

**TENTAMEN**

KURSNAMN	Digital- och datorteknik
PROGRAM:	Elektro Åk 1/ lp 4
KURSBETECKNING	EDA216
EXAMINATOR	Lars-Eric Arebrink
TID FÖR TENTAMEN	2013-08-22 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	Lennart Hansson, tel. 772 1681 ca. 10.30
ANSLAG AV RESULTAT	När rättningen är färdig anslås resultatet med anonyma koder och tid för granskning på kursens hemsida.
ÖVRIG INFORM. BETYGSGRÄNSER. SLUTBETYG	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. 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-h nedan används 5-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. $X = 11011_2$ och $Y = 10111_2$.
- 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 penna och papper hur räkneoperationen $S = X + Y$ utförs i en 5-bitars ALU. (1p)
 - d) Vilka värden får flaggbitarna N , Z , V och C vid räkneoperationen i c)? (1p)
 - e) Visa med penna och papper hur räkneoperationen $D = X - Y$ utförs i en 5-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 upp) talet $C400\ 0000_{16}$, som är ett 32-bitars flyttal enligt standarden IEEE 754 (dvs. med 23 bitar av mantissan) till sin decimala motsvarighet. (2p)

2.

- a) Den distributiva lagen $x + (y \cdot z) = (x+y)(x+z)$ för "ELLER" gäller i boolesk algebra.

Gäller motsvarande lag $x \oplus (y \cdot z) = (x \oplus y)(x \oplus z)$ för "Exklusivt ELLER"?

Motivera svaret!

(3p)

- b) En boolesk funktion $f(a,b,c,d)$ har karnaughdiagrammet till höger. Realisera funktionen med så få grindar som möjligt. NOR-grindar med valfritt antal ingångar och NOT-grindar får användas. Endast insignalerna a , b , c och d finns tillgängliga. Rutor med - representerar "dont care"-termer som får användas vid behov.

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

(4p)

3.

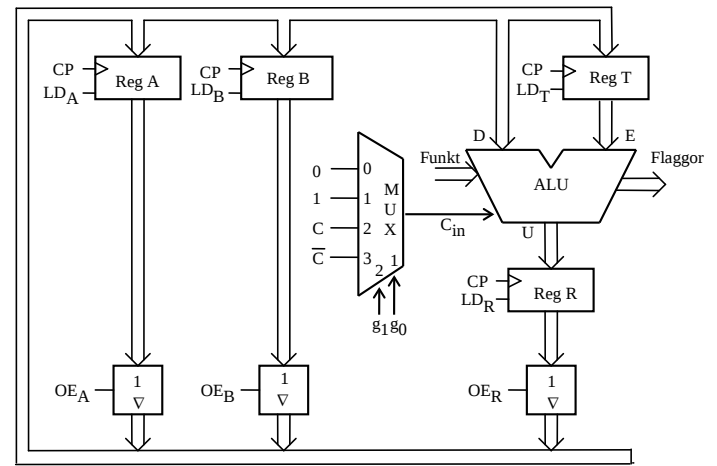
- a) Konstruera en T-vippa med hjälp av en SR-vippa och standardgrindar. (4p)
- b) Konstruera en 4-bitars autonom nedräknare. (Räknesekvens: 0, 15, 14, ..., 2, 1, 0, 15, 14, ...) Numrera tillstånden $q_3q_2q_1q_0$. T-vippor, AND-grindar med valfritt antal ingångar, XOR-grindar och NOT-grindar får användas.
- Ledning:** Det finns ett enkelt villkor som avgör om en bit i det binära talet $(q_3q_2q_1q_0)_2$ skall ändras från 0 till 1 (eller tvärt om) vid nedräkningen! (7p)

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

Rita av tabellen!

Fyll sedan i 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!



CP	Styrsignaler (=1)	RTN	Reg A(8)	Reg B(8)	Reg T(8)	Reg R(8)
1	OE _B , LD _T		64	C8	?	?
2	OE _A , f ₃ , f ₂ , 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 , g ₀ , LD _R					
6	OE _R , LD _B					
7						

(1+3p)

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 f₃ - f₀ och C_{in}.

I tabellerna nedan visas styrsignalerna för två olika EXECUTE-sekvenser för samma maskininstruktion, en för FLIS- och en för FLEX-processorn.

State	Summaterm	RTN-beskrivning	Styrsignaler
Q ₄	$Q_4 \cdot I_{xx}$		OE _{PC} , LD _{TA} , LD _T
Q ₅	$Q_5 \cdot I_{xx}$		MR, f ₃ , f ₁ , f ₀ , g ₀ , g ₁₄ , LD _R , IncPC
Q ₆	$Q_6 \cdot I_{xx} \cdot (C+Z)'$		OE _R , LD _{PC}
	$Q_6 \cdot I_{xx}$		NF

- a) Rita en tabell med "State"- och RTN-kolumner enligt ovan (FLISP) eller nedan (FLEX) för en av instruktionerna och fyll i RTN-beskrivningen! Förklara vilken assemblerinstruktion som beskrivs!

Som alternativ kan du använda FLEX-datorn som visas i bilaga 3 med ALU'n i bilaga 1 och tabellen nedan istället för FLIS-datorn.

State	Summaterm	RTN-beskrivning	Styrsignaler
Q ₅	$Q_5 \cdot I_{xx}$		OE _{PC} , LD _{MA} , LD _T
Q ₆	$Q_6 \cdot I_{xx}$		MR, f ₃ , f ₁ , g ₀ , LD _R , IncPC
Q ₇	$Q_7 \cdot I_{xx} \cdot (C+Z)'$		OE _R , LD _{PC}
	$Q_7 \cdot I_{xx}$		NF

(2p)

5(forts.)

- b) Instruktionen "Add memory bytes" nedan skall implementeras för FLIS- eller FLEX-processorn med hjälp av styrenheten med fast logik. Instruktionen består av tre ord. Den utför addition av dataorden $M(\text{Adr1})$ och $M(\text{Adr2})$. Summan placeras som $M(\text{Adr2})$. Flaggorna skall påverkas som vid en vanlig addition.

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

ADDM Adr1,Adr2 RTN: $M(\text{Adr1}) + M(\text{Adr2}) \rightarrow M(\text{Adr2})$
Flags $\rightarrow$ CC

OPKOD
Adr1
Adr2

Gör en tabell liknande tabellen i uppgift a) för den efterfrågade EXECUTE-sekvensen. **(4p)**

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

- a) Förklara varför det kan vara en fördel att använda instruktionen BRA Adr istället för JMP Adr. **(1p)**
- b) Före de villkorliga hoppen BHI och BGT i ett program utförs en subtraktion som påverkar flaggorna. Förklara vad som händer för vardera hoppet om subtraktionen före är $76_{16} - 81_{16}$. **(2p)**

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

- d) Subrutinen i c) anropas i FLISP-programmet (eller motsvarande FLEX-program) nedan.

```

ORG    $30
LDSP   #$F0
LDA     #$F6
BSR     WAIT
STOP    BRA    STOP

```

NUMB	EQU	4
;		
WAIT	ORG	\$80
;	PSHA	
LOOP1	PSHA	
;		
LOOP2	LDA	#NUMB
	DECA	
	BPL	LOOP2
	PULA	
;		
	INCA	
	BNE	LOOP1
;		
DELX	PULA	
	RTS	

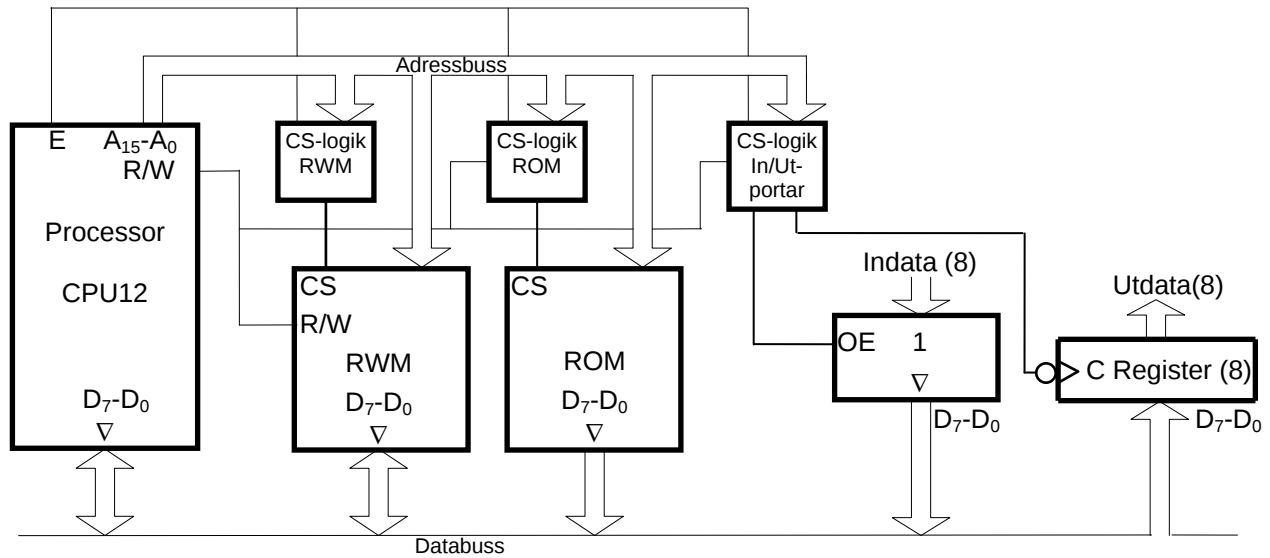
Hur lång tid tar subrutinen att köra från och med BSR till och med RTS om processorn klockas med frekvensen 1MHz? **(3p)**

7. I minnet i ett datorsystem med FLIS- eller FLEX-processorn finns en sträng med sju bitars ASCII-tecken. Strängen avslutas med en byte (slutmarkering) med värdet 0. Varje ASCII-tecken i strängen, utom slutmarkeringen, har kompletterats med en paritetsbit som åttonde bit (bit nr 7).

Skriv en subrutin DECCNT i assemblerpråk för processorn som tar reda på hur många av ASCII-tecknen i strängen som motsvarar decimala siffror (dvs. 0-9). Antalet decimala siffror skall finnas i A-registret vid återhopp. Vid anrop av subrutinen förutsätts att startadressen till strängen finns i X-registret.

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

8. Ett datorsystem visas nedan:



Figuren ovan visar principen för anslutning av externa minnesmoduler och externa in-/utportar till processorn CPU12.

En 64 kbyte ROM-modul skall i princip vara placerad från adress 0, men de första 256 adresserna i adressrummet skall inte aktivera någon minnesmodul eller port vid läsning eller skrivning.

En inport och en utport skall också anslutas. De skall ha samma adress och placeras direkt efter de första 256 adresserna i adressrummet. Det innebär att inporten skall prioriteras före ROM-modulen på denna adress.

En 4 kbyte RWM-modul skall dessutom placeras med start på adressen E000₁₆ och prioriteras före ROM-modulen.

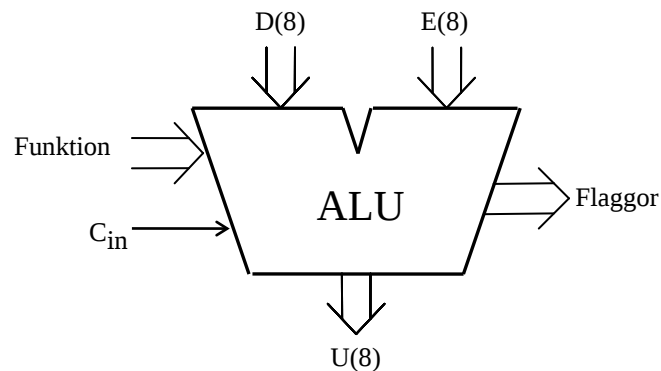
Rita CS-logiken för minnesmodulerna och portarna. Använd fullständig adressavkodning. Endast grundläggande logikgrindar med valfritt antal ingångar får användas. Visa inom vilka adressintervall de olika modulerna kan nås.

Ledning: Adressområdet som omfattar de första 256 adresserna kan betraktas som en modul med 256 olika adresser.

(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 **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

Assemblerspråket för FLEX- och FLIS-processorn.

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

Tabell 1

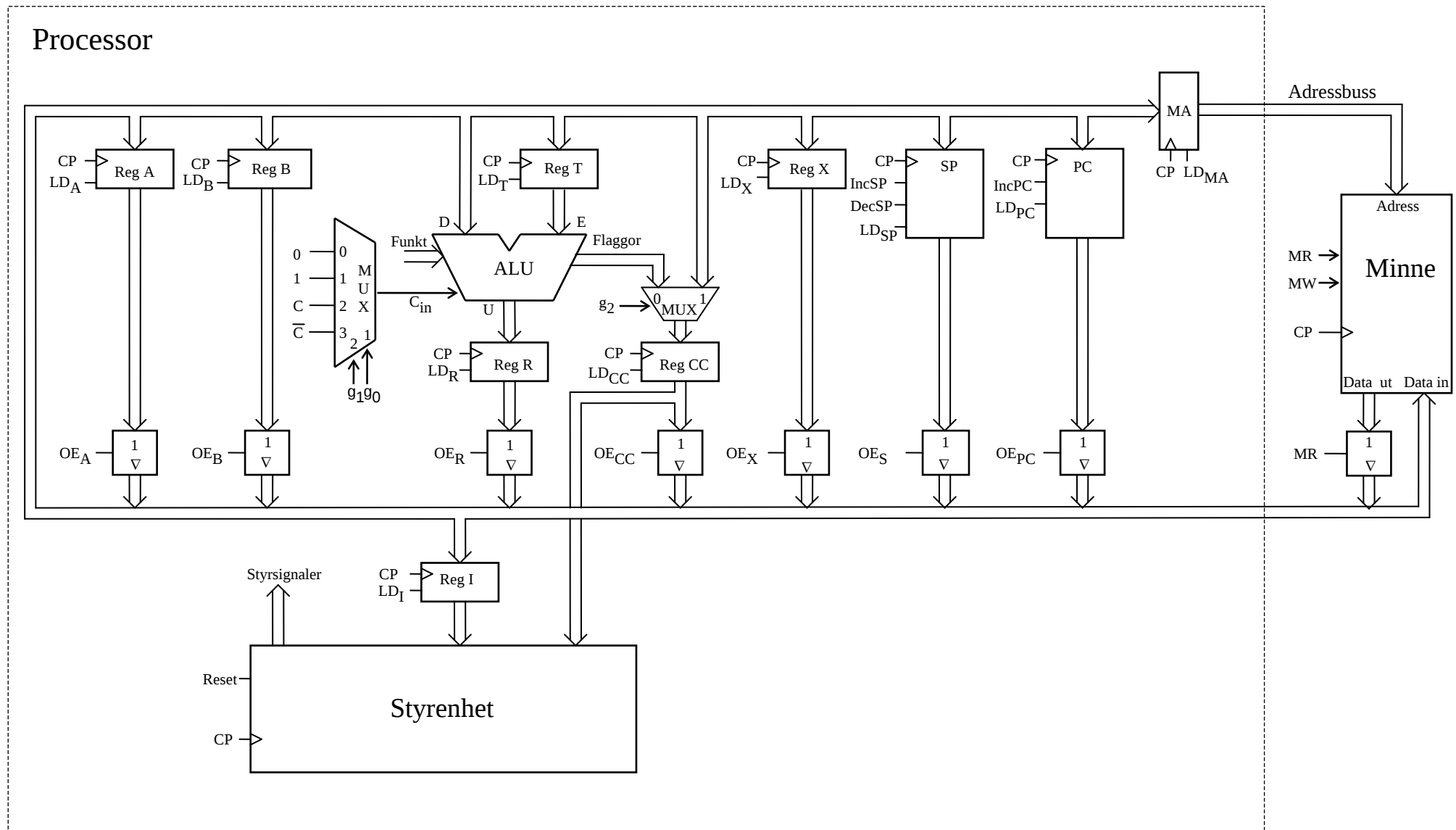
Direktiv	Förklaring
ORG N	Placerar den efterföljande koden med början på adress N. (ORG för ORiGin = ursprung)
L RMB N	Avsätter N bytes i följd i minnet (utan att ge dem värden), så att programmet kan använda dem. Följden placeras med början på adressen L. (RMB för Reseve Memory Bytes)
L EQU N	Ger symbolen 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 ₆ b ₅ b ₄ b ₃ b ₂ b ₁ b 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	{	1 0 1 1
FF	FS	,	<	L	\	ö		1 1 0 0
CR	GS	-	=	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)

8(8)



Figur 1. Datorn FLEX.