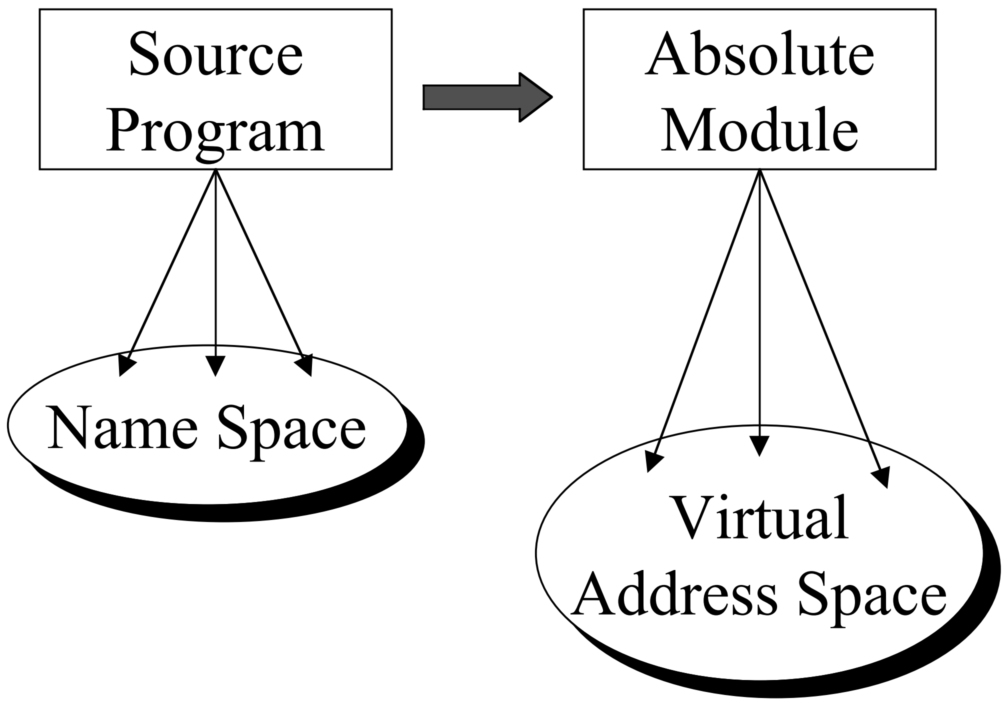
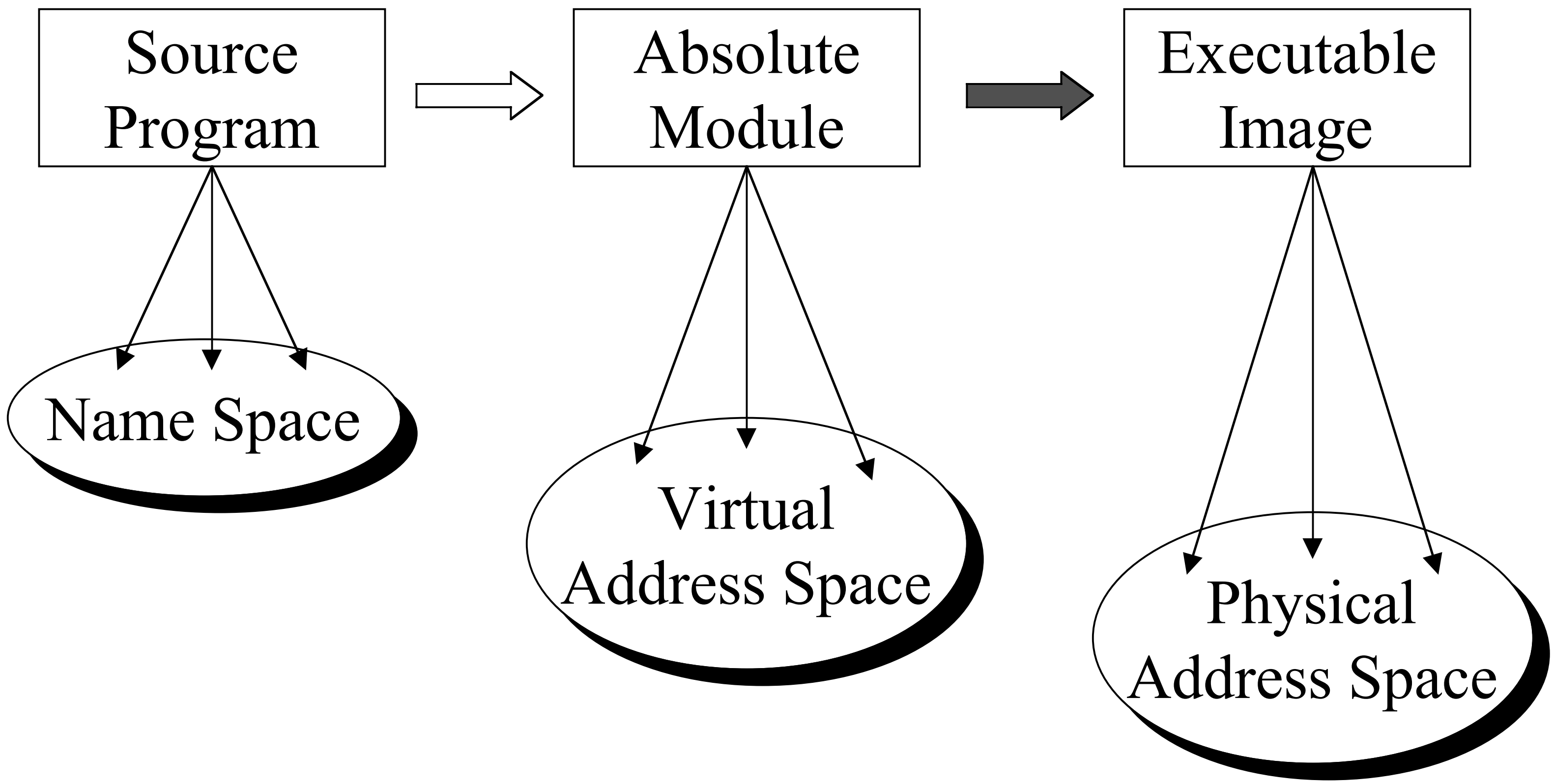
Virtual Memory

Names, Virtual Addresses & Physical Addresses



Compile/Link tools

Names, Virtual Addresses & Physical Addresses



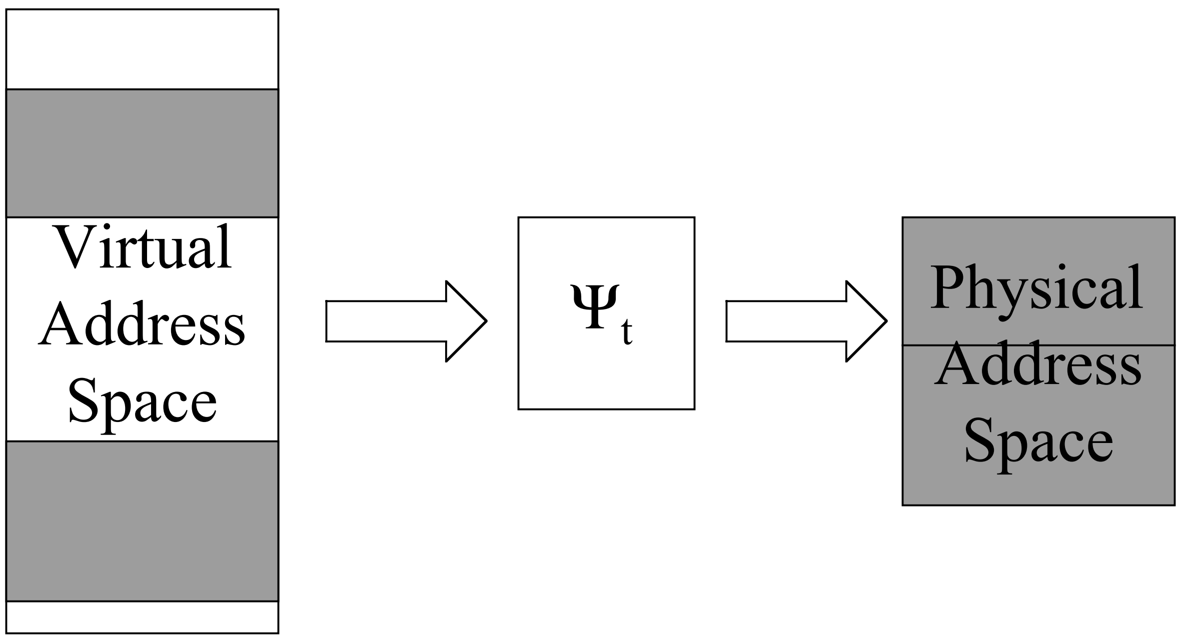
Ψ_t: Virtual Address Space $\rightarrow$ Physical Address Space

Virtual Memory

- Uses dynamic address relocation/binding
  - Generalization of base-limit registers
  - Physical address corresponding to a compile-time address is not known until run time
- Idea is that *only part of the address space is loaded* as process executes
- This works because of program and data *locality*

Virtual Memory (cont)

- Use a dynamic virtual address map, Ψ_t



Address Formation

- Translation system produces an address space, but address are *virtual* instead of physical
- A virtual address, x:
  - Is mapped to $y = \Psi_t(x)$ if x is loaded at physical address y
  - Is mapped to Ω if x is not loaded
- The map changes as the program executes
- Ψ_t: Virtual Address $\rightarrow$ Physical Address $\cup \{\Omega\}$

Size of Blocks of Memory

- Fixed size: *Pages* are moved back and forth between primary and secondary memory
- Variable size: Programmer-defined *segments* are the unit of movement
- Paging is the commercially dominant form of virtual memory today

Paging

- A *page* is a fixed sized block of virtual addresses
- A *page frame* is a fixed size block of physical memory, the same size as a page
- When a virtual address in page i is referenced by the CPU
  - If page i is loaded at page frame j, the virtual address is relocated to page frame j
  - If page is not loaded, the OS interrupts the process and loads the page into a page frame

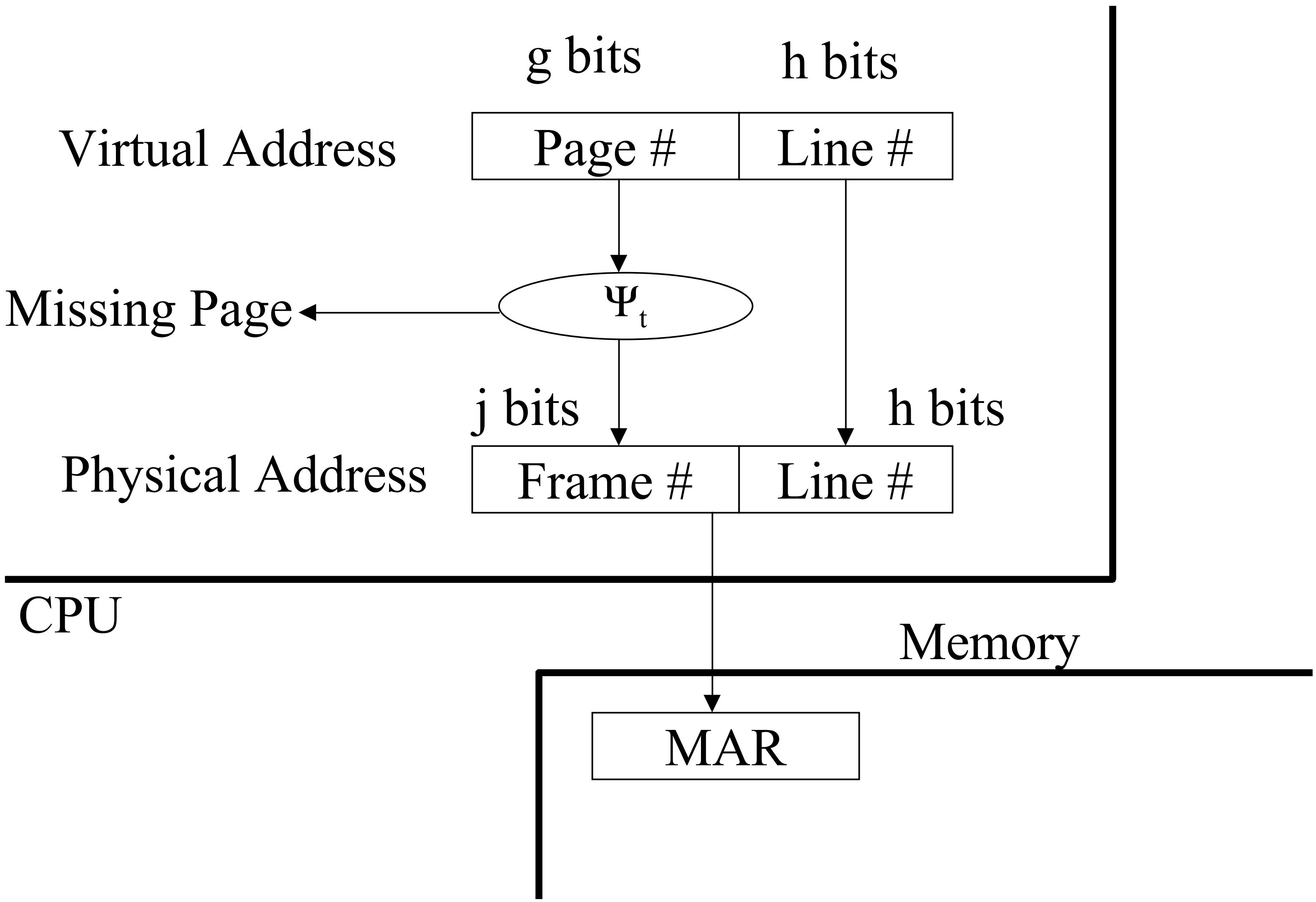
Addresses

- Suppose there are $G=2^{g+h}$ virtual addresses and $H=2^{j+h}$ physical addresses
  - Each page/page frame is 2^h addresses
  - There are 2^g pages in the virtual address space
  - 2^j page frames are allocated to the process
  - Rather than map individual addresses, Ψ_t maps the 2^g pages to the 2^j page frames

Address Translation

- Let $N = \{d_0, d_1, \ldots d_{n-1}\}$ be the pages
- Let $M = \{b_0, b_1, \ldots, b_{m-1}\}$ be page frames
- Virtual address, i, satisfies $0 \leq i < G = 2^{g+h}$
- Physical address, $k = U2^h + V$ $(0 \leq V < G = 2^h)$
  - U is page frame number
  - V is the line number within the page
  - $\Psi_t : [0:G-1] \rightarrow \langle U, V \rangle \cup \{\Omega\}$
  - Since every page is size $c=2^h$
    - page number $= U = \lfloor i/c \rfloor$
    - line number $= V = i \bmod c$

Address Translation (cont)



Demand Paging

- Page fault occurs
- Process with missing page is interrupted
- Memory manager locates the missing page
- Page frame is unloaded (replacement policy)
- Page is loaded in the vacated page frame
- Page table is updated
- Process is restarted

Modeling Page Behavior

- Let $\omega = r_1, r_2, r_3, \ldots, r_i, \ldots$ be a ***page reference stream***
  - r_i is the i^{th} page # referenced by the process
  - The subscript is the ***virtual time*** for the process
- Given a page frame allocation of m, the memory state at time t, $S_t(m)$, is set of pages loaded
  - $S_t(m) = S_{t-1}(m) \cup X_t - Y_t$
    - X_t is the set of fetched pages at time t
    - Y_t is the set of replaced pages at time t

More on Demand Paging

- If r_t was loaded at time t-1, $S_t(m) = S_{t-1}(m)$
- If r_t was not loaded at time t-1 and there were empty page frames
  - $S_t(m) = S_{t-1}(m) \cup \{r_t\}$
- If r_t was not loaded at time t-1 and there were no empty page frames
  - $S_t(m) = S_{t-1}(m) \cup \{r_t\} - \{y\}$
- The alternative is *prefetch* paging

Static Allocation, Demand Paging

- Number of page frames is static over the life of the process
- Fetch policy is demand
- Since $S_t(m) = S_{t-1}(m) \cup \{r_t\} - \{y\}$, the replacement policy <u>must choose y</u> -- which uniquely identifies the paging policy

Random Replacement

- Replaced page, y, is chosen from the m loaded page frames with probability 1/m

Let page reference stream, ϖ = `2031203120316457`

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

Random Replacement

- Replaced page, y, is chosen from the m loaded page frames with probability 1/m

Let page reference stream, ϖ = 203120312031645 7

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2													
1		$\underline{0}$	0													
2			$\underline{3}$													

Random Replacement

- Replaced page, y, is chosen from the m loaded page frames with probability 1/m

Let page reference stream, ϖ = 2031203120316457

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

Random Replacement

- Replaced page, y, is chosen from the m loaded page frames with probability 1/m

Let page reference stream, ϖ = 2031203120316457

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

Random Replacement

- Replaced page, y, is chosen from the m loaded page frames with probability 1/m

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2	2	2	2	$\underline{3}$									
1		$\underline{0}$	0	$\underline{1}$	1	1	1									
2			$\underline{3}$	3	3	$\underline{0}$	0									

Random Replacement

- Replaced page, y, is chosen from the m loaded page frames with probability 1/m

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2	2	2	2	$\underline{3}$	3	3							
1		$\underline{0}$	0	$\underline{1}$	1	1	1	1	1							
2			$\underline{3}$	3	3	$\underline{0}$	0	0	$\underline{2}$							

Random Replacement

- Replaced page, y, is chosen from the m loaded page frames with probability 1/m

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2	2	2	2	$\underline{3}$	3	3	$\underline{0}$						
1		$\underline{0}$	0	$\underline{1}$	1	1	1	1	1	1						
2			$\underline{3}$	3	3	$\underline{0}$	0	0	$\underline{2}$	1						

Random Replacement

- Replaced page, y, is chosen from the m loaded page frames with probability 1/m

Let page reference stream, ϖ = 2031203120316457

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

Random Replacement

- Replaced page, y, is chosen from the m loaded page frames with probability 1/m

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2	2	2	2	$\underline{3}$	3	3	$\underline{0}$	0	0	0			
1		$\underline{0}$	0	$\underline{1}$	1	1	1	1	1	1	$\underline{3}$	3	$\underline{6}$			
2			$\underline{3}$	3	3	$\underline{0}$	0	0	$\underline{2}$	1	1	1	1			

Random Replacement

- Replaced page, y, is chosen from the m loaded page frames with probability 1/m

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2	2	2	2	$\underline{3}$	3	3	$\underline{0}$	0	0	0	$\underline{4}$		
1		$\underline{0}$	0	$\underline{1}$	1	1	1	1	1	1	$\underline{3}$	3	$\underline{6}$	6		
2			$\underline{3}$	3	3	$\underline{0}$	0	0	$\underline{2}$	1	1	1	1	1		

Random Replacement

- Replaced page, y, is chosen from the m loaded page frames with probability 1/m

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2	2	2	2	$\underline{3}$	3	3	$\underline{0}$	0	0	0	$\underline{4}$	4	
1		$\underline{0}$	0	$\underline{1}$	1	1	1	1	1	1	$\underline{3}$	3	$\underline{6}$	6	$\underline{5}$	
2			$\underline{3}$	3	3	$\underline{0}$	0	0	$\underline{2}$	1	1	1	1	1	1	

Random Replacement

- Replaced page, y, is chosen from the m loaded page frames with probability 1/m

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2	2	2	2	$\underline{3}$	3	3	$\underline{0}$	0	0	0	$\underline{4}$	4	$\underline{7}$
1		$\underline{0}$	0	$\underline{1}$	1	1	1	1	1	1	$\underline{3}$	3	$\underline{6}$	6	$\underline{5}$	5
2			$\underline{3}$	3	3	$\underline{0}$	0	0	$\underline{2}$	1	1	1	1	1	1	1

Random Replacement

- Replaced page, y, is chosen from the m loaded page frames with probability 1/m

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2	2	2	2	$\underline{3}$	3	3	$\underline{0}$	0	0	0	$\underline{4}$	4	$\underline{7}$
1		$\underline{0}$	0	$\underline{1}$	1	1	1	1	1	1	$\underline{3}$	3	$\underline{6}$	6	$\underline{5}$	5
2			$\underline{3}$	3	3	$\underline{0}$	0	0	$\underline{2}$	1	1	1	1	1	1	1

13 page faults

- No knowledge of $\varpi \Rightarrow$ not perform well
- Easy to implement

Belady's Optimal Algorithm

- Replace page with maximal forward distance: $y_t = \max_{x \in S_{t-1}(m)} FWD_t(x)$

Let page reference stream, ϖ = 203120312031645 7

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	<u>2</u>	2	2													
1		<u>0</u>	0													
2			<u>3</u>													

Belady's Optimal Algorithm

- Replace page with maximal forward distance: $y_t = \max_{x \in S_{t-1}(m)} \mathrm{FWD}_t(x)$

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2													
1		$\underline{0}$	0													
2			$\underline{3}$													

$\mathrm{FWD}_4(2) = 1$
$\mathrm{FWD}_4(0) = 2$
$\mathrm{FWD}_4(3) = 3$

Belady's Optimal Algorithm

- Replace page with maximal forward distance: $y_t = \max_{x \in S_{t-1}(m)} FWD_t(x)$

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2	2												
1		$\underline{0}$	0	0												
2			$\underline{3}$	$\underline{1}$												

$FWD_4(2) = 1$
$FWD_4(0) = 2$
$FWD_4(3) = 3$

Belady's Optimal Algorithm

- Replace page with maximal forward distance: $y_t = \max_{xeS\ t\text{-}1(m)} FWD_t(x)$

Let page reference stream, ϖ = 2031203120316457

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

Belady's Optimal Algorithm

- Replace page with maximal forward distance: $y_t = \max_{x \in S_{t-1}(m)} FWD_t(x)$

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2	2	2	2	2									
1		$\underline{0}$	0	0	0	0	$\underline{3}$									
2			$\underline{3}$	$\underline{1}$	1	1	1									

$FWD_7(2) = 2$
$FWD_7(0) = 3$
$FWD_7(1) = 1$

Belady's Optimal Algorithm

- Replace page with maximal forward distance: $y_t = \max_{x \in S_{t-1}(m)} FWD_t(x)$

Let page reference stream, ϖ = `2031203120316457`

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2	2	2	2	2	2	2	$\underline{0}$						
1		$\underline{0}$	0	0	0	0	$\underline{3}$	3	3	3						
2			$\underline{3}$	$\underline{1}$	1	1	1	1	1	1						

$FWD_{10}(2) = \infty$

$FWD_{10}(3) = 2$

$FWD_{10}(1) = 3$

Belady's Optimal Algorithm

- Replace page with maximal forward distance: $y_t = \max_{xeS\ t\text{-}1(m)} \mathrm{FWD}_t(x)$

Let page reference stream, ϖ = 2031203120316457

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

$\mathrm{FWD}_{13}(0) = \infty$
$\mathrm{FWD}_{13}(3) = \infty$
$\mathrm{FWD}_{13}(1) = \infty$

Belady's Optimal Algorithm

- Replace page with maximal forward distance: $y_t = \max_{xeS\ t-1(m)} FWD_t(x)$

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2	2	2	2	2	2	2	$\underline{0}$	0	0	0	$\underline{4}$	4	4
1		$\underline{0}$	0	0	0	0	$\underline{3}$	3	3	3	3	3	$\underline{6}$	6	6	$\underline{7}$
2			$\underline{3}$	$\underline{1}$	1	1	1	1	1	1	1	1	1	1	$\underline{5}$	5

10 page faults

- Perfect knowledge of $\varpi \Rightarrow$ perfect performance
- Impossible to implement

Least Recently Used (LRU)

- Replace page with maximal forward distance: $y_t = \max_{x \in S_{t-1}(m)} BKWD_t(x)$

Let page reference stream, ϖ = `203120312031 6457`

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	<u>2</u>	2	2													
1		<u>0</u>	0													
2			<u>3</u>													

$BKWD_4(2) = 3$
$BKWD_4(0) = 2$
$BKWD_4(3) = 1$

Least Recently Used (LRU)

- Replace page with maximal forward distance: $y_t = \max_{x \in S_{t-1}(m)} BKWD_t(x)$

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2	$\underline{1}$												
1		$\underline{0}$	0	0												
2			$\underline{3}$	3												

$BKWD_4(2) = 3$
$BKWD_4(0) = 2$
$BKWD_4(3) = 1$

Least Recently Used (LRU)

- Replace page with maximal forward distance: $y_t = \max_{x \in S_{t-1}(m)} BKWD_t(x)$

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2	$\underline{1}$	1											
1		$\underline{0}$	0	0	$\underline{2}$											
2			$\underline{3}$	3	3											

$BKWD_5(1) = 1$
$BKWD_5(0) = 3$
$BKWD_5(3) = 2$

Least Recently Used (LRU)

- Replace page with maximal forward distance: $y_t = \max_{x \in S_{t-1}(m)} BKWD_t(x)$

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	<u>2</u>	2	2	<u>1</u>	1	1										
1		<u>0</u>	0	0	<u>2</u>	2										
2			<u>3</u>	3	3	<u>0</u>										

$BKWD_6(1) = 2$
$BKWD_6(2) = 1$
$BKWD_6(3) = 3$

Least Recently Used (LRU)

- Replace page with maximal forward distance: $y_t = \max_{x \in S_{t-1}(m)} BKWD_t(x)$

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2	$\underline{1}$	1	1	$\underline{3}$	3	3	$\underline{0}$	0	0	$\underline{6}$	6	6	$\underline{7}$
1		$\underline{0}$	0	0	$\underline{2}$	2	2	$\underline{1}$	1	1	$\underline{3}$	3	3	$\underline{4}$	4	4
2			$\underline{3}$	3	3	$\underline{0}$	0	0	$\underline{2}$	2	2	$\underline{1}$	1	1	$\underline{5}$	5

Least Recently Used (LRU)

- Replace page with maximal forward distance: $y_t = \max_{xeS\ t-1(m)} \mathrm{BKWD}_t(x)$

Let page reference stream, ϖ = 2031203120316457

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

- Backward distance is a good predictor of forward distance -- locality

Least Frequently Used (LFU)

- Replace page with minimum use:
  $y_t = \min_{x \in S_{t-1}(m)} \mathrm{FREQ}(x)$

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	<u>2</u>	2	2													
1		<u>0</u>	0													
2			<u>3</u>													

$\mathrm{FREQ}_4(2) = 1$
$\mathrm{FREQ}_4(0) = 1$
$\mathrm{FREQ}_4(3) = 1$

Least Frequently Used (LFU)

- Replace page with minimum use:
  $y_t = \min_{x \in S\ t-1(m)} \mathrm{FREQ}(x)$

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2	2												
1		$\underline{0}$	0	$\underline{1}$												
2			$\underline{3}$	3												

$\mathrm{FREQ}_4(2) = 1$
$\mathrm{FREQ}_4(0) = 1$
$\mathrm{FREQ}_4(3) = 1$

Least Frequently Used (LFU)

- Replace page with minimum use:
  $y_t = \min_{x \in S_{t-1}(m)} \mathrm{FREQ}(x)$

Let page reference stream, ϖ = 2031203120316457

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

$\mathrm{FREQ}_6(2) = 2$
$\mathrm{FREQ}_6(1) = 1$
$\mathrm{FREQ}_6(3) = 1$

Least Frequently Used (LFU)

- Replace page with minimum use:
  $y_t = \min_{\text{xeS t-1(m)}} \text{FREQ}(x)$

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2	2	2	2										
1		$\underline{0}$	0	$\underline{1}$	1	1										
2			$\underline{3}$	3	3	$\underline{0}$										

$\text{FREQ}_7(2) = ?$
$\text{FREQ}_7(1) = ?$
$\text{FREQ}_7(0) = ?$

First In First Out (FIFO)

- Replace page that has been in memory the longest: $y_t = \max_{x \in S_{t-1}(m)} AGE(x)$

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	<u>2</u>	2	2	<u>1</u>												
1		<u>0</u>	0	0												
2			<u>3</u>	3												

First In First Out (FIFO)

- Replace page that has been in memory the longest: $y_t = \max_{x \in S_{t-1}(m)} AGE(x)$

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2													
1		$\underline{0}$	0													
2			$\underline{3}$													

$AGE_4(2) = 3$
$AGE_4(0) = 2$
$AGE_4(3) = 1$

First In First Out (FIFO)

- Replace page that has been in memory the longest: $y_t = \max_{x \in S_{t-1}(m)} AGE(x)$

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	<u>2</u>	2	2	<u>1</u>												
1		<u>0</u>	0	0												
2			<u>3</u>	3												

$AGE_4(2) = 3$
$AGE_4(0) = 2$
$AGE_4(3) = 1$

First In First Out (FIFO)

- Replace page that has been in memory the longest: $y_t = \max_{x \in S_{t-1}(m)} AGE(x)$

Let page reference stream, ϖ = 2031203120316457

Frame	2	0	3	1	2	0	3	1	2	0	3	1	6	4	5	7
0	$\underline{2}$	2	2	$\underline{1}$												
1		$\underline{0}$	0	0												
2			$\underline{3}$	3												

$AGE_5(1) = ?$

$AGE_5(0) = ?$

$AGE_5(3) = ?$

Belady's Anomaly

Let page reference stream, ϖ = 012301401234

Frame	0	1	2	3	0	1	4	0	1	2	3	4
0	$\underline{0}$	0	0	$\underline{3}$	3	3	$\underline{4}$	4	4	4	4	4
1		$\underline{1}$	1	1	$\underline{0}$	0	0	0	0	$\underline{2}$	2	2
2			$\underline{2}$	2	2	$\underline{1}$	1	1	1	1	$\underline{3}$	3

Frame	0	1	2	3	0	1	4	0	1	2	3	4
0	$\underline{0}$	0	0	0	0	0	$\underline{4}$	4	4	4	$\underline{3}$	3
1		$\underline{1}$	1	1	1	1	1	$\underline{0}$	0	0	0	$\underline{4}$
2			$\underline{2}$	2	2	2	2	2	$\underline{1}$	1	1	1
3				$\underline{3}$	3	3	3	3	3	$\underline{2}$	2	2

- FIFO with m = 3 has 9 faults
- FIFO with m = 4 has 10 faults

Stack Algorithms

- Some algorithms are well-behaved
- Inclusion Property: Pages loaded at time t with m is also loaded at time t with m+1

LRU

Frame	0	1	2	3	0	1	4	0	1	2	3	4
0	$\underline{0}$	0	0	3								
1		$\underline{1}$	1	1								
2			$\underline{2}$	2								

Frame	0	1	2	3	0	1	4	0	1	2	3	4
0	$\underline{0}$	0	0	0								
1		$\underline{1}$	1	1								
2			$\underline{2}$	2								
3				$\underline{3}$								

Stack Algorithms

- Some algorithms are well-behaved
- Inclusion Property: Pages loaded at time t with m is also loaded at time t with m+1

LRU

Frame	0	1	2	3	0	1	4	0	1	2	3	4
0	0	0	0	3	3							
1		1	1	1	1							
2			2	2	0							

Frame	0	1	2	3	0	1	4	0	1	2	3	4
0	0	0	0	0	0							
1		1	1	1	1							
2			2	2	2							
3				3	3							

Stack Algorithms

- Some algorithms are well-behaved
- Inclusion Property: Pages loaded at time t with m is also loaded at time t with m+1

LRU

Frame	0	1	2	3	0	1	4	0	1	2	3	4
0	0	0	0	3	3	3						
1		1	1	1	0	0						
2			2	2	2	1						

Frame	0	1	2	3	0	1	4	0	1	2	3	4
0	0	0	0	0	0	0						
1		1	1	1	1	1						
2			2	2	2	2						
3				3	3	3						

Stack Algorithms

- Some algorithms are well-behaved
- Inclusion Property: Pages loaded at time t with m is also loaded at time t with m+1

LRU

Frame	0	1	2	3	0	1	4	0	1	2	3	4
0	0	0	0	3	3	3	4					
1		1	1	1	0	0	0					
2			2	2	2	1	1					

Frame	0	1	2	3	0	1	4	0	1	2	3	4
0	0	0	0	0	0	0	0					
1		1	1	1	1	1	1					
2			2	2	2	2	4					
3				3	3	3	3					

Stack Algorithms

- Some algorithms are well-behaved
- Inclusion Property: Pages loaded at time t with m is also loaded at time t with m+1

LRU

Frame	0	1	2	3	0	1	4	0	1	2	3	4
0	0	0	0	3	3	3	4	4	4	2	2	2
1		1	1	1	0	0	0	0	0	0	3	3
2			2	2	2	1	1	1	1	1	1	4

Frame	0	1	2	3	0	1	4	0	1	2	3	4
0	0	0	0	0	0	0	0	0	0	0	0	4
1		1	1	1	1	1	1	1	1	1	1	1
2			2	2	2	2	4	4	4	4	3	3
3				3	3	3	3	3	3	2	2	2

Stack Algorithms

- Some algorithms are well-behaved
- Inclusion Property: Pages loaded at time t with m is also loaded at time t with m+1

FIFO

Frame	0	1	2	3	0	1	4	0	1	2	3	4
0	0	0	0	3	3	3	4	4	4	4	4	4
1		1	1	1	0	0	0	0	0	2	2	2
2			2	2	2	1	1	1	1	1	3	3

Frame	0	1	2	3	0	1	4	0	1	2	3	4
0	0	0	0	0	0	0	4	4	4	4	3	3
1		1	1	1	1	1	1	0	0	0	0	4
2			2	2	2	2	2	2	1	1	1	1
3				3	3	3	3	3	3	2	2	2

Implementation

- LRU has become preferred algorithm
- Difficult to implement
  - Must record when each page was referenced
  - Difficult to do in hardware
- Approximate LRU with a reference bit
  - Periodically reset
  - Set for a page when it is referenced
- Dirty bit

Dynamic Paging Algorithms

- The amount of physical memory -- the number of page frames -- varies as the process executes
- How much memory should be allocated?
  - Fault rate must be "tolerable"
  - Will change according to the phase of process
- Need to define a *placement* & replacement policy
- Contemporary models based on *working set*

Working Set

- Intuitively, the working set is the set of pages in the process's locality
  - Somewhat imprecise
  - Time varying
  - Given k processes in memory, let $m_i(t)$ be # of pages frames allocated to p_i at time t
    - $m_i(0) = 0$
    - $\Sigma_{i=1}^{k} m_i(t) \leq$ |primary memory|
    - Also have $S_t(m_i(t)) = S_t(m_i(t-1)) \cup X_t - Y_t$
    - Or, more simply $S(m_i(t)) = S(m_i(t-1)) \cup X_t - Y_t$

Placed/Replaced Pages

- $S(m_i(t)) = S(m_i(t-1)) \cup X_t - Y_t$
- For the missing page
  - Allocate a new page frame
  - $X_t = \{r_t\}$ in the new page frame
- How should Y_t be defined?
- Consider a parameter, τ, called the ***window size***
  - Determine $BKWD_t(y)$ for every $y \in S(m_i(t-1))$
  - if $BKWD_t(y) \geq \tau$, unload y and deallocate frame
  - if $BKWD_t(y) < \tau$ do not disturb y

Working Set Principle

- Process p_i should only be loaded and active if it can be allocated enough page frames to hold its entire working set
- The size of the working set is *estimated* using τ
  - Unfortunately, a "good" value of τ depends on the size of the locality
  - Empirically this works with a fixed τ

Example ($\tau = 3$)

Frame	0	1	2	3	0	1	2	3	0	1	2	3	4	5	6	7
0	<u>0</u>															
#	1															

Example ($\tau = 4$)

Frame	0	1	2	3	0	1	2	3	0	1	2	3	4	5	6	7
0	<u>0</u>															
#	1															

Segmentation

- Unit of memory movement is:
  - Variably sized
  - Defined by the programmer
- Two component addresses, <Seg#, offset>
- Address translation is more complex than paging
    - Ψ_t: segments x offsets $\rightarrow$ Physical Address $\cup \{\Omega\}$
    - $\Psi_t(i, j) = k$

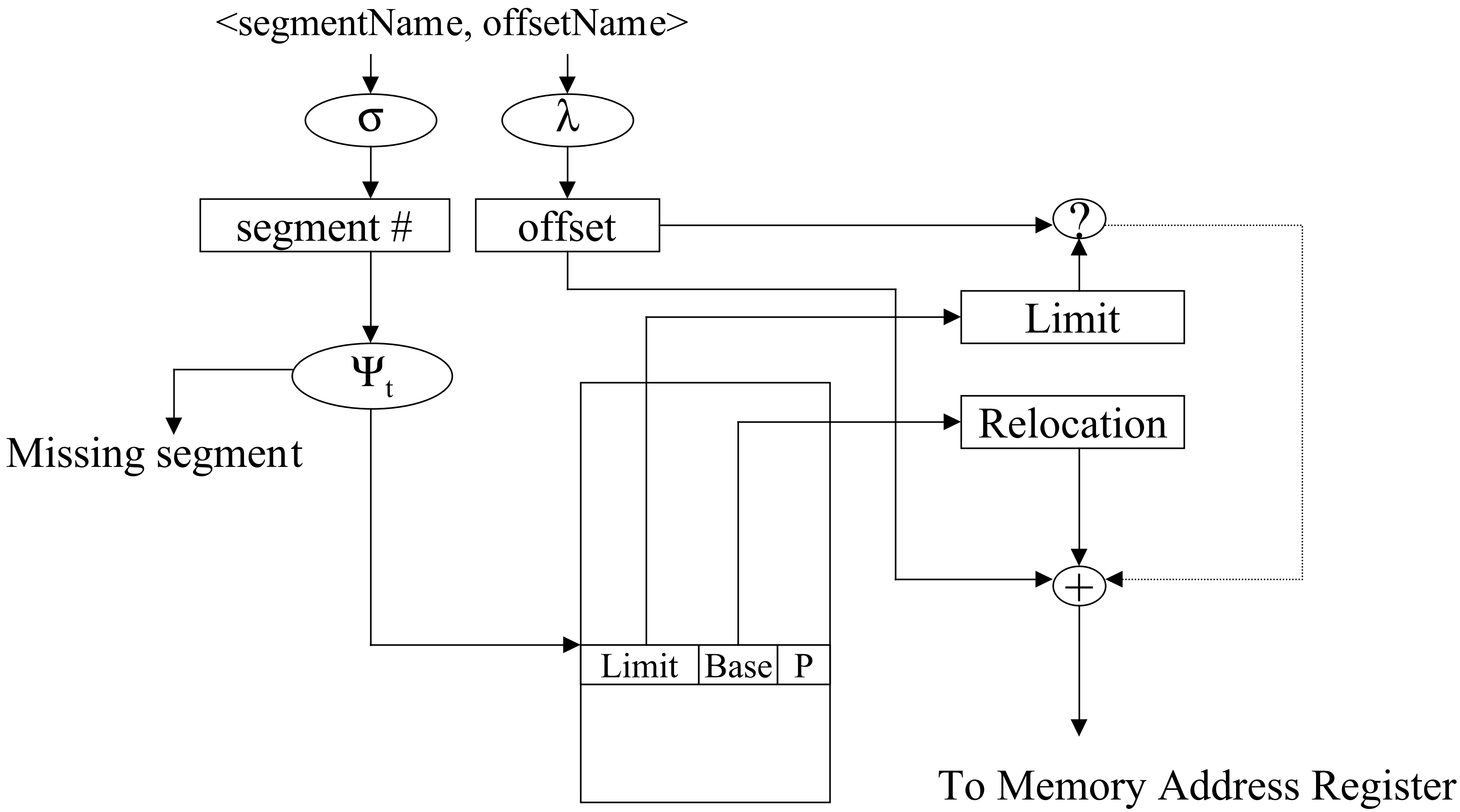
Segment Address Translation

- Ψ_t: segments x offsets $\rightarrow$ physical address $\cup \{\Omega\}$
- $\Psi_t(i, j) = k$
- σ: segments $\rightarrow$ segment addresses
- $\Psi_t(\sigma(\text{segName}), j) = k$

Segment Address Translation

- Ψ_t: segments x offsets $\rightarrow$ physical address $\cup \{\Omega\}$
- $\Psi_t(i, j) = k$
- σ: segments $\rightarrow$ segment addresses
- $\Psi_t(\sigma(segName), j) = k$
- λ: offset names $\rightarrow$ offset addresses
- $\Psi_t(\sigma(segName), \lambda(offsetName)) = k$
- Read implementation in Section 12.5.2

Address Translation



Implementation

- Segmentation requires special hardware
  - Segment descriptor support
  - Segment base registers (segment, code, stack)
  - Translation hardware
- Some of translation can be static
  - No dynamic offset name binding
  - Limited protection

Multics

- Old, but still state-of-the-art segmentation
- Uses *linkage segments* to support sharing
- Uses dynamic offset name binding
- Requires sophisticated memory management unit
- See pp 368-371