Lösningar till tentamen i kursen EDA330/EDA370/EDA440 för D/E/IT

Datorsystemteknik

24/5 2003

1.

Deluppgift	1	X	2
a			2
b		X	
c	1		
d		X	
e			2
f	1		
g		X	
h		X	
i		X	
j		X	
k	1		
l			2

2.

Deluppgift	1	X	2
a			2
b			2
c	1		
d		X	

3.

Deluppgift	Svar																			
a	19%																			
b	Branch on less or equal: `ble $t1, $t2, 1024 # if ($t1 <= $t2) go to L`																			
c	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	I	P	C	O	R	T	E	S	F	G	L	N	T	E	M	H	I	D	B	K

Följande är skisser till lösningar av uppgifterna. I en del fall är alternativa lösningar möjliga.

2.

a. Lägg till tre nop-instruktioner efter lw, sub, slti (pga datakonflikter) och efter beq (pga styrkonflikt). Dvs. totalt **12 st.**

b. De 3 nop-instruktionerna efter sub och slti kan tas bort helt eftersom datakonflikter istället löses med data forwarding. Efter lw kan endast två nop-instruktioner tas bort. Dvs. totalt **8 st.**

c. Extra tid = 1 s =
antal läs/skriv-instruktioner * miss rate * miss penalty cycles * cycle time =
I_{ls} * (1-0,9) * 12 / (600 * 10^6) => I_{ls}=1*600*10^6/((1-0,9)*12) = **500 * 10^6 data-accesser.**

d. 32 bitars adress delas upp i tag, index, och offset. Offset kräver $\log_2$ 64 = 6 bitar. Antal set fås som antal block i cache/associativitet = 256*1024/(64*4). Antalet bitar i index blir då $\log_2$ (256*1024/(64*4)) = $\log_2$ (256*1024) - $\log_2$ 64 - $\log_2$ 4. Tagstorleken blir då 32 – ($\log_2$ (256*1024) - $\log_2$ 64 - $\log_2$ 4) - $\log_2$ 64 = 32 - $\log_2$ (256*1024) + $\log_2$ 4 = **16 bitar**.

3.

a. Under en tid T hämtas T*f/CPI (f = processorfrekvens = 200 MHz) instruktioner. CPI = CPI0 + stoppcykler/instruktionshämtning, där CPI0 är CPI utan inverkan av instruktionshämtningsmissar. Enligt uppgiften är CPI0 = 1,6.
Stoppcykler/instruktionshämtning kan räknas ut som miss rate * miss penalty
Hämtningstid för instruktionsblock om 4 ord = (2 cykler (få bussen) + 1 cykel (skicka adress) + 5 cykler (hämta första två ord) + 2 cykler (överför två ord, hämta nästa två) + 2 cykler (överför två ord)) * 10 ns (busscykeltid) = 120 ns = 120 ns/(5 ns/processorcykel) = 24 processorcykler = miss penalty => stoppcykler/instruktionshämtning = 2%*24 = 0,48.

En instruktionsmiss belastar bussen under alla busscykler beräknade ovan utom två då bussen reserveras. Belastning på bussen (andel av tiden som bussen är reserverad) under en tid T blir:

antal hämtade instruktioner under tiden T * missannolikhet * busstid/miss / T = 200 MHz/(1,6 + stoppcykler/instruktionshämtning) * missannolikhet * busstid/miss =
200 MHz/(1,6 + 0,48) * 2% * 10/(100 MHz) = 2%*10*2/(1,6 + 0,48) = **19%.**

b. Bitmönstren representerar instruktionerna:
`000000 01010 01001 00001 00000 101010`$_2$ `= op = 000000 = 0 (format R), rs=01010=10=$t2, rt=01001=9=$t1, rd=00001=1=$at, shamt=00000=0 och funct=101010=42=slt`
`000100 00001 00000 0000010000000000`$_2$ `= op = 000100 = 4 = beq (format I), rs=00001=1=$at, rt=00000=0=$zero, imm=0000010000000000=1024`

```
slt $at, $t2, $t1       # testa om t1>t2, dvs icke t1<=t2
beq $at, $zero, 1024    # hoppa om så inte fallet, dvs t1<=t2
```

=> `ble $t1, $t2, 1024` dvs **branch on less or equal**

c. Räkna antal händelser som kommer rätt i förhållande till senaste händelse som kom rätt i ordningen, poäng ges i proportion till antal korrekt placerade händelser.

1. I. Minnesläsningen påbörjas.
2. P. Den virtuella adressen delas upp i virtuellt sidnummer och offset.
3. C. Uppslagning sker i TLB, som signalerar en miss.
4. O. Uppslagning sker genom hårdvara i MMU i sidtabellen i primärminnet.
5. R. Det visar sig att sidan inte finns i primärminnet, och en avbrottssignal skickas till CPU.
6. T. CPU börjar exekvera en avbrottshanterare.
7. E. Avbrottshanteraren lagrar undan CPU- och processtillstånd.
8. S. Avbrottshanteraren upptäcker att sidfel uppstått och anropar rutin för hantering av sidfel.
9. F. Kontroll sker att den önskade sidåtkomsten är tillåten.
10. G. En plats i primärminnet för den sökta sidan väljs ut med hjälp av en utbytesalgoritm.
11. L. Överföring av sidan till primärminnet med hjälp av DMA inleds.
12. N. CPU avbryts av en signal att DMA av sidan är klar.

13. T. CPU börjar exekvera en avbrottshanterare.
14. E. Avbrottshanteraren lagrar undan CPU- och processtillstånd.
15. M. Sidtabellen uppdateras.
16. H. Programmet där sidfelet uppstod fortsätter sin exekvering med uppdaterad TLB.
17. I. Minnesläsningen påbörjas.
18. D. Uppslagning sker i TLB, som ger en korrekt översättning till fysisk adress.
19. B. Uppslagning i cache ger en miss.
20. K. Minnesinnehållet returneras från cache till CPU

4.

a. **Ja.**
Typisk miss penalty för sidfel är 10 ms, vilket ger en inverkan på exekveringstiden som är 0,00001% * 10 ms = 1 ns per dataåtkomst.
Typisk missannolikhet för datacache är 0,1%-10%, så vi antar 5%. Miss penalty för cache är typiskt 10-100 cykler, och vi antar 30 cykler. Med en typiskt klockfrekvens för processorn på 1,5 GHz, så blir alltså miss penalty för cache 30 * 1/(1,5 GHz) = 20 ns. Inverkan på exekveringstiden för datacachemissar blir då 5% * 20 ns = 1 ns.
Med rimliga antaganden är det alltså möjligt att komma fram till att inverkan av sidfel och cachemissar vid dataåtkomster är av samma storleksordning.

b. **Se kursboken**. Några möjliga skäl är: Indatavariation, skillnad i vad som finns i cache och primärminne när programmet startas, olika antal avbrott, olika punkter när avbrott sker, olika väntetid på I/O.

5.

a.

instruktion	IF	ID	EX	MEM	WB	Total cykeltid	Antal CPU cykler (2 ns cykeltid)
flyttalsmult.	20ns	1ns	12ns	0	1ns	34ns	10+1+6+1=18
flyttalsdiv.	20ns	1ns	12ns	0	1ns	34ns	10+1+6+1=18
flyttalsadd.	20ns	1ns	6ns	0	1ns	28ns	10+1+3+1=15
lw	20ns	1ns	2ns	20ns	1ns	44ns	10+1+1+10+1=23
sw	20ns	1ns	2ns	20ns	0	43ns	10+1+1+10=22
branch	20ns	1ns	2ns	0	0	23ns	10+1+1=12
övriga	20ns	1ns	2ns	0	1ns	24ns	10+1+1+1=13

Totalt exekveras 10000 + 6000 + 2000 + 10000 + 72000 = 100000 instruktioner.
Cykeltiden bestäms av långsammaste instruktion (lw) vilket ger cykeltiden 44 ns ⇔ 22,7 MHz

Exekveringstid för programmet = 100000 * 44 ns = **4,4 ms**

b. 10% sw-instruktioner = 10000 och 20% lw-instruktioner = 20000, kvar i kategorin övriga instruktioner blir således 72000-10000-20000 = 42000
Totalt krävs 10000*18 + 6000*15 + 2000*18 + 10000*12 + 10000*22 + 20000*23 + 42000*13 CPU cykler = 165200 CPU cykler => 165200*2ns = **3,3 ms**

c. 30% minnes-accesser = 30000, varav 5% missar = 1500 missar
1% missar av alla 100000 instruktioner = 1000 missar
Vid miss krävs 9 busscykler (1 cykel för adress och 8 för överföring av 8 ord) á 1/(100 MHz) = 90 ns, dvs 90 ns/ 2ns = 45 CPU cykler, dvs. 44 extra CPU cykler.
Flyttalsmult/div kräver 5 extra CPU cykler i EX-steget
Flyttalsadd kräver 2 extra CPU cykler i EX-steget.
Utan pipelinekonflikter krävs 1 cykel per instruktion.
Totalt krävs alltså 100000*1 cykler + 10000*5 + 2000*5 + 6000*2 + 2500*44 CPU cykler = 282000 CPU cykler á 2 ns = **0,564 ms**