# Section 7: Memory Management

## Question 1

Describe both the logical and physical views of a process’s memory.

### Answer

Logical View: The logical view of the process image is the illusion that the process image is installed in a (more or less) contiguous range of memory addresses that starts at address zero. A process’s logical address space is owned by the process and is separate from all other process images. That is, that each process has its own 0-N logical address space.

Physical View: The physical view of the process image is the reality that all process images share physical memory. Each process is installed into a memory partition at unique addresses in physical memory. Partitions can be relocated to new addresses, and with paged memory management can be placed in non-contiguous (non-adjacent) memory regions.

The translation from logical to physical addresses is primarily accomplished by the system’s Memory Management Unit hardware.

## Question 2

Describe the three types of dynamic partition placement algorithms described in class.

### Answer

First Fit: The operating system scans main memory, starting from the top, to find the first unallocated region of memory into which the process will fit regardless of amount of external fragmentation it creates.   
  
Next Fit: The operating system scans main memory, starting from the partition last allocated, to find the next unallocated region of memory into which the process will fit.  
  
Best Fit: The operating system scans main memory to find the unallocated region of memory that is the closest / best fit i.e. is the smallest unallocated region into which the process will fit.

## Question 3

1. Describe the meaning of process image partition relocation?
2. How is relocation of a process image implemented in the Primitive Memory Management Unit (MMU) described in the slides?
3. What is the unit of relocation in paged memory management?
4. Does Paged Memory Management use absolute or relative logical addresses?

### Answer

1. A process image is placed into and executed from a memory partition. A processes’ image can be relocated (moved) from one partition (region of physical memory) to a different partition throughout its lifetime. For example, slide 4 illustrates P1 relocation from one partition to a second.
2. The Primitive MMU implements relocation with two registers: Base and Bounds. The base register maintains the start of the process’s partition and the bounds maintains the partition’s size. A logical relative address is translated into a physical address by adding the bounds register to the relative logical address.
3. In Paged Memory Management, the unit of relocation is a Page of the process image.
4. The Paged Memory Management uses absolute logical addressing. Paged Memory Management uses the high order N bits of a logical address as an offset into the process’s Page Table to lookup the base address of the physical memory frame assigned to the page. The lower M bits of the logial address provide an offset into the frame to generate the physical address.

## Question 4

What is the purpose of *memory* *overlaying* as described in the text?

### Answer

Overlaying is an outdated technique that allows the execution of programs that are too large to fit into the available physical memory. It allows the programmer to explicitly divide their program into multiple executable modules and then specify which of the modules are moved into memory for execution. Newly loaded modules will replace (overlay) program modules currently in memory.

For example, a video game has many levels and players leave one level and moves into another. The code, maps, texture images, etc. for the current level can be overlaid (replaced) with the next level’s content allowing for many levels to be loaded and played in the same physical memory.

## Question 5

1. Describe the purpose and contents of the page table as described in this section.
2. How many page tables are needed to maintain N processes?

### Answer

1. The page table maintains a number of page table entries each of which maintains process’s mapping from logical page numbers to physical-memory frame numbers.
2. Each process maintains its own page table.

## Question 6

NOTE: When creating the elearning assessment, be sure that superscripts are used.

Consider a simple paging system with the following parameters:

* 232 bytes of physical memory
* A page size of 210 bytes
* 216 pages of logical address space

Note: In order to reduce the max size of the page tables, this system’s memory management hardware constrains each process’s page table to 216 entries. This has the effect of reducing the size of the logical address range to less than the physical address range.

1. How many bits are needed to specify a logical address?
2. How many bytes (words) in a frame?
3. How many bits of a physical address are used to specify the frame address?
4. How many entries in a process’s page table?
5. How many bits in each page table entry?

### Answer

1. The number of bytes in the logical address space is (216 pages) x (210 bytes/page) = 226 bytes. Therefore, 26 bits are required for the logical address.
2. A frame is the same size as a page, 210 bytes.
3. The number of frames in main memory is (232 bytes of main memory) / (210 bytes/frame) = 222 frames. So 22 bits are needed to specify the frame address.
4. There is one entry for each page in the logical address space. Therefore there are 216 entries.
5. In addition to the present bit, 22 (32-10) bits are needed to specify the frame location in main memory, for a total of 22 bits.

## Question 7

NOTE: When creating the elearning assessment, be sure that superscripts are used.

Note: Intel hardware’s page size is 212 or 4096.

1. What is the largest internal fragment that can occur with paging on Intel hardware?
2. What is the largest external fragment that can occur with paging on Intel hardware?

### Answer

1. The worst case is a page with a single byte resulting in an internal fragment of size (PageSize -1) or 4095 bytes.
2. A trick question. There is no external fragmentation with paging. (Grader: Do not score this second question).

## Question 8

What is the maximum number of pages that can be addressed in Figure 7.12a?

### Answer

The page number is six bits long or 26 for 64 page entries.

## Question 9

This question refers to the buddy system described in Figure 7.7.

Initially the allocation map is:

128K (A) | 64K (C) | 64K free | 256K free | 256K (D) | 256K free

1. What will the allocation map look like when only Process A exits?
2. What will the allocation map look like when only Process C exits?
3. What will the allocation map look like when only Process D exits?

Please provide three allocation maps i.e. one each for 1, 2, & 3.

### Answer

1. 128K free | 64K (C) | 64K free | 256K free | 256K (D) | 256K free
2. 128K (A) | 128K free | 256K free | 256K (D) | 256K free
3. 128K (A) | 64K (C) | 64K free | 256K free | 512K free

The point is that only sibling nodes are consolidated when both are freed.

## Question 10

What is buffer overflow and how is the following code segment susceptible to it?

int main(void) {  
 char buff[50];  
 int age = 2;  
 …  
 gets(&buff);  
 printf(“You Entered %s\n”, buff);  
 …  
}

### Answer

Buffer overflow occurs when a flaw in the code causes an unchecked input operation to overwrite its assigned input buffer to corrupt adjacent data on the stack or in memory.

In the example, the gets() function will copy input from stdin into a buffer passed to the function until a newline is found. Notice that gets is passed the location of a character buffer ‘buff’ with a fixed length of 50. If the data read from stdin exceeds 50 bytes, gets will begin to overwrite the adjacent integer ‘age’ with the data it is copying.

NOTE: The Unix / Linux library function gets(3) was replaced with the function fgets(3) which requires an additional argument: the buffer size to check for and prevent overflow.