CIS 351 Practice Test 3 Solutions

March 19, 2022

Updated March 19, 2022

1. What is the role of registers on the CPU?

Fast storage for holding commonly-used values. A stored value must be in a register, rather than memory, if we want to perform a computation on it.

2. What is a computer's "word size"? Size of a typical piece of data operated on by the processor

3. How does a computer's word size affect its design?

Determines what size operands are sent to ALU, how many bytes are loaded from memory at once, what size the instructions are, what the maximum size of memory is, what size the registers are, etc.

4. What are the values stored as `here` and `there` in the code below?

```
0x00400000 | addi $s0, $s1, 2
0x00400004 | here: beq $s0, $s2, there
0x00400008 | addi $s0, $s0, 1
0x0040000C | j here
0x00400010 | there: sub $t0, $t0, $t1
```

`here` from the jump instruction is `0000 0100 0000 0000 0000 0000 01`.

`there` from the branch instruction is 2 and would be stored as `0000 0000 0000 0010`.

Note that the labels themselves do not have "values". How a label is stored depends on the particular jump or branch instruction that is under consideration.

5. Review both questions from Homework 4. Make sure you are comfortable assembling or disassembling instructions by hand, not just by using an online tool.

6. Give an example of a change you could make to a microarchitecture that would not affect the implemented architecture (it does not need to be a particularly major or useful change).

One example that we discussed previously was changing the implemenation of the adder inside the ALU. As long as the circuit produces the exact same result for the same input, the change will not change the architecture. For example, a carry-lookahead adder is faster than a ripple-carry adder, but both will produce the same answer in the end.

7. Give an example of an instruction for which `RegDest` is `X` (i.e., "Don't care").

There are several – one option is `beq`.

8. Modify the example MIPS datapath we have been working with to include the jump instruction. You may add additional control wires if needed.

The jump instruction modifies the program counter. Adding this would require

(a) getting the jump target from the instruction bits – this requires understanding pseudodirect addressing

(b) ensuring that the PC uses this information only during a jump instruction, which would require the addition of a mux and a jump control wire, similar to what was done when we added branching

9. Give the result of `0x0123 AND 0x0F04`.

`0000 0001 0000 0000`, which is `0x0100`

10. Give the result of `0x0123 XOR 0x0F04`.

`0000 1110 0010 0111`, which is `0x0E27`

11. Convert the following line of Java code to assembly: `t0 = t1 + t2 + t3 - t4 + t5`

There are many correct answers. Here is one:

```
add $t0, $t1, $t2
add $t0, $t0, $t3
sub $t0, $t0, $t4
add $t0, $t0, $t5
```

12. Convert the following line of Java code to assembly: `t0 = (t1 ^ t2) & (t3 | !t4)`

There are many correct answers. Here is one:

```
xor $t0, $t1, $t2   # computes t1 ^ t2
xori $t5, $t4, -1    # computes !t4
or $t5, $t5, $t3    # computes t3 | !t4
and $t0, $t0, $t5
```

13. Convert the following Java code to assembly. Your answer *must* use `slt` and `beq` or `bne`. Using branching pseudoinstructions, such as `blt`, will result in partial credit.

```
if (t1 - 6 < t2) {
  t0 = t1;
} else {
   t0 = t2 + 4;
}
t1 = t1 + 7
```

There are many correct answers. Here is one:

```
addi $t3, $t1, -6
slt $t4, $t3, $t2
beq $t4, $zero, else
addi $t0, $t1, 0  # does a move
j done
else: addi $t0, $t2, 4
done: addi $t1, $t1, 7
```

Please note the part about needing to use `slt` and `beq`. I have no issue with you using pseudoinstructions for your branch statements in lab, but to test your understanding I may very well limit you in this way on an exam.

For "Describe in common English" questions (which you *will* see on the exam), you need to give me a high-level description of what is happening. Saying something like "Two numbers are added and then divided by 2" will not earn as many points as saying "The average of the two numbers is returned."

14. Describe in common English what the following function does. Hint: It takes three parameters, all integers.

```
mysteryFunction1:
slt $t0, $a0, $a1
slt $t1, $a1, $a2
and $v0, $t0, $t1
jr $ra
```

It returns whether the parameters are in strictly increasing order.

15. Describe in common English what the following function does. Hint: It takes two integer parameters. `sra` stands for "shift right arithmetic". It moves all the bits in the register to the right the specified amount.

```
mysteryFunction2:
add $v0, $a0, $a1
sra $v0, $v0, 1
jr $ra
```

It returns the average of the two numbers.

16. How many distinct opcodes are possible for MIPS instructions? What would change about MIPS insructions if we wanted to allow 100 distinct opcodes? What problems might this cause?

6 bits in the opcode, so 2^6 possible opcodes. If we wanted to allow 100 opcodes, we would need at least 7 bits to represent them. If we added a bit to represent the opcode, we would need to take a bit off somewhere else to keep our instruction to 32 bits.