Chapter 1: Starting Assembler

In which we consider why we should pursue this study.

Why Z?

This book is about assembly language programming on IBM's System/z, a family of mainframe/enterprise machines that evolved from a historically significant machine called the IBM System/360, which appeared in the early 1960s. Over a remarkable 50+ year evolution, this family of machines has continued to drive the business applications of the world's largest corporations.

The study of assembly language in computer science programs has changed greatly over the years. What was once a course in the curriculum, is now a week's topic in a computer organization class. Times change. But if you consider yourself a programmer, there's a great value in learning the assembly language of the machine you work on daily. Assembly language mirrors the machine language of a computer – the raw instructions that are invoked by every language used on your box. Learning the assembly language of a machine will make you a better high-level language programmer. If you want to know what a machine is really doing, study its assembly language.

In the academic world, the use of enterprise machines for instruction declined for many years. That is changing. IBM has done much to provide access for academic institutions through the IBM Academic Initiative Program: https://www.ibm.com/academic/home Through a variety of efforts, IBM encourages mainframe training and makes enterprise machines available to colleges and universities all over the world. Your school can apply to have access to a world-class machine for free.

Once you have access to an enterprise machine, you'll need some software to connect to it. Traditionally, that has been a TN3270 emulator. There are a number of these commercially available. I use and can recommend:

Tom Brennan's Vista software for Windows:

https://www.tombrennansoftware.com/download.html

Mocha TN3270 for macOS:

https://mochasoft.dk/tn3270macx.htm

In recent years, connectivity to enterprise machines has changed because of The Open Mainframe Project:

https://www.openmainframeproject.org/

Using a combination of Zowe and Visual Studio Code, you can now access a mainframe using a modern interface. The enterprise machine you access has to be prepared to support this kind of connection, so this won't work on all computers. As a beginning assembler programmer, I would suggest starting with a 3270 emulator.

An alternative to remotely connecting to an enterprise machine is to download and use the z390 Portable Mainframe Assembler and Emulator:

http://www.z390.org/

Using this product, you can assemble and run programs on your laptop.

You might be thinking: *What's the market for mainframe skills these days?* Isn't this a little like learning Latin? You might be surprised to learn that most Fortune Five Hundred companies, particularly in banking and insurance, run their companies on mainframes. These same companies have an aging workforce of mainframe programmers who will soon retire. The demand for mainframe skills is growing, and the supply of programmers is shrinking. Contrast this to a similar supply and demand for Java or C# programmers. There is a substantial market for programmers with enterprise skills in many of the world's largest companies.

Besides traditional mainframe languages like COBOL, REXX, PL/I, and Assembler, System/z machines also run Linux and support the languages you probably studied in school – Java, C, and C++. System/z machines are some of the most complex and architecturally interesting machines on the planet, and the developing world of Big Data will only increase the demand for these kinds of machines. The truth is that many companies desperately need programmers with both mainframe and client/server skills.

There's another, nobler reason to master assembler on System/z: learning any assembly language will broaden your knowledge of how computers work at a very deep level. Assembly language skills will make you a better Java, Ruby, or C++ programmer, even if you never go on to program in assembler. And if you have to tackle an assembly language, there's hardly a better one to start with than IBM's HLASM (High-Level Assembler).