Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA

Tentamen med lösningsförslag

EDA482 (EDA481) Maskinorienterad programmering D
EDA487 (EDA486) Maskinorienterad programmering Z
DAT017 (DAT016) Maskinorienterad programmering IT
DIT151 Maskinorienterad programmering GU
LEU500 Maskinorienterad programmering D,E,ME (hing)

Fredag 24 augusti, kl. 8.30 - 12.30

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 dessa häften.

Tabellverk eller miniräknare får ej användas.

Lösningar
anslås senast dagen efter tentamen via kursens hemsida.

Granskning
Tid och plats anges på kursens hemsida.

Allmänt
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):
$20p \leq$ betyg $3 < 30p \leq$ betyg $4 < 40p \leq$ betyg 5
respektive (DIT):
$20p \leq$ betyg $G < 35p \leq$ VG

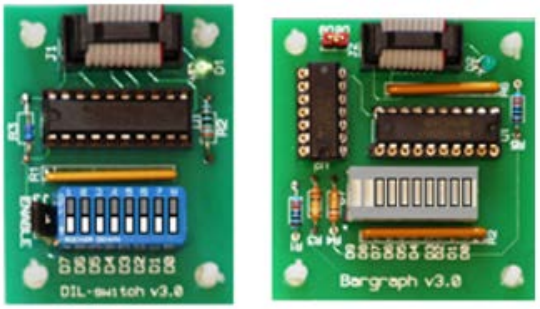
Uppgift 1 (10p)

a) (4p) Visa hur adressen till uttrycket: `vs[i+j+2]` evalueras till register R0 i ARM v6 assemblerspråk då följande deklarationer gjors på ”toppnivå”: `int, i,j; short vs[54];`

b) (6p) Följande funkion är given i C, visa hur funktionen kan kodas i ARM v6 assemblerspråk. Använd GCC:s kompilatorkonventioner.

```
int f( int p)
{
  int rv = reval( p << 2 );
  if( rv > 0) return rv | 1;
  if( rv < 0) return rv & 1;
  return 0;
}
```

Uppgift 2 (15p)



En 8-bitars omkopplare ansluts till port PE0-PE7 hos MD407.
En diodramp ansluts till PE8-PE15.

Bit 7 från omkopplaren ska anslutas till processorns avbrottsmekanism via EXTI-modulen, bit3-bit0 hos omkopplaren utgör ett 4-bitars ord (INP4).
Då bit 7 slås till (0→1) ska INP4 läsas från omkopplaren och teckenutvidgas till `unsigned char`.
Då bit 7 slås från (1→0) ska INP4 läsas från omkopplaren och teckenutvidgas till `signed char`.
Övergångarna ska detekteras med avbrottsmekanismen. Det får förutsättas att inga kontaktstudsar förekommer. Det teckenutvidgade värdet ska skrivas till diodrampen vid varje ändring hos bit 7.

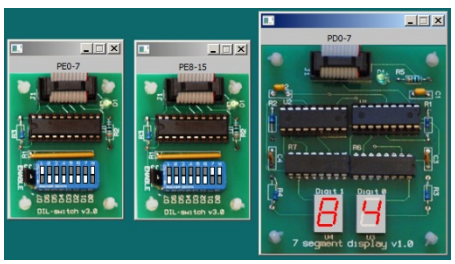
a) (4p) Visa med en funktion `void app_init(void)` hur port E ska initieras för applikationen. Ingångar ska vara ”flytande” och utgångar av typ ”push/pull”.

b) (5p) Visa med en funktion `void irq_init(void)` hur SYSCFG/EXTI/NVIC modulerna ska konfigureras för användning enligt uppgiftens beskrivning.

c) (6p) Det konverterade värdet från INP4 ska visas på diodrampen. Uppdatering ska ske vid varje ändring av bit 7, dvs. vid varje avbrott. Visa ett huvudprogram `void main(void)`, och en avbrottsfunktion `void irq_handler(void)`.

För full poäng krävs att dina lösningar är tydliga, fullständiga och att du använt lämpliga makrodefinitioner för registeradresser. Samla alla sådana definitioner under uppgift a).

Uppgift 3 (6p)



Två DIL-strömbrytare och en 7-segments indikator kopplas enligt:

- DIL1 till Port E (0-7),
- DIL2 till Port E (8-15)
- Display till port D (0-7):

Skriv ett program i ARM/Thumb assemblerspråk som:

- Initierar portar D och E.
- Kontinuerlig läser de inställda värdena från DIL-strömbrytarna, adderar dessa och skriver resultatet till displayen. Om summan är större än 255 ska FF skrivas till indikatorn.

Uppgift 4 (8p)

Visa en funktion `swap` som skiftar två pekare i minnet. Om det exempelvis från början gäller att `a` och `b` är pekare ska funktionen anropas enligt `swap( &a, &b )` och vi dessutom har att:
`*a = '1'` och `*b = '2'`,
ska det efter funktionen gälla att
`*b = '1'` och `*a = '2'`.

Uppgift 5 (11p)

a) Varför behöver vi i bland använda det reserverade ordet `volatile`? (2p)
b) Ange 2D-index för talvärdet 6 i följande deklaration. (1p)
```
int arrayOfArrays[3][4] = { {1,2,3,4}, {5,6,7,8}, {9,10,11,12} };
```
c) Beskriv principen för startbitsdetektering vid asynkron seriell överföring. (4p)
d) Hur kan man tolerera en mindre drift hos klockorna hos sändare och mottagare vid asynkron seriekommunikation? (4p)

Lösningsförslag

Uppgift 1a:

```
    LDR  R0,i     @ R0 <- i
    LDR  R1,j     @ R1 <- j
    ADD  R0,R0,R1 @ R0 <- i+j
    ADD  R0,R0,#2 @ R0 <- i+j+2
    LSL  R0,R0,#1 @ R0 <- (i+j+2) * sizeof(short)
    LDR  R1,=vs   @ R1 <- &vs
    ADD  R0,R0,R1 @ R0 <- &vs + [(i+j+2) * sizeof(short)]
```

Uppgift 1b:

```
f:
    PUSH  {LR}
    LSL   R0,R0,#1    @ R0 <- p<<2
    BL    reval       @ reval(p<<2)
    CMP   R0,#0       @ (rv > 0)?
    BLE   .1
    MOV   R1,#1
    ORR   R0,R0,R1    @ R0 <- rv|1
    B     .3          @ return (rv|1)
.1:
    BEQ   .2          @ (rv==0 )?
    MOV   R1,#1
    AND   R0,R0,R1    @ R0 <- rv&1
    B     .3          @ return (rv&1)
.2:
    MOV   R0,#0       @ return 0
.3  POP   {PC}
```

Uppgift 2a:

```
#define SYSCFG_EXTICR1  ((volatile unsigned int *)0x40013808)
#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 EXTI7_IRQVEC   0x2001C09C
#define NVIC_ISER0     0xE000E100

#define NVIC_EXTI7_IRQ_BPOS  (1<<23)
#define EXTI7_IRQ_BPOS       (1<<7)

#define GPIO_E         0x40021000
#define GPIO_MODER        ((volatile unsigned int *)  (GPIO_E))
#define GPIO_OTYPER   ((volatile unsigned short *)(GPIO_E+0x4))
#define GPIO_PUPDR     ((volatile unsigned short *)(GPIO_E+0x4))
#define GPIO_ODR_HIGH ((volatile unsigned char *) (GPIO_E+0x15))
#define GPIO_IDR_LOW  ((volatile unsigned char *) (GPIO_E+0x10))


void app_init ( void )
{
  *GPIO_MODER  = 0x55550000; /* PE0-7 in,  PE8-15 ut */
  *GPIO_OTYPER = 0;          /* Utgångar ”push/pull” */
  *GPIO_PUPDR  = 0;          /* Ingångar ”flytande” */
  value  = 0;
}
```

Uppgift 2b:

```
void irq_init ( void )
{
  *SYSCFG_EXTICR2 = 0x4000;     /* PE7->EXTI7  */

  *EXTI_IMR |= EXTI7_IRQ_BPOS;
  *EXTI_FTSR |= EXTI7_IRQ_BPOS;      /* trigger på negativ flank */
  *EXTI_RTSR |= EXTI7_IRQ_BPOS;      /* och trigger på positiv flank */
  *((void (**)(void) ) EXTI7_IRQVEC ) = irq_handler; /* sätt avbrottsvektor */
  *((unsigned int *) NVIC_ISER0) |= NVIC_EXTI7_IRQ_BPOS;    /* Aktivera detta avbrott */
}
```

Uppgift 2c:

```
char value;

void irq_handler ( void )
{
  if( *EXTI_PR & EXTI7_IRQ_BPOS )
  {
    *EXTI_PR |= EXTI7_IRQ_BPOS;   /* Kvittera avbrott */
    value = *GPIO_IDR_LOW & 0xF;  /* läs INP4 som unsigned char */
```

```
    if( (*GPIO_IDR_LOW & 0x80 )== 0)
    {  /* 1->0 */
      if( value & 0x8 )  /* MSB i INP4 */
        value  |= 0xF0;  /* teckenutvidga till signed char */
    }
    *GPIO_ODR_HIGH = value;
  }
}

void main(void)
{
    int dummy;
    app_init();
    irq_init();
    while(1); /* "Oändlig" slinga */
}
```

Uppgift 3:

```
start:
@   initiera port D0-D7 som utport
    LDR     R0,=0x00005555
    LDR     R1,=0x40020C00
    STR     R0,[R1]
@   initiera port E0-E15 som inport
    LDR     R0,=0
    LDR     R1,=0x40021000
    STR     R0,[R1]
@ adressen till port D:s ut-dataregister till R5
    LDR     R5,=0x40020C14
@ adressen till port E:s in-dataregister till R6
    LDR     R6,=0x40021010

main:
    LDRB    R0,[R6]
    LDRB    R1,[R6,#1]
    ADD     R0,R0,R1
    CMP     R0,#255
    BLE     main_2
    MOVW    R0,#0xFF
main_2:
    STRB    R0,[R5]
    B       main
```

Uppgift 4:

```
void    swap( char **a, char **b )
{
    char *c;
    c = *b;
    *b = *a;
    *a = c;
}
```

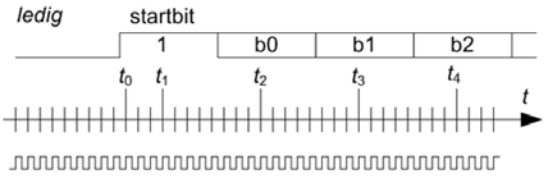
Uppgift 5a:

Ordet *volatile*(flyktig) används för att ange att data kan komma att ändras utanför en ordinär programflödeskontroll. Typiskt innebär detta data från något register mot omgivningen, eller en variabel som kan ändras av en avbrottsfunktion.

Uppgift 5b:

1,1

Uppgift 5c:



- Protokollet anger en ”ledig”-nivå på kommunikationsledningen. Då nivån växlar tolkas detta som en ”startbit” (t_0).
- Mottagaren läser därefter av ledningen (”samplar”) efter ett halvt bitintervall (t_1) och därefter ytterligare hela bitintervall (t_2,t_3,t_4 osv).
- Observera att sändare och mottagare på förhand måste veta vilken *bitrate* som används.

Uppgift 5d:

Genom att specificera maximalt antal bitar som skickas vid ett tillfälle, en ”ram”, kan man också bestämma en största tillåten differens och därmed också acceptera skillnader hos sändares respektive mottagare klockor.