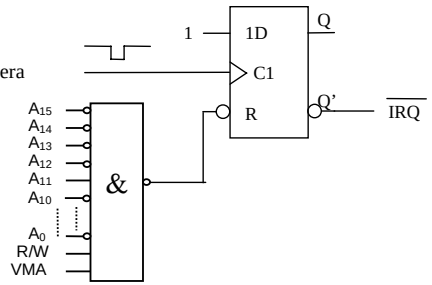
14.1 **a)** Avbrottsrutinen på adressen KNAPPINC (3400H) blir:

```
KNAPPINC  INC   $2800      * Variabel KNAPP ökas med ett
          LDAA  $0800      * Nollställ avbrottsvippan
          RTI              * Återvänd till huvudprogrammet
```

b) Nedan visas nödvändiga initieringar i huvudprogrammet för att avbrott på IRQ-ingången skall accepteras. Dessutom visas hur IRQ-vektorn ges sitt rätta värde KNAPPINC (C400H)

```
HUVUDPRG  LDS   #STACKADR    * BOS definieras först
          LDX   #$3400       * Adr KNAPPINC = 3400H
          STX   $FFF2        * Ge IRQ-vektorn värdet KNAPPINC
          LDAA  $0800        * Nollställ avbrottsvippan
          CLR   $2800        * Nollställ variabeln KNAPP (2800H)
          ANDCC #%11101111   * Aktivera avbrottssyst. för IRQ
          ...                * (Dvs nollställ I-biten i CC-reg)
```

14.2 **a)** D-vippan används som avbrottsvippa. Den måste nollställas före första pulsen och mellan pulserna. Chip-select för en oanvänd portadress (DUMMY) används för att generera resetpulsen till D-vippan



b)
```
PULSANT   EQU   $0B00
IRQSR     LDD   PULSANT
          ADDD  #1         * Öka variabeln PULSANT med ett
          STD   PULSANT
          LDAA  DUMMY      * Nollställ avbrottsvippan
DONE      RTI
```

c)
```
IRQINI    LDX   #IRQSR     * Adressen till avbrottsrutin
          STX   $FFF2      * Initiera avbrottsvektor
          LDAA  DUMMY      * Nollställ avbrottsvippan
          LDD   #0
          STD   PULSANT    * Nollställ variabeln PULSANT
          ANDCC #$EF       * Tillåt IRQ
```

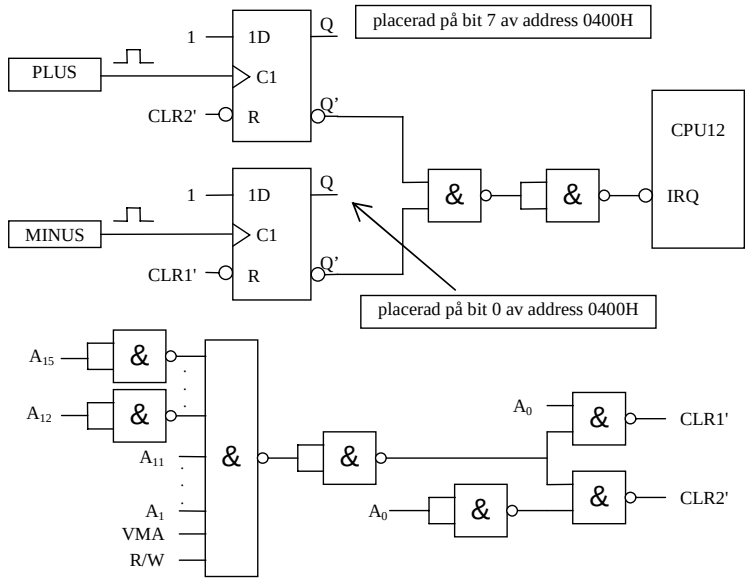
14.3 **a)**
```
IRQSR     LDAA  $0ED4      * Läs avbrottsvippor
          BITA  #%00000001 * Maska fram RTC-biten
          BNE   RTCINT
          BITA  #%00000010 * Maska fram EVENT-biten
          BNE   EVINT
          BRA   SLUT
RTCINT    JSR   RTC        * Realtidsklockan
          BRA   SLUT
EVINT     JSR   EVENT
SLUT      RTI
```


b)
```
EVENT     INC   $1C05      * Öka låg byte med ett
          BNE   NOCARRY
          INC   $1C04      * Öka hög byte med ett
NOCARRY   LDA   $0ED1      * Nollställ EVENT-vippa
          RTS
```

c)
```
RTC       COM   $1C06      * Invertera hjälpvariabeln
          BNE   KLAR       * Varannan gång:
          INC   $1C03      * Öka låg byte av CLOCK
          BNE   KLAR       * Ingen carry
          INC   $1C02      * Öka hög byte av CLOCK
KLAR      LDA   $0ED0      * Nollställ RTC-vippa
          RTS
```

d)
```
IRQINI    PSHS  A,CC,X
          CLR   $1C02      * Nollställ samtliga variabler
          CLR   $1C03
          CLR   $1C04
          CLR   $1C05
          CLR   $1C06
          LDX   #IRQSR     * Initiera IRQ-vektorn
          STX   $FFF2
          LDA   $0ED0      * Nollställ avbrottsvipporna
          LDA   $0ED1
          PULS  A,CC,X
          ANDCC #%11101111 * Tillåt avbrott
          RTS
```

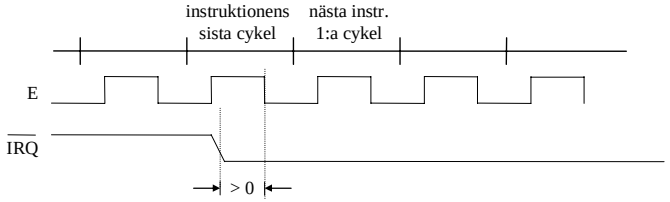
14.4 **a)** För att samtidig nedtryckning av knapparna skall kunna hanteras, måste vipporna kunna nollställas individuellt. Vi väljer att låta läsning av adresserna 0FFFH resp. 0FFEH nollställa MINUS- resp. PLUS-knappen.



b)
```
KNAPPAVB  LDA   $0400          * Läs vipporna
          BMI   PLUS
          BITA  #%00000001
          BNE   MINUS
          JSR   FEL           * Felhantering
          JMP   UT
PLUS      LDB   $0FFF          * Nollställ PLUS-vippan
          LDB   KNAPP
          CMPB  #$FF          * Kontrollera om KNAPP = FFH
          BEQ   UT
          INCB
          STB   KNAPP
          JMP   UT
MINUS     LDB   $0FFE          * Nollställ MINUS-vippan
          LDB   KNAPP
          BEQ   UT            * KNAPP = 0?
          DECB
          STB   KNAPP
UT        RTI
```

14.5
```
* Initiera två program (processer) A och B, samt starta program A

STKPTR    RMB   2             * Lagringsplats för inaktiv stackpekare

Init      LDX   #$3B00-12     * Utrymme för prog B stack
          STX   STKPTR        * B:s stackpekare inaktiv ibörjan
          LDAA  #%11000000    * B:s initialvärde för CC-reg
          STAA  ,X
          LDD   #$2200        * B:s startadress
          STD   10,X
          LDS   #$2200        * A:s initialvärde för stackpek
          LDX   #IrqHnd       * Initiera IRQ-vektorn
          STX   $FFF2
          LDA   TIMSTAT
          ANDCC #%11101111    * Aktivera IRQ
          LBRA  $1200         * Hoppa till program A

* Avbrottsrutin, anropas en gång var 20:e ms.
* Växla program, dvs byt ut innehållet i S-registret

IrqHnd    LDY   STKPTR        * Hämta inaktiv stackpekare
          STS   STKPTR        * Spara föregående stackpekare
          TFR   Y,S           * Växla stackpekare och därmed program
          LDAA  TIMSTAT
          RTI
```


14.6 **a)** När avbrottsbegäran kommer strax innan negativa E-flanken i instruktionens sista cykel.



b) När avbrottsbegäran kommer för sent för att den skall hinna upptäckas. Vid CPU12:s längsta instruktion. Då är den lika många cykler som instruktionen.

14.7 I båda fallen är $t_{res\_min} = t_{lat\_min} + t_{hantering} + t_{serv}$

där $t_{hantering} = 9T$ och händelsen servas omedelbart i avbrottsrutinen, vilket ger att $t_{serv} = 0$.

14.8 **a)** Den exekverande instruktionens längd.
Avbrottet maskerat, ex vis vid samtidig, annan avbrottsbegäran med högre prioritet.
Processorn i exekveringsuppehåll.

b) Latenstiden
Antal register som stackas.
Sätt att avgöra vilken yttre enhet som begärt avbrott.
Processorn i exekveringsuppehåll.
Avbrott med högre prioritet.

c) Antal register som stackats.
Processorn i exekveringsuppehåll.
Avbrott med högre prioritet.

14.9 Det finns ingen stack definierad. Vid avbrottshantering stackas register för att sedan kunna läsas tillbaka när avbrottet har hanterats.

14.10 Den har inge speciell "reset acknowledge"-signal utan det får ske genom adressavkodning i samband med läsning av resetvektorn.

14.11 1) RESET- och XIRQ-vektorerna måste vara bestämda och vara laddade.
2) Stackpekaren måste initieras med BOS.
3) Eftersom RESET maskerar avbrott, måste XIRQ demaskeras (tillåtas) med ANDCC #%10111111.

14.12 Vid stackningen av registerinnehåll skriver processorn på en ospecificerad plats i minnet.