

X86 Architecture – Lab 1

Description:

In this lab, we were asked to analyze, link, and run three programs, and create a fourth program that would display a '?', read two single-digit decimal numbers, and display the sum of the numbers on the next line with a message.

Equipment Used:

- GUI Turbo Assembler

Procedure:

I started by copying the programs into the assembler and reformatting them. This included indenting parts of the code and replacing the quotation marks with proper quotation marks (for some reason the assembler didn't recognize their original ones).

Next, I used the "Assemble, Build & Run" button to run and test the programs. This caused a window to appear, in which I was able to see and interact with the program.

To create the custom program, I utilized code from the provided examples in week 10.

Specifically, I utilized the program shown near the end of the X86Intro document.

Program Description:

Echo Program:

This program takes a user input and displays a copy of it on the screen. I think it does this by calling an interrupt and executing a handler that either reads input or displays a character, depending on which hex value is provided.

Print String Program:

This program uses a pseudo instruction to save the string “HELLO!” to memory and displays it by setting a value in a register and calling a certain interrupt.

Case Conversion Program:

This program saves two strings to memory: one that asks for a lower-case letter and another that is used when the result is displayed. After this, the program reads a character from the user using an interrupt. It assumes that this character is lowercase and does not check for errors. It converts the character to uppercase by subtracting 32. Then it displays the new character with the aforementioned output message.

Sum of Two Numbers:

This program saves several strings to memory, each of which is used in the final output. Then it displays a “?” character, indicating that the user should input some numbers. The program is only meant to read single digit numbers and will not allow the user to correct misinputs. Both the numbers are saved into the BL and CL registers. To compute the sum, the program will subtract 48 from each value (converting them from characters to integers), add them together (storing the sum in BL), and add 48 to the value in BL (converting it from an integer to a character). Then it will display the sum to the screen, along with the other parts of the final output message. The

program does not properly display numbers greater than 9 because the ASCII standard does not support them. In that case, it may display another character.

Results:

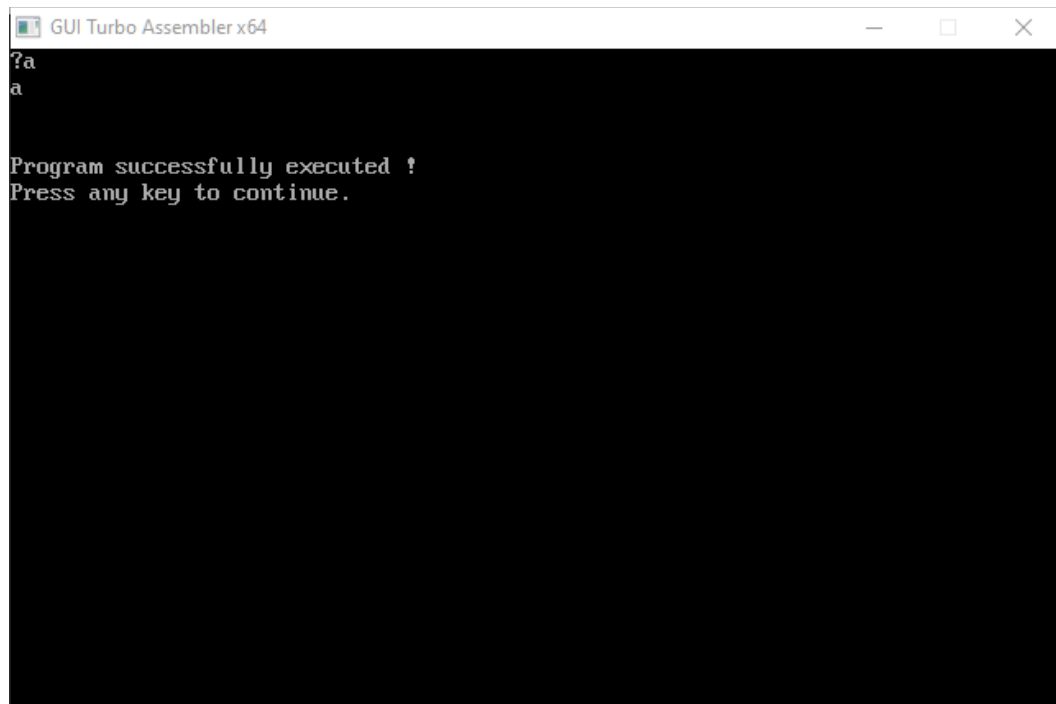


Figure 1 The output from the Echo program

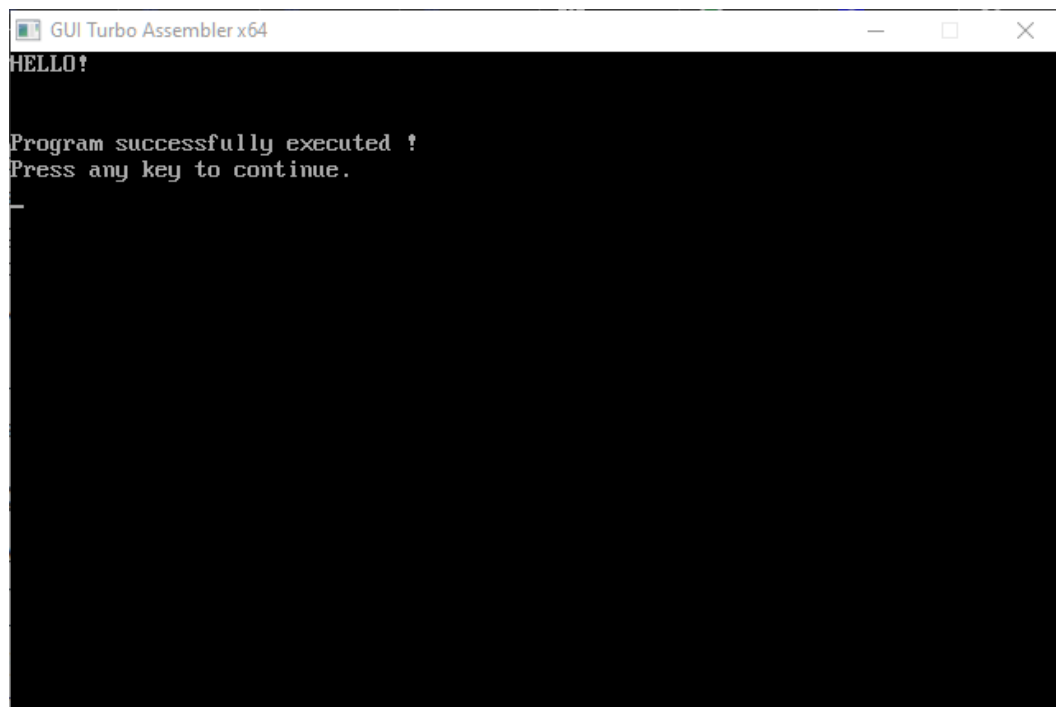
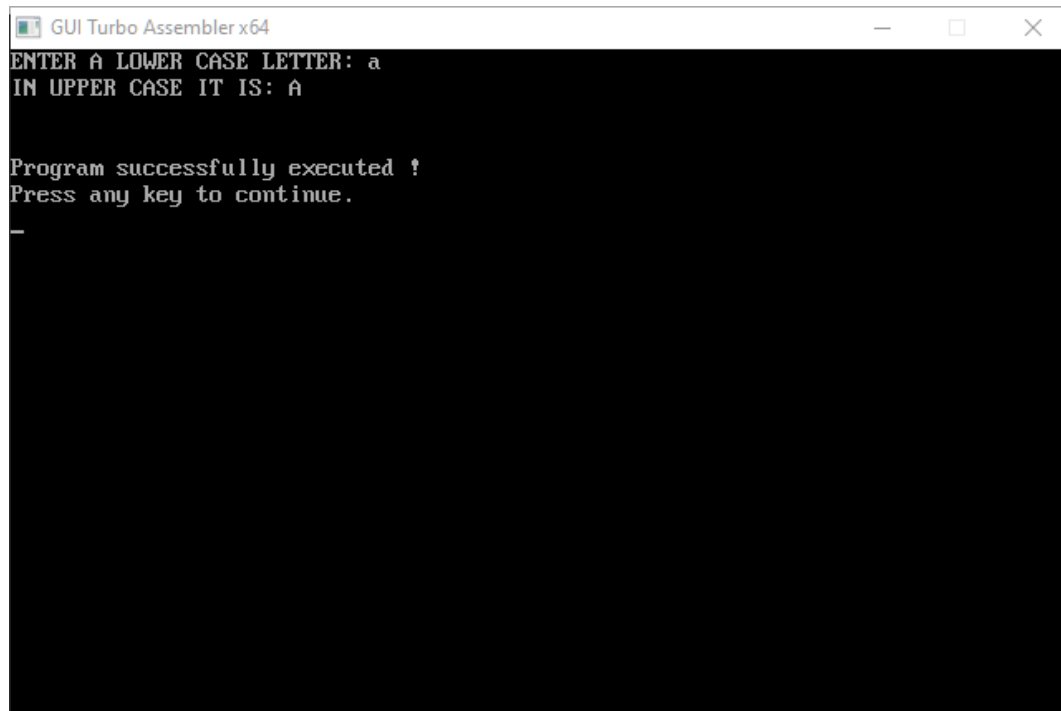


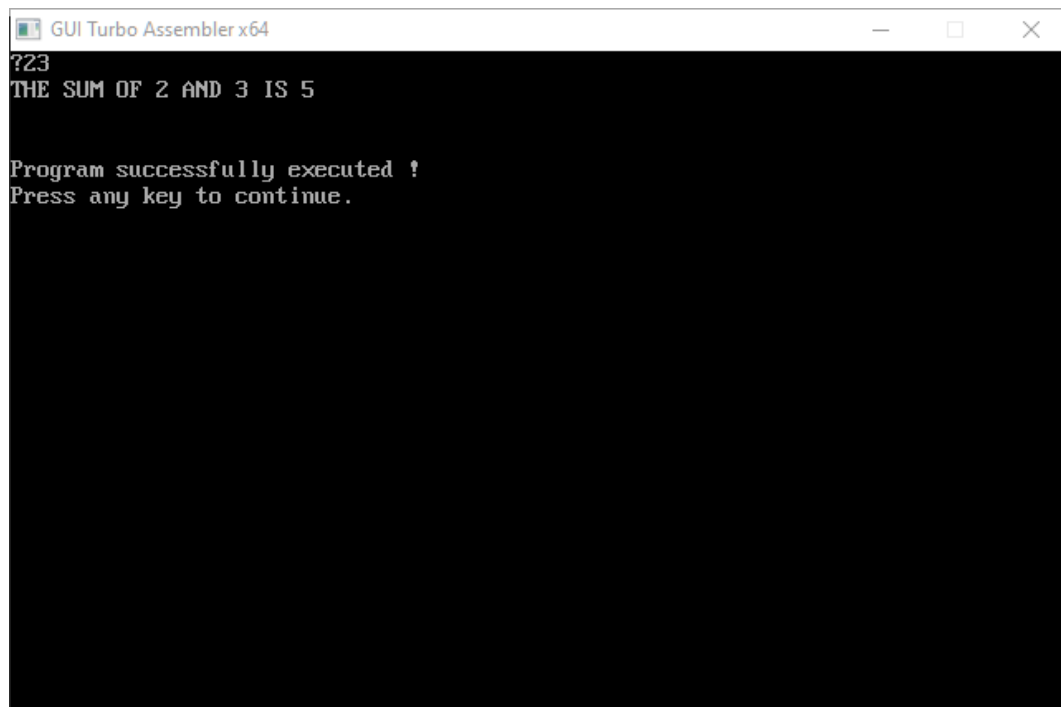
Figure 2 The output from the Print String program



```
GUI Turbo Assembler x64
ENTER A LOWER CASE LETTER: a
IN UPPER CASE IT IS: A

Program successfully executed !
Press any key to continue.
```

Figure 3 The output from the Case Conversion program



```
GUI Turbo Assembler x64
?23
THE SUM OF 2 AND 3 IS 5

Program successfully executed !
Press any key to continue.
```

Figure 4 The output from the Sum of Two Numbers program

Conclusions:

This lab helped me learn some of the basics of x86 assembly language. Firstly, I learned that the register naming scheme is a lot different than the other assembly languages we've used so far.

Most languages simply organize their registers by number, but x86 uses special names for all of them. EAX is used for 32-bit values, AX is used for 16-bit values, and AH/AL are used for 8-bit values. The smaller registers are actually subparts of the EAX register. I also learned that x86 assembly offers powerful functions that can be called by setting parameters in registers and generating an interrupt. To my understanding, these are calling functions from the operating system.

Code Listing:

Echo Program:

```
TITLE PGM4_1: ECHO PROGRAM
.MODEL SMALL
.STACK 100H
.CODE
MAIN PROC
;display prompt
    MOV AH,2 ;display character function
    MOV DL,'?' ;character is ???
    INT 21H ;display it
;input a character
    MOV AH,1 ;read character function
    INT 21H ;character in AL
    MOV BL,AL ;save it in BL
;go to a new line
    MOV AH,2 ;display character function
    MOV DL,0DH ;carriage return
```

```

    INT 21H ;execute carriage return
    MOV DL,0AH ;line feed
    INT 21H ;execute line feed
;display character
    MOV DL,BL ;retrieve character
    INT 21H ;and display it
;return to DOS
    MOV AH,4CH ;DOS exit function
    INT 21H ;exit to DOS
MAIN ENDP
    END MAIN

```

Print String Program:

```

TITLE PGM4_2: PRINT STRING PROGRAM
.MODEL SMALL
.STACK 100H
.DATA
MSG DB 'HELLO!$'
.CODE
MAIN PROC
;initialize DS
    MOV AX,@DATA
    MOV DS,AX ;initialize DS
;display message
    LEA DX,MSG ;get message
    MOV AH,9 ;display string function
    INT 21H ;display message
;return to DOS
    MOV AH,4CH
    INT 21H ;DOS exit
MAIN ENDP
    END MAIN

```

Case Conversion Program:

TITLE PGM4_3: CASE CONVERSION PROGRAM

.MODEL SMALL

.STACK 100H

.DATA

MSG1 DB 'ENTER A LOWER CASE LETTER: \$'

MSG2 DB 0DH,0AH,'IN UPPER CASE IT IS: '

CHAR DB ?, '\$'

.CODE

MAIN PROC

;initialize ds

MOV AX,@DATA ;get data segment

MOV DS,AX ;initialize DS

;print user prompt

LEA DX,MSG1 ;get first message

MOV AH,9 ;display string function

INT 21H ;display first message

;input a character and convert to upper case

MOV AH,1 ;read character function

INT 21H ;read a small letter into AL

SUB AL,20H ;convert it to upper case

MOV CHAR,AL ;and store it

;display on the next line

LEA DX,MSG2 ;get second message MSG2

MOV AH,9 ;display string function

INT 21H ;display message and upper case

;DOS exit

MOV AH,4CH

INT 21H ;DOS exit

MAIN ENDP

END MAIN

Sum of Two Numbers Program:

TITLE PG80_8: SUM OF TWO NUMBERS

.MODEL SMALL

.STACK 100H

.DATA

MSG1 DB 0Dh,0Ah,'THE SUM OF \$'

MSG2 DB ' AND \$'

MSG3 DB ' IS \$'

.CODE

MAIN PROC

;initialize DS

MOV AX,@DATA ;get data segment

MOV DS,AX ;initialize DS

;display prompt

MOV AH,2 ;display character function

MOV DL,'?' ;character is '?'

INT 21h ;display it

;input a character

MOV AH,1 ;read character function

INT 21h ;character in AL

MOV BL,AL ;save it in BL

MOV AH,1 ;read character function

INT 21h ;character in AL

MOV CL,AL ;save it in CL

;display the first part of message

LEA DX,MSG1 ;get first message

MOV AH,9 ;display string function

INT 21h ;display message

MOV AH,2 ;display character function

MOV DL,BL ;character is the one stored in BL

INT 21h ;display it

LEA DX,MSG2 ;get second message

```
MOV AH,9 ;display string function
INT 21h ;display it
MOV AH,2 ;display character function
MOV DL,CL ;character is the one stored in CL
INT 21h ;display it
LEA DX,MSG3 ;get third message
MOV AH,9 ;display string function
INT 21h ;display it
;calculate the sum
SUB BL,30h ;change BL to a right value for calculation
SUB CL,30h ;change CL to a right value for calculation
ADD BL,CL ;add the numbers up and store the sum in BL
ADD BL,30h ;change the result in BL to hex value
;display the result
MOV AH,2 ;display character function
MOV DL,BL ;character is the one stored in BL
INT 21h ;display it
;DOS exit
MOV AH,4CH
INT 21h ;DOS exit
MAIN ENDP
END MAIN
```