University of Regina
Computer Science

Bao Cao
200363431

CS 330 - Assignment 3

Question 2:

Length of logical address = 64 (pages) * 1024 (words - 1 word = 8 bits) * 8 (bits) = 524288 (bits)

Length of physical address = 256 (frames - each frame has the same length as a page) * 1024 * 8
= 2097152 (bits)

Since there are 256 frames, 8 bits are needed in each entry to represent frame numbers from 0 to 255.

Since there are 64 pages, 6 bits are needed to represent page numbers from 0 to 63.

Therefore, there are 14 bits in each table entry. Since there are 64 pages which means 64 entries, The length of a page table entry is 64 * 14 = 896 (bits)

Question 3:

a) First-fist (left column is partition size, right column is process size)

300 KB	
600 KB	
350 KB	
200 KB	
750 KB	
125 KB	

300 KB	115 KB
600 KB	
350 KB	
200 KB	
750 KB	
125 KB	

300 KB	115 KB
600 KB	500 KB
350 KB	
200 KB	
750 KB	
125 KB	

300 KB	115 KB
600 KB	500 KB
350 KB	
200 KB	
750 KB	358 KB
125 KB	

300 KB	115 KB
600 KB	500 KB
350 KB	200 KB
200 KB	
750 KB	358 KB
125 KB	

The last process could not be allocated space since there is no available space which is big enough

b) Best-fit

300 KB	
600 KB	
350 KB	
200 KB	
750 KB	
125 KB	

300 KB	
600 KB	
350 KB	
200 KB	
750 KB	
125 KB	115 KB

300 KB	
600 KB	500 KB
350 KB	
200 KB	
750 KB	
125 KB	115 KB

300 KB	
600 KB	500 KB
350 KB	
200 KB	
750 KB	358 KB
125 KB	115 KB

300 KB	
600 KB	500 KB
350 KB	
200 KB	200 KB
750 KB	358 KB
125 KB	115 KB

The last process could not be allocated space since there is no available space which is big enough

c) Worst-fit

300 KB	
600 KB	
350 KB	
200 KB	
750 KB	
125 KB	

300 KB	
600 KB	
350 KB	
200 KB	
750 KB	115 KB
125 KB	

300 KB	
600 KB	500 KB
350 KB	
200 KB	
750 KB	115 KB
125 KB	

300 KB	
600 KB	500 KB
350 KB	
200 KB	
750 KB	115 KB
125 KB	

300 KB	
600 KB	500 KB
350 KB	200 KB
200 KB	
750 KB	115 KB
125 KB	

Two processes with the size of 358 KB and 375 KB could not be allocated space since there are no available spaces which are big enough

Question 4:

Because when a paging system is used, every address which is used by a process will be translated by its own page table. Therefore, these addresses cannot be used in a different process which has a different page table.

Question 5:

a) page number = 3085 / 4096 = 0

offset = 3085 % 4096 = 3085

b) page number = 42095 / 4096 = 10

offset = 42095 % 4096 = 1135

c) page number = 215201 / 4096 = 52

offset = 215201 % 4096 = 2209

d) page number = 1048576 / 4096 = 256

offset = 1048576 % 4096 = 0

e) page number 16777217 / 4096 = 4096

offset = 16777217 % 4096 = 1

Question 6:

a) Since page table lookup time is equal to a physical memory reference which is 50 nanoseconds, a paged memory reference = page table lookup time + physical memory reference = 100 nanoseconds.

b) TLB lookup time = 2 nanoseconds

TLB access time = TLB lookup time + physical memory reference = 2 + 50 = 52 nanoseconds

hit ratio = 75%   ====> miss ratio = 25%

effective memory access time = hit ratio * TLB access time + miss ratio * page table access time

= 0.75 * 52 + 0.25 * 100 = 64 (nanoseconds)

Question 7:

a) Segment 0 begins at physical address 219 with length is 600. A reference to byte 430 of segment 0 in the logical address space is mapped to physical address 219 + 430 = 649

b) Segment 1 begins at physical address 2300 with length is 14. A reference to byte 10 of segment 1 in the logical address space is mapped to physical address 2300 + 10 = 2310

c) Segment 2 begins at physical address 90 with length is 100. A reference to byte 500 of segment 2 to the physical address would result in a trap to the operating system as segment 2 is only 100 byte long.

d) Segment 3 begins at physical address 1327 with length is 580. A reference to byte 400 of segment 3 in the logical address space is mapped to physical address 1327 + 400 = 1727

e) Segment 4 begins at physical address 1952 with length is 96. A reference to byte 112 of segment 4 to the physical address would result in a trap to the operating system as segment 4 is only 96 byte long.

Question 8:

Basic method for implementing paging:

Each process which is stored in logical memory is divided into fix-sized blocks called pages.

Page 0
Page 1
Page 2
Page 3

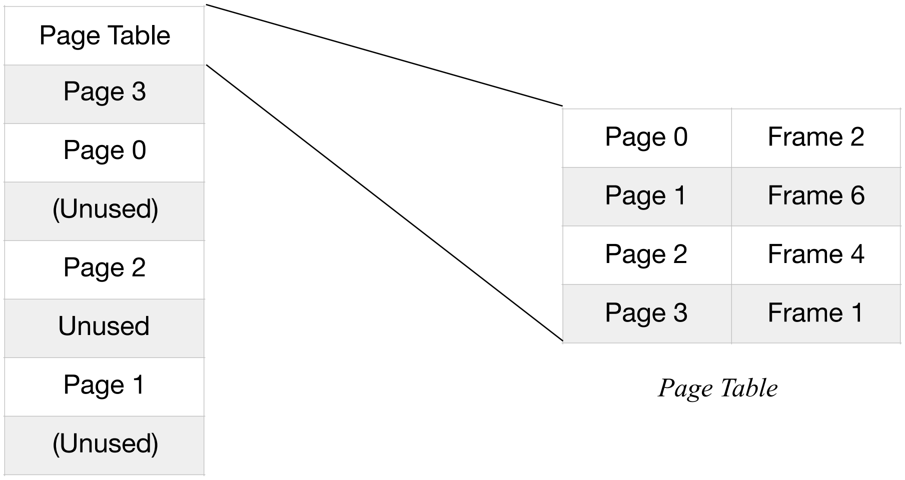
Logical memory

Physical memory is divided into fix-sized blocks called frames

Frame 0	(Unused)
Frame 1	Page 3
Frame 2	Page 0
Frame 3	(Unused)
Frame 4	Page 2
Frame 5	Unused
Frame 6	Page 1
Frame 7	(Unused)

Physical memory

A structure called *page table* is use to record where each page in logical memory is placed in physical memory (each process has its own page table). Page table can be stored in main memory.



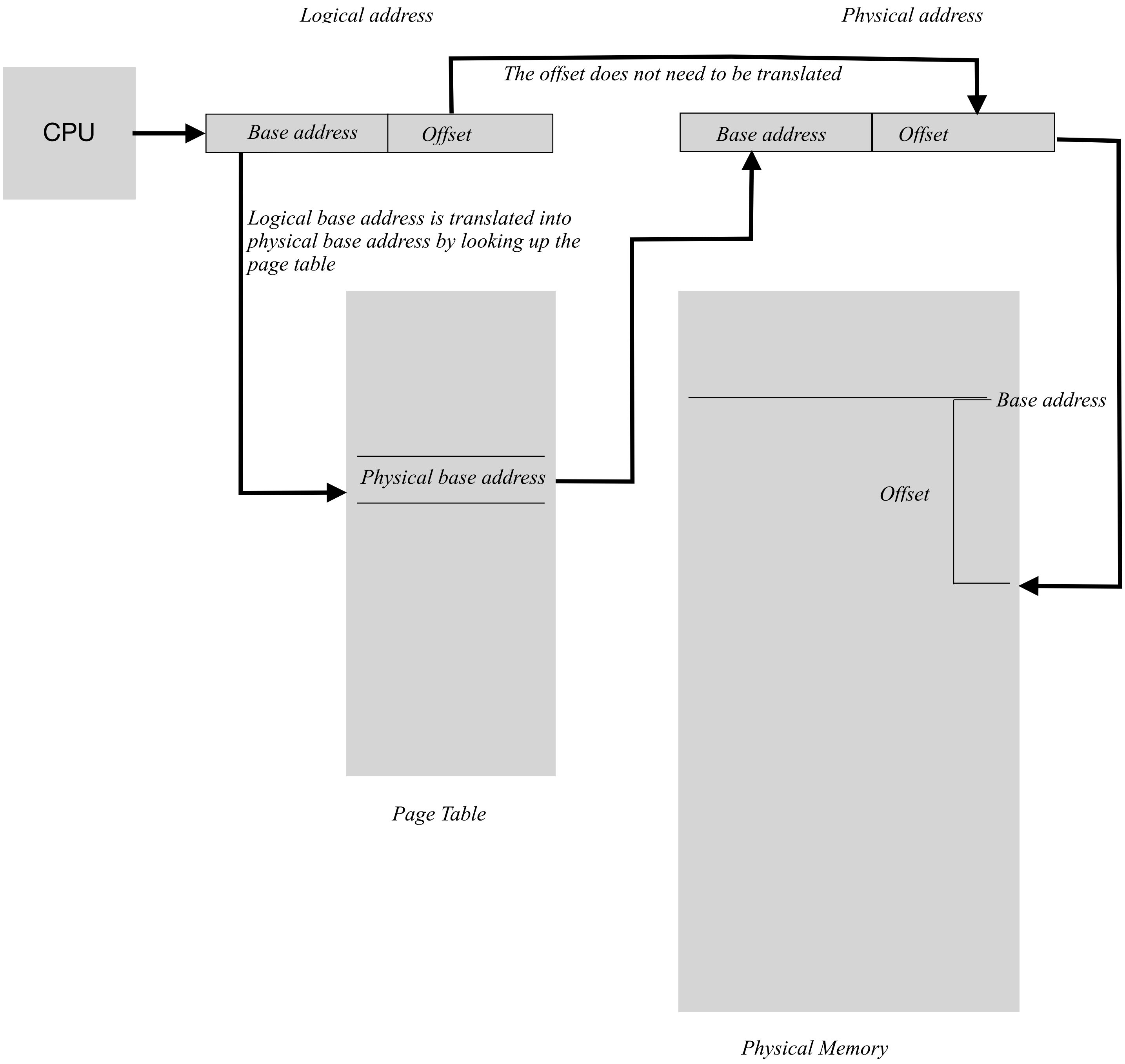
Page Table
Page 3
Page 0
(Unused)
Page 2
Unused
Page 1
(Unused)

Page 0	Frame 2
Page 1	Frame 6
Page 2	Frame 4
Page 3	Frame 1

Page Table

The backing store is used by the paging system to store information which is not currently in main memory, which facilitates efficient swapping. Backing store is divided into fix-sized blocks of size of frames

Hardware support is provided to convert process' logical address into physical address in main memory. When a process is run, its pages will be loaded into any available frames which doesn't need to be contiguous.


Logical address
Physical address
The offset does not need to be translated
CPU
Base address
Offset
Base address
Offset
Logical base address is translated into physical base address by looking up the page table
Physical base address
Base address
Offset
Page Table
Physical Memory