Lösningar till tentamen i kursen EDA330 för D och EDA370 för E

Datorsystemteknik

16/1 2003

1.

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

2.

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

3.

Deluppgift	Svar
a	5 cykler
b	200 MB/s
c	1,75
d	10%
e	98%
f	5248 bitar

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

2.

a. Antal cykler:
1 cykel för första instruktionen
5*4 cykler för loopen (5 instruktioner exekveras 4 ggr)
3 extra cykler för varje felaktig gissning vid hopp-instruktionen
1 cykel för sista instruktionen
4 cykler då sista instruktionen utförs i sista pipeline-steget

assume not taken => gissar fel 3 ggr => **1+5*4+3*3+1+4 cykler = 35 cykler**

b. 32 bitars adress delas upp i tag, index, och offset. Offset kräver $\log_2 32 = 5$ bitar. Antal set fås som antal block i cache/associativitet = 32*1024/(32*4) = 256. Antalet bitar i index blir då $\log_2 256 = 8$. Tagstorleken blir då 32 – 8 - 5 = **19 bitar**

c. Kostnad för missar i form av extra klockcykler är $1,2 * P_{miss} * T_{penalty}$
b = 4: 1,2 * 0,38 * (4+4/4) = 2,28
b = 16: 1,2 * 0,22 * (4+16/4) = 2,112
b = 64: 1,2 * 0,19 * (4+64/4) = 4,56
Exekveringstiden bestäms av $I*CPI*T_c$. I detta fall ändras dock endast CPI genom de extra klockcyklerna. **Blockstorleken 16 B bör väljas.**

d. Skivminnet rymmer 2*4 ytor * 30000 spår/yta * 500 sektorer/spår * 512 bytes/sektor = **$6,144*10^{10} / 2^{30}$ GB = 57 GB**

e. Utgå från formeln för CPU-tid = $T = I * CPI * T_c = I * CPI / f$ => $I = 45*10^{-3} * 50*10^6 / 1,5 =$ **$1,5*10^6$ instruktioner**

f. Tvåkomplementsform, och talet är negativt. Ta fram det positiva talet genom att ta tvåkomplementet:
Invertera: $01010000001111111011111111111111_2$
Addera 1: $01010000001111111100000000000000_2$
Detta tal kan skrivas $2^{30} + 2^{28} + 2^{21} + 2^{20} + 2^{19} + 2^{18} + 2^{17} + 2^{16} + 2^{15} + 2^{14} =$
$(2^{16} + 2^{14} + 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0) * 2^{14} =$
$(65536 + 16384 + 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1) * 2^{14} = 82175 * 2^{14}$
Det sökta talet är alltså **$-82175 * 2^{14}$**

g. `101011 00010 00000 0000000000000000` = op=`101011`=43=sw (format I), rs=`00010`=2 ($2), rt=`00000`=0 ($0), imm=`0000000000000000`=0.
Instruktionen är **sw $0, 0($2)**

h. För varje block ska lagras data, tag, valid-bit (och dirty-bit om write-back) och 2 bitar vid LRU. Data är block om 32 byte, vilket alltså är 8*32 bitar.
1 GB primärminne => $\log_2 2^{30} = 30$ bitars adresser som delas upp i tag, index, och offset. Offset kräver $\log_2 32 = 5$ bitar. Antal set fås som antal block i cache/associativitet = 32*1024/(32*4) = 256. Antalet bitar i index blir då $\log_2 256 = 8$. Tagstorleken blir då 30 - 8 - 5 bitar = 17 bitar. Totalt per block måste man alltså lagra 8*32+ 17 + 2 + 2 bitar = 277 bitar, och för hela cacheminnet (32*1024/32)*277 bitar = **283648 bitar**

i. För att indexera cachen krävs de minst signifikanta (index+offset) bitarna av adressen. Offset kräver $\log_2 32 = 5$ bitar. Antal set fås som antal block i cache/associativitet = 32*1024/(32*4) = 256. Antalet bitar i index blir då $\log_2 256 = 8$. Dvs. Totalt 5+8=13 bitar.
Den del av de virtuella adresserna som inte behöver översättas, sidoffset, är endast $\log_2 (4*1024)$ bitar = 12 bitar.
13 bitar > 12 bitar => översättning av sidnummer måste ske innan uppslagning i cacheminnet kan göras. Vid träff i TLB och cache bör därför en minnesåtkomst klaras på 3+5 ns = 8 ns. Då denna tid enligt uppgiften definierar en processorklockcykel, kan processorn som bäst köra med 1/(8 ns) = **125 MHz klockfrekvens**.

3.

a. Om ett nytt registervärde kan läsas samtidigt som det skrivs krävs 2 stall-cykler pga databeroendet mellan slt-instruktionen och den därpå följande hoppinstruktionen.
Varje hoppinstruktion orsakar dessutom tre stall-cykler (det tar två cykler tills hoppet beräknats, och ytterligare en cykel tills rätt instruktion hämtats in).
Om $a0>=$a1 kommer endast den första hoppinstruktionen att utföras, så bidraget från hoppkonflikter blir totalt tre stall-cykler. **Totalt krävs alltså 5 extra cykler** orsakade av pipelinekonflikter.

b. Dela upp händelserna i busscykler.

Busscykel 0: Begäran skickas ut om tillgång till bussen.
Busscykel 1: Tillstånd att använda bussen ges direkt eftersom det antas att bussen inte är belastad av annan trafik.
Busscykel 2: Adress till det cacheblock som ska skrivas ut sänds över bussen till minnet.
Busscykel 3 till 12: Åtkomst av blocket görs i minnet. Eftersom minnesåtkomsttiden är 50 ns och busscykeltiden är 5 ns (1/(200 MHz)), så tar detta 10 busscykler.
Busscykel 13: Första ordet skrivs till minnet.
...
Busscykel 16: Fjärde och sista ordet skrivs till minnet. I och med detta släpps bussen för denna transaktion.

En busstransaktion tar alltså 16 busscykler inklusive cykeln då tillstånd att använda bussen ges. Sådana transaktioner kan som mest följa direkt efter varandra. Då varje transaktion innebär att 4 ord = 16 byte data överförs på bussen, och bussen har busscykeltiden 5 ns, fås att den maximala effektiva bandbredden är 16 bytes/(16*5 ns) = **200 MB/s.**

c. User CPU time = $I*CPI*T_c = I*CPI/f$
CPI=f*User CPU time / I
$I = 200*10^6+20*2*10^6$ instruktioner = $240*10^6$ instruktioner => CPI = 100*4,2/240 = **1,75**

d. $CPI = 100*10^6*4,2/I = CPI_0 + I_D/I * m_D$ * miss penalty + m_I * miss penalty =
$1,5 + 10 * I_D/I * m_D + 10 * 0,01$
$I = 200*10^6+20*2*10^6$ instruktioner = $240*10^6$ instruktioner
$I_D = 0,1*200*10^6+0,4*20*2*10^6 = 36*10^6$
$m_D = (4,2/2,4 – 1,5 - 10 * 0,01)/(10 * 36/240)$ **= 10%**

e. Peak bandwidth för skivminnet är
(7200 varv/minut)/(60 s/minut)*512 bytes/sektor*500 sektorer/varv = $3,072*10^7$ bytes/s
Total accesstid = söktid + rotationstid + tid för läsning/skrivning =
10 ms + 0,5 varv/((7200 varv/minut)/(60 s/minut))+8*1024 bytes/($3,072*10^7$ bytes/s) =
14,43 ms
Varav tid som ägnas åt sökning = söktid + rotationstid =
10 ms + 0,5 varv/((7200 varv/minut)/(60 s/minut)) = 14,17 ms
=> **14,17 ms / 14,43 ms = 98% ägnas åt sökning**

f. TLB kan lagra information om 128 sidöversättningar. Den information som behöver lagras är fysiskt sidnummer ($\log_2 2^{30} – \log_2 (4*2^{10})$) bitar = 18 bitar, protection bit, och dirty bit (talar om ifall sidan uppdaterats), valid bit och tag. Eftersom TLB är fullt associativ och uppslagning sker med virtuella sidnumret som adress, så måste hela virtuella sidnumret ($32 – \log_2 (4*2^{10})$) bitar = 20 bitar lagras som tag. Alltså måste 18 + 1 + 1 + 1 + 20 bitar = 41 bitar lagras för varje plats i TLB, och totala antalet bitar i TLB blir 128* 41 bitar = **5248 bitar**.

4. Det finns fyra olika konflikter som kan orsaka pipeline-bubblor och därmed påverka CPI: Hoppkonflikter då hopp tas, datakonflikter då en load-instruktion följs direkt av en instruktion som använder det som läses in (kan ej hanteras med forwarding), cachemiss vid instruktionshämtning, och cachemiss vid läsning eller skrivning av data. Vi behandlar dess i tur och ordning:

Då ett hopp tas har redan en felaktig instruktion hämtats in som måste ersättas av en nop. Detta händer för 20% * 70% = 14% av instruktionerna och ger därmed ett bidrag på 0,14 * 1 = 0,14 till CPI.

Då en minnesläsning direkt följs av en instruktion som använder det som läses in måste stall göras under en cykel tills minnesläsningen genomförts i MEM-steget. En nop får då skjutas in. Detta händer för 20% * 50% = 10% av instruktionerna och ger därmed ett bidrag på 0,1 * 1 = 0,1 till CPI.

Vid en miss i ICACHE måste man vänta tills det sökta cacheblocket lästs in från primärminnet över processor-minnebussen. Den läsningen tar 2 busscykler för väntan på upptagen buss och arbitrering, 1 busscykel för att skicka adressen till minnet, 200 ns * 100 MHz = 20 busscykler för väntan på åtkomst av första ordet, 20 ns * 100 MHz = 2 busscykler för sändning av första ordet och åtkomst av andra ordet, 2 busscykler för sändning av andra ordet och åtkomst av tredje ordet, 2 busscykler för sändning av tredje ordet och åtkomst av fjärde ordet, och 1 busscykel för sändning av fjärde ordet. En busstransaktion tar alltså totalt 2+1+20+2+2+2+1 = 30 busscykler. Eftersom bussen och processorn har samma klockfrekvens betyder det att miss penalty blir 30 processorcykler. Detta inträffar för 1% (träffsannolikhet för ICACHE är 99%) av instruktionerna, och ger därmed ett bidrag på 0,01 * 30 = 0,3 till CPI.

Vid en miss i DCACHE måste en transaktion göras på samma sätt som vid missar i ICACHE för att hämta det sökta blocket, vilket alltså tar 30 cykler. Det kan dock ibland krävas en extra transaktion på grund av write-back för att skriva ett block som kastas ut till primärminnet. Det inträffar vid 50% av missarna, och en miss i DCACHE kräver därför i genomsnitt 30 + 0,5*30 = 45 cykler extra. Detta sker alltså för de (20% + 10%)*(1-90%) = 3% av instruktionerna som är läsningar eller skrivningar som orsakar en miss, och ger därmed ett bidrag på 0,03 * 45 = 1,35 till CPI.

Då CPI för en pipeline utan konflikter är 1 (vi bortser från effekter av uppstart), blir totala CPI i detta fall alltså 1 + 0,14 + 0,1 + 0,3 + 1,35 = **2,89**