DrMIPS – User Manual

Bruno Nova

September 17, 2015

Contents

Chapter 1

Introduction

DrMIPS is a simulator of the MIPS processor. It allows you to follow the execution of an assembly program step-by-step. A graphical representation of the datapath allows you to see how the processor works internally.

This manual is a short guide on how to use the simulator, focusing on the PC version. Chapter 2 provides an overview of the user interface. Chapter 3 explains the interface of the tabs/windows of the simulator and how to use them.

Chapter 2

Composition of the User Interface

The graphical user interface of the DrMIPS simulator is composed by the menu bar, the toolbar and the area where the main contents of the simulator are displayed. The icons in the toolbar are shortcuts for some frequently used actions in the menus. You can hover the mouse cursor over each icon to find out what it does.

The interface is displayed with a light theme by default. But you can switch to the dark theme by selecting ***View*** > ***Dark theme*** in the menu. DrMIPS supports multiple languages, so these names may be different. You can choose another language in the ***View*** > ***Language*** menu.

The main contents of the simulator are split in tabs by default. Each tab can be positioned on the left side of the window or on the right side. You can move a tab to the other side by right-clicking on it and selecting ***Switch side*** in the menu that appears. If you would prefer to see the contents split in windows instead, you can do that by selecting ***View*** > ***Internal windows*** in the menu. The position and size of the tabs/windows are remembered on exit.

The tabs or windows displayed are:

- **Code**: the code editor, where you can create or edit an assembly program to be executed by the simulator.
- **Assembled**: after the code is assembled successfully, this displays the resulting machine instructions.
- **Registers**: lists all the registers and their values.
- **Data memory**: displays all the values in the data memory.
- **Datapath**: the graphical representation of the datapath, and its state, is displayed here.

Chapter 3

Tabs/Windows

3.1 Code

This is the code editor, where you can write an assembly program. The editor displays the numbers of the lines and highlights the valid syntax.

You can undo/redo changes, cut/copy/paste text and find/replace words in the code. These actions can be accessed through the ***Edit*** menu or by right-clicking in the code editor. The code can be saved to or read from a file. These actions are available in the ***File*** menu.

You can press *Ctrl+Space* to auto-complete the keyword you are writing. Doing so displays a list of instructions, pseudo-instructions, directives and labels that can complete that keyword. It also displays a window that shows what the selected keyword in the list does and how it is used. Pressing *Ctrl+Space* on an empty space will list all available instructions, pseudo-instructions, directives and labels. You can also view this information in the ***CPU*** $>$ ***Supported instructions*** menu.

After writing your assembly program, you must assemble it. This is done by selecting ***Execute*** $>$ ***Assemble*** from the menu or by pressing the respective button in the toolbar. If the program has errors, a message box will indicate the first error and an exclamation icon will appear next to the numbers of the lines that have an error. You can hover the mouse cursor over those icons to discover what is the error. If the program is correct, no error message is displayed and you can then proceed to execute it.

3.2 Assembled

After the code is successfully assembled, the resulting machine code instructions will be displayed here in a table.

Each line of the table corresponds to an assembled instruction, and it contains its address, machine code and original instruction. The instruction currently being executed is highlighted.

If simulating a pipeline processor, all the instructions that are in the pipeline are highlighted in different colors, each one representing a different stage. The different colors mean:

- **Blue**: **I**nstruction **F**etch (**IF**) stage.
- **Green**: **I**nstruction **D**ecode (**ID**) stage.
- **Yellow**: **Ex**ecute (**EX**) stage.
- **Red**: **Mem**ory access (**MEM**) stage.
- **Magenta**: **W**rite **B**ack (**WB**) stage.

Hovering the mouse cursor over an instruction in the table will present a tooltip. This tooltip displays the type of the instruction and the values of its fields. [1]

The values are shown in decimal format by default. You can change that format to binary or hexadecimal by using the combo box at the bottom of the tab/window. This combo box is available in all tabs/windows, except in the code tab/window.

To control the simulation, you can use the ***Execute*** menu or the toolbar. Press ***Step*** to execute one instruction, ***Back step*** to revert one instruction, ***Run*** to execute the entire program and ***Restart*** to revert to the first instruction.

3.3 Registers and Data Memory

These two tabs/windows are very similar. The registers tab/window displays the values of the registers and the program counter, while the data memory tab/window displays the values in the data memory.

The values that are currently being accessed are highlighted in different colors. The colors mean:

- **Green**: the register/address is being read by the register bank/data memory.
- **Red**: the register/address is being written to the register bank/data memory.
- **Yellow**: the register/address is being read and written in the same cycle in the register bank/data memory.

[1] In the Android version, press the instruction to see its tooltip.

You can edit the value of any register or memory address by double-clicking it in the respective table. This includes the program counter. Constant registers (like the register `$zero`) cannot be edited. [2]

By default, the values of the registers and data memory are reset every time the code is assembled. If you don't want that to happen, uncheck ***Execute*** > ***Reset data before assembling*** in the menu.

3.4 Datapath

The graphical representation of the processor's datapath is displayed here. This is where you can see how the CPU works internally.

DrMIPS can simulate several different unicycle and pipeline datapaths. The name of the datapath currently being used is shown in the bottom of this tab/window. You can choose another datapath by selecting ***CPU*** > ***Load*** from the menu. Note that different datapaths may support different instructions.

The datapaths provided by default are:

- **Unicycle datapaths**
  - **unicycle.cpu**: The default unicycle datapath.
  - **unicycle-no-jump.cpu**: Simpler variant of the unicycle datapath that doesn't support the `j` (*jump*) instruction.
  - **unicycle-no-jump-branch.cpu**: An even simpler variant of the unicycle datapath that doesn't support *jumps* nor *branches*.
  - **unicycle-extended.cpu**: A variant of the unicycle datapath that supports some additional instructions, like multiplications and divisions.
- **Pipeline datapaths**
  - **pipeline.cpu**: The default pipeline datapath, which implements hazard detection. The pipeline datapaths don't support the `j` (*jump*) instruction.
  - **pipeline-only-forwarding.cpu**: Variant of the pipeline datapath that, in terms of hazard detection, only implements data forwarding (giving wrong results).
  - **pipeline-no-hazard-detection.cpu**: A variant of the pipeline datapath that doesn't implement any form of hazard detection at all (giving wrong results).
  - **pipeline-extended.cpu**: A variant that supports some additional instructions, like *unicycle-extended.cpu*.

[2] In the Android version, long-press the register/address to edit it.

On the top of the tab/window, the instruction or instructions currently being executed are displayed. They are highlighted with the same colors explained in section 3.2.

The datapath is displayed below the instructions. The components are represented by rectangles or squares, and the wires by lines that end with arrows. Wires that are in the control path are shown in blue. The control path can be hidden by unchecking ***Datapath > Control path*** in the menu.

Wires that are considered irrelevant in the current clock cycle are shown in gray. A wire is considered irrelevant if it's a control signal set to `0`, if its value is ignored by a component, if it's the output of the *hazard detection unit* and a stall isn't occurring, etc. You can hide the arrows in the end of the wires by unchecking ***Datapath > Arrows in wires***.

The values at some important inputs and outputs of some components are shown in the datapath as small "tips" with yellow background. You can hover the mouse cursor over these "tips" to find out what is the identifier of the input/output. The ***Datapath > Data in inputs and outputs*** menu contains some options to control these "tips": ***Enable*** to show/hide them, ***Show names*** to display the names of the inputs/outputs and ***Show for all components*** to display the "tips" in all components.

By hovering the mouse cursor over a component, a tooltip with some details about it will be displayed [3]. The tooltip presents the name of the component, a description of what it does and the values at all the inputs and outputs.

The datapath can be zoomed in and out. This can be accomplished by the ***Datapath > Zoom in***, ***Datapath > Zoom out*** and ***Datapath > Normal*** options in the menu or by the respective buttons in the toolbar. The zoom level can also be adjusted automatically to fit all of the available space by using the ***Datapath > Adjust automatically*** option.

The datapath can also be displayed in a "performance mode". You can switch to this mode by selecting ***Datapath > Performance mode*** in the menu. In this mode, the performance of the processor is simulated, and the critical path is displayed in red.

You can see either the critical path of the instruction currently being executed or the global critical path of the CPU (instruction independent). The combo box at the bottom of the tab/window is used to choose between these two options.

Each component has a latency, which can be consulted in its tooltip. The tooltip also shows the accumulated latencies at the inputs (the time it takes for that input to receive the correct value after the clock transition) and at the outputs (the time it takes for the component to generate the correct value for the output).

[3] In the Android version, press the component to see its tooltip.

The latencies of the components can be edited by double-clicking them while in performance mode [4]. Additionally, you can select ***Datapath*** > ***Restore latencies*** in the menu to restore the latencies of all components to their original values, and ***Datapath*** > ***Remove latencies*** to set all latencies to `0`.

You can also view some statistics about the simulation, like clock frequency and CPI (*Cycles Per Instruction*), by selecting ***Datapath*** > ***Statistics*** in the menu.

[4] In the Android version, long-press the component while in performance mode to edit its latency.