

DAT015 Maskinorienterad programmering IT

Tentamen

Fredag 21 april 2006, kl. 14.00 - 18.00 i V-salar

Examinatorer

Rolf Snedsböl, tel. 0734 30 93 79

Jan Skansholm, tel. 0707 16 32 30

Kontaktpersoner under tentamen

Som ovan

Tillåtna hjälpmedel

Häftet

Instruktionslista för CPU12

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

Du får också använda bladet

C Reference Card

samt boken

*Vägen till C, Bilting, Skansholm,
Studentlitteratur*

Även i denna får rättelser och understrykningar vara införda, inget annat.

Tabellverk och miniräknare får ej användas!

Allmänt

Siffror inom parentes anger full poäng på uppgiften. **Full 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 pseudopråk) och med kommenterat program i assemblerspråk, om inget annat anges i uppgiften.
- C-program är utformade enligt de råd och anvisningar som givits under kursen. I programtexterna skall raderna dras in så att man tydligt ser programmets struktur. När så anges skall programtexten 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) och 7p på C-delen (uppg 5-6). Tentamen ger slutbetyget:

$20p \leq \text{betyg } 3 < 30p \leq \text{betyg } 4 < 40p \leq \text{betyg } 5$

Lösningar

anslås på kursens [www hemsida](http://www.hemsida).

Betygslistan

anslås såsom anges på kursens [hemsida](http://www.hemsida).

Granskning

Tid och plats anges på kursens [hemsida](http://www.hemsida).



1. Konstruera adressavkodningen för ett MC12-system där vi önskar en yttre ROM-modul och en yttre RWM-modul. Dessutom skall två 8-bitars IO-portar finnas. Alla chip select signaler är aktiva låga

RWM-modulen som är 16-kbyte skall placeras med start på adress \$0000. ROM-modulen skall placeras med start på adress \$C000 och är 32-kbyte stor. Använd fullständig adressavkodningslogik för minnesmodulerna.

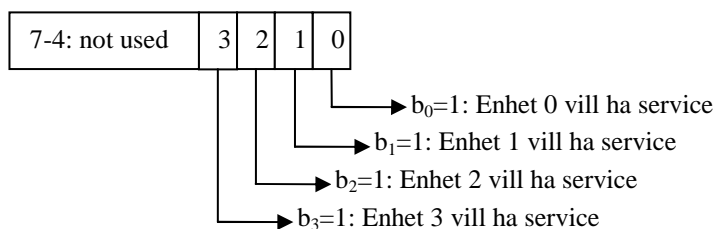
Inporten och utporten placeras båda på adress \$B000. Använd så få grindar som möjligt (ofullständig adressavkodning) när du konstruerar adressavkodningen för IO-portarna.

Rita adressavkodningen där du även gör en skiss av minnesmodulerna och IO-portarna. **(8p)**

2. Polling i ett icke avbrottsdrivet system.

Ett CPU12-system skall användas för att betjäna fyra yttre enheter numrerade 0 t o m 3.

Oberoende av varandra kan enheterna begära att bli servade. Begäran om service görs genom att en till enheten hörande statusflagga ettställs. Enheternas statusflaggor, som också numreras 0 t o m 3, har i ordningsföljd samlats i bitarna 0 - 3 av ett statusregister på adress 0C00H. Se figur



Enheternas servicerutiner finns tillgängliga och har lagrats som subrutiner med namnen DSR0 – DSR3. En enhets statusflagga nollställs i slutet av servicerutinen.

Skriv ett huvudprogram i CPU12 assemblerspråk som oavbrutet övervakar enheterna. De skall "pollas" i "round robin"-ordning. Programmet skall dokumenteras i enlighet med texten på försättsbladet. **(7p)**

3. Redogör för hur avbrott går till i ett HC12-system där du har en yttre enhet ansluten till IRQ-ingången på HC12:an!

Du skall ge kodexempel på en rutin som initierar systemet för avbrott och kodexempel på en typisk avbrottsrutin. **(6p)**

4. Småfrågor.

- a. I CAN-protokollet används ett asynkront protokoll där man skickar okodade bitar (dvs bitar utan inlagd klockinformation). Beskriv hur synkroniseringen bibehålls mellan sändare och mottagare. **(3p)**
- b. "Deadline" är ett vanligt begrepp för ett realtidssystem och det beskrivs med en värdefunktion. Vad menar man med en "deadline's värdefunktion"? **(3p)**
- c. Vad menas med ett synkront respektive asynkront protokoll i samband med seriekommunikation. Ange fördelar och nackdelar. **(3p)**

5. Skriv ett C-program som läser in ett antal heltal (högst 1000 st) och som skriver ut dem i samma ordning som de lästes in. Vid inmatningen av talen på tangentbordet skall de kunna skrivas i fritt format med ett eller flera tal på varje rad. Inmatningen avslutas med att användaren skriver ctrl-Z (eller motsvarande). När talen sedan skrivs ut av programmet skall ett visst tal bara skrivas en gång. Om det skrivits ut tidigare skall det alltså inte skrivas ut igen.

Exempel: Det kan se ut på följande sätt när man kör programmet. (De kursiverade raderna har skrivit av programmet.)

Skriv in talen:

```
45 77 -22 3 45 0
21
-1 3
```

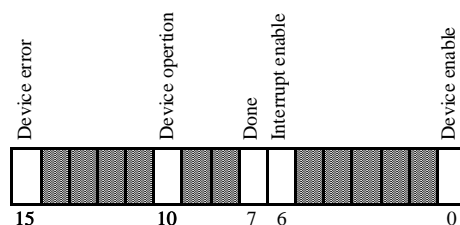
Talen är:

```
45 77 -22 3 0 21 -1
```

(8p)

6. I en gruva samlas allt vatten längst ner i en behållare. En pump har till uppgift att då och då pumpa upp vatten ur behållaren så att gruvan inte blir vattenfylld. Men i gruvan bildas metangas och blir halten av denna gas för hög kan man p.g.a. explosionsrisken inte starta pumpen, utan måste vänta tills metangashalten minskat. Uppgiften är att skriva ett program som övervakar metangashalten och kontrollerar vattennivån med hjälp av pumpen.

Det finns fyra externa enheter anslutna till den processor där programmet skall exekveras: två sensorer som känner om vattennivån i behållaren stiger över maximinivån respektive sjunker under miniminivån, en pumpmotor samt en A/D-omvandlare som mäter metangashalten. Till varje extern enhet hör ett 16-bitars kontrollregister med följande principiella utseende.



Till A/D-omvandlaren hör dessutom ett 16-bitars dataregister. Registren har följande adresser:

Sensor för maximinivå:	AA10
Sensor för miniminivå:	AA12
Pumpmotor:	AA14
A/D-omvandlare för metangas:	AA16
Dataregister till A/D-omvandlare:	AA18

Observera att kontrollregistren *inte* är läsbara. Man kan bara skriva till dem. Du måste därför använda dig av skuggregister. För samtliga externa enheter gäller att biten "Device enable" måste vara satt till ett för att enheten överhuvudtaget skall fungera. (Denna bit fungerar alltså som en slags "huvudströmbrytare.") Pumpen startas och stoppas genom att en etta resp. nolla skrivs i biten "Device operation". När man skall göra en avläsning av A/D-omvandlaren skriver man en etta i "Device operation" och måste sedan vänta en halv sekund innan värdet i dataregistret har stabiliserats. A/D-omvandlaren ger värden i intervallet 0 till 1023. Metangashalter större än 400 på denna skala räknas som farliga. De två vattennivåsensorerna ger utslag genom att generera avbrott till processorn. Deras avbrottsadresser är 40 respektive 44 (hexadecimalt).

(forts)

Du får utnyttja alla de funktioner och makrodefinitioner du skrev i den sista laborationen, t.ex. makrona `set` och `clear` samt funktionerna `init_clock` och `hold`. För enkelhets skull får du också anta att det finns två färdiga avbrottsrutiner med namnen `max_trap` och `min_trap`. Det enda de gör är att anropa C-funktionerna `max_inter` resp. `min_inter`. Du kan däremot *inte* förutsätta att adresserna till avbrottsrutinerna är inlagda i avbrottsvektorer.

Din uppgift är att göra de återstående definitioner som behövs samt att skriva C-funktionerna `main`, `max_inter` resp. `min_inter`. Funktionerna `max_inter` och `min_inter` skall, när de anropas, starta resp. stoppa pumpen. Funktionen `main` skall, efter att ha gjort nödvändiga initieringar, kontinuerligt (en gång var 10:e sekund) avläsa metangashalten. Om halten blir för hög skall ett alarmmeddelande ges till operatören (för enkelhets skull får du använda `printf`). Dessutom skall huvudströmbrytaren till pumpen slås av. När metangashalten åter sjunker under det farliga värdet skall operatören ges ett meddelande om detta. Huvudströmbrytaren till pumpen skall då också slås på.

(12 p)

Bilaga 1 - Assemblerspråket för mikroprocessorn CPU12.

Assemblerspråket använder sig av de mnemoniska beteckningar som processorkonstruktören MOTOROLA specificerat för maskininstruktioner och instruktioner till assemblern, så som pseudoinstruktioner eller assemblerdirektiv. 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 Caracater 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

Preliminära, kortfattade lösningar och svar

- ROM. 4kbyte $\Rightarrow 2^2 \cdot 2^{10}$ byte \Rightarrow 12 Adressbitar \Rightarrow [A11,A0] direkt till ROM-kapsel.
 RWM. 16kbyte $\Rightarrow 2^4 \cdot 2^{10}$ byte \Rightarrow 14 Adressbitar \Rightarrow [A13,A0] direkt till RWM-kapsel.

Minnesmodulerna och I/O-portarna tar upp följande adressområden:

		A ₁₅	A ₁₄	A ₁₃	A ₁₂	A ₁₁	A ₁₀	A ₉	A ₈	A ₇	A ₆	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀
RWM	Start: 0000 -	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Stop: 3FFF	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ROM	Start: C000	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Stop: CFFF	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1
I/O	8000	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CS-ROM: Rita en NAND-grind enligt {A₁₅·A₁₄·A₁₃·A₁₂·E R/W}'

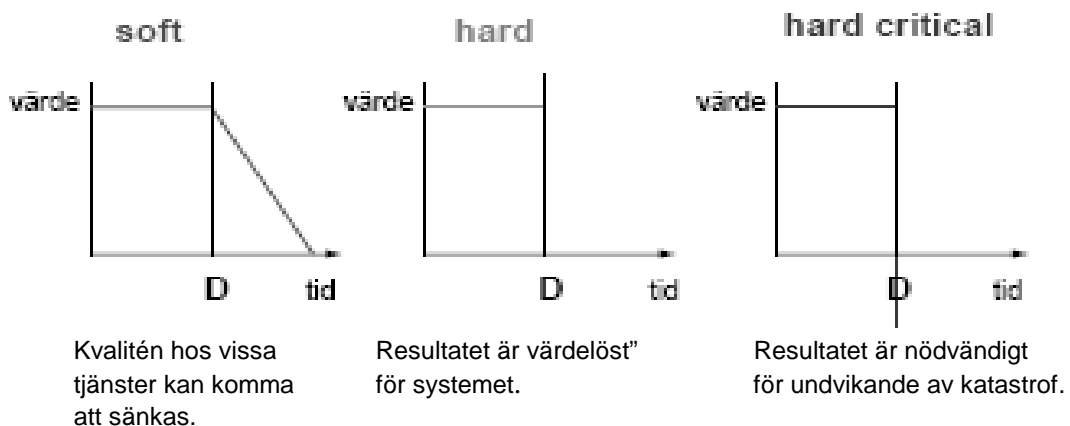
CS-RWM: Rita en NAND-grind enligt {A₁₅·A₁₄·E}'

I/O: Bilda först IOSEL för både portarna enligt {A₁₅ A₁₄·E}' Rita sedan OUTSEL enligt {IOSEL R/W}' och sedan INSEL enligt {IOSEL R/W}'

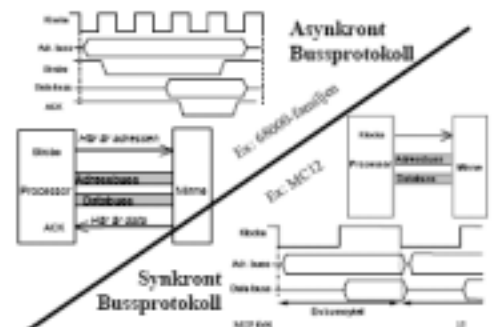
- Studera figuren på sid 45 i arbetsboken och utöka denna med en extra knapp, vippa och NAND-grind. Vidare krävs ett statusregister enligt ML19 (avbrottsmodulen du använde i labbet). Studera sedan sidorna 44-46, OH-material och laboration 2.

3. Småfrågor

- Hård synkronisering på startbiten och bibehållen synkronisering på flanker i mottaget data. Vid långa sända sekvenser av ettor eller nollor infogas stuffbitar i sändt data. Skickas minst 5 ettor i följd infogas en nolla och skickas minst 5 nollor infogas en etta i sänt data för att generera flanker.
- En deadline's värdefunktion beskriver resultatet av det som händer om systemet ej levererar sitt svar i tid: Man skiljer på



- Asynkront: Handskakning. Olika snabba minnen
 Synkront: Enbart klocka. Minnet: snabbare än klockan
- Pass 1: Skapa symboltabell;
 Pass 2: Lägg ut kod
- Studera lab 1 och 2



UPG 4

```
// Alternativ 1 - användning av indexering
#include <stdio.h>
#define MAXANTAL 1000
int main() {
    int a[MAXANTAL];
    int i,j,k;
    printf("Skriv talen:\n");
    for (i=0; i < MAXANTAL && scanf("%d", &a[i]) == 1; i++)
        ;
    printf("Talen är:\n");
    for (j=0; j<i; j++) {
        for (k=0; k<j && a[k] != a[j]; k++)
            ;
        if (k==j)
            printf("%d ", a[j]);
    }
}
```

```
// Alternativ 2 - användning av pekare
#include <stdio.h>
#define MAXANTAL 1000
int main() {
    int a[MAXANTAL];
    int *p, *q, *r;
    printf("Skriv talen:\n");
    for (p=a; p<a+MAXANTAL && scanf("%d", p) == 1; p++)
        ;
    printf("Talen är:\n");
    for (q=a; q<p; q++) {
        for (r=a; r<q && *r != *q; r++)
            ;
        if (r == q)
            printf("%d ", *q);
    }
}
```

UPG 5

```
// Filen gruva.h
typedef int port;
typedef int *portptr;
typedef void (**vecptr) (void); // pekare till element i avbrottsvektorn
```

```
// Användbara makron
#define set(r, mask) (r) = (r) | mask
#define clear(r, mask) (r) = (r) & ~mask

#define MAX_CTRL_ADR 0xaa10
#define MIN_CTRL_ADR 0xaa12
#define PUMP_CTRL_ADR 0xaa14
#define METAN_CTRL_ADR 0xaa16
#define METAN_DATA_ADR 0xaa18
```

```
#define MAX_CTRL      *((portptr) MAX_CTRL_ADR)
#define MIN_CTRL      *((portptr) MIN_CTRL_ADR)
#define PUMP_CTRL     *((portptr) PUMP_CTRL_ADR)
#define METAN_CTRL    *((portptr) METAN_CTRL_ADR)
#define METAN_DATA    *((portptr) METAN_DATA_ADR)

#define MAX_CTRL_VEC_ADR 0x40 // Adress till avbrottsvektor
#define MIN_CTRL_VEC_ADR 0x44 // Adress till avbrottsvektor
#define MAX_CTRL_VEC     *((vecptr) MAX_CTRL_VEC_ADR)
#define MIN_CTRL_VEC     *((vecptr) MIN_CTRL_VEC_ADR)

#define device_enable_bit 0x0001
#define interrupt_enable_bit 0x0040
#define device_operation_bit 0x0400

void max_trap(void);
void min_trap(void);

#include "gruva.h"
#include "clock.h"
#include <stdio.h>

#define farligt 400

// skuggregister
port max_ctrl = 0;
port min_ctrl = 0;
port pump_ctrl = 0;
port metan_ctrl = 0;

void max_inter(void) {
    set(pump_ctrl, device_operation_bit);
    PUMP_CTRL = pump_ctrl;
}

void min_inter(void) {
    clear(pump_ctrl, device_operation_bit);
    PUMP_CTRL = pump_ctrl;
}

int main() {
    port metanhalt;

    set(max_ctrl, device_enable_bit);
    set(min_ctrl, device_enable_bit);
    set(max_ctrl, interrupt_enable_bit);
    set(min_ctrl, interrupt_enable_bit);
    set(pump_ctrl, device_enable_bit);
    set(metan_ctrl, device_enable_bit);
    MAX_CTRL = max_ctrl;
    MIN_CTRL = min_ctrl;
    PUMP_CTRL = pump_ctrl;
```



```
METAN_CTRL = metan_ctrl;
MAX_CTRL_VEC = max_trap;
MIN_CTRL_VEC = min_trap;
init_clock();

while(1) {
    hold(10000);
    set(metan_ctrl, device_operation_bit);
    METAN_CTRL = metan_ctrl;
    hold(500);
    metanhalt = METAN_DATA;
    if (metanhalt > farligt) {
        if ((pump_ctrl & device_enable_bit)) {
            clear(pump_ctrl, device_enable_bit);
            PUMP_CTRL = pump_ctrl;
            printf("Varning! Hög metangashalt!");
        }
    }
    else {
        if (!(pump_ctrl & device_enable_bit)) {
            set(pump_ctrl, device_enable_bit);
            PUMP_CTRL = pump_ctrl;
            printf("Normal metangashalt!");
        }
    }
}
}
```