**CS322 – Lab 3**

Name: P. V. Sriram

Roll No.: 1801CS37

**P1\_1) Write an assembly language to find sum of 10 random numbers**

.model small

.stack 64

.data

arr1 db 05h,03h,07h,01h,09h,02h,06h,08h,04h,0Ah

.code

start:

MOV AX, @DATA ;Load Data in temp register

MOV DS, AX ;Load data into data into Data Segment

LEA SI, arr1 ;Load data into SI

MOV CX, 10 ;Initialize counter

MOV DL, 10 ;DL <- 10

MOV AH, 00h ;AX = 0000

L1:

ADD AH, [SI] ;Add data into accumulator

INC SI ;Increment pointer

DEC CX ;Decrement counter

JNZ L1 ;Loop until CX == 0

MOV BH, AH ;Store the answer

MOV AH,4CH

INT 21H

end start

.end

**Input**

**05H, 03H, 07H, 01H, 09H, 02H, 06H, 08H, 04H, 0AH (Data Segment)**

**Output**

**37H (BH register)**

**P1\_2) Write an assembly language program to find average of the given set of 16 bit number numbers.**

.model small

.stack 64

.data

arr1 dw 0001, 0002, 0003, 0004, 0005

.code

start:

MOV AX, @DATA ;Load Data in temp register

MOV DS, AX ;Load data into data into Data Segment

LEA SI, arr1 ;Load data into SI

MOV CX, 05H ;CX <-05H

MOV DL, 05H ;DL <-05H

MOV AX, 0000 ;AX = 0000

L1:

ADD AX, [SI] ;Add data to accumulator

INC SI ;Increment pointer

INC SI ;Increment pointer

DEC CX ;Decrement pointer

JNZ L1 ;If not zero, Jump to L1

DIV DL ;Divide Acc with DL

MOV BX, AX ;Store value

MOV AH,4CH

INT 21H

end start

.end

**Input**

**0001H, 0002H, 0003H, 0004H, 0005H (Data Segment)**

**Output**

**0003H (BX Register)**

**P1\_3) Write a program to find largest and smallest among an array of numbers and print the difference between them.**

.model small

.stack 64

.data

arr1 db 5,3,7,1,9,2,6,8,4,10

.code

start:

MOV AX, @DATA ;Load Data in temp register

MOV DS, AX ;Load data into data into Data Segment

LEA SI, arr1 ;Load data into SI

MOV CL, 10 ;Initialize counter

MOV AL, [SI] ;Load data into AL

MOV BL, [SI] ;Load data into BL

update\_min:

CMP AL, [SI] ;Compare data to AL

JC update\_max ;If carry, Jump to Update Max

MOV AL, [SI] ;Store data into AL

update\_max:

CMP BL, [SI] ;Compare data to BL

JNC L1 ;If carry, Jump to L1

MOV BL, [SI] ;Store data into BL

L1:

INC SI ;Increment pointer

DEC CX ;Decrement counter

JNZ update\_min ;If not zero, jump to update\_min

SUB BL, AL ;BL <- BL - AL

MOV DL, BL ;Store answer into DL

MOV AH,09H

INT 21H

end start

.end

**Input**

**5,3,7,1,9,2,6,8,4,10 (Data Segment)**

**Output**

**09 (BL Register)**

**P1\_4a) Assembly level program Addition of two 16-bit numbers**

.model small

.stack 64

.data

A DW 9384H

B DW 1845H

SUM DW ?

CARRY DB 00H

.code

START:

MOV AX, @data ;Load Data in temp register

MOV DS, AX ;Load data into data into Data Segment

MOV AX, A ;Move from A into accumulator

ADD AX, B ;Add eith B

JNC SKIP ;If no carry, skip

INC CARRY ;If carry present

SKIP:

MOV SUM, AX ;Store the answer

INT 03H

end start ;Terminate

.end

**Input**

**9384H, 1845H (Data segment)**

**Output**

**ABC9H (AX Register)**

**P1\_4b) Assembly level program to sort the given array of numbers in ascending order**

.model small

.stack 64

.data

A DB 0FFH,70H,90H,60H,0FEH,20H,10H,13H,25H,00H

.code

START :

MOV AX,@DATA ;Load Data in temp register

MOV DS,AX ;Load data into data into Data Segment

MOV CX,0009H ;Initialize counter

BACK:

MOV DX,0009H ;Initialize counter

LEA SI,A ;Initialize source index

BACK1:

MOV AL,[SI] ;Load Data

INC SI ;Increment Pointer

CMP AL,[SI] ;Compare with previous entry

JC SKIP ;If carry, skip

XCHG AL,[SI] ;Exchange two values

DEC SI ;Decrement Pointer

MOV [SI],AL ;Store value

INC SI ;Increment Pointer

SKIP:

DEC DX ;Decrement pointer

JNZ BACK1 ;Loop until DX is zero

LOOP BACK ;Loop until DX is zero

INT 03H

end START

.end

**Input**

**0FFH,70H,90H,60H,0FEH,20H,10H,13H,25H,00H (Data Segment)**

**Output**

**00, 10, 13, 20, 25, 60, 70, 90, FE, FF (Data Segment)**

**P1\_4c) Assembly level program to find the no of even & odd nos. from series of 10 numbers.**

.model small

.stack 64

.data

A DB 10H,15H,25H,16H,17H,19H,23H,77H,47H,34H

.code

START:

MOV AX,@DATA ;Load Data in temp register

MOV DS,AX ;Load data into data into Data Segment

LEA SI,A ;Load valus into SI

MOV BX,0000H ;Initialize BX with 0

MOV CX,000AH ;Initialize pointer

BACK:

MOV AL,[SI] ;Load data into Accumulator

ROR AL,1 ;Rotate right

JC ODD ;If carry, Jump to ODD

INC BL ;Increment BL

JMP NEXT ;Jump to next

ODD:

INC BH ;Increment BH

NEXT:

INC SI ;Increment SI

LOOP BACK ;Loop until pointer is 0

INT 03H

end START

.end

**Input**

**10H,15H,25H,16H,17H,19H,23H,77H,47H,34H**

**Output**

**07 (BH – Odd no.s) 03 (BL – Even No.s)**

**P1\_4d) Assembly level program to check for a Palindrome (single word, 5 letters)**

.MODEL SMALL

.STACK 64

.DATA

A DB 'M','A','D','A','M'

.CODE

START:

MOV AX,@DATA ;Load Data in temp register

MOV DS,AX ;Load data into data into Data Segment

MOV CH,00H ;Answer pointer

LEA SI,A ;Load data into SI

LEA DI,A+04H ;Load data into DI

MOV CL,02H ;CL <- 02H

BACK:

MOV AH,[SI] ;Load data into AH

MOV BH,[DI] ;Load data into BH

CMP AH,BH ;Compare AH, BH

JNZ SKIP ;If not zero skip

INC SI ;Increment pointer

DEC DI ;Decrement pointer

DEC CL ;Decrement pointer

JNZ BACK ;If not zero, jump to back

INC CH ;Increment of answer

SKIP:

INT 03H

END START

.END

**Input**

**'M','A','D','A','M'**

**Output**

**01 (CH – Indicating Palindrome, 0 otherwise)**

**P1\_4e) Assembly level program to convert an 8 bit BCD number into hexadecimal number**

.model small

.stack 64

.data

a db 25h

.code

MOV AX,@DATA

MOV DS,AX

LEA SI,A

LEA DI,A+100H

MOV BL, [SI]; BL <- [SI]

AND BL, 0Fh; BL = BL AND 0F

MOV AL, [SI]; AL <- [SI]

AND AL, 0F0h; BL = AL AND F0

MOV CL, 04h; CL = 04

ROR AL, CL; Rotate AL

MOV DL, 0Ah; DL = 0A

MUL DL; AX = AL \* DL

ADD AL, BL; AL = AL + BL

MOV [DI], AL; [DI] <- AL

HLT; End of Program

end

.end

**Input**

**25H (Data Segment)**

**Output**

**19H (AH)**

**P1\_4f) Assembly level program to find the factorial of a number**

.model small

.stack 64

.data

a dw 0004h

.code

START:

MOV AX,@DATA

MOV DS,AX

LEA SI,A

LEA DI,A+100H

MOV CX, [SI]; CX <- [0500]

MOV AX, 0001; AX <- 0001

MOV DX, 0000; DX <- 0000

L1:

MUL CX; DX:AX <- AX \* CX

DEC CX

JNZ L1

MOV [DI], AX; [0600]<-AX

MOV [DI + 1], DX; [0601]<-DX

HLT; Stop Execution

end start

.end

**Input**

**0004H**

**Output**

**24H (AX)**