EECS 665

Final Exam Review Questions

1. Give the three-address code (like the quadruples in the csem assignment) that could be emitted to translate the following assignment statement. However, you may **not** use the [ ] quadruple operation. Assume that `a` is a global integer array with 5 rows and 10 columns. Assume that `v` is a local integer variable (the first one defined) and `i` and `j` are integer parameters (the first two) passed to the function. Further assume that the arrays are stored in *row-major* order.

```
v = a[i][j];
```

2. Use Figures 6.36 and 6.37 to translate the following statement in two passes (no backpatching).

$$if(x > 10 \quad \&\& \;\; y! = 0 \quad || \quad x == y) \quad x = y;$$

3. Use Figures 6.43 and 6.46 to translate the expression in a single pass (use backpatching). Start outputting the code from address 20.

$$if(x > 10 \quad \&\& \quad y! = 0 \quad || \quad x == y) \quad x = y;$$

4. Below is a syntax directed definition, where **A**, **B**, **C**, and **D** are nonterminals and **a**, **b**, and **c** are terminals. Indicate for each rule if the attribute being assigned is synthesized or inherited attribute.

Production	Semantic Rules	Synthesized or Inherited?
A $\rightarrow$ B C	C.a := B.a	
	A.a := C.b	
B $\rightarrow$ D a	D.a := B.a	
C $\rightarrow$ C_1D	C_1.a := D.a	
	C.b := C_1.b	
C $\rightarrow$ b	C.b := b.val	
D $\rightarrow$ c	D.a := c.val	

5. The following RTLs represent instructions from part of a function that a compiler has generated for a machine with **no** delay slots. Identify the leaders and depict the RTLs in a flow graph of basic blocks (note that each RTL should appear in a basic block).

```
                 .
                 .
                 PC = IC < 0 → L9;          /* this is the only branch to L9 */
                 r[3] = M[r[1] + _a];
                 r[3] = r[3] * 8;
                 IC = r[2] ? r[3];
                 PC = IC < 0 → L10;         /* this is the only branch to L10 */
L9
                 PC = L11;
                 r[3] = r[3] + 1;
L10
                 r[4] = M[r[1] + _a];
                 r[4] = r[4] * 8;
                 r[2] = r[2] + r[4];        r[4]:
L11
                 .
                 .
```

6. The following RTLs represent **SPARC** instructions. The **SPARC**, like most RISC machines, is a 3 address machine and can only access memory by using load or store instructions. Apply the instruction selection optimization (as described in class) to the following RTLs. Show each step of combining RTLs separately. (10 points)

```
1              r[1] = r[30] + j;
2     {1}      r[2] = M[r[1]];          r[1]:
3              r[3] = 4;
4              r[4] = 1;
5     {2,4}    r[2] = r[2] + r[4];      r[4]:
6     {3}      r[3] = r[14] + r[3];
7     {5,6}    M[r[3]] = r[2];          r[2]:r[3]:
```

7. Transform the respective codes using the specified optimizations. **Also describe the benefit of performing the optimization**.

   (a) Procedure integration or inlining.

   ```
   int func1(){
     func2();
     func2();
     printf(``Inside func1'');
   }

   int func2(){
     printf(``Inside func2'');
   }
   ```

   (b) Procedure specialization or cloning

   ```
   int func1(int x, int y){
     int y;

     y = multiply(x, y);
     y = multiply(x, 16);
   }

   int multiply(int a, int b){
     return (a * b);
   }
   ```

   (c) Tail recursion elimination

   ```
   int func1(){
     int a, b;

     scanf(``Enter a, b: '', &a, &b);
     a = a * b;

     func1();
   }
   ```

   (d) Loop unrolling, using an unroll factor of 2

   ```
   int func1(){
     int i, a;
     a = 0;
     for(i=0 ; i<100 ; i++){
       a += i;
     }
   }
   ```

(e) Loop collapsing

```
int func1(){
  int a[100][300];

  for(i=0 ; i<100 ; i++){
    for(j=0 ; j<300 ; j++)
       a[i][j] = 0;
  }
}
```

(f) Loop fusion

```
int func1(){
  int a[100], b[100];

  for(i=0 ; i<100 ; i++){
    a[i] = 0;
  }
  for(i=0 ; i<100 ; i++){
    b[i] = 0;
  }
}
```

(g) Loop interchange

```
int func1(){
  int a[100][300];

  for(i=0 ; i<100 ; i++){
    for(j=0 ; j<300 ; j++)
       a[i][j] = 0;
  }
}
```

(h) Scalar replacement

```
int func1(){
  int a[101];

  for(i=0 ; i<100 ; i++){
    a[100] += a[i];
  }
}
```

(i) Also, note the examples on the slides for
    (i) loop invariant code motion, and
    (ii) register allocation,

8. Answer the following:

   (a) Give the advantages and disadvantages of using a three-address form of intermediate representation over a zero-address representation.

   (b) What is static checking? Why is static checking preferable to dynamic checking?

   (c) Describe the rules for type checking.

   (d) What is coercion, overloading, and polymorphism? Give an example of each in the C language.

   (e) Describe the difference between synthesized and inherited attributes. Which type of attribute does YACC support?

   (f) What is the purpose of: (i) register assignment and (ii) instruction selection during code generation?

   (g) What are addressing modes? When is each type typically used?

   (h) Explain the need for runtime stack management? What does it involve?

   (i) What do you understand by "evaluation order of arguments"?

   (j) What is code optimization ? Why is the term code *optimization* a misnomer?

   (k) How are the following types of optimizations important: (i) function call optimizations, (ii) loop optimizations, (iii) memory access optimizations, (iv) control flow optimizations, (v) data flow optimizations, and (vi) machine specific optimizations.

   (l) Explain each of the following optimizations in one-two lines: (i) procedure inlining (ii) tail recursion elimination (iii) loop invariant code motion (iv) loop strength reduction (v) induction variable elimination (vi) loop unrolling (vii) register allocation (Vs. register assignment)

   (m) What are activation records?

   (n) What are *caller-save* and *callee-save* registers?

   (o) What is a *call-graph*?

   (p) What are *non-local* names?

   (q) Identify the fields of a general activation record in Figure 7.5 in your text-book.

   (r) Describe characteristics, advantages, and disadvantages of the following data allocation strategies: (i) static allocation, (ii) stack allocation, and (iii) heap allocation.

   (s) Name and describe the features of the two types of *implicit* heap storage reclamation strategies. What is its main drawback?

   (t) What is an access link? How is access link different from a control link?