Laboratory - Microprocessor Based Systems

Laboratory Assignment 2: Instruction set.

This laboratory practice will be devoted to small matrixes. You will develop a programme which allows the user to calculate the determinant of 3x3 matrixes. The student must perform the requirement analysis and its implementation. To successfully complete this assignment, you will have to use a significant part of the instructions of 8086 microprocessor.

Task 1. Lab2.asm

Write a programme in assembler which calculates the determinant of a 3x3 matrixes of integer values. The initial values of the matrix might be pre-defined in memory (recommended for an initial version) or typed. For that purpose, the program will show the user a menu with the next options:

1) Calculate the determinant using the default values.

2) Calculate the determinant by Typing the new values

In case the user selects the second choice, a message will be printed with the instructions on how to input the data to be properly read by the application. To make it simpler, it is recommended to use only one interruption to take all the inputs from keyboard.

No matter which option is selected, the programme must print the matrix on the screen, calculate its determinant and store it into a variable named B, and then print it with the next format:

$$|A| = \begin{vmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{vmatrix} = B$$

To succeed in this task, you will have to implement an auxiliar subroutine which converts an integer number into a ASCII String to print the results as decimal notation number.

i.e.: the number 0122h must be convert in to the ASCII String “274” (32h, 37h and 34h).

Important:

- To access the numbers of the A matrix, the based indexed addressing must be used.
- The A values are 8 bits while B values are 16 bits.
- The subroutine that converts from binary to ASCII must receive the number that needs to be converted in the register BX, and then it returns in DX:AX the memory address where the ASCII String begins (DX contains the segment while AX contains the offset). The String must be ended with '$' to ease the printing process by using the function 9 of the INT 21h of MS-DOS (read ANEX of Lab0 and Alumno.ASM).
- Keep in mind that A values can be negatives. In this case, the values must be stored in memory using 2 complement formatting.

ANEX I: Functions of the operating system to print text and end the execution.

The operating system MS-DOS provides most of its services through the interruption INT 21h. Firstly, the function number must be loaded into the AH register. Moreover, individual functions can require some other input parameters whose value must be stored in other registers. Three functions for programmes developed in these laboratory assignments are detailed below.

Function 2H: Send an ASCII code to the screen peripheral.

Description: Print a single character into the screen.

Input parameters:

- AH = 2h
  DL = ASCII code to be printed.

Output parameters: none.

Involved registers: none.

Example:

```
MOV AH, 2h    ; Function number
MOV DL, 'A'   ; Character to be printed
INT 21h          ; Software interruption
```

Function 9H: Send an ASCII string to the screen peripheral.

Description: Print a string ended with character '$' into the screen.

Input parameters:

- AH = 9h
  DX = Memory offset of the data segment where the first character of the string is stored.

Output parameters: none.

Involved registers: none.

Example:

```
.DATA
TextPrint DB "Hello world, 13, 10, $"  ; String ended with the characters CR, LF and
'$'
.CODE
; If DS is the segment where the text to print is stored
MOV DX, OFFSET TextPrint   ; DX: Offset at the beginning of the text to print string
MOV AH, 9h   ; Printing String function
INT 21h          ; Software interruption
```

Function 0AH: Reading characters introduced by the keyboard peripheral.

Description: It captures the characters introduced using the keyboard and prints them into the screen in a local echo mode and transfers them to the application. The function ends when the users introduce the ASCII 13 (carriage return). The operating system allows the character's input up to the maximum numbers of characters previously defined. Otherwise, these overflow characters won't be printed and stored.

Input parameters:

- DX = Memory offset of the data segment where the space to store the captured characters is defined. In the first byte of that reserved memory space the maximum number of characters desired must be defined.

Output parameters: In the second byte of the reserved memory space pointed by DX the operating system stores the number of characters stored.

Involved registers: none.

Example:

```
MOV AX, 0Ah ; Capture keyboard function.
MOV DX, OFFSET NAME ; Memory space reserved named NAME
MOV NAME[0], 60   ; Maximum number of characters is 60
INT 21h          ; Software interruption

In NAME[1] the operating system will store the number of registered characters.
```

Function 4C00h: End of programme.

Description: This function indicates to the operating system that the process (programme) has ended correctly.

Input parameters: none.

Output parameters: none

Involved registers: none.

Example:

```
MOV AX, 4C00h ; End of programme.
INT 21h          ; Software interruption
```