Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA

Tentamen med lösningsförslag

EDA482 Maskinorienterad programmering D
EDA487 Maskinorienterad programmering Z
DIT151 Maskinorienterad programmering GU

Torsdag 2 juni 2022, 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 (”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 deklarationerna:

```
unsigned short a,b,*c;
```

Visa hur variabeldeklarationerna kodas i assembler i ARMv6 assemblerspråk. För funktionen `fcall` gäller deklarationen:

```
short fcall( short, short *);
```

visa också hur följande funktionsanrop kodas i ARMv6 assemblerspråk (6p):

```
a = fcall(b,c);
```

(b) Vi har deklarationerna:

```
int x,y,z, v[64];
```

Visa en kodsekvens i ARMv6 assemblerspråk där resultatet av uttrycket `x*(y+z)-v[x]`evalueras 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).

```
char * skipUntilSpace( char *s )
{
   while( s && *s && (*s != 0x20) )
   {
      s++;
   }
   return s;
}
```

Uppgift 2 (10p)

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).

c) Visa hur portens register CREG kan avbildas med en bitfältdeklaration där de olika bitfälten kan refereras var för sig. Visa speciellt hur fältet f1 refereras (4p).

Uppgift 3 (8p)

Konstruera en C-funktion `unsigned int distance( char c, char *s)` som bestämmer avståndet mellan två likadana tecken (ges av parameter `c`) i en noll-terminerad textsträng (ges av parametern `s`). Det får förutsättas att textsträngen inte innehåller fler än två instanser av tecknet. Om textsträngen innehåller färre än två instanser ska funktionen returnera värdet 0.

Uppgift 4 (8p)

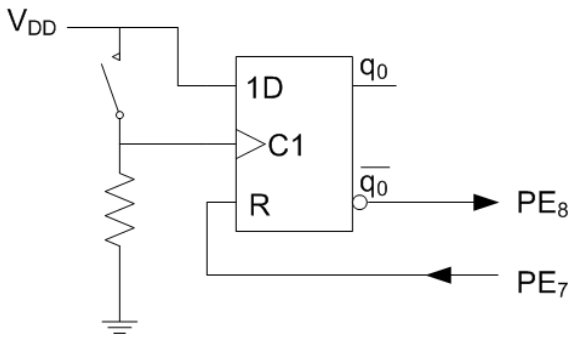
Man vill skapa en tidtagarfunktion, med upplösningen 1 ms. Det får antas att mättiderna aldrig överskrider 10 sekunder. Visa hur en sådan tidtagarfunktion kan implementeras i ett *MD407* laborationssystem, med modulen SYSTICK. Laborationssystemets klockfrekvens är 168 MHz. Följande funktioner, ska finnas:

- `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 5 (8p)

En D-vippa är ansluten till två portpinnar hos en *MD407* laborationsdator enligt följande:



Vi påminner kortfattat om D-vippans funktion: Vid positiv flank hos C1 kommer vippans tillstånd q0 att anta värdet från 1D. Man vill detektera då strömställaren sluts och detta ska ske med användning av avbrott.

a) Visa en initieringsrutin `void init( void )` där:

- GPIO modulen initieras för dessa portpinnar, observera att endast konfigurationen för dessa portpinnar får ändras vid konfigureringen.
- SYSCFG, EXTI och NVIC konfigureras
- Avbrottsvektor initieras med adress till avbrottsfunktionen `void at_interrupt(void)`.

b) Visa avbrottsrutinen `void at_interrupt( void )` som kvitterar avbrottet och återställer D-vippan.

Observera: speciellt gäller att bara dessa portpinnar och dess konfigurationer får påverkas av din kod. Eventuella tidigare initieringar (utöver portpinnar 7 och 8) får inte påverkas.

Lösningsförslag

Uppgift 1 a

```
   .align
a: .space 2
b: .space 2
c: .space 4

   LDR   R0,=b
   LDRH  R0,[R0]
   LDR   R1,=c
   LDR   R1,[R1]
   BL    fcall
   LDR   R1,=a
   STRH  R0,[R1]
```

Uppgift 1 b

```
   LDR   R0,x     @ R0= x
   LDR   R1,y     @ R1= y
   LDR   R2,z     @ R2= z
   ADD   R1,R1,R2 @ R1= y+z
   MUL   R0,R1    @ R0= x*(y+z)
   LDR   R1,=v    @ R0= &v
   LDR   R2,x     @ R2= x
   LSL   R2,R2,#2 @ R2= x*sizeof(int)
   ADD   R1,R1,R2 @ R1= &v+x*sizeof(int)
   LDR   R1,[R1]  @ R1= v[x*sizeof(int)]
   SUB   R0,R0,R1 @ R0= x*(y+z)- v[x]
```

Uppgift 1 c

```
@ Parameter s i register R0
@ Returnerar modifierad s i R0
skipUntilSpace:
   CMP   R0,#0                @ while( s ..)
   BEQ   skipNonBlancExit
   LDRB  R1,[R0]
   CMP   R1,#0                @ while( .. s* ..)
   BEQ   skipNonBlancExit
   CMP   R1,#0x20             @ while( .. s* != 0x20 .. )
   BEQ   skipNonBlancExit
   ADD   R0,R0,#1             @ s++;
   B     skipUntilSpace       @ while(..)
skipNonBlancExit:
   BX   LR                    @ return s
```

Uppgift 2 a

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

Uppgift 2 b

```
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 2 c

```
struct creg {
    volatile unsigned int f0:4;
    unsigned int :3;
    volatile unsigned int f1:6;
};
Referens: ((struct creg *) 0xFF000000)->f1;
```

alternativt:

```
#define CREG ( (volatile struct creg *) 0xFF000000)
CREG->f1;
```

Uppgift 3

```
unsigned int distance( char c, char *s )
{
  char *t;

  while( *s && *s != c ) s++;
  if( *s == 0 ) return 0;
  t = s+1;
  while( *t && *t != 0 ) t++;
  if( *t == 0 ) return 0;
  return (unsigned int) (t-s);
}
```

Uppgift 4

```
#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 5

```
#define   GPIO_MODER    ((volatile unsigned int *)   0x40021000)
#define   GPIO_OTYPER   ((volatile unsigned short *) 0x40021004)
#define   GPIO_ODR      ((volatile unsigned short *) 0x40021014)
#define   GPIO_IDR      ((volatile unsigned short *) 0x40021010)

/* PE8 (EXTI8, IRQ num 23 */
#define SYSCFG_EXTICR3       0x40013810
#define NVIC_EXTI8_IRQ_BPOS  (1<<23)
#define EXTI8_IRQ_BPOS       (1<<8)
#define EXTI8_IRQVEC         0x2001C09C
#define NVIC_ISER0           0xE000E100

void init( void )
{
   *GPIO_MODER  |= 0x00004000;  /* bit 7 är utgång */
   *GPIO_MODER  &= ~0x00030000; /* bit 8 är ingång */
   *GPIO_OTYPER &= ~0x00000080; /* output: push/pull */
   *((unsigned int *) SYSCFG_EXTICR3) &= 0x~000F;
   *((unsigned int *) SYSCFG_EXTICR3) |= 0x0004;     /* PE8->EXTI3  */
   *((unsigned int *) EXTI_IMR) |= EXTI8_IRQ_BPOS;
   *((unsigned int *) EXTI_FTSR) |= EXTI8_IRQ_BPOS;   /* trigger på negativ flank */
   *((unsigned int *) EXTI_RTSR) &= ~EXTI8_IRQ_BPOS;
   *((unsigned int *) NVIC_ISER0) |= NVIC_EXTI3_IRQ_BPOS;    /* Tillåt detta avbrott */
   *((void (**)(void) ) EXTI8_IRQVEC ) = at_interrupt;
}
void at_interrupt( void )
{   /* Avbrott för alla EXTI 5 t.o.m 9 */
   if(   *((unsigned int *) EXTI_PR) & EXTI8_IRQ_BPOS )
   {  /* EXTI8 avbrott */
      *((unsigned int *) EXTI_PR) |= EXTI8_IRQ_BPOS;   /* Kvittera avbrott */
      *GPIO_ODR  |= 0x80;  /* Återställ D-vippa */
      *GPIO_ODR  &= ~0x80;
   }
}
```