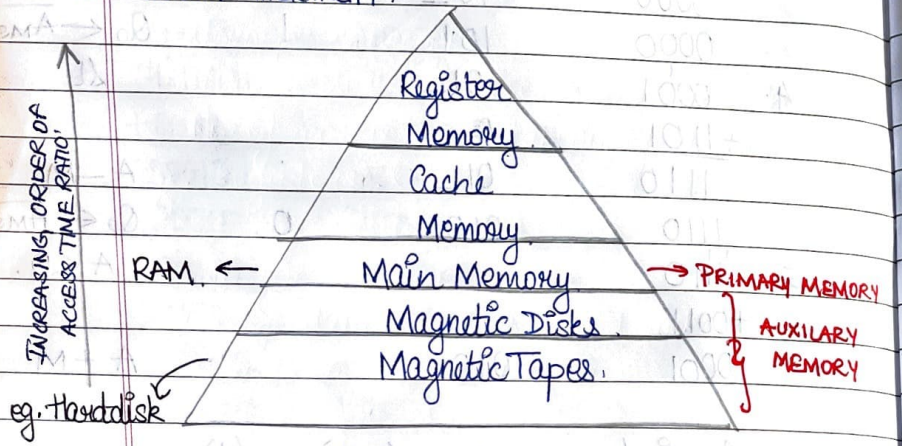
MEMORY ORGANIZATION.

Memory unit is collection of storage units or devices together.

VOLATILE MEMORY: loses data, when power is switched off.

NON-VOLATILE MEMORY: data is maintained, even after power is switched off.

* MEMORY HIERARCHY.



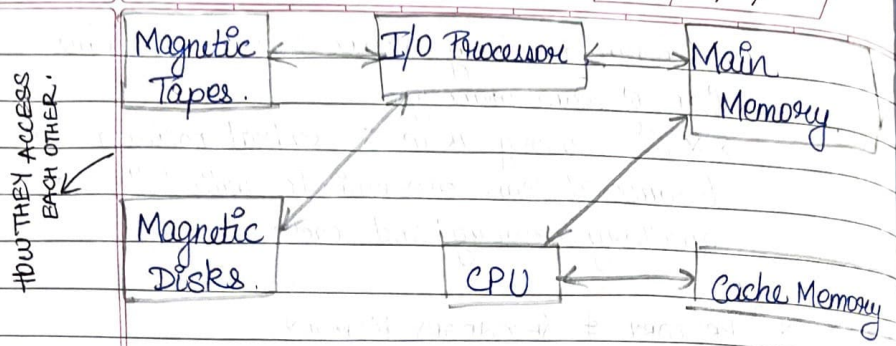
Cache Memory:
→ In processor.
→ Temporary storage to store ~~operation~~ operation results.
→ Helps in accessing.

Register Memory:
→ To store specific data.

→ Auxilary Memory access time is 1000 times that of main memory.
→ Main memory is in the central position because it can communicate with both the auxilary memory and cache memory.

* PRIMARY & SECONDARY MEMORY.

PRIMARY MEMORY	SECONDARY MEMORY
→ Main memory where the data & information are stored temporarily.	External memory where data is stored permanently.
→ Data is directly accessed by the processing unit.	Data cannot be directly accessed by processor.
→ Volatile memory.	Non-volatile memory
→ Stored in semiconductor chips which are expensive.	stored in external devices such as hard disks, flash drives, etc.
→ Categorized into cache memory and random access memory (RAM)	Permanent storage devices such as CD, DVD, HDD, floppy disk, etc.
→ Relatively faster than secondary memory due to volatile nature.	Slower than primary memory. It is like a back up memory.
→ Holds data or information that is currently being used by the processing unit.	Stores substantial amount of data and information ranging from gigabytes to terabytes.



* CHARACTERISTICS OF MEMORY.

1. LOCATION.
   CPU, Internal, External.

2. STORAGE CAPACITY
   → Word size → size of the location.
   No. of words → total no. of locations.
   eg. Registers. (of different types, limited no, each of 1 bit).

3. UNIT OF TRANSFER.
   Size of data transfered in 1 clock cycle.
   Depends on the bus size.

4. ACCESS METHOD.
   Random, Sequential, Direct

5. PERFORMANCE.
   (a) Access Time → Time b/w providing the add. & getting valid data from the memory.
   (b) Cycle Time → Time b/w 2 access, time required by memory ~~to~~ to recover before next access.

6. PHYSICAL MATERIAL
   Semiconductors, magnetic, optical (↗ CD's.)
   ↳ RAM ↳ AUXILARY MEMORY.

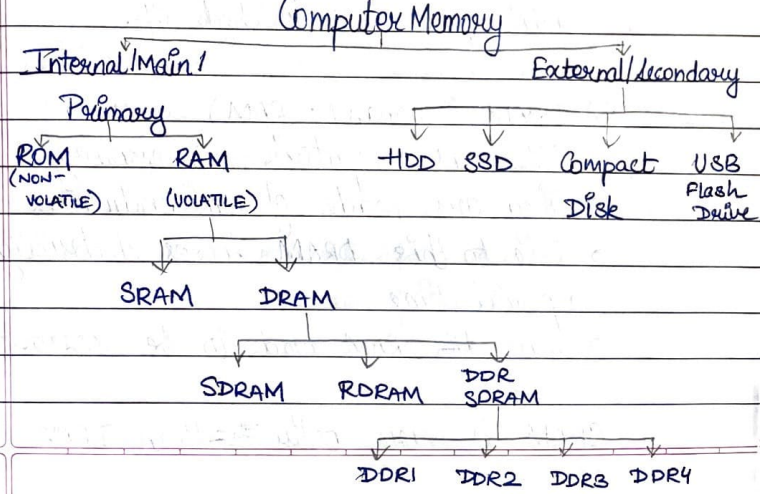
* MEMORY ACCESS METHOD.

1. RANDOM ACCESS.
   → Each memory has a unique address.
   → Using this unique address, memory can be ~~accessed~~ reached in the same amount of time in any order.

2. SEQUENTIAL ACCESS.
   → Allows memory access in a sequence or in order. (eg. arrays, linked list)

3. DIRECT ACCESS.
   → In this mode, information is stored in tracks, with each track having a seperate read/write head.

* TYPES OF MEMORY.



ROM → Read Only Memory.
→ It is non-volatile memory.
→ Used to store OS and data required when the computer is started or booted.

RAM → Random Access Memory.
→ After OS is loaded, RAM is used to temporarily store data for CPU to execute its tasks.
→ When more RAM is present, the CPU has to read lesser data from the external or secondary memory, allowing computer to run faster.
→ RAM is volatile, thus data present should be stored in storage device to avoid loss of data.

TYPES OF RAM.
1. SRAM (Static RAM)
→ Made of flip-flops.
→ The data stored in SRAM after power is switched on, the data remains in SRAM until power is switched off.

2. DRAM (Dynamic RAM)
→ Used as computer's main memory.
→ They are made of semiconductors.
→ Due to this DRAM looses electricity over a period time.
→ Thus DRAM needs to be researched.

SRAM is more costly than DRAM, it is used sparsely.
→ DRAM is used to make main memory.

TYPES OF DDRAM.
1. Synchronous DRAM (SDRAM).
→ Works on the rising edge of clock signal.
→ synchronizes the memory speed with the CPU clock speed.
→ It does not have a wait state.
→ Memories operate at the CPU-memory bus without imposing wait state.
→ Data transfer speed - 133 MgHz.

2. Double data rate SDRAM (DDR SDRAM).
→ Performs on both edges of clock signal.
→ Data transfer rate is doubled.

3. RAM based Solid state Drive (SSD-RAM / RD-RAM).
→ data is copied from volatile to non-volatile memory upon instruction or when the drive is powered down.
→ has its own battery, which helps when the computer crashes or powered down suddenly.
→ The battery keeps it alive for the time till data is transfered.

PAGE No.
DATE / /

TYPES OF ROM.

1. MROM (Masked ROM).
→ hard wired devices that contain a pre-programmed set of data or instructions.

2. PROM (Programmable ROM).
→ can only be modified once.
→ have small fuses which are burnt open during programming.
→ data once added can ~~be~~ not be ~~be~~ erased.

3. EPROM (Erasable Programmable ROM).
→ allows write & read & rewrite.
→ done by using UV rays.

4. EEPROM (Electrical Erasable Programmable ROM)
→ can be erased by exposing it to an electrical charge.

5. Flash ROM.
→ modern type of EEPROM.
→ non-volatile.
→ slower than RAM ~~or~~ but faster than hard disk.

* MEMORY ALLOCATION TECHNIQUES.

1. FIRST-FIT:
First sufficient block is allocated from the top of Main Memory.
Choosing is done while scanning.

* You cannot assign two tasks to a same block (which are of fixed sizes).

PAGE No.
DATE / /

All blocks are scanned first & then block is chosen based on condition

2. BEST-FIT:
Allocate the process to the partition which is the first smallest sufficient partition among the free available partition.

3. WORST FIT:
Allocate the process to the partition which is the largest sufficient among the freely available partition available in the memory.
→ Will be used if the programs/tasks have to be executed in the sequence that is given to the processor.

Q. Mem = 100, 500, 200, 300, 600
Proc. = 212, 417, 112, 426.

A. ① First Fit.

100	500	200	300	600
	212.	112		417

426 → X.

② Best fit

100	500	200	300	600
	417	112	212.	426.

③ Worst fit

100	500	200	300	600
	417		112.	212.

426 → X.

Efficiency = Total memory utilized / Total memory available.
(Memory Utilization)

① First Fit.

Efficiency = $\frac{741}{1700}$ = 0.435.

② Best Fit

Efficiency = $\frac{1167}{1700}$ = 0.686.

③ Worst Fit

Efficiency = $\frac{741}{1700}$ = 0.435.

Q. Mem = 200, 400, 600, 500, 300, 250.
Proc = 357, 210, 468, 491.

A. ① First Fit.

200	400	600	500	300	250
	357	210	468		

491 → X.

Efficiency = 0.46

② Best Fit

200	400	600	500	300	250
	357	491	468		210

Efficiency = 0.6782.

③ Worst fit.

200	400	600	500	300	250
		357	210		

488, 491 → X

Efficiency = 0.252.

When the memory locations are in a descending order & processes arrive in descending order of memory, worst fit will behave similar to best fit.

INTERNAL FRAGMENTATION → When memory is left after a task is allocated.

EXTERNAL FRAGMENTATION → When memory is not enough for a task to be allocated.

Fixed partition — Internal & External
Dynamic partition — External.

* VIRTUAL MEMORY.

If a program is active but it is waiting for execution activities, the program is shifted to the physical memory from RAM.
Thus, in this case, physical memory behaves as virtual memory.
This allows larger number of tasks to be run.
Entire main memory is, thus, not full and programs needing main memory can be allocated.

Due to shifting of programs, virtual memory is a little slower than virtual memory.

PAGE No.
DATE / /

* PAGING.

A function of memory management where a computer will store and retrieve data from a device's secondary storage to the primary storage.

Page Allocation Technique.
→ FIFO
→ LFU
→ LRU
→ Optimal.

* INTERLEAVED MEMORY.

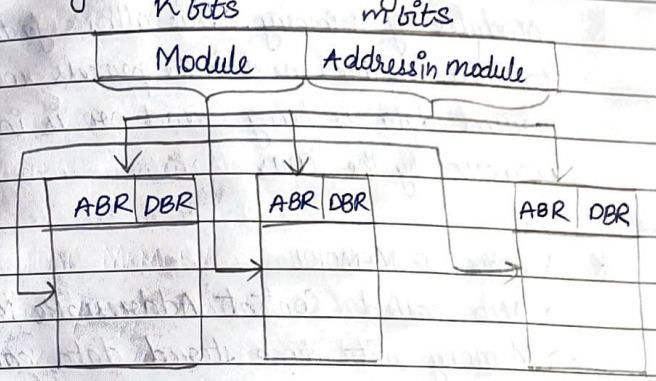
→ designed to compensate for the relatively slow speed of dynamic - random access memory (D-RAM) or core memory by spreading memory addresses evenly across memory banks.
→ Contagious memory read & write uses each ~~block~~ memory ~~bank~~, resulting in higher memory throughput.
→ Abstraction technique that divides memory into many modules, such that successive words in the address space are placed in different modules.

Example, Instead of placing the 10 elements of array in the same block; it will be seperated and placed evenly across all blocks. This reduces waiting time to execute instruction.

PAGE No.
DATE / /

② If we have 4 banks of 256 bytes, without interleaving, we will assign 0-255 to first bank, 256 - 511 to second bank.
But with interleaving, we will assign 0 to 1st, 1 to 2nd, 2 to 3rd, 3 to 4th, ~~and 4 to 1st~~ again and so on.

Types of Interleaved Memory.
→ High Order Interleaving



The MSB ~~decis~~ tells which block to refer. The LSB gives the memory address ~~in~~ of the data in the block.

One problem is that consecutive address tend to be in same block because the MSB decides which block to refer and LSB to refer address.

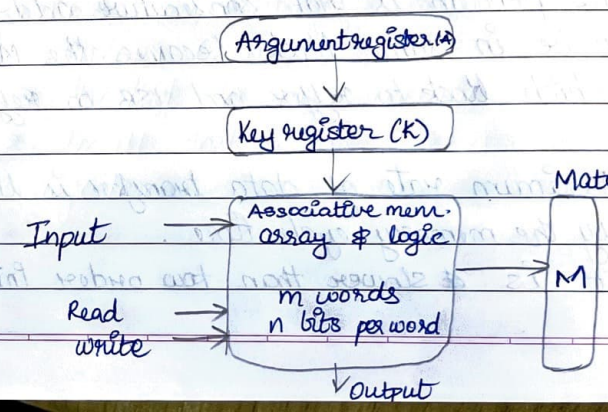
Maximum rate of data transfer is limited by the memory cycle time.
It is slower than low order interleaving

5M – Short note Interleaving
Associative Mem.

→ Low order interleaving
The ~~block~~ LSB decides the memory block.
This results in faster access time.

ADVANTAGES OF MEMORY INTERLEAVING.
→ Overall operations of the system (like arithematic operations) can be done faster.
→ It allows simultaneous access to different modules of memory. This allows faster access to data required next in the process while the current data is being read or is in a write process by the CPU.

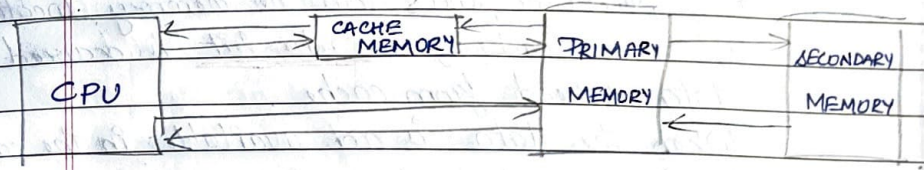
* ASSOCIATIVE MEMORY.
→ Also called Content Addressable Memory.
→ Memory unit whose stored data can be identified for access by the content of the data itself rather than by an address or memory location.
→ When write operation is performed, no address or memory location is given. The memory finds free memory and stores the info. (like First fit)



ADVATAGES OF ASSOCIATIVE MEMORY:
→ It decreases the search time.
→ It is used in parallel searching.

DISADVATAGES OF ASSOCIATIVE MEMORY.
→ It is more expensive RAM
→

* CACHE MEMORY.
→ extremely fast memory.
→ acts as a buffer between RAM and the CPU.



→ the information is stored in temporary location (special location which can easily be accessed) from its permanent location.
→ This done because this information may be required again or will be used frequently.
→ It will be as fast as Registers.
→ As it is located on the processor.

→ Cache memory is as fast as Registers as it is located on the CPU.
→ It is used to speed up & synchronize with high speed CPU. → If cache mem. is not present the CPU may require more time to fetch data. This time is decreased due to cache (as it holds data that may be required frequently).

nterleaving.
sociative Mem.

PAGE No.
DATE / /

nterleaving
decides the memory
faster access time.

MEMORY INTERLEAVING.

tions of the system (like
rations) can be done faster.

→ these consecutive memory address are in different memory modules.

→ Overall
arithmeti
→ It allow

→ Cache holds a copy of the data for address location of frequently used data from the memory.

* LEVELS OF MEMORY.
1. LEVEL 1 / REGISTER.
2. LEVEL 2 / CACHE MEMORY.
3. LEVEL 3 / MAIN MEMORY.
4. LEVEL 4 / SECONDARY MEMORY.

* BASIC OPERATIONS OF CACHE MEMORY.
→ First checks for a corresponding entry in the cache.
→ If processor finds that the memory location is in the cache, a cache hit is occured & data is read from cache.
When the data is not available in the cache, it is searched in the main memory (RAM) and then brought to the cache.
Then read/write process is done on the data.
This is called cache miss.

* TYPES OF CACHE.
1. PRIMARY CACHE.
→ located on the processor.
→ small and access time is comparable to registers.

* Explain LOR & its types.

2. SECONDARY CACHE.
→ placed between primary cache and other memory.
→ due to processor restrictions, secondary cache may not be present on the processor.

* LOCALITY OF REFERENCE.
→ deciding which task should be brought on the cache memory from RAM/main memory is known as Locality of Reference.

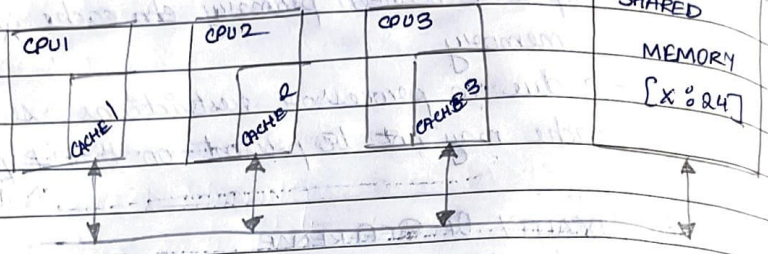
TYPES OF LOCALITY OF REFERENCE.
1. TEMPORAL LOCALITY OF REFERENCE.
→ Parts of the data which may be required repeatedly from the complete task, will be brought to the cache memory.

2. SPATIAL LOCALITY OF REFERENCE.
→ The data that is currently being executed and the it is highly possible that the data present in the consecutive locations will be accessed, and this data is not present in cache, then, this memory is brought to cache memory.
→ Thus, the data present near the current data is also brought to the cache memory.

* Explain the cache coherence problem.

PAGE No.
DATE / /

* CACHE COHERENCE (CACHE CONSISTENCY)



→ In a multiprocessor system, data inconsistency may occur among adjacent level or same levels of the memory.
→ If ~~the~~ each cache, if the same data is stored and ~~the~~ one processor update the value of the data, then this should be propagated ~~to the~~ throughout the other memories in a timely fashion.

* CACHE ORGANISATION

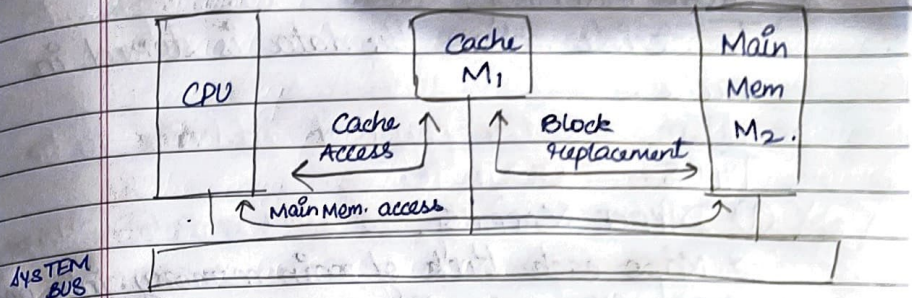
How cache memory is implemented.

2 ways.
→ Look aside cache design.
→ Look through cache design.

LOOK ASIDE CACHE DESIGN.

→ CPU initiates a read operation by placing the address on the address bus.
→ Immediately checks to see if the data is cache hit.

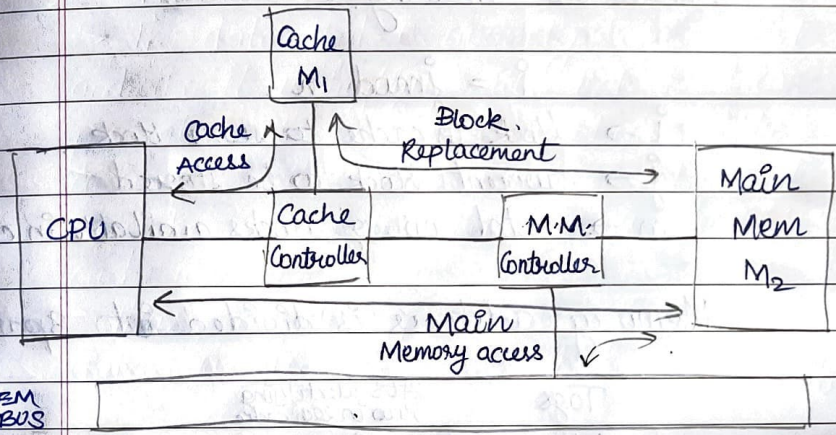
PAGE No.
DATE / /



Advantage.
→ Provides better access to the Main memory block.

Disadvantage.
→ If the system bus is being accessed for other uses, cache memory can not be accessed

LOOK THROUGH CACHE DESIGN.



→ Here, system bus is free for other units to communicate with main memory.
→ It is faster but expensive.

* CACHE MAPPING.

To decide where the data is stored in cache.

TYPES.

IMP ① DIRECT MAPPING.

→ Maps each block of main memory into only one possible cache line

→ Assign each memory block to a specific line in the cache.

→ If a line is already loaded when a new block is needed to be loaded, the old block is trashed.

→ Address space split.

index field | tag field.

Cache stores tag field, while main memory stores the remaining.

$i = j \bmod m$.

i → block in cache to store block.

j → current block to be stored

m → total no. of blocks available in cache.

Memory address is divided into 3 parts.

Tags	Bits identifying row in each cache	
t bits	l bit	w bits.
		Bits identifying word offset into block.

→ easy to implement

→ Not flexible as the formula remains constant for all the blocks, thus, for a certain set of blocks the same tag line will be chosen.

NUMERICALS ON DM

~~② FULL ASSOCIATIVE MAPPING.~~

Q. Main memory size 128 Bytes/words.
The number of words in each block is 4.
Total cache size = 16 Bytes/words.

A. Main memory.

Block 0
Block 1
//
Block 30
Block 31.

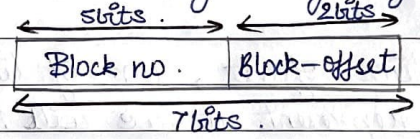
The number of words that can be stored is 4.

∴ The main memory will be divided into blocks each containing 4 words.

∴ No. of blocks = $\frac{128}{4}$

= 32.

No the physical address will be of 7 bits. as main memory is of size 128 bytes.



The no. of words is 4 so it only requires 2 bits to represent block-offset.

For this Q → 1. Description
2. Formula.
3. Example.

PAGE No.
DATE / /

Thus the remaining 5-bits represent block number.

Now, consider physical address to be
0001010.
Block No. ↑ 3rd Word in the given block.

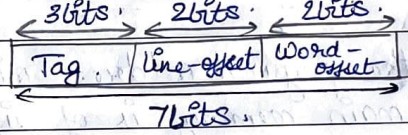
∴ This represents Word 10 from block 2.

Now, for cache memory, no. of words will be same.
So, the no. of lines in cache memory = 16/4
= 4.

No. of lines is 4.

Tag	Line 0
Tag	Line 1
Tag	Line 2
Tag	Line 3.

∴ Now, the ~~tt phy~~ address for cache memory will be of size 7 bits.



For line-offset, no. of lines present is 4.
∴ To represent it we will require 2 bits.

∴ Size of Tag = 7 − 4 = 3bits.

PAGE No.
DATE / /

Q. Consider a direct map cache of size of 32 KB with block size 32 Bytes. The CPU generates 32 bit address. Calculate no. of bits required for cache indexing & tag.

A. Total memory size = 32 KB.
$= 2^5 \times 2^{30}$
$= 2^{35}$ B. Omit

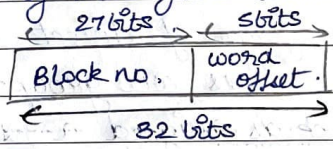
Physical address size = 32 bits.

No. of blocks in main memory = 32 bytes / 32 × 1KB

∴ Block offset = 5 bits.

Physical address is of size 32 bits.
Now, no. of words in each block is 32 bytes.

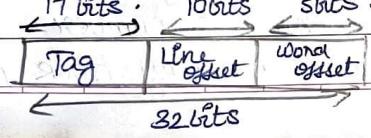
∴ No. of bits for word offset = 5bits.



Now, no. of blocks in cache = 32KB / 32B.
$= 2^{10}$.

∴ Line − offset is of 10 bits.

Cache address.



PAGE No.
DATE / /

② ASSOCIATIVE MAPPING.

→ associative memory is used to store content and address of the memory word
→ any block can go into any line of the cache
→ here line-offset is not present in cache address.
→ tag becomes all of the ~~rema~~ remaining bits leaving word offset
→ Thus, it is faster.
→ No. of cache hits is more than cache miss, as any block can go in any cache line.
→ Trashing is also minimized.
→ But, the time required to search for the word increases.

③ SET - ASSOCIATIVE MAPPING.

→ enhanced form of direct mapping.
→ here the lines are grouped into sets.
→ Each block is assigned a set.
→ In this set, the block can be assigned to any line.
→ Thus trashing decreases and time required to search is also less.

$$m = v * k.$$
$$i = j \bmod v.$$

i → cache set number.
j → main memory block number.

PAGE No.
DATE / /

v → no. of sets.
m → no. of lines in the cache.
k → no. of lines in each set.

Q. Explain min & max mode of 8086 processor

PAGE No.
DATE / /

INTEL 8086 ARCHITECTURE & ADDRESSING MODES.

* FEATURES.

→ 16 bit microprocessor.
→ 20 address lines & 16 data lines that provides upto 1MB storage.
→ provides operations like multiplication & division easily.
→ supports 2 modes of operation, i.e., Maximum mode & Minimum mode.
→ Maximum mode is suitable for system having multiple processors & minimum mode is suitable for system having a single processor.
→ It has an instruction queue, capable of storing 6 instructions bytes from the memory resulting in faster processing.
→ It has pipelining ~~-~~ when one process/task is executing, another process/task is kept in pipeline so that if the first process goes in waiting stage, in that time the next process can execute for some time.
→ 16-bit processor having 16 bit ALU, 16 bit registers, internal data bus & 16 bit external data bus.
internal / external ⇒ overall processing faster
→ 8086 : 5MHz.
8086-2 : 8MHz.
8086-1 : 10MHz.

PAGE No.
DATE / /

→ It uses two stages of pipelining, Fetch stage & Execute stage, which improves performance.
Fetch stage can pre-fetch up to 6 bytes of instructions & stores them in the queue.
Execute stage executes these instructions.
→ 256 vectored inputs.
→ 29000 transistors.

* ARCHITECTURE OF 8086.

PAGE No.
DATE / /

BIU → BUS INTERFACE UNIT
EU → EXECUTION UNIT.

BUS INTERFACE UNIT.
→ Unit which interacts ~~wh~~ with ~~t~~ data and address buses.

EXECUTION UNIT
→ All execution take place here.
→ Consists of ALU & flags.
→ gives instruction to BIU stating from where to fetch the data & then decoded & execute ~~t~~ instructions.
→ Has no direct connection to ~~BIU~~ system buses.

Flag registers are divided into 2 groups.
→ Conditional Flag.
→ Control Flag.
Flag registers change their state according to the results in accumulator.

FLAGS.
1] CONDITIONAL FLAGS.
Represents the result of the last arithmatic or logical instruction executed.

set → 1
reset → 0

→ CARRY FLAG
• Indicates an overflow condition.
1 → if Yes
0 → if No.

PAGE No.
DATE / /

→ AUXILARY FLAG.
• If there is a carry from lower nibble to higher nibble.
1 → if Yes 0 → if NO.

→ PARITY FLAG.
• Indicate the parity of the result.
1 → if even no. of 1's
0 → if odd no. of 1's.

→ ZERO FLAG.
• When accumulator ~~has~~ has 0 or any other no.
0 → set not 0 → reset.

→ SIGN FLAG
• holds the sign.
-ve → 1 +ve → 0.

→ OVERFLOW FLAG.
• When the system capacity is exceeded.