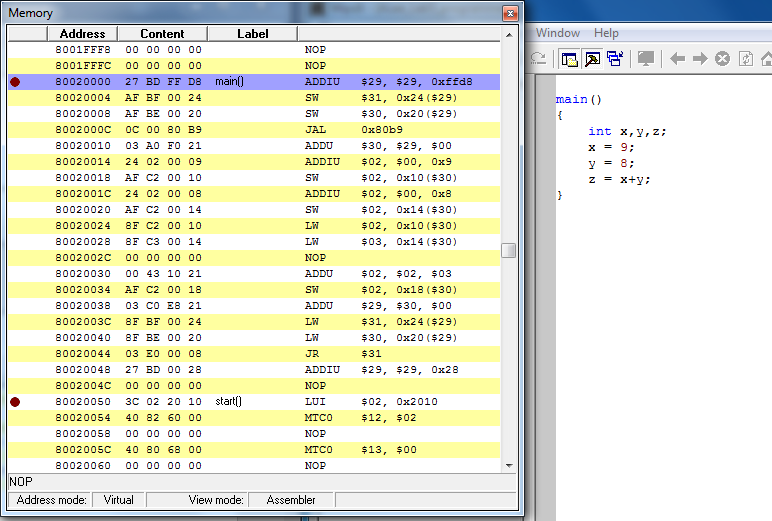
3. Build the executable and upload to the Mips simulator.

4. View the assembly version of the C program.



5. Identify the core assembly instructions in the RAM of the simulator that are the compiled version of your C program.

JAL : Jump and Link -To execute a procedure call within the current 256 MB-aligned region Pg: 149

SW : To store a word to memory Pg: 280

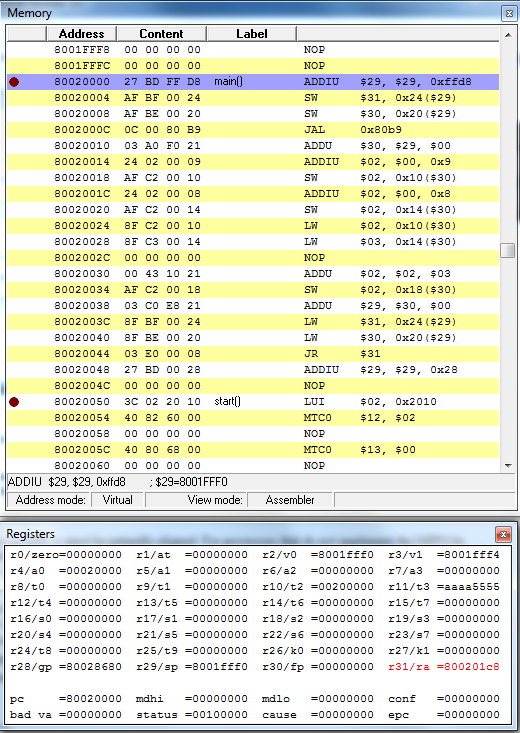
ADDIU $2, $0,9 : Add Immediate Unsigned Word 9 to R0 and store in R2 Pg: 47

SW $2, 0x10($30) store word from R2 to mem are pointed to by R30 offset 10 i.e. loc of X

ADDU : Add Unsigned Word - performs the same arithmetic operation but does not trap on overflow. Pg: 44

LW : Load Word- To load a word from memory as a signed value Pg: 171

JR : Jump register - To execute a branch to an instruction address in a register Pg: 155



Initialisation

Main code operations between JAL and JW

ADDIU $2, $0,9 Add Immediate Unsig Word 9 to R0 and store in R2

SW $2, 0x10($30) store word from R2 to mem are pointed to by R30 offset 10 i.e. loc of X

ADDIU $2, $0,8 Add Immediate Unsig Word 8 to R0 and store in R2

SW $2, 0x14($30) store word from R2 to mem are pointed to by R30 offset 14 i.e. loc of y

Load word from loc 0x10($30) and store into R2

Load word from loc 0x14($30) and store into R3

ADDU R2 and R3 and store in R2

SW $2, 0x18($30) store word from R2 to mem are pointed to by R30 offset 18 i.e. loc of z

6. What is the rest of the code doing there?

Within the main section we have the arithmetic procedures and intermediate procedures for storing and manipulating the values as per the design within the main code.

Beyond the main section we have some more code however this is not relevant to the code we have written and is background instructions required by the computer

7. Set a breakpoint at the beginning of this block of code.

8. Run the program up to the breakpoint.

9. Single step through your section of code and confirm that it produces the correct result.

