EX NO:1 DATE:	TO RECOGNIZE VARIOUS COMPONENTS OF PC INPUT OUTPUT SYSTEMS PROCESSING AND MEMORY UNITS

Aim: To Recognize Various Components of PC- Input Output Systems Processing And Memory Units

Theory:-

The basic components of a PC are

1. Input Unit 2. Output unit 3. Memory unit4. Control unit 5. Arithmetic logic unit

Input Unit:-

It is the unit through which data/instructions can be entered into thecomputers.

e.g. Keyboard , Mouse etc.

Output Unit:-

It is the unit by which we can get output from the entered input from thecomputer.

e.g. Monitors ,Printer, Speaker etc.

Memory Unit:-

It stores the information by providing facility to the CPU actively byproviding necessary data to CPU.

Memory Units are of two types

1. Primary memory 2. Secondary memory.

Primary Memory: - It is also of two types: i) RAM ii)ROM

RAM: - It can be randomly accessed.

Memory is temporarily used because when the power goes off ,allthe data stored in it are erased.So it is volatile in nature.It can be read andthe data can be written into it.

ROM:- It is also randomly accessed. It is only readmemory unit.It is nonvolatile in nature.

It can be read only but data's can't be written into it.

ALU:- It performs arithmetic operation like addition,subtraction etc andlogical operation like AND,OR,NAND etc. It works in electronic speed butthe device attached to it works in low speed.That's why processor canhandle all the peripheral devices at a time. IT establishes well coordination between other four functional units.

CU: - It is the unit which controls the flow of information through theprocessor and coordinate the activities the activities of other unit which arewithin it.So it is the brain within

the brain as it controls what happens inside theprocessor.It generate timing signal and control signal for well coordination.

Configurtion of a PC.

S.NO	Name of Component	Name of Manufacturer	Capacity/frequency interface
1	Processor		
2	MotherBoard		
3	RAM		
4	HDD		
5	FDD		
6	DVD		
7	TFT/CRT		
8	Keyboard		
9	Mouse		
10	UPS		
11	Cabinet		
12	Speaker		

Processor: - The microprocessor accepts inputs from the user in theform of data and instruction. It process the information and instruction andthen send the processed information to the output device.

Motherboard: - It is the main circuit of PC.It contains the interface forthe microprocessor,BIOSMemory and storage device need to controlperipheral devices such as monitor,keyboard,mouse etc.

RAM: - It stores data temporarily.So it is called volatile.

HDD: - IT is a secondary storage device for permanent data storagedevice i.e. placed in the system.It is similar to human brain where all thepast to present events are stored.

DVD RAM: The Digital versatile disc stored digitally.

A DVD writer is a DVD player as well as a writer.

FDD: It is a n external storage device. It is magnetic round disc enclosedin a plastic jacket. Today we have double size high quality to density disk with 1.44 MB ofsize.

Keyboard:- It is a primary input device of the PC similar to type writer.

Mouse:- It is used to point to the desired position in the computer. It isalso an input device.

UPS: - It is the device that produce supply to the PC.It provides all the time of power cut.So we can save the current data andshut down properly.

Speaker: It is an output device through which CPU can produce soundfor the user.

TFT monitor:- It is an output device through which we can read data.

Cabinet: - outer covering of CPU.

Conclusion:-

Hence theReorganization of Various Components of PC- Input Output Systems Processing And Memory Units has been done successfully.

EX NO:2 DATE:	TO UNDERSTAND HOW DIFFERENT COMPONENTS OF PC ARE CONNECTED TO WORK PROPERLY ASSEMBLING OF SYSTEM COMPONENTS

Aim: To understand different components of pc are connected work properly

Procedure

Step 1: Procuring Parts

1. Processor (CPU)
2. Computer Case
3. Optical Drive (DVD RW and SATA capable)
4. Memory (RAM)
5. Power Supply
6. SATA Cables
7. Motherboard (SATA Capable)
8. Processor Fan
9. Case Fan
10. Hard Drive (SATA Capable)

Step 2: Gather Tools and Supplies

Gathering the tools you will need for the project:

- Screwdriver
- Needle-nosed pliers
- Small flashlight
- Small container to hold screws
- Grounding Strap
- Wire cutters and strippers
- Utility knife
- Adjustable wrench
- Heat sink compound

Step 3: Open the Case

Open the computer case by removing the side panels. Find the screws that hold the side panels in place and remove them

Step 4: Prepare the Case for Assembly

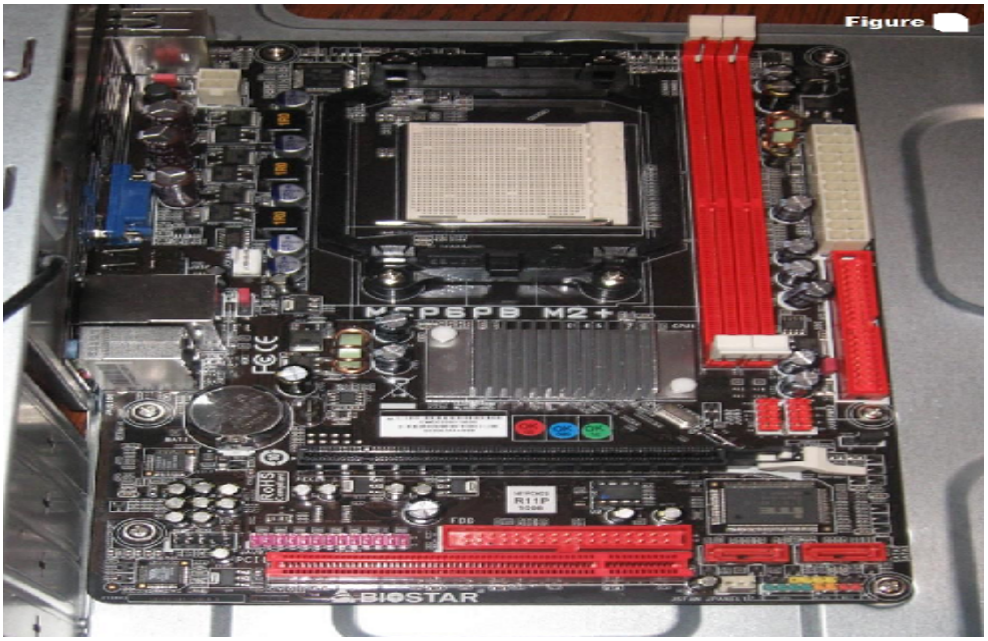
- Three things need to be done before assembly begins:
- Remove any parts or packaging materials that may have been shipped inside the case .
- Remove the cover for the optical drive. we will be removing the cover on the highest drive bay to mount our DVD drive
- These should be front panel connections for features such as the power switch, audio jacks and USB ports.

Step 5: Install Motherboard.

To install the motherboard we need parts that should have been included with your purchased components:

- I/O Bezel is a trim panel installed in the back of the case that surrounds the interface ports on the motherboard. It should be included with the motherboard.

- Standoffs are installed in the case screw holes to create a riser that separates the case and motherboard.
- Install standoffs in the case. The standoffs screw into the motherboard mounting holes Check the screw hole locations on the motherboard for exact placement.
- Lower the motherboard into the case and align with the I/O bezel.
- Install the screws. It works best to leave the screws loose until all of them have been started and the board is aligned with the bezel.



Step 6: Install Hard Drive

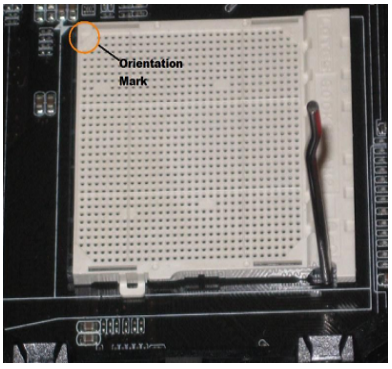
- The hard drive is the device that stores all of your data. It is 3.5" wide and needs to be mounted so that you can gain access to the cable connections on the back
- Find a 3.5" drive bay to install the drive in.
- Slide the drive into place until the screw holes on the sides are lined up with the holes in the case. And Install the screws.

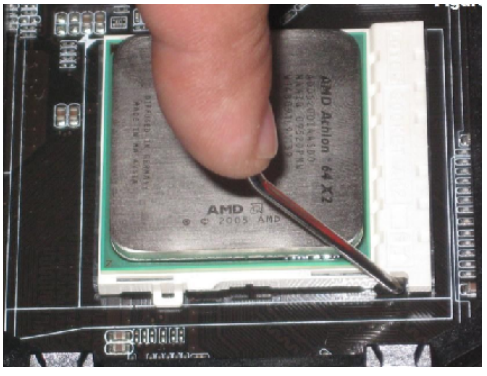
Step 7: Install Optical Drive

- The optical drive is 5.25" wide and is installed in the drive bay that we removed the cover from in a previous step. Cable access considerations apply to this drive also.
  To install the drive:
- Slide the drive into the drive bay until the screw holes are lined up and the front of the drive is flush with the front of the case
- Install the screws.

Step 8: Install the CPU

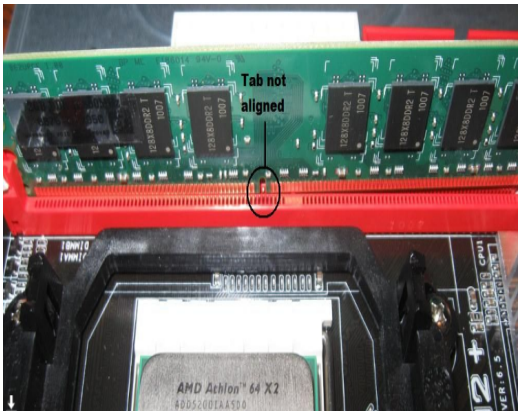
- The CPU is the brain of the computer. It is installed on the motherboard in the socket
  To install the CPU:
- Find the corner marking that designates pin 1 of the CPU , On this AMD brand processor, the corner is marked with an arrow. Consult the manufacturer's documentation for specific information about your processor.
- Lift the small metal rod next to the socket
- Find the corresponding marking on the CPU socket and insert the CPU so that the markings are lined up.
- Push the rod down to lock the processor in place

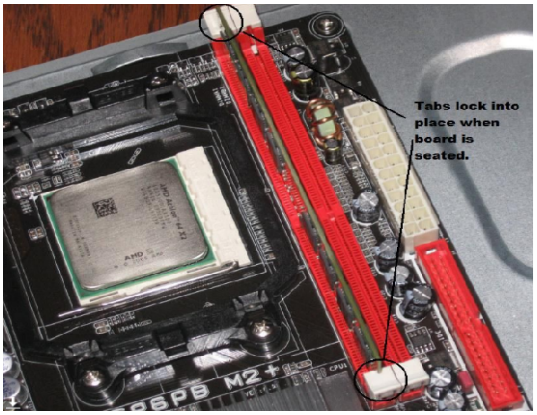




Step 09: Install RAM

- The RAM is the temporary memory location that the processor works from. Permanently stored data is pulled from disks and stored in RAM while the processor works with it.
- Set the RAM board in the socket. Check to see that the notch in the board is in the correct location. If it is not, turn it around 180º.
- Press firmly on both ends of the board to set it into the socket.





Step 10: Install the CPU Fan

- The CPU fan is really a combination of a heat sink and fan together. The unit draws heat away from the CPU.

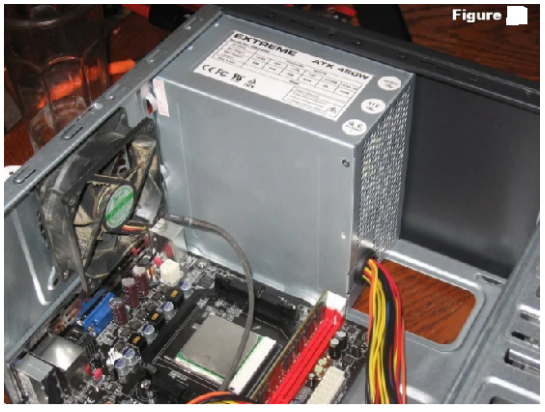
  To install the fan:
- Place thermal compound to the CPU following the instructions provided with the compound.
- Set the fan assembly on the CPU with mounting tabs aligned.
- Pull the locking rod down on the fan assembly to lock into place.
- Connect the fan assembly's power connector to the motherboard. Consult the manual to determine proper placement.

Step 11: Install Case Fan

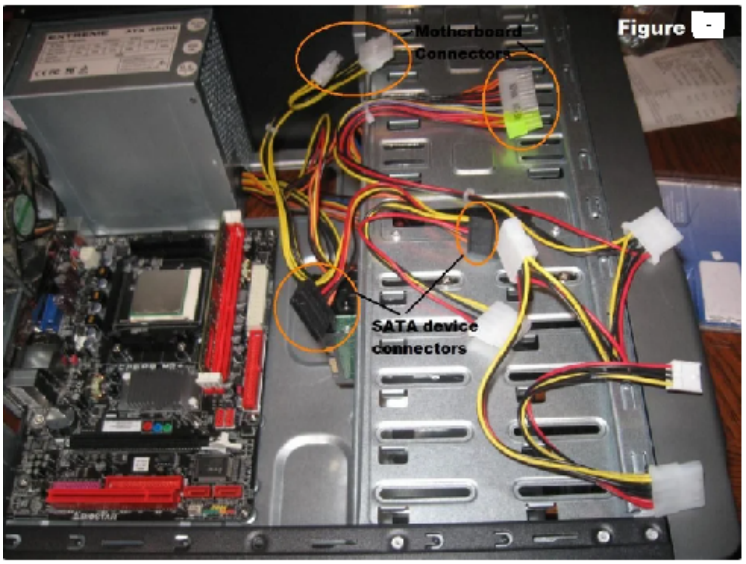
- The case fan is usually installed on the back panel of the case.

  To mount the fan:
- Align the mounting holes by holding the fan to the mounting pad on the inside of the case. The fan needs to be mounted so that it blows air out of the case.
- Insert the screws from the outside of the case and tighten.

Step 12: Install Power Supply

Step 13: Connect 

Cables



- Every device that has been installed needs power. The power supply connectors are shown. The motherboard has two power connections, and there are two connectors specifically for SATA devices (drives). The other connectors will run fans and other non-SATA devices.
- Data cables connect drives and front panel devices to the motherboard. Please consult the motherboard documentation for the exact placement of connectors.

Step 14: Wrap-up

Result : Thus the different components of pc are connected to work properly assembling of system components

EX NO:3 DATE:	TO UNDERSTAND HOW DIFFERENT COMPONENTS OF PC ARE CONNECTED TO WORK PROPERLY DISASSEMBLING OF SYSTEM COMPONENTS

AIM: To Understand How Different Components Of Pc Are Connected To Work Properly Disassembling Of System Components

PROCEDURE:

Step 1: Unplugging

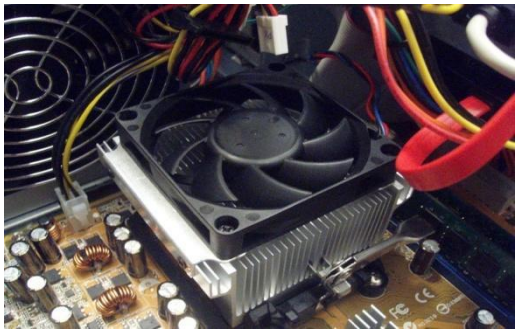
Unplug every cable that's plugged in to your computer. That includes the following cables:Power,USB, Firewire, Mouse, Keyboard, Internet, Ethernet, Modem, AM\FM Antenna, Cable TV

Step 2: Outer Shell/Casing

Unscrew the four screws on the back of the computer.Once the screws are removed, you can remove the side panels.

Step 3: System Fan

unplug the fan from the motherboard. You can find the plug by following the wire from the fan. It should be labeled"SYS_FAN1". Next, you will have to unscrew the fan from the outside. You should now be able to lift the fan out of the PC.



Step 4: Power Supply

The power supply supplies power to every component in a computer, unplug every wire coming from the power supply. The list below is everything that I had to disconnect:

- Motherboard (very large connector/plug)
- CD/DVD drive[s] power
- Internal hard drive power
- Portable hard drive slot power

Once everything is unplugged, unscrew the four screws holding the power supply in place, on the back of the computer.

Step 5: CD/DVD Drive[s]

The CD/DVD drive is one of the easiest components to remove. First, unplug the ribbon from the back of the drive.

Step 6: Hard Drive & Portable Hard Drive Slot

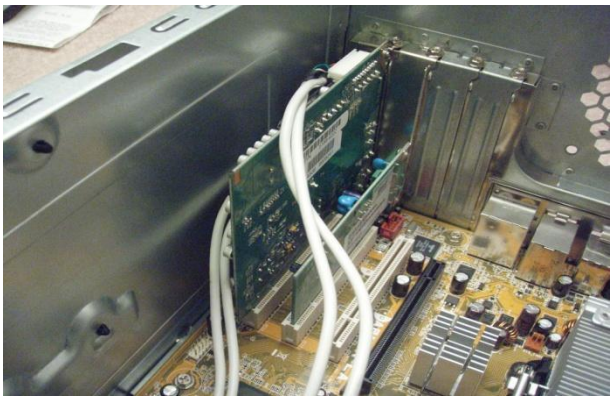
In order to remove the hard drive, you must remove the portable hard drive slot first.

First de-attach the connector at the back of the slot, and unplug the other end from the motherboard. unplug the SATA cable from the motherboard and the hard drive. The portable hard drive slot is secured the same way the CD/DVD drive is, with a tab. Pull on the tab, then slide the slot out.To remove the hard drive from the side of the slot, unscrew the four screws securing it in place.

Step 07: Expansion Cards

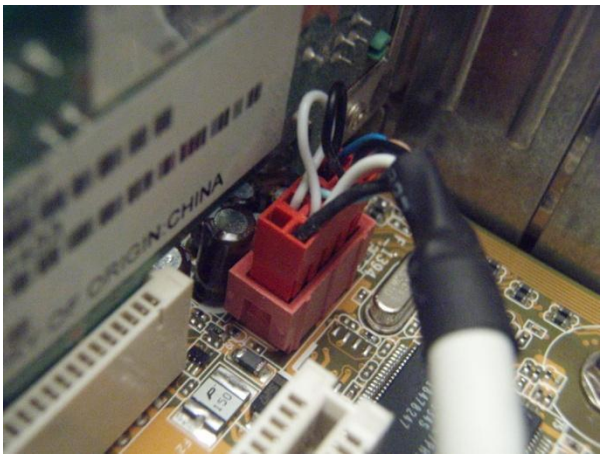
Expansion cards are like small upgrades to your computer.

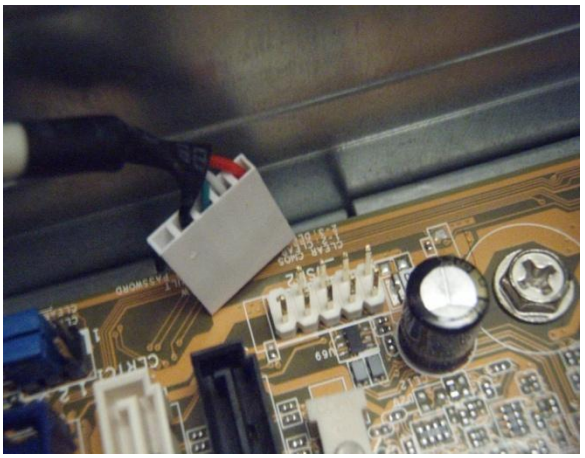
Expansion cards give a computer new capabilities, once installed. Different examples are:Bluetooth,WirelessInternet,Ethernet,TV



Step 08: Connectivity Centre Cables

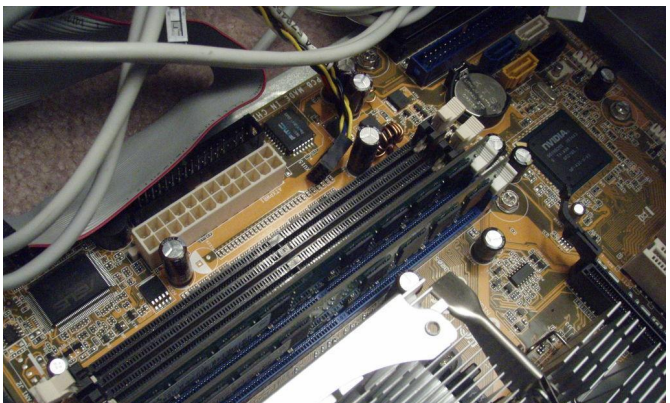
The connectivity center is the area on the front of the computer where there is many input sections, like usb, firewire, microphone, headphones, video, etc.. I won't remove the whole connectivity center in this step, but I will unplug all the cables coming from it.

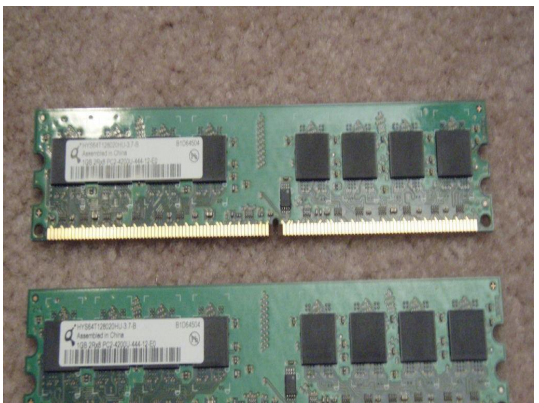




Step 09: RAM (Random Access Memory)

RAM you have, the faster your computer runs. Most computers have 4 RAM slots, and two RAM chips. To remove the RAM, push down on both tabs holding the RAM in place, which are located at both ends of the RAM. Please see the pictures.





Step 10: Motherboard

The motherboard links every component in the computer together. The CPU, RAM, and expansion cards are attached directly to it, and every other part of the computer is in one way or another attached to it.The motherboard has seven screws holding it to the frame, which are indicated by large white circles around them.

Step 11: Disassemble a Computer has been done

Result: Thus the different components of pc are connected to work properly Disassembling of system components

ExNo:4 i) Program involving Arithmetic Instructions on 8 bit data- Addition

AIM: To implement assembly language program for addition of two 08-bit numbers.

APPARTUS: TASM Software, P.C.

PROGRAM:

```
DATA SEGMENT

A DB 09H

B DB 02H

C DW ?

DATA ENDS


CODE SEGMENT

ASSUME CS:CODE,DS:DATA

START:

MOV AX,DATA

MOV DS,AX

MOV AL,A

MOV BL,B

ADD AL,BL

MOV C,AX

INT 3

CODE ENDS

END START
```

ExNo:4 ii) Program involving Arithmetic Instructions on 8 bit data- subtraction

AIM: To implement assembly language program for addition of two 08-bit numbers.

APPARTUS: TASM Software, P.C.

PROGRAM:

```
DATA SEGMENT

A DB 09H

B DB 02H

C DW ?

DATA ENDS


CODE SEGMENT

ASSUME CS:CODE,DS:DATA

START:

MOV AX,DATA

MOV DS,AX

MOV AL,A

MOV BL,B

SUB AL,BL

MOV C,AX

INT 3

CODE ENDS

END START
```

ExNo:5 i) Program involving Arithmetic Instructions on 16 bit data-

Addition

AIM: To implement assembly language program for addition of two 16-bit numbers.

APPARTUS: TASM Software, P.C.

PROGRAM:

```
DATA SEGMENT
N1 DW 1234H
N2 DW 2134H
RES DW ?
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START: MOV AX, DATA
MOV DS, AX
MOV AX, N1
MOV BX, N2
ADD AX, BX
MOV RES, AX
INT 21H
CODE ENDS
END START
```

RESULT:

AX = 3368h

ExNo:5 ii) Program involving Arithmetic Instructions on 16 bit data -

Subtraction

AIM: To implement assembly language program for subtraction of two 16-bit numbers.

APPARTUS: TASM Software, P.C.

PROGRAM:

```
DATA SEGMENT
N1 DW 4444H
N2 DW 2121H
RES DW ?
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START: MOV AX,DATA
MOV DS,AX
MOV AX,N1
MOV BX,N2
SUB AX,BX
MOV RES,AX
INT 21H
CODE ENDS
END START
```

RESULT:

AX = 2323h

ExNo:6i) Program involving Arithmetic Instructions on 16 bit data

Multiplication

AIM: To implement assembly language program for Multiplication of two 16-bit numbers.

APPARTUS: TASM Software, P.C.

PROGRAM:

```
DATA SEGMENT
A DW 1234H
B DW 5678H
C DD ?
DATA ENDS

CODE SEGMENT
ASSUME DS:DATA, CS:CODE
START:
MOV AX,DATA
MOV DS,AX
MOV AX,A
MOV BX,B
MUL BX
INT 3
CODE ENDS
END START
```

ExNo:6 ii) Factorial of a given number

AIM: To implement assembly language program to find factorial of a given number

APPARTUS: TASM Software, P.C.

Program:

```
DATA SEGMENT
A DB 5
DATA ENDS
CODE SEGMENT
        ASSUME DS:DATA,CS:CODE
START:
      MOV AX,DATA
      MOV DS,AX
      MOV AH,00
      MOV AL,A
 L1:  DEC A
      MUL A
      MOV CL,A
      CMP CL,01
      JNZ L1
      MOV AH,4CH
      INT 21H
CODE ENDS
END START
```

Input: 5

Output: 0078H

EXPERIMENT : 07 DATE :	**STUDY OF HALF ADDER AND FULL ADDER**

AIM: To realize Half Adder and Full Adder using Basic gates

_ To realize the half adder circuits using basic gates

_ To realize full adder using two half adders

COMPONENTS REQUIRED:

IC 7408, IC 7486, IC 7432, Patch Cords & IC Trainer Kit.

THEORY:

Adder circuit is a combinational digital circuit that is used for adding two numbers. A typical adder circuit produces a sum bit (denoted by S) and a carry bit (denoted by C) as the output. Typically adders are realized for adding binary numbers but they can be also realized for adding other formats like BCD (binary coded decimal, XS-3 etc. Besides addition, adder circuits can be used for a lot of other applications in digital electronics like address decoding, table index calculation etc. Adder circuits are of two types: Half adder ad Full adder.

Half-Adder: A combinational logic circuit that performs the addition of two data bits, A andB, is called a half-adder. Addition will result in two output bits; one of which is the sum bit, S, and the other is the carry bit, C.

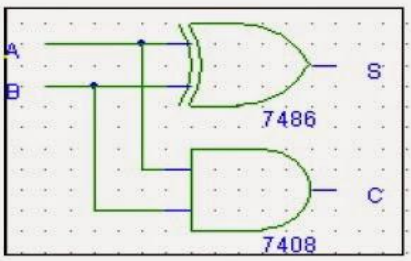
The Boolean functions describing the half-adder

BOOLEAN EXPRESSIONS:

$S=A \oplus B$
$C=A\ B$

are:

HALF ADDER CIRCUIT DIAGRAM:



TRUTH TABLE

INPUTS		OUTPUTS	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Full-Adder: The half-adder does not take the carry bit from its previous stage into account. This carry bit from its previous stage is called carry-in bit. A combinational logic circuit that adds two data bits, A and B, and a carry-in bit, C_{in} , is called a full-adder.

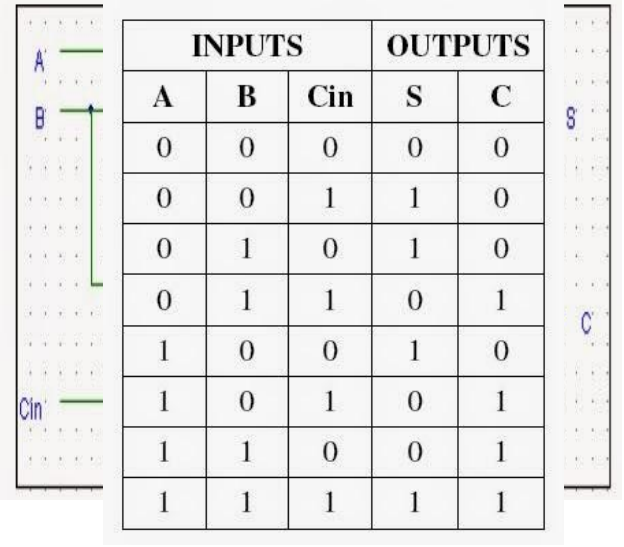
The Boolean functions describing the full-adder are:

BOOLEAN EXPRESSIONS:

$S= A \oplus B \oplus C$

$C=A\ B + B\ Cin + A\ Cin$

FULL ADDER CIRCUIT DIAGRAM :



FULL ADDER TRUTH TABLE

INPUTS			OUTPUTS	
A	B	Cin	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

RESULT

Thus the study of half adder and full adder has been done and components has been realized

EXPERIMENT : 08 DATE :	**STUDY AND DESIGN OF RIPPLE CARRY ADDER**

AIM: To realize ripple carry adder using Basic gates

_ To realize the ripple carry adder circuits using basic gates

_ To realize ripple carry adder using four full adders

COMPONENTS REQUIRED:

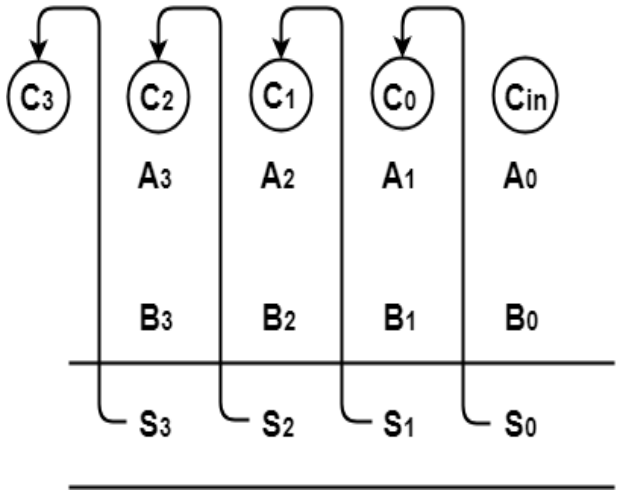
IC 7408, IC 7486, IC 7432, Patch Cords & IC Trainer Kit.

THEORY:

A ripple carry adder also known as "n-bit parallel adder" is a combinational logic circuit used for the purpose of adding two n-bit binary numbers and requires 'n' full adders in the circuit.

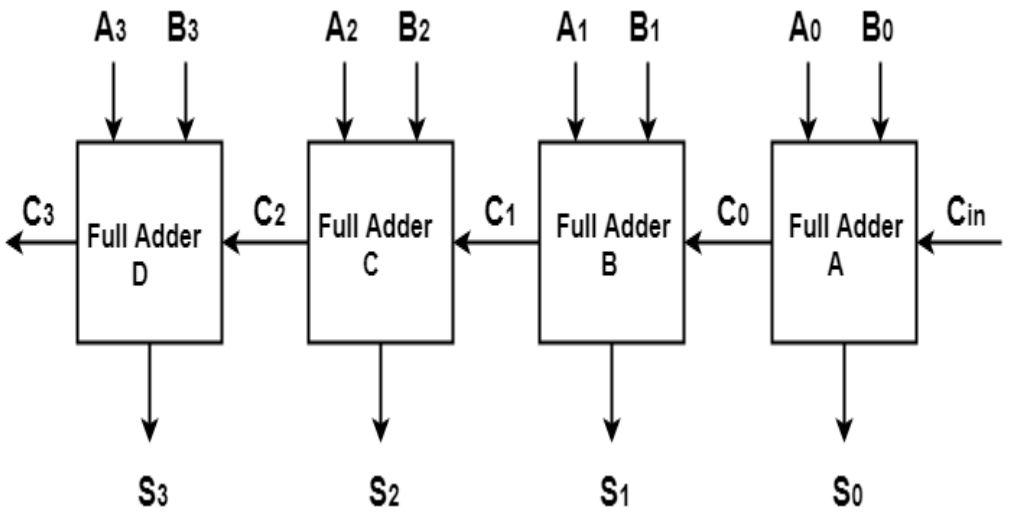
4-bit Ripple Carry Adder-

4-bit ripple carry adder is used for the purpose of adding two 4-bit binary numbers.
In mathematics, any two 4-bit binary numbers $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ will be added as-



Using ripple carry adder, this addition will be carried out as shown by the following logic diagram-

Using ripple carry adder, this addition will be carried out as shown by the following logic diagram-



Logic Diagram for 4-bit Ripple Carry Adder

As shown, Ripple Carry Adder works in different stages where the carry out produced by each full adder as output serves as the carry in input for its adjacent most significant full adder. When the carry in becomes available to the full adder, it activates that full adder and it comes into operation.

- Output Sum = $S_3S_2S_1S_0$ = 1111
- Output Carry = C_3 = 0

Why Ripple Carry Adder is called so?

- In Ripple Carry Adder, the carry out produced by each full adder as output serves as the carry in input for its next most significant full adder.
- Since in ripple carry adder, each carry bit ripples or waves into the next stage, that's why it is called by the name "Ripple Carry Adder".

Limitation of Ripple Carry Adder-

- Ripple Carry Adder does not allow all full adders to be used simultaneously and each full adder has to necessarily wait till the carry bit becomes available from its adjacent less significant full adder.
- This increases the propagation time and due to this reason, ripple carry adder becomes **extremely slow** which is considered to be the biggest disadvantage of using ripple carry adder.

RESULT

Thus the study of ripple carry adder has been done and components has been realized.

EXPERIMENT : 09 DATE :	**STUDY AND DESIGN OF CARRY LOOKAHEAD ADDER**

AIM: To realize ripple carry lookahead adder using Basic gates

_ To realize the carry lookahead adder circuits using basic gates

_ To realize carry lookahead adder using four full adders

COMPONENTS REQUIRED:

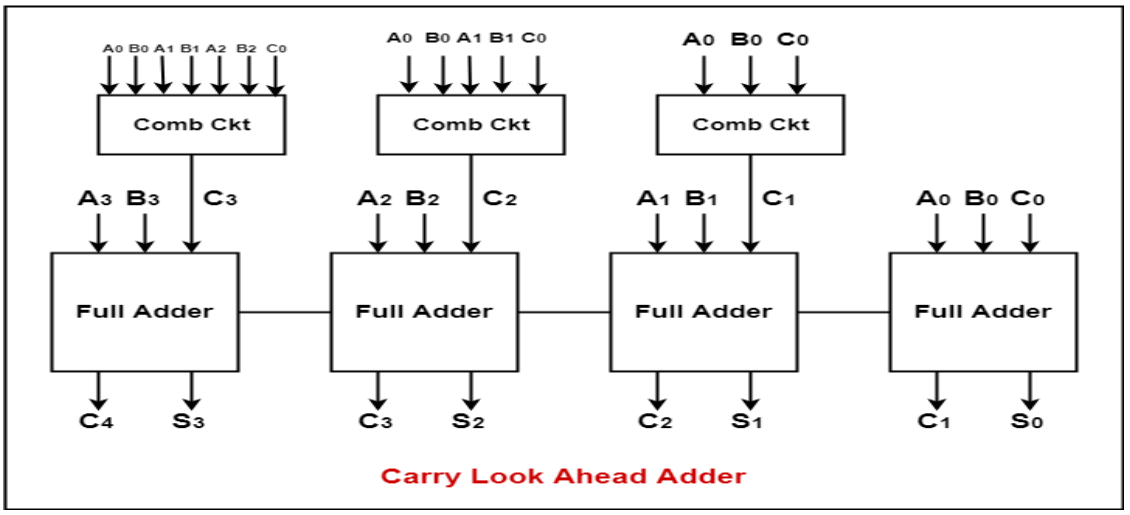
IC 7408, IC 7486, IC 7432, Patch Cords & IC Trainer Kit.

THEORY:

Carry Look Ahead Adder-

In Ripple Carry Adder, each full adder has to wait for its carry-in from its previous stage full adder to start its operation which causes an unnecessary delay.
Carry Look Ahead Adder is an improved version of the ripple carry adder which generates the carry-in of each full adder simultaneously without causing any delay.

Logic Diagram for Carry Look Ahead Adder-



Working of Carry Look Ahead Adder-

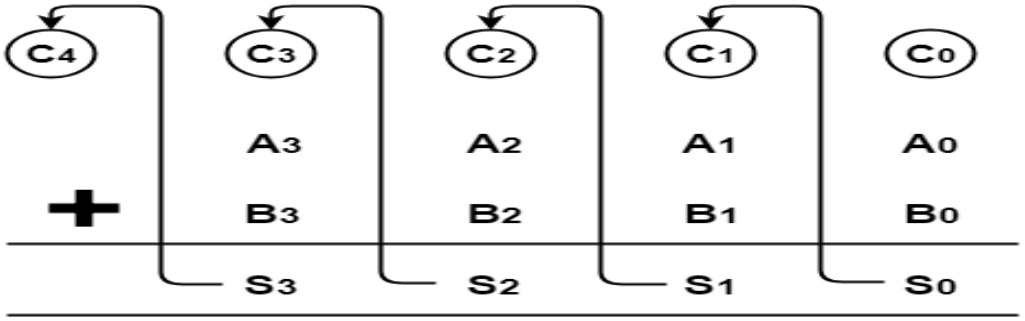
- The working of carry look ahead adder is based on the principle that the carry-in of any stage full adder is independent of the carry bits generated during intermediate stages and is only dependent of the following two parameters-

1. Bits being added in the previous stages
2. Carry provided in the beginning

- Since, the above two parameters are always known, the carry-in of any stage full adder can be evaluated at any instant of time. Thus, a full adder need not wait until its carry-in is generated by its previous stage full adder.

Example-

Suppose we want to add two 4-bit binary numbers A3A2A1A0 and B3B2B1B0
They will be added as-



From here, we have-

$C1 = C0 (A0 \oplus B0) + A0B0$

$C2 = C1 (A1 \oplus B1) + A1B1$

$C3 = C2 (A2 \oplus B2) + A2B2$

$C4 = C3 (A3 \oplus B3) + A3B3$

For simplicity, Let-

- $Gi = AiBi$ where G is called carry generator
- $Pi = Ai \oplus Bi$ where P is called carry propagator

Then, re-writing the equations, we have-

$$C1 = C0P0 + G0 \text{ …….. (1)}$$

$$C2 = C1P1 + G1 \text{ …….. (2)}$$

$$C3 = C2P2 + G2 \text{ …….. (3)}$$

$$C4 = C3P3 + G3 \text{ …….. (4)}$$

Now, let us remove C1, C2 and C3 from RHS of every equation as these are intermediate carry bits.

Substituting (1) in (2), we will get C2 in terms of C0 and then substituting (2) in (3), we will get C3 in terms of C0 and so on.

Finally, we will have the following equations showing that the carry in of any stage depends only on the bits being added in previous stages and the carry bit which was provided in the beginning-

- $C1 = C0P0 + G0$
- $C2 = C0P0P1 + G0P1 + G1$
- $C3 = C0P0P1P2 + G0P1P2 + G1P2 + G2$

- C4 =C0P0P1P2P3 + G0P1P2P3 + G1P2P3 + G2P3 + G3

Need of Carry Look Ahead Adder-

- In ripple carry adder, we have a cascade of n full adders and each full adder has to wait for its carry-in from its previous stage full adder to continue its operation.
- Thus, n^{th} full adder has to wait until all (n-1) full adders have completed their operations which causes a delay and makes ripple carry adder extremely slow.
- The situation becomes worst when the value of n becomes very large.
- To overcome this delay, we use carry look ahead adder in which carry for each full adder is generated simultaneously without any delay.

Advantage of Carry Look Ahead Adder-

- The advantage of carry look ahead adder is that it reduces propagation delay.

Disadvantage of Carry Look Ahead Adder-

- The disadvantage of carry look ahead adder is that it involves complex hardware.
- Because it involves complex hardware, it is costlier and it gets complicated as the number of bits increases.

Time Complexity-

Time complexity of Carry Look Ahead Adder = Θ (logn)

RESULT

Thus the study of carry lookahead adder has been done and components has been realized.

EXPERIMENT : 10 DATE :	**STUDY AND DESIGN OF ARRAY MULTIPLIER**

AIM: To realize array mutiplier using Basic gates

_ To realize the array mutiplier circuits using basic gates

_ To realize array mutiplier using four full adders

COMPONENTS REQUIRED:

IC 7408, IC 7486, IC 7432, Patch Cords & IC Trainer Kit.

THEORY:

Array multiplier is well known due to its regular structure. Multiplier circuit is based on add and shift algorithm. Each partial product is generated by the multiplication of the multiplicand with one multiplier bit. The partial product are shifted according to their bit orders and then added. The addition can be performed with normal carry propagate adder. N-1 adders are required where N is the multiplier length.

```
ARR1   DB 20 DUP (0)
MULTIPLYING PROC
        MOV AX, 0
        MOV CX, 19
        .WHILE CX != 0
                                MOV DI, CX
                                MOV AL, [DIGIT_ARR1+DI]
                                ;MOV BL, 2
                                ;MUL BL
                                SHL AX, 1
                                .IF AX > 9                  ; IF THE NEW DIGIT IS LARGER THAN
9
                                        SUB AX, 10
                                        MOV AH, 0
                                        MOV [DIGIT_ARR1+DI], AL
                                        DEC DI
                                        ADD [DIGIT_ARR1+DI], 1
                                .ELSEIF
                                        MOV [DIGIT_ARR1+DI], AL   ; IF IT IS LESS THAN 9,
THEN JUST INSERT IT BACK INTO THE ARRAY
                                .ENDIF
                DEC CX
        .ENDW
        RET
MULTIPLYING ENDP
MULTIPLYING PROC
        PUSH CX
        PUSH AX
        MOV CARRY, 0            ; START WITH EMPTY CARRY FLAG
```

```
        MOV CX, 19              ; ARRAY SIZE
        .WHILE CX > 0
                MOV DI, CX          ; GET ELEMENT ADDRESS
                MOV AL, [ARR1+DI]           ; READ THE ELEMENT
                SHL AL, 1           ; DOUBLING THE DIGIT
                ADD AL, CARRY       ; ADD THE CARRY FLAG
                MOV CARRY, 0        ; CLEAR THE CARRY FLAG
                .IF AL > 9                  ; IF THE NEW DIGIT IS LARGER THAN 9
                        SUB AL, 10
                        MOV CARRY, 1            ; SET CARRY FLAG
                        MOV [ARR1+DI], AL   ; INSERTING THE DOUBLED DIGIT BACK TO
THE ARRAY
                .ELSEIF
                        MOV [DIGIT_ARR1+DI], AL     ; IF IT IS LESS THAN 9, INSERT IT
BACK INTO THE ARRAY
                        MOV CARRY, 0
                .ENDIF
                DEC CX
        .ENDW; END OF MULTIPLICATION PROC
        POP AX
        POP CX
        RET
MULTIPLYING ENDP
```

RESULT

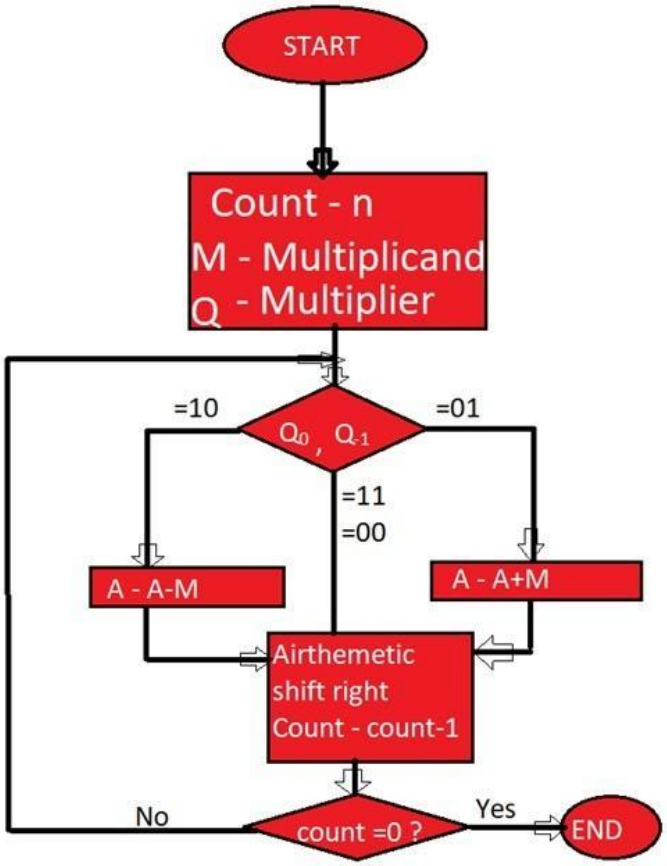
Thus the study of carry lookahead adder has been done and components has been realized.

EXPERIMENT : 11 DATE :	**STUDY OF BOOTH ALGORITHM**

Booth's algorithm

This is a kind of algorithm which uses a more straightforward approach. This algorithm also has the benefit of the speeding up the multiplication process and it is very efficient too. Binary multiplication which has signed number uses this type of algorithms named as **Booth's algorithm**.

Flowchart of Booth's algorithm:



Booth's algorithm for two complements multiplication:

1. Multiplier and multiplicand are placed in the Q and M register respectively.
2. Result for this will be stored in the AC and Q registers.
3. Initially, AC and Q_{-1} register will be 0.
4. Multiplication of a number is done in a cycle.
5. A 1-bit register Q_{-1} is placed right of the least significant bit Q_0 of the register Q.
6. In each of the cycle, Q_0 and Q_{-1} bits will be checked.
    i. If Q_0 and Q_{-1} are 11 or 00 then the bits of AC, Q and Q_{-1} are shifted to the right by 1 bit.
    ii. If the value is shown 01 then multiplicand is added to AC. After addition, AC, Q_0, Q_{-1} register are shifted to the right by 1 bit.
    iii. If the value is shown 10 then multiplicand is subtracted from AC. After subtraction AC, Q_0, Q_{-1}register is shifted to the right by 1 bit.

Basically, Booth's algorithm uses the concept of an arithmetic right shift in which the leftmost bit is not only shifted right by 1 bit but it also remains in the original position.

Example: Let us multiply (-6) and (2) using Booth's algorithm.

Solution: $(6)_{10} = (0110)_2$

As it is given multiplicand, **M=** $(-6)_{10}$ **=2** complement of **0110 = 1010**

Multiplier, **Q=** $(2)_{10}$ **= 0010**

AC	Q	Q_{-1}		
0000	0010	0	Initial	
0000	0001	0	shift	Q_0, Q_{-1} =00
0110	0001	0	AC=AC-M	Q_0 ,Q_{-1} = 10
0011	0000	1	shift	
1101	0000	1	AC=AC+M	Q_0, Q_{-1} =01
1110	1000	1	Shift	
1111	0100	0	Shift	Q_0, Q_{-1} =00

Product by Booth's algorithm= 1111 0100

RESULT

Thus the study of booth algorithm has been realized.

EXPERIMENT : 12 DATE :	**PROGRAM TO CARRY OUT BOOTH ALGORITHM**

Booths Algorithm

```
.model small
.stack
 .data
promptdb 13,10,"Enter first number to multiply. $"
prompt2db 13,10,"Enter second number to multiply. $"
resdb 13,10,"The answer is $"
ansdw 2
holddb 0
 n1=0
 n2=0
.code
start:
movax,seg prompt,prompt2,res,ans,hold,n1,n2
movds,ax
mov ah,09h
movdx,offset prompt
int 21h
call read
mov n1,bl
mov ah,09h
mov dx, offset prompt2
int 21h
call read
mov n2,bl
call Algorithm
mov [ans],ax
```

```
movbx,ax
movdx,offset res2
mov ah,09h
int 21h
call write
mov ah,4ch
int 21h
hlt
read:
mov ah,00h
mov [hold],bl
f0:
mov al,01h
int 21h
cmp al,0dh
je Copy
movcl,al
sub cl,30h
mov al,[hold]
mov bl,0ah
mulbl
mov [hold],al
add [hold],cl
jmp f0
Copy :
movbl,[hold]
ret
```

Algorithm:

```
mov ah,0
mov al,n1
mov cx,8
mov bh,n2
clc
f1:
movbl,al
and bl,1
jnz f2
  JNC f3
subah,bh
jmp f3
f2:
jc f3
addah,bh
 f3:
shr ax,1
loop f1
ret
write:
moval,bl
lea di,[ans]
mov bh,0ah
mov cl,24h
mov [di],cl
dec di
```

```
f4:
mov ah,00h
divbh
add ah,30h
mov [di],ah
dec di
cmp al,00h
jnz 4
inc di
mov ah,09h
movdx,di
int 21h
ret
end start
```

RESULT:

Thus the program of booth algorithm has been realized.

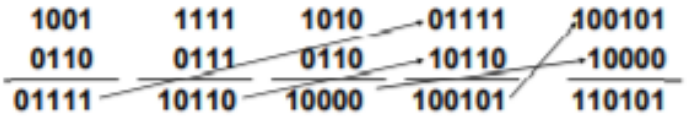
EXPERIMENT : 13 DATE :	**STUDY AND PROGRAM TO CARRY SAVE MULTIPLICATION**

Carry –Save Addition

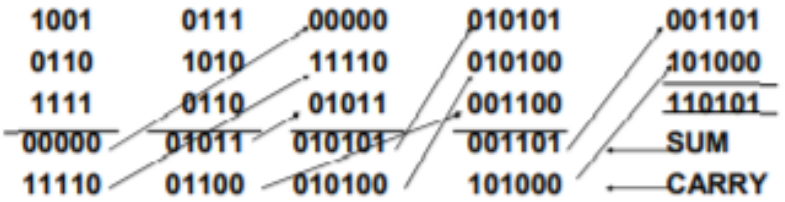
Consider adding six set of numbers (4 bits each in the example)

• The numbers are 1001, 0110, 1111, 0111, 1010, 0110 (all positive)

• One way is to add them pair wise, getting three results, and then adding them again



- Other method is add them three at a time by saving carry



Carry-Save Multiplication

• n-bit carry-save adder take 1FA time for any n

• For n x n bit multiplication, n or n/2 (for 2 bit at time Booth's encoding) partial products can be generated

• For n partial products n/3 n-bit carry save adders can be used

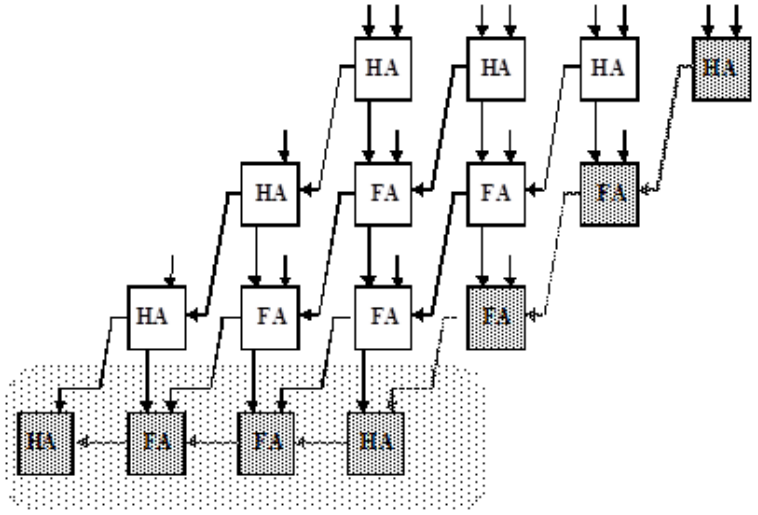
• This yields 2n/3 partial results

• Repeat this operation until only two partial results are remaining

• Add them using an appropriate size adder to obtain 2n bit result

• For n=32, you need 30 carry save adders in eight stages taking 8T time where T is time for one-bit full adder

• Then you need one carry-propagate or carry-look-ahead adder



Vector Merging Adder

PROGRAM TO CARRY OUT CARRY SAVE MULTIPLICATION

RESULT

Thus the study of carry lookahead adder has been done and components has been realized.

EXPERIMENT : 14 DATE :	**UNDERSTANDING OF PROCESSING UNIT AND DESIGN OF PRIMITIVE PROCESSING UNIT**

Definition

The central processing unit (CPU) is the unit which performs most of the processing inside a computer. To control instructions and data flow to and from other parts of the computer, the CPU relies heavily on a chipset, which is a group of microchips located on the motherboard.

The CPU has two components:

- Control Unit: extracts instructions from memory and decodes and executes them
- Arithmetic Logic Unit (ALU): handles arithmetic and logical operations

To function properly, the CPU relies on the system clock, memory, secondary storage, and data and address buses.

This term is also known as a central processor, microprocessor or chip.

The CPU is the heart and brain of a computer. It receives data input, executes instructions, and processes information. It communicates with input/output (I/O) devices, which send and receive data to and from the CPU. Additionally, the CPU has an internal bus for communication with the internal cache memory, called the backside bus. The main bus for data transfer to and from the CPU, memory, chipset, and AGP socket is called the front-side bus.

The CPU contains internal memory units, which are called registers. These registers contain data, instructions, counters and addresses used in the ALU's information processing.

Some computers utilize two or more processors. These consist of separate physical CPUs located side by side on the same board or on separate boards. Each CPU has an independent interface, separate cache, and individual paths to the system front-side bus. Multiple processors are ideal for intensive parallel tasks requiring multitasking. Multicore CPUs are also common, in which a single chip contains multiple CPUs.

Technical considerations are explained in the following sections:

1. Clock speed requirements for the CPU

The clock speed of the CPU is measured in Hertz and is an indication of the frequency in cycles per second of the operation of the CPU. The higher the clock speed of a CPU, the faster the performance of that CPU. For today’s personal computers, the clock speed of the processors vary from 1 GHz (Giga Hertz) to 3.5 GHz.

2. 32-bit vs. 64-bit processors

Most of the newer computers have 64-bit compatible processor units. The move to the newer 64-bit technology emerges from the need to allow computers to support larger RAM memory chips. The 32-bit architecture would only allow the use of a 4GB RAM, and even if a larger RAM is installed in the PC, it will only be able to use 4GB and the rest will be wasted.
Users with moderate computer usage, who want to simply browse the internet, write documents or send and receive e-mails can still select a 32-bit chipset. However users who heavily use their PC for multitask-heavy applications such as games and video editing will need to purchase a 64-bit chipset to optimize their computer performance.

3. Selection of the CPU's core and thread

The latest CPUs to hit the market contain up to eight cores. At the moment, it is possible to purchase CPUs containing from one to eight cores. The higher the number of cores in a CPU, the faster the processes and operations it can compete at one time. There are also multiple threads on each of those processors, these allow for a certain number of concurrent tasks to be completed per processor unit.

For moderate web and applications use, the user can simply purchase a single or a dual-core processor. However, for more heavy computer use such as multi-tasking applications, gaming, video editing and multimedia processing, it is recommended that the user selects a quad-core or higher in order to optimize the computer's performance during use.

4. Thermal Design Power

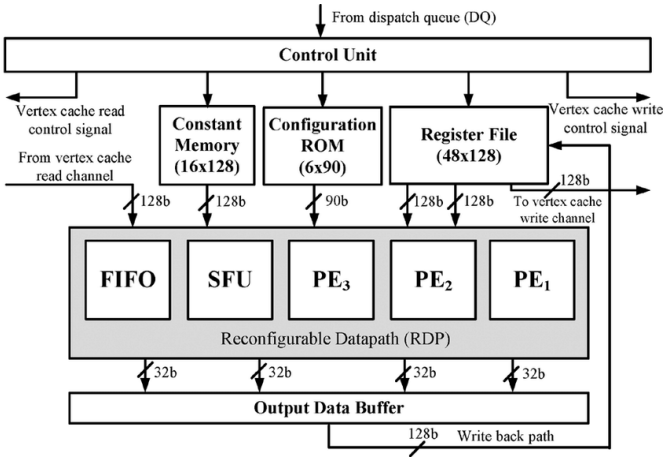
Thermal design power refers to the temperature at which a processor can continue its normal operation without damaging its own internal components. When buying a CPU, aim for a higher Thermal Design Power (TPD) number, as this will allow the CPU to produce more output without damaging its core.

5. Heatsink

When you buy a CPU unit, you need to keep in mind that the unit is prone to over-heating because of the amount of operations it processes per second. Other factors also influence and increase the over-heating of the CPU, such as the internal design of the computer's components, the distance between those components, the possibility of aeration of the CPU and the ambient temperature of the room or location where the PC is stored. If the ambient temperature is above average in hot summer days, it might also contribute to the over-heating of the processor. For these reasons, it is highly recommended that all processors have a heat sink installed to help cooling them down and preventing them from over-heating which will optimize their performance.

When the CPU in the computer experiences overheating and there is no heat sink attached to solve this problem, the user will start experiencing a range of erratic behaviour on the computer, such as computer lock ups, sudden reboots, application errors, etc. Keep in mind

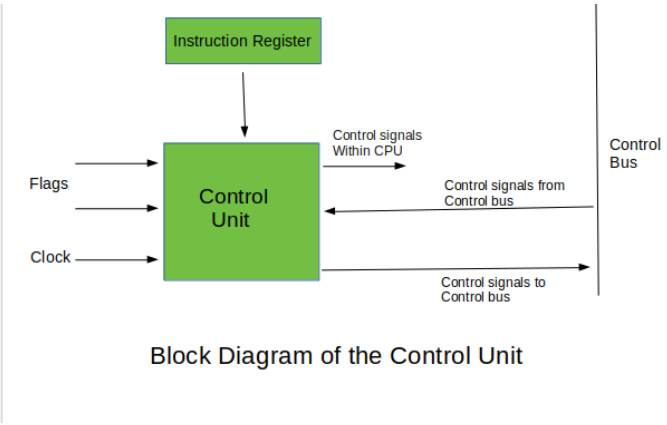
that the heat sink you buy should be compatible with the type of processor installed in your computer.



Design of primitive processing unit

Control Unit is the part of the computer's central processing unit (CPU), which directs the operation of the processor. It was included as part of the Von Neumann Architecture by John von Neumann. It is the responsibility of the Control Unit to tell the computer's memory, arithmetic/logic unit and input and output devices how to respond to the instructions that have been sent to the processor. It fetches internal instructions of the programs from the main memory to the processor instruction register, and based on this register contents, the control unit generates a control signal that supervises the execution of these instructions.
A control unit works by receiving input information to which it converts into control signals, which are then sent to the central processor. The computer's processor then tells the attached hardware what operations to perform. The functions that a control unit performs are dependent on the type of CPU because the architecture of CPU varies from manufacturer to manufacturer. Examples of devices that require a CU are:

- Control Processing Units(CPUs)
- Graphics Processing Units(GPUs)



Block Diagram of the Control Unit

Functions of the Control Unit –

1. It coordinates the sequence of data movements into, out of, and between a processor's many sub-units.
2. It interprets instructions.

3. It controls data flow inside the processor.
4. It receives external instructions or commands to which it converts to sequence of control signals.
5. It controls many execution units(i.e. ALU, data buffers and registers) contained within a CPU.
6. It also handles multiple tasks, such as fetching, decoding, execution handling and storing results.

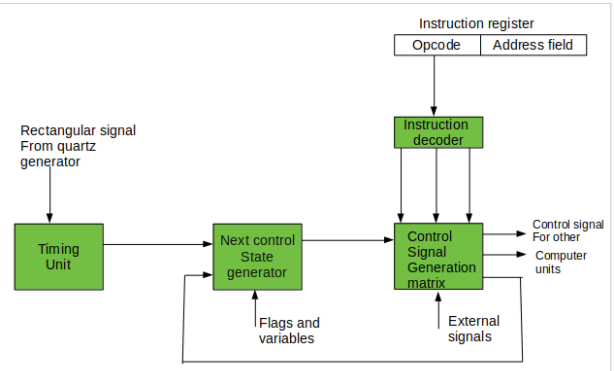
Types of Control Unit –

There are two types of control units: Hardwired control unit and Micro programmable control unit.

1. **Hardwired Control Unit**

   In the Hardwired control unit, the control signals that are important for instruction execution control are generated by specially designed hardware logical circuits, in which we can not modify the signal generation method without physical change of the circuit structure. The operation code of an instruction contains the basic data for control signal generation. In the instruction decoder, the operation code is decoded. The instruction decoder constitutes a set of many decoders that decode different fields of the instruction opcode.

   As a result, few output lines going out from the instruction decoder obtains active signal values. These output lines are connected to the inputs of the matrix that generates control signals for executive units of the computer. These matrixes implements logical combinations of the decoded signals from the instruction opcode with the outputs from the matrix that generates signals representing consecutive control unit states and with signals coming from the outside of the processor, e.g. interrupt signals. The matrices are built in a similar way as a programmable logic arrays.



Block diagram of a hardwired control unit of a computer

   Control signals for an instruction execution have to be generated not in a single time point but during the entire time interval that corresponds to the instruction execution cycle. Following the structure of this cycle, the suitable sequence of internal states is organized in the control unit.

   A number of signals generated by the control signal generator matrix are sent back to inputs of the next control state generator matrix. This matrix combines these signals with the timing signals, which are generated by the timing unit based on the rectangular patterns usually supplied by the quartz generator. When a new instruction arrives at the control unit, the control units is in the initial state of new instruction fetching.

Instruction decoding allows the control unit enters the first state relating execution of the new instruction, which lasts as long as the timing signals and other input signals as flags and state information of the computer remain unaltered. A change of any of the earlier mentioned signals stimulates the change of the control unit state.

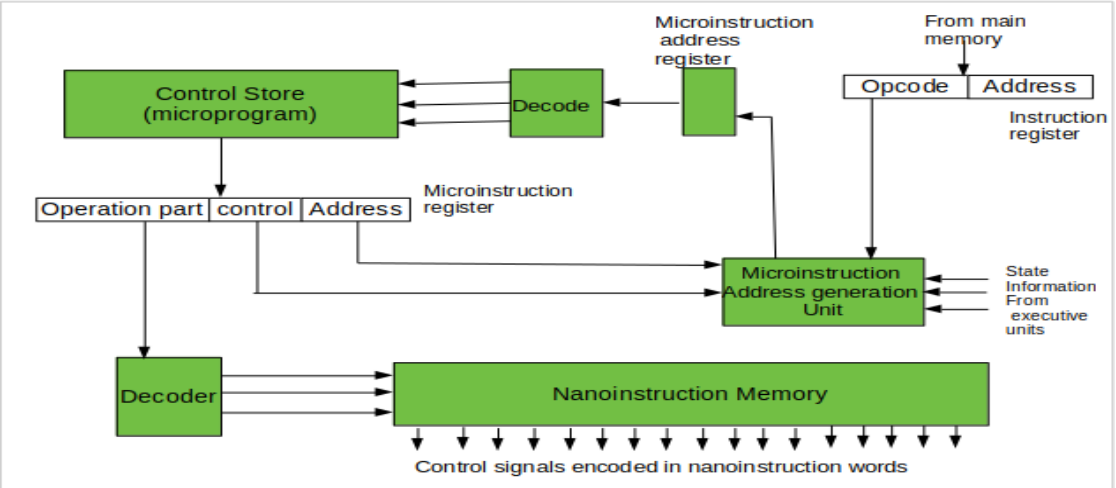
This causes that a new respective input is generated for the control signal generator matrix. When an external signal appears, (e.g. an interrupt) the control unit takes entry into a next control state that is the state concerned with the reaction to this external signal (e.g. interrupt processing). The values of flags and state variables of the computer are used to select suitable states for the instruction execution cycle.

The last states in the cycle are control states that commence fetching the next instruction of the program: sending the program counter content to the main memory address buffer register and next, reading the instruction word to the instruction register of computer. When the ongoing instruction is the stop instruction that ends program execution, the control unit enters an operating system state, in which it waits for a next user directive.

2. **Microprogrammable control unit**

The fundamental difference between these unit structures and the structure of the hardwired control unit is the existence of the control store that is used for storing words containing encoded control signals mandatory for instruction execution.

In microprogrammed control units, subsequent instruction words are fetched into the instruction register in a normal way. However, the operation code of each instruction is not directly decoded to enable immediate control signal generation but it comprises the initial address of a microprogram contained in the control store.



Microprogrammed control unit with a two-level control store

RESULT

Thus the study of understanding processing unit,design of primitive processing unit has been done and components has been realized.

EXPERIMENT : 15 DATE :	**UNDERSTANDING OF PIPELINE CONCEPT AND DESIGN OF BASIC PIPELINE**

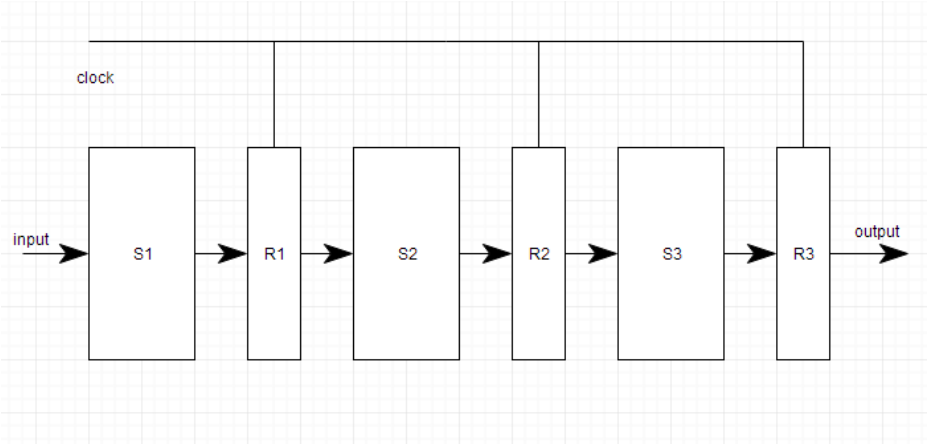
What is Pipelining?

Pipelining is the process of accumulating instruction from the processor through a pipeline. It allows storing and executing instructions in an orderly process. It is also known as pipeline processing.

Pipelining is a technique where multiple instructions are overlapped during execution. Pipeline is divided into stages and these stages are connected with one another to form a pipe like structure. Instructions enter from one end and exit from another end.

Pipelining increases the overall instruction throughput.

In pipeline system, each segment consists of an input register followed by a combinational circuit. The register is used to hold data and combinational circuit performs operations on it. The output of combinational circuit is applied to the input register of the next segment.



Pipeline system is like the modern day assembly line setup in factories. For example in a car manufacturing industry, huge assembly lines are setup and at each point, there are robotic arms to perform a certain task, and then the car moves on ahead to the next arm.

Types of Pipeline

It is divided into 2 categories:

1. Arithmetic Pipeline
2. Instruction Pipeline

Arithmetic Pipeline

Arithmetic pipelines are usually found in most of the computers. They are used for floating point operations, multiplication of fixed point numbers etc. For example: The input to the Floating Point Adder pipeline is:

X = A*2^a

Y = B*2^b

Here A and B are mantissas (significant digit of floating point numbers), while a and b are exponents.

The floating point addition and subtraction is done in 4 parts:

1. Compare the exponents.
2. Align the mantissas.
3. Add or subtract mantissas
4. Produce the result.

Registers are used for storing the intermediate results between the above operations.

Instruction Pipeline

In this a stream of instructions can be executed by overlapping *fetch*, *decode* and *execute* phases of an instruction cycle. This type of technique is used to increase the throughput of the computer system.

An instruction pipeline reads instruction from the memory while previous instructions are being executed in other segments of the pipeline. Thus we can execute multiple instructions simultaneously. The pipeline will be more efficient if the instruction cycle is divided into segments of equal duration.

Pipeline Conflicts

There are some factors that cause the pipeline to deviate its normal performance. Some of these factors are given below:

1. Timing Variations

All stages cannot take same amount of time. This problem generally occurs in instruction processing where different instructions have different operand requirements and thus different processing time.

2. Data Hazards

When several instructions are in partial execution, and if they reference same data then the problem arises. We must ensure that next instruction does not attempt to access data before the current instruction, because this will lead to incorrect results.

3. Branching

In order to fetch and execute the next instruction, we must know what that instruction is. If the present instruction is a conditional branch, and its result will lead us to the next instruction, then the next instruction may not be known until the current one is processed.

4. Interrupts

Interrupts set unwanted instruction into the instruction stream. Interrupts effect the execution of instruction.

5. Data Dependency

It arises when an instruction depends upon the result of a previous instruction but this result is not yet available.

Advantages of Pipelining

1. The cycle time of the processor is reduced.
2. It increases the throughput of the system
3. It makes the system reliable.

Disadvantages of Pipelining

1. The design of pipelined processor is complex and costly to manufacture.
2. The instruction latency is more.

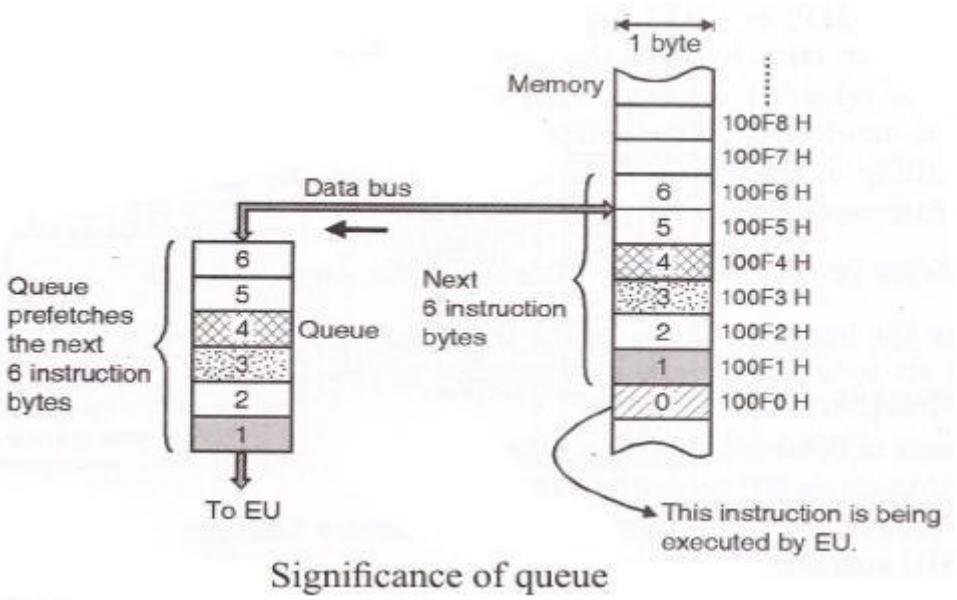
1. In 8086, to speed up the execution of program, the instruction fetching and execution of instructions are overlapped with each other.

2. This process of fetching the next instruction when the present instruction is being executed is called as pipelining.
3. In pipelining, when the nth instruction is executed, the (n+1) th instruction is fetched and thus the processing speed is increased.
4. Pipelining has become possible due to the use of queue.
5. BIU (Bus Interfacing Unit) fills in the queue until the entire queue is full.
6. BIU restarts filling in the queue when at least two locations of queue are vacant.

The Instruction Queue:

1. The execution unit (EU) is supposed to decode or execute an instruction.
2. Decoding does not require the use of buses.
3. When EU is busy in decoding and executing an instruction, the BIU fetches up to six instruction bytes for the next instructions.
4. These bytes are called as the pre-fetched bytes and they are stored in a first in first out (FIFO) register set, which is called as a queue.

Significance of Queue:



Significance of queue

1. As shown in the above figure, while the EU is busy in decoding the instruction corresponding to memory location 100F0, the BIU fetches the next six instruction bytes from locations 100F1 to 100F6 numbered as 1 to 6.
2. These instruction bytes are stored in the 6 byte queue on the first in first out (FIFO) basis.
3. When EU completes the execution of the existing instruction and becomes ready for the next instruction, it simply reads the instruction bytes in the sequence 1, 2…. from the Queue.
4. Thus the Queue will always hold the instruction bytes of the next instructions to be executed by the EU.

Advantages of pipelining:

1. The EU always reads the next instruction byte from the queue in BIU. This is much faster than sending out an address to the memory and waiting for the next instruction byte to come.
2. In short pipelining eliminates the waiting time of EU and speeds up the processing.
3. The 8086 BIU will not initiate a fetch unless and until there are two empty bytes in its queue. 8086 BIU normally obtains two instruction bytes per fetch.

Disadvantages of pipelining:

1. While pipelining can severely cut the time taken to execute a program, there are problems that cause it to not work as well as it perhaps should.
2. The three stages of the instruction execution process do not necessarily take an equal amount of time, with the time taken for 'execute' being generally longer than 'fetch'. This makes it much harder to synchronise the various stages of the different instructions.
3. Also, some instructions may be dependent on the results of other earlier instructions. This can arise when data produced earlier needs to be used or when a conditional branch based on a previous outcome is used.

RESULT

Thus the study and design of understanding pipeline cocepts has been done and components has been realized.