California Institute of Technology
Department of Computer Science
Computer Architecture

CS184a, Winter 2003 Assignment 3: Instructions Wednesday, January 22

Due: Monday, January 27, 9:00AM

Everyone should do problems 1–5. (If any of these take you more than 30 minutes to see the path to the solution, definitely let me know.)

You should use a schematic capture or drawing program for circuits (where appropriate).

1. Consider a simple, sequential, non-branching, programmable datapath with a single one-bit output computational unit. For this problem consider two possible functional units: a 3-input NAND and a 3-input LUT. Now, let us consider implementing a 5-input parity function (XOR5) on each of these programmable datapaths.

   - Define the primitive instruction (pinst) for each of the units (what bits are included, what do they do, how many of each).
   - How many instruction bits are required to specify the computation for each instruction in the two cases?
   - How many instruction cycles will it take to implement the parity function in each of these cases? (show μcode to support)
   - How many total operation instruction bits in memory are required to describe this operation in each case?

2. Consider your non-branch-enabled and branch-enabled multiplication from the previous assignment. For concrete comparison, use the 4b×4b→8b multiplication which you have already worked out. For each case summarize:

   - The number of bits per instruction.
   - The number of instructions required for your solution.
   - The total number of instruction bits to specify the computation.
   - The total cycles required to perform the computation (same as from last assignment, just summarize here with the rest).

3. Consider your final branching datapath from the previous assignment. Your operations now include 6b for the ALU, a couple of write bits, and some branch bits. So, your operation portion makes up $n \geq 9$ bits. Do you really use all $2^n \geq 512$ operations? with equal regularity? Now, consider limiting yourself to a 4b opcode. That means you only get 16 unique operations.

- Identify the 16 operations you want to keep (you will implement the multiply again, but please select sufficient operation to retain universallity), their encodings, and their expanded n-bit decoding.
- In terms of 2-input gates, how big is the decoder required to expand your 4b opcode into the decoded control bits for the ALU and branching hardware? (you may simply write boolean equations to justify; I don't need to see a circuit diagram. Parenthesize your boolean equations appropriately to show where the gate counts come from.)
- Rewrite your branching multiply from the previous example in terms of this restricted opcode space. Summarize the items from problem 2 for this new implementation (bits/instruction, instructions for solution, total instruction bits, cycles).
- Describe how you could use a memory (or ROM) as the decoder. How large is this memory?

4. Continuing with the branching datapath: Describe qualitatively the effects on instruction bits per cycle and total instruction bits per operation if we change the number of registers appearing in the instruction:

   (a) base case is the existing datapath. *i.e.* 2 operands/instruction with writeback being a separate operation on following cycle: accum = r1 op r2; r1=accum

   (b) more general 2 operands/instruction: r1 = r1 op r2 (allow overwrite in single instruction with operation)

   (c) 1 operand/instruction: accum = r op accum (also operations r1 = accum, accum=0, accum=r1)

   (d) 3 operands/instruction: r1 = r2 op r3

   There are some effects you probably don't know off hand, so identify the key variables and write the inequalities you would use to decide which resulted in the least instruction bits in memory or instruction bits issued (cycles to compute $\times$ instr bits/cycle). State additional assumptions as necessary.

5. Let's say you have an old design which is 70% instruction memory, and you've used techniques like the ones above to reduce the instruction memory size by 35% while keeping other things the same. Assume, for simplicity, technology is continuously improving such that you get a reduction in feature size by a factor of 2 every three years. How many months of technology scaling give the same size reduction as your improved design?

extra Describe other techniques which can be used to decrease issued instruction width and/or total instruction bits to describe a computation.