CHALMERS

Institutionen för data- och informationsteknik
Avdelningen för datorteknik

TENTAMEN

KURSNAMN	**Digital- och datorteknik**
PROGRAM:	**Mekatronik-, data- och elektroingenjör** **Åk 1/ lp 1o2**
KURSBETECKNING	**LEU431**
EXAMINATOR	**Lars-Eric Arebrink**
TID FÖR TENTAMEN	**2012-12-21 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.	**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-h nedan används 8-bitars tal X, Y, S och D. X = 11011001 och Y = 01010011.

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 8-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 8-bitars ALU. **(1p)**

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

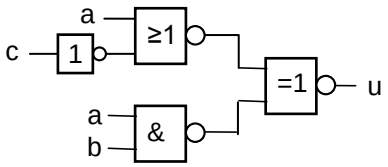
g) Tolka bitmönstren X, Y, S och D som tal *utan* tecken och ange deras decimala motsvarighet. Avgör och motivera med hjälp av flaggorna om resultaten S och D är korrekta eller felaktiga. **(1p)**

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) Genomför subtraktionen $196_{10} - 320_{10}$ med 10-komplementaritmetik. Hur många decimala sifferpositioner behövs? Hur skall man tolka resultatet? **(2p)**

j) Översätt (packa upp) flyttalet $424B8000_{16}$, som är givet enligt flyttalsstandarden IEEE 754-1985 (dvs. 23 bitar av mantissan) till decimal form. **(2p)**

k) Ge ett ”minimalt” boolesk SP-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.

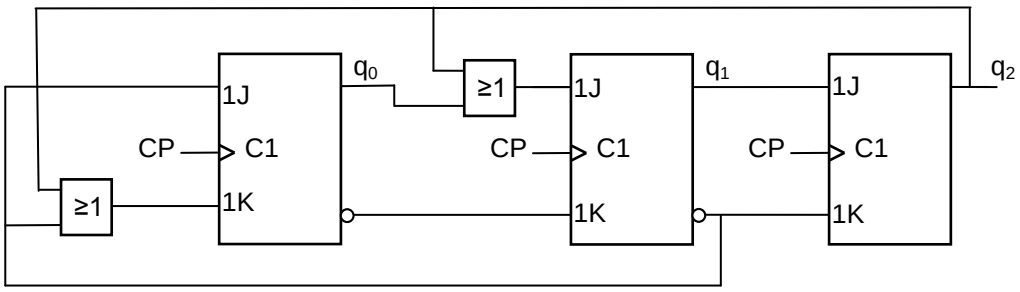
ab \ cd	00	01	11	10
00	1	1	1	1
01	0	0	1	1
11	1	1	0	0
10	0	0	1	1

(4p)

3.

a) Rita och fyll i T-vippans funktionstabell. **(1p)**

b) Tag fram en fullständig tillståndsgraf för räknaren nedan med tillstånden numrerade $q_2q_1q_0$. Med fullständig menas att samtliga tillstånd finns med.



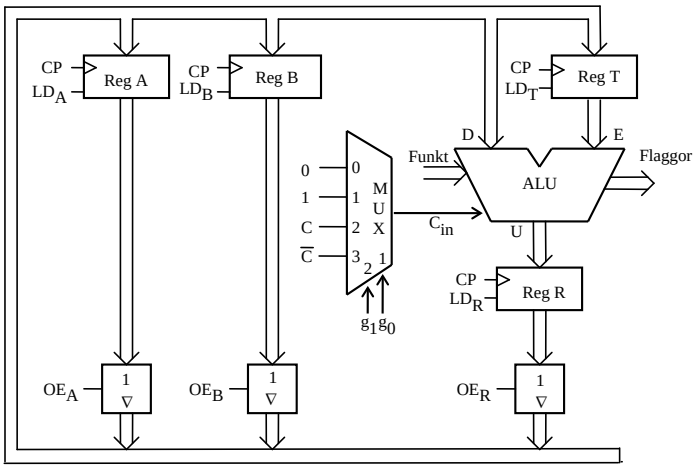
(5p)

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

RTN-beskrivning: $7 \cdot A - 5 \cdot B \rightarrow A$
(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 innehållen i register R och T är okända. Register B får inte ändras. Bortse från risken för overflow. Använd så få tillstånd som möjligt. Samtliga funktioner ALU:n kan utföra framgår av bilaga 1.



(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 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_A, f_3, f_1, LD_R , LD_{CC}
Q_8	$Q_8 \cdot I_{xx}$		OE_R, LD_A, NF

Styrsignalen NF i tabellens sista rad medför att nästa tillstånd blir det första i FETCH-sekvensen.

a) Ge RTN-beskrivningen för alla tillstånden i tabellen och förklara vilken assemblerinstruktion som beskrivs. **(3p)**

b) Instruktionen "Add to Memory" nedan skall implementeras för FLEX-processorn med hjälp av styrenheten med fast logik. Instruktionen består av tre ord. Den läser dataordet på adressen efter operationskoden i minnet och adderar det till minnesordet på adressen Adr som finns som data i instruktionens tredje ord. Flaggregistret får nya värden av additionen.
Register A, B, X eller SP får ej påverkas. Operationskoden FE_{16} skall användas.

ADDM #Data,Adr RTN: Data + M(Adr) → M(Adr), Flags → CC

OPKOD
Data
Adr

Gör en tabell liknande tabellen ovan för den efterfrågade EXECUTE-sekvensen.

(4p)

6. Besvara kortfattat följande frågor rörande FLEX-processorn.

Tips: Använd figur 1 i bilaga 3 som stöd när du löser uppgifterna nedan!

a) Vilken uppgift har RESET-sekvensen? **(2p)**

b) Vilken uppgift har programräknaren (PC) i FLEX-datorn? **(2p)**

c) För vilka värden på talet W ($0 \leq W \leq 255$) utförs hoppet nedan?

```
LDAA    #W
CMPA    #$80
BHI     Hopp
```
(2p)

d) För vilka värden på talet W ($0 \leq W \leq 255$) utförs hoppet nedan?

```
LDAA    #W
CMPA    #$40
BLE     Hopp
```
(3p)

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

```
XVAL    EQU   -10
*
        ORG   $80
TIME    PSHA
        PSHX
*
LOOP    LDX   #XVAL
*
LOOPX   INX
        NOP
        BNE   LOOPX
*
        DECA
        BEQ   TEXIT
        BRA   LOOP
*
TEXIT   PULX
        PULA
        RTS
```

f) Subrutinen i e) anropas i huvudprogrammet nedan.

```
        LDS   #$FC
        LDAA  #5
        JSR   TIME
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)**

g) Skriv ett program i assemblerspråk som adderar två 16-bitars tal P och Q i minnet och bildar summan R enligt: $R_{HB}:R_{LB} = P_{HB}:P_{LB} + Q_{HB}:Q_{LB}$.

Programmet skall placeras med start på adress 20_{16}, Summan R, som antas rymmas i 16 bitar, 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
40	P_{HB}
41	P_{LB}
42	Q_{HB}
43	Q_{LB}
44	R_{HB}
45	R_{LB}

(3)

7. I simulatorn för FLEX-datorn kan man ansluta strömbrytarmodulen DIPSWITCH till en inport och ljusdiodsindikatorn LED till en utport. På DIPSWITCH kan man ställa in ett 8-bitars datavärde och på LED kan man visa ett 8-bitars dataord som ett binärt tal.

Skapa ett program som kan visa ”rinnande ljus” på ljusdiodsindikatorn LED. ”Flödesriktningen” skall vara valbar (<- -> vänster/höger). Använd en ”bitmönstertabell” för att lösa uppgiften.

Skriv ett program med startadress 30_{16} i assemblerspråk för FLEX-datorn som fungerar enligt beskrivningen nedan:

1. Sätt stackpekaren till värdet $F0_{16}$.
2. Läs av DIPSWITCH från inporten på adress FD_{16}.
3. Om värdet på DIPSWITCH är 0------1_2, dvs. b_7=0 och b_0=1, skall ljuset ”rinna” åt höger.

   Om värdet på DIPSWITCH är 1------0_2, dvs. b_7=1 och b_0=0, skall ljuset ”rinna” åt vänster.

   För alla andra värden på DIPSWITCH skall ljuset ”stå stilla”.
4. Visa bitmönstret på LED-indikatorn via utporten på adress FE_{16}.
5. Anropa en färdig subrutin DELAY.
6. Hoppa tillbaka till 2.

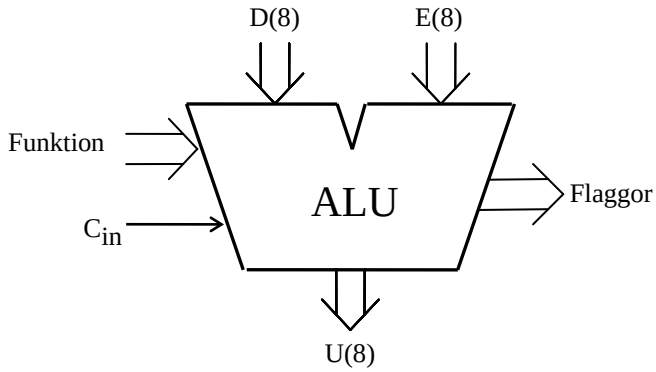
Följande bitmönstertabell skall börja på adressen 90_{16}: $01,$02,$04,$08,$10,$20,$40,$80

Bitmönstertabellen skall laddas i minnet tillsammans med programmet.

Korrekta radkommentarer skall finnas för full poäng! **(6p)**

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** = **(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.

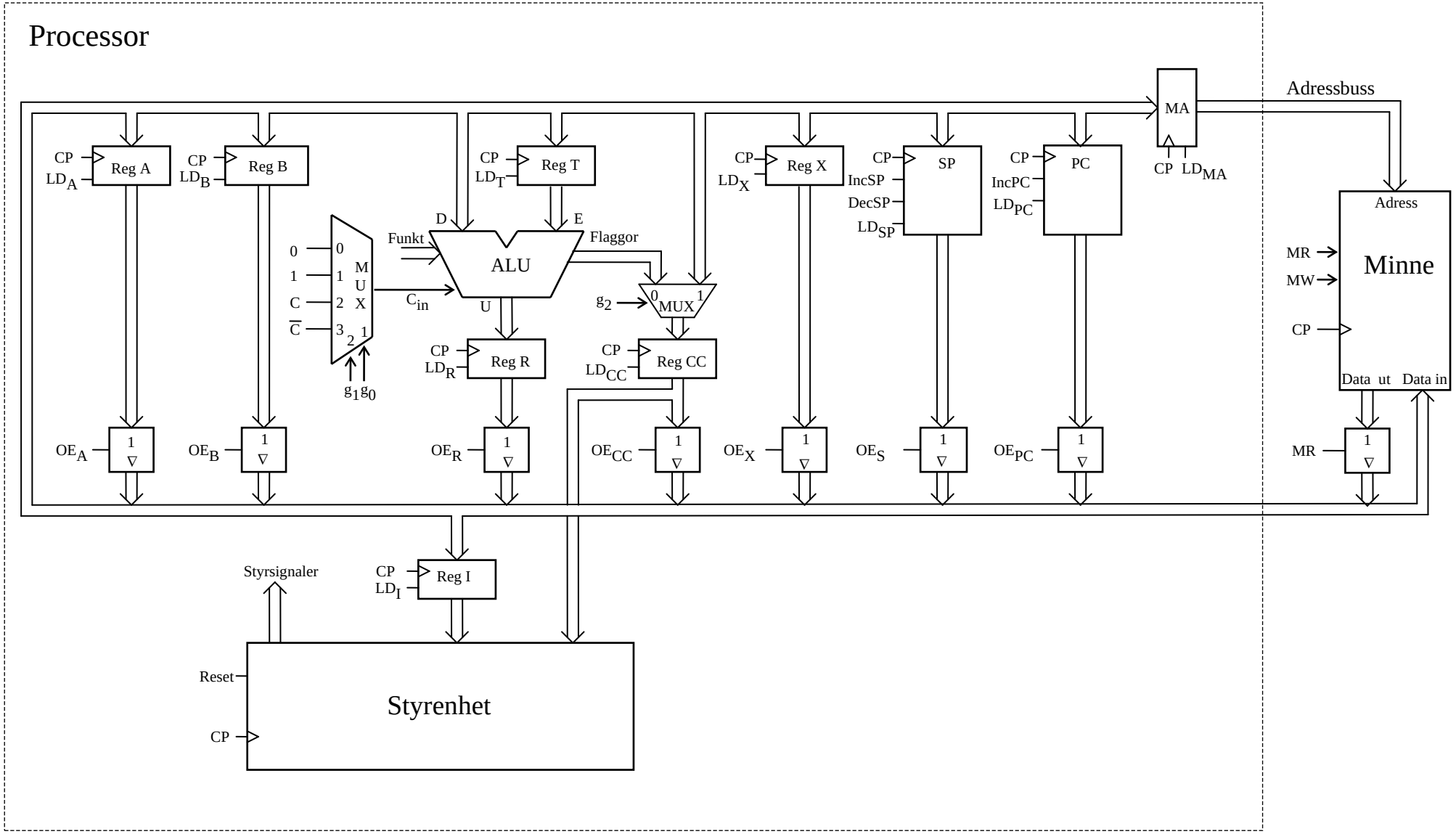
Tabell 1

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.