Kevin Gomes
Dr. Baudet
CSC 411
7 December 2017

Assignment 8

The figure in the reading simply has a list of instructions with binary encoding for each. But I was not sure with the assignment instructions, and decided to code this in assembly. Thus I will provide both.

Step 1.1: Provide machine code for Shift Left and Shift Right (limiting to 15 bits).
Instruction in format of figure:

Binary	**Mnemonic**	**Instruction**	**Meaning**
1111 1111 0000 nnnn	**SHFL**	**Shift left**	**ac := ac << n**
1111 1111 0001 nnnn	**SHFR**	**Shift right**	**ac := ac >> n**

Step 1.2: Write micro-code to perform a SHFL and SHFR. For SHFL, I start at address 100 and can only go to 120. For SHFR, can go from 120 to 255 (or 6).

(NEXT PAGE)

Shift Left (by n). Note that the n we load is not the same as n in COND, so I will call the n in COND capital n (N) if I use it.

```
100: a := ir && smask; //Load n into a. Basically, ignore the first 12 bits
                       //(which is the instruction) and only load the last 4 bits
                       //(n, or how much to shift by).

101: ac := ac + ac << 1; //Shift by 2. Now we need to subtract n by 2…
102: a := a + (-1);              //Subtracts n by 1.

103: alu := a; if Z then goto 107;  // Checks if n = 0. If so,
                                    //have shifted too far and must shift right by 1...

104: a := a + (-1);                 //a could never be -1 here, as we already checked for 0
                                    //above. Now we subtract again.

105: alu := a; if Z then goto 0; //Check for n = 0 again (so if Z = 1).
                                    //If Z = 1, we are done. So go back to start to increase pc.

106: goto 101;                      //loop back.

107: ac := ac >> 1;                 //We have shifted 1 too many before, and must go back 1.
                                    //By getting to this address, we assume we must of jumped.
                                   //That is the only way to get here.

108: goto 0;                        //Must be done with the shift now, so go back to
                                    //beginning to add 1 to pc.
```

Shift Right (by n). Note that the n we load is not the same as n in COND, so I will call the n in COND capital n (N) if I use it. This shift to the right seems VERY similar to the left shift...due to having a shifter that can already handle shifting by 1 bit. NOTE that I am assuming that the implementation of the shift by 1 that is included in the MAC-1 micro-program correctly handles remainders and rounding (for example, shifting 0001 by 1 bit to the right).

```
120: a := ir && smask; //Load n into a. Basically, ignore the first 12 bits
                       //(which is the instruction) and only load the last 4 bits
                       //(n, or how much to shift by).

121: ac := ac >> 1; //Shift by 1. Now we need to subtract n by 1…
122: a := a + (-1);    //Subtracts n by 1.

123: alu := a; if Z then goto 0;  // Checks if n = 0. If so,
                                  //we are done. Increase pc.

124: goto 121;                    //loop back.
```