BECE204P-Microprocessors & Microcontrollers Lab

LAB-6

INTRODUCTION TO KEIL IDE & ASSEMBLY PROGRAMMING WITH ARITHMETIC INSTRUCTION OF 8051

INTRODUCTION TO KEIL IDE

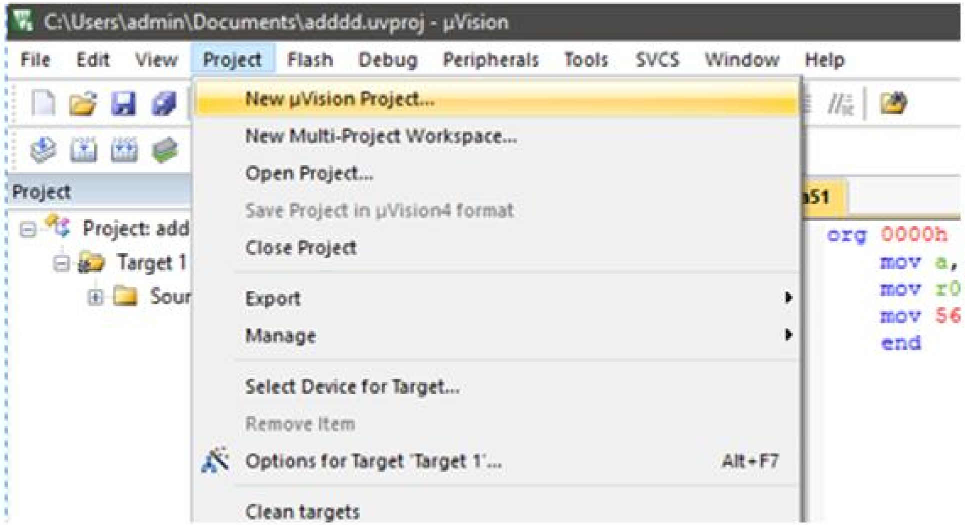
- Keil IDE: It is a software platform used to provide code editing, assembling and debugging capability
  - "Keil c51" is for 8051 microcontrollers
  - "Keil MDK-Arm" for ARM devices

- Step to run a program in Keil IDE:
  1. Create a Project (.uvproj)
  2. Create and write an assembly program (.a51)
  3. Build your project to check errors
  4. Select debug mode
  5. Run your code to verify the output

INTRODUCTION TO KEIL IDE

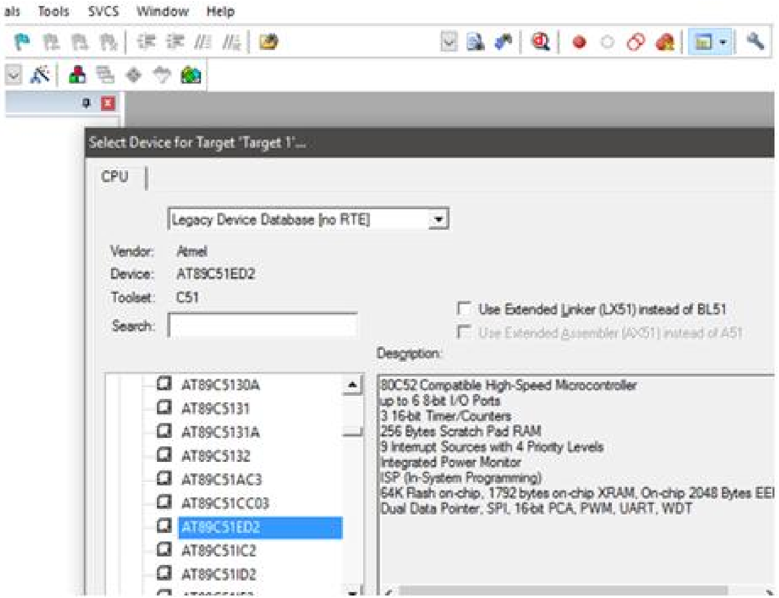
Step:1

- Create a Project (.uvproj) by selecting "Project -> New µvision Project"
- Save your project in D: drive with your register number as a file name
- Select device for target as "Atmel -> AT89C51ED2"
- Select "No" for adding STARTUP.A51 file to project



C:\Users\admin\Documents\adddd.uvproj - µVision

```
org 0000h
    mov a,
    mov r0
    mov 56
    end
```



Legacy Device Database [no RTE]

Vendor: Atmel

Device: AT89C51ED2

Toolset: C51

Search:

□ Use Extended Linker (LX51) instead of BL51

□ Use Extended Assembler (AX51) instead of A51

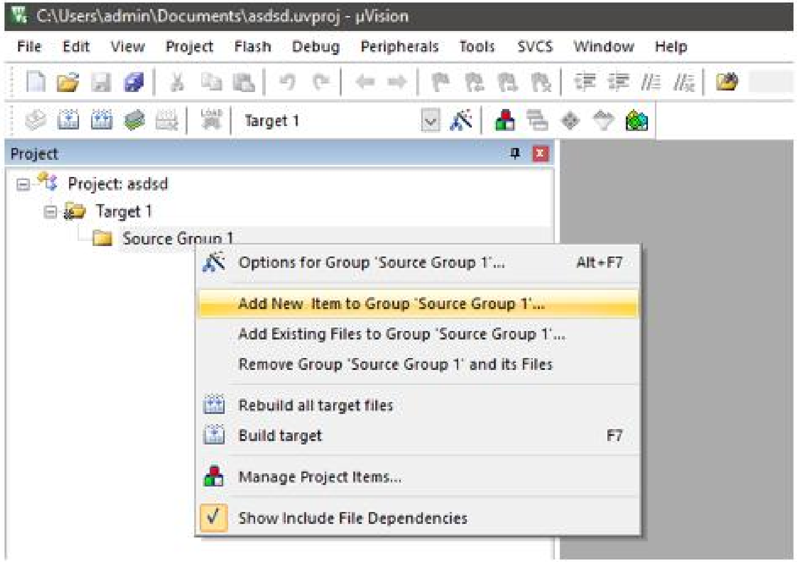
Description:

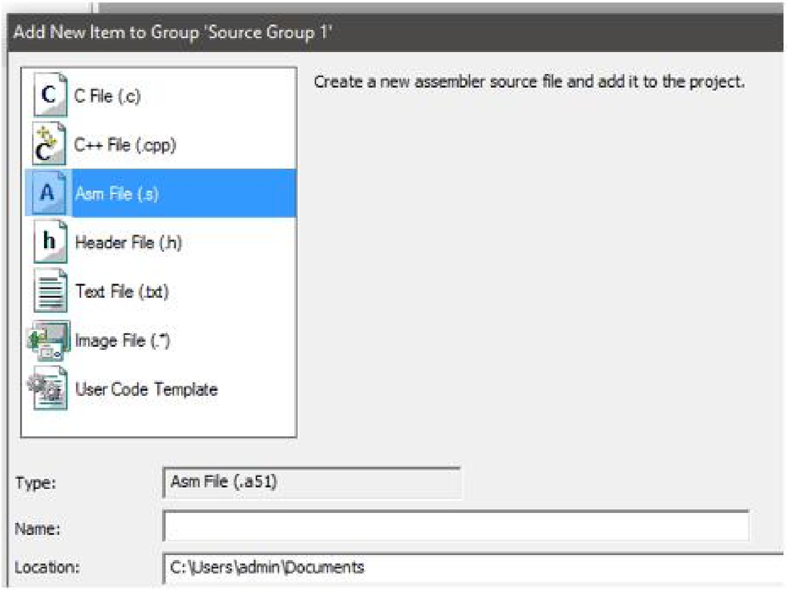
80C52 Compatible High-Speed Microcontroller
up to 6 8-bit I/O Ports
3 16-bit Timer/Counters
256 Bytes Scratch Pad RAM
9 Interrupt Sources with 4 Priority Levels
Integrated Power Monitor
ISP (In-System Programming)
64K Flash on-chip, 1792 bytes on-chip XRAM, On-chip 2048 Bytes EE
Dual Data Pointer, SPI, 16-bit PCA, PWM, UART, WDT

INTRODUCTION TO KEIL IDE

Step:2

- Create a asm file by right click on “Source Group 1) and select “Add items to Source group 1”
- Then select “asm file” option and give name for the asm file and save
- Write the asm program on the editor and save it


C:\Users\admin\Documents\asdsd.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Project: asdsd
Target 1
Source Group 1
Options for Group 'Source Group 1'... Alt+F7
Add New Item to Group 'Source Group 1'...
Add Existing Files to Group 'Source Group 1'...
Remove Group 'Source Group 1' and its Files
Rebuild all target files
Build target F7
Manage Project Items...
Show Include File Dependencies



Create a new assembler source file and add it to the project.

Type: Asm File (.a51)

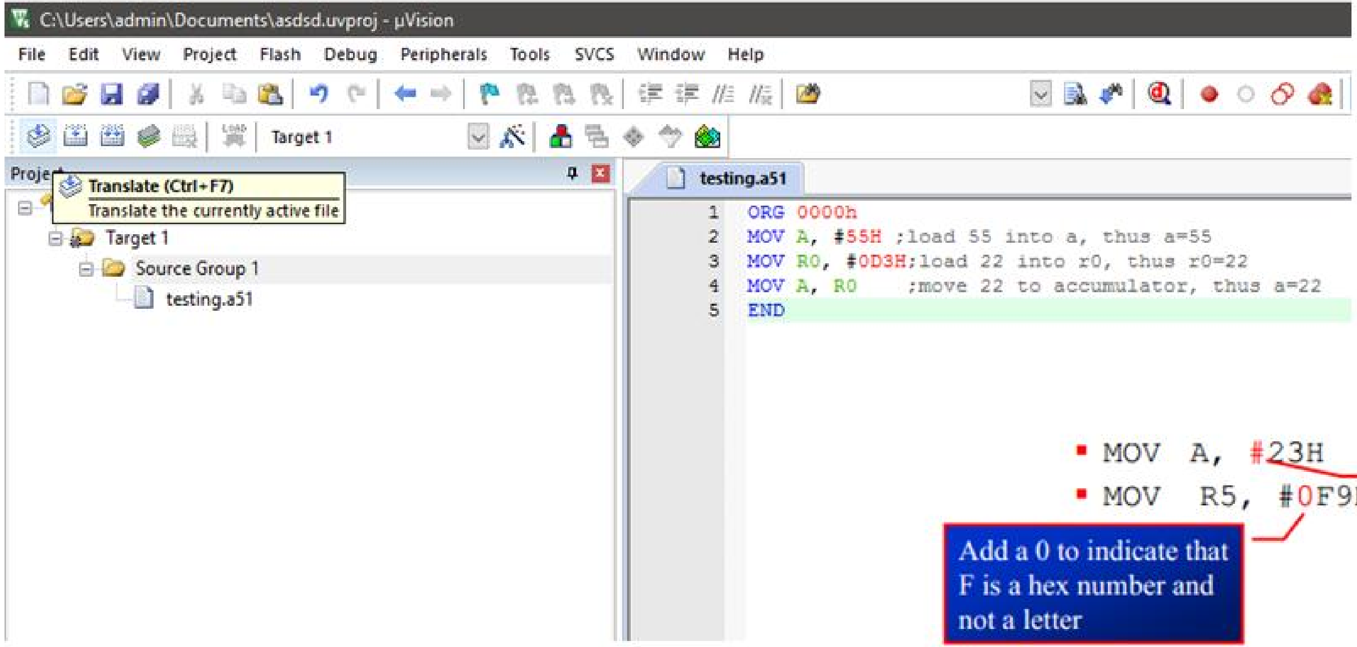
Name:

Location: C:\Users\admin\Documents

INTRODUCTION TO KEIL IDE

Step:3

- To verify errors in the program first select “Translate” option
- If any error, correct the error and save it before performing Translate option
- If no error then click on “Build” icon to generate supported files



```
ORG 0000h
MOV A, #55H ;load 55 into a, thus a=55
MOV R0, #0D3H;load 22 into r0, thus r0=22
MOV A, R0    ;move 22 to accumulator, thus a=22
END
```

- `MOV  A, #23H`
- `MOV  R5, #0F9H`

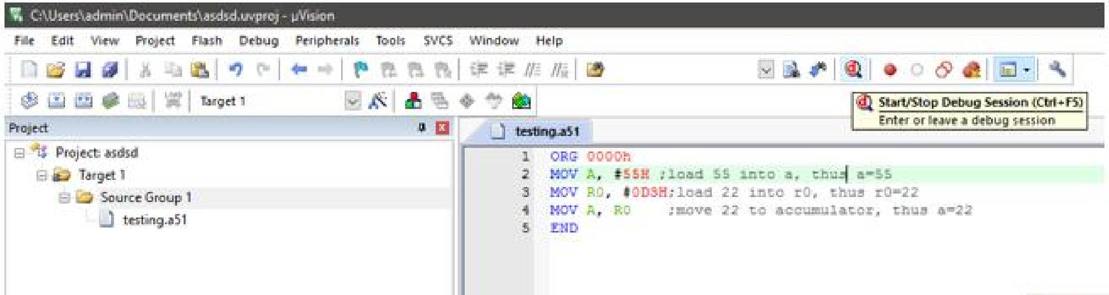
If it’s not preceded with #, it means to load from a memory location

Add a 0 to indicate that F is a hex number and not a letter

INTRODUCTION TO KEIL IDE

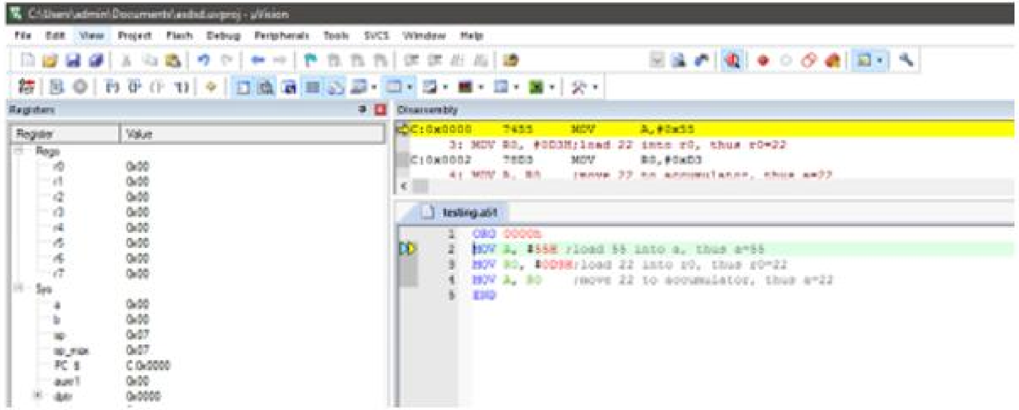
Step:4

- To run and verify your program first get into debug mode by selecting “start/stop debugging session” icon



Start/Stop Debug Session (Ctrl+F5)
Enter or leave a debug session

```
ORG 0000h
MOV A, #55H ;load 55 into a, thus a=55
MOV R0, #0D3H;load 22 into r0, thus r0=22
MOV A, R0     ;move 22 to accumulator, thus a=22
END
```



Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
PC $	C:0x0000
auxr1	0x00
dptr	0x0000

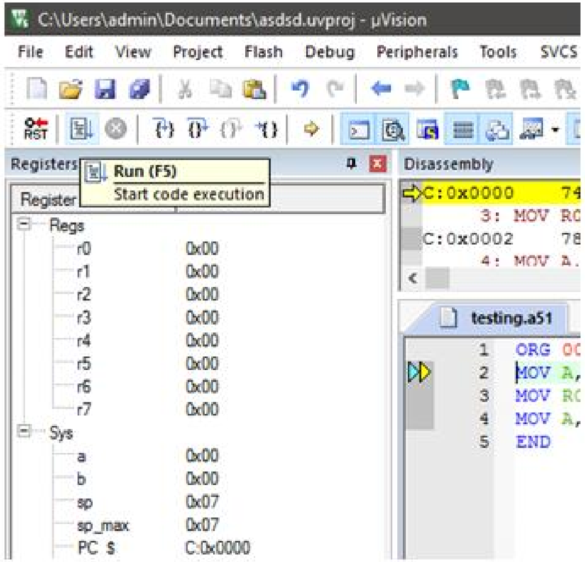
```
C:0x0000    7455     MOV      A,#0x55
     3: MOV R0, #0D3H;load 22 into r0, thus r0=22
C:0x0002    78D3     MOV      R0,#0xD3
     4: MOV A, R0     ;move 22 to accumulator, thus a=22
```

```
ORG 0000h
MOV A, #55H ;load 55 into a, thus a=55
MOV R0, #0D3H;load 22 into r0, thus r0=22
MOV A, R0     ;move 22 to accumulator, thus a=22
END
```

INTRODUCTION TO KEIL IDE

Step:5

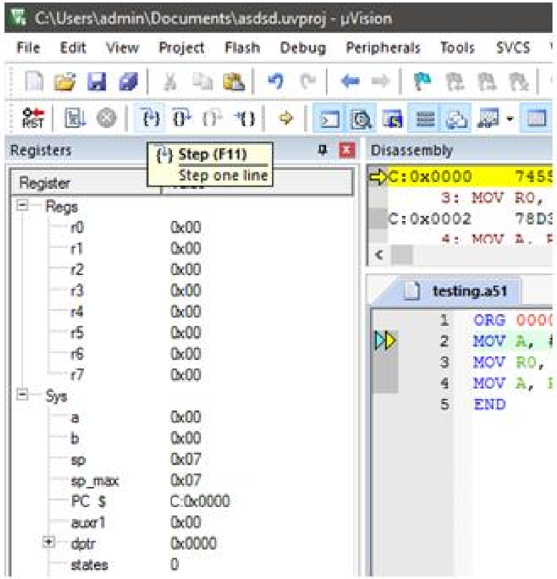
- To run and verify your program select run icon or press F5(Shows final output after executing entire program)
- To perform step by step execution of the program select "step" icon or F11 (shows output for every instruction execution)
- Analyze registers and memory locations to verify the correctness of the program



Register	
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
PC $	C:0x0000

```
C:0x0000    74
     3: MOV R0
C:0x0002    78
     4: MOV A.
```

```
ORG 00
MOV A,
MOV R0
MOV A,
END
```



Register	
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
PC $	C:0x0000
auxr1	0x00
dptr	0x0000
states	0

```
C:0x0000    7455
     3: MOV R0,
C:0x0002    78D3
     4: MOV A, R
```

```
ORG 000
MOV A, #
MOV R0,
MOV A, R
END
```

CHALLENGING TASKS

1. Write an 8051 ASM program to solve the following mathematical equation:

$$W=(Y+3Z-6X)/6D$$

Where D=03H, X=02H, Y=25H and Z=12H

2. Write an 8051 ASM program to solve the following mathematical equation:

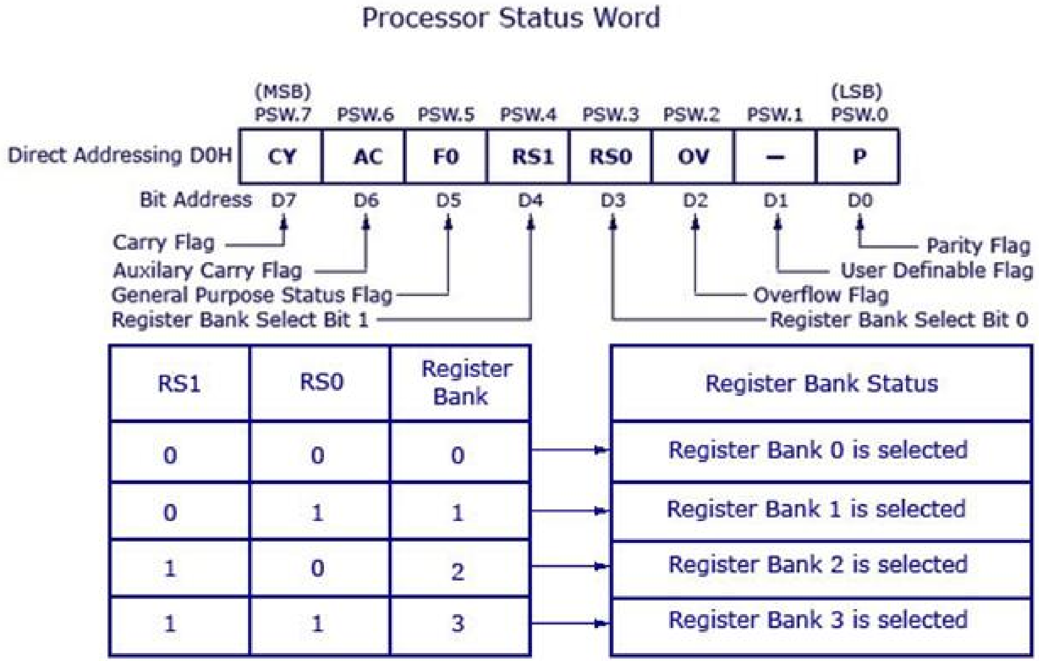
$$(a-b)^2 = a^2 + b^2 - 2ab$$

Where “a” & “b” are values at memory location 55H & 56H and store the result in 57H (High byte) & 58H (Low Byte). Assume “a” as first two digit and “b” as last two digit of your reg. No.

INSTRUCTIONS REQUIRED

ARITHMETIC INSTRUCTIONS

- List of arithmetic instructions in 8051:
  - ADD - ADDition
  - ADDC - ADDition with Carry
  - SUBB - SUBtraction with carry Borrow
  - MUL - MULtiply
  - DIV - DIVide
  - INC - INCrement
  - DEC – DECrement
  - DA - Decimal Adjust
  - CLR - CLeaR
  - CMP - CoMPlement
  - RL - Rotate Left
  - RLC – Rotate Left with Carry
  - RR – Rotate Right
  - RRC – Rotate Right with Carry
  - SWAP - SWAP



Processor Status Word

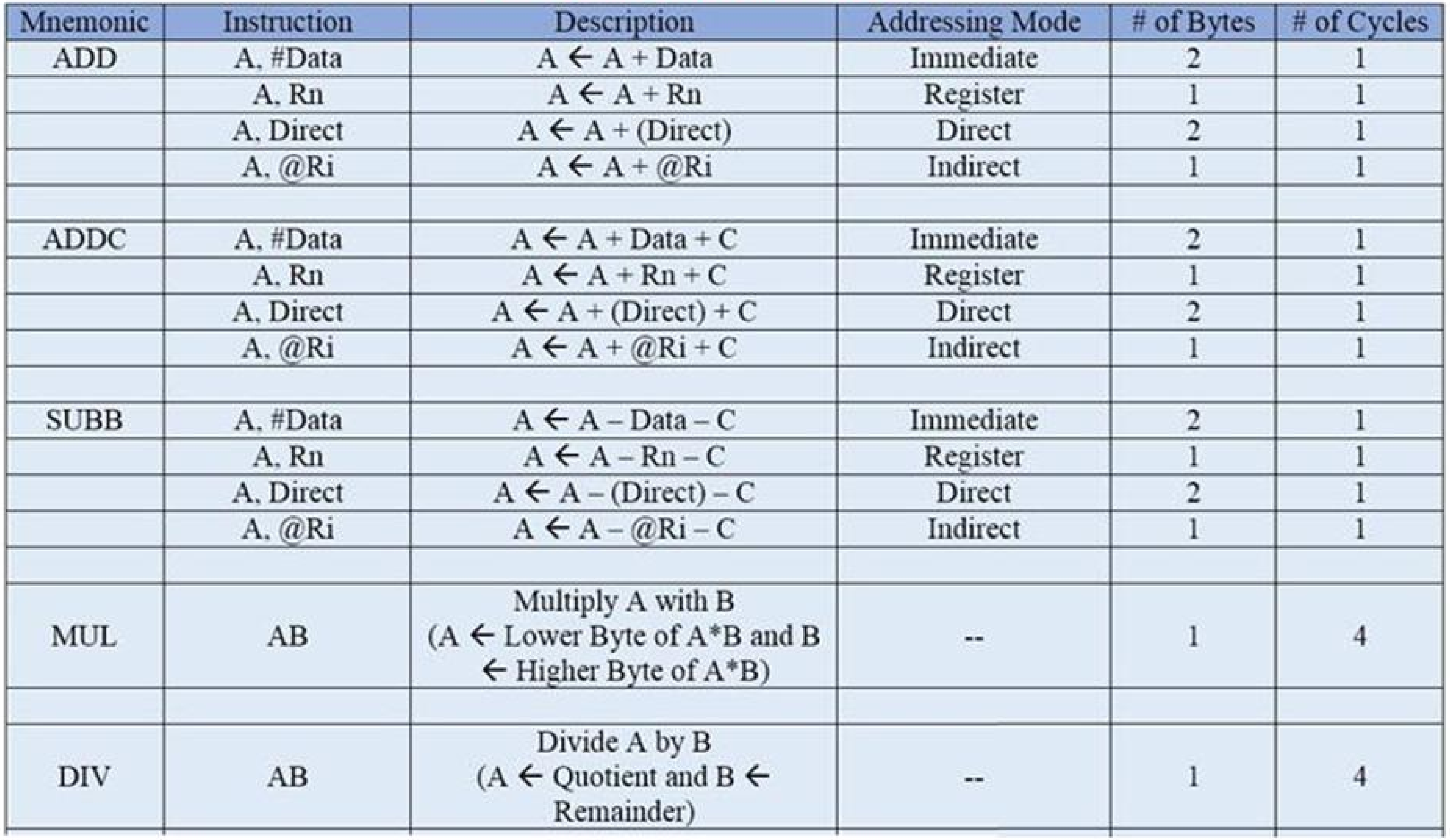
	(MSB) PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	(LSB) PSW.0
Direct Addressing D0H	CY	AC	F0	RS1	RS0	OV	–	P
Bit Address	D7	D6	D5	D4	D3	D2	D1	D0

Carry Flag
Auxilary Carry Flag
General Purpose Status Flag
Register Bank Select Bit 1
Parity Flag
User Definable Flag
Overflow Flag
Register Bank Select Bit 0

RS1	RS0	Register Bank	Register Bank Status
0	0	0	Register Bank 0 is selected
0	1	1	Register Bank 1 is selected
1	0	2	Register Bank 2 is selected
1	1	3	Register Bank 3 is selected

INSTRUCTIONS REQUIRED

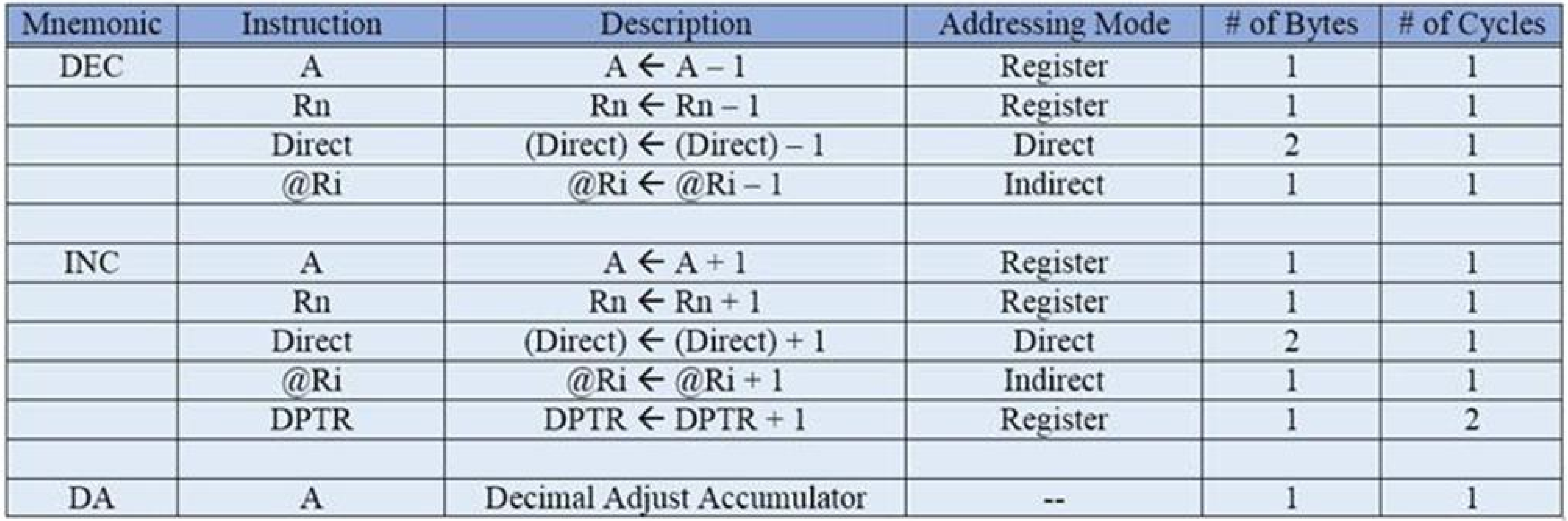
ARITHMETIC INSTRUCTIONS



Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
ADD	A, #Data	A ← A + Data	Immediate	2	1
	A, Rn	A ← A + Rn	Register	1	1
	A, Direct	A ← A + (Direct)	Direct	2	1
	A, @Ri	A ← A + @Ri	Indirect	1	1
ADDC	A, #Data	A ← A + Data + C	Immediate	2	1
	A, Rn	A ← A + Rn + C	Register	1	1
	A, Direct	A ← A + (Direct) + C	Direct	2	1
	A, @Ri	A ← A + @Ri + C	Indirect	1	1
SUBB	A, #Data	A ← A – Data – C	Immediate	2	1
	A, Rn	A ← A – Rn – C	Register	1	1
	A, Direct	A ← A – (Direct) – C	Direct	2	1
	A, @Ri	A ← A – @Ri – C	Indirect	1	1
MUL	AB	Multiply A with B (A ← Lower Byte of A*B and B ← Higher Byte of A*B)	--	1	4
DIV	AB	Divide A by B (A ← Quotient and B ← Remainder)	--	1	4

INSTRUCTIONS REQUIRED

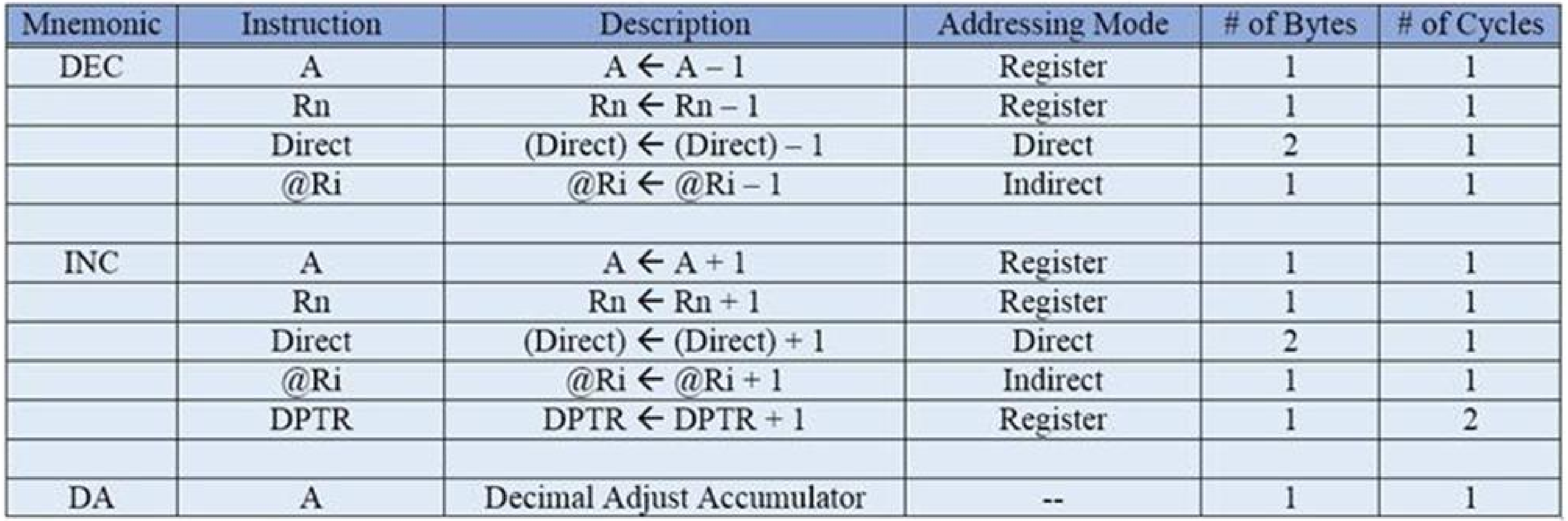
ARITHMETIC INSTRUCTIONS



Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
DEC	A	A ← A – 1	Register	1	1
	Rn	Rn ← Rn – 1	Register	1	1
	Direct	(Direct) ← (Direct) – 1	Direct	2	1
	@Ri	@Ri ← @Ri – 1	Indirect	1	1
INC	A	A ← A + 1	Register	1	1
	Rn	Rn ← Rn + 1	Register	1	1
	Direct	(Direct) ← (Direct) + 1	Direct	2	1
	@Ri	@Ri ← @Ri + 1	Indirect	1	1
	DPTR	DPTR ← DPTR + 1	Register	1	2
DA	A	Decimal Adjust Accumulator	--	1	1

INSTRUCTIONS REQUIRED

ARITHMETIC INSTRUCTIONS



Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
DEC	A	A ← A – 1	Register	1	1
	Rn	Rn ← Rn – 1	Register	1	1
	Direct	(Direct) ← (Direct) – 1	Direct	2	1
	@Ri	@Ri ← @Ri – 1	Indirect	1	1
INC	A	A ← A + 1	Register	1	1
	Rn	Rn ← Rn + 1	Register	1	1
	Direct	(Direct) ← (Direct) + 1	Direct	2	1
	@Ri	@Ri ← @Ri + 1	Indirect	1	1
	DPTR	DPTR ← DPTR + 1	Register	1	2
DA	A	Decimal Adjust Accumulator	--	1	1

MOV destination, source ;copy source to dest.

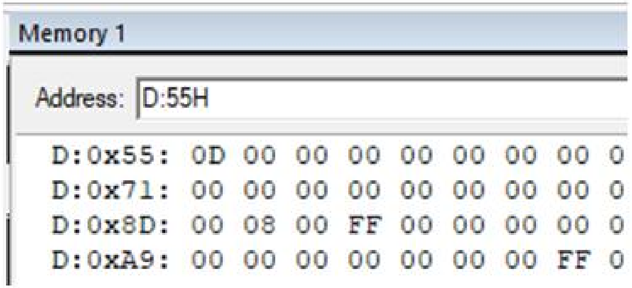
- The instruction tells the CPU to move (in reality, COPY) the source operand to the destination operand
- Source and destination can be A, R0 to R7, direct address, direct data(#), indirect address(@), DPTR

LAB TASK-1

Write an 8051 ASM program to perform addition of two 8-bit numbers 97H and 76H and store the result at address location 55H.

```
ORG 0000H
MOV  A, #97H                    ; 97H - 1001 0111
ADD A, #76H                     ;  76H -0111 0110
MOV  55H, A                   ; 1  0DH -  1  0000 1101
END
```

OUTPUT:



Address: D:55H

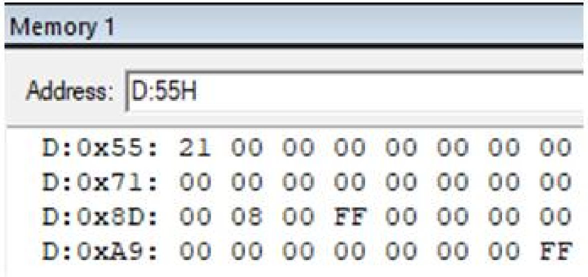
```
D:0x55: 0D 00 00 00 00 00 00 00 0
D:0x71: 00 00 00 00 00 00 00 00 0
D:0x8D: 00 08 00 FF 00 00 00 00 0
D:0xA9: 00 00 00 00 00 00 00 FF 0
```

LAB TASK-2

Write an 8051 ASM program to perform subtraction of two 8-bit numbers 76H and 97H and store the result at address location 55H.

```
ORG 0000H
MOV  A, #97H          ; 97H -  1001 0111
SUBB A, #76H          ; 76H -  0111 0110
MOV  55H, A         ;0  21H -  0 0010 0001
END
```

OUTPUT:



Address: D:55H

```
D:0x55: 21 00 00 00 00 00 00 00
D:0x71: 00 00 00 00 00 00 00 00
D:0x8D: 00 08 00 FF 00 00 00 00
D:0xA9: 00 00 00 00 00 00 00 FF
```

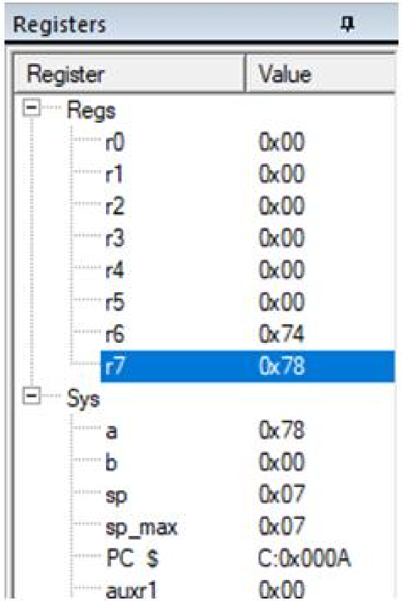
LAB TASK-3

Write an 8051 ASM program to perform addition of two 16-bit numbers. The numbers are 3CE7H and 3B8DH. Place the sum in R7 and R6; R6 should have the lower byte.

```
ORG 0000H
MOV  A, #0E7H
ADD A, #8DH
MOV  R6, A
MOV  A, #3CH
ADDC A,#3BH
MOV  R7, A
END
```

```
      1
     3C  E7
+    3B  8D
     ------
     78  74
```

OUTPUT:



```
Registers
Register        Value
⊟ Regs
    r0          0x00
    r1          0x00
    r2          0x00
    r3          0x00
    r4          0x00
    r5          0x00
    r6          0x74
    r7          0x78
⊟ Sys
    a           0x78
    b           0x00
    sp          0x07
    sp_max      0x07
    PC  $       C:0x000A
    auxr1       0x00
```

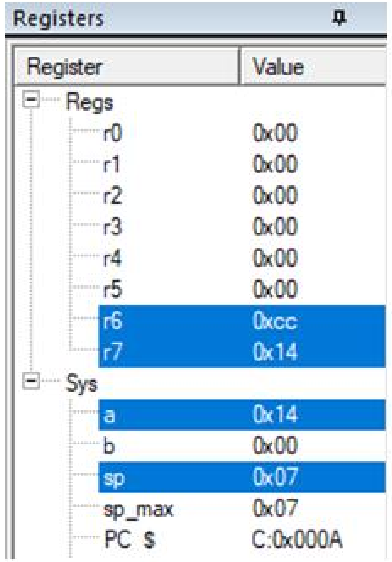
LAB TASK-4

Write an 8051 ASM program to perform subtraction of two 16-bit numbers. The numbers are 2762H and 1296H. Place the sum in R7 and R6; R6 should have the lower byte.

```
ORG 0000H
MOV  A, #0E7H
ADD A, #8DH
MOV  R6, A
MOV  A, #3CH
ADDC A,#3BH
MOV  R7, A
END
```

Ans: 14CCH

OUTPUT:



```
Registers
Register          Value
⊟ Regs
    r0            0x00
    r1            0x00
    r2            0x00
    r3            0x00
    r4            0x00
    r5            0x00
    r6            0xcc
    r7            0x14
⊟ Sys
    a             0x14
    b             0x00
    sp            0x07
    sp_max        0x07
    PC  $         C:0x000A
```

LAB TASK-5

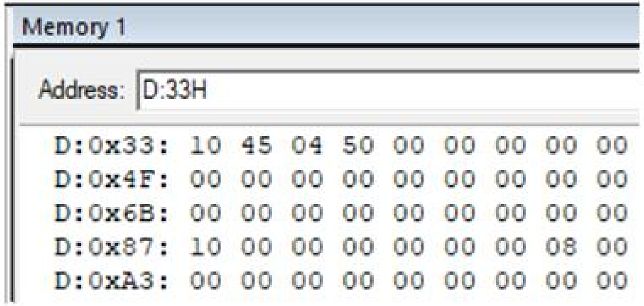
Write an 8051 ASM program to perform multiplication of two 8-bit numbers present in data memory address location 33H & 34H and store the result in 35H (higher byte) & 36H (Lower byte).

MUL AB; AxB, place 16-bit result in B and A

Multiplication	Operand-1	Operand-2	result
Byte x Byte	A	B	A= Low Byte, B=High Byte

```
ORG 0000H
MOV  A, 33H
MOV  B, 34H
MUL AB
MOV  35H, B
MOV  36H, A
END
```

OUTPUT:



Memory 1

Address: D:33H

```
D:0x33:  10  45  04  50  00  00  00  00  00
D:0x4F:  00  00  00  00  00  00  00  00  00
D:0x6B:  00  00  00  00  00  00  00  00  00
D:0x87:  10  00  00  00  00  00  00  08  00
D:0xA3:  00  00  00  00  00  00  00  00  00
```

LAB TASK-6

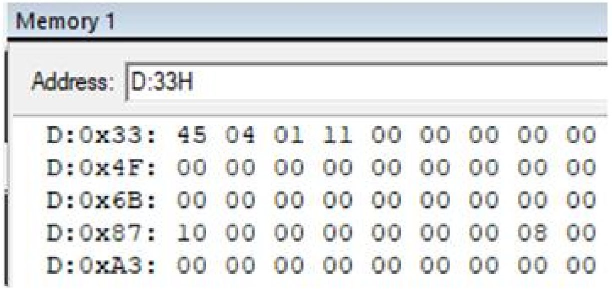
Write an 8051 ASM program to perform division on 8-bit numbers present in data memory address location 33H & 34H and store the result in 35H (Reminder) & 36H (Quotient).

DIV AB; A/B, place Quotient in A, Reminder in B

Division	Numerator	Denominator	Result
Byte / Byte	A	B	A= Quotient , B=Reminder

```
ORG 0000H
MOV A, 33H
MOV B, 34H
DIV AB
MOV 35H, B
MOV 36H, A
END
```

OUTPUT:



Address: D:33H

```
D:0x33: 45 04 01 11 00 00 00 00 00
D:0x4F: 00 00 00 00 00 00 00 00 00
D:0x6B: 00 00 00 00 00 00 00 00 00
D:0x87: 10 00 00 00 00 00 00 08 00
D:0xA3: 00 00 00 00 00 00 00 00 00
```