COMP2611: Computer Organization

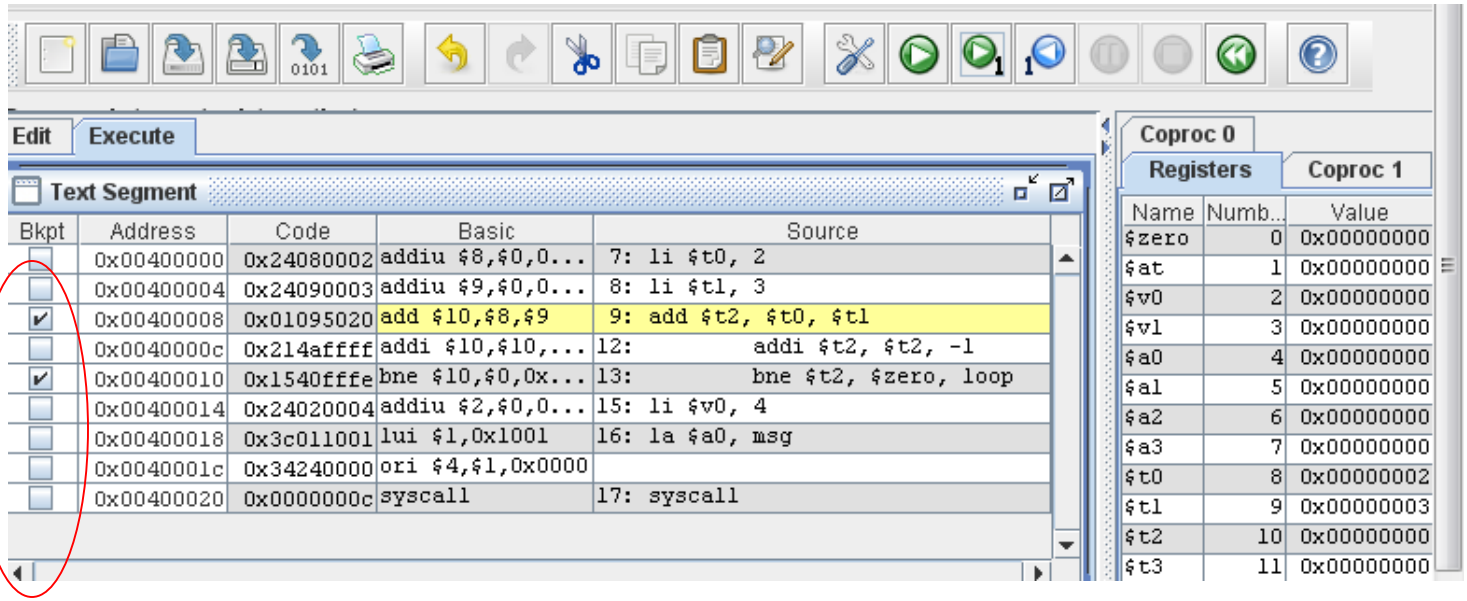
Debugging on Mars

Overview

- You will learn the following in this lab:
  - how to debug a MIPS program using the debugging features in Mars.

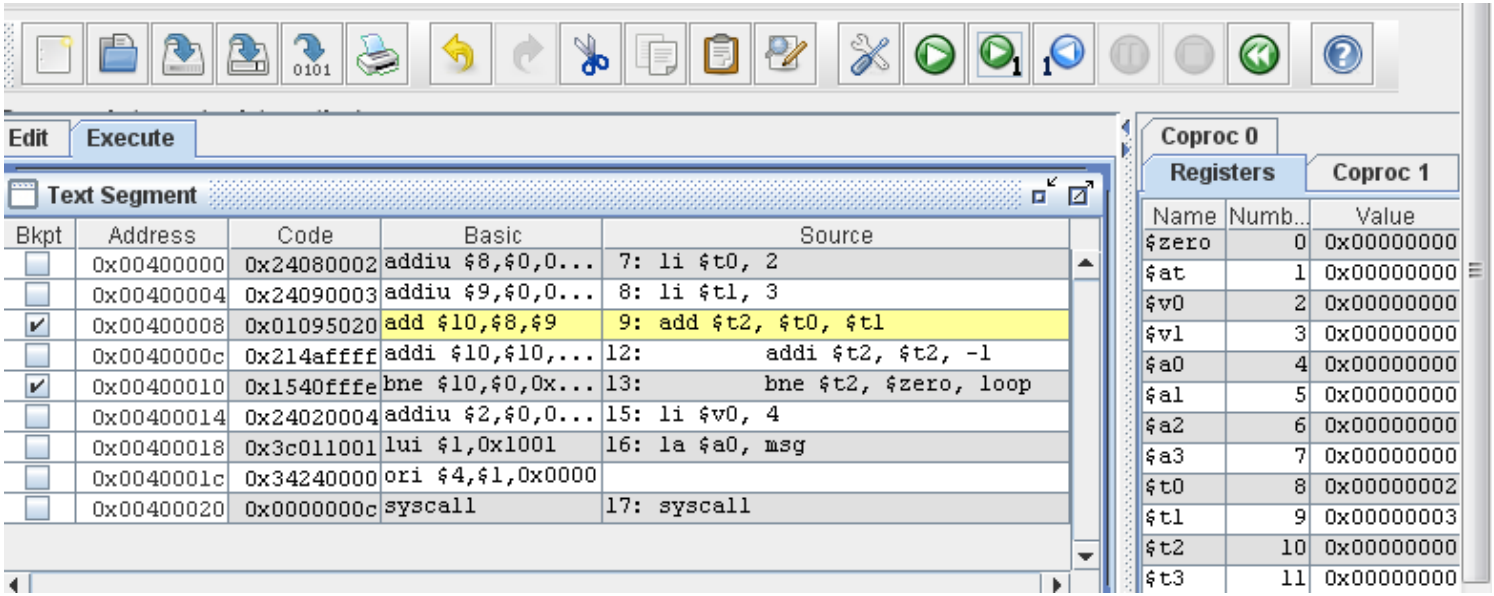
Tracing program execution

- Mars provides many debugging features:
  - Breakpoint -- pauses the execution at an instruction.
    - Click the box on the column "Bkpt" of an instruction to enable a breakpoint there.
    - All the instructions before the breakpoint were executed, and all the subsequent instructions, including the one at the breakpoint, are not executed yet.

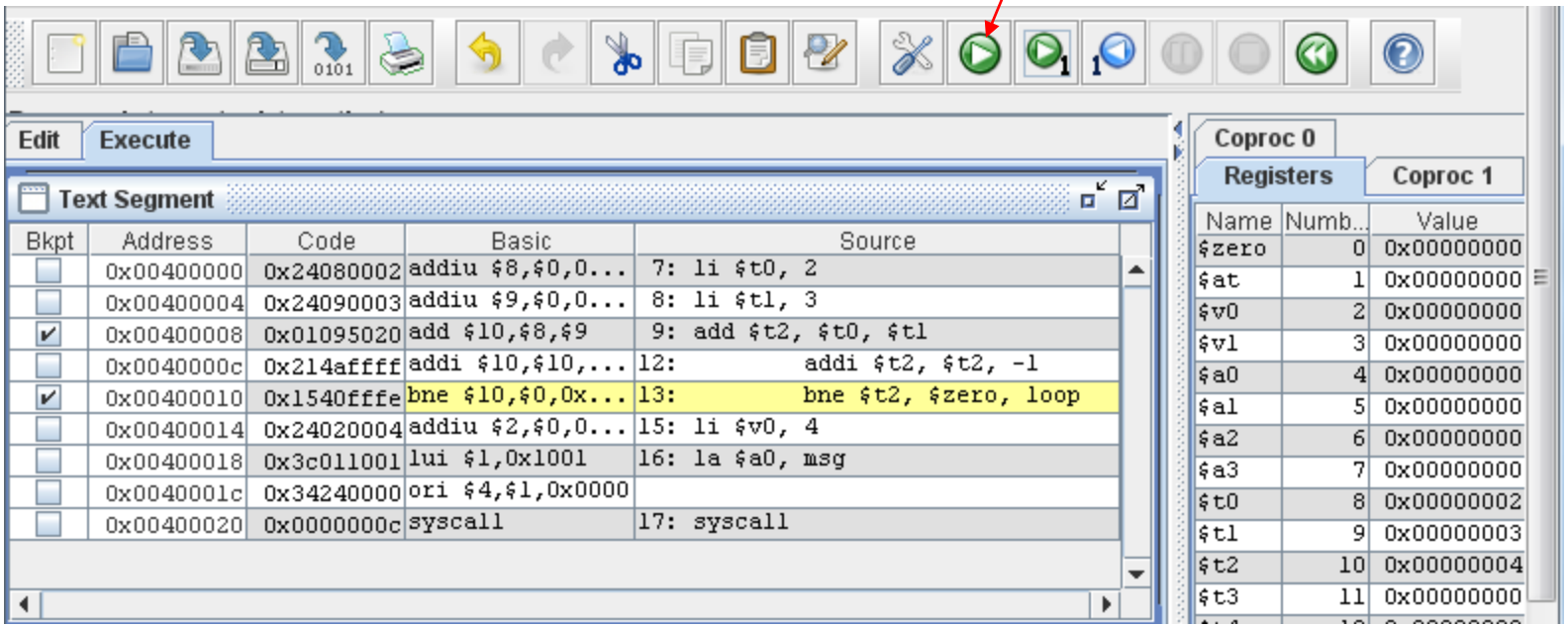


Mars Breakpoint

- Load the example program debug1.s, and try enabling the breakpoints shown below.
- Then start the execution to see how it pauses at the first breakpoint.
- The values of the registers and memory reflect the execution up to (but not including) the instruction at this breakpoint.

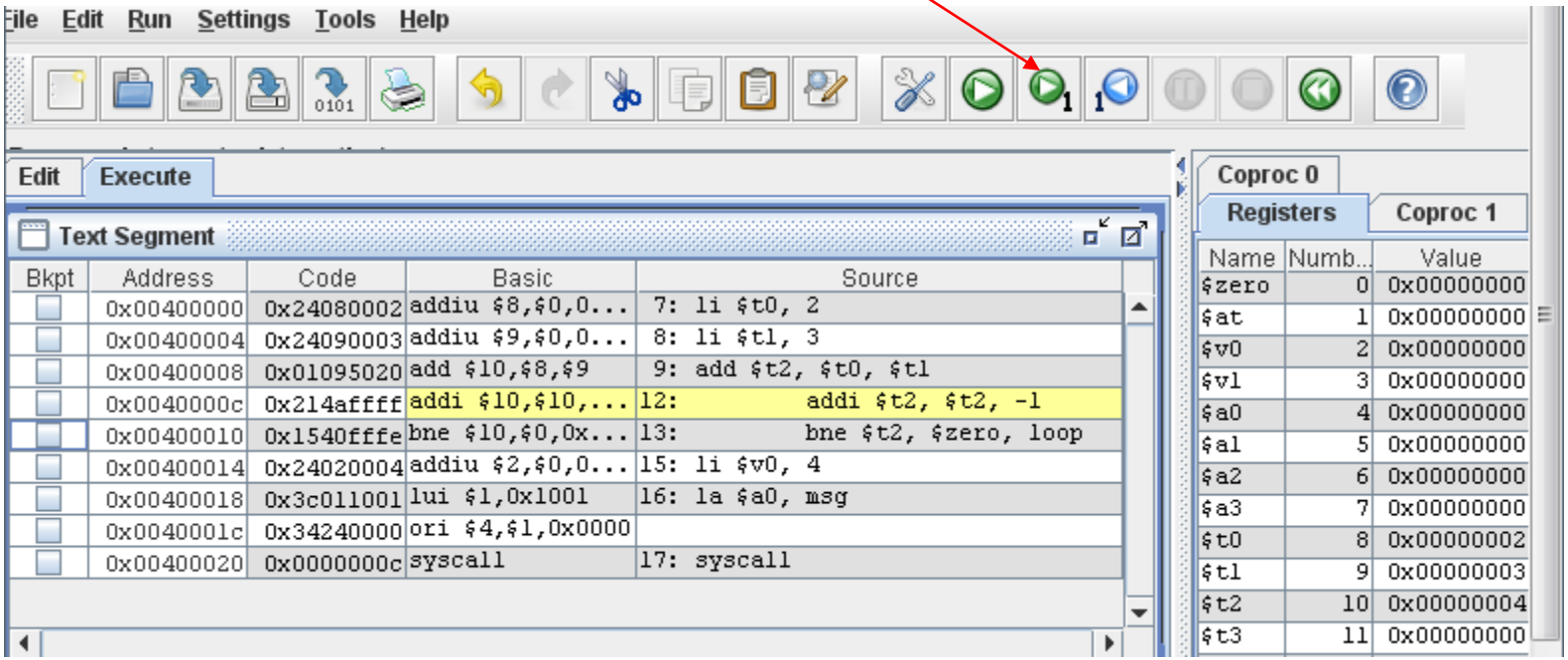


- Start the execution again (click the same button) to see how the execution pauses at the second breakpoint.



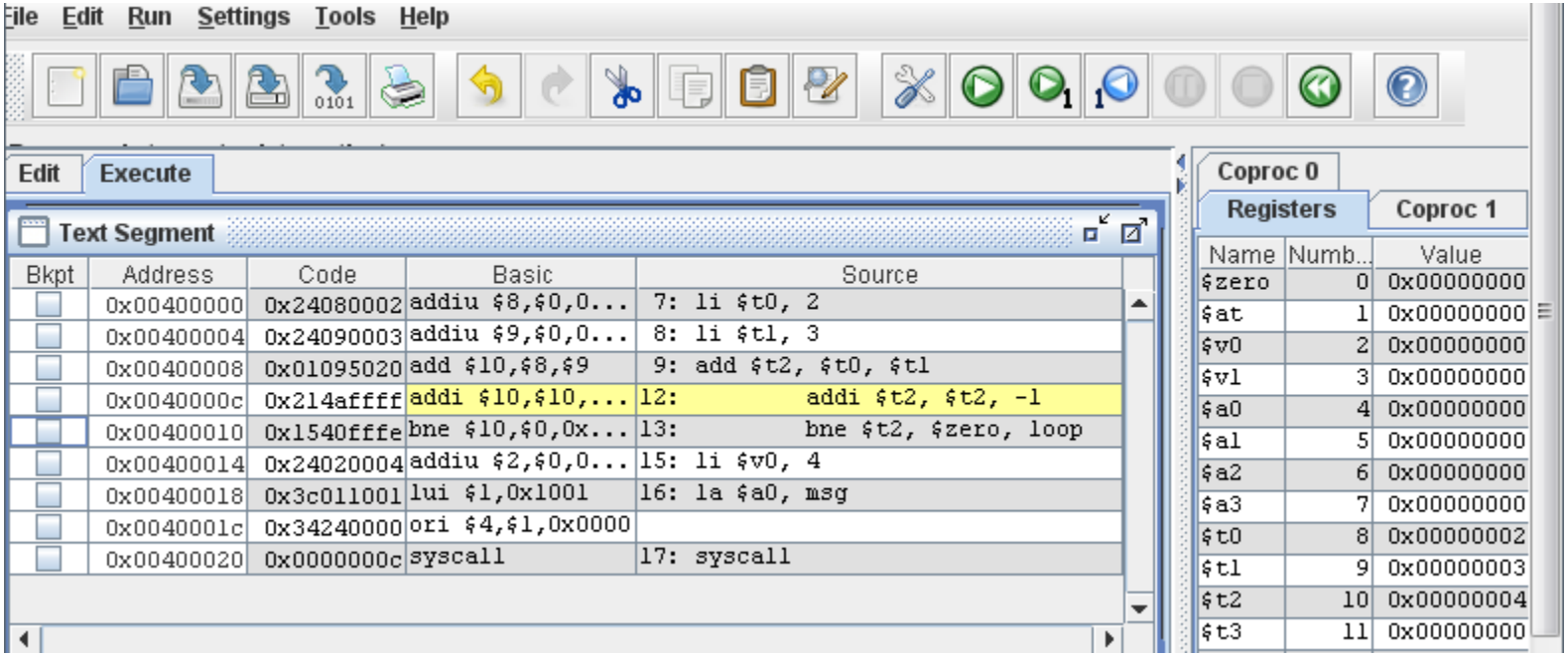
Mars Single Step

- Single Step – executes one instruction and then pauses the execution.
  - Click this **Single Step** button to do a Single Step.
  - It can be used at the very beginning to start executing the program (executing its first instruction) or at any time when the program execution is paused (e.g., by a breakpoint).



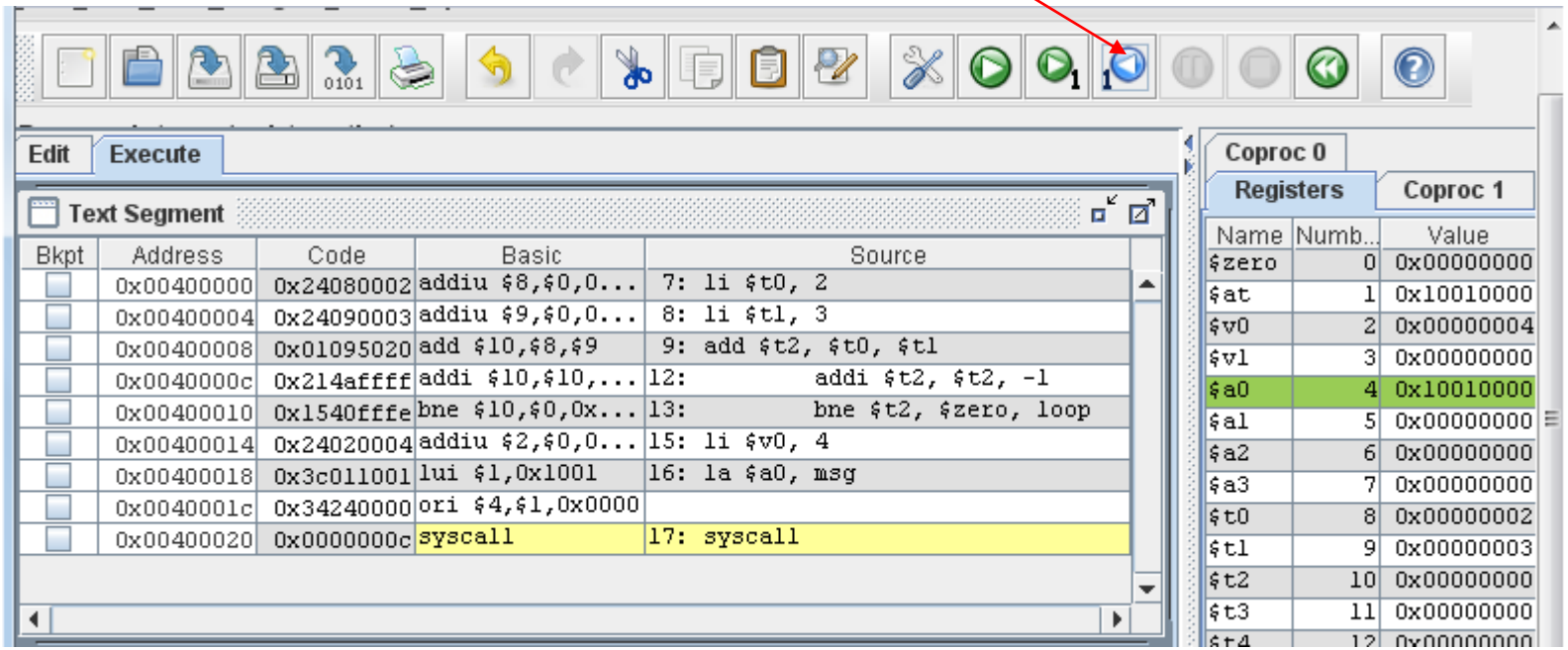
Mars Single Step

- Try executing each instruction in debug1.s using Single Step.
- See how the execution of the loop in debug1.s is traced in this way.
- See how the values of the registers and/or memory are changed to reflect the latest execution.



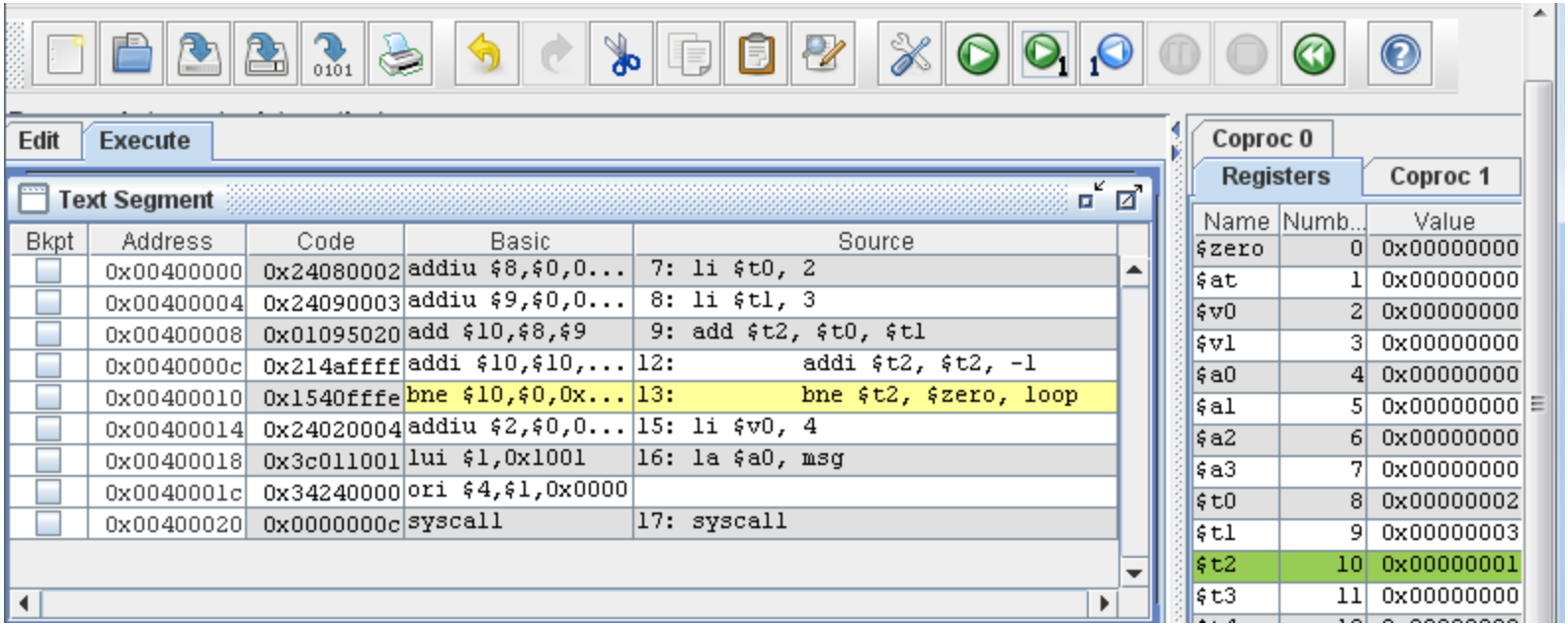
Mars Undo Step

- Undo Step – undo the last instruction (up to a maximum of 2000 instructions by default) and then pauses the execution.
  - Click this **Undo Step** button to do an Undo Step.
  - It can only used when the program execution is paused.
  - It can still be used after the program execution terminated (but before it is reset). Try this after executing debug1.s.



Mars Undo Step

- Try reversing the execution of the loop in debug1.s, too.
- See how the values of the registers and memory are changed back as the execution reverses.

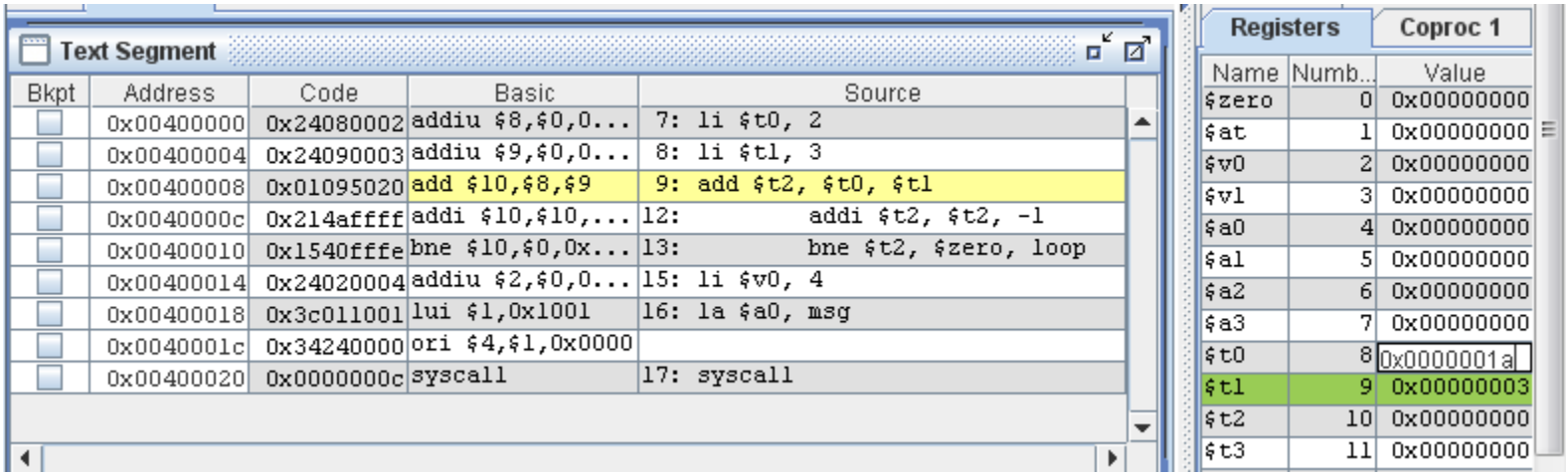


Modifying registers or memory

- When you figure out a possible solution to fix a buggy program, you can modify the program code to try it out.
- You can also just modify the values of the registers or memory (according to the solution) during the (buggy) program execution.
- This lets you get a sense of whether the solution should work before you modify any codes.
- To modify a register or memory,
  - double-click on it on the Registers or Data Segment window.
  - Type the new value in hexadecimal or decimal format.
  - Finally, press the Enter key to apply the new value.
- The modification can only be done before the program execution starts or when it is paused.
- The new value will be applied to all the subsequent executions.

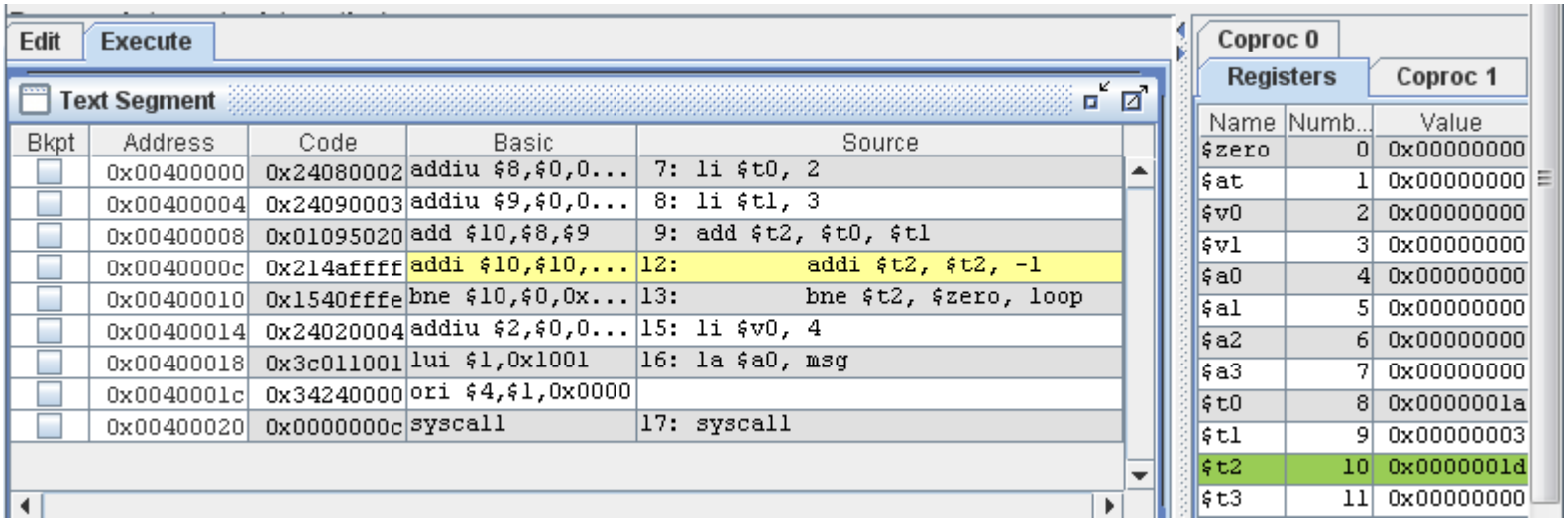
Modifying registers

- Try executing the program debug1.s.
- Then pause it at the instruction in Line 9 and modify the value of the register t0 to 0x0000001a (as shown by the image below).



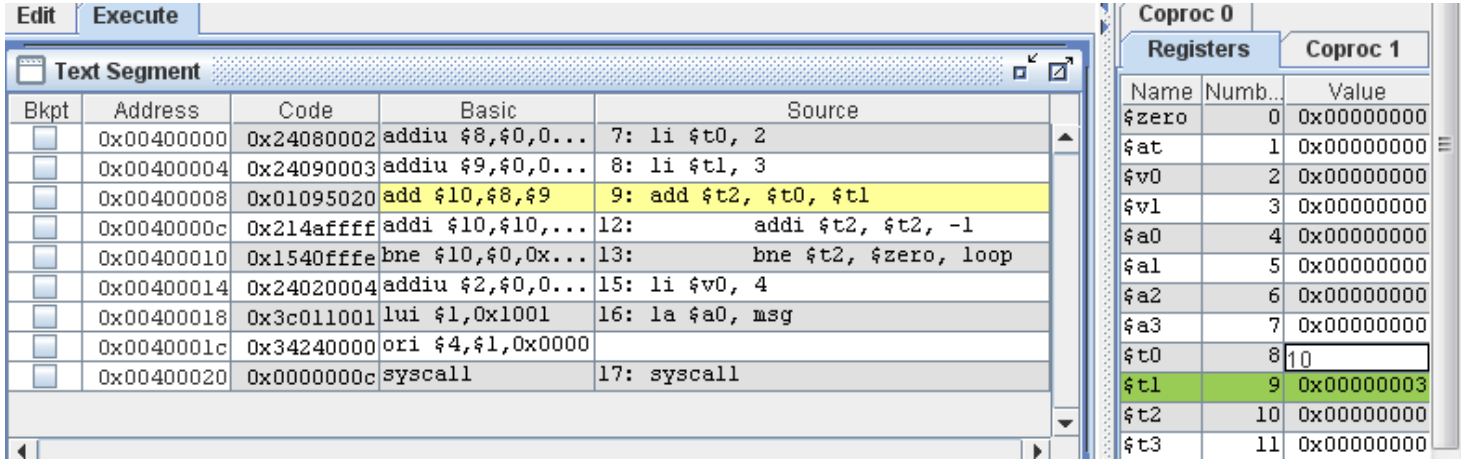
Modifying registers

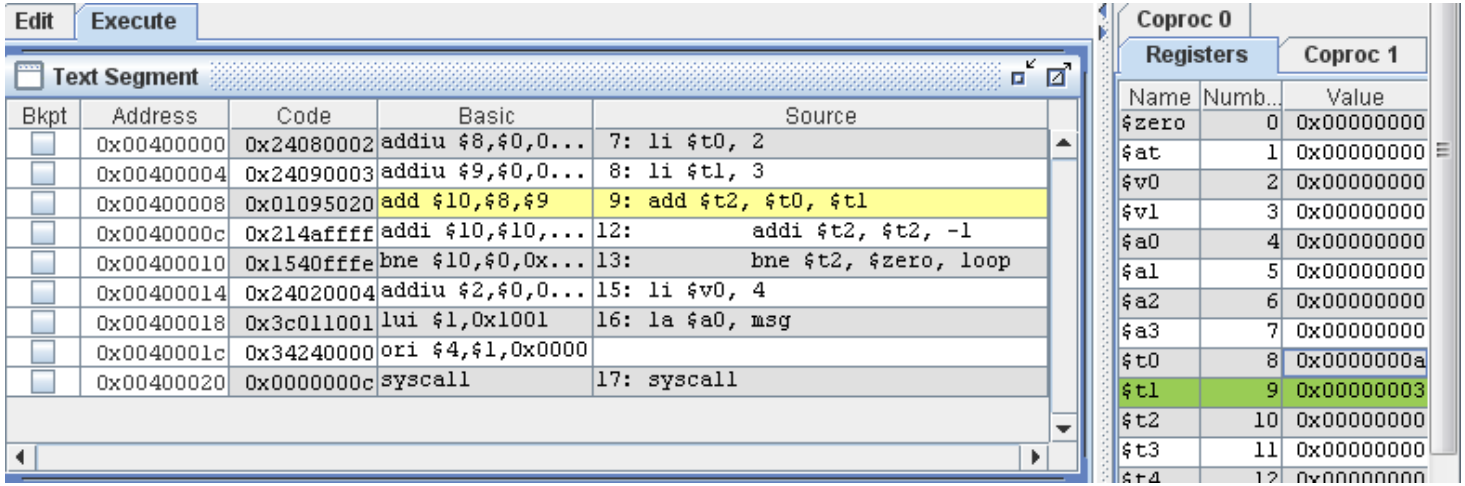
- Single Step the add instruction in Line 9.
- See how the addition used the new value of the register t0 (look at the sum in the register t2).



Modifying registers

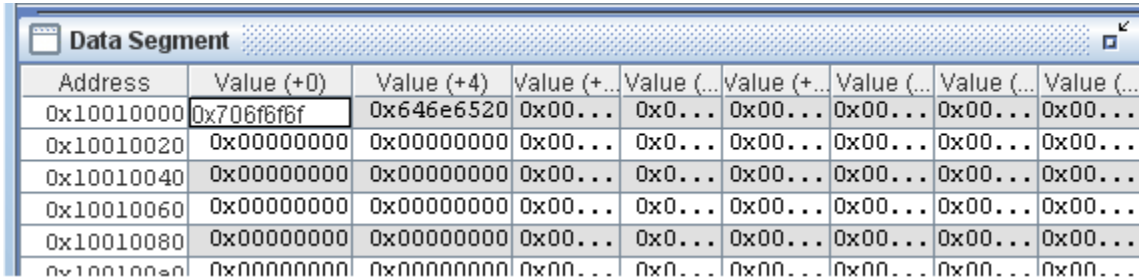
- New decimal values entered will be converted to the display format of the Registers window (hexadecimal by default).

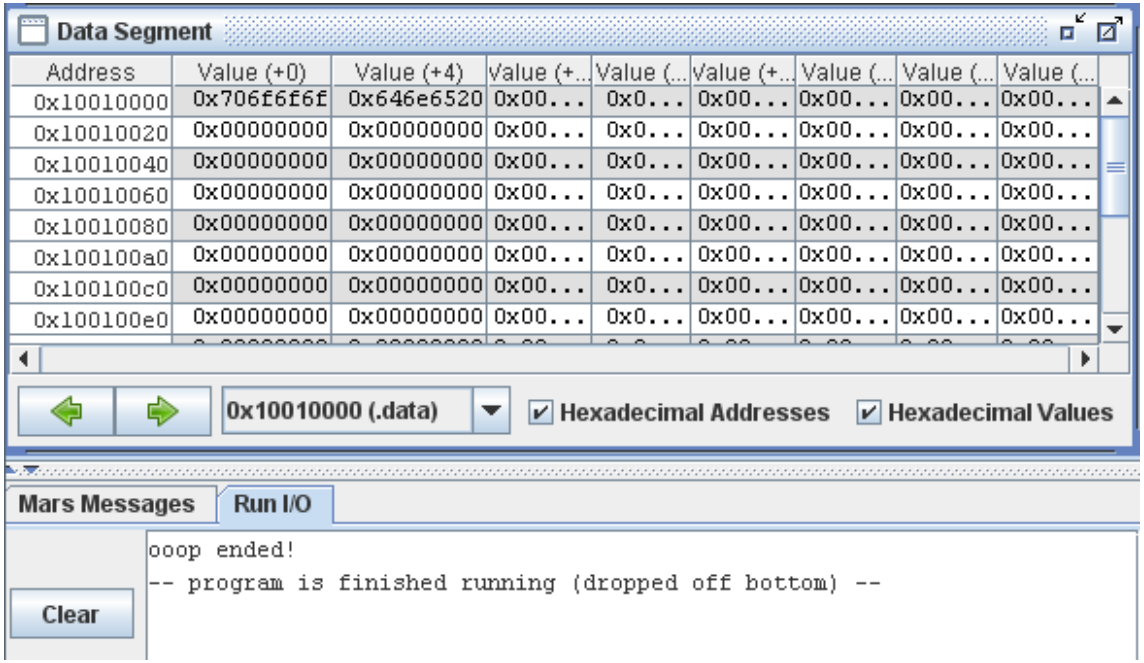




Modifying memory

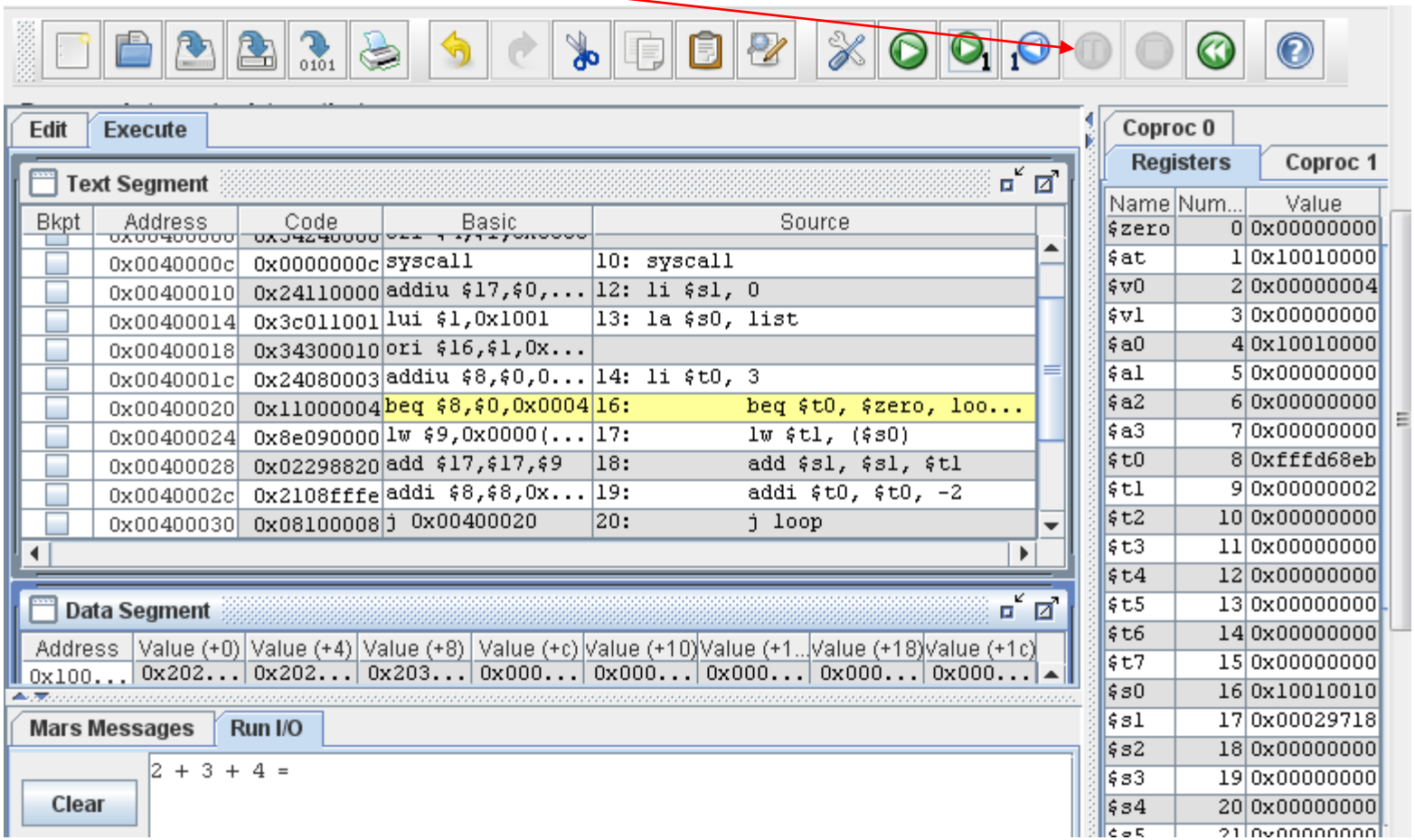
- For example, modify the string msg in debug1.s.





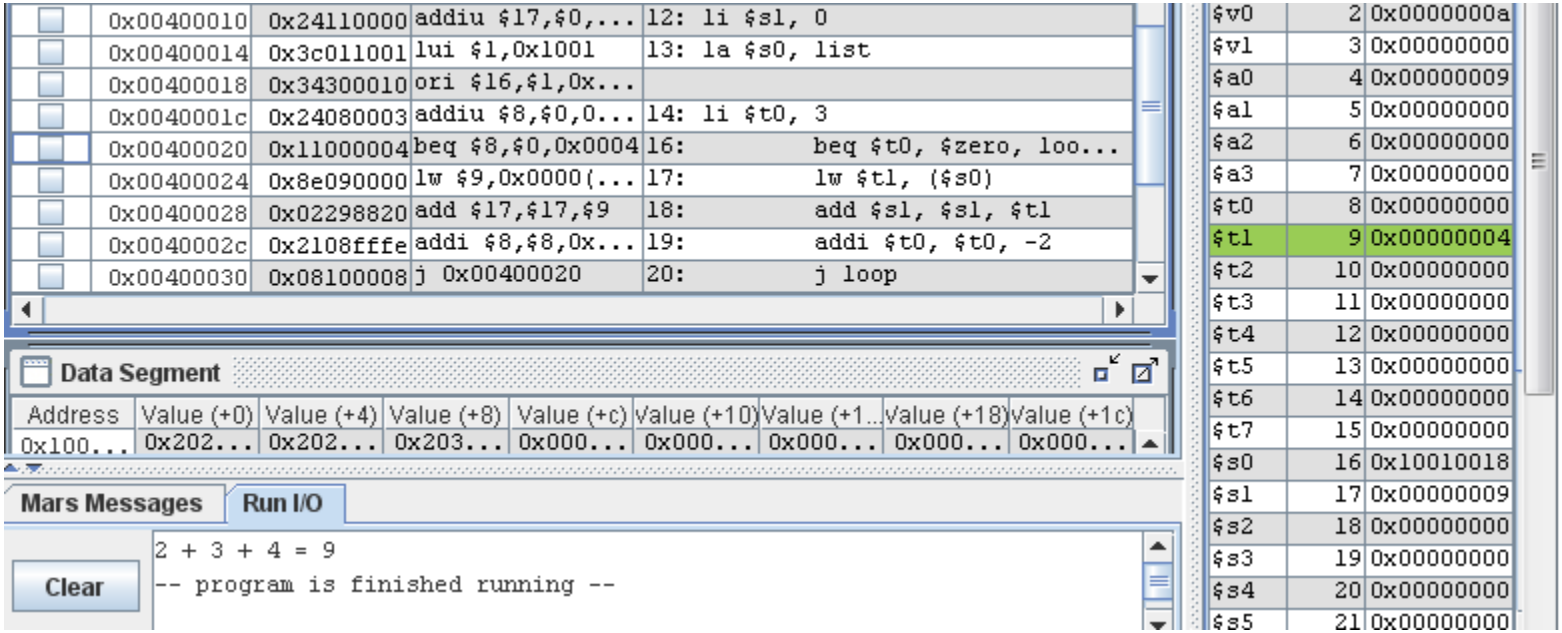
Example program

- Try to debug the example program debug2.s.
  - To pause the program execution (e.g., during its infinite loop), click this **Pause** button.



Example program

- After figuring out a possible solution to the bug, try it by modifying only the registers (not the program code) to get a sense of whether it should work (computing the correct sum).
- After a correct solution is found, fix the program code then.



Another example program

- Try to debug the program debug3.s.
  - You may debug it by modifying the registers or program code first (whichever way you feel efficient with).
  - The program execution may run for a while, looking like in an infinite loop initially, but will eventually terminates with an Exception error at a particular instruction code.
  - See the message about the Exception on Mars' Messages Window.
  - The Exception is about an invalid memory access by the instruction. What is the cause?

Conclusion

- You have learnt:
  - how to debug a MIPS program using the debugging features in Mars.