Name: _____ **Date:** _____

CS270 Homework Assignment 3 (HW3)

Due Thursday, April 17, 11:59pm, under door COMSC 256, no late submissions.
Homework and programming assignments are to be done individually.

Goals

To understand the stack convention of the LC3 compiler:

- How the stack pointer and frame pointer are managed
- Where parameters, return values, return addresses, and local variables are stored
- What the stack looks like before, during, and after a function call.

Instructions

This assignment makes you interpret the assembly output of the LC3 compiler (*lcc*) in order to figure out the stack convention. We supply the original C file and corresponding assembly code generated from the compiler:

http://www.cs.colostate.edu/~cs270/.Spring14/assignments/HW3/stack.c
http://www.cs.colostate.edu/~cs270/.Spring14/assignments/HW3/stack.asm

The C program and a fragment of the assembly file with the function code are shown below, these should be sufficient. Note that we have modified lines zero and one to do separate decrements of the stack pointer, instead of combining them as the compiler does. Examine the C code and the assembly code and answer the questions, which are worth 5 points each. For the last question you must draw a picture of the stack at a certain point in the program, this problem is worth 30 points. This assignment does not require you to compile or run code using the LC3 compiler. Reading pages 389-392 of the textbook will help explain this assignment.

The Assignment

Here is the C code for the assignment, comments have been removed to save space:

```
int add(int param0, int param1)
{
   int result;
   result = param0 + param1;
   return (result);
}

int main(int argc, char *argv[])
{
   int local0 = 1234;
   int local1 = 2345;
   printf("Result: %d\n", add(local0, local1));
   return (0);
}
```

Here is the assembly code generated by *lcc* for the add function in *stack.c*

;;;;;;;;;;;;;;;;;;;;;;;;;;;;add;;;;;;;;;;;;;;;;;;;;;;;;;;;;
lc3_add

;; stack entry

0: ADD R6, R6, #-1
1: ADD R6, R6, #-1
2: STR R7, R6, #0
3: ADD R6, R6, #-1
4: STR R5, R6, #0
5: ADD R5, R6, #-1

;; function body

6: ADD R6, R6, #-1
7: LDR R7, R5, #4
8: LDR R3, R5, #5
9: ADD R7, R7, R3
10: STR R7, R5, #0
11: LDR R7, R5, #0

;; stack exit

12: STR R7, R5, #3
13: ADD R6, R5, #1
14: LDR R5, R6, #0
15: ADD R6, R6, #1
16: LDR R7, R6, #0
17: ADD R6, R6, #1
18: RET

Answer the following questions, using the variable names from the original program, or one of the following: *stack pointer*, *frame pointer*, or *return address*. Do not tell me that R7 is getting pushed or R5 getting popped, we already know that from reading the code. When the answer is the *frame pointer*, identify whether it is the frame pointer from *main()* or *add()*.Be specific with names from the original C program: *local0*, *local1*, *param0*, *param1*, etc. Assume that the main program has pushed *param0* and *param1* before calling the *add()* function.

Question 1: The code at line 0 is making room on the stack for which value?

Question 2: What is getting pushed at lines 1 and 2?

Question 3: What is getting pushed at lines 3 and 4?

Question 4: What value is being setup at line 5 for which function?

_____

Question 5: The code at line 6 is making room on the stack for which value?

_____

Question 6: Which parameter is loaded, and from what frame pointer offset at line 7?

_____

Question 7: Which parameter is loaded, and from what frame pointer offset at line 8?

_____

Question 8: What is the code at line 9 doing?

_____

Question 9: What is being stored at line 10, and to which frame pointer offset is written?

_____

Question 10: What is being load at line 11, and from which frame pointer offset is read?

_____

Question 11: Is the instruction at line 11 redundant?

_____

Question 12: What is being stored at line 12, and to which frame pointer offset is written?

_____

Question 13: What is getting popped at line 14 and 16?

_____

Question 14: What are registers R5 and R6 used for by the compiler?

R5: _____

R6: _____

Question 15: Draw a picture of what the stack looks like after executing line 12 of the program. Include all the values from the stack frame for add() and the parameters pushed on the stack by main(). In the left column specify the name of the variable or use the names Frame Pointer, Return Value, or Return Address. In the second column, if the value is a local variable or parameter, specify which one. Also in the second column, if the value is a Frame Pointer, Return Value, or Return Address, tell whether it belongs to main() or add(). Draw an arrow to the address pointed at by the current Frame Pointer and Stack Pointer, again after executing line 12 of the program. In the third column, show the offset from the Frame Pointer in array notation, for example Frame Pointer[0], Frame Pointer[1], etc. You should need exactly 6 rows for a correct answer.
