Department of Electronics & Communication **Amruta Institute of Engineering & Management**

**Sciences**

Bidadi Industrial Area, Off Mysore Road,

Near Toyota Motors, Bengaluru – 562 109 INDIA

**MICROPROCESSOR LAB MANUAL**

**(06ECL68)**

VI SEMESTER

8086 Lab Manual AIeMS

DEPT OF ECE 1 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm**

**1) Declare memory model as small 2) Initialize 5 bytes of source data(X) in data**

**segments and create 3 memory locations of byte type for the destination data (Y). 3) Initialize the data segment registers. 4) Load the corresponding offset address of source and**

**destination into index registers. 5) Initialize the counter(equivalent to the no of bytes**

**in source index) 6) Start the loop for copying source index values to**

**Destination 7) Repeat the step6 till count become zero. 8) Exit to dos.**

DEPT OF ECE 2 VI SEMESTER

8086 Lab Manual AIeMS

**1) Develop and execute an Assembly language program to transfer a given block of data byte from source memory block to destination memory block with overlap.**

**Program:**

**.model small .data Y DB 3 DUP(?) X DB 11H, 22H, 33H, 44H, 55H .code**

**MOV AX,@data MOV DS, AX LEA SI, X LEA DI, Y MOV CX,0005H BACK: MOV AL, [SI] MOV [DI], AL INC SI INC DI DEC CX JNZ BACK MOV AH, 4CH INT 21H END**

**Result:**

**Before execution:**

**4AE2:0000 8A 00 88 05 46 47 49 75 F7 B4 4C CD 21 00 00 00 4AE2:0010 00 11 22 33 44 55 00 00 00 00 00 00 00 00 00 00**

**After execution:**

**4AE2:0000 8A 00 88 05 46 47 49 75 F7 B4 4C CD 21 00 11 22 4AE2:0010 33 44 55 33 44 55 00 00 00 00 00 00 00 00 00 00**

DEPT OF ECE 3 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm**

**1) Declare memory model as small 2) Initialize 5 bytes of source data(X) in data**

**segments and create 5 memory locations of byte type for the destination data (Y). 3) Initialize the data segment registers. 4) Load the corresponding offset address of source and**

**destination into index registers. 5) Initialize the counter(equivalent to the no of bytes**

**in source index) 6) Start the loop for copying source index values to**

**Destination 7) Repeat the step6 till count become zero. 8) Exit to dos.**

DEPT OF ECE 4 VI SEMESTER

8086 Lab Manual AIeMS

**2) Develop and execute an Assembly language program to transfer a given block of data byte from source memory block to destination memory block without overlap.**

**Program:**

**.model small .data Y DB 5 DUP(?)**

**X DB 11H, 22H, 33H, 44H, 55H .code MOV AX,@data**

**MOV DS, AX LEA SI, X LEA DI, Y MOV CX,0005H BACK: MOV AL, [SI] MOV [DI], AL INC SI INC DI DEC Cx JNZ BACK MOV AH, 4CH INT 21H END**

**Result:**

**Before execution:**

**4AE2:0000 8A 04 88 05 46 47 49 75 F7 B4 4C CD 21 00 00 00 4AE2:0010 00 00 00 11 22 33 44 55 00 00 00 00 00 00 00 00**

**After execution:**

**4AE2:0000 8A 04 88 05 46 47 49 75 F7 B4 4C CD 21 00 11 22 4AE2:0010 33 44 55 11 22 33 44 55 00 00 00 00 00 00 00 00**

DEPT OF ECE 5 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm**

**1) Declare memory model as small 2) Initialize 5 words of source data(X) in data**

**segments and create 3 memory locations of word type for the destination data (Y). 3) Initialize the data segment registers. 4) Load the corresponding offset address of source and**

**destination into index registers. 5) Initialize the counter(equivalent to the no of bytes**

**in source index) 6) Start the loop for copying source index values to**

**Destination 7) Repeat the step6 till count become zero. 8) Exit to dos.**

DEPT OF ECE 6 VI SEMESTER

8086 Lab Manual AIeMS

**3) Develop and execute an Assembly language program to transfer a given block of data word from source memory block to destination memory block with overlap.**

**Program:**

**.model small .data Y DW 3 DUP(?)**

**X DW 1111H,2222H,3333H,4444H,5555H .code**

**MOV AX,@data MOV DS, AX LEA SI, X LEA DI, Y MOV CX,0005H BACK: MOV AX, [SI] MOV [DI], AX INC SI INC SI INC DI INC DI DEC CX JNZ BACK MOV AH, 4CH INT 21H END**

**Result:**

**Before execution:**

**4AE2:0000 00 00 00 00 00 00 11 11 22 22 33 33 44 44 55 55 4AE2:0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**

**After execution:**

**4AE2:0000 11 11 22 22 33 33 44 44 55 55 33 33 44 44 55 55 4AE2:0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00** DEPT OF ECE 7 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm**

**1) Declare memory model as small 2) Initialize 5 words of source data(X) in data**

**segments and create 5 memory locations of word type for the destination data (Y). 3) Initialize the data segment registers. 4) Load the corresponding offset address of source and**

**destination into index registers. 5) Initialize the counter (equivalent to the no of bytes**

**in source index) 6) Start the loop for copying source index values to**

**Destination 7) Repeat the step6 till count become zero. 8) Exit to dos.**

DEPT OF ECE 8 VI SEMESTER

8086 Lab Manual AIeMS

**4) Develop and execute an Assembly language program to transfer a given block of data word from source memory block to destination memory block without overlap.**

**Program:**

**.model small .data Y DW 5 DUP(?)**

**X DW 1111H,2222H,3333H,4444H,5555H .code**

**MOV AX,@data MOV DS, AX LEA SI, X LEA DI, Y MOV CX,0005H BACK: MOV AX, [SI] MOV [DI], AX INC SI INC SI INC DI INC DI DEC CX JNZ BACK MOV AH, 4CH INT 21H END**

**Result:**

**Before execution:**

**4AE2:0000 00 00 00 00 00 00 00 00 00 00 11 11 22 22 33 33 4AE2:0010 44 44 55 55 00 00 00 00 00 00 00 00 00 00 00 00**

**After execution:**

**4AE2:0000 11 11 22 22 33 33 44 44 55 55 11 11 22 22 33 33 4AE2:0010 44 44 55 55 00 00 00 00 00 00 00 00 00 00 00 00**

DEPT OF ECE 9 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm:**

**1) Declare memory model as small 2) Initialize 5 bytes of source data (N1) and 5 bytes of**

**destination data (N2). 3) Initialize the data segment registers. 4) Load the corresponding offset address of source and**

**destination into index registers. 5) Initialize the counter (equivalent to the no of bytes**

**in source index) 6) Start the loop for copying source index values to**

**Destination after exchanging the data in registers. 7) Repeat the step6 till count become zero. 8) Exit to dos.**

DEPT OF ECE 10 VI SEMESTER

8086 Lab Manual AIeMS

**5) Develop and execute an Assembly language program to interchange two blocks of data.**

**Program:**

**.model small .data N1 db 04h,05h,06h,07h N2 db 08h,09h,10h,11h**

**.code MOV ax,@data**

**MOV ds,ax LEA si,N1 LEA di,N2 MOV cx,04h Back: MOV al,[si] MOV bl,[di] XCHG al,bl MOV [si],al MOV [di],bl INC si INC di DEC cx JNZ back MOV ah,4ch INT 21h END**

**Result:**

**Before execution:**

**4258:0000 4C C0 21 00 04 05 06 07 – 08 09 10 11 00 00 00 00 4258:0010 00 00 00 00 00 00 00 00 – 00 00 00 00 00 00 00 00**

**After execution:**

**4258:0000 4C C0 21 00 08 09 10 11 – 04 05 06 07 00 00 00 00 4258:0010 00 00 00 00 00 00 00 00 – 00 00 00 00 00 00 00 00**

DEPT OF ECE 11 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm**

**1) Declare memory model as small 2) Initialize the data segment for inputs x, y and output z**

**of type word 3) Initialize the data segment registers. 4) Initialize the counter to zero 5) Move the x data to ax register and add to y 6) If there is a carry form the addition go to loop(L1)and**

**increment the cx register else store the results in z locations 7) Exit to dos.**

DEPT OF ECE 12 VI SEMESTER

8086 Lab Manual AIeMS

**6) Develop and execute an Assembly language program to add**

**two 16 bit numbers.**

**Program:**

**.model small .data x dw 0ffffh y dw 0ffffh z dw ? .code MOV ax, @ data MOV ds, ax MOV ch, 00h MOV ax, x ADD ax, y JC l1 JMP l2 L1: INC ch L2: MOV z, al**

**MOV z+1, ah MOV z+2,ch MOV ah, 4ch INT 21h END**

**Result:**

**Before execution:**

**4258:0000 B4 4C CD 21 FF FF FF FF – 00 00 01 11 00 00 00 00 4258:0010 00 00 00 00 00 00 00 00 – 00 00 00 00 00 00 00 00**

**After execution:**

**4258:0000 B4 4C CD 21 FF FF FF FF – FE FF 01 11 00 00 00 00 4258:0010 00 00 00 00 00 00 00 00 – 00 00 00 00 00 00 00 00**

DEPT OF ECE 13 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm**

**1) Declare memory model as small 2) Initialize the data segment for inputs x, y and output z**

**of type word 3) Initialize the data segment registers. 4) Initialize the counter to zero 5) Move the x data to ax register and do the subtraction. 6) If there is a borrow form the subtraction go to**

**loop(L1)and increment the cx register else store the results in z locations 7) Exit to dos.**

DEPT OF ECE 14 VI SEMESTER

8086 Lab Manual AIeMS

**7) Develop and execute an Assembly language program to**

**subtract two 16 bit numbers.**

**Program:**

**.model small .data**

**x dw 0ffffh y dw 0ff44h z dw ? .code**

**mov ax, @ data mov ds, ax mov ch, 00h mov ax, x sub ax, y Jc l1 Jmp l2 L1: inc ch L2: mov z, al**

**mov z+1, ah mov z+2,ch mov ah, 4ch int 21h end**

**Result:**

**Before execution:**

**4258:0000 B4 4C CD 21 FF FF 44 FF – 00 00 00 11 00 00 00 00 4258:0010 00 00 00 00 00 00 00 00 – 00 00 00 00 00 00 00 00**

**After execution:**

**4258:0000 B4 4C CD 21 FF FF 44 FF – BB 00 00 11 00 00 00 00 4258:0010 00 00 00 00 00 00 00 00 – 00 00 00 00 00 00 00 00**

DEPT OF ECE 15 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm**

**1) Declare memory model as small 2) Initialize the data segment for inputs num1, num2 and**

**output res of type word 3) Initialize the data segment registers. 4) Move the num1 data to ax register and do the**

**Unsigned multiplication using mul instruction. 5) Store the result in res and res+2 locations from the ax**

**and dx registers 6) Exit to dos.**

DEPT OF ECE 16 VI SEMESTER

8086 Lab Manual AIeMS

**8) Develop and execute an Assembly language program to**

**Multiply two unsigned numbers.**

**Program:**

**.model small .data**

**num1 dw 1234h num2 dw 1234h res dw ? .code**

**mov ax,@data mov ds,ax mov ax,num1 mul num2 mov res,ax mov res+2,dx mov ah,4ch int 21h end**

**Result:**

**52E3:0000 16 0E 00 B4 4C CD 21 00–34 12 34 12 90 5A 4B 01 52E3:0010 FF FF 00 00 00 00 00 00-00 00 00 00 00 00 00 00**

DEPT OF ECE 17 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm**

**1) Declare memory model as small 2) Initialize the data segment for inputs num1, num2 and**

**output res of type word 3) Initialize the data segment registers. 4) Move the num1 data num2 to ax register and num1 to ax**

**register do the signed multiplication using imul instruction. 5) Store the result in res and res+2 locations from the ax**

**and dx registers 6) Exit to dos.**

DEPT OF ECE 18 VI SEMESTER

8086 Lab Manual AIeMS

**9) Develop and execute an Assembly language program to**

**Multiply two signed numbers.**

**Program:**

**.model small .data**

**num1 dw 0056h num2 dw -14h res dw ? res1 dw ?**

**.code**

**mov ax,@data mov ds,ax mov bx,num1 mov ax,num2 imul bx mov res,ax mov res1,dx mov ah,4ch int 21h end**

**Result:**

**4AE2:0000 00 89 16 10 00 B4 4C CD–21 00 56 00 EC FF 48 F9 4AE2:0010 FF FF 00 00 00 00 00 00-00 00 00 00 00 00 00 00**

DEPT OF ECE 19 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm**

**1) Declare memory model as small 2) Initialize the data segment for inputs num1, num2 and**

**output res of type word 3) Initialize the data segment registers. 4) Move the num1 data num2 to ax register and num1 to ax**

**register do the signed division using idiv instruction. 5) Store the result in res and res+2 locations from the ax**

**and dx registers 6) Exit to dos**

DEPT OF ECE 20 VI SEMESTER

8086 Lab Manual AIeMS

**10) Develop and execute an Assembly language program to**

**Divide two signed numbers.**

**Program:**

**.model small .data num1 dw 3276 num2 dw -351 res dw ? .code mov ax,@data**

**mov ds,ax mov ax,num1 mov bx,num2 idiv bx mov res,ax mov res+2,dx mov ah,4ch int 21h end**

**Result:**

**4AE2:0000 00 89 16 10 00 B4 4C CD–21 00 CC 0C A1 FE F7 FF 4AE2:0010 75 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00**

DEPT OF ECE 21 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm**

**1) Declare memory model as small. 2) Initialize the data segment for inputs num1, num2 and**

**output quot and rem of type word. 3) Initialize the data segment registers. 4) Move the num1 data num1 to ax register do the unsigned**

**division using div instruction. 5) Store the result in quot and rem locations from the ax**

**and dx registers respectively. 6) Exit to dos.**

DEPT OF ECE 22 VI SEMESTER

8086 Lab Manual AIeMS

**11) Develop and execute an Assembly language program to**

**Divide two unsigned numbers.**

**Program:**

**.model small .data num1 dw 0ffffh**

**num2 dw 0fah quot dw ? rem dw ? .code mov ax,@data**

**mov ds,ax mov ax,num1 div num2 mov quot,ax mov rem,dx mov ah,4ch int 21h end**

**Result:**

**4AE2:0000 16 0E 00 B4 4C CD 21 00–FF FF FA 00 06 01 23 00 4AE2:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00**

DEPT OF ECE 23 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm**

**1) Declare memory model as small 2) Initialize the data segment for inputs num1, num2 and**

**output res of type word 3) Initialize the data segment registers. 4) Move the num1 data num1 to ax register do the addition using**

**add instruction. 5) Use AAA to update the result (AL value) in ASCII (AH has**

**been cleared) 6) ADD 30H to get the display in ASCII format and store the**

**result from ax to res location. 7) Exit to dos**

DEPT OF ECE 24 VI SEMESTER

8086 Lab Manual AIeMS

**12) Develop and execute an Assembly language program to**

**Add two ASCII numbers.**

**Program:**

**.model small .data**

**num1 dw 31h num2 dw 51h res dw ? .code**

**mov ax, @data mov ds, ax mov ax, num1 add ax, num2 aaa add ax, 3030h mov res, ax mov ah, 4ch int 21h end**

**Result:**

**4AE2:0000 A3 0C 00 B4 4C CD 21 00–31 00 51 00 32 30 00 00 4AE2:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00**

DEPT OF ECE 25 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm**

**1) Declare memory model as small. 2) Initialize the data segment for inputs num1, num2 and**

**output res of type word. 3) Initialize the data segment registers. 4) Move the num1 data num1 to ax register do the subtraction**

**using sub instruction. 5) Use AAD to update the result (AL value) in ASCII (AH has**

**been cleared). 6) ADD 30H to get the display in ASCII format and store the**

**result from ax register to res location. 7) Exit to dos.**

DEPT OF ECE 26 VI SEMESTER

8086 Lab Manual AIeMS

**13) Develop and execute an Assembly language program to**

**subtract two ASCII numbers**

**Program:**

**.model small .data**

**num1 dw 6Oh num2 dw 73h res db ? .code**

**mov ax, @data mov ds, ax mov ax, num1 sub ax,num2 aas add ax,3030h mov res, ax mov ah,4ch int 21h end**

**Result:**

**4AE2:0000 A3 0C 00 B4 4C CD 21 00–60 00 73 00 37 2E 00 00 4AE2:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00**

DEPT OF ECE 27 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm**

**1) Declare memory model as small. 2) Initialize the data segment for inputs num1, num2 and**

**output res of type word. 3) Initialize the data segment registers. 4) Move the num1 data num1 to al register and num2 to bl**

**Register and multiply al with bl. 5) Use AAM to update the result (AL value) in ASCII (AH has**

**been cleared). 6) ADD 30H to get the display in ASCII format and store the**

**result from ax register to res location. 7) Exit to dos.**

DEPT OF ECE 28 VI SEMESTER

8086 Lab Manual AIeMS

**14) Develop and execute an Assembly language program to**

**multiply two ASCII numbers**

**Program:**

**.model small .data**

**num1 dw 0006h num2 dw 0007h res dw ?**

**.code**

**mov ax, @data mov ds, ax mov al, num1 mov bl, num2 mul bl aam add ax, 3030h mov res, ax mov ah,4ch int 21h end**

**Result:**

**4AE2:0000 05 30 30 A3 0E 00 B4 4C–CD 21 06 00 07 00 32 34 4AE2:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00**

DEPT OF ECE 29 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm**

**1) Declare memory model as small. 2) Initialize the data segment for inputs num1, num2 and**

**output res of type word. 3) Initialize the data segment registers. 4) Move the num1 data num1 to ax register and num2 to bl**

**Register. 5) Use AAM to update the result (AL value) in ASCII(AH has**

**been cleared). 6) divide with bl. 7) ADD 30H to get the display in ASCII format and store the**

**result from ax register to res location. 8) Exit to dos.**

DEPT OF ECE 30 VI SEMESTER

8086 Lab Manual AIeMS

**15) Develop and execute an Assembly language program to**

**divide two ASCII numbers**

**Program:**

**.model small .data**

**num1 dw 27h num2 dw 18h res dw ? .code**

**mov ax, @data mov ds, ax mov ax, num1 mov bl, num2 aad div bl add ax, 3030h mov res, ax mov ah,4ch int 21h end**

**Result:**

**4AE2:0000 05 30 30 A3 0E 00 B4 4C–CD 21 27 00 18 00 31 3F 4AE2:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00**

DEPT OF ECE 31 VI SEMESTER

8086 Lab Manual AIeMS

**Description: The LCM of two numbers is found by dividing the first number by the second number. If the remainder is zero, then the second number is the LCM. If there is a remainder, the first number is added to itself to get a new number. Again divide the new number by the second number. If there is no remainder, then the new number is the LCM. If there is a remainder, the new number is added to the first number and once again the number becomes the new number. The process continues till the remainder becomes zero. Example: First No=25 Second No =15 (decimal numbers) Iteration Operation Remainder New Number 1 25%15 10 25+25=50 2 50%15 5 50+25=75 3 75%15 0 LCM is 75 in decimal.**

**Algorithm:**

**1) Initialize data of type word in memory locations and Data**

**Segment register with appropriate address. 2) Fetch the 16-bit data into AX and BX from location X and**

**Y. 3) Initialize DX to 0000H. 4) Save both AX and DX to the top of the stack. 5) Divide AX-DX by contents of BX. 6) Is the remainder zero? 7) If yes go to step 10.If No then restore the data from the**

**top of the stack. Add the contents of AX-DX to X. 8) Go to step 4. 9) Result is popped from the top of the stack and stored in**

**memory.(Higher order 16-bits first and then lower order 16-bits) 10) Terminate the program.**

**RESULT:**

**Before Execution: 477F:0000 25 00 15 00 00 00 00 00 - 00 00 00 00 00 00 00 00 477F:0010 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00**

**After Execution: 477F:0000 25 00 15 00 09 03 00 00 - 00 00 00 00 00 00 00 00** DEPT OF ECE 32 VI SEMESTER

8086 Lab Manual AIeMS

**477F:0010 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00**

**16) Develop and execute an assembly language program to find the LCM of two 16 bit unsigned integers.**

**.model small**

**.data**

**X dw 25h Y dw 15h Z dw 2dup (?)**

**.code**

**MOV AX, @DATA MOV DS, AX MOV AX, X MOV BX, Y MOV DX, 00H BACK: PUSH AX PUSH DX DIV BX CMP DX, 00H JZ NEXT POP DX POP AX ADD AX, X JNC L1 INC DX L1: JMP BACK NEXT: POP Z+2**

**POP Z MOV AH, 4CH INT 21H END**

DEPT OF ECE 33 VI SEMESTER

8086 Lab Manual AIeMS

**Description: GCD of two numbers are performed by dividing the greater number by the smaller number till the remainder is zero. If it is zero, the divisor is the GCD. If not the remainder and the divisor of the previous division are the set of new numbers for division. The process is repeated by the dividing greater of the two numbers by the smaller number till the remainder is zero. Example: First No=90 Second No =120 (decimal numbers) Iteration Operation Remainder 1 120%90 30 2 90%30 0 GCD is 30 in decimal.**

**Algorithm:**

**1) Initialize data of type word in memory locations and Data**

**Segment register with appropriate address. 2) Load AX and BX registers with operands. 3) Are the two numbers equal? If yes, go to step 10. 4) Is Num1 greater than Num2? If yes, go to step 6. 5) Exchange AX and BX register contents such that AX contains**

**the bigger number. 6) Initialize DX register with 00H. DX will hold the**

**remainder of the division. 7) Perform division. 8) If there is no remainder go to step 10. 9) Otherwise move the remainder to AX register and go to step**

**4. 10) Save the contents of BX as GCD. 11) Terminate the program.**

**RESULT: Before Execution: 477F:0000 xx xx xx xx xx xx xx xx - 00 00 00 00 90 00 20 01 477F:0010 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 After Execution 477F:0000 xx xx xx xx xx xx xx xx - 00 00 00 00 90 00 20 01 477F:0010 90 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00** DEPT OF ECE 34 VI SEMESTER

8086 Lab Manual AIeMS

**17) Develop and execute an assembly language program to find the GCD of two 16 bit unsigned integers.**

**.model small**

**.data**

**NUM1 dw 0090h NUM2 dw 0120h GCD dw ?**

**.code**

**MOV AX, @DATA MOV DS, AX MOV AX, NUM1 MOV BX, NUM2 AGAIN: CMP AX, BX**

**JE STOP JB EXCHANGE BACK: MOV DX,00H**

**DIV BX CMP DX, 00H JE STOP MOV AX, DX JMP AGAIN EXCHANGE: XCHG AX, BX**

**JMP BACK STOP: MOV GCD, BX**

**MOV AH, 4CH INT 21H END**

DEPT OF ECE 35 VI SEMESTER

8086 Lab Manual AIeMS

**Description: The factorial of a number is obtained using the equation n!= n x(n-1) x (n-2) x......... or n!= n x (n-1)! Example: 5!= 5x4x3x2x1=120 (decimal)**

**Algorithm:**

**1) Initialize data of type byte in memory location and Data**

**Segment register with appropriate address. 2) Data is loaded into AX and CX registers. 3) CX is decremented and procedure to determine factorial is**

**called. 4) The procedure returns the factorial of the number using**

**the equation given above. 5) The result is then stored in suitable memory location. 6) Terminate the program.**

**RESULT: Before Execution: 477F:0000 xx xx xx xx xx 05 00 xx - xx xx xx xx xx xx xx xx After Execution: 477F:0000 xx xx xx xx xx 05 78 xx - xx xx xx xx xx xx xx xx**

DEPT OF ECE 36 VI SEMESTER

8086 Lab Manual AIeMS

**18) Develop and execute an assembly language program to find the factorial of a 8 bit number.**

**.model small**

**.data**

**X DB 05H RES DB ? .code**

**MOV AX,@DATA MOV DS,AX LEA SI,X LEA DI,RES MOV AX,[SI] MOV CX,AX DEC CX CALL FACT MOV [DI],AX MOV AH,4CH INT 21H FACT PROC NEAR L2: JZ L1**

**MOV BX,CX MUL BX LOOP L2 L1: RET**

**FACT ENDP END**

DEPT OF ECE 37 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm:**

**1) Initialize data of type word in memory location and Data**

**Segment register with appropriate address. 2) Clear the contents of AX and BX registers. 3) Copy the data to Both AX as well as BX. 4) Multiply the number with itself to determine the square. 5) The result may be greater than 16-bits and is loaded into**

**consecutive memory locations from DX-AX register pair. 6) Terminate the program.**

DEPT OF ECE 38 VI SEMESTER

8086 Lab Manual AIeMS

**19) Develop and execute an assembly language program to find the square of a 16 bit number.**

**.model small .data**

**x dw 0FFFFh res db ? .code**

**mov ax,@data mov ds,ax mov ax,00h mov bx,00h mov ax,x mov bx,x mul bx mov res+3,al mov res+2,ah mov res+1,dl mov res,dh mov ah,4ch**

**int 21h end**

**RESULT: Before Execution: 477F:0000 xx xx xx xx xx xx xx xx – FF FF xx xx xx xx xx xx After Execution: 477F:0000 xx xx xx xx xx xx xx xx – FF FF FF FE 00 01 xx xx**

**\**

DEPT OF ECE 39 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm:**

**1) Initialize data of type word in memory location and Data**

**Segment register with appropriate address. 2) Clear the contents of AX and BX registers. 3) Copy the data to Both AX as well as BX. 4) Multiply the number with itself to determine the square. 5) The result is again multiplied by the number to obtain the**

**cube. 6) The result may be greater than 16-bits and is loaded into**

**consecutive memory locations from DX-AX register pair. 7) Terminate the program.**

DEPT OF ECE 40 VI SEMESTER

8086 Lab Manual AIeMS

**20) Develop and execute an assembly language program to find the cube of a 8 bit number.**

**.model small**

**.data**

**x db 0FFh res db ?**

**.code**

**mov ax,@data mov ds,ax mov ax,00h mov bx,00h mov al,x mov bl,x mul bl mul bx mov res+3,al mov res+2,ah mov res+1,dl mov res,dh mov ah,4ch int 21h end**

**RESULT: Before Execution: 477F:0000 xx xx xx xx xx xx xx xx –00 FF xx xx xx xx xx xx After Execution: 477F:0000 xx xx xx xx xx xx xx xx –00 FF 00 FD 02 FF xx xx**

DEPT OF ECE 41 VI SEMESTER

8086 Lab Manual AIeMS

**ALGORITHM:**

**1) Initialise the data segment. 2)Initialise the code segment. 3) Input data is moved to al register from memory location „num‟. 4) Rotate right the contents of al reg. once & check for the carry bit. 5) Carry=1 indicates the no. is odd and hence the message ‟no. is odd‟**

**is displayed, jump to step (7). 6) Else Carry=0 indicates the no. is even and hence the message ‟no. is even‟ is displayed. 7) Exit from DOS. 8) End of program.**

DEPT OF ECE 42 VI SEMESTER

8086 Lab Manual AIeMS

**21) Develop and Execute an Assembly Language Program to find whether The given number is odd or even.**

**.model small .data**

**num db 12h MSG1 db „no. is odd‟ , „$‟ MSG2 db „no. is even‟ , „$‟ .code**

**mov ax, @data mov ds, ax mov al, num ROR al, 01h Jc Loc1 Lea dx, MSG1 Jmp Loc2 Loc1: lea dx, MSG2 Loc2: mov ah, 09h**

**int 21h mov ah,4ch int 21h end**

**Result:**

**Before execution: d ds:0000 00 12 00 00 00 00**

**After execution: d ds:0000 „no. is even‟ .**

DEPT OF ECE 43 VI SEMESTER

8086 Lab Manual AIeMS

**ALGORITHM:**

**1) Initialise the data segment. 2)Initialise the code segment. 3) Input data is moved to al register from memory location „num‟. 4) Rotate left the contents of al reg. once & check for the carry bit. 5) Carry=1 indicates the no. is negative and hence the message ‟negative‟**

**is displayed, jump to step (7). 6) Else Carry=0 indicates the no. is positive and hence the message ‟positive‟ is displayed. 7) Exit from DOS. 8) End of program.**

DEPT OF ECE 44 VI SEMESTER

8086 Lab Manual AIeMS

**22) Develop and Execute an Assembly Language Program to find whether the given number is Positive or Negative.**

**.model small .data**

**num db 80h MSG1 db „positive‟ , „$‟ MSG2 db „negative‟ , „$‟ .code mov ax, @data**

**mov ds, ax mov al, num ROL al, 01h Jc Loc1 Lea dx, MSG1 Jmp Loc2 Loc1: lea dx, MSG2 Loc2: mov ah, 09h**

**int 21h mov ah,4ch int 21h end**

**Result: Before execution: d ds:0000 00 80 00 00 00 00**

**After execution: d ds:0000 „negative‟ .**

DEPT OF ECE 45 VI SEMESTER

8086 Lab Manual AIeMS

**ALGORITHM:**

**1) Initialise the data segment. 2) Initialise the code segment. 3) Initialise „SI‟ with the address of the data variable „num‟. 4) Move the contents of memory location pointed by SI to al. 5) Initialise bx to 00h to store the count value of no. of one‟s in the 5**

**MSB bits of the data. 6) AND the contents of al with E0h to mask the LSB 5 bits & retain the**

**first 3 MSB bits. 7) If the result after AND operation is not zero,then one or two or all**

**three MSB bits are not zero & hence the message ‟Not a 2 out of 5**

**code‟ is displayed indicating that the given data is not a 2out of 5 code. 8) Else, we proceed to check the remaining 5 MSB bits for 2 ones. As a**

**first step count register cx is initialiased to 05h. 9) Original data is taken in ah reg. & rotated right once , checked for**

**carry flag. 10) If carry=1,bx reg. is incremented. 11) Else cx is decremented and step (9) & (10) are repeated until cx value**

**becomes zero. 12) Contents of bx is compared with 02h,If equal, message ‟ 2 out of 5 code‟ is displayed indicating that the given data is 2out of 5 code. 13) Exit from DOS. 14) End of program.**

DEPT OF ECE 46 VI SEMESTER

8086 Lab Manual AIeMS

**23) Develop and Execute an Assembly Language Program to check**

**whether the given number is 2 out of 5 code or not.**

**.model small .data num db 05h**

**msg2 db "2 out of 5 code$" msg1 db "not a 2 out of 5 code$" .code mov ax,@data**

**mov ds,ax mov si,num mov al,[si] mov bx,00h and al,0e0h jnz NOF mov cx,05h mov ah,num rotate: ror ah,01h jnc down inc bx down: dec cx**

**jnz rotate cmp bx,02h je TOF NOF: lea dx, msg2**

**mov ah,09 int 21h jmp exit TOF: lea dx, msg1**

**mov ah,09h int 21h exit: mov ah,4ch**

**int 21h end**

**Result: Before execution: d ds:0000 00 05 00 00 00 00**

**After execution: d ds:0000 „2 out of 5 code‟ .**

DEPT OF ECE 47 VI SEMESTER

8086 Lab Manual AIeMS

**ALGORITHM:**

**1) Initialise the data segment. 2) Initialise the code segment. 3) bx & dx registers are made zero to store the no. of zero‟s & one‟s**

**respectively. 4) Initialise cx to 0008h as the count value. 5) Input data is moved to al register from memory location „num‟. 6) Rotate right the contents of al once & check for carry. 7) If carry=0,increment the zero count reg. bx. 8) Decrement cx,repeat step (6) until cx=0, jump to step(10). 9) If carry=1,increment the one count reg. dx, repeat step(8). 10) Store the count values of zero‟s & one‟s in the initialized memory**

**locations. 11) Exit from DOS. 12) End of program.**

DEPT OF ECE 48 VI SEMESTER

8086 Lab Manual AIeMS

**24) Develop and Execute an Assembly Language Program to count no. of one‟s and zero‟s in the given number.**

**.model small .data num dw 07h**

**ones dw ? zeros db ? .code mov ax,@data**

**mov ds,ax mov dx,00h mov bx,00h mov cx,0008h mov ax,num rotate: ror ax,01h**

**jnc label1 inc dx jmp back label1: inc bx back: dec cx**

**jnz rotate mov ones, dx mov zeros, bx mov ah,4ch int 21h end**

**Result: Before execution: d ds:0000 00 07 00 00 00 00**

**After execution: d ds:0000 00 07 00 05 03 00**

DEPT OF ECE 49 VI SEMESTER

8086 Lab Manual AIeMS

**ALGORITHM:**

**1) Initialise the data segment. 2) Initialise the code segment. 3) Input BCD value is moved to al reg, from the memeory location**

**„num‟. 4) Save the BCD data in ah reg. for further use. 5) Initialise count register cl=04h. 6) Mask the MSB 4 bits to get the remaining 4 LSB bits by ANDing**

**the al with 0fh & store the anded value in bl register. 7) Obtain the original BCD data in al reg. 8) AND it with f0h to mask 4 LSB bits & retian 4 MSB bits. 9) Swap the contents of al. 10) Move the contents of al to cl & 0Ah to dl. 11) Contents of al is multiplied with dl with the 16 bit result stored in the reg. ax. 12) CL is decremented once & repeat (11) until contents of cl=0. 13) Add the contents of al with dl & store the binary value in al to the initialised memory location. 14) Exit from DOS. 15) End of program.**

DEPT OF ECE 50 VI SEMESTER

8086 Lab Manual AIeMS

**25) Develop and Execute an Assembly Language Program to perform the BCD to Binary code conversion**

**.model small .data num db 67**

**result db ? .code mov ax,@data**

**mov ds,ax mov al,num mov ah,al mov cl,04h And al,ofh mov bl,al mov al,ah and al,foh ror/rol al,cl mov ch,00h mov cl,al mov ah,00h mov dl,0Ah back:mul dl**

**loop back add al,bl mov result,al mov ah,4ch Int 21h End**

**Result: Before execution: d ds:0000 00 10 00 00 00 00**

**After execution: d ds:0000 00 10 00 00 0A 00**

**OR Alternatively**

DEPT OF ECE 51 VI SEMESTER

8086 Lab Manual AIeMS

**;PROGRAM TO CONVERT BCD NUMBER TO BINARY NUMBER .model small .data bcd equ 0255h result dw ? .code start:**

**mov ax,@data mov ds,ax mov bx,bcd mov ax,00h mov cx,00h**

**again: cmp bx,00h**

**jz endprg mov al,bl sub al,01h das mov bl,al mov al,bh sbb al,00h das mov bh,al inc cx jmp again**

**endprg: mov result,cx**

**mov ah,4ch int 21h end**

DEPT OF ECE 52 VI SEMESTER

8086 Lab Manual AIeMS

**ALGORITHM:**

**1) Initialise the data segment. 2) Initialise the code segment. 3) Input binary no. is stored in ax reg. 4)bx is initialized to 0Ah & cx to zero. 5) dx is initialized to 00h. 6) Perform 32 bit division where the contents odf dx-ax register is divided by bx ie 0Ah. Remainder is stored in dx & quotient is In ax reg. 7) Push the remainder on to stack. 8) Increment the cx reg. to store the no. of times the devision is performed. 9) Repeat step (5),(6),(7) & (8) until ax register contents becomes zero. 10) Pop the contents of stack(remainder) into dx. 11) Store the result in the initialized memory location.The result is displayed as unpacked BCD no. 12) Repeat step (10),(11) & (12) until contents of cx becomes zero. 13) Exit from DOS. 14) End of program.**

DEPT OF ECE 53 VI SEMESTER

8086 Lab Manual AIeMS

**26) Develop and Execute an Assembly Language Program to perform the Binary to BCD code conversion**

**.model small .data num dw 00ffh z db ? .code mov ax,@data**

**mov ds,ax mov ax,num mov bx,0Ah mov cx,00h repeat:mov dx,00h**

**div bx push dx inc cx cmp ax,00h jnz repeat pop dx mov z,dx dec cx pop dx mov z+1,dx dec cx pop dx mov z+2,dx mov ah,4ch int 21h end**

**Result: Before execution: d ds:0000 00 ff 00 00 00 00**

**After execution: 02 05 05**

DEPT OF ECE 54 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm Step 1: initialize the data segment. Step 2: call the interrupt to read the character from the standard input device (key board) and echoit to the output device (display screen). Step 3: store the character in to memory location. Step 4: end**

DEPT OF ECE 55 VI SEMESTER

8086 Lab Manual AIeMS

**27) PROGRAM TO READ THE CHARACTER WITH ECHO.**

**.MODEL SMALL**

**.DATA**

**KEY DB ?**

**.CODE**

**MOV AX, @DATA MOV DS, AX MOV AH, 01H INT 21H MOV KEY, AL MOV AH, 4CH INT 21H END**

**Result:**

**With echo the result will be stored at the AL register**

DEPT OF ECE 56 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm:**

**Step 1: initialize the data segment Step 2: call the interrupt to read the character from standard input device (key board) without echoing it to the Output device. Step 3: store the character in to memory location. Step 4: end**

DEPT OF ECE 57 VI SEMESTER

8086 Lab Manual AIeMS

**28) PROGRAM TO READ A CHARACTER WITHOUT ECHO**

**.MODEL SMALL**

**.DATA**

**.CODE**

**MOV AX, @DATA MOV DS, AX MOV AH, 08/07H INT 21H MOV AH, 4CH INT 21H END**

**Result:**

**Without echo the result will be stored at the AL register**

DEPT OF ECE 58 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm:**

**Step 1: initialize the data segment Step 2: store the year in to CX register, month in to DH register, date in to DL register. Step 3: call the interrupt for setting the date. Step 4: call the interrupt for reading the date. Step 5: end**

DEPT OF ECE 59 VI SEMESTER

8086 Lab Manual AIeMS

**29) PROGRAM TO SET AND READ SYSTEM DATE**

**.MODEL SMALL**

**.DATA**

**YEAR DW 2009 MONTH DB 04 DATE DB 20**

**.CODE**

**MOV AX, @DATA MOV DS, AX**

**MOV CX, YEAR MOV DH, MONTH MOV DL, DATE MOV AH, 2BH INT 21H MOV AH, 2AH INT 21H MOV AH, 4CH INT 21H END**

**Result:**

**The system date will be displayed at the dos prompt.**

DEPT OF ECE 60 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm:**

**Step 1: initialize the data segment Step 2: call the interrupt for displaying on the console. Step 3: end.**

DEPT OF ECE 61 VI SEMESTER

8086 Lab Manual AIeMS

**30) PROGRAM TO DISPLAY THE CHARACTER ON CONSOLE**

**.MODEL SMALL**

**.DATA CHAR DB “A$”**

**.CODE**

**MOV AX, @DATA MOV DS, AX**

**MOV DL, CHAR MOV DH, 02H INT 21H**

**MOV AH, 4CH INT 21H END**

**Result:**

**The character will be displayed on the dos prompt**

DEPT OF ECE 62 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm:**

**Step 1: initialize the data segment Step 2: call the interrupt for displaying on the console. Step 3: end.**

DEPT OF ECE 63 VI SEMESTER

8086 Lab Manual AIeMS

**31) PROGRAM TO DISPLAY THE STRING ON CONSOLE**

**.MODEL SMALL**

**.DATA STR DB „ELECTRONICS $”**

**.CODE**

**MOV AX, @DATA MOV DS, AX**

**MOV AH, 09H MOV DX, OFFSET STR / LEA DX, STR INT 21H**

**MOV AH, 4CH INT 21H END**

**Result:**

**The string will be displayed on the dos prompt**

DEPT OF ECE 64 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm:**

**Step 1: initialize the data segment Step 2: initialize the buf1. Step 3: read the input to the buf1 by using interrupt subroutine. Step 4: repeat the steps from 2 Step 5: result will be seen at buf1, buf2, buf3 memory locations Step 6: end**

DEPT OF ECE 65 VI SEMESTER

8086 Lab Manual AIeMS

**32) PROGRAM TO READ STRING OF DATA (BUFFERED KEYBOARD INPUT)**

**.MODEL SMALL**

**.DATA**

**BUF1 DB 257 DUP (?) BUF2 DB 257 DUP (?) BUF2 DB 257 DUP (?)**

**.CODE**

**MOV AX, @DATA MOV DS, AX**

**MOV BUF1, 255 MOV DX, OFFSET BUF1 CALL LINE MOV BUF2, 255 MOV DX, OFFSET BUF2 CALL LINE MOV BUF1, 255 MOV DX, OFFSET BUF1 CALL LINE**

**MOV AH, 4CH INT 21H LINE PROC NEAR MOV AH, 0AH INT 21H RET LINE ENDP END**

**Result:**

**Entered data available in the buf1, buf2, buf3 memory location**

DEPT OF ECE 66 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm:**

**Step 1: initialize the data segment Step 2: initialize the counter. Step 3: get the 1st register to AH register. Step 4: increment SI. Step 5: compare with 2nd data Step 6: if the 1st data is larger or equal that data stored in AH register and decrement CX. Step 7: if not get that data to AH register Step 8: AH will be having largest number Step 9: end**

DEPT OF ECE 67 VI SEMESTER

8086 Lab Manual AIeMS

**33) PROGRAM TO FIND LARGEST OF N NUMBERS**

**.MODEL SMALL**

**.DATA**

**DATA1 DB 20H, 15H, 25H, 30H LARGE DB ?**

**.CODE**

**MOV AX, @DATA MOV DS, AX**

**MOV CX, 03H MOV SI, 00H MOV AH, DATA1[SI]**

**BACK: INC SI**

**CMP AH, DATA1[SI] JAE SKIP MOV AH, DATA1 [SI] SKIP: DEC CX**

**JNZ BACK MOV LARGE, AH MOV AH, 4CH INT 21H END**

**Result:**

**30**

DEPT OF ECE 68 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm:**

**Step 1: initialize the data segment Step 2: initialize the counter. Step 3: get the 1st register to AH register. Step 4: increment SI. Step 5: compare with 2nd data Step 6: if the 1st data is smaller or equal that data stored in AH register and decrement CX. Step 7: if not get that data to AH register Step 8: AH will be having smallest number Step 9: end**

DEPT OF ECE 69 VI SEMESTER

8086 Lab Manual AIeMS

**34) PROGRAM TO FIND SMALLEST OF N NUMBERS**

**.MODEL SMALL**

**.DATA**

**DATA1 DB 20H, 15H, 25H, 30H SMALL DB ?**

**.CODE**

**MOV AX, @DATA MOV DS, AX**

**MOV CX, 03H MOV SI, 00H MOV AH, DATA1[SI]**

**BACK: INC SI**

**CMP AH, DATA1[SI] JBE SKIP MOV AH, DATA1[SI] SKIP: DEC CX**

**JNZ BACK MOV SMALL, AH MOV AH, 4CH INT 21H END**

**Result:**

**15**

DEPT OF ECE 70 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm:**

**Step 1: initialize the data segment Step 2: initialize the counter. Step 3: get the 1st data to AL register. Step 4: increment SI. Step 5: compare AL with 2nd data Step 6: if the 1st data is smaller or equal that data stored in AL register and decrement CX. Step 7: if not get that data to AL register and AL will be having smallest number Step 8: decrement the counter until the counter becomes zero Step 9: end**

DEPT OF ECE 71 VI SEMESTER

8086 Lab Manual AIeMS

**35) PROGRAM TO SORT N NUMBERS IN ASCENDING ORDER USING BUBBLE SORT**

**.MODEL SMALL**

**.DATA**

**X DB 10H, 05H, 04H, 12H**

**.CODE**

**MOV AX, @DATA MOV DS, AX**

**MOV BX, 04H DEC BX L3: MOV CX, BX**

**MOV SI, 00H L2: MOV AL, X [SI]**

**INC SI CMP AL, X [SI] JBE L1 XCHG AL, X [SI] MOV X [SI-1], AL L1: LOOP L2**

**DEC BX JNZ L3 MOV AH, 4CH INT 21H END**

**Result:**

**04 05 10 12**

DEPT OF ECE 72 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm:**

**Step 1: initialize the data segment Step 2: initialize the counter. Step 3: get the 1st data to AL register. Step 4: increment SI. Step 5: compare AL with 2nd data Step 6: if the 1st data is larger or equal that data stored in AL register and decrement CX. Step 7: if not get that data to AL register and AL will be having largest number Step 8: decrement the counter until the counter becomes zero Step 9: end**

DEPT OF ECE 73 VI SEMESTER

8086 Lab Manual AIeMS

**36) PROGRAM TO SORT N NUMBERS IN DECENDING ORDER USING BUBBLE SORT**

**.MODEL SMALL**

**.DATA**

**X DB 10H, 05H, 04H, 12H**

**.CODE**

**MOV AX, @DATA MOV DS, AX**

**MOV BX, 04H DEC BX L3: MOV CX, BX**

**MOV SI, 00H L2: MOV AL, X [SI]**

**INC SI CMP AL, X [SI] JAE L1 XCHG AL, X [SI] MOV X [SI-1], AL L1: LOOP L2**

**DEC BX JNZ L3 MOV AH, 4CH INT 21H END**

**Result:**

**12 10 05 04**

DEPT OF ECE 74 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm:**

**Step 1: Initialize the data segment. Step 2: Initialize the extra segment. Step 3: Load offset of source string to SI. Step 4: Load offset destination to DI. Step 5: Move the length of string to CX. Step 6: Clear destination flag to auto increment SI & DI. Step 7: Decrement CX and MOVSB until CX will be zero. Step 8: End .**

DEPT OF ECE 75 VI SEMESTER

8086 Lab Manual AIeMS

**37) PROGRAM TO TRANSFER A STRING FROM SOURCE TO DESTINATION**

**.MODEL SMALL**

**.DATA SRC DB „PROGRAM $‟**

**DST DB ? LEN EQU „$-SRC‟**

**.CODE**

**MOV AX, @DATA MOV DS, AX**

**MOV ES, AX LEA SI, SRC LEA DI, DST MOV CX, LEN CLD REP MOVSB MOV AH, 4CH INT 21H END**

**Result:**

**Before execution 4377:0000 09 00 FC F3 A4 4C CD - 21 00 70 72 6F 67 72 61 4377:0010 6D 24 70 72 6F 67 72 - 61 60 24 70 4C 21 90 80**

**After execution 4377:0000 09 00 FC F3 A4 B4 CD - 21 00 50 52 4F 47 52 41 4377:0010 4D 24 50 52 4F 47 52 - 41 4D 24 50 4C 21 90 80**

DEPT OF ECE 76 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm:**

**Step 1: Initialize the data segment. Step 2: Initialize the extra segment. Step 3: initialize the counter. Step 4: Load offset of source string to SI. Step 5: Load offset destination to DI. Step 6: add SI with CX Step 7: compare SI and DI Step 8: if data bellow/equal goto step 10. Step 9: if not equal exchange the data from SI to DI. Step 10: decrement SI and increment DI Step 11: repeat the steps from step 7 to until counter becomes zero. Step 12: End.**

DEPT OF ECE 77 VI SEMESTER

8086 Lab Manual AIeMS

**38) PROGRAM TO REVERSE A STRING**

**.MODEL SMALL**

**.DATA**

**SRC DB „HELLO $‟ DST DB 5 DUP (?) LEN EQU 05H**

**.CODE**

**MOV AX, @DATA MOV DS, AX**

**MOV ES, AX MOV CX, LEN DEC CX LEA SI, SRC LEA DI, DST ADD SI, CX BACK: CMP SI, DI**

**JBE LAST MOV AH, [SI] MOV AL, [DI] MOV [DI], AH MOV [SI], AL DEC SI INC DI JMP BACK LAST: LEA DX, SRC MOV AH, 09H INT 21H MOV AH, 4CH INT 21H END**

**Result:**

**INPUT = HELLO OUTPUT = OLLEH**

DEPT OF ECE 78 VI SEMESTER

8086 Lab Manual AIeMS

**Algorithm:**

**Step 1: Initialize the data segment. Step 2: Initialize the character in AL register. Step 3: Load offset of source string to SI. Step 4: make BH as 00H Step 5: initialize the counter. Step 6: increment BH and compare AL with contents of SI. Step 7: if it is equal then display the message 1. Step 8: if not equal then display the message 2 Step 9: end**

DEPT OF ECE 79 VI SEMESTER