"Advanced Compiler Techniques"

Midterm Examination

Fall 2008

You have 1 hour 40 minutes to work on this exam. The examination has 100 points, so you should roughly spend about one minute for each point. Please budget your time accordingly.

You are allowed to bring one single-sided cheat sheet on an A4 paper.

Write your answers in the space provided on the exam.

NAME: ________________

Student ID: ________________

Problems	**Points**	**Score**
1	15	____________
2	20	____________
3	20	____________
4	15	____________
5	30	____________
Total	**100**	____________

1. (15 points) True or False. No Explanation is necessary.

   a) "Dominator" relation is a partial order.

   b) "Immediate Dominator" relation is a partial order.

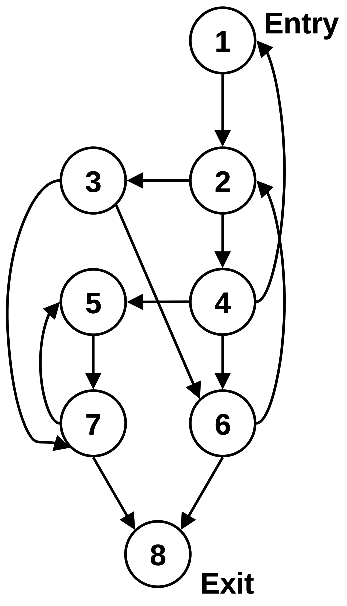
   c) If depth-first order is used, the data-flow algorithm for constant propagation converges in at most $d+2$ iterations, where d is the depth of the input flow graph.

   d) Applying constant propagation before lazy code motion can improve the applicability of lazy code motion.

   e) Applying lazy code motion before constant propagation can improve the applicability of constant propagation.

2. (20 points) Natural Loops

a) (5pt) Construct the dominator tree for the following flow graph.



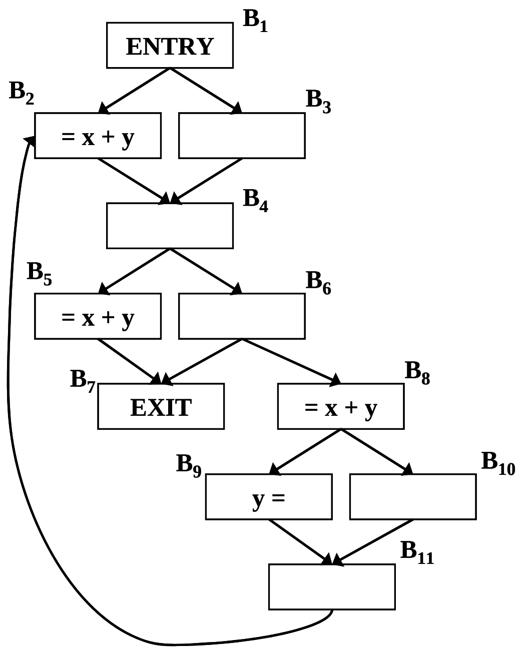
b) (5pt) What are the back edge(s) in the above flow graph?

c) (5pt) What are the natural loop(s) in the above flow graph?

d) (5pt) Is the above flow graph reducible? Explain your answer.

3. (20 points) Lazy Code Motion (PRE)

For the following program



a) (5pt) For which blocks B is x+y anticipated at the beginning? At the end? List those sets IN[B] and OUT[B] that contain x+y when we compute anticipated expressions.

b) (5pt) For which blocks B is x+y "available" (in the sense used in the PRE algorithm) at the beginning? At the end?

c) (3pt) For which blocks B is x+y in Earliest of B?

d) (7pt) Show the result of lazy code motion, introduce new blocks if necessary. (you may show the results on the above original graph)

4. (15 points) Convert the program below to SSA form.
   a) (5pt) Construct a CFG (add ENTRY & EXIT nodes)
   b) (6pt) Insert ɸ-functions at necessary places
   c) (4pt) Rename Variables

```
k = false;
i = 1;
j = 2;
while (i<=n) do
{
    j = j*2;
    k = true;
    i = i+1;
}
if (k) then
    print j;
else
    i = i+1;
```

Draw your CFG with code in SSA form in the space below. (Show work for partial credit)

5. (30 points) DFA on Value Range

In many cases knowing the range of variables is beneficial. For instance, knowing that variables *a* and *b* are between 0 and 127 may allow us to represent both variables within one byte instead of two words, thereby providing a more compact representation for certain data structures.

Suppose you are analyzing a program consisting of the following types of statements:

- `a = <const>`
- `a = b`
- `a = b + <const>`
- `a = b + c`

where all variables and constants are integers.

Your task is to formulate a dataflow problem called **VarRange** that would allow one to approximate the range of any given variable at any point in the program.

The range is to be represented by an interval **[*x*, *y*]** where both *x* and *y* are constants. Assume that **MAX** is the biggest representable integer and we are dealing with **positive** numbers (including zero) only.

a) (4 pt) What are the top and bottom elements of the lattice for the dataflow framework formulation of VarRange?

b) (3 pt) What is the JOIN ($\vee$) operator for VarRange?

c) (3 pt) What is the partial order ($\leq$) relation induced by the $\vee$ operator?

d) (8 pt) Assume for simplicity that each basic block consists of at most one statement. Define the transfer function for VarRange.

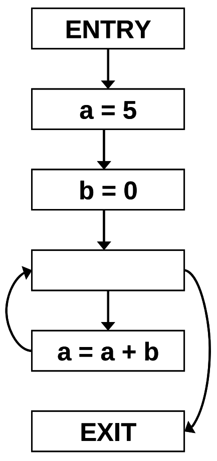
e) (3 pt) Is the transfer function you defined above monotonic?

Yes / No (circle one, no explanation needed)

f) (3 pt) Is the transfer function you defined above distributive?

Yes / No (circle one, no explanation needed)

g) (6 pt) What is the range for variable ***a*** [on EXIT] as computed by your algorithm for the CFG below?


ENTRY
a = 5
b = 0
a = a + b
EXIT