Computer Science 304
Computer Organization
Fall 2018
Assignment 4 – Accumulator II

Due: 11:59 p.m., Friday, 10/26/2018

For this project, write a program in C that extends the program you wrote for Assignment 2. Once again, the program will display values in various base representations, including binary, based on the current value of the program's accumulator. It will also perform various bit-wise and arithmetic operations involving the accumulator. The program's execution is driven by the following menu:

```
************************************
* Accumulator:                     *
*   Binary  :  0000 0000 0000 0000 *
*   Hex     :  0000                *
*   Octal   :  000000              *
*   Decimal :  0                   *
************************************

Please select one of the following options:

B  Binary Mode             &  AND with Accumulator          +  Add to Accumulator
O  Octal Mode              |  OR  with Acuumulator          -  Subtract from Accumulator
H  Hexidecimal Mode        ^  XOR with Accumulator          N  Negate Accumulator
D  Decimal Mode            ~  Complement Accumulator

C  Clear Accumulator       <  Shift Accumulator Left
S  Set Accumulator         >  Shift Accumulator Right

Q  Quit

Option:
```

Note that the value of the accumulator in various base representations is always displayed when the menu is displayed, as previously. Options for the menu are also entered/accepted as before.

Since C does not have a format for reading or printing values in binary format, the program must convert the short accumulator value to or from a character string containing the binary digits. For input, the user may enter a value in binary (with no spaces) of any length up to 16 bits. In the program, this input should occur along with other base types in `get_operand()`. Note that `get_binary_op()` will need to be called from this function so that the binary string can be converted to a short value and returned from the function. For output, the `acc` must be converted to a binary string using `convert_to_binary()`. Spaces must be inserted after every four binary digits for readability. For consistency, declare all binary strings as character strings of size 20 (to hold the 16 bits, the string terminator character, and spaces, if necessary).

For the new menu entries, &, |, ^, +, and –, the program will request an operand (in the current mode) from the user to perform the corresponding operation. For the + and – operators, a message should be printed if positive overflow or negative underflow is detected (though the operation will still be performed). When either shift operation (< or >) is selected, the user will be

asked to enter the number of bits to shift (in decimal). Note that no additional operand is needed for the ~ and N options. All operation results should be reflected in the accumulator.

Your code must be modular and use functions, as before. Include functions from the previous assignment, plus the following (but no others):

```
unsigned short get_binary_op (char *bin)       // convert bin str to short; return value
void convert_to_binary (short acc, char *bin)  // convert acc to binary str for output
void add (short *acc, char mode)               // call get_operand to get val in mode to
                                               //  add to accum; detect pos/neg overflow
void subtract (short *acc, char mode)          // similar to add, but subtract
```

All code should be submitted in a single source file named `calc.c`.

Your program can be compiled with the command:

```
gcc calc.c –o calc
```

..and run with the command:

```
./calc < test.txt
```

Don't forget to test your program on the CS department computers.