Precise Interrupts

Precise interrupts preserve the model that instructions execute in program-generated order, one at a time

- If an interrupt occurs, the processor can recover from it

What happens on a precise interrupt:

- ***identify the instruction that caused the interrupt***
- ***let the instructions before faulting instruction finish***
- disable writes for faulting & subsequent instructions
- force trap instruction into pipeline
- trap routine
    - save the state of the executing program
    - correct the cause of the interrupt
    - restore program state
- ***restart faulting & subsequent instructions***

How Pipelines Complicate Interrupt Handling

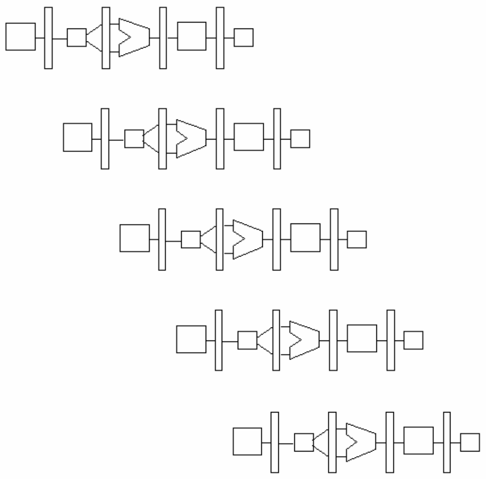
It’s fairly simple to maintain precise interrupts in the R3000 integer pipeline

- one instruction fetched and executed each cycle
- instructions executed in fetch order
- writes done at the end of the pipeline

Still, there are issues….

How Pipelines Complicate Interrupt Handling

How do you determine which instruction caused the exception?



How Pipelines Complicate Interrupt Handling

How do you handle simultaneous interrupts

- 2 stages cause an interrupt at the same time
- a solution: handle them in program order
- still precise

How Pipelines Complicate Interrupt Handling

How are interrupts that occur out of order wrt sequential instruction execution handled?

- an instruction later in the pipeline (younger) causes an interrupt before a previous instruction (older)
- interrupts still must be handled in program order for precise interrupts

A **solution**: interrupt is handled before the write stage

- interrupt recorded in a per-instruction bit vector which flows with it down the pipeline
- interrupts for instruction in write stage are handled before it changes any state
- restart all instructions in the pipeline