CIS 429/529
Advanced Memory Operations

Prof. Michel A. Kinsky
Memory Management

• The Fifties:
  ‣ Absolute Addresses
  ‣ Dynamic address translation

• The Sixties:
  ‣ Paged memory systems and TLBs
  ‣ Atlas’ Demand paging

• Modern Virtual Memory Systems
Names for Memory Locations

- Machine language address
  - as specified in machine code

- Virtual address
  - ISA specifies translation of machine code address into virtual address of program variable

- Physical address
  - operating system specifies mapping of virtual address into name for a physical memory location
Absolute Addresses

• EDSAC, early 50’s
  ‣ effective address = physical memory address

• Only one program ran at a time, with unrestricted access to entire machine (RAM + I/O devices)

• Addresses in a program depended upon where the program was to be loaded in memory

• But it was more convenient for programmers to write location-independent subroutines
  ‣ How could location independence be achieved?
Dynamic Address Translation

• Motivation:
  ‣ In the early machines, I/O operations were slow and each word transferred involved the CPU
  ‣ Higher throughput if CPU and I/O of 2 or more programs were overlapped. Why?
    • multiprogramming

• Location independent programs:
  ‣ Programming and storage management ease
  ‣ need for a base register
Dynamic Address Translation

• Protection:
  ‣ Independent programs should not affect each other inadvertently
    • need for a bound register
Simple Base and Bound Translation

Base and bounds registers only visible/accessible when processor running in kernel (a.k.a supervisor) mode
Separate Areas for Program and Data

- What is an advantage of this separation?
  - Used today on Cray vector supercomputers
Memory Fragmentation

• As users come and go, the storage is “fragmented”. Therefore, at some stage programs have to be moved around to compact the storage.
Paged Memory Systems

- Processor generated address can be interpreted as a pair `<page number, offset>`

- A page table contains the physical address of the base of each page

```
<table>
<thead>
<tr>
<th></th>
<th>page number</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Address Space of User-1

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Page Table of User-1
Private Address Space per User

- Each user has a page table
- Page table contains an entry for each user page
Where Should Page Tables Reside?

- Space required by the page tables is proportional to the address space, number of users, ...
  - Space requirement is large too expensive to keep in registers

- Special registers just for the current user:
  - What disadvantages does this have?
    - may not be feasible for large page tables

- Main memory:
  - needs one reference to retrieve the page base address and another to access the data word
    - doubles number of memory references!
Page Tables in Physical Memory

User 1

Page Table, User 1

Page Table, User 2

User 2
Demand Paging in Atlas (1962)

- “A page from secondary storage is brought into the primary storage whenever it is (implicitly) demanded by the processor.” Tom Kilburn

- Primary memory as a cache for secondary memory

---

**Diagram:**

- **Primary Memory:**
  - 32 Pages
  - 512 words/page

- **Secondary (Drum):**
  - 32x6 pages

User sees $32 \times 6 \times 512$ words of storage
Modern Virtual Memory Systems

- **Protection & Privacy**
  - several users, each with their private address space and one or more shared address spaces
    - page table ≡ name space

- **Demand Paging**
  - ability to run a program larger than the primary memory

- **What is another big benefit?**
  - The price is address translation on each memory reference
Address Translation and Protection

- Every instruction and data access needs address translation and protection checks.
- A good VM design needs to be fast (~ one cycle) and space efficient.

![Diagram showing address translation and protection processes]

- Virtual Address
- Kernel/User Mode
- Read/Write
- Exception?
- Protection Check
- Address Translation
- Virtual Page No. (VPN) offset
- Physical Page No. (PPN) offset
- Physical Address