CS 350 Notes

Matthew Visser

Oct 13, 2011

1 Memory Management

1.1 TLB

The TLB works because we have

- Large pages
- Locality
  - spacial — memory is in regions. If you're accessing one region, you're probably going to be accessing regions next to it.
  - temporal

We will formalize the notion of locality later.

What does an entry in the MIPS TLB look like?

The MIPS page size is 4kB = 2^{12} bytes $\implies$ offset is 12 bits.

Have

- first word
  - 20 bits Virtual Page Number (VPN)
  - 6 bits ASID — address space ID
  - 6 bits empty
- Second Word
  - 20 bits PFN
  - 1 bit N — not cacheable
  - 1 bit D — dirty

    * changes to 1 when written to.
    * if 0, can be trusted
  - 1 bit V — Valid bit
  - 1 bit G — Global bit. Tells the MMU to ignore the ASID when 1.
  - 8 bits null.

We can modify the TLB with the functions

- `TLB_Write`
- `TLB_Random`
- `TLB_Read`
- `TLB_Probe`

1.2 Segmentation

Different parts of the program are segments. The different segments are loaded into different parts of physical memory. The segments are described in the *segment table*. This has information on the physical address of the segment. See slide 18 of `vm.pdf`.

The segment table points to the page table when we want to combine them. See slide 20 for a diagram.

The segmentation addresses the problem of sparse addresses.

We don't, however, have segmentation in OS/161.

1.3 Address Translation

- The MMU translates addresses
- If there is an error, we have a TLB fault (called an *address exception*).
- The `vm_fault` function handles this exception for the kernel. It uses information fromt he current process' `addrspace` to construct and load a TLB entry for the page.
- On return from fault handling, the processor re-tries to do the previous action.

Virtual addresses must start at an address that is divisible by the page size. In our representation, last three digits of hex must be 0.