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	**2014-01-16 kl 8.30 – 12.30**
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.**
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 om ej annat anges.** **<u>En svarsblankett bifogas till tentamenstesen.</u>** **För de uppgifter där svaret skall ges på svarsblanketten behöver inga lösningar redovisas.**
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 6-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.

$X = 01\ 0110_2$ och $Y = 01\ 1111_2$.

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 6-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 6-bitars ALU. **(1p)**

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

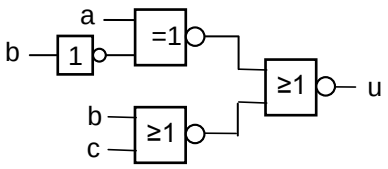
g) Tolka bitmönstren X, Y, S och D som tal *utan* tecken och ange deras decimala motsvarighet. Avgör och motivera med hjälp av flaggorna om resultaten S och D är korrekta eller felaktiga. **(1p)**

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 representeras "0" och "∞" med 32-bitar enligt flyttalsstandarden IEEE 754 (dvs 23 bitar av mantissan). Använd binär och hexadecimal form. **(2p)**

2.

a) Markera det korrekta minimala SP-uttrycket för u i grindnätet till höger på den bifogade svarsblanketten! **(2p)**



b) Ett karnaughdiagram för en boolesk funktion visas till höger. Markera ett minimalt uttryck på svarsblanketten som är lämpligt för realisering av den booleska funktionen f, då NOR-grindar med valfritt antal ingångar, XOR-grindar och NOT-grindar får användas.

Rutor med - representerar "dont care"-termer som får användas vid behov. Kretsrealiseringen behöver ej visas. **(4p)**

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

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 bestående av tre nollor följda av två ettor och en nolla.

Exempel: σ_x = ...0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 1 1 1 0 0 0 1 1 0 0 1 ...

σ_u = ...? 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 ...

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. (Sekvensnätet skall ej realiseras!)

Hur många vippor skulle minst krävas vid en realisering? **(4p)**

b) Konstruera en reversibel 2-bitars synkron räknare med räknevillkoret x.

För x = 0 skall räknesekvensen $(q_1q_0)_2$ vara: 00, 11, 01, 10, 00, …
För x = 1 skall räknesekvensen $(q_1q_0)_2$ vara: 00, 10, 01, 11, 00, …

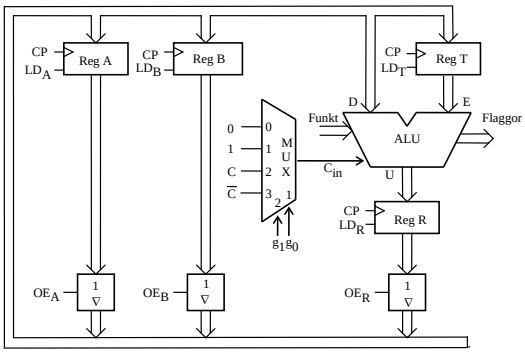
JK-vippor, NAND-grindar med valfritt antal ingångar, XOR-grindar och NOT-grindar får användas. **(6p)**

4. I datavägen till höger innehåller register A(8) och B(8) från början värdena 70_{10} resp. 165_{10} på binär form. Därefter ges styrsignaler och klockpulser enligt tabellen på svarsblanketten.

Komplettera tabellen på svarsblanketten 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!



(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 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 nedan "Test register A and branch if not equal to zero" skall implementeras för FLIS(FLEX)-processorn med hjälp av styrenheten med fast logik. Operationskoden DF_{16} skall användas.

TBNE A,Adr RTN: U = A, ALU(NZVC) → CC

If Z = 0: PC + offset → PC

(U i RTN-beskrivningen ovan betecknar utdatavärdet från ALU'n.)

Komplettera tabellen på svarsblanketten med RTN-beskrivning och styrsignaler för den efterfrågade EXECUTE-sekvensen. **(5p)**

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

a) Före det villkorliga hoppet "BMI Adr" i ett program utförs en subtraktion "$7A_{16} - W$", som påverkar flaggorna. För vilka värden på talet W utförs hoppet? (8-bitars tal används.) **(4p)**

b) 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.

```
CONST   EQU     -10
;
        ORG     $90
;
DELAY   PSHA
        PSHA
;
YLOOP   LDA     #CONST
ILOOP   INCA
        NOP
        BNE     ILOOP
;
        INC     0,SP
        BNE     YLOOP
;
        LEASP   1,SP
        PULA
        RTS
```

(3p)

c) Subrutinen i b) anropas i huvudprogrammet nedan.

```
        ORG     $10
        LDSP    #$FA
        LDA     #$FB
        BSR     DELAY
STOP    BRA     STOP
```

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

(För FLEX-processorn skall du använda motsvarande instruktioner enligt dess instruktionslista.)

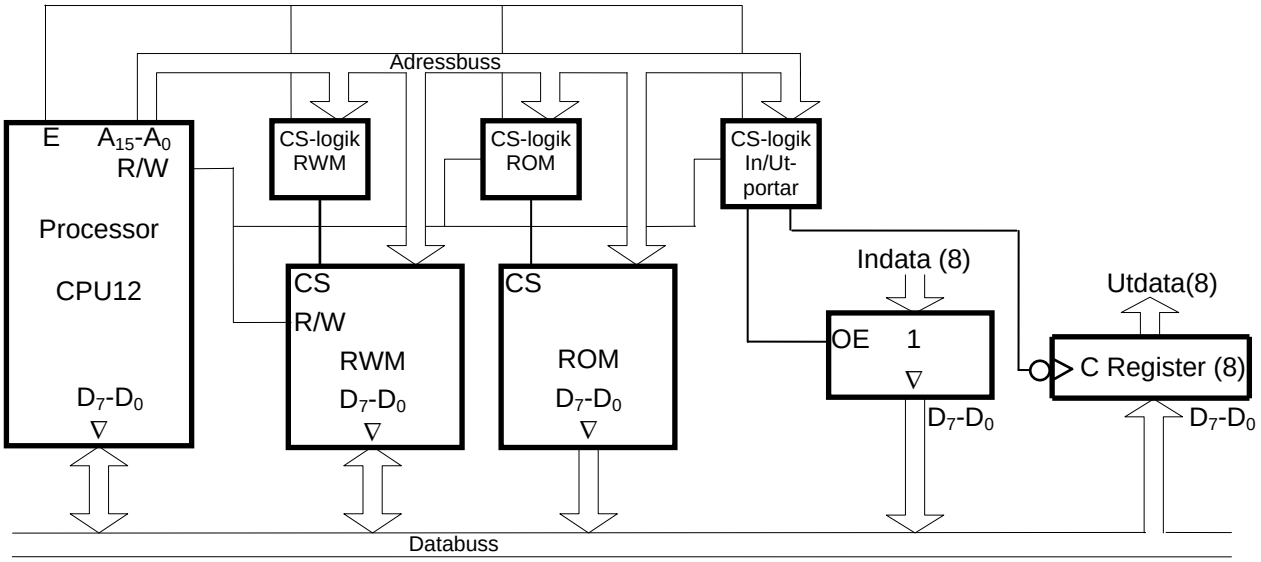
(3p)

7. I minnet i ett datorsystem med FLIS(FLEX)-processorn finns en sträng med sjubitars ASCII-tecken, där varje ASCII-tecken har kompletterats med en paritetsbit som åttonde bit (bit nr 7). Strängen avslutas med ett dataord med värdet noll.

Skriv en subrutin HEXCH i assemblerpråk för FLIS-processorn som tar reda på hur många av ASCII-tecknen i strängen som motsvarar stora eller små bokstäver A – F eller a – f. Antalet sådana bokstäver skall finnas i A-registret vid återhopp. Vid anrop av subrutinen antas startadressen till strängen finnas i X-registret. En tabell över ASCII-koden finns i bilaga 2.

Endast register A och register CC får vara förändrade vid återhopp från subrutinen. För full poäng på uppgiften skall programmet vara korrekt radkommenterat. **(7p)**

8. Ett mikrodatorsystem skall konstrueras med CPU12, 1 st 64 kbyte RWM-kapsel, 1 st 8 kbyte ROM-kapsel, 2 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.



Adressområdet som består av de första 2k adresserna skall reserveras för framtida bruk. Ingen modul får därför aktiveras inom detta adressområde.

ROM-modulen skall placeras i adressrummet så att dess slutadress blir $FFFF_{16}$.

På de två adresserna närmast före ROM-modulen skall två inportar och två utportar placeras. Det skall alltså finnas både inport och utport på var och en av dessa adresser.

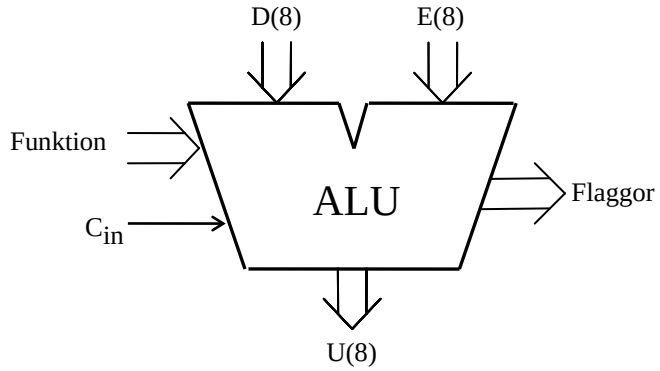
Alla övriga adresser skall användas för RWM-modulen.

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.

(6p)

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.

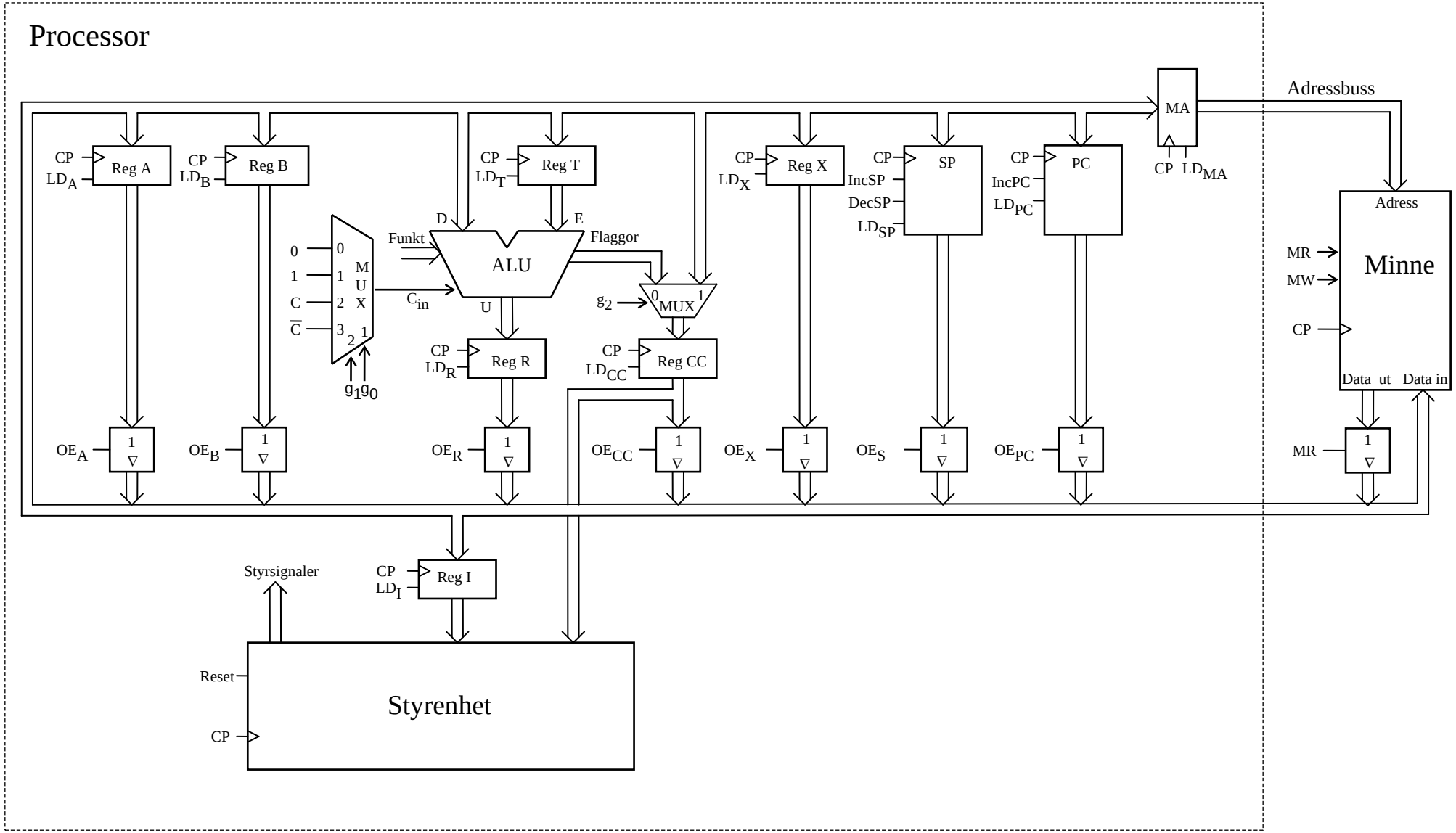
Tabell 1

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 (2014-01-16)

Anonym kod:

Uppgift 2:

a)

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

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: (Här används FLEX-ALU'n!)

CP	Styrsignaler (=1)	RTN	A(8)	B(8)	T(8)	R(8)
1	OE_A, LD_T		46	A5	?	?
2	OE_B, f_3, f_1, LD_R					
3	OE_R, f_3, f_1, f_0, LD_R					
4	OE_R, f_3, f_1, f_0, LD_R					
5	OE_R, f_3, f_2, LD_R					
6	OE_R, LD_A					
7	?					

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_{CC}, f_2, f_1, LD_R
Q_6	$Q_6 \cdot I_{xx}$		OE_R, g_{10}, g_8, g_6, g_4, g_2, LD_{CC}, 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_{CC}, f_2, f_0, LD_R
Q_8	$Q_8 \cdot I_{xx}$		OE_R, LD_{CC}, NF

Instruktionen är:

Vänd!

Uppgift 5 (forts):

b) (Du kan skriva samma statenummer på flera rader vid behov!)

State	Summaterm	RTN-beskrivning	Styrsignaler