MEMORY SYSTEM

Mahdi Nazm Bojnordi

Assistant Professor

School of Computing

University of Utah

Overview

- Notes
  - Homework 9 (deadline Apr. 9th)
    - Verify your submitted file before midnight
- This lecture
  - Pipeline hazards
    - Control
  - Memory system
    - Cache

Recall: Pipeline Hazards

- Structural hazards: multiple instructions compete for the same resource
- Data hazards: a dependent instruction cannot proceed because it needs a value that hasn't been produced
- Control hazards: the next instruction cannot be fetched because the outcome of an earlier branch is unknown

Control Hazards

- Sample C++ code

```
for (i=100; i != 0; i--) {
    sum = sum + i;
}
total = total + sum;
```

How many branches in this code?

Control Hazards

- Sample C++ code

```
for (i=100; i != 0; i--) {
    sum = sum + i;
}
total = total + sum;
```

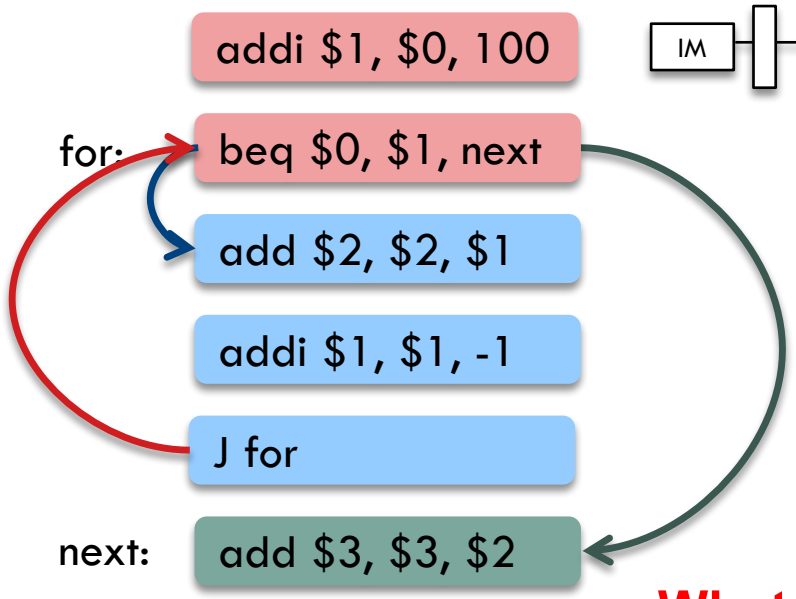
```
        addi $1, $0, 100
for:    beq $0, $1, next
        add $2, $2, $1
        addi $1, $1, -1
        J for
next:   add $3, $3, $2
```

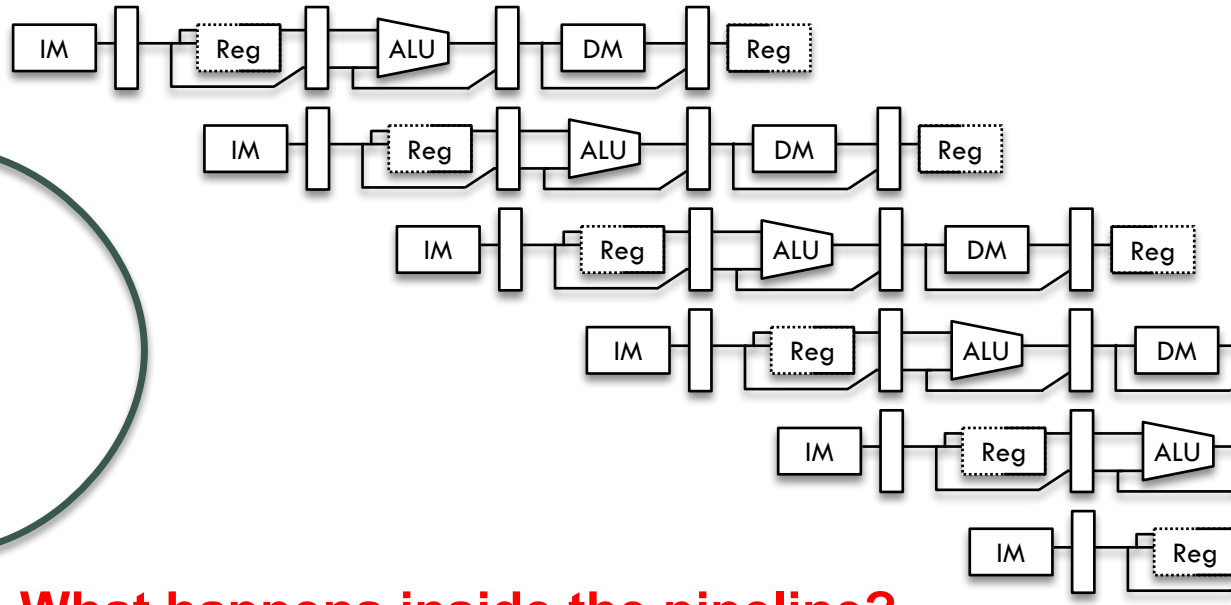
What are possible target instructions?

Control Hazards

- Sample C++ code

```
for (i=100; i != 0; i--) {
    sum = sum + i;
}
total = total + sum;
```

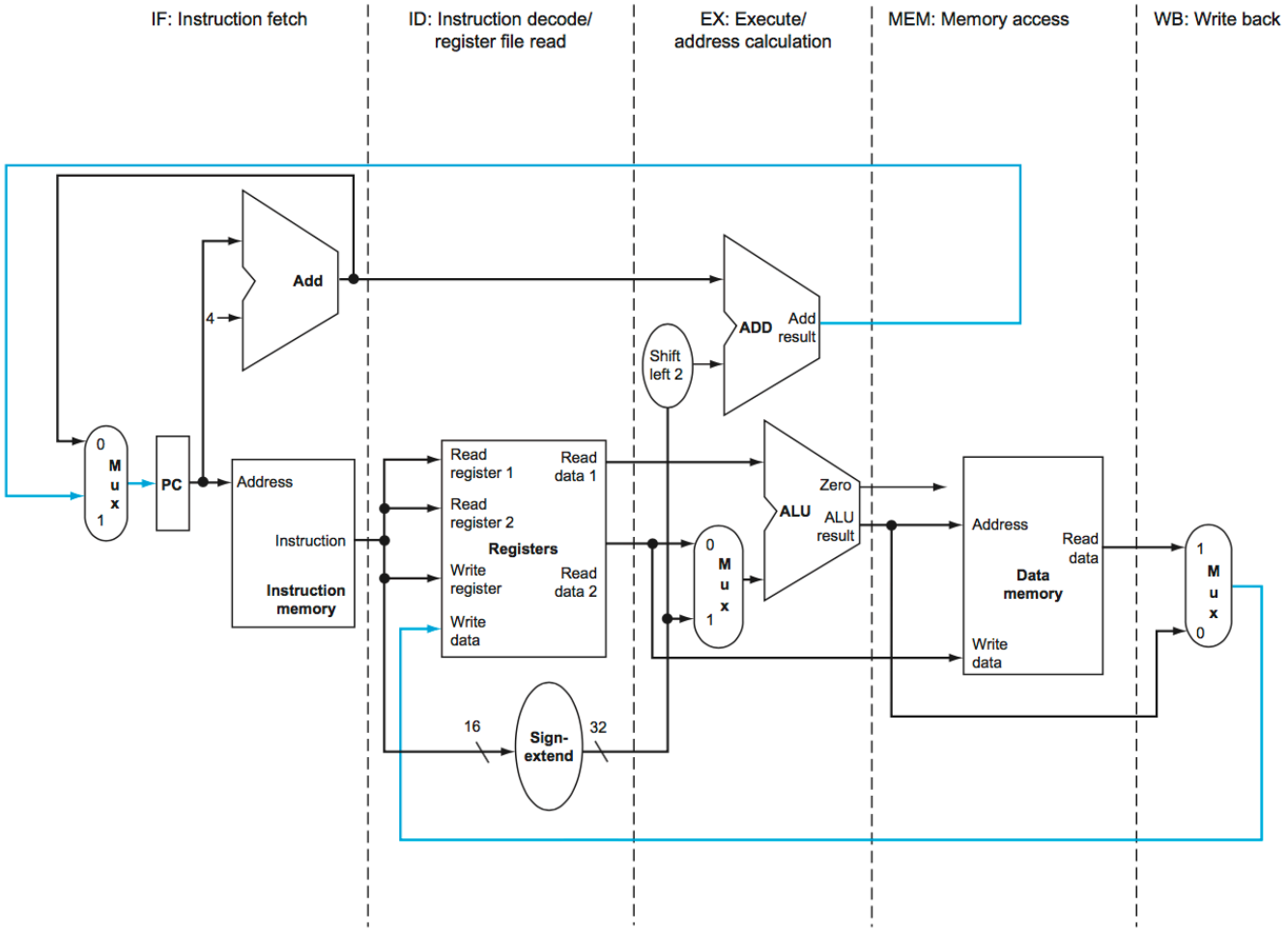




What happens inside the pipeline?

Control Hazards

- The outcome of the branch

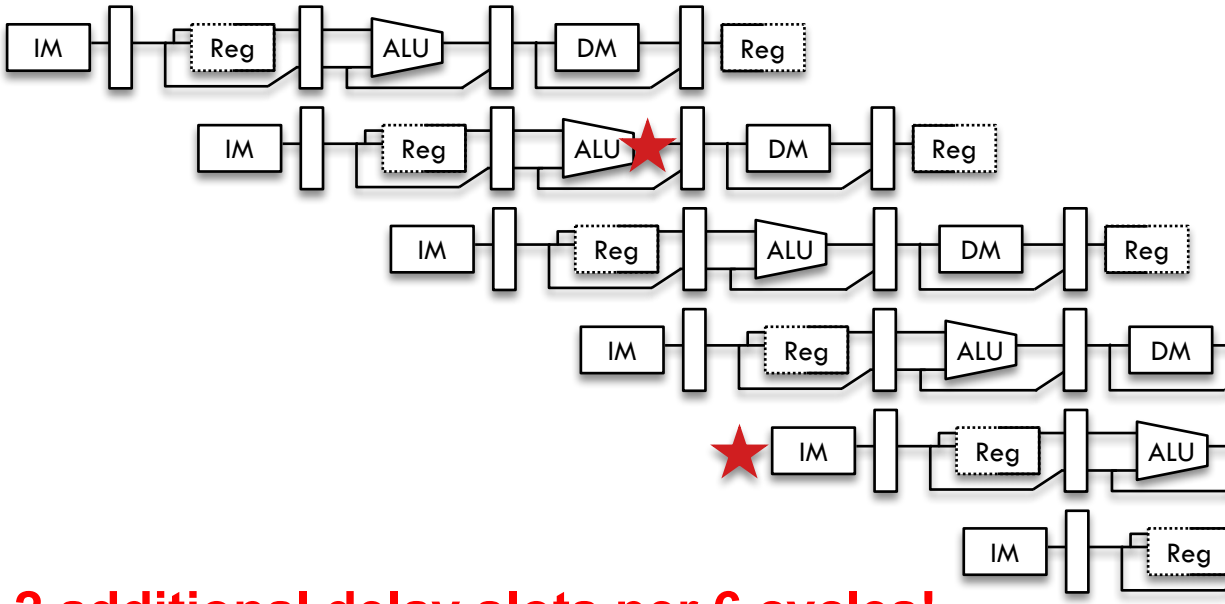


Handling Control Hazards

- 1. introducing stall cycles and delay slots
  - How many cycles/slots?
  - One branch per every six instructions on average!!

for:

```
addi $1, $0, 100
beq $0, $1, next
nothing
nothing
add $2, $2, $1
addi $1, $1, -1
J for
```



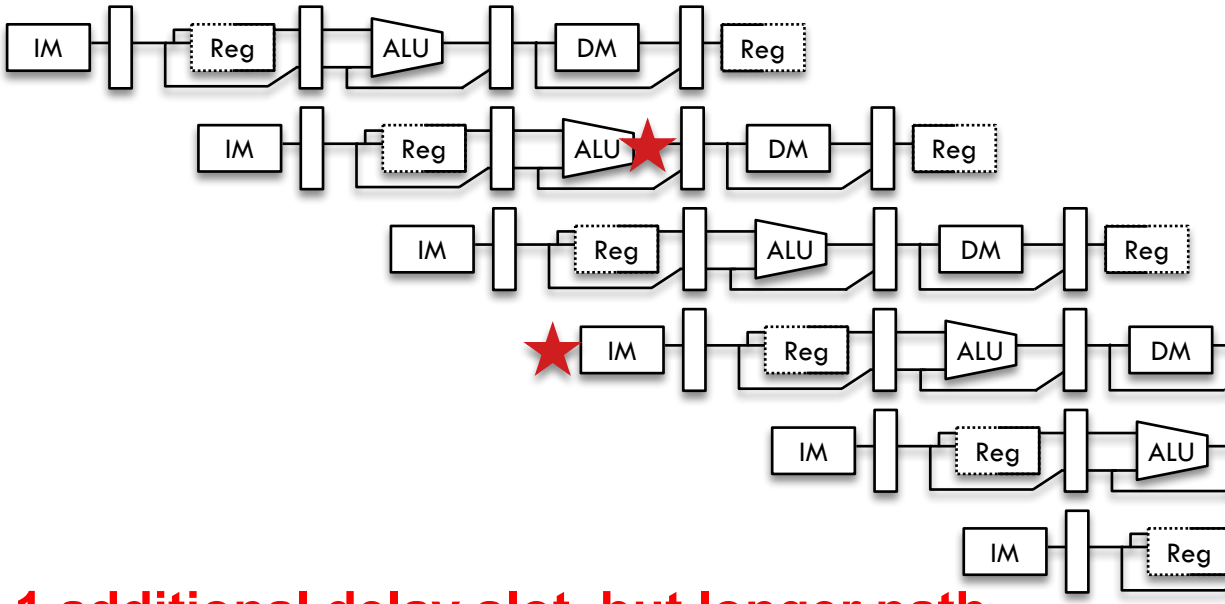
2 additional delay slots per 6 cycles!

Handling Control Hazards

- 1. introducing stall cycles and delay slots
  - How many cycles/slots?
  - One branch per every six instructions on average!!

for:

```
addi $1, $0, 100
beq $0, $1, next
nothing
add $2, $2, $1
addi $1, $1, -1
J for
nothing
```

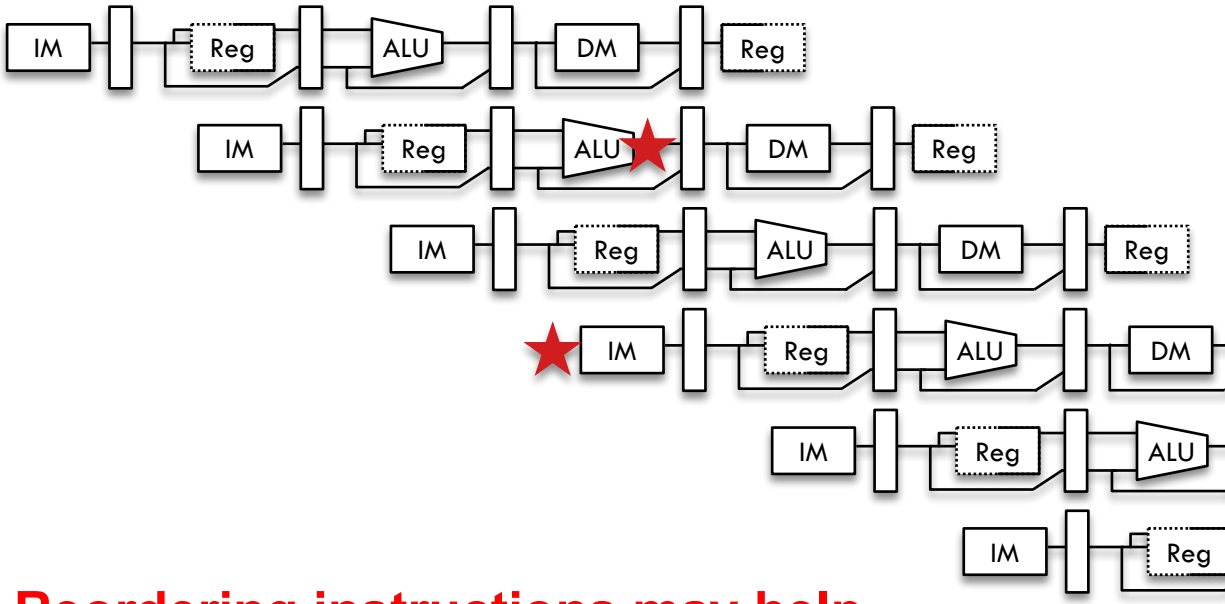


1 additional delay slot, but longer path

Handling Control Hazards

- 1. introducing stall cycles and delay slots
  - How many cycles/slots?
  - One branch per every six instructions on average!!

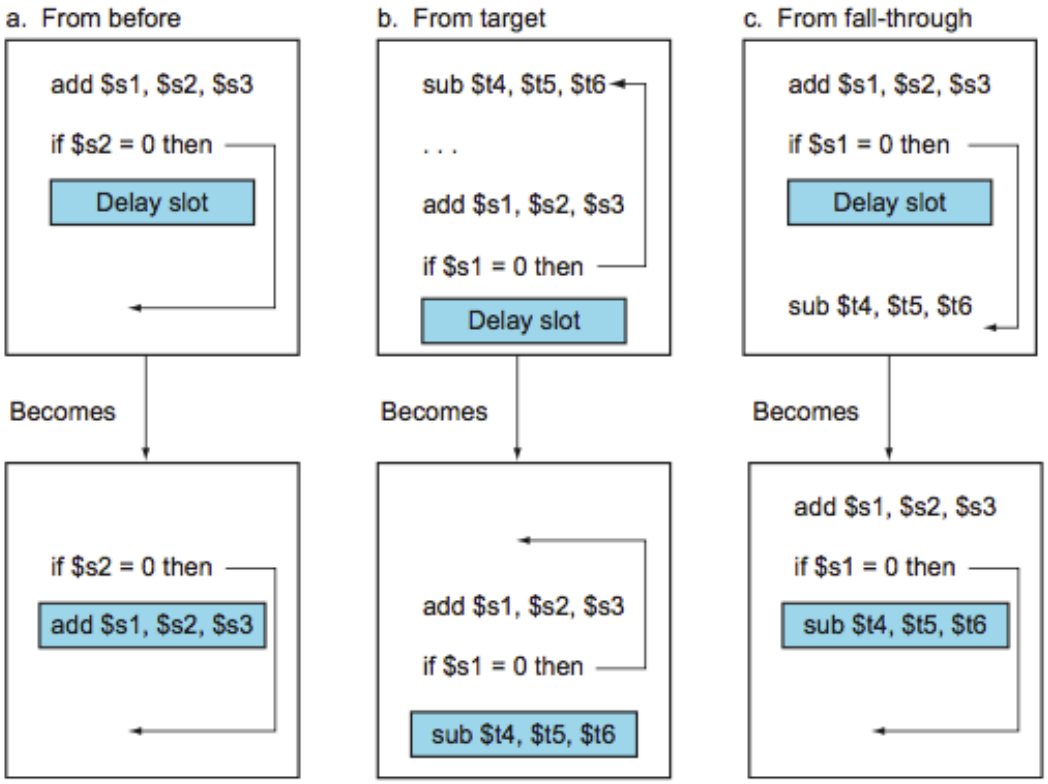
```
        addi $1, $0, 100
for:    beq $0, $1, next
        nothing
        add $2, $2, $1
        J for
        addi $1, $1, -1
next:   add r3, r3, r2
```



Reordering instructions may help

Handling Control Hazards

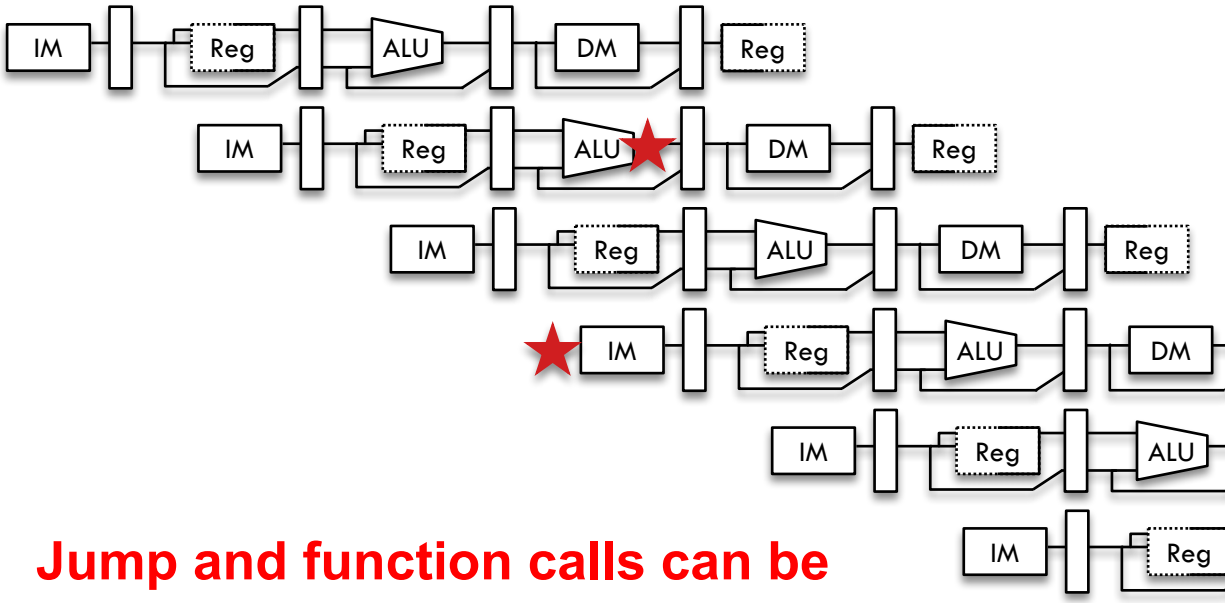
- Strategies for filling up the branch delay slot
  - (a) is the best choice; what about (b) and (c)?



Handling Control Hazards

- 1. introducing stall cycles and delay slots
  - How many cycles/slots?
  - One branch per every six instructions on average!!

```
        addi $1, $0, 100
for:    beq $0, $1, next
        nothing
        add $2, $2, $1
        J for
        addi $1, $1, -1
next:   add r3, r3, r2
```

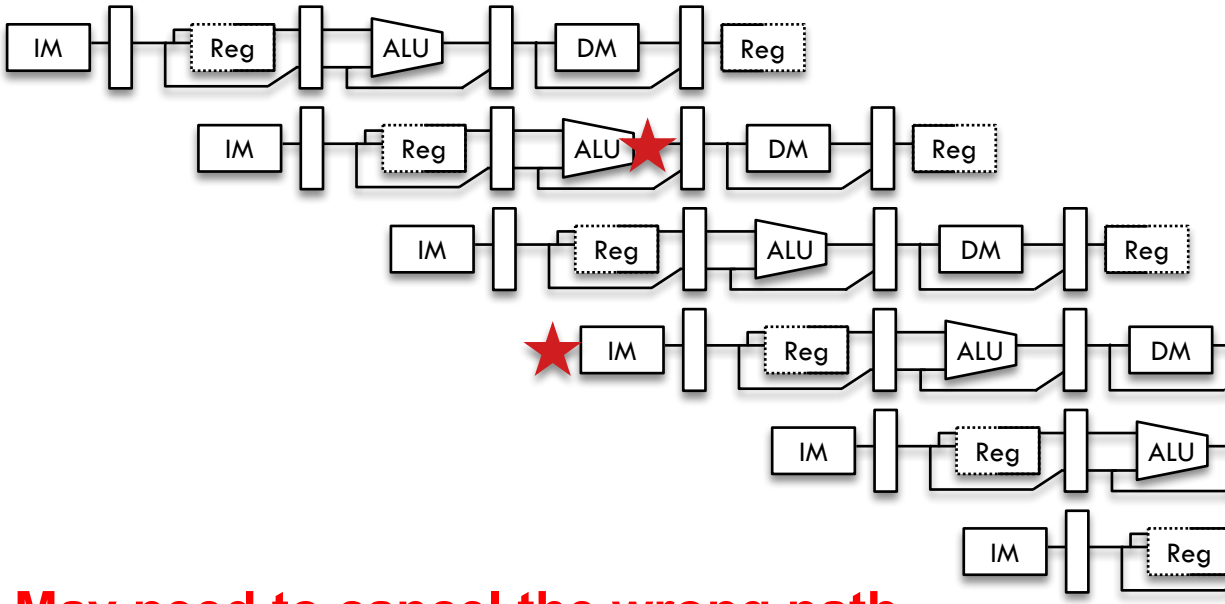


Jump and function calls can be resolved in the decode stage.

Handling Control Hazards

- 1. introducing stall cycles and delay slots
- 2. predict the branch outcome
  - simply assume the branch is taken or not taken
  - predict the next PC

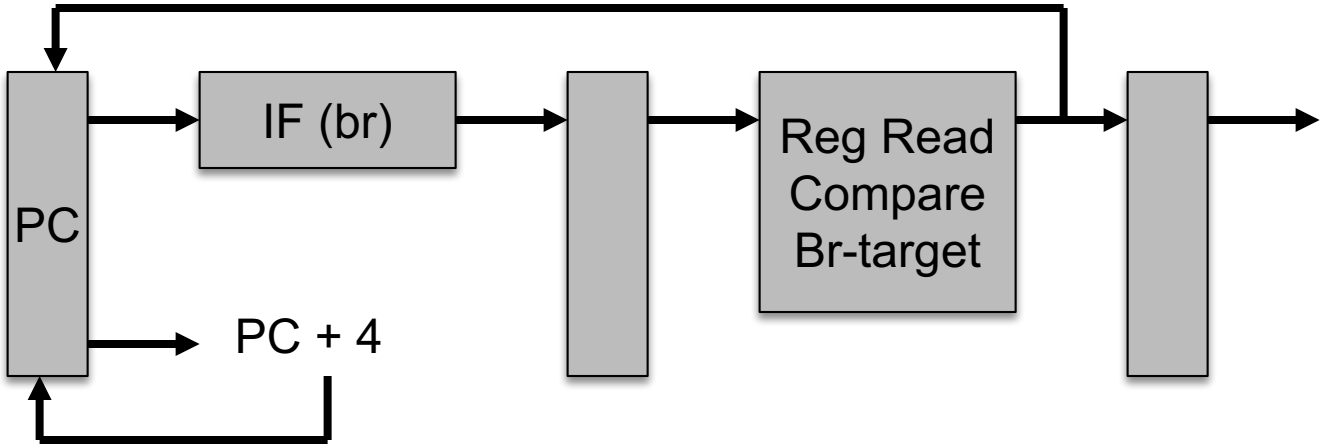
```
        addi $1, $0, 100
for:    beq $0, $1, next
        add $2, $2, $1
        addi $1, $1, -1
        J for
next:   add r3, r3, r2
```



May need to cancel the wrong path

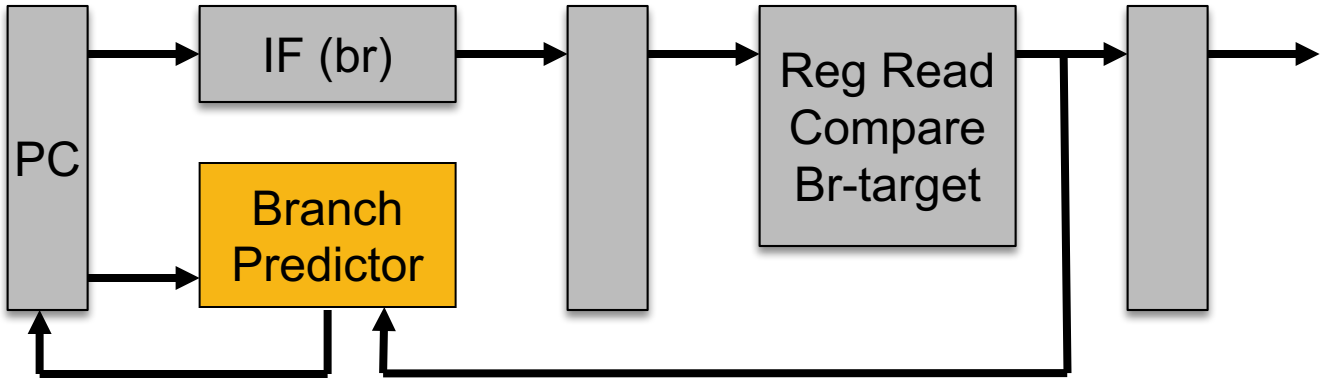
Handling Control Hazards

- Pipeline without branch predictor



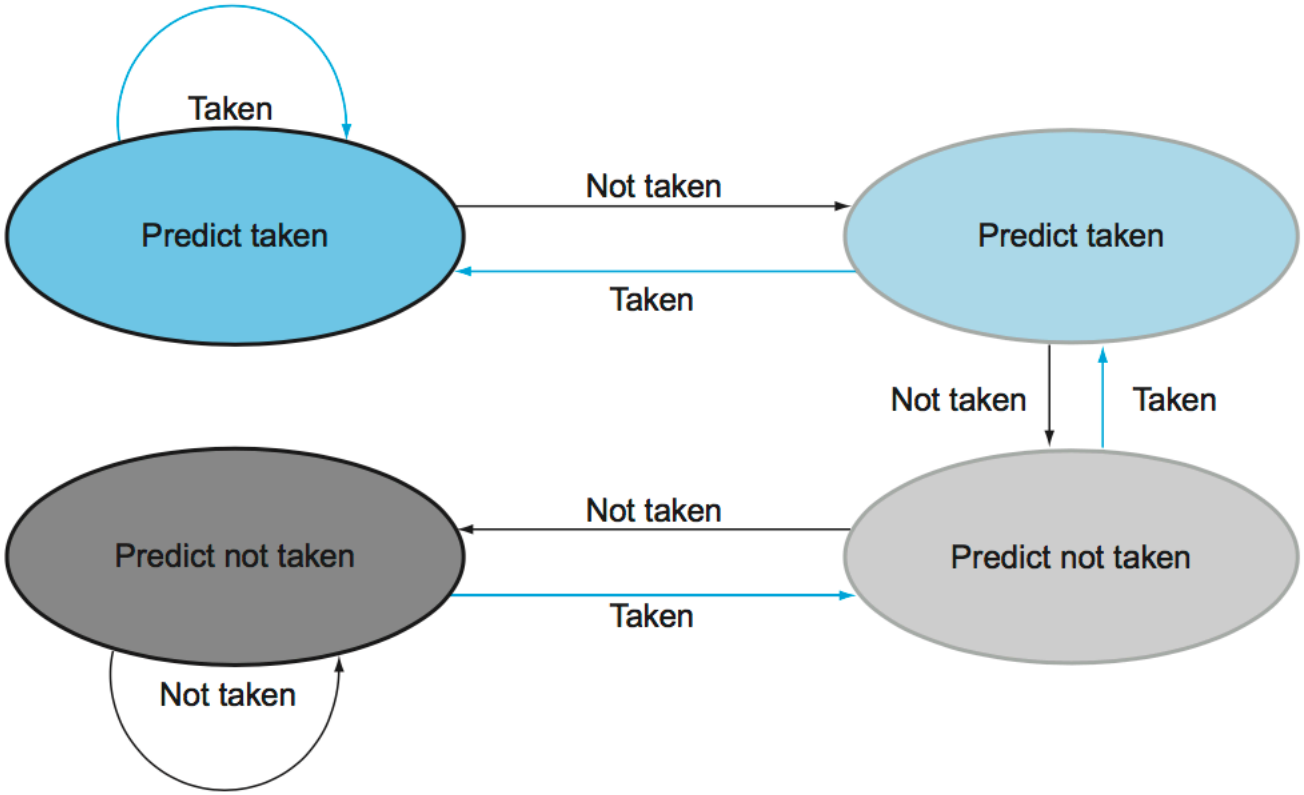
Handling Control Hazards

- Pipeline with branch predictor

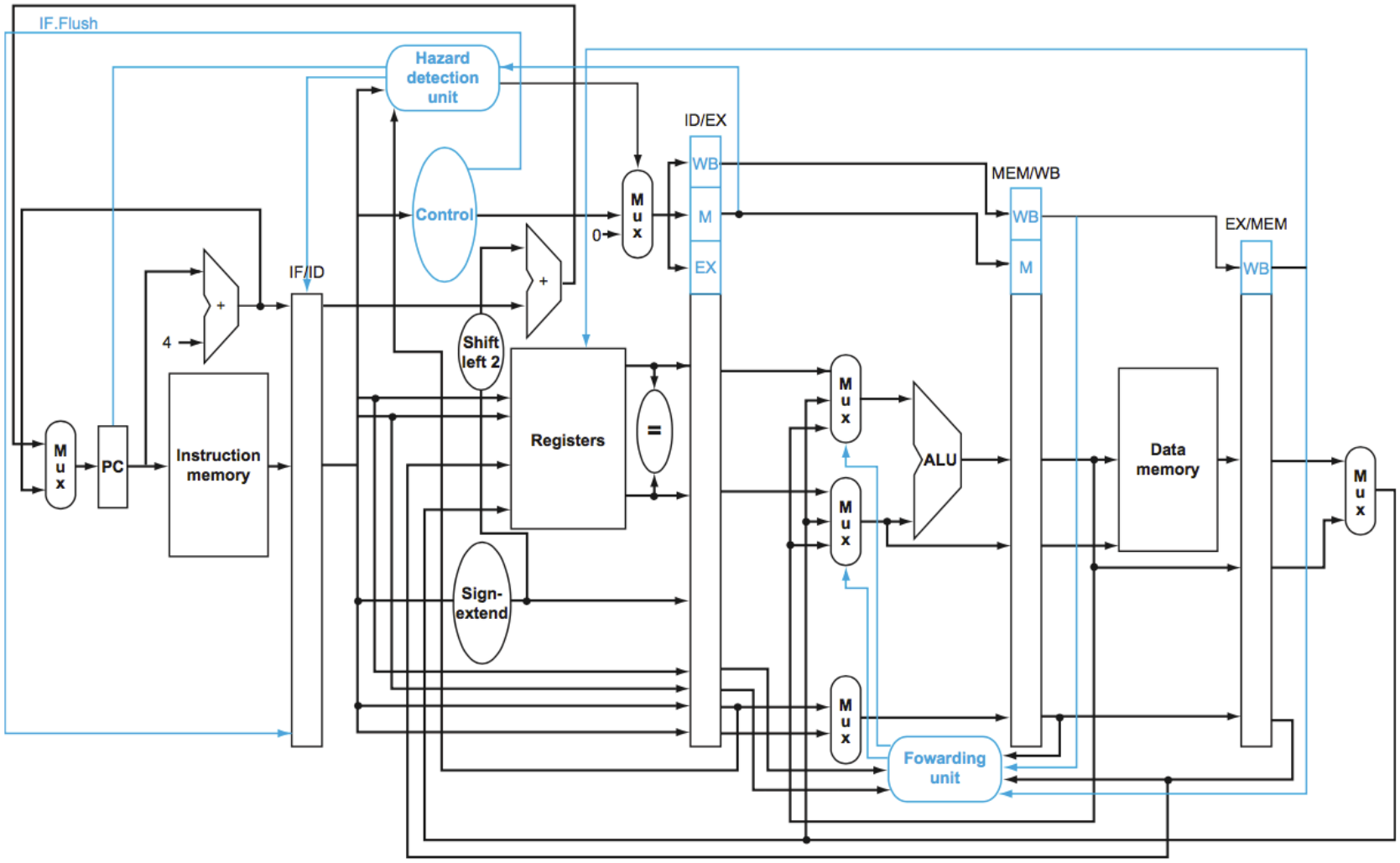


Handling Control Hazards

- The 2-bit branch predictor



Summary of the Pipeline

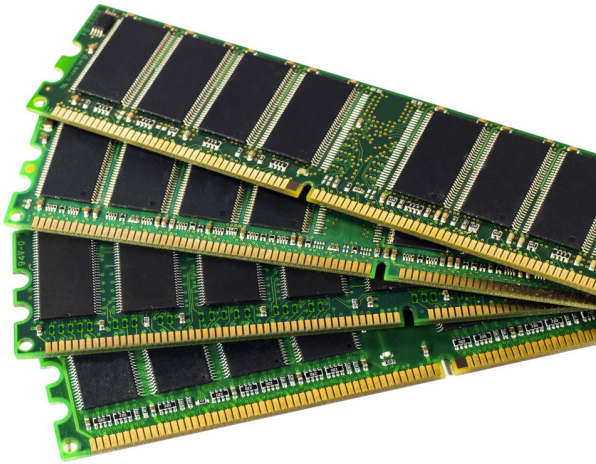


Memory System

- Data and instructions are stored on DRAM chips
  - DRAM has high bit density and low speed
  - An access DRAM may take about 300 processor cycles
- How to bridge the speed gap?

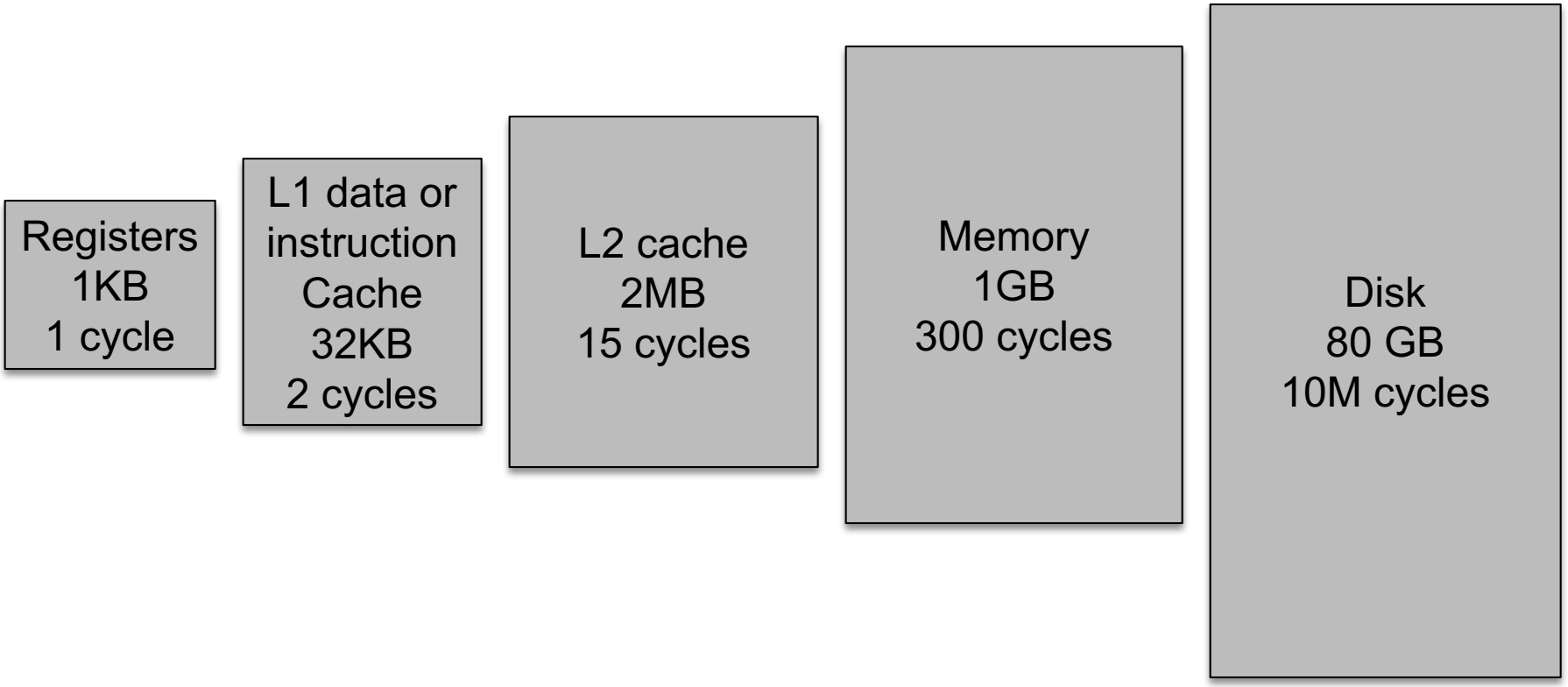
Processor **Memory**

~300X



Memory Hierarchy

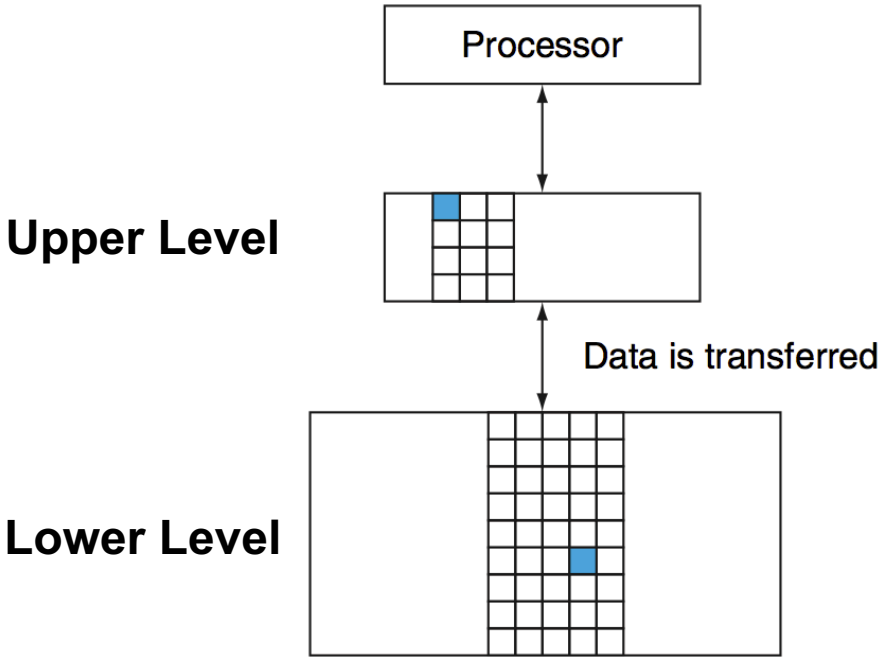
- The basic structure of a memory hierarchy.



Memory Hierarchy

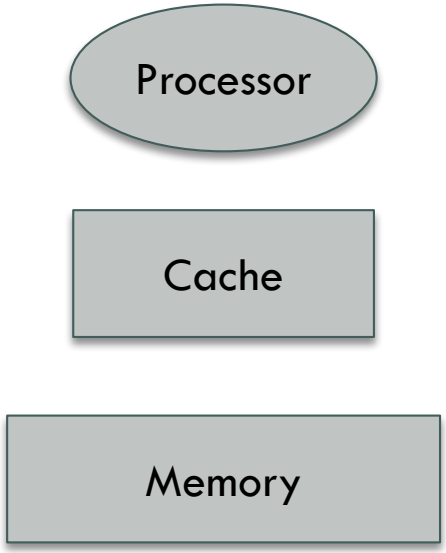
- The basic structure of a memory hierarchy.
- Multiple levels of the memory

Idea: keep important data closer to processor.



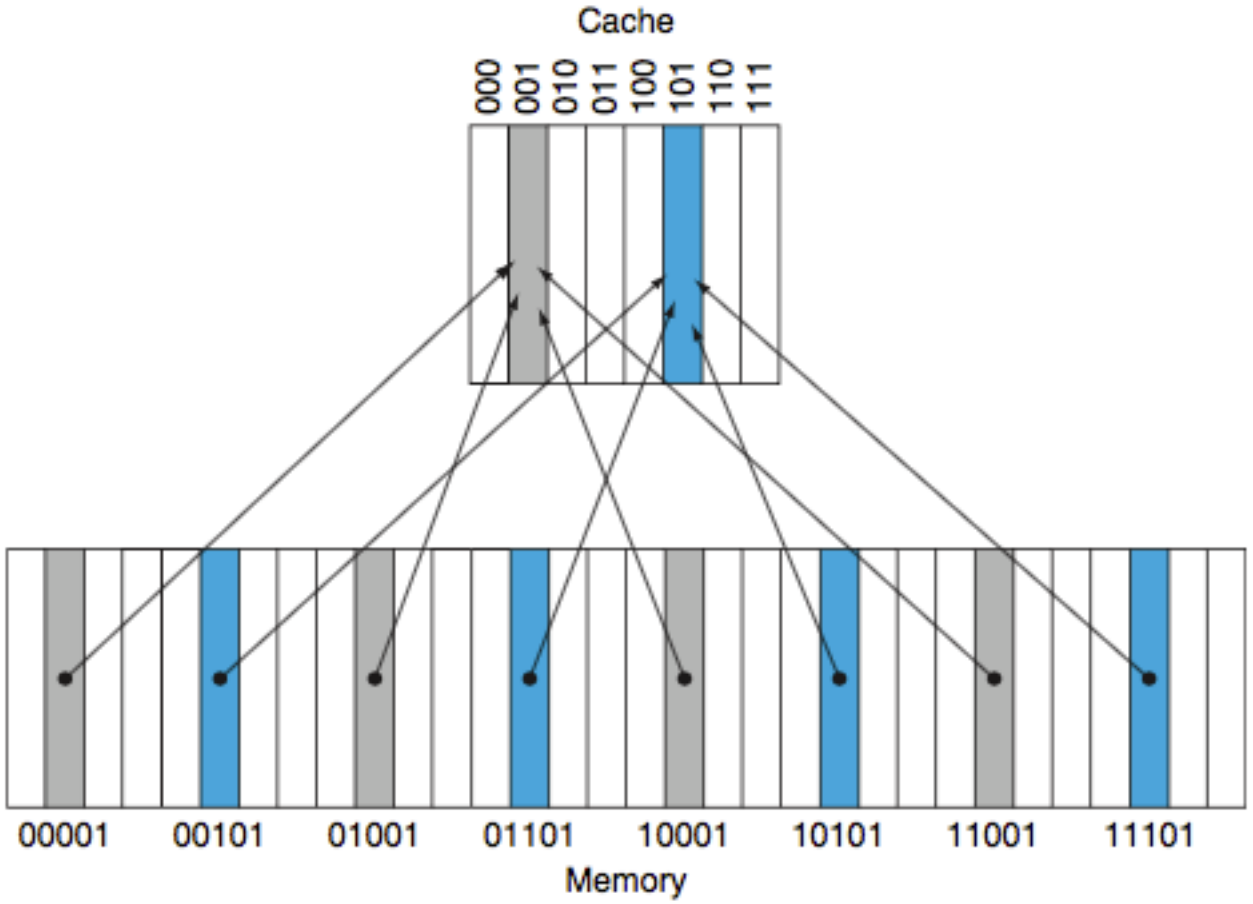
Cache Architecture

- Design principles
  - Temporal locality: if you used some data recently, you will likely use it again
  - Spatial locality: if you used some data recently, you will likely access its neighbors
- Cache terminology
  - Access time
  - Hit vs. miss
  - Miss penalty



Direct-Mapped Cache

- Cache address



Direct-Mapped Cache

- Cache lookup