Sri Krishna Engineering College

Panapakkam, Chennai – 601 301

Lab Manual

CS2259 MICROPROCESSORS LAB

Department of Electronics and Communication Engineering

List of Experiments:

Ex.No	Title of Experiment
	8085 Programming:
1	**Basic Arithmetic Operations**
2	**Searching & Sorting**
3	**Code Conversion & BCD Arithmetic**
	8086 Programming:
4	**Basic Arithmetic & Sorting**
5	**String Manipulation**
	Interfacing:
6	**8253 & 8279 Interfacing**
7	**8255 & 8251 Interfacing**
8	**Stepper & DC Motor Control**
9	**8051 Basic Programming**
10	**Communication between PC & 8085/8086/8051**

Ex. No: 01 **Basic arithmetic operations in 8085**

AIM

To perform basic arithmetic operations in 8085 Microprocessor.

APPARATUS REQUIRED:

8085 Microprocessor Kit.

ALGORITHM:

ADDITION:

1. Start the program
2. Load the value
3. Add the two numbers
4. Store the control of accumulator in 1 memory location
5. Check the carry flag
6. If the carry flag is set store 01 in another memory location
7. Halt the program

SUBTRACTION:

1. Start the program
2. Load the data in accumulator
3. Subtract the received data
4. If borrow=0, then go to step 6
5. Else increment array
6. Store the result
7. Stop the program

MULTIPLICATION:

1. Start the program
2. Get the multiplier to the multiplication
3. Initialize carry register
4. Initialize the register with multiplier
5. Clear accumulator
6. Add the accumulator with multiplicand
7. If carry, increment the carry register
8. Decrement register
9. If 0 go to 11
10. Else go to 6

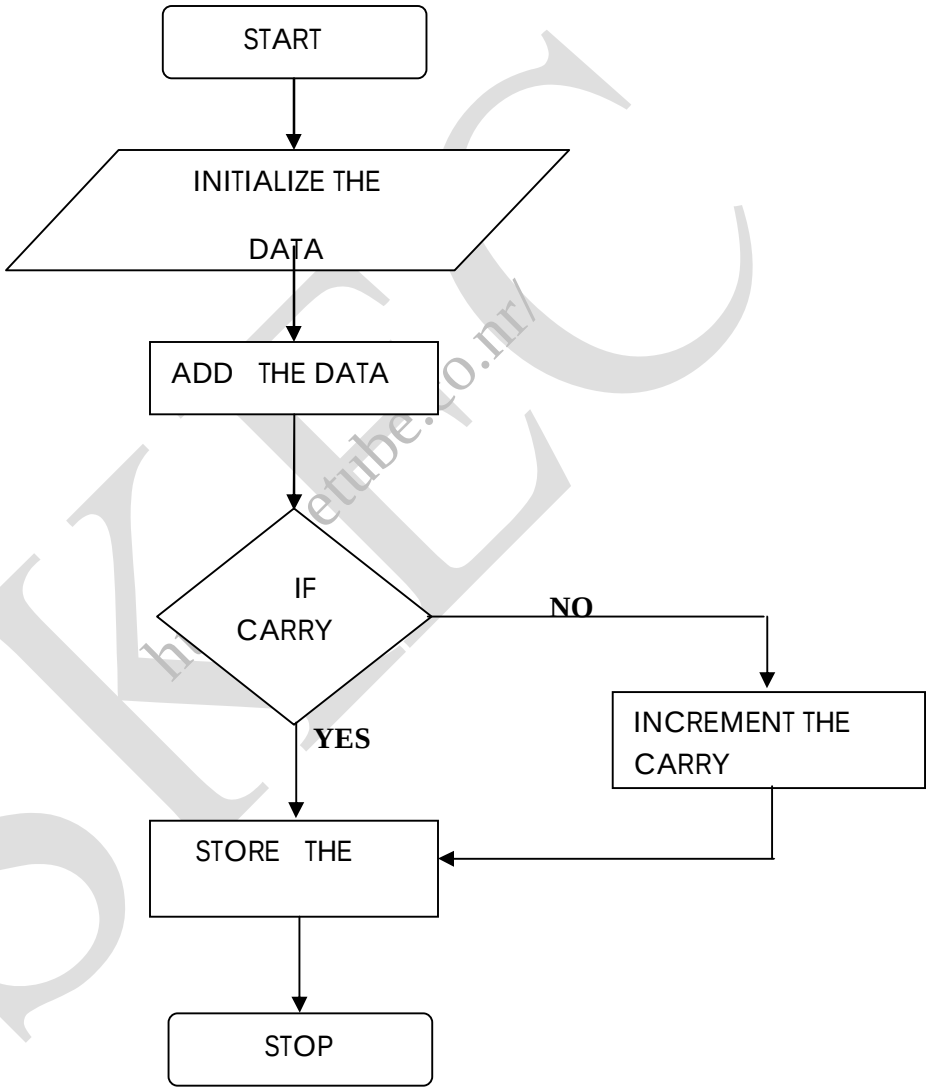
11. Store accumulator
12. Store carry register
13. Stop the program.

DIVISION:

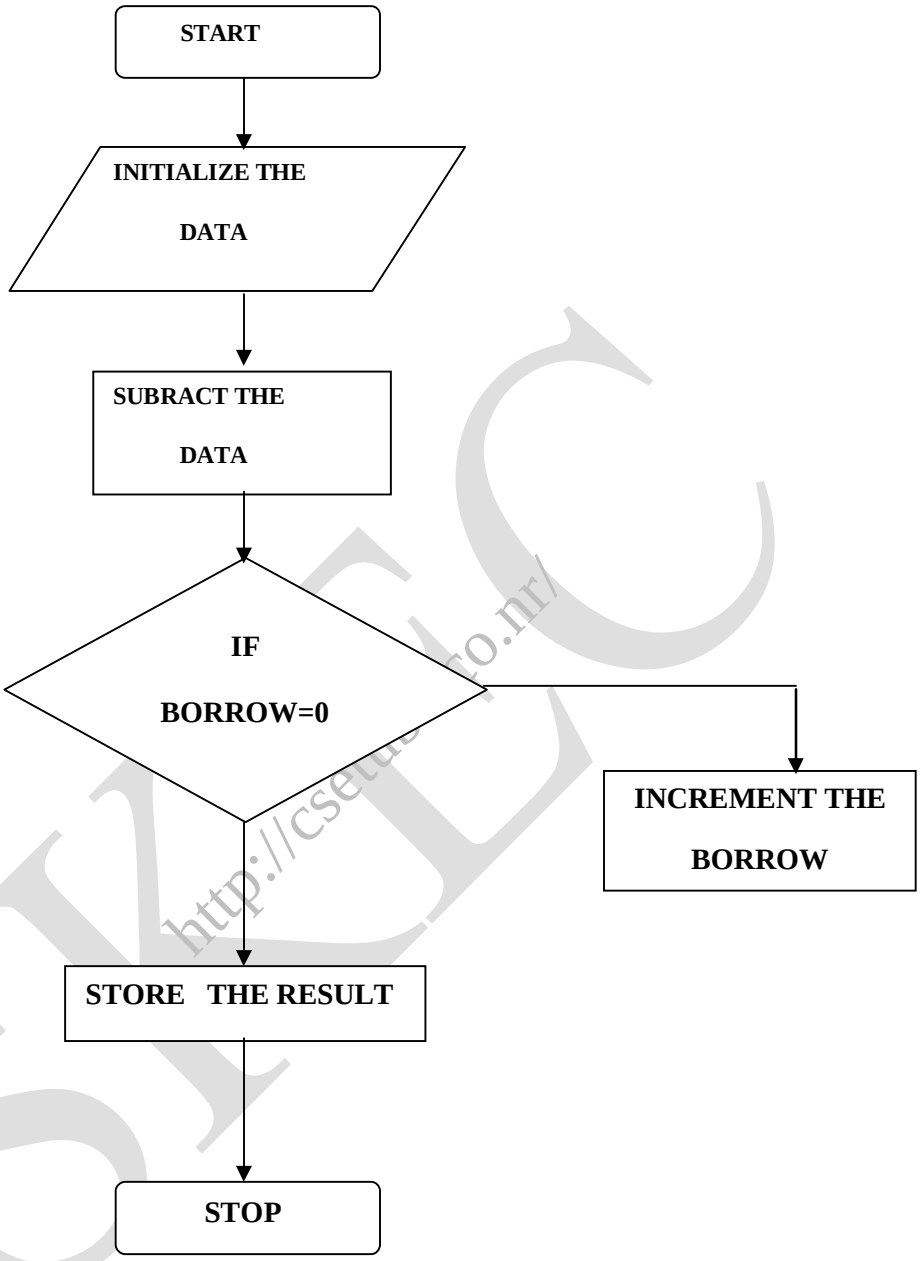
1. Start the program
2. Get the dividend and divisor
3. Initialize counter
4. Subtract divisor from dividend
5. If there is carry increment carry
6. Check the carry flag
7. Store the result
8. Halt the program

FLOWCHART:

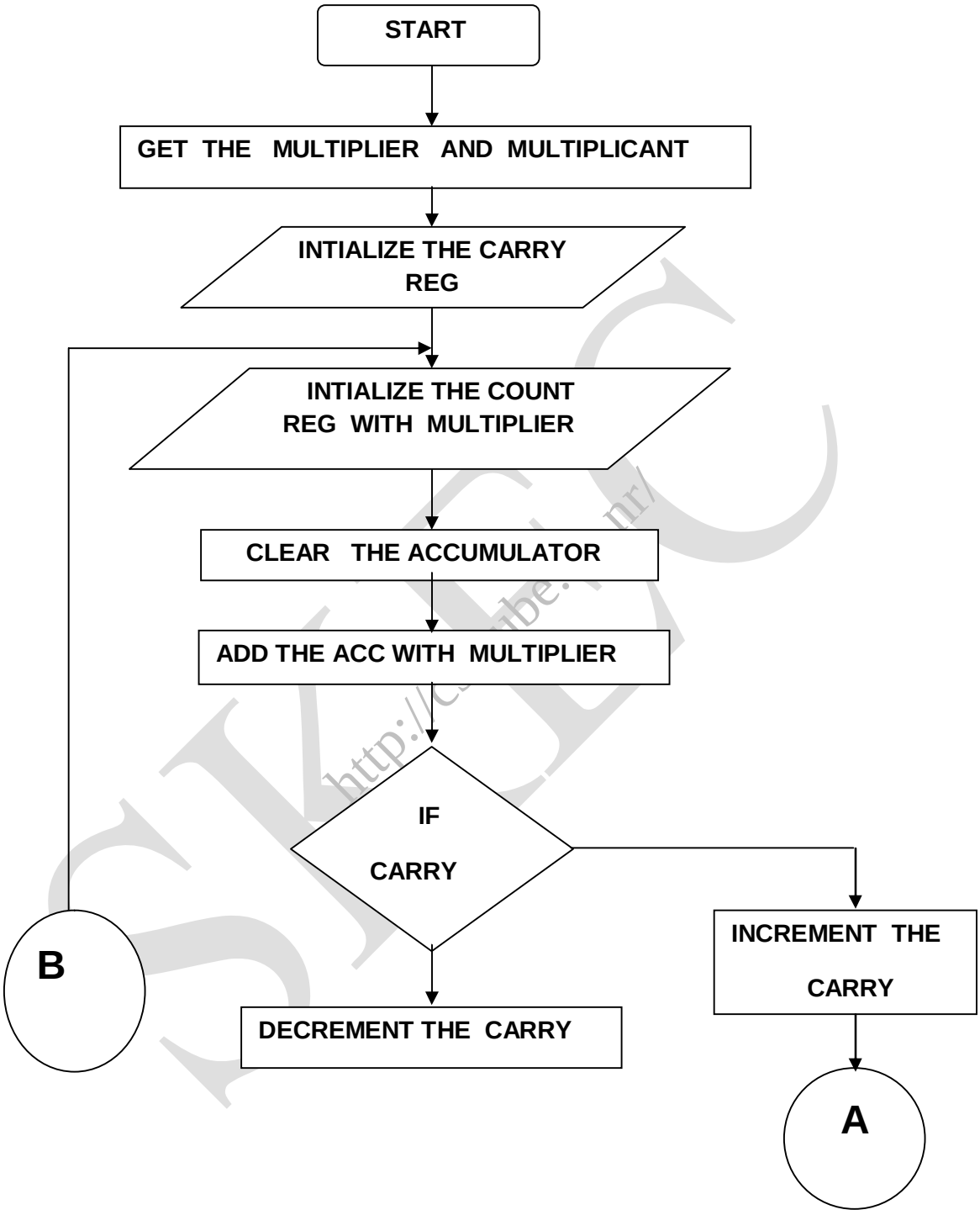
ADDITION:

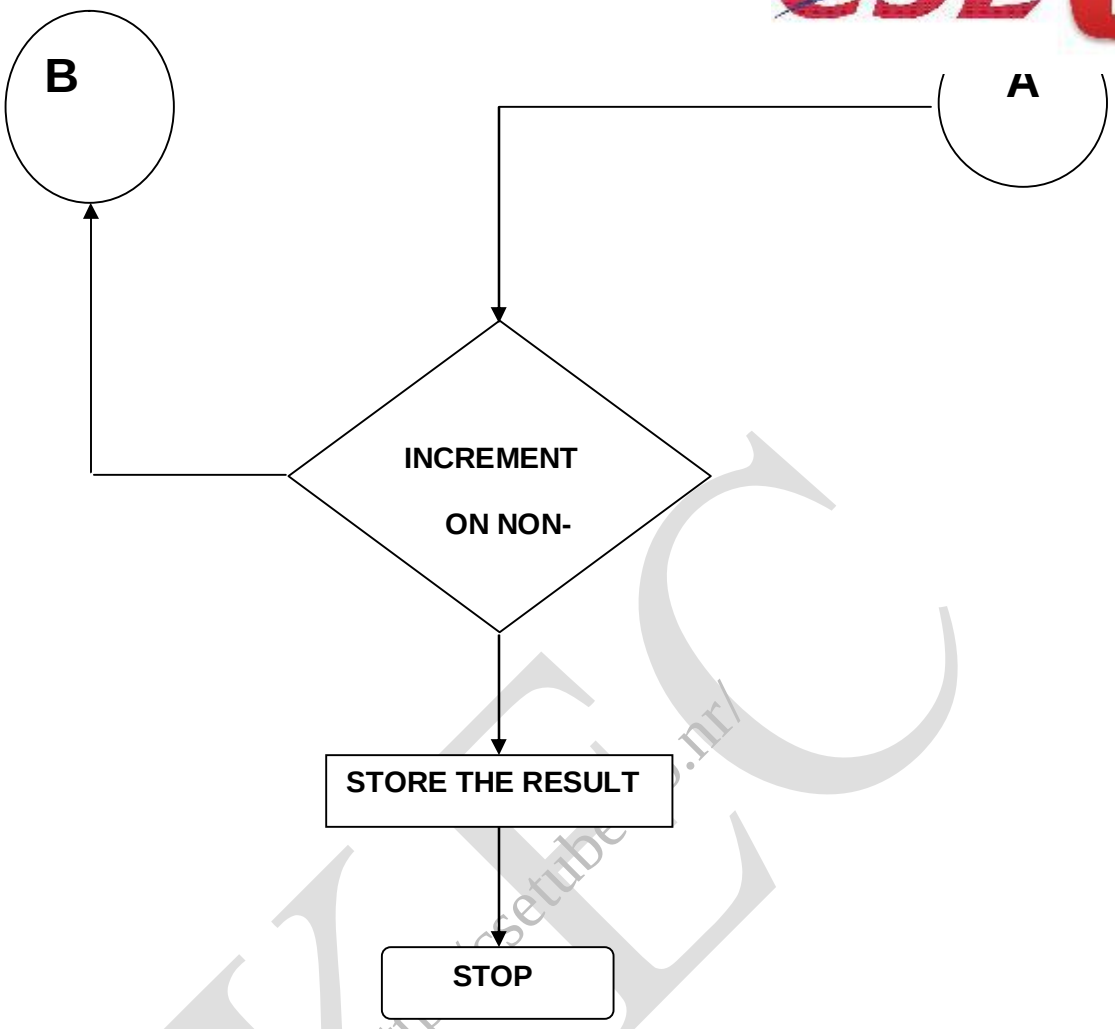


SUBTRACTION

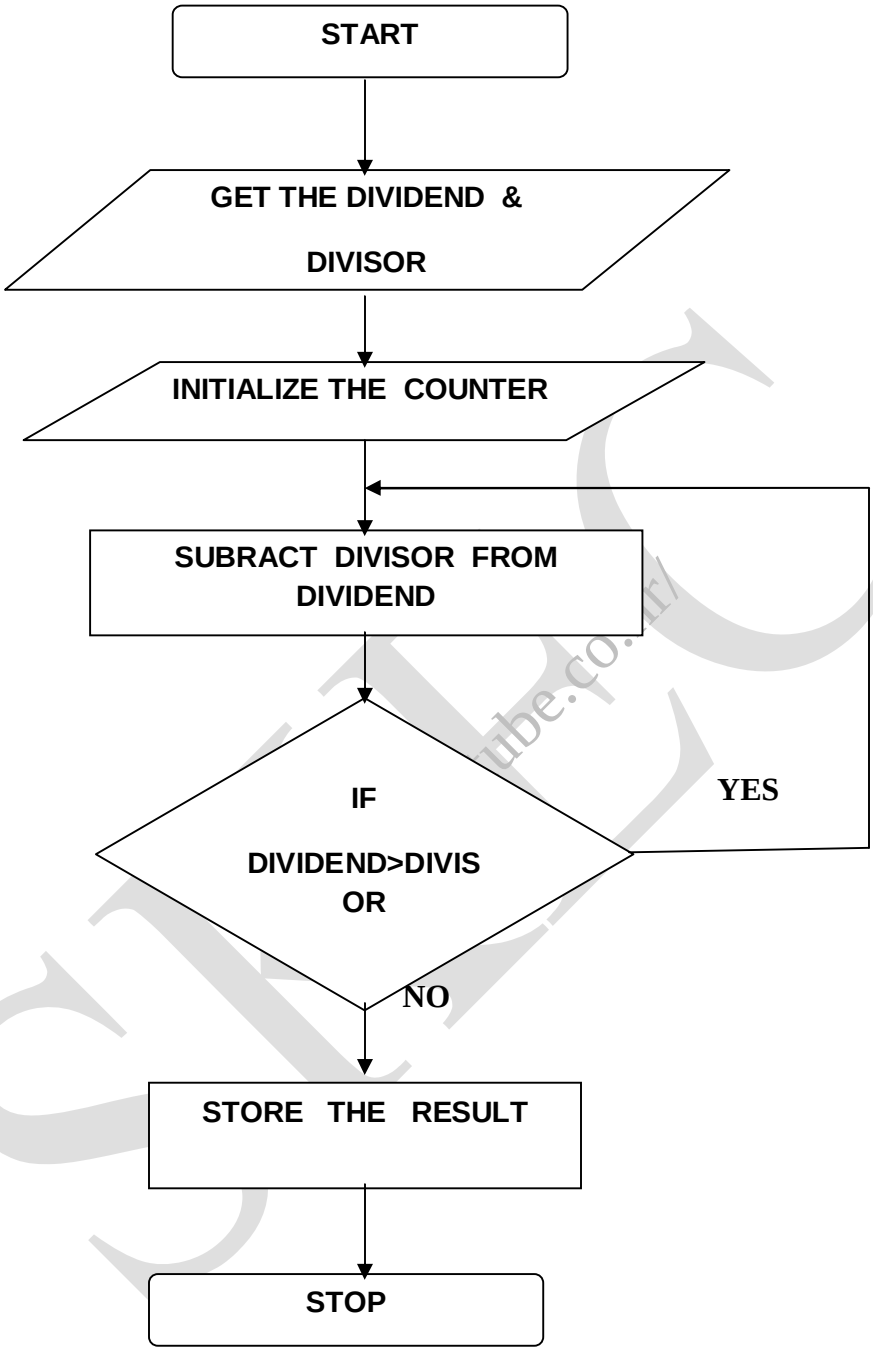


MULTIPLICATION





DIVISION



PROGRAM:

ADDITION

ADDRESS	LABEL	MNEMONICS	HEXCODE	COMMENTS
9100		LDA 9200	3A,00,92	loads the content of memory location to the accumulator
9103		MOV B,A	47	moves the accumulator content to b register
9104		LDA 9201	3A,01,92	loads the content of memory location to accumulator
9107		ADD B	80	add the B register content with accumulator
9108		MVI C,00	0E,00	initialize carry as 00
910A		JNC 910E	02,910E	checks the carry flag
910D		INR C	0C	increments carry flag
910E		STA 9600	32,00,96	stores the result
9111		MOV A,C	79	Moves the carry in accumulator
9112		STA 9601	32,01,96	Stores the carry in 9601
9115		RST 1	CF	Halt the program

SUBTRACTION

ADDRESS	LABEL	MNEMONICS	HEXCODE	COMMENTS
8100		LDA 8200	3A,00,32	Load the I value in accumulator
8103		MOV B,A	47	Move the content to the B register
8104		LDA 8201	3A,01,82	Load accumulator to the II data
8107		MVI C,00	0E,00	Initialize carry as 00
8109		SUB B	90	Content B reg from accumulator
810A		JNC 8110	02,10,81	Checks the carry flag
810D		CM A	2F	Complement of a
810E		INR A	3C	Increment the content of A register
810F		INR C	6C	Increment the content of c register
8110		STA 8600	32,00,86	Store the diff value
8113		MOV A,C	79	Move content of reg A to accumulator
8114		STA 8601	34,01,86	Store the borrow
8117		RST 1	CF	Terminate the program

MULTIPLICATION

ADDRESS	LABEL	MNEMONICS	HEXCODE	COMMENTS
9100		LDA 9200	3A,00,92	Load the multiplier
9103		MOV D,A	57	Move the content of accumulator to D reg
9104		LDA 9201	3A,01,92	Load the multiplier
9107		MOV B,A	47	Move the content of accumulator to B reg
9108		MVI C,00	0E,00	Move carry as 0
910A		XRA A	AF	Clear all
910B		ADD D	82	Add the content of D register
910C		JNC 9110	D2,02,01	If no carry jump to next
910F		INR C	OC	Increment the c register
9110		DCR B	05	Decrement the B register
9111		JNZ 910B	C2,DB,91	Store result in 9600
9114		STA 9600	32,00,96	Store the carry in 9601
9117		MOV A,C	79	Move the content of C to B register
9118		STA 9601	32,01,96	Store the result
911B		RST 1	CF	Stop the program

DIVISION

ADDRESS	LABEL	MNEMONICS	HEXCODE	COMMENTS
9100		MVI C,00	0E,00	Initialize the c register to 00
9102		LDA 9500	3A,00,95	Load the data divisor
9105		MOV B,A	47	Move the content of accumulator to B reg
9106		LDA 9501	3A,01,95	Get the dividend
9109		CMP B	B8	Compare B & C
910A		JC 9112	DA,12,91	Jump on carry
910D		SUB B	90	Subtract B from A
910E		INR C	0C	Increment the content of C register
910F		JMP 9109		
9112		STA 9600	32,00,96	Store the values of A
9115		MOV A,C	79	Move the content C to A
9116		STA 9601	32,01,96	Store the result
9119		RST 1	CF	Terminate the program

Output:

Addition

MEMORY ADDRESS	DATA-I	DATA-II	DATA-III
(Input)			
9201	F2	E1	D3
9200	F3	E2	D6
(Output)			
9600 (sum)	E5	C3	A9
9601(carry)	01	01	01

Subtraction

MEMORY ADDRESS	DATA-I	DATA-II	DATA-III
(Input)			
8200	07	04	0B
8201	03	06	0A
(Output)			
9600	04	02	01
9601	01	00	01

MULTIPLICATION

MEMORY ADDRESS	DATA-I	DATA-II	DATA-III
(Input)			
9200	06	07	0F
9201	02	04	0A
(Output)			
9600	0C	1C	96
9601	00	00	00

DIVISION

MEMORY ADDRESS	DATA-I	DATA-II	DATA-III
(Input)			
9500	04	04	05
9501	08	04	0A
(Output)			
9600	00	00	00
9601	02	01	02

RESULT:

Thus the Basic arithmetic operations have been performed successfully on 8085 microprocessor by Assembly Language Programming (ALP).

Ex. No: 02 **SEARCHING & SORTING**

Searching:

AIM

To search a number (largest, smallest) in the array using 8085

APPARATUS REQUIRED:

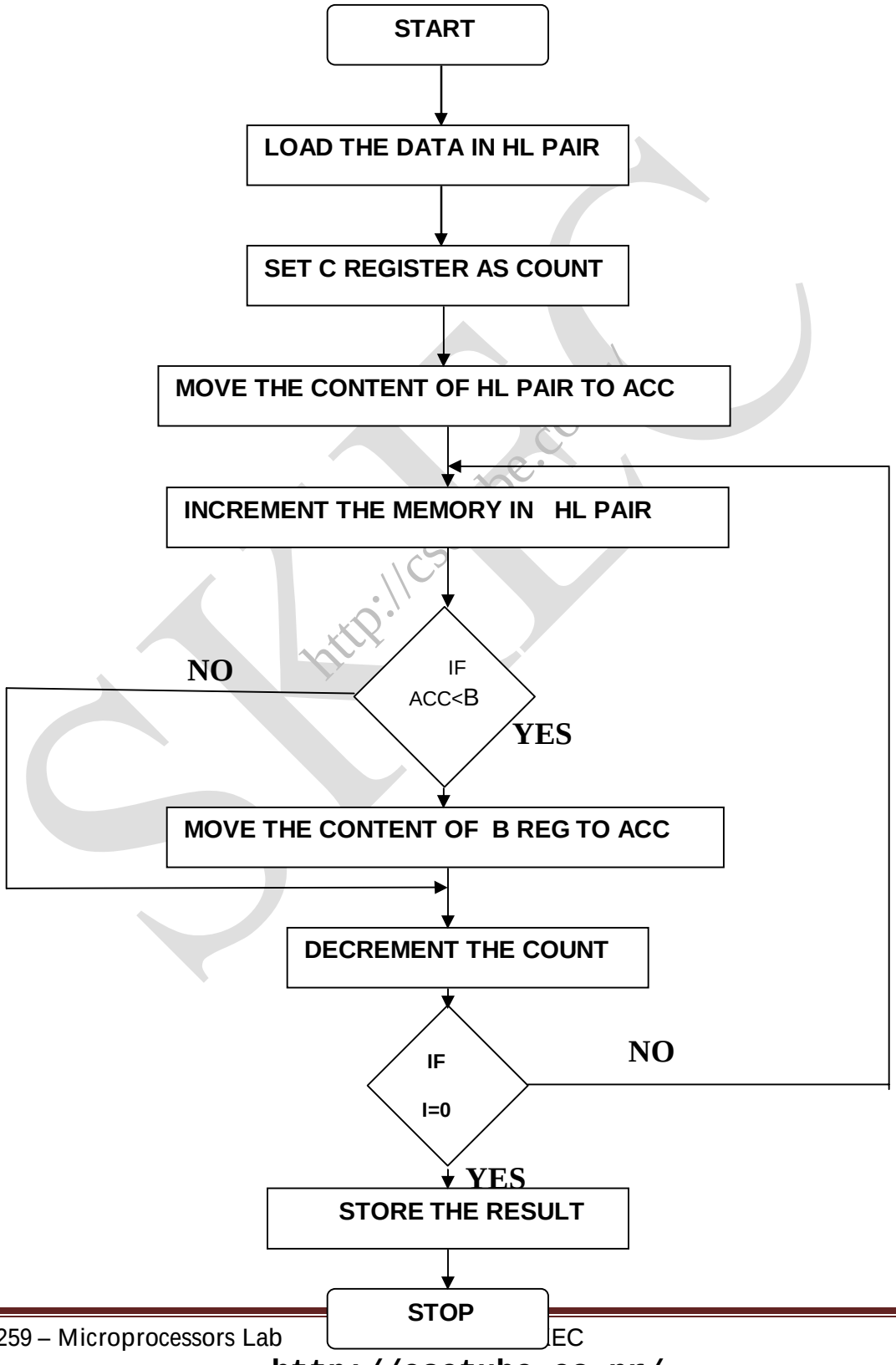
8085 Microprocessor Kit.

ALGORITHM

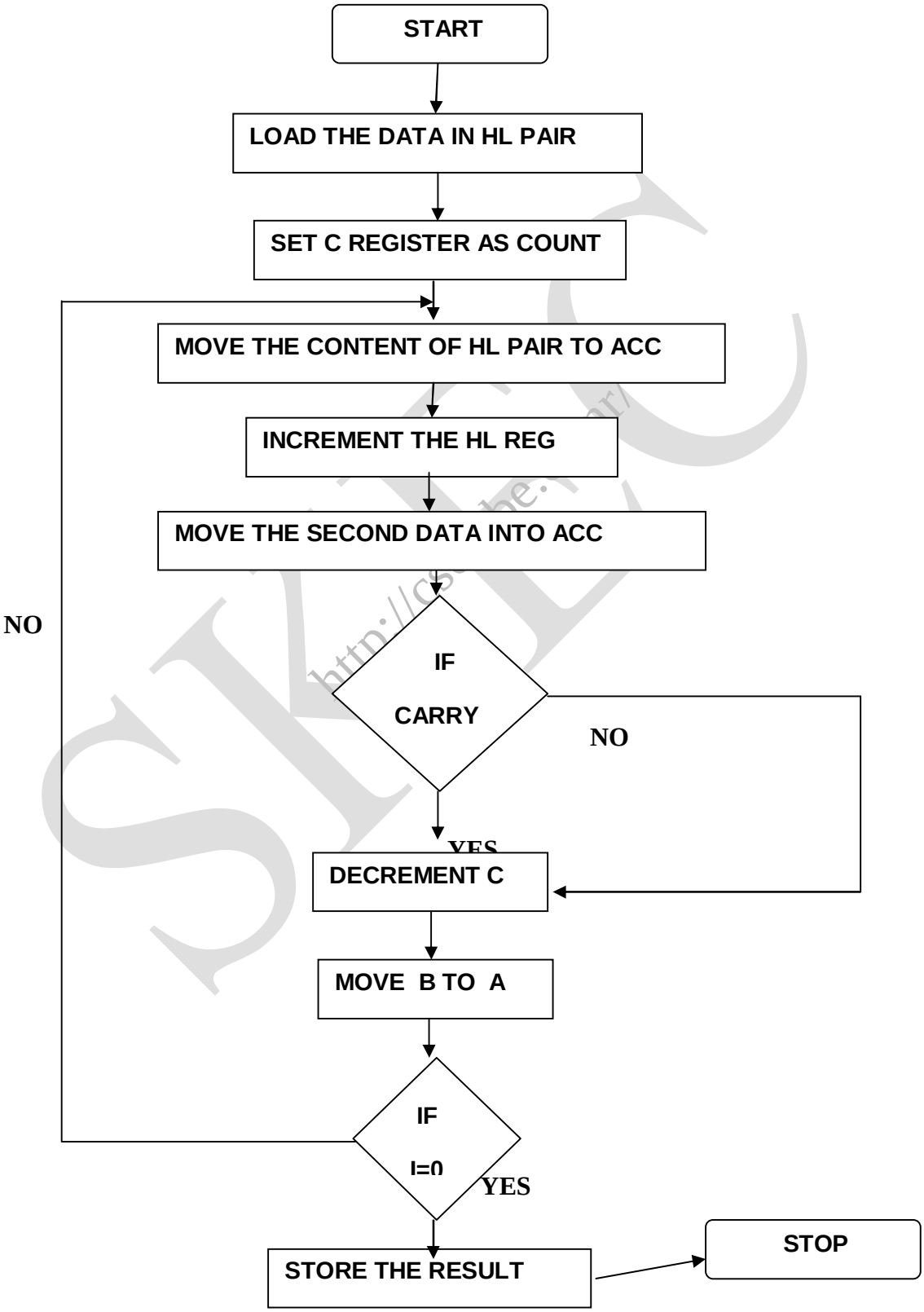
1. Get the number of elements
2. Get the starting address of an array
3. Move the first element into the accumulator
4. Compare the second element after moving into register
5. Check the carry flag
6. If carry/ no carry, exchange accumulator and register else go to next step
7. Decrement the count
8. Move the next element into the register
9. If not zero, then go to step 4
10. Else, store accumulator and stop the program.

FLOWCHART:

LARGEST NUMBER IN AN ARRAY



SMALLEST NUMBER IN AN ARRAY



PROGRAM:

ADDRESS	LABEL	MNEMONICS	HEXCODE	COMMENTS
9100		LXI H,9500	21,00,95	Load the data into memory
9103		MVI C,05	0E,05	
9105		MOV A,M	7E	Move the content A to accumulator
9106		INX H	23	Increment count
9107		MOV B,M	46	Move the content in to B
9108		CMP B	B8	Compare B
9109		JNC 910D (JC 910D)	D2,OD,91 (C2,0D,91)	Jump on no carry for Largest/ Jump on carry for smallest
910C		MOV A,B	78	Move the content B to A
910D		DCR C	00	Decrement C
910E		JNZ 9106	C2,06,91	Jump on 9106
9111		STA 9600	32,00,96	Store the result
9114		RST 1	CR	Stop the program

OUTPUT:

LARGEST NUMBER IN THE ARRAY

MEMORY ADDRESS	DATA-I	DATA-II	DATA-III
(Input)			
9500	09	01	09
9501	08	03	08
9502	07	04	07
9503	06	02	03
9504	05	06	01
9505	04	05	0A
(Output)			
9600	09	06	0A

SMALLEST NUMBER IN AN ARRAY

MEMORY ADDRESS	DATA-I	DATA-II	DATA-III
(Input)			
9500	09	0A	AA
9501	08	0B	BB
9502	07	0C	CC
9503	06	0D	DD
9504	05	0E	EE
9505	04	0F	FF
(Output)			
9600	04	0A	AA

SORTING :

AIM:

To perform Sorting of an array using 8085.

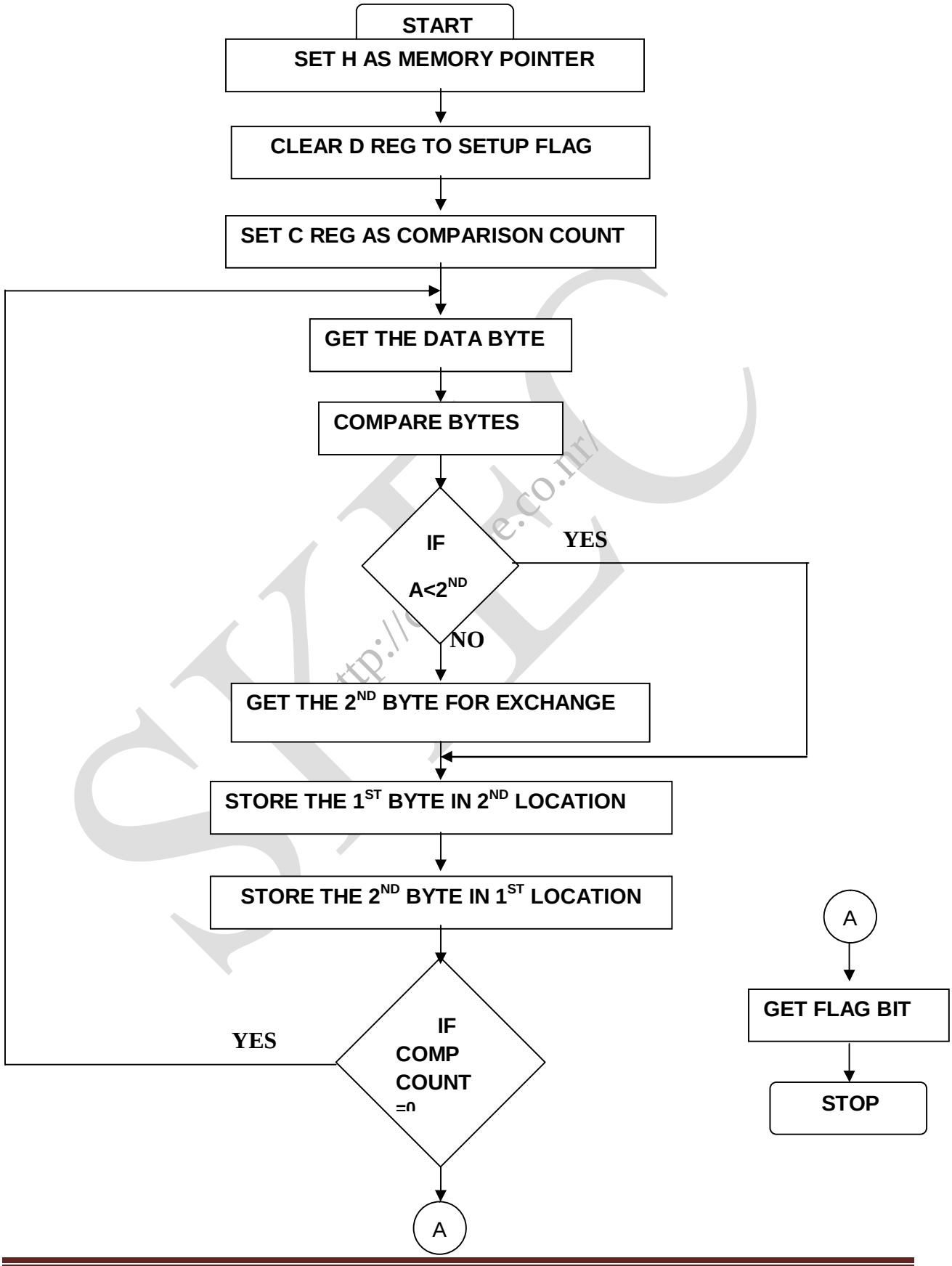
APPARATUS REQUIRED:

8085 Microprocessor Kit.

ALGORITHM:

1. Start the program
2. Set as memory register
3. Clear D register to setup flag
4. Set C register as comparison
5. Compare the bytes
6. If A<2nd byte, go to 9 decrement comparison
7. Store the I byte in II location
8. Store II byte in I location
9. Load 1 in D as remainder to exchange
10. Decrement comparison count
11. If comparison count is not equal to 0, go to 4 else get flag.
12. Place flag fit do in carry
13. If flag is 1, go to 1
14. Else, end of sorting
15. Stop the program

FLOWCHART:



PROGRAM:

ADDRESS	LABEL	MNEMONICS	HEXCODE	COMMENTS
9100		LXI H,9200	21,00,92	Set memory pointer
9103		MVI D,00	16,00	Clear D reg to set flag
9105		MVI C,04	DE,04	Set C reg for comparison count
9107		MOV A,M	4E	Get the data byte
9108		INX H	23	Point to next byte
9109		CMP M	BE	Compare byte
910A		JC 9114 (JNC 9114)	DA,14,91	If A<2nd byte don't exchange
910D		MOV B,M	46	Get 2nd byte for exchange
910E		MOV M,A	77	Store I byte in II location
910F		DCX H	28	Get ready to next comparison
9110		MOV M,B	70	Store II byte in II location
9111		INX H	23	Get ready for next comparison
9112		MVI D,01	1601	Load in D remainder for exchange
9114		DCR C	0D	Decrement comparison count if comparison 10,80 Get flag hit in A
9115		JNZ 9107	C2,07,91	Place flag hit do carry

9118		MOV A,D	7A	
9119		RRC	0F	If carry, exchange occurs
911A		JC 9100	DA,00,91	End of string
911D		RST 1	CF	

Output:

ASCENDING ORDER

MEMORY ADDRESS	DATA-I	DATA-II
(Input)		
9200	09	0D
9201	02	0A
9202	04	0C
9203	06	0B
9204	05	0F
(Output)		
9200	02	0A
9201	04	0B
9202	05	0C
9203	06	0D
9204	09	0F

DECENDING ORDER

MEMORY ADDRESS	DATA-I	DATA-II
(Output)		
9200	09	0F
9201	06	0D
9202	05	0C
9203	04	0B
9204	02	0A

RESULT:

Thus the SEARCHING & SORTING of an array has been performed using 8085 Microprocessor.

Ex.No: 03 **CODE CONVERSION**

AIM:

To perform code conversions like HEX to ASCII , Binary to BCD and Vice versa.

APPARATUS REQUIRED:

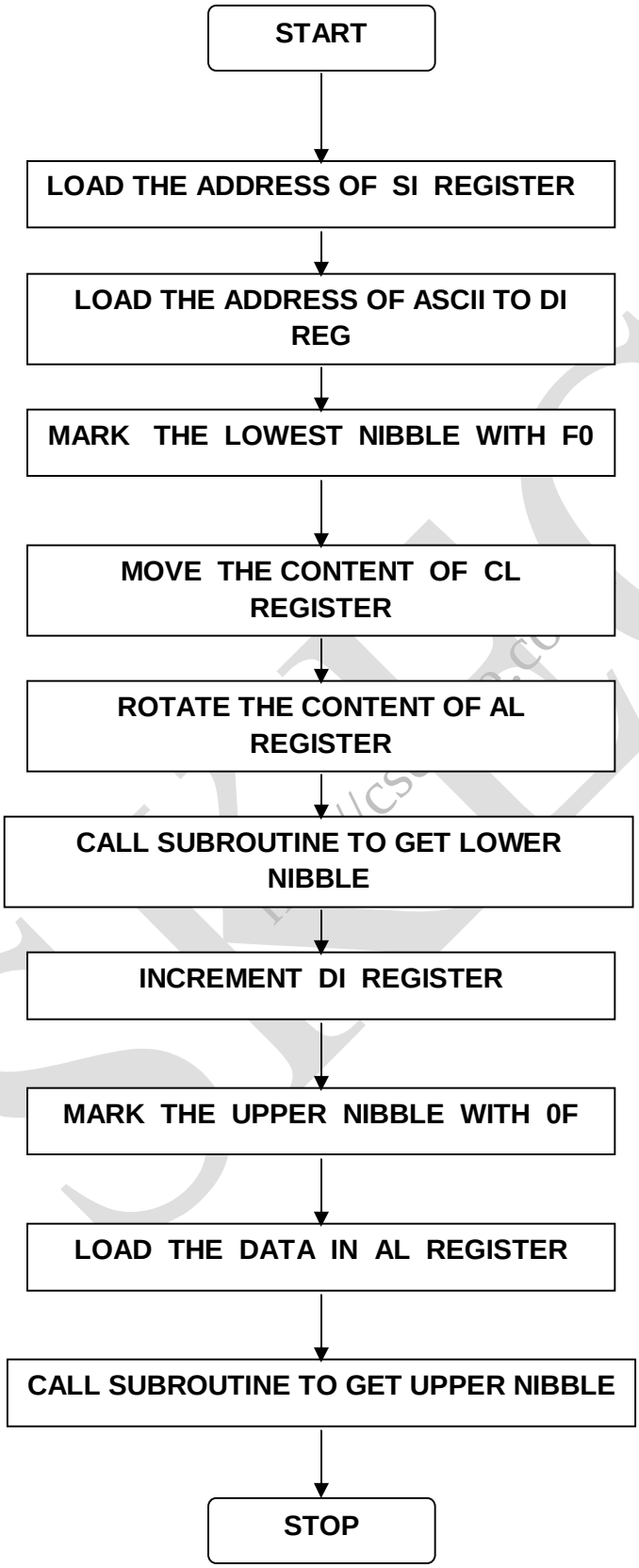
8085 Microprocessor Kit.

a.)HEXADECIMAL TO ASCII

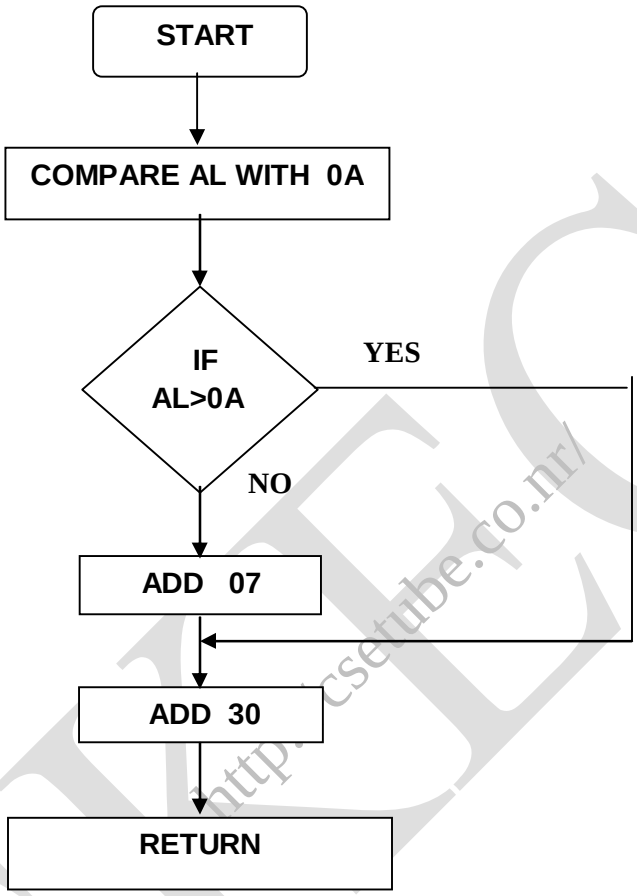
ALGORITHM:

1. Set the SI pointer for binary array
2. Set the DI pointer for ASCII
3. Mark the lower bit data with F0
4. Move me count to CL reg.
5. Rotate the lower nibble to upper nibble
6. Call the subroutine to compare with DA
7. If less than 10, then jump to next
8. Add with DA, with AL reg
9. If greater than 10, then add to 10H else return to 6
10. Store the data in D reg
11. Stop the program.

FLOWCHART:



SUB ROUTINE



PROGRAM:

ADDRESS	LABEL	MNEMONICS	HEXCODE	COMMENTS
9100		LXIH, 9000	01,00,90	Load address
9103		LXI D, 9200	11,00,92	Load data
9106		MOV A,M	7E	Mark the higher bit
9107		ANI, 0F	E6,0F	Add of
9109		CALL ASCII	CD	Call ASCII function
910C		MOV C,A	4F	Move A to C reg
910D		MOV A,B	78	Move B reg to A
910E		ANI F0	E6,F0	AND F0
9100		RLC	07	Rotate left without carry
9111		RLC	07	Rotate left without carry
9112		RLC	07	Rotate left without carry
9113		RLC	07	Rotate left without carry
9114		CALL ASCII	CD	Call the ASCII function
9117		STA 9201	32,01,92	Store the result
911A		MOV A,C	79	Move C to A

911B		STA 9202	32,02,192	Store the result
911E		RST 1	CF	Break point
911F	ASCII	CPI 09	FE,09	Compare with 09
9121		JNC Next		Jump no carry
9124		ADI 07	C6,07	Add immediately
9126	Next	ADI 30	C6,30	
9128		RET	C9	Return to main program

Output:

MEMORY ADDRESS	DATA-I	DATA-II
(Input) 9200	05	AC
(Output) 9201 9202	30 35	41 43

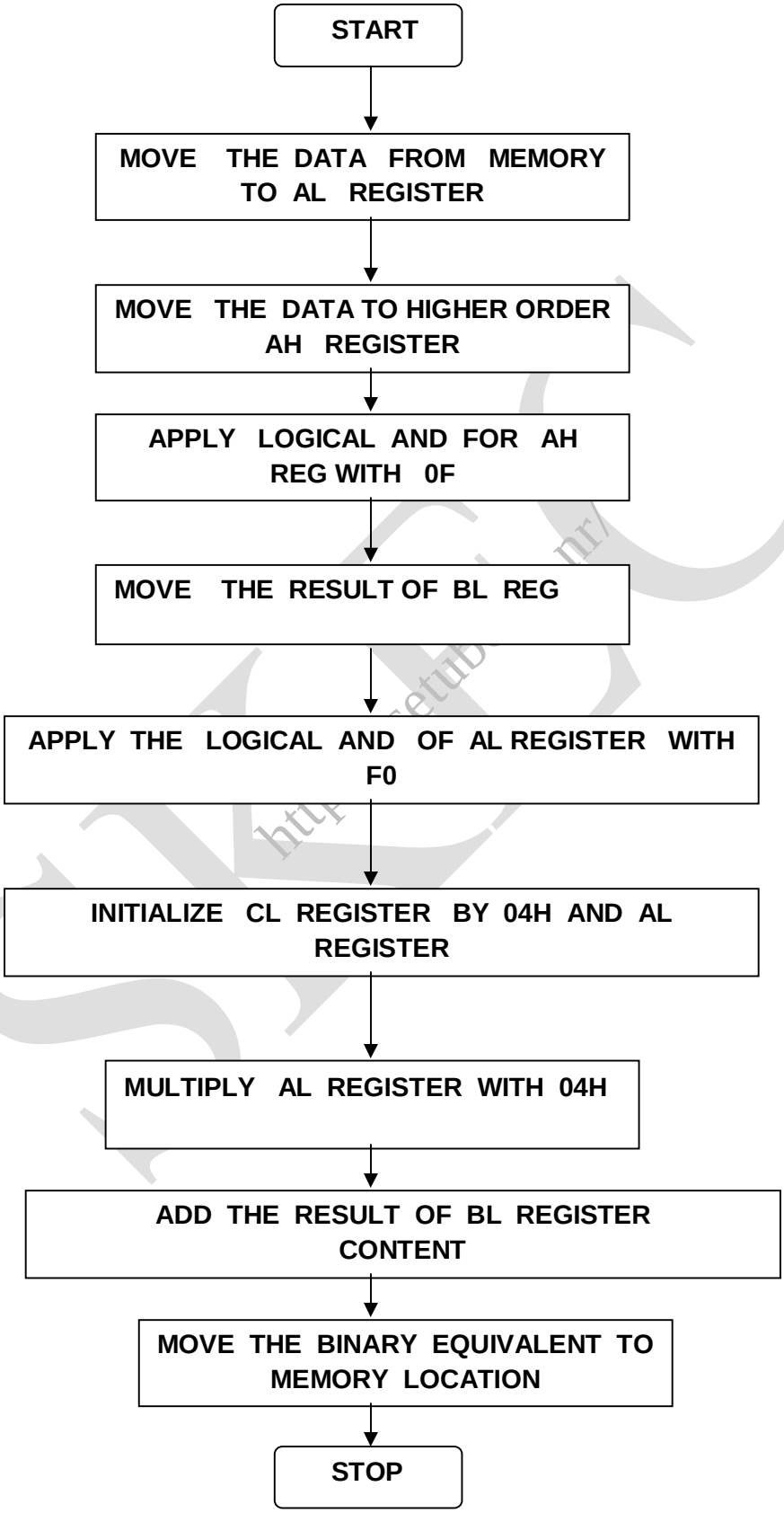
\

b.) BCD TO BINARY CODE CONVERSION

ALGORITHM:

1. Start the program
2. Move the data for memory location to reg AL
3. Move the data for memory location to reg AL
4. Apply the logical AND of AH reg content with 0F
5. Move the result of logical operation to BL.
6. Apply the logical AND of AL with F0H
7. Initialize the lower order C reg of 04H and rotate the AL reg content
8. Multiply the AL reg content with 04H
9. Add the result with BL reg content
10. Move the equivalent binary from AL reg content to memory location
11. Halt the program

FLOWCHART:



PROGRAM:

ADDRESS	LABEL	MNEMONICS	HEXCODE	COMMENTS
1000		MOV AL,[1200]	A0,00,12	Move the id to 1 reg
1003		MOV AH,AL	88,C4	Move data to reg A4
1005		AND AH,0F	80,EA,DF	Mark lower byte
1008		MOV BL,AH	888,E3	Move the result if marking
100A		AND AL,F0	24,F0	Mark the higher byte
100C		MOV CL,04	B1,DH	Move the CL by 04
100E		ROL AL,CL	D2,CD	Mark the higher byte
1010		MOV AH,0A	34,DA	Moves the data
1012		MUL AH	F6,E7	Multiply
1014		ADD AL,BL	C0,D8	Add the data
1016		MOV [1600],AL	A2,00,16	Move the binary equivalent memory
1019		INT 3	CC	Halt the program

Output:

MEMORY ADDRESS	DATA-I	DATA-II
(Input) 1200	0A	1D
(Output) 1600	0A	17

RESULT:

Thus the code conversion has been performed .

ADDITION OF TWO MATRIX

AIM:

To write a program for addition of two matrix by using 8086.

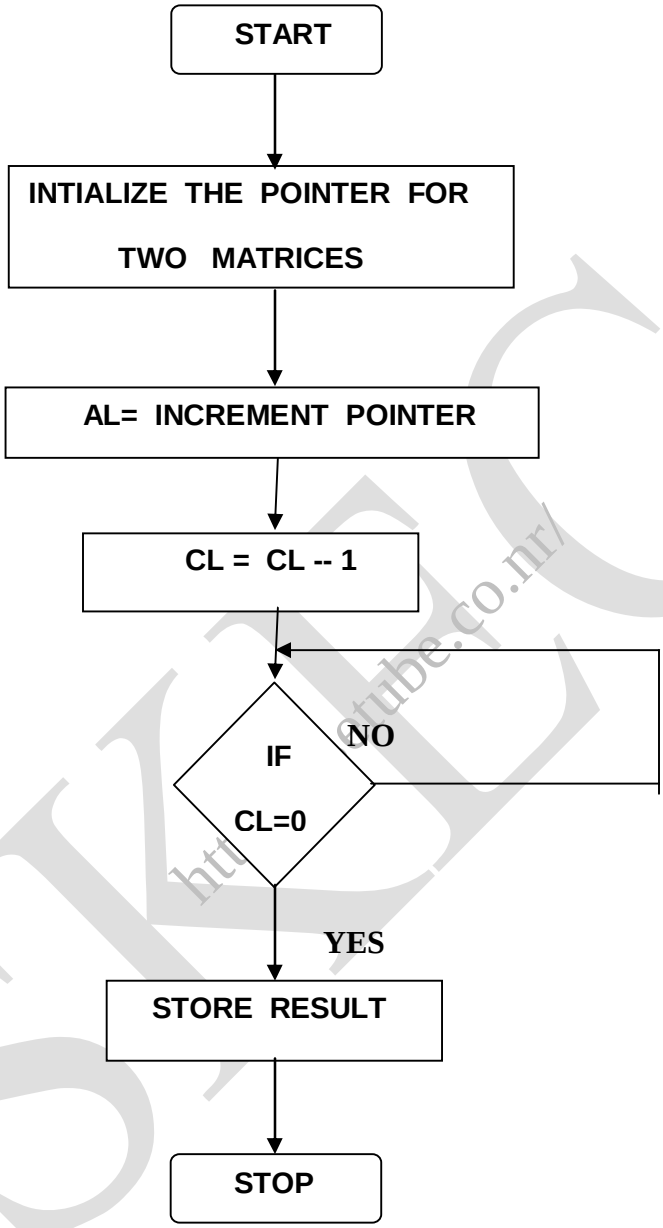
APPARATUS REQUIRED:

8086 Microprocessor Kit

ALGORITH:

1. Initialize the pointer only for data and result
2. Load AL with count
3. Add two matrix by each element
4. Process continues until CL is zero
5. Store result.

FLOWCHART:



PROGRAM:

ADDRESS	LABEL	MNEMONICS	HEXCODE	COMMENTS
1500		MOV CL,09	BL,09	Count for 3×3 matrix
1502		MOV SI,2000	BE,00,20	Address in SI
1505		MOV DI,300	BF,00,30	Address in DI
1508		MOV AL,[SI]	BA,04	Load AL with matrix
150A		MOV BL,[DI]	8A,10	Load BL with matrix
150C		ADD AL,BL	00,08	Add two data
150E		MOV [DI],AL	88,05	Store result
1510		INC DI	4F	Increment DI
1511		INC SI	46	Increment SI
1512		DEC CL	FE,C9	Decrement CL
1514		JNZ 1508	75,F2	Loop continues until zero
1516		INT 3	CC	Break point

Output:

INPUT		OUTPUT
2000-02	3000-02	3000-04
2001-02	3001-02	3001-04
2002-02	3002-02	3002-04
2003-02	3003-02	3003-04
2004-02	3004-02	3004-04
2005-02	3005-02	3005-04
2006-02	3006-02	3006-04
2007-02	3007-02	3007-04
2008-02	3008-02	3008-04

RESULT:

Thus the output for the addition for two matrix was executed successfully.

SEARCHING FOR NUMBER IN AN ARRAY

AIM:

To write a program to search a number in a given array using 8086 microprocessor.

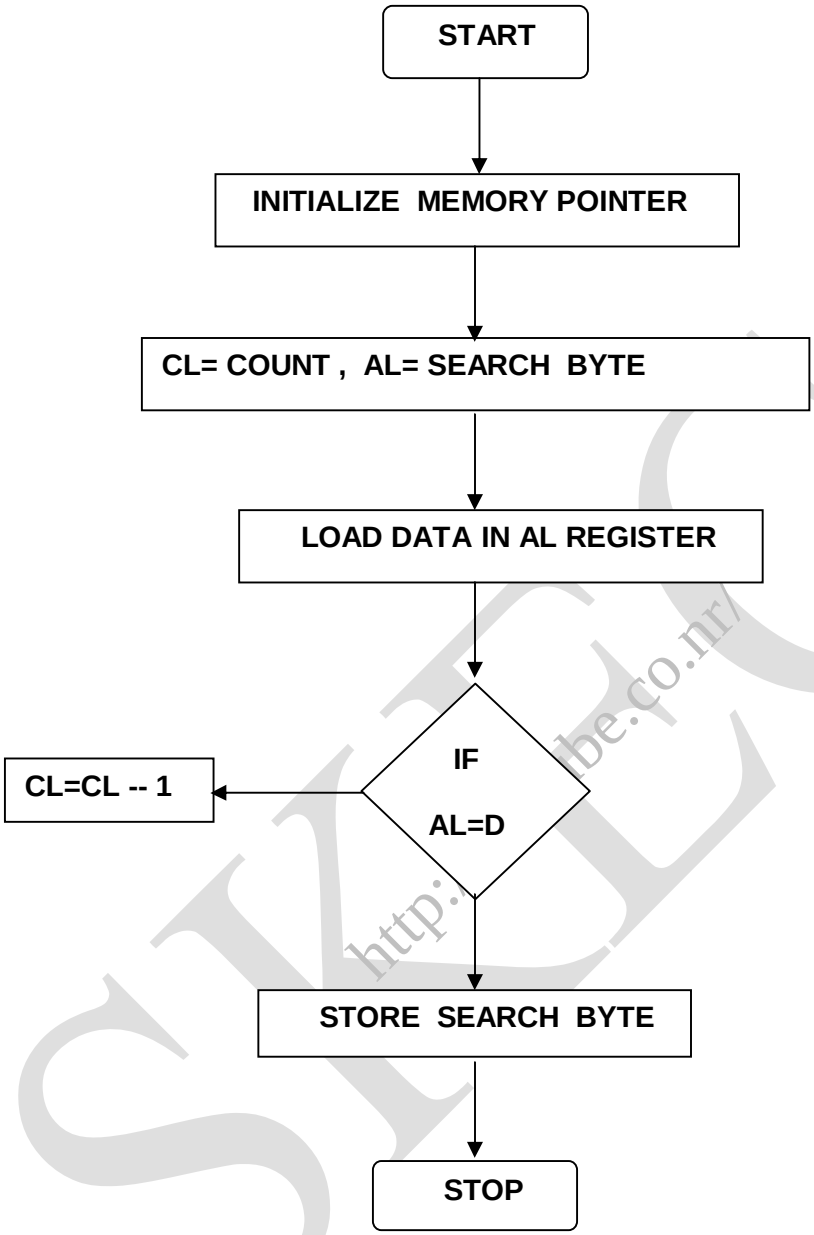
APPARATUS REQUIRED:

8086 Microprocessor Kit.

ALOGORITHM:

1. Initialize the counter to the memory for staring the data and result.
2. Load DI with search byte
3. Load CL with count
4. Load AC with data from memory
5. Compare AC with DL if its equal
6. Store result else go to 2
7. Store the result
8. Stop the program.

FLOWCHART:



PROGRAM:

ADDRESS	LABEL	MNEMONICS	HEXCODE	COMMENTS
1000	START	MOV SI,1100	BE,00,11	Set SI reg for array
1003		MOV DI,1200	BE,00,12	Load address of data to be searched
1006		MOV DI,[DL]	8A,15	Get the data to search in DL reg
1008		MOV BL,01	B3,DL	Set BL reg as want
100A		MOV AL,[SI]	8A,04	Get first element
100C	AGAIN	CMP AL,DL	35,00	Compare an element of array
100E		JZ AVAIL	74,15	If data are equal then jump to avail
1010		INC SI	46	Increment SI
1011		INC BL	FE,03	Increment BL count
1013		MOV AL,[SI]	8A,04	
1015		CMP AL,20	3C,20	Check for array
1017		JNZ AGAIN	72,F3	If not JZ to again
1019	NOT AVAIL	MOV CX,0000	B9,00,00	Initialize CX to zero
101C		MOV [DI+1],CX	89,4D,01	
101F		MOV [DI+3],CX	89,4D,03	

1022		JMP 102F	EB,0F,110	
1024	AVAIL	MOV BH,FF	B7,FF	
1026		MOV [DI+1],BH	88,7D,01	Store FF to result
1029		MOV [DI+2],BL	88,5D,02	Availability of data
102C		MOV [DI+3],SI	89,75,03	Store the address of data
102F		INT 3	CC	Stop the program

Output:

MEMORY ADDRESS	DATA-I	DATA-II
(Input)		
1100	05	05
1101	05	04
1102	03	09
1103	04	02
1104	07	03
1105	08	01
(Output)		
1200	05	02
1201	FF	FF
1202	01	04
1203	00	03
1204	11	11

RESULT:

Thus the output for searching an element in array using 8086 microprocessor was executed successfully.

Ex. No: 4 **16 BIT ARITHMETIC OPERATIONS & SORTING**

a) ADDITION

ALGORITHM:

I. Initialize the SI register to input data memory location
II. Initialize the DI register to output data memory location
III. Initialize the CL register to zero for carry
IV. Get the 1st data into accumulator.
V. Add the accumulator with 2nd data
VI. Check the carry flag, if not skip next line
VII. Increment carry(CL Reg)
VIII. Move the result from accumulator to memory.
IX. Also store carry register
X. Halt

Program:

Address	Label	Mnemonics	Machine Code	Comments
		MOV SI, 2000H **MOV DI, 3000H** **MOV CL, 00H** **MOV AX, [SI]** **ADD AX, [SI+2]** **JNC STORE** **INC CL** **STORE: MOV [DI], AX** **MOV [DI+2], CL** **INT 3**		

OUTPUT VERIFIACTION:

INPUT	DATA 1	DATA 2	DATA 2
2000H	34		
2001H	12		
2002H	78		
2003H	56		
OUTPUT			
3000H	AC		
3001H	68		
3002H	00		

b) SUBTRACTION

ALGORITHM:

I. Initialize the SI register to input data memory location
II. Initialize the DI register to output data memory location
III. Initialize the CL register to zero for borrow
IV. Get the 1st data into accumulator.
V. Subtract the accumulator with 2nd data
VI. Check the carry flag, if not set skip next line
VII. Increment carry(CL Reg)
VIII. 2's Compliment Accumalator
IX. Move the result from accumulator to memory.
X. Also store carry register
XI. Halt

Program:

Address	Label	Mnemonics	Machine Code	Comments
		MOV SI, 2000H **MOV DI, 3000H** **MOV CL, 00H** **MOV AX, [SI]** **SUB AX, [SI+2]** **JNC STORE** **INC CL** **NEG AX**		
	STORE:	**MOV [DI], AX** **MOV [DI+2], CL** **INT 3**		

OUTPUT VERIFIACTION:

INPUT	DATA 1	DATA 2	DATA 2
2000H	88		
2001H	44		
2002H	33		
2003H	22		
OUTPUT			
3000H	55		
3001H	22		
3002H	00		

C) MULTIPLICATION

ALGORITHM:

I. GET MULTIPLIER INTO ACCUMULATOR FROM MEMORY
II. GET MULTIPLICAND INTO BX REGISTER
III. MULTIPLY AX AND BX
IV. STORE LOWER ORDER WORD FORM ACCUMULATOR INTO MEMORY
V. STORE HIGHER ORDER WORD FROM DX INTO MEMORY
VI. HALT

Program:

Address	Label	Mnemonics	Machine Code	Comments
		MOV AX, [2000H] **MOV BX, [2002H]** **MUL BX** **MOV [3000], AX** **MOV [3002], DX** **INT 3**		

OUTPUT VERIFIACTION:

INPUT	DATA 1	DATA 2	DATA 2
2000H	0A		
2001H	00		
2002H	0A		
2003H	00		
OUTPUT			
3000H	64		
3001H	00		
3002H	00		
3003H	00		

D)DIVISION

I. GET DIVIDEND INTO ACCUMULATOR FROM MEMORY
II. GET DIVISOR INTO BX REGISTER
III. DIVIDE AX BY BX
IV. STORE QUOTIENT FORM ACCUMULATOR INTO MEMORY
V. STORE REMAINDER FROM DX INTO MEMORY
VI. HALT

Program:

Address	Label	Mnemonics	Machine Code	Comments
		MOV AX, [2000H] **MOV BX, [2002H]** **DIV BX** **MOV [3000], AX** **MOV [3002], DX** **INT 3**		

OUTPUT VERIFIACTION:

INPUT	DATA 1	DATA 2	DATA 2
2000H	68		
2001H	00		
2002H	0A		
2003H	00		
OUTPUT			
3000H	0A		
3001H	00		
3002H	04		
3003H	00		

To Search for a BYTE in an array.

Algorithm:

i. Load starting address of array in to SI reg. and initialize DI reg for result.
ii. Load the **byte** to be searched in DL reg. & initialize BH reg for position as 01
iii. Get the 1st element into Accumulator
iv. AGAIN : Compare Accumulator with DL reg
v. Check zero flag, if set goto AVAIL
vi. Update SI and Increment BL
vii. Get next element into Accumulator and compare with EOA
viii. If not zero, goto AGAIN.
ix. Initialize CX reg to zero
x. Store result for Not Available and goto END
xi. AVAIL: Get word for available in to BL reg.
xii. Store position, address and status of serach.
xiii. END: Halt

Program:

Address	Label	Mnemonics	Machine Code	Comments
		MOV SI, 2000H		
		MOV DI, 3000H		
		MOV DL, [DI]		
		MOV BH, 01		
		MOV AL, [SI]		
	AGAIN:	**CMP AL, DL**		
		JZ AVAIL		
		INC SI		

		INC BL		
		MOV AL, [SI]		
		CMP AL, *E0*		
		JNZ *AGAIN*		
	NOTAVL	**MOV CX, 0000H**		
		MOV [DI+1], CX		
		MOV [DI+3], CX		
		JMP *END*		
	AVAIL:	**MOV BL, FF**		
		MOV [DI+1], BX		
		MOV [DI+3], SI		
	END:	**INT 3**		

OUTPUT VERIFIACTION:

INPUT	DATA 1	DATA 2	DATA 2
3000H(BYTE)	**05**		
2000H	AA		
2001H	BB		
2002H	CC		
2003H	55		
2004H	77		

‘	‘		
‘	‘		
‘	‘		
‘	‘		
2018H	05		
‘	‘		
‘	‘		
‘	‘		
2032H	EO		
OUTPUT			
3000H	05		
3001H	FF		
3002H	19		
3003H	18		
3004H	20		

Ascending/Descending order

Algorithm:

i. Load SI reg with pointer to array
ii. Load array length to CL & CH for two counters (CL for repetitions & CH for comparisons)
iii. REPEAT : Get an element into accumulator
iv. NEXT: Compare with next element
v. Check carry flag, if set goto SKIP
vi. Swap elements of array
vii. SKIP: Decrement CH and if not zero go to NEXT
viii. Decrement CL , if not zero go to REPEAT
ix. Halt

Program:

Address	Label	Mnemonics	Machine Code	Comments
		MOV SI, 1500H		
		MOV CL, [SI]		
		DEC CL		
	REPEAT:	**MOV SI, 1500H**		
		MOV CH, [SI]		
		DEC CH		
		INC SI		
	NEXT:	**MOV AL, [SI]**		
		INC SI		
		CMP AL, [SI]		
		JC *SKIP/JNC SKIP*		
		XCHG AL, [SI]		
		XCHG AL, [SI - 1]		
	SKIP:	**DEC CH**		

		JNZ *NEXT* **DEC CL** **JNZ *REPEAT*** **INT 3**		

OUTPUT VERIFIACTION:

INPUT	DATA 1	DATA 2	DATA 2
1500(COUNT)	**07**		
1501	AA		
1502	BB		
1503	CC		
1504	55		
1505	77		
1506	11		
1507	99		
OUTPUT			
1501	11		
1502	55		
1503	77		

1504	99		
1505	AA		
1506	BB		
1507	CC		

Ex.No: 05 **STRING MANIPULATION**

AIM:

To perform string manipulation in 8086 microprocessor.

APPARATUS REQUIRED:

8086 Microprocessor Kit.

a) Block Transfer

Algorithm:

i. Initialize DS & ES registers
ii. Load source and destination locations to index regs.
iii. Load block size in counter reg.
iv. Clear Direction flag for Auto- incrementing mode
v. Copy contents from source to destination address until counter becomes zero.
vi. Halt.

Program:

Address	Label	Mnemonics	Machine Code	Comments
		MOV AX, 0000H **MOV DS, AX** **MOV ES, AX** **MOV SI, 2001H** **MOV DI, 3000H** **MOV CX ,[2000H]** **CLD** ***TRANSFER:*** **MOVSB** **LOOP** ***TRANSFER*** **INT 3**		

OUTPUT VERIFIACTION:

INPUT	DATA 1	DATA 2	DATA 2
2000H (Block Size)	05		
2001H	12		
2002H	34		
2003H	56		
2004H	78		
2005H	9A		

OUTPUT			
3000H	12		
3001H	34		
3002H	56		
3003H	78		
3004H	9A		

b) Table search

Algorithm:

i. Initialize BX reg with table size
ii. Get address of table and symbol in DI & SI reg.
iii. Clear direction flag.
iv. Load counter with 08(length)
v. Compare bytes if successful goto FOUND
vi. Point to the next symbol
vii. Decrement BX, if not zero repeat the steps
viii. Show not found
ix. FOUND: Show found status.
x. Halt

Program:

Address	Label	Mnemonics	Machine Code	Comments
		MOV BX, Table Size		
		LES DI, [3000H]		
		MOV DX, DI		
		LDS SI, [2000H]		
		CLD		
	LOOP1	**MOV CX , 08**		
		REPE CMPSB		
		JE FOUND		
		ADD DX, 0020H		
		MOV DI, DX		
		MOV SI, [SI+8]		
		DEC BX		
	NOTFOUND:	**JNE LOOP1**		
		MOV AL, 00H		
		MOV [3000H], AL		
	FOUND:	**JMP *END***		
		MOV AH, FFH		
	END:	**MOV [3000H], AH**		
		INT 3		

OUTPUT VERIFIACTION:

INPUT	
3000H	00
3001H	30
3002H	00
3003	00
2000H	00
2001H	12
2002H	00
2003H	00

Note: The table has to be loaded from memory location 3000H & the search string in 1200H. Table size can be any 16 bit number

c) Code Conversion

Algorithm

i. Initialize BX reg with address of translation table
ii. Load source and destination address in SI & DI reg.
iii. Clear direction flag
iv. Load counter with length
v. Get first element into AL , check whether the input is valid
vi. If invalid go to end
vii. Else translate code from table
viii. Store code
ix. Loop until counter becomes zero
x. Halt

Program:

Address	Label	Mnemonics	Machine Code	Comments
		MOV BX, 1200H		
		MOV SI, 2000H		
		MOV DI, 3000H		
		CLD		
		MOV CX , Length		
	***CONVERT*:**	**LODSB**		
		CMP AL, 09H		
		JA *INVALID*		
		XLAT		
		STOSB		
		LOOP *CONVERT*		
		INT 3		
	INVALID:	**MOV AL, 00**		
		MOV [DI], AL		
		INT 3		

OUTPUT VERIFIACTION:

INPUT	DATA 1	DATA 2	DATA 2
2000H (DECIMAL DIGIT)	03		
2001	05		
OUTPUT			
3000H	4F		
3001H	6D		

CONVERSION TABLE: **LOAD FROM Memory Address: 1200H**

BCD digit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
7 segment code	**3F**	**06**	**5B**	**4F**	**66**	**6D**	**7D**	**07**	**7F**	**6F**

d) Find & Replace

Algorithm:

i. Initialize DS & ES registers
ii. Load destination index regs. With address of string
iii. Load counter reg.
iv. Load AL with the string to be found
v. Clear Direction flag for Auto- incrementing mode
vi. Scan for the string
vii. If found replace with the other string in AH
viii. Continue until counter cleared
ix. Halt

Program:

Address	Label	Mnemonics	Machine Code	Comments
		MOV AX, 0000H **MOV DS, AX** **MOV ES, AX** **MOV AL, [2001H]** **MOV DI, 3000H** **MOV CX ,[2000H]** **CLD** ***LOOP1:*** **REPNZ SCASB** **JNZ *DONE*** **MOV AH, [2002H]** **MOV [DI – 1], AH** **JMP *LOOP1*** ***DONE:*** **INT 3**		

OUTPUT VERIFIACTION:

INPUT	DATA 1	DATA 2	DATA 2
2000H (COUNT)	05		
2001H	CC		
2002H	AA		
3000H	11		
3001H	CC		
3002H	22		
3003H	CC		
3004H	33		
OUTPUT			
3000H	11		
3001H	AA		

3002H	22		
3003H	AA		
3004H	33		

RESULT:

Thus the string manipulation has been performed successfully.

Ex.No: 06 **INTERFACING 8279 & 8253**

8279:

Keyboard Interfacing:

Algorithm:

i. Initialize 8279 with control word to define in 8 bit 8 character display

ii. Initialize clock pre scalar for frequency division

iii. Load display write inhibit word

iv. Read FIFO RAM status if empty wait

v. Else read Data code from input port and suppress unwanted bits(b7&b6)

vi. Break point

Address	Label	Mnemonics	Machine Code	Comments
		MOV AL,12h		Control word to define 8279 in 8 bit 8 character display
		MOV DX,FF52h		8279 control port
		OUT DX,AL		for frequency division
		MOV AL,3Eh		
		OUT DX,AL		display/write inhibit
		MOV AL,A0h		
		OUT DX,AL		
	Wait:	**IN AL,DX**		read status of 8279
		AND AL,07h		FIFO empty?
		JZ Wait		If Yes loop back to wait
		MOV DX,FF50h		Data register
				Read Code from data port
				Suppress bits 7&6

		IN AL,DX **AND AL,3Fh** **INT 3**		Break point

Scan codes for Keyboard:

Row/Column	1	2	3	4
1	**24**	**23**	**22**	**21**
2	**1c**	**1b**	**1a**	**19**
3	**14**	**13**	**12**	**10**
4	**0c**	**0b**	**0a**	**09**

Display interfacing:

Algorithm:

i. Initialize 8279 with control word to define in 8 bit 8 character display

ii. Initialize clock pre scalar for frequency division

iii. Load display write inhibit word and count for clearing display

iv. Clear all display and load BX reg with address of table

v. Get the input code from table and send to display through output port

vi. Break point

Program:

Address	Label	Mnemonics	Machine Code	Comments
		MOV AL,12h		Control word to define 8279 in 8 bit 8 character display
		MOV DX,FF52h		8279 control port
		OUT DX,AL		Load to control reg
		MOV AL,3Eh		for frequency division
		OUT DX,AL		send to control reg
		MOV AL,A0h		display/write inhibit
		OUT DX,AL		send to control reg
		MOV AH,08h		Count of 8 for clearing dis.
		MOV DX,0FF50h		data register address
	Clear:	**MOV AL,00h**		data =0
		OUT DX,AL		
		DEC AH		decrement loop count
		JNZ Clear		Clear up to AH= 00
		MOV DX,FF50h		Data reg
		MOV CL,06		Count for digits
		MOV BX,2000h		Input display code address in BX
	Back:	**MOV AL,[BX]**		Read the input code
		OUT DX, AL		Send to output port for display
		INC BX		Update BX & CL reg
		DEC CL		
		JNZ Back		Loop back

Note: Input data from 2000H

DIGIT	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
7 SEGMENT CODE	FC	60	BA	F2	66	D6	DE	70	FE	76	7E	CE	9C	EA	9E	1E

8253:

Algorithm:

i. Load division factor as count in CX reg.

ii. Select the counter reg by sending appropriate data to control reg.

iii. Get lower order count in to AL and send to selected counter reg

iv. Get higher order count in to AL and send to selected counter reg.

v. Wait and terminate.

Program:

Address	Label	Mnemonics	Machine Code	Comments
		MOV AX,0050H		Division Factor
		MOV CX,AX		
		MOV AL,36H		Counter0 Is Selected
		MOV DX, FF56H		Control Reg
		OUT DX,AL		
		MOV AL,CL		Count Lsb
		MOV DX, FF50H		Counter 0 Reg
		OUT DX,AL		
		MOV AL,CH		Count Msb

		OUT DX,AL **NOP** **INT 3H**		

Result:

Thus the Keyboard/Display controller and Interval Timer has been interfaced and the program for key scan and square wave executed successfully.

Ex.No: 07 **Interfacing 8255 & 8251**

AIM:

To perform parallel/serial communication using 8085 and 8086 by interfacing 8255 and 8251.

APPARATUS REQUIRED:

i) 8086 Microprocessor Kit-2
ii) Parallel data cable
i) PC with serial port
ii) 8085 microprocessor kit
iii) Serial data cable

ALGORITHM:

PARALLEL DATA COMMUNICATION USING 8085/8086

Transmitter:

1. Get the control word of 8255 so as to have one of its as an output port.
2. Sent it to control reg of 8255
3. Get the byte to the transmitted acc
4. Send it to output port

Receiver:

1. Get the control word of 8255 so as to have one of its port as an input port
2. Read it as control reg of 8255
3. Read the data at the input port, if there isn`t carry, check again after delay.

PARALLEL DATA COMMUNICATION USING 8086/8085

- Using a parallel data communication bus connected parallel ports of two 8086 microprocessor.
- Load transmission program in one of the bit to make it as transmitter.
- Load receiver program in another kit to have it as receiver.
- Execute the receiver to have it ready for receiving data.
- Execute the transmitter data on display/acc of receiver.

PROGRAM:

PARALLEL DATA COMMUNICATION USING 8085

ADDRESS	LABEL	MNEMONICS	HEXCODE	COMMENTS
Transmitter:				
8500		MVI A,82	3E,82	Move 82 to A reg
8502		OUT 23	D3,23	Get the input port
8504		MVI A,55	3E,55	Move 55 to A reg

8506		OUT 20	D3,20	Get the output
8508		RST 1	CF	Break point
Receiver:				
8500		MVI A,99	35,99	Move 99 to A reg
8502		OUT 23	D3,23	Get the input port
8504		IN 20	DB,20	
8506		NOP	00	No operation
8507		NOP	00	No operation
8508		NOP	00	No operation
8509		NOP	00	No operation
850A		RST 1	CF	Break point

PARALLEL DATA COMMUNICATION USING 8086

ADDRESS	LABEL	MNEMONICS	HEXCODE	COMMENTS
Transmitter:				
9000		MOV DX,FF26	BA,26,FF	Move FF26 to DX reg
9003		MOV AL,82	BO,82	Move 82 to AC reg
9005		OUT DX,AL	EE	

9006		MOV DX,FF24	BA,24,FF	Move FF24 to DX reg
9009		MOV AL,00	B0,00	Move 00 to A reg
900B		OUT DX,AL	EE	
900C		MOV DX,FF20	BA,20FF	Move FF24 to BX reg
900F		MOV AL,55	BO,55	Move 55 to A reg
9011		OUT DX,AL	EE	
9012		MOV DX,FF24	BA,24,FF	Move FF24 to D reg
9015		MOV AL,FF	BO,FF	Move FF to A reg
9017		OUT DX,AL	EE	
9018		INT 3	CC	Break point
Receiver:				
8300		MOV DX,FF26	BA,26,FF	Move FF26 to D reg
8303		MOV AL,99	BO,99	Move 99 to A reg
8305		OUT DX,AL	EE	

8306		MOV DX,FF24	3A,24,FF	Move FF24 to D reg
8309		IN AL,DX	EC	
830A		JZ 8309	74,FD	Jump on 8309
830C		MOV DX,FF20	BA,20,FF	Move FF29 to D reg
830F		IN AL,DX	EC	
8310		INT 3	CC	Break point

SERIAL DATA COMMUNICATION BETWEEN A PC AND 8085

ADDRESS	LABEL	MNEMONICS	HEXCODE	COMMENTS
9000		LDA 9200	3A,00,92	Load the content
9003		MOV A,B	47	Move the content
9004		LDA 9201	3A,01,92	Load the content
9007		ADD B	80	Add with A

9008		MVI C,00	0E,00	Initialize carry as 0
900A		JNC 910E	02,0E,90	Check carry flag
900D		INR C	0C	Increment C
900E		STA 9600	32,00,96	Store the result
9011		MOV A,C	79	Moves the carry
9012		STA 9601	32,01,96	Store the carry
9013		RST 1	CF	Stop the program

SERIAL DATA COMMUNICATION BETWEEN A PC AND 8086

1000		MOV AX,1100	A1,00,11	Move the values
1003		ADD AX,1102	D3,06,02,11	Add the value
1007		MOV 1104,AX	A3,04,11	Store the value

100A		INT 3	CC	Stop the program

PARALLEL DATA COMMUNICATION

Output 8085:

MEMORY ADDRESS	TRANSMITTER	RECEIVER
8500	55	55

Output 8086:

TRANSMITTER	RECEIVER
09FF (G9000)	CF55 (G8300)

SERIAL DATA COMMUNICATION

Output 8085:

MEMORY ADDRESS	INPUT	OUTPUT
9200 9201	02 02	(9600) 04

Output 8086:

MEMORY	ADDRESS	OUTPUT
2000-01	3000-01	02
2001-02	3001-02	04
2002-03	3002-03	06
2003-04	3003-04	08

2004-05	3004-05	0A

RESULT:

Thus the execution of data communication of parallel/serial is executed successfully by means of 8255 and 8251.

Ex.No:08 STEPPER MOTOR & DC MOTOR CONTROL

AIM:

To write a program to interface a stepper motor using 8085/8086/8051 microprocessor.

APPARATUS REQUIRED:

i) 8051 microprocessor kit.
ii) 8085 microprocessor kit.
iii) 8086 microprocessor kit.
iv) Stepper motor

ALGORITHM:

1. Start the program
2. Move the control word to accumulator
3. Move the control word to control register
4. Store the accumulator content through port A
5. Change the accumulator content with phase sequence
6. Stop the program.

PROGRAM FOR 8085:

ADDRESS	LABEL	MNEMONICS	HEXCODE	COMMENTS
9000		MVI A,80	3E,80	Control port of 8255

9002		OUT 23	D3,23	Control word
9004	BACK	MVI A,80	3E,A0	First step sequence
9006		OUT 20	D3,20	Out it in port A
9008		CALL DELAY	CD 0091	Call delay routine
900B		MVI A,E0	3E,E0	Second step sequence
900D		OUT 20	D3,20	Out it in port A
900F		CALL DELAY	CD 0091	Call delay routine
9012		MVI A,C0	3E,10	Third step sequence
9014		OUT 20	D3,20	Out it in port A

9016		CALL DELAY	CD 0091	Call delay routine
9019		MVI A,80	3E,C0	Fourth step sequence
901B		OUT 20	03,20	Out it in port A
901D		CALL DELAY	CD,00,91	Call delay routine
9020		JMP BACK	C3,04	Jump back
9100	DELAY	MVI B,10	06,10	Change count value
9102		MVI A,FF	3E,FF	Move the delay
9104		NOP	00	No operation
9105		NOP	00	No operation

9106		NOP	00	No operation
9107		NOP	00	No operation
9108		DCR A	3D	Decrement A
9109		JNZ 9104	C2,0491	Jump on non 2010
910C		DCR B	05	Decrement B
910D		JNZ 9102	C2,02,91	Jump on non zero
9110		RET	C9	Return to main program

PROGRAM FOR 8086:

ADDRESS	LABEL	MNEMONICS	HEXCODE	COMMENTS

9000		MOV DX,FF26	BA,26FF	Control port of 8255
9003		MOV AL,80	B0,80	Control word
9005		OUT DX,AL	EE	Mov bit output
9006		MOV DX,FF20	BA,20FF	Move control word to control reg
9009		MOV AL,A0	BO,AD	First step sequence
900B		OUT DX,AL	EE	Out it in port A
900C		CALL DELAY	E8,F100	Call delay
900F		MOV AL,E0	BO,E0	Second step segment
9011		OUT DX,AL	EE	Out it in port A
9012		CALL DELAY	E8,EB,00	Call delay

9015		MOV AL,C0	B0,E0	Fourth step sequence
9017		OUT DX,AL	EE	Out in port A
9018		CALL DELAY	E8,EB00	Jump back to 9006
901B		MOV AL,80	BO,80	
901D		OUT DX,AL	EE	
901E		CALL DELAY	E8,EB,00’	
9021		JMP 9006	EB,E3	Jump back to 9006
9023	DELAY	MOV BX,0002	BB,9000	Change count
9026		MOV AL,FF	BO,FF	
9028		NOP	00	No operation

9029		NOP	00	No operation
902A		NOP	00	No operation
902B		NOP	00	No operation
902C		DEC AL	FFF CD	Decrement AL
902E		JNZ 9028	75 F8	Jump on non-zero
9030		DEC BX		Decrement BX
9031		JNZ 9026		Jump on non zero
9033		RET		Return to main program

PROGRAM FOR 8051:

ADDRESS	LABEL	MNEMONICS	HEXCODE	COMMENTS
9000		MOV DPTR,# 4003	90,4003	Control port of 8255
9003		MOV A,#80	74,80	Control word
9005		MOV X@DPTR, A	F0	All bit output
9006		MOV DPTR,#4000	90,4000	Move control cord to control
9009		MOV A,#A0	74,A0	First step sequence
900B		MOV X@DPTR,A	F0	Out in A
900C		LCALL DELAY	12,9100	Call delay
900F		MOV A,#E0	74,A0	Second step sequence
9011		MOV X@DPTR, A	F0	Out it in port A

		LCALL DELAY		
9012			12,9100	Call delay
		MOV A,#C0		
9015			74,C0	Third step sequence
		MOV X@DPTR, A		
9017			F0	Out it in port A
		LCALL DELAY		
9018			12,9100	Call delay
		MOV A,#80		
901B			74,80	Fourth step sequence
		MOV X@DPTR,A		
901D		L CALL DELAY	F0	Out it in port A
901E		LJMP 9009	12,9100	Call delay
9021		MOV R1,#0A	02,9009	Jump back to 9009
9024	DELAY	MOV A,#A0	79,0A	Change count value

9026	DELAY1	NOP	74,40	Move the delay value
9028	BACK	NOP	00	No operation
9029		NOP	00	No operation
902A		NOP	00	No operation
902B		DEC A	00	No operation
902C		JNZ BACK	14	Jump on non-zero
902D		JNZ R1,DELAY1	70,F9	Jump on non-zero
902F		RET	D9,F5	Return to main program
9031			22	

PHASE SEQUENCE:

PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	
1	0	0	0	0	0	0	0	80
1	0	1	0	0	0	0	0	A0
1	1	1	0	0	0	0	0	E0
1	1	0	0	0	0	0	0	C0

DC Motor:

Algorithm:

i. Load control Reg address of 8255

ii. Get control word so as to program 8255 ports as O/P port & send to control reg

iii. Send the start pulse to ZCD

iv. Call delay high routine

v. Send stop pulse to ZCD

vi. Call delaylow

vii. Continue cycle

Program:

Address	Label	Mnemonics	Machine Code	Comments
		MOV DX , FF26H		8255 Control Reg Address
		MOV AL ,80H		Control word for output ports
		OUT DX ,AL		Send to control reg
	CONT:	MOV AL ,01H		Start Pulse
		MOV DX , FF20H		Port C Address
		OUT DX ,AL		
		CALL DELAYHIGH		
		MOV AL,00H		Stop Pulse
		OUT DX,AL		
		CALL DELAYLOW		
		JMP CONT		Continue Cycle
	DELAYHIGH:	MOV AH,01H		Motor Speed Count (01 TO FF)
	LOOP1:	NOP		
		NOP		
		NOP		
		DEC AH		

		JNZ LOOP1 RET		
	DELAYLOW: LOOP2:	MOV AL,01H NOP NOP DEC AL JNZ LOOP2 RET		

Procedure:

i) Connect DC motor interface with VIK-86 through data bus and connect DC motor with interface.

ii) Give power supply to interface kit

iii) Load program and execute to run motor

iv) Change delay Count to control Speed

RESULT:

Thus the interfacing of stepper motor and DC motor has been done successfully and the speed control achieved.

Ex.No: 09 **8051 BASIC PROGRAMMING**

Aim:

To program 8051 using its Arithmetic and Logical and Bit Manipulation instructions.

a) Arithmetic operations

Address	Label	Mnemonics	Machine Code	Comments

		MOV DPTR, #8500
		MOVX A, @DPTR
		MOV B, A
		MOV R0, A
		INC DPTR
		MOVX A, @DPTR
		MOV R1, A
		ADD A, B
		INC DPTR
		MOVX @DPTR, A
		MOV R2, A
		MOV A, R1
		SUBB A, B
		INC DPTR
		MOVX @DPTR, A
		MOV R3, A
		MOV B, R2
		MUL AB		

		INC DPTR **MOVX @DPTR, A** **MOV A, R2** **MOV B, R3** **DIV AB** **INC DPTR** **MOVX @DPTR, A** **LCALL 00BB**		

Input: M8500 - a

M8501 - b

Output: M8502 : sum (a+b)

M8503: difference (a-b)

M8504: Product ((a+b)×(a-b))

M8505: Quotient ((a+b)/(a-b))

b) 32 bit subtraction

Address	Label	Mnemonics	Machine Code	Comments
		CLR C **MOV A, 43** **SUBB A, 53** **MOV 63, A** **MOV A, 42** **SUBB A, 52** **MOV 62, A** **MOV A, 41** **SUBB A, 51** **MOV 61, A** **MOV A, 40** **SUBB A, 50** **MOV 60, A** **LCALL 00BB**		

Input: I40 to 43 – data 1

I50 to 53 – data 2

Output: I60 to 63 - difference

C) Fibonacci series

Address	Label	Mnemonics	Machine Code	Comments
		MOV R0, 60		
		MOV R1, #01		
		MOV R2, #01		
		MOV A, #00		
		MOV DPTR, # 9000		
		CJNE R0, #00, BEGIN		
		LJMP EXIT		
	BEGIN:	MOVX @DPTR, A		
		INC DPTR		
	RPT:	MOV R2, A		
		ADD A, R1		
		MOV 01, 02		
		MOVX @DPTR, A		

	EXIT:	INC DPTR DJNZ R0, RPT LCALL 00BB		

INPUT: I60 – COUNT

OUTPUT: M9000 – 00

M9001 – 01

M9002 – 01

M9003 – 02 & so on…

Ex.No: 10 **Communications between PC & 8085/8086/8051**

PROCEDURE:

PC to Kit:

1. Connect pc and microprocessor kit using serial data cable
2. Put kit in serial mode
3. Enter the starting address with mode of loading
4. Load the program
5. Reset to normal mode
6. Now check the data in the corresponding address of kit.

Kit to PC:

1. Connect pc and microprocessor kit using serial data cable
2. Load the program in kit as usual
3. Now put the kit in serial mode
4. Check data on the address of kit.