Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA

Tentamen med lösningsförslag

EDA482 Maskinorienterad programmering D
EDA487 Maskinorienterad programmering Z

Fredag 19 augusti 2022, kl. 8.30 - 12.30

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 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 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:
$20p \leq$ betyg $3 < 30p \leq$ betyg $4 < 40p \leq$ betyg 5

Uppgift 1 (16p)

(a) Vi har de globala deklarationerna:

```
long a; short *b;
```

Visa först hur variabeldeklarationerna kodas i assembler i ARMv6 assemblerspråk (2p).

För funktionen `f` gäller nu deklarationen:

```
short f( short, short *);
```

visa också hur då följande funktionsanrop kodas i ARMv6 assemblerspråk:

```
a = (long) f(*b,b);
```

(4p)

(b) Vi har de globala deklarationerna:

```
int i; char vc[20];
```

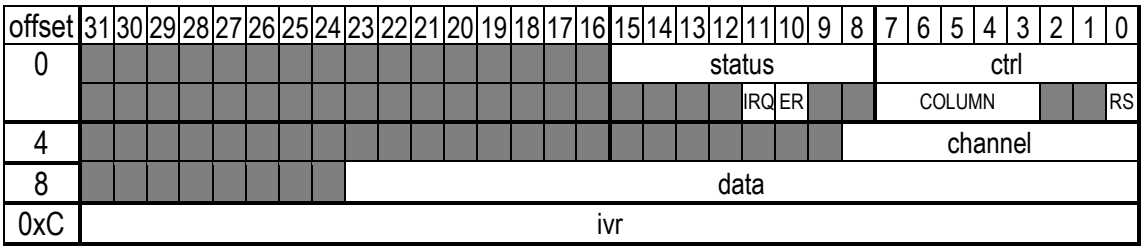
Visa en kodsekvens som evaluerar uttrycket `vc[i]-vc[i+1]` till register R0. (4p).

c) 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 (6p).

```
int f ( int a )
{
  extern int g(int); /* Funktionen är definierad tidigare */
  if ( a > g(a+1) )
    return 1;
  return 0;
}
```

Uppgift 2 (10p)

Följande port som utgör ett gränssnitt mot en yttre periferienhet är placerad på adress `0xFF600000`:



Bitarna IRQ och ER ingår alltså i status-delen av registret medan biten RS och bitfältet COLUMN ingår i ctrl-delen av registret med offset 0.

a) Visa lämpliga makrodefinitioner för referens (åtkomst) av portens register status respektive data. (2p)

b) Visa med en typdefinition `PORT`, hur porten kan avbildas med en *struct*-definition. (2p)
Visa speciellt hur delen data då refereras med hjälp av din typdefinition (2p).

c) Visa hur portens första register (offset=0) kan avbildas med en bitfältdeklaration av typen `REG0` där de olika bitarna kan refereras var för sig. (4p).

Uppgift 3 (8p)

Konstruera en C-funktion som undersöker en parameter med avseende på antalet 1-ställda bitar.

Funktionen deklareras:

```
int  bitcheck( unsigned int *pp, int * num );
```

`pp` är en pekare till det värde som ska undersökas
`num` är en pekare till en plats för returvärde, dvs. antalet 1-ställda bitar hos parametern
Funktionen ska returnera 1 om antalet ettor hos parametern är udda, annars ska funktionsvärdet vara 0.

Uppgift 4 (6p)

Visa hur man kan implementera *icke-blockerande* fördröjning via två funktioner:
`void delay10ms( void )` som initierar en fördröjning om 10 ms, och
`int delay_is_active( void )` som kontrollerar om fördröjningen är slutförd och i så fall återställer SysTick och returnerar 0, annars returneras 1. Systemets klockfrekvens är 168 MhZ.

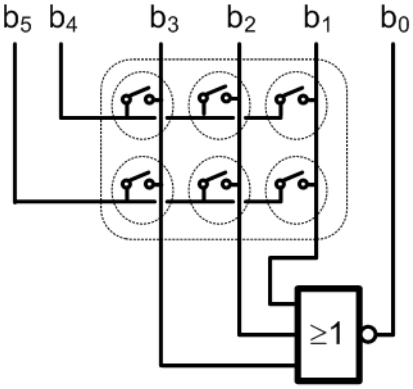
Följande sekvens illustrerar hur funktionerna är tänkta att användas:

```
…
…
delay10ms();
… /* Gör något annat */
…
if( delay_is_active()==0 )
{
  /* 10 ms har passerat … */
}else{
  /* Väntar fortfarande på fördröjningen */
}
…
```

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

Uppgift 5 (10p)

Ett tangentbord för inmatning av sex olika tecken ska konstrueras. Sex stycken återfjädrande omkopplaare 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 1. För att detta ska vara tillförlitligt måste dessa bitars ingångar förses med "*pull-down*" samtidigt som utgångarna ska vara "*push-pull*". Metoden kallas *koincidensavsökning*.

Kretsen kan också användas för att känna av en tangentnedtryckning med hjälp av avbrott. För detta kopplas kolumnerna till en NOR-grind, vars utgång är kopplad till bit 0 hos port E, denna måste ha "*pull-up*".

Anm: Vi påminner om NOR-grindens funktion: utsignalen är '1' så länge alla ingångar är '0'. Då någon av ingångarna blir '1' blir utsignalen '0'.

a) Visa en initieringsrutin `void init( void )` (8p) där:
   - GPIO modulen initieras för dessa portpinnar. Port E ska initieras för användning med tangentbordet. Bitarna b6 och b7 används inte, observera att endast konfigurationen för des portpinnar som används får ändras vid konfigureringen.
   - SYSCFG, EXTI och NVIC konfigureras för avbrott via b0.
   - Avbrottsvektor initieras med adress till avbrottsfunktionen `void at_interrupt(void)`. Antag att vektortabellen börjar på adressen `0x2001C000` i minnet.

b) Visa avbrottsrutinen `void at_interrupt( void )` som kontrollerar om avbrott begärts via b0 och i så fall kvitterar avbrottet. (2p)

Lösningsförslag

Uppgift 1 a

```
   .align
a: .space 4
b: .space 4
   LDR   R1,=b
   LDRH  R0,[R1]
   SXTH  R0,R0
   BL    fcall
   LDR   R1,=a
   STR   R0,[R1]
```

Uppgift 1 b

```
   LDR   R0,i       @ R0 =i
   LDR   R1,=vc     @ R1 =&vc
   ADD   R1,R0,R1   @ R1 =&vc[i]
   LDRB  R3,[R1]    @ R3 =vc[i]
   LDRB  R2,[R1,#1] @ R2 =vc[i+1]
   SUB   R0,R3,R2   @ R0 =vc[i]-vc[i+1]
```

Uppgift 1 c

```
@
@  R4: temp register
f: PUSH  {R4,LR}  @ R4 och LR ändras av subrutinen
   MOV   R4,R0    @ spill parameter till R4
   ADD   R0,R0,#1 @ R0=a+1
   BL    g        @ g(a+1)
   CMP   R4,R0    @ a > g(a+1)
   BGT   .L1
   MOV   R0,#0    @ return 0
   B     .L2
.L1:
   MOV    R0,#1   @ return 1:
.L2:
   POP    {R4,PC}
```

Uppgift 2a:

```
#define status (( unsigned char *) 0xFF600001)
#define data (( unsigned long *) 0xFF600008)
```

Uppgift 2b:

```
typedef struct{
  unsigned char ctrl;
  unsigned char status;
  short unused0;
  unsigned short channel;
  unsigned short unused1;
  unsigned long data;
  unsigned long ivr;
} PORT;
```

Referens: `( (PORT *) (0xFF600000))->data;`

Uppgift 2c:

```
typedef struct{
    unsigned int RS:1;
    unsigned int :2;
    unsigned int COLUMN:5;
    unsigned int :2;
    unsigned int ER:1;
    unsigned int IRQ:1;
}REG0;
```

Uppgift 3:

```
int bitcheck( unsigned int *pp, int *num)
{
  int   retval = 0;

  while(*pp)
  {
    if( *pp & 1 )
      retval++;
    *pp >>= 1;
  }
  *num = retval;
  return retval & 1;
}
```

Uppgift 4

```
#define STK_CTRL  ((volatile unsigned int *)(0xE000E010))
#define STK_LOAD  ((volatile unsigned int *)(0xE000E014))
#define STK_VAL   ((volatile unsigned int *)(0xE000E018))
void delay10ms( void )
{
 /* SystemCoreClock = 168000000 */
 *STK_CTRL = 0;
 *STK_LOAD = ( (1680000) -1 );
 *STK_VAL = 0;
 *STK_CTRL = 5;
}

int delay_is_active( void )
{
  if(*STK_CTRL & 0x10000 )
  {
    *STK_CTRL = 0;
    return 0;
  }
  return 1;
}
```

Uppgift 5a:

```
#define GPIO_E_MODER       ((volatile unsigned int *)0x40021000)
#define GPIO_E_OTYPER      ((volatile unsigned int *)0x40021004)
#define GPIO_E_PUPDR       ((volatile unsigned int *)0x4002100C)
#define NVIC_EXTI0_IRQ_BPOS   (1<<6)
#define EXTI0_IRQ_BPOS        (1<<0)
#define EXTI0_IRQVEC      0x2001C058

#define SYSCFG_EXTICR1 ((volatile unsigned short *)  0x40013808)
#define EXTI_IMR          ((volatile unsigned int *)    0x40013C00)
#define EXTI_RTSR     ((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 init( void )
{
  *GPIO_E_MODER = 0x55550500;
  /* bit 3,2,1 PULL DOWN, Bit 0 Pull up, ty IRQ aktiv låg */
  *GPIO_E_PUPDR = 0x000000A9;
  *GPIO_E_OTYPER = 0;

  *SYSCFG_EXTICR &= 0xFFF0;
  *SYSCFG_EXTICR |= 0x0004;          /* PE0->EXTI0  */
  *EXTI_IMR |= EXTI0_IRQ_BPOS;
  *EXTI_RTSR &= ~EXTI0_IRQ_BPOS;     /* EJ Avbrott på POSITIV flank */
  *EXTI_FTSR |= EXTI0_IRQ_BPOS;      /* Avbrott på NEGATIV flank */
  *NVIC_ISER0 |= NVIC_EXTI0_IRQ_BPOS; /* Aktivera avbrott i NVIC */
   *((void (**)(void) ) EXTI0_IRQVEC ) = at_interrupt;
}
```

Uppgift 5b:

```
void at_interrupt( void )
{
  if( *EXTI_PR & EXTI0_IRQ_BPOS )
    *EXTI_PR |= EXTI0_IRQ_BPOS;   /* Återställ avbrott */
}
```