CPO: 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

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x ffff ffff</td>
<td>Address Error</td>
</tr>
<tr>
<td>0x 8000 0000</td>
<td>2 GB Mapped</td>
</tr>
<tr>
<td>0x 7fff ffff</td>
<td></td>
</tr>
<tr>
<td>0x 0000 0000</td>
<td></td>
</tr>
<tr>
<td>kuseg</td>
<td></td>
</tr>
</tbody>
</table>
Kernel Mode Address Map

- **0x ffff ffff** 1GB Mapped
- **0x c000 0000** 0.5 GB Unmapped/Uncached
- **0x a000 0000** 0.5 GB Unmapped/Cached
- **0x 8000 0000** 2 GB Mapped
- **0x 7fff ffff**
- **0x 0000 0000**

- kseg2
- kseg1
- kseg0
- kuseg

High bit set to 0 in physical address
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

Cache

FIFO Write Buffer

Memory

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 ...

\[
\text{lw} \quad \$1,17($2) \quad \# \text{bad address} \\
\text{xx} \quad \ldots \quad \# \text{invalid opcode}
\]

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?

- IF
- RD
- EX
- MEM
- WB

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...

IF → RD → EX → MEM → WB

- IMEM: Must be canceled
- REGS: X
- ALU: X
- DMEM: Allowed to complete
I/O Interrupt in Pipeline II...

Instruction fetched from handler loc in Kernel mode

Instructions canceled

Allowed to complete WB
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:

```assembly
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?

IF  |  RD  |  EX  |  MEM  |  WB

IMEM  |  REGS  |  ALU  |  DMEM  |  

Bad OpCode
I Fetch from bad address

Load/Store bad address

Arithmetic overflow
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!