Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA

Tentamen med lösningsförslag

EDA482 Maskinorienterad programmering D
EDA483 Maskinorienterad programmering E
DAT017 Maskinorienterad programmering IT
DIT153 Maskinorienterad programmering GU

Fredag 2 juni 2023, kl. 8.30 - 12.30

Examinatorer
Roger Johansson, tel. 772 57 29
Erik Sintorn, tel. 772 52 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)

Vi har de globala deklarationerna:

```
short   v[8][9][10];
int     i,j,k;
```

a) Visa hur variabeldeklarationerna kodas i assembler i ARMv6 assemblerspråk (2p).
b) Visa ett *uttryck* (använd C-syntax) för hur adressen till `v[i][j][k]` bildas (4p).
c) För funktionen `f` gäller nu deklarationen:

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

visa hur då följande tilldelningssats och funktionsanrop kodas i ARMv6 assemblerspråk:

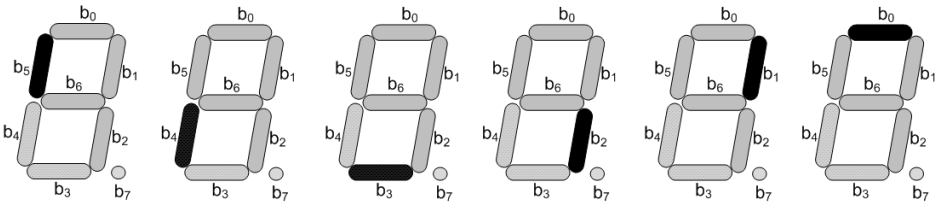
```
i = (int) f( (short) j, v);
```
(4p)

d) 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 användningen av register framgår och assemblerkoden kan kopplas till motsvarande konstruktion i C-programmet (6p).

```
int f ( int i, int j, char *v )
{
  if ( v[i]<v[j] )
    return j;
  return i;
}
```

Uppgift 2 (10p)

Under laborationerna har du arbetat med en sjusifferindikator. Man vill skapa ett roterande mönster, där segmenten tänds och släcks i ordning enligt:



dvs, ger intrycket av en ”mask” som kryper runt, varv efter varv... Sjusifferindikatorn ansluts till *MD407*, port D bitar 0-7. Skriv programkod i C enligt följande:

a) En initieringsfunktion `init_app` som initierar GPIO-modulen för användning med sjusifferindikatorn. Adresser ska definieras med *macros*, på det sätt som lärts ut i kursen.(2p)
b) Ett huvudprogram `main`, som: kontinuerligt, i tur och ordning tänder/släcker segmenten enligt figuren ovan.(4p)
c) Modifiera nu programmet så att det tar 1,2 sekunder att fullborda ett varv. Använd SysTick och inplementera blockerande fördröjning. Systemets klockfrekvens är 168 MHz. (4p)

Uppgift 3 (8p)

I en så kallad allokeringskarta representerar varje enskild bit tillståndet allokerad (=1) eller fri (=0). Betrakta en allokeringskarta (1 kByte) organiserad som 128x8 bitar, dvs: `char alloc_map[128];`

Vi vill nu skapa en funktion `int find_first_free( char *map, int *itemno, char *mask);`

som söker igenom en allokeringskarta. Om ingen ”fri” bit påträffas i kartan, ska funktionen returnera 0.

Annars ska returparametern `itemno` innehålla index för den medlem i fältet som innehåller biten, returparametern `mask` ska innehålla en bitmask som är 1 för den påträffade biten, noll för övriga bitar.

EXEMPEL:

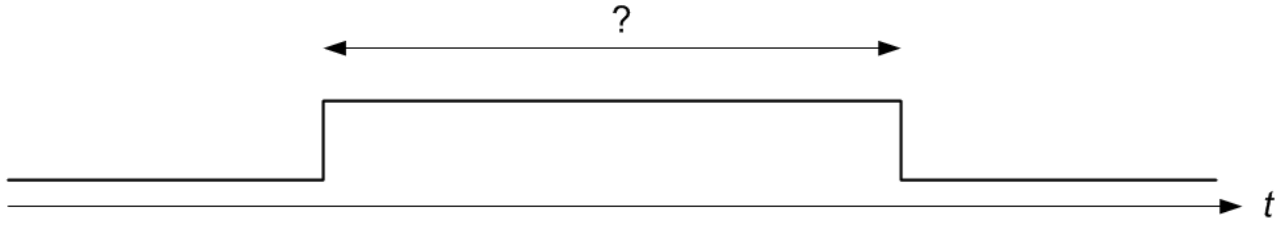
Om vi har en allokeringskarta `alloc_map[]={0xFF,0xFF,0x80,.......};` och deklarationerna `int itemno; char mask;` och utför följande anrop:

```
if( find_first_free( alloc_map , &itemno, &mask) ) ...
```

ska funktionen returnera 1, `itemno` ska vara 2 och `mask` ska vara 0x40 $(01000000)_2$.

Uppgift 4 (6p)

Vi vill skapa en mätapplikation som mäter längden av en positiv puls hos en signalledare.



För detta ansluter vi signalledaren till portpinne PB3 hos *MD407*.
De externa funktionerna `void start_M(void)` och `void stop_M(void)` finns givna.

a) Visa hur SYSCFG konfigureras så att PB3 kopplas till EXTI-modulen hos *MD407*. Tänk på att bara PB3 ska konfigureras och att övriga EXTI-linor inte får ändras av din kod. (1p)
b) Visa hur EXTI och NVIC konfigureras så att avbrott genereras på båda flanker. (2p)
c) Visa en avbrottsfunktion `void at_irq(void)` som startar mätning av en puls vid en positiv flank hos signalledaren, och avslutar en pågående mätning vid en negativ flank, genom att anropa de givna funktionerna. Visa också hur avbrottsvektorn initieras. (3p)

Uppgift 5 (10p)

En så kallad *flexskive-enhet*, med följande gränssnitt, är ansluten till *MD407* på adress 0xF0000100.

Registeruppsättning

offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0																	CTRL
1																	SEC
2																	S_TRACK
3																	STAT
4																	ADDR_LOW
6																	ADDR_HIGH

Åtkomst av register CTRL, SEC, S_TRACK och STAT måste vara *byte*-format

Åtkomst av register ADDR_LOW och ADDR_HIGH måste vara *halfword*-format

offset	7	6	5	4	3	2	1	0	Register
0x00				IR	EX	CMD			CTRL

offset	7	6	5	4	3	2	1	0	Register
0x01			SECNO						SEC

offset	7	6	5	4	3	2	1	0	Register
0x02	S	TRACK							S_TRACK

offset	7	6	5	4	3	2	1	0	Register
0x03		IRQ	BUSY	ERR	EIND				STAT

a) Visa lämpliga macrodefinitioner för porten med detta gränssnitt (2p)
b) Visa en lämplig typdeklaration (`struct fdc`) av porten (3p).
c) Visa en lämplig typdeklaration (`struct fdc`) med bitfältsdeklarationer av portens register och bitar. (5p)

Lösningsförslag

Uppgift 1 a

```
v:    .space   (8*9*10)*2
      .align   2
i:    .space   4
j:    .space   4
k:    .space   4
```

Uppgift 1 b

```
&v[i][j][k]= &v[0]+( ((i*9*10)+(j*10) + k ) * sizeof(short) )
```

Uppgift 1 c

```
    LDR   R0,=j
    LDR   R0,[R0]
    SXTH  R0,R0
    LDR   R1,=v
    BL    f
    LDR   R1,=i
    STR   R0,[R1]
```

Uppgift 1 d

```
@ int f(int i, int j, char *v)
@ Register:
@ R0: param i
@ R1: param j
@ R2: param v
@ R3: temp för v[i]
@ R4: temp för v[j]
f:
   PUSH   {R4}
   LDRB  R3,[R2,R0]   @ R3<-v[i]
   LDRB  R4,[R2,R1]   @ R4<-v[j]
   CMP   R3,R4        @ v[i]<v[j] ?
   BGE   .L1          @ return i
   MOV   R0,R1        @ return j
.L1:
   POP   {R4}
   BX    LR
```

Uppgift 2a

```
#define    PORT_D_BASE    0x40020C00
#define    GPIO_D_MODER   ((volatile unsigned int *)(PORT_D_BASE))
#define    GPIO_D_OTYPER  ((volatile unsigned short *)(PORT_D_BASE+0x4))
#define    GPIO_D_ODRLOW  ((volatile unsigned char *)   (PORT_D_BASE+0x14))

void   init_app( void )
{
    *GPIO_D_MODER = 0x5555;        /* Bit 0-7 är utgångar */
    *GPIO_D_OTYPER = 0x00000000;   /* " push/pull" */
    *GPIO_D_ODRLOW = 0;
}
```

Uppgift 2b

```
void   main( void )
{
  static unsigned char Worm[]={ 0x20,0x10,8,0x4,2,1 };
  init_app();
  while(1)
  {
    for( int i = 0; i < 6; i++ )
    {
      *GPIO_D_ODRLOW = Worm [i];
    }
  }
}
```

Uppgift 2c

Konstruera 200 ms fördröjning (`=16 800 000 * 2 klockpulser`). LOAD-registret tar max 24 bitar. Största värde som kan användas blir då: `16 777 215`. Skapa därför 10 ms fördröjning med SysTick

```
void delay_10ms( void )
{
    /* SystemCoreClock = 168000000 */
    *STK_CTRL = 0;
    *STK_LOAD = ( 1680000 -1 ); /* 10 ms */
    *STK_VAL = 0;
    *STK_CTRL = 5;
    while( (*STK_CTRL & 0x10000 )== 0 );
    *STK_CTRL = 0;
}
```

```
voidmain( void )
{
  static unsigned char Worm[]={ 0x20,0x10,8,0x4,2,1 };
  init_app();
  while(1)
  {
    for( int i = 0; i < 6; i++ )
    {
      *GPIO_D_ODRLOW = Worm[i];
      for( int j = 0; j < 20; j++ )
        delay_10ms();
    }
  }
}
```

Uppgift 3

```
int find_first_free( char *map, int *itemno, char *mask)
{
  for( int i = 0; i< 128; i++ )
  {
    if( map[i] != 0xFF )
    {
      *itemno = i;
      char m = 0x80;
      while(map[i] & m )
        m = m >> 1;
      *mask = m;
      return 1;
    }
  }
  return 0;
}
```

Uppgift 4

Macro-definitioner för register

```
#define PORT_B_BASE     0x40020400
#define GPIO_B_MODER    ((volatile unsigned int  *) (PORT_B_BASE))
#define GPIO_B_IDR      ((volatile unsigned char *) (PORT_B_BASE+0x11))

#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 EXTI3_IRQVEC    0x2001C064 (Räcker med offset 0x64 för full poäng)
#define NVIC_ISER0      ((volatile unsigned int  *) 0xE000E100 )
#define NVIC_ICPR0      ((volatile unsigned int  *) 0xE000E280 )
#define NVIC_IPR0       ((volatile unsigned int  *) 0xE000E400 )
```

Uppgift 4a

```
  *GPIO_B_MODER &= 0xFFFFFF3F;   /* PB3, inport */
  /* Antag avbrottssignal PUSH/PULL, annars kan PULL DOWN läggas på PB3 , behövs dock inte här för full poäng)
  *SYSCFG_EXTICR1 &= 0x0FFF;
  *SYSCFG_EXTICR1 |= 0x1000;     /* PB3->EXTI3  */
```

Uppgift 4b

```
  *EXTI_IMR   |= (1<<3);
  *EXTI_FTSR  |= (1<<3);
  *EXTI_RTSR  |= (1<<3);
  *NVIC_ISER0 |= (1<<9);
```

Uppgift 4c

```
void at_irq ( void )
{
   if( *GPIO_B_IDR & (1<<3) )
     start_M();
   else
     stop_M();
   *EXTI_PR |= (1<<3);
}
```

Initiering av avbrottsvektor:

```
  *((void (**)(void) ) EXTI3_IRQVEC ) = at_irq;
```

Uppgift 5a

```
#define CTRL      ((volatile unsigned char *)  0xF0000100)
#define SEC       ((volatile unsigned char *)  0xF0000101)
#define S_TRACK   ((volatile unsigned char *)  0xF0000102)
#define S_STAT    ((volatile unsigned char *)  0xF0000103)
#define ADDR_LOW  ((volatile unsigned short *) 0xF0000104)
#define ADDR_HIGH ((volatile unsigned short *) 0xF0000106)
```

Uppgift 5b

```
struct fdc{
  volatile unsigned char ctrl;
  volatile unsigned char sec;
  volatile unsigned char s_track;
  volatile unsigned char stat;
  volatile unsigned short addr_low;
  volatile unsigned short addr_high;
};
```

Uppgift 5c

```
struct fdc{
  volatile unsigned char cmd:3, ex:1, ir:1;
  volatile unsigned char :0, secno:6;
  volatile unsigned char :0, track:7, s:1;
  volatile unsigned char :0, eind:4, err:1, busy:1, irq:1;
  volatile unsigned short addr_low;
  volatile unsigned short addr_high;
};
```

Följande lösning ger **också** full poäng (liten otydlighet i uppgiftstextens formulering (”register och bitar”)

```
struct fdc{
  union{
    volatile unsigned char ctrl;
    volatile unsigned char cmd:3, ex:1, ir:1;
  };
  union{
    volatile unsigned char sec;
    volatile unsigned char :0, secno:6;
  };
  union{
   volatile unsigned char s_track;
   volatile unsigned char :0, track:7, s:1;
  };
  union{
    volatile unsigned char stat;
    volatile unsigned char :0, eind:4, err:1, busy:1, irq:1;
  };
  volatile unsigned short addr_low;
  volatile unsigned short addr_high;
};
```