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	**Lördag 2013-01-19 kl 8.30 – 12.30**
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 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 h) och i) avser CPU12.

a) Assemblerinstruktionen ANDCC #$FD kan skrivas som en annan assemblerinstruktion. Vilken är denna assemblerinstruktion? **(2p)**

b) Vilken assemblerinstruktion har den hexadecimala maskinkoden 0C EA 32 10 0F? **(2p)**

c) Översätt assemblerinstruktionen SUBA -$B6,X till maskinspråk. Visa hur maskinkoden placeras i minnet. **(2p)**

d) Vilken är skillnaden mellan instruktionerna av typ ASR och LSR? När skall ASR användas och när skall LSR användas? **(2p)**

e) Översätt assemblerinstruktionen IBEQ A,$2400 till maskinspråk. Operationskoden finns på adressen 2457_{16}. Visa hur maskinkoden placeras i minnet. **(2p)**

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

f)
```
LDAA  #W
NEGA
CMPA  #$A0
BGT   Hopp
```
(3p)

g)
```
LDAB  #$10
CMPB  #W        (Tänk på att ”overflow” kan inträffa!)
BPL   Hopp
```
(4p)

h) Förklara hur ett system där CAN-noder kommunierar via en buss kan behålla synkronismen inom meddelandena trots den stora längden på dem. **(2p)**

i) Hur kodas $-\infty$ enligt IEEE-standard 754 (23 bitar av mantissan och 8 bitars karakteristika)? Ge svaret på hexadecimal form. **(1p)**

2. I minnet i ett datorsystem med processorn CPU12 finns en tabell med 75 st 8-bitars dataord som är sammansatta av två fyrabitars tal utan tecken i bit 0-3 och bit 4-7 enligt principen i figuren till höger. Tabellen börjar på adressen 2800_{16} i minnet.

TAL(4)	TAL(4)
TAL(4)	TAL(4)
TAL(4)	TAL(4)
TAL(4)	TAL(4)

a) Skriv en subrutin **NIBADD** i assemblerpråk för CPU12-processorn som för varje dataord i tabellen adderar talet i bit 0-3 med talet i bit 4-7 och skriver tillbaka summans bit 0-3 i dataordets bit 0-3 i tabellen, men lämnar bit 4-7 oförändrade.

Antalet ”overflow” som inträffar vid additionen skall räknas och detta antal skall finnas i B-registret vid återhopp. ”Overflow” avser här 4-bitars tal utan tecken.

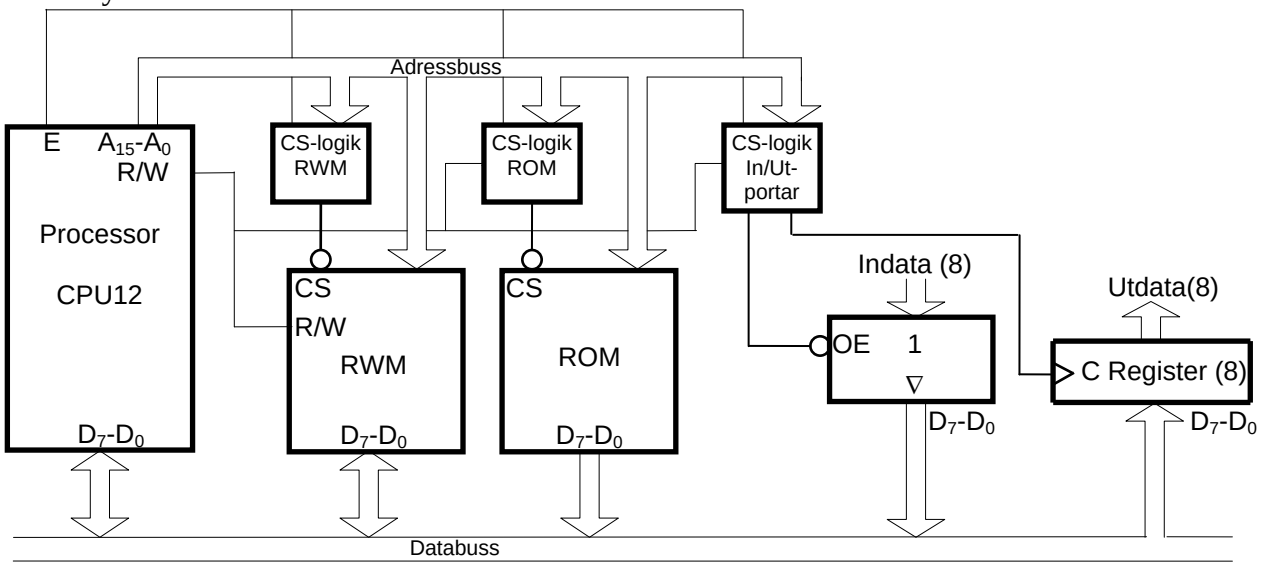
Subrutinen skall klara av tabeller med olika startadresser och olika antal dataord. Före anrop av subrutinen skall därför huvudprogrammet lägga in antalet dataord i tabellen i B-registret och startadressen till tabellen i Y-registret.

Endast register B och register CC får vara förändrade vid återhopp från subrutinen. **(8p)**

b) Skriv ett huvudprogram som initierar stacken till $3D00_{16}$ och sedan anropar subrutinen **NIBADD** på korrekt sätt enligt a). **(2p)**

För full poäng på uppgiften skall det finnas ”korrekta” radkommentarer.

3. Ett datorsystem visas nedan:



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

En 64 kbyte ROM-modul skall i princip vara placerad från adress 0, men de första 4096 adresserna i adressrummet skall inte aktivera någon minnesmodul eller port vid läsning eller skrivning.

En inport och en utport skall också anslutas. De skall ha samma adress och placeras direkt efter de första 4096 adresserna i adressrummet. Det innebär att inporten skall prioriteras före ROM-modulen på denna adress.

En 8 kbyte RWM-modul skall placeras med start på adressen $E000_{16}$, men de sista 1024 adresserna skall användas av ROM-modulen.

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

4. Man önskar använda en dator med CPU12 i styrenheten för ett mekaniskt system. Styrenheten skall kunna reagera på tre olika oberoende händelser i det mekaniska systemet genom att varje händelse genererar en avbrottssignal till CPU12 på dess IRQ'-ingång. Händelserna måste kunna identifieras så att rätt åtgärder kan vidtas av styrsystemets program.

a) Förklara hur huvudprogrammet för CPU12 skall vara utformat för att man skall kunna använda sig av avbrott så som det är tänkt. **(3p)**

b) Beskriv hur avbrottsrutinen kan se ut och varför den ser ut på detta sätt. **(3p)**

c) Vilken hårdvara krävs för att använda avbrott i detta fall? Rita ett förklarande logikschema. **(3p)**

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 assembler-språk för CPU12. Visa även hur stacken ser ut när for-satsen börjar utföras. **(7p)**

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

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

 char z = 50;

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

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

    for ( i = 0; i <= 4; i = i + 1){
      y = y + x;
      z = z + y;
    }
  return z;
  }
```

6. I häftet "Memory Systems" beskrivs "direct mapping" och två andra principer för att placera innehåll från "main memory" i cacheminnet.

a) Vad kallas de två andra principerna? **(2p)**

b) Beskriv "direct mapping" kortfattat och förklara fördelar och nackdelar med den. **(2p)**

Bilaga 1

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

000	001	010	011	100	101	110	111	$b_6b_5b_4$ $b_3b_2b_1b_0$
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