Virginia Tech 1872

Instructions:

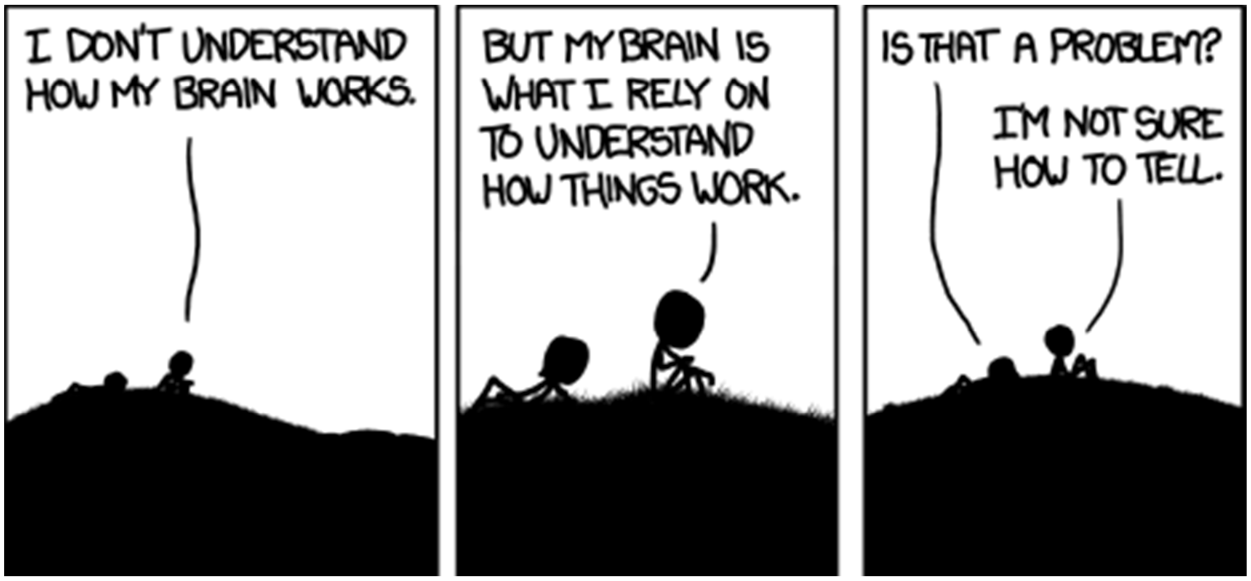
- Print your name in the space provided below.
- This examination is closed book and closed notes, aside from the permitted one-page formula sheet. No calculators or other computing devices may be used. The use of any such device will be interpreted as an indication that you are finished with the test and your test form will be collected immediately.
- Answer each question in the space provided. If you need to continue an answer onto the back of a page, clearly indicate that and label the continuation with the question number.
- If you want partial credit, justify your answers, even when justification is not explicitly required.
- There are 7 questions, some with multiple parts, priced as marked. The maximum score is 100.
- When you have completed the test, sign the pledge at the bottom of this page and turn in the test.
- If you brought a fact sheet to the test, write your name on it and turn it in with the test.
- Note that either failing to return this test, or discussing its content with a student who has not taken it is a violation of the Honor Code.

Do not start the test until instructed to do so!

Name ______________________________
printed

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

____
signed


I DON'T UNDERSTAND HOW MY BRAIN WORKS.
BUT MY BRAIN IS WHAT I RELY ON TO UNDERSTAND HOW THINGS WORK.
IS THAT A PROBLEM?
I'M NOT SURE HOW TO TELL.

xkcd.com

1. [10 points] A processor executes 2 billion instructions, with an average CPI of 2.4, in 4 seconds. What is the clock frequency of this processor? Justify your conclusion.

The processor is capable of executing 2/4 billion instructions/second, with an average of 2.4 cycles/instruction.

Therefore, if we multiply these two values, we obtain the number of cycles/second:

$$\frac{2}{4}\times 10^9 \text{ instructions/second} \square 2.4 \text{ cycles/instruction } = 1.2\times 10^9 \text{ cycles/second} = 1.2 \text{ GHz}$$

2. [12 points] A design team is considering enhancing a machine by adding SNX (social networking extension) hardware to a processor. Instructions that can be run on the SNX hardware, rather than the standard hardware, will execute 6 times faster. Call the percentage of time spent using the new SNX hardware the *percentage of social enhancement*, or PSE.

a) What PSE is needed in order to achieve an overall speedup of 2?

Suppose, without loss of generality, that the time required to execute benchmark code w/o using the SNX hardware is T_{before}, and that x is the amount of time spent executing instructions in that code that can be run on the SNX hardware. Then, Amdahl's Law implies that:

$$T_{after} = T_{before} - x + \frac{x}{6}$$

Now, if we achieve a speedup of 2, T_{after} must be half of T_{before}, so:

$$T_{before} - x + \frac{x}{6} = \frac{T_{before}}{2}$$

It follows that

$$\frac{T_{before}}{2} = \frac{5}{6}x \text{ so that } x = \frac{3}{5}T_{before}$$

Hence, a speedup of 2 will be achieved if 60% of the time was spent on instructions that can be executed on the SNX hardware.

b) Based on your analysis in the first part, what percentage of the run-time is spent executing the SNX hardware if a speedup of 2 is achieved?

We have that $T_{after} = T_{before} / 2$, and from the analysis above that

$$T_{after} = 0.40\times T_{before} + T_{SNX} = 0.40\times 2\times T_{after} + T_{SNX} = 0.8\times T_{after} + T_{SNX}$$

Therefore:

$$T_{SNX} = 0.2\times T_{after}$$

So, after the improvement, we would spend 20% of the time executing the SNX hardware.

3. A processor architect has to choose between two ISA designs, and wants to analyze the potential performance of the two designs on a large suite of software that is likely to be critical to users of the new system.

a) [8 points] The Alpha ISA involves three instruction classes: class Alpha-I instructions will execute in 3 clock cycles, class Alpha-II instructions will execute in 2 clock cycles, and class Alpha-III instructions will execute in 5 clock cycles.

The software suite in question will compile to Alpha machine code that includes 50% Alpha-I instructions, 40% Alpha-II instructions, and 10% Alpha-III instructions.

What would be the average CPI for this software suite on the Alpha ISA? Show your work.

$$\begin{aligned}\text{Average CPI} &= 0.50\times 3 + 0.40\times 2 + 0.10\times 5\\ &= 1.5 + 0.8 + 0.5\\ &= 2.8 \text{ cycles/instruction}\end{aligned}$$

b) [6 points] The Beta ISA involves a different classification of instructions (than the Alpha ISA), and the software suite in question will compile to Beta machine code that has an average CPI of 2.3.

What else, if anything, must the architect know in order to decide whether the Alpha or Beta ISA will offer better performance for her clients? Explain.

The average CPI, by itself, is meaningless.

The architect must also know the length of the clock cycles that can be supported for the two ISAs.

It is entirely possible that the Beta ISA will require a slower clock rate in order to achieve an improvement from 2.8 to 2.3 cycles per instruction. Or not…

4. [24 points] Refer to the single-cycle MIPS32 datapath diagram supplied with the test. Recall that this datapath supports the following instructions: `add`, `sub`, `and`, `or`, `slt`, `lw`, `sw`, `beq` and `j`.

a) For which supported instruction(s) would the data line labeled **4A** be necessary? Explain why.

4A supports supplying an address to the data memory unit.

This is necessary for the `lw` and `sw` instruction, and no others.

b) For which supported instruction(s) would the data line labeled **4B** be necessary? Explain why.

4B supplies the low 6 bits of the current instruction to the ALU control unit.

The only instructions for which those bits are used by the ALU control unit are the R-type instructions (`add`, `sub`, `and`, `or`, `slt`).

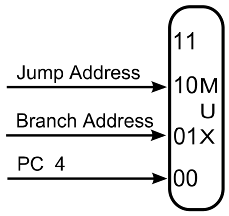
c) The **Shift left 2** unit labeled **4C** is needed in order to execute the `beq` instruction. Explain why. Be complete.

4C is applied to the sign-extended 16-bit immediate from the `beq` instruction, which is then added to the base register as an offset to the target instruction.

The effect of shifting left 2 bits is to multiply by 4; which yields a *word address*, that is an address that is a multiple of 4. Since all MIPS machine instructions are 4 bytes wide, their addresses are always multiples of 4.

Multiples of 4 end in two zero bits; by not storing those in the 16-bit immediate, we effectively have an 18-bit offset for the branch instruction.

5. [14 points] Refer again to the single-cycle MIPS32 datapath diagram supplied with the test. Suppose the two multiplexors labeled **5** were replaced by the single multiplexor shown below (input 11 is not used):



Explain how you would modify the given datapath in order to correctly control this multiplexor. You may not add any new control signals from the **Control** unit.

Note that the Jump Address is passed through iff the Jump signal is 1, and the Branch Address is passed through iff the Branch and Zero signals are both 1. Furthermore, Branch and Jump can never be 1 at the same time.

Furthermore, we need a two-bit selector signal for this multiplexor. Denote that signal by `UL`, where `U` is the high bit and `L` is the low bit of the selector signal.

Then, it's clear that:

`U` = Jump

`L` = Branch & Zero

It's true that you can compute the two bits by performing an addition, but that's extra work and hence less efficient than simply concatenating the bits.

---

6. [6 points] Consider the **Zero** signal emanating from the **ALU** in the single-cycle MIPS32 datapath. In an earlier assignment, you considered for what supported instructions the Zero signal might be set to 1.

The answer to that question implies the need for something else that was included in the datapath design. Identify that datapath element and explain why it is necessary.

The Zero signal only tells us whether the ALU just computed the value 0; in order to whether to fetch the instruction from the branch target address when executing the branch-equals instruction, we must also know whether that's what the current instruction is.

Therefore, we also had to add the Branch signal.

And, we had to add an AND gate to combine the Branch and Zero signals to take the branch appropriately.

The question asked about something that was NEEDED because the Zero line alone was apparently insufficient. Things like the ALU, multiplexor, etc, are needed to execute a beq, but that need is independent of the fact that the Zero line is not enough to make the decision as to whether to branch.

7. [18 points] The MIPS32 instruction set includes the add-immediate (I-format) instruction:

```
addi   $rs, $rt, Imm      # GPR[rt] = GPR[rs] + Imm
```

The given single-cycle datapath can be extended to support the add-immediate instruction by simply modifying the **Control** unit. No additional hardware (outside the **Control** unit) is necessary, not even any additional signals.

Explain how and why the **Control** unit must set the control signals emanating from it in order to execute an add-immediate instruction.

There are nine signals to be considered:

signal	set to	reason
RegDst	0	the destination register is spec'd by bits 20:16 of the addi instruction
Jump	0	we don't want to take the Jump address
Branch	0	we don't want to take the Branch target address
MemRead	X	we don't send a value from memory to the register file anyway
MemtoReg	0	we do send the value computed by the ALU to the register file
MemWrite	0	we don't want to write a value to memory
ALUop	??	whatever it takes to specify an addition operation
ALUSrc	1	the second operand is the sign-extended immediate from the instruction
RegWrite	1	we want to write the result into a register

All of the control signals must be considered here, especially since there's only one that's a don't-care. Saying something like "set the rest as if you were doing an add" was not only likely incorrect, but did not establish that you knew what that meant in detail.