IOS - MEMORY MANAGEMENT

Memory management in iOS was initially non-ARC *AutomaticReferenceCounting*, where we have to retain and release the objects. Now, it supports ARC and we don't have to retain and release the objects. Xcode takes care of the job automatically in compile time.

Memory Management Issues

As per Apple documentation, the two major issues in memory management are −

- Freeing or overwriting data that is still in use. It causes memory corruption and typically results in your application crashing, or worse, corrupted user data.
- Not freeing data that is no longer in use causes memory leaks. When allocated memory is not freed even though it is never going to be used again, it is known as memory leak. Leaks cause your application to use ever-increasing amounts of memory, which in turn may result in poor system performance or *iniOS* your application being terminated.

Memory Management Rules

- We own the objects we create, and we have to subsequently release them when they are no longer needed.
- Use Retain to gain ownership of an object that you did not create. You have to release these objects too when they are not needed.
- Don't release the objects that you don't own.

Handling Memory in ARC

You don't need to use release and retain in ARC. So, all the view controller's objects will be released when the view controller is removed. Similarly, any object's sub-objects will be released when they are released. Note that if other classes have a strong reference to an object of a class, then the whole class won't be released. So, it is recommended to use weak properties for delegates.

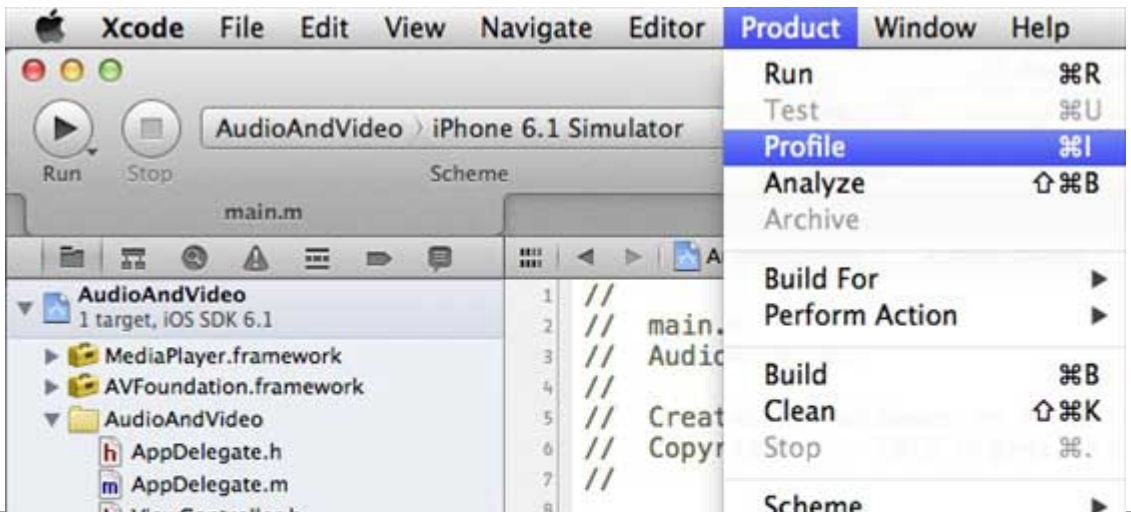
Memory Management Tools

We can analyze the usage of memory with the help of Xcode tool instruments. It includes tools such as Activity Monitor, Allocations, Leaks, Zombies, and so on.

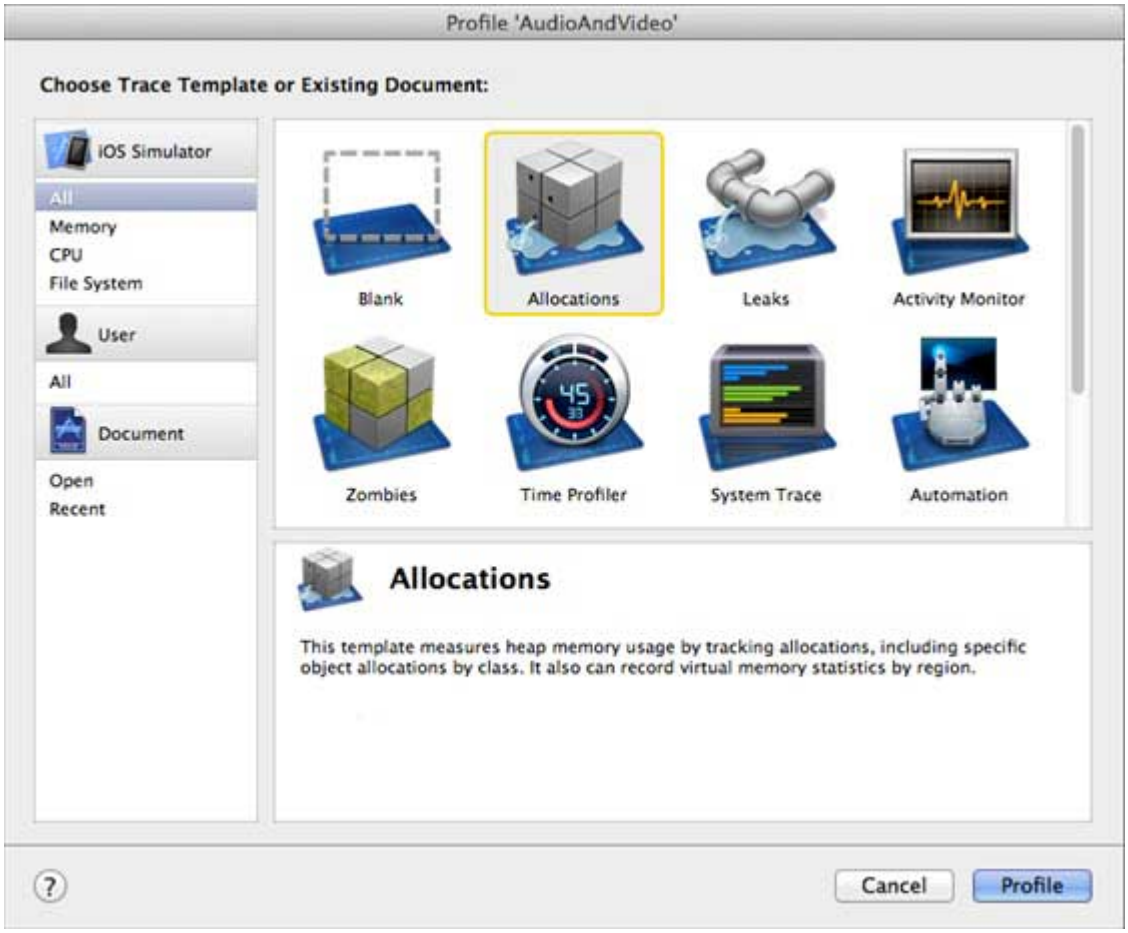
Steps for Analyzing Memory Allocations

Step 1. Open an existing application.

Step 2. Select Product and then Profile as shown below.

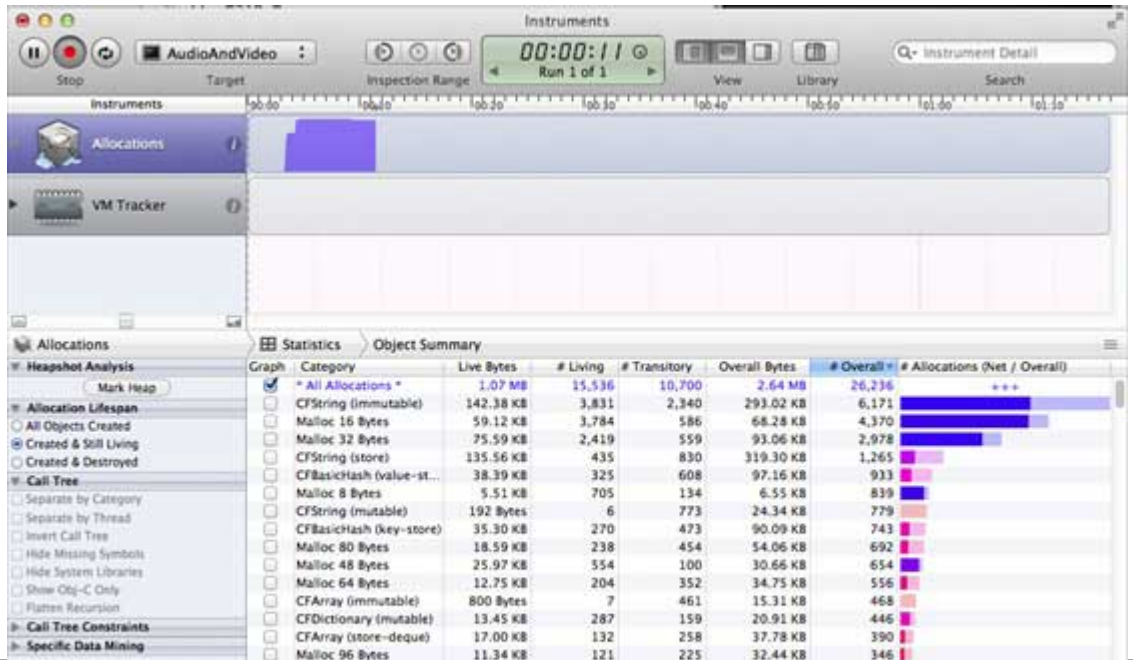


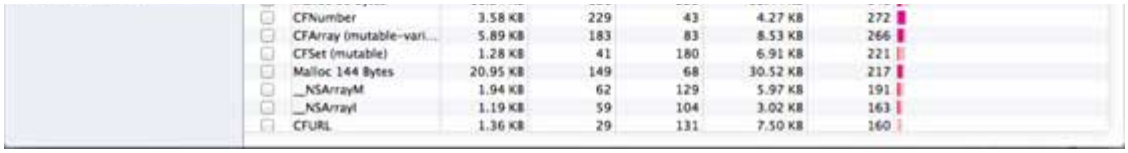
Step 3. Select Allocations in the next screen shown below and select Profile.



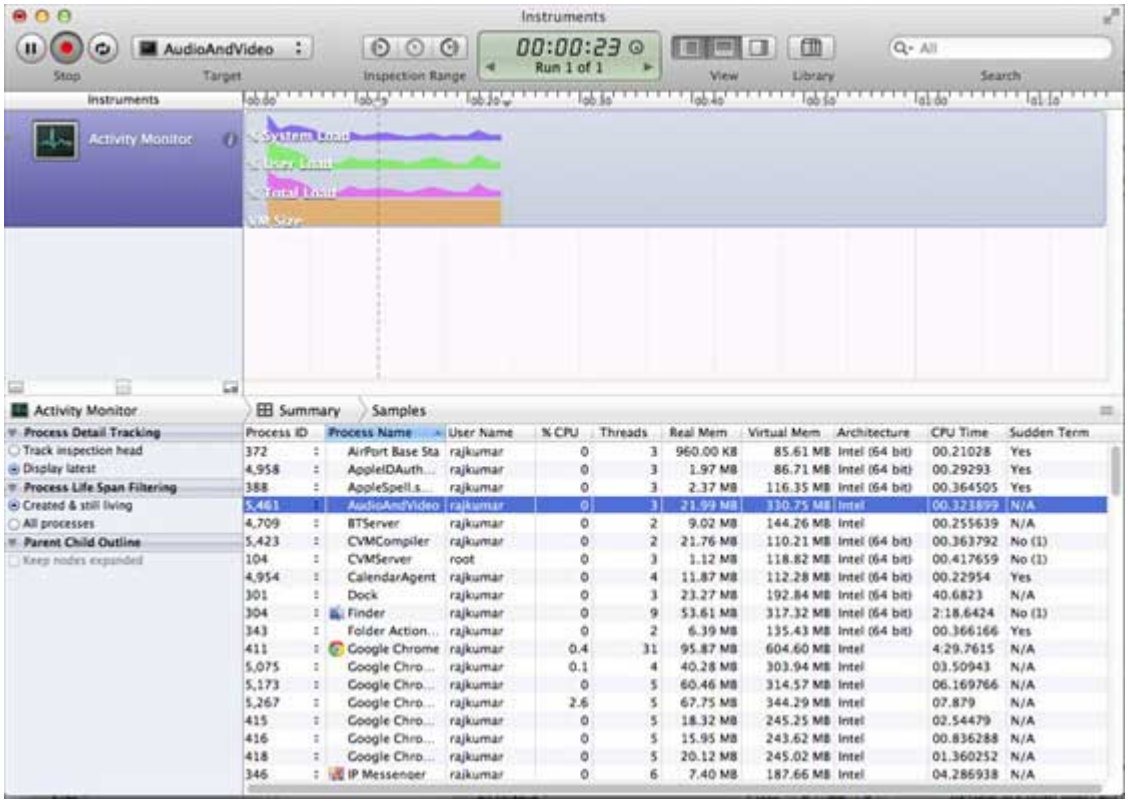
Step 4. We will see the allocation of memory for different objects as shown below.

Step 5. You can switch between view controllers and check whether the memory is released properly.





Step 6. Similarly, instead of Allocations, we can use Activity Monitor to see the overall memory allocated for the application.



Step 7. These tools help us access our memory consumption and locate the places where possible leaks have occurred.