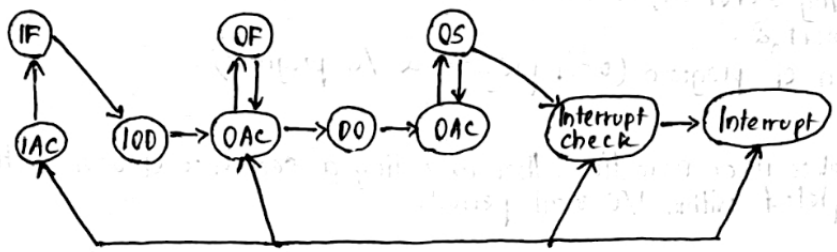
- The interfinder program is a part of the operating system which determines the nature of interrupt and performs whatever actions are needed.
- The handler also determine which I/O module generated in the interrupt and may branch to a program ISR (Interrupt service Routine). That will write more data out to that I/O module.
- When the interrupt and the routine is completed the processor can resume the execution of the program at the point of interruption.

- To accomodate an interrupt, interrupt cycle is added to the instruction cycle.
- In the instruction cycle, the processor checks to see if any interrupt has occured, indicated by the presence of interrupt signal (INTR).
- If no interrupt is pending, the processor proceed through the ~~phase~~ Fetch cycle and fetches the next interruption. If interrupt is pending, then the processor does the following:
  i). It suspends the execution of current program, save the content of the register and address of the next instruction in the stack.
  ii). Then it sets the PC to the starting address of ISR for the execution of handler program.

e.g:
```
0100: 0000  MOV AX, 8000H
0100: 0003  MOV BL, 02H
0100: 0006  DIV BL  ---> 1) Divide error
0100: 0007  HLT          INT 0
```

PC → F400: 0174
BIOS DI
INT 00H
IRET
} ISR

→ Instruction cycle with Interrupt:



Revised instruction cycle state diagram that includes interrupt cycle processing.

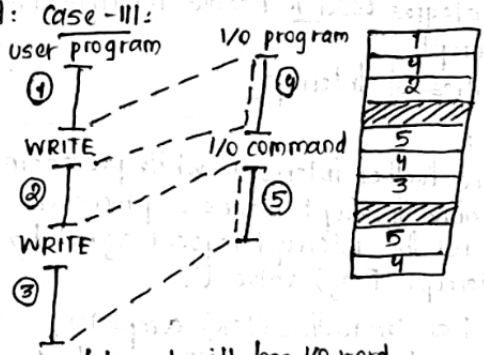
→ Interrupt with short I/O word & long I/O word: Case-III:

User Program
Code seg. 1: Statement 1, Statement 2, ..., Statement n

WRITE
Code seg. 2: Statement 1, Statement 2, ..., Statement n

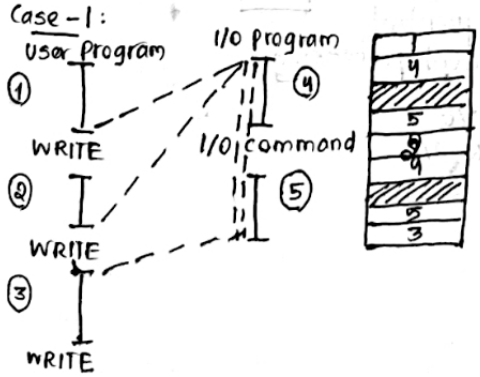
I/O Program
Code seg 3: Statement 1, Statement 2, ..., Statement n

I/O Command
Code seg 4: Statement 1, Statement 2, ..., Statement n



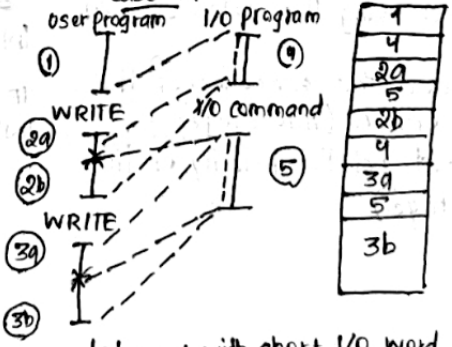
Interrupt with long I/O word

Case-I:



No Interrupt

Case-II:



Interrupt with short I/O word

- User program which has two write commands to respond to I/O program. User program consists of 3 code segments and two write commands. I/O program consists of three segments:
  i). sequence of instructions labelled as 4
  ii). Actual I/O command
  iii). Sequence of instruction to complete the operation labelled as 5.

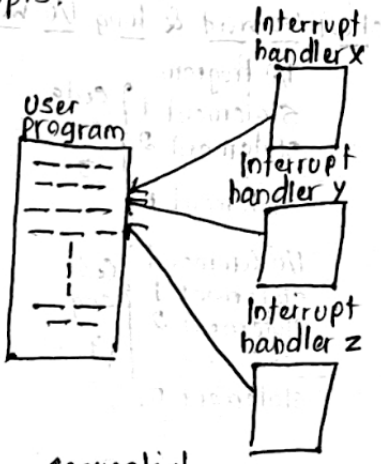
Case-I (normal operation, no interrupt):
- If the I/O program being executed within the write command, then there is no interrupt.

Case-II
- With the interrupt the processor is busy in executing other instructions, while an I/O operation is in progress.
- The I/O program which is invoked consists of preparation of code and actual I/O command.
- After the execution of few instruction in I/O program, the control is transferred to user program. The point at which interrupt occurs is denoted by a cross. The segment of code 2 is interrupted.
- Assume that the time required for the I/O operation is relatively short or less than the time required to complete the execution of the instructions in the user program. Then a portion of code 2 executes and interrupt occurs.
- After the interrupt is being serviced, the execution of user program resumes with the remainder of code segment 2.
- The sequence of execution of program (user program & I/O program).

Case-III :
- When the I/O operation takes much more time than executing a sequence of user instructions, the user program is completed within I/O wait period.
- ~~Use se~~

→ Multiple Interrupts:
Two approaches/strategies used to handle multiple interrupts:
  i). Disable Interrupt
  ii). Define priorities of Interrupt

i). Disable Interrupt:
- Processor will ignore further interrupts while processing one interrupt. Disable interrupt means a processor can and will ignore the interrupt request signal, by making the IF (Interrupt Flag) value 0.
- After the Interrupt handler routine (ISR) completes, interrupts are enabled by setting the IF value to 1. Interrupts are enabled before resuming the user program, and the processor checks to see if additional interrupts have occured (by checking the status of INTR bit).
- It is very nice and simple and the execution of the interrupt is in sequential order. (Advantage)
- It does not take into account relative priority or time critical leads. (Disadvantage)



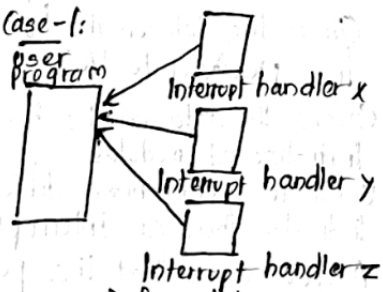
sequential Interrupt processing

IF 1/0
INTR 1/0

iii). Define Priorities of Interrupt:

To overcome these difficulties, another approach is used where the priority among the interrupts must be defined and it allows an interrupt of highest priority to be responded than the lower priority interrupt.

Case-I:

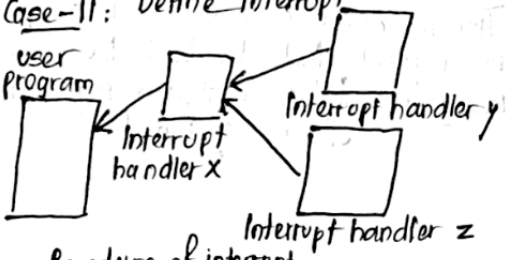


→ Interconnection Structure:

Computer consists of 3 main modules:

i). Memory
ii). I/O module
iii). CPU

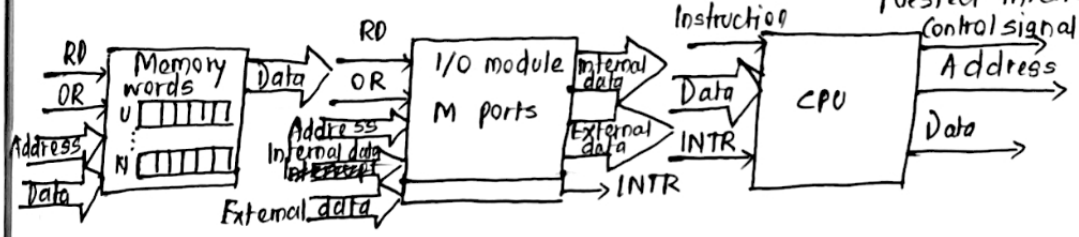
Case-II: Define Interrupt



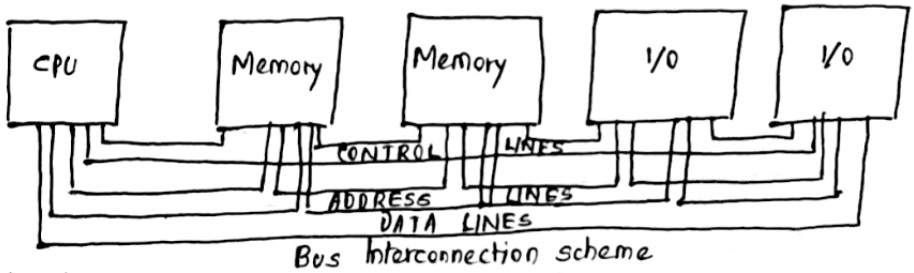
Procedure of interrupt

x < z < y

Nested Interrupt



i). Bus Interconnection structure:



Bus Interconnection scheme

System Bus:

i). Address Bus: Carries address information of memory / I/O.
ii). Data Bus: Carries the data
iii). Control Bus: Carries control information

Types of Control signal required for different operations:

i). Memory Read (MEMRD)
ii). Memory write (MEMWR)
iii). I/O Read (IORD)
iv). I/O write (IOWR)
v). RESET
vi). CLK
vii). INTERRUPT SIGNAL (INTR)
viii). INTERRUPT Acknowledgement (INTA)
ix). Bus Request
x). Transfer Ack
xi). Bus Grant

- A computer consists of a set of components/modules of three basic types: memory, I/O and CPU that communicate with each other. So a computer is a network of different modules. Thus there must be paths for connecting these modules. The collection of paths connecting the various module is called as interconnection structure. The design of this structure depends on the exchanges that must be made among modules.

i). MEMRD: Causes data on the bus from the addressed location (data transfer from memory to processor).
ii). MEMWR: Causes the data on the data bus to be written into the addressed location (data transfer from processor to memory).

iii). IOWR: Causes the data on the bus to be output to the addressed (I/O) port. (Transfer from processor to I/O).

iv). IORD: Causes the data on the bus from the addressed I/O port. (Transfer from I/O to processor)

v). RESET: Initialize all modules

vi). CLK: Used to synchronize different operations.

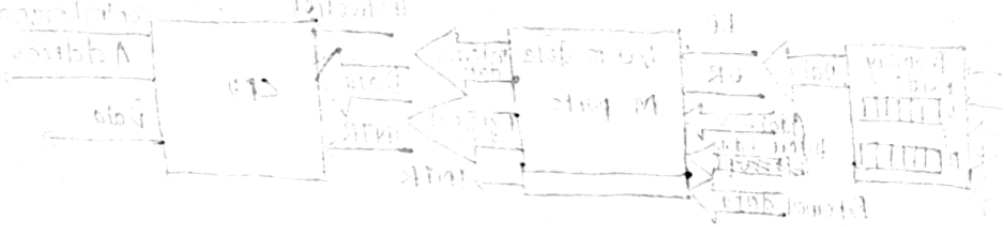
vii). INTR: Indicates that an interrupt is pending.

viii). INTA: Acknowledges that the pending interrupt has been recognised by processor.

ix). Transfer ACK: Data has been accepted from or placed on the data bus.

x). Bus request: Indicates that I/O module needs to gain the control of the bus (system bus).

xi). Bus grant: Indicates that a requesting module has been granted by the processor control of the system bus.



CH: CACHE MEMORY

→ Characteristics of Memory:

i). Location
ii). capacity
iii). Unit of transfer
iv). Access Methods
v). Performance
vi). Physical type
vii). Physical characteristics
viii). Memory organization

i). Location: refers to whether memory is external or internal to the system.

ii). Capacity: Amount of information to be stored. For external, data is stored in bytes. For internal data may be stored in the form of bytes / words.

iii). Unit of Transfer: No. of electrical lines into or out of the memory module. For main memory this the number of bits read out of or written into the memory at a time. For external memory, data are often transferred in much larger units referred as blocks.

iv). Access method: It is the method of accessing unit of data.

a) sequential: Memory is organised into units of data called as records. Access must be made in a specific linear manner. Access time is variable and depends on the current location & previous location. e.g: Magnetic tape

b). Direct: It involves shared read/write mechanism. Individual blocks/records have a unique address based on the physical location. Access time is variable.
e.g: Disk cleaning

c). Random: Each addressable location in memory has a unique addressing mechanism. The Access time of a given location is independent of the sequence of prior access, thus any location can be selected at random and directly accessed and addressed.
e.g: main memory, cache system

d). Associative: This is a Random Access type memory that enables one to make a desirable bit location within a word for a specific match. Access time is independent of the sequence of prior access.
e.g: cache memory

v). Performance: Three parameters to measure the performance of a memory are:

a). Access time (Latency): . Time taken to perform read/write operation for RAM.
. For non-RAM, it is the time taken to position read/write mechanism at the desired levels.

b) Memory cycle Time: . For RAM, it is the time required to access the data (access time + any additional time required to get the data into the system (data) bus)

c). Transfer rate: . This is the rate at which data can be transferred into/out of the memory unit.
. For RAM, transfer rate = $\frac{1}{\text{cycle time}}$
For non-RAM, transfer rate: $T_n = T_A + \frac{n}{R}$

$\Rightarrow \boxed{R = \frac{n}{T_n - T_A}}$

n = no. of bits to be transferred
T_n = avg. access time to read n-bit
T_A = Access time

vi). Physical Type: a). Semiconductor: RAM, ROM
b). Optical: CD, DVD
c). Magnetic: Magnetic tape, Magnetic disc

vii). Physical Characteristics:

a). Volatile / non-volatile:

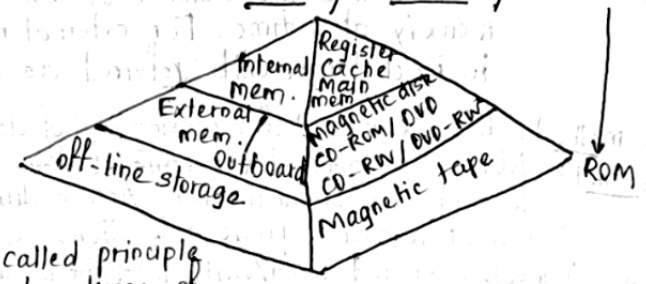
viii). Organisation : • The arrangement of bits in a memory.
• In RAM, the organisation of memory is a key design issue.

→ Memory Hierarchy: Three main constraints in designing a memory unit:
i). How much : Refers to capacity of the memory
ii). How fast : Refers to the access time
iii). How expensive : Refers to the cost of each bit of the memory.

• The dilemma associated with most of the memory technologies:
i). Greater capacity, but smaller cost per bit.
ii). Faster Access bus but greater cost per bit.
iii). Greater capacity but slower access time.

• The way of these dilemmas is not to rely on a single memory technology, but employ a memory hierarchy.

Memory Hierarchy:



As we go down in the hierarchy:
i). Cost per bit decreases
ii). Capacity increases
iii). Access time increases
iv). Frequency of access of memory by the processor decreases.

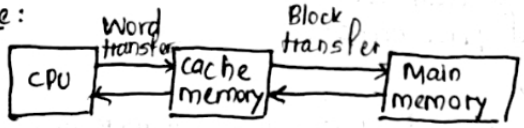
• The validity of (iv) is based on a principle called principle of locality / locality of references. There are two types of principle of locality:

i). Temporal reference: If an item is accessed, we tend to be accessed or referenced again soon.
e.g: operation based on iterative loops and subrouting.

ii). Special reference: If an item is accessed/referenced, the items whose addresses are closed by tend to be referenced soon.
e.g: operation based on array and tables.

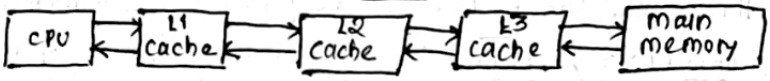
→ Cache memory Principle:

i). Single cache:

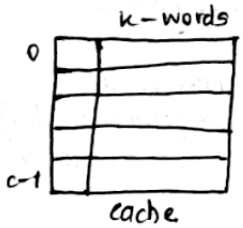


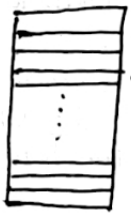
(a)

ii). Multiple levels of cache:



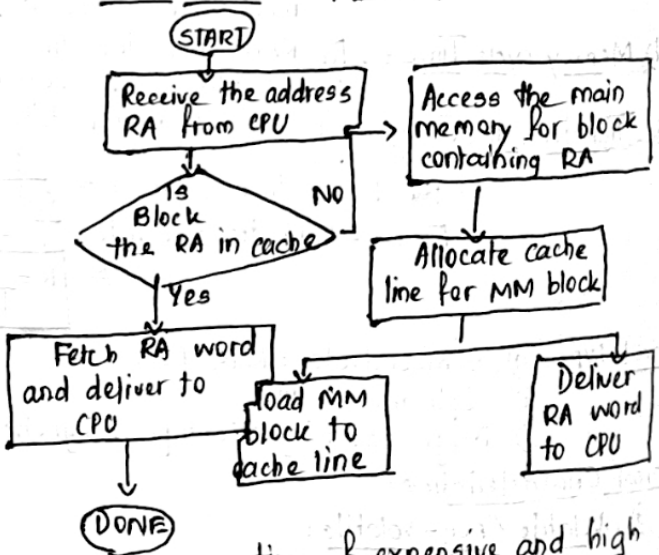
(b)





A structure of cache memory/ main memory

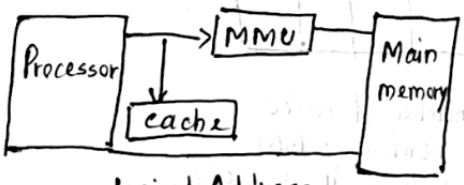
Cache Read Operation



• Cache memory is designed to combine the memory access time of expensive and high speed memory with a larger memory size of less expensive and lower speed.
• The cache contains a copy of portions of main memory. When the processor search for a data word to read, a check is made to determine if the word is in the cache. If the data word is in the cache then it delivers to the CPU.

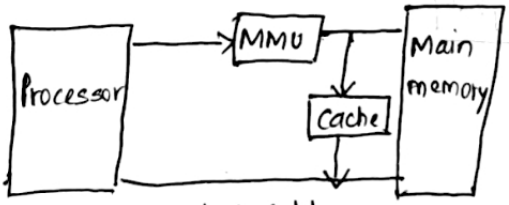
- Fig(b) shows multiple uses of cache, where cache is divided into 3 levels. L1 cache is nearer to processor, then L2 and L3. So L1 cache is smaller but faster than L2 cache. Similarly L2 cache is faster than L3 cache and so on.
- The processor first generates the RA (Read Address) of the word to read it. If the word is present in the cache, then the word must be delivered to CPU, and the process is called as 'Hit'.
- If the word is not available (Miss) in the cache, then the block containing the word is loaded into the cache and delivered to CPU.
- In some organisation, the last two operations (load mm block to cache and deliver the RA word to the processor) occur in parallel because the cache is connected to the processor via data address and control lines. The data and address lines are also attached to the data and address buffer of the system bus from which main memory is reached.
- When cache hit occurs, these buffers are disabled. When a cache miss occurs, data is loaded into the system bus and the data returned from main memory are transferred through data buffer to both cache and processor.
- In other organisation, when a cache miss occurs, the desired word is first read into the cache and delivered to the processor from the cache.

→ Cache Address :



MMU : Memory management unit

Logical Address



Physical Address

Q : Multiplication of 05H, 04H without using arithmetic instruction.

Sol^n:
```
MOV AL, 05H
SHL AL, 02H
HLT
```

Q: Div two 16-bit numbers without DIV Instruction.

Sol^n:
```
    MOV AX, [5000H]
    MOV BX, [5002H]
L1  SUB AX, BX (X)
    DEC BX
    JNZ L1
```

```
    MOV AX, [5000H]
    MOV BX, [5002H]
    MOV DX, 0000H
L2: CMP AX, BX  → CMP BX, DX
    JC  L1        JZ L3
    SUB AX, BX
    INC DX
    JMP L2
L1: MOV [5004H], AX ; R
    MOV [5006H], DX ; Q
    HLT
```

→ Direct Mapping:

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

Soln: Main memory size = 64 words

No. of bit used to access 1MM = $\log_2(64) = 6$ MM bits

Block size = 4 words

No. of blocks = $\frac{64}{4} = 16$

No. of bits used as word instruction = $\log_2 4 = 2$ 2 bits: word identifier

PA of 1 data word = 6 bits 4 bits: block identifier

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 = $\frac{16}{4} = 4$

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

0				
1				
2				
3				

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

i: line number of cache

j: block number of MM

m: no. of lines in a cache

$i = j \% m$