Homework 3

Due Date: Thursday, February 9, 2006, 2:00

Your Name: ________________________________
Your Email: ________________________________

Question 1 One option for a compiler is to go straight from the AST to the target code (e.g., assembly code). A more common approach is to use 2 steps, first going from AST to an intermediate code and then going from intermediate code to target code. Give 3 reasons why this second approach is a good idea.

__________________________________________
__________________________________________
__________________________________________

Question 2 In class, we used a syntax-directed approach to translations. What 2 attributes are used for the non-terminal E?

___________
___________

Question 3 How many operations can each 3-address instruction perform? ________
How many operands may appear in any single 3-address instruction (maximum, including the result)? ________

Question 4 Assume we are compiling a source language that has 3 sizes of floating point numbers (single, double, and quad) and 3 sizes of integers (byte, halfword, and word). How many “add” instructions would our intermediate language likely have?

__________

Question 5 In the SPARC architecture, a “ba” instruction takes 4 bytes; thus a “goto” instruction in the IR language will ultimately consume 4 bytes of memory. How many bytes in the executable will our “label” instruction consume? __________
**Question 6**  Is our PCAT compiler using Quadruples, Triples, or Indirect Triples to represent the IR instruction sequence? ________________

**Question 7**  In the syntax-directed translations discussed in class, what does the synthesized attribute “E.code” contain?

__________________________________________

__________________________________________

**Question 8**  In the syntax-directed translations discussed in class, what does the synthesized attribute “E.place” contain?

__________________________________________

__________________________________________

**Question 9**  Consider this grammar rule:

\[ E_0 \rightarrow E_1 + E_2 \]

When computing \( E_0 \).code and \( E_0 \).place, can we assume \( E_1 \).code and \( E_1 \).place are already computed and available? __________

**Question 10**  Consider this grammar rule:

\[ S_0 \rightarrow \text{if } E \text{ then } S_1 \text{ end} ; \]

Here is how we will translate this (ignoring trueLabel, falseLabel and short circuit behavior):

\[
< \text{code for } E > \\
\text{if } E.\text{place} = 0 \text{ then goto Label_A} \\
< \text{code for } S_1 > \\
\text{Label_A:}
\]

Show how we would translate this rule:

\[ S_0 \rightarrow \text{if } E \text{ then } S_1 \text{ else } S_2 \text{ end} ; \]
**Question 11** Here is a grammar rule for a “do-until” statement:

\[ S_0 \rightarrow \text{do } S_1 \text{ until } E \text{ end ;} \]

The idea is that we will always execute the body \( S_1 \) at least once. We will test after each execution and we will terminate once the condition becomes true. Show how we would translate this rule.

**Question 12** Is “static” associated with compile-time or run-time?

_____________

Is “dynamic” associated with compile-time or run-time? ______________

**Question 13** Consider a program that contains one routine named “foo”. Perhaps “foo” is recursive. At one instant at run-time, foo will be “alive” or “running” either zero, once, or many times. We call each of these invocations of foo a ...what?

_____________

**Question 18** What happens to the activation stack at runtime when a routine is invoked (i.e., push or pop)? __________

What happens when a routine returns? __________

Where are the local variables for a routine found? ______________

**Question 19** Define “environment” and “state.”

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________
**Question 20** In UNIX, is data in the “.text” segment read/write or read-only?
____________________________

What about the “.data” segment? __________________

**Question 21** In C and C++, when objects / structs are allocated, they are placed in the heap. Later, the program may free storage that was previously allocated. Will objects / structs ever be compacted by the garbage collector in C / C++? ______

**Question 22** Name 3 languages that use automatic garbage collection.
________________________________________________________________________

**Question 23** In an automatic garbage collector, objects may be “compacted.” Describe object compaction?
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________

**Question 24** In the activation record stack, each frame contains 2 pointers: the static link and the dynamic link. Which link is used to point to the routine’s caller?
____________________________

**Question 25** Routines may contain local variables. When we compile a routine, we will have to “lay out” the activation record. What must we assign to each local variable?
____________________________

**Question 26** In our implementation of PCAT on the SPARC, how many bytes will be allocated to each variable? __________

Will all variables occupy the same amount of space? __________

**Question 27** In other compilers, where different types of data may have different sizes and where alignment restrictions must be followed, what will we have to insert between the variables? ______________________________