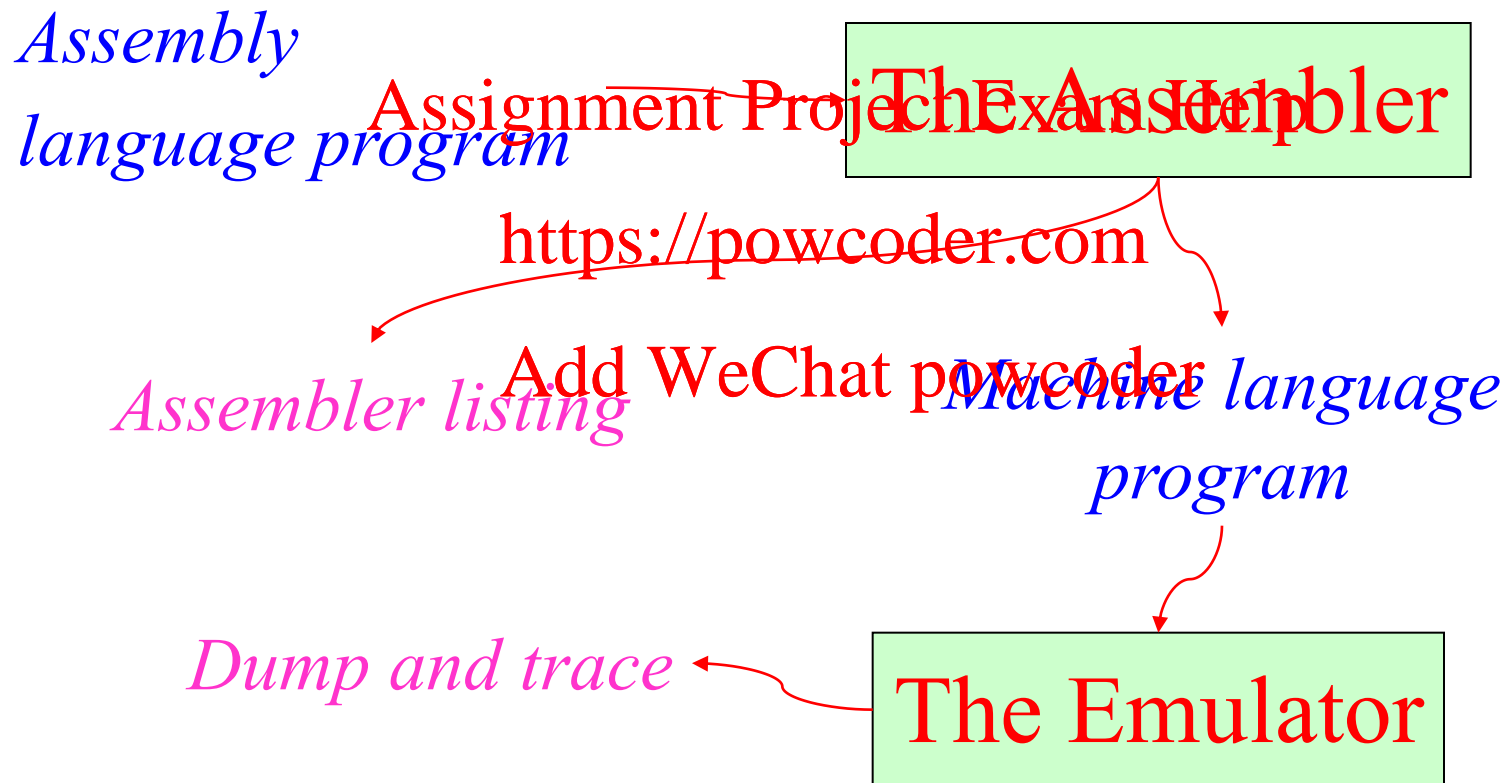


Assembling and Running Programs with Signal 6

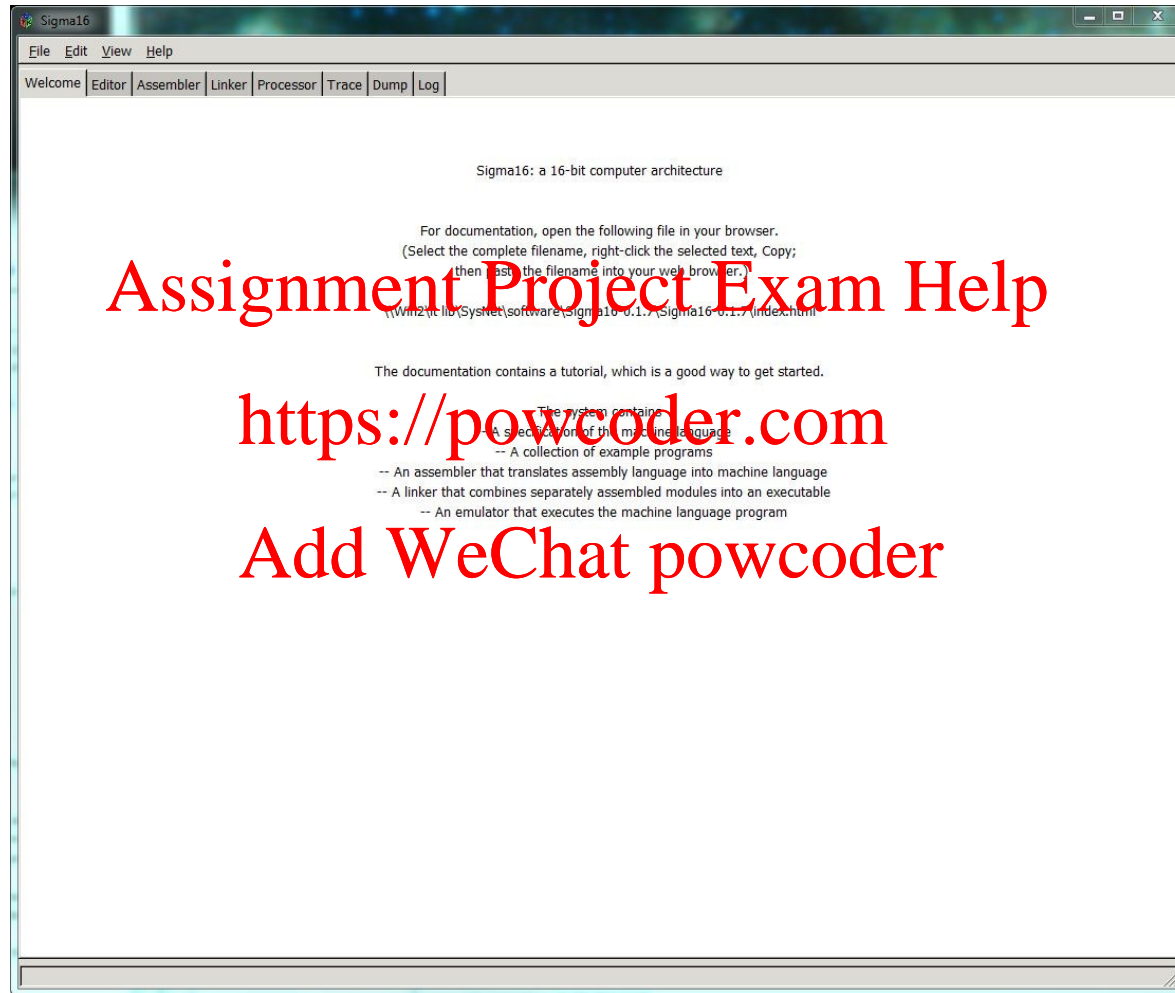
<https://powcoder.com>

Add WeChat powcoder

Software Tools

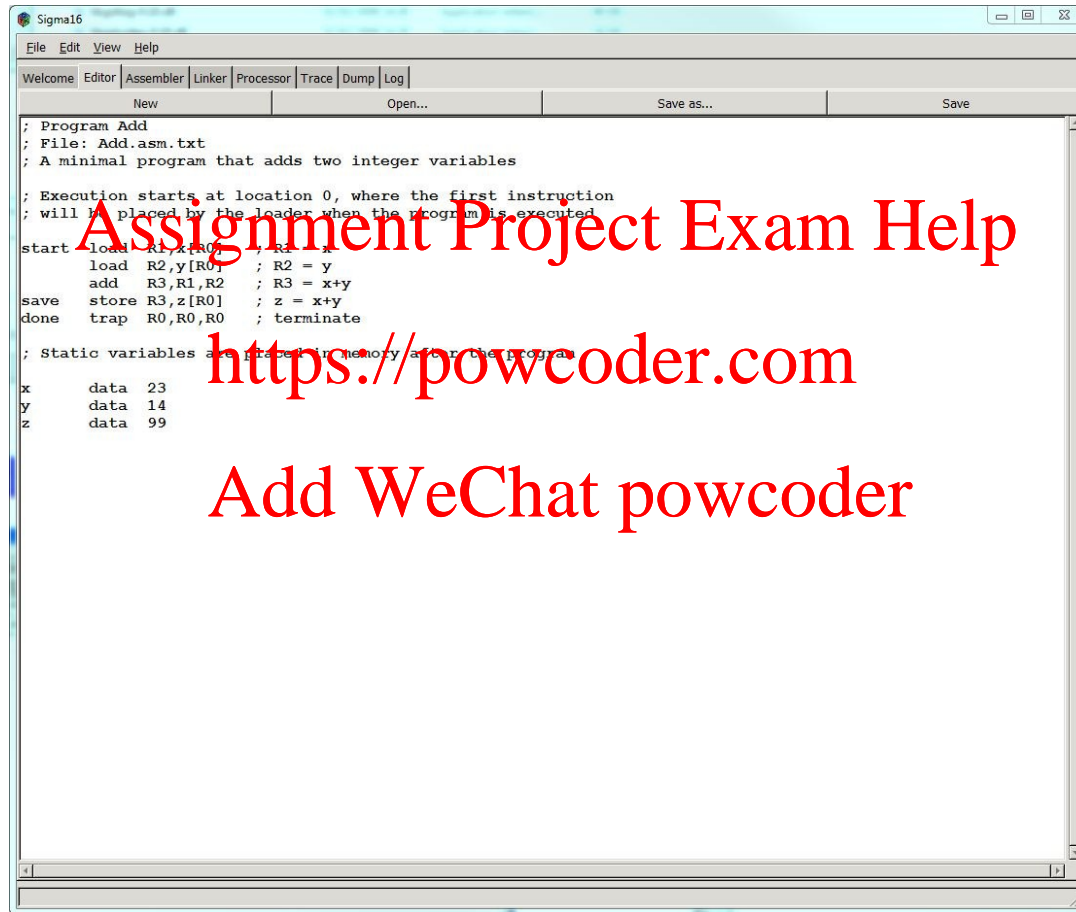


Welcome Screen



The Editor

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



The screenshot shows the Sigma16 editor window with the following assembly code:

```
; Program Add
; File: Add.asm.txt
; A minimal program that adds two integer variables

; Execution starts at location 0, where the first instruction
; will be placed by the loader when the program is executed
start load R1,x[R0] ; R1 = x
      load R2,y[R0] ; R2 = y
      add  R3,R1,R2 ; R3 = x+y
save   store R3,z[R0] ; z = x+y
done   trap R0,R0,R0 ; terminate

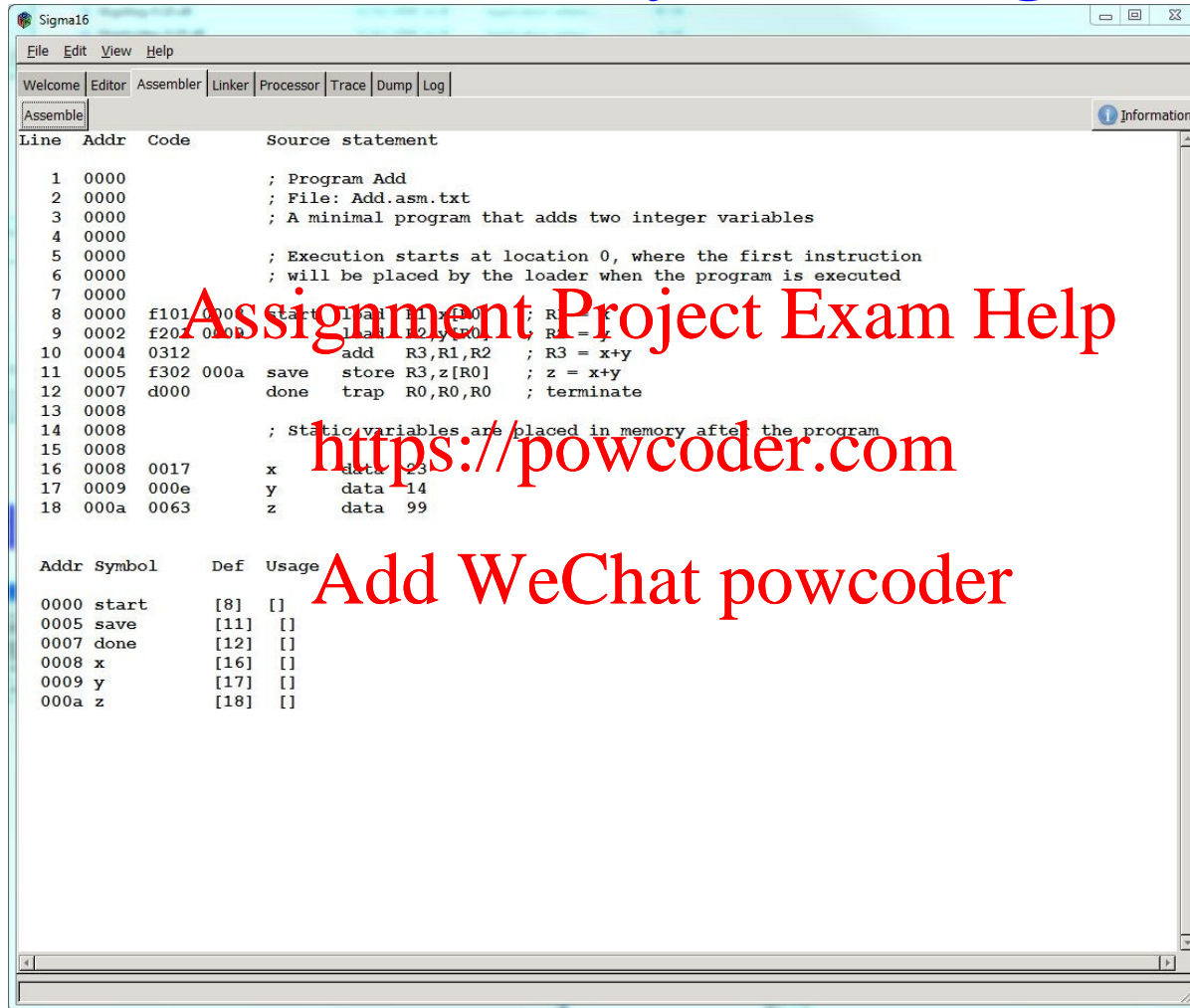
; Static variables are placed in memory after the program
x      data 23
y      data 14
z      data 99
```

Overlaid on the screenshot is the red text: <https://powcoder.com> and the text "Add WeChat powcoder".

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



The screenshot shows the Sigma16 assembly editor interface. The main window displays an assembly listing with columns for Line, Addr, Code, and Source statement. The program is a minimal assembly that adds two integer variables, x and y, and stores the result in z. The source statements include comments and instructions like load, add, store, and trap. A symbol table at the bottom lists the addresses and symbols used in the program.

```
Line  Addr  Code  Source statement
1      0000      ; Program Add
2      0000      ; File: Add.asm.txt
3      0000      ; A minimal program that adds two integer variables
4      0000
5      0000      ; Execution starts at location 0, where the first instruction
6      0000      ; will be placed by the loader when the program is executed
7      0000
8      0000 f101 000 start load R1,x[R0] ; R1 = x
9      0002 f201 000 load R2,y[R0] ; R2 = y
10     0004 0312 add R3,R1,R2 ; R3 = x+y
11     0005 f302 000a save store R3,z[R0] ; z = x+y
12     0007 d000 done trap R0,R0,R0 ; terminate
13     0008
14     0008      ; Static variables are placed in memory after the program
15     0008
16     0008 0017 x data 23
17     0009 000e y data 14
18     000a 0063 z data 99
```

Addr	Symbol	Def	Usage
0000	start	[8]	[]
0005	save	[11]	[]
0007	done	[12]	[]
0008	x	[16]	[]
0009	y	[17]	[]
000a	z	[18]	[]

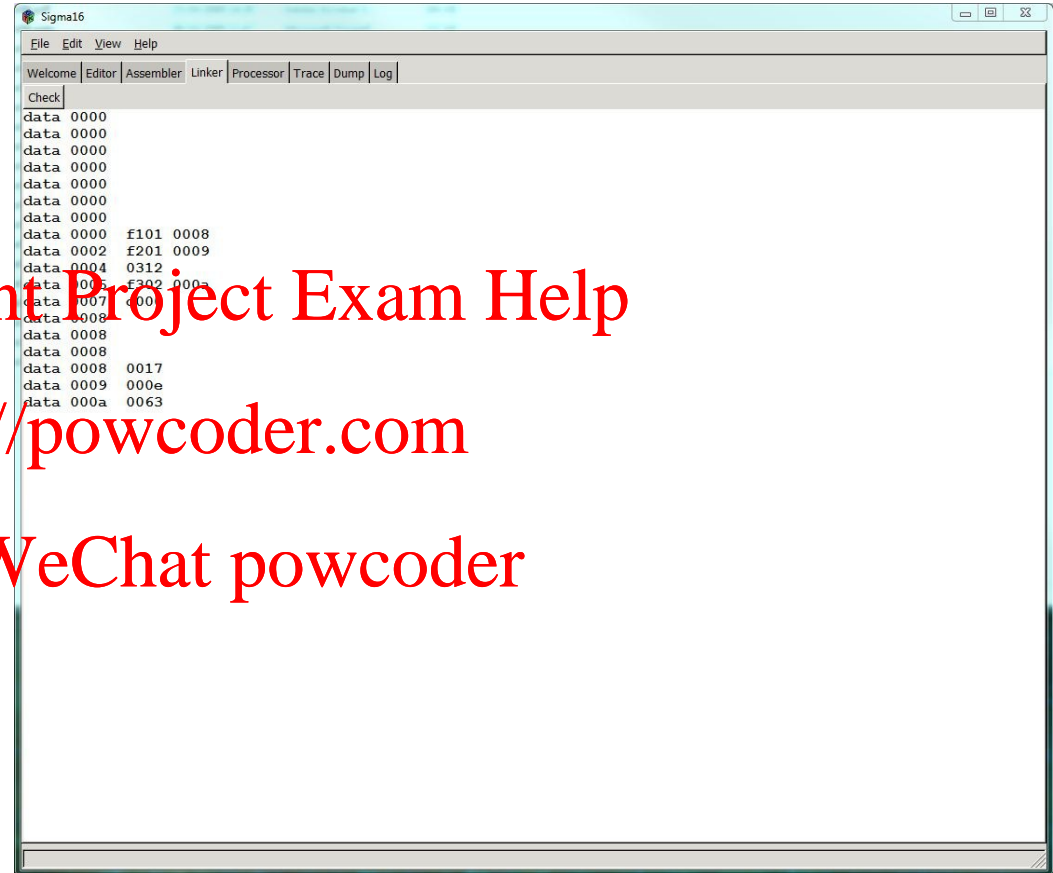
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

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



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

The Processor Page

The screenshot shows the Sigma16 Processor Page interface. The top menu bar includes File, Edit, View, and Help. Below it is a toolbar with buttons for Welcome, Editor, Assembler, Linker, Processor, Trace, Dump, and Log. The main area is divided into several sections: Registers, Instruction Decode, Register File, Memory, and Input/Output. The Registers section shows a list of registers (pc, ir, adr, dat) and their values. The Instruction Decode section shows the current instruction's operation, operands, type, and effect. The Register File section shows a list of registers (0-15) and their values. The Memory section shows a list of memory addresses (0000-0017) and their values. The Input/Output section shows a large text area for input and output data. The Control section shows the processor status (NotReady), instructions executed (0), speed (50), and checkboxes for show access and trace instructions. The Assembly listing section shows the current instruction being assembled.

Buttons to control operations

Instruction decode

Control Registers

Control

Register File

Memory

Assembly listing

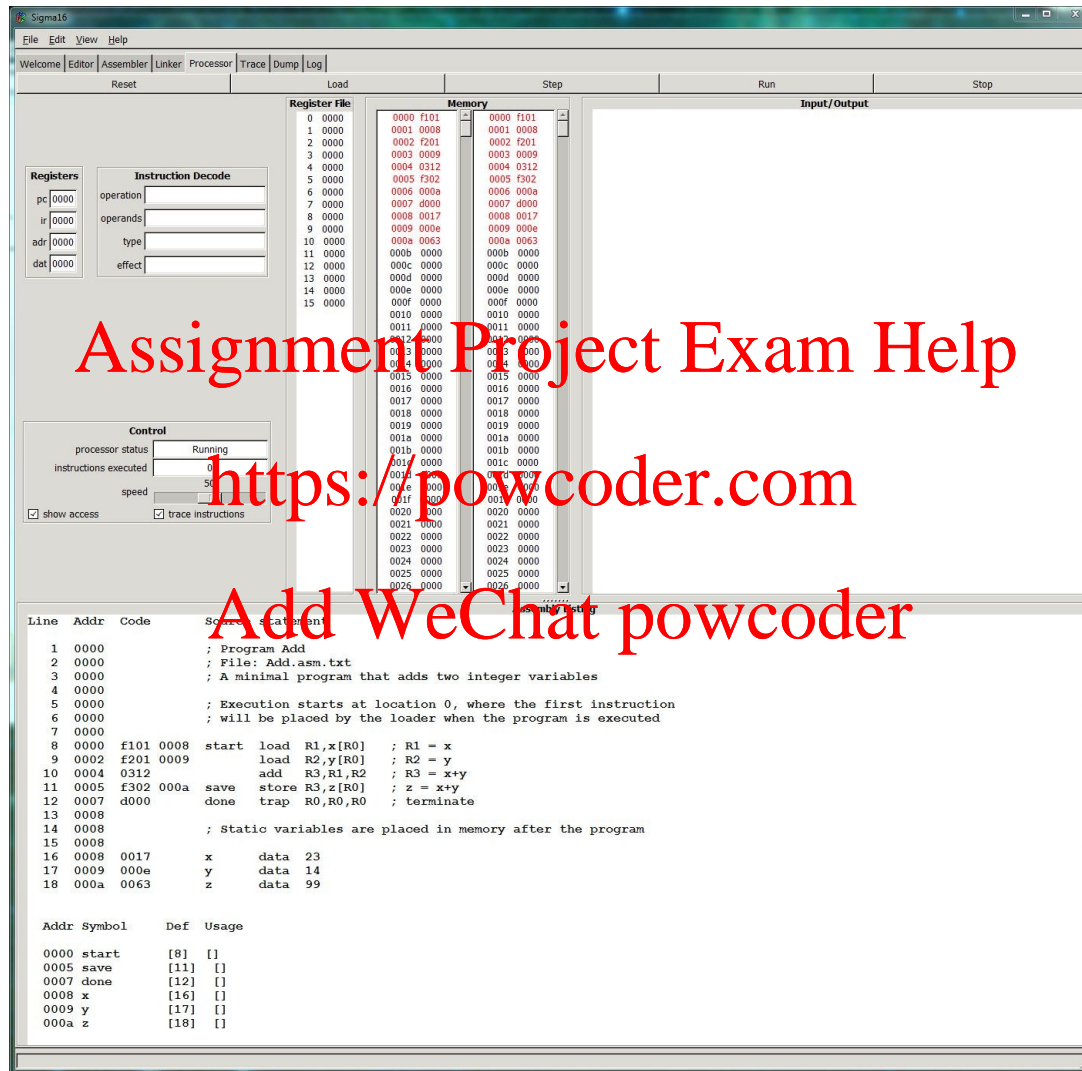
Input/output

Assignment Project Exam Help

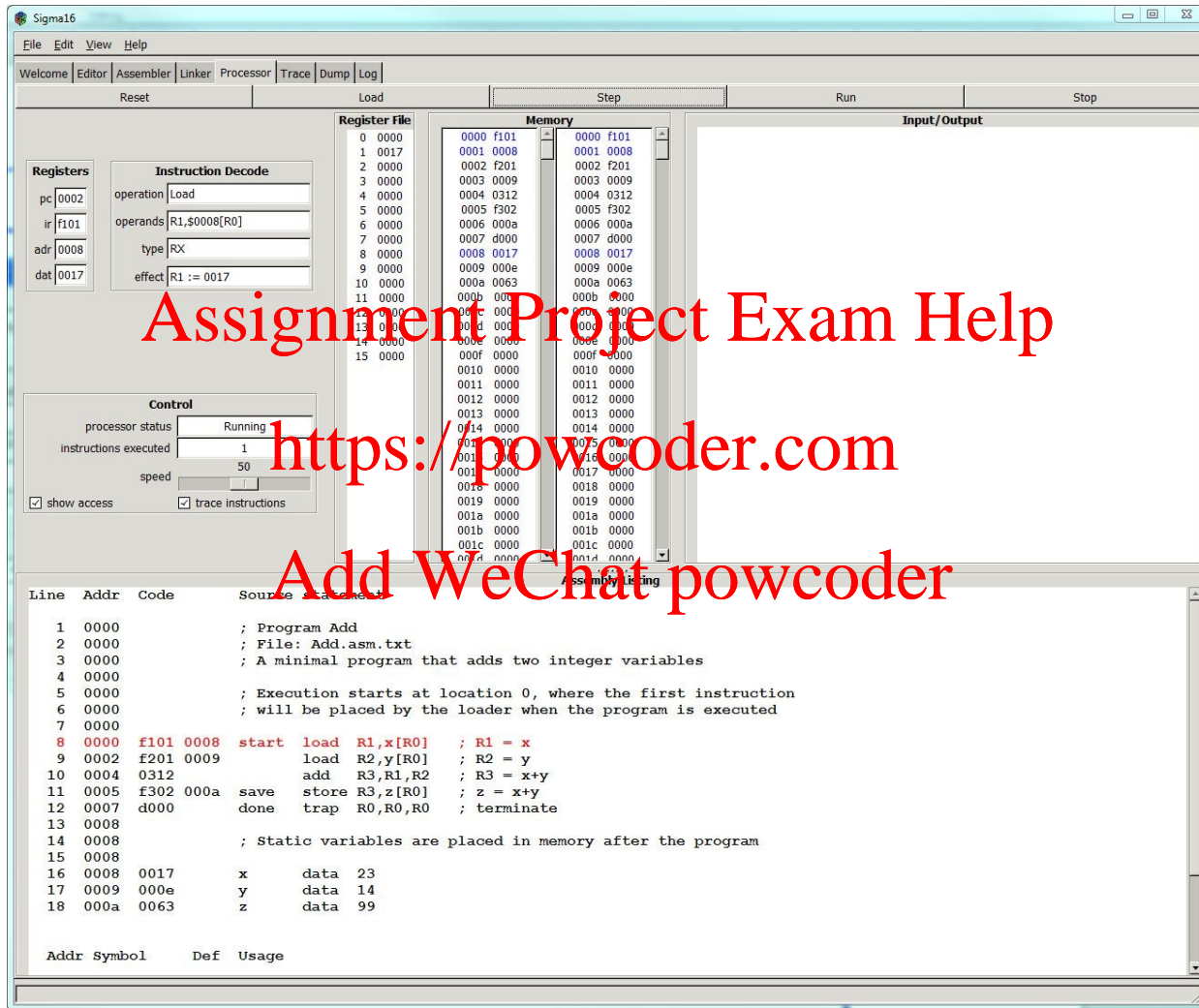
<https://powcoder.com>

Add WeChat powcoder

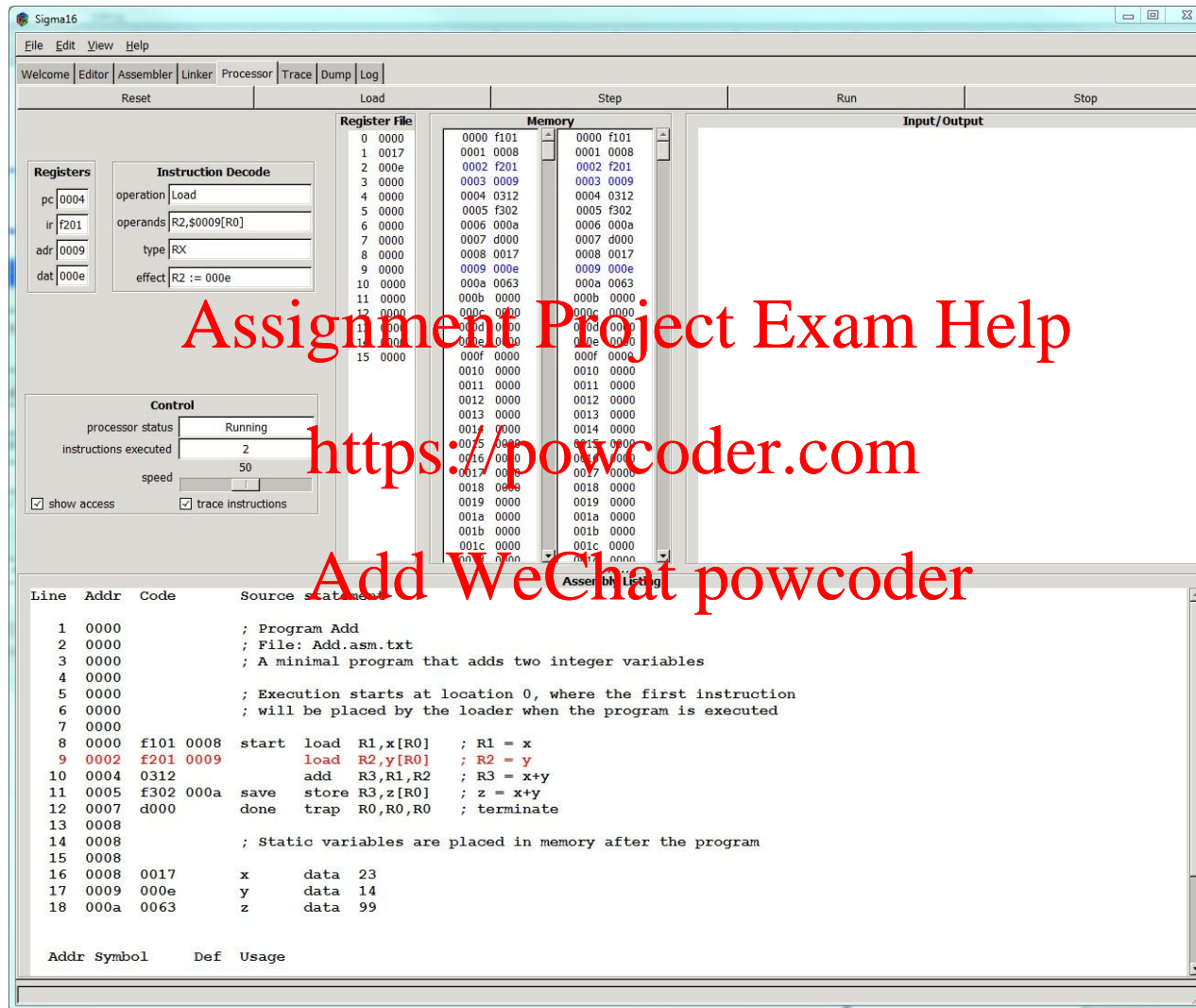
Loading (click “Load”)



Click Step button



Click Step again

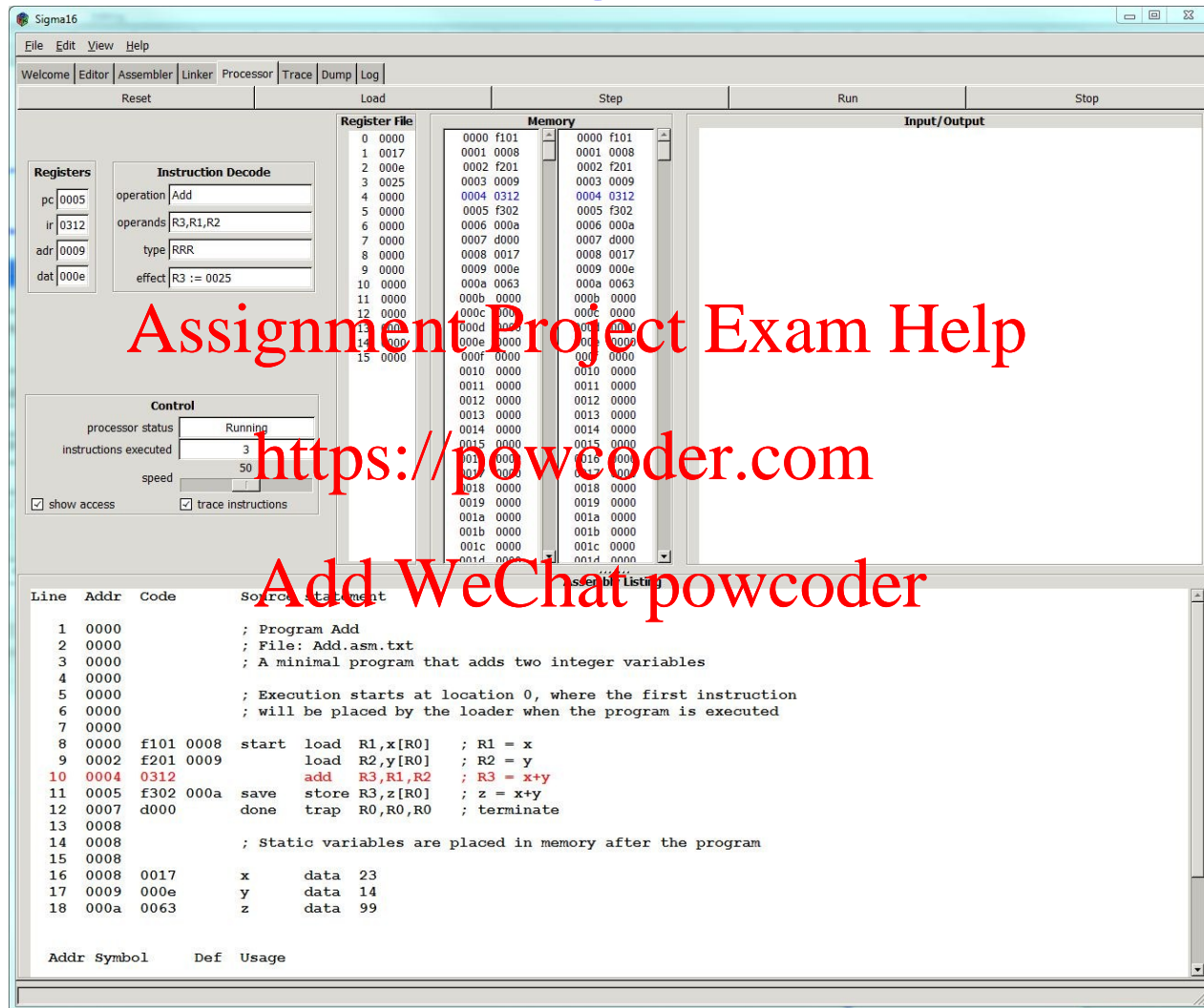


Assignment Project Exam Help

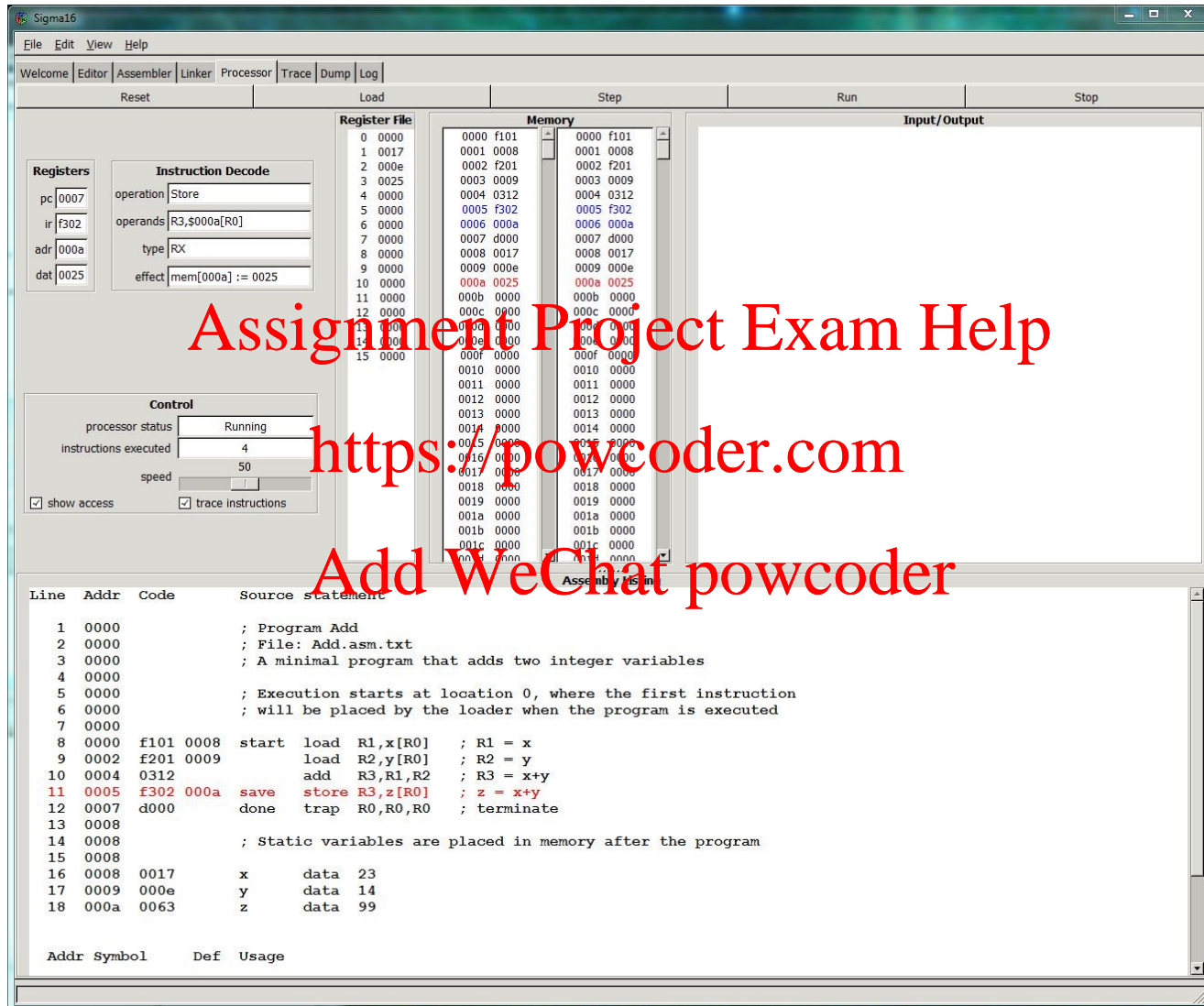
<https://powcoder.com>

Add WeChat powcoder

And again...



Yet again...

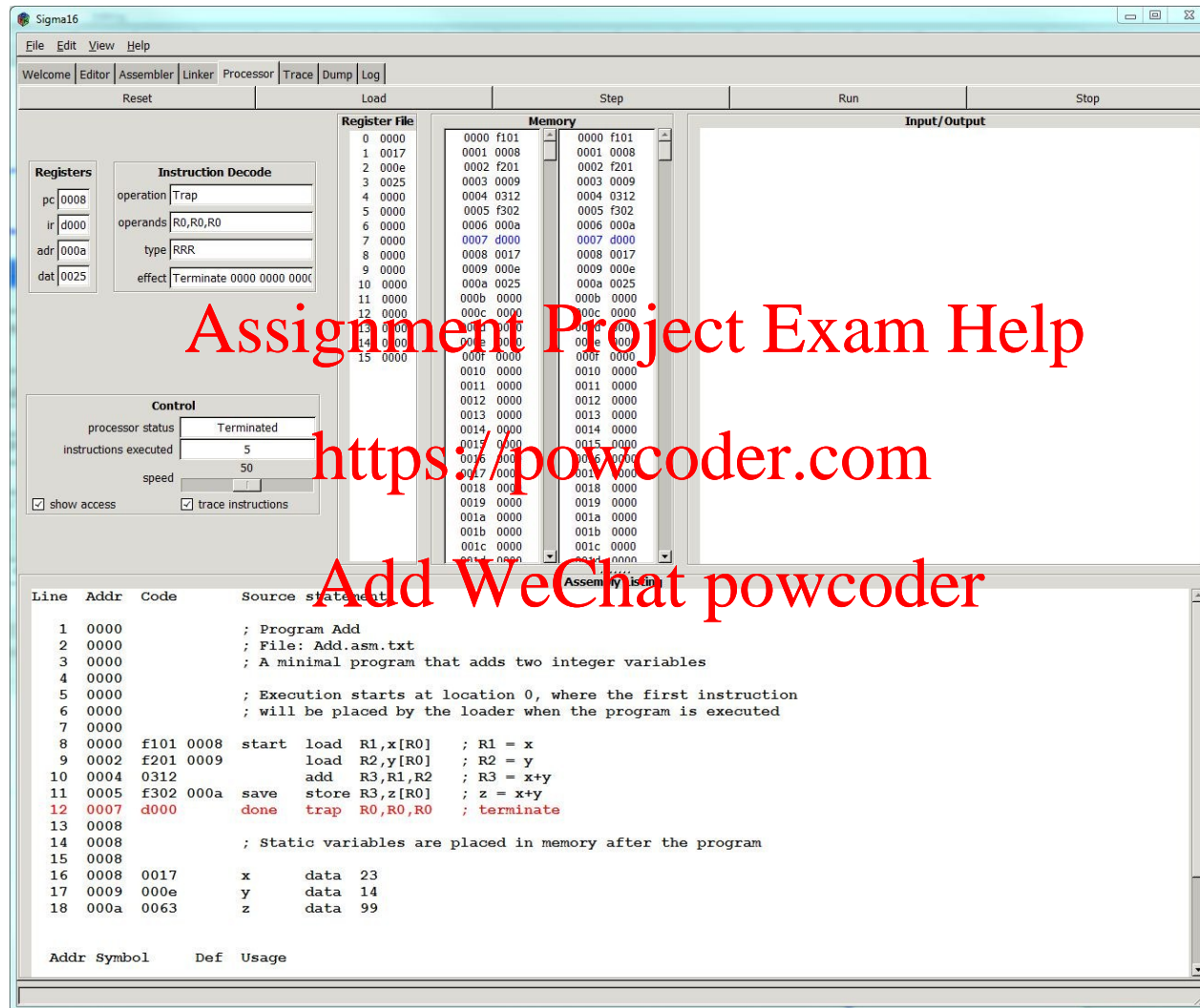


Assignment Project Exam Help

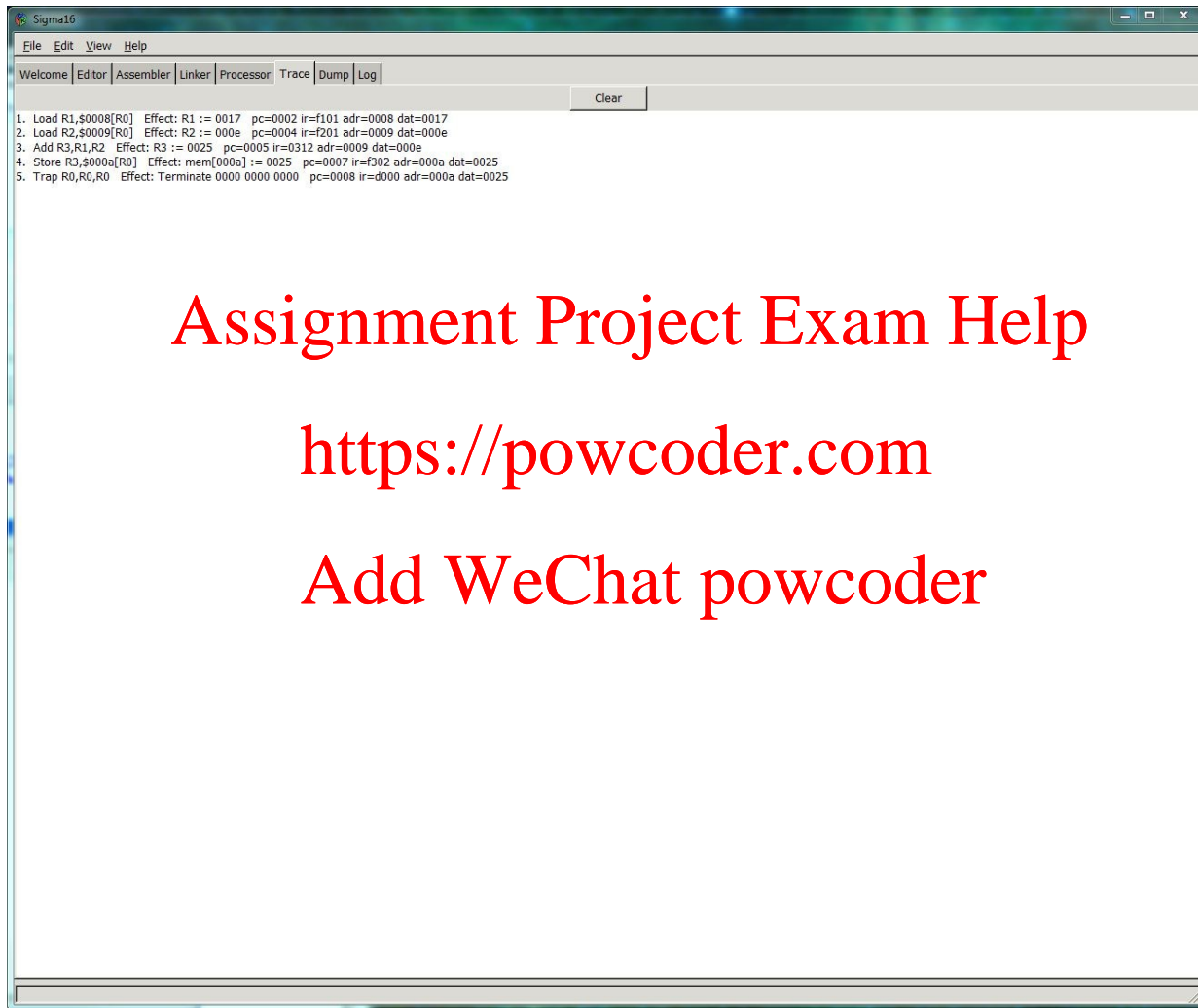
<https://powcoder.com>

Add WeChat powcoder

One last instruction to do...



Look at the Trace page



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

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

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Trace of Machine Instructions

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

<https://powcoder.com>

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

Assignment Project Exam Help

<https://powcoder.com>

Add WhatsApp powder

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

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder