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	**2012-05-24 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	**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.** **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. X = 10011 och Y = 01101.

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)**

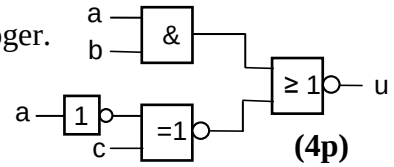
g) Tolka bitmönstren X, Y, S och D som tal *utan* tecken och ange deras decimala motsvarighet. Avgör och motivera med hjälp av flaggorna om resultaten S och D är korrekta eller felaktiga. **(1p)**

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) Flera av de olika binära koderna för representation siffrorna i decimala tal har egenskapen att det är enkelt att ta fram koden för 9-komplementet av varje siffra. Förklara varför denna egenskap är viktig. **(1p)**

j) Talet $C3C8D000_{16}$ är ett 32-bitars flyttal packat enligt IEEE-standard 754 (23 bitar av mantissan). Skriv talet på binär och decimal form. **(2p)**

k) Tag fram de två minimala booleska PS-uttrycken för u i grindnätet till höger.



(4p)

2. 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.

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

(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 ”exakt” en etta följd av två nollor och två ettor hos x. Med ”exakt” menas här att det inte får vara fler än en etta.

Exempel: σ_x = ...0100110011001001101000100100 11...

σ_u = ...0000010000000000100000000000001...

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? **(4p)**

b) Realisera en autonom räknare med räknesekvensen $q_2q_1q_0$: 000, 001, 011, 010, 110, 111, 101, 100, 000, …

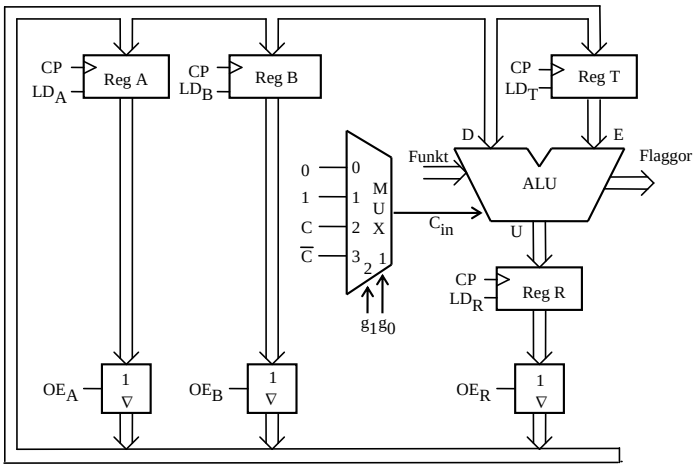
D-vippor, NAND-grindar med valfritt antal ingångar, XOR-grindar och NOT-grindar får användas. **(6p)**

4. I datavägen till höger innehåller register A och B från början värdena 27_{10} resp. 50_{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	Reg B	Reg T	Reg R
1	OE_B, LD_T		1B	32	?	?
2	OE_A, f_3, f_1, g_0, LD_R					
3	OE_R, f_3, f_1, f_0, LD_R, LD_B					
4	OE_R, LD_T					
5	OE_A, f_3, f_2, g_0, LD_R					
6	OE_R, f_2, f_1, LD_R					
7	OE_R, LD_A					
8	?					

(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}, LD_T
Q_6	$Q_6 \cdot I_{xx}$		MR, f_3, f_1, g_0, LD_R, IncPC
Q_7	$Q_7 \cdot I_{xx} \cdot ((N \oplus V)+Z)$		NF
	$Q_7 \cdot I_{xx} \cdot ((N \oplus V)+Z)'$		OE_R, LD_{PC}, 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 ”Branch if bits set” nedan skall implementeras för FLEX-processorn med hjälp av styrenheten med fast logik. Instruktionen består av fyra ord. Den utför bitvis AND mellan masken i instruktionen och 1-komplementet av dataordet på Adress1 och låter resultatet påverka flaggorna. Om Z-flaggan får värdet 1 så utförs ett programräknarrelativt hopp till Adress2.

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

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

OPKOD
Adr1
mask
offset

BRSET Adr1,#mask,Adr2 RTN: $M(Adr1)_{1k}$ *AND* mask
If Z = 1: PC + offset → PC
else: (*Next instruktion*)

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

(6p)

6. Besvara kortfattat följande frågor rörande FLEX-processorn.

a) Före de villkorliga hoppen BMI och BLT i ett program utförs en addition som påverkar flaggorna. Förklara vad som händer för vardera hoppet om additionen före är 87_{16} + $A3_{16}$. **(3p)**

b) 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
        BNE   LOOPX
*
        DECA
        BNE   LOOP
*
TEXIT   PULX
        PULA
        RTS
```

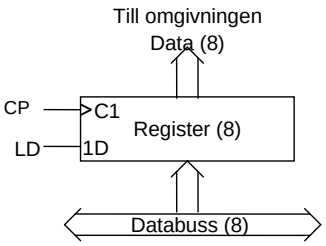
c) Subrutinen i b) anropas i huvudprogrammet nedan.

```
        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)

d) FLEX-datorn som visas i bilaga 3 saknar in- och utportar. Den skall nu kompletteras med en utport på adressen 0. Principen för en utport visas i figuren till höger. Konstruera ett grindnät som bildar signalen LD. Standardgrindar med valfritt antal ingångar får användas.



(2p)

7. I minnet i ett datorsystem med FLEX-processorn finns en nollterminerad sträng med sjubitars ASCII-tecken. Varje ASCII-tecken har en paritetsbit för udda paritet som bit nr 7.

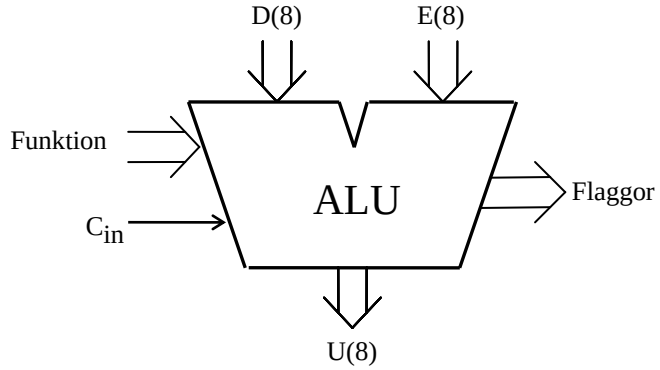
Skriv en subrutin HXCONV i assemblerpråk för FLEX-processorn som ersätter de ASCII-tecken i strängen som motsvarar hexadecimala siffror (0 - 9 och A – F) med motsvarande binära värden 0000-1111_2 (och skriver tillbaka dem i strängen). De ASCII-tecken som inte motsvarar hexadecimala siffror ersätts med värdet 80_{16} i strängen och den avslutande nollan (nollterminering) ersätts med värdet FF_{16} i strängen. Den avslutande nollan saknar paritetsbit.

Vid anrop av subrutinen skall startadressen till strängen finnas i X-registret.

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

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.

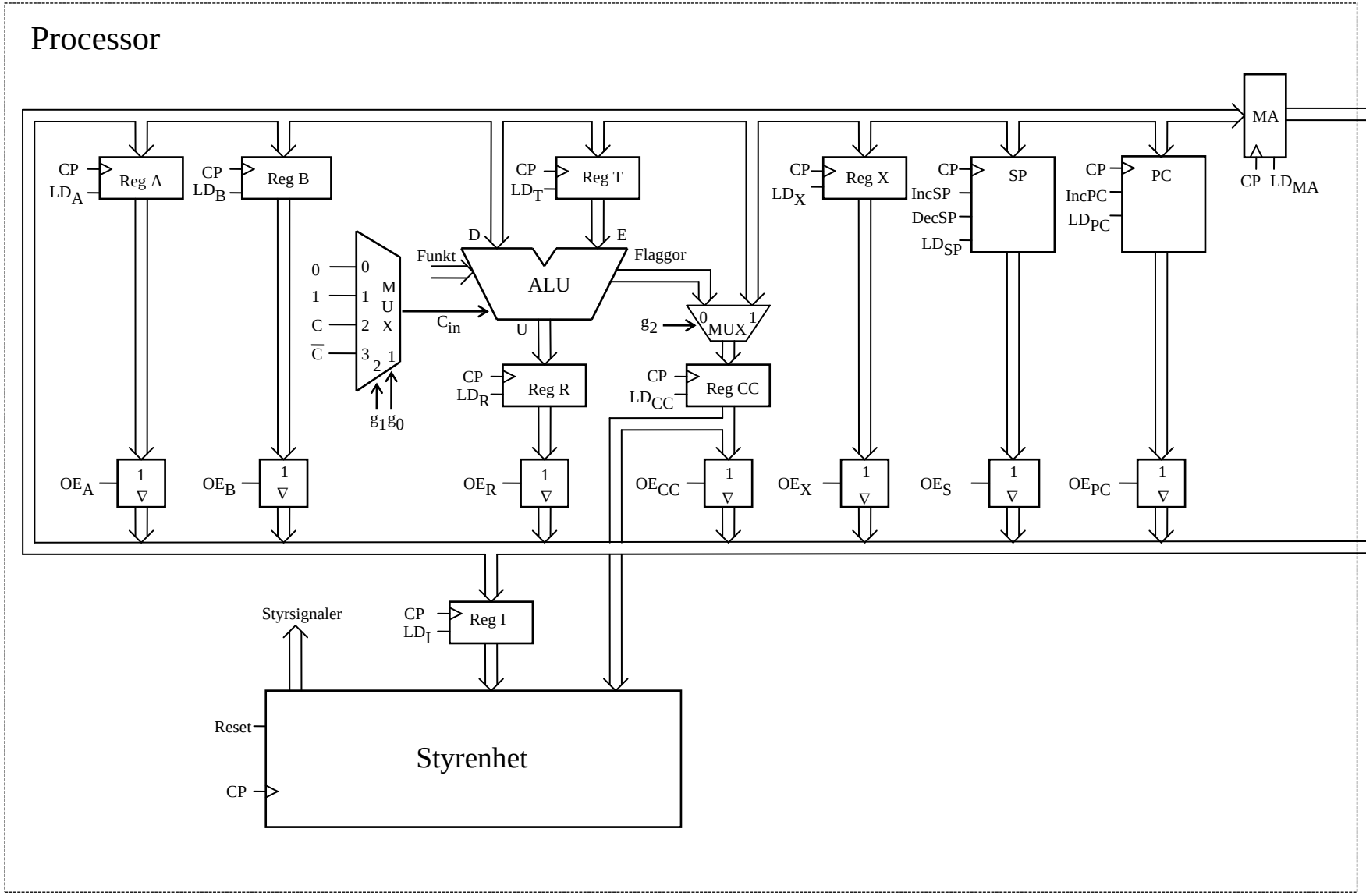
Tabell 1

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.