Lösningar till tentamen i kursen EDA330

Datorsystemteknik

22/8 1998

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.

a. $5{,}25 = 21/4 = 21 * 2^{-2} = 10101_2 * 2^{-2} = (-1)^0 * 1{,}0101_2 * 2^{129-127}$
Sign = 0
Significand field = $01010000000000000000000_2$
Exponent field = $129 = 10000001_2$
Den binära representationen är $\mathbf{01000000101010000000000000000000_2}$

b. Sign = 1 (negativt)
Exponent field = $01011000_2 = 88_{10}$ = exponent +127 => exponent = -39
Significand field = $10000000000000000000000_2$ => significand = 1.1_2
Flyttalet är $-1.1_2 * 2^{-39} = -11_2 * 2^{-40} = \mathbf{-3 * 2^{-40}}$

c. **Se kursboken.**
1. Dividera talet som har minst exponent med 2 (=högerskift, glöm ej inledande ettan) och inkrementera exponenten tills båda talen har samma exponent.
2. Addera signifikanderna.
3. Normalisera resultatet genom upprepade höger- eller vänsterskift av signifikanden kombinerat med motsvarande ökning/minskning av exponenten. Kontrollera om overflow/underflow uppstår.
4. Avrunda signifikanden till rätt antal bitar. Upprepa från steg 3 om resultatet efter avrundning inte är normaliserat.

2.

a. **Programmet adderar en fem-elements vektor av 32-bitars heltal på adress 0 till en motsvarande vektor på adress 100**.

b. **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**.

c. **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.

d.

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%.

e.

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%.

3. **Se kursboken**.
Always stall: Stanna alltid pipelinen från det att en hoppinstruktion hämtats in tills hoppadress och hoppvillkor är beräknade. Ger stor effektivitetsförlust, men är relativt enkelt att implementera.
Assume not taken: Fortsätt exekvera instruktioner direkt efter ett hopp som om hoppet inte skulle tas. Om hoppet tas måste pipelinen tömmas på felaktigt startade instruktioner. Kräver möjlighet att tömma pipelinen, men är effektivare än *always stall*.
Assume taken. Så snart en hoppinstruktion hämtats börjar instruktioner från hoppdestinationen hämtas. Om hoppet inte tas måste pipelinen tömmas på felaktigt startade instruktioner. Kräver möjlighet att snabbt upptäcka hoppinstruktioner och beräkna hoppadresser, men är effektivare än *assume not taken* eftersom det är fler hopp som tas än som inte tas.
Delayed branch. Ändra betydelsen av hoppinstruktioner så att programmeraren/kompilatorn vet att ett visst antal instruktioner efter en hoppinstruktion utförs innan hoppet tas. Kräver ingen extra hårdvara att implementera, men kan vara svår att utnyttja effektivt för programmerare/kompilator.

4.

a. **Se kursboken**.
1. Processor: aktivera Request och lägg ut adress på Data
2. Minne: läs av adress från Data och aktivera Acknowledge
3. Processor: avaktivera Request och släpp Data
4. Minne: avaktivera Acknowledge
5. Minne: aktivera DataReady och lägg ut data på Data när åtkomsten klar
6. Processor: läs av data från Data och aktivera Acknowledge
7. Minne: avaktivera DataReady och släpp Data
8. Processor: avaktivera Acknowledge

b. **Se kursboken**.
Asynkron buss kräver ej distribution av klocksignal, tillåter kommunikation mellan komponenter med helt olika hastighet, och kan göras lång. Synkron buss kan göras snabbare, och får oftast en enklare protokollimplementering.

c. **Se kursboken**.

Processor-minne	I/O
Synkron	Asynkron
Kort	Lång
Datorspecifik	Standardiserad

5.

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