



Tentamen med lösningsförslag

DAT017 (DAT016) Maskinorienterad programmering IT

DIT151 Maskinorienterad programmering GU

EDA482 (EDA481) Maskinorienterad programmering D

EDA487 (EDA486) Maskinorienterad programmering Z

Måndag 18 mars, 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 häftet.

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 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: (EDA/DAT):

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

respektive (DIT):

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

Uppgift 1 (6p)

Vi har deklARATIONERNA

```
short a,b;
int x,i,j;
int ai[32][8]
```

på "toppnivå". Visa hur tilldelningarna:

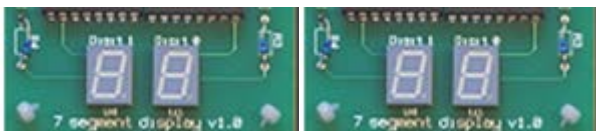
(a) $x = (a+b) * (a-b)$

(b) $ai[i][j] = x$


kodas i ARMv6 assemblerspråk.

Uppgift 2 (20p)

En tidtagaranläggning med tre oberoende tidtagarur (Clock1, Clock2, Clock3) ska konstrueras. Tid ska visas som 4 hexadecimala siffror på två moduler av typ "7-segment display" med upplösningen 10 ms. Alltså kan ett 16 bitars tal visas på de båda displayerna där den mest signifikanta delen ansluts till GPIO port D15-8 och den minst signifikanta delen ansluts till GPIO port D7-0. Det kan förutsättas att en tidmätning aldrig överskrider en **unsigned short** (dvs. ca 11 minuter).



Anläggningen styrs via en 8-polig DIL-omkopplare, kopplad till GPIO port E7-0 enligt följande:

	b ₇ -b ₆ : Väljer visning: 00: visar '0000' 01: Clock1 10: Clock2 11: Clock3	b ₅ -b ₃ : Clock "reset": b ₅ =1: Clock1 RESET b ₄ =1: Clock2 RESET b ₃ =1: Clock3 RESET	b ₂ -b ₀ : Clock start/stop b ₂ : Clock1 b ₁ : Clock2 b ₀ : Clock3 Startar vid positiv flank Stoppar vid negativ flank
--	--	--	--

Konstruera programpaketet för tidtagaranläggningen. Start/stopp-funktionen ska implementeras med hjälp av avbrottsmekanismer. För full poäng ska du dessutom använda lämpliga definitioner av typer och makron så som anvisats under kursen.

Ledningar:

Låt PE2, PE1 och PE0 generera avbrott både vid positiv och negativ flank.

Inför variabler som anger om en klocka är startad eller stoppad. Uppdatera endast startade klockor vid avbrott.

- Definiera de symboliska adresser med lämpliga typkonverteringar som krävs för uppgiften. Deklarera också lämpliga globala variabler. (3p)
- Visa en funktion `void portInit(void)` som initierar GPIO-portarna. (2p)
- Använd SYSTICK för att skapa en realtidsklocka som genererar avbrott med 10 ms intervall. Vid varje avbrott ska de klockor som är startade uppdateras. Systemets klockfrekvens är 168 MHz. Två funktioner ska implementeras (4p):
 - `void systickInit(void)` som gör alla nödvändiga initieringar och
 - `void systick_irq_handler(void)` som hanterar avbrotten från SYSTICK.
- Använd EXTI (0,1,2) för att implementera start/stopp- funktionerna. Tre avbrottsrutiner och en initieringsfunktion ska implementeras (6p):
 - `void extiX_irq_handler(void)` X=0,1,2 hanterar de olika avbrotten
 - `void extiInit(void)` gör alla nödvändiga initieringar för att använda PE-portpinnar för avbrott.
- Konstruera ett huvudprogram som: Initierar systemet med de specificerade initieringsfunktionerna och därefter, kontinuerligt, skriver ut det valda klockvärdet till 7-segmentsdisplayerna. En klocka som hålls i RESET ska stoppas och nollställas.(5p)

Uppgift 3 (6p)

Funktionen:

```
void int_concat (int *i1, const int * i2, unsigned int n, unsigned int pos);
```

konkatenerar (lägger till) ett fält med n heltal från i2 till position pos i fältet i1.

Implementera funktionen `int_concat` med användning av pekare, du får inte använda indexering. Din lösning får heller inte använda någon standardfunktion.

Uppgift 4 (6p)

I denna uppgift ska du förutsätta samma konventioner som i GCC för ARM.

a) Implementera följande funktionsdefinition i assemblerspråk för ARMv6. (4p)

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

Följande deklarationer av data och definitioner av funktioner är givna på toppnivå.

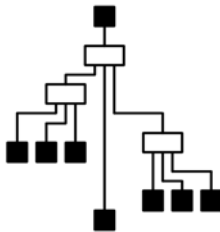
```
int c; char *a,b;
```

b) Visa hur följande funktionsanrop implementeras i assemblerspråk för ARM-v6. (2p)

```
c = match (a,b);
```

Uppgift 5 (12p)

a) Vad kallas den nätverkstopologi som illustreras på följande sätt? Ge också ett exempel på protokoll som använder denna topologi. (2p)



b) Ge två olika exempel på kommunikationsmedia för användning vid seriekommunikation. (2p)

c) Visa hur man deklarerar en funktionspekare till en funktion som returnerar en `int`, har en parameter `char`, som en typ `fp`. (2p)

Betrakta följande fragment av deklarationer och kod:

```
int i;
void yyy( int i )
{
    int a,x
    .....
}
```

d) Vilket värde har den globala variabeln `i` från början? (1p)

e) Hur kan den globala variabeln `i` refereras i funktionen `yyy` enbart med användning av C? (2p)

f) Hur kan den globala variabeln `i` refereras i funktionen `yyy` om alla medel är tillåtna? (3p)

Lösningsförslag

Uppgift 1:

```
a)
LDR  R0,=a      @ R0← &a
LDRH R0,[R0]   @ R0← a
SXTB R0,R0     @ R0← (int) a
LDR  R1,=b      @ R1← &b
LDRH R1,[R1]   @ R1← b
SXTB R1,R1     @ R1← (int) b
ADD  R2,R0,R1  @ R2← (int)a+(int)b
SUB  R3,R0,R1  @ R3← (int)a-(int)b
MUL  R2,R3     @ R2← (a+b)*(a-b)
LDR  R1,=x      @ R1← &x
STR  R2,[R1]   @ x ← ((int)a+(int)b) * ((int)a-(int)b)
```

```
b):
LDR  R0,x      @ R0← x
LDR  R1,i      @ R1← i
LSL  R1,R1,#3  @ R1← i*8
LDR  R2,j      @ R1← j
ADD  R1,R1,R2  @ R1← (i*8)+j
LSL  R1,R1,#2  @ R1← ((i*8)+j)*sizeof(int)
LDR  R2,=ai    @ R2← &ai
ADD  R1,R1,R2  @ R1← &ai+ index*sizeof(int)
STR  R0,[R1]   @ ai[i,j] ← x
```

Uppgift 2:

```
a)
#define GPIO_D      0x40020C00
#define GPIO_D_MODER ((volatile unsigned int *) (GPIO_D))
#define GPIO_D_OTYPER ((volatile unsigned short *) (GPIO_D+0x4))
#define GPIO_D_ODR  ((volatile unsigned short *) (GPIO_D+0x14))
#define GPIO_E      0x40021000
#define GPIO_E_MODER ((volatile unsigned int *) (GPIO_E))
#define GPIO_E_PUPDR ((volatile unsigned int *) (GPIO_E+0xC))
#define GPIO_E_IDR_LOW ((volatile unsigned char *) (GPIO_E+0x10))
#define SYSCFG_EXTICR1 ((volatile unsigned int *) 0x40013808)
#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)
#define EXTI0_IRQVEC  ((volatile unsigned int *) 0x2001C058)
#define EXTI1_IRQVEC  ((volatile unsigned int *) 0x2001C05C)
#define EXTI2_IRQVEC  ((volatile unsigned int *) 0x2001C060)
#define STK_CTRL      ((volatile unsigned int *) (0xE000E010))
#define STK_LOAD      ((volatile unsigned int *) (0xE000E014))
#define STK_VAL       ((volatile unsigned int *) (0xE000E018))
#define SYSTICK_IRQVEC ((volatile unsigned int *) 0x2001C03C)
alternativt:
#define SYSTICK_IRQVEC ((volatile unsigned int *) 0x0000003C)
```

```
static unsigned char Clock1running, Clock2running, Clock3running;
static unsigned short Clock1, Clock2, Clock3;
```

```
b)
void portInit ( void )
{
    *GPIO_E_MODER = 0;
    *GPIO_D_MODER = 0x55555555;
}
```

```
c)
void systick_irq_handler( void )
{
    if( Clock1running ) Clock1++;
    if( Clock2running ) Clock2++;
    if( Clock3running ) Clock3++;
}
```

```
void systickInit( void )
{
    /* SystemCoreClock = 168000000 */
    *SYSTICK_IRQVEC = systick_irq_handler;
    *STK_CTRL = 0;
    *STK_LOAD = ( 1680000-1 );
    *STK_VAL = 0;
```

```

*STK_CTRL = 7;
}
d)
void exti0_irq_handler( void )
{
    if( *((unsigned int *) EXTI_PR) & 1 )
    {
        *EXTI_PR |= 1;
        if( *GPIO_E_IDR_LOW & 1) Clock1running = 1; else Clock1running = 0;
    }
}

void exti1_irq_handler( void )
{
    if( *((unsigned int *) EXTI_PR) & 2 )
    {
        *EXTI_PR |= 2;
        if( *GPIO_E_IDR_LOW & 2) Clock2running = 1; else Clock2running = 0;
    }
}

void exti2_irq_handler( void )
{
    if( *((unsigned int *) EXTI_PR) & 4 )
    {
        *EXTI_PR |= 4;
        if( *GPIO_E_IDR_LOW & 4) Clock3running = 1; else Clock3running = 0;
    }
}

void extiInit ( void )
{
    *SYSCFG_EXTICR1 |= 0x0444;      /* PE0->EXTI0, PE1->EXTI1,PE2->EXTI2 */

    /* Configure the mask bit of the interrupt line (EXTI_IMR) */
    *EXTI_IMR |= (1<<0)|(1<<1)|(1<<2);

    /* Configure the Trigger selection bit of the interrupt line (EXTI_RTSR and EXTI_FTSR) */
    *EXTI_RTSR |= (1<<0)|(1<<1)|(1<<2);      /* enable trigger on falling edge */
    *EXTI_FTSR |= (1<<0)|(1<<1)|(1<<2);      /* enable trigger on rising edge */
    *NVIC_ISEER0 |= (1<<6)|(1<<7)|(1<<8);

    *EXTI0_IRQVEC = exti0_irq_handler;
    *EXTI1_IRQVEC = exti1_irq_handler;
    *EXTI2_IRQVEC = exti2_irq_handler;
}

e)
void main(void)
{
    portInit();
    extiInit();
    systickInit();
    while( 1 )
    {
        switch( *GPIO_E_IDR_LOW >> 6 )
        {
            case 0: *GPIO_D_ODR = 0; break;
            case 1:
                if(*GPIO_E_IDR_LOW & (1<<3) ) /* RESET */
                {
                    Clock1running = 0;
                    Clock1 = 0;
                }
                *GPIO_D_ODR = Clock1 ; break;
            case 2:
                if(*GPIO_E_IDR_LOW & (1<<4) ) /* RESET */
                {
                    Clock2running = 0;
                    Clock2 = 0;
                }
                *GPIO_D_ODR = Clock2; break;
            case 3:
                if(*GPIO_E_IDR_LOW & (1<<5) ) /* RESET */
                {
                    Clock3running = 0;
                    Clock3 = 0;
                }
                *GPIO_D_ODR = Clock3; break;
        }
    }
}

```

Uppgift 3:

```
void int_concat (int *i1, const int *i2, unsigned int n, unsigned int pos)
{
    int *insert = i1 + pos;

    while ( n-- )
    {
        *insert++ = *i2++;
    }
}
```

Uppgift 4:

a)

R0: parameter 's'
R1: parameter 'm'
R2: preset return value
R3: temp register

```
match:
    MOV    R2,#0        @ preset return value 0
match1:
    LDRB   R3,[R0]      @ R3 <- *s
    CMP    R3,#0        @ (*s == 0) ?
    BEQ    match_exit
    CMP    R3,R1        @ (*s != m) ?
    BNE    match2
    MOV    R2,#1        @ preset return value 1
    B      match_exit
match2:
    ADD    R0,R0,#1    @ s++
    B      match1
match_exit:
    MOV    R0,R2        @ return 0/1
    BX    LR
b)
    LDR    R0,a
    LDR    R1,=b
    LDRB   R1,[R1]
    BL    match
    LDR    R1,=c
    STR    R0,[R1]
```

Uppgift 5:

a)
Svar: "Träd", USB (Universal Serial Bus)

b)
Svar: Koppartråd, koaxialkabel och optofiber.

c)
Svar: typedef int (*fp) (char);

d)
Enligt standarden ska oinitierade variabler nollställas av run-time systemet, men för fristående miljöer (som i inbyggda system) kan man inte alltid förutsätta detta. Dvs. 0 eller obestämt.

e)
Det går inte.

f)
asm{ " LDR R0,i" }