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 2012-03-05 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** **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 samt plats 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.** **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 i) - k) avser CPU12.

a) Översätt assemblerinstruktionen LDY $4000 till maskinspråk och visa hur maskinkoden placeras i minnet. **(1p)**

b) En instruktion har den hexadecimala maskinkoden 18 03 24 18 FF F2. Skriv instruktionen med assemblerspråk. **(2p)**

c) Assemblerinstruktionen ANDCC #$EF kan skrivas som en alternativ assemblerinstruktion. Vilken? **(2p)**

d) Översätt assemblerinstruktionen STD 4,X till maskinspråk. Visa hur maskinkoden placeras i minnet. **(2p)**

e) Assemblerinstruktionen DBEQ X,$1580 har operationskoden på adressen 1600_{16}. Visa hur maskinkoden placeras i minnet. På vilken adress utförs nästa instruktion om X-registret från början innehåller värdet 5? **(2p)**

f) Vid IRQ-avbrott sätts I-flaggan automatiskt till 1. Varför sker detta? **(1p)**

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

g)
```
LDAA  #$87
CMPA  Wadr
BGT   Hopp
```
(2p)

h)
```
LDAB  #$AA
ADDB  Wadr
BHI   Hopp
```
(2p)

i) När CAN-noder kopplas in till en buss så ser man till att logikvärdet ’0’ dominerar över logikvärdet ’1’. Förklara vad detta innebär och varför det är nödvändigt. **(2p)**

j) Skriv talet -130,125 som ett packat flyttal på binär och hexadecimal form enligt IEEE-standard 754 (23 bitar av mantissan och 8 bitars karakteristika). **(2p)**

k) Varför är inte ”flash”-minnen lämpliga att använda som RWM i en dator? **(2p)**

2.

a) I minnet i ett datorsystem med processorn CPU12 finns ett antal 8-bitars dataord lagrade i en tabell på adressen DTAB och framåt (ökande adress). Tabellen avslutas med talet FF_{16}.

Skriv en subrutin TEST i assemblerpråk för CPU12-processorn som tar reda på hur många av dataorden som har värdet ”1” i bitpositionerna 5, 2 och 1 samt värdet ”0” i bitpositionerna 6 och 3. Antalet dataord i tabellen som uppfyller detta skall finnas i D-registret vid återhopp. Endast register D och register CC får vara förändrade vid återhopp från subrutinen. **(5p)**

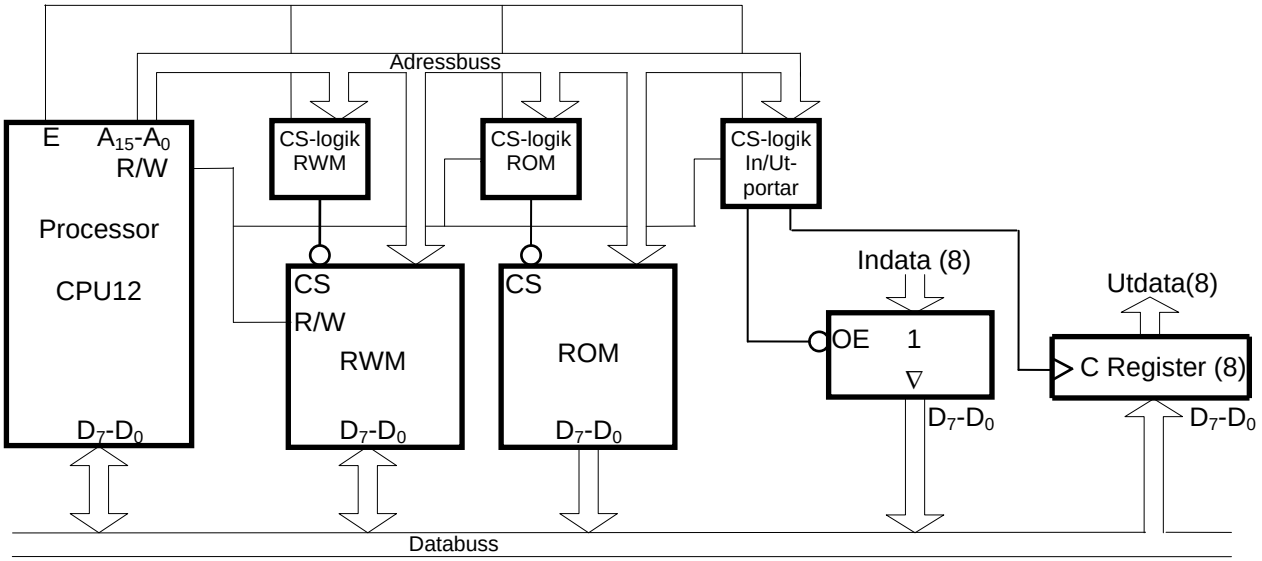
b) Skriv en subrutin TURN som vänder på bitarnas ordningsföljd i register A så att bit 7 blir bit 0, bit 6 blir bit 1, bit 5 blir bit 2 osv. enligt nedan:

$$b_7b_6b_5b_4b_3b_2b_1b_0 \rightarrow b_0b_1b_2b_3b_4b_5b_6b_7.$$

Vid anrop av subrutinen skall dataordet som skall ”vändas” finnas i register A och vid återhopp skall det ”vända” dataordet finnas i samma register. Endast A-registret och flaggregistret får vara förändrade vid återhopp. För full poäng skall programmet vara ”korrekt” radkommenterat.

(7p)

3. Ett datorsystem visas nedan:



Figuren ovan visar principen för anslutning av externa minnesmoduler och externa in-/utportar till processorn CPU12.

I detta fall skall halva adressrummet med start på adress 0 i princip fyllas ut med en 32 kbyte RWM-modul och andra halvan med en 32kbyte ROM-modul, men på vissa adresser skall in- och utportar och en 4 kbyte RWM-modul vara placerade. Dessutom skall de första 100_{16} adresserna inte aktivera någon port eller minnesmodul vid läsning eller skrivning. 4kbyte RWM-modulen skall vara placerad inom ROM-modulens adressområde med start på adress $C000_{16}$. En inport och en utport skall placeras på adressen 100_{16}.

Rita CS-logiken för minnesmodulerna och portarna. Ange adressintervallen för minnesmodulerna. Använd fullständig adressavkodning. Endast grundläggande logikgrindar med valfritt antal ingångar får användas. **(8p)**

4. En process skall styras med hjälp av en dator med mikroprocessorn CPU12. I samband med detta skall en flödesgivare som är ansluten till en av datorns inportar på adressen 600_{16} läsas av en gång per minut. Det avlästa värdet är ett 8-bitars tal utan inbyggt tecken och är ett mått på flödet i en rörledning.

Eftersom datorn normalt är upptagen med beräkningsarbete för processtyrningen skall avbrott användas för avläsningen av flödesgivaren. CS-signalen till inporten finns ej tillgänglig. Adressintervallet 400_{16}-$4FF_{16}$ är helt tomt, dvs. inga moduler är aktiva i detta intervall. Adressintervallet $2FF0_{16}$-$2FFF_{16}$ i RWM är ledigt och kan användas för globala variabler.

Det finns en binär signal med den konstanta frekvensen 50 Hz tillgänglig för generering av avbrott.

a) Föreslå en koppling för avbrottsgenerering på processorns IRQ'-ingång. En D-vippa med positiv flanktriggning och asynkron resetingång samt standardgrindar med valfritt antal ingångar får användas. Det finns inga andra avbrottskällor i systemet. **(2p)**

b) Skriv avbrottsrutinen IRQR som läser av flödesgivaren en gång per minut och placerar det avlästa värdet på adressen $CF00_{16}$ i minnet. **(4p)**

c) Skriv ett avsnitt av huvudprogrammet där IRQ-avbrott initieras. IRQ-vektorn antas vara placerad i RWM på adresserna $FFF2_{16}$ och $FFF3_{16}$. **(2p)**

Assemblerspråk för processorn CPU12 skall användas. Radkommentarer skall finnas!

5.

a) Du använder en korskompilator för HCS12 med följande konventioner för C-funktioner:

- Inparameterlistan behandlas från höger till vänster, 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 lämnas i D- eller B-registret beroende på storlek.
- För XCC gäller dessutom: char 8 bitar, short och int 16 bitar, long 32 bitar.

Antag att funktionen funca anropas från main och definieras på följande sätt samt att inga övriga lokala variabler behövs:

```
int funca( unsigned char a, char b, unsigned long c )
{
  char d;
  int e;
  unsigned long f;
    . . . .
 }
```

Visa stackens innehåll direkt efter det att funktionens prolog har körts. Platsen för samtliga variabler skall visas. **(3p)**

5. (forts.)

b) Översätt C-programmet nedan (main och funcb) till assemblerspråk för CPU12. Visa även stackens innehåll innan while-satsen börjar utföras. (Antag att konventionerna i a-uppgiften gäller.)

```
char funcb(unsigned char var1,char var2);

void main(){
 funcb(100,2);
}

char funcb(unsigned char var1, char var2){
  unsigned char i = 2;
  char j = -1;

  while( j <= var2)
    if(i < var1)
      i = i + 75;
    else{
      j = j + 2;
      i = i + 20;
    }
return i;
}
```

(6p)

c) Vilket värde returneras från funktionen funcb? Motivera svaret! **(3p)**

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