

[Document title]

[Document subtitle]



[Date]

[Company name]

[Company address]

Contents

[List of Figure 2](#_Toc32790909)

[Introduction 3](#_Toc32790910)

[SimpleScalar Overview 3](#_Toc32790911)

[Tool set 3](#_Toc32790912)

[Instruction set 3](#_Toc32790913)

[Platform 4](#_Toc32790914)

[Experiments 4](#_Toc32790915)

[Observation 7](#_Toc32790916)

[Comparison of performance 7](#_Toc32790917)

[First case 7](#_Toc32790918)

[Second case 7](#_Toc32790919)

[CPI Account 8](#_Toc32790920)

[Conclusion 8](#_Toc32790921)

[References 9](#_Toc32790922)

# List of Figure

[Figure 1: SimplScalar overview [1] 3](#_Toc32790923)

[Figure 2:/usr/local/3rdparty/simplescalar/simplesim-3.0/sim-profile -iclass go.ss 4 6 go.in 4](#_Toc32790924)

[Figure 3:/usr/local/simplescalar/simplesim-3.0/sim-profile –iclass applu.ss < applu.in 5](#_Toc32790925)

[Figure 4: /usr/local/simplescalar/simplesim-3.0/sim-profile –iclass apsi.ss 5](#_Toc32790926)

[Figure 5: /usr/local/simplescalar/simplesim-3.0/sim-profile –iclass cc1.ss –O2 cc1.in –o test.s 6](#_Toc32790927)

[Figure 6:/usr/local/simplescalar/simplesim-3.0/sim-profile –iclass compress95.ss < compress95.in 6](#_Toc32790928)

[Figure 7: Performance of the sample scalar [3] 7](#_Toc32790929)

[Figure 8: given data 7](#_Toc32790930)

# Introduction

The goal of this project is to use simulator to profile the executions of different benchmark programs, to analyze the performance of different computer systems based on such benchmark programs, and to examine the roles of the compiler.

# SimpleScalar Overview

A tool that reproduces the behavior of a computing device

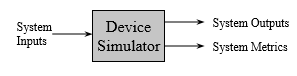


Figure : SimplScalar overview [1]

Why uses a simulator?

* leverage faster, more flexible S/W development cycle
* permits more design space exploration
* facilitates validation before H/W becomes available
* level of abstraction can be throttled to design task
* possible to increase/improve system instrumentation

## Tool set

The SimpleScalar tool set is a system software infrastructure used to build modeling applications for program performance analysis, detailed microarchitectural modeling, and hardware-software co-verification. Using the SimpleScalar tools, users can build modeling applications that simulate real programs running on a range of modern processors and systems. The tool set includes sample simulators ranging from a fast-functional simulator to a detailed, dynamically scheduled processor model that supports non-blocking caches, speculative execution, and state-of-the-art branch prediction. The SimpleScalar tools are used widely for research and instruction, for example, in 2000 more than one third of all papers published in top computer architecture conferences used the SimpleScalar tools to evaluate their designs. In addition to simulators, the SimpleScalar tool set includes performance visualization tools, statistical analysis resources, and debug and verification infrastructure [2].

SimpleScalar LLC is unique in the EDA community. We distribute our software under an open source model, trusting that our users will license the software that they use. The tool set is distributed with all source code, making it possible for users extend SimpleScalar, and to adapt existing models to their own ideas.

## Instruction set

SimpleScalar simulators can emulate the Alpha, PISA, ARM, and x86 instruction sets. The tool set includes a machine definition infrastructure that permits most architectural details to be separated from simulator implementations. All of the simulators distributed with the current release of SimpleScalar can run programs from any of the above listed instruction sets. Complex instruction set emulation (e.g., x86) can be implemented with or without microcode, making the SimpleScalar tools particularly useful for modeling CISC instruction sets.

The PISA instruction set (a.k.a. the portable instruction set architecture) is a simple MIPS-like instruction set maintained primarily for instructional use. A GNU GCC-based cross-compiler and pre-built libraries are also available for this target. The PISA target is particularly useful for computer engineering instruction as the tools can be built on a wide range of host platforms, including Linux/x86, Win2000, SPARC Solaris, and others.

## Platform

SimpleScalar builds on most 32-bit and 64-bit flavors of UNIX and Windows NT-based operating systems. The internal software architecture of the tool set includes a host interface module, permitting fast porting to other host platforms. The host interface module permits cross-endian emulation; thus it is possible to use emulate a target on a host platform with a different endian, e.g., running Alpha ISA emulation on a SPARC Solaris host platform. Most SimpleScalar users and developers (including SimpleScalar LLC) use SimpleScalar on Linux/x86.

# Experiments

For the five case, just need to install the packages of the simple scalar and run these packages. We can observe the different in every block by running the commands.

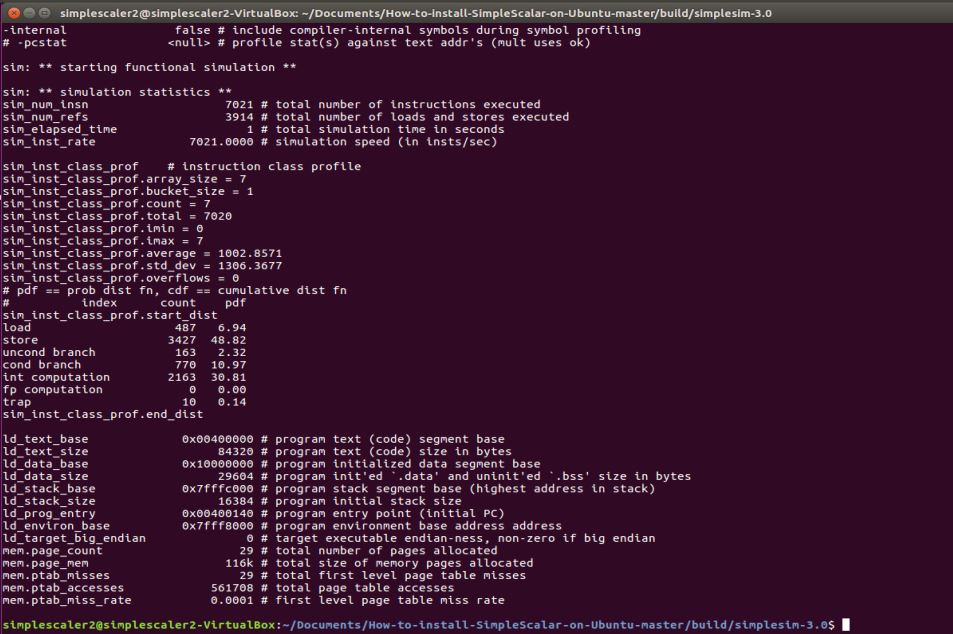


Figure :/usr/local/3rdparty/simplescalar/simplesim-3.0/sim-profile -iclass go.ss 4 6 go.in

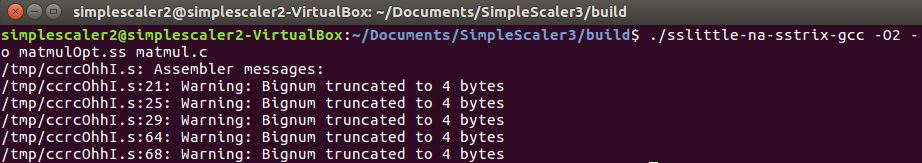


Figure :/usr/local/simplescalar/simplesim-3.0/sim-profile –iclass applu.ss < applu.in

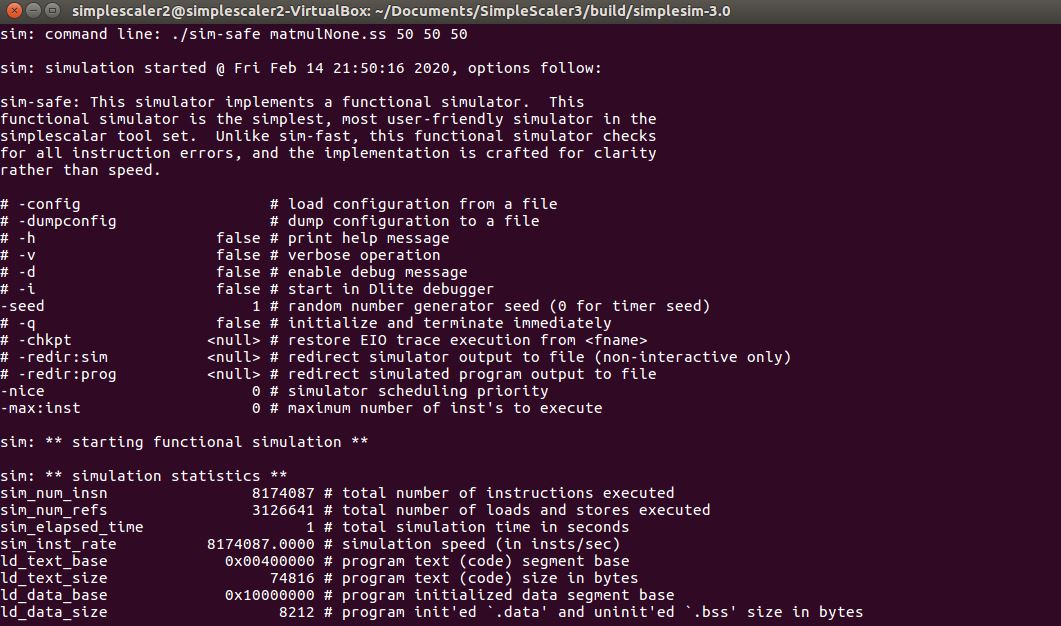


Figure : /usr/local/simplescalar/simplesim-3.0/sim-profile –iclass apsi.ss

The whole information of every command shows in details. We can check and verify the details after execution the commands.

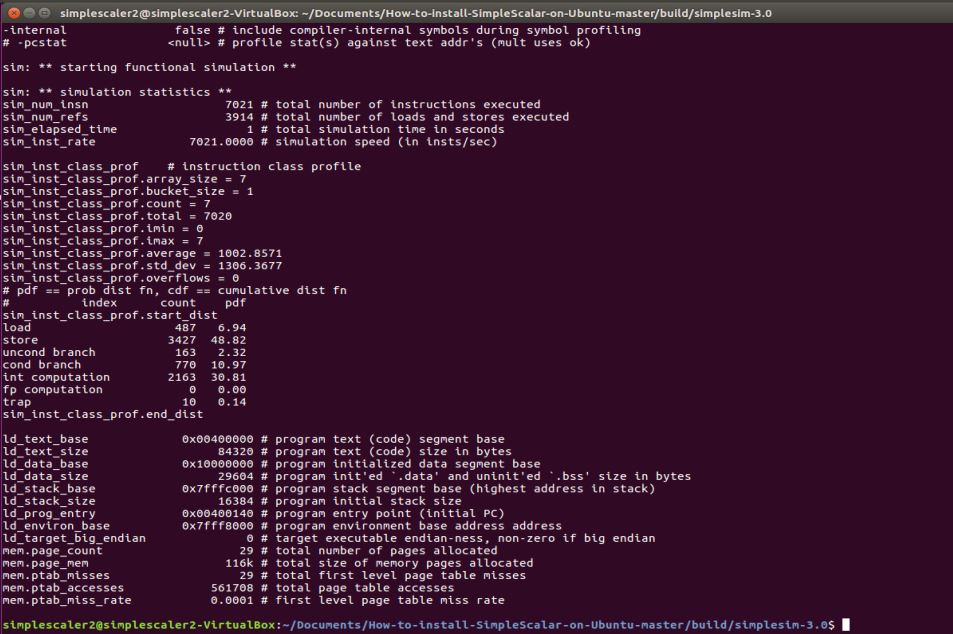


Figure : /usr/local/simplescalar/simplesim-3.0/sim-profile –iclass cc1.ss –O2 cc1.in –o test.s

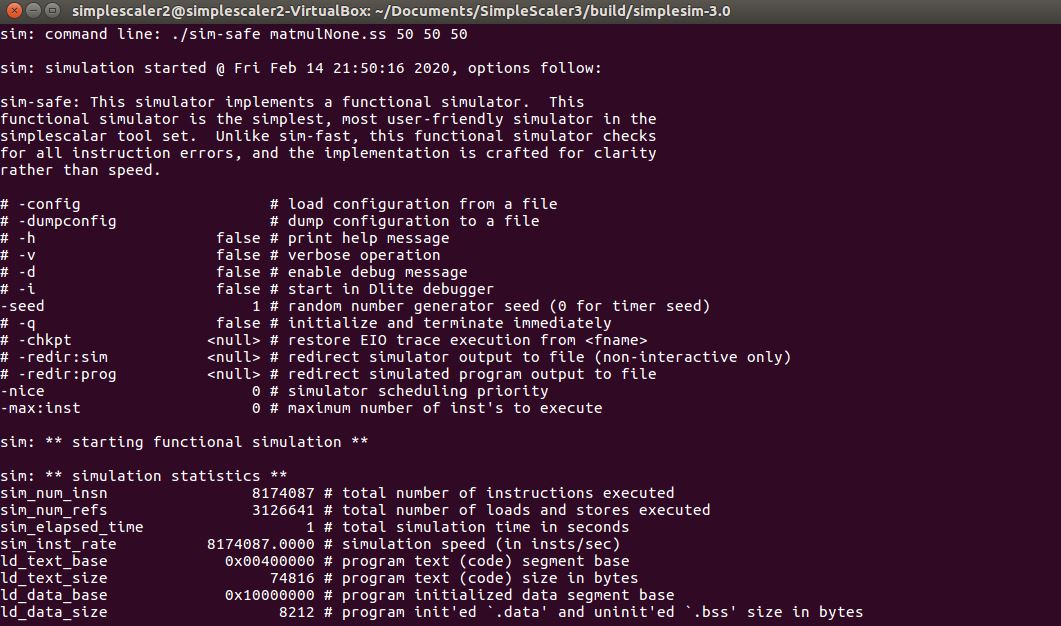


Figure :/usr/local/simplescalar/simplesim-3.0/sim-profile –iclass compress95.ss < compress95.in

## Observation

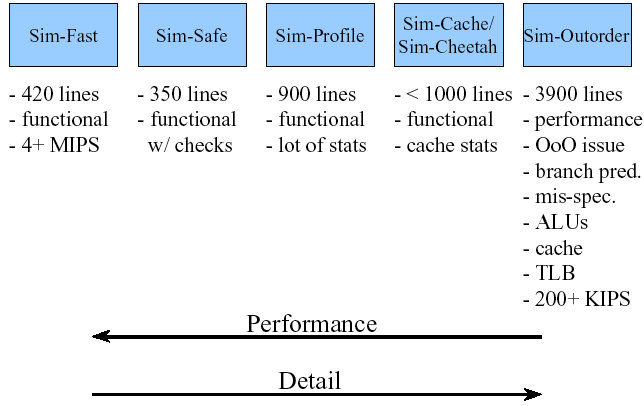


Figure : Performance of the sample scalar [3]

# Comparison of performance



Figure : given data

## First case

In the first we take the system 1 as reference model. In the both machine we have the same cycle time and have different CPI. First, we can see clock cycle is same in both machine and now only matter the CPI for performance. If we assume the first machine as reference then second system is faster at load and store time. It means data reload and store as compare to reference machine is better than in second system.

## Second case

In the second we take the system 2 as reference model. In the both machine we have the same cycle time and have different CPI. First, we can see clock cycle is same in both machine and now only matter the CPI for performance. If we assume the second machine as reference then first system is slower at load and store time. It means data reload and store as compare to reference machine is not better than in first system.

# CPI Account

We are also check the CPI and assume the best case first. We now for the integer CPI take 1 instruction and for the multiplication we get the multiplication N2 times and addition N times. it means total instruction are N3. It depends the number of row and column on every case for the CPI value. Let’s take 100-100-100 parameter then it means 1000000 CPI is required.

# Conclusion

In the lesson we learn about the architecture of the sample scalar and working of the sample scalar. First case check working and install the packages and then run the required commands and observe the result on each case. In last we also check the CPI for each case. CPI is matter for the performance. In the multiplication of the matrix CPI is give the complex case and take long CPI.

# References

|  |  |
| --- | --- |
| [1] | S. LLC. [Online]. Available: https://courses.cs.washington.edu/courses/cse471/06sp/hw/simpleScalar3.0guide.pdf. [Accessed 16 02 2020]. |
| [2] | [Online]. Available: http://www.simplescalar.com/overview.html. |
| [3] | "SimpleScalar CS401," [Online]. Available: http://www2.cs.siu.edu/~rahimi/cs401/slides/SimpleScalarppt.ppt. [Accessed 16 02 2020]. |