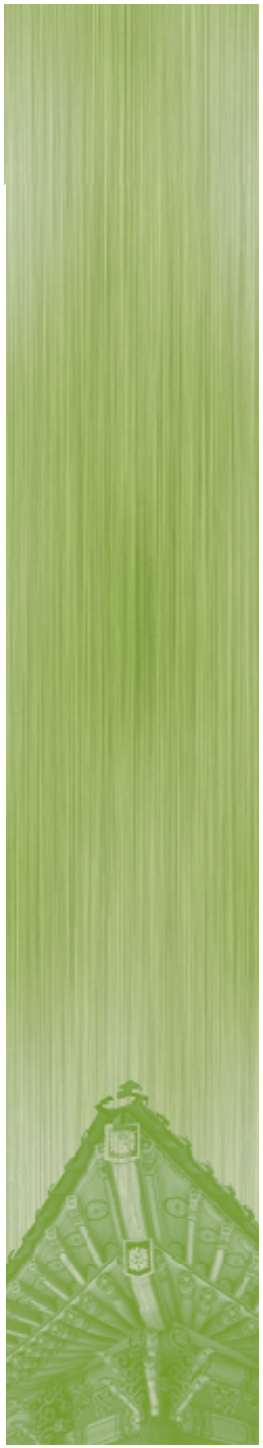


Exceptions

Jin-Soo Kim (jinsookim@skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu

SKK UNIVERSITY

Exceptions and Interrupts

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

Handling Exceptions (1)

- **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 `0x8000 0180`

Handling Exceptions (2)

- **An alternate mechanism**
  - Vectored interrupts
    - Handler address determined by the cause
  - Example:
    - Undefined opcode: `0xC000 0000`
    - Overflow: `0xC000 0020`
    - ...: `0xC000 0040`
  - Instructions either
    - Deal with the interrupt, or
    - Jump to real handler

Handling Exceptions (3)

- **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, …

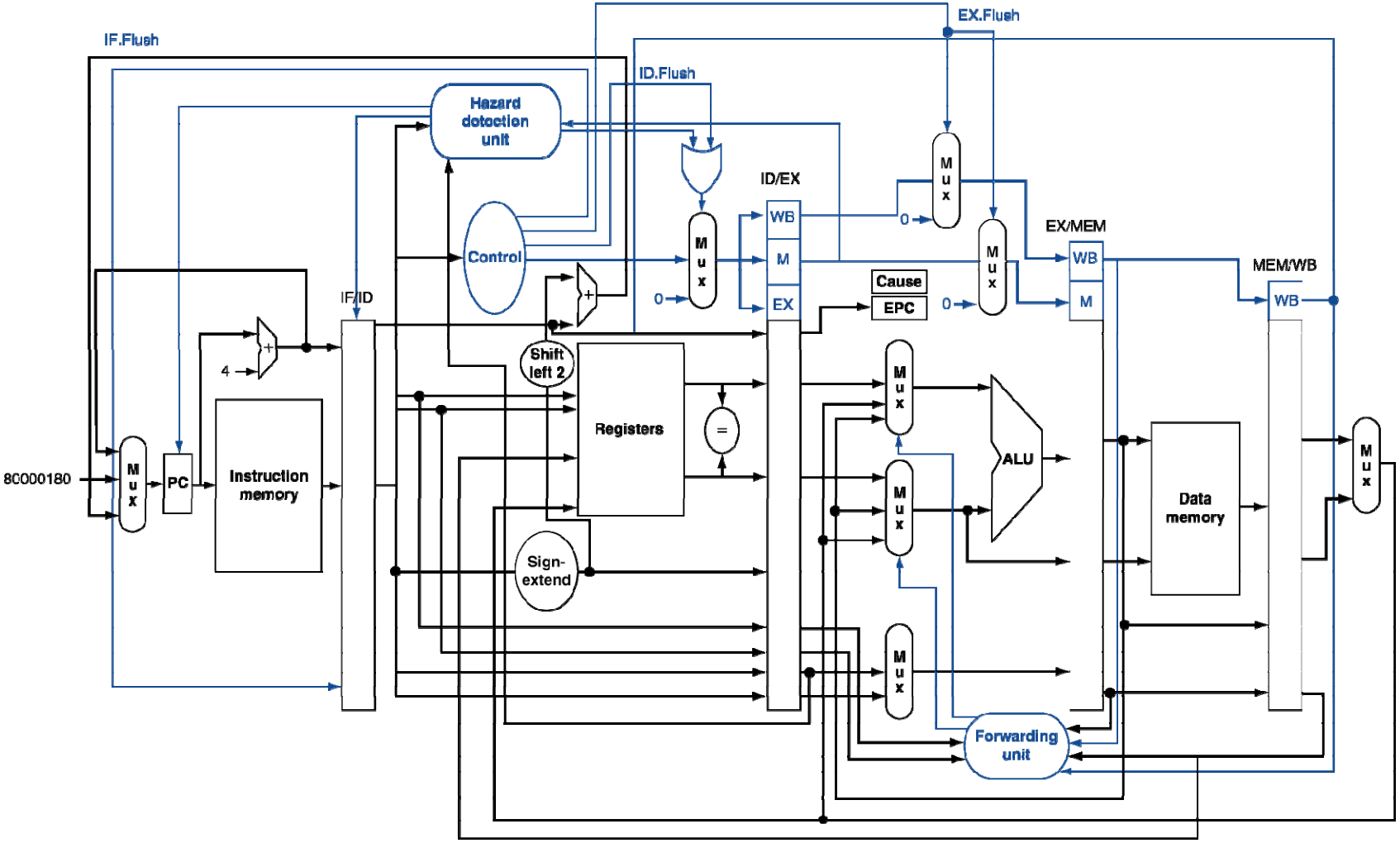
Exceptions in a Pipeline

- **Another form of control hazard**
  - Consider overflow on add in EX stage

```
add   $1, $2, $1
```

    - Prevent `$1` from being clobbered
    - 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 (1)

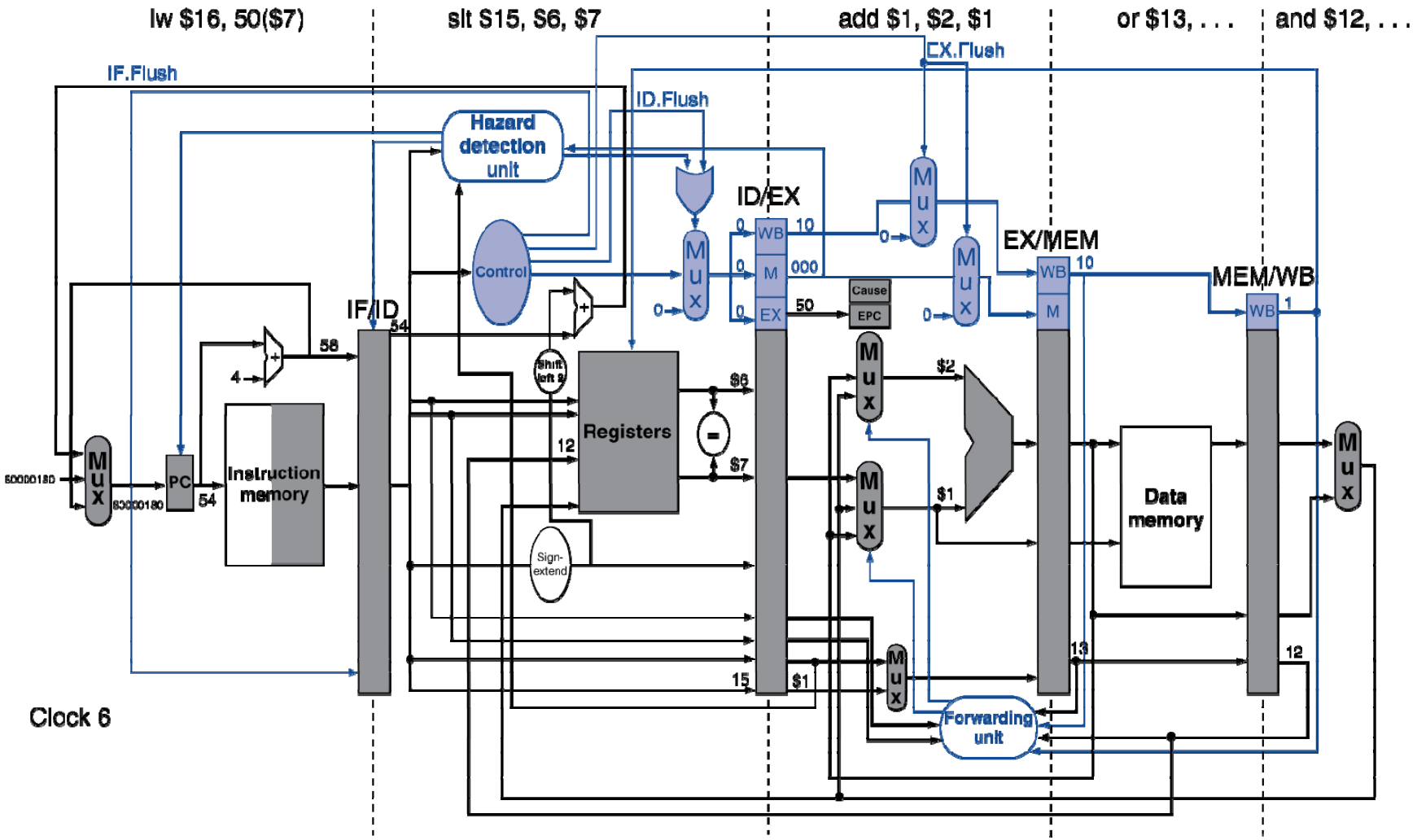
- Exception on add in

```
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)
      ...
```

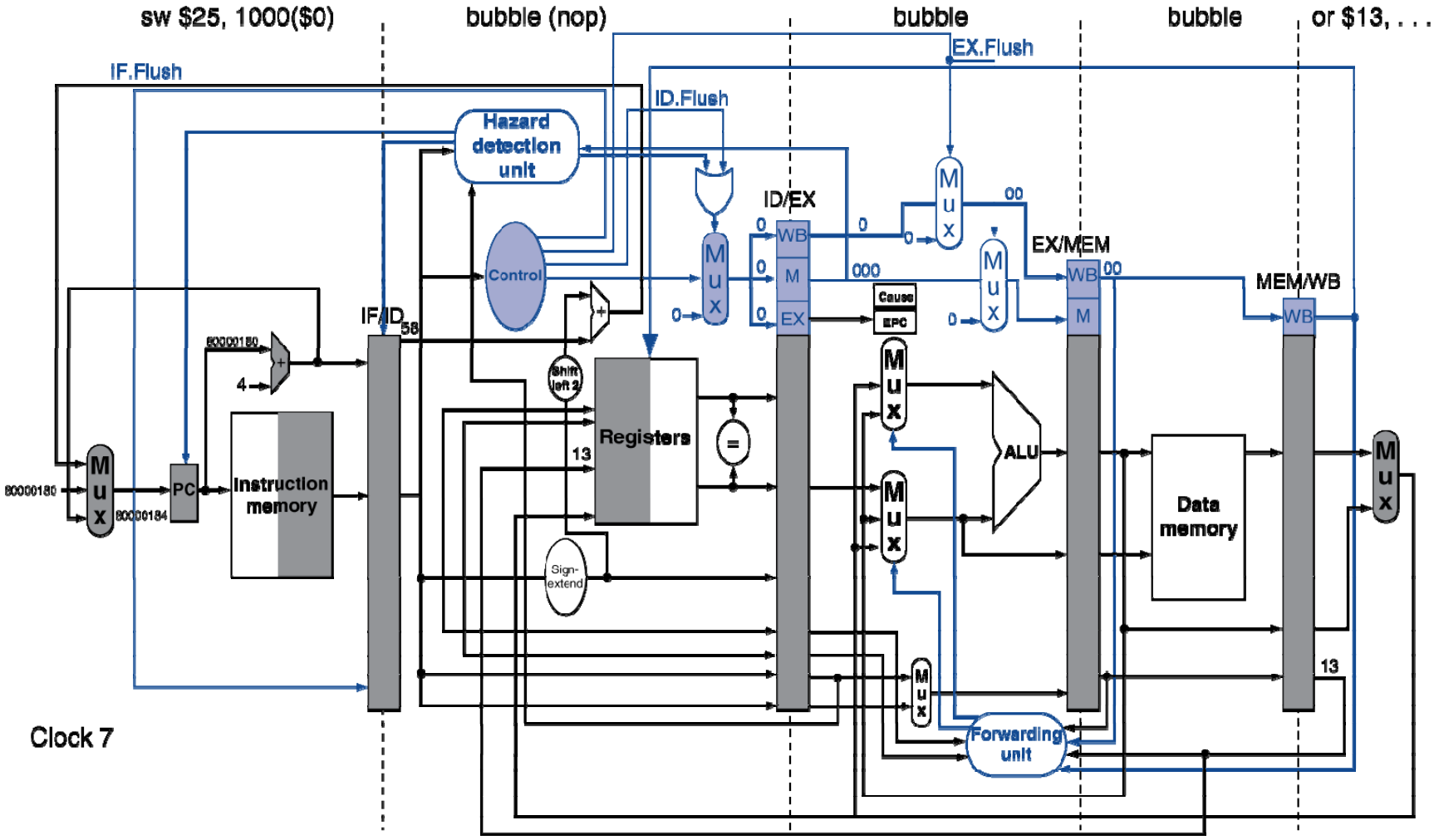
- Handler

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

Exception Example (2)



Exception Example (3)



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!

Imprecise Exceptions

- **Imprecise exceptions**
  - Just stop pipeline and save state
    - Including exception cause(s)
  - Let the handler work out
    - Which instruction(s) had exceptions
    - Which to complete or flush
      - » May require "manual" completion
  - Simplifies hardware, but more complex handler software
  - Not feasible for complex multiple-issue out-of-order pipelines