INSTRUCTION LEVEL PARALLELISM

Mahdi Nazm Bojnordi

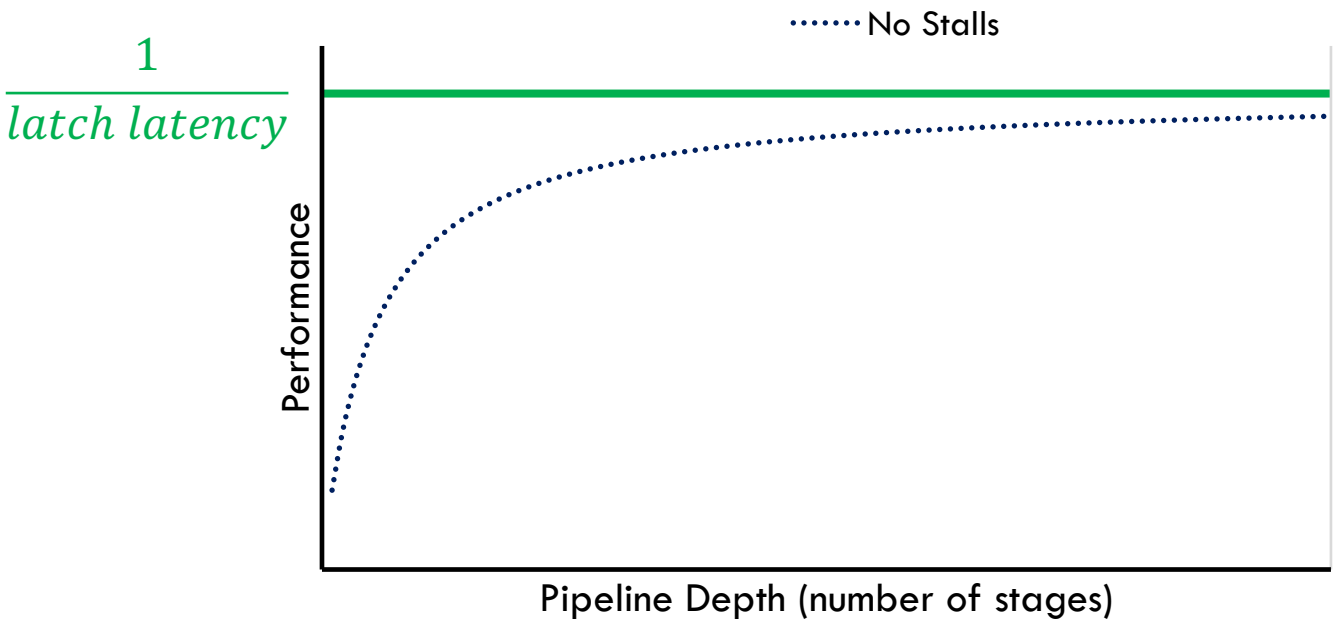
Assistant Professor

School of Computing

University of Utah

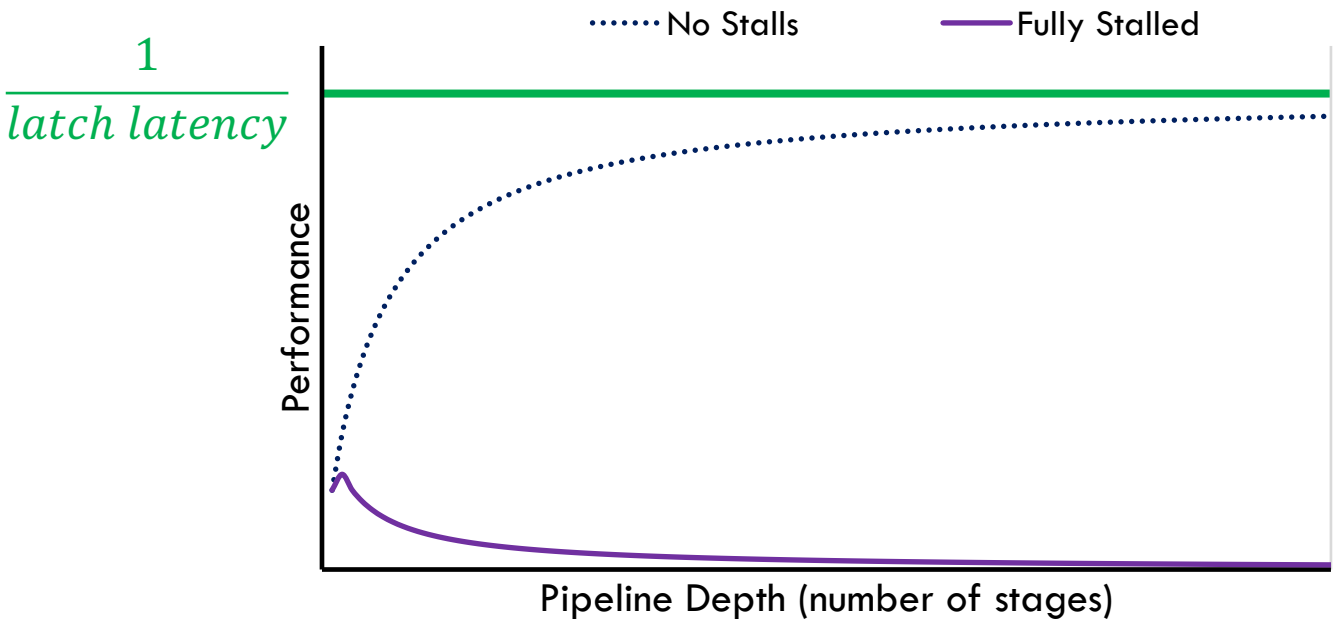
Performance vs. Pipeline Depth

- Impact of stall cycles on performance
  - Independent instructions
  - Dependent instructions



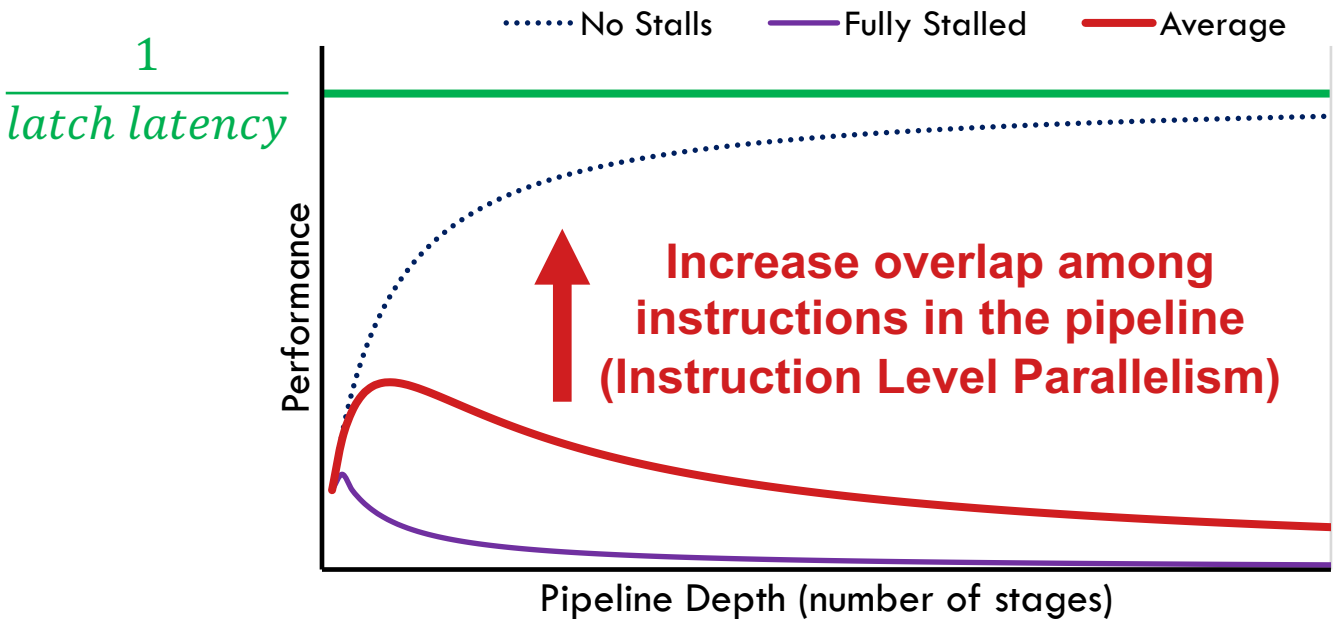
Performance vs. Pipeline Depth

- Impact of stall cycles on performance
  - Independent instructions
  - Dependent instructions



Performance vs. Pipeline Depth

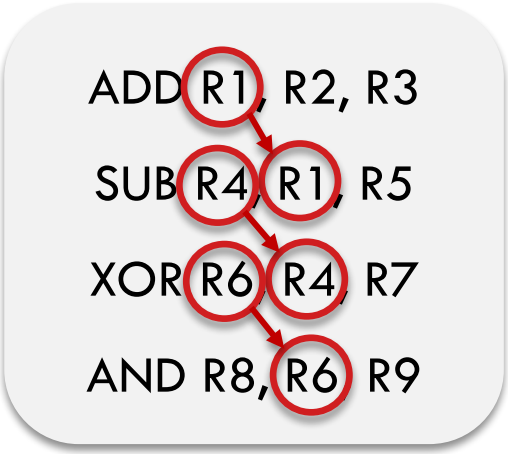
- Impact of stall cycles on performance
  - Independent instructions
  - Dependent instructions



Instruction Level Parallelism

- Potential overlap among instructions
  - A property of the program dataflow

Code 1



ILP = 1
Fully serial

Code 2

```
ADD R1, R2, R3
SUB R4, R6, R5
XOR R8, R2, R7
AND R9, R6, R0
```

ILP = 4
Fully parallel

Instruction Level Parallelism

- Potential overlap among instructions
  - A property of the program dataflow
  - Influenced by compiler

X ← A + B + C + D

Code 1:

```
ADD  R5, R1, R2
ADD  R5, R5, R3
ADD  R5, R5, R4
```

Instruction Level Parallelism

- Potential overlap among instructions
  - A property of the program dataflow
  - Influenced by compiler

X ← A + B + C + D

Code 1:

```
ADD  R5, R1, R2
ADD  R5, R5, R3
ADD  R5, R5, R4
```

Average ILP = 3/3 = 1
Five registers

Code 2:

```
ADD  R6, R1, R2
ADD  R7, R3, R4
ADD  R5, R6, R7
```

Average ILP = 3/2 = 1.5
Seven registers

Instruction Level Parallelism

- Potential overlap among instructions
  - A property of the program dataflow
  - Influenced by compiler
- An upper limit for attainable IPC for a given code
  - IPC represents exploited ILP

```
ADD  R5, R1, R2
ADD  R5, R5, R3
ADD  R5, R5, R4
```

Average ILP = 3/3 = 1
Five registers

```
ADD  R6, R1, R2
ADD  R7, R3, R4
ADD  R5, R6, R7
```

Average ILP = 3/2 = 1.5
Seven registers

Instruction Level Parallelism

- Potential overlap among instructions
  - A property of the program dataflow
  - Influenced by compiler
- An upper limit for attainable IPC for a given code
  - IPC represents exploited ILP
- Can be exploited by HW-/SW-intensive techniques
  - Dynamic scheduling in hardware
  - Static scheduling in software (compiler)