B. Používateľská príručka

B.1 Úvod

Simulátor SKRUPULUS MIPS slúži ako učebná pomôcka pre študentov a nadšencov informatiky, ktorí sa chcú vzdelávať v oblasti prúdového spracovania inštrukcií (pipeline) v architektúre MIPS. Umožňuje jednoduchým a vizuálnym spôsobom sledovať, ako inštrukcie postupne prechádzajú jednotlivými fázami spracovania, čím napomáha lepšiemu pochopeniu fungovania procesora na nízkej úrovni.

Aplikácia je implementovaná ako progresívna webová aplikácia (PWA), čo umožňuje nainštalovať si ju ako natívnu aplikáciu na svoje zariadenie. Je navrhnutá tak, aby plne fungovala aj v offline režime.

B.1.1 Spustenie simulátora

Simulátor je dostupný online na adrese `https://www.skrupulus.com`. V prípade, že stránka nie je dostupná, je možné použiť lokálnu verziu simulátora podľa inštrukcií uvedených v súbore `README.md`, ktorý je súčasťou projektu.

B.2 Prúdové spracovávanie inštrukcií

Prúdové spracovávanie inštrukcií je technika používaná v moderných mikroprocesoroch, pri ktorej sa zvyšuje priepustnosť inštrukcií v procesore. Keď inštrukcia prejde do nasledujúcej fázy, tak procesor do predošlej fázy načíta nasledujúcu inštrukciu paralelne.

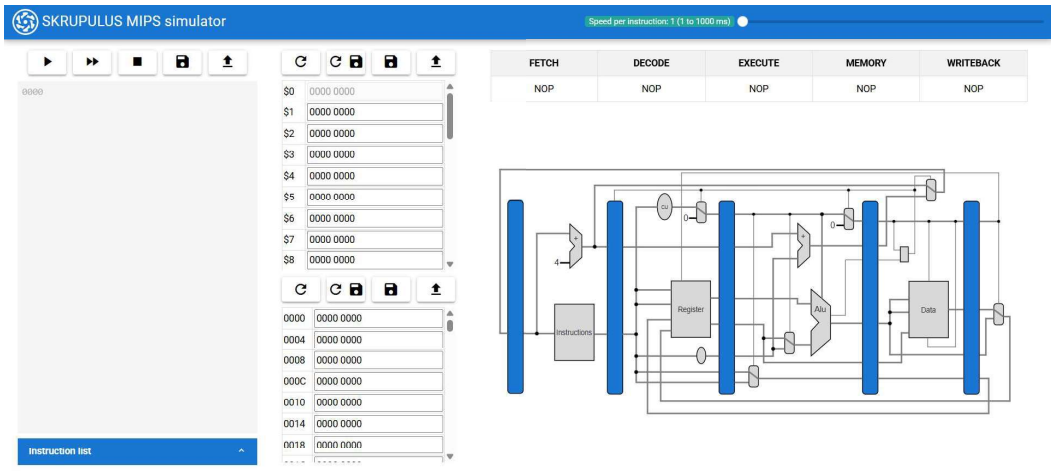
MIPS obsahuje prúdové spracovávanie v piatich fázach, tu je ich krátke vysvetlenie:

- **FETCH**: Načítava inštrukcie programu.
- **DECODE**: Dekóduje inštrukciu a získa dáta z registrov.
- **EXECUTE**: Vykoná aritmetické alebo logické inštrukcie.
- **MEMORY**: Zapisuje alebo číta dáta z pamäte.
- **WRITEBACK**: Zapisuje výsledky do registrov procesora.

B.3 Hazardy

Pri paralelnom spracovávaní v MIPS môže dôjsť k situáciám, kedy chceme pristúpiť k registrom alebo pamäti, ktorá bola už prepísaná inštrukciou, ale ešte inštrukcia nezapísala túto zmenu. Týmto situáciám sa predchádza tak, že procesor medzi tieto dve inštrukcie vloží prázdne inštrukcie, ktoré nič nerobia. Čo spomaľuje priepustnosť procesora.

B.4 Návod používanie



Obr. B.1: SKRUPULUS MIPS simulator

Editor kódu nachádzajúci sa v pravej časti aplikácie slúži na písanie MIPS inštrukcií. Správnu syntax inštrukcií nájdete na konci manuálu a v aplikácii pod editorom. Nad editorom sa nachádza ovládací panel simulácie.

Tlačidlo	**Popis**
Krok	Slúži na krokovanie programu každé stlačenie spustí nasledujúcu inštrukciu.
Spustiť	Spustí program rýchlo. Rýchlosť sa dá upraviť v hornom paneli aplikácie vpravo.
Zastaviť	Zastaví vykonávaný program, či už rýchle spustenie alebo krokovanie.
Uložiť	Uloží textový súbor s kódom inštrukcií.
Nahrať	Otvorí možnosť nahrávania, kde môžeme vložiť kód, ktorý sme predtým uložili.

Tabuľka B.1: Vysvetlenie tlačidiel ovládacieho panelu

Registre a pamäť sa nachádzajú napravo od editora kódu. Vizualizácia registrov obsahuje 32 registrov s 32-bitovými hexadecimálnymi hodnotami. Register 0 má vždy hodnotu 0. Vizualizácia pamäti rozdeľuje pamäť na slová o veľkosti štyroch bajtov. Nad registrami a pamäťou sa nachádza ovládací panel.

Tlačidlo	**Popis**
Vynulovať	Vynuluje registre alebo pamäť.
Obnoviť	Znovu vloží posledné nahrané registre alebo pamäť.
Uložiť	Uloží textový súbor s hodnotami v registroch alebo pamäti.
Nahrať	Otvorí okno, kde môžeme nahrať textový súbor s pamäťou alebo registrami.

Tabuľka B.2: Vysvetlenie ovládacích panelov registrov a pamäti

Ako prvé na vytvorenie spustiteľného kódu ho musíte napísať do editora, ktorý sa nachádza v ľavej časti simulátora. Zoznam a syntax inštrukcií nájdete pod editorom a na konci manuálu.

B.4.1 Príklad: Výpočet n-faktoriálu

Nasledujúci program vypočíta n-faktoriál:

```
LI $2, 1          # načítame počiatočnú hodnotu 1
NOP
NOP               # počkáme, kým sa 1 zapíše do registra
MUL $2, $2, $1    # vynásobíme registre 1 a 2
SUBI $1, $1, 1    # znížime počítadlo
NOP
NOP               # čakáme, kým sa zníži počítadlo
BNE $1, $0, 0003  # cyklus, kým počítadlo nebude 0
NOP
NOP
NOP
NOP
```

Do registra $1 zapíšeme číslo n nad ktorým chceme robiť faktoriál. Inštrukcie, ktoré zapisujú do registrov, zapíšu až v poslednej fáze. Preto ak po takejto inštrukcii ideme čítať novú hodnotu, musíme počkať dva cykly, kým sa zapíše zmena.

B.5 Zoznam inštrukcií

Nasledujúca tabuľka obsahuje zoznam podporovaných inštrukcií spolu s ich syntaxou a popisom.

Tabuľka B.3: Zoznam inštrukcií MIPS simulátora

Inštrukcia	**Syntax**	**Popis**
ADD	ADD $rd, $rs, $rt	Sčíta $rs a $rt, výsledok uloží do $rd. Príklad: ADD $1, $2, $3 # $1 = $2 + $3
ADDI	ADDI $rt, $rs, immediate	Sčíta $rs s konštantou, výsledok uloží do $rt. Príklad: ADDI $1, $2, 5 # $1 = $2 + 5
SUB	SUB $rd, $rs, $rt	Odčíta $rt od $rs, výsledok uloží do $rd. Príklad: SUB $1, $2, $3 # $1 = $2 - $3
SUBI	SUBI $rt, $rs, immediate	Odčíta konštantu od $rs, výsledok uloží do $rt. Príklad: SUBI $1, $2, 5 # $1 = $2 - 5
MUL	MUL $rd, $rs, $rt	Vynásobí $rs a $rt, výsledok uloží do $rd. Príklad: MUL $1, $2, $3 # $1 = $2 * $3
MULU	MULU $rd, $rs, $rt	Vynásobí neznamienkové hodnoty $rs a $rt, výsledok uloží do $rd. Príklad: MULU $1, $2, $3 # $1 = $2 * $3 (neznamienkové)
DIV	DIV $rd, $rs, $rt	Vydelí hodnotu v $rs hodnotou v $rt. Kvocient sa uloží do LO, zvyšok do HI. Príklad: DIV $2, $3 # LO = $2 / $3

Inštrukcia	**Syntax**	**Popis**
DIVU	DIVU $rd, $rs, $rt	Vydelí neznamienkovú hodnotu v $rs hodnotou v $rt. Kvocient sa uloží do LO, zvyšok do HI. Príklad: DIVU $2, $3 # LO = $2 / $3 (neznamienkové)
MFHI	MFHI $rd	Skopíruje hodnotu z registra HI do $rd. HI obsahuje zvyšok z delenia. Príklad: MFHI $1 # $1 = HI
SLL	SLL $rd, $rt, shift	Posunie bity v $rt o shift bitov doľava, výsledok uloží do $rd. Príklad: SLL $1, $2, 2 # $1 = $2 « 2
SRL	SRL $rd, $rt, shift	Posunie bity v $rt o shift bitov doprava, výsledok uloží do $rd. Príklad: SRL $1, $2, 2 # $1 = $2 » 2
AND	AND $rd, $rs, $rt	Bitovo porovná $rs a $rt operáciou AND, výsledok uloží do $rd. Príklad: AND $1, $2, $3 # $1 = $2 & $3
ANDI	ANDI $rt, $rs, immediate	Bitovo porovná $rs s konštantou operáciou AND, výsledok uloží do $rt. Príklad: ANDI $1, $2, FF # $1 = $2 & FF
OR	OR $rd, $rs, $rt	Bitovo porovná $rs a $rt operáciou OR, výsledok uloží do $rd. Príklad: OR $1, $2, $3 # $1 = $2 | $3

Inštrukcia	**Syntax**	**Popis**
ORI	ORI $rt, $rs, immediate	Bitovo porovná $rs s konštantou operáciou OR, výsledok uloží do $rt. Príklad: ORI $1, $2, FF # $1 = $2 | FF
NOR	NOR $rd, $rs, $rt	Vykoná bitovú operáciu NOR nad $rs a $rt, výsledok uloží do $rd. Príklad: NOR $1, $2, $3 # $1 = ($2 | $3)
XOR	XOR $rd, $rs, $rt	Vykoná bitovú operáciu XOR nad $rs a $rt, výsledok uloží do $rd. Príklad: XOR $1, $2, $3 # $1 = $2 $\hat{}$ $3
XORI	XORI $rt, $rs, immediate	Vykoná XOR medzi $rs a konštantou, výsledok uloží do $rt. Príklad: XORI $1, $2, FF # $1 = $2 $\hat{}$ FF
BEQ	BEQ $rs, $rt, address	Vetví program na adresu, ak sú $rs a $rt rovnaké. Príklad: BEQ $1, $2, 0000 # Skok na adresu 0000, ak $1 = $2
BNE	BNE $rs, $rt, address	Vetví program na adresu, ak $rs a $rt nie sú rovnaké. Príklad: BNE $1, $2, 000A # Skok na adresu 000A, ak $1 != $2

Inštrukcia	**Syntax**	**Popis**
LW	LW $rt, offset($rs)	Načítava slovo z pamäte na adrese ($rs + offset) do $rt. Príklad: LW $1, 4($2) # $1 = Pamäť[$2 + 4]
SW	SW $rt, offset($rs)	Uloží slovo z $rt do pamäte na adrese ($rs + offset). Príklad: SW $1, 4($2) # Pamäť[$2 + 4] = $1
LI	LI $rd, immediate	Načítava 16-bitovú konštantu do registra $rd. Príklad: LI $1, 100 # $1 = 100
LUI	LUI $rt, immediate	Načítava 16-bitovú hodnotu do hornej polovice registra $rt. Príklad: LUI $1, FFFF # $1 = FFFF0000
NOP	NOP	Nič nerobí.