Tenta 2011-03-14

Uppgift 1
Besvara kortfattat följande frågor, som alla utom i) -m) avser CPU12.

c) En instruktion med den hexadecimala maskinkoden 0E E9 00 0F 80 är placerad med operationskoden på adressen 1800_{16}. Skriv instruktionen med assemblerspråk. **(3p)**

Tenta 2012-01-14

Uppgift 1
För vilka värden W ($0 \leq W \leq 255$) utförs hoppet i programavsnitten e) och f)?

e)
```
LDAA  #W
NEGA                (Tänk på vad som händer om w = 0!)
CMPA  #$50
BHI   Hopp
```
(3p)

f)
```
LDAB  #$A3
CMPB  #W            (Tänk på att ”overflow” kan inträffa!)
BMI   Hopp
```
(4p)

Uppgift 2
b) Skriv en subrutin SWAP för processorn CPU12, som byter plats på databitarna i ett block i minnet så att $b_7b_6b_5b_4b_3b_2b_1b_0$ ersätts med $b_3b_2b_1b_0b_7b_6b_5b_4$ i hela blocket. Vid anrop av subrutinen finns antalet 8-bitars dataord i blocket i B-registret (högst 255 st) och begynnelseadressen i X-registret. Endast flaggregistret får vara förändrat vid återhopp från subrutinen. Skriv subrutinen i assemblerspråk för CPU12. Radkommentarer skall finnas. **(7p)**

Uppgift 4
En dator med processorn CPU12 skall användas i styrenheten i en maskin. Styrprogrammet skall läsa av två inportar via IRQ-avbrott, PORTA 10 gånger per sekund och PORTB en gång per minut. CS-signalerna för portarna är inte tillgängliga.

Det finns en binär signal med den konstanta frekvensen 400 Hz tillgänglig för generering av avbrott. På adresserna $30F8_{16}$-$30FF_{16}$ aktiveras inga minnesmoduler eller portar. Endast avläsningen av portarna skall vara avbrottsstyrd i systemet.

a) Föreslå en koppling med vars hjälp man kan generera IRQ-avbrott 400 gånger per sekund. D-vippor, AND- och NOT-grindar får användas. Avbrottssystemet används inte till något annat i datorn. **(2p)**

b) Skriv en avbrottsrutin IRQR, som läser av portarna enligt beskrivningen ovan och placerar de avlästa värdena på de symboliska adresserna ADRA och ADRB i minnet. Ledigt utrymme för globala variabler finns på adresserna $1FF0_{16}$-$1FFF_{16}$. **(4p)**

c) Skriv ett avsnitt av huvudprogrammet där IRQ-avbrott initieras. IRQ-vektorn antas vara placerad i RWM på adresserna $FFF2_{16}$ och $FFF3_{16}$. **(3p)**

De symboliska adresserna ovan är definierade på annat ställe i programmet. Assemblerspråk för processorn CPU12 skall användas. Radkommentarer skall finnas!

Tenta 2012-08-27

Uppgift 5
Du använder korskompilatorn XCC för CPU12, som har följande konventioner för C-funktioner:

- Inparameterlistan behandlas från höger till vänster och samtliga inparametrar överförs via processorns stack.
- Lokala variabler som deklareras placeras på stacken i den ordning de deklareras, dvs sist behandlad finns överst i stacken. Övriga lokala variabler placeras också på stacken i den ordning behovet av dem uppstår, dvs den sista finns överst på stacken.
- Varje funktion som har lokala variabler inleds med prologen `LEAS -?,SP` och avslutas med epilogen `LEAS ?,SP` följt av `RTS`.
- Returparameter (16- eller 8-bitars) lämnas i D- eller B-registret beroende på storlek.
- För XCC gäller dessutom: char 8 bitar, short och int 16 bitar, long 32 bitar.

a) Översätt hela C-programmet till höger till assemblerspråk för CPU12. Visa även hur stacken ser ut när for-satsen börjar utföras. **(6p)**

b) Vilket värde returneras från funktionen func? Motivera svaret! **(2p)**

```
char func(char xx, char yy);

char z = 0x60;

void main(){
 func(10,20);
}

 char func(char x, char y){
   char i;

   for ( i = 0; i <= 15; i = i + 3)

     if(i < x)
        y = z + y;

 return y;
}
```

Tenta 2011-03-14

(Lösning)
Uppgift 1

c) 1800_{16}: 0E E9 00 0F 80 Operationskoden 0E gäller för instruktionen BRSET med indexerad adressering.

Andra byten xb = E9 ger att typen är –n,Y med 9-bitars offset med formatet: 0E E9 *ff mm rr*, där *ff* är de 8 låga bitarna av 9-bitarstalet –n = 100_{16}. n = $–(100000000_2)_{2k} = –100000000_2 = –100_{16}$

mm = $0F_{16}$ är en mask och *rr* = 80_{16} en 8-bitars offset för ett PC-relativt hopp, från adressen till nästa OP-kod, dvs 1805_{16}.

Offset = Tilladr – Frånadr ger att *Tilladr = Offset + Frånadr* = $FF80_{16} + 1805_{16} = 1785_{16}$.

Instruktionen är alltså: BRSET -$100,Y,#$0F,$1785 **(3p)**

Tenta 2012-01-14

(Lösningar)
Uppgift 1

e) BHI avser tal utan tecken. Det innebär att vi skall tolka data som tal i intervallet [0, 255].
Talet $50_{16} = 5 \cdot 16 = 80_{10}$.

NEGA bildar talet $W_{2k} = 256 – W$ för W > 0. (Fallet W = 0 behandlas separat nedan.)
CMPA utför subtraktionen: 256 –W – 80 = 176 –W.

Hoppvillkoret blir: 176 –W > 0 eller W < 176.
När hänsyn tas till talområdet blir hoppvillkoret: $1 \leq W < 176$.

Om W = 0 gäller att NEGA bildar talet 0. CMPA utför då subtraktionen: 0 – 80 < 0. (Ej hopp.)

Hoppvillkoret är alltså: $1 \leq W < 176$ **(3p)**

f) Hoppet utförs om teckenflaggan N = 1. Det innebär att vi skall tolka data som tal med tecken, med talområdet [-128, 127].

Talet $A3_{16} = 10 \cdot 16 + 3 = 163_{10}$ tolkas som det negativa talet –(256 – 163) = –93.

CMPB utför subtraktionen: –93 – W.

Hoppvillkoret blir: –93 – W < 0 (dvs. negativt, N = 1) eller W > –93
När hänsyn tas till talområdet blir hoppvillkoret: $–93 < W \leq 127$.

Om overflow inträffar får dock N-flaggan fel värde.

Här inträffar overflow vid subtraktionen om –93 – W < –128, dvs. 128 – 93 < W, eller W > 35.

Vi får då hoppvillkoret: $–93 < W \leq 35$

Eftersom vi använder 2k-representation delar vi upp talintervallet i en positiv och en negativ del:
$0 \leq W \leq 35$ och $–93 < W \leq –1$, vilket motsvarar $256 –93 < W \leq 256 –1$ eller $163 < W \leq 255$
Hoppet utförs om: $0 \leq W \leq 35$ eller $163 < W \leq 255$ **(4p)**

Tenta 2012-01-14

(Lösningar)

Uppgift 2

b)

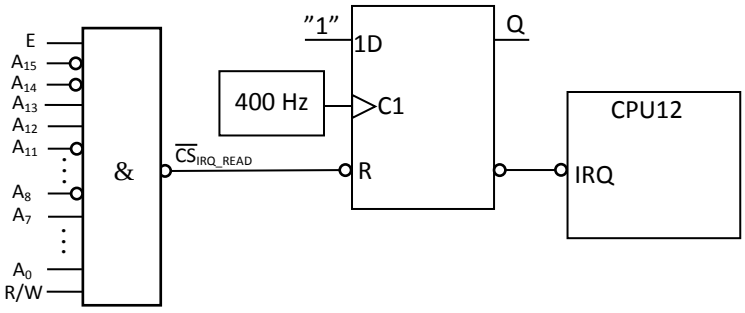
```
SWAP    PSHX            Spara på stack
        PSHD
        PSHB            Ordräknare till stack

        TST   ,SP       Färdigt?
        BEQ   SWEXIT    Ja
LOOP3   LDAA  ,X        Hämta dataord från tabell
        TFR   A,B       Gör kopia
        LSLD            Flytta låg nibble i reg A till vänster
        LSLD            och hög nibble i reg B till höger i reg A
        LSLD
        LSLD
        STAA  1,X+      Uppdatera tabell
        DEC   ,SP       Minska ordräknare
        BNE   LOOP3     Nästa byte om ej färdigt
SWEXIT  LEAS  1,SP      Justera stacken för ordräknaren
        PULD            Återställ stack
        PULX
        RTS
```

(7p)

Uppgift 4

a) Adress för nollställning av avbrottsvippa:

$30FF_{16} = 0011\ 0000\ 1111\ 1111_2$



(2p)

b) Avbrottsrutin

```
IRQR    TST   IR_FF           Nollställ avbrottsvippa
        DEC   ACOUNT          10 Hz-räknare
        BNE   CHK_MIN         Ej 10 Hz

        MOVB  PORTA,ADRA
        MOVB  #40,ACOUNT      Ominitiera räknare för 10 Hz
CHK_MIN LDX   BCOUNT          Minuträknare
        DEX
        STX   BCOUNT
        BNE   EX_IR           Ej hel minut

        MOVB  PORTB,ADRB
        MOVW  #24000,BCOUNT   Ominitiera räknare för minut
EX_IR   RTI
```

(4p)

c) Avsnitt av huvudprogram

```
ACOUNT  EQU   $1FF0
BCOUNT  EQU   $1FF1
IR_FF   EQU   $30FF

START   LDS   #BOS            Initiera stackpekaren
        MOVW  #IRQR,$FFF2     Sätt avbrottsvektorn
        TST   IR_FF           Nollställ avbrottsvippa
        MOVB  #40,ACOUNT      Räknare för 10 Hz
        MOVW  #24000,BCOUNT   Räknare för 1 minut
        CLI                   Aktivera avbrottssystem
        .
        .
        .
```

(3p)

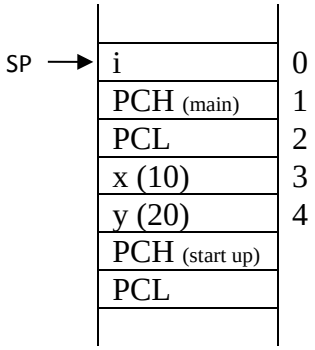
Tenta 2012-08-27

(Lösning)

Uppgift 5

5. a)

```
char z = 0x60;                    z       FCB     $60     (96)

void main(){                              main
 func(10,20);                             LDAB    #20
                                          PSHB
                                          LDAB    #10
                                          PSHB
                                          JSR     func
                                          LEAS    2,SP
}                                         RTS

 char func(char x, char y){               func    LEAS    -1,SP
  char i;

  for ( i = 0; i <= 15; i = i + 3)        for     CLR     0,SP

                                          floop   LDAB    0,SP
                                                  CMPB    #15
                                                  BLE     if
                                                  BRA     return

     if(i < x)                            if      LDAB    0,SP
                                                  CMPB    3,SP
                                                  BGE     big
    y = z + y;                                    LDAB    4,SP
                                                  ADDB    z
                                                  STAB    4,SP

                                          big     LDAB    0,SP
                                                  ADDB    #3
                                                  STAB    0,SP
                                                  BRA     floop

 return y;                                return  LDAB    4,SP
                                                  LEAS    1,SP
 }                                                RTS
```



(6p)

b) För *i = 0* till 9 adderas 96 4 gånger till y-värdet 20 i ”if-satsen”. 20 + 4*96 = 404. Eftersom y är en char (8 bitar) representeras värdet modulo 2^8 = 256. 404 – 256 = 148. Dessutom är y ett tal med tecken i intervallet [–128, 127]. Det innebär att 148 skall tolkas som det negativa talet –(256 – 148) = –108
Returvärde: *148 som skall tolkas som –108* **(2p)**