EDA215 Digital och datorteknik Z (*Inte för E*)

EDA432 Digital och datorteknik IT

EDA311 Digital och datorteknik E

INN790 Digital och datorteknik GU

Gamla och nya kursen

Tentamen

Torsdag 24 augusti 2006, kl. 14.00 – 18.00 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 (max 50p):

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

Lösningar
anslås på kursens www hemsida (EDA432).

Betygslistan
anslås såsom anges på kursens hemsida (EDA432).

Granskning
Tid och plats anges på kursens hemsida (EDA432).

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 = 11101 och Y = 01001.

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) *Gray-kod*. Vi behöver 4 kodord där vi skall kunna räkna modulo 4 (0,1,2,3,0,1, etc).
Ange kodorden för siffrorna 0 till 3. Du skall använda 3-bitars Gray-kod. **(2p)**

f) Om det är möjligt, skriv det decimala talet -31 som ett 6-bitars tvåkomplementstal. Argumentera. **(1p)**

Uppgift 2 *Digitalteknik - Småfrågor*

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

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

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

d) Rita grindnätet för följande funktion: $f(x,y,z,w)=x'z+(w \oplus y+(x \cdot z \cdot w)')$
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=0 för $3 \leq (xyz)_2 \leq 5$. För alla andra värden på insignalen är f=1.
Insignalerna bildar ett binärtal $(xyz)_2 \in [0,7]$.

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 **00,11,01,10,** 00,11,01,10 etc. på de två utsignalerna Q_1Q_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	
1	M → MA	
2	M → T	
3	A-T-C → R, Flaggor → CC	
4	R → A, NF	

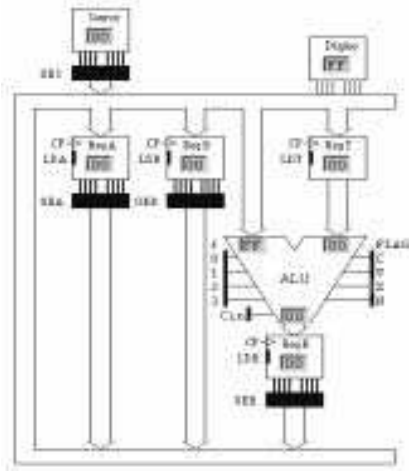
b) Förklara i ord vad instruktionen ovan utför i varje klockcykel.
Ange sedan instruktionen med assemblerspråk för FLEX-processorn (Ex: **LDA Adr**). **(2p)**

c) Ange de styrsignaler som krävs för att utföra operationerna enligt nedanstående RTN-beskrivning:

RTN-beskrivning: $5 \cdot B \rightarrow B$

Använd den enkla datavägen till höger och ge ditt svar i tabellform liknande den ovan.

Förutsätt att register B innehåller data som skall multipliceras med 5. Register A får inte ändras. Använd så få tillstånd som möjligt. **(4p)**



Uppgift 5 *Småfrågor rörande FLEX-processorn*

a) Ett 24-bitars tal är lagrad på adresserna 05, 06 och 07 med mest signifikanta byten på adress 05. Skriv en instruktionssekvens som negerar (två-komplementerar) detta 24-bitars tal. Resonera! **(2p)**

b) Nedan visas ett assemblerprogram. Ange maskinkoden för BPL-instruktionen.
Det skall klart framgå hur du beräknat instruktionens offset

Assemblerprogram:

```
      ORG  $30
Start LDB  #$0C
LOOP  LDA  ,X+
      STA  $FE
      DECB
      BPL  LOOP
      NOP
```

(3p)

c) Studera programmet ovan (Uppgift 5b). När du testkör detta i en FLEX-dator i labbet fungerar det inte som tänkt. Orsaken är att BPL-instruktionen inte fungerar korrekt i denna labdator. Däremot fungerar alla andra instruktioner korrekt.
Skriv om programmet så att det fungerar som tänkt (du får alltså inte använda BPL-instruktionen) **(3p)**

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 (Inport, 8 bitar) utför en addition och skriver summan till sifferindikatorn.

Från den 8-bitars inporten läses två 4-bitars binära tal P och Q samtidigt. P hittas på $[b_7,b_4]$ och Q hittas på $[b_3,b_0]$. Summan skall placeras i $[b_3,b_0]$ för att omvandlas till segmentkod och skrivas till sifferindikatoren. Om summan P+Q är större än nio skall ett E (ERROR) skrivas ut.

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

```
Inport    EQU  qqqq          Adress för inport
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. 2006-08-24

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} = 10110.

```
                111111
 X               11101
+Y1komp        + 10110
=R             = 10100
```

1b) N=1; Z=0; V=0; C_5=1 ⇒ C=0

1c) X=29 Y=9; R=20 (Kontroll: 29-9=20); verkar rimligt ty C=0)
C=1 anger att resultatet är fel vid tal utan tecken

1d) X= -3; Y= 9; R= -12 (Kontroll: -3-9 = -12); verkar rimligt ty V=0)
V anger fel vid tal med tecken.

1e) Studera tabell 2.2, mittersta kolumnen, i blåa boken. För att erhålla en reflekterande kod används kodorden 011, 010, 110 och 111.

1f) 100001_2. Det går att representera -31 på 6 bitar ty 2^6=64 och vilket ger talområdet [-32,31]

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**	0	1	**0**
010	1	0	**1**	1	0	**0**
011	0	0	**0**	0	1	**0**
100	0	0	**0**	1	1	**1**
101	1	1	**1**	1	1	**1**
110	1	0	**1**	1	0	**0**
111	0	1	**1**	1	1	**1**

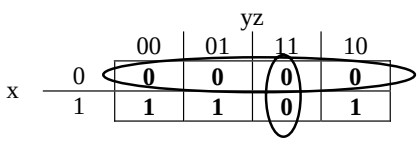
2b) Se blåa boken del 1 exempel 5.12

2c)

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

Disjunktiv normal form:
f= (xy'z')+(xy'z)+(xyz')

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



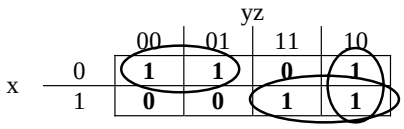
2d) Rita nätet

Upg 3 a)

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

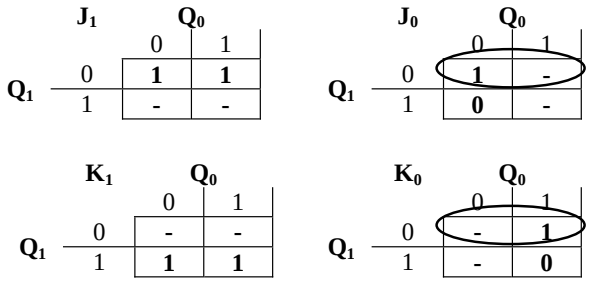
Minimerat blir f=(x'y')+(xy)+(yz')

Rita nätet med NAND/NAND-logik



Upg 3b)

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



Rita figur med följande insignaler till vipporna

$J_1 = 1_0$ $K_1 = 1$	$J_0 = q_1'$ $K_0 = q_1'$

Upg 4

4a)

State	RTN-beskrivning	Styrsignaler (=1)
0	PC→ MA, PC+1 → PC	OE_{PC}, LD_{MA}, IncPC,
1	M → MA	MR, LD_{MA}
2	M → T	MR, LD_T
3	A-T-C → R, Flaggor → CC	OE_A, f_3, f_2, g_1, g_0, LD_R, LD_{CC},
4	R → A, NF	OE_R, LD_A, NF

4b)

0) Förbered för läsning av adressoperand i minnet, Öka PC med ett, Minska stackpekaren
1) Läs adressoperanden från minnet till register MA
2) Läs dataoperanden från minnet till register T
3) Utför subtraktion med Carry, spara resultatet i register R och påverka flaggbitarna
4) Flytta resultatet till register A, Ny Fetch

Instruktionen är SBCA $Adr

4c)

State nr	RTN-beskrivning	Styrsignaler (=1)
0	B→T, 2B→R	OE_B, LD_T, f_3, f_1, f_0, LD_R,
1	2R→R	OE_R, f_3, f_1, f_0, LD_R,
2	R+T→R	OE_R, f_3, f_1, LD_R,
3	R→A,	OE_R, LD_A,

Upg 5

5a)

```
        COM   $05    Invertera högsta
        COM   $06    Invertera mellersta
        NEG   $07    2-komplementera lägsta
        BCC   SLUT   Hoppa om klart
        INC   $06    Öka mellersta
        BCC   SLUT   Hoppa om klart
        INC   $05    Öka mellersta
SLUT    NOP
```

5b)

Adr	Kod			
			ORG	$30
30		Start	LDB	#$0C
31				
32		LOOP	LDA	,X+
33			STA	$FE
34				
35			DECB	
36	5C		BPL	LOOP
37	FA			
38			NOP	

TillAdr – FrånAdr = Offset
$32 - $38 = $FA

Maskininstruktion: $5C $FA

5c)

```
                ORG   $30
     Start      LDB   #$0C
     LOOP       LDA   ,X+
                STA   $FE
                DECB
                BMI   KLART
                BRA   LOOP
     KLART      NOP
```

Upg 6b

```
Start       LDX   #SegCode   Pekare till tabell
            LDAB  Inport     Läs inporten
            TFR   B,A        Kopiera
            LSRA             Skifta fram P
            LSRA
            LSRA
            LSRA
            ANDB  #$0F       Maska fram Q
            ABA              Summan
            CMPA  #10        Giltigt värde
            BLO   OK         ..hoppa om JA
            LDAB  #Error     Skriv Error
            STAB  Utport
            BRA   End
OK          LDAB  A,X        Översätt indata till Segmentkod
            STAB  Utport     .. och skriv ut
End         BRA   Start
```