(i) Addition:

Memory Address	Mnemonics	operands	Hex codes
8000	MOV	AX, 1234 H	B8, 34, 12
8003	MOV	BX, 1234 H	BB, 34, 12
8006	ADD	AX, BX	03, C3
8008	MOV	[8500], AX	A3, 00, 85
800B	HLT		F4

sample Input/output:

AX : 1234 H
BX : 1234 H } → operands

AX : 2468 H → Result

(ii) Subtraction:

Memory Address	Mnemonics	operands	Hex codes
8000	MOV	AX, 1234H	B8, 34, 12
8003	MOV	BX, 1234H	BB, 34, 12
8006	SUB	AX, BX	2B, C3
8008	MOV	[8500], AX	A3, 00, 85
800B	HLT		F4

NO: 1

Date: 27/6/19

16 BIT HEXADECIMAL OPERATIONS

Aim:

To perform the following 16 bit hexadecimal operation

(i) 16 bit Addition of two numbers
(ii) 16 bit subtraction "
(iii) 16 bit multiplication of "
(iv) 32 bit by 16 bit division.

Algorithm:

(i) 16 bit Addition:

1) Get the operand 1 from user using AX.
2) Get the operand 2 from user by BX register
3) Add the value in AX & BX and store them in AX register
4) Transfer the result from AX to an address Location say 8500.

(ii) 16 bit subtraction:

1) Get operand 1 & 2 from user using AX & BX.
2) Subtract the value of BX from AX and store them in AX register
3) Transfer the result stored in AX to an address Location say 8500

sample Input/output :

AX : 1234 H } → operands
BX : 1234 H }

AX : 0000 H → Result

(ii) Multiplication :

Memory Address	Mnemonics	operands	Hexcodes
8000	MOV	AX, 1234H	B8, 34, 12
8003	MOV	BX, 0002H	BB, 02, 00
8006	MOV	DX, 0000	BA, 00, 00
8009	MUL	BX	F7, E3
800B	MOV	[8500], AX	A3, 00, 85
800E	MOV	[8502], DX	89, 16, 02, 85
8012	HLT		F4

sample Input/output :

AX : 0004 H }
BX : 0002 H } operands
DX : 0000 H }

AX : LSB of result → 08 H
DX : MSB of result → 00 H

(iii) Multiplication :

1) Load the starting address of the program
2) Load the first data in AX
3) Load the second data in BX.
4) Multiply AX and BX.
5) Move result stored in AX to destination address 8500.

(iv) 32 bit by 16 bit Division :

1) Load the starting address of the program
2) Load first data in AX and second data ie Divisor in BX.
3) Initilize DX with 0000H ie MSB of dividend
4) Divide AX By BX and move quotient stored in AX toaddress 8500
5) Move the Quotient stored in DX to an address 8502.

(iv) 32 bit by 16 bit Division :

Memory Address	Mnemonics	Operands	Hexcode
8000	MOV	AX, 0008 H	B8, 08, 00
8003	MOV	BX, 0002 H	BB, 02, 00
8006	MOV	DX, 0000 H	BA, 00, 00
8009	DIV	BX	F6, F3
800B	MOV	[8500], AX	A3, 00, 85
800E	MOV	[8502], DX	89, 16, 02, 85
8012	HLT		F4

Sample Input/output :

DX : 0000 H
AX : 0008 H } → Dividend

BX : 0002 H → Divisor

AX : 0004 H → quotient

DX : 0000 H → Remainder

Result :

The 16 bit Hexadecimal operations such as

(i) 16 bit Addition of two numbers
(ii) 16 bit by 16 bit subtraction
(iii) 16 bit by 16 bit Multiplication and
(iv) 32 bit by 16 bit Division

are executed succesfully using 8086 microprocessor

(i) Addition with carry :

Mem. Address	Mnemonics	operands	Hexcodes
8000	MOV	AL, 12H	B0, 12
8002	MOV	AH, 12H	B4, 12
8004	MOV	CL, 00H	B1, 00
8006	ADD	AL, AH	02, C4
8008	JNC	Result	73, 02
800A	INC	CL	FE, C1
800C	Result: MOV	[8500], AL	A2, 00, 85
800F	MOV	[8502], CL	88, 0E, 02, 85
8013	HLT		FA

Sample Input/output :

AL : 12 H } → operand
AH : 12 H }

[8500] : 24 H → result

Addition without carry :

Mem Address	Mnemonics	operands	Hexcodes
8000	MOV	AL, 12	B0, 12
8002	MOV	AH, 12	B4, 12
8004	ADD	AL, AH	02, C4
8006	MOV	[8500], AL	A2, 00, 85
8009	HLT		F4

Exp No : 2

Date : 11/7/19

8 BIT HEXADECIMAL OPERATIONS

Aim :

To perform 8 bit hexadecimal operations such as Addition, subtraction, multiplication and 16 bit by 8 bit division using 8086 microprocessor.

Algorithm :

(i) Addition :

1) Load starting address of the program
2) Load Data 1 in AL register and Data 2 in AH register
3) Add AL and AH if carry is generated then it is also added along with the result
4) Store the result from AL register to Address 8500.

Sample Input/output :

AL : 12 H
AH : 12 H
} → operands

[8500] : 24H → Result

(ii) Subtraction With Borrow :

Mem. Address	Mnemonics	operands	Hexcode
8000	MOV	AL, 05	B0, 05
8002	MOV	AH, 02	B4, 02
8004	SUBB	AL, AH	2A, C4
8006	MOV	[8500], AL	A2, 00, 85
8009	HLT		F4

sample input/output :

AL : 05 H
AH : 02 H
} → operand

[8500] : 03 H → result

subtraction without Borrow :

Mem. Address	Mnemonics	operands	Hexcodes
8000	MOV	AL, 05	B0, 05
8002	MOV	AH, 02	B4, 02
8004	SUB	AL, AH	2A, C4
8006	MOV	[8500], AL	A2, 00, 85
8009	HLT		F4

(ii) Subtraction :

1) Load the Data 1 in AL and Data 2 in AH
2) Subtract AL with AH and store the result in AL.
3) Move result from AL to address 8500.

(iii) Multiplication :

1) Load data 1 in AL and initilize DL with 0000H
2) Load data 2 ie Multiplier in AH.
3) Multiply AL with AH.
4) Move MSB result into AL register
5) Transfer result from AL to address location 8500.

(iv) Division :

1) Load data 1 ie Dividend in AL and Initialize BL with 00
2) Load data 2 ie Divisor in BL
3) Divide AL by BL
4) Move result ie quotient from AL to an address 8500
5) Move remainder from DL to an address 8501.

Sample input/output :

AL : 05 H
AH : 02 H
} → operands

[8500] : 03 H → result

(iii) Multiplication :

Mem. Address	Mnemonics	operands	Hexcodes
8000	MOV	AL, 12	B0, 12
8002	MOV	AH, 08	B1, 08
8004	MUL	AH	F6, E3
8006	MOV	[8500], AX	A2, 00, 85
8009	HLT		F4

sample Input/output :

AL : 12 H → Dividend / Multiplicand
AH : 08 H → Divisor / Multiplier
AX : 60 H → Result

(iv) Division :

Mem. Address	Mnemonics	operands	Hexcodes
8000	MOV	AL, 12	B0, 12
8002	MOV	AH, 00	B4, 00
8004	MOV	BL, 12	B3, 12
8006	DIV	BL	F6, F3
8008	MOV	[8500], AL	A2, 00, 85
800B	MOV	[8501], AH	88, 26, 01, 85
800F	HLT		F4

Sample Input/output :

AL : 12 } → Dividend
AH : 00 }

BL : 12

AL : 01 → Quotient
AH : 00 → Remainder

Result :

The 8 bit Hexadecimal operations such as Addition, subtraction, Multiplication & 16 bit by 8 bit Division are Executed successfully using 8086 microprocessor.

(i) 8bit Decimal Addition:

Mem. Address	Mnemonics	operand	Hexcode
8000	MOV	AL,12H	B0,12
8002	MOV	BL,12 H	B3,12
8004	ADD	AL, BL	02,C3
8006	DAA		27
8007	MOV	[8500], AL	A2,00,85
800A	HLT		F4

Sample Input/output:

AL : 12 H
BL : 12 H } → operands

[8500] : 24 H → result

(ii) 8bit Decimal Subtraction:

Mem. Address	Mnemonics	operand	Hexcode
8000	MOV	AL,12H	B0,12
8002	MOV	BL,02H	B3,02
8004	SUB	AL, BL	2A,C3
8006	DAS		2F
8007	MOV	[8500], AL	A2,00,85
800A	HLT		F4

Sample Input/output:

AL : 12 H
BL : 02 H } → operands

[8500] : 10H → result

Exp No: 3
Date: 25/7/19

8 BIT DECIMAL OPERATIONS

(12)

Aim:

To perform 8 bit decimal operations such as addition, subtraction, Multiplication & Division using 8086 microprocessor.

Algorithm:

(i) 8bit Decimal Addition:

1) Load Data 1 in AL register
2) Load Data 2 in BL register
3) Add AL and BL and convert it from hexadecimal and decimal.
4) Store the result in 8500 Address Location from AL

(ii) 8bit Decimal Subtraction:

1) Load Data 1 in AL register
2) Load Data 2 in BL register
3) Subtract BL from AL and convert hexadaimal to decimal number.
4) Store the result in an address Location 8500

(iii) 8bit Decimal Multiplication:

Memory Address	Mnemonics	Operands	Hexcode
8000	MOV	AL, 12H	B0, 12
8002	MOV	BL, 02H	B3, 02
8004	MUL	BL	F6, E3
8006	AAM		D4, 0A
8008	MOV	[8500], AX	A2, 00, 85
800B	HLT		F4

Sample Input/output:

AL: 12H → Multiplicand
BL: 02H → Multiplier
[8500]: 24H → result

(iv) 8bit Decimal Division:

Mem. Address	Mnemonics	Operands	Hexcode
8000	MOV	AH, 00H	B4, 00
8002	MOV	AL, 24H	B0, 24
8004	MOV	BL, 12H	B3, 12
8006	DIV	BL	F6, F3
8008	AAD		D5, 0A
800A	MOV	[8500], AL	A2, 00, 85
800D	MOV	[8501], AH	88, 02, 06, 04, 85
8012	HLT		F4

sample Input/output:

AH: 00 H
AL: 24 H } → Divisor

[8500]: 02H → quotient

(iii) 8bit Multiplication:

1) Load Multiplicand in AL register
2) Load Multiplier in BL register
3) Multiply AL with BL register values and convert hexadecimal value into decimal value
4) Store the result in an address 8500.

(iv) 8bit Decimal Division:

1) Load Dividend in AL register & Initilize AH register with 00H.
2) Load Divisor in BL register
3) Divide AL By BL and convert Hexadecimal Value into Decimal Value
4) store the quotient in Addres 8500 from AL
5) Store the remainder in Addres 8501 from AH

Result:

The 8 bit decimal operations such as addition, subtraction, Multiplication & Division are executed successfully using 8086 microprocessor.

(i) Finding Minimum Element :

Mem. Address	Mnemonics	operands	Hex codes
8000	MOV	SI, 8500	BE, 00, 85
8003	MOV	CX, 000AH	B9, 0A, 00
8006	DEC	CX	49
8007	MOV	AL, [SI]	8A, 04
8009	TOP: INC	SI	46
800B	CMP	AL, [SI]	3A, 04
800D	JNC	DOWN	73, 04
800F	LOOP	TOP	E2, F9
8011	JMP	LAST	EB, 04
8013	DOWN: XCHG	AL, [SI]	86, 04
8015	LOOP	TOP	E2, F3
8017	Last: MOV	SI, 8600	BE, 00, 86
801A	MOV	[SI], AL	88, 04
801C	HLT		F4

sample Input/Output :

8500 - 00
8501 - 01
8502 - 02
8503 - 03
8504 - 04
8505 - 05
8506 - 06
8507 - 07
8508 - 08
8509 - 09

→ Input

8600 : 00 H → output

No: 4

Date: 1/8/19

SEARCHING AND SORTING

(14)

Aim :

To perform -

(i) To find smallest element
(ii) To find largest element
(iii) Sorting elements in Ascending order
(iv) Sorting elements in Descending order

Algorithm:

(i) To find the smallest element :

step 1 : Input the elements at 8500.
step 2 : Load CX with no. of elements
step 3 : compare two elements if first element < second element jump to bottom else continue Loop
step 4 : In bottom exchange those elements
step 5 : Finally store the smallest element at 8600.

(i) Finding Maximum element :

Mem. Address	Mnemonics	operands	Hexcodes
8000	MOV	SI, 8500	BE, 00, 85
8003	MOV	CX, 000AH	B9, 0A, 00
8006	DEC	CX	49
8007	MOV	AL, [SI]	8A, 04
8009	TOP: INC	SI	46
800B	CMP	AL, [SI]	3A, 04
800D	JC	DOWN	72, 1A, 80
800F	LOOP	TOP	E2, 08, 80
8011	JMP	LAST	F9, 1E, 80
8013	DOWN: XCHG	AL, [SI]	86, 04
8015	LOOP	TOP	E2, F3
8017	LAST: MOV	SI, 8600	BE, 00, 86
801A	MOV	[SI], AL	88, C6
801C	HLT: HLT		F4

Sample Input/output :

8600 : 09H

(ii) To find Largest element :

step 1 : Input the elements at 8500.

step 2 : Load CX with no. of elements

step 3 : Compare No.1 and No.2 if No.1 > No.2 then exchange them else continue Loop.

step 4 : Store the result in the memory location 8600.

(iii) Sort elements in Ascending order :

step 1 : Input the elements at 8500

step 2 : Load CX with no of elements

step 3 : Compare two elements if first element < second element then continue looping

step 4 : else exchange them

(iii) Ascending order sorting :

Mem. Address	Mnemonics	operands	Hexcode
8000	MOV	SI, 8500	BE, 00, 85
8003	MOV	CH, 08H	B5, 08
8005	UP2: MOV	CL, 08H	B1, 08
8007	UP1: LEA	AL, [SI]	89, 04
8009	MOV	BL, [SI+1]	8A, 5C, 01
800C	CMP	AL, BL	38, D8
800D	JC	DOWN	72, 08
800F	MOV	DL, [SI+1]	8A, 54, 01
8012	XCHG	[SI], DL	88, 54, 01
8015	DOWN: INC	SI	46
8016	DEC	CL	FE, C9
8018	JNZ	UP1	75, EA
801A	DEC	CH	FE, CD
801C	JNZ	UP2	75, E0
801E	HLT		F4

Sample Input/output :

	Before sorting	After sorting
8500 -	09	00
8501 -	08	01
8502 -	07	02
8503 -	06	03
8504 -	05	04
8505 -	04	05
8506 -	03	06
8507 -	02	07
8508 -	01	08
8509 -	00	09

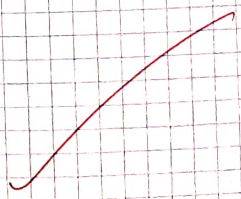
(iv) Sort elements in Descending order :

Step 1: Input the elements at 8500

Step 2: Load cx with no. of elements

Step 3: compare two elements if first element > second element then continue looping

Step 4: Else exchange them



(iv) Ascending order sorting :

Mem. Address	Mnemonics	operands	Hexcode
8000	MOV	SI, 8500	BE, 00, 85
8003	MOV	CH, 08H	B5, 08
8005	UP2: MOV	CL, 08H	B1, 08
8007	UP1: LEA	AL, [SI]	89, 04
8009	MOV	BL, [SI+1]	8A, 5C, 01
800C	CMP	AL, BL	38 D8
800D	JNC	DOWN	73, 08
800F	MOV	DL, [SI+1]	8A, 54, 01
8012	XCHG	[SI], DL	86, 14
8015	MOV	[SI+1], DL	88, 54, 01
8016	DOWN: INC	SI	46
8018	DEC	CL	FE, C9
801A	JNZ	UP1	75, EA
801C	DEC	CH	FE, CD
801E	JNZ	UP2	75, E0
	HLT		F4

Sample Input/output :

8500 - 00	07
8501 - 01	06
8502 - 02	05
8503 - 03	04
8504 - 04	03
8505 - 05	02
8506 - 06	01
8507 - 07	00

Result :

The searching and sorting operations are implemented and executed successfully using 8086 microprocessor.

Memory Address	Mnemonics	operands	Hexcodes
8000	MOV	SI,8100	BE,00,81
8003	MOV	DI,8200	BF,00,82
8006	MOV	CX,00AH	B9,0A,00
8009	CLD		FC
800A	REPMOVSB		F3,A4
800C	HLT		F4

Sample Input/output :

Address	Value
8100	00
8101	01
8102	02
8103	03
8104	04
8105	05
8106	06
8107	07
8108	08
8109	09

Address	Value
8200	00
8201	01
8202	02
8203	03
8204	04
8205	05
8206	06
8207	07
8208	08
8209	09

Ex: 5

Date: 8/8/19

BLOCK TRANSFER

(18)

Aim:

To Transfer block of elements from one location to another location.

Algorithm:

step 1: Input elements at some location

step 2: Load SI with the starting address

step 3: Load DI with the destination address

step 3: Load cx with no. of elements

step 4: Move contents present at SI to DI. until cx becomes zero.

10/10 10/9/19

Result:

A Block of elements are transfered from one location to another successfully.

Mem. Address	Mnemonics	Operand	Hexcode
8000	MOV	SI, 9000	BE, 00, 90
8003	MOV	BL, 0AH	B3, 0A
8005	MOV	AH, 00H	B4, 00
8007	MOV	AL, [SI]	8A, 04
8009	TOP: CMP	AL, BL	38, D8
800B	JC	STORE	72, 07
800D	DIV	BL	F6, F3
800F	INC	SI	46
8010	MOV	[SI], AH	88, 24
8012	JMP	TOP	EB, F5
8014	STORE: INC	SI	46
8015	MOV	[SI], AL	88, 04
8017	HLT		F4

sample Input/output:

9000 : 10H } Input

9001 : 06 }

9002 : 1 } output

Exp no: 6

Date: 29/8/19 NUMBER CONVERSIONS 19

Aim:

To perform -

(i) Hexadecimal to Decimal conversions

(ii) Decimal to Hexadecimal conversion.

Algorithm:

(i) Hexadecimal to Decimal conversion:

Step 1: Input the Hexadecimal value at 9000

Step 2: Load BL with 0AH.

Step 3: Compare AL & BL if AL < BL then Increment SI and MOV AL to SI.

Step 4: Else Divide it by BL and continue looping

Step 5: store the result at 9001 and 9002.

Mem. Address	Mnemonics	Operand	opcodes
8000	MOV	SI, 9000	BE, 00, 90
8003	MOV	AL, [SI]	8A, 04
8005	MOV	BH, 01	B7, 01
8007	AND	AL, 0FH	24, 0F
8009	MUL	BH	F6, E7
800B	MOV	DX, AX	8B, D0
800D	MOV	AL, [SI]	8A, 04
800F	AND	AL, F0H	24, F0
8011	MOV	CL, 04H	B1, 04
8013	SHR	AL, CL	D2, F8
8015	MOV	BH, 0AH	B7, 0A
8017	MUL	BH	F6, E7
8019	ADD	AX, DX	03, C2
801B	INC	SI	46
801C	MOV	[SI], AX	89, 04
801E	HLT		F4

Sample Input/output :

9000 : 11 H } → Input

9001 : 0B } → output

(ii) Decimal to Hexadecimal conversion :

step 1 : Input Decimal value at 9000.

step 2 : convert decimal value to Hexadecimal value

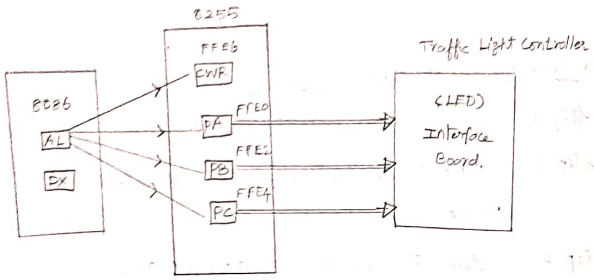
step 3 : store the result at 9001 and 9002

step 4 : stop the program.

10/10 19/9/19

Result :

The Number conversions are performed successfully using 8086 microprocessor.



Label	Mnemonics	operands	Memory	opcodes
	MOV	AL,80H	8000	B0,80
	MOV	DX,0FFE6H	8002	BA,E6,FF
	OUT	DX,AL	8005	EE
Repeat:	MOV	SI,8027	8006	BE,27,80
Next:	MOV	AL,[SI]	8009	8A,04
	MOV	DX,0FFE0H	800B	BA,E0,FF
	OUT	DX,AL	800E	EE
	INC	SI	800F	46
	MOV	DX,0FFE2H	8010	BA,E2,FF
	MOV	AL,[SI]	8013	8A,04
	OUT	DX,AL	8014	EE
	INC	SI	8015	46
	MOV	DX,0FFE4H	8016	BA,E4,FF
	MOV	AL,[SI]	8019	8A,04
	OUT	DX,AL	801B	EE

EXNO:7

Date: 19/9/19 INTERFACING TRAFFIC LIGHT CONTROLLER WITH 8086 21

Aim:

To interface the traffic light controller with 8086 microprocessor.

Algorithm:

step 1 : Write the control word for initialization

step 2 : All three ports as output port in the control word register

step 3 : Write the first byte of state 1 into the port A whose address is FFE0H

step 4 : Increment SI register by 1 to get the next address

step 5 : Call the delay routine to maintain the traffic state for some duration of time

step 6 : If the last state is reached then stop the execution

step 7 : Repeat the above steps

Label	Mnemonics	Operands	Memory	Opcodes
	MOV	SI	8015	46
	MOV	DX, 0FFE4H	8018	BA, E4, FF
	MOV	AL, [SI]	8019	8A04
	OUT	DX, AL	801B	EE
	INC	SI	801C	46
	CALL	delay	801D	E8, 09, 00
	CMP	SI, 8055	8020	81, FF, 55, 80
	JNZ	Next	8024	75, E2
	JMP	Repeat	8026	E9, DC, FF
Delay :	MOV	CX, 00FFH	8029	B9, FF, 00
TOP1 :	Push	CX	802C	51
	MOV	CX, 03FFH	802D	B9, FF, 03
TOP2 :	NOP		8030	90
	LOOP	TOP2	8031	E2, FD
	POP	CX	8033	59
	LOOP	TOP1	8034	E2, F6
	RET		8036	C3

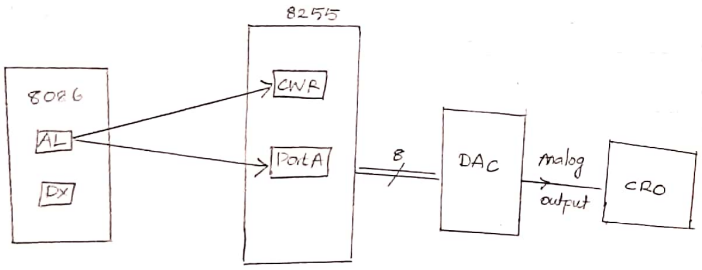
Sample Input/output:

Data Memory	Memory Data	Comments
8037	10H, 81H, 7AH	state 1
803A	44H, 44H, F0H	All Ambers on
803D	08H, 11H, E5H	state 2
8040	44H, 44H, F0H	All Ambers on
8043	81H, 10H, 0DH	state 3
8046	44H, 44H, F0H	All Ambers on
8049	11H, 08H, B5H	state 4
804C	44H, 44H, F0H	All Ambers on
804F	88H, 88H, 00H	state 5
8052	44H, 44H, F0H	All Ambers on
8055	00	End of Last state

10/10 26/9/19

Result :

Interfacing the traffic light controller with 8086 microprocessor have been successfully executed.



Label	Mnemonics	operands	opcodes	Memory
	MOV	AL, 80H	B0, 80	8000
	MOV	DX, FFE6H	BA, E6, FF	8002
	OUT	DX, AL	EE	8005
Repeat	MOV	AL, 00H	B0, 00	8006
	MOV	DX, FFE0H	BA, E0, FF	8008
Again1	OUT	DX, AL	EE	800B
	INC	AL	FE, C0	800C
	CMP	AL, FF	3C, FF	800E
	JB	Again1	72, F9	8010
Again2	OUT	DX, AL	EE	8012
	DEC	AL	FE, C8	8013
	CMP	AL, 00H	3C, 00	8015
	JNZ	Again2	75, F9	8017
	JMP	Repeat	E9, EB	8019

Exno: 8
Date: 9.9.19

GENERATION OF TRIANGULAR WAVEFORM USING DAC WITH 8086

Aim:

To generate the Waveform using DAC with 8086 microprocessor namely square, Triangular & sawtooth

Algorithm:

step 1: Move the control word into CWR

step 2: output the control word

step 3: Write 00H into port A until it reaches maximum of FFH

step 4: Then decrease the value it reaches the minimum value 00H

step 5: Repeat the steps above.

Label	Mnemonics	operands	opcodes	Memory
	MOV	AL,80H	B0,80	8000
	MOV	DX,FFE6H	BA,E6,FF	8002
	OUT	DX,AL	EE	8005
Repeat	MOV	AL,00H	B0,00	8006
	MOV	DX,AL	BA,E0,FF	8008
	call	Delay	EE	800B
	MOV	DX,FFH	BA,E0,FF	800C
	MOV	BX,FFE0H	B0,FF	800F
	OUT	DX,AL	BA,E0,FF	8011
	JMP	Repeat	EE	8014
Delay	MOV	CX,FF00H	E8,02,00	8015
TOP	DEC	CX	EB,EC	8018
	JNZ	TOP	49	8019
	RET		C3	801A

GENERATION OF SQUARE WAVEFORM USING DAC WITH 8086

Algorithm :

step 1 : Move the control word into CWR

step 2 : output the control word

step 3 : Write 00H into port A

step 4 : call the delay for the suitable frequency

step 5 : Repeat the loop again for continuous generation of waveform.

Label	Mnemonics	operands	Hexcodes	Memory
	MOV	AL,80H	BA,80	8000
	MOV	DX/FFE6H	BA,E6,FF	8002
	OUT	DX/AL	EE	8004
Repeat	MOV	AL,00H	B0,00	8006
JNC :	MOV	DX/FFE0H	BA,E0,FF	8008
	OUT	DX/AL	EE	800A
	INC	AL	FE,C0	800C
	CMP	AL,FFH	3C,FF	800D
	JB	JNC	72,76	800F
	JMP	Repeat	EB,F2	8010

GENERATION OF SAWTOOTH WAVEFORM USING DAC WITH 8086

Algorithm:

step 1 : Move the control word into the CWR

step 2 : output the control word

step 3 : write 00H into port A and increases towards FFH then decrease to 00H.

step 4 : call the delay for the suitable frequency

step 5 : Repeat the loop above.

Result:

The generation of waveform using DAC with 8086 microprocess have been generated successfully.

Memory	Label	Mnemonics	operands	Hex codes
8000		MOV	DX, FFE6H	BA, E6, FF
8003		MOV	AL, 89H	B0, 89
8005		OUT	DX, AL	EE
8006		MOV	CL, 00H	B1, 00
8008		MOV	DX, FFE4H	BA, E4, FF
800C	Loop1:	IN	AL, DX	EC
800E		AND	AL, 02H	24, 02
8010		JZ	Loop1	74, FB
8013		call	Delay	E8, 24, 00
8014				
8016	Loop2:	IN	AL, DX	EC
8018		AND	AL, 02H	24, 02
801A		JNZ	Loop2	75, FB
801D	Loop3:	MOV	AL, CL	8A, C1
801E		MOV	DX, FFE0H	BA, E0, FF
8021		OUT	DX, AL	EE
8024		CALL	Delay	E8, 16, 00
8025		MOV	DX, FFE4H	BA, E4, FF
8027		IN	AL, DX	EC
8029		AND	AL, 01	24, 01
802B		JZ	finish	74, 04
802D		INC	CL	FE, C1
802F		JMP	Loop3	EB, EB
8031	finish:	MOV	AL, CL	8A, C1
8036		MOV	AH, 00H	B4, 00
8037		CALL	0B0A: FF00	9A, 0A, 0B, 00, FF
803A		HLT		F4
803B	delay:	MOV	BX, 05FFH	BB, FF, 05
803D	Loop4:	DEC	BX	4B
		JNZ	Loop4	75, FD
		RET		C3

EXNO: 9

Date: 26/9/19 CONVERSION OF ANALOG TO DIGITAL SIGNAL

26

Aim:

To convert the analog signal to digital signal using assembly language program.

Algorithm:

step 1: Move the data to DX to be converted

step 2: output the initial inputs from DX to AL

step 3: use three looping functions for the conversion of analog to digital signal

step 4: call the delay function

step 5: use the loop for repeated steps

10/10 26/9/19

Result:

The Assembly language program to convert analog to digital signal has been executed successfully.

Memory	Label	Mnemonics	operands	opcodes
8000		MOV	AL,00H	B0,00
8002		MOV	DX,0FFEBH	BA,EB,FF
8005		OUT	DX,AL	EE
8006		MOV	AL,CH	B0,c1
8008		OUT	DX,AL	EE
8009		MOV	AL,90H	B0,90
800B		OUT	DX,AL	EE
800C	Repeat:	MOV	CX,00FH	B9,0F,00
800F		MOV	SI,8500	BE,00,85
8012		MOV	DX,FFE9H	BA,E9,FF
8015	Next:	MOV	AL,SI	8A,04
8017		OUT	DX,AL	EE
8018		CALL	Delay	E8,05,00
801B		INC	SI	46
801C		Loop	next	E2,F7
801E		JMP	Repeat	EB,EC
8020	Delay:	MOV	BX,0005H	BB,05,00
8023	Sub:	MOV	DI,FFFFH	BF,FF,FF
8026	Subt:	DEC	DI	4F
8027		JNZ	Subt	75,FD
8029		DEC	BX	4B
802A		JNZ	sub	75,F7
802C		RET		C3

Sample Input/output:

8500 : FF
8501 : 7F
8502 : 06

A B I A E I

EXNo: 10

Date: 26/9/19 DISPLAY OF CHARACTERS IN ROLLING FASHION 27

Aim:

To write an assembly language program to display the characters in rolling function.

Algorithm:

step 1: Write initialization control word into the register

step 2: Declare the keyboard and display controller

step 3: use the repeat and next function to get the element of input to display

step 4: use and call the delay function

step 5: use subagain and subt loop or function to display character in rolling function.

Result:

The Assembly Language program to display the character in Rolling fashin has been executed successfully.

Memory	Label	Mnemonics	operands	opcodes
8000		MOV	DX, FFE6H	BA, E6, FF
8003		MOV	AL, 80 H	B0, 80
8005		OUT	DX, AL	EE
8006		MOV	DX, FFE0H	BA, E0, FF
8009		MOV	AL, 88H	B0, 80
800B	Back:	OUT	DX, AL	EE
800C		CALL	Delay	E8, 04, 00
800F		ROR/ROL	AL, 01H	D0, C8
8011		JMP	Back	EB, F8
8013	Delay :	MOV	CX, 4000H	B9, 00, 40
8016	Repeat :	Loop	repeat	E2, FE
8018		Ret		C3

EXNO: 11

Date: 26/9/19 STEPPER MOTOR INTERFACE 28

Aim:

To write an assembly language program to implement stepper motor interface with 8086

Algorithm:

step 1: Initialize all the 8255 outputs

step 2: Declare and initialize of register is done

step 3: output the data to the port

step 4: call the delay function

step 5: Rotate the data byte either in clockwise or anticlockwise direction.

26/9/19

Result:

The Assembly language program to implement stepper motor interface has been executed successfully

Address	Label	Mnemonics	operands	opcodes
8000		MOV	DPTR, #8500	90, 85, 00
8003		MOVX	A, @ DPTR	E0
8004		MOV	R0, A	F8
8005		INC	DPTR	A3
8006		MOVX	A, @DPTR	E0
8007		ADDC	A, R0	38
8008		MOV	R1, A	F9
8009		INC	DPTR	A3
800A		MOVX	A, @ DPTR	E0
800B		MOV	R0, A	F8
800C		INC	DPTR	A3
800D		MOVX	A, @DPTR	E0
800E		ADDC	A, R0	38
800F		INC	DPTR	A3
8010		MOVX	@DPTR, A	F0
8011		INC	DPTR	A3
8012		MOV	A, R1	E9
8013		MOVX	@DPTR, A	F0
		SJMP	FF	80, FF

Sample I/o:

8500	01	LSB Data 1
8501	00	LSB Data 2
8502	01	MSB Data 1
8503	00	MSB Data 2
8504	01	MSB Result
8505	01	MSB Result

Exno: 12

Date: 03/11/19

8051 ARITHMETIC OPERATIONS

29

Aim:

To write assembly language program to perform arithmetic operations -

1) Addition
2) subtraction
3) Multiplication
4) Division using 8051 Microcontroller

Algorithm:

step 1: Move the first data into the data pointer.

step 2: Move second data to the data pointer

step 3: Add two numbers in the data pointer

step 4: Display or store the result in the location.

Address	Mnemonics	operands	opcodes
8000	MOV	DPTR,#8500	90,85,00
8003	MOVX	A,@DPTR	E0
8004	MOV	R0,A	F8
8005	INC	DPTR	A3
8006	MOVX	A,@DPTR	E0
8007	SUBB	A,R0	98
8008	MOV	R1,A	F9
8009	INC	DPTR	A3
800A	MOVX	A,@DPTR	E0
800B	MOV	R0,A	F8
800C	INC	DPTR	A3
800D	MOVX	A,@DPTR	E0
800E	SUBB	A,R0	98
800F	INC	DPTR	A3
8010	MOVX	@DPTR,A	F0
8011	INC	DPTR	A3
8012	MOV	A,R1	E9
8013	MOVX	@DPTR,A	F0
8014	SJMP	FE	80,FE
8015			

Sample I/o:

8500 → 40H } LSB Data 1
8501 → 60H } " Data 2
8502 → 00H } MSB Data 1
8503 → 00H } " Data2
8504 → 00 H } Result MSB
8505 → 20 H } LSB

2) Subtraction:

Algorithm:

1) Move the data into register A using DPTR
2) Move the Data 2 into register R0 using DPTR.
3) Subtract the two numbers in the data pointer
4) Display or store the result in the specified location.

Address	Mnemonics	Operands	Opcodes
8000	MOV	DPTR, #8500	90,85,00
8003	MOVX	A, @DPTR	E0
8004	MOV	F0, A	F5,F0
8006	INC	DPTR	A3
8007	MOVX	A, @DPTR	E0
8008	MUL	AB	A4
8009	INC	DPTR	A3
800A	MOVX	@DPTR, A	F0
800B	SJMP	FE	80,FE

Sample I/o:

8500 → 04 H } Data
8501 → 02 H }
8502 → 08 H → Result

3) MULTIPLICATION:

Algorithm:

1) Move the Data A into the Data pointer
2) Move the second data to multiply to the Data pointer
3) Multiply the two numbers in the data pointer
4) Store the result in the specified Location.

Address	Mnemonics	Operands	Opcodes
8000	MOV	DPTR, #8500	90,85,00
8003	MOVX	A, @DPTR	E0
8004	MOV	F0, A	F5, F0
8006	INC	DPTR	A3
8007	MOVX	A, @DPTR	E0
8008	DIV	AB	84
8009	INC	DPTR	A3
800A	MOVX	@DPTR, A	F0
800B	INC	DPTR	A3
800C	MOV	A, F0	E5, F0
800E	MOVX	@DPTR, A	F0
800F	SJMP	FE	80, FE

Sample I/O:

8500 → 02 H Divisor
8501 → 05 H Dividend
8502 → 02 H Quotient
8503 → 01 H Remainder

A) DIVISION:

Algorithm:

1) Move the first data to the Data pointer
2) Move the second data to divide the number in the data pointer
3) Divide the two numbers in the Data pointer
4) Display or store the result.

10/10 DD 4/10/19.

Result:

The Arithmetic operations are executed successfully using 8051 microcontroller.

EXNO:13

Date: 8/10/19

PROGRAM TO FIND 2's complement of a Number

Aim:

To Find the 2's complements of a number using 8051 microcontroller.

Algorithm:

1) Move the Data into the Data pointer
2) Store the Data
3) Increment the Data pointer
4) Find the complement of a number
5) Add 1 to it to find 2's complement.

Address	Mnemonics	Operands	Opcodes
8000	MOV	DPTR, #8500	90,85,00
8003	MOVX	A, @DPTR	E0
8004	CPL	A	F4
8005	ADD	A, #01	24,01
8007	INC	DPTR	A3 F0
8008	MOVX	@DPTR, A	F5, A0
800A	SJMP	FE	80, FE

Sample I/o:

8500 → 02 H Input
8501 → FE 2's complement

Result:

2's complement of a number is executed successfully using 8051 microcontroller.

Address	Mnemonics	Operands	Opcodes
8000	MOV	DPTR, #8500	90, 85, 00
8003	MOVX	A, @DPTR	E0
8004	MOV	F0, A	F5, F0
8006	MOV	R0, A	F8
8007	MUL	AB	A4
8008	INC	DPTR	A3
8009	MOVX	@DPTR, A	F0
800A	MOV	F0, A	F5, F0
800C	MOV	A, R0	E8
800D	MUL	AB	A4
800E	INC	DPTR	A3
800F	MOVX	@DPTR, A	F0
8010	SJMP	FF	80, FF

Sample I/O:

8500 → 02 Input
8501 → 04 square
8502 → 08 cube

Date: 20/10/23

PROGRAM TO FIND SQUARE AND CUBE OF A NUMBER

Aim:

To find the square and cube of a number using 8051 microcontroller.

Algorithm:

1) Move the Data into register A using data pointer
2) Move the data again into Register B
3) Multiply A & B to get square of a number
4) Again multiply the result with regiter B to get the cube of a number

Result:

The square and cube of a number is found successfully using 8051 microcontroller

Address	Mnemonics	operands	opcodes
8000	MOV	DPTR, #8500	90,85,00
8003	MOV	A, #35	74,35
8005	CLR	C	C3
8006	SUBB	A, #30	94,30
8008	CLR	C	C3
8009	SUBB	A, #0A	94,0A
800B	JC	STR	40,04
800D	MOV	A, #FF	74,FF
800F	SJMP	L1	80,02
8011			
8013	STR: ADD	A, #0A	24,0A
8015	L1: MOVX	@DPTR, A	F0
8016	SJMP	FF	80,FF

Sample I/O:

Accumulator → 35H (Ascii)

8500 → 05 H (BCD unpacked)

EX.No: 15

Date: 23/10/19 PROGRAM TO CONVERT ASCII TO BCD 35

Aim:

To write an assembly Language program to Convert ASCII To BCD (unpacked) using 8051 microcontroller.

Algorithm:

1) Move data to the Data pointer
2) Increment the Data pointer
3) Input the Data that is to be converted ASCII
4) convert the number to equivalent BCD (unpacked)

23/10/19

Result:

The program to convert ASCII To BCD (unpacked) is executed successfully using 8051 Microcontroller.

Address	Mnemonics	operands	opcodes
8000	MOV	R5, #23H	7D, 23
8002	MOV	R7, #58H	7F, 58
8004	Loop: MOV	A, R5	ED
8005	MOV	61H, A	F5, 61
8007	MOV	A, R7	EF
8008	MOV	60H, A	F5, 60
800B	LCALL	UPDATEADD	12, 02, 0B
800D	UP: MOV	R1, #00H	79, 00
8010	Repeat: MOV	A, R1	E9
8012	MOV	R6, A	FE
8013	MOV	60H, A	F5, 60
8014	LCALL	UPDATE DATA	12, 01, 9B
8016	LCALL	DELAY	12, 80, 44
8019	MOV	A, R6	FE
801C	MOV	A, R1	E9
801D	ADD	A, #01H	24, 01
801E	DA	A	D4
8020	MOV	R1, A	F9
8021	CJNE	R1, #60H, Repeat	B9, 60, ED
8024	MOV	A, #00H	74, 00
8026	MOV	60H, A	F5, 60
8028	LCALL	UPDATE DATA	12, 01, 9B
802B	MOV	A, R7	EF
802C	ADD	A, #01H	24, 01
802D	DA	A	D4
802E	MOV	R7, A	FF
8032	CJNE	R7, #60H, MINUTE	BF, 60, 26
8033	MOV	R1, #00H	74, 00
8035	MOV	60, A	F5, 60
8037	MOV	A, R5	ED
8038	ADD	A, #01H	24, 01
803A	DA	A	D4
803B	MOV	R5, A	FD
803C	CJNE	R7, #24H, HOURS	BD, 24, 23

EXNO:16 36

Date: 24/10/19 PROGRAM TO IMPLEMENT DIGITAL CLOCK USING 8051

Aim:

TO write a program to implement digital clock using 8051 microcontroller

Algorithm:

1) store hours in R5 register
2) store minutes in R7 register
3) Move the hours to A register
4) Move hours from accumulator to the internal RAM memory address '60H'
5) Move minutes from accumulator to internal RAM memory address 61H
6) call subroutine to enable the data field to display seconds
7) call the delay for one second
8) Repeat the steps again.

Address	Mnemonics	operands	opcodes
803D	MOV	R5, #00H	7D,00
803F	MOV	R7, #00H	7F,00
8040	LJMP	LOOP	02,80,04
8041	DELAY: MOV	R2, #04H	7A,04
8044	BACK3: MOV	R4, #FFH	7C,FF
8046	BACK2: MOV	R3, #FFH	7B,FF
8048	BACK1: DEC	R3	1B
8049	CJNE	R3, #00H, BACK1	BB,00,FC
804B	DEC	R4	1C
804E	CJNE	R4, #00H, BACK2	BC,00,F6
804F	DEC	R2	1A
8052	CJNE	R2, #00H, BACK3	BA,00,F0
8054	RET		22
8057	MINUTE: MOV	A, R7	EF
8058	MOV	60H, A	F5,69
805A	LCALL	UPDATEADD	12,02,0B
805C	LJMP	UP	02,80,0D
805E	HOURS: MOV	A, R5	ED
8060	MOV	61H, A	F5,61
8061	LCALL	UPDATEADD	12,02,0B
8062	MOV	R7, #00H	7F,00
8063	LJMP	UP	02,80,0D

4/10/19

Result:

The Assembly language program to implement digital clock using 8051 has been executed successfully.