EDA215 Digital och datorteknik Z

EDA432 Digital och datorteknik IT

EDA311 Digital och datorteknik E

INN790 Digital och datorteknik GU

Gamla och nya kursen

Tentamen

Tisdag 18 april 2006, kl. 8.30 - 12.30 i V-salar

Examinatorer
Rolf Snedsböl, tel. 772 1665

Kontaktpersoner under tentamen
Som ovan

Tillåtna hjälpmedel

Häftet

Instruktionslista för FLEX

och

Instruktionslista för MC6809

eller

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.

Betygslistan
anslås såsom anges på kursens hemsida.

Granskning
Tid och plats anges på kursens hemsida.

Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA

Uppgift 0 *Vi behöver din hjälp*.

Ni är många olika grupper/linjer som tentar nu.
Kan du hjälpa till genom att på rad 10 på försättsbladet ange den kurskod (EDAxxx, DITxxx, DATxxx) som du är registrerad på. Skriv även när du läste kursen ex: HT05, VT 03 etc .Om du inte vet vilken kurskod du har så skriv ”Vet Ej” på rad 10. *Tack för din hjälp/rolf*

Uppgift 1 *Talomvandling, koder, aritmetik och flaggor*.

I uppgift a-d nedan används 5-bitars tal. X = 01101 och Y = 11001.

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.
Vilken(vilka) flaggbit(ar) anger om resultatet är korrekt vid tal utan tecken? **(1p)**

d) Tolka bitmönstren R, X och Y som tal *med* tecken och ange dess decimala motsvarighet.
Vilken(vilka) flaggbit(ar) anger om resultatet är korrekt vid tal med tecken? **(1p)**

e) *ASCII-kod och Paritet*. Du har mottagit fyra 8-bitars kodord som skall bilda ett svenskt namn. Kodorden är 96_{16}, CB_{16}, $E4_{16}$, $E8_{16}$. Varje kodord innehåller 7-bitars ASCII-kod [b_7-b_1]och en jämn paritetsbit [b_0]. I ett av kodorden har en bit ändrats vilket gett upphov till paritetsfel.
Vilket kodord har paritetsfel? Vilken bit i kodordet har ändrats? Visa hur du resonerat! **(3p)**

Uppgift 2 *Digitalteknik - Småfrågor*

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

b) Ange funktionstabellen och excitationstabellen för en D-vippa. **(2p)**

c) Den booleska funktionen f(x,y,z) = xy’z + xz + xy’z’ är given. Ange denna på konjunktiv och disjunktiv minimal form. **(4p)**

d) Rita grindnätet för följande funktion: f(x,y,z,w)=x’z+w⊕y+(x·z·w)’
Du har tillgång till två-ingångs AND, OR, XOR samt inverterare **(2p)**

Uppgift 3 *Digitalteknik - Konstruktion*

a) Konstruera ett kombinatoriskt nät med tre insignaler (x, y, z) och en utsignal (f) enligt följande:
Utsignalen f=1 för $3 \leq xyz \leq 6$. För alla andra värden på insignalen är f=0.
Insignalerna bildar ett binärtal $xyz \in [0,7]$.

Gör en minimal lösning och rita upp det kombinatoriska nätet.
Använd NAND/NAND-logik (disjunktiv form). Du har tillgång till NAND-grindar med valfritt antal ingångar (Inga andra grindar får förekomma i din lösning). **(4p)**

b) Konstruera och rita upp en räknare som har räknarsekvensen **0,1,3,2,** 0,1,3,2 etc. på de två utsignalerna Q_1 och Q_0. Du kan bortse från hur räknaren startas. Använd JK-vippor. Du har tillgång till vanliga grindar (AND, NAND, OR, NOR) med valfritt antal ingångar, samt INVERTERARE.

(6p)

Uppgift 4 *Styrenhet och dataväg för FLEX.*

a) I tabellen nedan visas RTN-beskrivningen för EXECUTE-sekvensen för en **instruktion** för FLEX-processorn. 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..4) och Styrsignaler. Endast styrsignaler = 1 skall anges. Du kan utelämna RTN-beskrivningen i din tabell **(2p)**

State	RTN-beskrivning	Styrsignaler (=1)
0	PC→MA, PC+1→PC, S - 1→S	
1	M→T	
2	S→MA	
3	PC → M, T → R	
4	R→ PC, NF	

b) Förklara i ord vad instruktionen ovan utför i varje klockcykel.
Ange sedan instruktionen med assemblerspråk för FLEX-processorn (Ex: **LDA Adr**). **(2p)**

c) Rita en tabell motsvarande den ovan, som visar utförandefasen för maskininstruktionen **LDA Adr** för FLEX-processorn. I instruktionslistan för FLEX-processorn beskrivs instruktionen. Ange RTN-beskrivning och styrsignaler. **(5p)**

Uppgift 5 *Småfrågor rörande FLEX-processorn*

a) Hur många databitar kan det maximalt finnas i ett minne som ansluts till FLEX-processorn? Resonera! **(2p)**

b) Nedan visas en del av ett assemblerprogram. Koden gör en beräkning på data som hittas i minnesområdet [$30,$3F]. Följande minnesinnehåll gäller i detta exempel: Adress $30 innehåller 10, adress $31 innehåller 11, adress $32 innehåller 12, etc. fram till adress $3F som innehåller $1F.

Vad skrivs till adress $F0 när koden har exekverat klart? Visa hur du resonerat.

```
        -
        -
      LDX  #$38
      LDA  #0
      LDB  #3
LOOP  ADCA ,-X       (Förutsätt att denna instruktion finns för flex, se även nedan)
      DECB
      BNE  LOOP
      STA  $F0       Vad skrivs till adressen $F0?
        -
        -
```

(Du bör veta hur instruktionerna **ADCA #Data** / **ADCA Adr** och **LDA ,-X** / **STA ,-X** fungerar. **ADCA ,-X** använder adressering via X-registret,. Se instruktionslistan) **(3p)**

c) Koden ovan innehåller fel. Kan du hitta felet? (Enbart svaret JA godtas inte, utan motivation krävs) **(2p)**

Uppgift 6 *Assemblerprogrammering av MC12 eller MC6809-processorn*

Lös *antingen* uppgift 6a *eller* 6b, inte båda.
Om du är registrerad på gamla kursen och haft MC6809-processorn löser du uppgift 6a.
Om du är registrerad på nya kursen med MC12 löser du uppgift 6b.

Uppgift 6a

Avbrott och assemblerprogrammering. Ett 6809-system är bestyckad med en pulsgenerator som genererar avbrott var 10:de ms.

Du behöver en rutin (IRQINIT) som initierar systemet och en avbrottsrutin (IRQ) som uppdaterar en klock-variabel. Klockvariabeln skrivs till en display av huvudprogrammet. När programmet startar skall displayen visa (börja på) 00:00:00. Avbrott kvitteras genom en skrivning på den symboliska adressen IRQRES. Initieringsrutinen (IRQINIT): Skriv en rutin som initierar systemet för avbrott. Det finns inga andra avbrottskällor än pulsgeneratorn i systemet.

Avbrottsrutinen (IRQ): Skriv en avbrottsrutin som uppdaterar klock-variabeln. Variabeln hittas på den symboliska adressen CLOCK och består av 3 bytes enligt

```
CLOCK   RMB 3   Variabel innehållande klockan tt:mm:ss
```

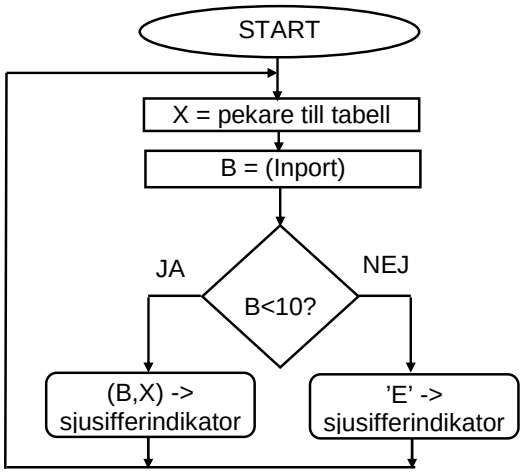
där tt är 0-23 timmar, mm är 0-59 minuter och ss 0-59 sekunder som alla tre lagras på binär form.
.
Du får själv skapa ytterligare hjälpvariabler för klockavbrotten. **(7p)**

Uppgift 6b

Vid simulatorpassen och i labbet använde du stömbrytarna (ML4 INPUT) och sifferindikatorn (ML4 OUTPUT).

Du skall nu
skriva ett program som hela tiden läser strömbrytarna, översätter avläst värde till segmentkod och skriver ut till sifferindikatorn. Om avläst värde från strömbrytarna är större än nio skall ett E (ERROR) skrivas ut.

Studera även den givna flödesplanen i marginalen.



Du har tillgång till en tabell med segmentkoder och följande defenitioner:

```
Inport   EQU  xxxx          Adress för inporten
Utport   EQU  zzzz          Adress för utport
Error    EQU  pp            Segmentkod för E (Error)
SegCode  FCB  xx,yy,zz,etc  Tabell med segmentkoder för [0,9]
```

(7p)

Bilaga 1 - Assemblerspråket för mikroprocessorn CPU12.

Assemblerspråket använder sig av de mnemoniska beteckningar som processorkonstruktören MOTOROLA specificerat för maskininstruktioner och instruktioner till assemblatorn, s k pseudoinstruktioner eller assemblatordirektiv. Pseudoinstruktionerna framgår av följande tabell:

Direktiv		Förklaring
	ORG N	Placerar den efterföljande koden med början på adress N (ORG för ORiGin = ursprung)
L	RMB N	Avsätter N bytes i följd i minnet (utan att ge dem värden), så att programmet kan använda dem. Följden placeras med början på adress L. (RMB för Reserve Memory Bytes)
L	EQU N	Ger label L konstantvärdet N (EQU för EQUates = beräknas till)
L	FCB N1, N2	Avsätter i följd i minnet en byte för varje argument. Respektive byte ges konstantvärdet N1, N2 etc. Följden placeras med början på adress L. (FCB för Form Constant Byte)
L	FDB N1, N2	Avsätter i följd i minnet ett bytepar (två bytes) för varje argument. Respektive bytepar ges konstantvärdet N1, N2 etc. Följden placeras med början på adress L. (FDB för Form Double Byte)
L	FCS "ABC"	Avsätter en följd av bytes i minnet, en för varje tecken i teckensträngen "ABC". Respektive byte ges ASCII-värdet för A, B, C etc. Följden placeras med början på adress L. (FCS för Form Caracter String)

Bilaga 2 - ASCII-koden.

0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1	$b_6b_5b_4$ $b_3b_2b_1b_0$
NUL	DLE	SP	0	@	P	`	p	0 0 0 0
SOH	DC1	!	1	A	Q	a	q	0 0 0 1
STX	DC2	“	2	B	R	b	r	0 0 1 0
ETX	DC3	#	3	C	S	c	s	0 0 1 1
EOT	DC4	$	4	D	T	d	t	0 1 0 0
ENQ	NAK	%	5	E	U	e	u	0 1 0 1
ACK	SYN	&	6	F	V	f	v	0 1 1 0
BEL	ETB	‘	7	G	W	g	w	0 1 1 1
BS	CAN	(	8	H	X	h	x	1 0 0 0
HT	EM	)	9	I	Y	i	y	1 0 0 1
LF	SUB	*	:	J	Z	j	z	1 0 1 0
VT	ESC	+	;	K	[Ä	k	{ä	1 0 1 1
FF	FS	,	<	L	\Ö	l	|ö	1 1 0 0
CR	GS	-	=	M	]Å	m	}å	1 1 0 1
S0	RS	.	>	N	^	n	~	1 1 1 0
S1	US	/	?	O	_	o	RUBOUT (DEL)	1 1 1 1

Tentamen i Digital o Datorteknik för E, GU, IT, Z. 2006-04-18

Kortform av lösningar till tentan. För full poäng krävs fullständiga lösningar enligt typtentan

1a) R=X-Y utförs som R=X+Y_{1k}+1; Y_{1komp} = 00110.

	011111
X	01101
$+Y_{1komp}$	+ 00110
=R	= 10100

1b) N=1; Z=0; V=1; C_5=0 $\Rightarrow$ C=1

1c) X=13 Y=25; R=20 (Kontroll: 13-25≠20); verkar rimligt ty C=1)
C anger att resultatet är fel vid tal utan tecken

1d) X= 13; Y= -7; R= -12 (Kontroll: 13-(-7) ≠ -12); verkar rimligt ty V=1)
V anger fel vid tal med tecken.

1e) Jämn paritet. Kodordet CB_{16} innehåller ett udda antal ettor vilket innebär fel. Kodorden motsvarar texten ”Kert”. Svenskt namn kan vara ”Kurt”. ASCII för ”e” = 1100101. ASCII för ”u” = 1110101. Troligen är b_5 i kodordet ändrat.

Upg 2

2a) Enligt tabellen är g(xyz) ≠ f(xyz)

xyz	y⊕z	xz	f	(x'+z')	(y'+z)	g
000	0	0	**0**	1	1	**1**
001	1	0	**1**	1	1	**1**
010	1	0	**1**	1	0	**0**
011	0	0	**0**	1	1	**1**
100	0	0	**0**	1	1	**1**
101	1	1	**1**	0	1	**0**
110	1	0	**1**	1	0	**0**
111	0	1	**1**	0	1	**0**

2b) Se blåa boken del 1 exempel 5.13

2c)

xyz	F
000	0
001	0
010	0
011	0
100	1
101	1
110	0
111	1

Disjunktiv minimal form:
f= (xy')+(xz)

Konjunktiv minimal form:
f=(x)(y'+z)

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

2d) Rita nätet

Upg 3 a)

xyz	F
000	0
001	0
010	0
011	1
100	1
101	1
110	1
111	0

Minimerat blir f=(xy')+(xz')+(x'yz)

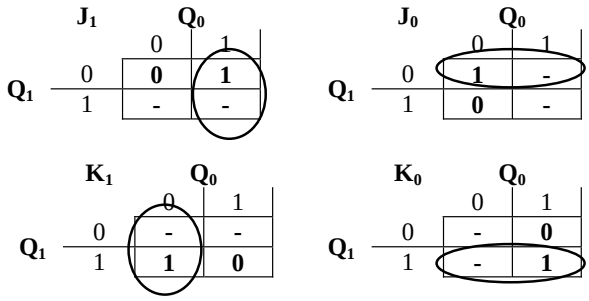
Rita nätet med NAND/NAND-logik

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

X Y Z

Upg 3b)

Detta Tillst	Nästa tillst		
q_1q_0	$q_1^+q_0^+$	J_1K_1	J_0K_0
00	01	0 -	1 -
01	11	1 -	- 0
10	00	- 1	0 -
11	10	- 0	- 1



Rita figur med följande insignaler till vipporna

$J_1 = q_0$ $K_1 = q_0'$	$J_0 = q_1'$ $K_0 = q_1$

Upg 4

4a)

State nr	RTN-beskrivning	Styrsignaler (=1)
0	PC→MA, PC+1→PC, S-1→S	OE_{PC}, LD_{MA}, IncPC, DecS
1	M→T	MR, LD_T
2	S→MA	OE_S, LD_{MA}
3	PC→M, T→R	OE_{PC}, MW, f_1, LD_R
4	R→PC, NF	OE_R, LD_{PC}, NF

4b)

0) Förbered för läsning av adressoperand i minnet, Öka PC med ett, Minska stackpekaren
1) Läs adressoperanden från minnet till register T
2) Förbered för att spara PC
3) Spara PC på stacken, Flytta adressoperanden till R
4) Och vidare till PC, Ny Fetch

Instruktionen är JSR $Adr

4c)

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→A, NF	MR, LD_A, NF

Upg 5

5a) PC är 8 bitar 2^8=256 adresser. Databussen är 8 bitar bred; 256*8=2048. Ant bitar totalt= 2048.

5b) Alla siffror = Hexsiffror! Först minskas X, och vi får 3 varv 17+16+15=42 som skrivs till adr F0

5c) ADCA och DEC påverkar båda C-flaggan vilket verkar konstigt. Å andra sidan kan det verka fel att utnyttja ADCA i snurran.

Upg 6a

```
IRQINIT  psha
         pshx
         movw     0,CLOCK        nollställ kolckan
         movb     0,CLOCK+2
         ldaa     #100           Avbrottsräknare
         staa     TEMP
         staa     IRQRES         nollställ avbrottsvippan
         ldx      #IRQ           avbrottsvektor
         stx      $fff2          (alt 3ff2)
         cli
         pulx
         pula
         rts
TEMP    rmb       1          Avbrottsräknare (100 IRQ = 1s)

IRQ      sta      IRQRES         nollställ avbrottsvippan
         dec      TMP            100 avbrott?
         bne      IExit            nej
         ldaa     #100           Avbrottsräknare
         staa     TEMP
 * Öka sekunder
         ldaa     CLOCK+2
         adda     #1
         daa
         staa     CLOCK+2
         cmpa     #60            Hel minut?
         bne      IExit            nej
 * Öka minuter
         clr      CLOCK+2          Nolla sekunder
         ldaa     CLOCK+1
         adda     #1
         daa
         staa     CLOCK+1
         cmpa     #60            Hel timme?
         bne      IExit            nej
 * Öka timmar
         clr      CLOCK+1          Nolla minuter
         ldaa     CLOCK
         adda     #1
         daa
         staa     CLOCK
         cmpa     #24            24 timmar?
         bne      IExit            nej
         clr      CLOCK
IExit   rti                        (Plus programhuvud och flödesplan)
```

Upg 6b

```
Start      LDX   #SegCode   Pekare till tabell
           LDAB  Inport     Läs inporten
           CMPB  #10        Giltigt värde
           BLO   OK         ..hoppa om JA
           LDAA  #Error     Skriv Error
           STAA  Utport
           BRA   End
OK         LDAA  B,X        Översätt indata till Segmentkod
           STAA  Utport     .. och skriv ut
End        BRA   Start
```

Upg 7 – för D-linjen

7. a) Synkront sekvensnät

b) Tillstånds- och utsignalstabellen

Ur kopplingen kan vi teckna de Booleska uttrycken för q_1^+ , q_0^+ och u:

$q_1^+ = x_2'x_1x_0'q_0$

$q_0^+ = x_2'x_1x_0$

$u = (x_2x_1'x_0'q_1)'$

Tillstånd		Insign	Nästa tillst	Utsign
	q_1q_0	$x_2x_1x_0$	$q_1^+q_0^+$	u
τ_0	0 0	0 1 0 0 1 1 1 0 0 övr	0 0 0 1 0 0 0 0	1 1 1 1
τ_1	0 1	0 1 0 0 1 1 1 0 0 övr	1 0 0 1 0 0 0 0	1 1 1 1
τ_2	1 0	0 1 0 0 1 1 1 0 0 övr	0 0 0 1 0 0 0 0	1 1 0 1
τ_3	1 1	0 1 0 0 1 1 1 0 0 övr	1 0 0 1 0 0 0 0	1 1 0 1

c) Funktionsbeskrivning i ASM-plan