SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM NAGAR, KATTANKULATHUR – 603 203.

DEPARTMENT OF MEDICAL ELECTRONICS

LABORATORY MANUAL

190661 MICROPROCESSOR AND MICROCONTROLLER LABORATORY

III-YEAR VI-SEM

ACADEMIC YEAR: 2024-2025(EVEN SEMESTER)

Prepared by

Mr. M.SELVARAJ,

Assistant Professor / MDE

SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur -603 203

DEPARTMENT OF MEDICAL ELECTRONICS

VISION OF THE INSTITUTE

"Educate to excel in social transformation"

To accomplish and maintain international eminence and become a model institution for higher learning through dedicated development of minds, advancement of knowledge and professional application of skills to meet the global demands.

MISSION OF THE INSTITUTE

- To contribute to the development of human resources in the form of professional engineers and managers of international excellence and competence with high motivation and dynamism, who besides serving as ideal citizen of our country will contribute substantially to the economic development and advancement in their chosen areas of specialization.
- To build the institution with international repute in education in several areas at several levels with specific emphasis to promote higher education and research through strong institute-industry interaction and consultancy.

VISION OF THE DEPARTMENT

To excel in the field of Medical Electronics and to develop highly competent technocrats with global intellectual qualities.

MISSION OF THE DEPARTMENT

- To educate the students with the state of art technologies to compete internationally, able to produce creative solutions to the society`s needs, conscious to the universal moral values, adherent to the professional ethical code
- To encourage the students for professional and software development career
- To equip the students with strong foundations to enable them for continuing education and research.

PROGRAMME OUTCOMES (POs)

PO1: **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: **The Engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAMME SPECIFIC OUTCOMES (PSOs) of ECE DEPARTMENT

PSO1: Ability to apply the acquired knowledge of basic skills, mathematical foundations, and principles of electronics, modeling and design of electronics based systems in solving engineering Problems.

PSO2: Ability to understand and analyze the interdisciplinary problems for developing innovative sustained solutions with environmental concerns.

PSO3: Ability to update knowledge continuously in the tools like MATLAB, NS2, XILINIX and technologies like VLSI, Embedded, Wireless Communications to meet the industry requirements.

PSO4: Ability to manage effectively as part of a team with professional behavior and ethics.

190661 –MICROPROCESSOR AND MICROCONTROLLER LAB

LIST OF EXPERIMENTS

8086 Programs using kits and MASM

1. Basic arithmetic and Logical operations
2. Move a data block without overlap
3. Code conversion, decimal arithmetic and Matrix operations.
4. Floating point operations, string manipulations, sorting and searching
5. Password checking, Print RAM size and system date
6. Counters and Time Delay

Peripherals and Interfacing Experiments

7. Traffic light control
8. Stepper motor control
9. Digital clock
10. Key board and Display
11. Printer status
12. Serial interface and Parallel interface
13. A/D and D/A interface and Waveform Generation

8051 Experiments using kits and MASM

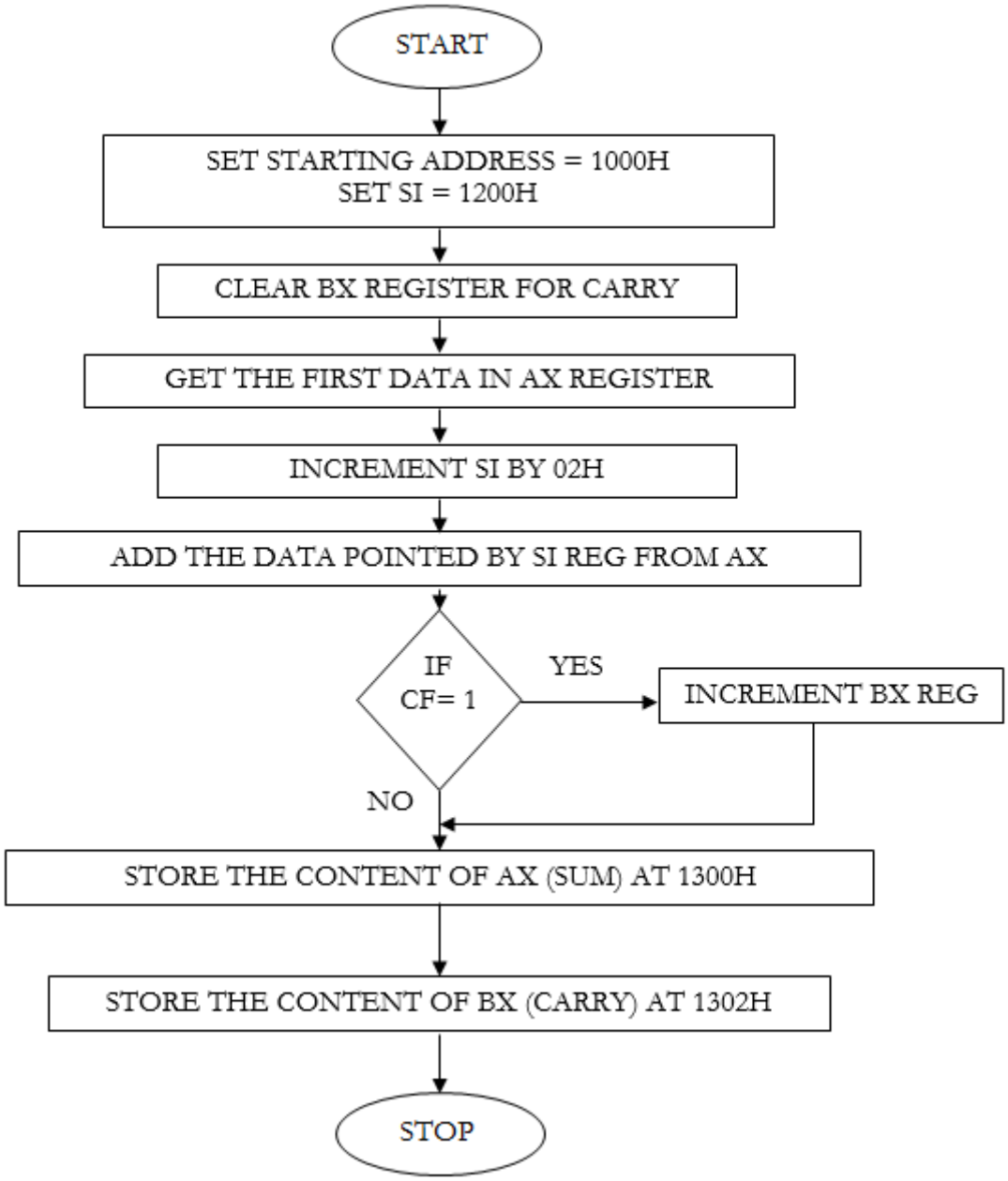
14. Basic arithmetic and Logical operations
15. Square and Cube program, Find 2‟s complement of a number
16. Unpacked BCD to ASCII

BEYOND THE SYLLABUS

17. Square wave generation using 8051

CONTENTS

Flow Chart for Addition of Two Numbers:



Ex. No. 1
Date:

PROGRAMS FOR BASIC ARITHMETIC AND LOGICAL OPERATIONS

Objective:

To write an Assembly Language Program (ALP) to perform basic Arithmetic and Logical Operations

(a) Addition of two numbers
(b) Subtraction of two numbers
(c) Multiplication of two numbers
(d) Division of two numbers
(e) Logical operation

(A) ADDITION OF TWO 16 BIT NUMBERS

Description:

To perform addition in 8086, one of the data should be stored in a register and another data can be stored in register / memory. After addition the sum will be available in the destination register / memory. The sum of two 16-bit data can be either 16 bits (sum only) or 17 bits (sum and carry). The destination register / memory can accommodate only the sum and if there is a carry the 8086 will indicate by setting carry flag. Hence one of the register is used for the account of carry.

Algorithm:

1. Start the program.
2. Set the origin as 1000H.
3. Store the 1st data in AX register.
4. Clear BX register pair for carry.
5. Set SI to 1202H to point the second data.
6. Add the content in AX with data pointed by SI register.
7. If carry occurs, increment BX register by one.
8. Move the content of AX to 1300H.
9. Move the content of BX to 1302H.
10. End of segment.
11. Stop the program

PROGRAM

Label	Program	Comments
	ORG 1000H	Set starting address as 1000H.
	MOV BX, 0000H	Initialize BX to 0000H
	MOV SI, 1200H	Move immediate data to SI
	MOV AX, [SI]	Move content of SI to AX
	ADD SI, 02H	ADD SI with immediate data.
	ADD AX, [SI]	Add content of SI with AX register
	JNC Next	Jump if no carry to loop
	INC BX	Increment BX register
Next:	MOV DI, 1300H	Move immediate data to DI.
	MOV [DI], AX	Move AX to DI.
	ADD DI, 02H	ADD DI with immediate data
	MOV [DI], BX	Move BX to DI
	HLT	

Example 1:

With Carry

Input:

1200: 46
1201: B6 [Addend]
1202: D3
1203: 98 [Augend]

Output:

1300: 19
1301: 4F [Sum]
1302: 01
1303: 00 [Carry]

Example 2:

Without Carry

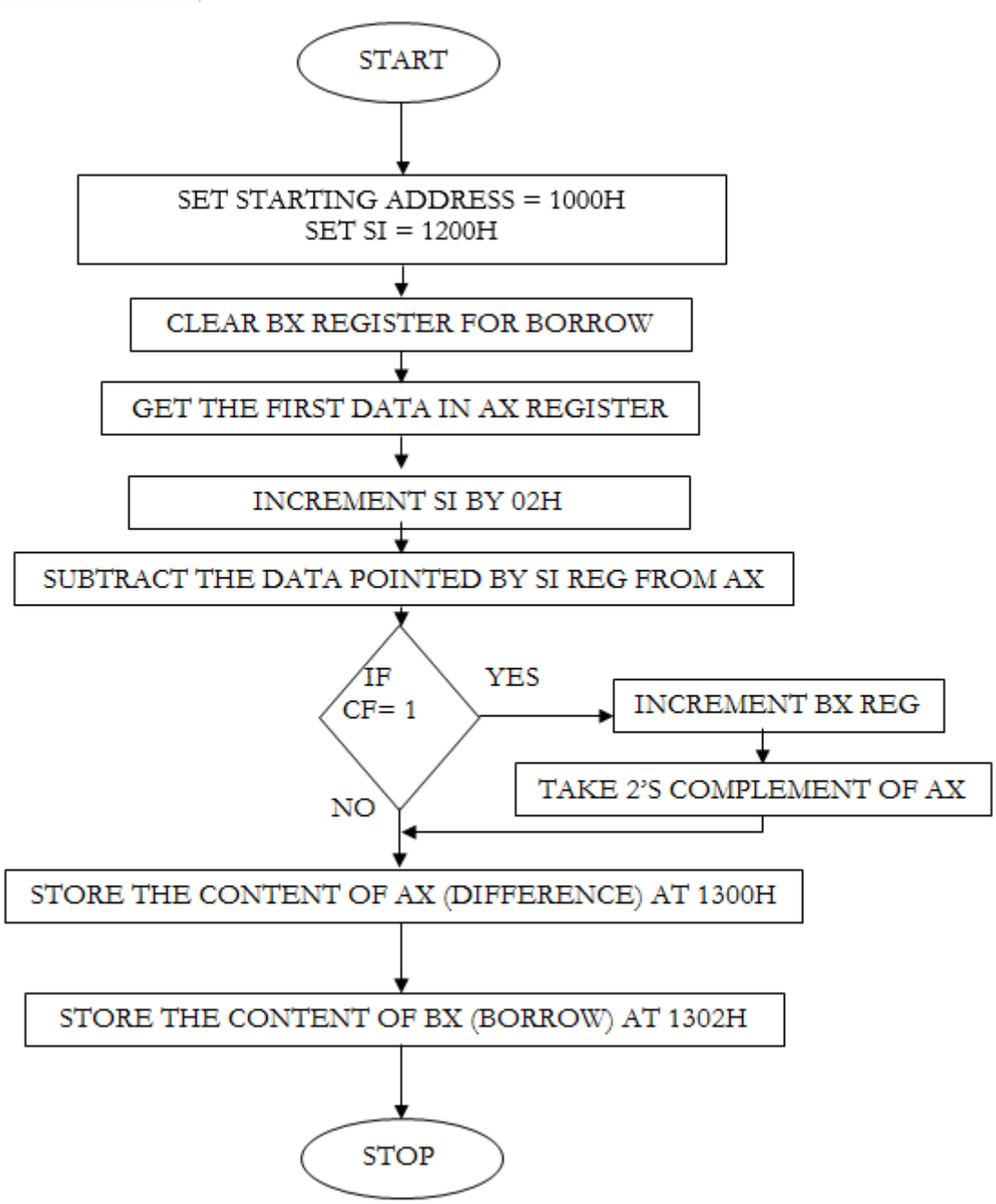
Input:

1200: 34
1201: 44 [Addend]
1202: 24
1203: 24 [Augend]

Output:

1300: 58
1301: 68 [Sum]
1302: 00
1303: 00 [Carry]

Flow Chart of Subtraction of Two Numbers:



(B) SUBTRACTION OF TWO 16 BIT NUMBERS

Description:

To perform subtraction in 8086 one of the data should be stored in register and another data should be stored in register or memory. After subtraction the result will be available in destination register/memory. The 8086 will perform 2's complement subtraction and then complement the carry. Therefore if the result is negative then carry flag is set and the destination register/memory will have 2's complement of the result. Hence one of the registers is used to account for sign of the result. To get the magnitude of the result again take 2's complement of the result.

Algorithm:

1. Start the program.
2. Set the starting address as 1000H.
3. Set the SI register to 1200H address.
4. Move the 16 bit data to AX register pair.
5. Increment the SI register to 1202.
6. Get the second data.
7. Move this second value to BX register.
8. Subtract the content pointed by SI from AX and store result in AX.
9. If carry occurs go to step 13.
10. Increment BX register, then perform inversion operation to AX register.
11. Increment AX register.
12. Move the resultant to DI register.
13. Display the output.
14. End of segment.
15. Stop the program.

PROGRAM

Label	Program	Comments
	ORG 1000H	Set starting address as 1000H
	MOV BX, 0000H	Move immediate data to BX register.
	MOV SI, 1200H	Move immediate data to SI
	MOV AX, [SI]	Move contents of SI to AX
	ADD SI, 02H	Increment SI by 02H
	SUB AX, [SI]	Move contents of SI to AX
	JNC Next	Jump if no carry loop
	INC BX	Increment BX
	NOT AX	Perform NOT operation of AX
	INC AX	Increment AX register
Next:	MOV DI, 1300H	Move immediate data to DI.
	MOV [DI], AX	Move AX to DI.
	ADD DI, 02H	Increment DI by 02H
	MOV [DI], BX	Move BX to DI
	HLT	

Example 1:

With Borrow

Input:

1200: 03
1201: 00 (minuend)
1202: 05
1203: 00 (subtrahend)

Output:

1300: 02
1301: 00 (Difference)
1302: 01
1303: 00 (Borrow)

Example 2:

Without Borrow

Input:

1200: 31
1201: 82 (minuend)
1202: 06
1203: 34 (subtrahend)

Output:

1300: 2B
1301: 4E (Difference)
1302: 00
1303: 00 (Borrow)

Observation:

Input:

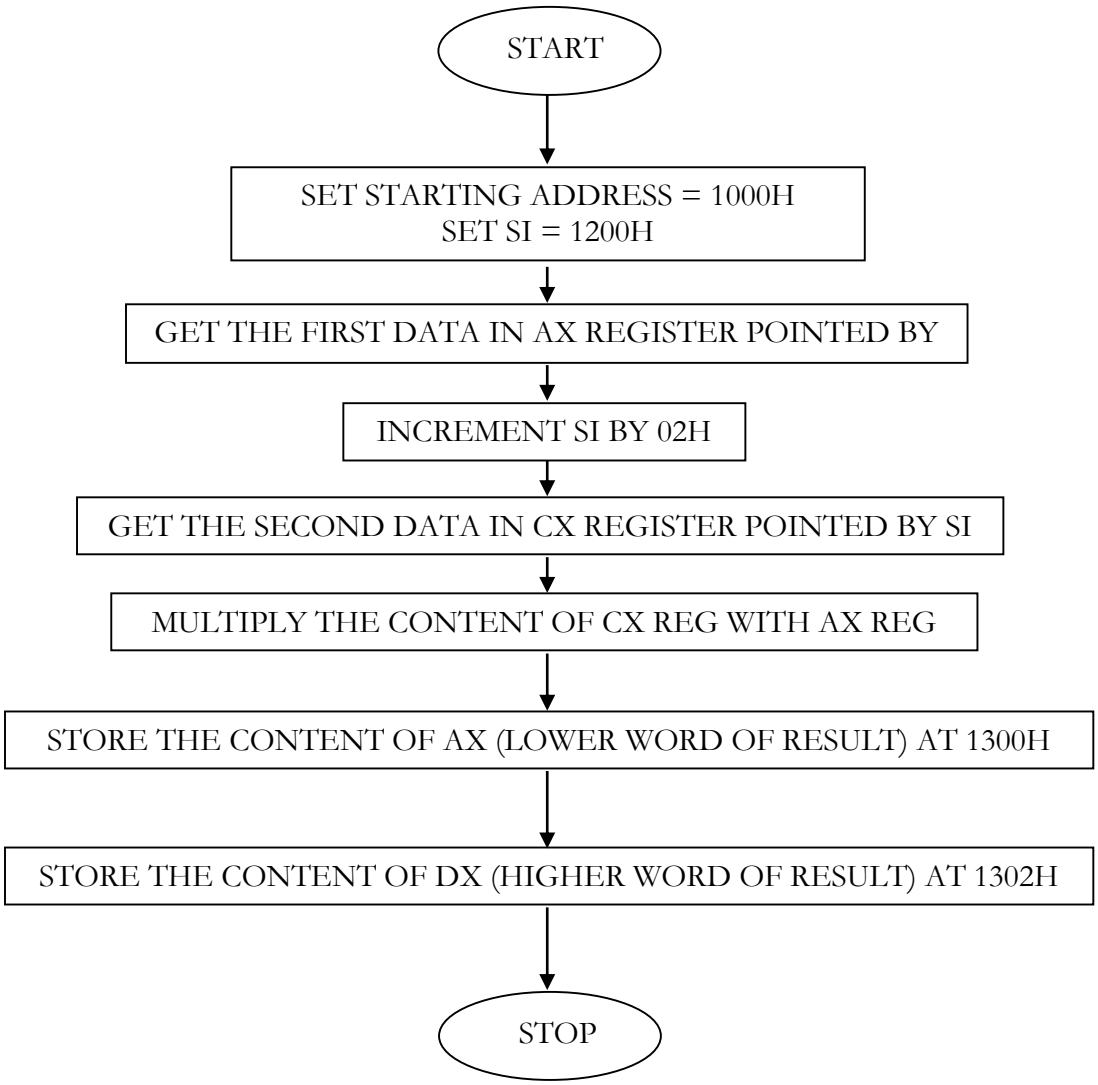
1200:
1201: (minuend)
1202:
1203: (subtrahend)

Output:

1300:
1301: (Difference)
1302:
1303: (Borrow)

Manual Calculation:

Flow Chart for Multiplication of Two Numbers:



(C) MULTIPLICATION OF TWO 16 BIT NUMBERS

Description:

To perform multiplication in 8086 processor one of the data should be stored in AX register and another data can be stored in register/memory. After multiplication the product will be in AX [lower word] and DX register [Higher word].

Algorithm:

1. Start the program
2. Set the starting address as 1000H
3. Set the SI register to point the location 1200H.
4. Set the DI register to point the location 1300H.
5. Move the 16 bit data pointed by SI to AX register
6. Move this data to BX register
7. Increment SI register to 1202 and get the second data in AX register
8. Multiply the data in AX with BX register
9. Store the data in DX [higher word] and AX [lower word] addressed by DI register.
10. Display the result
11. End of segment
12. Stop the program

PROGRAM

Label	Program	Comments
	ORG 1000H	Set starting address as 1000H.
	MOV SI, 1200H	Move immediate data to SI
	MOV AX,[SI]	Move contents of SI to AX
	ADD SI,02H	Increment SI value to 02H
	MOV BX, [SI]	Move contents of SI to BX
	MUL BX	Multiply BX with AX
	MOV DI, 1300H	Move immediate data to DI
	MOV [DI], AX	Move AX to DI register
	MOV DI, 1302H	Move immediate data to DI
	MOV [DI], DX	Move DX to DI register
	HLT	

Example:

Input:

1200: 02
1201: 06 (Multiplicand)
1202: 02
1203: 06 (Multiplier)

Output:

1300: 04
1301: 18 (Lower word of the Product)
1302: 24
1303: 00 (Higher word of the Product)

Observation:

Input:

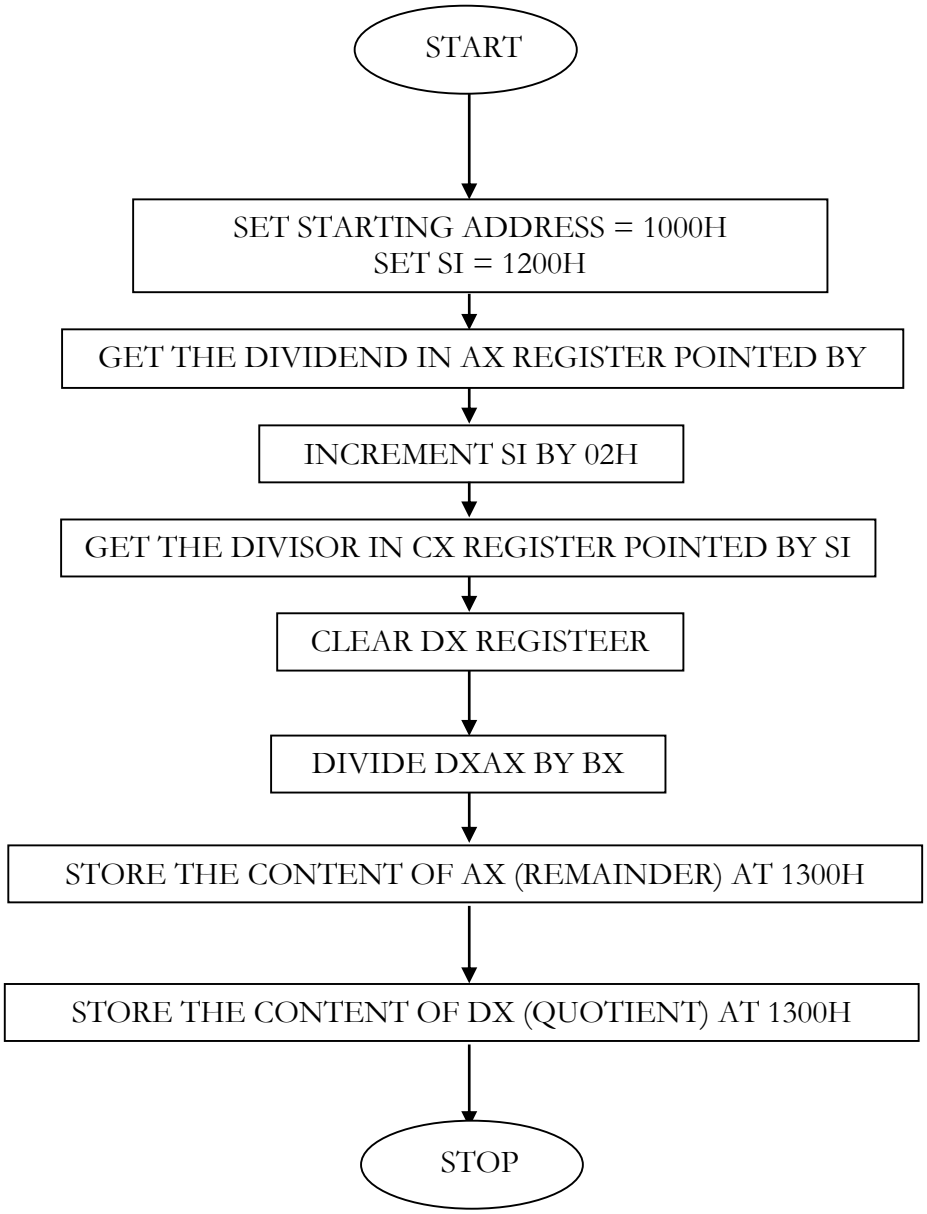
1200:
1201: (Multiplicand)
1202:
1203: (Multiplier)

Output:

1300:
1301: (Lower word of the Product)
1302:
1303: (Higher word of the Product)

Manual Calculation:

Flow Chart for Division of Two Numbers:



(D) DIVISION OF TWO NUMBERS

Description:

To perform division in 8086 processor, the 16 bit dividend should be stored in AX and DX register (The lower word in AX and Upper word in DX). The 16 bit divisor can be stored in register / memory. After division the quotient will be in AX register and the remainder will be in DX register.

Algorithm:

1. Start the program
2. Set the origin as 1000H
3. Set SI as 1200H.
4. Clear DX register for 16 bit dividend. For 16 bit dividend higher word is zero.
5. Load the lower word of dividend in AX register
6. Increment SI by 02H. Load the divisor in BX register.
7. Perform division of data in DX AX by BX
8. Set DI as 1300H
9. Store the quotient in AX register at the location pointed by DI register.
10. Set DI as 1302H
11. Store the remainder in DX register at the location pointed by DI register.
12. Display the result, End of Segment
13. Stop the program

Label	Program	Comments
	ORG 1000H	Set starting address as 1000H.
	MOV SI, 1200H	Move immediate data to SI
	MOV AX,[SI]	Move contents of SI to AX
	ADD SI,02H	Add 02H to SI
	MOV BX, [SI]	Move contents of SI to BX
	MOV DX, 0000H	Initialize DX to 0000H
	DIV BX	Divide DXAX by BX
	MOV DI, 1300H	Move immediate data to DI
	MOV [DI], AX	Store the quotient
	MOV DI, 1302H	Move immediate data to DI
	MOV [DI], DX	Store the remainder
	HLT	

Example:

Input:

1200: 06
1201: 06 (Dividend)
1202: 03
1203: 03 (Divisor)

Output:

1300: 02
1301: 00 (Quotient)
1302: 00
1303: 00 (Remainder)

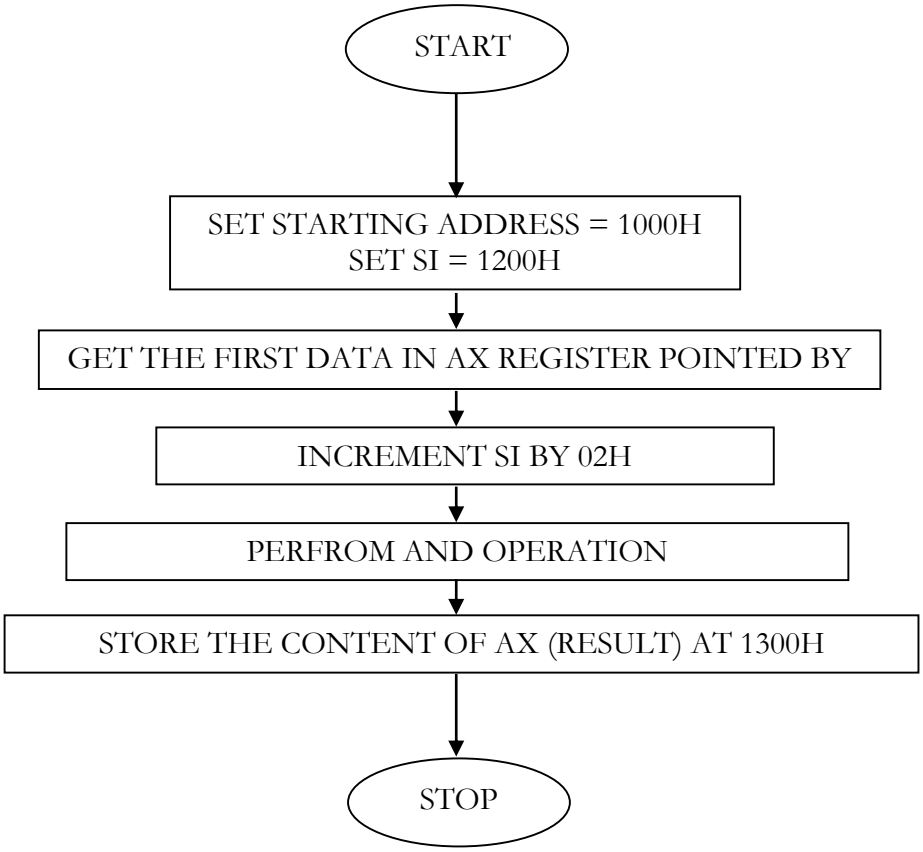
Observation:

Input:
1200:
1201: (Dividend)
1202:
1203: (Divisor)

Output:
1300:
1301: (Quotient)
1302:
1303: (Remainder)

Manual Calculation:

FLOWCHART



Label	Program	Comments
	ORG 1000H	Set starting address as 1000H.
	MOV SI,1200H	Initialize SI
	MOV AX,[SI]	Get the first data in AX – reg
	ADD SI,02H	Increment SI to point next data
	AND AX,[SI]	Perform AND operation of two data
	MOV DI,1300H	
	MOV [DI],AX	Store the result in memory
	HLT	

(E) LOGICAL OPERATIONS OF 16 BIT NUMBERS

Description:

The two values from memory are logically AND then the result is stored in memory.

Algorithm:

1. Start the program and Set the origin as 1000H
2. Set SI as 1200H.
3. Get the first data in AX – reg
4. Increment SI to point next data
5. Perform AND operation of the data
6. Store the result in memory
7. Stop the program

Example:

Input

1200: 01
1201: 01
1202: 00
1203: 00

Output

1300: 00
1301: 00

Observation:

Input

1200:
1201:
1202:
1203:

Output

1300 :
1301 :

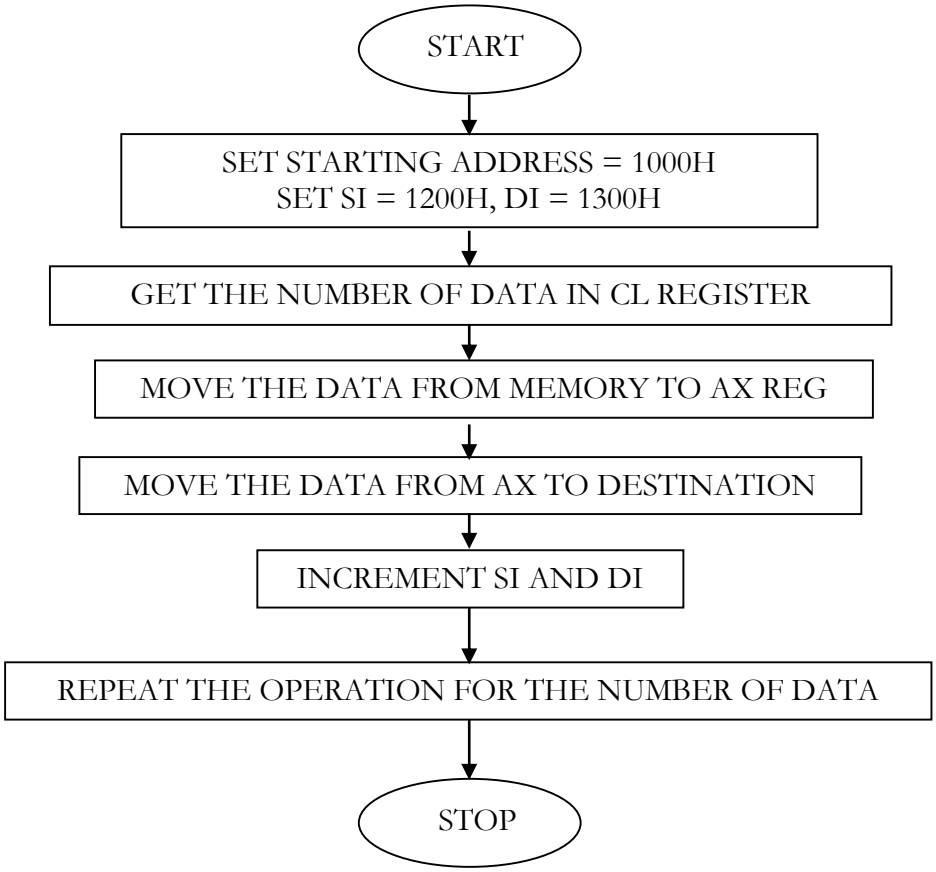
REVIEW QUESTIONS:

1. Write the size of the data bus of 8086.
2. Write the size of the address bus of 8086.
3. What is meant by physical addressing in 8086?
4. What is meant by an Opcode?
5. What is meant by an Operand?
6. What is meant by a Mnemonics?
7. What are the other possibilities of writing ADD, SUB and MUL instructions in other addressing modes?
8. What is the difference between microprocessor and microcontroller?
9. What is meant by LATCH?
10. What is the difference between primary & secondary storage device?
11. What is the difference between static and dynamic RAM?
12. What is an interrupt?
13. Differentiate between RAM and ROM?
14. Define – Compiler
15. Define – Flag
16. Define – Stack
17. How clock signal is generated in 8086 microprocessor?
18. State the functions of queue status line QS0 and QS1 in 8086 microprocessor.
19. What is the purpose of BIU& EU?
20. List out the two examples of assembler directives.

Result:

Thus the program for arithmetic and logic operation was written and executed.

Flow Chart to Move a Block of Data without Overlap:



Ex. No. 2
Date:

MOVE A DATA BLOCK WITHOUT OVERLAP

Objective:

To write an 8086 ALP to move a block of data from source to destination without overlap

Description:

The block of data to be moved from one location (source) to another location (destination) in memory. The source and destination of memory is pointed by SI and DI respectively. The size of the block is stored in CL register. The data from source are moved to register and then back to destination location. The steps are repeated till the value of CL register is Zero.

Algorithm:

1. Start the program.
2. Set the starting address as 1000H.
3. Set the SI register to 1200H address.
4. Set the DI register to 1300H address.
5. Set the CL register to hold the number of data to be moved.
6. Move the 16 bit data from memory pointed by SI to AX register pair.
7. Move the 16 bit from AX register to memory pointed by DI.
8. Increment the SI register by 02H.
9. Increment the DI register by 02H.
10. Repeat steps 6 to 9 till the cl value is zero
11. Stop the program.

Label	Program	Comments
	ORG 1000H	Set starting address as 1000H.
	MOV SI, 1200H	Initialise SI to 1200
	MOV DI,1300H	Initialise DI to 1300
	MOV CL,05H	Initialise CL for number of data
Next:	MOV AX,[SI]	
	MOV [DI],AX	
	ADD SI, 02H	
	ADD DI, 02H	
	LOOP Next	
	HLT	

Example:

Input:

1200: 05
1201: 03
1202: 02
1203: 01
1204: 00

Output:

1300: 05
1301: 03
1302: 02
1303: 01
1304: 00

Observation:

Input:

1200:
1201:
1202:
1203:
1204:

Output:

1300:
1301:
1302:
1303:
1304:

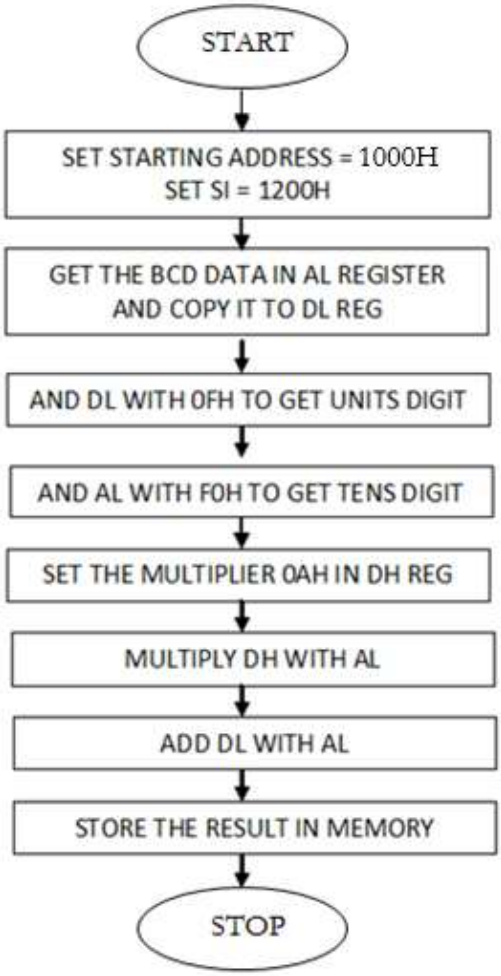
REVIEW QUESTIONS:

1. List out the Flag manipulation instruction.
2. Define – Variables
3. Define – Segment Override Prefix
4. How is the memory segment accessed by 8086 microprocessor identified?
5. List out the advantages of using Direct Memory Access (DMA).
6. What is BIOS function call in 8086? (May 2012)
7. List out the difference between procedures and Macros.
8. What is meant by Maskable interrupts& Non-Maskable interrupts?
9. What is the Maximum clock frequency in 8086?
10. Which Stack is used in 8086?
11. Define pipeline (Dec 2011)
12. How many address lines are available in 8086? What is the maximum address possible?
13. What is an assembler(May 2012)
14. What is the purpose of LEA instruction in 8086? (May 2012)
15. Give the function of index and pointers in 8086
16. What are the different instruction set of 8086?
17. Give the various addressing modes in 8086
18. Give the differences between JUMP and LOOP instruction
19. Give the physical address formation of any two addressing mode
20. Give the use of "ASSUME" in 8086 programming

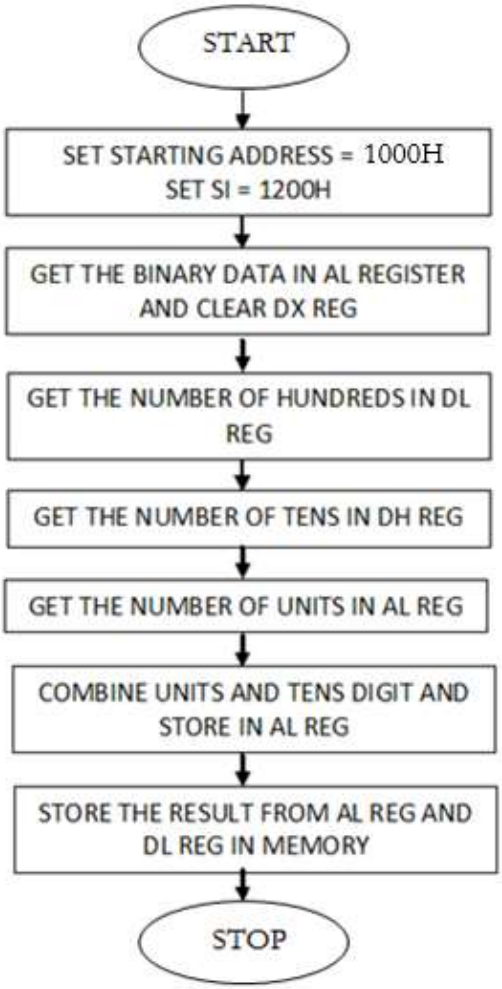
Result:

Thus the program for moving a block of data without overlap was written and executed.

Flow Chart for Binary to BCD Conversion



Flow Chart for BCD to Binary Conversion



Ex. No. 3
Date:

CODE CONVERSION, DECIMAL ARITHMETIC & MATRIX OPERATIONS

Objective:

To write an Assembly Language Program (ALP) to perform the following operations

(a) Code Conversion
    BCD to Binary
    Binary to BCD
(b) Decimal Arithmetic
    BCD Addition
    BCD Subtraction
(c) Matrix Operations
    Matrix Addition
    Matrix Multiplication

(A) CODE CONVERSION – BCD to Binary

Description:

The 2 –digit BCD data will have units digits and tens digits. When the tens digit is multiplied by 0A H and the product is added to units digit, the result will be in binary, because the microprocessor will perform binary arithmetic. In order to separate the units and tens digit, masking technique is used.

Algorithm:

1. Start the program.
2. Set the origin as 1000H.
3. Get the BCD data in AL register
4. Copy the BCD data in DL register
5. Logically AND DL with 0F to mask upper nibble and get the units digit in DL
6. Logically AND AL with F0 to mask lower nibble and get the tens digit in AL
7. Rotate the content of AL register 4 times in order to change upper nibble as lower nibble.
8. Set the multiplier 0A H in DH register.
9. Multiply AL with DH register, the product will be in AL register.
10. Add the units digit in DL register to the product in AL register
11. Save the binary digit (AL) in memory
12. Stop the program.

Label	Program	Comments
	ORG 1000H	Set starting address as 1000H.
	MOV SI, 1200H	Initialize SI
	MOV AL,[SI]	Move the BCD data in AL
	MOV DL,AL	Copy the BCD data in DL
	AND DL,0F	AND DL with 0F
	AND AL,0F0	AND AL with F0
	MOV CL,04	
	ROR AL,CL	Rotate AL for 4 – times
	MOV DH,0A	Move 0A to DH
	MUL DH	Multiply DH with AL
	ADD AL,DL	Add AL with DL
	MOV DI,1201H	
	MOV [DI],AL	Store the result in memory
	HLT	

Example:

Input:

1200: 85 [BCD data]

Output:

1201: 55

Observation:

Input:

1200: [BCD data]

Output:

1201:

Manual Calculation:

Label	Program	Comments
	ORG 1000H	Set starting address as 1000H.
	MOV SI, 1200H	Initialize SI
	MOV AL,[SI]	Move the binary data in AL
	MOV DX,0000H	Clear the counter
HUND:	CMP AL, 64H	To count number of hundreds
	JC TEN	
	SUB AL,64H	
	INC DL	
	JMP HUND	
TEN:	CMP AL,0AH	To count number of tens
	JC UNIT	
	SUB AL,0AH	
	INC DH	
	JMP TEN	
UNIT:	MOV CL,04	
	ROL DH,CL	
	ADD AL,DH	Add tens and units
	MOV DI,1201H	
	MOV [DI],AL	Store in memory
	INC DI	
	MOV [DI],DL	
	HLT	

CODE CONVERSION – BINARY TO BCD

Description:

The maximum value of 8 bit binary is FFH. The BCD equivalent is 256. Hence when an 8 – bit binary is converted into BCD, the BCD data will have hundreds, tens and units digit. So two counters are used to count hundreds and tens. The tens and units digit are added and stored in a memory location and the hundreds digit is stored in the next location.

Algorithm:

1. Start the program.
2. Set the origin as 1000H.
3. Get the binary data in AL register
4. Clear DX register for storing Hundreds and tens
5. Compare AL with 64H (100 in decimal)
6. Check carry flag. If CF = 1, then go to step 10, else go to next step
7. Subtract 64H from AL register
8. Increment Hundreds register (DL)
9. Go to Step 5
10. Compare AL with 0AH (10 in decimal)
11. Check carry flag. If CF = 1, then go to step 15, else go to next step
12. Subtract 0AH from AL register
13. Increment Tens register (DH)
14. Go to step 10
15. Rotate the content of DH four times
16. Add DH to AL to combine tens and Units digit
17. Save AL and DL in memory.
18. Stop the program

Example:

Input:

1200: 55 [Binary data]

Output:

1201:85

Observation:

Input:

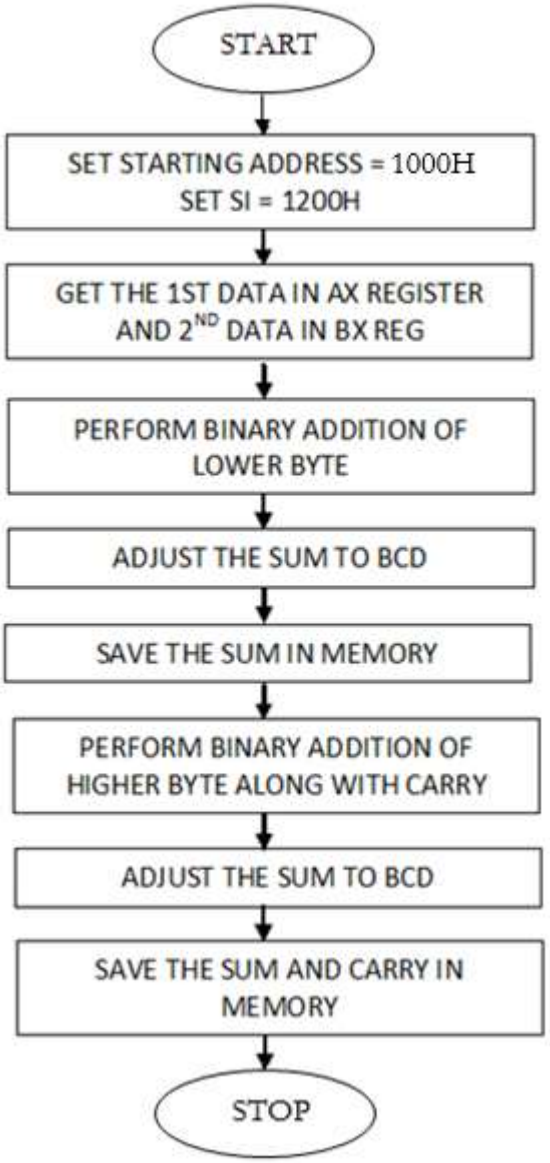
1200: [Binary data]

Output:

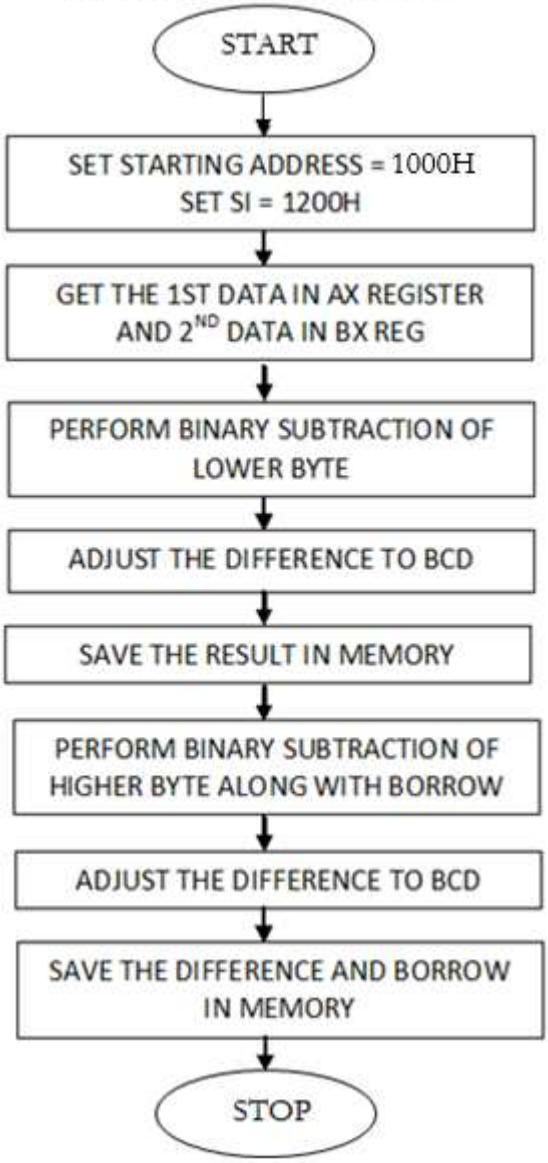
1201:

Manual Calculation:

Flow Chart for BCD Addition



Flow Chart for BCD Subtraction



DECIMAL ARITHMETIC – BCD ADDITION

Description:

The binary addition is performed and then the sum is corrected to get the result in BCD. If the sum of the lower nibble exceeds 9 or if there is auxiliary carry then 6 is added to the lower nibble. if the sum of the upper nibble exceeds 9 or if there is a carry then 6 is added to upper nibble. These conversions are taken care by DAA instruction.

Algorithm:

1. Start the program.
2. Set the origin as 1000H.
3. Initialise SI to 1200H
4. Clear the CL register for Carry
5. Load the first data in AX reg and second data in BX reg.
6. Perform Binary addition of lower byte
7. Adjust the sum of lower bytes to BCD
8. Save the sum in memory.
9. Perform Binary addition of Higher byte along with carry from lower byte.
10. Adjust the sum of higher bytes to BCD
11. Save the sum in memory
12. Save the carry in memory
13. Stop the program.

Label	Program	Comments
	ORG 1000H	Set starting address as 1000H.
	MOV SI, 1200H	Initialize SI
	MOV CL,00H	Clear CL register for carry
	MOV AX,[SI]	Get the 1st number in AX reg
	MOV BX,[SI+2]	Get the 2nd number in BX reg
	ADD AL,BL	Add the lower nibble
	DAA	Decimal adjust for BCD
	MOV DL,AL	
	MOV AL,AH	
	ADC AL,BH	Add the higher nibble with carry
	DAA	Decimal adjust for BCD
	MOV DH,AL	
	JNC AHEAD	Check for Carry
	INC CL	
AHEAD:	MOV DI,1204H	
	MOV [DI],DX	Store the result in memory
	MOV [DI+2],CL	
	HLT	

Example:

Input:

1200: 01 [1st data – BCD]
1201: 04
1202: 08 [2nd data – BCD]
1203: 02

Output:

1204: 09
1205: 06

Observation:

Input:

1200: [1st data – BCD]
1201:
1202: [2nd data – BCD]
1203:

Output:

1204:
1205:

Manual Calculation:

Label	Program	Comments
	ORG 1000H	Set starting address as 1000H.
	MOV SI, 1200H	Initialize SI
	MOV CL,00H	Clear CL register for borrow
	MOV AX,[SI]	Get the 1st number in AX reg
	MOV BX,[SI+2]	Get the 2nd number in BX reg
	SUB AL,BL	Subtract the lower nibble
	DAS	Decimal adjust for BCD
	MOV DL,AL	
	MOV AL,AH	
	SBB AL,BH	Subtract the higher nibble with Borrow
	DAS	Decimal adjust for BCD
	MOV DH,AL	
	JNC AHEAD	Check for Borrow
	INC CL	
AHEAD:	MOV DI,1204H	
	MOV [DI],DX	Store the result in memory
	MOV [DI+2],CL	
	HLT	

Observation:

Input:

1200: 18 [1st data – BCD]
1201: 04
1202: 09 [2nd data – BCD]
1203: 02

Output:

1204: 09
1205: 02

Observation:

Input:

1200: [1st data – BCD]
1201:
1202: [2nd data – BCD]
1203:

Output:

1204:
1205:

Manual Calculation:

DECIMAL ARITHMETIC – BCD SUBTRACTION

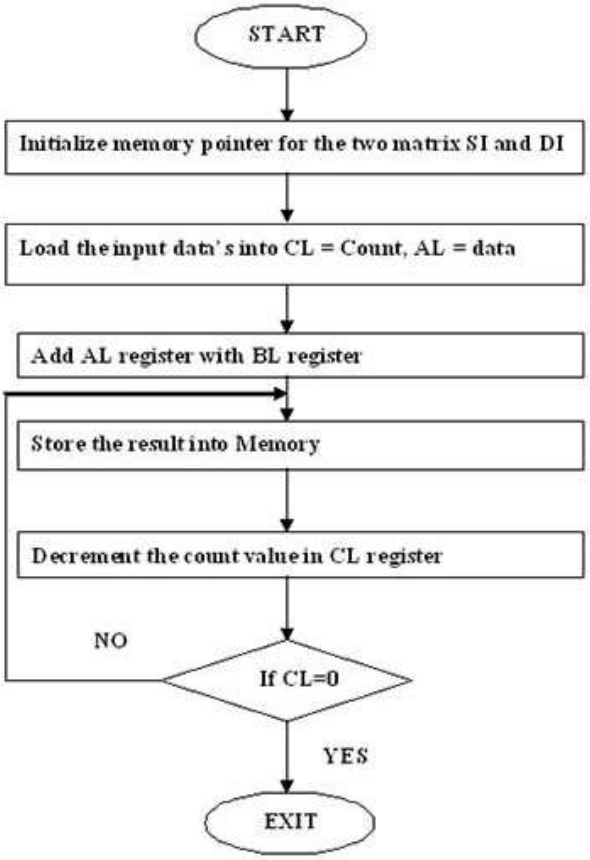
Description:

The binary subtraction is performed and then the difference is corrected to get the result in BCD. If the difference of the lower nibble exceeds 9 or if there is auxiliary carry then 6 is subtracted from the lower nibble. if the difference of the upper nibble exceeds 9 or if there is a carry then 6 is subtracted from upper nibble. This conversion is taken care by DAS instruction.

Algorithm:

1. Start the program.
2. Set the origin as 1000H.
3. Initialise SI to 1200H
4. Clear the CL register for borrow
5. Load the first data in AX reg and second data in BX reg.
6. Perform Binary subtraction of lower byte
7. Adjust the difference of lower bytes to BCD
8. Save the result in memory.
9. Perform Binary subtraction of Higher byte along with borrow from lower byte.
10. Adjust the difference of higher bytes to BCD
11. Save the difference in memory
12. Save the borrow in memory
13. Stop the program.

Flow Chart for Matrix Addition:



MATRIX ADDITION

Description:

The matrix addition is performed by loading the size of the matrix in CL reg and then adding the individual elements of the matrix.

Algorithm:

1. Start the program.
2. Set the origin as 1000H.
3. Initialize the pointer to memory for data and result.
4. Load CL with count.
5. Add two matrices by each element.
6. Process continues until CL is 0.
7. Store the result into Memory.
8. Stop the program.

LABEL	PROGRAM	COMMENTS
	MOV CL, 09	Initialize 09 into CL register
	MOV SI, 2000	Load 2000 into SI for 1st matrix
	MOV DI, 3000	Load 3000 into DI for 2nd matrix
NEXT	MOV AL, [SI]	Load AL with data of first matrix
	MOV BL, [DI]	Load BL with data of second matrix
	ADD AL, BL	Add two data of AL and BL
	MOV [DI], AL	Store AL with data into DI
	INC DI	Increment DI
	INC SI	Increment SI
	DEC CL	Decrement CL
	JNZ NEXT	Loop continues until all elements of Matrix to added
	HLT	Halt the Program

Observation:
Input:

Matrix A
2000: 00
2001: 01
2002: 02
2003: 03
2004: 04
2005: 05
2006: 06
2007: 07
2008: 08

Matrix B
3000: 09
3001: 08
3002: 07
3003: 06
3004: 05
3005: 04
3006: 03
3007: 02
3008: 01

Output
3000: 09
3001: 09
3002: 09
3003: 09
3004: 09
3005: 09
3006: 09
3007: 09
3008: 09

Observation:
Input:

Matrix A
2000:
2001:
2002:
2003:
2004:
2005:
2006:
2007:
2008:

Matrix B
3000:
3001:
3002:
3003:
3004:
3005:
3006:
3007:
3008:

Output
3000:
3001:
3002:
3003:
3004:
3005:
3006:
3007:
3008:

REVIEW QUESTIONS:

1. Write the function of the following 8085 instructions: JP, JPE, JPO, and JNZ.
2. What is the purpose of the following commands in 8086?
    a) AAD
    b) RCL
3. List out the addressing modes in 8086.
4. List out the various string instructions that are available in 8086.
5. What are the 8086 instructions used for BCD arithmetic?
6. What flags get affected after executing ADD instruction?
7. Which instruction is used to add immediate data?
8. What is BCD code? Where it is used?
9. What is ASCII code? Where it is used?
10. What is the difference between carry flag and overflow flag?
11. What are the special function register associated with interrupts?
12. List the flags affected by arithmetic instructions.
13. After executing ADDC instruction, what flags get affected?
14. How many bytes the instruction ADDC will add?
15. Name the signals used by the processor to communicate with an I/O processor
16. What is the function of IP?
17. What is the use of base pointer register?
18. Mention the index registers of 8086.
19. How many memory locations are available in 8086 microprocessor?
20. What are the flags in 8086? What are the various interrupts in 8086?

Result:

Thus the program for Matrix addition was successfully executed.

Label	Program	Comments
	MOV CH,03H	Initialize CH reg with no of rows
	MOV BX,1400H	Initialize BX reg to 1400H
	MOV SI,0200H	Initialize SI to 1200H
ROW:	MOV DI,1300H	Initialize DI to 1300
	MOV CL,03H	Initialize CL reg with no of columns
COLUMN:	MOV DL,03H	Move 03 to DL
	MOV BP,0000H	Initialize BP to 0000H
	MOV AX,0000H	Initialize AX to 0000H
	SAHF	Store AH register into flags
NEXT:	MOV AL,[SI]	Move the value pointed by SI to AL
	MUL [DI]	Multiply the value pointed by DI with AL
	ADD BP,AX	Add the result with BP reg
	INC SI	Increment SI
	ADD DI,03H	Add 03 to point the next row element
	DEC DL	Decrement DL
	JNZ NEXT	If not zero go to NEXT
	SUB DI,08H	Subtract DI with 08H
	SUB SI,03H	Subtract SI with 03H
	MOV [BX],BP	Move the result to memory pointed by BP
	ADD BX,02H	Add 02 to BX
	DEC CL	Decrement the value of CL
	JNZ COLUMN	If not zero jump to COLUMN
	ADD SI,03H	Add 03H to SI
	DEC CH	Decrement CH
	JNZ ROW	If not Zero Jump to ROW
	HLT	Halt

MATRIX MULTIPLICATION

Description:

The matrix multiplication is performed by loading the number of rows in CH reg and number of columns in CL reg and then multiplying the individual elements of the matrix.

Algorithm:

1. Initialize CH reg with no of rows
2. Initialize BX reg to 1400H
3. Initialize SI to 1200H
4. Initialize DI to 1300
5. Initialize CL reg with no of columns
6. Move 03 to DL
7. Initialize BP to 0000H
8. Initialize AX to 0000H
9. Store AH register into flags
10. Move the value pointed by SI to AL
11. Multiply the value pointed by DI with AL
12. Add the result with BP reg
13. Increment SI
14. Add 03 to point the next row element
15. Decrement DL
16. If not zero go to NEXT
17. Subtract DI with 08H
18. Subtract SI with 03H
19. Move the result to memory pointed by BP
20. Add 02 to BX
21. Decrement the value of CL
22. If not zero jump to COLUMN
23. Add 03H to SI
24. Decrement CH
25. If not Zero Jump to ROW
26. Halt

Example:
Input:

Matrix A
1200:02
1201:02
1202:02
1203:02
1204:02
1205:02
1206:02
1207:02
1208:02

Matrix B
1300:02
1301:02
1302:02
1303:02
1304:02
1305:02
1306:02
1307:02
1308:02

Output
1400:0C
1401:00
1402:0C
1403:00
1404:0C
1405:00
1406:0C
1407:00
1408:0C

Observation:
Input:

Matrix A
1200:
1201:
1202:
1203:
1204:
1205:
1206:
1207:
1208:

Matrix B
1300:
1301:
1302:
1303:
1304:
1305:
1306:
1307:
1308:

Output
1400:
1401:
1402:
1403:
1404:
1405:
1406:
1407:
1408:

REVIEW QUESTIONS:

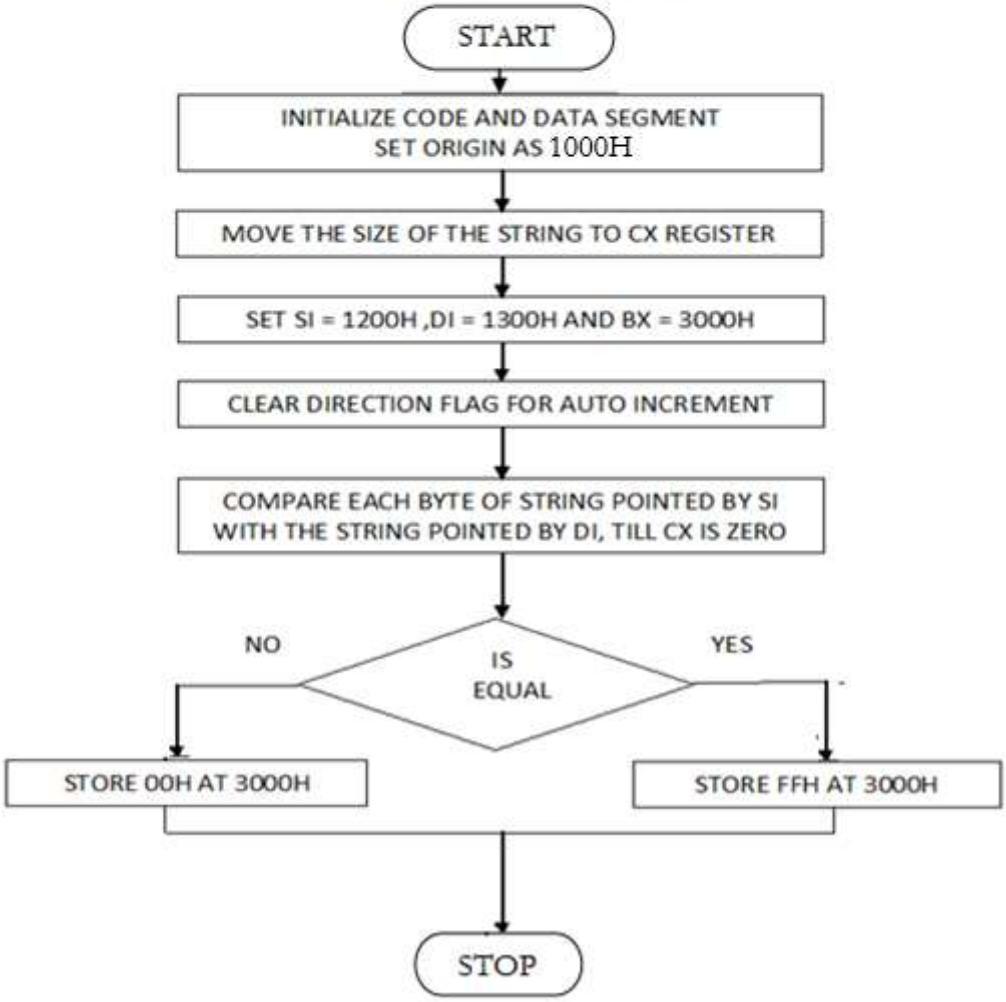
1. Write an ALP for 8086 to multiply two 16 bit unsigned numbers.
2. What is an accumulator?
3. List out the segment register available in 8086
4. List out any four program control instructions that are available in 8086
5. What is program counter?
6. Give any four logical instructions in 8086
7. How many memory locations are available in 8086 microprocessor?
8. What are the general purposes registers in 8086?
9. What are the functional units in 8086?
10. How much memory location allotted for the particular segments registers in 8086?
11. When the 8086 processor is in minimum mode and maximum mode?
12. Define – Segment Override Prefix.
13. Define – Macro and procedure
14. Define – Assembler and assembler directives
15. Define – compiler and linker
16. What is meant by modular programming?
17. Explain the uses of PUSH and POP instruction
18. Explain the uses of CALL and RET instruction
19. Identify the addressing modes in the following instructions.
    AND AL, BL
    SUB AL, 24H
    MOV AL, (BP)
    MOV CX, 1245H
20. What are the 8086 instructions used for BCD arithmetic?

Result:

Thus the program for Matrix multiplication was successfully executed.

Flowchart for Copying a String

Flowchart for Comparing two Strings



Ex. No. 4
Date:

STRING MANIPULATION, SORTING AND SEARCHING

Objective:

To write an 8086 ALP to perform the following functions

a) String Manipulation
   - Copying a String
   - Comparing Two Strings
   - Scan a character in a string

b) Sorting
   - Ascending order
   - Descending order

c) Searching

STRING MANIPULATION – COPYING A STRING

Description:

In 8086, a dedicated string instruction MOVSB is used to copy a string. On the MOVSB will move or copy the string of data pointed by SI to the location pointed by DI register on copying each byte of data, the SI register and DI register are incremented or decremented depending on the status of the direction flag DF. The CX register will hold the size of the string to be moved from one location to another location.

Algorithm:

1. Start the program.
2. Set the starting address as 1000H.
3. Get the array size & move it to CX segment.
4. Let the starting address of elements be 1200H & move it to SI.
5. Let starting address of another set of elements 1300H & move it to DI.
6. Clear Directional Flag.
7. Repeat the move single byte instruction till the count CX is zero.
8. End of segment.
9. Stop the program.

Label	Program	Comments
	ORG 1000H	Set starting address as 1000H.
	MOV CX, 0005H	Move immediate data to CX.
	MOV SI, 1200H	Move immediate data to SI.
	MOV DI, 1300H	Move immediate data to DI.
	CLD	Clear Directional Flag.
	REP MOV SB	Repeat, Move single byte
	HLT	

Example:

Input:

1200: AA
1201: AB
1202: AC
1203: DA
1204: OA

Output:

1300: AA
1301: AB
1302: AC
1303: DA
1304: OA

Observation:

Input:

1200:
1201:
1202:
1203:
1204:

Output:

1300:
1301:
1302:
1303:
1304:

Label	Program	Comments
	ORG 1000H	Set starting address as 1000H.
	MOV CX, 0005H	Move immediate data to CX.
	MOV SI, 1200H	Move immediate data to SI.
	MOV DI, 1300H	Move immediate data to DI.
	MOV BX, 3000H	Move immediate data to BX.
	CLD	Clear directional flag.
	REPE CMPSB	Repeat if equal, compare single byte
	JNZ L1	Jump if no zero to loop1.
	MOV AH, 0FFH	Move immediate data to AH.
	MOV [BX], AH	Move AH to BX register
	JMP LAST	Jump to last.
L1:	MOV AH, 00H	Move immediate data to AH.
	MOV [BX], AH	Move AH to BX register.
LAST:	HLT	

Observation:

Same String Input:

1200: 02
1201: 03
1202: 04
1203: 05
1204: 06

1300: 02
1301: 03
1302: 04
1303: 05
1304: 06

Output:

3000: FFH

Different String Input:

1200: 02
1201: 03
1202: 04
1203: 05
1204: 06

1300: 03
1301: 04
1302: 05
1303: 06
1304: 07

Output:

3000: 00H

STRING MANIPULATION – COMPARE TWO STRINGS

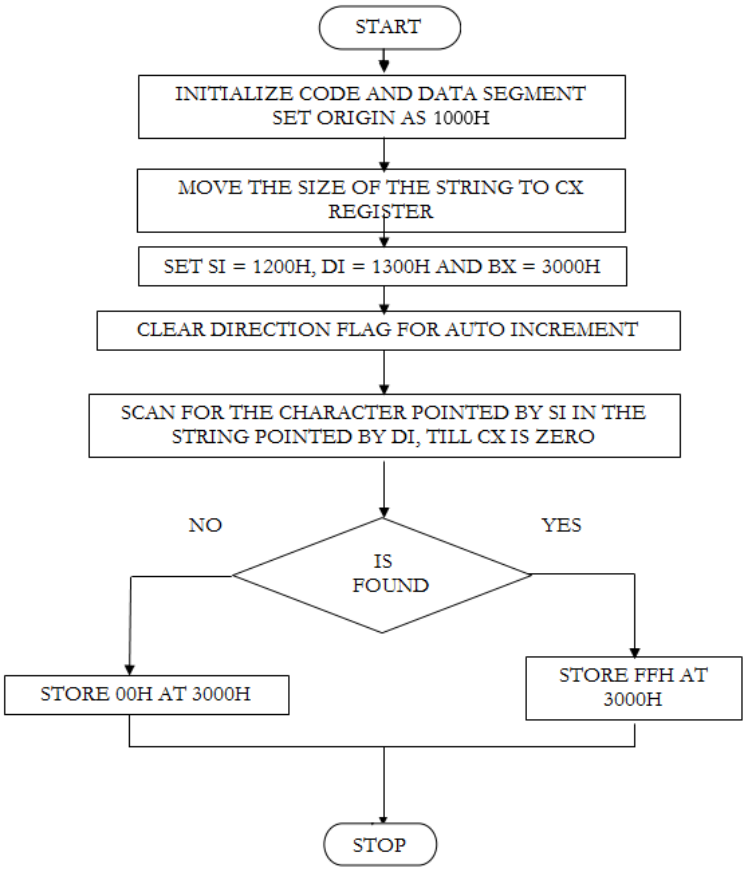
Description:

In 8086, a dedicated string instruction CMPSB is used to compare two strings. The CMPSB will compare two strings of data pointed by SI and DI register. The REPE is used to repeat compare operation for each byte of the string. If both the strings are equal the CMPSB will set zero flag. If they are unequal ZF=0. The CX register will hold the size of the string.

In this program, if both the strings are equal, 00FFH is stored at 5000H else 0000H will be stored at 5000H.

Algorithm:

1. Start the program.
2. Set the starting address as 1000H.
3. Get array size and move it to CX register.
4. The starting address of a string is moved to SI register.
5. The starting address of another string is moved to DI register.
6. The BX register is initialized to point 3000H.
7. Clear directional flag
8. Compare each byte of string pointed by SI with the string pointed by DI till CX is zero.
9. If both the strings are equal, 0FFH is stored at the location pointed by BX register (3000H). Else store 00H at the location pointed by BX register.
10. End of the segment
11. Terminate the program



STRING MANIPULATION - SCAN A CHARACTER IN A STRING

Description:

In 8086, a dedicated string instruction SCASB is used to scan a character. The SCASB will scan for the character pointed by SI, in the string pointed by DI register. If the character is available in the string zero flag is set. Else zero flag is reset. The CX register will hold the size of the string.

In this program, if the given character is available 0FFH is stored at 5000H. If it is unavailable, 00H is stored at 5000H.

Algorithm:

1. Start the program.
2. Set the origin as 1000H.
3. Move the data pointed by SI to AL register.
4. Assign 0004H [count] to CX register.
5. The starting address of the string is moved to DI register
6. Clear Directional Flag for auto increment mode.
7. Repeatedly scan for the character at AL with DI till CX is zero.
8. If the character is found in the string, store 0FFH at location 3000H pointed by BX register. Else store 00H at location 3000H pointed by BX register.
9. End of segment.
10. Stop the program.

Label	Program	Comments
	ORG 1000H	Set the starting address as 1000H.
	MOV CX, 0004H	Move immediate data to CX.
	MOV SI, 1200H	Move immediate data to SI.
	MOV AL, [SI]	Move contents of SI to AL.
	MOV DI, 1300H	Move immediate data to DI.
	MOV BX, 3000H	Move immediate data to BX.
	CLD	Clear directional flag.
	REPNE SCASB	Repeat not equal, Scan single byte
	JNZ L1	Jump if no zero to loop1.
	MOV AH, 0FFH	Move immediate data to AH.
	JMP L2	Jump to loop 2.
L1:	MOV AH, 00H	Move immediate data to AH.
L2:	MOV [BX], AH	Move AH to BX register.
	HLT	

Example:

Input:

1200:AD (Data to be scanned)

1300:AA
1301:AB
1302:AA
1303:AD

Output:

3000:FF

Input:

1200: BB (Data to be scanned)

1300:AA
1301:AB
1302:AA
1303:AD

Output:

3000:00

Observation:

Input:

1200: (Data to be scanned)

1300:
1301:
1302:
1303:

Output:

3000:

Input:

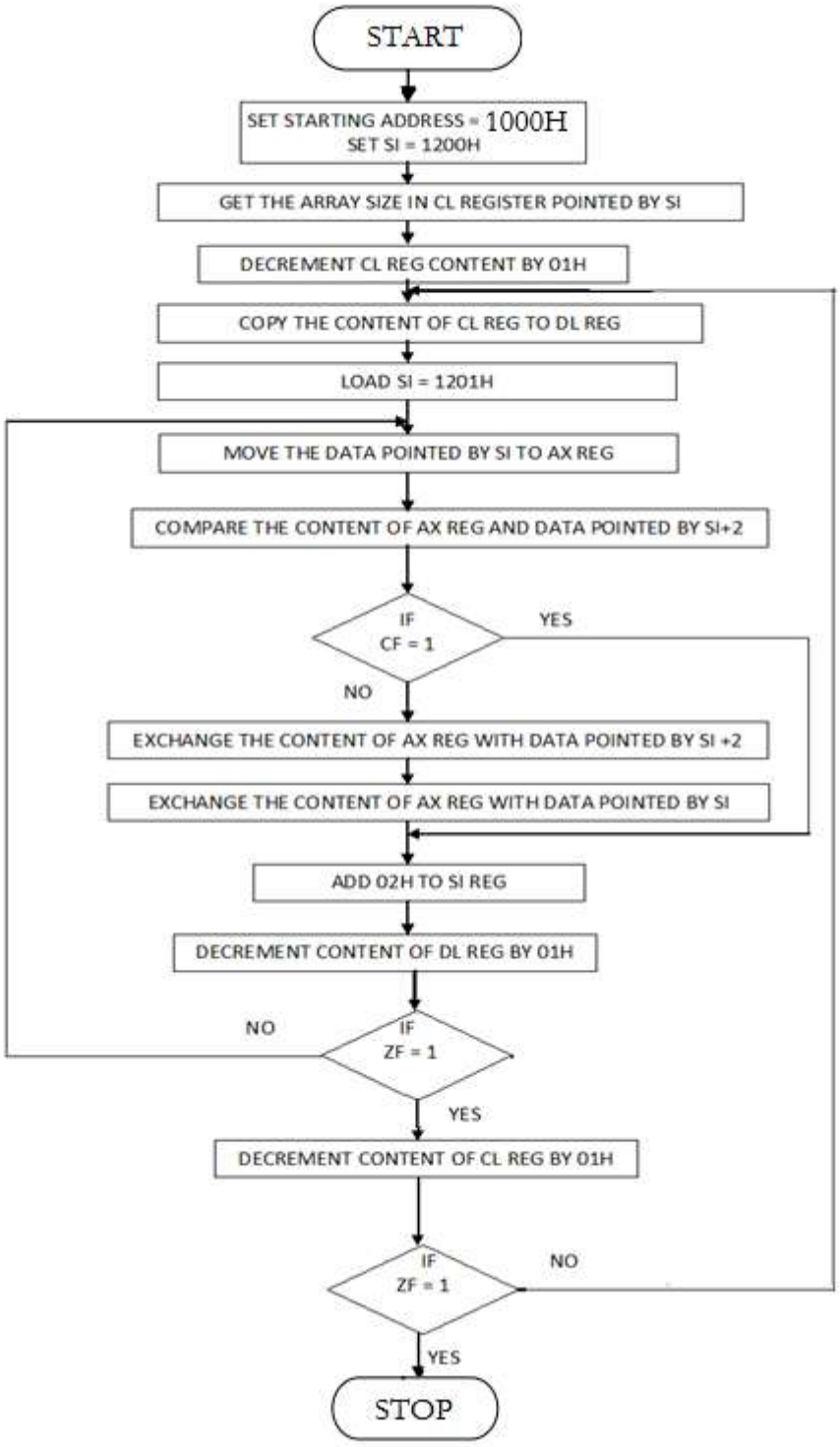
1200: (Data to be scanned)

1300:
1301:
1302:
1303:

Output:

3000:

Flow Chart for Sorting – Ascending Order



SORTING – ASCENDING ORDER

Description:

The array can be sorted in ascending order by bubble sort algorithm. In bubble sorting of M-data, M-1 comparisons are performed by tasking two consecutive data at a time. After each comparison the two data can be re-arranged in the ascending order in the same memory locations i.e., smaller first and larger next. When the above M-1 comparisons are performed M-1 times, the array will be sorted in ascending order in the same locations.

Algorithm:

1. Start the program
2. Initialize Code and Data Segment.
3. Set starting address as 1000H
4. Set SI register to 1200H address
5. Get the count in CL & decrement CL register by one
6. Copy the content of CL register to DL register.
7. Initialize SI as 1202H.
8. Move the data pointed by SI to AX
9. Compare the data in AX & data pointed by SI+2
10. If there is no carry, exchange the data and go toe next step. If there is carry go to next step.
11. Increment the content of SI by 02H
12. Decrement the content of DL register by 01H.
13. Check whether the content of DL is zero. If zero, go to step next step. Else go to step 8
14. Decrement the content of CL register by 01H.
15. Check whether the content of CL is zero. If zero, go to step next step. Else go to step 6
16. Display the result
17. Stop the program

Label	Program	Comments
	ORG 1000H	Set starting address as 1000H.
	MOV SI, 1200H	Move immediate data to SI
	MOV CL, [SI]	Move contents of SI to CL
	DEC CL	Decrement CL
L3:	MOV DL,CL	Move CL to DL register
	MOV SI, 1201H	Move immediate data to SI
L2:	MOV AX, [SI]	Move contents of SI to AX
	CMP AX, [SI+2]	Compare AX with SI
	JC L1	Jump if carry to loop1
	XCHG [SI+2], AX	Exchange data of AX with SI+2
	XCHG [SI], AX	Exchange data of AX with SI
L1:	ADD SI,02H	Increment SI twice
	DEC DL	Decrement DL register
	JNZ L2	Jump if no zero to loop 2
	DEC CL	Decrement CL register
	JNZ L3	Jump if no zero to loop 3
	HLT	

Example:

Input:

1200: 04 (Array Size)
1201: 39
1202: 40
1203: 30
1204: 78
1205: 62
1206: 42
1207: 32
1208: 38

Output:

1200: 04 (Array Size)
1201: 30
1202: 32
1203: 38
1204: 39
1205: 40
1206: 42
1207: 62
1208: 78

Observation:

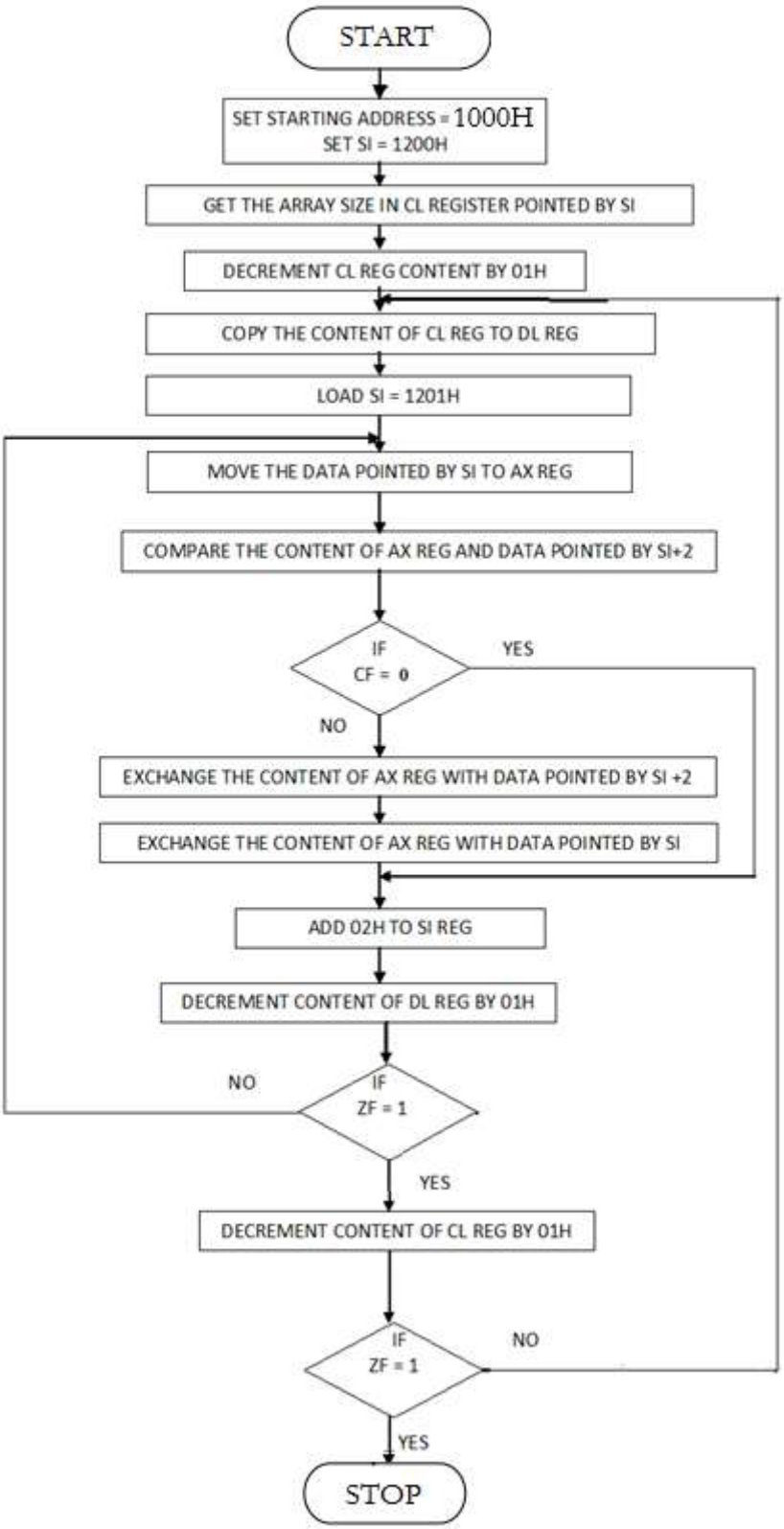
Input:

1200: 04(Array Size)
1201:
1202:
1203:
1204:
1205:
1206:
1207:
1208:

Output:

1200: 04(Array Size)
1201:
1202:
1203:
1204:
1205:
1206:
1207:
1208:

Flow Chart for Sorting - Descending Order



SORTING – DESCENDING ORDER

Description:

The array can be sorted in descending order by bubble sort algorithm. In bubble sorting of M-data, M-1 comparisons are performed by taking two consecutive data at a time. After each comparison, the two data can be re-arranged in the descending order in the same memory locations, ie., larger first and smaller next. When the above M-1 comparisons are performed M-1 timer, the array will be stored in descending order.

Algorithm:

1. Start the program
2. Set starting address as 1000H
3. Set SI register to 1200H address
4. Get the count in CL & decrement CL register by one
5. Copy the content of CL register to DL register.
6. Initialize SI as 1202H.
7. Move the data pointed by SI to AX
8. Compare the data in AX & data pointed by SI+2
9. If there is carry, exchange the data and go toe next step. If there is no carry go to next step.
10. Increment the content of SI by 02H
11. Decrement the content of DL register by 01H.
12. Check whether the content of DL is zero. If zero, go to step next step. Else go to step 8
13. Decrement the content of CL register by 01H.
14. Check whether the content of CL is zero. If zero, go to step next step. Else go to step 6
15. Display the result
16. Stop the program

Label	Program	Comments
	ORG 1000H	Set starting address as 1000H.
	MOV SI, 1200H	Move immediate data to SI
	MOV CL, [SI]	Move contents of SI to CL
	DEC CL	Decrement CL
L3:	MOV DL,CL	Move CL to DL register
	MOV SI, 1201H	Move immediate data to SI
L2:	MOV AX, [SI]	Move contents of SI to AX register
	CMP AX, [SI+2]	Compare SI+2 with AX register
	JNC L1	Jump if no carry to loop1
	XCHG [SI+2], AX	Exchange content of AX with SI+2
	XCHG [SI], AX	Exchange content of AX with SI
L1:	ADD SI, 02	Increment address of SI by 02
	DEC DL	Decrement DL register
	JNZ L2	Jump if no zero to loop 2
	DEC CL	Decrement CL register
	JNZ L3	Jump if no zero to loop 3
	HLT	

Example:

Input:

1200: 04 (Array Size)
1201:39
1202:40
1203:30
1204:78
1205:62
1206:42
1207:32
1208:38

Output:

1200: 04 (Array Size)
1201:78
1202:62
1203:42
1204:40
1205:39
1206:38
1207:32
1208:30

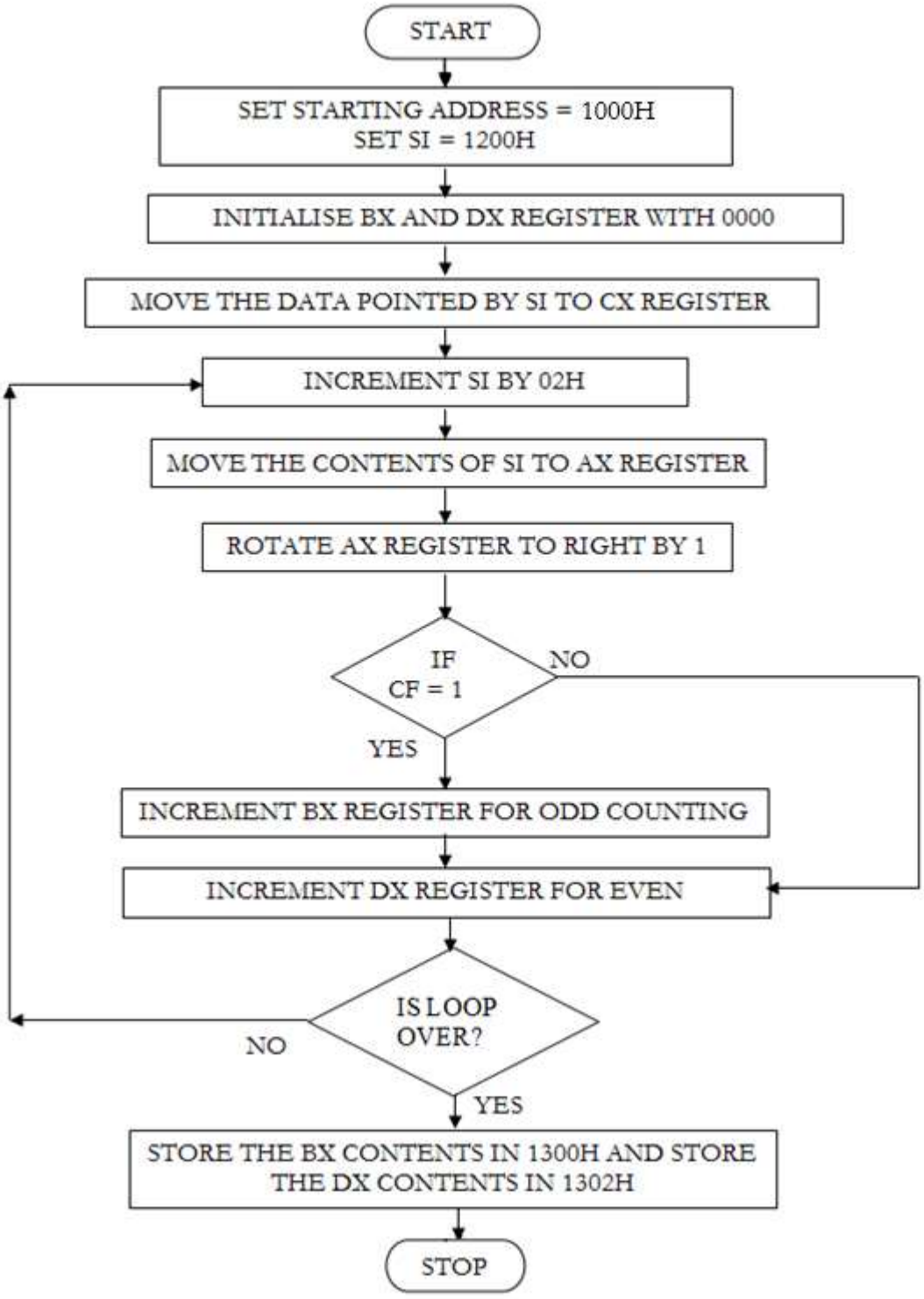
Observation:

Input:

1200: 04(Array Size)
1201:
1202:
1203:
1204:
1205:
1206:
1207:
1208:

Output:

1200: 04(Array Size)
1201:
1202:
1203:
1204:
1205:
1206:
1207:
1208:


START
SET STARTING ADDRESS = 1000H
SET SI = 1200H
INITIALISE BX AND DX REGISTER WITH 0000
MOVE THE DATA POINTED BY SI TO CX REGISTER
INCREMENT SI BY 02H
MOVE THE CONTENTS OF SI TO AX REGISTER
ROTATE AX REGISTER TO RIGHT BY 1
IF
CF = 1
NO
YES
INCREMENT BX REGISTER FOR ODD COUNTING
INCREMENT DX REGISTER FOR EVEN
IS LOOP
OVER?
NO
YES
STORE THE BX CONTENTS IN 1300H AND STORE
THE DX CONTENTS IN 1302H
STOP

SEARCHING – EVEN AND ODD NUMBERS

Description:

This program is used to count the number of even numbers and odd numbers in given array. Here one right rotate operation is performed to detect the even or odd number. After rotating operation, if carry is present, the given number is odd else it is even.

Algorithm:

1. Start the program
2. Initialize Code and Data Segment.
3. Set starting address as 1000H
4. Set SI register to 1200H address
5. Get the count in CL & decrement CL register by one
6. Initialize SI as 1202H.
7. Move the data pointed by SI to AX
8. Rotate AX register by right to one
9. If there is no carry, count the DX register for even counting else count the BX register for odd counting
10. Check loop is over or not
11. Increment the content of SI by 02H go to step 7.
12. Store the BX contents in 1300h
13. Store the DX contents in 1302h
14. Display the result
15. Stop the program

Label	Program	Comments
	ORG 1100H	Set starting address as 1100H.
	MOV SI, 1200H	Move immediate data to SI
	MOV DX, [SI]	Move contents of SI to DX
	MOV CL,01H	
	MOV BL,00H	
	MOV BH,00H	
L3:	ADD SI, 02H	INCREMENT SI BY 02H
	MOV AX, [SI]	Move contents of SI to AX
	RCR AX, CLH	Rotate AX to right by one.
	JNC L1	Jump if no carry to loop1
	INC BL	count the BL register for odd counting
	JMP L2	Jump to l2
L1:	INC BH	count the BH register for even counting
L2:	DEC DX	Count is performed until DX=0.
	JNZ L3	
	MOV DI, 1300H	
	MOV [DI],BL	Store the BL(ODD) contents in 1300h
	INC DI	
	MOV [DI], BH	Store the BH(EVEN) contents in 1301h
	HLT	

Example:

Input:

1200: 05 (Array Size)
1201:00
1202:01
1203:02
1204:04
1205:06

Output:

1300:01 odd
1301:03 even

Observation:

Input:

1200: 08 (Array Size)
1201:
1202:
1203:
1204:
1205:
1206:
1207:
1208:

Output:

1300: odd
1301: even

REVIEW QUESTIONS:

1. What is the relation between 8086 processor frequency & crystal Frequency?
2. What is the position of the stack pointer after the POP instruction?
3. Compare CALL and JMP instructions.
4. Define – Baud Rate
5. What is the size of instruction queue in 8086?
6. Compare JNC and JMP instructions.
7. What happens when HLT instruction is executed in processor?
8. What is the maximum internal clock frequency of 8086 processor?
9. What are the functions of BIU?
10. Write an ALP program to search a number 05 from a given array.
11. What is cache memory?
12. Can ROM be used as stack?
13. What are the 8086 instructions used for BCD arithmetic
14. What are the 8086 instructions used for ASCII arithmetic?
15. List the various string instructions available in 8086.
16. How will carry and zero flags reflect the result of instruction CMP BX, CX?
17. Give any four miscellaneous instructions in 16-bit Microprocessor
18. List the flags in 8086 and state its functions.
19. What is the purpose of segment registers in 8086?
20. What is virtual addressing mode?

Result:

Thus the program for string manipulations, searching and sorting operations was written and executed.

Label	Program	Comments

	DATA SEGMENT
	PASSWORD DB 'MASM1234'
	LEN EQU ($-PASSWORD)
	MSG1 DB 10,13,'ENTER YOUR PASSWORD: $'
	MSG2 DB 10, 13,'YOUR PASSWORD IS CORRECT!!$'
	MSG3 DB 10, 13, 'INCORRECT PASSWORD!$'
	NEW DB 10,13,'$'
	INST DB 10 DUP(0)
	DATA ENDS
	CODE SEGMENT
	ASSUME CS:CODE,DS:DATA
	ORG 1000H
	START:
	MOV AX,DATA
	MOV DS,AX
	LEA DX,MSG1
	MOV AH,09H
	INT 21H
	MOV SI,00
	UP1:
	MOV AH,08H
	INT 21H
	CMP AL,0DH
	JE DOWN
	MOV [INST+SI],AL	

Ex. No. 5
Date:

PASSWORD CHECKING, PRINT RAM SIZE, SYSTEM DATE

Objective:

To write an 8086 ALP to perform the following operations

d) Password Checking
e) Print RAM Size
f) Print System Date

PASSWORD CHECKING

Description:

The password checking is done using the DOS calls and functions. First Display the message "Enter your Password". Then read the pass word using Dos calls and compare with previous password "MASM1234".If it matches, then display the message password is correct. Else display it as incorrect password

Algorithm:

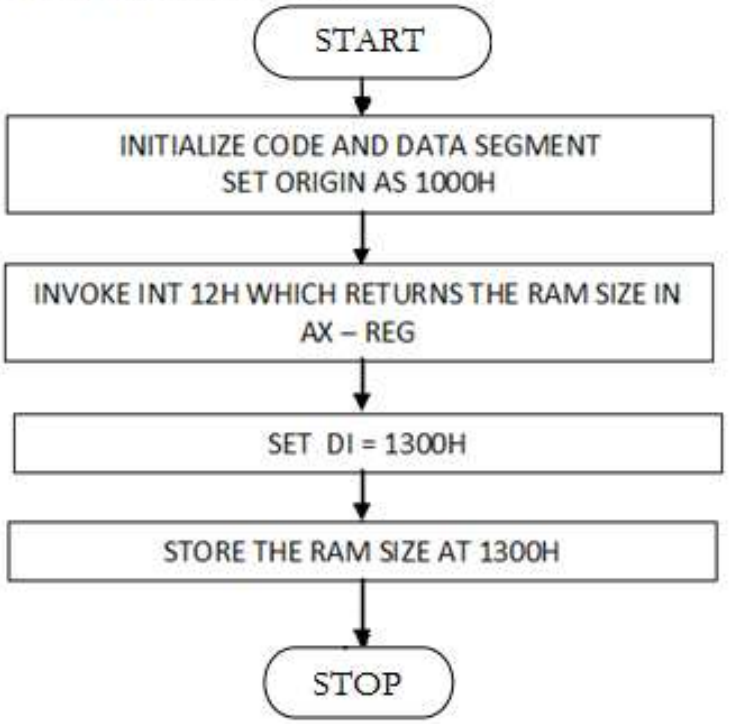
1. Start the program.
2. Set the starting address as 1000H.
3. Display the message "Enter your Password"
4. Read the pass word using Dos calls and compare with previous password "MASM1234"
5. If it matches, then display the message password is correct
6. Else display it as incorrect password
7. Stop the program.

Label	Program	Comments

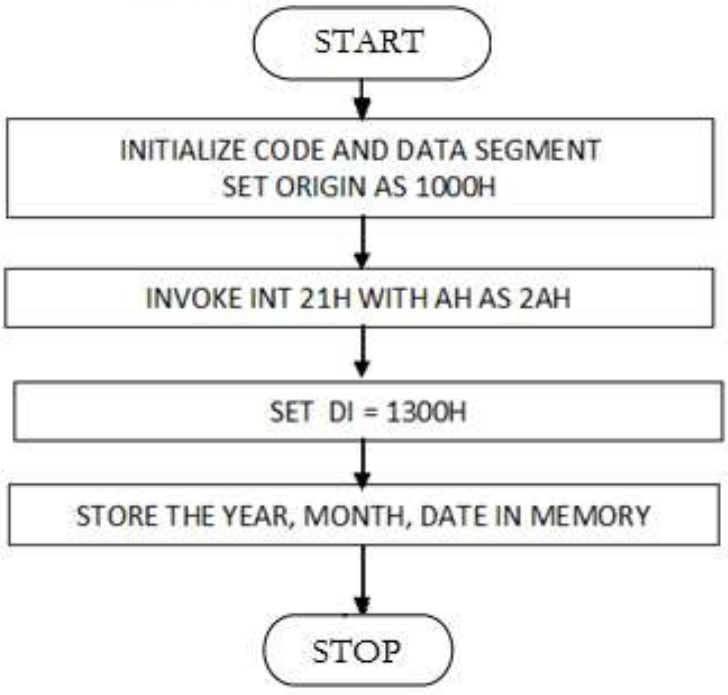
	MOV [INST+SI],AL
	MOV DL,'*'
	MOV AH,02H
	INT 21H
	INC SI
	JMP UP1
	DOWN:
	MOV BX,00
	MOV CX,LEN
	CHECK:
	MOV AL,[INST+BX]
	MOV DL,[PASSWORD+BX]
	CMP AL,DL
	JNE FAIL
	INC BX
	LOOP CHECK
	LEA DX,MSG2
	MOV AH,09H
	INT 21H
	JMP FINISH
	FAIL:
	LEA DX,MSG3
	MOV AH,009H
	INT 21H
	FINISH:
	INT 3
	CODE ENDS
	END START
	END	

Observation:

Flow Chart to Print RAM Size



Flow Chart to Print System Date



TO PRINT RAM SIZE

Description:

INT 12h interrupt stores in AX the amount of RAM memory in kilobytes. For modern computers it usually returns the value 0280h (640), representing the main memory. So this interrupt doesn't return the extended memory. The value returned in AX by this interrupt could also be found at address 0040:0013h.

Algorithm:

1. Start the program.
2. Initialize the Segments.
3. Set the starting address as 1000H.
4. Initiate INT21H which returns the RAM size in AX – reg.
5. Initialize DI as 1300H
6. Store the value at 1300H
7. End of the segment
8. Terminate the program

Program:

Label	Program	Comments
	ASSUME	Initialize Segments
	CS:CODE,DS:CODE	
	CODE SEGMENT	Set the starting address as 1000H
	ORG 1000H	12H interrupt is invoked
	INT 12H	
	MOV DI, 1300H	Store the size of the RAM at 1300H
	MOV [DI],AX	
	MOV AH,4CH	
	INT 21H	
	CODE ENDS	

Example:

Output:

1300: 80

Observation:

Output:

1300:

Program:

Label	Program	Comments
	ASSUME CS:CODE,DS:CODE	Initialize Segments
	CODE SEGMENT ORG 1000H	Set the starting address as 1000H
	MOV AH,2AH INT 21H	21H interrupt is invoked
	MOV DI, 1300H MOV [DI],CX	Store the year at 1300H
	ADD DI,02H MOV [DI],DX	Store the value of Month and day
	MOV AH,4CH INT 21H CODE ENDS	

Manual Calculation:

TO PRINT SYSTEM DATE

Description:

INT 21h interrupt with AH as 2AH will return the system date. The year (1980 – 2099) will be returned in CX register. The month will be available in DH register and day will be available in DL register. All the returned values will be in Hex.

Algorithm:

1. Start the program.
2. Initialize the Segments.
3. Set the starting address as 1000H.
4. Initiate INT21H with AH value as 2A H.
5. Initialize DI as 1300H
6. Store the value of year at 1300H
7. Store the value of Month and Day in the consecutive memory locations
8. End of the segment
9. Terminate the program

Example:

Output:

1300:	D	(Year)	1302:	0B	(Day)
1301:	07		1303:	08	(Month)

Observation:

Output:

1300: (Year)
1301:
1302: (Day)
1303: (Month)

REVIEW QUESTIONS:

1. What is the role of Stack?
2. What is the difference between DOS and BIOS interrupts?
3. What is an interrupt vector Tabulation: of 8086?
4. What .model small stands for?
5. Define – Interrupt Vector Tabulation
6. What are the contents of AL and CY after the execution of the following segments?
7. What is the purpose of CLK signal in an 8086 system?
8. What is the need for MN/MX pin in 8086 system?
9. What is the purpose of QUEUE in 8086 processor?
10. Give the operation of TEST instructions of 8086?
11. List few string instructions of 8086?
12. What is the use of LOCK prefix?
13. What is the purpose of REP prefix?
14. What are the types of Multiprocessor configuration?
15. Define – Co-processor?
16. List any four program control instructions available in 8086?
17. How the data and address lines are demultiplexed?
18. Define – Instruction
19. Define – Machine cycle
20. Define – T-State

Result:

Thus the program for password checking, printing RAM size, and System date was written and executed.

PROGRAM

Label	Program	Comments
	ASSUME CS:CODE,DS:CODE	
	CODE SEGMENT	
	ORG 1000H	
	MOV CL,08H	
	MOV DI,1200H	
	MOV AL,80H	
	MOV [DI],AL	
ABOVE:	INC DI	
	SAR AL,1	
	MOV [DI],AL	
	DEC CL	
	JNZ ABOVE	
	MOV CL,08H	
NEXT:	SHR AL,1	
	MOV [DI],AL	
	INC DI	
	DEC CL	
	JNZ NEXT	
	MOV AH, 4CH	
	INT 21H	
	CODE ENDS	
	END	

Ex. No. 6
Date:

COUNTERS AND TIME DELAY

Objective:

To write an Assembly Language Program (ALP) for 8 bit Johnson counter and creating time delay

8 BIT JOHNSON COUNTER

Description:

The Johnson counter is a special type of counter whose truth table is given below.

CLK	DATA								HEX
1	1	0	0	0	0	0	0	0	80
2	1	1	0	0	0	0	0	0	C0
3	1	1	1	0	0	0	0	0	E0
4	1	1	1	1	0	0	0	0	F0
5	1	1	1	1	1	0	0	0	F8
6	1	1	1	1	1	1	0	0	FC
7	1	1	1	1	1	1	1	0	FE
8	1	1	1	1	1	1	1	1	FF
9	0	1	1	1	1	1	1	1	7F
10	0	0	1	1	1	1	1	1	3F
11	0	0	0	1	1	1	1	1	1F
12	0	0	0	0	1	1	1	1	0F
13	0	0	0	0	0	1	1	1	07
14	0	0	0	0	0	0	1	1	03
15	0	0	0	0	0	0	0	1	01
16	0	0	0	0	0	0	0	0	00

Initialize AL with 8000H, the first value in truth table. Shift the data by inserting the same value of MSB of previous data. Repeat the same by shifting left to generate next half of truth table

Algorithm:

1. Start the program.
2. Set the origin as 1000H.
3. Load the count 08 in CL register
4. Initialize DI with 1200H
5. Initialize AL with 8000H, the first value in truth table
6. Shift the data by inserting the same value of MSB of previous data
7. Check for count. If it is not zero go to step 6. Else go to next step.
8. Repeat the same by shifting left to generate next half of truth table
9. Store the result
10. Stop the program.

Example:
Output:
1200: 80
1201: C0
1202: E0
1203: F0
1204: F8
1205: FC
1206: FE
1207: FF
1208: 7F
1209: 3F
120A: 1F
120B: 0F
120C: 07
120D: 03
120E: 01
120F: 00

OBSERVATION:
Output:
1200:
1201:
1202:
1203:
1204:
1205:
1206:
1207:
1208:
1209:
120A:
120B:
120C:
120D:
120E:
120F:

Label	Program	Comments
	DATA SEGMENT	
	MSG2 DB 10,13,'Time $'	
	MSG1 DB 10,13,' delay!!$'	
	DATA ENDS	
	ASSUME CS:CODE,DS:DATA	
	CODE SEGMENT	
	ORG 1000H	
	MOV AX,DATA	
	MOV DS,AX	
	LEA DX,MSG2	
	MOV AH,09H	
	INT 21H	
	MOV BL,20	
BACK:	MOV CX,33150	
WAITF1:	IN AL,61H	
	AND AL,10H	
	CMP AL,AH	
	JE WAITF1	
	MOV AH,AL	
	LOOP WAITF1	
	DEC BL	
	JNZ BACK	
	MOV AX,DATA	
	MOV DS,AX	
	LEA DX,MSG1	
	MOV AH,09H	
	INT 21H	
	MOV AH,4CH	
	INT 21H	
	CODE ENDS	
	END	

TIME DELAY

Description:

Time delays are often needed for various applications. Using the instructions of the x86 CPU to generate the delay is unreliable since the CPU speed varies among the x86 PCs. To create a CPU-independent delay, x86 makes PB4 of port 61H toggle every 15.085 microseconds. That means that by monitoring PB4 of port 61H, a fixed time delay can be obtained, as shown below. Upon entering this sub-routine called WAITF, register CX must hold the number of 15.085-microsecond time delays needed.
Since the 1.5-second delay requires the counter to be set to 99,436 (1.5s/15.085 μs = 99,436) and the maximum value of CX is 65,536, the another register is used to generate the 1.5-second delay

Algorithm:

1. Start the program.
2. Set the origin as 1000H.
3. Display the message “Time”
4. load the count value to CX and BL reg
5. Generate the time delay
6. Display the message “Delay” after the required delay
7. Stop the program.

OUTPUT:

Time Delay!!

OBSERVATION:

REVIEW QUESTIONS:

1. What are the 8086 instructions used for BCD arithmetic?
2. What is the function of BX register?
3. How Physical address is generated?
4. List out the pointers available in 8086
5. Compare PUSH and PULL instructions
6. What is ALE? When will the data bus AD0-AD7 be enabled?
7. Define – HOLD in 8086
8. Define – HLDA in 8086
9. Give the significance of RQ/GTO and IO/M signals.
10. Name any two coprocessors and their use.
11. State the importance of sample and hold circuit
12. List the applications of programmable interval timer.
13. What is key denouncing? What are the methods to detect the denouncing?
14. Name the two modes of operation of DMA controller?
15. Give the different types of command words in 8259
16. Give the comments for MOV r,M.
17. How many T-states are in MOV instruction?
18. Explain the addressing mode of MOV r,M
19. How many machine cycles are in MOV instruction?
20. Give the comments for MOV M,r

RESULT:

Thus the program to generate Johnson counting pattern and to generate time delay has been executed successfully.

Ex. No. 7
Date:

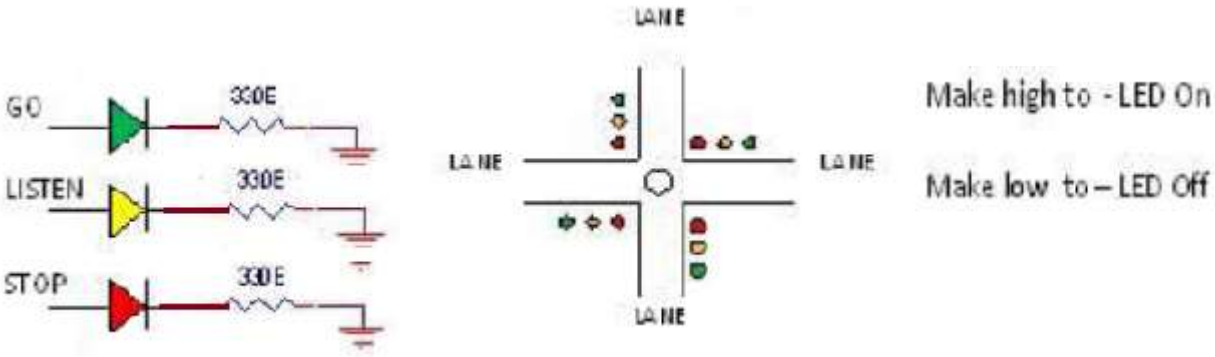
TRAFFIC LIGHT CONTROLLER

AIM

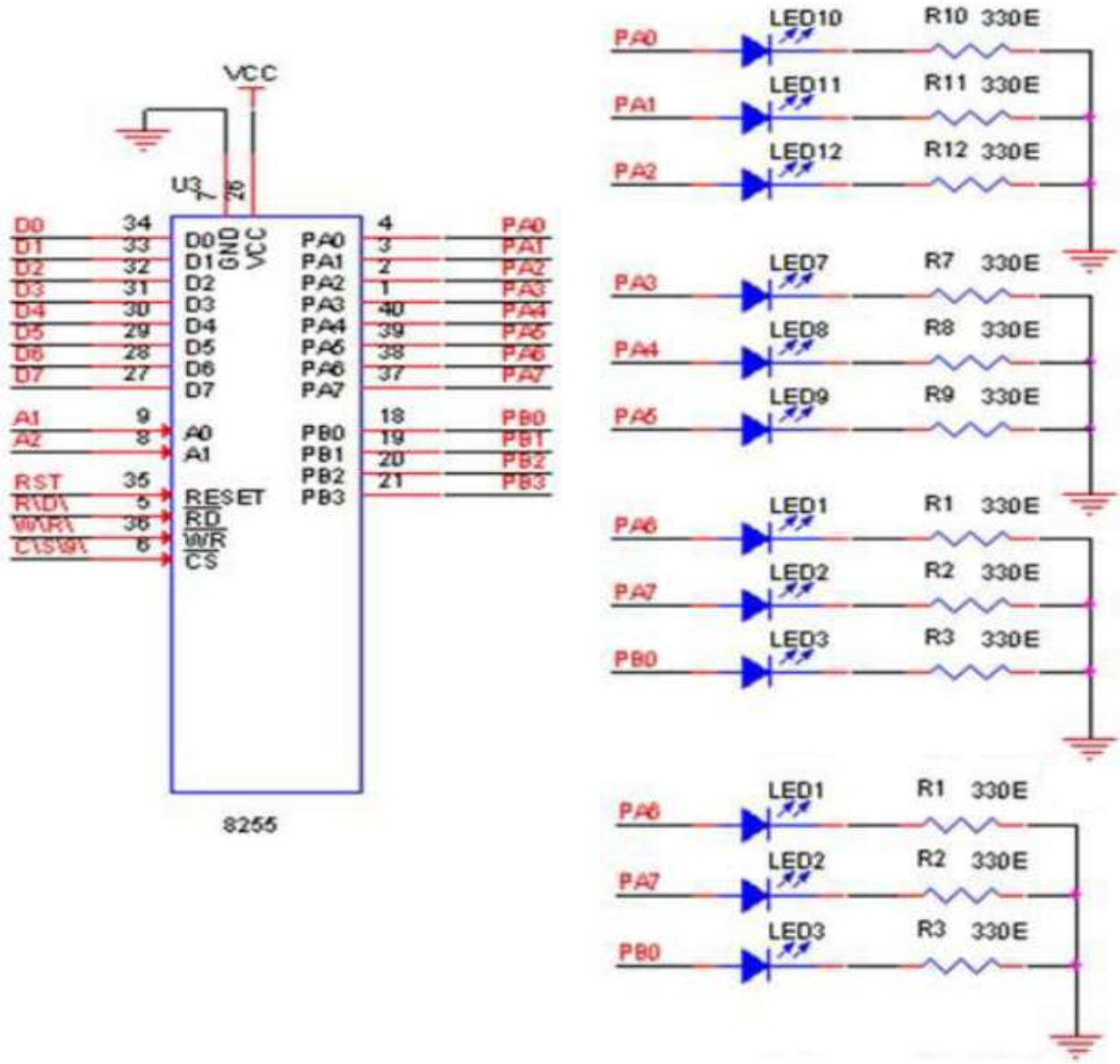
To write an 8086 assembly language program to interface the traffic light controller with 8255 and verify the operation.

DESCRIPTION

The system is a simple contraption of a traffic control system wherein the signaling lights are simulated by the blinking or ON-OFF control of light-emitting diodes. The signaling lights for the pedestrian crossing are simulated by the ON-OFF control of dual colour light emitting diodes.
A model of a four road – four lane junctions, the board has green, orange and red signals of an actual system. Twelve LEDs are used on the board. In addition eight dual colour LEDs are used which can be made to change either to red or to green.



CIRCUIT DIAGRAM TO INTERFACE TRAFFIC LIGHT WITH 8086



Program:

Label	Mnemonics
START	ORG 1100H
	MOV BX, 1200
	MOV CX, 000C
	MOV AL, [BX]
	OUT 26, AL
	INC BX
	MOV AL, [BX]
	OUT 20, AL
	INC BX
	MOV AL, [BX]
	OUT 22, AL
	CALL DELAY
	INC BX
	LOOP NEXT
	JMP START
DELAY	PUSH CX
	MOV CX,0005
REPEAT	MOV DX, FFFF
AGAIN	DEC DX
	JNZ AGAIN
	LOOP REPEAT
	POP CX
	RET

OBSERVATION

INPUT

1200: 80, 1A, A1, 64
1204: A4, 81, 5A, 64
1208: 54, 8A, B1, A8
120C: B4, 88, DA, 68
1210: D8, 1A, E8, 46
1214: E8, 83, 78, 86, 74

OUTPUT

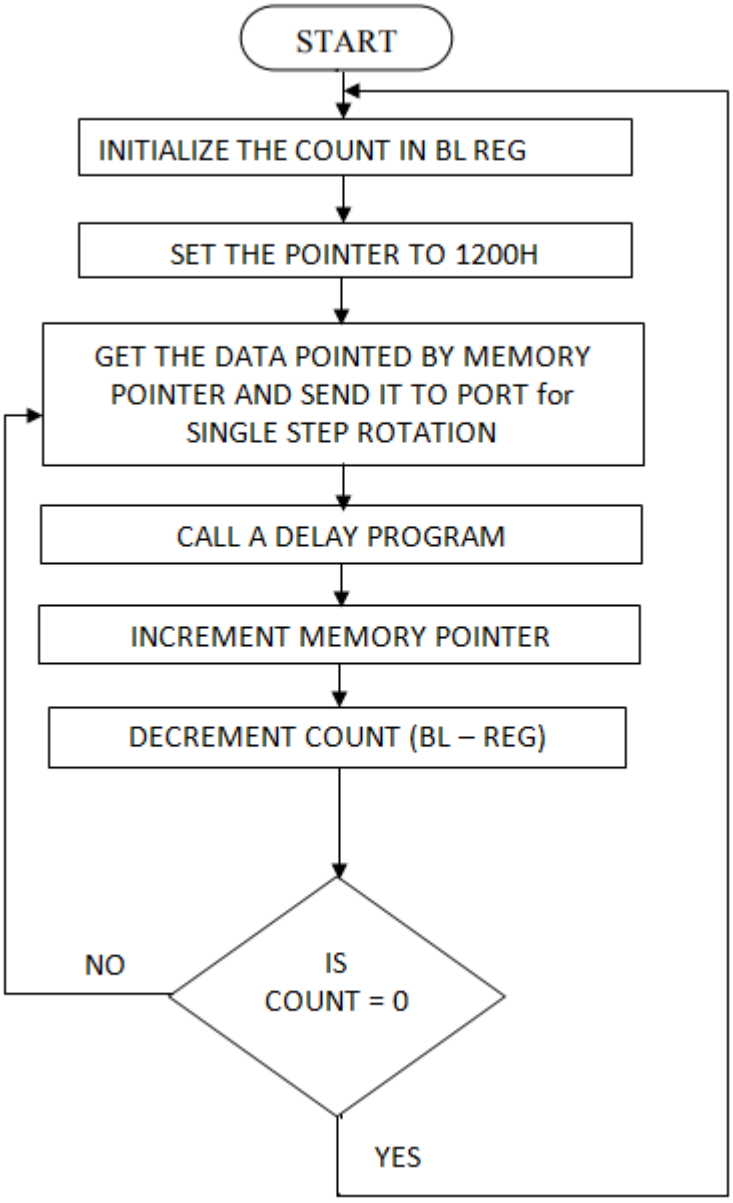
REVIEW QUSETIONS:

1. List out the control ports in traffic light controller
2. What are the functions of conditional instructions?
3. List out the LAN ports in traffic light controller
4. What are the functions of Loop instructions?
5. List out the Modules in traffic light controller
6. List out the control ports in traffic light controller
7. What are the functions of conditional instructions?
8. List out the LAN ports in traffic light controller
9. What are the functions of Loop instructions?
10. List out the Modules in traffic light controller
11. List out the difference between INT 0 and INT 4.
12. Describe the steps required in the execution of an assembly language program.
13. Explain the use of EXTRN and PUBLIC directives with an example.
14. Explain the memory structure in a general purpose desktop computer Illustrate the use of following assembler directives: DD, DW, EVEN, GROUP, ORG, ASSUME, ENDP, PTR, OFFSET
15. Discuss how “even” and “odd” memory banks are accessed using control signals.

RESULT

Thus the interface the traffic light controller using 8086 microprocessor with 8255 has been executed and verified.

Flow Chart


START
INITIALIZE THE COUNT IN BL REG
SET THE POINTER TO 1200H
GET THE DATA POINTED BY MEMORY POINTER AND SEND IT TO PORT for SINGLE STEP ROTATION
CALL A DELAY PROGRAM
INCREMENT MEMORY POINTER
DECREMENT COUNT (BL – REG)
IS COUNT = 0
NO
YES

Ex No: 8
Date:

INTERFACING STEPPER MOTOR WITH 8086 MICROPROCESSOR

AIM:

To write an 8086 assembly language program to interface stepper motor and vary speed of motor, direction of motor.

THEORY:

A motor in which the rotor is able to assume only discrete stationary angular position is a stepper motor. The rotor motion occurs in a stepwise manner from one equilibrium position to next.

The motor under our consideration uses 2 – phase scheme of operation. In this scheme, any two adjacent stator windings are energized. The switching condition for the above said scheme is shown in Table.

Clockwise				Anti - Clockwise			
A1	**B1**	**A2**	**B2**	**A1**	**B1**	**A2**	**B2**
1	0	1	0	1	0	0	1
0	1	1	0	0	1	0	1
0	1	0	1	0	1	1	0
1	0	0	1	1	0	1	0

In order to vary the speed of the motor, the values stored in the registers R1, R2, R3 can be changed appropriately.

ALGORITHM:

1. Store the look up table address in DI
2. Move the count value (04) to one of the register (BL)
3. Load the Data to accumulator for motor rotation.
4. Send data to out port 1 for single step rotation of motor.
5. Call the delay program
6. Decrement value of BL, if not zero go to step 3.
7. Perform steps 1 to 6 repeatedly.

PROGRAM:

Label	Program	Comments
	ORG 1000H	Set starting address as 4100H.
START:	MOV BL,04H	Load 04h to BL register
	MOV DI, LOOK UP (1200)	Load data from look up table
AGAIN:	MOV AL, [DI]	Load Data to AL from DI
	OUT PORT1(C0), AL	Send data to out port1 for rotation
	CALL DELAY	Call delay for one movement of rotation
	INC DI	Increment DI to load another data
	DEC BL	Decrement BL if not zero goes to
	JNZ AGAIN	Again
	JMP START	
DELAY:	MOV DX, 1010H (SPEED)	Go to Starting position
RPT:	NOP	
	DEC DX	Delay program
	JNZ RPT	
	RET	
LOOK UP:	09, 05, 06, 0A	
	(ANTICLOCKWISE)	
	0A,06, 05, 09	
	(CLOCKWISE)	

REVIEW QUSETIONS:

1. What is meant by Prefix?
2. Difference between small, medium, tiny, huge?
3. Define –DD, DW and DB.
4. List out the Interrupts in 8086
5. What is meant by half wave scheme?
6. Give examples for 8 / 16 / 32 bit Microprocessor?
7. What is 1st / 2nd / 3rd / 4th generation processor?
8. What is meant by interrupt?
9. What is meant by Scratch pad of computer?
10. What is NV – RAM?
11. Which interrupts are generally used for critical events?
12. What is the position of the Stack Pointer after the PUSH instruction?
13. What is the position of the Stack Pointer after the POP instruction?
14. Logic calculations are done in which type of registers?
15. Explain how to generate the physical address with respect to code segment and any other segment.

RESULT:

Thus the speed and direction of motor were controlled using 8086 trainer kit.

Ex No: 9
Date:

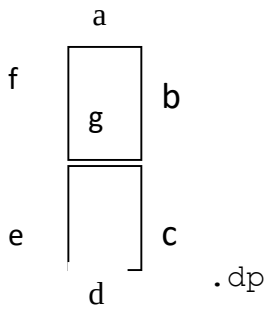
DIGITAL CLOCK

AIM:

To write an 8086 assembly language program to interface digital clock

DESCRIPTION:

The Keyboard section consists of 16 keys, besides two keys for CNTL and SHFT. The key board is arranged as two rows of eight keys each. The keyboard and display is configured in the encoded mode. Display is a 6 digit, multiplexed display.
The segment definitions of the 7 segment display are shown below.



In order to light up a segment the corresponding bit of data has to be written into the display RAM should be "0".

For example to display the character 'A' the segments except decimal point (dp) and d all other segments should be ON. Hence

The data to display "A" will be "88".

The 8279 controller IC has 10 numbers of control words. Display/Keyboard mode set word and clear word will take care of basic initialization of 8279. However, before sending codes to the display RAM, a write display RAM control word should be sent.

Then the data is fetched from 4500H and displayed in the first digit of the display. The next data is displayed in the second digit of the display, since in the command word for 'write display RAM' auto increment flag is set.

A time delay is given between successive digits for a lively display.

Data Bus	D7	D6	D5	D4	D3	D2	D1	D0
Segments	d	c	b	a	dp	g	f	e

Program:

Memory Location	Label	Mnemonics
1000	START	CALL
		CONVERT
		CALL DISPLAY
1006	DELAY	MOV AL,BOH
		OUT 16H,AL
		MOV CL,07H
100E	S2	MOV AL,88H
		OUT 14H,AL
		MOV AL,80H
		OUT 14H,AL
1018	S1	MOV AL,80H
		OUT 16H,AL
		NOP
		NOP
		NOP
		NOP
1021		IN AL,14H
		MOV DL,AL
		IN AL,14H
		OR AL,DL
		JNZ S1
		DEC CL
102D		JNZ S2
		MOV SI,1500H
		MOV AL,[SI]
		INC AL
		MOV [SI],AL
		CMP AL,3CH
103C		JNZ START
103E		MOV AL,00H
		MOV [SI],AL
		INC SI
		MOV AL,[SI]
		INC AL
		MOV [SI],AL
		CMP AL,3CH
104D		JNZ START
		MOV AL,00H
		MOV [SI],AL
		INC SI
		MOV AL,[SI]
		INC AL
		MOV [SI],AL
		CMP AL,18H
		JNZ START
		MOV AL,00
		MOV [SI],AL
1065		JMP START
1068	DIPLAY	MOV AH,06H
		MOV DX,1600H
		MOV CH,01H
		MOV CL,0H
		INT 5
		RET
1078	CONVERT	MOV SI,1500H
		MOV BX,1608H
		MOV AL,24H
		MOV [BX],AL
1085	SECONDS	MOV AL,[SI]
		MOV AH,00
		MOV DH,0AH
		DIV DH
108F		ADD AH,30H
		DEC BX
		MOV [BX],AH
		DEC BX
1096		ADD AL,30H
		MOV [BX],AL
		DEC BX
		MOV AL,3AH
		MOV [BX],AL
10A1		DEC BX
10A2	MINUTES	INC SI
		MOV AL,[SI]
		MOV AH,00H
		MOV DH,0AH
10AB		DIV DH
		ADD AH,30H
		MOV [BX],AH
		DEC BX
10B3		ADD AL,30H
		MOV [BX],AL
		DEC BX
		MOV AL,3AH
		MOV [BX],AL
10BE		DEC BX
	HOURS	INC SI
		MOV AL,[SI]
		MOV AH,00H
		MOV DH,0AH
		DIV DH
10CA		ADD AH,30H
		MOV [BX],AL
		DEC BX
10D0		ADD AL,30H
		MOV [BX],AL
		RET
		GETC
		IN AL,02H
		AND AL,FFH
		CMP AL,FOH
		JNE GETC

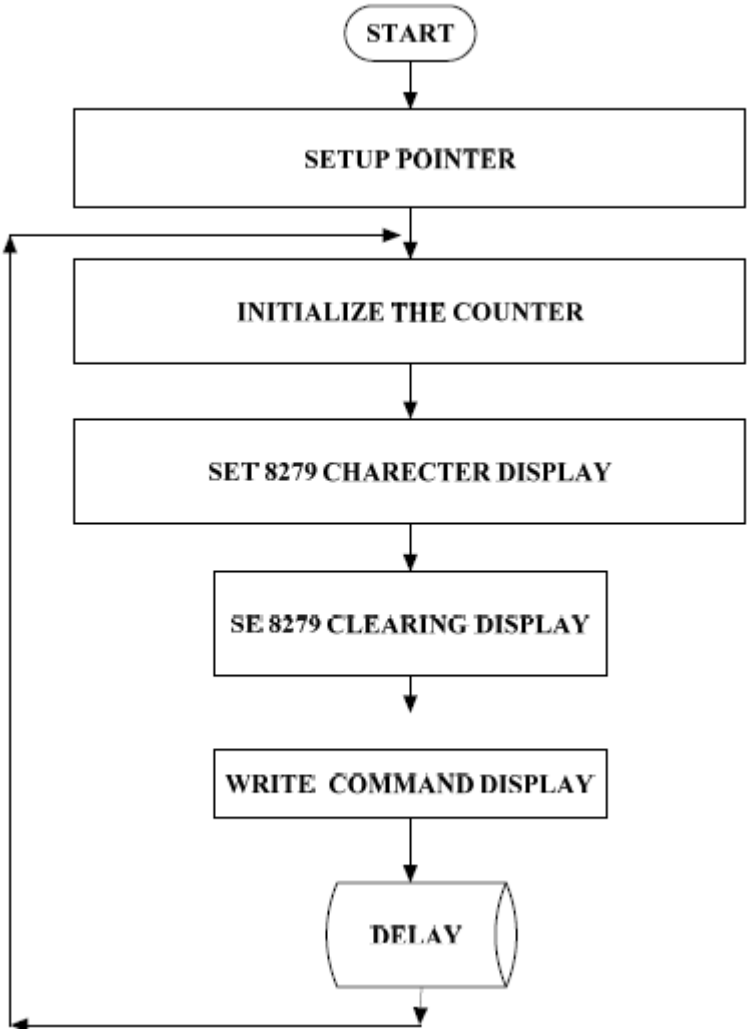
REVIEW QUSETIONS:

1. What is the difference between near and far procedure?
2. What are the different string instructions of 8086?
3. What is the difference between near and far procedure?
4. What is the difference between macro and sub-routine?
5. What are the functions of SI and DI registers?
6. Discuss the use of following instructions:
    a. CLI
    b. LOOP
    c. CALL
    d. AAM
7. Define – ALE
8. Where is the READY signal used?
9. What is the need for timing diagram?
10. What operation is performed during first T-state of every machine cycle in 8085?
11. What is interrupt acknowledge cycle?
12. What is vectored and non-vectored interrupt?
13. List the software and hardware interrupts of 8085?
14. Define – TRAP
15. How clock signals are generated in 8085 and what is the frequency of the internal clock?

Result:

Thus the program for Digital clock has been executed successfully.

Flow Chart


START
SETUP POINTER
INITIALIZE THE COUNTER
SET 8279 CHARECTER DISPLAY
SE 8279 CLEARING DISPLAY
WRITE COMMAND DISPLAY
DELAY

Ex. No. 10
Date:

INTERFACING KEYBOARD/DISPLAY CONTROLLER (8279) WITH 8086 MICROPROCESSOR

AIM:

To write an 8086 assembly language program to interface the 8279 and display the register number, as rolling message.

ALGORITHM:

1. Set the pointer to 1200H
2. Initialize the counter (CX – Reg) to 0FH
3. Send Mode Display Command word (10H) to C2H.
4. Send Clear Display Command word (0CCH) to C2H.
5. Send Write Display Command Word (90H) to C2H
6. Get the data pointed by pointer
7. Output it to C0H
8. Call a Delay program for Lively display
9. Increment memory pointer to point next data.
10. Decrement count.
11. Check if Count is zero. If yes go to step 1. Else go to step 6

Program

Label	Program	Comments
START:	ORG 1000H	Set starting address as 1000H.
	MOV SI, 1200H	Set Pointer
	MOV CX, 000FH	Initialize counter.
	MOV AL, 10H	Set Mode and Display
	OUT C2H, AL	
	MOV AL, 0CCH	Clear display.
	OUT C2H, AL	
	MOV AL, 90H	Write Display
	OUT C2H, AL	
NXTCHR:	MOV AL, [SI]	Get the data
	OUT C0H, AL	
	CALL DELAY	Call Delay program for lively display
	INC SI	Increment Pointer
	LOOP NXTCHR	Decrement Count, If not zero go to NXTCHR
	JMP START	
DELAY:	MOV DX, 0A0FFH	
LOOP1:	DEC DX	
	JNZ LOOP1	
	RET	

DISPLAY MODE SETUP:

0	0	0	D	D	K	K	K

DD		DISPLAY MODE
00	8	8-bit character display – Left Entry
01	16	8-bit character display – Left Entry
10	8	8-bit character display – Right Entry
11	6	8-bit character display – Right Entry

KKK	KEYBOARD MODE
000	Encoded Scan Keyboard – 2 key lock out
001	Decoded Scan Keyboard – 2 key lock out
010	Encoded Scan Keyboard – N Key Roll Over
011	Decoded Scan Keyboard – N Key Roll Over
100	Encoded Scan Sensor Matrix
101	Decoded Scan Sensor Matrix
110	Strobed input, Encoded Display Scan
111	Strobed input, Decoded Display Scan

CLEAR DISPLAY:

1	1	0	CD	CD	CD	CF	CA

CD CD CD - The lower two CD bits specify the blanking code to be sent to the segments to turn them OFF while the 8279 is switching from one digit to next

CD CD CD

Enables clear display when CD = 1.
The rows of display RAM are cleared by the code set by lower two CD Bits.
If CD = 0 then the contents of RAM will be displayed

0 X	A0 – 3	B0 – 3 = 00 = 0000 0000
1 0	A0 – 3	B0 – 3 = 20 = 0010 0000
1 1	A0 – 3	B0 – 3 = FF = 1111 1111

CF – If CF = 1, FIFO status is cleared, Interrupt output line is reset. Sensor RAM pointer is set to row 0.
CA – Clear All bit has the combined effect of CD and CF. It uses CD clearing code on Display RAM and clears FIFO status.

WRITE DISPLAY RAM:

1	1	0	AI	A	A	A	A

AI – Auto Increment Flag. If AI = 1, the row address selected will be incremented after each read or write to the Display RAM.
AAA – Selects one of the 16 rows of the display RAM.

Example:

Input Data:

1200:FF
1201:FF
1202:28
1203:0C
1204:1A
1295:FF
1206:98
1207:68
1208:7C
1209:C8
120A:FF
120B:1C
120C:29
120D:F7
120E:FF

Output:

GOD HELP US

Observation:

Input Data:

1200:
1201:
1202:
1203:
1204:
1295:
1206:
1207:
1208:
1209:
120A:
120B:
120C:
120D:
120E:
120F:

Output:

REVIEW QUSETIONS:

1. What is the size of flag register?
2. Can you perform 32 bit operation with 8086? How?
3. What is the difference between instructions DIV & IDIV?
4. What is the size of each segment?
5. What is the difference between instructions MUL & IMUL?
6. What is meant by LED/LCD?
7. How do you place a specific value in DPTR register? (Dec 2013)
8. Which of the 8051 ports need pull-up registers to functions as I/O port ? (Dec 2013)
9. What are the control words of 8251A and what are its functions?
10. What are the display modes supported by the 8279 chip?
11. Give the format of program clock word of 8279 and mention its purpose.
12. What is 2 key lockout and n key rollover?
13. Define – PPI
14. What is the use of direction flag?
15. What are the alternate functions of port0, port1, port2 and port3?

Result:

Thus the program to display the register number, as rolling message, in the display by interfacing 8279 with 8086 was done successfully.

Ex. No. 11
Date:

PRINTER INTERFACE

Aim:

To write an ALP to print a single character "A" by checking the printer status using 8086 Kit and Printer Interface.

Description:

The Printer is initialized by writing 05 to the control register which makes STROBE high and SELECT low. Then check for the BUSY and ERROR signals sent out by the printer by reading its status. After the printer is ready, the ASCII code for "A" is sent to the printer. A carriage return (OD Hex) is sent as the next character in order to print the character.

In the CHECK routine the printer status is again read and the printer is checked for the paper error signals. If no error is encountered, then the data will be printed in the paper by making STROBE low and high after 1 µS.

Algorithm:

1. Start the program.
2. Set the origin as 1000H.
3. Initialize the printer interface with 05 to make strobe high and select low.
4. Check for busy and error signals. If printer is not ready wait. Else go to next step
5. Print the character.
6. Stop the program.

Program

Label	Program	Comments
	ORG 1000H	
	MOV AL,05H	
	OUT D0,AL	
	IN AL,C0H	
	AND AL,20H	
	CMP AL,20H	
	JNZ ERR	
	MOV AL,41H	
	CALL PRINT	
	MOV AL,0AH	
	CALL PRINT	
	HLT	
PRINT:	MOV BL,AL	
	CALL CHECK	
STAS:	MOV AL,BL	
	OUT C8,AL	
	MOV AL,01H	
	OUT D0,AL	
	NOP	
	NOP	
	NOP	
	MOV AL,05H	
	OUT D0,AL	
	RET	
CHECK:	IN AL,C0H	
	AND AL,20H	
	JZ CHECK	
	IN AL,C0H	
	AND AL,80H	
	CMP AL,80H	
	JNZ STAS	
	JMP CHECK	
	HLT	
ERR:	INT 2	

OUTPUT: A

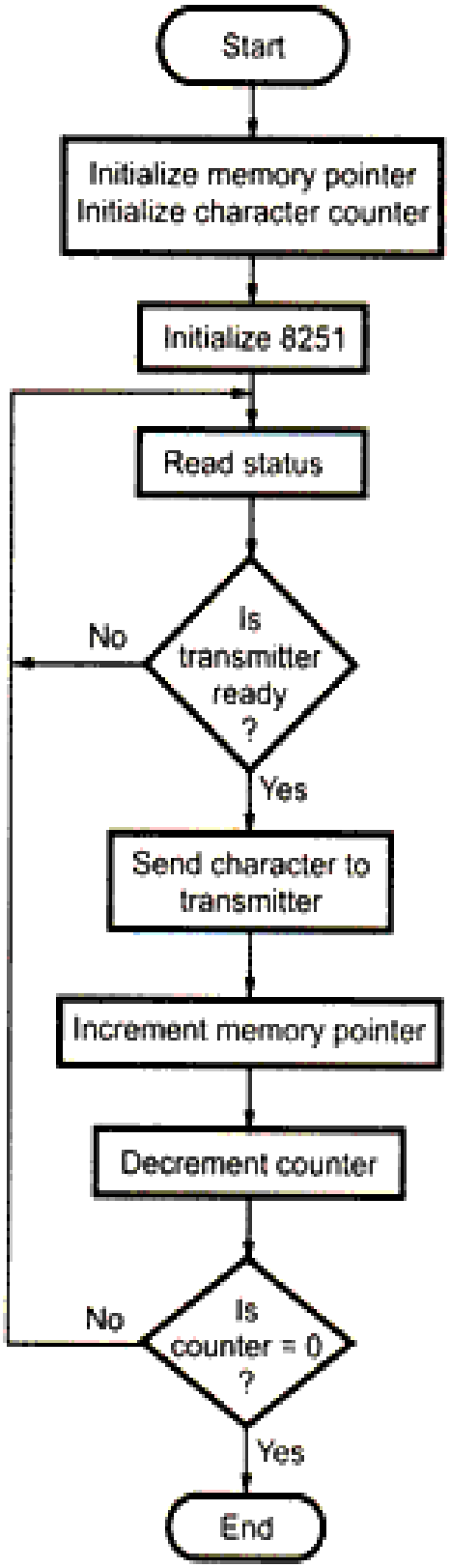
REVIEW QUSETIONS:

1. In what mode the printer interfaced?
2. Which IC used interface microprocessor and printer?
3. Define Strobe mode.

Result:

Thus the program to print a single character "A" by checking the printer status using 8086 Kit and Printer Interface was executed successfully.

Flowchart


Start
Initialize memory pointer
Initialize character counter
Initialize 8251
Read status
Is transmitter ready ?
No
Yes
Send character to transmitter
Increment memory pointer
Decrement counter
Is counter = 0 ?
No
Yes
End

Ex. No. 12
Date:

SERIAL INTERFACE AND PARALLEL INTERFACE

Aim:

To write an ALP to demonstrate

(a) Serial Interface - transmit a data 41H serially by interfacing 8086 with 8251
(b) Parallel Interface

SERIAL INTERFACE

Description:

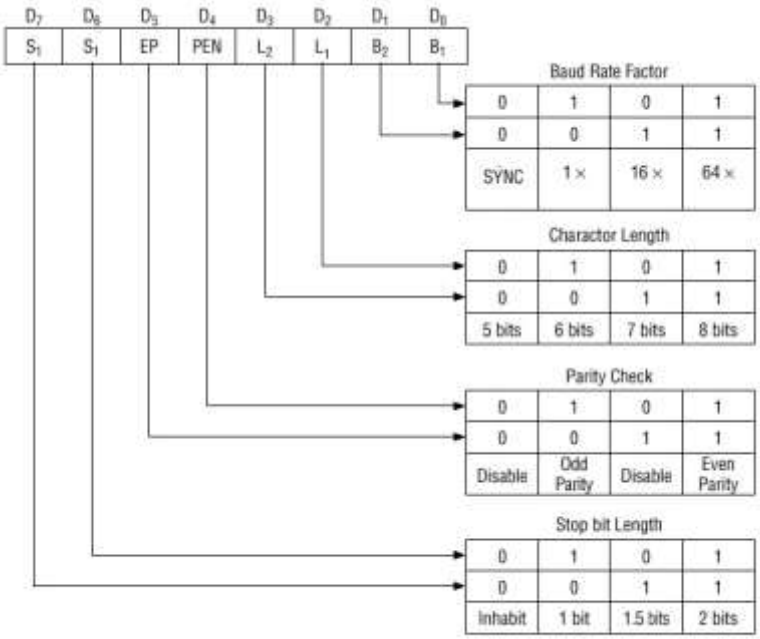
The 8253 and 8251 should be initialized before transmitting the character. The Program first initialize 8253 to give an output clock frequency of 150 KHz at channel 0 which will give a 9600 baud rate of 8251. The 8251 mode instruction (refer mode instruction format) is initialized with the following specifications: 8bit data, No parity, Baud rate factor (16x), 1 stop bit. Thus the mode command word is 4E for the above said specifications. The 8251 command instruction(refer command instruction format) is initialized with 37H which enables the transmit enable and receive enable bits, force DTR output to zero, resets the error flags, and forces RTS output to zero.

Algorithm:

1. Start the program.
2. Set the origin as 1100H.
3. Initialize the 8253 Timer in Mode 3
4. Initialize the 8251
5. Transmit the data at transmitter end
6. Reset the system
7. At the receiver end receive the data and reset the system
8. Stop the program.

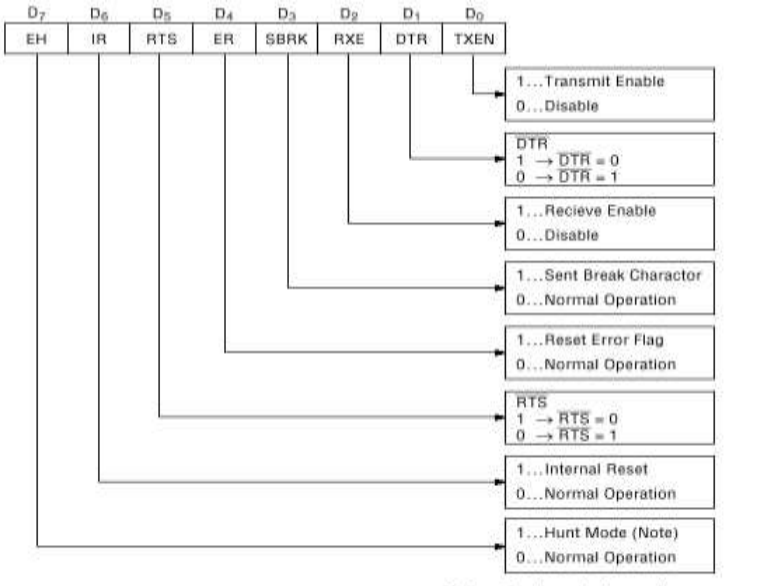
PROGRAM:

Label	Program	Comments
	ORG 1000H	Set starting address as 1000H.
	MOV AL, 36	Mode set for 8253 – Channel 0 in Mode 3
	OUT CE, AL	
	MOV AL, 10	
	OUT C8, AL	
	MOV AL, 00	
	OUT C8, AL	
	MOV AL, 4E	Mode instruction for 8251
	OUT C2, AL	
	MOV AL, 37	Command Instruction for 8251
	OUT C2, AL	
	MOV AL, 41	
	OUT C0, AL	Sent the data 41
	INT 2	Reset
	ORG 1200H	
	IN AL,C0	Receive the data 41
	MOV BX,1250	
	MOV [BX],AL	Store the data at 1250H
	INT 2	Reset



Bit Configuration of Mode Instruction (Asynchronous)

CONTROL WORD FORMAT OF 8255



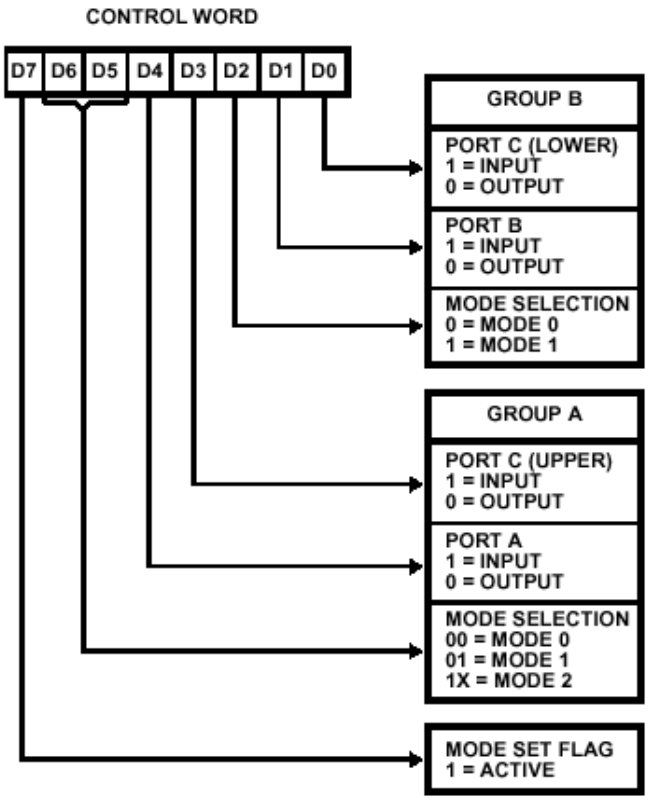
Note: Seach mode for synchronous charactors in synchronous mode.

Bit Configuration of Command

Observation:

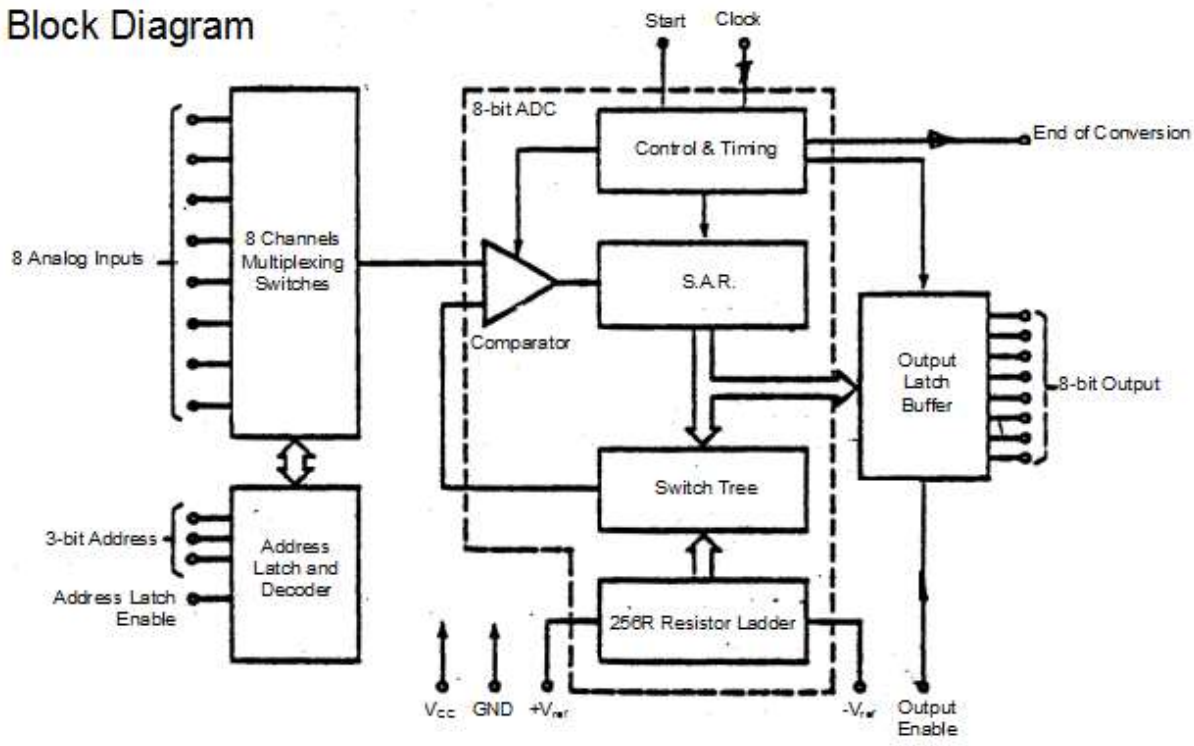
Output:

1250:



REVIEW QUSETIONS:

1. Expand USART?
2. Where do we prefer the serial communication?
3. What is the function of instruction pointer (IP) register?
4. What is the difference between IN and OUT instructions?
5. What is MODEM?



PARALLEL INTERFACE

Description:

Initialize the Port A as Input port and Port B as Output port in Mode – 0. The input port reads the data set by the SPDT switches and the output port outputs the same data to port B to glow LEDs accordingly.

Algorithm:

1. Start the program.
2. Set the origin as 1100H.
3. Initialize the port A as input port
4. Initialize the port B as output port
5. Configure 8255 in mode 0
6. Read the input port
7. Write the read data to the output port
8. Stop the program.

Parallel Interface Program

Label	Program	Comments
	ORG 1100H	Set starting address as 1100H.
	MOV AL,90	Initialize 8255 in mode 0 with port A as
	OUT C6,AL	input port and port B as output port.
	IN AL,C0	Read the data from SPDT switch
	OUT C2,AL	Write the data to LEDs
	HLT	

Example:

Input:

SPDT switch position: 10110011

Output:

LED status: 10110011

Observation:

Input:

SPDT switch position:

Output:

LED status:

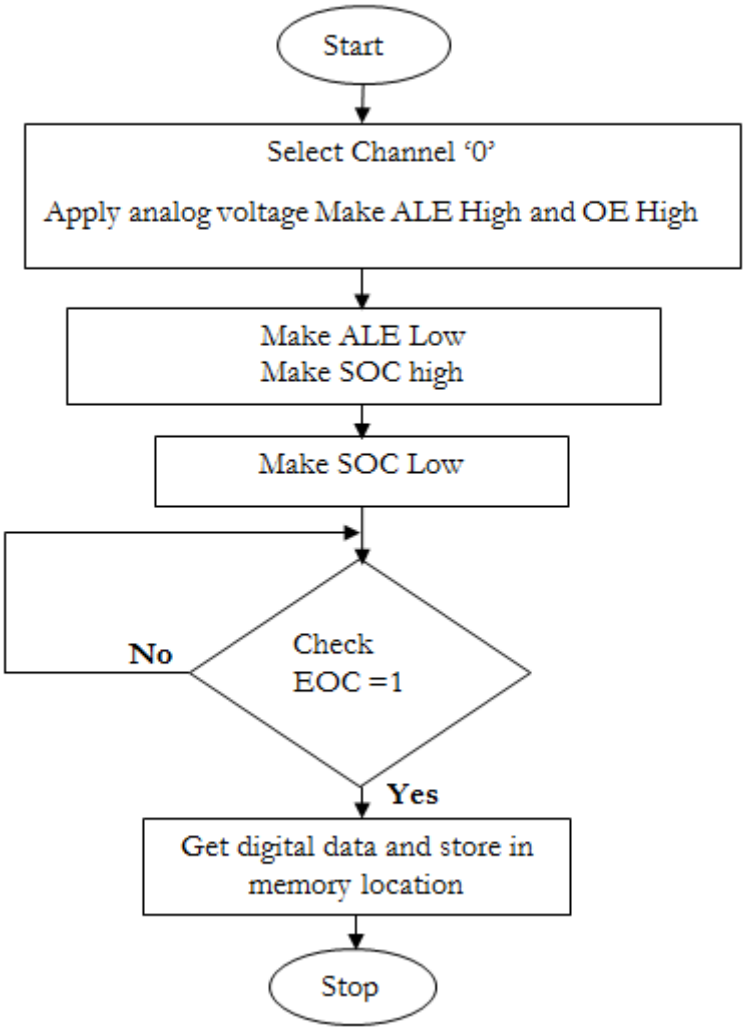
REVIEW QUSETIONS:

1. What is the difference between min mode and max mode of 8086?
2. What is the difference between near and far procedure?
3. What is difference between shifts and rotate instructions?
4. Which are strings related instructions?
5. Which are addressing modes and their examples in 8086?
6. Discuss the use of following instructions:
    a. SCASB
    b. LAHF
    c. ROL
    d. SHR
    e. IDIV
7. List out the internal devices of 8255.
8. Define – USART
9. What is scanning in keyboard and what is scan time?
10. What is programmable peripheral device?
11. What are the tasks involved in keyboard interface?
12. How a keyboard matrix is formed in keyboard interface using 8279?
13. Define – GPIB?
14. Advantages of differential data transfer?
15. What are the modes used in keyboard display interface?

Result:

Thus the programs for serial and parallel interface are executed successfully.

FLOWHCART


Start
Select Channel '0'
Apply analog voltage Make ALE High and OE High
Make ALE Low
Make SOC high
Make SOC Low
Check
EOC =1
No
Yes
Get digital data and store in memory location
Stop

Ex. No. 13
Date:

A/D AND D/A INTERFACE

Aim:

To write an assembly language program to demonstrate
(a) Analog to Digital Conversion
(b) Digital to Analog Conversion

ANALOG TO DIGITAL CONVERSION

Features of ADC 0809

ADC 0809 is a monolithic CMOS device, with an 8-bit analog to digital converter, 8 channel multiplexer and microprocessor compatible control logic

1. 8 bit resolution
2. 100 μs Conversion time
3. 8 channel multiplexer with latched control logic
4. No need for external zero or full scale adjustments
5. Low power consumption time
6. Latched tristate output

The device contains an 8 channel single ended analog signal multiplexer. A particular input channel. A particular input channel is selected by using the address decoding. Table shows the input states for the address lines to select any channel. The address is latched into the decoder of the chip on low to high transition of the address latch enable. The A/D converter‘s successive approximation register reset on the positive edge of the start of the conversion pulse. The conversion is begun on the falling edge of the SOC pulse. End of conversion will go low between 0 and 8 clock pulses after the rising edge of start of conversion

SELECTED ANALOG CHANNEL	ADDRESS LINE		
	ADD C	ADD B	ADD A
IN0	0	0	0
IN1	0	0	1
IN2	0	1	0
IN3	0	1	1
IN4	1	0	0
IN5	1	0	1
IN6	1	1	0
IN7	1	1	1

Algorithm

1. Select Channel ‘0’ and apply analog voltage
2. Send Start of conversion
3. Check End of conversion
4. Get digital data for corresponding analog voltage and display at stored location.

The buffer 74LS244 which transfers the converted data outputs to data bus is selected when

A7	A6	A5	A4	A3	A2	A1	A0
1	1	0	0	0	X	X	X

=C0H

The I/O address for the latch 74LS 714 which latches the data bus to ADD A, ADD B and ADDC and ALE 1 and ALE 2 is

A7	A6	A5	A4	A3	A2	A1	A0
1	1	0	0	1	X	X	X

=C8H

The flip flop 74LS74 which transfers the D0 line status to the start of conversion pin of ADC0809 is selected when

A7	A6	A5	A4	A3	A2	A1	A0
1	1	0	1	0	X	X	X

=D0H

The EOC output of ADC 1 and ADC 2 is transferred to D0 line by means of two tristate buffers.

The EOC 1 is selected when

A7	A6	A5	A4	A3	A2	A1	A0
1	1	0	1	1	X	X	X

=D8H

The EOC 2 is selected when

A7	A6	A5	A4	A3	A2	A1	A0
1	1	1	0	0	X	X	X

=E0H

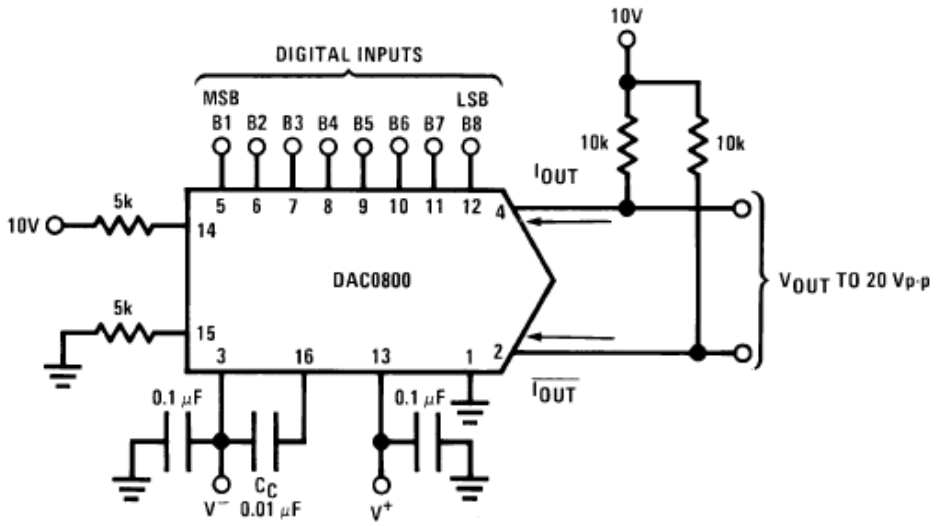
SL. NO	CHANNEL NUMBER	EOC ADDRESS	CHNO. ALE LOW OE HIGH	CHNO. ALE HIGH OE LOW	CHNO. ALE LOW OE HIGH
1	CH0	D8	10	18	10
2	CH1	D8	11	19	11
3	CH2	D8	12	1A	12
4	CH3	D8	13	1B	13
5	CH4	D8	14	1C	14
6	CH5	D8	15	1D	15
7	CH6	D8	16	1E	16
8	CH7	D8	17	1F	17

PROGRAM

Label	Program	Comments
	ORG 4100H	Set starting address as 4100H.
	MOV AL, 10H	Selection Channel '0'
	OUT 0C8H, AL	
	MOV AL, 18H	Make ALE1 and OE1 high
	OUT 0C8H, AL	
	MOV AL, 01H	Make SOC High
	OUT 0DOH, AL	
	MOV AL, 00H	
	MOV AL, 00H	
	MOV AL, 00H	
	MOV AL, 00H	Make SOC low
	OUT 0DOH, AL	
LOOP	IN AL, 0D8H	Check EOC
	AND AL, 01H	
	CMP AL, 01H	
	JNZ LOOP	
	IN AL, 0C0	Output Digital Data
	MOV BX, 1200H	
	MOV [BX], AL	
	HLT	

Observation:

DAC 0800



REVIEW QUSETIONS:

1. Which is by default pointer for CS/ES?
2. How many segments present in it?
3. What is the size of each segment?
4. Basic difference between 8085 and 8086?
5. Which operations are not available in 8085?
6. What is the difference between Macro and procedure?
7. Which is by default pointer for CS/ES?
8. Basic difference between 8085 and 8086?
9. Which operations are not available in 8085?
10. What is the difference between instructions RET & IRET?
11. What are the functions performed by 8279?
12. What is PPI?
13. Give the control word format for I/O mode of 8255?
14. Give the BSR mode format of 8255.

INTERFACING DAC WITH 8086

THEORY:

DAC 0800 is an 8 – bit DAC and the output voltage variation is between – 5V and + 5V.The output voltage varies in steps of 10/256 = 0.04 (appx.). The digital data input and the corresponding output voltages are presented in the Table1.

Input Data in HEX	Output Voltage
00	- 5.00
01	- 4.96
02	- 4.92
…	…
7F	0.00
…	…
FD	4.92
FE	4.96
FF	5.00

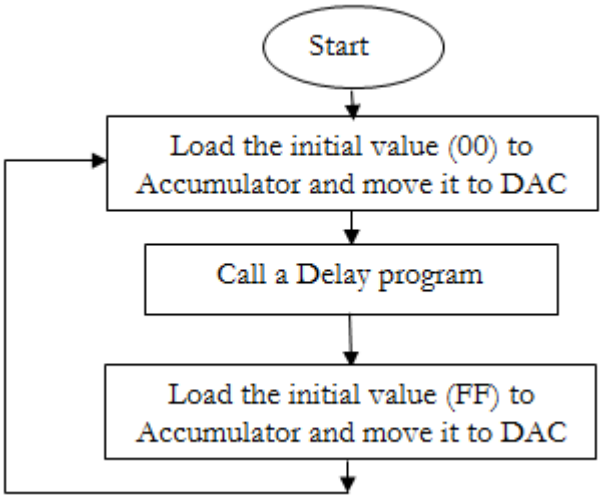
Referring to Table1, with 00 H as input to DAC, the analog output is – 5V. Similarly, with FF H as input, the output is +5V. Outputting digital data 00 and FF at regular intervals, to DAC, results in different wave forms namely square, triangular, etc,. The port address of DAC is 08 H

ALGORITHM:

(a) Square Wave Generation

1. Load the initial value (00) to Accumulator and move it to DAC
2. Call the delay program
3. Load the final value(FF) to accumulator and move it to DAC
4. Call the delay program.
5. Repeat Steps 2 to 5

FLOWCHART



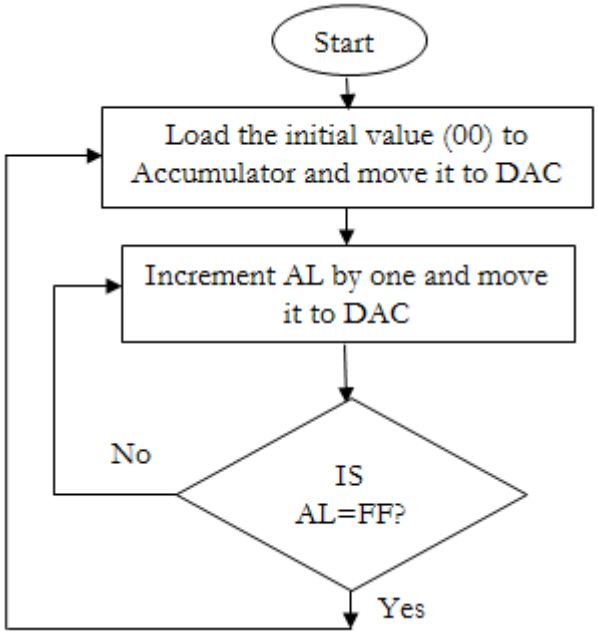
PROGRAM

Label	Program	Comments
	ORG 4100H	
START:	MOV AL, 00H	
	OUT 0C0H,AL	
	CALL DELAY	Set starting address as 4100H.
	MOV AL, 0FFH	
	OUT 0C0H,AL	
	CALL DELAY	
	JMP START	
DELAY:	MOV CX, 05FFH	
L1:	LOOP L1	
	RET	

(b) Saw tooth Wave Generation

1. Load the initial value (00) to Accumulator
2. Move the accumulator content to DAC
3. Increment the accumulator content by 1.
4. Repeat Steps 3 and 4.

FLOWCHART



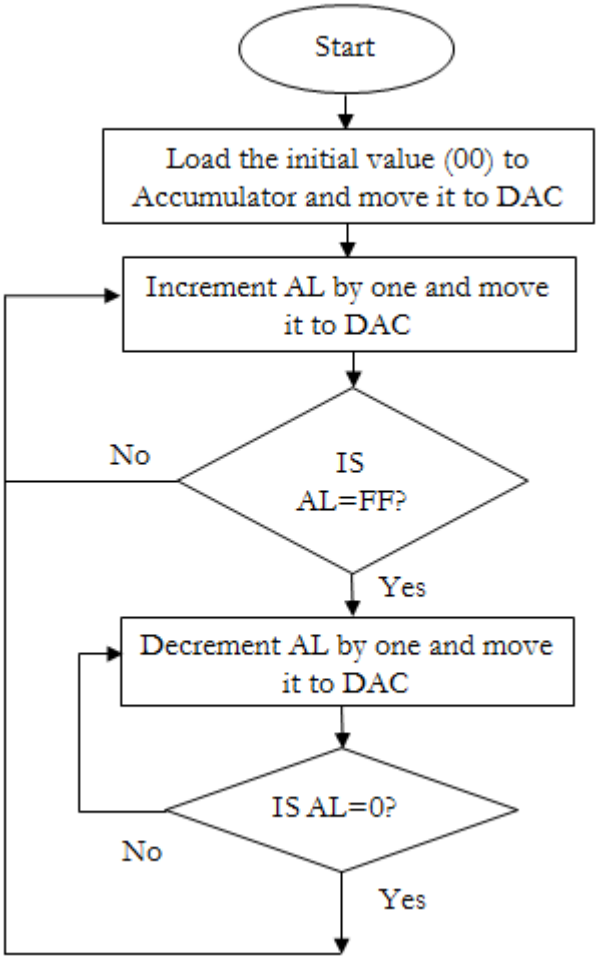
PROGRAM

Label	Program	Comments
	ORG 4100H	Set starting address as 4100H.
START	MOV AL, 00H	
L1	OUT 0C0H, AL	
	INC AL	
	JNZ L1	
	JMP START	

(c) Triangular Wave Generation

1. Load the initial value (00) to Accumulator
2. Move the accumulator content to DAC
3. Increment the accumulator content by 1.
4. If accumulator content is zero proceed to next step. Else go to step 3.
5. Load value (FF) to Accumulator
6. Move the accumulator content to DAC
7. Decrement the accumulator content by 1.
8. If accumulator content is zero go to step2. Else go to step 7.

FLOWCHART



PROGRAM

Label	Program	Comments
	ORG 4100H	Set starting address as 4100H.
START:	MOV BL, 00H	
L1:	MOV AL, BL	
	OUT 0C0H,AL	
	INC BL	
	JNZ L1	
	MOV BL, 0FFH	
L2:	MOV AL, BL	
	OUT 0C0H,AL	
	DEC BL	
	JNZ L2	
	JMP START	

Example:

Waveform	Amplitude	Time Period(ms)
Square	2	56
Sawtooth	2	3
Triangular	2	2.4

Observation:

Waveform	Amplitude	Time Period(ms)
Square		
Sawtooth		
Triangular		

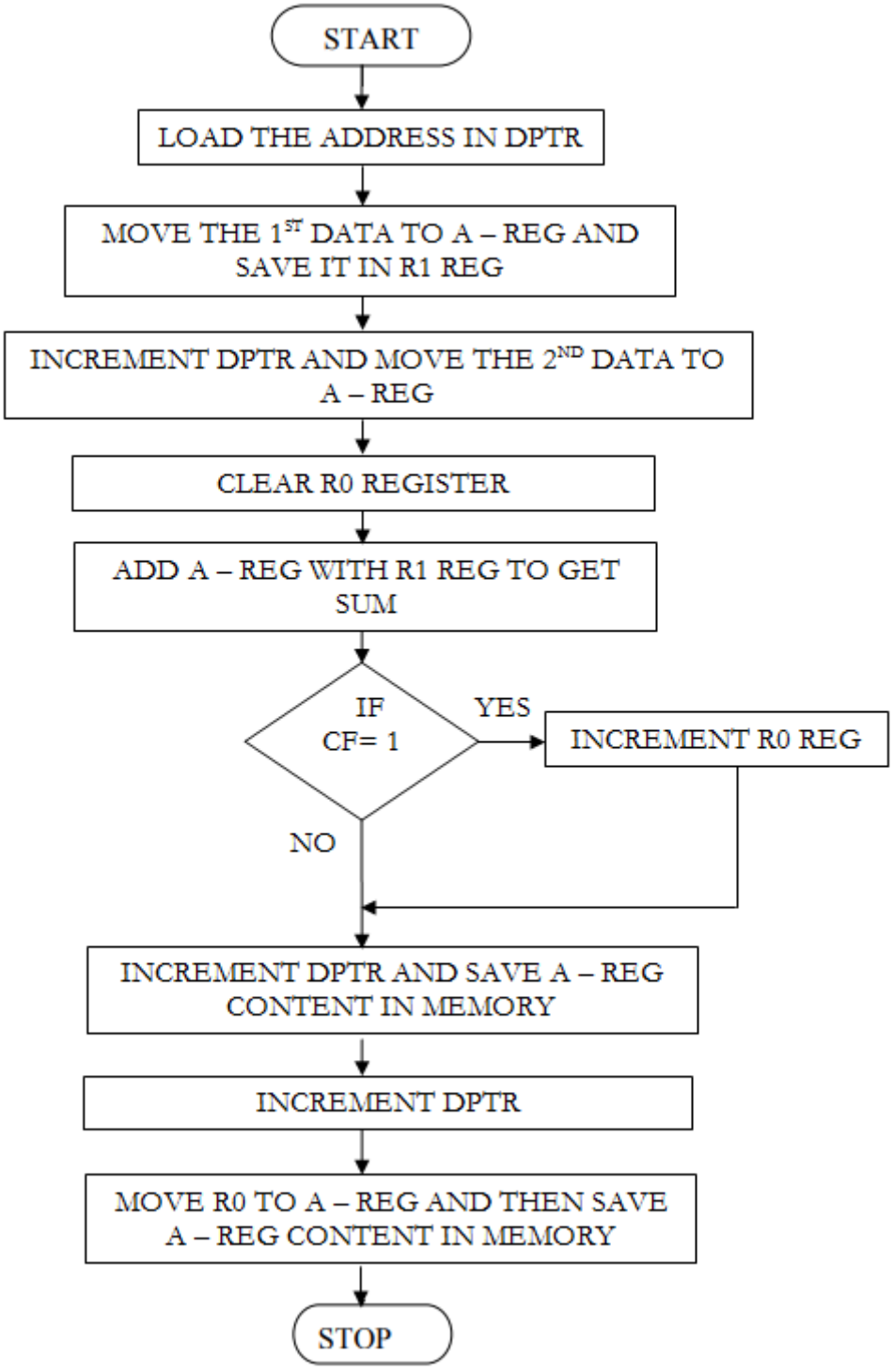
REVIEW QUSETIONS:

1. Whether 8086 is compatible with Pentium processor?
2. Write an ALP program for multiplication of given number in location mode a) 0060, b) 0002
3. What is 8087? How it is different from 8086?
4. Write an ALP program for addition of multi byte numbers.
5. What is the size of flag register?
6. List the operating modes of 8253 timer.
7. Give the control word format of timer.
8. What is the use of USART?
9. Compare the serial and parallel communications.
10. What is the use of Keyboard and display controller?
11. What is meant by synchronous data transfer scheme?
12. Define – Interrupt I/O?
13. Why interfacing is needed for I/O devices?
14. When the 8085 processor checks for an interrupt?
15. How the 8085 processor differentiates a memory access and I/O access?

RESULT

Thus the program to demonstrate the ADC and DAC were executed.

Flow Chart


START
LOAD THE ADDRESS IN DPTR
MOVE THE 1ST DATA TO A – REG AND SAVE IT IN R1 REG
INCREMENT DPTR AND MOVE THE 2ND DATA TO A – REG
CLEAR R0 REGISTER
ADD A – REG WITH R1 REG TO GET SUM
IF CF= 1
YES
INCREMENT R0 REG
NO
INCREMENT DPTR AND SAVE A – REG CONTENT IN MEMORY
INCREMENT DPTR
MOVE R0 TO A – REG AND THEN SAVE A – REG CONTENT IN MEMORY
STOP

Ex. No. 14
Date:

BASIC ARITHMETIC AND LOGIC OPERATIONS

Objective:

To write an ALP to perform the following operations using 8051 instruction set

(a) Addition
(b) Subtraction
(c) Multiplication
(d) Division
(e) Logical operation

ADDITION OF TWO 8 BIT NUMBERS

Description:

In order to perform addition in 8051, one of the data should be in accumulator and another data can be in any SFR/internal RAM or can be an immediate data. After addition the sum is stored in accumulator. The sum of two 8 – bit data can be either 8 bits (sum only) or 9 bits (sum and carry). The accumulator can accommodate only the sum and if there is carry, the 8051 will indicate by setting carry flag. Hence one of the internal register/RAM locations can be used to account for carry.

Algorithm:

1. Set DPTR as pointer for data.
2. Move first data from external memory to accumulator and save it in R1 register.
3. Increment DPTR.
4. Move second data from external memory to accumulator
5. Clear R0 register to account for carry.
6. Add the content of R1 register to accumulator.
7. Check for carry. If carry is not set go to step 8. Otherwise go to next step.
8. Increment R0 register.
9. Increment DPTR and save the sum in external memory.
10. Increment DPTR, move carry to accumulator and save it in external memory.
11. Stop

Program

Label	Program	Comments
	MOV DPTR,#4500	Load address of 1st data in DPTR
	MOVX A,@DPTR	Move the 1st data to A
	MOV R1,A	Save the first data in R1
	INC DPTR	Increment DPTR to point 2nd data
	MOVX A,@DPTR	Load the 2nd data in A
	MOV R0,#00	Clear R0 for the account of carry
	ADD A,R1	Get the sum in A reg
	JNC AHEAD	Check carry flag
	INC R0	If carry is set increment R0
AHEAD:	INC DPTR	Increment DPTR
	MOVX @DPTR,A	Save the sum in external memory
	INC DPTR	Increment DPTR
	MOV A,R0	Move carry to A reg
	MOVX @DPTR,A	Save the carry in external memory
HERE:	SJMP HERE	Remain idle in infinite loop

Example:

Input:

4500: 05 [Addend]
4501: 06 [Augend]

Output:

4502: 0B [Sum]
4503:00 [Carry]

Observation:

Input:

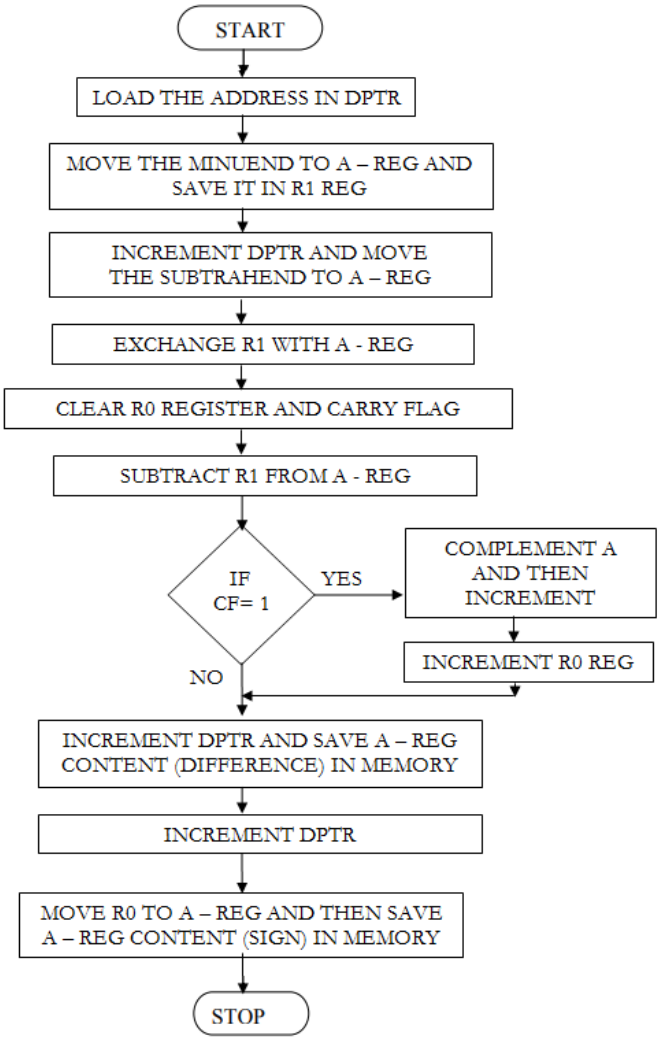
4500: [Addend]
4501: [Augend]

Output:

4502: [Sum]
4503: [Carry]

Manual Calculation:

Flow Chart



SUBTRACTION OF TWO 8 BIT NUMBERS

Description:

In order to perform subtraction in 8051, one of the data should be in accumulator and another data can be in any SFR/internal RAM or can be an immediate data. After subtraction the result is stored in accumulator. The 8051 perform 2's complement subtraction and then complement the carry. Therefore if the result is negative carry flag is set and the accumulator will have 2's complement of the result. In order to get the magnitude of the result again take 2's complement of the result. One of the register is used to account for the sign of the result. The 8051 will consider previous carry while performing subtraction and so the carry should be cleared before performing subtraction.

Algorithm:

1. Set DPTR as pointer for data.
2. Move the minuend from external memory to accumulator and save it in R1 register.
3. Increment DPTR.
4. Move subtrahend from external memory to accumulator
5. Exchange the contents of R1 and A such that minuend is in A and subtrahend is in R1
6. Clear R0 register to account for sign.
7. Clear carry flag.
8. Subtract the content of R1 register from accumulator.
9. Check for carry. If carry is not set go to step 12. Otherwise go to next step.
10. Complement the content of A – reg and increment by 1 to get 2's complement of result in A – reg
11. Increment R0 register.
12. Increment DPTR and save the result in external memory.
13. Increment DPTR, move R0 (sign bit) to accumulator and then save it in external memory.
14. Stop

Program

Label	Program	Comments
	MOV DPTR,#4500	Load address of minuend in DPTR
	MOVX A,@DPTR	Move the minuend to A
	MOV R1,A	Save the minuend in R1
	INC DPTR	Increment DPTR to point subtrahend
	MOVX A,@DPTR	Load the subtrahend in A
	XCH A,R1	Get minuend in A and Subtrahend in R1
	MOV R0,#00	Clear R0 for the account of Sign
	CLR C	Clear carry
	SUBB A,R1	Subtract R1 from A
	JNC AHEAD	Check Carry flag. If carry is set then
	CPL A	Get 2's complement of result in A
	INC A	
	INC R0	Set R0 to indicate negative sign
AHEAD:	INC DPTR	Increment DPTR
	MOVX @DPTR,A	Save the result in external memory
	INC DPTR	Increment DPTR
	MOV A,R0	Move sign bit to A reg
	MOVX @DPTR,A	Save the sign in external memory
HERE:	SJMP HERE	Remain idle in infinite loop

Example:

Input:

4500: 0A	[Minuend]
4501:05	[Subtrahend]

Output:

4502:05	[Difference]
4503:00	[Sign Bit]

Observation:

Input:

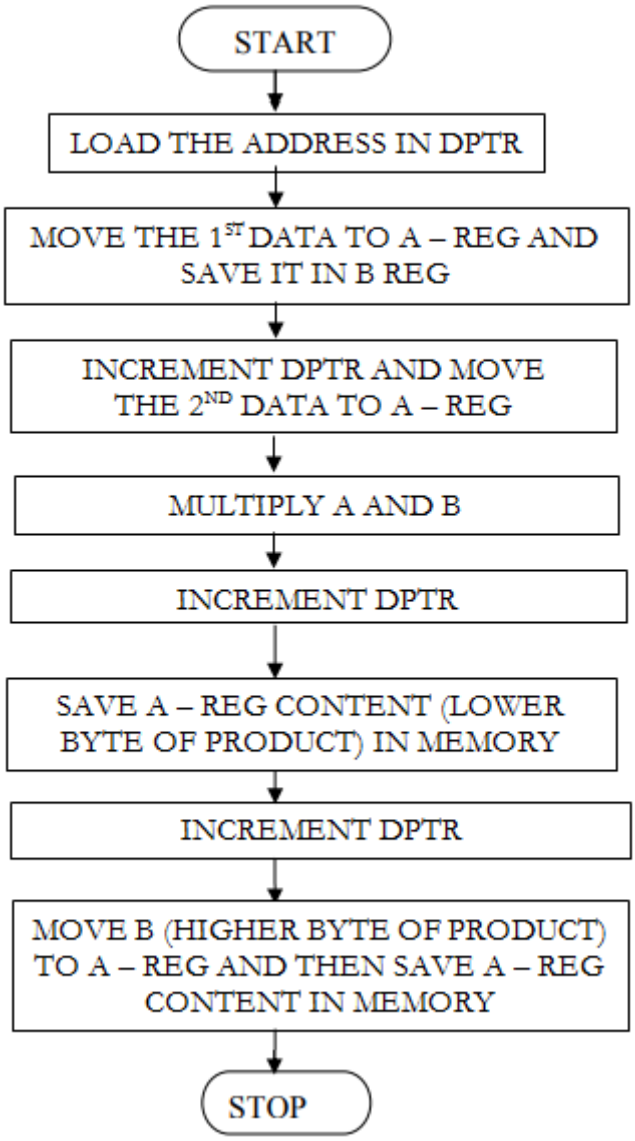
4500:	[Minuend]
4501:	[Subtrahend]

Output:

4502:	[Difference]
4503:	[Sign Bit]

Manual Calculation:

Flow Chart


START
LOAD THE ADDRESS IN DPTR
MOVE THE 1ST DATA TO A – REG AND SAVE IT IN B REG
INCREMENT DPTR AND MOVE THE 2ND DATA TO A – REG
MULTIPLY A AND B
INCREMENT DPTR
SAVE A – REG CONTENT (LOWER BYTE OF PRODUCT) IN MEMORY
INCREMENT DPTR
MOVE B (HIGHER BYTE OF PRODUCT) TO A – REG AND THEN SAVE A – REG CONTENT IN MEMORY
STOP

MULTIPLICATION OF TWO 8 BIT NUMBERS

Objective:

To write an ALP to multiply two numbers of 8-bit data using 8051 instruction set

Description:

In order to perform subtraction in 8051, the two 8 – bit data should be stored in A and B registers, then multiplication can be performed by using "MUL AB" instruction. After multiplication the 16 – bit product will be in A and B register such that lower byte in A and higher byte in B register.

Algorithm:

1. Load address of data in DPTR
2. Move the first data from external memory to A and save in B.
3. Increment DPTR and move second data from external memory to B.
4. Perform multiplication to get the product in A and B.
5. Increment DPTR and save A (lower byte of product) in memory
6. Increment DPTR , move B (lower byte of product) to A and save it in memory
7. Stop

Label	Program	Comments
	MOV DPTR,#4500	Load address of 1st data in DPTR
	MOVX A,@DPTR	Move the 1st data to A
	MOV B,A	Save the 1st data in B
	INC DPTR	Increment DPTR to point 2nd data
	MOVX A,@DPTR	Load the 2nd data in A
	MUL AB	Get the product in A and B
	INC DPTR	Increment DPTR
	MOVX @DPTR,A	Save the lower byte of result in external memory
	INC DPTR	Increment DPTR
	MOV A,B	Move the higher byte of product to A reg
	MOVX @DPTR,A	Save it in external memory
HERE:	SJMP HERE	Remain idle in infinite loop

Example:

Input:

4500:02	[1st data]
4501:03	[2nd data]

Output:

4502:06	[Lower byte of product]
4503:00	[Higher byte of product]

Observation:

Input:

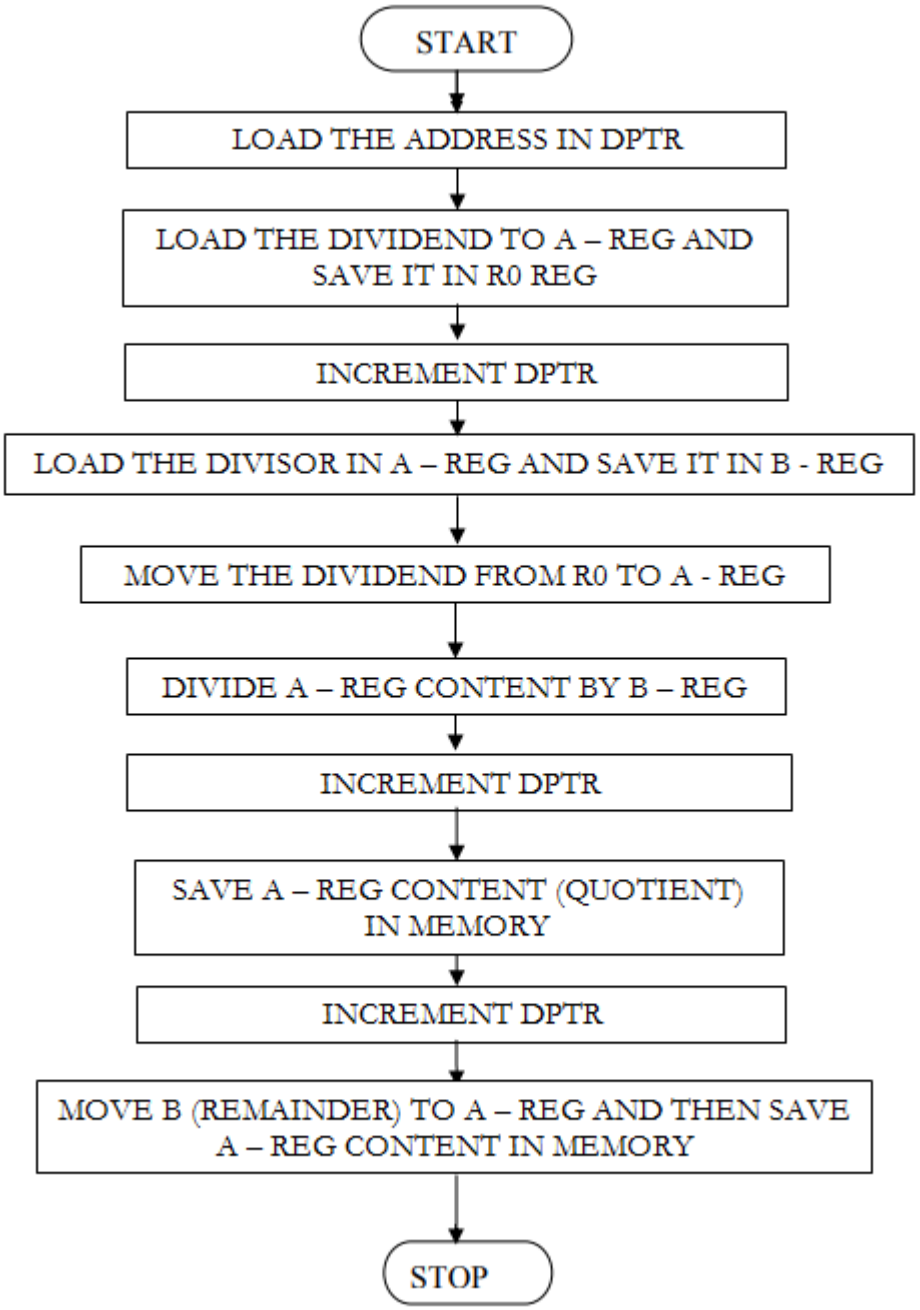
4500:	[1st data]
4501:	[2nd data]

Output:

4502:	[Lower byte of product]
4503:	[Higher byte of product]

Manual Calculation:

FLOWCHART


START
LOAD THE ADDRESS IN DPTR
LOAD THE DIVIDEND TO A – REG AND SAVE IT IN R0 REG
INCREMENT DPTR
LOAD THE DIVISOR IN A – REG AND SAVE IT IN B - REG
MOVE THE DIVIDEND FROM R0 TO A - REG
DIVIDE A – REG CONTENT BY B – REG
INCREMENT DPTR
SAVE A – REG CONTENT (QUOTIENT) IN MEMORY
INCREMENT DPTR
MOVE B (REMAINDER) TO A – REG AND THEN SAVE A – REG CONTENT IN MEMORY
STOP

DIVISION OF TWO 8 BIT NUMBERS

Description:

In order to perform subtraction in 8051, the dividend should be stored in A – reg and divisor should be stored in B – reg. then the content of A can be divided by B using the instruction "DIV AB". After division the quotient will be in A – reg and remainder will be in B – reg.

Algorithm:

1. Load address of data in DPTR
2. Move the dividend from external memory to A and save it in R0 register.
3. Increment DPTR and move the divisor from external memory to A and save it in B reg.
4. Move the dividend from R0 to A.
5. Perform division to get quotient in A and remainder in B.
6. Increment DPTR and save quotient (content of A - reg) in memory
7. Increment DPTR.
8. Move the remainder (Content of B – reg) to A and save in memory.
9. Stop

Label	Program	Comments
	MOV DPTR,#4500	Load address of dividend in DPTR
	MOVX A,@DPTR	Move the dividend to A
	MOV R0,A	Save the dividend in R0
	INC DPTR	Increment DPTR to point divisor
	MOVX A,@DPTR	Load the divisor in A
	MOV B,A	Move the divisor to B
	MOV A,R0	Move the dividend to A
	DIV AB	Divide the content of A by B
	INC DPTR	Increment DPTR
	MOVX @DPTR,A	Save the quotient in external memory
	INC DPTR	Increment DPTR
	MOV A,B	Move the remainder to A reg
	MOVX @DPTR,A	Save it in external memory
HERE:	SJMP HERE	Remain idle in infinite loop

Example:

Input:

4500: 04 [Dividend]
4501:02 [Divisor]

Output:

4502:02 [Quotient]
4503:00 [Remainder]

Observation:

Input:

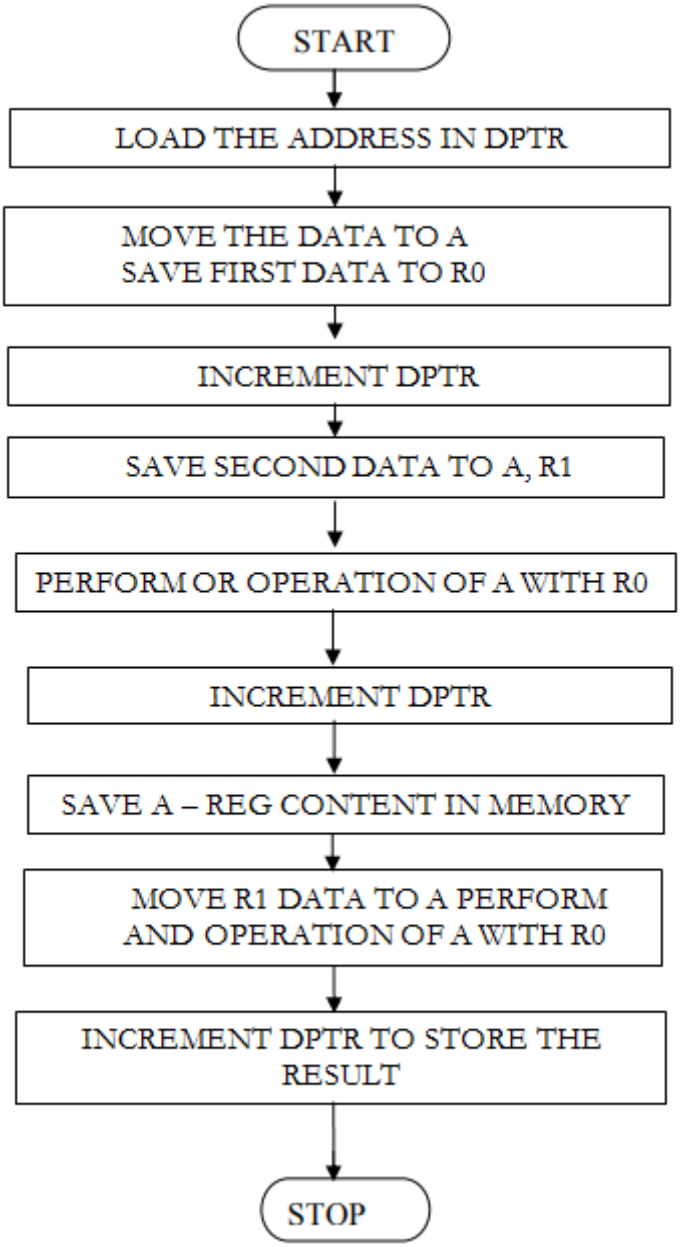
4500: [Dividend]
4501: [Divisor]

Output:

4502: [Quotient]
4503: [Remainder]

Manual Calculation:

FLOWCHART


START
LOAD THE ADDRESS IN DPTR
MOVE THE DATA TO A
SAVE FIRST DATA TO R0
INCREMENT DPTR
SAVE SECOND DATA TO A, R1
PERFORM OR OPERATION OF A WITH R0
INCREMENT DPTR
SAVE A – REG CONTENT IN MEMORY
MOVE R1 DATA TO A PERFORM
AND OPERATION OF A WITH R0
INCREMENT DPTR TO STORE THE
RESULT
STOP

LOGICAL OPERATIONS OF 8 BIT NUMBERS

Description:

The first value should be stored in R0 -reg, second value should be stored in R1 – reg, First move R1 value to A, perform OR operation with R0 reg and store the result. Second move R1 value to A performs AND operation with R0 reg stores the result.

Algorithm:

1. Load address of first data in DPTR
2. Move the data to A
3. Save first data to R0
4. Increment DPTR to Load address of second data in DPTR
5. Save second data to A, R1
6. Perform OR operation of A with R0
7. Increment DPTR to store the result
8. Move R1 data to A
9. Perform AND operation of A with R0
10. Increment DPTR to store the result

Label	Program	Comments
	MOV DPTR,#4500	Load address of first data in DPTR
	MOVX A,@DPTR	Move the data to A
	MOV R0, A	Save first data to R0
	INC DPTR	Increment DPTR to Load address of second data in DPTR
	MOVX A,@DPTR	
	MOV R1,A	Save second data to A, R1
	ORL A, R0	Perform OR operation
	INC DPTR	Increment DPTR to store the result
	MOVX @DPTR, A	
	MOV A, R1	
	ANL A, R0	Perform AND operation
	INC DPTR	Increment DPTR to store the result
	MOVX @DPTR, A	
HERE:	SJMP HERE	

Example:

Input

4500 :00

4501:01

Output

4502 :01 (OR operation)

4503 :00 (AND operation)

Observation

Input

4500 :

4501:

Output

4502 : (OR operation)

4503 : (AND operation)

Label	Program	Comments
	ORG 4100H	Set starting address as 4100H.
	MOV DPTR, #4500H	Initialise the dptr
	MOVX A,@DPTR	Get the data in A – reg
	MOV B,A	Copy it in B – reg
	MUL AB	Multiply A and B
	INC DPTR	Increment dptr
	MOVX @DPTR,A	Store the lower order in memory
	INC DPTR	Increment dptr
	MOV A,B	
	MOVX @DPTR,A	Store the higher order in memory
HERE:	SJMP HERE	

Example:

Input:

4500:03

Output:

4501:09
4502:00

Observation:

Input:

4500:

Output:

4501:
4502

REVIEW QUSETIONS:

1. What is a microcontroller? How does it differ from a microprocessor?
2. List out the features of 8051.
3. Draw the PIN diagram of the 8051 microcontroller.
4. What is the role of the program counter in 8051?
5. Write the significance of oscillators in a microcontroller.
6. What are the types of memory in 8051?
7. What are special function register?
8. List some of the data transfer instructions.
9. List some of the arithmetic instructions.
10. Explain the instructions RLA, RRC.
11. Write the addressing modes of 8051.
12. What is PSW?
13. Draw the format of TMOD register.
14. Explain the 16 bit registers DPTR and SP.
15. What is the importance of SFRs available in 8051?

Result:

Thus the program for arithmetic and logic operation was written and executed.

Ex. No. 15
Date:

SQUARE, CUBE and 2'S COMPLIMENT OF A NUMBER

Objective:

To write 8051 ALP to determine the square, cube and 2's compliment of a number

SQUARE OF A NUMBER

Description:

The square of a number is determined by multiplying the value by itself. In this program the input is obtained in A – reg and then it is copied to B – reg. The values of A and B registers are multiplied and the result is stored in memory.

Algorithm:

1. Start the program.
2. Set the origin as 4500H.
3. Initialize DPTR
4. Get the value in A – reg and copy it in B – reg
5. Multiply the values of A – reg and B – reg
6. Store the result
7. Stop the program.

PROGRAM

Label	Program	Comments
	ORG 4100H	Set starting address as 4100H.
	MOV DPTR,#4500H	Initialise the dptr
	MOVX A,@DPTR	Get the data in A – reg
	MOV R0,A	Copy it in r0 – reg
	MOV B,A	Copy it in B – reg
	MUL AB	Multiply A and B
	PUSH B	Push higher order to stack
	MOV B,A	
	MOV A,R0	
	MUL AB	
	INC DPTR	
	MOVX @DPTR,A	Store the lower order of result
	MOV R2,B	
	POP B	
	MOV A,R0	
	MUL AB	
	ADD A,R2	
	INC DPTR	
	MOVX @DPTR,A	
	MOV A,B	
	INC DPTR	
	MOVX @DPTR,A	Store the higher order of the result
HERE	SJMP HERE	

CUBE OF A NUMBER

Description:

The square of a number is determined by multiplying the value by itself for two times. In this program the input is obtained in A – reg and then it is copied to B – reg and r0 - reg. The values are multiplied and stored tin the memory.

Algorithm:

1. Start the program.
2. Set the origin as 4100H.
3. Initialize DPTR
4. Copy the data to A – reg, B- eg , R0 – reg
5. Multiply the data to find the cube
6. Store the result
7. Stop the program

Label	Program	Comments
	MOV DPTR,#4500	Load address of data in DPTR
	MOVX A,@DPTR	Move the data to A
	CPL A	Complement A
	INC A	Increment A by 1.
	INC DPTR	Increment DPTR to store the result of 2's
	MOVX @DPTR, A	complement of A
HERE:	SJMP HERE	

Example:

Input:

4500:03
Output:
4501:27
4502:00

Example:
Input
4500 :01
Output
4501 :F2 (Two's complement)
Observation:
Input
4500 :
Output
4501 : (Two's complement)

2'S COMPLIMENT OF A NUMBER

Description:

In order to perform 2's complement in 8051, the given value should be stored in A – reg then take one's complement of A and add value one to LSB.

Algorithm:

1. Load address of data in DPTR
2. Move the data to A
3. Complement A
4. Increment A by 1.
5. Increment DPTR to store the result of 2's complement of A
6. Stop

REVIEW QUESTIONS:

1. Explain the instruction MOV DPTR, #4500H.
2. What does the PUSH instruction do?
3. What instruction is used to multiply any two numbers?
4. What is the function of POP instruction?
5. Which instruction is used to increment the value?
6. What does the ORL instruction do?
7. Explain ANL R1,#0F**.**
8. How do we take two's complement of number? Give example.
9. What does the ORG 4100H mean?
10. Explain the mode 0 operating mode of 8051 serial ports.
11. Explain the mode 2 operating mode of 8051 serial ports.
12. Explain the mode 3 operating mode of 8051 serial ports.
13. What are the pins used for serial communication?
14. What is the use of SBUF register?
15. What are the methods to double the baud rate?

Result:

Thus the program to determine square, cube and 2's compliment of a number are executed successfully.

Label	Program	Comments
	ORG 4100H	
	MOV A,#7	Input 07
	ANL A,#0F	
	ORL A,#30	Get the equivalent ASCII
	MOV DPTR,#4500	
	MOVX @DPTR,A	
	MOV R1,#4	
	ANL R1,#0F	Input 04
	ORL A,#30	Get the equivalent ASCII
	INC DPTR	
	MOVX @DPTR,A	
HERE:	L1:SJMP HERE	

Ex. No. 16
Date:

UNPACKED BCD TO ASCII

Objective:

To write an Assembly Language Program (ALP) to convert unpacked BCD to ASCII using 8051 instruction set.

Description:

The 2 –digit unpacked BCD data will be directly given to the A reg. The equivalent ASCII code is obtained by logically OR with 30. i.e., adding 30 to the BCD value will result in its ASCII value.

Algorithm:

1. Start the program.
2. Set the origin as 4100H.
3. Get the BCD data (units Digit)in A register
4. Logically AND A with 0F to mask upper nibble
5. Logically OR A with 30 to get ASCII value
6. Store the result
7. Get the BCD data (tens digit) in A register
8. Logically AND A with 0F to mask upper nibble
9. Logically OR A with 30 to get ASCII value
10. Store the result
11. Stop the program.

OBSERVATION:
Output:
4500: 37
4501:34

REVIEW QUESTIONS:

1. Mention any two applications that use ADC and DAC.
2. Write the format of IE register.
3. What is the function of ITX bits in the TCON register?
4. What are the special function registers that controls the serial communication of 8051?
5. What are the pins used for the serial communication in 8051?
6. Write down the two activation levels for the external interrupts in 8051.
7. Explain the level triggered input.
8. Draw the organization of Interrupt Priority register.
9. Draw the format of the Interrupt Enable Register.
10. Explain ISR in a microcontroller.
11. Write about the jump statement.
12. Explain DJNZ instructions of Intel 8051 microcontroller.
13. What instruction can be used to swap two numbers?
14. Specify the single instruction which clears the MSB of the B register of 8051 without affecting the remaining bits.
15. Write about CALL statement in 8051.

RESULT:

Thus the program to convert the unpacked BCD to ASCII has been executed successfully.

Ex. No. 17
Date:

Square wave generation using 8051

Objective:

To write an Assembly Language Program (ALP) to generate square waveform using 8051 instruction set.

Description:

Square waves of any frequency (limited by the controller specifications) can be generated using the 8051 timer. The technique is very simple. Write up a delay subroutine with delay equal to half the time period of the square wave. Make any port pin high and call the delay subroutine. After the delay subroutine is finished, make the corresponding port pin low and call the delay subroutine gain. After the subroutine is finished, repeat the cycle again. The result will be a square wave of the desired frequency at the selected port pin.

Steps:

1. Assume Duty Cycle 50%
2.Assume 12MHZ Clock is Connected to Micro-Controller
3.Use Timers
4.Check output in P3.2

Program for 1 KHz Square wave using 8051 timer

```
        ORG 0000H
        MOVTMOD, #01H
        UP: SETB P3.2
        LCALL DELAY
        CLR P3.2
        LCALL DELAY
        SJMP UP
DELAY: MOV TH0, #0FEH
        MOV TL0, #0CH
        CLR TF0
        SETB TR0
HERE:JNB TF0,HERE
        RET
        END
```

Result:

Thus the square waveform has been generated successfully.