1. General questions. (20%) 

1.1. (5%) What causes control hazards in a pipelined machine? Other than stalling the pipeline, give one technique that can effectively bypass the control hazard.

(1) Control flow instructions, i.e. branch and jump instructions
(2) branch prediction, or using delay slots in the ISA definition.

1.2. (15%) Illustrate the following dependency scenarios using MIPS-like assembly instruction sequence. (i) True dependency (or Read-After-Write dependency); (ii) Output dependency (or Write-After-Write dependency); (iii) Anti dependency (or Write-After-Read dependency). Please indicate (using arrow) to indicate each category of the dependencies.

Register $5 contains the true, output and anti-dependencies in (i), (ii), (iii) below respectively.

(i) 
add $5, $6, $7 
add $8, $5, $6

(ii) 
add $5, $6, $7 
add $5, $9, $10

(iii) 
add $6, $5, $7 
add $5, $8, $9
2. Multi-cycle vs. pipelined machine. (20%) Given 3 different instruction types --- A, B and C. Each type-A, B, and C instruction takes 30ns, 20ns and 40ns to complete, respectively. An assembly program is written with 20 type-A, 30 type-B and 50 type-C instructions.

2.1. (5%) Assume a single-issue multi-cycle datapath, how much time (in nano-seconds) is required to complete the execution of this program?

$$20 \times 30 + 30 \times 20 + 50 \times 40 = 3200\text{ns}$$ (with a cycle time of 10ns)

2.2. (15%) Now let us pipeline these instructions based on a cycle time of 10ns. To pipeline these instructions equally and ideally using this cycle time, (1) what is the latency of each instruction (in cycles)? (2) how much time (in nano-seconds) is required for this pipelined machine? (Assume the program does not encounter whatsoever pipeline hazards during the execution.) (3) what is the speedup over the multi-cycle datapath in 2.1?

(1) Max(30ns, 20ns, 40ns) = 40ns
40ns/10ns = 4 cycles = the latency of each instruction

(2) Total number of instructions = 20+30+50 = 100
cc1 cc2 cc3 cc4 cc5 cc6 ..........
I1 +-------------------+
I2 +-------------------+
I3 +-------------------+
......
I99 +-------------------+
I100 +-------------------+
cc100 101 102 103

It takes 103*10ns = 1030ns to finish for the pipelined version

(3) Speedup = 3200/1030 = 3.11x
3. Multi-cycle data path (30%) Given the following multi-cycle datapath and table indicating the corresponding operation in each phase of a multi-cycle datapath. For the instruction sequence below, what values need to be asserted to which control signals (red color except ALUop) at (1) the very first cycle? (2) the 9-th cycle? (3) the 19-th cycle? (Assume there is no delay slot. Instruction encodings are provided in next page.)

<table>
<thead>
<tr>
<th>stage</th>
<th>R-type inst</th>
<th>Memory inst.</th>
<th>Branch inst.</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF</td>
<td>(1) Fetch instruction into instruction register; (2) PC+=4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>(1) Read both register operands; (2) calculate branch target address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EX</td>
<td>ALU operation</td>
<td>Generate address</td>
<td>Test and branch</td>
</tr>
<tr>
<td>MEM</td>
<td>Write to destination register</td>
<td>Load from and store to memory</td>
<td></td>
</tr>
<tr>
<td>WB</td>
<td>Write to destination register</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
sub $7, $4, $4
lw $4, 8($10)
beq $7, 0, L1
add $5, $4, $0
lw $10, $5, $5
L1:
lw $9, 16($10)
add $8, $9, $4
```
(1) The 1st cycle: sub $7, $4, $4 in IF stage
   To perform Fetch instruction into IR
   IRwrite = 1, IorD = 0
   MemRead = 1, MemWrite = 0

   To perform PC+=4
   PCWrite = 1
   ALUSrcA = 0, ALUSrcB = 1
   PCsource = 0

(2) The 9th cycle: lw $4, 8($10) in WB stage
   To perform Write to destination register
   RegWrite = 1
   RegDst = 0
   Memtoreg = 1

(3) The 19th cycle: add $8, $9, $4 in ID stage
   Note that the beq instruction is taken because $7 is
   cleared to 0 by the first sub instruction.
   To perform Read both register operands require no control
   signal (except the clock)

   To perform calculate branch target address
   ALUSrcA = 0
   ALUSrcB = 3 (11 to the lower Mux)
4. Pipeline Forwarding. (30%) Due to data hazards, the processor requires a forwarding logic unit to handle true dependency in the following code to get rid of pipeline bubbles. The pipeline shown below is simplified to the relevant part for forwarding. Please draw the necessary signals (1) going into the forwarding unit and (2) generated by the forwarding unit. Also draw the data signals that (3) go into the muxes of the ALU. Please (4) label all your signals with the correct data values (for only those used) based on the given code. (Assume registers are preloaded with the value equal to its register ID number. Also assume that the each memory byte location is preloaded with the value equal to its lowest byte address.)

\[
\begin{align*}
0x10010000: & \quad \text{lw} \quad \$13, \ 8(\$9) \\
& \quad \text{addi} \quad \$8, \ \$7, \ -3 \\
& \quad \text{sub} \quad \$11, \ \$8, \ \$13 \\
& \quad \text{sw} \quad \$15, \ 12(\$9) \\
& \quad \text{add} \quad \$16, \ \$8, \ \$11
\end{align*}
\]