CS525 Parallel Computing

Ⓒ Xiyuan Chen

Spring Term, 2024

Information

- No slides, no textbook.

Contents

1 Application properties of parallelism

- Simulations. e.g., weather forecasting, molecular dynamics, etc.
- Learning & Data Analysis. e.g., machine learning, data mining, etc.

2 Parallel Programming Platforms

There is no slides but I found a past course slide for reference. here.

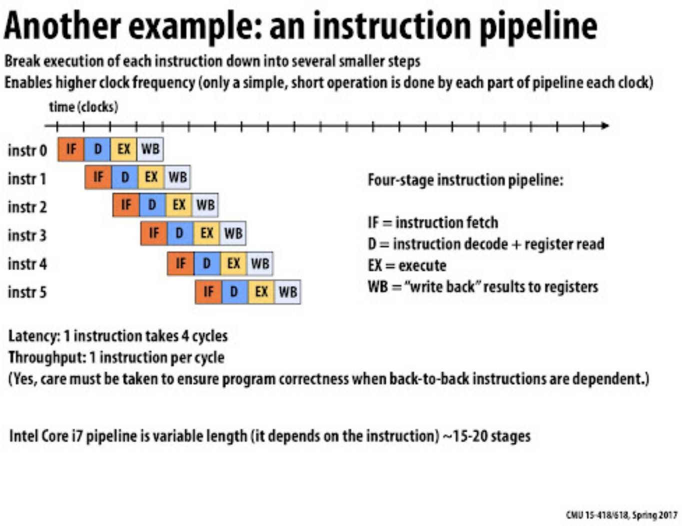
2.1 Processor

Jobs of a processor: 1. fetch instructions. 2. decode instructions. 3. execute instructions. 4. store results.

Most of the time, the instructions are of few types: e.g. MOVE, ADD, LD, FSD

2.1.1 Pipeline Parallelism

Basic idea: an instruction can be executed while the next one is being decoded and the next one is being fetched.



Another example: an instruction pipeline

Break execution of each instruction down into several smaller steps

Enables higher clock frequency (only a simple, short operation is done by each part of pipeline each clock)

Four-stage instruction pipeline:

IF = instruction fetch
D = instruction decode + register read
EX = execute
WB = "write back" results to registers

Latency: 1 instruction takes 4 cycles
Throughput: 1 instruction per cycle
(Yes, care must be taken to ensure program correctness when back-to-back instructions are dependent.)

Intel Core i7 pipeline is variable length (it depends on the instruction) ~15-20 stages

CMU 15-418/618, Spring 2017

Figure 1: Pipeline Parallelism

Limitations:

- **Bottleneck**: the slowest stage determines the speed of the pipeline. We solve by 1. decpmosing the slowest stage into multiple stages. 2. Use multiple pipelines (Superscalar Excution, see below).
- **Conditional jumps**: the pipeline must be flushed when a branch is encountered. We solve by branch prediction.

Superscalar Execution Having multiple excution units (pipelines) to execute multiple instructions at the same time.

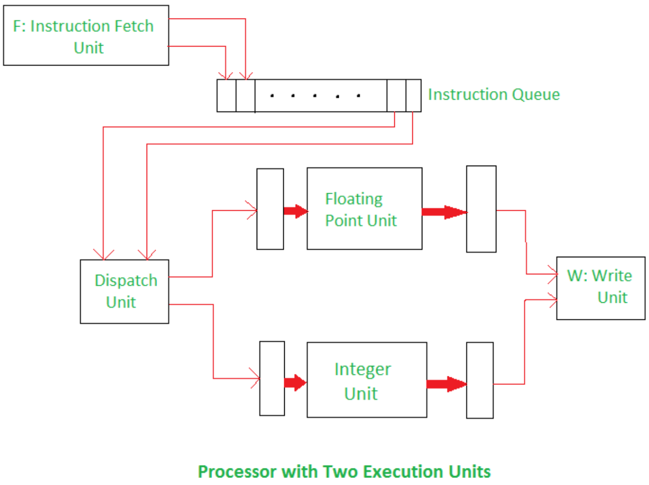


Figure 2: Superscalar Execution Example with 2 Units

How to schedule instructions to multiple pipelines? We consider:

- **Data Dependencies**: the result of one operation is an input to the next.
- **Read Dependencies**: the result of one operation is read by the next.
- **Control Dependencies**: the next operation depends on the result of a conditional branch.
- **Resource Dependencies**: the next operation requires the same resource as the current one.

We also **unroll the loops** to expose more parallelism.

Difference between pipelining and superscalar

- pipelining focuses on breaking down an instruction into sequential stages, whereas superscalar emphasizes on executing multiple instructions at the same time.
- pipelining typically has one execution unit, whereas superscalar has multiple execution units.
- superscalar is more parallel than pipelining.