1 Introduction

`litmus` is a Ti*k*Z library for typesetting litmus test diagrams. It supports hand written diagrams and diagrams generated by the `rmem` tool (the former being slightly easier for humans to read and write).

2 Minimal examples

```
\documentclass{standalone}

\usepackage{tikz}
\usetikzlibrary{litmus}

\begin{document}
  \inputlitmuspicture[AArch64]{events}{A64/MP+dmb.sy+ctrl}
\end{document}
```

The examples in this document make use of `<litmus>.tikz` and `<litmus>.states.tex` files that `rmem` generates. See the included `Makefile` as to how to generate those files. `<litmus>.tikz` is a Ti*k*Z picture, and it can be used by invoking one of the input macros described in the next section (§3, Input a .tikz file) as in the example above, or by simply wrapping it with a `tikzpicture` environment:

```
%{sidebyside, lefthand width=8.5cm}
\begin{tikzpicture}[/litmus/kind=events,/litmus/AArch64]
  \input{A64/MP+dmb.sy+ctrl.tikz}
\end{tikzpicture}
```

`<litmus>.states.tex` defines the macros `\litmusname`, `\litmusarch`, `\initstate` and `\finalstate` that the `\inputlitmustable` macro uses.

3 Input a .tikz file

Inputting a `<litmus>.tikz` file that was generated by `rmem`:

```
\inputlitmuspicture[<options>]{<kind>}{<.tikz file>} % WITHOUT the .tikz suffix
  % input the .tikz file wrapped in a tikzpicture environment with '[/litmus/kind=<kind>, <options>]'

\inputlitmus[<options>]{<kind>}{<.tikz file>} % WITHOUT the .tikz suffix
  % to be used inside a tikzpicture environment;
  % input the .tikz file wrapped in a scope environment with '[/litmus/kind=<kind>, <options>]'

\inputlitmustable[<options>]{<kind>}{<.tikz file>} % WITHOUT the .tikz suffix
  % input the .tikz file and a .states.tex file wrapped in a litmus test table
```

The `<kind>` argument can be one of:

`full` - typeset a diagram with assembly code and memory events;

`events` - typeset a diagram of memory events; or

`assem` - typeset the assembly code.

The following section describes the different things that can be provided, as a comma separated list, in the `<options>` argument.

4 Options

All the options below (which are just regular pgf keys) have the path `/litmus/<option>`. Hence to globally set the `non-mixed-size` option, for example, one has to call `\tikzset{/litmus/non-mixed-size}`. Since `/litmus` is set to be a key-family, you can also use `\tikzset{/litmus,<options>}`, here `/litmus` changes the default path to `/litmus/`, so `<options>` do not need to include this path for every option. The macro `\litmusset{<options>}` is the same as doing `\tikzset{/litmus,<options>}`, which is the same as doing `\pgfkeys{/litmus,<options>}`.
When the default path is `/litmus` and an unknown key is encountered, the key is also search under the `/tikz` path.

When the macros from §3 evaluate `<options>` they first set the default path to `/litmus/` so it can be omitted from the options. Note that when you define `<key>/.style={...}` the default path inside the curly braces depends on the context in which the style is used. All the styles that are described below are used when `/litmus/` is the default path so it can be omitted from the keys there too.
When writing a diagram by hand, the default path for the options of `\node`, `\path`, `edge` etc. is always `/tikz` and there is no simple way of changing it globally. Hence, you will have to prefix all the options with `/litmus/` which is annoying. The `litmus` library defines the key `/litmus/append to tikz path search` so that when it is called the way unknown keys in the `/tikz` path are handled is changed, inside the current TEX group, to also search for the unknown key in the `/litmus` path. Therefore, after using the key `/litmus/append to tikz path search` in the options of a `tikzpicture` environment, you can use `/litmus` options anywhere within the picture environment without the `/litmus/` prefix. Note that this is somewhat of a hack (the key redefines the pgf key `/errors/unknown key`), so it is better not to use it in global settings (e.g. do not do `\tikzset{/litmus/append to tikz path search}` in the preamble) but it should be safe to use it per-`tikzpicture` environment.

Specify the architecture of the test:

```
base language % minimal definitions
AArch64
Power
RISCV
x86
every <arch>/.style={...}
  % where <arch> is one of the above architectures.
```

Specify a new architecture:

```
\litmusset={
  <arch>/.style={
    /litmus/listing options/.if defined=then {
      /litmus/listing options/.prefix={language=<arch>,}
    },
    /litmus/base language, % definitions for R, W, po, data, etc.
    % Memory access kinds as generated in pp.ml (pp_brief_write_kind
    % and pp_brief_read_kind):
    /litmus/mem access/<kind>/.code={...},% PP memory access <kind>
    % Edge labels as generated in tikz.ml
    /litmus/<edge>/.litmus relation=<color>,
    /litmus/.cd scope={every <arch>/.try},
  },
}
```

Typesetting footprints of memory accesses and `rf` edges:

```
non-mixed-size  % e.g. "a:W x=1" (this is the default)
mixed-size      % e.g. "a:W x[0..3]=1"
old-mixed-size  % e.g. "a:W x+4/2=1"

memfp/.code n args={3}{...} % (no default, initial value is set by the options above)
memrf/.code 2 args={...} % (no default, initial value is set by the options above)

is mixed-size/.if true=then {<true>}[ else {<false>}]
  % (It only makes sense to use this inside a style like 'every event')
  % Evaluates <true> when mixed-size or old-mixed-size are set,
  % otherwise evaluates <false>.
is mixed-size/.if false=then {<true>}[ else {<false>}]
```

```
threads distance=<dimension> % (no default, initially 3em (full/events) or 0 (assem))
header to instructions distance=<dimension> % (no default, initially 0.5ex (full/events) or 0 (assem))
every thread/.style={...} % #1 is set to tid
thread <tid>/.style={...}
add thread header=<true|false> % (default true, initially true)
  % This option, when given in one of the two styles above, determines
  % if the thread will have a header (e.g. "Thread 0") or not.
every thread header/.style={...} % #1 is set to tid
thread <tid> header/.style={...}
thread header text={...} % Typeset the thread header text
  % e.g. 'every thread header/.style={thread header text={Thread~#1}}'
```

```
every instructions/.style={...}
add row after={<ioid>}{<text>} % (no default)
  % can be used multiple times with the same ioid. The new lines are referenced with 'assem <ioid>-1',
  % 'assem <ioid>-2', ...

add space after row=<dimension>
  % For example:
  % every instructions/.style={
  %    /tikz/row 3/.style={/litmus/add space after row=4ex},
  % },
```

Note that `instructions` is a Ti*k*Z matrix and as such all the regular matrix keys can be used to manipulate it. In particular the key 'row sep' can be used to change the space between the instructions. If you set this key you probably want to use the 'between origins' keyword to make the threads align properly.

```
compute assem text width={<unique name>} % (no default)
```

This will initially unset the `text width` of all the assem instructions and save the width of each assem instructions column to the .aux file. In the next run of latex, if `text width` was not set manually, the `text width` of all the assem instructions will be set to the saved value. In addition, the pgf math function `computedwidthofthread("<unique name>",<tid>)` will expand to the saved width of thread `<tid>` of `<unique name>`, and `computedwidthof("<unique name>")` will expand to the saved width of the thread in context of `<unique name>`. Finally, `computedwidth` will expand to the saved width of the assem instructions column in context.

```
every assem/.style={...}
assem <ioid>/.style={...}

code label position=<above|inline|left|none> % (no default, initially 'above')
code label/.code 2 args={...} % (no default, initially prints "#1:" if #1 is not empty)
  % When 'code label position' is not set to 'none', this code will be executed for
  % every assembly instruction, with #1 set to the location label and #2 set to the ioid,
  % to typeset the code label.

show eiids comment=<true|false> % (default: true, initially true (assem) or false (full/events))
eiids comment/.code={...} % (no default, initially "\hfill//#1")
  % This code is used to typeset, at the end of the assembly line, the
  % eiids (#1) associated with the instruction.
eiids comment <ioid>/.code={...}

assem <ioid>/.style={
  % Prepend text to the assembly code
  /tikz/execute at begin node={...},
  % Append text to the assembly code (between the assembly code and the eiids comment)
  /tikz/execute at end node={...}
}
```

```
every events/.style={...} % #1 is set to ioid
events <ioid>/.style={...}
eiids order={<eiid list>}
  % For example: events 1-2/.style={eiids order={d0,d1}}
  % This key changes the order in which events are typeset for mixed-size
  % memory accesses. This key also affects the eiids comment.
  % This key can be used to totally change the events. For example,
  % adding the key 'd/.style={event=d0..1, alias=d0, alias=d1, node contents={R x=0x12}}'
  % and then setting 'events 1-2/.style={eiids order={d}}' will typeset
  % both events (d0, d1) in a single line.

every event/.style={...} % #1 is set to eiid
event <eiid>/.style={...}

show event label=<true|false> % (default: true, initially true)

every event label/.style={...} % #1 is set to eiid
event label <eiid>/.style={...}

event label/.code={...} % (no default, initially "#1:")
  % Executed for every event, with #1 set to the event's eiid, to typeset a label.
every init/.style={...} % #1 is the eiid
event init-<eiid>/.style={...}
init right=<eiid>
init left=<eiid>
init above right=<eiid>
init below right=<eiid>
init above left=<eiid>
init below left=<eiid>
every events box/.style={...} % #1 is set to ioid
events box <ioid>/.style={...}
```

Presets for nicer mixed size events:

```
boxed events=<color> % (default: black!10, initially not set)
  % You should probably set the text width of events to something
  % fixed:
  %   every event/.style={text width={width("W x=2")}},
  %   every event label/.style={text width={width("a")}},
```

```
every relation/.style={...}

every event relations/.style={...}
every instruction relations/.style={...}
label pos=<fraction> % (no default, initially 0.5)
label text=<text> % (no default, initially the edge name (e.g. 'po'))
every label/.style={...}
/tikz/<edge name>=<text> % (default: <edge name>)
  % The possible values of <edge name> depend on the loaded architecture.
  % The following are part of all architectures: po, co, rf, fr, ctrl,
  % ctrl+addr, addr, data
every <edge name>/.style={...}
every <edge name> label/.style={...}
hide <edge name>
hide edges={<edge name>, <edge name>, ...} % (no default)
hide % e.g. '(1-4) isb (1-7)/.style=hide'
vertical align=<anchor> % (no default, initially \tikztostart of the first edge on which the style was
                        % applied, in the current scope)
```

To change the edge color one can use the option:

```
every <edge name>/.style={<color>}
```

or use the `\colorlet` macro like this:

```
\colorlet{<edge name> color}{<color>}
```

```
(<node 1>) <edge name> (<node 2>)/.style={...}
  % where nodes can be ioids or eiids, changes this specific edge.
```

The above only works on existing edges, see §6 for adding edges.

```
mem access/<R* or W*>/.code={...} % (no default, initially something reasonable)
  % e.g.: mem access/Wrel/.code={write-release}
  % will produce: "a:write-release x=1"
```

By default, only in the `events` kind, edges are curved using predefined styles, based on the `{<dtid>,<drow>}` distance of the edge. For example, in the minimal example at the beginning, the edge `(e) fr (a)` has the distance `{-1,-1}`, and so, the predefined style `edge distance {-1,-1}` is applied to the edge. The style `edge distance {<dtid>,<drow>}` uses the keys `in`, `out`, `bend`, `looseness`, `label pos` and such like to curve the edge, in what is mostly a satisfying way. This does not work very well for mixed-size tests.

```
disable edge routing
enable edge routing
edge distance {<dtid>,<drow>}/.style={...}
% Below, nodes can be ioids or eiids:
from (<node>)/.disable routing                    % disable routing for all edges going out of <node>
to (<node>)/.disable routing                      % disable routing for all edges coming into <node>
(<node>)/.disable routing                         % disable all routing for <node>
(<node 1>) <edge name> (<node 2>)/.disable routing % disable routing for this specific edge
<edge name>/.disable routing                      % disable all routing for <edge name>
  % The last two can also be achived like this:
  %   (<node 1>) <edge name> (<node 2>)/.style={disable edge routing}
  %   every <edge name>/.style={disable edge routing}
```

5 Additional options for `\inputlitmustable`

```
litmus name={...}   % (initially loaded from the .state.tex file)
architecture={...}  % (initially loaded from the .state.tex file)
initial state={...} % (initially loaded from the .state.tex file)
final state={...}   % (initially loaded from the .state.tex file)

initial state prefix={...}  % (initially "Initial state:~")
final state prefix={...}    % (initially "Final state:~")

% The following options set 'final state prefix', and in addition, if
% the argument is not empty, set 'final state' to the argument:
final state allowed={<state>}               % "Allowed:~"
final state allowed not observed={<state>}  % "Allowed (not observed):~"
final state forbidden={<state>}             % "Forbidden:~"
final state forbidden observed={<state>}    % "Forbidden (observed):~"

state prefix font={...} % (initially empty)

every table header/.style={...}
every box/.style={...}
every state/.style={...}
every initial state/.style={...}
every final state/.style={...}
every threads box/.style={...}

every border between threads/.style={...}
every thread headers border/.style={...}

every final state allowed/.style={...}
every final state forbidden/.style={...}
every final state unknown/.style={...}

every code line/.style={...}
every odd code line/.style={...}
every even code line/.style={...}
  % In all the above styles, #1 is set to the line number.
every code line <i>/.style={...} % where <i> is a line number (1 for the first line)

table construction/.style={...}
  % (initially "{table header={above=1pt of threads box, anchor=base},
  %              initial state box={below=2pt of threads box},
  %              final state box={below=of initial state box}})
  % Redefine this style to construct the table in a different way.
```

6 Patching memory accesses

One can always brute force it using `node contents`:

```
event <eiid>/.style={node contents={<text>}}
```

A better way is to use one or several of the following `event <eiid>/.change <...>` handlers. Those take an argument of the form `from <old> to <new>`, where `<old>` is the value specified in the `.tikz` file and `<new>` is the value you want it to change to. If the `<old>` value does not match the `.tikz` value, or the `<eiid>` does not exist, a compilation error is raised.

```
event <eiid>/.change addr=from x to y
event <eiid>/.change offset=from 0 to 4
event <eiid>/.change size=from 4 to 8
event <eiid>/.change footprint=from x+0/4 to y+4/8 % do all the above in one go
event <eiid>/.change value=from 1 to 2
event <eiid>/.change mem access={from x+0/4=1 to y+4/8=2} % the { } are mandatory
event <eiid>/.change eiid=to <new eiid> % this one does not have a 'from' part

check value/.code={}  % disables the old value check
check exists/.code={} % disables the existence check
```

Adding missing edges:

```
% for rf/co/fr we want the edge to be between two eiids (maybe from different threads):
(<eiid 1>) <edge name> (<eiid 2>)/.add events edge={...}
  % this works even when <eiid 1> is an 'init-<eiid>' and even if the init node does
  % not exist, in which case the init node will be added too.

% for all the other edges we want the edge to be between two ioids (from the same thread):
(<ioid 1>) <edge name> (<ioid 2>)/.add instructions edge={...}
```

7 Global options

In addition to the options from the previous sections, one can also use:

```
every litmus/.style={...}
every full litmus/.style={...}
every events litmus/.style={...}
every assem litmus/.style={...}
every litmus table/.style={...}
every litmus picture/.style=

is assem litmus/.if true=then {<true>}[ else {<false>}]
is assem litmus/.if false=then {<true>}[ else {<false>}]
is events litmus/.if true=then {<true>}[ else {<false>}]
is events litmus/.if false=then {<true>}[ else {<false>}]
  % It only makes sense to use these inside a style like 'every litmus'
```

For example, this document uses this:

```
\newif\ifdrawbox
\tikzset{
  draw box/.is if flag=drawbox, draw box,
  /litmus/every litmus/.style={
    AArch64,
    % Add a dashed line to all the litmus pictures that follow, showing their bounding box
    /tikz/draw box/.if true=then {
      execute at end scope={
        \draw[dashed, thin, opacity=0.2]
          (current bounding box.south west) rectangle (current bounding box.north east);
      }
    },
    % Don't include the edges (and their labels) when calculating the
    % bounding box (makes the pictures very tight)
    every relation/.append style=overlay,
    % DEBUG:
    %every node/.style=draw,
    %code label position=left, code label/.code 2 args={{\color{black}\tiny ##2}},
  },
}
```

The assembly code is typeset using `\assem|<code>|`. Initially `\assem` is defined to be `\verb`. One should take care when redefining `\assem` as it needs to handle unescaped input. In addition, for technical reasons, line numbers, code labels and the eiids comments (when enabled) can not be typeset using `\verb`. Instead, initially, they are typeset using `\ttfamily`.

`litmus.listings` is an extension to the `litmus` Ti*k*Z library that makes it easier to redefine `\assem` to use the listings package. After loading the library the Ti*k*Z key `use listing={<opts>}` will redefine `\assem` to use `\lstinline[<opts>]`. `<opts>` can be changed later using the Ti*k*Z key `listing options={<opts>}` (or `listing options/.append={<opts>}` or `listing options/.prefix={<opts>}`). In addition, `use listing` will cause code labels and the eiids comments to be typeset using the same macro. This document loads Ti*k*Z like this:

```
\usepackage{tikz}
\usetikzlibrary{litmus, litmus.listings}
\litmusset{use listing={style=litmus, basicstyle=\footnotesize\ttfamily}}
```

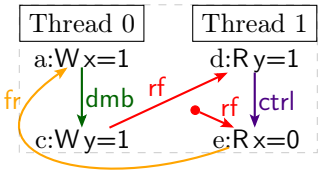
8 Examples

A few practical tips before the examples:

- empty lines between the square brackets will produce cryptic compilation errors;
- it can be useful to use `.append style` instead of `.style`;
- it is always safe to enclose the right hand side of = in curly braces, but if the right hand side expression contains "=" or "," it must be enclosed in curly braces;
- do not edit the .tikz and .states.tex files generated by rmem, instead use `<options>` to patch them;
- code labels with special characters (_, $, etc.) will cause compilation errors with the default `\assem` macro; this can be fixed by using the listings package as described in §7;
- if the document has a large number of litmus diagrams you probably want to use Ti*k*Z's externalization library (see the PGF manual) to speed up compilation.

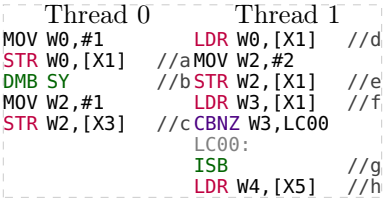
8.1 A simple events example

```
% Example 1.
\inputlitmuspicture{events}{A64/MP+dmb.sy+ctrl}
```



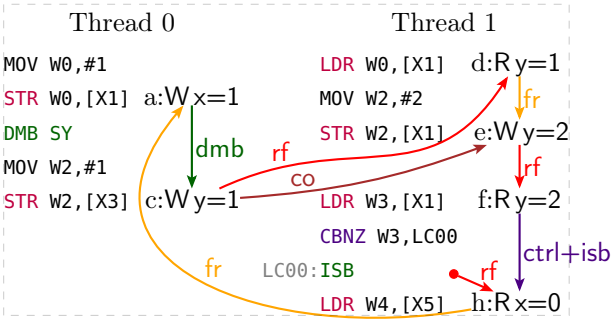
8.2 A simple assem example

```
% Example 2.
\inputlitmuspicture[%
  every thread header/.style={inner sep=0},
  header to instructions distance=0.5ex,
  threads distance=1pt,
  compute assem text width=example2,
  every assem/.append style={
    text width=computedwidth + 1em,
  },
]{assem}{A64/MP+dmb.sy+fri-rfi-ctrlisb}
```



8.3 A simple full example

```
% Example 3.
\inputlitmuspicture[%
  code label position=left,
  add row after={1-3}{},
% edge routing:
  hide edges={po, isb},
  (1-4) ctrl (1-7)/.style={label text=ctrl+isb},
  (c) rf (d)/.style={label pos=0.2, out=25, in=-123,
                     looseness=1.2},
  (c) co (e)/.style={bend right=8, label pos=0.3},
  % edges between instructions will automatically
  % appear to be on a straight line. For other edges
  % we have to do it by hand by using
  % 'vertical align=<node>' as below. The edge will
  % be aligned with '<node>.center'. The
  % auto-aligned edges will align with the node from
  % which the first edge exits. In this case we have
  % hidden the first two edges ('hide edges' above),
  % so the first node is '1-4' (the ioid 'ctrl'
  % exits from).
  (d) fr (e)/.style={vertical align=1-4},
  (e) rf (f)/.style={vertical align=1-4},
  (h) fr (a)/.style={out=-170, in=-130,
                     looseness=1.5, swap},
]{full}{A64/MP+dmb.sy+fri-rfi-ctrlisb}
```



8.4 Assem table examples

```
% Example 4a.
\tikzset{draw box=false}
\inputlitmustable[%
  code label position=inline,
  compute assem text width=example4a,
  every assem/.append style={
    inner xsep=1pt,
    text width=computedwidth + 1.5em,
  },
  every odd code line/.style={fill=black!10},
  % Show line numbers:
  every code line/.style={
    append after command={
      node[anchor=west, at=(\tikzlastnode.west), inner xsep=1pt
        % Alternative (outside the table): anchor=east
      ] {\scriptsize\texttt{#1.}}
    },
  },
  thread 0/.style={
    % Leave some room for the line number
    execute at begin node={\phantom{\scriptsize\texttt{0.}}},
    add row after={0-5}{}, % 'assem 0-5-1'
    add row after={0-5}{\ttfamily//comment}, % 'assem 0-5-2'
  },
  % Alternative line numbers:
  % every assem/.append style={
  %   execute at begin node={%
  %     {\scriptsize\texttt{\the\pgfmatrixcurrentrow.}}%
  %   },
  % },
]{assem}{A64/MP+dmb.sy+fri-rfi-ctrlisb}
```

MP+dmb.sy+fri-rfi-ctrlisb AArch64

Thread 0	Thread 1
1.MOV W0,#1	LDR W0,[X1] //d
2.STR W0,[X1] //a	MOV W2,#2
3.DMB SY //b	STR W2,[X1] //e
4.MOV W2,#1	LDR W3,[X1] //f
5.STR W2,[X3] //c	CBNZ W3,LC00
6.	LC00:ISB //g
7.//comment	LDR W4,[X5] //h

Initial state: 0:X3=y; 0:X1=x; 1:X5=x; 1:X1=y;

Allowed: 1:X0=1; 1:X3=2; 1:X4=0; y=2; x=1;

```
% Example 4b.
\tikzset{draw box=false}
\inputlitmustable[%
  compute assem text width=example4b,
  every assem/.append style={
    inner xsep=1pt,
    text width=computedwidth + 1.5em,
  },
  thread 1/.style={
    % Make sure Thread 1 and 2 have the same width
    every assem/.append style={
      text width={max(computedwidth,
          computedwidthofthread("example4b",2)) + 1.5em},
    },
  },
  thread 2/.style={
    % Place Thread 2 below Thread 1 (using Thread 1's chain alias
    % "threads-2")
    every instructions/.style={below=4ex of threads-2.south west},
    % Make sure Thread 1 and 2 have the same width
    every assem/.append style={
      text width={max(computedwidth,
          computedwidthofthread("example4b",1)) + 1.5em},
    },
  },
  execute at end scope={
    % Draw the vertical lines below and above Thread 2's header
    \draw (threads-3.north west) -- (threads-3.north east);
    \draw (thread 2 header.north -| threads-3.west)
      -- (thread 2 header.north -| threads-3.east);
  },
]{assem}{A64/WRC+addrs}
```

WRC+addrs AArch64

Thread 0	Thread 1
MOV W0,#1	LDR W0,[X1] //b
STR W0,[X1] //a	EOR W2,W0,W0
	MOV W3,#1
	STR W3,[X4,W2,SXTW] //c
	Thread 2
	LDR W0,[X1] //d
	EOR W2,W0,W0
	LDR W3,[X4,W2,SXTW] //e

Initial state: 0:X1=x; 1:X4=y; 1:X1=x; 2:X4=x; 2:X1=y;

Allowed: 1:X0=1; 2:X0=1; 2:X3=0;

8.5 A full table example

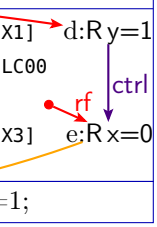
```
% Example 5.
\tikzset{draw box=false}
\inputlitmustable[%
  final state allowed,
  architecture=\color{red}ARMv8,
  % change the borders:
  every box/.style={draw=blue!70!black},
  every thread headers border/.style={draw=blue!70!black},
  every border between threads/.style={draw=blue!70!black},
  % reorder the boxes:
  table construction/.style={
    initial state box={above=of threads box},
    table header={above=1pt of initial state box},
    final state box={below=of threads box},
  },
  every assem/.append style={inner xsep=1pt},
% edge routing:
  (c) rf (d)/.style={out=35, in=172, out looseness=2, in looseness=2.2, label pos=0.3},
  (e) fr (a)/.style={out=-160, in=-135, looseness=1.8, label pos=0.9},
]{full}{A64/MP+dmb.sy+ctrl}
```

MP+dmb.sy+ctrl ARMv8

Initial state: 0:X3=y; 0:X1=x; 1:X3=x; 1:X1=y;	
Thread 0	Thread 1

MOV W0,#1
STR W0,[X1] a:W x=1
DMB SY
MOV W2,#1
STR W2,[X3] c:W y=1
	LDR W0,[X1] d:R y=1
	CBNZ W0,LC00
	LC00:
	LDR W2,[X3] e:R x=0

Allowed: 1:X0=1; 1:X2=0; y=1; x=1;	



8.6 A table and diagram example

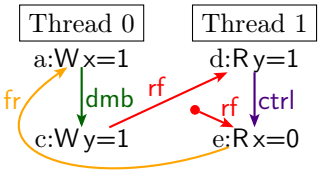
```
% Example 6.
\litmusset{
  every litmus/.append style={
    thread 0/.append style={every thread header/.style={alias=thread-0}},
    baseline=(thread-0.base),
  },
}
\inputlitmustable[%
  compute assem text width=example6a,
  every assem/.append style={
    inner xsep=1pt,
    text width=computedwidth + 1em,
  },
]{assem}{A64/MP+dmb.sy+ctrl}
\hspace{1em}
\inputlitmuspicture{events}{A64/MP+dmb.sy+ctrl}
```

MP+dmb.sy+ctrl AArch64

Thread 0	Thread 1

MOV W0,#1
STR W0,[X1] //a
DMB SY //b
MOV W2,#1
STR W2,[X3] //c
	LDR W0,[X1] //d
	CBNZ W0,LC00
	LC00:
	LDR W2,[X3] //e

Initial state: 0:X3=y; 0:X1=x; 1:X3=x; 1:X1=y;	
Allowed: 1:X0=1; 1:X2=0; y=1; x=1;	

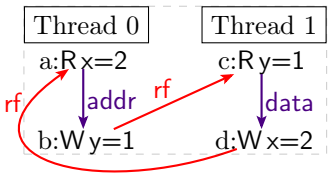


The above is nice as an inline diagram that has no caption. If you use this inside a figure environment with a caption and you want to remove the litmus name and architecture from on top of the table, add this to the options:

```
table construction/.style={
  initial state box={below=2pt of threads box},
  final state box={below=of initial state box},
},
```

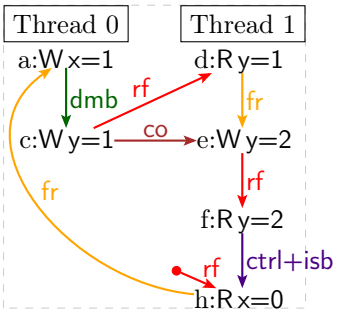
8.7 Patching edges and values example

```
% Example 7.
\inputlitmuspicture[%
% remove bad edges:
  (c) fr (b)/.style=hide,
  (init-c) rf (c)/.style=hide,
  event init-c/.style=hide,
% add missing edges and values:
  (b) rf (c)/.add events edge,
  event a/.change value=from 1 to 2,
  event c/.change value=from 0 to 1,
  event d/.change value=from 1 to 2,
]{events}{A64/LB+addr+data}
```



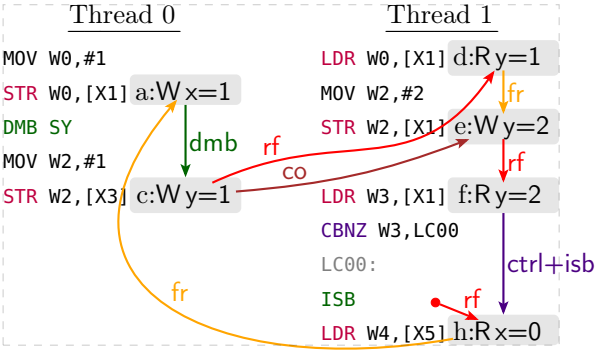
8.8 A simple events example

```
% Example 8.
\inputlitmuspicture[%
  hide edges={po, isb},
  (1-4) ctrl (1-7)/.style={label text=ctrl+isb},
  (d) fr (e)/.style={vertical align=1-4},
  (e) rf (f)/.style={vertical align=1-4},
  (h) fr (a)/.style=swap,
]{events}{A64/MP+dmb.sy+fri-rfi-ctrlisb}
```



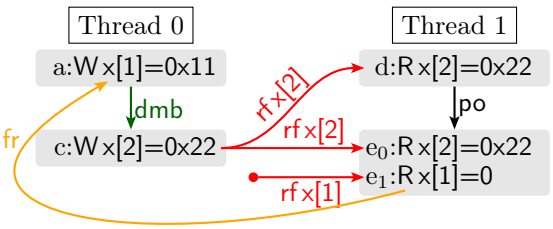
8.9 A full example with boxed events

```
% Example 9.
\inputlitmuspicture[%
  boxed events,% adds the gray boxes around the events
  every event/.style={text width={width("W y=2")}},
  every event label/.style={text width={width("a:")}},
  every thread header/.style={inner sep=0, below delimiter=|},
  header to instructions distance=1ex,
  every instructions/.style={column sep=0},
  add row after={1-3}{},
  assem 1-5/.style={overlay},
% edge routing:
  hide edges={po, isb},
  (1-4) ctrl (1-7)/.style={label text=ctrl+isb},
  (c) rf (d)/.style={label pos=0.2, out=25, in=-125, looseness=1.2},
  (c) co (e)/.style={bend right=8, label pos=0.3},
  (d) fr (e)/.style={vertical align=1-4},
  (e) rf (f)/.style={vertical align=1-4},
  (h) fr (a)/.style={out=-170, in=-130, looseness=1.7, swap},
]{full}{A64/MP+dmb.sy+fri-rfi-ctrlisb}
```



8.10 A mixed-size example with boxed events

```
% Example 10.
\tikzset{draw box=false}
\inputlitmuspicture[%
  mixed-size,
  boxed events,
  every event/.style={text width={width("W x[2]=0x22")}},
  every event label/.style={text width={width("e0:")}},
  threads distance=5em,
  thread 0/.append style={every thread header/.style={alias=thread-0}},
  baseline=(thread-0.base),
  every rf label/.style={anchor=south, sloped},
  event e0/.change eiid=to {e\textsubscript{0}},
  event e1/.change eiid=to {e\textsubscript{1}},
  events 1-2/.style={eiids order={e0,e1}},
% edge routing:
  (c) rf (e0)/.style={label pos=0.65},
  (c) rf (d)/.style={out=0, in=180},
  event init-e1/.style={left=4em of e1},
  (init-e1) rf (e1)/.style={every rf label/.style={anchor=north, sloped}},
  (e1) fr (a)/.style={out=-165,in=-150,looseness=2.2},
]{events}{A64/MP+dmbsy+misaligned2+1}
\hspace{1em}
\inputlitmustable[%
  mixed-size,
  thread 0/.append style={every thread header/.style={alias=thread-0}},
  baseline=(thread-0.base),
  compute assem text width=example10b,
  every assem/.append style={
    inner xsep=1pt,
    text width=computedwidth,
  },
  events 1-2/.style={eiids order={e0,e1}},
]{assem}{A64/MP+dmbsy+misaligned2+1}
```



MP+dmbsy+misaligned2+1 AArch64

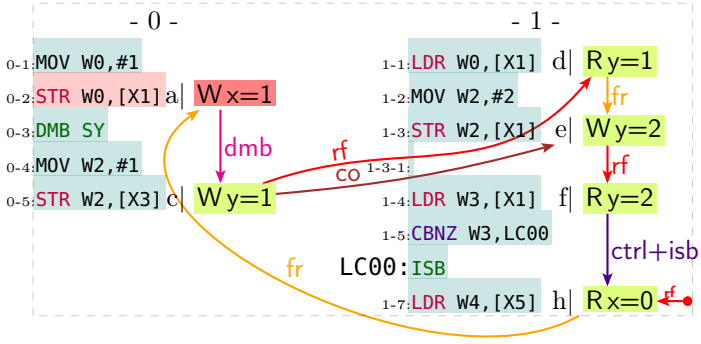
Thread 0	Thread 1
STRB W1,[X5,#1] //a	LDRB W1,[X5,#2] //d
DMB SY //b	LDRH W2,[X5,W6,SXTW] //e0,e1
STRB W2,[X5,#2] //c	

Initial state: 0:X2=34; 0:X1=17; 0:X5=x; 1:X6=1; 1:X5=x; x=0;

Allowed: 1:X1=34; 1:X2=8704;

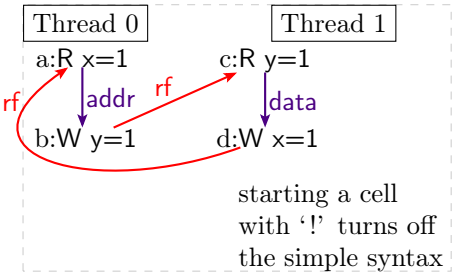
8.11 A crazy full example to show more options

```
% Example 11.
\inputlitmuspicture[%
  threads distance=5em,
  % every node/.style=draw, % darw a box around all nodes
  every instructions/.style={column sep=0, row 1/.style={every assem/.append style={text height=2ex}}},
  add row after={1-3}{},
  % modify all the assembly nodes:
  every assem/.append style={fill=teal, opacity=0.2, text opacity=1,
    inner xsep=1pt,
    minimum height=2ex,
  },
  every thread/.style={thread header text={- #1 -}},
  % modify the assembly node of specific instruction:
  assem 0-2/.style={fill=red},
  % modify the label of assembly instructions:
  code label/.code 2 args={%
    \if\relax\detokenize{#1}\relax%
      % if the location has no label we can use the ioid (#2)
      {\tiny #2:}%
    \else%
      % the location has a label (#1)
      \texttt{#1:}%
    \fi%
  },
  code label position=left,
  % modify all events:
  every event/.style={fill=lime, opacity=0.5, text opacity=1, inner sep=1pt},
  % modify specific event:
  event a/.style={fill=red},
  event label/.code={{\color{black}#1|~}}, % or, remove the label: 'event label/.code='
  every event relations/.style={on background layer},
  every rf label/.style={anchor=south, sloped},
  every dmb/.style={magenta}, % alternative: \colorlet{dmb color}{magenta}
% edge routing:
  hide po,
  every init/.style={node distance=1em, right=of #1},
  (1-4) isb (1-7)/.style=hide,
  (1-4) ctrl (1-7)/.style={label text=ctrl+isb},
  (c) rf (d)/.style={label pos=0.2, out=20, in=-130, looseness=1.1},
  (c) co (e)/.style={bend right=5, label pos=0.3},
  (d) fr (e)/.style={vertical align=1-4},
  (e) rf (f)/.style={vertical align=1-4, every rf label/.style={}},
  (h) fr (a)/.style={out=-150, in=-145, swap},
  (init-h) rf (h)/.style={label text=\tiny rf},
]{full}{A64/MP+dmb.sy+fri-rfi-ctrlisb}
```



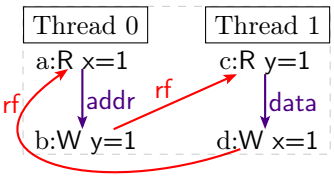
8.12 A simple hand written events example

```
% Example 12a.
\begin{tikzpicture}[/litmus/append to tikz path search,kind=events]
  % Thread 0
  \node[simple instructions] {
    % <eiid>:<text> \\
    a:R x=1 \\
    b:W y=1 \\
  };
  \draw[vertical align] (a) edge[addr] (b);

  % Thread 1
  \node[simple instructions] {
    c:R y=1 \\
    d:W x=1 \\
    !\node[text width=8em] {starting a cell with '!' %
                            turns off the simple syntax}; \\
  };
  \draw[vertical align] (c) edge[data] (d);

  \draw (d) edge[rf] (a)
        (b) edge[rf] (c);
\end{tikzpicture}
```



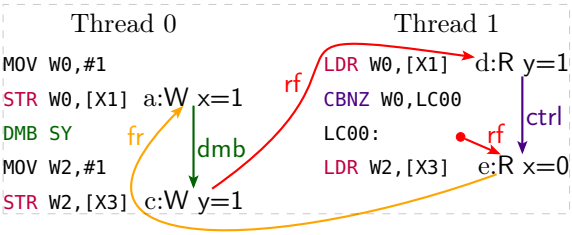
The above example is very easy to write but it hides some of the underlying TikZ elements. This makes it harder to customise them. The following example exposes those elements.

```
% Example 12b.
\begin{tikzpicture}[/litmus/append to tikz path search,kind=events]
  \node[instructions=0] (thread 0) {
    \node[event=a] {R x=1}; \\
    \node[event=b] {W y=1}; \\
  };
  \draw[vertical align] (a) edge[addr] (b);
  \node[thread header, above=of thread 0] {Thread 0};

  \node[instructions=1] (thread 1) {
    \node[event=c] {R y=1}; \\
    \node[event=d] {W x=1}; \\
  };
  \draw[vertical align] (c) edge[data] (d);
  \node[thread header, above=of thread 1] {Thread 1};

  \draw (d) edge[rf] (a)
        (b) edge[rf] (c);
\end{tikzpicture}
```



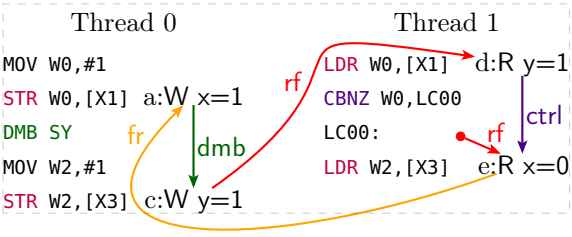
8.13 A simple hand written full example

```
% Example 13a.
\begin{tikzpicture}[/litmus/append to tikz path search,kind=full]
  % Thread 0
  \node[simple instructions] {
    % |<assembly>| & <eiid>:<text> \\
    |MOV W0,#1  |            \\
    |STR W0,[X1]|  & a:W x=1 \\
    |DMB SY     |            \\
    |MOV W2,#1  |            \\
    |STR W2,[X3]|  & c:W y=1 \\
  };
  \draw[vertical align] (a) edge[dmb] (c);

  % Thread 1
  \node[simple instructions] {
    |LDR W0,[X1] | & d:R y=1 \\
    |CBNZ W0,LC00|           \\
    |LC00:       |           \\
    |LDR W2,[X3] | & e:R x=0 \\
  };
  \draw[vertical align] (d) edge[ctrl] (e);

  \draw (c) edge[rf, out=35, in=172, out looseness=2, in looseness=2.2, label pos=0.3] (d)
        node[init=e] {} edge[rf] (e)
        (e) edge[fr, out=-160, in=-135, looseness=1.8, label pos=0.9] (a);
\end{tikzpicture}
```



The above example is very easy to write but it hides some of the underlying Ti*k*Z elements. This makes it harder to customise them. The following example exposes those elements.

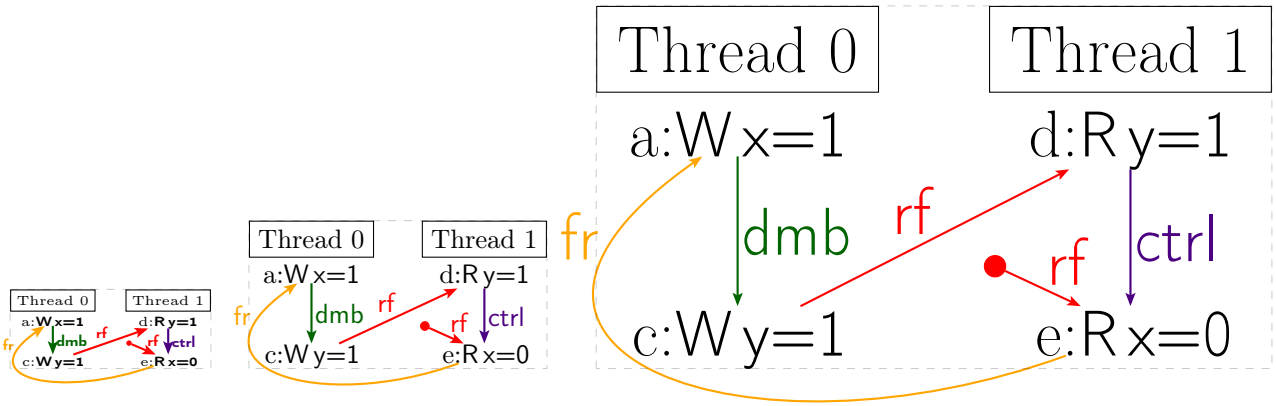
```
% Example 13b.
\lstset{language=AArch64, style=litmus, basicstyle=\footnotesize\ttfamily}%
\begin{tikzpicture}[/litmus/append to tikz path search,kind=full]
  \node[instructions=0] (thread 0) {
    \node[assem] {\lstinline|MOV W0,#1  |};                          \\
    \node[assem] {\lstinline|STR W0,[X1]|};  & \node[event=a] {W x=1}; \\
    \node[assem] {\lstinline|DMB SY     |};                          \\
    \node[assem] {\lstinline|MOV W2,#1  |};                          \\
    \node[assem] {\lstinline|STR W2,[X3]|};  & \node[event=c] {W y=1}; \\
  };
  \draw[vertical align] (a) edge[dmb] (c);
  \node[thread header, above=of thread 0] {Thread 0};

  \node[instructions=1] (thread 1) {
    \node[assem] {\lstinline|LDR W0,[X1] |}; & \node[event=d] {R y=1}; \\
    \node[assem] {\lstinline|CBNZ W0,LC00|};                         \\
    \node[assem] {\lstinline|LC00:       |};                         \\
    \node[assem] {\lstinline|LDR W2,[X3] |}; & \node[event=e] {R x=0}; \\
  };
  \draw[vertical align] (d) edge[ctrl] (e);
  \node[thread header, above=of thread 1] {Thread 1};

  \draw (c) edge[rf, out=35, in=172, out looseness=2, in looseness=2.2, label pos=0.3] (d)
        node[init=e] {} edge[rf] (e)
        (e) edge[fr, out=-160, in=-135, looseness=1.8, label pos=0.9] (a);
\end{tikzpicture}
```



8.14 Scaling

The diagrams scale with the text automatically:

```
% Example 14a.
\foreach \s [count=\c] in {\tiny,\normalsize,\Huge} {%
  \s%
  \ifx\c\one\else\hfill\fi%
  \inputlitmuspicture{events}{A64/MP+dmb.sy+ctrl}%
}
```



Notice that in §7 we globally set `basicstyle=\footnotesize\ttfamily`. Therefore, the assembly text will not scale with the surrounding text. Instead, we have to explicitly reset this key with the appropriate size as shown below.

```
% Example 14b.
\foreach \s/\t [count=\c] in {\scriptsize/\tiny,\normalsize/\footnotesize,\huge/\Large} {%
  \s%
  \ifx\c\one\else\hfill\fi%
  \inputlitmuspicture[%
    listing options/.append={,basicstyle=\t\ttfamily},% <-- SET ASSEM SIZE
    every thread header/.style={inner sep=0},
    header to instructions distance=0.5ex,
    threads distance=1pt,
  ]{assem}{A64/MP+dmb.sy+fri-rfi-ctrlisb}%
}
```

Thread 0	Thread 1
MOV W0,#1	LDR W0,[X1]//d
STR W0,[X1]//a	MOV W2,#2
DMB SY//b	STR W2,[X1]//e
MOV W2,#1	LDR W3,[X1]//f
STR W2,[X3]//c	CBNZ W3,LC00
	LC00:
	ISB//g
	LDR W4,[X5]//h

Thread 0	Thread 1
MOV W0,#1	LDR W0,[X1]//d
STR W0,[X1]//a	MOV W2,#2
DMB SY//b	STR W2,[X1]//e
MOV W2,#1	LDR W3,[X1]//f
STR W2,[X3]//c	CBNZ W3,LC00
	LC00:
	ISB//g
	LDR W4,[X5]//h

Thread 0	Thread 1
MOV W0,#1	LDR W0,[X1]//d
STR W0,[X1]//a	MOV W2,#2
DMB SY//b	STR W2,[X1]//e
MOV W2,#1	LDR W3,[X1]//f
STR W2,[X3]//c	CBNZ W3,LC00
	LC00:
	ISB//g
	LDR W4,[X5]//h

Alternatively, you can simply use \scalebox{<factor>}{<litmus>}:

```
% Example 14c.
\foreach \s [count=\c] in {0.5,1,2} {%
  \ifx\c\one\else\hfill\fi%
  \scalebox{\s}{\inputlitmuspicture{events}{A64/MP+dmb.sy+ctrl}}%
}

\vspace{3ex}
\foreach \s [count=\c] in {0.5,1,2} {%
  \ifx\c\one\else\hfill\fi%
  \scalebox{\s}{%
    \inputlitmuspicture[%
      every thread header/.style={inner sep=0},
      header to instructions distance=0.5ex,
      threads distance=1pt,
    ]{assem}{A64/MP+dmb.sy+fri-rfi-ctrlisb}%
  }%
}
```