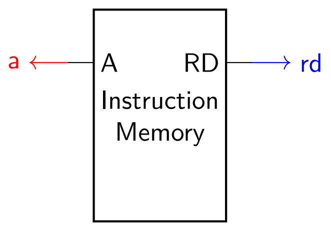
RISC-V Processor CircuiTi*k*Z Library

March 12, 2025

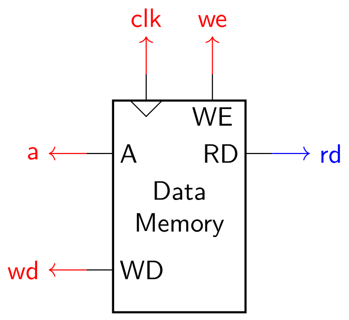
1 Component List

1.1 Instruction Memory



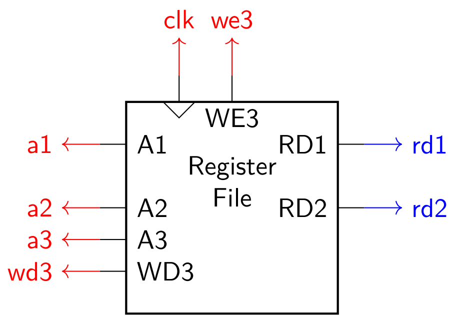
```
\begin{circuitikz}
   \node[instrmem, align=center] (comp) {Instruction\\
       Memory};

   \draw[->, red] (comp.a) -- ++(-.5, 0) node[left] {a
       };
   \draw[->, blue] (comp.rd) -- ++(.5, 0) node[right]
       {rd};
\end{circuitikz}
```

1.2 Data Memory



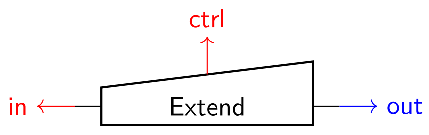
```
\begin{circuitikz}[]
   \node[datamem, align=center] (comp) {Data\\Memory
       };
   \draw[->, red] (comp.a) -- ++(-.5, 0) node[left]
       {a};
   \draw[->, red] (comp.wd) -- ++(-.5, 0) node[left]
       {wd};
   \draw[->, red] (comp.clk) -- ++(0, .5) node[above
       ] {clk};
   \draw[->, red]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

1.3 Register File



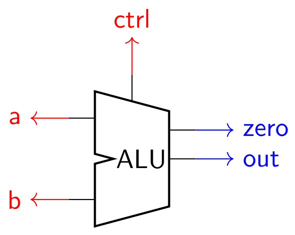
```
\begin{circuitikz}[]
   \node[regfile, align=center] (comp) {
       Register\\File};
   \draw[->, red] (comp.a1) -- ++(-.5, 0)
       node[left] {a1};
   \draw[->, red] (comp.a2) -- ++(-.5, 0)
       node[left] {a2};
   \draw[->, red] (comp.a3) -- ++(-.5, 0)
       node[left] {a3};
   \draw[->, 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}
```

1.4 Extend Unit



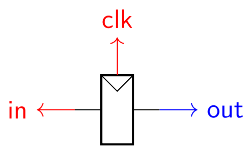
```
\begin{circuitikz}[]
   \node[extend, align=center] (comp) {Extend
       };
   \draw[->, red] (comp.in) -- ++(-.5, 0) node
       [left] {in};
   \draw[->, red] (comp.ctrl) -- ++(0, .5)
       node[above] {ctrl};
   \draw[->, blue] (comp.out) -- ++(.5, 0)
       node[right] {out};
\end{circuitikz}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

1.5 Arithmetic Logic Unit



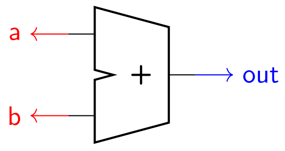
```
\begin{circuitikz}[]
   \node[alu, align=center] (comp) {ALU};
   \draw[->, red] (comp.a) -- ++(-.5, 0) node[left] {a};
   \draw[->, red] (comp.b) -- ++(-.5, 0) node[left] {b};
   \draw[->, red] (comp.ctrl) -- ++(0, .5) node[above] {
       ctrl};
   \draw[->, blue] (comp.out) -- ++(.5, 0) node[right]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

1.6 Register



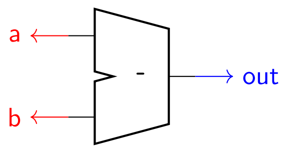
```
\begin{circuitikz}[]
   \node[reg, align=center] (comp) {};
   \draw[->, red] (comp.in) -- ++(-.5, 0) node[left] {in};
   \draw[->, red] (comp.clk) -- ++(0, .5) node[above] {clk};
   %\draw[->, red] (comp.en) -- ++(0, -.5) node[below] {en};
   \draw[->, blue] (comp.out) -- ++(.5, 0) node[right] {out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

1.7 Adder



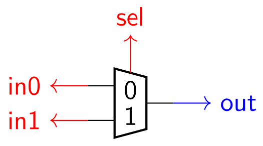
```
\begin{circuitikz}[]
   \node[adder, align=center] (comp) {};
   \draw[->, red] (comp.a) -- ++(-.5, 0) node[left] {a};
   \draw[->, red] (comp.b) -- ++(-.5, 0) node[left] {b};
   \draw[->, blue] (comp.out) -- ++(.5, 0) node[right] {
       out};
\end{circuitikz}
```

1.8 Subtractor



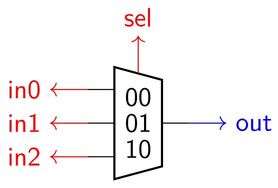
```
\begin{circuitikz}[]
   \node[sub, align=center] (comp) {};
   \draw[->, red] (comp.a) -- ++(-.5, 0) node[left] {a};
   \draw[->, red] (comp.b) -- ++(-.5, 0) node[left] {b};
   \draw[->, blue] (comp.out) -- ++(.5, 0) node[right] {
       out};
\end{circuitikz}
```

1.9 Multiplexer



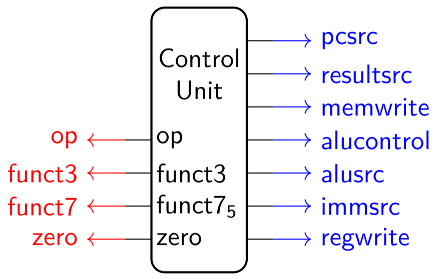
```
\begin{circuitikz}[]
   \node[mux, align=center] (comp) {};
   \draw[->, red] (comp.in0) -- ++(-.5, 0) node[left] {in
       0};
   \draw[->, red] (comp.in1) -- ++(-.5, 0) node[left] {in
       1};
   \draw[->, red] (comp.sel) -- ++(0, .5) node[above] {sel
       };
   \draw[->, blue] (comp.out) -- ++(.5, 0) node[right] {out
       };
\end{circuitikz}
```

1.10 Multiplexer with 3 inputs



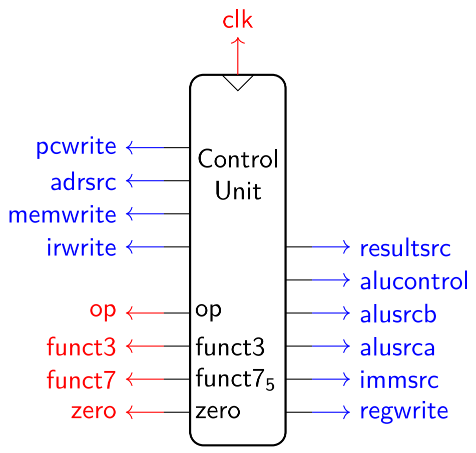
```
\begin{circuitikz}[]
   \node[3mux, align=center] (comp) {};
   \draw[->, red] (comp.in0) -- ++(-.5, 0) node[left] {in
       0};
   \draw[->, red] (comp.in1) -- ++(-.5, 0) node[left] {in
       1};
   \draw[->, red] (comp.in2) -- ++(-.5, 0) node[left] {in
       2};
   \draw[->, red] (comp.sel) -- ++(0, .5) node[above] {sel
       };
   \draw[->, blue] (comp.out) -- ++(.5, 0) node[right] {
       out};
\end{circuitikz}
```

1.11 Single-Cycle Control Unit



```
\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}
```

1.12 Multi-Cycle Control Unit



```
\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}
```

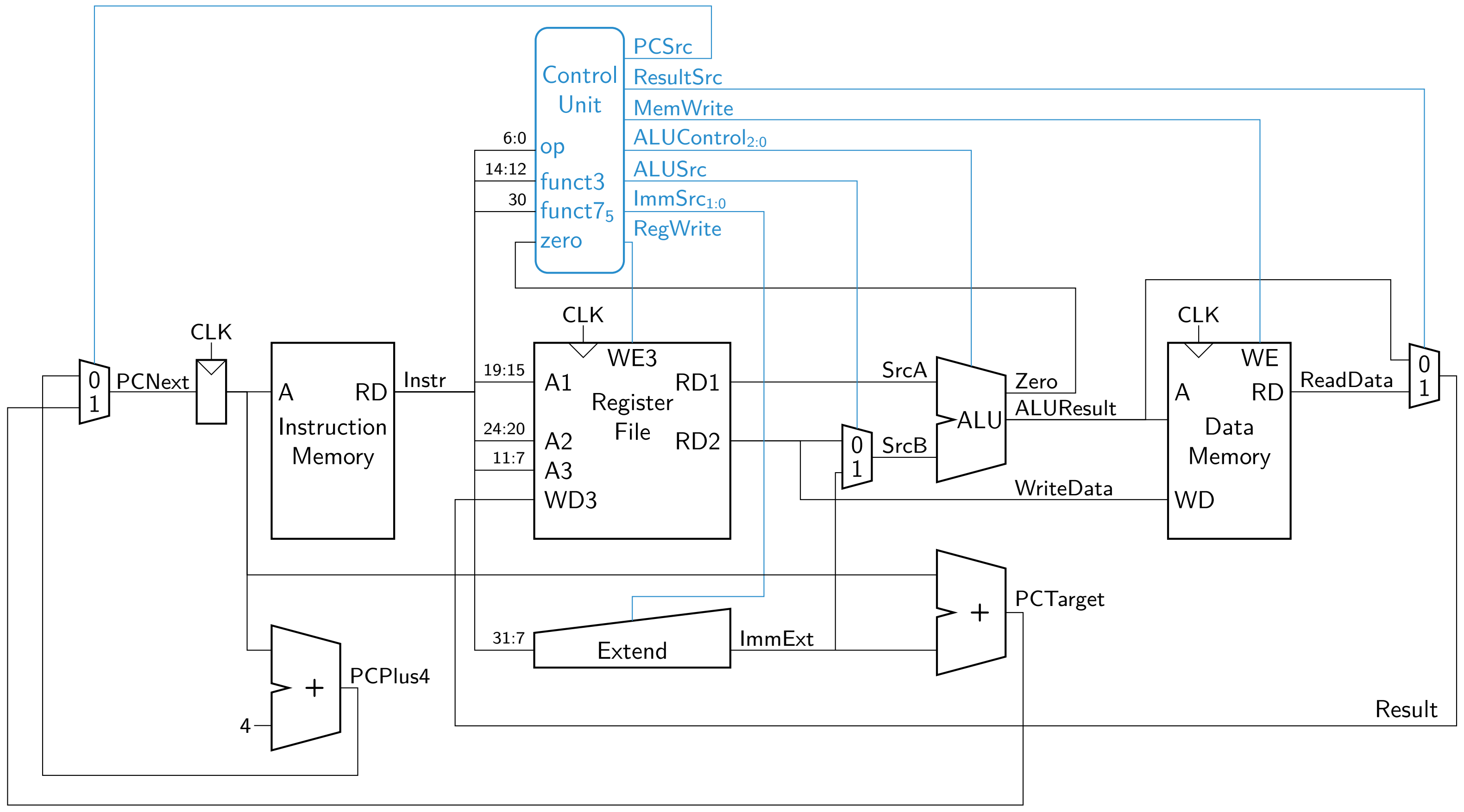
2 Keys

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.

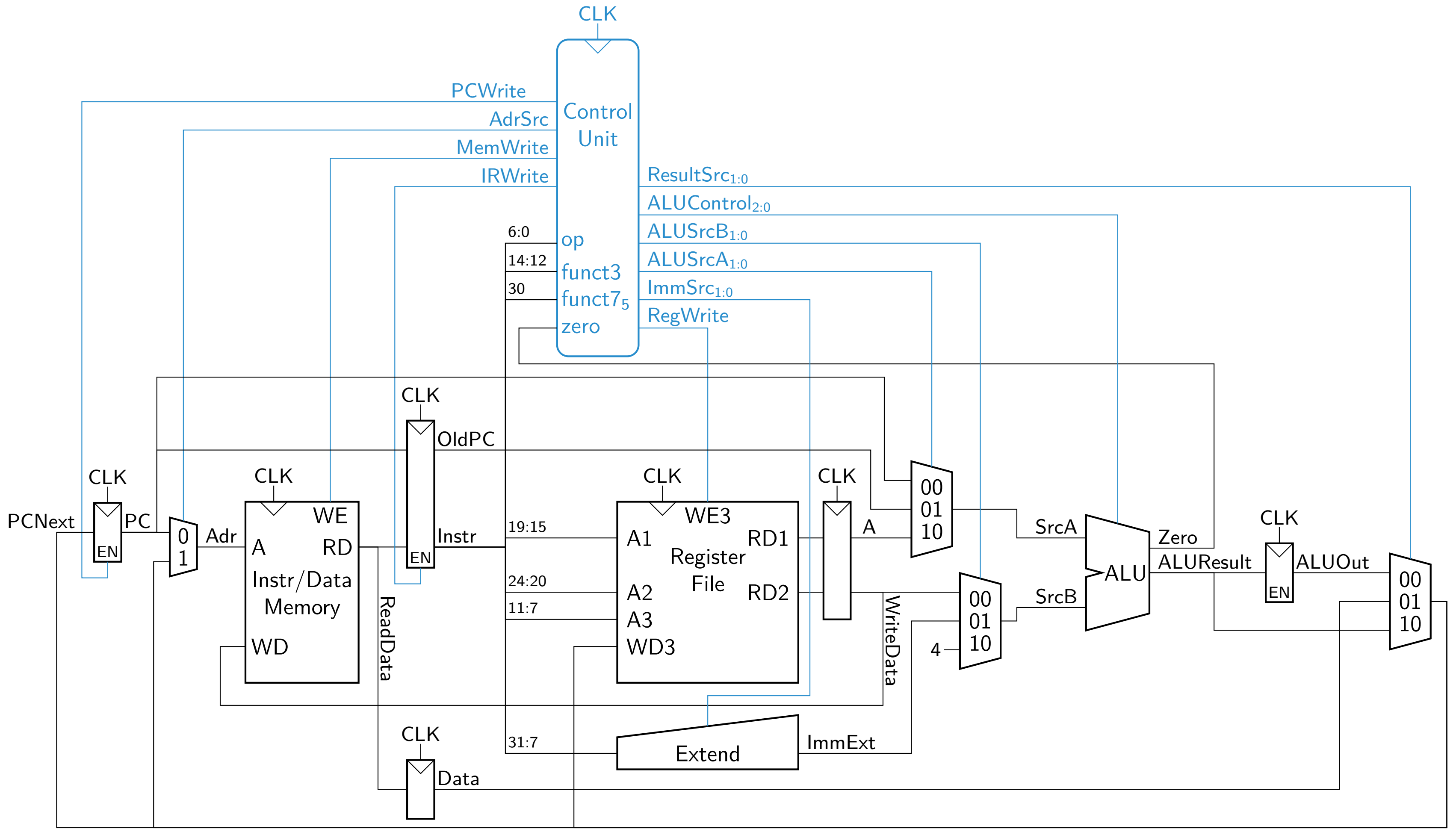
Key	Description	Default value
`scale`	Sets scale for all processor components.	1
`thickness`	Sets line width for all processor components.	2
`font`	Sets font family for all labels of processor components.	`\rmfamily`
`memory/height`	Sets height for all memory components.	2
`memory/width`	Sets width for 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 height for control components.	`3.5`
`control/radius`	Sets the 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 of the big side of the extend unit.	`0.6`
`extend/width`	Sets height of the extend unit.	2
`extend/slope`	Sets slope of the extend unit in degrees.	7
`mux/slope`	Sets slope of the 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 of input and output leads.	`0.25`

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

3 Single-Cycle RISC-V Processor



4 Multi-Cycle RISC-V Processor


CLK
Control Unit
PCWrite
AdrSrc
MemWrite
IRWrite
ResultSrc$_{1:0}$
ALUControl$_{2:0}$
ALUSrcB$_{1:0}$
ALUSrcA$_{1:0}$
ImmSrc$_{1:0}$
RegWrite
op
funct3
funct7$_5$
zero
6:0
14:12
30
PCNext
CLK
EN
PC
0
1
Adr
CLK
WE
A
RD
Instr/Data Memory
WD
ReadData
CLK
OldPC
EN
Instr
19:15
24:20
11:7
31:7
CLK
WE3
A1
A2
A3
WD3
RD1
RD2
Register File
Extend
ImmExt
CLK
A
WriteData
00
01
10
4
SrcA
SrcB
ALU
Zero
ALUResult
CLK
EN
ALUOut
CLK
Data