CS31 Worksheet: Week 11: Virtual Memory

Which memory allocation algorithm would you choose? Why?

A. first-fit
B. worst-fit
C. best-fit

Is leaving small fragments a good thing or a bad thing?

Where would worst-fit place this memory chunk?

A. 5 MB
B. 7 MB
C. 9 MB
Problem Summary: Placement

- What if a process’s memory can’t fit into a contiguous memory slot?

---

Problem Summary: Placement

- General solution: don’t require all of a process’s memory to be in one piece!
- What problems does this generate for a compiler?

OS may choose not to place parts in memory at all.
Hardware for Virtual Addressing

- With help from the hardware, we can translate a process from a virtual address to a physical address by changing the base.
- Are we done?
- Is our model safe?

Example: 32-bit virtual addresses

- Suppose we have 8-KB (8192-byte) pages.
- We need enough bits to individually address each byte in the page.
  - How many bits do we need to address 8192 items?

<table>
<thead>
<tr>
<th>2^10</th>
<th>2^11</th>
<th>2^12</th>
<th>2^13</th>
<th>2^14</th>
<th>2^15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
<td>2048</td>
<td>4096</td>
<td>8192</td>
<td>16384</td>
<td>32768</td>
</tr>
</tbody>
</table>
Address Partitioning

Virtual address:
We’ll call these bits $p$.

OS Page Table
For Process

Where is this page in physical memory?
(In which frame?)

Physical address:
We’ll call these bits $f$.

Once we’ve found the frame, which byte(s) do we want to access?

We’ll (still) call these bits $i$.

The bits $p$ (page) in the virtual address and bits $f$ (frame) in physical address do not have to match.

Address Translation

Logical Address

<table>
<thead>
<tr>
<th>Page $p$</th>
<th>Offset $i$</th>
</tr>
</thead>
</table>

Given the meta data we have:
- table base register, table top register
- valid bit, reference bit, dirty bit
- frame number and, permissions
what checks do we have to do for a page hit?

Physical Address
Sizing the Page Table

Logical Address

Page \( p \) | Offset \( i \)
---|---

Number of bits \( n \) specifies max size of table, where number of entries = \( 2^n \)

Number of bits needed to address physical memory *in units of frames*

Number of bits specifies page/frame size

Example of Sizing the Page Table

Page \( p \): 20 bits | Offset \( i \): 12 bits
---|---

How many rows in the page table?

How big is the frame?

How many bits do we need to store the frame number?

Given: 32 bit virtual addresses, 1 GB physical memory

- Address partition: 20 bit page number, 12 bit offset

<table>
<thead>
<tr>
<th>( 2^{10} )</th>
<th>( 2^{20} )</th>
<th>( 2^{30} )</th>
<th>( 2^{40} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1KB</td>
<td>1MB</td>
<td>1GB</td>
<td>1TB</td>
</tr>
</tbody>
</table>