Evaluating RISC-V Instructions Natively with Narvie

Harry Sarson,* Ryan Voo,† and Phillip Stanley-Marbell
University of Cambridge

Developing systems software for new hardware architectures is challenging. Developing low-level embedded system firmware or low-level operating system software while evolving a processor architecture is harder still. Narvie is a *read eval print loop (REPL)* that runs on a host PC and provides an interactive interface to an open-source RISC-V (RV32I) processor design running in an FPGA. Narvie allows operating systems researchers and hardware architects to interact with a RISC-V processor implementation running in an iCE40 low-power low-pincount FPGA. Whilst RISC-V is gaining popularity, the tools available to work with the *instruction set architecture (ISA)* are often tailored to advanced users. When teaching computer architecture and low-level software, it can be a struggle to adapt powerful but complicated toolchains to the needs of students. Narvie exposes the RISC-V ISA in a very basic form and demonstrates in a concrete way what individual instructions do. By allowing evaluation of individual instructions, Narvie will help programmers gain an understanding of the ISA. Narvie will soon be released under a GPL licence at github.com/physical-computation/narvie.

1. What Does Narvie Do?

Running on a host computer, Narvie's user interface allows a user to enter assembly mnemonics via a terminal. Narvie converts these inputs into binary RISC-V instructions. Narvie sends the generated binary over a serial port to the FPGA as well as displaying it to the user. On the FPGA, a modified RV32I processor runs instructions received over the UART. When not executing an instruction, the processor's clock is held high. Then, once an instruction is received, that instruction is sent to the processor and the processor clock is made to complete one cycle. The processor has a five-stage pipeline and thus four more cycles are required to fully execute the instruction. By cycling the clock four more times whilst sending no-op instructions to the processor, the instruction is passed through the entire pipeline. Once the instruction is retired, all the registers are read and transmitted back to the host computer. Once it has read the 32 register values, Narvie displays them to the user. Narvie supports all instructions in the base RISC-V integer instruction set (47 instructions in total) and all corresponding pseudo instructions — except those that assemble into multiple instructions [2].

2. How does Narvie Perform?

The vast majority of the time taken for Narvie to evaluate an instruction is spent receiving the instruction and transmitting the registers. Table 1 summarises the duration of the three stages in both clock cycles and seconds whilst Figure 1 shows the distribution of measured latencies. Figure 1 shows that the

Task	Cycles	Time (ms)	Fraction (%)
Receive Instruction	2048	0.339	3.1
Evaluate Instruction	5	0.000833	0.01
Transmit Registers	64294	10.7	96.9
Total	66347	11.1	100

Table 1: RTL simulation cycle counts with a UART baud rate of 115200 bits/s. Timings are estimated given a clock frequency of 6MHz. The evaluation time is dominated by I/O overhead.

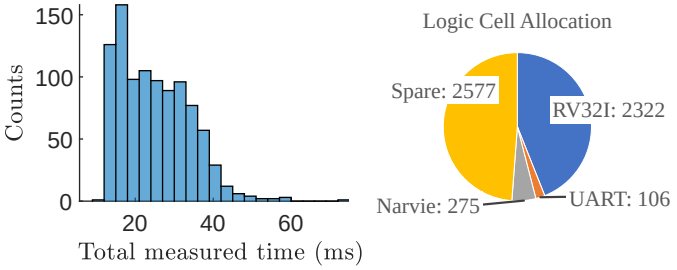

Figure 1: Left: Histogram showing the instruction evaluation time as measured by the user interface. Right: FGPA resources used for the processor, UART, Narvie and those that are spare. The resources Narvie requires are minimal.

FPGA resources required for Narvie and for the UART are small compared to those needed by the processor.

3. Why is Narvie Needed?

Whilst programs like rappel (Linux) and WinREPL (Windows) evaluate individual x86 instructions, no such program for RISC-V instructions currently exists. As all current assembly REPLs evaluate instructions natively, they can only support the system they are running on.

Computers running RISC-V are not widely available and so this project integrates custom hardware supporting RISC-V with software designed to run on a desktop computer. Programmers could use GNU binutils to work with RISC-V instructions, provided the version is greater than 2.28 [1]. However, these tools are too new to be released as part of package managers for stable Linux distributions and installing and building binutils is not a simple task. Furthermore, binutils is not straight-forward to use and available documentation is unfortunately quite sparse.

The software required for Narvie is simpler to run and clearly shows the relationship between RISC-V assembly and machine code.

References

[1] Free Software Foundation. *Binutils changelog*, 2017.
[2] A. Waterman, Y. Lee, D. Patterson, and K. Asanovic. The risc-v instruction set manual, volume i: User-level isa, version 2.0. Technical report, EECS Department, University of California, Berkeley, 2014.

*M.Eng. student, will be presenting.
†M.Eng. student.