Scheme Of Evaluation
Internal Assessment Test 2 – September 2019

Sub:	Operating system							**Code:**	17EC553
Date**:**	15/10/2019	Duration:	90mins	Max Marks:	50	**Sem:**	V	**Branch:**	ECE

Note: Answer Any Five Questions

Question #	**Description**	**Marks Distribution**	**Total Marks**
1	**Describe address translation using TLB with help of block diagram?** • Block Diagram of address translation using TLB • Explanation physical address calculation	5 M 5M	10 M
2	**Explain the terms** • Internal & External fragmentation • Virtual memory • Logical & Physical address • Page & Page frame • Address translation	2M 2 M 2 M 2 M 2M	10 M
3	**Explain Long, Medium, short- term scheduling with respect to** • Event handling and scheduling • Batch processing, Multiprogramming and Time sharing systems.	5 M 5 M	10 M
4	**For the given set of processes perform FCFS and SRN scheduling. Compare their performance in terms of mean turn-around time and mean weighted turn-around time** Process Id | P1 | P2 | P3 | P4 | P5 Arrival time | 0 | 2 | 3 | 5 | 9 Service Time | 3 | 3 | 2 | 5 | 3 • FCFS scheduling • SRN scheduling	5 M 5 M	10M

5	**Define Contiguous and non-contiguous memory allocations and compare them.** • Explanation of Contiguous memory allocations • Explanation of non-contiguous memory allocations • Comparison	3 M 3 M 4M	10 M
6	**Give the Comparison between paging and segmentation. List the functions of paging hardware.** • Comparison between paging and segmentation. • Functions of paging hardware	5 M 5 M	10 M
7	**Explain FIFO and LRU page replacement policies. Show FIFO and LRU page replacement policies for the page reference string: 0,1,0,2,0,1,2 and time reference string t1, t2, t3, t4, t5, t6, t7 and find out the number of page faults. Given number of page frames=2** • Explaination of FIFO • Explaination of LRU • Problem solving using FIFO • Problem solving using LRU	2M 2M 3M 3M	10M

1. **Describe address translation using TLB with help of block diagram?** **10M**

Scheme- Block Diagram of address translation using TLB 5M

Explanation physical address calculation 5M

Solution- A translation lookaside buffer (TLB) is a memory cache that stores recent translations of virtual memory to physical addresses for faster retrieval.

When a virtual memory address is referenced by a program, the search starts in the CPU. First, instruction caches are checked. If the required memory is not in these very fast caches, the system has to look up the memory's physical address. At this point, TLB is checked for a quick reference to the location in physical memory. When an address is searched in the TLB and not found, the physical memory must be searched with a memory page crawl operation. As virtual memory addresses are translated, values referenced are added to TLB. When a value can be retrieved from TLB, speed is enhanced because the memory address is stored in the TLB on processor. Most processors include TLBs to increase the speed of virtual memory operations through the inherent latency-reducing proximity as well as the high-running frequencies of current CPU's.

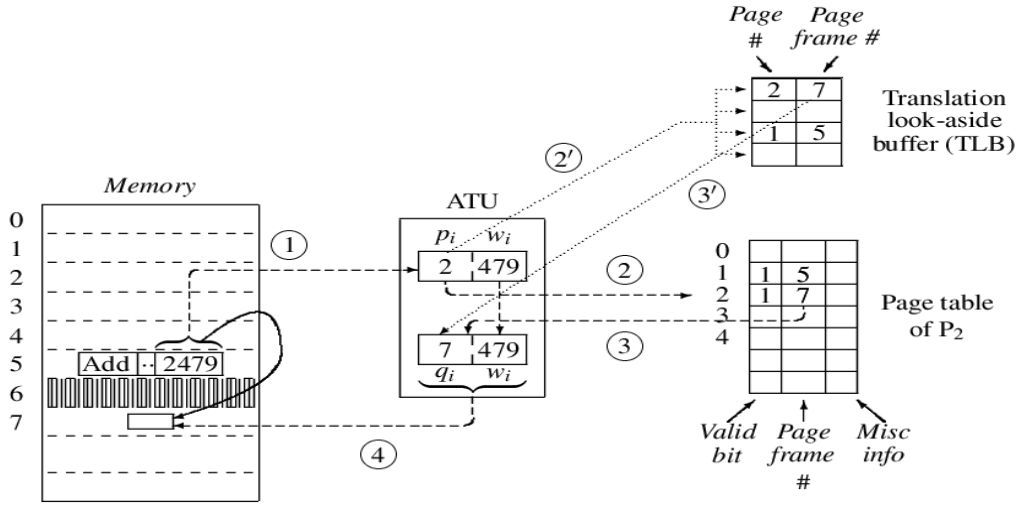


Fig-Address translation using TLB

- The MMU uses **a translation look-aside buffer (TLB)** to speed up address translation
  - The TLB contains entries of the form (page #, frame #) for recently referenced pages
  - The TLB access time is much smaller than the memory access time

  A hit in the TLB eliminates one memory access to lookup the page table entry of a page

  MMU first searches the TLB for page #, t he page table is looked up if the TLB search fails.

2. **Explain the terms**

- **Internal & External fragmentation**
- **Virtual memory**
- **Logical & Physical address**
- **Page & Page frame**
- **Address translation**

Scheme- 2M for Each terms 2x5=10

Solution-

(1) Internal fragmentation is the wasted space within each allocated block because of rounding up from the actual requested allocation to the allocation granularity. External fragmentation is the various free spaced holes that are generated in either your memory or disk space. External fragmented blocks are available for allocation, but may be too small to be of any use.

(2)Virtual memory is the memory that appears to exist as main storage although most of it is supported by data held in secondary storage, transfer between the two being made automatically as required.

(3) Address uniquely identifies a location in the memory. We have two types of addresses that are logical address and physical address. The logical address is a virtual address and can be viewed by the user. The user can't view the physical address directly. The logical address is used like a reference, to access the physical address. The fundamental difference between logical and physical address is that **logical address** is generated by CPU during a program execution whereas, the **physical address** refers to a location in the memory unit.

(4)A **page** is a fixed-length contiguous block of memory, described by a single entry in the page table. It is the smallest unit of data for memory management in a virtual memory operating system. Similarly, a **page frame** is the smallest fixed-length contiguous block of physical memory into which memory pages are mapped by the operating system.

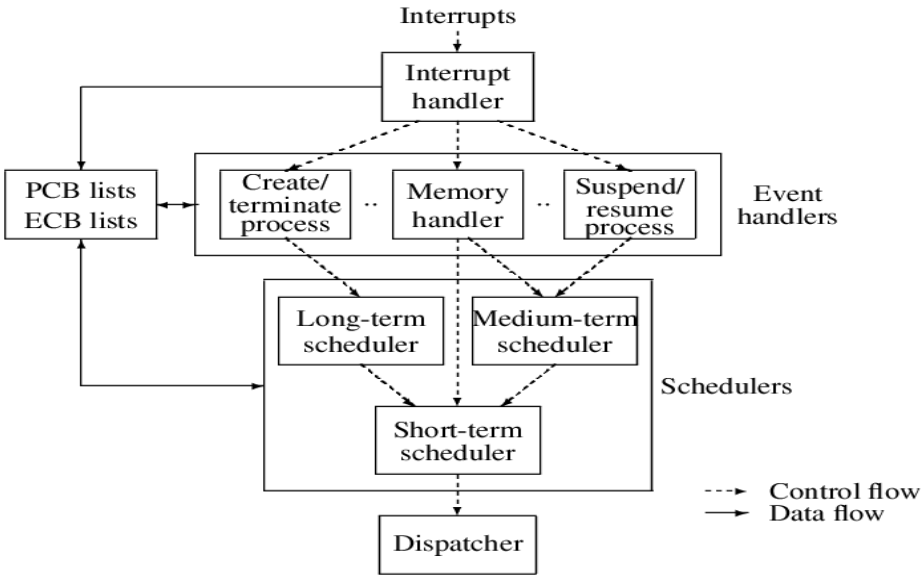
(5).The addresses that appear in programs are the virtual addresses or program addresses. For every memory access, either to fetch an instruction or data, the CPU must translate the virtual address to a real physical address. A virtual memory address can be considered to be composed of two parts: a page number and an offset into the page. The page number determines which page contains the information and the offset specifies which byte within the page. The size of the offset field is the log base 2 of the size of a page.

3. **Explain Long,Medium,short- term scheduling with respect to**
   a)Event handling and scheduling
   b)Batch processing,Multiprogramming and Time sharing systems.

Scheme - Explaination with diagram of Event handling and scheduling 5M
Explaination of scheduling in Batch processing, Multiprogramming and Time sharing systems. 5M

Solution-A single scheduler cannot provide the desired combination of performance and user service, so an OS uses three schedulers

- Long-term scheduler
  - * Decides when to admit an arrived process
    - Uses nature of a process, availability of resources to decide
- Medium-term scheduler
  - * Performs swapping
    - Maintains a sufficient number of processes in memory
- Short-term scheduler
  - * Decides which ready process should operate on the CPU



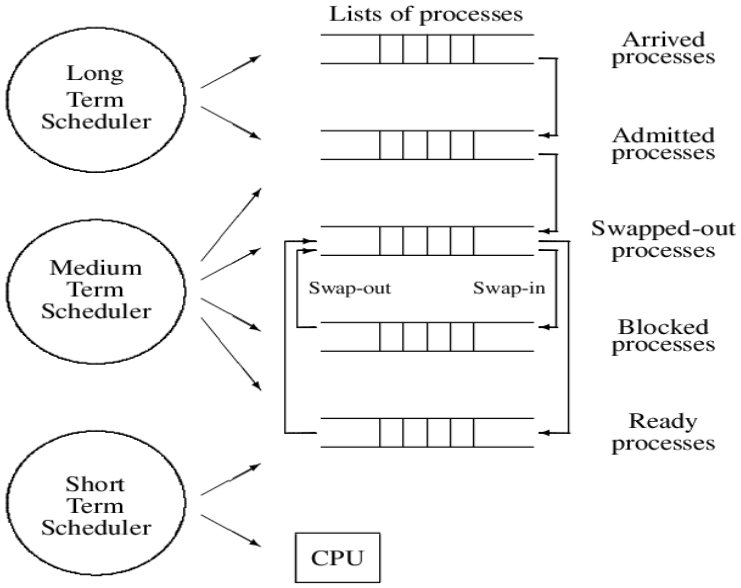
- An event handler passes control to the long- or medium-term scheduler
- These schedulers pass control to the short-term scheduler

Scheduling in a time sharing system

The medium term scheduler swaps *blocked* and *ready* processes and changes their states accordingly

Scheduling in a batch and multiprogramming system

Both batch and multiprogramming system consist of long term and short term schedular but it doesn't have a medium term schedular as it is a non-preemptive scheduling mechanism.



4. For the given set of processes perform FCFS and SRN scheduling.Compare theire performance in terms of mean turn around time and mean weighted turn around time

Process Id	P1	P2	P3	P4	P5
Arrival time	0	2	3	5	9
Service Time	3	3	2	5	3

Scheme - Problem solving 8 marks
Comparision 2 marks

	FCFS					SRN				
	Completed			*Processes*	*Sche-*	*Completed*			*Processes*	*Sche-*
Time	*id*	*ta*	*w*	*in system*	*duled*	*id*	*ta*	*w*	*in system*	*duled*
0	–	–	–	$\{P_1\}$	P_1	–	–	–	$\{P_1\}$	P_1
3	P_1	3	1.00	$\{P_2, P_3\}$	P_2	P_1	3	1.00	$\{P_2, P_3\}$	P_3
5						P_3	2	1.00	$\{P_2, P_4\}$	P_2
6	P_2	4	1.33	$\{P_3, P_4\}$	P_3					
8	P_3	5	2.50	$\{P_4\}$	P_4	P_2	6	2.00	$\{P_4\}$	P_4
13	P_4	8	1.60	$\{P_5\}$	P_5	P_4	8	1.60	$\{P_5\}$	P_5
16	P_5	7	2.33	$\{\}$	–	P_5	7	2.33	$\{\}$	–

$\overline{ta} = 5.40$ seconds
$\overline{w} = 1.75$

$\overline{ta} = 5.20$ seconds
$\overline{w} = 1.59$

Table shows sequence of decisions made by the scheduling policies

- SRN schedules P_3 ahead of P_2 because it is shorter
- Mean turn-around time is shorter by SRN than by FCFS scheduling

5. Define Contiguous and non-contiguous memory allocations and compare them. 5x2=10M

Solution- Contiguous memory allocation

- Allocates a single contiguous memory area to a request
  - Suffers from fragmentation
  - Requires provision to counter fragmentation

Both forms of fragmentation may be present

- Internal fragmentation
  - When a memory area allocated to a process is not fully used
  - Happens when kernel allocates only memory blocks with standard sizes (they are called *partitions*)
- External fragmentation
  - A free memory area is too small to be allocated
  - Compaction can be performed to obtain a single free area
    - It involves relocation of a program
    - it requires provision of a relocation register

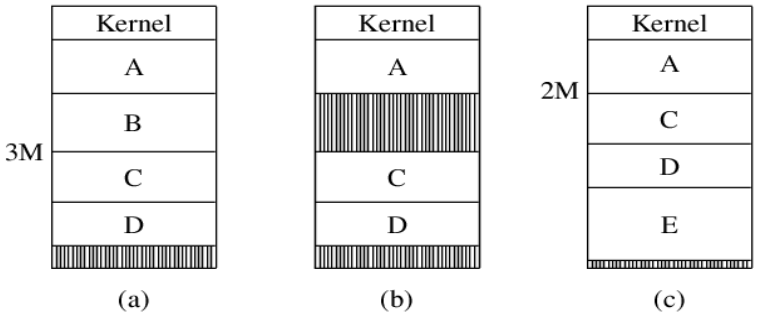


Fig- Contiguous memory allocation

- **Noncontiguous memory allocation**
  - Allocates a set of disjoint memory areas to a request
    * Reduces fragmentation
    * Requires *address translation* during execution of programs to correctly address the allocated memory areas.
- Memory allocation can be done as follows:
  - A process is considered to consist of a set of *components*
    * Each component is allocated a contiguous area of memory that can accommodate it
    * Each operand address in an instruction is considered to be a logical address. It consists of a pair (component #, byte #)
  - The *memory management unit* (MMU) facilitates operation of the process
    * MMU is a hardware unit
    * It converts a (component #, byte #) pair into an absolute memory address

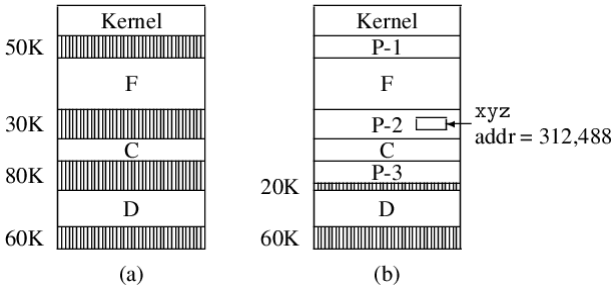


Fig- Noncontiguous memory allocation

Comparison of contiguous and noncontiguous memory allocation

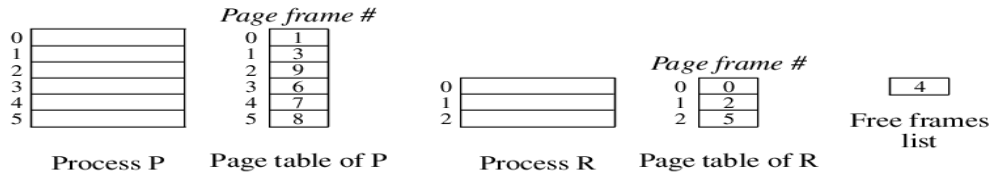
- Contiguous memory allocation
  - No allocation / de-allocation overhead during program execution
  - Internal fragmentation exists in partitioned allocation
  - External fragmentation exists in first/best/next fit allocation
  - Relocation register necessary if a swapped-out process is to be swapped-in in a different memory location
- Noncontiguous memory allocation
  - Address translation is performed during program execution
  - Paging: No external fragmentation; internal fragmentation exists
  - Segmentation: only external fragmentation exists
  - Swapped-in process may be loaded in any part of memory

6. Give the Comparison between paging and segmentation. List the functions of paging hardware.

Solution –Paging

- A process consists of components of a fixed size, called *pages*
  - The page size is a power of 2 and is fixed in the hardware
- Memory is divided into parts called *page frames*, each page frame has the same size as a page
- Kernel functions in paging
  - Allocate memory to all pages of a process
    - Internal fragmentation exists in the last page of the process
    - No external fragmentation
  - Build a *page table* that stores information about addresses of page frames allocated to processes
    - The MMU uses information in the page table to perform address translation

- Address translation in paging
  - A logical address is viewed as a pair (page #, byte #)
    - This splitting is performed using bit-splitting of the logical address since the page size is a power of 2
  - The page # is used to index the page table of the process to obtain the frame # of the page frame where the page has been loaded
  - The corresponding physical address is obtained as follows:
    - The byte # is juxtaposed with the start address of the page frame



- **Segmentation**
  - Each component in a process is a logical unit called a *segment*, e.g., a function, a module, or an object
    - Components can have different sizes
  - The kernel allocates memory to all components and builds a *segment table*
    - Avoids internal fragmentation, but external fragmentation exists
  - Each logical address is a pair of the form (segment id, byte id)
    - Segment id could be a segment name or segment number
    - Byte id could be a byte name or byte number
  - The MMU uses the segment table to perform address translation

segmentation and paging comparison

- Segmentation requires more complicated hardware for address translation.
- Segmentation suffers from external fragmentation and Paging only yield a small internal fragmentation.
- Segmentation is visible to the programmer whereas paging is transparent.
- Segmentation can be viewed as commodity offered to the programmer to organize logically a program into segments and using different kinds of protection (ex: execute-only for code but read-write for data) for this we need to use protection bits in segment table entries

Functions of the paging hardware

The paging hardware performs three functions

- Address translation and generation of page faults
  - MMU contains features to speed up address translation
- Memory protection
  - A process should not be able to access pages of other processes
- Supporting page replacement
  - Collects information about references and modifications of a page
    - Sets the reference bit when a page is referenced
    - Sets the 'modify' bit when a write operation is performed
  - The VM handler uses this information to decide which page to replace when a page fault occurs

7. Explain FIFO and LRU page replacement policies. Show the FIFO and LRU page replacement policies for the page reference string:0,1,0,2,0,1,2 and time reference string t1,t2,t3,t4,t5,t6,t7 and find out the number of page faults.Given number of page frames=2 10M

		Optimal			FIFO			LRU		
Time instant	Page ref	*Valid bit*	*Misc info*	Replacement	*Valid bit*	*Misc info*	Replacement	*Valid bit*	*Misc info*	Replacement
		0 1			0 1	t_1		0 1	t_1	
t_1	0	1 0		–	1 0		–	1 0		–
		2 0			2 0			2 0		
		0 1			0 1	t_1		0 1	t_1	
t_2	1	1 1		–	1 1	t_2	–	1 1	t_2	–
		2 0			2 0			2 0		
		0 1			0 1	t_1		0 1	t_3	
t_3	0	1 1		–	1 1	t_2	–	1 1	t_2	–
		2 0			2 0			2 0		
		0 1			0 0			0 1	t_3	
t_4	2	1 0		1 by 2	1 1	t_2	0 by 2	1 0		1 by 2
		2 1			2 1	t_4		2 1	t_4	
		0 1			0 1	t_5		0 1	t_5	
t_5	0	1 0		–	1 0		1 by 0	1 0		–
		2 1			2 1	t_4		2 1	t_4	
		0 0			0 1	t_5		0 1	t_5	
t_6	1	1 1		0 by 1	1 1	t_6	2 by 1	1 1	t_6	2 by 1
		2 1			2 0			2 0		
		0 0			0 0			0 0		
t_7	2	1 1		–	1 1	t_6	0 by 2	1 1	t_6	0 by 2
		2 1			2 1	t_7		2 1	t_7	