# MICROPROCESSOR AND MICROCONTROLLER - LAB

## LIST OF CONTENTS IN SYLLABUS

## CONTENTS

**I Addition and Subtraction**

1. 8 - bit addition
2. 16 - bit addition
3. 8 - bit subtraction
4. BCD subtraction

**II Multiplication and Division**

1. 8 - bit multiplication
2. BCD multiplication
3. 8 - bit division

**III Sorting and Searching**

1. Searching for an element in an array.
2. Sorting in Ascending and Descending order.
3. Finding the largest and smallest elements in an array.
4. Reversing array elements.
5. Block move.

**IV Code Conversion**

1. BCD to Hex and Hex to BCD
2. Binary to ASCII and ASCII to binary
3. ASCII to BCD and BCD to ASCII

**V Simple programs on 8051 Microcontroller**

1. Addition
2. Subtraction
3. Multiplication
4. Division

**5. Interfacing Experiments using 8051**

A) Realisation of Boolean Expression through ports.

B) Time delay generation using subroutines.

C) Display LEDs through ports

# 8055-MICROPROCESSOR

**I - ADDITION AND SUBTRACTION**

**EX.NO:1A** **8 - BIT ADDITION**

**AIM:**

To write an 8085 assembly language program for 8-bit addition

**ALGORITHM:**

1) Start the program by loading the first data into Accumulator.
2) Move the data to a register (B register).
3) Get the second data and load it into Accumulator.
4) Add the two register contents.
5) Check for " carry ".
6) Store the value of the sum and " carry " in the memory location.
7) Terminate the program.

**PROGRAM:**

| Address | Labels | HEX Codes | Mnemonics | Comments |
|---|---|---|---|---|
| 8000 | | 0E, 00 | MVI C,00H | Clear C register |
| 8002 | | 21, 50, 80 | LXI H,8050H | Load initial address to get operand |
| 8005 | | 7E | MOV A,M | Load Acc with memory element |
| 8006 | | 23 | INX H | Point to next location |
| 8007 | | 46 | MOV B,M | Load B with second operand |
| 8008 | | 80 | ADD B | Add B with A |
| 8009 | | D2, 0D, 80 | JNC **STORE** | When CY = 0, go to STORE |
| 800C | | 0C | INR C | Increase C by 1 |
| 800D | **STORE** | 32, 52, 80 | STA 8052H | Store the sum value |
| 8010 | | 79 | MOV A, C | Move carry to accumulator |
| 8011 | | 32, 53, 80 | STA 8053H | Store the carry |
| 8014 | | 76 | HLT | Terminate the program |

**INPUT:**

| Address | Data | |
|---|---|---|
| 8050 | FF | Data 1 |

| | | |
|---|---|---|
| 8051 | FF | Data 2 |

**OUTPUT:**

| Address | Data | |
|---|---|---|
| 8052 | FE | SUM |
| 8053 | 01 | CARRY |

**RESULT:**

Thus the above program executed successfully.

**EX.NO:1B** **16 - BIT ADDITION**

**AIM:**

To write an 8085 assembly language program for 16-bit addition

**ALGORITHM:**

1. Load both the lower bit and higher bit of first number at once.
2. Copy the content HL pair to DE pair register.
3. Load the lower and higher bit of second number in HL pair register.
4. ADD both the register pair content using DAD operation.
6. Store the value of the sum and " carry " in the memory location.
7. Terminate the program.

**PROGRAM:**

| Address | Labels | HEX Codes | Mnemonics | Comments |
|---|---|---|---|---|
| 8000 | | 06, 00 | MVI B,00H | Save the carry |
| 8002 | | 2A, 50, 80 | LHLD 8050H | Get first 16 bit number in DE |
| 8005 | | EB | XCHG | Save first 16 bit number in DE |
| 8006 | | 2A, 52, 80 | LHLD 8052H | Get second 16 bit number in HL |
| 8009 | | 19 | DAD D | Add DE and HL |
| 800A | | D2, 0E, 80 | JNC STORE | Jump no Carry to store sum |
| 800D | | 04 | INR B | Increment B |
| 800E | **STORE** | 22, 54, 80 | SHLD 8054H | Store the sum value |
| 8011 | | 78 | MOV A, B | Move carry to accumulator |
| 8012 | | 32, 56, 80 | STA 8056H | Store the carry value |
| 8015 | | 76 | HLT | Terminate the program |

**INPUT:**

| Address | Data | |
|---|---|---|
| 8050 | FF | Data 1-LSB |
| 8051 | FF | Data 1-MSB |
| 8052 | FF | Data 2-LSB |
| 8053 | FF | Data 2-MSB |

**OUTPUT:**

| Address | Data | |
|---|---|---|
| 8054 | FE | SUM-LSB |
| 8055 | FF | SUM-MSB |
| 8056 | 01 | CARRY |

**RESULT:**

Thus the above program executed successfully.

**EX.NO:1C** **8 - BIT SUBTRACTION**

**AIM:**

To write an 8085 assembly language program for two 8-bit Subtraction

**ALGORITHM:**

1. Load 00 in a register C (for borrow)
2. Load two 8-bit number from memory into registers
3. Move one number to accumulator
4. Subtract the second number with accumulator
5. If borrow is not equal to 1, go to step 7
6. Increment register for borrow by 1
7. Store accumulator content in memory
8. Move content of register into accumulator
9. Store content of accumulator in other memory location
10. Stop

**PROGRAM:**

| Address | Label | Hex Codes | Mnemonics | Comments |
|---|---|---|---|---|
| 8000 | | 0E, 00 | MVI C, 00H | Clear C register |
| 8002 | | 21, 50, 80 | LXI H, 8050H | Load initial address to get operand |
| 8005 | | 7E | MOV A, M | Load Acc with memory element |
| 8006 | | 23 | INX H | Point to next location |
| 8007 | | 46 | MOV B, M | Load B with second operand |
| 8008 | | 90 | SUB B | SUB B with A |
| 8009 | | D2, 0F, 80 | JNC **LOOP** | When CY = 0, go to STORE |
| 800C | | 2F | CMA | Complement A |

| | | | | |
|---|---|---|---|---|
| 800D | | 3C | INR A | Increment Accumulator |
| 800E | | 0C | INR C | Increase C by 1 |
| 800F | **LOOP** | 32, 52, 80 | STA 8052H | Store the sum value |
| 8012 | | 79 | MOV A, C | Move carry to accumulator |
| 8013 | | 32, 53, 80 | STA 8053H | Store the carry |
| 8016 | | 76 | HLT | Terminate the program |

**INPUT:**

| Address | Data | |
|---|---|---|
| 8050 | 07H | Data 1 |
| 8051 | 0AH | Data 2 |

**INPUT:**

| Address | Data | |
|---|---|---|
| 8050 | 0AH | Data 1 |
| 8051 | 07H | Data 2 |

**OUTPUT: NEGATIVE VALUE**

| Address | Data | |
|---|---|---|
| 8052 | 03H | DIFFERENCE |
| 8053 | 01H | BORROW |

**OUTPUT: POSITIVE VALUE**

| Address | Data | |
|---|---|---|
| 8052 | 03H | DIFFERENCE |
| 8053 | 00H | BORROW |

**RESULT:**

Thus the above program executed successfully.

**EX.NO:1D** **BCD SUBTRACTION**

**AIM:**

To write an 8085 assembly language program to subtract two BCD (Binary Coded Decimal) numbers and store the result in memory

**ALGORITHM:**

1. Load the first BCD number from memory into the accumulator.
2. Copy the accumulator to register B
3. Load the second BCD number from memory into the accumulator.
4. Subtract the second BCD number from the B register, storing the result in the accumulator.
5. Perform a Decimal Adjust after Subtraction (DAA) to correct any borrow that may have occurred during subtraction.
6. Store the result of subtraction in memory at a specified address.
7. Halt the program.

**PROGRAM:**

| Address | Label | Hex Codes | Mnemonics | Comments |
|---|---|---|---|---|
| 8000 | | 3A, 51, 80 | LDA 8051H | Load the accumulator with 2nd BCD number in 8051H |
| 8003 | | 4F | MOV C, A | Copy the first BCD number to register C |

| | | | | |
|---|---|---|---|---|
| 8004 | | 3E, 99 | MVI A, 99H | Move Immediate 99H to accumulator |
| 8006 | | 91 | SUB C | Subtract with accumulator |
| 8007 | | 3C | INR A | Increment accumulator |
| 8008 | | 47 | MOV B,A | Move accumulator to register B |
| 8009 | | 3A, 50, 80 | LDA 8050H | Load the accumulator with 1st BCD number in 8050H |
| 800C | | 80 | ADD B | Add with register B |
| 800D | | 27 | DAA | Decimal adjust after subtraction to correct any borrow |
| 800E | | 32, 52, 80 | STA 8052H | Store the result at 9000H |
| 8011 | | 76 | HLT | Terminate the Program |

**INPUT:**

| **Address** | **Data** | |
|---|---|---|
| 8050 | 72H | Data 1 |
| 8051 | 35H | Data 2 |

**OUTPUT:**

| **Address** | **Data** |
|---|---|
| 8052 | 37H |

**RESULT:**

Thus the above program executed successfully.

## II - MULTIPLICATION AND DIVISION

**EX.NO:2A** **8 - BIT MULTIPLICATION**

**AIM:**

To write an 8085 assembly language program for two 8-bit Multiplication

**ALGORITHM:**

1. Load 00 in a register D (for carry)
2. Load two 8-bit number from memory into registers
3. Load B with second operand
4. ADD B with A
5. When CY = 0, go to step 7
6. Increment D register
7. Decrement C register
8. When CY = 1, go to step 4
9. Store the Product value
10. Move carry to accumulator
11. Stop

**PROGRAM:**

| Address | Label | Hex Codes | Mnemonics | Comments |
|---|---|---|---|---|
| 8000 | | 16, 00H | MVI D, 00H | Clear C register |
| 8002 | | AF | XRA A | And or with register A |
| 8003 | | 21, 50, 80 | LXI H, 8050H | Load initial address to get operand |
| 8006 | | 46 | MOV B, M | Load Acc with memory element |
| 8007 | | 23 | INX H | Point to next location |
| 8008 | | 4E | MOV C, M | Load B with second operand |
| 8009 | **STORE** | 80 | ADD B | ADD B with A |
| 800A | | D2, 0E, 80 | JNC **LOOP** | When CY = 0, go to LOOP |
| 800D | | 14 | INR D | Increment D register |
| 800E | **LOOP** | 0D | DCR C | Decrement C register |
| 800F | | C2, 09, 80 | JNZ **STORE** | When CY = 1, go to STORE |
| 8012 | | 32, 52, 80 | STA 8052H | Store the Product value |
| 8015 | | 7A | MOV A, D | Move carry to accumulator |
| 8016 | | 32, 53, 80 | STA 8053H | Store the carry |
| 8019 | | 76 | HLT | Terminate the program |

**INPUT:**

| Address | Data | |
|---|---|---|
| 8050 | 05H | MULTIPLICANT |
| 8051 | 06H | MULTIPLIER |

**OUTPUT:**

| Address | Data | |
|---|---|---|
| 8052 | 1EH | PRODUCT |
| 8053 | 00H | CARRY |

**RESULT:**

Thus the above program executed successfully.

**EX.NO:2B** **BCD MULTIPLICATION**

**AIM:**

To write an 8085 assembly language program for BCD Multiplication

**ALGORITHM:**

1. Initialize the data segment.
2. Get the first unpacked BCD number.
3. Get the second unpacked BCD number.
4. Multiply the two numbers.
5. Adjust result to valid unpacked BCD number in AX.
6. Display the result.
7. Stop

**PROGRAM:**

| Address | Label | Hex Codes | Mnemonics | Comments |
|---|---|---|---|---|
| 8000 | | 2A, 50, 80 | LHLD 8050H | Load HL register |
| 8003 | | 0E, 00 | MVI C, 00H | Move Immediate C with00H |
| 8005 | | 7C | MOV A, H | Move the first operand to Accumulator |
| 8006 | | 85 | ADD L | Add with L register |
| 8007 | | 27 | DAA | Move the pointer with Decimal Adjust Accumulator |
| 8008 | | D2, 0C, 80 | JNC **LOOP** | Jump no carry to store the result |
| 800B | | 0C | INR C | Increment register C |
| 800C | **LOOP** | 32, 50, 90 | STA 9050H | Store the result |
| 800F | | 79 | MOV A,C | Move carry to Accumulator |
| 8010 | | 32, 51, 90 | STA 9051H | store the carry |
| 8013 | | 76 | HLT | Terminate the program |

**INPUT:**

| Address | Data | |
|---|---|---|
| 8050 | 32H | MULTIPLICANT |
| 8051 | 77H | MULTIPLIER |

**OUTPUT:**

| Address | Data | |
|---|---|---|
| 8052 | 09H | PRODUCT |
| 8053 | 01H | CARRY |

**RESULT:**

Thus the above program executed successfully.

**EX.NO:2C** **8 - BIT DIVISION**

**AIM:**

To write an 8085 assembly language program for two 8-bit Division

**ALGORITHM:**

1. Start the program by loading the HL pair registers with address of memory location.
2. Move the data to B Register.
3. Load the second data into accumulator.
4. Compare the two numbers to check carry.
5. Subtract two numbers.
6. Increment the value of carry.
7. Check whether the repeated subtraction is over.
8. Then store the results(quotient and remainder) in given memory location.
9. Terminate the program.

**PROGRAM:**

| Address | Label | Hex Codes | Mnemonics | Comments |
|---|---|---|---|---|
| 8000 | | 06, 00H | MVI B, 00H | To store the quotient value |
| 8002 | | 21, 50, 80 | LXI H, 8050H | Load initial address to get operand |
| 8005 | | 7E | MOV A, M | Load Acc with memory element |
| 8006 | | 23 | INX H | Point to next location |
| 8007 | | 4E | MOV C, M | Move content M with C |
| 8008 | **STORE** | B9 | CMP C | Compare C register |
| 8009 | | DA, 11, 80 | JC **LOOP** | When CY = 0, go to LOOP |
| 800C | | 91 | SUB C | Subtact with C |
| 800D | | 04 | INR B | Increment B register |
| 800E | | C3, 08, 80 | JMP **STORE** | When CY = 1, go to STORE |

| | | | | |
|---|---|---|---|---|
| 8011 | **LOOP** | 32, 53, 80 | STA 8053H | Store the remainder value |
| 8014 | | 78 | MOV A, B | Move result to accumulator |
| 8015 | | 32, 52, 80 | STA 8052H | Store the result |
| 8018 | | 76 | HLT | Terminate the program |

**INPUT:**

| **Address** | **Data** | |
|---|---|---|
| 8050 | 0FH | DIVIDENT |
| 8051 | 02H | DIVISOR |

**OUTPUT:**

| **Address** | **Data** | |
|---|---|---|
| 8052 | 07H | QUOTIENT |
| 8053 | 01H | REMAINDER |

**RESULT:**

Thus the above program executed successfully.

## III - SORTING AND SEARCHING

**EX.NO:3A**

## SEARCHING AN ELEMENT IN AN ARRAY

**AIM:**

To write an 8085 ALP program to searching an element in an array

**ALGORITHM:**

1. Make the memory pointer points to memory location 8050 by help of LXI H 8050 instruction
2. Store value of array size in register C
3. Store number to be search in register B
4. Increment memory pointer by 1 so that it points to next array index
5. Store element of array in accumulator A and compare it with value of B
6. If both are same i.e. if ZF = 1 then store F0 in A and store the result in memory location 9000 and go to step 9
7. Otherwise store 00 in memory location 9000
8. Decrement C by 01 and check if C is not equal to zero i.e. ZF = 0, if true go to step 3 otherwise go to step 9
9. End of program

**PROGRAM:**

| Address | Label | Hex Codes | Mnemonics | Comment |
|---|---|---|---|---|
| 8000 | | 21,50,80 | LXI H, 8050H | Pointer points to memory location 8050 |
| 8003 | | 46 | MOV B,M | Store value of array size |
| 8004 | | 23 | INX H | Increment memory pointer |
| 8005 | | 4E | MOV C,M | Move the value to C |
| 8006 | | 23 | INX H | Increment memory pointer |
| 8007 | L2 | 7E | MOV A,M | Move the value to Accumulator |
| 8008 | | B9 | CMP C | Compare tha value of C |
| 8009 | | CA,1A,80 | JZ L1 | if c=0 then jump to desired value |
| 800C | | 23 | INX H | Increment memory pointer |
| 800D | | 05 | DCR B | Decrement memory B register |
| 800E | | C2,07,80 | JNZ L2 | if c not equal to zero then jump to next value |
| 8011 | | 21,00,00 | LXI H, 0000H | point to null value |
| 8014 | | 22,00,90 | SHLD 9000H | store null value |
| 8017 | | C3,1D,80 | JMP L3 | go to stop |
| 801A | L1 | 22,00,90 | SHLD 9000H | store value of searched element |
| 801D | L3 | 76 | HLT | stop |

**INPUT:**

| Address | Data | |
|---|---|---|
| 8050 | 06 | <-Array size |
| 8051 | 08 | <-Searching Element |
| 8052 | 20 | |
| 8053 | 18 | |
| 8054 | 45 | |
| 8055 | 33 | |
| 8056 | 08 | |
| 8057 | 71 | |

**OUTPUT: IF VALUE FOUND**

| Address | Data |
|---|---|
| 9000 | 56 |
| 9001 | 80 |

**OUTPUT: IF VALUE NOT FOUND**

| Address | Data |
|---|---|
| 9000 | 00 |
| 9001 | 00 |

**RESULT:**

Thus the above program executed successfully.

**EX.NO:3B**

**SORTING IN ASCENDING ORDER**

**AIM:**

To write an 8085 Assembly language program to sort numbers in ascending order

**ALGORITHM :**

1) Move the data to a register B
2) Store value in register C to B
3) Make the memory pointer points to memory location to store the data with the help of HL register pair
4) Store value of array size in register A, M
5) Increment memory pointer by 1 so that it points to next array index
6) Move to the next element of array in D register and compare it with value of D
7) If carry=1
8) Move the next element in accumulator to memory and decrement the H register
9) Move next element from D register to M
10) Increment HL register and decrement B register
11) If both are same i.e. if ZF = 1 then store F0 in A and store the result in memory location 8050
12) Decrement C by 01 and check if C is not equal to zero i.e. ZF = 0, if true go to step 3 otherwise go to step 3
13) End of program

**PROGRAM:**

| Address | Labels | HEX Codes | Mnemonics | Comments |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 8000 | | 06,04 | MVI B,04H | Move immediate B<-04H |
| 8002 | L3 | 48 | MOV C,B | Move C<-B |
| 8003 | | 21,50,80 | LXI H,8050 | Load HL register pair |
| 8006 | L2 | 7E | MOV A,M | A<-M |
| 8007 | | 23 | INX H | Increment to next pointer |
| 8008 | | 56 | MOV D,M | Move the Data M<-D |
| 8009 | | BA | CMP D | Compare D with next element |
| 800A | | DA,11,80 | JC L1 | If Jump carry=1, then point to next location |
| 800D | | 77 | MOV M,A | Move Acc to M register |
| 800E | | 2B | DCX H | Decrement H |
| 800F | | 72 | MOV M,D | Move D<-M |
| 8010 | | 23 | INX H | Increment to next pointer |
| 8011 | L1 | 0D | DCR C | Decrement C |
| 8012 | | C2,06,80 | JNZ L2 | If Jump Non Zero=1, then decrement B else Move to 8006 memory location |
| 8015 | | 5 | DCR B | Decrement B |
| 8016 | | C2,02,80 | JNZ L3 | If Jump Non Zero=1, then stop else Move to 8002 memory location |
| 8019 | | 76 | HLT | Terminate the program |

**INPUT:**

| Address | Data |
|---|---|
| 8050 | 15 |
| 8051 | 45 |
| 8052 | 35 |
| 8053 | 25 |
| 8054 | 65 |

**OUTPUT:**

| Address | Data |
|---|---|
| 8050 | 15 |
| 8051 | 25 |
| 8052 | 35 |
| 8053 | 45 |
| 8054 | 65 |

**RESULT:**

Thus the above program executed successfully.

**EX.NO:3C**

## SORTING IN DESCENDING ORDER

**AIM:**

To write an 8085 Assembly language program to sort numbers in descending order

**ALGORITHM:**

1) Move the data to a register B
2) Store value in register C to B
3) Make the memory pointer points to memory location to store the data with the help of HL register pair
4) Store value of array size in register A, M
5) Increment memory pointer by 1 so that it points to next array index
6) Move to the next element of array in D register and compare it with value of D
7) If carry=0
8) Move the next element in accumulator to memory and decrement the H register
9) Move next element from D register to M
10) Increment HL register and decrement B register
11) If both are same i.e. if ZF = 1 then store F0 in A and store the result in memory location 8050
12) Decrement C by 01 and check if C is not equal to zero i.e. ZF = 0, if true go to step 3 otherwise go to step 3
13) End of program

**PROGRAM:**

| Address | Labels | HEX Codes | Mnemonics | Comments |
|---|---|---|---|---|
| 8000 | | 06,04 | MVI B,04H | Move immediate B<-04H |
| 8002 | L3 | 48 | MOV C,B | Move C<-B |
| 8003 | | 21,50,80 | LXI H,8050 | Load HL register pair |
| 8006 | L2 | 7E | MOV A,M | A<-M |
| 8007 | | 23 | INX H | Increment to next pointer |
| 8008 | | 56 | MOV D,M | Move the Data M<-D |
| 8009 | | BA | CMP D | Compare D with next element |
| 800A | | D2,11,80 | JNC L1 | If Jump carry=0, then point to next location |
| 800D | | 77 | MOV M,A | Move Acc to M register |
| 800E | | 2B | DCX H | Decrement H |
| 800F | | 72 | MOV M,D | Move D<-M |
| 8010 | | 23 | INX H | Increment to next pointer |
| 8011 | L1 | 0D | DCR C | Decrement C |

| | | | | |
|---|---|---|---|---|
| 8012 | | C2,06,80 | JNZ L2 | If Jump Non Zero=1, then decrement B else Move to 8006 memory location |
| 8015 | | 05, | DCR B | Decrement B |
| 8016 | | C2,02,80 | JNZ L3 | If Jump Non Zero=1, then stop else Move to 8002 memory location |
| 8019 | | 76 | HLT | Terminate the program |

**INPUT:**

| Address | Data |
|---|---|
| 8050 | 15 |
| 8051 | 45 |
| 8052 | 35 |
| 8053 | 25 |
| 8054 | 65 |

**OUTPUT:**

| Address | Data |
|---|---|
| 8050 | 65 |
| 8051 | 45 |
| 8052 | 35 |
| 8053 | 25 |
| 8054 | 15 |

**RESULT:**

Thus the above program executed successfully.

**EX.NO:3D**

**FINDING THE LARGEST ELEMENT IN AN ARRAY**

**AIM:**

To write an 8085 Assembly language program to find the largest element in an array

**ALGORITHM:**

1) Load the address of the first element of the array in HL pair
2) Move the count to B – reg.
3) Increment the pointer.
4) Get the first data in A – reg.
5) Decrement the count.
6) Increment the pointer.
7) Compare the content of memory addressed by HL pair with that of A – reg.
8) If Carry = 0, go to step 10 or if Carry = 1 go to step 9.
9) Move the content of memory addressed by HL to A – reg.

10) Decrement the count
11) Check for Zero of the count. If ZF = 0, go to step 6, or if ZF = 1 go to next step.
12) Store the largest data in memory.
13) Terminate the program.

**PROGRAM:**

| Address | Labels | HEX Codes | Mnemonics | Comments |
|---|---|---|---|---|
| 8000 | | 21, 40, 80 | LXI H,8040H | Point to get an array size |
| 8003 | | 4E | MOV C, M | Get the size of array |
| 8004 | | 23 | INX H | Point to actual array |
| 8005 | | 46 | MOV B, M | Load the first number into B |
| 8006 | | 0D | DCR C | Decrease C |
| 8007 | LOOP | 23 | INX H | Point to next location |
| 8008 | | 7E | MOV A, M | Get the next number from memory to Acc |
| 8009 | | B8 | CMP B | Compare Acc and B |
| 800A | | DA, 0E, 80 | JC SKIP | if B > A,then skip |
| 800D | | 47 | MOV B, A | If CY is 0, update B |
| 800E | SKIP | 0D | DCR C | Decrease C |
| 800F | | C2, 07, 80 | JNZ LOOP | When count is not 0, go to LOOP |
| 8012 | | 21, 00, 90 | LXI H,9000H | Point to destination address |
| 8015 | | 70 | MOV M, B | Store the minimum number |
| 8016 | | 76 | HLT | Terminate the program |

**INPUT:**

| Address | Data |
|---|---|
| 8040 | 5 |
| 8041 | 55 |
| 8042 | 22 |
| 8043 | 44 |
| 8044 | 11 |
| 8045 | 66 |

| Address | Data |
|---|---|
| 9000 | 66 |

**RESULT:**

Thus the above program executed successfully.

**EX.NO:3E**

## FINDING THE SMALLEST ELEMENT IN AN ARRAY

**AIM:**

To write an 8085 Assembly language program to find the smallest element in an array

**ALGORITHM:**

1. Load the address of the first element of the array in HL pair
2. Move the count to B register.
3. Increment the pointer.
4. Get the first data in Accumulator.
5. Decrement the count.
6. Increment the pointer.
7. Compare the content of memory addressed by HL pair with that of Accumulator.
8. If carry = 1, go to step 10 or if Carry = 0 go to step 9.
9. Move the content of memory addressed by HL to Accumulator.
10. Decrement the count.
11. Check for Zero of the If ZF = 0, go to step 6, or if ZF = 1 go to next step.
12. Store the smallest data in memory.
13. Terminate the program.

**PROGRAM:**

| Address | Labels | HEX Codes | Mnemonics | Comments |
|---|---|---|---|---|
| 8000 | | 21, 40, 80 | LXI H,8040H | Point to get an array size |
| 8003 | | 4E | MOV C, M | Get the size of array |
| 8004 | | 23 | INX H | Point to actual array |
| 8005 | | 46 | MOV B, M | Load the first number into B |
| 8006 | | 0D | DCR C | Decrease C |
| 8007 | LOOP | 23 | INX H | Point to next location |
| 8008 | | 7E | MOV A, M | Get the next number from memory to Acc |
| 8009 | | B8 | CMP B | Compare Acc and B |
| 800A | | D2, 0E, 80 | JNC SKIP | if B > A,then skip |
| 800D | | 47 | MOV B, A | If CY is 0, update B |
| 800E | SKIP | 0D | DCR C | Decrease C |

| | | | | |
|---|---|---|---|---|
| 800F | | C2, 07, 80 | JNZ LOOP | When count is not 0, go to LOOP |
| 8012 | | 21, 00, 90 | LXI H,9000H | Point to destination address |
| 8015 | | 70 | MOV M, B | Store the minimum number |
| 8016 | | 76 | HLT | Terminate the program |

**INPUT:**

| Address | Data |
|---|---|
| 8040 | 5 |
| 8041 | 55 |
| 8042 | 22 |
| 8043 | 44 |
| 8044 | 11 |
| 8045 | 66 |

| Address | Data |
|---|---|
| 9000 | 11 |

**RESULT:**

Thus the above program executed successfully.

**EX.NO:3F**

**REVERSING ARRAY ELEMENTS**

**AIM:**

To write an 8085 assembly language program to reverse an array elements

**ALGORITHM:**

1. The DE register pair is set to point at destination address 9000H.
2. The HL pair is set to point to the last element of the block.
3. If the block size is 0A, then the last block will be at 800A
4. HL is pointing to 8050, and took the block size from there and store to C
5. Now add C with the L register to get the last block address.
6. Now take each element from memory pointed by HL
7. Store it back to the memory pointed by the DE.
8. Increase the DE
9. Decrease the HL
10. Entire block will be moved in reverse direction

**PROGRAM:**

| Address | Labels | HEX Codes | Mnemonics | Comments |
|---|---|---|---|---|
| 8000 | | 21, 50, 80 | LXI H,8050 | Point to 8050 to get block size |
| 8003 | | 4E | MOV C,M | Take the block size into C |
| 8004 | | 11, 00 90 | LXI D, 9000 | Point to the destination address |
| 8007 | | 7D | MOV A,L | Load L into A |
| 8008 | | 81 | ADD C | Add C to point to last address of block |
| 8009 | | 6F | MOV L,A | Store A to L again |
| 800A | LOOP | 7E | MOV A,M | Load memory to A |
| 800B | | 12 | STAX D | Store A into destination pointed by DE |
| 800C | | 13 | INX D | Point destination to next address |
| 800D | | 2B | DCX H | Point source to previous address |
| 800E | | 0D | DCR C | Decrease C by 1 |
| 800F | | C2, 0A, 80 | JNZ LOOP | if Z is not set jump to LOOP |
| 8012 | | 76 | HLT | Terminate the program |

**INPUT:**

| Address | Data |
|---|---|
| 8050 | 05 |
| 8051 | 11 |
| 8052 | 22 |
| 8053 | 33 |
| 8054 | 44 |
| 8055 | 55 |

**OUTPUT:**

| Address | Data |
|---|---|
| 9000 | 55 |
| 9001 | 44 |
| 9002 | 33 |
| 9003 | 22 |
| 9004 | 11 |

**RESULT:**

Thus the above program executed successfully.

**EX.NO:3G**

## MOVE BLOCK FROM SOURCE TO DESTINATION

**AIM:**

To write an 8085 assembly language program to move a block from source to destination

**ALGORITHM:**

1. Load register pair H-L with the address 8050H
2. Load register pair D-E with the address 8070H
3. Move the content at memory location into accumulator
4. Store the content of accumulator into memory pointed by D-E
5. Increment value of register pair H-L and D-E by 1
6. Decrements value of register C by 1
7. If zero flag not equal to 1, go to step 3
8. Stop

**PROGRAM:**

| Address | Label | Hex Codes | Mnemonic | Comment |
|---|---|---|---|---|
| 8000 | START: | 21,50,80 | LXI H, 8050H | Setup HL pair as a pointer for source memory. |
| 8003 | | 11,70,80 | LXI D, 8070H | Set up DE pair as a pointer for destination memory |
| 8006 | | 6,10 | MVI B, 10H | Set up B to count 16 bytes |
| 8008 | LOOP: | 7E | MOV A, M | Get data byte from source. |
| 8009 | | 12 | STAX D | Store data byte as destination |
| 800A | | 23 | INX H | Point HL to next source location |
| 800B | | 13 | INX D | Point DE to next destination |
| 800C | | 5 | DCR B | Decrement count |
| 800D | | C2,08,80 | JNZ LOOP | If counter is not 0, go back to transfer next byte. |

| | | | | |
|---|---|---|---|---|
| 8010 | | 76 | HLT | Stop |

**INPUT:**

| Address | Value |
|---|---|
| 8050H | 0 |
| 8051H | 11 |
| 8052H | 22 |
| 8053H | 33 |
| 8054H | 44 |
| 8055H | 55 |

**OUTPUT:**

| Address | Value |
|---|---|
| 8070H | 0 |
| 8071H | 11 |
| 8072H | 22 |
| 8073H | 33 |
| 8074H | 44 |
| 8075H | 55 |

**RESULT:**

Thus the above program executed successfully.

## IV - CODE CONVERSION

**EX.NO:4A**

### BCD TO HEX CONVERSION

**AIM:**

To write an 8085 assembly language program for BCD to HEXA DECIMAL conversion

**ALGORITHM:**

1. Initialize memory pointer to 8050
2. Get the most significant digit
3. Multiply the MSD by 10 using repeated addition
4. Add LSD to result obtained in above step
5. Store the converted result in memory 8052

**PROGRAM:**

| ADDRESS | LABEL | HEX CODE | MNEMONICS | COMMENT |
|---|---|---|---|---|
| 8000 | | 3A, 50,80 | LDA 8050 | Load Input in accumulator |
| 8003 | | 47 | MOV B, A | Move accumulator to B |
| 8004 | | E6, 0F | ANI 0FH | And immediate 0F |
| 8006 | | 4F | MOV C, A | Move accumulator to C |
| 8007 | | 78 | MOV A, B | Move B to accumulator |
| 8008 | | E6, F0 | ANI F0H | And immediate F0 |
| 800A | | 0F | RRC | Rotate Right Accumulator |
| 800B | | 0F | RRC | Rotate Right Accumulator |
| 800C | | 0F | RRC | Rotate Right Accumulator |
| 800D | | 0F | RRC | Rotate Right Accumulator |
| 800E | | 57 | MOV D, A | Move accumulator to D |
| 800F | | 1E, 0A | MVI E, 0AH | Move Immediate 0A to E |
| 8011 | | AF | XRA A | Exclusive OR Accumulator |
| 8012 | **LOOP** | 82 | ADD D | Add content D |
| 8013 | | 1D | DCR E | Decrement E |
| 8014 | | C2, 12, 80 | JNZ **LOOP** | Jump Non Zero |
| 8017 | | 81 | ADD C | Add the value of C |
| 8018 | | 32, 52, 80 | STA 8052 | Store Hex value in 8052 |
| 801B | | 76 | HLT | Stop |

**INPUT:**

| Address | Value |
|---|---|
| 8050H | 35 |

**OUTPUT:**

| Address | Value |
|---|---|
| 8052H | 23 |

**RESULT:**

Thus the above program executed successfully.

**EX.NO:4B**

## HEXA-DECIMAL TO BCD CONVERSION

**AIM:**

To write an 8085 assembly language program for HEXA DECIMAL to BCD conversion

**ALGORITHM:**

1. Initialize memory pointer to 8050 H
2. Get the Hexa decimal number in C - register
3. Perform repeated addition for C number of times
4. Adjust for BCD in each step
5. Store the BCD data in Memory

**PROGRAM:**

| Address | Labels | HEX Codes | Mnemonics | Comments |
|---|---|---|---|---|
| 8000 | | 21, 50, 80 | LXI H,8050H | Initialize memory pointer |
| 8003 | | 16, 00 | MVI D,00H | Clear D- reg for Most significant Byte |
| 8005 | | AF | XRA A | Clear Accumulator |
| 8006 | | 4E | MOV C, M | Get HEX data |
| 8007 | LOOP | C6, 01 | ADI 01H | Count the number one by one |
| 8009 | | 27 | DAA | Adjust for BCD count |
| 800A | | D2, 0E, 80 | JNC SKIP | Jump to SKIP |
| 800D | | 14 | INR D | Increase D |
| 800E | SKIP | 0D | DCR C | Decrease C register |
| 800F | | C2, 07, 80 | JNZ LOOP | Jump to LOOP |
| 8012 | | 6F | MOV L, A | Load the Least Significant Byte |
| 8013 | | 62 | MOV H, D | Load the Most Significant Byte |
| 8014 | | 22, 52, 80 | SHLD 8052H | Store the BCD |
| 8017 | | 76 | HLT | Terminate the program |

**INPUT:**

| Address | Value |
|---|---|
| 8050H | 23 |

**OUTPUT:**

| Address | Value |
|---|---|
| 8052H | 35 |

**RESULT:**

Thus the above program executed successfully.

**EX.NO:4C**

## BINARY TO ASCII CONVERSION

**AIM:**

To write an 8085 assembly language program for BINARY TO ASCII conversion

**ALGORITHM:**

1. Check if the digit is greater than or equal to 0A (in hexadecimal).
2. If it is, add 37 to the digit to convert it into ASCII value.
3. If it is not, add 30 to the digit to convert it into ASCII value.
4. Check the carry status.
5. If the carry status is 0, add 30H with the value to get ASCII value.

**PROGRAM:**

| Address | Labels | HEX Codes | Mnemonics | Comments |
|---|---|---|---|---|
| 8000 | | 3A, 50, 80 | LDA 8050H | Initialize memory pointer |
| 8003 | | FE, 0A | CPI 0AH | Compare data with 0AH |
| 8005 | | DA, 0A, 80 | JC LOOP | Jump if Carry |
| 8008 | | C6, 07 | ADI 07H | Add immediate 07H |
| 800A | LOOP | C6, 30 | ADI 30H | Add immediate 07H |
| 800C | | 32, 52, 80 | STA 8052H | Store the ASCII value |
| 800F | | 76 | HLT | Terminate the program |

**INPUT:1**

| Address | Value |
|---|---|
| 8050H | 0C |

**OUTPUT:1**

| Address | Value |
|---|---|
| 8052H | 43 |

**INPUT:2**

| Address | Value |
|---|---|
| 8050H | 08 |

**OUTPUT:2**

| Address | Value |
|---|---|
| 8052H | 38 |

**RESULT:**

Thus the above program executed successfully.

**EX.NO:4D**

## ASCII TO BINARY CONVERSION

**AIM:**

To write an 8085 assembly language program for ASCII TO BINARY conversion

**ALGORITHM:**

1. Input the content of 8050 in accumulator

2. Subtract 30H to accumulator
3. Compare with 0AH
4. Jump if Carry
5. Subtract offset to get Alphabetic character
6. Store content of Hex value to memory location 8052
7. Terminate the process

**PROGRAM:**

| Address | Labels | HEX Codes | Mnemonics | Comments |
|---|---|---|---|---|
| 8000 | | 3A, 50, 80 | LDA 8050H | Initialize memory pointer |
| 8003 | | D6, 30 | SUI 30H | Subtract 30to get numeric value |
| 8005 | | FE, 0A | CPI 0AH | Compare with ASCII |
| 8007 | | DA, 0C, 80 | JC LOOP | Jump if carry |
| 800A | | D6, 07 | SUI 07H | Subtract offset to get Alphabetic character |
| 800C | LOOP | 32, 52, 80 | STA 8052H | Store the Hex value |
| 800F | | 76 | HLT | Terminate the program |

**INPUT:1**

| Address | Value |
|---|---|
| 8050H | 43 |

**OUTPUT:1**

| Address | Value |
|---|---|
| 8052H | 0C |

**INPUT:2**

| Address | Value |
|---|---|
| 8050H | 38 |

**OUTPUT:2**

| Address | Value |
|---|---|
| 8052H | 08 |

**RESULT:**
Thus the above program executed successfully.

**EX.NO:4E**

**BCD TO ASCII CONVERSION**

AIM:

To write an 8085 assembly language program for BCD TO ASCII conversion

**ALGORITHM:**

1. Input the content of 8050 in accumulator
2. ADD 30 to accumulator
3. Store the ASCII value
4. Terminate the program

**PROGRAM:**

| Address | Labels | HEX Codes | Mnemonics | Comments |
|---|---|---|---|---|
| 8000 | | 3A, 50, 80 | LDA 8050H | Initialize memory pointer |
| 8003 | | C6, 30 | ADI 30H | Add 30H to get numeric value |
| 8005 | | 32, 52, 80 | STA 8052H | Store the ASCII value |
| 8007 | | 76 | HLT | Terminate the program |

**INPUT:**

| Address | Value |
|---|---|
| | 08 |

**OUTPUT:**

| Address | Value |
|---|---|
| 8052H | 38 |

**RESULT:**

Thus the above program executed successfully.

**EX.NO:4F**

**ASCII TO BCD CONVERSION**

AIM:

To write an 8085 assembly language program for ASCII TO BCD conversion

**ALGORITHM:**

1. Input the content of 8050 in accumulator
2. SUB 30 to accumulator
3. Store the BCD value
4. Terminate the program

**PROGRAM:**

| Address | Labels | HEX Codes | Mnemonics | Comments |
|---|---|---|---|---|
| 8000 | | 3A, 50, 80 | LDA 8050H | Initialize memory pointer |
| 8003 | | D6, 30 | SUI 30H | Sub 30H to get numeric value |
| 8005 | | 32, 52, 80 | STA 8052H | Store the BCD value |
| 8007 | | 76 | HLT | Terminate the program |

**INPUT:**

| Address | Value |
|---|---|
| 8050H | 39 |

**OUTPUT:**

| Address | Value |
|---|---|
| 8052H | 09 |

**RESULT:**

Thus the above program executed successfully.

# 8051-MICROPROCESSOR

**EX-5A** **SIMPLE PROGRAMS ON 8051 MICROCONTROLLER ADDITION**

**AIM:**

To develop a program that performs the addition of two 8-bit numbers using the 8051 microcontroller.

**ALGORITHM:**

1. Initialize the first number: Load the first number into register A.
2. Initialize the second number: Load the second number into another register (e.g., R0).
3. Perform Addition: Add the two numbers and store the result in register A.
4. Store/Display the result: Store the result in a memory location or output port.

**SOURCE CODE:**

| **MNEMONICS** | **COMMENTS** |
|---|---|
| ORG 0 H | Orgin directives to set the starting address as 0H |
| MOV A, #20H | Move the first data 20H in accumulator |
| MOV R0, #10H | Move the second data 10H in R0 register |
| ADD A, R0 | Add the values |
| END | stop |

**RESULT:**

Thus the above program executed successfully

**EX-5B** **SIMPLE PROGRAMS ON 8051 MICROCONTROLLER**
**SUBTRACTION**

**AIM:**

To develop a program that performs the subtraction of two 8-bit numbers using the 8051.

**ALGORITHM:**

1. Initialize the first number: Load the first number into register A.
2. Initialize the second number: Load the second number into another register (e.g., R0).
3. Perform Subtraction: Subtract the second number from the first number and store the result in register A.
4. Store/Display the result: Store the result in a memory location or output port.

**SOURCE CODE:**

| MNEMONICS | COMMENTS |
|---|---|
| ORG 0 | Orgin directives to set the starting address as 0H |
| MOV A,#20H | Move the first data 20H in accumulator |
| MOV R0,#10H | Move the second data 10H in R0 register |
| SUBB A,R0 | Subtract the values |
| END | stop |

**RESULT:**
Thus the above program executed successfully

**EX-5C** **SIMPLE PROGRAMS ON 8051 MICROCONTROLLER**
**MULTIPLICATION**

**AIM:**
To develop a program that performs the multiplication of two 8-bit numbers using the 8051 microcontroller

**Algorithm**

1. Initialize the first number: Load the first number into register A.
2. Initialize the second number: Load the second number into register B.
3. Perform Multiplication: Use the MUL AB instruction to multiply the numbers in registers A and B.
4. Store/Display the result: Store the result in designated registers or memory locations.

**SOURCE CODE:**

| MNEMONICS | COMMENTS |
|---|---|
| CLR C | Clear the content |
| MOV A,#04H | Move the first data 04H in accumulator |
| MOV B,#03H | Move the second data 03H in B register |
| MUL A B | Multiply the values |
| MOV R4,A | Move the result in R4 register |
| JMP $ | Jump and stop |

**RESULT:**
Thus the above program executed successfully

r.

**EX:5D** **SIMPLE PROGRAMS ON 8051 MICROCONTROLLER**
**DIVISION**

**AIM:**

To develop a program that performs integer division of two 8-bit numbers using the 8051 microcontroller.

**ALGORITHM:**

1. Initialize the dividend: Load the dividend into register A.
2. Initialize the divisor: Load the divisor into register B.
3. Perform Division:
o Subtract the divisor from the dividend until the dividend is less than the divisor.
o Count the number of subtractions to get the quotient.
o The remainder is left in register A after the division process.
4. Store/Display the result: Store the quotient and remainder in designated registers or memory location

**SOURCE CODE:**

| **MNEMONICS** | **COMMENTS** |
|---|---|
| CLR C | Clear the content |
| MOV A, #06H | Move the first data 06H in accumulator |
| MOV B, #03H | Move the second data 03H in B register |
| DIV A B | Divide the values |
| MOV R4,A | Move the result in R4 register |
| JMP $ | Jump and stop |

**RESULT:**
Thus the above program executed successfully

**EX:5E** REALISATION OF BOOLEAN EXPRESSION THROUGH PORTS

**AIM:**

To develop an experiment using the 8051 microcontroller to realize a Boolean expression through its ports.

**ALGORITHM**:

1. Initialize the ports: Configure the ports of the 8051 microcontroller for input and output.
2. Implement the Boolean expression: Write the code to evaluate a Boolean expression using the inputs from the ports.
3. Display the result: Output the result of the Boolean expression using the LEDs connected to the ports

**SOURCE CODE:**

| **MNEMONICS** | **COMMENTS** |
|---|---|
| ORG 0000H | Configure Port 0 (P0) for input (switches). |
| MOV P0, #0FFH | Configure P0.0-P0.7 as input. Configure Port 1 (P1) for output (LEDs). |
| MOV P1, #00H | Initialize P1.0-P1.7 as output ; Read inputs (switches) |
| MOV A, P0 | Read the state of Port 0 (switches). Extract variables from inputs. |
| MOV B, A | Store inputs in register B ; Implement Boolean expression: Y = A * B' + C ; A is connected to P0.0, B is connected to P0.1, and C is connected to P0.2 |
| MOV C, A | Move inputs to C. |
| ANL C, #01H | Bitwise AND operation and store the result in C register |
| ANL A, #02H | Bitwise AND operation and stop the execution |

**RESULT:**
Thus the above program executed successfully

**EX-5F SIMPLE PROGRAMS ON 8051 MICROCONTROLLER**

**Time delay generation using subroutines**

**AIM:**

To develop an experiment using the 8051 microcontroller to generate time delays using subroutines.

**ALGORITHM:**

1. Initialize the ports: Configure the ports of the 8051 microcontroller for output (LEDs).
2. Implement time delay subroutine: Write a subroutine to generate a specific time delay.
3. Call the subroutine: Call the subroutine from the main program to generate the desired time delay.
4. Display the time delay: Use LEDs to indicate the duration of the time delay.

**SOURCE CODE:**

| MNEMONICS | COMMENTS |
|---|---|
| ORG 0000H | Configure Port 1 (P1) for output (LEDs) |
| MOV P1, #00H | Initialize P1.0-P1.7 as output |
| MAIN: | Main program loop |
| CALL DELAY_1S | Generate a delay of 1 second (assuming a clock frequency of 11.0592 MHz) |
| SJMP MAIN | Loop indefinitely |
| DELAY_1S: | Subroutine to generate a delay of approximately 1 second |
| MOV R2, #250 | Move the values to R2 register |
| OUTER_LOOP: | Initialize outer loop counter |
| MOV R1, #250 | Move the values to R1 register |
| INNER_LOOP: | Initialize inner loop counter |
| DJNZ R1, INNER_LOOP | Decrement R1, jump if not zero |
| DJNZ R2, OUTER_LOOP | Decrement R2, jump if not zero |
| RET | Return from subroutine |

**RESULT:**

Thus the above program executed successfully

**EX-5G** **SIMPLE PROGRAMS ON 8051 MICROCONTROLLER**

## Display LEDs through ports

**AIM:**

To develop an experiment using the 8051 microcontroller to display LEDs through its ports.

**ALGORITHM:**

1. Initialize the ports: Configure the ports of the 8051 microcontroller for output (LEDs)
2. Toggle LEDs: Write a program to toggle LEDs connected to the ports.
3. Display the output: Observe the LEDs to verify they are toggling correctly.

**SOURCE CODE:**

| MNEMONICS | COMMENTS |
|---|---|
| ORG 00H | Configure Port 1 (P1) for output (LEDs) |
| BACK: MOV A,#0FEH | Move data to Accumulator |
| MOV P1,A | Move data from Accumulator to P1 |
| ACALL DELAY | Call function |
| MOV A,#0FFH | Move data to Accumulator |
| MOV P1,A | Move data from Accumulator to P1 |
| ACALL DELAY | Call function |
| SJMP BACK | Jump Label |
| DELAY: MOV R0,#0FFH | Move data to Register R0 |
| AGAIN: MOV R1,0FFH | Move data to Register R1 |
| HERE: DJNZ R1,HERE | Decrement the value R1 if JNZ=1 |
| DJNZ R0,AGAIN | Decrement the value R0 if JNZ=1 |
| RET | Return and stop the execution |

**RESULT:**

Thus the above program executed successfully