Memory Addressing

Pradipta De
pradipta.de@sunykorea.ac.kr
Today’s Topic

• Allow processes (and kernel) to reliably access physical memory
  – How to create illusions of large RAM

• Ensure memory protection
  – Do not let other processes corrupt memory
Outline

- Processor Modes
- Segmentation
- Paging
- Advanced Features
  - Extended Paging
  - PAE (Physical Address Extension)
  - Windows support
Addresses ...

- Logical Address
  - specified as a <segment:offset> tuple
- Linear or virtual address
  - A (20-bit) or 32 bit integer to address (1 MB) or upto 4 GB of RAM
  - What if RAM > 4 GB ?
- Physical address
  - Address memory cells in memory chip
- Some useful facts:
  - 32 bit (or 36-bit) address for 32 bit machines
  - 64 bit address for 64 bit machines
    - What is addressable memory ? Theoretically 16 ExaByte
    - Current: 48 bits i.e. 256TB (AMD64) and 42 bits i.e. 4 TB (Intel)
    - [http://en.wikipedia.org/wiki/X86_64#Virtual_address_space_details](http://en.wikipedia.org/wiki/X86_64#Virtual_address_space_details)
Different Modes

• **Real Mode:** uses real mode addressing
  – Used at boot time …
  – $CS << 4 + IP = Linear\ Address\ Of\ Instruction$
  – Segmentation (no paging)

• **Protected Mode:** hardware supported
  – Switches by setting PE bit in CR0 register
  – Paging available (80386 onwards)
  – Privilege levels
Different Modes (contd)

• Long Mode: addressing in 64-bit mode
  – not much different from 32 bit mode
  – Linear addresses are now 64 bit
  – 64-bit registers are available
Segmentation
Segmentation Hardware

- Logical address → linear address
- Memory viewed as collection of segments
- **Segment selector**: chooses one of 8K segment descriptors
- **Segment descriptor**: provides segment base address
Segment Selection

- **Segment Selector**
  - 13 bits: pointer to segment descriptor
  - 1 bit: LDT/GDT
  - 2 bits: privilege level

- **Segment Descriptor**
  - Segment Base address
  - Other control fields
    - Limit: to check if the address exceeds the segment boundary → used in memory protection
  - One GDT, multiple LDTs (one for each process)
Segment Descriptor

Data Segment Descriptor

Code Segment Descriptor

System Segment Descriptor
Linux does not really use segmentation!!

4 segments – user code, user data, kernel code and kernel data

<table>
<thead>
<tr>
<th>Segment</th>
<th>Base</th>
<th>G</th>
<th>Limit</th>
<th>S</th>
<th>Type</th>
<th>DPL</th>
<th>D/B</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>user code</td>
<td>0x00000000</td>
<td>1</td>
<td>0xfffff</td>
<td>1</td>
<td>10</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>user data</td>
<td>0x00000000</td>
<td>1</td>
<td>0xfffff</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>kernel code</td>
<td>0x00000000</td>
<td>1</td>
<td>0xfffff</td>
<td>1</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>kernel data</td>
<td>0x00000000</td>
<td>1</td>
<td>0xfffff</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Paging

Convert linear address to physical address
Regular Paging

- 10 bits: directory
- 10 bits: page table
  - 1024 entries
- 12 bits: offset $\rightarrow$ 4KB page

- Why 3 levels only?
  - Saves memory for housekeeping
Translation Lookaside Buffers

• Cache: linear (virtual) address ➔ physical address

• Why cache?
  – MMU takes time to translate

• Multiprocessor systems
  – Each CPU has its own TLB

• When do we flush TLB?
  – On context switch … when cr3 register, that’s points to a page directory, is repopulated
Additional tricks (or hacks !!)

• Extended Paging
  – Page frames > 4 KB … it can be 4 MB
  – Do not use a page table … only page directory
    ➔ 1 level translation
  – Saves the space for page table
  – TLB entries are more long-lived
Additional tricks (or hacks !!)

- Physical address extension (PAE)
  - Address RAM > 4 GB
  - Soln: 4 extra pins for addressing, 36-bit address (64GB)
  - Convert: 32 bit linear address to 36 bit physical address
  - Page table entry:
    - 24 bit (to address $2^{24}$ page frames) + 12 bit (offset)
    - Each PT entry is 64 bits … total 512 PT entries
Advanced Features

Moving on to 64 bit arch
64-bit addressing in Linux

• Lets review 3 level paging …
  – 4KB page $\Rightarrow$ 12 bit offset
  – 48 bits virtual address in AMD64
  – (48-12)=36bits for PT and PD
  – Using 18 bits, $2^{18}$ entries in PT and PD
    • Imagine the size of the PT and PD (per process !!)
  – More levels are introduced ( 9+9+9+9+12)
Specify number of bits for each level to trigger different levels of paging
Windows: large memory support

• 4 GB Tuning
  – Win 32 bit systems, allow user apps to use 2 GB virtual address space
  – Feature to allow app use 3 GB $\rightarrow$ system mem reduces to 1 GB
  – On 64bit, it is 4 GB for app

• Large page support
  – Allows apps to use large memory regions
  – Specially useful for 64 bit Windows

• Fun: check the memory limits for diff win version
Putting It Together

• Early days
  – Segmentation
• Evolution
  – Paging
• Tricks to accommodate more memory
• Finally 64 bit arrived …

• Linux uses
  – Flat address mode $\rightarrow$ ignores segmentation, uses paging for addressing, and memory protection
  – Single data structure accommodates all hardware architectures