CHALMERS

Institutionen för data- och informationsteknik
Avdelningen för datorteknik

TENTAMEN

KURSNAMN	**Digital- och datorteknik**
PROGRAM:	**Mekatronikingenjör (samt data- och elektroingenjör)** **Åk 1/ lp 2**
KURSBETECKNING	**LEU431**
EXAMINATOR	**Lars-Eric Arebrink**
TID FÖR TENTAMEN	**2011-12-12 kl 8.30 – 12.30**
HJÄLPMEDEL	**Av institutionen utgiven** **”Instruktionslista för FLEX-processorn” (INS1)** **Tabellverk eller miniräknare får ej användas.**
ANSV LÄRARE: **Besöker tentamen**	**Lars-Eric Arebrink, tel. 772 5718** **vid flera tillfällen**
ANSLAG AV RESULTAT	**När rättningen är färdig anslås resultatet med anonyma koder och tid för granskning på kursens hemsida.** **Lägg din anonyma kod på minnet! Den används när resultatet anslås.**
ÖVRIG INFORM. **BETYGSGRÄNSER.** **SLUTBETYG**	**Tentamen omfattar totalt 60 poäng.** **Onödigt komplicerade lösningar kan ge poängavdrag. Svar på uppgifter skall motiveras.** **Betyg 3: 24 poäng** **Betyg 4: 36 poäng** **Betyg 5: 48 poäng** **För slutbetyg 3, 4 eller 5 på kursen fordras betyg 3, 4 eller 5 på tentamen och godkända laborationer.**

1. I uppgift a-h nedan används 4-bitars tal X, Y, S och D. X = 1100 och Y = 0101.

a) Vilket talområde måste X, Y, S och D tillhöra om de tolkas som tal utan tecken? **(1p)**

b) Vilket talområde måste X, Y, S och D tillhöra om de tolkas som tal med tecken? **(1p)**

c) Visa med penna och papper hur räkneoperationen S = X + Y utförs i en 4-bitars ALU. **(1p)**

d) Vilka värden får flaggbitarna N, Z, V och C vid räkneoperationen i c)? **(1p)**

e) Visa med penna och papper hur räkneoperationen D = X – Y utförs i en 4-bitars ALU. **(1p)**

f) Vilka värden får flaggbitarna N, Z, V och C vid räkneoperationen i e)? **(1p)**

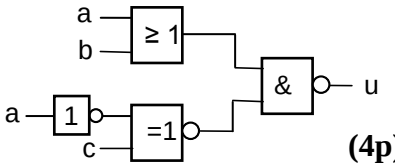
g) Tolka bitmönstren X, Y, S och D som tal *utan* tecken och ange deras decimala motsvarighet. Avgör och motivera med hjälp av flaggorna om resultaten S och D är korrekta eller felaktiga. **(1p)**

h) Tolka bitmönstren X, Y, S och D som tal *med* tecken och ange deras decimala motsvarighet. Avgör och motivera med hjälp av flaggorna om resultaten S och D är korrekta eller felaktiga. **(1p)**

i) Skriv talen $X = 351_{10}$ och $Y = -594_{10}$ med 10-komplementrepresentation och genomför additionen X + Y med 10-komplementaritmetik. Hur många decimala sifferpositioner behövs? Hur skall man tolka resultatet? **(2p)**

j) I ett packat flyttal enligt standarden IEEE 754 används en sk ”fraction” istället för mantissan. Förklara orsaken till detta! **(1p)**

k) Ge ett minimalt boolesk PS-uttryck för u i grindnätet till höger.



(4p)

2. En boolesk funktion f(a,b,c,d) har karnaughdiagrammet till höger.

Realisera funktionen med så få grindar som möjligt. NAND-grindar med valfritt antal ingångar, XOR-grindar och NOT-grindar får användas. Endast insignalerna a, b, c och d finns tillgängliga.

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

(5p)

3.

a) Ett synkront sekvensnät skall ha en insignal x och en utsignal u. Utsignalen u skall ges värdet "1" under ett bitintervall om de 3 senaste x-värdena har varit lika.

Exempel: σ_x = ...01111100001100111000111101 1...

σ_u = ...?00111001100000010010011000...

Utsignalen skall som i exemplet ges värdet "1" när den sista biten i en korrekt insignalsekvens anländer på x-ingången och behålla värdet "1" så länge x har kvar sitt värde under detta bitintervall.

Rita en tillståndsgraf för sekvensnätet. Hur många vippor skulle minst krävas vid realiseringen?

(4p+1p)

b) Realisera en räknare med räknevillkoret x och räknesekvensen q_1q_0:

x = 0: 00, 01, 10, 11, 00, …

x = 1: 00, 11, 10, 01, 00, …

JK-vippor, NAND-grindar med valfritt antal ingångar, XOR-grindar och NOT-grindar får användas.

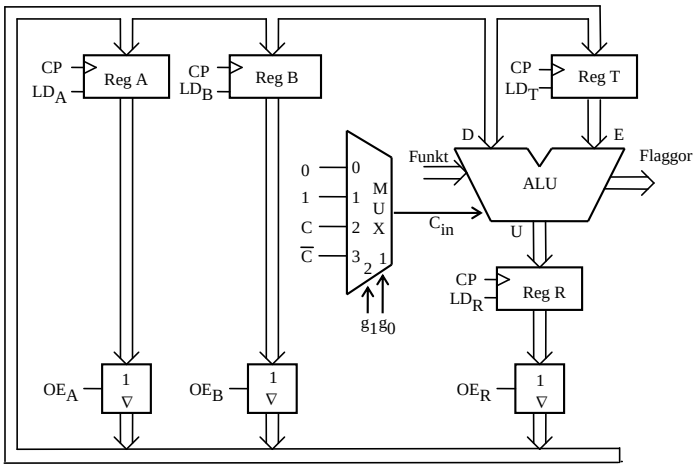
(5p)

4. I datavägen till höger innehåller register A och B från början värdena 5 resp. 9. Därefter ges styrsignaler och klockpulser enligt tabellen nedan.

Rita av tabellen!

Fyll sedan i RTN-beskrivning samt hexadecimalt registerinnehåll för alla klockpulsintervall. Vid laddning av ett register skall det nya innehållet ”synas” på raden efter laddningen i tabellen.

Samtliga funktioner ALU:n kan utföra framgår av bilaga 1.



CP	Styrsignaler (=1)	RTN	Reg A	Reg B	Reg T	Reg R
1	OE_A, LD_T		05	09	?	?
2	OE_B, f_3, f_2, LD_R					
3	OE_R, f_3, f_1, f_0, LD_R, LD_T					
4	OE_R, f_3, f_1, LD_R					
5	OE_A, LD_T					
6	OE_R, f_3, f_2, g_0, LD_R					
7	OE_R, f_3, f_2, g_0, LD_R					
8	OE_R, LD_B					
9	?	?				

(5p)

5. Figur 1 i bilaga 3 visar hur datorn FLEX är uppbyggd. Bilaga 1 visar hur ALU’ns funktion väljs med styrsignalerna f_3 - f_0 och C_{in}.

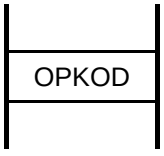
I tabellen nedan visas styrsignalerna för de olika tillstånden i EXECUTE-sekvensen för en av FLEX-processorns instruktioner.

State	S-term	RTN-beskrivning	Styrsignaler (=1)
Q_5	$Q_5 \cdot I_{xx}$		OE_{PC}, LD_{MA}, IncPC
Q_6	$Q_6 \cdot I_{xx}$		MR, LD_T
Q_7	$Q_7 \cdot I_{xx}$		OE_X, f_3, f_1, LD_R
Q_8	$Q_8 \cdot I_{xx}$		OE_R, LD_{MA}
Q_9	$Q_9 \cdot I_{xx}$		MR, LD_{MA}
Q_{10}	$Q_{10} \cdot I_{xx}$		MR, LD_B, NF

a) Rita en tabell med ”State”- och RTN-kolumner enligt ovan och fyll i RTN-beskrivningen. Förklara vilken assemblerinstruktion som beskrivs. **(2p)**

b) Instruktionen ”Move byte” nedan skall implementeras för FLEX-processorn med hjälp av styrenheten med fast logik. Instruktionen skall bestå av endast operationskoden. Den skall läsa dataordet på adressen som finns i X-registret och skriva det på stacken. Efter skrivningen på stacken skall innehållet i X-registret ökas med 1. Register A, B eller CC får ej påverkas. Operationskoden FA_{16} skall användas.

MOVB 1,X+,1,-SP RTN: $SP-1 \rightarrow SP$
$M(X) \rightarrow M(SP)$
$X+1 \rightarrow X$



Gör en tabell liknande tabellen ovan för den efterfrågade EXECUTE-sekvensen. **(4p)**

6. Besvara kortfattat följande frågor rörande FLEX-processorn.

a) Innan de villkorliga hoppen BHI och BGT utförs brukar man påverka flaggorna med en subtraktion. Förklara vad som händer för vardera hoppet om subtraktionen före är 85_{16} - 67_{16}. **(2p)**

b) Vad händer om man låter FLEX-processorn utföra en instruktion där man har glömt att sätta styrsignalen NF = 1 i sista tillståndet i ”EXECUTE”, om man använder styrenheten med fast logik? **(2p)**

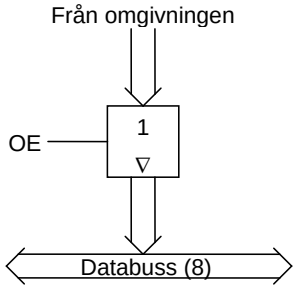
c) Förklara vad som händer under processorns FETCH-fas. **(2p)**

d) Översätt instruktionssekvensen till höger till maskinkod och visa hur den placeras i minnet. Det skall framgå hur offset för branch-instruktionen beräknas. **(3p)**

```
        ORG     $80
        LDAA    #-10
LOOP    LDAB    1,X+
        NEGB
        ANDB    #15
        STAB    $FE
        INCA
        BLE     LOOP
        NOP
```

e) Hur lång tid tar instruktionssekvensen i d) att köra om FLEX-processorn klockas med frekvensen 1MHz? **(2p)**

f) FLEX-datorn som visas i bilaga 3 saknar inportar. Den skall nu kompletteras med en inport på adressen FE_{16}. Principen för en inport visas i figuren till höger. Konstruera ett grindnät som bildar signalen OE_{FE}. Standardgrindar med valfritt antal ingångar får användas.



(2p)

7. I minnet i ett datorsystem med FLEX-processorn finns en tabell med ett antal 8-bitars dataord som är sammansatta av två fyrabitars tal utan tecken i bit 0-3 och bit 4-7 enligt figuren till höger. Skriv en subrutin, NIBADD, i assemblerpråk för FLEX-processorn som för varje dataord i tabellen adderar talet i bit 0-3 med talet i bit 4-7 och skriver tillbaka summan (5-bitar) i bit 0-4, med bit 5-7 nollställda.

TAL(4)	TAL(4)
TAL(4)	TAL(4)
TAL(4)	TAL(4)
TAL(4)	TAL(4)

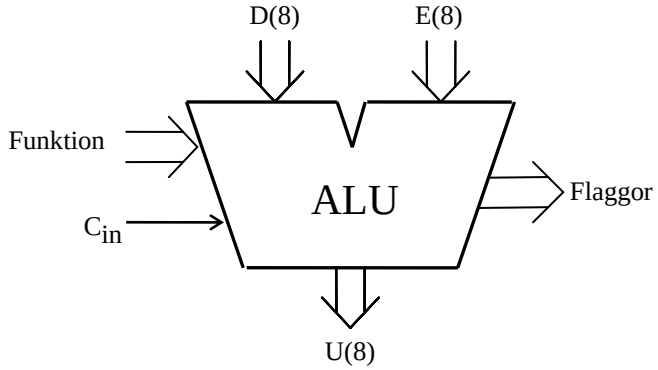
Vid anrop av subrutinen finns antalet 8-bitars dataord i tabellen i B-registret och startadressen till tabellen i X-registret.

Endast register CC får vara förändrat vid återhopp från subrutinen. För full poäng på uppgiften skall programmet vara korrekt radkommenterat.

(6p)

Bilaga 1

ALU:ns funktion



ALU:ns **logik-** och **aritmetikoperationer** på indata **D** och **E** definieras av ingångarna **Funktion (F)** och $\mathbf{C_{in}}$ enligt tabellen nedan. **F** = $\mathbf{(f_3, f_2, f_1, f_0)}$.

I kolumnen Operation förklaras hur operationen utförs.

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	bitvis nollställning	0
1 1 1 0	bitvis nollställning	0
1 1 1 1	bitvis ettställning	FFH

Carryflaggan (C) innehåller minnessiffran ut (carry-out) från den mest signifikanta bitpositionen (längst till vänster) om 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 om en aritmetisk operation ger "overflow" enligt reglerna för 2-komplementaritmetik.

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

Zeroflaggan (Z) visar om 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.

I tabellen ovan avser "**+**" och "**–**" **aritmetiska operationer**. Med t ex $\mathbf{D_{1k}}$ menas att samtliga bitar i **D** inverteras.

Bilaga 2

Assemblerspråket för FLEX-processorn.

Assemblerspråket använder sig av mnemoniska beteckningar liknande dem som processorkonstruktören MOTOROLA (FREESCALE) specificerat för maskininstruktioner för mikroprocessorerna 68XX och instruktioner till assemblatorn, s k pseudoinstruktioner eller assemblatordirektiv. Pseudoinstruktionerna listas i tabell 1.

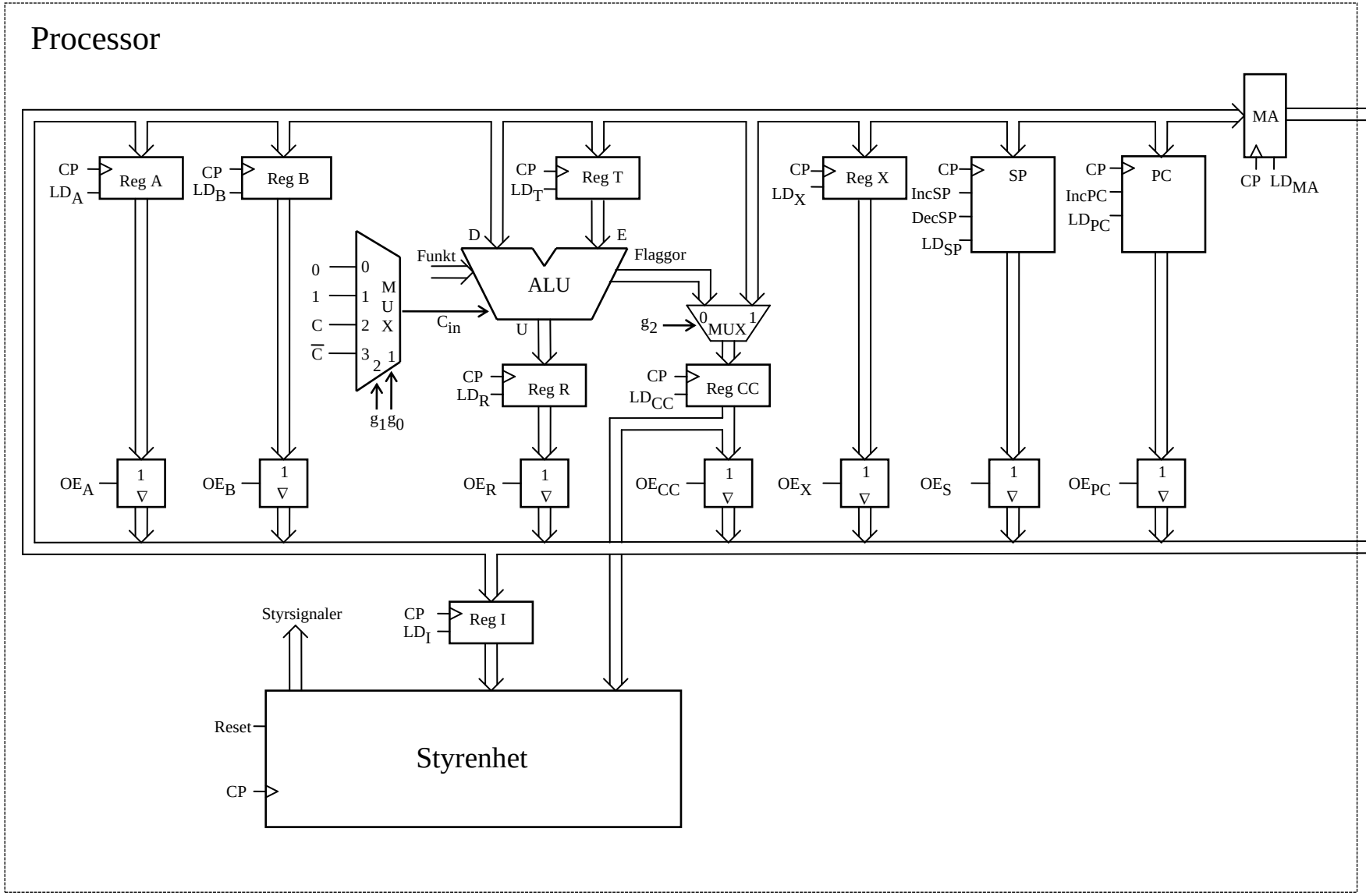
Tabell 1

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 adressen L. (RMB för Reserve Memory Bytes)
L EQU N	Ger symbolen L konstantvärdet N. (EQU för EQUates = beräknas till)
L FCB N1,N2	Avsätter en byte för varje argument i följd i minnet. Respektive byte ges konstantvärdet N1, N2 etc. Följden placeras med början på adressen L. (FCB för Form Constant Byte)
L FCS ”ABC”	Avsätter en byte för varje tecken i teckensträngen ”ABC” i följd i minnet. Respektive byte ges ASCII-värdet för A B C, etc. Följden placeras med början på adressen L. (FCS för Form Character String)

Tabell 2 7-bitars ASCII

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

Bilaga 3



Figur 1. Datorn FLEX.