A Guide to Pep/8

1. What is it?

An integrated environment for creating, assembling and running programs for the Pep/8 virtual machine,

2. Where can I get it?

Pep/8 is under active development; the developers of Pep/8 have a web site

http://code.google.com/p/pep8-1/

from which you can download the latest version. Versions are available for Linux, Mac and Windows. Versions of Pep/8 are installed in the Computer Science labs (both Mac and Windows) and on some systems in the Broome Library.

3. How do I use it?

3.1 Entering programs

You can enter an object program written in hex machine code (see chapter 4) or enter a source program written in Pep/8 assembly language (see Chapters 5 and 6). The editor is pretty standard; for example you can cut and paste sections of programs.

An example of a machine code program is

41 00 04 00 48 65 6C 6C 6F 20 57 6F 72 6C 64 00
zz

Hex programs should be entered in the Object Code window. Hex pairs should be separated by a single space or a new line and the last pair of characters should be zz.

An example of an assembly language program is

stro mess,d
stop
mess: .ascii "Hello World\x00"
.end

Assembly language programs should be entered in the Source Code window.
3.2 Running programs

In the case of an object program in the Object Code window, select Run Object from the Build menu.

In the case of a source program entered in the Source Code window, select Run Source from the Build menu. If there are syntax errors in the program, error messages will appear in the Source Code window. (In the Edit sub-menu there is a command to remove the error message text from the window) If the source program has no syntax errors then its translation into hex will appear in the Object code window and a listing showing the source and object codes together will appear in the Assembler Listing window.

Input options

If your program requires user input you can pre-enter the all the input required before you run it or you can enter it as the program needs it.

If you wish to pre-enter all the input, type it into the Input window and select the Batch tab. If you would prefer to enter the input as the program is running then select the Terminal I/O tab. Note that in the case of this interactive input, the system prompts with an underscore when it is waiting for you to type something.

3.3 Debugging programs

There is a comprehensive guide to debugging programs under the Help option in the main menu. You can single step through a program for example and set break points in assembly language.

<Example>

An alternative to using the debugging tools is to insert simple output instructions in your program. These outputs indicate the flow of control and the value of key registers and memory locations.

<example - program getting wrong max – biggest of N from rubric? Output L from inside lop>