

Complete the following problems. Be sure to show your work for partial credit.

1. Answer the following questions regarding pipelined execution of this instruction sequence:

```
lw $1,40($6)
add $6,$2,$2
sw $6,50($1)
```

(a) Indicate dependences and their type.

There are two data dependencies: on \$6 between the sw and the add, and on \$1 between the sw and the lw.

(b) Assume there is no forwarding in this pipelined processor. Indicate hazards and add nop instructions to eliminate them.

| Instr | CC1 | CC2 | CC3 | CC4 | CC5 | CC6 | CC7 | CC8 | CC9 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| lw | F | D | X | M | W | | | | |
| add | | F | D | X | M | W | | | |
| nop | | | - | - | - | - | - | | |
| nop | | | | - | - | - | - | - | |
| sw | | | | | F | D | X | M | W |

(c) Assume there is full forwarding. Indicate hazards and add nop instructions to eliminate them.

| Instr | CC1 | CC2 | CC3 | CC4 | CC5 | CC6 | CC7 |
|-------|-----|-----|-----|-------------------------|-----|-----|-----|
| lw | F | D | X | M (\rightarrow sw X) | W | | |
| add | | F | D | X (\rightarrow sw X) | M | W | |
| sw | | | F | D | X | M | W |

(d) Assuming the following clock cycle times,

$ClockPeriod_{without-forwarding} = 300ps,$

$ClockPeriod_{with-full-forwarding} = 400ps,$

$ClockPeriod_{with-alu-alu-forwarding-only} = 360ps$

What is the total execution time of this instruction sequence without forwarding and with full forwarding? What is the speedup achieved by adding full forwarding to a pipeline that had no forwarding?

with no forwarding: $9CC \times 300ps = 2700ps$

with full forwarding: $7CC \times 400ps = 2800ps$

speedup = .96 (really a slowdown)

(e) Add nop instructions to this code to eliminate hazards if there is ALU-ALU forwarding only (no forwarding from the WB stage, i.e., results of the MEM stage, to the EX stage).

| Instr | CC1 | CC2 | CC3 | CC4 | CC5 | CC6 | CC7 | CC8 | CC9 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| lw | F | D | X | M | W | | | | |
| add | | F | D | X | M | W | | | |
| sw | | | F | D | D | D | X | M | W |

(can't get \$1 from lw-W)

(can't get \$6 from add-W)

- (f) What is the total execution time of this instruction sequence with only ALU-ALU forwarding?
 What is the speedup over a no-forwarding pipeline?
 with no forwarding: $9CC \times 300ps = 2700ps$
 with ALU-ALU forwarding only: $9CC \times 360ps = 3240ps$
 $speedup = .83$ (again, really a slowdown)

2. Assume that the instructions executed by a pipelined processor are broken down as follows:

| | |
|-----|-----|
| add | 50% |
| beq | 25% |
| lw | 15% |
| sw | 10% |

- (a) Assuming there are no stalls and that 60% of all conditional branches are taken, in what percentage of clock cycles does the branch adder in the EX stage generate a value that is actually used?
 Because there are no stalls, you can assume that there is always an instruction in the execute stage. 25% of the time it will be a **beq** ($p(X=beq)$). When it is a **beq**, 60% of the time the computed branch target will be used ($p(\text{beq-taken})$).

$$\begin{aligned}
 p(\text{X-target-used}) &= p(\text{X=beq}) \times p(\text{beq-taken}) \\
 &= 0.25 \times 0.6 \\
 &= 0.15
 \end{aligned}$$

- (b) Assuming there are no stalls, how often (as a percentage of all cycles) do we actually need to use all three register ports (two reads and a write) in the same cycle?
 The pipeline will use all three register ports when there is an instruction that reads from two registers in the decode stage at the same time there is an instruction that writes to the register file in the write-back stage.

$$\begin{aligned}
 p(\text{three-ports-used}) &= p(\text{D=2-reader}) \times p(\text{W=writer}) \\
 &= (p(\text{D=add}) + p(\text{D=beq}) + p(\text{D=sw})) \times (p(\text{W=add}) + p(\text{W=sw})) \\
 &= (0.5 + 0.25 + 0.10) \times (0.5 + 0.15) \\
 &= 0.5525
 \end{aligned}$$

- (c) Assuming there are no stalls, how often (as a percentage of all cycles) do we use the data memory?

$$\begin{aligned}
 p(\text{data-mem-used}) &= p(\text{M=lw}) + p(\text{M=sw}) \\
 &= 0.15 + 0.10 \\
 &= 0.25
 \end{aligned}$$