Lösningar till tentamen i kursen EDA440 för IT

Datorsystemteknik

12/3 2003

1.

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

2.

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

3.

Deluppgift	Svar
a	`lui $8, 1; addi $8, $8, 4`
b	**MEM.MemRead & MEM.DestReg ≠ 0 & (MEM.DestReg=EX.rs | (MEM.DestReg=EX.rt & EX.ALUSrc=1 & EX.MemWrite=0))**
c	**4736 bitar**.
d	**60% ägnas åt sökning**
e	**15.6 sekunder**
f	**$-321 * 2^{22}$**

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

2.

a. Antag lw hämtas i cykel 0. Stalling innebär tre extra cykler fördröjning efter lw, sub, slti, beq (nop eller "bubbla" läggs in av hårdvaran efter ID-steget). Det går alltså 4 cykler per instruktion fram tills add-instruktionen hämtas i cykel 0+4*4=16. Det tar sedan ytterligare 5 cykler (en för varje pipelinesteg) tills add lämnat WB. Totala antalet cykler blir alltså 16+5 cykler = **21 cykler**.

b. Med data forwarding från MEM och WB till EX kan alla datakonflikter undvikas helt utom konflikten mellan lw och sub som fortfarande kräver en cykel stalling (två stall-cykler försvinner dock). Styrkonflikten i samband med beq kan inte lösas på detta sätt. Totalt försvinner alltså 2+3+3=8 stallcykler, så totala antalet cykler minskar med **8** till 13.

c. 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 = 64*1024/(64*4) = 256. Antalet bitar i index blir då $\log_2 256 = 8$. Tagstorleken blir då 32 – 8 - 6 = **18 bitar**

d. Sidstorlek 16 KB => sidoffset kräver $\log_2 (16*1024)$ bitar = 14 bitar. 32-bitars adresser => virtuella sidnummer kräver (32 – 14) bitar = 18 bitar, vilket motsvarar 2^{18} sidor. 1GB fysiskt minne => fysiska sidnummer kräver $\log_2 (1024*1024*1024) – \log_2 (16*1024)$ bitar = 30 – 14= 16 bitar. I sidtabellen ska för varje virtuell sida kunna lagras fysiskt sidnummer, valid bit, dirty bit, protection bit, och två bitar för utbytesalgoritmen, dvs totalt (16 +1 + 1 + 1 + 2) bitar= 21 bitar. Eftersom varje sidtabell-entry måste vara ett helt antal bytes krävs 24 bitar = 3 bytes. Varje sidtabell kräver därmed 2^{18} sidor * 3 B/sida = 256 KB * 3 = **768 KB**.

e. Bussen utnyttjas effektivast vid överföring av 8 dataord åt gången. 8 ord överförs då på (8+1) bussyckler, och det går 166 * 10^6 busscykler/s (166 MHz). Maximal databandbredd för bussen är alltså 8 ord/((8+1) cykler) * 32/8 B/ord * 166 * 10^6 busscykler/s = **590 MB/s**

f. Processorn utför $600*10^6/1{,}5$ instruktioner/s = $400*10^6$ instruktioner/s. Dela upp bandbreddskravet i bidrag från läsningar i instruktionscache, läsningar i datacache, och skrivningar i datacache.
Läsning i instruktionscache: $400*10^6$ läsningar/s * (1-0,999) missar/läsning = $0{,}4*10^6$ missar/s, (9 cykler/miss)/(166 * 10^6 busscykler/s) = $5{,}42*10^{-8}$ s/miss => $0{,}4*10^6$ missar/s * $5{,}42*10^{-8}$ s/miss = 2,17 % av tiden.
Läsning i datacache: 0,2 läsningar/instruktion * $400*10^6$ instruktioner/s * (1-0,98) missar/läsning = $1{,}6*10^6$ missar/s, (9 cykler/miss)/(166 * 10^6 busscykler/s) = $5{,}42*10^{-8}$ s/miss => $1{,}6*10^6$ missar/s * $5{,}42*10^{-8}$ s/miss = 8,67 % av tiden.
Skrivning i datacache: 0,1 skrivningar/instruktion * $400*10^6$ instruktioner/s, write-through innebär 2 busscykler för varje skrivning, 0,1 * $400*10^6$ skrivningar/s * (2 busscykler/skrivning)/(166 * 10^6 busscykler/s) = 48,19% av tiden.
Totalt för cacheåtkomster 2,17%+8,67%+48,19% = **59,03%**

g. User CPU time = I*CPI*Tc = I*CPI/f
CPI=f*User CPU time / I
f=$500*10^6$ Hz
User CPU time = 8,4 s
I = $2*10^8$ + 100 * $2*10^7$ instruktioner = 2,2 * 10^9 instruktioner => CPI = $500*10^6$ *8,4/($2{,}2*10^9$) = **1,91**

h. Utgå från formeln för CPU-tid = I * CPI * Tc = I * CPI / f
Extra tid t = 11-9 s = 2 s =
antal läs/skriv-instruktioner * miss rate * miss penalty cycles * cycle time =
I_{ls} * (1-0,95) * 8 / ($250*10^6$)
$I_{ls}=2*250*10^6/((1-0{,}95)*8) = 1{,}25*10^9$ läs/skriv-instrukltioner
Totala antalet instruktioner = CPU-tid * f / CPI = $9*250*10^6$ / 1,0 = $2{,}25*10^9$ instruktioer
Andel dataåtkomster = $1{,}25*10^9$ / $2{,}25*10^9$ = **56%**

i. $27_{10} = 11011_2 = 1.1011_2 * 2^4$. Sign = 0 (positivt). Exponent (8 bitar) = $4_{10}+127_{10}= 131_{10} = 10000011_2$. Significand (32-1-8=23 bitar) = 10110000000000000000000. Hela talet i IEEE 754 single precision = sign, exponent, significand = $\texttt{01000001110110000000000000000000}_2$.

3.

a. 65540 = 65536+4, laddas med
`lui $8, 1; addi $8, $8, 4`

b. Detta uppstår om det finns en minnesläsning i MEM som ska skriva till något av källregistren för instruktionen i EX. Observera att vi utgår ifrån att en minnesläsning implicerar en registerskrivning varför vi inte behöver kolla MEM.RegWrite. Det är dock inte fel att göra det. Det är också nödvändigt att kontrollera att källregistren används av instruktionen i EX. I den aktuella pipelinen används alltid rs, men rt används inte om immediate används istället (ALUSrc=1) och det inte är en store-instruktion (MemWrite=1).
Ett fall av nedanstående räcker som svar:
MEM.MemRead & MEM.DestReg ≠ 0 & (MEM.DestReg=EX.rs | (MEM.DestReg=EX.rt & EX.ALUSrc=1 & EX.MemWrite=0))

c. TLB kan lagra information om 128 sidöversättningar. Den information som behöver lagras är fysiskt sidnummer ($\log_2 2^{30} – \log_2 (16*2^{10})$) bitar = 16 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 ($32 – \log_2 (16*2^{10})$) bitar = 18 bitar lagras som tag. Alltså måste 16 + 1 + 1 + 1 + 18 bitar = 37 bitar lagras för varje plats i TLB, och totala antalet bitar i TLB blir 128* 37 bitar = **4736 bitar**.

d. Peak bandwidth för skivminnet är
(6000 varv/minut)/(60 s/minut)*512 bytes/sektor*200 sektorer/varv = $10,24*10^6$ bytes/s
Total accesstid = söktid + rotationstid + tid för läsning/skrivning = (10 ms + 0,5 varv/((6000 varv/minut)/(60 s/minut))+1024 bytes/($10,24*10^6$ bytes/s))*(1-0,99) +1024 bytes/($10,24*10^6$ bytes/s)*0,99 = 0,151 + 0,099 ms = 0,25 ms
Varav tid som ägnas åt sökning = (10 ms + 0,5 varv/((6000 varv/minut)/(60 s/minut))*(1-0,99) = 0,15 ms =>
(tid som ägnas åt sökning) / (Total accesstid) = 0,15 / 0,25 = 0,6 = **60% ägnas åt sökning**

e. CPI=CPIperfect+MemStall.
Först behöver vi ta reda på CPIperfect för den gamla processorn.
CPI=800MHz*20sek/10e9=1.6 clocks per instr.
För att räkna ut MemStall för den gamla processorn behöver vi veta hur stor andel av alla accesser som missar i de olika cachenivåerna.
Missar i L1 per instr.=0.01+0.05*0.25 = 0.0225
Missar i L2 per instr.=(0.01+0.05*0.25)*(1-0.8) = 0.0045
MemStall=(0.01+0.05*0.25)*30e-9*800e6+(0.01+0.05*0.25)*(1-0.8)*
*60e-9*800e6=0.54 + 0.216 = 0.756 clocks per instr
CPIperfect=1.6-0.756 = 0.844 clocks per instr
För att få CPI för den nya processorkonfigurationen behöver vi räkna ut MemStall:
MemStall_256kB=(0.01+0.05*0.25)*5e-9*1000e6+(0.01+0.05*0.25)*
*(1-0.55)*60e-9*1000e6=0.72 clocks per instr
CPI_256kB = CPIperfect+ 0.72 = 1.564
T=I*CPI*Tc=10e9*1.564*1/1000e6= **15.64 sekunder**.

f. Tvåkomplementsform, och talet är negativt. Ta fram det positiva talet genom att ta tvåkomplementet:
Invertera: $\texttt{01010000001111111111111111111111}_2$
Addera 1: $\texttt{01010000001000000000000000000000}_2$
Detta tal kan skrivas $2^{30} + 2^{28} + 2^{22} = (2^8 + 2^6 + 2^0) * 2^{22} = (256 + 64 + 1) * 2^{22} = 321 * 2^{22}$
Det sökta talet är alltså $\mathbf{-321 * 2^{22}}$

4.

a. Dela upp händelserna i busscykler, och översätt sedan till processorcykler. Det går två processorcykler på en busscykel. Eftersom write-back tillämpas måste man också ta hänsyn till att en tillbakaskrivning av ett block kan krävas innan det sökta blocket läses in. Observera att var och en av dessa två blocköverföringar måste hanteras som en separat transaktion.

Busscykel 0: Cachemissen detekteras och begäran skickas ut om tillgång till bussen. I värsta fall inträffar cachemissen i den första av de två processorcyklerna som utgör denna busscykel, varför ett bidrag på en processorcykel fås till totala väntetiden. Påföljande busscykler bidrar alla med två processorcykler.
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 3-17: Åtkomst av blocket görs i minnet. Eftersom minnesåtkomsttiden är 75 ns och busscykeltiden är 5 ns (1/(200 MHz)), så tar detta 15 busscykler.
Busscykel 18: Första ordet skrivs till minnet.
Busscykel 19: Andra ordet skrivs till minnet.
Busscykel 20: Tredje ordet skrivs till minnet.
Busscykel 21: Fjärde ordet skrivs till minnet. I och med detta släpps bussen för denna transaktion. Eftersom det sökta cacheblocket också ska läsas in i ytterligare en transaktion så begärs dock omedelbart tillstånd för ytterligare en transaktion.
Busscykel 22: Tillstånd att använda bussen ges direkt eftersom det antas att bussen inte är belastad av annan trafik.
Busscykel 23: Adress till det cacheblock som ska skrivas ut sänds över bussen till minnet.
Busscykel 24-38: Åtkomst av blocket görs i minnet. Eftersom minnesåtkomsttiden är 75 ns och busscykeltiden är 5 ns (1/(200 MHz)), så tar detta 15 busscykler.
Busscykel 39: Första ordet skrivs till cacheminnet.
Busscykel 40: Andra ordet skrivs till cacheminnet.
Busscykel 41: Tredje ordet skrivs till cacheminnet.
Busscykel 42: Fjärde ordet skrivs till cacheminnet. I och med detta släpps bussen för denna transaktion. När denna cykel är slut så är också cachemissen hanterad.

Totalt tar det alltså som mest 42 busscykler = 84 processorcykler + 1 processorcykel (från busscykel 0) = **85 processorcykler** från det att en cachemiss upptäcks till det sökta blocket laddats i cacheminnet.

b. Av resonemanget i förra deluppgiften framgår att en busstransaktion tar 21 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 16B/(21*5 ns) = **152 MB/s**.

c. **Se avsnitt 8.4 i kursboken.**