CS 243 Final Examination

Winter 2023

March 21, 2023

This is an open-book, open-notes, open-laptop, *closed*-network exam. You have 3 hours to complete the exam. The examination has 8 problems worth 180 points. Please budget your time accordingly. Write your answers in the space provided on the exam. If you use additional scratch paper, please turn that in as well.

Your Name: _____ SUNet ID: _____

The following is a statement of the Stanford University Honor Code:

1. The Honor Code is an undertaking of the students, individually and collectively:
   (a) that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;
   (b) that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.
2. The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.
3. While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

Signature: _____

Problem	#1	#2	#3	#4	#5	#6	#7	#8	**Total**
Max	21	14	15	20	25	25	30	30	180

Problem 1. Short questions. You must **justify your answer** to each question in 1–2 sentences. [21 points, 3 points each]

1. True or False. A semi-lattice of infinite domain size must have an infinite descending chain.

2. The Cartesian product of two sets is defined as $A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}$. $\mathbb{Z}^+$ is the set of all positive integers. Consider the set of $\mathbb{Z}^+ \times \mathbb{Z}^+$ as the domain. We define a meet operator $\min((x_1, x_2), (y_1, y_2)) = (\min(x_1, y_1), \min(x_2, y_2))$. Does this form a semilattice? If yes, write down the top element and bottom element if they exist. If no, briefly explain why.

3. Let a be the number of live ranges in a program, b be the number of *merged* live ranges (as defined for register allocation), and c be the number of variables in the same program translated to static single-assignment (SSA) form. Express the relationship between a, b, and c using one or more inequalities. Explain your answer.

4. After successfully software pipelining a loop with an initiation interval T, registers `R1` and `R2` in the original program have live ranges of $3 \times T$ and $4 \times T$ cycles, respectively. Suppose you have 10 registers to devote to `R1` and `R2`, how many times would you unroll the steady state of the loop?

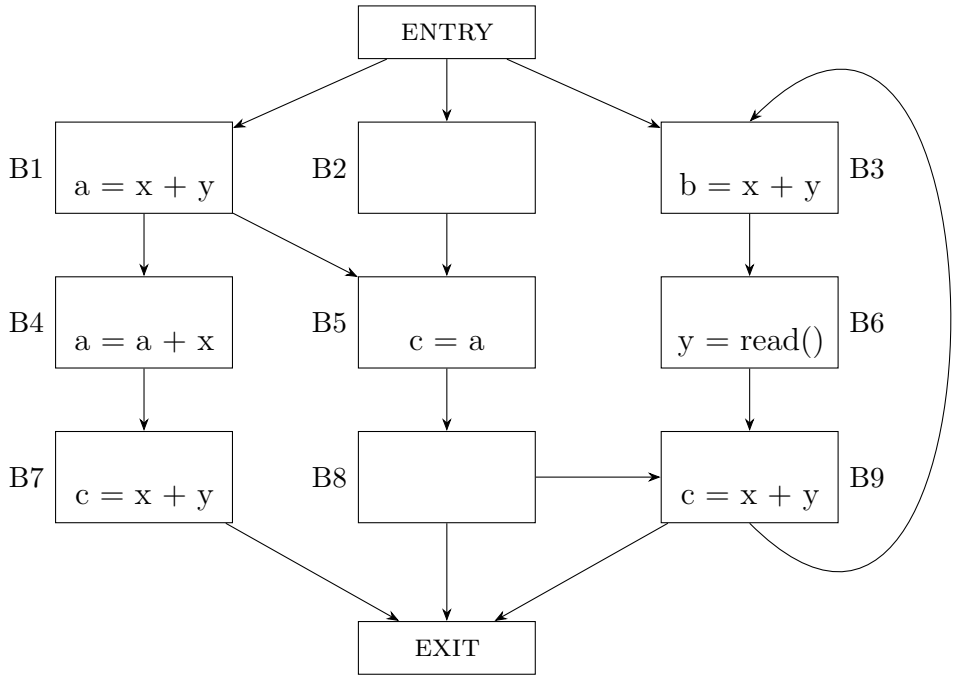
5. True or False. A single-issue machine will not benefit from global instruction scheduling because it cannot issue more than one instruction per cycle.

6. True or False. The Fourier–Motzkin elimination (FME) algorithm returns real solutions rather than integer solutions. But since loop indices are always integers, FME is unsuitable for conducting data dependence analysis for automatic loop parallelization.

7. True or False. If the two outermost loops of a 3-deep loop can be parallelized without using any synchronization, then the loop must be fully permutable.

Problem 2. Partial Redundancy Elimination [14 points]

Answer the following question based on the program below.



Show the result of running partial redundancy elimination. **Don't apply any optimization other than PRE.** What's the final optimized flow graph? You don't need to show the intermediate steps.

Problem 3. Garbage Collection [15 points, 5 points each]

You have the following automatic memory management systems at your disposal:

(a) Reference counting

(b) Generational garbage collector with incremental marking

(c) Non-generational tracing garbage collector

Which garbage collector would you choose for each of the following scenarios? Explain your choice based on performance and usability.

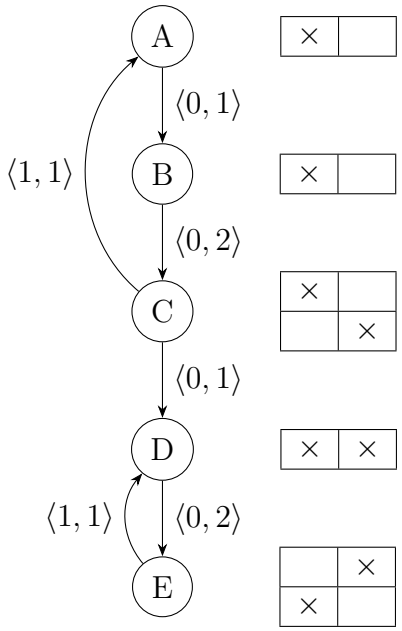
1. A numerical program running on a multiprocessor that only allocates memory at the beginning of the program, and makes ten passes through all allocated memory.

2. A programming language designed for resource-constrained embedded devices, where small executable code size is more important than speed.

3. A single-threaded dynamic language that allocates a lot of short-living intermediate objects on the heap.

Problem 4. Software Pipelining [20 points]

Consider the following dependence graph for a single iteration of a loop, with resource constraints:



1. What are the bounds on the initiation interval T according to the precedence and resource constraints for this program? [4 points]

2. Is it possible to software-pipeline this loop with the minimum bound found in the previous part? If not, what is the minimum possible initiation interval?

   Show the modulo reservation table for the optimal software-pipelined schedule. Also show the code schedule for an iteration in the source loop. [8 points]

3. Can the scheduling algorithm described in class produce the optimal schedule for this loop? If not, show the modulo reservation table and code schedule generated by the algorithm. [6 points]

Problem 5. Pointer Analysis [25 points]

Consider the following simple implementation of a singly-linked list. We use static methods to avoid `this` pointers which may be ambiguous.

```
1  public class Node {
2      public Node n = null;
3
4      // This function is callable as: Node.extend(a, b)
5      public static void extend(Node a, Node b) {
6          if (b == null)
7              b = new Node();    // h3
8          Node c = Node.getTail(a);
9          c.n = b;
10     }
11
12     // This function is callable as: Node.getTail(d)
13     public static Node getTail(Node d) {
14         if (d.n == null)
15             return d;
16         Node e = d.n;
17         return Node.getTail(e);
18     }
19
20     public static void main(String[] args) {
21         Node x = new Node();    // h1
22         Node y = new Node();    // h2
23         Node.extend(x, y);
24     }
25 }
```

Apply flow-insensitive context-insensitive pointer analysis to the program and write down all `vP` and `hP` tuples derived from the analysis. `h1`, `h2`, `h3` are the allocation sites, and `a`, `b`, `c`, `d`, `e`, `x`, `y` are all the variables.

(This page is intentionally blank.)

Problem 6. Satisfiabiliy Modulo Theories [25 points]

In this problem, we use SMT to analyze the following function. We assume that `data` is a zero-indexed array of length `N`.

```
1  int foo(int data[], int N) { {
2      int v = 0;
3      int i = 0;
4      while (i <= N) {
5          if (data[i] > v) {
6              v = data[i];
7          }
8          i = i + 1;
9      }
10     return v;
11  }
```

1. Notice the while loop on lines 4–9. One method for handling a loop in SMT analysis is to unroll it several times. Show the SSA form of the code when you unroll the loop twice (convert the first two iterations of the `while` loop into `if` statements). [12 points]

2. Let P be the SMT constraints based on the SSA form of the code with the loop unrolled twice. Write a SMT formula that is satisfied if there is an out-of-bound array access. Make sure to take into account path conditions for each array access.

   Will an SMT solver return `SAT` or `UNSAT`? If `SAT`, give a satisfying model. If `UNSAT`, briefly explain why. [5 points]

3. Suppose we can safely assume that the `data` array has at least 10 elements. Still unrolling the loop twice, write down the updated SMT formula that is satisfied if there is an out-of-bound array access. Will an SMT solver return `SAT` or `UNSAT`? If `SAT`, give a satisfying model. If `UNSAT`, briefly explain why. [4 points]

4. Again assume that the `data` array has at least 10 elements. What if we unroll the loop 20 times? Will an SMT solver return `SAT` or `UNSAT` on the updated constraint? If `SAT`, give a satisfying model. If `UNSAT`, briefly explain why. [4 points]

Problem 7. Parallelization with Locality Optimization [30 points]

Consider a multiprocessor with 8 processors, where each processor has two levels of caching, and there is a **significant overhead** in moving data between processors. It is therefore very important to maximize locality. The following function is a simplification of the ADI Integration Method; it is executed repeatedly in the full program. Your task is to parallelize the function, while maximizing locality. Assume that the array `A` is much larger than 8 times the size of the first-level cache in each processor.

```
1  for i = 1 to N {
2      for j = 1 to N {
3          A[i,j] = 0.5 * (A[i,j] + A[i,j-1]);
4      }
5  }
6  for i = 1 to N {
7      for j = 1 to N {
8          A[i,j] = 0.5 * (A[i,j] + A[i-1,j]);
9      }
10 }
```

1. Without performing any code transformations, indicate which of the loops in this function are do-all loops? (Refer to the loops by their line numbers.) Suppose all such do-all loops are parallelized by inserting barriers at the end of each loop. Discuss the cache performance for the individual processors, as well as the synchronization and communication cost between processors. [6 points]

2. Does this function have any communication-free parallelism (i.e., can you transform the program into a **single** nested loop nest whose one or more outer loops are do-all loops)? If so, show the SPMD code. If not, briefly explain why. [6 points]

3. Does this function have pipelined parallelism? If so, show the transformed code with as many outermost fully permutable loops as possible. (You only need to show the transformed sequential code, there is no need to parallelize it with synchronization.) [9 points]

4. Describe how you would parallelize this function. Discuss the cache performance for the individual processors, as well as the synchronization and communication cost between processors. [9 points]

Problem 8. Optimizing Dynamically Typed Languages II [30 points]

Consider a dynamically-typed language similar to JavaScript and Python. The language has two kinds of instructions of interest to us:

- $x = T(\ldots)$, where T is one of `int`, `float`, and `string`. This sets the variable x to a value of type T.
- $x = y + z$, where x, y, z are all variables.

Unlike statically-typed languages like C and Java, the type of a program variable is allowed to change in this language. As an example, the program `x = int(1); x = float(3.14)` will result in variable `x` having an `int` value at first, but afterwards a `float` value.

The addition operator `+` accepts operands of any types with the following rules:

1. The sum of two values of the same type has that same type.
2. The sum of an `int` and a `float` is a `float`.
3. The sum of a `string` and a value of any type is a `string`.

Suppose we are writing an optimizing compiler for this language. Because there are special instructions for adding two integers (`iAdd`), adding two floats (`fAdd`), and adding two strings (`sAdd`), your task is to identify those additions that can be replaced with such special instructions at compile time.

To get full points, your dataflow analysis should identify as many opportunities for optimization as possible.

This problem has four parts: (1) Define your dataflow analysis by filling in the table below. (2) State clearly how you perform the optimization. (3) Is your dataflow analysis monotone? Explain. (4) Is your dataflow analysis distributive? Explain.

(Questions begin on the next page.)

1. Dataflow analysis specification. You may assume that each instruction is in its own basic block. [22 points]

Direction of your analysis (forward/backward)	
Lattice elements and meaning	
Meet operator	
Is there a top element? If yes, what is it?	
Is there a bottom element? If yes, what is it?	
Transfer function	
Boundary condition	
Interior points	

2. How do you optimize the program? [2 points]

3. Is your dataflow analysis monotone? Briefly sketch a proof for it if so, or provide a counterexample if not. [3 points]

4. Is your dataflow analysis distributive? Briefly sketch a proof for it if so, or provide a counterexample if not. [3 points]