RAJALAKSHMI ENGINEERING COLLEGE

(An ISO 9001:2000 Certified Institution)

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

CS1306- MICROPROCESSORS AND MICROCONTROLLERS

MASTER RECORD

IV SEM IT

PREPARED BY:

R.TAMILSELVI

CS1306- MICROPROCESSORS AND MICROCONTROLLERS

CONTENTS:

1. Programming with 8085 – 8 bit /16 bit multiplication/division using repeated addition/subtraction

2. Programming with 8085- code conversion, decimal arithmetic, bit manipulations.

3. Programming with 8085- matrix multiplication, floating point operation

4. Programming with 8086- String manipulations, search find and replace, copy operation, sorting (PC required)

5. Using BIOS/DOS calls: Keyboard control, display, file manipulation. (PC required)

6. Using BIOS/DOS calls: Disk operation. (PC required)

7. Interfacing with 8085/8086 – 8255,8253

8. Interfacing with 8085/8086 – 8279, 8251

9. 8051 Microcontroller based experiments – simple assembly language programs.(cross assembler required)

10. 8051 Microcontroller based experiments – simple control application. (cross assembler required)

1. **Programming with 8085 – 8 bit /16 bit multiplication/division using repeated addition/subtraction**

Program No: 1(a)

a) 8 BIT MULTIPLICATION USING REPEATED ADDITION

AIM:

To write an assembly language program using 8085 processor to multiply two 8 bit data using repeated addition method.

ALGORITHM:

1. Get the multiplier and multiplicand from the memory
2. Initialize register pair=0000
3. Perform double addition as register pair = register pair + multiplicand
4. Decrement multiplier as multiplier = multiplier-1
5. Is multiplier=0, if yes go to step 6 otherwise go to step 3
6. Store the contents of the register pair in the memory location
7. Stop.

PROGRAM:

LABEL	ADDRESS	MNEMONICS	OPCODE			COMMENTS
	4100	LDA 4502	3A	02	45	Load multiplier and get the multiplier to B
	4103	MOV B, A	47			
	4104	LXI D, 0000 H	11	00	00	Load multiplicand in DE
	4107	LHLD 4500	2A	00	45	
	410A	XCHG	EB			
L1:	410B	DAD D	19			If not zero loop again. Else store the result
	410C	DCR B	05			
	410D	JNZ L1	C2	0B	41	
	4110	SHLD 5000	22	00	50	
	4113	HLT	76			

RESULT:

Thus an assembly language program using 8085 processor to multiply two 8 bit data using repeated addition method was written an executed.

BEFORE EXECUTION AFTER EXECUTION

ADDRESS	DATA	ADDRESS	DATA
4500	**93**	**4500**	**93**
4501	**00**	**4501**	**00**
4502	**23**	**4502**	**23**
4503	**00**	**4503**	**00**
		5000	**19**
		5002	**14**

Program No:1(b)

b) 16 BIT MULTIPLICATION USING REPEATED ADDITION

AIM:

To write an assembly language program using 8085 processor to multiply two 16 bit data using repeated addition method.

ALGORITHM:

1. Load the first data in the HL pair and move to SP
2. Load the second data in HL and move to DE (count)
3. Clear the HL pair (Initial sum)
4. Clear BC pair for overflow (carry)
5. Add the contents of SP to HL
6. Check for carry. If carry =1, go to step 7 or if carry = 0 , go to step 8.
7. Increment BC pair
8. Decrement the count
9. Check whether count has reached zero
10. To check for the count , move the content of E register to A register and logically OR with D register
11. Check the zero flag. If ZF = 0 , repeat steps 5 through 11 or if ZF = 1, go to next step.
12. Store the content of HL in memory. (Least significant 16 bits of the product)
13. Move the content of C to L and B to H and store HL in memory. (Most significant 16 bits of the product).
14. Stop.

PROGRAM:

LABEL	ADDRESS	MENMONICS	OPCODE			COMMENTS
	4100	LHLD 4500	2A	00	45	Load the HL register pair
	4103	SPHL	F9			Store the HL content in Stack pointer
	4104	LHLD 4502	2A	02	45	Load the HL register pair
	4107	LXI H, 0000 H	21	00	00	Load the HL register pair with 0000
	410A	LXI B, 0000 H	01	00	00	Load the DE register pair with 0000
L2	410D	DAD SP	39			Double add the stack pointer with HL and the result is stored in the HL register paper
	410E	JNC L1	D2	12	41	Jump on no carry to location L1
	4111	INX B	03			Increment the B register
L1	4112	DCX D	1B			Decrement DE register pair

	4113	MOV A, E	7B			Move the content form E register to A register
	4114	ORA D	B2			OR the content with D register
	4115	JNZ L2	C2	0D	41	Jump on no zero to location L2
	4118	SHLD 5002	22	02	50	Store the content
	411B	MOV L, C	69			Move the C content to L register
	411C	MOV H, B	60			Move the B register to H register
	411F	SHLD 5000	22	00	50	Store the content
	4122	HLT	76			Stop

RESULT:

Thus an assembly language program is written and executed to perform a 16 bit multiplication using repeated addition method.

BEFORE EXECUTION

ADDRESS	DATA
4500	**22**
4501	**22**
4502	**00**
4503	**22**

AFTER EXECUTION

ADDRESS	DATA
4500	**22**
4501	**22**
4502	**00**
4503	**22**
5000	**00**
5001	**04**
5002	**88**
5003	**84**

Program No:1(c)

c) 8 BIT DIVISION USING REPEATED SUBTRACTION

AIM:

To write an assembly language program using 8085 processor to divide two 8 bit data using repeated subtraction method.

ALGORITHM:

1. Load the divisor to register B and dividend to A reg
2. Initialize quotient = 0
3. Its dividend<divisor , if yes go to step 7 otherwise go to step 4
4. Dividend = dividend – divisor
5. Quotient = quotient+1
6. Is dividend < divisor, if yes go to step 7 otherwise go to step 4
7. Store the quotient and reminder(=dividend now)
8. Stop.

PROGRAM:

LABEL	ADDRESS	MNEMONICS	OPCODE			COMMENTS
	4100	LDA 4500	3A	00	15	Load the content in A register
	4103	MOV B, A	47			Move the A register to B register
	4104	LDA 4502	3A	02	45	Load the A register
	4107	MVI C, 00	0E	00		00 is loaded immediately in C register
L2	4109	CMP B	B8			Compare the two register content
	410A	JC L1	DA	12	41	Jump on carry to location L1
	410D	SUB B	90			Subtract the B register content
	410E	INR C	0C			Increment the C register
	410F	JMP L2	C3	09	41	Jump to location L2
L1	4112	STA 5000	32	00	50	Store the result
	4115	MOV A, C	79			Move the content to A register
	4116	STA 5002	32	02	50	Store the content in 5002
	4119	HLT	76			Stop

RESULT:

Thus an assemble language program is written and executed to perform a 8 bit division using repeated subtraction.

BEFORE EXECUTION

ADDRESS	DATA
4500	22
4502	02

AFTER EXECUTION

ADDRESS	DATA
4500	22
4502	02
5000	00
5002	11

Program No:1(d)

d) 16 BIT DIVISION USING REPEATED SUBTRACTION

AIM:

To write an assembly language program using 8085 processor to divide two 16 bit data using repeated subtraction method.

ALGORITHM:

1. Load the DE pair with 0000 for storing the quotient
2. Load the immediate multiplier and multiplicand in the register pair HL and BC register pair
3. Move the contents in the L register to the A register
4. Subtract the contents in the A register with C
5. Put the result back in the L register
6. Move the H register content to A register
7. Subtract the contents of A register with B
8. Put the result back in the H register
9. Check for carry , if carry = 1 go to step 12 other wise go to the next step
10. Increment the DE register pair
11. Jump to step 3
12. Perform double addition and store the reminder and quotient.
13. Stop

PROGRAM:

LABEL	ADDRESS	MNEMONICS	OPCODE			COMMENTS
	4100	LXI D, 0000	11	00	00	Clear the DE register pair
	4103	LXI H, 4444	23	44	44	Load the HL register pair with 4444
	4106	LXI B, 2222	01	22	22	Load the DC register pair with 2222
L2:	4109	MOV A, L	7D			Move the content from L to A
	410A	SUB C	91			Subtract the content from C register
	410B	MOV L, A	6F			Move the content from A to L
	410C	MOV A, H	7C			Move the content from H to A
	410D	SBB B	98			Subtract with borrow
	410E	MOV H, A	67			Move the results to H
	410F	JC L1	DA	16	41	Jump on carry to location L1

	4112	INX D	13			Incitement the DE register pair
	4113	JMP L2	C3	09	41	Jump to location L2
L1:	4116	DAD B	09			Double add the content in BC with Hl
	4117	SHLD 5000	22	00	50	Store the result
	4120	MOV H, D	62			Move the content from D to H
	4121	MOV L, E	6B			As well as E to L
	4122	SHLD 5002	22	02	50	Store the result
	4125	HLT	76			Stop

RESULT:

Thus an assembly language program is written and executed to perform a 16 bit division.

BEFORE EXECUTION

ADDRESS	DATA
4103	**4444**
4106	**2222**

AFTER EXECUTION

ADDRESS	DATA
4103	**4444**
4106	**2222**
5000	**0000**
5002	**0002**

2. Programming with 8085 - Code Conversion, Decimal Arithmetic, Bit Manipulations

Program No:2

I) CODE CONVERSION

a) ASCII TO DECIMAL

AIM:

To write an assembly language program to convert the ASCII number in memory to its equivalent decimal number

ALGORITHM:

1. Initialize the memory address
2. Move the content from the memory location the A register
3. Subtract the value 30 form the memory data
4. Compare the result with 0A
5. If carry = 1 the go to step 7
6. Otherwise move the data "FF" to accumulator
7. Increment the HL register pair
8. Move the content form the memory to the accumulator
9. Stop

LABEL	ADDRESS	MNEMONICS	OPCODE			COMMENTS
	4100	LXI H, 4500	21	00	45	Initialize the memory location
	4103	MOV A, M	7E			Move the Memory content to A register
	4104	SUI 30	DE	30		Subtract the content with 30
	4106	CPI 0A	FE	0A		Compare the result with 0A
	4108	JC L1	DA	0D	41	Jump on carry to location L1
	410B	MVI A, FF	3E	FF		Move the immediate content FF to A register
L1	410D	INX H	23			Increment the HL register pair
	410E	MOV M, A	77			Move the content form A reg to M
	410F	HLT	76			Stop

RESULT:

Thus the assembly language program for converting a ASCII number to decimal number is written and executed

BEFORE EXECUTION

ADDRESS	DATA
4500	39

AFTER EXECUTION

ADDRESS	DATA
4501	09

b) HEXADECIMAL TO DECIMAL

AIM:

To write an assembly language to convert the hex number in memory to its equivalent decimal number.

ALGORITHM:

1. Initialize the memory address
2. Load the BC register pair with 0000
3. Move the contents from memory to A register
4. Subtract the content with 64
5. Check for carry , If carry = 1 then go to step 8 other wise execute the next step
6. Increment the B register
7. Jump to step 4
8. Add the result with 64
9. Subtract the result with 0A
10. Check for carry , If carry = 1 then go to step 13 other wise execute the next step
11. Increment the C register
12. Jump to step 9
13. Add 0A with the content
14. Increment the Hl register pair
15. Move the content form the B register to memory
16. Move the content form the A register to B
17. Move the content form the C register to A
18. Rotate the obtained value 4 times to the left
19. Add the B register content with the accumulator
20. Increment the HL register pair
21. Move the accumulator content to memory
22. Stop

PROGRAM:

LABEL	ADDRESS	MNEMONICS	OPCODE			COMMENTS
	4100	LXI H, 4500	21	50	45	Load the HL register pair with the location 4500
	4103	LXI B, 0000	01	00	00	Clear the BC register pair
	4106	MOV A, M	7E			Move the content from memory to A register
L2	4107	SUI 64	D6	64		Subtract the content with 64
	4109	JC L1	DA	10	41	If there is carry go to location L1
	410C	INR B	04			Increment the B register
	410D	JMP L2	C3	07	41	Jump to location L2
L1	4110	ADI 64	C6	64		Add the value 64
L4	4112	SUI 0A	D6	0A		Subtract the value with 0A
	4114	JC L3	DA	1B	41	If there is carry go to location L3

	4117	INR C	0C			Increment the C register
	4118	JMP L4	C3	12	41	Jump to location L4
L3	411B	ADI 0A	C6	0A		Add the value 0A
	411D	INX H	23			Increment he HL register pair
	411E	MOV M, B	70			Move the content from B register to M
	411F	MOV B, A	47			Move the content from A register to B register
	4120	MOV A, C	79			Move the content from C register to A register
	4121	RLC	07			Rotate the content to left
	4122	RLC	07			Rotate the content to left
	4123	RLC	07			Rotate the content to left
	4124	RLC	07			Rotate the content to left
	4125	ADD B	80			Add the content with B register
	4126	INX H	23			Increment the HL register pair.
	4127	MOV M, A	77			Move the content to Memory location
	4128	HLT	76			Stop

RESULT:

Thus the assembly language program for converting a hexa number to decimal number is written and executed

BEFORE EXECUTION

ADDRESS	DATA
4500	**0A**

AFTER EXECUTION

ADDRESS	DATA
4502	**10**

c) HEXADECIMAL TO BINARY

AIM:

To write an assembly language program to convert an 8 bit hex number to its binary form and store in memory.

ALGORITHM:

1. Initialize the memory address
2. Load the value 08 to B register
3. Load the hexadecimal number to accumulator
4. Rotate the content towards right once
5. Check for carry, if carry = 1 got to step 8 otherwise execute the next step
6. Clear the memory location
7. Jump to step 9
8. Move the 01 to memory
9. Increment the memory location
10. Decrement the B register content
11. Jump on non zero to 4 other wise stop.

PROGRAM:

LABEL	ADDRESS	MNEMONICS	OPCODE			COMMENTS
	4100	LXI H, 4500	21	00	45	Initialize the memory location
	4103	MVI B, 08	06	08		Move the content 08 to B register
	4105	MVI A, 5A	3E	5A		Move the content 5A to A register
L3	4107	RRC	0F			Rotate the content through right
	4108	JC L1	DA	10	41	Jump on carry to location L1
	410B	MVI M, 00	36	00		Move the memory location 00
	410D	JMP L2	C3	12	41	Jump to Location L2
L1	4110	MVI M, 01	36	01		Move the content to 01 to Memory location
L2	4112	INX H	23			Increment the HL
	4113	DCR B	05			Decrement to B register
	4114	JNZ L3	C2	04	41	Jump on no zero to L3
	4117	HLT	76			Stop

RESULT:

Thus the assembly language program for converting a hexa number to binary number is written and executed

BEFORE EXECUTION

ADDRESS	DATA
4500	FF

AFTER EXECUTION

ADDRESS	DATA
4501	01
4502	01
4503	01
4504	01
4505	01
4506	01
4507	01
4508	01

II) DECIMAL ARIHMETIC OPERATION

a) ADD TWO 2 – DIGIT BCD DATA

AIM:

To write an assembly language program two add 2 digit BCD data

ALGORITHM:

1. Load the first data and move to B register
2. Load the second data and in accumulator
3. Clear C register for storing the carry
4. Add the contents of B register to accumulator
5. Execute DAA instruction
6. Check for carry, If carry = 1 , go to step 7 otherwise execute the next step
7. Increment the C register to account for carry
8. Store the sum and carry
9. Stop

PROGRAM:

LABEL	ADDRESS	MNEMONICS	OPCODE			COMMENTS
	4100	LDA 4500	3A	00	45	Load the accumulator from the location 4500
	4103	MOV B, A	47			Move the content from A to B register
	4104	LDA 4502	3A	02	45	Load the accumulator from the location 4502
	4107	MVI C, 00	0E	00		Load the C register with 00
	4109	ADD B	80			Add the content with B register
	410A	DAA	27			Decimal Adjust to the result of the accumulator
	410B	JNC L1	D2	0F	41	Jump on no carry to location to L1
	410E	INR C	0C			Increment the C register
L1	410F	STA 5000	32	00	50	Store the result in location 5000
	4112	MOV A, C	79			Move the content from C register to A register
	4113	STA 5002	32	02	50	Store the result in location 5002
	4116	HLT	76			Stop

RESULT:

Thus an assembly language program to subtract 2 digit BCD data

BEFORE EXECUTION

ADDRESS	DATA
4500	05
4502	05

AFTER EXECUTION

ADDRESS	DATA
4500	05
4502	05
5000	00
5002	10

b) SUBTRACT TWO 2 - DIGIT BCD DATA

AIM:

To write an assembly language program to subtract 2 digit BCD data

ALGORITHM:

1. Load the subtrahend in accumulator and move it to B register.
2. Move 99 to accumulator and subtract the contents of B register from accumulator
3. Increment the accumulator
4. Move the content of A reg to B reg
5. Load the minuend in the A reg
6. Add the contents of B reg to A reg
7. Execute the DAA operation
8. Store the result in memory
9. Stop.

PROGRAM:

LABEL	ADDRESS	MNEMONICS	OPCODE			COMMENTS
	4100	LDA 4500	3A	00	45	Load the accumulator to location 4500
	4103	MOV B, A	47			Move the content from A register to B register
	4104	LDA 4502	3A	02	45	Load the accumulator to location 4502
	4107	MVI C, 00	0E	00		Load the C register with 00
	4109	SUB B	90			Subtract the content with B register
	410A	DAA	27			Decimal Adjust to the result of the accumulator
	410B	JNC L1	D2	0F	41	Jump on no carry to location to L1
	410E	INR C	0C			Increment the C register
L1	410F	STA 5000	32	00	50	Store the result in location 5000
	4112	MOV A, C	79			Move the content from C register to A register
	4113	STA 5002	32	02	50	Store the result in location 5002
	4116	HLT	76			Stop

RESULT:

Thus an assembly language program to subtract 2 digit BCD data is written and executed.

BEFORE EXECUTION

ADDRESS	DATA
4500	05
4502	05

AFTER EXECUTION

ADDRESS	DATA
4500	05
4502	05
5000	00
5000	00

III) BIT MANIPULATION

a) MASKING OF MSB

AIM:

To write an assembly language program to mask the lower order nibbles of the given data.

ALGORITHM:

1. Load the data in the A reg
2. AND the data with the required task
3. Store the result
4. Stop

PROGRAM:

LABEL	ADDRESS	MNEMONICS	OPCODE			COMMENTS
	4100	LDA 4500	3A	00	45	Load the accumulator content to location 4500
	4103	ANI 0F	E6	0F		AND the content with the data 0F
	4105	STA 5000	32	00	50	Store the result in location 5000
	4108	HLT	76			Stop

RESULT:

Thus masking of lower order nibble of the given data is written and executed.

BEFORE EXECUTION

ADDRESS	DATA
4500	**FF**

AFTER EXECUTION

ADDRESS	DATA
4500	**FF**
5000	**F0**

b) SETTING OF MSB

AIM:

To write an assembly language program to set the lower order nibbles of the given data

ALGORITHM:

1. Load the data in the A reg
2. OR the data with the required task
3. Store the result
4. Stop

PROGRAM:

RESULT:

LABEL	ADDRESS	MNEMONICS	OPCODE			COMMENTS
	4100	LDA 4500	3A	00	45	Load the accumulator content to location 4500
	4103	ORI F0	F6	0F		OR the content with the data 0F
	4105	STA 5000	32	00	50	Store the result in location 5000
	4108	HLT	76			Stop

RESULT:

Thus the lower order nibbles of the given data are set**.**

BEFORE EXECUTION

ADDRESS	DATA
4500	**F0**

AFTER EXECUTION

ADDRESS	DATA
4500	**F0**
5000	**FF**

3. Programming with 8086- String manipulations, search find and replace, copy operation, sorting (PC required)

Program No: 3(a)

Date:

a) STRING MANIPULATIONS

i) String comparison

Aim:

To compare two set of strings and to display 00 if the two string are equal and to display FF if they are not equal.

Algorithm:

1. **Move the source address and the destination address to SI and DI respectively**
2. **Clear direction flag**
3. **Compare the string word. If they are equal go to step 5**
4. **Move FF to a location. Go to step 6**
5. **Move 00 to a location.**
6. **Stop the execution.**

Program:

```
CODE SEGMENT
        ASSUME CS: CODE, DS: CODE
        ORG 1000 H
        MOV SI, 1100 H
        MOV DI, 2100 H
        CLD
        CMPSW
        JE L1
        MOV [1200], 00FF H
        JMP L2
L1      MOV [1200], 0000 H
```

L2 HLT

CODE ENDS

END

RESULT:

Thus the two strings are compared.

AFTER EXECUTION:

	ADDRESS	DATA
STRING 1:	**1100**	**4587**
STRING 2:	**2100**	**4587**
RESULT	**1200**	**0000**

ii) EQUALITY OF A SET OF STRING

AIM;

To compare a set of string with another set and find the number of equal and non equal strings.

ALGORITHM:

1. **Clear the DX and BX registers**
2. **Move source address and destination address to SI and DI**
3. **Initialize count CX. Clear direction flag**
4. **Compare string word**
5. **If they are equal go to step 7**
6. **Increment DX go to step 8**
7. **Increment BX**
8. **Decrement the CX register, If CX not equal to zero go to step 4**
9. **Move BX, AX to some memory location.**
10. **Stop the process**

PROGRAM:

```
CODE SEGMENT
        ASSUME CS: CODE, DS: CODE
        ORG 1000 H
        MOV DX, 0000 H
        MOV BX, DX
        MOV SI, 1100 H
        MOV DI, 1200 H
        MOV CX, 0005 H
        CLD
L3      CMPSW
        JZ L1
        INC DX
        JMP L2
L1      INC BX
L2      LOOP L3
        MOV [1300], BX
        MOV [1302], DX
        HLT
CODE ENDS
```

END

RESULT:

Thus number of equal and non equal strings in a set of array is found

BEFORE EXECUTION

ADDRESS	DATA
1100	1111
1102	2222
1104	3333
1106	4444
1108	5555

SOURCE INDEX

ADDRESS	DATA
1200	1111
1202	2222
1204	4444
1206	6666
1208	8888

DESTINATION INDEX

AFTER EXECUTION

ADDRESS	DATA
1300	0002
1302	0003

Program No:3(b)

Date:

b) COPY A STRING TO A SET OF MEMORY LOCATION

Aim:

To copy a string in a set of memory location.

Algorithm:

1. **Move the content to CX register, destination address to DI and data to AX.**
2. **Clear the direction flag.**
3. **Store the string word.**
4. **Decrement the CX and if CX not equal to 0 go to step 3.**
5. **Otherwise halt the program.**

Program:

```
CODE SEGMENT
        ASSUME CS: CODE, DS: CODE
        ORG 1000 H
        MOV SI, 1100 H
        MOV DI, 1200 H
        MOV CX, 0005 H
        CLD
L1      MOVSW
        LOOP L1
        HLT
CODE ENDS

END
```

RESULT:

Thus data available in a source is copied to destination

BEFORE EXECUTION

ADDRESS	DATA
1100	1111
1102	2222
1104	3333
1106	4444
1108	5555

AFTER EXECUTION

ADDRESS	DATA
1200	1111
1202	2222
1204	3333
1206	4444
12058	5555

Program No:3(c)

Date:

c) SEARCH FIND AND REPLACE

AIM:

To search for a data in an array and to replace that data with the given value "DD"

ALGORITHM:

1. **Move the count to CX register**
2. **Move the starting address to SI and the string to be searched into AL**
3. **Compare AL with data pointed by SI**
4. **If both are equal, replace the string with another value or else Increment the SI**
5. **Decrement the CX register and check for zero.**
6. **If there is no string found to be equal move 0000 to a location**
7. **Stop the program**

PROGRAM:

```
CODE SEGMENT
        ASSUME CS: CODE, DS: CODE
        ORG 1000 H

        MOV CX, 0004 H

        MOV SI, 1100 H

        MOV AL, 77 H
L2      CMP AL, [SI]

        JNZ L1

        MOV DL, 88 H

        MOV [SI], DL
L1      INC SI

        LOOP L2

        MOV BX, 0000 H

        MOV [1200] , BX

        HLT
CODE ENDS

END
```

RESULT:

`Thus the data is found and replace with the given value

BEFORE EXECUTION

ADDRESS	DATA
1100	52
1101	56
1102	57
1103	77

AFTER EXECUTION

ADDRESS	DATA
1200	88

Program No:3(d)

Date:

d) SORTING

ASCENDING ORDER

AIM:

To sort a set of numbers in ascending order

ALGORITHM

1. **Move the content to CX , decrement the count by one**
2. **Move start address to BX**
3. **Move CX to DI**
4. **Move BX pointed data to AX**
5. **Compare AX with next data**
6. **If borrow is set , go to step 9**
7. **Exchange AX and BX+2 value**
8. **Move accumulator value to BX**
9. **Add 2 to BX**
10. **Decrement DI**
11. **If DI not equal to zero go to step 4**
12. **Decrement CX. If CX not equal to zero goto step 2**
13. **Halt the execution**

PROGRAM:

```
CODE SEGMENT
        ASSUME CS: CODE, DS: CODE
        ORG 1000 H

        MOV CX, 0005 H
        DEC CX
L3      MOV BX, 1100 H
        MOV DI, CX
L2      MOV AX, [BX]
        CMP AX, [BX+2]
        JB L1
        XCHG AL, [BX+2]
        MOV [BX], AX
L1      ADD BX, 0002
        DEC DI
        JNZ L2
        LOOP L3
        HLT
CODE ENDS
```

END

RESULT:

Thus the set of numbers are stored in ascending order

BEFORE EXECUTION

ADDRESS	DATA
1100	0132
1102	B342
1104	000A
1106	1396
1108	5321

AFTER EXECUTION

ADDRESS	DATA
1100	000A
1102	0132
1104	1396
1106	5321
1108	B342

5. Interfacing with 8085

Program No: 5(a)

Date:

a) INTERFACING 8255 WITH 8085

AIM:

To initialize port A as a input port in mode 0 and to output the data set by switches through port A and store the data in the RAM location.

ALGORITHM:

1. Start
2. Move immediately the data to the accumulator then to the control register
3. Move the content of Port A to accumulator
4. Store the content of accumulator at some memory location
5. Stop the execution of the program.

PROGRAM:

LABEL	ADDRESS	MNEMONICS	OPCODE			COMMENTS
	4100	MVI A, 90	3E	90		Move the control word to the control register
	4102	OUT C6	D3	C6		
	4104	IN C0	DB	C0		Get the data from the 8255 and store it in location 5000
	4106	STA 5000	32	00	50	
	4109	HLT	76			Stop

RESULT:

Thus to initialize port A as a input port in mode 0 and to output the data set by switches through port A and store the data in the RAM location is written and executed.

The Switches : 1 1 1 1 1 1 1 1
Before execution : 5000 - default value
After execution : 5000 - FF

AIM:

To initialize port C as output port in mode 0 and to output data at port C to glow the LEDS accordingly.

ALGORITHM:

1. Load the control word to the control registers
2. Load the data in the A register
3. Put that data on the port C
4. Stop

PROGRAM:

LABEL	ADDRESS	MNEMONICS	OPCODE			COMMENTS
	4100	MVI A, 90	3E	90		Move the control word to the control register
	4102	OUT C6	D3	C6		
	4104	MVI A, 80	3E	80		Load the data 80 in accumulator and output in port C4
	4106	OUT C4	D3	C4		
	4108	HLT	76			Stop

RESULT:

Thus to initialize port C as output port in mode 0 and to output data at port C to glow the LEDS accordingly is written and executed.

When the input is 01 in the switch

Before execution:

PC 7	PC 5	PC 3	PC 2	PC 0
0	0	0	0	0

After execution:

PC 7	PC 5	PC 3	PC 2	PC 0
0	0	0	0	1

AIM:

To initialize port A as input port. Port B as output port in mode 0, to input data at port A as set by SPDT switches and to output the same data to port B to glow the LEDs accordingly.

ALGORITHM:

1. Load the control word to the control registers
2. Get the data in port A
3. Put the data in port B
4. Stop

PROGRAM:

LABEL	ADDRESS	MNEMONICS	OPCODE			COMMENTS
	4100	MVI A, 90	3E	90		Move the control word to the control register
	4102	OUT C6	D3	C6		
	4104	IN C0	DB	C0		Get the data and output at the port C
	4106	OUT C2	D3	C4		
	4108	HLT	76			Stop

RESULT:

Thus to initialize port A as input port. Port B as output port in mode 0, to input data at port A as set by SPDT switches and to output the same data to port B to glow the LEDs accordingly is written and executed.

When the input is 01 in the switch

Port B before execution:

PB 7	PB 6	PB 5	PB 4	PB 3	PB 2	PB 1	PB 0
0	0	0	0	0	0	0	0

Port B after execution:

PB 7	PB 6	PB 5	PB 4	PB 3	PB 2	PB 1	PB 0
0	0	0	0	0	0	0	1

Program No: 5(b)

Date:

b) INTERFACING 8253 WITH 8085

AIM:

To write an assembly language program to generate a square wave using 8253

ALGORITHM:

1. Select the channel and load the count
2. Use the proper control word and control register
3. Stop

PROGRAM:

LABEL	ADDRESS	MNEMONICS	OPCODE			COMMENTS
	4100	MVI A, 36	3E	36		Select the channel and put it in the proper control register
	4102	OUT CE	D3	CE		
	4104	MVI A, 0A	3E	0A		Load the count to the 8253
	4106	OUT C8	D3	C8		
	4108	MVI A, 00	3E	00		
	410A	OUT C8	D3	C8		
	410C	HLT	76			Stop

RESULT:

Thus an assembly language program to generate a square wave using 8253 is written and executed.

Program No: 5(c)

Date:

c) INTERFACING 8251 WITH 8085

AIM:

To write an assembly language program to interface 8251 with 8085 and to transmit and receive a data

ALGORITHM:

1. Initialize the 8253 and 8251
2. Load the control word in the control register
3. Give the input in the main program
4. Start a separate program in another memory location to get the transmitted data
5. Give the proper control word for the reception.
6. Stop

PROGRAM:

TRANSMISSION:

LABEL	ADDRESS	MNEMONICS	OPCODE			COMMENTS
	4100	MVI A, 36	3E	36		Move the control word to the control register
	4102	OUT CE	D3	CE		
	4104	MVI A, 0A	3E	0A		Get the data and output at the port C
	4106	OUT C8	D3	C8		
	4108	MVI A, 00	3E	00		
	410A	OUT C8	D3	C8		
	410C	MVI A, 4E	3E	4E		Initialize the control words from 8251
	410E	OUT C2	D3	C2		
	4110	MVI A, 37	3E	37		
	4112	OUT C2	D3	C2		
	4114	MVI A, 55	3E	55		Transmit the data
	4116	OUT C0	D3	C0		
	4118	RST 1	CF			Reset the processor

RECEPTION:

LABEL	ADDRESS	MNEMONICS	OPCODE			COMMENTS
	4200	IN C0	DB	C0		Receive the data
	4202	STA 4500	32	00	45	Store in the location 4500
	4205	RST 1	CF			Reset

RESULT:

Thus an assembly language program to interface 8251 with 8085 and to transmit and receive a data is written and executed.

The transmitted data is on location 4115 '55'

The received data is on location 4500 '55'

Program No: 5 (d)

Date:

d) INTERFACING 8279 WITH 8085

AIM:

To write an assembly language program to initialize and display a letter by interfacing 8279 with 8085

ALGORITHM:

1. With the help of control word set and clear the words
2. Initialize the write mode
3. Display the word
4. Stop

PROGRAM:

LABEL	ADDRESS	MNEMONICS	OPCODE			COMMENTS
	4100	MVI A, 00	3E	00		Load the content in 00 in A register
	4102	OUT C2	D3	C2		Output the data in proper control register
	4104	MVI A, CC	3E	CC		Clear the segments and put that data in the control register
	4106	OUT C2	D3	C2		
	4108	MVI A, 90	3E	90		Set the 8279 in write mode with the data 90
	410A	OUT C2	D3	C2		Put that control word in control register
	410C	MVI A, 88	3E	88		Load the first data 88 in A register
	410E	OUT C0	D3	C0		Put the data in the proper data bus C0
	4110	MVI A, FF	3E	88		Load the next data FF in the A register
	4112	OUT C0	D3	C0		Put the data in the proper data bus C0
	4114	OUT C0	D3	C0		Put the data in the proper data bus C0
	4116	OUT C0	D3	C0		Put the data in the proper data bus C0
	4118	OUT C0	D3	C0		Put the data in the proper data bus C0
	411A	OUT C0	D3	C0		Put the data in the proper data bus C0
	411C	HLT	76			Stop

RESULT:

Thus an assembly language program to initialize and display a letter by interfacing 8279 with 8085 is written and executed.

AIM:

To write an assembly language program to initialize and to roll the word by interfacing 8279 with 8085 is written and executed successfully.

ALGORITHM:

1. With the help of control word set and clear the words
2. Initialize the write mode
3. Roll the word.
4. Stop

PROGRAM:

LABEL	ADDRESS	MNEMONICS	OPCODE			COMMENTS
START	4100	LXI H, 4500	21	00	45	Initialize the memory location
	4103	MVI D, 0F	16	0F		Move the data 0F in the D register
	4105	MVI A, 10	3E	10		Load the data 10 in the A register and put the data in the control register
	4107	OUT C2	D3	C2		
	4109	MVI A, CC	3E	CC		Clear the segments and put that data in the control register
	410B	OUT C2	D3	C2		
	410D	MVI A, 90	3E	90		Set the 8279 in write mode with the data 90
	410F	OUT C2	D3	C2		Put that control word in control register
LOP	4111	MOV A, M	7E			Load the first data in the A register
	4112	OUT C0	D3	C0		Out put that data in the data bus
	4114	CALL DELAY	CD	1F	41	Call to the location DELAY
	4117	INX H	23			Increment the HL register pair
	4118	DCR D	15			Decrement the D register
	4119	JNZ LOP	C2	11	41	Jump on no zero to location labeled LOP
	411C	JMP START	C3	00	41	Jump to location labeled START
DELAY	411F	MVI B, A0	06	A0		Move the data A0 to B register
LOP1	4121	MVI C, FF	0E	FF		Load the data FF in the C register
LOP2	4123	DCR C	0D			Deferment the C register
	4124	JNZ LOP2	C2	23	41	Jump on no zero to LOP 2
	4127	DCR B	05			Decrement the B register

	4128	JNZ LOP1	C2	21	41	Jump on no zero to location LOP 1
	412B	RET	C9			Return to the main program

RESULT:

Thus an assembly language program to initialize and to roll the word by interfacing 8279 with 8085 is written and executed successfully.

6. Interfacing with 8086

INTERFACING 8255 WITH 8086

AIM:

To write an assembly language program to initialize Port A as an input port in mode 0

ALGORITHIM:

1. Start
2. Move immediately the data to the accumulator then to the control register
3. Move the content of Port A to accumulator
4. Store the content of accumulator at some memory location
5. Stop the execution of the program.

PROGRAM:

LABEL	ADDRESS	MNEMONICS	COMMENTS
	1000	MOV SI, 1500	Initialize the Source index with location 1500
	1003	MOV AL, 90	Move the data 90 to AL register
	1005	OUT C6, AL	Out the content from AL register to address C6
	1007	IN AL, C0	Get the data from C0 and out in A register
	1009	MOV [SI], AL	Move the content from AL register to SI
	100B	HLT	Stop

RESULT:

Thus an assembly language program to initialize Port A as an input port in Mode 0 is written and executed.

The Switches : 1 1 1 1 1 1 1 1
Before execution : 1500 - default value
After execution : 1500 - FF

AIM:

To write an assembly language program to initialize Port A as input port and port B as output port in mode 0

ALGORITHIM:

1. Load the control word to the control registers
2. Load the data in the A register
3. Put that data on the port C
4. Stop

PROGRAM:

LABEL	ADDRESS	MNEMONICS	COMMENTS
	1000	MOV AL, 90	Load the data 90 in location AL
	1002	OUT C6, AL	Out the data to C6 port address
	1004	IN AL, C0	Get the data from C0 and put it in location Al
	1006	OUT C2, AL	Out put the data from AL register to port address C2
	1008	HLT	Stop

RESULT:

Thus an assembly language program to initialize Port A as input port and port B as output port in mode 0 is written and executed.

When the input is 01 in the switch

Port B before execution:

PB 7	PB 6	PB 5	PB 4	PB 3	PB 2	PB 1	PB 0
0	0	0	0	0	0	0	0

Port B after execution:

PB 7	PB 6	PB 5	PB 4	PB 3	PB 2	PB 1	PB 0
0	0	0	0	0	0	0	1

AIM:

To write an assembly language program to initialize Port C as output port in Mode 0

ALGORITHIM:

1. Load the control word to the control registers
2. Get the data in port A
3. Put the data in port B
4. Stop

PROGRAM:

LABEL	ADDRESS	MNEMONICS	COMMENTS
	1000	MOV AL, 90	Load the data 90 in location AL
	1002	OUT C6, AL	Out the data to C6 port address
	1004	MOV AL, 80	Move the data 80 to AL
	1006	OUT C4, AL	Out the data present in AL register to C4 port address
	1008	HLT	Stop

RESULT:

Thus an assembly language program to initialize Port C as output port in Mode 0 is written and executed.

When the input is 01 in the switch

Before execution:

PC 7	PC 5	PC 3	PC 2	PC 0
0	**0**	**0**	**0**	**0**

After execution:

PC 7	PC 5	PC 3	PC 2	PC 0
0	**0**	**0**	**0**	**1**

INTERFACING 8253 WITH 8086

AIM:

To write an assembly language program to generate a square wave form by interfacing 8253 with 8086

ALGORITHIM:

1. Select the channel and load the count
2. Use the proper control word and control register
3. Stop

PROGRAM:

LABEL	ADDRESS	MNEMONICS	COMMENTS
	1000	MOV AL, 36	Move the control word to the control register
	1002	OUT CE, AL	
	1004	MOV AL, 10	Load the count to the proper address
	1006	OUT C8, AL	
	1008	MOV AL, 00	
	100A	OUT C8, AL	
	100C	HLT	Stop

RESULT:

Thus an assembly language program to generate a square wave form by interfacing 8253 with 8086 is written and executed.

INTERFACING 8251 WITH 8086

AIM:

To write an assembly language program to interface 8251 with 8086 and to transmit and receive a data serially

ALGORITHIM:

1. Initialize the 8253 and 8251
2. Load the control word in the control register
3. Give the input in the main program
4. Start a separate program in another memory location to get the transmitted data
5. Give the proper control word for the reception.
6. Stop

PROGRAM:

TRANSMISSION:

LABEL	ADDRESS	MNEMONICS	COMMENTS
	1000	MOV AL, 36	Move the control word to the control register
	1002	OUT CE, AL	
	1004	MOV AL, 10	Get the data and output at the port C
	1006	OUT C8, AL	
	1008	MOV AL, 00	
	100A	OUT C8, AL	
	100C	MOV AL, 4E	Initialize the control words from 8251
	100E	OUT C2, AL	
	1010	MOV AL, 37	
	1012	OUT C2, AL	
	1014	MOV AL, 55	Transmit the data
	1016	OUT C0, AL	
	1018	INT 2	Reset the processor

RECEPTION:

LABEL	ADDRESS	MNEMONICS	COMMENTS
	1200	IN AL, C0	Receive the data
	1202	MOV BX, 1250	Store in the location
	1205	MOV [BX], AL	
	1207	INT 2	Reset

RESULT:

Thus an assembly language program to interface 8251 with 8086 and to transmit and receive a data serially is written and executed.

The transmitted data is on location 1255 '55'

The received data is on location 1250 '55'

INTERFACING 8279 WITH 8086

AIM:

a) To write an assembly language program to initialize and display a word by interfacing 8279 with 8085

ALGORITHIM:

1. With the help of control word set and clear the words
2. Initialize the write mode
3. Display the word
4. Stop

PROGRAM:

LABEL	ADDRESS	MNEMONICS	COMMENTS
	1000	MOV AL, 00	Load the content in 00 in AL register
	1002	OUT C2, AL	Output the data in proper control register
	1004	MOV AL, CC	Clear the segments and put that data in the control register
	1006	OUT C2, AL	
	1008	MOV AL, 90	Set the 8279 in write mode with the data 90
	100A	OUT C2, AL	Put that control word in control register
	100C	MOV AL, 88	Load the first data 88 in Al register
	100E	OUT C0, AL	Put the data in the proper data bus C0
	1010	MOV AL, 88	Load the next data 88 in the Al register
	1012	OUT C0, AL	Put the data in the proper data bus C0
	1014	MOV AL, FF	Load the next data 88 in the Al register
	1016	OUT C0, AL	Put the data in the proper data bus C0
	1018	OUT C0, AL	Put the data in the proper data bus C0
	101A	OUT C0, AL	Put the data in the proper data bus C0
	101C	HLT	Stop

RESULT:

Thus an assembly language program to initialize and display a word by interfacing 8279 with 8085 is written and executed successfully.

AIM:

To initialize an to roll the a word by interfacing 8279 with the 16 bit processor 8086

ALGORITHIM:

1. With the help of control word set and clear the words
2. Initialize the write mode
3. Roll the word.
4. Stop

PROGRAM:

LABEL	ADDRESS	MNEMONICS	COMMENTS
START	1000	MOV SI, 1500	Initialize the memory location
	1003	M0V DL, 0F	Move the data 0F in the D register
	1006	MOV AL, 10	Load the data 10 in the A register and put the data in the control register
	1008	OUT C2, AL	
	100A	MOV AL, CC	Clear the segments and put that data in the control register
	100C	OUT C2, AL	
	100E	MOV AL, 90	Set the 8279 in write mode with the data 90
	1010	OUT C2, AL	Put that control word in control register
LOP	1012	MOV AL, [SI]	Load the first data in the A register
	1013	OUT C0, AL	Out put that data in the data bus
	1015	CALL DELAY	Call to the location DELAY
	1018	ADD [SI+2]	Increment the SI register pair
	1019	DCR D	Decrement the D register
	101A	JNZ LOP	Jump on no zero to location labeled LOP
	101D	JMP START	Jump to location labeled START
DELAY	1020	MOV BL, A0	Move the data A0 to B register
LOP1	1022	MOV CL, FF	Load the data FF in the C register
LOP2	1024	DCR C	Deferment the C register
	1025	JNZ LOP2	Jump on no zero to LOP 2
	1028	DCR B	Decrement the B register
	1029	JNZ LOP1	Jump on no zero to location LOP 1
	102C	RET	Return to the main program

RESULT:

Thus an assembly language program to initialize and to roll the word by interfacing 8279 with the 16 bit processor 8086 is written and executed successfully.

PROGRAM FOR WRITE SOME DATA IN A FILE USING DOS CALLS

Aim:

To write an assembly language program to write some data in a file using BIOS/DOS calls.

Algorithm:

1. Initialisé thé data segment
2. Assign number of datas to be read form Key board
3. Store the values in a memory
4. Initialize interrupt for data read from key board
5. Initialize interrupt for data write from key board
6. Assign number of datas to be written
7. Initialize interrupt for file close.
8. Terminate the program.

Program:

```
        CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
        ORG 1000H
        MOV AX,DATA       ;DATA SEGMENT INITIALISATOIN
        MOV DS,AX

        MOV CX,0010H      ;NO OF DATA TO BE READ FROM KEYBOARD
        MOV SI,1400H      ;STORE THE VALUES IN A MEMORY
    L1: MOV AH,00H
        INT 16H           ;INTERRUPT FOR DATA READ FROM KEYBOARD
        MOV [SI],AL
        INC SI
        LOOP L1

        MOV AH,3DH        ;FILE OPENED FOR DATA WRITE
        MOV AL,01H
        LEA DX,FILENAME   ;FILE NAME
        INT 21H
        MOV SI,1500H
        MOV [SI],AX
        MOV AH,40H        ;INTERRUPT FOR DATA WRITE IN A FILE
        MOV DX,1400H
        MOV CX,0050H      ;NO OF DATA'S TO BE WRITTEN
        MOV BX,[SI]
        INT 21H
        MOV AH,3EH        ;INTERRUPT FOR FILE CLOSE
        INT 21H
```

```
        MOV AH,4CH
        INT 21H         ;PROGRAM TERMINATION
CODE  ENDS
DATA    SEGMENT
          ORG 1300H
         FILENAME DB 'TEST.ASM'
        ORG 1400H
DATA    ENDS
        END
```

Result:

Thus the ALP is written for reading the key board using BIOS/DOS calls.

TEST.ASM: 4 5 8 79 6 3 21 47 5 8

PROGRAM FOR DISPLAY A STRING IN MONITOR DISPLAY

Aim:

To write an assembly language program to display a string using BIOS calls.

Algorithm:

1. Initialisé thé data segment
2. Initialisé the interrupt for display
3. Assign the string to display
4. Terminate the programme

Program:

```
        CODE SEGMENT
        ASSUME CS:CODE,DS:CODE
        ORG 1000H
        MOV DX,1200H
        MOV AH,09H
        INT 21H
        MOV AH,4CH
        INT 21H
        ORG 1200H
        DB 'HAI $'
CODE    ENDS
        END
```

Result:

Thus an ALP has written to display a string using BIOS calls.

Displayed Word: HAI

PROGRAM FOR CREATING A FILE USING BIOS/DOS CALLS

Aim:

To write an assemble language program to create a file using BIOS/DOS calls

Algorithm:

1. Initialisé thé data segment
2. Give the name of the file to be created
3. Initialize interrupt for file creation
4. Initialize interrupt for program termination
5. Terminate the program.

Program:

```
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
ORG 1000H
MOV AX, DATA          ;INITIALISATION OF DATA SEGMENT
MOV DS, AX
MOV DX, 1300H         ;FILE NAME
MOV CX, 0000H         ; FILE ATRIBUTES
MOV AH, 3CH           ;INTERRUPT FOR FILE CREATION
INT 21H
MOV AH,4CH            ;INTERRUPT FOR PROGRAM TERMINATION
INT 21H
CODE ENDS
DATA SEGMENT
ORG 1300H
DB 'SIT.ASM'      ; FILE NAME TO BE CREATED
DATA ENDS
END
```

RESULT:

Thus an ALP is written and executed to create a file using BIOS/DOS calls

Created File:

SIT.ASM

PROGRAM FOR DELETE A FILE

Aim:

To write an ALP to delete a file using BIOS/DOS calls.

Algorithm:

1. Initialisé thé data segment
2. Give the name of the file to be deleted
3. Initialize interrupt for file deletion
4. Initialize interrupt for program termination
5. Give the name of the file to be deleted
6. Terminate the program.

Program:

```
        CODE    SEGMENT
        ASSUME CS:CODE, DS:DATA
        ORG 1000H
        MOV AX, DATA
        MOV DS, AX
        MOV DX, 1300H ; FILE NAME TO BE DELETED
        MOV AH, 41H     ; INTERRUPT FOR FILE DELETION
        INT 21H
        MOV AH, 4CH    ; INTERRUPT PROGRAM TERMINATION
        INT 21H
    CODE    ENDS
    DATA    SEGMENT
        ORG 1300H
        DB 'SIT1.ASM'     ; FILE TO BE DELETED
    DATA    ENDS
        END
```

RESULT:

Thus an ALP was written to delete a file using BIOS/DOS calls

Deleted File:

SIT1.ASM

PROGRAM FOR FILE RENAME

Aim:
To write an ALP to rename a file using BIOS/DOS calls.

Algorithm:
1. Initialisé thé data segment
2. Initialisé the extra data segment
3. Initialize interrupt for file renaming
4. Initialize interrupt for file termination
5. Give the name of the file to be renamed
6. Terminate the program.

Program:

```
        CODE    SEGMENT
        ASSUME CS:CODE,DS:DATA,ES:DATA
        ORG 1000H
        MOV AX,DATA
        MOV DS,AX      ;INITIALIZE DATA SEGMENT
        MOV ES,AX      ;INITIALIZE EXTRA DATA SEGMENT
        MOV DX,1300H
        MOV [DI],1500H
        LEA DX,OLDNAME
        LEA DI,NEWNAME
        MOV AH,56H    ;INTERRUPT FOR FILE RENAMING
        INT 21H
        MOV AH,4CH     ;FILE TERMINATION INTERRUPT
        INT 21H
    CODE    ENDS
    DATA    SEGMENT
        ORG 1300H
        OLDNAME DB 'SIT.ASM'
        ORG 1500H
        NEWNAME DB 'SIT1.ASM'
    DATA    ENDS
        END
```

RESULT:

Thus an ALP was written to rename a file using BIOS/DOS calls

OLDNAME 'SIT.ASM'
NEWNAME 'SIT1.ASM'

8051 Microcontroller based experiments – simple assembly language programs

MULTIPLICATION OF 8 BIT NUMBERS

Aim:

To write a ALP for multiplying 8 bit numbers

Algorithm:

1. Get the multiplier and multiplicand form a memory location pointed by the data pointer
2. Multiply the two data by using the instruction MUL
3. Store the product.
4. Stop the execution.

PROGRAM:

```
Org 4100
MOV DPTR, #4500
MOVX A, @DPTR
MOV B, A
INC DPTR
MOV A, @DPTR
MUL AB
INC DPTR
MOVX @DPTR, A
L1:SJMP L1
```

RESULT:

Thus the two 8 bit numbers are multiplied and the product is stored.

Before Execution:		**After execution:**	
4500	02	4500	02
4501	04	4501	04
4502	00	4502	08

DIVISION OF TWO 8 BIT NYMBERS

Aim:

To write a ALP for dividing two 8 bit numbers

Algorithm:

1. Get the divisor and dividend form a memory location pointed by the data pointer
2. Multiply the two data by using the instruction DIV
3. Store the quotient and reminder.
4. Stop the execution.

Programs:

```
Org 4100
MOV DPTR, #4500
MOVX A, @DPTR
MOV B, A
INC DPTR
MOV A, @DPTR
DIV AB
INC DPTR
MOVX @DPTR, A
INC DPTR
MOV A, B
MOVX @DPTR, A
L1:SJMP L1
```

RESULT:

Thus the two 8 bit numbers are divided and the reminder and quotient is stored.

Before Execution:		**After execution:**	
4503	02	4500	02
4504	04	4501	04
4505	00	4502	02
4506	00	4503	00

STEPPER MOTOR

Aim:

To run stepper motor in both forward and reverse direction.

Algorithm:

1. Initialise the Look up Table

2. Output the data.

3. Choose the Direction of Rotation.

4. Initiliase the Delay for rotation.

5. Decrement the count.

6. Stop the Program.

Program:

```
ORG 4100H
START:    MOV DPTR,#4500H
MOV R0,#04
J0:MOVX A,@DPTR
PUSH DPH
PUSH DPL
MOV DPTR,#FFC0H
MOV R2,#04H
MOV R1,#FFH
DLY1:MOV R3,#FFH
DLY:DJNZ R3,DLY
DJNZ R1,DLY1
DJNZ R2,DLY1
MOV @DPTR,A
POP DPL
POP DPH
INC DPTR
DJNZ R0.JO
```

SJMP START

END

TABLE;

DB 0A

09 06

05 05

06 09

0A DB

RESULT:

Thus an program has written to run stepper motor in both forward and reverse direction.

Appendix 1

To enter the program in 8085 processor:

Step 1: <Restart>

Step 2: <Sub>

Step 3: <Enter the address field>

Step 4: <Next> (address field increments automatically)

Step 5: <Enter the opcode/operand>

Step 6: <Next>

To execute the program:

Step 1: <Restart>

Step 2: <Go>

Step 3: <Enter the starting address of the program>

Step 4: <Execute>

To check the result stored in the address field:

Step 1: <Restart>

Step 2: <Sub>

Step 3: <Enter the address where the result has been stored>

Step 4: <Next>

Appendix 2

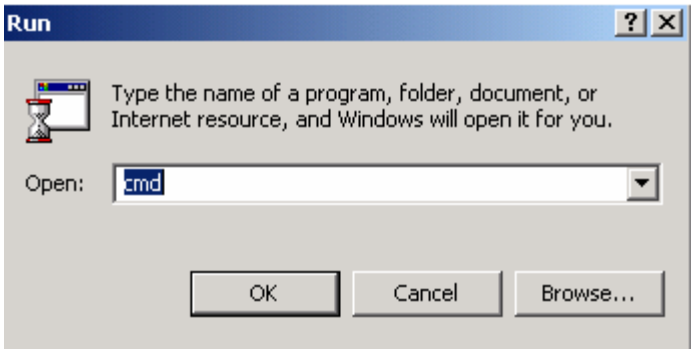
8255	Programmable peripheral interface
8279	Key board and display controller
8253	Programmable Interval timer
8251	Universal synchronous asynchronous receiver transmitter

Appendix 3

How to Interface the 8086 with the PC??

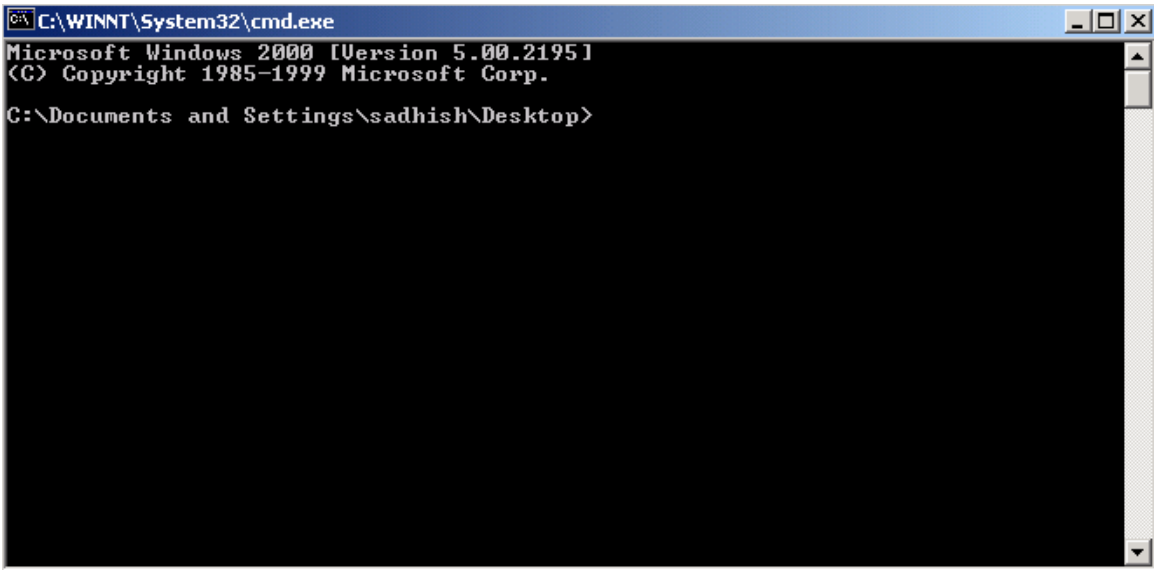
Step1:

From the task bar of the PC go to START menu and select RUN



Step 2:

In the RUN command type “cmd or command” and press OK it will go to the DOS prompt



Step3:

Exit from the current directory – type **CD**

Step 4:

Now type **CD 8086** (to enter into the 8086 sub directory)

Step 5:

Then type **EDIT** (to enter into the program window)

Step 6:

Create a new file to enter the 8086 programs

Step 7:

Before entering the program the following default statement are to be entered

CODE SEGMENT
ASSUME CS: CODE, DS: CODE
ORG *Starting Address*
.
.
.
.program
.
.
.HLT
CODE ENDS

END

Step 8:

Then save the program with the extension ".ASM" and then go to the command window.

Step 9:

C:\8086\> MASM filename, , ; (to create the opcode and machine language)

C:\8086\> LINK filename , , ; (to create the executable file)

C:\8086\> DEBUG filename.exe (to create binary file for downloading from host to 8086)

-RCX (Register CX for getting the count)

The count will be displayed after executing the above syntax

i. Load the last two bytes and press enter

-N file name (enter the name if the file) with extension ". BIN"

-WCS: staring address of the program (Write the Code Segement)

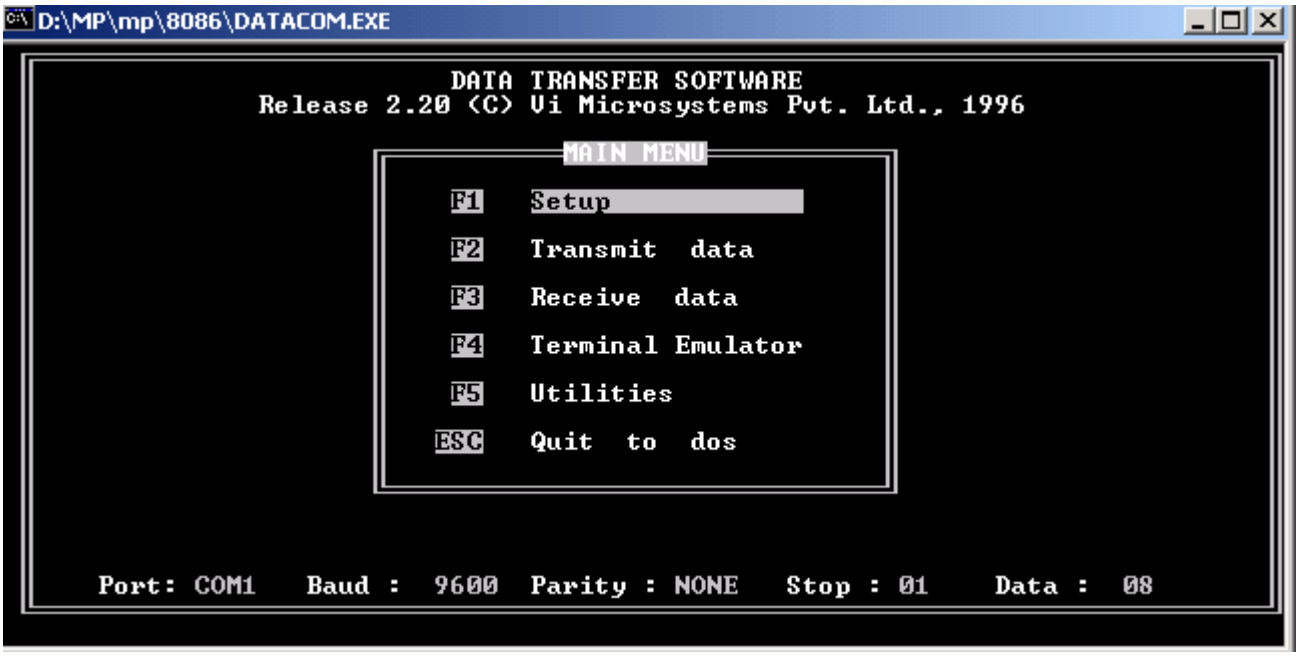
-Q (quit)

To download the binary data(program) from host to 8086 for that follow the steps.

Step 10

C:\8086\> DATACOM

The DATACOM menu is displayed



In this menu select "Transmit data" and press enter

Step 11:

Type the filename with extension ". BIN"

Step 12:

Press "enter" thrice

Mean while

In the processor kit

Type <SI staring address of the program>

Then press enter and execute the program in the processor by the following steps

(Wait for the downloading to be completed)

After the prompt appears,

<GO starting address of the program >

<RESET>

SW Enter the address in which the result is stored. (To view the output data)

<RESET>

To unassemble the program

<U staring address>

To give the input:

<SW address in which the input is to be given>

APPENDIX IV

Applications of Microprocessors

Microprocessor Applications and Real-Time Systems

- Microprocessors are used to handle a set of tasks that control one or more external events or systems.
- Microprocessors are typically used in either ***reactive*** or ***embedded*** systems.
    - ***Reactive systems*** are those that have an ongoing interaction with their environment - for example, a fire-control system that constantly reacts to buttons pressed by a pilot.
    - ***Embedded systems*** are those used to control specialized hardware in which the computer system is installed - for example, the microprocessor system used to control the fuel/air mixture in the carburetor of many automobiles.
        - In embedded systems the software system is completely encapsulated by the hardware that it controls.
- Often the processor is required to manage various different tasks that have to be scheduled somehow and must also deal with outside interrupt sources such as an alarm when something goes wrong.
- ***Real-time systems*** are those in which timeliness is as important as the correctness of the outputs, although this does NOT mean that they have to be "fast systems".
    - A real-time system does not have to process data in microseconds to be considered real-time - it must simply have response times that are constrained and thus predictable.
    - Control and measurements
    - Data processing and communications

Applications of Microcontrollers:

Microcontroller perfectly fits many uses, from automotive industries and controlling home appliances to industrial instruments, remote sensors, electrical door locks and safety devices. It is also ideal for smart cards as well as for battery supplied devices because of its low consumption.

EEPROM memory makes it easier to apply microcontrollers to devices where permanent storage of various parameters is needed (codes for transmitters, motor speed, receiver frequencies, etc.). Low cost, low consumption, easy handling and flexibility make

PIC16F84 applicable even in areas where microcontrollers had not previously been considered (example: timer functions, interface replacement in larger systems, coprocessor applications, etc.).

In System Programmability of this chip (along with using only two pins in data transfer) makes possible the flexibility of a product, after assembling and testing have been completed. This capability can be used to create assembly-line production, to store calibration data available only after final testing, or it can be used to improve programs on finished products.