CHALMERS

Institutionen för data- och informationsteknik
Avdelningen för datorteknik

TENTAMEN

KURSNAMN	**Digital- och datorteknik**
PROGRAM:	**Elektro** **Åk 1/ lp 4**
KURSBETECKNING	**EDA216**
EXAMINATOR	**Lars-Eric Arebrink**
TID FÖR TENTAMEN	**2013-01-14 kl 8.30 – 12.30**
HJÄLPMEDEL	**Av institutionen utgiven** **”Instruktionslista för FLEX-processorn” (INS1)** **Tabellverk eller miniräknare får ej användas.**
ANSV LÄRARE: **Besöker tentamen**	**Lennart Hansson, tel. 772 1681** **vid flera tillfällen**
ANSLAG AV RESULTAT	**När rättningen är färdig anslås resultatet med anonyma koder och tid för granskning på kursens hemsida.** **Lägg din anonyma kod på minnet! Den används när resultatet anslås.**
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-h nedan används 7-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. $X = 0110011_2$ och $Y = 1001101_2$

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 7-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 7-bitars ALU. **(1p)**

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

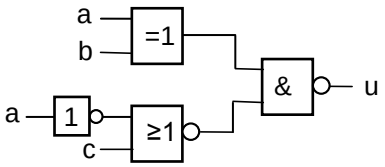
g) Tolka bitmönstren X, Y, S och D som tal *utan* tecken och ange deras decimala motsvarighet. Avgör och motivera med hjälp av flaggorna om resultaten S och D är korrekta eller felaktiga. **(1p)**

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) I kurslitteraturen bekrivs ett antal olika BCD-koder. En av dessa är "Excess-3-Gray"-koden. Förklara i vilka sammanhang och varför denna då bör användas. **(2p)**

j) När man översätter (packar upp) ett 32-bitars flyttal som är givet enligt flyttalsstandarden IEEE 754 (dvs. 23 bitar av mantissan) till decimal form får man ett antal värdesiffror (giltiga siffror). Visa approximativt hur många dessa är! **(2p)**

k) Ge ett minimalt boolesk PS-uttryck för u i grindnätet till höger. **(3p)**



2. En boolesk funktion f(a,b,c,d) har karnaughdiagrammet till höger.

Realisera funktionen med så få grindar som möjligt. NAND-grindar med valfritt antal ingångar, XOR-grindar och NOT-grindar får användas. Endast insignalerna a, b, c och d finns tillgängliga.

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

(4p)

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 som består av antingen en etta följd av två nollor eller en nolla följd av två ettor hos x.

Exempel: σ_x = ...0100111100110000101001...

σ_u = ...?0010100010101000000010...

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. Hur många vippor skulle minst krävas vid realiseringen? **(5p)**

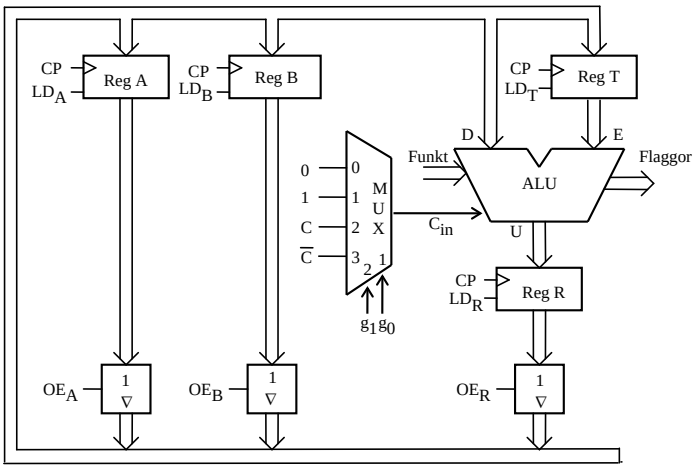
b) Realisera en autonom räknare med räknesekvensen $q_2q_1q_0$: 000, 101, 010, 111, 100, 001, 110, 011, 000, … T-vippor, NAND-grindar med valfritt antal ingångar och NOT-grindar får användas. **(6p)**

4. I datavägen till höger innehåller register A och B från början värdena 50_{10} resp. 30_{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.

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



CP	Styrsignaler (=1)	RTN	Reg A(8)	Reg B(8)	Reg T(8)	Reg R(8)
1	OE_B, LD_T		32	1E	?	?
2	OE_A, 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, g_0, LD_R					
6	OE_R, LD_A					
7	?					

(4p)

5. Figur 1 i bilaga 3 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 styrsignalerna för de olika tillstånden i EXECUTE-sekvensen för en av FLEX-processorns instruktioner.

State	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	$Q_9 \cdot I_{xx}$		OE_A, MW, NF

a) Rita en tabell med ”State”- och RTN-kolumner enligt ovan och fyll i RTN-beskrivningen. Förklara vilken assemblerinstruktion som beskrivs. **(2p)**

b) Instruktionen ”Test and branch if zero” nedan skall implementeras för FLEX-processorn med hjälp av styrenheten med fast logik. Instruktionen består av två ord. Den testar innehållet i aktuellt register och påverkar flaggorna. Om Z-flaggan får värdet 1 så utförs ett programräknarrelativt hopp till Adress.

Tänk på att programräknarrelativa hopp alltid utförs från adressen till nästa OP-kod.

Register A, B, X eller SP får ej påverkas. Operationskoden FD_{16} skall användas.

OPKOD
offset

TBEQ X,Adress RTN: U = X + 0, Flags → CC
If Z = 1: PC + offset → PC

Gör en tabell liknande tabellen ovan för den efterfrågade EXECUTE-sekvensen. **(5p)**

6. Besvara kortfattat följande frågor rörande FLEX-processorn.

a) Vad är det i styrenheten som bestämmer det maximala antalet klockpulser som kan användas i EXECUTE-fasen av en instruktion? **(2p)**

b) Vilket är det maximala antalet instruktioner som kan finnas i FLEX-processorns instruktions-repertoar? **(1p)**

c) Före de villkorliga hoppen BPL och BLE i ett program utförs en subtraktion som påverkar flaggorna. Förklara vad som händer för vardera hoppet om subtraktionen före är $A6_{16}$ - 75_{16}. **(2p)**

d) Översätt instruktionssekvensen till höger till maskinkod på hexadecimal form och visa hur den placeras i minnet. Det skall framgå hur offset för branch-instruktionerna beräknas. **(3p)**

```
BVAL    EQU   2
*
        ORG   $30
DELAY   PSHA
        PSHB
        PSHX
*
        LDX   #DETAB
        LDAA  A,X
LOOPA   LDAB  #BVAL
*
LOOPB   DECB
        BNE   LOOPB
*
        DECA
        BNE   LOOPA
*
DEXIT   PULX
        PULB
        PULA
        RTS
*
DETAB   FCB   5,10,20,50,100,200
```

e) Subrutinen i d) anropas i huvudprogrammet nedan.

```
        LDS   #$FC
        LDAA  #4
        JSR   DELAY
STOP    BRA   STOP
```

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

f) Skriv ett program i assemblerspråk som subtraherar två 16-bitars tal P och Q på adresserna 60_{16} och 62_{16} i minnet enligt figuren till höger och bildar skillnaden R enligt: R_{HB}:R_{LB} = P_{HB}:P_{LB} – Q_{HB}:Q_{LB}.

Programmet skall placeras med start på adress 80_{16}, Skillnaden R skall placeras i minnet enligt figuren till höger.

Översätt programmet till maskinkod på hexadecimal form och visa hur det placeras i minnet.

Adr(hex)	Data
60	P_{HB}
61	P_{LB}
62	Q_{HB}
63	Q_{LB}
64	R_{HB}
65	R_{LB}

(3)

7. Skriv en subrutin GRAY2BIN i assemblerspråk för FLEX-processorn som översätter ett graykodat 4-bitars ord i bit 3-0 i register A till motsvarande binära tal. Innehållet i bit 7-4 i A-registret är okänt.

Vid återhopp skall det binära talet finnas i bit 3-0 i register A med bitarna 7-4 nollställda.

Tabellen nedan innehåller samtliga 16 graykoder för talen 0-15, dvs 0000_2 – 1111_2. Tabellen, med start på adressen GTAB, finns tillgänglig på binär form i minnet med graykoden i bitarna 3-0 och bitarna 7-4 nollställda i varje dataord.

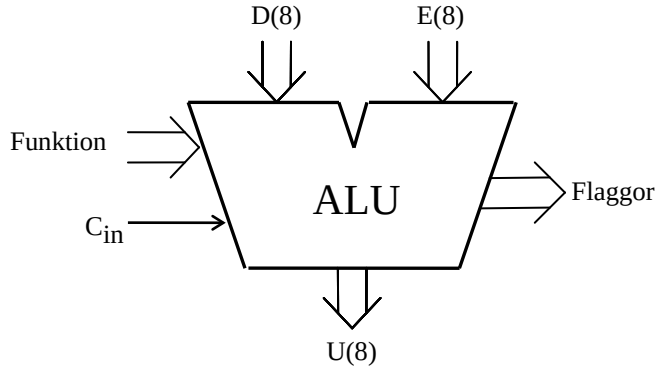
GTAB: 0,1,3,2,6,7,5,4,$C,$D,$F,$E,$A,$B,9,8

Vid återhopp får endast register A och CC får vara förändrade.

För full poäng på uppgiften skall subrutinen vara korrekt radkommenterad. **(5p)**

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

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-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.

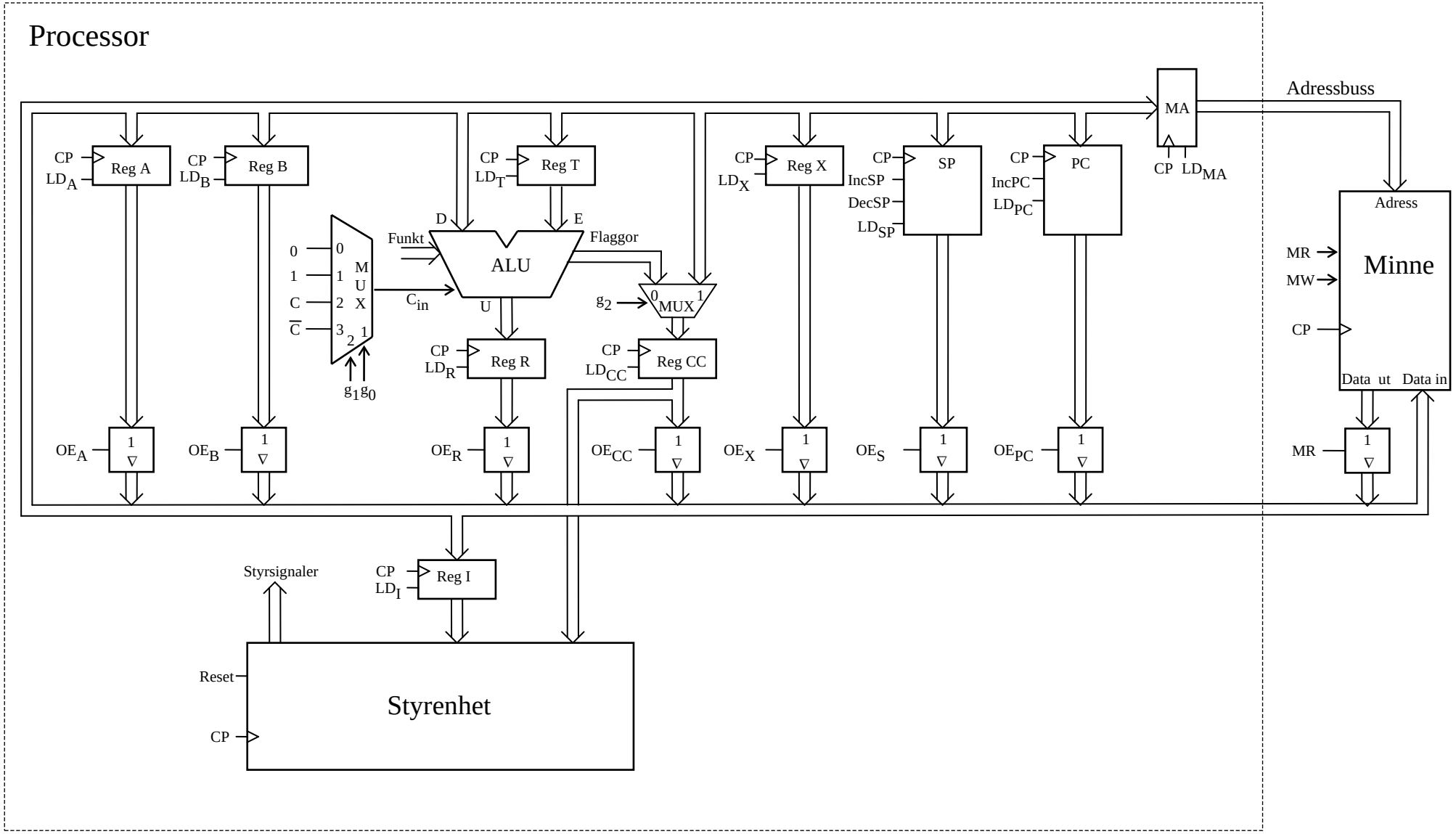
Tabell 1

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



Figur 1. Datorn FLEX.