COMP 3221

Microprocessors and Embedded Systems

Lectures 36: Virtual Memory - I

http://www.cse.unsw.edu.au/~cs3221

October, 2003

Saeid Nooshabadi

saeid@unsw.edu.au

Some of the slides are adopted from David Patterson (UCB)

Overview

- Virtual Memory
- Page Table

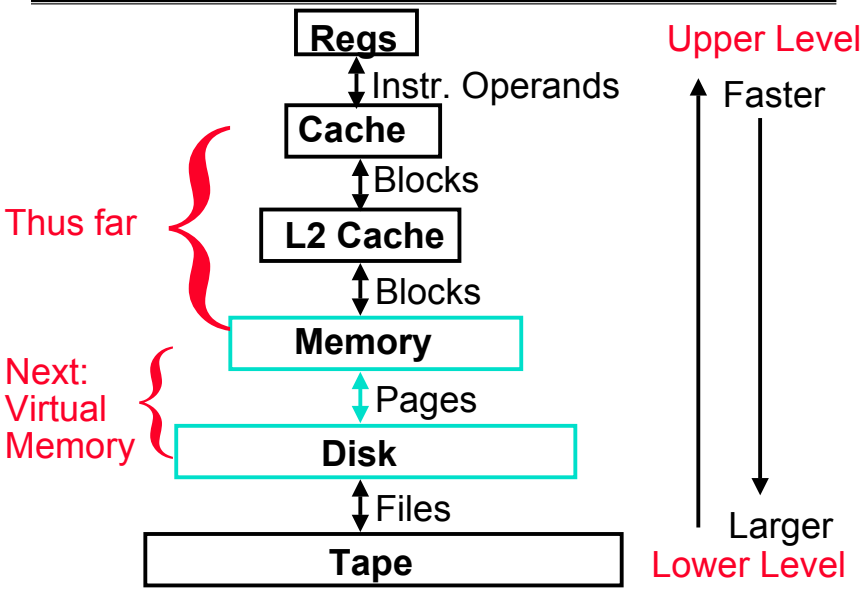
Cache Review (#1/2)

- Caches are NOT mandatory:
  - Processor performs arithmetic
  - Memory stores instructions & data
  - Caches simply make things go *faster*
- Each level of memory hierarchy is just a subset of next higher level
- Caches speed up due to Temporal Locality: store data used recently
- Block size > 1 word speeds up due to Spatial Locality: store words adjacent to the ones used recently

Cache Review (#2/2)

- Cache design choices:
  - size of cache: speed vs. capacity
  - direct-mapped vs. associative
  - for N-way set assoc: choice of N
  - block replacement policy
  - 2nd level cache?
  - Write through vs. write back?
- Use performance model to pick between choices, depending on programs, technology, budget, ...

Another View of the Memory Hierarchy

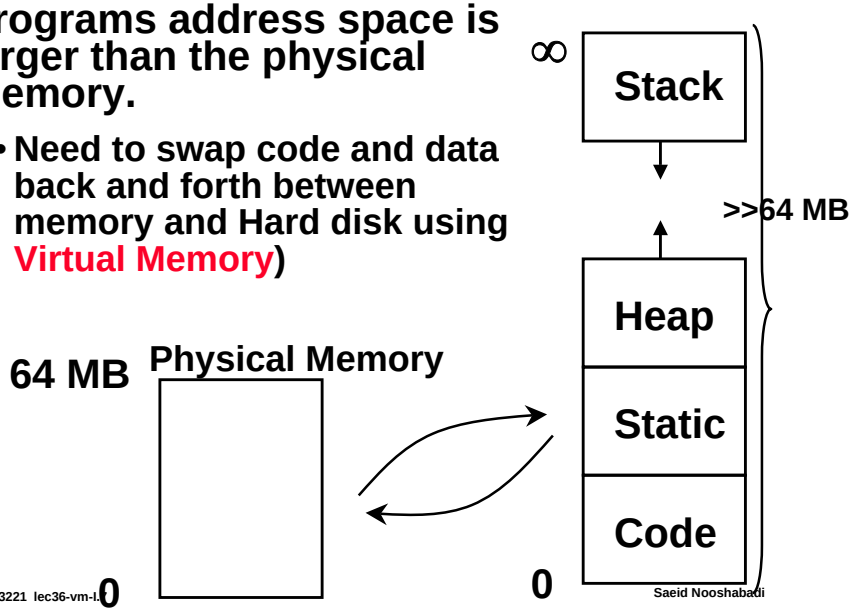


Virtual Memory

- If Principle of Locality allows caches to offer (usually) speed of cache memory with size of DRAM memory, then why not, recursively, use at next level to give speed of DRAM memory, size of Disk memory?
- Called "Virtual Memory"
  - Also allows OS to share memory, protect programs from each other
  - Today, more important for protection vs. just another level of memory hierarchy
  - Historically, it predates caches

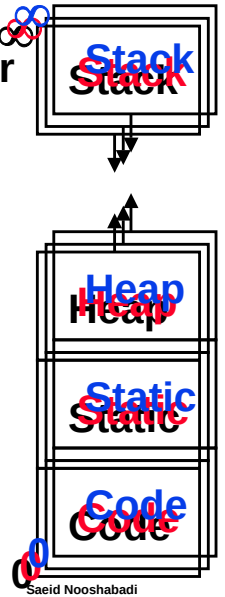
Problems Leading to Virtual Memory (#1/2)

- Programs address space is larger than the physical memory.
  - Need to swap code and data back and forth between memory and Hard disk using Virtual Memory)



Problems Leading to Virtual Memory (#2/2)

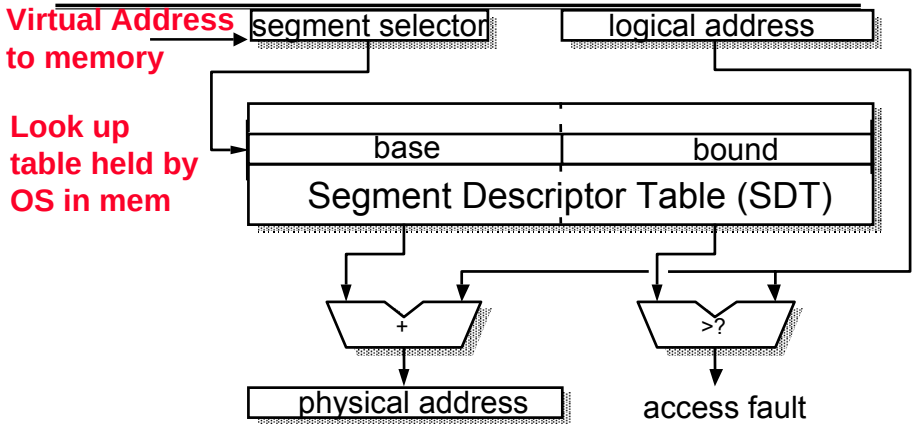
- Many Processes (programs) active at the same time. (Single Processor - many Processes)
  - Processor appears to run multiple programs all at once by rapidly switching between active programs.
  - The rapid switching is managed by Memory Management Unit (MMU) by using Virtual Memory concept.
- Each program sees the entire address space as its own.
- How to avoid multiple programs overwriting each other.



Segmentation Solution

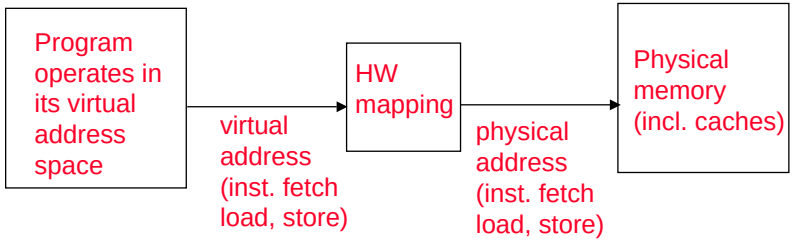
- Segmentation provides simple MMU
  - Program views its memory as set of segments. Code segment, Data Segment, Stack segment, etc.
  - Each program has its own set of private segments.
  - Each access to memory is via a segment selector and offset within the segment.
  - It allows a program to have its own private view of memory and to coexist transparently with other programs in the same memory space.

Segmentation Memory Management Unit



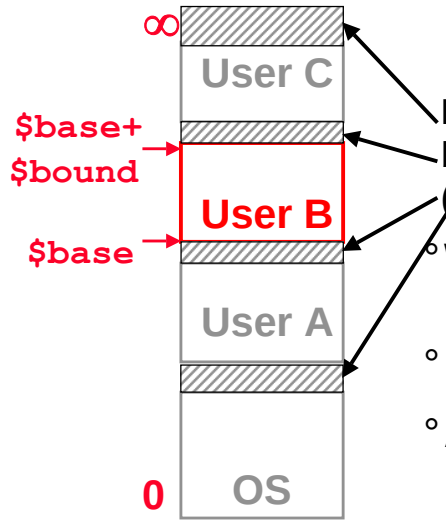
- **Base:** The base address of the segment
- **Logical address:** an offset within a segment
- **Bound:** Segment limit
- **SDT:** Holds Access and other information about the segment

Virtual to Physical Addr. Translation



- Each program operates in its own virtual address space;
- Each is protected from the other
- OS can decide where each goes in memory
- Hardware (HW) provides virtual -> physical mapping

Simple Example: Base and Bound Reg

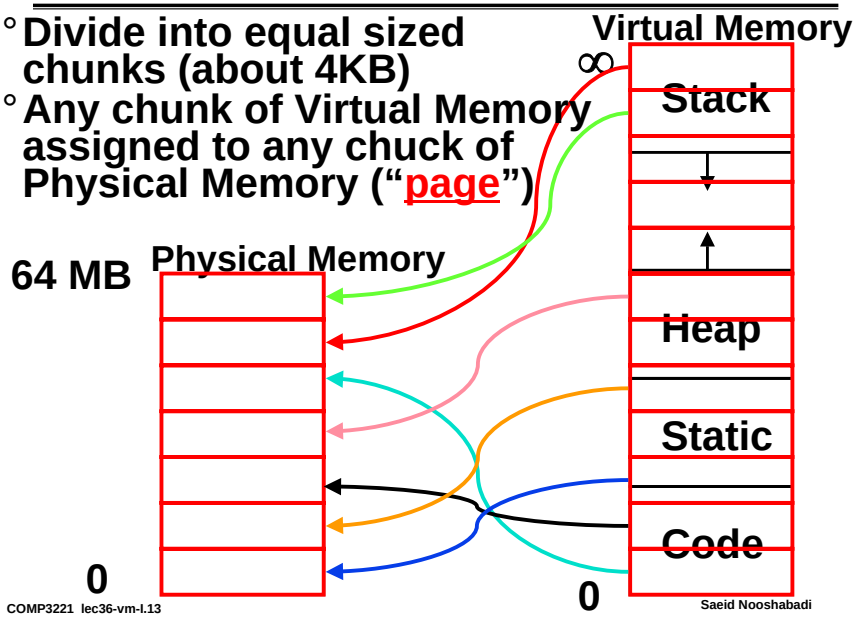


Enough space for User D, but discontinuous ("fragmentation problem")

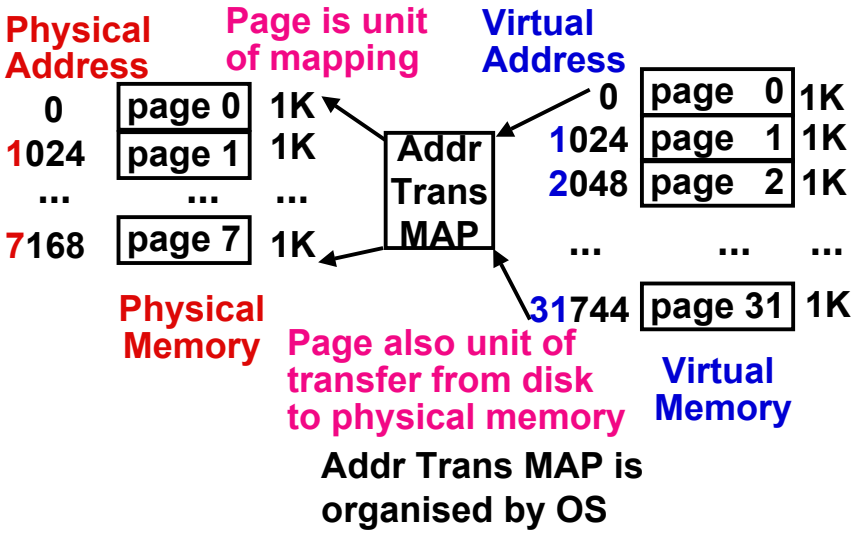
- Want discontinuous mapping
- Process size >> mem
- Addition not enough!

Mapping Virtual Memory to Physical Memory

- Divide into equal sized chunks (about 4KB)
- Any chunk of Virtual Memory assigned to any chuck of Physical Memory ("page")



Paging Organization (assume 1 KB pages)



Addr Trans MAP is organised by OS

Virtual Memory Mapping Function

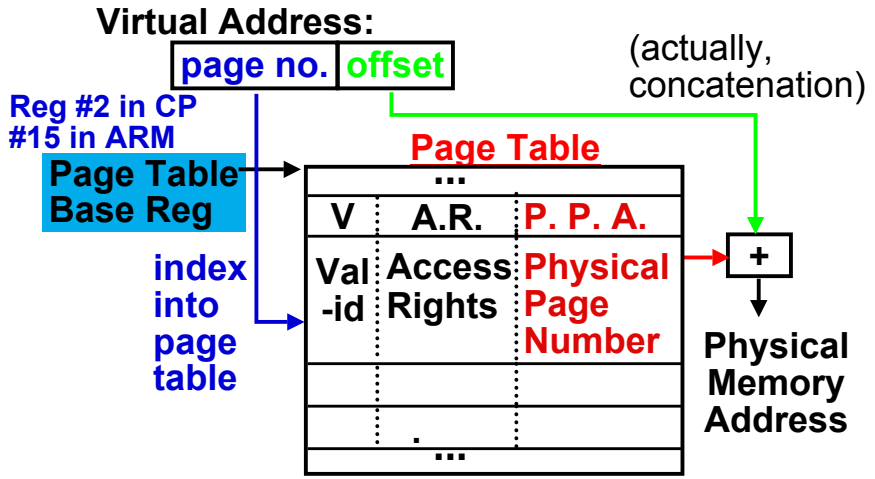
- Cannot have simple function to predict arbitrary mapping
- Use table lookup of mappings

Page Number	Offset

  - Use table lookup ("Page Table") for mappings: Page number is index
  - Virtual Memory Mapping Function
    - Physical Offset = Virtual Offset
    - Physical Page Number = PageTable[Virtual Page Number]

(P.P.N. also called "Page Frame")

Address Mapping: Page Table



Page Table located in physical memory

Page Table

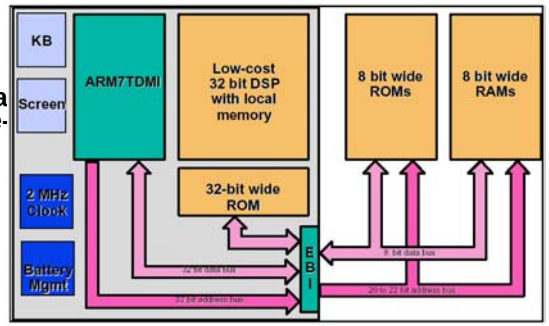
- A page table is an operating system structure which contains the mapping of virtual addresses to physical locations
  - There are several different ways, all up to the operating system, to keep this data around
- Each process running in the operating system has its own page table
  - "State" of process is PC, all registers, plus page table
  - OS changes page tables by changing contents of Page Table Base Register

Reading Material

- Steve Furber: ARM System On-Chip; 2nd Ed, Addison-Wesley, 2000, ISBN: 0-201-67519-6. Chapter 10.

Smart Mobile Phones

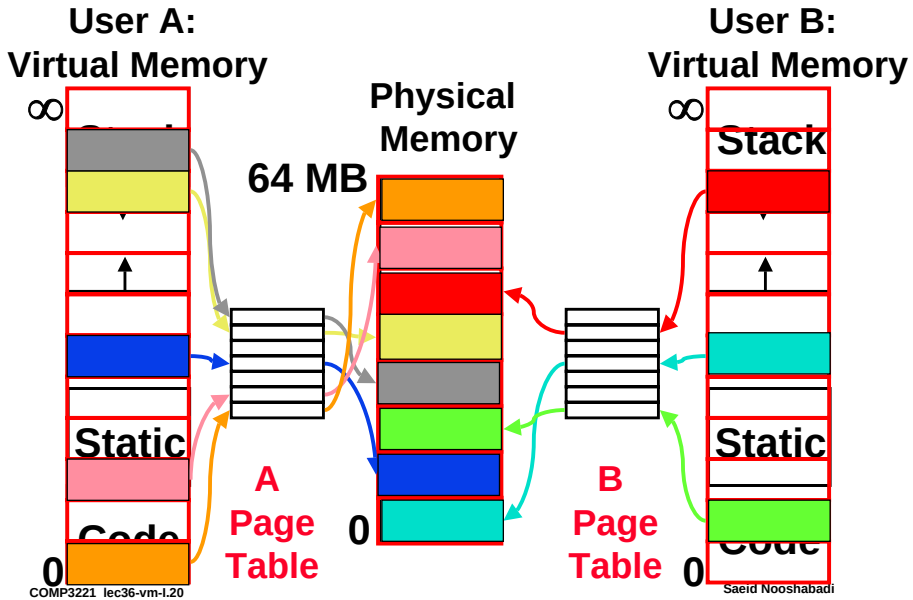
- The Nokia's Series 60 Platform:
  - software product for smart phones that Nokia licenses to other mobile-handset manufacturers.
  - runs on top of the Symbian OS.
  - The Series 60 Platform includes mobile
    - browsing,
    - multimedia messaging and content downloading,
    - personal information management and telephony applications.
    - software platform includes a complete and modifiable user interface library.



- Licensees: Panasonic Mobile Communications, Samsung, Sendo, and Siemens (60% of market

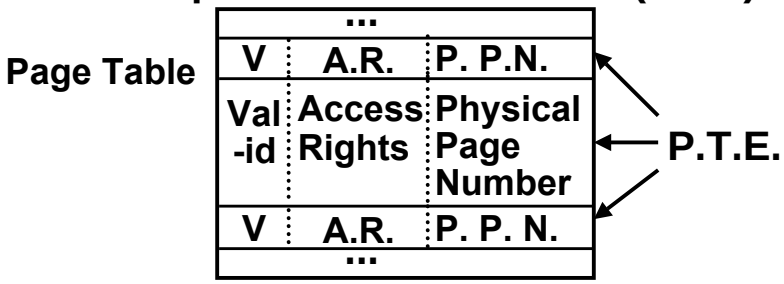
•ARM PrimeXsys tools supplies the suite of pre-validated hardware abd software

Paging/Virtual Memory for Multiple Pocesses



Page Table Entry (PTE) Format

- Contains either Physical Page Number or indication not in Main Memory
- OS maps to disk if Not Valid (V = 0)



- If valid, also check if have permission to use page: Access Rights (A.R.) may be Read Only, Read/Write, Executable

Things to Remember

- Apply Principle of Locality Recursively
- Manage memory to disk? Treat as cache
  - Included protection as bonus, now critical
  - Use Page Table of mappings vs. tag/data in cache
- Virtual Memory allows protected sharing of memory between processes with less swapping to disk, less fragmentation than always swap or base/bound
- Virtual Memory allows protected sharing of memory between processes with less swapping to disk, less fragmentation than always swap or base/bound in Segmentation