Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA

Tentamen med lösningsförslag

EDA488 Maskinorienterad programmering Z

Fredag 22 augusti 2025, 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 (”över-strykningar”) 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

Uppgifter 1 t.o.m 3 utgörs av flervalsfrågor.
Försök först lösa uppgifterna, så som du lärt dig i kursen, jämför sedan med flervalsalternativen och ange det alternativ du anser är korrekt. Högst ett svarsalternativ får anges. Siffror inom parentes anger möjliga poäng på uppgiften.

Observera att, bland svarsalternativen finns alternativ som *kan* ge poängavdrag. Om du är mycket osäker på ditt svar bör du därför i stället välja att avstå.

Avge dina flervalssvar på blanketten, längst bak i tentamenstesen. Skicka med denna blankett med dina övriga lösningar.

För övriga uppgifter gäller:

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 40 och tentamenspoäng ger slutbetyg enligt:
16p ≤ betyg 3 < 24p ≤ betyg 4 < 32p ≤ betyg 5

Uppgift 1 (-1..4p)

Följande deklarationer är givna på "toppnivå".

```
static unsigned char f,c;
int a;
int foo2(unsigned char);
short foo1(unsigned char,int);
```

Vi har nu tilldelningen:

```
a = foo1 ( c, foo2(f) );
```

Vilken av följande implementeringar i ARM v6-kod är korrekt?

A

```
LDR   R0,=f
LDRB  R0,[R0]
BL    foo2
LDR   R0,=c
LDRB  R0,[R0]
BL    foo1
SXTH  R0,R0
LDR   R1,=a
STRH  R0,[R1]
```

B

```
LDR   R0,=f
LDRB  R0,[R0]
BL    foo2
LDR   R0,=c
LDRB  R0,[R0]
BL    foo1
SXTH  R0,R0
LDR   R1,=a
STR   R0,[R1]
```

C

```
LDR   R0,=f
LDRB  R0,[R0]
BL    foo2
MOV   R1,R0
LDR   R0,=c
LDRB  R0,[R0]
BL    foo1
SXTH  R0,R0
LDR   R1,=a
STR   R0,[R1]
```

D

```
LDR   R0,=f
LDR   R0,[R0]
BL    foo2
MOV   R1,R0
LDR   R0,=c
LDR   R0,[R0]
BL    foo1
SXTH  R0,R0
LDR   R1,=a
STR   R0,[R1]
```

E

```
LDR   R0,=f
LDRB  R0,[R0]
BL    foo2
MOV   R1,R0
LDR   R0,=c
LDRB  R0,[R0]
BL    foo1
LDR   R1,=a
STRH  R0,[R1]
```

F

```
LDR   R0,=f
LDR   R0,[R0]
BL    foo2
LDR   R0,=c
LDR   R0,[R0]
BL    foo1
LDR   R1,=a
STRH  R0,[R1]
```

G

```
LDR   R0,=f
LDR   R0,[R0]
BL    foo2
LDR   R0,=c
LDR   R0,[R0]
BL    foo1
LDR   R1,=a
STR   R0,[R1]
```

H

```
LDR   R0,=f
LDRB  R0,[R0]
BL    foo2
MOV   R1,R0
LDR   R0,=c
LDRB  R0,[R0]
BL    foo1
LDR   R1,=a
STR   R0,[R1]
```

Uppgift 2 (-1..6p)

C-funktionen `bitcheck` 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 returnerar 1 om antalet ettor hos parametern är udda, annars ska funktionsvärdet vara 0.

Sex olika programmerare får i uppgift att implementera funktionen. Resultaten blir likartade men bara en programmerare presterar ett helt korrekt resultat. De olika alternativen finns nedan, ange alternativet med den korrekta lösningen.

A

```
int bitcheck( unsigned int *pp, int *num)
{
  int  retval = 0;
  while(pp)
  {
    if( pp & 1 )
      retval++;
    pp >>= 1;
  }
  num = retval;
  return retval & 1;
}
```

B

```
int bitcheck( unsigned int *pp, int *num)
{
  int  retval = 0;
  while(*pp)
  {
    if( *pp & 1 )
      retval++;
    *pp >>= 1;
  }
  num = retval;
  return retval & 1;
}
```

C

```
int bitcheck( unsigned int *pp, int *num)
{
  int  retval = 0;
  while(*pp)
  {
    if( *pp & 1 )
      retval++;
    *pp >>= 1;
  }
  *num = retval;
  return retval & 1;
}
```

D

```
int bitcheck( unsigned int *pp, int *num)
{
  int  retval = 0;
  while(pp)
  {
    if( pp & 1 )
      retval++;
    pp >>= 1;
  }
  *num = retval;
  return retval & 1;
}
```

E

```
int bitcheck( unsigned int *pp, int *num)
{
  int  retval = 0;
  while(pp)
  {
    if( pp & 1 )
      retval = retval+sizeof(int);
    pp >>= 1;
  }
  num = retval;
  return retval & 1;
}
```

F

```
int bitcheck( unsigned int *pp, int *num)
{
  int  retval = 0;
  while(*pp)
  {
    if( *pp & 1 )
      retval = retval+sizeof(int);
    *pp >>= 1;
  }
  *num = retval;
  return retval & 1;
}
```

Uppgift 3 (-1..4p)

Följande deklarationer är givna på "toppnivå".

```
static short a,b;
static int index;
static int vi[100];
```

Vi har nu tilldelningen:

```
vi[ index ] = (a+b) * (a-b);
```

Ange det alternativ (A-F) som visar en korrekt evaluering av hur uttrycken kodas i ARM v6 assemblerspråk. Om flera lösningar ger samma resultat ska du ange den kortaste.

A

```
LDR   R0,a
LDR   R1,b
ADD   R2,R0,R1
SUB   R3,R0,R1
MUL   R2,R3
LDR   R1,vi
LDR   R0,index
LSL   R0,R0,#2
ADD   R0,R1,R0
STR   R2,[R0]
```

B

```
LDR   R0,a
LDR   R1,b
ADD   R2,R0,R1
SUB   R3,R0,R1
MUL   R2,R3
LDR   R1,vi
LDR   R0,index
LDR   R0,[R0]
ADD   R0,R1,R0
STR   R2,[R0]
```

C

```
LDR   R0,a
LDR   R1,b
ADD   R2,R0,R1
SUB   R3,R0,R1
MUL   R2,R3
LDR   R1,=vi
LDR   R0,=index
LDR   R0,[R0]
LSL   R0,R0,#2
ADD   R0,R1,R0
STR   R2,[R0]
```

D

```
LDR   R0,a
LDRH  R0,[R0]
SXTH  R0,R0
LDR   R1,b
LDRH  R1,[R1]
SXTH  R1,R1
ADD   R2,R1,R0
SUB   R3,R1,R0
LSL   R2,R3,R3
LDR   R1,vi
LDR   R0,index
LDR   R0,[R0]
LSL   R0,R0,#2
ADD   R0,R1,R0
STR   R2,[R0]
```

E

```
LDR   R0,=a
LDRH  R0,[R0]
SXTB  R0,R0
LDR   R1,=b
LDRH  R1,[R1]
SXTB  R1,R1
ADD   R2,R1,R0
SUB   R3,R1,R0
LSL   R2,R3
LDR   R1,=vi
LDR   R0,=index
LDR   R0,[R0]
LSL   R0,R0,#2
ADD   R0,R1,R0
STR   R2,[R0]
```

F

```
LDR   R0,=a
LDRH  R0,[R0]
SXTH  R0,R0
LDR   R1,=b
LDRH  R1,[R1]
SXTH  R1,R1
ADD   R2,R0,R1
SUB   R3,R0,R1
MUL   R2,R3
LDR   R1,=vi
LDR   R0,=index
LDR   R0,[R0]
LSL   R0,R0,#2
ADD   R0,R1,R0
STR   R2,[R0]
```

Uppgift 4 (6p)

Följande port som utgör ett gränssnitt mot en yttre periferienhet är placerad på adress `0xFF000000`:

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0																				f1									f0				CREG
4	XHIGH																XLOW																XREG

a) Visa lämpliga makrodefinitioner för åtkomst av portens register XREG respektive CREG. Visa också speciellt hur innehållet i XREG (b31..b0) refereras (2p).

b) Visa hur porten kan avbildas med en *struct*-definition där register XREG ska kunna refereras i sin helhet, (bit31..bit0), såväl som delarna XHIGH (bit31..bit16) respektive XLOW (bit15..bit0). Visa speciellt hur delen XHIGH då refereras (4p).

Uppgift 5 (8p)

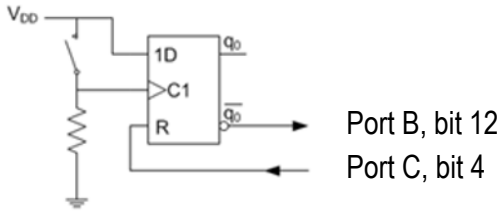
Man vill skapa en tidtagarfunktion, med upplösningen 1 ms. Det får antas att mättiderna aldrig överskrider 10 sekunder. Visa hur en sådan tidtagarfunktion kan implementeras i ett *MD407* laborationssystem, med modulen SYSTICK. Laborationssystemets klockfrekvens är 168 MHz. Följande funktioner, ska finnas:

- `void start_clock(void);` Nollställer och startar klockan
- `void stop_clock(void);` Stannar klockan
- `unsigned int get_clock( void);` Returnera klockans aktuella värde i millisekunder

Klockan ska kunna användas för tidmätning parallellt med andra funktioner och måste därför vara *icke-blockerande*.

Uppgift 6 (6p)

En avbrottsvippa kopplas till *MD407* enligt följande figur:



Vi påminner om vippans funktion:

Då nivån på C1 ändras till 1 (positiv flank) ändras vippans interna tillstånd q_0 till 1D (1), om R samtidigt är 0.

Då R har värdet 1, tvingas q_0 till 0.

Den externa avbrottsmekanismen (EXTI) ska användas för att detektera knapptryckningar. Ingen hänsyn behöver tas till eventuella kontaktstudsar. Avbrott ska ske på negativ flank.

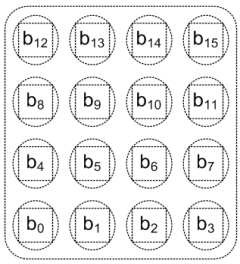
Förutsätt att makrodefinitioner för pekare till använda register finns givna med namnkonventioner enligt *QuickGuide*.

Visa med en funktion `app_init` hur avbrottsmekanismerna initieras, dvs. SYSCFG, EXTI, Port C, Port B och NVIC ska initieras.

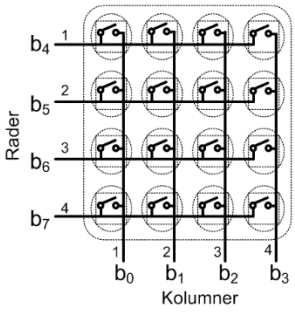
Tänk på att bara de portpinnar som berörs får ändras vid konfigureringen.

Uppgift 7 (6p)

Under laborationerna har du arbetat med ett enkelt tangentbord där tangenterna organiserats i rader och kolumner. Man vill kunna detektera flera samtidigt nedtryckta tangenter och därför konstruera en tangentbordsrutin som returnerar ett statusord (16 bitar) där varje tangent har en bitposition och värdet 1 indikerar en nedtryckt tangent medan värdet 0 indikerar en uppsläppt tangent. På grund av tangentbordets konstruktion är det lämpligt att välja avbildning enligt figuren till höger mellan bitposition och tangent.



Antag att port D, bit 0..7, kopplats till tangentbordet enligt figuren till höger. Eftersom flera tangenter kan tryckas ned samtidigt vill vi göra en ”kortslutningssäker” lösning för koincidensavsökning av tangentbordet. Utgångarna ska därför vara OPEN-DRAIN. De avlästa kolumnerna ska vara PULL UP.
Skapa en initieringsfunktion `void init_app( void )` som konfigurerar port D för användning med tangentbordet och en tangentbordsrutin `unsigned short keyb (void)` som söker av tangentbordet och placerar kolumnvärde för varje rad i en `unsigned short` och returnerar denna.



Förutsätt att makrodefinitioner för pekare till använda register finns givna med namnkonventioner enligt *QuickGuide*.

Svarsblankett

Anonym kod

Maskinorienterad programmering, tentamen flervalsfrågor.

Kryssa i det alternativ du anser vara korrekt, eller avstå.
Lämna in denna blankett tillsammans med dina övriga lösningar.

Fråga	A	B	C	D	E	F	G	H
1	0	1	4	0	0	-1	-1	2
2	0	-1	6	-1	0	0		
3	-1	-1	0	0	0	4		

Lösningsförslag

Uppgift 1 (4p):

Svarsalternativ C

```
    LDR     R0,=f       @ R0 <- &f
    LDRB    R0,[R0]     @ R0 <- (unsigned char) f
    BL      foo2        @ R0 <- foo2 (f)
    MOV     R1,R0       @ R1 <- foo2 (f)
    LDR     R0,=c       @ R0 <- &c
    LDRB    R0,[R0]     @ R0 <- (unsigned char) c
    BL      foo1        @ R0 <- foo1 ( c, foo2 (f))
    SXTH    R0,R0       @ R0 <- (int) foo1 ( c, d(f))
    LDR     R1,=a       @ R1 <- &a
    STR     R0,[R1]     @ a <- (int) foo1 ( c, foo2 (f))
```

Uppgift 2 (-1..6p):

Svarsalternativ C

```
int bitcheck( unsigned int *pp, int *num)
{
  int   retval = 0;
  while(*pp)
  {
    if( *pp & 1 )
      retval++;
    *pp >>= 1;
  }
  *num = retval;
  return retval & 1;
}
```

Uppgift 3 (-1..6p)

Svarsalternativ F.

```
    LDR   R0,=a     @ R0← &a
    LDRH  R0,[R0]   @ R0← a
    SXTH  R0,R0     @ R0← (int) a
    LDR   R1,=b     @ R1← &b
    LDRH  R1,[R1]   @ R1← b
    SXTH  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,=vi    @ R1← &vi
    LDR   R0,=index @ R0← &index
    LDR   R0,[R0]   @ R0← index
    LSL   R0,R0,#2  @ R0← index*sizeof(int)
    ADD   R0,R1,R0  @ R0← &vi+ index*sizeof(int)
    STR   R2,[R0]   @ vi[index] ← ((int)a+(int)b) * ((int)a-(int)b)
```

Uppgift 4 a (2p)

```
#define CREG ( (volatile unsigned long *) 0xFF000000)
#define XREG ( (volatile unsigned long *) 0xFF000004)
Referens: *XREG
```

Uppgift 4 b (4p)

```
struct port{
  unsigned long creg;
  union{
    unsigned long xreg;
    struct{
      unsigned short xlow;
      unsigned short xhigh;
    };
  };
};
Referens: (((volatile struct port *) 0xFF000000)->xhigh);
alternativt:
#define PORT ( (volatile struct port *) 0xFF000000)
PORT->xhigh;
```

Uppgift 5 (8p)

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

static volatile unsigned int systick_irqs;

static void  systick_irq_handler( void )
{
   systick_irqs++;
}
void start_clock( void )
{
    systick_irqs = 0;
   *((void (**)(void) ) 0x2001C03C ) = systick_irq_handler;
   /* SystemCoreClock = 168000000 */
   *STK_CTRL = 0;
   *STK_LOAD = ( 168000-1 );
   *STK_VAL = 0;
   *STK_CTRL = 7;
}
void stop_clock(void)
{
   *STK_CTRL = 0;
}
unsigned int get_clock( void )
{
   return systick_irqs;
}
```

Uppgift 6 (6p)

```
#define NVIC_EXTI12_IRQ_BPOS (1<<(40-32))
#define EXTI12_IRQ_BPOS          (1<<12)
void app_init( void )
{
  /* Port C, pin 4 utgång */
  *GPIOC_MODER  &= ~(0x300);
  *GPIOC_MODER  |= (0x100);
  /* Utgång ”push/pull” */
  *GPIOC_OTYPER &= ~(1<<4);

  /* Port B, pin 12 ingång */
  *GPIOB_MODER  &= ~(0x30000000);
  /* Ingång ”floating” */
  *GPIOB_PUPDR  &= ~(0x30000000);
  /* PB12->EXTI12  */
  *SYSCFG_EXTI4 &= ~(0x000F);
  *SYSCFG_EXTI4 |= 0x0001;
  *EXTI_IMR |= EXTI12_IRQ_BPOS;
  *EXTI_FTSR |= EXTI12_IRQ_BPOS;
  *EXTI_RTSR &= ~EXTI12_IRQ_BPOS;
  *NVIC_ISER1 |= NVIC_EXTI12_IRQ_BPOS;
}
```

Uppgift 7 (6p)

```
void init_app( void )
{
    /*  PORT D
    b8-b4 är utgångar till rader
    b3-b0  är ingångar från kolumner
    */
    *GPIO_D_MODER   = 0x00005500;
    /* Input, pull up */
    *GPIO_D_PUPDR   = 0x00000055;
    /* outputs are open drain */
    *GPIO_D_OTYPER  = 0x00F0;
}

unsigned short keyb (void)
{
    int row;
    unsigned short pad;
    *GPIO_D_ODR_LOW = ~0x80;
    pad = (unsigned short)   (*GPIO_D_IDR_LOW & 0xF);
    *GPIO_D_ODR_LOW = ~0x40;
    pad |= (unsigned short)  ((*GPIO_D_IDR_LOW <<4) & 0x00F0);
    *GPIO_D_ODR_LOW = ~0x20;
    pad |= (unsigned short)  ((*GPIO_D_IDR_LOW <<8) & 0x0F00);
    *GPIO_D_ODR_LOW = ~0x10;
    pad |= (unsigned short)  ((*GPIO_D_IDR_LOW <<12) & 0xF000);
    *GPIO_D_ODR_LOW =  0xFF;
    return pad^0xFFFF;  /* Bitmönstret är inverterat... */
}
```