Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

School of Electrical and Computer Engineering

Third Year B. Tech. (EE)

Microcontroller and Applications
Course Code: ECE2003B

Laboratory Manual

2023-2024

Preface

Microcontroller and Applications

Microcontrollers are single chip computers, integrating processor, memory and other peripheral modules into a single System-on-Chip (SoC). The single chip solution makes the footprint of the computational element small in the overall system package, eliminating the necessity of additional chips on board. Microcontrollers like 8051, PIC belong to this category. This course includes Silicon Labs' 8-bit Microcontroller C8051F340 which implements the standard 8051 organization. It introduces undergraduate students to the field of microcontrollers – what they are, how they work, how they interface with their I/O components, and what considerations the programmer has to observe in hardware-based and embedded programming. Understanding the architecture is the basis followed by hardware intricacies of these processors and their programming will be covered. Different system design examples built around these processors will also be elaborated. The lab for this course covers experiments based on interfacing of peripherals with C8051F340. Programming is in assembly and embedded C. IDE is Keil µvision and Simplicity Studio. After completion of this lab, students should able to develop a microcontroller-based interfaces for real life applications.

Microcontroller and Applications

I N D E X

Sr. No.	Name of the Experiment	Page	Date of Checking	Signature of Batch I/C
1	Simple assembly language programming.			
2	Complex assembly language programming			
3	Interfacing of LED, Buzzer, Relay and Switch with C8051F340			
4	Interfacing of LCD with C8051F340			
5	Interfacing of 8-bit DAC with C8051F340			
6	Programming of on chip ADC			
7	Interfacing DC motor and control its speed using PWM with C8051F340			
8	Implement UART with C8051F340			
9	Interfacing Stepper Motor with C8051F340			
10	Interfacing EEPROM using SPI with C8051F340			
11	Project Based Learning - Design and implement a microcontroller-based project			

CERTIFICATE

Certified that Mr./Ms._____ of Class **T. Y. B. Tech.** Division_____ Roll No._____ has completed the laboratory work in the subject **Microcontroller and Applications** during the **Semester V** in the School of Electrical and Computer Engineering during the Academic Year 2023-2024.

Signature of the Faculty **Head of the School**

Laboratory Instructions

Microcontroller and Applications

1. Students should have a valid ID card before entering the laboratory.
2. Playing of games on computer in the lab is strictly prohibited.
3. Before leaving the lab, users must close all programs/windows and turn-off the computer.
4. Do not modify or delete any important file and install any software or settings in the computer
5. Internet facility is to be used only for educational/ study purpose.
6. If any problem arises, please bring the same to the notice of Lab In-Charge.
7. Each student must take mobile phones in “Switched Off” or “Vibration” mode while entering and or working in laboratory.
8. In case of theft / destruction of the computers or peripherals, double the cost of that item will be charged from the student/user.
9. Tampering with the hardware or software settings will not be tolerated.
10. Lab Assistants are available to assist with basic computer and software problems.
11. Copying and/or installing of pirated software not permitted.
12. Personal files are not to be stored on the local drive. Students are responsible for their own means of digital storage. All lab computers are configured to remove any data stored or any programs installed by users.
13. Do not leave your personal belongings at the computer desk. The School is not responsible for items left behind.
14. Users are strictly prohibited from downloading, viewing or distributing any offensive materials.
15. Internet surfing or chatting for personal reasons is not allowed.

Guidelines to use µVISION IDE

Keil MicroVision is an Integrated Development Environment (IDE), which includes a text editor to write programs, a compiler to convert the source code to hex files and a debugger to test, verify, and optimize the application code.

Keil µVision can be used for:

- Writing programs in C/C++ or Assembly language
- Compiling and Assembling Programs
- Debugging program
- Creating Hex and Axf file
- Testing the program without real Hardware (Simulator Mode)

Step 1: After opening Keil uV, Go to **Project** tab and

Create new µVision project

Now Select new folder and give name to Project.

Step 2: After Creating project now **Select your device model** eg. Silicon Lab's C8051F340

Step 3: So now your project is created and **Message** window will appear to add start-up file of your device. Click on **Yes** so it will be added to your project folder you are programming in C and click on no if you are programming in assembly.

Step 4: Now go to File and create new file and save it with **.asm** extension in the folder where you created the project if you want to write program in assembly language

Step 5: Now write your program and save it again.

Step 6: After that on left you see project window [if it's not there….go to View tab and click on restore window to default].

Now come on Project window.

Right click on target and click on **options for target**

Here you can change your device also.

Step 7: Now Expand target and you will see source group

Right click on group and click on **Add files to source group**

Now add your program file which you have written in C/assembly.

You can see program file added under source group.

Step 8: Now Click on **Build target**. You can find it under Project tab or in toolbar. It can also be done by pressing **F7** key.

Step 9: you can see Status of your program in **Build output** window

[If it's not there go to view and click on Build output window]

Now you are done with your program.

Keil is a tool which can be used as debugger and simulator.

NOTE: *Use this tool for Experiment No. 1 and 2.*

T. Y. B. Tech (Electrical and Computer Engineering)

Trimester: V **Subject: Microcontroller and Applications**
Name: Shreerang Mhatre **Class: TY**

Roll No: 52 **Batch: A3**

Experiment No: 01 and 02

Name of the Experiment:
1. Simple Assembly language programming.
2. Complex Assembly language programming.

Performed on: 22/08/2023

Submitted on: 03/10/2023

Marks	Teacher's Signature with date

Aim:

1. Simple Assembly language programming.
   a. Write assembly language program for addition of two 8-bit numbers.
   b. Write assembly language program for addition of N 8-bit numbers. Take the input numbers from memory and store result in memory.

2. Complex Assembly language programming.
   a. Find square of a number using DPTR.

Theory:

Programming in the sense of Microcontrollers (or any computer) means writing a sequence of instructions that are executed by the processor in a particular order to perform a predefined task. The three levels of Programming Languages are as shown in figure 1.1.

Figure1.1: Programming Languages

Assembly Language is a pseudo-English representation of the Machine Language. The 8051 Microcontroller Assembly Language is a combination of English like words called Mnemonics and Hexadecimal codes with the following advantages:

- The Programs written in Assembly gets executed faster and they occupy less memory.
- With the help of Assembly Language, you can directly exploit all the features of a Microcontroller.
- Using Assembly Language, you can have direct and accurate control of all the Microcontroller's resources like I/O Ports, RAM, SFRs, etc.
- Compared to High-level Languages, Assembly Language has less rules and restrictions.

Syntax of CIP51 Instruction:

[Label:] Instructions [;Comments]

CIP51 has about 109 instructions. These can be grouped into the following categories

1. Arithmetic Instructions
2. Logical Instructions
3. Data Transfer instructions
4. Boolean Variable Instructions
5. Program Branching Instructions

The ADD instruction is used for the addition of two operands. The destination operand is always in register A, while the source operand can be register, immediate data or memory. The AF, CY and P bits of the flag register are affected by the ADD instruction depending on the operands. Program Status Word (PSW) is the flag register as shown in figure 1.2.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CY	AC	FO	RS1	RS0	OV	UD	P

Figure1.2: Bit fields in Program Status Word (PSW) Register

Table 1.1: Functions of flags

Symbol	**Function**
CY	Carry flag
AC	Auxiliary Carry flag (For BCD Operations)
F0	Flag 0 (Available to the user for General Purpose)

RS1, **RS0**	Register bank select: RS1 RS0 Working Register Bank and Address 0 0 Bank0 0 1 Bank1 1 0 Bank2 1 1 Bank3
OV	Overflow flag
UD	User definable flag
P	Parity flag; P=1 for odd no. of 1's; P=0 for Even no. of 1's

Algorithm:

Sample Example:

Program:

Note: *Attach the printout of the code.*

Conclusion:

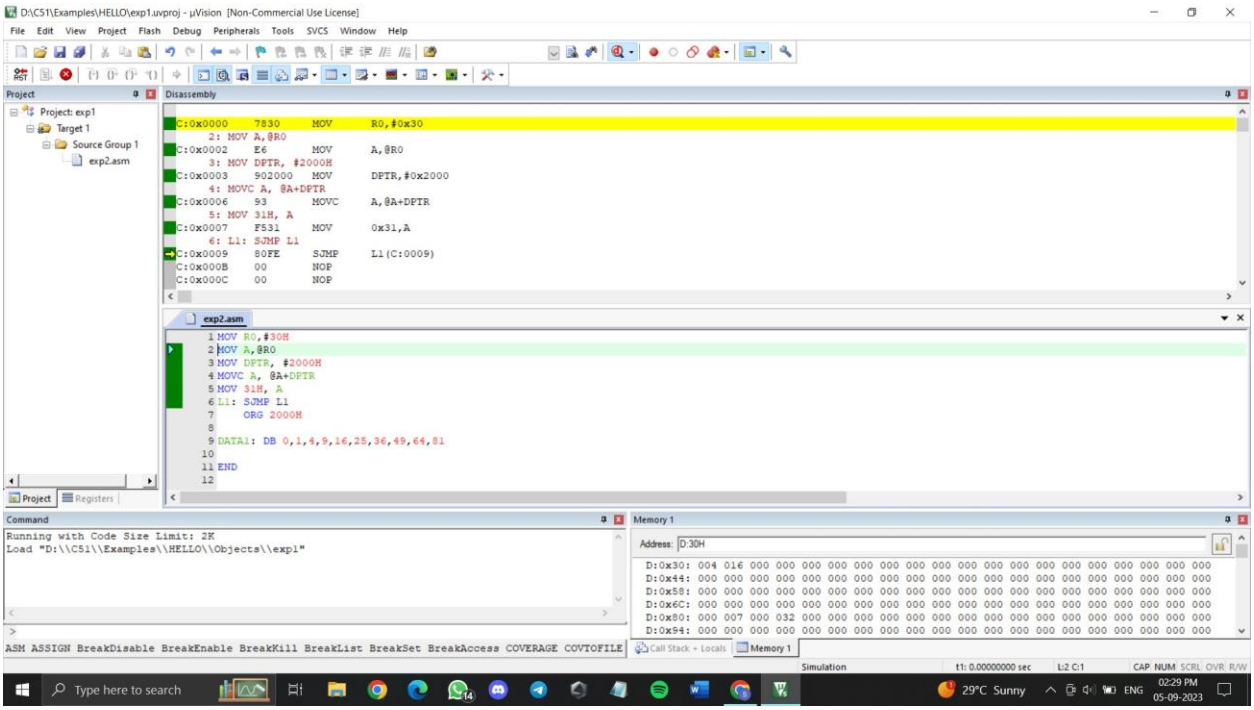
Study Question:

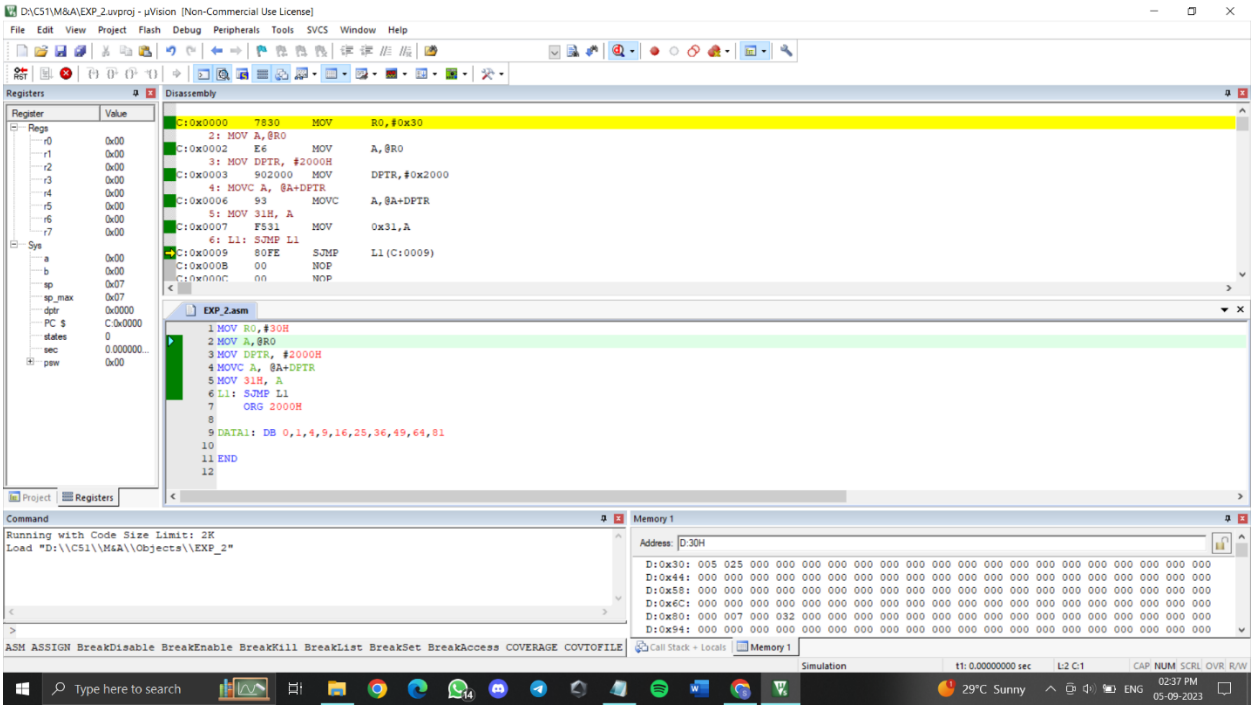
1. Explain the instructions: DJNZ, INC, SUBB.
2. What is an IDE?

Additional links:

1. *https://nptel.ac.in/courses/108105102*
2. *https://www.electronicshub.org/8051-microcontroller-assembly-language-programming/*

Practical-1


D:\C51\Examples\HELLO\exp1.uvproj - µVision [Non-Commercial Use License]
Disassembly
C:0x0000 7830 MOV R0,#0x30
2: MOV A,@R0
C:0x0002 E6 MOV A,@R0
3: MOV DPTR, #2000H
C:0x0003 902000 MOV DPTR,#0x2000
4: MOVC A, @A+DPTR
C:0x0006 93 MOVC A,@A+DPTR
5: MOV 31H, A
C:0x0007 F531 MOV 0x31,A
6: L1: SJMP L1
C:0x0009 80FE SJMP L1(C:0009)
C:0x000B 00 NOP
C:0x000C 00 NOP
exp2.asm
1 MOV R0,#30H
2 MOV A,@R0
3 MOV DPTR, #2000H
4 MOVC A, @A+DPTR
5 MOV 31H, A
6 L1: SJMP L1
7 ORG 2000H
9 DATA1: DB 0,1,4,9,16,25,36,49,64,81
11 END
Command
Running with Code Size Limit: 2K
Load "D:\\C51\\Examples\\HELLO\\Objects\\exp1"
Memory 1
Address: D:30H
D:0x30: 004 016 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
D:0x94: 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000


D:\C51\M&A\EXP_2.uvproj - µVision [Non-Commercial Use License]
Registers
Disassembly
C:0x0000 7830 MOV R0,#0x30
2: MOV A,@R0
C:0x0002 E6 MOV A,@R0
3: MOV DPTR, #2000H
C:0x0003 902000 MOV DPTR,#0x2000
4: MOVC A, @A+DPTR
C:0x0006 93 MOVC A,@A+DPTR
5: MOV 31H, A
C:0x0007 F531 MOV 0x31,A
6: L1: SJMP L1
C:0x0009 80FE SJMP L1(C:0009)
C:0x000B 00 NOP
EXP_2.asm
1 MOV R0,#30H
2 MOV A,@R0
3 MOV DPTR, #2000H
4 MOVC A, @A+DPTR
5 MOV 31H, A
6 L1: SJMP L1
7 ORG 2000H
9 DATA1: DB 0,1,4,9,16,25,36,49,64,81
11 END
Command
Running with Code Size Limit: 2K
Load "D:\\C51\\M&A\\Objects\\EXP_2"
Memory 1
Address: D:30H
D:0x30: 005 025 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000

22/8/23

PAGE No.
DATE 22/8/23

Exp 1

(3) Write assembly language program for addition of n numbers of 8 bit

Note- MOV D, S JC/JNC → Jump if no carry.
D → Destination, S → Source, JC/JNC → Jump if carry

```
                          initialize pointer to i/p
MOV R0, #50H ;  R0 = 50 H

MOV R1, #05 H ; Initialize Counter, R1 = 5H

MOV A, # 00H

MOV 55 H, #00H ; Initialize Carry=00

L2: ADD A, @R0 ; A = A + @R0 ⇔ Sum = Sum + a[i]
                A = 00 + 29H = 29H
                A = 29H + 35H = 64H
                A = 64H + 64H = ...
                :
carry → JNC L1
        INC 55H

L1: INC R0
    DJNZ R1, L2
    MOV 56H, A

L3: SJMP L3

    END
```

(General Purpose Resistor R0 -> R7)

8-bit Adder

Data Memory (8-bit)

00		
50 H	29 H	+1
51 H	35 H	+1
52 H	64 H	+1
53 H	56 H	+1
54 H	99 H	+1
55 H	Carry	+1
56 H	Sum	

Binary 8-bit addition

```
   1111 1111
   1111 1111
   1111 1111
+
[1] 1111 1110
carry   sum
```

Equivalent C code

```
# include <stdio.h>
int main () {
    int sum = 0;
    int arr[5] = { 10, 20, 30, 40, 50 };
    for (int i=0; i<5; i++)
    {
        sum = sum + arr[i];
    }
    printf (sum);
}
```

assembler

ALP → MLP

IDE
- Editor

Embedded C → ALP → MLP

~~Assembler~~ ~~compiler~~
compiler assembler

assembly language → machine language

assembler

embedded C → assembly language → machine language

- C8051F340

PAGE No.
DATE / /

Keil, microvision5

Project > new uvision project

Search your microcontroller > C8051F340

File > new > save as source group1 >
Add existing files to group 1 sourcegrp

select 'add.asm' > open add.asm

write the program. a.sm
save
Build target.
Debug the program
↳ start
↳ click memory 1 (Bottom right)
to access memory → d:40H → enter.
right click on memory.
Modify memory

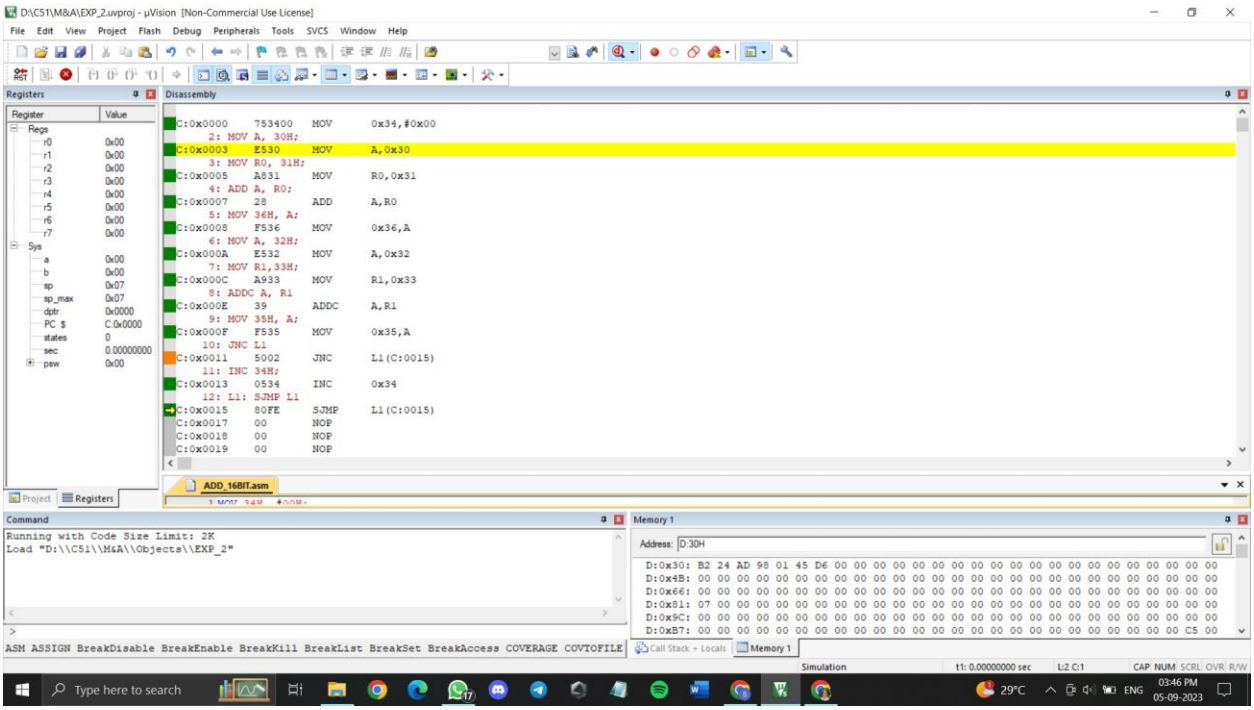
Eg –

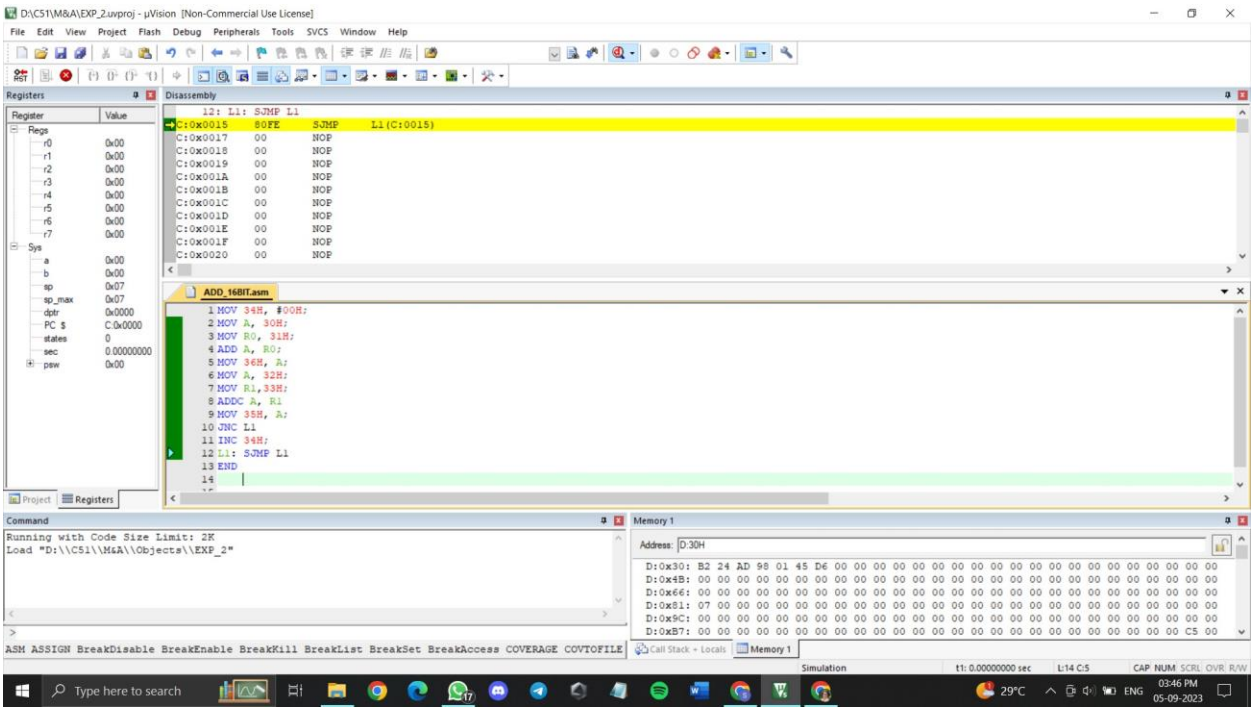
```
mov r0, #40h
mov a, #05h
Add a, @R0
MOV r1, A
mov 42h, a
end.
```

FFH, 18H, 56H, 7BH, DEH.

22/8/23

Practical-2


D:\C51\M&A\EXP_2.uvproj - µVision [Non-Commercial Use License]
Disassembly
C:0x0000 753400 MOV 0x34,#0x00
2: MOV A, 30H;
C:0x0003 E530 MOV A,0x30
3: MOV R0, 31H;
C:0x0005 A831 MOV R0,0x31
4: ADD A, R0;
C:0x0007 28 ADD A,R0
5: MOV 36H, A;
C:0x0008 F536 MOV 0x36,A
6: MOV A, 32H;
C:0x000A E532 MOV A,0x32
7: MOV R1,33H;
C:0x000C A933 MOV R1,0x33
8: ADDC A, R1
C:0x000E 39 ADDC A,R1
9: MOV 35H, A;
C:0x000F F535 MOV 0x35,A
10: JNC L1
C:0x0011 5002 JNC L1(C:0015)
11: INC 34H;
C:0x0013 0534 INC 0x34
12: L1: SJMP L1
C:0x0015 80FE SJMP L1(C:0015)
C:0x0017 00 NOP
C:0x0018 00 NOP
C:0x0019 00 NOP
ADD_16BIT.asm
Command
Running with Code Size Limit: 2K
Load "D:\\C51\\M&A\\Objects\\EXP_2"
Memory 1
Address: D:30H


D:\C51\M&A\EXP_2.uvproj - µVision [Non-Commercial Use License]
Disassembly
12: L1: SJMP L1
C:0x0015 80FE SJMP L1(C:0015)
C:0x0017 00 NOP
C:0x0018 00 NOP
C:0x0019 00 NOP
C:0x001A 00 NOP
C:0x001B 00 NOP
C:0x001C 00 NOP
C:0x001D 00 NOP
C:0x001E 00 NOP
C:0x001F 00 NOP
C:0x0020 00 NOP
ADD_16BIT.asm
1 MOV 34H, #00H;
2 MOV A, 30H;
3 MOV R0, 31H;
4 ADD A, R0;
5 MOV 36H, A;
6 MOV A, 32H;
7 MOV R1,33H;
8 ADDC A, R1
9 MOV 35H, A;
10 JNC L1
11 INC 34H;
12 L1: SJMP L1
13 END
Command
Running with Code Size Limit: 2K
Load "D:\\C51\\M&A\\Objects\\EXP_2"
Memory 1
Address: D:30H

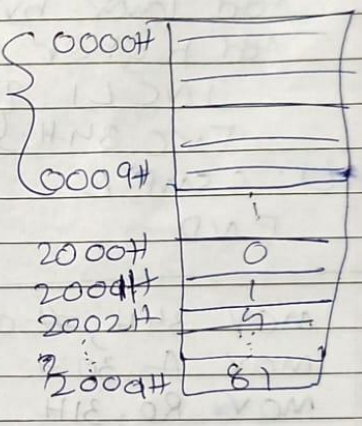
PAGE No.
DATE 5/9/23

Exp 2

Q1)

```
MOV  R0, #30H
MOV  A, @R0
MOV  DPTR, #2000H
MOVC A, @A+DPTR
MOV  31H, A
L1: SJMP L1
    ORG 2000H
Data1: DB 0, 1, 4, 9, 16, 25, 36, 49, 64, 81
END
```

ADH	B2H
98H	24H
45H	D6H



PAGE No.
DATE / /

Q2) Addn of two 16-bit nos

> ~~Initialize carry with zero~~

CY	
HB1	LB1
HB2	LB2

30H, 31H
↓ ↓
Add A, R0

Addc A, R1

HB(35H) LB(36H)

30H	LB1
31H	LB2
32H	HB1
33H	HB2
34H	Carry
35H	HB
36H	LB

4 7
5 9
106

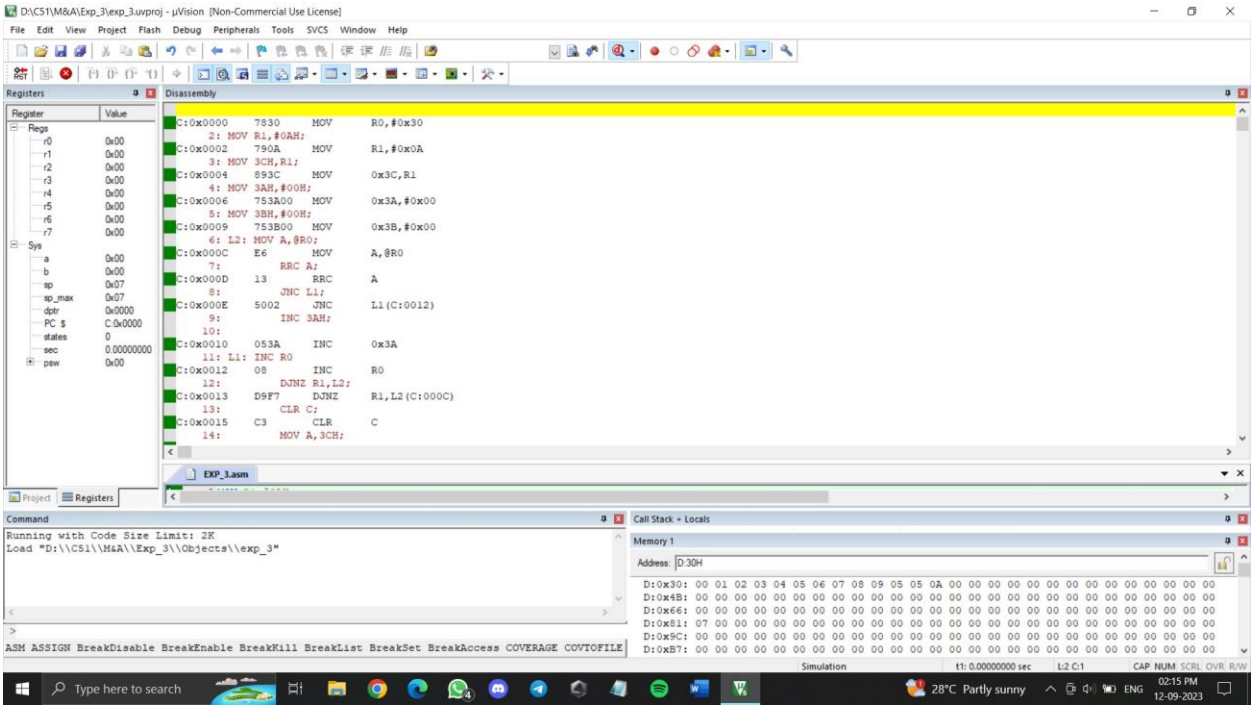
Initialize carry with zero
Add lower bytes & store result
Add higher — " —
JNC L1 } check for
INC 34H } cary &
L1: SJMP L — store if there is a carry
END

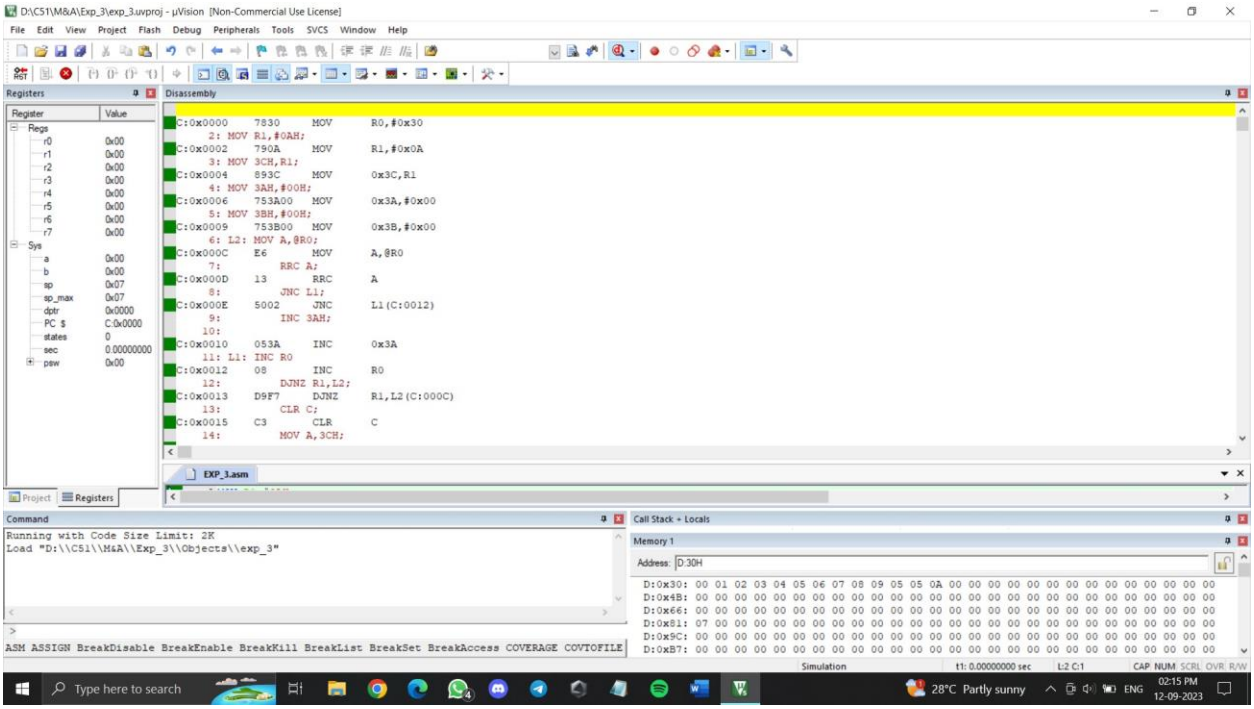
Add A, Rn
A, direct
A, #imm
A, @Ri

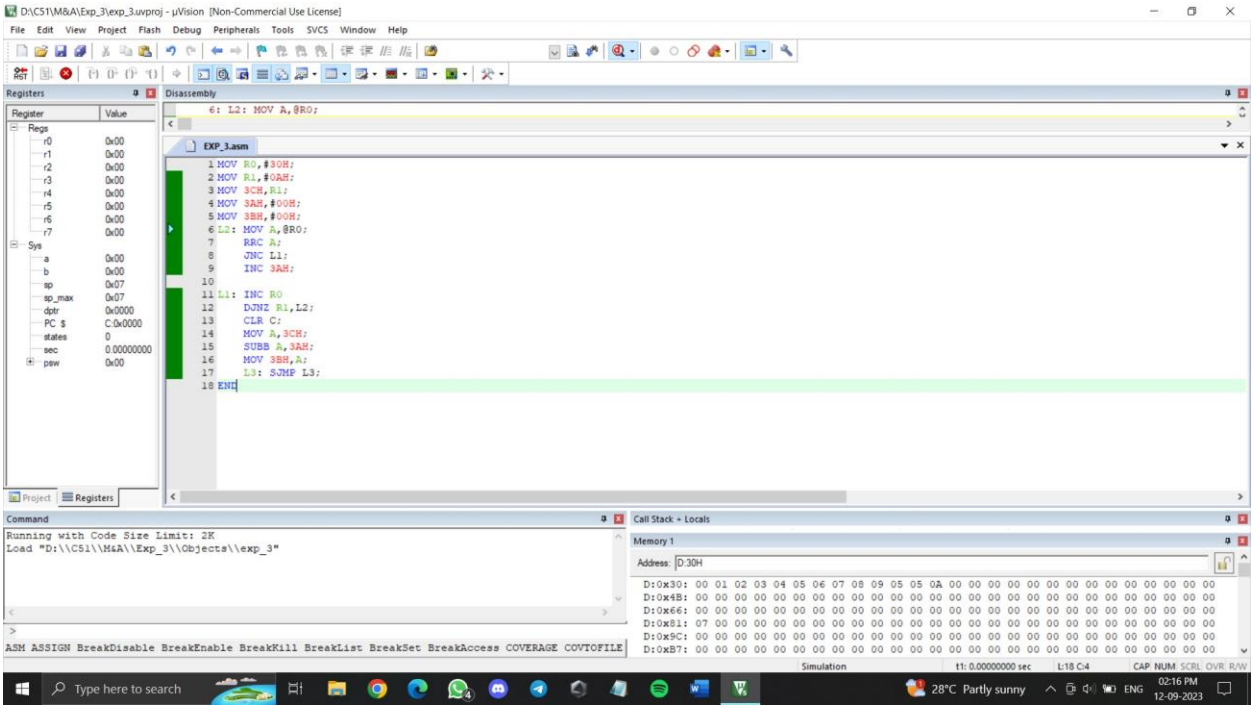
```
MOV 34H, #00H;
MOV A, 30H
MOV R0, 31H
Add A, R0
MOV 36H, A
MOV A, 32H
MOV R1, 33H
Addc A, R1
JNC L1
INC 34H
L1: SJMP L1
    END
```

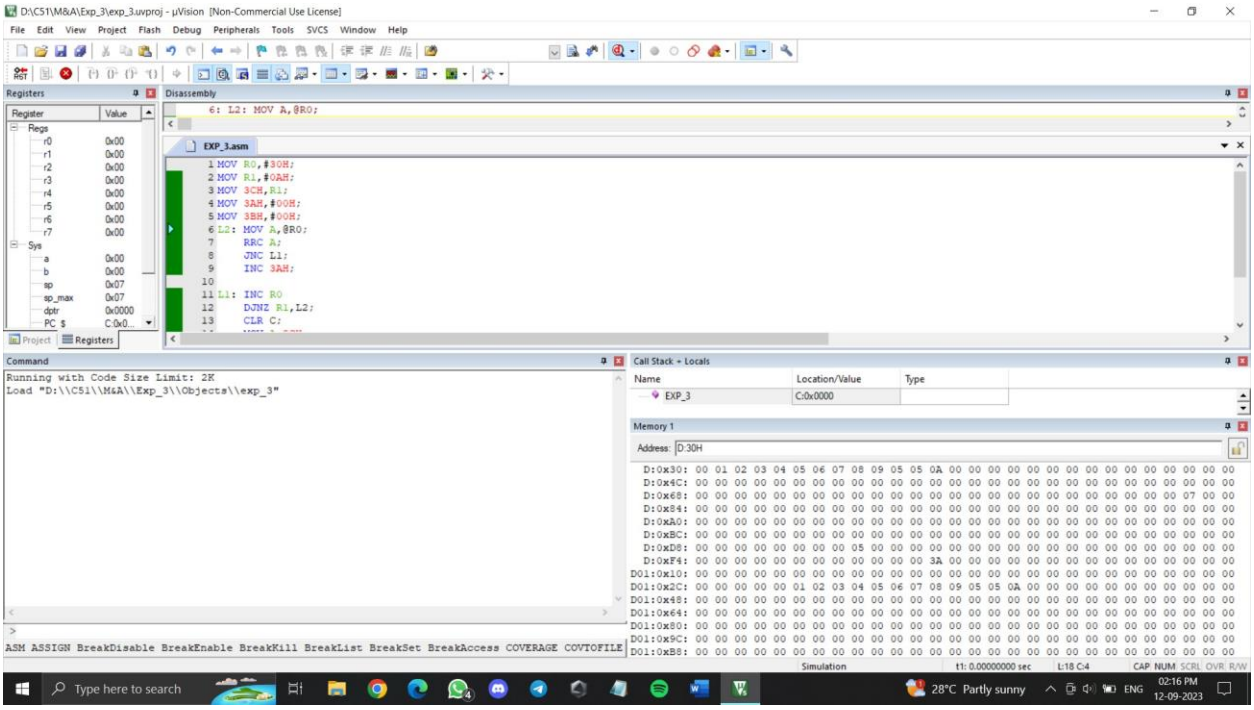
HD2
5/9/23

Practical-3


D:\C51\M&A\Exp_3\exp_3.uvproj - µVision [Non-Commercial Use License]
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
C:0x0000 7830 MOV R0,#0x30
2: MOV R1,#0AH;
C:0x0002 790A MOV R1,#0x0A
3: MOV 3CH,R1;
C:0x0004 893C MOV 0x3C,R1
4: MOV 3AH,#00H;
C:0x0006 753A00 MOV 0x3A,#0x00
5: MOV 3BH,#00H;
C:0x0009 753B00 MOV 0x3B,#0x00
6: L2: MOV A,@R0;
C:0x000C E6 MOV A,@R0
7: RRC A;
C:0x000D 13 RRC A
8: JNC L1;
C:0x000E 5002 JNC L1(C:0012)
9: INC 3AH;
10:
C:0x0010 053A INC 0x3A
11: L1: INC R0
C:0x0012 08 INC R0
12: DJNZ R1,L2;
C:0x0013 D9F7 DJNZ R1,L2(C:000C)
13: CLR C;
C:0x0015 C3 CLR C
14: MOV A,3CH;
EXP_3.asm
Command
Running with Code Size Limit: 2K
Load "D:\\C51\\M&A\\Exp_3\\Objects\\exp_3"
Call Stack + Locals
Memory 1
Address: D:30H
D:0x30: 00 01 02 03 04 05 06 07 08 09 05 05 0A
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE
Simulation t1: 0.00000000 sec L2:C1
Type here to search
28°C Partly sunny
02:15 PM
12-09-2023


D:\C51\M&A\Exp_3\exp_3.uvproj - µVision [Non-Commercial Use License]
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
C:0x0000 7830 MOV R0,#0x30
2: MOV R1,#0AH;
C:0x0002 790A MOV R1,#0x0A
3: MOV 3CH,R1;
C:0x0004 893C MOV 0x3C,R1
4: MOV 3AH,#00H;
C:0x0006 753A00 MOV 0x3A,#0x00
5: MOV 3BH,#00H;
C:0x0009 753B00 MOV 0x3B,#0x00
6: L2: MOV A,@R0;
C:0x000C E6 MOV A,@R0
7: RRC A;
C:0x000D 13 RRC A
8: JNC L1;
C:0x000E 5002 JNC L1(C:0012)
9: INC 3AH;
10:
C:0x0010 053A INC 0x3A
11: L1: INC R0
C:0x0012 08 INC R0
12: DJNZ R1,L2;
C:0x0013 D9F7 DJNZ R1,L2(C:000C)
13: CLR C;
C:0x0015 C3 CLR C
14: MOV A,3CH;
EXP_3.asm
Command
Running with Code Size Limit: 2K
Load "D:\\C51\\M&A\\Exp_3\\Objects\\exp_3"
Call Stack + Locals
Memory 1
Address: D:30H
D:0x30: 00 01 02 03 04 05 06 07 08 09 05 05 0A
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE
Simulation t1: 0.00000000 sec L2:C1
Type here to search
28°C Partly sunny
02:15 PM
12-09-2023


D:\C51\M&A\Exp_3\exp_3.uvproj - µVision [Non-Commercial Use License]
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
6: L2: MOV A,@R0;
EXP_3.asm
1 MOV R0,#30H;
2 MOV R1,#0AH;
3 MOV 3CH,R1;
4 MOV 3AH,#00H;
5 MOV 3BH,#00H;
6 L2: MOV A,@R0;
7 RRC A;
8 JNC L1;
9 INC 3AH;
10
11 L1: INC R0
12 DJNZ R1,L2;
13 CLR C;
14 MOV A,3CH;
15 SUBB A,3AH;
16 MOV 3BH,A;
17 L3: SJMP L3;
18 END
Command
Running with Code Size Limit: 2K
Load "D:\\C51\\M&A\\Exp_3\\Objects\\exp_3"
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE
Call Stack + Locals
Memory 1
Address: D:30H
D:0x30: 00 01 02 03 04 05 06 07 08 09 05 05 0A 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x4B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x66: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x81: 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x9C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0xB7: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Simulation
t1: 0.00000000 sec
L:18 C:4
CAP NUM SCRL OVR R/W
Type here to search
28°C Partly sunny
ENG
02:16 PM
12-09-2023


D:\C51\M&A\Exp_3\exp_3.uvproj - µVision [Non-Commercial Use License]
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
6: L2: MOV A,@R0;
EXP_3.asm
1 MOV R0,#30H;
2 MOV R1,#0AH;
3 MOV 3CH,R1;
4 MOV 3AH,#00H;
5 MOV 3BH,#00H;
6 L2: MOV A,@R0;
7 RRC A;
8 JNC L1;
9 INC 3AH;
10
11 L1: INC R0
12 DJNZ R1,L2;
13 CLR C;
Command
Running with Code Size Limit: 2K
Load "D:\\C51\\M&A\\Exp_3\\Objects\\exp_3"
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE
Call Stack + Locals
Name
Location/Value
Type
EXP_3
C:0x0000
Memory 1
Address: D:30H
D:0x30: 00 01 02 03 04 05 06 07 08 09 05 05 0A 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x4C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x68: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 07 00 00
D:0x84: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0xA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0xBC: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0xD8: 00 00 00 00 00 00 00 00 00 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0xF4: 00 00 00 00 00 00 00 00 00 00 00 00 3A 00 00 00 00 00 00 00 00 00 00 00 00 00
D01:0x10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D01:0x2C: 00 00 00 00 00 00 01 02 03 04 05 06 07 08 09 05 05 0A 00 00 00 00 00 00 00 00
D01:0x48: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D01:0x64: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D01:0x80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D01:0x9C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D01:0xB8: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Simulation
t1: 0.00000000 sec
L:18 C:4
CAP NUM SCRL OVR R/W
Type here to search
28°C Partly sunny
ENG
02:16 PM
12-09-2023

12/9/23

Exp 3

Q) Program to count the number of odd and even numbers from set of given 10 numbers.

```
MOV R0, #30H
MOV R1, #0AH
MOV R2, R1
MOV 3AH, #00H
MOV 3BH, #00H
L2: MOV A, @R0
RRC A
JNC L1
INC 3AH

L1: INC R0
DJNZ R1, L2
~~SUBB A, 3AH~~
CLR C
MOV A, R2
SUBB A, 3AH
MOV 3BH, A
L3: SJMP L3
```

30H	
31H	
⋮	
39H	
3AH	odd
3BH	even

ACC
RRC
7 0 → C

HOD
12/9/23

Q) Sort nos in ascending (1) & descending (2) order

Post Lab Questions:

Exp 1 & 2

PAGE No.
DATE / /

PLQ -

Q1) Explain the instruction: DJNZ, INC, SUBB

→ ① DJNZ (Decrement and Jump if Not zero):
- DJNZ is an assembly language instruction used for control flow in a program.
- It is often used in loops to decrement a register or memory location and then jump to a specified label or address if the result is not zero.

② INC (Increment):
- INC is an assembly language instruction used to increment the value of a register or memory location by one.
- It is a simple arithmetic operation
- The typical syntax of the INC instruction
  INC Operand.

③ SUBB (subtract with Borrow):
- SUBB is an assembly language instruction used to subtract one value from another, taking into account any borrow from a previous subtraction operation.
- It is often used when performing binary subtraction, especially in situations where borrow from the lower-order bits affects the subtraction of higher-order bits.

PAGE No.
DATE / /

Q2) What is an IDE?

→ An IDE, or Integrated Development Environment, is a software application or a suite of software tools that provides comprehesive facilities for computer programmers and software developers to write, test, and manage their software projects more efficiently. It's a one-stop-shop for various tasks involved in software development. IDEs typically include a combination of code editor, debugger, build automation tools, and more. Here are some key components and features commonly found in an IDE.