ARMSim: Simulating Advanced RISC Machine Architecture

Shuqiang Zhang
Department of Computer Science
Columbia University
New York, NY
sz184@columbia.edu

Abstract

This paper discusses the design and implementation of the ARMSim, a simulator implemented in the Java and C programming languages for the Advanced RISC Machine (ARM) processor. The intended users of this tool are those individuals interested in learning computer architecture, particularly those with an interest in the Advanced RISC Machine processor family.

ARMSim facilitates the learning of computer architecture by offering a hands on approach to those who have no access to the actual hardware. The core of the simulator is implemented in C with and models a fetch-decode-execute paradigm; a Java GUI is included for portability. The details of the ARM architecture, including registers, instruction set and implementation will be discussed in later sections.

A binary tree traversal algorithm is introduced to make the decoding part of the simulator more efficient. This increases the overall performance of the simulator.

1. Introduction:

This paper describes how to simulate an ARM processor using the C programming language. In the course of this discussion, the reader is introduced to the details of the ARM processor architecture and discover how the hardware specifications are simulated in software using execution-driven simulation. Execution driven simulation is also know as instruction-level simulation, register-cycle simulation or cycle-by-cycle simulation [Sykes 3]. Instruction level simulation consists of fetch, decode and execution phases [Barbieri 4].

ARM processors were first designed and manufactured by Acorn Computer Group in the mid 1980's [funkysh 1]. Due to its high performance and power efficiency, ARM processors can be found on wide range of electronic devices, such as Sony Playstation, Nintendo Game Boy Advance and Compaq iPAQs. The 32-bit microprocessor was designed using a RISC architecture with data processing operations occurring in registers instead of memory. The processor has 16 visible 32 bit registers and a reduced instruction set that is 32-bits wide. The details on the registers and instructions can be obtained from [ARM 2].

2. Related Works:

This section discusses different types of simulators available today and their different approaches in design and implementation. Most simulation tools can be classified as user level simulators: these simulate the execution of a process and emulate any system calls made on the target computer using the operating system of the host computer [Clarke 5]. ARMSim is an example of this type of simulator; it executes ARM instructions on a host Pentium x86 processor, but it will deviate from the conventional decoding method in an attempt to improve execution time. KScalar Simulator [Moure 6], PPS suite [Gunther 7], CPU Sim3.1 [Skrien 8] and OAMulator [Menczer 9] are simulators best suited for educational purposes. They show the basic ideas of computer organization with relatively few details and complexity. They are specifically designed for students who have little or no background in computer architecture and who need a simple introduction

[Moure 6]. ARMSim also belongs in this category because it provides a concise and straightforward introduction to the ARM architecture. On the other extreme of the spectrum is the SPARC V9 Complete Machine Simulator, one of the few well-know complete machine simulators developed to date. These simulate the target computer from the boot stage, including all codes executed by the PROM, the OS that is loaded by the PROM, and any processes subsequently created [Clarke 5]. Another approach to processor simulation can be seen in the Simx86 simulator. The Simx86 abandons the traditional simulator implementation approach of pre-decoding instructions and cross compilation. Instead, Simx86 favors an object oriented approach to improve extensibility of the simulator at the cost of increased execution time. The Simx86 provides a straightforward way to build a simulator for a processor by allowing each component of the processor to be represented directly in the simulator by an object. The simulator can easily be extended by adding new classes of instructions without the daunting task of constructing a new simulator [Shealy 10]. ARMSim will retain the traditional approach for building simulators in favor of execution time.

References:

[1] funkysh, "Into my ARMs" www.phrack.org/show.php?p=58&a=10

[2] ARM Architectural Reference Manual – Issue D, 2000 Advanced RISC Machines LTD

[3] D. A. Sykes, B.A. Malloy, The Design of an Efficient Simulator for the Pentium Pro Processor, In *Proceedings of the 1996 Winter Simulation Conference,* pp. 840-847, 1996.

[4] I. Barbieri, M. Bariani, M. Raggio, A VLIW Architecture Simulator Innovative Approach for HW-SW Co-Design, *2000 IEEE international Conference on Multimedia and Expo*, Vol. 3. pp1375-1378, 2000.

[5] B. Clarke, A. Czezowski, P. Strazdins, Implementation Aspects of a SPARC V9 Complete Machine Simulator, In *Conferences in Research in Information Technology, Vol. 4.* Australian Computer Society, pp. 23-32, 2001.

[6] J. C. Moure, D. I. Rexachs, E. Luque, The KScalar Simulator, *ACM Journal of Educational Resources in Computing*, Vol. 2, No. 1, pp. 73-116 March, 2002.

[7] B. K. Gunther, Facilitating Learning in Advanced Computer Architecture through Appropriate Simulation, *ACSC 23rd Australasian Computer Science Conference, 2000*. pp. 104-112, 1999.

[8] D. Skrien, CPU Sim3.1: A Tool for Simulating Computer Architectures for Computer Organization Classes, *ACM Journal of Educational Resources in Computing*, Vol. 1, No. 4 , pp. 46-59, December, 2001.

[9] F. Menczer, A. M. Segre, OAMulator: A Teaching Resource to Introduce Computer Architecture Concepts, *Journal of Educational Resources in Computing,* Vol. 1, No. 4, pp18-30, December, 2001.

[10] A. R. Shealy, B. A. Malloy, Simx86: An Extensible Simulator for the Intel 80x86 Processor Family, In *Proceedings of the 30th Annual Simulation Symposium*, pp. 157-166, 1997.