CS 203 F18 Project 1: Building An Assembler
Due:Sunday, November 4, 2018

C W Liew

October 14, 2018

Goals

The goals of this project are:

1. gain experience in designing and implementing C programs,
2. gain a better understanding of the process of converting/compiling an assembly language program into an executable binary.

Description

In this project, you will design and implement (in the C programming language) an assembler that will compile a program written in assembly language for the MC6502 processor. The assembly language specification is given in the accompanying manual (starting at page 202 in the PDF - Appendix B) Your program will read in an assembly language program (filename given as part of the command line) and generate two files:

- this file contains the executable form of the program (in binary). There will be a header section that has the following format:

  ```
  STARTHEADER
  <symbol> <location>
  ...
  ENDHEADER
  ```

  This will be followed by the binary image. The entries in the header section correspond to the labels in the assembler language program and specify the memory address of each label. Each label in the program has an entry in the header.

- this file contains the text equivalent of the executable, with each line showing the printable version of each byte in hexadecimal format.

The assembly language program will have the suffix ".asm", the executable will have the suffix ".exc" and the printable version will have the suffix ".prt" Some examples of 6502 programming are provided.

1 General Program Flow

Other than reading and writing files, and translating mnemonics (operations) into binary, the program will have to construct and maintain a symbol table. The table contains the symbols from the assembler program and the addresses of the symbols. Each symbol corresponds to a label - location of code or a variable - in the program. Sometimes symbols are used before they are defined so one of the things that your program will have to do is to keep track of those uses and then update them when the address of the symbol is defined. There are two general ways of doing this:

1. The first way is to go over the assembler program twice. On the first pass, the program does not keep track of the usage of symbols but only enough work to correctly determine the location of every symbol. The second pass does the rest, since the address of every symbol is known.

2. The second way uses only one pass. When encountering the use of a symbol that has not yet been defined, a flag is set indicating this and the program maintains a linked list of places where the symbol is used. When the address is defined, the program checks the symbol table and then uses the linked list to update all places where the symbol has been used.

You can choose any method that you would prefer.

2 More Information

More information on the 6502 can be found on the Internet. Some examples include:

- `http://skilldrick.github.io/easy6502/`
- `https://dwheeler.com/6502/oneelkruns/asm1step.html`
- `http://www.6502.org/tutorials/6502opcodes.html` - a complete listing of opcodes that you can use as an alternative to Appendix B

3 Grading

Your program will be graded on the following criteria:

- functionality - how much of the specified functionality works?
- design - is your program decomposed in an appropriate manner?
- documentation - how well commented is the program