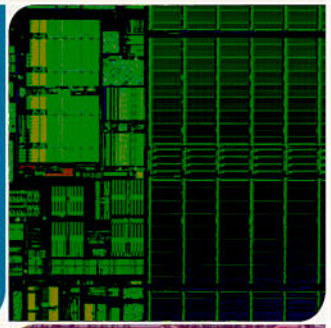
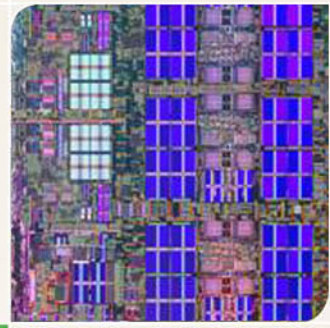
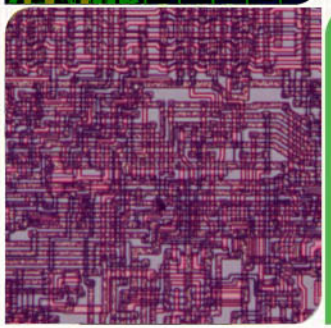
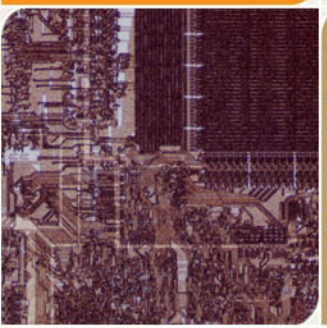




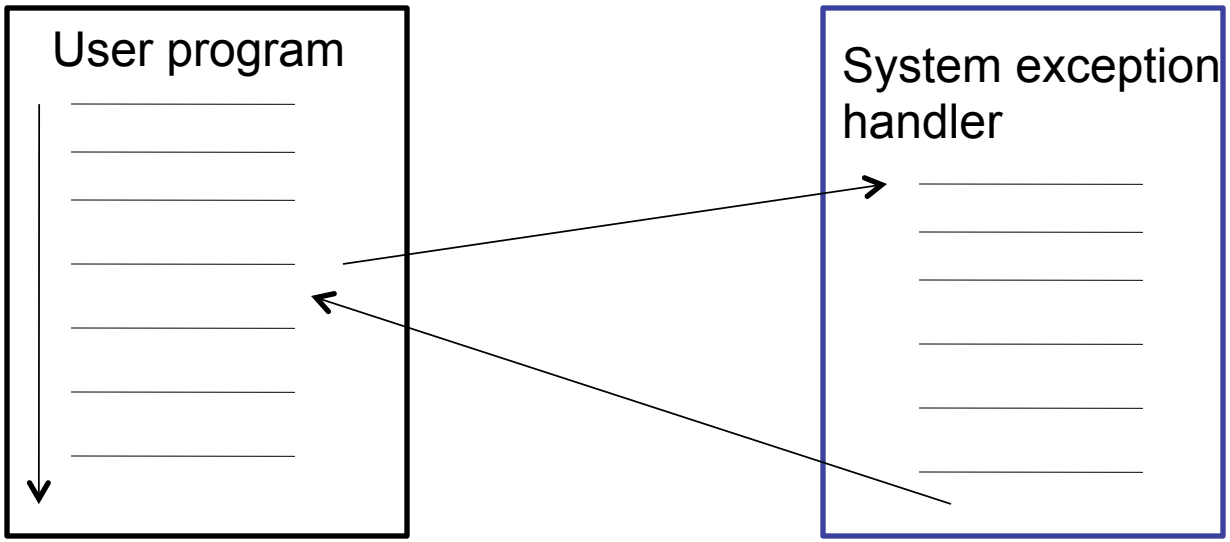
CS4290/CS6290

Fall 2011

Prof. Hyesoon Kim

Georgia Tech College of Computing

Exceptions - Basic



- Exception = unprogrammed control flow
  - System takes action to handle the exception



Interrupts vs. Exceptions (traps)

- Interrupts
  - Cased by external events
    - (network, keyboard, internal timer..)
  - Asynchronous to program execution
  - May be handled between instructions
  - Simply suspend and resume user program
- Exceptions (Traps)
  - Caused by internal events
    - Exceptional conditions (overflow), Errors (parity), Page faults
  - Synchronous to program execution
  - Condition must be remedied by the handler
  - Instruction may be retired and program continued or program may be aborted.



Which Stages can Generate Exceptions ?

- Fetch:
  - Page fault on instruction fetch, misaligned memory access, memory-protection violation
- Decode:
  - Undefined or illegal opcode
- Execution stage:
  - Arithmetic exception
- Memory stage:
  - Page fault of data fetch, misaligned memory access, memory protection violation, memory error



Q

- Which instructions can generate exceptions?

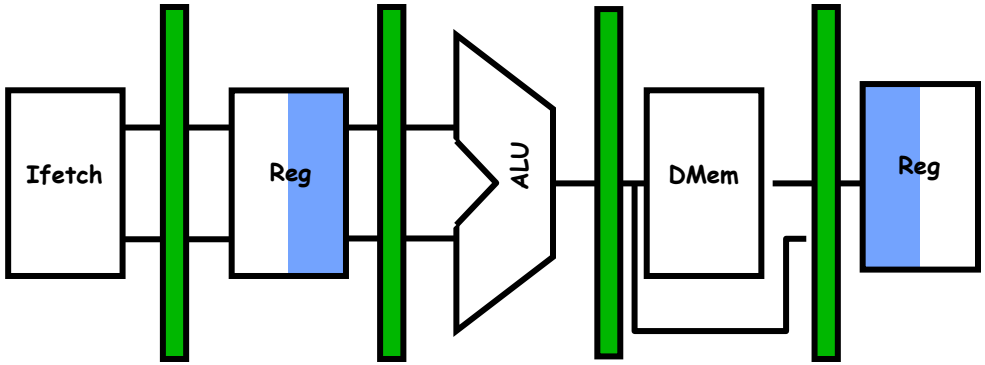
- A: all
  - (instruction fetch)



Precise Exception

- Exceptions should be precise.
- Precise→ state of the machine is preserved as if program executed up to the exception cause instruction.
  - All previous instructions
  - All following instructions the process state
  - The exception cause instruction been executed, depending on the definition of the architecture and the cause of the exception.

Exceptions in the Pipeline



Load

Div Load

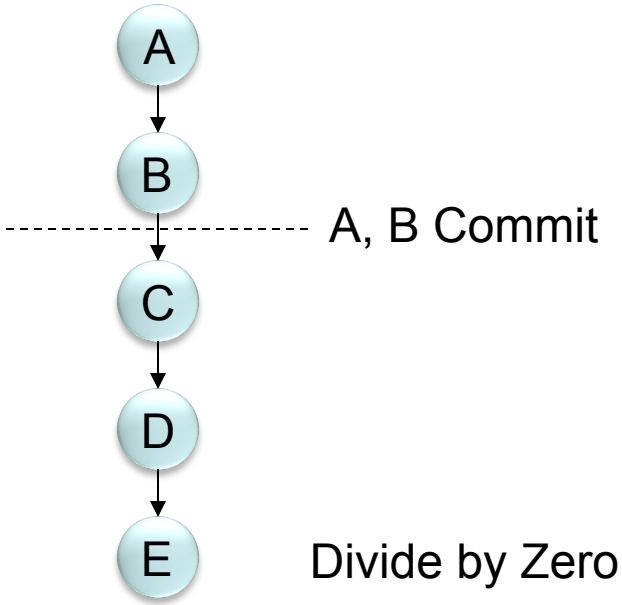
Div Load

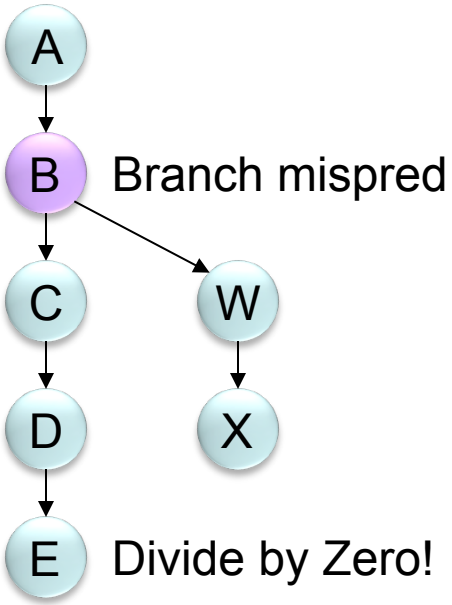
Exception

Load

Commit, Exceptions

- What happens if a speculatively executed instruction faults?





Outside world sees: A, B, fault!

Should have been: A, B, C, D, fault!

Fault should never be seen!



Treat Fault Like a Result

- Regular register results written to ROB until
  - Instruction is oldest for in-order state update
  - Instruction is known to be non-speculative
- Do the same with faults!
  - At exec, make note of any faults, but don't expose to outside world
  - If the instruction gets to commit, then expose the fault

Example

A LOAD R1 = 0[R2]

B ADD R3 = R1 + R4

C SUB R1 = R5 – R6

D DIV R4 = R7 / R1

E LOAD R6 = 0[R7]