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 2011-08-22 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.** **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) -l) avser CPU12.

a) Översätt assemblerinstruktionen TFR D,X till maskinspråk och visa hur maskinkoden placeras i minnet. **(1p)**

b) Vilken assemblerinstruktion har den hexadecimala maskinkoden E4 35? **(1p)**

c) En instruktion med den hexadecimala maskinkoden 0D E2 12 80 55 är placerad med operationskoden på adressen 2000_{16}. Skriv instruktionen med assemblerspråk. **(2p)**

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

e) Översätt assemblerinstruktionen ADDB %100,Y till maskinspråk. Visa hur maskinkoden placeras i minnet. **(2p)**

f) Assemblerinstruktionen DBNE A,$1136 har operationskoden på adressen 1200_{16}. Visa hur maskinkoden placeras i minnet. **(2p)**

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

g)
```
LDAA  #$3F
BITA  #W
BNE   Hopp
```
(2p)

h)
```
LDAB  #W
COMB
ADDB  #$3F      (Tänk på att overflow kan inträffa vid additionen!)
BMI   Hopp
```
(4p)

i) Vid synkron seriell överföring av data är det viktigt att klocksignal och data anländer till mottagaren ”tillräckligt” samtidigt. Hur löser man detta? **(1p)**

j) I CAN-protokollet ingår sk ”stuffbitar”. Förklara varför dessa behövs. **(2p)**

k) 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)**

l) För att minska accesstiden för ett stort långsamt minne kan man koppla in ett cacheminne före det stora minnet. Vad är det som krävs för att detta skall löna sig? **(2p)**

2.

a) STRING1 och STRING2 är två strängar med 8-bitars NBCD-tal, som alltså tillhör intervallet $[0,99_{10}]$. Strängarna avslutas med datavärdet FF_{16} efter det sista NBCD-värdet.

Skriv en subrutin ADD10, som adderar alla NBCD-värden från STRING1 med motsvarande NBCD-värden i STRING2 och placerar NBCD-summan i STRING2 enligt principen:

M(X) + M(Y) → M(Y)
M(X+1) + M(Y+1) → M(Y+1), osv.

Alla additioner som resulterar i NBCD-tal > 99_{10} skall räknas och antalet sådana additioner skall returneras i D-registret vid återhoppet.

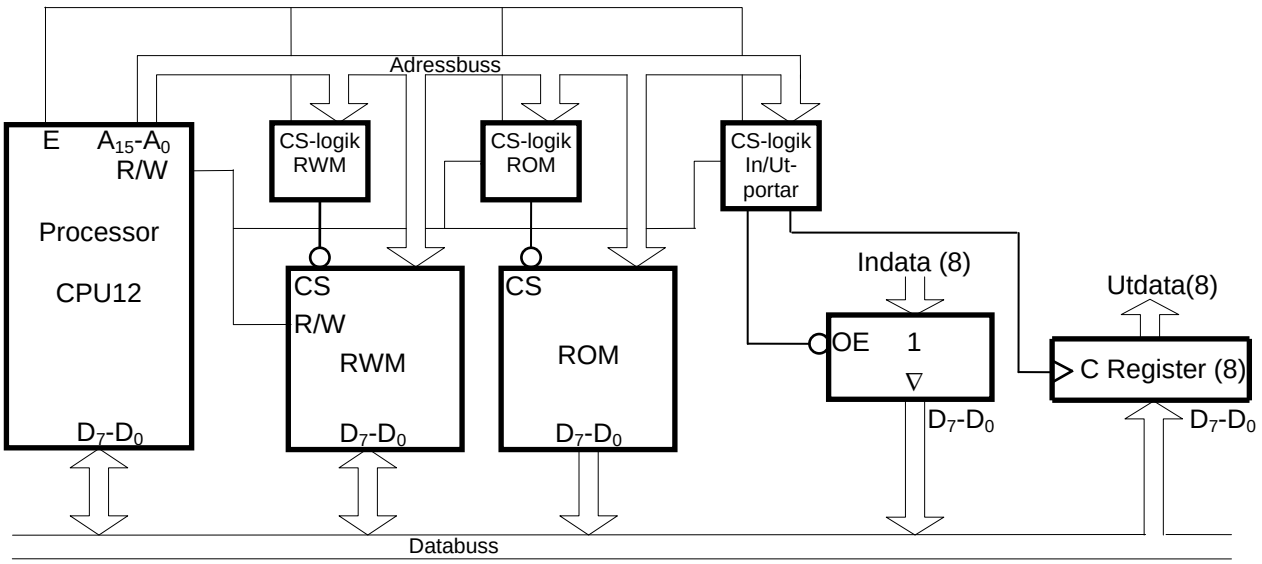
Vid anrop av subrutinen skall startadressen till STRING1 finnas i X-registret och startadressen till STRING2 i Y-registret. Endast D-registret och flaggregistret får vara förändrade vid återhopp.

(7p)

b) Skriv ett huvudprogram som sätter stackpekaren till värdet 2800_{16}, anropar subrutinen ADD10 och sedan hoppar till adressen 1900_{16}. De två strängarna STRING1 och STRING2 skall placeras direkt efter huvudprogrammet. De färdiga strängarna 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 RWM-modul, men på vissa adresser skall in- och utportar och en 16 kbyte ROM-modul vara placerade. Dessutom skall de första 800_{16} adresserna inte aktivera någon port eller minnesmodul vid läsning eller skrivning. ROM-modulen skall ha slutadressen $FFFF_{16}$. En inport och en utport skall placeras på adressen 800_{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. En dator med processorn CPU12 skall användas i styrenheten i en maskin. Styrprogrammet skall läsa av två inportar via IRQ-avbrott, PORTA 25 gånger per sekund och PORTB en gång per sekund.

Det finns en binär signal med den konstanta frekvensen 100 Hz tillgänglig för generering av avbrott. Adresserna $24F8_{16}$-$24FF_{16}$ används inte av systemet och det finns inga fler avbrottskällor.

a) Föreslå en koppling med vars hjälp man kan generera IRQ-avbrott 100 gånger per sekund. D-vippor, NAND- och NOT-grindar får användas. Avbrottssystemet används inte till något annat i datorn. **(2p)**

b) Skriv en avbrottsrutin IRQR, som läser av portarna enligt beskrivningen ovan och placerar de avlästa värdena på de symboliska adresserna VALA och VALB i minnet. Ledigt utrymme för globala variabler finns på adresserna 1420_{16}-$142F_{16}$. **(4p)**

c) Skriv ett avsnitt av huvudprogrammet där IRQ-avbrott initieras. IRQ-vektorn antas vara placerad i RWM på adresserna $FFF2_{16}$ och $FFF3_{16}$. **(3p)**

De symboliska adresserna ovan är definierade på annat ställe i programmet. 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 en funktion definieras på följande sätt och att inga övriga lokala variabler behövs:

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

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

5. (forts.)

b) Översätt C-funktionen nedan till assemblerspråk för CPU12. Visa även stackens innehåll innan for-satsen börjar utföras. (Antag att konventionerna i a-uppgiften gäller.)

```
char funcb(char m, char n) {
  char i;
  char j = n;
  for ( i = 0; i < 2; i++ )
    do j = j + m;
    while (j < 100);
return j;
}
```

(6p)

c) Vilket värde returneras från funktionen funcb om den anropas med värdena m = 20 och n = 50? 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