1. TASM, EDIT, DEBUG and Emu8086 Assembler Tools

1.1 Objective

TASM is one of the well known **8086 Assembler** programs. This experiment will introduce you **TASM**, its input, and output file types.
Our objective covers hands-in experience to use
“**Notepad**” to create an assembler source file,
“**TASM**” to assemble the a source file into an object code
“**TLink**” to link an object code into an executable file.
“**TD**” and “**Emu8086**” debuggers to trace an executable file.

1.2 Introduction

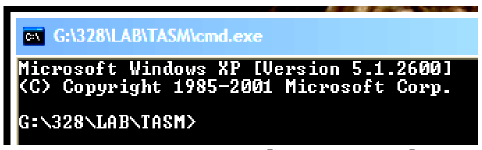
Assembly language is the lowest level of symbolic programming for a computer system. It has several advantages and disadvantages over the higher level programming languages. Assembly language requires an understanding of the machine architecture, and provides huge flexibility in developing hardware/software interface programs such as interrupt service routines, and device drivers. **8086 Turbo Assembler** is one of the well known assembler programs used for PC-XT and AT family computers.

1.2.1. Editing the source file

The source for an assembly program is written into a text file with the extension -.ASM, in ASCII coding. Any ASCII text editor program can be used to write an assembly source file. We recommend to use **NOTEPAD** as a general purpose text editor, or the source editor of the **Emu86**, which is especially tailored to write **8086 Flat ASM** sources for your experiments.

1.2.2. Assembling to an object file

Once the source file is ready for assembling, you will need **TASM** program to be executed on the source file. **TASM** is a quite old program, written for **DOS** environment. Indeed, in most embedded system application DOS operating system is preferred over Windows because Windows is unnecessary, too bulky and too expensive for most embedded applications. In the **Windows** operating system, you can invoke a **DOS** command window by running the “**CMD.EXE**” executable. Figure 1 shows a Command Window, with its typical cursor. You may change the font and the colors of the Command window by the defaults and properties dialog which is opened with a left-click on the windows title. Colors such as screen text black on white, popup text blue on gray, and fonts Lucida-Console 18 point will make your command window much more readable. Whenever you want, you can use **CLS** command of DOS to clear the screen and the screen buffer.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

G:\328\LAB\TASM>
```

Figure 1. A typical Command Window in the Windows Environment.

The Turbo Assembler program (**TASM.EXE**) can be started in the command window by writing TASM <source-file-name> , and transmitting it to DOS using the“ENTER” key. The full syntax of TASM command is:

```
>TASM [options] source [,object] [,listing] [,xref]
```

TASM command line options are shown in Table 1.

Table 1. Possible Switches of the Turbo Assembler Program.

/a,/s	Alphabetic or Source-code segment ordering
/c	Generate cross-reference in listing
/dSYM[=VAL]	Define symbol SYM = 0, or = value VAL
/e,/r	Emulated or Real floating-point instructions
/h,/?	Display this help screen
/iPATH	Search PATH for include files
/jCMD	Jam in an assembler directive CMD (eg. /jIDEAL)
/kh#,/ks#	Hash table capacity #, String space capacity #
/l,/la *	Generate listing: l=normal listing, la=expanded listing
/ml,/mx,/mu	Case sensitivity on symbols: ml=all, mx=globals, mu=none
/n	Suppress symbol tables in listing
/p	Check for code segment overrides in protected mode
/t	Suppress messages if successful assembly
/w0,/w1,/w2	Set warning level: w0=none, w1=w2=warnings on
/w-xxx,/w+xxx	Disable (-) or enable (+) warning xxx
/x	Include false conditionals in listing
/z	Display source line with error message
/zi,/zd	Debug info: zi=full, zd=line numbers only

In DOS and Assembly programming, the names are not case-dependent, which means writing **TASM FIRST**, **Tasm first**, **tasm FIRST** or **tasm firST** does not make any difference.

Assume that you have written the following simple assembly program into a text file with the name **first.asm**. To assemble it into **first.obj** file, you shall simply write the command

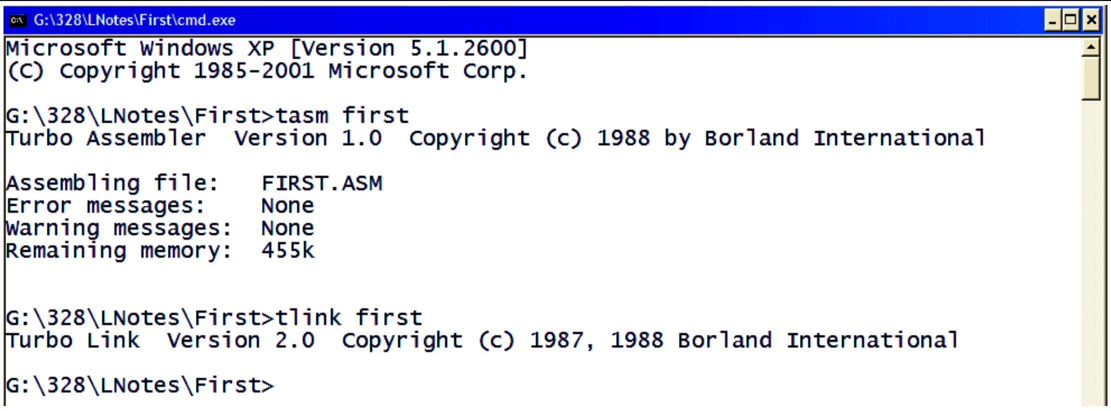
```
>tasm first
```

1.2.3. Linking to an Executable or Command File

The object files contains the program code but some of the labels are still in symbolic form. A linker converts them into the executable file replacing all symbols with their corresponding values. The use of library procedures, and splitting the large programs into modules are possible since a linker can calculate a label referred from a different object file. The file first.obj is converted to an executable by the DOS command

```
>tlink first
```

Figure 2 shows typical command window message after tasm and tlink is executed.



```
G:\328\LNotes\First\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

G:\328\LNotes\First>tasm first
Turbo Assembler  Version 1.0  Copyright (c) 1988 by Borland International

Assembling file:   FIRST.ASM
Error messages:    None
Warning messages:  None
Remaining memory:  455k


G:\328\LNotes\First>tlink first
Turbo Link  Version 2.0  Copyright (c) 1987, 1988 Borland International

G:\328\LNotes\First>
```

Figure 2 Command Window after tasm and tlink are executed.

After running Tlink, you shall find the executable file **first.exe** in your working folder. First.exe terminates with a return to DOS interrupt, without giving any message. An assembly debugging tool can trace what happens during the execution of the first.exe file.

1.2.4. Tracing and Debugging of an EXE file

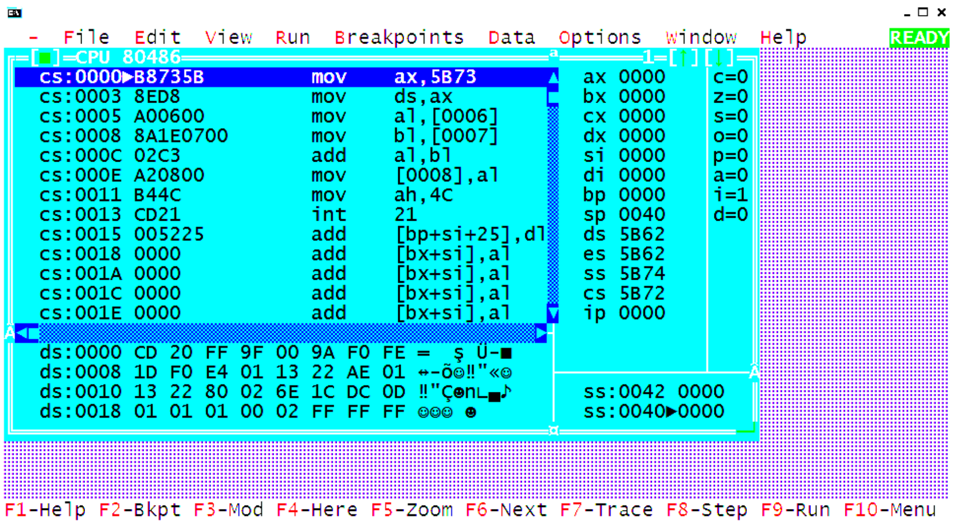
Turbo Debugger, **td.exe**, is an 8086 debugging tool which gives a convenient view of the CPU status, and the memory segments. The command line syntax of TD has options, program-file-name, and arguments

```
>TD [options] [program [arguments]]   -x-  = turn option x off
```

The options of td.exe is shown in Table 2.

Table 2. Command Line Options for Turbo Debugger TD.EXE

-c<file>	Use configuration file <file>
-do,-dp,-ds	Screen updating: do=Other display, dp=Page flip, ds=Screen swap
-h,-?	Display this help screen
-i	Allow process id switching
-k	Allow keystroke recording
-l	Assembler startup
-m<#>	Set heap size to # kbytes
-p	Use mouse
-r	Use remote debugging
-rn<L;R>	Debug on a network with local machine L and remote machine R
-rp<#>	Set COM # port for remote link
-rs<#>	Remote link speed: 1=slowest, 2=slow, 3=medium, 4=fast
-sc	No case checking on symbols
-sd<dir>	Source file directory <dir>
-sm<#>	Set spare symbol memory to # Kbytes (max 256Kb)
-sn	Don't load symbols
-vg	Complete graphics screen save
-vn	43/50 line display not allowed
-vp	Enable EGA/VGA palette save
-w	Debug remote Windows program (must use -r as well)
-y<#>	Set overlay area size in Kb
-ye<#>	Set EMS overlay area size to # 16Kb pages


File Edit View Run Breakpoints Data Options Window Help
READY
CPU 80486
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```
cs:0000►B8735B        mov    ax,5B73            ax 0000   c=0
cs:0003 8ED8          mov    ds,ax              bx 0000   z=0
cs:0005 A00600        mov    al,[0006]          cx 0000   s=0
cs:0008 8A1E0700      mov    bl,[0007]          dx 0000   o=0
cs:000C 02C3          add    al,bl              si 0000   p=0
cs:000E A20800        mov    [0008],al          di 0000   a=0
cs:0011 B44C          mov    ah,4C              bp 0000   i=1
cs:0013 CD21          int    21                 sp 0040   d=0
cs:0015 005225        add    [bp+si+25],dl      ds 5B62
cs:0018 0000          add    [bx+si],al         es 5B62
cs:001A 0000          add    [bx+si],al         ss 5B74
cs:001C 0000          add    [bx+si],al         cs 5B72
cs:001E 0000          add    [bx+si],al         ip 0000

ds:0000 CD 20 FF 9F 00 9A F0 FE =  ş Ú-■
ds:0008 1D F0 E4 01 13 22 AE 01 ↔-õ☺‼"«☺
ds:0010 13 22 80 02 6E 1C DC 0D ‼"Ç☻n∟▄♪        ss:0042 0000
ds:0018 01 01 01 00 02 FF FF FF ☺☺☺ ☻           ss:0040►0000
```

Figure 3. The turbo debugger started with first.exe file.

Entering the command

```
>td first
```

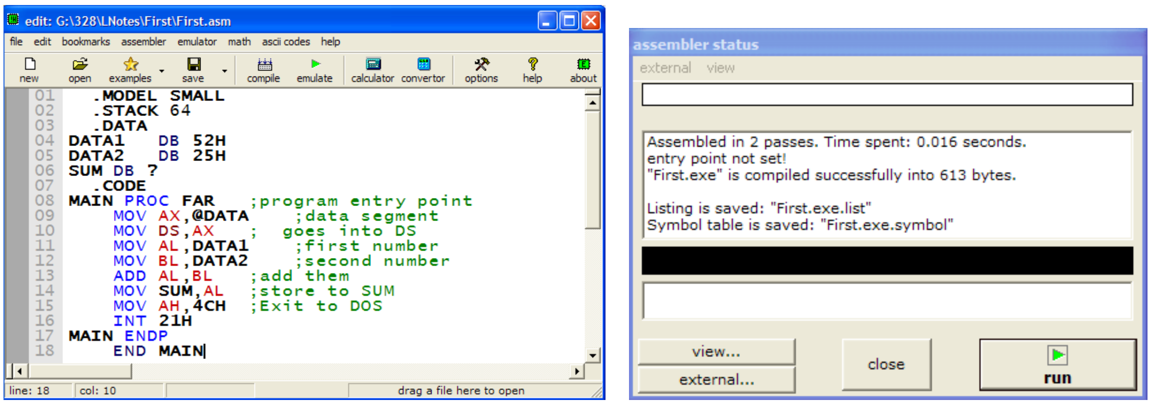
into the command window will start the debugger to load the executable **first.exe** to its memory space. The screenshot of **TD** is shown in Figure 3. In Turbo debugger, you can execute the instructions step by step and trace the execution of the code. Any message written to the screen will invoke the screen display mode to let you observe the message.

1.2.5. Emu86 IDE

An Integrated Development Environment (**IDE**) provides a convenient environment to write a source file, assemble and link it to a **-.COM** or **-.EXE** file, and trace it in both source file, and machine code. Emu86 is an educational IDE for assembly program development. You can download the latest version of EMU86 from the web page **www.emu8086.com**. It is a Windows program, and will run by dragging an **-.ASM, -.OBJ, -.LST, -.EXE**, or **-.COM** file into the **emu86** shortcut icon. By this action, asm or lst files will start the 8086 assembler source editor, while obj and exe files starts the disassembler and debugger units.

1.2.6. EMU8086 Source Editor

The source editor of EMU86 is a special purpose editor which identifies the 8086 mnemonics, hexadecimal numbers and labels by different colors as seen in Figure 4.


edit: G:\328\LNotes\First\First.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
line: 18 col: 10 drag a file here to open
assembler status
external view
view... external... close run

```
   .MODEL  SMALL
   .STACK 64
   .DATA
DATA1    DB  52H
DATA2    DB  25H
SUM DB  ?
   .CODE
MAIN PROC  FAR     ;program entry point
     MOV AX,@DATA      ;data segment
     MOV DS,AX     ;  goes into DS
     MOV AL,DATA1      ;first number
     MOV BL,DATA2      ;second number
     ADD AL,BL     ;add them
     MOV SUM,AL    ;store to SUM
     MOV AH,4CH    ;Exit to DOS
     INT 21H
MAIN ENDP
     END MAIN
```

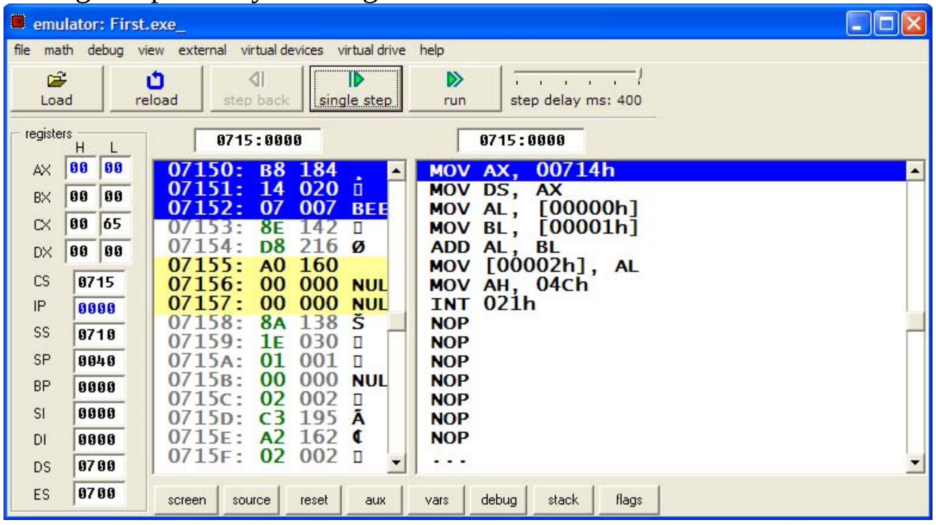
```
Assembled in 2 passes. Time spent: 0.016 seconds.
entry point not set!
"First.exe" is compiled successfully into 613 bytes.

Listing is saved: "First.exe.list"
Symbol table is saved: "First.exe.symbol"
```

a) b)

Figure 4. a) EMU8086 Source Editor, and b) assembler status report windows.

The compile button on the taskbar starts assembling and linking of the source file. A report window is opened after the assembling process is completed. Figure 5 shows the emulator of 8086 which gets opened by clicking on emulate button.



```
registers
     H   L
AX   00  00
BX   00  00
CX   00  65
DX   00  00
CS   0715
IP   0000
SS   0710
SP   0040
BP   0000
SI   0000
DI   0000
DS   0700
ES   0700
```

```
0715:0000
07150: B8 184 .
07151: 14 020 
07152: 07 007 BEE
07153: 8E 142 
07154: D8 216 Ø
07155: A0 160
07156: 00 000 NUL
07157: 00 000 NUL
07158: 8A 138 Š
07159: 1E 030 
0715A: 01 001 
0715B: 00 000 NUL
0715C: 02 002 
0715D: C3 195 Ã
0715E: A2 162 ¢
0715F: 02 002 
```

```
0715:0000
MOV AX, 00714h
MOV DS, AX
MOV AL, [00000h]
MOV BL, [00001h]
ADD AL, BL
MOV [00002h], AL
MOV AH, 04Ch
INT 021h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Figure 5. first.exe in the emulator window of EMU8086 debugging environment

Emul8086 environment contains templates to generate command and executable files. Another benefit of Emul8086 is its emulation of a complete system, including the floppy disk, memory, CPU, and I/O ports, which raises opportunity to write custom bios and boot programs together with all other coding of a system. More over, its help is quite useful even for a beginner of **asm** programming.

1.2.7. EMU8086 / MASM / TASM compatibility

Syntax of emu8086 is fully compatible with all major assemblers including *MASM* and *TASM*; though some directives are unique to this assembler.

1) If required to compile using any other assembler you may need to comment out these directives, and any other directives that start with a '#' sign:
```
#make_bin#
#make_boot#
#cs=...#
```
   etc...

2) Emu8086 ignores the ASSUME directive. manual attachment of CS:, DS:, ES: or SS: segment prefixes is preferred, and required by emu8086 when data is in segment other then DS. for example:
```
mov ah, [bx]        ; read byte from DS:BX
mov ah, es:[bx]     ; read byte from ES:BX
```

3) emu8086 does not require to define segment when you compile segmentless COM file, however MASM and TASM may require this, for example:
```
name test
CSEG    SEGMENT     ; code segment starts here.
ORG 100h
start:  MOV AL, 5   ; some sample code...
        MOV BL, 2
        XOR AL, BL
        XOR BL, AL
        XOR AL, BL

        RET
CSEG    ENDS        ; code segment ends here.
END     start       ; stop compiler, and set entry point.
```

4) entry point for COM file should always be at 0100h, however in MASM and TASM you may need to manually set an entry point using END directive even if there is no way to set it to some other location. emu8086 works just fine, with or without it; however error message is generated if entry point is set but it is not 100h (the starting offset for com executable). the entry point of com files is always the first byte.

5) if you compile this code with Microsoft Assembler or with Borland Turbo Assembler, you should get *test.com* file (11 bytes). Right click it and select send to and emu8086. You can see that the disassembled code doesn't contain any directives and it is identical to code that emu8086 produces even without all those tricky directives.

6) emu8086 has almost 100% compatibility with other similar 16 bit assemblers. the code that is assembled by emu8086 can easily be assembled with other assemblers such as TASM or MASM, however not every code that assembles by TASM or MASM can be assembled by emu8086.

7) a template used by emu8086 to create **EXE** files is fully compatible with *MASM* and *TASM*.

8) The majority of **EXE** files produced by *MASM* are identical to those produced by *emu8086*. However, it may not be exactly the same as TASM's executables because *TASM* does not calculate the checksum, and has slightly different EXE file structure, but in general it produces quite the same machine code. There are several ways to encode the same machine instructions for the 8086 CPU, so generated machine code may vary when compiled on different compilers.

9) Emu8086 integrated assembler supports shorter versions of **byte ptr** and **word ptr**, these are: **b.** and **w.** For *MASM* and *TASM* you have to replace **w.** and **w.** with **byte ptr** and **word ptr** accordingly.

for example:
```
lea bx, var1
mov word ptr [bx], 1234h ; works everywhere.
mov  w.[bx],  1234h           ;  same  instruction  /  shorter  emu8086
syntax.
hlt

var1  db  0
var2  db  0
```

10) LABEL directive may not be supported by all assemblers, for example:
```
TEST1 LABEL BYTE
; ...
LEA DX,TEST1
```
the above code should be replaced with this alternative construction:
```
TEST1:
    ; ...
MOV DX, TEST1
```
the offset of TEST1 is loaded into DX register. this solutions works for the majority of leading assemblers.

1.3 Experimental Part

In this experiment you will use TASM, TLINK, and EMU8086 to generate an executable from an assembly source, and to trace the step-by-step execution of the executable in TD debugger and in EMU8086 emulator

1.3.1. Writing a Source File

Objective: to practice writing and editing an ASCII assembly source file using notepad.
Procedure: Generate a folder asm. Copy the files tasm.exe, tlink.exe, td.exe into asm folder. Generate a working folder with name **exp1**, and start a text file in your working folder In the explorer while folder is open

- click on right button of mouse, and
- select new, select text document. "New Text Document.txt" will be generated.
- Rename it "**exp1.asm**"

Now, you have an empty text file, with the name **exp1.asm**. Use **windows-start** > **all-programs** > **accessories** > **notepad** to open the Notepad text editor. Drag the file **exp1.asm** to the title-bar of the Notepad. The title will change to **exp1.asm – Notepad**. It means that you successfully opened the file **exp1.asm** for editing in notepad. Write the following source program into the edit window.

------file: exp1.asm-----
```
; STUDENT NAME and SURNAME:
; STUDENT NUMBER:

TITLE PROG2-2 (EXE) PURPOSE :ADD 4 WORDS OF DATA
PAGE  60,132
      .MODEL SMALL
      .STACK 64
;----------------------------------------------------------
      .DATA
DATA_IN     DW 234DH,1DE6H,3BC7H,566AH
      ORG 10H
SUM   DW ?
;----------------------------------------------------------
      .CODE
MAIN  PROC FAR    ;THIS IS THE PROGRAM ENTRY POINT
      MOV   AX,@DATA          ;load the data segment adress
      MOV   DS,AX             ;assign value to DS
      MOV   CX,04             ;set up loop counter CX=4
      MOV   DI,OFFSET DATA_IN ;set up data pointer DI
      MOV   SI,OFFSET SUM
      MOV   BX,00             ;initialize BX
ADD_LP:
      ADD   BX,[DI]           ;add contents pointed at by [DI] to BX
      INC   DI                ;increment DI twice
      INC   DI                ;to point to next word
      DEC   CX                ;decrement loop counter
      JNZ   ADD_LP            ;jump if loop counter not zero
      MOV   SI,OFFSET SUM    ; SI points SUM
      MOV   [SI],BX           ;store BX to SUM in data segment
      MOV   AH,4CH            ;set up return
      INT   21H               ;return to DOS
MAIN  ENDP
      END MAIN                ;this is the program exit point
```
------end of file ------

Use tabs to start the mnemonics at the same column.

Reporting:

Start a text file (you may use *notepad*) with name **exp1.txt**. Fill in the following title to your text file.

```
CMPE 323 Experiment-1 Report. <your name surname, student number>
PART1 Assembly source file
```

Copy-and-paste your exp1.asm into your report file.

```
; STUDENT NAME and SURNAME: ALI VELI
; STUDENT NUMBER:   012345

TITLE  PROG2-2 (EXE) PURPOSE :ADD 4 WORDS OF DATA
PAGE   60,132
       .MODEL SMALL
…
…
```

Keep your report file in a safe place until you complete the experiment and e-mail it to the specified address.

1.3.2. Assembling with TASM

Objective: Assembling the source file with TASM, and tracing it in TD.

Procedure: You have already written the source file **exp1.exe** .

- Organize a folder structure such as
  ASM folder contains
  files **TASM.EXE**, **TLINK.EXE**, and **TD.EXE**.
  folder **exp1**, which contains **exp1.asm** and **exp1.bat.**

-Edit **exp1.asm** to contain the complete source text by copy and paste. Fill your student name and number to the first two lines.

-Edit **exp1.bat** to have the following text lines in it.

```
..\tasm -l exp1
pause
..\tlink exp1
pause
..\td  exp1
pause
```

-Click on **exp1.bat** to execute assembler. You will observe a DOS window opened, and tasm executed on **exp1**.asm, with the list option active. DOS window will pause and will allow you to read the messages generated by TASM. You will observe **exp1.obj**, **exp1.lst**, and **exp1.map** files generated in folder **exp1**.

-If you press on space-bar, bat file will continue to execution, and it will execute the linker tlink on **exp1.obj**. Tlink will generate **exp1.exe** file into the **exp1** folder. Batch file will pause until you press the space-bar.

-Press the space-bar again to execute turbo debugger on **exp1.exe** file. In the debugger, you can trace the execution by executing each line of the assembly program stepwise.

Reporting:

In **td** read the hexadecimal contents of the program code **exp1.exe** (28 bytes), and the contents of the memory location cs:0009. Start `PART2` in your report file, and fill in (as text, i.e., `A3 02` etc)

```
PART2
B8 68 5B 8E D8 ...
cs:0009 contains ....
```

Then open exp1.lst, which is generated by turbo assembler in a text editor (notepad). Copy-and-paste the first page of the listing into your report file

```
exp1.lst contains ------------------------
Turbo Assembler  Version 1.0       01/13/11 11:32:32        Page 1
EXP1.ASM

      1                            ; STUDENT NAME and SURNAME:
      2                            ; STUDENT NUMBER:
      3
      4 0000                                .MODEL SMALL
```

```
 5 0000                              .STACK 64
 6                           ;---------------------------------------------------------
 7 0000                              .DATA
 8 0000  234D 1DE6 3BC7 566A    DATA_IN DW 234DH,1DE6H,3BC7H,566AH
 9                                   ORG 10H
10 0010  ????                   SUM     DW ?
11                           ;---------------------------------------------------------
12 0012                              .CODE
13 0000                         MAIN    PROC FAR        ;THIS IS THE PROGRAM ENTRY POINT
14 0000  B8 0000s                       MOV     AX,@DATA                ;load the data segment address
      ...
      ...
```

Save your report file in a safe place until you complete the experiment and e-mail it to the specified address.

1.3.3. Assembling with Emu8086

Objective: Assembling a source file with Emu8086 assembler/emulator

Procedure:

-Start Emu8086, and close the welcome window. Use “**open**” in taskbar to start the file browser. Select the folder **exp1**, and open **exp1.asm**.

-Emu8086 cannot use **title**, **page**, and **org** directives. Put a semicolon to make them a comment line. Then, use emulate in taskbar to assemble, and start the emulator window with the **exp1.exe**.

-Use the taskbar-button “**single step**” to execute each line of the assembly source.

Reporting

In `PART3` of your report answer the following questions in full sentences.

a) How many times the loop passes through the **add** instruction?

b) What is the effective address of the **add** instruction in the code segment?

After completing the experiment, write an e-mail that contains

```
Please find the attached report file of experiment 1.
Regards.
012345 Ali Veli
```

attach the report file to the e-mail and send it

- from your student-e-mail account
- to the e-mail address **cmpe323lab@gmail.com**
- with the subject: **”exp1”.**

Late and early deliveries will have 20% discount in grading. No excuse acceptable.