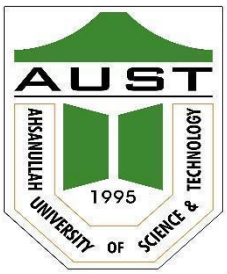

Ahsanullah University of Science & Technology

Department of Computer Science & Engineering

Course No : **CSE2214**
Course Title : **Assembly Language Programming Sessional**
Assignment No : **09**
Date of Performance : **25.02.2021**
Date of Submission : **04.03.2021**

Submitted To : **Ms. Moumita Choudhury & Ms. Tanjila Broti**

Submitted By-
Group : **A_2**
Name : **Minhajul Islam Jim**
Id : **17.01.04.001**
Section : **A**

Question No: 01
Question: Write a program that lets the user enter time in seconds, up to 65535, and outputs the time as hours, minutes, and seconds.
Answer:

```
.MODEL SMALL

.STACK 100H

.DATA

 MSG1 DB 'Enter the time in seconds up to 65535 = $\'

 MSG2 DB 0DH,0AH,'The time in hh:mm:ss format is = $\'

 SEPARATOR DB ' : $\'

.CODE

 MAIN PROC

 MOV AX,@DATA ;initialize DS

 MOV DS,AX

 LEA DX,MSG1 ;load and display the string MSG1

 MOV AH,9

 INT 21H

 CALL INDEC ;call the procedure INDEC

 PUSH AX ;puah AX onto the STACK

 LEA DX,MSG2 ;load and display the string MSG2

 MOV AH,9

 INT 21H

 POP AX ;pop a value from STACK into AX

 XOR DX,DX ;clear DX

 MOV CX,3600 ;set CX=3600

 DIV CX ;set AX=DX:AX\\CX , DX=DX:AX%CX

 CMP AX,10 ;compare AX with 10
```

```
JGE HOURS ;jump to label HOURS if AX>=10
PUSH AX ;push AX onto the STACK
MOV AX,0 ;set AX=0
CALL OUTDEC ;call the procedure OUTDEC
POP AX ;pop a value from STACK into AX
HOURS:
CALL OUTDEC ;call the procedure OUTDEC
MOV AX,DX ;set AX=DX
PUSH AX ;push AX onto the STACK
LEA DX,SEPARATOR ;load and display the string SEPARATOR
MOV AH,9
INT 21H
POP AX ;pop a value from STACK into AX
XOR DX,DX ;clear DX
MOV CX,60 ;set CX=60
DIV CX ;set AX=DX:AX\\CX , DX=DX:AX%CX
CMP AX,10 ;compare AX with 10
JGE MINUTES ;jump to label MINUTES if AX>=10
PUSH AX ;push AX onto the STACK
MOV AX,0 ;set AX=0
CALL OUTDEC ;call the procedure OUTDEC
POP AX ;pop a value from STACK into AX
MINUTES:
CALL OUTDEC ;call the procedure OUTDEC
MOV BX,DX ;set BX=DX
LEA DX,SEPARATOR ;load and display the string SEPARATOR
MOV AH,9
```

```
INT 21H
MOV AX,BX ;set AX=BX
CMP AX,10 ;compare AX with 10
JGE SECONDS ;jump to label SECONDS if AX>=10
PUSH AX ;push AX onto the STACK
MOV AX,0 ;set AX=0
CALL OUTDEC ;call the procedure OUTDEC
POP AX ;pop a value from STACK into AX
SECONDS:
CALL OUTDEC ;call the procedure OUTDEC
MOV AH,4CH ;return 0
INT 21H
MAIN ENDP

;Procedure Definitions: INDEC
;this procedure will read a number in decimal form
;input : none
;output : store binary number in AX
;uses : MAIN
INDEC PROC
 PUSH BX ;push BX onto the STACK
 PUSH CX ;push CX onto the STACK
 PUSH DX ;push DX onto the STACK
 JMP READ ;jump to label READ
 SKIP_BACKSPACE:
 MOV AH,2 ;set output function
```

```
MOV DL,20H ;set DL=' \'
INT 21H ;print a character
READ:
XOR BX,BX ;clear BX
XOR CX,CX ;clear CX
XOR DX,DX ;clear DX
MOV AH,1 ;set input function
INT 21H ;read a character
CMP AL,"-" ;compare AL with "-"
JE MINUS ;jump to label MINUS if AL="-"
CMP AL,"+" ;compare AL with "+"
JE PLUS ;jump to label PLUS if AL="+"
JMP SKIP_INPUT ;jump to label SKIP_INPUT
MINUS:
MOV CH,1 ;set CH=1
INC CL ;set CL=CL+1
JMP INPUT ;jump to label INPUT
PLUS:
MOV CH,2 ;set CH=2
INC CL ;set CL=CL+1
INPUT:
MOV AH,1 ;set input function
INT 21H ;read a character
SKIP_INPUT:
CMP AL,0DH ;compare AL with CR
JE JUMP_TO_END_INPUT ;jump to label JUMP_TO_END_INPUT
CMP AL,8H ;compare AL with 8H
```

```
 JNE NOT_BACKSPACE ;jump to label NOT_BACKSPACE if AL!=8
 CMP CH,0 ;compare CH with 0
 JNE CHECK_REMOVE_MINUS ;jump to label CHECK_REMOVE_MINUS if
CH!=0
 CMP CL,0 ;compare CL with 0
 JE SKIP_BACKSPACE ;jump to label SKIP_BACKSPACE if CL=0
 JMP MOVE_BACK ;jump to label MOVE_BACK
 JUMP_TO_END_INPUT:
 JMP END_INPUT ;jump to label END_INPUT
 CHECK_REMOVE_MINUS:
 CMP CH,1 ;compare CH with 1
 JNE CHECK_REMOVE_PLUS ;jump to label CHECK_REMOVE_PLUS if CH!=1
 CMP CL,1 ;compare CL with 1
 JE REMOVE_PLUS_MINUS ;jump to label REMOVE_PLUS_MINUS if CL=1
 CHECK_REMOVE_PLUS:
 CMP CL,1 ;compare CL with 1
 JE REMOVE_PLUS_MINUS ;jump to label REMOVE_PLUS_MINUS if CL=1
 JMP MOVE_BACK ;jump to label MOVE_BACK
 REMOVE_PLUS_MINUS:
 MOV AH,2 ;set output function
 MOV DL,20H ;set DL=\' \'
 INT 21H ;print a character
 MOV DL,8H ;set DL=8H
 INT 21H ;print a character
 JMP READ ;jump to label READ
 MOVE_BACK:
 MOV AX,BX ;set AX=BX
```

```
MOV BX,10 ;set BX=10
DIV BX ;set AX=AX/BX
MOV BX,AX ;set BX=AX
MOV AH,2 ;set output function
MOV DL,20H ;set DL=\' \'
INT 21H ;print a character
MOV DL,8H ;set DL=8H
INT 21H ;print a character
XOR DX,DX ;clear DX
DEC CL ;set CL=CL-1
JMP INPUT ;jump to label INPUT
NOT_BACKSPACE:
INC CL ;set CL=CL+1
CMP AL,30H ;compare AL with 0
JL ERROR ;jump to label ERROR if AL<0
CMP AL,39H ;compare AL with 9
JG ERROR ;jump to label ERROR if AL>9
AND AX,000FH ;convert ascii to decimal code
PUSH AX ;push AX onto the STACK
MOV AX,10 ;set AX=10
MUL BX ;set AX=AX*BX
MOV BX,AX ;set BX=AX
POP AX ;pop a value from STACK into AX
ADD BX,AX ;set BX=AX+BX
JC ERROR
CMP CL,5
JG ERROR
```

```
 JMP INPUT ;jump to label @INPUT
 ERROR:
 MOV AH,2 ;set output function
 MOV DL,7H ;set DL=7H
 INT 21H ;print a character
 XOR CH,CH ;clear CH
 CLEAR:
 MOV DL,8H ;set DL=8H
 INT 21H ;print a character
 MOV DL,20H ;set DL=\' \'
 INT 21H ;print a character
 MOV DL,8H ;set DL=8H
 INT 21H ;print a character
 LOOP CLEAR ;jump to label CLEAR if CX!=0
 JMP READ ;jump to label READ
 END_INPUT:
 CMP CH,1 ;compare CH with 1
 JNE EXIT ;jump to label EXIT if CH!=1
 NEG BX ;negate BX
 EXIT:
 MOV AX,BX ;set AX=BX
 POP DX ;pop a value from STACK into DX
 POP CX ;pop a value from STACK into CX
 POP BX ;pop a value from STACK into BX
 RET ;return control to the calling procedure
INDEC ENDP
;Procedure Definitions: OUTDEC
```

```
;this procedure will display a decimal number
;input : AX
;output : none
;uses : MAIN
OUTDEC PROC
 PUSH BX ;push BX onto the STACK
 PUSH CX ;push CX onto the STACK
  PUSH DX ;push DX onto the STACK
 CMP AX,0 ;compare AX with 0
 JGE START ;jump to label START if AX>=0
 PUSH AX ;push AX onto the STACK
 MOV AH,2 ;set output function
 MOV DL,"-" ;set DL='-'
 INT 21H ;print the character
 POP AX ;pop a value from STACK into AX
 NEG AX ;take 2\'s complement of AX
 START:
 XOR CX,CX ;clear CX
 MOV BX,10 ;set BX=10
 OUTPUT:
 XOR DX,DX ;clear DX
 DIV BX ;divide AX by BX
 PUSH DX ;push DX onto the STACK
 INC CX ;increment CX
 OR AX, AX ;take OR of Ax with AX
 JNE OUTPUT ;jump to label OUTPUT if ZF=0
 MOV AH,2 ;set output function
```

```
DISPLAY:
POP DX ;pop a value from STACK to DX
OR DL, 30H ;convert decimal to ascii code
INT 21H ;print a character
LOOP DISPLAY ;jump to label @DISPLAY if CX!=0
POP DX ;pop a value from STACK into DX
POP CX ;pop a value from STACK into CX
POP BX ;pop a value from STACK into BX
RET ;return control to the calling procedure
OUTDEC ENDP
END MAIN
```

Question No: 02 Question: Write a program to find the greatest common divisor (GCD) of two integers M and N, according to the following algorithm: a. Divide M by N, getting quotient Q and remainder R. b. If R = 0 then stop. N is the GCD of M and N. c. If R <> 0 replace M by N, N by R, and repeat step 1

Answer:

```
.MODEL SMALL
.STACK 100H
.DATA
MSG1 DB 'Enter the value of M = $'
MSG2 DB 0DH,0AH,'Enter the value of N = $'
MSG3 DB 0DH,0AH,'The GCD of M and N is = $'
.CODE
MAIN PROC
MOV AX,@DATA ;initialize DS
MOV DS,AX
LEA DX,MSG1 ;load and display the string MSG1
```

```
MOV AH,9
INT 21H
CALL INDEC ;call the procedure INDEC
PUSH AX ;push AX onto the STACK
LEA DX,MSG2 ;load and display the string MSG2
MOV AH,9
INT 21H
CALL INDEC ;call the procedure INDEC
MOV BX,AX ;set BX=AX
POP AX ;pop a value from STACK into AX

REPEAT:
XOR DX,DX ;clear DX
DIV BX ;set AX=DX:AX\BX , AX=DX:AX%BX
CMP DX,0 ;compare DX with 0
JE END_LOOP ;jump to label END_LOOP if CX=0
MOV AX,BX ;set AX=BX
MOV BX,DX ;set BX=DX
JMP REPEAT ;jump to label REPEAT
END_LOOP:
LEA DX,MSG3 ;load and display the string MSG3
MOV AH,9
INT 21H
MOV AX,BX ;set AX=BX
CALL OUTDEC ;call the procedure OUTDEC
MOV AH,4CH ;return 0
INT 21H
```

```
 MAIN ENDP
;Procedure Definition: INDEC
;this procedure will read a number indecimal form
;input : none
;output : store binary number in AX
;uses : MAIN
INDEC PROC
 PUSH BX ;push BX onto the STACK
 PUSH CX ;push CX onto the STACK
 PUSH DX ;push DX onto the STACK
 JMP READ ;jump to label READ
 SKIP_BACKSPACE:
 MOV AH,2 ;set output function
 MOV DL,20H ;set DL=' '
 INT 21H ;print a character
 READ:
 XOR BX,BX ;clear BX
 XOR CX,CX ;clear CX
 XOR DX,DX ;clear DX
 MOV AH,1 ;set input function
 INT 21H ;read a character
 CMP AL,"-" ;compare AL with "-"
 JE MINUS ;jump to label MINUS if AL="-"
 CMP AL,"+" ;compare AL with "+"
 JE PLUS ;jump to label PLUS if AL="+"
 JMP SKIP_INPUT ;jump to label SKIP_INPUT
 MINUS:
```

```
MOV CH,1 ;set CH=1
INC CL ;set CL=CL+1
JMP INPUT ;jump to label INPUT

PLUS:
MOV CH,2 ;set CH=2
INC CL ;set CL=CL+1
INPUT:
MOV AH,1 ;set input function
INT 21H ;read a character
SKIP_INPUT:
CMP AL,0DH ;compare AL with CR
JE END_INPUT ;jump to label END_INPUT
CMP AL,8H ;compare AL with 8H
JNE NOT_BACKSPACE ;jump to label NOT_BACKSPACE if AL!=8
CMP CH,0 ;compare CH with 0
JNE CHECK_REMOVE_MINUS ;jump to label CHECK_REMOVE_MINUS if CH!=0

CMP CL,0 ;compare CL with 0
JE SKIP_BACKSPACE ;jump to label SKIP_BACKSPACE if CL=0
JMP MOVE_BACK ;jump to label MOVE_BACK
CHECK_REMOVE_MINUS:
CMP CH,1 ;compare CH with 1
JNE CHECK_REMOVE_PLUS ;jump to label CHECK_REMOVE_PLUS if CH!=1
CMP CL,1 ;compare CL with 1
JE REMOVE_PLUS_MINUS ;jump to label REMOVE_PLUS_MINUS if CL=1
CHECK_REMOVE_PLUS:
```

```
CMP CL,1 ;compare CL with 1
JE REMOVE_PLUS_MINUS ;jump to label REMOVE_PLUS_MINUS if CL=1
JMP MOVE_BACK ;jump to label MOVE_BACK
REMOVE_PLUS_MINUS:
MOV AH,2 ;set output function
MOV DL,20H ;set DL=' '
INT 21H ;print a character
MOV DL,8H ;set DL=8H
INT 21H ;print a character
JMP READ ;jump to label READ

MOVE_BACK:
MOV AX,BX ;set AX=BX
MOV BX,10 ;set BX=10
DIV BX ;set AX=AX/BX
MOV BX,AX ;set BX=AX
MOV AH,2 ;set output function
MOV DL,20H ;set DL=' '
INT 21H ;print a character
MOV DL,8H ;set DL=8H
INT 21H ;print a character
XOR DX,DX ;clear DX
DEC CL ;set CL=CL-1
JMP INPUT ;jump to label @INPUT
NOT_BACKSPACE:
INC CL ;set CL=CL+1
CMP AL,30H ;compare AL with 0
```

```
JL ERROR ;jump to label ERROR if AL<0
CMP AL,39H ;compare AL with 9
JG ERROR ;jump to label ERROR if AL>9
AND AX,000FH ;convert ascii to decimal code
PUSH AX ;push AX onto the STACK
MOV AX,10 ;set AX=10
MUL BX ;set AX=AX*BX
MOV BX,AX ;set BX=AX
POP AX ;pop a value from STACK into AX
ADD BX,AX ;set BX=AX+BX
JS ERROR ;jump to label ERROR if SF=1
JMP INPUT ;jump to label INPUT
ERROR:
MOV AH,2 ;set output function
MOV DL,7H ;set DL=7H
INT 21H ;print a character
XOR CH,CH ;clear CH
CLEAR:
MOV DL,8H ;set DL=8H
INT 21H ;print a character
MOV DL,20H ;set DL=' '
INT 21H ;print a character
MOV DL,8H ;set DL=8H
INT 21H ;print a character
LOOP CLEAR ;jump to label CLEAR if CX!=0
JMP READ ;jump to label READ
END_INPUT:
```

```
 CMP CH,1 ;compare CH with 1
 JNE EXIT ;jump to label EXIT if CH!=1
 NEG BX ;negate BX
 EXIT:
 MOV AX,BX ;set AX=BX
 POP DX ;pop a value from STACK into DX
 POP CX ;pop a value from STACK into CX
 POP BX ;pop a value from STACK into BX
 RET ;return control to the calling procedure
INDEC ENDP
;Procedure Definition: OUTDEC
;this procedure will display a decimal number
;input : AX
;output : none
;uses : MAIN
OUTDEC PROC
 PUSH BX ;push BX onto the STACK
 PUSH CX ;push CX onto the STACK
 PUSH DX ;push DX onto the STACK
 CMP AX,0 ;compare AX with 0
 JGE START ;jump to label START if AX>=0
 PUSH AX ;push AX onto the STACK
 MOV AH,2 ;set output function
 MOV DL,"-" ;set DL='-'
 INT 21H ;print the character
 POP AX ;pop a value from STACK into AX
 NEG AX ;take 2's complement of AX
```

```
START:
XOR CX,CX ;clear CX
MOV BX,10 ;set BX=10
OUTPUT:
XOR DX,DX ;clear DX
DIV BX ;divide AX by BX
PUSH DX ;push DX onto the STACK
INC CX ;increment CX
OR AX, AX ;take OR of Ax with AX
JNE OUTPUT ;jump to label OUTPUT if ZF=0
MOV AH,2 ;set output function
DISPLAY:
POP DX ;pop a value from STACK to DX
OR DL,30H ;convert decimal to ascii code
INT 21H ;print a character
LOOP DISPLAY ;jump to label DISPLAY if CX!=0
POP DX ;pop a value from STACK into DX
POP CX ;pop a value from STACK into CX
POP BX ;pop a value from STACK into BX
RET ;return control to the calling procedure
OUTDEC ENDP
END MAIN
```