

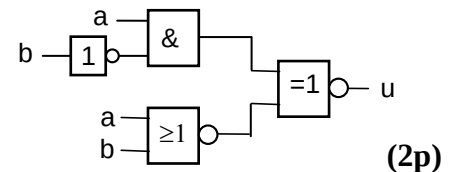
**TENTAMEN**

KURSNAMN	Digital- och datorteknik E (Något redigerad)
PROGRAM:	Elektro Åk 1/ lp 4
KURSBETECKNING	EDA216/DIT790
EXAMINATOR	Lars-Eric Arebrink
TID FÖR TENTAMEN	2009-05-27 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 anslås senast 2009-06-11 på kursens hemsida. Granskning av rättning på institutionen 2009-06-10 och 2009-06-11 kl 9.30-10.00.
ÖVRIG INFORM. BETYGSGRÄNSER. SLUTBETYG	Tentamen omfattar totalt 60 poäng. Onödigt komplicerade lösningar kan ge poängavdrag. Svar på uppgifter skall motiveras. Betyg 3: 24 poäng Betyg 4: 36 poäng Betyg 5: 48 poäng 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 6-bitars tal X, Y och R. $X = 100111$ och $Y = 010001$.

- 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 -32 som ett 6-bitars tal med tvåkomplements-representation. (1p)
- h) Talet 45834200H är ett flyttal enligt IEEE-standard 754 (23 bitars mantissa och 8 bitars karakteristika). Skriv talet på binär och decimal form. (2p)

i) Ge ett minimalt boolesk uttryck för u i grindnätet till höger.



2. Det booleska uttrycket $f(x,y,z) = xz + yz' + x'y'z' + x'yz$ beskriver en boolesk funktion.

Konstruera ett "minimalt" grindnät som realiserar den booleska funktionen. NAND-grindar med valfritt antal ingångar, XOR- och NOT-grindar får användas. Endast insignalerna x, y och z finns tillgängliga. (4p)

3. Ett synkront sekvensnät skall ha en insignal x och två utsignaler u_1 och u_0 . Utsignalerna (u_1u_0) skall ges värdet "01" under sista bitintervallet för varje insignalsekvens ...001... hos x och värdet "10" under sista bitintervallet för varje insignalsekvens ...011... hos x. Alla andra insignalsekvenser skall ge utsignalerna värdet "00".

Exempel: $\sigma_x = \dots 0010110011\dots$

$\sigma_{u_0} = \dots 0010000010\dots$

$\sigma_{u_1} = \dots 0000010001\dots$

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.

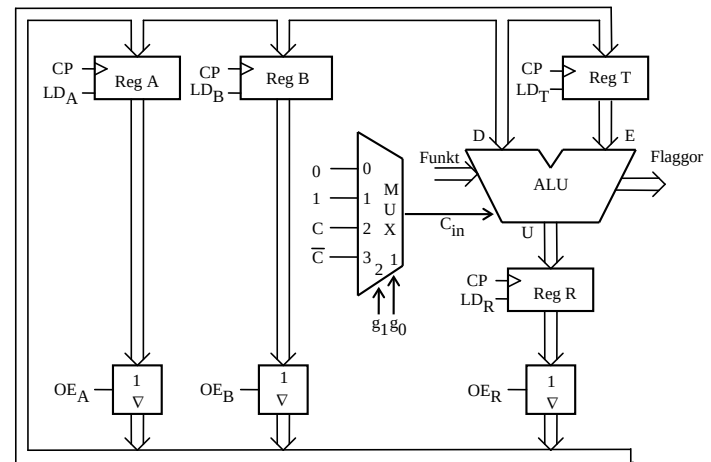
- a) Rita en tillståndsgraf för sekvensnätet. (6p)
- b) Realisera en JK-vippa med hjälp av en D-vippa och standardgrindar. (4p)

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

RTN-beskrivning: $3 \cdot A - 5 \cdot (B + 1) \rightarrow B$
(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 innehållet 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	S-term	RTN-beskrivning	Styrsignaler
Q ₅	Q ₅ ·I _{xx}	X→MA	
Q ₆	Q ₆ ·I _{xx}	M→T	
Q ₇	Q ₇ ·I _{xx}	B OR T→R, Flaggor→CC	
Q ₈	Q ₈ ·I _{xx}	R→B, 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.

IBNE X,Adr RTN: $X + 1 \rightarrow X$, Flags $\rightarrow$ CC

If Z = 0: PC + offset $\rightarrow$ 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

(4p)

6. Besvara kortfattat följande frågor rörande FLEX-processorn.

- a) Stackpekaren är en laddningsbar upp/ner-räknare. Vad är skälet till detta? (2p)
- b) Vad har X- registret för funktion? Redogör för när X-registret används (2p)
- c) Hur många tillstånd får EXECUTE-fasen innehålla om styrenheten med fast logik används. Motivera svaret. (2p)
- d) Subrutinen nedan anropas med datavärdena 10H och 11H i A- resp. B-registret. Vad innehåller dessa register vid återhoppet från subrutinen?

```

        ORG    $30
SUBR    CLR    VAR1
        CLR    VAR2
        TSTA
        BEQ    EXIT
        TSTB
        BEQ    EXIT
        STAB   VAR2
        CLRB
LOOP    ADDB   VAR2
        BCC    GOON
        INC    VAR1
GOON    DECA
        BNE    LOOP
        STAB   VAR2
EXIT    LDAA   VAR1
        LDAB   VAR2
        RTS
VAR1    RMB    1
VAR2    RMB    1

```

(3p)

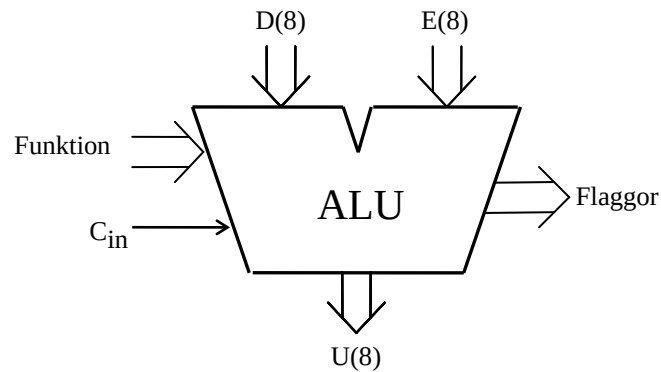
- e) Översätt subrutinen i d) till maskinkod. (3p)
- f) Hur lång tid tar subrutinen i d) att utföra om klockfrekvensen är 1MHz? (3p)

7. Skriv en subrutin i assemblerpråk för FLEX-processorn som tar reda på om bit 6 – bit 0 i register A motsvarar ASCII-tecknet för en hexadecimal siffra, dvs 0-9 eller A-F (endast stora bokstäver). Om så är fallet skall siffervärdet i binär form finnas i A-registret vid återhopp. Annars skall A-registret ha värdet FFH vid återhopp. ASCII-tabell finns i tabell 2 på sidan 7.

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

Bilaga 1

ALU:ns funktion



ALU:ns **logik-** och **aritmetikoperationer** på indata **D** och **E** definieras av ingångarna **Funktion (F)** och **C_{in}** enligt tabellen nedan. **F = (f₃, f₂, f₁, f₀)**.

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.

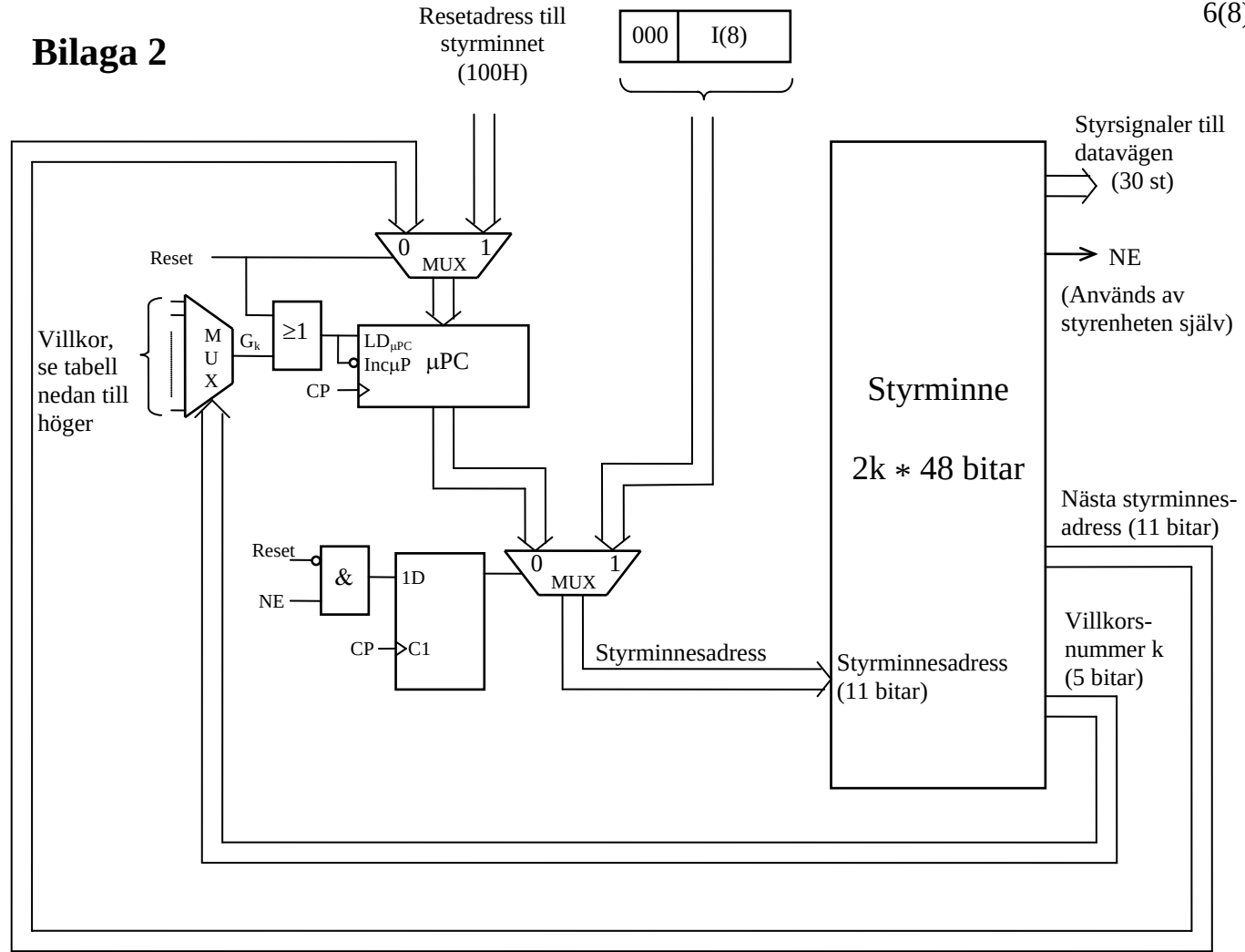
Half-carryflaggan (H) är minnessiffran (carry) mellan de fyra minst signifikanta och de fyra mest signifikanta bitarna i ALU:n.

H-flaggans värde är 0 vid andra operationer än aritmetiska.

I tabellen ovan avser "+" och "-" **aritmetiska operationer**.

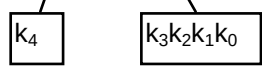
Med t ex **D_{1k}** menas att samtliga bitar i **D** inverteras.

Bilaga 2



Mikroprogrammerad EXECUTE-sekvens för instruktionen DEC Adr:

CM-adress (hex)	Hoppvillkor		Hopp-adress (hex)	RTN	Styrsignaler (aktiva)
	Kod (hex)	G _K			
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



$$K = k_4k_3k_2k_1k_0$$

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

$k_3k_2k_1k_0$	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	$N \oplus V$
1 0 1 1	$Z + (N \oplus V)$
1 1 0 0	0
1 1 0 1	0
1 1 1 0	0
1 1 1 1	1

Bilaga 3

Assemblerspråket för FLEX-processorn.

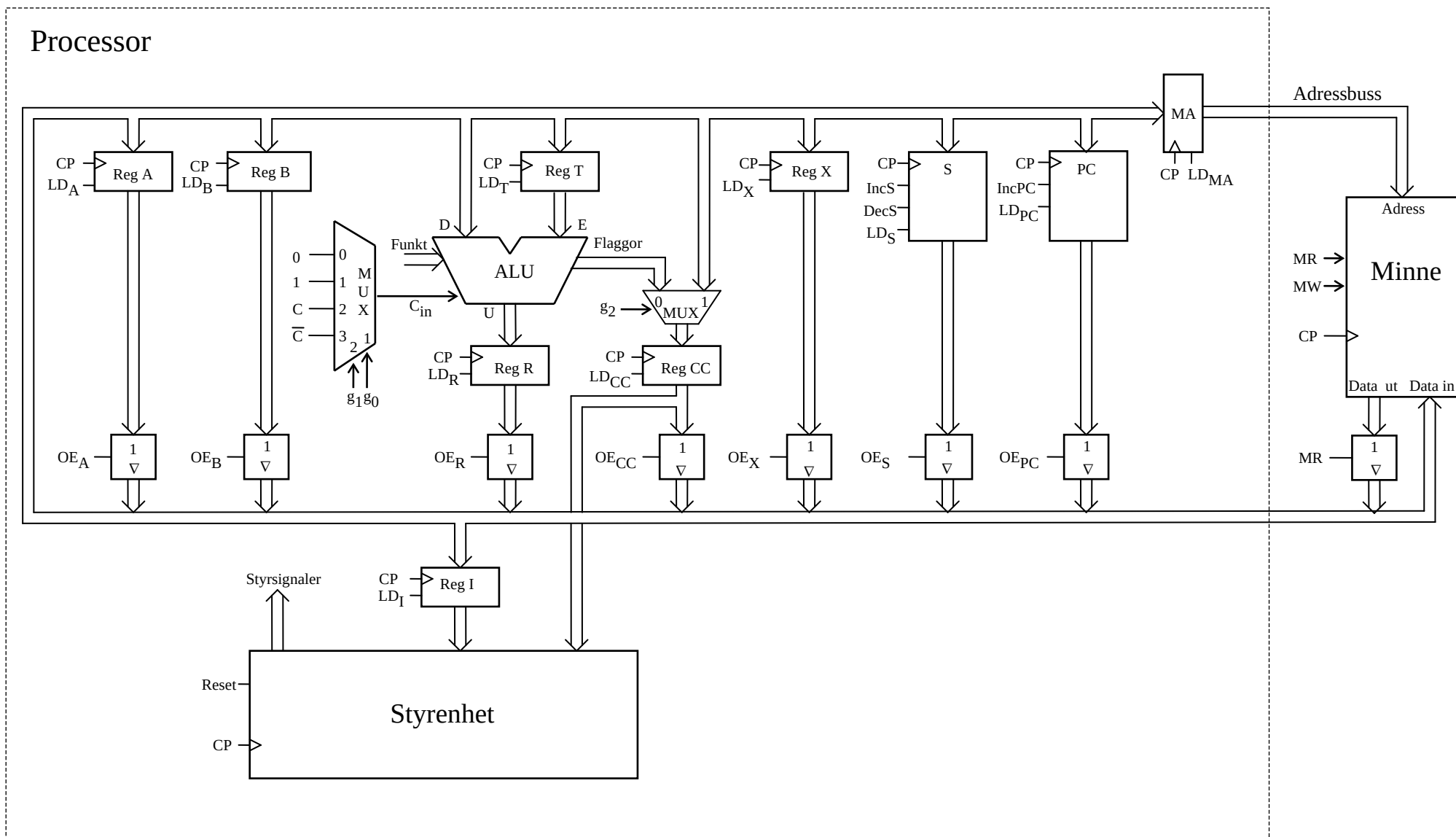
Assemblerspråket använder sig av mnemoniska beteckningar liknande dem som processorkonstruktören MOTOROLA specificerat för maskininstruktioner för mikroprocessorerna 68XX och instruktioner till assemblatorn, såsom pseudoinstruktioner eller assemblatordirektiv. Pseudoinstruktionerna listas i tabell 1.

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	lÖ	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.