# Solutions for Homework 5

**CIS 501, Spring 1998**
**February 25, 1998**

1. Consider a pipeline where the total work to be done would take 40ns. If the overhead for each pipeline stage is 1 ns, plot the speedup of the pipelined machine vs the number of stages, upto 20 stages. If the overhead is fixed, what could be the highest speedup achievable?

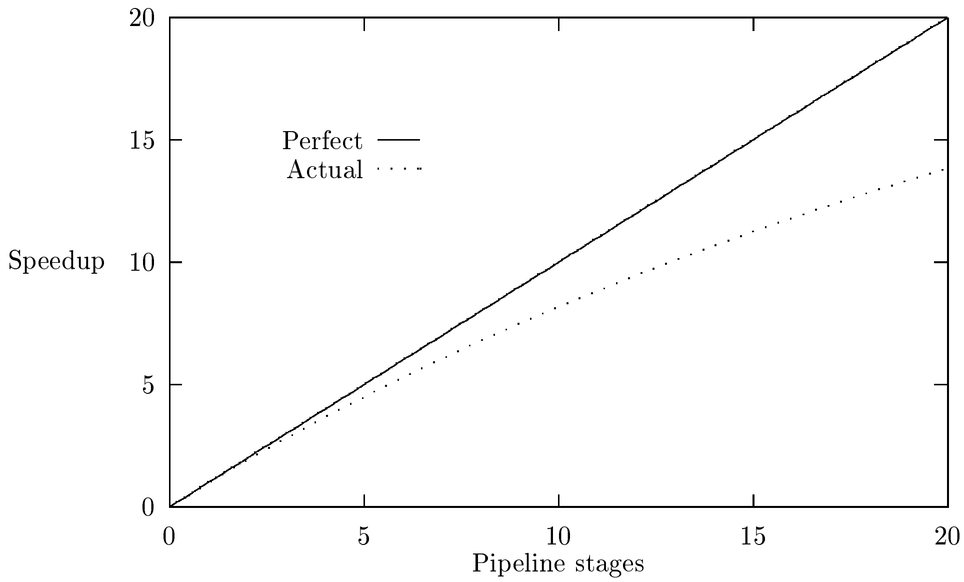
   The speedup due to pipelining for an $n$ stage pipeline is:

$$\begin{aligned}\text{Speedup from pipelining} &= \frac{\text{Average instruction time unpipelined}}{\text{Average instruction time pipelined}} \\ &= \frac{40 \text{ ns}}{\frac{40 \text{ ns}}{n} + 1 \text{ ns}}.\end{aligned}$$

   For a "perfect" pipeline, we incur no overhead, so the speedup is:

$$\begin{aligned}\text{Speedup from pipelining} &= \frac{\text{Average instruction time unpipelined}}{\text{Average instruction time pipelined}} \\ &= \frac{40 \text{ ns}}{\frac{40 \text{ ns}}{n}} = n.\end{aligned}$$

   The graph looks like this:




2. Suppose that 20% of all instrucitnos are conditional branches, while 5% are unconditional (jumps and calls). Also assume that 60% of the conditional branches are taken. Consider a four stage pipeline. All unconditional branches get resolved by the end of the second stage, while conditional branches take one more cycle. Assume that the first stage can be done independent of branch resolution. Ignoring all other stalls, how much faster would the machine be without branch hazards?

   We will analyze the timing diagrams for unconditional branches, conditional branches that are taken, and conditional branches that are not taken. We will assume that the pipeline uses a predict untaken scheme for branches. However, since only the first stage can be "thrown away", we still will get one stall in the untaken case (see below).

(a) **Jumps** Here we get 1 stall per unconditional branch:

| | clock cycle | | | | | |
|---|---|---|---|---|---|---|
| Instruction | 1 | 2 | 3 | 4 | 5 | 6 |
| Jump | S1 | S2 | S3 | S4 | | |
| Jump successor | | - | S1 | S2 | S3 | S4 |

(b) **Taken branch** Here we get 2 stall cycles:

| | clock cycle | | | | | | |
|---|---|---|---|---|---|---|---|
| Instruction | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Taken branch | S1 | S2 | S3 | S4 | | | |
| branch + 1 | | S1 | - | flush | | | |
| Branch successor | | | - | S1 | S2 | S3 | S4 |

(c) **Untaken branch** Here we get just 1 stall cycle:

| | clock cycle | | | | | |
|---|---|---|---|---|---|---|
| Instruction | 1 | 2 | 3 | 4 | 5 | 6 |
| Jump | S1 | S2 | S3 | S4 | | |
| Jump successor | | S1 | - | S2 | S3 | S4 |

Assume a base CPI of 1 without branch hazards. Then the speedup from removing the hazards is:

$$\text{Speedup} = \frac{\text{CPI}_{\text{with}}}{\text{CPI}_{\text{without}}} = \frac{1 + \text{Branch stalls per instruction}}{1}$$

The branch stalls per instruction $= (.05)(1) + (.12)(2) + (.08)(1) = .37$. So the machine would run $\frac{1.37}{1} = 1.37$ times faster without any branch hazards.

3. The table in figure 3.19 is not complete - in particular, it ignores the following case :

```
ADD     R1, R3, R4
SW      R1, 0(R2)
```

The result of the `ADD` needs to be forwarded to the LMD register. Add entries to the table to take care of this case.

| Pipeline register containing source instruction | Opcode of source instruciton | Pipeline register containing destination instruction | Opcode of destination instruction | Destination of the forwarded result | Comparison (if equal then forward) |
|---|---|---|---|---|---|
| MEM/WB | Register-register ALU | EX/MEM | Store | Data Memory | $MEM/WB.IR_{16..20} = EX/MEM.IR_{11..15}$ |