

# Laboratory Information for 03–60–266

Dr. Alioune Ngom

## 1 Using Borland's Turbo Assembler Command Lines

All of our programs will be Win32 console applications. These are 32-bit protected-mode programs that run in a MS-DOS box (but they are not MS-DOS programs).

To perform simple I/O operations, you should use the set of macros that are in the `Cs266.inc` file. Just place the directive `include Cs266.inc` immediately after the `.Model Flat` directive in your source file. The course notes explains how to use these macros.

In the labs, we will assemble our programs with Borland's TASM32 assembler. To assemble and link in a single command, it is convenient to use the C compiler front-end. Assuming that your assembly language source file is `Hello.asm`, to assemble, link, and produce an executable file `Hello.exe`, just use the `bcc32` command from the Win32 console (i.e.: the MS-DOS box) prompt

```
bcc32 -v Hello.asm
```

Note that `bcc32.exe` is the 32-bit C/C++ compiler of Borland C++ and Borland C++ Builder. But since our source file has the `.asm` extension, this command will call `tasm32` to assemble your program (in case sensitive mode) and then call Borland's incremental linker (called `ilink`) to link your program together with the C standard library (and other libraries that will give you access to the Win32 API) and finally produce the executable file `hello.exe`.

The `-v` option is to generate full debugging information. To display the list of all available options just type `bcc32` without any arguments. To store this information into a file named `bcc32-options.txt`, just redirect the output to this file; i.e.: just type

```
bcc32 > bcc32-options.txt
```

Do not forget to put your source file into the same directory as your `Cs266.inc` file.

To execute your `Hello.exe` program you can either type `Hello` from the Win32 console or double-click on the `Hello.exe` icon. However, double-clicking on the icon will first create a MS-DOS box, then it will run `Hello.exe` into this box and then it will close the MS-DOS box when the program has finished running. Hence to prevent your program of running until completion just insert the `PROMPT` macro just before the `RET` instruction in your source file. The `PROMPT` macro will just prompt the user to type a key to continue execution.

Of course you will need an editor to produce your source files. You can use `NotePad` but it is very limited. You can use the editor of Borland C++ 5.0 or Borland C++ Builder (very good), if you installed them. In the lab, you can use `TextPad`, which is a good editor. There are other good editors such as `PFE` (*Programmer's File Editor*, can be downloaded for free at <http://my.oh.voyager.net/~csomervi/pfe.html>) and `UltraEdit32` (from <http://www.idmcomp.com>, an excellent editor but not free). There are many good editors you can download free at <http://www.thefreecountry.com/programming/editors.shtml>.

Finally we will need a debugger. I recommend that you use Borland's Turbo Debugger `td32`. This debugger runs in a MS-DOS box (so it looks rather clumsy) but it has all the functionality that we need (and you can use your mouse). It is installed in the labs. To debug `Hello.exe`, just type

```
td32 Hello
```

A number of convenient windows are available from the *View* menu, namely

- The *Registers* window to view the content of registers
- The *Numeric Processor* window to view the content of the FPU registers

- The *CPU* window to view both your source code and the machine language equivalent
- The *Watches* window to view the content of variables and memory

A number of convenient commands are available from the *Run* menu, namely

- The *Trace Into* (F7) command to single step into your program
- The *Step Over* (F8) command to single step into your program but avoid entering into procedures
- The *Instruction Trace* (Alt-F7) command to trace to the next instruction and force the entrance into procedures
- The *Run* command to run your program up to the position of the next breakpoint (you define a breakpoint just by clicking on the point to the left of the instruction where you want your program to stop). If you have not defined any breakpoints, this will run your program to completion.

## 2 Potential Problem with TD32 under Windows2000 and WindowsNT

It may happen that `td32` complains that it cannot find the symbol table even though you have compiled with the `-v` option. If this happens, then you will not be able to see your source code while debugging. Apparently, `ilink32` sometimes put a time/date stamp for the `.exe` and `.tds` files such that `td32` will think that the `.tds` file is out of date with respect to the `.exe` file (although both files are correct). The *touch* utility (distributed with the `FreeCommandLineTools` which can be downloaded free from <http://www.borland.com>) apparently reassigns good time/date stamps to these files under Windows2000. After compiling, just do

```
touch hello.exe hello.tds
```

Then TD32 should be able to read the symbol table. That solution may not work under WindowsXP. In that case you may write a program that sets the time and date of the `.tds` file one second older than the corresponding `.exe` file. This apparently always works. Here is an example, `touchtds.exe.zip` (distributed with `FreeCommandLineTools`), of a program that does exactly that; use it in the same way as `touch.exe`.

## 3 Getting and installing the free software from Borland

`bcc32` and `td32` are distributed for free by Borland under the names `FreeCommandLineTools` and `TurboDebugger` respectively. Just download the provided `.exe` files and install them on your PC. Assuming that you have installed `FreeCommandLineTools` and `TurboDebugger` in the `C:\borland\bcc55` directory, you will then need to do the following

1. Place the following files in your `C:\borland\bcc55\bin` directory

```
bcc32.cfg
```

and

```
ilink32.cfg
```

2. Add `C:\BORLAND\BCC55\BIN` to your Path environment variable

- For Win98/95: you need to do this by changing your `autoexec.bat` file in your `C:\` directory. For this, just add the following line at the end of your `autoexec.bat` file

```
PATH=%PATH%;C:\BORLAND\BCC55\BIN
```

If you do not have an `autoexec.bat` file, then create one that contains the line above and place it in `C:\`. Then, to update your path variable just type `autoexec.bat` at the `C:\>` prompt

- For Windows2000 (and WindowsNT): Right click on *My Computer* and choose *Properties*. Then, on the *Advanced* tab, click on *Environment Variables*. Then edit the Path System variable (not the PATH user variable for administrator) by adding to it: C:\BORLAND\BCC55\BIN.

The only thing that will be missing is Turbo Assembler which is contained in the single file named `tasm32.EXE`. This file (about 184 KB in size) should be in the C:\BORLAND\BCC55\BIN directory. Apparently, Turbo assembler is not free and is only distributed with C++ Builder professional . . . .

## 4 Additional Notes

`Bcc32`, `tasm32`, and `ilink` are also part of the distribution of C++ Builder Professional (version 3 and higher). Hence you will be able to assemble and link programs with `bcc32` if you have this product. But you will not be able to debug with `td32` because it is not part of the distribution of C++ Builder; rather, it is part of the distribution of Turbo Debugger (free from Borland ). You can also use the IDE of Borland C++ Builder (version 3 or higher) to assemble, link, and debug assembly language programs. However, C++ Builder 3 has no CPU debug window so you will not be able to view registers (not very useful). But C++ Builder 4 (and higher) has a CPU window that will show registers but not the FPU registers (so it will not be helpful when you will try to debug programs that use the FPU). I do not know anything about the IDE of Borland C++ 5.0.

**Using C++ Builder 4 (or higher) to Assemble, Link, and Debug** To use the IDE of C++ Builder 4 to assemble, link and debug assembly language programs, you can do the following: First place both your source file, say `Hello.asm`, and your `Cs266.inc` file in the same directory (i.e.: inside your user directory). After launching C++ Builder 4, choose *New...* from the *File* menu. Choose the *Console Wizard* and make sure that you have selected *Console* and "EXE" from the options offered by the Wizard and then click *Finish*. Now, choose *Save Project As...* from the *File* menu and save it under a name, say `ProjectHello.bpr`, in the same place as your source file (here `Hello.asm`). Then choose *Add to Project...* from the *Project* menu and select your assembly source `Hello.asm`. The IDE constantly reads and writes into two files, here they should be named `ProjectHello.cpp` (a C++ source file) and `ProjectHello.bpr` (your makefile). You absolutely need these two files: do not delete them. To view your source file under this IDE just use *Open...* from the file menu. Next, edit your `ProjectHello.cpp` file so that it contains "only" the following lines (yes, we do get rid of the `main` function since you already defined a `main` function in `Hello.asm`):

```
#include <condefs.h>
USEASM("hello.asm");
#define main
```

Then save again your project with the *Save* command. Next, choose *Build ProjectHello* under the *Project* menu to assemble and link your program. Then choose *Run* to run your program. Your program's output will appear on the Win32 console but the console will immediately disappear. To avoid this disappearance, use the `PROMPT` macro just before the `RET` instruction in your source file. Then the console will disappear only after you type a key. To debug your program just choose the *Trace Into* command from the *Run* menu and the CPU window should appear. The debugger is very similar to `td32`. Other debugging windows are available from the *Debug Window* command under the *View* menu. Note the absence of a *Numeric Processor* window.