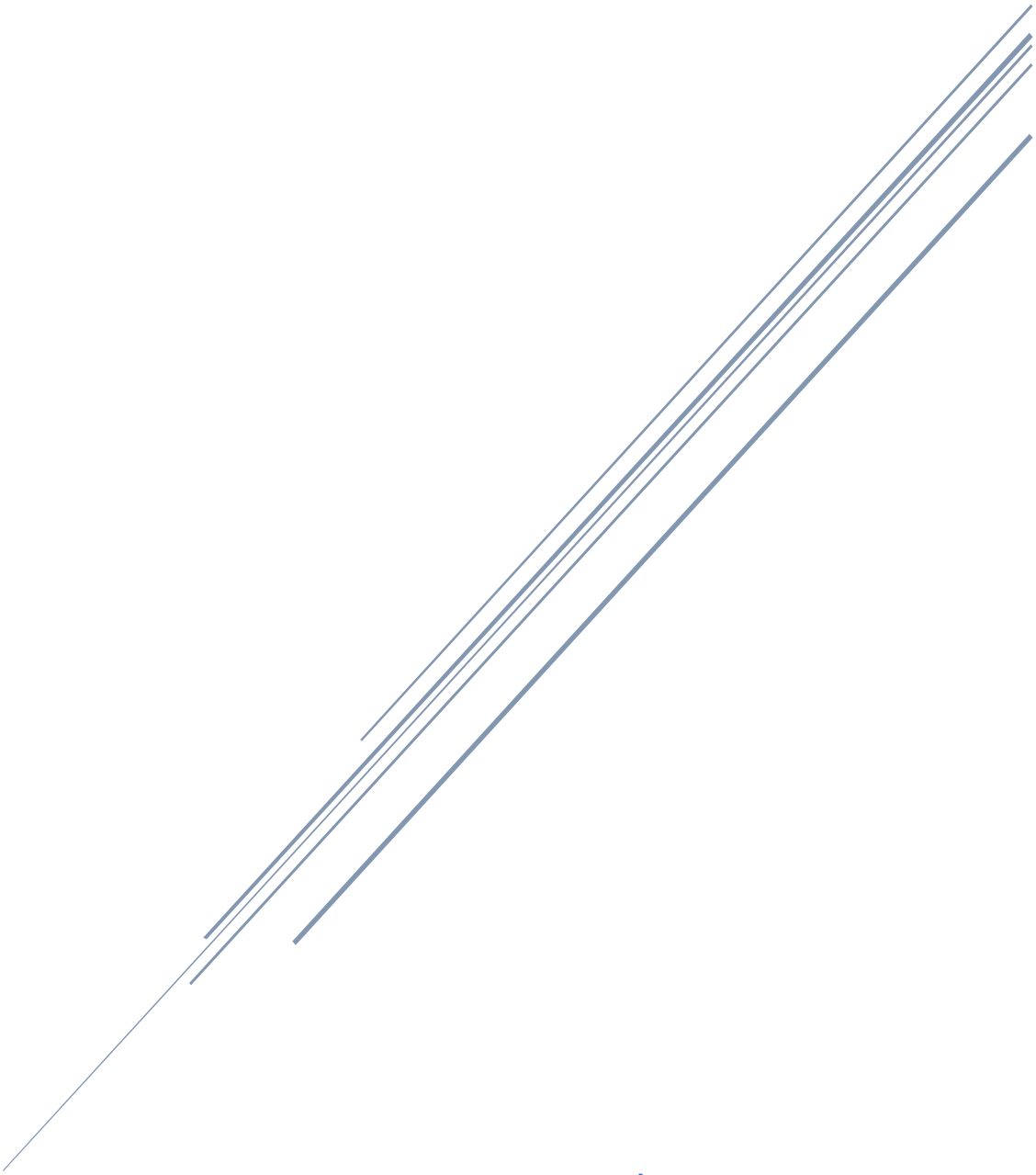
MİCROPROCESSOR AND EMBEDDED SYSTEMS MİDTERM REPORT

Decimal to Hexadecimal Convertor and Four Operations



Esma ÇELİKTEN /160316043
Fatma KURTULUŞ/ 170316064

Table of contents

INTRODUCTION

Mathematics is found in many areas of our life. Numbers and operations make many situations easier. These numbers have many forms. We have used the numbers on base of 10 .To facilitate operations with these numbers, we created a program in assembly language. We used the structure of emu8086 for this program. Let's explain the steps we have done in this program one by one.

FIRST PART

First, we determine the operations we want to do with the numbers on the base of 2. These are addition, subtraction, multiplication, division and convert. When the program is run, we want the user to choose what he wants to do. To do this, we define five separate decimal digits. Each number indicates the desired job (transaction).

```
org 100h

   OUTD DW ?
    IND DB ?

    SELECT DB ?

    INP1 DW ?   ; save decimal input

    COUNT DB ?


    D1 DB ?     ; save 1st decimal digit
    D2 DB ?     ; save 2nd decimal digit
    D3 DB ?     ; save 3rd decimal digit
    D4 DB ?     ; save 4th decimal digit
    D5 DB ?     ; save 5th decimal digit

jmp start       ; jump over data declaration

msg:    db      "1-Add",0dh,0ah,"2-Multiply",0dh,0ah,"3-Subtract",0dh,0ah,"4-Divide", 0Dh,0Ah,"5-Convert", '$'
msg2:    db      0dh,0ah,"Enter First No : $"
msg3:    db      0dh,0ah,"Enter Second No : $"
msg4:    db      0dh,0ah,"Choice Error $"
msg5:    db      0dh,0ah,"Result : $"
msg6:    db      0dh,0ah ,'thank you for using the calculator! press any key... ', 0Dh,0Ah, '$'
```

These ; 1-Addition , 2-Multiplication , 3- Subtract , 4-Divide , 5-Convert. We make these statements appear as a message. We enable the user to choose and switch to other stages to be done accordingly.

```
start:   mov ah,9
         mov dx, offset msg
         int 21h
         mov ah,0
         int 16h
         cmp al,31h
         je Addition
         cmp al,32h
         je Multiply
         cmp al,33h
         je Subtract
         cmp al,34h
         je Divide
         cmp al,35h
         je Convert
         mov ah,09h
         mov dx, offset msg4
         int 21h
         mov ah,0
         int 16h
         jmp start
```

It is requested to select the transaction to be used first. Then we will use int 16h to read a key press, to know the operation he choosed. Cmp al,31h the key press will be stored . The process is repeated for each expression.

ADDITION

```
Addition:   mov ah,09h ;then let us handle the case of addition operation
            mov dx, offset msg2  ;first we will display this message enter first no also using int 21h
            int 21h
            mov cx,0 ;we will call InputNo to handle our input as we will take each number seprately
            call InputNo  ;first we will move to cx 0 because we will increment on it later in InputNo
            push dx
            mov ah,9
            mov dx, offset msg3
            int 21h
            mov cx,0
            call InputNo
            pop bx
            add dx,bx
            push dx
            mov ah,9
            mov dx, offset msg5
            int 21h
            mov cx,10000
            pop dx
            call View
            jmp exit
```

We take each number separately. InputNo is called for this.

```
InputNo:    mov ah,0
            int 16h
            mov dx,0
            mov bx,1
            cmp al,0dh
            je FormNo
            sub ax,30h
            call ViewNo
            mov ah,0
            push ax
            inc cx
            jmp InputNo
```

InputNo stores the keypress in 'al'. That way we keep what action the user wants to do. We check if the choice is made correctly. If the selection is between 1 and 5, we continue the process. To get another number, you need to press enter. It is provided to get another number.

```
FormNo:     pop ax
            push dx
            mul bx
            pop dx
            add dx,ax
            mov ax,bx
            mov bx,10
            push dx
            mul bx
            pop dx
            mov bx,ax
            dec cx
            cmp cx,0
            jne FormNo
            ret
```

We got each number separately, so we need to create our number and store it, for example, in one bit. For this, we create the FormNo part.

```
View:   mov ax,dx
        mov dx,0
        div cx
        call ViewNo
        mov bx,dx
        mov dx,0
        mov ax,cx
        mov cx,10
        div cx
        mov dx,bx
        mov cx,ax
        cmp ax,0
        jne View
        ret

ViewNo:     push ax
            push dx
            mov dx,ax
            add dl,30h
            mov ah,2
            int 21h
            pop dx
            pop ax
            ret
```

We ensure that the specified values are displayed. We will push ax and dx to the stack because we will change there values while viewing then we will pop them back from. The stack we will do these so, we don't affect their contents. We will mov the value to dx as interrupt 21h expect that the output is stored in it. Add 30 to its value to convert it to ASCII.

MULTIPLY ,SUBTRACT and DIVIDE

We repeat the steps we took in the addition process section for subtraction, multiplication and division.

```
Multiply:   mov ah,09h
            mov dx, offset msg2
            int 21h
            mov cx,0
            call InputNo
            push dx
            mov ah,9
            mov dx, offset msg3
            int 21h
            mov cx,0
            call InputNo
            pop bx
            mov ax,dx
            mul bx
            mov dx,ax
            push dx
            mov ah,9
            mov dx, offset msg5
            int 21h
            mov cx,10000
            pop dx
            call View
            jmp exit
```

```
Subtract:   mov ah,09h
            mov dx, offset msg2
            int 21h
            mov cx,0
            call InputNo
            push dx
            mov ah,9
            mov dx, offset msg3
            int 21h
            mov cx,0
            call InputNo
            pop bx
            sub bx,dx
            mov dx,bx
            push dx
            mov ah,9
            mov dx, offset msg5
            int 21h
            mov cx,10000
            pop dx
            call View
            jmp exit
```

```
Divide:         mov ah,09h
                mov dx, offset msg2
                int 21h
                mov cx,0
                call InputNo
                push dx
                mov ah,9
                mov dx, offset msg3
                int 21h
                mov cx,0
                call InputNo
                pop bx
                mov ax,bx
                mov cx,dx
                mov dx,0
                mov bx,0
                div cx
                mov bx,dx
                mov dx,ax
                push bx
                push dx
                mov ah,9
                mov dx, offset msg5
                int 21h
                mov cx,10000
                pop dx
                call View
                pop bx
                cmp bx,0
                je exit
                jmp exit
```

CONVERT

The fifth option is 'convert'. In this section, numbers on the base of ten are converted to numbers on the base of hex. The user clicks on option 5 and writes which number she/he wants to hexadecimal.

```
Convert:

    OP1 db ' $'
    OP2 db '1. DECIMAL TO HEXADECIMAL $'

    REM DW ?

    NEWL DB 10,13,'$'   ;newline


    DECIMALINP DB 'First Insert + or - than insert input value between (0-9) Maximum Limit is :  65,535 $'

    INDECIMAL DB 'DECIMAL VALUE: $'
    INBINARY DB 'BINARY VALUE: $'
    INHEXADECIMAL DB 'HEXADECIMAL VALUE: $'

    ERRORMSG DB 'Input Error! Wrong Key Inserted $'


    ENDMSG DB 9,'THANKS $'
.CODE
```

LAST PART

In unwanted situations, we ensure that the system gives an error message.

```
;;ERROR message
     ERROR:
          MOV AH,9
          LEA DX,NEWL
          INT 21H
          LEA DX,NEWL
          INT 21H

          LEA DX,ERRORMSG
          INT 21H

          JMP AGAIN
```

Finally, we have created the Exit section if you want to exit the program.

```
;;end programm
     EXIT:
          MOV AH,9
          LEA DX,NEWL
          INT 21H
          LEA DX,NEWL
          INT 21H
          LEA DX,NEWL
          INT 21H

          LEA DX,ENDMSG
          INT 21H

          MOV AH,4CH ; ignore emulator haulted
          INT 21H
ret
```

OUTPUT PART

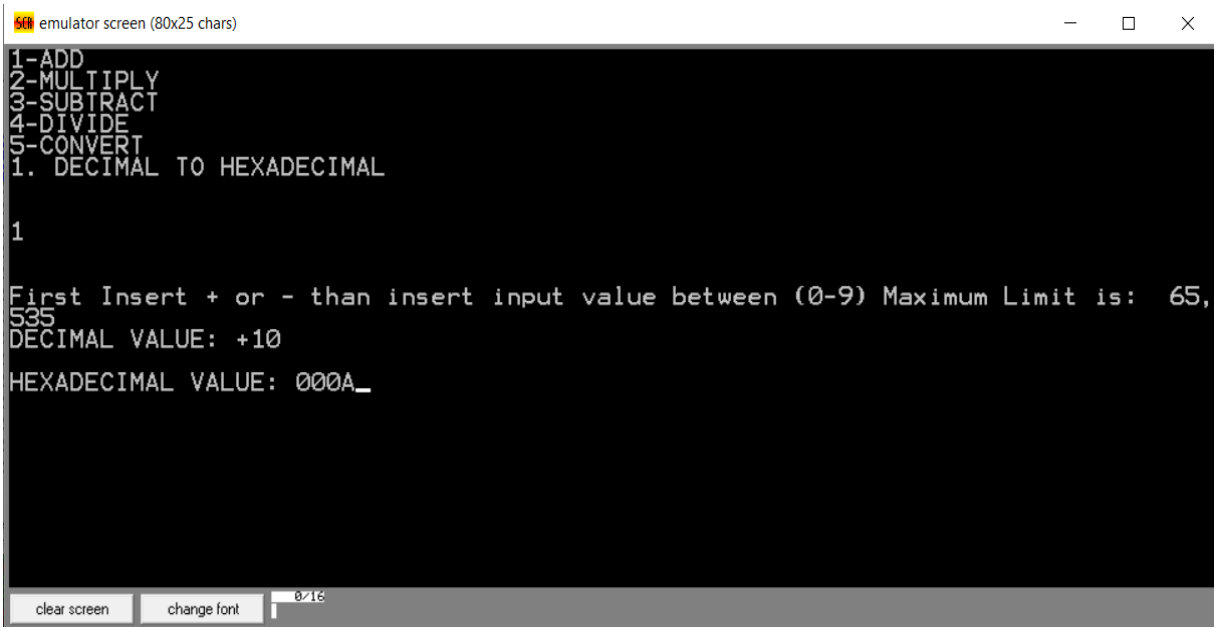
This is the first image that appears on the screen when the program runs. Here, the user is asked to choose.

The user who chooses the first option wants to add and the program takes the first and second value that she/he wants to add and shows the result.

The user who chooses the second option wants to Multiply. Numbers are requested and the result is shown.

The third option is subtraction. Two values to be taken are taken and the result is shown.

The fourth option is dividing. The value to be divided and the dividing number is requested and the operation is taking place.



And finally, the fifth option is the conversion option. Here, the user who clicks option 5 presses hexadecimal option 1 to the decimal place, then enters the decimal value he wants to convert from decimal to hexadecimal. Before entering the number in the decimal place, we need to specify whether the number to be entered will be positive(+) or negative(-). Then we enter the number and press enter. The program shows us the transformation of the number.

COMMENT

In this program, we saw the mathematical operations we can do with the assembly language, which is the machine language. We witnessed the necessary steps to ensure these transactions. We have seen that there are more steps in this language compared to other languages and these steps must be completed carefully. Our program is a program that can perform four arithmetic operations. In order to perform these operations, each expression entered is first translated into binary. It is then processed in binary and recycled to decimal places. In the statements that are intended to be converted to hexadecimal, firstly, it is converted from decimal to binary base, then it is turned from base to hexadecimal and turned and shown. The system gives errors in every unwanted situation.