A RISC-V Processor Components CircuiTi*k*Z Library

Niklas Ladurner

March 28, 2025

Contents

1 Introduction

1.1 Motivation

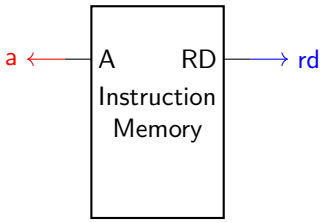
This CircuiTi*k*Z library offers some components to efficiently draw RISC-V processors in LaTeX. The library was designed with the goal of resembling the RISC-V processor schematics as presented in 'Digital Design and Computer Architecture: RISC-V Edition' by Sarah L. Harris and David Harris.

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}
...
```

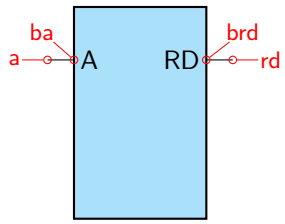
Components are then availiable in `circuitikz` environments:



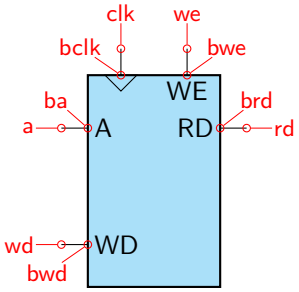

```
\begin{circuitikz}
  \node[instrmem, align=center] (comp) {Instruction\\
      Memory};
  \draw[->, red] (comp.a) -- ++(-.5, 0) node[left] {a};
  \draw[->, blue] (comp.rd) -- ++(.5, 0) node[right] {rd};
\end{circuitikz}
```

2 Component List

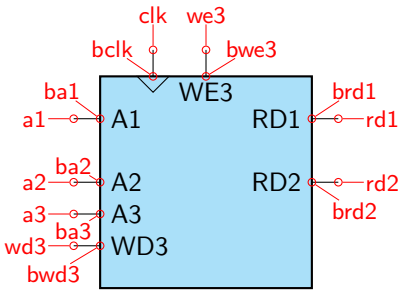
2.1 Memory Components



Instruction memory, type: node, fillable (`node[instrmem]{}`). Class: `processor`.

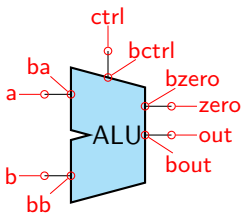


Data Memory, type: node, fillable (`node[datamem]{}`). Class: `processor`.

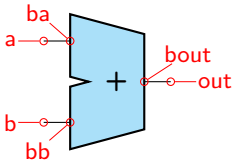


Register File, type: node, fillable (`node[regfile]{}`). Class: `processor`.

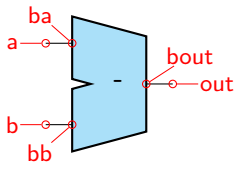
2.2 Arithmetic Components



Arithmetic Logic Unit, type: node, fillable (`node[alu]{}`). Class: `processor`.

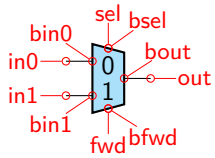


Adder, type: node, fillable (`node[adder]{}`). Class: `processor`.

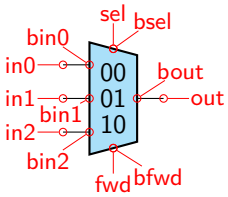


Subtractor, type: node, fillable (`node[subtr]{}`). Class: `processor`.

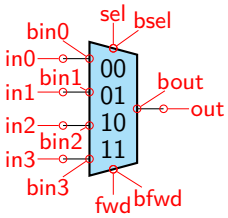
2.3 Multiplexers



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

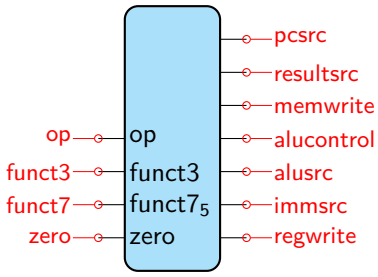


Multiplexer with 3 inputs, type: node, fillable (`node[3mux]{}`). Class: `processor`.

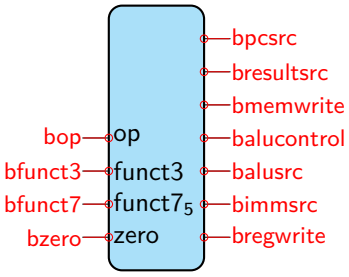


Multiplexer with 4 inputs, type: node, fillable (`node[4mux]{}`). Class: `processor`.

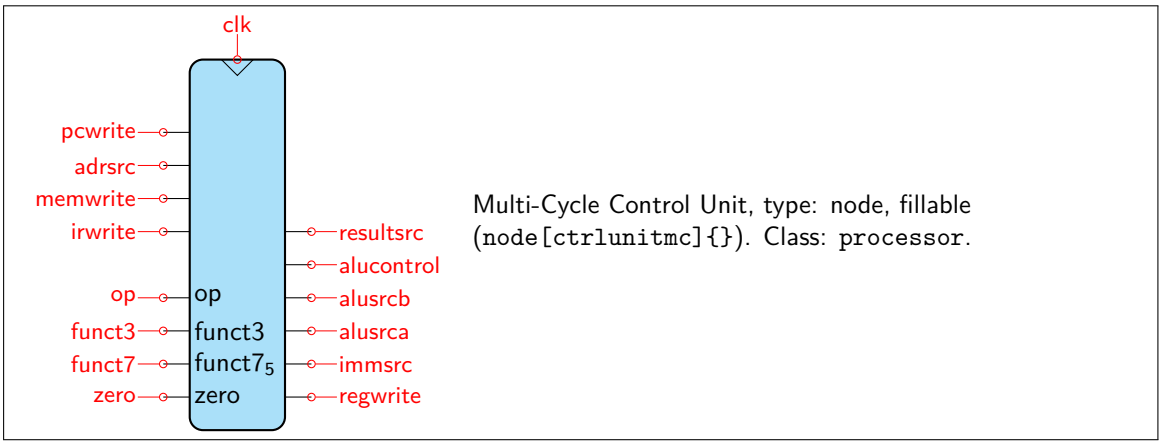
2.4 Control Units



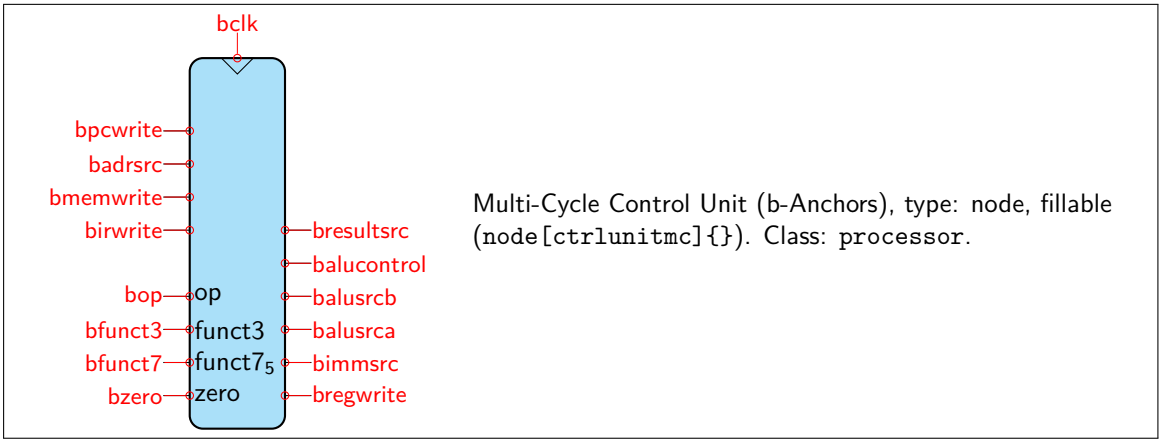
Single-Cycle Control Unit, type: node, fillable (`node[ctrlunitsc]{}`). Class: `processor`.



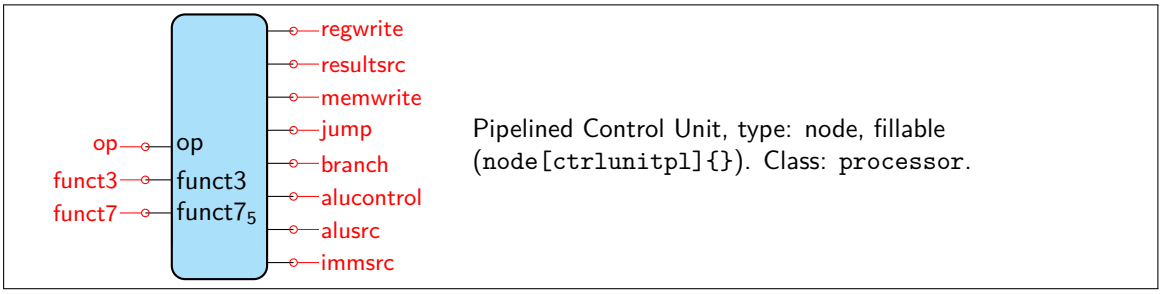
Single-Cycle Control Unit (b-Anchors), type: node, fillable (`node[ctrlunitsc]{}`). Class: `processor`.



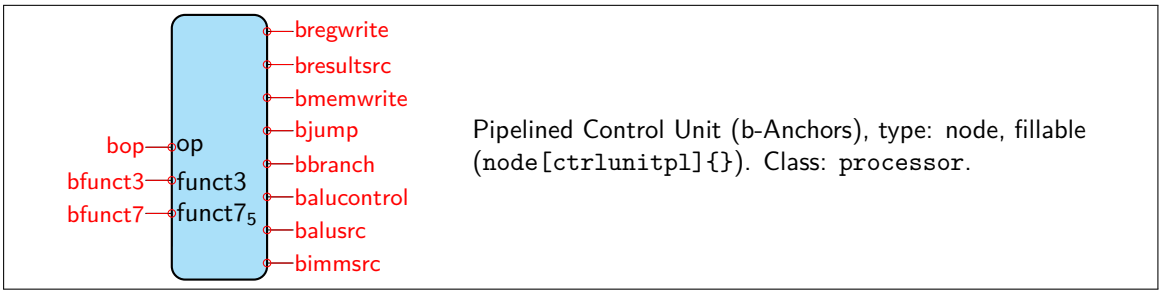
Multi-Cycle Control Unit, type: node, fillable (`node[ctrlunitmc]{}`). Class: `processor`.



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

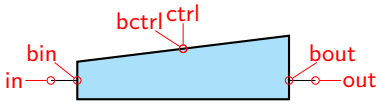


Pipelined Control Unit, type: node, fillable (`node[ctrlunitpl]{}`). Class: `processor`.

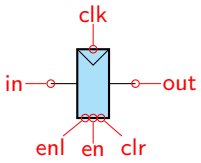


Pipelined Control Unit (b-Anchors), type: node, fillable (`node[ctrlunitpl]{}`). Class: `processor`.

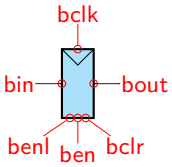
2.5 Miscellaneous Components



Extend Unit, type: node, fillable (`node[extend]{}`). Class: `processor`.



Register, type: node, fillable (`node[reg]{}`). Class: `processor`.



Register (b-Anchors), type: node, fillable (`node[reg]{}`). Class: `processor`.

Hazard Unit, type: node, fillable (`node[hazunit, scale=0.2]{}`). Class: `processor`.

3 Keys

3.1 CircuiTi*k*Z keys

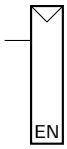
The desired CircuiTi*k*Z key can be set via `\ctikzset{processor/<key>=value}`. E.g. if one whishes to set the line width of all components to 4, the line `\ctikzset{processor/thickness=4}` would have to be included in the specific circuitikz picture. A list of all CircuiTi*k*Z keys can be found in Table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

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

Table 1: List of CircuiTi*k*Z keys

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




```
\begin{circuitikz}
  \node[reg, align=center, stacks=2, no output leads, enable input] (comp) {};
\end{circuitikz}
```

Figure 1: Passing options to a node

Key	Description	applicable to
`input leads`	Specifies wether to draw input leads.	all components
`output leads`	Specifies wether to draw output leads.	all components
`leads`	Specifies wether 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 wether to draw an enable input or not. This also gives two new anchors, `en` and `ben`.	`reg`
`clear input`	Specifies wether 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 `enl` and `benl` anchors.	`reg`
`clock`	Specifies wether to draw a clk input on a component that supports it.	all timed components

Table 3: List special node keys

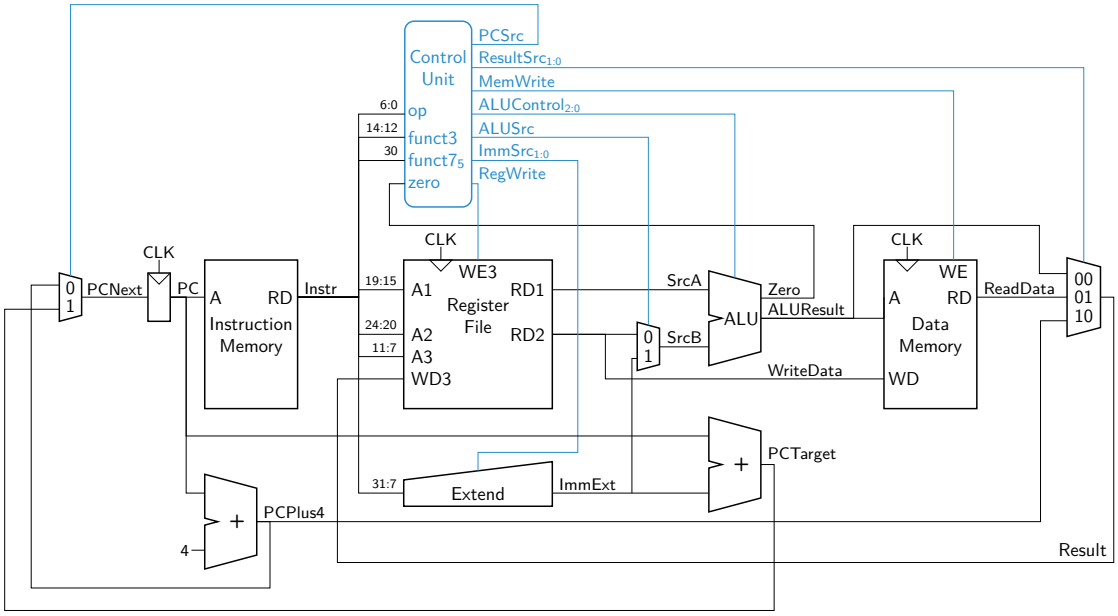
3.2 Special node keys

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.

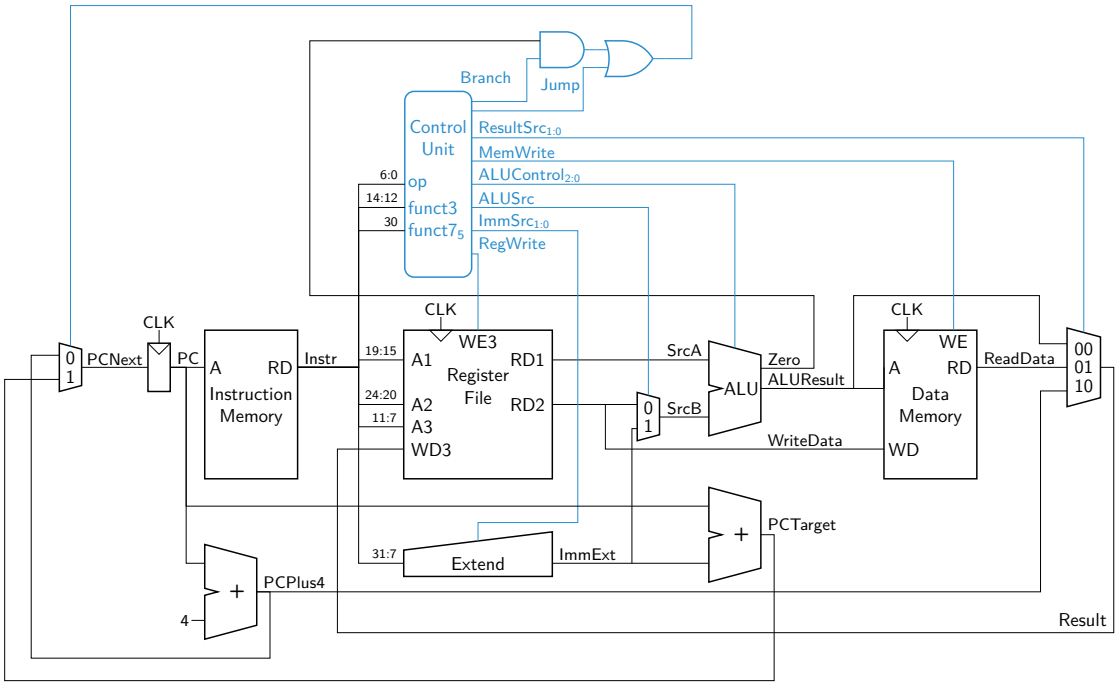
More keys might be added in future.

4 Examples

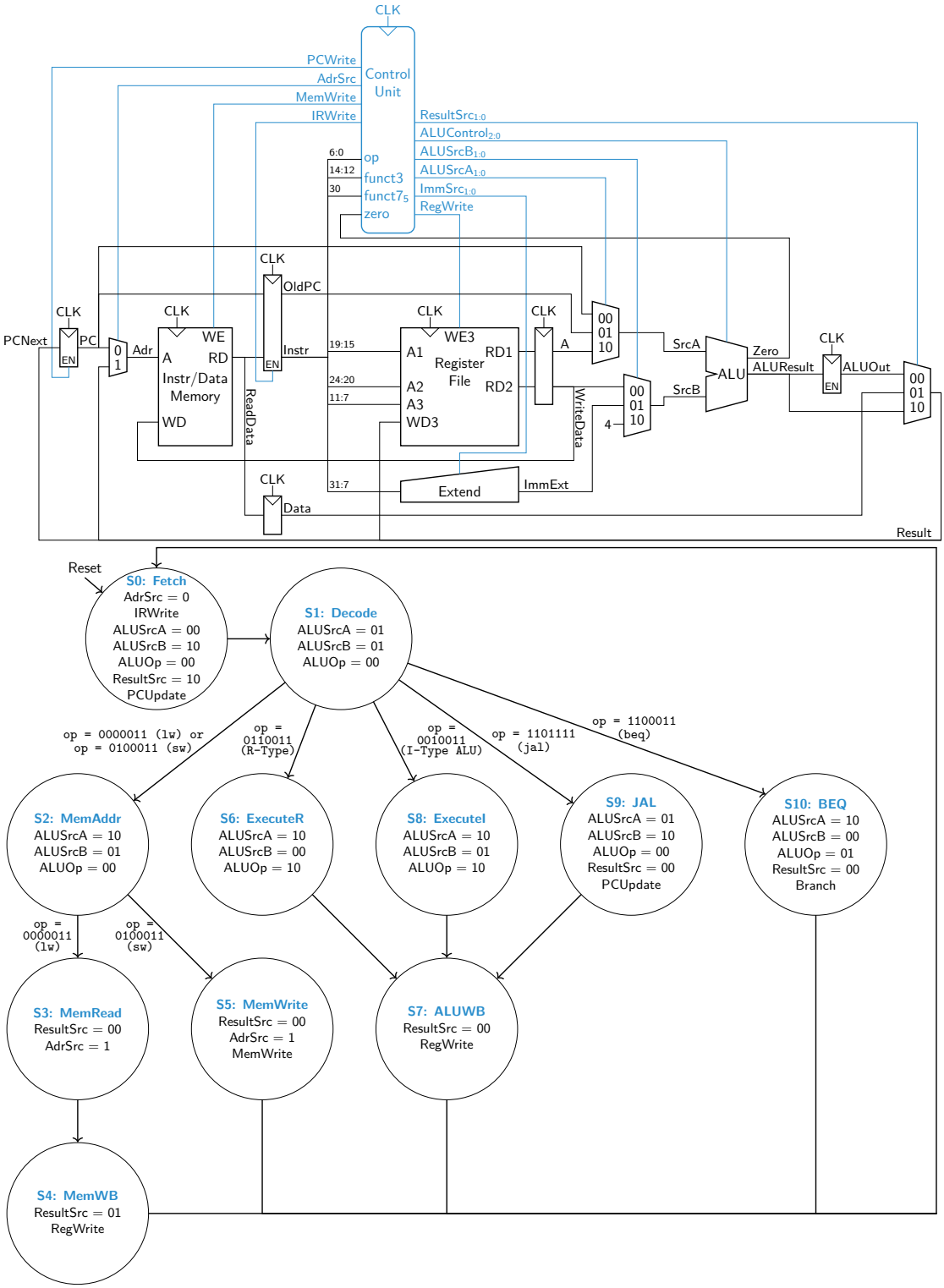
4.1 Single-Cycle RISC-V Processor



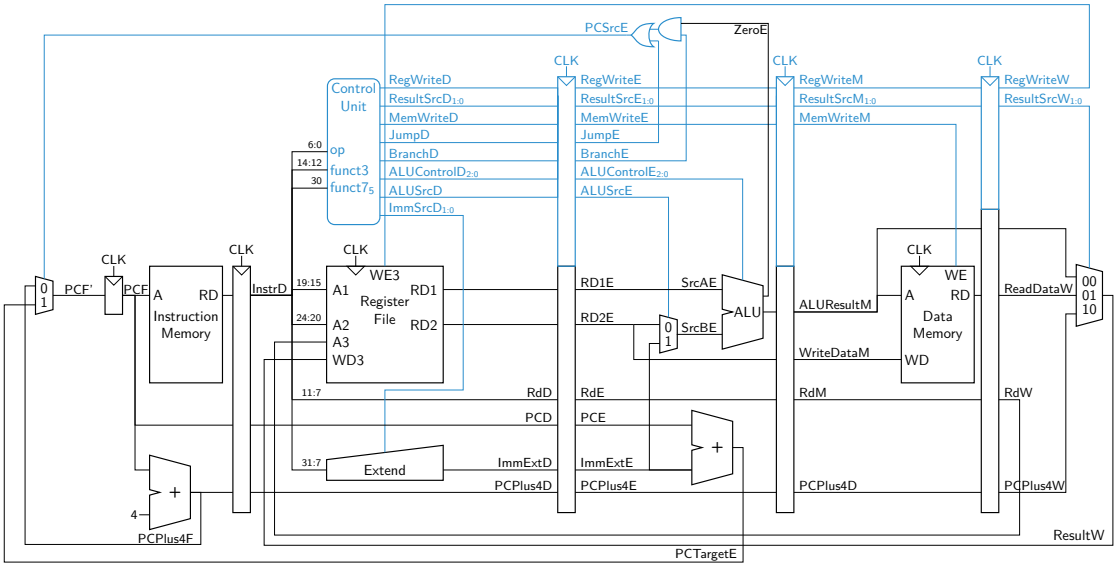
4.2 Single-Cycle RISC-V Processor (with Branch Logic)



4.3 Multi-Cycle RISC-V Processor



4.4 Pipelined RISC-V Processor



4.5 Pipelined RISC-V Processor with Hazard Unit