CLOSED BOOK  Work each problem in the space provided on its sheet. Be sure the work you present is clear so I can understand what you have done. You may use one 3'' by 5'' card of notes. No other aids, animate or inanimate, are permitted. Please do your own work.

**Problem 1** [25 points] - The code below is to be run on the pipelined datapath shown below. Assume the datapath has hardware support for forwarding and the branch operation is performed in the Decode stage. Also, a register can be written and read during the same clock cycle.

Show the state of the datapath for the following code sequence when the first add instruction is in the WB stage by filling the slots below with the values for the indicated lines. If you do not know the value of the signal, use the notation SX to indicate the contents of register X and M(SX) to indicate the contents of memory location X. Assume the “zero” input to all MUXs is the upper input. **Be sure to write the instruction above each stage.**

**PC**
- A00h  li $1, 8
- A004h  li $2, 5
- A008h  li $3, 1
- A00Ch  Loop: add $2, $3, $1
- A010h  sw $3, 10($2)
- A014h  lw $4, 10($2)
- A018h  sub $1, $4, $3
- A01Ch  bne $1, $4, $3

**Answer:**
- A. A018h  A018h
- B. 0
- C. $2 = 9  sub
- D. Rewrite = 1
- E. $2 = 9
- F. $3 = 1
Problem 2 [25 points] - Suppose each stage of a pipeline takes the following times:

<table>
<thead>
<tr>
<th>IF</th>
<th>ID</th>
<th>EX</th>
<th>MEM</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1ns</td>
<td>1ns</td>
<td>1ns</td>
<td>3ns</td>
<td>1ns</td>
</tr>
</tbody>
</table>

a. How often can a new instruction be fetched in this pipeline?

\[ 3\text{ns} \]

A design team decides to split the MEM stage into two stages. The times for the new pipeline are:

<table>
<thead>
<tr>
<th>IF</th>
<th>ID</th>
<th>EX</th>
<th>MEM1</th>
<th>MEM2</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1ns</td>
<td>1ns</td>
<td>1ns</td>
<td>1.5ns</td>
<td>1.5ns</td>
<td>1ns</td>
</tr>
</tbody>
</table>

b. How often can an instruction be fetched in this new pipeline?

\[ 1.5\text{ns} \]

c. Assume the pipeline above has data forwarding from both MEM and WB stages to the EX stage, but does not stall for data hazards. Therefore, you must insert \texttt{nop} instructions explicitly into your code so that the loop executes correctly on the new pipeline. Also assume that there is no forwarding to the ID stage. (To save space, you may designate multiple \texttt{nop}s with "nop N" notation, where N is an integer)

\[ \text{li } \texttt{st0, 0} \quad \text{IF ID EX m1 w2 WB} \]
\[ \text{li } \texttt{st1, 20} \quad \text{IF ID EX m1 w2 WB} \]
\[ \text{lw } \texttt{st2, 100(st0)} \quad \text{IF ID EX m1 w2 WB} \]
\[ \text{addi } \texttt{st2, st2, 10} \]
\[ \text{sw } \texttt{st2, 100(st0)} \]
\[ \text{addi } \texttt{st0, st0, 4} \]
\[ \text{bne } \texttt{st0, st1, Loop} \]
\[ \text{Branch done in } \text{Ex stage} \]

\d. For what is the 40000040 input to the MUX to the left of the IF/ID pipeline register used on the datapath on page 1?
Problem 3 [25 points] - Short Answer.

a) Describe how a Combined Predictor works.
   It uses 2 predictors and notes which one is works best.
   It uses the best predictor.

b) Why does a Combined Predictor get better performance than non-combined predictors?
   It can pick the best of the non-combined predictors.

c) Given the pipelined MIPS implementation, a friend suggest adding dynamic branch prediction to speed things up. Is this a good idea? Explain.
   No. There is only 1 delay slot, so the extra hardware won't help.

d) According to Keith Diefendorff in *PC Processor Microarchitecture*, what is the biggest performance obstacle? Why?
   Memory speed.

e) For gshare, which of these stores only the global history?
   a. GR
   b. PC
   c. GR@PC
   d. Counts table

f) For gshare, which of these stores the local history, but not the global?
   a. GR
   b. PC
   c. GR@PC
   d. Counts table
Problem 4 [25 points] - The figure below shows three different forwarding paths labeled A, B, and C.

a. On the datapath on page 1, highlight the hardware and the signal lines that are used for the forwarding path A.

b. What is necessary for forwarding path C to work?

Write Reg half of a cycle
Read Reg to second half of cycle

c. Below is some control logic for handling the forwarding. Give a code sequence that will satisfy the forward logic below and cause the data to be forwarded to the first input of the ALU.

If (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0) and (EX/MEM.RegisterRd ≠ ID/EX.RegisterRs) and (MEM/WB.RegisterRd = ID/EX.RegisterRs)) forwardA = 01

\[
\begin{align*}
\text{add } & s1, s1, s1 \\
\text{sub } & s1, y, y \\
\text{add } & z1, s1, s1
\end{align*}
\]

\[
\begin{align*}
\text{add } & s1, x, x \\
\text{sub } & s1, y, y \\
\text{add } & s1, x, s1
\end{align*}
\]

\[
\begin{align*}
\text{add } & x, s1, x \\
\text{sub } & s1, y, y \\
\text{add } & x, s1, s1
\end{align*}
\]

d. In the forwarding control logic above, why is the line “and (EX/MEM.RegisterRd ≠ ID/EX.RegisterRs)” needed?

So this \( \rightarrow \) add \( s1, x, x \) will work right, i.e. sub is forwarded, not first add.