Operating system

Part VIII: Memory (Advanced)

By KONG LingBo (孔令波)

- **Mapping** 1 – map files into main memory
  - Basic: fundamental ideas and old ways
  - Advanced: so-called virtual memory
- **Mapping** 2 – map files into persistence storage medias (HDD space as instance)
  - Basic: fundamental understanding about HDD space
  - Advanced: File system
  - Other: RAID, Spooling, etc.

Goals

- Know the related concepts
  - Virtual Memory, Logical address, Physical address, Address translation
- Virtual memory
  - **(on-demand) Paging** scheme
  - **(on-demand) Segmentation**
  - Segmentation + Paging → **Hybrid**

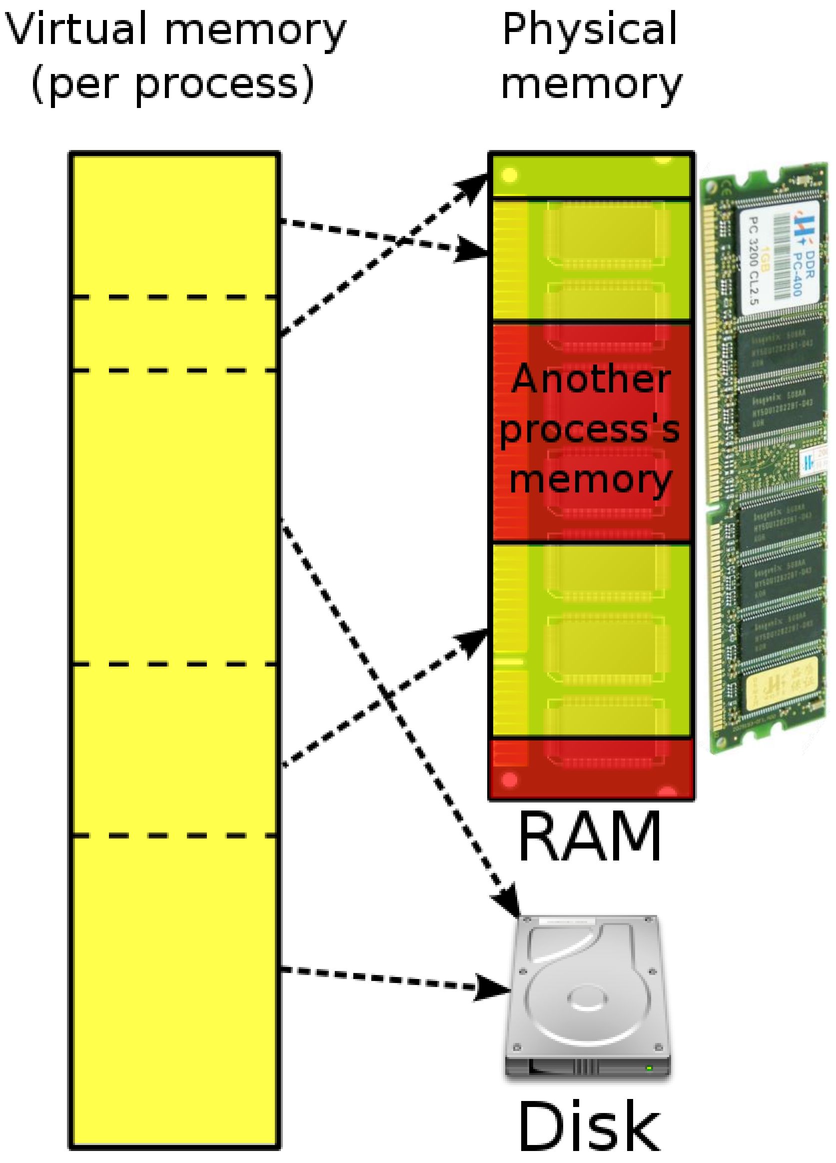
Now

- It’s time to learn the real techniques used by current OSs to provide **<u>VIRTUAL MEMORY</u>**

- We have known the fact
  - The instructions and data of a program should be stored in Main Memory first before its execution
  - With the experience on Windows®, we can infer that a program whose size is larger than the MM can still run

  ➔ it seems that there is a “**<u>virtual</u>**” memory!

http://searchstorage.techtarget.com/definition/virtual-memory



http://en.wikipedia.org/wiki/Virtual_memory

- Virtual memory is a feature of an operating system that
  - enables a process to use a memory (RAM) address space that is independent of other processes running in the same system,
  - and use a space that is **larger than the actual amount of RAM** present, temporarily relegating some contents from RAM to a disk, with little or no overhead.
- Virtual memory combines active RAM and inactive memory to form a large range of contiguous addresses.

Virtual Memory

- Paging
  - Basic paging
  - Paging-based VM
    - How to support the transparency of using space larger than the physical memory space
  - Page replacement algorithms
- Segmenting
  - Basic segmenting
  - Segmentation-based VM
    - How to support the transparency of using space larger than the physical memory space
- Segment-page scheme (Hybrid)

Paging (分页)

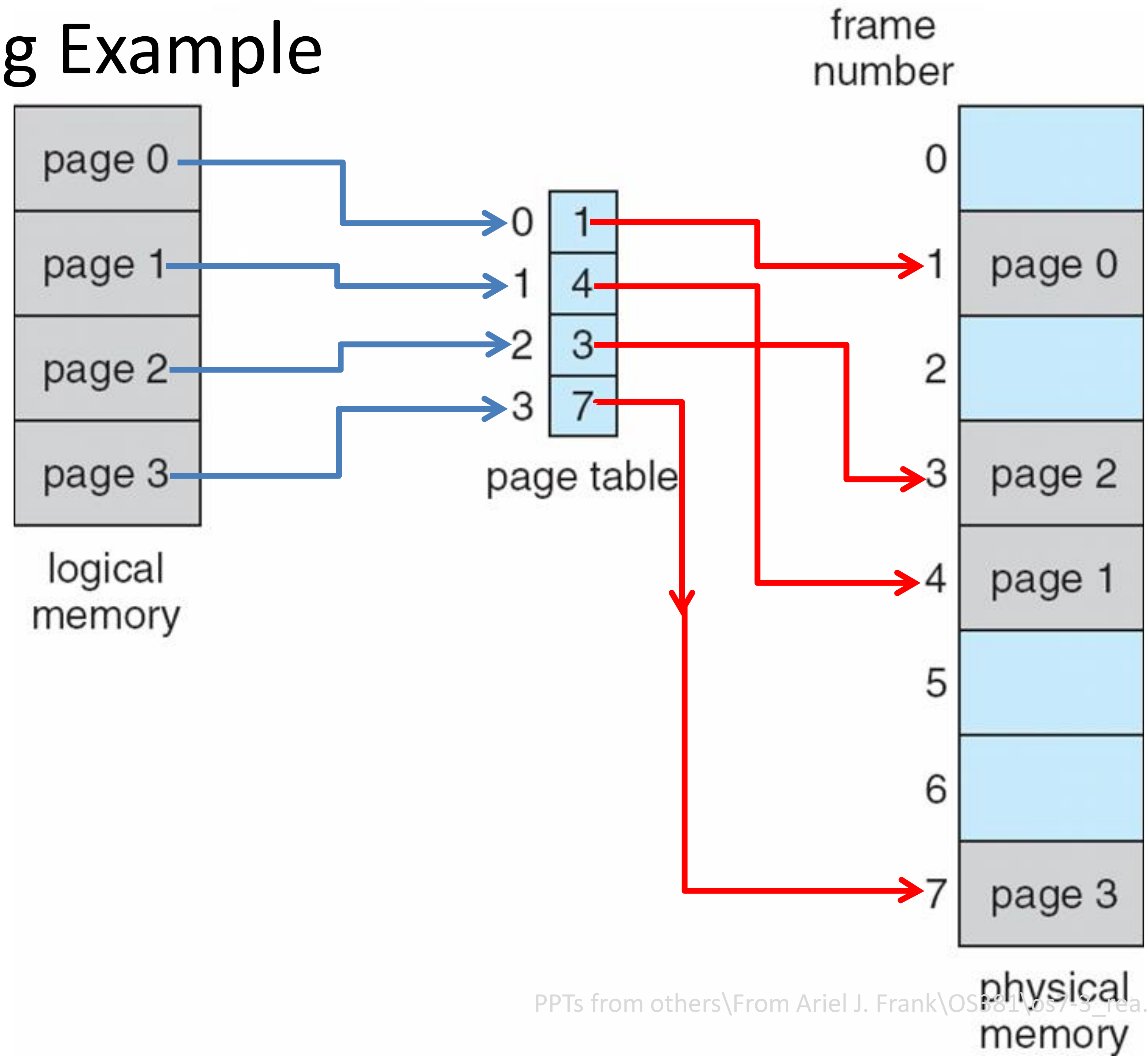
- The fundament of paging is **Fixed Partitioning** but with smaller size!
  - It "partition/cut" the logical space of a process into **pages** [页]
  - It "partition/cut" the physical space of MM into **frames** [帧]
  - By default, the sizes of page and frame are same
    - In 32 bit Windows, 4KB
- It seems that the mapping from the process space to MM is easy now
  - If there is available frame, we can assign a page to that frame
  - Since a page corresponds to a set of instructions, that process could run when executing those instructions
- Yes, that is in fact the basic functions of paging scheme!

Needed data structures

- To carry out the paging scheme
  - Besides the pages of a process, OS should know the mapping relationship between pages of that process and frames of the MM
- That is Page table [页表]

Page	Frame

Paging Example



Virtual Memory: Large as you wish!

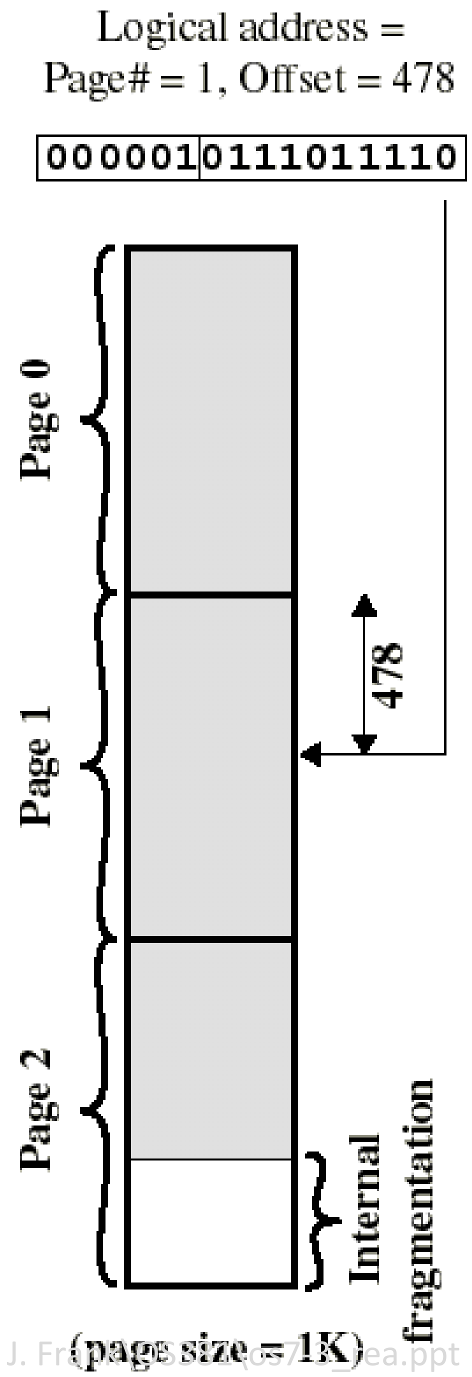
- Example:
  - Just 16 bits are needed to address a physical memory of 64KB.
  - Let's use a page size of 1KB so that 10 bits are needed for offsets within a page.
  - For the page number part of a logical address we may use a number of bits larger than 6, say 22 (a modest value!!), "**pretending**" a 32-bit address.
    - Now we have 2^{22} (=4M) pages, each of which is 1KB, so the VM of a process/OS seems 4GB (← $2^{22}*2^{10}=2^{32}$=4GB)

PPTs from others\From Ariel J. Frank\OS381\os8-1_vir.ppt

Logical address now

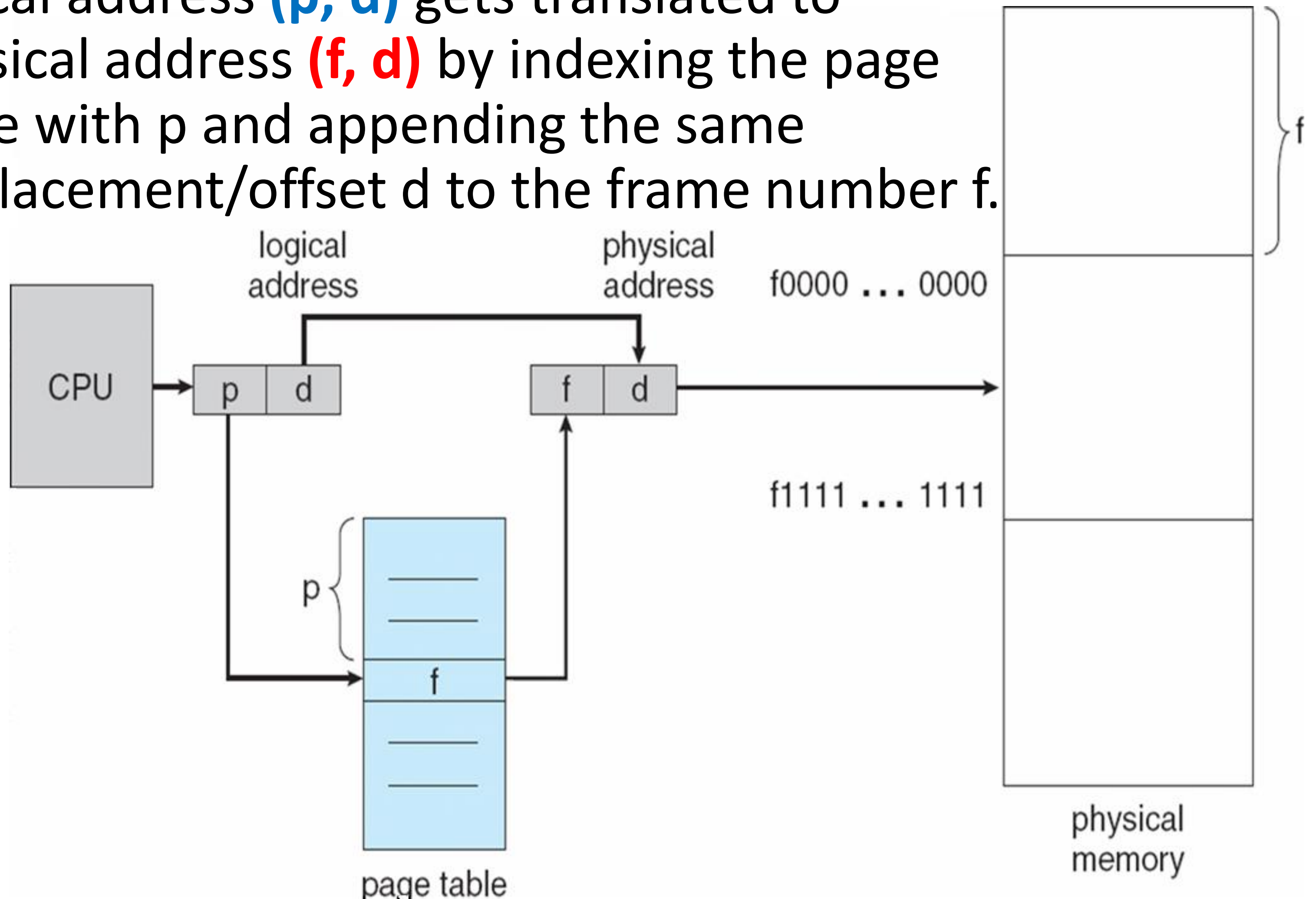
- Logical address now is divided into two parts:
  - *Page number* (***p***) – used as an index into a *page table* which contains the base address of each page in physical memory.
  - *Page offset/displacement* (***d***) – combined with base address to define the physical memory address that is sent to the memory unit.

page number	page offset
p	***d***
m – n	***n***



Address Translation now

- Address translation at run-time is then easy to implement in hardware:
  - logical address **(p, d)** gets translated to physical address **(f, d)** by indexing the page table with p and appending the same displacement/offset d to the frame number f.



In a paging memory management system, there is a page table as following:

Page No	0	1	2	3	4
Frame	2	1	6	3	7

If the page size is 4KB, then paging address hardware will convert logical address 0 into physical address（　　）.

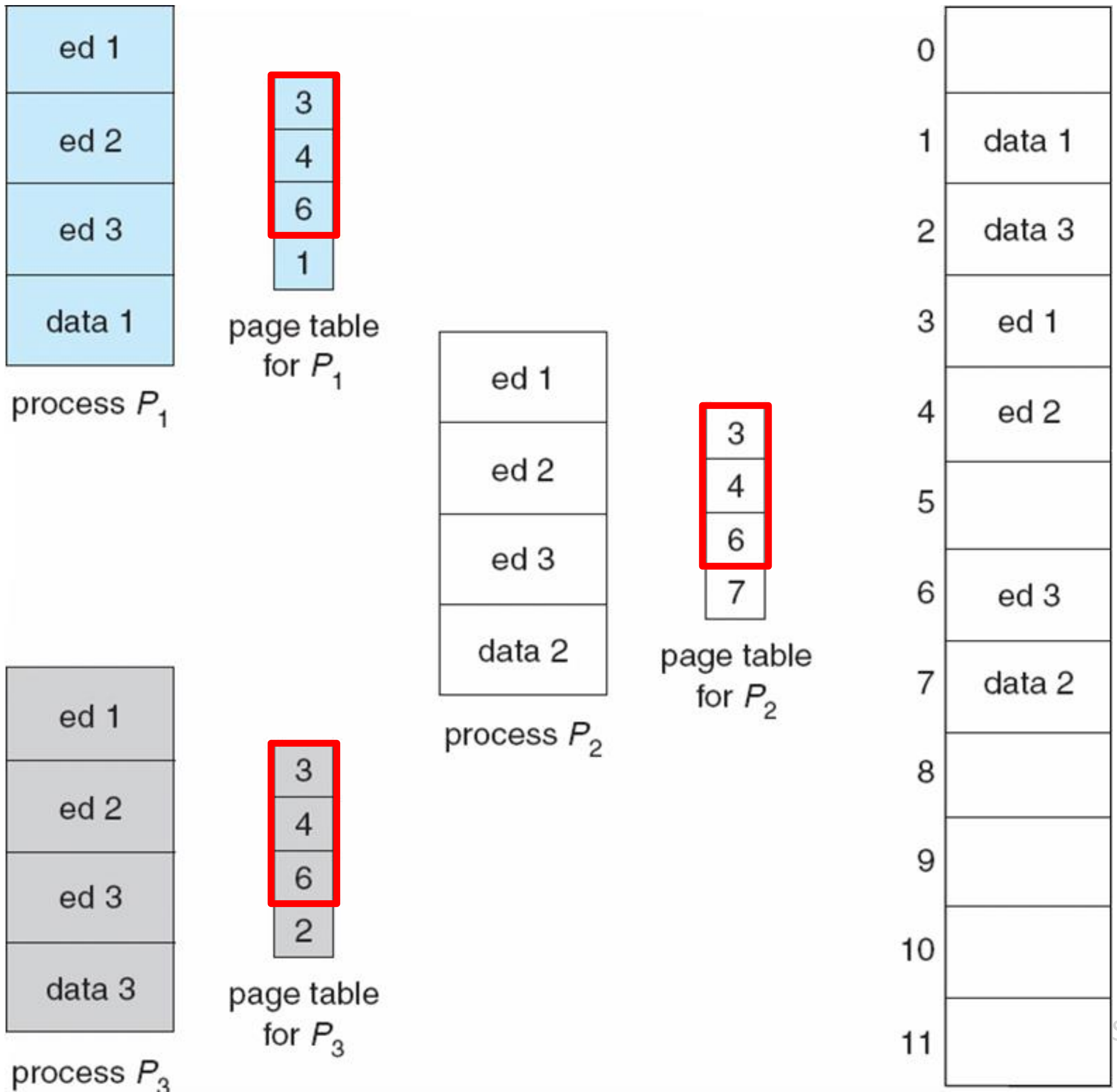
A.8192　　B.4096　　C.2048　　D.1024

Sharing

- What can be shared
  - Reentrant code (pure code)
    - If the code is reentrant, then it never changes during execution. ➔Two or more processes can execute the same code at the same time.
  - Read-only data
- What can not be shared
  - Each process has its own copy of registers and data storage to hold the data for the process's execution.
- The OS should provides facility to enforce some necessary property for sharing.

PPTs from others\OS PPT in English\ch09.ppt

Shared Pages Example



Paging： The OS Concern

- What should the OS do?
  - –Which frames are **available**?
  - –Which frames are **allo**
  - –How to allocate fram arrived process?
    - **Placement** algorithm (放置算法)
    - **Replacement** algorithms (替换算法)

Similar as those discussed in Variable partitioning part!

PPTs from others\OS PPT in English\ch09.ppt

Virtual Memory

- Paging
  - Basic paging
  - Paging-based VM
    - How to support the transparency of using space larger than the physical memory space
  - Page replacement algorithms
- Segmenting
  - Basic segmenting
  - Segmentation-based VM
    - How to support the transparency of using space larger than the physical memory space
- Segment-page scheme (Hybrid)

Of course, we need some support for Virtual Memory

- Memory management hardware must support paging and/or segmentation [MMU🡐 Discussed in former part].

- OS must be able to manage [🡒 **Concern of this part**]
  - the movement of pages and/or segments between external memory and main memory,
  - including placement and replacement of pages/segments.

PPTs from others\From Ariel J. Frank\OS381\os8-1_vir.ppt

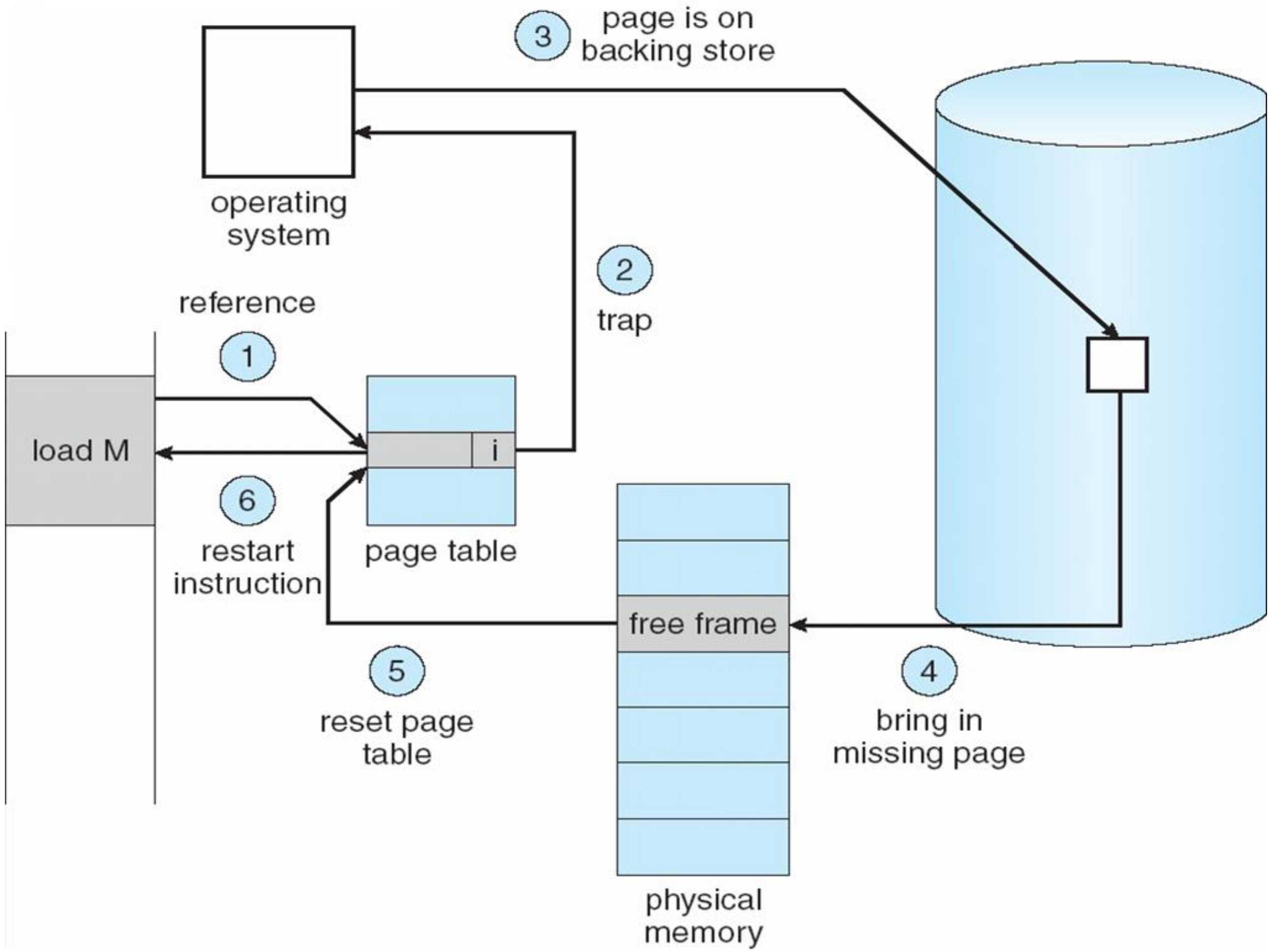
Now, the execution of a process looks like …

1. The OS brings into main memory only a few pieces of the program (including its starting point).
2. An interrupt (memory fault) is generated when the memory reference is on a piece that is not present in main memory – **<u>is needed (On-demand)</u>**.
   a. OS places the process in a Bl
   b. OS issues a disk I/O Read
   memory the piece referenc
   c. Another process is dispatched to run while the disk I/O takes place.
3. An interrupt is issued when disk I/O completes; this causes the OS to place the affected process back in the Ready state.

Called Demand Paging!

PPTs from others\From Ariel J. Frank\OS381\os8-1_vir.ppt

If one page is not in MM yet (**Page Fault: 缺页**)



1. If there is ever a reference to a page not in memory, first reference will cause **page fault**.
2. Page fault is handled by the appropriate OS service routines.
3. Locate needed page on disk (in file or in backing store).
4. Swap page into free frame (assume available).
5. Reset page tables – valid-invalid bit = 1.
6. Restart instruction.

Steps in handling a Page Fault

What happens if there is no free frame?

- Page replacement – find some page in memory, but not really in use, **swap** it out.
- Need **page replacement algorithm**.
- Performance – want an algorithm which will result in minimum number of page faults.
- Same page may be brought into memory several times.

PPTs from others\From Ariel J. Frank\OS381\os8-1_vir.ppt

Effective Access Time (EAT)

- EAT = (1 – **p**) x memory access
  + **p** (page fault overhead
  + swap page out
  + swap page in
  + restart overhead)

- "p" means Page Fault Rate
  – $0 \le p \le 1.0$
  – if p = 0, no page faults
  – if p = 1, every reference is a fault

- Demand Paging Example
  - Memory access time = **200 nanoseconds [纳秒]**
  - Average page-fault service time = 8 milliseconds [毫秒]
  - EAT = (1 – p) x 200 + p (8 milliseconds)
    = (1 – p) x 200 + p x 8,000,000
    = 200 + p x 7,999,800
  - If one access out of 1,000 causes a page fault, then
    EAT = 200+0.001*7999800
    = 8199.8 nanoseconds
    = **8.2 microseconds [微秒]**.

1 millisec = 1,000 microsec = 1,000,000 nanosec

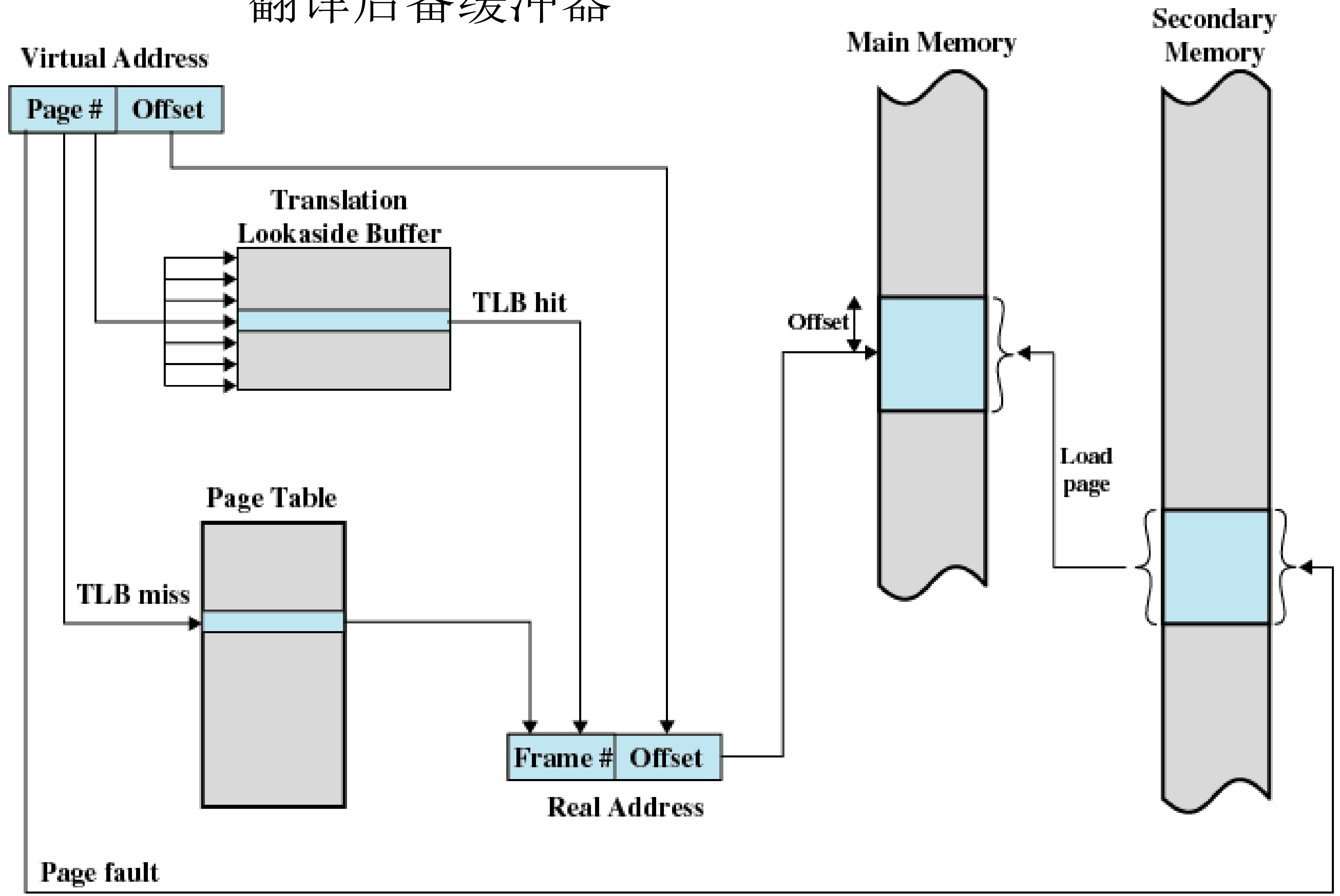
How to improve this kind of **slowdown**?

- The solution could be derived from the EAT equation - Improve the **access speed**, and decrease the **page fault**
- Two strategies
  - Keeping page table in a higher access speed media
    - Cache (high speed but quite expensive), and the **TLB** (**translation lookaside buffer**)
  - Prefetching the possible future accessed pages
    - When page 3 causes a page fau[illegible] to MM

Of course, decision is needed – namely the algorithms

a Translation Look-aside Buffer (TLB)

翻译后备缓冲器



PPTs from others\From Ariel J. Frank\OS381\os8-1_vir.ppt
PPTs from others\OS5e after William Stallings\Chapter08.ppt

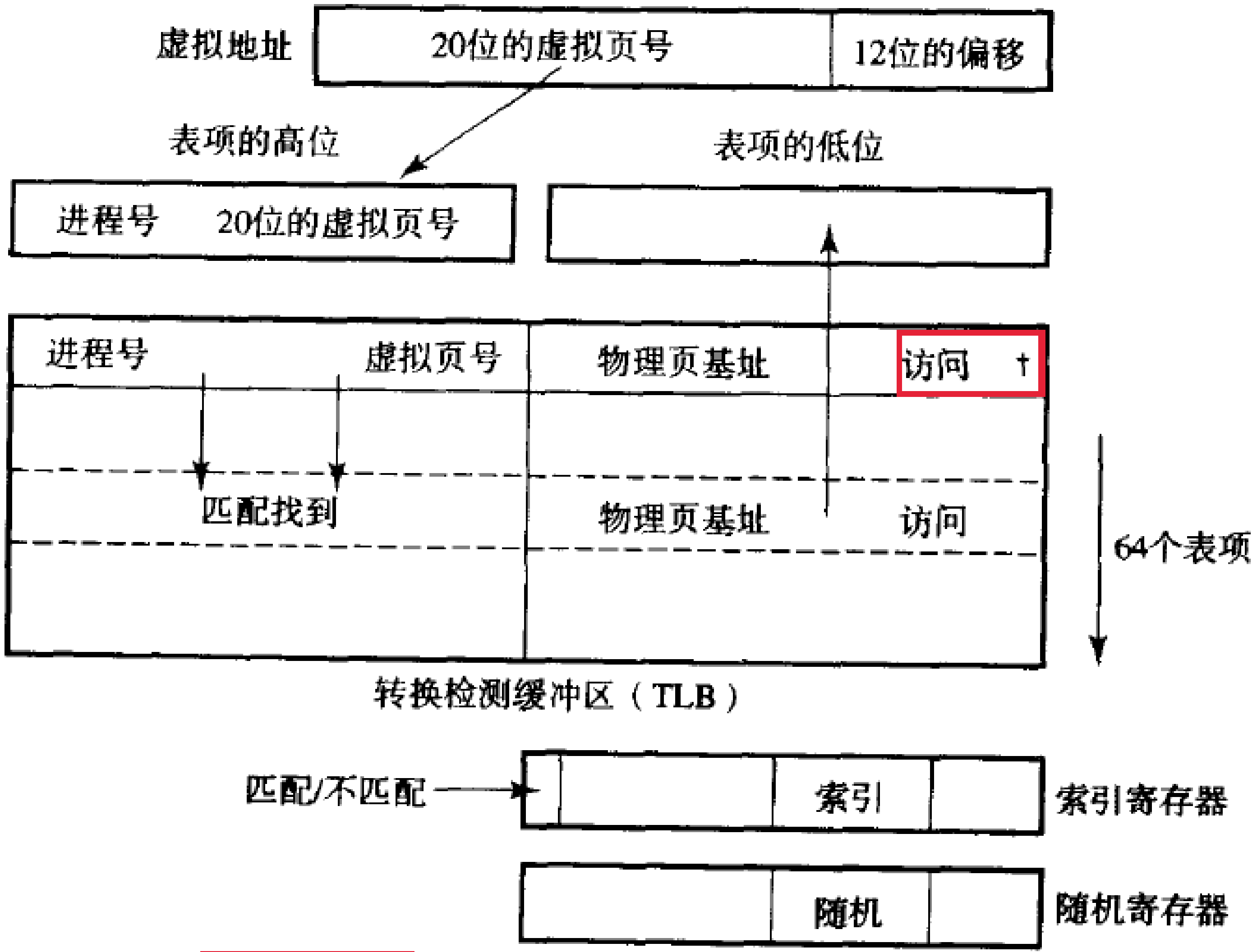
Figure 8.7 Use of a Translation Lookaside Buffer

Now the **EAT** with TLB

- Parameters
  - TLB Lookup = **20 nanoseconds**
  - Hit ratio (percentage of times that a particular page is found in the TLB) = 80%
  - Memory access time = **200 nanoseconds**
  - Average page-fault service time = 8 milliseconds
  - If one access for Page table out of 1,000 causes a page fault
- EAT = (20)*0.8 + [200+(8000000-200)*0.001]*0.2

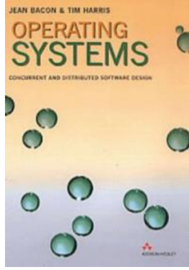
  = 16+ 8199.8*0.2

  = 1655.96≈1.6 **microseconds**

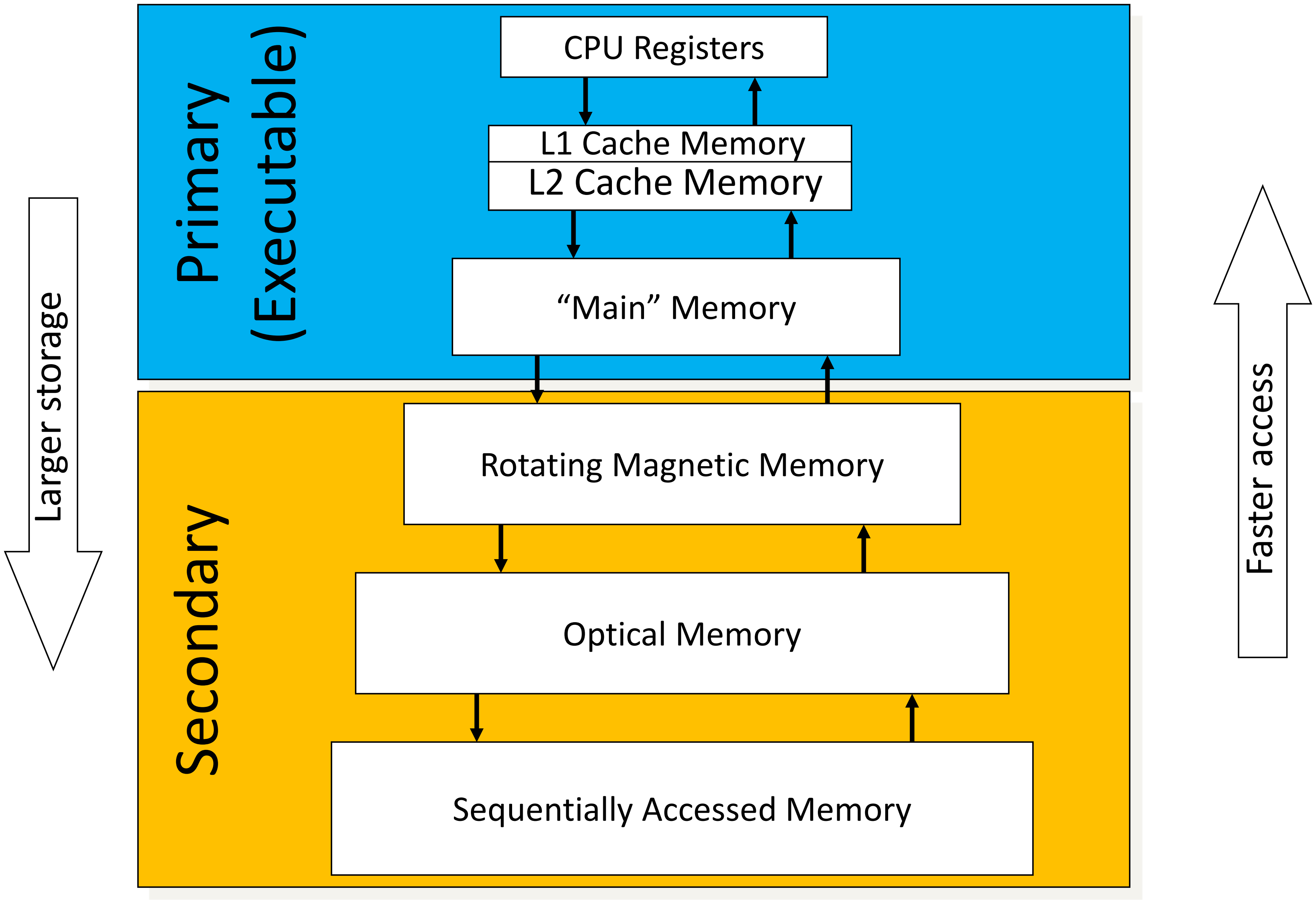


†访问控制位　G——全局（被所有进程共享）
D——脏位（写保护）
V——有效位（供操作系统使用）
N——被缓存还是没有被缓存
1——操作系统中的地址

图 5.15　MIPS R2000/3000 地址转换



Contemporary Memory Hierarchy & Dynamic Loading



PPTs from others\cms.dt.uh.edu\chap11.ppt

Virtual Memory

- Paging
  - Basic paging
  - Paging-based VM
    - How to support the transparency of using space larger than the physical memory space
  - Page replacement algorithms
- Segmenting
  - Basic segmenting
  - Segmentation-based VM
    - How to support the transparency of using space larger than the physical memory space
- Segment-page scheme (Hybrid)

Example

- A process makes references to 4 **pages**: A, B, E, and R
  - Reference stream: BEERBAREBEAR
- Physical memory size: 3 (page) **frames**

Memory frame\Pages	B	E	E	R	B	A	R	E	B	E	A	R
1												
2												
3												

The FIFO Policy

- Treats page frames allocated to a process as a circular buffer:
  - When the buffer is full, the oldest page is replaced. Hence first-in, first-out:
    - A frequently used page is often the oldest, so it will be repeatedly paged out by FIFO.
  - Simple to implement:
    - requires only a pointer that circles through the page frames of the process.

PPTs from others\From Ariel J. Frank\OS381\os8-3_vir.ppt

FIFO

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2												
3												

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

FIFO

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2		E										
3												

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

FIFO

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2		E	*									
3												

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

FIFO

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2		E	*									
3				R								

FIFO

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*							
2		E	*									
3				R								

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

FIFO

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*							
2		E	*									
3				R								

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

FIFO

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*									
3				R								

FIFO

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*									
3				R			*					

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

FIFO

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*					*				
3				R			*					

FIFO

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*					*				
3				R			*					

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

FIFO

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*					*	B			
3				R			*					

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

FIFO

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*					*	B			
3				R			*					

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

FIFO

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*					*	B			
3				R			*			E		

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

FIFO

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A					*	
2		E	*					*	B			
3				R			*			E		

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

FIFO

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A					*	
2		E	*					*	B			
3				R			*			E		

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

FIFO

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A					*	R
2		E	*					*	B			
3				R			*			E		

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

FIFO

- 7 page faults

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A					*	R
2		E	*					*	B			
3				R			*			E		

FIFO

- 4 compulsory cache misses

Compulsory D.J.[kəmˈpʌlsəri]
adj.必须做的, 强制性的

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	***A***					*	***R***
2		E	*					*	***B***			
3				R			*			***E***		

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

Optimal Page Replacement

- The Optimal policy selects for replacement the page that will not be used for longest period of time.

- **Impossible to implement** (need to know the future) but serves as a standard to compare with the other algorithms we shall study.

PPTs from others\From Ariel J. Frank\OS381\os8-3_vir.ppt

Optimal (MIN)

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2		E	*									
3				R								

Optimal (MIN)

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*							
2		E	*									
3				R								

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

Optimal (MIN)

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*							
2		E	*									
3				R								

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

MIN

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*									
3				R								

MIN

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*									
3				R			*					

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

MIN

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*					*				
3				R			*					

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

MIN

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*					*				
3				R			*					

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

MIN

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*					*				
3				R			*		B			

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

MIN

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A						
2		E	*					*		*		
3				R			*		B			

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

MIN

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A					*	
2		E	*					*		*		
3				R			*		B			

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

MIN

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A					*	R
2		E	*					*		*		
3				R			*		B			

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

MIN

- 6 page faults

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*	A					*	R
2		E	*					*		*		
3				R			*		B			

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

The LRU Policy
[least recently used:最近最少使用算法]

- Replaces the page that has not been referenced for the longest time recently:
  - By the principle of locality, this should be the page least likely to be referenced in the near future.
  - performs nearly as well as the optimal policy.

LRU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2		E	*									
3				R								

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

LRU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*							
2		E	*									
3				R								

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

LRU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*							
2		E	*									
3				R								

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

LRU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*							
2		E	*			A						
3				R								

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

LRU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*							
2		E	*			A						
3				R			*					

LRU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*							
2		E	*			A						
3				R			*					

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

LRU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*			E				
2		E	*			A						
3				R			*					

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

LRU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*			E				
2		E	*			A						
3				R			*					

LRU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*			E				
2		E	*			A			B			
3				R			*					

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

LRU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*			E		*		
2		E	*			A			B			
3				R			*					

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

LRU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*			E		*		
2		E	*			A			B			
3				R			*					

LRU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*			E		*		
2		E	*			A			B			
3				R			*				A	

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

LRU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*			E		*		
2		E	*			A			B			
3				R			*				A	

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

LRU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*			E		*		
2		E	*			A			B			R
3				R			*				A	

LRU

- 8 page faults

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				*			E		*		
2		E	*			A			B			R
3				R			*				A	

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

The Clock (Second Chance) Policy

- Replaces an old page, but not the oldest page
- Arranges physical pages in a circle
  - With a clock hand
- Each page has a ***used bit***
  - Set to 1 on reference
  - On page fault, sweep the clock hand
    - If the used bit == 1, set it to 0 and **advance the hand**
    - If the used bit == 0, pick the page for replacement

PPTs from others\From Ariel J. Frank\OS381\os8-3_vir.ppt

The Clock (Second Chance) Policy

- The set of frames candidate for replacement is considered as a circular buffer.
- When a page is replaced, a pointer is set to point to the next frame in buffer.
- A reference bit for each frame is set to 1 whenever:
  - a page is first loaded into the frame.
  - the corresponding page is referenced.
- When it is time to replace a page, the first frame encountered with the reference bit set to 0 is replaced:
  - During the search for replacement, each reference bit set to 1 is changed to 0.

PPTs from others\From Ariel J. Frank\OS381\os8-3_vir.ppt

CLOCK

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1												
2												
3												

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

CLOCK

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*											
2												
3												

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

CLOCK

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*											
2		E*										
3												

CLOCK

E is accessed again, so still E*

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*											
2		E*	*									
3												

CLOCK

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*											
2		E*	*									
3				R*								

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

CLOCK

B is accessed again, so still B*

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*				*							
2		E*	*									
3				R*								

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

CLOCK

Since there is an "*", clear "*" and advance the hand
Here we need a replacement

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*				*							
2		E*	*									
3				R*								

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

CLOCK

Since there is an "*", clear "*" and advance the hand

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*				*	**B**						
2		E*	*			**E**						
3				R*								

CLOCK

Since there is an “*”, clear “*” and advance the hand

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*				*	B						
2		E*	*			E						
3				R*		R						

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

CLOCK

Now, we can assign A to this position

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*				*	A*						
2		E*	*			E						
3				R*		R						

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

CLOCK

R is accessed again, so R* again

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*				*	A*						
2		E*	*			E						
3				R*		R	*					

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

CLOCK

E is accessed again, so E* again

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*				*	A*						
2		E*	*			E		*				
3				R*		R	*					

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

CLOCK

A page fault again, clear “*” and advance the hand

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*				*	A*						
2		E*	*			**E**		*	**E**			
3				R*		**R**	*					

CLOCK

Clear “*” and advance the hand

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*				*	A*						
2		E*	*			**E**		*	**E**			
3				R*		**R**	*		**R**			

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

CLOCK

Clear “*” and advance the hand

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*				*	A*			**A**			
2		E*	*			E		*	**E**			
3				R*		R	*		**R**			

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

CLOCK

Now put B here, because there is no “*” here

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*				*	A*			**A**			
2		E*	*			**E**		*	B*			
3				R*		**R**	*		**R**			

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

CLOCK

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*				*	A*			**A**			
2		E*	*			**E**		*	B*			
3				R*		**R**	*		**R**	E*		

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

CLOCK

A is accessed again, so **A* again**

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*				*	A*			**A**		*	
2		E*	*			**E**		*	B*			
3				R*		**R**	*		**R**	E*		

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

CLOCK

A page fault again, clear "*"
and advance the hand

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*				*	A*			**A**		*	**A**
2		E*	*			**E**		*	B*			
3				R*		**R**	*		**R**	E*		

CLOCK

Clear "*" and advance the hand

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*				*	A*			**A**		*	**A**
2		E*	*			**E**		*	B*			**B**
3				R*		**R**	*		**R**	E*		

CLOCK

Clear "*" and
advance the hand

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*				*	A*			**A**		*	**A**
2		E*	*			**E**		*	B*			**B**
3				R*		**R**	*		**R**	E*		**E**

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

CLOCK

7 page faults

Now put R here !

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B*				*	A*			**A**		*	R*
2		E*	*			**E**		*	B*			**B**
3				R*		**R**	*		**R**	E*		**E**

Counting-based Algorithms

- Keep a counter of the number of references that have been made to each page.
- Two possibilities: Least/Most Frequently Used (**LFU**/**MFU**).
- LFU Algorithm:
  – replaces page with smallest count; others were and will be used more.
- MFU Algorithm:
  – based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

PPTs from others\From Ariel J. Frank\OS381\os8-3_vir.ppt

LFU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2												
3												

LFU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2		E										
3												

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

LFU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2		E	2									
3												

LFU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B											
2		E	2									
3				R								

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

LFU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				2							
2		E	2									
3				R								

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

LFU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				2							
2		E	2									
3				R		A						

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

LFU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				2							
2		E	2									
3				R		A	R					

LFU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				2							
2		E	2					3				
3				R		A	R					

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

LFU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				2				3			
2		E	2					3				
3				R		A	R					

LFU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				2				3			
2		E	2					3		4		
3				R		A	R					

LFU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				2				3			
2		E	2					3		4		
3				R		A	R				A	

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

LFU

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				2				3			
2		E	2					3		4		
3				R		A	R				A	R

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

LFU

- 7 page faults

Memory frame	B	E	E	R	B	A	R	E	B	E	A	R
1	B				2				3			
2		E	2					3		4		
3				R		A	R				A	R

PPTs.2012\PPTs from others\www.cs.fsu.edu_~awang_courses_cop4610_f2005\lecture_14_virtual_memory.ppt

Does adding RAM always reduce misses?

- Yes for LRU and MIN
  - Memory content of X pages $\subseteq$ X + 1 pages

- **No for FIFO**
  - Due to modulo math
  - ***Belady's anomaly***: getting more page faults by increasing the memory size

Belady’s Anomaly

- 9 page faults

Memory frame	A	B	C	D	A	B	E	A	B	C	D	E
1	A			D			E					*
2		B			A			*		C		
3			C			B			*		D	

Belady's Anomaly

- 10 page faults

Memory frame	A	B	C	D	A	B	E	A	B	C	D	E
1	A				*		E				D	
2		B				*		A				E
3			C						B			
4				D						C		

Possibility of **Thrashing**

- If a process does not have “enough” pages, the page-fault rate is very high. This leads to:
  - low CPU utilization.
  - operating system thinks that it needs to increase the degree of multiprogramming.
  - another process added to the system.
  - This just increases the load on physical memory.

- **Thrashing** = a process is busy swapping pages in and out.

PPTs from others\From Ariel J. Frank\OS381\os8-2_vir.ppt

You can try those algorithms by yourself

- Assume:
  - 3 frames
  - Instruction References: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1
  - ~~Each of the numbers refers to a page number~~
  - Page size = 1? 2? 4?
- Your task now
  - FIFO
  - MIN (Optimal)
  - LRU
  - Clock

Virtual Memory

- Paging
  - Basic paging
  - Paging-based VM
    - How to support the transparency of using space larger than the physical memory space
  - Page replacement algorithms
- Segmenting
  - Basic segmenting
  - Segmentation-based VM
    - How to support the transparency of using space larger than the physical memory space
- Segment-page scheme (Hybrid)

Motivation of Segmenting

- Paging
  - Mapping to allow differentiation between logical memory and physical memory.
  - Separation of the user's view of memory and the actual physical memory.
  - Chopping a process into equally-sized pieces.

  → Paging division is arbitrary; no natural/logical boundaries for protection/sharing.

→ Any scheme for dividing a process into a collection of semantic units?

(syntactic [语法的], semantic [语义的])

PPTs from others\OS PPT in English\ch09.ppt

Segmentation(分段)

- Segmentation could be seen as the extension of variable partitioning
  - Each program is subdivided into blocks of non-equal size called **segments**.
    - Cut your program according to **semantic** organization, such as following function, or class etc.
  - Allocate MM region **whose size is just the size of the needed segment**
    - When a process gets loaded into main memory, its different segments can be located anywhere.

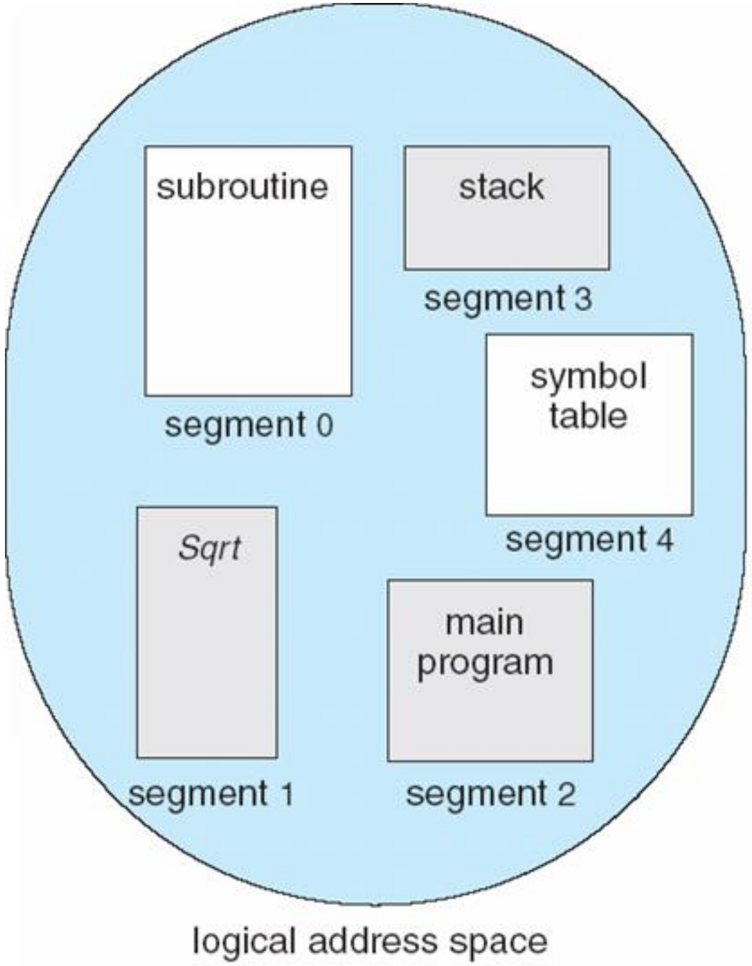
PPTs from others\SCU_Zhaohui\OS\Chapter07.ppt

Dynamics of Simple Segmentation

- There is external fragmentation; it is reduced when using small segments.
  - Each segment is fully packed with instructions/data; **no internal fragmentation**.
- In contrast with paging, segmentation is visible to the programmer:
  - provided as a convenience to organize logically programs (**example: data in one segment, code in another segment**).
  - must be aware of segment size limit.
- The OS maintains a segment table for each process. Each entry contains:
  - the starting physical addresses of that segment.
  - the length of that segment (for protection).

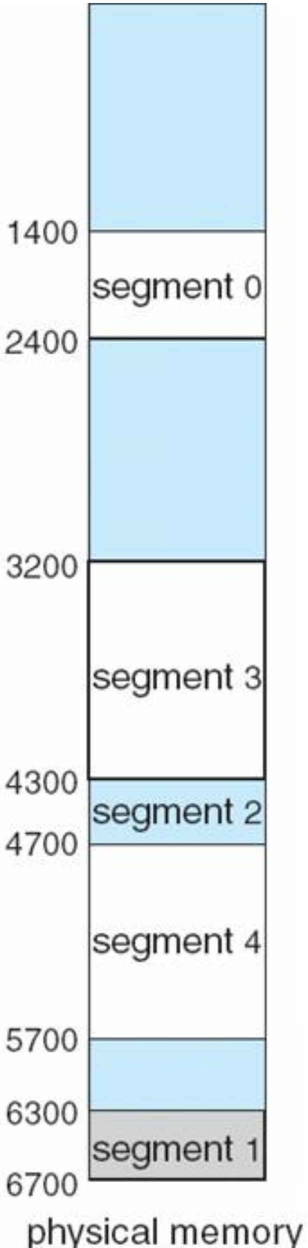
PPTs from others\From Ariel J. Frank\OS381\os7-4_rea.ppt

Example of Segmentation



	limit	base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

segment table

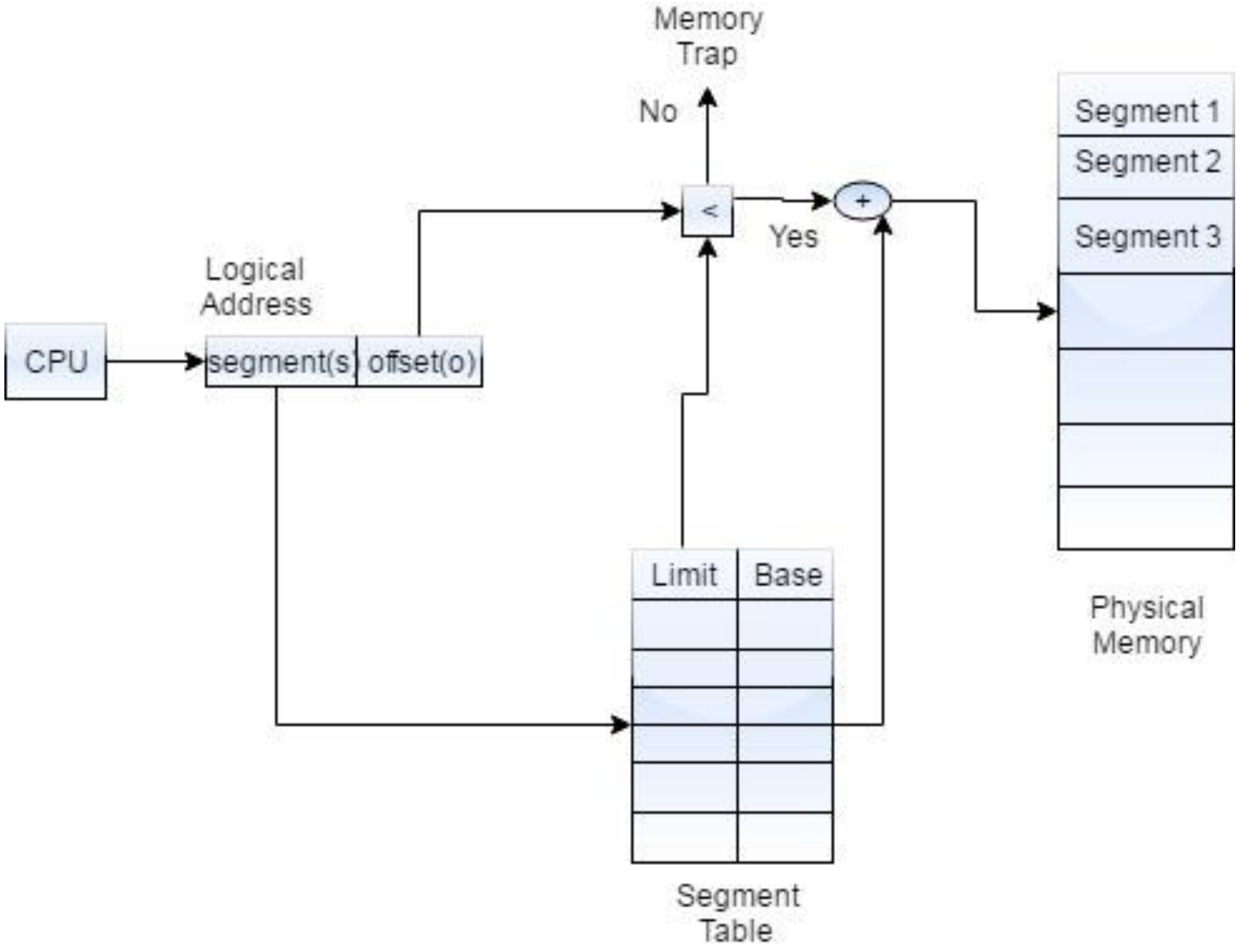


Addressing consist of two parts - a segment number(段号) and an offset(偏移量)

Logical address used in segmentation

- Logical address now is divided into two parts:
  – (segment number, offset) = (s, d), the CPU indexes (with s) the **segment table** to obtain the starting physical address **b** and the length l of that segment.
- The physical address is obtained by adding d to b (in contrast with paging):
  – The hardware also compares the offset d with the length l of that segment to determine if the address is valid.

- Virtual address translation scheme with segmentation

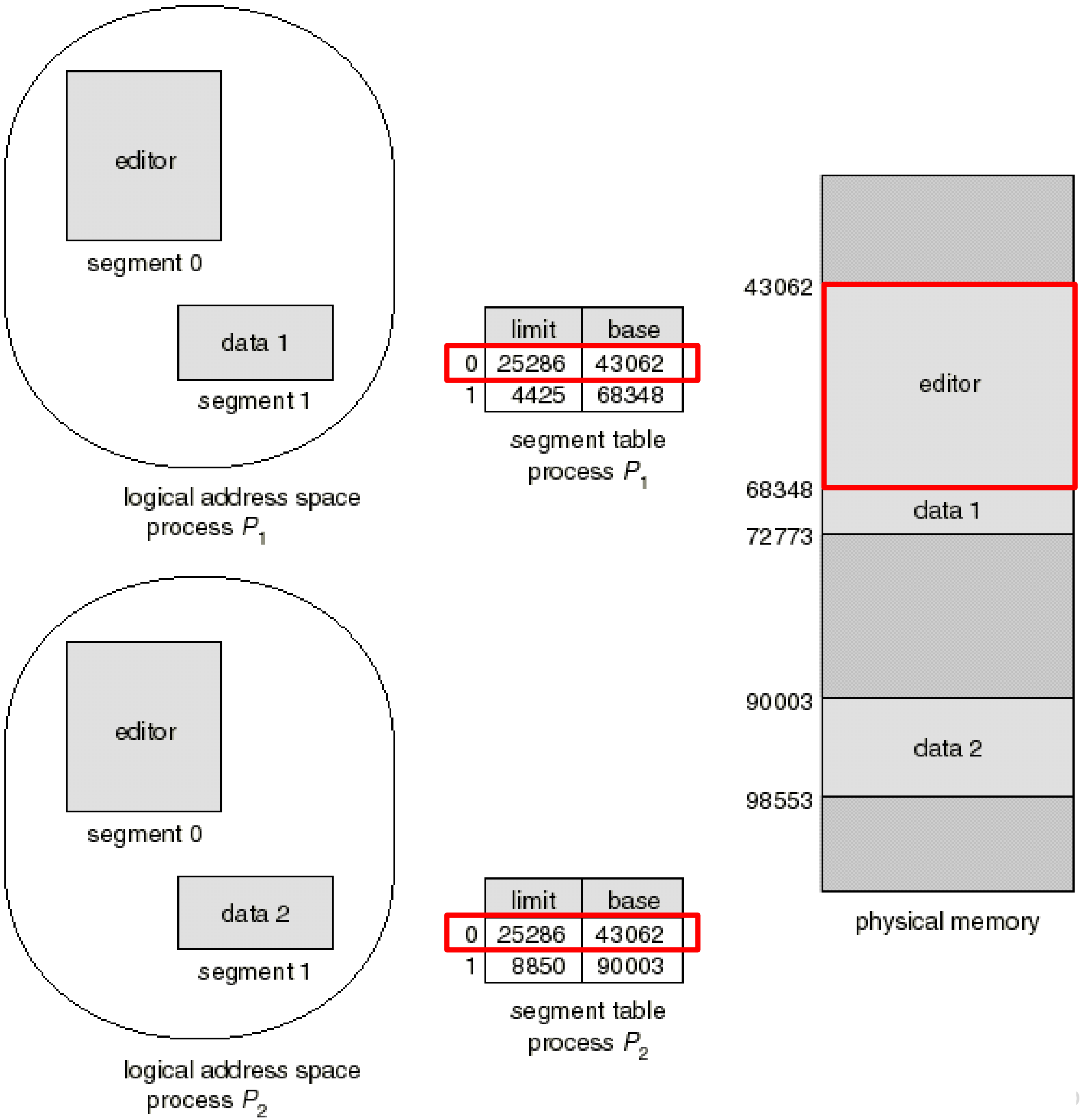


Sharing in Segmentation Systems

- Segments are shared when entries in the segment tables of 2 different processes point to the same physical locations.
- Example: the same code of a text editor can be shared by many users:
  - Only one copy is kept in main memory.
- But each user would still need to have its own private data segment.

PPTs from others\From Ariel J. Frank\OS381\os7-4_rea.ppt

Shared Segments Example



Virtual Memory

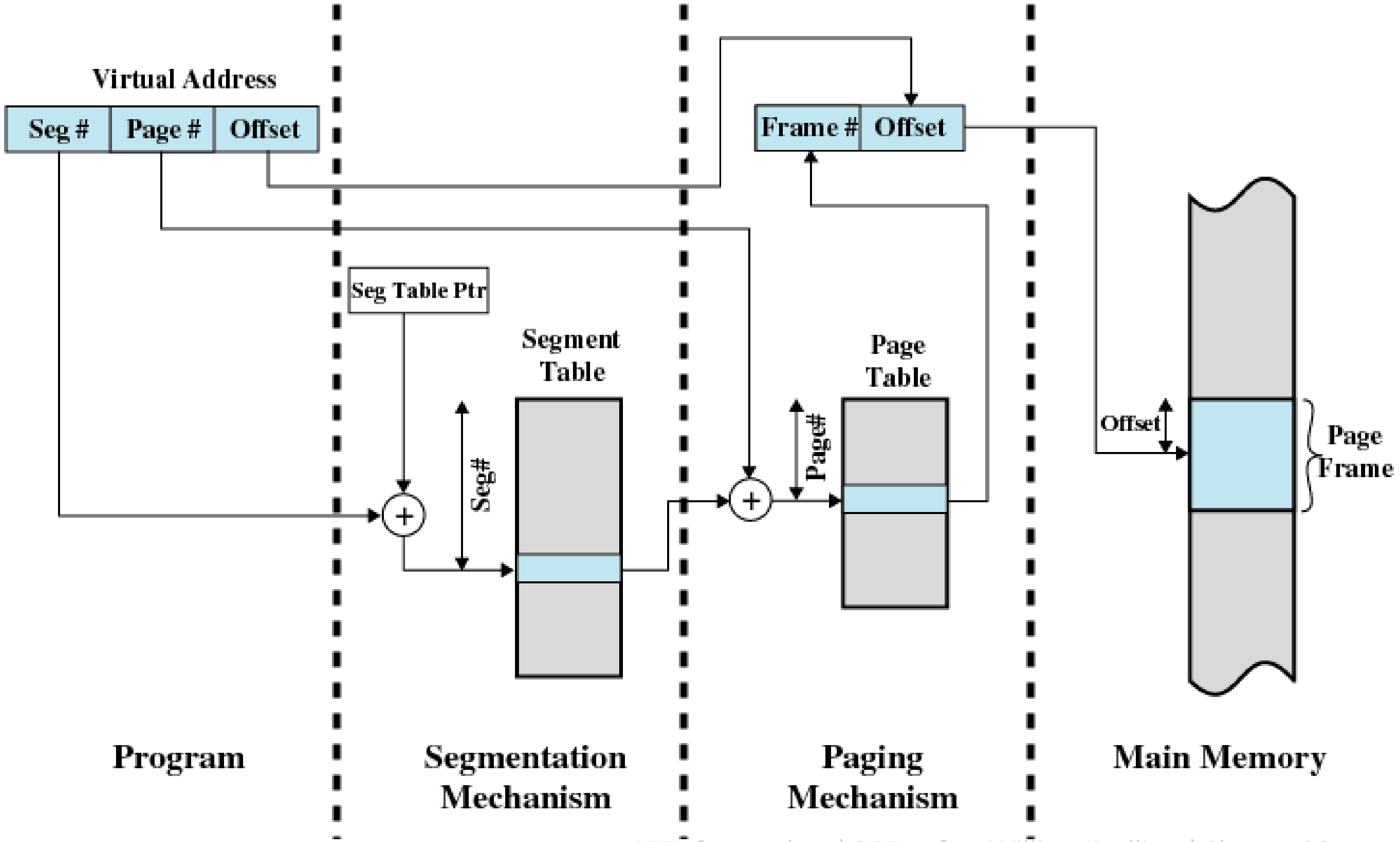
- Paging
  - Basic paging
  - Supporting VM
- Segmenting
  - Basic segmenting
  - Supporting VM
- Segment-page scheme (Hybrid)

Segmentation + Paging (Hybrid)

- Paging or segmentation?
- In the old days,
  - Motorola 68000 paging.
  - Intel 80x86 segmentation.
- Now
  - **combines both paging and segmentation**
- The OS for I386
  - OS/2 from IBM
  - NT from MS

PPTs from others\OS PPT in English\ch09.ppt

Address Translation in hybrid method



PPTs from others\OS5e after William Stallings\Chapter08.ppt
PPTs from others\From Ariel J. Frank\OS381\os8-2_vir.ppt

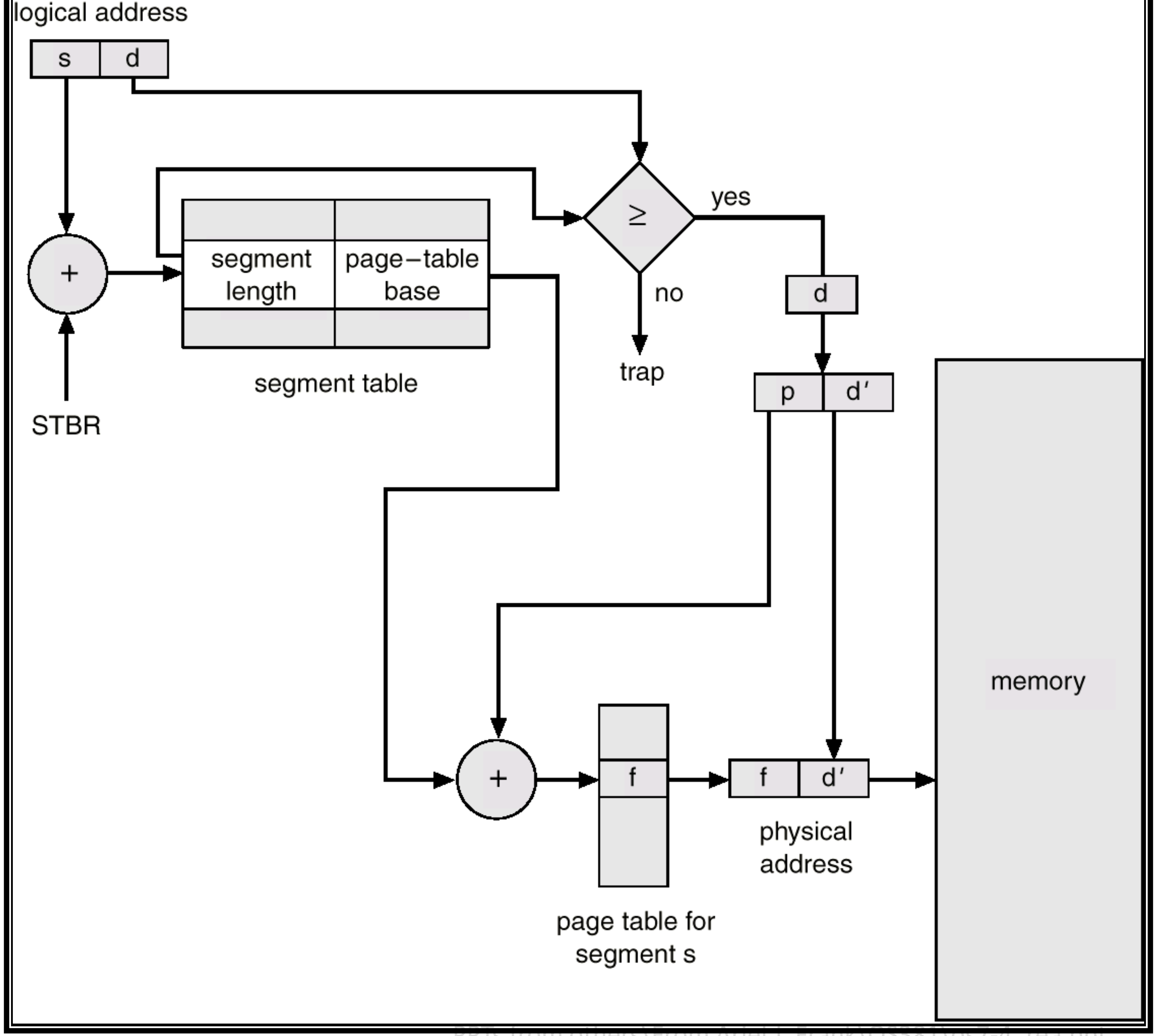
Figure 8.13 Address Translation in a Segmentation/Paging System

Segmentation + Paging: **MULTICS**

- The MULTICS system solved problems of external fragmentation and lengthy search times by paging the segments.
- Solution differs from pure segmentation in that the segment-table entry contains not the base address of the segment, but rather the base address of a *page table* for this segment.

PPTs from others\From Ariel J. Frank\OS381\os7-4_rea.ppt

MULTICS Address Translation Scheme



Case Study – Windows

- Virtual memory with **(on-)demand page**
- Can support 32 or 64 bits
- Has a pool of free frames
- Uses pre-paging (called **clustering**)
- What happens if the amount of free memory falls below some threshold?
  - Each process has a minimum number of processes
  - Windows will take away pages that exceed that minimum
  - Applies LRU Locally

Case Study - Windows

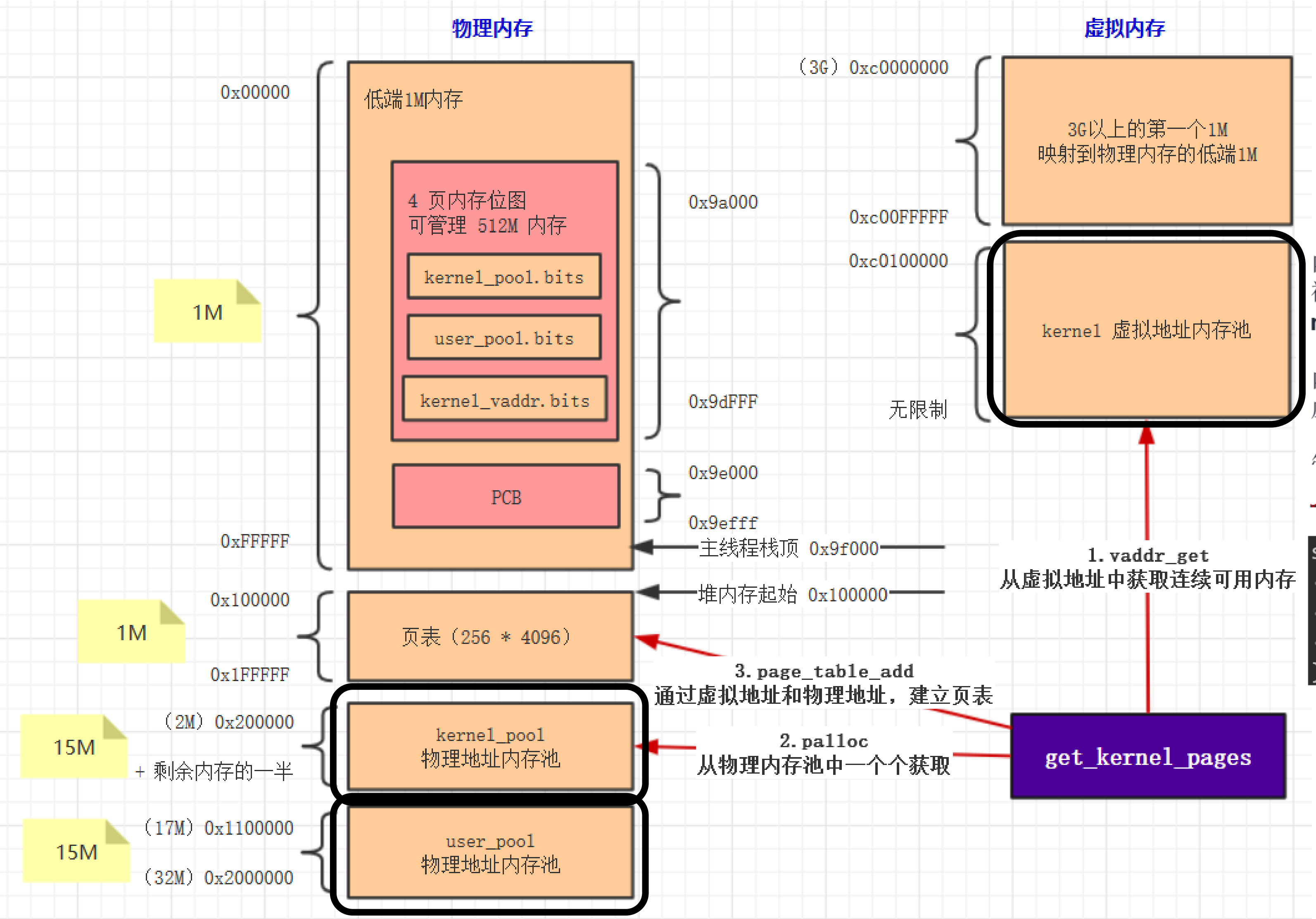
- Each process is guaranteed to have a minimum number of frames
- Each process has a maximum number of frames
- If a page fault occurs for a process that has the maximum number of frames a **local replacement policy** is used
- If a page fault occurs for a process that is below its working set maximum a free frame is used.

Case Study - Linux

- Virtual memory with (on-)**demand paging**
- Can support 32 or 64 bits
- Replacement
  - Least recently used (**LRU**) policy
  - Different implementations for different systems

- ICQ C [10 pts]
  - Next week
  - 1 hour, close
  - Multiple choice, Computation

- Data structures used to manage physical memory



内存池是实现申请内存函数的基础，主要目的就是管理一段内存，说明哪块内存被占用了，哪块内存是空闲的。管理这些内存占用情况的数据结构，用的是 **bit map**，每一个比特对应着一块 4K 的内存。

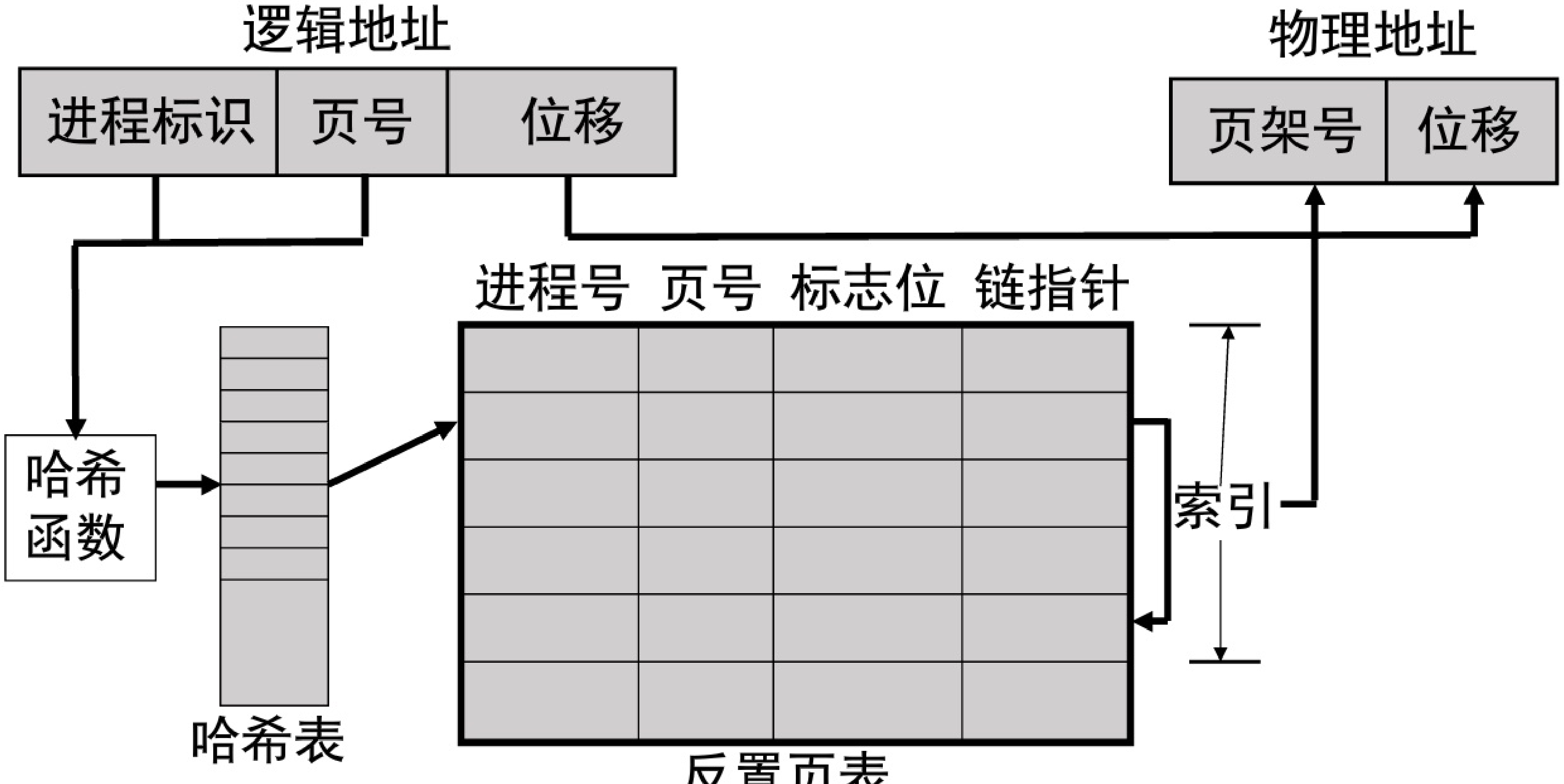
内存池一共分为四个，内核的物理地址内存池、用户的物理地址内存池、内核的虚拟地址内存池、用户的虚拟地址内存池。

管理物理地址的内存池的结构为 **pool**，两个内存池变量为 **kernel_pool**，**user_pool**

```
struct pool {
    struct bitmap pool_bitmap;
    uint32_t phy_addr_start; //本内存池管理的物理内存起始
    uint32_t pool_size;
};
```

https://bbs.huaweicloud.com/blogs/150951

- 反置页表(IPT: Inverted page table)



https://www.cnblogs.com/flashsun/p/10669864.html