

Prof. L. Thiele

Technische Informatik 1 - HS 2011

Übung 9

Datum: 8.12.2011

1 Instruktionsparallelität – VLIW

Gegeben sei folgendes Programm für den MIPS-Prozessor (ähnlich dem Programm in Übung 7). Es initialisiert ein Array mit den Werten [12, 13, 14, 15] und erhöht diese Werte anschliessend um 1. Das Register `s1` zeigt auf die Basisadresse des Arrays (im folgenden Programm wird die Basisadresse 0 angenommen). Das Register `zero` ist fest mit 0 verdrahtet.

```
main:  ADDI    s1, zero, 0
      ADDI    t7, zero, 12
      SW     t7, 0(s1)
      ADDI    t7, zero, 13
      SW     t7, 4(s1)
      ADDI    t7, zero, 14
      SW     t7, 8(s1)
      ADDI    t7, zero, 15
      SW     t7, 12(s1)
      ADDI    t3, zero, 4
loop:  LW     t1, 0(s1)
      ADDI    t1, t1, 1
      SW     t1, 0(s1)
      ADDI    s1, s1, 4
      SUBI    t3, t3, 1
      BNEZ   t3, loop
```

1.1 Minimieren der Codegrösse

Das Programm soll nun für den MIPS VLIW (Very Long Instruction Word) Instruktionssatz umgeschrieben werden. Dabei soll die Codegrösse minimiert werden. Der Prozessor ist mit 100 MHz getaktet und benutzt keine Pipeline, d.h. der Prozessor beendet eine VLIW Instruktion bevor die nächste ausgeführt wird.

- Das Programm soll für den MIPS VLIW Instruktionssatz umgeschrieben werden. Benutzen Sie dazu die Struktur von Tabelle 1.
- Wie viel Platz benötigt der VLIW Code im Speicher? Vergleichen Sie mit dem Platzbedarf des ursprünglichen MIPS Codes.
- Wie viele Taktzyklen werden pro Instruktion (CPI) im Durchschnitt benötigt? Wie lange dauert die Ausführung des Programms?

Label	ALU oder Verzweigung	Datentransfer	CC
main:			1
			2

Tabelle 1: Vorlage für die Darstellung von MIPS VLIW Instruktionen.

1.2 Minimieren der Ausführungszeit (Zusatzaufgabe)

Das Programm soll wiederum für den MIPS VLIW Instruktionssatz umgeschrieben werden. Es wird der gleiche Prozessor wie in der vorhergehenden Teilaufgabe verwendet. Dabei soll die Ausführungszeit minimiert werden. Tipp: Verwenden Sie Schleifenentfaltung (loop unrolling).

- Passen Sie den Code entsprechend an. Zusätzlich soll die Anzahl der benötigten Register für die Schleifenentfaltung minimiert werden.
- Wie viel Platz beansprucht der Code im Speicher?
- Wie viele Taktzyklen werden pro Instruktion (CPI) im Durchschnitt benötigt? Wie lange dauert die Ausführung des Programms?
- Vergleichen sie die Codegrösse, die Ausführungszeit und den CPI der beiden Versionen (Aufgabe 1.1 und 1.2) des Programmes.

2 Pipelining mit VLIW

Gegeben sei ein VLIW MIPS Prozessor welcher mit einer 5-stufigen Pipeline (IF ID EX MEM WB) arbeitet. Insbesondere wird bei Verzweigungen mit der (für MIPS typischen) statischen Sprungvorhersage not taken gearbeitet. Die Sprungentscheidung ist nach der MEM Stufe bekannt. Es gibt keinen Branch Delay Slot.

Auf dem Prozessor wird die folgende Funktion minVec ausgeführt:

Label	ALU oder Verzweigung	Datentransfer	CC
minVec:	addi a0,a0,4	lw t0,0(a0)	1
	addi a1,a1,4	lw t1,0(a1)	2
	slt t2,t0,t1		3
	bne t2,zero,l1		4
	addi t0,t1,0		5
l1:	addi a3,a3,-1		6
	addi a2,a2,4	sw t0,0(a2)	7
	bne a3,zero,minVec		8
	jr ra		9

Hinweis: `slt t2,t0,t1` \Rightarrow Setze `t2=1` falls `t0<t1`; ansonsten `t2=0`.

Die Funktion `minVec(int* A0, int* A1, int* A2, int n)` überführt die beiden Arrays `A0` und `A1` der Grösse `n` in ein Array `A2` derselben Grösse. Für alle Elemente `i` des neuen Arrays gilt: $A_2[i] = \min(A_0[i], A_1[i])$. Die Startadressen der drei Arrays (`A0`, `A1`, `A2`) werden in den Registern `a0`, `a1` und `a2` übergeben. Die Anzahl Elemente `n` des Arrays steht im Register `a3`.

- Die Pipelining-Architektur unterstützt kein Forwarding. Simulieren Sie das Pipeline-Verhalten der Funktion, wenn diese für die Arrays `A0 = [8]` und `A1 = [5]` ausgeführt wird (d.h. `n = 1`). Vervollständigen Sie dazu die Tabelle 2. Markieren sie gegebenenfalls Stellen wo ein Leeren (Flush) der Pipeline wegen einer falschen Sprungvorhersage nötig ist.

- (b) Die Pipelining-Architektur unterstützt nun Forwarding von EX nach EX sowie von MEM nach EX. Simulieren Sie das Pipeline-Verhalten der Funktion, wenn diese für die Arrays $A_0 = [4]$ und $A_1 = [9]$ ausgeführt wird. Vervollständigen Sie dazu die Tabelle 3. Markieren Sie gegebenenfalls Stellen wo ein Leeren (Flush) der Pipeline nötig ist. Kennzeichnen Sie im Diagramm wo der Forwarding-Mechanismus zum Tragen kommt, indem Sie Pfeile zwischen den entsprechenden Pipeline-Stufen der abhängigen Instruktionen eintragen.
- (c) Die Pipeline-Architektur unterstützt weiterhin Forwarding. Die Funktion wird nun mit den Werten $A_0 = [5, 4, 12, 7, 15]$ und $A_1 = [7, 4, 9, 13, 0]$ ausgeführt. Wie oft muss die Pipeline geleert werden? Wie viele Zyklen gehen dadurch verloren?

3 Diskussion über Instruktionsparallelität

- (a) Um 1990 wurde der *Branch Delay Slot* eingeführt um Ablauf Hazards zu vermeiden. Wieso wird dieses damals erfolgreiche Konzept heute nicht mehr benutzt?
- (b) In der Vorlesung wurden drei verschiedene Ansätze zur Optimierung der Ausführungszeit von Programmen vorgestellt: Superpipelining, VLIW Prozessoren sowie (dynamische) Superskalare Prozessoren. Was sind die Vor- und Nachteile der Ansätze im Bezug auf folgende Eigenschaften?
- Codegröße
 - Compilerbau
 - Logikaufbau
 - Einfluss von Branch Misses
 - Variable Instruktionslänge

Was für eine Architektur wird heutzutage für leistungsstarke CPUs gewählt?

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
minVec: addi a0,a0,4	IF	ID																		
lw t0,0(a0)	IF	ID																		
addi a1,a1,4		IF																		
lw t1,0(a1)		IF																		
slt t2,t0,t1																				
bne t2,zero,l1																				
addi t0,t1,0																				
l1: addi a3,a3,-1																				
addi a2,a2,4																				
sw t0,0(a2)																				
bne a3,zero,minVec																				
jr ra																				

Tabelle 2: Vorlage für VLIW Pipelining ohne forwarding.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
minVec: addi a0,a0,4	IF	ID																		
lw t0,0(a0)	IF	ID																		
addi a1,a1,4		IF																		
lw t1,0(a1)		IF																		
slt t2,t0,t1																				
bne t2,zero,l1																				
addi t0,t1,0																				
l1: addi a3,a3,-1																				
addi a2,a2,4																				
sw t0,0(a2)																				
bne a3,zero,minVec																				
jr ra																				

Tabelle 3: Vorlage für VLIW Pipelining mit forwarding.