Project 4 FAQs

Project 4 FAQs

Q) I don't know where to get started.

A) Suggested list of steps:

1. Read the project worksheet carefully to understand the specifications.
2. Read the MIPS code generation part of the Runtime lecture slides to get an idea of code you need to generate for different types of cases.
3. Try running src1.s distributed as part of the test input on top of spim.linux to get a feel of how the emulator works.
4. Try reading src1.s and other generated assembly code and try to understand what the assembly code is doing. Refer to the SPIM manual as needed if you don't understand what a particular instruction does.
5. Add offset attributes to your symbol table entries to calculate the position of each variable in memory. Refer to the slides to understand how to calculate offsets. Try printing them out to make sure they are correct.
6. Try writing a very simple program that prints the value of a global variable. Then try a local variable. Then try simple arithmetic. Then try function calls. Then try control flow.

Q) The reference output from the reference implementation sometimes does stuff that I don't understand.

A) Again, the reference output is just an example output and you are not obliged to follow the code generation strategy. It is correct code but it is far from optimal. One of the things the reference implementation did in src1.s was to store the address $fp - 4 on to the stack. The address $fp - 4 is the address of the local variable 'x' in src1. The code stored the address $fp - 4 on to the stack so that it can load the address from the stack whenever 'x' is used in the code, instead of recomputing $fp - 4 repeatedly. But this is not optimal actually. Recomputing $fp - 4 happens to be cheaper than loading it from the stack in most machine architectures. So in your

code, you can choose to recalculate $fp - 4 on every access to 'x', which is also more optimal anyways.

Q) The spim.linux binary does not run with a 'Permission denied' message.

A) You have to give execute permissions before running any linux binary. Try doing 'chmod +x ./spim.linux' before running.

Q) How are the values we may need (like the first $ra, $sp, and $fp) initialized?

A) $sp is initialized by SPIM to point to the top of the stack (See section 3. Memory Usage in the SPIM manual). $ra and $fp remain 0 until you initialize them yourself. An example output from SPIM:

```
-bash-4.1$ ./spim.linux
SPIM Version 6.0 of July 21, 1997
Copyright 1990-1997 by James R. Larus (larus@cs.wisc.edu).
All Rights Reserved.
See the file README for a full copyright notice.
Loaded: ./trap.handler
(spim) print $sp
Reg 29 = 0x7fffeffc (2147479548)
(spim) print $fp
Reg 30 = 0x00000000 (0)
(spim) print $ra
Reg 31 = 0x00000000 (0)
```

$ra is the return address register and will be eventually set when the trap handler first calls the main function using the jal instruction (see the .text section in trap.handler). Inside your main function, you can use the $ra with the jr instruction to jump to the return address. Now, if you call another function inside main using jal, $ra will be overwritten so it is your responsibility to save it on the stack as we learned in class. As for $fp, you should set it immediately upon entering main (and any other function for that matter), to start a new frame for that function.

Comments

You do not have permission to add comments.

Sign in | Recent Site Activity | Report Abuse | Print Page | Powered By **Google Sites**