EDA215 Digital- och datorteknik för Z

Tentamen

Måndag 17 december 2007, kl. 08.30 - 12.30 i M-salar

Examinatorer
Rolf Snedsböl, tel. 772 1665

Kontaktpersoner under tentamen
Som ovan.

Tillåtna hjälpmedel

Häftet

Instruktionslista för FLEX

och

Instruktionslista för CPU12

I dessa får rättelser och understrykningar vara införda, inget annat.

Tabellverk och miniräknare får ej användas!

Allmänt
Siffror inom parentes anger full poäng på uppgiften. **Full poäng kan fås om**:

- redovisningen av svar och lösningar är läslig och tydlig. **OBS!** Ett lösningsblad får endast innehålla redovisningsdelar som hör ihop med en uppgift.
- din lösning ej är onödigt komplicerad.
- du motiverat dina val och ställningstaganden
- redovisningen av en hårdvarukonstruktion innehåller funktionsbeskrivning, lösning och realisering.
- redovisningen av en mjukvarukonstruktion i assembler är fullständigt dokumenterad, d v s är redovisad både i strukturform (flödesplan eller pseudospråk) och med kommenterat program i assemblerspråk, om inget annat anges i uppgiften.

Betygsättning
För godkänt slutbetyg på kursen fordras att både tentamen och laborationer är godkända. Tentamen ger slutbetyget:

20p ≤ **betyg 3** < 30p ≤ **betyg 4** < 40p ≤ **betyg 5**

Lösningar
anslås på kursens www hemsida (EDA215).

Betygslistan
anslås såsom anges på kursens hemsida

Granskning
Tid och plats anges på kursens hemsida.

Avdelningen för datorteknik
Institutionen data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA

1. Koder, talomvandling, aritmetik och flaggor

I uppgift a-d nedan används 5-bitars tal. X=01011 och Y=11010

a. Visa med penna och papper hur räkneoperationen R = X-Y utförs i en dator (i en ALU). **(1p)**

b. Ange sedan flaggbitarna N, Z, V, C **(1p)**

c. Tolka bitmönstren R, X och Y som tal *utan* tecken och ange dess decimala motsvarighet **(1p)**

d. Tolka bitmönstren R, X och Y som tal *med* tecken och ange dess decimala motsvarighet. **(1p)**

I uppgift **e** och **f** nedan används ett 4-bitars ord bestående av en 3-bitars Graykod ($b_3b_2b_1$) och en paritetsbit (b_0).

e. *Graykoden*: I Gray-koden används sex stycken kodord, [0,5] enligt modulo-6.
Ange dessa kodord. Det skall klart framgå vilken siffra de olika kodorden svarar mot. **(2p)**

f. *Paritetsbiten*: Komplettera varje kodord med en paritetsbit. De skall ha jämn paritet. **(1p)**

---

2. Digitalteknik, kombinatoriska nät

a. Man behöver en 2-ingångs OR-grind men har bara 2- ingångs AND-grindar. Om det är möjligt, hur kopplar man upp OR-grinden med dessa? (Du har inga inverterare) **(1p)**

b. Undersök med fullständig binär evaluering om de booleska funktionerna
f(x,y,z) och g(x,y,z) är lika.
f(x,y,z) = x⊕z + yz. g(x,y,z) = (x+z')(y+z)' **(2p)**

c. Konstruera ett kombinatoriskt nät med fyra insignaler (x, y, z, w) och en utsignal (f) enligt följande:
Utsignalen f=1 för $2 \leq (xyzw)_2 \leq 10$. För alla andra värden på insignalen är f=0.
Insignalerna bildar ett binärtal $(xyzw)_2 \in [0,15_{10}]$.

Gör en minimal lösning och rita upp det kombinatoriska nätet.
Använd NAND/NAND-logik (disjunktiv form). Du har tillgång till inverterare och NAND-grindar med valfritt antal ingångar (Inga andra grindar får förekomma i din lösning).
(5p)

---

3. Digitalteknik, sekvensnät

a. Ange excitationstabellen för en JK-vippa. **(1p)**

b. Visa hur du konstruerar en JK-vippa av en SR-vippa och AND-grindar. **(2p)**

c. En räknare har två utsignaler Q_1 och Q_0. Konstruera en räknare som har räknarsekvensen **1,3,1,0,**1,3,1,0 etc. på utsignalerna Q_1 och Q_0.

Använd T-vippor. Du har tillgång till vanliga grindar (AND, NAND, OR, NOR) med valfritt antal ingångar, samt INVERTERARE. Du kan bortse från hur räknaren startas. Konstruktionen skall vara minimal.

Det skall tydligt framgå hur utsignalerna Q_1 och Q_0 bildas. **(6p)**

---

4. Styrenheten för FLEX

a. I tabellen nedan visas RTN-beskrivningen för EXECUTE-sekvensen för en av FLEX-processorns instruktioner. NF i tabellens sista rad anger att nästa tillstånd (state) skall vara det första i FETCH-sekvensen. Rita en tabell där du anger State nr (0..3) och Styrsignaler. Endast styrsignaler = 1 skall anges.
Du kan utelämna RTN-beskrivningen i din tabell

State nr	RTN-beskrivning	Styrsignaler (=1)
0	PC→MA, PC+1→PC	
1	M→MA	
2	M-1 →R, Flaggor→CC	
3	R→M, NF	

Förklara sedan med ord vad instruktionen ovan utför i varje klockcykel.
Skriv instruktionen med assemblerspråk. **(2p)**

b. Rita en tabell motsvarande den ovan, som visar utförandefasen för den nya maskininstruktionen *Arithmetic Shift Left Register D* **ASLD** för FLEX-processorn. Instruktionen utför **2D →D** där D är register A och B ihopslagna. (Jfr CPU12-instruktionen ASLD) **(5p)**

5. Småfrågor och assemblerprogrammering för FLEX

a. Instruktionerna CMPA, CMPB, CMPS och CMPX fungerar inte i ett FLEX-system i labbet. Kan du skriva en instruktionssekvens som utför

```
CMPA  #$7F
```

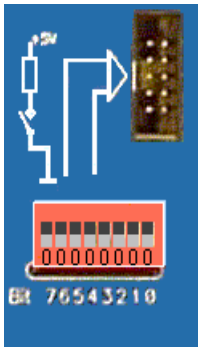
där du inte utnyttjar CMP-instruktioner. Register A, B, S och X skall ha samma värden som tidigare när din instruktionssekvens är klar. **(2p)**

b. Instruktionen BEQ Loop har sin OP-kod på adress $5C_{16}$. Loop har adressen $3A_{16}$.
Ange instruktionens maskinkod. **(2p)**

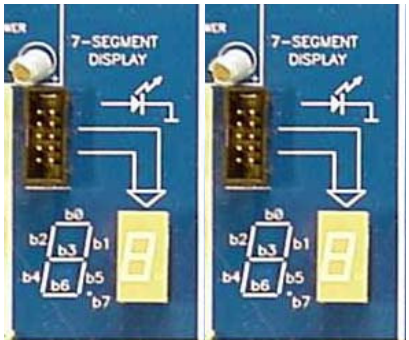
c. Studera följande dummy-program för FLEX.

```
        ORG  $30
Start
        LDB  #13
        LDX  $30
        STB  ,X+
        ADDB $34
        TFR  CC,A
Stop
        NOP
```

Programmet startas och körs fram till Stop.
Ange vad register A, B, X och PC för värden vid Stop? Visa hur du resonerat. **(5p)**



6. Assemblerprogrammering för MC12.



I kursen använde du 7-sifferindikatorn som var anslutet till utporten på MC12. Vidare använde du 8 strömbrytare som inport (Dip Switch Input).

Du skall skriva subrutiner till följande program som om och om igen läser inporten (ett NBCD-tal [$0,99_{10}$]) och skriver detta till två sifferindikatorer.

```
    org     $2000
    LDS     #$3000      Init Stack
Loop
    JSR     Read        Läs NBCD-tal till register A
    JSR     Display     Skriv register A på 2 sifferindikatorer
    JMP     Loop
```

Vissa definitioner är givna. Se nedan.

Subrutinen Display visar det NBCD-tal som finns lagrad i register A. Innehåller register A exempelvis **0101 1001** skall 5 visas på UtPort1 och 9 visas på UtPort2. (Segmentkoder är givna nedan)

Skriv subrutinen Display!

Subrutinen Read läser InPort. Tyvärr har inporten ett konstruktionsfel så bitarna är omkastade enligt följande figur. (Bit b_7 är ju normalt till vänster och b_0 till höger)

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7

Detta medför att när vi ställer in NBCD-talet 53 (**0101 0011**) på strömbrytarna så läses **1100 1010** från inporten (ty det spegelvänds).

Subrutinen skall därför

1. läsa inporten
2. spegelvända det inlästa
3. lämna utdata i i register A..

Skriv subrutinen Read!

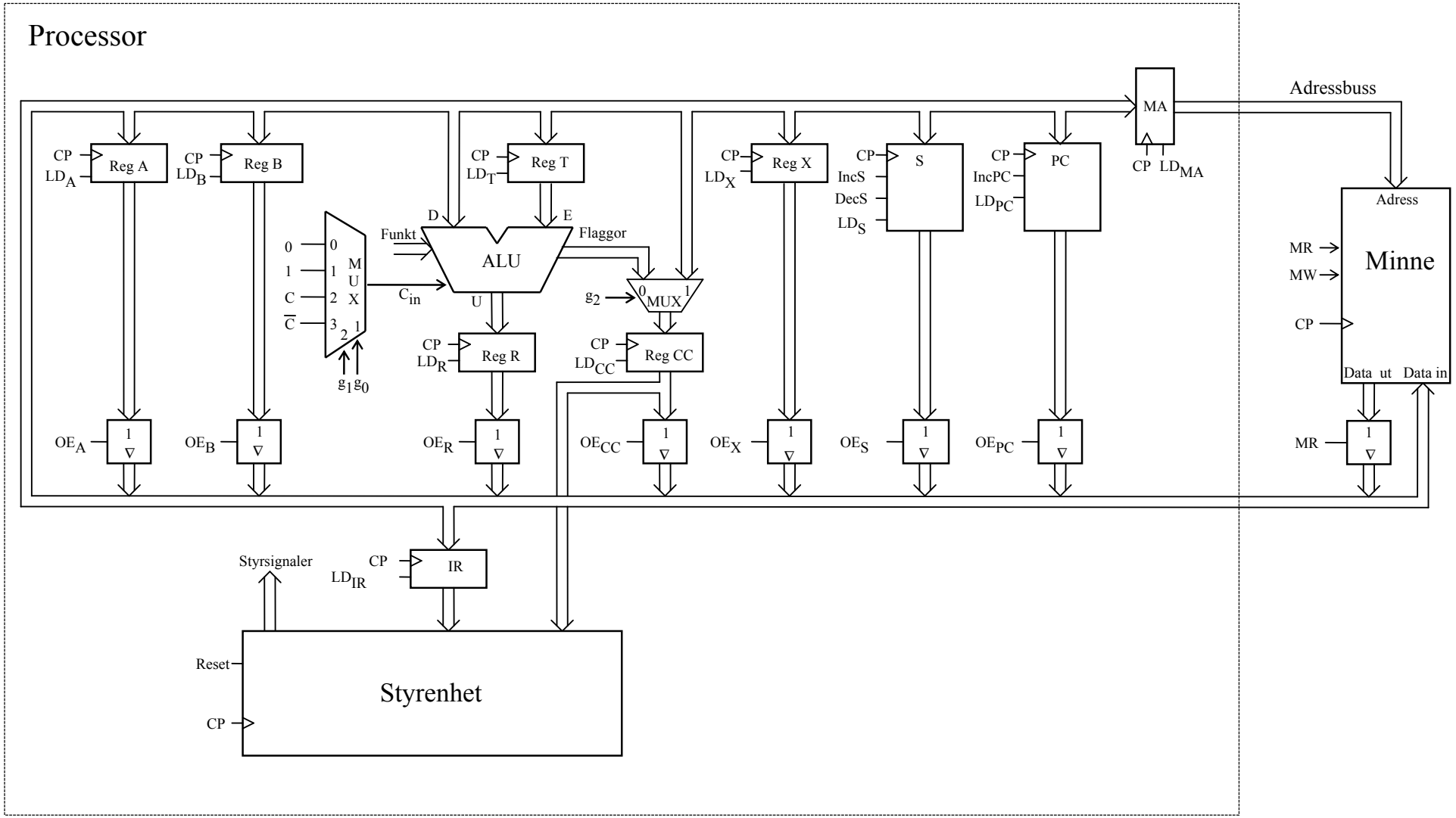
Du har tillgång till följande definitioner och en tabell i minnet med segmentkoder (mönster för sifferindikatorn) enligt:

```
InPort      equ     $600        Strömbrytare
UtPort1     equ     $400        Sifferindikator 1
UtPort2     equ     $401        Sifferindikator 2
SegCodes    FCB     $77,$22,$5B,$6B, etc.
```
(segmentkoder för siffrorna [0,9]).

(10p)

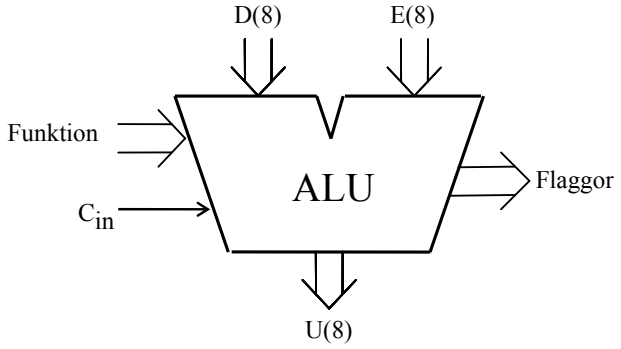
FLEX-datorn

Bilaga 1



ALU:ns funktion

Bilaga 2



ALU:ns **operation (logik- eller aritmetik-)** på indata **D**, **E** och $\mathbf{C_{in}}$ bestäms av insignalerna **Funktion [F = ($\mathbf{f_3}$, $\mathbf{f_2}$, $\mathbf{f_1}$, $\mathbf{f_0}$)]** enligt tabellen nedan**.**. I kolumnen Operation förklaras, när det behövs, hur operationen utförs. Med"**+**" och "**–**" avses **aritmetiska operationer**.

f_3 f_2 f_1 f_0	U = f(D,E,C_{in})	
	Operation	**Resultat**
0 0 0 0	bitvis nollställning	0
0 0 0 1		D
0 0 1 0		E
0 0 1 1	bitvis invertering	D_{1k}
0 1 0 0	bitvis invertering	E_{1k}
0 1 0 1	bitvis OR	D OR E
0 1 1 0	bitvis AND	D AND E
0 1 1 1	bitvis XOR	D XOR E
1 0 0 0	D + 0 + C_{in}	D + C_{in}
1 0 0 1	D + FFH + C_{in}	D – 1 + C_{in}
1 0 1 0		D + E + C_{in}
1 0 1 1	D + D + C_{in}	2D + C_{in}
1 1 0 0	D +E_{1k} + C_{in}	D – E – 1 + C_{in}
1 1 0 1		0
1 1 1 0		0
1 1 1 1	bitvis ettställning	FFH

Flaggorna är utsignaler och för de gäller:

Carryflaggan (C) är minnessiffran ut (carry-out) från den mest signifikanta bitpositionen (längst till vänster) när en aritmetisk operation utförs av ALU:n.

Vid **subtraktion** gäller för denna ALU att
C = 1 om lånesiffra (borrow) uppstår och
C = 0 om lånesiffra inte uppstår.

Carryflaggans värde är 0 vid andra operationer än aritmetiska.

Overflowflaggan (V) visar när en aritmetisk operation ger "overflow" enligt reglerna för 2-komplementaritmetik.

V-flaggans värde är 0 vid andra operationer än aritmetiska.

Zeroflaggan (Z) visar när en ALU-operation ger värdet noll som resultat på U-utgången.

Signflaggan (N) är identisk med den mest signifikanta biten (teckenbiten) av utsignalen U från ALU:n.

Half-carryflaggan (H) är minnessiffran (carry) mellan de fyra minst signifikanta och de fyra mest signifikanta bitarna i ALU:n.

H-flaggans värde är 0 vid andra operationer än aritmetiska.

Tentamen i Digital och Datorteknik, 2007-12-17

1a) R=X-Y utförs som R=X+Y_{1komp}+1 Y_{1komp} = 00101.

1b) N=0; Z=0; V=1; C_5=0 ⇒ C=1

1c) X=11; Y= 26; R=17 (11-26≠17, verkar rimligt ty C=1)

1d) X= 11; Y= -6: R=-15 (11-(-6)) ≠-15, verkar rimligt ty V=1)

1e) Se s 2.16 i blåa boken. 0011,0110,0101,1100,1111,1010 motsvarar 0,1,2,3,4,5

```
                01111
 X               01011
+Y1komp       + 00101
+1            +     1
=R            = 10001
```

Upg 2

2a) Morgan: (A+B)'=A'B'. NEJ inte möjligt, ty det saknas en inverterande funktion.

2b)

			VL			HL		
x	**y**	**z**	**x⊕z**	**yz**	**f**	**x+z'**	**(y+z)'**	**g**
0	0	0	0	0	**0**	1	1	**1**
0	0	1	1	0	**1**	0	0	**0**
0	1	0	0	0	**0**	1	0	**0**
0	1	1	1	1	**1**	0	0	**0**
1	0	0	1	0	**1**	1	1	**1**
1	0	1	0	0	**0**	1	0	**0**
1	1	0	1	0	**1**	1	0	**0**
1	1	1	0	1	**1**	1	0	**0**

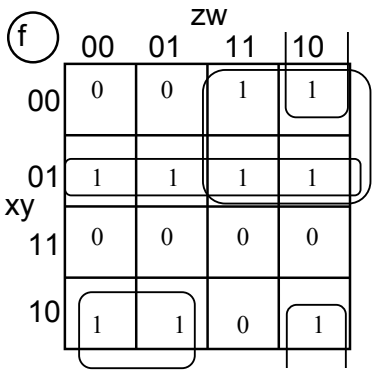
Enligt tabellen så är
g(x,y,z) ≠ f(x,y,z)

x	y	z	w	f_a
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

2c)

Analys / Funktionsbeskrivning

Lösning o minimering



f = x'y + x'z + xy'z'+y'zw'

Realisering

Rita upp nätet med AND/OR och konvertera till NAND/NAND enligt:

f = ((x'y)'(x'z)'(xy'z')'(y'zw')')'

Upg 3

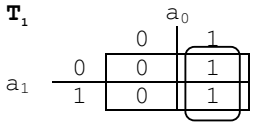
3a) Se lösning uppgift 5.18 i blåa boken del 1

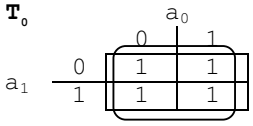
3b) Se sid 5.14, Figur 5.26 i kursboken

3c) Räknaren har totalt 4 tillstånd ⇒ A0,A1,A2,A3 på två utsignaler a_1 och a_0 ⇒ 2 vippor

Detta tillst	Nästa tillst		
a_1a_0	a_1a_0	T_1	T_0
00	01	0	1
01	10	1	1
10	11	0	1
11	00	1	1

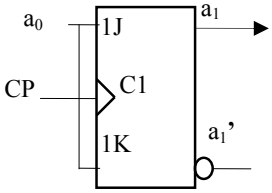
Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

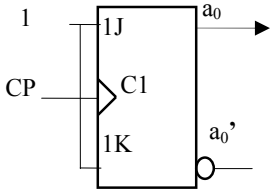




$T_2 = a_0$ $T_1 = 1$

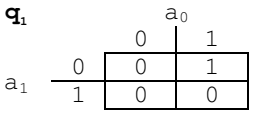
Realiseras med två T-vippor

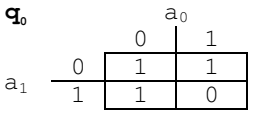




Koda om dessa 4 A-tillstånd till 1,3,1,0 på Q_1 och Q_0.

A	a_1	a_0	q_1	q_0	Q
0	0	0	0	1	1
1	0	1	1	1	3
2	1	0	0	1	1
3	1	1	0	0	0





Utgångarna Q_1 och Q_0 blir således: $q_1 = a_1'a_0$ och $q_0 = (a_1a_0)'$

Upg 4

4a)

State nr	RTN-beskrivning	Styrsignaler (=1)
0	PC→MA, PC+1→PC	OE_{PC}, LD_{MA}, IncPC
1	M→MA	MR, LD_{MA}
2	M-1 →R, Flaggor→CC	MR, f_3, f_0, LD_R, LD_{CC}
3	R→M, NF	OE_R, MW, NF

0) Förbered för läsning av adressoperand i minnet, Öka PC med ett
1) Läs adressoperanden från minnet till register MA för att förbereda läsning av data
2) Läs data från minnet, minska med ett ospara resultatet i register R. Påverka flaggbitarna.
3) Skriv resultatet till minnet

Instruktionen är DEC Adr

4b)

State nr	RTN-beskrivning	Styrsignaler (=1)
0	2B → R, Flaggor → CC	OE_B, LD_R, f_3, f_1, f_0, LD_{CC},
1	R → B	OE_R, LD_B,
2	2A+C → R, Flaggor → CC	OE_A, LD_R, f_3, f_1, f_0, g_1, LD_{CC},
3	R → A, NF	OE_R, LD_A,

Upg 5

5a) (CMP = Påverka flaggorna med en SUB utan att påverka registret)

```
PSHA            Spara A
SUBA   #$7F     Påverka flaggor
PULA            Återhämta A
```

5b) FrånAdress = $5C+2, TillAdress = 3A. Offset = TillAd-FrånAdr = $DC.
Maskininstruktionen blir $5D, $5C

5c)

Adr				Registerpåverkan (HEX-koder)
30		Start		PC=30
30	10	LDB	#13	0D→B
31	0D			
32	0D	LDX	$30	M(30)→X *(10→X)*
33	30			
34	8C	STB	,X+	X+1→X=10+1=11
35	29	ADDB	$34	B+M(34)=0D+8C=99→B, 08→CC (N=1)
36	34			
37	04	TFR	CC,A	08→A
38		Stop		PC=38
38	00	NOP		

Svar: A=08, B=99, X=11, PC=38. Enbart Hex-värden.

Upg 6

```
* Subrutin Display visar ett NBCD-tal i A på två sifferindikatorer
* Indata: Register A, Ett NBCD-tal [0,99]

Display pshb
        pshx

        ldx     #SegCodes  Pekare
        tfr     a,b        Spara kopia

* Register A används för EN-talen och...
* Register B används för TIO-talen

        lsrb            Skifta fram TIO-talen
        lsrb
        lsrb
        lsrb

        ldab    b,x        ..och visa TIO-talen
        stab    UtPort1

        anda    #$0f       Ta fram EN-talen
        ldaa    a,x        ..och visa EN-talen
        staa    UtPort2

        pulx
        pulb
        rts


* Subrutin Display laser inporten, spegelvänder data
* och lämnar detta I Register A
* Utdata: Register A.
Display pshb
        pshx
        ldab   InPort
        ldx    #8        Skifta 8 bitar

Loop    lsrb             Skifta ut...
        rola             ... och in
        dex              Sista?
        bne    Loop      Nej

        pulx
        pulb
        rts
```