Problems 1-3 refer to the MIPS instruction `jalr` (jump and link register) which is described in Appendix A of the book (see p. A-65). The assembly language form of `jalr` and its register transfers are shown below:

**Assembly language:**

`jalr rs, rd`

**Register Transfers**

- `Reg[rd] <- PC + 4;`
- `PC <- Reg[rs];`

1. **Multicycle Processor Design**  
   20 Points

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

Modify the pipelined processor datapath and control to implement the **jalr** instruction assuming that the PC is changed during the MEM stage and the **rd** register is stored during the WB stage.

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 ALU Op1 ALU Op0 ALU Src</td>
<td>Branch Mem Read Mem Write</td>
<td>Reg Write Memto Reg</td>
</tr>
<tr>
<td>jalr</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3. Data and Control Hazards 15 Points

The following sequence of MIPS instructions includes the `jalr` instruction. Assume that this sequence is executing on the modified pipeline design from Problem 3, but that the design is altered to perform forwarding, stalls, and flushing as required to deal with data and control hazards.

```
xor $5, $7, $8
jalr $5, $6  # Assume rs ($5) contains address of L
add $7, $3, $4
sub $8, $9, $10
and $12, $8, $14
...
L: lw $11, 200($4)
```

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

(b) Mark any of the above instructions that will be flushed due to control hazards.

(c) Fill in the multicycle diagram shown below to show the execution of the instruction sequence, including stalls, forwarding, and flushes (if any). Shade active stages.
4. Processor Timing  

Assume that the datapath components in the designs of Problems 1 and 2 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. However, further measurements indicate that the logic in the control unit has a non-negligible delay of 0.5ns.

(a) What impact does this delay have on the clock period of the multicycle design?

(b) What impact does this delay have on the clock period of the pipelined design?

5. Logic and Arithmetic  

The MMX (Multimedia Extension) instructions supported by recent Pentium Processors use a modified 64-bit ALU to perform eight 8-bit arithmetic operations (e.g. addition or subtraction) simultaneously.

(a) Assuming that a simple ripple-carry ALU is used, describe how the design of the ALU must be modified to support these operations. You can draw a diagram if it helps you explain how to do this, but it is not necessary if you can explain it clearly in words.

(b) What impact do these modifications have on the delay of the ALU?
6. **Short Answers**  

Provide a short answer for each of the following questions:

(a) The Intel 8087 floating point arithmetic instructions use a stack to store operands. As part of the SSE2 extensions in the Pentium 4, Intel has provided a new set of floating point instructions which access operands in new floating point registers instead of the stack. Why would Intel do this?

(b) What security flaw is usually exploited by Internet worms to attack systems over a network? How can such attacks be prevented?

(c) What is the biggest advantage of the Multicycle style of processor implementation?

(d) Why is virtual memory used in virtually all modern computer systems?

(e) Can a processor which supports out-of-order execution continue to execute instructions after a miss on the data cache occurs? Why or why not?
7. Cache Memories 15 Points

The diagram below shows a direct-mapped cache memory design which contains 8 blocks. Each block stores one 32-bit word plus tag and valid bit.

(a) How many bits will there be in the “Index” 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) Assume that the cache shown above is initially empty (i.e. a cold cache). Fill in the chart below to show the hits and misses encountered for each 32-bit word reference. Write the cache contents in the diagram shown above.

<table>
<thead>
<tr>
<th>Reference</th>
<th>1</th>
<th>4</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>7</th>
<th>1</th>
<th>3</th>
<th>4</th>
<th>9</th>
<th>10</th>
<th>35</th>
<th>4</th>
<th>1</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hit/Miss</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>