NAME : BATIN, KATRINA RICCI C.________________________ **DATE : OCTOBER 10, 2024____________________+**

SUBJ/SEC : COMPUTER ARCHITECTUIRE AND ORANIZATION___ **PROFESSOR : Engr. Rolito L. Mahaguay, *MSE-CpE, PCpE***

LABORATORY EXPERIMENT #2
INTRODUCTION TO TURBO ASSEMBLER

OBJECTIVE (S):

1. to be able to learn the basics of turbo assembler.
2. To properly assemble and link a program using tasm and tlink.
3. To be able to create an assembly language program using turbo assembler with SK as editor.

REQUIREMENTS:

1 personal computer
1 system disk
1 data disk

- Asembler – **TASM.EXE**
- Loader - **TLINK.EXE**
- Editor - **SK**

DISCUSSION:

Writing a program in Assembly Language follows the same procedures as in high-level languages such as Pascal.

1. Type the program instructions into the computer using a text editor, then save the program on the disk.
2. Translate the text file into a machine language program using an assembler. If the assembler finds errors, correct them with the text editor and reassemble the program.
3. Convert the assembler output to an executable ***run module*** using the loader.
4. Execute the program.
5. Check the results. If they differ from what you expected, you must find the error or ***bugs***, that is, you must ***debug*** the program.

Text_Editor

A text editor is a program that allows you to enter and prepare your program from the ordinary keyboard into a computer-readable form. It also allows you to save this file on the disk for later use. The assembler and loader program requires the inputs to be saved files from the disk.

A text editor can be any popular work processor or edit program that can produce pure ***ASCII*** text.

As a standard convention, programs written in assembly language are usually given the filename with an extension of ***.ASM***. This is also the default filename extension that ***TASM*** searches.

Assembler

A program written in assembly language is translated to machine code by an assembler. Assemblers and the corresponding assembly language mnemonics are generally limited to use with one particular microprocessor, which limits their portability, or use on other machines. Today's assemblers do much more than translate assembly language mnemonics into binary code.

Loader

Before the microprocessor can execute any machine instruction, it must first be loaded into the memory accessible to it. The loader is the program that actually takes the machine instructions (object code) and places it in memory at the specified starting address for execution. Loaders range from very simple to very complex.

PROCEDURES:

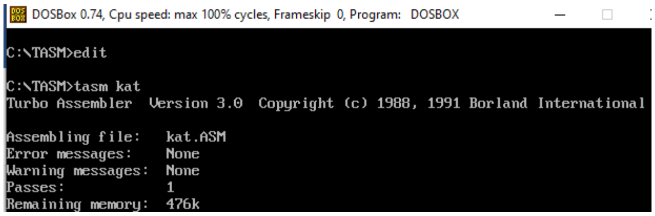
1. Write the following program using EDIT as your text editor. Assign the filename to your ***nickname.asm*** (Take note that filename should be a maximum of 8 characters only)

```
                .model small
                .code
                org 100h

start           mov ah,2
                mov dl, 'R'
                int 21h
                mov dl, 'E'
                int 21h
                mov dl, 'D'
                int 21h
                int 20h
end start
```

2. Save the program by pressing F2.

3. Go to the DOS prompt and then assemble the program by using the command below:

   C:\>tasm nickname ***ex: C:\>tasm tolitz***


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: DOSBOX

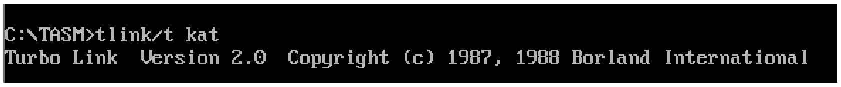
```
C:\TASM>edit

C:\TASM>tasm kat
Turbo Assembler  Version 3.0  Copyright (c) 1988, 1991 Borland International

Assembling file:   kat.ASM
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  476k
```

4. If there are warnings and errors in the assembling process, go back to the source code and fix those errors. If no error occurs proceed to the next step.

5. Link the program by using the command :

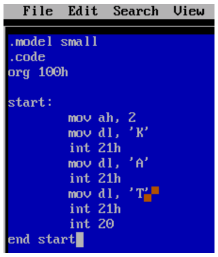
   C:\>tlink/t nickname ***ex: C:\>tlink/t tolitz***

   There should be no BAD OBJECT FILE message in the linking process if a *bad object file occurs* assemble the program once again.



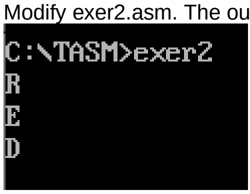
```
C:\TASM>tlink/t kat
Turbo Link  Version 2.0  Copyright (c) 1987, 1988 Borland International
```

6. Execute the program by simply typing the filename on the prompt. What is the output of the given program?



```
.model small
.code
org 100h

start:
        mov ah, 2
        mov dl, 'K'
        int 21h
        mov dl, 'A'
        int 21h
        mov dl, 'T'
        int 21h
        int 20
end start
```

7. Modify exer2.asm. The output should be:



```
C:\TASM>exer2
R
E
D
```

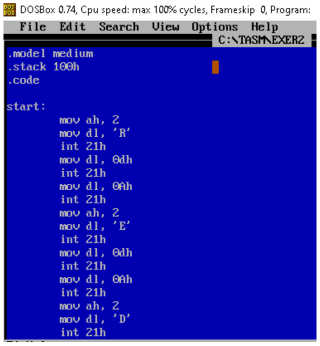
R
E
D

8. Write down the modified program in the space below.

```
.model medium
.stack 100h
.code

start:
      mov ah, 2
      mov dl, ‘R’
      int 21h
      mov dl, 0dh
      int 21h
      mov dl, 0Ah
      int 21h
      mov dl, ‘E’
      int 21h
      mov dl, 0dh
      int 21h
      mov dl, 0Ah
      int 21h
      mov dl, ‘D’
      int 21h
      mov dl, 0dh
```

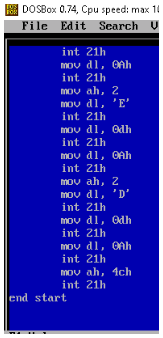
```
        int 21h
        mov dl, 0Ah
        int 21h

        mov ah, 4ch
        int 21h

    end start
```

9. Copy the program to your data disk.



DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program:
File Edit Search View Options Help
C:\TASM\EXER2

```
.model medium
.stack 100h
.code

start:
        mov ah, 2
        mov dl, 'R'
        int 21h
        mov dl, 0dh
        int 21h
        mov dl, 0Ah
        int 21h
        mov ah, 2
        mov dl, 'E'
        int 21h
        mov dl, 0dh
        int 21h
        mov dl, 0Ah
        int 21h
        mov ah, 2
        mov dl, 'D'
        int 21h
```



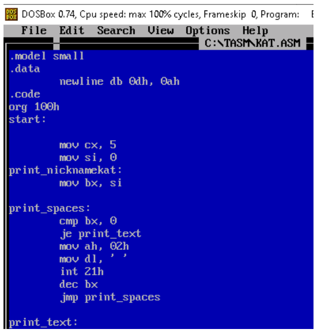
```
        int 21h
        mov dl, 0Ah
        int 21h
        mov ah, 2
        mov dl, 'E'
        int 21h
        mov dl, 0dh
        int 21h
        mov dl, 0Ah
        int 21h
        mov ah, 2
        mov dl, 'D'
        int 21h
        mov dl, 0dh
        int 21h
        mov dl, 0Ah
        int 21h
        mov ah, 4ch
        int 21h
end start
```

EXERCISES:

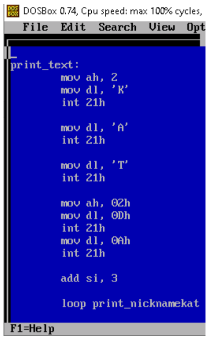
1. Write down the required program output from your instructor/professor.

```
. model small
.data
        newline db 0dh, 0ah
.code
org 100h
start:
        mov cx, 5
        mov si, 0

print_nicknamekat:
        mov bx, si

print_spaces:
        cmp bx, 0
        je print_text
        mov ah, 02h
        mov dl, ‘ ‘
        int 21h
        dec bx
        jmp print_spaces

print_text:
        mov ah, 2

        mov dl, ‘K’
        int 21h

        mov dl, ‘A’
        int 21h

        mov dl, ‘T’
        int 21h

        mov ah, 02h
        mov dl, 0Dh
        int 21h
        mov dl, 0Ah
        int 21h

        add si, 3

        loop print_nicknamekat

        mov ah, 4Ch
        int 21h

end start
```

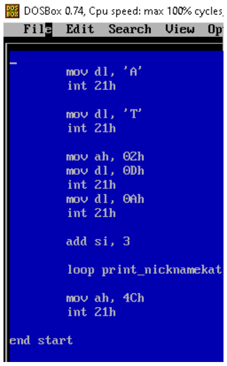
2. Encode the program.


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program:
File Edit Search View Options Help
C:\TASM\KAT.ASM

```
.model small
.data
        newline db 0dh, 0ah
.code
org 100h
start:

        mov cx, 5
        mov si, 0
print_nicknamekat:
        mov bx, si

print_spaces:
        cmp bx, 0
        je print_text
        mov ah, 02h
        mov dl, ' '
        int 21h
        dec bx
        jmp print_spaces

print_text:
```


DOSBox 0.74, Cpu speed: max 100% cycles,
File Edit Search View Opt
F1=Help

```
print_text:
        mov ah, 2
        mov dl, 'K'
        int 21h

        mov dl, 'A'
        int 21h

        mov dl, 'T'
        int 21h

        mov ah, 02h
        mov dl, 0Dh
        int 21h
        mov dl, 0Ah
        int 21h

        add si, 3

        loop print_nicknamekat
```



```
        mov dl, 'A'
        int 21h

        mov dl, 'T'
        int 21h

        mov ah, 02h
        mov dl, 0Dh
        int 21h
        mov dl, 0Ah
        int 21h

        add si, 3

        loop print_nicknamekat

        mov ah, 4Ch
        int 21h

end start
```

3. Ask your instructor to check your work, then copy it to your data disk.

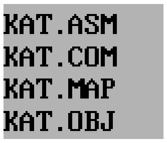
4. How many files are created using the turbo assembler?

-4 Files

5. What are those files?

(using dir kat)

1. **KAT ASM**
2. **KAT EXE**
3. **KAT MAP**
4. **KAT OBJ**



```
KAT.ASM
KAT.COM
KAT.MAP
KAT.OBJ
```

6. Which files are created in the assembling process?
   -KAT.ASM

7. Which files are created in the linking process?
   -KAT.OBJ

8. What is the purpose of /t in the linking process?

The /t option in the linking process tells the linker to create a .COM file instead of an .EXE file. .COM files are smaller, simpler executable files with a size limit of 64 KB.0 To make the linker create a .com file instead of an .exe file. A .com file is a simpler and smaller type of executable compared to an .exe file.

9. Compare DEBUG and TURBO ASSEMBLER use in assembly language programming.

DEBUG is primarily a debugging tool that allows programmers to examine and modify memory, set breakpoints, and step through code, making it essential for testing and troubleshooting.

TASM is an assembler that converts assembly language code into machine code, providing advanced features such as support for macros and various memory models. While DEBUG operates in a command-line interface, which can be less user-friendly, TASM often includes a more intuitive interface for easier code management. Ultimately, DEBUG focuses on runtime analysis, whereas TASM facilitates the development and assembly of executable programs.

10. What is the purpose of org 100h in your program?

Org 100h sets the program's starting address to 100h (which equals 256 in decimal). This is the standard starting point for .COM files, ensuring that when the program is loaded into memory, it begins execution in the correct memory segment.

SUMMARY & CONCLUSION:

Assembly language programs are written and prepared in text editors and typically remain with a '.ASM' extension. Assembler Turbo (TASM) produces multiple files, including ".ASM," ".COM," ".MAP," and ".OBJ," and translates these programs into machine code. The ".ASM" file is created throughout the assembly process, and linking creates the ".OBJ" file. When linking, the '/t' option ensures a '.COM' file, which is easier to understand than an '.EXE' is created.

Programming in assembly language can benefit from the use of DEBUG and Turbo Assembler. In short, simple programs, DEBUG works well, but Turbo With larger and more complex ones, assembler works better. Knowing basics of program assembly, linking, and loading as well as important commands like "org 100h," helps programmers in creating and efficiently executing assembly code.