**Programming Assignment 5 -- CS429 Fall 2013**

Due Date: 10 November 2013

**Purpose**

The purpose of this assignment is to introduce you to how an assembler works, and to give you experience in reading and modifying a large existing program. Also, some experience in binary object file formats.

**Program Specification**

You are working at TIC, and your manager tells you that a key employee left and his project is unfinished. He wants you to finish it.

At [asm8.tar](http://www.cs.utexas.edu/users/peterson/src/asm8/asm8.tar)is a partially working version of a PDP-8 assembler. This is needed to ship to customers on 10 Nov. It mostly works, but has a couple of problems:

* The previous coder had used double-slashes (//, like C++) for comments, but the PDP-8 documents use a single slash (/)
* The previous coder had used colon (:) for labels, but the PDP-8 documents use a comma (,).
* The object file format needs to be changed.

The first two changes are minor, but this last could be a problem. The previous coder generated some lame ASCII version of the output object file, and, in the beta version, the customers were all confused and worried about where the EP was, and extra spaces or newlines and such. So we want to change it to a binary format.

In the binary format, the first 4 characters of a PDP-8 object file will be "OBJ8". Next, we will have two bytes which will store the 12-bit starting address for the object file. The PDP-8 has 12 bit words, so we need at least 2 bytes. We will put the high-order 6 bits in the first byte and the low-order 6 bits in the second byte. This same representation (two bytes, 6-bits in each, high-order first), will be used for all PDP-11 addresses and 12-bit values in the object file format.

After the magic number ("OBJ8") and the entry point address (2 bytes) will be a series of blocks. There will be as many blocks as necessary. Each block will be a set of bytes. The first byte (n) is the number of bytes in the entire block, including the first byte. So this first byte cannot be less than 1, and is of no practical value if it's less than 5. After this first byte will be a PDP-8 address (in the two byte format of upper 6 bits/lower 6 bits), followed by a set of values. The values will be PDP-8 12-bit values that are to be stored, starting at the address given, and continuing word after word, until we have exhausted the number of bytes at the beginning of the block.

See [objspec.gif](http://www.cs.utexas.edu/users/peterson/prog5/objspec.gif)

**Example**

The previous ASCII object file:

EP: 067 064: 000 065: 000 066: F02 067: 664 068: 275 069: E21 ...

would be represented as (printing the binary object file in hex):

4F424A380127330124000000003C02192409353821 ...

If we split this by structure we get:

4F424A38 0127 33 0124 0000 0000 3C02 1924 0935 3821 ...

* The first 4 bytes are 4F424A38, which is "OBJ8" in ASCII.
* The next 2 bytes (0127) are a 12-bit address, split into two 6-bit pieces: 000001 from the first byte, and 100111 in the second, for 000001100111, which is 0x67, the starting address (EP).
* The first block is of length 0x33 bytes (51 bytes)
* The address for this block is 0124, which, again, is a 12-bit address, split two 6-bit pieces, and decodes as 000001 100100 = 0x64.
* Now we have 51 bytes - 1 (for the length) - 2 (for the address) = 48 bytes of data to be loaded. Each two bytes is one 12-bit word, so 24 values. The first value (0000) goes in 0x64, the second value (0000) goes into 0x65, the third value (3C02 -> F02) goes in 0x66, and so on.

**Test Cases**

The following PDP-8 assembly language programs are available for testing. You can also write your own, reusing subroutines from these if you want.

* [all.asm](http://www.cs.utexas.edu/users/peterson/prog5/all.asm)with output object file [all.obj](http://www.cs.utexas.edu/users/peterson/prog5/all.obj)
* [gar.asm](http://www.cs.utexas.edu/users/peterson/prog5/gar.asm)with output object file [gar.obj](http://www.cs.utexas.edu/users/peterson/prog5/gar.obj)
* [jan.asm](http://www.cs.utexas.edu/users/peterson/prog5/jan.asm)with output object file [jan.obj](http://www.cs.utexas.edu/users/peterson/prog5/jan.obj)
* [prime.asm](http://www.cs.utexas.edu/users/peterson/prog5/prime.asm)with output object file [prime.obj](http://www.cs.utexas.edu/users/peterson/prog5/prime.obj)
* [test.asm](http://www.cs.utexas.edu/users/peterson/prog5/test.asm)with output object file [test.obj](http://www.cs.utexas.edu/users/peterson/prog5/test.obj)

**Submission**

You need to submit files to create an executable called asm8 for program5. More specifically, for submission you should use

turnin --submit zxy program5 \*.h \*.c [Mm]akefile

Turn in enough that we can say "make asm8" to build an executable for testing. This can be a single C file (asm8.c) or it can be multiple C files and header files, with a Makefile that knows how to "make asm8".

Due Date: 10 November 2013