Assembly Language LAB

Islamic University – Gaza
Engineering Faculty
Department of Computer Engineering
2013
ECOM 2125: Assembly Language LAB
Created by: Eng. Ahmed M. Ayash
Modified and Presented By: Eihab S. El-Radie

Lab # 11

Keyboard Input with int 16h
&
Macros

Part1: Bios Level Programming (Keyboard Input with int 16h):

Difference between BIOS and DOS?

- BIOS refers to (Basic Input Output System) it is responsible for booting of your system.
- While DOS(Disk Operating System) is an operating system that user uses by entering commands in command prompt.
- The BIOS is a computer program embedded on a chip on a computer's motherboard that recognizes and controls various devices that make up the computer.
- **Int 16h and Int 10h** are a Bios-level access interrupts, while **Int 21h** is an MS-Dos- level Access interrupt.

Bios Int 16h: Keyboard Input:

- **Function 01 Int 16h:**

  Check if a key is pressed or not.

```
mov ah,01
Int 16h
```

Result: If ZF=0 >>> then a key is pressed

ZF=1 >>> a key is not pressed

- **Function 00 Int 16h:**

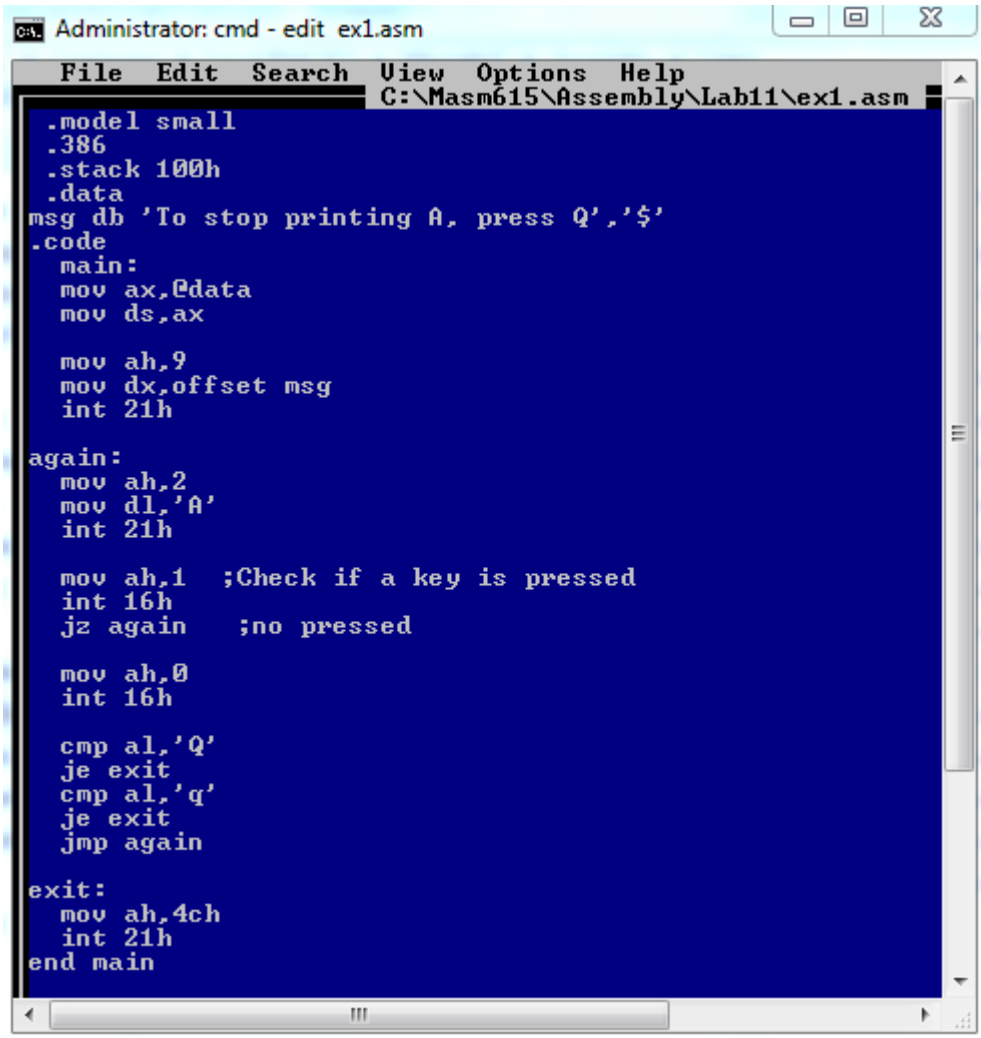
  Check which key is pressed, and store the ASCII code in **AL**

```
mov ah,0
Int 16h
```

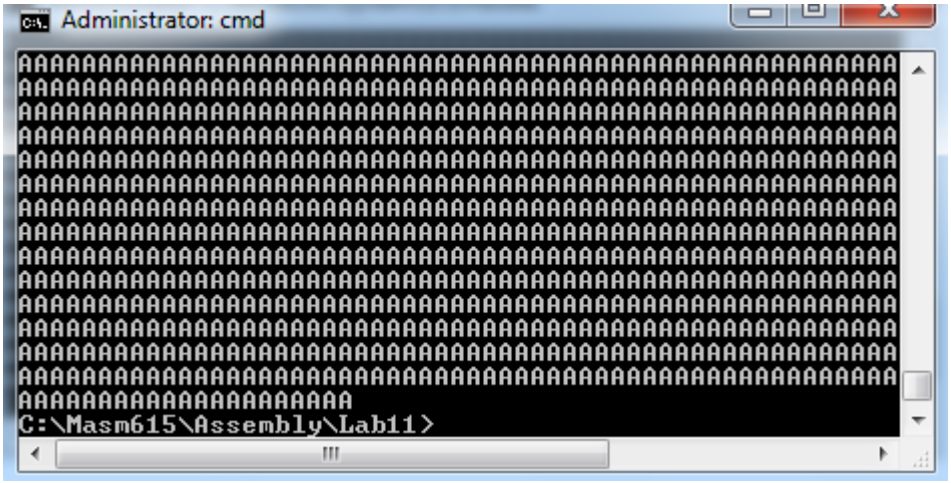
Lab work:

Excercise1:

Write a program that prints the letter A continuously, when Q or q is pressed only it will stop.


Administrator: cmd - edit ex1.asm
File Edit Search View Options Help
C:\Masm615\Assembly\Lab11\ex1.asm
.model small
.386
.stack 100h
.data
msg db 'To stop printing A, press Q','$'
.code
main:
mov ax,@data
mov ds,ax
mov ah,9
mov dx,offset msg
int 21h
again:
mov ah,2
mov dl,'A'
int 21h
mov ah,1 ;Check if a key is pressed
int 16h
jz again ;no pressed
mov ah,0
int 16h
cmp al,'Q'
je exit
cmp al,'q'
je exit
jmp again
exit:
mov ah,4ch
int 21h
end main

Output:


Administrator: cmd
C:\Masm615\Assembly\Lab11>

Part2: Macros

Macro is a named block of assembly language statements. Once defined, it can be invoked (called) as many times as you wish. When you invoke a macro a copy of its statements is inserted directly into the program.

Macro definition

Macros are either coded **directly** anywhere in the source program (usually at the beginning on the program before the **.data**), **or** they are **placed in a separate text file** (library) and inserted into the source program during assembly using the **INCLUDE** directive.

The Syntax of a Macro is:

```
macroname MACRO [parameter-1, parameter-2 ,parameter-3..]
.
.
.
set of instructions
.
.
ENDM
```

The statement between the MACRO and ENDM directive are **not assembled until the macro is invoked**. There can be any number of parameters in the macro definition, as long as they are separated by commas.

Invoking Macros

To use a macro it is only necessary to call it by its name, as if it were another assembler instruction, since directives (the call directive) are no longer necessary as in the case of the subroutine. A macro is invoked by inserting its name into a program's source code, possibly followed by macro arguments. The syntax for calling a macro is

```
macroname [argument-1, argument-2, argument-3..]
```

Macroname must be a name of a macro defined prior to this point in the source code. Each argument is a text value that replaces a parameter in the macro. The order of the arguments must correspond to the order of the parameters, but *the number of the arguments do not have to match the number of parameters*. If too many arguments are passed, the assembler issues a warning. If too few arguments are passed to the macro, the unfilled parameters are left blank. When calls to the macro are found, each macro call is replaced by a copy of the macro.

As mentioned above, in some cases we separate the macros in other file as a library
So it reduces the size of the code, and the macros can be used in more than one assembly program.
We can create a file and name it as ***macro.lib*** and at the beginning of the .asm file you can include the library using the include directive so you can use any macro that existed in the library.

Include macro.lib

The macros have the following advantages:

1- Reduce the written Code.

2- Make the code readable.

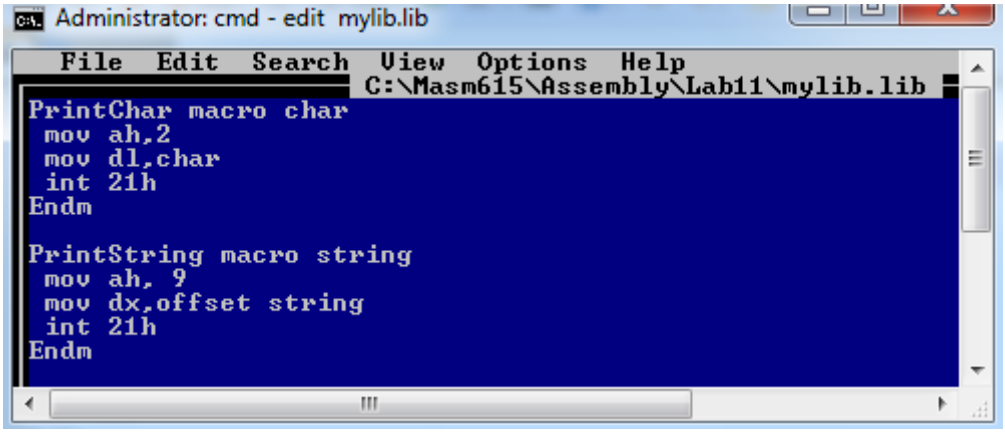
3- Reduce the errors.

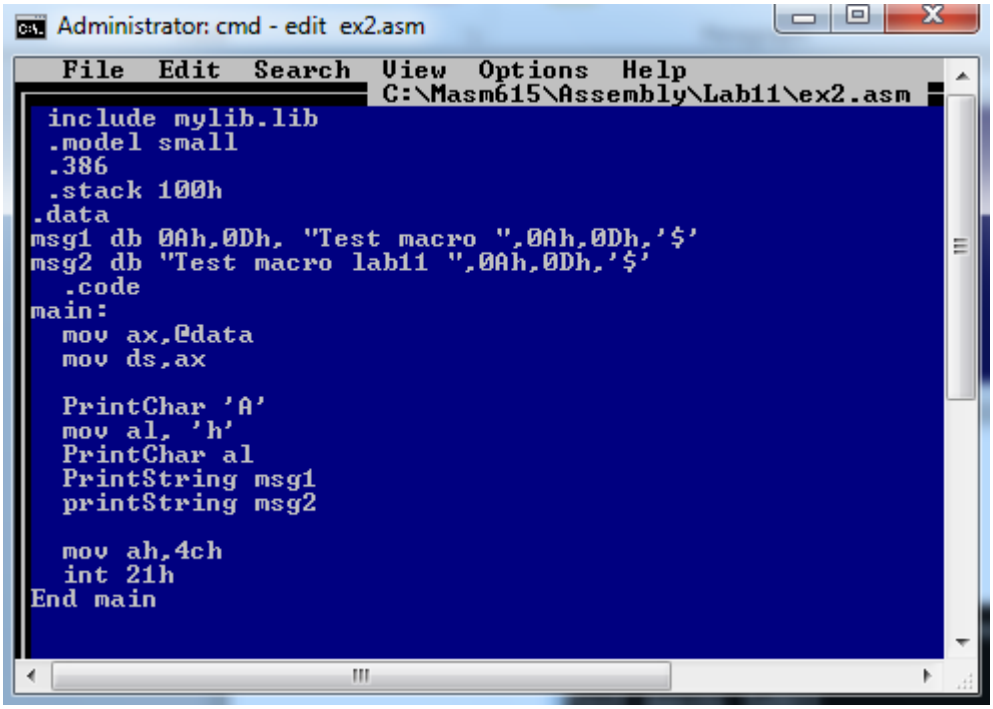
4- Faster than the subroutine since it do not need to jump to another address with the ***call*** and return to the previous address with the ***ret***.

But the macro causes the assembly program to be large. After replacing each call to a macro with the macro instruction, the program will be large and will reserve a large area in the code segment. On the other hand, the subroutine is slower but is smaller in the memory since the subroutine call will remain a single instruction.

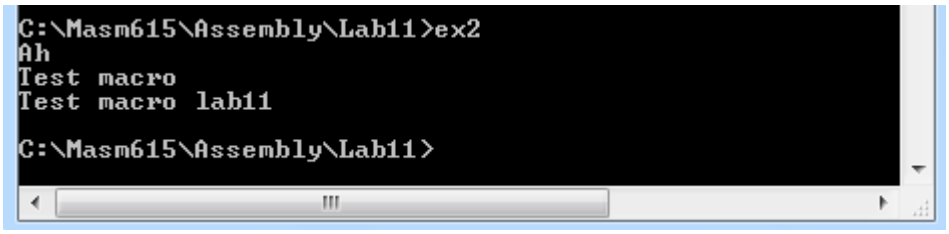
Excercise2:

Write these macros in file and name it **mylib.lib** and save it in your main program directory path.





Output:



Homework:

Write a macro code named ***Chars_Print*** taking two arguments ***startingChar*** and ***numOfchars*** you want to print starting with ***startingChar*** in one line. Save this macro in a file named ***hw.lib***, and then include it in your code. Try to invoke this macro to print characters from A to Z.