

Emulátor ARMv6 procesoru pro emulaci prostředí Raspberry Pi

Bc. Jakub Šilhavý

vedoucí práce: Ing. Martin Úbl

Katedra informatiky a výpočetní techniky
Fakulta aplikovaných věd
Západočeská univerzita v Plzni

17. 06. 2024

Cíl práce

- návrh a implementace emulátoru platformy **Raspberry Pi Zero**
 - emulace instrukcí procesoru ARM1176JZF-S (ARMv6)
 - emulace základních periferií µC BCM2835 (GPIO, Mini_UART, BSC, ...)

① vzdělávací účely

- vizualizace principů OS
- embedded vývoj

② testování a ladění SW

③ prototypování HW

- připojení externích periferií
- návrh vlastního systému



Figure 1: Raspberry Pi Zero

- použití **KIV-RTOS** pro ověření správnosti výsledného emulátoru

ARM procesory a jejich aplikace



Figure 2: ARM aplikace

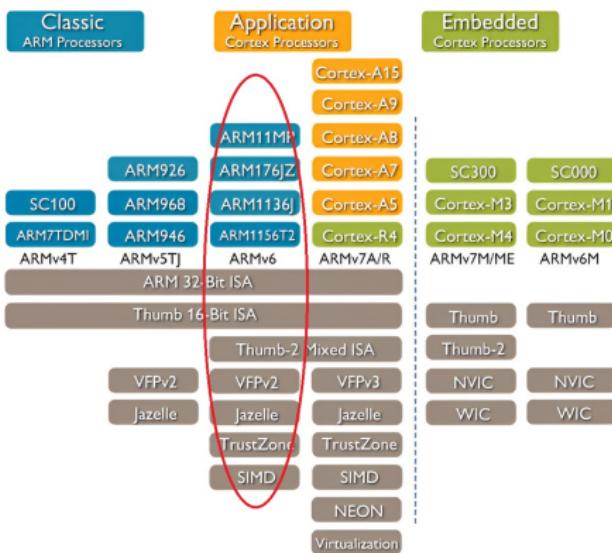


Figure 3: ARM CPU roadmap

- vhodné pro seznámení se s programováním v JSA
 - vizualizace, ladění programu
- floating-point instrukce
- platformová nezávislost

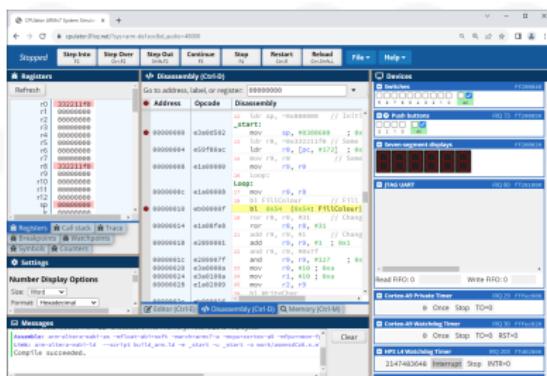


Figure 4: ▶ CPUlator

Omezení

- emulace pouze CPU (ne celého SoC)
- minimální podpora pokročilých systémových operací
 - ⇒ nevhodné pro testování principů OS
- limitovaná podpora připojení externích periferií



Figure 5: ▶ QEMU

- podpora vícero architektur
 - x86, MIPS, ARM, ...
- připojení externího debuggeru (GNU)
- plná podpora systémových operací

Omezení

- emulace pouze CPU (ne celého SoC)
- limitovaná podpora připojení externích periferií
- `_start` symbol očekáván na adrese 0x00010000 (ARM)
 - ⇒ nekompatibilita s *first-stage BL Rpi0* (0x00008000)
- problémy se `systimer`

ARM1176JZF-S

- ARMv6 instrukce
- přepínání režimů CPU
- ALU, MAC a MMU (+ TLB)
- vyjímky a přerušení
- podpora ko-procesorů
 - CP15, CP10
- systémová sběrnice

BCM2835

- RAM
- interrupt controller
- ARM timer
- TRNG
- GPIO
- BSC_1 (I^2C)
- AUX (Mini_UART)
- debug monitor*

- **cílem bylo emulovat nejčastěji používané periferie**
- dekompoziční návrh architektury
 - ⇒ snadné rozšíření o další periferie

- jednotné rozhraní pro externí periferie
- načtené za běhu jako sdílené knihovny
 - možnost načtení více instancí (např. led.dll)
- nezávislé na toolchainu jádra emulátoru
 - `extern "C" __declspec(dllexport)`

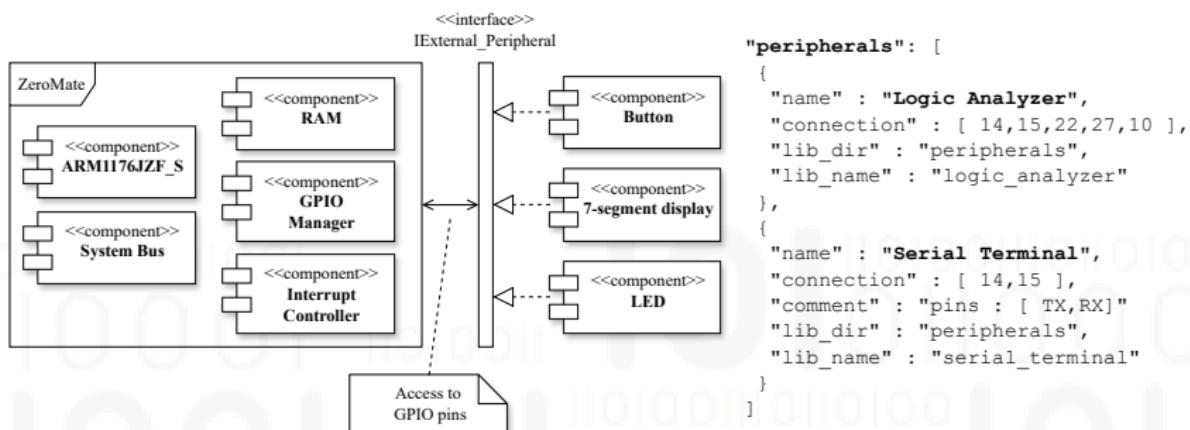
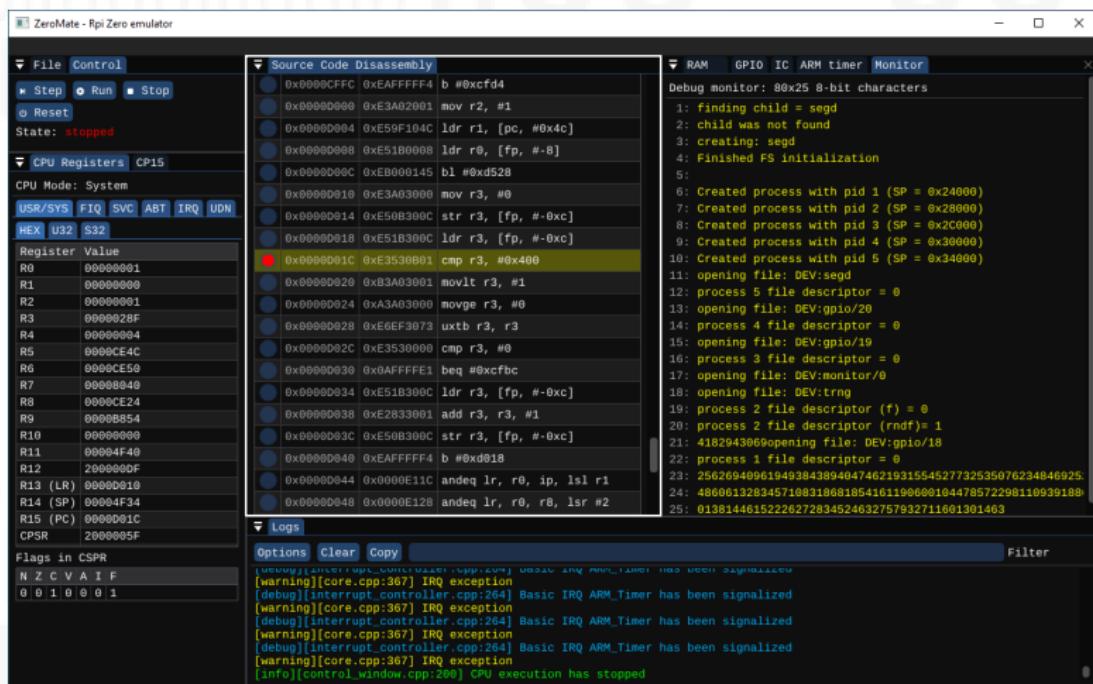
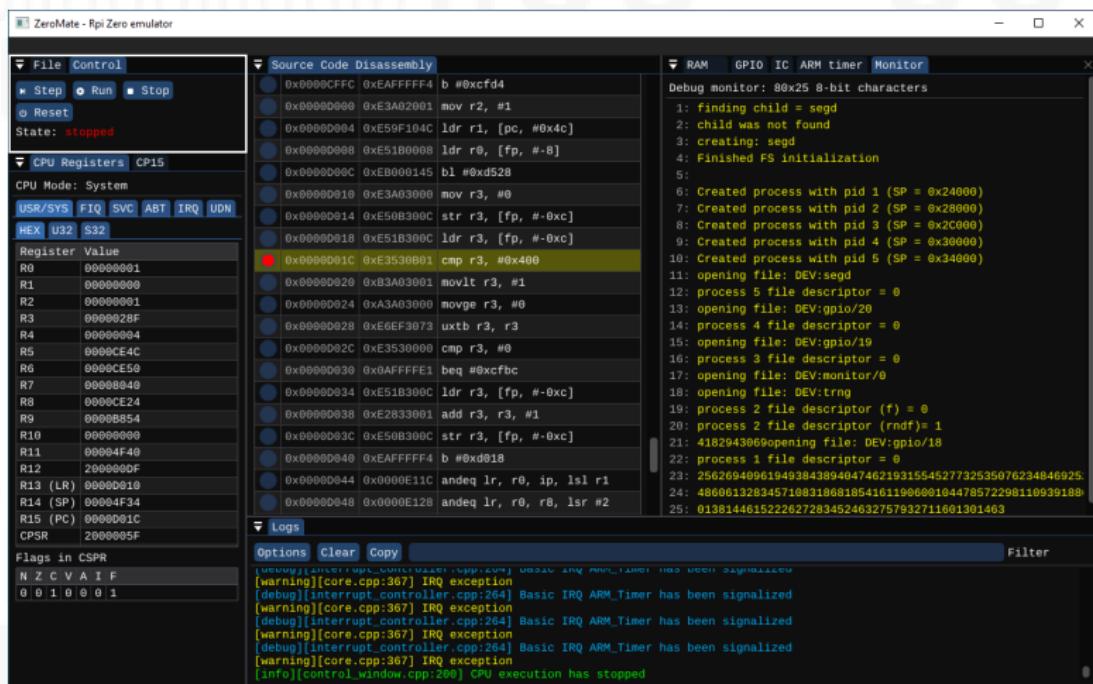


Figure 6: Rozhraní externích periferií

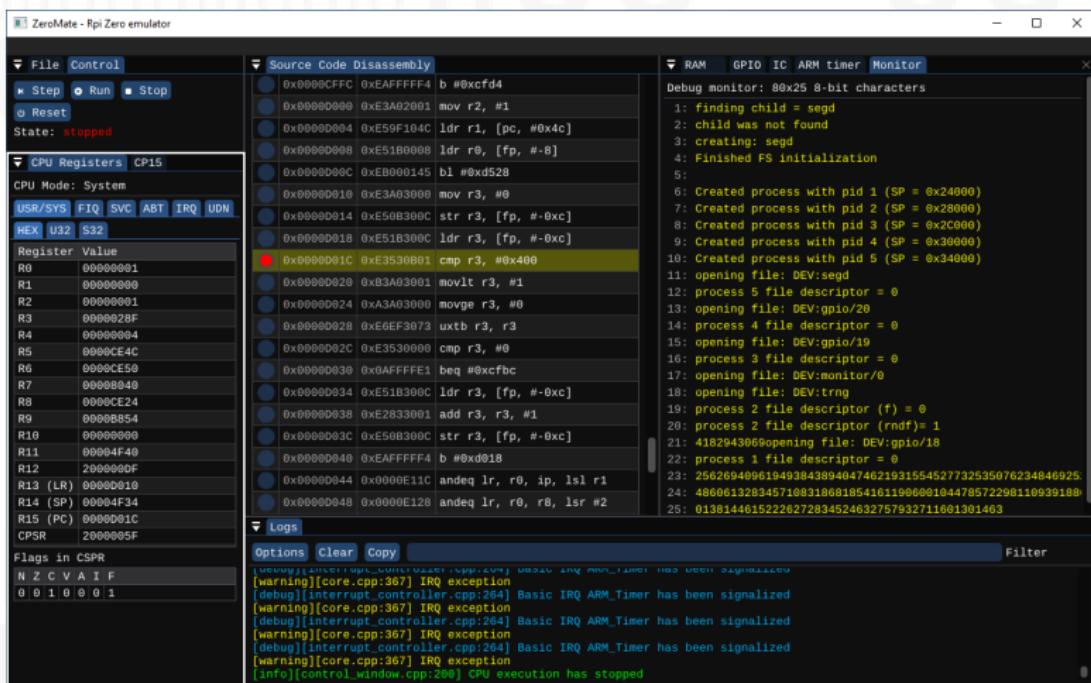
(1) disassembly vstupního ELF souboru



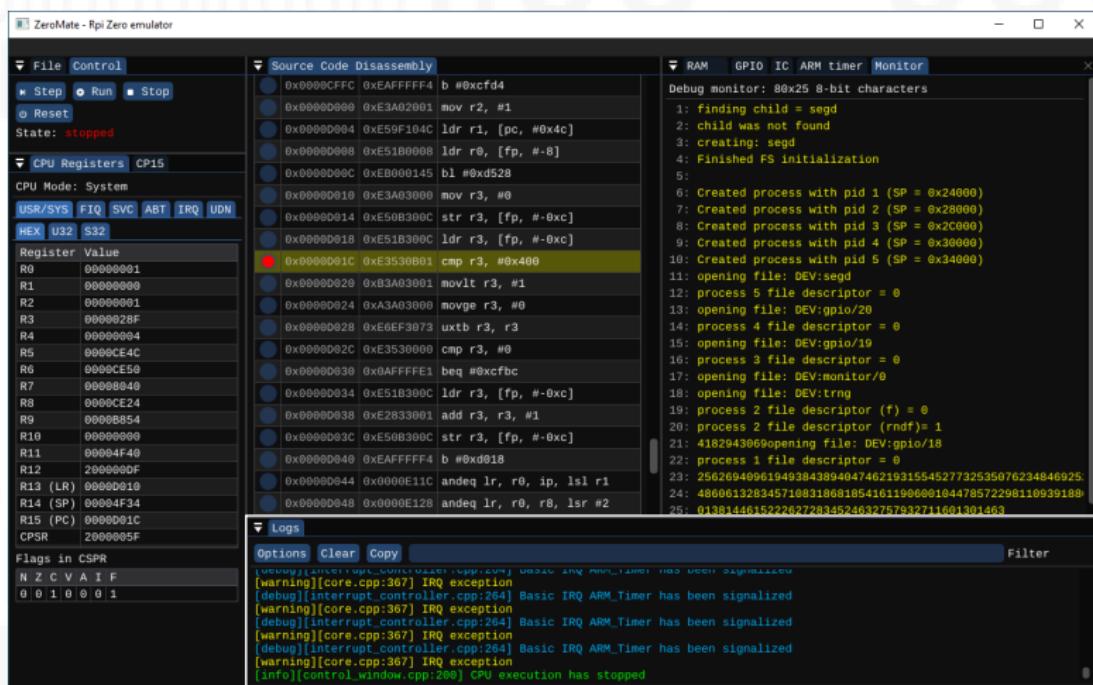
(2) řízení emulace (start, stop, step a reset)



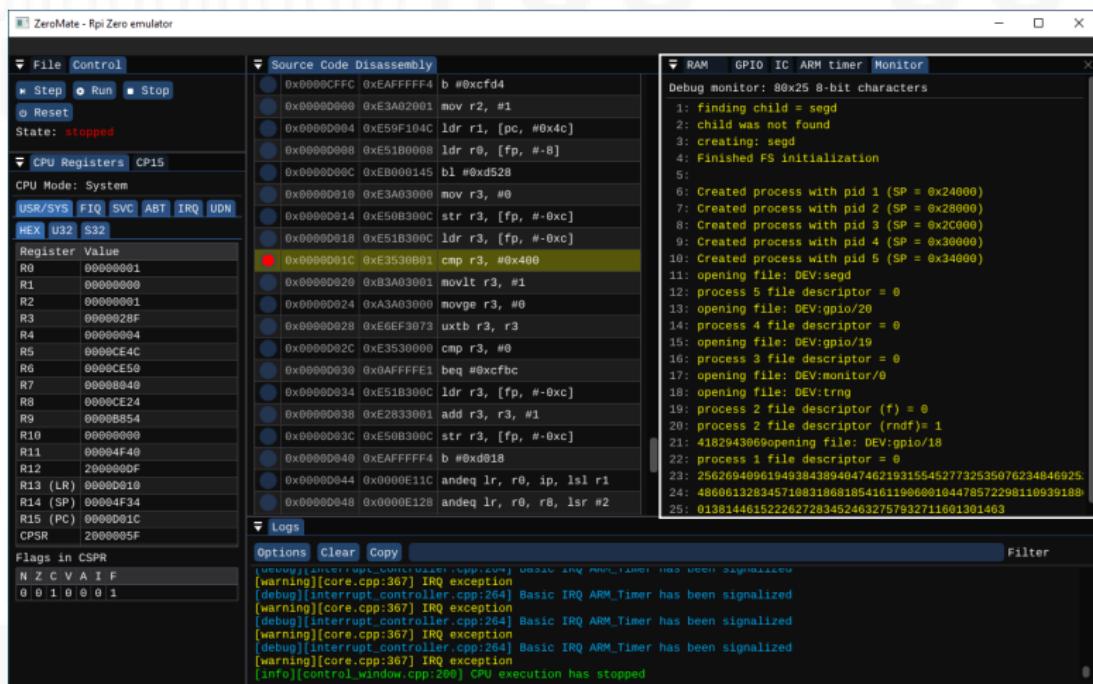
(3) registry CPU a systémový koprocessor CP15



(4) logování událostí (přerušení, vyjímky, ...)



(5) BCM2835 periferie (registry + interpretace hodnot)



ZeroMate - externí periferie

(1) tlačítko (další periferie viz ▶ KIV-DPP-01)

The screenshot shows the ZeroMate - Rpi Zero emulator interface. The main window displays assembly code, register values, and memory dump sections. A floating window titled "SOS btn" shows "GPIO pin: 16 Press". The monitor section lists various system events and processes.

Source Code Disassembly:

```

0x00000000 0xEFFFFF4 b #0xfcfd4
0x00000000 0xE3A02001 mov r2, #1
0x00000004 0xE59F104C ldr r1, [pc, #0x4c]
0x00000008 0xE51B0008 ldr r0, [fp, #-8]
0x0000000C 0xE0000145 bl #0xd528
0x00000010 0xE3A03000 mov r3, #0
0x00000014 0xE50B300C str r3, [fp, #-0xc]
0x00000018 0xE51B300C ldr r3, [fp, #-0xc]
0x0000001C 0xE3530001 cmp r3, #0x400
0x00000020 0xE3A03001 movlt r3, #1
0x00000024 0xA3A03000 movge r3, #0
0x00000028 0xE6EF3073 uxtb r3, r3
0x0000002C 0xE3530000 cmp r3, #0
0x00000030 0x0AFFFF11 beq #0xcfbc
0x00000034 0xE51B300C ldr r3, [fp, #-0xc]
0x00000038 0xE2833001 add r3, r3, #1
0x0000003C 0xE50B300C str r3, [fp, #-0xc]
0x00000040 0xEAFFFFF4 b #0xd018
0x00000044 0x0000E11C andeq lr, r0, ip, lsl r1
0x00000048 0x0000E128 andeq lr, r0, r0, lsr #2

```

Registers:

Register	Value
R0	00000001
R1	00000000
R2	00000001
R3	0000028F
R4	00000004
R5	0000CE4C
R6	0000CE50
R7	00000840
R8	0000CE24
R9	0000B854
R10	00000000
R11	00004F40
R12	2000000DF
R13 (LR)	0000D010
R14 (SP)	00004F34
R15 (PC)	0000001C
CPSR	2000005F

Flags in CPSR:

N	Z	C	V	A	I	F
0	1	0	0	1	1	1

Logs:

```

[warning][core.cpp:367] IRQ exception
[debug][interrupt_controller.cpp:264] Basic IRQ ARM_Timer has been signaled
[warning][core.cpp:367] IRQ exception
[debug][interrupt_controller.cpp:264] Basic IRQ ARM_Timer has been signaled
[warning][core.cpp:367] IRQ exception
[debug][interrupt_controller.cpp:264] Basic IRQ ARM_Timer has been signaled
[warning][core.cpp:367] IRQ exception
[debug][interrupt_controller.cpp:264] Basic IRQ ARM_Timer has been signaled
[warning][core.cpp:367] IRQ exception
[info][control_window.cpp:200] CPU execution has stopped

```

(2) SSD1036 OLED displej řízený přes I²C

The screenshot shows the ZeroMate - Rpi Zero emulator interface. On the left, there's a sidebar with 'File' and 'Control' buttons (Step, Run, Stop, Reset), CPU Registers (CP15), CPU Mode (System), and various memory and register sections. The 'Register Value' section shows values for R0-R15 and CPSR. The 'Flags in CSR' section shows N, Z, C, V, A, I, F flags. The main area has tabs for 'Source Code Disassembly' (containing assembly code for the SSD1036 driver), 'RAM', 'GPIO', 'IC', 'ARM timer', and 'Monitor'. The 'Monitor' tab is active, showing a 'SOS btn' button with 'GPIO pin: 16' and a 'Press' button. Below it is a box for the 'SSD1036 OLED' display, which shows the text 'My favourite sport is ARM wrestling'. The bottom part of the interface is a 'Logs' window with a 'Filter' bar, displaying multiple entries from the kernel log.

```

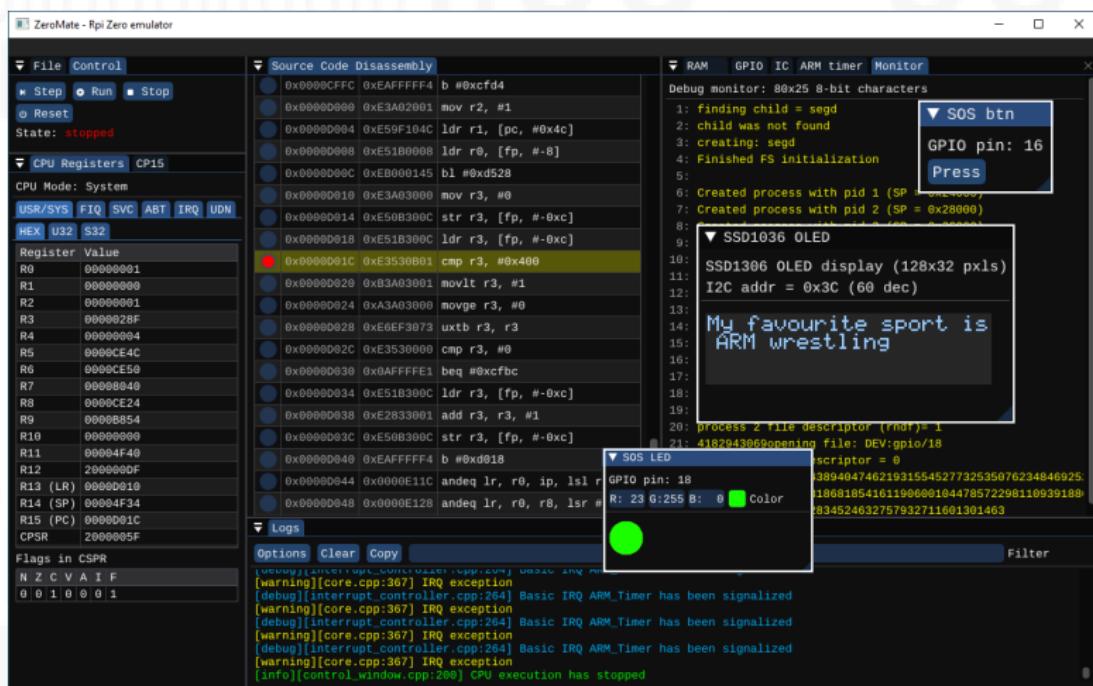
Source Code Disassembly
0x00000CFFC 0xEFFFFFF4 b #0xfcfd4
0x00000000 0xE3A02001 mov r2, #1
0x00000004 0xE59F104C ldr r1, [pc, #0x4c]
0x00000008 0xE51B0008 ldr r0, [fp, #-8]
0x0000000C 0xE0000145 bl #0xd528
0x00000010 0xE3A03000 mov r3, #0
0x00000014 0xE50B300C str r3, [fp, #-0xc]
0x00000018 0xE51B300C ldr r3, [fp, #-0xc]
0x0000001C 0xE3530B01 cmp r3, #0x400
0x00000020 0xB3A03001 movlt r3, #1
0x00000024 0xA3A03000 movge r3, #0
0x00000028 0xE6EF3073 uxtb r3, r3
0x0000002C 0xE3530000 cmp r3, #0
0x00000030 0x0AFFFFE1 beq #0xfcfc
0x00000034 0xE51B300C ldr r3, [fp, #-0xc]
0x00000038 0xE2833000 add r3, r3, #1
0x0000003C 0xE50B300C str r3, [fp, #-0xc]
0x00000040 0xEFFFFFF4 b #0xd018
0x00000044 0x00000E11C andeq lr, r0, ip, lsl r1
0x00000048 0x00000E128 andeq lr, r0, r0, lsr #2

Logs
[warning][core.cpp:367] IRQ exception
[debug][interrupt_controller.cpp:264] Basic IRQ ARM_Timer has been signaled
[warning][core.cpp:367] IRQ exception
[debug][interrupt_controller.cpp:264] Basic IRQ ARM_Timer has been signaled
[warning][core.cpp:367] IRQ exception
[debug][interrupt_controller.cpp:264] Basic IRQ ARM_Timer has been signaled
[warning][core.cpp:367] IRQ exception
[debug][interrupt_controller.cpp:264] Basic IRQ ARM_Timer has been signaled
[warning][core.cpp:367] IRQ exception
[info][control_window.cpp:200] CPU execution has stopped

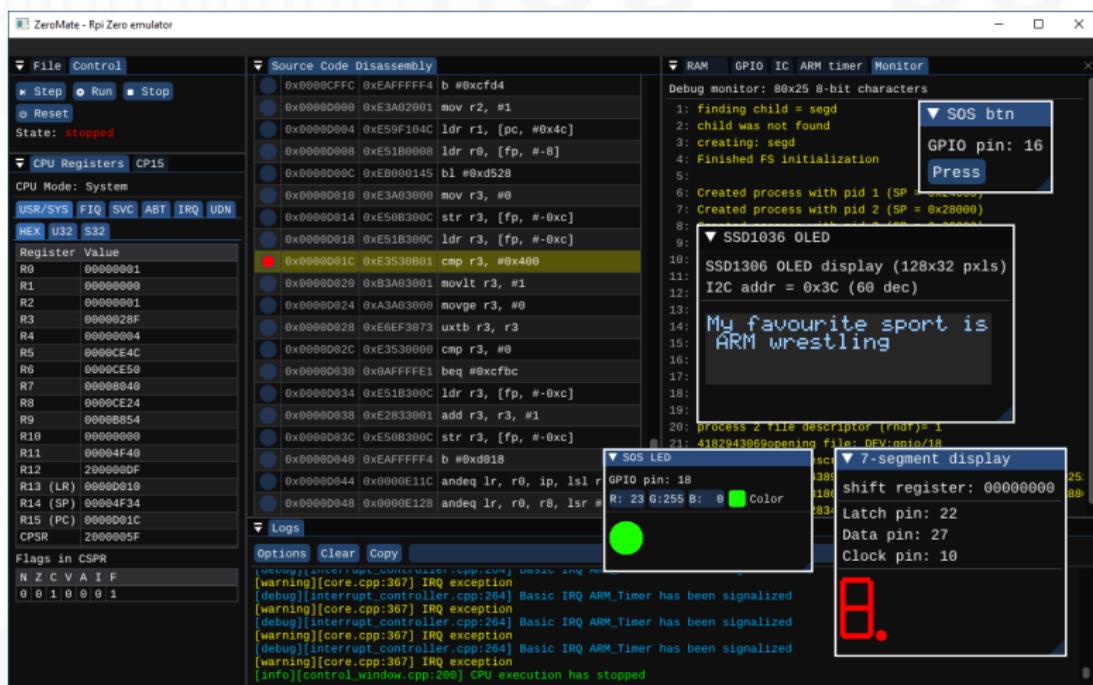
```

ZeroMate - externí periferie

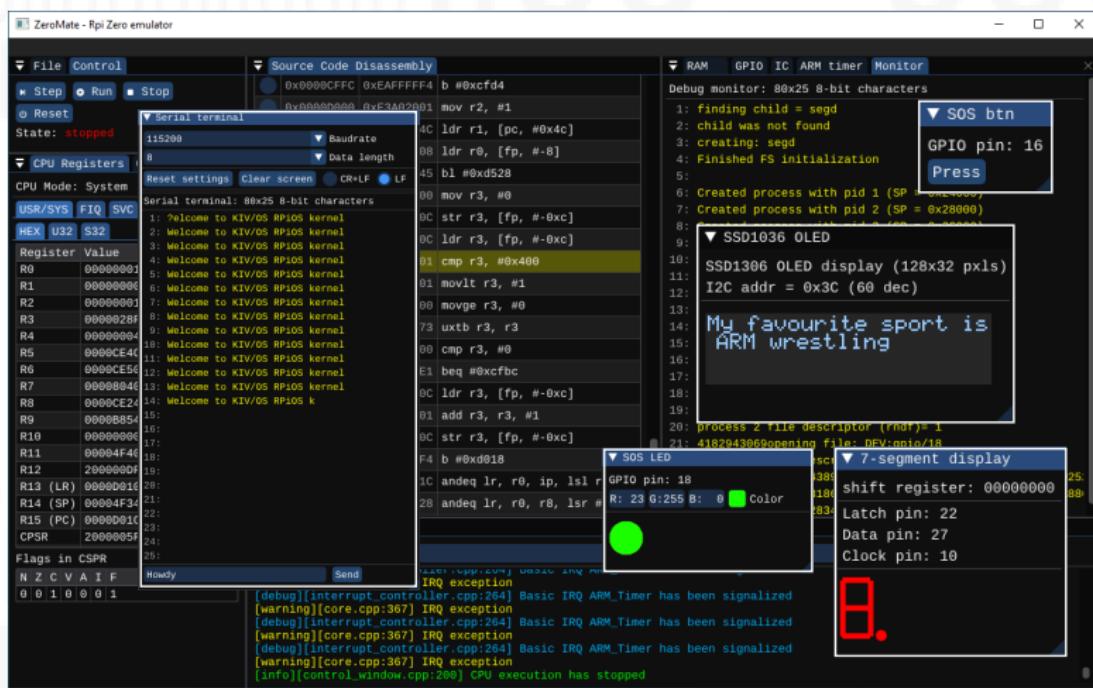
(3) RGB LED



(4) sedmisegmentový LED displej řízený posuvným registrem

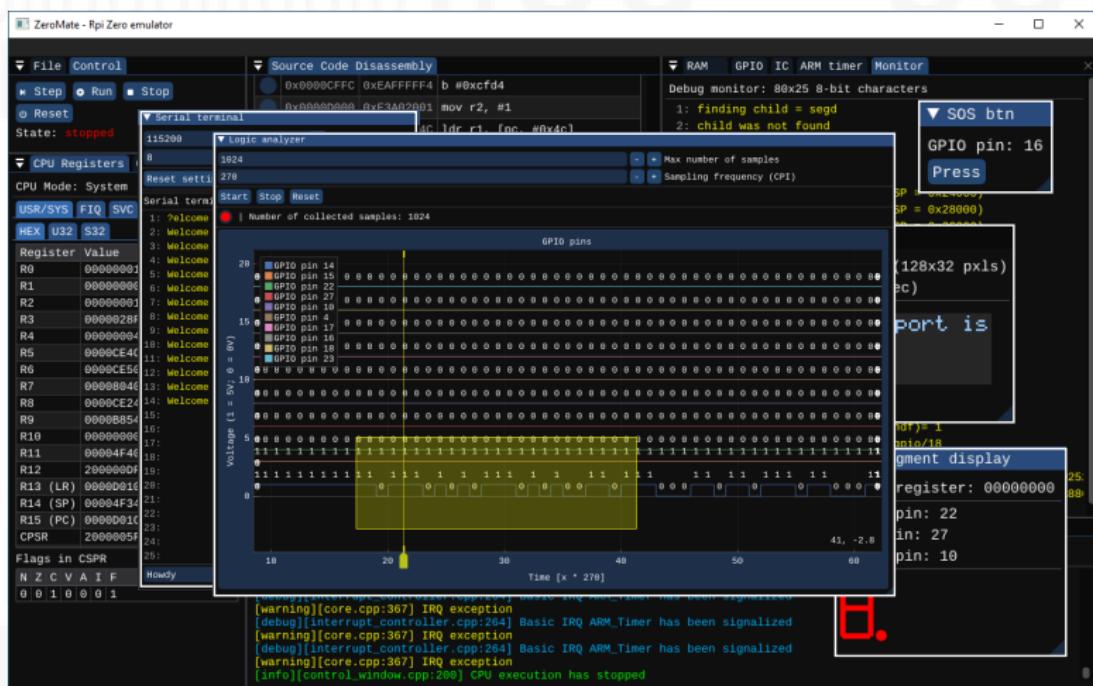


(5) sériový terminál (UART)



ZeroMate - externí periferie

(6) logický analyzátor



Vývoj a použité technologie



► <https://github.com/silhavyj/ZeroMate>



- knihovny třetích stran automaticky staženy a sestaveny (.gitmodules + CMake)

- [GoogleTest](#) [ImGui](#) [dylib](#) [ELFIO](#) [GLFW](#) [capstone](#)

1) Unit testy

- jádro emulátoru (exekuce ARMv6 instrukcí)
- GoogleTest, CI (regresní testy), **pokrytí $\approx 78\%$**

2) Funkční testy

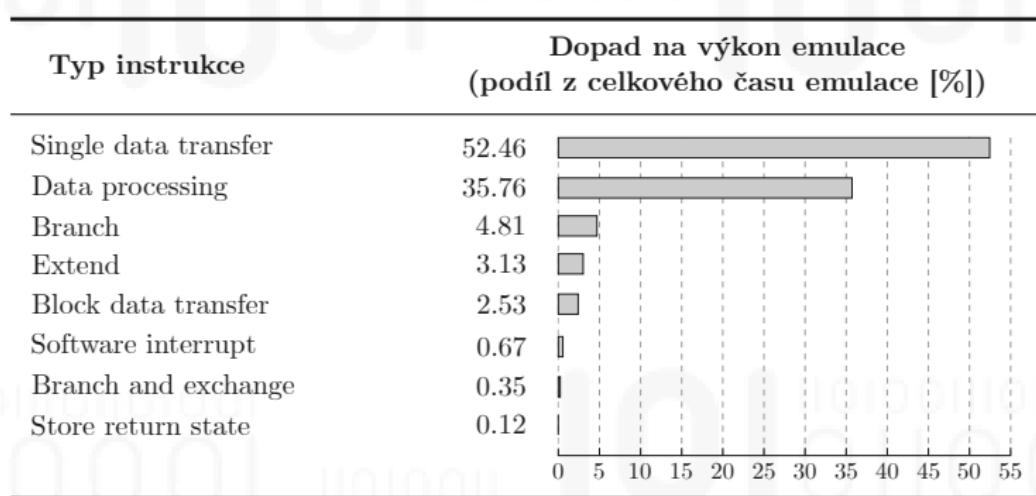
- BCM2835 periferie
- připravená sada příkladů (.ELF)
 - UART, FPU, GPIO, I²C, plánování procesů (EDF, RTL), ARM timer, ...
 - **součástí repozitáře jako examples**

3) Aplikační testy, GUI

- KIV/OS cvičení
- dotazník hodnocení celkové použitelnosti

Výkonnost emulace ARMv6 instrukcí

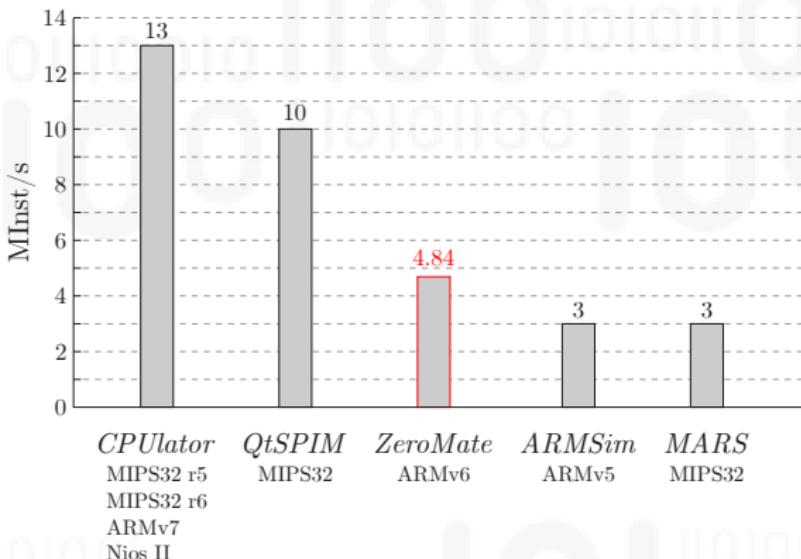
- **výkon emulátoru $\approx 4\text{-}5 \text{ Minst/s}$** ([► KIV-RTOS](#))
- analýza rychlosti emulace jednotlivých instrukcí
 - \Rightarrow potenciální optimalizace



1

¹ Měřeno na: *Lenovo ThinkPad P50 laptop; Windows 10; Intel(R) Core(TM) i7-6820HQ; 2.70GHz; 16GB*

Porovnání výkonnosti s ostatními emulátory



- pouze orientační (<https://cpulator.01xz.net/>, 2023)
- ZeroMate jako jediný podporuje MMU
 - využití s každým přístupem do paměti ($\approx 42\%$ instrukcí)

IPC

- např. pomocí socketů
- spuštění vícero instancí ZeroMate emulátoru
 - ⇒ emulace distribuovaných systémů & algoritmů

CLI mód

- integrace do CI/CD pipeline
- validace programů v rámci výuky
 - např. definováním očekáváné UART komunikace (I/O)

Další viz text ▶ diplomové práce ...

Návrhy na budoucí vylepšení

Děkuji Vám za pozornost

▶ <https://github.com/silhavyj/ZeroMate>



Otázka

Odpověď

Otázka

Odpověď