Lösningsförslag tenta 2011-05-26 (Med reservation för eventuella fel)

1. $X = 100101_2$; $Y = 011001_2$ (6 bitars ordlängd)

a) $[-2^{n-1}, +2^{n-1}-1] = [-2^{6-1}, +2^{6-1}-1] = [-32, +31]$ **(1p)**

b) $[0, 2^n-1] = [0, 2^6-1] = [0, 63]$ **(1p)**

c)
```
S = X+Y
6543210   bitnummer
0000010   carry
 100101   X
+011001   Y
 111110 = S
```
(1p

d) $\underline{N} = s_5 = \underline{1}$
$\underline{Z} = \underline{0}$ ($S \neq 0$)
$\underline{V} = x_5*y_5*s_5' + x_5'*y_5'*s_5 = 1*0*1' + 1'*0'*1 = \underline{0}$
$\underline{C} = c_6 = \underline{0}$

(1p)

e)
```
D = X+Y1k+1
6543210   bitnummer
1001111   carry
 100101   X
+100110   Y1k
 001100 = D
```
(1p)

f) $\underline{N} = d_5 = \underline{0}$
$\underline{Z} = \underline{0}$ ($D \neq 0$)
$\underline{V} = x_5*y_{5_{1k}}*d_5' + x_5'*y_{5_{1k}}'*d_5 = 1*1*0' + 1'*1'*0 = \underline{1}$
$\underline{C} = c_6' = 1' = \underline{0}$

(1p)

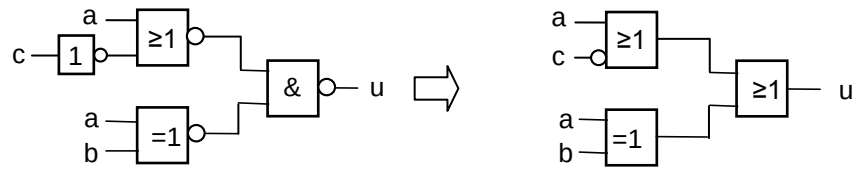
g) $\underline{X} = 100101_2 = 25_{16} = = 2*16 + 5 = 32 + 5 = \underline{37}$
$\underline{Y} = 011001_2 = 19_{16} = 1*16 + 9 = \underline{25}$
$\underline{S} = 111110_2 = 3E_{16} = 3*16 + 14 = \underline{62}$ Resultatet S är korrekt eftersom C = 0.
$\underline{D} = 001100_2 = 0C_{16} = 12 = \underline{12}$ Resultatet D är korrekt eftersom C = 0. **(1p)**

h) ($x_5 = 1$, neg) $X_{2k} = 2^6 - 37 = 64 - 37 = 27$ $\underline{X}$ motsvarar $\underline{-27}$
($y_5 = 0$, pos) $\underline{Y} = \underline{25}$
($s_5 = 1$, neg) $S_{2k} = 2^6 - 62 = 64 - 62 = 2$ $\underline{S}$ motsvarar $\underline{-2}$ Resultatet S är korrekt. (V = 0).
($d_5 = 0$, pos) $\underline{D} = \underline{12}$ Resultatet D är felaktigt eftersom V = 1. **(1p)**

i) $N_{max} = 1.111...1*2^{127} \approx 2*2^{127} = 2^{128} = 0{,}25*2^{130} = 0{,}25*(2^{10})^{13} \approx 0{,}25*(10^3)^{13} = 2{,}5*10^{38}$ **(2p)**

j) En sifferposition läggs till för tecknet. 0287 – 0859 byts ut mot $0287 + 0859_{9k} + 1 =$
$= 0287 + 9140 + 1 = 9428$ vilket motsvarar $-(9428_{10k}) = -(0571 + 1) = -572$ **(2p)**

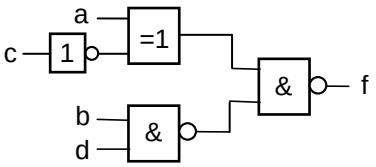
k)



$\underline{u} = a+c'+(a \oplus b) = a(b+b')+c'+(a'b+ab') = ab+ab'+a'b+c' = (a'b')'+c' = \underline{a+b+c'}$ **(3p)**

2. $\underline{f} = a'c+bd+ac' = (a \oplus c)+bd$

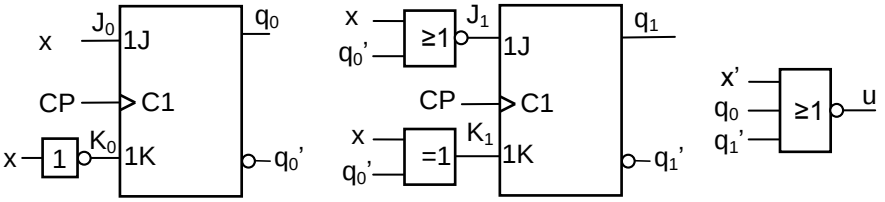
ab \ cd	00	01	11	10
00	0	0	1	1
01	0	1	1	1
11	1	1	1	0
10	1	1	0	0



(4p)

3.

x	q_1	q_0	q_1^+	q_0^+	J_1	K_1	J_0	K_0	u
0	0	0	0	0	0	-	0	-	0
0	0	1	1	0	1	-	-	1	0
0	1	0	0	0	-	1	0	-	0
0	1	1	1	0	-	0	-	1	0
1	0	0	0	1	0	-	1	-	0
1	0	1	0	1	0	-	-	0	0
1	1	0	1	1	-	0	1	-	1
1	1	1	0	1	-	1	-	0	0



J_1 = q_1q_0

x \	00	01	11	10
0	0	1	-	-
1	0	0	-	-

K_1 = q_1q_0

x \	00	01	11	10
0	-	-	0	1
1	-	-	1	0

J_0 = q_1q_0

x \	00	01	11	10
0	0	-	-	0
1	1	-	-	1

K_0 = q_1q_0

x \	00	01	11	10
0	-	1	1	-
1	-	0	0	-

u = q_1q_0

x \	00	01	11	10
0	0	0	0	0
1	0	0	0	1

$J_1 = x'q_0$; $K_1 = (x+q_0')(x'+q_0) = (x \oplus q_0)'$; $J_0 = x$; $K_0 = x'$; $u = xq_1q_0'$ **(6p)**

4. $3(B+1) - 5(A+1) = 3(B-A) - 2(A+1)$

CP	RTN	Styrsignaler (=1)
1	A → T	OE_A, LD_T
2	B – T → R	OE_B, f_3, f_2, g_0, LD_R
3	2R → R, R → T	OE_R, f_3, f_1, f_0, LD_R, LD_T
4	R + T → R	OE_R, f_3, f_1, LD_R
5	A → T	OE_A, LD_T
6	R – T – 1 → R	OE_R, f_3, f_2, LD_R
7	R – T – 1 → R	OE_R, f_3, f_2, LD_R
8	R → B	OE_R, LD_B

(5p)

5. a)

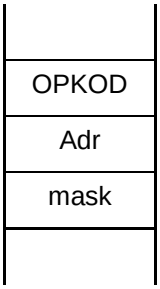
State nr	S-term	RTN-beskrivning	Styrsignaler (=1)
Q_5	$Q_5 \cdot I_{xx}$	X → MA	OE_X, LD_{MA}
Q_6	$Q_6 \cdot I_{xx}$	M → T	MR, LD_T
Q_7	$Q_7 \cdot I_{xx}$	U = B – T, Flags → CC, Next Fetch	OE_B, f_3, f_2, g_0, LD_{CC}, NF

(1p)

b) Instruktionen består av ett ord (Op-kod). Minnet adresseras med innehållet i X-registret. Dataordet som X pekar på i minnet hämtas till T-registret. ALU:n bildar B – T och flaggorna laddas, men skillnaden sparas inte. Detta är CMPB ,X. **(2p)**

c)

State	S-term	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_5	$Q_5 \cdot I_{FA}$	PC → MA, PC+1→PC	OE_{PC}, LD_{MA}, IncPC
Q_6	$Q_6 \cdot I_{FA}$	M → R	MR, f_0, LD_R
Q_7	$Q_7 \cdot I_{FA}$	PC → MA, PC+1→PC	OE_{PC}, LD_{MA}, IncPC
Q_8	$Q_8 \cdot I_{FA}$	M → T	MR, LD_T
Q_9	$Q_9 \cdot I_{FA}$	R→MA	OE_R, LD_{MA}
Q_{10}	$Q_{10} \cdot I_{FA}$	M *OR* T → R, Flags → CC	MR, f_2, f_0, LD_R, LD_{CC}
Q_{11}	$Q_{11} \cdot I_{FA}$	R → M, Next Fetch	OE_R, MW, NF



(4p)

6.

a) Alla subrutiner avslutas med instruktionen RTS, som skall utföra återhopp till det program som anropade subrutinen, dvs RTS hämtar översta värdet på stacken och placerar det i PC. Det förutsätts alltså att stackens översta värde är återhoppsadressen när RTS utförs. Eftersom JMP eller BRA inte placerar någon återhoppsadress på stacken så kommer programmet att spåra ur vid återhoppet. **(2p)**

b) Om overflow (2-komplement) inträffar vid den aritmetiska operation som påverkar flaggorna före det villkorliga hoppet, så har N-flaggan (tecknet) fel värde. Genom att istället för N använda N⊕V får man rätt tecken, även om overflow inträffar. **(2p)**

c) Innan ett villkorligt hopp utförs gör man en jämförelse mellan två tal som vi kan kalla α och β. Jämförelsen utförs som en subtraktion α – β som påverkar flaggorna. Instruktionerna BHI och BGT avser båda villkoret >, dvs. hopp utförs om α > β. Skillnaden är att BHI avser tal utan tecken medan BGT avser tal med tecken. För 8-bitars tal gäller att talområdet för tal utan tecken är [0, 255] medan det för tal med tecken är [-128, +127]. Detta innebär i praktiken att alla 8-bitars tal i intervallet [128, 255] tolkas som negativa och därför uppfattas som mindre än alla tal i intervallet [0, 127]. **(2p)**

d)

Adr	Data	~	Läge			
20	11 40	4		LDX	#$40	
22	0F F6	4		LDAA	#-10	
24	10 05	4		LDAB	#5	
26	41	4	LOOP1	INCA		
27	45	4		DECB		
28	5E FC	5		BNE	LOOP1	26 – 2A = FC
2A	42	4	LOOP2	INCB		
2B	41	4		INCA		
2C	5E FC	5		BNE	LOOP2	2A – 2E = FC
2E	94	6		STAB	B,X	
2F	22 0F	6		EORB	#$0F	
31	8E	5		STAB	1,-X	

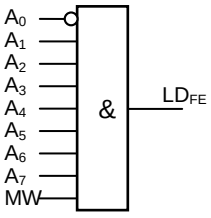
(2p)

e) STAB B,X: 5 skrivs på adressen 45_{16}; STAB 1,-X: A_{16} skrivs på adressen $3F_{16}$ **(1p)**

f) N = 4+4+4+(4+4+5)*5+(4+4+5)*5+6+6+5 = 29+13*10 = 159 klockpulser **(3p)**

7.

a) Utporten skall laddas från databussen vid skrivning på adress FE_{16}. = 11111110_2
(Ring på ingången i figuren till höger betyder att signalen inverteras innan den når OCH-grinden.)



(2p)

b) I detta fall kommer samma värde som matas ut till utporten att skrivas i minnet på adress FE_{16}. Man kan därför läsa i minnet på adress FE_{16} och får då det senast utskrivna värdet på utporten. En annan enkel möjlighet är att skapa en kopia i minnet av det utskrivna värdet, dvs varje gång man skriver ut ett värde på utporten så skriver man samma värde till kopian i minnet. Man kan sedan läsa kopians värde som är identiskt med utportens. **(2p)**

8.

```
*           Subrutin CNT2

CNT2        PSHX                Spara register på stack

            CLRA                Nollställ räknare 1
            CLR     ICNT2       Nollställ räknare 2

CNTLOOP     LDAB    1,X+        Hämta data från tabell. Öka pekare
            CMPB    #$80        Slutmarkering?
            BEQ     CNTEX       Ja, avsluta

            CMPB    #-75        Kolla undre gräns för intervall 1
            BLT     CNTLOOP     Ej intervall 1 och inte heller det andra. Testa nästa data
            CMPB    #75         Kolla övre gräns för intervall 1
            BGT     I_NR2       Ej intervall 1. Testa intervall 2
            INCA                Träff i intervall 1. Räkna

I_NR2       CMPB    #50         Kolla undre gräns för intervall 2
            BLT     CNTLOOP     Ej intervall 2. Testa nästa data

            CMPB    #100        Kolla övre gräns för intervall 2
            BGT     CNTLOOP     Ej intervall 2. Testa nästa data
            INC     ICNT2       Träff i intervall 2. Räkna

            BRA     CNTLOOP     Testa nästa

CNTEX       LDAB    ICNT2       Hämta räknare för intervall 2
            PULX                Återställ register
            RTS

ICNT2       RMB     1           Räknare för intervall 2
```

(7p)