

Write your answers on another sheet of paper. Homework assignments are to be completed individually. Hand-written submissions are fine, but they must be readable. Due at the beginning of class. Total points: 100, 5% of course grade

1. [10 points] Garbage Collection. For the following code,

```
class Main {
    public static void main(String[] a){ System.out.println(new Foo().testing());}
}
class Foo {
    Foo next;
    public int nextFoo(Foo p) { next = p; return 7; }
    public int testing() {
        Foo local; int i; i = 0;
        while (i<N) {
            local = new Foo(); local.nextFoo(local);
            i = i + 1;
        }
        return 42;
    }
}
```

- a) indicate the number of objects created by the whole program as a function of N ,
- b) indicate the number of objects as a function of N that will leak (not be destroyed even though it is unreachable) if a reference counting scheme is used,
- c) and indicate the number of times a mark and sweep collector will be called as a function of N assuming 12 bytes of GC overhead (as in the MiniJava GC assignment), 4 bytes to store a pointer, and a 36 byte heap.

2. [5 Points] Garbage Collection. Please write a one paragraph answer to the following question: How does type safety or lack thereof affect GC?
3. [25 Points] Instruction Scheduling. The loop shown below

```
for (i=1000; i>0; i--) {
    x[i] = s + x[i] + y[i];
}
```

can be converted to assembly as follows:

```
// body of the loop
// assume s is in the register F6
Loop:
    L.D      F0, 0(R1)      // F0 = *R1, R1 points at x
    ADD.D   F4, F6, F0     // F4 = F6 + F0
    L.D      F2, 0(R2)     // F2 = *R2, R2 points at y
    ADD.D   F4, F4, F2     // F4 = F4 + F2
    S.D      F4, 0(R1)
    DADDI   R3, R3, #-1    // R3 = R3-1
    DADDI   R1, R1, #-8    // R1 = R1-8
    BNE     R3, R2, Loop
```

ADD.D is the instruction for adding doubles and DADDI is the instruction for adding integers. Use the following set of latencies between different instructions when the second instruction has a flow dependence on the first instruction. If no latency is given for a pair of instructions, then assume the latency is one.

first instruction	second instruction	
	ADD.D	S.D
ADD.D	4	3
L.D	3	2

For an example of how to read the table, if the result of an ADD.D instruction is used by a following ADD.D instruction, the following ADD.D instruction will have to wait until 4 cycles later to begin execution. In other words, there will essentially be 3 nops or interlocks between the two ADD instructions. If only one instruction is scheduled in between the two, there will still be two cycles of interlock.

The body of the loop given above requires 14 cycles to execute (plus any cycles it takes to finish the BNE instruction). Use list scheduling with the following priorities to find a better schedule:

- (a) Avoid stalls with previously scheduled instructions.
- (b) Does the instruction interlock with any immediate successors?
- (c) How many successors does the instruction have?
- (d) Is the instruction on the critical path?

Show the DAG that you use to perform the scheduling. Assume that the BNE instruction must occur last.

Create a schedule using software pipelining. Show your DAG copies and the register assignments. How many cycles per iteration are needed when the software pipelined code reaches steady state?

4. [20 Points] Do Exercise 9.3.5 in book. “Suppose the set F of functions for a framework are all of gen-kill form. That is, the domain V is the power set of some set, and $f(x) = G \cup (x - K)$ for some sets G and K . Prove that if the meet operator is either (a) union or (b) intersection, then the framework is distributive.”
5. [20 Points] Activity analysis is a data-flow analysis needed in the context of Automatic Differentiation. One piece of activity analysis is a forward data-flow analysis called vary analysis. The goal of vary analysis is to determine the set of variables in a procedure that depend on a specified subset of independent variables at various points in the program. For example, in the below program, if x is the independent variable of interest, then $OUT(s1) = \{a, x\}$, $OUT(s2) = \{a, b, x\}$, $OUT(s3) = \{b, x\}$, and $OUT(s4) = \{b, x, y\}$.

```
// independent = {x}
s1:  a = x + 3;
s2:  b = a * 2;
s3:  a = c;
s4:  y = a + b;
```

Vary analysis has the following specification:

- must or may: may
- direction: forward
- meet: union
- data-flow values: sets of variables
- initial value: empty set
- $OUT[\text{entry}] =$ the set of independent variables
- transfer function: $f(X) = GEN \cup (X - KILL)$

GEN is defined as the set of variables being defined in the statement, if a variable in X is being used in the statement. $KILL$ is the set of variables being defined in the statement.

Perform vary analysis on the following procedure using iterative data-flow analysis. For each statement in the loop, show the OUT data-flow set. How many iterations of iterative data-flow analysis are required for convergence assuming that the statements are visited in the order they are listed in the program?

```
// independent = { a }
for ( i=0; i<N; i++ ) {
    e = a + b + c + d;
    d = c + b;
```

```
        c = b * a;
        b = a - 3;
    }
```

6. [10 Points] For the following program, draw the control-flow graph and perform copy propagation followed by dead-code elimination.

```
S1:    j = 0
S2:    y = read()
S3:    x = y
S4:    z = x
S5:    L1:
S6:    if j > 10 goto L2
S7:    z = x
S8:    j = j + 1
S9:    goto L1
S10:   L2:
S11:   print x, y, z
```

7. [10 Points] Show the following sets for each block in Figure 9.33 in the book: `anticipated_out`, `anticipated_in`, `available_in`, `available_out`, `earliest`, `postponable_in`, `postponable_out`, `postponable_in`, `latest`, `used_in`, and `used_out`. A grid format as was done in class would be preferable.