

README.PDF

5/31/25

Getting started - Documentation

Upload LearnRisc.zip

[Web Tool](#) [Led Bar](#) [Getting Started](#) [About Us](#) [Support](#) [Cornell Interpretor](#)

We provide a WEB Tool package for the Risc-V Instruction set. This tools provides users the ability to assemble and execute risc-v code. Risc CPU code is the new standard for machine language. Used by billions of computer chips, and it is one of the easiest instruction set to understand. Learn Risc-V by running the examples provided, and writing new code. Submit your new code to us, we will add several to our package! We are focused on demonstrating simple but interesting examples that in writing teach the language.

Use AI as your instructor! AI can be your expert source of information. It can explain how exch instruction works, write new code you describe and even help in your debugging. It's easier than searching the web, a great way to learn. My favorite is chat-gpt.

To get started, download our Document Package. It is in the form of a zip file that will appear in your Download folder. To install this package move the zip file to your desired destination, highlight it with your file manager and and select "uncompress".

Your computer now has a folder with example code, Use it to practice compiling and running code. From the Web Tool, select a provided example program to run. Click "assemble" and then "run". The results will appear in the "out" window. Modify or write new code as desired!

Web Tool Programming

[LearnRisc](#)

[Documentation](#) [RISC-V Programs](#)

The "LearnRisc" directory serves as the main repository for RISC-V learning resources. Inside, the "RISC-V Programs" folder contains practical code samples, "Documentation" contains manuals and "Getting Started" sheets to help new users understand the tools and options available.

Download your Zip file

[risc folders](#)

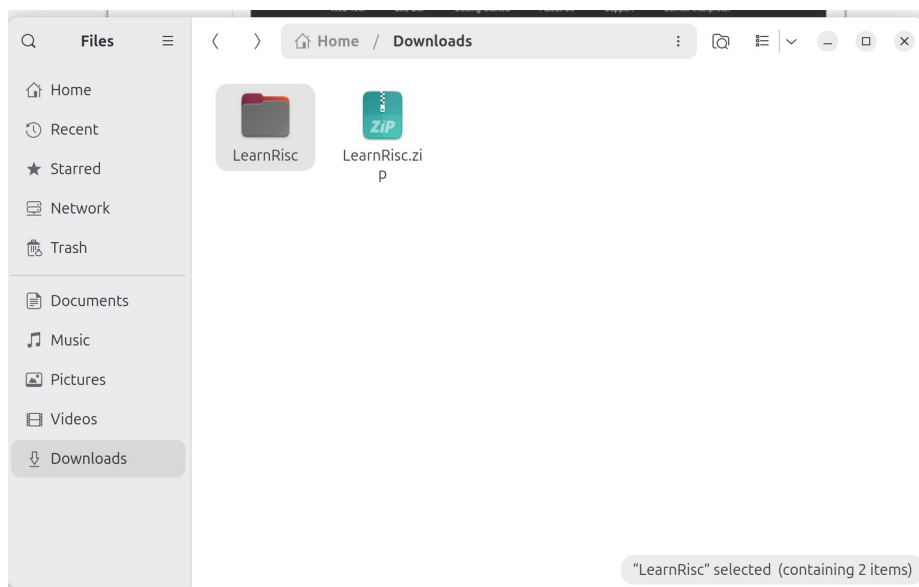
Click on the button above to download the ZIP file to your Downloads directory.

Step 1 – Download Documentation (blue button)

Step 2 - Open file manager to Download folder

Step 3 - Highlight Zip file and select Extract (folder LearnRisc)

You now should have a readme file and a folder of code examples!

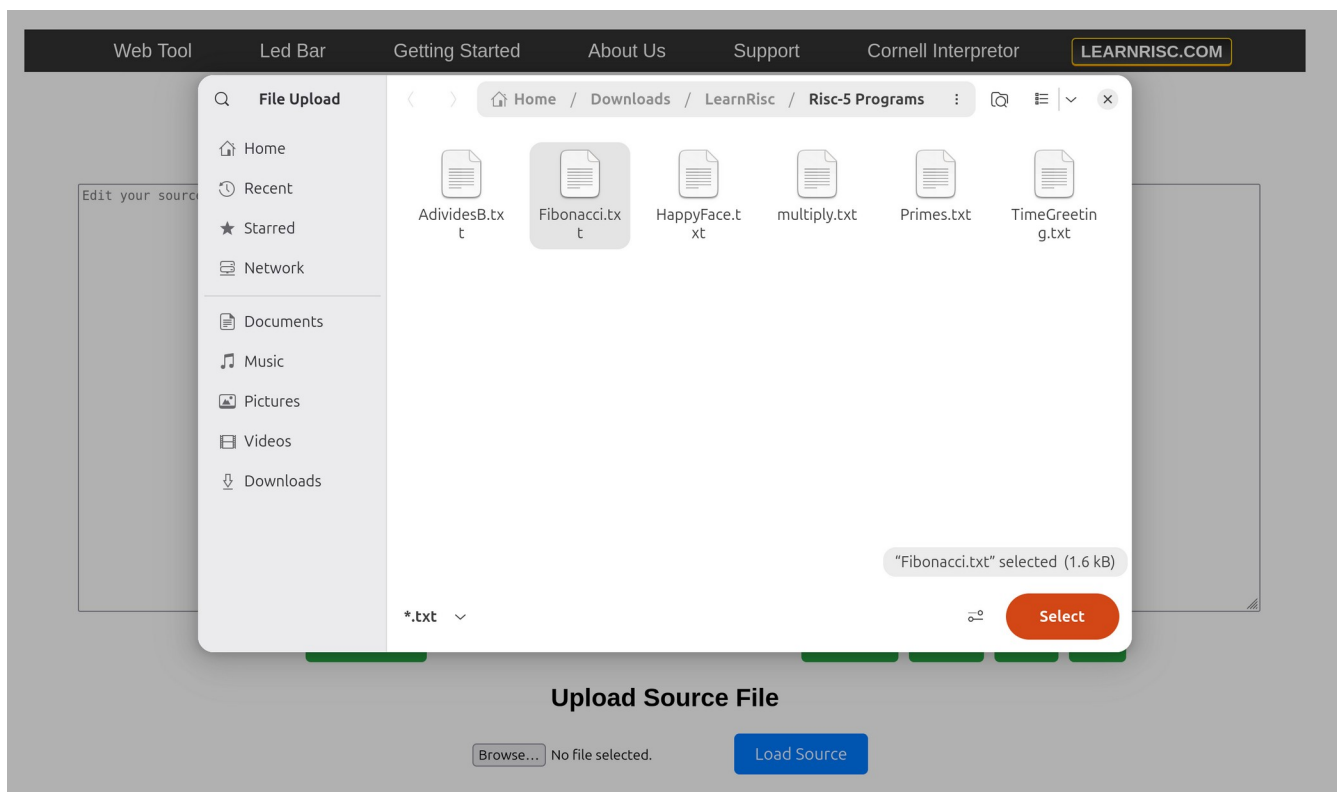


First code example - Fibonacci

A simple loop generates a series of numbers where each number is equal to the sum of the last two numbers.

Step 1 = Load example

To load the example, from the WebTool page: select the browse button, highlight the Fibonacci.txt file, and Press “Select”. Finally press the “Load Source” button.



Step 2: You now can process the source to generate a flow chart, a listing file and a load file. The flowchart is created by characters embedded in the program's comment field!

Once a load file is generated and displayed you can “Run” the program and the results will appear in the Output display.

Source Editor

```
// Title: Fibonacci Series

void main()
{
    // Calculate 18 numbers starting with seed
    int Seed ;
    int Number ;

    Seed = 6;
    Number = 18;

    _Fibonacci(Seed,Number);

    _Exit();
}

/*c-g

    FIBONACCI FLOW CHART

Creation Date- 2/28/2025

This routine calculates a series of numbers. Each new value is
equal to the sum of the previous two numbers.
```

Save Changes

Output Display

```
FIBONACCI FLOW CHART

Creation Date- 2/28/2025

This routine calculates a series of numbers. Each new value is
equal to the sum of the previous two numbers.

The values are placed in memory and sent to a terminal display.

void _Fibonacci(FirstNumber,NumberToCalculate)
{
    last_value = 0
    Memory_Address = 0x2000
    display "Fibonacci Series"
    [Start]->
    [Finish]<-Y-< > NumberToCalculate == 0 ?
    last_value temp = Number
    Number = Number + last_value
```

Flowchart

Listing

Load

Run

Upload Source File

Browse... Fibonacci.txt

Load Source

Source Editor

```
// Title: Fibonacci Series

void main()
{
    // Calculate 18 numbers starting with seed
    int Seed ;
    int Number ;

    Seed = 6;
    Number = 18;

    _Fibonacci(Seed,Number);

    _Exit();
}

/*c-g

    FIBONACCI FLOW CHART

Creation Date- 2/28/2025

This routine calculates a series of numbers. Each new value is
equal to the sum of the previous two numbers.
```

Save Changes

Output Display

```
RISC-v Assembler by RWescott - Version 1.05
Flag = -n

_Setup();
1    jal t0,148          // Call to _Setup()

// Title: Fibonacci Series

void main()
{
2    sw t0,-4(sp)        // Save ret, caller left space

// Calculate 18 numbers starting with seed
int Seed ;
3    addi t1,zero,33
4    sw t1,0(sp)
5    addi sp,sp,4

int Number ;
6    addi t1,zero,33
7    sw t1,0(sp)
8    addi sp,sp,4

Seed = 6;
9    addi zero,zero,0    // nop
10   addi t1,zero,6      // macro- t1 = 6
```

Flowchart

Listing

Load

Run

Upload Source File

Browse... Fibonacci.txt

Load Source

Source Editor

```
// Title: Fibonacci Series

void main()
{
    // Calculate 18 numbers starting with seed
    int Seed ;
    int Number ;

    Seed = 6;
    Number = 18;

    _Fibonacci(Seed,Number);
    _Exit();
}

/*c-g

    FIBONACCI FLOW CHART

Creation Date- 2/28/2025

This routine calculates a series of numbers. Each new value is
equal to the sum of the previous two numbers.
```

Save Changes

Output Display

```
jal t0,148
sw t0,-4(sp)
addi t1,zero,33
sw t1,0(sp)
addi sp,sp,4
addi t1,zero,33
sw t1,0(sp)
addi sp,sp,4
addi zero,zero,0
addi t1,zero,6
sw t1,-8(sp)
addi zero,zero,0
addi t1,zero,18
sw t1,-4(sp)
lw t1,-8(sp)
lw t2,-4(sp)
jal t0,20
jal t0,92
addi sp,sp,-12
lw t0,0(sp)
jalr zero,t0,0
add x28,zero,zero
lui x30, 0x2
addi zero,x31,0
beq t2,zero,36
addi x27,t1,0
add t1,t1,x28
sw t1,0(x30)
```

Flowchart

Listing

Load

Run

Upload Source File

Browse... Fibonacci.txt

Load Source

Source Editor

```
// Title: Fibonacci Series

void main()
{
    // Calculate 18 numbers starting with seed
    int Seed ;
    int Number ;

    Seed = 6;
    Number = 18;

    _Fibonacci(Seed,Number);
    _Exit();
}

/*c-g

    FIBONACCI FLOW CHART

Creation Date- 2/28/2025

This routine calculates a series of numbers. Each new value is
equal to the sum of the previous two numbers.
```

Save Changes

Output Display

```
Fibonacci Series
6
12
18
30
48
78
126
204
330
534
864
1398
2262
3660
5922
9582
15504
25086
Complete

Program Complete
```

Flowchart

Listing

Load

Run

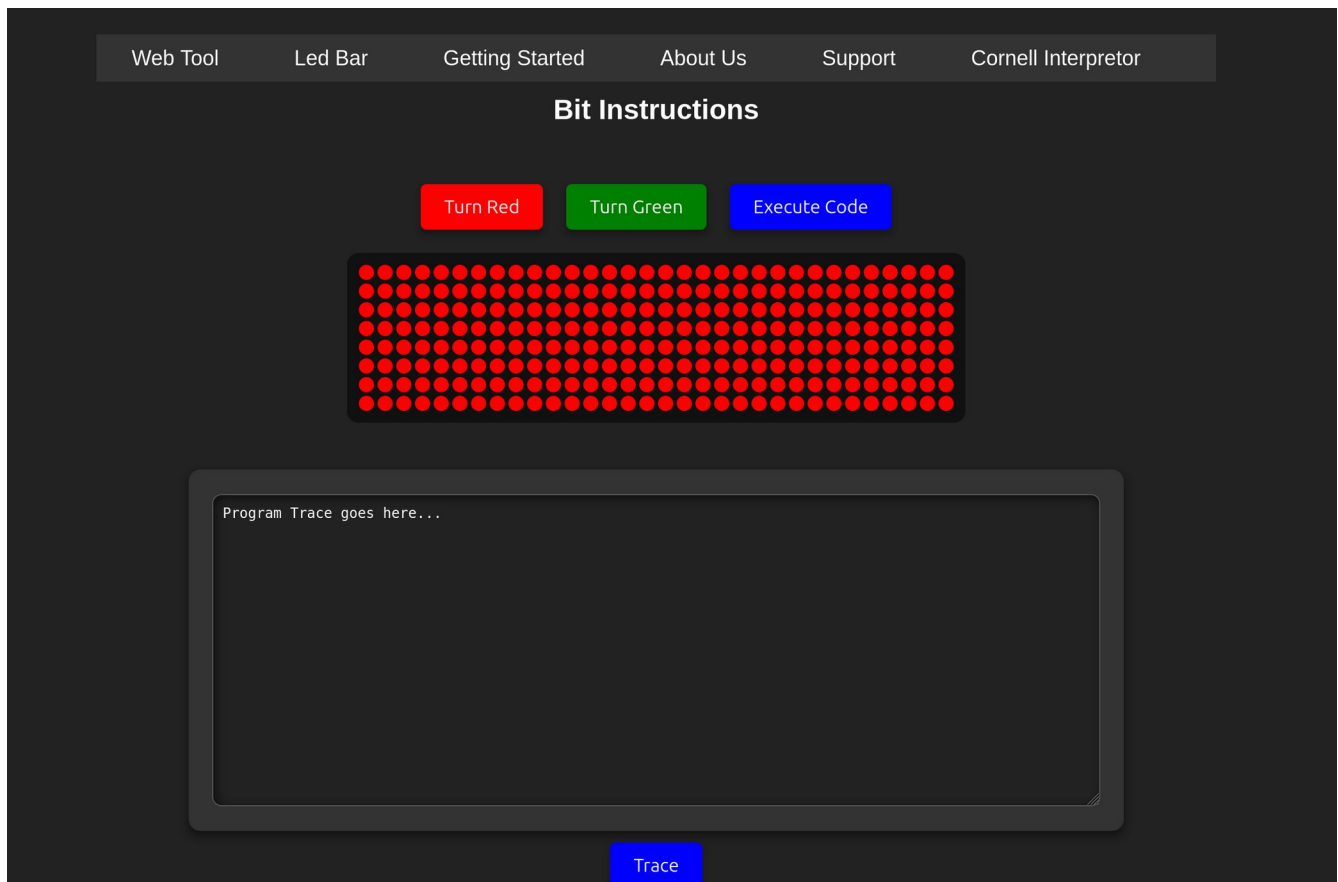
Upload Source File

Browse... Fibonacci.txt

Load Source

Bit Operation are supported by the LED Bar Page. The program is compiled as before. After a load file has been generated, switch to the Led bar page to run the code. On this page, register x23 to x31 are mapped a led display! A special “IO:LEDS” instruction writes to the display.

For viewing an delay is inserted when the LEDS are written to.



Flowchart Quick sheet

Overview:

Comments in C programs and many others support two type of comments. The first is the `//` , these double slash characters designate all character that follow in the same line as comment characters that should not be compiled or processed. The second type is the `/* */` pair implies all lines between as comments.

```
/*  
This is a  
multi-line comment  
*/
```

The program **flow** program generates a flowchart from characters embedded in your program files and defined by control characters such as `//c-x` that the flow program interpreters, but the assembler/compilers ignore. For multi-line comments `/*c-g` at the start of a comment implies the text between it and the `*/` to follow should be included in the flowchart usually as a summary of the code.

Embedded into the comments of a program. The FlowChart output is generated in a txt file named FlowChart.txt.

Here are the options:

```
//s- void Computer::PrintCout(unsigned offset)  
-----  
void Computer::PrintCout(unsigned offset)  
|  
//s-x void Computer::PrintCout(unsigned offset)  
-----  
void Computer::PrintCout(unsigned offset)  
//c- This is a comment!  
-----
```

Output:

```
// This is a comment!  
//c-3 This is a two comment,blank lines added above  
-----  
//c- Second comment//e-b entry point b  
-----
```

Output:

```
[b]->|  
|  
//g-d goto point d  
-----  
[d] ← o  
//Y-b Is the sub red?
```

```
-----  
[b]<-N-< > Is the sun red?  
|Y  
//N-c Is the sub red?
```

```
-----  
[c]<-Y-< > Is the sun red?  
|N  
//x- exit
```

```
-----  
o exit  
/*c-g to start multiline comments, */ to end it
```

```
-----  
this is  
a multi  
line comment
```