COMP 3511
Operating Systems

Lab 06 Review

Q. 1

- What is required to support dynamic memory allocation in the following schemes:
  - *contiguous-memory allocation*
  - *pure paging*
  - *pure segmentation*

Q. 1

- contiguous-memory allocation:
  - might require relocation of the entire program
  - since there is not enough space for the program to grow its allocated memory space

- pure paging:
  - incremental allocation of new pages is possible in this scheme without requiring relocation of the program’s address space

Q. 1

- pure segmentation:
  - might require relocation of the segment that needs to be extended
  - since there is not enough space for the segment to grow its allocated memory space

Q. 2

- Briefly explain the concept of *logical address* and *physical address*
- Logical address
  - Generated by the CPU
  - Also referred to as *virtual address, i.e., the address starts from* zero
- Physical address
  - Address seen by the memory management unit (MMU) which maps logical address to physical address
- The user program deals with *logical addresses*; it never sees the real *physical addresses*

Q. 3

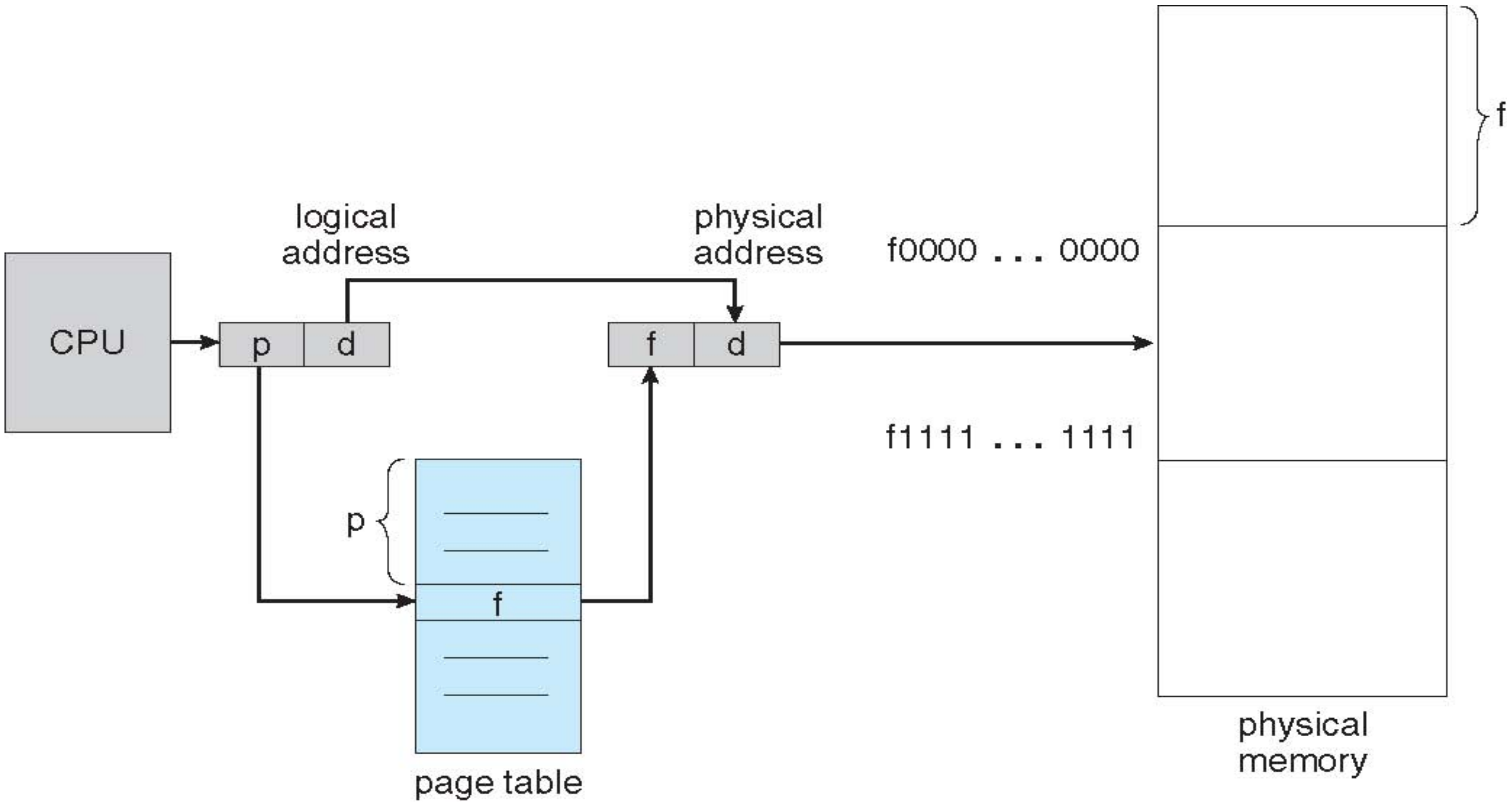
- Suppose a computer has an 8-bit address space, i.e., each logical address is 8-bit long. Each Page has size of 32 Bytes.

  (a) How many entries does the page table contain?

  (b) Part of the page table is shown here:

Page Number	Frame Number
0	5
1	1
2	3
3	2
4	7

Q. 3



logical address
physical address
f0000 . . . 0000
CPU
p
d
f
d
f1111 . . . 1111
p
f
f
page table
physical memory

Q. 3

- What are the physical addresses in decimal for the following logical addresses in binary?

  i. 00111111

  ii. 11000000

  iii. 10101010

  iv. 01010101

Q. 3

- Suppose a computer has an 8-bit address space, i.e., each logical address is 8-bit long. Each Page has size of 32 Bytes.

  (a) How many entries does the page table contain?

  Answer:

a) (a) 3 bits are left for the page number in logical address, so there are total 8 entries in the page table.

b) (b)

   i. 1 x 32 + 31 = 63
   ii. No translation can be done
   iii. No translation can be done
   iv. 3 x 32 + 21 = 117

Q. 4

- Consider a paging system with the page table stored in memory

- If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?

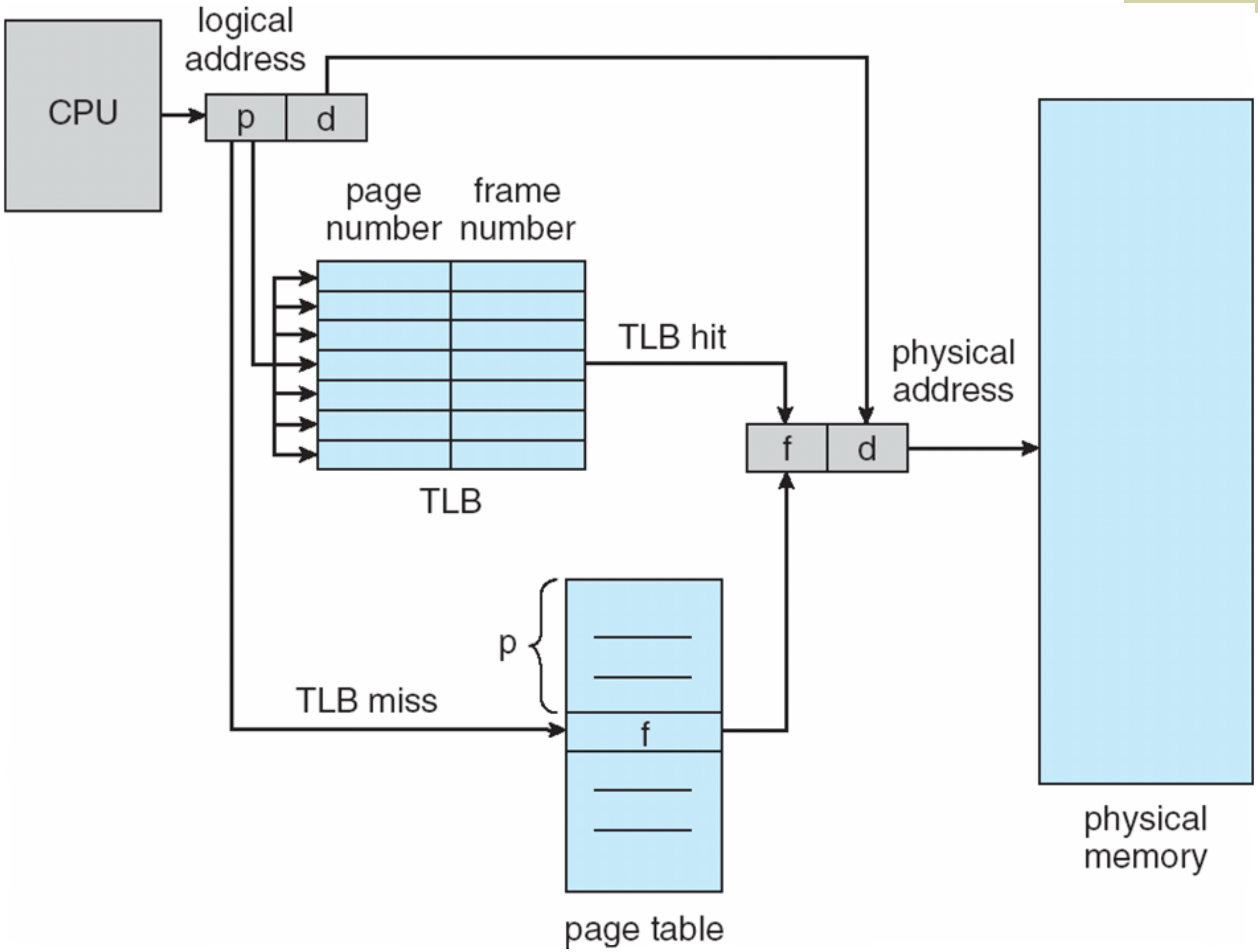
Q. 4

- 400 nanoseconds:
  - 200 nanoseconds to access the page table
  - 200 nanoseconds to access the word in memory

Q. 4

- Assume that finding a page-table entry in the associative registers takes zero time, if the entry is there
- If we add associative registers, and 75% of all page-table references are found in the associative registers
- what is the effective memory reference time?

Paging Hardware With TLB



Q. 4

- Effective access time

  = 0.75 × (200 nanoseconds) + 0.25 × (400 nanoseconds)

  = 250 nanoseconds.

Q. 5

- Compare the memory organization schemes of contiguous memory allocation, pure segmentation, and pure paging with respect to the following issues

- External fragmentation
- Internal fragmentation
- Ability to share code across processes

Q. 5

- **External Fragmentation** – total memory space exists to satisfy a request, but it is not contiguous
- **Internal Fragmentation** – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used

Q. 5

- The contiguous memory allocation scheme suffers from external fragmentation
  - Address spaces are allocated contiguously and holes develop as old processes die and new processes are initiated
- It also does not allow processes to share code
  - Process's memory space is not broken into noncontiguous fine-grained segments

Q. 5

- Pure segmentation also suffers from external fragmentation
  - A segment of a process is laid out contiguously in physical memory and fragmentation would occur as segments of dead processes are replaced by segments of new processes
- It enables processes to share code
  - For instance, two different processes could share a code segment but have distinct data segments

Q. 5

- Pure paging does not suffer from external fragmentation, but instead suffers from internal fragmentation
  - Processes are allocated in page granularity and if a page is not completely utilized, it results in internal fragmentation and a corresponding wastage of space
- Paging also enables processes to share code at the granularity of pages