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:

$S_0 \rightarrow$ **if** E **then** S_1 **end ;**

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

```
        < code for E >
        if E.place = 0 then goto Label_A
        < code for S1 >
    Label_A:
```

Show how we would translate this rule:

$S_0 \rightarrow$ **if** E **then** S_1 **else** S_2 **end ;**

Question 11 Here is a grammar rule for a "do-until" statement:

$S_0 \rightarrow$ **do** S_1 **until** E **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 by 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? _____