

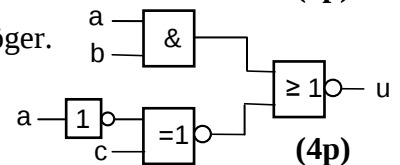


TENTAMEN *(Nu anpassad till FLIS-processorn)*

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- eller FLIS-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. Den som är inskriven före HT 2012 kan använda FLEX-processorn istället för FLIS-processorn! 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 $Y = 01101$.

- Vilket talområde måste X, Y, S och D tillhöra om de tolkas som tal utan tecken? (1p)
- Vilket talområde måste X, Y, S och D tillhöra om de tolkas som tal med tecken? (1p)
- Visa med penna och papper hur räkneoperationen $S = X + Y$ utförs i en 5-bitars ALU. (1p)
- Vilka värden får flaggbitarna N, Z, V och C vid räkneoperationen i c)? (1p)
- Visa med penna och papper hur räkneoperationen $D = X - Y$ utförs i en 5-bitars ALU. (1p)
- Vilka värden får flaggbitarna N, Z, V och C vid räkneoperationen i e)? (1p)
- 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)
- 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)
- 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)
- 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)
- 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.

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

(4p)

3.

- 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: $\sigma_x = \dots 010011001100100110100010010011\dots$

$\sigma_u = \dots 000001000000000010000000000001\dots$

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)

- 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.

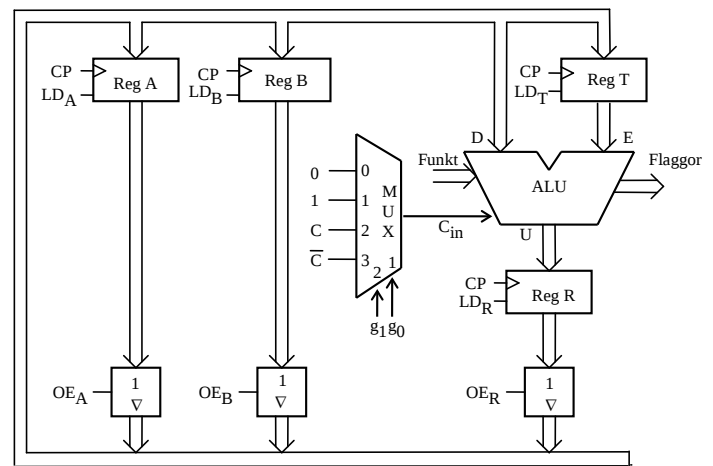
(6p)

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. Figuren på sidan 45 i INS1 visar hur FLISP-datorn är uppbyggd. På sidan 44 i INS1 visas 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 FLIS-processorns instruktioner.

State	S-term	RTN-beskrivning	Styrsignaler (=1)
Q ₄	Q ₄ ·I _{xx}		OE _{PC} , LD _T , LD _{TA}
Q ₅	Q ₅ ·I _{xx}		MR, f ₃ , f ₁ , f ₀ , g ₀ , g ₁₄ , LD _R , INC _{PC}
Q ₆	Q ₆ ·I _{xx} ·((N⊕V)+Z)'		OE _R , LD _{PC}
	Q ₆ ·I _{xx}		NF

- a) Rita en tabell med "State"- och RTN-kolumner enligt ovan eller nedan och fyll i RTN-beskrivningen. Förklara vilken assemblerinstruktion som beskrivs.

Som alternativ kan FLEX-datorn som visas i bilaga 3, ALU'n i bilaga 1 och tabellen nedan användas istället för FLISP-datorn.

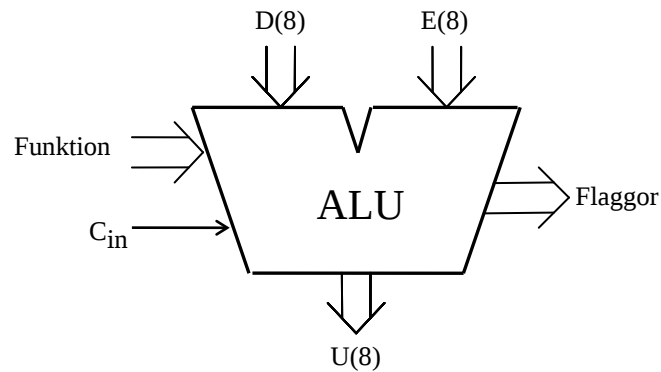
State	S-term	RTN-beskrivning	Styrsignaler (=1)
Q ₅	Q ₅ ·I _{xx}		OE _{PC} , LD _{MA} , LD _T
Q ₆	Q ₆ ·I _{xx}		MR, f ₃ , f ₁ , g ₀ , LD _R , IncPC
Q ₇	Q ₇ ·I _{xx} ·((N⊕V)+Z)'		OE _R , LD _{PC}
	Q ₇ ·I _{xx}		NF

(2p)

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 **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, såsom pseudoinstruktioner eller assemblatordirektiv. Pseudoinstruktionerna listas i tabell 1.

Tabell 1

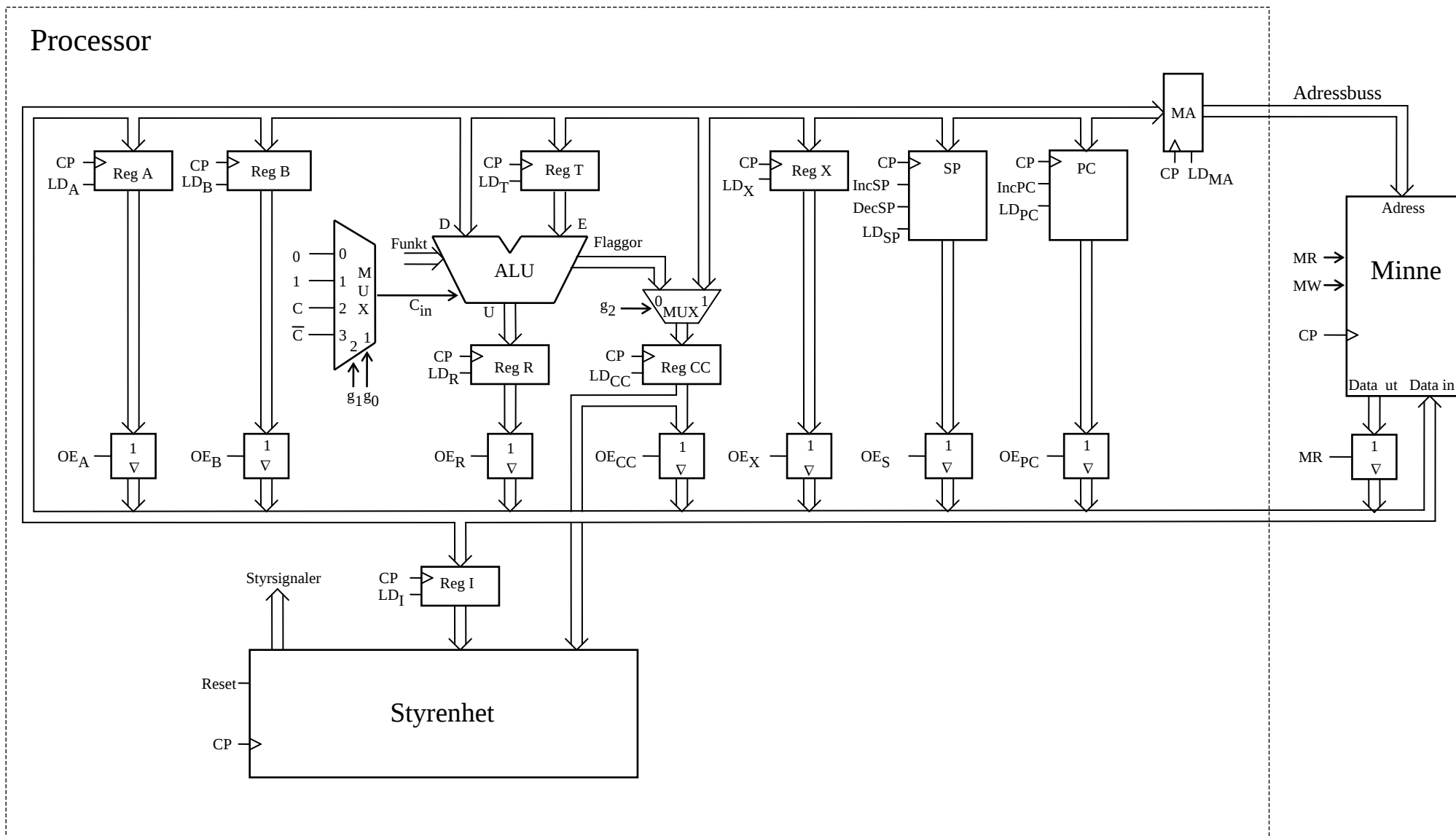
Direktiv	Förklaring
ORG N	Placerar den efterföljande koden med början på adress N. (ORG för ORiGin = ursprung)
L RMB N	Avsätter N bytes i följd i minnet (utan att ge dem värden), så att programmet kan använda dem. Följden placeras med början på adressen L. (RMB för ReseVe Memory Bytes)
L EQU N	Ger symbolen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

7(7)



Figur 1. Datorn FLEX.