



Tentamen med lösningsförslag

DAT017 Maskinorienterad programmering IT

DIT151 Maskinorienterad programmering GU

Måndag 14 mars 2022, kl. 14.00 - 18.00

Examinator

Erik Sintorn, tel. 772 52 29

Kontaktperson under tentamen:

Erik Sintorn, tel. 772 52 29

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 detta häfte.

Tabellverk eller miniräknare får ej användas.

Lösningar

Anslås senast dagen efter tentamen via kursens hemsida.

Granskning

Tid och plats anges på kursens hemsida.

Allmänt

Svar kan avges på svenska eller engelska.

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 programmets struktur.

Betygsättning

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

Maximal poäng är 50 och tentamenspoäng ger slutbetyg enligt: (DAT):

$20p \leq \text{betyg } 3 < 30p \leq \text{betyg } 4 < 40p \leq \text{betyg } 5$
respektive (DIT):

$20p \leq \text{betyg } G < 35p \leq VG$

Uppgift 1 (12p)

(a) Vi har deklARATIONERNA: `int j,k;` Visa en kodsekvens, i ARMv6 assemblerspråk, som evaluerar uttrycket $(k*9)-1$ till register R0. (3p)

(b) I funktionen `g` har vi registerallokeringen:

R4: lokal variabel `my_g`

R5: lokal variabel `my_rv`

I funktionen `g` görs följande funktionsanrop:

`my_rv = f(my_g);`

Visa hur detta då kodas i ARMv6 assemblerspråk. (3p)

(c) Vi har följande typdeklARATIONER:

```
struct coord;
typedef struct {
    unsigned int  a;
    unsigned char b;
    unsigned short c;
    struct coord *d;
} ST;
```

Visa hur följande deklARATION görs i ARM v6 assemblerspråk. (2p)

```
ST point;
```

(d) Visa en kodsekvens som evaluerar följande uttryck i register R0. (4p)

```
point.a = point.b + point.c;
```

Uppgift 2 (6p)

Följande funktion är given i C. Visa hur den kodas i ARMv6 assemblerspråk. För full poäng måste de kompilatorkonventioner vi använt i kursen följas, dessutom ska kommentarer i assemblerprogrammet utformas så att assemblerkoden kan kopplas till motsvarande konstruktion i C-programmet.

```
int contain( char *s, char c )
{
    while( *s )
    {
        if( *s == c )
            return 1;
        s++;
    }
    return 0;
}
```

Uppgift 3 (8p)

Konstruera en C-funktion som undersöker en parameter med avseende på antalet 0-ställda bitar.

Funktionen deklarerar:

```
short bitcheck( unsigned int *p, int * num );
```

- `p` är en pekare till det värde som ska undersökas
- `num` är en pekare till en plats dit antalet 0-ställda bitar hos parametern `p`, ska skrivas

Returvärdet för funktionen ska vara skilt från 0 om antalet nollor hos parametern är jämt delbart med 2, annars ska returvärdet vara 0.

Exempel: Följande kodsekvens:

```
num = 0xFFF;
even = bitcheck( &num, &result);
if( even )
    printf ( "Even (%d) number of zeroes", result );
else
    printf ( "Odd (%d) number of zeroes", result );
```

ger utskriften:

```
Even (20) number of zeroes
```

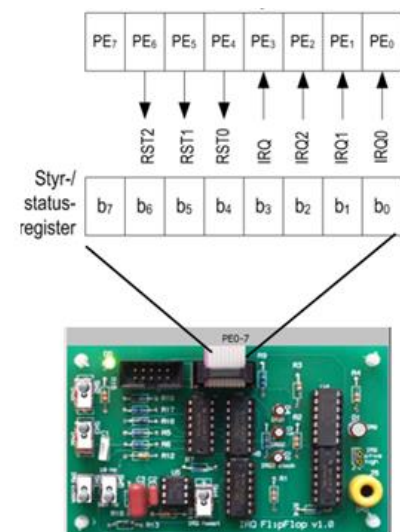
Uppgift 4 (6p)

Under laborationerna har du arbetat med ett laborationskort som genererar olika typer av avbrott, och kopplats till *MD407* enligt figuren till höger. Signalerna *RSTx* är utsignaler medan *IRQ*-signalerna är insignaler.

Den externa avbrottsmekanismen (EXTI) ska användas för att detektera ett godtyckligt avbrott (negativ flank) från laborationskortet. Samma avbrottsvektor ska användas för alla avbrott. Ingen hänsyn behöver tas till eventuella kontaktstudsar.

- Visa med en funktion `app_init` hur avbrottsmekanismerna initieras, dvs. IO-pinnar konfigureras, EXTI och NVIC initieras. (4p)
- Visa en komplett avbrottsrutin `irq_handler` som kvitterar (återställer) avbrott efter en knappnedtryckning så att systemet kan detektera nästa nedtryckning. Du får förutsätta att inga andra avbrott förekommer. (2p)

För full poäng krävs att din lösning är tydlig, fullständig och att du använt lämpliga makrodefinitioner för registeradresser.



Uppgift 5 (18p)

En elektroniskt styrd ventilationslucka ska konstrueras. Luckan har ett 7-bitars digitalt gränssnitt (se figur till höger) som beskrivs av följande:

Operatörsignaler:

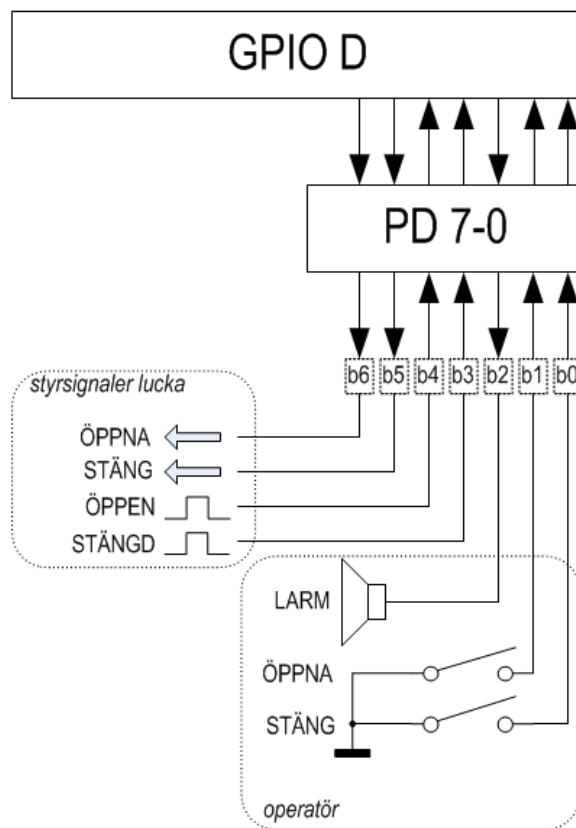
En funktion för att ge larm via en högalare aktiveras med en etta på bit 2.

Funktioner för att öppna och stänga luckan utgörs av två återjädrande strömställare anslutna via b1 och b0. Då strömställarna är öppna är dessa ingångar flytande och måste därför förses med programmerad *pull-up*. Om båda strömställarna aktiveras samtidigt ska larmsignal ges.

Signaler till/från ventilationslucka:

Indikatorer för helt öppen respektive helt stängd lucka är anslutna via b4 och b3. Signalerna är aktiva vid hög nivå. Om båda dessa signaler är 0 betyder det att luckan håller på att öppnas eller stängas. Om båda signaler är 1 innebär detta något fel och larmet ska då aktiveras.

Signalerna b6 och b5 aktiveras (sätts till 1) för att öppna respektive stänga luckan. Dessa signaler får inte aktiveras samtidigt. I mekaniken som reglerar luckan finns en viss tröghet så det kan ta maximalt 1 sekund att öppna och maximalt 2 sekunder att stänga luckan. Om det skulle ta längre tid ska larmsignal ges.



- För att kontrollera att det inte tar mer än 1 sekund att öppna eller stänga luckan ska en funktion för realtidsfördröjning användas. Denna ska konstrueras med hjälp av SYSTICK som en blockerande funktion `void delay_100ms(void)`; som fördröjer det anropande programmet i 1/10 sekund. (3p)
- Konstruera tre funktioner, `init`, `open` och `close` enligt följande specifikationer: (9p)


```
void init ( void );
```

 initiera GPIO D för användning med luckan. Oanvända pinnar i porten ska ställas som ingångar.

```
int open ( void );
```

 öppna luckan, vänta maximalt 1 sekunder i 100 ms intervall.
 Om dörren öppnas inom 1 sekund ska funktionen returnera 1, annars ska funktionen returnera 0.

```
int close ( void );
```

 stäng lucka, vänta maximalt 2 sekund i 100 ms intervall.
 Om dörren är stängd inom 2 sekunder ska funktionen returnera 1, annars ska funktionen returnera 0.
 Det är tillåtet att använda fördröjningsfunktionen `delay_100ms()` även om du inte löst den uppgiften.
- Konstruera ett huvudprogram som implementerar följande algoritm: (6p)

```
init;
Repetera:
  Om operatörsignal ÖPPNA och STÄNG:
    Aktivera larm
  annars:
    Om operatörsignal ÖPPNA
      Om open = 0
        Aktivera larm
      annars:
        Deaktivera styrsignaler
    Om operatörsignal STÄNG
      Om close = 0
        Aktivera larm
      annars:
        Deaktivera styrsignaler
```

För full poäng krävs att dina lösningar är tydliga, fullständiga och att du använt *lämpliga makrodefinitioner för registeradresser*.

Lösningsförslag

Uppgift 1 a

```
LDR R1,=k
LDR R0,[R1]    @ R0←k
LDR R1,[R1]    @ R1←k
LSL R0,R0,#3   @ R0←(k*8)
ADD R0,R0,R1   @ R0←(k*9)
SUB R0,R0,#1   @ R0←(k*9)-1
```

Uppgift 1 b

```
MOV R0,R4
BL  f
MOV R5,R0
```

Uppgift 1 c

```
.ALIGN
point:
.SPACE 4 @ unsigned int a
.SPACE 1 @ unsigned char b
.ALIGN 1
.SPACE 2 @ unsigned short c
.ALIGN
.SPACE 4 @ struct coord *d
```

Uppgift 1 d

```
LDR  R0,=point @ R0 = &point
LDRB R0,[4,R0] @ R0 = point.b
LDR  R1,=point @ R1 = &point
LDRH R1,[6,R1] @ R1 = point.c
ADD  R0,R0,R1  @ R0 = point.b+point.c
LDR  R1,=point @ R1 = &point
STR  R0,[R1]   @ point.a = point.b+point.c
```

Uppgift 2

```
@ int contain( char *s, char c )
@ R0: parameter s
@ R1: parameter c
@ R0: Returvärde
contain:
    B    contain1 @ while(*s)
contain2:
    CMP  R2,R1
    BEQ  contain3 @ (*s==c)?
    ADD  R0,R0,#1 @ s++
    B    contain1
contain3:
    MOV  R0,#1
    BX  LR @ return 1

contain1:
    LDRB R2,[R0]
    CMP  R2,#0
    BNE  contain2 @ (*s==0)?
    MOV  R0,#0
    BX  LR @ return 0
```

Uppgift 3:

```
short bitcheck( unsigned int *p, int *num)
{ /* Vi räknar ettorna, det är enklast... */
    *num = 0;
    while(*p)
    {
        if( *p & 1 ) (*num)++;
        *p >>= 1;
    }
    *num = (short) ((sizeof(int) * 8) - *num); /* Antal nollor */
    return !(*num & 1);
}
```

Uppgift 4 a:

```
#define NVIC_EXTI3_IRQ_BPOS (1<<9)
#define EXTI3_IRQ_BPOS      (1<<3)

#define GPIOE_MODER        ((volatile unsigned int *) 0x40021000)
#define GPIOE_OTYPER       ((volatile unsigned short *) 0x40021004)
#define GPIOE_PUPDR        ((volatile unsigned int *) 0x4002100C)
#define GPIOE_ODRLOW       ((volatile unsigned char *) 0x40021014)

#define SYSCFG_EXTICR1     ((volatile unsigned short *) 0x40013808)
#define EXTI_IMR           ((volatile unsigned int *) 0x40013C00)
#define EXTI_RTSCR         ((volatile unsigned int *) 0x40013C08)
#define EXTI_FTSR          ((volatile unsigned int *) 0x40013C0C)
#define EXTI_PR            ((volatile unsigned int *) 0x40013C14)

#define NVIC_ISER0         ((volatile unsigned int *) 0xE000E100)

void app_init( void )
{
    *GPIOE_MODER = 0x00001500;          /* Bit 4,5,6, digital ut, bit 3 (övriga) digital in */
    *GPIOE_PUPDR = 0;                  /* Nollställ alla bitar: "Input floating" */
    *GPIOE_OTYPER = 0;                 /* Nollställ alla bitar: "Output push/pull" */

    *GPIOE_ODRLOW = 0x70;              /* Återställ alla vippor */
    *GPIOE_ODRLOW = ~0x70;

    *SYSCFG_EXTICR1 &= 0xF000;
    *SYSCFG_EXTICR1 |= 0x4000;         /* PE3->EXTI3 */
    *EXTI_IMR |= EXTI3_IRQ_BPOS;
    *EXTI_RTSCR &= ~EXTI3_IRQ_BPOS;   /* EJ Avbrott på POSITIV flank */
    *EXTI_FTSR |= EXTI3_IRQ_BPOS;     /* Avbrott på NEGATIV flank */
    *NVIC_ISER0 = NVIC_EXTI3_IRQ_BPOS; /* Aktivera avbrott i NVIC */
}
```

Uppgift 4 b:

```
void irq_handler ( void )
{
    *GPIOE_ODRLOW = 0x70;              /* Återställ D-vippan */
    *GPIOE_ODRLOW = ~0x70;
    *EXTI_PR |= EXTI3_IRQ_BPOS;       /* Återställ EXTI3 "Pending Register" */
}
```

Uppgift 5 a:

```
#define STK_CTRL ((volatile unsigned int *)0xE000E010)
#define STK_LOAD ((volatile unsigned int *)0xE000E014)
#define STK_VAL ((volatile unsigned int *)0xE000E018)

void delay_100ms( void )
{
    /* SystemCoreClock = 168000000 */
    *STK_CTRL = 0;
    *STK_LOAD = ( (16800000) - 1 );
    *STK_VAL = 0;
    *STK_CTRL = 5;
    while( (*SysTickCtrl & 0x10000 )== 0 );
    *STK_CTRL = 0;
}
```

Uppgift 5 b:

```
#define GPIO_D_BASE    0x40020C00 /* MD407 port D */
#define GPIO_D_MODER  ((volatile unsigned int *) (GPIO_D_BASE))
#define GPIO_D_OTYPER ((volatile unsigned short *) (GPIO_D_BASE+0x4))
#define GPIO_D_PUPDR  ((volatile unsigned int *) (GPIO_D_BASE+0xC))
#define GPIO_D_IDR    ((volatile unsigned short *) (GPIO_D_BASE+0x10))
#define GPIO_D_ODR    ((volatile unsigned short *) (GPIO_D_BASE+0x14))

void init ( void )
{
    *GPIO_D_MODER = 0x00001410; /* b6,b5,b2 digital ut, övriga digital in */
    *GPIO_D_OTYPER = 0; /* utgångar är push/pull */
    *GPIO_D_PUPDR  = 5; /* b0, b1 är pull-up */
    *GPIO_D_ODR    = 0; /* Deaktivera styrsignaler */
}
```

```
int open ( void )
{
    *GPIO_D_ODR = (1<<6); /* b6=1 */
    for( int i = 0; i < 10; i++ )
    {
        if( (*GPIO_D_IDR & (1<<4)) )
            return 1; /* Dörr öppnad */
        delay_100ms();
    }
    return 0; /* Kunde inte öppna dörren */
}
```

```
int close( void )
{
    *GPIO_D_ODR = (1<<5); /* b5=1 */
    for( int i = 0; i < 20; i++ )
    {
        if(*GPIO_D_IDR & (1<<3))
            return 1; /* Dörr stängd */
        delay_100ms();
    }
    return 0; /* Kunde inte stänga dörren */
}
```

Uppgift 5 c:

```
int main( void )
{
    while(1)
    {
        if( (*GPIO_D_IDR & 3 ) == 0)
            *GPIO_D_ODR = (1<<2);
        else if ( (*GPIO_D_IDR & 2 ) == 0)
        {
            if( ! open() )
                *GPIO_D_ODR = (1<<2);
            else
                *GPIO_D_ODR = 0;
        }
        else if ( (*GPIO_D_IDR & 1 ) == 0)
        {
            if( ! close() )
                *GPIO_D_ODR = (1<<2);
            else
                *GPIO_D_ODR = 0;
        }
    }
}
```