Computer Science 320: Final Examination

May 22, 2013

Name _____

You have exactly 3 hours to answer the following questions. This final is closed book/closed notes. For partial credit, show all work. Put your name on the bottom of every page. Write out and sign the Honor Code pledge on this page before turning in the test.

"I pledge my honor that I have not violated the Honor Code during this examination."

Problem 1: (5%)
Build a *Deterministic Finite Automaton* (DFA) that recognizes the regular expression $(ab|c)^*$.

Problem 2: (10%)
Write a regular expression for:

A: Binary numbers that are multiples of four

B: Strings over the alphabet $\{a, b, c\}$ where the first a precedes the first b

Problem 3: (10%)
Is the following grammar in SLR?

$S' \rightarrow S\$$
$S \rightarrow$ b A c
$A \rightarrow$ d
$A \rightarrow A$ a A

Problem 4: (20%)
Given the *control-flow graph* (CFG) in Figure 1:
A: Can constant propagation be applied to this program? If so, apply this optimization to the program. What about constant folding? If so, apply this optimization to the program.

B: Does the program contain any common subexpressions? If so, transform the program so that all common subexpressions are eliminated. Show the results of all dataflow analyses, and specify exactly what conditions are satisfied that enable the optimization to be performed.

Problem 5: (40%)
Given the *control-flow graph* (CFG) in Figure 2:

A: What algorithm does the compiler use to identify loops in a CFG? Apply the algorithm to the CFG in order to systematically identify all natural loops in the program.

B: A microprocessor has 3 registers. Use liveness analysis to determine the live ranges of each variable, create an interference graph, and then use graph coloring to do register allocation for registers a, b, c, d. Is there any spilling required?

C: Rewrite the CFG so that it is in SSA form. Make sure that a minimum number of ϕ-functions are inserted.

Problem 6: (15%)
Assume we have the following program, which iterates over a 32-bit integer array `A` of `n` elements and finds the sum of all elements in the array:

```
sum = 0
for(i = 0; i < n; i++) sum += A[i]
```

The following is the assembly code for the program, where the base location of array `A` is located in `r4`:

```
      r2 = 0
      r3 = n*4
      r4 = A
loop: r1 = M[r4]
      r2 = r2 + r1
      r4 = r4 + 4
      branch r4 < r3, loop
```

Transform this program using software pipelining. Assume a machine with infinite issue width, 32 registers, 2 ALUs, and one memory unit. Also assume that memory operations take 2 cycles, and all other operations take 1 cycle.
You may use the instruction `brsh` as a "branch and shift" instruction, which in addition to branching shifts registers `r10`, ..., `r32` up by 1 (e.g. the contents of `r10` is moved into `r11`, etc.). `r32` is shifted into `r10`. `brsh` does not need to use an ALU and takes 1 cycle.
Make sure to define the prologue, kernel, and epilogue.

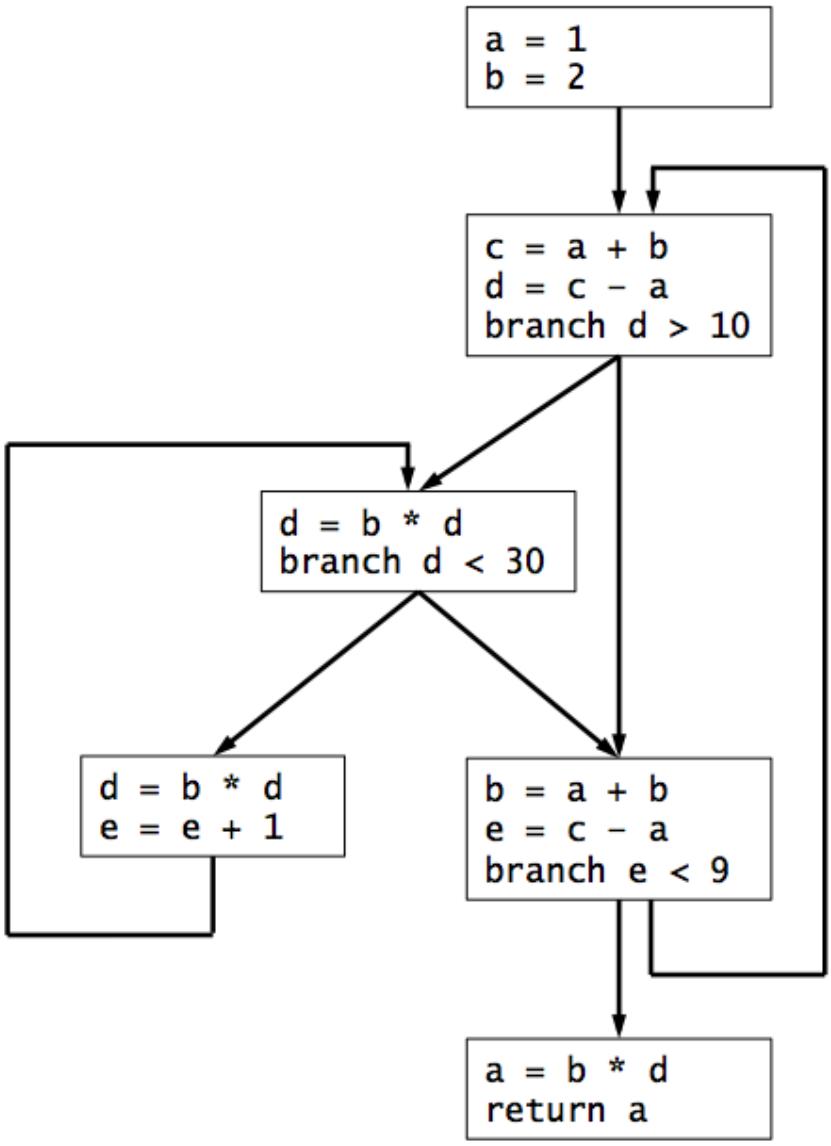

Figure 1: CFG for Problem 4

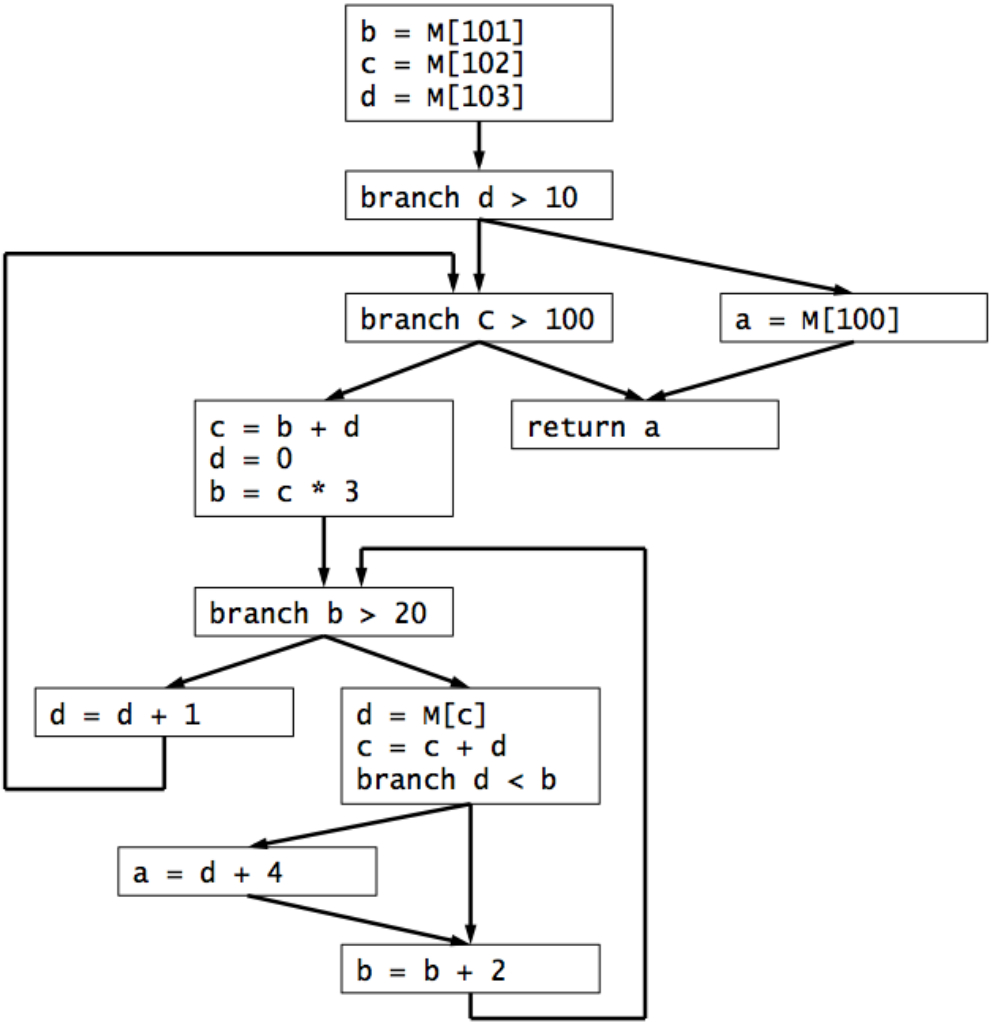

Figure 2: CFG for Problem 5