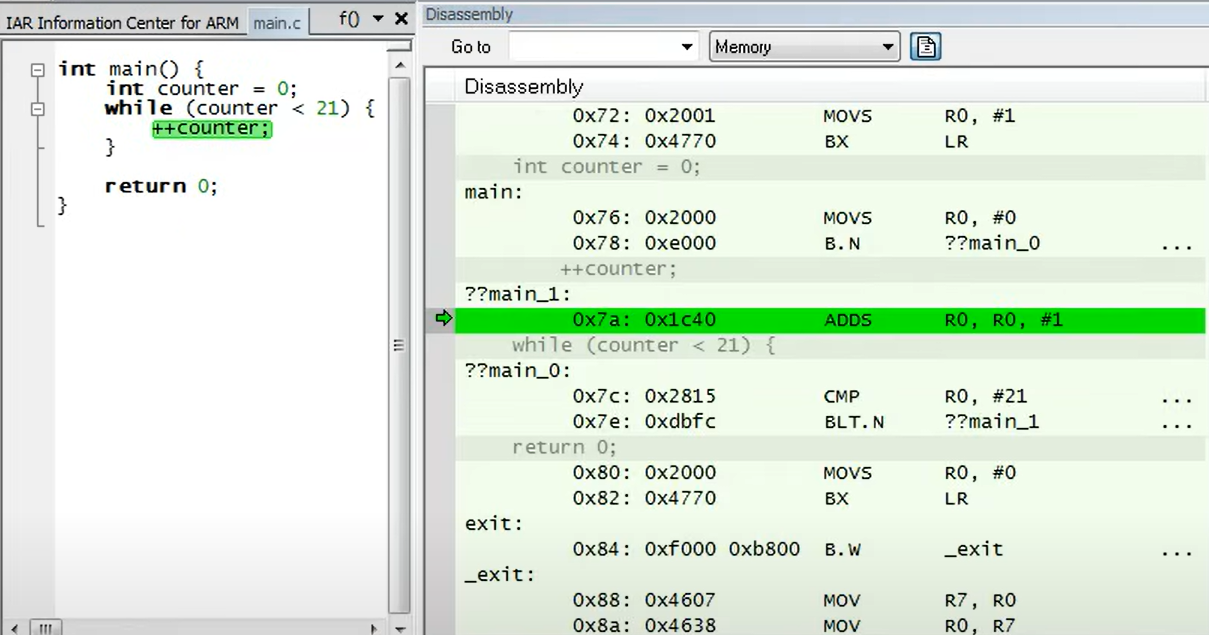
**Embedded C Programming**

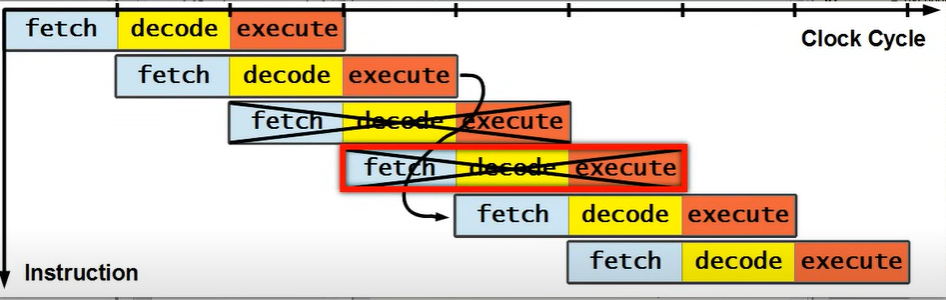
1. How do computers count?  
   - The computer stores data in binary, in the memory.

* The Hexadecimal system is preferred by programmers since it maps exactly to the binary system.
* **Int (Integer):** +ve range = 0-0x7FFFFFFF and -ve range = 0x80000000-0Xffffffff (-1).

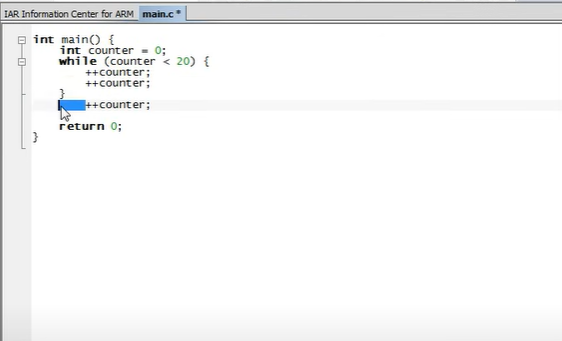
1. How to change the flow of control in your code?

* Loops are used to execute the same code for the required number of iterations.
* But loops carry with them an overhead what is it for that I have attached a SS of the disassembly view of the loop.  
  

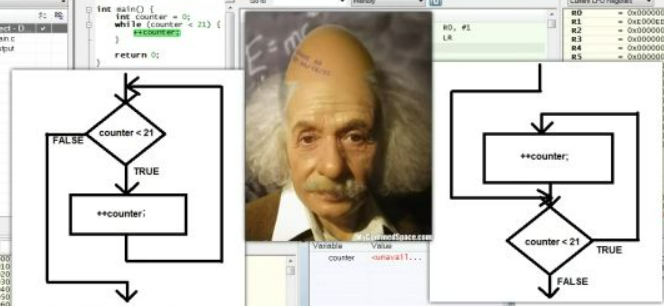
**The** **overhead**: There are branch instructions used, that stall the pipeline for a few cycles since the normal flow is disrupted. As can be seen the pipeline partially processes instructions which are discarded and then restarts execution from the new instructions since a branch instruction was executed.



1. Hence to reduce this overhead in time constrained applications we need to unroll the loop to increase the speed of execution. Like this



1. Compiler Magic: Even though the flow in the while loop is first compare and then increment the compiler uses a different approach as follows.



But both the flows are same then how can I say the compiler is smart?

Well the answer can be seen in the disassembly view check this out.



In the assembly code generated by the compiler it needs just a single branch instruction at the end of the loop, rather if the assembly code was generated as is then there would be two branch instructions BLT.N and BLT.P (to branch to return 0, when 21<21).

1. **How does the branch instruction know which address to branch to**?

Well based on the arm cortexs documentation this information is encoded in the instruction.

So in our example 0xdbfc (BLT.N) 0xd means encoding type T1 (look in figure below), 0xb means the LT condition type, and 0xfc is the offset to be added to current PC to jump to the other address. \*Now offset is a signed quantity hence it means it’s represented in twos complement therefore 0xfc = -4 now subtract -4 from current PC value to get the new PC value.

