



Tentamen med lösningsförslag

DAT390 (LEU500) Maskinorienterad programmering D,E,ME (hing)

Lördag 8 juni 2019, kl. 8.30 - 12.30

Examinator

Pedro Trancoso, tel. 772 63 19

Kontaktperson under tentamen:

Pedro Trancoso, tel. 772 63 19

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

Frågor kan även besvaras på 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 programmens struktur.

Betygsättning


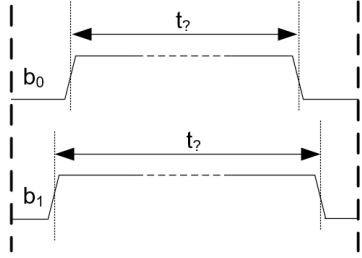
För godkänt slutbetyg på kursen fordras att både tentamen och laborationer är godkända.

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

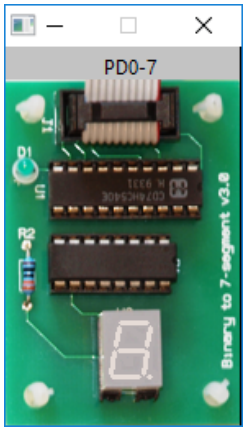
$20p \leq \text{betyg } 3 < 30p \leq \text{betyg } 4 < 40p \leq \text{betyg } 5$

Uppgift 1 (18p)

En enkel modul för pulstidsmätning ska konstrueras. Två signalkällor ska kunna mätas samtidigt och tiden mäts med upplösningen 10 ms. Modulen styrs via en 8-polig DIL-omkopplare, kopplad till GPIO port E7-0.

	<p>b0: Signalkälla 1 b1: Signalkälla 2</p> <p>En mätning startas vid positiv flank och avbryts vid negativ flank</p> <p>b7: Visning 0: visar resultatet av senaste pulstidsmätning från signalkälla 1. 1: visar resultatet av senaste pulstidsmätning från signalkälla 2.</p>	
---	---	---

Dessutom ansluts en visningsenhet enligt följande:

	<p>Som utmatningsenhet används en enkel sifferindikator för 7-segmentsvisning. Den ansluts till port D0-7 hos MD407.</p> <p>Tabellen till höger beskriver hur en binär siffra översätts till sju-segmentskod innan den matas ut till sifferindikatorn</p>	<table border="1"> <thead> <tr> <th rowspan="2">Siffra</th> <th colspan="2">Sju-segmentskod</th> </tr> <tr> <th>Binär kod</th> <th>Hexadecimal form</th> </tr> </thead> <tbody> <tr><td>0</td><td>0000</td><td>3F</td></tr> <tr><td>1</td><td>0001</td><td>06</td></tr> <tr><td>2</td><td>0010</td><td>5B</td></tr> <tr><td>3</td><td>0011</td><td>4F</td></tr> <tr><td>4</td><td>0100</td><td>66</td></tr> <tr><td>5</td><td>0101</td><td>6D</td></tr> <tr><td>6</td><td>0110</td><td>7D</td></tr> <tr><td>7</td><td>0111</td><td>07</td></tr> <tr><td>8</td><td>1000</td><td>7F</td></tr> <tr><td>9</td><td>1001</td><td>67</td></tr> <tr><td>A</td><td>1010</td><td>77</td></tr> <tr><td>B</td><td>1011</td><td>7C</td></tr> <tr><td>C</td><td>1100</td><td>39</td></tr> <tr><td>D</td><td>1101</td><td>5E</td></tr> <tr><td>E</td><td>1110</td><td>79</td></tr> <tr><td>F</td><td>1111</td><td>71</td></tr> </tbody> </table>	Siffra	Sju-segmentskod		Binär kod	Hexadecimal form	0	0000	3F	1	0001	06	2	0010	5B	3	0011	4F	4	0100	66	5	0101	6D	6	0110	7D	7	0111	07	8	1000	7F	9	1001	67	A	1010	77	B	1011	7C	C	1100	39	D	1101	5E	E	1110	79	F	1111	71
Siffra	Sju-segmentskod																																																						
	Binär kod	Hexadecimal form																																																					
0	0000	3F																																																					
1	0001	06																																																					
2	0010	5B																																																					
3	0011	4F																																																					
4	0100	66																																																					
5	0101	6D																																																					
6	0110	7D																																																					
7	0111	07																																																					
8	1000	7F																																																					
9	1001	67																																																					
A	1010	77																																																					
B	1011	7C																																																					
C	1100	39																																																					
D	1101	5E																																																					
E	1110	79																																																					
F	1111	71																																																					

Konstruera programpaketet för modulen. För full poäng ska du använda lämpliga definitioner av typer och makron så som anvisats under kursen.

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

- Definiera de symboliska adresser med lämpliga typkonverteringar som krävs för uppgiften. Deklarera också lämpliga globala variabler. och visa en funktion `void portInit(void)` som initierar GPIO-portarna. (4p)
- 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 (3p):
 - `void systickInit(void)` som gör alla nödvändiga initieringar och
 - `void systick_irq_handler(void)` som hanterar avbrotten från SYSTICK.
- Använd EXTI (0,1) för att implementera detektera signalkällornas nivåer. Två avbrottsrutiner och en initieringsfunktion ska implementeras (4p):
 - `void extiX_irq_handler(void) X=0,1` hanterar de olika avbrotten
 - `void extiInit(void)` gör alla nödvändiga initieringar för att använda PE-portpinnar för avbrott.
- Visa en funktion `void out7seg(unsigned char c)` som skriver en hexadecimal siffra till sifferindikatorn. (3p)
- Konstruera ett huvudprogram som: Initierar systemet med de specificerade initieringsfunktionerna och därefter, kontinuerligt, skriver ut den valda, senast uppmätta, pulslängden till 7-segmentsdisplayerna som multipel av 10ms, dvs. 0..10ms →1, 11-20ms →2, osv. För pulslängder större än 9 ska E skrivas till sifferindikatorn. Om mätning pågår ska sifferindikatorn släckas. (4p)

Uppgift 2 (7p)

Vi har deklARATIONERNA

```
short j;
short vc[30];
int i, va[100], vb[20];
```

på "toppnivå". Visa hur tilldelningen:

```
vc[j] = va[ vb[i] ];
```

kodas i ARMv6 assemblerspråk.

Uppgift 3 (6p)

Följande deklARATIONER är givna:

```
int go( int a, int b);
int next( void );
```

Visa hur funktionen `g` kan kodas i ARMv6 assemblerspråk.

```
int g( int x, int y)
{
    int a = next();
    if( a-1 )
        return go(y,x);
    return go(x,y);
}
```

Uppgift 4 (8p)

Deklarationen:

```
typedef struct post
{
    int a,b,c;
    char *root;
} POST, *PPOST;
```

är given. Vi behöver en funktion

```
PPOST append_post( PPOST p1, unsigned int n1, PPOST p2, unsigned int n2)
```

som lägger samman två fält med poster. Vi har alltså två fält av typen `POST`: `p1`, med `n1` poster, och `p2`, med `n2` poster, som vi vill sammanfoga till en ny uppsättning typ `POST` som innehåller alla element av `p1` följt av alla element i `p2`. Funktionen ska använda `malloc` för att skapa utrymme för att lagra resultatet och returnera en pekare till detta. Din lösning får inte använda någon annan standardfunktion än `malloc`.

Uppgift 5 (6p)

Skriv en funktion `void encrypt(char *str, unsigned char b)` som tar som parameter en sträng och "krypterar" strängen (dvs. ändrar den strängen) på ett sådant sätt att en bit i varje tecken "flippas" (inverteras). Biten som ska ändras ges av parameter `b` som därför kan vara 0-7.

Uppgift 6 (5p)

Besvara följande frågor (1p för varje):

- Vi har deklARATIONEN: `void *x`; Vilket värde genererar operatorn `sizeof(x)` hos en 32-bitars maskin?
- Vi har: `union abc { int a; char b; } x; x.a = 23; x.b = 0`; Vilket värde har nu `x.a`?
- Vi har: `enum abc { one=1, two=5, three }; int a = three+10`; Vilket värde har nu `a`?
- Vi har:


```
char str[] = "longer string"; int *y; char c;
y = (int*)str; y++;
c = *((char*)y);
```

 Vilket värde har nu `c`?

e) Betrakta följande sekvens av preprocessor direktiv och kod:

```
#define FOOY
#ifdef FOOX
int foo(int a, int b) { return a-b; }
#endif
#ifdef FOOY
int foo(int a, int b) { return a+b; }
#endif

int main(void) {
    int x;
    x = foo( 20, 5);
}
```

Vilket värde har `x` efter tilldelningen i `main`?

Lösningsförslag

Uppgift 1:

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_LOW ((volatile unsigned char *) (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_RTSCR      ((volatile unsigned int *) 0x40013C08)
#define EXTI_PR        ((volatile unsigned int *) 0x40013C14)
#define NVIC_ISERO      ((volatile unsigned int *) 0xE000E100)
#define EXTI0_IRQVEC    ((volatile unsigned int *) 0x2001C058)
#define EXTI1_IRQVEC    ((volatile unsigned int *) 0x2001C05C)
#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)
```

```
static volatile unsigned char Pulse0measure, Pulse1measure;
static volatile unsigned int Pulse0time, Pulse1time;
```

```
void portInit ( void )
```

```
{
    *GPIO_E_MODER = 0;
    *GPIO_E_PUPDR = 0;
    *GPIO_D_MODER = 0x00005555;
    *GPIO_D_OTYPER = 0x00000000;
}
```

c)

```
void systick_irq_handler( void )
```

```
{
    if( Pulse0measure ) Pulse0time++;
    if( Pulse1measure ) Pulse1time++;
}
```

```
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( (*EXTI_PR) & 1 )
    {
        *EXTI_PR |= 1;
        if( *GPIO_E_IDR_LOW & 1 )
        { /* Positive edge */
            Pulse0measure = 1;
            Pulse0time = 0;
        }else{ /* Negative edge */
            Pulse0measure = 0;
        }
    }
}
```

```
void extil_irq_handler( void )
```

```
{
    if( (*EXTI_PR) & 2 )
    {
        *EXTI_PR |= 2;
        if( *GPIO_E_IDR_LOW & 2 )
        { /* Positive edge */
            Pulse1measure = 1;
            Pulse1time = 0;
        }
    }
}
```

```

    }else{ /* Negative edge */
        Pulse1measure = 0;
    }
}

void extiInit ( void )
{
    *SYSCFG_EXTICR1 &= 0xFF00; /* Only EXTI0, EXTI1 */
    *SYSCFG_EXTICR1 |= 0x0044; /* PE0->EXTI0, PE1->EXTI1 */

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

    /* Configure the Trigger selection bit of the interrupt line (EXTI_RTSR and EXTI_FTSR) */
    *EXTI_RTSR |= (1<<0)|(1<<1); /* enable trigger on falling edge */
    *EXTI_FTSR |= (1<<0)|(1<<1); /* enable trigger on rising edge */
    *NVIC_ISER0 |= (1<<6)|(1<<7);

    *EXTI0_IRQVEC = exti0_irq_handler;
    *EXTI1_IRQVEC = exti1_irq_handler;
}

```

e)

```

void out7seg( unsigned char c )
{
    static unsigned char segCode[]={
        0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,
        0x7F,0x67,0x77,0x7C,0x39,0x5E,0x79,0x71 };

    if( c > 15){
        *GPIO_D_ODR_LOW = 0;
        return;
    }
    *GPIO_D_ODR_LOW = segCode[c];
}

```

f)

```

void main(void)
{
    char c;
    Pulse0measure = 0;
    Pulse1measure = 0;
    Pulse0time = 0;
    Pulse1time = 0;
    portInit();
    extiInit();
    systickInit();
    while( 1 )
    {
        if( *GPIO_E_IDR_LOW & 0x80)
        { /* Show signal source 2 */
            if( Pulse1measure )
                c = 16; /* Turn off... */
            else
            {
                if(Pulse1time >9 )
                    c = 0xE;
                else
                    c = Pulse1time;
            }
        }
        else{ /* Show signal source 1 */
            if( Pulse0measure )
                c = 16; /* Turn off... */
            else
            {
                if(Pulse0time >9 )
                    c = 0xE;
                else
                    c = Pulse0time;
            }
        }
        out7seg( c );
    }
}

```

Uppgift 2:

```

LDR R0,=i @ R0← &i
LDR R0,[R0] @ R0← i
LSL R0,R0,#2 @ R0← (i*sizeof(int) )
LDR R1,=vb @ R1← &vb
ADD R0,R0,R1 @ R0← &vb + (i*sizeof(int) )
LDR R0,[R0] @ R0← vb[i]
LDR R1,=va @ R0← &va
LSL R0,R0,#2 @ R0← (vb[i]*sizeof(int) )
ADD R0,R1,R0 @ R0← &va + (vb[i]*sizeof(int) )
LDR R0,[R0] @ R0← va[ vb[i] ]

LDR R1,=j @ R1← &j
LDRH R1,[R1] @ R1← j
LSL R1,R1,#1 @ R0← (j*sizeof(short) )
LDR R2,=vc @ R2← &vc
ADD R1,R1,R2 @ R1← &vc + (j*sizeof(short) )
STRH R0,[R1] @ vc[j]← va[ vb[i] ]

```

Uppgift 3:

```

@ Registers:
@ R4 spill x
@ R5 spill y
g: PUSH {LR,R4,R5}
MOV R4,R0
MOV R5,R1
BL next
SUB R0,R0,#1
CMP R0,#0
BNE .g1
MOV R0,R4
MOV R1,R5
B .g2
.g1:
MOV R0,R5
MOV R1,R4
.g2:
BL go
POP {PC,R4,R5}

```

Uppgift 4:

```

PPOST append_post( PPOST p1, unsigned int n1, PPOST p2, unsigned int n2)
{
    PPOST target, new_post;
    new_post = (PPOST) malloc( (n1+n2)*sizeof(POST) );
    if( new_post == NULL )
        return NULL;
    target = new_post;
    while( n1-- )
        *target++ = *p1++;
    while( n2-- )
        *target++ = *p2++;
    return new_post;
}

```

Uppgift 5:

```

void encrypt( char *str, unsigned char b ) {
    char *x = str;
    unsigned char mask = 0x1 << b;
    while( *x ) {
        if( *x & mask )
            *x = *x & (~mask);
        else
            *x = *x | mask;
    }
}

```

Uppgift 6:

- a) 4
- b) 0
- c) 16
- d) 'e'
- e) 25