CPEG 421/621 - Homework 3
Issue Date: Tuesday, 20 March, 2012
Due Date: Tuesday, 3 April, 2012

Instructions

Please write the answer to each problem on one sheet of paper. Be as concise and clear as you can. Although we accept hand written answers, we encourage you to hand in print-out solutions or email your solutions to `aron+ta@udel.edu` and `szuckerm@eecis.udel.edu`. If you do hand in hand-written solutions, please make sure the writing is clearly readable. To avoid misplacement of various components of your assignment, make sure that all answer sheets are stapled together. You may discuss the problems with your classmates, but all solutions must be derived independently.

Problem 1

Find all data dependence relations for the following statements:

```
1    b = a+1
2    c = 2*a
3    if (c>0) then
4        b = c + a
5    else
6        b = b + 1
7    endif
8    d = a+b
```

Problem 2

Tree-pattern matching assumes that its input is a tree.

a. How would you extend these ideas to handle DAGs, where a node can have multiple parents?

b. How do control-flow operations fit into this paradigm?

Problem 3

In this problem, you will study the basic techniques of instruction scheduling. Examine the code below (The architecture is a single clean pipeline. Here the add and subtract operations take 1 cycle, the multiply operation takes 2 cycles, and the load takes 3 cycles):

```
    R1 := load X
    R2 := load Y
    R3 := R1 + 5
    R4 := R2 * 7
    R5 := R1 * 6
    R6 := R4 * R5
    R7 := R2 * 9
    R8 := R3 + R6
    R3 := R3 + R8
```

a. Draw the DDG of the code.

b. Formulate the problem of instruction scheduling using this example.

c. Perform list scheduling as presented in the lecture and derive the schedule.

- Show the ranking of all instructions.
- Show the final scheduled code.
- How many cycles will your schedule run?

d. Search literature and find at least one reference to an instruction scheduler which uses a different rank function. Give the reference, and explain how it works.

Problem 4

Consider the instruction sequence shown below. Draw the live ranges of the variables and construct the interference graph. Perform simple register allocation using k=4 registers without coalescing or live-range splitting for the code.

```
    R1 := 2
    R2 := 0
    R3 := c
    R4 := R1 + R3
    R5 := R4 + 1
    if R1 > R4 goto L2
      L1: R2 := R2 + 1
    R6 := R1 + R2
    R5 := R4 + R6
    goto L3
      L2: R3 = R3 + 1
    R3 := R3 * 2
    if R2 < 10 goto L2
      L3: print (R2, R3, R4, R5)
```