18/4/2023, 2PM to 5PM. CS323-Compilers, End-semester Examination, Spring 2023. Max Points: 40

There are six questions in this paper. State your assumptions clearly. This exam is open-book, open-notes. No electronic devices allowed.

1. Of the following optimizations, name the ones that are always safe and ones that are always profitable. **(2 points)**
   a. Factoring loop invariants
   b. Constant propagation

2. Perform Common Subexpression Elimination (CSE) on the sequence shown on the right.
   a. For each statement, after executing that statement, show available and killed expression(s) **(2.5 points)**
   b. Suppose E and C were aliased (recall that aliased means referring to same location in memory) in the sequence shown. How will that change the results of CSE? **(1 point)**
   c. Suppose B and D were aliased in the sequence shown. How will that change the results of CSE? **(1 point)**
   d. Describe the effect that aliasing has on CSE w.r.t. number of sub expressions and code redundancy. Your answer must be drawn from your answers to parts b and c. **(0.5 points)**

```
1. A = 7;
2. B = A + 2;
3. C = A + B;
4. D = C + B;
5. B = C + B;
6. A = A + B;
7. E = C + D;
8. F = C + D;
9. G = C + B;
10. H = E + F;
```

3. For the sequence shown in the previous question (above), perform register allocation with 3 registers. When choosing registers to allocate, always allocate the lowest-numbered register available. When choosing registers to spill, choose the register holding a value that will be used farthest in the future (in case of a tie, choose the lowest-numbered register).
   a. Show what variables are live at each statement (assume no variables are live out of statement 10) **(1 point)**
   b. For each statement, show register-variable mapping after performing bottom-up register allocation. You must use the method discussed in class. Mark dirty registers with a '*'. assignment. **(5 points)**
   c. For each statement, show code generated. **(5 points)**
   d. Draw interference graph for the code. **(1 point)**

4. Consider the loop nest:

```
for(i=0;i<n;i++) {
  for(j=0;j<n;j++) {
       X[2*i+1][j] = Y[i+1][j];
       Y[i][j] = X[2*i-1][j];
   }
}
```

   a. Describe the dependences that exist on the X and Y arrays. For each of the arrays, if no dependence exists say "No dependence". If dependence exists, say what kind of dependence exists ("Anti", "Flow", "Output"), Give the distance and the direction vectors. **(6 points)**
   b. Which loop can be parallelized above? Assume no other optimizations are performed. **(1 point)**
   c. Which of the following transformations i) strength reduction ii) identifying invariants and factoring/moving out of the loop iii) loop fusion iv) loop distribution / loop splitting increases the number of loops that can be executed in parallel OR that makes it more easier to exploit existing parallelism? Show the resulting code after transformation and explain how it enhances parallelism **(4 points)**

5. You are moving from a system with no cache memory to a system with cache memory. For each of the optimizations below, tell whether it is "more useful", "less useful", or "does not matter" when you move. Explain 'Why' in less than 3 sentences for each.
   a. Register allocation **(1.5 points)**
   b. Loop interchange **(1.5 points)**
   c. Instruction Scheduling **(1.5 points)**
   d. Loop tiling **(1.5 points)**

6. Given below brief notes about a couple of analysis problems:

Very-busy expression: an expression is very busy if its value will be used on all possible execution paths. *Available expression*: an expression is available if it has already been computed and recomputing it would be redundant. In addition, consider all the analysis problems that you have studied in class (live variable analysis, constant propagation, reaching definitions etc.). Draw a table like the one shown and fill in the cells giving at least one example problem. **(4 points)**

	Forward-Flow	Backward-Flow
All-Path		
Any-Path		