CPSC 213

Lab 5

Translating Procedures with Parameters and Local Variables

Done in groups of two in labs: July 31 and Aug 2

Due Sunday, Aug 4, 2013, 10:00pm

Goal

We'll continue with the translation of Java and C statements into sm213 machine instructions. The goal of this assignment is to make you understand:

- how procedures with parameters and local variables are implemented in assembly language
- how to use more complicated structures containing pointers to other structures

Provided Code

Download lab5_examples.zip, unzip in the directory you'll create for this lab. You will find the following files:

- `lab5_ex1.c, lab5_ex1.java, lab5_ex1.s` : the C, java and sm2113 code for example 1.
- `lab5_ex2.c, lab5_ex2.java, lab5_ex2.s` : the C, java and sm2113 code for example 2.
- `lab5_ex3.c, lab5_ex3.java, lab5_ex3.s` : the C, java and sm2113 code for example 3.

Download lab5_code.zip, and unzip it in the directory you've created for this lab. It contains the following files:

- **lab5_1.c** : a snippet of C code with a couple of functions without any parameters and a main program (this is a warm-up exercise).
- **lab5_2.c** : a snippet of C code with a linked structure for the notes of a bst tree and a procedure that searches a bst tree to find the node that has a given value.
- **lab5_3.s** : a snippet of sm213 assembly code for a C function and its caller program

Your Task

Study the examples provided for this lab. They contain examples of the concepts you are asked to translate to and from sm213 assembly language in this assignment. Using the simulator, step through and convince yourself of the correspondence between the high-level expression and the

instructions the machine executes. You are not expected to completely understand the algorithms, but it may help if you get an idea of what each function is doing. You are encouraged to experiment with modifying their inputs and running in the simulator to observe their result.

Your task for this lab assignment is to

- translate the C code in the C files **lab5_1.c** and **lab5_2.c** into sm213 assembly code, producing two new files **lab5_1.s** and **lab5_2.s**
- translate the sm213 assembly code in the file **lab5_3.s** into C code, producing a new file **lab5_3.c** .

The file **lab5_2.c** defines the node of a binary search tree (bst) and the search function for such a tree. This is the same structure and function that is defined in the example **lab5_ex3.c** file with the following differences. In the example the tree is stored in an array of nodes. Each node has the item and the subscripts of the left and right child of the node. The program in **lab5_2.c** defines a linked structure for the tree. Each node of the tree has the item and two pointers pointing to the left and right child of the node. Each node is created dynamically by a call to `maloc(sizeof(bst_node))` when a new item is added to the tree and the whole tree is represented by a pointer to its root. The code in **lab5_ex3.c** will help you in that part. You can start with this code and modify it to reflect the new structure. To test your program you need to define the whole tree, node-by-node on the data part of your **lab5_2.s** file. We have started building the tree in the last comment in **lab5_2.c**. Continue from there and complete the tree.

When writing your solutions for this assignment, keep in mind the following points:

- Follow the SM213 stack-based calling convention for the procedures you write.
- Do not assume any initial value for the registers, except as specified by the calling convention.
- You may optimize within individual procedures in the snippet. Treat each function/procedure independently and do not optimize across them (conform to the calling convention).
- Local variables may be kept in a register, unless there are insufficient registers or their value needs to persist across e.g. a recursive call, in which case they should be kept on the stack. Local *arrays* must be kept on the stack.
- Add comments to denote the correspondence between expressions and/or parts thereof in the HLL snippet, and the machine instructions you generated.
- Use the simulator for syntax checking and testing. The code you produce **must** load without syntax errors in order to receive credit for it.
- Provide a "stub", as shown in the examples, to set up the stack and call the initial function (passing parameters, if any). End the stub with a halt instruction.
- It is recommended to start the top of the stack at the 32K position (initial stack pointer of `0x8000`, to leave plenty of available stack space.
- These are *snippets*, not complete programs; thus they will not compile and run (despite looking very much like they will). In particular, notice that the stub code cannot be expressed in C or Java.

- Assume any pointer variables have been already initialized to hold the address of a structure or an array of unspecified but adequate size. For testing purposes you may do this initialization statically in the respective data section of the assembly code as illustrated in the examples. Several of the snippets also use a separate variable to hold the size of the dynamic array. (You will learn how this happens at runtime later.)
- There are both C and Java versions of the HLL snippets. Note that the C version is closer to the sm213 assembly as Java contains additional features (like implicit array bounds checking) that do not appear in the sm213. Treat the Java version as a loose conceptual equivalent, not a direct translation.

What to Hand In

Please hand in exactly and only the following files with the specified names.

1. The assignment cover page filled in for this assignment with information for each person in the group.
2. `lab5_1.s` containing your solution for `lab5_1.c`
3. `lab5_2.s` containing your solution for `lab5_2.c`
4. `lab5_3.c` containing your solution for `lab5_3.s`

Hand In Instructions

Only one person in each group will submit the assignment using the department's handin tool. To submit it you should do the following:

- Create a directory cs213 in your Unix home directory, if you have not done so. In your cs213 directory, create a directory **lab**.
- Place all the files you have to submit in the lab5 directory.
- At the Unix command line execute
  > handin cs213 lab5
- Type **handin help** to see the options that are available for the handin tool.

Last modified: 7/28/2013 3:24 PM