

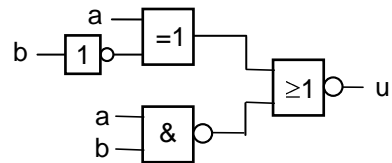
**TENTAMEN** (*Något redigerad*)

<b>KURSNAMN</b>	<b>Digital- och datorteknik E</b>
<b>PROGRAM:</b>	Elektro Åk 1/ lp 4
<b>KURSBETECKNING</b>	EDA216/DIT790
<b>EXAMINATOR</b>	Lars-Eric Arebrink
<b>TID FÖR TENTAMEN</b>	2009-01-14 kl 14.00 – 18.00
<b>HJÄLPMEDEL</b>	Av institutionen utgiven ”Instruktionslista för FLEX-processorn” (INS1) Tabellverk eller miniräknare får ej användas.
<b>ANSV LÄRARE:</b>	Rolf Snedsböl, tel. 772 1665 (Lars-Eric Arebrink, tel. 772 5718)
<b>ANSLAG AV RESULTAT</b>	Resultatlistor anslås senast 2009-01-28 på kursens hemsida. Granskning av rättning på institutionen 2009-01-28 och 2009-01-30 kl 12.30-13.00.
<b>ÖVRIG INFORM.</b>	Tentamen omfattar totalt 60 poäng. Onödigt komplicerade lösningar kan ge poängavdrag. Svar på uppgifter skall motiveras.
<b>BETYGSGRÄNSER.</b>	Betyg 3: 24 poäng Betyg 4: 36 poäng Betyg 5: 48 poäng
<b>SLUTBETYG</b>	För slutbetyg 3, 4 eller 5 på kursen fordras betyg 3, 4 eller 5 på tentamen och godkända laborationer.

1. I uppgift a-d nedan används talen  $X = 100101$  och  $Y = 001010$ .

- a) Visa med penna och papper hur räkneoperationen  $R = X - Y$  utförs i en 6-bitars ALU. (1p)
- b) Vilka värden får flaggbitarna  $N$ ,  $Z$ ,  $V$  och  $C$  vid räkneoperationen? Motivera varför. (1p)
- c) Tolka bitmönstren  $R$ ,  $X$  och  $Y$  som tal *utan* tecken och ange deras decimala motsvarighet. Hur vet man i detta fall om den uträknade skillnaden är korrekt? (1p)
- d) Tolka bitmönstren  $R$ ,  $X$  och  $Y$  som tal *med* tecken och ange deras decimala motsvarighet. Hur vet man i detta fall om den uträknade skillnaden är korrekt? (1p)
- e) Genomför subtraktionen  $(5872)_{10} - (7594)_{10}$  med 10-komplementaritmetik. Hur många decimala sifferpositioner behövs? Hur skall man tolka resultatet? (3p)
- f) Omvandla talet  $(10111011110111)_2$  till hexadecimal form. (1p)

g) Ge ett minimalt boolesk uttryck för  $u$  i grindnätet till höger. (3p)



2. Det booleska uttrycket  $f(x,y,z,w) = (x' + y + w')(y' + z' + w)(y' + z + w)(x + y + w')$  beskriver en boolesk funktion. Konstruera ett "minimalt" grindnät som realiserar den booleska funktionen. NOR-grindar med valfritt antal ingångar, XOR-grindar och NOT-grindar får användas. Endast signalerna  $x$ ,  $y$ ,  $z$  och  $w$  finns tillgängliga. (4p)

3. 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 exakt två st nollor följda av en etta och en nolla hos  $x$ .

Exempel:  $\sigma_x = \dots 100101100010010\dots$   
 $\sigma_u = \dots 0?0010000000001\dots$

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.

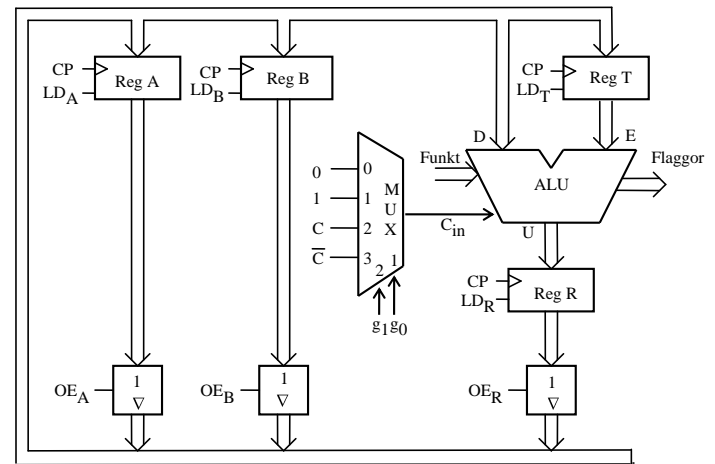
- a) Rita en tillståndsgraf för sekvensnätet. (4p)
- b) Realisera en T-vippa med hjälp av en SR-vippa och standardgrindar. (4p)

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

RTN-beskrivning:  $4 \cdot (A + 1) - 5 \cdot (B + 1) \rightarrow B$   
(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 innehållena i register R och T är okända. Register A 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 4 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 RTN-beskrivningen av EXECUTE-sekvensen för en av FLEX-processorns instruktioner.

State	Summaterm	RTN-beskrivning	Aktiva styrsignaler (=1)
Q <sub>5</sub>	Q <sub>5</sub> ·I <sub>xx</sub>	PC → MA	
Q <sub>6</sub>	Q <sub>6</sub> ·I <sub>xx</sub>	M → MA	
Q <sub>7</sub>	Q <sub>7</sub> ·I <sub>xx</sub>	M → MA	
Q <sub>8</sub>	Q <sub>8</sub> ·I <sub>xx</sub>	M → PC, NF	

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

- a) Fyll i styrsignalkolumnen i tabellen. Endast styrsignaler = 1 skall anges. (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. Operationskoden F0H skall användas.

IBNE B,Adr RTN:  $B + 1 \rightarrow B, \text{Flags} \rightarrow \text{CC}$

If  $Z = 0: \text{PC} + \text{offset} \rightarrow \text{PC}$

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.

OPKOD
offset

(5p)

6. Besvara kortfattat följande frågor rörande FLEX-processorn.

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

b) I instruktionsuppsättningen finns de två ovillkorliga hoppen JMP och JSR. Vilken skillnad är det mellan dessa? **(2p)**

c) Det finns två principer för att ansluta inportar och utportar till processorns bussar, separatadresserad resp. minnesorieterad in- och utmatning. Vilken av dessa används i FLEX-processorn? Vilka är fördelarna med denna metod? **(2p)**

d) De villkorliga hoppinstruktionerna i instruktionsuppsättningen använder en eller flera flaggor för att bilda hoppvillkoret. Förklara hur och varför flaggorna används då hoppvillkoret avser tal med tecken. **(4p)**

e) Översätt instruktionssekvensen nedan till maskinkod och visa hur den placeras i minnet.

```

                ORG    0
                LDX   #DATA
                LDAA  3,X
*
WAIT1          LDAB  ,X
*
WAIT2          DECB
                NOP
                BNE  WAIT2
*
                NOP
                DECA
                BNE  WAIT1
*
                BRA  P2
*
DATA          FCB   0,1,2,4,8,16,32,64,128
*
P2            LDAA  $FD

```

**(3p)**

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

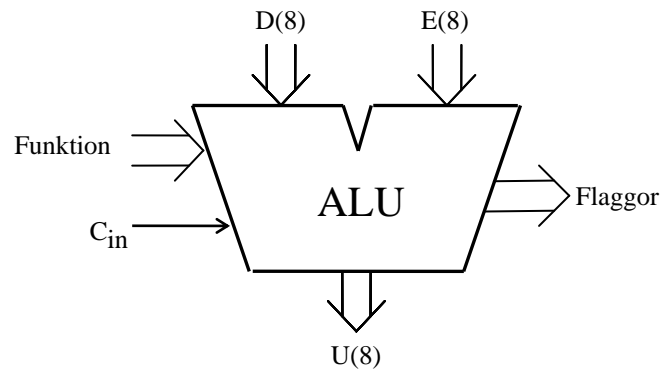
7. En FLEX-dator skall användas för att styra ett elektroniskt lås. I låsprogrammet skall ingå en subrutin som jämför en sträng med inmatade ASCII-tecken med en annan sträng med 8-bitars ord, som innehåller ”nyckeln” till låset. Bit 7 i varje inmatat ASCII-tecken har okänt värde medan motsvarande bit i ”nyckelsträngen” är noll. Bägge strängarna är lika långa, relativt korta och nollterminerade.

Skriv en subrutin i assemblerpråk för FLEX-processorn som jämför de två strängarna och returnerar antal fel i A-registret. Vid anrop av subrutinen skall startadressen till den inmatade strängen finnas i X-registret. Startadressen till ”nyckelsträngen” finns i minnet på adressen COH.

Endast register A och register CC får vara förändrade vid återhopp från subrutinen. För full poäng på uppgiften skall programmet vara korrekt kommenterat. **(8p)**

## Bilaga 1

## ALU:ns funktion



ALU:ns **logik-** och **aritmetikoperationer** på indata **D** och **E** definieras av ingångarna **Funktion (F)** och **C<sub>in</sub>** enligt tabellen nedan. **F = (f<sub>3</sub>, f<sub>2</sub>, f<sub>1</sub>, f<sub>0</sub>)**.

I kolumnen Operation förklaras hur operationen utförs.

f <sub>3</sub> f <sub>2</sub> f <sub>1</sub> f <sub>0</sub>	U = f(D,E,C <sub>in</sub> )	
	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 <sub>inv</sub>
0 1 0 0	bitvis invertering	E <sub>inv</sub>
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 <sub>in</sub>	D + C <sub>in</sub>
1 0 0 1	D + FFH + C <sub>in</sub>	D - 1 + C <sub>in</sub>
1 0 1 0		D + E + C <sub>in</sub>
1 0 1 1	D + D + C <sub>in</sub>	2D + C <sub>in</sub>
1 1 0 0	D + E <sub>inv</sub> + C <sub>in</sub>	D - E - 1 + C <sub>in</sub>
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.

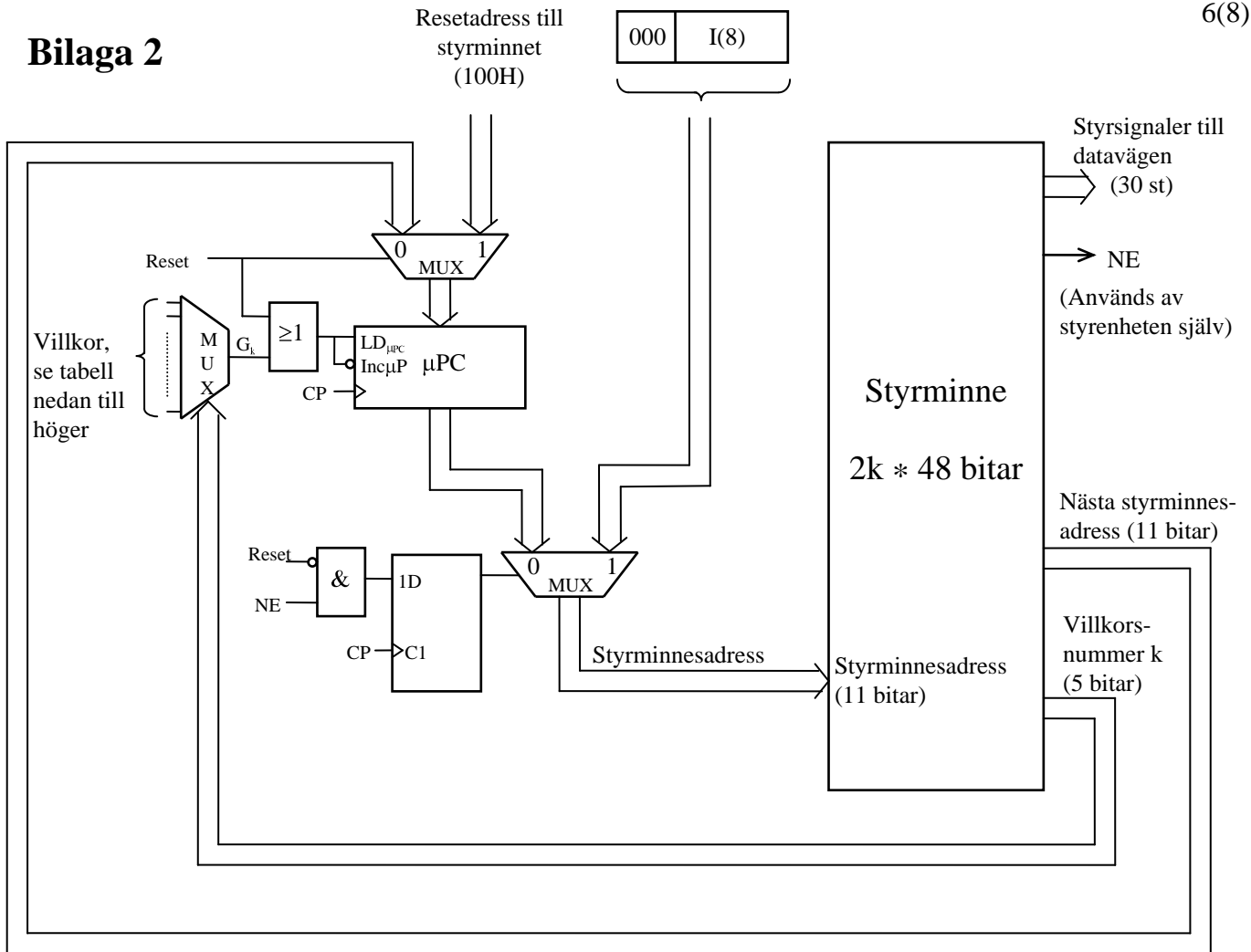
**Signflaggan (N)** är identisk med den mest signifikanta biten (teckenbiten) av utsignalen U från ALU:n.

**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 **D<sub>1k</sub>** menas att samtliga bitar i **D** inverteras.

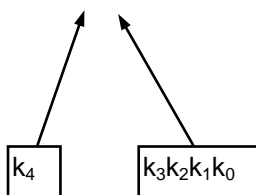
## Bilaga 2



### Mikroprogrammerad EXECUTE-sekvens för instruktionen DEC Adr:

CM-adress (hex)	Hoppvillkor		Hopp-adress (hex)	RTN	Styrksignaler (aktiva)
	Kod (hex)	G <sub>K</sub>			
046	0 F	G <sub>0F</sub> = 1	230	PC → MA, PC+1 → PC	OE <sub>PC</sub> , LD <sub>MA</sub> , Inc <sub>PC</sub>
230	0 0	G <sub>00</sub> = 0	-	M → MA	MR, LD <sub>MA</sub>
231	0 0	G <sub>00</sub> = 0	-	M-1 → R, Flags → CC	MR, f <sub>3</sub> , f <sub>0</sub> , LD <sub>R</sub> , LD <sub>CC</sub>
232	0 F	G <sub>0F</sub> = 1	108	R → M, Next Fetch	OE <sub>R</sub> , MW

k <sub>3</sub> k <sub>2</sub> k <sub>1</sub> k <sub>0</sub>	G <sub>K</sub>
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<sub>K</sub> i tabellen till höger.

## Bilaga 3

Assemblerspråket för FLEX-processorn.

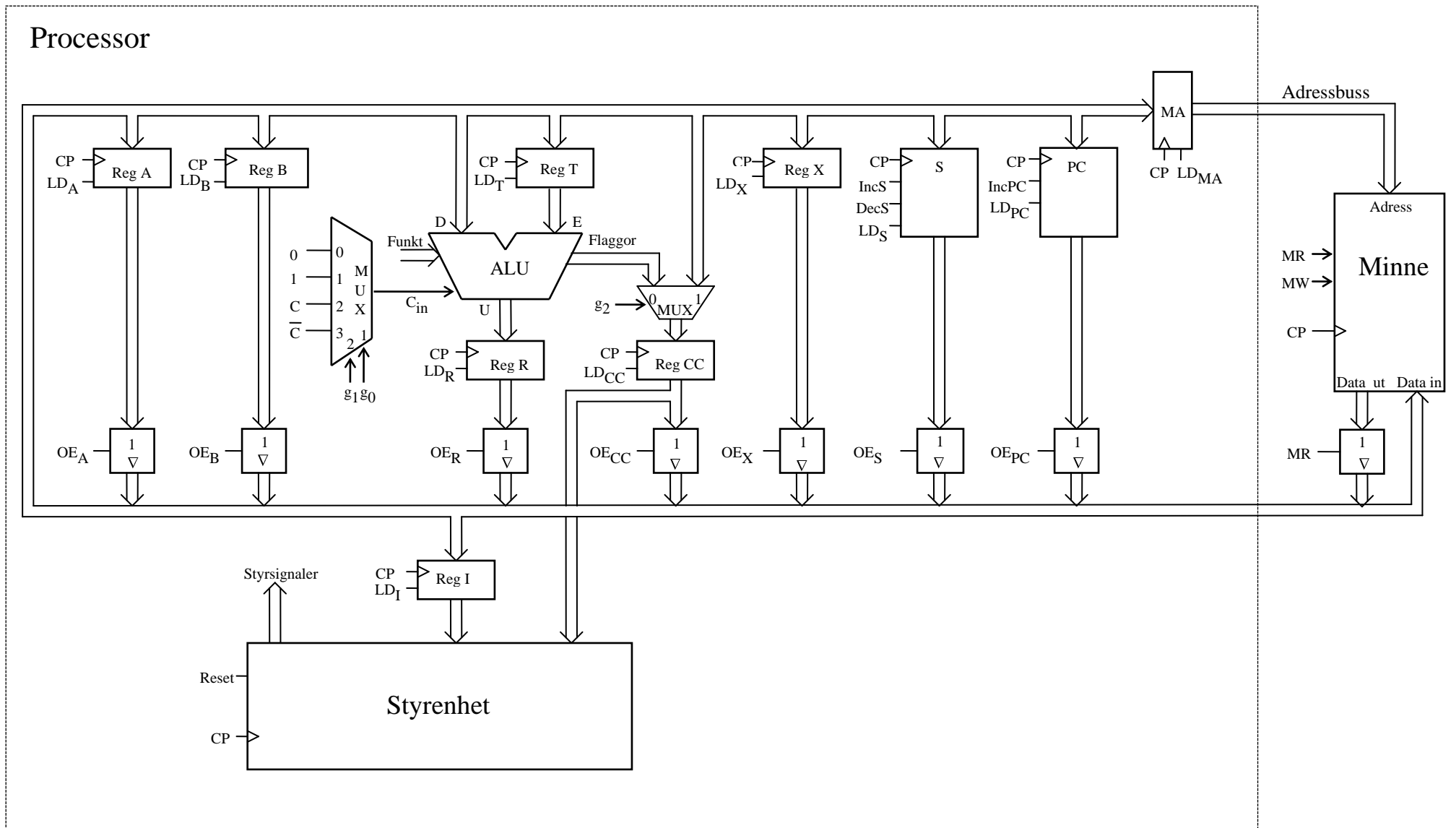
Assemblerspråket använder sig av mnemoniska beteckningar liknande dem som processorkonstruktören MOTOROLA specificerat för maskininstruktioner för mikroprocessorerna 68XX och instruktioner till assemblatorn, såsom pseudoinstruktioner eller assemblatordirektiv. Pseudoinstruktionerna listas i tabell 1.

**Tabell 1**

Direktiv	Förklaring	Exempel	Förklaring
ORG N	Sets the origin to location N, so succeeding 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**

000	001	010	011	100	101	110	111	$b_6b_5b_4$ $b_3b_2b_1b_0$
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



Figur 1. Datorn FLEX.