

**Submitted by:** Muhammad Raffey

**Submitted to:** Sir Faizan

**Sap ID:** 70153209

**Department:** CS

**Section:** 3E

**Assignment no:** 2

# **Question No.1**

Write an assembly program to convert all hexadecimal numbers present in arr1 into binary.

# **Solution**

**.model small**

**.stack 100h**

**.data**

**arr1 dw 3FDEh, 1A2Bh**

**msg db "Binary Output:$"**

**newline db 13,10,'$'**

**.code**

**main proc**

**mov ax, @data**

**mov ds, ax**

**lea dx, msg**

**mov ah, 09h**

**int 21h**

**lea dx, newline**

**mov ah, 09h**

**int 21h**

**mov cx, 2**

**mov si, 0**

**next\_word:**

**mov bx, arr1[si]**

**mov ch, 16**

**bit\_loop:**

**rol bx, 1**

**mov dl, '0'**

**jc set\_one**

**jmp print\_bit**

**set\_one:**

**mov dl, '1'**

**print\_bit:**

**mov ah, 02h**

**int 21h**

**dec ch**

**jnz bit\_loop**

**lea dx, newline**

**mov ah, 09h**

**int 21h**

**add si, 2**

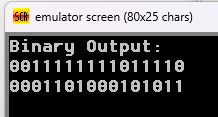
**loop next\_word**

**mov ah, 4Ch**

**int 21h**

**main endp**

**end main**

****

# **Question No.2**

Palindrome Check

# **Solution**

**.model small**

**.stack 100h**

**.data**

**var1 db 'MADAM','$'**

**palMsg db 13,10,'String is a Palindrome$'**

**notPalMsg db 13,10,'String is NOT a Palindrome$'**

**.code**

**main proc**

**mov ax, @data**

**mov ds, ax**

**lea si, var1**

**xor cx, cx**

**pushLoop:**

**mov al, [si]**

**cmp al, '$'**

**je comparePhase**

**xor ah, ah**

**push ax**

**inc si**

**inc cx**

**jmp pushLoop**

**comparePhase:**

**lea si, var1**

**mov dx, cx**

**cmpLoop:**

**cmp dx, 0**

**je isPalindrome**

**pop ax**

**cmp al, [si]**

**jne notPalindrome**

**inc si**

**dec dx**

**jmp cmpLoop**

**isPalindrome:**

**lea dx, palMsg**

**mov ah, 09h**

**int 21h**

**jmp finish**

**notPalindrome:**

**lea dx, notPalMsg**

**mov ah, 09h**

**int 21h**

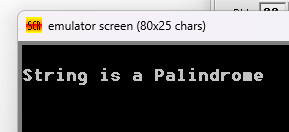
**finish:**

**mov ah, 4Ch**

**int 21h**

**main endp**

**end main**



# **Question No.3**

Hex to Binary and Binary to Hex converter

**Solution**

**.model small**

**.stack 100h**

**.data**

**menuMsg db 13,10,'1) Hex -> Binary',13,10,'2) Binary -> Hex',13,10**

**db 'Select option (1/2): $'**

**hexPrompt db 13,10,'Enter 4-digit hex (0-FFFF): $'**

**binPrompt db 13,10,'Enter 16-bit binary : $'**

**newline db 13,10,'$'**

**.code**

**PrintNL proc**

**lea dx, newline**

**mov ah, 09h**

**int 21h**

**ret**

**PrintNL endp**

**HexToBinary proc**

**call PrintNL**

**lea dx, hexPrompt**

**mov ah, 09h**

**int 21h**

**xor bx, bx**

**mov cx, 4**

**ReadHexDigit:**

**mov ah, 01h**

**int 21h**

**cmp al, '0'**

**jb ReadHexDigit**

**cmp al, '9'**

**jbe StoreHex**

**cmp al, 'A'**

**jb ReadHexDigit**

**cmp al, 'F'**

**ja ReadHexDigit**

**sub al, 37h**

**jmp Merge**

**StoreHex:**

**sub al, 30h**

**Merge:**

**mov si, 4**

**ShiftLoop:**

**rol bx, 1**

**dec si**

**jnz ShiftLoop**

**or bl, al**

**loop ReadHexDigit**

**call PrintNL**

**mov cl, 16**

**OutputBit:**

**rol bx, 1**

**mov dl, '0'**

**jnc Emit**

**mov dl, '1'**

**Emit:**

**mov ah, 02h**

**int 21h**

**dec cl**

**jnz OutputBit**

**call PrintNL**

**ret**

**HexToBinary endp**

**BinaryToHex proc**

**call PrintNL**

**lea dx, binPrompt**

**mov ah, 09h**

**int 21h**

**xor bx, bx**

**mov cx, 16**

**ReadBinBit:**

**mov ah, 01h**

**int 21h**

**cmp al, '0'**

**je RotateOnly**

**cmp al, '1'**

**jne ReadBinBit**

**rol bx, 1**

**or bl, 1**

**loop ReadBinBit**

**jmp ShowHex**

**RotateOnly:**

**rol bx, 1**

**loop ReadBinBit**

**ShowHex:**

**call PrintNL**

**mov cl, 4**

**NextNibble:**

**xor al, al**

**mov ch, 4**

**BuildNibble:**

**rol bx, 1**

**rcl al, 1**

**dec ch**

**jnz BuildNibble**

**cmp al, 9**

**jbe Add30**

**add al, 37h**

**jmp OutNib**

**Add30: add al, 30h**

**OutNib:**

**mov dl, al**

**mov ah, 02h**

**int 21h**

**dec cl**

**jnz NextNibble**

**call PrintNL**

**ret**

**BinaryToHex endp**

**main proc**

**mov ax, @data**

**mov ds, ax**

**lea dx, menuMsg**

**mov ah, 09h**

**int 21h**

**mov ah, 01h**

**int 21h**

**cmp al, '1'**

**je DoHexToBin**

**cmp al, '2'**

**je DoBinToHex**

**jmp ExitProg**

**DoHexToBin: call HexToBinary**

**jmp ExitProg**

**DoBinToHex: call BinaryToHex**

**ExitProg:**

**mov ah, 4Ch**

**int 21h**

**main endp**

**end main**

