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

Datorsystemteknik

24/8 2002

Följande är skisser till lösningar av uppgifterna. Full poäng på en uppgift kräver i de flesta fall fylligare motivering. I en del fall är alternativa lösningar möjliga.

1.

a. Figuren visar ett exempel på datakonflikt hos register $t2 mellan instruktionerna `lw $t2, 200($t5)` och `add $t3, $t1, $t2`.

b. Konflikten kan lösas antingen med data forwarding (**se avsnitt 6.5 i kursboken**), pipeline stall eller med hjälp av kompilatorn (kasta om instruktionerna eller stoppa in nop instruktioner).

c. 4 instruktioner för att fylla pipeline + 2 instruktioner innan loop L1 + (5 + 2 + (10 för subrutin L3) + (1 + 2 + (10 för subrutin L3) + 2)*40 för inre loop L2 + 2)*10 för loop L1 + 3 = 6199 instruktioner (=I). Med formeln för CPU tid = I * CPI * T_C där antal klockcykler per instruktion CPI = 1 och en klockcykeltid T_C = 1 ns (1 GHz klockfrekvens) fås CPU-tid = **6199 ns**.

d. Ersätt `nop` i delay branch luckorna med andra instruktioner där så är möjligt. Ersätt additioner med multiplikation med 4 (skifta 2 steg). Sätt $a2 till 40 i början av loop L1 ($t1 alltid 10). Flytta initiering av $t0 utanför loop L1. T ex:

```
        addi    $a1,    $zero, 10
        addi    $t0,    $zero, 1
L1:     addi    $a2,    $zero,  40
        jal     L3
        sub     $a1,    $a1,    $t0
L2:     jal     L3
        sub     $a2,    $a2,    $t0
        bne     $a2,    $zero, L2
        sw      $v0,    0($a3)
        bne     $a1,    $zero, L1
        nop
        j       L4
        sw      $v0,    4($a3)
L3:     sw      $t1,    0($sp)
        lw      $t1,    0($a1)
        add     $v0,    $v0,    $t1
        lw      $t1,    0($a2)
        add     $v0,    $v0,    $t1
        sll     $v0,    2
        jr      $ra
        lw      $t1,    0($sp)
L4:
```

2.

a. **Se avsnitt 5.6 i kursboken.** I/O device request, invoke OS from user program arithmetic overflow, undefined instruction, hardware malfunction.

b. **Se avsnitt 6.7 i kursboken**.

c. **Se avsnitt A.7 i kursboken, laboration 1 och specialuppgift**. Förhindra ytterligare avbrott genom att maska bort dem. Spara undan EPC och Cause. Avgör orsak till avbrottet. Spara eventuellt undan tillstånd för användarprogrammet. Tillåt ytterligare avbrott igen.

3.

a. 1 block = 64 ord = 2^6 ord = 128 byte = **1024 bitar**. Av adressen krävs alltså 6 bitar i block-offset. Cacheminnet kan lagra 4 K ord = 64 block. 4-vägs associativitet innebär 4 ord/set, dvs. totalt 16 set i cacheminnet. Av adressen krävs därför 4 indexbitar för att hitta i vilket set ett block kan ligga. Det finns maximalt 1 M ord primärminne så de fysiska adresserna som används av cachen måste vara 20 bitar breda (2^{20} = 1 M ord). För varje block i cacheminnet måste därför 20 - 6 - 4 = **10 bitar tag** lagras. Dessutom krävs **2 bitar för LRU** och **1 valid bit** per block. Det krävs alltså 1024 + 10 + 2 + 1 = 1037 bitar per block. Med 64 block blir totala antalet bitar cachen måste lagra 64*1037 = **66368 bitar**.

b. Från access av adress 0 till adress 4095 fylls cachen med 64 ord åt gången vilket innebär 64 missar. Från access av adress 4096 till 4351 sker ytterligare 4 missar. Eftersom LRU används är det blocken med innehållet i de lägsta adresserna som ersätts. När man sedan börjar accessa adress 0 och framåt igen sker därför först 4 missar och de block som ersätts är nu de med innehållet i de numera lägsta (tidigare näst lägsta) adresserna. Det betyder att det fås ytterligare 4 missar sammanlagt fem gånger per varv. Det hela upprepas ytterligare 8 gånger. Alltså fås totalt (64 + 4 + 5*4*9) = 248 missar. Resterande 4352*10-248 = 43272 accesser är träffar.
Vid miss tar det 9 busscykler för 8 ord (1 cykel för adress och 8 cykler för de 8 orden). Detta upprepas 8 gånger (= 64 ord) => 9*8 busscykler = 72 busscykler => 720 ns vid 100 MHz klockfrekvens.
Vid träff accessas cachen med klockfrekvensen 400 MHz => 2,5 ns per access.
Den totala accesstiden vid användning av cache blir således 248*720 + 43272*2,5 ns = 286740 ns.
Den totala accesstiden då cachen inte används blir 4352 accesser per varv * 10 varv * 2 busscykler per access *10 ns/busscykel = 870400 ns
Dvs. 870400/286740 = **3,0 ggr snabbare med cachen**.

c. Direct mapped cache innebär att inga LRU bitar behövs. Fortfarande krävs 6 bitars block-offset (1 block = 64 ord = 2^6 ord = 128 byte = **1024 bitar**). Cacheminnet kan lagra 4 K ord = 64 block. = 2^6 block. Av adressen krävs därför 6 indexbitar. Med 20 bitars adresser måste därför 20 – 6 – 6 = **8 bitar tag** lagras. Dessutom krävs fortfarande **1 valid bit** per block. Det krävs alltså 1024 + 8 + 1 = 1033 bitar per block. Med 64 block blir totala antalet bitar cachen måste lagra 64*1033 = **66112 bitar**.
Från access av adress 0 till adress 4095 fylls cachen med 64 ord åt gången vilket innebär 64 missar. Från access av adress 4096 till 4351 sker ytterligare 4 missar. Det blir blocken med innehållet i de lägsta adresserna som ersätts. När man sedan börjar accessa adress 0 och framåt igen sker därför först 4 missar fram till access av adress 4096 då ytterligare 4 missar fås. Det hela upprepas ytterligare 8 gånger. Alltså fås: (64 + 4 + (4+4)*9) = 140 missar och således 4352*10-140 = 43380 träffar.
Samma accesstid vid träff och miss som i uppg. b gör att den totala accesstiden vid användning av cache blir 140*720 + 43380*2,5 ns = 209250 ns.
Den totala accesstiden utan cache är enligt uppg. b 870400 ns. Dvs. 870400/209250 = **4,2 ggr snabbare med cachen**.

d. Bussen utnyttjas effektivast vid överföring av 8 dataord åt gången. Åtta ord överförs då på nio bussyckler, och det går $100 * 10^6$ busscykler/s (100 MHz). Maximal databandbredd för bussen är alltså 8 ord/9 cykler * 2 B/ord * $100 * 10^6$ busscykler/s = **177,8 MB/s**.

e. Processorn utför 400 MHz/1,0 = $400 * 10^6$ instruktioner/s. Dela upp bandbreddskravet i bidrag från läsningar i instruktionscache, läsningar i datacache, och skrivningar i datacache.
Läsning i instruktionscache: 100% träffsannolikhet => 0 MB/s.
Läsning i datacache: 0,25 läsningar/instruktion * $400 * 10^6$ instruktioner/s * 0,001 missar/läsning * 128 B/miss = 12,8 MB/s.
Skrivning i datacache: 0,05 skrivningar/instruktion * $400 * 10^6$ instruktioner/s = $2 * 10^7$ skrivningar/s, write-through innebär 1 ord = 2 B på bussen för varje skrivning, $2 * 10^7$ skrivningar/s * 2 B/skrivning = 40 MB/s.
Total databandbredd för cacheåtkomster 0 + 12,8 + 40 MB/s = **52,8 MB/s**.

f. Överföring av 128 KB tar 0,2 ms + 7 ms + 128 KB/(10 MB/s) = 20,3 ms. På 20,3 ms överförs 2*128 KB, vilket leder till en effektiv databandbredd på 256 KB/20,3 ms = **12,9 MB/s**.

g. Här måste beaktas hur stor andel av tiden som systembussen är upptagen av trafik till och från cacheminnen. Dela som tidigare upp i tre bidrag. Vid miss i cache skall 64 ord överföras i block om 8 ord som tar 9 busscykler/block.
Läsningar i instruktionscache tar därför 8*9 busscykler/miss * $1/(100 * 10^6)$ s/busscykel = 0,72 µs/ miss, missar i instruktionscachen inträffar i stort sett aldrig (100% träffsannolikhet) och upptar därför 0% av tiden på bussen.
Läsningar i datacache tar 8*9 busscykler/miss * $1/(100 * 10^6)$ s/busscykel = 0,72 µs/ miss, och det blir $400 * 10^6 * 0,001 * 0,25$ missar/s, och läsningarna tar därför upp $0,72 * 10^{-6} * 400 * 10^6 * 0,001 * 0,25$ = 7,2% av tiden.
Skrivningar i datacache tar 2 busscykler * $1/(100 * 10^6)$ s/busscykel = 0,02 µs, och tar därför upp $0,02 * 10^{-6} * 400 * 10^6 * 0,05$ = 40% av tiden.
Totalt är alltså bussen upptagen med cacheminnestrafik 47,2% av tiden, och är därmed ledig för I/O 52,8% av tiden. Vid I/O utnyttjas bussens maximala databandbredd 177,8 MB/s. I/O kan därför totalt tillåtas en databandbredd på 0,528 * 177,8 MB/s = 93,9 MB/s. Det innebär att bakplansbussen kan belastas med maximalt 93,9/12,9 = **7 I/O-bussar**

4.

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