INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR
Department of Computer Science and Engineering
Autumn End-Semester Examination, 2018-19

Time: **3 hours** Full Marks: **99**

Subject: **Compilers (CS31003)**

Instructions: *(1) Answer all the questions. (2) In case of reasonable doubt, make practical assumptions and write that on your answer script. (3) The part of each question must be together.*

1.

```
void func_1(int a[], int n)
{
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = a[i];
        j = i-1;
        while (j >= 0 && a[j] > key) {
            a[j+1] = a[j];
            j = j-1;
        }
        a[j+1] = key;
    }
}
```

(a) Generate the intermediate representation (IR) as quadruples corresponding to the above source code. Assume the size of *int* is 4 bytes and the first address location is 1. DO NOT optimize your code while generating. No credit for optimized IR generation.

(b) Draw the control flow graph (CFG). Mark the basic blocks, and loops in your graph. You may use the address locations to mark the basic blocks and basic blocks to mark the loops.

(c) Optimize each basic block by

i. Eliminating common local sub-expression;

ii. Copy propagation and

iii. Dead code elimination.

Explain your steps by mentioning the optimization done and strictly follow the order of optimization.

(d) Write down the final optimized version of IR by renumbering each IR statement as $s1$, $s2$, ... Mark the control flow.

Marks: 5+4+9+2=20

2. (a) Formulate a suitable DFA problem for Available Expressions to determine the common sub-expressions across blocks at the entry and exit of every basic block that you obtained from **question 1**. Clearly state the definitions for the data-flow direction, confluence operator, initial conditions, and IN, OUT, GEN and KILL sets for the DFA. Outline the algorithm to solve the DFA equations.

(b) Using your formulation, compute the Available Expressions at the entry and exit of every block of the control flow graph as you obtained and renumbered in **question 1d**. Show the iterations for this computation.

(c) Using the Available Expressions, eliminate the common global sub-expressions.

(d) Next, perform the copy propagation. Show the passes.

(e) For each basic block eliminate dead code.

(f) Write down the final optimized version of IR by renumbering each IR statement as $d1$, $d2$, ... Redraw the control flow graph.

Marks: 4+10+4+6+3+2=29

3. (a) Write down an algorithm to identify the basic and derived induction variable of a loop from a set of IR.

(b) Use your algorithm to determine the induction variable(s) from the optimized IR as you obtained from **question 2f**. Assume 4-bytes are required to store an integer variable.

(c) Further loop optimize the IR that you obtained at **question 2f**. Mention the optimizations you have done.

(d) Comment on the unrolling of loops for this optimized IR. You may use some example.

Marks: 5+5+6+4=20

(d) Write down the final optimized version of IR by renumbering each IR statement as $s1$, $s2$, ….. Mark the control flow.

Marks: 5+4+9+2=20

2. (a) Formulate a suitable DFA problem for Available Expressions to determine the common sub-expressions across blocks at the entry and exit of every basic block that you obtained from **question 1**. Clearly state the definitions for the data-flow direction, confluence operator, initial conditions, and IN, OUT, GEN and KILL sets for the DFA. Outline the algorithm to solve the DFA equations.

(b) Using your formulation, compute the Available Expressions at the entry and exit of every block of the control flow graph as you obtained and renumbered in **question 1d**. Show the iterations for this computation.

(c) Using the Available Expressions, eliminate the common global sub-expressions.

(d) Next, perform the copy propagation. Show the passes.

(e) For each basic block eliminate dead code.

(f) Write down the final optimized version of IR by renumbering each IR statement as $d1$, $d2$, ….. Redraw the control flow graph.

Marks: 4+10+4+6+3+2=29

3. (a) Write down an algorithm to identify the basic and derived induction variable of a loop from a set of IR.

(b) Use your algorithm to determine the induction variable(s) from the optimized IR as you obtained from **question 2f**. Assume 4-bytes are required to store an integer variable.

(c) Further loop optimize the IR that you obtained at **question 2f**. Mention the optimizations you have done.

(d) Comment on the unrolling of loops for this optimized IR. You may use some example.

Marks: 5+5+6+4=20

4. (a) Perform Live Variable analysis over the control flow graph (as you obtained at **question 3c**) to determine live variables at the entry to and exit from every basic block. Show the necessary steps of DFA.

(b) Using the liveness information draw the interference graph and estimate the number of registers required.

(c) Explain a technique for register allocation and use your method to assign variables to your estimated registers.

Marks: 6+4+5=15

5. (a) Consider the following basic block. Assume only a is live on exit. Determine the minimum number of registers required for code generation as per the Sethi-Ullman algorithm of the given IR.

Addr	*IR*	*Addr*	*IR*
1	$t1 = i * 10$	6	$t3 = t3 + j$
2	$t1 = t1 + k$	7	$t3 = t3 * 4$
3	$t1 = t1 * 4$	8	$t4 = c[t3]$
4	$t2 = b[t1]$	9	$a = t2 + t4$
5	$t3 = k * 10$		

(b) Let us assume your estimation on the minimum number of register required as per Sethi-Ullman algorithm is r. Use r numbers of registers to generate the shortest sequence of instructions (code) following the Sethi-Ullman algorithm. For each IR indicate the content of the register descriptors and the address descriptors. Make a table as given below. No credit for code generation without the Sethi-Ullman algorithm.

IR	*Target Code*	*Register Descriptor*			*Address Descriptor*						
		R1	....	*Rr*	*a*	*b*	*c*	*t1*	*t2*	*t3*	*t4*