Solution

Internal Assessment Test 2 – June 2021

Sub:	Operating Systems							Code:	18CS43
Date:	22-6-2021	Duration:	90mins	Max Marks:	50	Sem:	IV	Branch:	ISE

SECTION A

DESCRIPTIVE QUESTIONS – 30 marks

Note: Answer Any Five Questions (5*6 = 30)

1. **List the classical Problems of synchronization and explain any one in detail.**

Classic Problems of Synchronization

1. Bounded-Buffer Problem
2. Readers and Writers Problem
3. Dining-Philosophers Problem

The Bounded-Buffer Problem

- The bounded-buffer problem is related to the producer consumer problem.
- There is a pool of n buffers, each capable of holding one item.

```
int n;
semaphore mutex = 1;
semaphore empty = n;
semaphore full = 0
```

- Shared-data

where,

- ¤ mutex provides mutual-exclusion for accesses to the buffer-pool.
- ¤ empty counts the number of empty buffers.
- ¤ full counts the number of full buffers.

- The symmetry between the producer and the consumer.
  - ¤ The producer produces full buffers for the consumer.
  - ¤ The consumer produces empty buffers for the producer.

Producer Process:

```
do {
    . . .
   /* produce an item in next_produced */
    . . .
   wait(empty);
   wait(mutex);
    . . .
   /* add next_produced to the buffer */
    . . .
   signal(mutex);
   signal(full);
} while (true);
```

Consumer Process:

```
do {
   wait(full);
   wait(mutex);
    . . .
   /* remove an item from buffer to next_consumed */
    . . .
   signal(mutex);
   signal(empty);
    . . .
   /* consume the item in next_consumed */
    . . .
} while (true);
```

- The following images shows how semaphore helps in synchronization for a ***bounded buffer producer-consumer problem.***
  NOTE: down () & up () are wait () & signal () respectively.

2. **Illustrate the following: How mutual exclusion may be violated If wait() and signal() semaphore operations are not executed atomically. How a binary semaphore can be used to implement mutual exclusion among n processes.**

A wait operation atomically decrements the value associated with a semaphore. If two wait operations are executed on a semaphore when its value is 1, if the two operations are not performed atomically, then it is possible that both operations might proceed to decrement the semaphore value, thereby violating mutual exclusion.
The n processes share a semaphore, mutex, initialized to 1. Each process Pi is organized as follows:

```
do {
    wait(mutex); /* critical section */
    signal(mutex); /* remainder section */
} while (true);
```

3. **Given five memory partitions of 100KB, 500KB, 200KB, 300KB, 600KB (in order). How would the best-fit algorithm place processes of 212 KB, 417 KB, 112 KB, and 426 KB (in order)? Which algorithm (First Fit/Best Fit/Worst Fit) makes the most efficient use of memory?**

Process State:

First fit

212 K is put in 500 K partition.

417 K is put in 600 K partition.

112 K is put in 288 K partition. (New partition 288 K = 500 K - 212 K)

426 K must wait.

Best-fit

212 K is put in 300 K partition.

417 K is put in 500 K partition.

112 K is put in 200 K partition.

426 K is put in 600 K partition.

Worst-fit

212 K is put in 600 K partition.

417 K is put in 500 K partition.

112 K is put in 388 K partition. (600 K - 212 K)

426 K must wait.

In this example Best-fit is the best solution.

4. **Consider a logical address space of 32 pages with 1024 words per page; mapped onto a physical memory of 16 frames. Find the number of bits required to represent a logical address and a physical address. Justify with a neat diagram.**

Address Translation:

- Logical address is generated fig the CPU, this address is also called as virtual address.

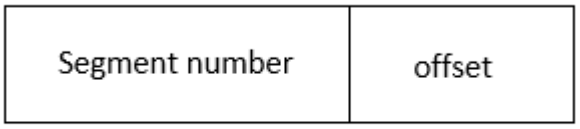
- Main memory address uses physical address and this address is called as read address.
- Set of all logical address generated by a program is called as logical address space.
- Memory management unit (MMU) is responsible for routine address mapping from virtual to physical address.

- They are of 2 types

a) paging.

b) segmentation.

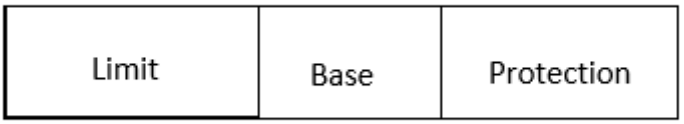
- In paging, operating system divides each incoming program into pages of equal size. the sections of main memory are called as page frames which are of fixed size block.
- Breaking of logical memory into block of same size are called as pages.
- In segmentation, a program data and instruction are divided into block called segment.
- A segment is a logical entity, in a program.
- All segment size may be equal or may not be equal and collection of segment is called as logical address space and each segment in identified fig its name.
- Processor generates logical address. there address consist of segment number and an effect into the segment.
- Segment number is used as an index to segment page table and segment names are normally symbolics name.

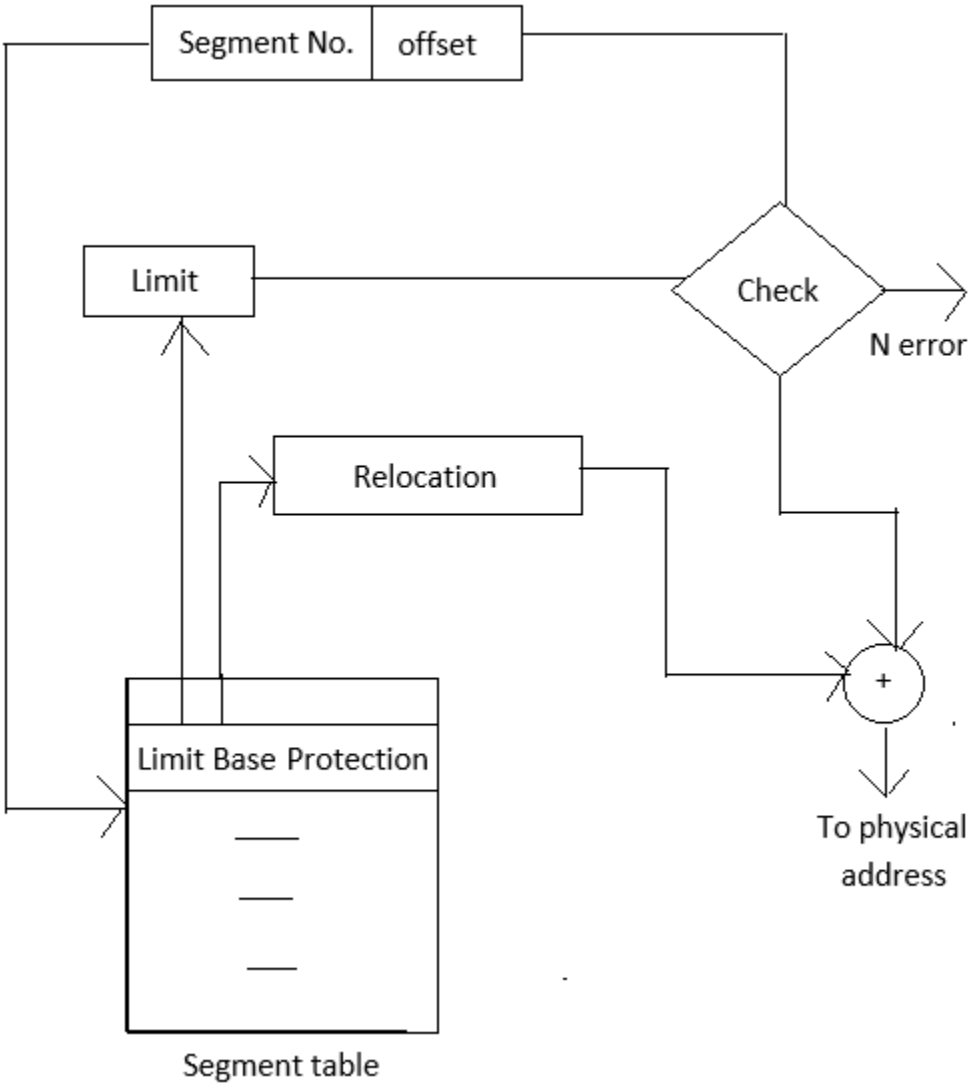


Logical Address

- Operating system maintain a segment table for process. it is usually stored in main memory as a segment that in net to be loaded as long as the process can run.
- Each entry in the segment table has a segment base and segment relocation register for the target segment.
- The segment limit field contains the length of the segment.
- Segment table contains the physical address of the start of segment then add the effort to the phase and generates the physical address.
- If the required reference is not found in one of the segment register, then the error is generated.
- At the same time operating system look up in segment table and loads the new segment descriptor into the register.

Segment descriptor:

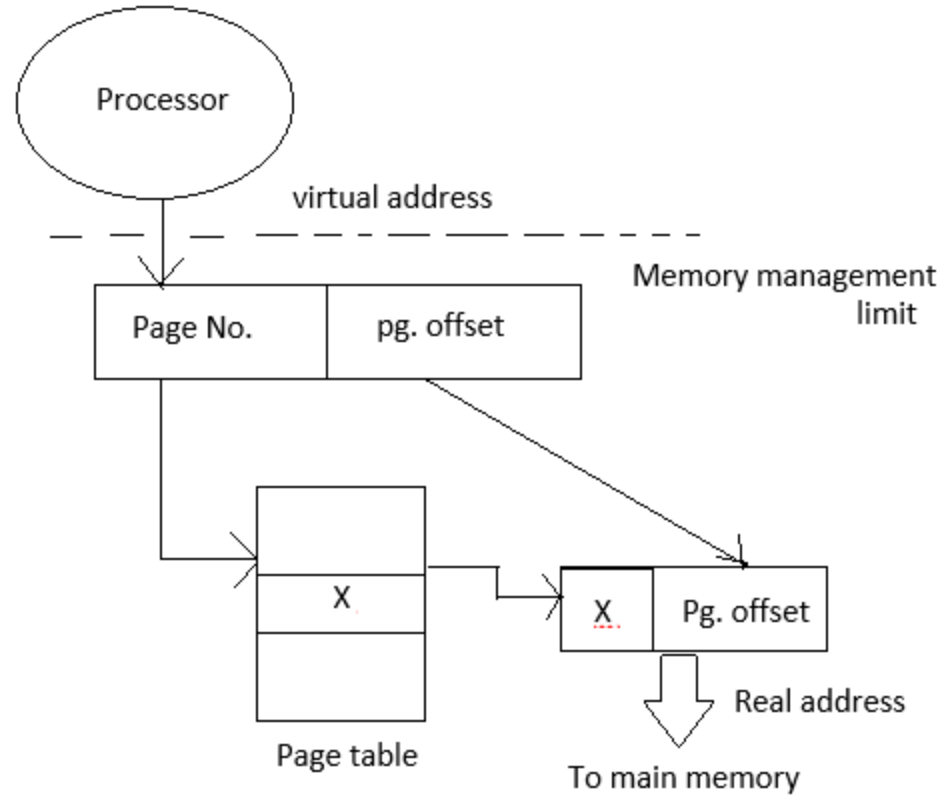




paging:

- Fixed size block in the memory are called frames and breaking of logical memory into a block of same size called as pages.
- Memory manager prepares following things before executing a program.

1. Find out number of pages in program.
2. Find free space in main memory.
3. Loading of all the program pages into memory.



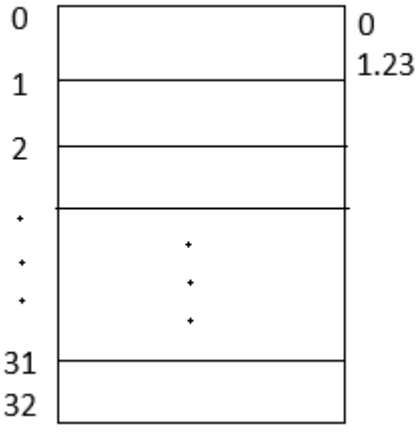
- The processor generates virtual (logical) address and it consist of 2 parts.

1. page number.
2. page offset.

page table contain page number and index frame number, page number is used as an index into page table.

- The physical address is the portion of the primary. memory address allocated to the process and page frame allocated to the process need not be continuous.
- Logical address space.

Total number of page = 32 = $2^5$25
page size = 1024 words = 2^{10}210



$\therefore LA = 2^{10} \times 2^{5} \therefore LA = 2^{10} \times 25$
$= 2^{15} 215$
i.e. Bit required in logical address = 15 bits.

physical memory.

Total no. of frame = 16 = $2^{4} 24$
size of frame = 1034 words = $2^{10} 210$
physical address = $2^{10} \times 2^{4} = 2^{14} 210 \times 24 = 214$
$\therefore\therefore$ bit required in physical address = 14 bits

5. Assuming a 1 KB page size, what are the page numbers and offsets for the following address references (provided as decimal numbers): 1. 2375 2. 19366 3. 30000 4. 256 5. 16385

Address	Page No.	Offset
2375	3	327
19366	19	934
30000	30	304
256	1	256
16385	16	1

6. What are Translation Lookaside Buffers (TLB)? Explain TLB using a simple paging system with a neat diagram.

Translation Lookaside Buffer

- The TLB is associative, high-speed memory.
- The TLB contains only a few of the page-table entries.
- Working:
- When a logical-address is generated by the CPU, its page-number is presented to the TLB.
- If the page-number is found (**TLB hit**), its frame-number is
    - → immediately available and
    - → used to access memory.
- If page-number is not in TLB (**TLB miss**), a memory-reference to page table must be made.
- The obtained frame-number can be used to access memory (Figure 3.19).
- In addition, we add the page-number and frame-number to the TLB, so that they will be found quickly on the next reference.
- If the TLB is already full of entries, the OS must select one for replacement.
- Percentage of times that a particular page-number is found in the TLB is called **hit ratio**.
- Advantage: Search operation is fast.
    - Disadvantage: Hardware is expensive.
- Some TLBs have wired down entries that can't be removed.
- Some TLBs store ASID (address-space identifier) in each entry of the TLB that uniquely
    - → identify each process and
    - → provide address space protection for that process.

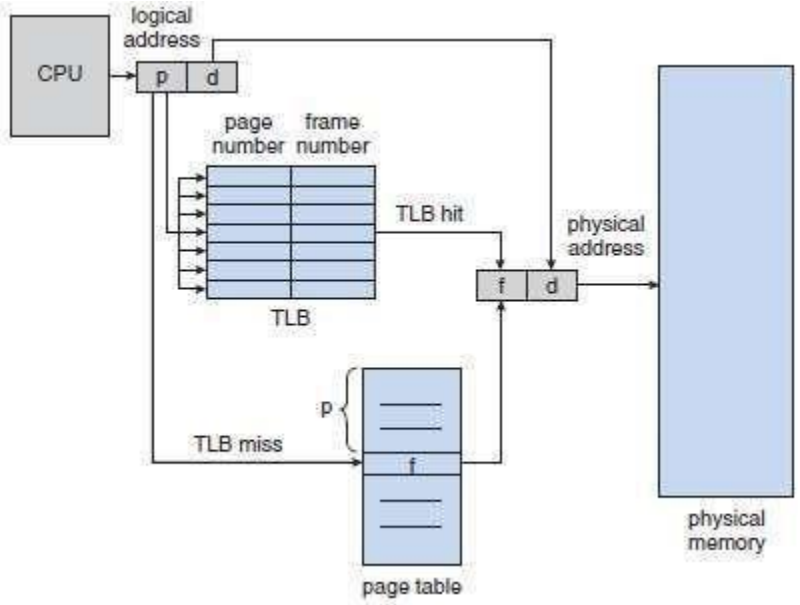


Figure 3.19 Paging hardware with TLB

- For example, suppose that it takes 100 nanoseconds to access main memory, and only 20 nanoseconds to search the TLB. So a TLB hit takes 120 nanoseconds total (20 to find the frame number and then another 100 to go get the data), and a TLB miss takes 220 (20 to search the TLB, 100 to go get the frame number, and then another 100 to go get the data.) So with an 80% TLB hit ratio, the average memory access time would be:
    - 0.80 * 120 + 0.20 * 220 = 140 nanoseconds
    - for a 40% slowdown to get the frame number. A 98% hit rate would yield
        - 0.98 * 120 + 0.02 * 220 = 122 nanoseconds
    - average access time (you should verify this), for a 22% slowdown.

7.

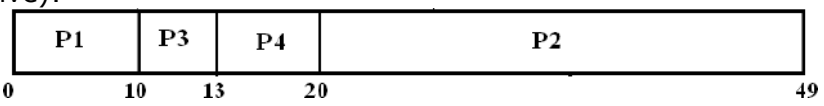
Calculate the average waiting time and average response time by drawing Gantt chart using SRTF and RR(time quantum=10ms)

Processes	Burst Time(ma)	Arrin=val Time(ms)
P1	10	0
P2	29	1
P3	3	2
P4	7	3

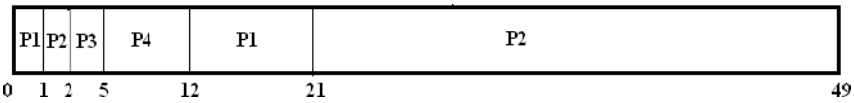
$$\text{Average turn around time} = \frac{\text{Sum of waiting time of individual process}}{\text{Number of processes}}$$

$$\text{Average waiting time} = \frac{\text{Sum of turn around time of individual process}}{\text{Number of processes}}$$

SJF (non-preemptive):

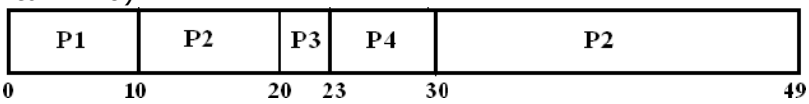


Average waiting time = (0+19+8+10)/4 = 9.25 Average turnaround time = (10+49+13+20)/4 = 19



SJF (preemptive):
Average waiting time = (11+19+0+2)/4 = 8
Average turnaround time = (21+49+5+12)/4 = 21.75

Round Robin (Quantum=10):



Average waiting time =
(0+19+18+20)/4 = 14.25 Average
turnaround time =
(10+49+23+30)/4 = 28