

Lösningsförslag tenta 2009-01-14

1.

a) $R = X + Y_{1k} + 1$

6543210	bitnummer
1001011	1 carry
100101	X
+110101	Y _{1k}
011011	= R

(1p)

b) $N = r_5 = 0$;
 $Z = 0$ ($R \neq 0$);
 $V = x_5 * y_5 * r_5' = 1 * 1 * 0' = 1 * 1 * 1 = 1$; (Vid "subtraktion" är y_5 motsvarande bit i Y_{1k} .)
 $C = c_6' = 1' = 0$

(1p)

c) $\underline{R} = 011011_2 = 1B_{16} = 16 + 11 = \underline{27}$; $\underline{X} = 100101_2 = 25_{16} = 32 + 5 = \underline{37}$; $\underline{Y} = 001010_2 = 0A_{16} = \underline{10}$;
 Korrekt resultat om $C = 0$.

(1p)

d) $\underline{R} = \underline{27}$ ($r_5 = 0$, pos); $\underline{X} = 2^6 - 37 = 64 - 37 = \underline{-27}$ ($x_5 = 1$, neg); $\underline{Y} = \underline{10}$ ($y_5 = 0$, pos);
 Korrekt resultat om $V = 0$.

(1p)

e) 5 decimala sifferpositioner krävs (4 siffror + teckensiffra). $A = 05872_{10}$; $B = 07594_{10}$; $B_{9k} = 92415$
 Princip: $D = A + B_{9k} + 1$

43210	siffernummer
01001	1 carry
05872	A
+92405	B _{9k}
98278	= D

(Resultatet är negativt då $d_4=9$.)

För att kontrollera resultatet kan man 10-komplementera det för att se beloppet.

$$D_{10k} = D_{9k} + 1 = 01721 + 1 = 01722$$

(3p)

f) $10\ 1110\ 1111\ 0111_2 = 2EF7_{16}$

(1p)

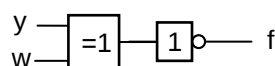
g)

$u = (a'b + ab')ab = a'bab + ab'ab = 0 + 0 = 0$

(3p)

2. $f(x,y,z,w) = (x' + y + w')(y' + z' + w)(y' + z + w)(x + y + w')$

$$f = (y + w')(y' + w) = yw + y'w' = (y \oplus w)'$$

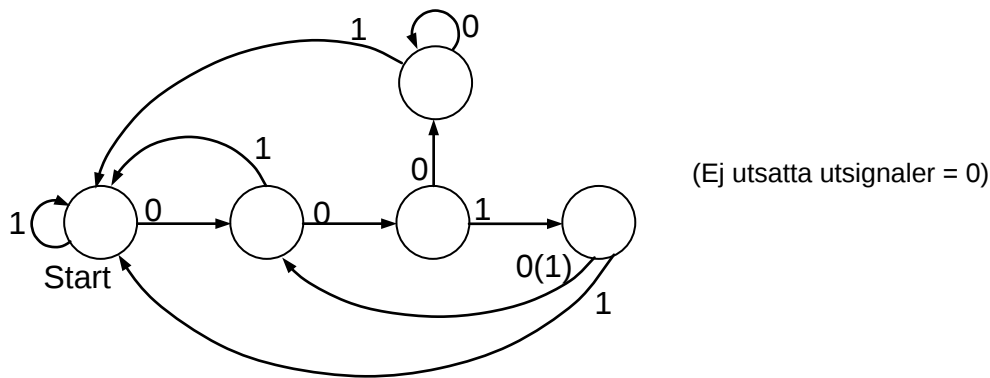


		zw			
		00	01	11	10
xy	00	1	0	0	1
	01	0	1	1	0
	11	0	1	1	0
	10	1	0	0	1

(4p)

3.

a)

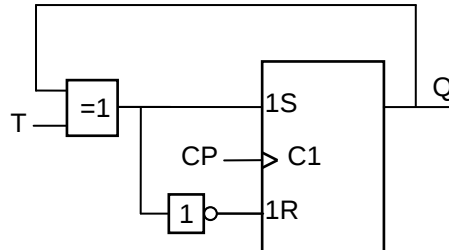


(4p)

b)

T	Q	Q ⁺	S	R
0	0	0	0	1
0	1	1	1	0
1	0	1	1	0
1	1	0	0	1

$$S = R' = T'Q + TQ' = T \oplus Q$$



(4p)

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

CP	RTN	Styrsignaler (=1)
1	B → T	OE _B , LD _T
2	A - T → R	OE _A , f ₃ , f ₂ , g ₀ , LD _R
3	2R → R	OE _R , f ₃ , f ₁ , f ₀ , LD _R
4	2R → R	OE _R , f ₃ , f ₁ , f ₀ , LD _R
5	R - T - 1 → R	OE _R , f ₃ , f ₂ , LD _R
6	R → B	OE _R , LD _B

(5p)

5.

a)

State	Summaterm	RTN-beskrivning	Aktiva styrsignaler (=1)
Q ₅	Q ₅ ·I _{xx}	PC→MA	OE _{PC} , LD _{MA}
Q ₆	Q ₆ ·I _{xx}	M→MA	MR, LD _{MA}
Q ₇	Q ₇ ·I _{xx}	M→MA	MR, LD _{MA}
Q ₈	Q ₈ ·I _{xx}	M→PC, NF	MR, LD _{PC} , NF

(1p)

- b) Q₅: Minnesadressen efter OP-koden placeras i MA.
 Q₆: Dataordet efter OP-koden i minnet placeras i MA. (Alltså en adress.)
 Q₇: Dataordet som pekas ut av adressen efter OP-koden i minnet placeras i MA. (Ny adress!)
 Q₈: Dataordet på den nya adressen placeras i PC. (Alltså ett hopp!)

Detta skulle innebära ett indirekt-indirekt hopp, dvs JMP [[Adress]]

(2p)

c)

State	Summaterm	RTN-beskrivning	Aktiva styrsignaler (=1)
Q ₅	Q ₅ ·I _{F0}	B + 1 → R, Flags → CC	OE _B , f ₃ , g ₀ , LD _R , LD _{CC}
Q ₆	Q ₆ ·I _{F0}	R → B	OE _R , LD _B
Q ₇	Q ₇ ·I _{F0} Z Q ₇ ·I _{F0} Z'	If Z = 1: PC + 1 → PC, NF else: PC → MA, T	IncPC, NF OE _{PC} , LD _{MA} , LD _T
Q ₈	Q ₈ ·I _{F0}	M + T + 1 → R	MR, f ₃ , f ₁ , g ₀ , LD _R
Q ₉	Q ₉ ·I _{F0}	R → PC, NF	OE _R , LD _{PC} , NF

(5p)

6.

- a) Q₃: PC → MA, PC + 1 → PC Kopiera PC-värdet till MA. Öka PC.
 Q₄: M → IR Läs där "gamla" PC pekade (OP-kod) och placera den i IR. (2p)
- b) JSR placerar adressen till nästa instruktion (dvs. återhoppadressen till huvudprogrammet) överst på stacken innan hoppet utförs. JMP påverkar ej stacken. (2p)
- c) FLEX-datorn använder minnesorienterad I/O. Fördelen med det är att vanliga instruktioner som läser eller skriver i minnet kan användas för I/O. I fallet separatadresserad I/O krävs speciella I/O-instruktioner. (2p)
- d) Före ett villkorligt hopp antas att flaggorna har påverkats av en subtraktion mellan två tal som skall jämföras. Storleksrelationen mellan talen kan då bestämmas med hjälp av flaggornas värden. För villkorliga hopp där hoppvillkoret avser tal utan tecken används flaggorna C och Z. C visar om lånesiffra uppstod och Z visar om talen var lika. För motsvarande hopp som avser tal med tecken används istället $N \oplus V$ som visar om skillnaden blev negativ och Z som visar om talen var lika. Uttrycket $N \oplus V$ har den goda egenskapen att det visar skillnadens korrekta tecken även om overflow uppstod vid subtraktionen. (4p)

e)

Adr	Data	~	Läge	
00	11 0F	4		LDX #DATA
02	81 03	7		LDAA 3,X
04	7A	4	WAIT1	LDAB ,X
05	45	4	WAIT2	DECB
06	00	3		NOP
07	5E FC	5		BNE WAIT2
09	00	3		NOP
0A	44	4		DECA
0B	5E F7	5		BNE WAIT1
0D	5A 09	5		BRA P2
0F	00 01 02 04 08 10 20 40 80	-	DATA	FCB 0,1,2,4,8,16,32,64,128
18	0B FD	5	P2	LDAA \$FD

05 – 09 = FC; 04 – 0D = F7; 18 – 0F = 09 (3p)

f) $t = [4 + 7 + (4 + (4 + 3 + 5) \cdot 256 + 3 + 4 + 5) \cdot 4 + 5 + 5] \mu s = [21 + (16 + 12 \cdot 256) \cdot 4] \mu s = 12373 \mu s$ (3p)

7.	KEY	EQU	\$C0	
	KEYCMP	PSHB PSHX		Spara register på stack
		CLRA		Nollställ felräknare
		PSHX		Spara pekare till "User string"
		LDX #KEY		Initiera adressen till nyckelsträngen
		STX KEYADR		Uppdatera nyckelpekare
		PULX		Återställ pekaren till "User string"
	KLOOP	LDAB 1,X+		Hämta data från "User string"
		TSTB		Kolla om strängslut
		BEQ KEXIT		Ja, strängslut
		ANDB #\$7F		Nej, fortsätt med att jämföra med nyckelsträng
		PSHX		Spara pekare till "User string"
		LDX KEYADR		Hämta nyckelpekare
		INC KEYADR		Uppdatera nyckelpekare
		CMPB ,X		Jämför "User data" med nyckeldata
		PULX		Återställ pekare till "User string"
		BEQ KLOOP		Data matchar, testa nästa
		INCA		Data stämmer ej med nyckel. Öka felräknare
		BRA KLOOP		Testa nästa
	KEXIT	PULX PULB RTS		Återställ register
	KEYADR	RMB 1		Variabel för pekare till nyckelsträng

(8p)