**Assembly Level Program 1a – Binary Search**

Write an Assembly Level Program to search a key element in a list of ‘n’ 16-bit numbers using the Binary Search Algorithm.

**Program**

.model SMALL

.data

ARRAY dW 1234h, 2345h, 3456h, 4567h, 5678h, 6789h

LEN dW ($-ARRAY)/2

KEY dW 6789h

STR1 dB 10, 13, 'Element Found at Position '

POS dB ?, 10, 13, '$'

STR2 dB 10, 13, 'Element Not Found!$'

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Clear AX Register

MOV AX, 00h

MOV CX, LEN

MOV DX, KEY

Search:

CMP CX, AX

JB NotFound

MOV BX, CX

ADD BX, AX

SHR BX, 01h ; Divides by 2

MOV SI, BX

SHL SI, 01h ; Multiply with 2

CMP ARRAY[SI], DX

JB newLow

JE Found

CMP BX, 00h

JE NotFound

DEC BX

MOV CX, BX

JMP Search

newLow:

INC BX

MOV AX, BX

JMP Search

Found:

INC BL

MOV POS, BL

LEA DX, STR1

JMP Display

NotFound:

LEA DX, STR2

Display:

; Display Message

MOV AH, 09h

INT 21h

Exit:

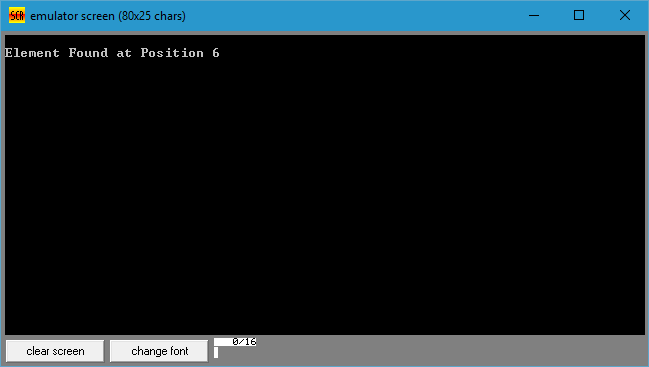
; Terminate the Program

MOV AH, 4Ch

INT 21h

END

**Output**



**Assembly Level Program 1b – Parity Checker**

Read the status of eight input bits from the Logic Controller Interface and display 'FF' if it is the parity of the input read is even; otherwise display 00.

**Program**

.model SMALL

.data

PA EQU 0E400h

PB EQU 0E401h

PC EQU 0E402h

CR EQU 0E403h

CW dB 82h

M1 dB 10, 13, 'Select an 8-bit Number from the Logic Controller Interface...$'

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Set Control Word Format

MOV DX, CR

MOV AL, CW

OUT DX, AL

; Display Message

LEA DX, M1

MOV AH, 09h

INT 21h

; Take Input from Logic Controller Interface

MOV DX, PB

IN AL, DX

; Dummy Operation to Set Flags

OR AL, AL

JPO OddParity

; IF Even Parity

MOV DX, PA

MOV AL, 0FFh

OUT DX, AL

JMP Exit

OddParity:

; IF Odd Parity

MOV DX, PA

MOV AL, 00h

OUT DX, AL

Exit:

; Terminate the Program

MOV AH, 4Ch

INT 21h

END

**Assembly Level Program 2a – Reading & Printing String**

Write 2 ALP modules stored in two different files; one module is to read a character from the keyboard and the other one is to display a character. Use the above two modules to read a string of characters from the keyboard terminated by the carriage return and print the string on the display in the next line.

**PrintCharacter.inc**

PRINTCH MACRO CHAR

MOV DL, CHAR

MOV AH, 02h

INT 21h

ENDM

**ReadCharacter.inc**

READCH MACRO

MOV AH, 01h

INT 21h

ENDM

**Program**

.model SMALL

Include ReadCharacter.inc

Include PrintCharacter.inc

.data

LOC dB 100 DUP(0)

STR1 dB 10, 13, 'Enter a String: $'

STR2 dB 10, 13, 'Entered String is: $'

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Clear Counter Register

MOV CX, 00h

; Display Message

LEA DX, STR1

MOV AH, 09h

INT 21h

; Point SI to First Position of LOC

LEA SI, LOC

Read:

; Call READCH Macro

READCH

; Check if Return/Enter Key was pressed

CMP AL, 0Dh

JE Display

MOV [SI], AL

INC SI

INC CL

JMP Read

Display:

; Display Message

LEA DX, STR2

MOV AH, 09h

INT 21h

; Point SI to First Position of LOC

LEA SI, LOC

Print:

; Call PRINTCH Macro

PRINTCH [SI]

INC SI

Loop Print

Exit:

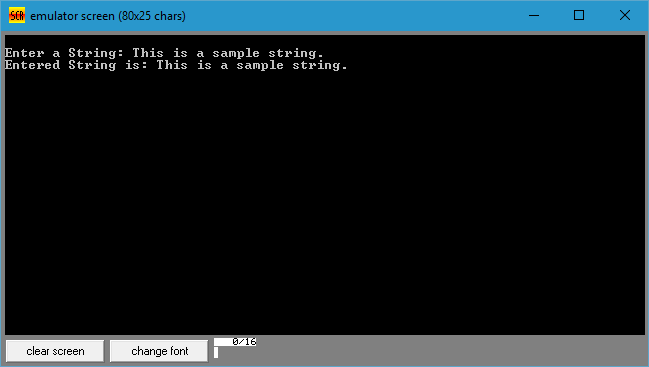
; Terminate the Program

MOV AH, 4Ch

INT 21h

END

**Output**



**Assembly Level Program 2b – BCD Up/Down Counter**

Implement a BCD Up-Down Counter on the Logic Controller Interface.

**Program**

.model SMALL

.data

PA EQU 0E400h

PB EQU 0E401h

PC EQU 0E402h

CR EQU 0E403h

CW dB 80h

COUNT dB 00h

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Set Control Word Format

MOV DX, CR

MOV AL, CW

OUT DX, AL

; Set Counter to 00

MOV DX, PA

MOV AL, 00h

UpCounter:

OUT DX, AL

CALL Delay

INC AL

; Decimal Adjust AL after Addition

DAA

CMP AL, 00h

JNZ UpCounter

; Set Counter to 99

MOV DX, PA

MOV AL, 99h

DownCounter:

OUT DX, AL

CALL Delay

DEC AL

; Decimal Adjust AL after Subtraction

DAS

CMP AL, 99h

JNZ DownCounter

Exit:

; Terminate the Program

MOV AH, 4Ch

INT 21h

Delay PROC NEAR

MOV SI, 0FFFFh

Loop1:

MOV DI, 04FFFh

Loop2:

DEC DI

JNZ Loop2

DEC SI

JNZ Loop1

RET

Delay ENDP

END

**Assembly Level Program 3a – Bubble Sort**

Write an Assembly Level Program to sort a given set of 'n' numbers in ascending and descending orders using the Bubble Sort algorithm.

**Program**

.model SMALL

.data

ARRAY dB 05h, 07h, 06h, 04h, 10h, 09h

LEN dB $-ARRAY

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Clear Counter Register

MOV CX, 00h

MOV CL, LEN

DEC CL

OuterLoop:

MOV BX, CX

; Point SI to First Position of ARRAY

LEA SI, ARRAY

InnerLoop:

MOV AL, [SI]

INC SI

CMP [SI], AL

JAE NoSwap

XCHG [SI], AL

MOV [SI-1], AL

NoSwap:

DEC BX

JNZ InnerLoop

LOOP OuterLoop

Exit:

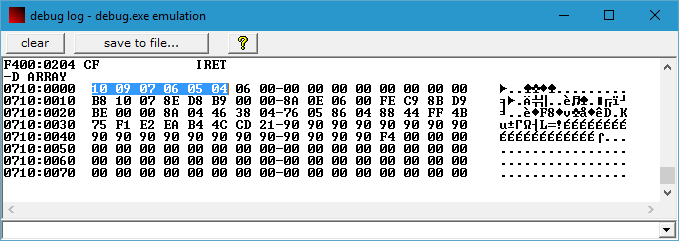
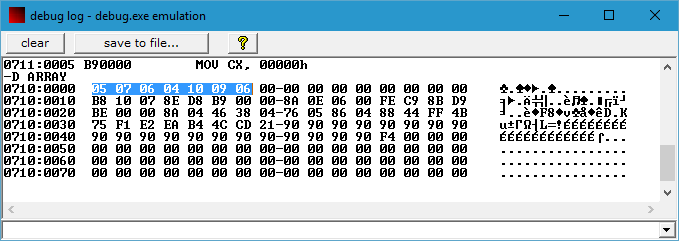
; Terminate the Program

MOV AH, 4Ch

INT 21h

END

**Output**



**Assembly Level Program 3b – Read & Multiply from LCI**

Read the status of two 8-bit inputs (X & Y) from the Logic Controller Interface and display X\*Y.

**Program**

.model SMALL

.data

PA EQU 0E400h

PB EQU 0E401h

PC EQU 0E402h

CR EQU 0E403h

CW dB 80h

M1 dB 10, 13, 'Enter the first 8-bit Number...$'

M2 dB 10, 13, 'Enter the second 8-bit Number...$'

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Set Control Word Format

MOV DX, CR

MOV AL, CW

OUT DX, AL

; Display Message

LEA DX, M1

MOV AH, 09h

INT 21h

; Read Input from Keyboard, without Echo

MOV AH, 08h

INT 21h

; Read Input from Logic Controller Interface

MOV DX, PB

IN AL, DX

MOV BL, AL

; Display Message

LEA DX, M1

MOV AH, 09h

INT 21h

; Read Input from Keyboard, without Echo

MOV AH, 08h

INT 21h

; Read Input from Logic Controller Interface

MOV DX, PB

IN AL, DX

; Multiply BL x AL

MUL BL

; Display First Byte of AX (AL)

MOV DX, PA

OUT DX, AL

CALL Delay

; Display Last Byte of AX (AH)

MOV DX, PA

MOV AL, AH

OUT DX, AL

Exit:

; Terminate the Program

MOV AH, 4Ch

INT 21h

Delay PROC NEAR

MOV SI, 0FFFFh

Loop1:

MOV DI, 04FFFh

Loop2:

DEC DI

JNZ Loop2

DEC SI

JNZ Loop1

RET

Delay ENDP

END

**Assembly Level Program 4a – ASCII Code**

Write an Assembly Level Program to read an alphanumeric character and display its equivalent ASCII code at the center of the screen.

**Program**

.model SMALL

.data

MSG1 dB 10, 13, 'Enter an alphanumeric character: $'

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Clear Screen

MOV AH, 00h

MOV AL, 03h

INT 10h

; Print Message in Data Segment

LEA DX, MSG1

MOV AH, 09h

INT 21h

; Read Character from User

MOV AH, 01h

INT 21h

MOV AH, 00h

MOV BX, 10d

PUSH BX

Conversion:

MOV DX, 00h

DIV BX

PUSH DX

CMP AX, 00h

JNE Conversion

; Set Cursor to Center of the Screen

MOV AH, 02h

MOV BH, 00h

MOV DH, 12d

MOV DL, 39d

INT 10H

Display:

POP DX

CMP DX, 10

JE Exit

ADD DL, 30h

MOV AH, 02h

INT 21h

JMP Display

Exit:

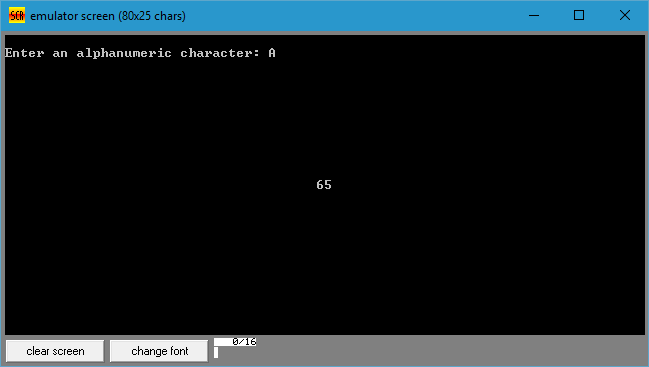
; Terminate the Program

MOV AH, 4Ch

INT 21h

END

**Output**



**Assembly Level Program 4b – Fire & Help Display**

Display messages FIRE and HELP alternately with flickering effects on a 7-segment display interface for a suitable period of time. Ensure a flashing rate that makes it easy to read both the messages (Examiner does not specify these delay values nor is it necessary for the student to compute these values).

**Program**

.model SMALL

.data

PA EQU 0E400h

PB EQU 0E401h

PC EQU 0E402h

CR EQU 0E403h

CW dB 80h

M1 dB 86h, 0AFh, 0F9h, 8Eh ; E, R, I, F

M2 dB 8Ch, 0C7h, 86h, 89h ; P, L, E, H

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Set Control Word Format

MOV DX, CR

MOV AL, CW

OUT DX, AL

Looper:

; Point SI to First Position of M1

LEA SI, M1

CALL Display

CALL Delay

; Point SI to First Position of M2

LEA SI, M2

CALL Display

CALL Delay

; Wait for User Keyboard Input Interrupt

MOV AH, 01h

INT 16h

JZ Looper

Exit:

; Terminate the Program

MOV AH, 4Ch

INT 21h

Display PROC NEAR

; Each Message Contains 4 Bytes

MOV CX, 04h

L2:

; Each Character Contains 8 Bits

MOV BL, 08h

MOV AL, [SI]

L1:

; Rotate AL without Carry

ROL AL, 01h

MOV DX, PB

OUT DX, AL

PUSH AX

; Toggle 0 to Port C

MOV DX, PC

MOV AL, 00h

OUT DX, AL

; Toggle 1 to Port C

MOV DX, PC

MOV AL, 01h

OUT DX, AL

POP AX

DEC BL

JNZ L1

INC SI

LOOP L2

RET

Display ENDP

Delay PROC NEAR

MOV SI, 0FFFFh

Loop1:

MOV DI, 04FFFh

Loop2:

DEC DI

JNZ Loop2

DEC SI

JNZ Loop1

RET

Delay ENDP

END

**Assembly Level Program 5a – Palindrome Checker**

Write an ALP to reverse a given string and check whether it is a palindrome or not.

**Program**

.model SMALL

.data

BUF1 dB 20d

LEN1 dB ?

STR1 dB 20d DUP(0)

RSTR dB 20d DUP(0)

MSG1 dB 10, 13, 'Enter a String: $'

MSG2 dB 10, 13, 'String is a Palindrome!$'

MSG3 dB 10, 13, 'String is not a Palindrome!$'

.code

; Initialize Data & Extra Segment

MOV AX, @DATA

MOV DS, AX

MOV ES, AX

; Display Message

LEA DX, MSG1

MOV AH, 09h

INT 21h

; Read String from Keyboard

LEA DX, BUF1

MOV AH, 0Ah

INT 21h

; Point SI to First Position of STR1

LEA SI, STR1

; Decrement to Skip Reading 0Dh

DEC SI

; Clear and Set Counter Register

MOV CX, 00h

MOV CL, LEN1

; Point DI to Last Position of STR1

ADD SI, CX

MOV DI, SI

; Point SI to First Position of RSTR

LEA SI, RSTR

CopyString:

MOV AL, [DI]

MOV [SI], AL

INC SI

DEC DI

LOOP CopyString

; Point SI to First Position of STR1

LEA SI, STR1

; Point DI to First Position of STR1

LEA DI, RSTR

; Clear and Set Counter Register

MOV CX, 00h

MOV CL, LEN1

; Clear Direction Flag

CLD

; Repeatedly Compare String Byte-by-Byte

REPE CMPSB

JE Palindrome

; Display Message

LEA DX, MSG3

MOV AH, 09h

INT 21h

JMP Exit

Palindrome:

; Display Message

LEA DX, MSG2

MOV AH, 09h

INT 21h

Exit:

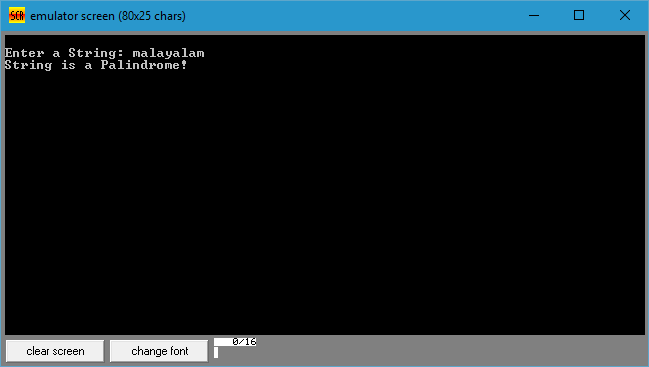
; Terminate the Program

MOV AH, 4Ch

INT 21h

END

**Output**



**Assembly Level Program 5b – Rolling LCD Display**

Assume any suitable message of 12 characters’ length and display it in the rolling fashion on a 7-segment display interface for a suitable period of time. Ensure a flashing rate that makes it easy to read both the messages. (Examiner does not specify these delay values nor is it necessary for the student to compute these values).

**Program**

.model SMALL

.data

PA EQU 0E400h

PB EQU 0E401h

PC EQU 0E402h

CR EQU 0E403h

CW dB 80h

MSG dB 0FFh, 0FFh, 0FFh, 0FFh, 8Eh, 0F9h,

88h, 86h, 89h, 86h, 0C7h, 8Ch

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Set Control Word Format

MOV DX, CR

MOV AL, CW

OUT DX, AL

Looper:

; Message Contains 12 Bytes

MOV CX, 12d

; Point SI to First Position of Message

LEA SI, MSG

RepeatDisplay:

CALL Display

CALL Delay

INC SI

LOOP RepeatDisplay

; Wait for User Keyboard Input Interrupt

MOV AH, 01h

INT 16h

JZ Looper

Exit:

; Terminate the Program

MOV AH, 4Ch

INT 21h

Display PROC NEAR

; Each Character Contains 8 Bits

MOV BL, 08h

MOV AL, [SI]

L1:

; Rotate AL without Carry

ROL AL, 01h

MOV DX, PB

OUT DX, AL

PUSH AX

; Toggle 0 to Port C

MOV DX, PC

MOV AL, 00h

OUT DX, AL

; Toggle 1 to Port C

MOV DX, PC

MOV AL, 01h

OUT DX, AL

POP AX

DEC BL

JNZ L1

RET

Display ENDP

Delay PROC NEAR

MOV SI, 0FFFFh

Loop1:

MOV DI, 04FFFh

Loop2:

DEC DI

JNZ Loop2

DEC SI

JNZ Loop1

RET

Delay ENDP

END

**Assembly Level Program 6a – String Equality**

Write an ALP to read two strings, store them in locations STR1 and STR2. Check whether they are equal or not and display appropriated messages. Also display the length of the stored strings.

**Program**

.model SMALL

.data

BUF1 dB 20d

LEN1 dB ?

STR1 dB 20d DUP(0)

BUF2 dB 20d

LEN2 dB ?

STR2 dB 20d DUP(0)

MSG1 dB 10, 13, 'Enter String 1: $'

MSG2 dB 10, 13, 'Enter String 2: $'

MSG3 dB 10, 13, 'Length of String 1: $'

MSG4 dB 10, 13, 'Length of String 2: $'

MSG5 dB 10, 13, 'Strings are Equal!$'

MSG6 dB 10, 13, 'Strings are Not Equal!$'

.code

; Initialize Data & Extra Segment

MOV AX, @DATA

MOV DS, AX

MOV ES, AX

; Display Message

LEA DX, MSG1

MOV AH, 09h

INT 21h

; Read String from Keyboard

READSTR BUF1

; Display Message

LEA DX, MSG2

MOV AH, 09h

INT 21h

; Read String from Keyboard

READSTR BUF2

; Display Message

LEA DX, MSG3

MOV AH, 09h

INT 21h

; Display Length of First String

MOV AL, LEN1

ADD AL, 30h

MOV AH, 02h

INT 21h

; Display Message

LEA DX, MSG4

MOV AH, 09h

INT 21h

; Display Length of Second String

MOV AL, LEN2

ADD AL, 30h

MOV AH, 02h

INT 21h

; Compare Size of Both Strings

MOV CL, LEN1

CMP CL, LEN2

JNE NotEqual

; Point SI to First Position of STR1

LEA SI, STR1

; Point DI to First Position of STR2

LEA DI, STR2

; Clear and Set Counter Register

MOV CH, 00h

MOV CL, LEN1

; Clear Direction Flag

CLD

; Repeatedly Compare String Byte-by-Byte

REPE CMPSB

JE Equal

NotEqual:

; Display Message

LEA DX, MSG6

MOV AH, 09h

INT 21h

JMP Exit

Equal:

; Display Message

LEA DX, MSG5

MOV AH, 09h

INT 21h

Exit:

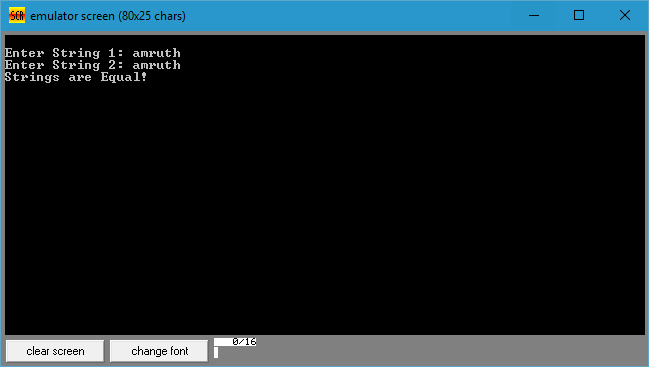
; Terminate the Program

MOV AH, 4Ch

INT 21h

END

**Output**



**Assembly Level Program 6b – BCD Display**

Convert a 16-bit binary value (assumed to be an unsigned integer) to BCD and display it from left to right and right to left for specified number of times on a 7-segment display interface.

**Program**

.model SMALL

.data

PA EQU 0E400h

PB EQU 0E401h

PC EQU 0E402h

CR EQU 0E403h

CW dB 80h

BCD dB 5d DUP(0)

NUM dB 0FFFFh ; 65535 in Hexadecimal

TABLE dB 0C0h, 0FPh, 0A4h, 0B0h, 99h,

92h, 82h, 0F8h, 80h, 98h

LIST dB 0FFh, 0FFh, 0FFh, 0FFh, ?, ?, ?, ?, ?,

0FFh, 0FFh, 0FFh, 0FFh

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Set Control Word Format

MOV DX, CR

MOV AL, CW

OUT DX, AL

; Point SI to First Position of BCD

LEA SI, BCD

MOV AX, NUM

MOV BX, 10000d

CALL CONV

MOV BX, 1000d

CALL CONV

MOV BX,100d

CALL CONV

MOV BX,10d

CALL CONV

MOV [SI], DL

MOV CX, 05h

; Point SI to First Position of BCD

LEA SI, BCD

LEA DI, LIST+8

Loop1:

MOV AL, [SI]

LEA BX, TABLE

XLAT

MOV [DI], AL

INC SI

DEC DI

LOOP Loop1

MOV BH, 10h

LEA DI, LIST

Loop2:

MOV SI, DI

CALL Display

CALL Delay

INC DI

DEC BH

JMP Loop2

MOV BH, 09H

LEA DI, LIST + 8

Loop3:

MOV SI,DI

CALL Display

CALL Delay

DEC DI

DEC BH

JNZ L3

Exit:

; Terminate the Program

MOV AH, 4Ch

INT 21h

Convert PROC NEAR

XOR DX, DX

DIV BX

MOV [SI], AL

MOV AX, DX

INC SI

RET

Convert ENDP

Display PROC NEAR

; Each Message Contains 4 Bytes

MOV CX, 04h

L2:

; Each Character Contains 8 Bits

MOV BL, 08h

MOV AL, [SI]

L1:

; Rotate AL without Carry

ROL AL, 01h

MOV DX, PB

OUT DX, AL

PUSH AX

; Toggle 0 to Port C

MOV DX, PC

MOV AL, 00h

OUT DX, AL

; Toggle 1 to Port C

MOV DX, PC

MOV AL, 01h

OUT DX, AL

POP AX

DEC BL

JNZ L1

INC SI

LOOP L2

RET

Display ENDP

Delay PROC NEAR

MOV SI, 0FFFFh

Loop1:

MOV DI, 04FFFh

Loop2:

DEC DI

JNZ Loop2

DEC SI

JNZ Loop1

RET

Delay ENDP

END

**Assembly Level Program 7a – What Is Your Name?**

Write an Assembly Level Program to read your name from the keyboard and display it at a specified location on the screen in front of the message What is your name?  
You must clear the entire screen before display.

**Program**

.model SMALL

.data

MSG1 dB 10, 13, 'Enter your name: $'

MSG2 dB 10, 13, 'What is your name? $'

ARRAY dB 40h DUP(?)

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Point SI to First Position of ARRAY

MOV SI, ARRAY

; Display Message

LEA DX, MSG1

MOV AH, 09h

INT 21h

ReadName:

; Read Input from Keyboard

MOV AH, 01h

INT 21h

; Copy Input to ARRAY

MOV [SI], AL

INC SI

; Check for Return/Enter Key

CMP AL, 0Dh

JNZ ReadName

; Add Terminating Character at End of String

MOV [SI], '$'

; Clear Screen

MOV AH, 00h

MOV AL, 03h

INT 10h

; Set Cursor to 2x20

MOV AH, 02h

MOV BH, 00h

MOV DH, 2d

MOV DL, 20d

INT 10h

; Display Message

LEA DX, MSG2

MOV AH, 09h

INT 21h

DisplayName:

; Display Message

LEA DX, ARRAY

MOV AH, 09h

INT 21h

Exit:

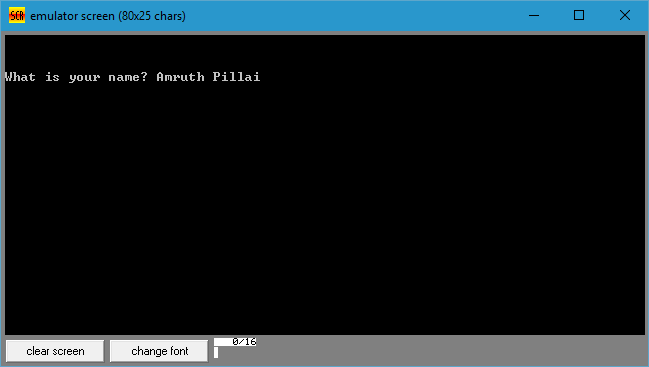
; Terminate the Program

MOV AH, 4Ch

INT 21h

END

**Output**



**Assembly Level Program 7b – 8x3 Keypad**

Scan a 8 x 3 keypad for key closure and to store the code of the key pressed in a memory location or display on screen. Also display row and column numbers of the key pressed.

**Program**

.model SMALL

PRINTMSG MACRO MSG

LEA DX, MSG

MOV AH, 09h

INT 21h

ENDM

PRINTNUM MACRO NUM

MOV DL, NUM

MOV AH, 02h

INT 21h

ENDM

.data

PA EQU 0E400h

PB EQU 0E401h

PC EQU 0E402h

CR EQU 0E403h

CW dB 90h

MSG1 dB 10, 13, 'Row Number: $'

MSG2 dB 10, 13, 'Column Number: $'

MSG3 dB 10, 13, 'Code: $'

ROW dB ?

COL dB ?

KEYS dB '0123456789ABCDEFGHIJKLMN'

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Set Control Word Format

MOV DX, CR

MOV AL, CW

OUT DX, AL

Looper:

; Listen for Key Press at First Row

MOV DX, PC

MOV AL, 01h

OUT DX, AL

; Scan for User Input from Keypad

MOV DX, PA

IN AL, DX

CMP AL, 00h

JNZ FirstRow

; Listen for Key Press at Second Row

MOV DX, PC

MOV AL, 02h

OUT DX, AL

; Scan for User Input from Keypad

MOV DX, PA

IN AL, DX

CMP AL, 00h

JNZ SecondRow

; Listen for Key Press at Third Row

MOV DX, PC

MOV AL, 04h

OUT DX, AL

; Scan for User Input from Keypad

MOV DX, PA

IN AL, DX

CMP AL, 00h

JNZ ThirdRow

JMP Looper

FirstRow:

MOV ROW, 31h

MOV COL, 31h

LEA SI, KEYS

Loop1:

; Divide By 2

SHR AL, 01h

JC Display

INC COL

INC SI

JMP Loop1

SecondRow:

MOV ROW, 32h

MOV COL, 31h

LEA SI, KEYS+8

Loop2:

; Divide By 2

SHR AL, 01h

JC Display

INC COL

INC SI

JMP Loop2

ThirdRow:

MOV ROW, 33h

MOV COL, 31h

LEA SI, KEYS+16

Loop3:

; Divide By 2

SHR AL, 01h

JC Display

INC COL

INC SI

JMP Loop3

Display:

PRINTMSG MSG1

PRINTNUM ROW

PRINTMSG MSG2

PRINTNUM COL

PRINTMSG MSG3

PRINTNUM [SI]

Exit:

MOV AH, 4Ch

INT 21h

END

**Assembly Level Program 8a – Calculate NCR**

Write an Assembly Level Program to compute nCr using recursive procedure. Assume that ‘n’ and ‘r’ are non-negative integers.

**Program**

.model SMALL

.data

N dB 21d

R dB 19d

NCR dW ?

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Clear NCR

MOV AX, 00h

MOV AL, N

MOV BL, R

MOV NCR, 00h

CALL NCRProcedure

Exit:

MOV AH, 4Ch

INT 21h

NCRProcedure PROC NEAR

CMP AX, BX

JE IncrementBy1

CMP BX, 00h

JE IncrementBy1

CMP BX, 01h

JE IncrementByN

DEC AX

CMP AX, BX

JE IncrementByNPlus1

PUSHA

CALL NCRProcedure

POPA

DEC BX

PUSHA

CALL NCRProcedure

POPA

RET

IncrementBy1:

INC NCR

RET

IncrementByN:

ADD NCR, AX

RET

IncrementByNPlus1:

ADD NCR, AX

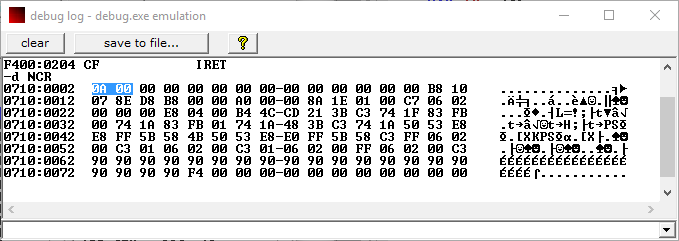
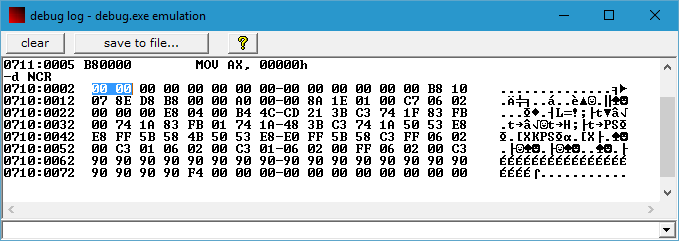
INC NCR

RET

NCRProcedure ENDP

END

**Output**



**Assembly Level Program 8b – Stepper Motor**

Drive a Stepper Motor interface to rotate the motor in specified direction (clockwise or counter-clockwise) by N steps (Direction and N are specified by the examiner). Introduce suitable delay between successive steps. (Any arbitrary value for the delay may be assumed by the student).

**Program**

.model SMALL

.data

PA EQU 0E400h

PB EQU 0E401h

PC EQU 0E402h

CR EQU 0E403h

CW dB 80h

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Set Control Word Format

MOV DX, CR

MOV AL, CW

OUT DX, AL

; 360 Degree Rotation (200x1.8)

MOV CX, 200d

MOV DX, PA

MOV AL, 88h

Rotate:

; Rotate Clockwise

ROR AL, 01h

; Rotate Counter-Clockwise

; ROL AL, 01h

OUT DX, AL

CALL Delay

LOOP Rotate

Exit:

; Terminate the Program

MOV AH, 4Ch

INT 21h

Delay PROC NEAR

MOV SI, 0FFFFh

Loop1:

MOV DI, 04FFFh

Loop2:

DEC DI

JNZ Loop2

DEC SI

JNZ Loop1

RET

Delay ENDP

END

**Assembly Level Program 9a – System Time**

Write an Assembly Level Program to read the current time from the system and display it in the standard format on the screen.

**Program**

.model SMALL

DISPLAY MACRO

; ASCII Adjust after Multiplication

AAM

MOV BX, AX

; Print Higher Nibble

MOV DL, BH

ADD DL, 30h

MOV AH, 02h

INT 21h

; Print Lower Nibble

MOV DL, BL

ADD DL, 30h

MOV AH, 02h

INT 21h

ENDM

COLON MACRO

MOV DL, ':'

MOV AH, 02h

INT 21h

ENDM

.data

MSG1 dB 10, 13, 'The Current System Time is $'

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Display Message

LEA DX, MSG1

MOV AH, 09h

INT 21h

; Interrupt to Fetch System Time

MOV AH, 2Ch

INT 21h

; CH -> Hours

MOV AL, CH

DISPLAY

COLON

; CL -> Minutes

MOV AL, CL

DISPLAY

COLON

; DH -> Seconds

MOV AL, DH

DISPLAY

Exit:

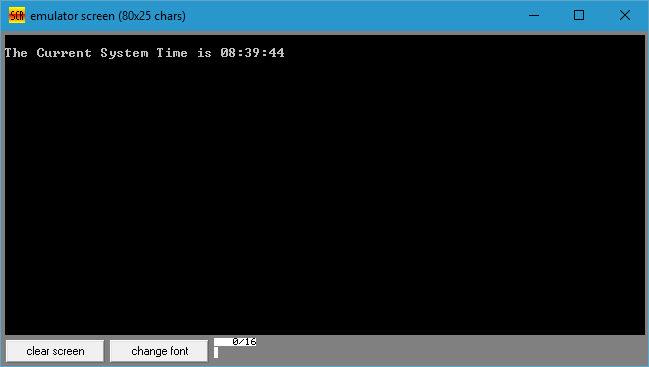
; Terminate the Program

MOV AH, 4Ch

INT 21h

END

**Output**



**Assembly Level Program 9b – Sine Wave**

Generate the Sine Wave using Digital-to-Analog Converter interface (The output of the DAC is to be displayed on the CRO).

**Program**

.model SMALL

.data

PA EQU 0E400h

PB EQU 0E401h

PC EQU 0E402h

CR EQU 0E403h

CW dB 80h

TABLE dB 127, 144, 161, 177, 191, 204, 214, 221, 225,

227, 225, 221, 214, 204, 191, 177, 161, 144,

127, 110, 93, 77, 63, 50, 40, 33, 29, 27,

29, 33, 40, 50, 63, 77, 93, 110, 127

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Set Control Word Format

MOV DX, CR

MOV AL, CW

OUT DX, AL

Looper:

; Number of Values in TABLE

MOV CX, 37d

; Point SI to the First Position of TABLE

LEA SI, TABLE

MOV DX, PA

Repeater:

; Clear Direction Flag

CLD

; Loads [SI] to AL and Auto-Advances SI

LODSB

; Send AL to DAC Interface

OUT DX, AL

LOOP Repeater

; Wait for User Keyboard Input Interrupt

MOV AH, 01h

INT 16h

JZ Looper

Exit:

; Terminate the Program

MOV AH, 4Ch

INT 16h

END

**Assembly Level Program 10a – Decimal Up Counter**

Write an Assembly Level Program to simulate a Decimal Up Counter to display 00 to 99.

**Program**

.model SMALL

.code

; Load 0 (in ASCII) to AL

MOV AL, 30h

FirstLoop:

; Print Lower Digit on Screen

MOV DL, AL

MOV AH, 02h

INT 21h

PUSH AX

; Load 0 (in ASCII) to BL

MOV BL, 30h

SecondLoop:

; Print Higher Digit on Screen

MOV DL, BL

MOV AH, 02h

INT 21h

INC BL

; Get Current Cursor Position

MOV AH, 03h

INT 10h

; Set Cursor to 2nd Column

MOV AH, 02h

MOV DL, 01h

INT 10h

CMP BL, 039h

JLE SecondLoop

; Set Cursor to 1st Column

MOV AH, 02h

MOV DL, 00h

INT 10h

POP AX

INC AL

; Check if Digit has exceeded 9

CMP AL, 039h

JLE FirstLoop

Exit:

; Terminate the Program

MOV AH, 4Ch

INT 21h

END

**Output**



**Assembly Level Program 10b – Half Rectified Sine Wave**

Generate a Half Rectified Sine Wave form using the DAC interface. (The output of the DAC is to be displayed on the CRO).

**Program**

.model SMALL

.data

PA EQU 0E400h

PB EQU 0E401h

PC EQU 0E402h

CR EQU 0E403h

CW dB 80h

TABLE dB 127, 144, 161, 177, 191, 204, 214, 221, 225,

227, 225, 221, 214, 204, 191, 177, 161, 144,

127, 127, 127, 127, 127, 127, 127, 127, 127,

127, 127, 127, 127, 127, 127, 127, 127, 127, 127

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Set Control Word Format

MOV DX, CR

MOV AL, CW

OUT DX, AL

Looper:

; Number of Values in TABLE

MOV CX, 37d

; Point SI to the First Position of TABLE

LEA SI, TABLE

MOV DX, PA

Repeater:

; Clear Direction Flag

CLD

; Loads [SI] to AL and Auto-Advances SI

LODSB

; Send AL to DAC Interface

OUT DX, AL

LOOP Repeater

; Wait for User Keyboard Input Interrupt

MOV AH, 01h

INT 16h

JZ Looper

Exit:

; Terminate the Program

MOV AH, 4Ch

INT 16h

END

**Assembly Level Program 11a – Cursor Movement**

Write an Assembly Level Program to read a pair of input co-ordinates in BCD and move the cursor to the specified location on the screen.

**Program**

.model SMALL

.data

MSGX dB 10, 13 ,'Enter X Coordinates: $'

MSGY dB 10, 13 ,'Enter Y Coordinates: $'

X dB ?

Y dB ?

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Display Message

LEA DX, MSGX

MOV AH,09h

INT 21h

CALL ReadBCD

MOV X, BH

; Display Message

LEA DX, MSGY

MOV AH,09h

INT 21h

CALL ReadBCD

MOV Y, BH

; Clear Screen

MOV AH, 00h

MOV AL, 03h

INT 10h

; Set Cursor Interrupt

MOV AH, 02h

MOV DH, X ; Row Position

MOV DL, Y ; Column Position

MOV BH, 00h ; Page Number

INT 10h

; Direct Console Output

MOV DL, '-'

MOV AH, 06h

INT 21h

Exit:

; Terminate the Program

MOV AH, 4Ch

INT 21h

ReadBCD PROC NEAR

; Read 1st Digit from User

MOV AH, 01h

INT 21h

MOV BH, AL

; Read 2nd Digit from User

MOV AH, 01h

INT 21h

MOV BL, AL

MOV CL, 04h

; Convert ASCII to BCD

SUB BH, 30h

SUB BL, 30h

SHL BH, CL

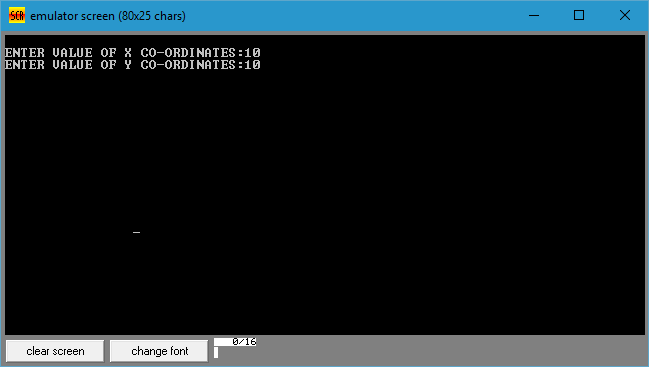
OR BH, BL

RET

ReadBCD ENDP

END

**Output**



**Assembly Level Program 11b – Fully Rectified Sine Wave**

Generate a Fully Rectified Sine waveform using the DAC interface. (The output of the DAC is to be displayed on the CRO).

**Program**

.model SMALL

.data

PA EQU 0E400h

PB EQU 0E401h

PC EQU 0E402h

CR EQU 0E403h

CW dB 80h

TABLE dB 127, 144, 161, 177, 191, 204, 214, 221, 225, 227,

225, 221, 214, 204, 191, 177, 161, 144, 127

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Set Control Word Format

MOV DX, CR

MOV AL, CW

OUT DX, AL

Looper:

; Number of Values in TABLE

MOV CX, 19d

; Point SI to the First Position of TABLE

LEA SI, TABLE

MOV DX, PA

Repeater:

; Clear Direction Flag

CLD

; Loads [SI] to AL and Auto-Advances SI

LODSB

; Send AL to DAC Interface

OUT DX, AL

LOOP Repeater

; Wait for User Keyboard Input Interrupt

MOV AH, 01h

INT 16h

JZ Looper

Exit:

; Terminate the Program

MOV AH, 4Ch

INT 16h

END

**Assembly Level Program 12a – File Handling**

Write an Assembly Level Program to create a file (input file) and to delete an existing file.

**CreateFile.asm**

.model SMALL

.data

FNAME dB 'SampleFile.txt', 00h

SUCCESS dB 10, 13, 'File has been created successfully!$'

FAILURE dB 10, 13, 'An Error Occured during File Creation!$'

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Set File Attribute

MOV CX, 20h

; Interrupt to Create a File

LEA DX, FNAME

MOV AH, 3Ch

INT 21h

JC ErrorOccured

; Display Success Message

LEA DX, SUCCESS

MOV AH, 09h

INT 21h

JMP Exit

ErrorOccured:

; Display Error Message

LEA DX, FAILURE

MOV AH, 09h

INT 21h

Exit:

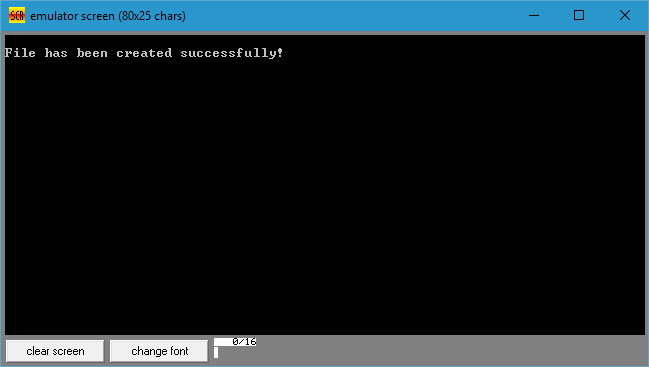
; Terminate the Program

MOV AH, 4Ch

INT 21h

END

**Output**



**DeleteFile.am**

.model SMALL

.data

FNAME dB 'SampleFile.txt', 00h

SUCCESS dB 10, 13, 'File has been deleted successfully!$'

FAILURE dB 10, 13, 'An Error Occured during File Deletion!$'

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Set File Attribute

MOV CX, 20h

; Interrupt to Delete a File

LEA DX, FNAME

MOV AH, 41h

INT 21h

JC ErrorOccured

; Display Success Message

LEA DX, SUCCESS

MOV AH, 09h

INT 21h

JMP Exit

ErrorOccured:

; Display Error Message

LEA DX, FAILURE

MOV AH, 09h

INT 21h

Exit:

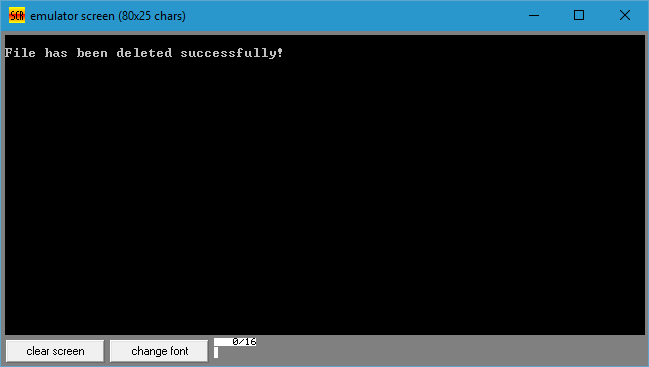
; Terminate the Program

MOV AH, 4Ch

INT 21h

END

**Output**



**Assembly Level Program 12b – Elevator Interface**

**Drive an elevator interface in the following way:**

1. Initially the elevator should be in the ground floor, with all requests in OFF state.
2. When a request is made from a floor, the elevator should move to that floor, wait there for a couple of seconds (approximately), and then come down to ground floor and stop. If some requests occur during going up or coming down, they should be ignored.

**Program**

.model SMALL

.data

PA EQU 0E400h

PB EQU 0E401h

PC EQU 0E402h

CR EQU 0E403h

CW dB 82h

CLEAR dB 0E0h, 0D0h, 0B0h, 70h

.code

; Initialize Data Segment

MOV AX, @DATA

MOV DS, AX

; Set Control Word Format

MOV DX, CR

MOV AL, CW

OUT DX, AL

; Move to Ground Floor Initially

MOV DX, PA

MOV AL, 0Fh

OUT DX, AL

; Point SI to First Position of CLEAR Table

LEA SI, CLEAR

NoRequest:

CALL Request

JZ NoRequest

SHR AL, 01h

JNC GroundFloor

SHR AL, 01h

JNC FirstFloor

SHR AL, 01h

JNC SecondFloor

JMP ThirdFloor

GroundFloor:

CALL Delay

CALL Reset

JMP Exit

FirstFloor:

MOV CX, 03h

LEA SI, CLEAR+1

CALL MoveUp

CALL Delay

CALL Reset

MOV CX, 03h

CALL MoveDown

JMP Exit

SecondFloor:

MOV CX, 06h

LEA SI, CLEAR+2

CALL MoveUp

CALL Delay

CALL Reset

MOV CX, 06h

CALL MoveDown

JMP Exit

ThirdFloor:

MOV CX, 09h

LEA SI, CLEAR+3

CALL MoveUp

CALL Delay

CALL Reset

MOV CX, 09h

CALL MoveDown

Exit:

MOV AH, 4Ch

INT 21h

Request PROC NEAR

; Wait for Key Press from User

MOV DX, PB

IN AL, DX

; Logical AND with Lower Nibble of AL

AND AL, 0Fh

CMP AL, 0Fh

RET

Request ENDP

Reset PROC NEAR

PUSH AX

MOV DX, PA

; Fetch Value from CLEAR Table

MOV AL, [SI]

OUT DX, AL

POP AX

RET

Reset ENDP

MoveUp PROC NEAR

MOV AL, 0F0H

MOV DX, PB

GoUp:

OUT DX, AL

CALL Delay

INC AL

Loop GoUp

OUT DX, AL

RET

MoveUp ENDP

MoveDown PROC NEAR

MOV DX, PB

GoDown:

OUT DX, AL

CALL Delay

DEC AL

Loop GoDown

OUT DX, AL

RET

MoveDown ENDP

Delay PROC NEAR

MOV SI, 0FFFFh

Loop1:

MOV DI, 04FFFh

Loop2:

DEC DI

JNZ Loop2

DEC SI

JNZ Loop1

RET

Delay ENDP

END