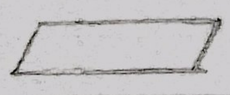
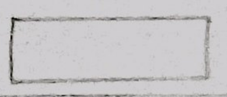
Symbol	Name
	Terminal
	Input / output
	Process
	Decision
	Connector
	Flow lines
	Predefined Process

16-08-23

16-aug-23 CIT - 235 Wed

Practical No : 1

(Introduction to Flowchart)

- **Terminal :-**
  It indicates the starting or ending of the program, process or interrupt program.
- **Input/output :-**
  It is used for any I/O operation.
- **Process :-**
  The process symbol indicates any internal operation inside the processor.
- **Decision :-**
  It is used to ask a question that can be answered as True/False
- **Connector :-**
  It allows the flowchart to be drawn without intersecting lines.
- **Flow lines :-**
  Flow lines show direction of flow.
- **Predefined process :-**
  It is used to invoke a subroutine or an interrupt program.

Algorithm

Start

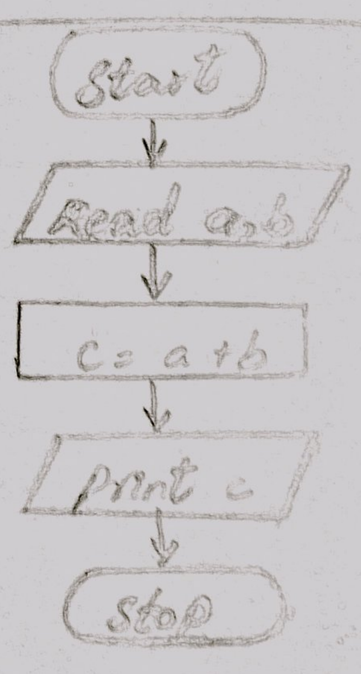
Read a, b

c = a + b

Print c

Stop

Flowchart



06-09-23

6-Sep-23 CIT-235 Wed

Practical # 2
(Algorithm from flowchart)

Program :-

```
#include <stdio.h>
int main()
{
    int a, b, c;
    printf("Enter value of a:");
    scanf("%d", &a);
    printf("Enter value of b:");
    scanf("%d", &b);
    c = a+b;
    printf("Sum of given two numbers is : %d", c);
    return 0;
}
```

Digits	ASCII
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57

Characters	ASCII
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90

Characters	ASCII
a	97
b	98
c	99
d	100
e	101
f	102
g	103
h	104
i	105
j	106
k	107
l	108
m	109
n	110
o	111
p	112
q	113
r	114
s	115
t	116
u	117
v	118
w	119
x	120
y	121
z	122

13-Sep-23 Microprocessor Architecture wednesday

Practical # 3

(Applications of ASCII code)

Character	ASCII	64	32	16	8	4	2	1
R	82	1	0	1	0	0	1	0
A	65	1	0	0	0	0	0	1
J	74	1	0	0	1	0	1	0
A	65	1	0	0	0	0	0	1
A	65	1	0	0	0	0	0	1
B	66	1	0	0	0	0	1	0
D	68	1	0	0	0	1	0	0
U	85	1	0	1	0	1	0	1
R	82	1	0	1	0	0	1	0
R	82	1	0	1	0	0	1	0
A	65	1	0	0	0	0	0	1
H	72	1	0	0	1	0	0	0
E	69	1	0	0	0	1	0	1
E	69	1	0	0	0	1	0	1
M	77	1	0	0	1	1	0	1

Copied

13-09-23

The Binary Number System :-

Bit value	1	1	1	1	1	1	1	1
Position value as a Power of base 2	128	64	32	16	8	4	2	1
Bit number	7	6	5	4	3	2	1	0

20-sep-23 Microprocessor Architecture wednesday

Practical # 4

(Introduction To Assembly language)

*. What is Assembly language?

Each personal computer has a microprocessor that manages the computer's arithmetical, logical and control activities. Each family of processors has its own set of instructions for handling various operations like getting input from keyboard, displaying information on screen and performing various other jobs. These sets of instructions are called 'machine language instruction'. Processor understands only machine language instructions which are strings of 1s and 0s. However machine language is too obscure and complex for using in software development. So the low level assembly language is designed for a specific family of processors that represents various instructions in symbolic code and a more understandable form.

*. Advantages of Assembly language :-

An understanding of assembly language provides knowledge of :

- Interface of programs with OS, processor and BIOS;
- Representation of data in memory and other external devices;
- How processor accesses and executes instructions;
- How instructions accesses and process data;
- How a program access external devices.

Other advantages of using assembly language are :

The Hexadecimal Number System :-

Decimal number	Binary representation	Hexadecimal representation
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1001	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Good

04-10-23

- It requires less memory and execution time ;
- It allows hardware - specific complex jobs in an easier way ;
- It is suitable for time - critical jobs ;

Basic Features of Pc Hardware :-

The main internal hardware of a pc consists of the processor, memory and the registers. The registers are processor components that hold data and address. To execute a program the system copies it from the external device into internal memory. The processor executes the program instructions.

The fundamental unit of computer storage is a bit, it could be on (1) or off (0). A group of nine related bits makes a byte. So the parity bit is used to make the number of bits in a byte odd. If the parity is even, the system assumes that there had been a parity error which might have caused due to hardware fault or electrical disturbance.

The processor supports the following data sizes :

- Word : a 2-byte data item
- Doubleword : a 4-byte (32 bit) data item
- Quadword : an 8-byte (64 bit) data item
- Paragraph : a 16-byte (128 bit) area
- Kilobyte : 1024 bytes
- Megabyte : 1,048,576 bytes.

DOS BOX Commands :-

Edit filename : asm
NASM filename : asm ;
Link filename : obj ;
file name . exe.

Program to print a single character :-

Dass eg
- model small
- Stack 100h
- data
- code

main Program
↓
mein end p
end main

(CIT-235)

4 oct-23 Microprocessor Architecture wednesday

practical # 05

(Familiarize Environment Setup)

Assembly language is dependant upon the instruction set E, architecture of the processors. In this tutorial, we focus on Intel 32 processors like pentium To follow this tutorial, you will need:

- An IBM PC or any equivalent compatible computer
- A copy of linux operating system
- A copy of NASM assembler

Programs, like;

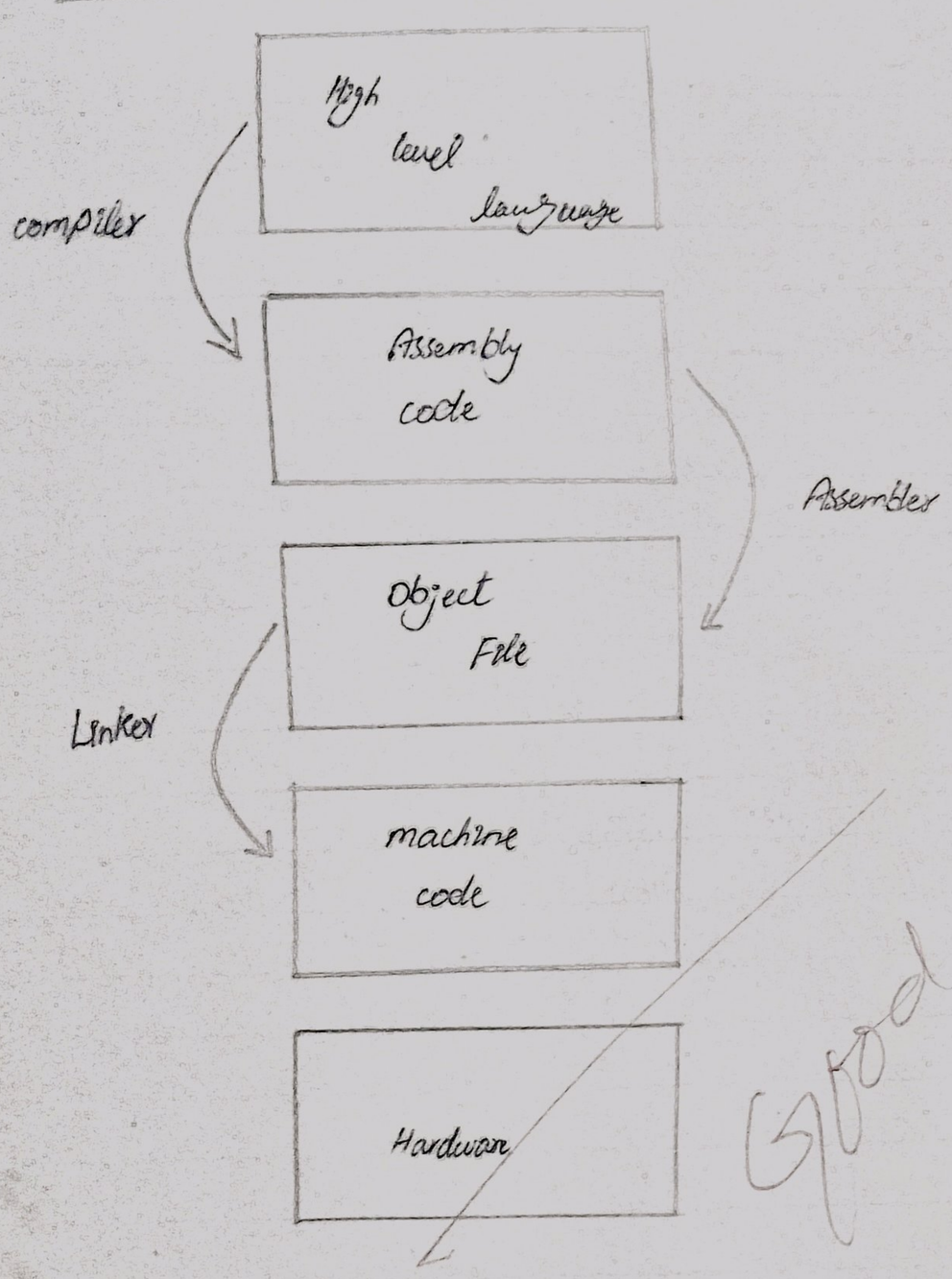
There are many good assembler Programs, like:

- Microsoft Assembler (MASM).
- Borland Turbo Assembler (TASM).
- The GNU assembler (GAS).

We will use the NASM assembler, as it is:

- Free. You can download it from various web sources.
- well documented E, you will get lots of information on net.
- Could be used on both linux E, window.

DOS BOX Emmulator :-



Good.

Installing NASM :-

If you select "Development Tools" while installed linux, you may NASM installed along with the linux operating system &, you don't need to download & Install it separately. For you checking whether you already have NASM installed, Take the following steps :

- Open a linux terminants
- Type where is NASM &, Press ENTER
- If it is already installed then a line like, NASM : /user/bin/ nasm appears. Otherwise, you will see just NASM then your need to install NASM take the following step.

```
Z:\> mount c c:\mp
Z:\> c:
C:\> edit file.asm
C:\> masm file.asm;
C:\> link file.obj;
C:\> file.exe
```

18-Oct-23 Microprocessor Architecture (CIT-235) Wednesday

Practical # 06

"; Program to print a single character."

```
dos seg
.model small
.stack 100h
.data
.code
main proc
mov dl, 'A'
Mov ah, 2
int 21h
mov ah, 4ch
int 21h
main endp
end main
```

18-10-23

```
Z:\> mount c c:\mp
Z:\> c:
C:\> edit file2.asm
C:\> masm file2.asm ;
C:\> link file2.obj ;
C:\> file2.exe
```

1-Nov-23 CIT - 235

practical No : 07

3 Program to print a name with character.

```
dosseg
.model small
.stack 100h
.data
.code
  main proc
  mov dl, '9'
  mov ah, 2
  int 21h
  mov dl, 'L'
  mov ah, 2
  int 21h
  mov dl, 'Z'
  mov ah, 2
  int 21h
  mov ah, 4ch
  main endp
  end main
```

5-Dec-23 (Microprocessor Architecture) Page

Practical # 8

1. program to print characters from A to Z.

```
Z:\> mount c c:\mp
Z:\> C:
C:\> edit file3.asm
C:\> masm file3.asm
C:\> link file3.obj;
C:\> file3.exe
```

```
Dosseg
.model small
.stack 100h
.data
.code
main proc
mov dl, 'A'
mov ah, 2
int 21h
mov dl, 'B'
mov ah, 2
int 21h
mov dl, 'C'
mov ah, 2
int 21h
mov dl, 'D'
mov ah, 2
int 21h
mov dl, 'E'
mov ah, 2
int 21h
mov dl, 'F'
mov ah, 2
int 21h
mov dl, 'G'
mov ah, 2
int 21h
mov dl, 'H'
mov ah 2
int 21h
mov dl, 'I'
mov ah, 2
int 21h
mov dl, 'J'
mov ah, 2
int 21h
mov dl, 'K'
mov ah, 2
int 21h
mov dl, 'L'
mov ah, 2
int 21h
mov dl, 'M'
mov dl, 'N'
mov ah, 2
int 21h
mov dl, 'O'
mov ah, 2
int 21h
mov dl, 'P'
mov ah, 2
int 21h
mov dl, 'Q'
mov ah, 2
int 21h
mov dl, 'R'
mov ah, 2
int 21h
mov dl, 'S'
int 21h
mov dl, 'T'
mov ah, 2
int 21h
mov dl, 'U'
int 21h
mov dl, 'V'
int 21h
mov dl, 'W'
int 21h
mov dl, 'X'
int 21h
mov dl, 'Y'
int 21h
mov dl, 'Z'
```

01-11-24

mained P
end main

Z:\> mount c c:\mp

C:\> drive mounted as local directory

Z:\>C:

C:\> edit File5.asm

C:\> masm File5.asm;

C:\> link File5.obj;

C:\> File.exe;

01-24

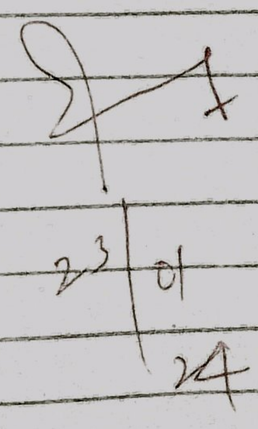
CiT_235

Tuesday

Practical # 09

"Program to multiply two numbers."

```
; program to multiply two Numbers.
Dosseg
.model small
.stack 100h
.data
.code
main proc
mov al, 5
mov bl, 2
mul bl
AAM
mov ch, ah
mov cl, al
mov dl, ch
add dl, 48
mov ah, 2
int 21h
mov dl, cl
add dl, 48
mov ah, 2
int 21h
mov ah, 4ch
int 21h
main endp
end main
```



```
Z:\>    mount   c   c:\mp
C:>     drive   mounted   as
        local   directory
Z:\> C:
C:\> edit    file4.asm
C:\> masm    file4.asm ;
C:\> link    file4.obj ;
C:\>         file4.exe ;
```

30.01.24

CiT - 235

Tues

practical # 10

"Program to subtract two numbers."

```
; program to subtract two numbers.
    Dosseg
    .model small
    .stack 100h
    .data
    .code
    main proc
    mov bl, 3

    mov cl, 1

    sub bl, cl
    add bl, 48

    mov dl, bl
    mov ah, 2
    int 21h

    mov ah, 4ch
    int 21h

    main endp
    end main
```

30-01-24

```
Z:\> mount c c:\MP
c :> drive mounted as local
                directory

Z:\> c:
c:\> edit file 6.asm
c:\> masm file 6.asm;
c:\> LINK file 6.obj;
c:\>_       file 6.exe;
```

Practical No - 11

Program to Add two numbers.

;Program to add two numbers:

```
.Dosseg
.model small
.Stack 100h
.data
.Code
 main proc

 mov bl, 1

 mov cl , 2

 add bl, cl

 add , bl, 48

 mov dl, bl

 mov ah, 2

 int 21h
```

```
main endP
end main.
```

```
z:\> mount c c:\MP
c :> drive mounted as local
        disedory
z:\> c:
c:\> edit file 5.asm
c:\> masm file 5.asm;
c:\> link file 5.obj;
c:\> file 5.exe;
```

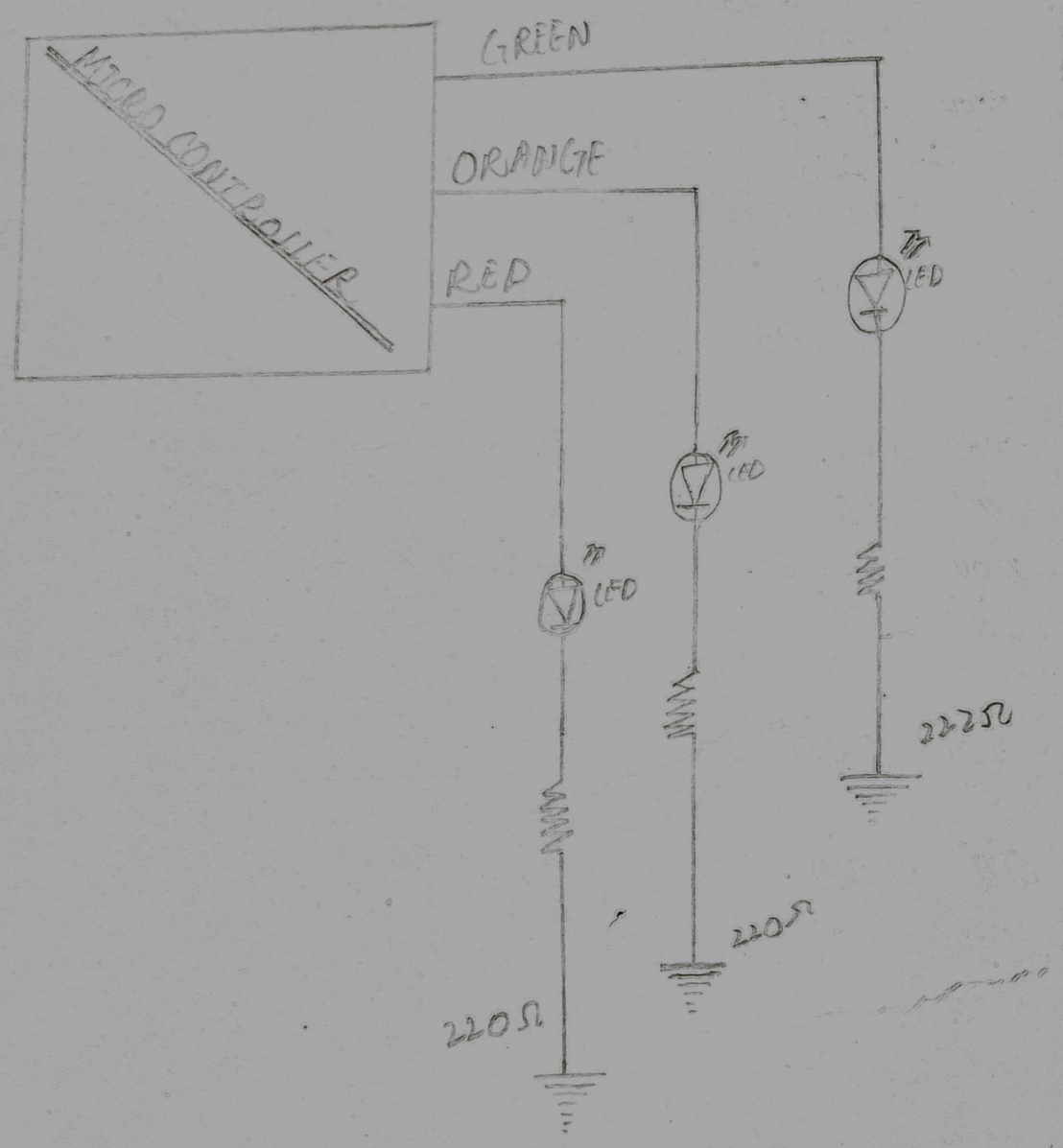
Program Practical No - 12

to divided - two numbers

• Program to Divide two numbers.

```
•Dosseg
•model small
•Stack 100h
• data
     q db:
     r db:
•code
 main proc

 mov ax , 26
 mov bl , 5

 Div bl
 mov q , al
 mov r , ah

 mov dl , q
 add dl , 48
 mov , ah , 2
  int 21h
```

```
mov dh , 0
add dl , 48

mov ah , 2
  int 21h

mov dh, 0
add dl, 48

mov ah, 2
 int 21h
mov ah, 42h
 int 21h

main endP

end main.
```



Practical No-13

Using Instruction of Assembly Language Contsol Traffid Lights

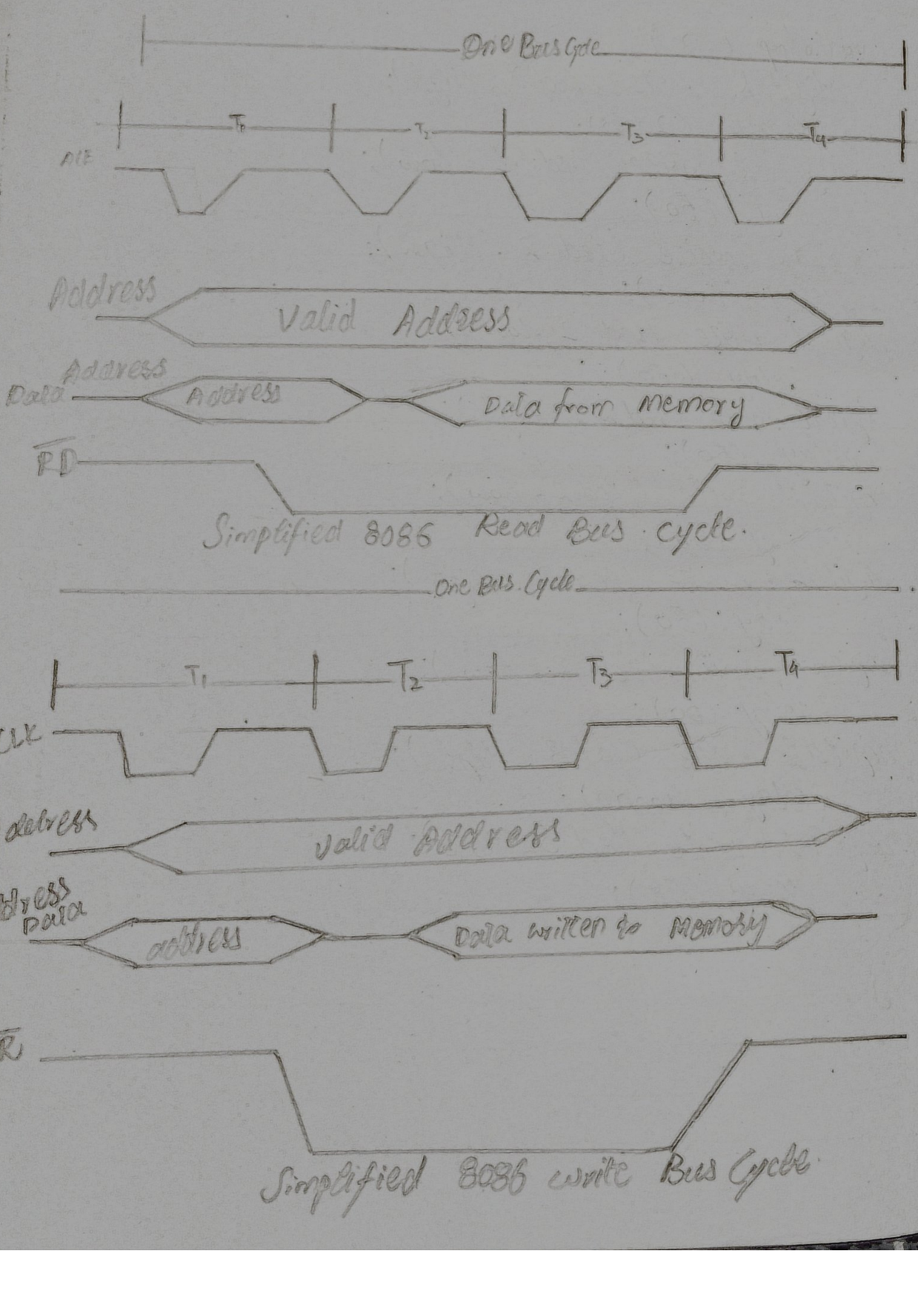
Apparatus:

- Bread board
- Micro Computer
- LEDs, RED, GREEN, ORANGE
- CPU for Programming
- Connecting leads
- Power Supply
- Resistor 220 Ω

Program:

```
int led 1 = 6;
int led 2= 9;
int led 3 = 12;

void Setup ( ) {
  pin mode (6, output);
  Pin mode (9, output);
  Pin mode (12, output);
}
```

```
void loop ( ) {
  digitah write (Led ), High);
   delay (2000);
  digitah write (led 1 , low);
   delay (50);
digitah write (led 2 , High);
   delay (5000);
digital write (Led 2 ,low );
     delay (50);
digital write (led 2 , high);
   delay (50);
digitah write ( led 2 , how );
     delay (50);
digitah write (led 2 , high );
     delay (50);
digitah write (led 2 , how );
     delay (50);
digitah write ( led 3 , high );
     delay (40000);
digitah write (led 3 , how);
     delay (50);

}
```



Practical NO - 14

Familiarize with 8086/88 System timing :-

Memory Read Bus Cycle :-

In this first phase of this cycle memory address is used on the address bus while control signals (ALE or M/$\overline{IO}$) indicate that the address on address bus is a memory location address. $\overline{RD}$ signal is send in the second phase of this cycle.

Memory Write Bus Cycle :

In this phase of this cycle memory address is sent through address bus. The $\overline{WR}$ signal is sent in second phase of this cycle. Due to bar on WR, it is a low active signal.

The process of write starts in third & fourth phase of this cycle.