

Practical Memory Management

Jonathan Windle

University of East Anglia

J.Windle@uea.ac.uk

June 10, 2017

Overview I

1 Memory Management Unit

- Provides
- Segment/page table
- Mapping

2 Paging

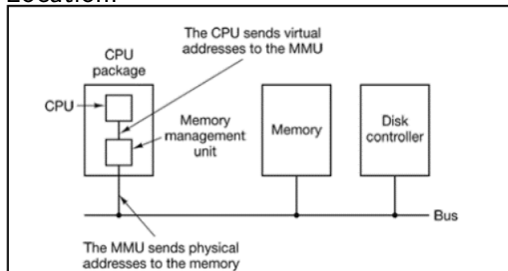
- Page Table Entries
- MMU registers for Paged Virtual Memory
- Page Faults

3 Working Sets

- Working Sets Example
- Pagin and Segmentation

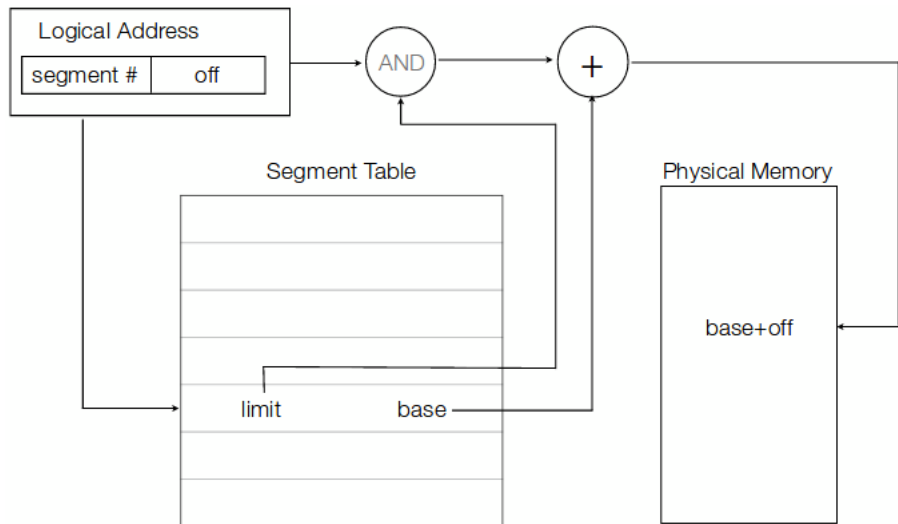
Memory Management Unit (MMU)

- MMU is hardware that provides the translation from a virtual address to a physical address
- Works in conjunction with OS memory manager that allocates/deallocates memory
- MMU sits between CPU and system bus
- MMU allows implementation of virtual/logical addresses
- Location:

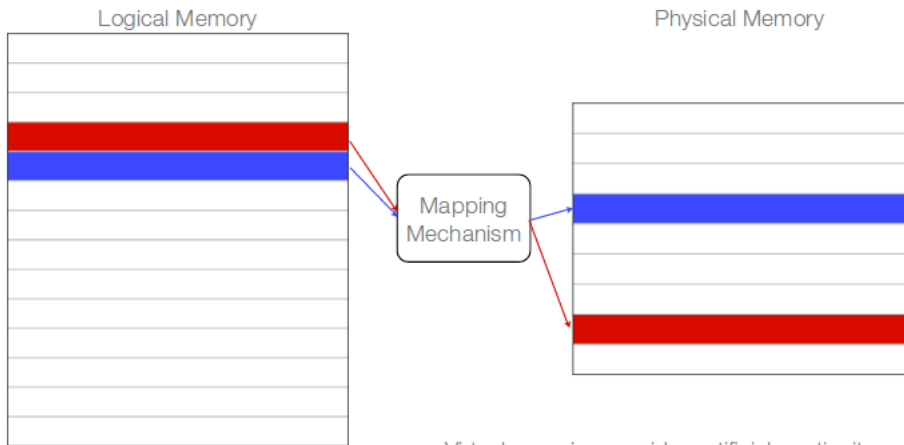


- Non-Contiguous memory segmentation by maintaining segment table for each process
- A segment index and address of each segment
- Segment table is really just an array:
 - Segment number is index into array
 - Contents are base and limit values

Segment/page table



Mapping



Virtual mapping provides artificial contiguity.

Paging

- Need to efficiently map between address spaces
- Cannot map individual locations, would take more memory than is physically available
- Instead system tracks blocks of memory
- Virtual memory is divided into pages
- Physical memory is divided into page frames
- Typical page sizes range from 512Bytes to 16KB.
- Each process has a page table:
 - An array of page table entries
 - One entry for each page in memory
- A page table entry is typically a 32-bit word
- Allows memory to be protected
- Process can address only pages in page table
- Switching processes means switching only page table address in MMU.

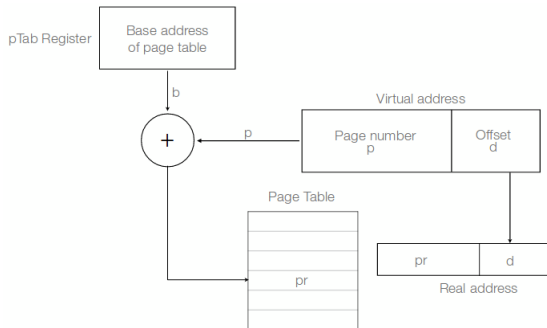
Page Table Entries

- Elements in the page table entry:
 - **pFNo**: Page frame number
 - **pAccess**: Protection information
 - **pAvail**: Present/Absent flag
 - **pUsed**: Referenced flag
 - **pDirty**: Modified flag
- For each memory reference, the bits of a page are tested to ensure access is valid
- All tests are done in parallel in hardware

Cache	pAccess	pAvail	pUsed	pDirty	pFNo
0	R/W	T	T	F	0x0012

MMU registers for Paged Virtual Memory

- MMU must be able to locate pages in memory
- For this it uses:
 - **pTab**: Page table (real) address for current process
 - **pTabSize**: Number of pages in virtual address space
- Demand Paged Virtual Memory:



Page Faults

[a;llowpagebreaks]

- pAvail signifies if a page is available in main memory
 - CPU cannot access secondary memory directly
 - How does it read from page?
- MMU generates a page fault
- CPU is interrupted and required page is swapped in.
- Control is transferred to scheduler via a trap
- Faulted process is blocked
- Control transferred to memory manager
- DMA used to transfer requested page
- Memory manager selects a frame to store the page.
- If no page frames are empty, one must be replaced
- Possiblepage replacement strategies
 - Least frequently used
 - Least recently used
 - First-in, First-out
- Least frequently used is most favoured
- MEmory manager checks pDirty bit for old page, if set.

Working Sets

- The working set is the active pages over a period of time
 - Usually much less than entire virtual address space
 - Page numbers change gradually over time
- Memory manager imposes a fixed working set size:
 - size is important, to avoid thrashing
 - Excessive disc activity introduces bottlenecks
- Can schedule process only if all working set is available

Working Sets Example

- Suppose:
 - A process has 8 virtual pages
 - Memory manager allocates a working set of 4
 - Page frames in working set are initially empty
 - Memory manager uses least-frequently-used policy
- Assume following sequence of page references:
 - 0,2,4,6,2,1,4,3,2,7,5,3,2,7,6,3,7,2

Fault Count	LRU	Pages loaded after request.				Seq before next fault.
1	-	0	-	-	-	0
2	-	0	2	-	-	2
3	-	0	2	4	-	4
4	-	0	2	4	6	6 2
5	0	1	2	4	6	1 4
6	6	1	2	4	3	3 2
7	1	7	2	4	3	7
8	4	7	2	5	3	5 3 2 7
9	5	7	2	6	3	6 3 7 2

0 2 4 6 2 1 4 3 2 7 5 3 2 7 6 3 7 2

Paging and Segmentation

- Fixed block sizes are referred to as pages
- Variable block sizes are referred to as segments
 - Concept of virtual address being formed of a segment ID and an offset is similar
- Possible to combine both paging and segmentation
- Also possible to have multi-level page tables

Summary

- Programs need to be bound to memory:
 - Some instructions aren't complete without address
 - Three situations: compile time, load time, run time
- Memory must be protected and shared
 - MMU enables this
 - Process operates in logical address space
 - MMU translates logical address to physical address.

The End