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 18 augusti 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:

```
int   v[10][20];
int   i,j;
```

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

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

visa hur då följande funktionsanrop kodas i ARMv6 assemblerspråk:

```
k = f( j , v);
```

(4p)

d) Funktionen `int o( int )` är definierad. Visa hur följande funktion **g** kan kodas i ARM v6 assemblerspråk. (6p)

```
int g ( int a )
{
  if ( a > o(a+5) )
    return 1;
  return 0;
}
```

Uppgift 2 (8p)

Konstruera en C-funktion som undersöker en parameter med avseende på antalet 0-ställda bitar.
Funktionen deklareras:

```
short bitcheck( unsigned int *p, int * num );
```

- `p` är en pekare till det värde som ska undersökas
- `num` är en pekare till en plats dit antalet 0-ställda bitar hos parametern `p`, ska skrivas

Returvärdet för funktionen ska vara skilt från 0 om antalet nollor hos parametern är jämt delbart med 2, annars ska returvärdet vara 0. Koden ska skrivas utan att förutsätta någon specifik storlek hos datatyperna.

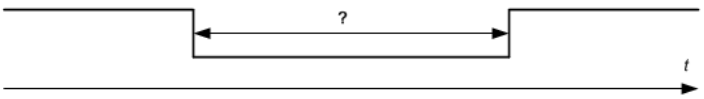
Exempel: Följande kodsekvens:

```
num = 0xFFF;
even = bitcheck( &num, &result);
if( even )
   printf ( "Even (%d) number of zeroes", result );
else
   printf ( "Odd (%d) number of zeroes", result );
```

ger utskriften: `Even (20) number of zeroes`

Uppgift 3 (6p)

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

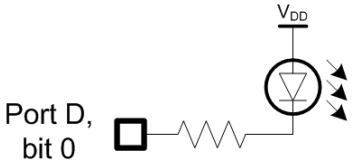


För detta ansluter vi signalledaren till portpinne PC4 hos *MD407*. De externa funktionerna `void start_M(void)` för att starta en mätning, och `void stop_M(void)` för att avsluta mätningen finns givna.

a) Visa hur SYSCFG konfigureras så att PC4 kopplas till EXTI-modulen hos *MD407*. Tänk på att bara PC4 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 negativ flank hos signalledaren, och avslutar en pågående mätning vid en positiv flank, genom att anropa de givna funktionerna. Systemets vektortabell startar på adress `0x2001C000`, visa också hur avbrottsvektorn initieras. (3p)

Uppgift 4 (10p)

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_250us( void);`.
Systemets klockfrekvens är 168 MHz. (6p)
b) Skriv en funktion `void init_app( void );` som sätter upp port D, bit 0 som utsignal, ”push-pull”. Övriga inställningar för porten ska behållas, dvs. får inte ändras av denna initiering. (2p).
c) Skriv huvudprogrammet som får dioden att blinka. Du får använda `delay_250us()` och `init_app()` även om du inte besvarat a) och/eller b). (2p)

Uppgift 5 (10p)

En enhet för direkt överföring till minnet (DMA, *direct memory access*), med följande gränssnitt, är ansluten till *MD407* på adress `0xF0008000`.

Registeruppsättning

offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register	Åtkomst av register CTRL och STAT måste vara *byte*-format Åtkomst av register ADDR_LOW kan vara word-format och ger då 32 bitars adress (little endian format)
0																	CTRL	
2																	STAT	
4																	ADDR_LOW	
6																	ADDR_HIGH	
8																	SIZE	

CTRL (*Control*) Styrregister, samtliga bitar är läs- och skrivbara.

offset	7	6	5	4	3	2	1	0	EX: (*Execute*), starta utförande av DMA-överföring. TRIG: (*Trigger*), villkor för att starta överföringen. IRQPRI: (*Interrupt priority*), 0 är högsta prioritet, 0xF är ”inga avbrott”.
0x00	IRQPRI				TRIG			EX	

STAT (*Status*) Statusregister, samtliga bitar är läsbara.

offset	7	6	5	4	3	2	1	0	BUSY: (*Busy*), upptagen med att utföra kommando. Sätts till 1 av hårdvaran då EX sätts till 1. Återställs automatiskt då överföringen är klar. IRQ: (*Interrupt Request*), sätts till 1 av hårdvara då BUSY blir 0. Återställs då BUSY sätts till 1 av hårdbvara. FAIL: (Failure): Fel har uppstått, beskrivs av FAILURECODE. Återställs automatiskt då EX sätts till 1. FAULURECODE: då FAIL=1 innehåller fältet en kod som beskriver felet.
0x02	FAILURECODE					FAIL	IRQ	BUSY	

ADDR_LOW (Address Low)

offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x04	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

Registret håller de 16 minst signifikanta bitarna (A15..A0) av den 32-bitars adressen för direkt överföring av helt block (1kByte) till eller från minnet.

ADDR_HIGH (Address High)

offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x06	A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16

Registret håller de 16 mest signifikanta bitarna (A31-A16) av den 32-bitars adressen för direkt överföring av helt block (1kByte) till eller från minnet.

SIZE, samtliga bitar är läs- och skrivbara.

offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	NUMBLOCKS, heltal utan tecken, antalet minnesblock (1024kByte) vid överföringen
0x08	NUMBLOCKS																

a) Visa en lämplig typdeklaration (*struct*-format) av porten. Deklarationen ska tillåta att DMA-adress ges på såväl 32-bitars, som 16-bitars format. (4p).
b) Använd deklarationen i a) och visa hur följande initiering kodas i ’C’. (2p)

```
ADDR<- 0x20010000
SIZE<- 0x24
TRIG<- 4
IRQPRI<- 0xF
EX<- 1
```

c) Visa en lämplig typdeklaration (*struct*-format) med bitfältsdeklarationer för fälten i portens register (4p)

Lösningsförslag

Uppgift 1 a (2p)

```
v:    .space   (10*20)*4
i:    .space   4
j:    .space   4
```

Uppgift 1 b (4p)

```
&v[i]= &v[0]+(((i*20)+ j ) * sizeof(int) )
```

Uppgift 1 c (4p)

```
    LDR   R0,=j
    LDR   R0,[R0]
    LDR   R1,=v
    BL    f
    LDR   R1,=k
    STRH  R0,[R1]
```

Uppgift 1 d (6p)

```
f:  PUSH  {R4,LR}      @ R4 och LR spills i funktionen
    MOV   R4,R0        @ kopia av parameter till R4
    ADD   R0,R0,#5
    BL    o            @ R0<-o(a+5)
    CMP   R4,R0        @ a > R0 ?
    BGT   .L1
    MOV   R0,#0        @ return 0
    B     .L2
.L1:
    MOV   R0,#1        @ return 1
.L2:
    POP   {R4,PC}
```

Uppgift 2 (8p):

```
short   bitcheck( unsigned int *p, int *num)
{   /* Actually simpler to count number of ones...
        then determine number of zeroes as:
        Size of an integer *8 – number of ones */
    *num = 0;
    while(*p)
    {
        if( *p & 1 ) (*num)++;
        *p >>= 1;
    }
    *num = (short) ((sizeof(int) * 8) - *num);
    return !(*num & 1); /* 0 if odd */
}
```

Uppgift 3

Macro-definitioner för register

```
#define PORT_C_BASE     0x40020800
#define GPIO_C_MODER    ((volatile unsigned int  *) (PORT_C_BASE))
#define GPIO_C_IDR      ((volatile unsigned char *) (PORT_C_BASE+0x11))
#define SYSCFG_EXTICR2  ((volatile unsigned int  *) 0x4001380C )
#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 EXTI4_IRQVEC    0x2001C068 (Räcker med offset 0x64 för full poäng)
#define EXTI4_EXTI_BITNO    4
#define EXTI4_NVIC_BITNO    10
#define NVIC_ISER0      ((volatile unsigned int  *) 0xE000E100 )
#define NVIC_ICPR0      ((volatile unsigned int  *) 0xE000E280 )
#define NVIC_IPR0       ((volatile unsigned int  *) 0xE000E400 )
```

Uppgift 3a (1p)

```
  *GPIO_C_MODER &= 0xFFFFFCFF;   /* PC4, inport */
  /* Antag avbrottssignal PUSH/PULL, annars kan PULL DOWN läggas på PC4 , behövs dock inte här för full poäng)
  *SYSCFG_EXTICR2 &= 0xFFF0;
  *SYSCFG_EXTICR2 |= 0x0002;     /* PC4->EXTI4  */
```

Uppgift 3b (2p)

```
  *EXTI_IMR   |= (EXTI4_EXTI_BITNO);
  *EXTI_FTSR  |= (EXTI4_EXTI_BITNO);
  *EXTI_RTSR  |= (EXTI4_EXTI_BITNO);
  *NVIC_ISER0 |= (EXTI4_NVIC_BITNO);
```

Uppgift 3c (3p)

```
void at_irq ( void )
{
   if( *GPIO_C_IDR & (1<<4) )    /* Signal high ? */
     stop_M();                   /* end measure */
   else
     start_M();                  /* Start measure */
   *EXTI_PR |= (1<<3);           /* Acknowledge interrupt */
}
```

Initiering av avbrottsvektor:

```
  *((void (**)(void) ) EXTI4_IRQVEC ) = at_irq;
```

Uppgift 4a (6p)

```
#define     SysTickCtrl ((volatile unsigned int *)(0xE000E010))
#define     SysTickLoad ((volatile unsigned int *)(0xE000E014))
#define     SysTickVal      ((volatile unsigned int *)(0xE000E018))
void delay_250us( void )
{
    /* SystemCoreClock = 168000000 */
    *SysTickCtrl = 0;
    *SysTickLoad = ( (168000/4)-1 );
    *SysTickVal = 0;
    *SysTickCtrl = 5;
    while( (*SysTickCtrl & (1<<16) )== 0 );
    *SysTickCtrl = 0;
}
```

Uppgift 4b (2p)

```
#define     PORT_D_BASE     0x40020C00
#define     GPIO_D_MODER    ((volatile unsigned int  *) (PORT_D_BASE))
#define     GPIO_D_OTYPER   ((volatile unsigned int  *) (PORT_D_BASE+4))
#define     GPIO_D_OUT_LOW  ((volatile unsigned int  *) (PORT_D_BASE+0x10))
void init_app( void )
{
    /* PORT D */
    * GPIO_D_MODER      &= ~3;  /* reset bit 0 mode */
    * GPIO_D_MODER      |= 1;   /* set bit 0 as output */
    * GPIO_D_OTYPER &=  ~1;     /* reset bit 0 type to push/pull */
}
```

Uppgift 4c (2p)

```
void main(void)
{
    int i;
    init_app();
    /* 0,5 sec= 500 ms = 2000*250us */
    while(1)
    {
        * GPIO_D_OUT_LOW = 0;
        for(i=0;i<2000;i++) delay_250us();
        * GPIO_D_OUT_LOW = 0xFF;
        for(i=0;i<2000;i++) delay_250us();
    }
}
```

Uppgift 5a (4p)

```
struct dma{
  volatile unsigned char ctrl;
  volatile unsigned char sec;
  union{
    volatile unsigned long addr;
  struct{
    volatile unsigned short addr_low;
    volatile unsigned short addr_high;
    }
  };
  volatile unsigned short size;
};
```

Uppgift 5b (2p)

```
    ((struct dma *) 0xF0008000)->addr = 0x21000000;
    ((struct dma *) 0xF0008000)->size = 24;
    ((struct dma *) 0xF0008000)->ctrl = (4<<1)|(0xF<<4)|1;
```

Uppgift 5c (4p)

```
struct dma {
  volatile unsigned char ex:1, trig:3, irqpri:4;
  volatile unsigned char :0, busy:1,irq:1,fail:1,failurecode:5;
  union{
    volatile unsigned long addr;
  struct{
    volatile unsigned short addr_low;
    volatile unsigned short addr_high;
    }
  };
  volatile unsigned short size;
};
```