Problem 1: In the MIPS implementation below some wires are labeled with cycle numbers and values that will then be present. For example, \(c_{2:4}\) indicates that at cycle 2 the wire will hold a 4. Other wires are labeled just with cycle numbers, indicating that the wire is used at that cycle. If a value on any labeled wire is changed the code would execute incorrectly. Note that instruction addresses have been provided. [50 pts]

- Finish a program consistent with these labels.
- All register numbers and immediate values can be determined.
- Be sure to fill the two blocks marked Fill In.
- Provide an explanation for the EX-stage fill-in block.

<table>
<thead>
<tr>
<th>Cycle: 0 1 2 3 4 5 6 7 8</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IF</strong> <strong>ID</strong> <strong>EX</strong> <strong>ME</strong> <strong>WB</strong></td>
</tr>
<tr>
<td><strong>0x1000</strong></td>
</tr>
<tr>
<td><strong>Instruction</strong></td>
</tr>
<tr>
<td>(0x1000)</td>
</tr>
<tr>
<td><strong>Field</strong></td>
</tr>
<tr>
<td><strong>IF</strong></td>
</tr>
<tr>
<td><strong>ID</strong></td>
</tr>
<tr>
<td><strong>EX</strong></td>
</tr>
<tr>
<td><strong>ME</strong></td>
</tr>
<tr>
<td><strong>WB</strong></td>
</tr>
</tbody>
</table>

**Fill in cycle number and value.**

- \(c_{2:4}\): \(0x20a2000e = 0010 0000 1010 0010 0000 0000 1110\)
- \(c_{5:0}\): _____

**Fill in and explain:**

- \(c_{1:8}\): 8
- \(c_{2:6}\): 6
- \(c_{3:0}\): Input
- \(c_{5:0}\): Output

Cycle: 0 1 2 3 4 5 6 7 8
Problem 2: Answer the following questions about, or inspired by, ARM.

(a) [10 pts] MIPS lacks a counterpart to the ARM instruction shown below. *Hint: This has nothing to do with MIPS’ movn.*

\[
\text{mov r1, #5} \quad \text{// Move the constant 5 to register r1.}
\]

\[
\text{Explain how r0 makes such a MIPS instruction unnecessary.}
\]

\[
\text{Show how to perform the same operation using MIPS instruction(s).}
\]

(b) [5 pts] The ARM ISA states that the result of executing an instruction like \text{str r15, [r1]} is that either PC+8 or PC+12 is stored in memory, depending on the implementation. (Remember that ARM r15 is an alias for PC.)

\[
\text{What is the benefit of making the result of the store implementation dependent?}
\]

(c) [5 pts] Consider an ISA which stated that the number of branch delay slots could be either zero or one, depending on the implementation.

\[
\text{As an ISA feature, how does this delay-slot implementation dependence compare in practicality to ARM’s store PC implementation dependent behavior?}
\]
Problem 3: In RISC ISAs instructions fixed size, that is, all instructions are the same size. For example, in MIPS, all instructions are 32 bits. The character size in MIPS (and all ISAs mentioned in this test) is 8 bits.

(a) [5 pts] Describe a benefit of having fixed-size instructions. (This answer can be mentioned in the next answer’s explanation.)

☐ Fixed-size instruction benefit for RISC.

(b) [5 pts] In the Itanium VLIW ISA all instructions are 41 bits. Why would 41-bit instructions be difficult or wasteful to implement in a RISC ISA, such as MIPS, but cause no difficulties in VLIW ISA implementations, including Itanium implementations.

☐ Forty-one bit RISC instructions difficult or wasteful because:

☐ Forty-one bit VLIW instructions not wasteful and make sense because:
Problem 4: Answer the following compiler questions.

(a) [10 pts] A company is considering removing a bypass connection in a design of an implementation. Analysis on good test programs shows that performance drops by 5% with the bypass removed (and no other changes).

What can compiler writers do about the performance drop?

(b) [10 pts] Dead-code elimination is a commonly used compiler optimization, while profiling is used less often because it is a multi-step process.

Using an example briefly describe dead-code elimination.

Briefly describe the steps used in profiling.

Which is more likely to result in disappointing results that surprise the programmer? Explain.