Chapter 9: Memory Organization

↳ multiplication algorithms (Booth), Division algorithms
↳ Memory Hierarchy, main memory, auxillary memory
↳ Associative memory, cache memory

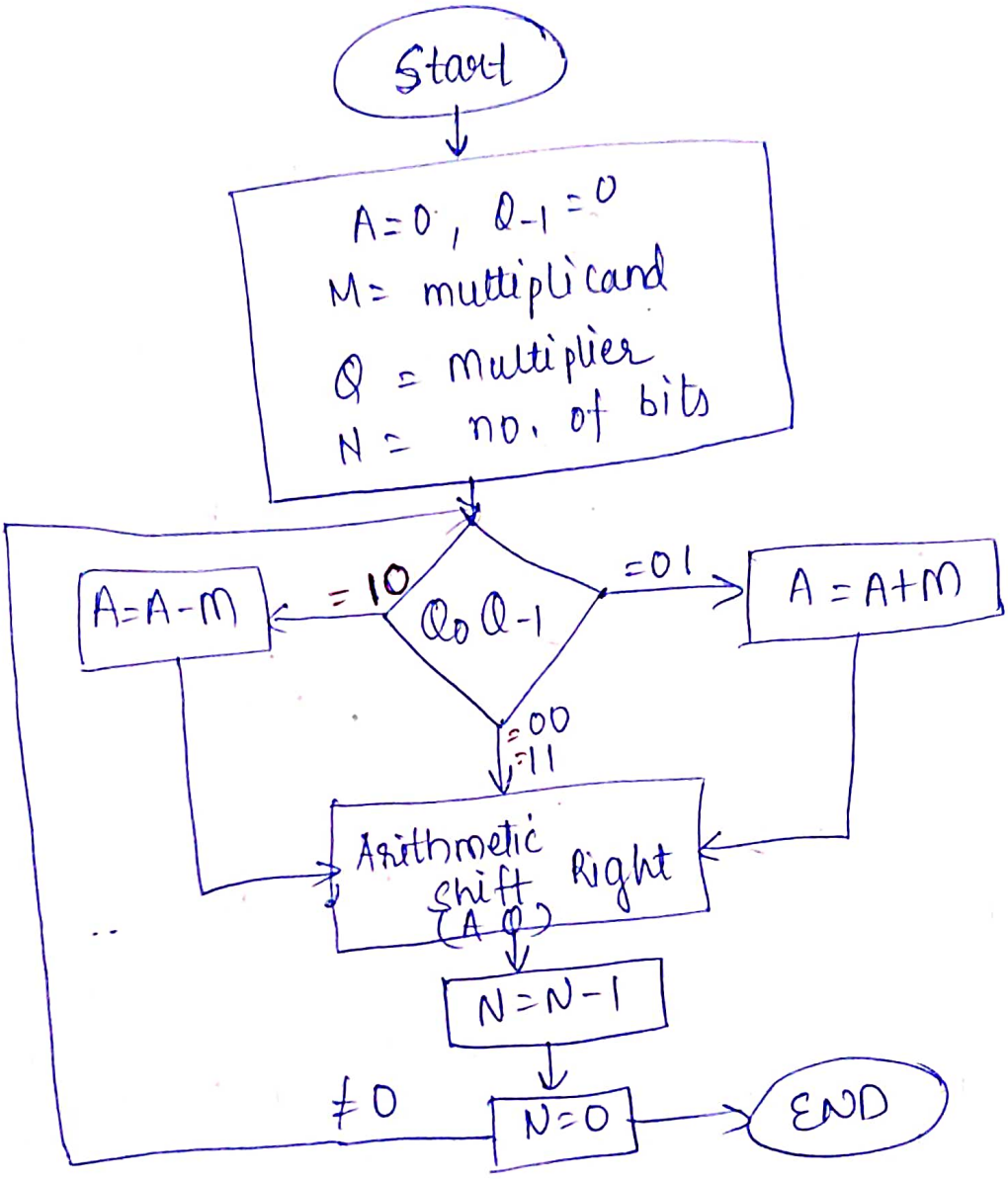


fig. Booth Algorithm for Multiplication

Multiplication of 2 signed nos.

Q. Multiply -7 and $+3$, using booth's algo. (register bits = 5)

$m = (-7)_{10}$

$m \rightarrow 00111 \rightarrow (11001)_2$

$Q \rightarrow (3)_{10} \rightarrow (00011)_2$

$(-m) \rightarrow (00111)_2$

AC	Q	Q_0	Q_{-1}	Operation
00000	0001	1	0	i) AC = AC − M = 00000 + 00111 = 00111
00111	00011	1	0	ii) ASR
00011	10001		1	
				i) ASR
00001	11000		1	
				i) AC = AC + M = 00001 + 11001 = 11010
11010	11000		1	ii) ASR
11101	01100		0	
				i) ASR
11110	10110		0	
				i) ASR
11111	01011		0	

$(11111\ 01011)_2$

2's comp. $-0000010101 = (-21)_{10}$

Q. (-ve) × (-ve) = (+ve)

multiply $(-7)_{10}$ and $(-3)_{10}$ using Booth's algorithm.

register size = 4 bits.

$M \rightarrow (-7)_{10} \rightarrow (1001)_2$

$-M \rightarrow (7)_{10} \rightarrow (0111)_2$

$Q \rightarrow (-3)_{10} \rightarrow (1101)_2$

AC	Q	Q_0	Q_{-1}	Operation
0000	1101	1	0	i) AC = AC − M = 0000 + 0111 0111
0111	1101	0		
0011	1110	1		ii) ASR
1100	1110	1		i) AC = AC + M = 0011 + 1001 1100
1110	0111	0		ii) ASR
0101	0111	0		(i) AC = AC − M = 1110 0111 1) 0101
0010	1011	1		ii) ASR
0001	0101	1		i) ASR

$(0001\ 0101)_2 = (21)_{10}$

Examples

Q.1 multiply $(-9)_{10}$ and $(-13)_{10}$ using booth's algorithm. Show all steps. (Register size = 5 bits)

Q.2 Multiply $(-2)_{10}$ and $(-3)_{10}$ using booth's algorithm. Show all steps. (Register size = 4 bits)

Q.3 $(9)_{10} * (-13)_{10}$

Q.4 $(-9)_{10} * (13)_{10}$

} Reg. size = 5 bits

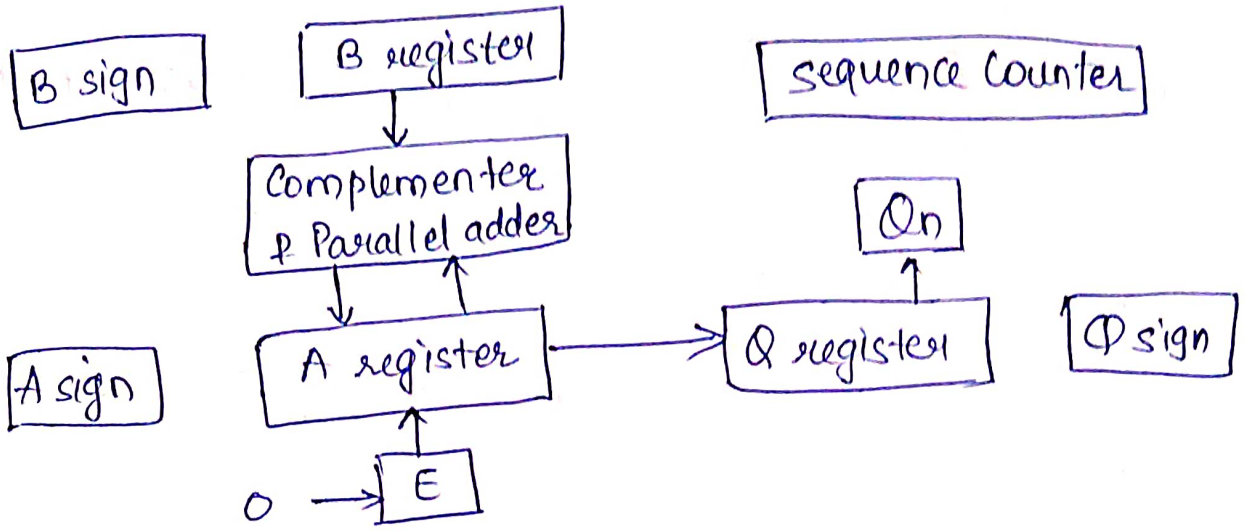
Q.5 $(9)_{10} * (-7)_{10}$

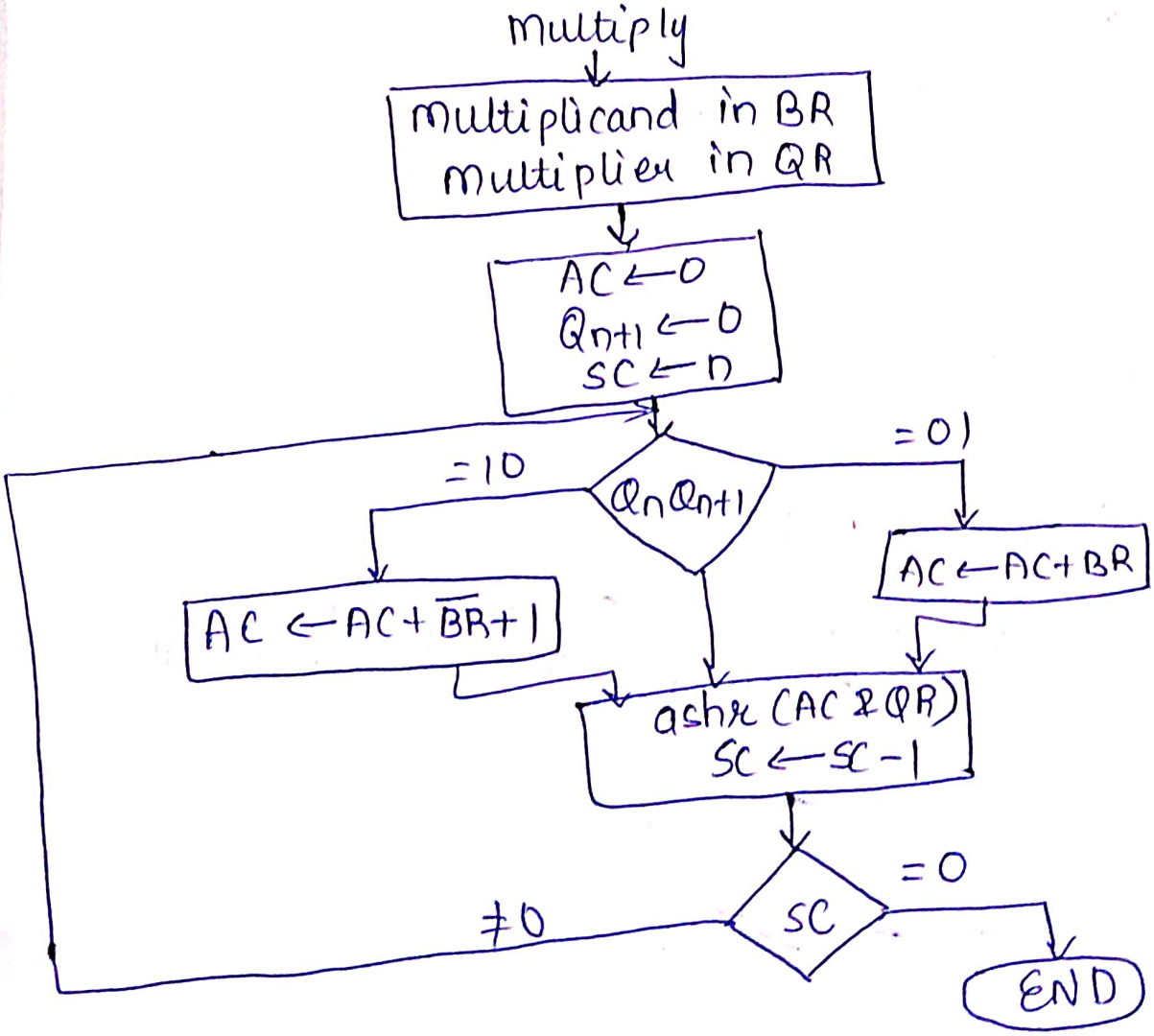
Q.6 $(-5)_{10} * (4)_{10}$

} Reg. size = 4 bits

Chapter 9 :- MULTIPLICATION ALGORITHMS

- (Booth multiplication Algorithm)





eg. $7 \times 3 = 21 = 10101$

M = 0111 Q = 0011

$Q_3 Q_2 Q_1 Q_0 \mid Q_{-1}$

1)

A	Q	Q-1	
0000	0011	0	
1001	0011	0	
1100	1001	1	shr →

A − m ⇒ A + 2's comp.(M)

$$\begin{array}{r} 0000 \\ +\ 1001 \\ \hline 1001 \end{array}$$

2) 1100 1001 1 shift right

1110 0100 1

3) A = A+m

$$\begin{array}{r} 1110 \\ +\ 0111 \\ \hline 0101 \end{array}$$

0101 0100 1

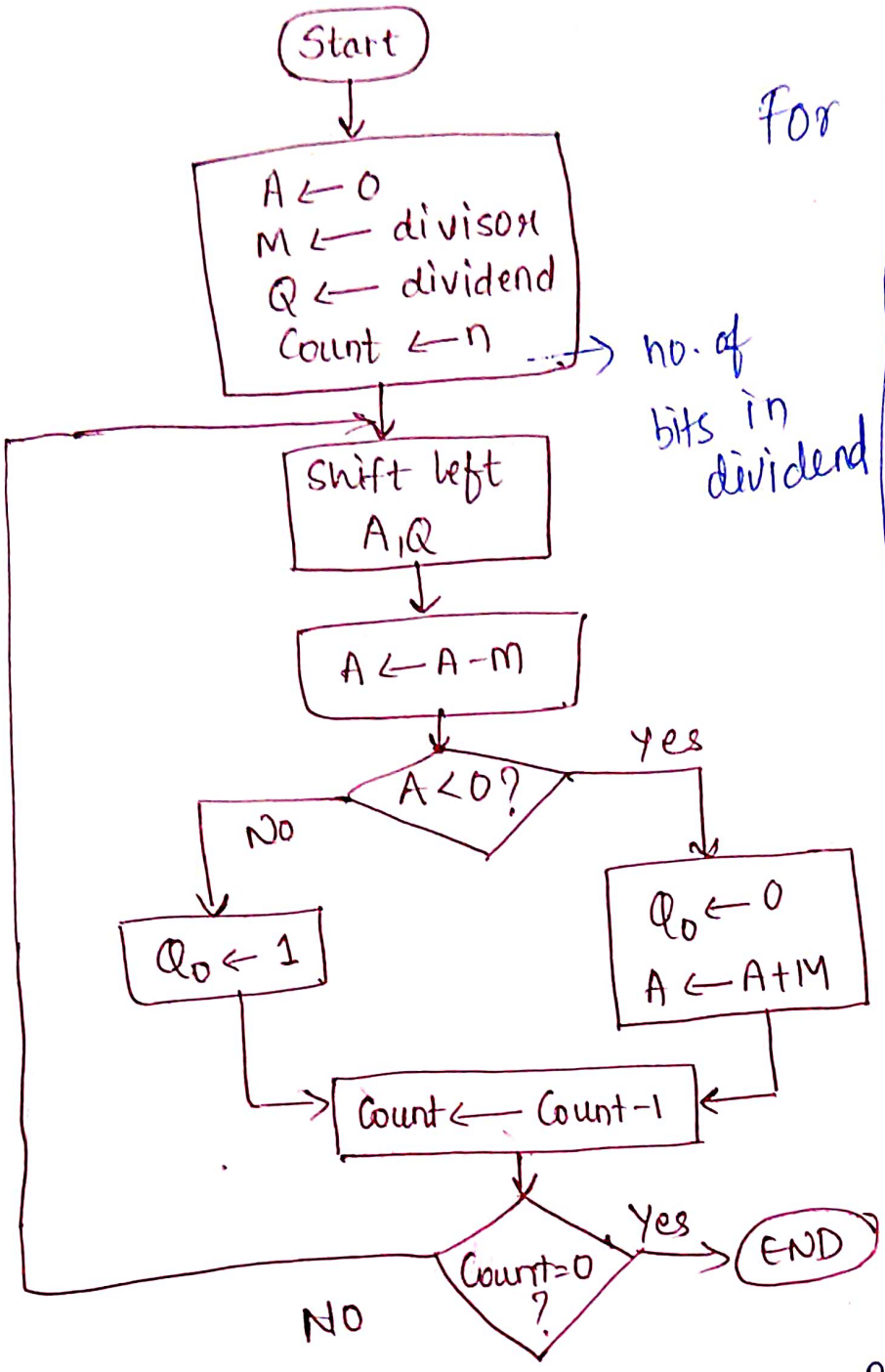
0010 1010 0 shr →

4) shift right

0001 0101 0

Division example :-

A	Q	m	Count	Operation
0000	0111	0011	4	initialization
0000				



For Unsigned nos. only.

Q → quotient
A → Remainder
end of algo.

Quotient in Q
Remainder in A

Booth's Restoring Division Algo.

(2) eg. Divide 7 by 3

Divisor = 0011
Dividend = 0111

(1) eg. Dividend = 11
Divisor = 3

(1)

N	M	A	Q	Operation
4	00011	00000	1011	initialize
	00011	00001	011_	shift left AQ
	00011	11110	011_	A = A-M
	00011	00001	0110	Q[0] = 0 and restore A
3	00011	00010	110_	shift left AQ
	00011	11111	110_	A = A-M
	00011	00010	1100	Q[0] = 0 restore A
2	00011	00101	100_	shift left AQ
	00011	00010	100_	A = A-M
	00011	00010	1001	Q[0] = 1
1	00011	00101	001_	shift left AQ
	00011	00010	001_	A = A-M
	00011	00010	0011	Q[0] = 1

(2)

N	m	A	Q	Operation
4	0011	0000	0111	initialize
	0011	0000	111_	shift left AQ.
	0011	1101	111_	A = A-m
	0011	0000	1110	Q[0] = 0 and restore
3	0011	0001	110_	shift left AQ
	0011	1110	110_	A = A-m
	0011	0001	1100	Q[0] = 0 restore A
2	0011	0011	100_	shift left AQ
	0011	0000	100_	A = A-m
	0011	0000	1001	Q[0] = 1
1	0011	0001	001_	shift left AQ
	0011	1110	001_	A = A-m
			0010	Q[0] = 0 & restore

Division example :-

Q.1 Dividend = 1010
Divisor = 0011
$-M = 1101$

	M	A	Q
	0011	0000	1010
4	0011	0001	010_
	0011	~~0000~~ 1101	010_
	0011	0001	0100
3	0011	0010	100_
	0011	1111	100_
	0011	0010	1000
2	0011	0101	000_
	0011	0010	0001
1	0011	0100	001_
	0011	0001	0011

A → Remainder = 0001 = $(1)_{10}$
Q → Quotient = 0011 = $(3)_{10}$

Q.2 Dividend = 10111
Divisor = 0101
$-M = 11011$

	M	A	Q
	00101	00000	10111
5	00101	00001	0111_
	00101	11100	0111_
	00101	00001	01110
4	00101	00010	1110_
	00101	11101	1110_

	00101	00010	11100
3	00101	00101	1100 –
	00101	00000	1100 1
2	00101	00001	1001 —
	00101	11100	1001 –
	00101	00001	10010
1	00101	00011	0010 –
	00101	00110	0010 –
	00101	00011	00100

Remainder = $00011 = (3)_{10}$

Quotient = $00100 = (4)_{10}$

$$\frac{23}{5} = Q = 4, \quad R = 3$$

Memory Hierarchy :-

- main memory
- Auxillary memory
- Associative memory
- Cache memory
- Virtual memory

- The memory unit is an essential component in any digital computer since it is needed for store programs & data.
- A very small computer with a limited application may be able to fulfill its intended task without the need of additional storage capacity.

Most general purpose computers would run more efficiently if they were equipped with additional storage beyond the capacity of main memory.

Not enough space to accomodate all programs in single memory space, so most computer users accumulate and continue to accumulate large amounts of data-processing software.

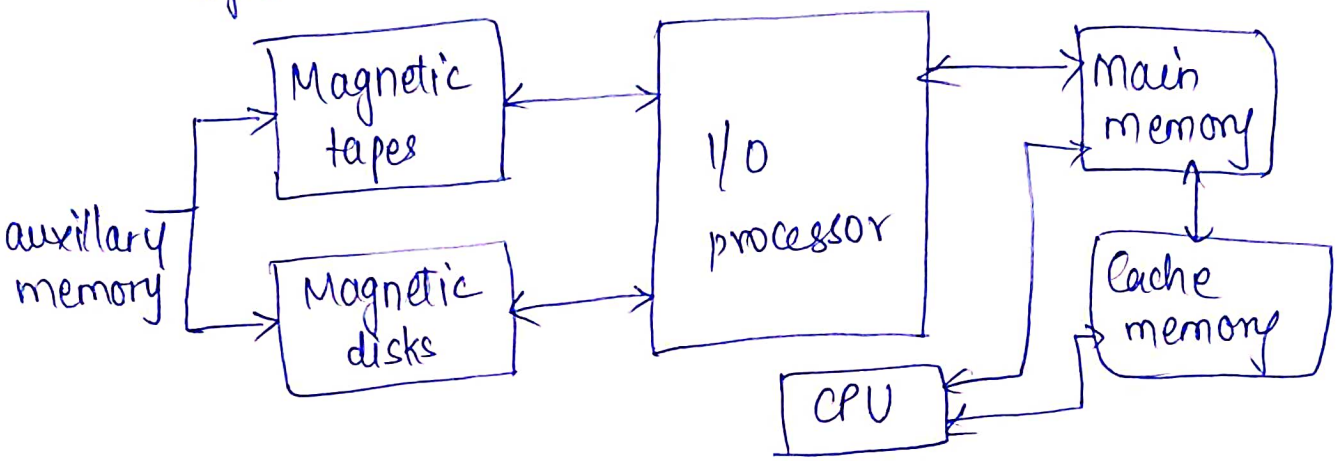
Therefore, it is economical to use low-cost storage devices to serve as a backup for storing the information that is not currently used by the CPU.

→ The memory unit that communicates directly with the CPU is called the main memory.

→ Devices that provide backup storage are called auxillary memory. eg. magnetic disks and tapes.

→ They are used for storing system programs, large data files and other backup information. only programs and data currently needed by the processor reside in main memory.

- All other information is stored in auxillary memory and transferred to main memory when needed.
- The memory hierarchy system consists of all storage devices employed in a computer system from the slow but high capacity auxillary memory to a relatively faster main memory, to an even smaller & faster cache memory accessible to the high-speed processing logic.



Memory Hierarchy

- At the bottom of hierarchy are relatively slow magnetic tapes used to store removable files. Next are the magnetic disks used as backup storage.
- The main memory occupies a central position by being able to communicate directly with the CPU and with auxillary memory devices through an IO processor. When programs not residing in main memory are needed by the CPU, they are brought in from auxillary memory.
- Programs not currently needed in main memory are transferred into auxillary memory to provide space for currently used programs & dat

A very special high speed memory called a Cache is sometimes used to increase the speed of processing by making current programs & data available to the CPU at a rapid rate. Cache is utilized for storing segments of programs currently being executed in the CPU & temporary data frequently needed in the present calculations.

While the I/O transfer / processor manages data transfer between auxillary memory and main memory, the cache organization is concerned with the transfer of information between main memory & CPU.

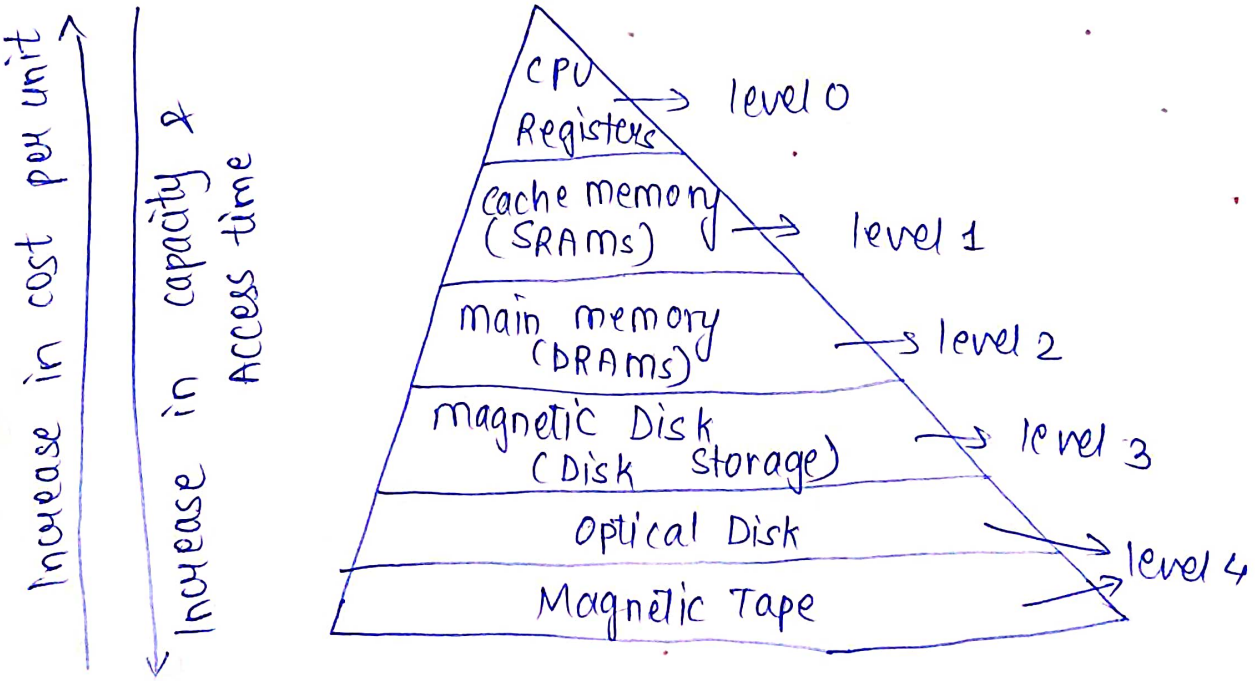


fig. Memory Hierarchy Design

As the storage capacity of the memory increase, the cost per bit for storing binary information decreases & the access time of the memory becomes longer.

The auxillary memory has a large storage capacity, is relatively inexpensive, but has low access speed compared to main memory.

→ The cache memory is very small, relatively expensive and has very high access speed. Thus as the memory access speed increases, so does its relative cost.

→ The overall goal of using a memory hierarchy is to obtain the highest possible average access speed while minimizing the total cost of the entire memory system.

① Cache holds those parts of the program and data that are most heavily used, while auxillary memory holds those parts that are not presently used by the CPU.

② the CPU has direct access to both cache & main memory but not to auxillary memory.

③ The transfer from auxillary to main memory is usually done by means of direct memory access of large blocks of data.

⇒ the typical access time ratio between cache and main memory is about 1 to 7. eg. a typical cache memory may have an access time of 100ns, while main memory access time may be 700ns.

→ Auxillary memory average access time is usually 1000 times that of main memory.

→ Block size in auxillary memory typically ranges from 256 to 2048 words, while cache block size is typically from 1 to 16 words.

MAIN MEMORY :- central storage unit in a computer system.

→ It is a relatively large & fast memory used to store programs & data during the computer operation.

The principal technology used for the main memory is based on semiconductor integrated circuits.

RAM: random access memory. ⌜ static
⌞ dynamic

① Static RAM :- consists essentially of internal flipflops that store the binary information. The stored information remains valid as long as power is applied to the unit.

② Dynamic RAM :- stores binary information in the form of electric charges that are applied to capacitors. The capacitors are provided inside the chip by mos transistors.

→ The stored charge on the capacitors tend to discharge with time & the capacitors must be periodically recharged by refreshing the dynamic memory.

→ Refreshing is done by cycling through the words every few milliseconds to restore the decaying charge.

→ DRAM offers reduced power consumption & larger storage capacity in a single memory chip. Static RAM is easier to use & has shorter read & write cycles.

→ ROM :- read only memory

- Most of the main memory in a general purpose computer is made up of RAM integrated circuit chips, but a portion of the memory may be constructed with ROM chips.
- Originally, RAM was used to refer to a random-access memory, but now it is used to designate a read write memory to distinguish it from a read only memory although ROM is also random access.

→ RAM is used for storing the bulk of the programs & data that are subject to change.

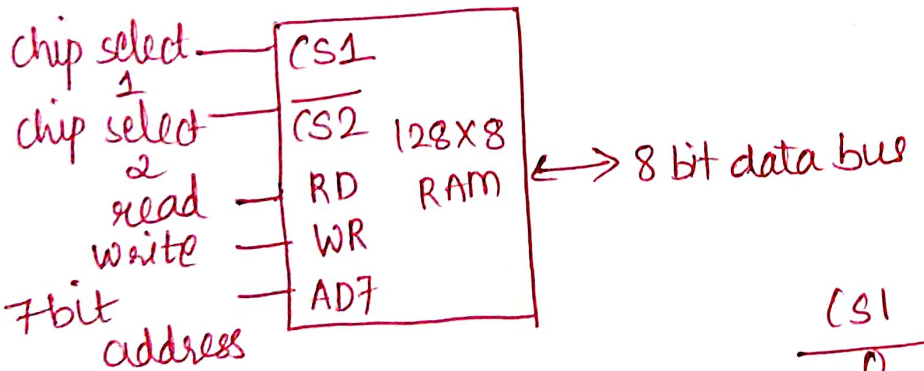
→ ROM is used for storing programs that are permanently resident in the computer & for tables of constants that do not change in value once the production of the computer is completed.

→ Rom portion of main memory is needed for storing an initial program called a bootstrap loader.

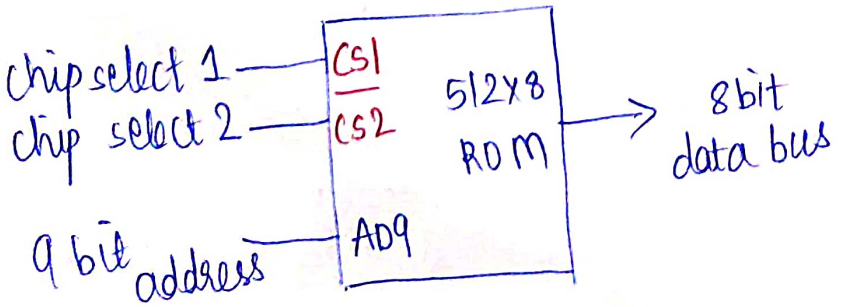
→ The bootstrap loader is a program whose function is to start the computer software operating when power is turned on.

- Since RAM is volatile, its contents are destroyed when power is turned off.
- Contents of ROM remain unchanged after power is turned off and on again.

⇒ RAM & ROM chips :- chip select 1 : active high
chip select 2 : active low



CS1	CS2	RD	WR	memory function	state of data bus
0	0	X	X	inhibit	Z
0	1	X	X	inhibit	Z
1	0	0	0	inhibit	Z
1	0	0	1	write	input data to RAM
1	0	1	X	read	output data to RAM
1	1	X	X	inhibit	Z



Memory Address Map :-

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

→ The address bus lines are shown upto 10 only because the other 6 are not used in this example and are assumed to be zero.

→ The small x's under the address bus lines designate those lines that must be connected to the address inputs in each chip.

→ RAM chips have 128 bytes & need seven address lines.

→ ROM chip have 512 bytes and need 9 address lines.

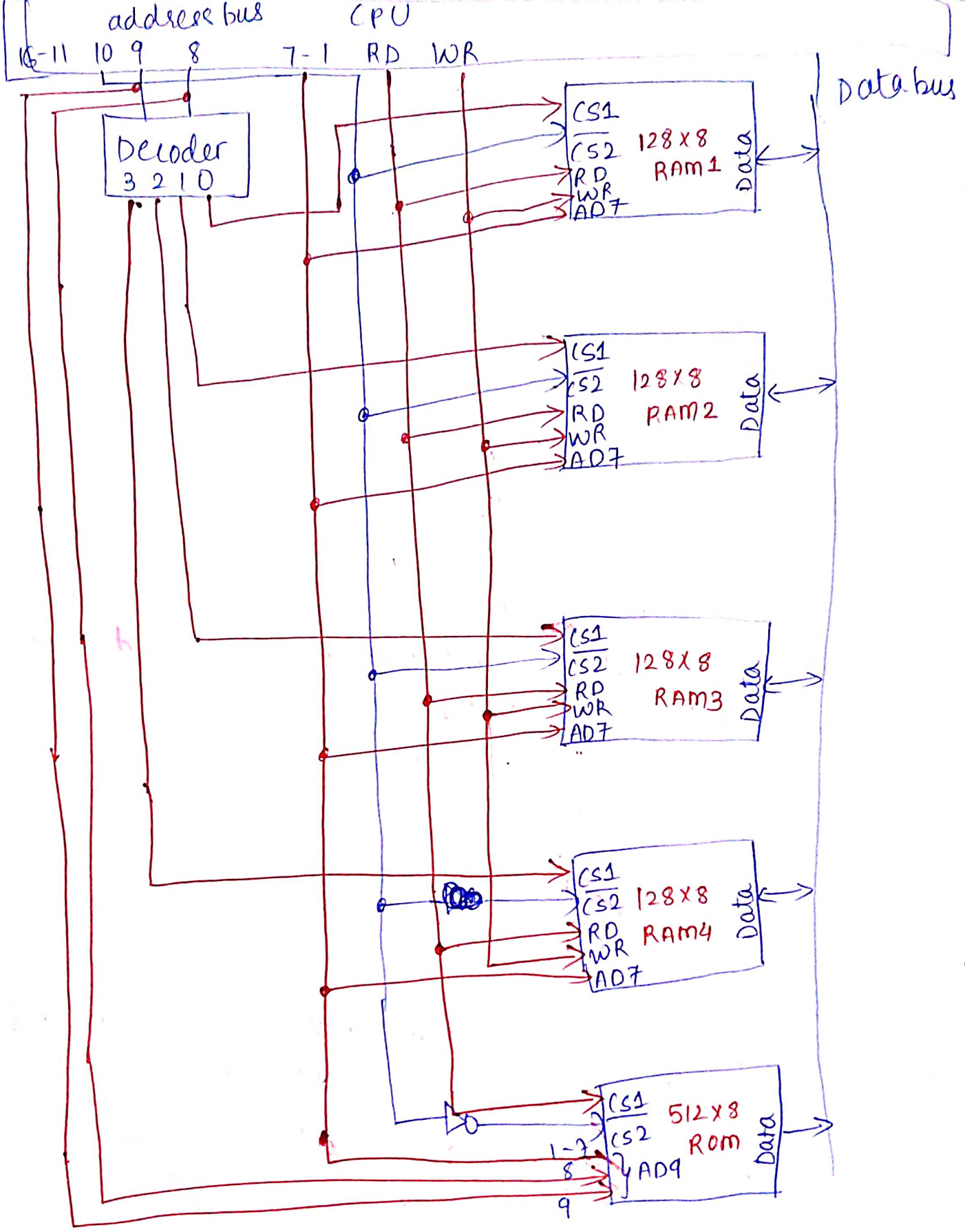
→ The x's are always assigned to the low order bus lines : lines 1 through 7 for the RAM and lines 1 through 9 for the ROM.

→ It is necessary to distinguish between 4 RAM chips by assigning different address to each a eg. bus lines 8 & 9 are used as 4 distinct binary combinations.

line 10 : 0 (RAM) ⟶
1 (ROM)

line 9	8	
0	0	- RAM 1
0	1	- RAM 2
1	0	- RAM 3
1	1	- RAM 4

Ending address = Starting address + Memory chipsize - 1



→ Memory connection to CPU

* Memory address map examples

Q.1 Given 3 RAM chips & 2 ROM chips having 128 x 8 and 512 x 8 size respectively Design memory address map.

Soln :- RAM1 : 0000 – 007F
RAM2 : 0080 – 00FF
RAM3 : 0100 – 017F
ROM1 : [redacted] 0200-03FF
ROM2 : [redacted] 0400-05FF

Q.2 Given RAM of 64KB byte size having starting address 00000. find ending address.

soln :- Size = 64KB = 2^{16}

1 0000 0000 0000 0000
1 0 0 0 0

Ending address = 00000 + 10000 – 1
= 10000 – 1 = FFFF H

* if starting address = FFFF
then ending address = 1FFFF – 1
= 1FFFE H

Q.3 If starting address of ROM is 2000H having 1K size. Find its ending address.

memory size = 1024

$= \underset{4}{1000}\,\underset{0}{0000}\,\underset{0}{00}$

$= 400\,H$

Ending address = 2000 + 400 − 1

$= 23FF\,H$

Q.4 Write chip starting and ending addresses in binary.

→ bits : 15 − 0

1] 4K RAM (4096) 2^{12} (1000H), SA = 0000H

2] 1K EEPROM (1024) 2^{10} (400H), SA = 2000H

3] 32K ROM (32768) 2^{15} (8000), SA = 8000H

(1)

SA = 0000

EA = 0000 + 1000 − 1

= 0FFF H

(2) SA = 2000

EA = 2000 + 400 − 1

= 23FF H

(3) SA = 8000

EA = 8000 + 8000 − 1

= 8000 + 7FFF

= FFFF H

AUXILIARY MEMORY :- Devices that provide backup storage are called auxiliary memory.

→ eg. Magnetic disks, tapes, magnetic drums, magnetic bubble memory, optical disks.

→ The average time required to reach a storage location in memory & obtain its contents is called the access time.

→ In electromechanical devices with moving parts such as disks & tapes, the access time consists of a seek time required to position the read-write head to a location & a transfer time required to transfer data to or from the device.

→ Because the seek time is usually much longer than the transfer time, auxillary storage is organized in records or blocks.

→ A record is a specified number of characters or word.

→ Reading / writing is always done on entire records.

Magnetic disks / drums :-

→ It consists of high speed rotating surfaces coated with a magnetic recording medium.

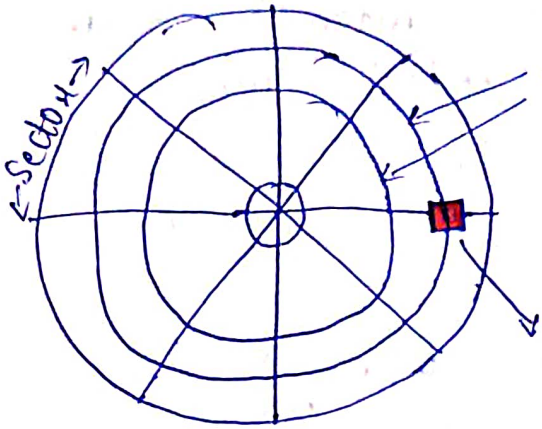
→ Rotating surface of drum is cylinder & disk is round flat plate

→ The recording surface rotates at uniform speed & is not started or stopped during access operations

→ Bits are recorded as magnetic spots on the surface as it passes a stationary mechanism called a write head.

→ Stored bits are detected by a change in magnetic field produced by a change in magnetic field produced by a recorded spot on the surface as it passes through a readhead.

→ A magnetic disk is a circular plate constructed of metal or plastic coated with magnetized material.



→ All disks rotate together at high speed & are not stopped or started for access purposes

→ Bits are stored in the magnetized surface in spots along concentric circles called tracks. The tracks are commonly divided into sections called sectors. The minimum quantity of information that can be transfer is a sector.

→ The amount of surface available for recording in a disk is greater than in a drum of equal physical size.

→ Disks that are permanently attached to the unit assembly and cannot be removed by the occasional user are called hard disks.

→ A magnetic tape transport consists of the electrical, mechanical and electronic components to provide the parts and control mechanism for a magnetic tape unit.

→ 7 or 9 bits are recorded simultaneously to form a character together with a parity bit. Read/write heads are mounted one each track so that data can be recorded & read as a sequence of characters.

→ Magnetic tape units can be stopped, started to move forward or in reverse or can be rewo

→ However they cannot be ~~recorded~~ started / stopped fast enough between individual characters. For this reason, information is recorded in blocks referred as records.

→ ASSOCIATIVE MEMORY :- The time required to find an item stored in memory can be reduced considerably if stored data can be identified for access by the content of the data itself rather than by an address.

→ A memory unit accessed by content is called an associative memory or content addressable memory (CAM).

→ This type of memory is parallely accessed by basic data content rather than by specific address or location.

→ When a word is written in an associative memory, no address is given, The memory is capable of finding an empty unused location to store the word.

→ When a word is to be read from an associative memory, content of word or part of word is specified. The memory locates all words which match the specific content & marks them for reading.

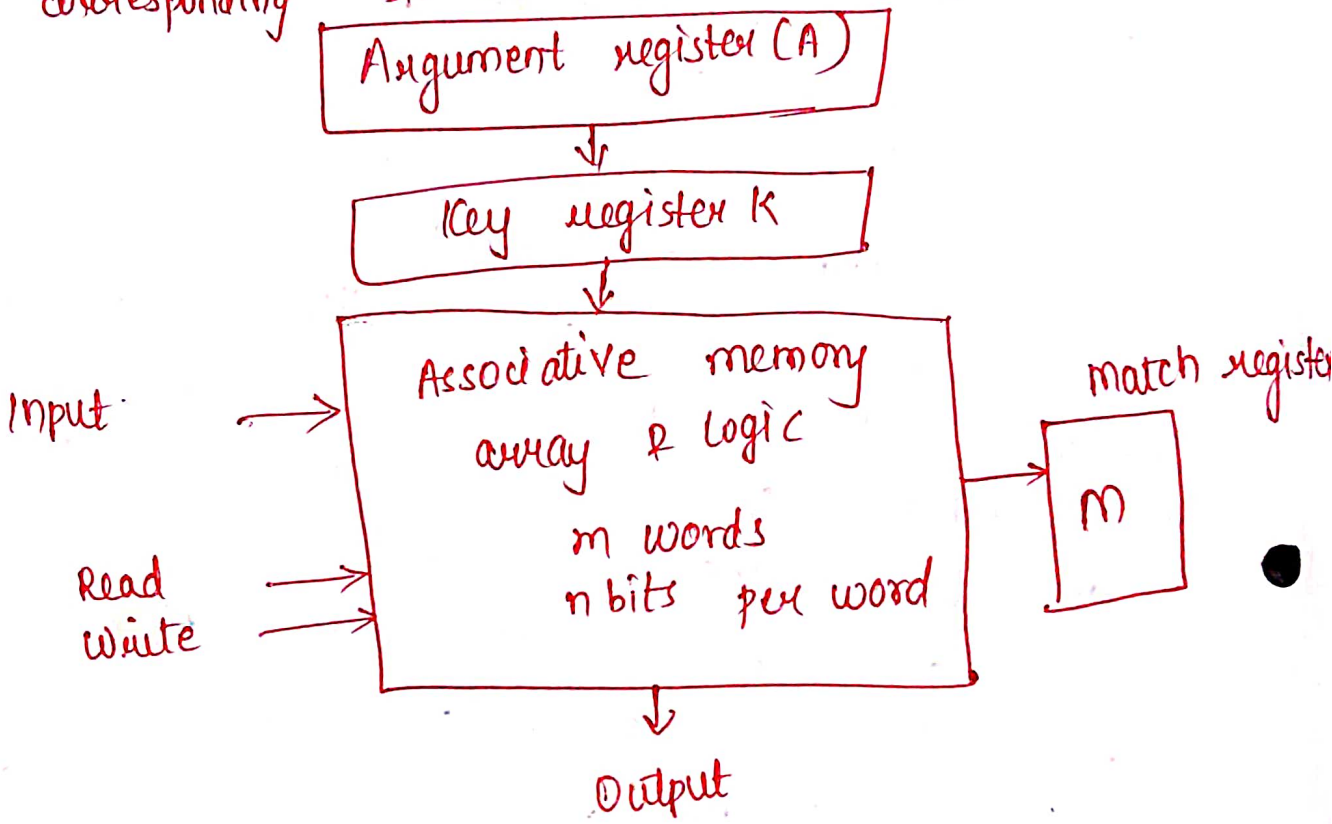
→ Searches can be done on an entire word or on speafic field within a word. An associative memory is more expensive than RAM because each cell must have storage capability as well

as logic circuits for matching its content with an external argument.

Applications :- where search time is short / critical.

Hardware Organization :- A and K registers are each of n bits, one for each bit of a word. M has m bits, one for each memory word. each word in memory is compared in parallel with the content of the argument register

→ the words that match the bits of argument register set a corresponding bit in the match register.



→ After the matching process, those bits in the match register that have been set indicate the fact that their corresponding words have been matched.

→ Reading is accomplished by sequential access to memory for those words whose corresponding bits in the match register have been set.

→ The key register provides a mask for choosing a particular field or key in the argument word.

→ The entire argument is compared with each memory word if the key register contains all 1's. otherwise, only those bits in the argument that have 1's in their corresponding position of key register are compared.

→ Thus the key provides a mask on identifying piece of information which specifies how reference to memory is made.

Example:- Suppose argument & key register have following configuration:

Only the 3 leftmost bits of A are compared with memory words because K has 1's in these positions

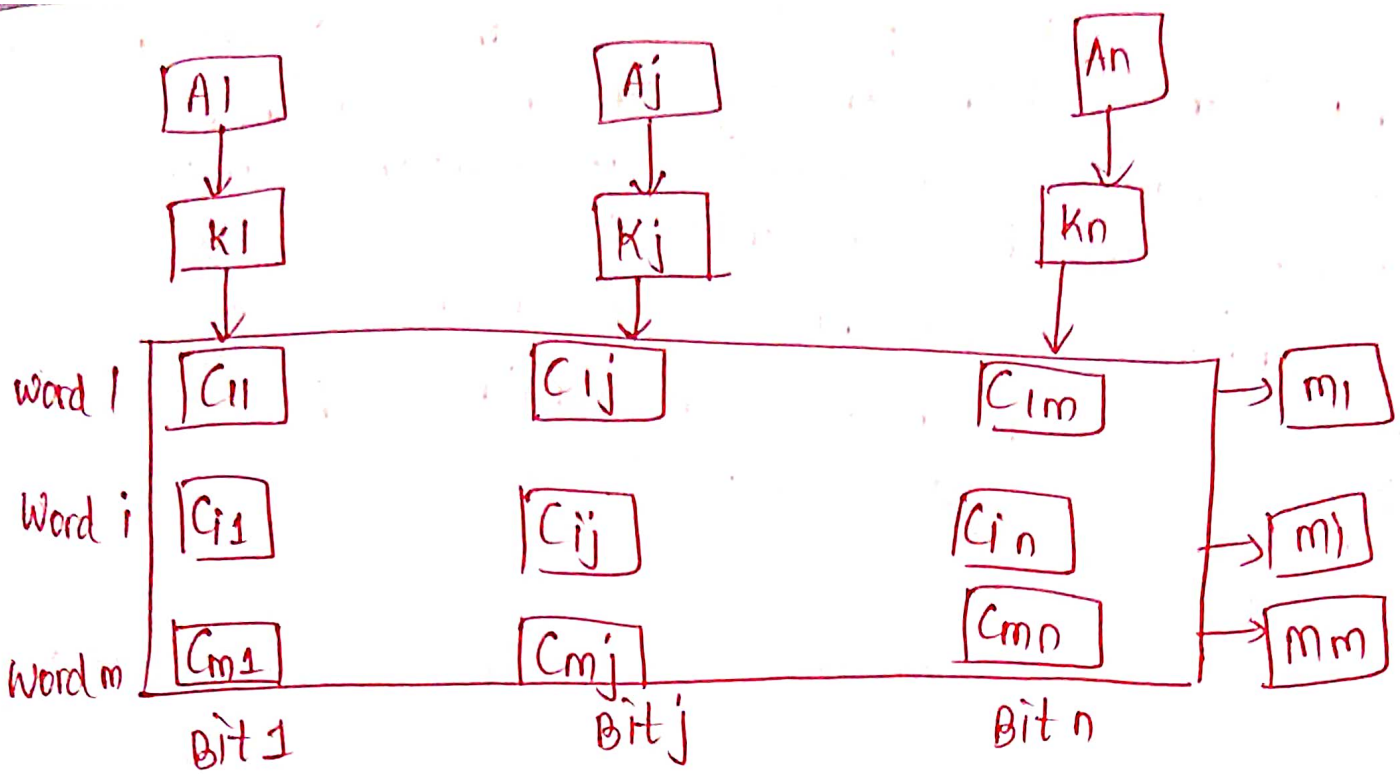
```
A       101 111100
K       111 000000

Word 1  100 111100   no match
Word 2  101 000001   match
```

→ Word 2 matches the unmasked argument field because the 3 leftmost bits of the argument & the word are equal.

Hardware:- Cell Cij is the cell for bit j in word i. A bit Ai in the argument register is compared with all the bits in column j of array provided that Ki = 1.

→ This is done for all columns j = 1, 2, ... n. If a match occurs between all the unmasked bits of the



argument & the bits in word i, the corresponding bit m_i in the match register is set to 1.

→ If one or more unmasked bits of the argument & the word donot match, m_1 is cleared to 0.

here i = word number
j = bit position in word

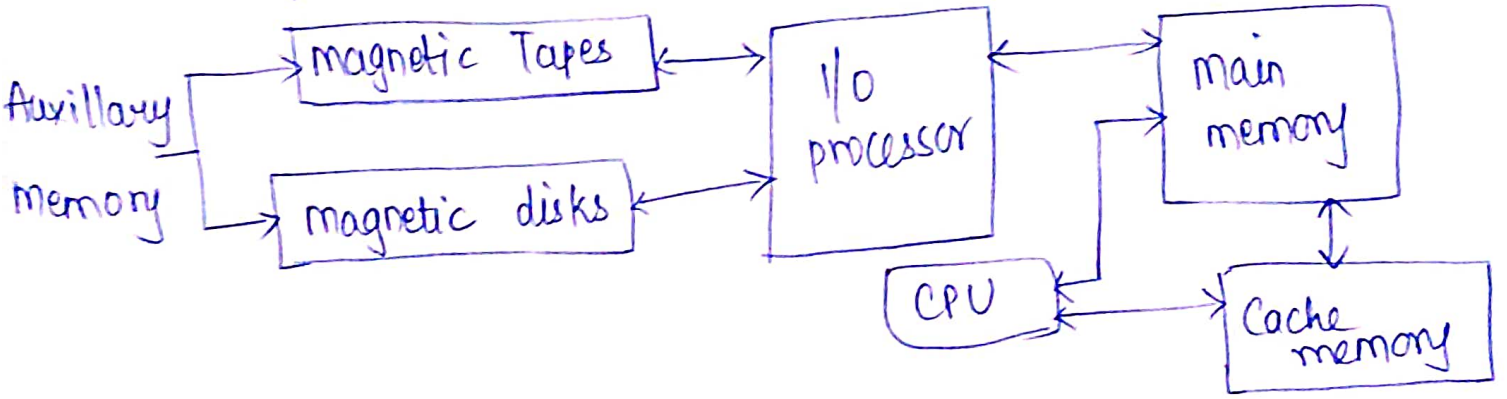
CACHE MEMORY:- If the active portion of the program & data are placed in fast small memory, the average memory access time can be reduced, thus reducing total execution time of program. Such a fast small memory is referred to as a cache memory.

→ It is placed between main memory & CPU.

→ It's access time is 5 to 10 times lesser than main memory.

→ It is the fastest component in the memory hierarchy & approaches the speed of CPU components.

- Memory Hierarchy in a computer system:



- the fundamental idea of cache organization is that by keeping the most frequently accessed instructions & data in the fast cache memory, the average memory access time will approach the access time of cache.

→ when the CPU needs to access memory, cache is examined.

→ If the word addressed by the CPU is not found in cache, the main memory is accessed to read the words.

→ A block of words containing just accessed ones are transferred from main to cache memory

→ The block size may vary from 1 word to 16 words adjacent to the one just accessed.

→ Performance of cache memory is measured in terms of quantity called hit ratio.

→ The ratio of no. of hits divided by the total CPU references to memory (hits + misses) is the hit ratio.

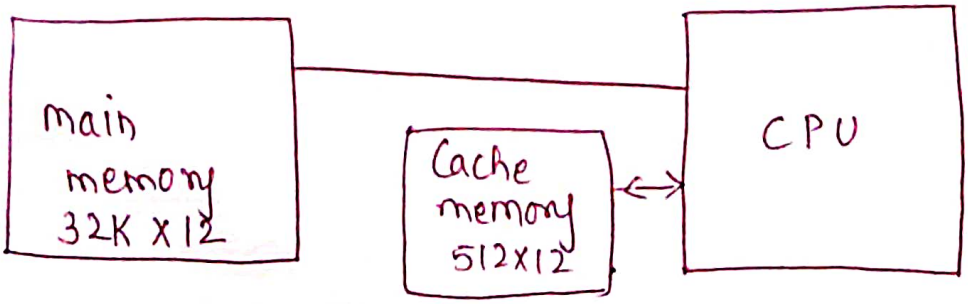
→ Hit ratios of ≥ 0.9 are ideally reported by running representative programs in computer.

→ When the CPU refers to memory & finds the word in cache, it will produce a hit.
If word not found in cache, it is in main memory then it counts as a miss.

→ the transformation of data from main memory to cache memory is referred as a mapping process.

→ 1] Associative mapping
2] Direct mapping
3] Set-associative mapping

eg.



* Auxillary Memory :-

Accessing data on disk requires the following :-

(1) Seek Time :- The time taken to move the read/write head to the desired track is known as seek time.

(2) Latency time or Search Time :- It is the time required to bring the starting position of the address sector of the track to come under read/write head.

(3) Data transfer rate :- Once the read/write head is positioned at the right a sector, the data has to be written to disk or read from disk. The rate at which data is written to disk or read from disk is known as data transfer rate.

(4) Access Time :- It is the time taken to move the read/write head to the address sector.

Access Time = Seek Time + Latency Time

* Associative memory :- Avoids address based search, To reduce search Time in memory, content based search is used.

8.1

eg.

A	101	111100
K	111	000000
	unmasked	masked

			M	
Word 1	100	111100	0	— no match
Word 2	101	000001	1	— match
Word 3	101	111100	1	
Word 4	110	001010	0	
Word 5	111	000111	0	

Q.2 A = 101 00101

K = 110 00000

			M	
Word 1	001	11111	0	no match
" 2	100	00000	1	match
" 3	110	00101	0	
" 4	101	00101	1	match
" 5	100	11010	1	match

Q. To resolve issue of dropping hit ratio, what we need to do ?

↳ Division of index based on increase block size.

↳ previously we took 1 Block = 1 word.

Now if we want 1 Block = 8 words

then No. of blocks = $\frac{512}{8}$ = 64 blocks.

So;

	Index	Tag	Data	
Block 0	000	01		
	001			
	007	01		hit ratio is improved here
Block 1	010	02		
	017	02		
Block 63	100			
	777			

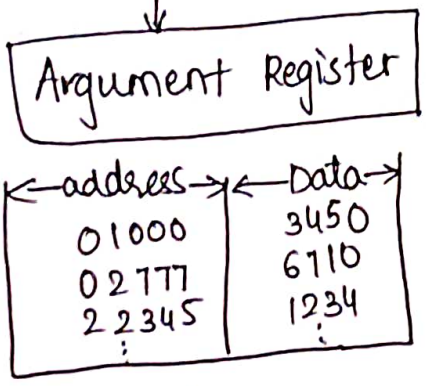
* For every word stored in cache, there is a duplicate copy in main memory.

→ the CPU communicates with both memories. It first sends a 15-bit address to cache.

→ If there is a hit, CPU accepts the 12-bit data from cache. If there is a miss, CPU reads the word from main memory & the word is then transferred to cache.

→ the fastest & most flexible cache organization uses an associative memory.

CPU address (15 bits)



} associative mapping

→ the associative memory stores both the address & content (data) of memory word.

→ Here the address value of 15 bit is shown as a 5 digit octal no. and its corresponding 12-bit word is shown as 4 digit octal no.

→ A CPU address of 15-bits is placed in the argument register & the associative memory is searched for a matching address. If address is found, corresponding 12-bit data is read & sent to the CPU.

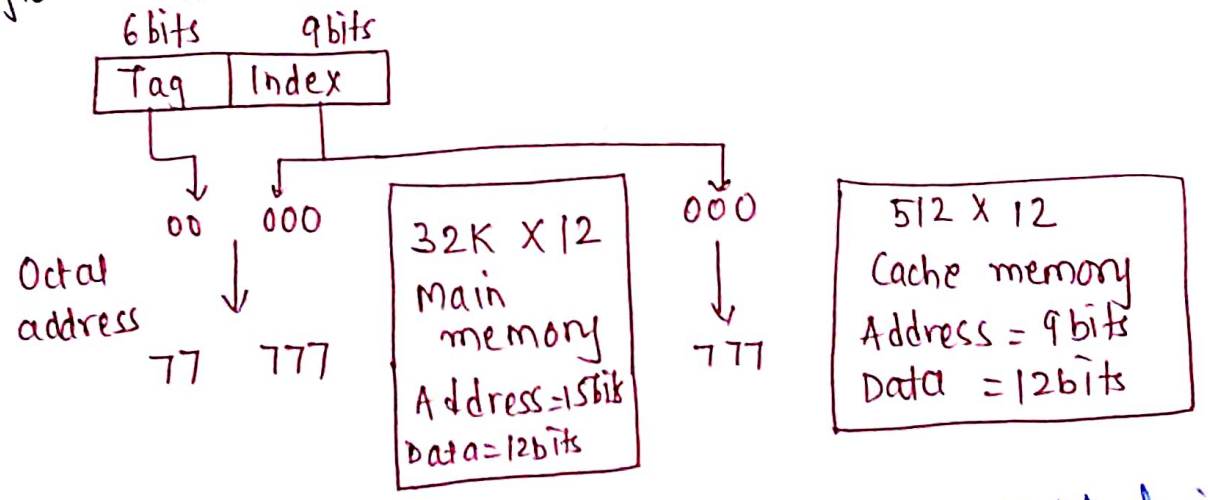
→ If no match occurs, the main memory is accessed for the word. The address-data pair

is then transferred to the associative cache memory. If cache is full, an address-data pair must be displaced to make room for a pair that is needed and not presently in the cache.

→ The decision as to what pair is replaced is determined from replacement algorithm that the designer chooses for the cache.

→ A simple procedure is to replace cells of the cache in round-robin order whenever a new word is requested from main memory. It constitutes a FIFO (first in first out) replacement policy.

(2) DIRECT MAPPING:- Associative mapping is expensive compared to RAM because of added logic associated with each cell.



→ The CPU address of 15 bits is divided into 2 fields. The nine LSBs constitute the index field & the remaining 6 bits form the tag field.

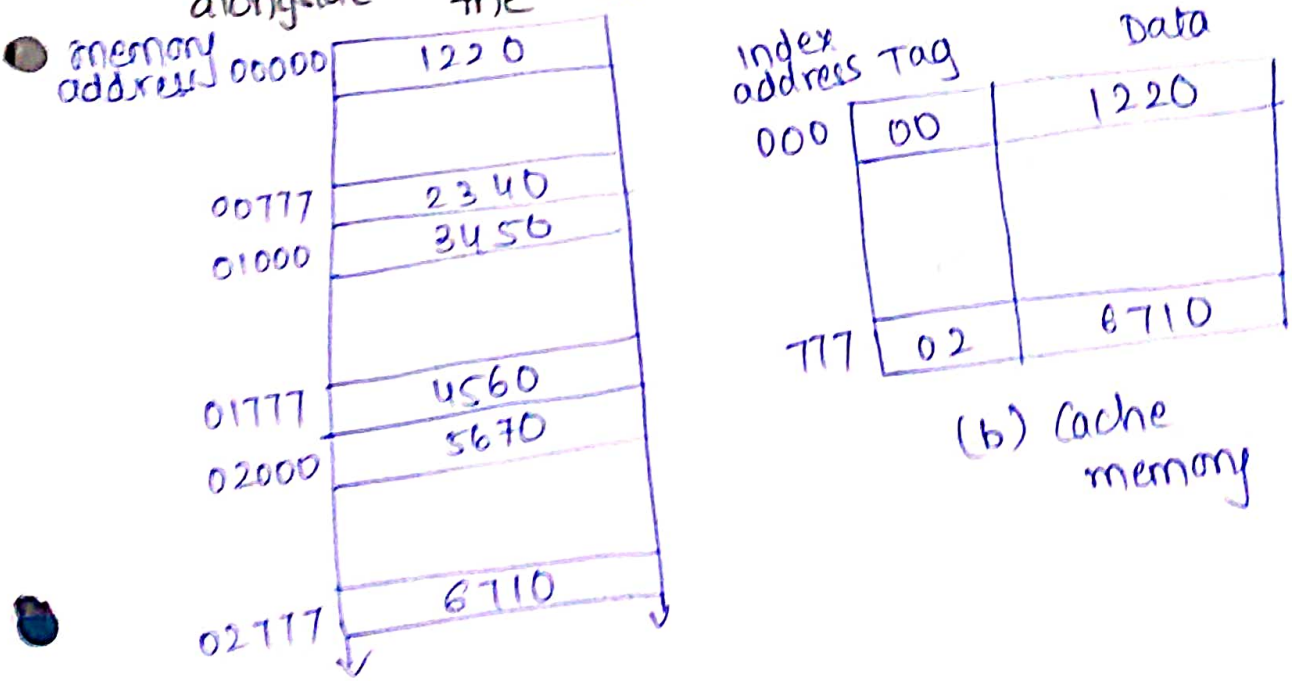
→ the no. of bits in the index field is equal to no. of address bits required to access the cache memory.

DIRECT MAPPING:- there are 2^k words in cache memory & 2^n words in main memory

→ The n-bit memory address is divided into 2 fields k bits for index field & n-k bits for tag field.

→ The direct mapping cache organization uses the n-bit address to access the main memory & k-bit index to access the cache.

→ Each word in cache consists of the data word & its associated tag. When a new word is first brought into the cache, the tag bits are stored alongside the data bits.



(a) Main memory

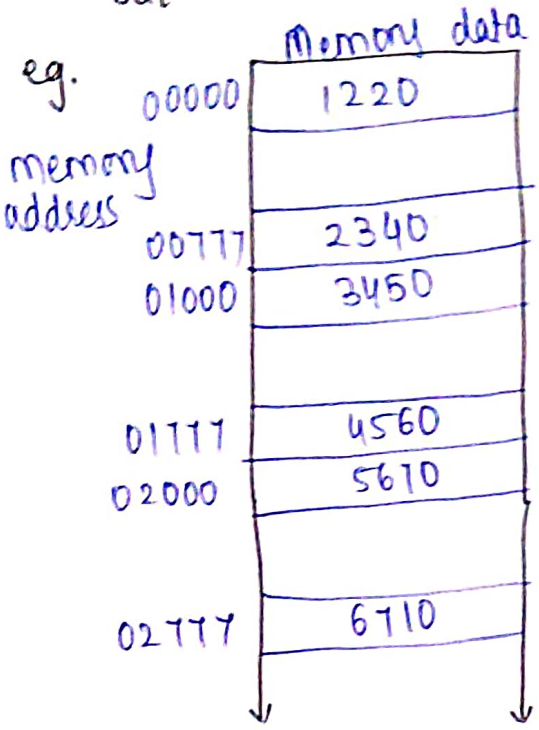
(b) Cache memory

→ When the CPU generates a memory request, the index field is used for address to access the cache. The tag field of the CPU address is compared with the tag in the word read from the cache. If the 2 tags match, there is a hit & the desired data word is in cache. If there is no match, there is a miss & the required word is read from main memory.

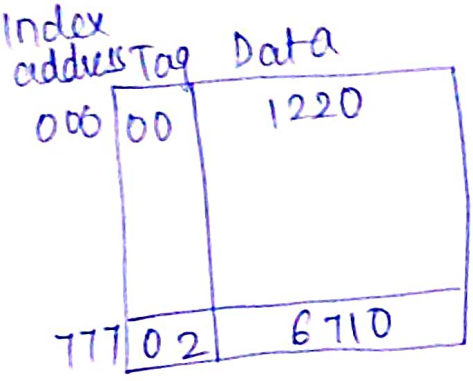
→ It is then stored in the cache together with the new tag, replacing the previous value.

→ The disadvantage of direct mapping is the hit ratio can drop considerably if 2 or more words whose addresses have the same index but different tags are accessed repeatedly.

eg.



(a) main memory



(b) Cache memory

→ The word at address zero is presently stored in the cache (index = 000, tag = 00, data = 1220)

→ Suppose that the CPU now wants to access the word at address 02000. The index address is 000, so it is used to access cache. The two tags are then compared.

→ The cache tag is 00 but the address tag is 02, which doesnot produce a match. Therefore main memory is accessed & the data word 5670 is transferred to the CPU. The cache word at index address 000 is then placed with a tag of 02 & data of 5670

Set associative mapping :-

→ Disadvantage of direct mapping is that 2 words with the same index in their address but with different tag values cannot reside in cache memory at same time.

→ A third type of cache organization called set associative mapping, is an improvement over the direct mapping organization in that each word of cache can store 2 or more words of memory under the same index address.

→ Each data word is stored together with its tag & no. of tag data items in 1 word of cache is said to form a set.

Index	Tag	Data	Tag	Data
000	01	3450	02	5670
:				
111	02	6710	00	2340

Each index address refers to 2 data words & their associated tags. Each tag requires 6 bits & each data word has 12 bits,

so the word length is $2(6+12) = 36$ bits.

→ An index address of 9 bits can accomodate 512 words. Thus the size of cache memory is 512×36.

→ It can accomodate 1024 words of main memory since each word of cache contains 2 data words.

In general, a set associative cache of set size k will accomodate k words of main memory in each word of cache.

→ The words stored at addresses 01000 and 02000 of main memory are stored in cache memory at index address 000.

→ Same for words at addresses 02777 & 00777 stored in cache at index address 777.

→ When the CPU generates a memory request, the index value of address is used to access cache.

→ The tag field of CPU address is then compared with both tags in cache to determine if a match occurs.

→ The comparison logic is done by an associative search of tags in set similar to associative memory, thus name as "set associative".

→ The hit ratio will improve as the set size increases because more words with the same index but different tags can reside in cache.

→ However, an increase in the set size increases the no. of bits in words of cache & requires more complex comparison logic.

→ When a miss occurs here, set is full, it is necessary to replace 1 of tag data items with a new value.

→ Algorithms used are : random replacement, FIFO, LRU (least recently used).

→ In random replacement policy the control chooses 1 tag data item for replacement at random.

→ FIFO procedure selects for replacement the item that has been in the set the longest.

→ LRU algorithm selects for replacement the item that has been least recently used by CPU.

→ Both FIFO & LRU can be implemented by adding a few extra bits in each word of cache.

WRITING A CACHE:-

(1) Write through (simplest & most common)

- update main memory with every memory write operation with cache memory being updated in parallel if it contains the word at the specified address.

Advantage:- main memory always contains the same data as cache.

- It is important in direct memory access transfer systems.

- It ensures that the data residing in main memory are valid at all times so that an I/O device communicating through DMA would receive most recent updated data.

(2) Write Back :- only cache location is updated during a write operation.

→ The location is then marked by a flag so that later when the word is removed from the cache it is copied into main memory.

→ The reason for the write back method is that during the time a word resides in cache, it may be updated several times.

→ As long as word remains in cache, it doesnot matter whether the copy in main memory is out of date, since requests from word are filled from the cache.

→ It is only when the word is displaced from cache that an accurate copy need be rewritte into main memory.

→ Analytical results indicate that no. of memory wri -es in a typical program ranges between 10 & 30 % of total references of memory.

Numericals

① Hit ratio (H) = $\frac{\text{hit}}{\text{hit + miss}} = \frac{\text{no. of hits}}{\text{total accesses}}$

② Miss ratio = $\frac{\text{miss}}{\text{hit + miss}} = \frac{\text{no. of misses}}{\text{total accesses}}$

Q. If a cache has 51 hits & 3 misses over a period of time. what will be hit & miss ratio of cache ?

→ Hit ratio = $\frac{51}{51+3} = \frac{51}{54}$

→ miss ratio = $\frac{3}{54}$

Chapter 10 :- I/O organization

↳ Input-Output interface, Asynchronous Data Transfer
↳ Memory mapped I/O, I/O mapped I/O, modes of transfer, priority interrupt.

(1) **Input-Output Interface** : provides a method for transferring information between internal storage & external I/O devices.

→ Peripherals connected with computer need communication link for interfacing with CPU.

i) Conversion of signal values is required as peripherals are electromechanical & electromag-netic devices and their operation is different from CPU & memory.

ii) Data transfer rate of peripherals is slower than transfer rate of CPU, consequently synchronization mechanism may be needed.

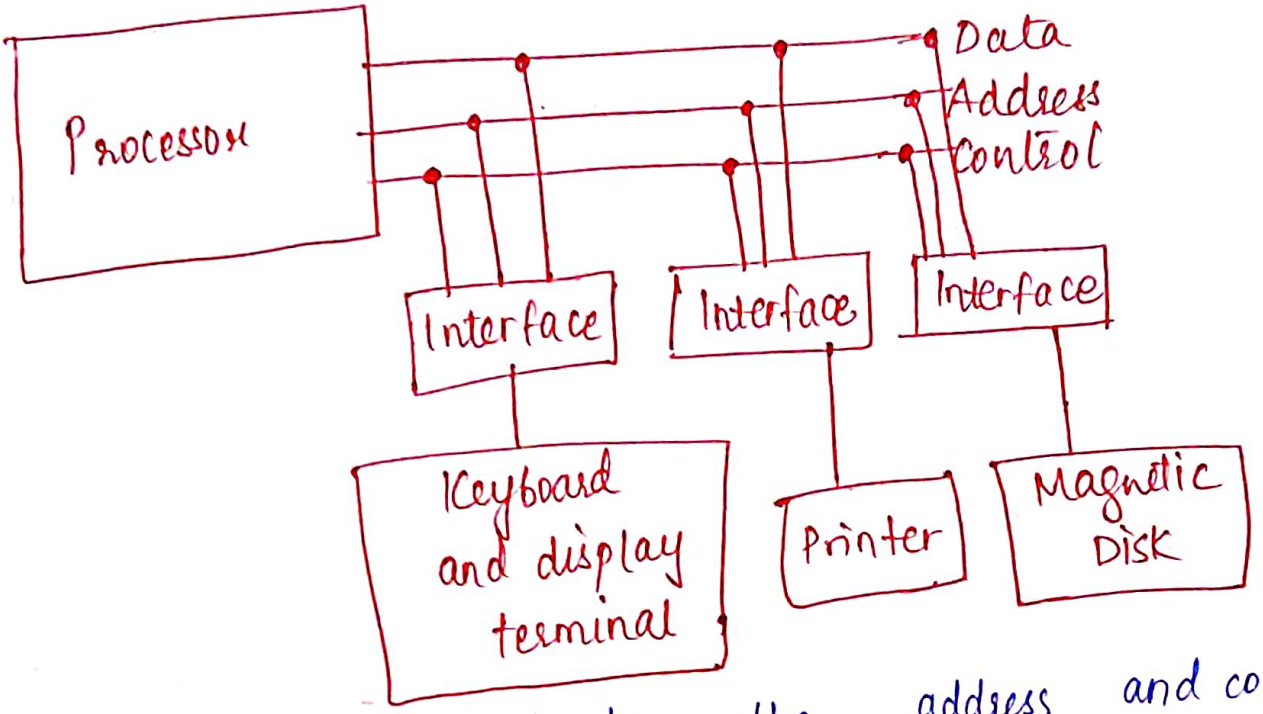
iii) Data codes & formats in peripherals differ from word format in the CPU & memory.

iv) Operating modes of peripherals are different from each other & each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

→ To resolve this differences, special synchronization hardware is required between CPU & peripherals called interface units as it interfaces between processor bus and peripheral device. Two interfaces { CPU interface, I/O interface

I/O device — I/O interface — System bus.

→ Main function of I/O interface are data conversion, synchronization & device selection.



→ Each interface decodes the address and control received from IO bus, interprets them for the peripheral, and provides signals for the peripheral controller.

→ It also synchronizes the data flow and supervis the transfer between peripheral and processor.

→ Each peripheral has its own controller that operate the particular electromechanical device.

→ eg. printer controller controls the paper motion the print Timing & the selection of printing characters. A controller may be housed separately or may be physically integrated with the peripheral.

→ The I/O bus from the processor is attached to a peripheral interface.

→ To communicate with a particular device, the processor places a device address on the address lines.

→ Each interface attached to the I/O bus contains an address decoder that monitors the address lines. when the interface detects its own address, it activates the path between the bus lines & the device that it controls.

→ All peripherals whose address does not correspon -d to the address in the bus are disabled by their interface. Same times processor provides a function code in the control lines.

- there are 4 types of commands that an interface may receive:
  1) control 2) status 3) data output 4) data input.

- A control command is issued to activate the peripheral & to inform it what to do.
  eg. magnetic tape may be instructed to backspace, rewind or start the tape moving in forward direction.

- A status command used to test various status conditions in the peripheral interface & the peripheral
  eg. computer may wish to check the status of the peripheral. before a transfer is iniated.

- Data output command causes the interface to respond by

transferring data from bus into one of its registers

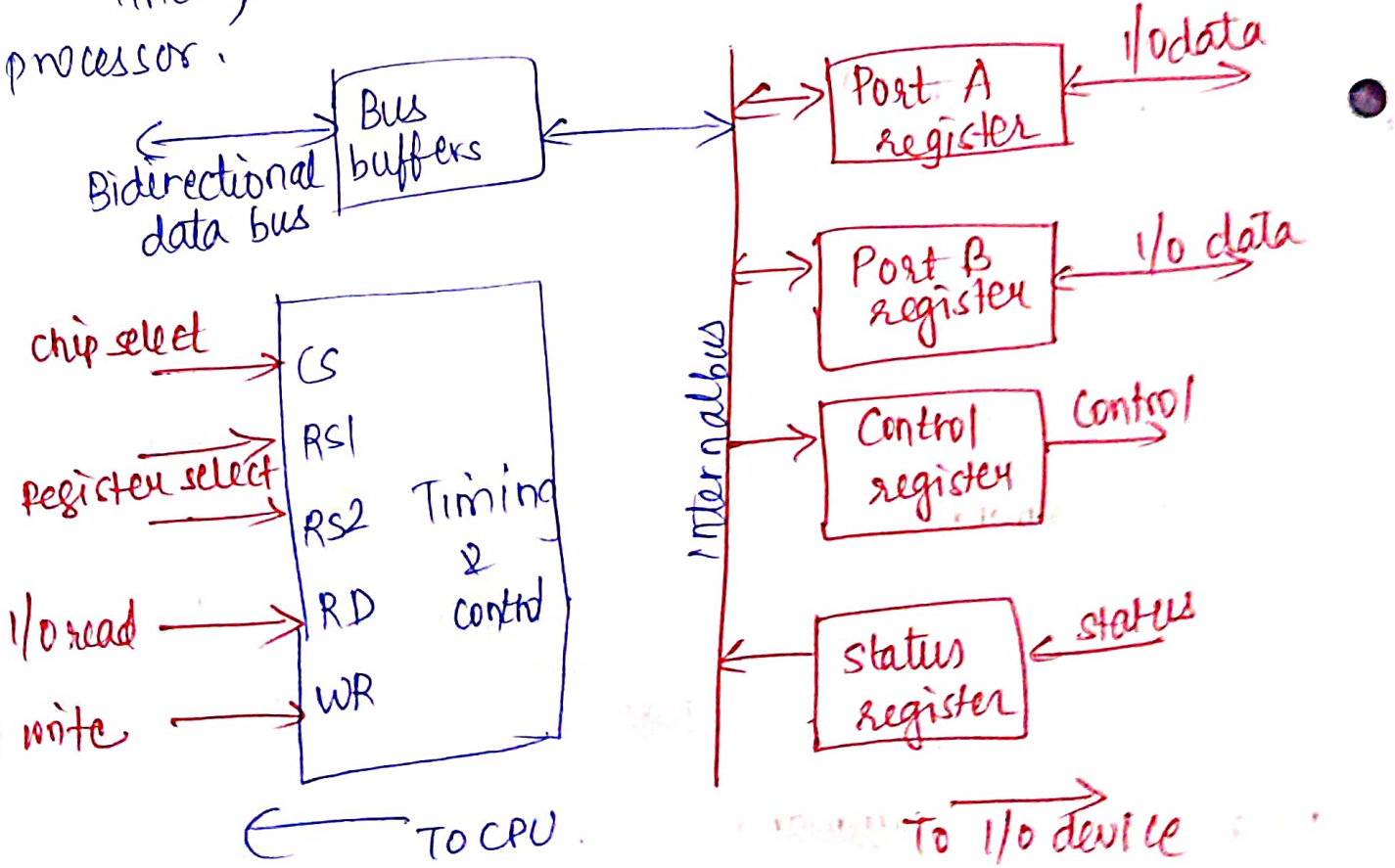
→ The computer starts tape moving by issuing a control command. The processor then monitors the status of the tape by means of a status command & if it is in correct position, the processor issues a data output command.

→ Data input command is opposite to data output.

→ In this case the interface receives an item of data from the peripheral & places it in its buffer register.

→ The processor checks if data are available by means of status command & then issue a data input command.

→ The interface places the data on the data lines, where they are accepted by the processor.



→ CS input (chip select) enables when the interface is selected by the address bus.

→ The two register select inputs RS1 & RS0 are usually connected to the two least signi-ficant lines of the address bus.

→ It is used to select one of 4 registers:

CS	RS1	RS2	Register Selected
0	X	X	None: data bus in Z
1	0	0	Port A register
1	0	1	Port B register
1	1	0	Control register
1	1	1	status register

→ The content of the selected register is transf-erred into the CPU via the data bus when the I/O read signal is enables.

→ The CPU transfers binary information into selected register via data bus when the I/O write input is enabled.

* ISOLATED Vs MEMORY MAPPED I/O :-

I/O mapped I/O

→ uses common bus for data transfer isolated I/O.
→ complex but high flexible
→ Separate I/O instruction set
→ uses separate memory or I/O R/W lines
→ own address spaces for memory & I/O

Memory mapped I/O :- → Only one set of R/W signals

→ uses same address space with memory
→ Interface registers considered as a part of memory system.
→ Reduces memory address space available
→ same instruction set

Asynchronous data transfer

→ Internal operations in digital system controlled by a common pulse generator.
→ Each register unit are applied a common clock for simultaneous transfer between 2 units called synchronous transfer.
→ While in most cases, independent private clock is applied to each unit for independent internal timing for registers.
Two units in this case share asynchronous modes of transfer.
→ Asynchronous data transfer between 2 independent units requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted.

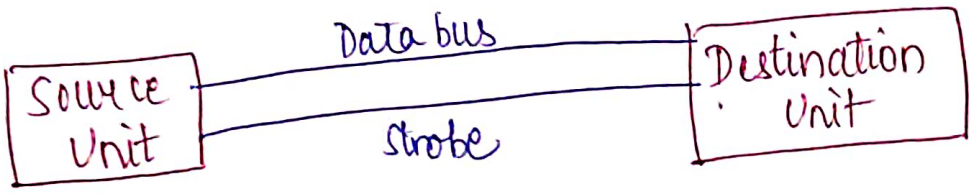
(1) Strobe pulse (s) supplied by one of the units to indicate to other unit when transfer has occur.

(2) To accompany each item being transferred with a control signal that indicates presence of data in the bus.

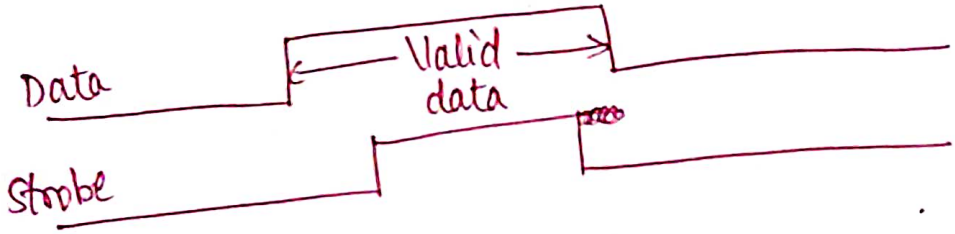
→ The unit receiving the data item responds with another control signal to acknowledge receipt of the data. This type of agreement between 2 independent units is referred to as Handshaking.

STROBE CONTROL:- employs a single control line to time each transfer.

1] Source initiated strobe for data transfer.
2] Destination initiated strobe for data transfer.



(a) Block Diagram



(b) timing diagram

→ the data bus carries the binary information from source unit to destination unit.

→ the strobe is a single line that informs the destination unit when a valid data word is available in the bus.

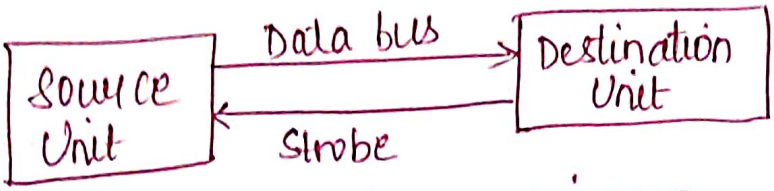
→ the ~~strobe~~ source unit first places the data on the data bus.

→ After a delay to ensure that the data settle to a steady value, the source activates the strobe pulse. The information on the data bus and the strobe signal

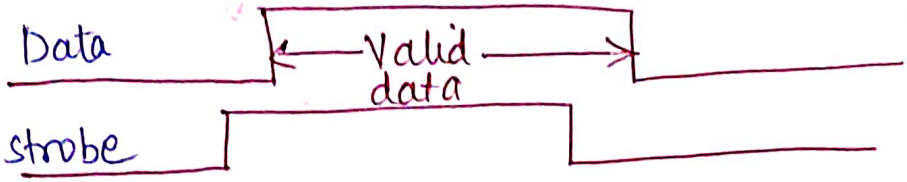
remain in the active state for a sufficient time period to allow the destination unit to receive the data.

→ The source removes the data from bus a brief period after it disables its strobe pulse.

- Destination Iniated strobe for data transfer



(a) block diagram



(b) timing diagram

→ The destination unit activates the strobe pulse, informing the source to provide the data.

→ The source unit responds by placing the requested binary information on the data bus.

→ The data must be valid & remain in bus as long as destination unit accept it.

→ The falling edge of strobe pulse can be used again to trigger a destination register.

→ The destination unit then disables the strobe. the source removes the data from the bus after a predetermined time interval. The transfer of data between the CPU & an interface unit is similar.

Disadvantage :- no acknowledgement for source unit whether destination unit has actually received the data item that was placed in bus.

→ Similarly, a destination unit initiates the transfer has no way of knowing whether the source unit has actually placed the data on the bus.

HANDSHAKING METHOD :- solves disadvantage of strobe method by introducing a second control signal that provides a reply to the unit that initiates the transfer.

(1) Source initiated transfer using handshaking
(2) Destination initiated transfer using handshaking

(1) Source iniated transfer using handshaking

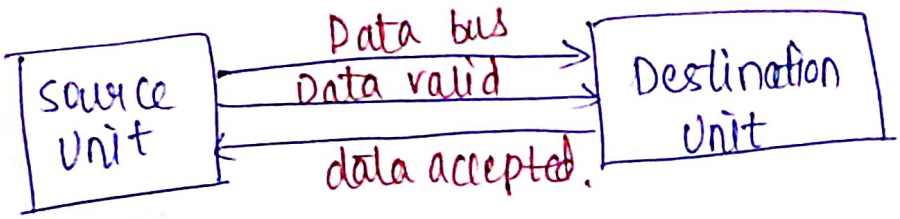
→ one control line is in the same direction as the data flow in the bus from source to destination.

→ It is used by the source unit to inform the destination unit whether it is valid data on bus.

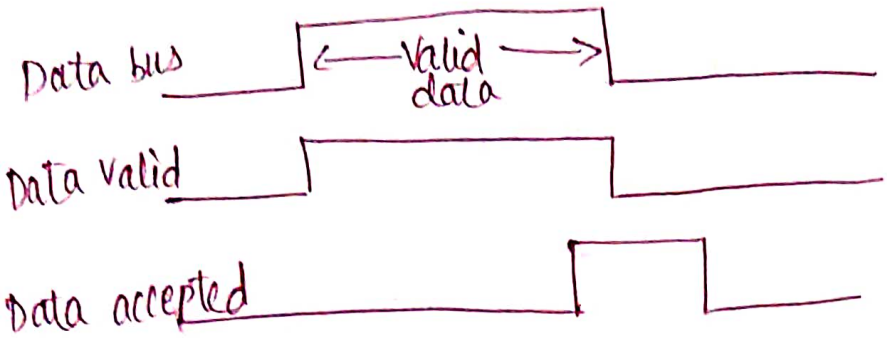
→ Other control line is in opposite direction from destination to source.

→ It is used by the destination unit to inform the source whether it can accept data.

→ The sequence of control during the transfer depends on the unit that initiates the transfer.



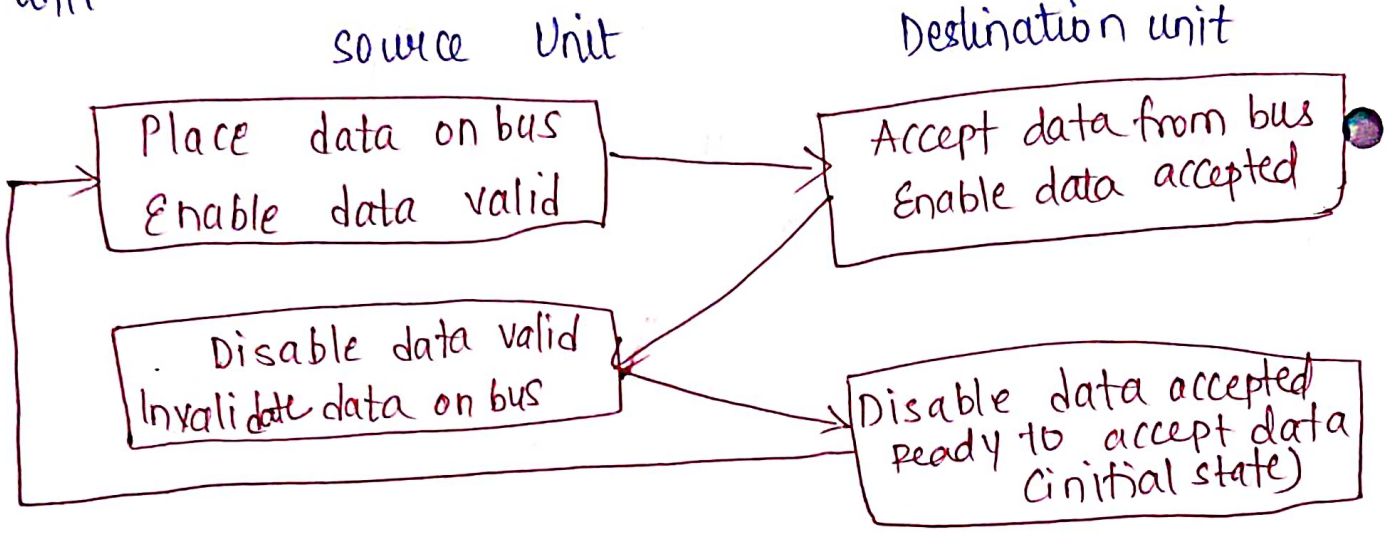
(a) block diagram



(b) timing diagram

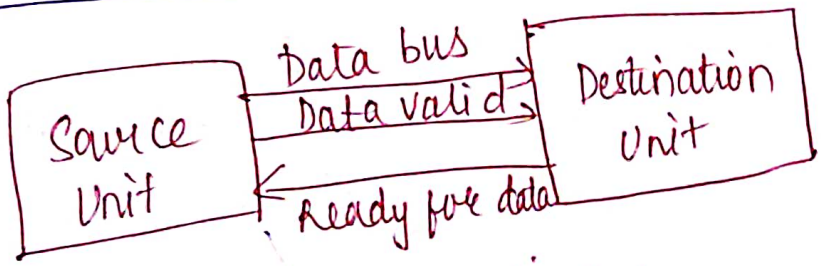
→ two handshaking lines the data valid generated by source unit & data accepted by destination unit

→



(c) Sequence of events

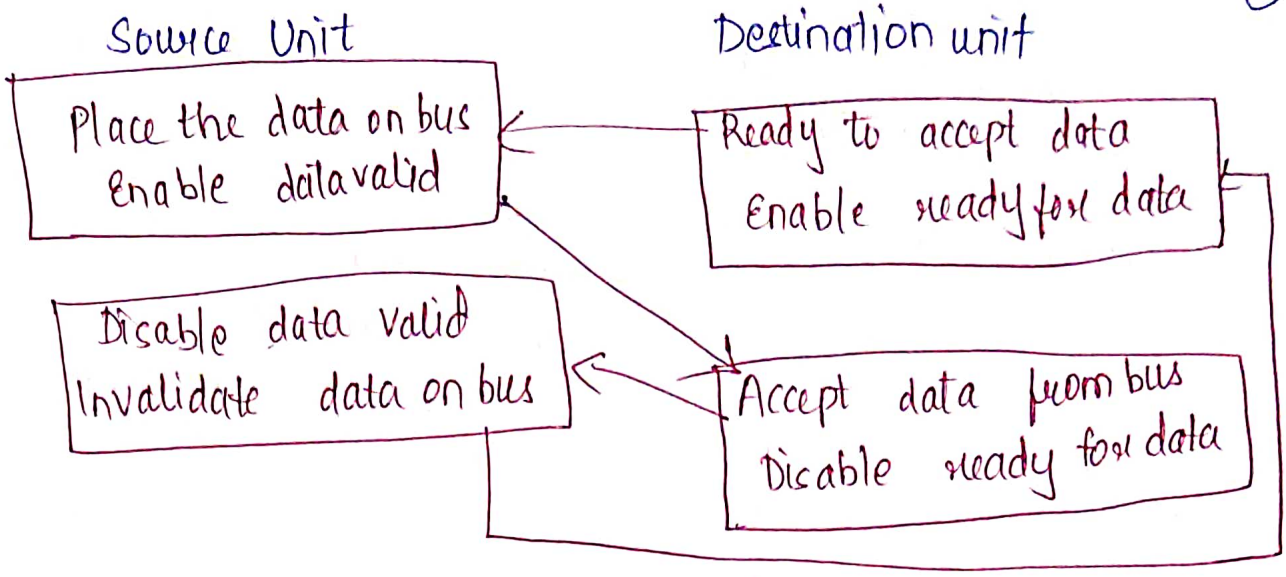
(2) Destination iniated transfer using handshaking :-



(a) block diagram

→ Name of signal generated by the destination unit has been changed to ready for data.

→ Source unit here places data only after signal from destination unit



(b) Sequence of events.

MODES OF TRANSFER :- (Data transfer between CPU & I/O)

(1) Programmed I/O
(2) Interrupt I/O
(3) Direct memory access (DMA)

① Programmed I/O :- This are result of I/O instruction written in the computer program.

→ Constant monitoring by CPU is required for peripheral as each data item is transferred by initiating an instruction in program.

→ Once a data transfer is initiated, the CPU is required to monitor the interface to see when a transfer can again be made.

② Interrupt initiated I/O :- programmed I/O is time consuming as CPU has to stay in program loop until the I/O unit indicates that it is ready for data transfer.

→ To avoid it interrupt facility & special commands to inform the interface to issue an interrupt request signal when the data are available from the device.

→ In meantime CPU can proceed to execute another program. The interface meanwhile keeps monitoring the device.

→ When the interface determines that the device is ready for data transfer, it generates an interrupt request to the computer.

→ Upon detecting the external interrupt signal, the CPU momentarily stops the task it is processing, branches to a service program to process the I/O transfer & then returns to the task it was originally performing.

(3) DMA :- Here the interface transfers data into and out of the memory unit through the memory bus.

→ The CPU initiates the transfer by supplying the interface with the starting address & the no. of words needed to be transferred & then proceeds to execute other tasks.

→ When the transfer is made, DMA requests memory cycles through the memory bus.

→ When the request is granted by the memory controller, the DMA transfers the data directly into memory.

→ The CPU merely delays its memory access operation to allow the direct memory I/O transfer.

→ Since peripheral speed is slower than processor, I/O memory transfers are infrequent compared to processor access memory.

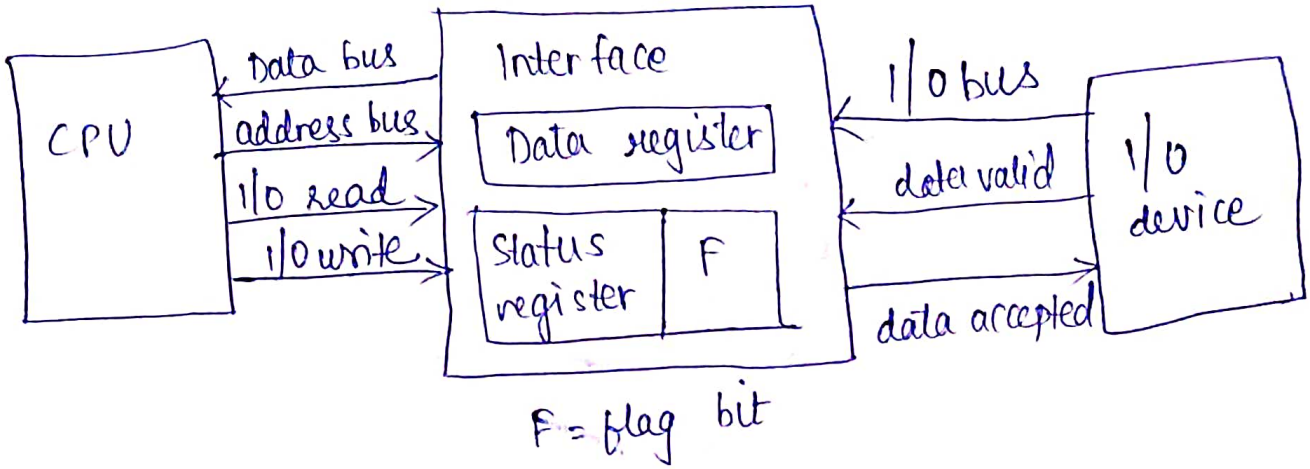


fig: Programmed I/O

→ No direct access between CPU & I/O device

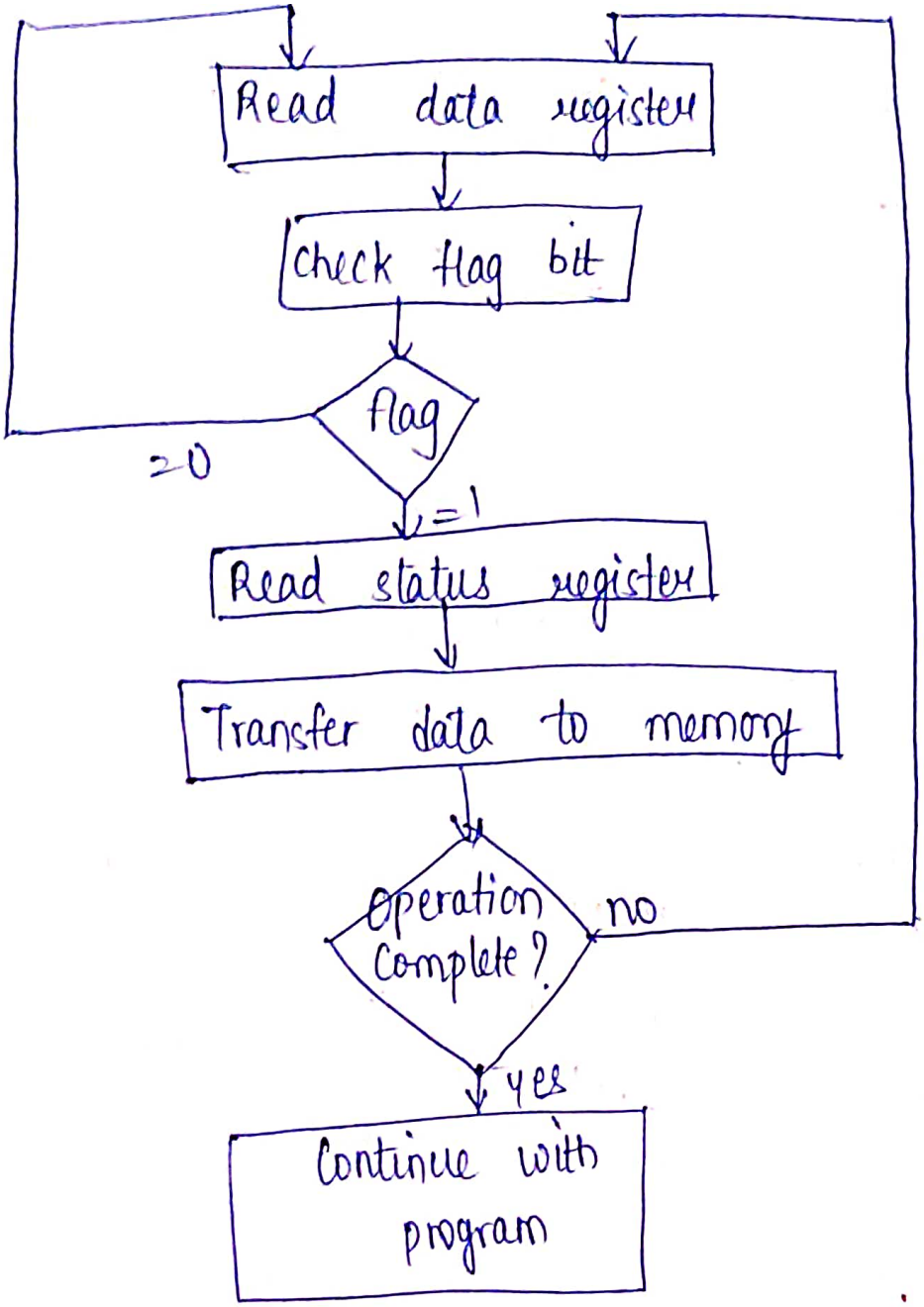
→ the interface sets a bit in the status register that is referred as F or flag bit.

→ The device can now disable the data valid line but it will not transfer another byte until the data accepted line is disables by the interface

→ A program is written to check flag of status register for if a byte is placed in data register by I/O device.

→ Once the flag is cleared, interface disables the data accepted line & the device can then transfer the next data byte.

Example of programmed I/O : H is assumed here that the device is sending a sequence of bytes that must be stored in memory.



Transfer of each byte requires 3 instructions

(1) Read the status register
(2) Check the status of the flag bit & branch to step 1 if not set or to step 3 if set.
(3) Read the data register.
(4) Each byte is read into a CPU register & then transferred to memory with a store instruction.

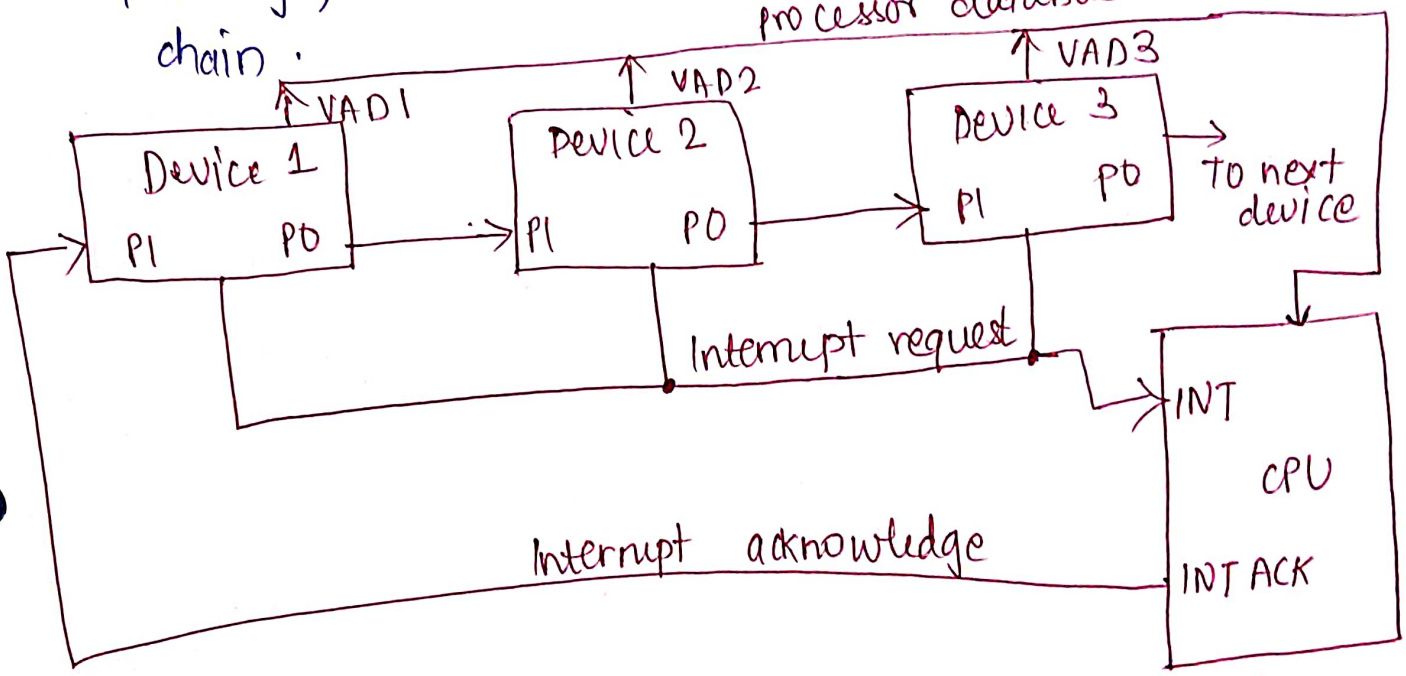
→ A common I/O programming task is to transfer a block of words from an I/O device & store them in a memory buffer.

PRIORITY INTERRUPT:- It determines which interrupt is to be served first when two or more requests are made simultaneously.

→ Higher priority interrupts can make requests while servicing a lower priority interrupt.

→ Daisy chaining priority:- It consist of serial connection of all devices that request an interrupt.

→ the device with the highest priority is placed in the first position, followed by lower priority devices upto the device with lowest priority, which is placed last in the chain.



→ if any device has its interrupt signal in the low-level state, the interrupt line goes to the low-level state & enables the interrupt input in the CPU.

→ When no interrupts are pending, the interrupt line stays in the high level state & no interrupts are recognized by the CPU.

→ The CPU responds to an interrupt request by enabling the interrupt acknowledge line.

→ This signal passes on to the next device through the PO (priority out) output only if device 1 is not requesting an interrupt.

→ If device 1 has a pending interrupt, it blocks the acknowledge signal from the next device by placing a 0 in the PO output.

→ It then proceeds to insert its own interrupt vector address (VAD) into the data bus for the CPU to use during the interrupt cycle.

→ A device with a 0 in its PI input generates a 0 in its PO output to inform the next lower priority device that the acknowledge signal has been blocked.

→ A device that is requesting an interrupt & has a 1 in its PI input will intercept the acknowledge signal by placing a 0 in its PO output

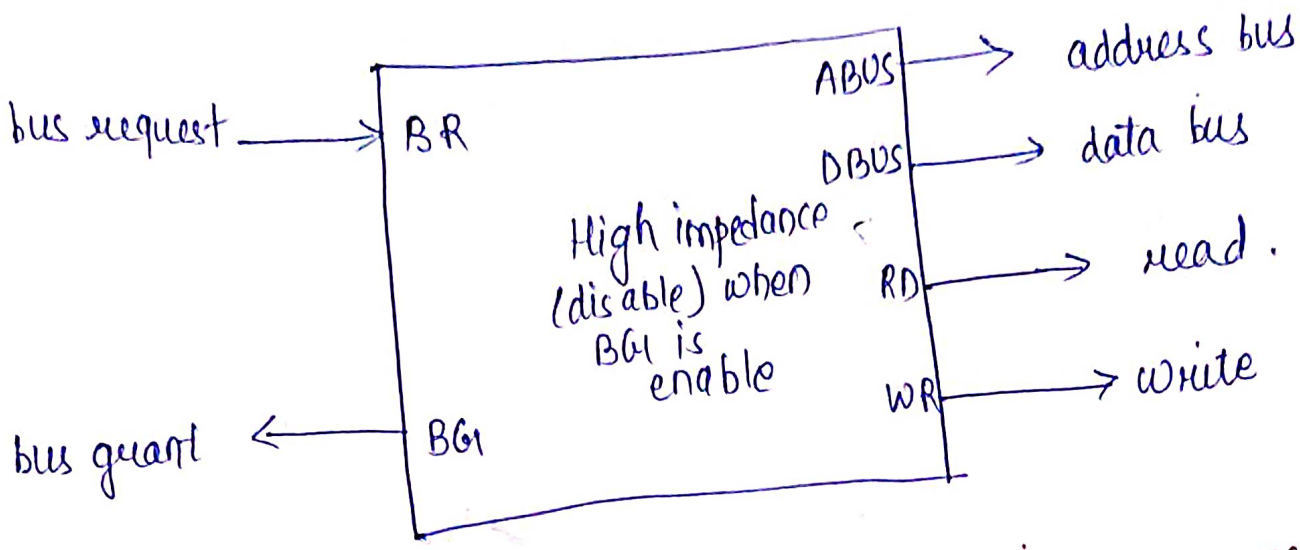
→ If the device does not have pending interrupts, it transmits the acknowledge signal to the next device by placing a 1 in its PO output.

→ Thus the device with PI = 1 & PO = 0 is the one with highest priority requesting an interrupt, & this device places its VAD on the data bus.

→ **Direct memory Access:-** transfer through memory bus.

→ DMA controller interface which allows I/O transfer directly between memory & device freeing CPU for other tasks.

→ CPU initializes DMA controller by sending memory address & the block size (no. of words)



→ Address register and address lines are used for direct communication with the memory.

→ The word count register specifies the no. of words that must be transferred. The data transfer is done between device & memory under control of DMA

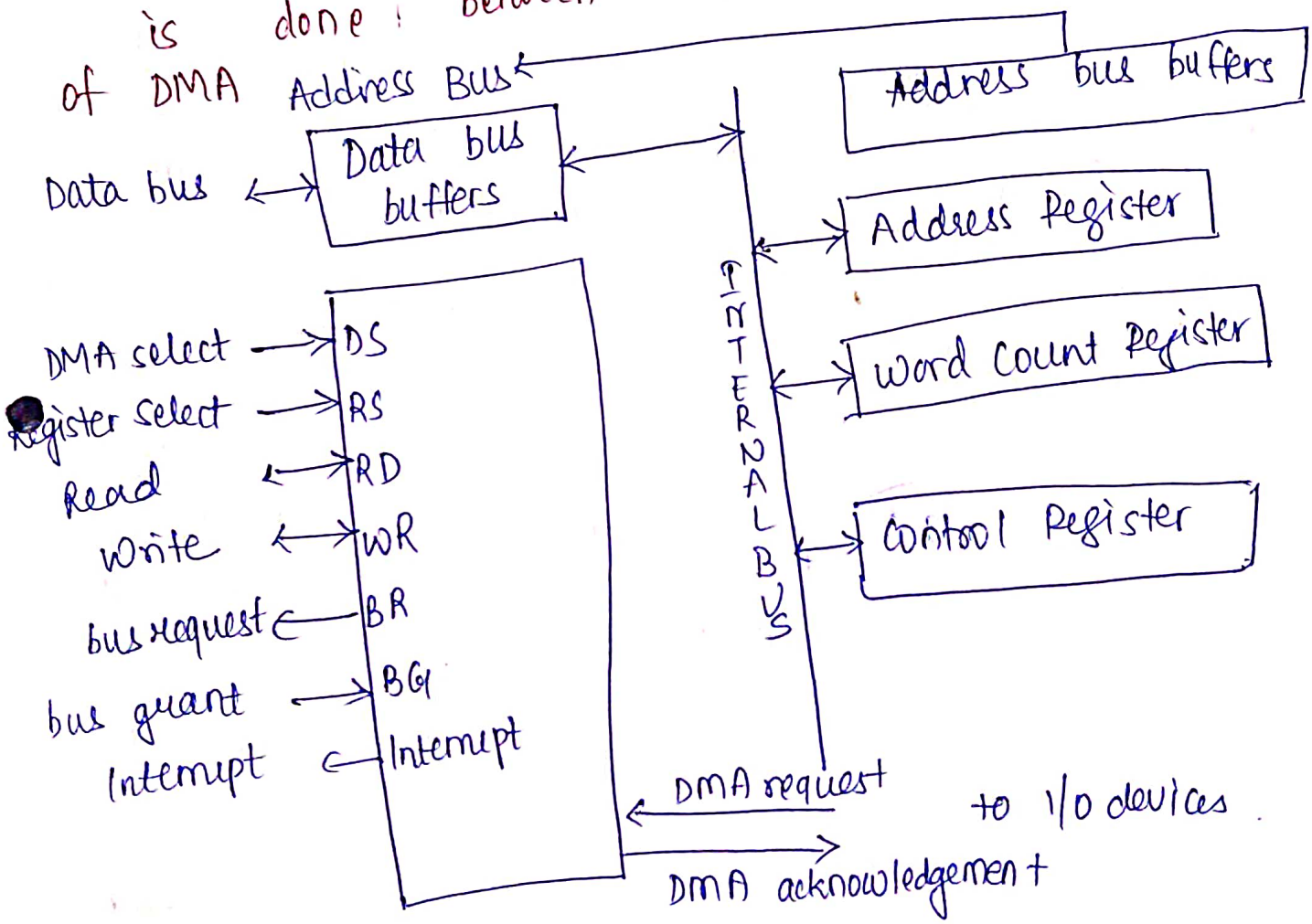


fig. DMA Controller.

→ the register in the DMA are selected by CPU via address bus by enabling the DS (DMA select) and RS (register select) inputs.

→ RD & WR are bidirectional inputs

→ When BG (bus grant) input is 0, CPU can communicate with the DMA registers through the data bus to read from. or write to the DMA registers.

→ When BG = 1, CPU relinquishes the buses & the DMA can communicate directly with the memory by specifying an address in the address bus & activating the RD/WR control.

→ DMA communicates with the external peripheral through the request & acknowledge lines by using a prescribed handshaking procedure.

→ DMA controller has 3 registers
- address register
- word count register
- control register

→ Control Register specifies mode of transfer.

→ Address Register contains address to specify the desired location in memory.

→ Word count register holds the no. of words to be transferred.

→ this register is decremented by 1 after each word transfer & internally tested for zero.

→ CPU can read / write from / into the DMA register

under program control via the data bus.

→ DMA is first initialized by CPU. After that DMA starts & continues to transfer data between memory & peripheral until the entire block is transferred.

→ CPU initializes the DMA by sending the following :-

1) starting address of memory block where data are available (for read) or where data are to be stored (for write)
2) word count which is the no. of words in memory block.
3) Control to specify mode of transfer. eg. RD/WR
4) Starting address stored in address register.