



Tentamen med lösningsförslag

DAT017 (DAT016) Maskinorienterad programmering IT

EDA482 (EDA481) Maskinorienterad programmering D

EDA487 (EDA486) Maskinorienterad programmering Z

DIT151 Maskinorienterad programmering GU

LEU500 Maskinorienterad programmering D,E,ME (hing)

Onsdag 4 april 2018, kl. 14.00 - 18.00

Examinator

Roger Johansson, tel. 772 57 29

Kontaktperson under tentamen:

Roger Johansson, tel. 772 57 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 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):

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

respektive (DIT):

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

Uppgift 1 (10p)

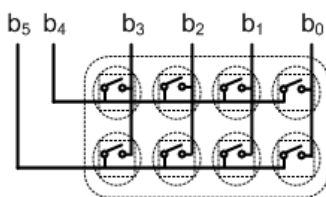
- a) (4p) Deklarationerna `int a, b, c;` är givna på "toppnivå". Visa hur tilldelningen `a |= (b & c);` kodas i ARM v6 assemblerspråk.
- b) (6p) Följande funktion `sum` beräknar en summa $n+(n-1)+(n-2)+\dots+1$ med hjälp av rekursion. Visa hur funktionen kan kodas i assemblerspråk.

```
int sum( int n )
{
    if ( n == 0 )
        return 0;

    return ( n + sum(n-1) );
}
```

Uppgift 2 (16p)

Ett tangentbord för inmatning av åtta olika tecken ska konstrueras. Åtta stycken återfjädrande omkopplare ansluts därför till port E hos MD407, på följande sätt:



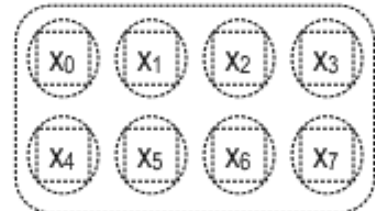
Bit 0-3 kopplas till ingångar, bit 4 och bit 5 kopplas till utgångar hos port E.

Nedtryckta tangenter kan detekteras genom att '1' skrivs till någon av bit 4 eller bit 5, därefter avläses bit 3-bit 0. Omkopplarna har försetts med dioder som likriktar strömriktningen. Flera tangenter kan därför tryckas ned samtidigt utan risk för kortslutning mellan b_5 och b_4 .

För att detta ska vara tillförlitligt måste ingångarna förses med "pull-down" samtidigt som utgångarna ska vara "push-pull". Metoden kallas *koincidensavsökning*.

- a) (4p) Visa en funktion `init_app`, som initierar port E för användning med tangentbordet. Bitarna b_6 och b_7 används inte och motsvarande portpinnar ska ställas som ingångar, bitarna b_8 - b_{15} ska konfigureras som utport ("push-pull").

- b) (4p) Tangenternas tillstånd kan representeras med en `char`, där en nedtryckt tangent motsvaras av att motsvarande bit är 1, annars är biten 0. Visa en tangentbordsfunktion `unsigned char get_kbd_stat(void)`, som returnerar tangenternas tillstånd.



- c) (3p) För att detektera nedtryckta tangenter korrekt måste eventuella *kontaktstudsar*, elimineras. För detta kan funktioner för realtidsfördröjning användas. Dessa ska implementeras med hjälp av SYSTICK. Använd SYSTICK för att konstruera en blockerande fördröjningsfunktion `void delay10ms(void);` som blockerar det anropande programmet i 10 ms.

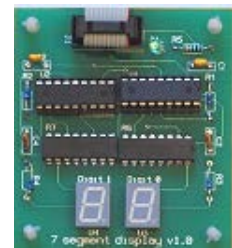


- d) (5p) En enkel utmatningsenhet för visning av två hexadecimal siffror, i form av en `char` ska användas som indikator och ansluts därför till port E, bit 8-15. Skriv en funktion `void disp_kbd(void)` som kontinuerligt läser av tangentbordet. Du får använda lösningar från tidigare uppgifter även om du inte gjort dem.

`disp_kbd` ska utformas enligt följande:

En giltig tangentbordsnedtryckning bestäms genom att två avläsningar, med 10 ms fördröjning emellan, ger samma resultat.

- Om ingen tangent är nedtryckt ska `0xFF` skrivas till visningsenheten.
- Om exakt en tangent är nedtryckt, ska dess tangentkod ($x_0=0, x_1=1, \dots, x_7=7$), skrivas till visningsenhetens högra indikator.
- Om två eller flera tangenter är nedtryckta ska antalet nedtryckta tangenter skrivas till den vänstra indikatorn.



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

RS: Reset, då denna bit sätts till 1 återställs bitarna ERR, IRQ, IE och EN till 0 av robotens gränssnitt. RS-biten måste därefter återställas till 0 för att robotarmen ska kunna aktiveras.

För att starta robotarmen krävs att:

1. Dataregistren initieras med måлкоординaterna.
2. Robotarmen aktiveras.
3. Om avbrott ska användas måste också avbrottsmekanismen aktiveras.
4. Då robotarmen nått målkordinaterna ska den deaktiveras.

För full poäng ska du i denna uppgift tydligt ange i vilken typ av fil (.h eller .c) varje lösningsdel ska placeras.

a) (4p) Visa en typdefinition i C ("typedef") ROBOT, av en sammansatt datatyp (**struct**) som beskriver gränssnittet till roboten. Typdefinitionen ska utformas så att varje register kan adresseras individuellt via en medlem i datastrukturen. Visa också lämpliga symboliska definitioner, i form av preprocessordirektiv, av bitarna i styrregistret.

b) (4p) Implementera följande funktioner utan att använda avbrottsmekanismerna:

void init (ROBOT *p)	initierar roboten, placerar robotarmen i läge 0,0.
void move (ROBOT *p, int x, int y)	utför förflyttning av robotarmen till (x, y)

Följande datastruktur kan användas för skapa en lista av punkter som beskriver en rörelse där armen utför en serie förflyttningar mellan olika koordinater och eventuellt utför en fördröjning efter varje förflyttning.

```
typedef struct point
{
    unsigned short x;
    unsigned short y;
    int delay
    struct point * next;
} POINT;
```

Koordinaterna x,y anger målkordinater för förflyttningen, delay anger en blockerande fördröjning (millisekunder) som ska göras då robotarmen nått målkordinaterna. Slutligen pekar next-komponenten på nästa punkt. Slutet av listan anges genom att next-komponenten är 0.

c) (4p) En funktion för att utföra en sammansatt rörelse hos robotarmen deklareras enligt:

```
void trip( ROBOT *p, POINT *pt );
```

Implementera funktionen.

Lösningsförslag

Uppgift 1a:

```
LDR    R3,b      @ R3← b
LDR    R2,c      @ R2← c
AND    R3,R3,R2  @ r3← (b&c)
LDR    R2,=a     @ R2← &a
LDR    R1,[R2]   @ R1← a
ORR    R1,R1,R3  @ R1← a|(b&c)
STR    R1,[R2]   @ a← a|(b&c)
```

Uppgift 1b:

```
sum:
    PUSH    {LR}
    CMP     R0,#0
    BEQ     .L2
    PUSH    {R0}
    SUB     R0,R0,#1
    BL      sum
    POP     {R1}
    ADD     R0,R0,R1
.L2:
    POP     {PC}
```

Uppgift 2a:

```
#define GPIO_E_MODER      ((volatile unsigned int *)0x40021000)
#define GPIO_E_OTYPER     ((volatile unsigned int *)0x40021004)
#define GPIO_E_PUPDR     ((volatile unsigned int *)0x4002100C)

void init_app( void )
{
    *GPIO_E_MODER = 0x55550500; /* bit 8-15, 5,4 sätts som utgångar, övriga sätts som ingångar */
    *GPIO_E_PUPDR = 0x000000AA; /* bit 3,2,1,0 PULL DOWN */
    *GPIO_E_OTYPER = 0;         /* utgångar, PUSH/PULL */
}
```

Uppgift 2b:

```
#define GPIO_E_ODR_LOW    ((volatile unsigned char *)0x40021014)
#define GPIO_E_IDR_LOW    ((volatile unsigned char *)0x40021010)
unsigned char get_kbd_stat(void)
{
    unsigned char row1, row2;
    *GPIO_E_ODR_LOW = 0x10;
    row1 = (*GPIO_E_IDR_LOW << 4);
    *GPIO_E_ODR_LOW = 0x20;
    row2 = *GPIO_E_IDR_LOW;
    return ( row1 | row2 );
}
```

Uppgift 2c:

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

void delay_10ms( void )
{
    /* SystemCoreClock = 168000000 */
    *STK_CTRL = 0;
    *STK_LOAD = ( (1680000) -1 );
    *STK_VAL = 0;
    *STK_CTRL = 5;
    while( (*SysTickCtrl & 0x10000 )== 0 );
    *STK_CTRL = 0;
}
```

Uppgift 2d:

```

#define GPIO_E_ODR_HIGH ((volatile unsigned char *)0x40021015)
void disp_kbd(void)
{
    unsigned char c;
    int i;
    while(1)
    {
        c = get_kbd_stat ();
        delay10ms();
        if ( c == get_kbd_stat() )
        {
            switch(c)
            {
                case 0: *GPIO_E_ODR_HIGH = 0xFF; break;
                case 1: *GPIO_E_ODR_HIGH = 0x07; break;
                case 2: *GPIO_E_ODR_HIGH = 0x06; break;
                case 4: *GPIO_E_ODR_HIGH = 0x05; break;
                case 8: *GPIO_E_ODR_HIGH = 0x04; break;
                case 0x10: *GPIO_E_ODR_HIGH = 0x03; break;
                case 0x20: *GPIO_E_ODR_HIGH = 0x02; break;
                case 0x40: *GPIO_E_ODR_HIGH = 0x01; break;
                case 0x80: *GPIO_E_ODR_HIGH = 0x00; break;
                default:
                    /* flera tangenter nedtryckta */
                    i = 0;
                    while( c )
                    {
                        if( c & 1 ) i++;
                        c >>= 1;
                    }
                    *GPIO_E_ODR_HIGH = (unsigned char) ( i << 4);
            }
        }
    }
}

```

Uppgift 3a:

```

#define NVIC_EXTI0_IRQ_BPOS (1<<6)
#define EXTI0_IRQ_BPOS (1<<0)

#define GPIOC_MODER ((volatile unsigned int *) 0x40020800)
#define GPIOC_OTYPER ((volatile unsigned short *) 0x40020804)
#define GPIOC_PUPDR ((volatile unsigned int *) 0x4002080C)
#define GPIOC_ODRLow ((volatile unsigned char *) 0x40020814)

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

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

void app_init( void )
{
    *GPIOC_MODER = 0x100; /* Bit 4 utgång, övriga ingångar */
    *GPIOC_PUPDR = 0; /* Ingång "floating" */
    *GPIOC_OTYPER = 0; /* Utgång "push/pull" */

    *GPIOC_ODRLow = 0x10; /* Återställ Flipflop */
    *GPIOC_ODRLow = ~0x10;

    *SYSCFG_EXTICR1 |= 0x0002; /* PC0->EXTI0 */
    *EXTI_IMR |= EXTI0_IRQ_BPOS;
    *EXTI_FTISR |= EXTI0_IRQ_BPOS;
    *EXTI_RTISR &= ~EXTI0_IRQ_BPOS;

    *NVIC_ISER0 |= NVIC_EXTI0_IRQ_BPOS;
}

```

Uppgift 3b:

```

void irq_handler ( void )
{
    *GPIOC_ODRLow = 0x10; /* Återställ Flipflop */
    *GPIOC_ODRLow = ~0x10;
    *EXTI_PR |= EXTI3_IRQ_BPOS; /* Kvittera EXTI-avbrott */
}

```

Uppgift 4:

```

int oddbit (int a, int *num)
{
    int numbits = 0;
    /* Enklast att räkna antalet ett-ställda bitar, resten,
       dvs. sizeof(int)*8 - antal ettställda bitar ger resultatet */
    while(a)
    {
        if( a & 1 )
            numbits ++;
        (( unsigned int) a) >>= 1;
    }
    *num = (sizeof(int)*8) - numbits;
    return *num & 1;
}

```

Uppgift 5a:

```

/* i .h-fil */
typedef struct sROBOT{
    volatile unsigned char ctrl;
    volatile unsigned char status;
    volatile unsigned short reserved;
    volatile unsigned short dataY;
    volatile unsigned short dataX;
    volatile unsigned short posY;
    volatile unsigned short posX;
}ROBOT;
#define IRQ      (1<<4)
#define ER       (1<<2)
#define IE       (1<<7)
#define IA       (1<<6)
#define EN       (1<<3)
#define RS       (1<<0)

```

Uppgift 5b:

```

/* i .c-fil */
void move(ROBOT *p, int x, int y )
{
    p->dataX = x;
    p->dataY = y;
    p->ctrl |= EN;

    while( ( p->posx != x ) || (p->posy != y ) ) ;
    p->ctrl &= ~EN;
}
void init(ROBOT *p)
{
    p->ctrl = RS;
    p->ctrl = 0;
    move( 0,0 );
}

```

Uppgift 5c:

```

void trip( ROBOT *p, POINT *pt )
{
    POINT *pt;
    init( p );
    while( pt->next != 0 )
    {
        move( p, pt->x, pt->y );
        delaylms( pt->delay );
        pt = pt->next;
    }
}

```