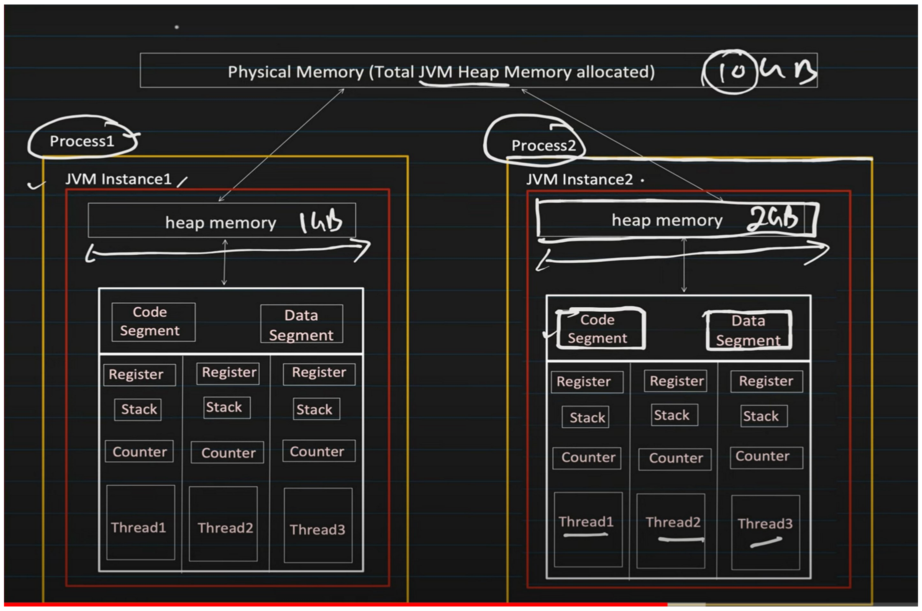
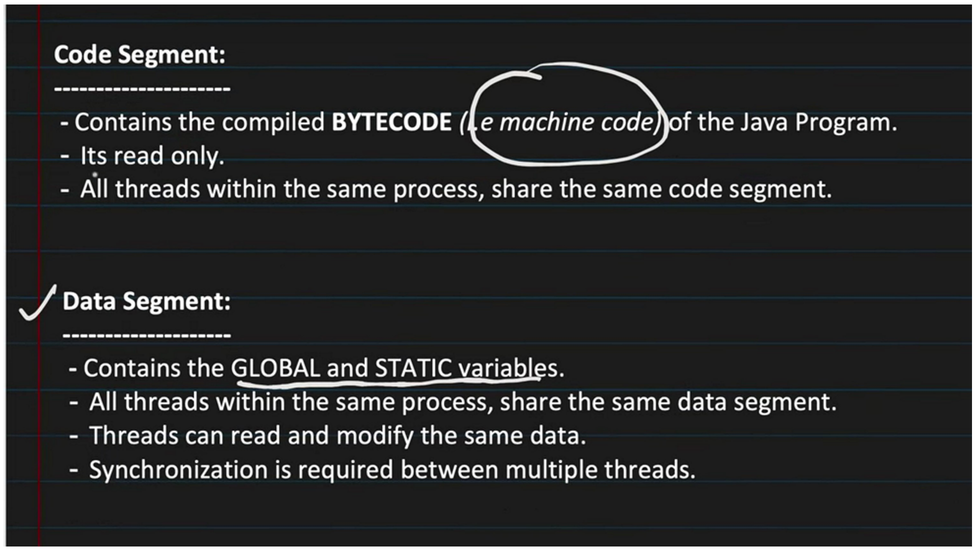
Lesson 1 : Process vs Threads

Friday, April 26, 2024 4:51 PM

Process vs Threads

Process is an instance of a program getting executed. It has its own resources like memory, thread etc that OS allocates when process is created.
Process are independent of each other,never share resources
One process can have multiple threads. Threads are known as lightweight process, they are the smallest sequence of instructions that are executed by CPU independently.
When a process is created, it starts with one thread initially knows as main thread and from that we can create multiple threads to perform task concurrently.
When you run two java programs, two separate JVM instance are created. But total JVM heap memory is fixed and the two share from it.
java -Xms256m -Xmx2g Program1 --- Xms sets initial heap memory while Xmx sets max heap memory for process1, if more memory is required then "Out of Memory" error will occur



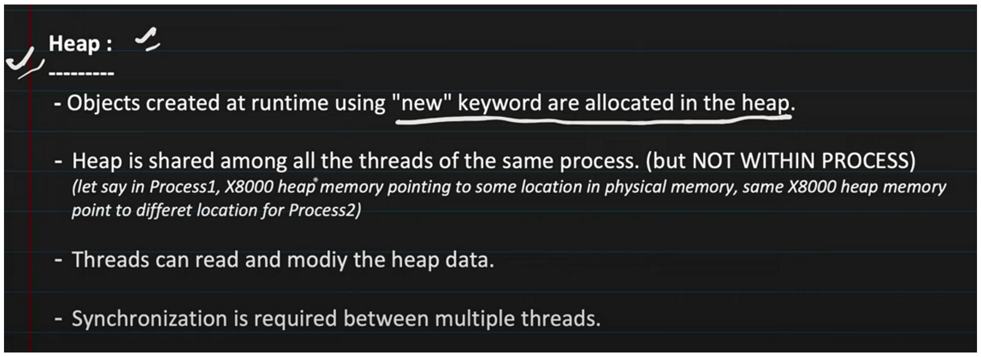


Code Segment:

- Contains the compiled **BYTECODE** *(i.e machine code)* of the Java Program.
- Its read only.
- All threads within the same process, share the same code segment.

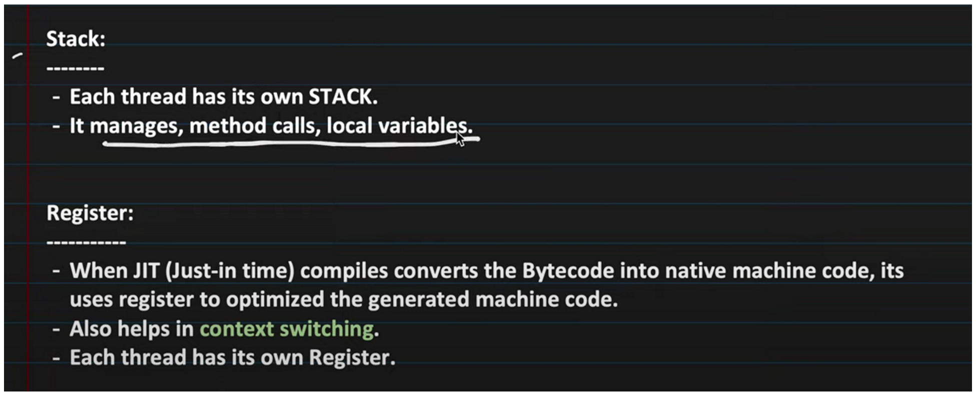
Data Segment:

- Contains the GLOBAL and STATIC variables.
- All threads within the same process, share the same data segment.
- Threads can read and modify the same data.
- Synchronization is required between multiple threads.



Heap :

- Objects created at runtime using "new" keyword are allocated in the heap.
- Heap is shared among all the threads of the same process. (but NOT WITHIN PROCESS)
  (let say in Process1, X8000 heap memory pointing to some location in physical memory, same X8000 heap memory point to differet location for Process2)
- Threads can read and modiy the heap data.
- Synchronization is required between multiple threads.

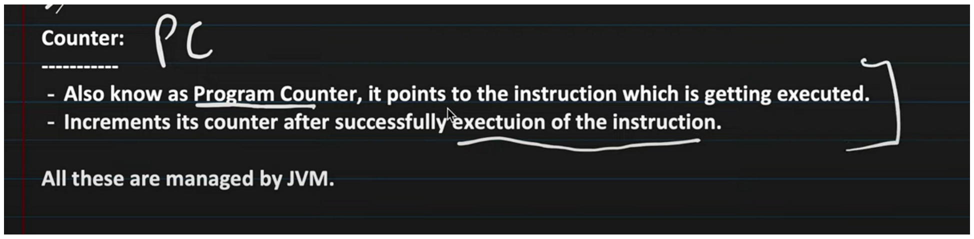


Stack:

- Each thread has its own STACK.
- It manages, method calls, local variables.

Register:

- When JIT (Just-in time) compiles converts the Bytecode into native machine code, its uses register to optimized the generated machine code.
- Also helps in context switching.
- Each thread has its own Register.

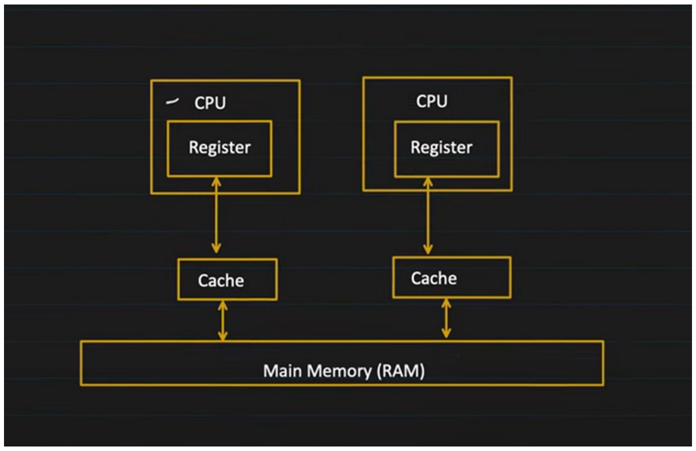


Counter: PC

- Also know as Program Counter, it points to the instruction which is getting executed.
- Increments its counter after successfully exectuion of the instruction.

All these are managed by JVM.

CPU model is somewhat like this, it also has register which is used to load data from thread register for whatever thread is running

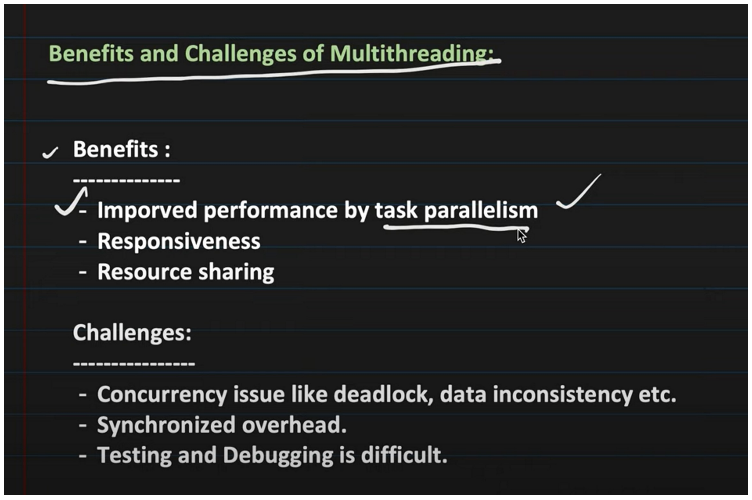


Counter points to what part of code segment does the thread needs to point to.
Register is used to store intermediate results

Once the machine code is waiting to get executed, JVM scheduler (Managed by OS scheduler) now assigns it to CPU.

Context Switching : Suppose it is single core CPU, then if CPU decides to give chance to another thread, it will store the intermediate results of thread1 and store it in the register of thread1 i.e data moves from CPU register to thread register thus freeing CPU register. This is known as context switching.

Definition of Multithreading :
-- Allows a program to perform multiple operations at the same time.
-- Multiple threads share the same resources such as memory space but still can perform task independently



Benefits and Challenges of Multithreading:

✓ **Benefits :**

- **Imporved performance by task parallelism** ✓
- **Responsiveness**
- **Resource sharing**

Challenges:

- **Concurrency issue like deadlock, data inconsistency etc.**
- **Synchronized overhead.**
- **Testing and Debugging is difficult.**