5.5 Instruction Pipelining

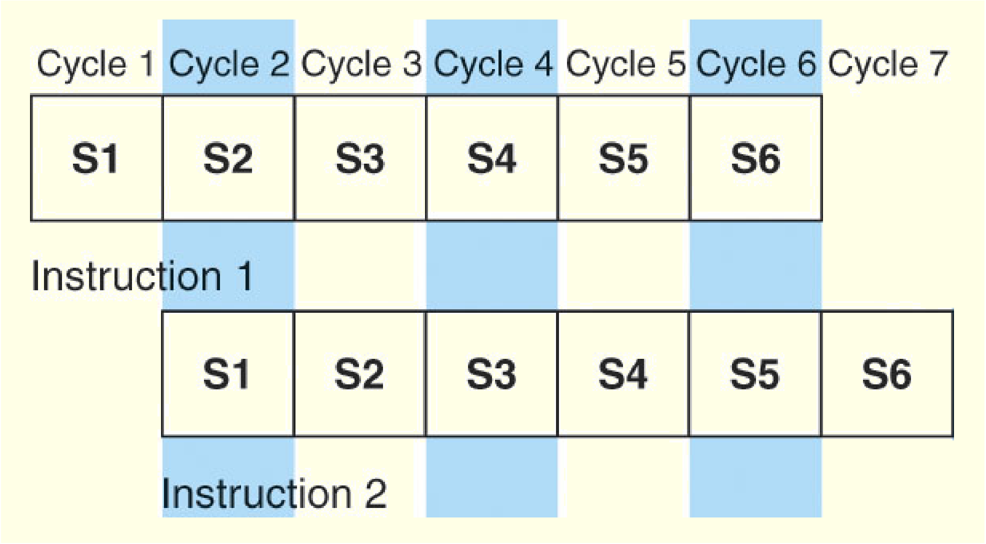
- Some CPUs divide the fetch-decode-execute cycle into smaller steps.
- These smaller steps can often be executed in parallel to increase throughput.
- Such parallel execution is called *instruction pipelining*.
- Instruction pipelining provides for *instruction level parallelism* (*ILP*)

Decompose the fetch-decode-execute cycle into component steps.

1. Fetch instruction
2. Decode opcode
3. Calculate effective address of operands
4. Fetch operands
5. Execute instruction
6. Store result

Imagine a six-stage pipeline

- S1 fetches the instruction
- S2 decodes the instruction
- S3 determines the effective address of operands
- S4 fetches operands
- S5 executes the instruction
- S6 stores the result.



Two Instructions Going through a 6-stage Pipeline

- The theoretical speedup offered by a pipeline can be determined as follows: Let t_p be the time per stage. Each instruction represents a task, *T*, in the pipeline. The first task (instruction) requires $k \times t_p$ time to complete in a *k*-stage pipeline. The remaining (n - 1) tasks emerge from the pipeline one per cycle. So the total time to complete the remaining tasks is

$$(n - 1)t_p.$$

Thus, to complete *n* tasks using a *k*-stage pipeline requires:

$$(k \times t_p) + (n - 1)t_p = (k + n - 1)t_p.$$

- If we take the time required to complete *n* tasks without a pipeline and divide it by the time it takes to complete *n* tasks using a pipeline, we find:

$$\text{Speedup } S = \frac{nt_n}{(k + n - 1)\, t_p}$$

- If we take the limit as *n* approaches infinity, ($k + n$ - 1) approaches *n,* which results in a theoretical speedup of:

$$\text{Speedup } S = \frac{kt_p}{t_p} = k$$

However:

- An instruction pipeline may stall, or be flushed for any of the following reasons:
    - Resource conflicts.
        - Example: in the same time period, one instruction seeks to read memory while another needs to store a value in memory
    - Data dependencies.
        - The result of one instruction, not yet completed, is to be used as an operand to a following instruction.
    - Conditional branching.
        - Control flow is altered based on computed conditions.
- Measures can be taken at the software level as well as at the hardware level to reduce the effects of these hazards, but they cannot be totally eliminated.