SAVEETHA SCHOOL OF ENGINEERING
SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES
CHENNAI – 602105

Engineer to Excel

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ECA10 MICROPROCESSORS AND MICROCONTROLLERS LAB

INDEX

	EXPERIMENTS BEYOND SYLLABUS	
26.	**Traffic Light controller**	
27.	**Square wave generation using 8085/8086**	
28.	**DAC - 8085/8086 Interface**	

Arithmetic Operations in 8085 Microprocessors

[a] 8-bit Addition using direct addressing

To write an assembly language program to add tow numbers of 8-bit data stored in memory locations 4200H and 4201H and store the result in 4202H and 4203H.

Problem Analysis

To perform addition is 8085 one of the data should be in accumulator and another data can be in any one of the general-purpose register or in memory. After addition the sum will be in accumulator. The sum of two 8 bit data can be either 8 bits (sum only) or 9 bit (sum and carry). The accumulator can accommodate only the sum and if here is a carry, the 8085 will indicate by setting carry flag. Hence one of the registers is used to account for carry.

In algorithm – 1 direct addressing is used to indicate data. But in algorithm – 2 register indirect addressing is used to indicate data. Here HL register is used to hold the address of the data and it is called pointer.

Algorithm 1

- Load the first data from memory to accumulator and move it to B register.
- Load the second data from memory to accumulator.
- Clear C – register.
- Add the content of B – register to accumulator
- Check for carry. If carry = 1, go to step 6 or f carry = 0, go to step 7.
- Increment the C – register
- Store the sum in memory.
- Move the carry to accumulator and store in memory.
- Stop.

PROGRAM TO ADD TWO 8-BIT DATA

Memory address	Label	Instruction	Opcode	Comments
		LDA 4200H		Get 1st data in A and save in B.
		MOV B, A		
		LDA 4201H		Get 2nd data in A-register
		MVI C,00H		Clear C register to account for carry.
		ADD B		Get the sum in A register
		JNC AHEAD		IF CY = 0, go to AHEAD
		INR C		IF CY = 1, increment C register
	AHEAD	STA 4202H		Store the sum in memory
		MOV A,C		
		STA 4203H		Store the carry in memory
		HLT		Stop the Execution

Input		Output	
Address	**Data**	**Address**	**Data**
4200		**4202**	**(Sum)**
4201		**4203**	**(Carry)**

Arithmetic Operations in 8085 Microprocessors
[a] 8-bit Addition using Indirect addressing

To write an assembly language program to add tow numbers of 8-bit data stored in memory locations 4200H and 4201H and store the result in 4202H and 4203H using indirect addressing modes

Problem Analysis

To perform addition is 8085 one of the data should be in accumulator and another data can be in memory. After addition the sum will be in accumulator. The sum of two 8 bit data can be either 8 bits (sum only) or 9 bit (sum and carry). The accumulator can accommodate only the sum and if here is a carry, the 8085 will indicate by setting carry flag. Hence one of the registers is used to account for carry.

In algorithm register indirect addressing is used to indicate data. Here HL register is used to hold the address of the data and it is called pointer.

Algorithm 1

- Load the first data from memory to accumulator.
- Clear C – register.
- Add the content of A– register to Memory
- Check for carry. If carry = 1, go to step 5 or if carry = 0, go to step 6.
- Increment the C – register
- Store the sum in memory.
- Move the carry to accumulator and store in memory.
- Stop.

PROGRAM TO ADD TWO 8-BIT DATA

Memory address	Label	Instruction	Opcode	Comments
		LXI H,4200H		Load the HL with data address
		MOV A,M		Get 1st data in A
		INX H		
		MVI C,00H		Clear C register to account for carry.
		ADD M		Get the sum in A register
		JNC AHEAD		IF CY = 0, go to AHEAD
		INR C		IF CY = 1, increment C register

	AHEAD	INX H		
		MOV M, A		Store the sum in memory
		INX H		
		MOV C, M		Store the carry in memory
		HLT		Stop the Execution

Input		**Output**	
Address	**Data**	**Address**	**Data**
4200		**4202**	**(Sum)**
4201		**4203**	**(Carry)**

Arithmetic Operations in 8085 Microprocessors
[a] 8-bit Subtraction with borrow
[b] 8-bit Subtraction without borrow

To write an assembly language program to subtract two numbers of 8-bit data stored in memory 4200H and 4201H. Store the magnitude of the result in 4202H. If the result is positive store 00 in 4203H or if the result is negative store 01 in 4203H.

Problem Analysis

To perform subtraction in 8085 one of the data should be in accumulator and another data can be in any one of the general-purpose registers of in memory. After subtraction the result will be in accumulator. The 8085 performs 2's complement subtraction and then complements the carry. Therefore if the result is negative then carry flag is set and accumulator will have 2's complement of the result. Hence one of the register s used ;O account for sign of the result. To get the magnitude of the result again take 2's complement of the result.

Algorithm

- Load the subtrahend (the data to be subtracted) from memory to accumulator and move if to B-register.
- Load the minuend from memory to accumulator.
- Clear C-register to account for sign of the result.
- Subtract the content of B-register (subtrahend) from the content of accumulator (minuend).
- Check for carry. If carry = 1, go to step 6 or if carry = 0, go to step 7.
- Increment C-register, Complement the accumulator and store in memory.
- Store the difference in memory.
- Move the content of C-register (sign bit) to accumulator and store in memory.
- Stop.

PROGRAM TO SUBTRACT TWO 8-BIT DATA

Memory address	Label	Instruction		Opcode	Comments
		LDA	4201H		; Get the subtrahend in B register.
		MOV	B.A		
		LDA	4200H		;Get the minuend in A register
		MVI	C,00H		; Clear C register, to account for sign.
		SOB	B		; Get the difference in A register.
		JNC	AHEAD		; if CY = 0, then go to AHEAD
		INR	C		; if CY = 1, then increment C register
		CMA			;Get 2's complement of difference
		ADI	01H		;(result) in A register.
	AHEAD	STA	4202H		Store the result in memory
		MOV	A,C		
		STA	4203H		Store the sign bit in memory
		HLT			Stop the Execution

Sample data

Address	Input Data	Address	Output Data
4200	5E (Minuend)	4202	2A (Difference)
4201	34 (Subtrahend)	4203	00 (Sign bit)

8-bit Multiplication operations using 8085 Microprocessor

To write an assembly language program to multiply two numbers of 8 bit data stored in memory 420H and 4201H and store the product in 4202H and 4203H.

Problem Analysis

In 8085 the multiplication is performed as repeated additions. The initial value of sum is assumed as zero. One of the data is used as count (N) for number of additions to be performed. Another data is added to the sum N times, where N is the count. The result of the product of two 8 bit data will be 16 bits. Hence another register is used to account for overflow.

Algorithm

- Load the first data in HL pair and move to SP.
- Load the second data in HL and move to DE (count)
- Clear HL pair (Initial sum)
- Clear BC pair for overflow (carry)
- Add the content of SP to HL
- Check for carry. If carry = 1, go to step 7 or If carry = 0, go to step 8.
- Increment BC pair

- Decrement the count.
- Check whether count has reached zero.
- To check for zero of the count, move the content of E-register to A-register and logically, OR with D-register.
- Check the zero flag. If ZF = 0, repeat step 5 through 11 or If ZF = 1, go to next step
- Store the content of HL in memory. (Least significant 16 bits of the product)
- Move the content of C to L and B to H and store HL in memory. (Most significant) 16 bits of the product).
- Stop.

PROGRAM TO MULTIPLY TWO NUMBERS OF 16-BIT DATA

Memory address	Label	Instruction	Opcode	Comments
		LDA 4200H		;Get 1 st data in A
		MOV E,A		;Save 1st data in E
		LDA 4201H		;Get 2nd data in A
		MOV B,A		;save 2nd data in B
		LXI H,0000H		;Clear HL pair(initial sum=0)
		MVI D,00H		;Clear E for accounting overflow.
	NEXT:	DAD D		;Add the content of DE to sum(HL)
		DCR B		Decrement data 2 for every addition
		JNZ NEXT		;Repeat Addition until count is zero.
		SHLD 4202H		;Store the product in memory
		HLT		Stop the Execution

Sample data

Address	Input Data	Address	Output Data
4200	(Data-1)	4202	(First byte of product)
4201	(Data-2)	4203	(Second byte of product)

DIVISION OF TWO 8 BIT NUMBERS

Stored in memory location 4200H and 4201H. Store the remainder in 4202H and the quotient in 4203H.

Problem Analysis:

The division in 8085 is performed as repeated subtraction. The dividend is stored n A-register and divisor in B-register. The initial value of quotient is assumed as zero. Subtraction should be performed only when dividend is greater than divisor. Subtraction is continued until dividend is lesser than the divisor. For each subtraction quotient is incremented by one. Then store the quotient and remainder in memory.

Algorithm

- Load the divisor in accumulator and move if to B-register
- Load t5he dividend in accumulator.
- Clear C-register to account for quotient
- Check whether divisor is less than dividend
- If divisor is less than dividend, go to step 8, otherwise go to next step
- Subtract the content of B-register (quotient)
- Increment the content of C-register (quotient)
- Go to step 4
- Store the content of accumulator (remainder) in memory.
- Move the content of C-register (quotient) to accumulator and store in memory
- Stop.

PROGRAM TO DIVIDE TWO NUMBERS OF 8-BIT DATA

Memory address	Label	Instruction	Opcode	Comments
		LDA 4201H		
		MOV B,A		;Get the divisor in B register
		LDA 4200H		;Get the dividend in A register
		MVI C,00H		;Clear C register for quotient
	AGAIN:	CMP B		
		JC STORE		;If divisor is less than dividend go to store
		SUB B		;Subtract divisor from dividend. Increment
		INR C		;quotient by one for each subtraction.
		JMP AGAIN		
	STORE:	STA 4203H		;Store the remainder in memory
		MOV A,C		
		STA 4202H		;Stare the quotient in memory
		HLT		Stop the Execution

Sample data

Address	Input Data	Address	Output Data
4200	C9 (Divident)	4202	14 (Quotient)
4201	0A (Divisor)	4203	01 (Remainder)

Program for finding 8-bit Sum of an array of numbers

To write an assembly language program to add an array of data stored in memory from 4200H to (4200H +N). The first element of the array, gives the number of elements in the array. Store the result in 4300H and 4301H. Assume that sum does not exceeds 16 bit.

Problem Analysis

The number of bytes (data) N, is used as count for number of additions. The initial sum is assumed as zero. The HL register pair is used as pointer for data. Each element of the array is added to sum and for accounting overflow one of the register is used.

Algorithm

- Load the address of the first element of the array in HL pair (pointer).
- Move the count to B-register
- Clear C- register for carry.
- Clear accumulator for sum
- Increment the pointer (HL pair)
- Add the content of memory addressed by HL to accumulator.
- Check for carry = 1, go to step 8, or. If corn = 0, go to step 9
- Increment C- register
- Decrement the count
- Check for zero of the count. If ZF = 0 go to step 5 or If ZF = 1, go the next step
- Store the content of accumulator (LSB of sum)
- Move the content of C to accumulator. Store the content of accumulator in memory. (MSB of sum)
- Stop.

PROGRAM TO ADD AN ARRAY OF DATA

Memory address	Label	Instruction	Opcode	Comments
		LXI H,4200H		;Set Pinter for data
		MOV B,M		;Set counter for number of data
		MVI C,00H		;Clear C register to account for carry.
		XRA A		;Clear accumulator. Initial sum = 0
	REPT	INX H		
		ADD M		;Add on element of the array to sum
		JNC AHEAD		
		INR C		;If CY = 1, increment C register
	AHEAD	DCR B		
		JNZ REPT		;Repeat addition until count is zero
		STA 4300H		;Store LSB of sum in memory
		MOV A,C		
		STA 4301H		;Store MSB of sum in memory
		HLT		Stop the Execution

Sample data

Address	Input Data	Address	Output Data
4200	07 (Count)	4300	2D (LSB of sum)
4201	C2 (Data – 1)	4301	04 (MSB of sum)

4202	45 (Data – 2)		
4203	B3 (Data – 3)		
4204	F4 (Data – 4)		
4205	7C (Data – 5)		
4206	ED (Data – 6)		
4207	16 (Data – 7)		

To write an assembly language program to convert a two-digit BCD (8 bit) data to binary number. The BCD data is stored in 4200H and store the binary value in 4250H.

Program Analysis

The 2 digit BCD data will have units digit and tends digit. The tens digit (upper nibble) is multiplied by 0AH and the product is added to units digit (lower nibble.) The microprocessor performs binary arithmetic and so the result will be in binary.

Algorithm

1. Get the BCD data in A-register and save in E-register
2. Mask the lower nibble (units) of the BCD data in A-register
3. Rotate the upper nibble to lower nibble position and save in B-register
4. Clear the accumulator
5. Move OAH to C-register
6. Add B-register to A-register
7. Decrement C-register. If ZF = 0 go to step 6. If ZF = 1 , go to next step
8. Save the product in B-register
9. Get the BCD data in A-register from E-register and mask the upper nibble (tens)
10. Add the units (?A- register) to product (B-register)
11. Store the binary value (A-register)
12. Stop.

PROGRAM TO CONVERT 2 DIGIT BCD TO ENARY NUMBER

Memory address	**Label**	**Instruction**	**Opcode**	**Comments**
		LDA 4200H		;Get the data n A register
		MOV E,A		;and save in E register
		ANI F0H		;Mask the lower nibble (Units digit)
		RLC		;Rotate the upper nibble (Tens digit)
		RLC		;to lower nibble position and save in B
		RLC		
		RLC		
		MOV B,A		
		XRA A		;Clear accumulator
		MVI C,OAH		;Get the product of the digit multiplied
	REP:	ADD B		;by 0AH in A-register

		DCR C		
		JNZ REP		
		MOV B,A		;Save the product in B register
		MOV A,E		;Get the BCD data in A register
		ANI 0FH		;Mask the upper nibble (Tens digit)
		ADD B		;Get the sum of units digit and the product in B register
		STA 4250H		;Save the binary value in memory
		HLT		

Sample Data

Address	Input Data	Address	Output Data
4200	45 (BCD data)	4250	2D (Binary data)

To write an assembly language program to sort an array of data in ascending order. The array is stored in memory starting form 4200H. The first element of the array gives the count value for the number of elements in the array.

Problem Analysis

The algorithm for bubble sorting is given below. In bubble sorting of N-data, (N-1) comparisons are carried by taking two consecutive data at a time. After each comparison, the data are rearranged such that smallest among the two is in first memory location and the largest in the next memory location. (Here the data are rearranged within the two memory locations whose contents are compared).

When we perform (N-1) comparisons as mentioned above, for (N-1) times then the array consisting of N-data will be sorted in the ascending order.

Algorithm

1. Load the count value from memory to A-register and save it in B-register
2. Decrement B-register (B is a count for (N-1) repetitions)
3. Set HL pair as data address pointer
4. Set C-register as counter for (N-1) comparisons.
5. Load a data of the array in accumulator using the data address pointer
6. Increment the HL pair (data address pointer)
7. Compare the data pointed by HL wait accumulator
8. If carry flag is set (If the content of accumulator is smaller than memory) then go to step 10, otherwise go to next step
9. Exchange the content of memory pointed by HL and the accumulator
10. Decrement C-register. If zero flag is reset go to step 6 otherwise go to next step
11. Decrement B-register. If zero flag is reset go the step 3 otherwise go to next step
12. Stop.

PROGRAM TO SORT AN ARRAY OF DATA IN ASCENDING ORDER

Memory address	Label	Instruction	Opcode	Comments
		LDA	4200H	;Load the count value
		MOV	B,A	;Set counter for (N-1) repetitions
		DCR	B	;of (N-1) comparisons
	LOOP 2	LXI	H,4200H	;Set pointer for array
		MOV	C,M	;Set count for (N-1) comparisons
		DCR	C	
		INX	H	;Increment pointer
		MOV	A,M	;Get one data of array in A
	LOOP 1	INX	H	
		CMP	M	;Compare next data with A register
		JC	AHEAD	;If content of A is less than memory then go to AHEAD
		MOV	D,M	;If the content of A is greater than
		MOV	M,A	;then content of memory
		DCX	H	;pointed by HL and previous location
		MOV	M,D	
		INX	H	
	AHEAD	DCR	C	;Repeat comparisons until C count is zero
		JNZ	LOOP l	
		DCR	B	;Repeat until B count is zero
		JNZ	LOOP 2	
		HLT		Stop the Execution

Sample Data

Address	Data Array (Before sorting)	Address	Data Array (After sorting)
4200	07 (Count)	4200	07 (Count)
4201	AB (Data -1)	4201	34 (Data -1)
4202	92 (Data -2)	4202	4F (Data -2)
4203	84 (Data -3)	4203	69 (Data -3)
4204	4F (Data -4)	4204	84 (Data -4)
4205	69 (Data -5)	4205	92 (Data -5)
4206	F2 (Data -6)	4206	AB (Data -6)
4207	34 (Data -7)	4207	F2 (Data -7)

Example Program 16

To write an assembly language program to son than array of data in descending order. The array is stored in memory stored in memory starting from 4200H. The first element of the array gives the count value for the number of elements in the array.

Problem Analysis

The algorithm for bubble sorting is given below. In bubble sorting of N-data, (N-1) comparisons are carried by taking two consecutive data at as time. After each comparison, the data are rearranged such that largest among the two is in first memory location and the smallest in the next memory location. (Here the data are rearranged within the two memory locations whose contents are compared).

When we perform (N-1) comparisons as mentioned above, for (N-1) times then the array consisting of N-data will be sorted in descending order.

Algorithm

The algorithm is same as algorithm of example program 15 except step 8.

Step 8 : If carry flag is reset (If content of accumulator is larger than memory) then go to step 10, otherwise go to next step

PROGRAM TO SORT AN ARRAY OF DATA IN DESCENDING ORDER

Memory address	Label	Instruction	Opcode	Comments
		LDA	4200H	;Load the count value
		MOV	B,A	;Set counter for (N-1) repetitions
		DCR	B	;of (N-1) comparisons
	LOOP 2	LXI	H,4200H	;Set pointer for array
		MOV	C,M	;Set count for (N-1) comparisons
		DCR	C	
		INX	H	;Increment pointer
		MOV	A,M	;Get one data of array in A
	LOOP 1	INX	H	
		CMP	M	;Compare next data with A register
		JNC	AHEAD	;If content of A is less than memory then go to AHEAD
		MOV	D,M	;If the content of A is greater than
		MOV	M,A	;then content of memory
		DCX	H	;pointed by HL and previous location
		MOV	M,D	
		INX	H	

	AHEAD DCR	C	;Repeat comparisons until C count is zero
	JNZ	LOOP l	
	DCR	B	;Repeat until B count is zero
	JNZ	LOOP 2	
	HLT		Stop the Execution

Sample Data

Address	Data Array (Before sorting)	Address	Data Array (After sorting)
4200	07 (Count)	4200	07 (Count)
4201	AB (Data -1)	4201	F2 (Data -7)
4202	92 (Data -2)	4202	AB (Data -6)
4203	84 (Data -3)	4203	92 (Data -5)
4204	4F (Data -4)	4204	84 (Data -4)
4205	69 (Data -5)	4205	69 (Data -3)
4206	F2 (Data -6)	4206	4F (Data -2)
4207	34 (Data -7)	4207	34 (Data -1)

To write an assembly language program to search the smallest data in an array of N data sored in memory from 4200H to (4200H + N). The first element of the array gives the number of data in the array. Store the smallest data in 4300H.

Problem Analysis

The HL register pair is used as pointer for the array. One of the general propose register is used as count. A data in the array is moved to A-register and compared with next data. After each comparison, the smallest data is brought to accumulator. The comparisons are carried (N-1) time. The smallest data will be in A – register and store in memory.

Algorithm

- Load the address of the first element of the array in HL register pair. (Pointer)
- Move the count to B-register
- Increment the pointer
- Get the first data in accumulator.
- Decrement the count
- Increment the pointer
- Compare the content of memory addressed by HL pair with that of accumulator
- If carry = 1, go to step 10 or if carry = 0, go to step 9
- Move the content memory addressed HL to accumulator.
- Decrement the count.

- Check for zero of the count. If ZF = 0, Go to step 6, or If ZF = 1 go to next step
- Store the smallest data in memory.
- Stop.

PROGRAM TO SEARCH SMALLEST DATA IN AN ARRAY

Memory address	Label	Instruction	Opcode	Comments
		LXI H,4200H		;set pointer for array
		MOV B,M		;set count for number of elements in array
		INX H		
		MOV A,M		;Set 1st element of array as smartest data
		DCR B		;Decrement the count.
	LOOP	INX H		;Compare on element of array
		CMP M		;with current smallest data
		JC AHEAD		;If CY = 1, go to AHEAD
		MOV A,M		;If CY = 0 then content of memory
				:is smaller than A. Hence if CY = 0,
				;Make memory as smallest by moving to A
	AHEAD	DCR B		
		JNZ LOOP		; Repeat Comparison until count is zero
		STA 4300H		;Store the smallest data in memory.
		HLT		Stop the Execution

Sample data

Address	Input Data	Address	Output Data
4200	07 (Count)	4300	1 C (Smallest data in the array)
4201	42 (Data -1)		
4202	3A (Data -2)		
4203	1C (Data -3)		
4204	24 (Data -4)		
4205	B4 (Data -5)		
4206	25 (Data -6)		

Example Problem 13

To write an assembly language program to search the largest data in an array of N data stored memory from 4200H to (4200H + N). The first element of the array is the number of data (N) in the array. Store the largest data in 4300H.

Problem Analysis

The HL register pair is used as pointer for the array. One of the general purpose register is used as count. A data in the array is moved to A-register and compared with next data. After each comparison, the largest data is brought to A-register. The comparisons are performed (N-1) times. After (N-1) comparisons the largest data will be in A-register and store it in memory.

Algorithm

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

Flowchart for example program 13

PROGRAM TO SERCH LARGEST DATA IN AN ARRAY

Memory address	Label	Instruction		Opcode	Comments
		LXI	H,4200H		;set pointer for array
		MOV	B,M		;set count for number of elements in array
		INX	H		
		MOV	A,M		;Set 1st element of array as smartest data
		DCR	B		;Decrement the count.
	LOOP	INX	H		;Compare on element of array
		CMP	M		;with current smallest data
		JNC	AHEAD		;If CY = 1, go to AHEAD
		MOV	A,M		;If CY = 0 then content of memory is greater than A. Hence if CY = 0, Make memory as smallest by moving to A
	AHEAD	DCR	B		
		JNZ	LOOP		; Repeat Comparison until count is zero
		STA	4300H		;Store the smallest data in memory.
		HLT			Stop the Execution

Sample data

Address	Input Data	Address	Output Data
4200	07 (Count)	4300	FC (Largest data in the array)
4201	62 (Data -1)		
4202	7D (Data -2)		
4203	FC (Data -3)		
4204	24 (Data -4)		
4205	C2 (Data -5)		
4206	0F (Data -6)		

To write an assembly language program to convert an array of ASSCII codes to corresponding binary (Hex) value. The ASCII array is stored starting from 4200H. The first element of the array gives the number of elements in the array.

Problem Analysis

The Hex digit 0 through 9 are represented by 30H to 39H in ASCII. Hence for ASSCII code 30H to 39H if we subtract 30H then we will get the corresponding binary (Hex) value. The Hex digit A through F are represented by 41H to 46H in ASSCII. Hence for ASCII code 41H to 46H we have to subtract 37H to get corresponding binary (Hex) value.

In the following algorithm, a subroutine have been written to subtract either 30H to 37H from the given data.

Algorithm

1. Set HL pair as pointer for ASSCII array
2. Set D-register as counter for number of data in the array
3. Set BC pair as pointer for binary (Hexa) array
4. Increment HL pair and move a data of ASCII array to A-register
5. Call subroutine BIN to find the binary (Hexa) value
6. The binary (Hexa)value available in A-register is stored in memory
7. Increment BC pair
8. Decrement D-retgister. If ZF = 0, then go to step 4. If ZF = 1, then stop

Algorithm for subroutine BIN

1. Subtract 30H from A-register
2. Compare the content of A-register with 0AH
3. If CY = 1 go to step 5. If CY = 0, go to next step
4. Subtract 07H from A-register
5. Return to main program

PROGRAM TO CONVERT ASCII CODE TO BINARY VALUE

Memory address	Label	Instruction		Opcode	Comments
		LXI	H,4200H		;Set pointer for ASCII array
		MOV	D,M		;Set count for number of data
		LXI	B,4300H		;Set pointer for binary (Hexa) array
	LOOP	INX	H		
		MOV	A,M		;Get an ASCII data in A register
		CALL	BIN		;Call subroutine to get binary
		STAX	B		;value in A and store in memory
		INX	B		;Increment the binary array pointer
		DCR	D		
		JNZ	LOOP		;Repeat conversion until count is zero.
		HLT			

SBROUTINE BIN

Memory address	Label	Instruction		Opcode	Comments
	BIN	SUI	30H		;Subtract 30h from the data
		CPI	0AH		
		RC			;If CY = 1, Return to main program
		SUI	07H		;If data is greater than 0AH then subtract 07H
		RET			return to main program

Sample Data

ASCII array [source array]		Binary (Hex) array [Destination array]	
4200	07 (Count)	4300	01
4201	31	4301	0B
4202	42	4302	05
4203	35	4303	0F
4204	46	4304	0C
4205	43	4305	09
4206	39	4306	08
4207	38		

Ex. No: 1 **ARITHMETIC & LOGICAL OPERATIONS**

ARITHMETIC OPERATIONS

AIM:

To write and execute an assembly language program for add, subtract, multiply and divide two 16 bit unsigned numbers in 8086 kit and MASM.

APPARATUS:

1. 8086 microprocessor kit/MASM---- 1
2. Power card---- 1
3. Keyboard --- 1
4. PC with Intel/AMD Processor----1

ALGORITHM:

1. Load the First Data in AX-register.
2. Load the First Data in BX-register.
2. Add the two data and get the sum in AX-register.
3. Store the sum and carry in memory locations.
4. Stop the program.

By Using KIT:

ADDRESS	LABEL	MNEMONICS	OPCODE	COMMENT
1100		MOV CX,0000H		Initialize counter CX
1103		MOV AX,[1200]		Get the first data in AX register.
1106		MOV BX, [1202]		Get the second data in BX register.
110A		ADD AX,BX		Add the contents of both the register AX & BX
110C		JNC L1		Check for carry
110E		INC CX		If carry exists, increment the CX
110F	LI	MOV [1206],CX		Store the carry
1113		MOV [1204], AX		Store the sum
1116		HLT		Stop the program

OUTPUT FOR ADDITION:

	ADDRESS	DATA
INPUT	1200 1201 1202 1203	

OUTPUT	1204 1205 1206	

16- BIT SUBTRACTION:

ALGORITHM:

- Initialize the MSBs of difference to 0
- Get the first number
- Subtract the second number from the first number.
- If there is any borrow, increment MSBs of difference by 1.
- Store LSBs of difference.
- Store MSBs of difference.

PROGRAMM

By using 8086 kit:

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENT
1100			***MOV CX,0000H***	Initialize counter CX
1103			***MOV AX,[1200]***	Get the first data in AX register
1106			***MOV BX, [1202]***	Get the second data in BX register.
110A			***SUB AX,BX***	Subtract the contents of both the register AX & BX
110C			***JNC STORE***	Check the Borrow.
110E			***INC CX***	If carry exists, increment the CX
110F			**NOT AX**	Complement the AX Register
1111			**INC AX**	Increment the AX
1112		STORE	***MOV [1206],CX***	Store the Borrow.
1117			***MOV [1204], AX***	Store the difference.
111B			***HLT***	Stop the program

OUTPUT FOR SUBTRACTION:

	ADDRESS	DATA

INPUT	**1200** **1201** **1202** **1203**	
OUTPUT	**1204** **1205** **1206**	

16- BIT MULTIPLICATION ALGORITHM:

1. Get the multiplier.
2. Get the multiplicand
3. Initialize the product to 0.
4. Product = product + multiplicand
5. Decrement the multiplier by 1.
6. If multiplicand is not equal to 0, repeat from step (d) otherwise store the product.

PROGRAM:

By using 8086 kit:

ADDRESS	OPCODE	LABEL	MNEMONIC	COMMENTS
1100	C7,C0, 4444		MOV AX,4444H	Load AX-register with 1st data
1104	C7, C3, 4444		MOV BX,4444H	Load BX-register with 2nd data
1108	F7, E3		MUL BX	Multiply the contents of AX with BX-register
110A	C7, C6, 00,12		MOV SI,1200H	Assign SI-reg to 1200H
110E	89, 14		MOV [SI],AX	Store the result
1110	C7, C6, 02,12		MOV SI,1202H	Assign SI-reg to 1202H
1114	89,04		MOV [SI],DX	Store the result
1116	F4		HLT	Stop the program

OUTPUT:

INPUT		OUTPUT	
Register	**Data**	**Address**	**Data**
		1200	
		1201	
		1202	
		1203	

32- IT BY 16-BIT DIVISION

ALGORITHM:

1. Load the First Data in AX-register.
2. Load the First Data in BX-register.
3. Subtract the content of BX-reg from AX-register.
4. Store the result in memory location.
5. Stop the program.

PROGRAM:

i) By using 8086 kit:

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENTS
1100	C7, C2,00,00		MOV DX,0000H	Initialize DX-register with Lsb of Dividend
1104	C7, C0,88,88		MOV AX,8888H	Load AX-register with Msb of Dividend
1108	C7, C1,44,44		MOV CX, 4444H	Load CX-register with Divisor
110C	F7, F1		DIV CX	Divide AX by CX-register
110E	C7, C6,00,12		MOV SI,1200H	Assign SI-reg to 1200H
1112	89, 04		MOV [SI],AX	Store the Quotient
1114	C7, C6,02,12		MOV SI,1202H	Assign SI-reg to 1202H
1118	89, 14		MOV [SI],DX	Store the Remainder
111A	F4		HLT	Stop the program

OUTPUT:

INPUT		OUTPUT	
Register	**Data**	**Address**	**Data**
AX		**1200H**	**(quotient)**
CX		**1201H**	**(quotient)**
		1202H	**(remainder)**
		1203H	**(remainder)**

LOGICAL OPERATION

AIM:

To write and execute an assembly language program for performing logical OR, AND, NAND operation in 8086 kit and MASM.

APPARATUS:

1. 8086 microprocessor kit/MASM---- 1
2. Power card---- 1
3. Keyboard---- 1
4. PC with Intel/AMD Processor--- 1

LOGICAL OR OPERATION

ALGORITHM:

1. Load the First Data in AL-register.
2. Load the Second Data in BL-register.
3. Logically OR the content of AL with BL-register.
4. Store the result in memory location.
5. Stop the program

PROGRAM:

i) By using 8086 kit:

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENTS
1100	C6,C0,85		MOV AL,85H	Load AL-register with 1st Data
1103	C6,C3,99		MOV BL,99H	Load BX-register with 2nd Data
1106	08,D8		OR AL, BL	OR the contents of AL with BL-Register
1108	C7,C6,00,12		MOV SI,1200H	Assign SI-reg to 1200H
110C	89,04		MOV [SI],AX	Store the Result
110E	F4		HLT	Stop the program

OUTPUT:

INPUT		OUTPUT	
Register	**Data**	**Address**	**Data**
		1200H	
		1201H	

LOGICAL AND OPERATION

ALGORITHM:

1. Load the First Data in AL-register.
2. Load the Second Data in BL-register.
3. Logically AND the content of AL with BL-register.
4. Store the result in memory location.
5. Stop the program

PROGRAM:

i) By using 8086 kit:

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENTS
1100			Org 1100h	Starting address of the program
1100	C6,C0,85		MOV AL,85H	Load AL-register with 1st Data
1103	C6,C3,99		MOV BL,99H	Load BX-register with 2nd Data
1106	20,D8		AND AL, BL	AND the contents of AL with BL-Register
1108	C7,C6,00,12		MOV SI,1200H	Assign SI-reg to 1200H
110C	89,04		MOV [SI],AX	Store the Result
110E	F4		HLT	Stop the program

OUTPUT:

INPUT		OUTPUT	
Register	**Data**	**Address**	**Data**
		1200H	
		1201H	

RESULT:

Thus an assembly language program for performing logical OR, AND, NAND operation was executed using MASM and 8086 kit.

Ex. No: 2 **MOVE A DATA BLOCK WITHOUT OVERLAP**

AIM:

To write and execute an assembly language program for transferring data from one block to another block without overlapping using 8086 kit and MASM.

APPARATUS:

1. 8086 microprocessor kit/MASM---- 1
2. Power card---- 1
3. Keyboard---- 1
4. PC with Intel/AMD Processor--- 1

ALGORITHM:

1. Initialize counter.
2. Initialize source block pointer.
3. Initialize destination block pointer.
4. Get the byte from source block.
5. Store the byte in destination block.
6. Increment source, destination pointers and decrement counter.
7. Repeat steps 4, 5 and 6 until counter equal to zero.
8. Stop.

PROGRAM:

i) By using 8086 kit:

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENTS
1100			Org 1100h	Starting address of the program
1100	C7 C6 0012		MOV SI, 1200H	Initialize the source address.
1104	C7 C7 0013		MOV DI,4700H	Initialize the destination address.
1108	C7 C1 0600		MOV CX,0006 H	Initialize count value to the count register.
110C	88,25		MOV [DI], AH	Move content of AH to DI-register.
110F	FC	REPEAT:	CLD	Clear the direction flag.
1110	A4		MOVSB	Move the string byte.
1111	E2,F3		LOOP REPEAT	Unconditional loop to address specified by the label REPEAT.
1113	F4		HLT	Stop the program

OUTPUT:

INPUT		OUTPUT	
Address	**Data**	**Address**	**Data**
4600.		4700.	
4601.		4701.	
4602.		4702.	
4603.		4703.	
4604.		4704.	

4605.		4705.	

VIVA QUESTIONS:

1. What is the fabrication technology used for 8086?
2. What are the functional units available in 8086 architecture?
3. Write the flags of 8086.
4. What are control bits?
5. What are the flag manipulation instructions of 8086?
6. What is Macro?
7. Which bus controller used in maximum mode of 8086?
8. What is the size of data bus and address bus in 8086?
9. What are the various segment registers in 8086?
10. What is the maximum memory addressing capability of 8086?

RESULT:

Thus an assembly language program for transferring data from one block to another block without overlapping was executed using 8086 kit and MASM.

Ex. No: 4 a) **STRING OPERATIONS**

AIM:

To write and execute an assembly language program to find the length and copying string, using 8086 kit and MASM.

APPARATUS:

1. 8086 microprocessor kit/MASM---- 1
2. Power card---- 1
3. Keyboard---- 1
4. PC with Intel/AMD Processor--- 1

STRING LENGTH

ALGORITHM:

1. Load the source and destination index register with starting and the ending address respectively.
2. Load the terminate data value as "FF".
3. Load the content of source index to the AL register.
4. Increment SI register and compare with register AH.
5. If it is non- zero value, increment the memory location by using the control instruction to store the data.
6. Compare the string byte with FF, if it is not equal, repeat.
7. Count the string byte till zero flag is set. If zero flag is set then store the count value to the memory.
8. Terminate the program when FF is matched.
9. Stop

PROGRAM:

i) By using 8086 kit:

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENTS
1100	C7,C6,00,12		MOV SI,1200H	SI ←1200
1104	C7,C7,FF,FF		MOV DX, FFFFH	DX←FFFF
1108	C6,C4,FF		MOV AH, FFH	AH← FF, check FF
110B	42	NOEND:	INC DX	Increment the DX reg by 1
110C	8A,04		MOV AL,[SI]	AL ←[SI]
110E	12		INC SI	Increment the SI reg. by 1
110F	38,C4		CMP AH,AL	Compare the content of AH & AL
1011	75,FB		JNZ : NOEND	If the Compared result is not „0" go to by the address specified label **NOEND.**
1013	89,16,00,11		MOV [1110],DX	[1110] ←DX ie. store the result
1017	F4		HLT	Terminate the program

INPUT:

ADDRESS (Hex)	Data (Hex)
1200	
1201	
1202	
1203	

1204	
1205	
1206	
1207	

OUTPUT:

ADDRESS (Hex)	String length Hex)
1110	

STRING COPYING

ALGORITHM:

1. Load the source and destination index register with starting and the ending address respectively.
2. Initialize the counter with the total number of words to be copied.
3. Clear the direction flag for auto incrementing mode of transfer.
4. Use the string manipulation instruction MOVSW with the prefix REP to copy a string from source to destination.
5. Stop

PROGRAM:

i) By using 8086 kit:

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENTS
1100	C7,C6,00,20		MOV SI,2000H	Initialize the source address.
1103	C7,C7,00,21		MOV DI,2110H	Initialize the destination address.
1107	C7,C1,06,00		MOV CX,0006 H	Initialize count value to the count register.
110B	C6,C4,55		MOV AH,55H	Move 55 to AH register.
110E	FC	REPEAT:	CLD	Clear the direction flag.
110F	A4		MOVSB	Move the string byte.
1010	E2,FB		LOOP:REPEAT	Unconditional loop to address specified by the label REPEAT.
1013	F4		HLT	Stop the program.

OUTPUT:

INPUT		**OUTPUT**	
ADDRESS	DATA	**ADDRESS**	**DATA**
2000		**2110**	
2001		**2101**	
2002		**2102**	
2003		**2103**	
2004		**2104**	
2005		**2105**	

RESULT:

Thus an assembly language program for copying, finding the length of a string was implemented and its output was verified.

Ex. No: 7 TRAFFIC LIGHT CONTROL

AIM:

To write and execute an assembly language Program for interfacing the Traffic Light Controller in 8086 kit.

APPARATUS:

1. 8086 microprocessor kit/MASM---- 1
2. Traffic Light Controller Interface board ----1
3. Power card---- 1
4. Keyboard---- 1
5. PC with Intel/AMD Processor--- 1

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENTS
1100		START:	MOV AL,80H	Move 80 to AL-register
1102			OUT CONTROL,AL	Copy the AL-reg content to output port.
1104			MOV SI, LOOKUP	Copy the lookup data value to BX-register
1107		REPEAT	MOV AL, [SI]	Copy the direction of rotation value to SI-reg.
110A			OUT PORTA,AL	Call to OUT port
			INC SI	
110D			MOV AL, [SI]	Copy the contents of SI-reg to Al-register
110F			OUT PORTB, AL	Move the AL-reg value to PORTA
1011			INC SI	
1014			MOV AL, [SI]	
1015			OUT PORTC, AL	Call Delay program
1016			INC SI	Increment SI-register
1019			MOV DI, 00040H	Move 40h to DI-register
101B		A	MOV DX, 0FFFFH	Move 0FFFFh to DX-register
101D		A1	DEC DX	Decrement DX-register
1020			JNZ A1	Jump on no zero to A1.
1021			DEC D1	Decrement D1-register
1022			JNZ A	Jump on no zero to A
1025			JMP REPEAT	Jump to repeat
1027		LOOKUP:	DB 12H,27H,44H,10H	
			2BH,92H,10H,9DH	
			84H,48H,2EH,84H	

VIVA QUESTIONS:

1. Which Segment is used to store interrupt and subroutine return address registers?
2. What is the difference between instructions RET & IRET?
3. What .model small means?
4. Difference between small, medium, tiny, huge?
5. What is dd, dw, db?
6. What is the function of 01h of Int 21h?
7. What is the function of 02h of Int 21h?
8. What is the function of 09h of Int 21h?
9. What is the function of 0Ah of Int 21h?
10. What is the function of 4ch of Int 21h?

RESULT:

Thus the assembly language program for interfacing the traffic light controller was implemented in 8086 kit.

Ex.No:8 **STEPPER MOTOR INTERFACING**

AIM:

To write and execute an assembly language Program to run a stepper motor at different speed, and to control its speed of direction.

APPARATUS:

1. 8086 microprocessor kit/MASM ---- 1
2. Stepper Motor ----1
3. Stepper Motor Interface board ---- 1
4. Power card --- 1
5. Keyboard ----1
6. PC with Intel/AMD Processor --- 1

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENTS
1100			Org 1100H	Starting address of the program
1100	C7,C7,18,10	Ahead	MOV DI, 1018	Copy the data 1018 to DI-reg
1104	C6, C1,04		MOV CL, 04	Copy the value 04 to CL- register
1107	8A, 05	Loop1	MOV AL, [DI]	Copy the content of DI-reg to AL-register
1109	E6 , C0		OUT C0, AL	The content of AL is moved to Out port
110B	C7,C2,10,10		MOV DX, 1010	Copy the data 1010 to DX-reg
110F	4A	loop	DEC DX	Decrement DX-register
1010	75,FD		JNZ loop	Jump on no zero to loop
1012	47		INC DI	Increment DI-register
1013	E2,F2		LOOP loop1	Loop to label loop1
1015	E9,E8, FF		JMP Ahead	Jump to ahead
1018	09,05,06,0A	TABLE	09 05 06 0A	
			HLT	Stop the program.

DIRECTION CONTROL OF STEPPER MOTOR

PROGRAM:

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENTS
1100	C6,C3, 20		MOV BL, 20	Copy the data 20h to BL-register.
1103	C7,C7,3F,10		MOV DI, 103F	Copy the data 103F to DI-reg
1107	E8 1B 00		CALL 1025	Call subroutine program
110A	FE CB		DEC BL	Decrement BL-register
110C	75 F5		JNZ 1103	Jump on no zero to label 1103
110E	E8 2A 00		CALL 103B	Call subroutine program

1011	C6 C3 20		MOV BL, 20	Copy the data 20 to BL-register
1014	C7 C7 43 10		MOV DI, 1043	Copy the data 1043 to DI-reg
1018	E8 0A 00		CALL 1025	Call subroutine program
101B	FE CB		DEC BL	Decrement BL-register
101D	75 75		JNZ 1014	Jump on no zero to 1014
101F	E8 19 00		CALL 103B	Call subroutine program
1022	E9 DB FF		JMP 1100	Jump to label 1100
1025	C6 C1 04		MOV CL, 04	Copy the data 04 to CL-register
1028	8A 05		MOV AL, [DI]	Copy the content of DI to AL-register
102A	E6 C0		OUT C0, AL	Assign the content of AL to Output
102C	C7 C2 10 10		MOV DX, 1010	Copy the content 1010 to Dx-register
1030	4A		DEC DX	Decrement DX-register
1031	75 FD		JNZ 1030	Jump on no zero to label 1030
1033	47		INC DI	Increment DI-register
1034	E2 F2		LOOP 1028	Loop to label 1028
1036	C3		RET	Return from subroutine
1037	C7,C2,FF,FF		MOV DX, 0FFFF	Copy FFFF in DX-register
103B	4A		DEC DX	Decrement DX-register
103C	75 FD		JNZ 103B	Jump on no zero 103B
103E	C3		RET	Return from subroutine
103F	09,05,06,0A		FORWARD DATA	To rotate in forward direction
1043	0A,06,05, 09		REVERSE DATA	To rotate in Reverse direction
1047			HLT	Stop the program.

OUTPUT

VIVA QUESTIONS:

1. What MASM is?
2. What do u mean by emulator?
3. What is SI, DI and their functions?
4. What do you mean by IVT in 8086?
5. Why we indicate FF as 0FF in program?
6. .stack 110 means?
7. What do you mean by 20 dup (0)?
8. What is 8087? How it is different from 8086?
9. What do you mean by debugger?
10. When divide overflow error occurs?

RESULT:

Thus an assembly language Program to run the stepper motor in both forward and reverse direction with delay was executed and its output was verified.

Ex.No:10 **KEYBOARD AND DISPLAY**

AIM:

To write and execute an assembly language Program to display the rolling message "HELP US "in the display.

APPARATUS:

1. 8086 microprocessor kit/MASM ---- 1
2. 8279 Interface board---- 1
4. Power card --- 1
5. Keyboard ----1
6. PC with Intel/AMD Processor ---1

ALGORITHM:

Display of rolling message "HELP US "

1. Initialize the counter
2. Set 8279 for 8 digit character display, right entry
3. Set 8279 for clearing the display
4. Write the command to display
5. Load the character into accumulator and display it
6. Introduce the delay
7. Repeat from step 1.

PROGRAM:

ADDRESS	OPCODE	LABEL	PROGRAM	COMMENTS
1100	BE 00	START	MOV SI,1200H	Initialize array
1103	B9 0F		MOV CX,000FH	Initialize array size
1106	B0 10		MOV AL,10	Store the control word for display mode
1108	E6 C2		OUT C2,AL	Send through output port
110A	B0 CC		MOV AL,CC	Store the control word to clear display
110C	E6 C2		OUT C2,AL	Send through output port
110E	B0 90		MOV AL,90	Store the control word to write display
1010	E6 C2		OUT C2,AL	Send through output port
1012	8A 04	NEXT	MOV AL,[SI]	Get the first data
1014	E6 C0		OUT C0,AL	Send through output port
1016	E8 E7		CALL DELAY	Give delay
1019	12		INC SI	Go & get next data
101A	E2 F6		LOOP NEXT	Loop until all the data"s have been taken
101C	EB E2		JMP START	Go to starting location
1500	BA FF A0	DELAY	MOV DX,0A0FFH	Store 16bit count value
1503	4A	LOOP1	DEC DX	Decrement count value
1504	74 FD		JNZ LOOP1	Loop until count values becomes zero
1506	C3		RET	Return to main program

LOOK-UP TABLE:

1200	98	68	7C	C8
1204	FF	1C	29	FF

OUTPUT:

MEMORY LOCATION	7-SEGMENT LED FORMAT								HEX DATA
	d	c	b	a	dp	e	g	f	
1200H	1	0	0	1	1	0	0	0	**98**
1201H	0	1	1	0	1	0	0	0	**68**
1202H	0	1	1	1	1	1	0	0	**7C**
1203H	1	1	0	0	1	0	0	0	**C8**
1204H	1	1	1	1	1	1	1	1	**FF**
1205H	0	0	0	0	1	1	0	0	**1C**
1206H	0	0	1	0	1	0	0	1	**29**
1207H	1	1	1	1	1	1	1	1	**FF**

VIVA QUESTIONS:

1. What are the types of interfacing?
2. Compare memory interfacing and IO interfacing.
3. What is the difference between memory mapped IO and IO mapped IO interfacing?
4. What IC 8279 is?
5. What are the tasks involved in keyboard interface?
6. What is scanning in keyboard and what is scan time?
7. What is the difference between 2-key and n-key rollover?
8. What is the control registers available in 8279?
9. What is key debouncing?
10. What are the command words available in 8279?

RESULT:

Thus the rolling message “HELP US” is displayed using 8279 interface kit.

Ex.No:12 **SERIAL & PARELLEL INTERFACE**

AIM:

To write and execute assembly language program to establish serial & parallel communication between two 8086 kits.

APPARATUS REQUIRED:

1. 8086 microprocessor kit/MASM---- 2
2. Power card---- 1
3. Keyboard---- 1
4. PC with Intel/AMD Processor--- 1

SERIAL INTERFACE

ALGORITHM:

1. Initialize 8253 and 8251 to check the transmission and reception of a character
2. Initialize8253 to give an output of 150 KHz at channel 0 which will give a 9600 baud rate of 8251.
3. The command word and mode word is written to the 8251 to set up for subsequent operations
4. The status word is read from the 8251 on completion of a serial I/O operation, or when the host CPU is checking the status of the device before starting the next I/O operation

PROGRAM:

TRANSMITTER:

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENTS
1100	C6,CO,36		MOV AL,36	
1103	E6,16		OUT 16,AL	
1105	C6,CO,0A		MOV AL,0A	
1108	E6,10		OUT 10,A1	
110A	C6,C0,00		MOV A1,00	
110D	E6,10		OUT 10,A1	
110F	E4,0A	LOOP	IN A1,0A	
1012	F6,C0,04		TEST A1,04	
1015	74,F9		JZ LOOP	
1017	C6,C0,41		MOV A1,41	41 is data
101A	E6,08		OUT 08,A1	
101C			HLT	

RECEIVER:

ADDRESS	OPCODE	LABEL	MNEMONICS	COMMENTS
1100	C6,CO,36		MOV AL,36	
1103	E6,16		OUT 16,AL	
1105	C6,CO,0A		MOV AL,0A	
1108	E6,10		OUT 10,A1	
110A	C6,C0,00		MOV A1,00	
110D	E6,10		OUT 10,A1	
110F	E4,0A	LOOP1	IN A1,0A	
1012	F6,C0,02		TEST A1,02	
1015	74,F9		JZ LOOP1	
1017	E4,08		IN A1,08	
1019	88,06,00,11		MOV [1110],A1	
101D	F4		HLT	

OUTPUT:

INPUT(Transmitter)		OUTPUT (Receiver)	
Address	**Data**	**Address**	**Data**

PARALLEL INTERFACE

ALGORITHM:

1. Initialize accumulator to hold control word
2. store control word in control word register
3. Read data port A.
4. Store data from port A in memory
5. Place contents in port B

PROGRAM:

TRANSMITTER:

ADDRESS	OPCODE	LABEL	PROGRAM	COMMENTS
			MOV AL,80	
			OUT 56,AL	
			MOV AL,DATA	
			OUT 50,AL	
			HLT	

RECEIVER:

ADDRESS	OPCODE	LABEL	PROGRAM	COMMENTS
			MOV AL,90	
			OUT 56,AL	
			IN AL,50	
			MOV [1110],AL	
			HLT	

OUTPUT:

INPUT(Transmitter)		OUTPUT (Receiver)	
Address	Data	Address	Data

VIVA QUESTIONS:

1. What is Baud rate?
2. What is the difference between synchronous and asynchronous communication?
3. What is the expansion of RS?
4. What is the expansion of USART?
5. What is the role of RS-232?
6. What type of interface is 8255?
7. What are the different types of modes in 8255?
8. What are the features of mode 0 in 8255?
9. What are the features of mode 1 in 8255?
10. What are the features of mode 2 in 8255?

RESULT:

Thus an assembly language program to establish a serial & parallel communication between two 8086 kits was implemented and its output was verified.

EXP. NO: 13 a) **ADC INTERFACE**

AIM:

To write and execute an assembly language program to convert an analog signal into a digital signal using an ADC interfacing.

APPARATUS REQUIRED:

1. 8086 microprocessor kit/MASM ----1
2. ADC Interface board ----1
4. Power card --- 1
5. Keyboard ----1
6. PC with Intel/AMD Processor ---1

ALGORITHM:

(i) Select the channel and latch the address.
(ii) Send the start conversion pulse.
(iii) Read EOC signal.
(iv) If EOC = 1 continue else go to step (iii)
(v) Read the digital output.
(vi) Store it in a memory location.

PROGRAM:

ADDRESS	OPCODE	LABEL	PROGRAM	COMMENTS
1100	B0 10		MOV AL,00	Load accumulator with value for ALE high
1102	E6 C8		OUT 0C8H,AL	Send through output port
1104	80 18		MOV AL,08	Load accumulator with value for ALE low
1106	E6 C8		OUT 0C8H,AL	Send through output port
1108	B0 01		MOV AL,01	Store the value to make SOC high in the accumulator
110A	E6 D0		OUT 0D0H,AL	Send through output port
110C	B0 00		MOV AL,00	Introduce delay
110E	B0 00		MOV AL,00	
1010	B0 00		MOV AL,00	
1012	B0 00		MOV AL,00	Store the value to make SOC low the accumulator
1014	E6 D0		OUT 0D0H,AL	Send through output port
1016	E4 D8	L1	IN AL, 0D8H	Read the EOC signal from port & check for end of conversion
1018	24 01		AND AL,01	
101A	3C 01		CMP AL,01	

101C	75 F8		JNZ L1	If the conversion is not yet completed, read EOC signal from port again
101E	E4 C0		IN AL,0C0H	Read data from port
1020	BB 00 11		MOV BX,1110	Initialize the memory location to store data
1022	88 07		MOV [BX],AL	Store the data
1024	F4		HLT	Stop

OUTPUT:

ANALOG VOLTAGE	DIGITAL DATA ON LED DISPLAY	HEX CODE IN MEMORY LOCATION

RESULT:

Thus the ADC was interfaced with 8086 and the given analog inputs were converted into its digital equivalent.

EXP. NO: 13 b) **DAC INTERFACE**

AIM:

To write and execute an assembly language program for digital to analog conversion.

APPARATUS REQUIRED:

1. 8086 microprocessor kit/MASM ---- 1
2. DAC Interface board ----1
3. Power card --- 1
4. Keyboard ----1
5. PC with Intel/AMD Processor ---1

ALGORITHM:

Square Waveform:

(i) Send low value (00) to the DAC.
(ii) Introduce suitable delay.
(iii) Send high value to DAC.
(iv) Introduce delay.
(v) Repeat the above procedure.

Saw-tooth waveform:

(i) Load low value (00) to accumulator.
(ii) Send this value to DAC.
(iii) Increment the accumulator.
(iv) Repeat step (ii) and (iii) until accumulator value reaches FF.
(v) Repeat the above procedure from step 1.

Triangular waveform:

(i) Load the low value (00) in accumulator.
(ii) Send this accumulator content to DAC.
(iii) Increment the accumulator.
(iv) Repeat step 2 and 3 until the accumulator reaches FF, decrement the accumulator and send the accumulator contents to DAC.
(v) Decrementing and sending the accumulator contents to DAC.
(vi) The above procedure is repeated from step (i)

PROGRAM: SQUARE WAVE

ADDRESS	OPCODE	LABEL	PROGRAM	COMMENTS
1100	B0 00	L2	MOV AL,00H	Load 00 in accumulator
1102	E6 C8		OUT C8,AL	Send through output port
1104	E8 09 00		CALL L1	Give a delay
1107	B0 FF		MOV AL,FFH	Load FF in accumulator
1109	E6 C8		OUT C8,AL	Send through output port
110B	E8 02 00		CALL L1	Give a delay
110E	EB F0		JMP L2	Go to starting location
1010	B9 FF C5	L1	MOV CX,05FFH	Load count value in CX register
1013	E2 FE	L3	LOOP L3	Decrement until it reaches zero
1015	C3		RET	Return to main program

PROGRAM: SAWTOOTH WAVE

ADDRESS	OPCODE	LABEL	PROGRAM	COMMENTS

1100	B0 00	L2	MOV AL,00H	Load 00 in accumulator
1102	E6 C0	L1	OUT C0,AL	Send through output port
1104	FE C0		INC AL	Increment contents of accumulator
1106	75 FA		JNZ L1	Send through output port until it reaches FF
1108	EB F6		JMP L2	Go to starting location

PROGRAM: TRIANGULAR WAVE

ADDRESS	OPCODE	LABEL	PROGRAM	COMMENTS
1100	B0 00	L3	MOV AL,00H	Load 00 in accumulator
1102	E6 C8	L1	OUT C8,AL	Send through output port
1104	FE C0		INC AL	Increment contents of accumulator
1106	75 F8		JNZ L1	Send through output port until it reaches FF
1108	B0 FF		MOV AL,0FFH	Load FF in accumulator
110A	E6 C8	L2	OUT C8,AL	Send through output port
110C	FE C0		DEC AL	Decrement contents of accumulator
110E	75 F8		JNZ L2	Send through output port until it reaches 00
1110	EB EA		JMP L3	Go to starting location

OUTPUT:

WAVEFORM GENERATION

WAVEFORMS	AMPLITUDE	TIME PERIOD
Square wave		
Saw-tooth wave		
Triangular wave		

VIVA QUESTIONS:

1. What are the different types of ADC?
2. What is meant by conversion time?
3. What are the different types of ADC techniques available?
4. Name the ADC IC available using serial interface technique?
5. Name the ADC IC available using parallel interface technique?
6. What are the different types of DAC techniques available?
7. Name the DAC IC available
8. What is resolution?
9. What is settling time?
10. What is range of operation in DAC?

RESULT:

Thus the DAC was interfaced with 8086 and different waveforms have been generated.

EXP. NO: 14 a) **ARITHMETIC OPERATIONS USING 8051**

AIM:

To write and execute an assembly language program for Add, subtract, multiply and division of two 8-bit numbers using 8051.

APPARATUS:

1. 8051 microcontroller kit---- 1
2. Power card---- 1
3. Keyboard --- 1
4. PC with Intel/AMD Processor----1

ALGORITHM:

1. Load the First Data in A-register.
2. Load the Second Data in B-register.
3. Add the two data with carry.
4. Store the sum in memory location.
5. Stop the program.

PROGRAM:

ADDITION

ADDRESS	OPCODE	LABEL	PROGRAM	COMMENTS
4110	74,05		MOV A,#data	Load data 1 in accumulator.
4102	75,F0,05		MOV B,#data	Load data 2 in B-register
4105	35,F0		ADDC A,B	Add the contents of accumulator and B-reg with carry.
4107	90,11,00		MOV DPTR,#1100_H	Initialize DPTR with address 1100_H
410A	F0		MOVX @ DPTR,A	Store the result in 1100_H
410B	80, FE	STOP:	SJMP STOP	Stop the program

OUTPUT:

INPUT		OUTPUT	
Register	**Data**	**Address**	**Data**
A-register		**1100**	
B-register			

SUBTRACTION

ALGORITHM:

1. Load the First Data in A-register.
2. Load the Second Data in B-register.
3. Subtract the two data with borrow.
4. Store the sum in memory location.
5. Stop the program.

PROGRAM:

ADDRESS	OPCODE	LABEL	PROGRAM	COMMENTS
4110	74,05		MOV A,#data	Load data 1 in accumulator.
4102	75,F0,04		MOV B,#data	Load data 2 in B-register
4105	95,F0		SUBB A,B	Subtract the contents of B-reg from accumulator with borrow.
4107	90 11 00		MOV DPTR,#1100_H	Initialize DPTR with address 1100_H
410A	F0		MOVX @ DPTR,A	Store the result in 1100_H
410B	80, FE	STOP:	SJMP STOP	Stop the program

OUTPUT:

INPUT		OUTPUT	
Register	**Data**	**Address**	**Data**
A-register		**1100**	
B-register			

MULTIPLICATION

ALGORITHM:

1. Get the multiplier in the accumulator.
2. Get the multiplicand in the B register.
3. Multiply A with B.
4. Store the product in memory location.
5. Stop the program.

PROGRAM:

ADDRESS	OPCODE	LABEL	PROGRAM	COMMENTS
4110	74,05		MOV A,#data	Load data 1 in accumulator.
4102	75,F0,05		MOV B,#data	Load data 2 in B-register
4105	A4		MUL AB	A*B, Higher byte of result in B and lower byte of result in A.
4106	90,11,00		MOV DPTR,#1100_H	Initialize DPTR with address 1100_H
4109	F0		MOVX @ DPTR,A	Store the LSB in 1100_H
410A	A3		INC DPTR	Increment Data pointer
410B	E5,F0		MOV A,B	Copy the content of B-reg to A-register.
410D	F0		MOVX @ DPTR,A	Store the MSB in 1101_H
410E	80, FE	STOP:	SJMP STOP	Stop the program

OUTPUT:

INPUT		OUTPUT	
REGISTER	**DATA**	**ADDRESS**	**DATA**
A		**1100**	
B			

DIVISION

ALGORITHM:

1. Get the Dividend in the accumulator.
2. Get the Divisor in the B register.
3. Divide A by B.
4. Store the Quotient and Remainder in memory.
5. Stop the program.

PROGRAM:

ADDRESS	OPCODE	LABEL	PROGRAM	COMMENTS
4110	74,data1		MOV A,#data 1	Load data 1 in accumulator.
4102	75,data2		MOV B,#data 2	Load data 2 in B-register
4104	84		DIV AB	Divide. Remainder in A and quotient in B
4105	90,11,00		MOV DPTR,#1100_H	Initialize DPTR with address 1100_H
4108	F0		MOVX @ DPTR,A	Store the Remainder in 1100_H
4109	A3		INC DPTR	Increment Data pointer
410A	E5,F0		MOV A,B	Copy the content of B-reg to A-register.
410C	F0		MOVX @ DPTR,A	Store the quotient in 1101_H
410D	80, FE	STOP:	SJMP STOP	Stop the program

OUTPUT:

INPUT		OUTPUT	
REGISTER	**DATA**	**ADDRESS**	**DATA**
A-Register		**1100**	**(quotient)**
B-Register		**1101**	**(remainder)**

RESULT:

Thus, an assembly language program for Division of two 8-bit numbers and 16-bit numbers using 8051 was performed and its output was verified.