More Pipelining

Pitfall: Expecting the improvement of one aspect of a machine to increase performance by an amount proportional to the size of the improvement.

Problem: Suppose a program runs in 100 seconds on a machine, with multiply operations responsible for 80 seconds of this time. How much do I have to improve the speed of multiplication if I want my program to run five times faster?

Execution time after improvement = (Execution time affected by improvement/amount of improvement) + Execution time unaffected

Solve the above problem.

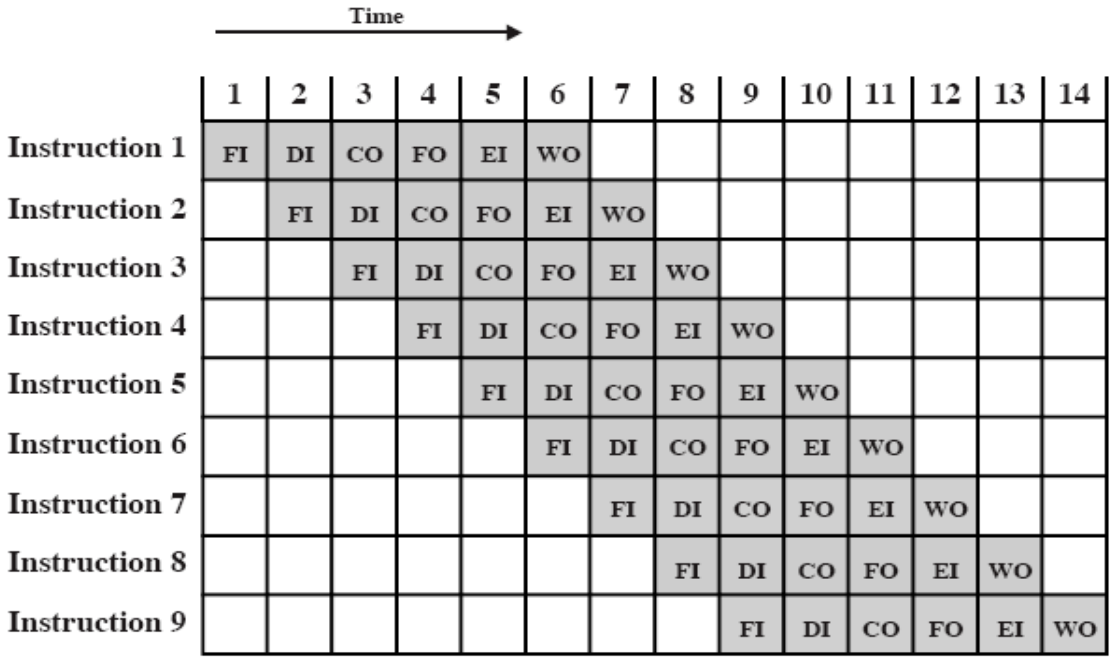

Figure 12.10 Timing Diagram for Instruction Pipeline Operation

Some important things about pipelining to note:

1) Each instruction does not necessarily go through each stage of the pipeline:

Q1: Give two different examples of instructions where this is the case.

2) The hope is that all of the stages can be performed in parallel.

Q2: Give an example where this is not possible.

3) The conditional branch presents a particular problem.

Q3: Why?

MIPS instructions classically take five steps:

1. Fetch instructions from memory.
2. Read registers while decoding the instruction.
3. Execute the operation or calculate an address.
4. Access an operand in data memory.
5. Write the result into a register.

The MIPS instruction set was designed for pipeline execution.

1. MIPS instructions are the same length. x86 instructions vary from 1 byte to 17 bytes and pipelining is much more challenging.
2. MIPS has only a few instruction formats, with the source operand being located in the same place in each instruction.
3. Memory operands only appear in loads or stores in MIPS.
4. Operands must be aligned in memory.

Pipeline Hazards

1. Structural hazards: Hardware cannot support the combination of instructions.
2. Control hazards: Need to make a decision based on the results of one instruction while others are executing.
    a. Stall.
    b. Predict:
        i. Always Fail
        ii. Taken/Not Taken
        iii. Dynamic Prediction
    c. Delayed decision
3. Data hazards: An instruction depends on the results of a previous instruction still in the pipeline.

Let's examine the effect of a conditional branch in the pipeline.

Q4: Explain the following diagram in detail identifying exactly what the branch penalty is and why it is the duration it is.

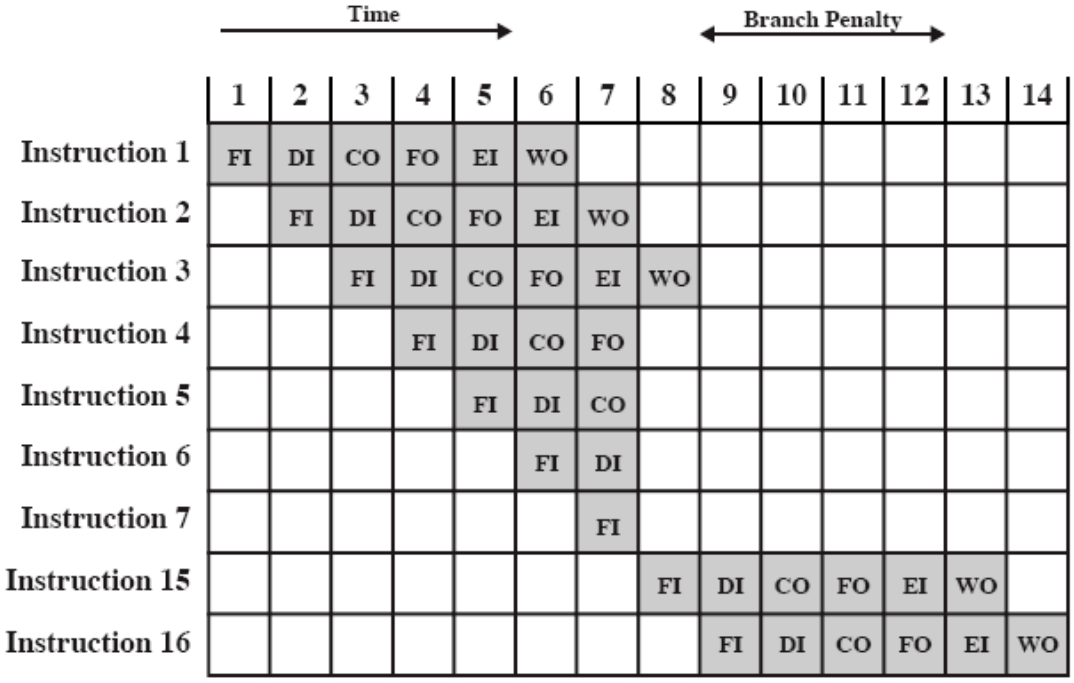

Figure 12.11 The Effect of a Conditional Branch on Instruction Pipeline Operation

Another way to look at the effects of a branch is:

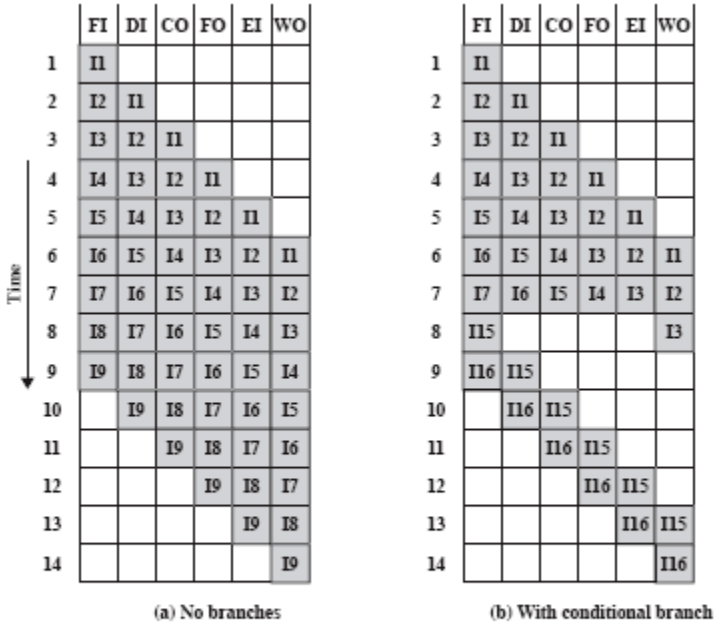

Figure 12.13 An Alternative Pipeline Depiction

Yet one more way to look at the effects of a conditional branch.

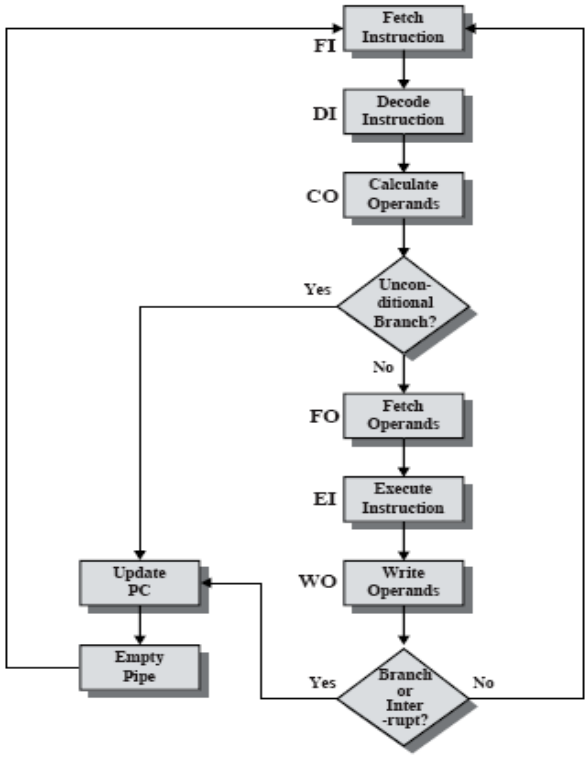

Figure 12.12 Six-Stage Instruction Pipeline

4) Different stages of the pipeline might have different durations.

Q5: Give an example of different stages having different durations.

5) The CO stage might be delayed because the value used as part of the calculation depends on a register that will be affected by an instruction still in the pipeline.

Q6: Give an example of this and show the pipeline as it would look when the problem is encountered.

6) It is not necessarily the case that adding more stages to the pipeline increases performance because of the overhead involved with moving data along in the various buffers of the pipeline.

7) Increasing the number of stages increases the overhead of the control logic used to handle memory & register dependencies and in optimizing the pipeline.