Problem 1 (Architecture versus Microarchitecture)
True or false: The following is architecturally visible (exposed by the architecture)?

   (1) Register file entries in a classical RISC pipeline:
   (2) The stack in a stack architecture:
   (3) Pipeline registers:
   (4) Branch-delay/load-delay slots:
   (5) NOPs:
   (6) Pipeline bubbles:
   (7) Condition code/flags:
   (8) Memory address width:
   (9) Instruction/data caches:

Problem 2.1 (Microcoded versus Pipelined)
How does a microcoded implementation differ from a classic RISC pipeline?
Why does microcoding generally facilitate a simpler microarchitecture?

Problem 2.2 (Microprogramming)
Implement a conditional memory-to-memory move instruction in microcode for the single-
bus RISC-V machine described in Handout 1. This instruction has the following format:

\[
\text{CMOV} \text{M} (rd), (rs1), rs2
\]

CMOV performs the following operation: If the value in \( rs2 \) is true (non-zero), then the
memory word loaded from the address specified by \( rs1 \) is stored to the address in \( rd \).

\[
\text{if } R[rs2] \neq 0 \text{ then} \\
M[R[rd]] \leftarrow M[R[rs1]]
\text{end if}
\]

Fill in the following table with the microinstructions and control signals. Optimize your
microprogram to minimize the number of cycles and to set entries to don’t-cares ("*")
wherever possible.

Date: January 31, 2020.
<table>
<thead>
<tr>
<th>State</th>
<th>PseudoCode</th>
<th>IdR</th>
<th>Reg Sel</th>
<th>Reg Wr</th>
<th>en Reg</th>
<th>IdA</th>
<th>IdB</th>
<th>ALUOp</th>
<th>en ALU</th>
<th>Id MA</th>
<th>Mem Wr</th>
<th>en Mem</th>
<th>Imm Sel</th>
<th>en Imm</th>
<th>μBr</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>FETCH0:</td>
<td>MA ← PC; A ← PC</td>
<td>*</td>
<td>PC</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>0</td>
<td>1</td>
<td>*</td>
<td>0</td>
<td>0</td>
<td>N</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>IR ← Mem</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>0</td>
<td>0</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>*</td>
<td>0</td>
<td>S</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>PC ← A+4</td>
<td>0</td>
<td>PC</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>*</td>
<td>INC_A_4</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>0</td>
<td>*</td>
<td>0</td>
<td>D</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>. . .</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOP0: microbranch back to FETCH0</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>0</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>0</td>
<td>*</td>
<td>*</td>
<td>0</td>
<td>*</td>
<td>0</td>
<td>J</td>
<td>FETCH0</td>
</tr>
<tr>
<td>CMOVM:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>