ELEC 377 – Operating Systems

Week 7 – Class 1

Last Class

- Virtual Memory
- ◊ Concept
- ◊ Demand Paging

Hierarchical Tables Example

- Program with 222k of Code and Data, 30K of Stack
  - ◊ 32 bit address space, 1k pages, p1 is 12 bits
    - how much space is taken by the page tables??
    - assume 4 bytes for each entry of p1 table and 4 bytes for each entry of each p2 table.

    p2 = 10 bits

how many pages of code and data? -> 222 pages

how many pages for stack? -> 30 pages

(Given and used below) How many p2 tables? -> 2 tables

(Given and used below) How many p1 tables? -> 1 table

size of p1 table = $2^{12} * 4 = 2^{14}$

size of each p2 table = $2^{10} * 4 = 2^{12}$

Total table space = $2^{14}+2^{12}+2^{12} = 2^{14}+2^{13}$=16k+8k=24k

Today

- Virtual Memory
- ◊ Page Replacement Algorithms
- ◊ Frame Allocation
- ◊ Thrashing
- ◊ Working Set

Page Replacement

- What happens when we run out of memory?
  - ◊ running more processes than memory (*over allocation*)
  - ◊ no spare frames
  - ◊ select some other frame in physical memory
  - ◊ write its contents to disk
  - ◊ invalidate the MMU registers that point to the frame
  - ◊ reuse the frame
- Two transfers (write old contents, read new contents)

Page Replacement

- Dirty Bit?
  - ◊ add another flag to the page table
  - ◊ indicates that the page has been changed (dirty)
  - ◊ only write dirty pages (otherwise matches copy on the disk)

- Code pages are mapped from the program executable
  - ◊ since code doesn’t change (reentrant), never have to write code pages.
  - ◊ Backing store only saves data and stack pages

Page Replacement Algorithms

- Similar to scheduling algorithms
  - ◊ Want to minimize page faults
  - ◊ Each page fault represents 1 or two disk transfers
- FIFO (First In First Out)
  - ◊ Page that is replaced is the oldest page
  - ◊ Not particularly good
  - ◊ Belady's Anomaly
    – more memory, more page faults
- Optimal
  - ◊ similar to Shortest Job First scheduling algorithm
  - ◊ page that will not be used for the longest time
  - ◊ future knowledge - provides a baseline

Page Replacement Algorithms

- LRU - Least recently used
  - ◊ past behavior predicts future behavior
  - ◊ page referenced longest ago gets replaced
    - –hardware support(page counters, stack)
- Approximation
  - ◊ reference bits (history of page references)
  - ◊ second chance algorithm (FIFO with 1 ref bit)
- Alternatives
  - ◊ include modified bit
    - – prefer clean pages to dirty pages
    - – not as important as recently used reference bit, breaks ties

Page Reference Strings

- A list of the pages that are referenced over time
  e.g. 1,2,3,4,1,2,5,1,2,3,4,5

◊ determine how many page faults for each algorithm

◊ draw the page frames allocated

ex. 3 pages, FIFO

1 2 3 4 1 2 5 1 2 3 4 5

1

Page Reference Strings

- A list of the pages that are referenced over time

e.g. 1,2,3,4,1,2,5,1,2,3,4,5

◊ determine how many page faults for each algorithm

◊ draw the page frames allocated

ex. 3 pages, FIFO

1 2 3 4 1 2 5 1 2 3 4 5

1 1
  2

Page Reference Strings

- A list of the pages that are referenced over time

e.g. 1,2,3,4,1,2,5,1,2,3,4,5

◊ determine how many page faults for each algorithm

◊ draw the page frames allocated

ex. 3 pages, FIFO

1 2 3 4 1 2 5 1 2 3 4 5

1 1 1

2 2

3

Page Reference Strings

- A list of the pages that are referenced over time

e.g. 1,2,3,4,1,2,5,1,2,3,4,5

◊ determine how many page faults for each algorithm

◊ draw the page frames allocated

ex. 3 pages, FIFO

---

1 2 3 4 1 2 5 1 2 3 4 5

1	1	1	4
	2	2	2
		3	3

Page Reference Strings

- A list of the pages that are referenced over time

e.g. 1,2,3,4,1,2,5,1,2,3,4,5

◊ determine how many page faults for each algorithm

◊ draw the page frames allocated

ex. 3 pages, FIFO

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	4	4							
	2	2	2	1							
		3	3	3							

Page Reference Strings

- A list of the pages that are referenced over time

e.g. 1,2,3,4,1,2,5,1,2,3,4,5

◊ determine how many page faults for each algorithm

◊ draw the page frames allocated

ex. 3 pages, FIFO

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	4	4	4						
	2	2	2	1	1						
		3	3	3	2						

Page Reference Strings

- A list of the pages that are referenced over time

e.g. 1,2,3,4,1,2,5,1,2,3,4,5

◊ determine how many page faults for each algorithm

◊ draw the page frames allocated

ex. 3 pages, FIFO

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	4	4	4	5					
	2	2	2	1	1	1					
		3	3	3	2	2					

Page Reference Strings

- A list of the pages that are referenced over time

e.g. 1,2,3,4,1,2,5,1,2,3,4,5

◊ determine how many page faults for each algorithm

◊ draw the page frames allocated

ex. 3 pages, FIFO

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	4	4	4	5			5		
	2	2	2	1	1	1			3		
		3	3	3	2	2			2		

Page Reference Strings

- A list of the pages that are referenced over time

e.g. 1,2,3,4,1,2,5,1,2,3,4,5

◊ determine how many page faults for each algorithm

◊ draw the page frames allocated

ex. 3 pages, FIFO

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	4	4	4	5			5	5	
	2	2	2	1	1	1			3	3	
		3	3	3	2	2			2	4	

Page Reference Strings

- A list of the pages that are referenced over time

e.g. 1,2,3,4,1,2,5,1,2,3,4,5

◊ determine how many page faults for each algorithm

◊ draw the page frames allocated

ex. 3 pages, FIFO

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	4	4	4	5			5	5	
	2	2	2	1	1	1			3	3	
		3	3	3	2	2			2	4	

9 page faults

Belady's Anomaly

- ex. 4 pages, FIFO

1 2 3 4 1 2 5 1 2 3 4 5

1	1	1	1
	2	2	2
		3	3
			4

Belady’s Anomaly

- ex. 4 pages, FIFO

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1			5					
	2	2	2			2					
		3	3			3					
			4			4					

Belady's Anomaly

- ex. 4 pages, FIFO

1 2 3 4 1 2 5 1 2 3 4 5

1	1	1	1			5	5				
	2	2	2			2	1				
		3	3			3	3				
			4			4	4				

Belady's Anomaly

- ex. 4 pages, FIFO

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1			5	5	5			
	2	2	2			2	1	1			
		3	3			3	3	2			
			4			4	4	4			

Belady’s Anomaly

- ex. 4 pages, FIFO

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1			5	5	5	5		
	2	2	2			2	1	1	1		
		3	3			3	3	2	2		
			4			4	4	4	3		

Belady’s Anomaly

- ex. 4 pages, FIFO

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1			5	5	5	5	4	
	2	2	2			2	1	1	1	1	
		3	3			3	3	2	2	2	
			4			4	4	4	3	3	

Belady's Anomaly

- ex. 4 pages, FIFO

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1			5	5	5	5	4	4
	2	2	2			2	1	1	1	1	5
		3	3			3	3	2	2	2	2
			4			4	4	4	3	3	3

Belady's Anomaly

- ex. 4 pages, FIFO

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1			5	5	5	5	4	4
	2	2	2			2	1	1	1	1	5
		3	3			3	3	2	2	2	2
			4			4	4	4	3	3	3

10 page faults

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, Optimal

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

---

7 7 7
0 0
1

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, Optimal

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

---

7 7 7 2
  0 0 0
    1 1

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, Optimal

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7 7 7 2
0 0 0
1 1

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, Optimal

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2		2														
	0	0	0		0														
		1	1		3														

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, Optimal

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7	7	7	2		2
	0	0	0		0
		1	1		3

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, Optimal

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2		2		2												
	0	0	0		0		4												
		1	1		3		3												

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, Optimal

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1	4
7	7	7	2		2		2													
	0	0	0		0		4													
		1	1		3		3													

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, Optimal

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2		2		2			2									
	0	0	0		0		4			0									
		1	1		3		3			3									

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, Optimal

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1	3
7	7	7	2		2		2			2										
	0	0	0		0		4			0										
		1	1		3		3			3										

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, Optimal

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2		2		2			2			2						
	0	0	0		0		4			0			0						
		1	1		3		3			3			1						

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, Optimal

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1 2

7	7	7	2	2	2	2	2
	0	0	0	0	4	0	0
		1	1	3	3	3	1

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, Optimal

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2		2		2			2			2				7		
	0	0	0		0		4			0			0				0		
		1	1		3		3			3			1				1		

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, Optimal

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2		2		2			2			2				7		
	0	0	0		0		4			0			0				0		
		1	1		3		3			3			1				1		

9 page faults

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, LRU

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

---

7 7 7
  0 0
    1

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, LRU

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7	7	7	2
	0	0	0
		1	1

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, LRU

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

---

7 7 7 2

0 0 0

1 1

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, LRU

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2		2														
	0	0	0		0														
		1	1		3														

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, LRU

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

---

7 7 7 2   2
   0 0 0   0
      1 1   3

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, LRU

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7	7	7	2		2		4
	0	0	0		0		0
		1	1		3		3

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, LRU

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7	7	7	2		2	4
	0	0	0		0	0
		1	1		3	3

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, LRU

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2		2		4	4											
	0	0	0		0		0	0											
		1	1		3		3	2											

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, LRU

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2		2		4	4											
	0	0	0		0		0	0											
		1	1		3		3	2											

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, LRU

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2		2		4	4	4										
	0	0	0		0		0	0	3										
		1	1		3		3	2	2										

Page Reference Strings

e.g. 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

ex. 3 pages, LRU

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2		2		4	4	4	0			1		1		1		
	0	0	0		0		0	0	3	3			3		0		0		
		1	1		3		3	2	2	2			2		2		7		

12 page faults

- Both Optimal and LRU are stack algorithms

◊ set of resident pages for n frames is always a subset of the set of resident pages for n+1 frames.

Frame Allocation

- Minimum Number of Frames is Architecture Dependent
  - ◊ Instructions may straddle two pages
  - ◊ Data referenced by instruction may straddle two or more pages
  - ◊ Indirection may also require more pages
- How do we allocate to more than one process??
  - ◊ equal allocation?
  - ◊ proportional allocation
    - – number of frames proportional to size of process
    - – priority?

Frame Allocation

- global vs local
- local
  - ◊ choose from a frame already owned by process
  - ◊ number of frames remains fixed!!
  - ◊ cannot surrender extra pages
- global
  - ◊ processes can steal frames from other processes
  - ◊ allows pages to be reshuffled
  - ◊ more commonly used

Thrashing

- amount of memory less than locality of reference
  - ◊ always paging out a page that you will need very soon!
  - ◊ CPU utilization goes down (lots of I/O)
  - ◊ OS adds another process

- Thrashing
  - ◊ high paging activity, low CPU utilization
  - ◊ system spends all its time swapping pages

Thrashing

- locality of reference (related storage locations being frequently accessed) - changes over time
- reasonable minimum number of frames
  - ◊ not absolute minimum
  - ◊ loops + functions called from loop

```
…
int x,y;
…
while (x) {
    x = f(y);
    y = y+1;
}
…
int f(int){
    …
}
```

Working Set

- amount of memory for locality of reference
  - ◊ given a period of time (# of memory references)
    – working set window
  - ◊ The pages referenced over that time
  - ◊ if the window is too small, then working set is not entire locality of reference
  - ◊ if window is too large, then more than locality

- Thrashing
  - ◊ If the total size of all of the working sets is bigger than available memory
  - ◊ suspend one of the processes

Page Fault Frequency

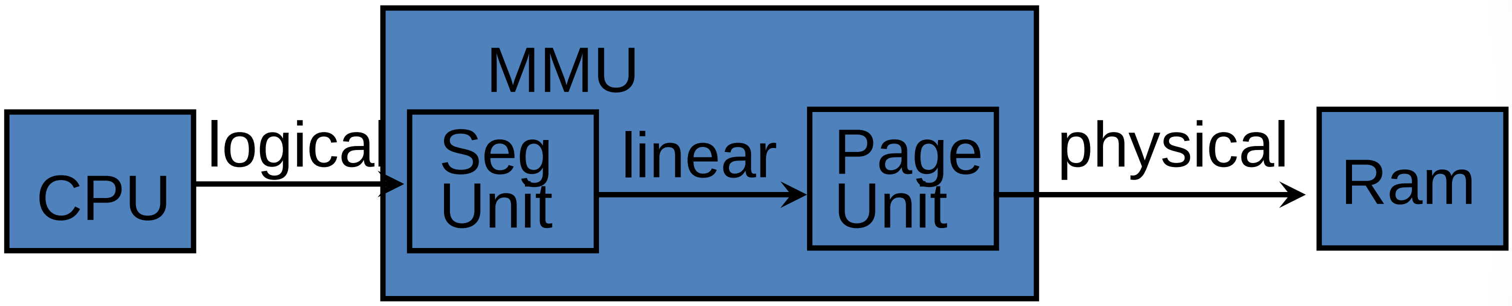
- instead of calculating working set, look at effect
  - ◊ thrashing – high level of paging activity

- Monitor page fault frequency of processes
  - ◊ set min and max rate for page faults
  - ◊ if page fault rate is too small, then take frames away
  - ◊ if page fault rate is too large, then give process more frames
  - ◊ may have to suspend another process to give more frames – over commitment of memory

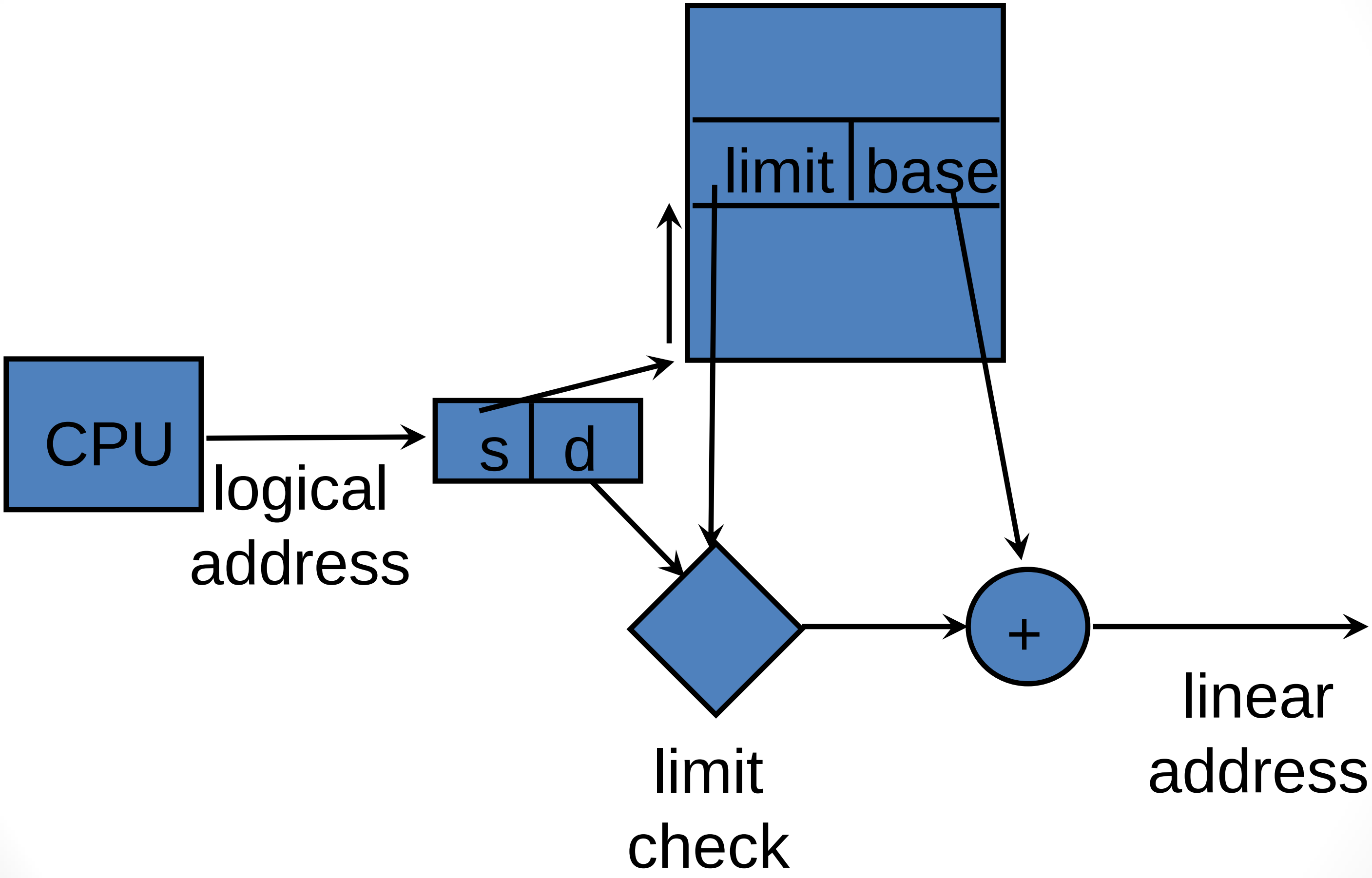
Segmentation

- not the same as previous segmentation
  - ◊ not: text segment, data segment stack segment
- A segment is a separate logical address space.
  - ◊ used for various elements of the program process
    - code, variables, heap, stack, shared libraries
      -- all get their own segment...
    - different logical entities
  - ◊ Segments can be different sizes/different permissions (execute, modify)
  - ◊ Segment address space is translated to a linear logical (virtual) address.
  - ◊ Linear logical address is passed to the paging unit

Segmentation



Segmentation



Segmentation

- Used in some embedded systems
- Used in AS/400 (to some extent)

- Segment values are often stored in segment registers.

- Not really used in many consumer systems
  C language
  Pointers are castable, single address space model

Segmentation

- Intel Architecture has Segmentation
  - required
    - ◊ stack operations access stack segment
    - ◊ instruction fetches use code segment
    - ◊ memory register loads use data segment
  - one set of segments for OS
  - one set of segments for each user programs
  - all overlay each other
    - not a separate address space for each segment.
    - can't tell in advance that pointer is code/data/stack