



# **ARMv6 Processor Emulator for Raspberry Pi Environment Emulation**

Jakub Šilhavý<sup>1</sup>

## **1** Introduction

ARM stands out as one of the most widely embraced computer architectures, finding application across a diverse range of domains. Its utility extends from low-power solutions and affordable microcontrollers to real-time applications and safety-critical systems, encompassing fields such as medical devices, the automotive industry, and aviation. Furthermore, ARM plays a significant role in personal computers and the cell phone industry, currently powering more than 99% of the world's smartphones (Arm Editorial Team (2023)).

Emulating such an extensively adopted architecture can assist in illustrating concepts of computer organization and operating systems. Additionally, it offers advantages in the field of software development, 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.

### 2 Existing Solutions

Several available solutions, such as QEMU, CPUlator, and ARMSim#, can be employed for emulating the 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 indispensable 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 (Martin Úbl (2021)).

## **3** Proposed Solution - ZeroMate

ZeroMate was developed to address the challenges associated with the 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. It was created with the aim of giving users a comprehensive visual overview of the entire system, allowing them to examine the current contents of various core components, such the CPU registers and RAM.

<sup>&</sup>lt;sup>1</sup> student of the master degree program Applied Sciences, field of study Software Engineering, e-mail: silhavyj@students.zcu.cz

Furthermore, it offers insights into the current configurations of various BCM2835 peripherals, including the GPIO pins, interrupt controller, ARM timer, MiniUART, BSC (I2C), and more. Moreover, it boasts support for fundamental debugging features, thereby simplifying the debugging process for cross-compiled applications. Additionally, the emulator incorporates coprocessor support, empowering users to harness features like floating-point numbers or virtual addressing.

ZeroMate - Rpi Zero emulator		- 🗆 X
▼ File Control	➡ Source Code Disassembly	▼ RAM GPIO IC ARM timer Monitor ×
▼ File Control ■ Step ● Run ● Stop ₩ Reset State: stopped ▼ CPU Registers CPIS CPU Mode: System USR/SYS FIQ SVC ABT IRQ UDM HEX U32 [S32] Register Value R0 00000001 R1 00000000	<ul> <li>Dource Lobe Usassenity</li> <li>Ox0000CFC 0x2FFFFF b #0xCfd4</li> <li>Ox0000CFC 0x2FFFFF b #0xCfd4</li> <li>Ox0000D004 0xE5A02001 mov r2, #1</li> <li>Ox0000D04 0xE51B0008 ldr r0, [fp, #-8]</li> <li>Ox0000D008 0xE51B0008 ldr r0, [fp, #-8]</li> <li>Ox0000D010 0xE5A03000 mov r3, #0</li> <li>Ox0000D010 0xE5A03000 ldr r3, [fp, #-0xc]</li> <li>Ox0000D010 0xE51B300C ldr r3, [fp, #-0xc]</li> <li>Ox0000D010 0xE51B300C ldr r3, [fp, #-0xc]</li> <li>Ox0000D010 0xE5A03000 cmp r3, #0x400</li> <li>Ox0000D010 0xE5A03000 mov1t r3, #1</li> </ul>	<pre>v bad 0-10 ic Arch timer monitor</pre>
R2         00000001           R3         0000028F           R4         0000028F           R5         0000CE10           R6         0000CE20           R7         00000840           R8         0000CE24           R9         0000E24           R0         0000E24           R0         00000E24           R0         00000E24           R10         00000E0           R11         00004F40           R12         200000F           R13         (LR)         00004F34           R14         (SP)         00004F34           R15         00004F34         R15           R14         (SP)         00004F34           R15         (CC)         00004F34	0x80000024         0x3303000         movpe r3, #0           0x00000024         0x56EF3073         uxth r3, r3           0x00000025         0x5253000         emp r3, #0           0x00000026         0x5253000         emp r3, #0           0x00000030         0x0AFFFFE         beq #0xcfbc           0x00000030         0x52833001         add r3, r3, #1           0x00000030         0xE2833001         add r3, r3, #1           0x00000040         0xEAFFFFF4         b #0xd015           0x00000040         0xEAFFFF74         b #0xd015           0x00000040         0xEAFFFF74         b #0xd015           0x00000044         0x00000E112         andeq 1r, r0, r0, 1s1 r           0x00000048         0x00000E128         andeq 1r, r0, r8, 1sr #	13:         My favourite sport is           13:         ARM wrestling           16:         ARM wrestling           17:         18:           18:         Process 2 tile descriptor (Matje 1           21:         1829430600pening file: DEV:0010/18           LED         10:           0:         21:           18:         6 Color           19:         12:           20:         18:           21:         18:000000           18:         Shift register: 00000000           19:         Latch pin: 27
Flags in CSPR N Z C V A I F 0 0 1 0 0 0 1	Options Clear Copy Learning Core copy (Constraint Constraint) (debug] (interrupt.controller copy:264) Basic IRQ ARM_Timer (warning] (core.cop):267) IRQ exception (debug] (interrupt.controller.cop):264) Basic IRQ ARM_Timer (warning] (core.cop):267) IRQ exception (debug] (interrupt.controller.cop):264) Basic IRQ ARM_Timer (warning] (core.cop):267) IRQ exception (interligence).com/Constraint) (interligenc	r has been signalized r has been signalized

Figure 1: ZeroMate's graphical user interface

## **4** Conclusion

The core of the emulator underwent rigorous testing through an extensive set of unit tests, addressing its fundamental yet crucial functionalities that other parts of the system heavily depend on. Unit tests cover approximately 78% of the core's functionality. 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 class to assess its overall usability in practice. Using KIV-RTOS, ZeroMate's average speed of emulation was measured to be 4.84 mega instructions per second <sup>1</sup>.

# References

- Arm Editorial Team. (2023). *The Official History of Arm*. Available from: https://newsroom.arm.com/arm-official-history [Accessed 25th September 2023].
- Martin Úbl (2021). KIV-RTOS An educational operating system for bare-metal Raspberry Pi Zero W (BCM2835-based board). Available from: https://github.com/ MartinUbl/KIV-RTOS [Accessed 27th September 2023].

<sup>&</sup>lt;sup>1</sup>The experiment was conducted on the Lenovo ThinkPad P50 laptop, running the Windows 10 operating system. The laptop is equipped with an Intel(R) Core(TM) i7-6820HQ CPU running at 2.70GHz and 16GB of RAM. In terms of C++, the std::chrono::high\_resolution\_clock class was employed for measuring execution speed in nanoseconds.