

Lösningförslag tenta 2012-03-05 (v5 med reservation för eventuella fel!)

1. a) LDY \$4000 → FD 40 00 (1p)
- b) 18 03 24 18 FF F2 → MOVW #\$2418,\$FFF2 (2p)
- c) ANDCC #\$EF ↔ CLI (2p)
- d) STD 4,X → 6C 04 (2p)
- e) 1600_{16} : DBEQ X,\$1580 → 04 15 rr (lb = 15 eftersom hoppet görs bakåt i minnet)
Nästa OP-kod finns på adr 1603_{16} . Offset(rr) = Till – Från = $1580_{16} - 1603_{16} = \text{FF}7\text{D}_{16}$
DBEQ X,\$1580 → 04 15 7D
X-värdet minskas från 5 till 4 ($\neq 0$) varför hoppet inte utförs. Nästa instruktion utförs därför på adressen 1603_{16} . (2p)
- f) När processorn tar emot ett avbrott lägger den alla registerinnehåll på stacken och hoppar sedan till den adress som finns i avbrottsvektorn. I avbrottsrutinen fortsätter den sedan med FETCH på normalt sätt, men då måste avbrottsystemet vara avstängt ($I = 1$) för annars skulle den direkt göra ett nytt avbrott om IRQ-ingången fortfarande är aktiv. (1p)
- g) BGT (>) avser tal med tecken. Det innebär att vi skall tolka data som tal i intervallet $[-128, 127]$.
Talet $87_{16} = 8 \cdot 16 + 7 = 135_{10}$ tolkas som det negativa talet $-(256 - 135) = -121$.
Eftersom flaggvillkoret tar hänsyn till overflow behöver man inte testa det fallet.
CMPA utför subtraktionen: $-121 - W$ och hoppvillkoret blir: $-121 - W > 0$ eller $W < -121$.
När hänsyn tas till talområdet blir hoppvillkoret: $-128 \leq W < -121$.
Eftersom vi använder 2k-representation blir det verkliga intervallet: $256 - 128 \leq W < 256 - 121$
Hoppvillkoret blir då: $128 \leq W < 135$ (2p)
- h) BHI (>) avser tal utan tecken. Det innebär att vi skall tolka data som tal i intervallet $[0, 255]$.
Talet $AA_{16} = 10 \cdot 16 + 10 = 170_{10}$.
ADDB bildar talet $170 + W$. Hoppvillkoret (HI) är att både C och Z har värdet 0.
Det innebär att $170 + W < 256$, dvs. $W < 86$.
När hänsyn tas till talområdet blir hoppvillkoret: $0 \leq W < 86$. (2p)
- i) Att logikvärdet "0" dominerar på bussen betyder att om en nod sänder "0" blir bussvärdet "0" oavsett vad andra noder sänder. Detta är nödvändigt för att arbitreringsen (bussåtkomsten) skall fungera. Samma sak gäller när en nod skall signalera att ett meddelande har mottagits korrekt. (2p)
- j) Format: s/c/f $N = -130,125 = -10000010.001_2 = -1.0000010001_2 \cdot 2^7$
 $s = 1 (-)$; $c = \text{exp} + 127 = 7 + 127 = 6 + 128 = 1000\ 0110_2$; $f = 000001000100\dots 0$
 $N_{\text{flyt}} = 1/100\ 0011\ 0/000\ 0010\ 0010\ 0000\ 0000\ 0000_2 = \text{C}3022000_{16}$ (2p)
- k) "Flash"-minnen klarar bara ett begränsat antal skrivningar på samma adress förutom att de raderas blockvis och att skrivningar tar förhållandevis lång tid. (2p)

2. a) Adressen DTAB anses känd och kan därför användas i subrutinen.

TEST	PSHX		Spara på stack
	PSHY		
	LDX	#DTAB	Pekare till tabell
	LDY	#0	Räknare för träff
LOOP	LDAA	1,X+	
	CMPA	#\$FF	Slut?
	BEQ	TSTEX	Ja
	ANDA	##01101110	Nej, maska bort "don't care"-bitar (7,4,0)
	CMPA	##00100110	
	BNE	LOOP	Ej träff, fortsätt med nästa
	INY		Träff, öka träffräknare
	BRA	LOOP	Fortsätt med nästa.
TSTEX	TFR	Y,D	Returnera räknare
	PULY		Återställ register
	PULX		
	RTS		

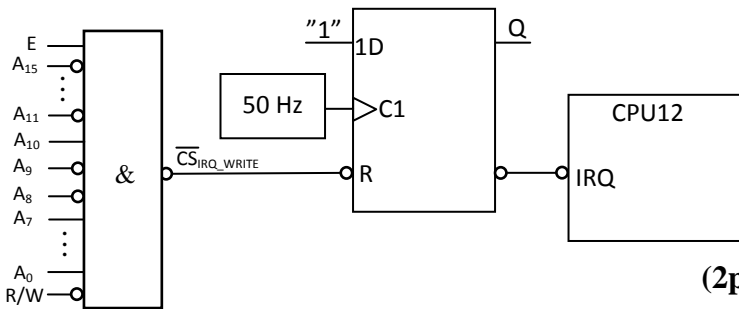
(5p)

b)

TURN	PSHB		Spara på stack
	MOVB	#8,BCNT	Räknare för antal skift
	CLRB		Nollställ plats för inskiftade bitar
LOOP	ASLA		Bit 7 till carry
	RORB		Carry till bit 7 i reg B
	DEC	BCNT	Minska skifträknare
	BNE	LOOP	8 skift? Nej
*			Ja, nu finns bitarna i omvänd ordning i reg B
	TFR	B,A	Returnera i reg A
	PULB		Återställ från stack
	RTS		
BCNT	RMB	1	

(7p)

4. a) Adress för nollställning av avbrottsvippa:
 $04FF_{16} = 0000\ 0100\ 1111\ 1111_2$



(2p)

b) Avbrottsrutin

* Avbrottsrutin

IRQR	CLR	IR_FF	Nollställ avbrottsvippa
	LDX	CNT1MIN	Minska minuträknare
	DEX	CNT1MIN	
	STX	CNT1MIN	
	BNE	EX_IR	Ej 1 min
	MOVB	INPORT,MADR	
	MOVW	#3000,CNT1MIN	Ominitera räknare för 1 min
EX_IR	RTI		

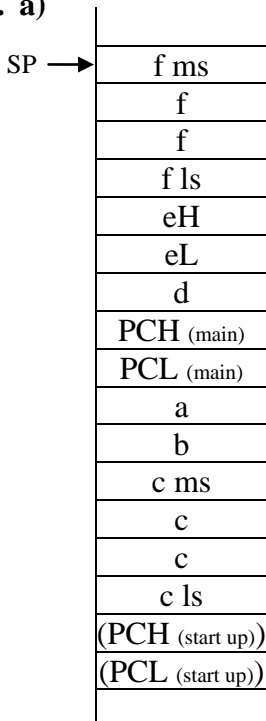
(4p)

c) Avsnitt av huvudprogram

CNT1MIN	EQU	\$2FF0	
MADR	EQU	\$CF00	
IR_FF	EQU	\$4FF	
INPORT	EQU	\$600	
(START	LDS	#BOS	Sätt SP)
	MOVW	#IRQR,\$FFF2	Sätt avbrottsvektorn
	CLR	IR_FF	Nollställ avbrottsvippa
	MOVW	#3000,CNT1MIN	Räknare för 1 min
	CLI		Aktivera avbrottsystem

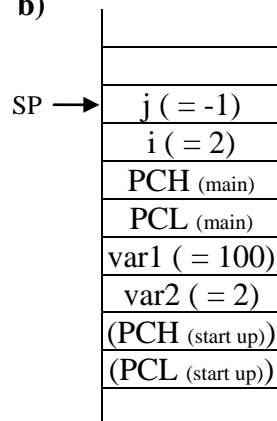
(2p)

5. a)



(3p)

b)



c) $i = 2, 77(\text{if}), 152(\text{if}), 172(\text{else}), 192(\text{else})$.
 $j = -1, 1(\text{else}), 3(\text{else})$.

Returvärde: $i = 192 = (-64 \text{ för char})$

(3p)

main	LDAB	#2	var1
	PSHB		
	LDAB	#100	var2
	PSHB		
	JSR	funcb	
	LEAS	2,SP	Justera stack
	RTS		
funcb	LEAS	-2,SP	Prolog
	LDAB	#2	
	STAB	1,SP	$i = 2$
	LDAB	#-1	
	STAB	0,SP	$j = -1$
w_loop	LDAB	0,SP	j
	CMPB	5,SP	$j - \text{var2}$
	BGT	w_exit	Hopp om > 0 (med tecken)
if_tst	LDAB	1,SP	i
	CMPB	4,SP	$i - \text{var1}$
	BHS	else	Hopp om ≥ 0 (= BCC)
	ADDB	#75	$i + 75$
	STAB	1,SP	$i = i + 75$
	BRA	w_loop	
else	LDAB	0,SP	j
	ADDB	#2	$j + 2$
	STAB	0,SP	$j = j + 2$
	LDAB	1,SP	i
	ADDB	#20	$i + 20$
	STAB	1,SP	$i = i + 20$
	BRA	w_loop	
w_exit	LDAB	1,SP	Returparameter i
	LEAS	2,SP	Epilog
	RTS		

(6p)