EE573 Final Exam, Fall 97

points in parentheses indicate an approximate relative effort for answering the questions.

1. Describe 3 parallelizing transformations. Show the original and the transformed code. (15)

2. Explain how the GCD test tests the dependence between statement s1 and s2 in the following code: (10)

```
    DO i=1,n
s1:   a(2*i+3) = b(i)
s2:   c(i+2)   = a(3*i+4)
    ENDDO
```

3. Explain how the strength reduction technique relates to induction variable substitution. Which technique is applicable in what situations? (10)

4. Consider the following program fragment:

```
s1:  a = b + c*d
     ...
sn:  e = c*d
```

a) the term c*d in s1 can be reused in statement sn. What is the name of the compiler technique that performs this optimization? (5)

b) what test(s) need to be done by the compiler to be sure that the temporary holding the result of c*d from s1 can be reused in sn? (5)

5. Give 3 examples of peephole optimizations. (15)

6. Draw the stack after the following subroutine has been entered. Name the various parts on the stack. (10)

```
double f(Rec r, int i)    // Rec is a struct of size 100 bytes
{  double d;
   int m;
   ...
}
```

7. a) Write the dataflow equations for the "constant propagation" problem. The goal of the analysis is to identify variables that hold a known, constant value. Describe all terms of your equations and how you compute them. (20)

b) Indicate if this is a forward/backward flow and an all-path/any-path problem. (5)

8. Name 3 reasons why one does subroutine inline expansion. (10)

9. Consider the production

<stmt> ::= IF <condition> #start-if
THEN <stmt-list> ENDIF #final-if

explain the semantic actions performed in #start-if and in #final-if (10)

10. How would you classify a parser that operates like this: (10)

1: select the goal production
2: process the RHS of the production from left to right:
if the symbol is a terminal: match the next input token
else
if the nonterminal has one production, select it and goto 2
else look at the next input token, select the production that
starts with this symbol, goto 2

11. Write a CFG for an expression containing * and +, so that the usual precedence rules are reversed (+ before *). The expression can also contain IDs and INTLITERALs. (15)

12. Add action symbols to your CFG of question 11. Describe the actions performed in each corresponding semantic action routine. (15)

13. Give an example of a non-LL(1) grammar that can easily be turned into LL(1). Show the LL(1) form. (10)

14. Determine the register need for the expression g*h + f*(d+e), using the recursive tree algorithm. How many registers would a naive algorithm need? Assume that at most one memory location can be accessed per instruction. (15)