DIRECT MEMORY ACCESS

An instruction that has to transfer input or output data is executed only after the processor determines that I/O device is ready. To do this considerable overhead is incurred if processor has to keep intervening.

∴ To transfer large blocks of data at high speed, a special control unit maybe provided to allow transfer of the entire data block b/w external device and main memory without the continuous intervention by processor ⇒ DMA

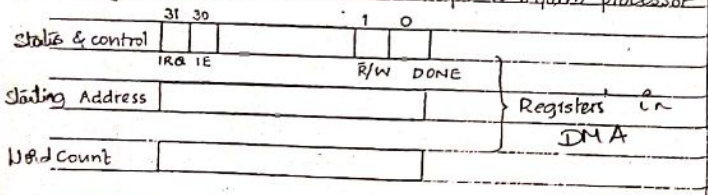
DMA transfers are performed by a control circuit that is part of I/O interface called *DMA Controller*

DMA controller performs operations that would normally be carried out by Processor

To initiate transfer of block of words, processor sends starting address, no. of words in block & direction of transfer.

DMA controller proceeds to do all of it. While processor can do something else

Once job is done, it interrupts an interrupt to inform processor



2 registers are used for storing starting address & word count. The third has status & control flags.

R/W : determines direction of transfer bit=1 Read bit=0 Write

Done : set to 1 if task is completed and ready to receive another command.

IE (Interrupt Enable) : set to 1 after task is completed to let processor know

IRQ : set to 1 when it has requested an interrupt.

DMA controller modes

1) Cycle stealing: Since processor originates most memory access cycles, DMA controller is said to be 'stealing' memory cycles from the processor

2) Block/Burst mode: DMA controller may be given exclusive access to transfer a block of data without interruption

BUS ARBITRATION

Need: A conflict may arise if both the processor and DMA controller or 2 DMA controllers try to use the bus at same time to access main memory. ∴ To resolve this, we need an arbitration procedure

Bus Master: Device that is allowed to initiate data transfers on the bus at any time.

"Bus Arbitration is the process by which next device to become bus master is selected & bus mastership transferred to it"

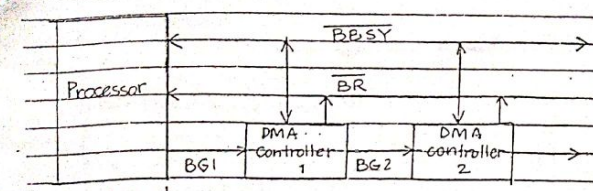
2 approaches

i) Centralized : single bus arbiter performs req'd arbitration

ii) Distributed : All devices participate in selection of next bus master

Teacher's Signature

1. CENTRALIZED ARBITRATION



$\overline{BR}$: Bus Request
$\overline{BUSY}$: Busy Busy
BG : Bus Grant. BG1, BG2

1) Processor is normally the bus master unless it grants membership to DMA controllers

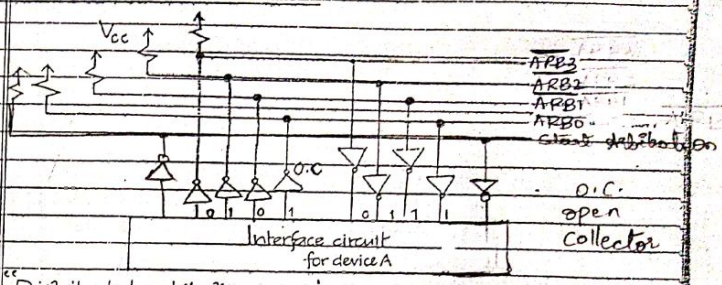
2) DMA controller sends $\overline{BR}$ signal if it needs to be the bus master.

3) The $\overline{BR}$ line is open drain line. When it's activated, the processor activates Bus Grant signal BG1, indicating to DMAC that they may use the bus when it becomes free.

4) This signal is in Daisy chain so if the first one isn't (DMAC) requesting, it's passed on.

5) Current bus master indicates that it's using the bus by activating 'busy' $\overline{BBSY}$ line. to prevent other devices from using the bus at the same time.

2. DISTRIBUTED ARBITRATION



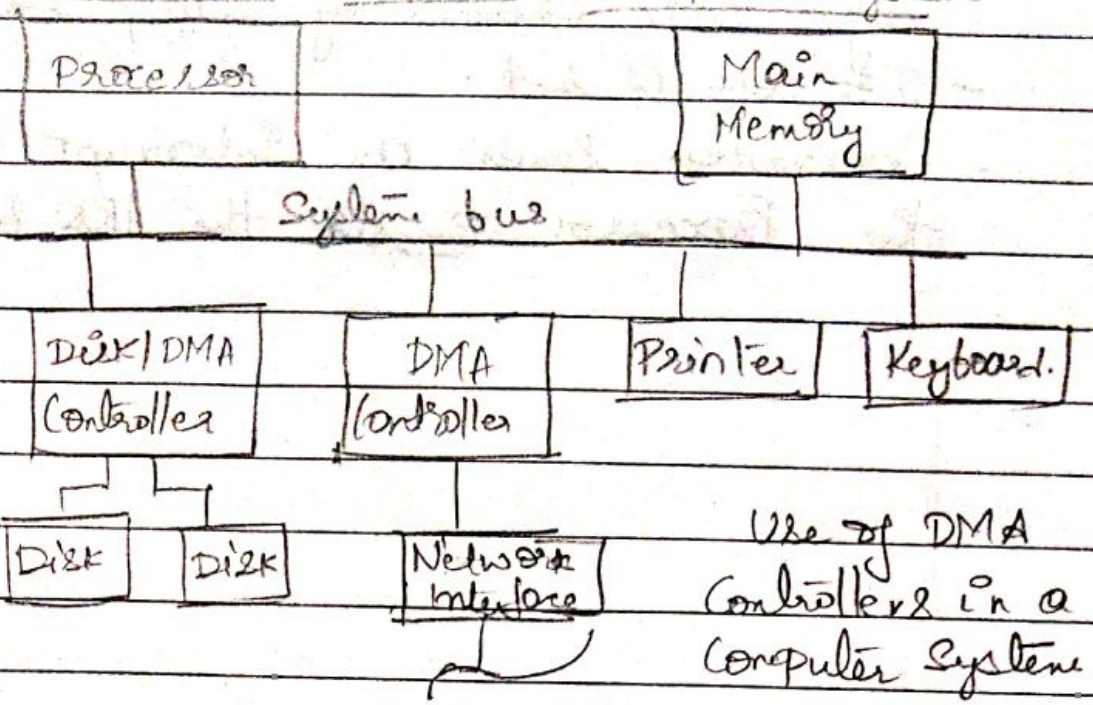
"Distributed arbitration means that all devices waiting to use the bus have equal responsibility in selection."

1) Each device on bus is assigned a 4-bit ID
2) When a device requests the bus, they assert Start-Arbitration signal and place their 4-bit ID no.s on 4 open collector lines.
3) Winner is selected as a result of interaction among the signals transmitted over these lines by all contenders.
4) Net outcome is that the code on lines represents request that has highest ID no. (ID number)

Advantages.
1) Higher reliability as operation of bus isn't dependent on a single device.

Date
Page

* Use of DMA Controllers in a Computer System



- A example of a computer system how DMA (DMAC) Controllers may be used.
- A DMAC connects a high-speed n/w to the Computer bus.
- The disk controller also has DMA capability & provides two DMA channels.
- To start a DMA transfer a block of data from the main memory to one of the disks, a program writes → Address
  → Word count information

  to registers of the corresponding channel of the disk Controller.
- When the DMA transfer is completed, it is recorded in the status and control register of DMA channel by setting the DDNE Bit.
- IE bit is set.
- Controller sends an interrupt signal (request) to the Processors.
  Sets IRQ bit

BUSES

The processor, main memory & I/O devices can be interconnected by means of a common bus whose primary function is to provide a communications path for data transfer.

Bus protocol is the set of rules that govern the behaviour of various devices connected to bus.

Bus lines can be grouped into 3 types: data, address & control lines.

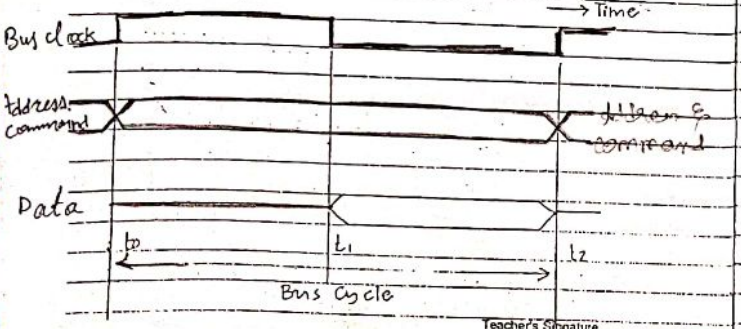
Control signals specify whether R/W. They also convey the timing information. Read when 1 to 1 & write when 1 to 0

Master → initiates data transfers (Processor or DMA c)

Slave → Device addressed by master

Synchronous Bus

In a synchronous bus, all devices derive timing info from a common clock line. Each equally spaced pulses define equal time intervals which constitute a bus-cycle, during which one data transfer can take place.



Timing of an input transfer on a Synchronous bus.

Sequence of events during i/p operation

1) Master places device address on address lines & sends appropriate command on control lines at time t_0.
2) Info travels over bus determined by its physical & electrical characteristics.
3) Clock pulse width ($t_1 - t_0$) must be longer to allow all devices to decode address, control signals, so that the addressed device can respond at t_1.
4) Addressed slave places requested i/p data on data lines at time t_1
5) At t_2, master strobes data & stores it in input buffer.

Similarly, master places o/p data on data lines when it transmits address & command info. At t_2, addressed slave captures data & loads it into buffer.

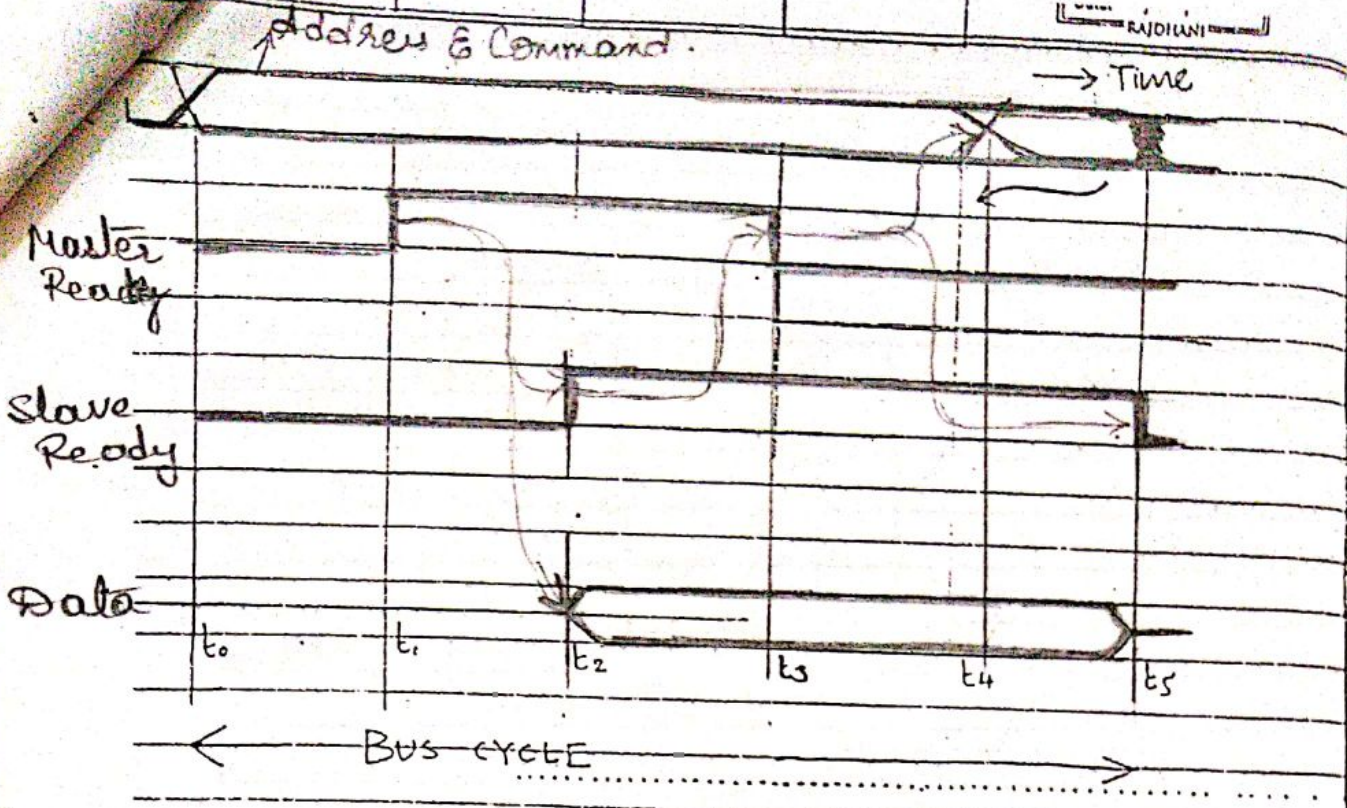
Asynchronous Bus

Based on use of a 'handshake' between the master and the slave.

Principle: The master places the address & command info on the bus. It then indicates to all devices by activating Master-ready line. This causes all devices on bus to decode the address. The selected slave performs read operation & informs processor with Slave-ready line. Master then removes its signal from bus.

Teacher's Signature

Teacher's Signature



t_0 - Master places address & command info on bus & All devices on bus begin to decode it.

t_1 - Master sets the Master-ready line to 1 (informs all I/O devices) Delay $t_1 - t_0$ is to allow for any 'skew' that may occur. The

t_2 - Selected slave, having decoded the address & info performs the read input operation & sets slave-ready to 1. ($t_2 - t_1$ depends on distance b/w master & slave.)

t_3 - Slave-ready signal arrives at the master. After a delay eqvt to max. bus skew, it captures data & drops the Master-ready signal.

t_4 - Master removes the address & command info from the bus.

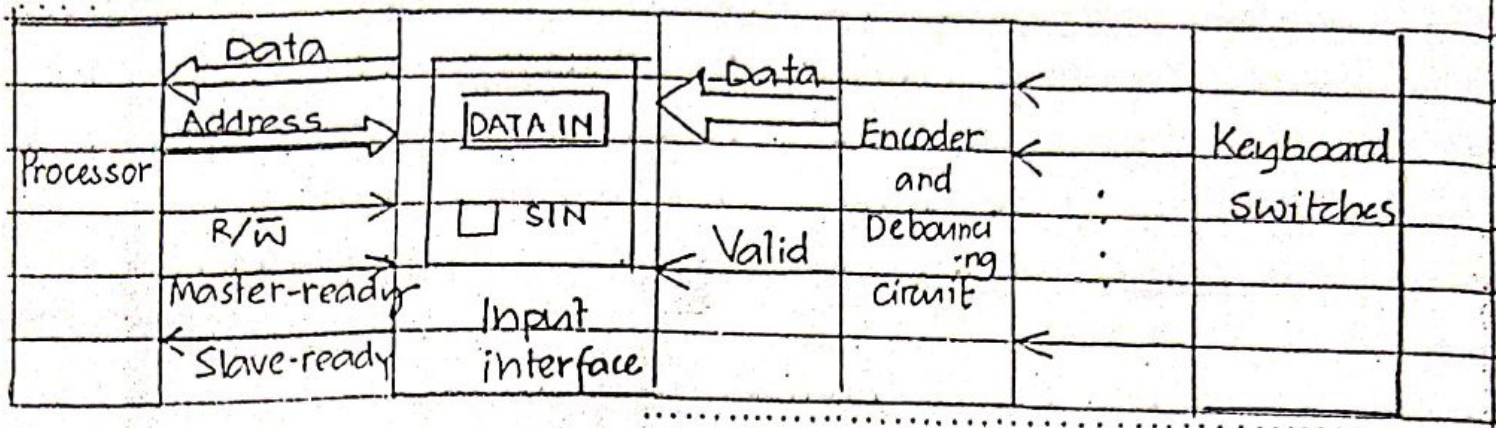
t_5 - When device interface sees MR signal as 0, removes data & SR signal. This completes i/p transfer.

Teacher's Signature

MR - Master Ready
SR - Slave Ready

PARALLEL PORT

Keyboard - Processor Connection



When a key is pressed, switch closes establishing a path for electrical signal

Signal is detected by Encoder that generates ASCII code for corresponding character.

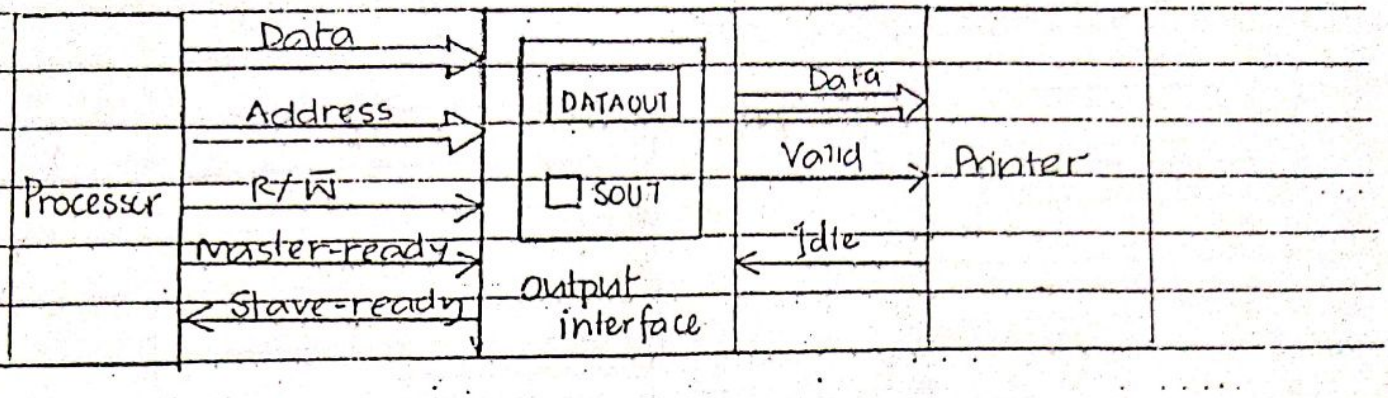
Debouncer ensures I/O routine waits long enough to take one input as one

O/P of encoder is in bits that represent encoded character & control signal called Valid → indicates a key being pressed. ASCII code is loaded into DATAIN & SIN is set to 1

O/P of DATAIN connected to datalines of bus that're turned on when processor issues a read instruction that ~~clears register~~ reads the contents of the DATAIN register.

Interface circuit is connected to asynchronous bus. Master ready and Slave ready signals. R/W - Read & Write transfers.

b) Printer - Processor Connection



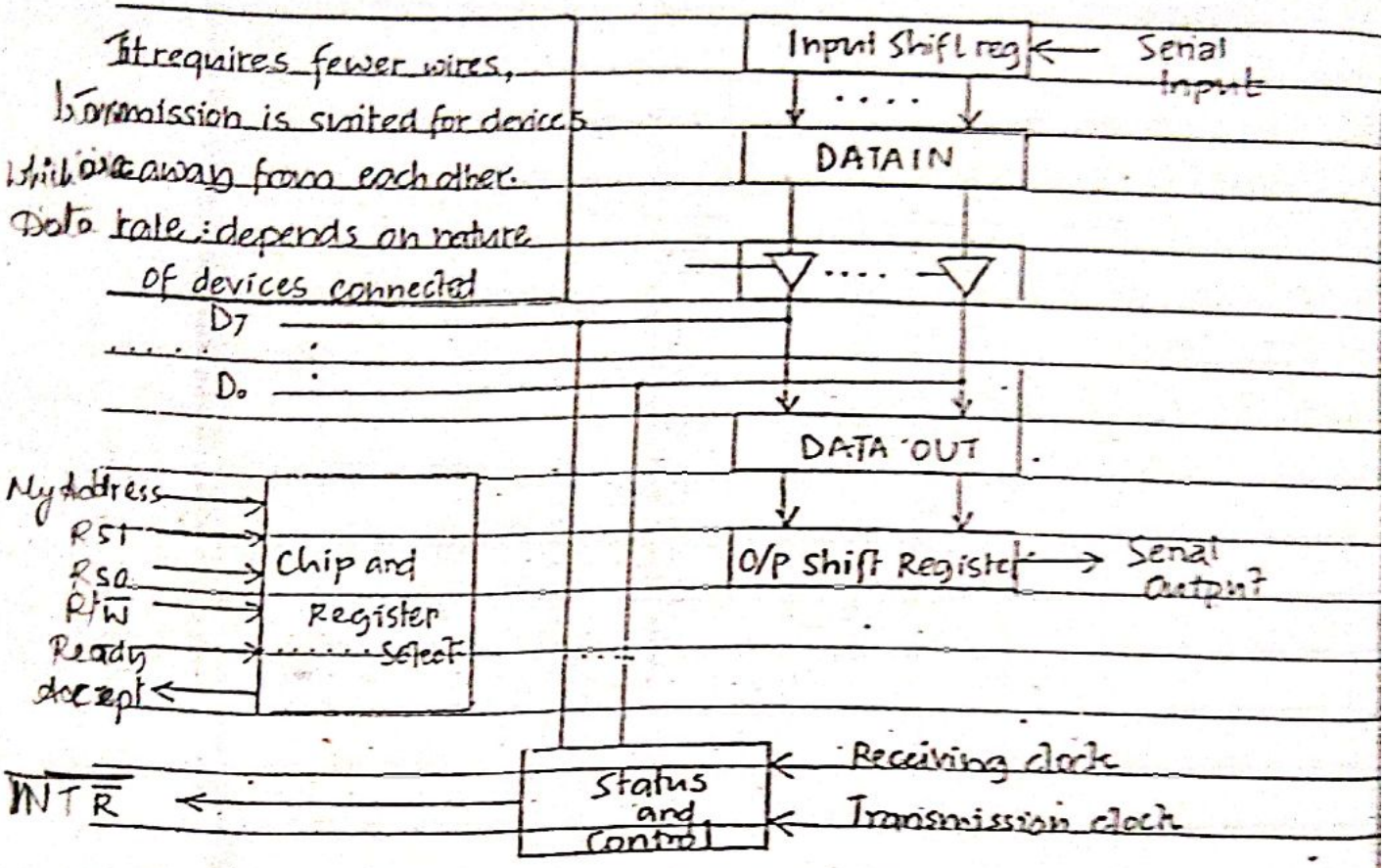
1) Printer operates under control of handshake signals Valid, Idle
2) When it's ready to accept a character, it asserts Idle signal.
3) The interface ckt can then place a new character on the data lines & activate Valid signal
4) In response, printer starts printing new character & negates Idle
5) SOUT flag is set to 1 when printer is ready to accept another character and cleared out when a new character is loaded into DATAOUT by the Processor.

SERIAL PORT

A serial port is used to connect processor to I/O devices that require transmission of data 1-bit at a time.

Key feature : It's interface circuit is capable of communicating bit-serially on device side & bit-parallel on bus-side

It requires fewer wires,
Transmission is suited for devices which are away from each other.
Data rate : depends on nature of devices connected



Includes DATAIN & DATAOUT. The i/p shift register accepts Serial input. When all 8 bits are received, contents are loaded into DATAIN in parallel.

As soon as data are transferred to DATAIN, shift register starts accepting next 8-bit character

Double buffer : Transfer of second character can begin as soon as first character is loaded into DATAIN

Similarly, o/p data in DATAOUT register are loaded into o/p shift register, from which bits are shifted out to I/O device

The Status flags SIN & SOUT have similar function

Teacher's Signature

* Explain DATAOUT, DATAIN, SIN & SOUT.

SPEED, SIZE, AND COST

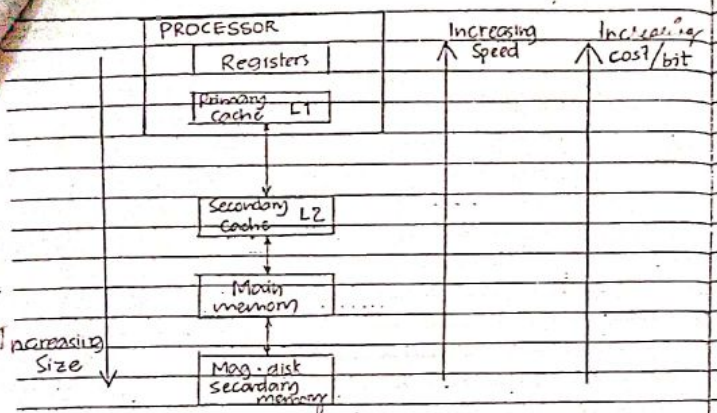
1. A very fast memory can be implemented if SRAM chips are used. But these chips are very expensive. Thus for cost reasons it's impractical to build large memory (Alternatively DRAM is used)
2. Dynamic RAM can be implemented at a reasonable cost but affordable size is still small compared to demand.
3. ∴ Secondary storage (mag. disks) are used to implement large memory. However they're much slower.

CONCLUSION

1. Huge amt. of cheap storage → mag. disks
2. Large & affordable main memory → DRAM
3. Cache memories → SRAM.

(P.T.O)

mag. disks (Magnetic disks)



(Memory Hierarchy)

The entire computer memory can be viewed as hierarchy

Fastest access is to data at Processor Registers.
(top in terms of speed of access but minisicule)

At next level : PROCESSOR cache
It is implemented on processor & holds copies of current data & instructions

- L1 cache → implemented by SRAM [Primary cache]
- L2 cache → Secondary placed b/w L1 & Main memory

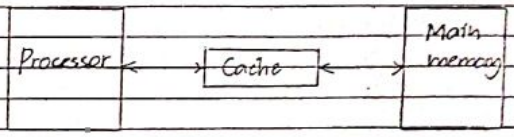
Primary Cache is always located on the processor chip.

At third level : MAIN MEMORY
Implemented by DRAMS like RIMMs etc. Larger than cache significantly but slower

Last level : SECONDARY
- Provide bulk storage
- Very slow access.

Teacher's Signature

CACHE MEMORIES



The speed of main memory is limited by electronic & packaging constraints ∴ An efficient solution is to use 'cache memory' that makes main memory appear faster to processor than it really is.

Operation

"Locality of Reference"

It manifests in 2 ways

i) Temporal : means that a recently executed instruction is likely to be executed again

ii) Spatial : means that instructions in close proximity of recently executed one are likely to be executed soon.

So if active segments like these can be placed in a fast cache memory, then total execution time can be reduced significantly.

Note :-

Load through or early restart :- Word sent to the processor as soon as it is read from main memory.

Write miss :- address word not in Cache during write opn.

Teacher's Signature

- When a Read request is received from the Processor,

→ the contents of a block of memory words containing the location specified are transferred into the cache one word at a time.

→ When the program references any of the locations in this block, the desired contents are directly read from cache.

Read or write hit: Processor issues R/W requests the cache control circuitry determines whether the requested word currently exists in the cache. If it does, the Read or write opn is performed on the appropriate cache location.

Write through Protocol:- The cache location & Main memory location are updated simultaneously.

Write back or Copy back:- Update only the cache location & to mark it as updated with associated flag bit called dirty or modified bit. The main memory location updated, the block containing marked is to be removed from cache.

Read Miss:- When the addressed word in the Read opn is not in the cache.

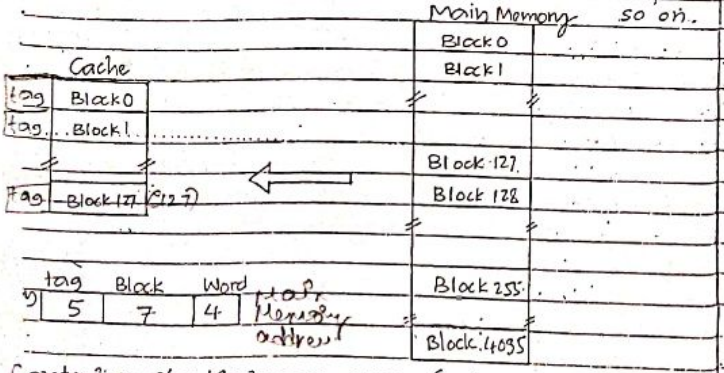
MAPPING FUNCTIONS

"The correspondence between the main memory blocks and those in cache is specified by mapping function"
It decides which block should contain what and which block needs replacing.

Direct Mapping

Simplest way to determine cache locations in which to store memory blocks. In this technique, block j of main memory maps onto block j modulo 128 of cache.

Blocks 0, 128, 256 etc of main memory → Block 0 of cache
1, 129, 257 etc → Block 1 and so on.



tag	Block	Word
5	7	4

Main Memory address

Contention of a block may arise for long programs which is solved by overwriting old block.
Placement determined by memory address.
Contention may arise even when the cache is not full.
eg:- Program starts with block 1 & continue to block 129 possibly after a branch.
Each of these must be transferred to block 1 position in cache.

Teacher's Signature

Memory address can be divided into 3 fields.

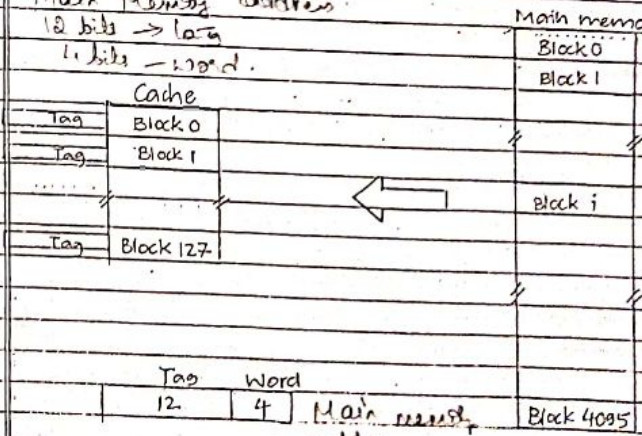
- Lower 4 bits → Select one of 16 words in a block
  7-bit field → determines cache position
  5-bits → tag bits
- CONTENTION PROBLEM
- EASY TO IMPLEMENT BUT - NOT VERY FLEXIBLE

Advantage | Disadvantage

b) Associative Mapping

Much more flexible. Main memory block can be placed into any cache position.

- In this case, 12 tag bits are req^d to identify a resident block.
- It gives complete freedom in choosing the cache location in which to place the memory block.
- Space in cache can be used more EFFICIENTLY
- COSTLIER. Needs associative search & Replacement algo.



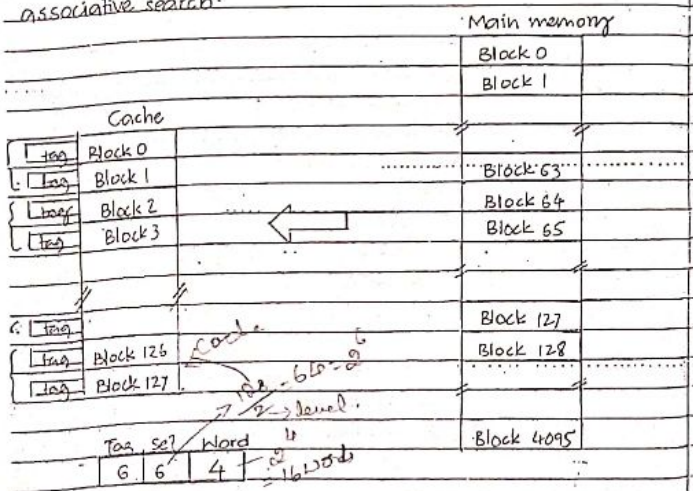
Tag	Word
12	4

Main memory address

Teacher's Signature

Associative Mapping

It's a combination of direct- and associative-mapping
blocks of cache are grouped into sets & the mapping allows a block to reside in any block of specific set
→ Contention problem eliminated
→ Hardware cost reduced by decreasing size of associative search.



Having 64 sets ⇒ 6 bits field for tag
One control bit → VALID BIT provided to indicate if block contains valid data.
Cache-coherence problem.

REPLACEMENT ALGORITHMS

In cache memories where cache controller has to decide which instructions to keep- that'll boost processor speed the best, several replacement algorithms come into picture.

i) When a block is being overwritten, it's sensible to overwrite the one that has gone the longest time unused. It's called LRU (Least recently used) block & the technique is LRU replacement algorithm.
- Cache controller tracks references to all blocks
- Extensively used.
- Can even lead to poor performance often
- ∴ performance is improved by a small amt. of randomness in decision.

ii) Another reasonable rule is to remove "oldest" block from a full set.
Since it doesnot take into account the recent pattern of access, it's generally not very effective

iii) Simplest: Randomly chose a blocks to overwrite.
QUITE EFFECTIVE too.

:RFORMANCE CONSIDERATIONS

key factors in commercial success of a computer are [illegible]ormance and cost. A common measure of success is :e/performance ratio. I) Interleaving
2) Hit Rate & Miss Penalty (3) Caches on Processor chip.

HITRATE and MISS PENALTY

An excellent indicator of effectiveness of particular implementation of memory hierarchy is the success rate of accessing info at various levels.

HIT RATE: No. of hits stated as a fraction of all attempted accesses.

MISS RATE: No. of misses stated as a fraction of all Attempted accesses.

MISS PENALTY: It is the extra time needed to bring desired information into cache after a failed attempt (miss).

* It can be reduced by efficiently implementing memory hierarchy.
* Also if load-through approach is used.

How to improve hit rate?

Increase size of cache → but costly

Increase only block size → miss penalty increases

∴ Block sizes that're neither very small nor very large (16-128 bytes) give best results.

Average access time experienced by Processor:

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

h - hit rate
M - Miss penalty.
C - time to access info in the cache.

Teacher's Signature

Caches on Processor chip

If both L1 & L2 caches are used;

a) L1 should be designed to allow fast access as it'll have large effect on clock rate

b) L2 cache can be slower but larger to ensure high hit rate.
↳ SRAM chips.

AVERAGE ACCESS TIME

$$t_{avg} = h_1C_1 + (1-h_1)h_2C_2 + (1-h_1)(1-h_2)M$$

where

h_1 → hit rate in L1 cache
h_2 → hit rate in L2 cache
C_1 → time to access info in L1
C_2 → time to access info in L2
M → time to access info from main memory.

eg:- Two separate caches for instr & data
68040, P-3 & P-4 (Pentium)
Single for data & instr — ARM 710T

→ A practical way to speed up access of cache is to access more than one word simultaneously.

Teacher's Signature

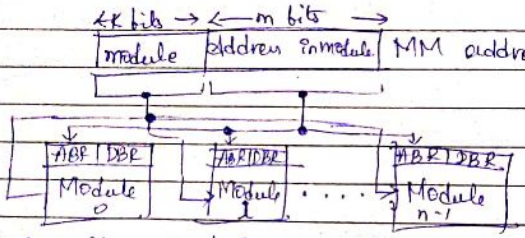
Interleaving

- If the main memory of a computer is structured as collection of physically separate modules, Each module has its own
  1) ABR (address buffer register)
  2) DBR (Data buffer Register)

So, memory access ops may proceed in more than one module at the same time.

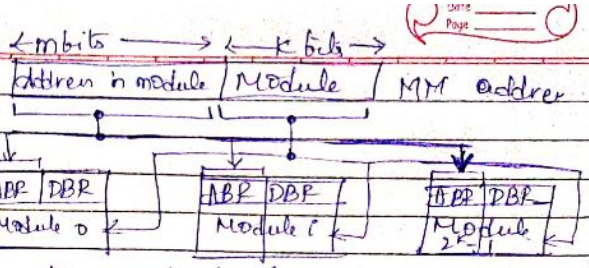
→ Thus, the aggregate-rate of transmission of words to/from the main memory can be increased.

Methods of address layout:



Consecutive words in a module.

- High order k bits of memory address determine the module.
- Low order m bits – word within module.
- When a block of words is transferred from MM to Cache, only one module is busy at a time.



→ Consecutive words in Consecutive module.

- Consecutive addresses can be located in consecutive modules.
- While transfering a block of data several memory modules can be kept busy at the same time.

→ Faster access

- Higher avg utilization of the memory system as a whole.