CHALMERS

Institutionen för data- och informationsteknik
Avdelningen för datorteknik

TENTAMEN

KURSNAMN	**Digital- och datorteknik**
PROGRAM:	**Data-, elektro- och mekatronikingenjör åk 1/ lp 1 och 2**
KURSBETECKNING	**LEU431**
EXAMINATOR	**Lars-Eric Arebrink**
TID FÖR TENTAMEN	**2013-12-16 kl 14.00 – 18.00**
HJÄLPMEDEL	**Av institutionen utgiven ”Instruktionslista för FLEX- eller FLIS-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.** **Observera! Rättningen kommer inte att vara klar före anmälningstidens utgång (23/12) för omtentamen 16 januari. Om du tänker tentera i januari måste du därför anmäla dig innan du vet resultatet på denna tentamen. Du kan sedan avanmäla dig om du vill.**
ÖVRIG INFORM.	**Tentamen omfattar totalt 60 poäng.** **Den som är inskriven före HT 2012 kan använda FLEX-processorn istället för FLIS-processorn vid lösning av assembleruppgifter!** **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 8-bitars tal X, Y, S och D. Aritmetiska operationer utförs på samma sätt och flaggor sätts på samma sätt som i ALU'n i bilaga 1. För tal med tecken används 2k-representation.

$X = 1010\ 0101_2$ och $Y = 0101\ 1011_2$.

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 papper och penna hur räkneoperationen S = X + Y utförs i en 8-bitars ALU. **(1p)**

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

e) Visa med papper och penna hur räkneoperationen D = X – Y utförs i en 8-bitars ALU. **(1p)**

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

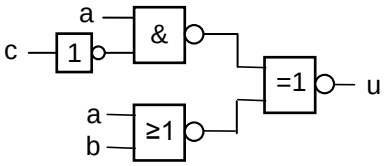
g) Tolka bitmönstren X, Y, S och D som tal *utan* tecken och ange deras decimala motsvarighet. Avgör och motivera med hjälp av flaggorna om resultaten S och D är korrekta eller felaktiga. **(1p)**

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) Översätt (packa) det decimala talet 125,125 till ett 32-bitars flyttal enligt flyttalsstandarden IEEE 754 (dvs 23 bitar av mantissan). Ge det packade flyttalet på binär och hexadecimal form. **(2p)**

2.

a) Markera det korrekta minimala PS-uttrycket för u i grindnätet till höger på svarsblanketten på sista sidan!



(3p)

b) Markera ett lämpligt uttryck på svarsblanketten för den booleska funktionen f, vars karnaughdiagram visas till höger.

NAND-grindar med valfritt antal ingångar, XOR-grindar och NOT-grindar skall användas. Rutor med - representerar "dont care"-termer som får användas vid behov. Kretsrealiseringen behöver ej visas.

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

(4p)

3.

a) Konstruera en T-vippa med hjälp av en D-vippa och standardgrindar. **(2p)**

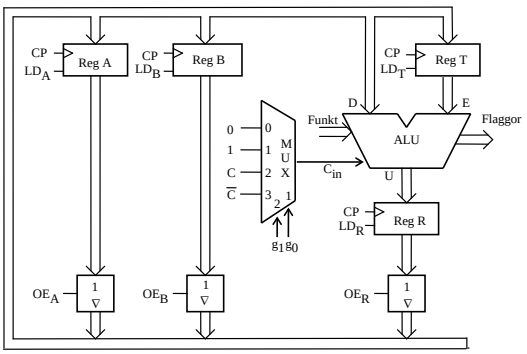
b) Konstruera en 3-bitars autonom räknare. (Räknesekvens: 0, 1, 3, 2, 6, 7, 5, 4, 0,…)
Numrera tillstånden $q_2q_1q_0$. T-vippor, AND-grindar med valfritt antal ingångar, XOR-grindar och NOT-grindar får användas. **(6p)**

4. I datavägen till höger innehåller register A(8) och B(8) från början värdena 200_{10} resp. 100_{10} på binär form. Därefter ges styrsignaler och klockpulser enligt tabellen på svarsblanketten.

Komplettera tabellen med RTN-beskrivning samt hexadecimalt registerinnehåll för alla klockpulsintervall.

Vid laddning av ett register skall det nya innehållet "synas" på raden efter laddningen i tabellen.

ALU-funktioner: Se bilaga 1!
Svarsblanketten finns på sista sidan.



(4p)

5. Figuren på sidan 45 i INS1 visar hur FLIS-datorn är uppbyggd. På sidorna 43 och 44 i INS1 visas hur ALU'ns funktion väljs med styrsignalerna f_3 - f_0 och C_{in}.

I tabellerna på svarsblanketten på sista sidan visas styrsignalerna för två olika EXECUTE-sekvenser för samma maskininstruktion, en för FLIS- och en för FLEX-processorn.

a) Komplettera den övre (FLISP) eller den undre (FLEX) tabellen med RTN-beskrivning! Förklara vilken assemblerinstruktion som beskrivs!

Som alternativ kan du alltså använda FLEX-datorn som visas i bilaga 3 med ALU'n i bilaga 1 och den undre tabellen istället för motsvarande för FLIS-datorn. **(2p)**

b) Instruktionen "Add memory bytes" nedan skall implementeras för FLIS- eller FLEX-processorn med hjälp av styrenheten med fast logik. Instruktionen består av två ord. Den utför addition av dataorden M(X) och M(Adr). Summan placeras som M(X). Flaggorna skall påverkas som vid en vanlig addition. Efter additionen och lagringen i minnet ökas innehållet i X-registret med 1.

Register som är synliga för programmeraren får ej påverkas, förutom X och CC. Operationskoden DF_{16} skall användas.

ADMB ,X+,Adr RTN: $M(X) + M(Adr) \rightarrow M(X)$, Flags $\rightarrow$ CC
$X + 1 \rightarrow X$

Komplettera tabellen på svarsblanketten med RTN-beskrivning och styrsignaler för den efterfrågade EXECUTE-sekvensen. **(4p)**

6. Besvara kortfattat följande frågor rörande FLIS- eller FLEX-processorn.

a) Vad händer när man utför en instruktion där man har glömt att aktivera signalen NF i sista tillståndet i EXECUTE? **(2p)**

b) Före det villkorliga hoppet "BPL Adr" i ett program utförs en subtraktion "$W - 97_{16}$" som påverkar flaggorna. För vilka värden på talet W utförs hoppet? 8-bitars tal $[0,255]_{10}$ används. **(3p)**

c) Översätt subrutinen till höger till maskinkod på hexadecimal form och visa hur den placeras i minnet. Det skall framgå hur offset för branch-instruktionerna beräknas. **(3p)**

```
CONST   EQU     5
;
        ORG     $80
;
DELAY   PSHA
        PSHA
;
YLOOP   LDA     #CONST
ILOOP   DECA
        BNE     ILOOP
;
        DEC     0,SP
        BNE     YLOOP
;
        PULA
        PULA
        RTS
```

d) Subrutinen i c) anropas i huvudprogrammet nedan.

```
        ORG     $10
        LDSP    #$FA
        LDA     #10
        JSR     DELAY
STOP    BRA     STOP
```

Hur lång tid tar subrutinen att köra från och med JSR till och med RTS om FLIS-(FLEX)-processorn klockas med frekvensen 1MHz?

(3p)

7. Denna uppgift innebär att du skall skriva ett program i assemblerspråk för FLIS- eller FLEX-processorn som liknar programmet "Rinnande ljus" i laboration 4.

Istället för att tända en ensam ljusdiod som "flyttar sig" åt ena hållet på utenheten "LED" skall programmet tända två ljusdioder som "flyttar sig" åt var sitt håll. Det innebär att bitmönstret vid start, 10000001, även kommer att vara bitmönstret vid slut, dvs. när en tänd ljusdiod har "vandrat" hela vägen från vänster till höger eller tvärtom på rampen. När en tänd ljusdiod försvinner utanför kanten skall den direkt återkomma på andra sidan av ljusdiodrampen.

För att få en lagom hastighet på förloppet får du använda en färdig subrutin DELAY, som fördröjer 200 ms. Subrutinen DELAY ändrar inga register eller flaggor. Programmet, som innehåller en evighetsslinga, beskrivs punktvis nedan.

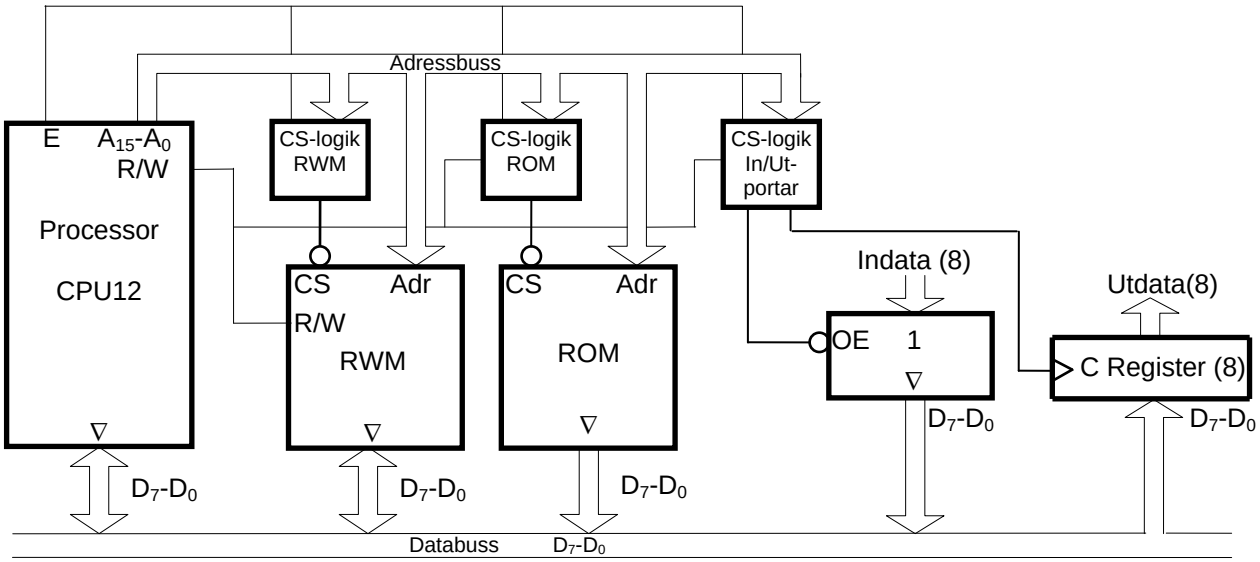
Programmet skall i tur och ordning:

1) Utföra nödvändiga initieringar bl. a. nollställa utport FC_{16}, som är kopplad till utenheten "LED".
2) Läsa av inport FB_{16}. Om bit 7 är nollställd utföra 2) igen, annars skall det fortsätta till 3).
3) Placera bitmöstret 10000001 i register A.
4) Mata ut bitmöstret i register A på utport FC_{16}.
5) Utföra en fördröjning på 200 ms (DELAY).
6) Läsa av inport FB_{16}. Om bit 7 är nollställd utföra 6) igen. Annars fortsätta till 7).
7) Placera nästa bitmönster i register A och fortsätta till 4).

Programmets startadress skall vara 10_{16}. Subrutinen DELAY finns på adressen 80_{16}.

För full poäng skall programmet vara ”korrekt” radkommenterat. **(7p)**

8. Ett mikrodatorsystem skall konstrueras med CPU12, 1 st 64 kbyte ROM-kapsel, 1 st 2 kbyte RWM-kapsel, 4 st 8-bitars "three-state"-buffertar som inportar och 2 st 8-bitars register som utportar. Figuren nedan visar principen för anslutning av olika moduler till CPU12.



RWM-modulen skall placeras så att den upptar de sista 2k adresserna i adressrummet, förutom de sista två adresserna, där inportarna nr 3 och 4 skall placeras, med inport nr 4 på den högsta adressen.

På de två adresserna närmast före RWM-modulen skall två inportar och två utportar, med nr 1 och 2 placeras, med nr 2 på den högsta adressen. Det skall alltså finnas både inport och utport på var och en av dessa adresser.

Adressområdet som består av de första 512 adresserna skall reserveras för framtida bruk. Ingen modul får därför aktiveras inom detta adressområde.

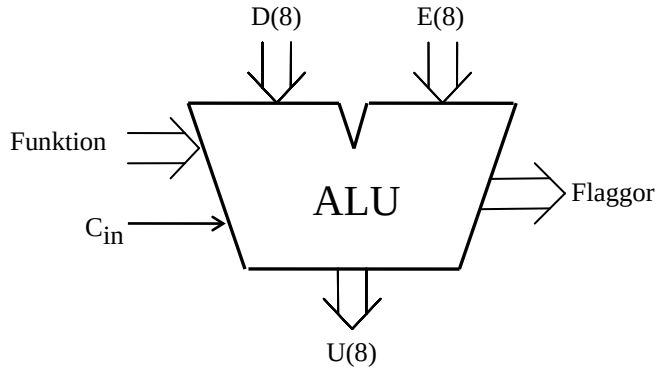
Uppgift:
Använd lämpliga signaler från processorn för att konstruera CS-logiken för ROM-modulen, RWM-modulen, inportarna och utportarna. Använd fullständig adressavkodning. Endast grundläggande logikgrindar med valfritt antal ingångar får användas. De användbara adressintervallen för de två minnesmodulerna skall anges på hexadecimal form.

Ledning: Adressområdet som omfattar de första 512 adresserna kan betraktas som en modul med 512 olika adresser.

(7p)

Bilaga 1

ALU:ns funktion (FLEX-ALU'n)



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- och FLIS-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.

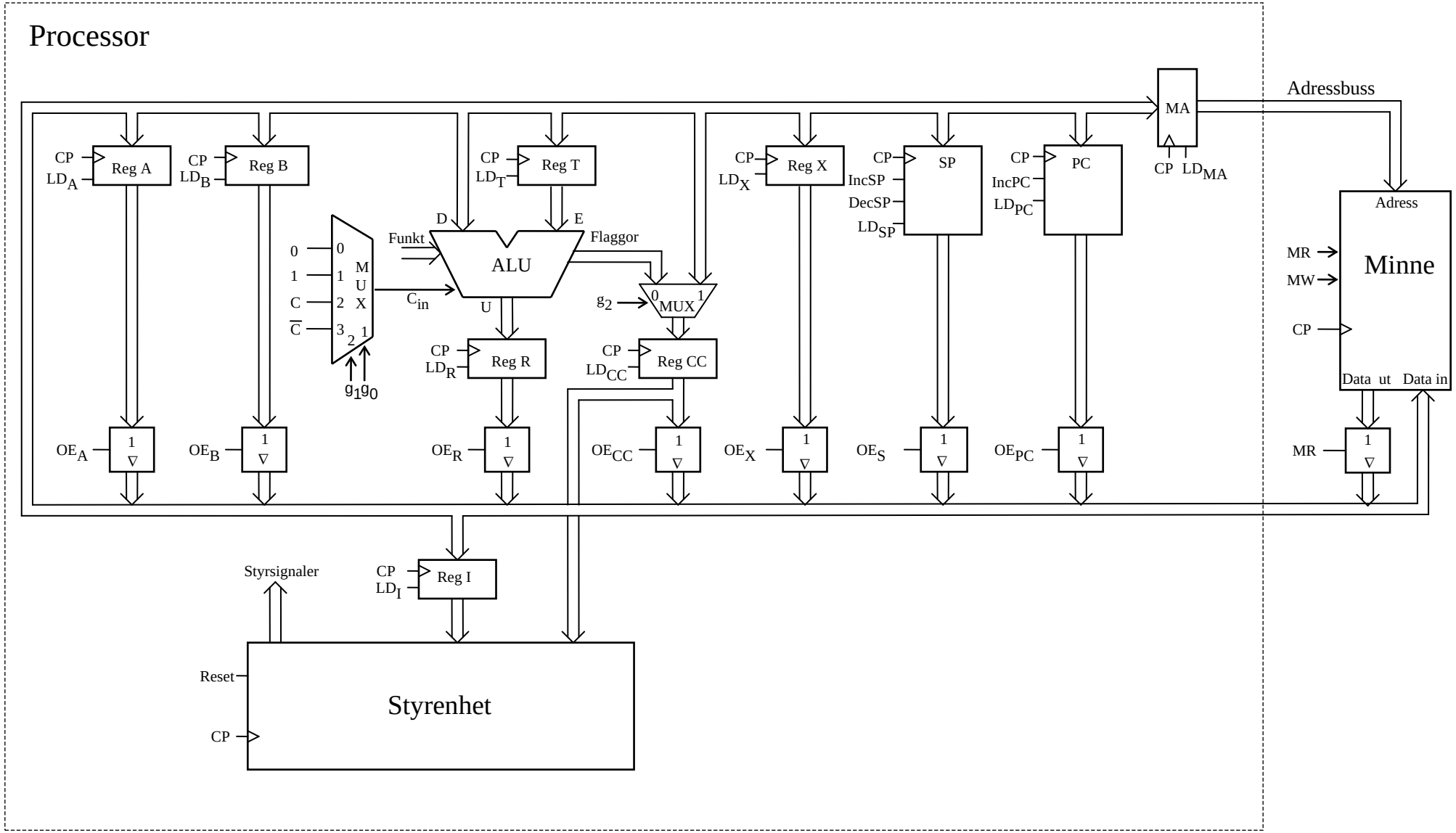
Tabell 1

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 (Observera att bilden visar FLEX-datorn)



Figur 1. Datorn FLEX.

Svarsblankett (2013-12-16)

Anonym kod:

Uppgift 2:

a)

1. u = a'b' + ac'
2. u = ab + a'c'
3. u = (a' + b)(a + c)
4. u = (a + b')(a' + c')
5. u = (a' + b)(b + c)(a + c)
6. u = (a + b')(b' + c')(a' + c')
7. u = ab + bc + a'c
8. u = a'b' + b'c' + ac'

b)

1. f = [a(b⊕c⊕d)] + a'd
2. f = a(b⊕c⊕d)' + a'd'
3. f = [a' + (b⊕c⊕d)'](a + d')
4. f = [a' + (b⊕c⊕d)](a + d)
5. f = a'(b⊕c⊕d) + ad
6. f = [a'(b⊕c⊕d)'] + ad'
7. f = [a + (b⊕c⊕d)'](a' + d')
8. f = [a + (b⊕c⊕d)](a' + d)

Uppgift 4:

CP	Styrsignaler (=1)	RTN	A(8)	B(8)	T(8)	R(8)
1	OE_B, LD_T		C8	64	?	?
2	OE_A, 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_R, LD_B					
6	?					

Uppgift 5:

a) (FLISP)

State	Summaterm	RTN-beskrivning	Styrsignaler
Q_4	$Q_4 \cdot I_{xx}$		MR, LD_T, INC_{PC}
Q_5	$Q_5 \cdot I_{xx}$		OE_X, f_3, f_1, f_0, LD_R
Q_6	$Q_6 \cdot I_{xx}$		OE_R, LD_X, NF

(FLEX)

State	Summaterm	RTN-beskrivning	Styrsignaler
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_X, NF

Vänd!

Uppgift 5 (forts):

b)

State	Summaterm	RTN-beskrivning	Styrsignaler