DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Bc. Jakub Šilhavý

Master's Thesis

Computer Science and Engineering
Software Engineering (SWI)
2023/2024

Thesis supervisor:
Ing. Martin Úbl

FACULTY OF APPLIED SCIENCES
UNIVERSITY OF WEST BOHEMIA

ARMv6 Processor Emulator for Raspberry Pi Environment

Abstract

This paper examines the potential of emulating Raspberry Pi Zero, a selected example of one of the most widely adopted architectures for embedded systems — the ARM architecture. The initial chapters delve into a general introduction to the ARM architecture, highlighting its profound significance evidenced by billions of electronic that leverage it. Transitioning to the second part, the thesis addresses the benefits of utilizing an ARM emulator, delineating overall requirements, and reviewing existing methodologies. The second part centers on the development of a custom Raspberry Pi Zero emulator whose functionality is systematically tested using a set of examples pertinent to operating system development. The thesis concludes with an objective evaluation of the emulator's performance, identifying its key benefits, and suggesting areas for further enhancements.

Introduction

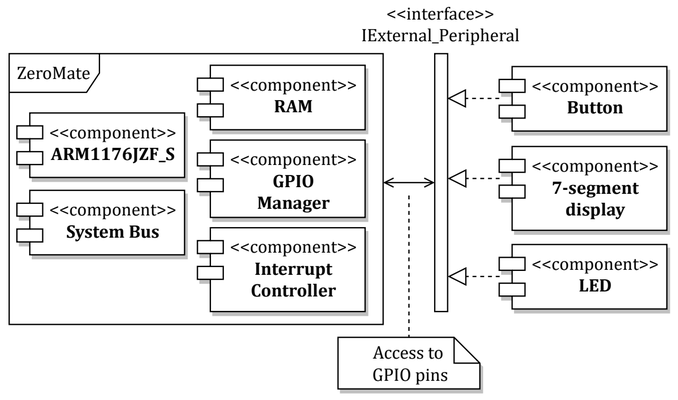
ARM stands out as one of the most widely embraced computer architectures, finding application across a diverse range of domains. Its uses span from low-power solutions and affordable microcontrollers to real-time applications and safety-critical systems, including medical devices, automotive technology, and aviation. Moreover, ARM plays a significant role in personal computers and the cellular phone industry, currently powering over 99% of the world's smartphones. Emulating such an extensively adopted architecture can assist in illustrating concepts of computer organization and operating systems. Additionally, it offers advantages in a software development process, particularly when immediate access to a development board may not be feasible. Furthermore, it provides a layer of abstraction, allowing developers to experiment with potentially risky code without the concern of damaging real hardware.

Existing Solutions

Several available solutions, such as QEMU, CPUlator, and ARMSim#, can be employed for emulating ARM architecture. However, as the thesis concludes, their emulation capabilities may be limited, as most of them lack some of the more advanced system-related features, such as the ability to switch CPU modes or implement paging, which is essential in operating system development. Consequently, the thesis aimed to develop an extensible Raspberry Pi Zero emulator capable of emulating KIV-RTOS, an educational real-time operating system developed at the University of West Bohemia.

Proposed Solution – ZeroMate

ZeroMate was developed as a Raspberry Pi Zero emulator to address the challenges associated with existing solutions. It was meticulously designed in a modular fashion, enabling users to seamlessly connect custom third-party peripherals such as displays, sensors, and actuators. These peripherals can be developed independently of the core system, providing users with flexibility and customization options.



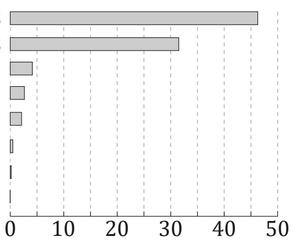
ZeroMate's public interface for external peripherals

It was designed with the aim of giving users a comprehensive visual overview of the entire system, revealing real-time information on the current contents of both RAM and CPU registers. Furthermore, it offers insights into the current configurations of various BCM2835 peripherals, encompassing GPIO, interrupt controller, ARM timer, MiniUART, BSC (I^2C), and more. Moreover, it boasts support for fundamental debugging features, thereby simplifying the debugging processes for cross-compiled applications. Additionally, the emulator incorporates coprocessor support, empowering users to harness features like floating-point numbers or paging.

Testing & Achieved Results

The core of the emulator underwent rigorous testing through an extensive suite of unit tests, addressing its fundamental yet crucial functionalities that other parts of the system heavily depend on. Functional testing integrated different components of the emulator, working alongside to execute specific tasks, such as blinking an LED using a timer interrupt or scheduling processes running in userspace. The final phase of testing, system testing, was conducted by students enrolled in the KIV/OS to assess its overall usability in practice. Using KIV-RTOS, the emulator average speed of emulation was calculated to be 4.84 mega instructions per second.

Instruction type	Impact on performace [avg. execution time [s] * absolute count]
Single data transfer	46.293
Data processing	31.553
Branch	4.244
Extend	2.760
Block data transfer	2.232
Software interrupt	0.595
Branch and exchange	0.306
Store return state	0.110



Top 8 most performance-affecting instructions