Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA

Hemtentamen med lösningsförslag

DAT390 Maskinorienterad programmering D,E,ME (hing)

Måndag 16 mars 2019, kl. 14.00 - 18.00 (+10min)

Examinatorer
Pedro Trancoso

Kontaktpersoner under tentamen:
Pedro Trancoso, epost (Canvas)

Tillåtna hjälpmedel

Alla hjälpmedel är tillåtna så länge uppgifterna löses på individuell basis utan att konsultera någon annan än examinator under skrivningstiden.

Speciellt:
För uppgifterna 2-4 krävs, utöver kod, *beskrivningar* av vad som görs och hur koden är tänkt att fungera.

För uppgifterna 1, 5 och 6 krävs *väl dokumenterad* programkod.

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: (EDA/DAT/LEU):
20p ≤ betyg 3 < 30p ≤ betyg 4 < 40p ≤ betyg 5. För betyg 5 krävs dessutom samt nöjaktigt svar på uppgift 7.

respektive (DIT):
20p≤ betyg G < 35p ≤ VG, för betyg VG krävs dessutom nöjaktigt svar på uppgift 7.

Uppgift 1 (6p)

(a) Vi har deklarationerna: `char j; int k;` Visa en kodsekvens, i ARMv6 assemblerspråk, som evaluerar uttrycket `j+(2*k)-1` till register R0. (2p)

(b) I funktionen `g` har vi registerallokeringen:
`R4: lokal variabel my_g`
`R5: lokal variabel my_rv`
I funktionen `void g(int param)` görs följande funktionsanrop:
`my_rv = f( my_g, param );`
Visa hur detta då kodas i ARMv6 assemblerspråk. (2p)

(c) Visa koden för allokeringen av dessa fyra variabler med det minsta möjliga minnesutrymme (2p)

```
int i1;
char c1;
short s1;
unsigned char c2;
```

Uppgift 2 (4p)

Följande funktion, `meanvalue`, bestämmer medelvärdet av tal i en vektor:

```
unsigned int meanvalue( int n, unsigned short *v )
{
   unsigned int sum; int i;
   if( n <= 0 )
      return 0xFFFF;
   sum = 0;
   for( i = 0; i < n; i++ )
   {
      sum += (unsigned int) v[i];
   }
   return sum/n;
}
```

Antag att instruktionsuppsättningen även innehåller en divisionsinstruktion:
`DIVU Rx,Rx,Ry`, som utför heltalsdivision (tal utan tecken) Rx ← Ry/Rx och koda funktionen i ARMv6 assemblerspråk.

Uppgift 3 (8p)

En funktion för manipulering av bitar i en operand ska konstrueras. Funktionen deklareras:

```
unsigned int   bitconvert(int method, unsigned int value);
```

Returvärdet bestäms av parametern `value` och de två minst signifikanta bitarna i parametern `method` (oberoende av parametern i övrigt) enligt:
`method` (b_1b_0)`=00`: returnera *bitvis and* av `value` med dina första 8 siffror från ditt persunnummer (0xYYYYMMDD).
`method` (b_1b_0)`=01`: returnera *bitvis komplementet* av `value`.
`method` (b_1b_0)`=10`: returnera de 16 mest signifikanta bitarna av `value`.
`method` (b_1b_0)`=11`: returnera det 32-bitars tal som bildas från det 16 bitars teckenutvidgade extraherade tal från b_{24}-b_8 hos `value`.
För denna uppgift gäller att `sizeof(int)` är 4.

Uppgift 4 (6p)

En funktion `CircularShiftRight` permuterar textsträngar i en vektor, Följande exempel visar hur funktionen kan användas:

```
char *list[]={"Hello","World","Again"};
int main(void)
{
   printf( "%s %s %s\n", list[0],list[1],list[2] );
   CircularShiftRight  ( &list[0], &list[1], &list[2]);
   printf( "%s %s %s\n", list[0],list[1],list[2] );
}
```

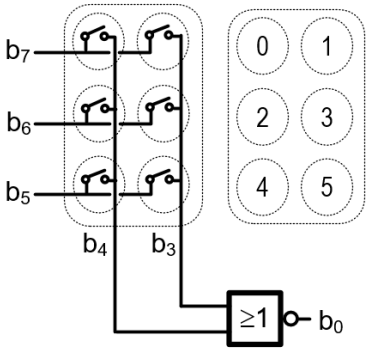
ger utskriften:

```
Hello World Again
Again Hello World
```

Visa hur funktionen `CircularShiftRight` kan implementeras i C.

Uppgift 5 (22p)

Ett tangentbord för inmatning av sex olika tecken ska konstrueras. Sex stycken återfjädrande omkopplare ansluts därför till port E hos *MD407*, enligt figuren till höger. En nedtryckt tangent ska representera ett binärt värde 0-5.

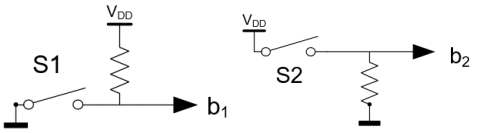


Kretsen kan också användas för att känna av en tangentnedtryckning med hjälp av avbrott. För detta kopplas kolumnerna till en NOR-grind, vars utgång är kopplad till bit 0 hos port E.

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'.

Nedtryckta tangenter kan detekteras genom att '1' skrivs till någon av bit 7, bit 6 eller bit 5, därefter avläses bit 4 respektive bit 3. För att detta ska vara tillförlitligt måste dessa bitars ingångar förses med "*pull-down*" samtidigt som utgångarna ska vara "*push-pull*".

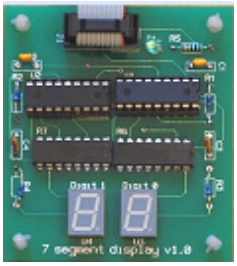
Utöver tangentbordet ansluts också två återfjädrande strömställare, S1 ("Unlock keyboard") med "pull-up"-resistor och S2 ("Lock keyboard") med "pull-down"-resistor enligt figuren till höger.



Dessa används för att generera ytterligare avbrott.

Visningsenhet

Utöver detta ansluts en utmatningsenhet för visning av två hexadecimala siffror till port E b8-b15.



För full poäng krävs att dina lösningar är tydliga, fullständiga och att du använt *lämpliga makrodefinitioner för registeradresser*.

a) Visa med en funktion `void gpio_init(void)` hur port E ska initieras för användning med tangentbord och visningsenhet. (3p)

b) Visa en funktion `void init_exti(void)` som konfigurerar så att bit 0, bit 1 och bit 2 kan kopplas till avbrottssystemet hos *MD407* via EXTI-modulen. Avbrott ska ske då omkopplarna sluts, betrakta kopplingarna i figurerna ovan och fundera på vilken flank avbrotten ska ske på. De tre avbrotts-funktionerna namnges `KeybIRQ`, `S1IRQ`, `S2IRQ`. Vektortabellen börjar på adress 0x2001C000 (6p)

c) Visa en tangentbordsfunktion `unsigned char get(void)`, som returnerar tangentbordets tillstånd. Detta kan vara 0-5 om en tangent tryckts ned, eller 0xF om tangentbordet är låst. (6p)

d) En tangentnedtryckning ska resultera i ett avbrott via b0. Strömbrytare S2 används för att 'låsa' tangentbordet med ett avbrott via b2. I låst tillstånd ignoreras alla tangentnedtryckningar. Strömbrytare S1 används för att 'låsa upp' tangentbordet med ett avbrott via b1. I upplåst tillstånd registreras och sparas den senast nedtryckta tangentens kod. Visa de tre avbrottsfunktionerna `KeybIRQ`, `S1IRQ`, `S2IRQ` och ange dessas avbrottsvektorer. Du får använda funktionen `get` från uppgift c även om du inte löst uppgiften. *Ledning*: Använd en statisk variabel `lastkey`, för att lagra den senaste tangenten och en variabel `locked` för tangentbordets tillstånd. (3p)

e) En enkel utmatningsenhet för visning av två hexadecimala siffror, i form av en `char` ska användas som indikator och ansluts därför till port E, bit 8-15. Skriv ett huvudprogram som kontinuerligt skriver ut information från tangentbordet (du får använda lösningar från tidigare uppgifter även om du inte gjort dessa). Om tangentbordet är upplåst ska senast nedtryckta tangent (0-5) visas, om tangentbordet är låst ska siffran 0xF visas. Tangentbordet ska initialt vara låst och b7,b6,b5 satta till 1 så att avbrott kan detekteras. (4p)

Uppgift 6 (4p)

Antag att en bild på 400x200 pixlar, där varje “pixel” är 8 bitar (00000000 till 11111111) och lagrats med start på minnesstartadress 0x80000000. Visa en kodsekvens i ARMv6 assemblerspråk som går över alla ”pixlar” och summerar i register R0 hur många ”pixlar”, av dessa som har värde större än 128.

Uppgift 7 (För bedömning av överbetyg)

Förbered en beskrivning av en ny laboratorieuppgift (t.ex. kan baseras på ett av de avancerade ämnena som byggnadsbibliotek, dynamisk minnesallokering eller ett specifikt spel). Du bör presentera en god beskrivning av problemet och vad du förväntar dig att eleverna ska lösa. Efter det bör du också presentera en översikt över svaret som du förväntar dig att eleverna ska göra för att få full poäng.

Lösningsförslag

Uppgift 1 a

```
    LDR     R0,=j
    LDRB    R0,[R0]     @ R0←j
    SXTB    R0, R0
    LDR     R1,=k
    LDR     R1,[R1]     @ R1←k
    LSL     R1,R1,#1    @ R1←(2*k)
    ADD     R0,R0,R1    @ R0←j+(2*k)
    SUB     R0,R0,#1    @ R0←j+(2*k)-1
```

Anm: I stället för

```
    LDR R0,=X
    LDR R0,[R0]     @ R0←X
```

kan

```
    LDR R0,X
```

användas

Uppgift 1 b

```
    MOV R1,R0
    MOV R0,R4
    BL  f
    MOV R5,R0
```

Uppgift 1 c

```
    i1: .SPACE 4
    c1: .SPACE 1
    c2: .SPACE 1
    s1: .SPACE 2
```

Uppgift 2:

```
@ subrutin meanvalue
@ registeranvändning:
@ R0: n
@ R1: v
@ R2: sum
@ R3: i
@ R4: tempregister
meanvalue:
        PUSH    {R4}
        CMP     R0,#0
        BGT     .L1             @ (n <= 0)?
        LDR     R0,=0xFFFF
        B       .L2             @ return 0xFFFF;
.L1:    MOV     R2,#0           @ sum = 0;
        MOV     R3,#0           @ i = 0;
.L3:    CMP     R3,R0
        BGE     .L4             @ (i < n)?
        LSL     R4,R3,#1        @ R4←i*sizeof(short)
        ADD     R4,R4,R1        @ R4←&v+i*sizeof(short)
        LDRH    R4,[R4]         @ R4← (unsigned int) (v+i*sizeof(short) )
        ADD     R2,R2,R4        @ sum = sum + (unsigned int) (v+i*sizeof(short) )
        ADD     R3,R3,#1        @ i++
        B       .L3

.L4:    DIVU    R0,R0,R2        @ R0←sum/n
.L2:    POP     {R4}
        BX      LR
```

Uppgift 3:

```
unsigned int   bitconvert(int method, unsigned int value)
{
    unsigned int rv;
    switch( method & 3 )
    {
        case 0: rv = value; break;
        case 1: rv = ( value ^ 0xFFFFFFFF ); break;  // or rv = ~value;
        case 2: rv = ( value >> 16 ); break;
        case 3: rv = ( (value >>8) & 0xFFFF);
                if (rv & 0x8000 )
                    rv = (rv | 0xFFFF0000 );
                break;
    }
    return rv;
}
```

Uppgift 4:

```
Void CircularShiftRight(char** p1, char **p2, char **p3)
{
    char *p = *p3;
    *p3=*p2;
    *p2=*p1;
    *p1=p;
}
```

Uppgift 5 a

```
/* Port E */
#define    PORT_E_BASE    0x40021000
#define    GPIO_E_MODER   ((volatile unsigned int *)(PORT_E_BASE))
#define    GPIO_E_OTYPER  ((volatile unsigned short *)(PORT_E_BASE+0x4))
#define    GPIO_E_OSPEEDR     ((volatile unsigned int *)(PORT_E_BASE+0x8))
#define    GPIO_E_PUPDR   ((volatile unsigned int *)(PORT_E_BASE+0xC))
#define    GPIO_E_IDRLOW  ((volatile unsigned char *)(PORT_E_BASE+0x10))
#define    GPIO_E_IDRHIGH     ((volatile unsigned char *)(PORT_E_BASE+0x11))
#define    GPIO_E_ODRLOW  ((volatile unsigned char *)   (PORT_E_BASE+0x14))
#define    GPIO_E_ODRHIGH     ((volatile unsigned char *)   (PORT_E_BASE+0x15))
void init_gpio( void )
{
    /*  PORT E
    b15-b8  used for hexadecimal display
    b7-b5   used for output to rows
    b4-b0   used for input from columns, NOR gate and switches
    */
    *GPIO_E_MODER = 0x55555400;
    *GPIO_E_PUPDR = 0x000002A4;       /* Input, b4, b3 and b2 pull down */
    *GPIO_E_OTYPER= 0x00000000;       /* outputs are push/pull */
}
```

Uppgift 5 b

```
#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    NVIC_ISER0     ((volatile unsigned int *)0xE000E100)

#define    NVIC_EXTI2_IRQ_BPOS  (1<<8)
#define    NVIC_EXTI1_IRQ_BPOS  (1<<7)
#define    NVIC_EXTI0_IRQ_BPOS  (1<<6)

#define    EXTI2_IRQ_BPOS  (1<<2)
#define    EXTI1_IRQ_BPOS  (1<<1)
#define    EXTI0_IRQ_BPOS  (1<<0)

#define    EXTI2_IRQVEC    0x2001C060
#define    EXTI1_IRQVEC    0x2001C05C
#define    EXTI0_IRQVEC    0x2001C058

void init_exti( void )
{
    *SYSCFG_EXTICR1|= 0x0444;     /* PE2..PE0->EXTI2--EXTI0  */
    /* Configure the mask bit of the interrupt line (EXTI_IMR) */
    *EXTI_IMR |= (EXTI2_IRQ_BPOS |  EXTI1_IRQ_BPOS | EXTI0_IRQ_BPOS );
    /* Configure the Trigger selection bit of the interrupt line (EXTI_RTSR and EXTI_FTSR) */
    *EXTI_FTSR |= (EXTI1_IRQ_BPOS | EXTI0_IRQ_BPOS );      /* enable trigger on falling edge */
    *EXTI_RTSR &= ~(EXTI1_IRQ_BPOS | EXTI0_IRQ_BPOS );     /* disable trigger on rising edge */
    *EXTI_RTSR |= EXTI2_IRQ_BPOS ;       /* enable trigger on rising edge */
    *EXTI_FTSR &= ~EXTI2_IRQ_BPOS;       /* disable trigger on falling edge */
    /* set the interrupt vectors */
    *((void (**)(void) ) EXTI0_IRQVEC)  = KeybIRQ;
    *((void (**)(void) ) EXTI1_IRQVEC)  = S1IRQ;
    *((void (**)(void) ) EXTI2_IRQVEC)  = S2IRQ;
    /* Enable these interrupts */
    *NVIC_ISER0 |= (NVIC_EXTI2_IRQ_BPOS | NVIC_EXTI1_IRQ_BPOS | NVIC_EXTI0_IRQ_BPOS );
}
```

Uppgift 5 c

```
static char lastkey;
unsigned char get( void )
{
    unsigned short c;
    *GPIO_E_ODRLOW = 0x80 ;
    c = *GPIO_E_IDRLOW & 0x18;
    if ( c & 0x10 ){ lastkey = 0; return;}
    if ( c & 8 ){ lastkey = 1; return;}
```

```
    *GPIO_E_ODRLOW = 0x40 ;
    c = *GPIO_E_IDRLOW & 0x18;
    if ( c & 0x10 ){ lastkey = 2; return;}
    if ( c & 8 ){ lastkey = 3; return;}
    *GPIO_E_ODRLOW = 0x20 ;
    c = *GPIO_E_IDRLOW & 0x18;
    if ( c & 0x10 ){ lastkey = 4; return;}
    if ( c & 8 ){ lastkey = 5; return;}
}
```

Uppgift 5 d

```
static char locked;
void KeybIRQ ( void )
{
    lastkey = get();
    *GPIO_E_ODRLOW = 0xE0 ;       /* Activate keypad */
    *EXTI_PR = EXTI0_IRQ_BPOS;   /* Clear this interrupt */
}
void S1IRQ ( void )
{
    *EXTI_PR = EXTI1_IRQ_BPOS;   /* Clear this interrupt */
    locked = 0;
}
void S2IRQ ( void )
{
    *EXTI_PR = EXTI2_IRQ_BPOS;   /* Clear this interrupt */
    locked = 1;
}
```

Uppgift 5 e

```
void main(void)
{
    init_gpio();
    init_exti();
    locked = 1;
    *GPIO_E_ODRLOW = 0xE0 ;       /* Activate keypad */
    while( 1 )
    {
        if( locked ) *GPIO_E_ODRHIGH = 0xF;
        else *GPIO_E_ODRHIGH = lastkey;
    }
}
```

Uppgift 6

```
            LDR R1,=0x80000000 @ starting address for the image
            LDR R2,=80000      @ size in pixels of the picture
            MOV R3,#0          @ iteration conter and offset from starting address, initialized to 0
            MOV R0,#0          @ counter for values larger than 128, initialized to 0
            MOV R4,#128        @ 128 value to be used for compare
    .LOOP:  CMP R3,R2
            BEQ .EXIT
            LDR R5,[R1,R3]
            CMP R5,R4
            BSE .SKIP
            ADD R0,R0,#1
    .SKIP:  ADD R3,R3,#1
            B .LOOP
    .EXIT:
```