

# **A RISC-V Processor Components CircuiTikZ Library**

Niklas Ladurner

December 12, 2025

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Motivation . . . . .	3
1.2	Usage . . . . .	3
<b>2</b>	<b>Component List</b>	<b>4</b>
2.1	Memory Components . . . . .	4
2.2	Arithmetic Components . . . . .	4
2.3	Multiplexers . . . . .	5
2.4	Control Units . . . . .	5
2.5	Miscellaneous Components . . . . .	6
<b>3</b>	<b>Keys</b>	<b>8</b>
3.1	CircuitikZ keys . . . . .	8
3.2	Special node keys . . . . .	8
<b>4</b>	<b>Examples</b>	<b>10</b>
4.1	Single-Cycle RISC-V Processor . . . . .	10
4.2	Single-Cycle RISC-V Processor (with Branch Logic) . . . . .	10
4.3	Multi-Cycle RISC-V Processor . . . . .	11
4.4	Pipelined RISC-V Processor . . . . .	12
4.5	Pipelined RISC-V Processor with Hazard Unit . . . . .	12

# 1 Introduction

## 1.1 Motivation

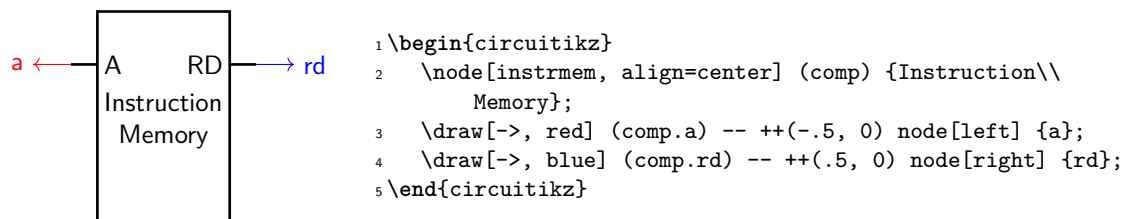
This CircuiTikZ library offers some components to efficiently draw RISC-V processors in  $\LaTeX$ . The library was designed with the goal of resembling the RISC-V processor schematics as presented in ‘Digital Design and Computer Architecture: RISC-V Edition’ by Sarah L. Harris and David Harris.

## 1.2 Usage

To use the predefined components, you must include the library `riscvproc`. Your preamble should look like this:

```
...  
\usepackage{tikz}  
\usepackage{circuitikz}  
\usetikzlibrary{riscvproc}  
...
```

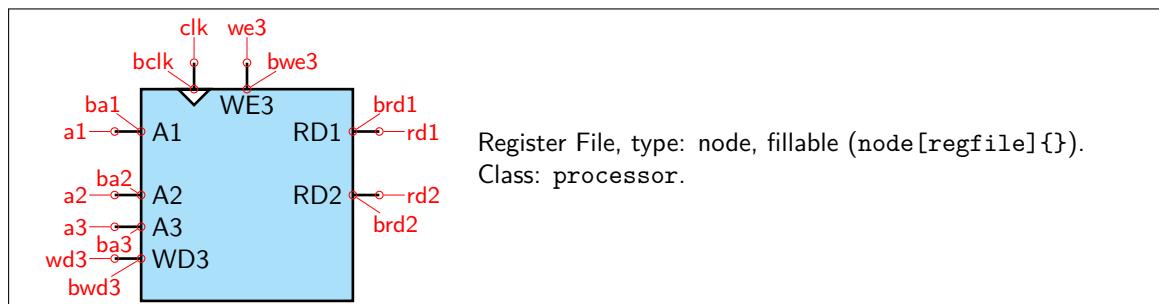
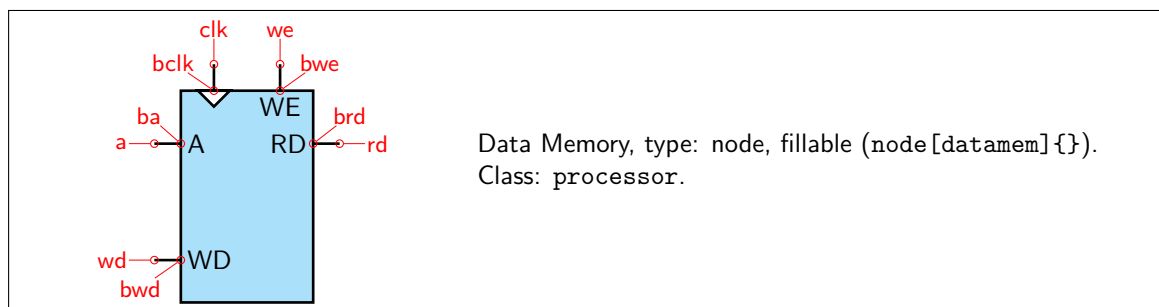
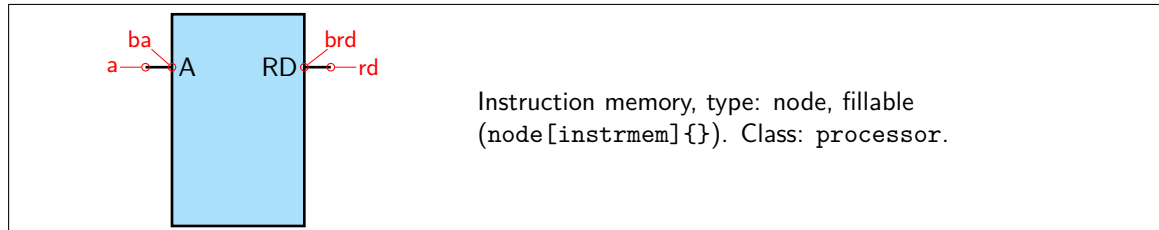
Components are then available in `circuitikz` environments:



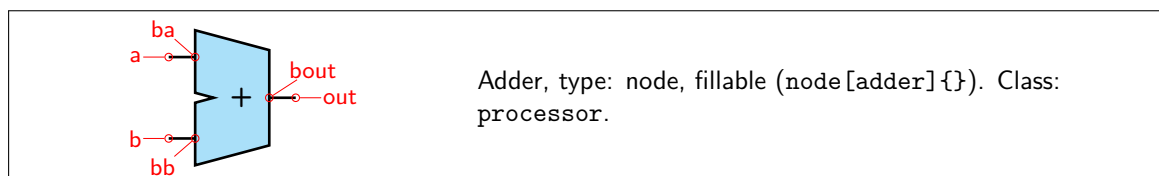
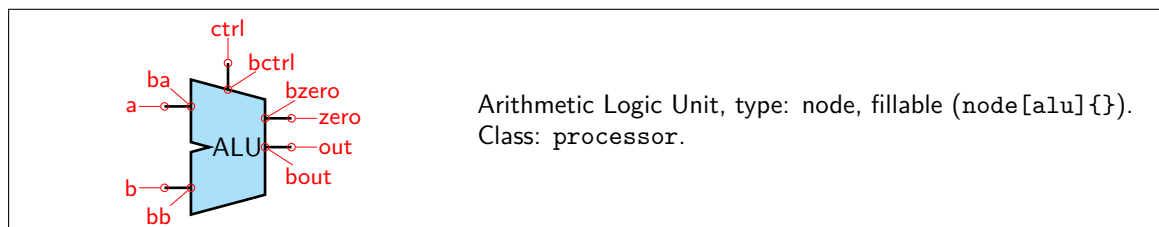
```
1\begin{circuitikz}  
2  \node[instrmem, align=center] (comp) {Instruction\\  
    Memory};  
3  \draw[->, red] (comp.a) -- ++(-.5, 0) node[left] {a};  
4  \draw[->, blue] (comp.rd) -- ++(.5, 0) node[right] {rd};  
5\end{circuitikz}
```

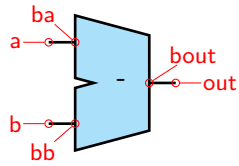
## 2 Component List

### 2.1 Memory Components



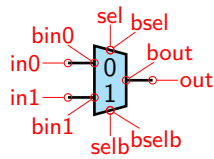
### 2.2 Arithmetic Components



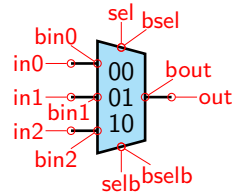


Subtractor, type: node, fillable (node[subtr]{}). Class: processor.

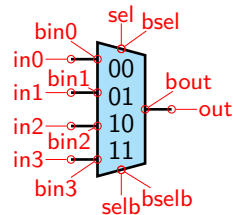
## 2.3 Multiplexers



Multiplexer, type: node, fillable (node[mux]{}). Class: processor.

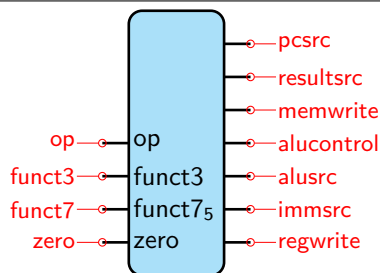


Multiplexer with 3 inputs, type: node, fillable (node[3mux]{}). Class: processor.

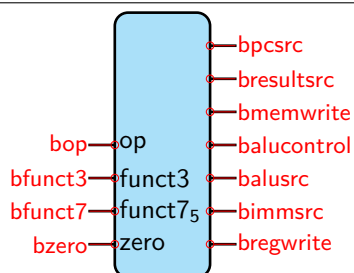


Multiplexer with 4 inputs, type: node, fillable (node[4mux]{}). Class: processor.

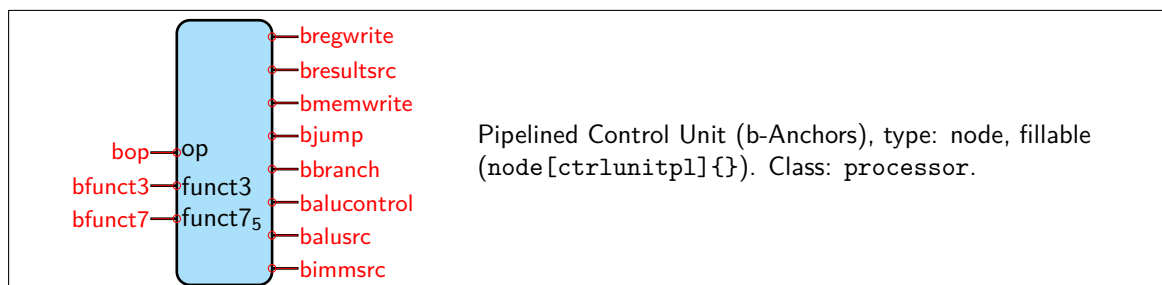
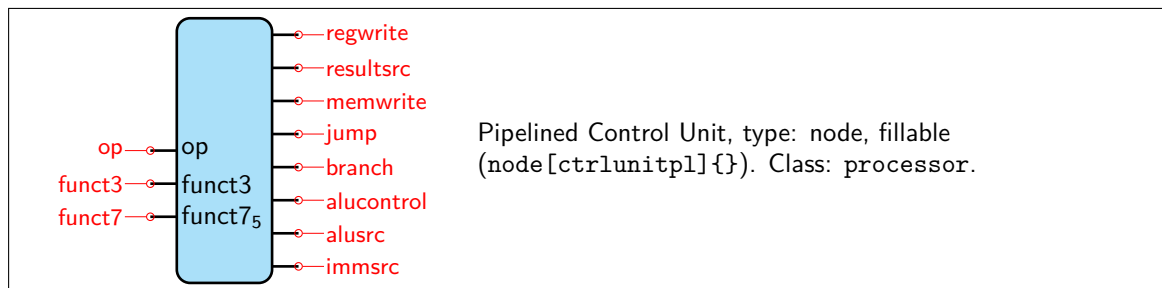
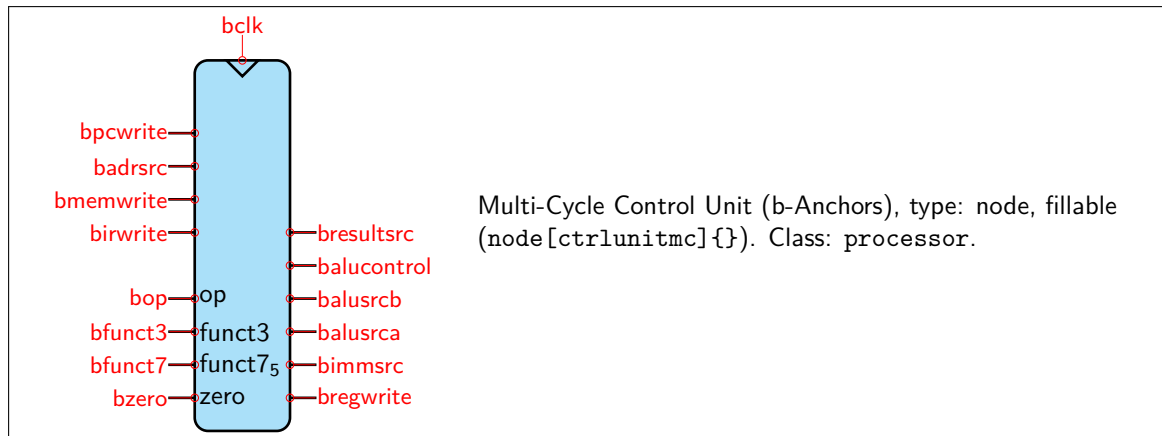
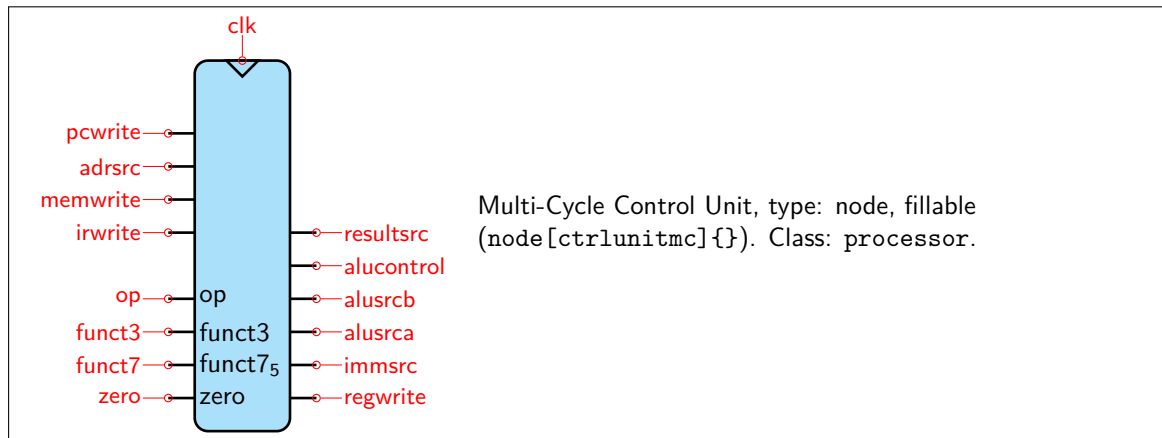
## 2.4 Control Units



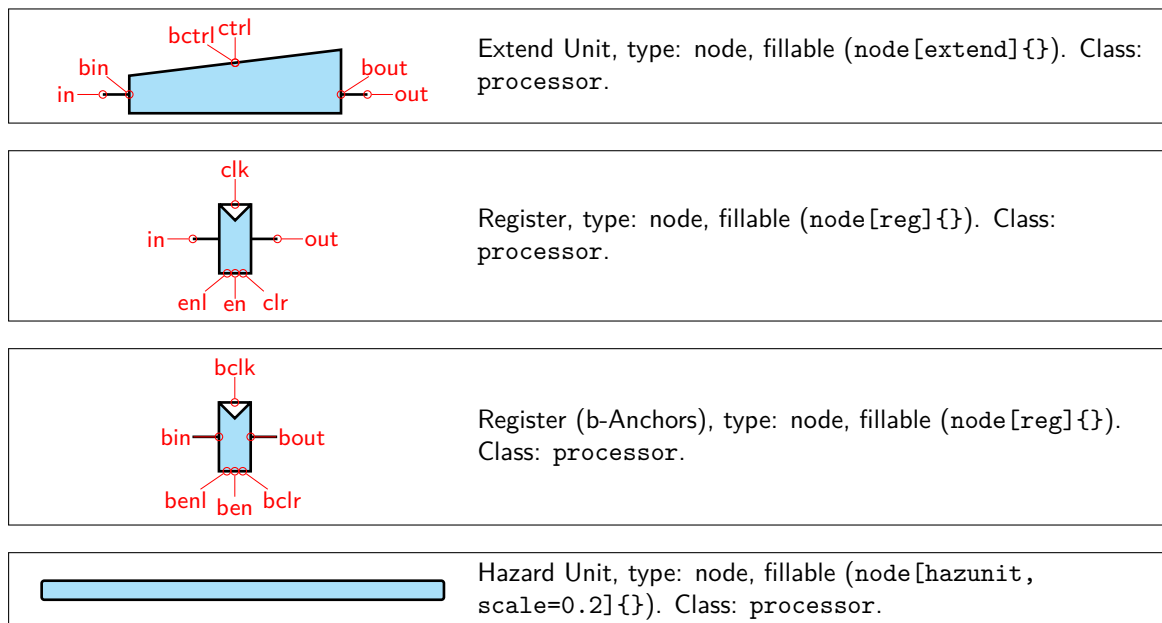
Single-Cycle Control Unit, type: node, fillable (node[ctrlunitsc]{}). Class: processor.



Single-Cycle Control Unit (b-Anchors), type: node, fillable (node[ctrlunitsc]{}). Class: processor.



## 2.5 Miscellaneous Components



## 3 Keys

### 3.1 CircuiTikZ keys

The desired CircuiTikZ key can be set via `\ctikzset{processor/<key>=value}`. E.g. if one wishes to set the line width of all components to 4, the line `\ctikzset{processor/thickness=4}` would have to be included in the specific circuitikz picture. A list of all CircuiTikZ keys can be found in Table 1. A list of component families can be found in Table 2.

Key	Description	Default value
<code>scale</code>	Sets scale for all processor components.	1
<code>thickness</code>	Sets line width for all processor components.	1
<code>leadthickness</code>	Sets line width for all leads.	1
<code>font</code>	Sets font family for all labels of processor components.	<code>\rmfamily</code>
<code>memory/height</code>	Sets height for all memory components.	2
<code>memory/width</code>	Sets width for all memory components except <code>regfile</code> .	1.25
<code>control/heightsc</code>	Sets height for <code>ctrlunitsc</code> .	2.5
<code>control/heightmc</code>	Sets height for <code>ctrlunitmc</code> .	3.5
<code>control/width</code>	Sets width for control components.	0.9
<code>control/radius</code>	Sets border radius for control components.	5
<code>arith/height</code>	Sets height for arithmetic components.	0.9
<code>arith/width</code>	Sets height for arithmetic components.	0.7
<code>arith/slope</code>	Sets slope for arithmetic components in degrees.	15
<code>extend/height</code>	Sets height for big side of extend components.	0.6
<code>extend/width</code>	Sets height for extend components.	2
<code>extend/slope</code>	Sets slope for extend components in degrees.	7
<code>mux/slope</code>	Sets slope for multiplexers in degrees.	15
<code>misc/smallheight</code>	Sets height for small components.	0.65
<code>misc/smallwidth</code>	Sets width for small components. Also affects the CLK input triangle.	0.3
<code>misc/leadlen</code>	Sets length for input and output leads.	0.25
<code>hazard/height</code>	Sets height for <code>hazunit</code> .	0.9
<code>hazard/width</code>	Sets width for <code>hazunit</code> .	18
<code>hazard/radius</code>	Sets border radius for <code>hazunit</code> .	5

Table 1: List of CircuiTikZ keys

### 3.2 Special node keys

Some keys are also defined as Tikz keys and can therefore be directly passed to nodes likes shown in Figure 1. A list of all these keys can be found in Table 3.

More keys might be added in future.



Component family	Component list
memory components	instrmem, datamem, regfile
control components	ctrlunitsc, ctrlunitmc
arithmetic components	alu, add, subtr
extend components	extend
small components	mux, reg

Table 2: List of component families

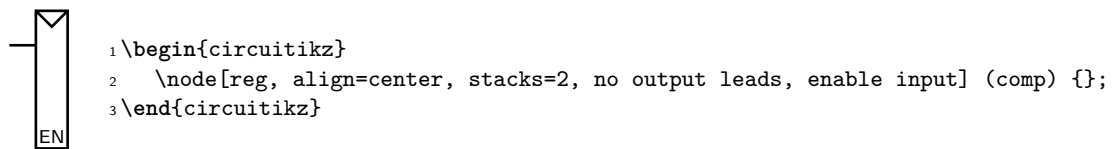


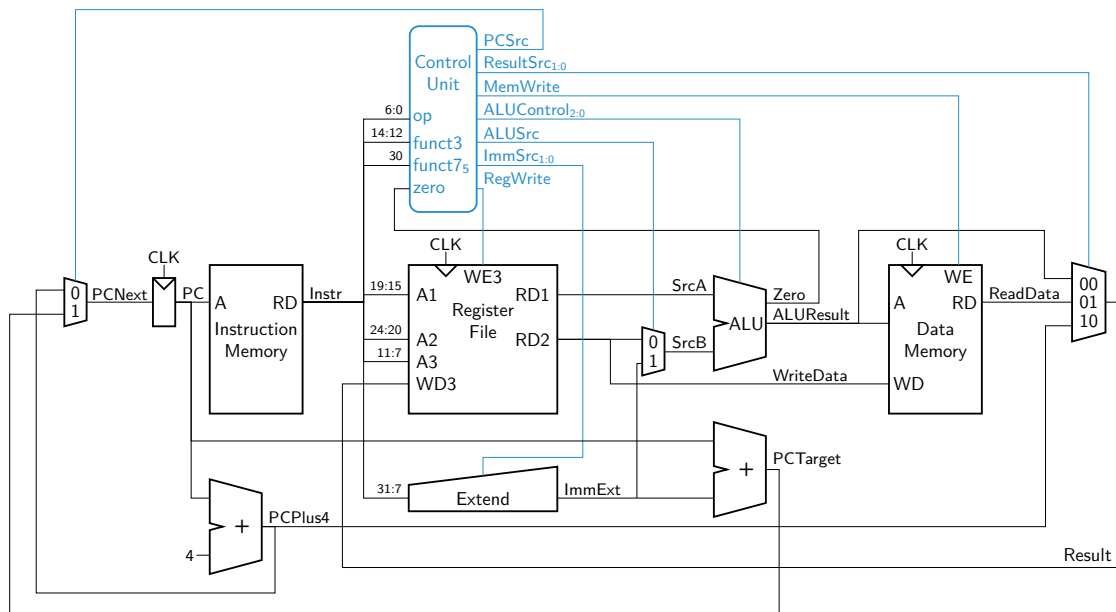
Figure 1: Passing options to a node

Key	Description	applicable to
input leads	Specifies whether to draw input leads.	all components
output leads	Specifies whether to draw output leads.	all components
leads	Specifies whether to draw leads at all.	all components
stacks	Sets height of a register in multiples of the default height, allows for stretched registers.	reg
enable input	Specifies whether to draw an enable input or not. This also gives two new anchors, en and ben.	reg
clear input	Specifies whether to draw a clear input or not. This also gives two new anchors, clr and bclr. For Usage of enable and clear inputs, use the en1 and ben1 anchors.	reg
clock	Specifies whether to draw a clk input on a component that supports it.	all timed components

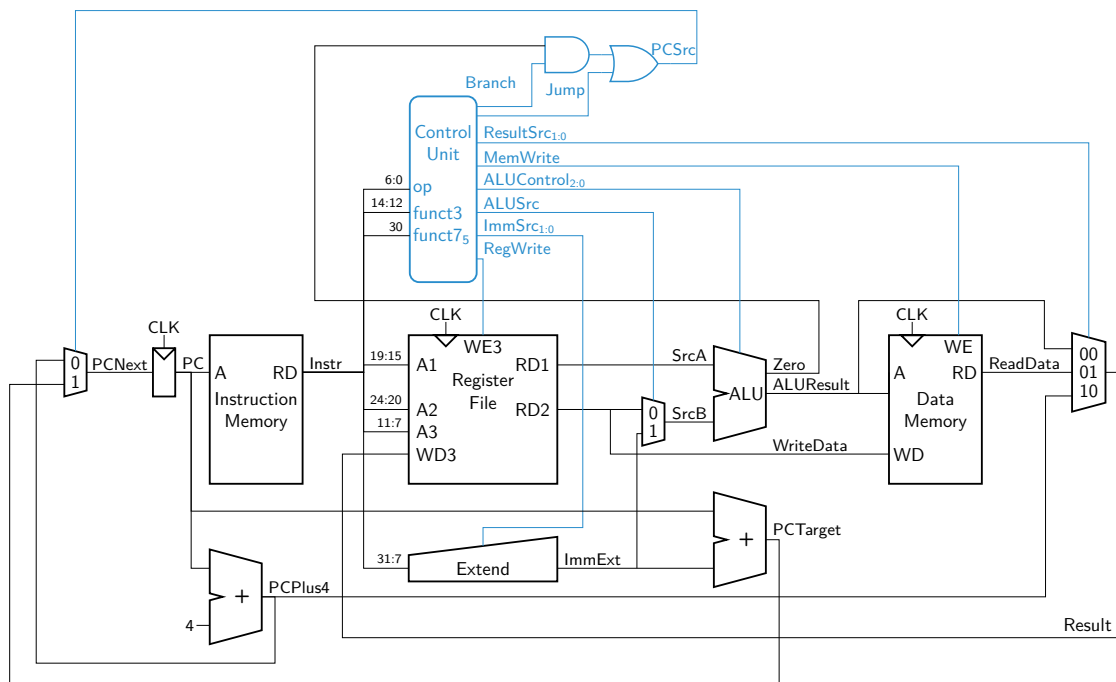
Table 3: List special node keys

## 4 Examples

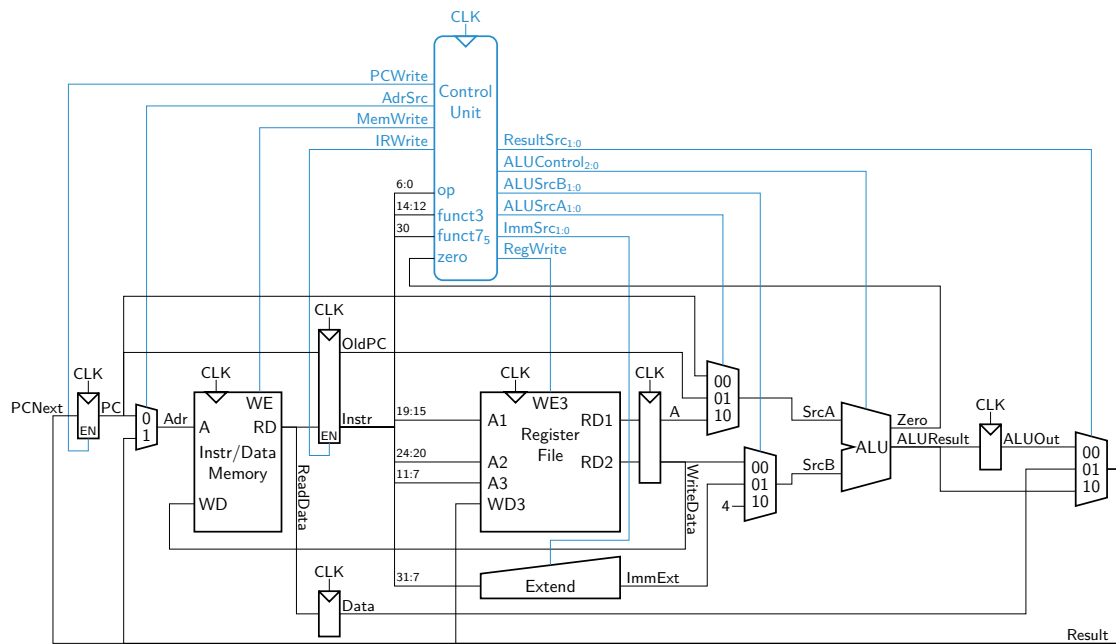
### 4.1 Single-Cycle RISC-V Processor



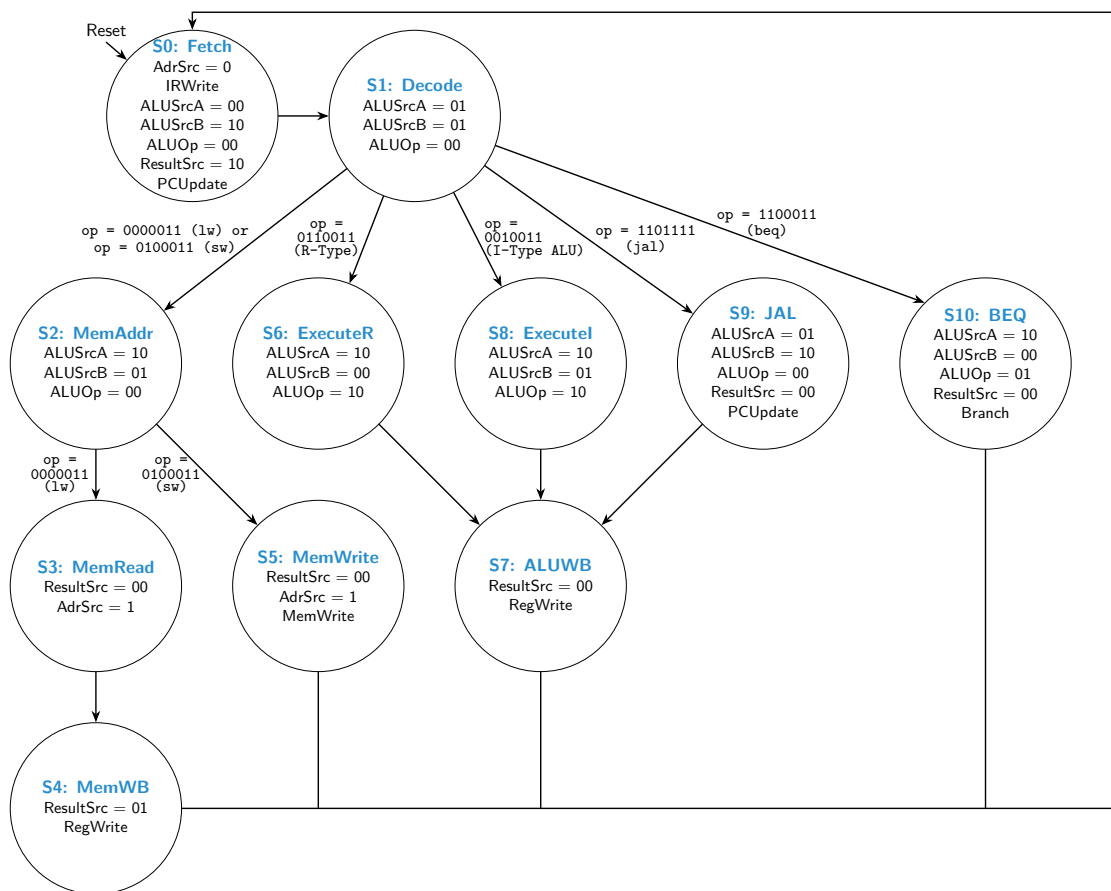
### 4.2 Single-Cycle RISC-V Processor (with Branch Logic)



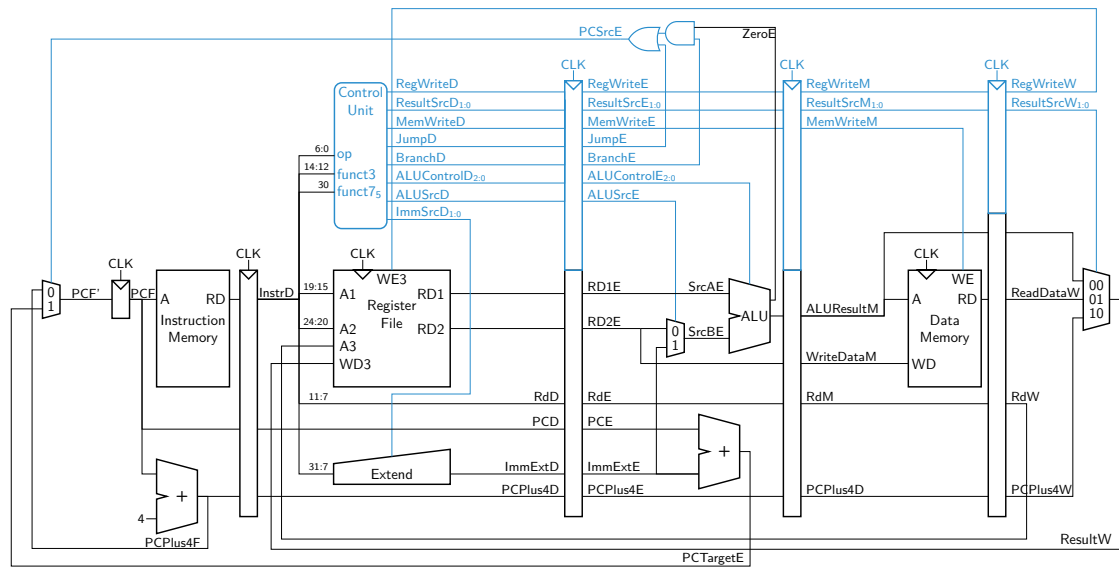
### 4.3 Multi-Cycle RISC-V Processor



Automaton:



## 4.4 Pipelined RISC-V Processor



## 4.5 Pipelined RISC-V Processor with Hazard Unit

