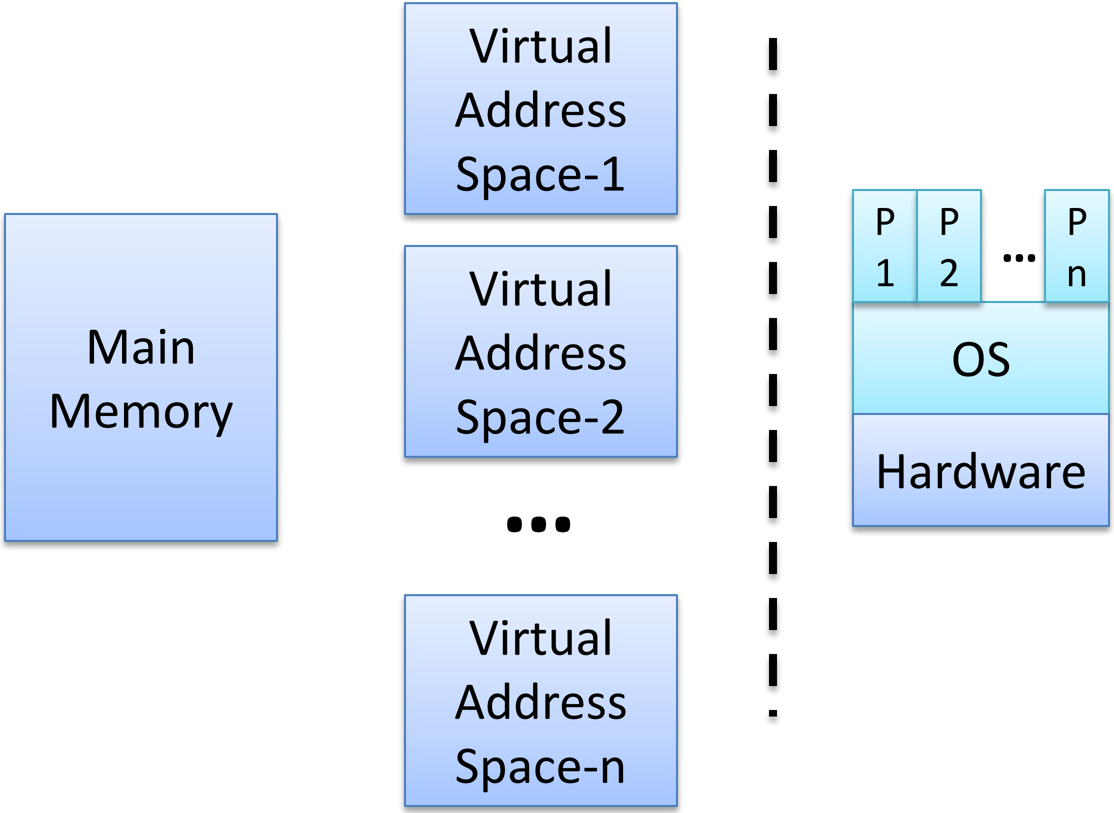
CS305
Computer Architecture

Hardware and OS Interaction for Virtual Memory

Bhaskaran Raman
Room 406, KR Building
Department of CSE, IIT Bombay

http://www.cse.iitb.ac.in/~br

Necessity of Hardware-OS Interaction



Main Memory

Virtual Address Space-1

Virtual Address Space-2

…

Virtual Address Space-n

P 1

P 2

…

P n

OS

Hardware

- OS manages processes
- Manages their VA spaces
- Hardware-OS interaction crucial for VM
- This is unlike the case of caches (which are mostly transparent to all software)

TLB, PT Management

- PT miss (page fault) handled in software: exception handler
- TLB miss can be handled in hardware or software
  - MIPS handles TLB miss in software: exception handler
  - Need special instructions for TLB access
- What if regular programs write TLB or PT?
  - Need (at least) two processor modes: kernel or supervisor mode, regular user mode
  - TLB, PT writing allowed only in kernel mode

Switching Processor Modes

- Switching modes needs to be controlled
- User-to-kernel:
  - On exception, enter kernel mode automatically
  - syscall or trap instructions: also called software exceptions
- Kernel-to-user:
  - eret (exception return)
- While triggering exception, the hardware:
  - Switches to kernel mode
  - Disables further exceptions (will be enabled at a safe stage)

The MIPS TLB Miss Handler

```
TLB miss exception handler at 0x80000000
mfc0 $k1, Context # spl reg with addr of relevant PT entry
lw   $k1, 0($k1)  # load PT entry (1 word) into reg
mtc0 $k1, EntryLo # prepare to load TLB
tlbwr             # EntryLo --> random locn in TLB
eret              # done handling TLB miss
```

- Invalid PT entries may be loaded onto TLB too!
- TLB miss considered more common than page fault
- Common case is made fast (~ a dozen cycles)

TLB and Multiple Processes

- VA space is per process ➔
- VA-to-PA mapping is per process ➔
- TLB entries (cache of this mapping) is per process
- What to do on context switch?
  - Option-1: flush TLB
  - Option-2: have a PID (process ID) tag field in TLB, and process has a PID register (filled by OS)

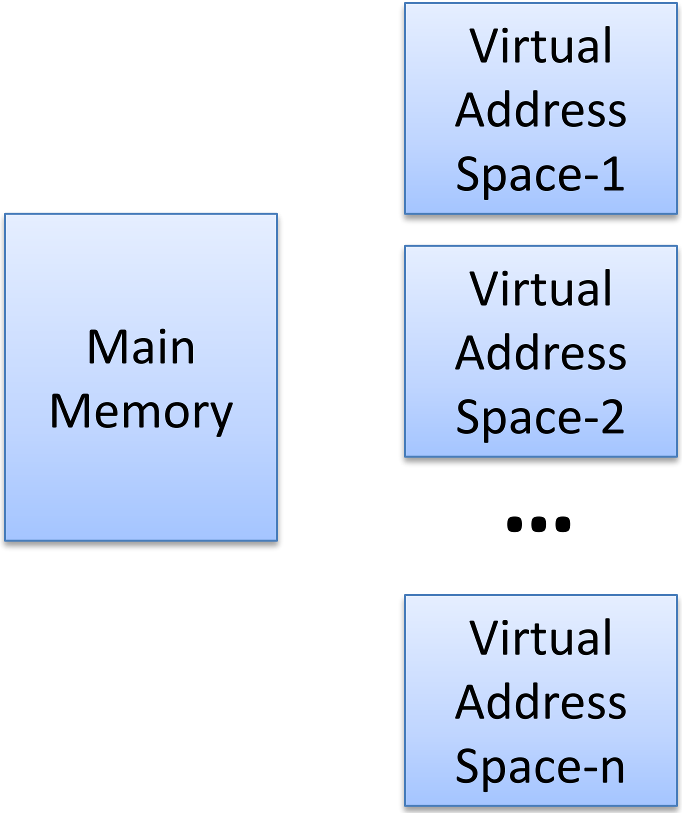
Page Fault Handler

- At 0x8000 0180, separate from TLB miss handler
  - Optimize common case of TLB miss
- Page fault handler has to:
  - Save process state: all GPRs, Hi, Lo onto exception stack
  - Re-enable exceptions
  - Read PT from HD (may need to write dirty page first)
  - Typically, switch to another process which is ready to run

Some Remarks

- Restarting exceptions is NOT easy (e.g. string copy instruction)
- Exception handling itself is not virtual
  - Unmapped memory
  - In MIPS: 0x8000 0000 to 0x8000 FFFF mapped statically to lower portion of physical memory

Thrashing, Working Set



Main Memory

Virtual Address Space-1

Virtual Address Space-2

…

Virtual Address Space-n

- Working set: the set of pages a program or set of "active" programs need in main memory
- Thrashing: when working set size exceeds the main memory size
- High page fault rate
- Processor stalls, waiting for disk: terrible performance

Summary

- VM: hardware and OS need to work together
  - Special instructions for TLB access
  - Processor modes
  - Special instructions for switching modes
- Next: Input/Output systems