Projects

Continuous integration and performance optimisation - Mutalib

The core of this project is centred around identifying and resolving areas of SimEng which exhibit poor performance and provide resolutions through researched optimisations. One of SimEng's core principles is to be fast yet simple to use, therefore, this project serves as an integral development for its upcoming open-source release. The project outline can be split into 3 sections.

Performance testing

Currently, our continuous integration pipeline supports functional testing, focusing only on the assurance that SimEng builds and runs correctly under various compilers. You'll be expected to expand this pipeline to support performance testing. These tests are envisioned to run a set of chosen programs through SimEng, on a specific piece of hardware, and detect regressions in performance metrics from previously measured values. Ideally, these tests would be expanded in the future to run on multiple pieces of hardware/platforms. We also have a performance goal to exceed the current industry standard processor simulator, gem5, by a factor of 3-5X, so comparative performance benchmarking against gem5 will be required.

During this period of the project, it's recommended that you get comfortable with C++17 so that you'll be confident with inspecting and optimising areas of code within SimEng later in the project.

Poor performance hot-spots

The creation of the performance testing suite will provide you with a methodology to detect areas of poor performance within SimEng. By inspecting the performance metrics across multiple commitments to the SimEng repository, developments that caused a performance regression can be identified. A selection of areas can be compiled through this method. You may wish to further identify/narrow the areas of poor performance through profiling techniques.

Optimisation

With a selection of areas that exhibit poor performance, the final stage of the project is to optimise them. The optimisations implemented will be up to you and can be quantified with the performance testing suite you created. This section of the project provides the opportunity to develop the core of the SimEng codebase which, if successful, will be kept and maintained moving forward.

Maintaining ease of use

One of SimEng's primary goals is for the code to be easy to read and modify. Any performance optimisations will need to balance speed with readability and maintainability.

Improving ARMv8-A ISA coverage - Seunghun & Finn

SimEng currently implements partial support for the ARMv8-A ISA (specifically armv8.4-a+sve), equating to ~10% of the instruction set (480 instructions). Rather than attempt to provide support for all instructions in the ISA, we provide support for those instructions required by compiled programs run through SimEng. The core of this project will be to further the coverage of the ARMv8-A ISA, providing the opportunity to develop the core of SimEng and explore the details of a real ISA.

Following are 3 methods you'll be expected to use to generate new instructions currently unsupported by SimEng. With SimEng planned to be released as open-source this summer, the methods detailed below have been chosen to have the greatest impact on simulated program coverage.

Benchmarks

A set of benchmarks that we would like to provide support to within SimEng have been chosen. You will be expected to develop support for a subset of these benchmarks by implementing the instructions they require.

Compilers

To generate more unique instructions from the benchmarks run through SimEng, multiple compilers are used. This set of compilers create differing binaries from the same source code. With the same goal as the previous method, expanding this compiler set will provide new instructions to develop support for. Candidates to expand this set include the NVIDIA, Cray and Fujitsu compilers.

Benchmark languages

The benchmarks currently supported by SimEng are all written in C. It's desirable to provide support for other languages, such as C++ and Fortran, to expand the range of programs that can be simulated. Therefore, the final method is to further the ARMv8-A ISA support to include those benchmarks not written in C. This final method may require developments beyond instruction implementation, providing the opportunity to explore more of the codebase.

Reproducibility

Throughout this project, you should aim to provide methods to reproduce the programs you have run through SimEng. Ideally, these methods would be in the form of a script that outputs a binary. This will help us continue the work you have done at a later date if required, and expand the set of benchmarks run through SimEng.

Supporting a new ISA in SimEng - Daniel

Whilst SimEng's current focus is the ARMv8-A ISA, the external library used to disassemble instructions (https://www.capstone-engine.org/) supports other ISAs. New users may wish to simulate new non-Arm architectures, therefore, providing support for other ISAs is desirable. The core of this project is to extend SimEng by developing support for a new ISA, namely RISC-V. RISC-V is an exciting, up and coming, open-source RISC ISA that is seeing use in several developing architectures, e.g. European Processor Initiative (EPI), RISC-V Accelerator Stream (https://www.european-processor-initiative.eu/accelerator/).

There are many components associated with a SimEng Architecture model, each of which will require implementation for the RISC-V ISA. You'll be expected to provide developments to support an instructions' lifecycle including:

- Fetching logic
- Designing an implementation of the SimEng instruction class, to best suit RISC-V ISA
- Implementing the decoding infrastructure for the RISC-V ISA
- Execution logic for each instruction decoded

There may also be opportunities to explore other areas of the SimEng codebase that require architecture-specific functionality.

Eventually, we'd like to provide support for a more complex ISA that is less similar to the ones currently implemented. A candidate for this would be Intel's x86, however, for this project, the primary focus should be RISC-V.

Overall, this project offers an opportunity to make developments within SimEng and serves as its first example of multiple ISA support.

Reproducibility

Any inputs used to develop the support of the RISC-V ISA in SimEng should be documented. The documentation can take the form of a piece of text or, ideally, a script that generates the input. This will help us continue the work you have done at a later date.