Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA

Tentamen med lösningsförslag

EDA482 (EDA481) Maskinorienterad programmering D
EDA487 (EDA486) Maskinorienterad programmering Z
DAT017 (DAT016) Maskinorienterad programmering IT
DIT151 Maskinorienterad programmering GU

Lördag 2 juni 2018, kl. 8.30 - 12.30

Examinator
Roger Johansson, tel. 772 57 29

Kontaktperson under tentamen:
Andreas Wieden, tel. 772 10 23

Tillåtna hjälpmedel

Utgåvor som distribuerats inom ramen för kursen, häftet:

- *Quick Guide, Laborationsdator MD407 med tillbehör*

Inget annat än understrykningar (”överstrykningar”) får vara införda i dessa häften.

Tabellverk eller miniräknare får ej användas.

Lösningar
anslås senast dagen efter tentamen via kursens hemsida.

Granskning
Tid och plats anges på kursens hemsida.

Allmänt
Siffror inom parentes anger full poäng på uppgiften.

För full poäng krävs att:

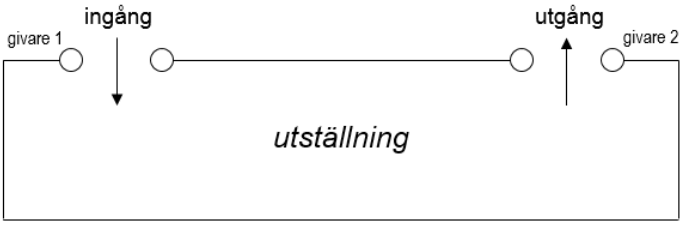
- redovisningen av svar och lösningar är läslig och tydlig. Ett lösningsblad får endast innehålla redovisningsdelar som hör ihop med en uppgift.
- lösningen ej är onödigt komplicerad.
- du har motiverat dina val och ställningstaganden
- assemblerprogram är utformade enligt de råd och anvisningar som getts under kursen.
- C-program är utformade enligt de råd och anvisningar som getts under kursen.
  I programtexterna skall raderna dras in så att man tydligt ser programmens struktur.

Betygsättning
För godkänt slutbetyg på kursen fordras att både tentamen och laborationer är godkända.

Maximal poäng är 50 och tentamenspoäng ger slutbetyg enligt: (EDA/DAT/LEU):
$20p \leq$ betyg 3 $< 30p \leq$ betyg 4 $< 40p \leq$ betyg 5
respektive (DIT):
$20p \leq$ betyg G $< 35p \leq$ VG

Uppgift 1 (16p)

I en utställningslokal har man installerat givare vid såväl ingång som utgång, se figuren till höger.



Man behöver nu ett mikrodatorbaserat system som håller reda på hur många besökare som befinner sig i lokalen och konstruerar därför en *givarkrets* med gränssnitt där givarsignalerna ansluts till de åtta minst signifikanta bitarna i port E hos en *MD407*.

De åtta mest signifikanta bitarna i porten ansluts till en sifferindikator där antalet besökare visas.

GPIO	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E	gränssnitt indikator								gränssnitt givare							
	w	w	w	w	w	w	w	w	r	w		rw	w	w	r	r

För varje person som passerar en givare ettställs någon av bitarna b0 och b1, Om avbrott från givarkretsen aktiverats ettställs också b7. Bitarna kan återställas genom att en etta skrivs till b6, b2 och/eller b3. Det får förutsättas att inpassering endast kan ske genom ”ingång” och utpassering endast kan ske genom ”utgång”.

Anm: r: biten är läsbar, w:biten är skrivbar
PE0: biten ettställs vid passage av givare 1.
PE1: biten ettställs vid passage av givare 2.
PE2: en skrivning av ’1’ till bit 2 nollställer (återställer) bit 0.
PE3: en skrivning av ’1’ till bit 3 nollställer (återställer) bit 1.
PE4: biten sätts till 1 för att aktivera avbrott från givarkretsen.
PE6: en skrivning av ’1’ till bit 6 nollställer (återställer) bit 7.
PE7: om avbrott från givarkretsen är aktiverat sätts denna bit till 1 då någon av bitarna b0 eller b1 har satts av givarkretsen. Systemets avbrottstabell har relokerats till adress 0x2001C000.

a) (3p) Visa en funktion **`void`** `init_app(`**`void`**`)`, där port E initieras för att användas tillsammans med givarkretsen. Alla statussignaler från givarkretsen är ”flytande” och aktivt höga. Ett internt *pull-down* motstånd ska därför programmeras. Styrsignalerna till givarkretsen ska vara av typen *push-pull*.

b) (4p) Avbrottssignalen från givarkretsen ska kopplas till avbrottsystemet.
Funktionen **`void`** `sensor_irq(`**`void`**`)` ska anropas vid varje avbrott.
Visa en funktion **`void`** `init_irq(`**`void`**`)`där modulerna SYSCFG, EXTI och NVIC konfigureras för användning med givarkretsens avbrott. Endast de bitar som ska konfigureras får ändras, övriga ska ha kvar sina initialvärden.

c) (3p) Deklarera en variabel `visitors` av heltalstyp och med synlighet bara i källtextfilen. Variabeln används för att ange antalet personer som för tillfället befinner sig i utställningslokalen.Visa en avbrottsrutin **`void`** `sensor_irq(`**`void`**`)` som håller reda på antalet samtidiga besökare av utställningen genom att addera/subtrahera in- och utpasseringar och placera resultatet i variabeln. Observera att in- och utpassering kan ske samtidigt.

Antalet besökare i utställningen ska skrivas till en visningsenhet. Utskriften ska uppdateras var tionde sekund.

d) (3p) Visa hur SYSTICK kan användas för att skapa en blockerande fördröjning om 10 sekunder. Observera att kretsen inte medger så långa fördröjningar, basera därför i stället din lösning på 100ms fördröjning.

e) (3p) En utskriftsenhet enligt figuren till höger visar två hexadecimala siffror. Konstruera en funktion **`void`** `main(`**`void`**`)` som visar antalet besökare (max 255 st.) av utställningen. Programmet ska först göra alla nödvändiga initieringar och därefter kontinuerligt visa antalet besökare på visningsenheten. Visningen ska uppdateras var tionde sekynd. Du får använda lösningar från tidigare uppgifter även om du inte gjort dessa.

Uppgift 2 (10p)

a) (4p) Följande C-funktion `pack` packar födelsedata (år, månad dag) i en `unsigned int`, enligt:

31	30	29	28	27	26	25	34	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	year													month				day				

```
int pack( unsigned short year, unsigned short month, unsigned short day)
{
  unsigned int rval = year << 9;
  rval |= month << 5;
  rval |= day;
  return rval;
}
```

Visa hur funktionen kan kodas i ARM v6 assemblerspråk. Använd GCC:s kompilatorkonventioner.

b) (6p) Visa hur adressen till uttrycket: `vs[i+8][j+2]` evalueras till register `R0` i ARM v6 assemblerspråk då följande deklarationer gjors på ”toppnivå” `int i,j; short vs[15][32];`

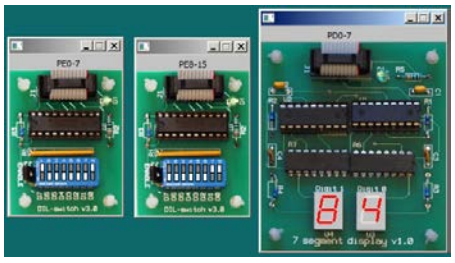
Uppgift 3 (8p)

För ett fält med tecken (`char`) kan man definiera en operation *rotation vänster* på fältet som en operation som placerar fältets första tecken på den sista platsen i fältet, det andra tecknet på den första platsen, det tredje tecknet på den andra platsen o.s.v.
Skriv en funktion `rotateleft` som roterar ett fält med tecken ett godtyckligt antal steg åt vänster. Funktionen skall ha tre parametrar, en pekare till fältet som skall roteras samt två heltal (`int`) vilka anger fältets längd respektive antalet rotationssteg. Parametrarna ska ges i denna ordning, funktionen har inget returvärde och får inte ha sidoeffekter.
Lösningen får inte använda anrop till standardfunktioner från bibliotek, koden får heller inte använda indexering utan måste implementeras med hjälp av pekartyper.

Uppgift 4 (6p)

a) (4p) Två DIL-strömbrytare och en 7-segments display kopplas enligt:

- DIL1 till Port E (0-7),
- DIL2 till Port E (8-15)
- Display till port D (0-7):



Utgå från att alla portar är korrekt initierade och skriv ett program i ARM v6 assemblerspråk som kontinuerlig läser de inställda värdena från DIL-strömbrytarna, adderar dessa och skriver resultatet till displayen. Om summan är större än 255 ska ***FF*** skrivas till displayen.

b) (1p) Varför behöver vi i bland använda det reserverade ordet `volatile`?

c) (1p) När behöver vi använda så kallade *include-guards*?

Uppgift 5 (10p)

En signalenhet som omvandlar externa signaler är ansluten till en dator. Signalenheten har 512 ingångar (kanaler), numrerade från 0 till 511. Signalenheten kan bara avläsa en ingång i taget. Det avlästa värdet ges som ett positivt 24-bitars tal. Signalenheten kopplas till datorn via ett gränssnitt med följande register:

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																	status								ctrl							
																				IRQ		ER		DN	IE	IA			EN			RS
																				r		r		r	rw	rw			w1			w1
4																								channel								
																								w	w	w	w	w	w	w	w	w
8									data																							
									r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
0xC	ivr																															
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w		

Anm: r: biten är läsbar, w:biten är skrivbar, w1: biten återställs automatiskt till 0 av gränssnittet efter en skrivning '1'.

IRQ: *Interrupt Request*. Om avbrott är aktiverat sätts även denna bit till '1' då någon av DN- eller ER- bitarna sätts till '1' av gränssnittet.

ER: *Error*. Vid fel som gör att omvandlingen inte kan fullföljas, sätts denna bit till '1' av gränssnittet.

DN: Done, sätts till '1' av gränssnittet då en omvandling är slutförd och dataregistret är klart för avläsning.

IE: *Interrupt Enable*, sätts till '1' för att aktivera avbrottsmekanismen i gränssnittet. Då avbrottsmekanismen är aktiverad genereras ett avbrott till processorn via IRQ-biten i statusregistret. Avbrottet är kopplat till processorn som använder innehållet i ivr som avbrottsvektor för signalenheten.

IA: *Interrupt Acknowledge*, då denna bit sätts till '1' återställs biten IRQ till 0 av signalenhetens gränssnitt.

EN: *Enable*, då '1' skrivs till denna bit aktiveras signalenheten och en omvandling startas.

RS: *Reset*, då '1' skrivs till denna bit återställs signalenheten. Detta måste göras inför varje omvandling.

En omvandling börjar med att signalenheten återställs och aktiveras. Därefter kontrolleras statusregistret och då omvandlingen är klar kan resultatet läsas från dataregistret. Då signalenheten används med avbrott måste vektor registret (ivr) först ha initierats med adressen till avbrottsfunktionen.

a) (6p) Visa hur gränssnittet kan avbildas med en `struct`-typ (`SIGUNIT`). Signalenhetens register har placerats med start på adress 0xF0000000 i minnet. Visa också hur du definierar en konstant som en pekare (`SIGUNITP`) som sedan kan användas för att adressera signalenhetens olika register (ctrl, status, channel, data, ivr). Ledning: gör först en typdefinition av funktionspekaren till en avbrottsfunktion. Visa dessutom tydliga definitioner av de enskilda bitarna som används i gränssnittet.

b) (4p) Konstruera en funktion **`int`** `read(`**`unsigned short`** `channel,` **`unsigned int`** `*value)` som:
1. startar en omvandling från kanal `channel`.
2. väntar tills omvandlingen är klar.
3. om fel uppstått, returnerar funktionen värdet -1, annars returnerar funktionen värdet 0 och värdet som avlästs via signalenheten returneras via parametern `value`. Typer och definitioner från uppgift a) ska användas.

För full poäng ska du alltså också visa hur preprocessordirektiv och typdeklarationer används för att skapa begriplig programkod samt tydligt ange i vilken typ av fil (.h, eller .c) varje lösningsdel ska placeras.

Lösningsförslag

Uppgift 1a:

```
#define    GPIO_E_BASE       0x40021000     /* MD407 port E */
#define    GPIO_E_MODER      ((volatile unsigned int *)    (GPIO_D_BASE))
#define    GPIO_E_OTYPER     ((volatile unsigned short *)  (GPIO_D_BASE+0x4))
#define    GPIO_E_PUPDR      ((volatile unsigned int *)    (GPIO_D_BASE+0xC))
#define    GPIO_E_IDR        ((volatile unsigned char *)   (GPIO_D_BASE+0x10))
#define    GPIO_E_ODR_LOW    ((volatile unsigned char *)   (GPIO_D_BASE+0x14))
#define    GPIO_E_ODR_HIGH   ((volatile unsigned char *)   (GPIO_D_BASE+0x15))
void init_app( void )
{
    *GPIO_E_MODER  = 0x55551150;
    *GPIO_E_OTYPER = 0;          /* använd push/pull */
    *GPIO_D_PUPDR  = 0x0000800A;  /* Flytande, aktivt höga -> ”pull down” */
}
```

Uppgift 1b:

```
#define SYSCFG_EXTICR2 ((volatile unsigned int *) 0x4001380C)
#define EXTI_IMR       ((volatile unsigned int *) 0x40013C00)
#define EXTI_FTSR      ((volatile unsigned int *) 0x40013C0C)
#define EXTI_RTSR      ((volatile unsigned int *) 0x40013C08)
#define EXTI_PR        ((volatile unsigned int *) 0x40013C14)
#define NVIC_ISER0     ((volatile unsigned int *) 0xE000E100)

void init_irq( void )
{  /* IRQ är ansluten till PE7, dvs. EXTI7 ska användas */
  *SYSCFG_EXTICR2 &= 0x0FFF;      /* PE7->EXTI7  */
  *SYSCFG_EXTICR2 |= 0x4000;
  *EXTI_IMR |= (1<<7);            /* aktivera avbrott EXTI7 */
  *EXTI_RTSR |= (1<<7);           /* aktivera trigger på positiv flank */
  *EXTI_FTSR &= ~(1<<7);          /* deaktivera trigger på negativ flank */
  *NVIC_ISER0 |= (1<<23);         /* aktivera avbrott i NVIC */
  *( (void (**) (void)) 0x2001C09C)  = sensor_irq;  /* sätt avbrottsvektor */
}
```

Uppgift 1c:

```
void sensor_irq( void )
 {
    if( *GPIO_E_IDR & 1 )
    {   /* inpassering */
        visitors++;
        *GPIO_E_ODR_LOW |= (1<<2);
    }
    if( *GPIO_E_IDR & 2 )
    {   /* utpassering */
        visitors--;
        *GPIO_E_ODR_LOW |= (1<<3);
    }
    *GPIO_E_ODR_LOW |= (1<<6);
    *EXTI_PR |= (1<<7);
}
```

Uppgift 1d:

```
#define    STK_CTRL   ((volatile unsigned int *)(0xE000E010))
#define    STK_LOAD   ((volatile unsigned int *)(0xE000E014))
#define    STK_VAL    ((volatile unsigned int *)(0xE000E018))
void delay10sec( void )
{
  for( int i = 0; i < 100; i++)
  { /* SystemCoreClock = 168000000 */
    *STK_CTRL = 0;
    *STK_LOAD = ( (16800000) -1 );
    *STK_VAL = 0;
    *STK_CTRL = 5;
    while( (*SysTickCtrl & 0x10000 )== 0 );
    *STK_CTRL = 0;
  }
}
```

Uppgift 1e:

```
static unsigned int visitors;
void main( void )
{
  visitors = 0;
  init_app();
  init_irq();
  while(1){
    *GPIO_E_ODR_HIGH = (unsigned char) visitors;
    delay_10sec();
  }
}
```

Uppgift 2a:

```
pack:
  LSL  R0,R0,#9     @ R0 <- year<<9;
  LSL  R1,R1,#5     @ R1 <- month<<5
  ORR  R0,R0,R1     @ R0 <- (year<<9) | (month<<5)
  ORR  R0,R0,R2     @ R0 <- (year<<9) | (month<<5) | day
  BX   LR           return ((year<<9) | (month<<5) | day)
```

Uppgift 2b:

```
  LDR  R0,i         @ R0 <- i
  ADD  R0,R0,#8     @ R0 <- i+8
  LSL  R0,R0,#5     @ R0 <- (i+8) * 32
  LDR  R1,j         @ R1 <- j
  ADD  R1,R1,#2     @ R1 <- j+2
  ADD  R0,R0,R1     @ R0 <- (i+8) * 32 + j+2
  LSL  R0,R0,#1     @ R0 <- ((i+8) * 32 + j+2) * sizeof(short)
  LDD  R1,=vs       @ R1 <- &vs
  ADD  R0,R0,R1     @ R0 <- &vs + ((i+8) * 32 + j+2) * sizeof(short)
```

Uppgift 3:

```
void rotateleft(char *f, int l, int steps) {
  int i
  char *current, *next, save;

  for (;steps > 0; steps--) {
    current  = next = f;
    next++;
    i = l;
    save = *current;  /* kommer att bli det sista elementet */
    while( i>1 )      /* för alla element utom det sista... */
    {
      *current++ = *next++;   /* skifta element vänster */
      i--;
    }
    *current = save;  /* sista element på plats */
  }
}
```

Uppgift 4a:

```
@ adressen till port D:s ut-dataregister till R5
    LDR     R5,=0x40020C14
@ adressen till port E:s in-dataregister till R6
    LDR     R6,=0x40021010

main:
    LDRB    R0,[R6]
    LDRB    R1,[R6,#1]
    ADD     R0,R0,R1
    CMP     R0,#255
    BLE     main_2
    MOV R0,#0xFF
main_2:
    STRB    R0,[R5]
    B       main
```

Uppgift 4b:

För att undvika att kompilatorn optimerar bort skrivningar/läsningar som den tror är överflödiga. Optimeraren vet inte att en viss minnesadress är en port där skrivningar/läsningar ger sidoeffekter utanför dess synfält

Uppgift 4c:

Då en .c-fil inkluderar en och samma .h-fil flera gånger - tex genom andra .h-filer.

Uppgift 5a:

```
typedef void (*interrupt_function) (void);
typedef struct{
  volatile unsigned char ctrl;
  volatile unsigned char status;
  short unused0;
  volatile unsigned short channel;
  short unused1;
  volatile unsigned int data;
  volatile interrupt_function ivr;
} SIGUNIT;
#define SIGUNITP ( ( SIGUNIT *) 0xF0000000 )
#define RS (1<<0)
#define EN (1<<3)
#define IA (1<<6)
#define IE (1<<7)
#define DN (1<<0)
#define ER (1<<2)
#define IRQ (1<<4)
```

Uppgift 5b:

```
int read ( unsigned short channel, unsigned int *result )
{
  SIGUNITP->ctrl = RS;  /* återställ signalenheten */
  SIGUNITP->channel = channel;
  SIGUNITP->ctrl |= EN; /* aktivera... */
  while( (SIGUNITP->status & DN )== 0) ; /* vänta tills klar..*/
  if(SIGUNITP->status & ER )
    return -1;  /* returnera kod FEL */
  *result = SIGUNITP->data;  /* returnera värde */
  return 0;  /* returnera kod OK */
}
```