Maskinorienterad programmering - har du uppnått kursmålen

Exempel på tentamen 1

Tisdag xx januari 2005, kl. 08.30 - 12.30 i M-salar

Examinatorer
Stig-Göran Larsson, tel. 772 1693
Jan Skansholm, tel. 772 1012

Kontaktpersoner under tentamen
Som ovan

Tillåtna hjälpmedel

Häftet

"Instruktionslista för CPU12"

I den får rättelser och understrykningar vara införda, inget annat.

Tabellverk och miniräknare får ej användas!

Allmänt
Siffror inom parentes anger maximal poäng på uppgiften. Maximal poäng kan fås om:

- redovisningen av svar och lösningar är läslig och tydlig. **OBS!** Ett lösningsblad får endast innehålla redovisningsdelar som hör ihop med en uppgift.
- din lösning ej är onödigt komplicerad.
- du motiverat dina val och ställningstaganden
- redovisningen av en hårdvarukonstruktion innehåller funktionsbeskrivning, lösning och realisering.
- redovisningen av en mjukvarukonstruktion i assembler är fullständigt dokumenterad, d v s är redovisad både i strukturform (flödesplan eller pseudospråk) och med kommenterat program i assemblerspråk, om inget annat anges i uppgiften.
- av en mjukvarukonstruktion i C är dokumenterad på det sätt som lärts ut i kursen, dvs med indenterade rader. När så behövs skall den också vara indelad i moduler med användning av include-filer.

Betygsättning
För godkänt slutbetyg på kursen fordras att både tentamen och laborationer är godkända. På tentamen fordras 20p, varav minst 10p på datorteknikdelen (uppg 1-4,7) och 7p på C-delen (uppg 5-6). Tentamen ger slutbetyget:

20p ≤ **betyg 3** < 30p ≤ **betyg 4** < 40p ≤ **betyg 5**

Betygslistan
anslås såsom anges på kursens hemsida.

Granskning
Tid och plats anges på kursens hemsida.

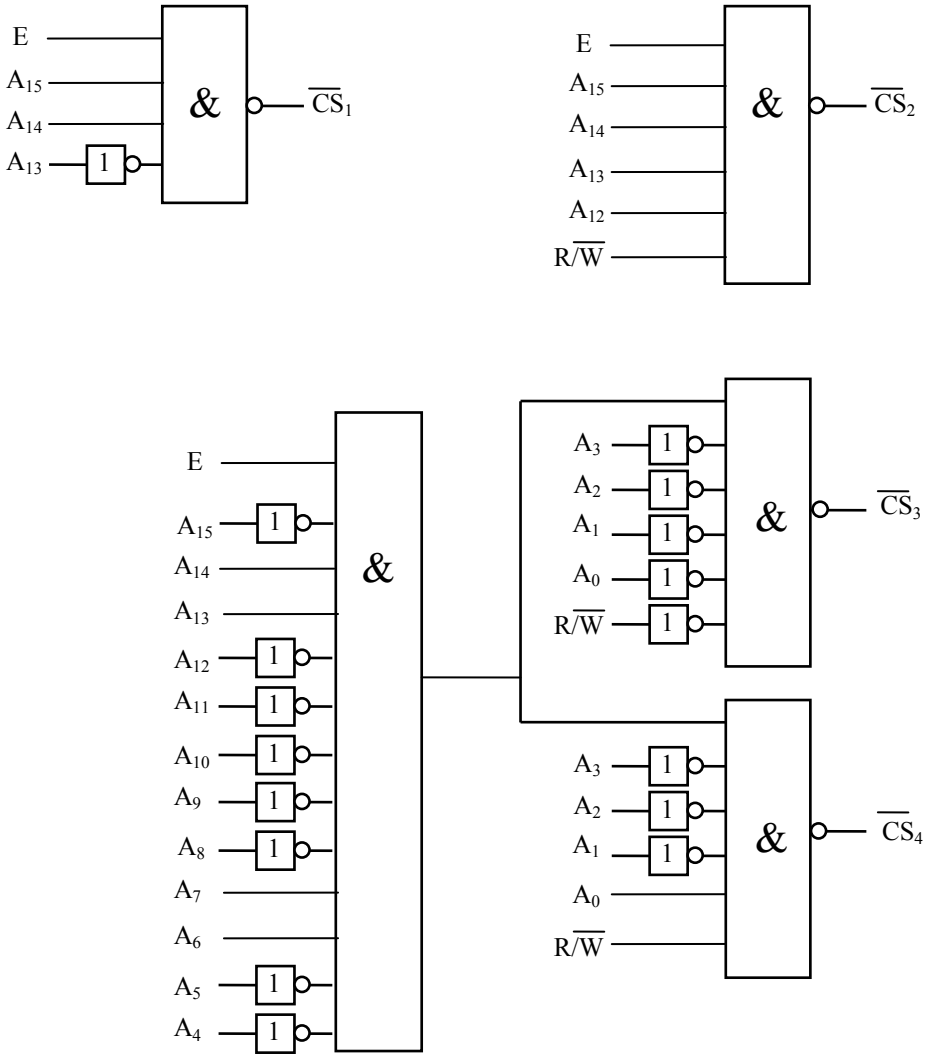
Institutionen för datorteknik
CHALMERS TEKNISKA HÖGSKOLA

1. I figur 1 visas "chip select"-logiken för ett CPU12-system. Via den ansluts diverse enheter till processorn.

a. De fyra "chip select"-signalerna är kopplade till fyra olika enheter.
Vilken typ av enhet används respektive signal till?
Förklara hur du kommer fram till ditt svar. **(3p)**

b. Vilka adresser upptar respektive enhet? **(4p)**

c. Vad är anledningen till att signalen E finns med i avkodningen? **(1p)**



Figur 1

2. Här visas början av ett program i CPU12-assemblerspråk (se instruktionslista och bilaga 3):

```
            ORG     $3250
KLIR        EQU     $3720
Port        EQU     $04D0
Number      FCB     %10111111
Portcopy    RMB     1
            LDAA    Number
            JSR     KLIR
            STAB    Port
            STAB    Portcopy
              :
```

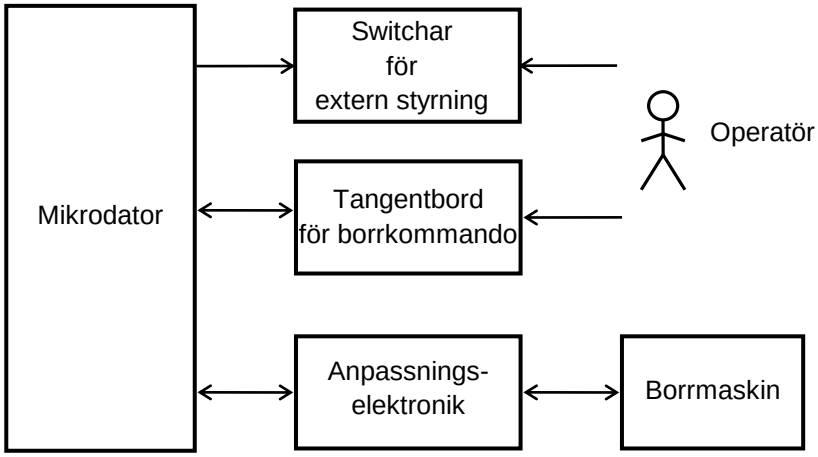
a. Rita en bild av aktuellt minnesinnehåll efter att den givna sekvensen är assemblerad och laddad till labbsystemet. **(3p)**

b. Vid användningen av monitorn i laboratoriet testade (exekverade) du dina program med ”g-kommandot” (go-kommandot), som startade exekveringen. Vilken startadress skall du använda efter g-kommandot för programmet i uppgift a)? **(1p)**

3. I borrmaskinlaborationen har du som uppgift att göra ett tillägg till systemet, så att operatören via två switchar skall kunna

- avbryta pågående program och ge ett larm, samt
- nödstoppa systemet och förbereda det för en ny borruppgift.

Genom tillägget får systemet det blockmässiga utseendet i figur 3. De två nya funktionerna skall realiseras m h a avbrott som begärs på CPU12-processorns IRQ-ingång.



Figur 3

Switcharna skall anslutas till avbrottsingången via var sin D-vippa och en NOR-grind. En nedtryckning (= positiv puls) av någon av switcharna skall medföra att IRQ-signalen (Interrupt ReQuest) aktiveras. D-vipporna kan anslutas till bitpositionerna **7** och **6** av inporten **IRQSRC**, så att den innehåller avbrottsvippornas status.

a. Visa kopplingen för hur switcharna ansluts till CPU12 och port IRQSRC. **(2p)**

b. Skriv initieringsrutinen IRQINIT i CPU12-assemblerspråk. **(4p)**

c. Skriv avbrottsrutinen IRQR i CPU12-assemblerspråk. **(4p)**

IRQINIT och IRQR har beskrivningarna:

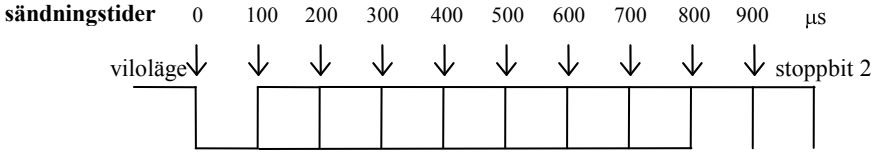
IRQINIT

Beskrivning:	Rutinen initierar kopplingen och CPU12 för IRQ-avbrott. Den använder laborationsdatorns inportar IRES1 och IRES2 för eventuella "dummy-läsningar".
Anrop:	BSR IRQINIT
Indata:	Inga
Utdata:	Inga
Registerpåverkan:	Ingen
Anropade subrutiner:	Inga

IRQR

Beskrivning:	Avbrottsrutinen reagerar på begäran om nödstopp och larm beroende på på vilken switch som aktiverats. Förloppen är mycket snabba varför man kan förutsätta att endast en av switcharna begär avbrott åt gången. Skulle både nödstopp och larm begäras samtidigt så skall nödstopp ha högre prioritet än larm. Därför skall också ett larm avbrytas av en begäran om nödstopp.
Anrop:	Avbrottsbegäran på IRQ
Indata:	Status på port IRQSRC
Utdata:	Inga
Registerpåverkan:	Ingen
Anropade subrutiner:	SIGNAL

4. Antag att ASCII-tecken överförs m h a ett asynkront serieinterface. Överenskommelsen är ett format om 6 teckenbitar, jämn paritet och två stoppbitar, samt bithastigheten 10 kbps. Relativt startbitens början läggs då bitarna ut vid de tidpunkter som visas i figur 4.



Figur 4

Mottagaren använder sig av frekvensen $f_m = k \cdot 10$ kHz = 160 kHz för att sampla den mottagna signalen. Periodtiden är då $T_m = 1/ f_m = 6{,}25$ µs. Mottagaren samplar då signalen vid tidpunkterna

$$t_{s1} = \varepsilon \qquad t_{s2} = \varepsilon + 8 \cdot T_m \qquad t_{s3} = \varepsilon + 24 \cdot T_m \qquad t_{s4} = \varepsilon + 40 \cdot T_m \quad \text{osv}$$

a. Det första samplet tas alltså vid tidpunkten ε. Förklara bakgrunden till det. Vilka värden kan ε ha? **(3p)**

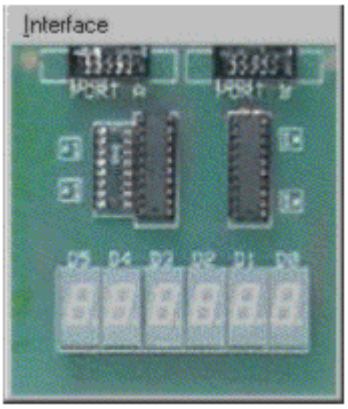
I en tillämpning upptäcker man på mottagarsidan, vid undersökning av serieinterfacets statusregister, att fel uppstår i överföringen. För ett visst sänt ord så får man både paritetsfel och ramfel ("framing error").

b. Vad beror detta på? **(1p)**

c. Förklara hur det uppstår! **(2p)**

5. Till laborationsutrustningen kan man som bekant ansluta ett kort (ML3) som bl.a. innehåller en displayenhet. En sådan innehåller sex indikatorer vilka var och en kan visa en hexadecimal siffra. Följande beskrivning är klippt ur hjälptexten till XCC:

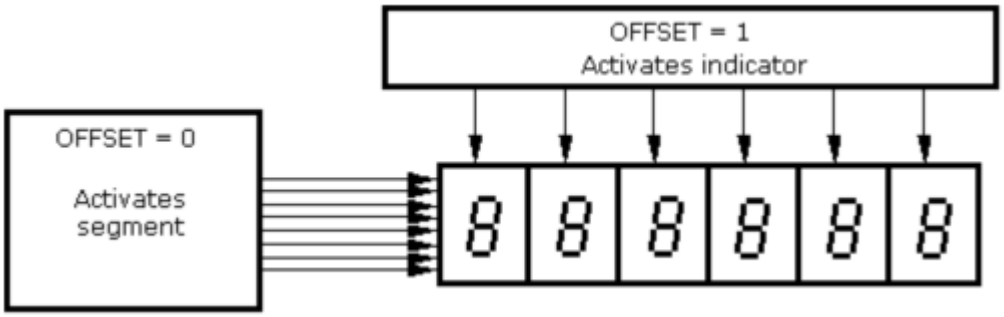
ML3 Display can be used with two different interfaces; *ML15* or *ML5*.



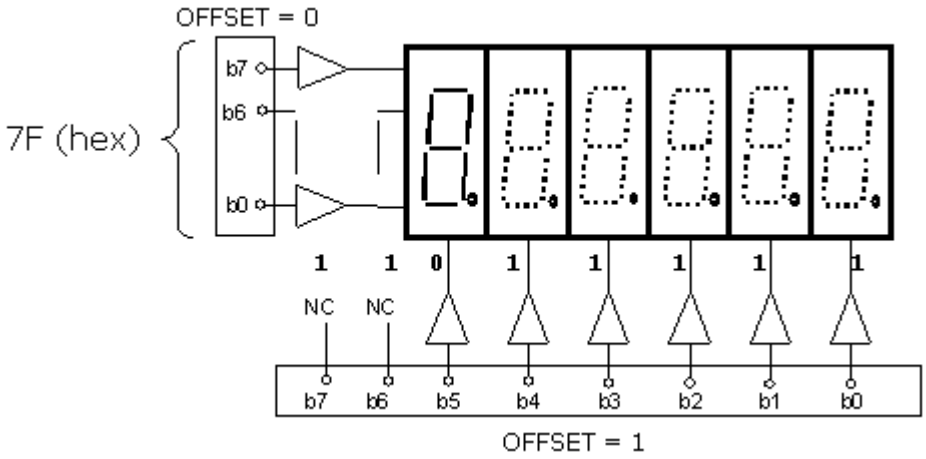
ML3 illustrates a ramp of six "7-segment-display ".
- *ML3* can show six digits at a time
- *ML3* maps bit-pattern to segments different

ML5 emulation

The interface is two 8-bit registers located at addresses BASE and BASE+1.:



An indicator is activated by writing zero to the corresponding bit in the register at address BASE+1.



The following table shows how bit patterns (Decimal numbers 0 to 15) are translated to segment patterns.

Decimal	Segment codes	
	Binary	Hex
0	0111 0111	77
1	0010 0100	24
2	0101 1101	5D
3	0110 1101	6D
4	0010 1110	2E
5	0110 1011	6B
6	0111 1011	7B
7	0010 0101	25
8	0111 1111	7F
9	0110 1111	6F
10	0011 1111	3F
11	0111 1010	7A
12	0101 1000	58
13	0111 1100	7C
14	0101 1011	5B
15	0001 1011	1B

Uppgiften är att skriva en C-funktion `display_hex` som använder gränssnittet ML5 för att visa ett heltal på hexadecimal form på displayenheten. Basadressen till ML5 är 0xC02. Parametern till funktionen skall vara av typen **`unsigned int`**. Alla sex indikatorerna i displayenheten skall användas för att visa talet. Om en **`unsigned int`** består av fler än 24 bitar så skall de 24 minst signifikanta bitarna visas. För enkelhets skull får du förutsätta att när man tänt en viss indikator i displayenheten så kommer denna att förbli tänd, även om den inte längre är aktiverad. (Så fungerar det inte i praktiken, men det kanske borde göra det.)

Utorma din lösning så att att definitioner av portar samt de typer som behövs för att beskriva dem ligger i en egen include-fil. Använd också en separat includefil som innehåller en *deklaration* av funktionen `display_hex`.

(10p)

6. Skriv en egen version av C-standardfunktionen `strcmp`. Du *måste* använda pekare, inte indexering! I denna uppgift får du inte använda dig av några färdiga standardfunktioner, utan du måste skriva allt själv.

(8p)

---

7. Frågor

a. Förklara kortfattat begreppen *händelse*, *aktivitet* och *svarstid* så som de används i realtidssammanhang. **(2p)**

b. Förklara, gärna med en figur, begreppen *seriell exekvering* (non-pre-emptive) och *pseudoparallell exekvering* (pre-emptive) i realtidssammanhang. **(2p)**

Bilaga 1 - Assemblerspråket för mikroprocessorn CPU12.

Assemblerspråket använder sig av de mnemoniska beteckningar som processorkonstruktören MOTOROLA specificerat för maskininstruktioner och instruktioner till assemblatorn, s k pseudoinstruktioner eller assemblatordirektiv. Pseudoinstruktionerna framgår av följande tabell:

Direktiv	Förklaring
ORG N	Placerar den efterföljande koden med början på adress 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 adress L. (RMB för Reserve Memory Bytes)
L EQU N	Ger label L konstantvärdet N (EQU för EQUates = beräknas till)
L FCB N1, N2	Avsätter i följd i minnet en byte för varje argument. Respektive byte ges konstantvärdet N1, N2 etc. Följden placeras med början på adress L. (FCB för Form Constant Byte)
L FDB N1, N2	Avsätter i följd i minnet ett bytepar (två bytes) för varje argument. Respektive bytepar ges konstantvärdet N1, N2 etc. Följden placeras med början på adress L. (FDB för Form Double Byte)
L FCS "ABC"	Avsätter en följd av bytes i minnet, en för varje tecken i teckensträngen "ABC". Respektive byte ges ASCII-värdet för A, B, C etc. Följden placeras med början på adress L. (FCS för Form Caracter String)

Bilaga 2 - ASCII-koden.

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