Virtual Memory

Read Text 9.3

Real versus Virtual Memory Systems

Basic Concepts about Virtual Memory

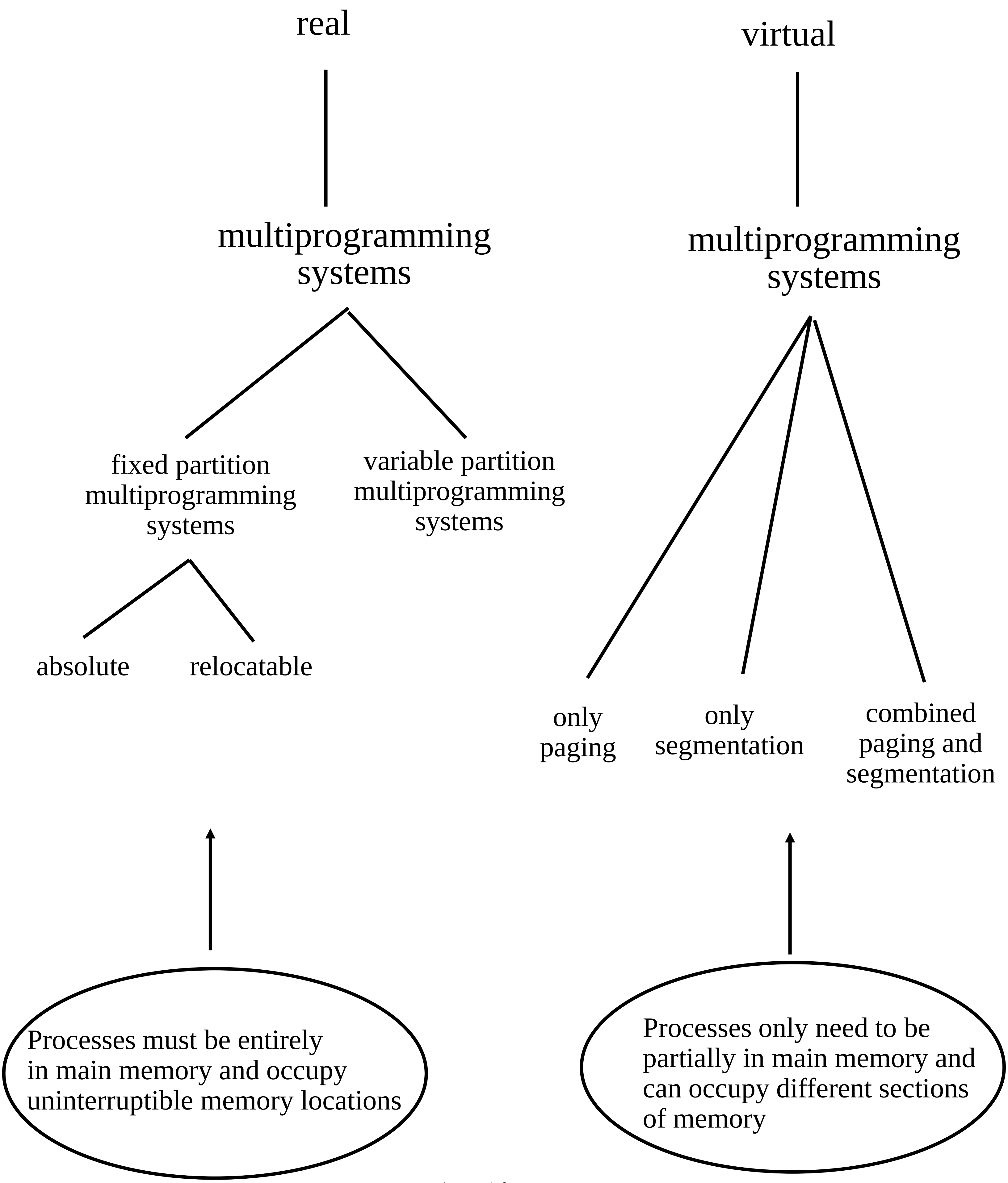
Virtual Address Translation

Paging

Address Translation in Paging

Sharing Pages

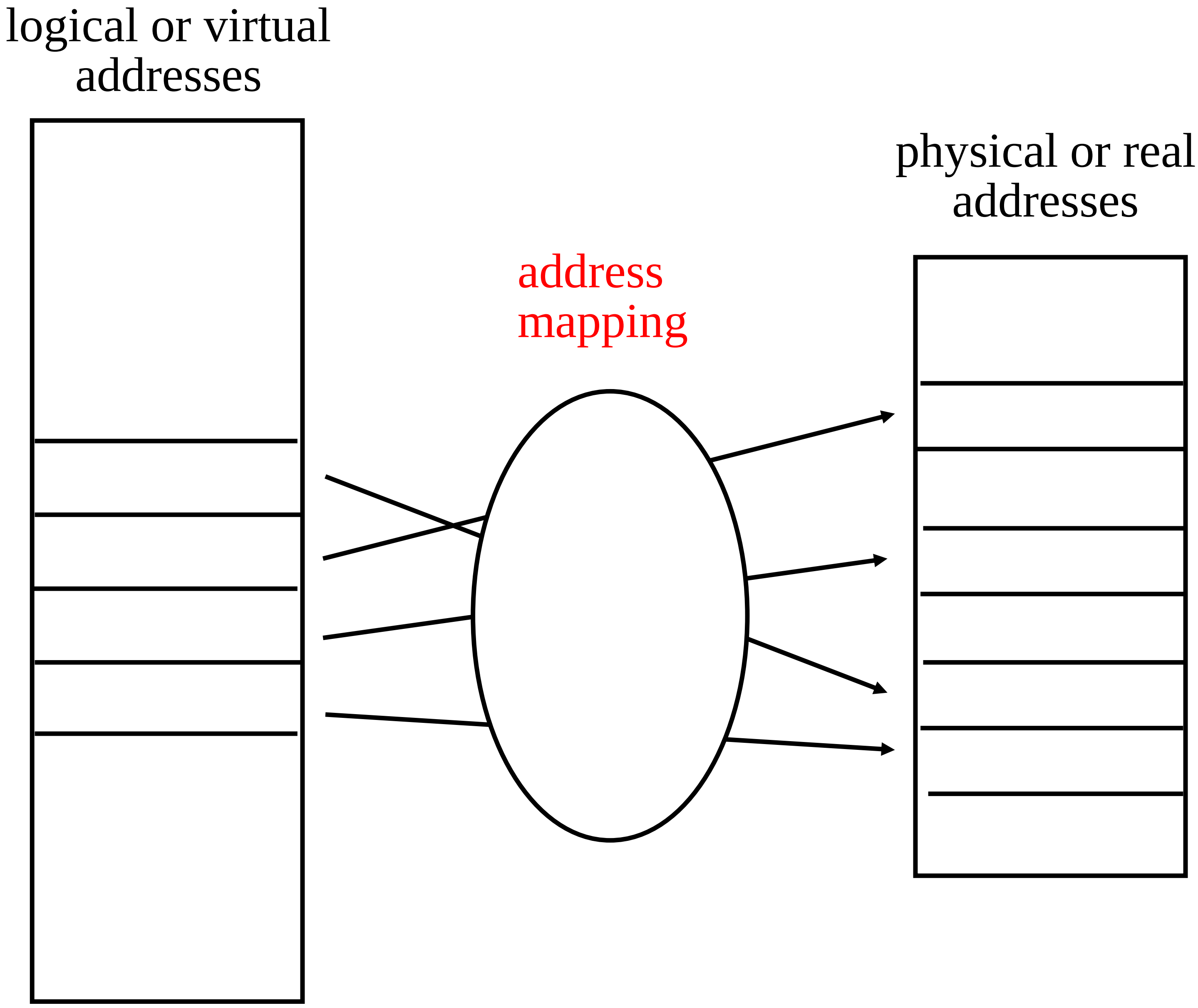
Different Memory Organizations



Virtual Addresses

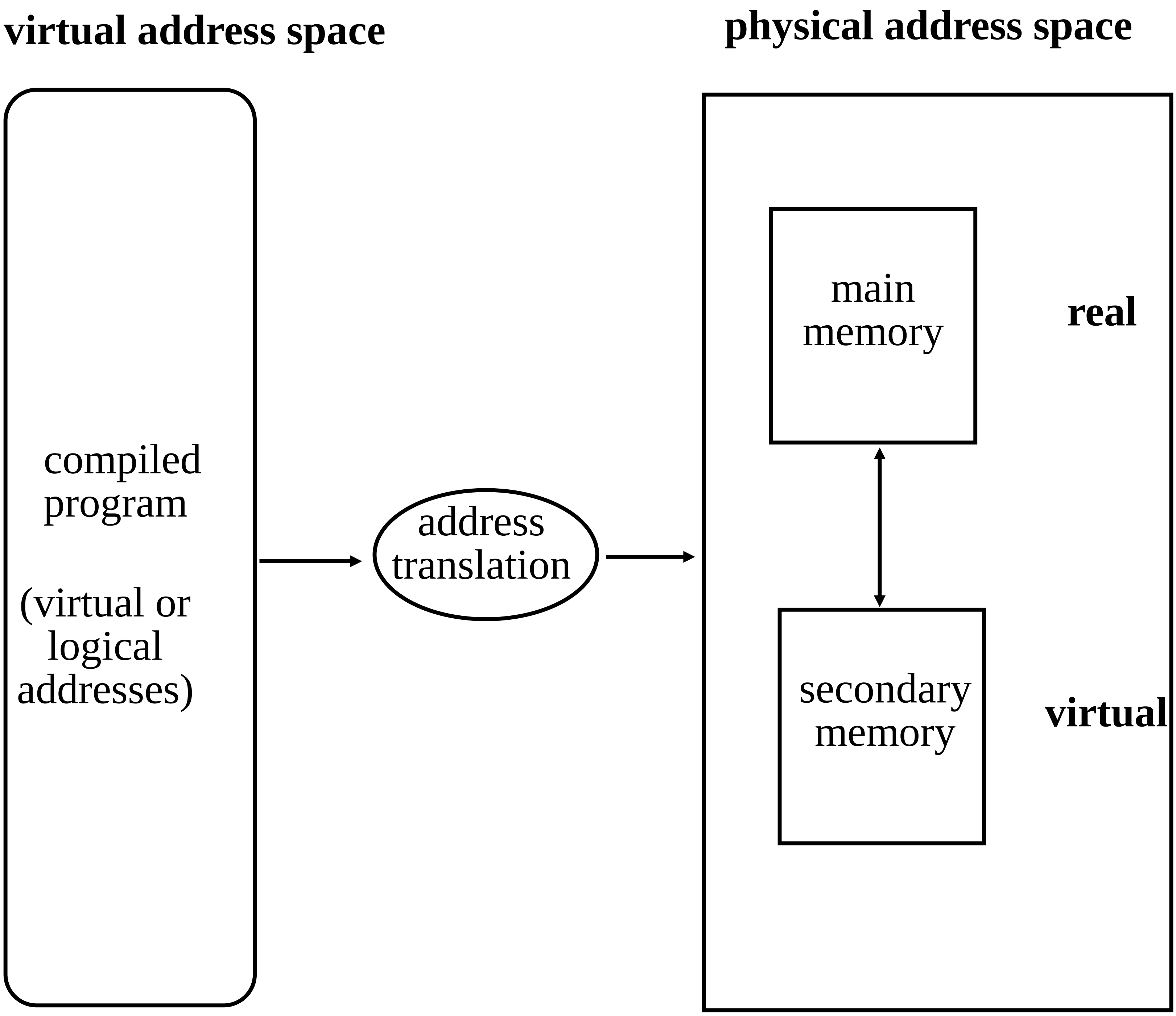
A process can address a storage space (through virtual addresses) larger than that available in main memory (real addresses)

Process instructions must be in main memory before executing. This means that virtual addresses must be converted to real addresses as a process executes



Virtual Memory

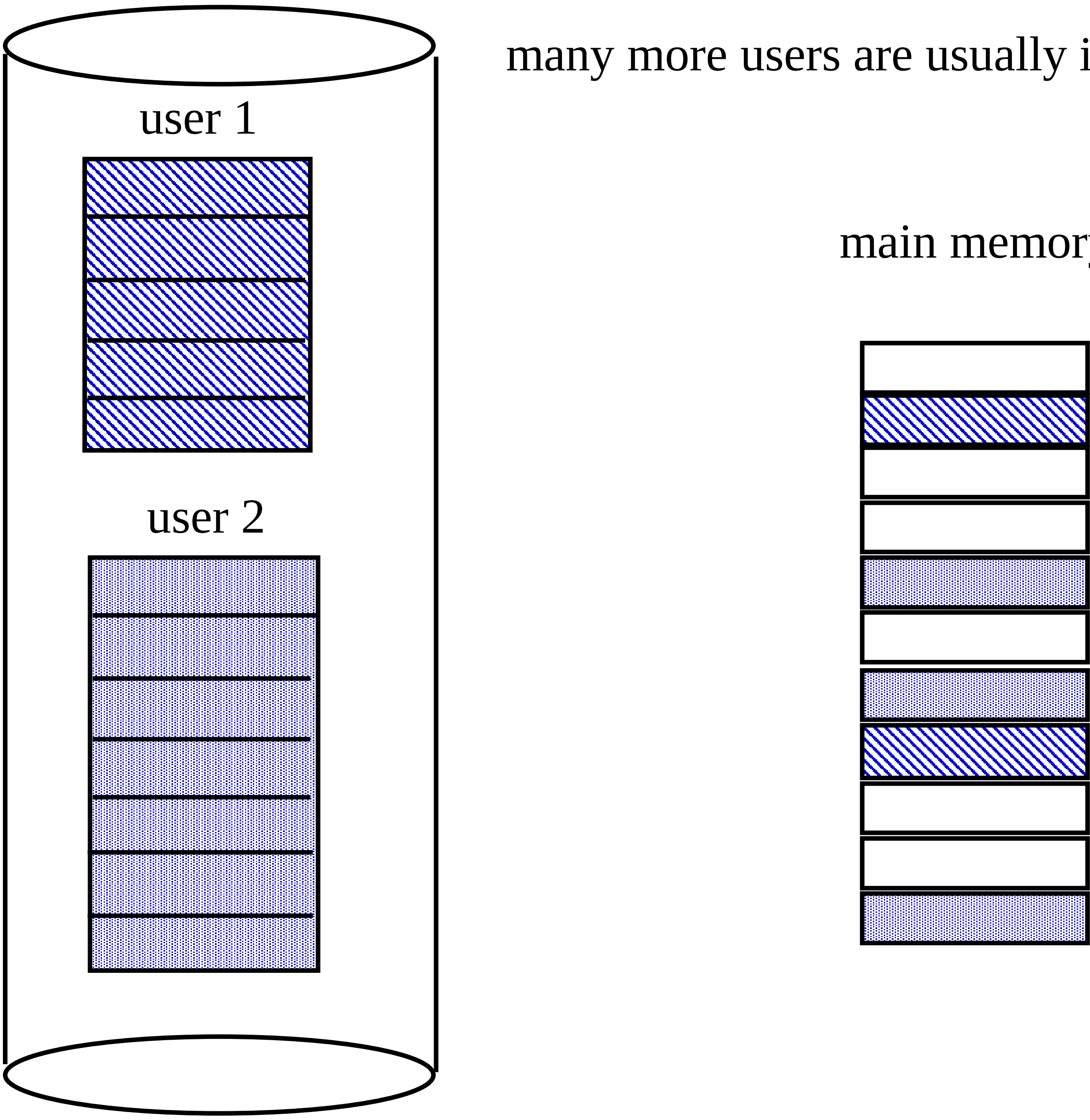
The ability to extend main memory into secondary memory



Multiple Users of Virtual Storage

disk -- virtual

many more users are usually involved



Main memory is shared among multiple users

Main memory holds the process instructions that are presently being executed. and data that is being used.

Advantages of Virtual Memory

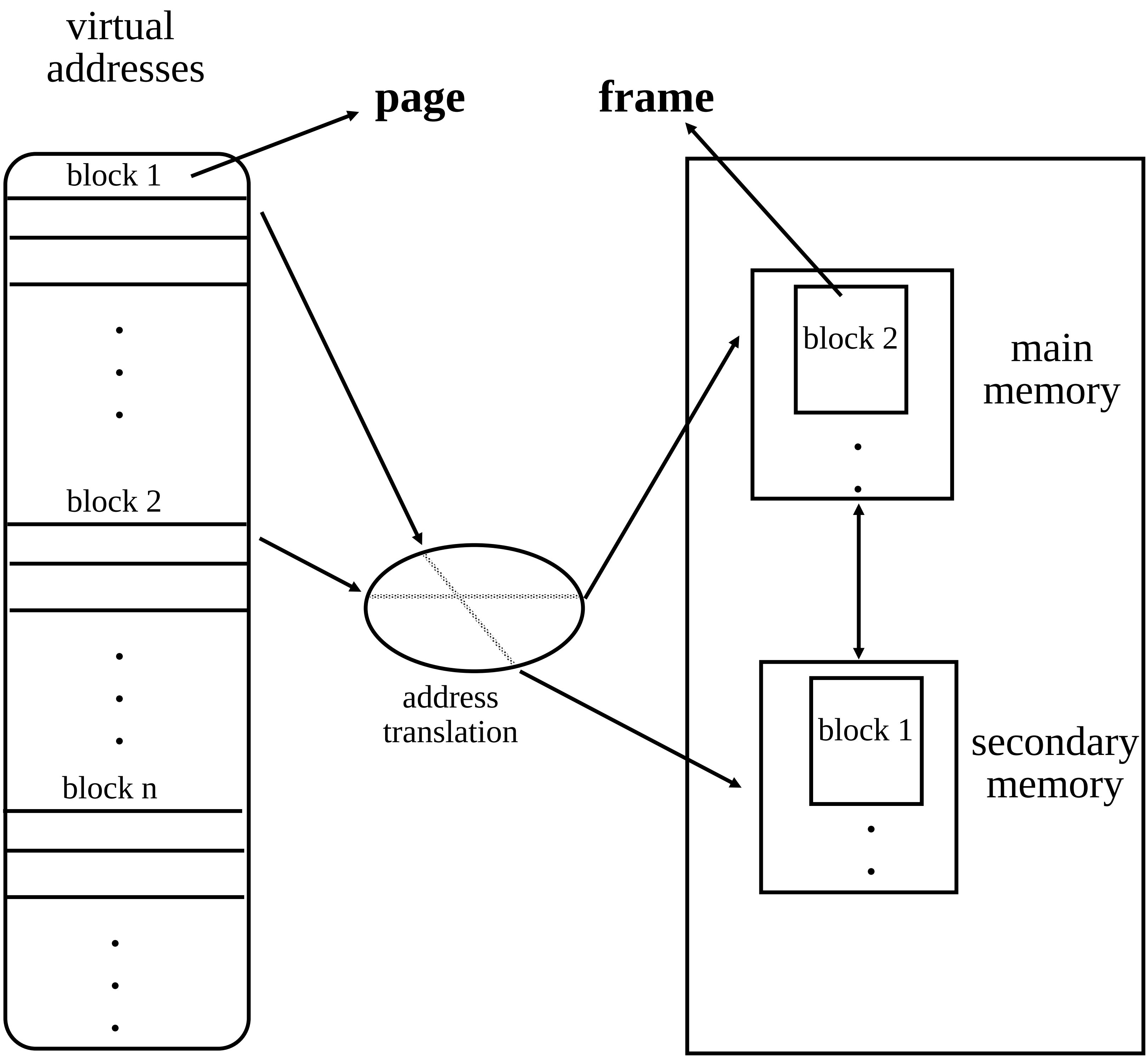
Greater number of processes can execute concurrently because only part of the process needs to be in main memory

External fragmentation is eliminated in paging systems because every frame can be assigned to a process
In segmentation systems it is still possible to have free memory where an entire segment cannot fit.

Programs can be larger in size than available main memory. The logical address space can be 64 bits while the physical address space is much less than 2^{64} bytes.

Virtual Addresses Translation

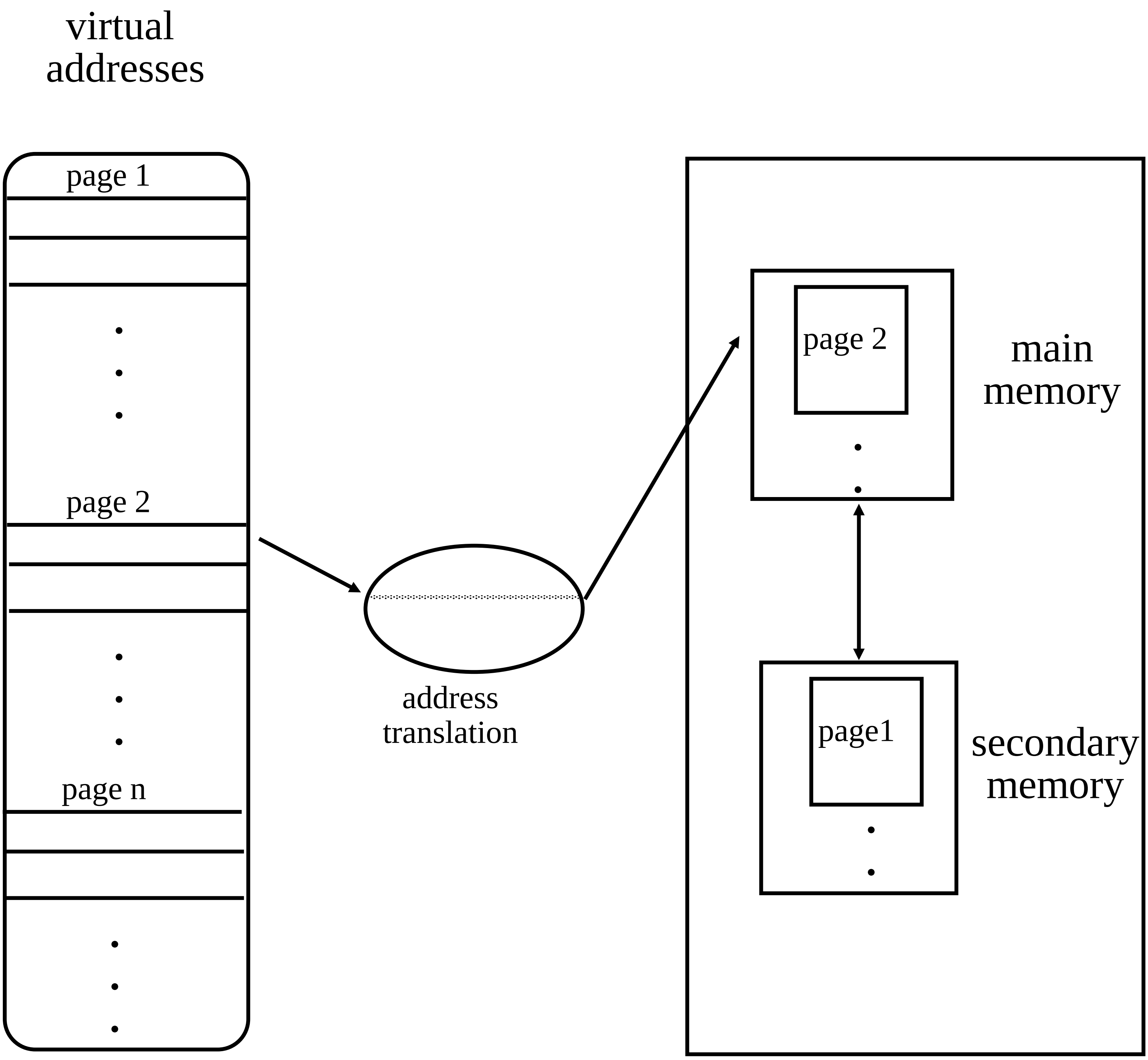
Contiguous blocks of virtual addresses are mapped onto contiguous blocks of locations in real or secondary storage



A one to one mapping of virtual addresses to actual addresses would require a large amount of storage space just to store the address translation information

Paging

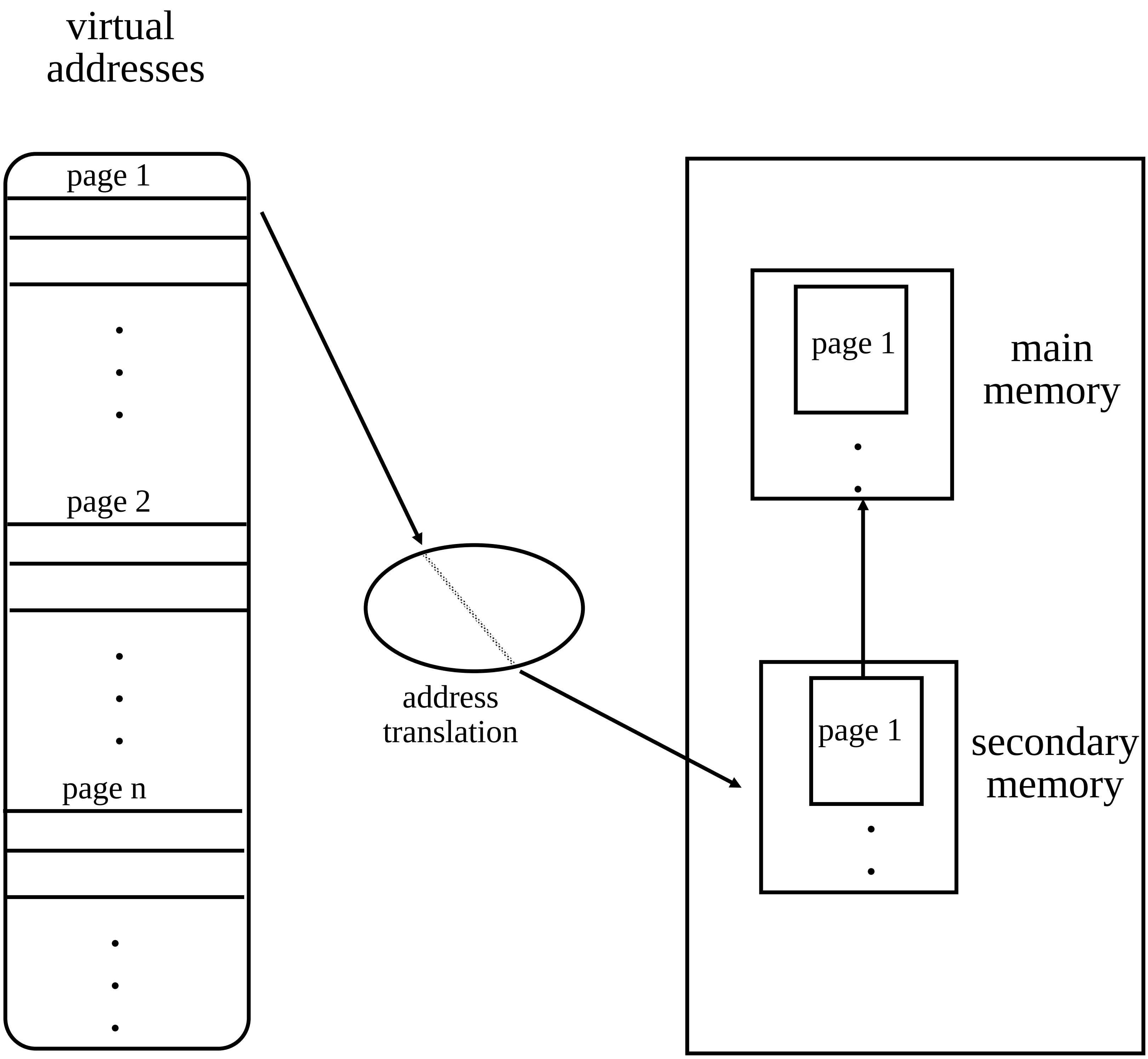
Paging uses virtual blocks that all have the same size
Typically between 512 to 1 GB per page. In Linux, use
"getconf PAGESIZE" to find the page size used by the system



Pages in main memory are called frames and have the same size
as in secondary memory and as in virtual address space.

Paging

If a page is not in main memory it is transferred from secondary storage before the instruction can be executed



The address translation mechanism can determine if a page is in main memory, and if it is not, where in secondary memory it is so that it can be transferred into main memory

Bringing a Page From Disk

1. Generation of a page fault interrupt

    This happens when a logical address is translated and the page table indicates that this page is not in main memory. The process is blocked until the desired page is brought into main memory

2. Service the page fault interrupt

    A free frame is found for the page to be brought in from disk

3. Read the page into main memory

    The disk read operation must wait its turn in the ready queue together with all the other processes waiting for the CPU

4. Restart the process

    The process is unblocked and placed in the ready queue until it is scheduled to run

Paging and **Fragmentation**

Internal Fragmentation - On the average half frame per process

virtual addresses

	0	x1
page0	1	x2
	2	x3
	3	x4
	0	x5
page1	1	x6
	2	x7
	3	x8
page2	0	x9
	1	x10

logical **physical**

1,2 → 2

page table

1	sec	2
1	sec	0
1	sec	4

physical **logical**

9 → 0, 1

explained in next slide

main memory

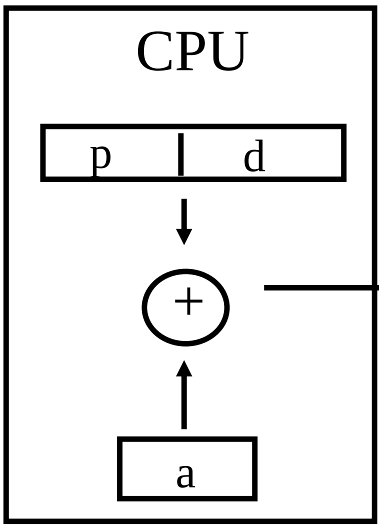
0	x5
	x6
	x7
	x8
4	
8	x1
	x2
	x3
	x4
12	
16	x9
	x10
20	

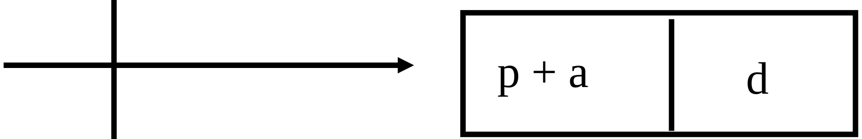
Paging

virtual address = (p d)

"p" is the virtual page number and "d" is the displacement from the start of the page

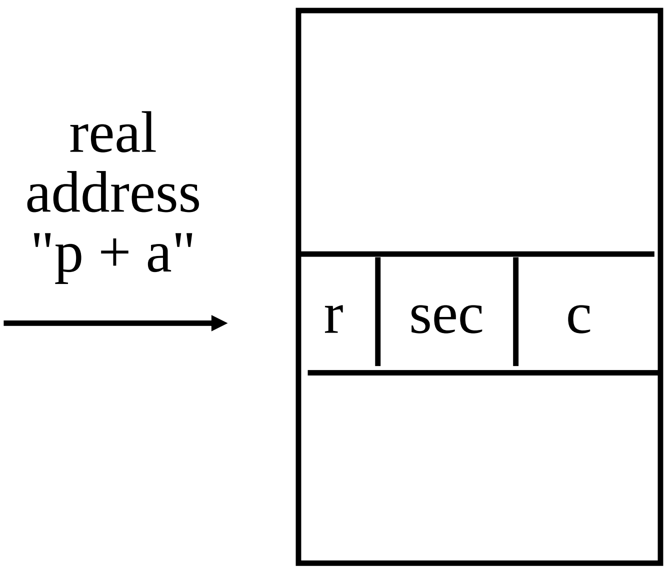
"a" is the base address of the page table for this particular process and is loaded into the CPU at context switching time





"p + a" is the real address where the page information is

Page Table



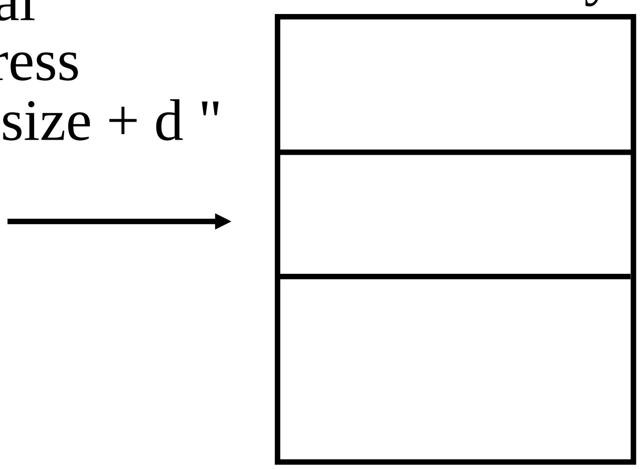
" r " is a residence bit (0,1) that indicates if the page is in real memory or not

" sec " is the disk storage address if page is not in main memory

" c ” is the frame number

real
address
" c* page-size + d "

main memory



" c* page-size + d " is the real address for virtual memory address (p, d)

Example of Page Tables and Frames

process 1 page table

page 0		3
page 1		1
•		
•		

process 2 page table

page 0		0
page 1		2
•		
•		

The page table for each process must be kept in sequential order

main memory **frame size = 4096**

Address	Frame	Process	
0	frame 0	process 2	**Page 0 process 2**
4095	frame 1	process 1	
8191	frame 2	process 2	**Page 1 process 2**
12287	frame 3	process 1	
16383	•		
	•		

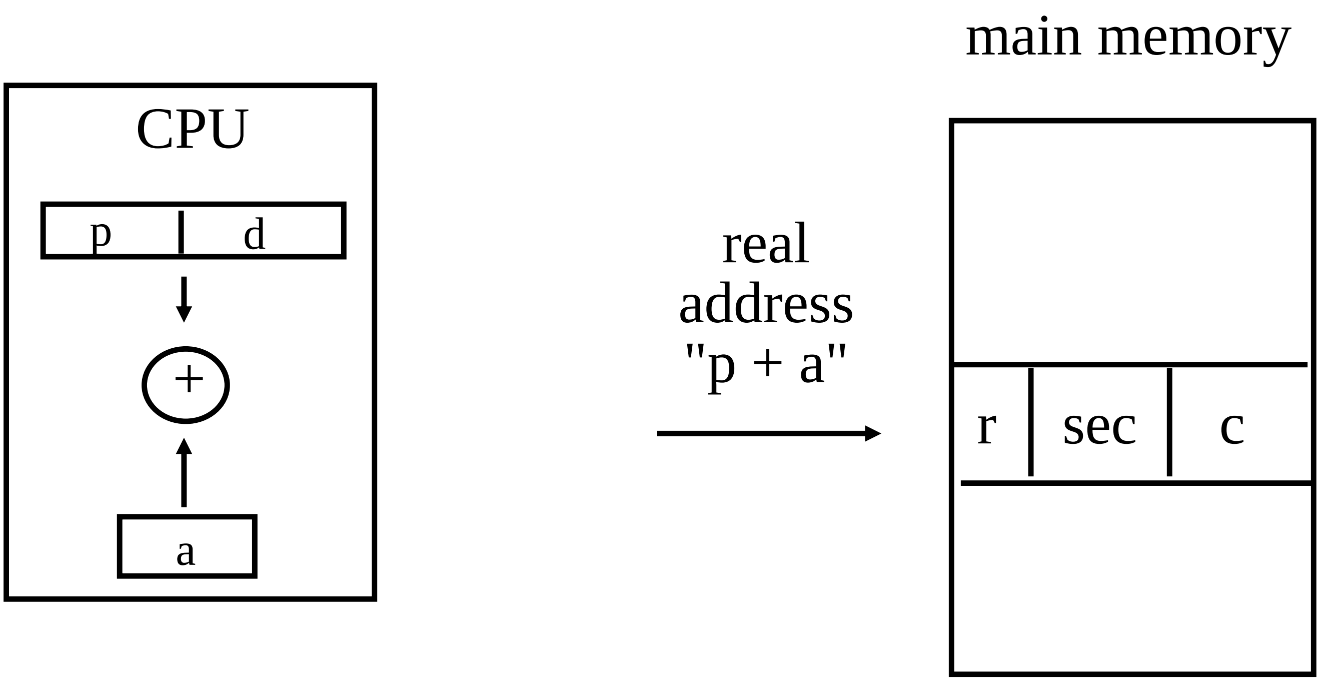
Paging Address Translation Mechanisms

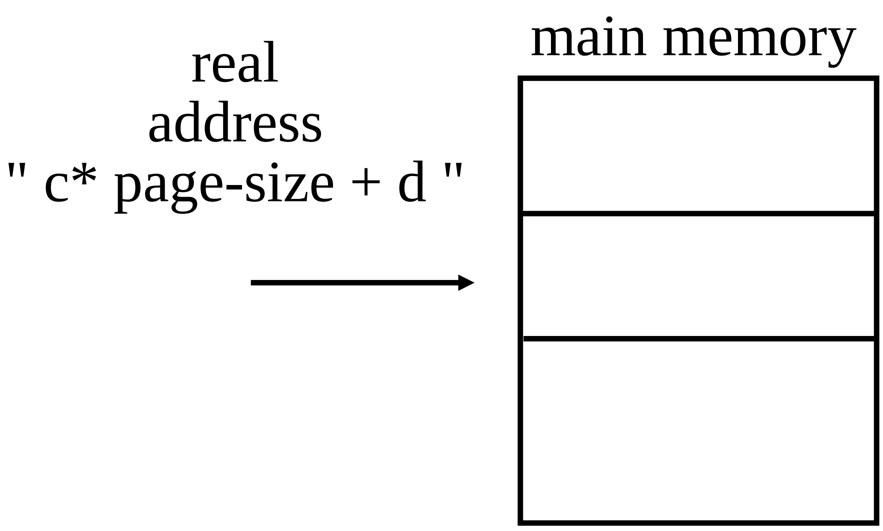
Direct Mapping

read “p + a” location in main memory to get “c”

access "c* page-size + d ” in main memory

2 memory accesses instead of 1





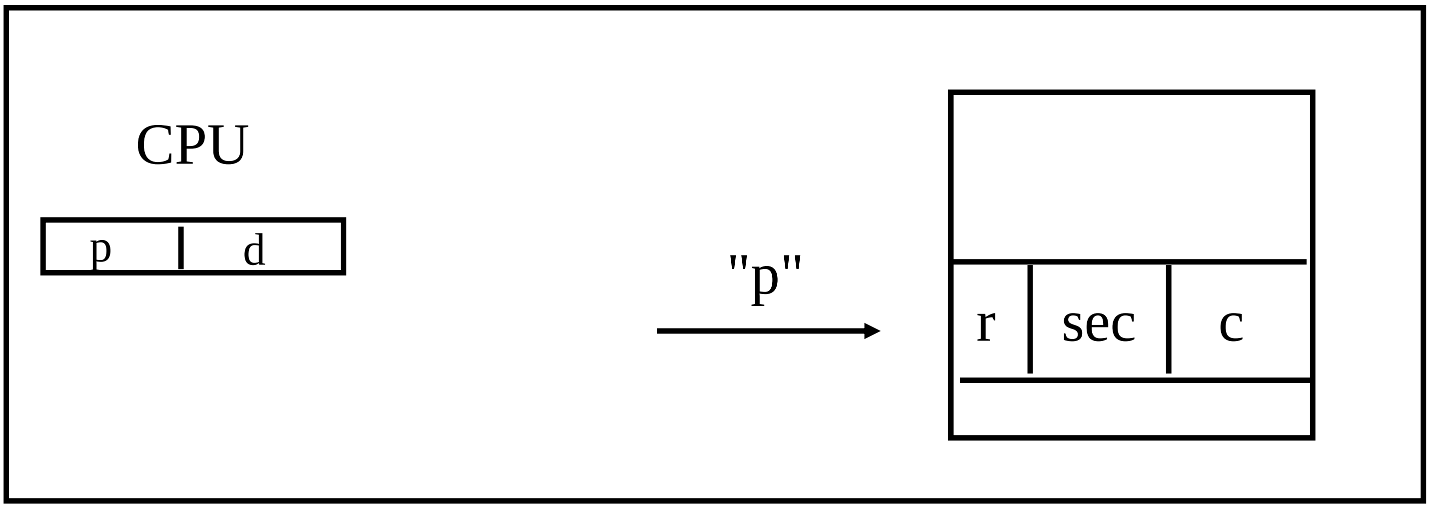
Paging Address Translation Mechanisms

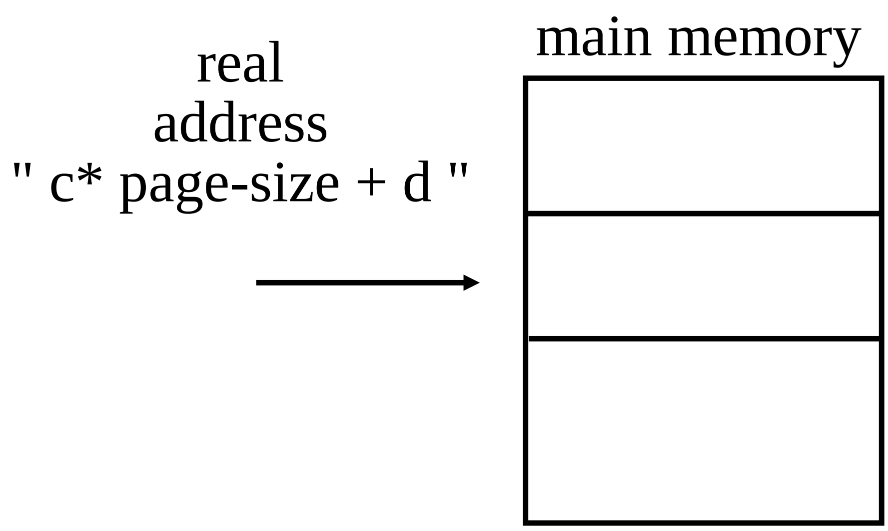
CPU Registers

read “p ” in CPU registers to get “c”

access "c* page-size + d ” in main memory

1 register access
1 memory access





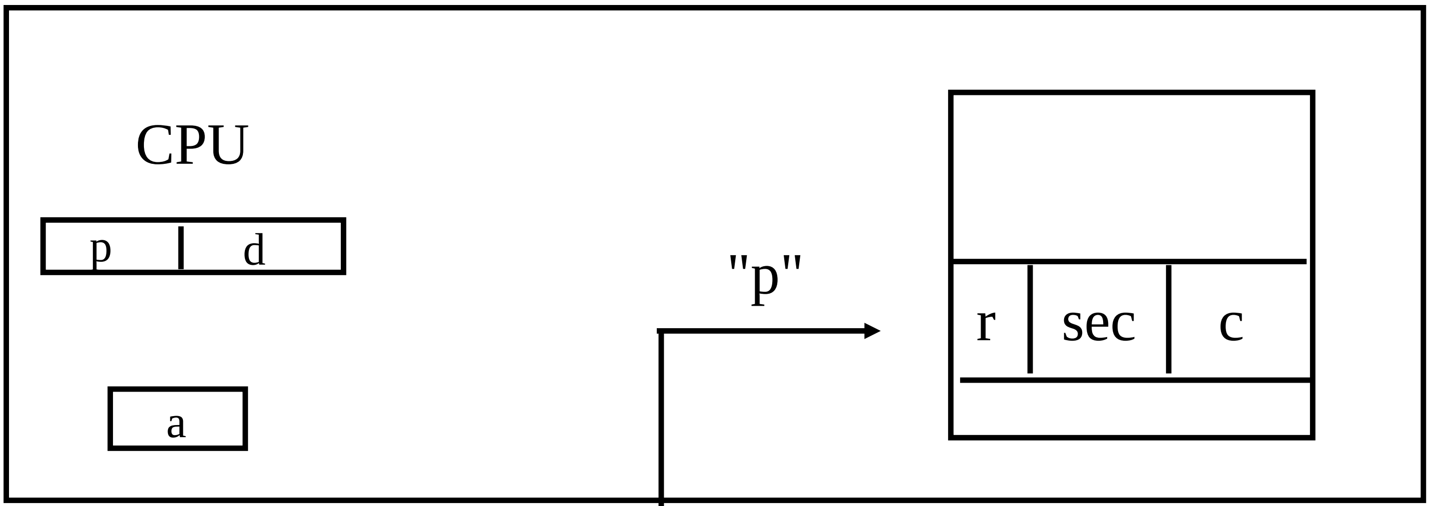
But - How many pages are processes allowed to have ?

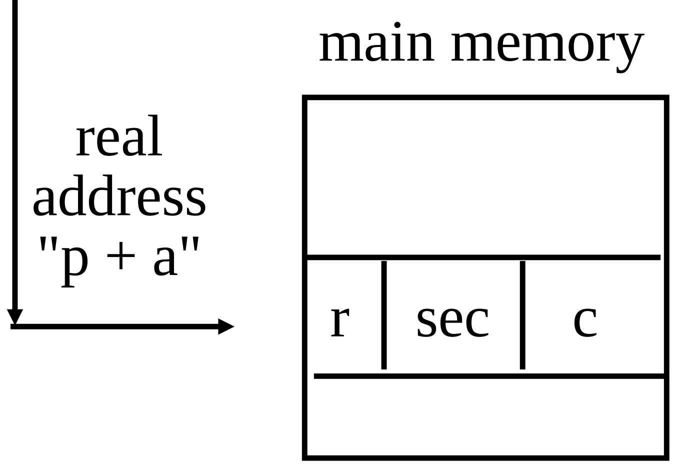
Paging Address Translation Mechanisms

Combined CPU Registers and Direct Mapping

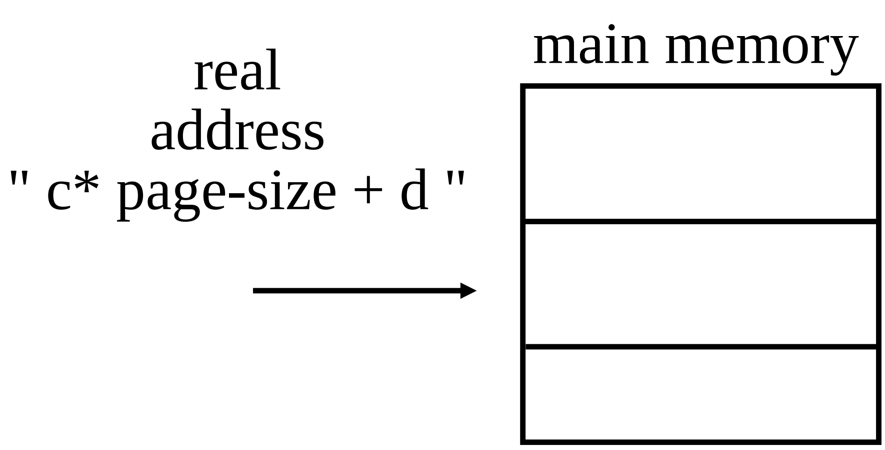
read “p ” in CPU registers to get “c”
If “p” not here then read “p+a” in main memory to get c
access "c* page-size + d ” in main memory

MIN	1 register access 1 memory access
MAX	1 register access 2 memory access





Read text description of
“translation look-aside buffer”
# of locations
Access speed
“wired-down” entries

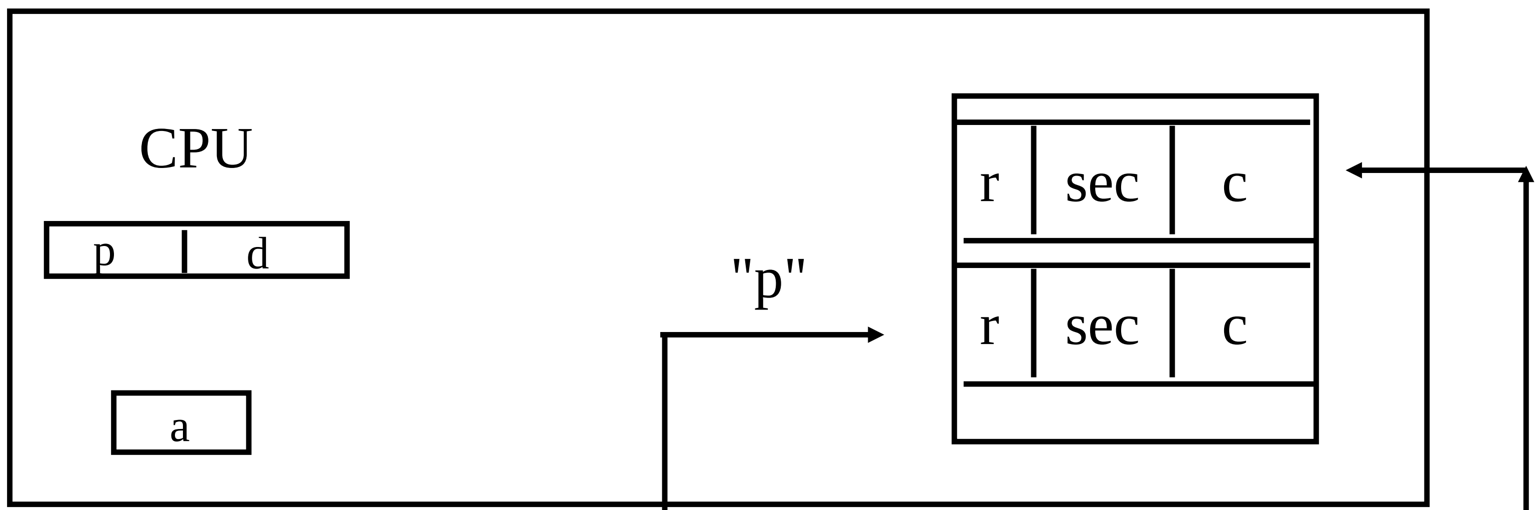


Paging Address Translation Mechanisms

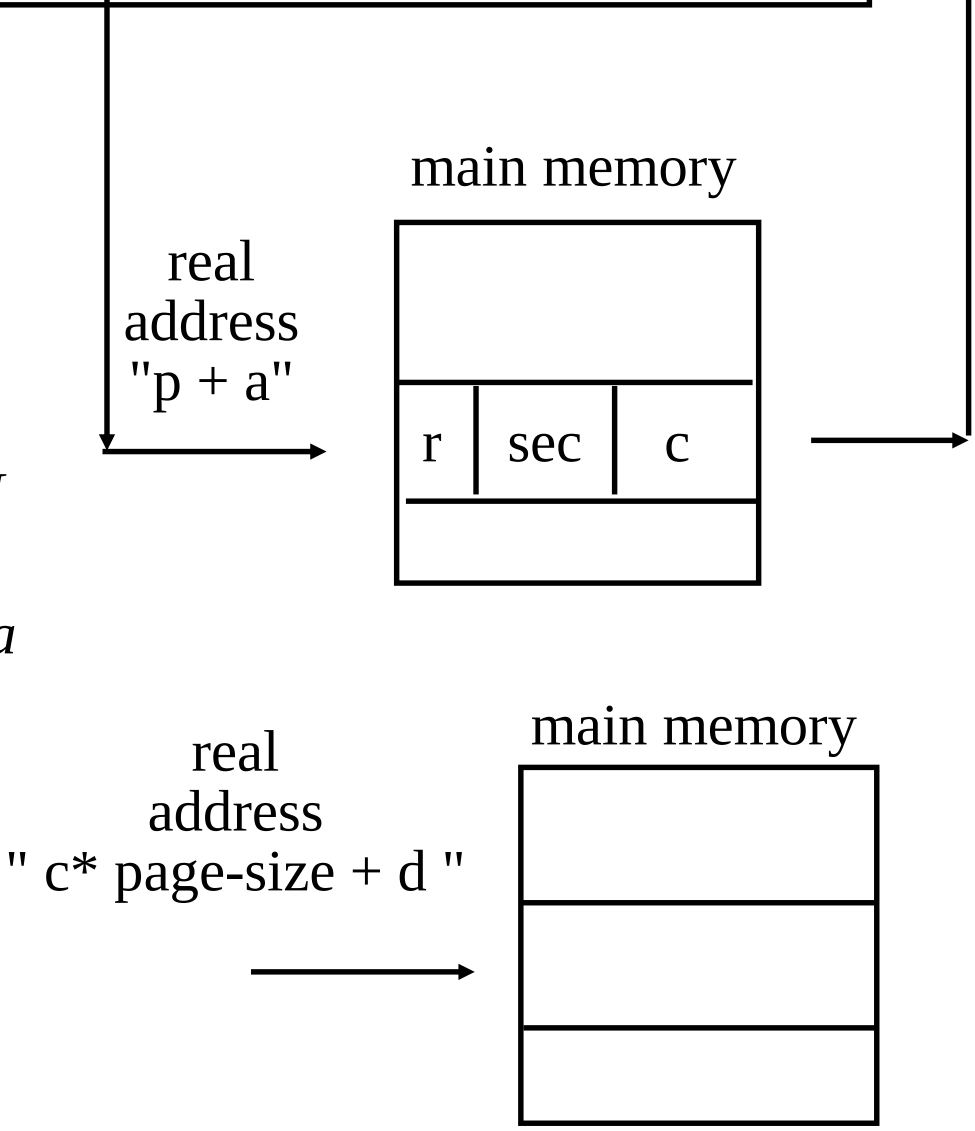
Combined CPU Registers and Direct Mapping

When p is not found in CPU registers, it is brought in from main memory

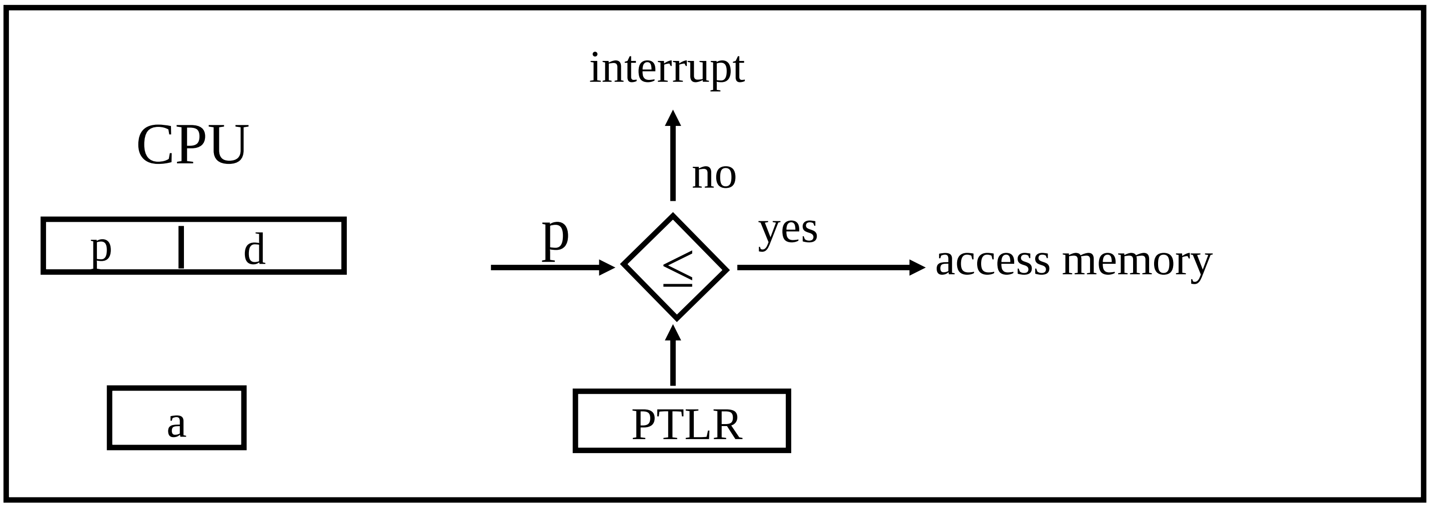
It replaces the page number in CPU registers least likely to be referenced



What contributes to the high "hit ratio" (99%) claimed even though the number of registers in CPU is small when compared to a large number of pages in a process ?



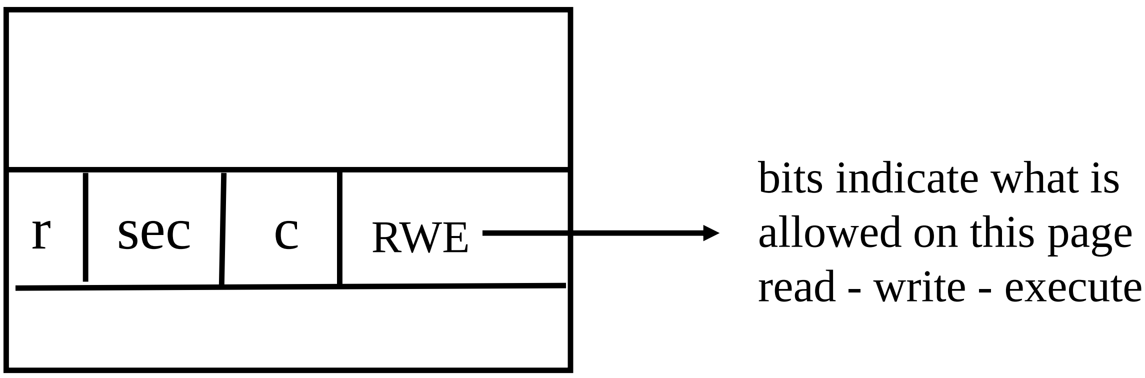
Main Memory Protection for Paging



PTLR = Page Table Length Register

How can a process access another process address space by generating a page number larger than its own?

Page Table



Why should a process not be allowed to write to its own address space?

See next page!

Sharing Pages

In timesharing systems it is common to have several users using the same system program.– like the "standard C library"

A paging system can implement sharing of a system program by including its page information in each user's page table.