CHALMERS

Institutionen för data- och informationsteknik
Avdelningen för datorteknik

TENTAMEN

KURSNAMN	**Digital- och datorteknik E**
PROGRAM:	**Elektro** **Åk 1/ lp 4**
KURSBETECKNING	**EDA216/DIT790**
EXAMINATOR	**Lars-Eric Arebrink**
TID FÖR TENTAMEN	**2010-05-27 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	**Resultatlistor med anonyma koder anslås senast 2010-06-10 på kursens hemsida. Granskning av rättning på institutionen 2010-06-10 och 2010-06-11 kl 9.30-10.00.**
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-f nedan används 9-bitars tal X, Y och R. X = 100000101 och Y = 000011001.

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

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

c) Visa med penna och papper hur räkneoperationen R = X – Y utförs i en dators ALU. **(1p)**

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

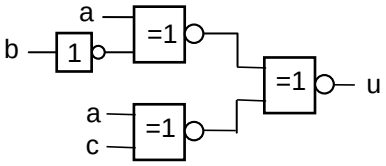
e) Tolka bitmönstren R, X och Y som tal *utan* tecken och ange deras decimala motsvarighet. Vilken eller vilka flaggbitar visar om resultatet är korrekt vid tal utan tecken? **(1p)**

f) Tolka bitmönstren R, X och Y som tal *med* tecken och ange deras decimala motsvarighet. Vilken eller vilka flaggbitar anger om resultatet är korrekt vid tal med tecken? **(1p)**

g) Skriv, om det är möjligt, det decimala talet –250 som ett 9-bitars tal med tvåkomplements-representation. **(1p)**

h) Visa approximativt hur många bitar man skulle behöva i mantissan på ett flyttal för att man skall kunna ”lita på” 15 decimala siffror i talet översatt till decimal form. **(3p)**

i) Ge ett minimalt boolesk uttryck för u i grindnätet till höger. **(4p)**



2. Insignalerna a, b, c och d till ett grindnät skall tolkas som ett binärt tal ($abcd_2$). Grindnätets utsignal u skall ha värdet 1 om det binära talet motsvarar en otillåten NBCD-siffra.

Konstruera ett ”minimalt” grindnät som bildar utsignalen u. NOT-grindar och NOR-grindar med valfritt antal ingångar får användas. Insignalerna finns inte tillgängliga på inverterad form. **(4p)**

3. Realisera en T-vippa med hjälp av en D-vippa och standardgrindar. **(4p)**

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

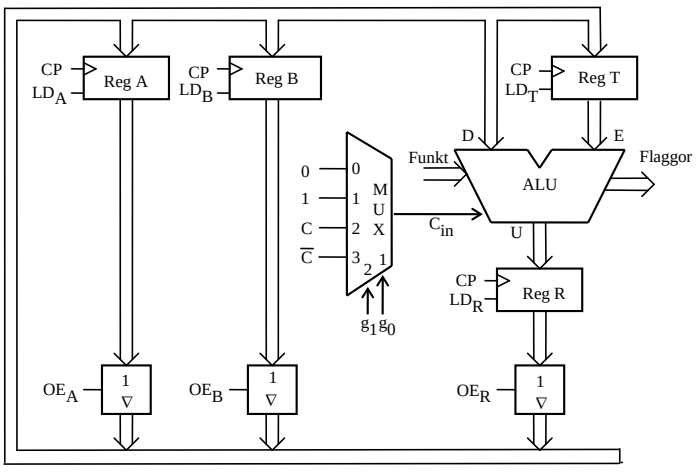
RTN-beskrivning: $4\cdot(A - 1) - 5\cdot B \rightarrow A$
(Aritmetisk multiplikation avses)

Register B får inte ändras. Bortse från risken för overflow. Använd så få tillstånd som möjligt.

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.

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

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



(5p)

5. Figur 1 i bilaga 4 visar hur datorn FLEX är uppbyggd. Bilaga 1 visar hur ALU'ns funktion väljs med styrsignalerna f_3 - f_0 och signalen C_{in}.

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

State nr	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}	$Q_{10} \cdot I_{xx}$		OE_A, MW, NF.

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

a) Ge RTN-beskrivningen för tillstånden Q_5-Q_{10}. **(1p)**

b) Förklara vad instruktionen med EXECUTE-sekvensen ovan utför i varje klockcykel. Skriv instruktionen med assemblerspråk för FLEX-processorn. **(2p)**

c) Instruktionen nedan skall implementeras för FLEX-processorn med hjälp av styrenheten med fast logik.

TBEQ B,Adr RTN: U = B + 0, Flags → CC

If Z = 1: PC + offset → PC

Samtliga funktioner ALU:n kan utföra framgår av bilaga 1. FLEX-datorn visas i bilaga 4. Gör en tabell liknande tabellen ovan för den efterfrågade EXECUTE-sekvensen. **(4p)**

6. Besvara kortfattat följande frågor rörande FLEX-processorn.

a) Varför behövs programräknaren i FLEX-datorn, som är en von Neumanndator? **(2p)**

b) Vad har X- registret för funktion? Redogör för när X-registret används **(2p)**

c) Instruktionerna BLO Adr och BLT Adr innebär bägge att hopp skall utföras om villkoret "<" är uppfyllt. Förklara skillnaden mellan instruktionerna och vilken praktisk betydelse den har. **(2p)**

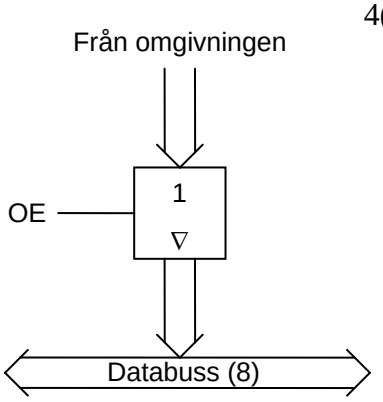
d) Översätt programavsnittet nedan till maskinkod. Det skall framgå hur "offset" för branchinstruktionerna beräknas. Programavsnitttets startadress är 50_{16}.

```
        LDAA  #$F6
ALOOP   LDX   #$FC
XLOOP   INX
        BNE   XLOOP
        NOP
        INCA
        BNE   ALOOP
        NOP
```

(3p)

e) Hur många klockpulser krävs för att köra hela programavsnittet i d). **(3p)**

7. FLEX-datorn som visas i bilaga 4 skall kompletteras med en inport på adressen FC_{16}. Principen för inporten visas i figuren till höger.



a) Konstruera ett grindnät som bildar signalen OE.

b) Vid inkoppling av inporten får man problem med minnet. Visa hur man kan lösa detta problem.

Standardgrindar med valfritt antal ingångar får användas.

(3p + 3p)

8. Skriv en subrutin CONV i assemblerspråk för CPU12 som söker igenom en textsträng med 7-bitars ASCII-tecken och ändrar alla ASCII-tecken (a-z) i strängen till motsvarande ASCII-tecken (A-Z). Textsträngen avslutas med dataordet 0. ASCII-tabell finns i tabell 2 på sidan 7.

Vid anrop av subrutinen skall adressen till textsträngen finnas i X-registret.

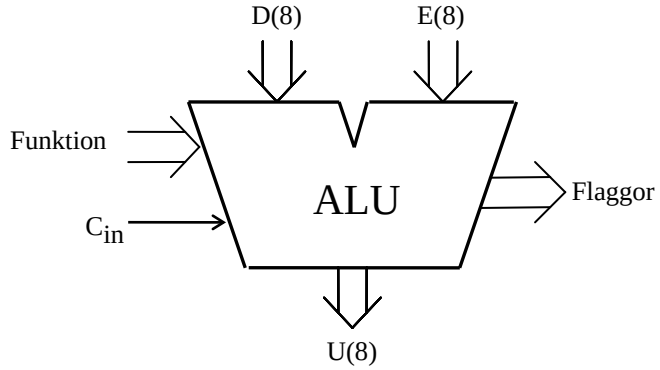
ASCII-tecknen är lagrade i bit6-bit0 i varje dataord i textsträngen. Bit7 i textsträngens dataord har okänt värde och får inte ändras.

Processorns flaggregister får vara förändrat vid återhopp. Övriga register skall vara oförändrade.

För full poäng på uppgiften skall programmet vara korrekt radkommenterat. **(8p)**

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

Med t ex $\mathbf{D_{1k}}$ menas att samtliga bitar i **D** inverteras.

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.

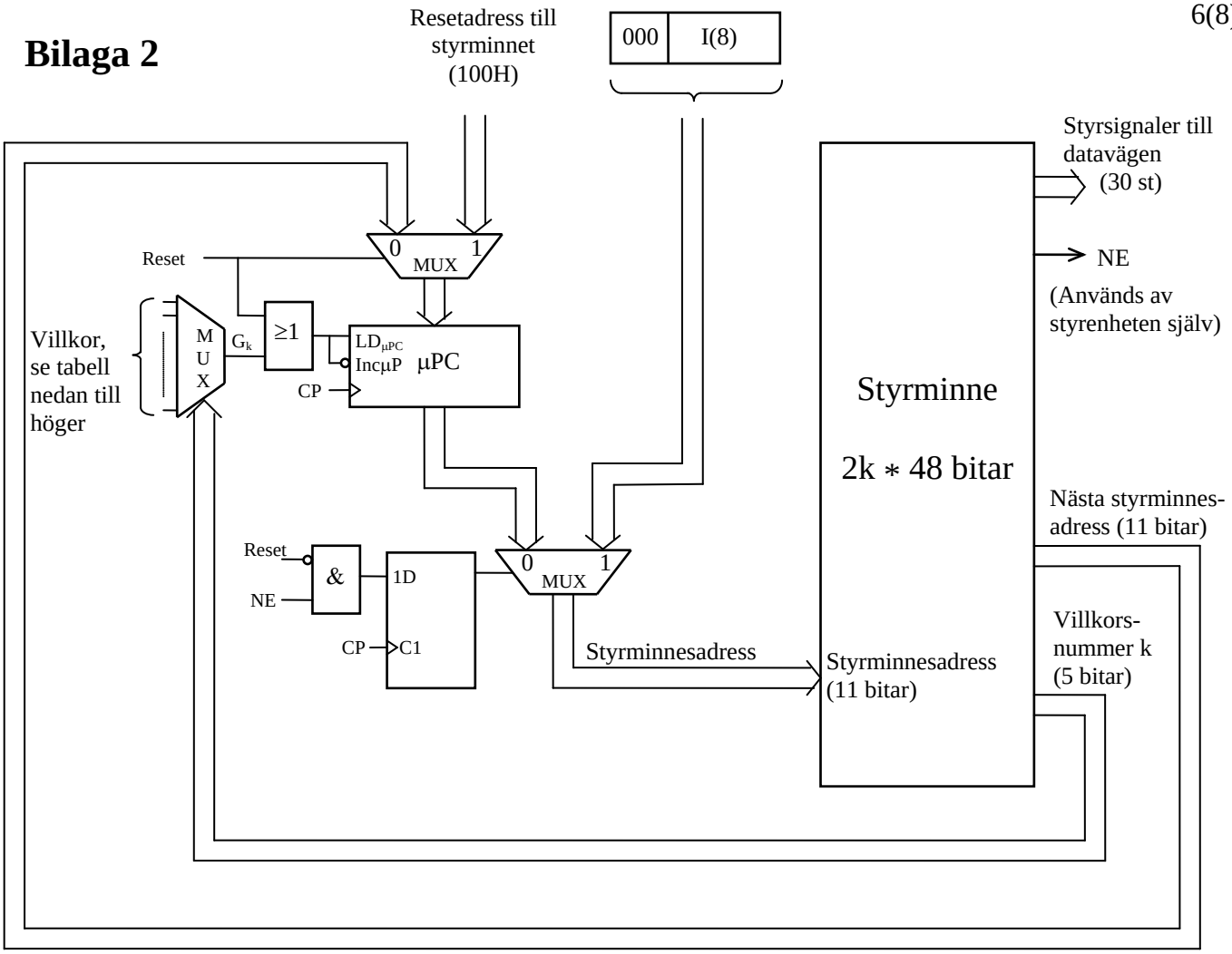
Signflaggan (N) är identisk med den mest signifikanta biten (teckenbiten) av utsignalen U från ALU:n.

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**.

Bilaga 2



Mikroprogrammerad EXECUTE-sekvens för instruktionen DEC Adr:

CM-adress	Hoppvillkor		Hopp-adress	RTN	Styrsignaler
(hex)	Kod (hex)	G_K	(hex)		(aktiva)
046	0 F	$G_{0F} = 1$	230	PC→MA, PC+1→PC	OE_{PC}, LD_{MA}, IncPC
230	0 0	$G_{00} = 0$	-	M→MA	MR, LD_{MA}
231	0 0	$G_{00} = 0$	-	M-1→R, Flags→CC	MR, f_3, f_0, LD_R, LD_{CC}
232	0 F	$G_{0F} = 1$	108	R→M, Next Fetch	OE_R, MW

k_4 $k_3k_2k_1k_0$

$K = k_4k_3k_2k_1k_0$

$k_3k_2k_1k_0$	G_K
0 0 0 0	0
0 0 0 1	C
0 0 1 0	V
0 0 1 1	Z
0 1 0 0	N
0 1 0 1	0
0 1 1 0	?
0 1 1 1	0
1 0 0 0	0
1 0 0 1	C+Z
1 0 1 0	N⊕V
1 0 1 1	Z+(N⊕V)
1 1 0 0	0
1 1 0 1	0
1 1 1 0	0
1 1 1 1	1

Med $k_4 = 1$ inverteras villkorssignalen G_K i tabellen till höger.

Bilaga 3

Assemblerspråket för FLEX-processorn.

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

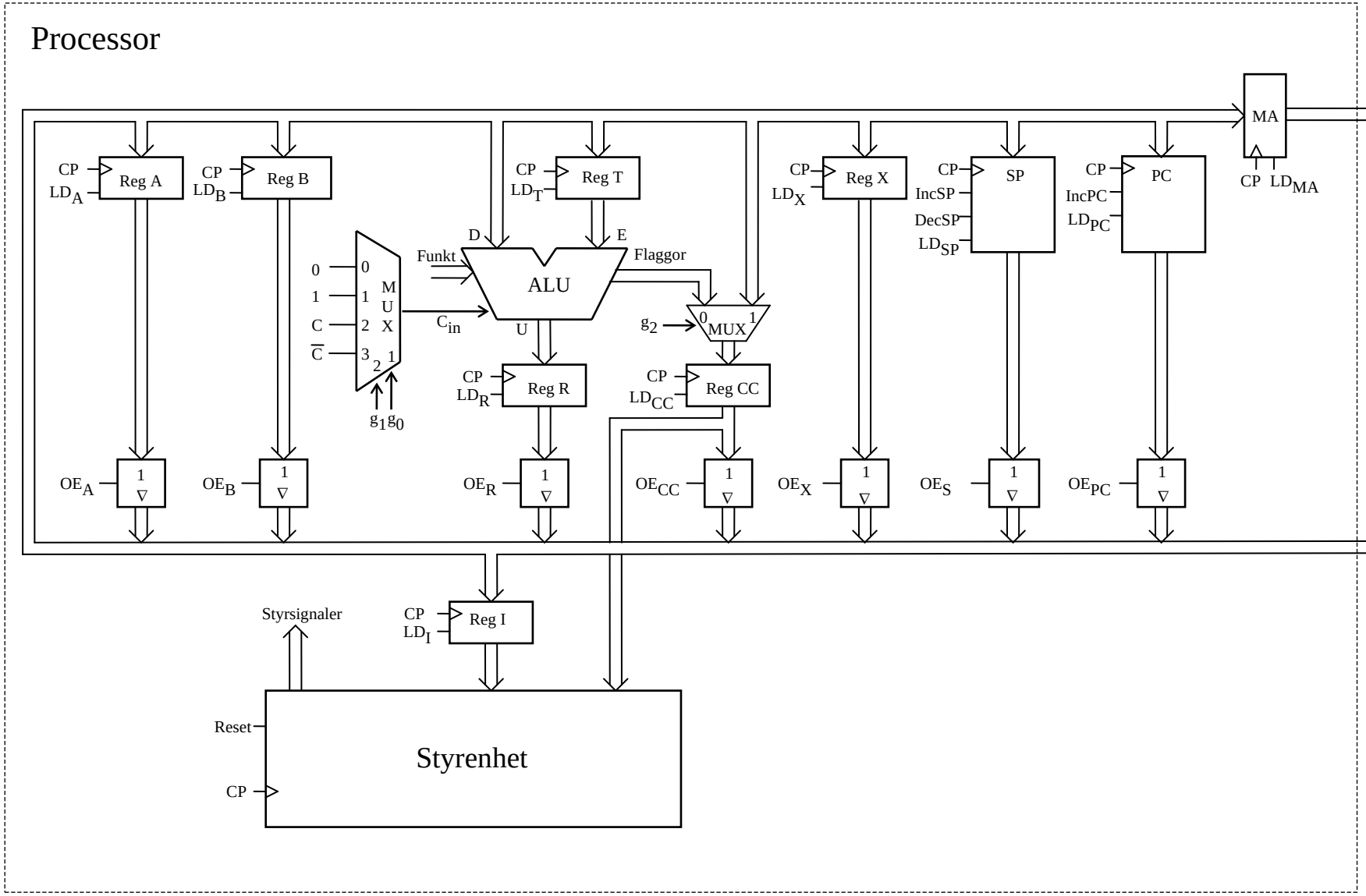
Tabell 1

Direktiv	Förklaring	Exempel	Förklaring
ORG N	Sets the origin to location N, so succeding code is written, starting at location N.	ORG $100	Puts the next and following bytes beginning in location $100.
L RMB N	Allocates N words and designates L as the label of the first word of the area being allocated.	L1 RMB 10	Makes the label L1 have the value of the current location; increments the location counter by 10.
L EQU N	Declares label L to have value N.	L2 EQU $10	Makes the symbol L2 have the value $10.
L FCB N1, N2	Forms (generates) one byte per argument, assigns label L to the first byte.	L3 FCB $20, $34	Makes the symbol L3 have the value of the current location; initializes the current location to the value $20 and the next location to $34.

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 4



Figur 1. Datorn FLEX.