Superscalar Organization

Nima Honarmand

Review: Instruction-Level Parallelism (ILP)

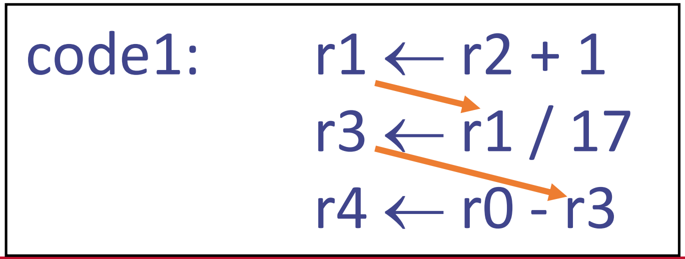
- “Parallelism is the number of independent tasks available”
- ILP is a measure of inter-dependencies between insns
- Average ILP = num. instruction / num. cyc required in an “ideal machine”

code1: ILP = 1

i.e. must execute serially

code2: ILP = 3

i.e. can execute at the same time



```
code1:      r1 ← r2 + 1
            r3 ← r1 / 17
            r4 ← r0 - r3
```

```
code2:      r1 ← r2 + 1
            r3 ← r9 / 17
            r4 ← r0 - r10
```

ILP != IPC

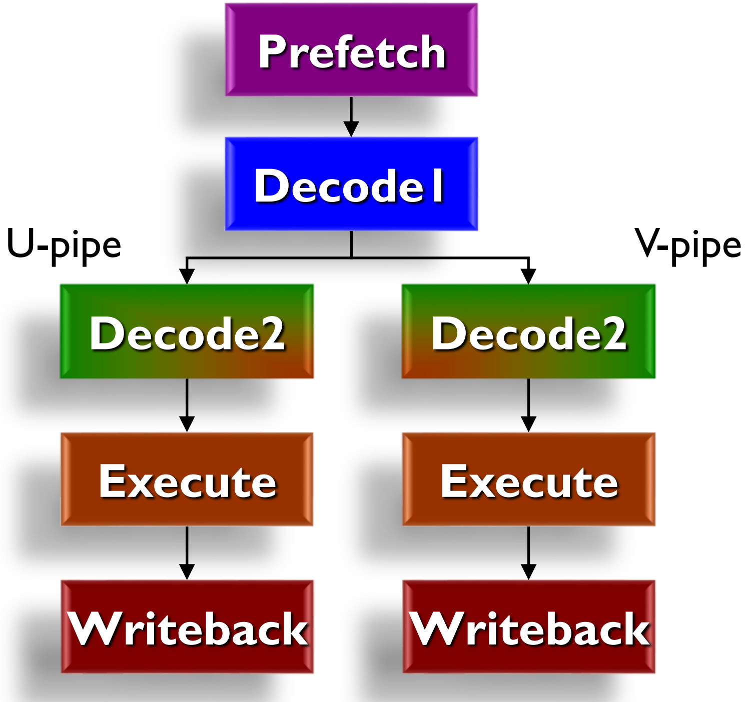
- ILP usually assumes
  - Infinite resources
  - Perfect fetch and branch prediction
  - Unit-latency for all instructions
- ILP is a property of the program dataflow
- IPC is the “real” observed metric
  - How many insns. are executed per cycle
- ILP is an upper-bound on the attainable IPC
  - Specific to a particular program

Purported Limits on ILP

Weiss and Smith [1984]	1.58
Sohi and Vajapeyam [1987]	1.81
Tjaden and Flynn [1970]	1.86
Tjaden and Flynn [1973]	1.96
Uht [1986]	2.00
Smith et al. [1989]	2.00
Jouppi and Wall [1988]	2.40
Johnson [1991]	2.50
Acosta et al. [1986]	2.79
Wedig [1982]	3.00
Butler et al. [1991]	5.8
Melvin and Patt [1991]	6
Wall [1991]	7
Kuck et al. [1972]	8
Riseman and Foster [1972]	51
Nicolau and Fisher [1984]	90

ILP Limits of Scalar Pipelines (1)

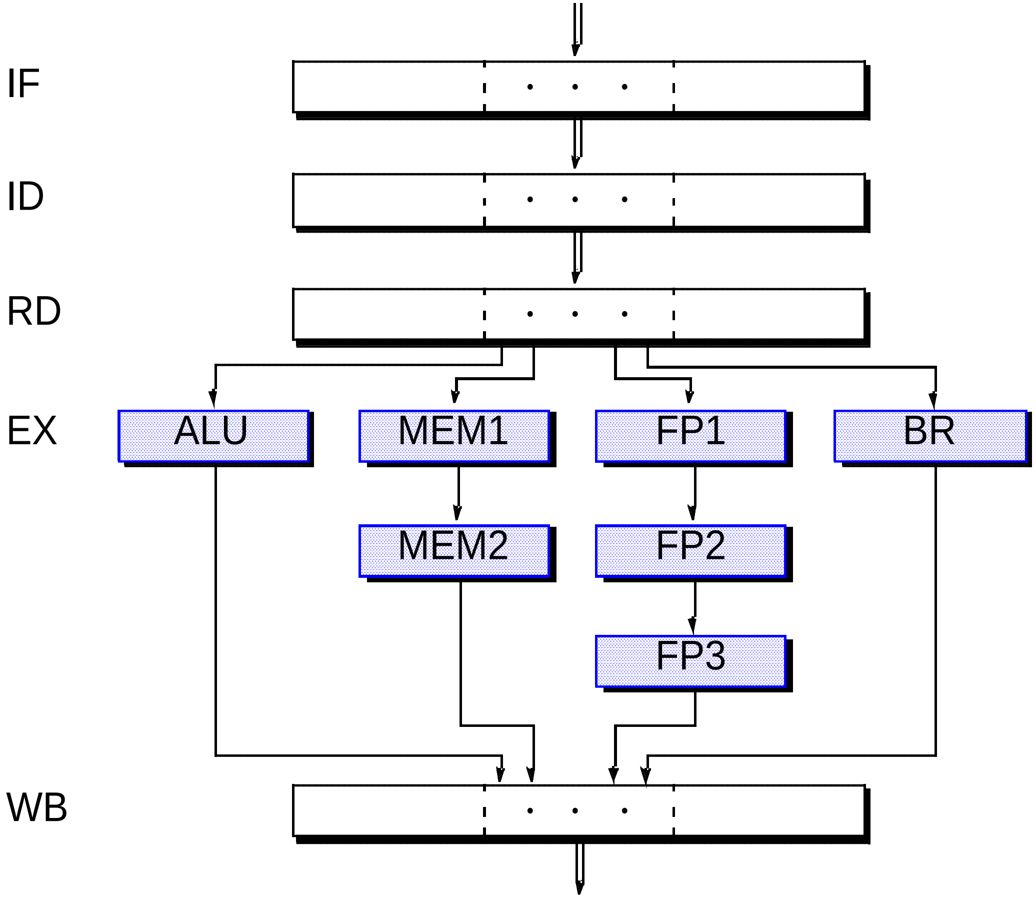
- Scalar upper bound on throughput
  - Limited to IPC <= 1
  - Solution: **superscalar** pipelines with multiple insns at each stage



Pentium Pipeline

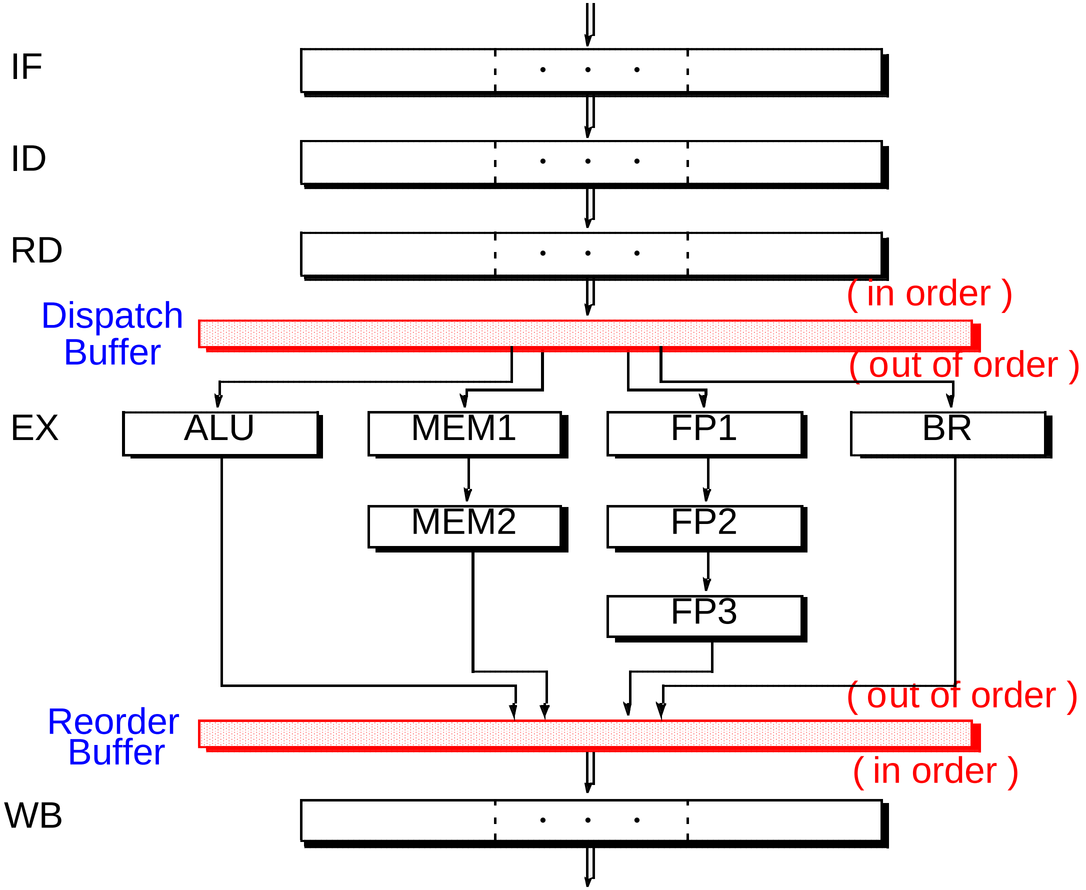
ILP Limits of Scalar Pipelines (2)

- **Unified pipeline**: a pipeline where all instructions go through the same stages
  - Like our 5-stage pipeline
- Unified pipelines are inefficient
  - Lower resource utilization and longer instruction latency
  - Solution: **diversified** pipelines



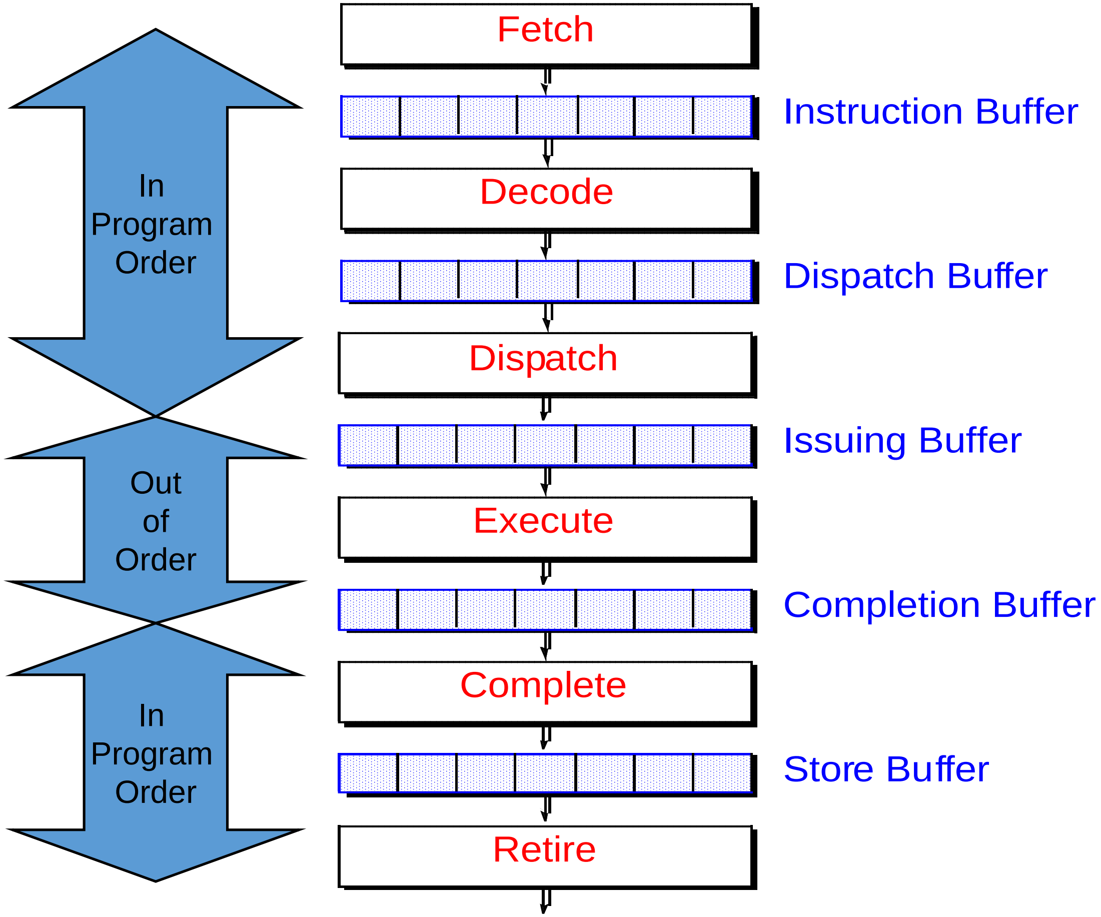
ILP Limits of Scalar Pipelines (3)

- Rigid pipeline stall policy
  - A stalled instruction stalls all newer instructions
  - Solution 1: **out-of-order** execution



ILP Limits of Scalar Pipelines (3)

- Rigid pipeline stall policy
  - A stalled instruction stalls all newer instructions
  - Solution 1: **out-of-order** execution
  - Solution 2: **inter-stage buffers**



ILP Limits of Scalar Pipelines (4)

- Instruction dependencies limit parallelism
  - Frequent stalls due to data and control dependencies
  - Solution 1: **renaming** – for WAR and WAW register dependences
  - Solution 2: **speculation** – for control dependences and memory dependences

Summary : ILP Limits of Scalar Pipelines

1) Scalar upper bound on throughput
   - Limited to IPC <= 1
   - Solution: **superscalar** pipelines with multiple insns at each stage

2) Inefficient unified pipeline
   - Lower resource utilization and longer instruction latency
   - Solution: **diversified** pipelines

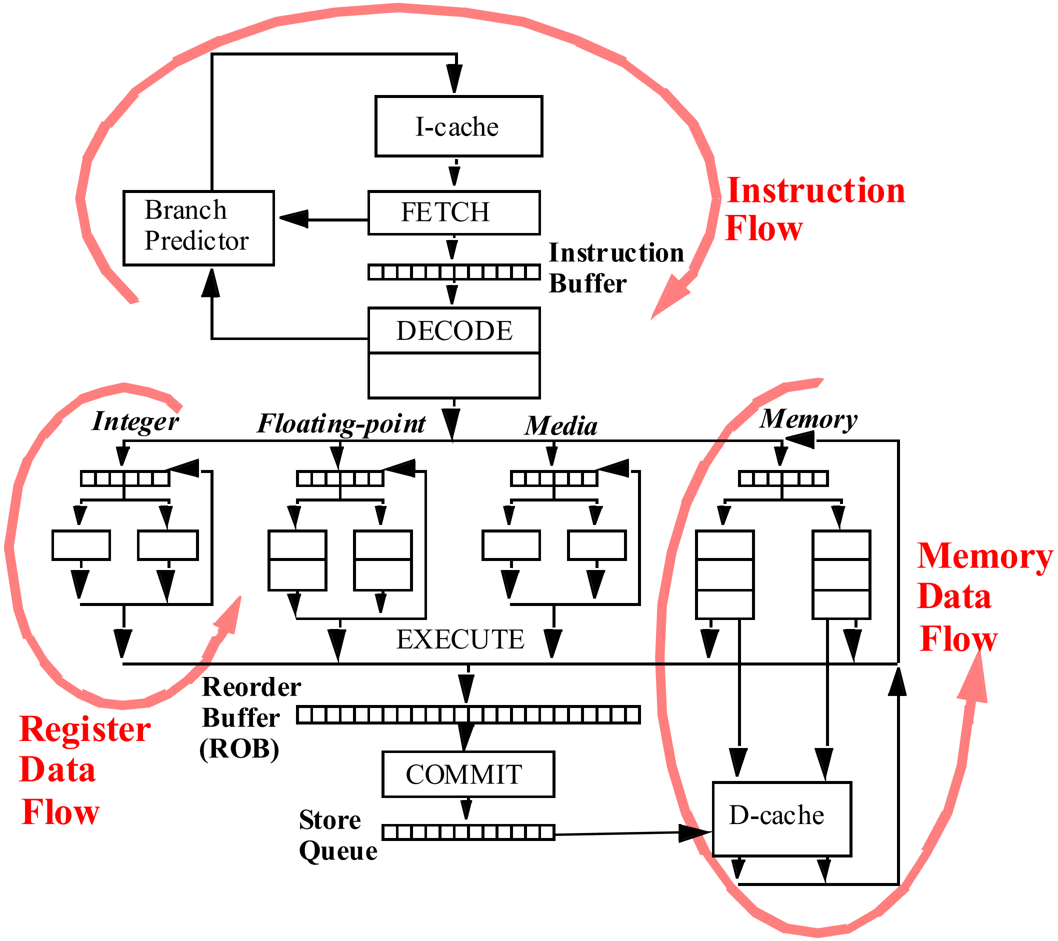
3) Rigid pipeline stall policy
   - A stalled instruction stalls all newer instructions
   - Solution: **out-of-order** execution and **inter-stage buffers**

4) Instruction dependencies limit parallelism
   - Frequent stalls due to data and control dependencies
   - Solutions: **renaming** and **speculation**

State of the art: ***Out-of-Order Superscalar Speculative Pipelines***

Superscalar Pipelines: Overall Picture

- Fetch issues:
  - Fetch multiple isns
  - Branches and speculation
- Decode issues:
  - Identify insns
  - Find dependences
- Execution issues:
  - Dispatch insns
  - Resolve dependences
  - Forwarding networks
  - Multiple outstanding memory accesses
- Completion issues:
  - Out-of-order completion
  - Speculative instructions
  - Precise exceptions



State of the art: ***Out-of-Order Superscalar Speculative Pipelines***