CS270 Programming Assignment 4
“Shift Right Program with subroutines”

Program due Tuesday, April 2 via RamCT by 1 PM.

Goals

Write an LC3 program that will do shift a number in a register right by a specific number of bits. The resulting number will be printed out in binary. For example two number read are 34 (0000 000 0010 0010) and 2 then the first number is shifted by one bit to the right twice. The result printed by LC3 should be 0000 0000 0000 1000.

To allow the TAs to evaluate your program quickly, your program with run normally if location TAMODE is 0. If it is 1, it will run in the TAmode. Code submitted should have TAMODE 1.

- Normal mode: The program prompts and inputs a number using IN and saved it at SRCount. It takes the number in location NUM and shifts it right by the number of bits given by SRCount. The program obtains an ASCII string representing the number in binary saved in locations starting at BINBUFF which is printed out using PUTS.

- TAmode: The program takes the numbers saved at NUM and SRCount. The program obtains an ASCII string representing the number in binary saved in locations starting at BINBUFF. The input and output operations are skipped.

Algorithm: Note that a shift left is easy, just add a register to itself. Shift right is harder. You can choose your own approach. One possible approach to achieve a one-bit shift right is to do 31 one-bit shifts to left circularly (i.e. outgoing bit is brought in at the other end).

Program Structure Specification

Your program should use these subroutines with these labels.

SHIFTR: Thus subroutine takes the number in R1 and shifts it one bit to the right and leaves the result in R1. It should not disturb any other registers i.e. save any registers used at the beginning and restore them before returning.

BINTOSTR: It takes a binary number in R1 and for each bit places the corresponding ASCII character in a buffer (a set of locations) with the most significant bit starting at location. It should not disturb any other registers.

ASCIIbin: converts a single ASCII character in R1 into the corresponding binary number (using location ASCIIoff(and leaves the result in R1. It should not disturb any other registers.

The program must start with this code.

```
;File:      PA4.asm
;Description: --> fill this in (single line only)
;Author:   --> fill this in
;Date:        --> fill this in
; R2: pointer that holds an address        R3: counter
; R4: ->fill if used                       R5: -> fill if used
; ********************** BEGIN RESERVED SECTION **********************
; You may initialize values here (i.e., change the value of a .FILL
; statement), but you may *not* add or remove instructions or perform
; any edit which changes the address of a label in this section
            .ORIG x3000      ; DO NOT change any code before label ZAXB
            BR BEGIN_CODE    ; jump to beginning of code
TAMODE      .FILL 1
NUM         .FILL 55
SRCOUNT     .FILL 5         ;limited to 0 to 9 right shifts
ASCIIoff    .FILL x0030
BINBUFF     .BLKW 16;
NULL        .FILL 0         ;end of specified locations
; ********************** END RESERVED SECTION **********************
;---------you may add more variables or code below this line --------
BEGIN_CODE                  ; your code begins here
```

At the end of the code include result of a test run as seen on the console in form of a comment.

Integration and testing:

Write the three subroutines first as independent programs. Test them thoroughly. Then write the main program with the two subroutines. Test them in the normal mode using a combination of values. Finally test the program in the TAmode before submitting.

Submission Instructions

- Submit the pa4.asm file will be submitted directly to RamCT.

Reminders:

1. Adds comments appropriately.
2. Illegal inputs should generate message "Illegal Input".

Grading Criteria

To grade the assignment, we will examine and run program (30 points), and we will verify that your computation gets the right answers on the example data and our own test data (35 points). In addition points will be given for coding style and comments (15 points), following assignment directions (20 points). The grading factors we consider for coding style include having clear and concise comments, consistent indentation, and the minimal amount of code to solve the problem.

Late Policy

There is a 25% penalty for late submission. They cannot each be more than 24 hours late. Late hardcopy documentation must be submitted electronically by email to cs270@cs.colostate.edu.