**DEPT. OF ELECTRICAL & ELECTRONICS ENGINEERING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, Kattankulathur – 603203.**

|  |  |
| --- | --- |
| Title of Experiment | : Arithmetic operation - 8051 |
| Name of the candidate | : GAUTAM NAG |
| Register Number | : RA1811005010278 |
| Date of Experiment | : 04/02/2020 |
| Date of submission | **: 05/02/2021** |

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO**  **:** | **MARKS SPLIT UP** | **MAXIMUM MARKS (50)** | **MARKS OBTAINED** |
| 1 | PRE LAB | 5 |  |
| 2 | PROGRAM | 25 |  |
| 3 | EXECUTION | 15 |  |
| 4 | POST LAB | 5 |  |
| TOTAL | | 50 |  |

**Staff Signature**

# PRE-LAB

1. **Specify the number of registers in a 2K memory chip?**

1kb=1024 byte is multiply by the total memory which is given to you as bellow: (1024\*2) = 2048 register

# What is an assembler?

a program for converting instructions written in low-level symbolic code into machine code.

# What are the advantages of an assembly language in comparison with high level language?

Assembly language can control the machine code better as compared to high level languages. Manipulation of bits is easier in assembly language as compared to high level languages. Assembly language can access any memory but the high level languages can't.

# List the components of computer?

A motherboard.

A Central Processing Unit (CPU)

A Graphics Processing Unit (GPU), also known as a video card. Random Access Memory (RAM), also known as volatile memory. Storage: Solid State Drive (SSD) or Hard Disk Drive (HDD)

# What is an operating system?

An operating system is system software that manages computer hardware, software resources, and provides common services for computer programs.

**Aim:**

**2. ARTHMETIC OPERATIONS USING 8051**

To do the arithmetic operations using 8051 microprocessor

**Apparatus required:**

**Hardware Requirement :**

8051 Microcontroller kit, Power supply

**Software Requirement :**

8051 EdSim

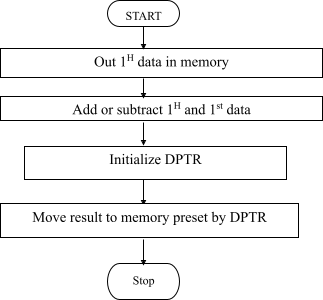
**Algorithm:**

**Addition / Subtraction**

Step 1 : Move 1H data to memory

Step 2 : Add or subtract 1H data with 2nd data Step 3 : Initialize data pointer.

Step 4 : Move result to memory pointed by DPTR.



**Multiplication / Division**

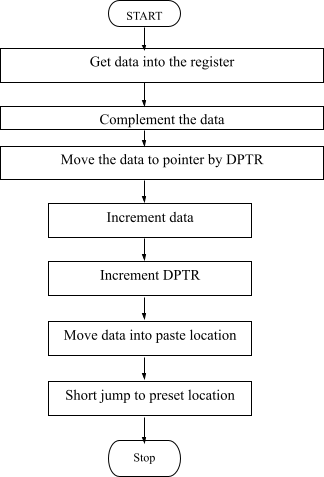
Step 1 : Get 1H data and 2nd data to memory

Step 2 : Multiply or divide 1H data with 2nd data Step 3 : Initialize data pointer.

Step 4 : Move result to memory pointed by DPTR (first port) Step 5 : Increment DPTR

Step 6 : Move 2nd part of result to register A

Step 7 : Move result to 2nd memory location pointer by DPTR



**Program: 8-bit Addition:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Memory**  **Location** | **Label** | **Opcode** | **Mnemonics** | **Comments** |
| 0000 | **START** | 74 01 | MOV A, #01 | Moves data 1 to  register A |
| 0002 | **START** | 24 02 | ADD A, #02 | Add content of A and  data 2 and store in A |
| 0005 | **SUBS** | 90 45 00 | MOV DPTR,#4500 | Moves data 4500 to DPTR |
| 0007 | **STORE** | F0 | MOVX @DPTR,A | Moves control of A to location pointed DTPR |

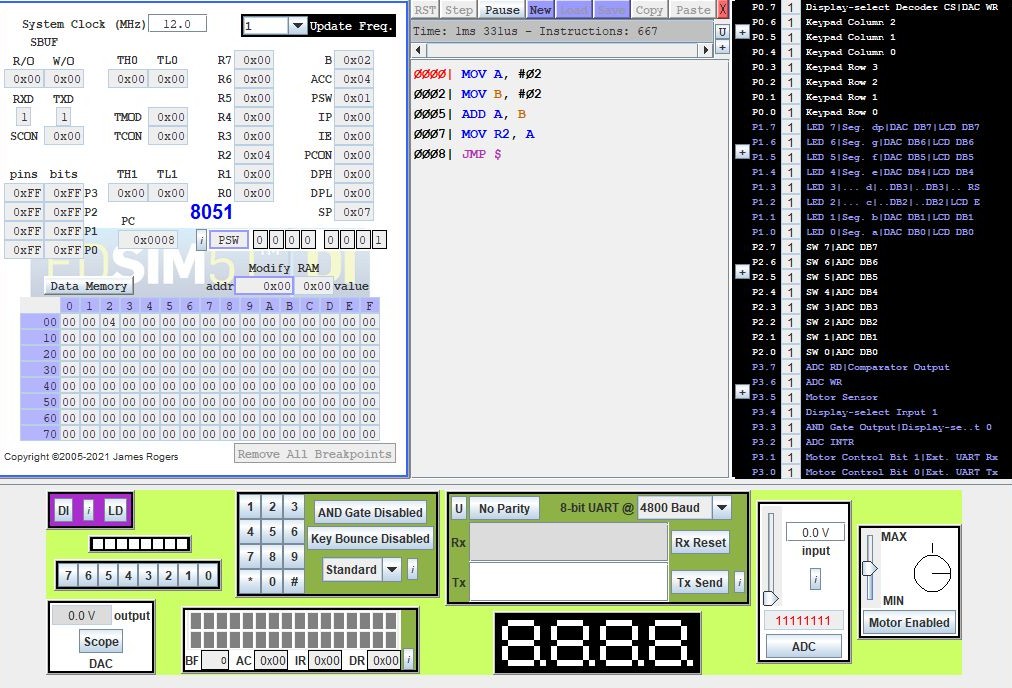
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0008 |  | 80 FE | SJMP 4108 | Short jump to 4108 |

**EDSIM51 PROGRAM-ADDITION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ADDRESS** | **LABEL** | **MNEMONICS** | **OPCODE** | **COMMENTS** |
| 0000 | **START** | MOV A, #Data | 74 01 | Move a data (8 bit) to Acc |
| 0002 | **START** | MOV B, #Data | 24 02 | Move a data (8 bit) to B Reg |
| 0005 | **SUBS** | ADD A, B | 90 45 00 | Add them, and store result in A |
| 0007 | **STORE** | MOV R2, A | F0 | Copy the result to a Reg |
| 0008 | **RUN** | JMP $ | 74 01 | Jump to Address location |

|  |  |
| --- | --- |
| **INPUT ADDRESS** | **DATA** |
| **0000** | **02** |
| **0002** | **02** |
| **OUTPUT ADDRESS** | **DATA** |
| **R2** | **04** |
| **ACC** | **04** |

**SIMULATION**



**Program: 8-bit Subtraction:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Memory**  **Location** | **Label** | **Opcode** | **Mnemonics** | **Comments** |
| 0000 | **START** | 74 05 | MOV A,#05 | Moves data 1 to  register A |
| 0002 | **START** | 94 02 | SUBB A,#02 | Subtract data 2 from content of A and store  result in A |
| 0005 | **SUBS** | 90 45 00 | MOV DPTR,#4500 | Moves 4500 to DPTR |
| 0007 | **STORE** | F0 | MOVX @DPTR,A | Moves result by  location by DTPR |
| 0008 | **RUN** | 80 FE | SJMP 4108 | Short jump to 4108 |

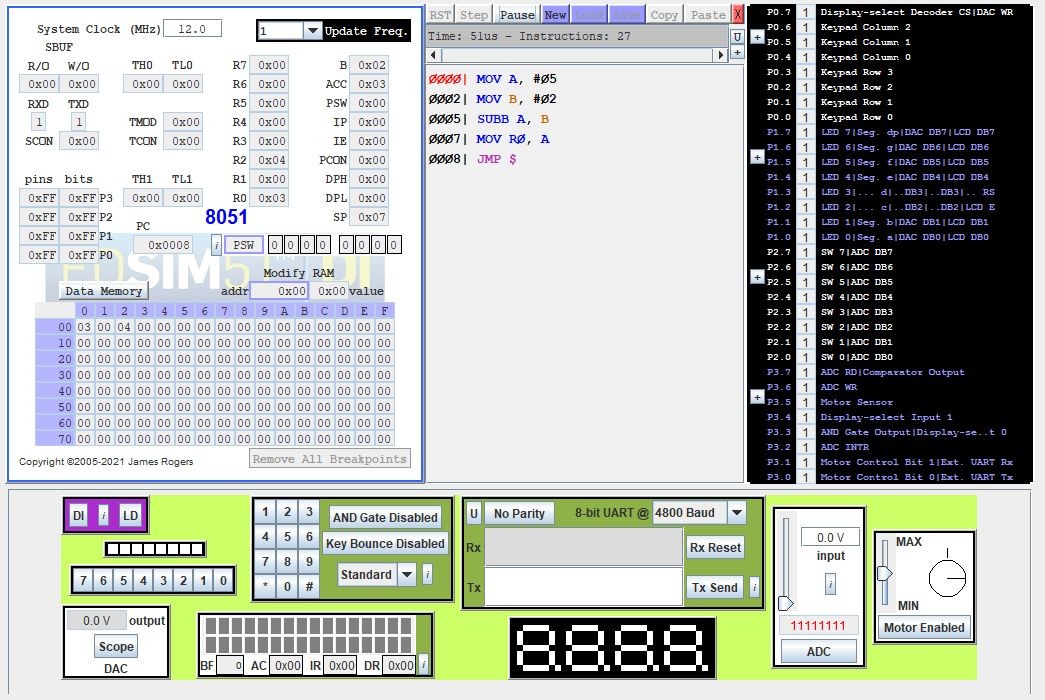
**EDSIM51 PROGRAM-SUBTRACTION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ADDRESS** | **LABEL** | **MNEMONICS** | **OPCODE** | **COMMENTS** |
| 0000 | **START** | MOV A, #Data | 74 05 | **Move a data (8 bit) to Acc** |
| 0002 | **START** | MOV B, #Data | 94 02 | **Move a data (8 bit) to B Reg** |
| 0005 | **SUBS** | SUBB A, B | 90 45 00 | **Subtract them, and store result in A** |
| 0007 | **STORE** | MOV R0, A | F0 | **Copy the result to a Reg** |
| 0008 | **RUN** | JMP $ | 80 FE | **Jump to Address location** |

|  |  |
| --- | --- |
| **IN PUT ADDRESS** | **DATA** |
| 0000 | **05** |
| 0002 | **02** |
| **OUT PUT ADDRESS** | **DATA** |
| 4500 | **03** |
| **R0** | **03** |

**SIMULATION**

**Program: 8-bit Multiplication:**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Memory**  **Location** | **Label** | **Opcode** | **Mnemonics** | **Comments** |
| 4100 | **START** | 74 03 | MOV A,#03 | Move immediate data  to accumulator |
| 4102 | **START** | 75 F0 02 | MOV B,#02 | Move 2nd data to B register |
| 4105 | **MULTI PLY** | A4 | MUL AB | Get the product in A & B |
| 4106 | **STORE** | 90 45 00 | MOV DPTR, # 4500 | Load data in 4500 location |
| 4109 | **STORE** | F0 | MOVX @DPTR,A | Move A t ext RAM |
| 410A | **RUN** | A3 | INC DPTR |  |
| 410B | **START** | E5 F0 | MOV A,B | Move 2nd data in A |
| 410D | **STORE** | F0 | MOVX @DPTR,A | Same the ext RAM |
| 410E | **JMP** | 80 FE | SJMP 410E | Remain idle in infinite  loop |

**Execution:**

**Multiplication:**

|  |  |
| --- | --- |
| ML | Input |
| 4101 | 03 |
| 4104 | 02 |

|  |  |
| --- | --- |
| Output Address | Value |
| 4500 | 00 |
| 4501 | 06 |

**EDSIM51 PROGRAM-MULTIPLICATION**

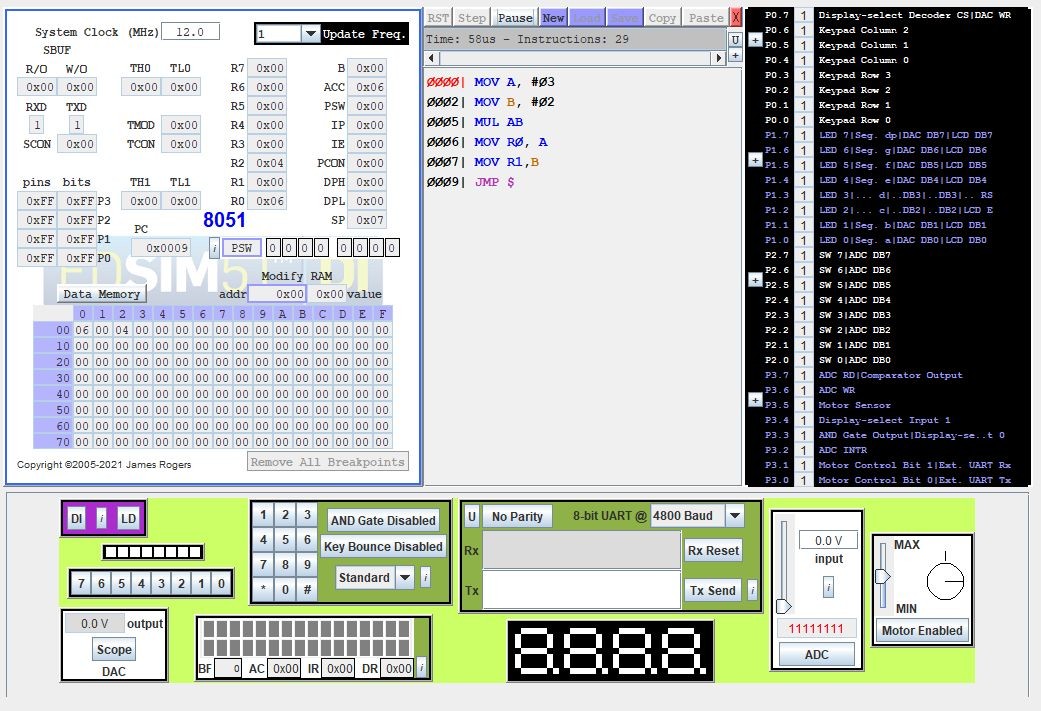
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ADDRESS** | **LABEL** | **MNEMONICS** | **OPCODE** | **COMMENTS** |
| 0000 | **START** | MOV A, #Data | 74 03 | **Move a data (8 bit) to Acc** |
| 0002 | **START** | MOV B, #Data | 75 F0 02 | **Move a data (8 bit) to B Reg** |
| 0005 | **MULTI PLY** | MUL AB | A4 | **Multiply them, and store result in A (and B- in case of 16 bit result)** |
| 0007 | **STORE** | MOV R0, A | 90 45 00 | **Copy Lower half in A Reg** |
| 0008 | **STORE** | MOV R1,B | F0 | **Copy Higher half in B Reg** |
| 0009 | **RUN** | JMP $ | A3 | **Jump to Address location** |
|  |  |  | E5 F0 |  |

|  |  |
| --- | --- |
| **OUT PUT ADDRESS** | **DATA** |
| 0000 | **03** |
| 0002 | **02** |

|  |  |
| --- | --- |
| **IN PUT ADDRESS** | **DATA** |
| R1 | **06** |
| psw | **00** |

**SIMULATION:**

**DIVISION**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Memory**  **Location** | **Label** | **Opcode** | **Mnemonics** | **Comments** |
| 0000 | **START** | 74 04 | MOV A,#04 | Move immediate data to accumulator |
| 0002 | **START** | 75 F0 02 | MOV B,#02 | Move immediate to B reg. |
| 0005 | **MULTI PLY** | 84 | DIV AB | Divide content of A &  B |
| 0007 | **STORE** | 90 45 00 | MOV DPTR, # 4500 | Load data pointer with 4500 location |
| 0008 | **STORE** | F0 | MOVX @DPTR,A | Move A to ext RAM |
| 0009 |  | A3 | INC DPTR | Increment data pointer |
|  |  | E5 F0 | MOV A,B | Move remainder to A |
|  |  | F0 | MOVX @DPTR,A | Move A to ext RAM |
|  |  | 80 FE | SJMP 410E | Remain idle in infinite loop |

**EDSIM 51: DIVISION**

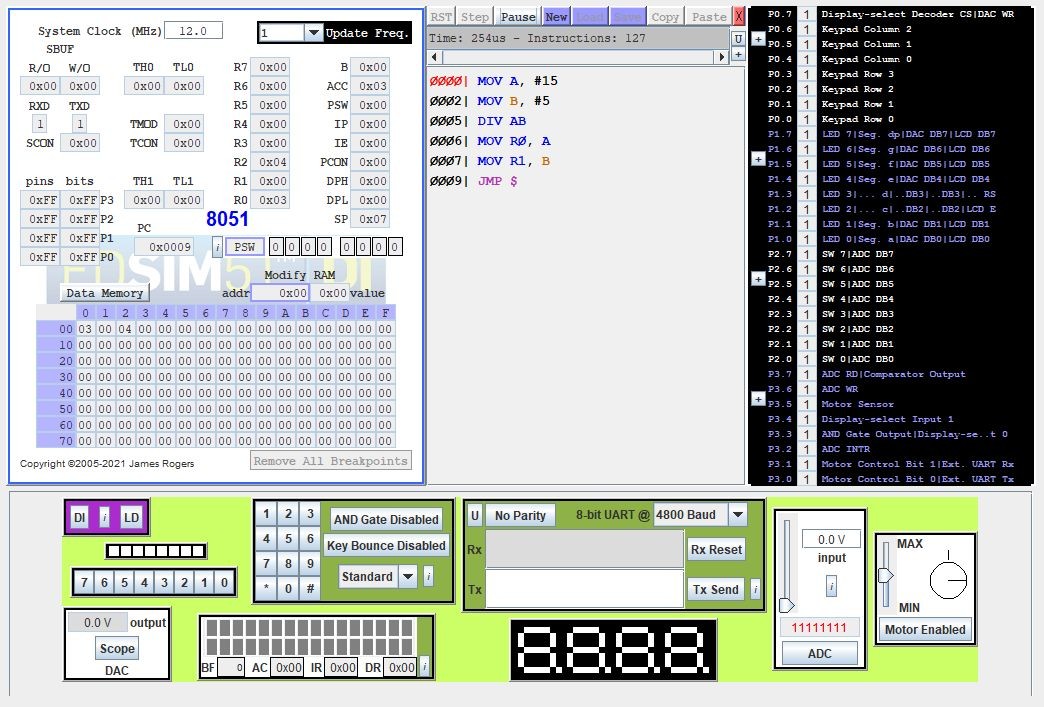
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ADDRESS** | **LABEL** | **MNEMONICS** | **OPCODE** | **COMMENTS** |
| 0000 | **START** | MOV A, #Data | 74 04 | **Move a data (8 bit) to Acc** |
| 0002 | **START** | MOV B, #Data | 75 F0 02 | **Move a data (8 bit) to B Reg** |
| 0005 | **MULTI PLY** | DIV AB | 84 | **Divide them, and store result in A (and B- in case of 16 bit result)** |
| 0007 | **STORE** | MOV R0, A | 90 45 00 | **Copy Lower half in A Reg** |
| 0008 | **STORE** | MOV R1, B | F0 | **Copy Higher half in B Reg** |
| 0009 | **RUN** | JMP $ | A3 | **Jump to Address location** |

|  |  |
| --- | --- |
| **IN PUT ADDRESS** | **DATA** |
| **0000** | **15** |
| **0002** | **5** |

|  |  |
| --- | --- |
| **OUT PUT ADDRESS** | **DATA** |
| **R0** | **5** |
| **r1** | **0** |

**SIMULATION:**

**Result:**



Thus 8-bit addition, subtraction, multiplication and division is performed using 8051.

**POST-LAB**

1. Define OPCODE and Operand, and specify the opcode and the operand in the instruction MOV H, L.

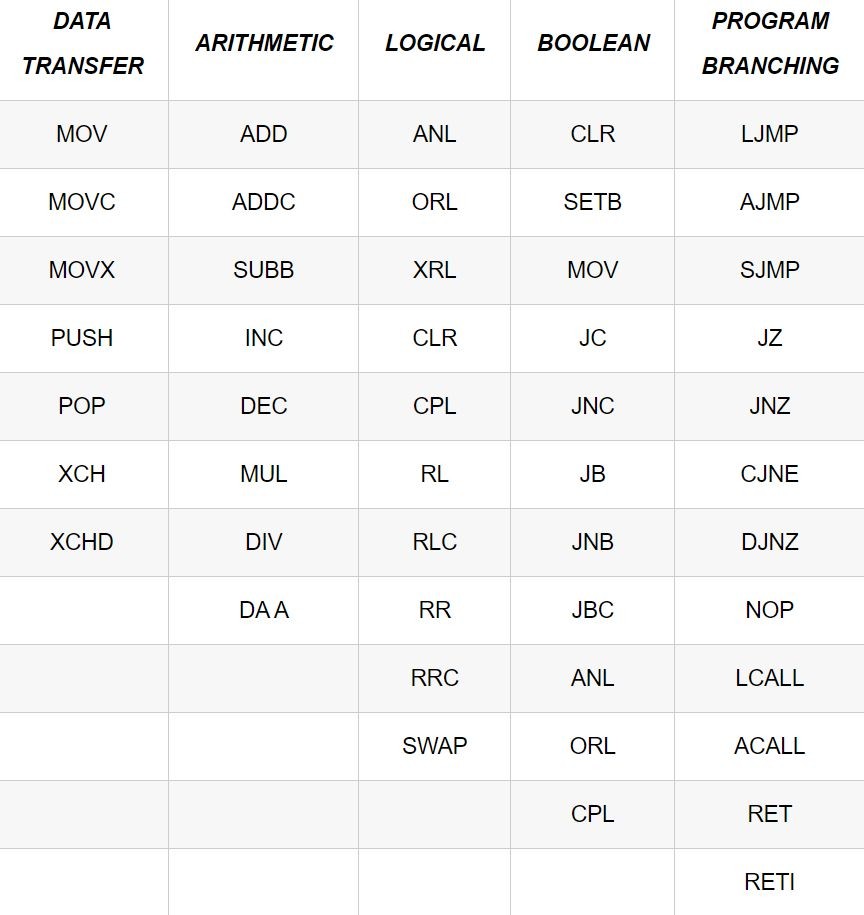
Each assembly language statement is split into an opcode and an operand

. The opcode is the instruction that is executed by the CPU and the operand is the data or memory location used to execute that instruction.

1. Find the machine codes and the number of bytes of for the following instructions. Identify the opcode and the operands.
   1. MVI H,47H - 10 bytes
   2. ADI F5H - 8 bytes
   3. SUB C - 6 bytes
2. Find the HEX codes for the following instructions, identify the opcodes and operands, and show the order of entering the codes in memory

STA 2050H - 32, 3 bytes JNZ 2070H - C2 , 3 bytes

1. Classification of 8051Instruction set.



1. Find the hex machine code for the following instruction from the instruction set and identify the number of bytes of each instruction and assume that the starting address is 2000H.

MVI B,45H - 2000: Opcode 2001: 45 MVI C, 78H - 77

MOV A,C - 79 OPCODE , 1 BYTE

ADD B - 55 opcode , 2 byte