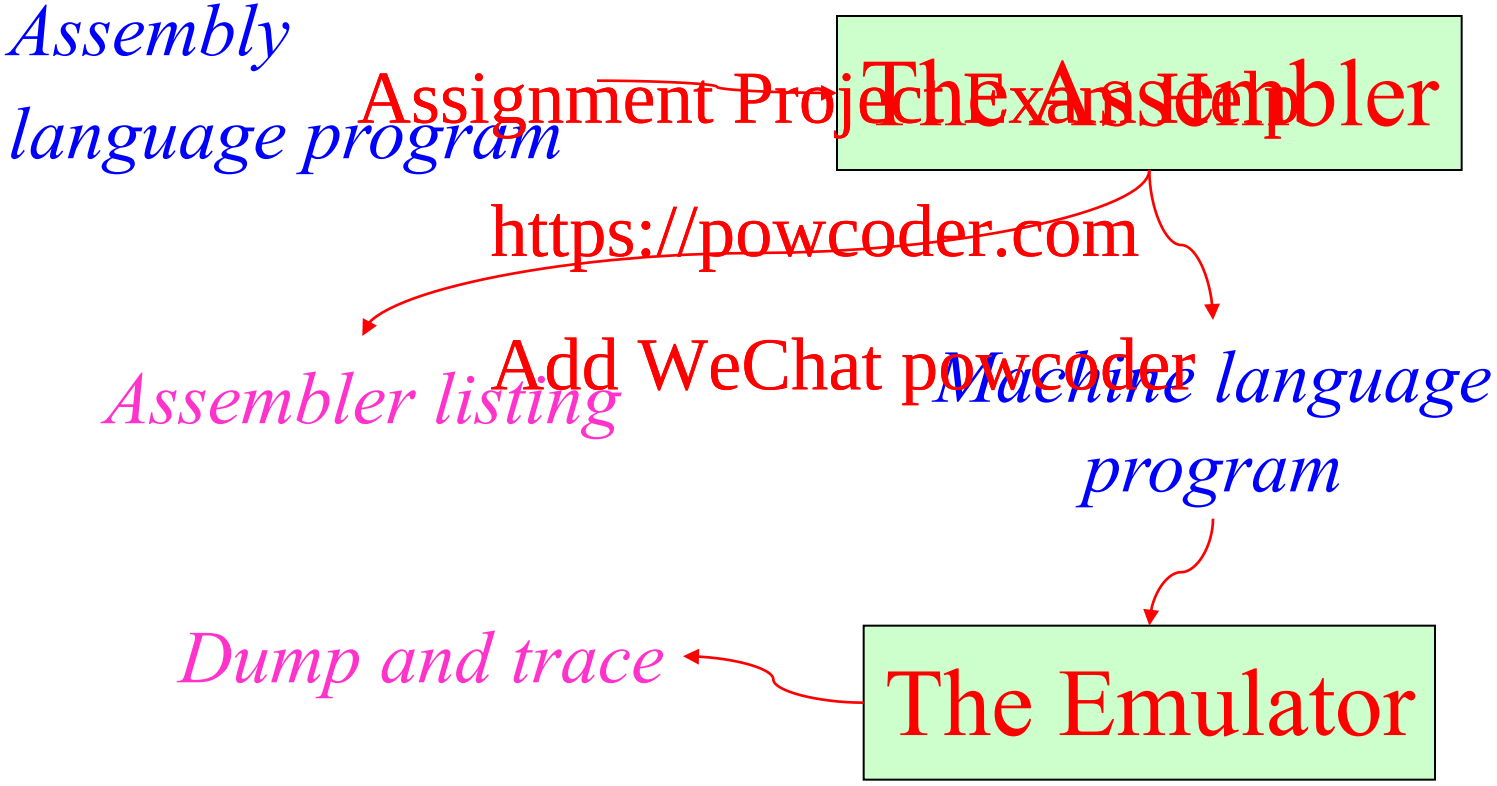
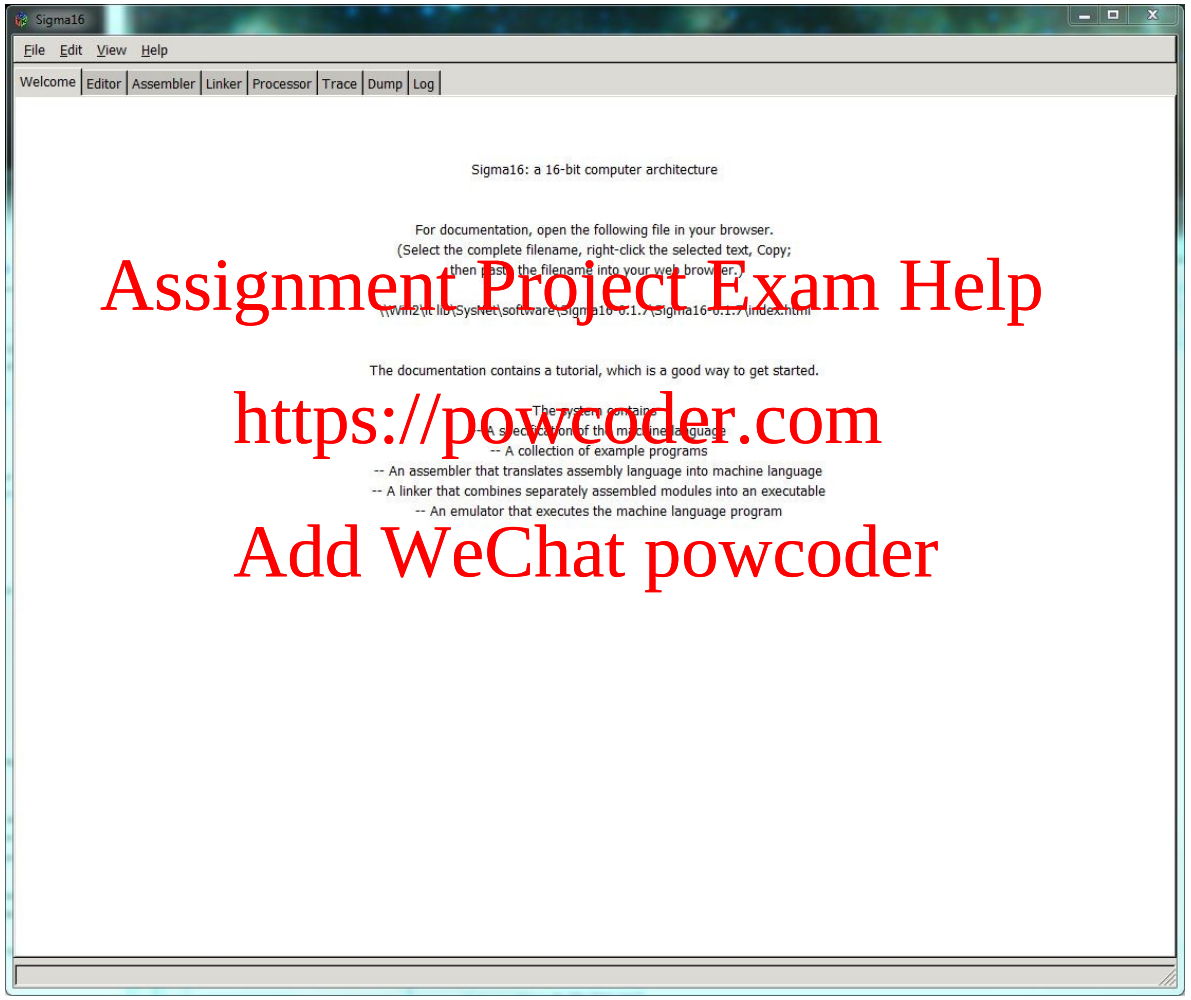
Assembling and Running Programs with Sigma16

Assignments Project Examples

Software Tools

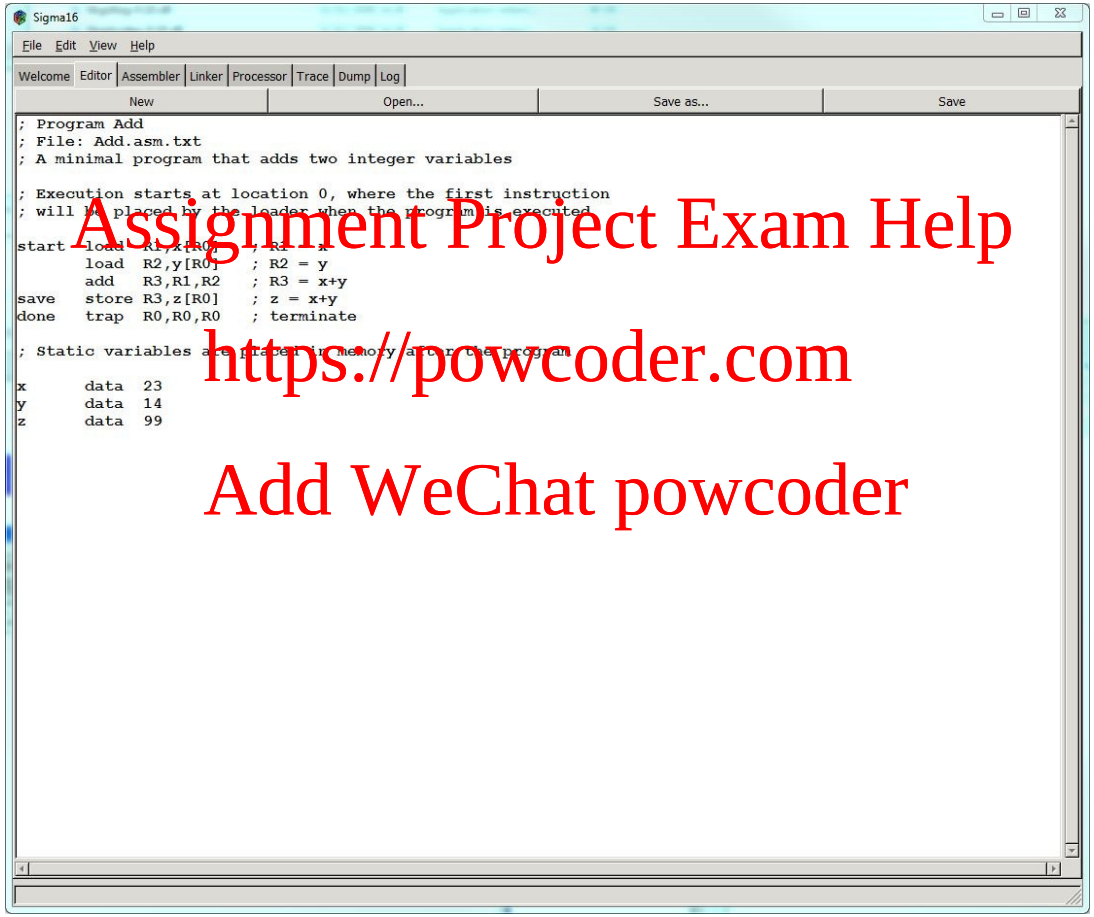


Welcome Screen



The Editor

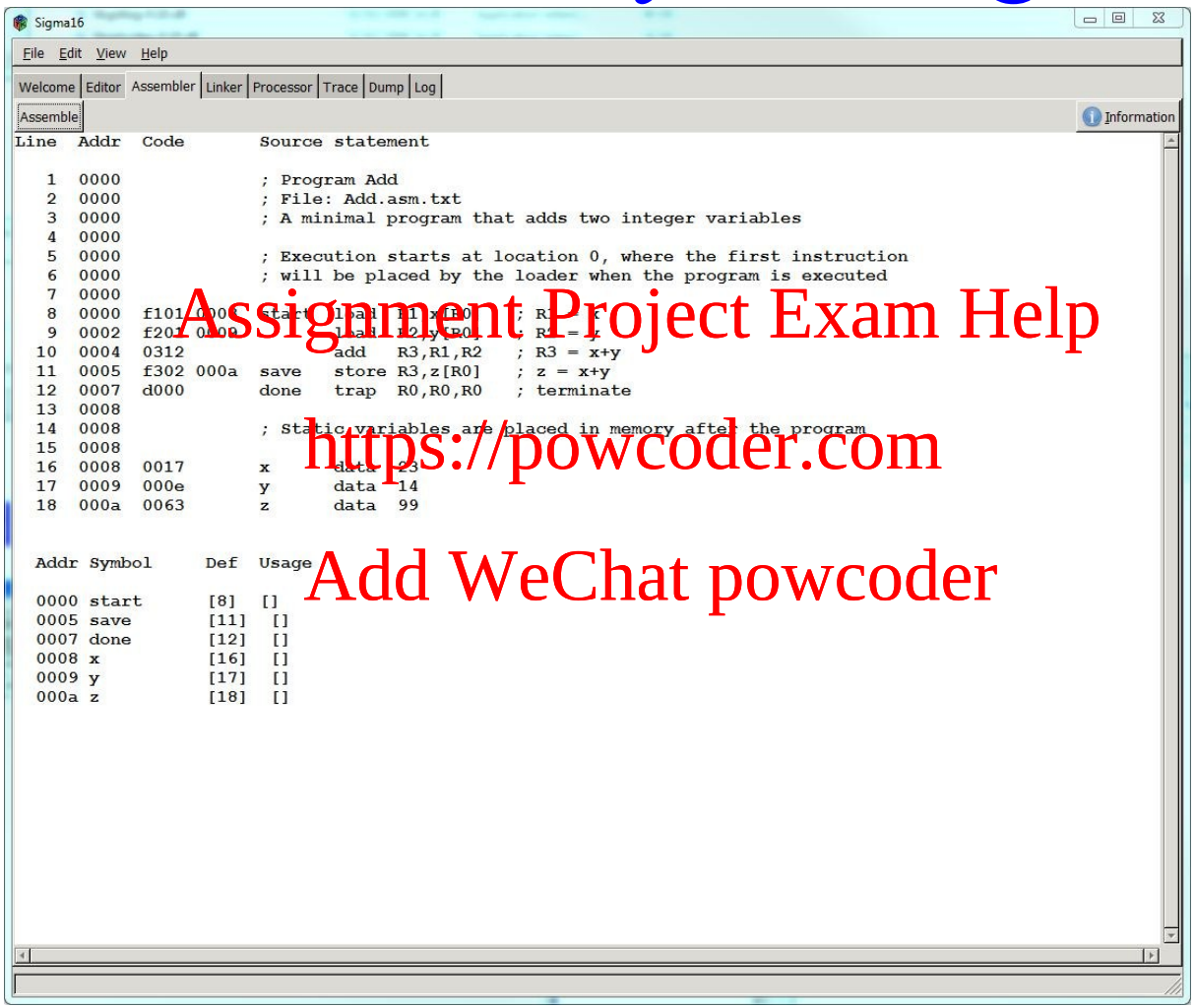
- Open the file add.asm.txt in the Examples folder:



Editing the program

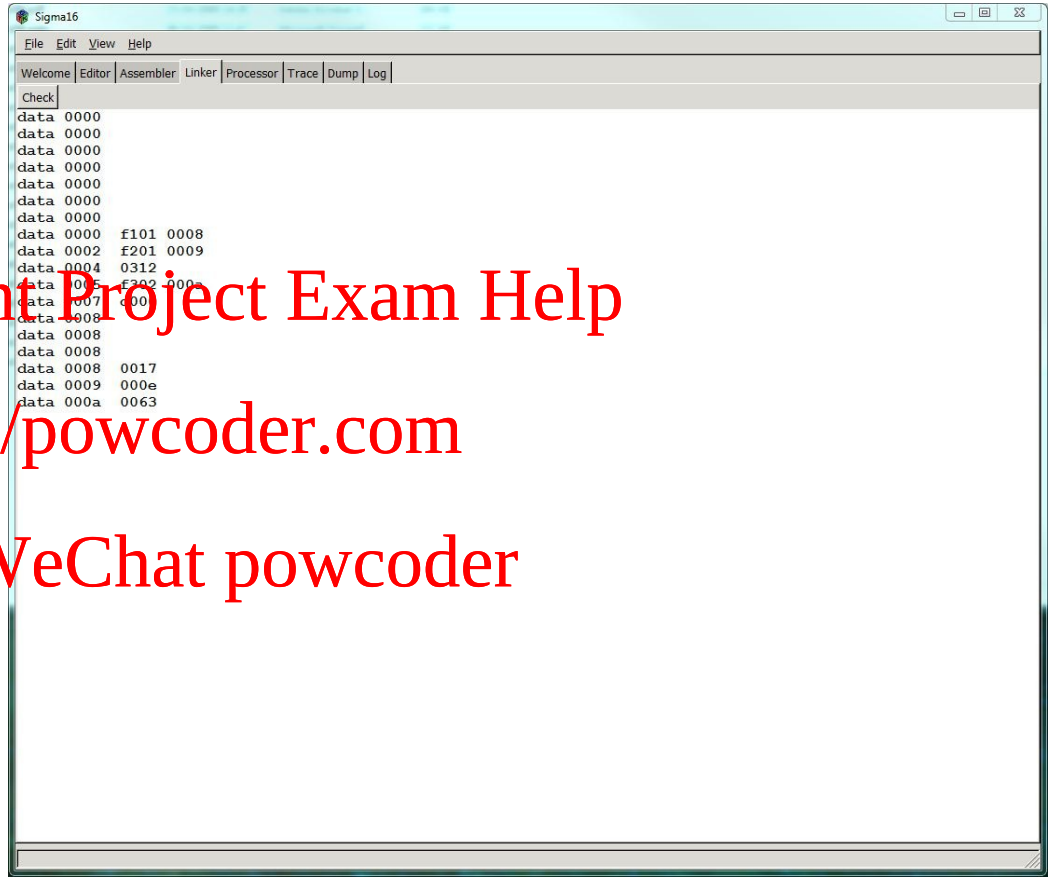
- You can use an external editor (e.g. WordPad), save the document after making changes, and open it.
  - You must save as plain text
  - Do not try to save inside the Sigma16 folder itself
- Alternatively, you can just use the editor in the Editor pane of Sigma16.
- The examples don't need any editing!
- Whatever editor you use, save your document from time to time!
- Now the program needs to be assembled
- Go to the Assembler page, and click Assemble

Assembly Listing

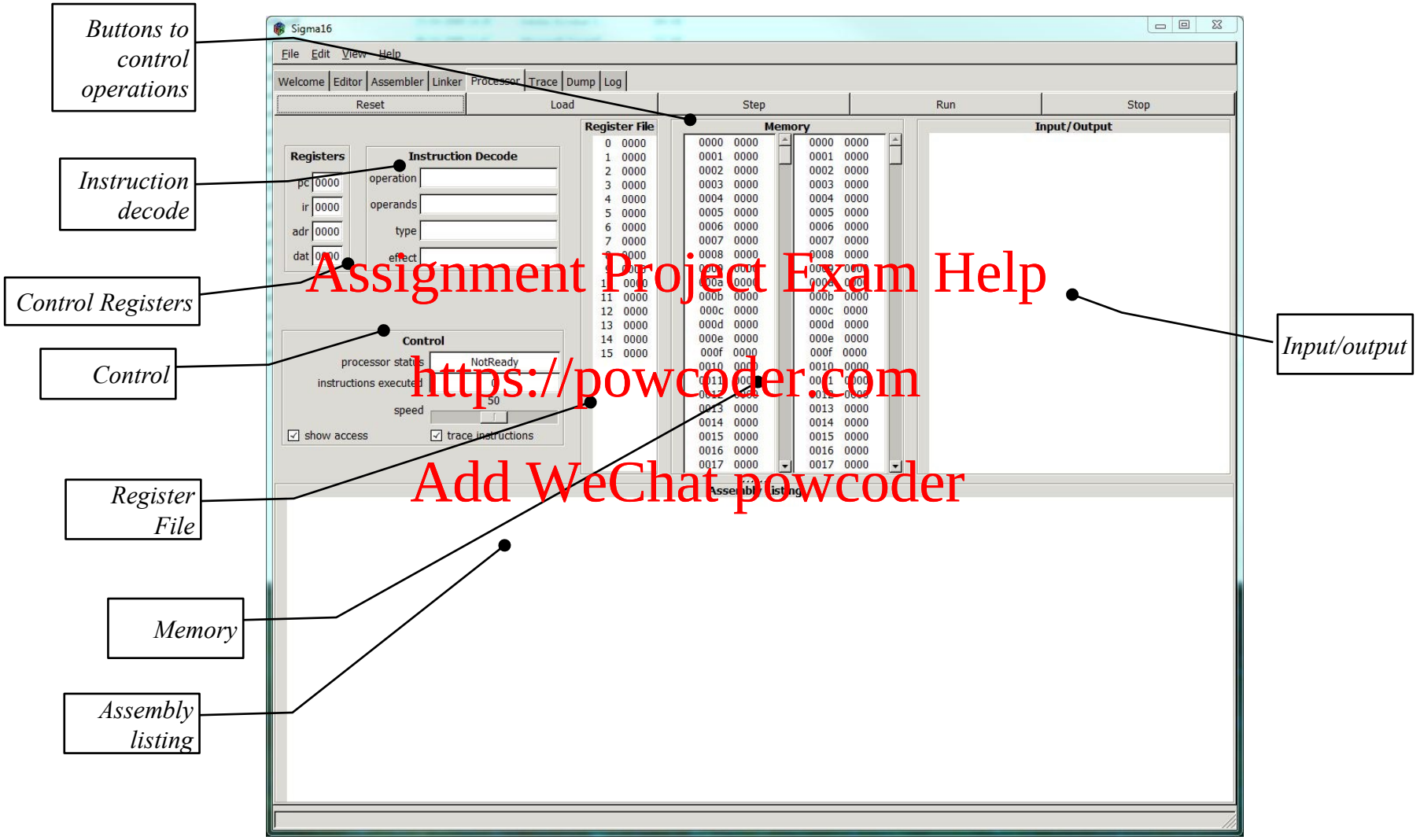


Linker

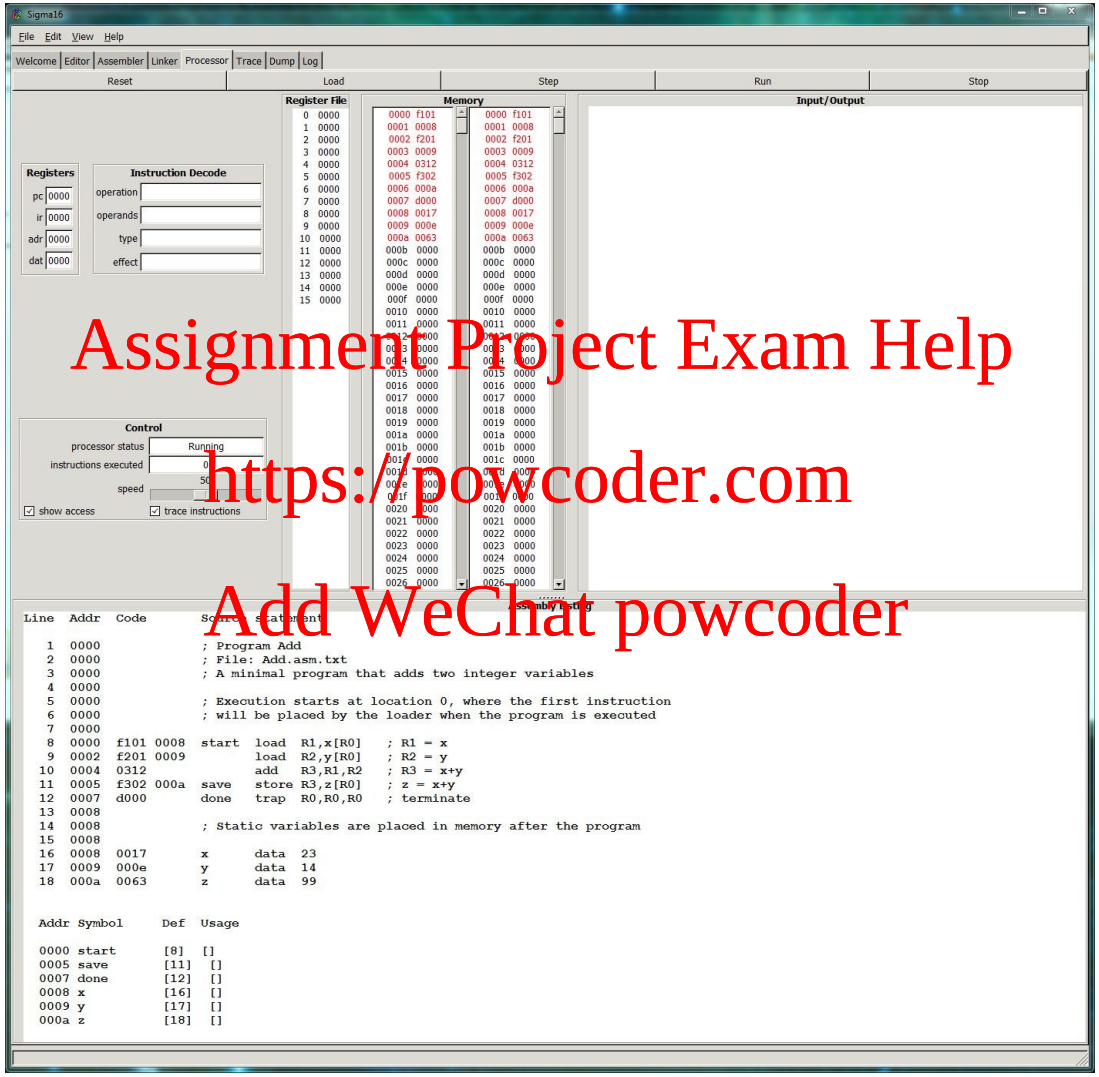
- The linker page shows your object code
- This is essentially what gets saved on disk as the result of compiling a program; it is raw machine language
- Actually, the linker does some major work: it combines separately compiled modules into an executable
- But we are working with single-module programs, so you don't have to worry about the linker



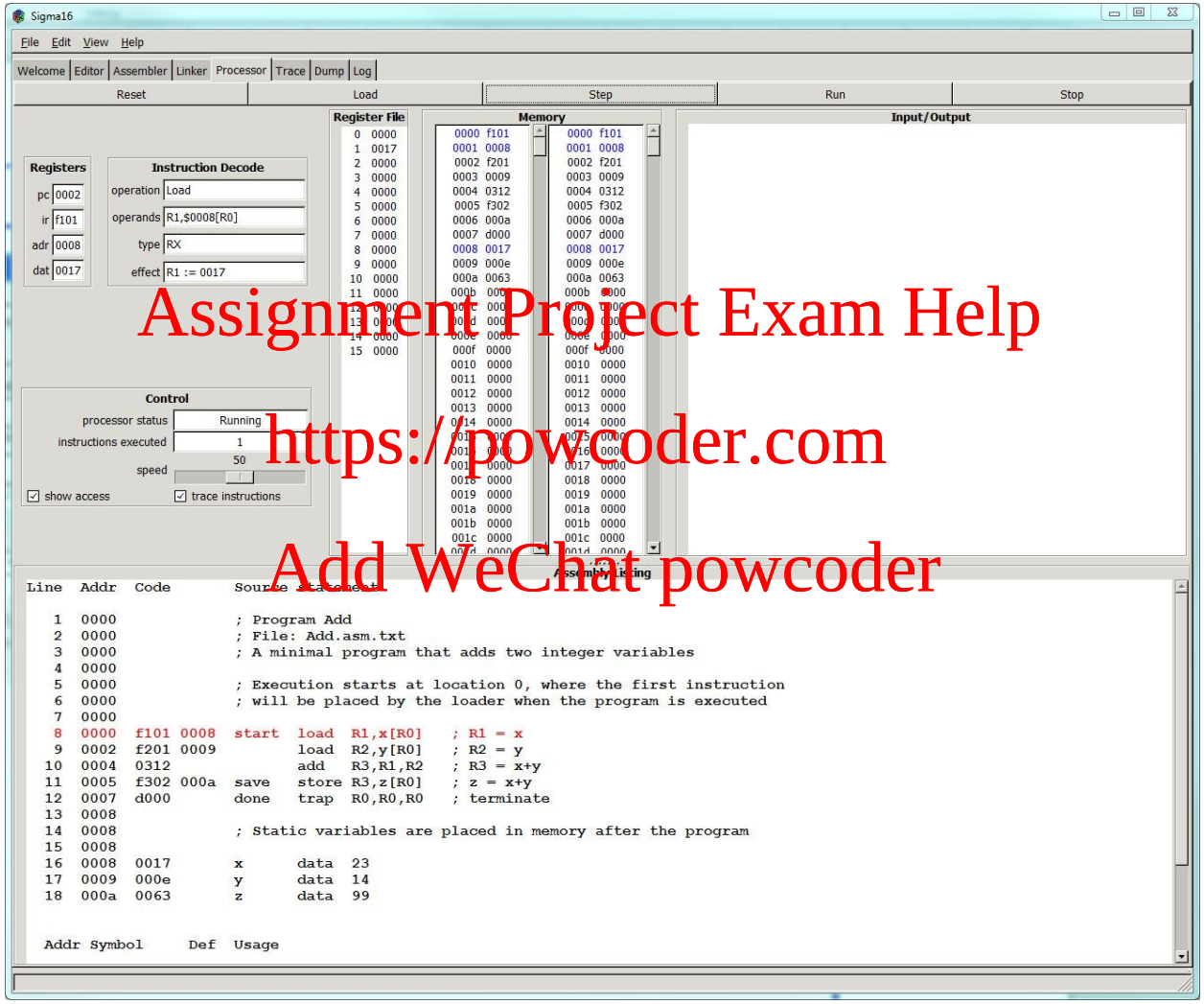
The Processor Page



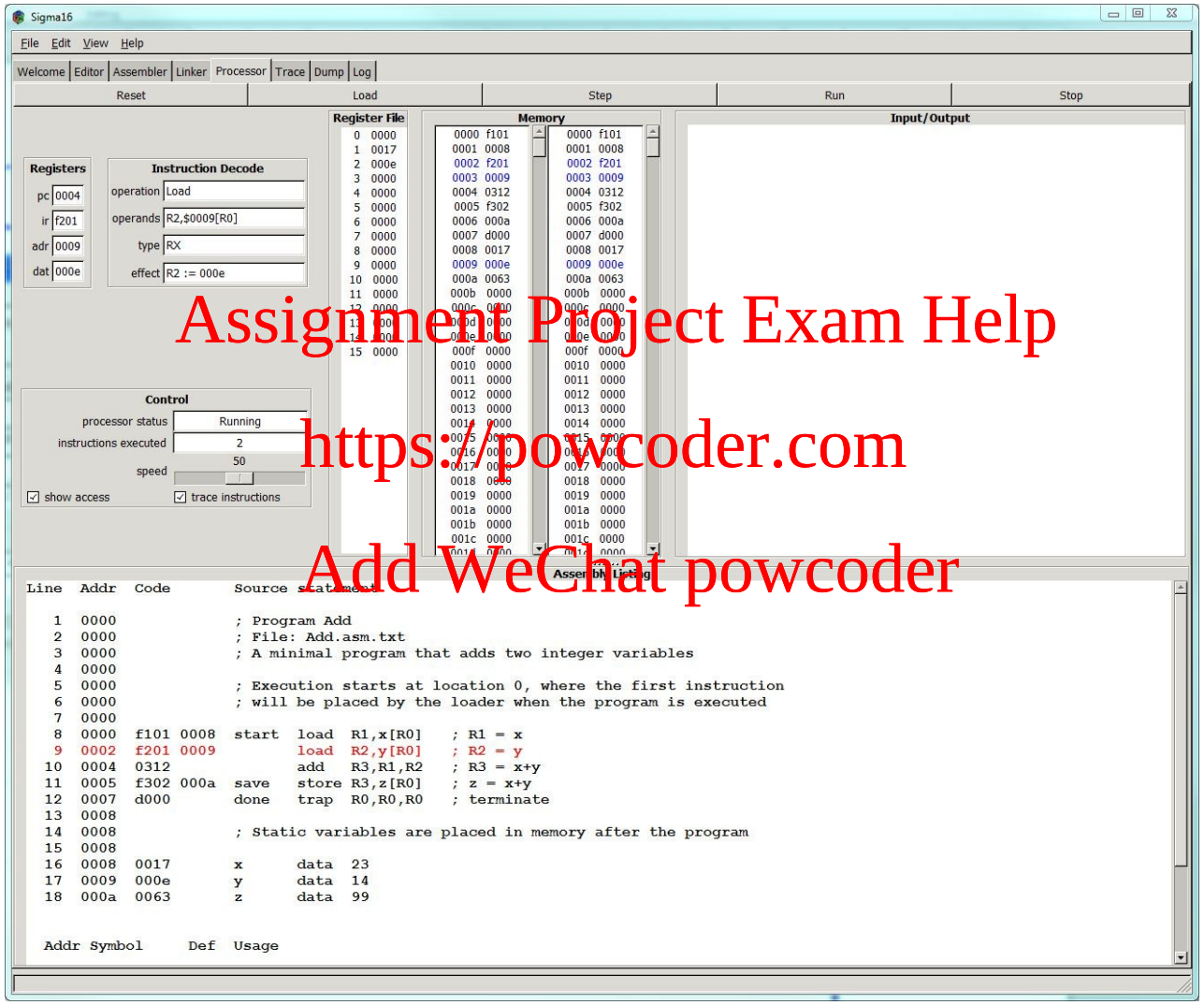
Loading (click “Load”)



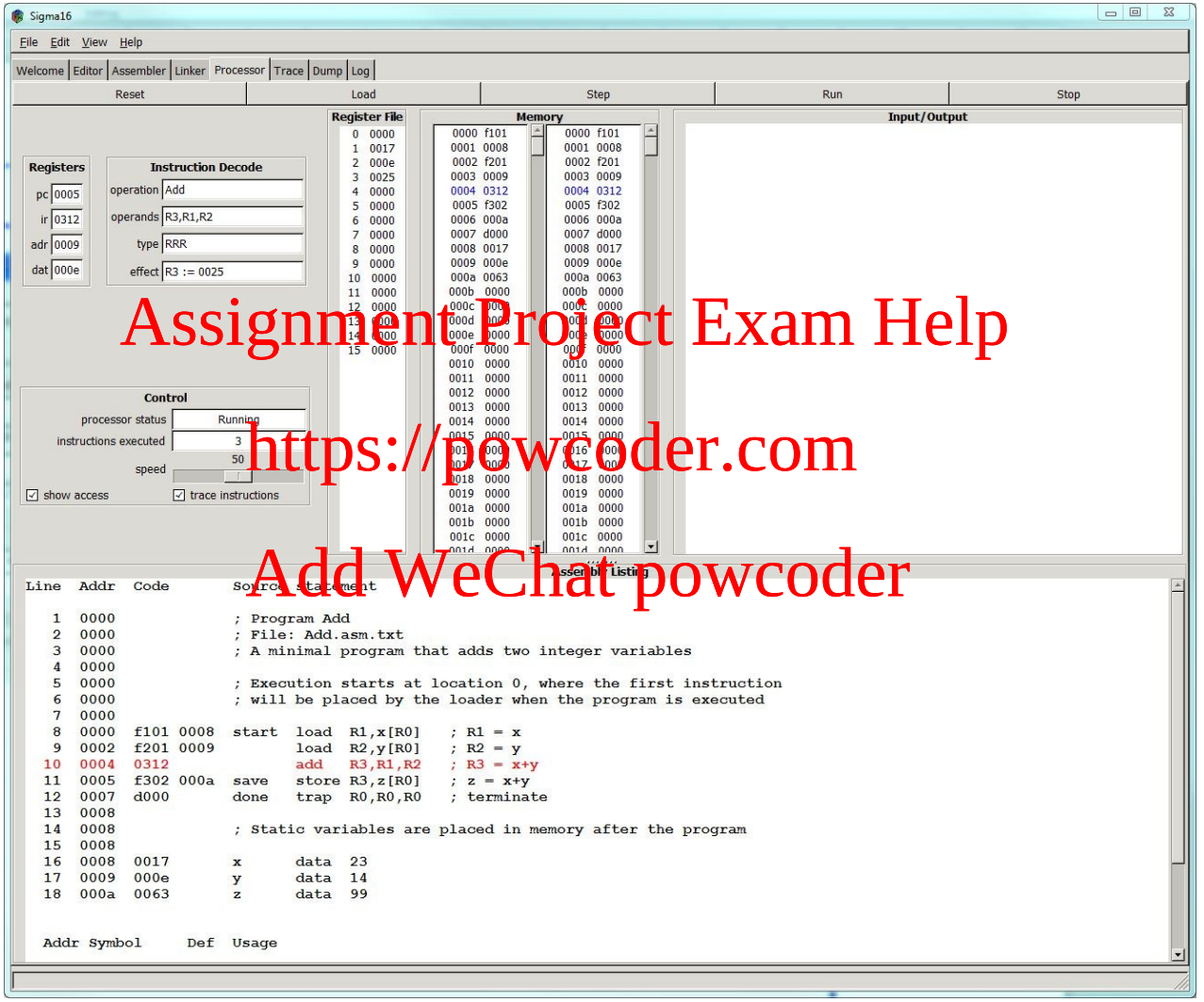
Click Step button



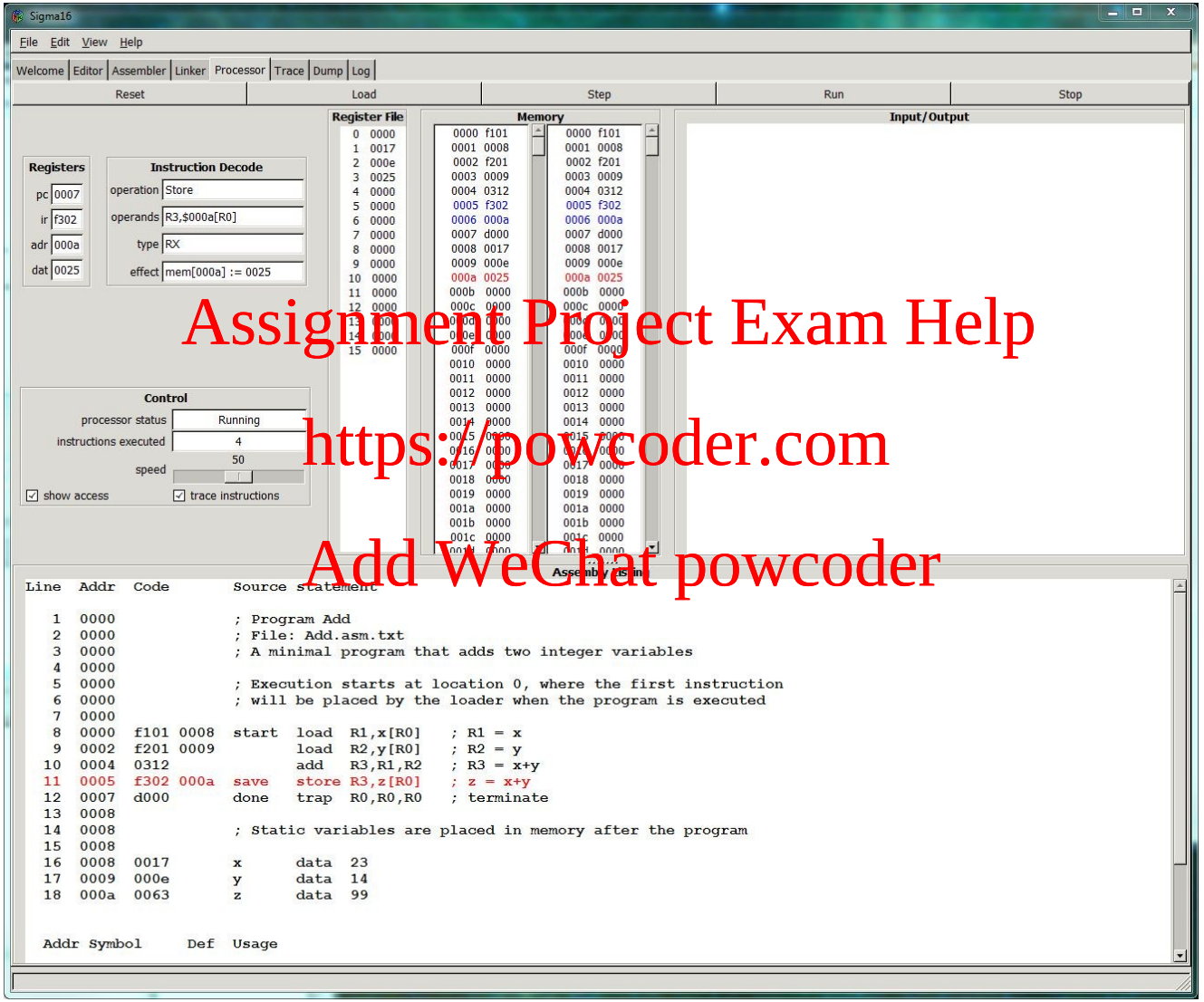
Click Step again



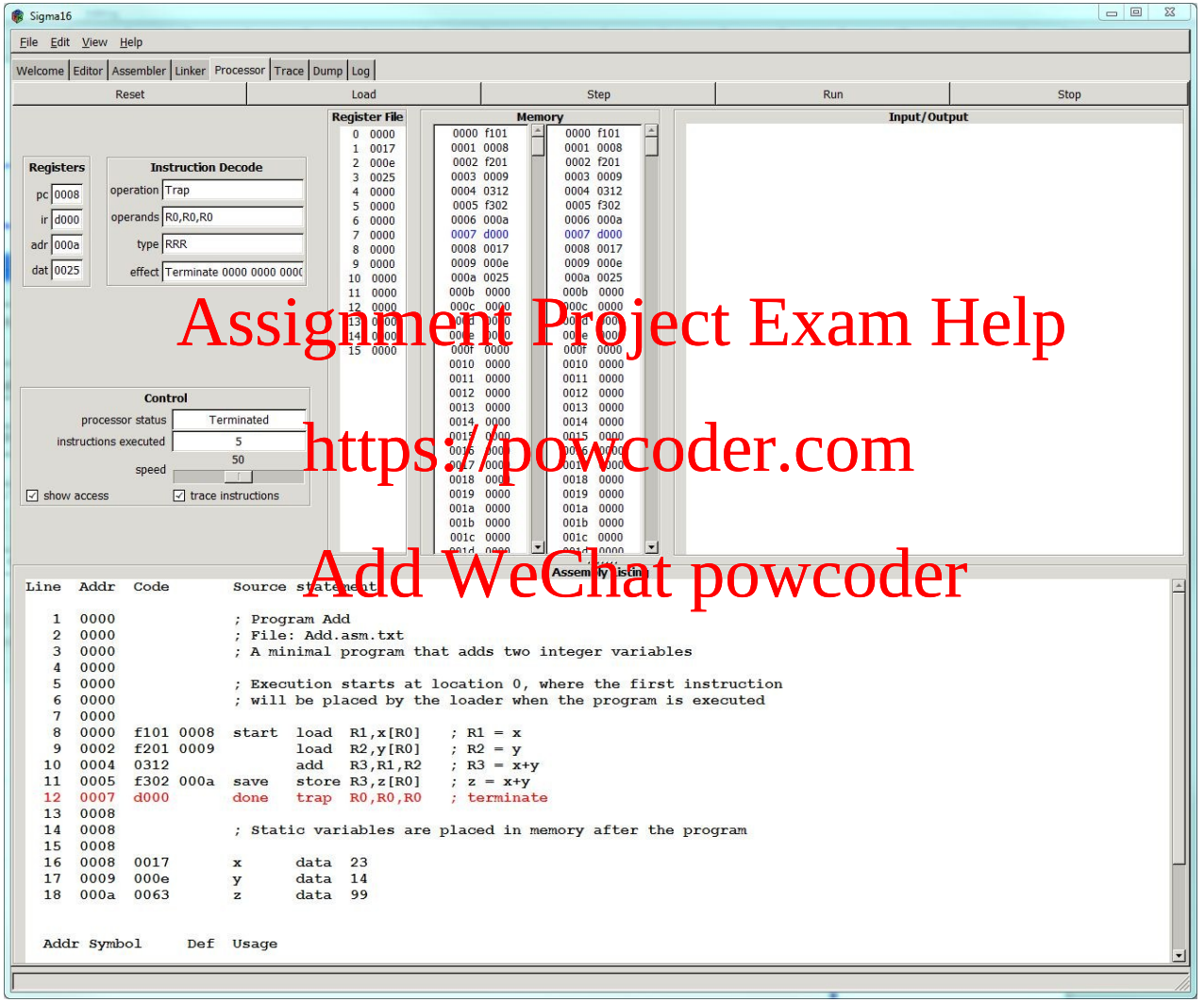
And again...



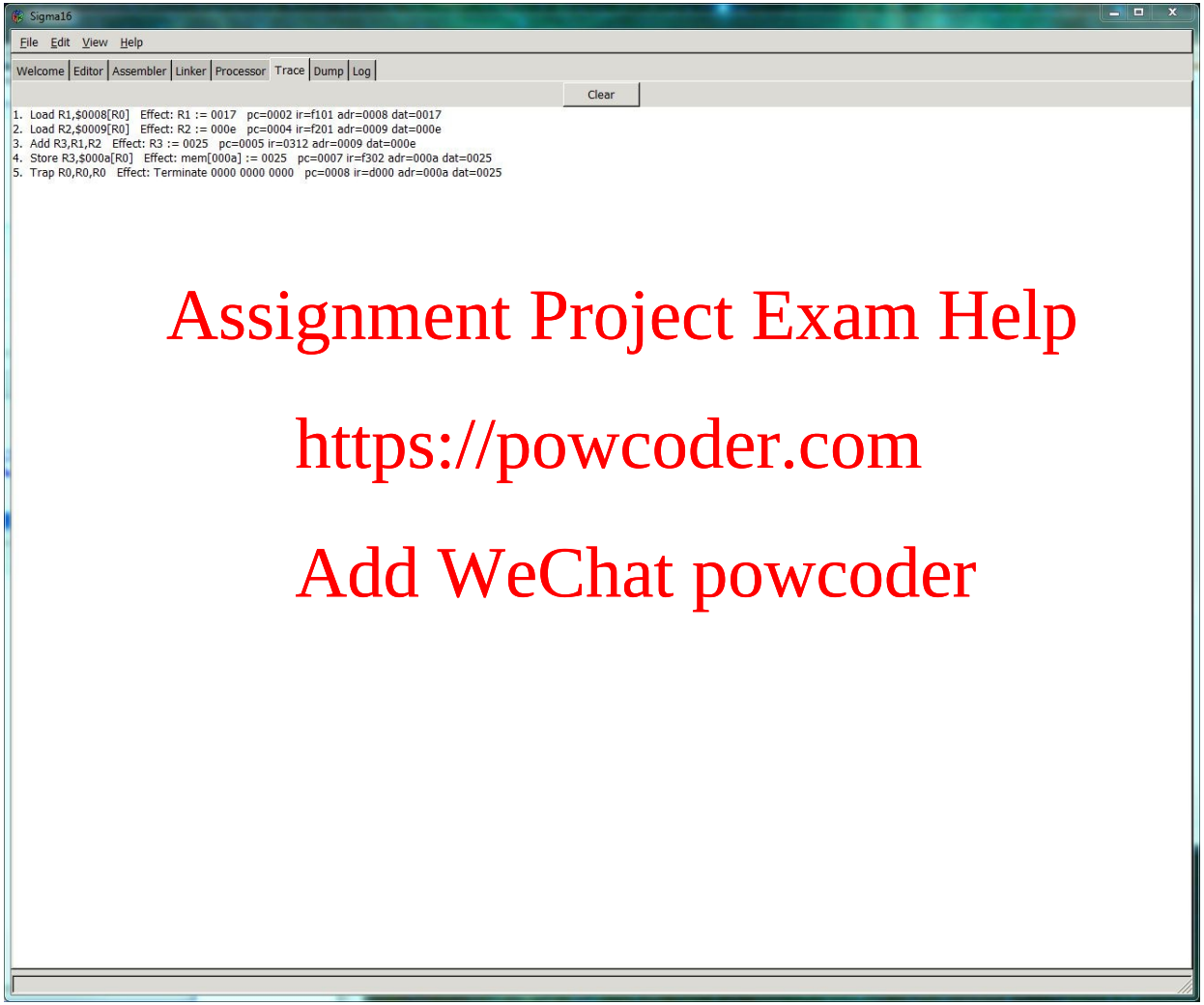
Yet again...



One last instruction to do...



Look at the Trace page



Letting the machine run

- To re-run a program,
  - Go to Processor page
  - Click Load (this reloads the executable program into memory)
- To run automatically, click the Run button
- Adjust the execution speed with the speed slider
- To stop execution, click Stop; resume with Step or Run

Trace of Machine Instructions

The emulator provides a readable trace showing each instruction that it executes. It can also catch some runtime errors.

This is a big advantage of using an emulator!

Computer hardware doesn't do this. The emulator trace shows you what the hardware does.

Coping With Errors

There are two ways you can go wrong:

- If there is a syntax error in the program, the Assembler will indicate an error and the program won't run.
- If the program is well-formed, the system will launch it. But it's still possible that the program contains a bug—this means that blindly doing what the instructions say to do produces chaos.

Assembly language syntax

- Look at the examples to see what correct statements & instructions look like
- For operations, load LOAD Load are all ok
- For names, MyVar myvar MYVAR are considered to be distinct (case sensitive)
- ; indicates that the rest of the line is a comment

Using comments

- Give a preliminary comment identifying the program
- Give the algorithm as pseudocode, or Java etc.
- Use blank lines to break up the instructions into blocks
- Use full-line comments to say what a block of code does
- Use detailed comments on every instruction