

# 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



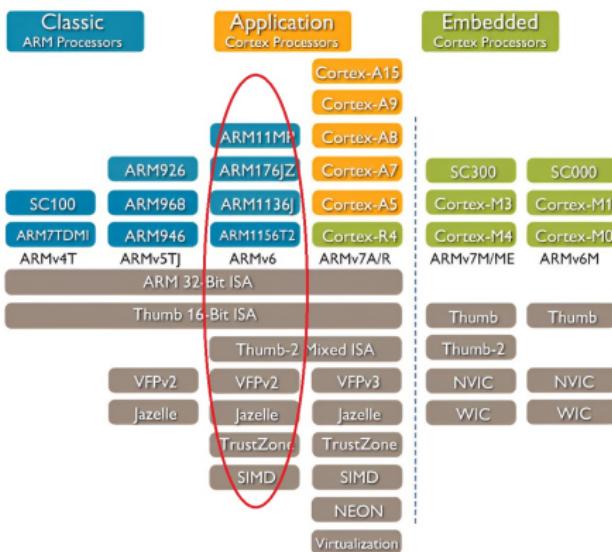
Obrázek 1: Raspberry Pi Zero

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

## ARM procesory a jejich aplikace

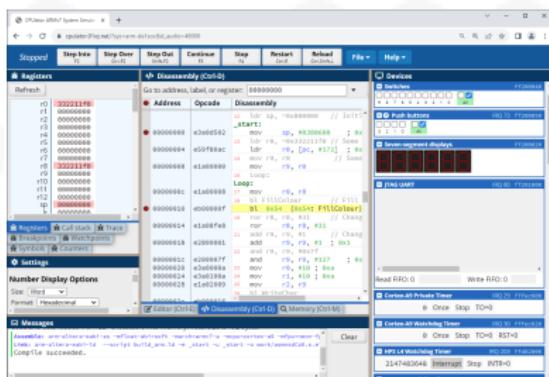


Obrázek 2: ARM aplikace



Obrázek 3: ARM CPU roadmap

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



Obrázek 4: ▶ CPUlator

## Omezení

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

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



Obrázek 5: ▶ QEMU

## Omezení

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

## ARM1176JZF-S

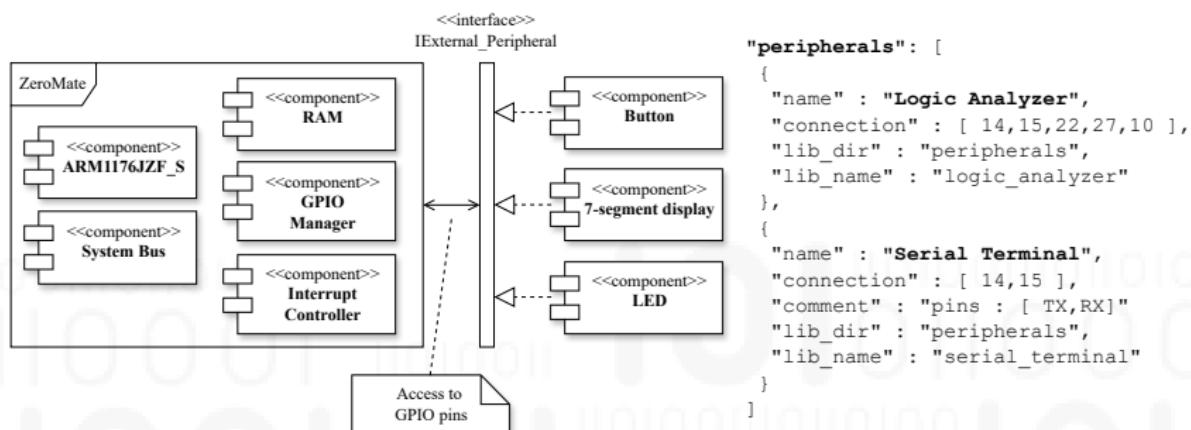
- ARMv6 instrukce
- přepínání režimů CPU
- ALU, MAC a MMU (+ TLB)
- vyjímky a přerušení
- podpora ko-procesorů
  - CP15, CP10
- systémová sběrnice

## BCM2835

- RAM
- interrupt controller
- ARM timer
- TRNG
- GPIO
- BSC\_1 ( $I^2C$ )
- AUX (Mini\_UART)
- debug monitor\*

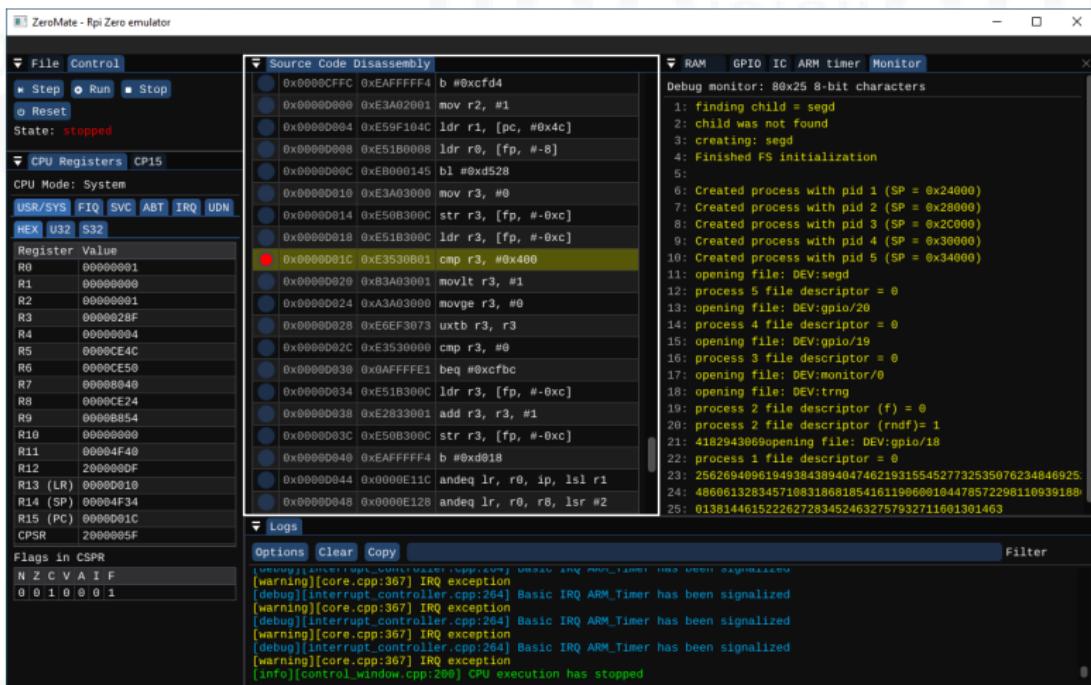
- **cílem bylo emulovat nejčastěji používané periferie**
- dekompoziční návrh architektury
  - ⇒ snadné rozšíření o další periferie

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



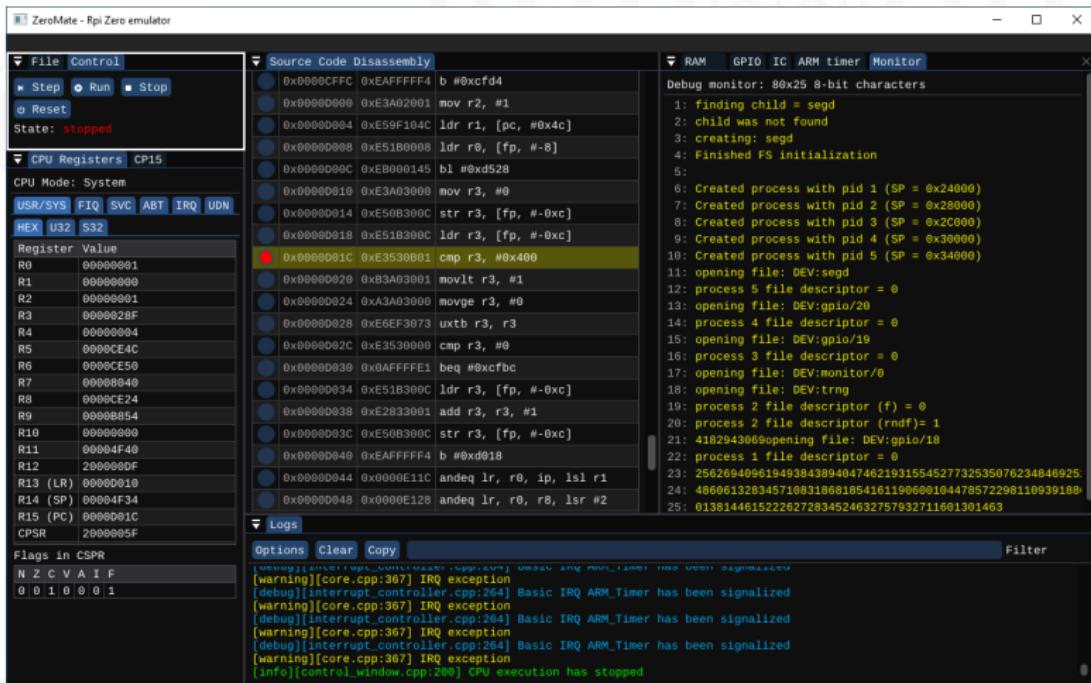
Obrázek 6: Rozhraní externích periferií

## ZeroMate - uživatelské rozhraní



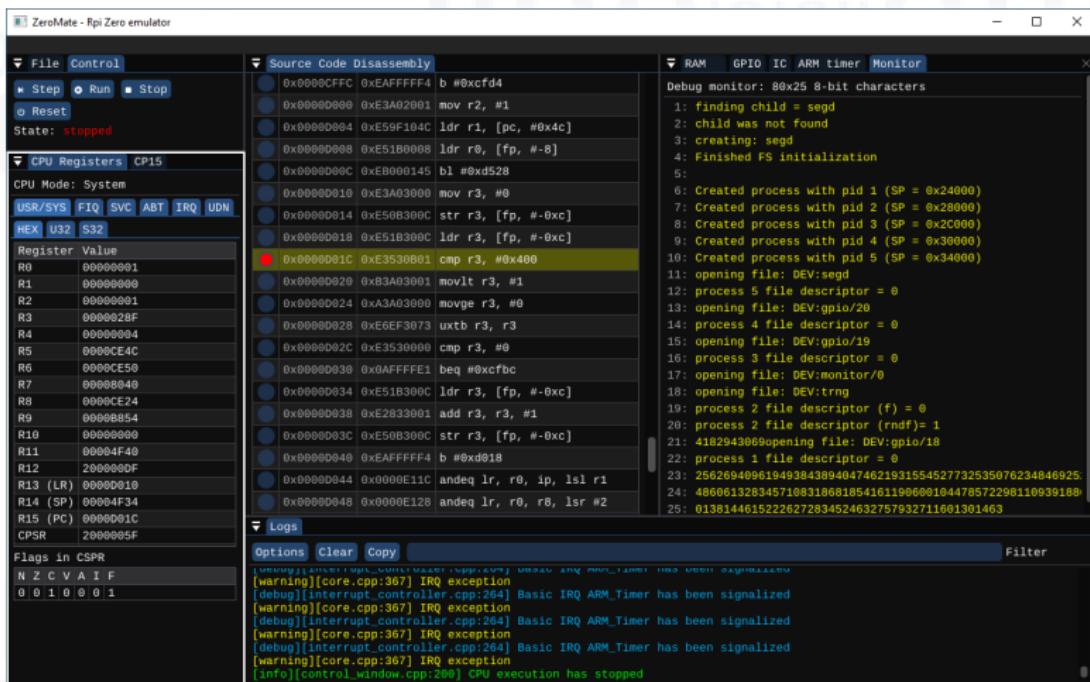
Obrázek 7: a) Disassembly vstupního ELF souboru

## ZeroMate - uživatelské rozhraní



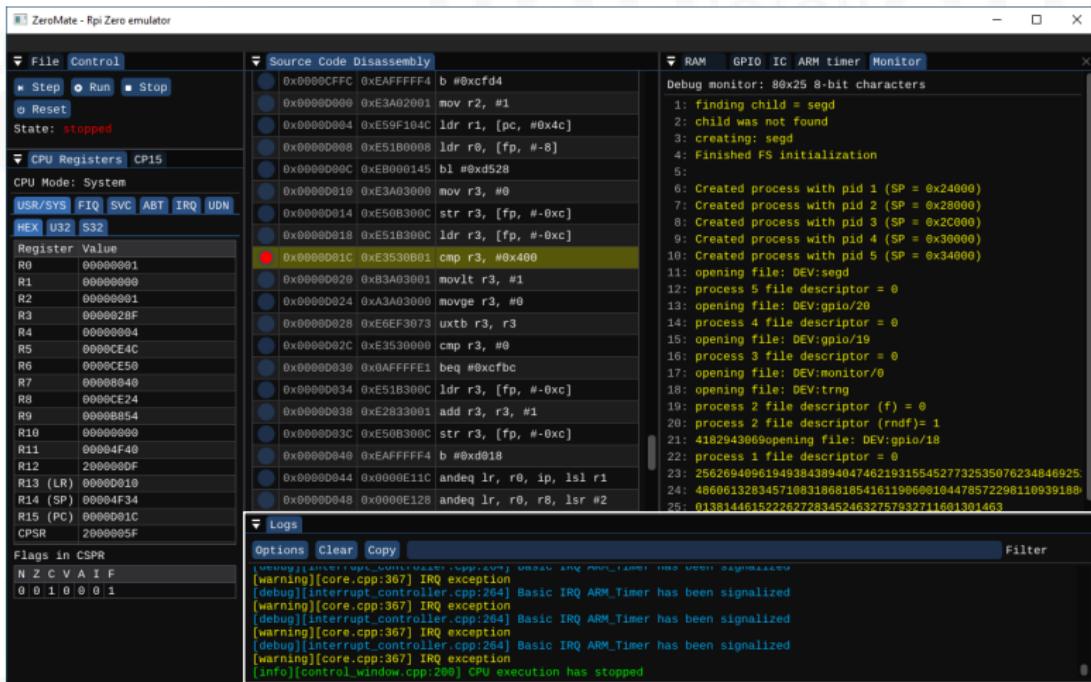
Obrázek 7: b) Řízení emulace (start, stop, step a reset)

## ZeroMate - uživatelské rozhraní



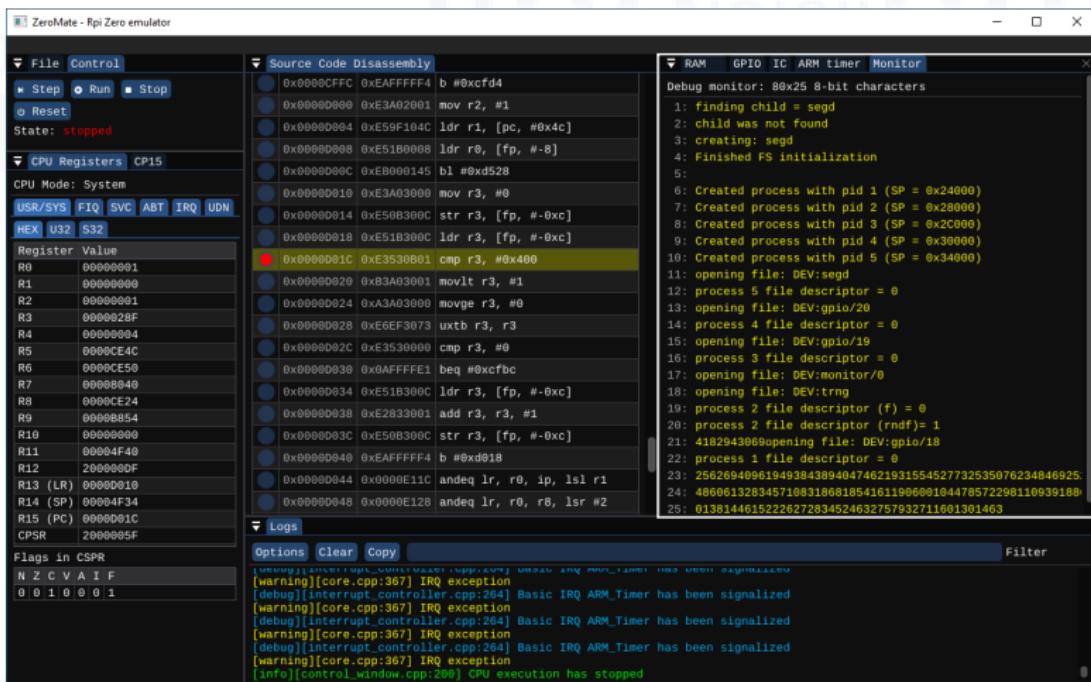
Obrázek 7: c) Registry CPU a systémový koprocesor CP15

## ZeroMate - uživatelské rozhraní



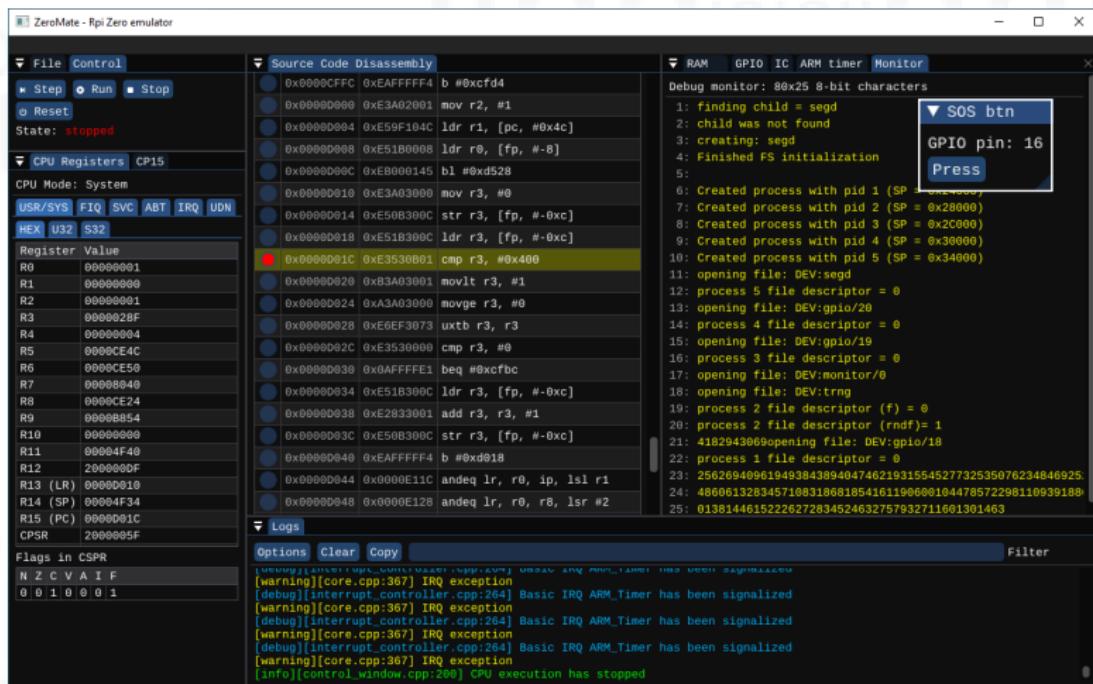
Obrázek 7: d) Logování událostí (přerušení, vyjímky, ...)

## ZeroMate - uživatelské rozhraní



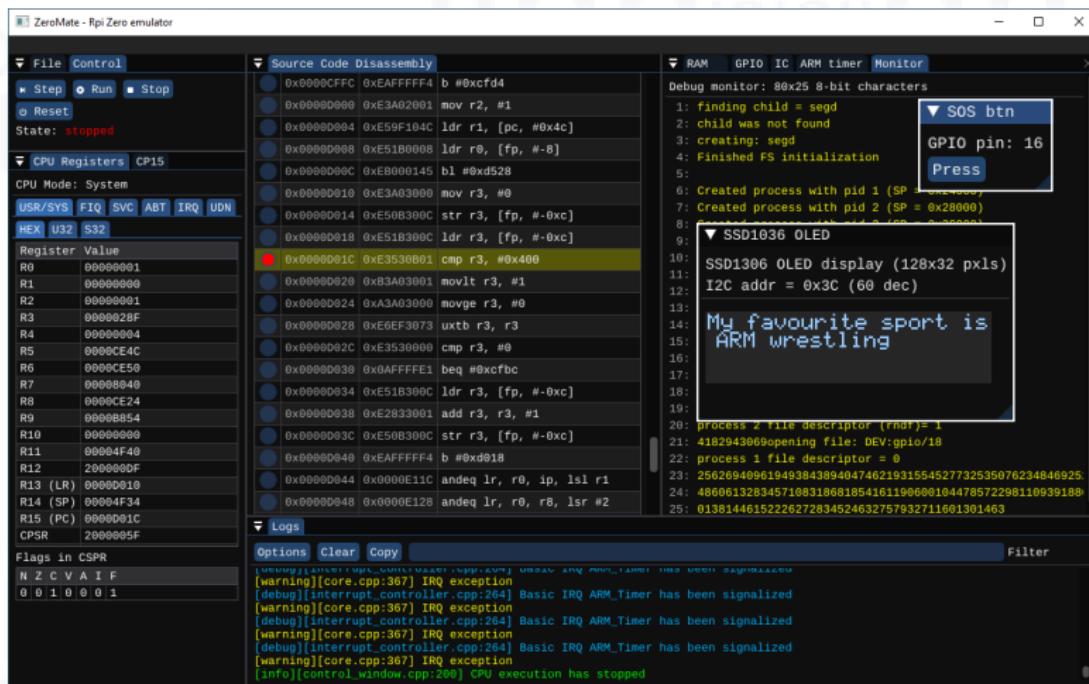
Obrázek 7: e) BCM2835 periferie (registry + interpretace hodnot)

## ZeroMate - externí periferie

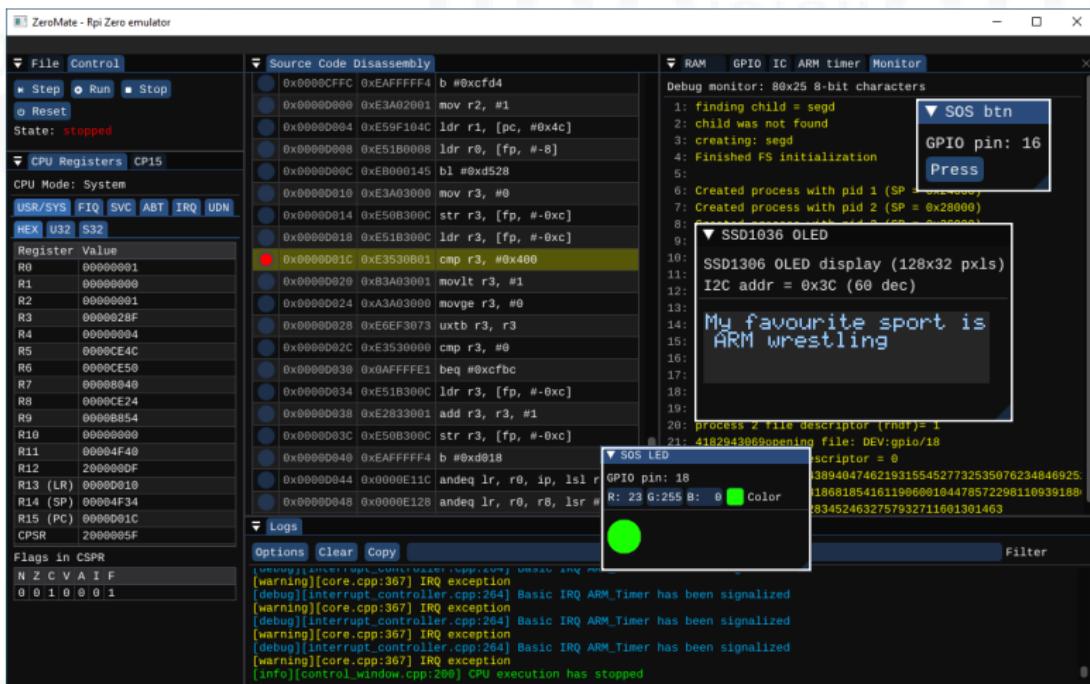


Obrázek 8: a) Tlačítko (další periferie viz ▶ KIV-DPP-01)

## ZeroMate - externí periferie

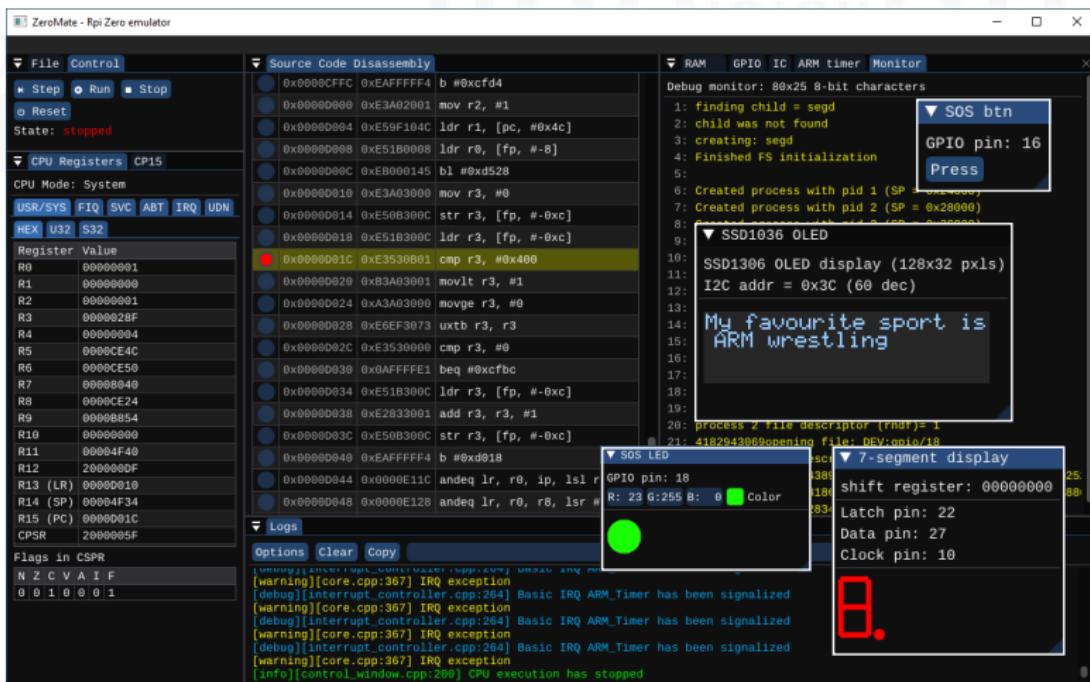
Obrázek 8: b) SSD1036 OLED displej řízený přes I<sup>2</sup>C

## ZeroMate - externí periferie



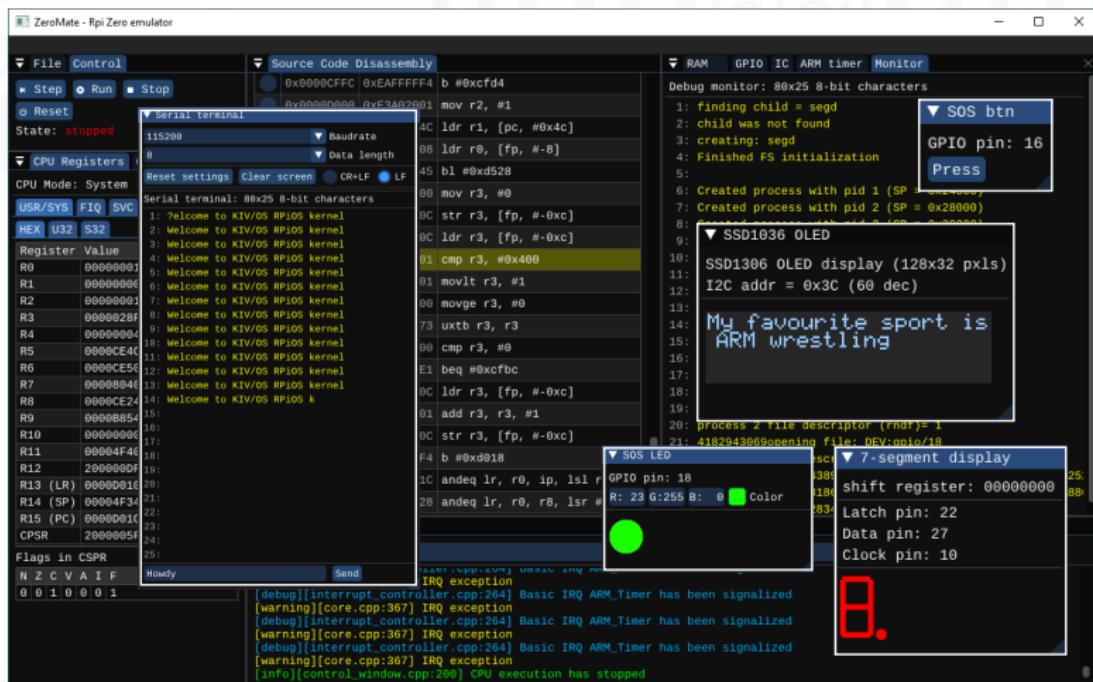
Obrázek 8: c) RGB LED

## ZeroMate - externí periferie



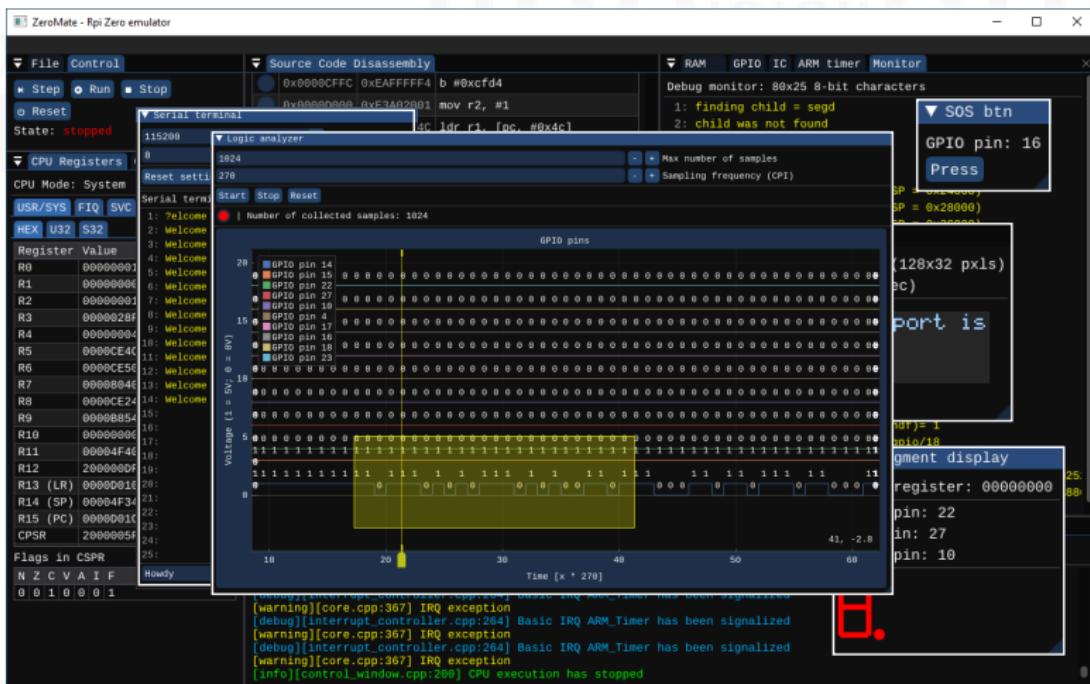
Obrázek 8: d) Sedmisegmentový LED displej řízený posuvným registrém

## ZeroMate - externí periferie



Obrázek 8: e) Sériový terminál (UART)

## ZeroMate - externí periferie



Obrázek 8: f) Logický analyzátor



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

## vývoj

- C++20 (Clang, GCC a MSVC)
- CMake
- Linux, Windows, MacOS\*

## CI pipeline

- ► Codecov
- ► Codacy
- ► Doxygen

- 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  $\approx 78\%$**

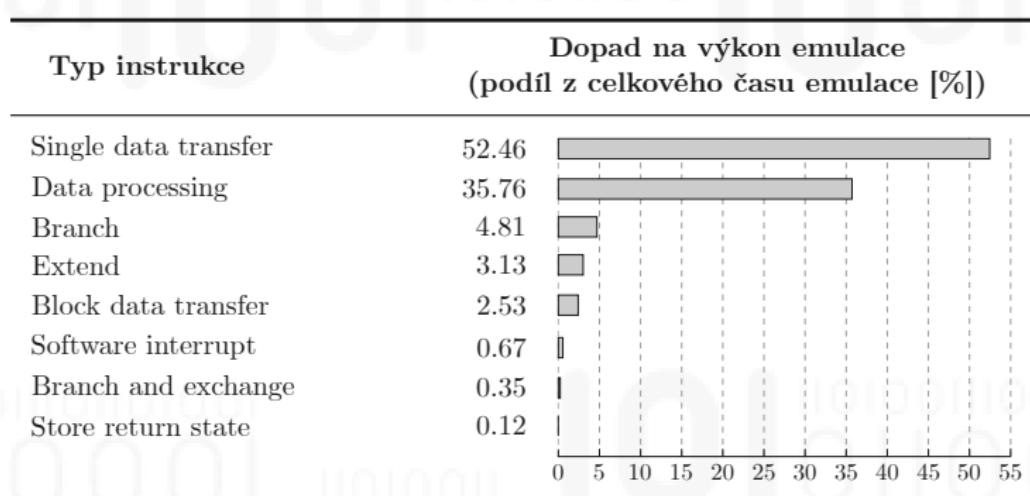
## 2) Funkční testy

- BCM2835 periferie
- připravená sada příkladů (.ELF)
  - UART, FPU, GPIO, I<sup>2</sup>C, plánování procesů (EDF, RTL), ARM timer, ...
  - součástí repozitáře jako examples

## 3) Aplikační testy, GUI

- KIV/OS cvičení
- dotazník hodnocení celkové použitelnosti

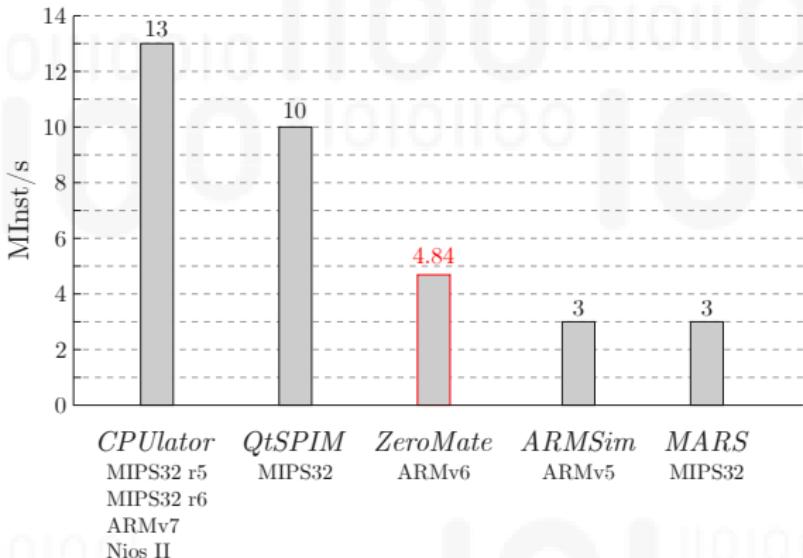
- **výkon emulátoru  $\approx 4\text{-}5 \text{ Minst/s}$**  ([► KIV-RTOS](#))
- analýza rychlosti emulace jednotlivých instrukcí
  - $\Rightarrow$  potenciální optimalizace



1

<sup>1</sup> 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 ...

Děkuji Vám za pozornost

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



## Otázka

Byla zvažována možnost modifikace existujících emulátorů, namísto vývoje vlastního?

## Odpověď

analýza existujících emulátorů

- ① omezená podpora pro pokročilé systémové operace
- ② limitovaná/nestandardní podpora externích periferií

⇒ jejich modifikace nebyla považována za vhodné řešení

- modifikace a údržba legacy C kódu
- licenční podmínky

## Otázka

Jaký vliv měla implementace periférií na celkový výkon emulátoru?

## Odpověď

- R/W instrukce ⇒ lookup periferie dle adresy
  - většina R/W instrukcí přistupuje do RAM ⇒ optimalizace?
- implementace R/W callbacku
- **ISystem\_Clock\_Listener**

## Otázka

Jaká je rychlosť emulátoru vůči reálnému Raspberry Pi?

## Odpověď

- měření provedeno na osobním PC
- výkonnější PC/server ⇒ dosažení lepsího výsledku?
- teoreticky
  - MIPS(ZeroMate) ≈ 4.8
  - MIPS(ARM11) ≈ 740 (► Dhrystone)
  - ⇒ reálný HW je ≈ 154x rychlejší