

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. [20 Points] Given the following context-free grammar, draw the parse tree that is generated by SableCC for the given input.

Tokens

```
t_assign = '=';
t_l_brc  = '{';
t_l_par  = '(';
t_plus   = '+';
t_r_brc  = '}';
t_r_par  = ')';
t_sc     = ';';
t_equal  = "=="
```

```
t_false   = 'false';
t_if      = 'if';
t_true    = 'true';
```

```
t_id      = letter (letter | digit | underscore)*;
```

```
t_blank   = (' ' | eol | tab)+;
```

Ignored Tokens

```
t_blank;
```

Productions

```
StmtList = StmtList Stmt | /*empty */ ;
```

```
Stmt = IfStmt | AssignStmt | ForStmt | ... ;
```

```
AssignStmt = t_id t_assign Expr t_sc ;
```

```
Expr = t_id | Expr t_plus Expr | Expr t_equal Expr ;
```

```
IfStmt: t_if t_l_par t_true t_r_par t_l_brc StmtList t_r_brc
        | t_if t_l_par t_false t_r_par t_l_brc StmtList t_r_brc ;
```

```

=====
Input
=====
    if ( true ) {
        x = a + b + c;
    }

```

2. [20 Points] Instruction Scheduling. The loop shown below

```

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

```

can be converted to MIPS assembly as follows:

```

// body of the loop
Loop:
L.D      F0, 0(R1)      // F0 = &R1
ADD.D    F4, F0, F2     // F4 = F0 + F2
S.D      F4, 0(R1)
DADDI    R3, R3, #-1    // R3 = R3-1
DADDI    R1, R1, #-8    // R1 = R1-8
BNE      R3, R2, Loop

```

Use the following set of interlocks/latencies between different instructions when the second instruction depends on the first instruction. For example, the architecture will interlock for 3 cycles between two ADD.D instructions if the instructions are scheduled one after the other. If only one instruction is scheduled in between the two, there will still be two cycles of interlock.

first instruction	second instruction	
	ADD.D	S.D
ADD.D	3	2
L.D	1	0

The body of the loop given above requires 9 cycles to execute. Use list scheduling with the following priorities to find a schedule that only requires 7 cycles:

- (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.

3. [20 Points] On page 43 of the data-flow analysis handout, the authors sketch a proof showing that if statement (9.22) holds, then statement (9.23) holds. Write up the same proof but put in some of the details that they do not explicitly describe. For example, what specific parts of the definition of greatest upper bound support the final step of the proof?

4. [20 Points] Do Exercise 9.3.5 in the data-flow analysis handout. "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;
}

```