CS/ECE 552: Pipelining and Exceptions

Prof. Matthew D. Sinclair

Lecture notes based in part on slides created by Mark Hill, Mikko Lipasti, David Wood, Guri Sohi, John Shen and Jim Smith

Exceptions

- What happens?
  - Instruction fetch page fault
  - Illegal opcode
  - Privileged opcode
  - Arithmetic overflow
  - Data page fault
  - I/O device status change
  - Power-on/reset

Exceptions

- For some, we could test for the condition
  - Arithmetic overflow
  - I/O device ready (polling)
- But most tests uselessly say “no”
- Solution:
  - Surprise “procedure call”
  - Called an exception

Exceptions: Big Picture

- Two types:
  - Interrupt (asynchronous) or
  - Trap (synchronous)
- Hardware handles initial reaction
- Then invokes a software exception handler
  - By convention, at e.g., 0xC00
  - O/S kernel provides code at the handler address

Exceptions: Hardware

- Sets state that identifies cause of exception
  - MIPS: in exception_code field of Cause register
- Changes to kernel mode for dangerous work ahead
- Disables interrupts
  - MIPS: recorded in status register
- Saves current PC (MIPS: exception PC)
- Jumps to specific address (MIPS: 0x80000080)
  - Like a surprise JAL – so can’t clobber $31

Exceptions: Software

- Exception handler:
  - MIPS: .ktext at 0x80000080
- Set flag to detect incorrect entry
  - Nested exception while in handler
- Save some registers
- Find exception type
  - E.g., I/O interrupt or syscall
- Jump to specific exception handler

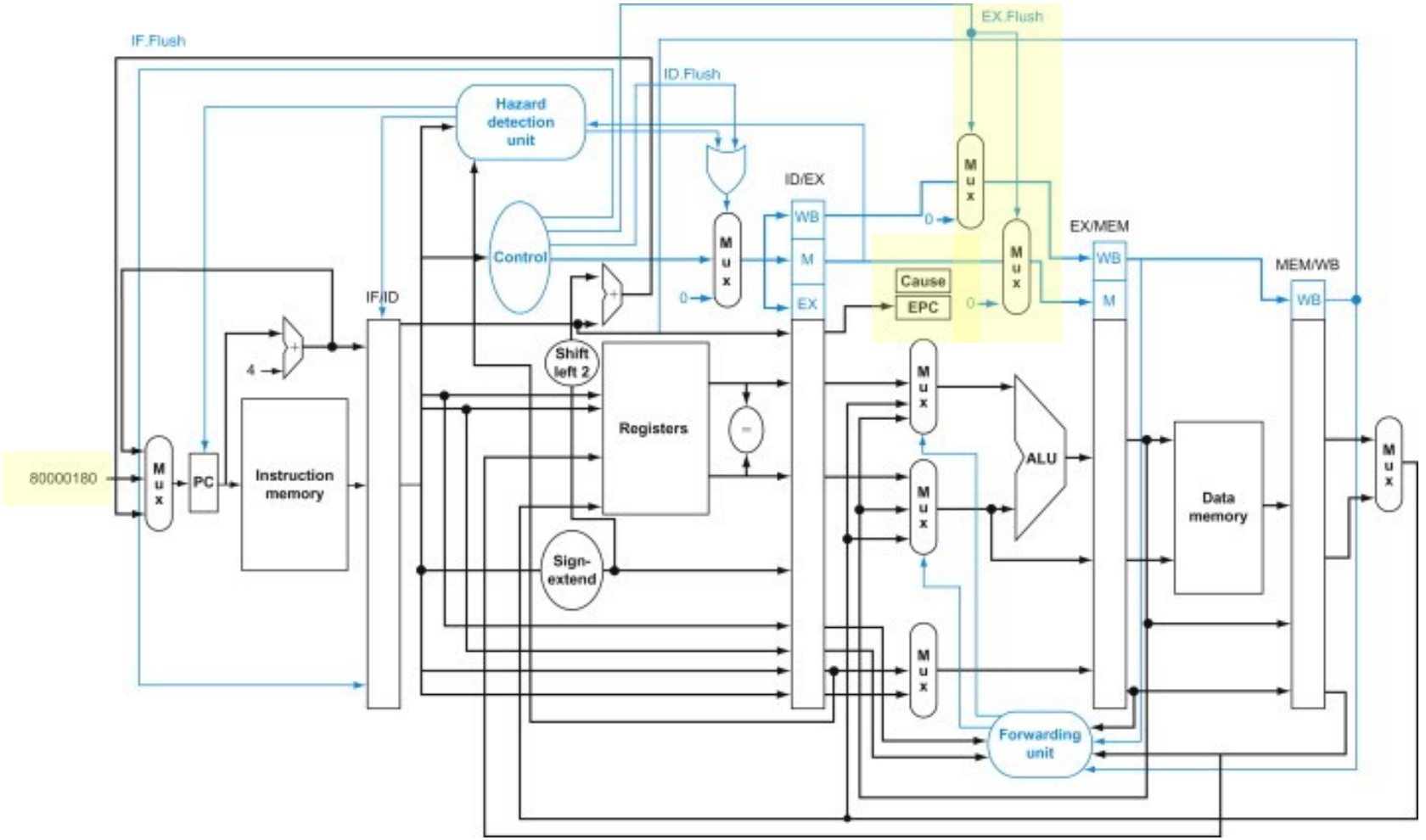
Exceptions: Software, cont'd

- Handle specific exception
- Jump to clean-up to resume user program
- Restore registers
- Reset flag that detects incorrect entry
- Atomically
  - Restore previous mode (user vs. supervisor)
  - Enable interrupts
  - Jump back to program (using EPC)

Implementing Exceptions

- We worry only about hardware, not s/w
- IntCause
  - 0 undefined instruction
  - 1 arithmetic overflow
- Changes to the datapath (e.g., project extra credit)
  - Detect exception
  - Add additional source for next PC (e.g., another 2-1 mux)
  - Storage for exception cause, return address, spare register
- Additional complexity in control logic

Pipeline With Exceptions (Fig 4.66)



Summary and Review

- Exceptions
- Handling exceptions in a pipelined design