Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA

Tentamen med lösningsförslag

EDA488 Maskinorienterad programmering Z

Fredag 23 augusti 2024, 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 (”över-strykningar”) får vara införda i detta häfte.

Tabellverk eller miniräknare får ej användas.

Lösningar
Anslås dagen efter tentamen via kursens hemsida.

Granskning
Tid och plats anges på kursens hemsida.

Allmänt

Uppgifter 1 t.o.m 3 utgörs av flervalsfrågor.
Försök först lösa uppgifterna, så som du lärt dig i kursen, jämför sedan med flervalsalternativen och ange det alternativ du anser är korrekt. Avge dina flervalssvar på blanketten, längst bak i tentamens-tesen. Skicka med denna blankett med dina övriga lösningar. Högst ett svarsalternativ får anges i varje uppgift. Siffror inom hakparenteser anger möjligt intervall för poängbedömning.

Observera att, bland svarsalternativen finns alternativ som *kan* ge poängavdrag. Om du är mycket osäker på ditt svar bör du därför i stället välja att avstå.

För övriga uppgifter gäller:

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 40 och tentamenspoäng ger slutbetyg enligt:
$16p \leq$ betyg $3 < 24p \leq$ betyg $4 < 32p \leq$ betyg 5

Uppgift 1 (-1..4p)

Följande deklarationer är givna på "toppnivå".

```
short a;
short b(int,unsigned short);
int c;
unsigned short d(long);
long e;
```

Vi har nu tilldelningen:

```
a = b ( c, d(e) );
```

Vilken av följande implementeringar i ARM v6-kod är korrekt?

A

```
LDR   R0,=e
LDR   R0,[R0]
BL    d
LDR   R0,=c
LDR   R0,[R0]
BL    b
LDR   R1,=a
STRH  R0,[R1]
```

B

```
LDR   R0,=e
LDR   R0,[R0]
BL    d
UXTH  R0,R0
LDR   R0,=c
LDR   R0,[R0]
BL    b
LDR   R1,=a
STRH  R0,[R1]
```

C

```
LDR   R0,=e
LDR   R0,[R0]
BL    d
MOV   R1,R0
LDR   R0,=c
LDR   R0,[R0]
BL    b
LDR   R1,=a
STRH  R0,[R1]
```

D

```
LDR   R0,=e
LDR   R0,[R0]
BL    d
UXTH  R0,R0
MOV   R1,R0
LDR   R0,=c
LDR   R0,[R0]
BL    b
SXTH  R0,R0
LDR   R1,=a
STRH  R0,[R1]
```

E

```
LDR   R0,=e
LDR   R0,[R0]
BL    d
UXTH  R0,R0
MOV   R1,R0
LDR   R0,=c
LDR   R0,[R0]
BL    b
SXTH  R0,R0
LDR   R1,=a
STR   R0,[R1]
```

F

```
LDR   R0,=e
LDR   R0,[R0]
BL    d
MOV   R1,R0
LDR   R0,=c
LDR   R0,[R0]
BL    b
LDR   R1,=a
STR   R0,[R1]
```

G

```
LDR   R0,=e
LDR   R0,[R0]
BL    d
UXTH  R0,R0
LDR   R0,=c
LDR   R0,[R0]
BL    b
SXTH  R0,R0
LDR   R1,=a
STR   R0,[R1]
```

H

```
LDR   R0,=e
LDR   R0,[R0]
BL    d
LDR   R0,=c
LDR   R0,[R0]
BL    b
SXTH  R0,R0
LDR   R1,=a
STR   R0,[R1]
```

Uppgift 2 (-1..6p)

En C-funktion `contain` är definierad enligt listning till höger.

Ange det alternativ (A-H) som visar en korrekt implementering (dvs. hur funktionen ska kodas) i ARM v6 assemblerspråk.

```
int contain( char *s, char c )
{
   while( *s )
   {
      if( *s == c )
         return 1;
      s++;
   }
   return 0;
}
```

A

```
contain:
   B     contain1
contain2:
   CMP   R2,R1
   BEQ   contain3
   ADD   R0,R0,#1
   B     contain1
contain3:
   MOV   R0,#1
   BL    LR

contain1:
   LDR   R2,[R0]
   CMP   R2,#0
   BNE   contain2
   MOV   R0,#0
   BX    LR
```

B

```
contain:
   B     contain1
contain2:
   CMP   R2,R0
   BEQ   contain3
   ADD   R0,R0,#1
   B     contain1
contain3:
   MOV   R0,#1
   BL    LR

contain1:
   LDR   R2,[R0]
   CMP   R2,#0
   BNE   contain2
   MOV   R0,#0
   BX    LR
```

C

```
contain:
   B     contain1
contain2:
   CMP   R2,R0
   BEQ   contain3
   ADD   R0,R0,#1
   B     contain1
contain3:
   MOV   R0,#1
   BX    LR

contain1:
   LDRB  R2,[R0]
   CMP   R2,#0
   BNE   contain2
   MOV   R0,#0
   BX    LR
```

D

```
contain:
   B     contain1
contain2:
   CMP   R2,R1
   BEQ   contain3
   ADD   R0,R0,#1
   B     contain1
contain3:
   MOV   R0,#1
   BX    LR

contain1:
   LDRB  R2,[R0]
   CMP   R2,#0
   BNE   contain2
   MOV   R0,#0
   BX    LR
```

E

```
contain:
   B     contain1
contain2:
   CMP   R2,R1
   BEQ   contain3
   B     contain1
contain3:
   MOV   R0,#1
   BX    LR

contain1:
   LDR   R2,[R0]
   CMP   R2,#0
   BNE   contain2
   MOV   R0,#0
   BX    LR
```

F

```
contain:
   B     contain1
contain2:
   CMP   R2,R0
   BEQ   contain3
   B     contain1
contain3:
   MOV   R0,#1
   BX    LR

contain1:
   LDR   R2,[R0]
   CMP   R2,#0
   BNE   contain2
   MOV   R0,#0
   BX    LR
```

G

```
contain:
   B     contain1
contain2:
   CMP   R2,R0
   BEQ   contain3
   B     contain1
contain3:
   MOV   R0,#1
   BL    LR

contain1:
   LDRB  R2,[R0]
   CMP   R2,#0
   BNE   contain2
   MOV   R0,#0
   BX    LR
```

H

```
contain:
   B     contain1
contain2:
   CMP   R2,R1
   BEQ   contain3
   B     contain1
contain3:
   MOV   R0,#1
   BL    LR

contain1:
   LDRB  R2,[R0]
   CMP   R2,#0
   BNE   contain2
   MOV   R0,#0
   BX    LR
```

Uppgift 3 (-2..6p)

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.
Sex olika programmerare får i uppgift att implementera funktionen. Resultaten blir likartade men bara en programmerare presterar ett helt korrekt resultat. De olika alternativen finns nedan, ange alternativet med den korrekta lösningen.

A

```
short   bitcheck( unsigned int *p, int *num)
{
    num = 0;
    while(*p)
    {
        if( *p ) (num)++;
        *p >>= 1;
    }
    num = (short) ((sizeof(int) * 8) - num);
    return !(num & 1);
}
```

B

```
short   bitcheck( unsigned int *p, int *num)
{
    num = 0;
    while(p)
    {
        if( p & 1 ) (num)++;
        p >>= 1;
    }
    num = (short) ((sizeof(int) * 8) - num);
    return !(num & 1);
}
```

C

```
short   bitcheck( unsigned int *p, int *num)
{
    *num = 0;
    while(*p)
    {
        if( *p ) (*num)++;
        *p >>= 1;
    }
    *num = (short) ((sizeof(int)) - *num);
    return !(*num & 1);
}
```

D

```
short   bitcheck( unsigned int *p, int *num)
{
    *num = 0;
    while(*p)
    {
        if( *p ) (*num)++;
        *p >> 1;
    }
    *num = (short) ((sizeof(int)) - *num);
    return !(*num & 1);
}
```

E

```
short   bitcheck( unsigned int *p, int *num)
{
    *num = 0;
    while(*p)
    {
        if( *p & 1 ) (*num)++;
        *p >>= 1;
    }
    *num = (short) ((sizeof(int) * 8) - *num);
    return !(*num & 1);
}
```

F

```
short   bitcheck( unsigned int *p, int *num)
{
    num = 0;
    while(p)
    {
        if( p & 1 ) (num)++;
        p >>= 1;
    }
    num = (short) ((sizeof(int)) - num);
    return !(num & 1);
}
```

Uppgift 4 (8p)

En signalenhet som omvandlar externa signaler är ansluten till en dator. Signalenheten har 512 ingångar (kanaler), numrerade från 0 till 511. Signalenheten kan bara avläsa en ingång i taget. Det avlästa värdet ges som ett positivt 24-bitars tal. Signalenheten är placerad på adress 0xFF600000 och kopplas till datorn via ett gränssnitt med följande register:

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
0																	status								ctrl							
																				IRQ		ER		DN	IE	IA			EN			RS
																				r		r		r	rw	rw			w1			w1
4																									channel							
																								w	w	w	w	w	w	w	w	w
8									data																							
									r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
0xC	ivr																															
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w		

Anm: **r**: biten är läsbar, **w**:biten är skrivbar, **w1**: biten återställs automatiskt till 0 av gränssnittet efter en skrivning '1'.

a) Visa lämpliga makrodefinitioner för referens (åtkomst) av portens register ctrl respektive `ivr`. Visa också hur din makrodefinition används vid skrivning av värdet 5 till register `ctrl` (2p)

b) Visa med en typdefinition `SIGUNIT`, hur porten kan avbildas med en *struct*-definition. (3p)
Visa speciellt hur delen data då refereras med hjälp av din typdefinition genom att värdet 0x123 skrivs till registret (1p).

c) Visa dessutom tydliga definitioner av de enskilda bitarna som används i gränssnittets register `ctrl` på sådant sätt som lärts ut i kursen, dvs. med användning av *bitfält* (2p).

Uppgift 5 (5p)

Använd SYSTICK för att skapa en realtidsklocka som genererar avbrott med 1 ms intervall. Systemets klockfrekvens är 168 MHz.

En avbrottsrutin, `void systick_irq_handler(void)` finns given, den ska utföras vid varje avbrott.

Implementera en funktion `void systick_init(void)` som gör *alla* nödvändiga initieringar och startar SYSTICK.

Följande makrodefinitioner kan förutsättas givna som pekare till respektive register.
`STK_CTRL, STK_LOAD` och `STK_VAL.` Vektortabellen startar på adress `0x2001C000`

Uppgift 6 (5p)

Förbered en enkel applikation som använder PD15 hos *MD407* som avbrottsingång. Dvs, skriv, i C, en sekvens som:

- Kopplar PD15 till EXTI15_10
- Konfigurerar EXTI15 för att generera avbrott på negativ flank
- Konfigurerar NVIC.

Ange också offset till den vektor som ska initieras för avbrottet.

Du kan förutsätta att alla moduler startats och behöver inte ta hänsyn till klockor (RCC). Observera dock att andra eventuella EXTI-konfigurationer *inte* får ändras av din programsekvens.

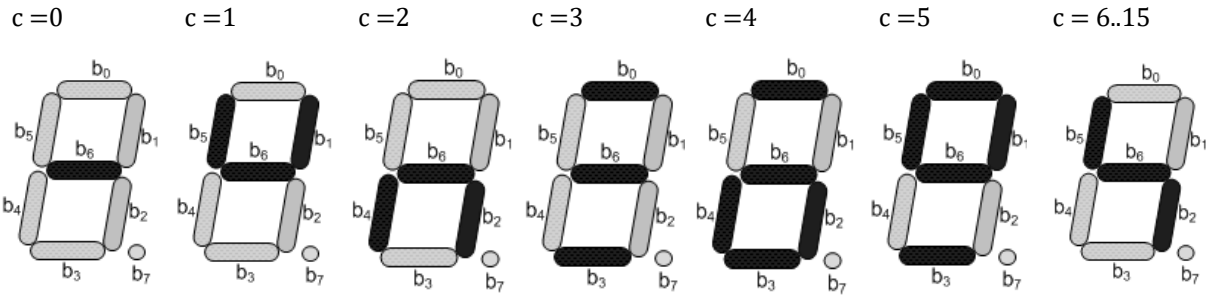
Följande makrodefinitioner kan förutsättas givna som pekare till respektive register.
`SYSCFG_EXTICR4, EXTI_IMR, EXTI_RTSR, EXTI_FTSR, NVIC_ISER1`

Uppgift 7 (6p)

En sjusegmentsdisplay är ansluten till GPIO D (bit 7..bit 0) till *MD407*.
Utskrift sker med en funktion: `void outFig( unsigned char c );`
Parameter c anger vad som ska skrivas ut.

- Om parametern är i intervallet 0..15 ska motsvarande figur (se nedan) skrivas ut till displayen.
- Om parametern är otillåten (större än 15) ska displayen släckas.

Följande figurer ska genereras vid utskrift:



a) Visa en funktion `void init_app(void)` som initierar port D för användning med displayen. Hänsyn behöver inte tas till eventuell användning av portens övriga bitar (1p).
b) Visa funktionen `outFig` (5p).

Följande makrodefinitioner kan förutsättas givna som pekare till respektive register.
`GPIO_D_MODER , GPIO_D_OTYPER, GPIO_D_ODRLOW`

Svarsblankett

Anonym kod

Maskinorienterad programmering, tentamen flervalsfrågor.

För varje fråga, kryssa i det alternativ du anser vara korrekt, eller avstå.
Lämna in denna blankett tillsammans med dina övriga lösningar.

Fråga	A	B	C	D	E	F	G	H
1	0	0	1	4	1	0	0	-1
2	0	0	2	6	0	0	-1	0
3	0	-1	2	1	6	-1		

Lösningsförslag

Uppgift 1 (-1..4p)

Svarsalternativ D

```
    LDR     R0,=e       @ R0<-&e
    LDR     R0,[R0]     @ R0<-e
    BL      d           @ R0<-d(e)
    UXTH    R0,R0       @ R0<-(unsigned short)d(e)
    MOV     R1,R0       @ R1<-d(e)
    LDR     R0,=c       @ R0<-&c
    LDR     R0,[R0]     @ R0<-c
    BL      b           @ R0<-b(c,d(e))
    SXTH    R0,R0       @ R0<-b(c,d(e))   (1
    LDR     R1,=a       @ R1<-&a
    STRH    R0,[R1]     @ a= b(c,d(e))

1)  REDUNDANT since only 16 bits are used in the
    assignment
    but formally ok.
```

Uppgift 2 (-1..6p):

Svarsalternativ D

```
@ int contain( char *s, char c )
@ R0: parameter s
@ R1: parameter c
@ R0: Returvärde
contain:
   B     contain1    @ while(*s)
contain2:
   CMP   R2,R1
   BEQ   contain3    @ (*s==c)?
   ADD   R0,R0,#1    @ s++
   B     contain1
contain3:
   MOV   R0,#1
   BX    LR          @ return 1

contain1:
   LDRB  R2,[R0]
   CMP   R2,#0
   BNE   contain2    @ (*s==0)?
   MOV   R0,#0
   BX    LR          @ return 0
```

Uppgift 3 (-2..6p):

Svarsalternativ E

```
short   bitcheck( unsigned int *p, int *num)
{   /* Vi räknar ettorna, det är enklast… */
    *num = 0;
    while(*p)
    {
        if( *p & 1 ) (*num)++;
        *p >>= 1;
    }
    /* Antal nollor */
    *num = (short) ((sizeof(int) * 8) - *num);
    return !(*num & 1);
}
```

Uppgift 4a (2p)

```
#define ctrl (( unsigned char *) 0xFF600000)
#define ivr (( unsigned long *) 0xFF60000C)
*(ctrl) = …;
```

Uppgift 4b (3+1p)

```
typedef struct{
  volatile unsigned char ctrl;
  volatile unsigned char status;
  short unused0;
  volatile unsigned short channel;
  short unused1;
  volatile unsigned int data;
  volatile unsigned int ivr;
} SIGUNIT;

( ( SIGUNIT *) 0xFF600000)->data;
```

Uppgift 4b (2p)

```
#define RS (1<<0)
#define EN (1<<3)
#define IA (1<<6)
#define IE (1<<7)
```

Uppgift 5 (5p)

```
#define SYSTICK_IRQVEC \
 ((volatile unsigned int *)0x2001C03C)

void systick_init( void )
{
    /* SystemCoreClock = 168000000 */
    *SYSTICK_IRQVEC = systick_irq_handler;
    *STK_CTRL = 0;
    *STK_LOAD = ( 168000-1 );
    *STK_VAL = 0;
    *STK_CTRL = 7;
}
```

Uppgift 6 (5p)

```
*SYSCFG_EXTICR4    &= 0x0FFF;     /* nollställ bitfält EXTI15 */
*SYSCFG_EXTICR4    |= 0x3000;     /* PD15->EXTI15  */
*EXTI_IMR          |= (1<<15);    /* aktivera avbrott EXTI15 */
*EXTI_FTSR         |= (1<<15);    /* aktivera trigger på negativ flank */
*EXTI_RTSR         &= ~(1<<15);   /* deaktivera trigger på positiv flank */
*NVIC_ISER1        |= (1<<8);     /* aktivera avbrott i NVIC */
```

Vektor nummer 40 (offset 0xE0)

Uppgift 7a (1p)

```
void init_app( void )
{
   *GPIO_D_MODER  = 0x00005555;
   *GPIO_D_OTYPER = 0x00000000;
   *GPIO_D_ODRLOW = 0;
}
```

Uppgift 7b (5p)

```
void outFig( unsigned char c )
{
   unsigned char code;
   unsigned char segCode[]={
   0x40,0x62,0x54,0x49,0x5D,0x6B };
   if( c >= 6){
      code = 0x64;
   }else
      code = segCode[c];
   *GPIO_D_ODRLOW = code;
}
```