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

Fredag 23 augusti 2019, kl. 8.30 - 12.30

Examinatorer
Roger Johansson, tel. 772 57 29
Pedro Trancoso, tel. 772 63 19

Kontaktpersoner under tentamen:
Roger Johansson, tel. 772 57 29
Pedro Trancoso, tel. 772 63 19 (Lindholmen)

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 (5p)

a) De globala variablerna `j` och `vecc` är definierade enligt: `int j; char vecc[80];` Visa en kodsekvens, i ARMv6 assemblerspråk, som evaluerar uttrycket `vecc[j]` till register R0. (2p)

b) Utgå från att följande deklarationer är gjorda på global nivå.

```
int     a,b,c;
```

Visa en kodsekvens, i ARMv6 assemblerspråk, som evaluerar följande uttrycks värde till register R0. (3p)

```
(a + b * c ) <<1
```

Uppgift 2 (5p)

Följande funktion `int f( char * str )` har kodats i ARMv6 assemblerspråk med tillämpade kompilatorkonventioner enligt:

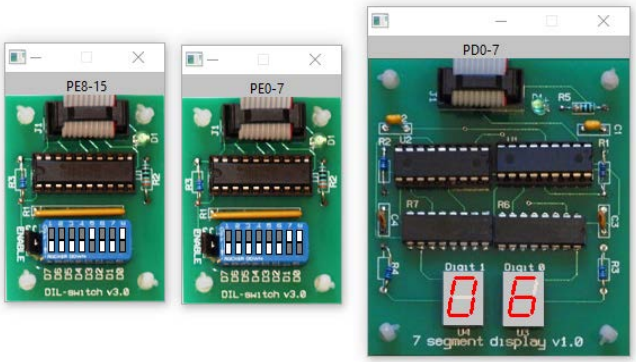
```
f:      MOV     R1,#0
.L0:
        LDRB    R2,[R0]
        CMP     R2,#0
        BEQ     .L1
        ADD     R0,R0,#1
        ADD     R1,R1,#1
        B       .L0
.L1:
        MOV     R0,R1
        BX      LR
```

Analysera assemblerkoden, och koda en motsvarande funktion i C.

Uppgift 3 (6p)

Två DIL-strömbrytare och en Double hexadecimal display kopplas enligt:

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



Skriv ett program i ARMv6 assemblerspråk som:

- Initierar portar D och E.
- Kontinuerlig läser de inställda värdena från DIL-strömbrytarna, subtraherar (DIL1) från (DIL2) och skriver resultatet till displayen.

Uppgift 4 (6p)

Följande deklaration är given:

```
short f( short a, short b);
```

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

```
int g( short x, short y)
{
   short a = f(x,y);
   if( a == x )
      return y;
   return x;
}
```

Uppgift 5 (6p)

Följande deklaration är given:

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

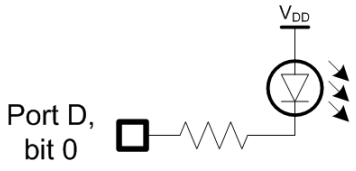
Funktionen: `int compare_post( PPOST pp, POST p);` undersöker om en identisk kopia av posten `p` finns i ett fält ("array") av poster `pp`, som avslutas med en 0-pekare. Implementera funktionen `compare_post` med användning av pekare, du får inte använda indexering. Din lösning får heller inte använda någon standardfunktion.

Uppgift 6 (8p)

Skriv en funktion `void encrypt( char *str )` som tar som parameter en sträng och "krypterar" strängen (dvs. ändrar den strängen) på ett sådant sätt att den första biten (bit 0)i det första tecknet "flippas" (inverteras), för det andra tecknet "flippas" den andra biten etc. Krypteringen är "cirkulär" runt åtta bitar så för det åttonde tecknet är bit 7 vänd. För det nionde tecknen är då den första biten vänd och så vidare ...

Uppgift 7 (8p)

En ljusdiod har anslutits till en pinne hos port D enligt figuren till höger.



Skriv ett program, i C, som kontinuerligt tänder dioden under en halv sekund och därefter släcker den under en halv sekund. Din lösning ska fördelas på följande deluppgifter:

a) Visa hur SysTick kan användas för att skapa en (blockerande) fördröjning om 250 mikrosekunder (en fjärdedels millisekund), med funktionen `void delay_250mys( void);`. Systemets klockfrekvens är 168 MHz. (4p)
b) Skriv en funktion `void init_app( void );` som sätter upp port D, bit 0 som utsignal, ”push-pull”. Övriga inställningar i porten ska behållas, dvs. får inte ändras av initieringen. (2p).
c) Skriv huvudprogrammet som får dioden att blinka. Du kan använda `delay_250mys()` och `init_app()` även om du inte besvarat a) och/eller b). (2p)

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

Uppgift 8 (6p)

Förbered en enkel applikation som använder PD15 hos MD407 som avbrottsingång. Dvs, skriv, i C, en sekvens som:

- Kopplar PD15 till EXTI15
- Konfigurerar EXTI15 för att generera avbrott på negativ flank
- Konfigurerar NVIC.

Ange också den vektor som ska initieras för avbrottet.

Du kan förutsätta att alla moduler startats och behöver inte ta hänsyn till klockor (RCC). Observera dock att andra eventuella EXTI-konfigurationer inte får ändras av din programsekvens. För full poäng ska du använda lämpliga definitioner av typer och makron så som anvisats under kursen.

Lösningsförslag

Uppgift 1:

a)
```
    LDR     R0,j
    LDR     R1,=vecc
    LDRB    R0,[R0,R1]
```
b)
```
    LDR     R0,a
    LDR     R1,b
    LDR     R2,c
    MUL     R1,R1,R2
    ADD     R0,R0,R1
    LSL     R0,R0,#1
```

Uppgift 2:
```
int f( char * str )
{  int rval = 0;
   while( *str++ ) rval++;
   return rval;
}
```

Uppgift 3:
```
start:
@   initiera port D0-D7 som utport
    LDR     R0,=0x00005555
    LDR     R1,=0x40020C00
    STR     R0,[R1]
@   initiera port E0-E15 som inport
    LDR     R0,=0
    LDR     R1,=0x40021000
    STR     R0,[R1]
@ 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    R1,[R6]
    LDRB    R0,[R6,#1]
    SUB     R0,R0,R1
    STRB    R0,[R5]
    B       main
```

Uppgift 4:
```
@ Registers:
@  R4 spill x
@  R5 spill y
g:   PUSH  {R4,R5,LR}
     MOV   R4,R0
     MOV   R5,R1
     BL    f
     CMP   R0,R4
     BNE   .L0
     MOV   R0,R5
     B     .L1
.L0: MOV   R0,R4
.L1: POP   {R4,R5,PC}
```

Uppgift 5:
```
int compare_post( PPOST pp, POST p)
{
    while ( pp )
    {
        if ((pp->a == p.a )&&(pp->b == p.b )&&(pp->c == p.c ))
        (eller; if (*pp == p) )
            return 1;
        pp++;
    }
    return 0;
}
```

Uppgift 6:

```
void encrypt( char *str ) {
   char *x = str;
   unsigned char mask = 0x1;
   while( *x ) {
      if( *x & mask )
         *x = *x & (~mask);
      else
          *x = *x | mask;
      mask = mask << 1;
      if(mask == 0)
          mask = 0x1;
      x++;
   }
}
```

Uppgift 7:

a)

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

void delay_250mys( void )
{
    /* SystemCoreClock = 168000000 */
    *STK_CTRL = 0;
    *STK_LOAD = ( (168000/4) -1 );
    *STK_VAL = 0;
    *STK_CTRL = 5;
    while( (*STK_CTRL & 0x10000 )== 0 );
    *STK_CTRL = 0;
}
```

b)

```
#define    PORT_D_MODER   ((volatile unsigned int *)(0x40020C00))
#define    PORT_D_OTYPER  ((volatile unsigned short int *)(0x40020C04))
#define    PORT_D_ODR_LOW ((volatile unsigned short int *)(0x40020C14))
void init_app( void )
{
    /* PORT D */
    *PORT_D_MODER      &= ~3;  /* återställ bit 0 mode */
    *PORT_D_MODER      |= 1;   /* bit 0 sätts som utgång */
    *PORT_D_OTYPER     &= ~1;  /* återställ bit 0 typ, är nu push/pull */
}
```

c)

```
void main(void)
{
    int i;
    init_app();
    while(1)
    {
        *PORT_D_ODR_LOW = 0;
        for(i=0;i<2000;i++) delay_250mys();
        *PORT_D_ODR_LOW = 0xFF;
        for(i=0;i<2000;i++) delay_250mys();
    }
}
```

Uppgift 8:

```
#define    SYSCFG_EXTICR4 ((volatile unsigned int *)(0x40013814))
#define    EXTI_IMR       ((volatile unsigned int *)(0x40013C00))
#define    EXTI_RTSR      ((volatile unsigned int *)(0x40013C08))
#define    EXTI_FTSR      ((volatile unsigned int *)(0x40013C0C))
#define    NVIC_ISER1     ((volatile unsigned int *)(0xE000E104))

*SYSCFG_EXTICR4    &= 0x0FFF;     /* nollställ bitfält EXTI15 */
*SYSCFG_EXTICR4    |= 0x3000;     /* PD15->EXTI15  */
*EXTI_IMR          |= (1<<15);    /* aktivera avbrott EXTI15 */
*EXTI_FTSR         |= (1<<15);    /* aktivera trigger på negativ flank */
*EXTI_RTSR         &= ~(1<<15);   /* deaktivera trigger på positiv flank */
*NVIC_ISER1        |= (1<<8);     /* aktivera avbrott i NVIC */
```

Vektor nummer 40 (offset 0xE0)