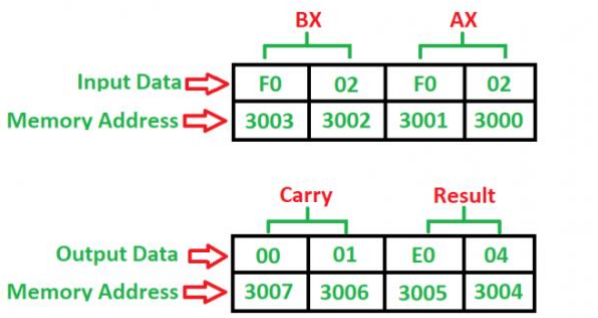
8086 program to add two 16-bit numbers with or without carry

1)Problem – Write a program to add two 16-bit numbers where starting address is **2000** and the numbers are at **3000** and **3002** memory address and store result into **3004** and **3006** memory address.



1. Load 0000H into CX register (for carry)
2. Load the data into AX(accumulator) from memory 3000
3. Load the data into BX register from memory 3002
4. Add BX with Accumulator AX
5. Jump if no carry
6. Increment CX by 1
7. Move data from AX(accumulator) to memory 3004
8. Move data from CX register to memory 3006
9. Stop

PROGRAM

Memory	Mnemonics	Operands	Comment
2000	MOV	CX, 0000	[CX] <- 0000
2003	MOV	AX, [3000]	[AX] <- [3000]
2007	MOV	BX, [3002]	[BX] <- [3002]
200B	ADD	AX, BX	[AX] <- [AX] + [BX]
200D	JNC	2010	Jump if no carry

Memory	Mnemonics	Operands	Comment
200F	INC	CX	[CX] <- [CX] + 1
2010	MOV	[3004], AX	[3004] <- [AX]
2014	MOV	[3006], CX	[3006] <- [CX]
2018	HLT		Stop

Explanation –

MOV is used to load and store data.

ADD is used to add two numbers where their one number is in accumulator or not.

JNC is a 2-bit command which is used to check whether the carry is generated from accumulator or not.

INC is used to increment an register by 1.

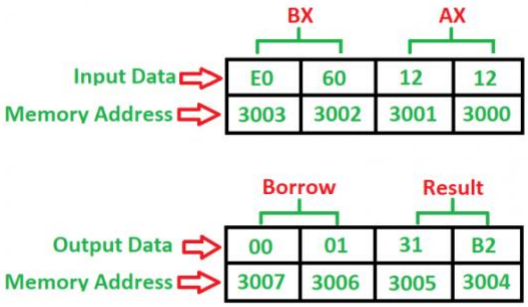
HLT is used to stop the program.

AX is an accumulator which is used to load and store the data.

BX, CX are general purpose registers where BX is used for storing second number and CX is used to store carry.

8086 program to subtract two 16-bit numbers with or without borrow

2) **Problem –** Write a program to subtract two 16-bit numbers where starting address is **2000** and the numbers are at **3000** and **3002** memory address and store result into **3004** and **3006** memory address.



Algorithm

1. Load 0000H into CX register (for borrow)
2. Load the data into AX(accumulator) from memory 3000
3. Load the data into BX register from memory 3002
4. Subtract BX with Accumulator AX
5. Jump if no borrow
6. Increment CX by 1
7. Move data from AX(accumulator) to memory 3004
8. Move data from CX register to memory 3006
9. Stop

Program –

Memory	Mnemonics	Operands	Comment
2000	MOV	CX, 0000	[CX] <- 0000
2003	MOV	AX, [3000]	[AX] <- [3000]
2007	MOV	BX, [3002]	[BX] <- [3002]
200B	SUB	AX, BX	[AX] <- [AX] – [BX]
200D	JNC	2010	Jump if no borrow
200F	INC	CX	[CX] <- [CX] + 1

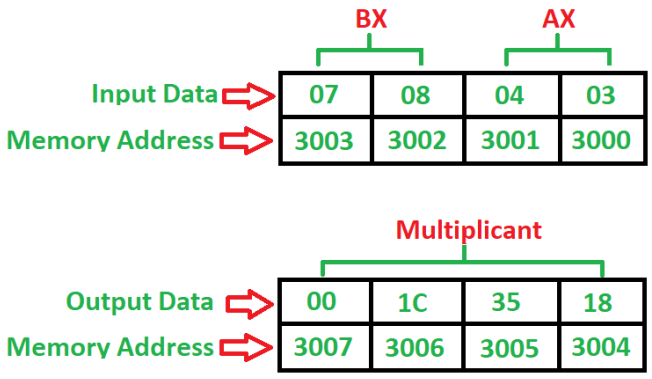
Memory	Mnemonics	Operands	Comment
2010	MOV	[3004], AX	[3004] <- [AX]
2014	MOV	[3006], CX	[3006] <- [CX]
2018	HLT		Stop

Explanation –

1. **MOV** is used to load and store data.
2. **SUB** is used to subtract two numbers where their one number is in accumulator or not.
3. **JNC** is a 2-bit command which is used to check whether the borrow is generated from accumulator or not.
4. **INC** is used to increment an register by 1.
5. **HLT** is used to stop the program.
6. **AX** is an accumulator which is used to load and store the data.
7. **BX, CX** are general purpose registers where BX is used for storing second number and CX is used to store borrow.

8086 program to multiply two 16-bit numbers

3)Problem – Write a program to multiply two 16-bit numbers where starting address is **2000** and the numbers are at **3000** and **3002** memory address and store result into **3004** and **3006** memory address.



Algorithm

1. First load the data into AX(accumulator) from memory 3000
2. Load the data into BX register from memory 3002
3. Multiply BX with Accumulator AX
4. Move data from AX(accumulator) to memory
5. Move data from DX to AX

6. Move data from AX(accumulator) to memory
7. Stop

Program –

Memory	Mnemonics	Operands	Comment
2000	MOV	AX, [3000]	[AX] <- [3000]
2004	MOV	BX, [3002]	[BX] <- [3002]
2008	MUL	BX	[AX] <- [AX] * [BX]
200A	MOV	[3004], AX	[3004] <- AX
200E	MOV	AX, DX	[AX] <- [DX]
2010	MOV	[3006], AX	[3006] <- AX
2014	HLT		Stop

Explanation –

1. **MOV** is used to load and store data.
2. **MUL** is used to multiply two 16-bit numbers.
3. **HLT** is used to stop the program.
4. **AX** is an accumulator which is used to store the result.
5. **BX, DX** are general purpose registers where BX is used for multiplication and DX is used for result.

8086 program to multiply two 8 bit numbers

4)Problem – Write a program in 8086 microprocessor to multiply two 8-bit numbers, where numbers are stored from offset 500 and store the result into offset 600.
Examples – Inputs and output are given in Hexadecimal representation.

Algorithm

1. Load data from offset 500 to register AL (first number)
2. Load data from offset 501 to register BL (second number)
3. Multiply them (AX=AL*BL)
4. Store the result (content of register AX) to offset 600
5. Stop

Program –

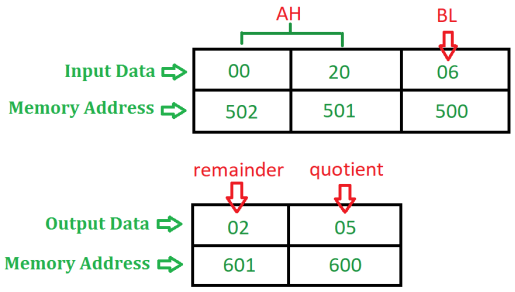
MEMORY ADDRESS	MNEMONICS	COMMENT
400	MOV SI, 500	SI=500
403	MOV DI, 600	DI=600
406	MOV AL, [SI]	AL<-[SI]
408	INC SI	SI=SI+1
409	MOV BL, [SI]	BL<-[SI]
40B	MUL BL	AX=AL*BL
40D	MOV [DI], AX	AX->[DI]
40F	HLT	END

Explanation –

1. **MOV SI, 500** set 500 to SI
2. **MOV DI, 600** set 600 to DI
3. **MOV AL, [SI]** load contents of offset SI to register AL
4. **INC SI** increase value of SI by 1
5. **MOV BL, [SI]** load contents of offset SI to register BL
6. **MUL BL** multiply contents of register AL and BL
7. **MOV [DI], AX** store the result (contents of register AX) to offset DI
8. **HLT** End.

8086 program to divide a 16 bit number by an 8 bit number

5)Problem – Write an assembly language program in 8086 microprocessor to divide a 16 bit number by an 8 bit number.



Algorithm –

1. Assign value 500 in SI and 600 in DI
2. Move the contents of [SI] in BL and increment SI by 1
3. Move the contents of [SI] and [SI + 1] in AX
4. Use **DIV** instruction to divide AX by BL
5. Move the contents of AX in [DI].
6. Halt the program.

Assumption – Initial value of each segment register is 00000.

Calculation of physical memory address –
Memory Address = Segment Register * 10(H) + offset,
where Segment Register and Offset is decided on the basis of following table.

OPERATIONS	SEGMENT REGISTER	OFFSET
Instruction fetching	Code Segment	Instruction Pointer
Data operation	Data Segment	Base Register [BX], Displacement [DISP]
Stack operation	Stack Segment	Stack Pointer (SP), Base Pointer (BP)
String as a source	Data Segment	Source Indexed (SI)
String as a destination	Extra Segment	Destination Indexed (DI)

Program –

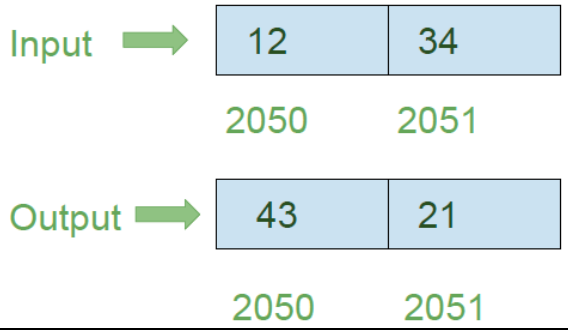
MEMORY ADDRESS	MNEMONICS	COMMENT
0400	MOV SI, 500	SI <- 500
0403	MOV DI, 600	DI <- 600
0406	MOV BL, [SI]	BL <- [SI]
0408	INC SI	SI <- SI + 1
0409	MOV AX, [SI]	AX <- [SI]
040B	DIV BL	AX <- AX / BL
040D	MOV [DI], AX	[DI] <- AX
040F	HLT	End of program

Explanation – Registers used **AX, BL, SI, DI**

1. **MOV SI, 500** assigns 500 to SI
2. **MOV DI, 600** assigns 600 to DI
3. **MOV BL, [SI]** moves the content of [SI] to BL register i.e. value of divisor will be stored in BL
4. **INC SI** increment the content of SI by 1
5. **MOV AX, [SI]** moves the content of [SI] and [SI + 1] to AX register i.e. value of dividend will be stored in AX
6. **DIV BL** divide the content of AX by BL, after execution of this instruction the quotient get stored in AL and remainder in AH
7. **MOV [DI], AX** moves the content of AX to [DI]
8. **HLT** stops executing the program and halts any further execution

8086 program to reverse 16 bit number using 8 bit operation

6)Problem – **Write an assembly language program in 8086 microprocessor to reverse 16 bit number using 8 bit operation.**
Example – **Assume 16 bit number is stored at memory location 2050 and 2051.**



ALGORITHM

1. Load contents of memory location 2050 in register AL
2. Load contents of memory location 2051 in register AH
3. Assign 0004 to CX Register Pair
4. Rotate the contents of AL by executing ROL instruction using CX
5. Rotate the contents of AH by executing ROL instruction using CX
6. Store the content of AH in memory location 2050
7. Store the content of AL in memory location 2051

Program –

Memory Address	Mnemonics	Comments
400	MOV AL, [2050]	AL<-[2050]
404	MOV AH, [2051]	AH<-[2051]
408	MOV CX, 0004	CX <- 0004
40B	ROL AL, CX	Rotate AL content left by 4 bits(value of CX)
40D	ROL AH, CX	Rotate AH content left by 4 bits(value of

Memory Address	Mnemonics	Comments
		CX)
40F	MOV [2050], AH	[2050]<-AH
413	MOV [2051], AL	[2051]<-AL
417	HLT	Stop Execution

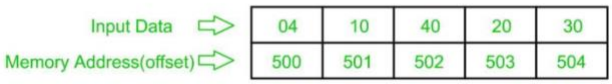
Explanation –

1. **MOV AL, [2050]:** loads contents of memory location 2050 in AL
2. **MOV AH, [2051]:** loads contents of memory location 2051 in AH
3. **MOV CX, 0004:** assign 0004 to CX register pair
4. **ROL AL, CX:** rotate the content of AL register left by 4 bits i.e. value of CX register pair
5. **ROL AH, CX:** rotate the content of AH register left by 4 bits i.e. value of CX register pair
6. **MOV [2050], AH:** stores the content of AH in 2050 memory address
7. **MOV [2051], AL:** stores the content of AL in 2051 memory address
8. **HLT:** stops executing the program

8086 program to determine largest number in an array of n numbers

7)Problem – Write a program in 8086 microprocessor to find out the largest among 8-bit n numbers, where size "n" is stored at memory address 2000 : 500 and the numbers are stored from memory address 2000 : 501 and store the result (largest number) into memory address 2000 : 600.

Example –



ALGORITHM

1. Load data from offset 500 to register CL and set register CH to 00 (for count).
2. Load first number(value) from next offset (i.e 501) to register AL and decrease count by 1.
3. Now compare value of register AL from data(value) at next offset, if that data is greater than value of register AL then update value of register AL to that data else no change, and increase offset value for next comparison and decrease count by 1 and continue this till count (value of register CX) becomes 0.
4. Store the result (value of register AL) to memory address 2000 : 600.

Program –

MEMORY ADDRESS	MNEMONICS	COMMENT
400	MOV SI, 500	SI<-500
403	MOV CL, [SI]	CL<-[SI]
405	MOV CH, 00	CH<-00
407	INC SI	SI<-SI+1
408	MOV AL, [SI]	AL<-[SI]
40A	DEC CL	CL<-CL-1
40C	INC SI	SI<-SI+1
40D	CMP AL, [SI]	AL-[SI]
40F	JNC 413	JUMP TO 413 IF CY=0
411	MOV AL, [SI]	AL<-[SI]
413	INC SI	SI<-SI+1
414	LOOP 40D	CX<-CX-1 & JUMP TO 40D IF CX NOT 0

MEMORY ADDRESS	MNEMONICS	COMMENT
416	MOV [600], AL	AL->[600]
41A	HLT	END

Explanation –

1. **MOV SI, 500** : set the value of SI to 500
2. **MOV CL, [SI]** : load data from offset SI to register CL
3. **MOV CH, 00** : set value of register CH to 00
4. **INC SI** : increase value of SI by 1.
5. **MOV AL, [SI]** : load value from offset SI to register AL
6. **DEC CL** : decrease value of register CL by 1
7. **INC SI** : increase value of SI by 1
8. **CMP AL, [SI]** : compares value of register AL and [SI] (AL-[SI])
9. **JNC 413** : jump to address 413 if carry not generated
10. **MOV AL, [SI]** : transfer data at offset SI to register AL
11. **INC SI** : increase value of SI by 1
12. **LOOP 40C** : decrease value of register CX by 1 and jump to address 40D if value of register CX is not zero
13. **MOV [600], AL** : store the value of register AL to offset 600
14. **HLT** : stop

SORTING-ASCENDING

AIM

Write a program to perform sorting

ALGORITHM

1. Load data from offset 500 to register CL (for count).
2. Travel from starting memory location to last and compare two numbers if first number is greater than second number then swap them.
3. First pass fix the position for last number.
4. Decrease the count by 1.
5. Again travel from starting memory location to (last-1, by help of count) and compare two numbers if first number is greater than second number then swap them.

6. Second pass fix the position for last two numbers.
7. Repeated.

PROGRAM

Alternate Program is available at the end of the PDF

ADDRESS	MNEMONICS
400	MOV SI, 500
403	MOV CL, [SI]
405	DEC CL
407	MOV SI, 500
40A	MOV CH, [SI]
40C	DEC CH
40E	INC SI
40F	MOV AL, [SI]
411	INC SI
412	CMP AL, [SI]
414	JC 41C
416	XCHG AL, [SI]
418	DEC SI
419	XCHG AL, [SI]
41B	INC SI
41C	DEC CH
41E	JNZ 40F
420	DEC CL
422	JNZ 407
424	HLT

INPUT

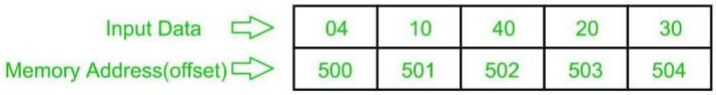
ADDRESS	VALUE
500	04
501	F9
502	F2
503	39
504	05

OUTPUT

ADDRESS	VALUE
501	05
502	39
503	F2
504	F9

8086 program to determine largest number in an array of n numbers

8)Problem – Write a program in 8086 microprocessor to find out the largest among 8-bit n numbers, where size "n" is stored at memory address 2000 : 500 and the numbers are stored from memory address 2000 : 501 and store the result (largest number) into memory address 2000 : 600.



Algorithm

1. Load data from offset 500 to register CL and set register CH to 00 (for count).
2. Load first number(value) from next offset (i.e 501) to register AL and decrease count by 1.
3. Now compare value of register AL from data(value) at next offset, if that data is greater than value of register AL then update value of register AL to that data else no change, and increase offset value for next comparison and decrease count by 1 and continue this till count (value of register CX) becomes 0.
4. Store the result (value of register AL) to memory address 2000 : 600.

Program –

MEMORY ADDRESS	MNEMONICS	COMMENT
400	MOV SI, 500	SI<-500
403	MOV CL, [SI]	CL<-[SI]
405	MOV CH, 00	CH<-00
407	INC SI	SI<-SI+1
408	MOV AL, [SI]	AL<-[SI]
40A	DEC CL	CL<-CL-1
40C	INC SI	SI<-SI+1
40D	CMP AL, [SI]	AL-[SI]
40F	JNC 413	JUMP TO 413 IF CY=0
411	MOV AL, [SI]	AL<-[SI]
413	INC SI	SI<-SI+1
414	LOOP 40D	CX<-CX-1 & JUMP TO 40D IF CX NOT 0
416	MOV [600], AL	AL->[600]
41A	HLT	END

Explanation –

1. **MOV SI, 500** : set the value of SI to 500
2. **MOV CL, [SI]** : load data from offset SI to register CL

3. **MOV CH, 00** : set value of register CH to 00
4. **INC SI** : increase value of SI by 1.
5. **MOV AL, [SI]** : load value from offset SI to register AL
6. **DEC CL** : decrease value of register CL by 1
7. **INC SI** : increase value of SI by 1
8. **CMP AL, [SI]** : compares value of register AL and [SI] (AL-[SI])
9. **JNC 413** : jump to address 413 if carry not generated
10. **MOV AL, [SI]** : transfer data at offset SI to register AL
11. **INC SI** : increase value of SI by 1
12. **LOOP 40C** : decrease value of register CX by 1 and jump to address 40D if value of register CX is not zero
13. **MOV [600], AL** : store the value of register AL to offset 600

HLT : stop

CODE CONVERSIONS

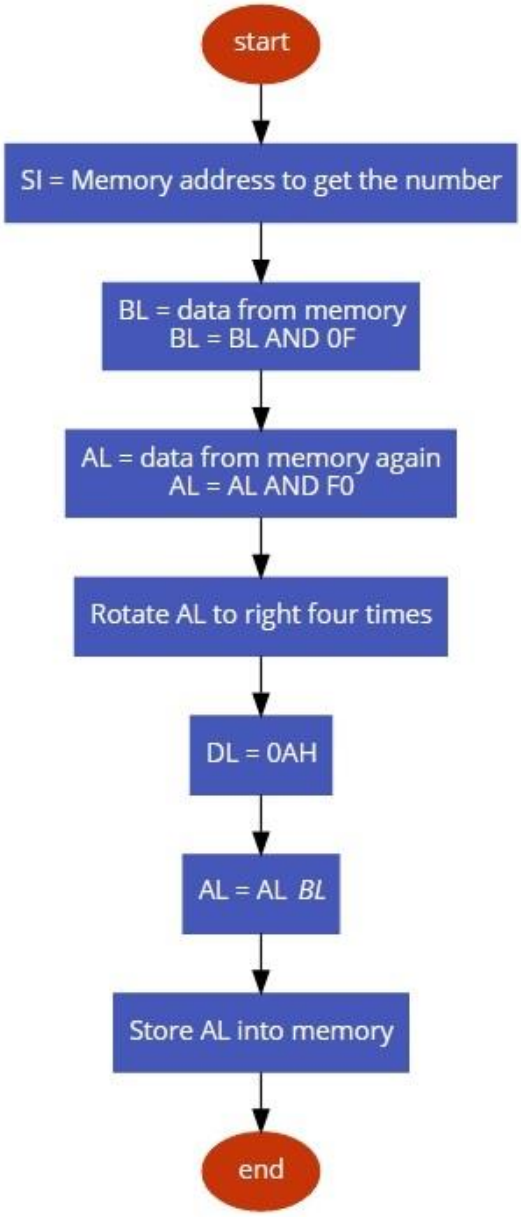
Pgm.No. 25

BCD TO HEXADECIMAL

AIM

Write a program to perform conversion of 8 bit BCD number into hexadecimal number

FLOWCHART



PROGRAM

ADDRESS	MNEMONICS
400	MOV SI, 500
403	MOV DI, 600
406	MOV BL, [SI]
408	AND BL, 0F
040A	MOV AL, [SI]

040C	AND AL, F0
040E	MOV CL, 04
410	ROR AL, CL
412	MOV DL, 0A
414	MUL DL
416	ADD AL, BL
418	MOV [DI], AL
041A	HLT

INPUT

ADDRESS	**VALUE**
500	25

OUTPUT

ADDRESS	**VALUE**
600	19

Pgm.No.

ASCII TO BCD

AIM

Write a program to perform conversion of ASCII(in hex) value of number to its BCD(decimal) number

ALORITHM

1. Load the value from memory into register AL
2. Then perform and operation on register AL with 0F
3. Move the result value from register AL to memory
4. Terminate the program

ASCII (in Hex)	30	31	32	33	34	35	36	37	38	39
BCD	00	01	02	03	04	05	06	07	08	09

PROGRAM

ADDRESS	MNEMONICS
400	MOV AL,[2050]
403	AND AL,0F
405	MOV [3050],AL
408	HLT

INPUT

ADDRESS	VALUE
2050	39

OUTPUT

ADDRESS	VALUE
3050	09

BCD TO ASCII

AIM

Write a program to perform conversion of 8 bit BCD number to ASCII code

ALGORITHM

- Load contents of memory location 2000 in register AL.
- Copy contents of register AL in register AH.
- Perform AND operation on register AL with 0F.
- Assign 04 to CL Register.
- Shift the contents of AH by executing SHR instruction using CL.

- Perform OR operation on register AX with 3030.

PROGRAM

ADDRESS	**MNEMONICS**
400	MOV AL, [2000]
404	MOV AH, AL
406	AND AL, 0F
408	MOV CL, 04
40A	SHR AH, CL
40C	OR AX, 3030
40F	MOV [3000], AX
413	HLT

INPUT

ADDRESS	**VALUE**
2000	98

OUTPUT

ADDRESS	**VALUE**
3000	38
3001	39

last address. In the second cycle of bubble sort (i.e., N-2 comparisons) second largest number will be stored in the last but one address.

If the instruction JB is replaced by JA, the numbers will be sorted in descending order.

Algorithm

1. Compare the first and second numbers.
2. If the first one is smaller go to next step. Otherwise, exchange the data.
3. Compare the second and third and do the previous step.
4. Repeat above two steps until N-2 comparisons are done. Now the second largest will be available in the last but one (penultimate) memory location.
5. Continue above step decrementing the number of comparisons by one each time, until all data are sorted.

Program

```
1000: 8B 1E 00 20        MOV BX, [2000]      ; Quantity of numbers in BX
1004: 4B                 DEC BX              ; Cycle counter set to N-1
1005: 89 D9      LOOP1:  MOV CX, BX          ; Comparison counter set to N-1
1007: C7 C6 02 20        MOV SI, 2002
100B: 8B 04      LOOP2:  MOV AX, [SI]        ; First number to AX
100D: 81 C6 02 00        ADD SI, 0002        ; Points to next number
1011: 3B 04              CMP AX, [SI]        ; Is first number is < next?
1013: 72 05              JB GO (101A)        ; If not go to GO
1015: 87 04              XCHG [SI],AX        ; Else exchange
1017: 89 44 FE           MOV [SI-02], AX     ; Copy small number in previous location
101A: E2 EF      GO:     LOOP LOOP2 (100B)   ; If CX ≠ 0; compare next
101C: 4B                 DEC BX
101D: 75 E6              JNZ LOOP1 (1005)    ; Do next cycle
101F: F4                 HLT
```

Procedure

1. Enter the length of array in memory location 2000H and the array in memory locations starting from 2002H.
2. Enter the program and execute it.
3. Verify the sorted data in memory location starting from 2002H.
4. Change the instruction JB to JA and verify the program sorts the numbers in descending order.

4.17 Square root of a number

Aim To write a program to calculate the square root of a number.

Exercise

1. A number is stored in a location. If the number is prime, stepper motor must rotate in clock wise direction. Otherwise, rotate in clockwise direction.
2. Write a program to rotate a stepper motor 90° in clock wise direction with 1 5 sec delay between steps. The speed of rotation should be doubled for the remaining 270°.

4.23 Rolling display

Aim To make a rotating display by programming display/keyboard interface chip 8279.

Theory The 8279 is a programmable keyboard/display controller IC. It connects a matrix type keyboard and a multi-digit seven segment display unit with CPU.

8279 contains two registers namely command register and data register. They are treated as ports.

Display is done in two modes. Left entry mode and right entry mode. Former is said to be the type writer mode and latter is calculator mode.

8279 must be programmed by writing a control word into the command register. The format of the command word is

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	D	D	K	K	K

DD is the display mode and KKK is the key board mode.

The four options corresponding to DD are given in table below. Key board mode KKK is not a matter of concern in this program.

DD	Display Mode	DD	Display Mode
00	16 digits, typewriter	10	16 digits, calculator
01	8 digits, typewriter	11	8 digits, calculator

Seven segment LED display is formed by 8 LEDs including one for decimal point. In order to display a letter or a digit, the segment code is generated and written into the data register of 8279. Segment code is generated by allocating a 0 or a 1 at that position. 0 corresponding ON and 1 corresponding OFF for a common cathode display. In common anode display it is vice versa. For example, to display the character 'H' segments b, c, e, f and g must be ON. So the bit pattern will be 01100111 (67H).

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
d	c	b	a	dp	e	g	f

Program

```
1000: C7 C6 00 12   START:  MOV SI,1200      ; Address of look up table
1004: C7 C1 0F 00           MOV CX,000F      ; 16 characters
1008: C6 C0 10              MOV AL,10        ; Control word in AL
100B: E6 C2                 OUT C2,AL        ; C2 is control word register address
100D: C6 C0 CC              MOV AL,0CC
1010: E6 C2                 OUT C2,AL
1012: C6 C0 90              MOV AL,90
1015: E6 C2                 MOV C2,AL
1017: 8A 04         NEXT:   MOV AL,[SI]
1019: E6 C0                 OUT C0,AL        ; Character to data register
101B: E8 E2 04              CALL DELAY (1500)
101E: 46                    INC SI
101F: E2 F6                 LOOP NEXT (1017)
1021: E9 DC FF              JMP START (1000)

DELAY:
1500: C7 C2 FF A0           MOV DX,0A0FF
1504: 4A            REPEAT: DEC DX
1505: 75 FD                 JNZ REPEAT (1504)
1507: C3                    RET

LOOKUP TABLE:
1200: FF FF FF FF
1204: FF FF FF FF
1208: 98 68 7C C8
120C: FF 1C 29 FF
```

Procedure

1. Enter the main program, delay program and look up table data in the memory addresses starting from 1000H, 1500H and 1200H respectively.
2. Execute the program and observe the word 'HELP US' rolling on the display panel.
3. Change the control word 10 to 00, 18 and 08 in the program and observe the changes happening in the display.

4.24 Waveform generation using DAC

Aim Generate a square wave of 50% duty cycle using DAC interface card.

Theory The DAC interface can be used to generate various waveforms using a microprocessor. VBMB-002 is a DAC card manufactured by Vi Microsystems, Chennai suitable for interfacing

```
1016: 01 C8          ADD AX, CX        ; AX = P1 + P2
1018: 81 D2 00 00    ADC DX, 00        ; DX = Carry + P3
101C: 89 C6          MOV SI, AX
101E: 8B 06 00 20    MOV DI, DX
1020: 8B 1E 06 20    MOV AX, [2000]    ; AX = X0
1024: F7 E3          MOV BX, [2006]    ; BX = X3
1028: 01 F0          MUL BX
102A: 89 06 00 212   ADD AX, SI        ; AX = P1 + P2 + P4
102C: 11 D7          MOV [1202], AX
1030: 9F             ADC DI, DX        ; D1 = P3 + P5 + Carry
1032: 80 E4 0        LAHF
1033: 31 C9          AND AH, 01
1036: 89 D7          XOR CX,CX
1038: 88 E1          MOV CL,AH         ; CX = Carry
103A: 8B 06 02 20    MOV AX, [2002]    ; AX = X1
103E: F7 E3          MUL BX
1040: 01 F8          ADD AX, DI        ; AX = P3 + P5P6
1042: 89 06 04 12    MOV [1204], AX
1046: 11 CA          ADC DX,CX         ; DX = Carry + P7
1048: 89 16 06 12    MOV [1206], DX
104C:F4              HLT
```

4.22 Stepper motor interface

Aim To write an assembly language program to rotate a stepper motor in anticlock-wise direction continuously.

Theory Stepper motors are used in big clocks, printers, digital x-y plotters, floppy disk drives, numerical control systems, industrial robots etc. They capable of accepting pulses from the microprocessors, and rotate in either direction taking discrete steps.

The step angle of the motor is 1.8°. It means that to rotate 180°, it requires 100 steps. The stepper motor has four stator windings. Four pins of 8255 viz; PA_0, PA_1, PA_2 and PA_3 are used to excite the windings of the stepper motor. Each of the four pins is connected to the respective windings through interface circuit. The stator poles are excited by feeding a pulse to it.

This program is written to rotate the stepper motor in anticlock wise direction.

Program

```
1000: C7 C7 18 10  START: MOV DI,TABLE(1018)
1004: C6 C1 04           MOV CL,04
1007: 8A 05        ROTATE: MOV AL,[DI]
1009: E6 C0              OUT C0,AL      ; C0 is port address of 8255 data reg.
100B: C7 C2 FF 1F        MOV DX,1FFF    ; Delay between steps
100F: 4A           BACK: DEC DX
1010: 75 FD              JNZ BACK(100F)
1012: 47                 INC DI
1013: E2 F2              LOOP ROTATE(1007)  ; Take next from look up table
1015: E9 E8 FF           JMP START(1000)    ; Continue rotation
```

Look up table

```
1018: 09 05 06 0A  TABLE: 09 05 06 0A
```

Procedure

1. Connect the stepper motor interface card with the microprocessor kit.
2. Enter and execute the program. The stepper motor starts rotating in anticlock wise direction. Press RES key to stop rotation.
3. Reverse the look up table entry and repeat the above step. Now the stepper motor rotates in clock wise direction.

Program to rotate for a specified number of steps

```
1000: C6 C3 C8           MOV BL,C8      ; C8 = 200 steps for 360°
1003: C7C72010   START:  MOV DI, TABLE(1020)
1007: C6 C1 04           MOV CL,04
100A:8A 05       ROTATE: MOV AL,[DI]
100C: E6 C0              OUT C0,AL
101E: FE CB              DEC BL
1010: 74 0D              JZ HALT(101F)  ; Stop after one rotation
1012: C7 C2 10 10        MOV DX,1010
1016: 4A         BACK:   DEC DX         ; Delay between steps
1017: 75 FD              JNZ BACK(1016)
1019: 47                 INC DI
101A: E2 EE              LOOP ROTATE(100A)  ; Get next entry from look up
101C: E9 E4 FF           JMP START(1003)    ; Continue rotation
101F: F4         HALT:   HLT
```

Look up table

```
1020: 09 05 06 0A  TABLE: 09 05 06 0A
```

Procedure

1. Enter and execute the program. The stepper motor starts rotating in anticlock wise direction for 360° and stops.
2. Reduce the number in DX (say, 1000) and execute the program. Motor rotates faster.
3. Vary the number in the register BL to vary the number of steps of rotation.
4. Reverse the look up table entry and repeat the above step. Now the stepper motor rotates in clock wise direction.

VBMB-002 is a DAC card manufactured by Vi Microsystems, Chenn