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	**2014-05-27 kl 14.00 – 18.00**
HJÄLPMEDEL	**Av institutionen utgiven instruktionlista ”FLEXIBLE INSTRUKTION SET PROCESSOR FLISP”** **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.	**Tentamen omfattar totalt 60 poäng.** **Onödigt komplicerade lösningar kan ge poängavdrag. Svar på uppgifter skall motiveras.** **<u>En svarsblankett finns sist i tentamenstesen.</u>** **För de uppgifter där svaret skall ges på svarsblanketten behöver inga lösningar redovisas.** **<u>Glöm inte lämna in svarsblanketten!</u>**
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 4-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.

$X = 0110_2$ och $Y = 1011_2$.

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 papper och penna hur räkneoperationen S = X + Y utförs i en 4-bitars ALU. **(1p)**

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

e) Visa med papper och penna hur räkneoperationen D = X – Y utförs i en 4-bitars ALU. **(1p)**

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

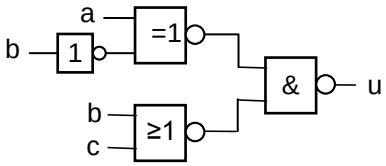
g) Tolka bitmönstren X, Y, S och D som tal *utan* tecken och ange deras decimala motsvarighet. Avgör och motivera med hjälp av flaggorna om resultaten S och D är korrekta eller felaktiga. **(1p)**

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) Flyttalsformatet "binary64" har 1 teckenbit, 11 bitar karakteristika och 52 bitar "fraction". Visa approximativt hur många decimala siffrors upplösning detta ger. **(2p)**

2.

a) Markera det korrekta minimala PS-uttrycket för u i grindnätet till höger på den bifogade svarsblanketten! **(2p)**



b) Ett karnaughdiagram för en boolesk funktion visas till höger. Markera ett minimalt uttryck på svarsblanketten som är lämpligt för realisering av den booleska funktionen f, då NOR-grindar med valfritt antal ingångar, XOR-grindar och NOT-grindar får användas.

Rutor med - representerar "dont care"-termer som får användas vid behov. Kretsrealiseringen behöver ej visas. **(4p)**

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

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 bestående av två nollor följda av en etta och en nolla.

Exempel: σ_x = ...0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 1 ...

σ_u = ...? 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 ...

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. (Sekvensnätet skall ej realiseras!)

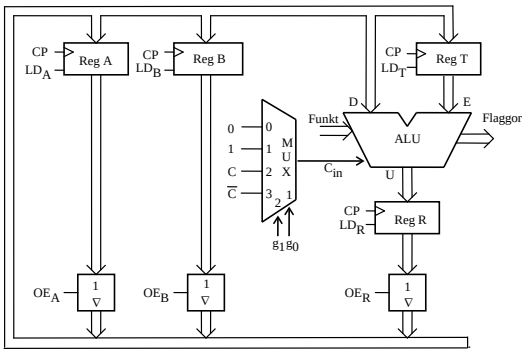
Hur många vippor skulle minst krävas vid en realisering? **(4p)**

b) Konstruera en reversibel 2-bitars synkron räknare med räknevillkoret x.

För x = 0 skall räknesekvensen $(q_1q_0)_2$ vara: 00, 01, 10, 11, 00, …
För x = 1 skall räknesekvensen $(q_1q_0)_2$ vara: 00, 11, 10, 01, 00, …

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

4. I datavägen till höger innehåller register A(8) och B(8) från början värdena 100_{10} resp. 160_{10} på binär form. Därefter ges styrsignaler och klockpulser enligt tabellen på svarsblanketten.



Komplettera tabellen på svarsblanketten med RTN-beskrivning samt hexadecimalt registerinnehåll för alla klockpulsintervall.

Vid laddning av ett register skall det nya innehållet ”synas” på raden efter laddningen i tabellen.

Flaggorna sparas inte mellan klockpulserna i kopplingen ovan. Ett tidigare C-värde kan alltså inte användas i ett senare klockpulsintervall!

ALU-funktioner: Se bilaga 1 på sidan 6 i tentamenstesen!

(4p)

5. Figuren på sidan 45 i instruktionslistan visar hur FLIS-datorn är uppbyggd. På sidorna 43 och 44 visas hur ALU’ns funktion väljs med styrsignalerna f_3 - f_0 och C_{in}.

I en tabell på svarsblanketten visas EXECUTE-sekvensens styrsignaler för en av FLIS-processorns instruktioner.

a) Komplettera tabellen med RTN-beskrivning och förklara vilken assemblerinstruktion som beskrivs! **(2p)**

b) Med instruktionen "**Bit SET**" nedan skall man kunna ettställa valfria bitar i ett minnesord. De bitpositioner som skall ettställas skall vara ettställda i instruktionens tredje ord, mask.

När instruktionen utförs skall det nya dataordet påverka N- och Z-flaggan.
V-flaggan skall nollställas och C-flaggan skall vara oförändrad.

Register A, X, Y eller SP får inte påverkas av instruktionen som skall implementeras för FLIS processorn med hjälp av styrenheten med fast logik.

BSET Adr,#mask RTN: M(Adr) *OR* mask → M(Adr), ALU(NZ) → CC, 0 → V, C → C

OPKOD
Adr
mask

Komplettera tabellen på svarsblanketten med RTN-beskrivning och styrsignaler för den efterfrågade EXECUTE-sekvensen. Använd operationskoden EF_{16}. **(5p)**

6. Besvara kortfattat följande frågor rörande FLIS-processorn.

a) Före det villkorliga hoppet "BPL Adr" i ett program utförs en addition "$7A_{16}$ + W", som påverkar flaggorna. För vilka värden på talet W utförs hoppet? (8-bitars tal används.) **(4p)**

b) Översätt programsekvensen till höger till maskinkod på hexadecimal form och visa hur den placeras i minnet. Det skall framgå hur offset för branch-instruktionerna beräknas.

```
        ORG     $30
START   LDY     #10
        LDX     #TAB
LOOP    LDA     ,X+
        STA     $FB
        LEAY    -1,Y
        CMPY    #0
        BPL     LOOP
        LDA     12
        BRA     NEXT
TAB     RMB     11
NEXT    NOP
```

(3p)

c) Hur lång tid tar programmet i b) att köra från och med LDY #10 till och med NOP om FLIS-processorn klockas med frekvensen 1MHz?

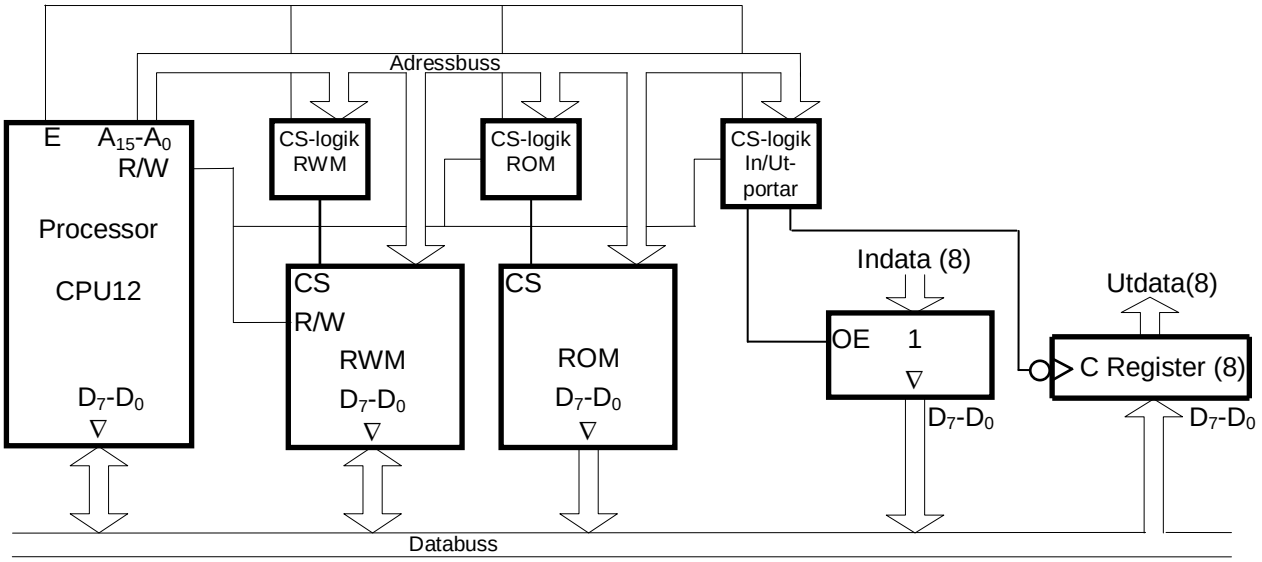
(3p)

7. I minnet i ett datorsystem med FLIS-processorn finns en sträng med sjubitars ASCII-tecken, där varje ASCII-tecken har kompletterats med en paritetsbit som åttonde bit (bit nr 7). Strängen avslutas med ett dataord med värdet noll.

Skriv en subrutin COUNT i assemblerpråk för FLIS-processorn som räknar det totala antalet ASCII-tecknen i strängen samt antalet ASCII-tecken som motsvarar decimala siffror (dvs 0-9). Slutmarkeringen räknas ej som ASCII-tecken. Det totala antalet ASCII-tecken skall finnas i A-registret och antalet "decimala siffertecken" i X-registret vid återhopp. Vid anrop av subrutinen antas startadressen till strängen finnas i Y-registret. En tabell över ASCII-koden finns i bilaga 2.

Endast registren A, X och CC får vara förändrade vid återhopp från subrutinen. För full poäng skall programmet vara "korrekt" radkommenterat. **(6p)**

8. Ett mikrodatorsystem skall konstrueras med CPU12, 1 st 32 kbyte ROM-kapsel, 1 st 32 kbyte RWM-kapsel, 2 st 8-bitars "three-state"-buffertar som inportar och 2 st 8-bitars register som utportar. Figuren nedan visar principen för anslutning av olika moduler till CPU12.



Adressområdet som består av de första **6k** adresserna skall reserveras för framtida bruk. Ingen modul får därför aktiveras inom detta adressområde.

ROM-modulens slutadress skall vara $FFFF_{16}$ och RWM-modulens slutadress skall vara $7FFF_{16}$.

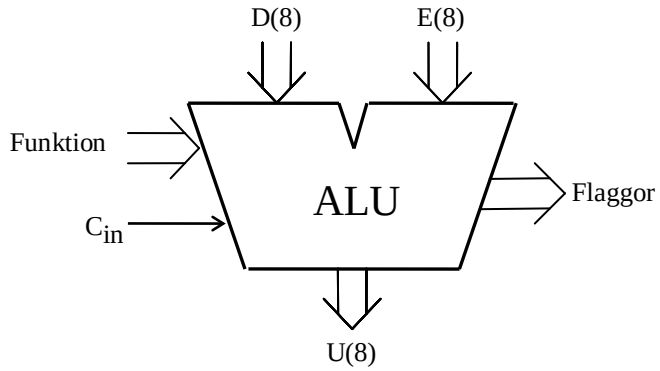
På de två adresserna direkt efter RWM-modulen skall de två inportarna och utportar placeras. Det skall alltså finnas både en inport och en utport på var och en av dessa adresser.

Uppgift:
Använd lämpliga signaler från processorn för att konstruera CS-logiken för ROM-modulen, RWM-modulen, inportarna och utportarna. Använd fullständig adressavkodning. Endast grundläggande logikgrindar med valfritt antal ingångar får användas. De användbara adressintervallen för de två minnesmodulerna skall anges på hexadecimal form.

(7p)

Bilaga 1

ALU för uppgift 4



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 FLIS-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

Svarsblankett (2014-05-27)

Anonym kod:

Uppgift 2:

a)

1. u = a' + ab + ac
2. u = ab'c'
3. u = (a + b)(a + c)
4. u = a' + b + c
5. u = a'bc
6. u = (a'bc)'
7. u = a + b' + c'
8. u = (ab'c')'

b)

1. f = d'(a⊕b⊕c)'(a' + d)
2. f = (a⊕b⊕c⊕d)(a' + d)
3. f = [d + (a⊕b⊕c)](a + d')
4. f = [d' + (a⊕b⊕c)'](a' + d)
5. f = [a + (b⊕c⊕d)](a' + d)
6. f = [d(a⊕b⊕c)] + ad'
7. f = [c + (a⊕b⊕c)](c' + d)
8. f = [a' + (b⊕c⊕d)](a' + d)

Uppgift 4: (Här används ALU'n i bilaga 1 på sidan 6!)

CP	Styrsignaler (=1)	RTN	A(8)	B(8)	T(8)	R(8)
1	OE_B, LD_T		64	A0	?	?
2	OE_A, f_3, f_1, LD_R, LD_T					
3	OE_R, f_3, f_1, f_0, g_0, LD_R					
4	OE_R, f_3, f_2, LD_R, LD_T					
5	OE_R, f_3, f_1, LD_R					
6	OE_R, LD_B					
7	?					

Uppgift 5:

a)

State	Summaterm	RTN-beskrivning	Styrsignaler
Q_4	$Q_4 \cdot I_{xx}$		MR, LD_T, INC_{PC}
Q_5	$Q_5 \cdot I_{xx}$		MR, g_{12}, LD_T
Q_6	$Q_6 \cdot I_{xx}$		OE_A, f_3, f_2, g_1, g_0, LD_R, LD_{CC}
Q_7	$Q_7 \cdot I_{xx}$		OE_R, LD_A, NF

Instruktionen är:

b) (Du kan skriva samma statenummer på flera rader vid behov!)

State	Summaterm	RTN-beskrivning	Styrsignaler