Lösningar till tentamen i kursen EDA330

Datorsystemteknik D

14/1 2000

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

1. 02801200_{16} = $00000010100000000001001000000000_2$

   a. op = 000000 = 0 => kolla funct = 000000 = 0 = sll => kolla shamt = 01000 = 8
   rs = 10100 = 20 = $s4
   rt = 00000 = 0 = $zero
   rd = 00010 = 2 = $v0
   Instruktionen är sll $v0, $s4, 8, vilken skiftar värdet i register $s4 logiskt åtta steg åt vänster och lägger resultatet i register $v0.

   b. sign = 0 (positivt)
   exponent field = $00000101_2 = 5_{10}$ = exponent +127 => exponent = -122
   significand field = $00000000001001000000000_2$ => significand = 1.00000000001001_2
   Flyttalet är $1.00000000001001_2 * 2^{-122} = 100000000001001_2 * 2^{-136} = (1+8+2^{14})*2^{-136}$ = **$16393 * 2^{-136}$**

   c. Tvåkomplementsform, och talet är positivt. Talet är $2^9 + 2^{12} + 2^{23} + 2^{25} = (2^0 + 2^3 + 2^{14} + 2^{16}) * 2^9 = (1 + 8 + 16384 + 65536) * 2^9$ = **$81929 * 2^9$**

2.

   a. En virtuell adress kräver 28 bitar. Sidstorleken 16 KB innebär att 14 bitar krävs för sidoffset, varför virtuella sidnummer består av 28-14 = 14 bitar. Härav fås att varje process har upp till 16 Ksidor. De fysiska adresserna kräver också 28 bitar (2^{28} B = 256 MB), varför fysiska sidnummer också består av 14 bitar. För varje virtuell sida behöver sidtabellen också lagra en valid bit (finns sidan i primärminnet?), en dirty bit (för write back), 4 protection bits (enligt uppgiften), och 8 replacement bits (tidsstämplar). För varje virtuell sida krävs alltså 14+1+1+4+8 = 28 bitar i sidtabellen. Eftersom varje sidöversättning ska ta upp ett jämnt antal byte krävs alltså 4 byte per virtuell sida. Med 64 processer och 16 Ksidor/process och 4 B/sida krävs totalt $64 \times 16K \times 4$ = **4 MB** för sidtabellerna.

   b. **Se kursboken**.

3.

a. Exekveringstid förutom I/O-väntetid = CPU-tid = I * CPI * T_c. Här påverkas ej I men CPI och T_c. Ny CPI blir 0,999*1,4 + 0,001*2 = 1,4006, dvs CPI ökar med 0,0006/1,4 = 0,04% (alltså försumbart). Ny Tc blir 1/(1,12*f) = 1/1,11 * gammal Tc. Exekveringstiden förändras alltså med en faktor 1,0004 * 0,9 = 0,9, dvs **exekveringstiden minskar med 10%**.

b. **Se kursboken**. T.ex. metod 3:
1. Initiera Produktregister (64 bitar) med multiplikator i minst signifikanta del
2. Om minst signifikanta bit i Produkt är 0, hoppa till steg 4.
3. Addera multiplikand till mest signifikanta del av Produkt.
4. Skifta Produkt logiskt ett steg åt höger.
5. Om inte alla bitar i multiplikator testats, hoppa till steg 2.

c.

```
                                # Initiera produktreg
     mtlo   $r1
     mthi   $zero
     addi   $t8, $zero, 32      # Initiera biträknare
     lui    $t9, 0x8000         # Konstant för skiftning
                                # Testa LSB i produktreg
L1:  mflo   $at
     andi   $at, $at, 1
     beq    $at, $zero, L2
                                # Om inte 1,
                                # addera multiplikand
     mfhi   $at
     addu   $at, $at, $r2
     mthi   $at
                                # Skifta LSB av produkt
L2:  mflo   $at
     srl    $at, $at, 1
     mtlo   $at
                                # Kolla om bit ska skiftas
                                # över från MSB
     mfhi   $at
     andi   $at, $at, 1
     beq    $at, $zero, L3
                                # I så fall, addera bit
                                # till LSB av produkt
     mflo   $at
     addu   $at, $at, $t9
     mtlo   $at
                                # Skifta MSB av produkt
L3:  mfhi   $at
     srl    $at, $at, 1
     mthi   $at

     addi   $t8, $t8, -1        # Räkna ner biträknare
     bne    $t8, $zero, L1      # Repetera om bitar kvar
```

I denna lösning utnyttjas att operandvärdena aldrig är större än 80000000_{16}, annars så måste en eventuell carry vid additionen av multiplikanden tas hänsyn till. **(OBS! Andra godkända lösningar är möjliga.)**

d. I detta fall påverkas I och T_c, men ej CPI. Ändring av T_c beräknades i delupp-

gift a. I genomsnitt tar multiplikationsrutinen i föregående deluppgift 4+17*32 = 548 instruktioner (vi antar att hoppen till L2 och L3 tas i genomsnitt varannan gång eftersom talen som multipliceras kan vara godtyckliga). Nytt I blir då 0,999*I + 0,001*548*I = 1,547*I. Totalt ändras alltså exekveringstiden med en faktor 1,547 * 0,9 = 1,39. **Exekveringstiden ökar med 39%, medan kostnaden minskar med 8% (OBS! Detta svar beror på svaret på föregående deluppgift).**

e. **Lösningen ska innehålla följande delsteg:**
1. Spara undan register (i detta fall åtminstone $at, $t8, $t9, $a0, $a1).
2. Spara undan återhoppsadress från EPC, t.ex. i $k0.
3. Kontrollera bitarna 2-6 i Cause. Om ej 10, hoppa till rutin för andra avbrottstyper.
4. Kontrollera om instruktionen som orsakade exception var en multu (slå upp instruktionen med hjälp av återhoppsadressen). Om inte multu, hoppa till annan hanterare.
5. Flytta Oper1 till $ao och Oper2 till $a1, och anropa subrutinen MULTU.
6. Återställ undansparade register.
7. Hoppa tillbaka från avbrottshanterare.

4.

a. **200, 0, 204, 100, 208, 212, 216, 100, 220, 224, 200, 4, 204, 104, 208, 212, 216, 104, 220, 224, 200, 8, 204, 108, 208, 212, 216, 108, 220, 224, 200, 12, 204, 112, 208, 212, 216, 112, 220, 224, 200, 16, 204, 116, 208, 212, 216, 116, 220, 224, 228.**

b. **Block-offset = byte-adress mod 8 (2 ord/block = 8 byte/block)**
Block = byte-adress/8
Index = block mod 4 ((16 ord/(2 ord/block))/(2 block/set) = 4 set)
Tag = block/4.

c.

Set/block					0		1		2		3	
Byte	Block	Index	Tag	Hit	0	1	0	1	0	1	0	1
200	25	1	6	n			**6**					
0	0	0	0	n	**0**		6					
204	25	1	6	y	0		**6**					
100	12	0	3	n	0	**3**	6					
208	26	2	6	n	0	3	6		**6**			
212	26	2	6	y	0	3	6		**6**			
216	27	3	6	n	0	3	6		6		**6**	
100	12	0	3	y	0	**3**	6		6		6	
220	27	3	6	y	0	3	6		6		**6**	
224	28	0	7	n	**7**	3	6		6		6	

Hit rate = 4/10 = 40%.

d.

Set/block					0		1		2		3	
Byte	Block	Index	Tag	Hit	0	1	0	1	0	1	0	1
200	25	1	6	y	7	3	**6**		6		6	
4	0	0	0	n	7	**0**	6					
204	25	1	6	y	7	0	**6**		6		6	
104	13	1	3	n	7	0	6	**3**	6		6	
208	26	2	6	y	7	0	6	3	**6**		6	
212	26	2	6	y	7	0	6	3	**6**		6	
216	27	3	6	y	7	0	6	3	6		**6**	
104	13	1	3	y	7	0	6	**3**	6		6	
220	27	3	6	y	7	0	6	3	6		**6**	
224	28	0	7	y	**7**	0	6	3	6		6	
200	25	1	6	y	7	0	**6**	3	6		6	
8	1	1	0	n	7	0	6	**0**	6		6	
204	25	1	6	y	7	0	**6**	0	6		6	
108	13	1	3	n	7	0	6	**3**	6		6	
208	26	2	6	y	7	0	6	3	**6**		6	
212	26	2	6	y	7	0	6	3	**6**		6	
216	27	3	6	y	7	0	6	3	6		**6**	
108	13	1	3	y	7	0	6	**3**	6		6	
220	27	3	6	y	7	0	6	3	6		**6**	
224	28	0	7	y	**7**	0	6	3	6		6	

Hit rate = 20/30 = 67%.

e. **Ordningen på minnereferenserna skulle troligen ändras, vilket i sin tur i princip kan påverka hit rate eftersom det är möjligt att block kastas ut i en annan ordning än tidigare**.

f. Vi sätter upp en tabell över vilka instruktioner som befinner sig i respektive pipelinesteg varje cykel genom att referera till deras minnesadresser. Adresser som håller på att slås upp markeras med fetstil.

Cyk	IF	ID	EX	MEM	WB	Kommentar
1	**200**	**-**	**-**	**-**	**-**	Miss
7	**204**	200	-	-	-	Träff
8	**208**	204	200	-	-	Miss
9	**208**	-	204	200/0	-	
14	208	-	204	200/**0**	-	Data först
20	-	208	-	204/**100**	200	Data först
21	-	208	-	204/**100**	-	
26	**212**	-	208	-	204	
27	**216**	212	-	208	-	
28	**216**	-	212	-	208	
29	**216**	-	-	212	-	
30	**216**	-	-	-	212	
31	**216**	-	-	-	-	
33	**220**	216	-	-	-	
34	**224**	220	216	-	-	
35	**224**	-	220	216/100	-	
40	224	-	220	216/**100**	-	
41	**228**	224	-	220	216	
42	**200**	228	224	-	220	
:						

Sekvensen av byte-adresser blir **200, 204, 208, 0, 100, 212, 216, 220, 224, 100**.

5.

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