# Section 1: Computer System Overview

## Question 1

What are the three regions / utilizations of a program / process’s memory we have discussed in class?

Describe briefly how each region is used.

### Answer

1. Instruction Memory – This is read-only memory that maintains the process’s instructions that are referenced by the PC and executed by the processor.
2. Data (Heap) Memory – This is the memory that is allocated to the process when a process requests (malloc) memory from the operating system.
3. Control Stack – The region of memory that maintains the process’s stack. The stack is used to push function arguments and pop return values from a function call.

## Question 2

1. Describe the two components of the Generic Instruction Format presented in the book and slides.
2. What is the purpose of the PC register?
3. What is the purpose of the IR register?
4. What is the purpose of the SP register?

### Answer

1. The generic instruction format included both the opcode and address regions. The OPCODE provides bits that describe the instruction / operation the process is to perform. The ADDRESS provides the address of the memory word of the instruction is to operate on.
2. The PC is the Program Counter register and contains the address of the instruction being executed.
3. The IR is the Instruction Register and contains the currently executing instruction (OP Code).
4. The SP is the Stack Pointer register and contains the address of the top of the stack.

## Question 3

1. Describe the two stages of the processor’s Instruction Execution Cycle.
2. Describe how interrupt processing is integrated into the execution cycle. See Figure 1.7.

### Answer

1. **Fetch Stage** – The instruction referenced by the Program Counter is read from instruction memory and placed into the processor’s instruction register (IR).   
     
   **Execution Stage** – The instruction (op code) in the IR is executed by the processor according to the logic hardwired into the processor’s circuitry / microcode.
2. Before the next program instruction is executed, the processor hardware checks if the interrupt signal (IRQ) has been raised. If so, processor saves the processor’s (program’s) state and the next instruction fetched and executed will be the first instruction of the interrupt handling code.

## Question 4

1. What is the definition of ‘asynchronous events’ as described in the slides and book?
2. Describe 3 examples of asynchronous events generated by computer system hardware.

### Answer

1. An asynchronous event is an event that occurs independently of the main program flow. It is an event that can occur at any moment i.e. the processor has no means of predicting when these events will occur. Interrupts are the mechanism used in hardware design to ‘notify’ the processor of the occurrence of an asynchronous event that requires the processor’s attention.
2. Examples event sources include:  
   - The user pressing a keyboard key or moving the mouse.  
   - The arrival of network message.   
   - A hardware controller that requires attention.

(Note to grader: Any three reasonable event sources is acceptable).

## Question 5

1. Describe the meaning of ‘polling a hardware device’.
2. What is the problem with device polling describe in class and book?
3. How does the use of interrupts eliminate the need for polling?

### Answer

1. Polling describes a situation where the processor determines if a hardware device requires attention by checking its status aka polling. Polling requires that the processor must repeatedly query the device to determine when the device requires attention i.e. Are you ready? Are you ready? Etc.
2. Polling wastes processing resources by repeatedly querying the device (possibly millions of times) to determine when the device requires its attention. These are processing resources that would be better spent executing user programs.
3. Devices can generate interrupts at the moment they require processor attention. The processor can instantly respond to these asynchronous events (signals) when they occur i.e. key & button presses, sensors, alarms, and other devices can produce signals that the system must respond to at any time.

## Question 6

1. In terms of program execution, why do we call this mechanism an “Interrupt”?
2. What do we call the instructions executed in response to an interrupt signal?
3. Where is the state of the interrupted (currently executing) program stored?

### Answer

1. It is called an interrupt because the occurrence of the event ‘interrupts’ the currently executing program instructions. Because the events are asynchronous, an event can occur at any point in the currently executing program’s execution.
2. The instructions that are executed in response to the interrupt are called the ‘Interrupt Handler’. The operating system provides interrupt handlers for each of the devices installed on the system.
3. When an interrupt signal is detected by the processor, the state of the currently executing program is saved (pushed) onto the control stack where it later retrieved to restore the execution of the interrupted program when the interrupt handler has completed.

## Question 7

Describe the “Locality of Reference” principle and provide the two examples discussed in class.

How does LOR make cache memory effective?

### Answer

The principle states that a program’s execution will tend to remain in distinct regions of memory addresses over time before moving to new regions. The program will tend to access the same instructions or data (heap) memory many times over a period time. For example: 1) If a program execute the instruction at location i, it is likely to execute instruction i+1. 2) A program may iterate over a list or array of data many times accessing both the data and the loop’s instructions.

If the processor hardware can move the referenced blocks of memory from main memory into cache memory when the block is first accessed, instructions and/or data will be found in cache (hit) for all the remaining access while in a region and the overall execution of the program / process will be that much faster.

## Question 8

What is the average time in *us* to access a byte in a two level memory if …

* Time to access level 1 is 0.1 us.
* Time to access level 2 is 1.0 us.
* The hit ratio is 96%.

### Answer

0.96 (.1us) + 0.04 (.1 us + 1us) = 0.14us

## Question 9

1. Describe the reason for a cache replacement policy in processor design.
2. What should a cache replacement policy minimize?
3. Describe what “Least Recently Used” means in terms of a replace policy.

### Answer

1. Because there is a very limited amount of cache memory when compared to main memory, the instructions and data being maintained in cache must be replaced with the memory blocks of the most recently executed instructions or accessed data. A replacement policy selects which of the cache lines (blocks) to replace with memory from a new program locality.
2. The replacement policy should minimize the number of cache misses. This is accomplished by attempting to select the cache blocks for replacement that are least like to be assessed again. Or, cache blocks that will be accessed furthest in the future (if at all).
3. LRU refers to the algorithm that determines which cache blocks are selected for replacement. The algorithm selects for replacement the cache blocks that were least recently accessed. The assumption is that the program instructions that caused the memory to be cached are no longer being executed and so its associated memory can be replaced with minimal impact on system performance.

## Question 10

1. Describe the three I/O techniques suggested by the book.
2. Using the slide “DMA Block Diagram” (Slide 48), explain under what circumstances the processor’s access to the system bus will not be hampered / slowed down by a DMA transfer. Hint: Consider the role of Cache memory.

### Answer

1. Programmed I/O, Interrupt-Driven I/O, and Direct Memory Access

Programmed I/O – Where the processor must continuously query (poll) the controller to determine when the controller’s data is ready to be transferred. The processor also takes responsibility for transferring blocks of data from one memory location to another e.g. from a disk controller into the application’s memory.

Interrupt-Driven I/O – Where the processor utilizes interrupts generated by the controller to determine when the controller’s data is ready to be transferred. The processor is still responsible for transferring blocks of data from one memory range to another e.g. from a disk controller into the application’s memory.

Direct Memory Access – When the processor makes use of a DMA controller to execute the transfer of data from one memory location to another.

1. If the instructions or data needed by the processor are located in cache, the processor will not need to access the system bus and both processor and DMA can execute without conflict.