NAME: _____

EE573 Final Exam, Fall 2000

1. (10pt) Explain in at most three sentences why a compiler would use more than one internal program representation.

2. (10pt) Give an example of an LR1 grammar. Explain why it is LR1.

3. Register Allocation:

a) (40pt) For the given tuple code, do both top-down register allocation and bottom-up register allocation. The machine has three registers. Use the given instructions. Fill the generated assembly code in the provided space and show variable-to-register assignments at the end of each tuple.

Available instruction:
Move reg1 → reg2
Move var → reg
Move reg → var
Mul reg1, reg2 → reg3
Add reg1, reg2 → reg3

Tuple code	Top-down		Bottom-up	
	Assembly code	Var-reg assignment R0 R1 R2	Assembly code	Var-reg assignment R0 R1 R2
ADD A B → C				
MOVE D → A				
MULT B C → D				
MOVE B → A				

b) (10pt) Assuming that only memory instructions count, which code performs better? Do you expect this to be true for all programs? In what situations do you expect which algorithm to perform better?

4. Consider the following two loops

Loop 1:

```
DO i =1,n
      a(5-i*<x2>) = b(i)
ENDDO
```

Loop 2:

```
ind = <x1>
DO i=1,n
      ind=ind-1
      a(ind)=b(i)
      ind = ind-2
ENDDO
```

a) (5pt) What expressions must <x1> and <x2> represent so that Loop 1 and Loop 2 perform the same operation?

<x1> :

<x2> :

b) (5pt) What are the names of the compiler techniques that can transform Loop 1 into Loop 2 and vice versa?

c) (10pt) Explain why the codes are not fully equivalent. Be specific in your answer.

5. Consider the code

s_1: a=b+c*d
...
s_n: call sub(<parameters>)
...
s_m: e=c*d

a) (10pt) While compiling this code, a CSE algorithm tries to reuse in statement s_m a temporary holding the value c*d, which was generated in statement s_1. In order to perform correctly and non-conservatively, what is the necessary and sufficient information that the CSE algorithm needs to know about the use of <parameters> in subroutine sub?

b) (10pt) For language *L* a compiler can perform CSE in the above code without the need for interprocedural analysis. What properties must *L* have for this to be true?

6. Formulate the following problem as a dataflow problem: given a set of boolean variables, determine whether or not the value of each variable is known to be *true* at any given point in the program.

a) (30pt) Define the functions Gen and Kill and write the dataflow equations.

Gen:

Kill:

Dataflow equations:

b) (10pt) The dataflow algorithm will determine the properties of boolean variables at the beginning and end of each BB. How can you determine the properties at *any* point in the program?