# Visualiser Documentation

Ruben Saunders

November 2024

## Contents

# 1 Overview

The visualiser offers an interactive view into code execution and the interaction between the compiler, the assembler, and the processor. The visualiser

- Is machine-code executor, able to run the loaded binary on the processor in a step-wise fashion.
- Has a register view/editor.
- Has a memory view/editor.
- Is a basic debugger.
- Shows the current machine-code instruction, and back-traces it to both the assembly and high-level source code.

The application is a TUI (a textual user interface), meaning it runs in the terminal. The interface is tab-based, with a navigation bar along the top. Later sections will cover the content of each tab.

## 1.1 Command-Line Interface

The visualiser executable is called as follows:

```
$ ./visualiser --asm <asm_file> --bin <bin_file> [flags]
```

The `--asm` and `--bin` flags set the source assembly and binary files, respectively. The file provided to `--asm` is the output from the assembler's reconstruction; if not, features such as trace-back will not be enabled.
The following flags are optional:

- `--stdout <file>` – pipe the processor's output to this file.
- `--stdin <file>` – set the processor's input stream to this file.

## 1.2 Execution Example

The visualiser relies on output from the assembler. Below is the sequence of steps to go from a source assembly file, to running it on the visualiser.

1. Write an assembly source, say `source.asm`.
2. Assemble the file, emitting a reconstruction.

   ```
   $ ./assembler source.asm -o source -r source.s
   ```

3. Launch the visualiser.

   ```
   $ ./visualiser --asm source.s --bin source
   ```

## 1.3 Navigation & Control Basics

As a TUI, everything may be done using keyboard controls. Navigation between key elements is done with the arrow keys (e.g., navigating between tabs, selecting registers in the list).
To use the tab-specific controls, the tab body must be focused. This can be achieved by pressing the down arrow. Aside from tab-specific controls, the following are global:

- `Ctrl+c` – exits the application.
- `F`$n$ – selects the $n^{th}$ tab.

# 2 Code Execution

The "main" tab, this view shows the source and compiler assembly side-by-side.

## 2.1 Layout

The top of the screen is split into two panes.

- The assembly source is displayed on the left, with its file name displayed.
- The reconstructed compiler assembly is display on the right.

The current line pointed to by $pc is highlighted in yellow, which is traced back to the source assembly. If enabled, the currently selected line is displayed in cyan, with the corresponding equivalent lines highlighted in light cyan in the other pane.
Below this is a section for the processor's debug messages, which is hidden by default unless messages are available. The fields below this display the

- Processor's status – either *halted* or *running*. This reflects the `is_running` bit in $flag.
- Cycle – indicates the processor's current cycle, i.e., number of executed steps.
- Program counter – the current value of $pc.

## 2.2 Keyboard Controls

- `End` – if line selection is enabled, set the selected line to the end of the file.
- `Enter` – if the processor is running, perform a step on the processor, i.e., execute the instruction at $pc.
- `h` – toggles the `is_running` bit in $flag.
- `Home` – if line selection is enabled, set the selected line to the start of the file.
- `j` – if line selection is enabled, sets $pc to the first selected line in the compiled assembly.
- `r` – resets $flag: sets `is_running` and clears any error bits.
- `s` – toggle line selection.
- `Up/Down arrows` – if line selection is enabled, moves the active selected line (cyan) up/down in the current file.
- `Left/Right arrows` – if line selection is enabled, scrolls through the current panes. The active selected line (cyan) is transferred to the topmost selected line in the new pane.

# 3 Registers

This view shows a list of the processor's registers.

## 3.1 Layout

The left-hand side of the view displays a list of the processor's registers, accompanied by their 64-bit hexadecimal value. When a register is selected, more information about that register is display in a pane on the right. This pane lists the register's value in hexadecimal, as a decimal integer (32-bit) and long (64-bit), as a float, and as a double. Each may be edited and updated. A brief description of the register is given below this.
The $flag register has an additional section below this. $flag contains state information about the processor, which are listed in a decomposed form here.

### 3.2 Keyboard Controls

When an input field is selected, controls work as they ordinarily would in an input box, overriding the below.

- `Backspace/Delete` – zeroes the current register.
- `c` – copies the contents of the current register.
- `r` – refresh all registers, and the selected register's pane. Useful when a register value has been updated in another tab, and changes have not been loaded.
- `Up/Down arrows` – scroll in the register list.
- `v` – paste the copied register's value into the current register.