Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Bakalárska práca

Používateľský manuál k simulátoru SKRUPULUS MIPS simulator

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.

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.

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.

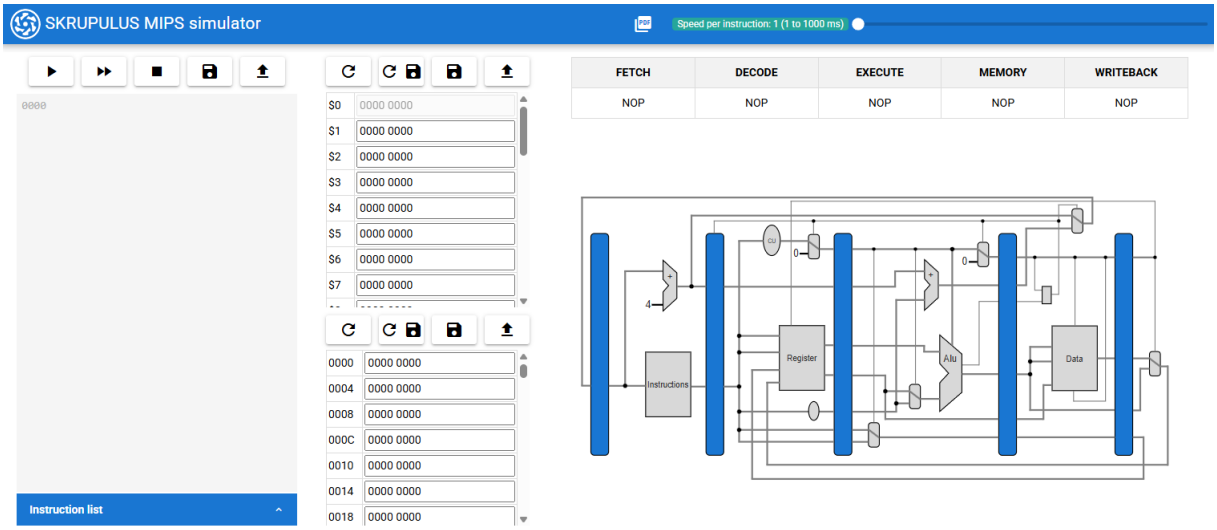
MEMORY – Zapisuje alebo číta dáta z pamäti.

WRITEBACK – Zapisuje výsledky do registrov procesora.

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.

Návod na používanie



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

Tabuľka z vysvetlením tlačidiel ovládacieho panelu:

Tlačidlo	Popis
	Slúži na krokovanie programu každé stlačenie spustí nasledujúcu inštrukciu.
	Spustí program rýchlo. Rýchlosť sa dá upraviť v hornom paneli aplikácie vpravo.
	Zastaví vykonávaný program, či už rýchle spustenie alebo krokovanie.
	Uloží textový súbor s kódom inštrukcií.

	Otvorí možnosť nahrávania kde môžeme vložiť kód, ktorý sme predtým uložili.

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.

Tabuľka z vysvetlením ovládacích panelov registrov a pamäti:

Tlačidlo	Popis
	Vynuluje registre/pamäť.
	Znovu vloží posledné nahrané registre/pamäť.
	Uloží textový súbor s hodnotami v registroch/pamäti.
	Otvorí okno kde môžeme nahrať textový súbor s pamäťou alebo registrami.

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.

Ako príklad si uvedieme program ktorý vypočíta n-faktoriál.

```
LI $2,1          # načítame počiatočnú hodnotu 1
NOP
NOP              # počkáme kým sa 1 zapíše do registra
MUL $2,$2,$1     # Vynásobíme regitre 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

NOP
```

Do registra $1 zapíšeme číslo n nad ktorým chceme robiť faktoriál.

Inštrukcie, ktoré zapisujú do registrov zapíšu až v poslednej fázy. Preto ak po takejto inštrukcií ideme čítať novú hodnotu musíme počkať dva cykly kým sa zapíše zmena.

Inštrukčná sada

Názov	Syntax	Opis
ADD	ADD $rd, $rs, $rt	Sčíta hodnoty v registroch $rs a $rt a výsledok uloží do registra $rd. Príklad: ADD $1, $2, $3 # $1 = $2 + $3
ADDI	ADDI $rt, $rs, immediate	Sčíta hodnotu v registri $rs s okamžitou hodnotou (immediate) a výsledok uloží do registra $rt. Okamžité hodnoty sú 16-bitové celé čísla. Príklad: ADDI $1, $2, 100 # $1 = $2 + 100
SUB	SUB $rd, $rs, $rt	Odčíta hodnotu v registri $rt od hodnoty v $rs a výsledok uloží do registra $rd. Príklad: SUB $1, $2, $3 # $1 = $2 - $3
SUBI	SUBI $rt, $rs, immediate	Odčíta okamžitú hodnotu od hodnoty v $rs a výsledok uloží do $rt. Príklad: SUBI $1, $2, 100 # $1 = $2 - 100
MUL	MUL $rd, $rs, $rt	Vynásobí hodnoty v registroch $rs a $rt a výsledok uloží do $rd. Príklad: MUL $1, $2, $3 # $1 = $2 * $3
MULU	MULU $rd, $rs, $rt	Vynásobí neznamienkové hodnoty v registroch $rs a $rt a 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 a zvyšok do HI. Príklad: DIV $2, $3 # LO = $2 / $3
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 a 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 a výsledok uloží do $rd. Príklad: SRL $1, $2, 2 # $1 = $2 >> 2
AND	AND $rd, $rs, $rt	Bitovo porovná $rs a $rt operáciou AND a výsledok uloží do $rd. Príklad: AND $1, $2, $3 # $1 = $2 & $3
ANDI	ANDI $rt, $rs, immediate	Bitovo porovná $rs s okamžitou hodnotou operáciou AND a výsledok uloží do $rt. Príklad: ANDI $1, $2, FF # $1 = $2 & FF
OR	OR $rd, $rs, $rt	Bitovo porovná $rs a $rt operáciou OR a výsledok uloží do $rd. Príklad: OR $1, $2, $3 # $1 = $2 | $3
ORI	ORI $rt, $rs, immediate	Bitovo porovná $rs s okamžitou hodnotou operáciou OR a 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 ^ $3
XORI	XORI $rt, $rs, immediate	Vykoná XOR medzi $rs a okamžitou hodnotou, výsledok uloží do $rt. Príklad: XORI $1, $2, FF # $1 = $2 ^ 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, adress	Vetví program na adresu, ak $rs a $rt nie sú rovnaké. Príklad: BNE $1, $2, 000A # Skok na adresu 000A ak $1 != $2
LW	LW $rt, offset($rs)	Načíta 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	Načíta 16-bitovú konštantu do registra $rd. Príklad: LI $1, 100 # $1 = 100
LUI	LUI $rt, immediate	Načíta 16-bitovú hodnotu do hornej polovice registra $rt. Príklad: LUI $1, FFFF # $1 = FFFF0000
NOP	NOP	Nič nerobí.