




POWERsim: A nanoPOWER ISA Functional Simulator

Samar Musthafa^{*1}, Mathew Joseph Kayyalackakom^{†1}, Adithya Gopakumar^{‡1}, Mohammed Siyad B^{§2}, and Basavaraj Talavar^{¶3}

¹ B. Tech Final year Student, TKM College of Engineering, Kollam ² Department of Computer Science and Engineering, TKM College of Engineering, Kollam ³ Department of Computer Science and Engineering, NITK, Surathkal

DOI: [10.21105/joss.03973](https://doi.org/10.21105/joss.03973)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Jarvist Moore Frost](#) 

Reviewers:

- [@dilawar](#)
- [@federeghe](#)

Submitted: 22 September 2021

Published: 03 December 2021

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

Summary

The Power ISA ([OpenPOWER Foundation, n.d.](#)) is an instruction set architecture (ISA) developed by the OpenPOWER Foundation, led by IBM. An Instruction Set Architecture ([Wikipedia, n.d.](#)) is essentially an abstract model or documentation that defines the instructions, data types, features, hardware support, etc for a particular hardware device capable of executing it. The relevance of an ISA as a whole is that understanding one aids research and development in assembly level applications on a specific hardware board. POWER10 or POWER ISA v3.1 is the latest version of POWER ISA. Power ISA is a RISC load/store architecture. It has multiple sets of registers. Instructions have a length of 32 bits, with the exception of the VLE (variable-length encoding) subset. Most instructions are triadic (Two sources and one destination). Memory operations are strictly load/store, but allow for out-of-order execution. With OpenPOWER joining the linux foundation, the need for an open source hardware-independent POWER ISA simulator has also come up. POWER ISA was made open source by IBM in August 2019. The currently existing solution for the given problem depends on an actual hardware target board or a ghdl simulator that is relatively more tedious to set up and get into. As can be inferred, the proposed software would be a hardware simulation project mainly focusing on providing a software that helps develop microprocessor assembly code for testing POWER ISA programs and developing applications following its open-sourcing.

Statement of need

The main goal of the proposed project is to develop a platform independent, open source POWER ISA functional simulator with minimal hardware requirements and aims to provide a sandbox for executing a reduced POWER ISA instruction set. The applications of proposed simulator would include the ability to test and develop POWER ISA based assembly code. Apart from the mentioned use-case of testing and developing assembly code, the software is also capable of being used as a teaching and learning tool for better understanding POWER ISA and how it works. Simply put, the target audience comprises of research personnel looking to test the ISA and learn it as well as developers aiming to test their assembly code. The niche nature of the ISA as well as the lack of a hassle-free open source simulator work in favor of the project. The educational aspect of the simulator is very significant as most of the

^{*}co-first author

[†]co-first author

[‡]co-first author

[§]project guide

[¶]project guide

open source ISAs have a very powerful simulators associated with it. Such a software further enables research and advancement of POWER ISA based applications. The proposed tool will simulate POWER ISA as well as provide register and memory access.

We aim to build a system to simulate POWER ISA architecture completely independent of the installed system's hardware. Essentially, everything including the memory devices will be simulated file objects that are used to serve this purpose. There are currently two other widely accepted software that simulates POWER ISA; Microwatt(Anton Blanchard, 2021) and Power10 Functional Simulator(IBM, n.d.). MicroWatt is an open source tool capable of executing POWER ISA assembly code. But, this software is written in VHDL and relies on a hardware board or a GHDL simulator to execute the code. Simply put, setting the tool up is tedious. Another alternative is Power10 Functional Simulator developed by IBM. The proposed software would be superior due to the closed-source nature of Power10 Functional Simulator. Our platform independent approach aims at providing software to help learn as well as develop POWER ISA based applications. With the rise in popularity of hardware accelerators in modern applications, an ISA simulator for the said architecture would create an environment where the applications can be tested before being deployed to specially designed FPGA chips or Hardware components for their intended purpose. This means that researchers and developers can both test and debug their assembly code before running it on actual hardware board. The process consists of two phases, the Assembler development and the Processor Development. The Assembler converts instructions to Hex code, handles assembler directives and comments and is a two pass assembler. The Processor simulates instruction execution and executes using a python(Van Rossum, Guido, 2021) based processor. We use Argparse(Foundation, 2021) for CLI and Tkinter(Foundation, 2021) and cx_Freeze(Tuininga, 2021) for GUI development.

Installation Instructions

Installation

- clone the repo

```
git clone git@github.com:god-s-perfect-idiot/POWER-sim.git
```

- install dependency packages

```
pip install argparse csx_Freeze tk
```

- generate executable package on windows if needed (Optional)

```
python3 generate_windows_exec.py build
```

- Launch modes:

- Run /build/gui.exe file for executing POWERSim
- run gui.py for gui mode

```
python3 launchers/gui.py
```

- run cli.py for cli mode

```
python3 launchers/cli.py -m [MODE] -i? [input] -o? [output]
```

Dependencies

- argparse
- Tkinter
- cx_Freeze

78 Example Usage

79 POWERSim has two different launch modes for accessing the application. - The GUI mode
80 launches an interactive session where the user can switch between assembly and processor
81 modes. The interactive console provides compiled binary results and the memory devies and
82 syntabs will be viewable as well.

83 GUI:

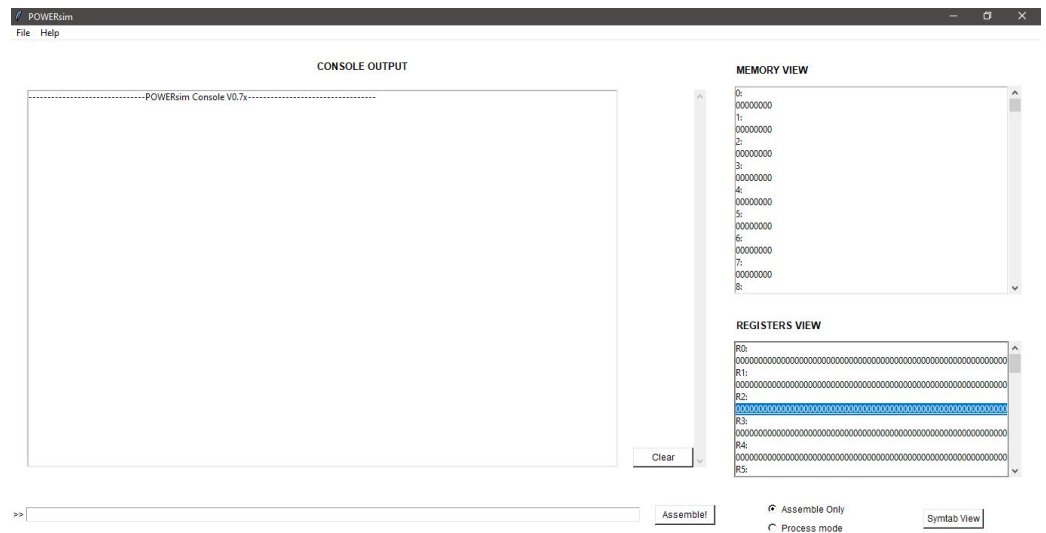


Figure 1: Capture

- 84 ▪ The CLI mode is exclusive to an execution mode and is capable of executing or assem-
85 bling instructions. It can also accept an input and output file.

86 An example use case is as follows:

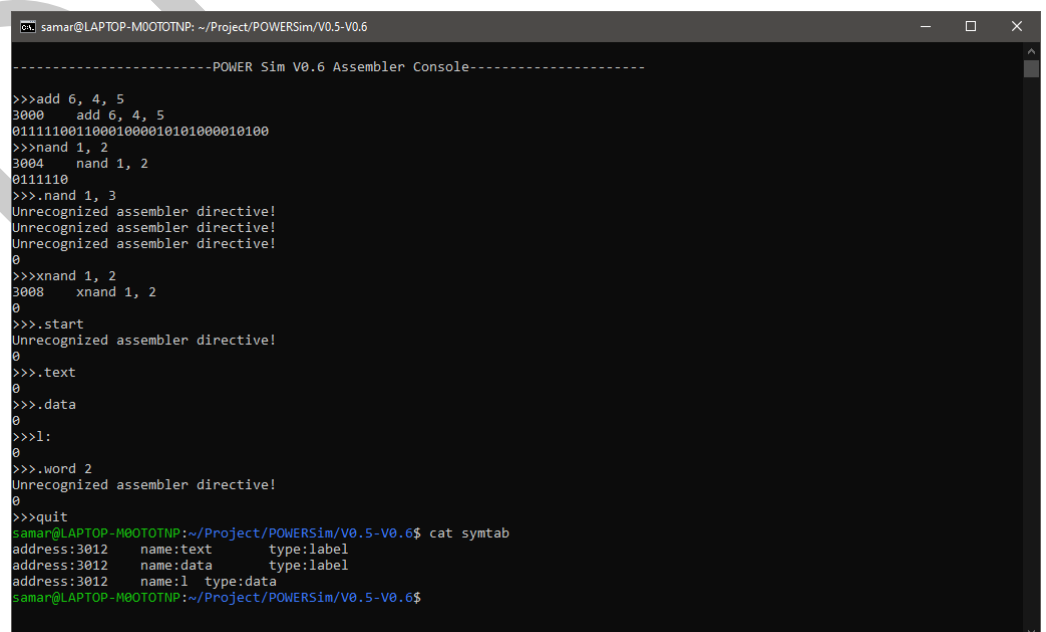


Figure 2: V0 6

Functionality Documentation

POWERSim has two launch options: A GUI mode and a CLI mode. Both have their own different usages and interfaces.

CLI Mode:

Launch CLI mode using the following command:

```
cd launchers
python3 cli.py -m [OPERATION MODE] -i [INPUT FILE] -o [OUTPUT FILE]
```

- Here, -m accepts two values.

The two operation modes are Assembler mode and Processor Mode.

Assembler Mode:

Assembles instructions and returns the binary value of the instruction

Processor Mode:

Processes instruction. Assembles first and then executes the instruction.

- The usage of -i is to provide an input file.

Any file of the type .s that contains input instructions can be assembled/ processed.

- The usage of -o is to provide an output file.

The response of the execution is stored there in the output file.

GUI Mode:

Launch GUI using the Python script or generated windows executable.

```
# script
cd launchers/
python3 gui.py

#generated exe
python3 generate_windows_exec.py build
# file is in /build
```

GUI Mode has the following sections:

Live console:

Assemble or process the given instruction and print the response in the console

116 Assemble-Process switch:
117 Switch between assembly and process modes

118 Symtab/Memory view:
119 Show the contents of the Symbol Table and the Memory

120 Memory/Symtab switcher:
121 Button to toggle between either devices

122 Register View:
123 Display content of the register.

124 Load an external File using the File Menu.
125 An example assembly code can be found in the file `/simple.s`
126 For detailed per-method documentation, please refer to [documentation.md](#)

127 Supported Instructions:

128 POWERsim currently supports a reduced instruction set of POWER ISA compiled and titled
129 [nanoPOWERISA](#). It also, only supports 16 memory bits. This shall be expanded upon in
130 future updates.

131 Acknowledgements

132 We acknowledge contributions from Samar Musthafa, Mathew Joseph Kayyalackakom and
133 Adithya Gopakumar during the development of the project. Our guides, Prof. Mohammad
134 Siyad B and Prof. Basavaraj Talavar also played significant roles during the ideation and
135 subsequent development reviews of the project.

136 Contributions

137 We welcome all types of contributions, either code fixes, new features or doc improvements.
138 For commits please follow conventional commits convention. All code must pass lint rules
139 before it can be merged into main.

140 References

- 141 Anton Blanchard. (2021). Microwatt. In *GitHub repository*. Github. <https://github.com/antonblanchard/microwatt>
142
143 Foundation, P. S. (2021). CPython: The python programming language. In *GitHub repository*.
144 Github. <https://github.com/python/cpython>
145 IBM. (n.d.). *Power10 functional simulator*. <https://openpowerfoundation.org/introducing-ibm-power10>

- 146 OpenPOWER Foundation. (n.d.). *Power ISA Manualv3.0B*. [https://ibm.ent.box.com/s/](https://ibm.ent.box.com/s/1hzcwkwf8rbju5h9iyf44wm94amnlcrv)
147 [1hzcwkwf8rbju5h9iyf44wm94amnlcrv](https://ibm.ent.box.com/s/1hzcwkwf8rbju5h9iyf44wm94amnlcrv).
- 148 Tuininga, A. (2021). In *GitHub repository*. GitHub. [https://github.com/marcelotduarte/cx_](https://github.com/marcelotduarte/cx_Freeze)
149 [Freeze](https://github.com/marcelotduarte/cx_Freeze)
- 150 Van Rossum, Guido. (2021). *The python library reference, release 3.8.2*. Python Software
151 Foundation.
- 152 Wikipedia. (n.d.). *Instruction set architecture*. [https://en.wikipedia.org/wiki/Instruction_](https://en.wikipedia.org/wiki/Instruction_set_architecture#:~:text=In%20computer%20science%2C%20an%20instruction,)%2C%20is%20called%20an%20implementation)
153 [set_architecture#:~:text=In%20computer%20science%2C%20an%20instruction,\)%2C%](https://en.wikipedia.org/wiki/Instruction_set_architecture#:~:text=In%20computer%20science%2C%20an%20instruction,)%2C%20is%20called%20an%20implementation)
154 [20is%20called%20an%20implementation](https://en.wikipedia.org/wiki/Instruction_set_architecture#:~:text=In%20computer%20science%2C%20an%20instruction,)%2C%20is%20called%20an%20implementation).

DRAFT