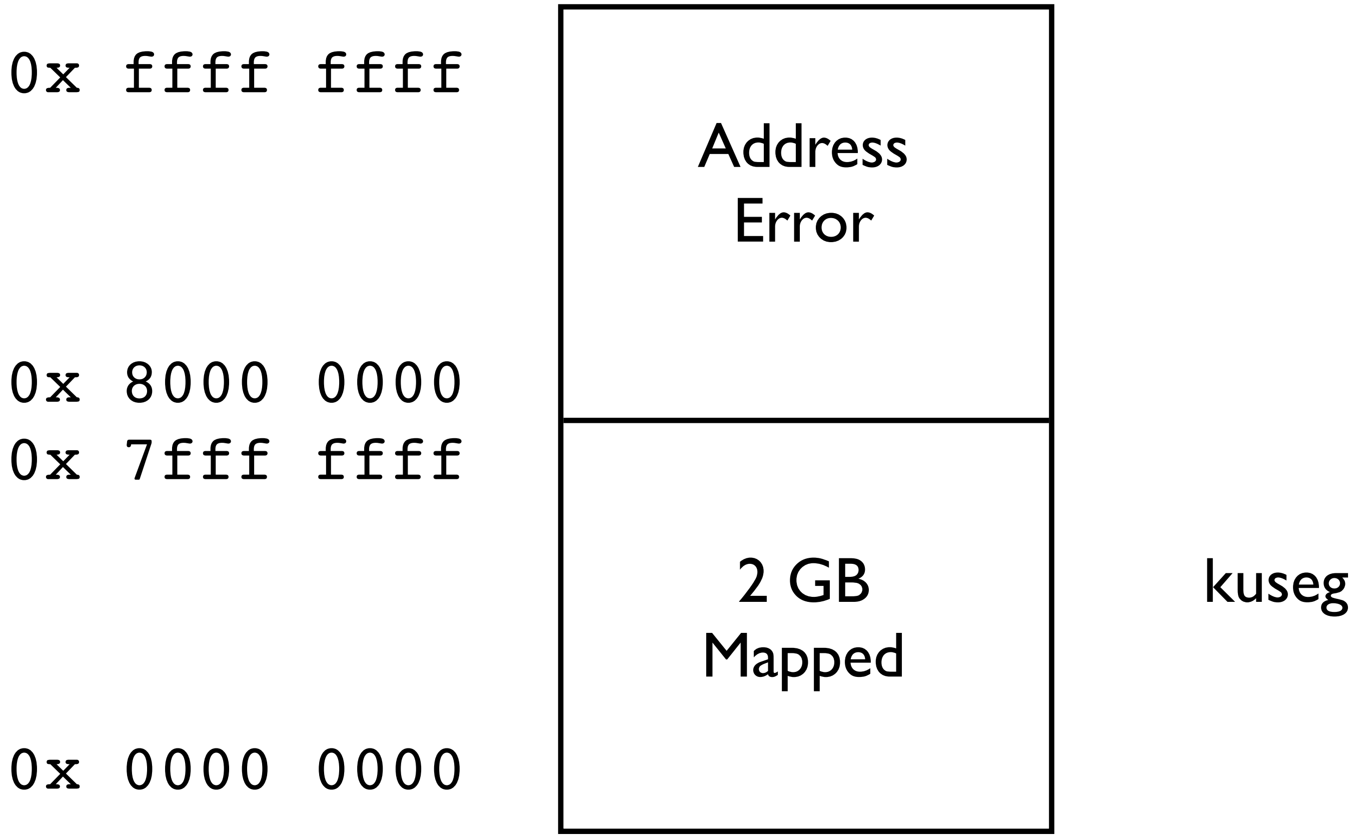
CP0: Exceptions and MMU

"Coprocessor 0"

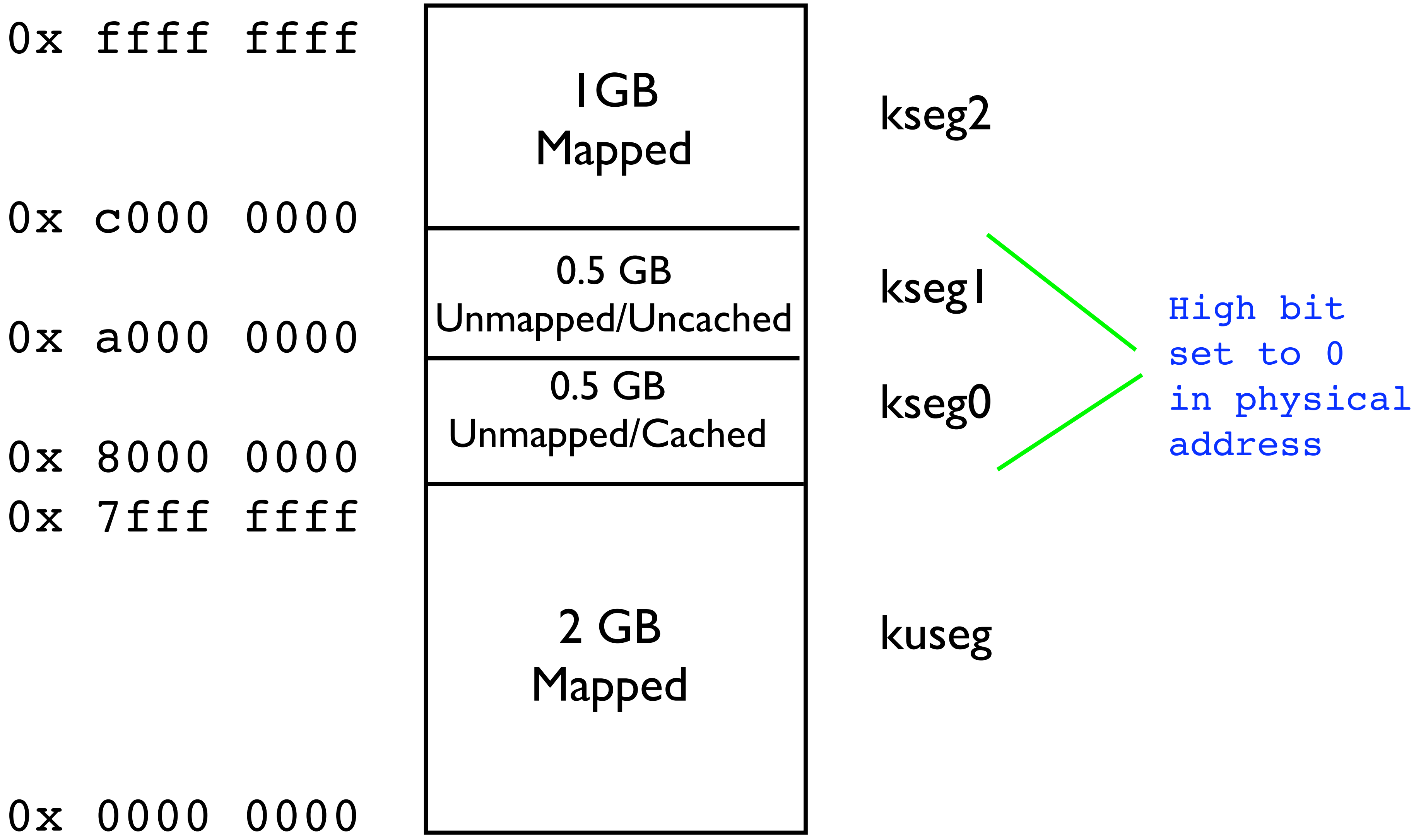
Contains Kernel/User Mode bit
Contains the TLB
Contains special registers for exception handling
- SR - Status Register
- EPC - Exception PC
- CR - Cause Register
- BadVAddr - Bad Virtual Address
- ... (many) others

Special instructions to read/write CP0 registers
CP0 usually set accessible only in Kernel mode

User Mode Address Map



Kernel Mode Address Map



TLB Entries

VPN	ASID	0
20	6	6

PFN	N	D	V	G	0
20	1	1	1	1	8

CP0 contains a 64-entry fully associative TLB

- Effective ASID in field of EntryHi/EntryLo regs in CP0
- G bit disables ASID match
- N bit disables caching

CP0 usually set accessible only in Kernel mode
so address map can be managed only by O/S

MIPS R3000 Caches

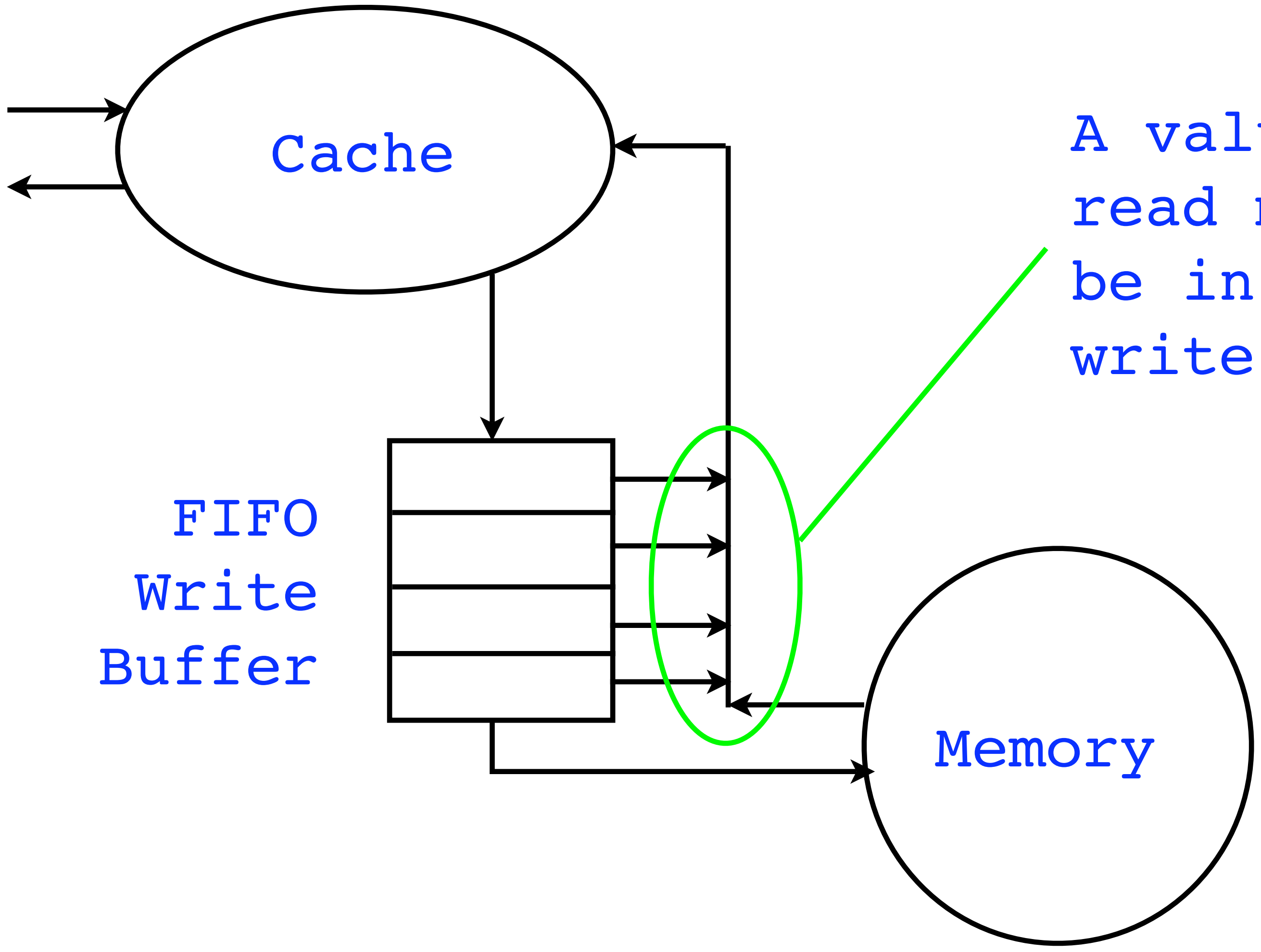
Separate I and D caches

4K (min) - 256K (max) bytes

Write-Through
 (FIFO Store Buffer)

Physical Addresses

MIPS R3000 Caches



A value being read may still be in the write buffer!

Buffer => average write rate can reach memory write bandwidth

Precise Exceptions

- Interrupt execution
  - Next instruction fetched from interrupt handler
  - Instruction error condition (overflow) or external event (I/O interrupt)
- Exception *victim* instruction
  - Every instruction logically before the victim completes
  - The victim *and every instruction after it* does not complete (has *no unrecoverable side effects*)
- Return mechanism
  - Restore state, restart victim

Subtleties ...

Exceptions should appear in instruction sequence ...

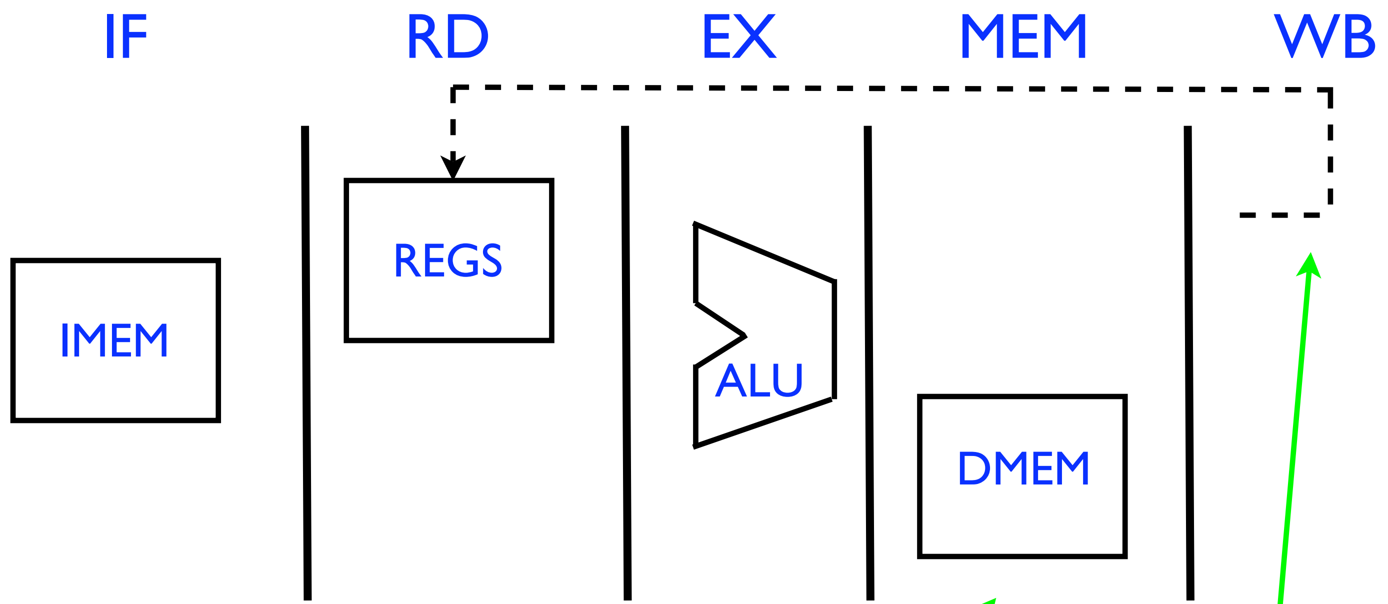
```
lw   $1,17($2)  # bad address
xx   .....      # invalid opcode
```

Instructions in the pipe after the victim must be canceled without side-effects

Exception handler software must be able to locate and restart the victim ...

What if the victim is in a branch delay slot?

Where Do Instructions Commit?

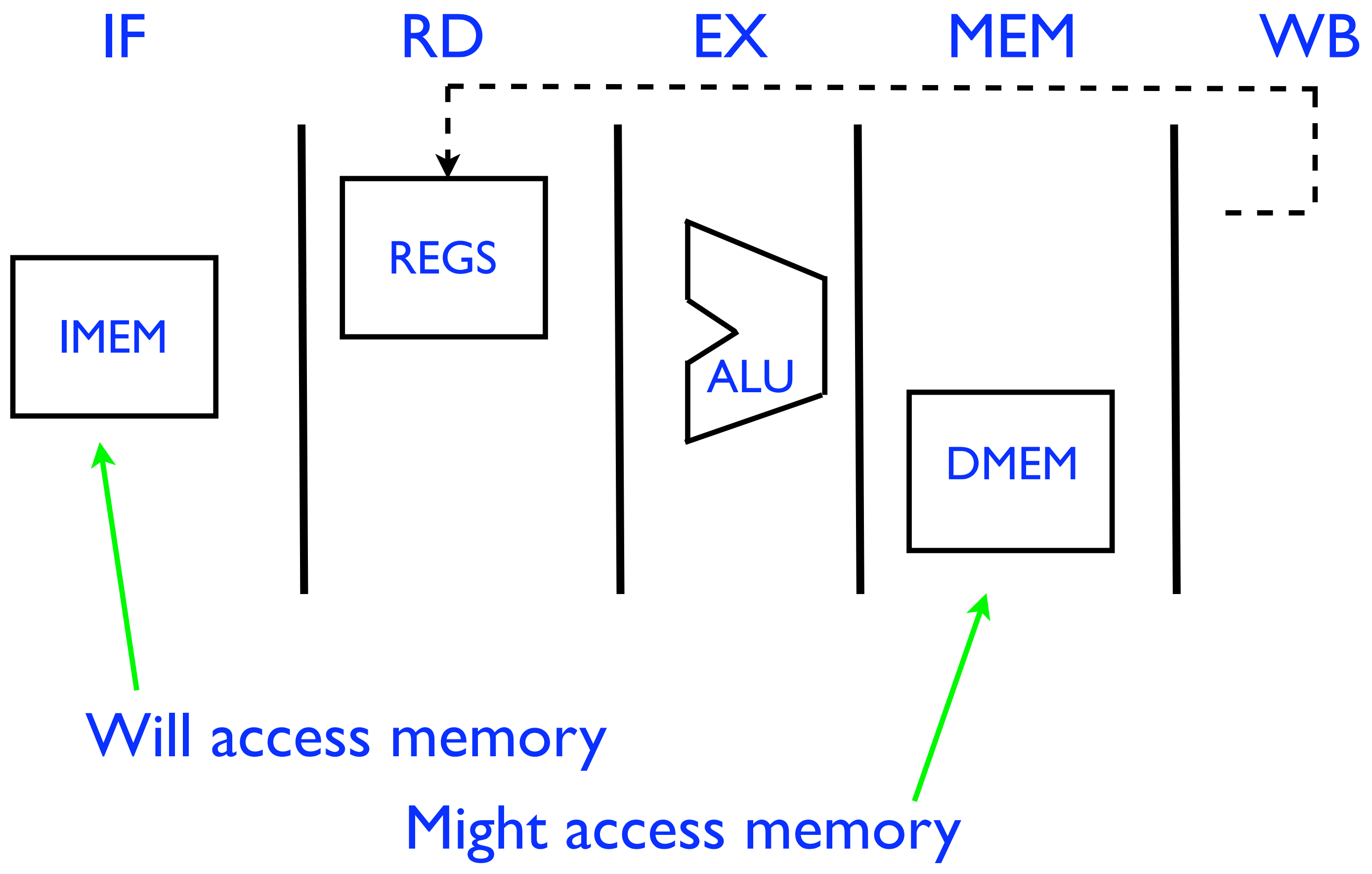


Might store into memory

Might write back into register file

An instruction may be canceled prior to this!
Creates a *bubble* in the pipeline

Where User/Kernel Mode Matters?



K/U mode determines virtual-to-physical address mapping!

External (I/O) Interrupt

(SR interrupt mask bits allow the interrupt)

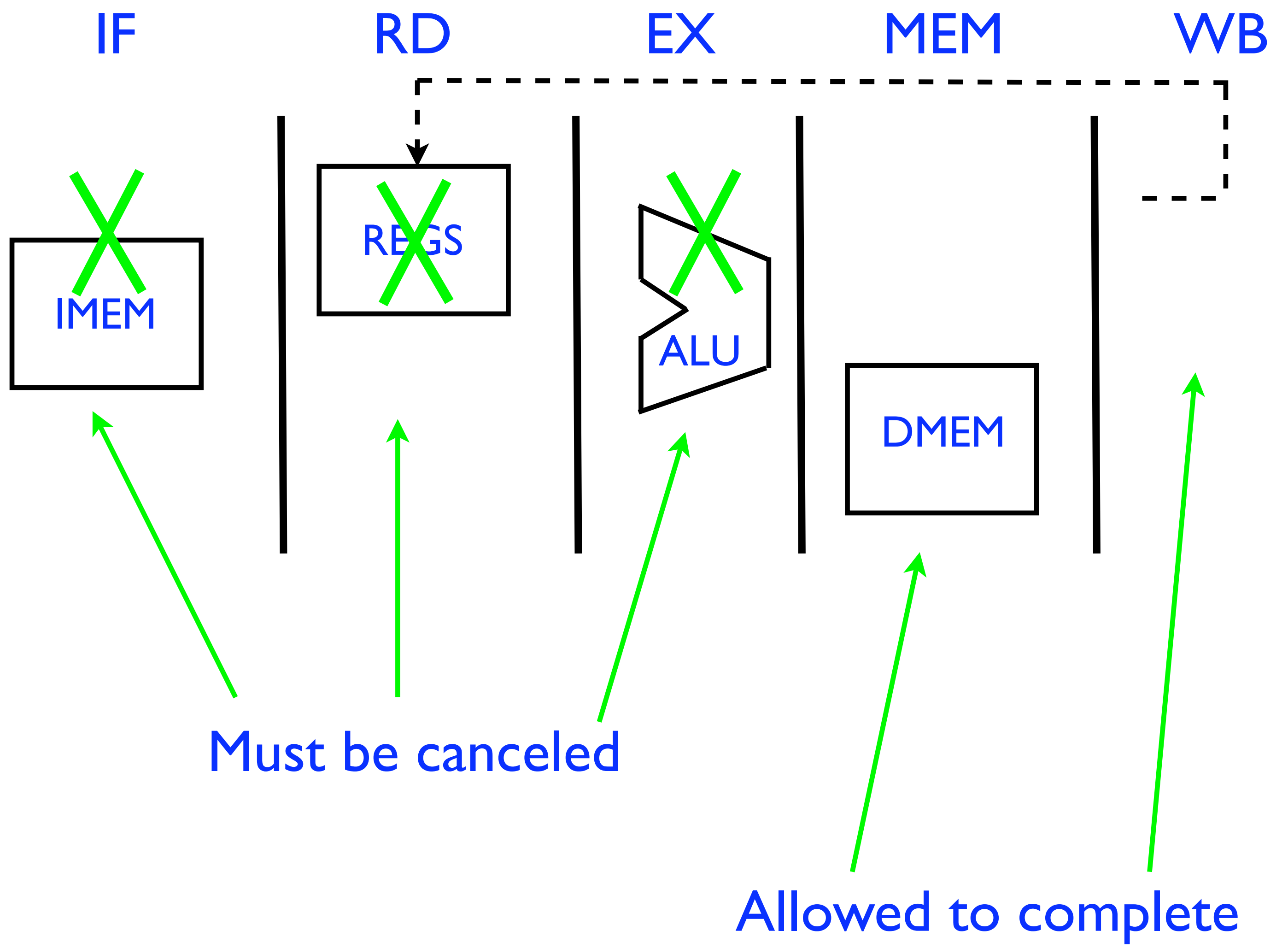
- Device sets interrupt request bit in CR

- Save victim address
  Set EPC = victim address if not in delay slot
  Set EPC = prior branch, CR(BD) = 1
    if in delay slot

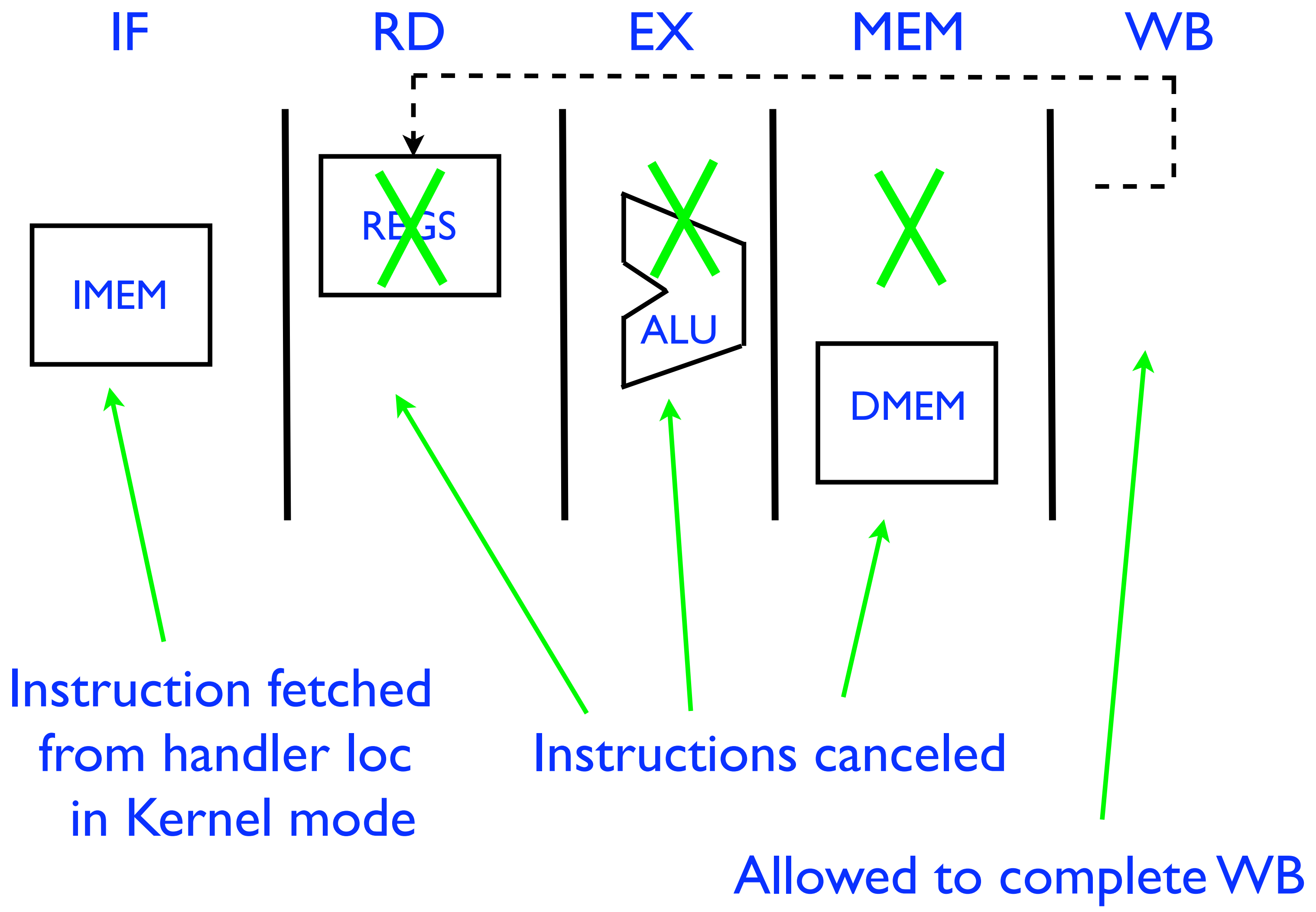
- Save current IE/K bits in SR,
  Set IE=0, K=1 in SR

- Fetch next instruction from handler
  (in Kernel mode)

I/O Interrupt in Pipeline ...



I/O Interrupt in Pipeline II ...



Returning From Interrupt

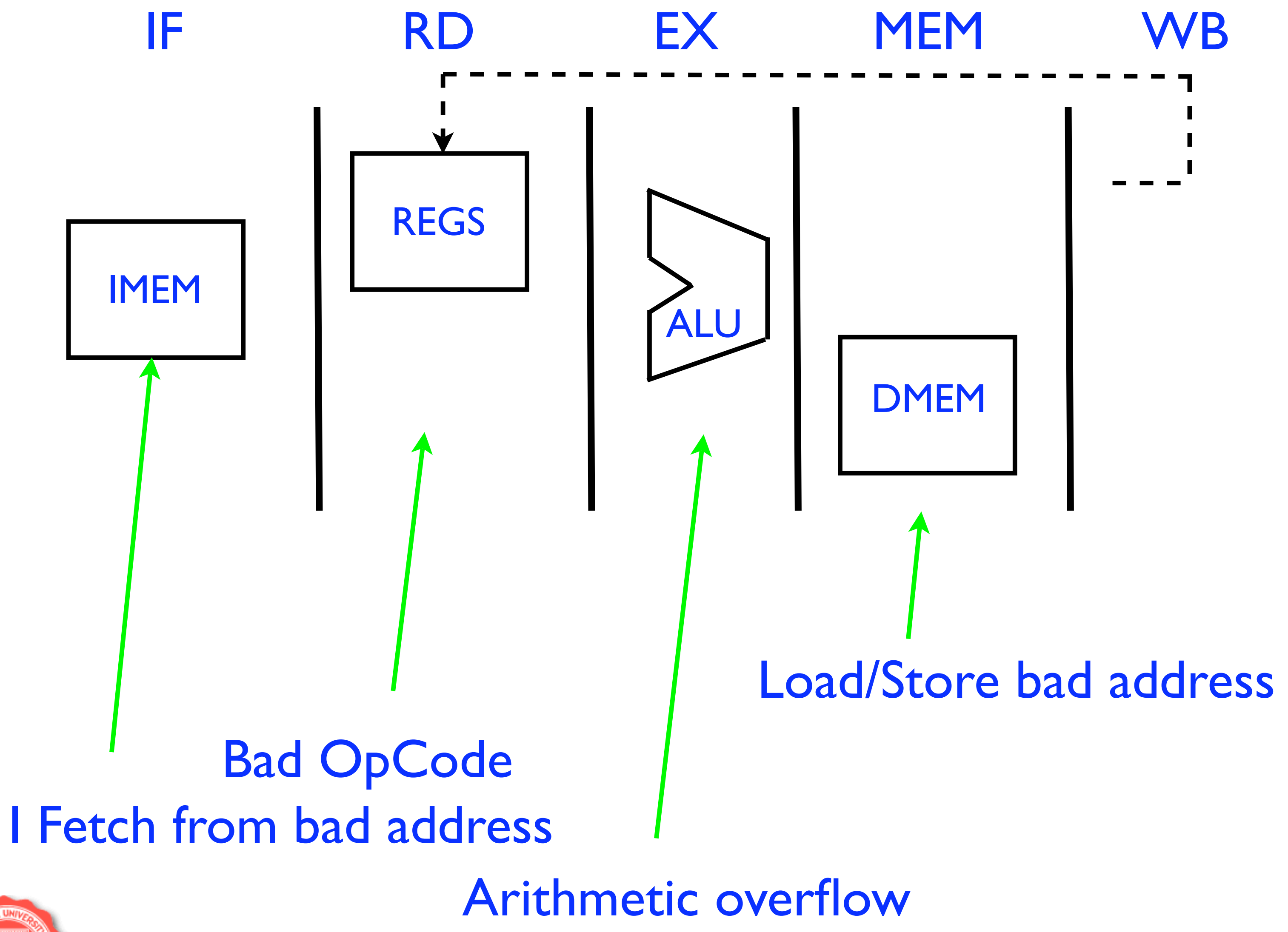
Must do two things:
- Return to EPC
- Restore K/U mode and IEnable state

Must do this atomically!

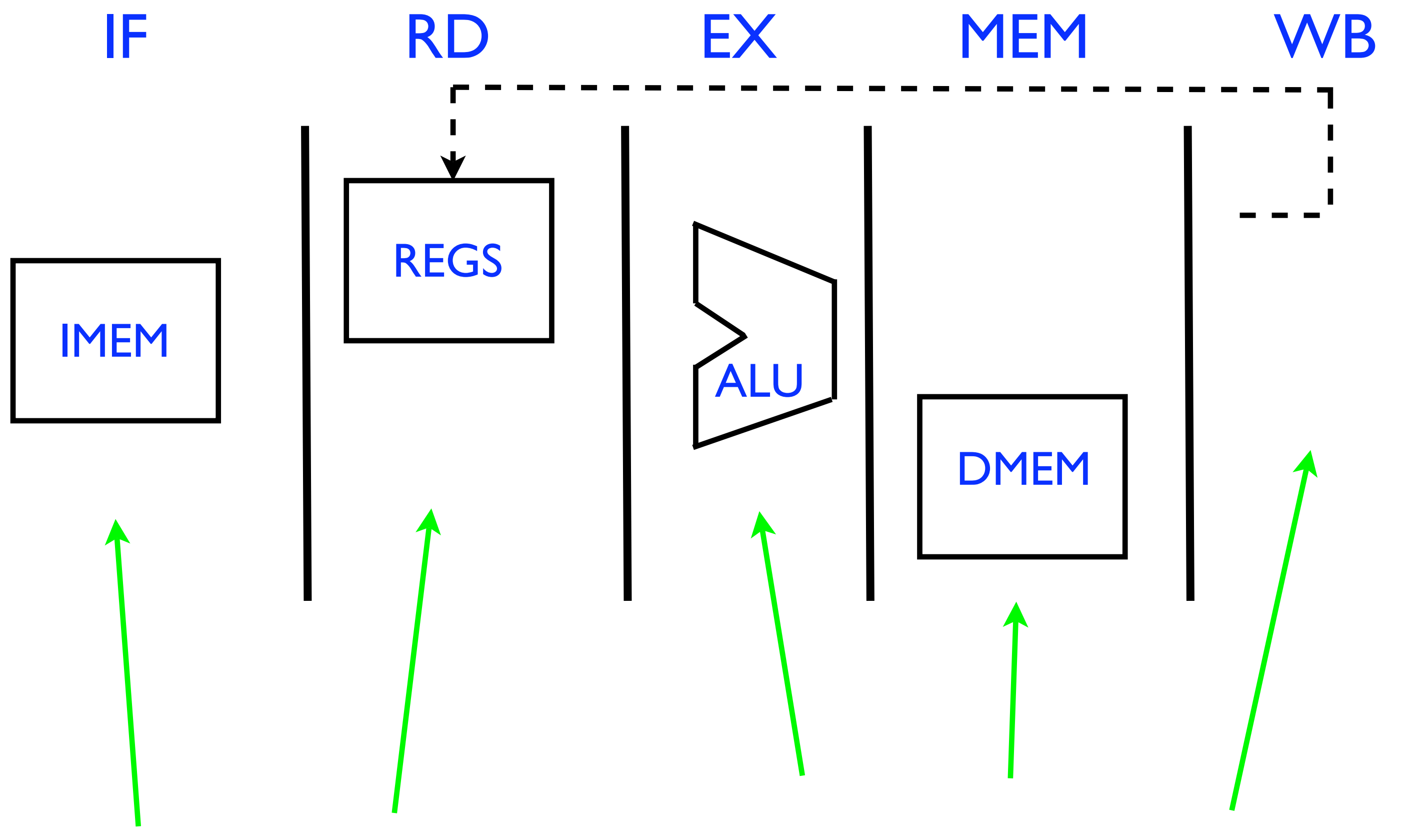
On MIPS 3000 this is done by exploiting branch delay slot:

```
nop
...
nop
jr    $k0    # contains EPC from CP0
rfe          # copy prev K/IE bits into
             # current K/IE bits in SR
```

Where is Exception Detected?



This is Costly ...



Can't raise exception from here

Until these instructions are certain to succeed!

Can't raise immediately -- must FLOP exception request thru pipeline stages!