Page Table is Per Process

• Each process has its own virtual address space
  – Full address space for each program
  – Simplifies memory allocation, sharing, linking and loading.

Virtual Address Space for Process 1:

0
VP 1
VP 2
...
N-1

Virtual Address Space for Process 2:

0
VP 1
VP 2
...
N-1

Physical Address Space (DRAM):

0
0
PP 2
PP 7
PP 10
M-1

(e.g., read/only library code)
VM as a Tool for Memory Access Protection

- Extend Page Table Entries (PTEs) with permission bits
- Page fault handler checks these before remapping
  - If violated, generate exception (Access Protection exception)

<table>
<thead>
<tr>
<th></th>
<th>Read?</th>
<th>Write?</th>
<th>Physical Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP 0:</td>
<td>Yes</td>
<td>No</td>
<td>PP 9</td>
</tr>
<tr>
<td>VP 1:</td>
<td>Yes</td>
<td>Yes</td>
<td>PP 4</td>
</tr>
<tr>
<td>VP 2:</td>
<td>No</td>
<td>No</td>
<td>XXXXXXXX</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Read?</th>
<th>Write?</th>
<th>Physical Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP 0:</td>
<td>Yes</td>
<td>Yes</td>
<td>PP 6</td>
</tr>
<tr>
<td>VP 1:</td>
<td>Yes</td>
<td>No</td>
<td>PP 9</td>
</tr>
<tr>
<td>VP 2:</td>
<td>No</td>
<td>No</td>
<td>XXXXXXXX</td>
</tr>
</tbody>
</table>
Superpages

• If a program’s working set size is 16 MB and page size is 8KB, there are 2K frequently accessed pages – a 128-entry TLB will not suffice
• By increasing page size to 128KB, TLB misses will be eliminated – disadvantage: memory waste, increase in page fault penalty
• Can we change page size at run-time?
• Note that a single page has to be contiguous in physical memory
Superpages Implementation

• At run-time, build superpages if you find that contiguous virtual pages are being accessed at the same time
• For example, virtual pages 64-79 may be frequently accessed – coalesce these pages into a single superpage of size 128KB that has a single entry in the TLB
• The physical superpage has to be in contiguous physical memory – the 16 physical pages have to be moved so they are contiguous
Ski Rental Problem

- Promoting a series of contiguous virtual pages into a superpage reduces TLB misses, but has a cost: copying physical memory into contiguous locations.
- Page usage statistics can determine if pages are good candidates for superpage promotion, but if cost of a TLB miss is $x$ and cost of copying pages is $Nx$, when do you decide to form a superpage?
- If ski rentals cost $50 and new skis cost $500, when do I decide to buy new skis?
Main Memory in the System
Some Fundamental Concepts

• **Physical address space**
  – Maximum size of main memory: total number of uniquely identifiable locations

• **Physical addressability**
  – Minimum size of data in memory can be addressed
  – Byte-addressable, word-addressable, 64-bit-addressable
  – Microarchitectural addressability depends on the abstraction level of the implementation
The DRAM Subsystem
DRAM Subsystem Organization

- Channel
- DIMM
  - Due In-line Memory Module
- Rank
- Chip
- Bank
- Row/Column
Memory Bank Organization

- Read access sequence:
  1. Decode row address & drive word-lines
  2. Selected bits drive bit-lines
     • Entire row read
  3. Amplify row data
  4. Decode column address & select subset of row
     • Send to output
  5. Precharge bit-lines
     • For next access
Interleaving

- **Interleaving (banking)**
  - **Problem**: A single monolithic memory array takes long to access and does not enable multiple accesses in parallel.
  - **Goal**: Reduce the latency of memory array access and enable multiple accesses in parallel.
  - **Idea**: Divide the array into multiple banks that can be accessed independently (in the same cycle or in consecutive cycles).
    - Each bank is smaller than the entire memory storage.
    - Accesses to different banks can be overlapped.
  - **A Key Issue**: How do you map data to different banks? (i.e., how do you interleave data across banks?)
Some Questions/Concepts

• Can banks be operated fully in parallel?
  – Multiple accesses started per cycle?

• What is the cost of this?
  – Crossbar

• Modern superscalar processors have L1 data caches with multiple, fully-independent banks; DRAM banks share buses
Page Mode DRAM

• A DRAM bank is a 2D array of cells: rows x columns
• A “DRAM row” is also called a “DRAM page”
• The activated row is put in a “row buffer”

• Each address is a <row, column> pair
• Access to a “closed row”
  – Activate command opens row (placed into row buffer)
  – Read/write command reads/writes column in the row buffer
  – Precharge command closes the row and prepares the bank for next access
• Access to an “open row”
  – No need for activate command
DRAM Bank Operation

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)

Row address 0

Row decoder

Columns

Rows

Row Buffer

CONFLICT!

Column address 85

Column mux

Data