

CMSC 430 (Spring 2011)

Practice Problems 5

1. Instruction scheduling

Consider scheduling the code below using list scheduling. All instructions must complete before executing the *jmp* instruction. Assume the following instruction latencies:

- 2-cycle latency for load
- 1-cycle latency otherwise

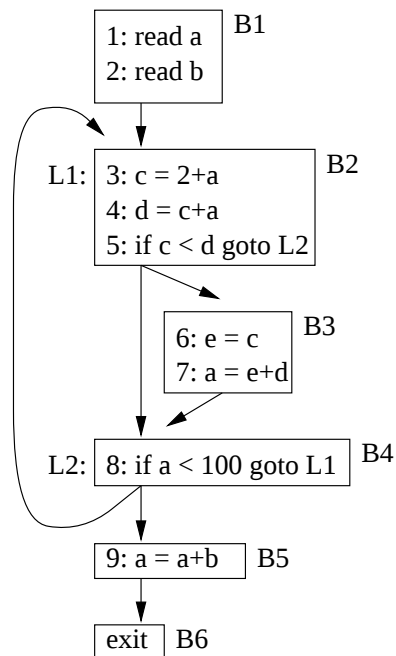
<op> <dst, s1, s2>

```
1  load  r1, a
2  add   r2, r1, #4
3  store x, r2
4  load  r3, b
5  mult  r4, r3, r2
6  load  r1, c
7  add   r5, r1, r3
8  store y, r5
9  load  r6, d
10 mult  r7, r5, #1
11 store z, r7
12 jmp
```

- Build the precedence graph for the instructions. Mark dependences as flow, anti, or output. You can ignore transitive dependences.
- Calculate the critical path for the instructions.
- Schedule the instructions for a single-issue processor, using forward list scheduling. Showing candidates instructions at each cycle. Prioritize candidates using 1) critical path, 2) latency of instruction, 3) number of children.
- Schedule the instructions as above, for a two-issue VLIW processor.
- How could you change register assignments to improve instruction schedules in the code?

2. Register allocation

Consider the flow graph below. Each statement is labeled by its statement number and each basic block is labeled in the upper right hand corner.



Use statement numbers or basic block numbers to indicate live ranges as appropriate.

- What are the live ranges for a global top-down allocator?
- Draw the interference graph for the live ranges.
- Use the graph-simplification method to find a coloring for this graph, using 4 registers.
- Can you color this graph with fewer colors?
- If spilling is needed, which live range would be spilled first? Why?
- Draw the spill code needed if the value for *c* is spilled.
- What is rematerialization? Are there any such opportunities in the code?

3. Dependence analysis

Consider the following loop in Fortran (where arrays are stored in column-major format):

```
A(N,N), B(N)
do i = 1,N
  do j = 2,N
    A(i,j) = A(i,j-1)+B(j)
  enddo
enddo
```

- (a) What are the dependences in the loop?
- (b) What reuse exists in the loop? For each, give type, reference(s), and loop carrying reuse.
- (c) Which loop permutation is preferred? Calculate by estimating number of cache lines accessed by each loop. Assume 4 array elements fit on each cache line. Show your work.
- (d) Is it legal to interchange the i & j loops to make j the outer loop and i the inner loop?
- (e) Are either of the loops parallel? Explain?