CS-UY 2214 — Recitation 7

Introduction

Complete the following exercises. Unless otherwise specified, put your answers in a plain text file named `recitation7.txt`. Number your solution to each question. When you finish, submit your file on Gradescope. Then, in order to receive credit, you must ask your TA to check your work. Your work should be completed and checked during the recitation session.

You may consult the E20 manual, which is available on Brightspace.

Problems

1. Give one way that a multiplexer is like an "if" statement. Give one way that a multiplexer is *not* like an "if" statement.

2. Consider the circuit diagram for the single-cycle E20 processor. Why do we need the sign-extend operation from the 7-bit immediate value before it's added to the program counter?

   Which opcode(s) makes use of that component?

3. Consider the circuit diagram for the single-cycle E20 processor. What should be the value of FUNC_{alu} during the `jeq` instruction? Why?

   What should be the value of FUNC_{alu} during the `j` instruction? Why?

4. Examine the single-cycle circuit diagram and its description in the E20 manual. In particular, pay attention to the meaning of the various control signals, and the possible values they can carry.

   For example, here are the control signals that would be enabled by the instruction `add`:

   - FUNC_{alu} — 0. We want the ALU to perform addition.
   - MUX_{alu} — 0. Both ALU inputs come from the registers.
   - MUX_{pc} — 1. This is a non-jumping instruction, so a simple increment suffices.
   - MUX_{rf} — 0. Based on the instruction encoding in the E20 manual, we can see that the source register selectors are in the rA (bits 12-10) and rB (bits 9-7) fields.
   - MUX_{tgt} — 0. The value we write to the register value comes from the ALU.
   - MUX_{dst} — 1. Based on the instruction encding in the E20 manual, we can see that the destination register selector is in the rC (bits 6-4) field.
   - WE_{rf} — 1. We do write to the register file.
   - WE_{dmem} — 0. We do not write to memory.

   For each of the following E20 instructions, indicate the correct value of each of the control lines. If the value of a signal doesn't matter, indicate so.

   (a) `addi`

(b) `jr`

(c) `lw`

5. Consider the following configuration of control lines in a single-cycle E20 processor:

   - FUNC_{alu} — 3
   - MUX_{alu} — 0
   - MUX_{pc} — 3
   - MUX_{rf} — 1
   - MUX_{tgt} — 1
   - MUX_{dst} — 1
   - WE_{rf} — 0
   - WE_{dmem} — 0

   Which E20 instruction does this configuration correspond to? Explain your answer by justifying the setting of each of the control lines.

6. Consider the following configuration of control lines in a single-cycle E20 processor:

   - FUNC_{alu} — 2
   - MUX_{alu} — 0
   - MUX_{pc} — 3
   - MUX_{rf} — 1
   - MUX_{tgt} — 1
   - MUX_{dst} — 2
   - WE_{rf} — 1
   - WE_{dmem} — 0

   Which E20 instruction does this configuration correspond to? Explain your answer by justifying the setting of each of the control lines.

7. Your friend Pat Benatar shows you the following complete E20 assembly program:

```
        movi $1, 32
        add $1, $1, $1
        add $1, $1, $1
        lw $2, it($0)
        add $2, $2, $1
        sw $2, it($0)
    it: addi $3, $0, 42    # the instruction
        halt
```

   Pat complains that although the program is supposed to store the value 42 into register `$3` (at the indicated instruction), it does not actually do that. Can you explain what happens instead? Describe the result and why it happens.