CS 270 PA 4: More Assembly Programming Loops and Conditionals

due Wed Apr 4, 5:00 pm

The assignment is to be done individually. ***This is the most citical assignment in the class, since it is required in order for you to do the next one.*** This is an instance where double jeopardy applies. The code that you will write for this assignment will be needed for PA5.

The main goal of this programming assignment is to advance the skills of designing complex assembly language programs. You will see that the conditional branch is all that you ned to implement almost all algorithmic logic. A second goal is to drive home the importance of simple, incremental program development and complete testing before moving on. We are asking you to submit multiple programs, but each one builds on the previous one.

Problem 1 Consider the following little snippet of C.

```
//  Declare and initialize input, output, i, and k as uint16_t
//  It is assured that k is non-negative and "small enough"
1:  output = 0;
2:  for (i=k; i<16; i++){
3:      if (input & 1<<i)
4:          output = output | 1 <<(i-k);
5:  }
```

Submit the answer to the following questions in your README file.
1. What is the value of `output` if `input` is initialized to `0x3A44` and `k` is 4?
2. What is "small enough" (i.e., for what range of vlaues of `k` does the program produces a meaningful answer).
3. Describe in one simple, short sentence what this code does.

PA4.1.asm You will incrementally write an LC-3 program that implements this little snippet. The first part is to simply implement the loop control structure. You LC3 program has a single variable `k`, and null terminated string `Hello\n` (note the newline character). Write, debug and test an LC3 program that implements the following single line of C.

```
for (i=k; i<16; i++){ printf("Hello\n"); }
```

```
.ORIG 3200
BR START
MESSAGE .STRINGZ "Hello\n"
K       .BLKW 1
;Other Data variables (input, output, mask1, mask2) may eventually come here
; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;
;  Fill code in this box (and uncomment it :-)
;  initialize the loop counter
;  test if it is at the limit
;  Conditionally branch to the code after/outside the loop
; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;

BODY     LEA R0, MESSAGE ; Initialize the address of the Hello string
         PUTS

; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;
;
;   You may also need to fill code in this box
;
; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;

; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;
;   Here's the code AFTER the loop:
         HALT
; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;
```

Problem PA4.2.asm Now we replace the loop body above with a simple `if` statement in C

```
if (input & 1<<i){ COND_BODY }
```

The expression uses C's bitwise-and, and left-shift operators. We want you to write LC3 assembly code that computes this expression, integrate it into the conditional expression, insert that back as at the loop body of the program `PA4.1.asm` and submit this modified file as `PA4.2.asm`. Hint: since this expression is inside a loop body, you will find it very convenient, in terms of both the simplicity and efficiency of the code to modify the original C snippet as follows:

```
mask1 = /* some expression E1 to initialize the mask */ ;
mask2 = /* some expression E2 to initialize the mask */ ;
for (i=k; i<16; i++){
    if (input & mask1) {
        printf("Cond is TRUE\n");
    }
    mask1 = /* expression E3 to update the mask */
    mask2 = /* expression E4 to update the mask */
}
```

In your README file you should answer the following questions:
4. Complete the snippet of C so that the expressions E1 ... E4 are as simple as possible. Based on this answer the following specific questions:
5. What is the expression E1, used to initialize mask1?
6. What is the expression E3, used to update mask1?

Problem PA4.3.asm Now further modify the previous program to implement the complete, modified C program. For this, you will use the second mask, since the condition body uses the expression `1«(i-k)` which you want to replace by `mask2` so that the assignment statement in the C code becomes

```
output = output | mask2
```

In your README file you should answer the following questions:
7. What is the expression E2 used to initialize mask2?
8. What is the expression E4, used to update mask2?

Submission Instructions Create a directory called PA4 to store the assembly files. Submit a file, PA4.tar.gz, generated from the following command (assuming you are

running this command from inside PA4 directory):

```
cd ..; tar ¡Vczvf PA4.tar.gz PA4
```

Do not submit the assignment with lc3tools inside PA4. There must be exactly four files in your PA4, namely, PA4.1.asm, PA4.2.asm, PA4.3.asm and README. The name of the README file should be exactly upppercase letters "README" without any file extension. PLEASE DO NOT SUBMIT YOUR .obj or .sym files.

Coding style includes the proper commenting of code.

Grading Criteria

README: 20 pts (3+3+3+7+4)
PA4.1.asm: 25
PA4.2.asm: 30
PA4.3.asm: 15
Coding Style, Clarity, Following Instructions: 10 pts