CPSC 213

Lab 3

Translating Dynamic Variables and Structs

Done in groups of two in labs: July 19 & 22

Due Tues July 23, 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 dynamic objects in Java and C are implemented
- how to translate Java to C

Provided Code

Download lab3_files.zip, and unzip it in the directory you'll create for this lab. It contains the following files:

- **lab3_1.c** : a snippet of C code with an array of struct's.
- **lab3_2.c** : a snippet of C code with a dynamic array of struct's which also contain dynamic data of their own.
- **Course.java** and **Transcript.java**: two java classes representing a simple student transcript containing a list of courses and marks.

Your Task

1. Translate the assignment statements in the two C files into sm213 assembly code producing three new files **lab3_1.s** and **lab3_2.s**. Translate only the statements that are in between the lines "TRANSLATE THE FOLLOWING n STATEMENTS" and "END OF TRANSLATION". The rest of the statements are there to declare the variables and allocate the space. The C programs are incomplete and cannot be compiled and executed.

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

   - Do not assume any initial value for the registers.
   - Do not optimize across statements within the snippet; treat each C statement independently of the other statements in the snippet. That is, do not optimize with the assumption registers will retain their initial or last-written value between statements. However, you *are* allowed to make use of the constants and invariants within a single statement to optimize your code.

- Add comments to denote the correspondence between expressions and/or parts thereof in the C snippet, and the machine instructions you generated.
- Use the reference simulator for syntax checking and testing. The code you produce **must** load without syntax errors in order to receive credit for it.
- End your code with a `halt` instruction to prevent the simulator from "running off the end".
- Assume any pointer variables have been already initialized to hold the address of a structure or array that is shown in the malloc call. For testing purposes you may do this initialization statically in the respective `.long`, as illustrated in the examples in the class.

2. The two java files, Course.java and Transcript.java, contain the definitions for a student transcript and a simple main function for testing it. Each course has two components, a course name and a mark. Each transcript has its student name and an array of courses the student has taken. Your task here is to translate these class definitions and the main program to C following the guidelines we discussed in the class. Use the name **transcript.c** for your C file with the transcript program and **include both classes and the main program in the same file**. Note that C allows you to have multiple struct definitions and their related structure in the same file.

In your C translation use **`char*`** to represent a Java string. Note that you can assign directly constant strings to a char* variable. For instance the following code will print "This summer we'll visit Europe and Asia".

```
char* s;
s = "Europe and Asia"
Printf("This summer we'll visit %s.", s);
```

To compile your **`transcript.c`** program and create an executable named **trans**, you should use:

`gcc -o trans transcript.c`

This will create an executable file **`trans`** in the same directory with **`transcript.c`** , which you can run by typing

`./trans`

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. `lab3_1.s` containing your solution for `lab3_1.c`
3. `lab3_2.s` containing your solution for `lab3_2.c`
4. `transcript.c` containing your C translation of `Transcript.java`

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 **lab3**.
- Place all the files you have to submit in the lab3 directory.
- At the Unix command line execute
  > handin cs213 lab3
- Type **handin help** to see the options that are available for the handin tool.

Last modified: 7/17/2013 4:13 PM