ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd

Úvod do počítačových architektur

Program pro procesor MIPS – Výpočet klouzavého průměru

Autor: **Antonín NEUMANN**
Akademický rok: 2013/2014

Zadání

Vytvořte program pro výpočet klouzavého průměru na procesoru MIPS a simulátoru SPIM.

Postupně se budou zadávat čísla int16, na výstupu se bude průběžně zobrazovat průměr posledních 8 zadaných čísel.

Požadavky na řešení

- Všechna vstupní data jsou zadávána z klávesnice, pokud není v zadání uvedeno jinak.
- Před každým vstupem se musí zobrazit „prompt“, ze kterého je zřejmé, co se má zadat.
- Všechny výstupy se zobrazují na displeji v přiměřeném formátu.
- Pro řešení úloh nejsou přípustné velmi nevhodné algoritmy.
- Případné nejasnosti zadání se konzultují se cvičícím.

Popis programu (algoritmu)

Program vyzve uživatele k zadání čísla. Pokud toto číslo není v rozsahu 16 bitového integeru, je o tom uživatel informován a vyzván, aby číslo zadal znovu. Je-li hodnota platná, je uložena do příslušného registru (po uložení osmého čísla se začíná ukládat opět od začátku – přepisují se tedy původní hodnoty) a je zavolán podprogram pro výpočet průměru. Nakonec je na konzoli vypsáno všech 8 aktuálně uložených čísel a jejich průměr.

Dojde-li k zadání čísla 33000 je program ukončen.

Optimalizace

V programu jsem se snažil o minimalizaci volání instrukce „`nop`“. K této minimalizaci jsem využil především přesun instrukce předcházející skoku až za něj.

Pseudoinstrukce

Simulátor SPIM, resp. překladač poskytuje programátorovi určitý komfort v podobě pseudoinstrukcí. Tyto pseudoinstrukce poté překladač sám převádí na instrukce, kterým rozumí procesor.

Ukázka pseudoinstrukcí v mém kódu:

Pseudoinstrukce	Instrukce procesoru MIPS
`li $v0, 4`	`addi $v0, $0, 4`
`la $a0, vstup`	`lui $a0, upper(vstup)` `ori $a0, $a0, lower(vstup)`
`la $a0, 0x12345678`	`lui $a0, 0x1234` `ori $a0, $a0, 0x5678`
`move $a0, $t0`	`add $a0, $t0, $0`

Datový hazard a zpožděné čtení dat z paměti

Tzv. datový hazard nastává, pokud chce nějaká instrukce přečíst obsah paměti bezprostředně po instrukci, která do tohoto místa v paměti nějaká data zapsala.

Ve svém programu jsem na potenciální datový hazard narazil v podprogramu počítajícím aktuální průměr, kdy druhá instrukce pro sčítání používá hodnotu uloženou předchozí instrukcí pro sčítání.

```
add $v0, $t0, $t1
add $v0, $v0, $t2
```

Zpožděné čtení dat z pamětí se projevuje tehdy, je-li ihned po načtení dat z paměti vykonána instrukce, která tato nově načtená data dále zpracovává.

Tento jev jsem ve svém programu nenalezl.

Závěr

Myslím, že jsem pochopil základní princip fungování procesoru MIPS a programování pro něj. Nejsložitější částí pro mě bylo vyřešit správnou funkčnost podprogramů a optimalizovat program pro používání zpožděných instrukcí pro skoky a čtení z paměti.