Jméno, příjmení: Login:

A0B36APO – Architektura počítačů

3. domácí úkol

Pipeline, optimalizace kódu, predikce skoků

LS 2010/2011

Katedra počítačů, FEL, ČVUT v Praze

Termín odevzdání: nejpozději do **22.4.2011, 12:00**

Pište do připravených papírů, pište **čitelně**, nepřikládejte k řešení žádné další papíry!

V příloze máte k dispozici kompletní výpis programu s nímž budete v tomto domácím úkolu pracovat. Předpokládá se znalost simulátoru Mips, používaného na cvičení, včetně variant MipsPipeS a MipsPipeXL.

1. Funkce programu (0,5b)

Stručně popište, co uvedený program dělá. Nepopisujte jednotlivé instrukce, ale celkovou funkci (tj. **ne** „program čte a zapisuje do paměti, mezitím počítá součet a rozdíl...“, ale např. „program počítá faktoriál zadaného čísla“).

- Co program dělá?
- Co je vstupem programu? (data, počet)
- Co je výstupem programu? (data, počet)

2. Dynamická predikce skoků (0,5b)

Pro uvedený program určete **počet špatných predikcí** pro dva případy:

- pro BHT (branch history table) s 1-bitovým počítadlem
- pro BHT (branch history table) s 2-bitovým počítadlem

Pro oba případy uveďte skutečné chování (taken/not taken) a predikované chování (taken/not taken) pro vnější a vnitřní smyčku a celkový počet chybných predikcí.

3. Pipeline, statická predikce skoku (4b)

Uvažujte níže uvedený fragment programu (vnitřní smyčka hlavního programu).

```
L2:  add  s4, s0, t2    // s4 - adresa aktualniho prkvu v okne
     lw   t0, 0(s4)     // t0 <- Mem[s4]
     add  t4, t4, t0    // Pripocetni dalsiho prvku k souctu v okne

     addi t2, t2, 0x4   // j += 4; - posun na dalsi prvek v okne
     slt  t3, t2, s3    // t3 = (t2 < s3) ? 1 : 0;
     bne  t3, $0, L2    // Test na ukonceni cyklu
```

Určete počet taktů pro vykonání tohoto kódu pro následující případy (za předpokladu, že přístupy do paměti se realizují přes cache jako cache hit):

1. Bez uvažování možnosti přeposílání (forwarding), avšak s možností zápisu do souboru registrů a čtení zapsaných dat v témže cyklu. Skoková instrukce je ošetřena vyprázdněním pipeline. Adresa větvení je určena ve stupni EX dle simulátoru MipsPipeS.

Instrukce	Takt																					
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	15	16	17	19	20	21	22
`L2: add s4, s0, t2`																						
`lw t0, 0(s4)`																						
`add t4, t4, t0`																						
`addi t2, t2, 0x4`																						
`slt t3, t2, s3`																						
`bne t3, $0, L2`																						

Počet taktů na vnitřní cyklus:

Celkový počet taktů na vykonání kompletního programu:

2. Bez uvažování možnosti přeposílání (forwarding), avšak s možností zápisu do souboru registrů a čtení zapsaných dat v témže cyklu. Skoková instrukce je ošetřena vyprázdněním pipeline. **Rozvrhněte pořadí instrukcí tak, abyste minimalizovali počet cyklů, nutných pro vykonání programu.**

Instrukce	Takt																					
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	15	16	17	19	20	21	22

Počet taktů na vnitřní cyklus:

Celkový počet taktů na vykonání kompletního programu:

3. Výpočet realizujte na procesoru dle simulátoru MipsPipeXL, tzn. mezivýsledky se přeposílají, skoky se predikují jako nerealizované (not taken).

Instrukce	Takt																					
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	15	16	17	19	20	21	22
`L2: add s4, s0, t2`																						
`lw t0, 0(s4)`																						
`add t4, t4, t0`																						
`addi t2, t2, 0x4`																						
`slt t3, t2, s3`																						
`bne t3, $0, L2`																						

Počet taktů na vnitřní cyklus:

Celkový počet taktů na vykonání kompletního programu:

4. Výpočet realizujte na procesoru dle simulátoru MipsPipeXL s tím rozdílem, že **můžeme rozvrhnout pořadí instrukcí** tak, aby se vyplnil jednocyklový delay slot skokové instrukce.

Instrukce	Takt																					
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	15	16	17	19	20	21	22

Počet taktů na vnitřní cyklus:

Celkový počet taktů na vykonání kompletního programu:

Bude velmi přísně postihováno, pokud se student pokusí odevzdat práci, která není jeho vlastním dílem. Takový pokus je důvodem k neudělení zápočtu a nedokončení předmětu a navíc bude předán disciplinární komisi FEL, která rozhodne o dalším postupu (vyloučení ze školy, a další).

Prohlašuji, že jsem domácí úkol vypracoval samostatně.

V: . . . . . Dne: . . . . . Podpis: . . . . . .

4. **Příloha: k**ompletní výpis programu, se kterým budete pracovat

```
#define t0 $8
#define t1 $9
#define t2 $10
#define t3 $11
#define t4 $12


#define s0 $16
#define s1 $17
#define s2 $18
#define s3 $19
#define s4 $20

.globl    pole
.data
.align    2

pole:
.word    5, 3, 4, 1, 15, 8, 9, 2, 10, 6, 11, 1, 6, 9, 12

.text

.globl start
.set noat
.set noreorder
.ent start

start:

  // s0 - bazova adresa zacatku vstupniho pole
  // s1 - pocet prvku vstupniho pole
  // s2 - bazova adresa zacatku vystupniho pole
  // s3 - velikost okna


  la   s0, pole     // Nastaveni adresy pocatku vstupniho pole do registru s0
  addi s1, $0, 15   // Nastaveni poctu prvku vstupniho pole do s1
  sll  s1, s1, 2    // Vypocteni velikosti vstupniho pole (v Bajtech)

  add  s2, s0, s1   // Nastaveni adresy pocatku vystupniho za konec vstupniho pole
  addi s3, $0, 4    // Nastaveni poctu prvku okna
  sll  s3, s3, 2    // Vypocteni velikosti vystupniho okna

  addi t1, $0, 0     // i = 0;
  sub  s1, s1, s3    // Nastaveni horni meze vnejsiho cyklu na velikost...
  addi s1, s1, 0x4   // ...vystupniho pole
L1:
  addi t2, $0, 0     // j = 0; - offset v okne (plati pro vnitrni smycku)
  addi t4, $0, 0     // t4 = 0;  - nulovani souctu v okne
L2:
  add  s4, s0, t2    // s4 - adresa aktualniho prkvu v okne
  lw   t0, 0(s4)     // t0 <- Mem[s4]
  add  t4, t4, t0    // Pripocetni dalsiho prvku

  addi t2, t2, 0x4   // j += 4; - posun na dalsi prvek v okne
  slt  t3, t2, s3    // t3 = (t2 < s3) ? 1 : 0;
  bne  t3, $0, L2    // Test na ukonceni cyklu

  srl  t4, t4, 2     //
  sw   t4, 0(s2)     // Ulozeni vysledku do vystupniho pole: Mem[s2] <- t4
  addi s2, s2, 0x4   // Posun na dalsi prvek ve vystupnim poli
  addi s0, s0, 0x4   // Posun okna ve vstupnim poli

  addi t1, t1, 0x4   // i += 4; - inkrementace pocitadla
  slt  t3, t1, s1    // t3 = (t1 < s1) ? 1 : 0;
  bne  t3, $0, L1


inf_loop:
beq $0,$0, inf_loop

.end start
```