

Chapter 4

The Processor

Fall 2018

Soontae Kim

School of Computing, KAIST

Announcement

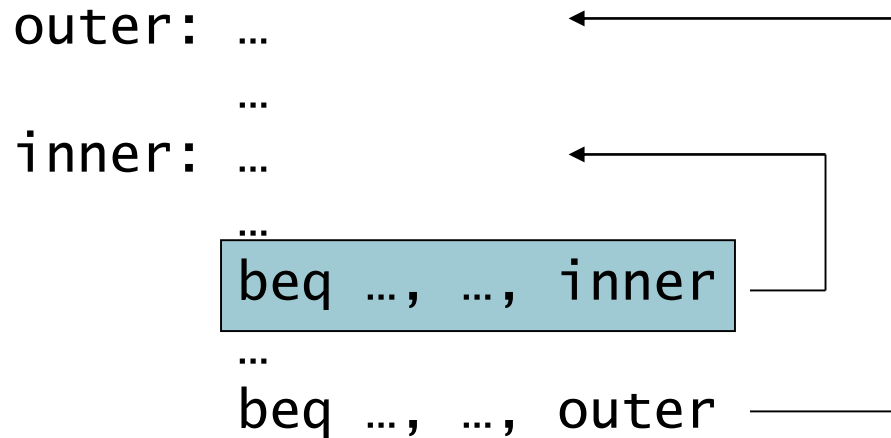
- Project #2
 - Due on Nov. 9
 - Studying pipelining with simulator

Dynamic Branch Prediction

- In deeper and superscalar pipelines, branch penalty is more significant
- Use dynamic prediction
 - Branch prediction buffer (aka branch history table)
 - Indexed by recent branch instruction addresses
 - Stores outcome (taken/not taken)
 - To execute a branch
 - Check table, expect the same outcome
 - Start fetching from fall-through or target
 - If wrong, flush pipeline and flip prediction

1-Bit Predictor: Shortcoming

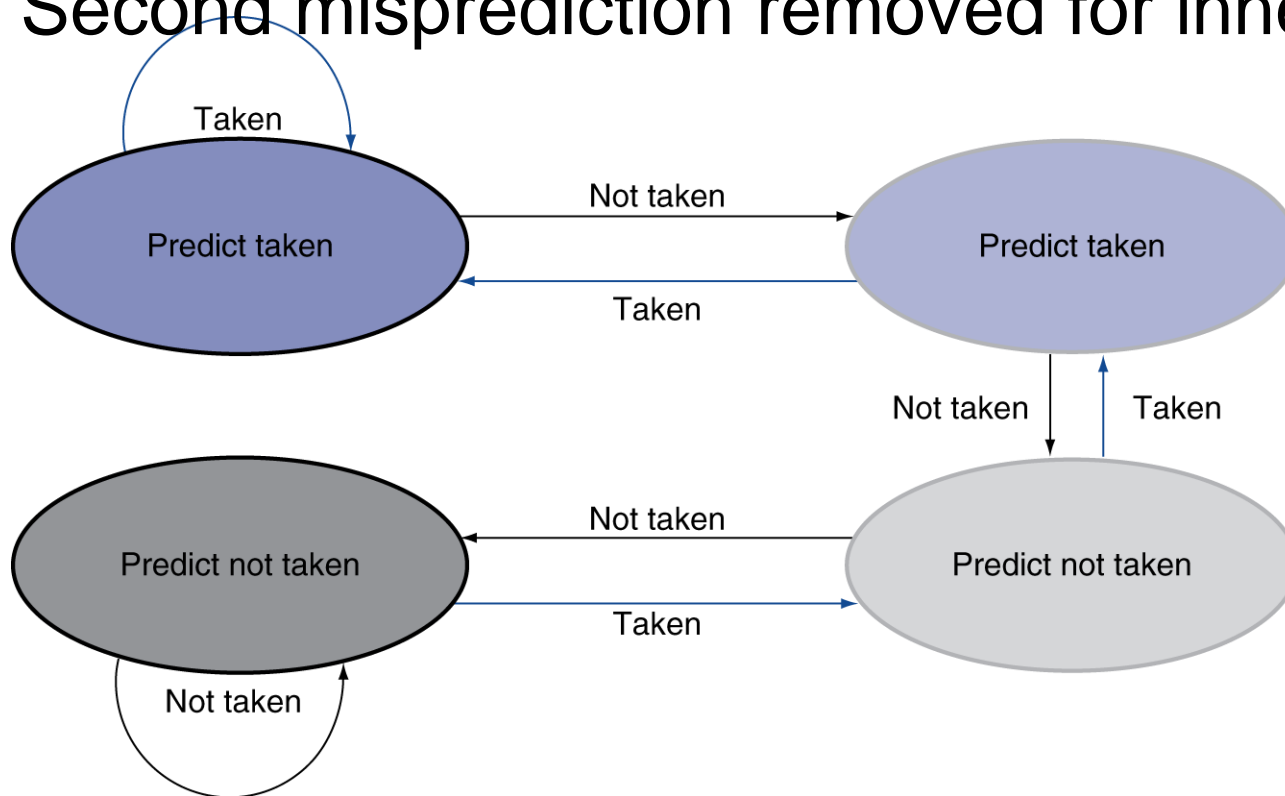
- Inner loop branches mispredicted twice!



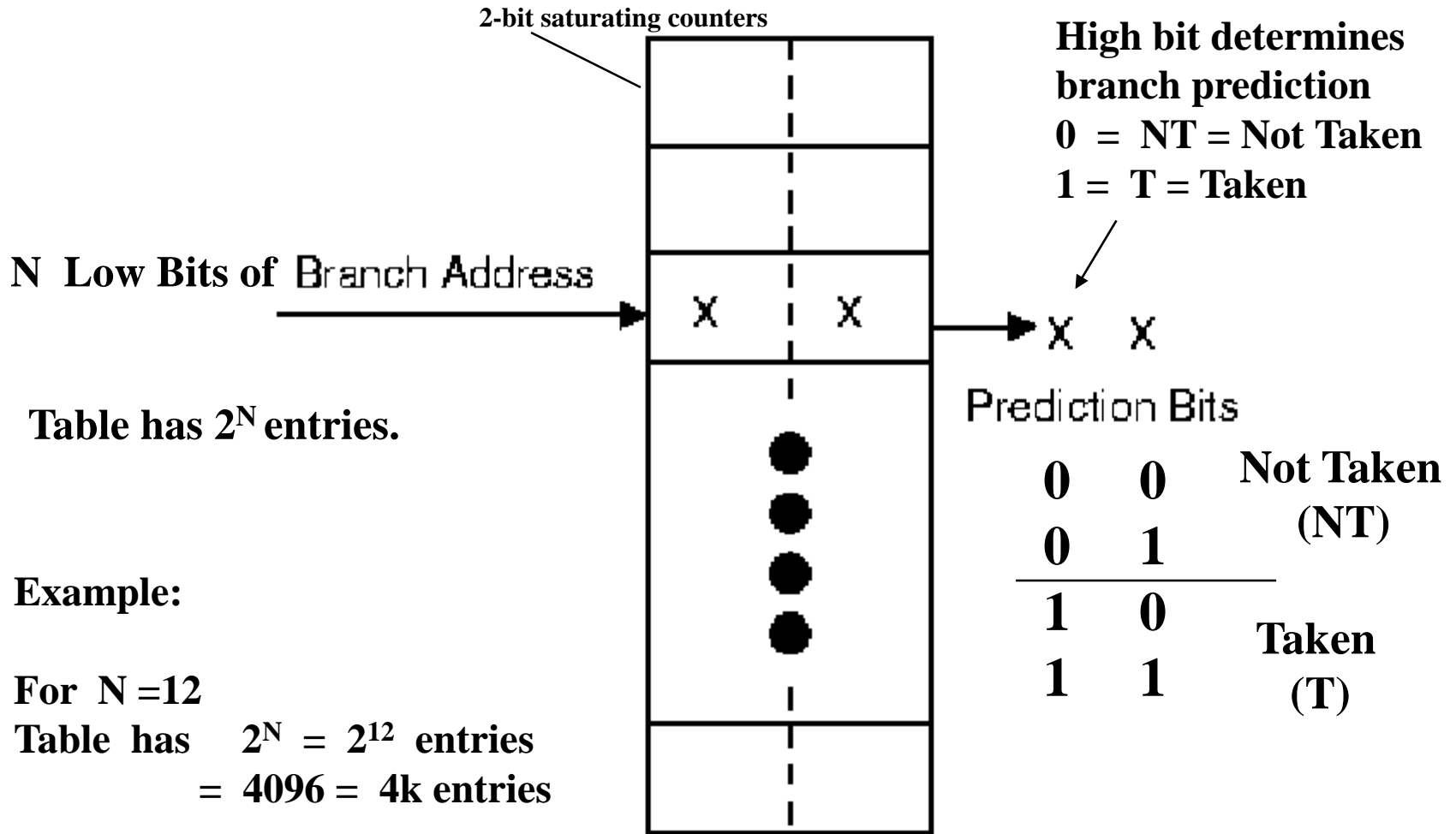
- Mispredict as taken on last iteration of inner loop
- Then mispredict as not taken on first iteration of inner loop next time

2-Bit Predictor

- Only change prediction on two successive mispredictions
 - Second misprediction removed for inner loop



One-Level Bimodal Branch Predictors



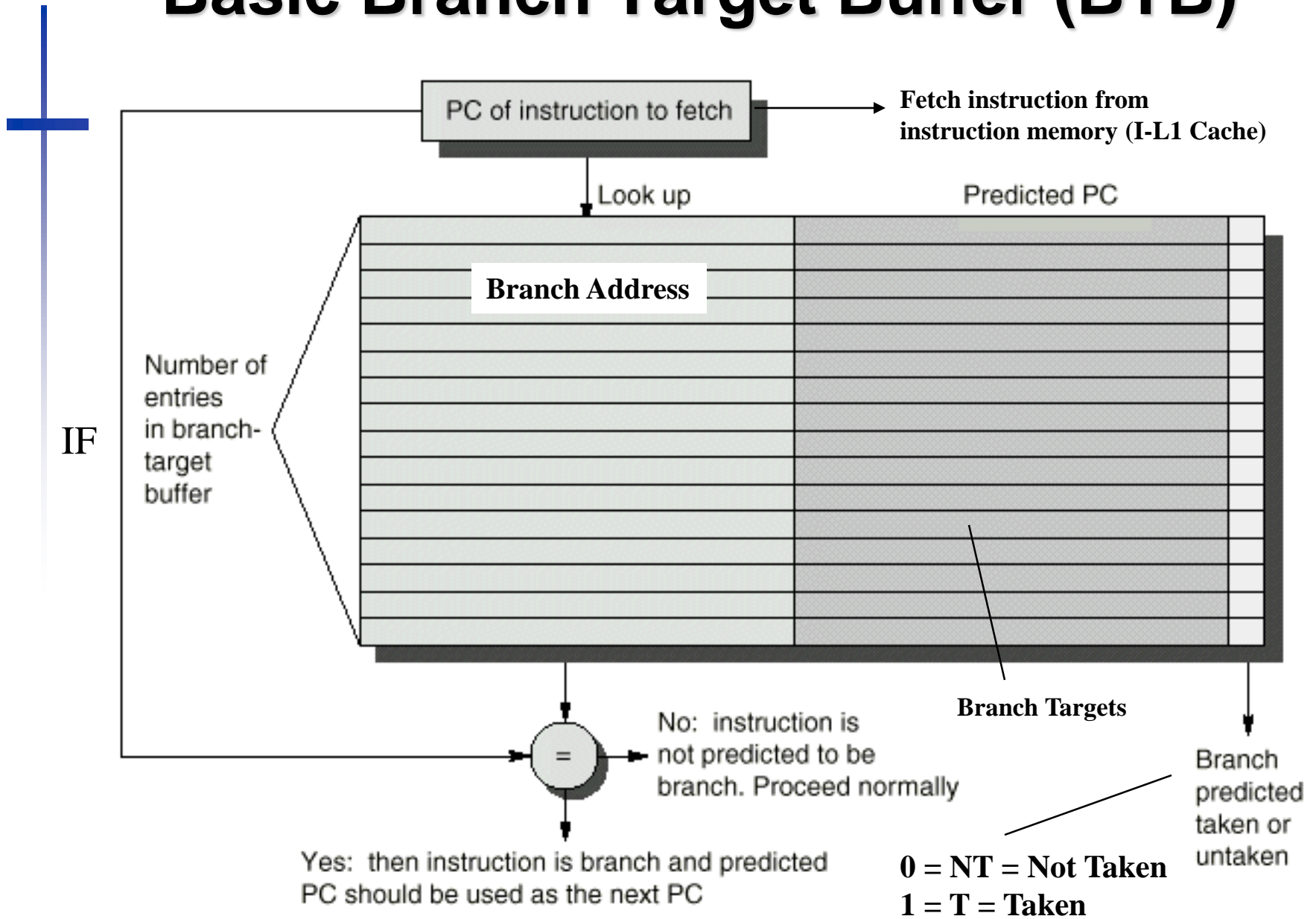
Number of bits needed = $2 \times 4k = 8k$ bits

Common one-level implementation

Calculating branch target addresses

- Even with predictor, still need to calculate the target address
 - 1-cycle penalty for a taken branch
- Branch target buffer (BTB)
 - Cache of branch target addresses
 - Indexed by PC when instruction fetched
 - If hit and instruction is branch predicted taken, can fetch target immediately

Basic Branch Target Buffer (BTB)



A branch-target buffer.

Exceptions and Interrupts

- “Unexpected” events requiring change in flow of control
 - Different ISAs use the terms differently
- Exception
 - Arises within the CPU
 - e.g., undefined opcode, overflow, syscall, ...
- Interrupt
 - From an external I/O controller
- Dealing with them without sacrificing performance is hard

Handling Exceptions

- In MIPS, exceptions managed by a System Control Coprocessor (CP0)
- Save PC of offending (or interrupted) instruction
 - In MIPS: Exception Program Counter (EPC)
- Save indication of the problem
 - In MIPS: Cause register
 - We'll assume 1-bit
 - 0 for undefined opcode, 1 for overflow
- Jump to handler at 8000 0180

An Alternate Mechanism

- Vectored Interrupts
 - Handler address determined by the cause
- Example:
 - Undefined opcode: C000 0000
 - Overflow: C000 0020
 -: C000 0040
- Instructions either
 - Deal with the interrupt, or
 - Jump to real handler

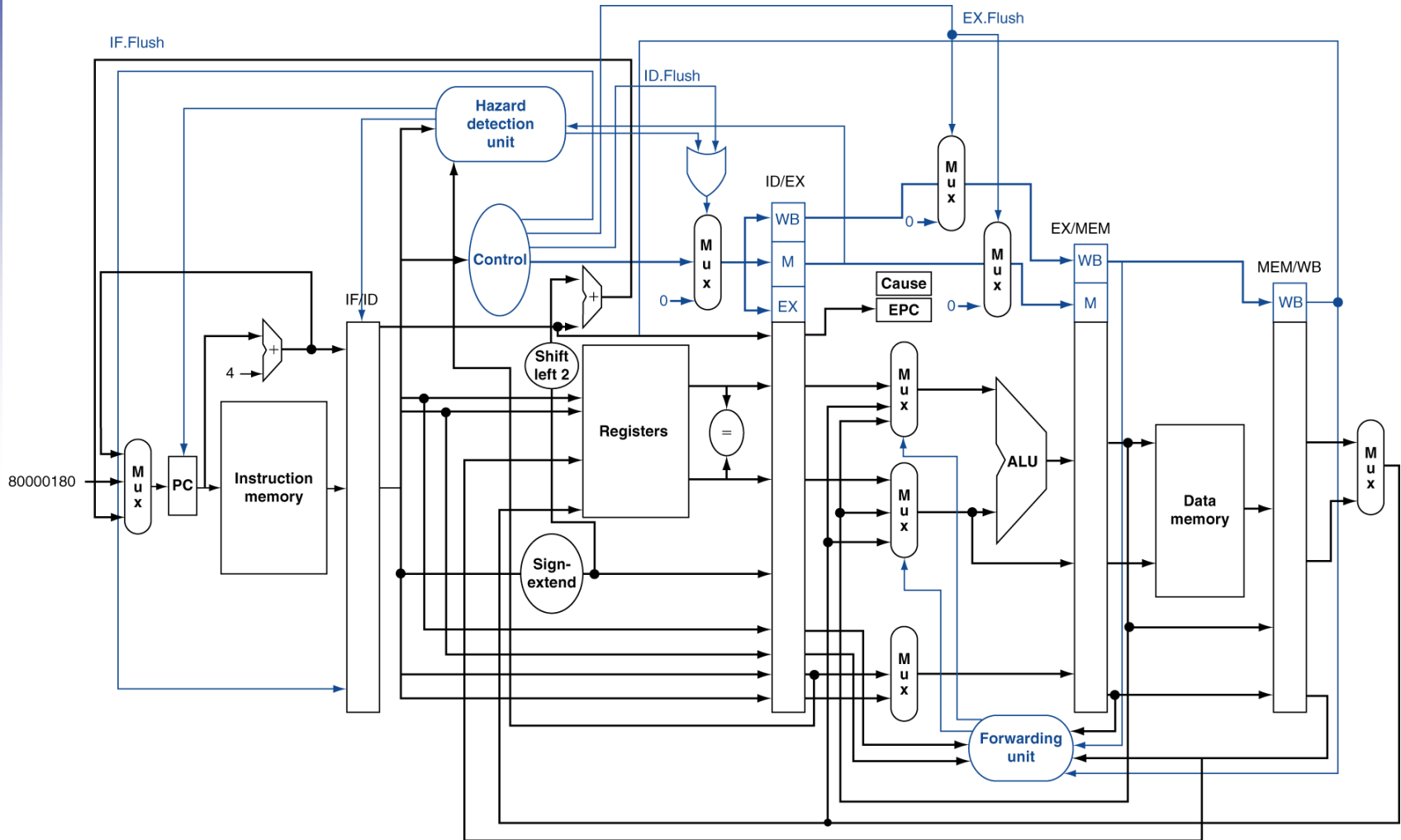
Handler Actions

- Read cause, and transfer to relevant handler
- Determine action required
- If restartable
 - Take corrective action
 - use EPC to return to program
- Otherwise
 - Terminate program
 - Report error using EPC, cause, ...

Exceptions in a Pipeline

- Another form of control hazard
- Consider overflow on add in EX stage
 - add \$1, \$2, \$1
 - Prevent \$1 from being written
 - Complete previous instructions
 - Flush add and subsequent instructions
 - Set Cause and EPC register values
 - Transfer control to handler
- Similar to mispredicted branch
 - Use much of the same hardware

Pipeline with Exceptions



Exception Properties

- Restartable exceptions
 - Pipeline can flush the instruction
 - Handler executes, then returns to the instruction
 - Refetched and executed from scratch
- PC saved in EPC register
 - Identifies causing instruction
 - Actually PC + 4 is saved
 - Handler must adjust

Exception Example

- Exception on `add` in

```
40      sub    $11, $2, $4
44      and    $12, $2, $5
48      or     $13, $2, $6
4C      add    $1,  $2, $1
50      slt   $15, $6, $7
54      lw    $16, 50($7)
```

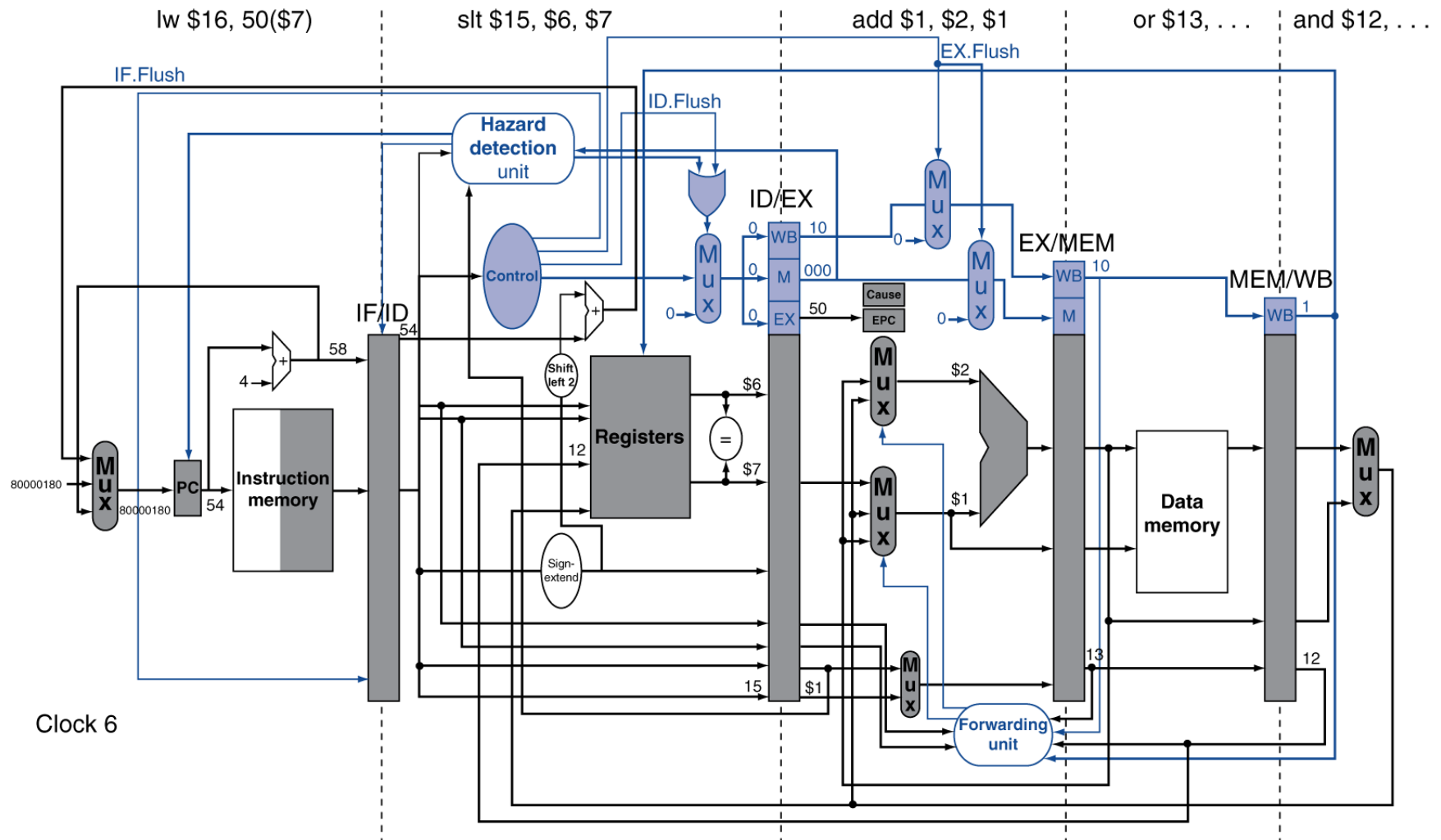
...

- Handler

```
80000180      sw    $25, 1000($0)
80000184      sw    $26, 1004($0)
```

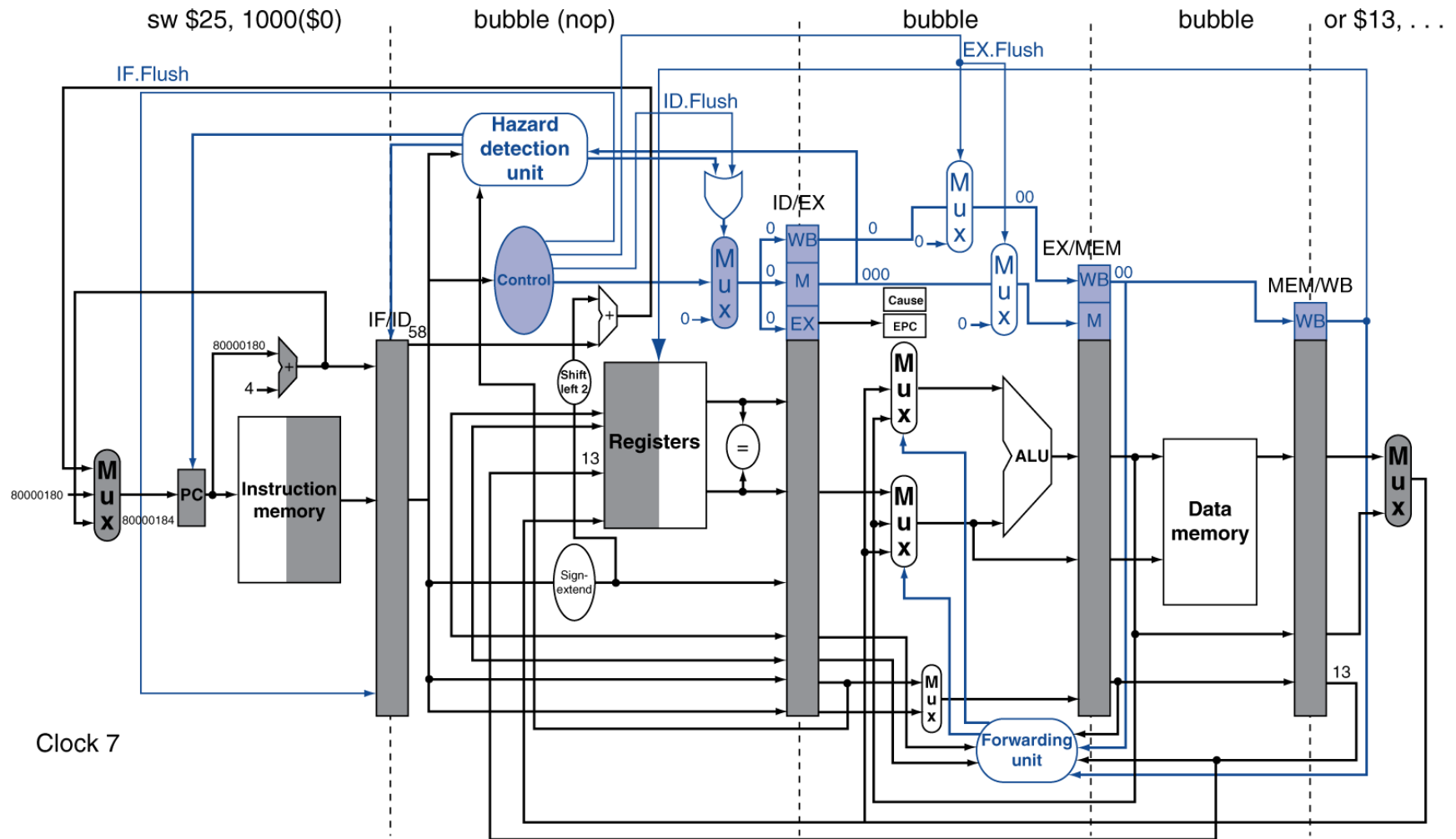
...

Exception Example



Clock 6

Exception Example



Multiple Exceptions

- Pipelining overlaps multiple instructions
 - Could have multiple exceptions at once
- Simple approach: deal with exception from earliest instruction
 - Flush subsequent instructions
 - “Precise” exceptions
- In complex pipelines
 - Multiple instructions issued per cycle
 - Out-of-order completion
 - Maintaining precise exceptions is difficult!