VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Komponenta serveru pro podporu výuky teoretické informatiky - Simulace Turingova stroje RAMem

Bc. Jakub Koběrský

Semestrální práce

Vedoucí práce: Ing. Martin Kot, Ph.D.

Ostrava, 2025

Zadání semestrálního projektu

Název: Komponenta serveru pro podporu výuky teoretické informatiky - Simulace Turingova stroje RAMem

Rok zadání: 2024/2025

Vedoucí: Ing. Martin Kot, Ph.D.

Student: Jakub Koběrský

Zaměření: Teoretická informatika

Forma studia: prezenční

Text zadání:

1. V rámci studentských prací vzniká sada dynamických webových stránek pro výuku předmětů teoretické informatiky.
2. Cílem tohoto projektu je doplnit stránku zabývající se simulací Turingova stroje strojem RAM
3. Uživatel bude mít možnost zadat specifikaci Turingova stroje a stránka mu zobrazí zdrojový kód pro stroj RAM.
4. Poté bude možné sledovat současně výpočet obou strojů na stejném (nebo odpovídajícím) vstupu.
5. Pro Turingův stroj bude předpřipravených alespoň 5 specifikací, aby si mohl uživatel vše vyzkoušet i bez zadávání vlastního vstupu.

Literatura:

Podle pokynů vedoucího semestrálního projektu.

Obsah

Seznam použitých zkratek a symbolů

RAM	–	Random Access Machine
IP	–	Instruction Pointer, ukazatel na právě prováděnou instrukci
JSON	–	JavaScript Object Notation
JS	–	JavaScript
DOM	–	Document Object Model, rozhraní pro manipulaci s HTML
UI	–	User Interface, uživatelské rozhraní
UX	–	User Experience, uživatelská zkušenost

Seznam obrázků

Kapitola 1

Úvod

Cílem této semestrální práce je vytvoření aplikace pro simulaci Turingova stroje strojem RAM, sloužící k výuce teoretické informatiky. Aplikace obsahuje simulaci Turingova stroje s předpřipravenými příklady, které jsou následně přeloženy do kódu stroje RAM. Oba stroje lze poté zároveň krokovat a v reálném čase vidět obsah pásek a paměti, včetně stavu a pozice strojů. K předpřipraveným příkladům lze definovat i své vlastní stroje, které je možné sdílet mezi jednotlivými zařízeními.

Práce je rozdělena do několika kapitol, kdy ve 2. kapitole je popsán teoretický základ obou strojů nutný pro pochopení průběhu simulace, následován popisem několika vybraných předpřipravených strojů a popisem samotného algoritmu simulace Turingova stroje. V další kapitole je probrán stručný popis použitých technologií, jak jsou použity a jaké výhody do aplikace přináší. V neposlední řadě, v kapitole 4, je probrána samotná webová aplikace, její návrh, implementace, uživatelské rozhraní a jsou zde také natíněny možné způsoby rozšíření aplikace.

Webová aplikace je v průběhu obhajoby dostupná na adrese `https://ram.koberskyj.cz/`, zprostředkované přes Github Pages [1].

Kapitola 2

Turingovy stroje a stroje RAM

2.1 Turingův stroj

Jedná se o idealizovaný model počítače, který lze použít ke zkoumání hranic algoritmicky řešitelných úloh. Stroj popsal v roce 1936 Alan Turing a od té doby je to jeden z klíčových formálních nástrojů pro definici pojmu `algoritmus` a pro charakterizaci rekurzivně vyčíslitelných jazyků [2]. Je na něm postavena **Churchova-Turingova teze**, podle které může být každý algoritmus realizován Turingovým strojem. Programovací jazyky a stroje, které umožňují vyjádřit libovolný takovýto algoritmus, se označují jako **Turingovsky úplné** [3].

Existuje několik různých variant Turingova stroje, všechny ale obsahují nějakou verzi *nekonečné pásky*, *čtecí hlavu* a *přepisovací pravidla*. V této práci je Turingův stroj implementován s oboustranně nekonečnou páskou, samotná simulace je však limitována na stroj s jednostranně nekonečnou páskou. Více je konkrétní implementace popsána v kapitole 4.1.

2.1.1 Definice stroje

Formálně je Turingův stroj definován jako šestice $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, kde [3]:

- Q je konečná neprázdná množina stavů,
- Γ je konečná neprázdná množina páskových symbolů,
- $\Sigma \subseteq \Gamma$ je konečná neprázdná množina vstupních symbolů,
- $\delta : (Q - F) \times \Gamma \to Q \times \Gamma \times \{-1, 0, +1\}$ je přechodová funkce,
- $q_0 \in Q$ je počáteční stav,
- $F \subseteq Q$ je množina koncových stavů.

Výsledek rozdílu $\Gamma - \Sigma$ je vždy speciální znak $\square$, který označuje prázdný znak (blank).

2.1.2 Příklady Turingových strojů

Aplikace obsahuje základních 5 příkladů, které pracují následovně:

- **Shodné délky**[1] - stroj přijímá slova, kde se symboly a, b, c opakují n-krát za sebou,
- **Zrcadlit** - stroj zrcadlí symboly a, b směrem doprava,
- **Kopírovat** - stroj kopíruje symboly a, b směrem doprava,
- **Palindrom** - stroj přijímá slova složená ze symbolů a, b, která jsou z obou stran stejná,
- **Sudý počet a** - stroj přijímá slova, kde se vyskytuje sudý počet symbolu a.

2.2 RAM stroj

RAM stroj (Random-Access Machine) představuje turingovsky úplný, idealizovaný model počítače. Skládá se z *programové jednotky* (sekvence instrukcí), *pracovní paměti* a *vstupní* a *výstupní* pásky. Buňky paměti i pásky obsahují pouze celá čísla ($\mathbb{N}$), nelze do nich tedy uložit znak. Pracovní paměť slouží pro vstup i výstup a je indexována od 0 (R_0) do n (R_n). Vstupní páska slouží pouze pro čtení a naopak výstupní pouze pro zápis. Stejně jako u Turingova stroje existuje i zde řada modifikovaných definic RAM stroje. Stroj navíc obsahuje ukazatel na právě prováděnou instrukci v programové jednotce (IP) a v základu obsahuje tyto instrukce [3]:

- $R_i := c$
- $R_i := R_j$
- $R_i := [R_j]$
- $[R_i] := R_j$
- $R_i := R_j$ `op` R_k - aritmetické instrukce, `op` $\in \{+, -, *, /\}$
  nebo $R_i := R_j$ `op` c
- `if` $(R_i$ `rel` $R_j)$ `goto` l - podmíněný skok, `rel` $\in \{=, \neq, \leq, \geq, <, >\}$
  nebo `if` $(R_i$ `rel` $c)$ `goto` l
- `goto` l - nepodmíněný skok
- $R_i :=$ `READ`() - čtení ze vstupu
- `WRITE`(R_i) - zápis na výstup

[1] Jedná se o ukázkový příklad z prezentace předmětu teoretické informatiky

- `halt` - zastavení programu

Všechny uvedené instrukce jsou v aplikaci plně podporovány a jejich detailní popis se nachází v kapitole 4.1.

2.3 Simulace Turingova stroje strojem RAM

Jelikož Turingův stroj pracuje se znaky, se kterými stroj RAM pracovat neumí, je zapotřebí nějakého slovníku, co by nám ke každému znaku přiřadil číselnou hodnotu. Například tímto předpisem:

$$\Sigma \longrightarrow \mathbb{N}, \qquad a_1 \to 1,\ a_2 \to 2,\ \ldots,\ a_n \to n$$

Hodnota 0 je rezervována pro znak $\square$, aby z počátku nulová (nezapsaná) buňka paměti RAM odpovídala prázdnému symbolu na pásce. Samotný program stroje RAM lze tvořit následujícím způsobem:

1. Program začne skokem na aktuální stav.
2. Pro každý stav definuj návěští, které si načte aktuální symbol na pásce. Poté následuje série skoků, které „rozřadí“ běh programu do konkrétních stavů s daným symbolem.
3. Pro každý stav s konkrétním symbolem definuj návěští. Aktualizuj symbol na pozici čtecí hlavy podle přepisovacího pravidla přechodové funkce a vykonej posun ($\{-1, 0, +1\}$). Pokud je nový stav koncový - konec programu (`halt`), jinak skoč na tento nový stav (zpět krok 2).

Kapitola 3

Použité technologie

V této kapitole jsou popsány základní technologie a jejich balíčky, které jsou v této práci použity.

3.1 JSON

Jedná se o standardní formát textu, přes který lze reprezentovat strukturovaná data. Výhodou oproti binárnímu kódu či standardnímu textového souboru je čitelnost, kdy jsou definovaná jasná pravidla formátu, který je snadno čitelný jak pro uživatele, tak počítače. Tento formát je založen na základně syntaxe JavaScript (JS) objektů a slouží především pro ukládání dat a komunikaci mezi zařízeními [4].

3.2 TypeScript

TypeScript je o open-source jazyk vyvinutý firmou Microsoft, který především rozšiřuje JS syntaxi o typy a jejich kontrolu. Zdrojový kód se nejprve přeloží (transpiluje) do JS kódu, který lze následne spustit ve webovém prohlížeči (nebo na serveru přes například Node.js). Mezi hlavní přednosti tohoto jazyka patří zejména dřívější odhalení chyb díky typové kontrole a našeptávání během psaní kódu [5].

3.3 Webové technologie

Webová stránka v dnešní době již není většinou skládaná pouze z HTML a CSS, mnohé stránky se neobejdou bez různé interaktivity pomocí jazyka JavaScript, případně WebAssembly. Stránka pak může díky této interaktivity komunitovat přímo s lokálním počítačem či internetem, může provádět různé dynamické upravy v DOM (Document Object Model) a slouží i ke zprostředkování složitějších animací.

3.3.1 Použité API a knihovny

Pro zjednodušení práce a přidání možnosti ukládání dat byly použity tyto webové technologie:

- **localStorage**, sloužící pro lokální dlouhodobé uložení dat ve formátu klíč-hodnota, která je formátu JSON [6].

- **Shadcn**, pro již předpřipravené komponenty, jako jsou například tlačítka, ikonky a vyskakovací okna. Samotná knihovna využívá Radix UI a Lucide ikonky [7].

- **Tailwind CSS**, který slouží pro jednoduché stylování přímo přes třídy na jednotlivých HTML elementech nebo jejich ekvivalentních prvků [8].

3.3.2 React.js

Samotná aplikace je napsána za pomocí frameworku React.js, který výrazně zjednodušuje zejména interaktivitu aplikace a rozděluje kód do jednotivých komponent.

React.js slouží ke tvorbě webových uživatelských rozhraní v JS nebo, v tomto případě, v jazyce TypeScript. Hlavní myšlenkou knihovny je deklarativní a komponentový přístup - rozhraní se skládá z malých, znovupoužitelných kompopent, které popisují, jak má aplikace vypadat pro konkrétní stav dat. HTML elementy komponent a i celkově jednotlivé kompomenty jsou v tomto případě reprezentovány ekvivalentní strukturou JSX/TSX s minimálními změnami. Díky této myšlence lze snadno vytvářet komplexní UI prvky, které jsou na pozadí složeny z menších a menších prvků. React kód je převážně spouštěn na straně uživatele v prohlížeči, avšak existují i alternativy v podobě mobilních aplikací přes React Native nebo kódů pro server přes Next.js. Při změně jednotlivých komponent React efektivně aktualizuje pouze dotčené části DOM (Document Object Model) pomocí své vlastní implementace zvané „virtual DOM“ [9].

3.4 Git

Jedná se o jeden z nejrozšířenějších verzovacích systémů, který výrazně usnadňuje práci v týmu, řešení záloh evidenci jednotlivých změn. Data mohou přes něj mohou být uložena jak lokálně, tak i na serveru [10]. V této práci byl použit server **GitHub**, který umožňuje správu privátních i věřejných projektů, obsahuje uživatelky přívětivé rozhraní a má i několik dodatečných funkcí navíc, jako jsou například GitHub pages [11].

V této práci je použit verzovací systém zejména kvůli tomu, aby měl vedoucí práce vždy k dispozici aktuální verzi projektu, včetně historie aktivity. Aplikace je také zveřejněna na již zmíněném GitHub Pages, jelikož je tato funkce zdarma a odpadá tak nutnost řešit hosting na jiných službách.

3.4.1 GitHub Pages

GitHub Pages slouží pro automatické publikování webových stránek s předpřipravenými či vlastními URL [1]. Vzhledem k použití frameworku je však nutné aplikaci nejdříve sestavit, a pak až následně publikovat. Aplikace se publikuje do vlastní větve v repozitáři projektu, takže nemusí být součástí zdrojových kódů.

Kapitola 4

Implementace

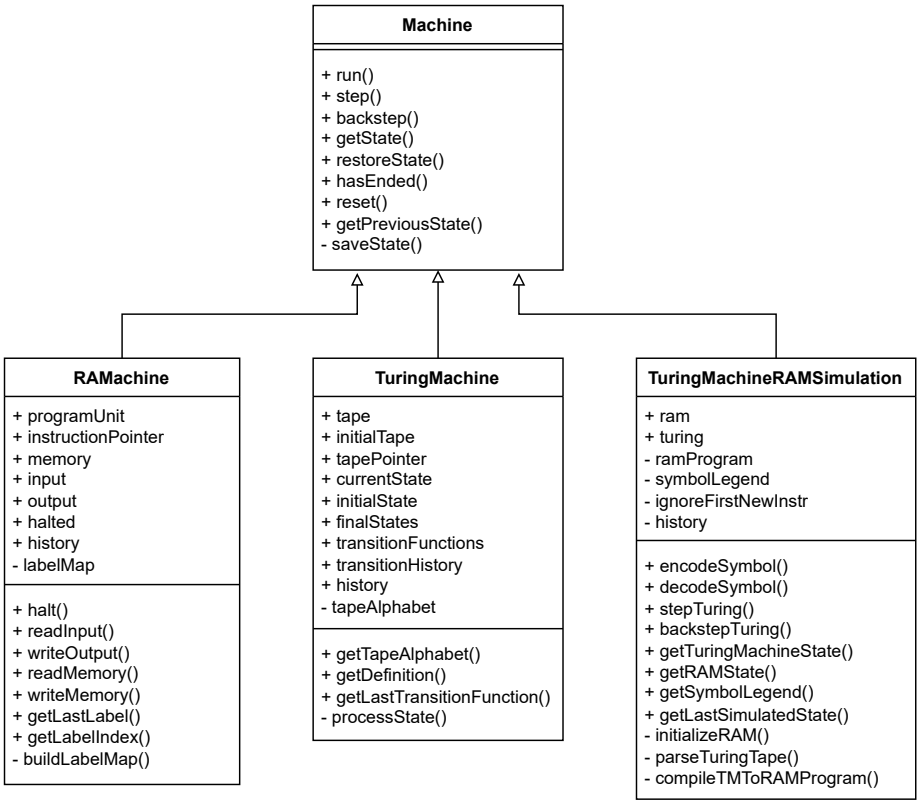
Tato kapitola obsahuje již konkrétní popis implementace aplikace, její návrh a je zde popsáno i UI, včetně UX. Samotná aplikace je celá v jednom React.js projektu a obsahuje již zmíněné technologie z kapitoly 3.

4.1 Implementace strojů

4.2 Průběh simulace

4.3 Uživatelské rozhraní

4.4 Možná rozšíření



Obrázek 4.1: Class diagram implementovaných strojů

Kapitola 5

Závěr

Aplikace je hotová.

Literatura

1. PAGES, GitHub. *GitHub Pages*. GitHub Pages, 2019. Dostupné také z: `https://pages.github.com/`.
2. GEEKSFORGEEKS. *Turing Machine in TOC*. GeeksforGeeks, 2016-05. Dostupné také z: `https://www.geeksforgeeks.org/turing-machine-in-toc/`.
3. SAWA, Zdeněk. *Teoretická informatika* [online]. [B.r.]. [cit. 2025-05-22]. Dostupné z: `https://www.cs.vsb.cz/sawa/ti/slides/ti-slides-01.pdf`.
4. BRAY, T. The JavaScript Object Notation (JSON) Data Interchange Format. *www.rfc-editor.org*. 2017-12. Dostupné z DOI: `10.17487/RFC8259`.
5. W3SCHOOLS. *TypeScript Introduction*. www.w3schools.com, [b.r.]. Dostupné také z: `https://www.w3schools.com/typescript/typescript_intro.php`.
6. MOZILLA. *Window.localStorage*. MDN Web Docs, 2019-03. Dostupné také z: `https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage`.
7. SHADCN. *Introduction*. ui.shadcn.com, [b.r.]. Dostupné také z: `https://ui.shadcn.com/docs`.
8. TAILWINDCSS. *Tailwind CSS - Rapidly Build Modern Websites without Ever Leaving Your HTML*. tailwindcss.com, [b.r.]. Dostupné také z: `https://tailwindcss.com/`.
9. MOZILLA. *Getting started with React - Learn web development | MDN*. MDN Web Docs, 2024-12. Dostupné také z: `https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Frameworks_libraries/React_getting_started`.
10. GIT. *Git*. Git-scm.com, 2024. Dostupné také z: `https://git-scm.com/`.
11. GITHUB. *GitHub*. GitHub, 2025. Dostupné také z: `https://github.com/`.