Beyond Pipelining

Virendra Singh
Associate Professor
Computer Architecture and Dependable Systems Lab
Department of Electrical Engineering
Indian Institute of Technology Bombay
http://www.ee.iitb.ac.in/~viren/
E-mail: viren@ee.iitb.ac.in

CS-683: Advanced Computer Architecture

Lecture 8 (21 Aug 2013)

CADSL

Pipelining

Single Lane Traffic (Chaining)

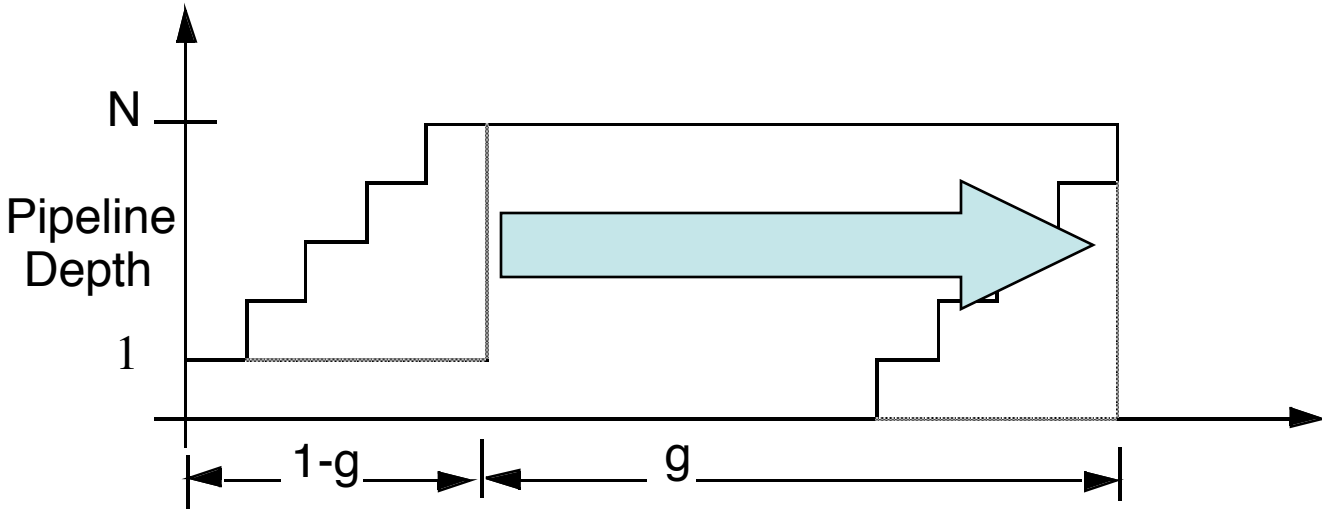
Limits of Pipelining

- IBM RISC Experience
  - Control and data dependences add 15%
  - Best case CPI of 1.15, IPC of 0.87
  - Deeper pipelines (higher frequency) magnify dependence penalties
- This analysis assumes 100% cache hit rates
  - Hit rates approach 100% for some programs
  - Many important programs have much worse hit rates

Limitations of Scalar Pipelines

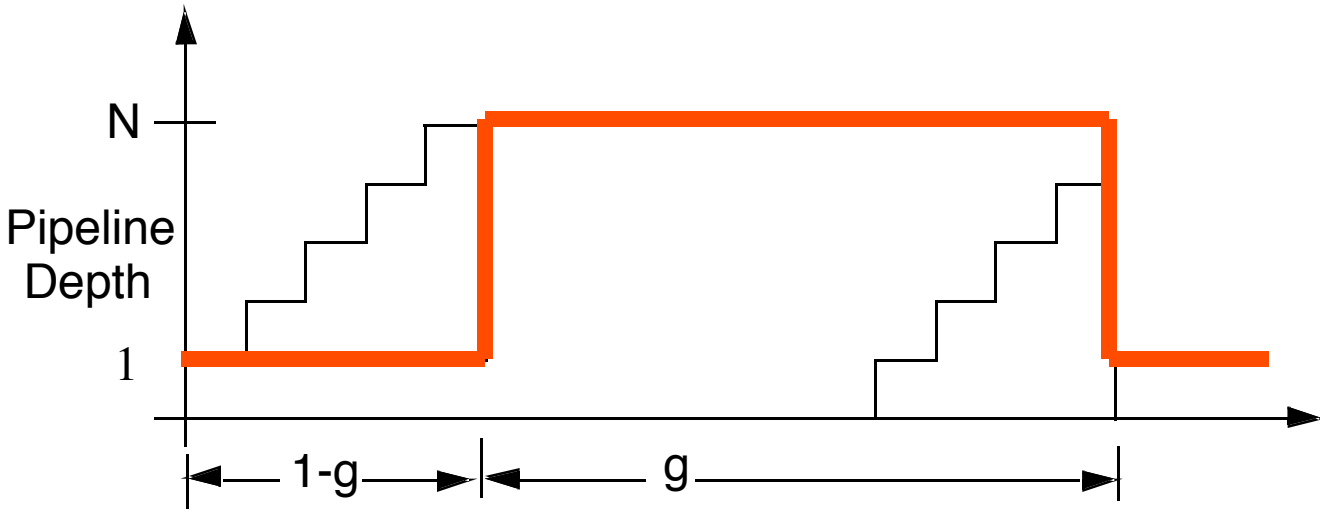
- Scalar upper bound on throughput
  - IPC <= 1 or CPI >= 1
- Unified pipeline
  - Long latency for each instruction
- Rigid pipeline stall policy
  - One stalled instruction stalls all newer instructions

Pipelined Performance Model



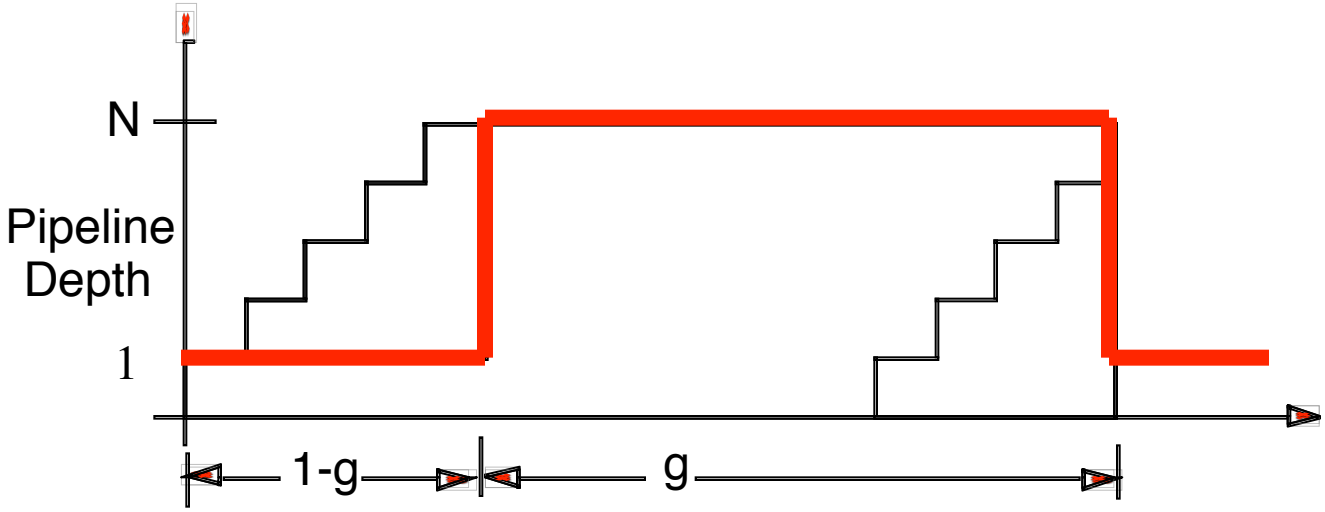
- g = fraction of time pipeline is filled
- 1-g = fraction of time pipeline is not filled (stalled)

Pipelined Performance Model



- g = fraction of time pipeline is filled
- 1-g = fraction of time pipeline is not filled (stalled)

Pipelined Performance Model



- Tyranny of Amdahl's Law [Bob Colwell]
  - When g is even slightly below 100%, a big performance hit will result
  - Stalled cycles are the key adversary and must be minimized as much as possible

Processor Performance

$$\text{Processor Performance} = \frac{\text{Time}}{\text{Program}}$$

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Time}}{\text{Cycle}}$$

(code size) **(CPI)** **(cycle time)**

- In the 1980's (decade of pipelining):
  - CPI: 5.0 => 1.15
- In the 1990's (decade of superscalar):
  - CPI: 1.15 => 0.5 (best case)
- In the 2000's (decade of multicore):
  - Marginal CPI improvement

Beyond Pipelining

IPC > 1

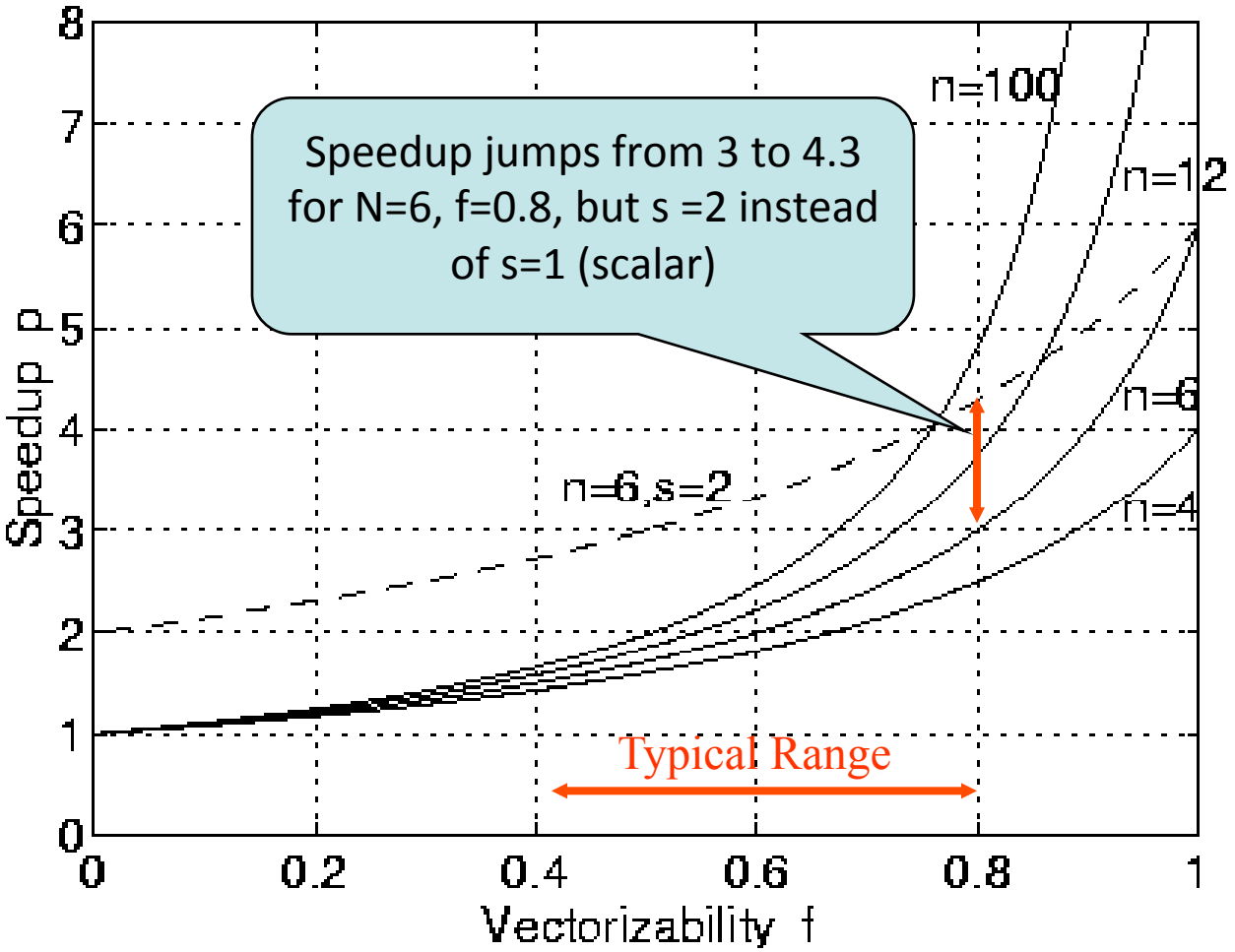
Limits on Instruction Level Parallelism (ILP)

Weiss and Smith [1984]	1.58
Sohi and Vajapeyam [1987]	1.81
Tjaden and Flynn [1970]	1.86 (Flynn's bottleneck)
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 (Jouppi disagreed)
Kuck et al. [1972]	8
Riseman and Foster [1972]	51 (no control dependences)
Nicolau and Fisher [1984]	90 (Fisher's optimism)

Superscalar Proposal

- Go beyond single instruction pipeline, achieve IPC > 1
- Dispatch multiple instructions per cycle
- Provide more generally applicable form of concurrency (not just vectors)
- Geared for sequential code that is hard to parallelize otherwise
- Exploit fine-grained or instruction-level parallelism (ILP)

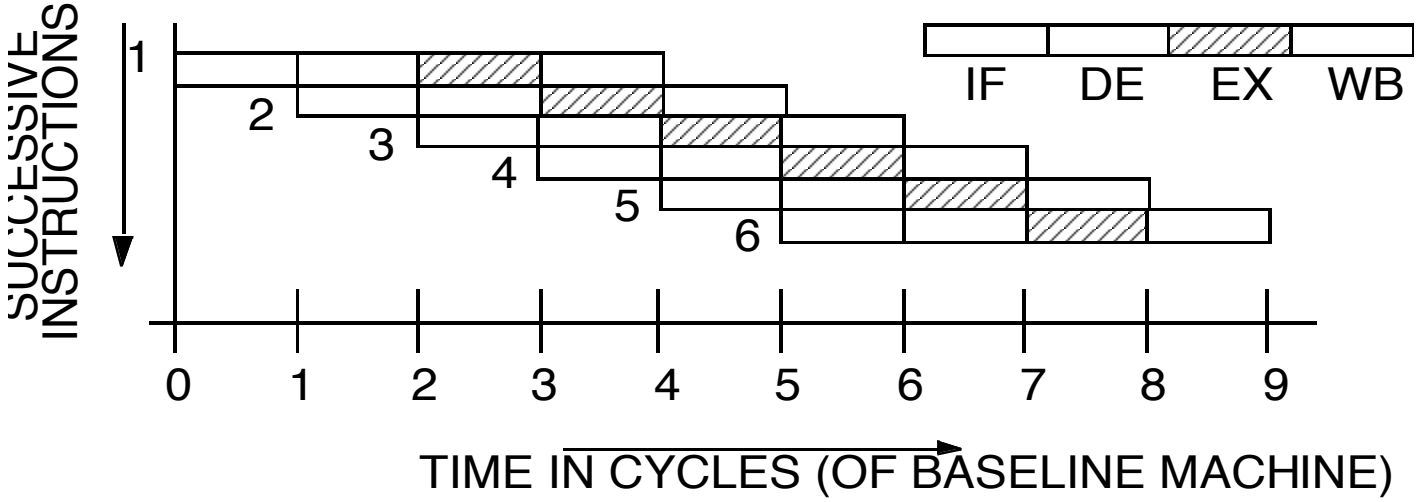
Motivation for Superscalar [Agerwala and Cocke]



Classifying ILP Machines

[Jouppi, DECWRL 1991]

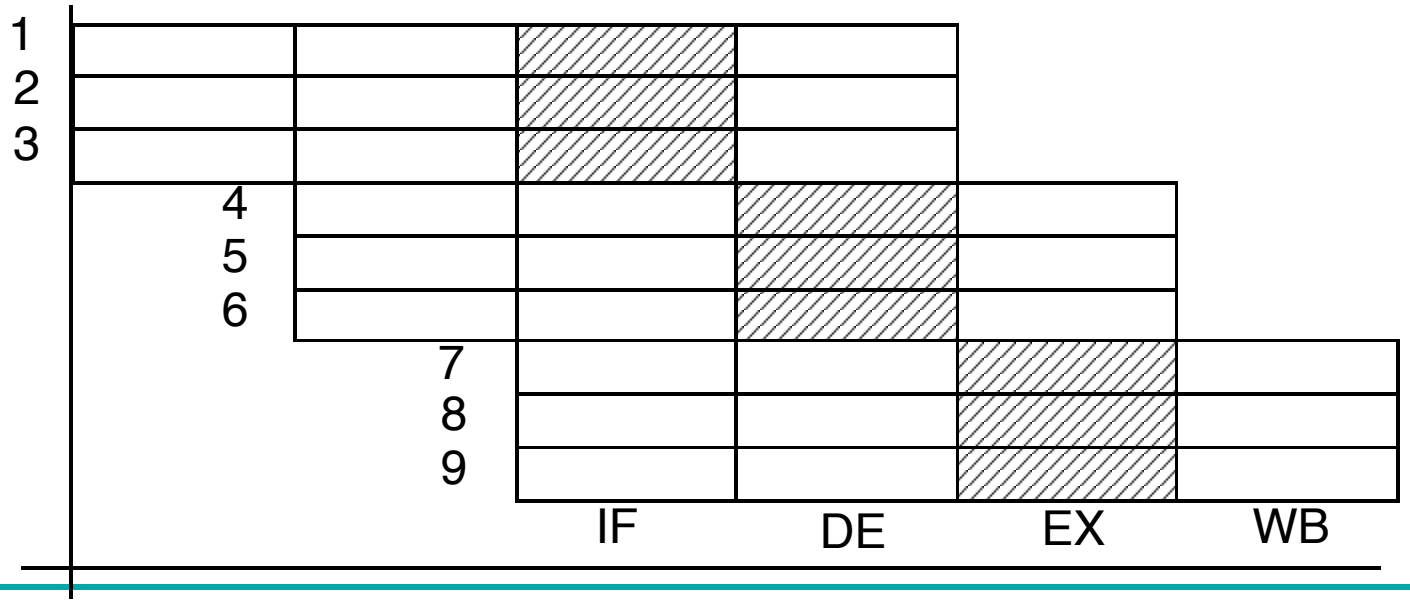
- Baseline scalar RISC
  - Issue parallelism = IP = 1
  - Operation latency = OP = 1
  - Peak IPC = 1



Classifying ILP Machines

[Jouppi, DECWRL 1991]

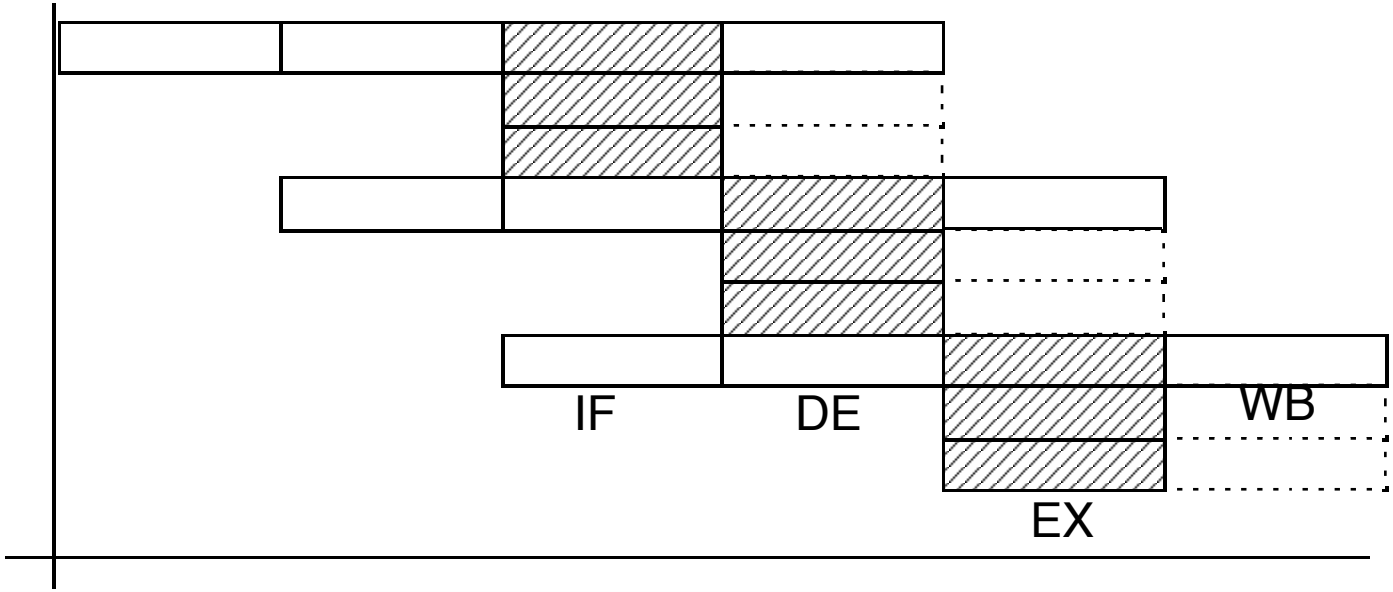
- Superscalar:
  - Issue parallelism = IP = n inst / cycle
  - Operation latency = OP = 1 cycle
  - Peak IPC = n instr / cycle (n x speedup?)



Classifying ILP Machines

[Jouppi, DECWRL 1991]

- VLIW: Very Long Instruction Word
  - Issue parallelism = IP = n inst / cycle
  - Operation latency = OP = 1 cycle
  - Peak IPC = n instr / cycle = 1 VLIW / cycle



Instruction-Level Parallelism

- When exploiting instruction-level parallelism, goal is to maximize IPC
  - **Pipeline IPC** =
    - Ideal pipeline IPC +
    - Structural stalls -
    - Data hazard stalls -
    - Control stalls -
- Parallelism with basic block is limited
  - Typical size of basic block = 3-6 instructions
  - Must optimize across branches

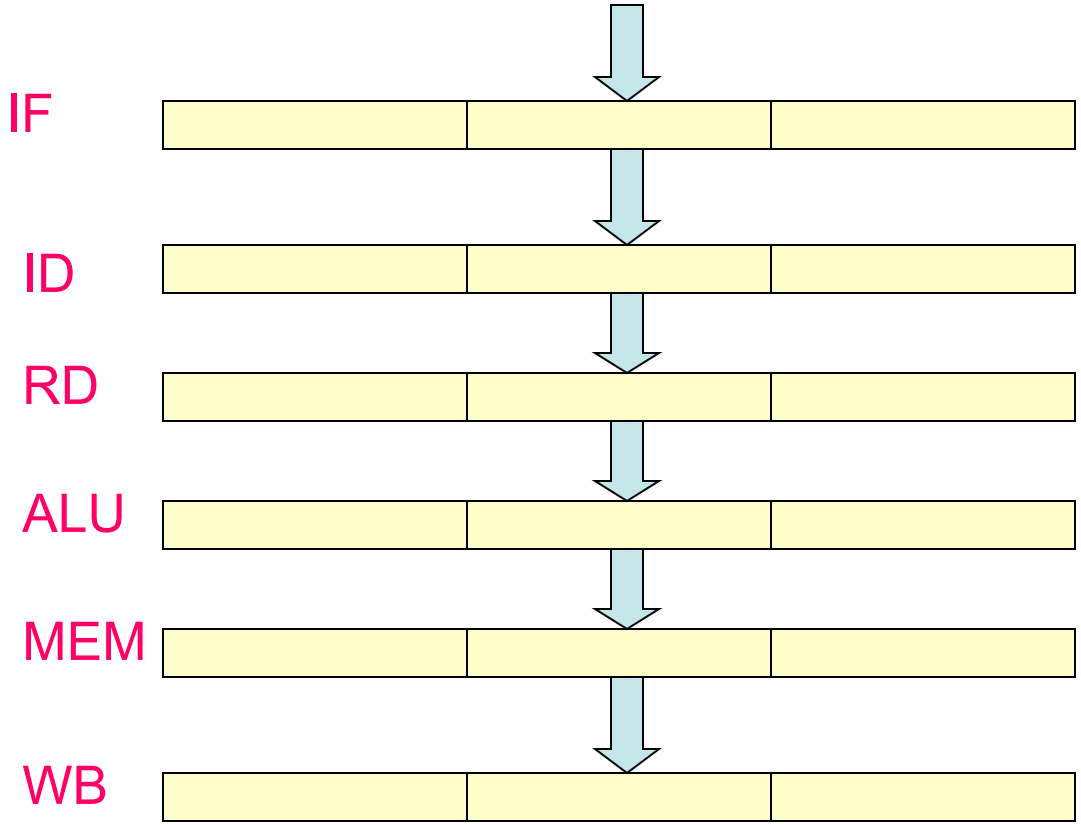
Limitations of Scalar Pipelines

- Scalar upper bound on throughput
  - IPC <= 1 or CPI >= 1
  - Solution: wide (superscalar) pipeline
- Inefficient unified pipeline
  - Long latency for each instruction
  - Solution: diversified, specialized pipelines
- Rigid pipeline stall policy
  - One stalled instruction stalls all newer instructions
  - Solution: Out-of-order execution, distributed execution pipelines

Superscalar Architecture

- Simple concept
- Wide pipeline
- Instructions are not independent
- Superscalar architecture is natural descendant of pipelined scalar RISC
- Superscalar techniques largely concern the processor organization, independent of the ISA and the other architectural features
- Thus, possibility to develop a processor code compatible with an existing architecture

Superscalar Pipelines



Highway

Bad Traffic

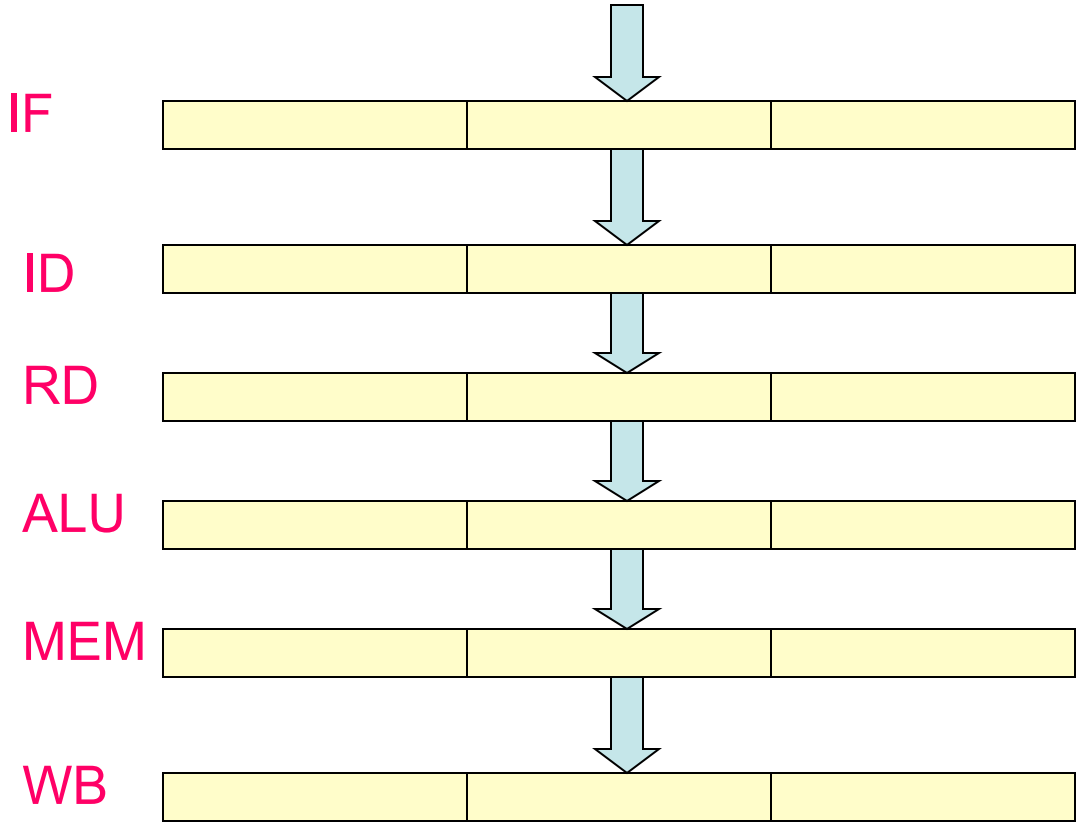
Instruction Level Parallelism

- Instruction parallelism of a program is a measure of the average number of instructions that a superscalar processor might be able to execute at the same time
- Mostly, ILP is determined by the number of true dependencies and the number of branches in relation to other instructions

Machine Level Parallelism

- Machine parallelism of a processor is a measure of the ability of processor to take advantage of the ILP
- Determined by the number of instructions that can be fetched and executed at the same time
- A challenge in the design of superscalar processor is to achieve good balance between instruction parallelism and machine parallelism

Superscalar Pipelines

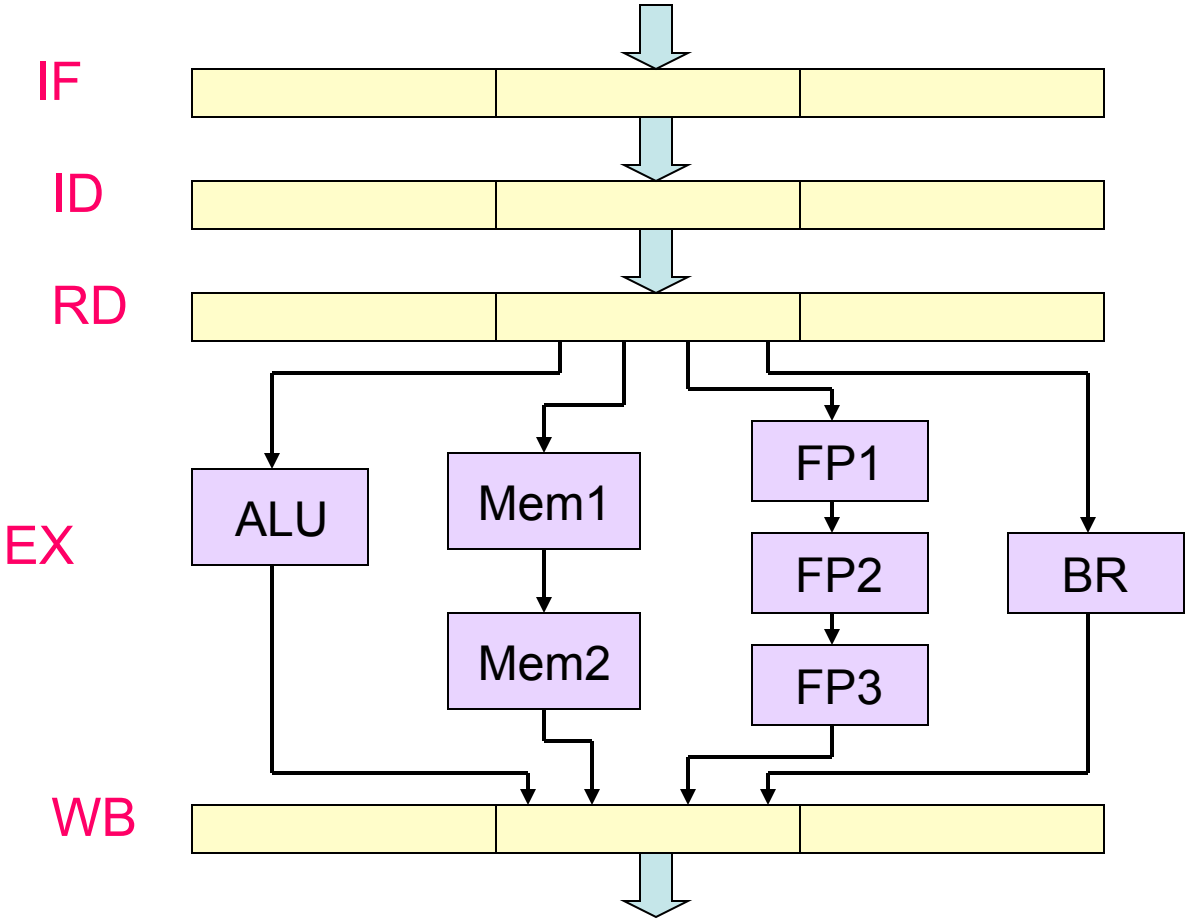


Superscalar Pipelines

Dynamic Pipelines

1. Alleviate the limitations of pipelined implementation
2. Use diversified pipelines
3. Temporal machine parallelism

Superscalar Pipelines (Diversified)



Superscalar Pipelines (Diversified)

Diversified Pipelines

- Each pipeline can be customized for particular instruction type
- Each instruction type incurs only necessary latency
- Certainly less expensive than identical copies
- If all inter-instruction dependencies are resolved then there is no stall after instruction issue

Require special consideration

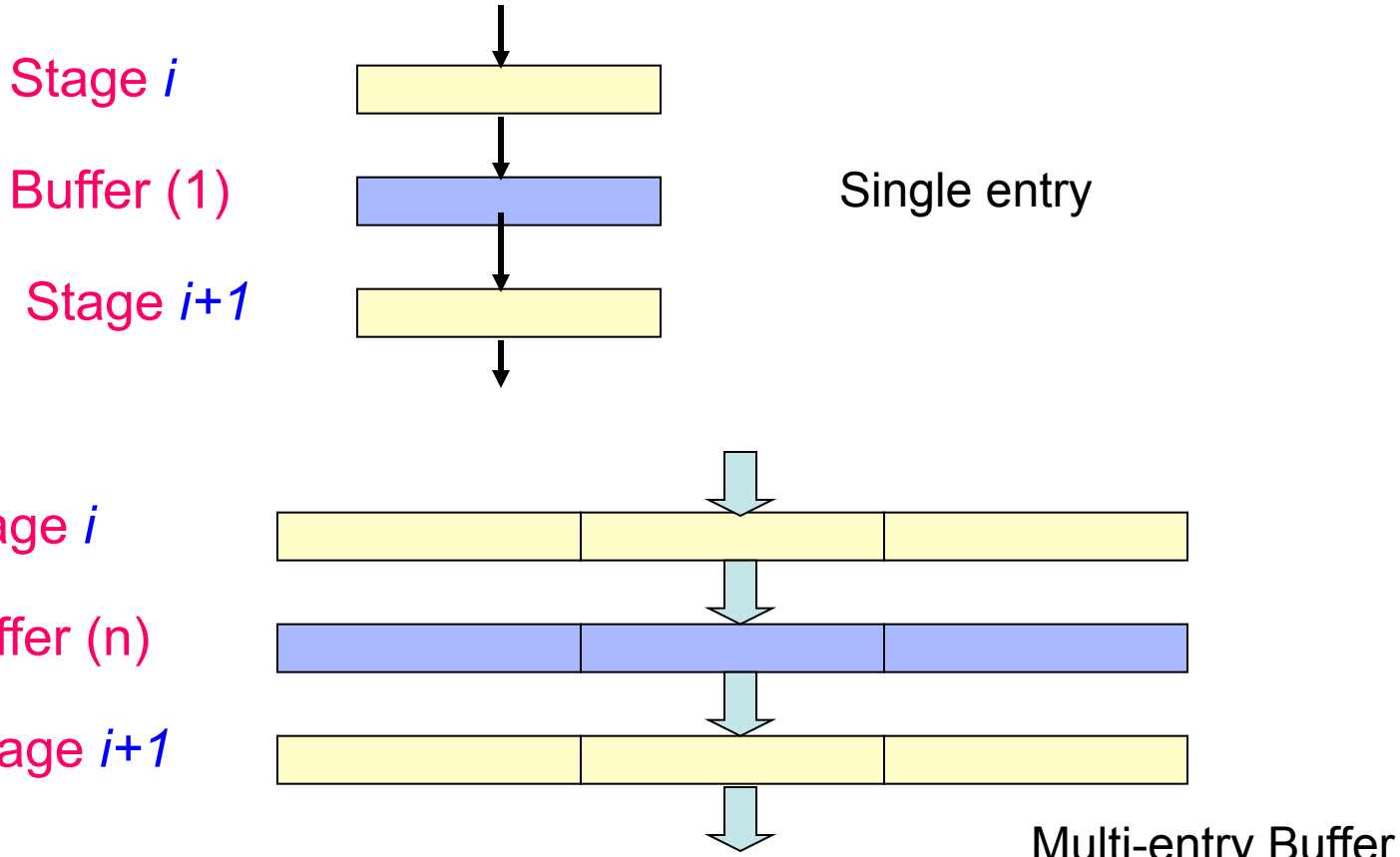
- Number and Mix of functional units

Superscalar Pipelines (Dynamic Pipelines)

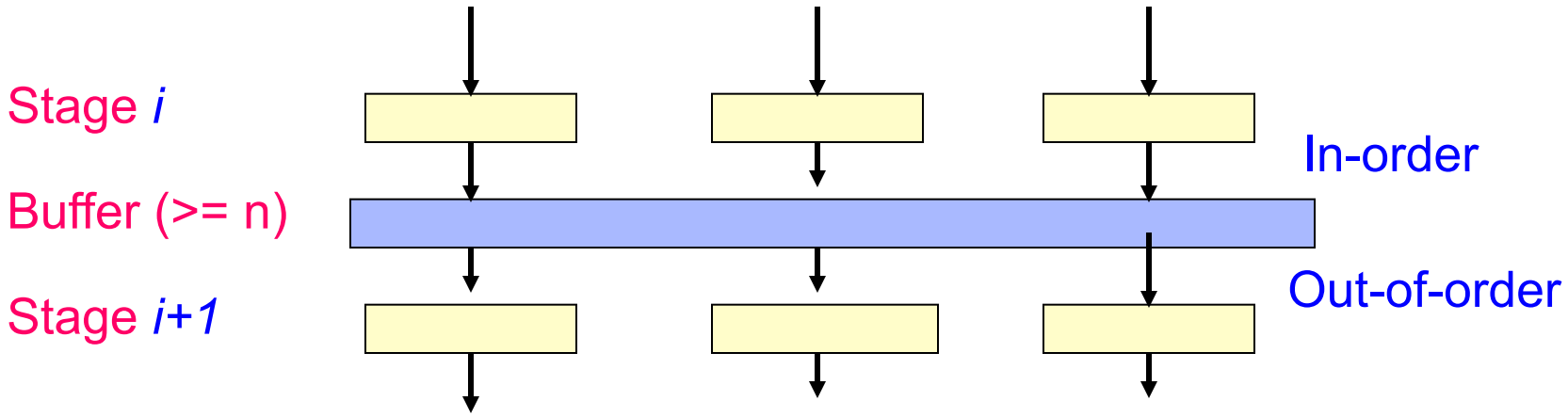
Dynamic Pipelines

- Buffers are needed
  - Multi-entry buffers
  - Every entry is hardwired to one read port and one write port
  - Complex multi-entry buffers
  - Minimize stalls

Superscalar Pipelines (Interpipeline-Stage Buffers)



Superscalar Pipelines (Interpipeline-Stage Buffers)

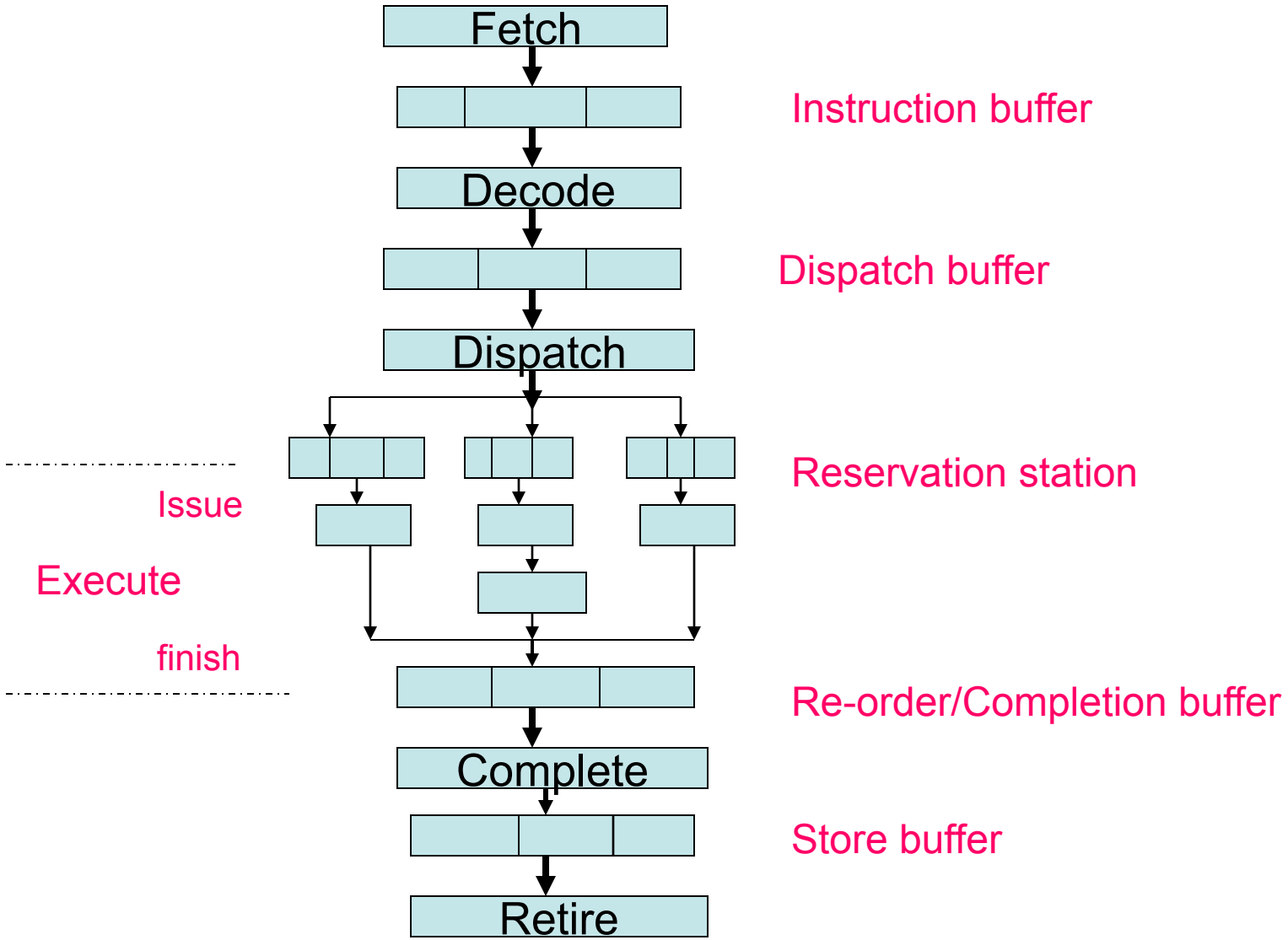


- Trailing instructions are allowed to bypass stalled instruction
- Out-of-order execution

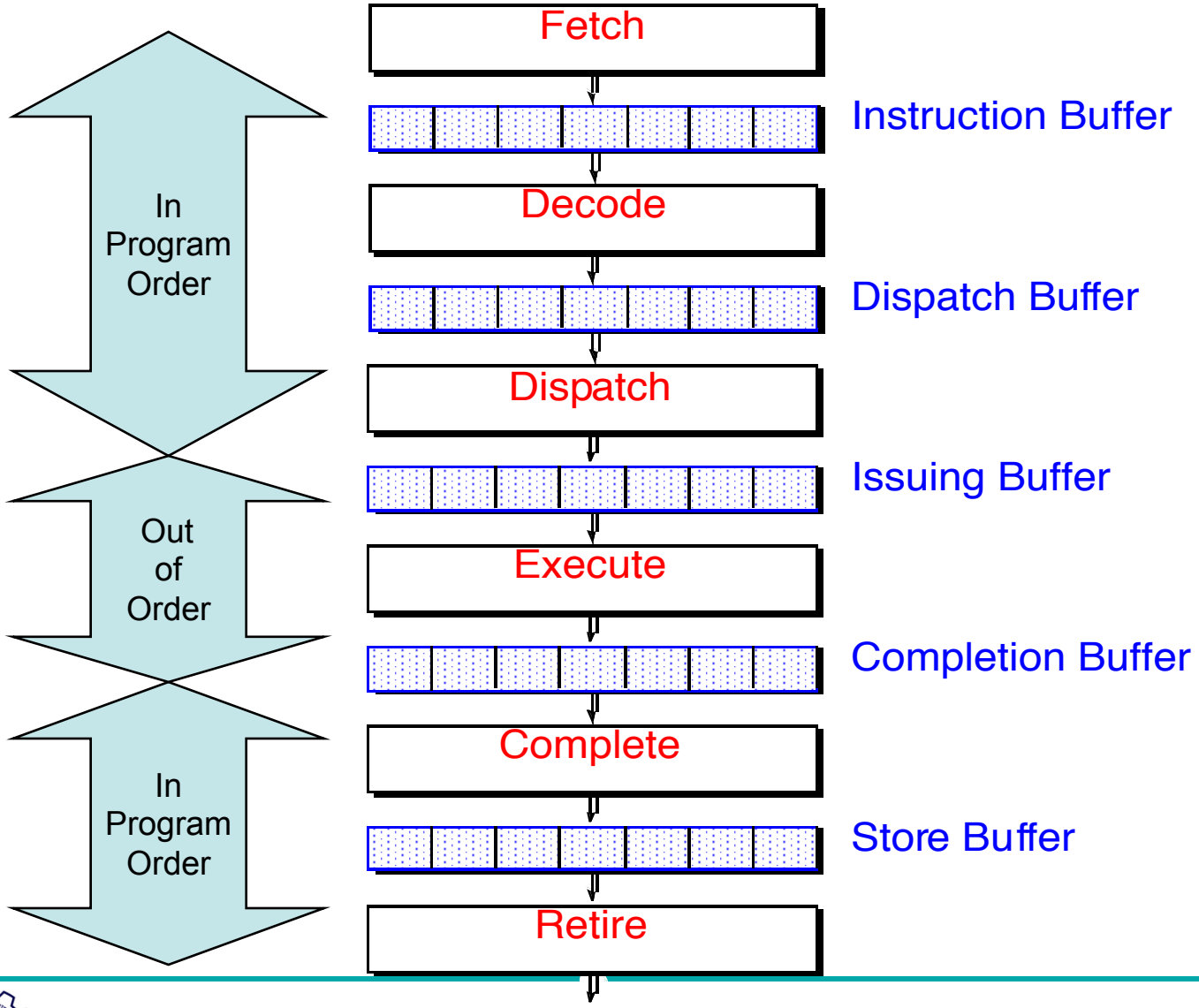
Instruction Issue & MLP

- ILP is not necessarily exploited by widening the pipelines and adding more resources
- Processor policies towards fetching decoding, and executing instruction have significant effect on its ability to discover instructions which can be executed concurrently
- Instruction issue is refer to the process of initiating instruction execution
- Instruction issue policy limits or enhances performance because it determines the processor's look ahead capability

Super-scalar Architecture

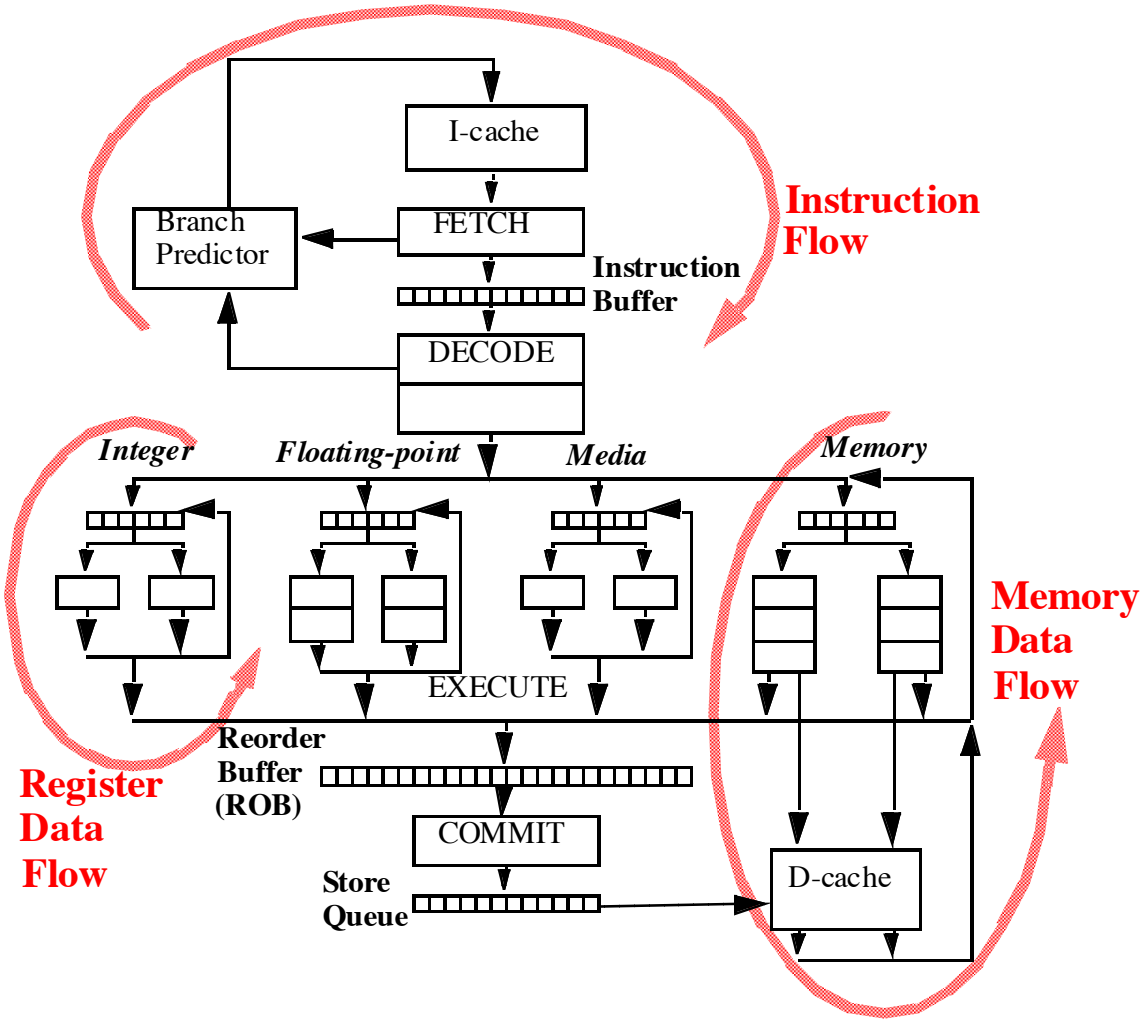


Superscalar Pipeline Stages



Express Way

Superscalar Challenges



Thank You