CHALMERS

Institutionen för data- och informationsteknik
Avdelningen för datorteknik

TENTAMEN

KURSNAMN	**Maskinorienterad programmering**
PROGRAM:	**Dataingenjör och elektroingenjör åk 1/ lp 3** **Mekatronikingenjör åk 2/ lp 3**
KURSBETECKNING	**LEU500**
EXAMINATOR	**Lars-Eric Arebrink**
TID FÖR TENTAMEN	**Måndag 2012-08-27 kl 14.00 – 18.00**
HJÄLPMEDEL	**Av institutionen utgiven** **”Instruktionslista för CPU12” (INS2)** **Tabellverk eller miniräknare får ej användas.**
ANSV LÄRARE: **besöker tentamen**	**Lars-Eric Arebrink, tel. 772 5718** **772 5718** **vid flera tillfällen**
DATUM FÖR ANSLAG av resultat samt av tid och plats för granskning	**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. **BETYGSGRÄNSER** **SLUTBETYG**	**Tentamen omfattar totalt 60 poäng.** **Onödigt komplicerade lösningar kan ge poängavdrag. Svar på uppgifter skall motiveras.** **24p ≤ betyg 3 < 36 p ≤ betyg 4 < 48 p ≤ betyg 5** **För godkänt slutbetyg 3, 4 eller 5 på kursen fordras betyg 3, 4 eller 5 på tentamen samt godkända laborationer.**

1. Besvara kortfattat följande frågor, som alla utom j) - m) avser CPU12.

a) Översätt assemblerinstruktionen EXG SP,Y till maskinspråk och visa maskinkoden i minnet. **(1p)**

b) Vilken assemblerinstruktion har den hexadecimala maskinkoden 18 07? **(1p)**

c) En instruktion med den hexadecimala maskinkoden 0F F2 42 3A 07 60 är placerad med operationskoden på adressen 1800_{16}. Skriv instruktionen med assemblerspråk. **(2p)**

d) Assemblerinstruktionen LSLA kan skrivas som en alternativ assemblerinstruktion. Vilken? **(1p)**

e) Översätt assemblerinstruktionen COM -$90,X till maskinspråk. Visa maskinkoden i minnet. **(2p)**

f) Assemblerinstruktionen LBNE $8000 har operationskoden på adressen 1500_{16}.
Visa maskinkoden i minnet. **(2p)**

För vilka värden W ($0 \leq W \leq 255$) utförs hoppet i g) och h)?

g)
```
LDAA  #$50
SUBA  #W
BHI   Hopp
```
(2p)

h)
```
LDAB  #W
ADDB  #$50    (Tänk på att overflow kan inträffa!)
BPL   Hopp
```
(3p)

i) Processorn CPU12 använder en 16-bitars adressbuss, men kan ändå använda mera arbetsminne än 64 kbyte. Vad kallas denna princip? Förklara kortfattat hur detta går till. **(3p)**

j) Vilken är den principella skillnaden mellan synkron och asynkron seriell överföring av data? **(1p)**

k) Förklara vad som menas med ”frame” i samband med asynkron seriell överföring av data. **(2p)**

l) I ett styrsystem är ett antal CAN-noder sammankopplade med en buss där logikvärdet ”0” dominerar. Vid ett tillfälle blir fyra av noderna samtidigt färdiga att sända var sitt meddelande med identifierarna $635A_{16}$, $636B_{16}$, 6343_{16} och $63FF_{16}$. Visa i vilken ordning meddelandena sänds. Visa också hur det går till när det avgörs i vilken ordning meddelandena sänds. **(3p)**

m) Visa approximativt det decimala värdet för det största och det minsta (ej talet 0) positiva flyttalet med full upplösning enligt IEEE-standard 754-1985 (23 bitar av mantissan och 8 bitars karakteristika). **(3p)**

2.

a) Skriv en subrutin EPTST i assemblerspråk för CPU12, som söker igenom en nollterminerad textsträng med ASCII-tecken i minnet och kontrollerar att paritetsbiten (b_7) för jämn paritet är korrekt. Jämn paritet innebär att kodordet inklusive paritetsbiten har ett jämnt antal ettor.

Vid anrop av subrutinen skall startadressen till textsträngen finnas i X-registret. När ett paritetsfel upptäcks skall återhopp göras direkt med C-flaggan ettställd och adressen till det felaktiga tecknet i X-registret. Om textsträngen är felfri skall återhopp göras med C-flaggan nollställd. Vid återhopp får endast flaggregistret och X-registret vara förändrat.

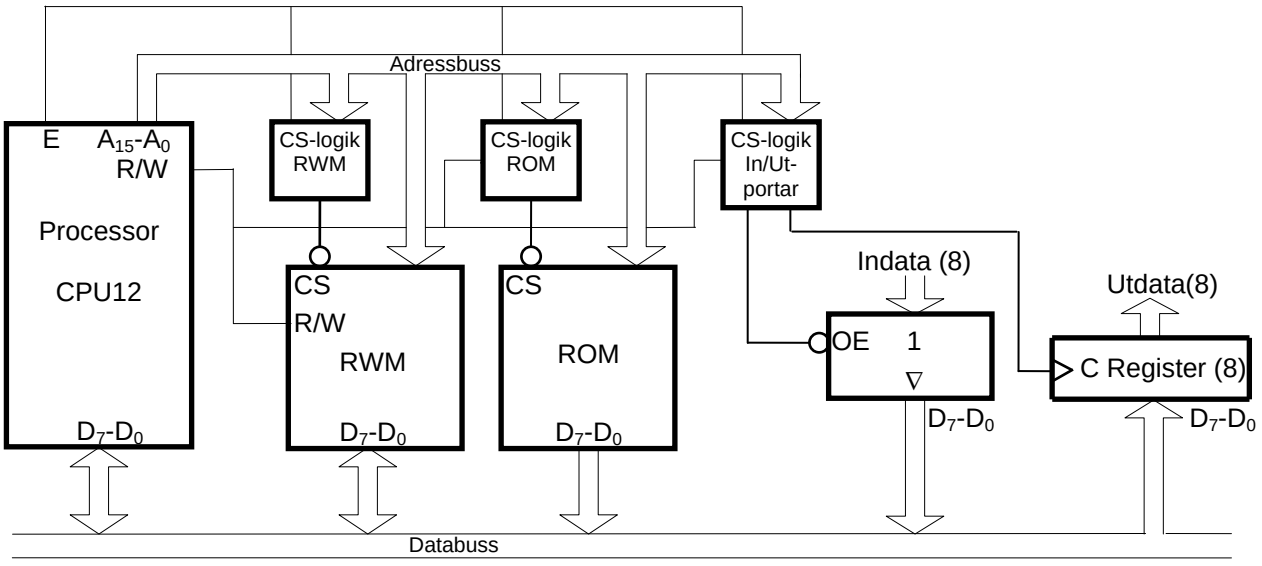
Du får använda en färdig subrutin, ACNT, som räknar antalet ettor i A-registret och returnerar detta antal i A-registret. Subrutinen ACNT påverkar endast flaggorna och A-registret.

(7p)

b) Skriv ett huvudprogram som sätter stackpekaren till värdet 2000_{16}, anropar subrutinen EPTST och sedan hoppar till adressen 1400_{16}. Textsträngen, med ett innehåll som du själv bestämmer, skall placeras direkt efter huvudprogrammet. Den skall placeras i minnet när huvudprogrammet laddas i minnet. Huvudprogrammets startadress skall vara 1000_{16}. **(3p)**

Assemblerspråk för CPU12 skall användas. För full poäng skall programmen vara ”korrekt” radkommenterade.

3. Ett datorsystem visas nedan:



Figuren ovan visar principen för anslutning av minnesmoduler och in-/utportar till processorn CPU12. Hela adressrummet skall i princip fyllas ut med en 64 kbyte ROM-modul, men på vissa adresser skall in- och utportar och en 8 kbyte RWM-modul vara placerade. Dessutom skall de första 1000_{16} adresserna inte aktivera någon port eller minnesmodul vid läsning eller skrivning. RWM-modulen skall ha startadressen 8000_{16}. Två inportar och två utportar skall placeras på adresserna 4000_{16} och 4001_{16}.

Rita CS-logiken för minnesmodulerna och portarna. Ange adressintervallen för minnesmodulerna. Använd fullständig adressavkodning. Endast grundläggande logikgrindar med valfritt antal ingångar får användas. **(6p)**

4. Avbrottssystemet i CPU12 skall användas till att köra två olika program, Prog_0 och Prog_1, pseudoparallellt. Programmen som utgörs av evighetsslingor har startadresserna 1000_{16} resp. 8000_{16}. De använder var sin stack med BOS på adresserna 7000_{16} resp. $F000_{16}$. En händelsedetektor som genererar en binärpuls vid varje händelse skall också ge avbrott. Vid varje sådant avbrott skall en 16-bitars variabel EVENT i minnet ökas med ett.

Man har tillgång till en digital binär signal med frekvensen 200 Hz. En oanvänd inport som får användas finns på adressen INPORT. CS-signalen för inporten är inte tillgänglig.

a) Rita ett kopplingsschema som visar hur man dels kan generera avbrott med frekvensen 200 Hz och dels då pulser kommer från händelsedetektorn. Det får förutsättas att adressområdet $7FF0_{16}$ – $7FFF_{16}$ är tomt. Det finns inga övriga avbrottskällor att ta hänsyn till i systemet! Processorns IRQ'-ingång skall användas. D-vippor, AND-, NAND och NOT-grindar får användas. **(2p)**

b) Prog_0 inleds med en del som bara körs vid starten och fortsätter sedan med en evighetsslinga. I den första delen av Prog_0 görs nödvändiga initieringar av avbrottssystemet för de två avbrotts-källorna och för processbytet vid varje 200Hz-avbrott. Det förutsätts att systemet startar med reset av processorn och att Prog_0 då startas. Skriv programmet Prog_0. Utrymme för globala variabler finns på adresserna $7E00_{16}$- $7E0F_{16}$. IRQ-vektorn finns i ett flyktigt minne på adresserna $FFF2_{16}$ och $FFF3_{16}$. **(4p)**

c) Skriv avbrottsrutinen som uppdaterar variabeln EVENT och utför processbyte. **(4p)**

Adressen INPORT ovan är definierad på annat ställe i programmet. Assemblerspråk för processorn CPU12 skall användas. Radkommentarer skall finnas!

5. Du använder korskompilatorn XCC för CPU12, som har följande konventioner för C-funktioner:

- Inparameterlistan behandlas från höger till vänster och samtliga inparametrar överförs via processorns stack.
- Lokala variabler som deklareras placeras på stacken i den ordning de deklareras, dvs sist behandlad finns överst i stacken. Övriga lokala variabler placeras också på stacken i den ordning behovet av dem uppstår, dvs den sista finns överst på stacken.
- Varje funktion som har lokala variabler inleds med prologen `LEAS -?,SP` och avslutas med epilogen `LEAS ?,SP` följt av `RTS`.
- Returparameter (16- eller 8-bitars) lämnas i D- eller B-registret beroende på storlek.
- För XCC gäller dessutom: char 8 bitar, short och int 16 bitar, long 32 bitar.

a) Översätt hela C-programmet till höger till assemblerspråk för CPU12. Visa även hur stacken ser ut när for-satsen börjar utföras. **(6p)**

b) Vilket värde returneras från funktionen func? Motivera svaret! **(2p)**

```
char func(char xx, char yy);

char z = 0x60;

void main(){
 func(10,20);
}

 char func(char x, char y){
   char i;

   for ( i = 0; i <= 15; i = i + 3)

     if(i < x)
        y = z + y;

 return y;
 }
```

Assemblerspråket för CPU12 .

Assemblerspråket använder sig av mnemoniska beteckningar som processorkonstruktören MOTOROLA specificerat för maskininstruktioner 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 address 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 FDB N1, N2	Avsätter ett bytepar (två bytes) för varje argument i följd i minnet med mest signifikant byte på den lägsta adressen. Respektive bytepar ges konstantvärdet N1, N2 etc. Följden placeras med början på adressen L. (FDB för Form Double 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)

ASCII-koden

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