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

March 13, 2025

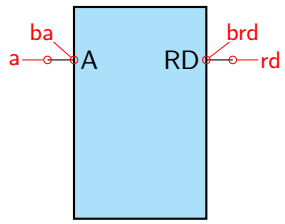
1 Introduction

1.1 Motivation

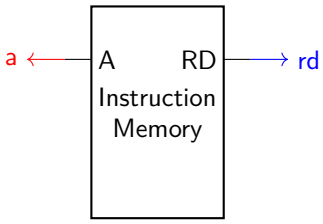
1.2 Usage

2 Component List

2.1 Instruction Memory

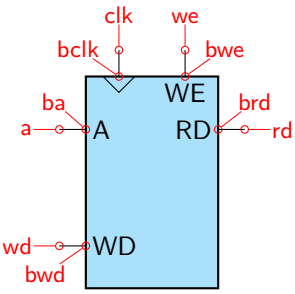


Instruction memory, type: node, fillable (`node[instrmem]{}`). Class: `processor`.

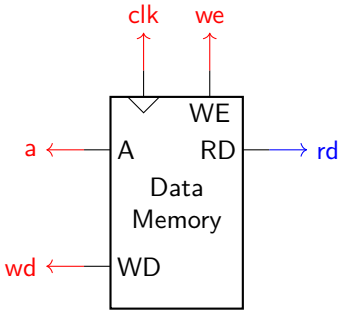



```
\begin{circuitikz}
  \node[instrmem, align=center] (comp) {Instruction\\
      Memory};
  \draw[->, red] (comp.a) -- ++(-.5, 0) node[left] {a};
  \draw[->, blue] (comp.rd) -- ++(.5, 0) node[right] {rd};
\end{circuitikz}
```

2.2 Data Memory

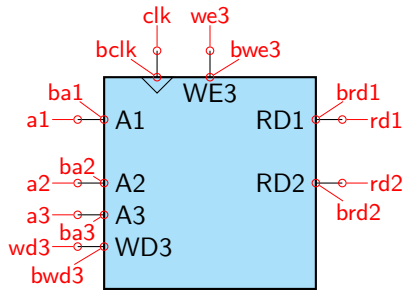


Data Memory, type: node, fillable (`node[datamem]{}`). Class: `processor`.

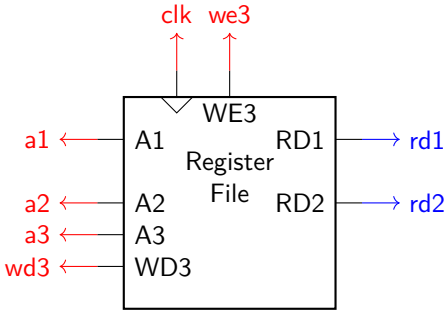



```
\begin{circuitikz}[]
  \node[datamem, align=center] (comp) {Data\\Memory};
  \draw[->, red] (comp.a) -- ++(-.5, 0) node[left] {a};
  \draw[->, red] (comp.wd) -- ++(-.5, 0) node[left] {wd};
  \draw[->, red] (comp.clk) -- ++(0, .5) node[above] {clk
      };
  \draw[->, red] (comp.we) -- ++(0, .5) node[above]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

2.3 Register File

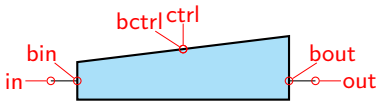


Register File, type: node, fillable (`node[regfile]{}`). Class: `processor`.

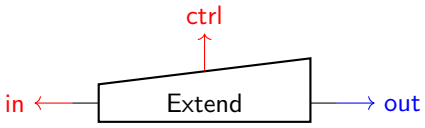



```
\begin{circuitikz}[]
  \node[regfile, align=center] (comp) {Register
      \\File};
  \draw[->, red] (comp.a1) -- ++(-.5, 0) node[
      left] {a1};
  \draw[->, red] (comp.a2) -- ++(-.5, 0) node[
      left] {a2};
  \draw[->, red] (comp.a3) -- ++(-.5, 0) node[
      left] {a3};
  \draw[->, red] (comp.wd3) -- ++(-.5, 0) node[
      left] {wd3};
  \draw[->, red] (comp.clk) -- ++(0, .5) node[
      above] {clk};
  \draw[->, red] (comp.we3) -- ++(0, .5) node[
      above] {we3};
  \draw[->, blue] (comp.rd1) -- ++(.5, 0) node[
      right] {rd1};
  \draw[->, blue] (comp.rd2) -- ++(.5, 0) node[
      right] {rd2};
\end{circuitikz}
```

2.4 Extend Unit

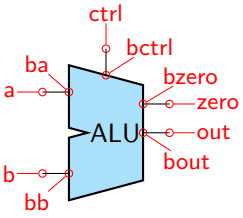


Extend Unit, type: node, fillable (`node[extend]{}`). Class: `processor`.

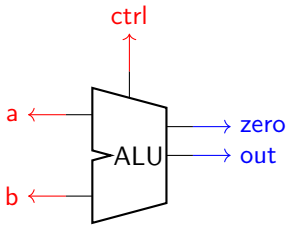



```
\begin{circuitikz}[]
  \node[extend, align=center] (comp) {Extend};
  \draw[->, red] (comp.in) -- ++(-.5, 0) node[left
      ] {in};
  \draw[->, red] (comp.ctrl) -- ++(0, .5) node[
      above] {ctrl};
  \draw[->, blue] (comp.out) -- ++(.5, 0) node[
      right] {out};
\end{circuitikz}
```

2.5 Arithmetic Logic Unit

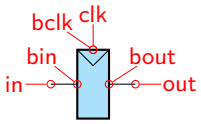


Arithmetic Logic Unit, type: node, fillable (`node[alu]{ALU}`). Class: `processor`.

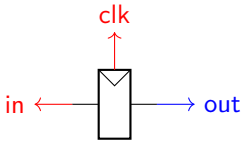



```
\begin{circuitikz}[]
  \node[alu, align=center] (comp) {ALU};
  \draw[->, red] (comp.a) -- ++(-.5, 0) node[left] {a};
  \draw[->, red] (comp.b) -- ++(-.5, 0) node[left] {b};
  \draw[->, red] (comp.ctrl) -- ++(0, .5) node[above] {ctrl};
  \draw[->, blue] (comp.out) -- ++(.5, 0) node[right] {out};
  \draw[->, blue] (comp.zero) -- ++(.5, 0) node[right] {zero
      };
\end{circuitikz}
```

2.6 Register

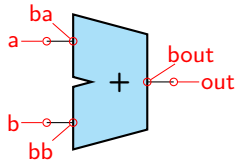


Register, type: node, fillable (`node[reg]{}`). Class: `processor`.

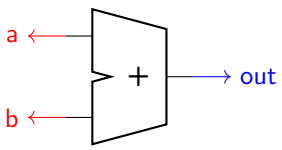



```
\begin{circuitikz}[]
  \node[reg, align=center] (comp) {};
  \draw[->, red] (comp.in) -- ++(-.5, 0) node[left] {in};
  \draw[->, red] (comp.clk) -- ++(0, .5) node[above] {clk};
  %\draw[->, red] (comp.en) -- ++(0, -.5) node[below] {en};
  \draw[->, blue] (comp.out) -- ++(.5, 0) node[right] {out};
\end{circuitikz}
```

2.7 Adder

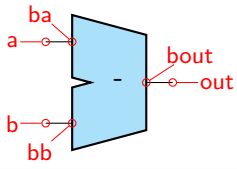


Adder, type: node, fillable (`node[adder]{}`). Class: `processor`.

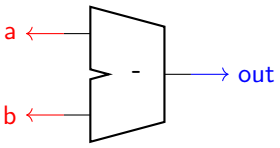



```
\begin{circuitikz}[]
  \node[adder, align=center] (comp) {};
  \draw[->, red] (comp.a) -- ++(-.5, 0) node[left] {a};
  \draw[->, red] (comp.b) -- ++(-.5, 0) node[left] {b};
  \draw[->, blue] (comp.out) -- ++(.5, 0) node[right] {out};
\end{circuitikz}
```

2.8 Subtractor

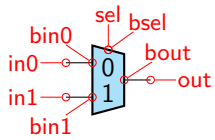


Subtractor, type: node, fillable (`node[subtr]{}`). Class: `processor`.

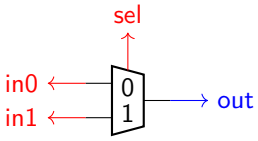



```
\begin{circuitikz}[]
  \node[subtr, align=center] (comp) {};
  \draw[->, red] (comp.a) -- ++(-.5, 0) node[left] {a};
  \draw[->, red] (comp.b) -- ++(-.5, 0) node[left] {b};
  \draw[->, blue] (comp.out) -- ++(.5, 0) node[right] {out};
\end{circuitikz}
```

2.9 Multiplexer

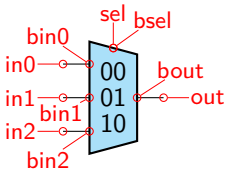


Multiplexer, type: node, fillable (`node[mux]{}`). Class: `processor`.

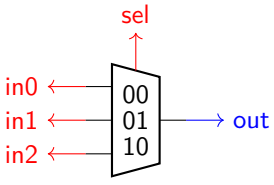



```
\begin{circuitikz}[]
  \node[mux, align=center] (comp) {};
  \draw[->, red] (comp.in0) -- ++(-.5, 0) node[left] {in0};
  \draw[->, red] (comp.in1) -- ++(-.5, 0) node[left] {in1};
  \draw[->, red] (comp.sel) -- ++(0, .5) node[above] {sel};
  \draw[->, blue] (comp.out) -- ++(.5, 0) node[right] {out};
\end{circuitikz}
```

2.10 Multiplexer with 3 inputs

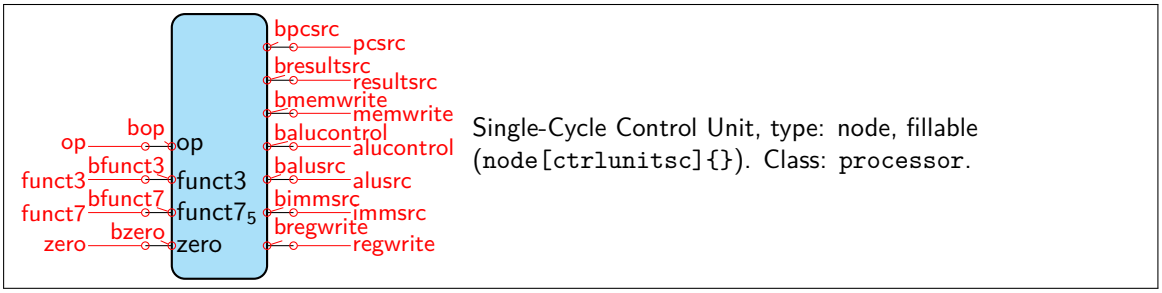


Multiplexer with 3 inputs, type: node, fillable (`node[3mux]{}`). Class: `processor`.

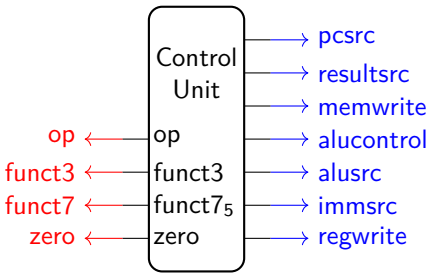



```
\begin{circuitikz}[]
  \node[3mux, align=center] (comp) {};
  \draw[->, red] (comp.in0) -- ++(-.5, 0) node[left] {in0};
  \draw[->, red] (comp.in1) -- ++(-.5, 0) node[left] {in1};
  \draw[->, red] (comp.in2) -- ++(-.5, 0) node[left] {in2};
  \draw[->, red] (comp.sel) -- ++(0, .5) node[above] {sel};
  \draw[->, blue] (comp.out) -- ++(.5, 0) node[right] {out};
\end{circuitikz}
```

2.11 Single-Cycle Control Unit



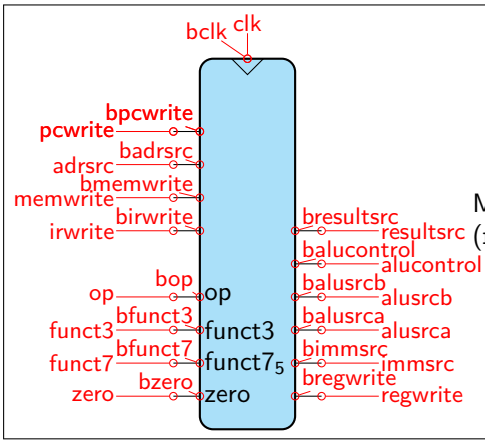
Single-Cycle Control Unit, type: node, fillable (`node[ctrlunitsc]{}`). Class: `processor`.



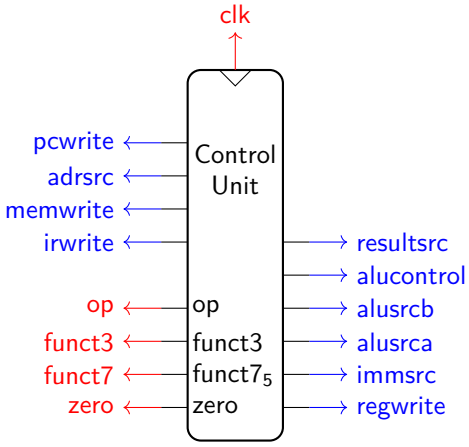

```
\begin{circuitikz}[]
  \node[ctrlunitsc, align=center] (comp) {Control
      \\Unit};
  \draw[->, red] (comp.op) -- ++(-.5, 0) node[
      left] {op};
  \draw[->, red] (comp.funct3) -- ++(-.5, 0) node
      [left] {funct3};
  \draw[->, red] (comp.funct7) -- ++(-.5, 0) node
      [left] {funct7};
  \draw[->, red] (comp.zero) -- ++(-.5, 0) node[
      left] {zero};

  \draw[->, blue] (comp.pcsrc) -- ++(.5, 0) node[
      right] {pcsrc};
  \draw[->, blue] (comp.resultsrc) -- ++(.5, 0)
      node[right] {resultsrc};
  \draw[->, blue] (comp.memwrite) -- ++(.5, 0)
      node[right] {memwrite};
  \draw[->, blue] (comp.alucontrol) -- ++(.5, 0)
      node[right] {alucontrol};
  \draw[->, blue] (comp.alusrc) -- ++(.5, 0) node
      [right] {alusrc};
  \draw[->, blue] (comp.immsrc) -- ++(.5, 0) node
      [right] {immsrc};
  \draw[->, blue] (comp.regwrite) -- ++(.5, 0)
      node[right] {regwrite};
\end{circuitikz}
```

2.12 Multi-Cycle Control Unit



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




```
\begin{circuitikz}[]
  \node[ctrlunitmc, align=center] (comp) {
      Control\\Unit};
  \draw[->, red] (comp.op) -- ++(-.5, 0) node[
      left] {op};
  \draw[->, red] (comp.funct3) -- ++(-.5, 0)
      node[left] {funct3};
  \draw[->, red] (comp.funct7) -- ++(-.5, 0)
      node[left] {funct7};
  \draw[->, red] (comp.zero) -- ++(-.5, 0)
      node[left] {zero};
  \draw[->, red] (comp.clk) -- ++(0,.5) node[
      above] {clk};

  \draw[->, blue] (comp.resultsrc) -- ++(.5,
      0) node[right] {resultsrc};
  \draw[->, blue] (comp.memwrite) -- ++(-.5,
      0) node[left] {memwrite};
  \draw[->, blue] (comp.alucontrol) -- ++(.5,
      0) node[right] {alucontrol};
  \draw[->, blue] (comp.alusrca) -- ++(.5, 0)
      node[right] {alusrca};
  \draw[->, blue] (comp.alusrcb) -- ++(.5, 0)
      node[right] {alusrcb};
  \draw[->, blue] (comp.immsrc) -- ++(.5, 0)
      node[right] {immsrc};
  \draw[->, blue] (comp.regwrite) -- ++(.5, 0)
       node[right] {regwrite};
  \draw[->, blue] (comp.irwrite) -- ++(-.5, 0)
       node[left] {irwrite};
  \draw[->, blue] (comp.adrsrc) -- ++(-.5, 0)
      node[left] {adrsrc};
  \draw[->, blue] (comp.pcwrite) -- ++(-.5, 0)
       node[left] {pcwrite};
\end{circuitikz}
```

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 ??. A list of component families can be found in Table ??.

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`

Table 1: List of CircuiTi*k*Z keys

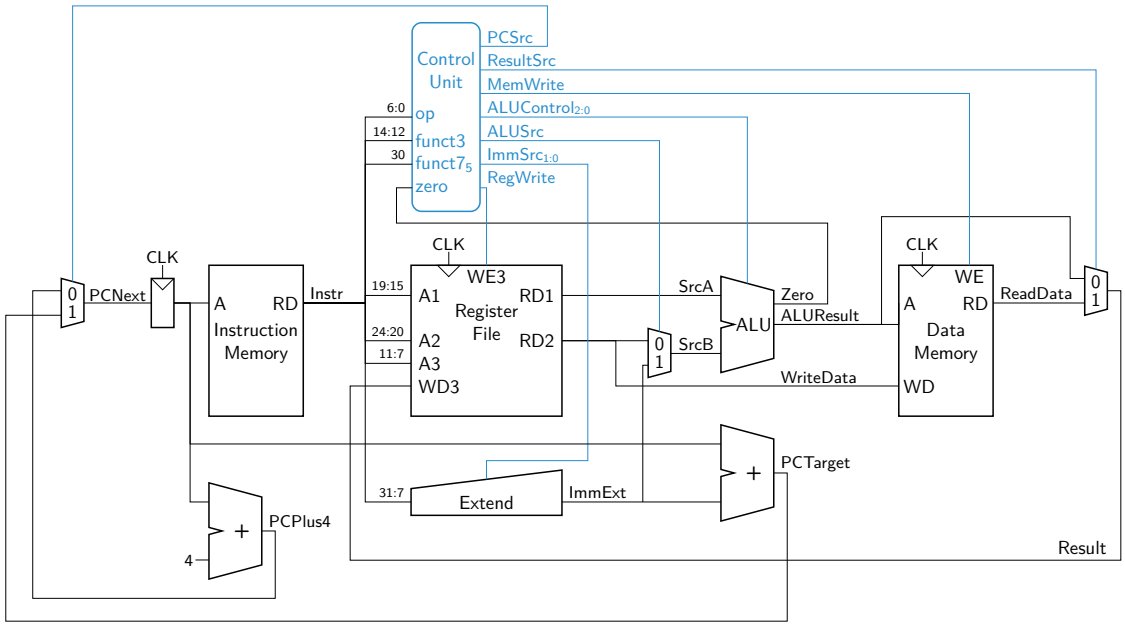
Component family	Component list
memory components	`instrmem`, `datamem`, `regfile`
control components	`ctrlunitsc`, `ctrlunitmc`
arithmetic components	`alu`, `add`, `sub`
extend components	`extend`
small components	`mux`, `reg`

Table 2: List of component families

3.2 Special node keys

4 Examples

4.1 Single-Cycle RISC-V Processor



4.2 Multi-Cycle RISC-V Processor