Problem 1a

Consider two different implementations, M1 and M2, of the same instruction set. There are three classes of instructions (A, B, and C) in the instruction set. For a given program, the average number of cycles for each instruction class is shown below. The table also shows how many instructions of a given class are in the program, as a percentage. E.g., if there are 100 instructions in total, there are 60 Class A instructions. Calculate the average CPI (Clocks Cycles per Instruction) for computer M1 and M2.

<table>
<thead>
<tr>
<th>Instruction Class</th>
<th>Computer M1 (Cycles Per Instruction Class)</th>
<th>Computer M2 (Cycles Per Instruction Class)</th>
<th>Percentage of total instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>2</td>
<td>60%</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>3</td>
<td>30%</td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>4</td>
<td>10%</td>
</tr>
</tbody>
</table>
Problem 1b

- Computer A has an overall CPI of 1.3 and can be run at a clock rate of 600MHz. Computer B has a CPI of 2.5 and can be run at a clock rate of 750 Mhz. We have a particular program we wish to run. When compiled for computer A, this program has exactly 100,000 instructions. How many instructions would the program need to have when compiled for Computer B, in order for the two computers to have exactly the same execution time for this program?
Problem 2a: Consider the instruction: AND Rd, Rt, Rs. What are the control signals generated by the control unit in this figure?
ANSWER:

- Regwrite asserted to write
- Mux (before ALUs) signaled to read from registers and not immediate
- Mux (before registers) signaled to use ALU and not data memory
- ALU is signaled to perform AND
- Branch not asserted
- Memwrite not asserted
- Memread not asserted
Problem 2b

Increment by 4 for next instruction
**Problem:** If the only thing we need to do in a processor is fetch consecutive instructions (see figure), what would the cycle time be? Consider that the logic blocks have the following latencies. Other units have negligible latencies.

<table>
<thead>
<tr>
<th>Operation</th>
<th>I-Mem</th>
<th>ADD</th>
<th>Mux</th>
<th>ALU</th>
<th>Regs</th>
<th>D-Mem</th>
<th>Sign-Extend</th>
<th>Shift-left-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (ps)</td>
<td>200</td>
<td>70</td>
<td>20</td>
<td>90</td>
<td>90</td>
<td>250</td>
<td>15</td>
<td>10</td>
</tr>
</tbody>
</table>
- ANSWER:
- 200ps because
  - I-Mem has larger latency than the Add unit
  - I-Mem and Add are in parallel paths
Problem 2 c

Consider the datapath shown in Figure. Assume that the processor has only one type of instruction: the unconditional jump instruction. Also, assume that there is NO pipelining. Consider that the logic blocks have the latencies as shown in Table 1. Other units have negligible latencies.

What would be the cycle-time for this datapath?
Assembly:
  j exit

Note: no registers!
Recall

- Take the 16 bit offset and add it to the address of next instruction following the branch instruction to obtain the branch target address.

- Add the value in program counter (PC) and the 16 bit offset:
  - need an ADDer, no need for ALU because we do not add to register values.
  - Sign extend the 16-bit offset to make it 32 bit like the size of the PC.
<table>
<thead>
<tr>
<th></th>
<th>I-Mem</th>
<th>ADD</th>
<th>Mux</th>
<th>ALU</th>
<th>Regs</th>
<th>D-Mem</th>
<th>Sign-Extend</th>
<th>Shift-left-2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>200ps</td>
<td>70ps</td>
<td>20ps</td>
<td>90ps</td>
<td>90ps</td>
<td>250ps</td>
<td>15ps</td>
<td>10ps</td>
</tr>
</tbody>
</table>
Recall

- Before adding,
  - Shift left 2 places (word displacement)
  - Add to PC + 4
    - Already calculated by instruction fetch
In parallel to instruction memory, so no need to add.

Unconditional jump does not need any comparison of values, so registers are not used.

Not used in this instruction.
Problem 3

In this question, assume that all branches are perfectly predicted (this eliminates all control hazards). Assume, we have only one memory (for both instructions and data), and there might be a structural hazard. To resolve this, the pipeline must be stalled in some cycles. Show the pipelined execution and the number of cycles.

- SW R2, 0(R3)
- OR R1, R2, R3
- BEQ R2, R0, Label (Assume R2== R0)
- OR R2, R2, R0
- Label : ADD R1, R4, R3
SW  R2,  0(R3)
OR  R1,  R2,  R3
BEQ  R2,  R0,  Label  (Assume  R2==  R0)
OR  R2,  R2,  R0
Label:  ADD  R1,  R4,  R3

First, note the effect of the branch instruction
Recall The 5 Stages

IF: Instruction fetch
ID: Instruction decode/ register file read
EX: Execute/ address calculation
MEM: Memory access
WB: Write back
• Draw the five stages for each instruction, while checking for structural hazard, and count 9 cycles as the result!

• Note that the question mentions that branch prediction is perfect and there are no control hazards!

```
SW R2,0(R3)      IF ID  EX   MEM WB
OR R1,R2,R3      IF ED  EX   MEM WB
BLQ R2,K0,Lbl    IF ID  EX   MEM WB
ADD R1,R4,R3     *** IF ID  EX   MEM WB
```
Problem 4

SW  R16,  12(R6)
LW  R16,  8(R6)
BEQ R5,  R4,  Label (Assume R5! = R4, i.e, the branch condition is false)
ADD R5,  R1,  R4
SLT R5,  R15, R4

First, note the branch condition. All 5 instructions will be executed.
Problem 4

SW R16, 12(R6)
LW R16, 8(R6)
BEQ R5, R4, Label (Assume R5! = R4, i.e., the branch condition is false)
ADD R5, R1, R4
SLT R5, R15, R4
Problem 5

<table>
<thead>
<tr>
<th>IF</th>
<th>ID</th>
<th>EX</th>
<th>MEM</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>250ps</td>
<td>350ps</td>
<td>150ps</td>
<td>300ps</td>
<td>200ps</td>
</tr>
</tbody>
</table>

The individual stages of the datapath have the latencies as shown above.

1. Assume a non-pipelined processor. What is the cycle time?

2. Now, assume a pipelined processor. What is the cycle time?
Problem 5

<table>
<thead>
<tr>
<th>IF</th>
<th>ID</th>
<th>EX</th>
<th>MEM</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>250ps</td>
<td>350ps</td>
<td>150ps</td>
<td>300ps</td>
<td>200ps</td>
</tr>
</tbody>
</table>

The individual stages of the datapath have the latencies as shown above.

1. Assume a non-pipelined processor. What is the cycle time?
   Add them up to get the cycle time!

2. Now, assume a pipelined processor. What is the cycle time?
   The largest stage determines the cycle time.
Pipelining = Overlap the stages

- 200 ps latency
- 100 ps latency
Clock Cycle

Single-cycle (T_c = 800ps)

Program execution order (in instructions)

```plaintext
lw $1, 100($0)
lw $2, 200($0)
lw $3, 300($0)
```

Pipelined (T_c = 200ps)

Program execution order (in instructions)

```plaintext
lw $1, 100($0)
lw $2, 200($0)
lw $3, 300($0)
```
Even if some stages take only 100ps instead of 200ps, the pipelined execution clock cycle must have the worst case clock cycle time of 200ps.
Good Luck!!