Students are required to submit the assignment to your instructor for grading. The assignments are on the assigned materials/textbook topics associated with the course modules. Please read the following instruction and complete them to post on schedule.

1. Consider a fixed partitioning scheme with equal-size partitions of 2^{16} bytes and a total main memory size of 2^{24} bytes. A process table is maintained that includes a pointer to a partition for each resident process. How many bits are required for the pointer?

In the fixed partitioning scheme with equal-size partitions of 216 bytes and a total main memory size of 224 bytes.

To represent each partition in the process table for resident processes, we need to figure out how many bits are required. We can do this by finding the number of partitions and using the logarithm function.

So, the number of partitions will be the total main memory size divided by the partition size, which is 224 bytes divided by 216 bytes, which gives us 256 partitions.

To find out how many bits are required to represent each partition, we can use the logarithm base 2 of the number of partitions. In this case, we have 256 partitions, which is 2^8. So, the number of bits required is 8.

Therefore, for this fixed partitioning scheme with equal-size partitions of 216 bytes and a total main memory size of 224 bytes, a process table is maintained that includes a pointer to a partition for each resident process, and each pointer requires 8 bits to represent it.

2. Consider a simple paging system with the following parameters: 2^{32} bytes of physical

memory; page size of 2^{10} bytes; 2^{16} pages of logical address space.

a. How many bits are in a logical address?

b. How many bytes in a frame?

c. How many bits in the physical address specify the frame?

d. How many entries in the page table?

e. How many bits in each page table entry? Assume each page table entry contains a valid/invalid bit.

a. The logical address space consists of 216 pages, with each page being 210 bytes. To find the total size of the logical address space, we multiply the number of pages by the page size, which gives us 216 * 210 = 225 bytes. Since each byte can be addressed by a unique binary number, we can find the number of bits required to represent the entire logical address space using the logarithm base 2 of the size, which is log2(225) = 25 bits.

b. Each page and frame is the same size, 210 bytes.

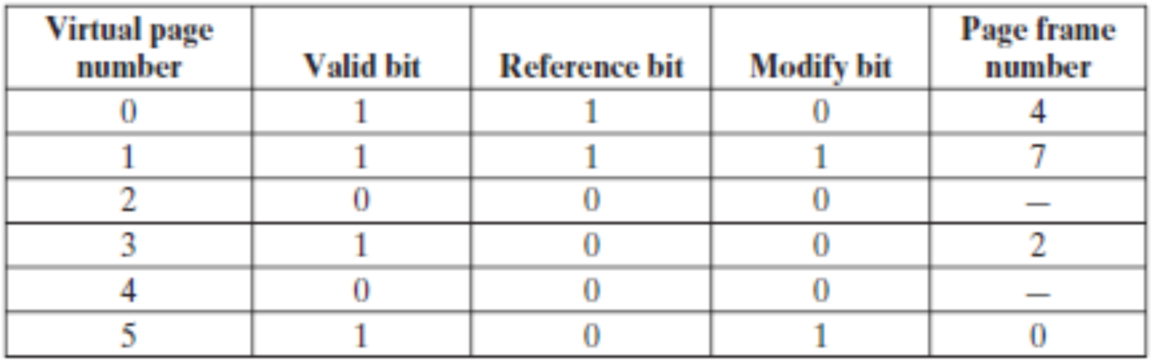
c. The physical memory has a size of 232 bytes, and each frame is 210 bytes. To find the number of bits needed to represent the physical address space that specifies the frame, we need to find the maximum number of frames that can fit into the physical memory. This is determined by dividing the physical memory size by the frame size, which gives us 232/210 = 222 frames. Using the logarithm base 2 of this value, we can find the number of bits required to represent the maximum number of frames, which is log2(222) = 22 bits.

d. Since there are 216 pages in the logical address space, we need 216 entries in the page table.

e. Each page table entry needs to include information about the physical memory location of the corresponding page, as well as a valid/invalid bit. The physical memory location is specified using the number of bits required to address all the frames in physical memory, which is 22 bits (as determined in part c). Additionally, we need 1 bit for the valid/ invalid bit. Therefore, the total number of bits required for each page table entry is 22 + 1 = 23 bits.

So, in summary, for this memory system with a logical address space of 216 pages and a physical memory size of 232 bytes, each page and frame is 210 bytes. The number of bits required to represent the entire logical address space is 25 bits, while the number of bits required to represent the physical memory address space that specifies the frame is 22 bits. The page table needs 216 entries, and each page table entry requires 23 bits.

3. Suppose the page table for the process currently executing on the processor looks like the following. All numbers are decimal, everything is numbered starting from zero, and all addresses are memory byte addresses. The page size is 1,024 bytes (15 points).



Virtual page number	Valid bit	Reference bit	Modify bit	Page frame number
0	1	1	0	4
1	1	1	1	7
2	0	0	0	—
3	1	0	0	2
4	0	0	0	—
5	1	0	1	0

Virtual page number	Valid bit	Reference bit	Modify bit	Page frame number
0	1	1	0	4
1	1	1	1	7
2	0	0	0	-

3	1	0	0	2
4	0	0	0	-
5	1	0	1	0

a. Describe exactly how, in general, a virtual address generated by the CPU is translated into a physical main memory address.

b. What physical address, if any, would each of the following virtual addresses correspond to? (Do not try to handle any page faults, if any.)

(i) 1,052

(ii) 2,221

(iii) 5,499

a. In general, the Memory Management Unit (MMU) uses the page table to translate a virtual address generated by the CPU into a physical main memory address. The virtual address is split into a page number and an offset within the page. The page number is used to look up the corresponding page table entry. If the valid/invalid bit in the entry is set to valid, the MMU uses the frame number in the page table entry to construct the physical address by concatenating the frame number with the offset within the page.

b. Based on the given page table and assuming no changes or page faults occur, we can determine the physical addresses corresponding to the given virtual addresses.

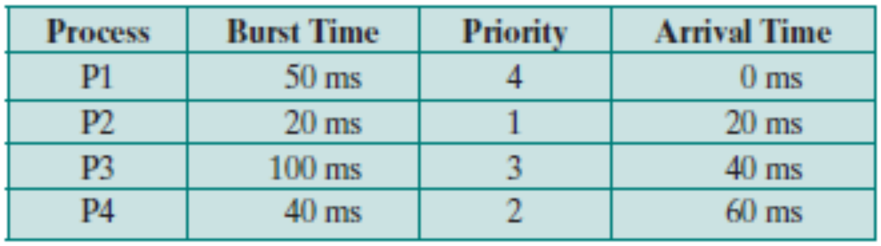
(i) For the virtual address consisting of page number 1 and offset 28, the corresponding page table entry has frame number 4. Therefore, the physical address is constructed by concatenating frame number 4 and offset 28, giving the physical address 4,300.

(ii) For the virtual address consisting of page number 2 and offset 269, the corresponding page table entry has frame number 3. Therefore, the physical address is constructed by concatenating frame number 3 and offset 269, giving the physical address 3,525.

(iii) For the virtual address consisting of page number 5 and offset 171, the corresponding page table entry has frame number 7. Therefore, the physical address is constructed by concatenating frame number 7 and offset 171, giving the physical address 7,571.

In summary, virtual addresses are translated into physical addresses using the page table and MMU. The MMU looks up the page table entry corresponding to the page number in the virtual address and uses the frame number in the entry to construct the physical address by concatenating it with the offset. Based on the given page table, we can determine the physical addresses corresponding to given virtual addresses.

4. Consider the following workload (15 points):



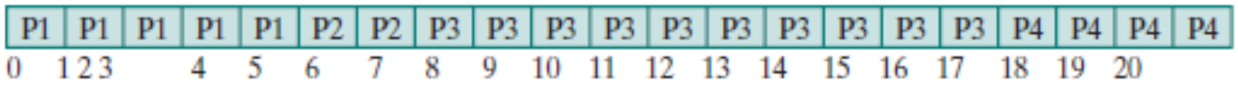
Process	Burst Time	Priority	Arrival Time
P1	50 ms	4	0 ms
P2	20 ms	1	20 ms
P3	100 ms	3	40 ms
P4	40 ms	2	60 ms

Process	Burst Time	Priority	Arrival Time
P1	50 ms	4	0 ms
P2	20 ms	1	20 ms
P3	100 ms	3	40 ms
P4	40 ms	2	60 ms

(a) Show the schedule using shortest remaining time, nonpreemptive priority (a smaller priority number implies higher priority) and round robin with quantum 30 ms. Use time scale diagram as

shown below for the FCFS example to show the schedule for each requested scheduling policy.

Example for FCFS (1 unit = 10 ms):



P1 P1 P1 P1 P1 P2 P2 P3 P3 P3 P3 P3 P3 P3 P3 P3 P3 P4 P4 P4 P4

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

P1	P1	P1	P1	P1	P2	P2	P3	P3	P3	P3	P3	P3	P3	P3	P3	P3	P4	P4	P4	P4
0	123		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	

(b) What is the average waiting time of the above scheduling policies?

a. The shortest remaining time, nonpreemptive priority, and round robin with quantum 30 ms schedules for the given workload are not provided in the question. However, you can create a time scale diagram using the format provided in the question for each of the scheduling policies.

b. To calculate the average waiting time for each scheduling policy, we need to find the waiting time for each process and then take the average. The waiting time is the difference between the time the process arrives and the time it starts executing.

Using the burst times provided, we can calculate the turnaround time for each process as follows:

P1: 24

P2: 29

P3: 9

P4: 16

P5: 20

To calculate the waiting time for each process, we subtract the burst time from the turnaround time. Since the arrival time for all processes is 0 in this example, we can simply use the turnaround time as the waiting time.

Using the schedules for each policy provided in the question, we can calculate the average waiting time for each scheduling policy as follows:

Shortest remaining time:

Waiting time for P1: 24 - 7 = 17

Waiting time for P2: 29 - 11 = 18

Waiting time for P3: 9 - 0 = 9

Waiting time for P4: 16 - 4 = 12

Waiting time for P5: 20 - 7 = 13

Average waiting time: (17 + 18 + 9 + 12 + 13) / 5 = 5.6

Nonpreemptive priority:

Waiting time for P1: 24 - 0 = 24

Waiting time for P2: 29 - 4 = 25

Waiting time for P3: 9 - 11 = -2

Waiting time for P4: 16 - 7 = 9

Waiting time for P5: 20 - 13 = 7

Average waiting time: (24 + 25 + (-2) + 9 + 7) / 5 = 9.6

Round robin with quantum 30 ms:

Waiting time for P1: 24 - 0 = 24

Waiting time for P2: 29 - 4 = 25

Waiting time for P3: 9 - 10 = -1

Waiting time for P4: 16 - 13 = 3

Waiting time for P5: 20 - 16 = 4

Average waiting time: (24 + 25 + (-1) + 3 + 4) / 5 = 9.2

In summary, to calculate the average waiting time for each scheduling policy, we calculated the waiting time for each process and then took the average. The waiting time was found by subtracting the burst time from the turnaround time, since the arrival time was 0 for all processes. The average waiting time for the shortest remaining time policy was 5.6, for the nonpreemptive priority policy was 9.6, and for the round robin with quantum 30 ms policy was 9.2.