LABORATORY MANUAL

PROGRAMME: B.Tech

SEMESTER /YEAR: 3rd year 5th Semester

SUBJECT CODE: BM0313

SUBJECT NAME: Microprocessor & Microcontroller Lab

Prepared By:

Name: A Bhargavi Haripriya

Designation: Assistant Professor (OG)

DEPARTMENT OF BIOMEDICAL ENGINEERING

SCHOOL OF BIOENGINEERING,
FACULTY OF ENGINEERING & TECHNOLOGY

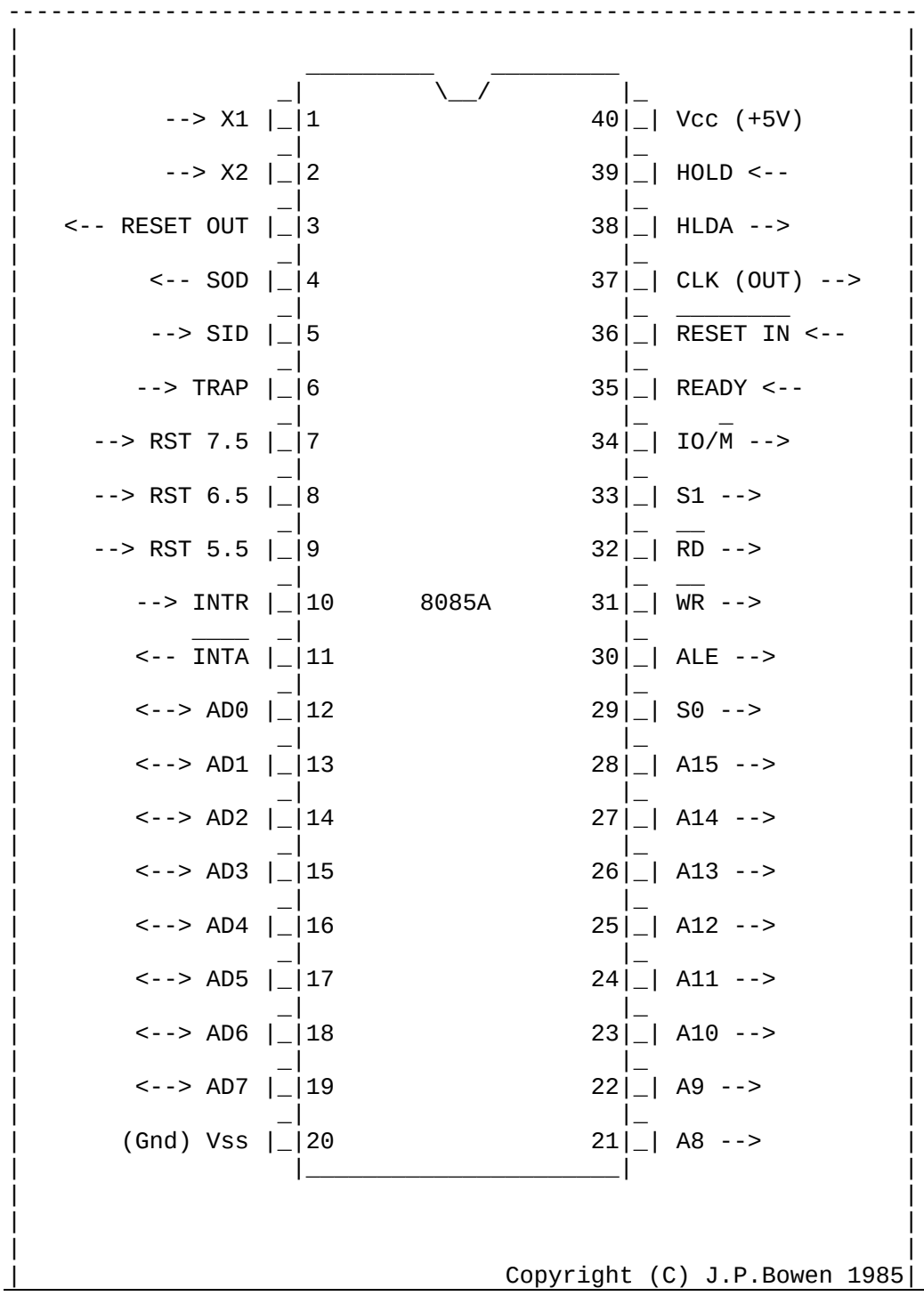
SRM UNIVERSITY

(UNDER SECTION 3 of UGC ACT 1956)
KATTANKULATHUR-603203
Tamil Nadu, India

Contents

- **Interfacing Programs**
  18. **Generation of Saw tooth waveform**
  19. **Generation of Triangular waveform**
  20. **Generation of Square waveform**
  21. **Generation of Sine waveform8086 Pin Diagram**
- **8086 Pin Diagram**
- **8086 Instruction Set**
- **8086 Programs**
  22. **16-bit addition using 8086 microprocessor**
  23. **Move contents of array**
  24. **Sum of 'n' consecutive numbers**
  25. **Conversion of BCD number to decimal**
  26. **Separating Odd and Even numbers**
- **8051 Pin Diagram**
- **8051 Instruction Set**
- **8051 Programs**
  27. **Addition of 8-bit numbers using 8051**
  28. **Subtraction of 8-bit numbers using 8051**

8085 Pin Diagram



8085 Instruction Set

Instructions can be categorized according to their method of addressing the hardware registers and/or memory.

Implied Addressing:
The addressing mode of certain instructions is implied by the instruction's function. For example, the STC (set carry flag) instruction deals only with the carry flag, the DAA (decimal adjust accumulator) instruction deals with the accumulator.

Register Addressing:
Quite a large set of instructions call for register addressing. With these instructions, you must specify one of the registers A through E, H or L as well as the operation code. With these instructions, the accumulator is implied as a second operand. For example, the instruction CMP E may be interpreted as 'compare the contents of the E register with the contents of the accumulator.

Immediate Addressing:
Instructions that use immediate addressing have data assembled as a part of the instruction itself. For example, the instruction CPI 'C' may be interpreted as 'compare the contents of the accumulator with the letter C. When assembled, this instruction has the hexadecimal value FE43. Hexadecimal 43 is the internal representation for the letter C. When this instruction is executed, the processor fetches the first instruction byte and determines that it must fetch one more byte. The processor fetches the next byte into one of its internal registers and then performs the compare operation.

Direct Addressing:
Jump instructions include a 16-bit address as part of the instruction. For example, the instruction JMP 1000H causes a jump to the hexadecimal address 1000 by replacing the current contents of the program counter with the new value 1000H.

Instructions that include a direct address require three bytes of storage: one for the instruction code, and two for the 16-bit address

Register Indirect Addressing:
Register indirect instructions reference memory via a register pair. Thus, the instruction MOV M,C moves the contents of the C register into the memory address stored in the H and L register pair. The instruction LDAX B loads the accumulator with the byte of data specified by the address in the B and C register pair.

Combined Addressing Modes:
Some instructions use a combination of addressing modes. A CALL instruction, for example, combines direct addressing and register indirect addressing. The direct address in a CALL instruction specifies the address of the desired subroutine; the register indirect address is the stack pointer. The CALL instruction pushes the current contents of the program counter into the memory location specified by the stack pointer.

Timing Effects of Addressing Modes:
Addressing modes affect both the amount of time required for executing an instruction and the amount of memory required for its storage. For example, instructions that use implied or register addressing, execute very quickly since they deal directly with the processor's hardware or with data already present in hardware registers. Most important, however is that the entire instruction can be fetched with a single memory access. The number of memory accesses required is the single greatest factor in determining execution timing. More memory accesses therefore require more execution time. A CALL instruction for example, requires five memory accesses: three to access the entire instruction and two more to push the contents of the program counter onto the stack.

The processor can access memory once during each processor cycle. Each cycle comprises a variable number of states. (See below and the appendix of "USING THE SDK-85 MICROPROCESSOR TRAINER"). The length of a state depends on the clock frequency specified for your system, and may range from 480 nanoseconds to 2 microseconds. Thus, the timing for a four state instruction may range from 1.920 microseconds through 8 microseconds. (The 8085 have a maximum clock frequency of 5 MHz and therefore a minimum state length of 200 nanoseconds.)

Instruction Naming Conventions:
The mnemonics assigned to the instructions are designed to indicate the function of the instruction. The instructions fall into the following functional categories:

Data Transfer Group:
The data transfer instructions move data between registers or between memory and registers.

MOV	Move
MVI	Move Immediate
LDA	Load Accumulator Directly from Memory
STA	Store Accumulator Directly in Memory
LHLD	Load H & L Registers Directly from Memory
SHLD	Store H & L Registers Directly in Memory

An 'X' in the name of a data transfer instruction implies that it deals with a register pair (16-bits);

LXI	Load Register Pair with Immediate data
LDAX	Load Accumulator from Address in Register Pair
STAX	Store Accumulator in Address in Register Pair
XCHG	Exchange H & L with D & E
XTHL	Exchange Top of Stack with H & L

Arithmetic Group:
The arithmetic instructions add, subtract, increment, or decrement data in registers or memory.

ADD	Add to Accumulator
ADI	Add Immediate Data to Accumulator
ADC	Add to Accumulator Using Carry Flag
ACI	Add Immediate data to Accumulator Using Carry
SUB	Subtract from Accumulator
SUI	Subtract Immediate Data from Accumulator
SBB	Subtract from Accumulator Using Borrow (Carry) Flag
SBI	Subtract Immediate from Accumulator Using Borrow (Carry) Flag

INR	Increment Specified Byte by One
DCR	Decrement Specified Byte by One
INX	Increment Register Pair by One
DCX	Decrement Register Pair by One
DAD	Double Register Add; Add Content of Register Pair to H & L Register Pair

Logical Group:
This group performs logical (Boolean) operations on data in registers and memory and on condition flags.

The logical AND, OR, and Exclusive OR instructions enable you to set specific bits in the accumulator ON or OFF.

ANA	Logical AND with Accumulator
ANI	Logical AND with Accumulator Using Immediate Data
ORA	Logical OR with Accumulator
OR	Logical OR with Accumulator Using Immediate Data
XRA	Exclusive Logical OR with Accumulator
XRI	Exclusive OR Using Immediate Data

The Compare instructions compare the content of an 8-bit value with the contents of the accumulator;

CMP	Compare
CPI	Compare Using Immediate Data

The rotate instructions shift the contents of the accumulator one bit position to the left or right:

RLC	Rotate Accumulator Left
RRC	Rotate Accumulator Right
RAL	Rotate Left Through Carry
RAR	Rotate Right Through Carry

Complement and carry flag instructions:

CMA	Complement Accumulator
CMC	Complement Carry Flag
STC	Set Carry Flag

Branch Group:

The branching instructions alter normal sequential program flow, either unconditionally or conditionally. The unconditional branching instructions are as follows:

JMP	Jump
CALL	Call
RET	Return

Conditional branching instructions examine the status of one of four condition flags to determine whether the specified branch is to be executed. The conditions that may be specified are as follows:

NZ	Not Zero (Z = 0)
Z	Zero (Z = 1)
NC	No Carry (C = 0)
C	Carry (C = 1)
PO	Parity Odd (P = 0)
PE	Parity Even (P = 1)
P	Plus (S = 0)
M	Minus (S = 1)

Thus, the conditional branching instructions are specified as follows:

Jumps	Calls	Returns	
JC	CC	RC	(Carry)
INC	CNC	RNC	(No Carry)
JZ	CZ	RZ	(Zero)
JNZ	CNZ	RNZ	(Not Zero)
JP	CP	RP	(Plus)
JM	CM	RM	(Minus)
JPE	CPE	RPE	(Parity Even)
JP0	CPO	RPO	(Parity Odd)

Two other instructions can affect a branch by replacing the contents or the program counter:

PCHL	Move H & L to Program Counter
RST	Special Restart Instruction Used with Interrupts

Stack I/O, and Machine Control Instructions:
The following instructions affect the Stack and/or Stack Pointer:

PUSH	Push Two bytes of Data onto the Stack
POP	Pop Two Bytes of Data off the Stack
XTHL	Exchange Top of Stack with H & L
SPHL	Move content of H & L to Stack Pointer

The I/0 instructions are as follows:

IN	Initiate Input Operation
OUT	Initiate Output Operation

The Machine Control instructions are as follows:

EI	Enable Interrupt System
DI	Disable Interrupt System
HLT	Halt
NOP	No Operation

Mnemonic	Op	SZAPC	~s	Description	Notes
ACI n	CE	*****	7	Add with Carry Immediate	A=A+n+CY
ADC r	8F	*****	4	Add with Carry	A=A+r+CY(21X)
ADC M	8E	*****	7	Add with Carry to Memory	A=A+[HL]+CY
ADD r	87	*****	4	Add	A=A+r (20X)
ADD M	86	*****	7	Add to Memory	A=A+[HL]
ADI n	C6	*****	7	Add Immediate	A=A+n
ANA r	A7	****0	4	AND Accumulator	A=A&r (24X)
ANA M	A6	****0	7	AND Accumulator and Memory	A=A&[HL]
ANI n	E6	**0*0	7	AND Immediate	A=A&n
CALL a	CD	-----	18	Call unconditional	-[SP]=PC,PC=a
CC a	DC	-----	9	Call on Carry	If CY=1(18~s)
CM a	FC	-----	9	Call on Minus	If S=1 (18~s)
CMA	2F	-----	4	Complement Accumulator	A=~A
CMC	3F	----*	4	Complement Carry	CY=~CY
CMP r	BF	*****	4	Compare	A-r (27X)
CMP M	BF	*****	7	Compare with Memory	A-[HL]
CNC a	D4	-----	9	Call on No Carry	If CY=0(18~s)
CNZ a	C4	-----	9	Call on No Zero	If Z=0 (18~s)
CP a	F4	-----	9	Call on Plus	If S=0 (18~s)
CPE a	EC	-----	9	Call on Parity Even	If P=1 (18~s)
CPI n	FE	*****	7	Compare Immediate	A-n
CPO a	E4	-----	9	Call on Parity Odd	If P=0 (18~s)
CZ a	CC	-----	9	Call on Zero	If Z=1 (18~s)
DAA	27	*****	4	Decimal Adjust Accumulator	A=BCD format
DAD B	09	----*	10	Double Add BC to HL	HL=HL+BC
DAD D	19	----*	10	Double Add DE to HL	HL=HL+DE
DAD H	29	----*	10	Double Add HL to HL	HL=HL+HL
DAD SP	39	----*	10	Double Add SP to HL	HL=HL+SP
DCR r	3D	****-	4	Decrement	r=r-1 (0X5)
DCR M	35	****-	10	Decrement Memory	[HL]=[HL]-1
DCX B	0B	-----	6	Decrement BC	BC=BC-1
DCX D	1B	-----	6	Decrement DE	DE=DE-1
DCX H	2B	-----	6	Decrement HL	HL=HL-1
DCX SP	3B	-----	6	Decrement Stack Pointer	SP=SP-1
DI	F3	-----	4	Disable Interrupts	
EI	FB	-----	4	Enable Interrupts	
HLT	76	-----	5	Halt	
IN p	DB	-----	10	Input	A=[p]
INR r	3C	****-	4	Increment	r=r+1 (0X4)
INR M	3C	****-	10	Increment Memory	[HL]=[HL]+1
INX B	03	-----	6	Increment BC	BC=BC+1
INX D	13	-----	6	Increment DE	DE=DE+1
INX H	23	-----	6	Increment HL	HL=HL+1
INX SP	33	-----	6	Increment Stack Pointer	SP=SP+1
JMP a	C3	-----	7	Jump unconditional	PC=a
JC a	DA	-----	7	Jump on Carry	If CY=1(10~s)
JM a	FA	-----	7	Jump on Minus	If S=1 (10~s)
JNC a	D2	-----	7	Jump on No Carry	If CY=0(10~s)
JNZ a	C2	-----	7	Jump on No Zero	If Z=0 (10~s)
JP a	F2	-----	7	Jump on Plus	If S=0 (10~s)

JPE a	EA	-----	7	Jump on Parity Even	If P=1 (10~s)
JPO a	E2	-----	7	Jump on Parity Odd	If P=0 (10~s)
JZ a	CA	-----	7	Jump on Zero	If Z=1 (10~s)
LDA a	3A	-----	13	Load Accumulator direct	A=[a]
LDAX B	0A	-----	7	Load Accumulator indirect	A=[BC]
LDAX D	1A	-----	7	Load Accumulator indirect	A=[DE]
LHLD a	2A	-----	16	Load HL Direct	HL=[a]
LXI B,nn	01	-----	10	Load Immediate BC	BC=nn
LXI D,nn	11	-----	10	Load Immediate DE	DE=nn
LXI H,nn	21	-----	10	Load Immediate HL	HL=nn
LXI SP,nn	31	-----	10	Load Immediate Stack Ptr	SP=nn
MOV r1,r2	7F	-----	4	Move register to register	r1=r2 (1XX)
MOV M,r	77	-----	7	Move register to Memory	[HL]=r (16X)
MOV r,M	7E	-----	7	Move Memory to register	r=[HL] (1X6)
MVI r,n	3E	-----	7	Move Immediate	r=n (0X6)
MVI M,n	36	-----	10	Move Immediate to Memory	[HL]=n
NOP	00	-----	4	No Operation	
ORA r	B7	**0*0	4	Inclusive OR Accumulator	A=Avr (26X)
ORA M	B6	**0*0	7	Inclusive OR Accumulator	A=Av[HL]
ORI n	F6	**0*0	7	Inclusive OR Immediate	A=Avn
OUT p	D3	-----	10	Output	[p]=A
PCHL	E9	-----	6	Jump HL indirect	PC=[HL]
POP B	C1	-----	10	Pop BC	BC=[SP]+
POP D	D1	-----	10	Pop DE	DE=[SP]+
POP H	E1	-----	10	Pop HL	HL=[SP]+
POP PSW	F1	-----	10	Pop Processor Status Word	{PSW,A}=[SP]+

Mnemonic	Op	SZAPC	~s	Description	Notes
PUSH B	C5	-----	12	Push BC	-[SP]=BC
PUSH D	D5	-----	12	Push DE	-[SP]=DE
PUSH H	E5	-----	12	Push HL	-[SP]=HL
PUSH PSW	F5	-----	12	Push Processor Status Word	-[SP]={PSW,A}
RAL	17	----*	4	Rotate Accumulator Left	A={CY,A}<-
RAR	1F	----*	4	Rotate Accumulator Righ	A=->{CY,A}
RET	C9	-----	10	Return	PC=[SP]+
RC	D8	-----	6	Return on Carry	If CY=1(12~s)
RIM	20	-----	4	Read Interrupt Mask	A=mask
RM	F8	-----	6	Return on Minus	If S=1 (12~s)
RNC	D0	-----	6	Return on No Carry	If CY=0(12~s)
RNZ	C0	-----	6	Return on No Zero	If Z=0 (12~s)
RP	F0	-----	6	Return on Plus	If S=0 (12~s)
RPE	E8	-----	6	Return on Parity Even	If P=1 (12~s)
RPO	E0	-----	6	Return on Parity Odd	If P=0 (12~s)
RZ	C8	-----	6	Return on Zero	If Z=1 (12~s)
RLC	07	----*	4	Rotate Left Circular	A=A<-
RRC	0F	----*	4	Rotate Right Circular	A=->A
RST z	C7	-----	12	Restart (3X7)	-[SP]=PC,PC=z
SBB r	9F	*****	4	Subtract with Borrow	A=A-r-CY
SBB M	9E	*****	7	Subtract with Borrow	A=A-[HL]-CY
SBI n	DE	*****	7	Subtract with Borrow Immed	A=A-n-CY
SHLD a	22	-----	16	Store HL Direct	[a]=HL

SIM	30	-----	4	Set Interrupt Mask	mask=A
SPHL	F9	-----	6	Move HL to SP	SP=HL
STA a	32	-----	13	Store Accumulator	[a]=A
STAX B	02	-----	7	Store Accumulator indirect	[BC]=A
STAX D	12	-----	7	Store Accumulator indirect	[DE]=A
STC	37	----1	4	Set Carry	CY=1
SUB r	97	*****	4	Subtract	A=A-r (22X)
SUB M	96	*****	7	Subtract Memory	A=A-[HL]
SUI n	D6	*****	7	Subtract Immediate	A=A-n
XCHG	EB	-----	4	Exchange HL with DE	HL<->DE
XRA r	AF	**0*0	4	Exclusive OR Accumulator	A=Axr (25X)
XRA M	AE	**0*0	7	Exclusive OR Accumulator	A=Ax[HL]
XRI n	EE	**0*0	7	Exclusive OR Immediate	A=Axn
XTHL	E3	-----	16	Exchange stack Top with HL	[SP]<->HL

PSW	-*01		Flag unaffected/affected/reset/set
S	S		Sign (Bit 7)
Z	Z		Zero (Bit 6)
AC	A		Auxilary Carry (Bit 4)
P	P		Parity (Bit 2)
CY	C		Carry (Bit 0)

a p	Direct addressing
M z	Register indirect addressing
n nn	Immediate addressing
r	Register addressing

DB n(,n)	Define Byte(s)
DB 'string'	Define Byte ASCII character string
DS nn	Define Storage Block
DW nn(,nn)	Define Word(s)

A B C D E H L	Registers (8-bit)
BC DE HL	Register pairs (16-bit)
PC	Program Counter register (16-bit)
PSW	Processor Status Word (8-bit)
SP	Stack Pointer register (16-bit)

a nn	16-bit address/data (0 to 65535)
n p	8-bit data/port (0 to 255)
r	Register (X=B,C,D,E,H,L,M,A)
z	Vector (X=0H,8H,10H,18H,20H,28H,30H,38H)

+ -	Arithmetic addition/subtraction
& ~	Logical AND/NOT
v x	Logical inclusive/exclusive OR
<- ->	Rotate left/right
<->	Exchange
[ ]	Indirect addressing
[ ]+ -[ ]	Indirect address auto-inc/decrement
{ }	Combination operands
(X)	Octal op code where X is a 3-bit code
If (~s)	Number of cycles if condition true

Experiment No:

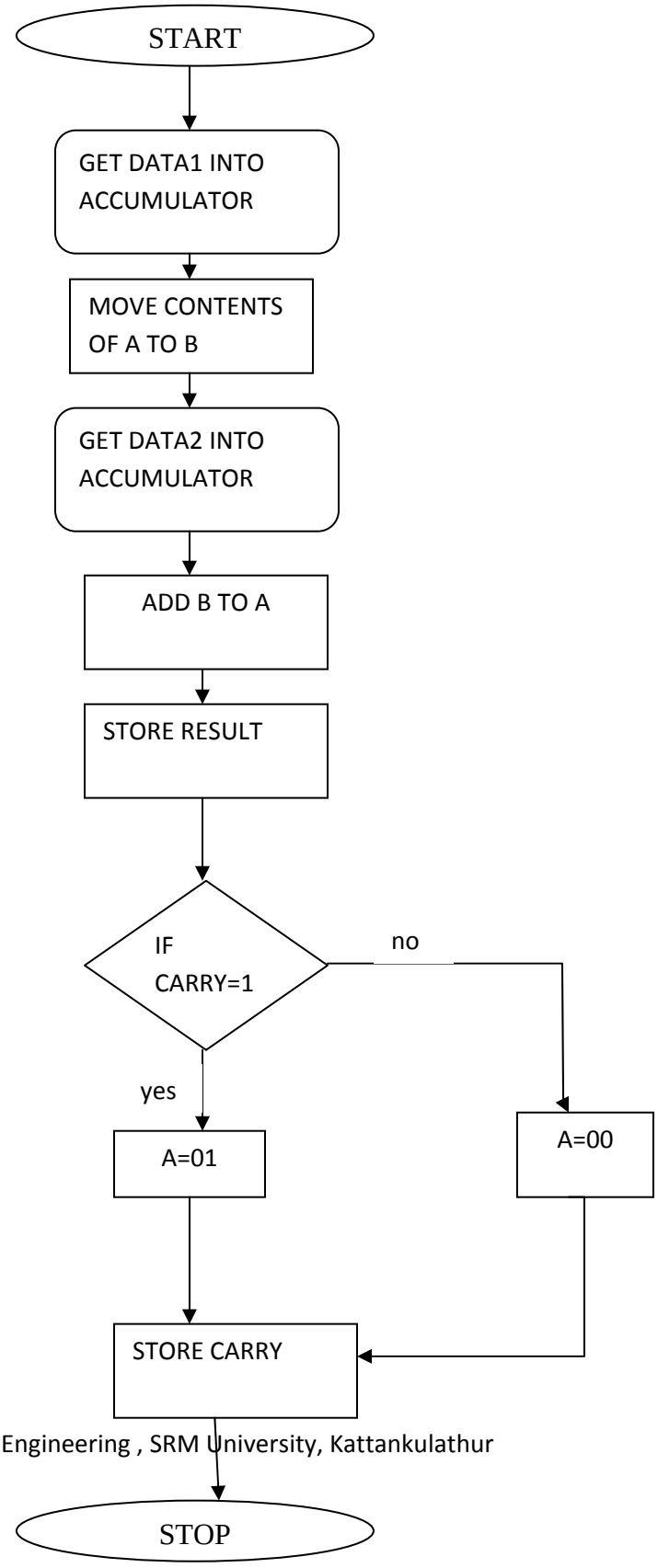
8-BIT ADDITION USING ACCUMULATOR

AIM: To perform 8-bit Addition of two hexadecimal numbers using accumulator in 8085

PROGRAM:

Address	Program	Explanation
	LDA 9000	Load contents of address 9000 (Input 1) into the accumulator
	MOV B,A	Shift contents of accumulator to register B
	LDA 9001	Load contents of address 9001(Input 2) into the accumulator
	ADD B	Add data in B to accumulator & store in the accumulator
	STA 9002	Store the (Result) contents of accumulator to address 9002
	JC Loc1	Jump to Loc1 if carry is 1
	MVI A,00	Store value 00 in accumulator
	STA 9003	Store the (Carry) contents of accumulator to address 9003
	HLT	End of Program
Loc 1:	MVI A,01	Store value 01 in accumulator
	STA 9003	Store the (Carry) contents of accumulator to address 9003
	HLT	End of Program

FLOWCHART


START
GET DATA1 INTO ACCUMULATOR
MOVE CONTENTS OF A TO B
GET DATA2 INTO ACCUMULATOR
ADD B TO A
STORE RESULT
IF CARRY=1
no
yes
A=01
A=00
STORE CARRY
STOP

MNEMONICS:

ADDRESS	OPCODE	OPERAND	LOCATION	COMMAND
8000	3A	00 90		LDA 9000
8003	47			MOV B,A
8004	3A	01 90		LDA 9001
8007	80			ADD B
8008	32	02 90		STA 9002
800B	DA	14 80		JC Loc1
800E	3E	00		MVI A,00
8010	32	03 90		STA 9003
8013	76			HLT
8014	3E	01	Loc 1	MVI A,01
8016	32	03 90		STA 9003
8019	76			HLT

INPUT & OUTPUT:

Location		Values
9000	INPUT 1	42
9001	INPUT 2	35
9002	RESULT	77
9003	CARRY	00

Location		Values
9000	INPUT 1	A8
9001	INPUT 2	F6
9002	RESULT	9E
9003	CARRY	01

CALCULATION:

i) Data 1 : 42 - 0100 0010
Data 2 : 35 - 0011 0101
Sum : 77 – 0111 0111
Carry : 00

ii) Data 1 : A8 - 1010 1000
Data 2 : 35 - 1111 0110
Sum : 9E - 1001 1110
Carry : 01

RESULT: The program for adding two 8-bit hexadecimal numbers using accumulator & registers has been performed with different sets of data.

Date:

Experiment No:

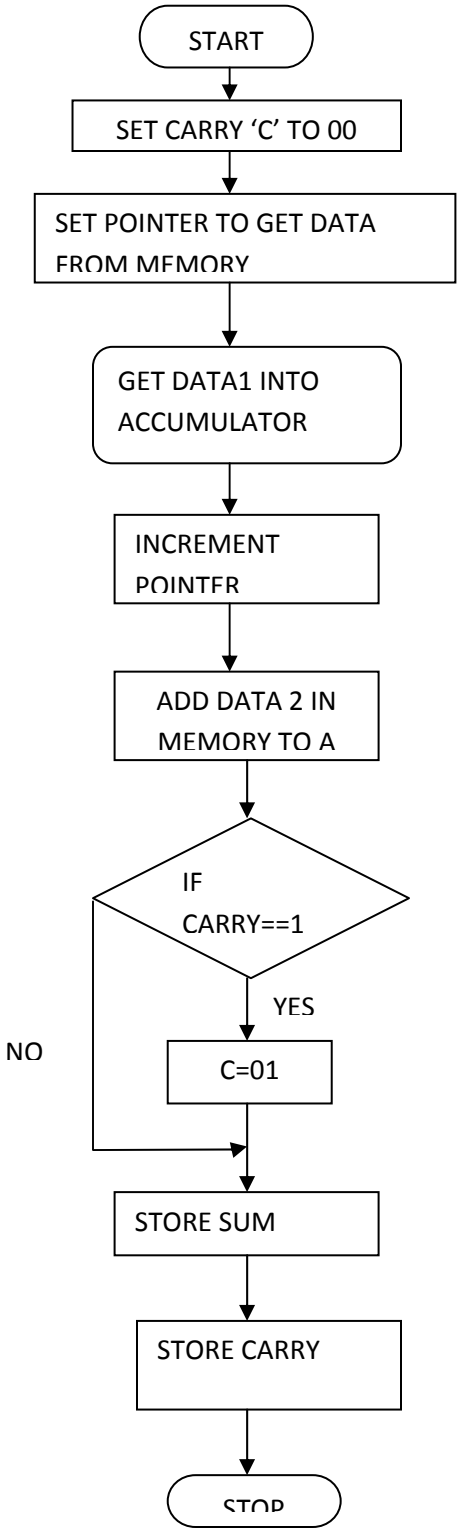
8-BIT ADDITION WITH MEMORY POINTER

AIM: To perform 8-bit Addition of two hexadecimal numbers using memory pointer in 8085

PROGRAM:

Address	Program	Explanation
	LXI H, 8201	Setting pointer for getting data
	MVI C, 00	Clearing register C for carry
	MOV A, M	Getting Data 1 into the accumulator from memory
	INX H	Increment pointer to the next memory location
	ADD M	Add second data to Accumulator and store it in accumulator
	JNC Ahead	If carry is '0' , go to location 'Ahead'
	INR C	If carry is '1' , Increment register C to 1
Ahead	INX H	Increment pointer to the next memory location
	MOV M,A	Store Sum in memory
	INX H	Increment pointer to the next memory location
	MOV M,C	Store Carry in memory
	HLT	End of Program

FLOWCHART


START
SET CARRY 'C' TO 00
SET POINTER TO GET DATA FROM MEMORY
GET DATA1 INTO ACCUMULATOR
INCREMENT POINTER
ADD DATA 2 IN MEMORY TO A
IF CARRY==1
YES
NO
C=01
STORE SUM
STORE CARRY
STOP

MNEMONICS:

ADDRESS	OPCODE	OPERAND	LOCATION	COMMAND
8000	21	01 82		LXI H, 8201
8003	0E	00		MVI C, 00
8005	7E			MOV A, M
8006	23			INX H
8007	86			ADD M
8008	D2	0C 80		JNC Ahead
800B	0C			INR C
800C	23		Ahead	INX H
800D	77			MOV M,A
800E	23			INX H
800F	71			MOV M,C
8010	76			HLT

INPUT & OUTPUT:

Location		Values
8200	INPUT 1	54
8201	INPUT 2	A1
8202	RESULT	F5
8203	CARRY	00

Location		Values
8200	INPUT 1	B3
8201	INPUT 2	69
8202	RESULT	1C
8203	CARRY	01

CALCULATION:

i) Data 1 : 54 - 0101 0100
Data 2 : A1 - 1010 0001
Sum : F5 - 1111 0101
Carry : 00

ii) Data 1 : B3 - 1011 0011
Data 2 : 69 - 0110 1001
Sum : 1C - 0001 1100
Carry : 01

RESULT: The program for adding two 8-bit hexadecimal numbers using memory pointer has been performed with different sets of data.

Date:
Experiment No:

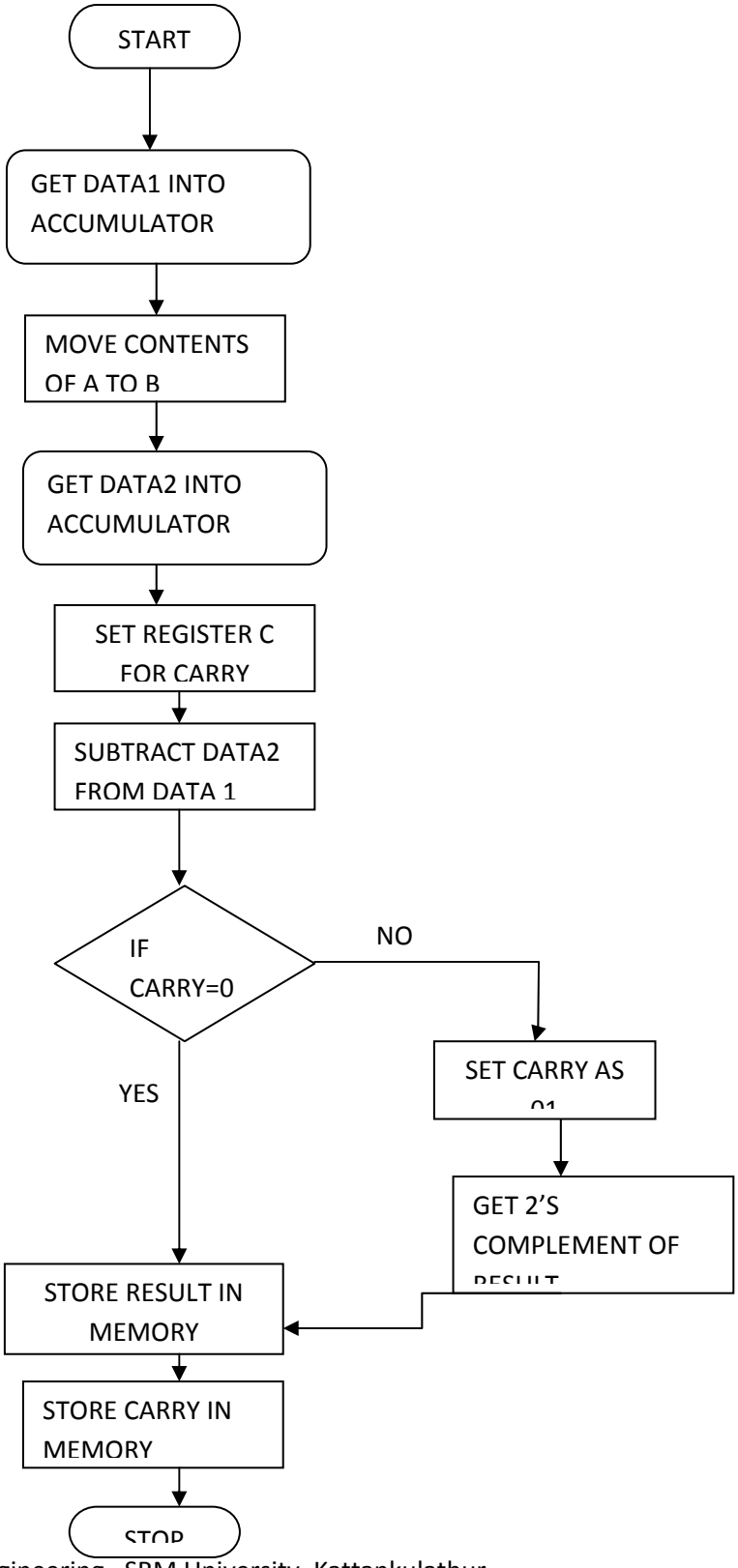
8-BIT SUBTRACTION USING ACCUMULATOR

AIM: To perform 8-bit Subtraction of two hexadecimal numbers using accumulator in 8085

PROGRAM:

	LDA 8201	Load the first data in the accumulator
	MOV B, A	Move contents of Accumulator to B
	LDA 8200	Load the second data in the accumulator
	MVI C, 00	Set C = '0' for carry
	SUB B	Subtract B from Accumulator & Store in Accumulator
	JNC Ahead	If carry is not set go Ahead
	INR C	if carry = '1' , increment register C to 01
	CMA	Get 2's complement for Accumulator
	ADD 01	Add one to accumulator
Ahead	STA 8202	Store result in given location
	MOV A,C	Move carry to Accumulator
	STA 8203	Store Carry in given location
	HLT	End of Program

FLOWCHART


START
GET DATA1 INTO ACCUMULATOR
MOVE CONTENTS OF A TO B
GET DATA2 INTO ACCUMULATOR
SET REGISTER C FOR CARRY
SUBTRACT DATA2 FROM DATA 1
IF CARRY=0
NO
YES
SET CARRY AS 01
GET 2'S COMPLEMENT OF RESULT
STORE RESULT IN MEMORY
STORE CARRY IN MEMORY
STOP

MNEMONICS:

ADDRESS	OPCODE	OPERAND	LOCATION	COMMAND
8000	3A	01 82		LDA 8201
8003	47			MOV B, A
8004	3A	00 82		LDA 8200
8007	0E	0		MVI C, 00
8009	90			SUB B
800A	D2	11 80		JNC Ahead
800D	0C			INR C
800E	2F			CMA
800F	C6	1		ADD 01
8011	32	02 82	Ahead	STA 8202
8014	79			MOV A,C
8015	32	03 82		STA 8203
8018	76			HLT

INPUT & OUTPUT:

i)

8200	INPUT 1	55
8201	INPUT 2	32
8202	DIFFERENCE	23
8203	BORROW	0

ii)

8200	INPUT 1	11
8201	INPUT 2	AA
8202	DIFFERENCE	99
8203	BORROW	1

CALCULATION:

i) Data 1 : 55 -0101 0101
Data 2 : 32 -0011 0010
Difference: 23 -0010 0011
Borrow: 00

ii) Data 1 : 11 -0001 0001
Data 2 : AA -1010 1010
Difference: 99 -1001 1001
Borrow: 01

RESULT: The program for subtracting two 8-bit hexadecimal numbers using 8085 was executed.

Date:

Experiment No:

8-BIT SUBTRACTION WITH MEMORY POINTER

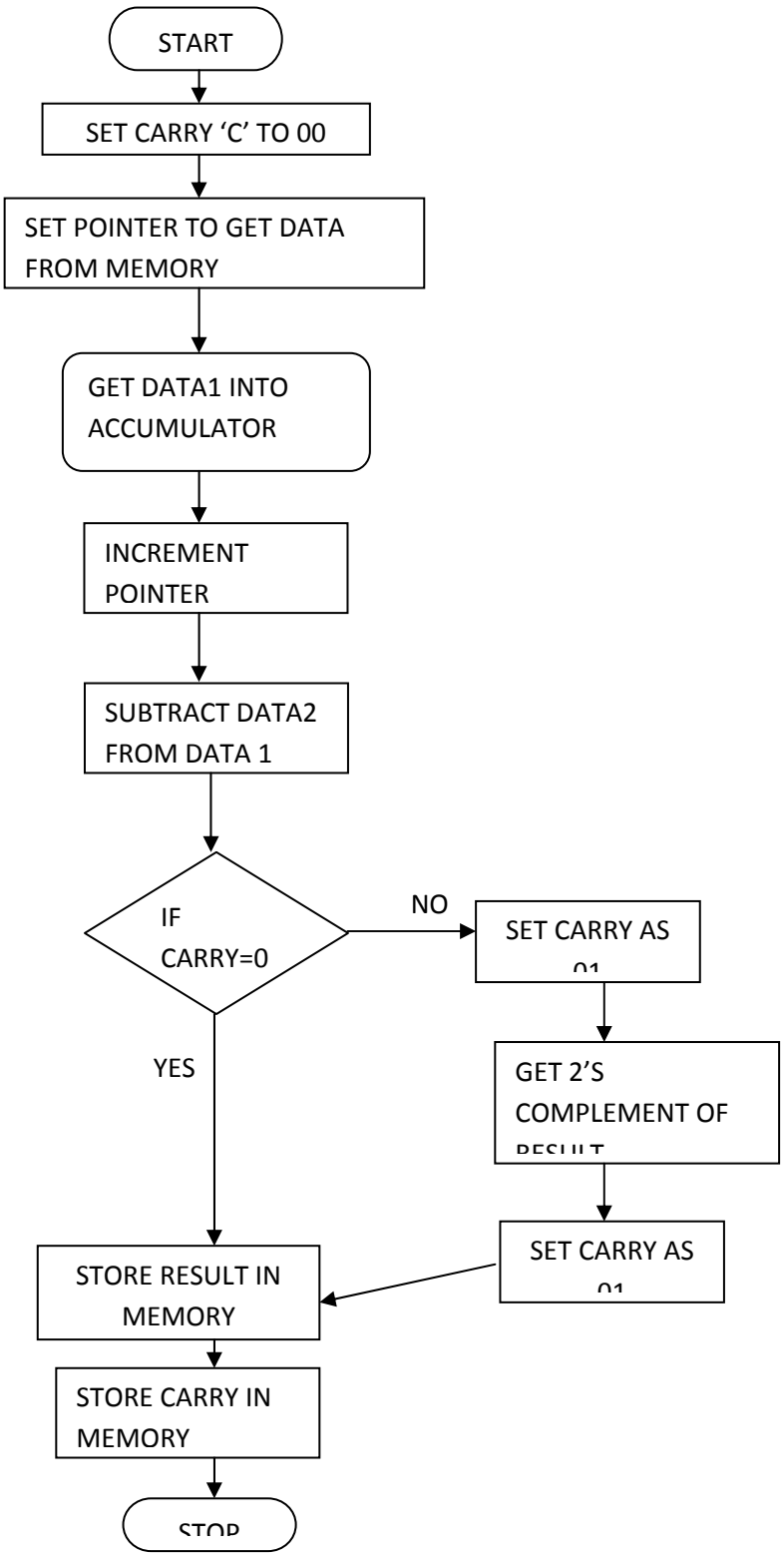
AIM: To perform 8-bit Subtraction of two hexadecimal numbers using memory pointer in 8085

PROGRAM:

Address	Program	Explanation
	LXI H, 8201	Setting pointer for data
	MVI C, 00	Clear register C to account for Carry
	MOV A, M	Get Data 1 into Accumulator
	INX H	Increment the memory pointer
	SUB M	Subtract Data 2 from Data 1 and store in Accumulator
	JNC Ahead	If carry==0 , go Ahead
	INR C	If carry==1, Increment register C by 1
	CMA	Get 2's complement for Accumulator data
	ADI 01	Add 1 to accumulator content
Ahead	INX H	Increment the memory pointer

	MOV M,A	Store Accumulator content (Result) to memory
	INX H	Increment the memory pointer
	MOV M,C	Store Register C (Carry) content to memory
	HLT	End of program

FLOWCHART


START
SET CARRY 'C' TO 00
SET POINTER TO GET DATA FROM MEMORY
GET DATA1 INTO ACCUMULATOR
INCREMENT POINTER
SUBTRACT DATA2 FROM DATA 1
IF CARRY=0
NO
SET CARRY AS 01
GET 2'S COMPLEMENT OF RESULT
SET CARRY AS 01
YES
STORE RESULT IN MEMORY
STORE CARRY IN MEMORY
STOP

MNEMONICS:

ADDRESS	OPCODE	OPERAND	LOCATION	COMMAND
8000	21	01 82		LXI H, 8201
8003	0E	00		MVI C, 00
8005	7E			MOV A, M
8006	23			INX H
8007	96			SUB M
8008	D2	0F 80		JNC Ahead
800B	0C			INR C
800C	2F			CMA
800D	C6	01		ADI 01
800F	23		Ahead	INX H
8010	77			MOV M,A
8011	23			INX H
8012	71			MOV M,C
8013	76			HLT

INPUT & OUTPUT:

i)

8200	INPUT 1	11
8201	INPUT 2	11
8202	DIFFERENCE	00
8203	BORROW	00

ii)

8200	INPUT 1	33
8201	INPUT 2	55
8202	DIFFERENCE	22
8203	BORROW	01

CALCULATION:

i) Data 1 : 11 -0001 0001

Data 2 : 11 -0001 0001

Difference: 00 -0010 0011

Borrow: 00

ii) Data 1 : 33 -0011 0011

Data 2 : 55 -0101 0101

2’ complement of 55: 1010 1010

+ 1 : 1010 1011

+ Data 1 (0011 0011): 1101 1110

2’s complement of the above number : 0010 0001

+ 01 : 0010 0010 = 22

Difference: 22 -0010 0010

Borrow: 01

RESULT: The program for subtracting two 8-bit hexadecimal numbers using memory pointer in 8085 was executed.

Date:
Experiment No:

8-BIT BCD ADDITION

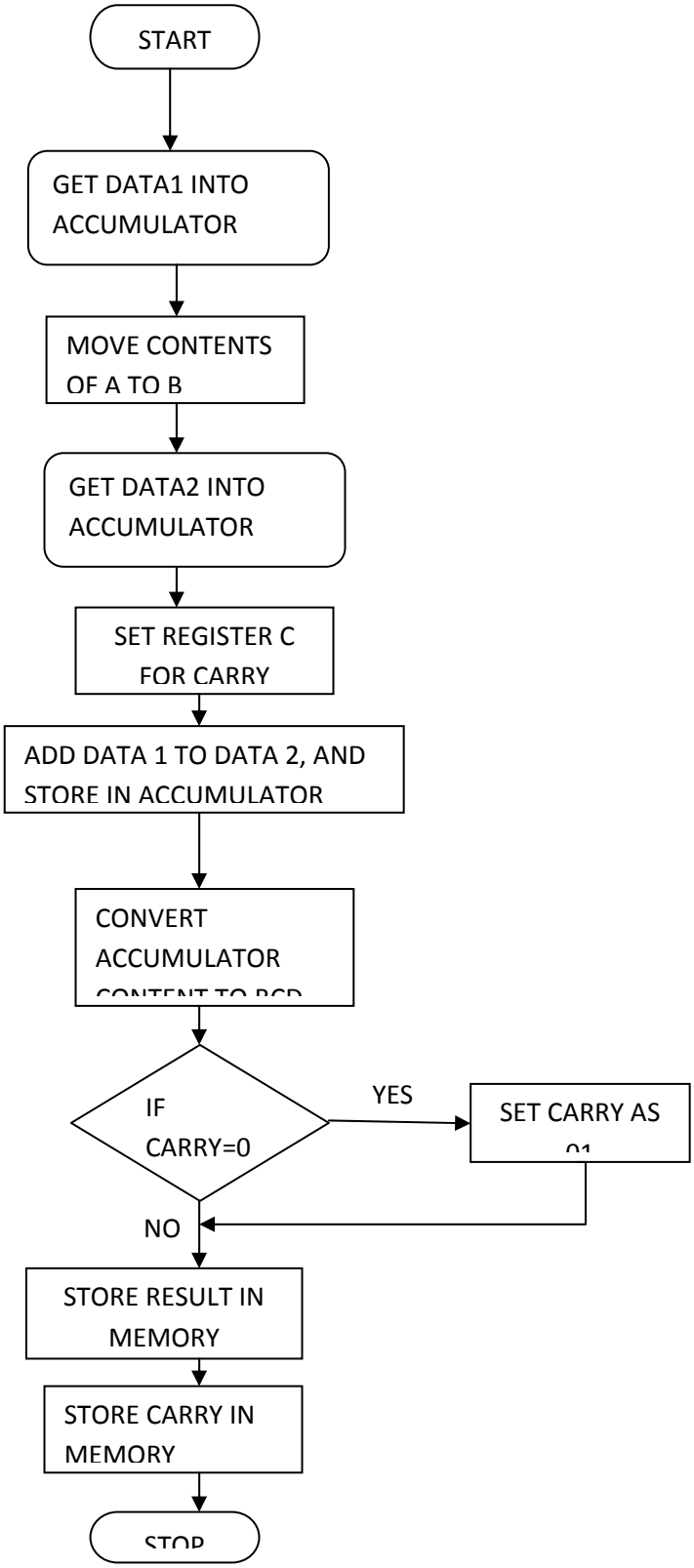
AIM: To perform addition of two 8-bit BCD numbers using 8085 microprocessor.

PROGRAM:

Address	Program	Explanation
	LDA 8200	Load Data 1 into accumulator
	MOV B,A	Move Accumulator contents to Register B
	LDA 8201	Load Data 2 into accumulator
	MVI C, 00	Clear register C to account for Carry
	ADD B	Add Data 2 to Data 1 and store in Accumulator
	DAA	Convert the accumulator value to BCD value
	JNC Ahead	If carry==0 , go Ahead
	INR C	If carry==1, Increment register C by 1
Ahead	STA 8203	Store Accumulator content (Result) to memory
	MOV A,C	Move contents of Register C to Accumulator

	STA 8204	Store Register C (Carry) content to memory
	HLT	End of program

FLOWCHART


START
GET DATA1 INTO ACCUMULATOR
MOVE CONTENTS OF A TO B
GET DATA2 INTO ACCUMULATOR
SET REGISTER C FOR CARRY
ADD DATA 1 TO DATA 2, AND STORE IN ACCUMULATOR
CONVERT ACCUMULATOR
IF CARRY=0
YES
SET CARRY AS
NO
STORE RESULT IN MEMORY
STORE CARRY IN MEMORY
STOP

MNEMONICS:

ADDRESS	OPCODE	OPERAND	LOCATION	COMMAND
8000	3A	00 82		LDA 8200
8003	47			MOV B,A
8004	3A	01 82		LDA 8201
8007	0E	0		MVI C, 00
8009	86			ADD B
800A	27			DAA
800B	D2	0F 80		JNC Ahead
800E	0C			INR C
800F	32	02 82	Ahead	STA 8203
8012	79			MOV A,C
8013	32	03 82		STA 8204
8016	76			HLT

INPUT & OUTPUT:

i)

8200	INPUT 1	13
8201	INPUT 2	4A
8202	SUM	63
8203	CARRY	0

ii)

8200	INPUT 1	13
8201	INPUT 2	16
8202	SUM	29
8203	CARRY	0

CALCULATION:

i) Data 1: 13 0001 0011
Data 2: 4A 0100 1010
Gives 0101 1101
After adding 06 0000 0110
Result: 63 0110 0011
Carry: 00

Note: If hexadecimal number is in units place 06 is added, if hexadecimal is in tens place 60 is added to sum by the DAA command.

RESULT: Program to add two BCD 8-bit numbers was performed using DAA command with 8085 microprocessor

Date:

Experiment No:

16 BIT ADDITION USING ACCUMULATOR

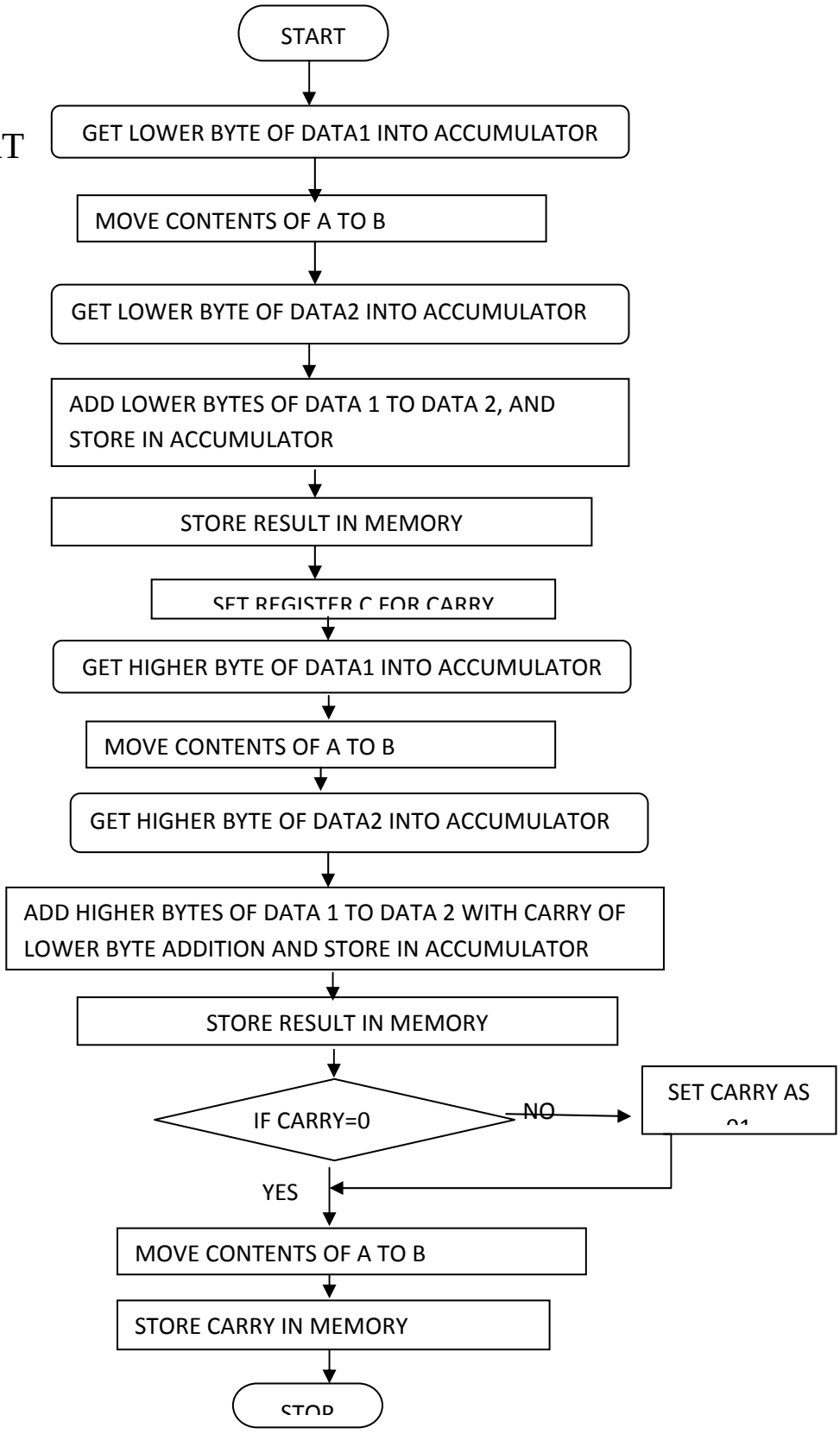
AIM: To add two 16 bit numbers (4 digits) using the accumulator with 8085 microprocessor.

PROGRAM:

Address	Program	Explanation
	LDA 8201	Get Data 1 (lower byte) into accumulator
	MOV B,A	Move Data to Register B from Accumulator
	LDA 8203	Get Data 2 (lower byte) into accumulator
	ADD B	Add Lower Bytes of Data 1 & Data 2
	STA 8205	Store content of Accumulator , Lower Byte Result in Memory
	MVI C, 00	Set C = '0' for carry
	LDA 8202	Get Data 1 (higher byte) into accumulator
	MOV B,A	Move Data to Register B from Accumulator
	LDA 8204	Get Data 2 (higher byte) into accumulator
	ADC B	Add Higher Bytes of Data 1 & Data 2 with carry of Lower Bytes
	STA 8206	Store content of Accumulator , Lower Byte

		Result in Memory
	JNC Ahead	If carry is not set go Ahead
	INR C	if carry = '1' , increment register C to 01
Ahead	MOV A,C	Move contents of Register C to Accumulator
	STA 8207	Store Register C (Carry) content to memory
	HLT	End of program

FLOWCHART


START
GET LOWER BYTE OF DATA1 INTO ACCUMULATOR
MOVE CONTENTS OF A TO B
GET LOWER BYTE OF DATA2 INTO ACCUMULATOR
ADD LOWER BYTES OF DATA 1 TO DATA 2, AND STORE IN ACCUMULATOR
STORE RESULT IN MEMORY
SET REGISTER C FOR CARRY
GET HIGHER BYTE OF DATA1 INTO ACCUMULATOR
MOVE CONTENTS OF A TO B
GET HIGHER BYTE OF DATA2 INTO ACCUMULATOR
ADD HIGHER BYTES OF DATA 1 TO DATA 2 WITH CARRY OF LOWER BYTE ADDITION AND STORE IN ACCUMULATOR
STORE RESULT IN MEMORY
IF CARRY=0
NO
SET CARRY AS
YES
MOVE CONTENTS OF A TO B
STORE CARRY IN MEMORY
STOP

MNEMONICS:

ADDRESS	OPCODE	OPERAND	LOCATION	COMMAND
8000	3A	01 82		LDA 8201
8003	47			MOV B,A
8004	3A	03 82		LDA 8203
8007	80			ADD B
8008	32	05 82		STA 8205
800B	0E	0		MVI C, 00
800D	3A	02 82		LDA 8202
8010	47			MOV B,A
8011	3A	04 82		LDA 8204
8014	88			ADC B
8015	32	06 82		STA 8206
8018	D2	1C 80		JNC Ahead
801B	0C			INR C
801C	79		Ahead	MOV A,C
801D	32	07 82		STA 8207
8020	76			HLT

INPUT & OUTPUT:

i) INPUT 1 4283
INPUT 2 2931
SUM 6BB4
CARRY 00

8201	83	Lower Byte of Data 1
8202	42	Higher Byte of Data 1
8203	31	Lower Byte of Data 2
8204	29	Higher Byte of Data 2
8205	B4	Lower Byte of Sum
8206	6B	Higher Byte of Sum
8207	0	Carry

ii)

INPUT 1 9312
INPUT 2 8856
SUM 1B68
CARRY 1

Lower Byte of Data 1	8201	12
Higher Byte of Data 1	8202	93
Lower Byte of Data 2	8203	56
Higher Byte of Data 2	8204	88

Lower Byte of Sum	8205	68
Higher Byte of Sum	8206	1B
Carry	8207	1

CALCULATION:

Lower Byte of Data 1: 83 1000 0011
Lower Byte of Data 2: 31 0011 0001
Lower Byte of Sum : B4 1011 0100
Higher Byte of Data 1: 42 0100 0010
Higher Byte of Data 2: 29 0010 1001
Higher Byte of Sum : 6B 0110 1011
Carry: 00

RESULT: Program to add two 16 bit numbers (4 digits) using the accumulator with 8085 microprocessor was executed

Date:

Experiment No:

16 BIT ADDITION USING REGISTER PAIR

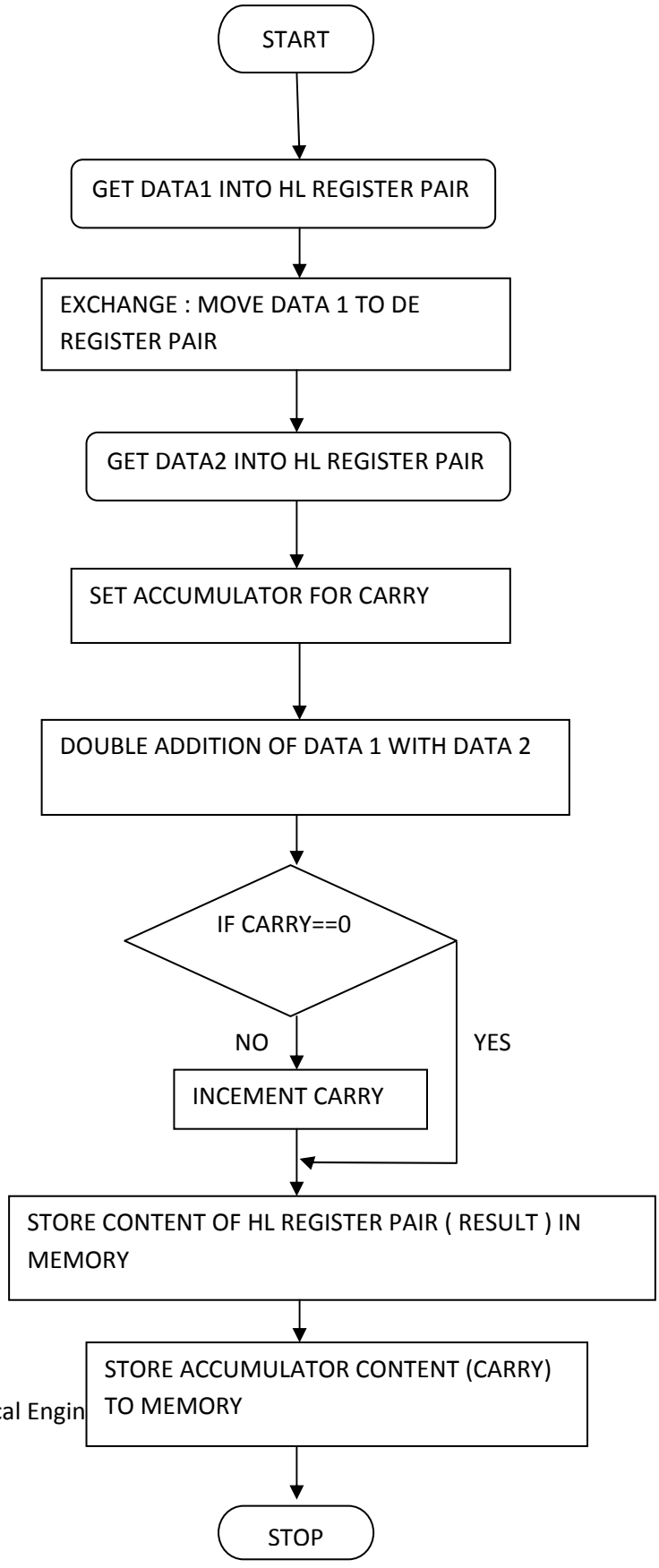
AIM: To add two 16 bit numbers (4 digits) using the register pairs in 8085 microprocessor.

PROGRAM:

Address	Program	Explanation
	LHLD 8201	Load HL register pair with data in 8201 & 8202
	XCHG	Exchange contents of HL with DE Register Pair
	LHLD 8203	Load HL register pair with data in 8203 & 8204
	MVI A, 00	Set Accumulator , A = '0' for carry
	DAD D	Double Addition
	JNC Ahead	If carry is not set go Ahead
	INR A	if carry = '1' , increment Accumulator to 01
Ahead	SHLD 8205	Store contents of HL Register pair in Memory

	STA 8207	Store Register C (Carry) content to memory
	HLT	End of program

FLOWCHART


START
GET DATA1 INTO HL REGISTER PAIR
EXCHANGE : MOVE DATA 1 TO DE REGISTER PAIR
GET DATA2 INTO HL REGISTER PAIR
SET ACCUMULATOR FOR CARRY
DOUBLE ADDITION OF DATA 1 WITH DATA 2
IF CARRY==0
NO
YES
INCEMENT CARRY
STORE CONTENT OF HL REGISTER PAIR (RESULT) IN MEMORY
STORE ACCUMULATOR CONTENT (CARRY) TO MEMORY
STOP

MNEMONICS:

ADDRESS	OPCODE	OPERAND	LOCATION	COMMAND
8000	2A	01 82		LHLD 8201
8003	E3			XCHG
8004	2A	03 82		LHLD 8203
8007	3E	0		MVI A, 00
8009	19			DAD D
800A	D2	0E 80		JNC Ahead
800D	3C			INR A
800E	22	05 82	Ahead	SHLD 8205
8011	32	07 82		STA 8207
8014	76			HLT

INPUT & OUTPUT:

INPUT 1 9437
INPUT 2 7259
SUM 0690
CARRY 01

Lower Byte of Data 1	8201	37

Higher Byte of Data 1	8202	94
Lower Byte of Data 2	8203	59
Higher Byte of Data 2	8204	72
Lower Byte of Sum	8205	90
Higher Byte of Sum	8206	6
Carry	8207	1

RESULT: Program to add two 16 bit numbers (4 digits) using the Register pairs in 8085 microprocessor was executed

Date:
Experiment No:

16 BIT HEXADECIMAL SUBTRACTION

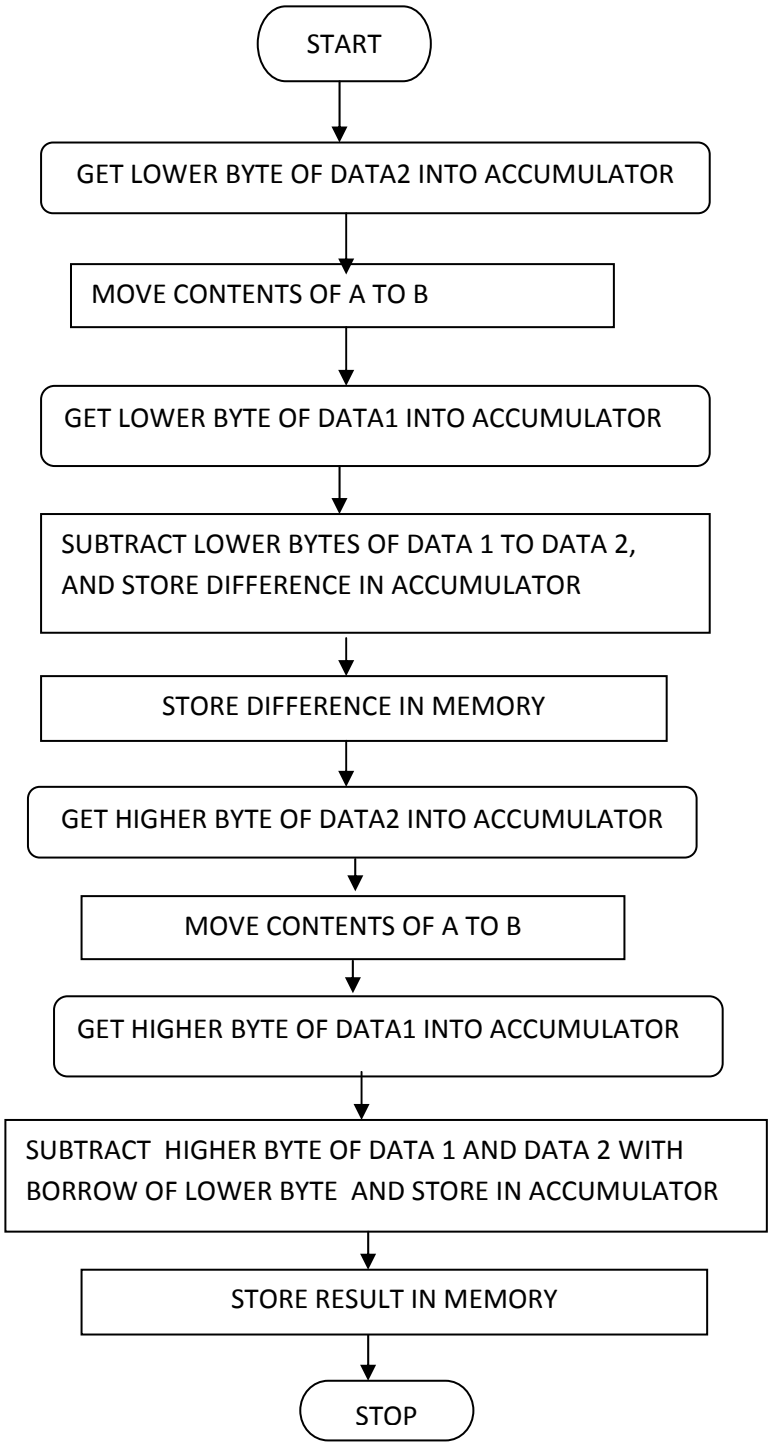
AIM: To subtract two 16-bit hexadecimal numbers using Accumulator with 8085 Microprocessor

PROGRAM:

Program	Explanation
LDA 8202	Load contents of Accumulator with Lower byte of Data 2
MOV B,A	Move contents of Accumulator to Register B
LDA 8200	Load contents of Accumulator with Lower byte of Data 1
SUB B	Subtract contents of B from Accumulator
STA 8204	Store Accumulator content (Lower byte of difference) to Memory
LDA 8203	Load contents of Accumulator with Higher byte of Data 2
MOV B,A	Move contents of Accumulator to Register B
LDA 8201	Load contents of Accumulator with Higher byte of Data 1

SBB B	Subtract contents of B from Accumulator along with Borrow, if any
STA 8205	Store Accumulator content (Higher byte of difference) to Memory
HLT	End of Program

FLOWCHART


START
GET LOWER BYTE OF DATA2 INTO ACCUMULATOR
MOVE CONTENTS OF A TO B
GET LOWER BYTE OF DATA1 INTO ACCUMULATOR
SUBTRACT LOWER BYTES OF DATA 1 TO DATA 2, AND STORE DIFFERENCE IN ACCUMULATOR
STORE DIFFERENCE IN MEMORY
GET HIGHER BYTE OF DATA2 INTO ACCUMULATOR
MOVE CONTENTS OF A TO B
GET HIGHER BYTE OF DATA1 INTO ACCUMULATOR
SUBTRACT HIGHER BYTE OF DATA 1 AND DATA 2 WITH BORROW OF LOWER BYTE AND STORE IN ACCUMULATOR
STORE RESULT IN MEMORY
STOP

MNEMONICS:

ADDRESS	OPCODE	OPERAND	COMMAND
8000	3A	02 82	LDA 8202
8003	47		MOV B,A
8004	3A	00 82	LDA 8200
8007	90		SUB B
8008	32	04 82	STA 8204
800B	3A	03 82	LDA 8203
800E	47		MOV B,A
800F	3A	01 82	LDA 8201
8012	98		SBB B
8013	32	05 82	STA 8205
8016	76		HLT

INPUT & OUTPUT:

i) INPUT 1 4398
INPUT 2 2931
Difference 5522
BORROW 00

8200	98	Lower Byte of Data 1
8201	43	Higher Byte of Data 1
8202	21	Lower Byte of Data 2
8203	46	Higher Byte of Data 2
8204	22	Lower Byte of Difference
8205	55	Higher Byte of Sum
8206	0	BORROW

ii)

INPUT 1	4531
INPUT 2	9576
SUM	AFB7
BORROW	00

Lower Byte of Data 1	8200	31
Higher Byte of Data 1	8201	45
Lower Byte of Data 2	8202	76
Higher Byte of Data 2	8203	95
Lower Byte of Difference	8204	B8
Higher Byte of Difference	8205	AF
BORROW	8206	01

CALCULATION:

Lower Byte of Data 1	:	43	0100 0011
Lower Byte of Data 2	:	21	0010 0001
Lower Byte of Difference	:	22	0010 0010
Higher Byte of Data 1	:	98	1001 1000
Higher Byte of Data 2	:	46	0100 0110
Higher Byte of Difference	:	55	0101 0101
Borrow	: 00		

RESULT: Program to subtract two 16 bit numbers (4 digits) using the accumulator with 8085 microprocessor was executed

Date:
Experiment No:

8 BIT MULTIPLICATION

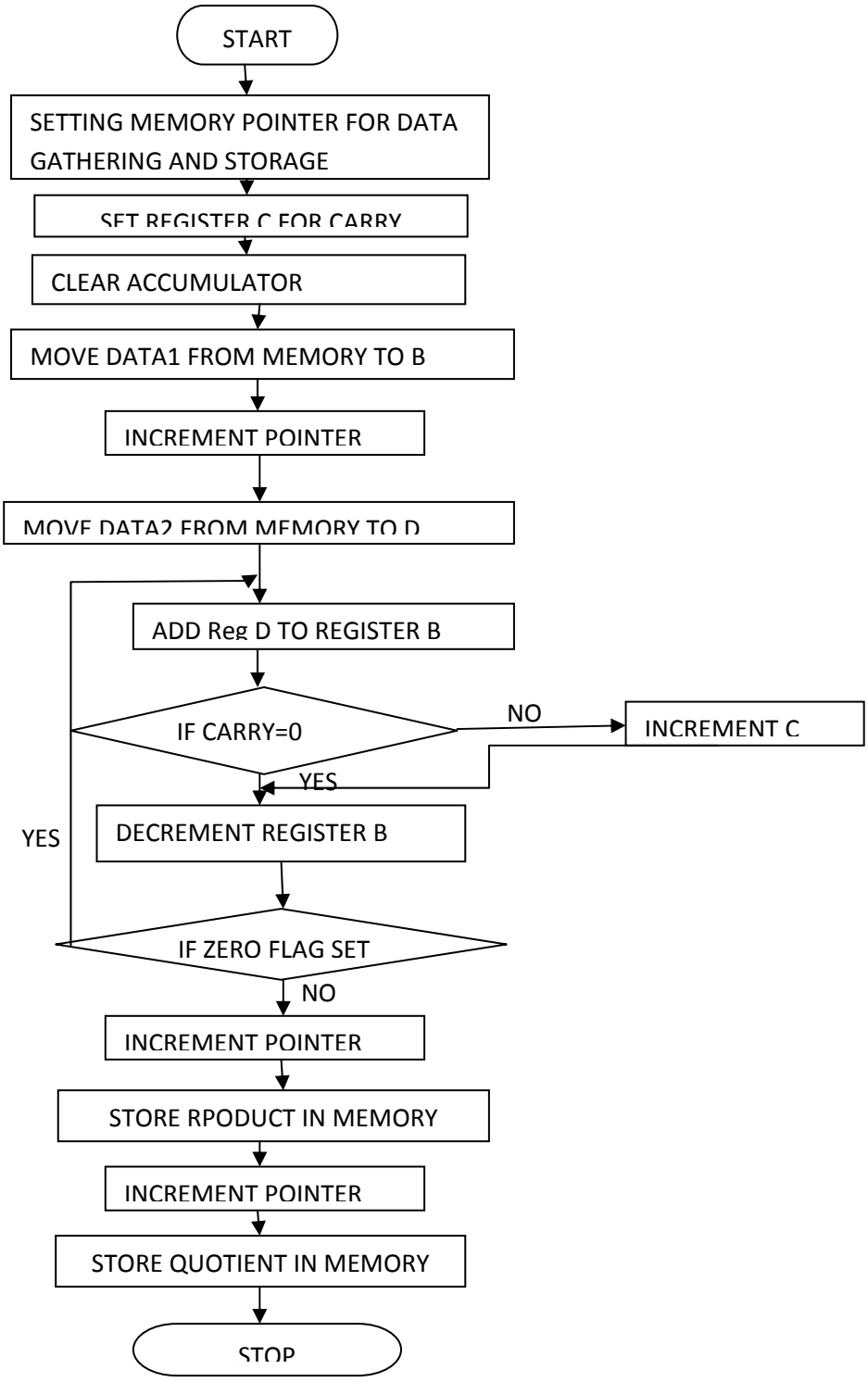
AIM: To multiply two 8-bit hexadecimal numbers using memory pointer with 8085 microprocessor

PROGRAM:

Address	Program	Explanation
	LXI H, 8201	Setting pointer for data
	MVI C, 00	Clear register C to account for Carry
	XRA A	Set Accumulator to '0' to account for product
	MOV B,M	Move Data 1 to Register B from Memory
	INX H	Increment the memory pointer
	MOV D,M	Move Data 2 to Register D from Memory
Repeat	ADD D	Add contents of Register D to Accumulator
	JNC Ahead	If carry==0 , go Ahead
	INR C	If carry==1, Increment register C by 1
Ahead	DCR B	Decrement Register B content by 1
	JNZ Repeat	If B is not Zero, got back to Repeat

	INX H	Increment the memory pointer
	MOV M,A	Store Accumulator content (Product) to memory
	INX H	Increment the memory pointer
	MOV M,C	Store Register C (Carry) content to memory
	HTL	End of program

FLOWCHART


START
SETTING MEMORY POINTER FOR DATA GATHERING AND STORAGE
SET REGISTER C FOR CARRY
CLEAR ACCUMULATOR
MOVE DATA1 FROM MEMORY TO B
INCREMENT POINTER
MOVE DATA2 FROM MEMORY TO D
ADD Reg D TO REGISTER B
IF CARRY=0
NO
INCREMENT C
YES
YES
DECREMENT REGISTER B
IF ZERO FLAG SET
NO
INCREMENT POINTER
STORE RPODUCT IN MEMORY
INCREMENT POINTER
STORE QUOTIENT IN MEMORY
STOP

MNEMONICS:

ADDRESS	OPCODE	OPERAND	LOCATION	COMMAND
8000	21	01 82		LXI H, 8201
8003	0E	0		MVI C, 00
8005	AF			XRA A
8006	46			MOV B,M
8007	23			INX H
8008	56			MOV D,M
8009	82		Repeat	ADD D
800A	D2	0E 80		JNC Ahead
800D	0C			INR C
800E	5		Ahead	DCR B
800F	C2	09 80		JNZ Repeat
8012	23			INX H
8013	77			MOV M,A
8014	23			INX H
8015	71			MOV M,C
8016	76			HLT

INPUT & OUTPUT:

8201	INPUT 1	5
8202	INPUT 2	3
8203	PRODUCT	F
8204	CARRY	0

CALCULATION:

```
      05     - 0000 0101
+     05     -0000 0101
----- 0A     - 0000 1010
+     05     -0000 0101
=     0F     - 0000 1111
```

RESULT: The program to multiply two 8-bit hexadecimal numbers using 8085 was executed.

Date:

Experiment No:

8-BIT DIVISION

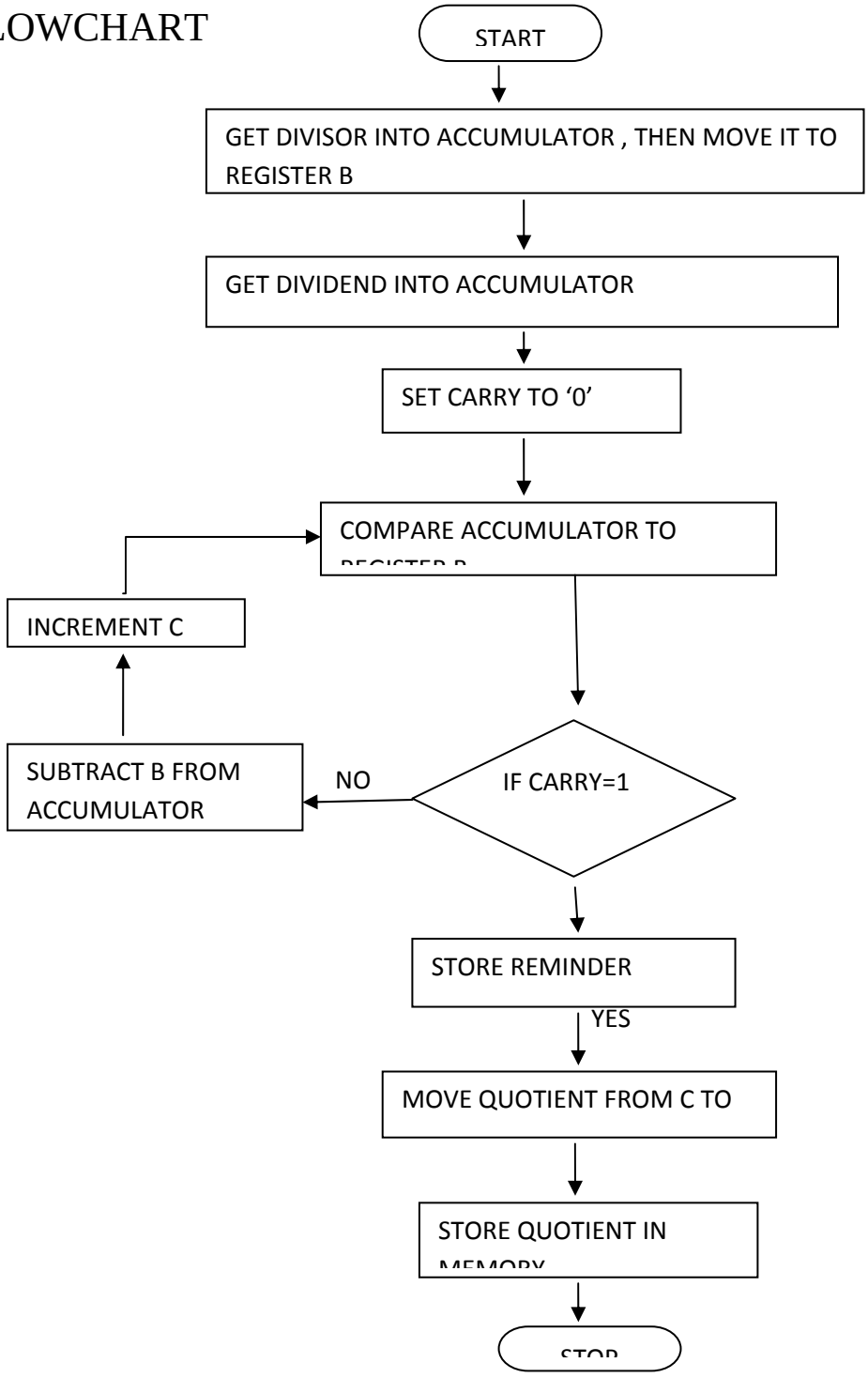
AIM: To divide two 8-bit hexadecimal numbers using 8085 microprocessor

PROGRAM:

Address	Program	Explanation
	LDA 8201	Load contents of Accumulator with Divisor from memory
	MOV B,A	Move contents of Accumulator to Register B
	LDA 8200	Load contents of Accumulator with Dividend from memory
	MVI C, 00	Clear register C to account for Carry
Again	CMP B	Compare Register B with Accumulator
	JC Store	If carry==1, jump to Store
	SUB B	If carry==0, subtract Register B from Accumulator
	INR C	Increment register C by 1
	JMP Again	Jump to Again , unconditionally
Store	STA 8203	Store Accumulator content (Reminder) to

		Memory
	MOV A,C	Move contents of Register C to Accumulator
	STA 8202	Store Accumulator content (Quotient) to Memory
	HLT	End of Program

FLOWCHART


START
GET DIVISOR INTO ACCUMULATOR , THEN MOVE IT TO REGISTER B
GET DIVIDEND INTO ACCUMULATOR
SET CARRY TO '0'
COMPARE ACCUMULATOR TO
INCREMENT C
SUBTRACT B FROM ACCUMULATOR
NO
IF CARRY=1
STORE REMINDER
YES
MOVE QUOTIENT FROM C TO
STORE QUOTIENT IN

MNEMONICS:

ADDRESS	OPCODE	OPERAND	LOCATION	COMMAND
8000	3A	01 82		LDA 8201
8003	47			MOV B,A
8004	3A	00 82		LDA 8200
8007	0E	0		MVI C, 00
8009	B8		Again	CMP B
800A	DA	12 80		JC Store
800D	90			SUB B
800E	0C			INR C
800F	C3	09 80		JMP Again
8012	32	03 82	Store	STA 8203
8015	79			MOV A,C
8016	32	02 82		STA 8202
8019	76			HLT

INPUT & OUTPUT:

i)

8200	DIVIDEND	6
8201	DIVISOR	3
8202	REMINDER	2
8203	QUOTIENT	0

ii)

8200	DIVIDEND	15
8201	DIVISOR	5
8202	REMINDER	4
8203	QUOTIENT	1

CALCULATION:

i) Dividend A=06 ; Divisor B=03

A – B = 06 -03 = 03 : While A>B , C = C+1

A – B= 03 – 03=00 : C=C+1

Reminder = 0 , Carry=02

RESULT: Thus program for dividing two hexadecimal numbers using 8085 Microprocessor has been executed

Date:

Experiment No:

ADDITION OF ‘n’ NUMBERS IN ARRAY

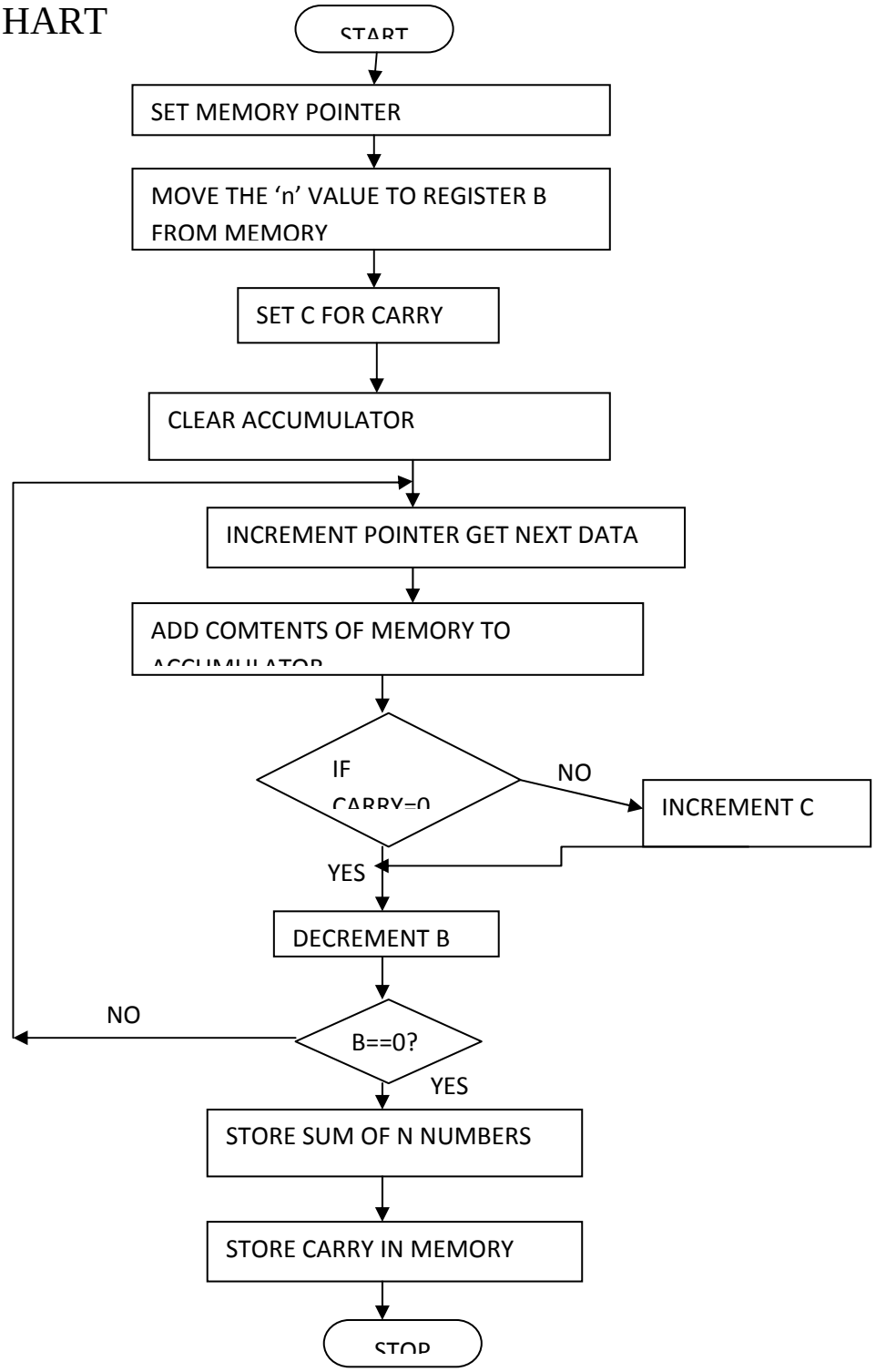
AIM: To add ‘n’ numbers in array and find the sum of those numbers using 8085 Microprocessor with memory pointer.

PROGRAM:

Address	Program	Explanation
	LXI H, 8201	Setting pointer for data
	MOV B,M	Move number of data to Register B from Memory
	MVI C, 00	Clear register C to account for Carry
	XRA A	Set Accumulator to '0' to account for product
Repeat	INX H	Increment the memory pointer
	ADD M	Add Contents of Memory to Accumulator & store in Accumulator
	JNC Ahead	If carry==0 , go Ahead
	INR C	Increment register C by 1, If carry ==1
Ahead	DCR B	Decrement Register B content by 1

	JNZ Repeat	If B is not Zero, got back to Repeat
	STA 8300	Store Accumulator content (Sum) to Memory
	MOV A,C	Move contents of Register C to Accumulator
	STA 8301	Store Accumulator content (Carry) to Memory
	HLT	End of Program

FLOWCHART


START
SET MEMORY POINTER
MOVE THE 'n' VALUE TO REGISTER B FROM MEMORY
SET C FOR CARRY
CLEAR ACCUMULATOR
INCREMENT POINTER GET NEXT DATA
ADD COMTENTS OF MEMORY TO
IF
NO
INCREMENT C
YES
DECREMENT B
NO
B==0?
YES
STORE SUM OF N NUMBERS
STORE CARRY IN MEMORY
STOP

MNEMONICS:

ADDRESS	OPCODE	OPERAND	LOCATION	COMMAND
8000	21	01 82		LXI H, 8201
8003	46			MOV B,M
8004	0E	0		MVI C, 00
8006	AF			XRA A
8007	23		Repeat	INX H
8008	86			ADD M
8009	D2	0D 80		JNC Ahead
800C	0C			INR C
800D	5		Ahead	DCR B
800E	C2	780		JNZ Repeat
8011	32	00 83		STA 8300
8014	71			MOV A,C
8015	32	01 83		STA 8301
8018	76			HLT

INPUT & OUTPUT:

8200	No. of Elements	6
8201	DATA 1	2
8202	DATA 2	4
8203	DATA 3	6
8204	DATA 4	8
8205	DATA 5	0A
8206	DATA 6	0C
8300	TOTAL	2A
8301	CARRY	0

CALCULATION:

DATA 1	:	02	- 0000 0010
DATA 2	:	04	- 0000 0100
	+	06	- 0000 0110
DATA 3	:	06	- 0000 0110
	+	0C	- 0000 1100
DATA 4	:	08	- 0000 1000
	+	14	- 0001 0100
DATA 5	:	0A	- 0000 1010
	+	1E	- 0001 1110
DATA 6	:	0C	- 0000 1100
TOTAL	:	2A	- 0010 1010
CARRY	:	00	

RESULT: Program to add ‘n’ numbers in array and find the sum of those numbers using 8085 Microprocessor with memory pointer was executed.

Date:
Experiment No:

SMALLEST IN ARRAY

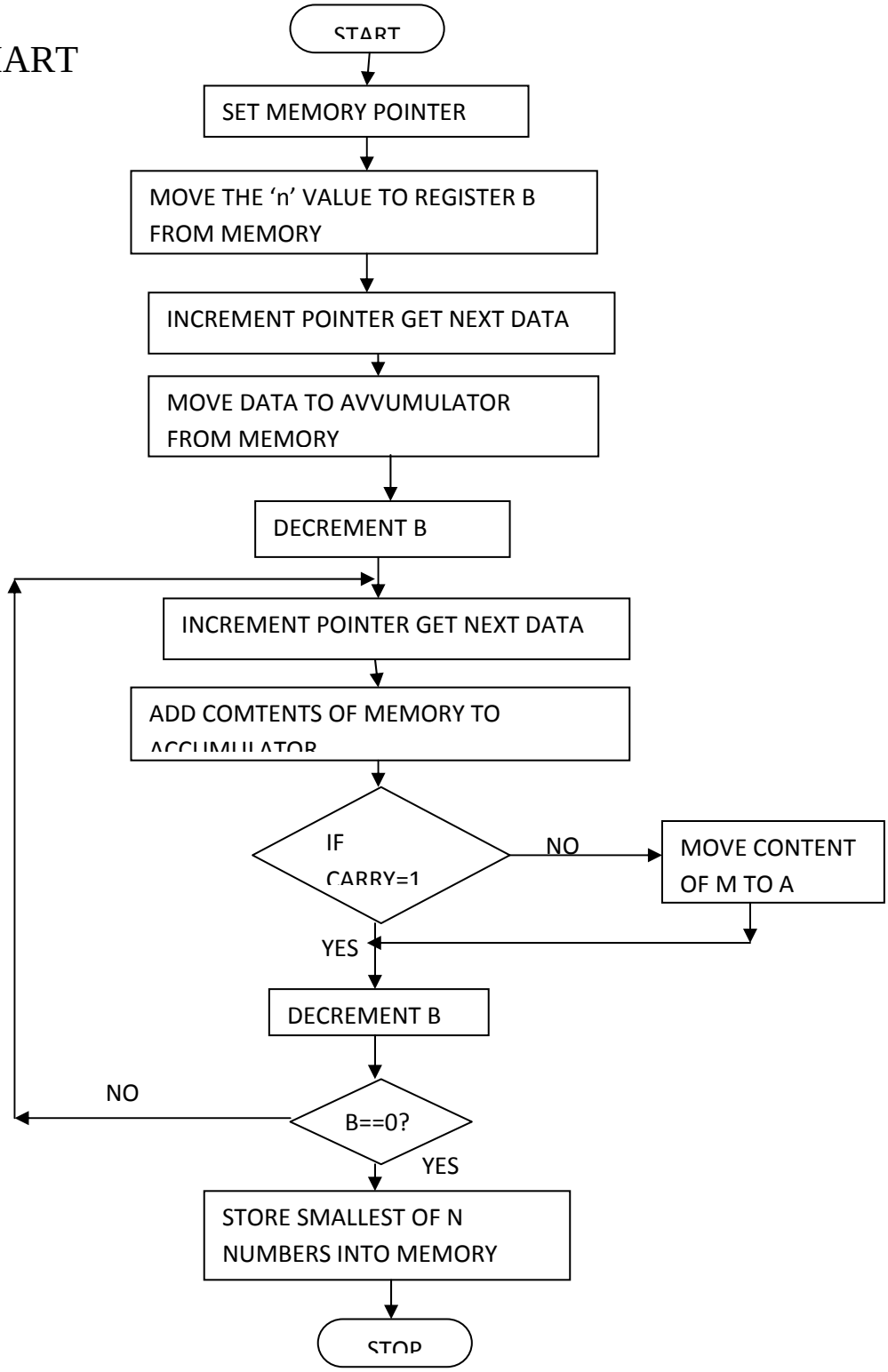
AIM: To find the smallest element in an array of size ‘n’ using 8085 Microprocessor.

PROGRAM:

Address	Program	Explanation
	LXI H, 8200	Setting pointer for data
	MOV B,M	Move number of data to Register B from Memory
	INX H	Increment the memory pointer
	MOV A,M	Move contents of memory to Accumulator
	DCR B	Decrement Register B content by 1
Loop	INX H	Increment the memory pointer
	CMP M	Compare contents of memory with Accumulator
	JC Ahead	If carry==1 , go Ahead i.e., number is larger
	MOV A,M	Move contents of memory to Accumulator

Ahead	DCR B	Decrement Register B content by 1
	JNZ Loop	If Register B is not equal to '0', go to Loop
	STA 8300	Store Accumulator content (Smallest No) to Memory
	HLT	End of Program

FLOWCHART


START
SET MEMORY POINTER
MOVE THE 'n' VALUE TO REGISTER B FROM MEMORY
INCREMENT POINTER GET NEXT DATA
MOVE DATA TO AVVUMULATOR FROM MEMORY
DECREMENT B
INCREMENT POINTER GET NEXT DATA
ADD COMTENTS OF MEMORY TO
IF
NO
MOVE CONTENT OF M TO A
YES
DECREMENT B
NO
B==0?
YES
STORE SMALLEST OF N NUMBERS INTO MEMORY
STOP

MNEMONICS:

ADDRESS	OPCODE	OPERAND	LOCATION	COMMAND
8000	21	00 82		LXI H, 8200
8003	46			MOV B,M
8004	23			INX H
8005	7E			MOV A,M
8006	5			DCR B
8007	23		Loop	INX H
8008	BE			CMP M
8009	DA	0D 80		JC Ahead
800C	7E			MOV A,M
800D	5		Ahead	DCR B
800E	C2	07 80		JNZ Loop
8011	32	00 83		STA 8300
8014	76			HLT

INPUT & OUTPUT:

8200	No. of Elements	6
8201	DATA 1	04
8202	DATA 2	03
8203	DATA 3	01
8204	DATA 4	02
8205	DATA 5	05
8206	DATA 6	0C
8300	TOTAL	01

CALCULATION:

B =6

i) A=04 ; B=5
ii) M=03 , Compare M with A: Carry=0
A=03 ; B=4
iii) M=01, Compare M with A: Carry=0
A=01; B=3
iv)M=02, Compare M with A: Carry=1, B=2
v) M=05, Compare M with A: Carry=1, B=1
vi)M=0C, Compare M with A: Carry=1, B=0

Hence, A=01 Smallest number

RESULT: Program to find the smallest element in an array of size ‘n’ using 8085 Microprocessor has been executed.

Date:
Experiment No:

LARGEST IN ARRAY

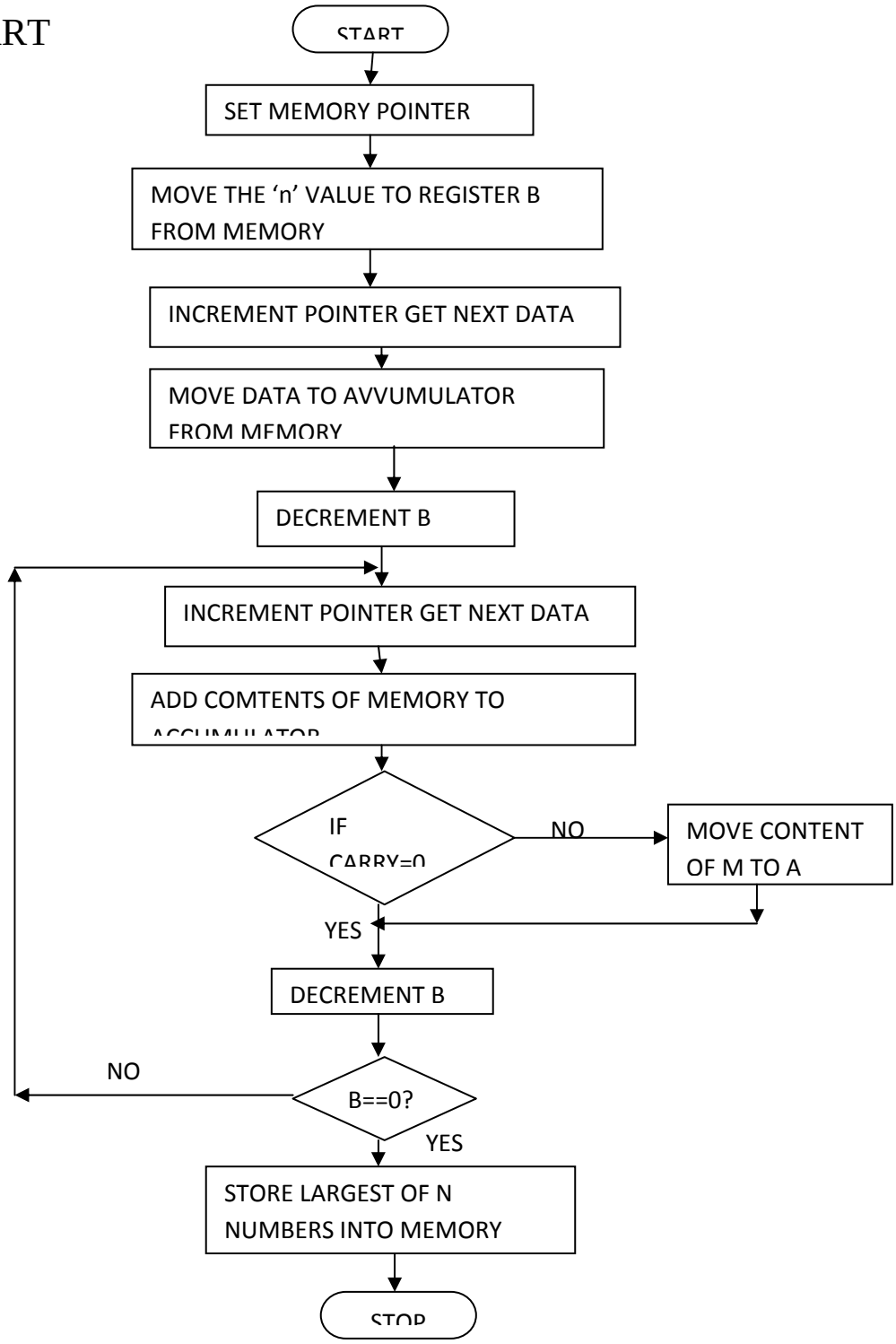
AIM: To find the Largest element in an array of size ‘n’ using 8085 Microprocessor.

PROGRAM:

Address	Program	Explanation
	LXI H, 8200	Setting pointer for data
	MOV B,M	Move number of data to Register B from Memory
	INX H	Increment the memory pointer
	MOV A,M	Move contents of memory to Accumulator
	DCR B	Decrement Register B content by 1
Loop	INX H	Increment the memory pointer
	CMP M	Compare contents of memory with Accumulator
	JNC Ahead	If carry==0, go Ahead i.e., number is smaller
	MOV A,M	Move contents of memory to Accumulator

Ahead	DCR B	Decrement Register B content by 1
	JNZ Loop	If Register B is not equal to '0', go to Loop
	STA 8300	Store Accumulator content (Sum) to Memory
	HLT	End of Program

FLOWCHART


START
SET MEMORY POINTER
MOVE THE 'n' VALUE TO REGISTER B FROM MEMORY
INCREMENT POINTER GET NEXT DATA
MOVE DATA TO AVVUMULATOR FROM MEMORY
DECREMENT B
INCREMENT POINTER GET NEXT DATA
ADD COMTENTS OF MEMORY TO
IF
NO
MOVE CONTENT OF M TO A
YES
DECREMENT B
NO
B==0?
YES
STORE LARGEST OF N NUMBERS INTO MEMORY
STOP

MNEMONICS:

ADDRESS	OPCODE	OPERAND	LOCATION	COMMAND
8000	21	00 82		LXI H, 8200
8003	46			MOV B,M
8004	23			INX H
8005	7E			MOV A,M
8006	5			DCR B
8007	23		Loop	INX H
8008	BE			CMP M
8009	D2	0D 80		JNC Ahead
800C	7E			MOV A,M
800D	5		Ahead	DCR B
800E	C2	07 80		JNZ Loop
8011	32	00 83		STA 8300
8014	76			HLT

INPUT & OUTPUT:

8200	No. of Elements	6
8201	DATA 1	04
8202	DATA 2	03
8203	DATA 3	01
8204	DATA 4	22
8205	DATA 5	05
8206	DATA 6	0C
8300	TOTAL	01

CALCULATION:

B =6

i) A=04 ; B=5
ii) M=03 , Compare M with A: Carry=1 B=4
iii) M=01, Compare M with A: Carry=1 ; B=3
iv)M=22, Compare M with A: Carry=0,
A=22, B=2
v) M=05, Compare M with A: Carry=1, B=1
vi)M=0C, Compare M with A: Carry=1, B=0

Hence, A=22 Largest number

RESULT: Program to find the Largest element in an array of size ‘n’ using 8085 Microprocessor has been executed

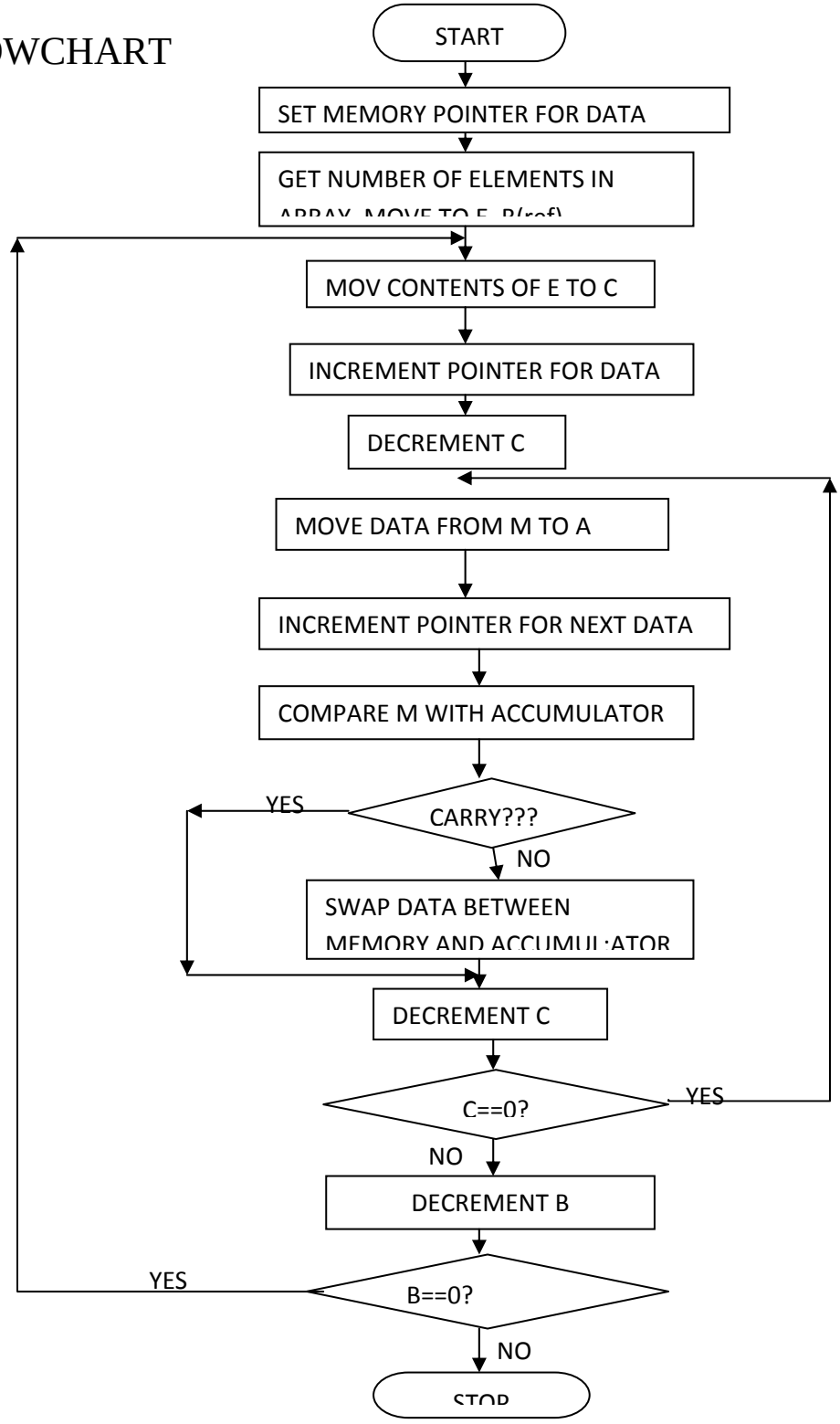
Date:
Experiment No:

SORTING IN ASCENDING ORDER

AIM: To Sort an array of ‘n’ element in ascending order PROGRAM:

	LXI H, 8200	Setting pointer for data
	MOV E,M	Move contents of memory(size of array) to Register E
	MOV B,M	Move number of data to Register B from Memory
Next	MOV C,E	Move register E content to Register C
	INX H	Increment the memory pointer
	DCR C	Decrement Register C by 1
Compare	MOV A,M	Move contents of memory to Accumulator
	INX H	Increment the memory pointer
	CMP M	Compare contents of memory with Accumulator
	JC Ahead	Jump on Carry to Ahead
	MOV D,M	Move contents of Memory to Register D
	MOV M,A	Move contents of Accumulator to Memory
	DCX H	Decrement the memory pointer
	MOV M,D	Move contents of Register D to Memory
	INX H	Increment the memory pointer
Ahead	DCR C	Decrement Register C by 1
	JNZ Compare	If Register C is not equal to '0', go to Compare
	DCR B	Decrement Register B content by 1
	JNZ Next	If Register B is not equal to '0', go to Next
	HLT	End of Program

FLOWCHART


START
SET MEMORY POINTER FOR DATA
GET NUMBER OF ELEMENTS IN
MOV CONTENTS OF E TO C
INCREMENT POINTER FOR DATA
DECREMENT C
MOVE DATA FROM M TO A
INCREMENT POINTER FOR NEXT DATA
COMPARE M WITH ACCUMULATOR
CARRY???
YES
NO
SWAP DATA BETWEEN
DECREMENT C
C==0?
YES
NO
DECREMENT B
B==0?
YES
NO
STOP

MNEMONICS:

ADDRESS	OPCODE	OPERAND	LOCATION	COMMAND
8000	21	00 82		LXI H, 8200
8003	5E			MOV E,M
8004	46			MOV B,M
8005	4B		Next	MOV C,E
8006	23			INX H
8007	0D			DCR C
8008	7E		Compare	MOV A,M
8009	23			INX H
800A	BE			CMP M
800B	DA	13 80		JC Ahead
800E	56			MOV D,M
800F	77			MOV M,A
8010	2B			DCX H
8011	72			MOV M,D
8012	23			INX H
8013	0D		Ahead	DCR C
8014	C2	08 80		JNZ Compare
8017	05			DCR B
8018	C2	05 80		JNZ Next
801B	76			HLT

INPUT & OUTPUT:

8200	No. of Elements	6
8201	DATA 1	5
8202	DATA 2	8
8203	DATA 3	3
8204	DATA 4	6
8205	DATA 5	7
8206	DATA 6	15
After Sorting		
8201	DATA 1	3

8202	DATA 2	5
8203	DATA 3	6
8204	DATA 4	7
8205	DATA 5	8
8206	DATA 6	15

8200	No. of Elements	05
8201	DATA 1	1A
8202	DATA 2	0A
8203	DATA 3	5C
8204	DATA 4	18
8205	DATA 5	23
After Sorting		
8200	No. of Elements	05
8201	DATA 1	0A
8202	DATA 2	1A
8203	DATA 3	18
8204	DATA 4	23
8205	DATA 5	5C

RESULT: Program to Sort an array of ‘n’ element in ascending order using 8085 Microprocessor was executed.

Date:
Experiment No:

SORTING IN DESCENDING ORDER

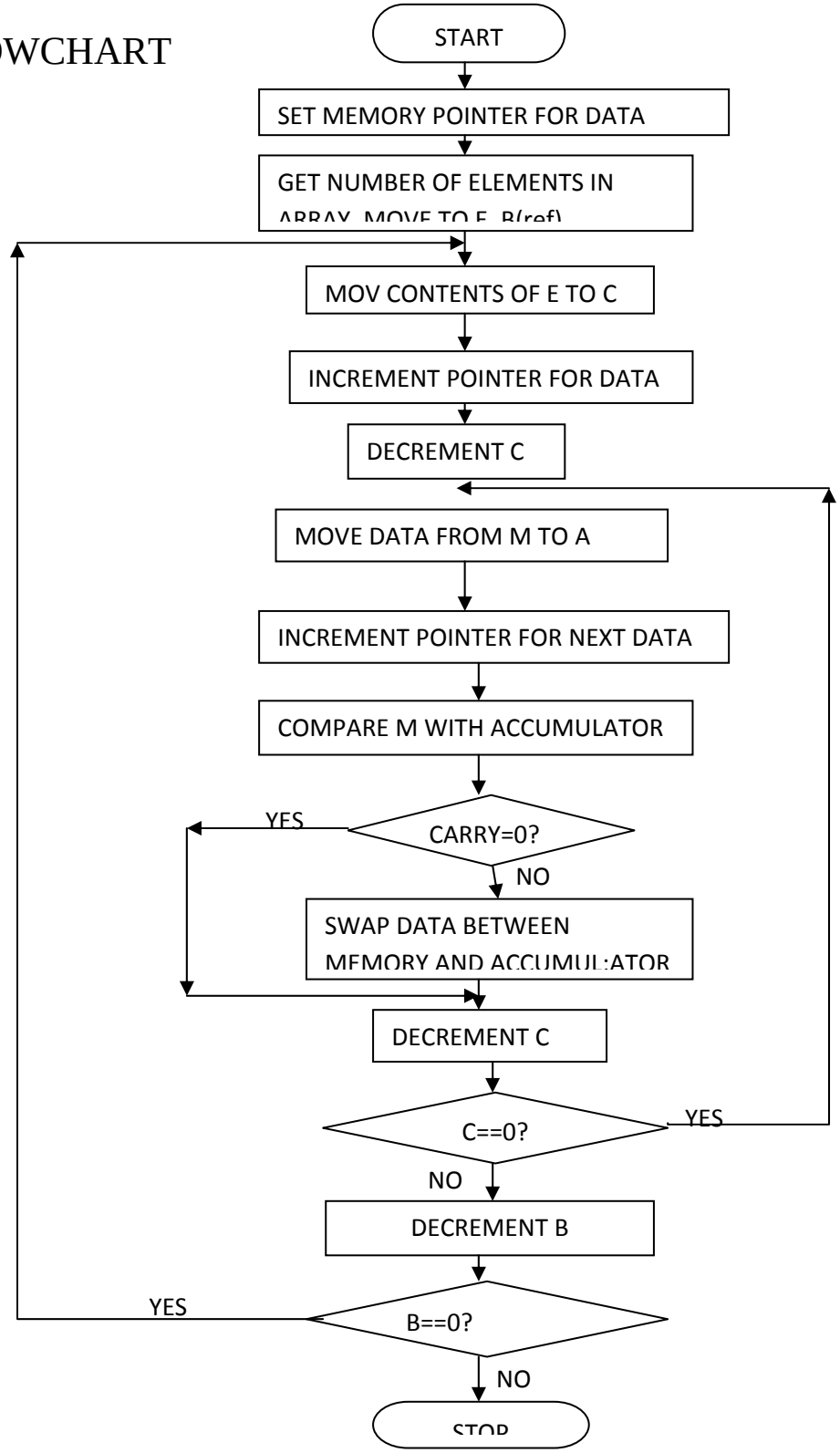
AIM: To Sort an array of ‘n’ element in ascending order using 8085 Microprocessor

PROGRAM:

Address	Program	Explanation
	LXI H, 8200	Setting pointer for data
	MOV E,M	Move contents of memory(size of array) to Register E
	MOV B,M	Move number of data to Register B from Memory
Next	MOV C,E	Move register E content to Register C
	INX H	Increment the memory pointer
	DCR C	Decrement Register C by 1
Compare	MOV A,M	Move contents of memory to Accumulator
	INX H	Increment the memory pointer
	CMP M	Compare contents of memory with Accumulator
	JNC Ahead	Jump on NO Carry to Ahead
	MOV D,M	Move contents of Memory to Register D
	MOV M,A	Move contents of Accumulator to Memory
	DCX H	Decrement the memory pointer
	MOV M,D	Move contents of Register D to Memory
	INX H	Increment the memory pointer
Ahead	DCR C	Decrement Register C by 1

	JNZ Compare	If Register C is not equal to '0', go to Compare
	DCR B	Decrement Register B content by 1
	JNZ Next	If Register B is not equal to '0', go to Next
	HLT	End of Program

FLOWCHART


START
SET MEMORY POINTER FOR DATA
GET NUMBER OF ELEMENTS IN
MOV CONTENTS OF E TO C
INCREMENT POINTER FOR DATA
DECREMENT C
MOVE DATA FROM M TO A
INCREMENT POINTER FOR NEXT DATA
COMPARE M WITH ACCUMULATOR
CARRY=0?
YES
NO
SWAP DATA BETWEEN
DECREMENT C
C==0?
YES
NO
DECREMENT B
B==0?
YES
NO
STOP

MNEMONICS:

ADDRESS	OPCODE	OPERAND	LOCATION	COMMAND
8000	21	00 82		LXI H, 8200
8003	5E			MOV E,M
8004	46			MOV B,M
8005	4B		Next	MOV C,E
8006	23			INX H
8007	0D			DCR C
8008	7E		Compare	MOV A,M
8009	23			INX H
800A	BE			CMP M
800B	D2	13 80		JC Ahead
800E	56			MOV D,M
800F	77			MOV M,A
8010	2B			DCX H
8011	72			MOV M,D
8012	23			INX H
8013	0D		Ahead	DCR C
8014	C2	08 80		JNZ Compare
8017	05			DCR B
8018	C2	05 80		JNZ Next
801B	76			HLT

INPUT & OUTPUT:

8200	No. of Elements	6
8201	DATA 1	06
8202	DATA 2	18
8203	DATA 3	1B
8204	DATA 4	2C
8205	DATA 5	1A
8206	DATA 6	15

After Sorting		
8201	DATA 1	2C
8202	DATA 2	1B
8203	DATA 3	1A
8204	DATA 4	18
8205	DATA 5	15
8206	DATA 6	06

8200	No. of Elements	05
8201	DATA 1	1A
8202	DATA 2	0A
8203	DATA 3	5C
8204	DATA 4	18
8205	DATA 5	23
After Sorting		
8200	No. of Elements	05
8201	DATA 1	5C
8202	DATA 2	23
8203	DATA 3	1A
8204	DATA 4	18
8205	DATA 5	0A

RESULT: Program to Sort an array of ‘n’ element in descending order using 8085 Microprocessor was executed.

Date:
Experiment No:

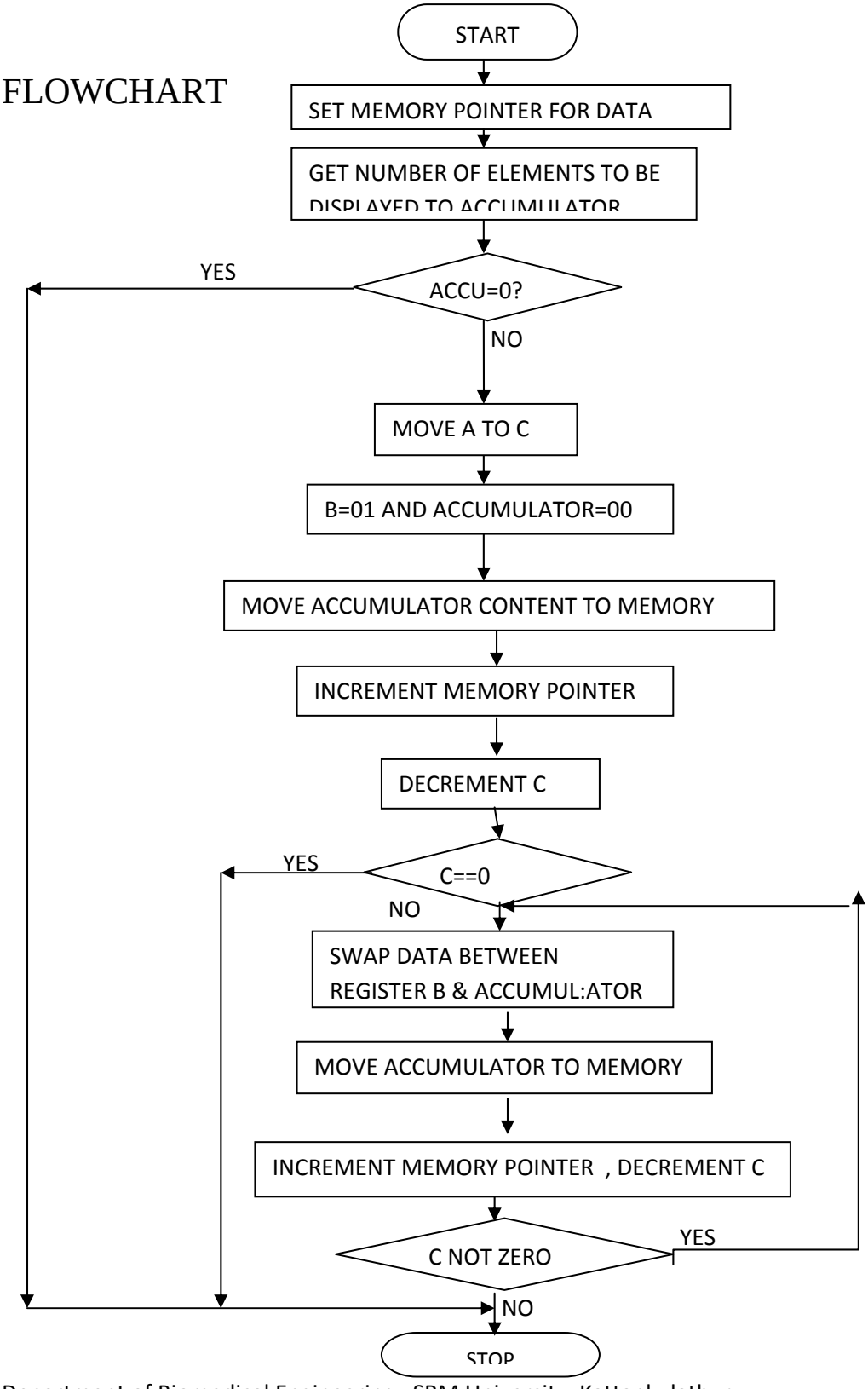
FIBONACCI SERIES

AIM: To display ‘n’ elements of the Fibonacci series using 8085 Microprocessor

PROGRAM:

Address	Program	Explanation
	LXI H, 8300	Setting pointer for storing/ displaying data
	LDA 8200	Load contents of Accumulator with no. of numbers to be generated
	CPI 00	Check if 00
	JZ End	If Zero , end program - no display
	MOV C,A	Move contents of Accumulator to Register C
	MVI B, 01	Move '01' (second number in Fibonacci Series) to Register B
	MVI A,00	Move '00' (First number in Fibonacci Series) to Accumulator
	MOV M,A	Move contents of Accumulator to Memory
	INX H	Increment the memory pointer
	DCR C	Decrement Register C by 1
	JZ End	If Zero , end program
Again	MOV D,A	Move contents of Accumulator to Register D
	ADD B	Add contents of Register B to Accumulator
	MOV B,D	Move contents of Register D to Register B
	MOV M,A	Move contents of Accumulator to Memory
	INX H	Increment the memory pointer
	DCR C	Decrement Register C by 1
	JNZ Again	If Register C is not equal to '0', go to Again
end	RST 1	Reset command

FLOWCHART


START
SET MEMORY POINTER FOR DATA
GET NUMBER OF ELEMENTS TO BE
DISPLAYED TO ACCUMULATOR
YES
ACCU=0?
NO
MOVE A TO C
B=01 AND ACCUMULATOR=00
MOVE ACCUMULATOR CONTENT TO MEMORY
INCREMENT MEMORY POINTER
DECREMENT C
YES
C==0
NO
SWAP DATA BETWEEN
REGISTER B & ACCUMUL:ATOR
MOVE ACCUMULATOR TO MEMORY
INCREMENT MEMORY POINTER , DECREMENT C
YES
C NOT ZERO
NO
STOP

MNEMONICS:

ADDRESS	OPCODE	OPERAND	LOCATION	COMMAND
8000	21	00 83		LXI H, 8300
8003	3A	00 82		LDA 8200
8006	FE	00		CPI 00
8008	CA	1F 80		JZ End
800B	4F			MOV C,A
800C	0B	01		MVI B, 01
800E	3E	00		MVI A,00
8010	77			MOV M,A
8011	23			INX H
8012	0D			DCR C
8013	CA	1F 80		JZ End
8016	57		Again	MOV D,A
8017	80			ADD B
8018	42			MOV B,D
8019	77			MOV M,A
801A	23			INX H
801B	0D			DCR C
801C	C2	16 80		JNZ Again
801F	CF		end	RST 1

INPUT & OUTPUT:

NO. of elements (8200)=06

8201	0
8202	1
8203	1
8204	2
8205	3
8206	5

CALCULATION:

Number of Elements in Series =06

```
    00
    01
+   01
    01
+   02
    01
+   03
    02
+   05
```

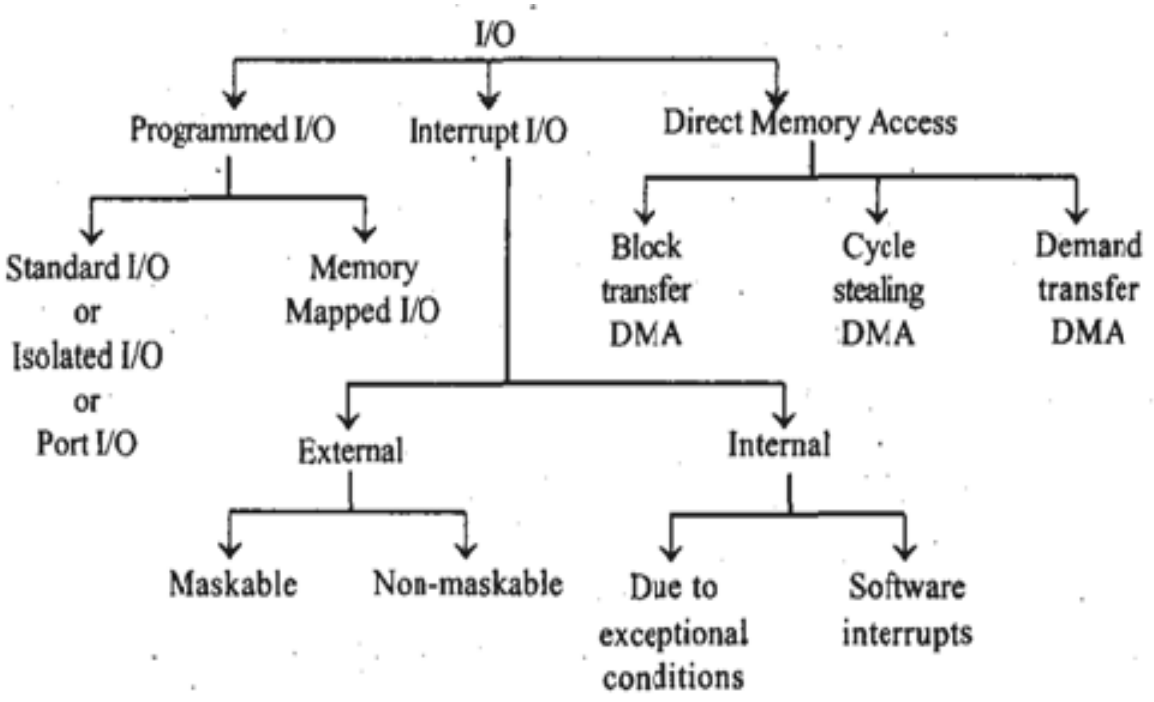
RESULT: Program to display ‘n’ elements of the Fibonacci series using 8085 Microprocessor was executed

I/O INTERFACING WITH 8085

I/O STRUCTURE OF A TYPICAL MICROCOMPUTER:

There are three major types of data transfer between the microcomputer and art I/O device. They are,

- **Programmed I/O : In programmed I/O the data transfer is accomplished through an I/O port and controlled by software.**
- **Interrupt driven I/O : In interrupt driven I/O, the I/O device will interrupt the processor, and initiate data transfer.**
- **Direct memory access (DMA) : In DMA, the data transfer between memory and I/O can be performed by bypassing the microprocessor.**



INTERFACING I/O AND PERIPHERAL DEVICES:

1. For data transfer from input device to processor the following operations are performed.

- **The input device will load the data to the port.**
- **When the port receives a data, it sends message to the processor to read the data.**
- **The processor will read the data from the port.**
- **After a data have been read by the processor the input device will load the next data into the port.**

2. For data transfer from processor to output device the following operations are performed.

- **The processor will load the data to the port.**
- **The port will send a message to the output device to read the data.**
- **The output device will read the data from the port.**
- **After the data have been read by the output device the processor can load the next data to the port.**
- **The various INTEL 110 port devices are 8212, 8155/8156, 8255, 8355 and 8755.**

8156:

- **It has two numbers of 8-bit parallel I/O port (port-A and B)**
- **One number of 6-bit parallel 1 port (port-C).**
- **It has 14 bit timer (operating in 4 modes).**
- **It has six internal addresses.**
- **It has one chip select pin CS (low).**

Internal Device	A_2	A_1	A_0
Control Register/ Status Register	0	0	0
Port-A	0	0	1
Port-B	0	1	0
Port-C	0	1	1
LSB of Timer	1	0	0
MSB of Timer	1	0	1

Fig - Internal address of 8156

8255:

- **It has 3 numbers of 8-bit parallel I/O ports (port A, B and C).**
- **Port-A can be programmed in mode-0 mode-1 or mode-2 as input or output port.**
- **Port-B can be programmed in mode-1 and mode-2 as 1/Oport.**
- **When ports A and B are in mode-0, the port-C can be used as I/O port.**
- **One logic low chip select (CS) pin.**
- **It requires four internal addresses**

Internal Device	A_1	A_0
Port-A	0	0
Port-B	0	1
Port-C	1	0
Control Register	1	1

Fig - Internal address of 8255

Device	Function	Internal addresses
INTEL 8279	Keyboard/display controller. Used for keyboard scanning and display refreshing.	**Two-internal addresses** $A_0 = 0 \rightarrow$ Data register $A_0 = 1 \rightarrow$ Control register
INTEL 8257 or INTEL 8237	DMA controller. Used for supporting DMA access to I/O device. It acts as a master during DMA mode. It is a slave device during programming mode.	**Sixteen-internal addresses** A_3 A_2 A_1 A_0 0 0 0 0 0 0 0 1 . . . . 1 1 1 1
INTEL 8259	Interrupt controller. Used to expand the hardware interrupt INTR to eight interrupts in 8085 based system and 256 interrupts in 8086 based system.	**Two-internal addresses** $A_0 = 0$ $A_0 = 1$
INTEL 8253/ 8254	Programmable Timer. Used in the system to produce various timing signals. It has three independent counters and can be programmed in six operating modes.	**Four-internal addresses** A_1 A_0 Counter-0 0 0 Counter-1 0 1 Counter-2 1 0 Control Register 1 1
INTEL 8251 USART	Universal Synchronous/Asynchronous Receiver Transmitter. Used for serial data communication.	**Two-internal addresses** $C/\overline{D} = 0 \rightarrow$ Data register $C/\overline{D} = 1 \rightarrow$ Control register

Experiment No:

GENERATE SAWTOOTH WAVEFORM

AIM: To generate Sawtooth waveform using 8085 microprocessor

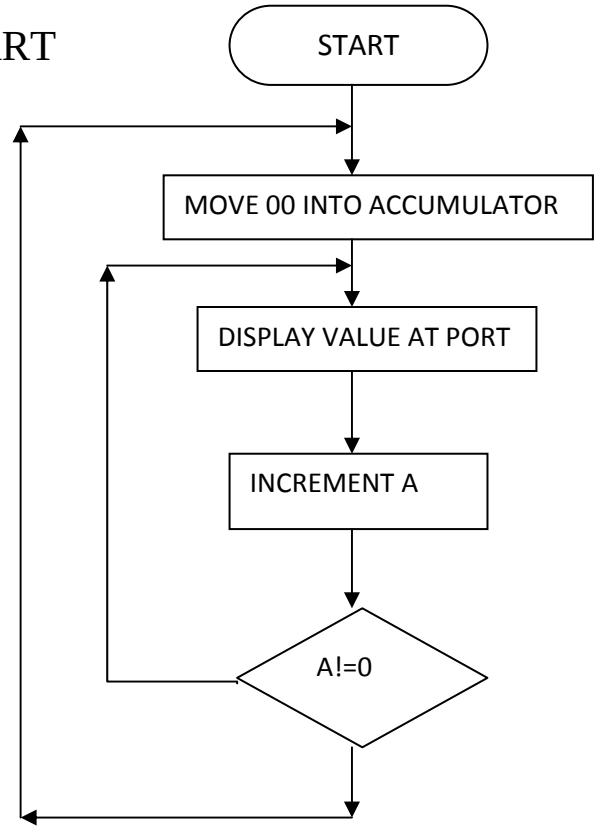
PROGRAM:

Start	MVI A,00	Make Accumulator '0'
Loop	OUT C0	Display Accumulator content at Port
	INR	Increment Accumulator by '1'
	JNZ Loop	If Accumulator is not equal to '0', go to Loop
	JMP Start	Jump to Start unconditionally, for continuous wave form

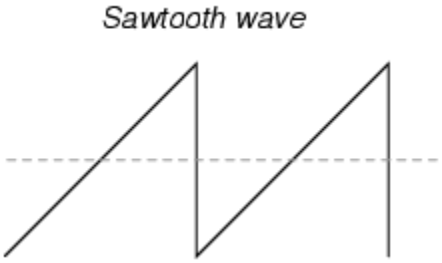
MNEMONICS:

ADDRESS	OPCODE	OPERAND	LOCATION	COMMAND
4100	3E	00	Start	MVI A,00
4102	D3	C0	Loop	OUT C0
4104	3C			INR
4105	C2	02 41		JNZ Loop
4108	C3	00 41		JMP Start

FLOWCHART



WAVEFORM:



CALCULATION:

Amplitude:

Time Period:

RESULT: The Sawtooth Waveform was generated and wave was plotted.

Date:
Experiment No:

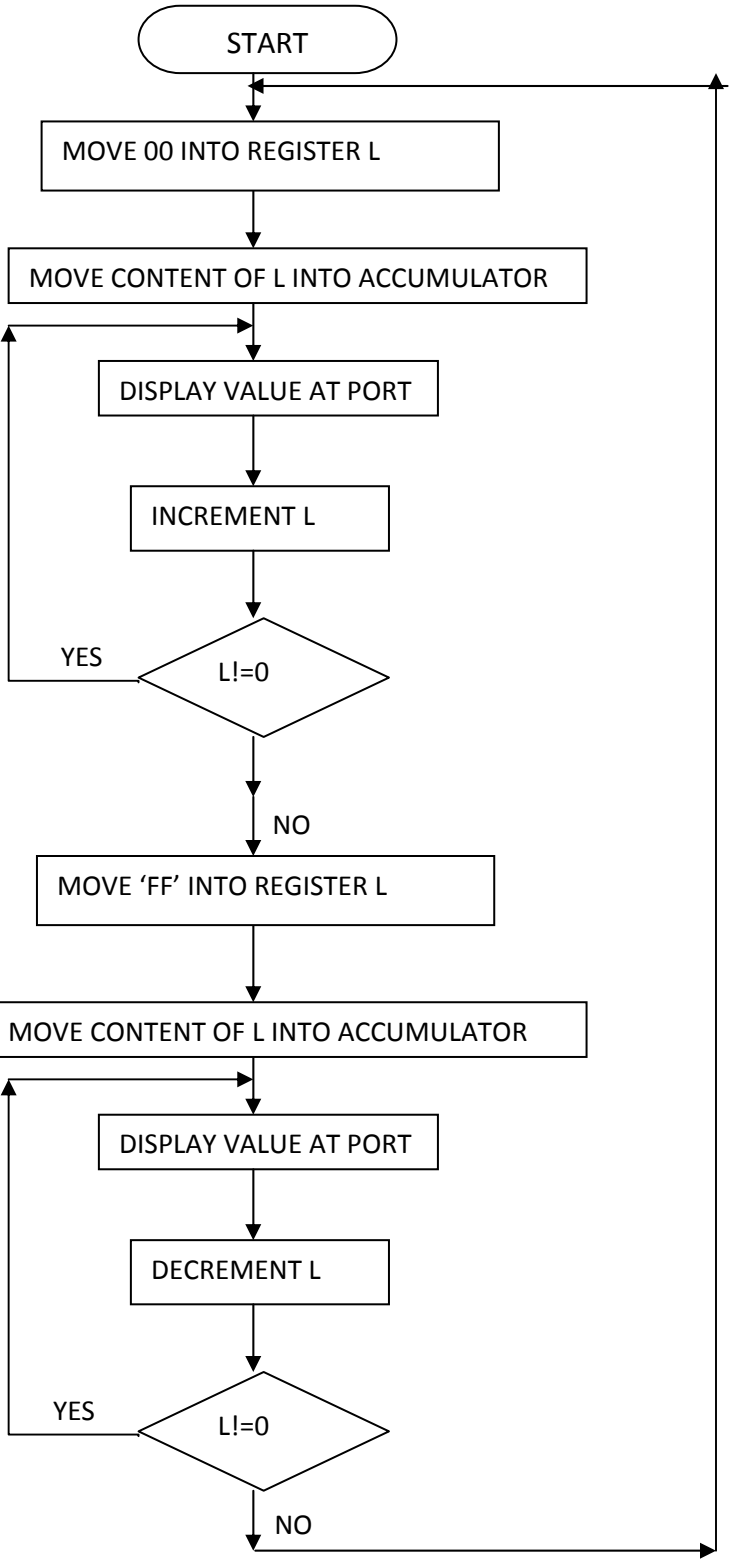
GENERATE TRIANGULAR WAVEFORM

AIM: To generate Triangular waveform using 8085 Microprocessor.

PROGRAM:

Start	MVI L,00	Set Register L equal to '0' -minimum
Loop 1	MOV A,L	Move contents of Register L to Accumulator
	OUT C8	Display Accumulator content at Port
	INR L	Increment Register L by '1'
	JNZ Loop1	If Register L is not '0' jump to Loop1
	MVI L,FF	Set Register L equal to 'FF' - maximum
Loop 2	MOV A,L	Move contents of Register L to Accumulator
	OUT C8	Display Accumulator content at Port
	DCR L	Decrement Register L by '1'
	JNZ Loop2	If Register L is not '0' jump to Loop2
	JMP Start	Jump to Start unconditionally, for continuous wave form

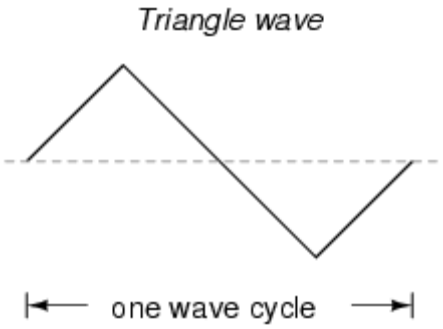
FLOWCHART


START
MOVE 00 INTO REGISTER L
MOVE CONTENT OF L INTO ACCUMULATOR
DISPLAY VALUE AT PORT
INCREMENT L
L!=0
YES
NO
MOVE 'FF' INTO REGISTER L
MOVE CONTENT OF L INTO ACCUMULATOR
DISPLAY VALUE AT PORT
DECREMENT L
L!=0
YES
NO

MNEMONICS:

ADDRESS	OPCODE	OPERAND	LOCATION	COMMAND
4100	2E	0	Start	MVI L,00
4102	7D		Loop 1	MOV A,L
4103	D3	C8		OUT C8
4105	2C			INR L
4106	C2	03 41		JNZ Loop1
4109	2E	FF		MVI L,FF
410B	7D		Loop 2	MOV A,L
410C	D3	C8		OUT C8
410E	2D			DCR L
410F	C2	0C 41		JNZ Loop2
4112	C3	00 41		JMP Start

WAVEFORM:



CALCULATION:

Amplitude:

Time Period:

RESULT: The Triangular Waveform was generated and wave was plotted.

Date:
Experiment No:

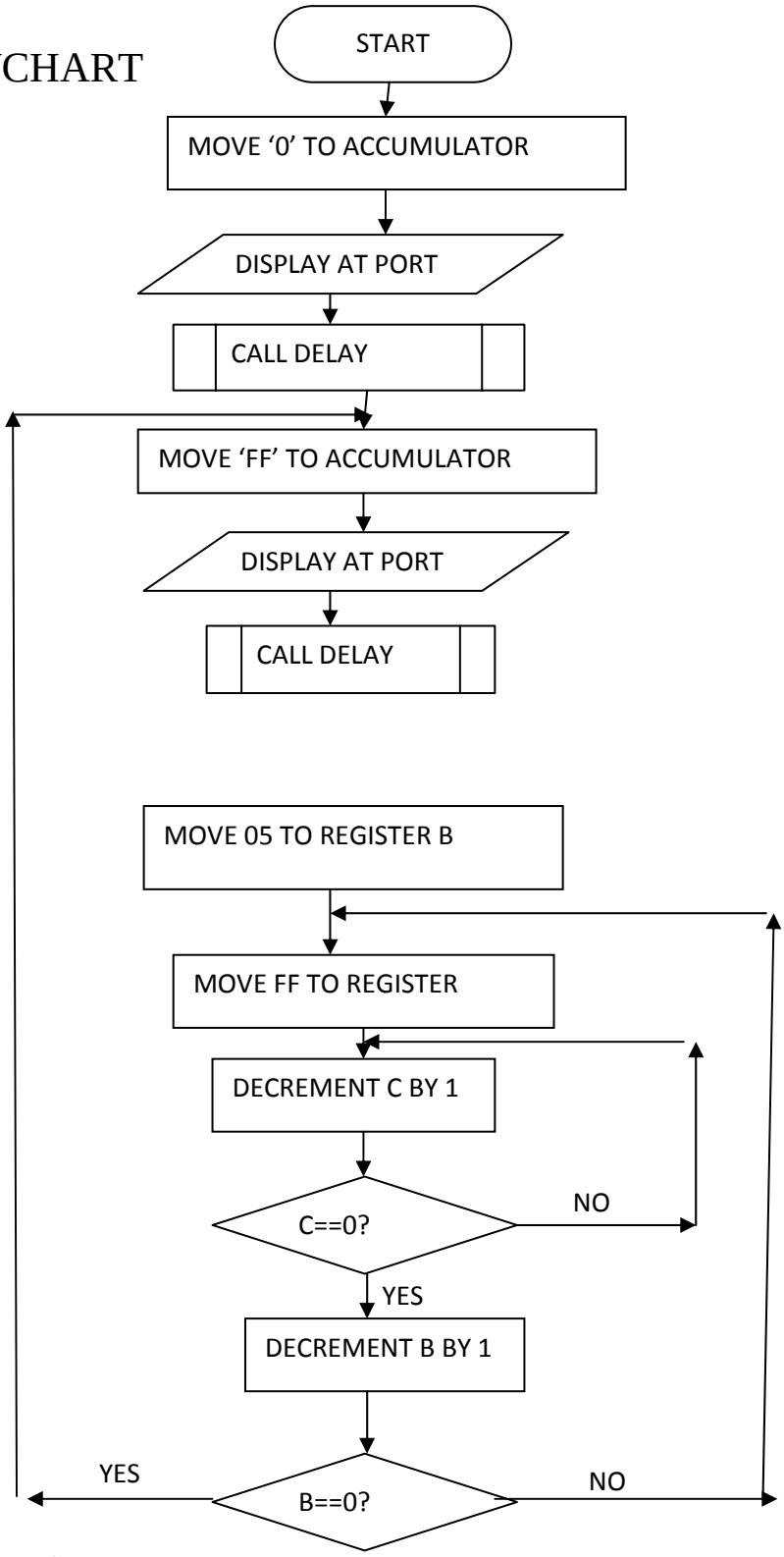
GENERATE SQUARE WAVEFORM

AIM: To generate Square waveform using 8085 Microprocessor

PROGRAM:

Start	MVI A,00	Move '0' into Accumulator
	OUT C8	Display 0 at port
	CALL DELAY	Call the DELAY subroutine
Return	MVI A,FF	Move 'FF' into Accumulator
	OUT C8	Display at port
	CALL DELAY	Call the DELAY subroutine
	JMP Start	Jump to start
DELAY	MVI B,05	Move 05 to Register B to give delay
Loop 1	MVI C,FF	Move FF to Register C
Loop 2	DCR C	Decrement Register C by 1
	JNZ Loop2	If C NOT '0' , decrement C again
	DCR B	Decrement Register B by 1
	JNZ Loop1	If B NOT '0', go to Loop1
	RET	Return to the main program

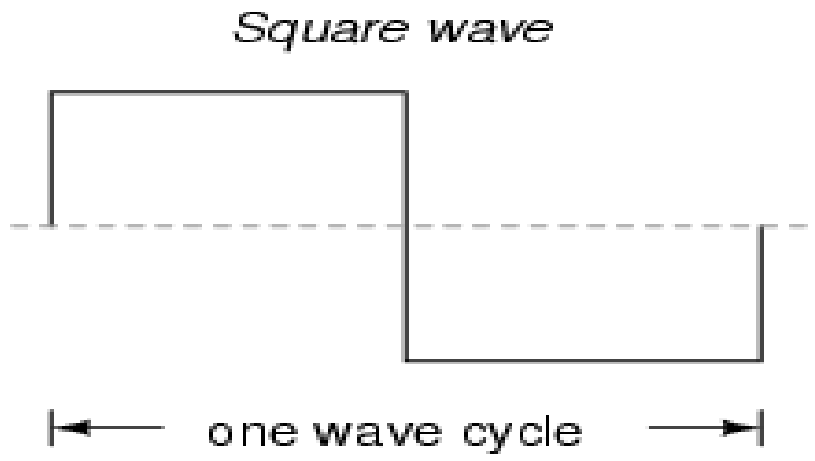
FLOWCHART


START
MOVE '0' TO ACCUMULATOR
DISPLAY AT PORT
CALL DELAY
MOVE 'FF' TO ACCUMULATOR
DISPLAY AT PORT
CALL DELAY
MOVE 05 TO REGISTER B
MOVE FF TO REGISTER
DECREMENT C BY 1
C==0?
NO
YES
DECREMENT B BY 1
B==0?
YES
NO

MNEMONICS:

ADDRESS	OPCODE	OPERAND	LOCATION	COMMAND
4100	3E	0	Start	MVI A,00
4102	D3	C8		OUT C8
4104	CD	20 41		CALL DELAY
4107	3E	FF	Return	MVI A,FF
4109	D3	C8		OUT C8
410B	CD	11 41		CALL DELAY
410E	C3	20 41		JMP Start
4120	6	5	DELAY	MVI B,05
4122	0E	FF	Loop 1	MVI C,FF
4124	0D		Loop 2	DCR C
4125	C2	24 41		JNZ Loop2
4128	5			DCR B
4129	C2	22 41		JNZ Loop1
412C	C9			RET

WAVEFORM:



CALCULATION:

Amplitude:

Time Period:

RESULT: The Square Waveform was generated and wave was plotted.

Date:
Experiment No:

GENERATING SINE WAVE

AIM: To generate Sine Waveform using 8085 Microprocessor

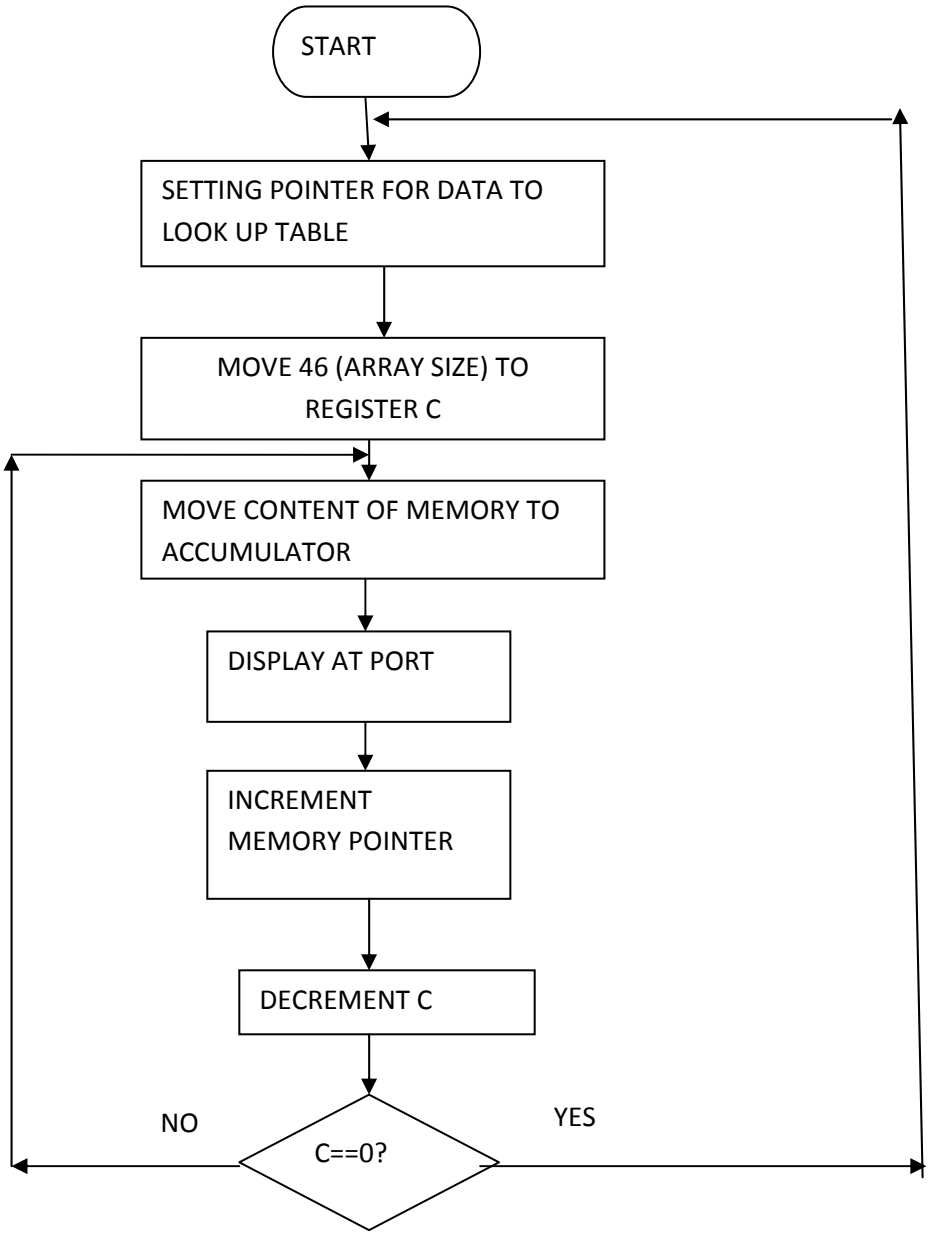
PROGRAM:

Start	LXI H, 4110	Set pointer to get data from Look-up table
	MVI C, 46	Set Register C for number of data in look up table
Loop	MOV A, M	Move contents of memory into Accumulator
	OUT C0	Display out put at port
	INX H	Increment memory pointer to next data
	DCR C	Decrement register C
	JNZ Loop	If C is not '0' , go to Loop
	JMP Start	Start again

Look -Up Table : Starting Address : 4110

74	84	95	A0	7F
8A	95	A0	AA	B5
BF	C8	D1	D9	E0
E7	ED	F2	F7	FA
FC	FE	FF	FE	FC
FA	F7	F2	ED	E7
E0	D9	D1	C8	BF
B5	AA	A0	95	8A
7F	74	69	5F	53
49	3F	36	2D	25
1D	17	10	0B	07
04	01	00	01	04
07	0B	10	17	1D
25	2D	36	3F	49
53	5F	69	74	

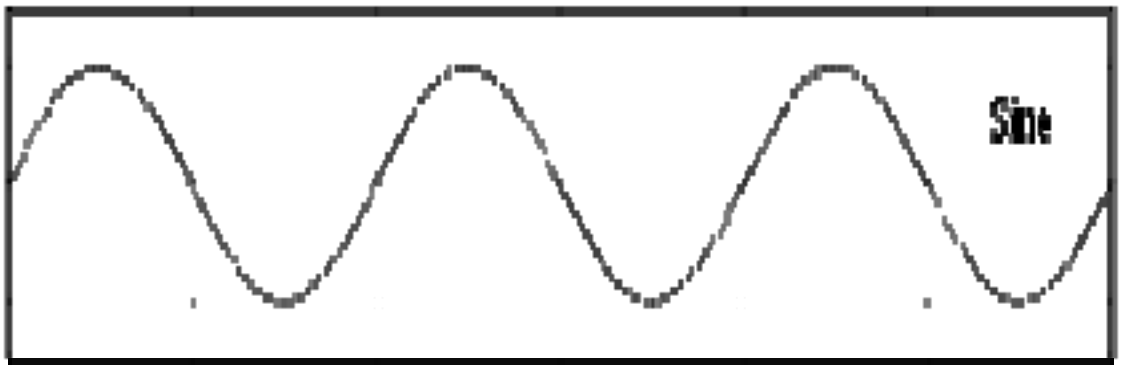
FLOWCHART


START
SETTING POINTER FOR DATA TO LOOK UP TABLE
MOVE 46 (ARRAY SIZE) TO REGISTER C
MOVE CONTENT OF MEMORY TO ACCUMULATOR
DISPLAY AT PORT
INCREMENT MEMORY POINTER
DECREMENT C
C==0?
NO
YES

MNEMONICS:

ADDRESS	OPCODE	OPERAND	LOCATION	COMMAND
4100	21	10 41	Start	LXI H, 4110
4103	0E	46		MVI C, 46
4105	7E		Loop	MOV A, M
4106	D3	C0		OUT C0
4108	23			INX H
4109	0D			DCR C
410A	C2	05 41		JNZ Loop
410D	C3	00 41		JMP Start

WAVEFORM:



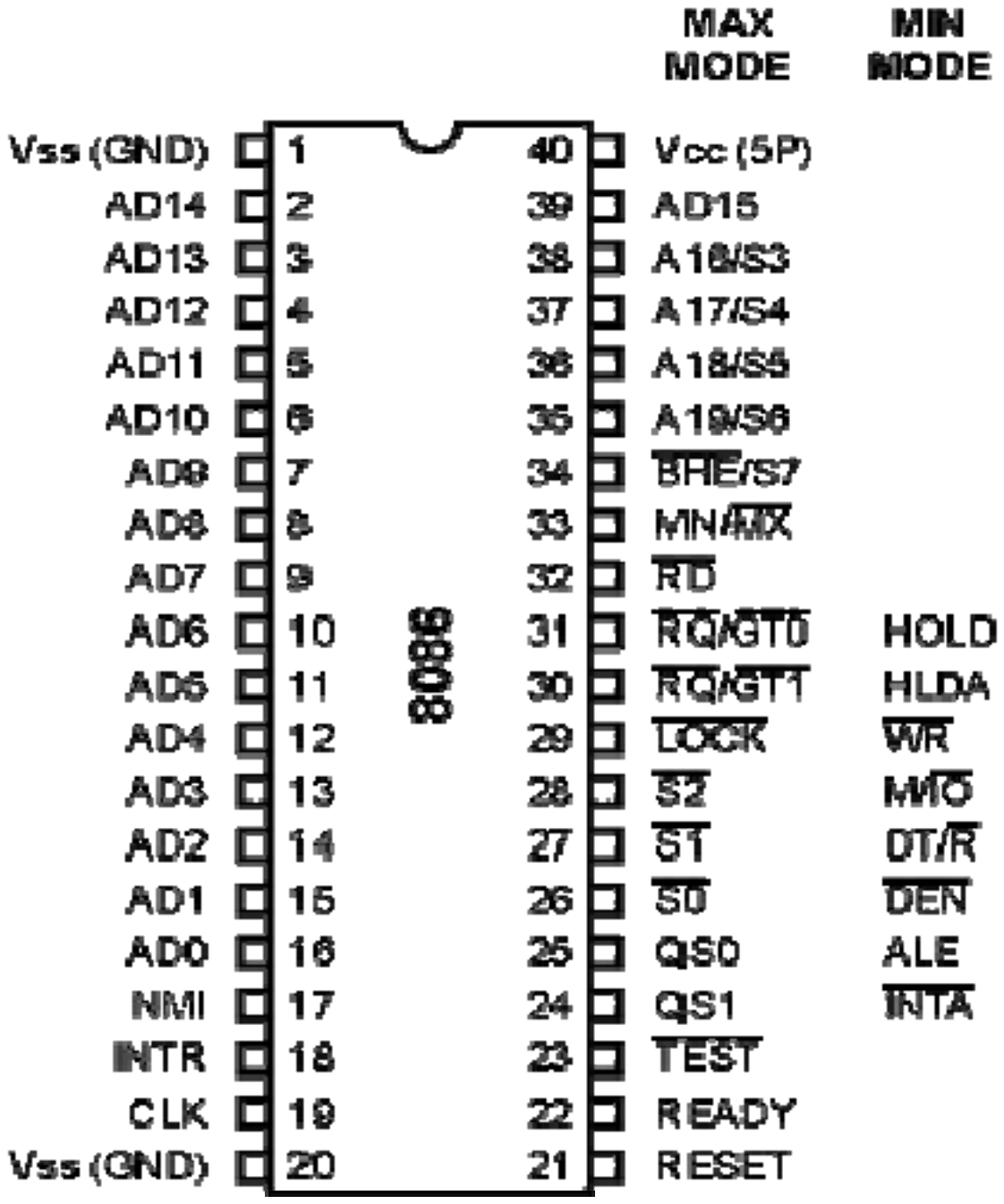
CALCULATION:

Amplitude:

Time Period:

RESULT: The Sine Waveform was generated and wave was plotted.

8086 Pin Diagram



8086 Instruction Set Summary

Data Transfer Instructions

MOV	Move byte or word to register or memory
IN, OUT	Input byte or word from port, output word to port
LEA	Load effective address
LDS, LES	Load pointer using data segment, extra segment
PUSH, POP	Push word onto stack, pop word off stack
XCHG	Exchange byte or word
XLAT	Translate byte using look-up table

Logical Instructions

NOT	Logical NOT of byte or word (one's complement)
AND	Logical AND of byte or word
OR	Logical OR of byte or word
XOR	Logical exclusive-OR of byte or word
TEST	Test byte or word (AND without storing)

Shift and Rotate Instructions

SHL, SHR	Logical shift left, right byte or word by 1 or CL
SAL, SAR	Arithmetic shift left, right byte or word by 1 or CL
ROL, ROR	Rotate left, right byte or word by 1 or CL
RCL, RCR	Rotate left, right through carry byte or word by 1 or CL

Arithmetic Instructions

ADD, SUB	Add, subtract byte or word
ADC, SBB	Add, subtract byte or word and carry (borrow)
INC, DEC	Increment, decrement byte or word
NEG	Negate byte or word (two's complement)
CMP	Compare byte or word (subtract without storing)
MUL, DIV	Multiply, divide byte or word (unsigned)
IMUL, IDIV	Integer multiply, divide byte or word (signed)
CBW, CWD	Convert byte to word, word to double word (useful before multiply/divide)

Adjustments after arithmetic operations:

AAA, AAS,	
AAM, AAD	ASCII adjust for addition, subtraction, multiplication, division (ASCII codes 30-39)
DAA, DAS	Decimal adjust for addition, subtraction (binary coded decimal numbers)

Transfer Instructions

JMP Unconditional jump (*short* 127/8, *near* 32K, *far* between segments)

Conditional jumps:

JA (JNBE)	Jump if above (not below or equal) +127, -128 range only
JAE (JNB)	Jump if above or equal(not below) +127, -128 range only
JB (JNAE)	Jump if below (not above or equal) +127, -128 range only
JBE (JNA)	Jump if below or equal (not above) +127, -128 range only
JE (JZ)	Jump if equal (zero) +127, -128 range only
JG (JNLE)	Jump if greater (not less or equal) +127, -128 range only
JGE (JNL)	Jump if greater or equal (not less) +127, -128 range only
JL (JNGE)	Jump if less (not greater nor equal) +127, -128 range only
JLE (JNG)	Jump if less or equal (not greater) +127, -128 range only
JC, JNC	Jump if carry set, carry not set +127, -128 range only
JO, JNO	Jump if overflow, no overflow +127, -128 range only
JS, JNS	Jump if sign, no sign +127, -128 range only
JNP (JPO)	Jump if no parity (parity odd) +127, -128 range only
JP (JPE)	Jump if parity (parity even) +127, -128 range only

Loop control:

LOOP	Loop unconditional, count in CX, short jump to target address
LOOPE (LOOPZ)	Loop if equal (zero), count in CX, short jump to target address
LOOPNE (LOOPNZ)	Loop if not equal (not zero), count in CX, short jump to target address
JCXZ	Jump if CX equals zero (used to skip code in loop)

Subroutine and Interrupt Instructions

CALL, RET Call, return from procedure (inside or outside current segment)

INT, INTO	Software interrupt, interrupt if overflow
IRET	Return from interrupt

String Instructions

MOVS	Move byte or word string
MOVSB, MOVSW	Move byte, word string
CMPS	Compare byte or word string
SCAS	Scan byte or word string (comparing to A or AX)
LODS, STOS	Load, store byte or word string to AL or AX

Repeat instructions (placed in front of other string operations):

REP	Repeat
REPE, REPZ	Repeat while equal, zero
REPNE, REPNZ	Repeat while not equal (zero)

Processor Control Instructions

Flag manipulation:

STC, CLC, CMC	Set, clear, complement carry flag
STD, CLD	Set, clear direction flag
STI, CLI	Set, clear interrupt enable flag
LAHF, SAHF	Load AH from flags, store AH into flags
PUSHF, POPF	Push flags onto stack, pop flags off stack

Coprocessor, multiprocessor interface:

ESC	Escape to external processor interface
LOCK	Lock bus during next instruction

Inactive states:

NOP	No operation
WAIT	Wait for TEST pin activity
HLT	Halt processor

Date:
Experiment No:

16 BIT ADDITION USING 8086

AIM: To add two 16 bit numbers (4 digits) using the 8086 microprocessor.

PROGRAM:

Program	Explanation
MOV SI, 1500	Set source index as 1500
LODSW	Load data from source memory and auto increment memory pointer
MOV AX, BX	Move data from AX to BX
LODSW	Load data from source memory
ADD BX, AX	Add contents of AX, BX
MOV DI,[1520]	Set Destination index as address in 1520
MOV [DI], BX	Move Result in BX to location pointed by Destination Index
INT 3	Break Point

INPUT & OUTPUT:

i) INPUT 1 4283
INPUT 2 2931
SUM 6BB4
CARRY 00

1500	83	Lower Byte of Data 1
1501	42	Higher Byte of Data 1
1502	31	Lower Byte of Data 2
1503	29	Higher Byte of Data 2
1520	B4	Lower Byte of Sum
1521	6B	Higher Byte of Sum

RESULT: Program to add two 16 bit numbers (4 digits) with 8086 microprocessor was executed

Date:
Experiment No:

MOVE CONTENTS OF ARRAY

AIM: To move contents of array from one memory location to another memory location.

PROGRAM:

Address	Program	Explanation
	MOV CL, 08	Set number of data i.e., count
	MOV SI, 1400	Set source index as 1400
	MOV DI, 1450	Set Destination index as memory address 1500
Loop	LODSB	Load data from source memory
	MOV [DI], AL	Move content of address pointed by destination Index to lower byte of accumulator
	INC DI	Increment Destination index
	DEC CL	Decrement count
	JNC Loop	Jump if not zero to Loop
	INT 3	Break Point

INPUT & OUTPUT:

INPUT		OUTPUT	
1400	11	1500	11
1401	22	1501	22
1402	33	1502	33
1403	44	1503	44
1404	21	1504	21
1405	31	1505	31
1406	41	1506	41
1407	51	1507	51

RESULT: Program to shift contents of an array from one location to another was executed

Date:
Experiment No:

SUM OF N CONSECUTIVE NUMBERS

AIM: To find the sum of N consecutive numbers with 8086 microprocessor

PROGRAM:

Address	Program	Explanation
	MOV SI, 2000	Set source index as 2000
	MOV CL, [SI]	Move content of address pointed by source index to CL
	MOV AL, 00	Clear AL to store sum
	MOV BL, 01	Move '1' to BL , as it's the first number
LOOP	ADD AL, BL	Add content of BL to AL
	INC BL	increment BL
	DEC CL	Decrement the count
	JNZ LOOP	Jump if not zero to Loop
	MOV DI, 2002	Set Destination index as memory address 2002
	MOV [DI], AX	Move content of Register A to destination, which is the total sum
	INT 3	Break Point

INPUT & OUTPUT:

INPUT	2000	05
OUTPUT	2002	0F

1+2+3+4+5 = 15 = F (HEXADECIMAL)

RESULT: Thus program for finding sum of ‘n’ consequent numbers was executed

Date:
Experiment No:

CONVERSION OF BCD TO HEXADECIMAL

AIM: To convert a BCD number into hexadecimal with 8086 microprocessor

PROGRAM:

Program	Explanation
MOV SI, 1600	Set source index as 1600
MOV DI, 1500	Set Destination index as memory address 1500
MOV AL, [SI]	Load BCD number into AL register
MOV AH, AL	Move content of AL to AH
AND AH, 0F	Mask MSD of BCD number
MOV BL, AH	Save LSD in BL register
AND AL, F0	Mask LSD of BCD number
MOV CL, 04	Load 04 to counter
ROR AL, CL	rotate AL by counter times
MOV BH, 0A	move 0A to BH register
MUL BH	Multiply BH register
ADD AL, BL	Add AL, BL
MOV [DI], AL	Move result to Destination
INT 3	Break Point

INPUT & OUTPUT:

INPUT	1600	56
OUTPUT	1500	38

RESULT: Thus program for converting BCD to Hexadecimal number was executed.

Date:
Experiment No:

SEPARATE ODD & EVEN

AIM: To separate odd and even numbers using 8086 microprocessor

PROGRAM:

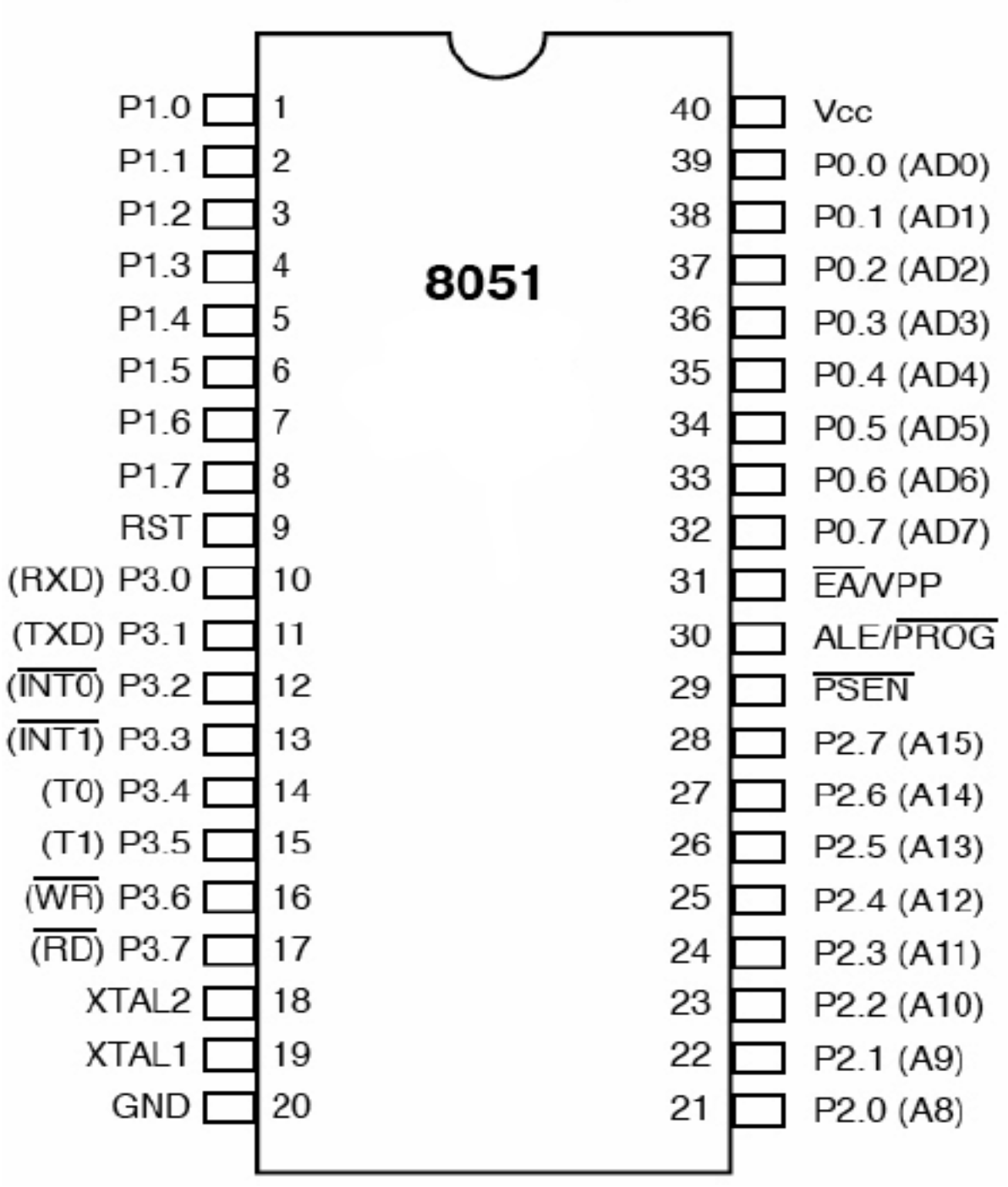
Address	Program	Explanation
	MOV CL, 06	Set counter in CL register
	MOV SI, 1600	Set source index as 1600
	MOV DI, 1500	Set Destination index as memory address 1500
Loop	LODSB	Load data from source memory
	ROR AL, 01	Rotate AL once to right
	JB Loop	If bit is one Jump to Loop
	ROL AL,01	Rotate AL once to left
	MOV [DI], AL	Move result to Destination
	INC DI	Increment Destination index
	DEC CL	Decrement the count
	JNZ Loop	Jump if not zero to Loop
	INT 3	Break Point

INPUT & OUTPUT:

INPUT	1600	2
	1601	5
	1602	7
	1603	6
	1604	12
	1605	15
OUTPUT	1500	5
	1501	7
	1503	15

RESULT: Thus program for separating Odd & Even numbers was executed.

8051 Pin Diagram



8051 Instruction Set

ARITHMETIC OPERATIONS

Mnemonic	Description
ADD A,Rn	Add register to Accumulator
ADD A,direct	Add direct byte to Accumulator
ADD A,@Ri	Add indirect RAM to Accumulator
ADD A,#data	Add immediate data to Accumulator
ADDC A,Rn	Add register to Accumulator with Carry
ADDC A,direct	Add direct byte to Accumulator with Carry
ADDC A,@Ri	Add indirect RAM to Accumulator with Carry
ADDC A,#data	Add immediate data to Acc with Carry
SUBB A,Rn	Subtract Register from Acc with borrow
SUBB A,direct	Subtract direct byte from Acc with borrow
SUBB A,@Ri	Subtract indirect RAM from ACC with borrow
SUBB A,#data	Subtract immediate data from Acc with borrow
INC A	Increment Accumulator
INC Rn	Increment register
INC direct	Increment direct byte
INC @Ri	Increment direct RAM
DEC A	Decrement Accumulator
DEC Rn	Decrement Register
DEC direct	Decrement direct byte
DEC @Ri	Decrement indirect RAM
INC DPTR	Increment Data Pointer
MUL AB	Multiply A & B
DIV AB	Divide A by B
DA A	Decimal Adjust Accumulator

LOGICAL OPERATIONS

Mnemonic	Description
ANL A,Rn	AND Register to Accumulator
ANL A,direct	AND direct byte to Accumulator
ANL A,@Ri	AND indirect RAM to Accumulator
ANL A,#data	AND immediate data toAccumulator
ANL direct,A	AND Accumulator to direct byte

ANL direct,#data	AND immediate data to direct byte
ORL A,Rn	OR register to Accumulator

LOGICAL OPERATIONS (continued)

Mnemonic	**Description**
ORL A,direct	OR direct byte to Accumulator
ORL A,@Ri	OR indirect RAM to Accumulator
ORL A,#data	OR immediate data to Accumulator
ORL direct,A	OR Accumulator to direct byte
ORL direct,#data	OR immediate data to direct byte
XRL A,Rn	Exclusive-OR register to Accumulator
XRL A,direct	Exclusive-OR direct byte to Accumulator
XRL A,@Ri	Exclusive-OR indirect RAM to Accumulator
XRL A,#data	Exclusive-OR immediate data to Accumulator
XRL direct,A	Exclusive-OR Accumulator to direct byte
XRL direct,#data	Exclusive-OR immediate data to direct byte
CLR A	Clear Accumulator
CPL A	Complement Accumulator
RL A	Rotate Accumulator Left
RLC A	Rotate Accumulator Left through the Carry
RR A	Rotate Accumulator Right
RRC A	Rotate Accumulator Right through the Carry
SWAP A	Swap nibbles within the Accumulator

DATA TRANSFER

Mnemonic	**Description**
MOV A,Rn	Move register to Accumulator
MOV A,direct	Move direct byte to Accumulator
MOV A,@Ri	Move indirect RAM to Accumulator
MOV A,#data	Move immediate data to Accumulator
MOV Rn,A	Move Accumulator to register
MOV Rn,direct	Move direct byte to register
MOV Rn,#data	Move immediate data to register
MOV direct,A	Move Accumulator to direct byte
MOV direct,Rn	Move register to direct byte
MOV direct,direct	Move direct byte to direct
MOV direct,@Ri	Move indirect RAM to direct byte

MOV direct,#data	Move immediate data to direct byte
MOV @Ri,A	Move Accumulator to indirect RAM
MOV @Ri,direct	Move direct byte to indirect RAM
MOV @Ri,#data	Move immediate data to indirect RAM
MOV DPTR,#data16	Load Data Pointer with a 16-bit constant

DATA TRANSFER (continued)

Mnemonic	**Description**
MOVC A,@A+DPTR	Move Code byte relative to DPTR to Acc
MOVC A,@A+PC	Move Code byte relative to PC to Acc
MOVX A,@Ri	Move External RAM (8- bit addr) to Acc
MOVX A,@DPTR	Move Exernal RAM (16- bit addr) to Acc
MOVX @Ri,A	Move Acc to External RAM (8-bit addr)
MOVX @DPTR,A	Move Acc to External RAM (16-bit addr)
PUSH direct	Push direct byte onto stack
POP direct	Pop direct byte from stack
XCH A,Rn	Exchange register with Accumulator
XCH A,direct	Exchange direct byte with Accumulator
XCH A,@Ri	Exchange indirect RAM with Accumulator
XCHD A,@Ri	Exchange low-order Digit indirect RAM with Acc

BOOLEAN VARIABLE MANIPULATION

Mnemonic	**Description**
CLR C	Clear Carry
CLR bit	Clear direct bit
SETB C	Set Carry
SETB bit	Set direct bit
CPL C	Complement Carry
CPL bit	Complement direct bit
ANL C,bit	AND direct bit to CARRY
ANL C,/bit	AND complement of direct bit to Carry
ORL C,bit	OR direct bit to Carry
ORL C,/bit	OR complement of direct bit to Carry
MOV C,bit	Move direct bit to Carry
MOV bit,C	Move Carry to direct bit
JC rel	Jump if Carry is set
JNC rel	Jump if Carry not set

JB bit,rel	Jump if direct Bit is set
JNB bit,rel	Jump if direct Bit is Not set
JBC bit,rel	Jump if direct Bit is set & clear bit

PROGRAM BRANCHING

Mnemonic	**Description**
ACALL addr11	Absolute Subroutine Call
LCALL addr16	Long Subroutine Call
RET	Return from Subroutine

PROGRAM BRANCHING(continued)

Mnemonic	**Description**
RETI	Return from interrupt
AJMP addr11	Absolute Jump
LJMP addr16	Long Jump
SJMP rel	Short Jump (relative addr)
JMP @A+DPTR	Jump indirect relative to the DPTR
JZ rel	Jump if Accumulator is Zero
JNZ rel	Jump if Accumulator is Not Zero
CJNE A,direct,rel	Compare direct byte to Acc and Jump if Not Equal
CJNE A,#data,rel	Compare immediate to Acc and Jump if Not Equal
CJNE R_n,#data,rel	Compare immediate to register and Jump if Not Equal
CJNE @R_i,#data,rel	Compare immediate to indirect and Jump if Not Equal
DJNZ R_n,rel	Decrement register and Jump if Not Zero
DJNZ direct,rel	Decrement direct byte and Jump if Not Zero
NOP	No Operation

Source : **Atmel 8051 Microcontrollers Hardware Manual**

Date:
Experiment No:

8 BIT ADDITION

AIM: To perform 8-bit addition using 8051 microcontroller

PROGRAM:

Address	Program	Explanation
	MOV a,#23	Move data 1 to a
	MOV r1,#11	Move data 2 to register 1
	MOV r2, #00	Set r2 for carry
	ADD a,r1	Add a, r1
	JNC ahead	Jump on no carry to ahead
	INC r2	If carry is '1' increment r2
ahead	MOV DPTR,#9200	Set datapointer to location 9200
	MOVX @ DPTR,a	move a contents to location pointed by data pointer (DPTR)
	INC DPTR	Increment pointer
	MOV a,r2	Move r2 to a
	MOVX @ DPTR,a	move a contents to location pointed by data pointer (DPTR)
	LCALL 00BB	End

INPUT & OUTPUT:

Data 1	23
Data 2	11
9200	34
9201	0

RESULT: Thus program for adding two 8 bit numbers had been executed.

Date:
Experiment No:

8 BIT SUBTRACTION

AIM: To perform 8-bit subtraction using 8051 microcontroller

PROGRAM:

Address	Program	Explanation
	MOV a,#23	Move data 1 to a
	MOV r1,#11	Move data 2 to register 1
	MOV r2, #00	Set r2 for carry
	SUBB a,r1	Subtract r1 from a and store in a
	JNC ahead	Jump on no carry to ahead
	INC r2	If carry is '1' increment r2
ahead	MOV DPTR,#9200	Set datapointer to location 9200
	MOVX @ DPTR,a	move a contents to location pointed by data pointer (DPTR)
	INC DPTR	Increment pointer
	MOV a,r2	Move r2 to a
	MOVX @ DPTR,a	move a contents to location pointed by data pointer (DPTR)
	LCALL 00BB	End

INPUT & OUTPUT:

Data 1	23
Data 2	11
9200	12
9201	0

RESULT: Thus program for subtracting two 8 bit numbers had been executed.