Segmentation and Paging
Segmentation
Segmentation

- Space between stack and heap is not used by process.
- Big address space of another process may not fit into memory.

- Segment is a contiguous block of address space.
  - Code
  - Stack
  - Heap

- Base and bound registers per segment
  - Segment registers
  - Virtual address: 4200
  - Offset within segment: 4200 – 4096 = 104
  - Physical address: 34KB (base) + 104 = 34920

- Segmentation fault/violation
  - Virtual address is beyond the end of a segment.
  - E.g., virtual address: 7KB

<table>
<thead>
<tr>
<th>Segment</th>
<th>Base</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>32KB</td>
<td>2KB</td>
</tr>
<tr>
<td>Heap</td>
<td>34KB</td>
<td>2KB</td>
</tr>
<tr>
<td>Stack</td>
<td>28KB</td>
<td>2KB</td>
</tr>
</tbody>
</table>
Segmentation

• Find out segment and offset in the virtual address
  ▪ Top two bits for finding out segment (01 means heap)
  ▪ Remaining bits determine offset (000001101000 = 104)

• Stack grows negatively.
• Add a bit (0 or 1) to indicate the direction of growth
• Virtual address: 15KB (11 1100 0000 0000)
• Negative offset: 3KB (offset) - 4KB (maximum segment size) = -1KB
• Physical address: 28KB + (-1KB) = 27KB
Question

• Find out address translations for the following virtual addresses (in hex):

<table>
<thead>
<tr>
<th>Segment</th>
<th>Base</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>32KB</td>
<td>2KB</td>
</tr>
<tr>
<td>Heap</td>
<td>34KB</td>
<td>2KB</td>
</tr>
<tr>
<td>Stack</td>
<td>28KB</td>
<td>2KB</td>
</tr>
</tbody>
</table>

0x0240
0x3c65
External Fragmentation

- External fragmentation
  - Little unused spaces between allocated spaces
  - Space may not be enough for allocating to another process segment
- Rearrange segments in the form of a contiguous block of memory
  - Update segment register values
  - Expensive (a lot of copying, processor time)
- Free-list management algorithms track a list of free memory spaces.
  - E.g., find an unused space that best fits a new segment
Paging
Paging

- Variable-sized segments for an address space
  - The address space becomes fragmented.
- Divide address space into fixed-sized units (virtual pages)
- Physical memory consists of page frames.
- OS uses page table to record placing virtual pages to physical frames
  - Virtual Page Number (VPN) → Physical Frame Number (PFN)
    virtual page 0 → physical frame 1, virtual page 1 → physical frame 7
- free list contains a list of free pages
Paging

- Perform address translation using virtual address
- 4 pages in address space

- Example:
  movl 21, %eax
  Virtual page number: 1 (01)
  Offset: 5 (0101)
  Physical page frame: 7 (111)

- Page table size
  32-bit address space
  4KB pages
  12-bit offset: $2^{12}$ bytes per page (4KB)
  20-bit virtual page number
  $2^{20}$ page table entries
  
  - Assume 4 bytes per entry
  - $2^{20} \times 4 = 4$MB
  - 400MB for 100 processes

- Page tables are stored somewhere in physical memory allocated for OS
A page table is an array.
Page Table Entries (PTE) are indexed by virtual page numbers.

PTE other contents
- Valid bit: unused pages are marked invalid. ❇️ Not need to allocate page frames (saves memory space)
- Protection bit: read, write, or execute page contents

Some pages may be stored on disk
- Present bit: whether a page is in memory
- Dirty bit: whether a page is modified
- Reference bit: whether a page is accessed
Page Table

• Page-table base register stores the address of a page table.

• Need to have an extra memory reference for fetching the physical frame number
  – Slows down processes
Question

• Given page sizes, how many bits in virtual address are needed to determine offset in page?

<table>
<thead>
<tr>
<th>Page size</th>
<th>Offset (# bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 bytes</td>
<td></td>
</tr>
<tr>
<td>1 KB</td>
<td></td>
</tr>
<tr>
<td>1 MB</td>
<td></td>
</tr>
</tbody>
</table>
Question

Given the number of bits in virtual address and offset, how many virtual pages can there be in address space?

<table>
<thead>
<tr>
<th>Page size</th>
<th>VA (# bits)</th>
<th>Offset (# bits)</th>
<th>VPN (# bits)</th>
<th># Virtual Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 bytes</td>
<td>10</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 KB</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 MB</td>
<td>32</td>
<td>20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Question

Find the physical address for VA = 0x2010
Assume 14-bit address space

Page table

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>