→ Direct Mapping:

Q: 64 words of main memory is mapped to a 16 word cache using direct mapping technique

Main memory size = 64 words
No. of bit used to access 1MM = $\log_2(64)$ = 6 MM bits
Block size = 4 words
No. of bits used as word instruction = $\log_2 4$ = 2 bits
PA of 1 data word = 6 bits
cache size = 16 word
No. of bits to be used to access 1 cache location = $\log_2 16$ = 4 bits
No. of lines in cache = $\frac{16}{4}$ = 4
No of blocks = $\frac{64}{4}$ = 16

2 bits: word identifier
4 bits: block identifier

Main memory

0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15
4	16	17	18	19
5	20	21	22	23
⋮		⋮		
15	60	61	62	63

Cache memory

	Block			
0	0	4	8	12
1	1	5	9	13
2	2	6	10	14
3	3	7	11	15

i: line number of cache
j: block number of MM
m: no. of lines in a cache

$i = j \% m$

Eg: word = 19

0	1	0	0	1	1

Block no = $\frac{19}{4}$ = 4 | word no = 3

Cache size = 16 words
Cache consists of no. of lines and line size = 4 words
No. of lines = $\frac{\text{cache size}}{\text{Line size}} = \frac{16}{4} = 4$

No. of bits used to identify each block: $\log_2 16 = 4$

No. of bits used to identify each line = r = $\log_2 4$

r = 00, line = 0
r = 01, line = 1
r = 10, line = 2
r = 11, line = 3

PA of data word = 4 bits

s bits	w bits

← Block identifier → ← word identifier →

The PA address is interpreted as word

s-r	r	w-bk

← Tag → ← line → ← identifier →

Block identifier | word

← Address length →

Address length = s+w
(MM size) Addressable unit = 2^{s+w}
Block = line size = 2^w words/bytes
No. of blocks in MM = $\frac{2^{s+w}}{2^w} = 2^s$
w = word identifier bits
s = block identifier bits
No. of lines in cache = m = 2^r
k = line identifier bits
Cache size = No. of lines × line size
= $2^r \times 2^w = 2^{r+w}$

tag size = s - r

Q: Find physical address of word 2 of block 10.

Sol: 1 1 1 1 1 0

PA = 62

Q: A 16 MB main memory is mapped into 64 kB cache memory using direct mapping function with block size = line size = 4 bytes

Sol: Address length = $\log_2^n = \log_2(16M)$

stw = $\log_2 2^4 \times 2^{20}$ = 24 bits

Block size = 4B

No. of blocks = $\frac{16\,MB}{4B}$ = 4M

No of bits used to identify 1 block =

w = 2 bits

s = 24 − 2 = 22 bits

Line size = 4B

cache size = 64 kB

No. of lines = $\frac{64\,kB}{4B} = \frac{16\,K}{} = 2^4 \times 2^{10} = 2^{14}$

Line identifier = r = 14 bits

Tag = s − r = 22 − 14

= 8 bits

a) Identify the tag number, line number and word number for the given memory address.

Sol: a) Memory address | Tag no | line no | Word no

FF0004H | FFH | 0001 | 0

1111 1111 | 0000 0000 0000 0100 | w-0

F F | 0 0 0 1 0

b) 160004H | 16H | 0001 | 0

0001 0110 | 0000 0000 0000 0100 | w-0

1 6 | 0 0 0 1 0

- Advantage of direct mapping: simple and inexpensive method
- Disadvantage: If processor keeps reference over data words that match to the same line number then the blocks continuously swap in the cache, which increase the miss penalty or decrease the hit ratio (thrashing).

The solution to this problem is victim cache.

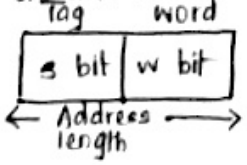
→ Victim cache: a) Based on the concept "to remember what was discarded in case it is needed again". Such recycling is possible by using a victim cache which is based on fully associative mapping technique.

b) It is a fully associative cache whose size is 4 to 16 lines.

→ Associative mapping:

a) It overcomes the disadvantage of direct mapping by permitting each main memory block to be loaded into any line of cache.

b) The physical address of dataword is interpreted as:



Address length = stw

No. of addressable unit (MM size) = 2^{stw}

Block size = line size = 2^w words / bytes

s = block identifier bits

w = word identifier bits

No. of cache lines = undefined

tags = s bits

No. of blocks = 2^s

Advantage:

Q: Find the tag & word number for the given address.

i). $16339C_H$ — Tag: $058CE7_H$, Word: 0

Sol: 16 3 3 9 C

0001 0110 0011 0011 1001 1100

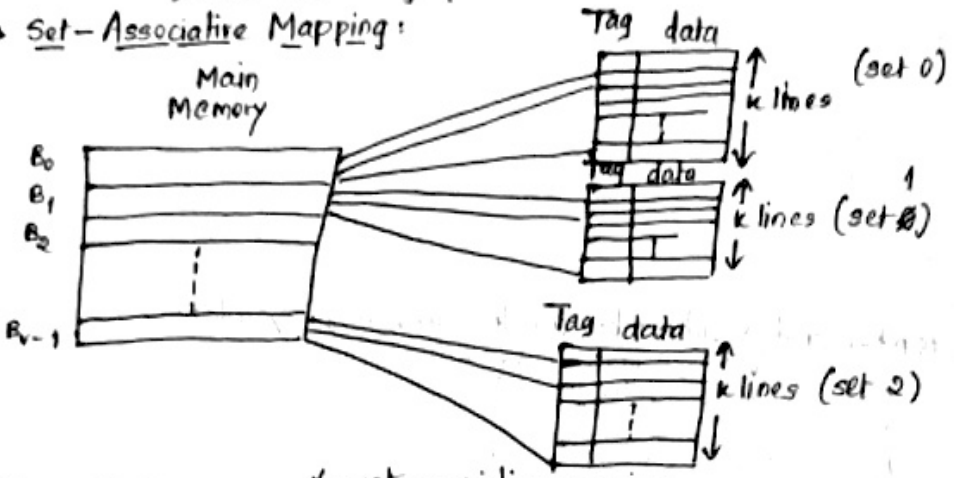
1 6 3 3 9 C

i). More flexible, as one main memory block can be placed into any cache line.

Disadvantage: i). It requires complex circuitry to examine the tags of all the lines in parallel

ii). Time consuming process

→ Set-Associative Mapping:



↓ - set associative mapping

Generalization:

i). Address length = (s+w) bytes

ii). No. of additional units (mm size) = 2^{s+w}

iii). Block size = Line size = 2^w words / bytes

iv). No. of blocks in MM = $\frac{2^{s+w}}{2^w} = 2^s$

v). s: block identifier bits

w: word identifier bits

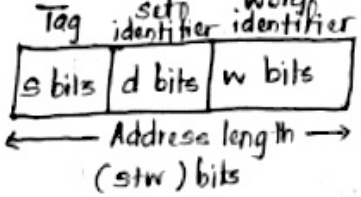
vi). No. of lines in set in cache = k lines

vii). No. of sets in cache = $v = 2^d$

d: bits used to identify each set

viii). No. of lines in cache = $k \times v = k \times 2^d$

PA is interpreted as:



ix). Size of cache = no. of lines × line size

$= k \times 2^d \times 2^w$

$= k \times 2^{w+d}$

. If v = no. of sets in cache

k = no. of lines in each set

total no. of lines: $m = k \times v$

. One MM block is mapped into any line of a specific (format) set using the formula

$i = j \% v$

where j = MM block

i = cache set number

. Set-associative mapping combines the strength of both direct and associative mapping by reducing their disadvantages.

Problems from ch-4:

4.1: No. of sets in cache = 4 = k = 2^d ⇒ d = 2 (No. of lines in each set)

Total no. of lines in cache : m = 64

$k = \frac{m}{4} = \frac{64}{4} = 16$ (No. of sets) $= 2^d$ ⇒ d = 4

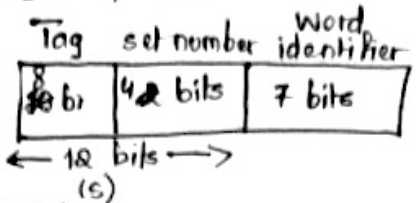
no. of blocks in MM = 4k = $2^2 \times 2^{10} = 2^{12}$ blocks

Block identifier (s) = 12 bits

Block size = line size = 128 words = 2^7 words

w = word identifier bits = 7 bits

PA interpretation:



4.3: Memory address: $111\,111_H$

Direct: 0001 0001 | 0001 0001 0001 00 | 01 — 01 — Tag: 11, line: 444, word: 1

tag: 11 | 0 4 4 4 | w

Associative: 0001 0001 0001 0001 0001 00 | 01 — Tag: 44444, word: 1

0 4 4 4 4 4 | w-1

set-associative: 00 01 0001 0001 | 0001 0001 0001 00 | 01 — Tag: 22, set: 444, word: 1

2 2 4 4 4 | w-01

Q: Suppose an 8 bit data word 11000010. Using the algorithm, generate the codeword.

Soln: m = 8 bit = 1 1 0 0 0 0 1 0
$D_8\ D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1$

According to inequality condition, no of check bits needed = 4 bits
k = 4 bits (C_8, C_4, C_2, C_1)
length of codeword = n+k = 8+4 = 12 bits

Code pos^n:	12	11	10	9	8	7	6	5	4	3	2	1
pos^n number:	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Check bit pos^n:					C_8				C_4		C_2	C_1
Data bits:	D_8	D_7	D_6	D_5		D_4	D_3	D_2		D_1		
	1	1	0	0	C_8	0	0	1	C_4	1	C_2	C_1

$C_1 = 3 \oplus 5 \oplus 7 \oplus 9 \oplus 11$
$= 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 = 0$

$C_2 = 3 \oplus 6 \oplus 7 \oplus 10 \oplus 11$
$= 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 = 1$

$C_4 = 5 \oplus 6 \oplus 7 \oplus 12$
$= 1 \oplus 0 \oplus 0 \oplus 1 = 0$

$C_8 = 9 \oplus 10 \oplus 11 \oplus 12$
$= 0 \oplus 0 \oplus 1 \oplus 1 = 0$

Codeword = 1 1 0 0 0 0 1 0 1 1 0

→ Code correction:

Q: Suppose the codeword is received 1100010. a) Determine whether error is occured:
b. Determine the position of error and also generate the correct codeword.

Soln: a) Received codeword: 1100010

codeword length = 7 bits
data bits, n = 4 bits
check bits = 7 − 4 = 3 bits (C_4, C_2, C_1)

1 1 0 0 0 1 0 — m = 1 1 0 0
$D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1$ — $k_{old} = [C_4\ C_2\ C_1]$
$C_4\ C_2\ C_1$ — $= [0\ 1\ 0]$

Determine the new check bits for the received data bits

$C_1 = 3 \oplus 5 \oplus 7$
$= 0 \oplus 0 \oplus 1 = 1$

$C_2 = 3 \oplus 6 \oplus 7 = 0 \oplus 1 \oplus 1 = 0$

$C_4 = 5 \oplus 6 \oplus 7 = 0 \oplus 1 \oplus 1 = 0$

$k_{new} = [C_4\ C_2\ C_1] = [0\ 0\ 1]$

Compute the syndrome by XORing k_{new} and k_{old}.

$S = k_{new} \oplus k_{old} = 001 \oplus 010$

011 ≠ 0, error has been detected.

Error position: decimal equivalent of syndrome (S) indicates the error position:
011 → 3rd position

Correct codeword: 1100110

i). If the syndrome contains all 0, no error has been detected.
ii). If syndrome contains one and only one bit 1, then error has been detected and detected in the parity/check bits. So no need to correct data bits.
iii). If syndrome contains more than one bit set to 1, then the numerical value/decimal value of the syndrome indicates the position of data bit error. This data bit is inverted for correction.

Q. For the 8 bit word 00111001, check bits stored : 0111. Suppose, when the word is read from the memory, the check bits calculated to be 1101. What is the data word that was read from the memory?

Soln: k_{old} = 0111 S = 0111 ⊕ 1101 = 1010 ≠ 0, error has been detected

k_{new} = 1101

position of error : 10^{th} pos

m = 00111001 → 00011001

Q: How many check bits are needed for the hamming error code is used in a 1024-bit data word?

Soln: m = 1024 bit

Using inequality condition: $2^k - 1 \geq m + k$

If k = 10: LHS : 1023

RHS : 1024 + 10 = 1034

LHS ≱ RHS

If k = 11: LHS : $2^{11} - 1 = 2047$

m + k = 1024 + 11 = 1035

⇒ LHS ≥ RHS

Q: a) Develop an SEC (Single error correcting) code for 16 bit data word. Generate the code for the dataword 0101000000111001.

b) Show that the code will correctly identify an error in data bit 5.

CH: EXTERNAL MEMORY

→ RAID : Redundant array of independent disks.

i). To increase the overall performance of the system, we need to improve the performance of another component, i.e disks.

ii). The term RAID was originally coined in a paper by a group of researchers at the University of California in Berkeley.

iii). Consists of 7 levels : RAID 0 – RAID 6

iv). Levels designate different hierarchial characteristics which are shared by all the levels.

• Level 0 (Stripping) :

i). Divides data into block units and writes them in a dispersed manner across all disks.

ii) As data is placed on every disk, so it is called striping.

iii). Reliability : 0 There is no duplication of data. Hence a block once lost cannot be recovered.

iv) capacity : N * B. The entire space is being used to store data. Since there is no duplication, N disks each having B blocks are fully utilized.

Advantage : a). Easy to implement
b). Utilizes the storage capacity in a better way.

Disadvantage : a). A single drive loss can result in the complete failure of the system.
b). Not a good choice for a critical system.

. Level 1 (Mirroring) :

i). Redundancy is achieved by the simple expedient of duplicating all the data.

ii). Data stripping is used but each logical strip is mapped to two separate physical disks so that every disk in the array has a mirror disk that contains the same data.

iii). RAID 1 can also be implemented without data striping.

iv). Reliability : 1 to N/2. 1 disk failure can be handled for certain because blocks of that disk would have duplicates on some other disk.

v). Capacity : N * B/2 . Only half the space is being used to store data. The other half is just a mirror of the already stored data.

Advantage : a). It covers complete redundancy
b). It can increase data security and speed.

Disadvantage : a). Highly expensive
b). Storage capacity is loss.

. Level 2 (Bit-Level stripping with Dedicated Parity):

i). This uses bit level stripping (instead of striping the blocks across the disks, it strips the bits across the disks).

ii). You need two groups of disks. One group of disks are used to write the data, another group is used to write the error correction codes.

iii). This uses Hamming error correction code (ECC) and stores this information in the redundancy disks.

iv). When data is written to the disks, it calculates the ECC code for the data on the fly, and stripes the data bits to the data-disks and writes the ECC code to the redundancy disks.

v). When data is read from the disks, it also reads the corresponding ECC code from the redundancy disks, and checks whether the data is consistent. If required, it makes appropriate corrections on the fly.

Advantages : a). In case of error correction, it uses hamming code.
b). It uses one designated drive to store parity.

Disadvantages : a). It has a complex structure and high cost due to extra drive.
b). It requires an extra drive for error correction.

• Level 3 (Byte-Level Stripping with Dedicated Parity):

i) This uses byte level stripping (instead of striping the blocks across the disks, it strips the bytes across the disks).

ii). Uses multiple data disks, and a dedicated disk to store parity.

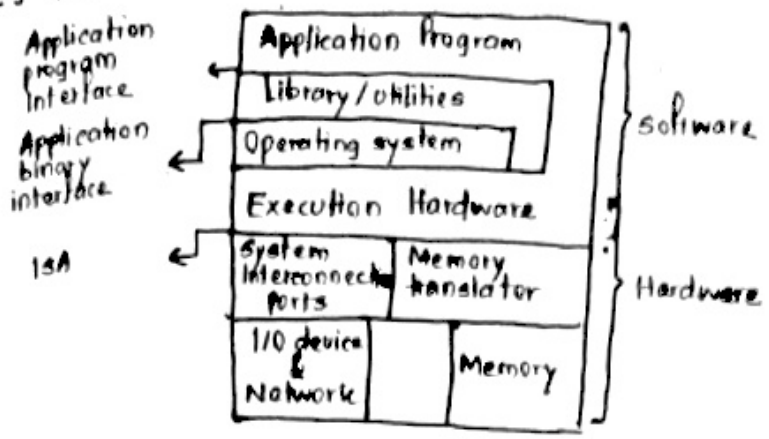
Advantages: a). Data can be transferred in bulk.
b). Data can be accessed in parallel.

Disadvantages: a). It requires an additional drive for parity.
b). In case of small-size files, it performs slowly.

• Level 4 (Block-Level Stripping with Dedicated Parity):

CH: OPERATING SYSTEM SUPPORT

→ Operating system Overview:



→ Operating system objectives & Function:

. OS is a program that controls the execution of the Application program.
. OS is acting as interface between hardware and software.

→ Two objectives of OS:

i). Convenience: To make the system more convenient to use.
ii). Efficient: OS allows the computer system resources to be used in efficient manner.

→ Two aspects of OS:

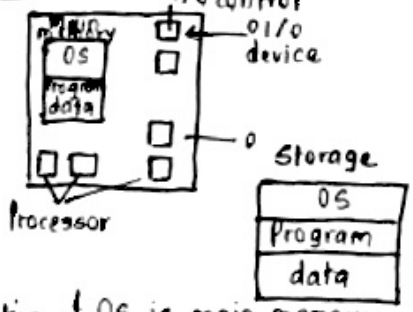
i). OS as a user/computer Interface:

. The hardware/software used in an application to a user can be viewed as layered/hierarchial manner.
. To develop the application program a set of data & program codes are used as utility.

Services provided by OS:

ii). Program creator
iii). Program execution
iv). Access to file system
v). Access to I/O devices
vi). System access
vii). Error detection & correction
viii). Accounting

ii). OS as a resource manager:



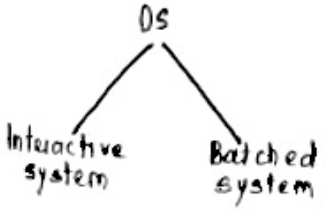
. A portion of OS is main memory.
. This portion includes the kernel/nucleus of OS.
. The remaining portion contains program & data.

Summary of functions of OS:

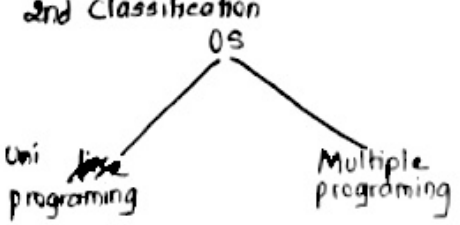
. Process management
. Device management
. Memory management
. File System management
. User Interface
. Error detection aids
. Job accounting
. Network management
. Security & protection

→ Classification of OS:

1st classification



2nd classification



Interactive system: User/programmer interact directly with the computer by giving request for the execution of a program.

Batched system: i). Opposite to interactive system
ii). User programs is batched together with programs from other users and submitted by computer operator.
e.g: resident monitor

Uni programing: System runs on only one program at a time.

Multiple programming: Attempt is made to keep the processor busy by executing multiple programs at a time.

CH: COMPUTER ARITHMETIC

→ Multiplication of unsigned binary integers:

Q: 9X3 ⇒ 1001 } n-bit
x 0011

1001
1001 } partial product
0000
0000

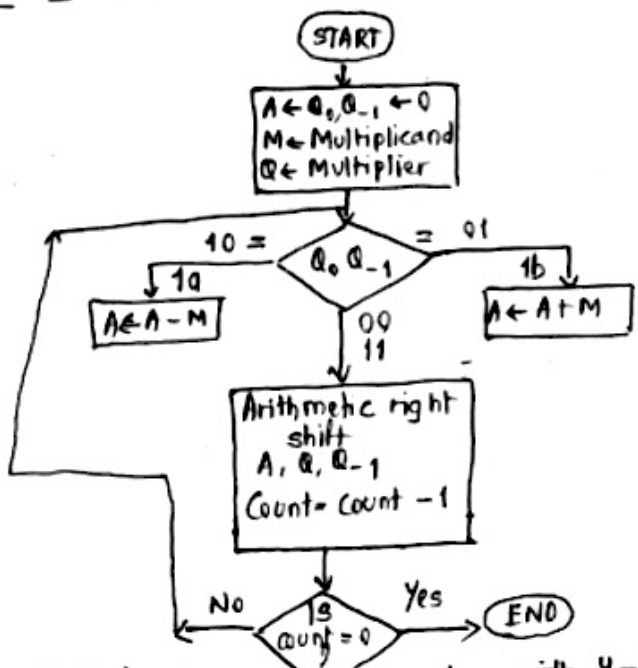
00011011

product = 2n bits

Problems:
i). Straightforward multiplication will not work when multiplicand / multiplier is negative.
ii). Each contribution of the negation multiplicand as a partial product must be negative number on a 2n product.
iii). If they are treated as unsigned number, then multiplication (9X3 = 27) produces correct result. But 1001 is treated as the complement of (7), then the multiplication produce wrong result.

Solution:
i). A number of way out of dilemma.
ii) One way is convert both the numbers as positive numbers, perform multiplication & then carry negative two & complement of the result of signed both the numbers are different.
iii). One most common method is Booth's algorithm.
iv). Benefit of speeding up the multiplication process relative to straightforward method.

→ Flowchart of Booth's algorithm:



A = 0000 (4 bit
Q : 4bit
Q_{-1} : 1 bit
M : 4 bit

Q: Perform 7 x (-3) using Booth's algorithm with 4-bit binary integer.

7 = 0111
-3 = x 1101

Sol^n:

	Count	step	A	Q	Q_{-1}	M	$Q_0 Q_{-1}$	
C=4		Initialize	0000	1101	0	0111	10	
		1a	1001	1101	0	0111		→ -0111
C=3		2	1100	1110	1	0111	01	
C=2		1b	0011	1110	1	0111		
		2	0001	1111	0	0111	10	→ -0111
		1a	1010	1111	0	0111		
		2	1101	0111	1	0111	11	
		2	1110	0111	1	0111		

END: -(00010101)

Q: Perform 7x3 using Booth's algorithm with 4-bit binary integer.

Soln:

Count	Step	A	Q	Q_{-1}	M	$Q_0 Q_{-1}$
C=4	Initialization	0000	0011	0	0111	10
C=3	1a	1001	0011	0	0111	11
	2	1000	1001	1	0111	
C=2	2	1110	0100	1	0111	01
C=1	1b	0101	0100	1	0111	00
	2	0010	1010	0	0111	
C=0	2	0001	0101	0	0111	

END : $-(00010101)$

→ Floating-point Number Representation:

. With a fixed notation (2s complement) it is possible to represent both positive and negative numbers centered near 1 or 0.

. By assuming a fixed binary/radix point, the format allows the representation of the numbers with fractional components as well.

limitation:

- Very large numbers cant be represented.
- Very small numbers cant be represented.

In general, floating point numbers can be represented as

$$\pm s * B^{\pm E} \quad \text{or} \quad (-1)^S * s * B^{\pm E}$$

where S = sign of the number

0 → +ve

1 → -ve

s = significant E = +ve/-ve (exponent of base b)

Q: 1.5×10^5

Soln: S = 0, s = 1.5

B = 10, exponent = 5

→ IEEE standard for binary/floating point number:

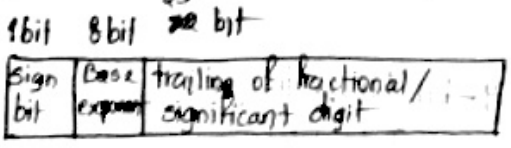
. The floating point representation is defined in IEEE standard 754, adopted in 1985 and revised in 2008.

Three types of floating point format defined by IEEE standard 754:

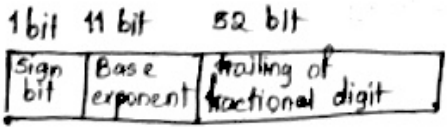
i). Arithmetic format: a). Uses a 32-bit format.
b). Most of the operands/results are defined using this format.

ii). Base format: a). Uses a 64 bit format for representation.
b). The format covers 5 fp representation: 3 for binary and 2 for decimal whose encoding are specified in this format.

iii). Interchange format: Uses 128 bit for representation.



32 bit (single precision format)



64 bit (double precision format)

Q: Convert the following numbers in IEEE standard 754 floating point format ?
a) -1.5 b) 0.384 c) +(1/32)
d) +(1/4)

Soln: a) i). sign bit = 1 (represent a -ve number)

ii). Binary representation : 1.5 = $(1.1)_2$

iii). Represent in scientific notation

$(-1)^S * (1+F) * B^{\pm E}$

So $(-1) * 1.1 * 2^0$

iv). Normalized scientific notation representation

e.g: a). $0.011 = 1.1 \times 2^{-2}$

b). $11101 = 1.1101 \times 2^4$

v). Determine the biased exponent (a fixed value is subtracted from the biased exponent to get the true exponent;
i.e true exponent = Biased exponent - 127)

$0 =$ Biased exponent $- 127$

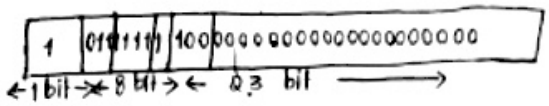
$\Rightarrow$ Biased exponent $= (127)_{10}$

Represent biased exponent in 8 bit $= (01111111)_2$

vi). Determine the significant bit by removing the leading 1 from the mantissa.

$(-1)^1 * 1.1 * 2^0$

Significant field = 10000000000000000000000



$= (BFC00000)_H$

b. 0.384

sign bit = 0

Binary representation:
$0.384 \times 2 = 0.768$
$0.768 \times 2 = 1.536$ $\rightarrow (0.384) \rightarrow (0.011000)_2$
$0.536 \times 2 = 1.072$
$0.072 \times 2 = 0.144$
$0.144 \times 2 = 0.288$
$0.288 \times 2 = 0.576$

Scientific notation: $(-1)^0 * 0.011000 * 2^0$

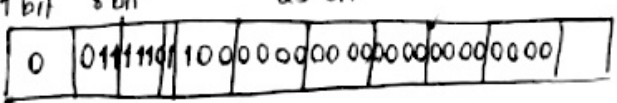
Normalized scientific notation: $(-1)^0 * 1.1000 * (2)^{-2}$

Biased exponent: $-2 =$ Biased exponent $- 127$

Biased exponent $= 125$

Biased exponent in 8 bit: $(01111101)_2$

Significant value : 10000000000000000000000



$= (3EC0000)_H$

Q: The following number uses the IEEE 32 bit floating point format. find out its equivalent decimal value. 11000001111000000000000000000000

Soln: i) identify the 3 field from the given 32 bit format

1 | 10000011 | 11000000000000000000000

1 bit | 8 bit | 23 bit

Determine value of sign bit, bias exponent and true exponent

ii) s = 1 (sign bit)

biased exponent $= (10000011)_2 \rightarrow (131)_{10}$

true exponent $= 131 - 127 = 4$

iii). Substitute these values in normalized scientific notation to get the decimal equivalent

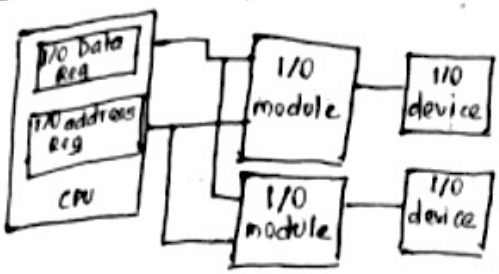
$(-1)^S * (1+F) * B^{\text{exponent}}$

$= (-1)^1 * (1+0.11) * 2^4 = -1.11 \times 2^4$

$= -(11100)_2 = -28$

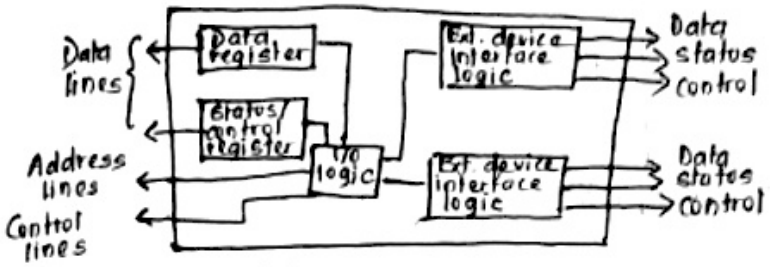
CH: INPUT & OUTPUT

→ Introduction:



→ Why peripherals/external devices are not connected directly to the processor:
i). Data transfer rates are mismatched.
ii). Uses different data formats and word length.

→ Structure of I/O module:



Major Functioning of I/O modules:
i). Interface to processor and memory via system bus.
ii). Interface to one/more peripheral devices.

Classification of I/O module:
. I/O modules are classified into three categories:
i). Human Readable
ii). Machine readable
iii). Communication
. The interface in I/O module is in the form of those signals:
i). Control
ii). Data
iii). status
. The major functions of the I/O module fall into following categories:
i). Control timing
ii). Processor communication
iii). Device communication
iv). Data buffering
. In addition to the processor and set of memory element, the third key element of computer system is a set of I/O modules. Each module interface to the system bus and controls one or more peripheral devices.
. Human Readable: suitable for communicating with computer user.
e.g: video display terminal, monitor, etc.
. Machine Readable: suitable for communicating with equipment.
e.g: magnetic disk, tape, sensors & accuators
. Communication: suitable for communicating with remote devices.
e.g: Human readable device such as terminal and Machine readable device / another computer

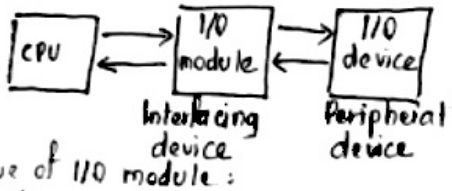
- Control signals: Determine the function the device will perform such as
  i). send data to I/O module (processor → I/O device)
  ii). accept data from I/O module
  iii). Report status
- Data: The data are in the form of a set of bits to be sent/received from I/O module.
- Status signals: Indicate state of the device
  e.g: READY / NOT READY
  To show whether the device is ready to accept the data or not.

→ Function:
i). Control & timing: Coordinates flow of traffic b/w internal resources and external devices
ii). Processor communication: Involves command decoding, status reporting, data & address recognition
iii). Device communication: Involves status information and data
iv). Data buffering: perform the needed buffering operation to balance device & memory speed.

→ I/O Technique:

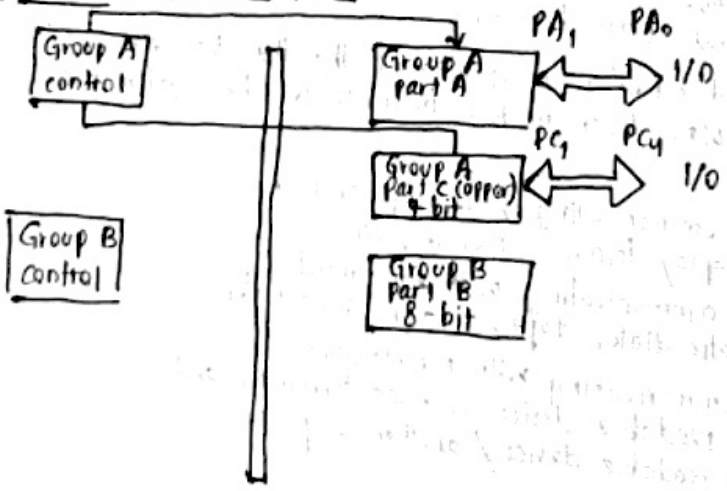
	No interrupt	Use of interrupt
I/O to memory transfer through processor	Programmable I/O	Interrupt driven I/O
I/O to memory transfer without processor.		DMA

- Three techniques are possible with I/O operations:
  i). programmed I/O
  ii). Interrupt driven I/O
  iii). Driver memory access



- Addressing technique of I/O module:
→ memory mapped I/O
→ I/O mapped I/O
- I/O commands: i). Control
  ii). test
  iii). RD
  iv). WR

→ 8255A Programmable peripheral device:



- $A_1 A_0$: Combination of these two pins determine which internal register of 8255A data are written into or read from.

A_1	A_0	
0	0	→ port A
0	1	→ port B
1	0	→ port C
1	1	→ control word register (CWR)

- Chip select: It is an active low input pin which is used to select the chip (8255A) before any data transfer.
- $\overline{RD}$, $\overline{WR}$: These are active low input. Data is written into 8255A when $\overline{WR}$ is 0. Data can be read from the 8255A when $\overline{RD}$ is 0.
- Reset: 8255A is placed into reset if this input line is active high (1).
- D0 - D7: Data I/O lines for the device. All the information read from or written into the 8255A via these data lines.
- CWR: The content of the CWR is control word. Control word is a 8-bit data format which determines the port A, port B and port C are operating as input/output ports and also determines the mode of operation of each port.

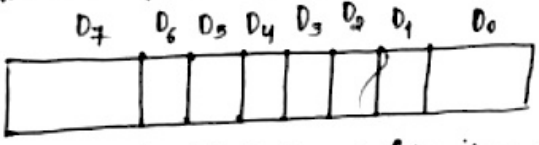
→ Mode of operation of 8255A:

i). Mode 0: Applicable to all ports PA, PB and PC acting as input port or output port.

ii). Mode 1: Applicable to ports P_A and P_B where one of them acts as input or output port. P_C acts as handshaking signals for P_A and P_B.
- P_{C0}, P_{C1}, P_{C2} : handshaking signals for port A
- P_{C3}, P_{C4}, P_{C5} : handshaking signals for port B
- P_{C6}, P_{C7} : Input port / output port

iii). Mode 2: Only applicable to port A where it acts as bidirectional (both i/p and o/p). Port C is acting as handshaking signals for port A.
- P_{C0}, P_{C1}, P_{C2} : handshaking signals for port A when port A is acting as output port
- P_{C3}, P_{C4}, P_{C5} : handshaking signals when port A acting as input port.
- P_{C6}, P_{C7} : Input / output port
- Port B : Input / output port (don't care)

→ Control Word:



i). D_0 meant for P_C (lower) as input/output port (1 - i/p port, 0 - o/p port)

ii). D_1 : P_B as input/output port

iii). D_2 : Mode of operation of port B
- D_2 - 0 indicates mode 0 of port B
- D_2 - 1 indicates mode 1 of port B

iv). D_3 : P_C upper as input/output port

v). D_4 : P_A as input/output port

vi). D_5, D_6 : mode of operation of port A
- 00 - mode 0 of port A
- 01 - mode 1 of port A
- 1X - dont care

vii). D_7 : D_7 = 1 indicates simple I/O mode
D_7 = 0, BSR mode