CHALMERS

Institutionen för data- och informationsteknik
Avdelningen för datorteknik

TENTAMEN

KURSNAMN	**Digital- och datorteknik**
PROGRAM:	**Data- och elektroingenjör** **Åk 1/ lp 1**
KURSBETECKNING	**LEU431**
EXAMINATOR	**Lars-Eric Arebrink**
TID FÖR TENTAMEN	**2011-10-17 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.	**Tentamen omfattar totalt 60 poäng.** **Onödigt komplicerade lösningar kan ge poängavdrag. Svar på uppgifter skall motiveras.**
BETYGSGRÄNSER.	**Betyg 3: 24 poäng** **Betyg 4: 36 poäng** **Betyg 5: 48 poäng**
SLUTBETYG	**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 7-bitars tal X, Y, S och D. X = 0110101 och Y = 1101011.

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

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

c) Visa med penna och papper hur räkneoperationen S = X + Y utförs i en 7-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 7-bitars ALU. **(1p)**

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

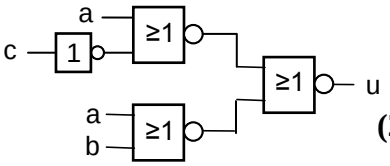
g) Tolka bitmönstren X, Y, S och D som tal *utan* tecken och ange deras decimala motsvarighet. Avgör och motivera med hjälp av flaggorna om resultaten S och D är korrekta eller felaktiga. **(1p)**

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) Hur många olika booleska funktioner f(x,y) finns det? **(1p)**

j) Översätt (packa) det decimala talet -75,75 till ett 32-bitars binärt flyttal enligt flyttalsstandarden IEEE 754-1985 (dvs 23 bitar av mantissan). Ge också det packade flyttalet på hexadecimal form. **(2p)**

k) Ge ett ”minimalt” boolesk SP-uttryck för u i grindnätet till höger. **(2p)**



2. Det booleska uttrycket f(a,b,c) = abc + a’b’c + abc’ + a’bc’ beskriver en boolesk funktion.

Konstruera ett grindnät med så få grindar som möjligt, som realiserar den booleska funktionen. NOR-grindar med valfritt antal ingångar, XOR-grindar och NOT-grindar får användas. Endast insignalerna a, b och c finns tillgängliga.

(4p)

3.

a) Ett synkront sekvensnät skall ha en insignal x och en utsignal u. Utsignalen u skall ges värdet "1" under ett bitintervall för varje insignalsekvens som består av två nollor följda av en etta och tre nollor hos x.

Exempel: σ_X = ...00100010001100010000100010100 0...

σ_U = ...??0001000100000000100001000000...

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? **(3p+1p)**

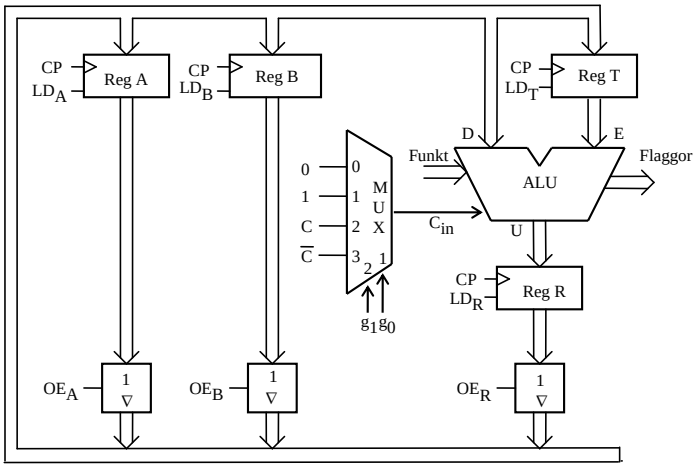
b) Realisera en autonom räknare med räknesekvensen $q_2q_1q_0$: 000, 010, 100, 110, 101, 011, 001, 000. T-vippor, NAND-grindar med valfritt antal ingångar och NOT-grindar får användas. **(6p)**

4. Ge RTN-beskrivning och styrsignaler för de tillstånd krävs för att utföra operationen enligt nedanstående RTN-beskrivning:

RTN-beskrivning: $5 \cdot A - 7 \cdot (B + 1) \rightarrow A$
(Aritmetisk multiplikation avses)

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

Förutsätt att register A och B från början innehåller de data som skall behandlas enligt uttrycket ovan och att innehållen i register R och T är okända. Register B får inte ändras. Bortse från risken för overflow. Använd så få tillstånd som möjligt. Samtliga funktioner ALU:n kan utföra framgår av bilaga 1.



(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 i EXECUTE-sekvensen för en av FLEX-processorns instruktioner.

State	S-term	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_5	$Q_5 \cdot I_{xx}$		OE_{PC}, LD_{MA}, IncPC
Q_6	$Q_6 \cdot I_{xx}$		MR, LD_{MA}
Q_7	$Q_7 \cdot I_{xx}$		MR, LD_T
Q_8	$Q_8 \cdot I_{xx}$		OE_A, f_3, f_2, g_1, g_0, LD_R, LD_{CC}
Q_9	$Q_9 \cdot I_{xx}$		OE_R, LD_A, NF

NF i tabellens sista rad anger att nästa tillstånd (state) skall vara det första i FETCH-sekvensen.

a) Fyll i RTN-kolumnen i tabellen och förklara vilken assemblerinstruktion som beskrivs. **(2p)**

b) Instruktionen BGE tar 5 klockcykler att utföra enligt instruktionslistan. En ny implementering av instruktionen skall göras som utförs på färre klockcykler när hoppvillkoret inte är uppfyllt. Styrenheten med fast logik skall användas och operationskoden $F1_{16}$ skall användas.

Gör en tabell liknande tabellen ovan för den nya EXECUTE-sekvensen.

(4p)

6. Besvara kortfattat följande frågor rörande FLEX-processorn.

a) Assemblatorn kan hantera uttryck som innehåller additioner eller subtraktioner av symboler och konstanter. Den borde dock protestera när man assemblerar följande rader. Förklara varför!

```
SYMBOL1  EQU  $15
SYMBOL2  EQU  SYMBOL1+SYMBOL3
SYMBOL3  EQU  $25
```
(2p)

b) Förklara varför man inte kan hoppa till en subrutin med JMP- eller BRA-instruktioner. **(2p)**

c) Förklara på vilket sätt X-registret kan användas! **(2p)**

d) För vilka värden på talet W ($0 \leq W \leq 255$) utförs hoppet nedan?

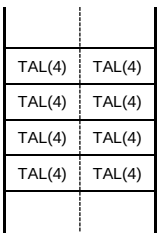
```
LDAA    #W
COMA
CMPA    #$50
BGT     Hopp
```
(3p)

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

f) Hur lång tid tar instruktionssekvensen i e) att köra om FLEX-processorn klockas med frekvensen 1MHz? **(3p)**

```
        ORG   $60
        LDX   #NEXT
        LDAA  -4,X
*
LOOP1   LDAB  -3,X
*
LOOP2   DECB
        NOP
        BNE   LOOP2
*
        INCA
        BMI   LOOP1
*
        BRA   NEXT
*
TAB     FCB   0,-2,10,-4,20
*
NEXT    NOP
```

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 i assemblerpråk för FLEX-processorn som ökar alla talen i bit 0-3 i tabellen med 5, men lämnar talen i bit 4-7 oförändrade. Antalet ”overflow” som inträffar vid ökningen med 5 skall räknas och detta antal skall finnas i B-registret vid återhopp. ”Overflow” avser här 4-bitars tal utan tecken.

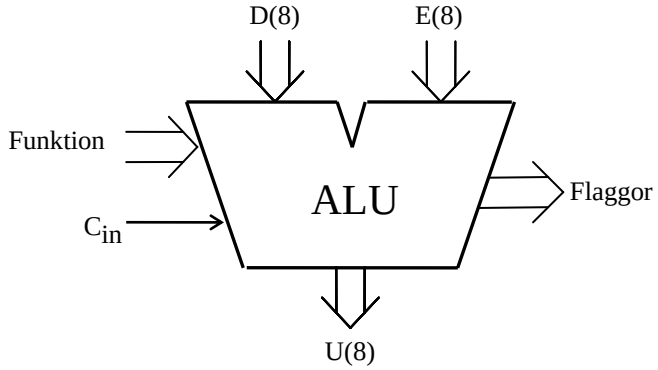
Före anrop av subrutinen skall huvudprogrammet lägga in antalet 8-bitars dataord i tabellen i B-registret och startadressen till tabellen i X-registret.

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

(7p)

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.

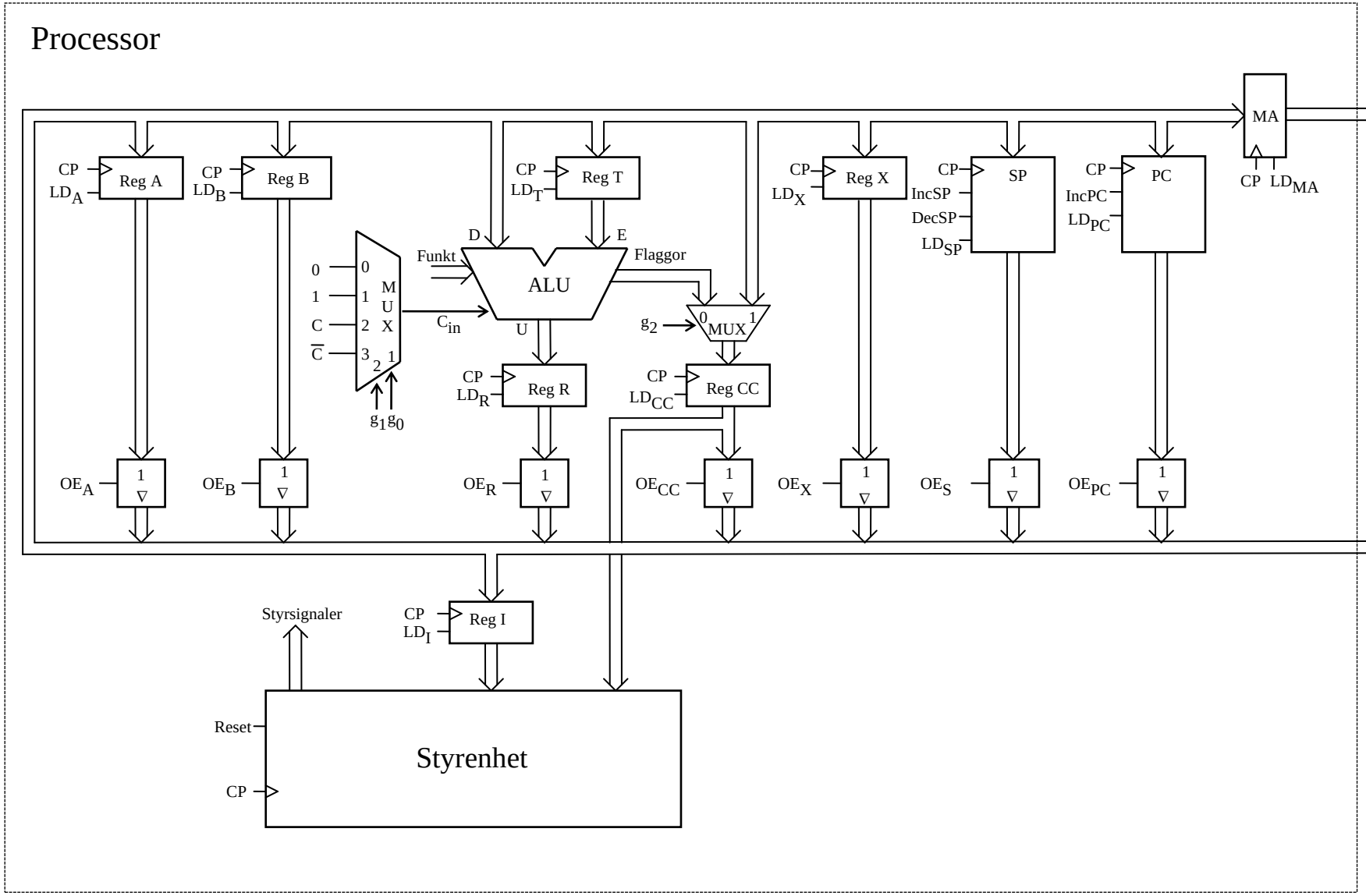
Tabell 1

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 Reseve 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.