Problems 1-4 refer to a proposed MIPS instruction `lwu` (load word - update) which implements *update addressing* – an addressing mode that is used in the PowerPC (see p. 177). The assembly language form of `lwu` and its register transfers are shown below:

**New Instruction**

`lwu rt, immed(rs)`

**Equivalent MIPS Instructions**

`lw rt, immed(rs);`
`addi rs, rs, immed`

**Register Transfers**

`address <- Reg[rs] + sign_extend(immed);`
`Reg[rt] <- MEM[address];`
`Reg[rs] <- address;`

1. **Multicycle Processor Design**  

20 Points

Modify the multicycle processor design to efficiently implement the `lwu` instruction. Mark changes on the state diagram below and the datapath diagram on the next page.
3. Pipelined Processor Design 20 Points

Modify the pipelined processor datapath and control to implement the \texttt{lwu} instruction. Note that in a pipelined implementation all register updates should take place during the WB stage. To make this possible, the register file has been extended with a second “write port” so that it can write two registers at the same time.

Mark any changes to the datapath on the diagram on the next page. In addition, show all control outputs in the table below:

<table>
<thead>
<tr>
<th>Instr.</th>
<th>EX Stage Control Lines</th>
<th>MEM Stage Control Lines</th>
<th>WB Stage Control Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reg Dst</td>
<td>ALU Op1</td>
<td>ALU Op0</td>
</tr>
<tr>
<td>\texttt{lwu}</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(insert pipeline datapath diagram here)
4. Data Hazards & Forwarding  

The following sequence of MIPS instructions includes the new \texttt{lwu} instruction. Assume that this sequence is executing on the modified pipeline design from Problem 3, but that this design is altered to perform forwarding as in Fig. 6.40 on p. 484.

\begin{verbatim}
\texttt{lwu} \$4, 200(\$1)
\texttt{add} \$6, \$1, \$7
\texttt{sub} \$5, \$6, \$4
\end{verbatim}

(a) Circle any data dependencies which exist between these instructions.

(b) Note that the \texttt{lwu} instruction writes the \texttt{rs} register as well as the \texttt{rt} register. Can the data dependencies on the \texttt{rs} register be resolved by forwarding alone, or will stalling be necessary? Why or why not?

(c) Fill in the multicyle diagram shown below to show the execution of the instruction sequence, including stalls (if any). Shade active stages and show forwarding.
5. Pipelined Processor Timing 10 Points

This problem refers to the pipelined datapath used in Problem 3. Assume that the pipelined datapath components have the same delay characteristics as the single-cycle components described on page 373 of the book – ALU and memory have a 2ns delay, register file read and write each have a 1ns delay.

(a) Assume that all other components have no delay. What is the minimum clock period at which this design can operate properly?

(b) Now assume that in addition to the delays given above, the delay of multiplexers is 0.1ns. What is the minimum clock period at which this design can operate properly? What stages limit the execution time?

6. Short Answers 10 Points

Provide a short answer for each of the following questions:

(a) When would a compiler be able to use the lwu instruction to increase the speed of a program?

(b) Why do RISC architectures use fixed-width instructions?

(c) What are the steps required to add two floating point numbers?

(d) What is the motivation for out-of-order execution in dynamic pipelining?

(e) When does a page fault occur?
7. **Cache Memories**  

The diagram on the next page shows a cache memory design which contains 8 blocks of four words each. Note that on a cache miss all four words in a block are replaced at the same time.

(a) How many bits will there be in the “Block Offset” field of the address?

(b) How many bits will there be in the “Tag” field of each address?

(c) How many bits of storage will be required for this cache memory?
(d) Given the word references below, mark each reference as a hit or miss and show the cache contents in the table below.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Hit/Miss</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Block 0</td>
<td>0</td>
</tr>
<tr>
<td>Block 1</td>
<td>4</td>
</tr>
<tr>
<td>Block 2</td>
<td>8</td>
</tr>
<tr>
<td>Block 3</td>
<td>12</td>
</tr>
<tr>
<td>Block 4</td>
<td>16</td>
</tr>
<tr>
<td>Block 5</td>
<td>20</td>
</tr>
<tr>
<td>Block 6</td>
<td>24</td>
</tr>
<tr>
<td>Block 7</td>
<td>28</td>
</tr>
</tbody>
</table>