Lösningar till tentamen i Realtidssystem för DAI 2/ EI3 2011-10-19
Version 2012-10-18 (Uppgift 8 justerat svar)

Uppgift nr 1.(3p)

a) Arbitreringskoden har följande funktion:

Värdet på koden ger prioritering avseende vilken nod som kommer att sända hela meddelandet om fler börjar sända samtidigt på bussen. Lägsta värde ger högsta prioritet.

Värdet styr vilka noder som läser in ett visst meddelandet genom jämförelse med en i noden befintligt kod i ett acceptans-register.

b)

```
  //Acceptence Reg 1o2
   ptrCAN4->CAN_AR_MR_ID[0]=0x0AAAAAAC;   // ID acc reg
   ptrCAN4->CAN_AR_MR_ID[1]=0x0AAAAAAF;   // ID acc reg
   ptrCAN4->CAN_AR_MR_ID[2]=0xF0000000;       // ID mask reg

  //---------------------- Huvudloop ----------------------------------------------------
    while(1){
          if(receive_CAN(ptrCAN4,ptrCAN_rec_message)==1){      //if received message
               length=ptrCAN_rec_message->length;                    // antal databytes
               for ( i=0;i<length;i++){
                          intxt[i]=(ptrCAN_rec_message->byte[i]);
                 }
                 intxt[i]='\0';                                                  // fixa strängslut
                 puts(intxt);
          }else
                   puts("no message\n\r");

   }// End Huvudloop
}// End Main
```

Uppgift 2 (3p)

Pre-emtive: Processen får avbrytas när som helst och kan ingå i en tidsdelad schemaläggning.
Non- pre-emtive : Processen får ej avbrytas utan måste tillåtas exekveras klart. Processer av denna typ schemalägges seriellt.

Round Robin: Körklara processer placeras i en cirkulär FIFO-kö (first in first out) där de tilldelas lika långa tidskvanta (timeslice).

Tidsdelad exekvering: Processerna exekveras i viss turordning under korta, återkommade lika långa tidsperioder.
Seriell exekvering: Processerna exekverar efter varandra där varje process exekverar klart innan nästa kan starta.

Statisk Schemaläggning: Processerna exekveringstider schemalägges före run-time.

Dynamisk Schemaläggning: Schemaläggning sker under run-time, vanligen genom tidsdelning av bestämd tidsperiod till processerna. Fördelningen styrs av en sk schemaläggare.

Uppgift 3: (3p)
a)

En semafor är en heltalsvariabel > = 0 som hanteras av realtidsoperativsystemet (RTOS) . Värdet på variabeln kan endast påverkas genom anrop av operationerna waitsem(int SemId) och signalsem(int SemId).

waitsem(S) : Räknar ner semaforen S om S>0, i annat fall kommer anropande process att suspenderas.
signalsem(S) : Om S= 0 och process väntar på semaforen så startas denna väntande processen annars räknas S upp.

b)
Kritisk region: Ett programsegment i en process som i förhållande till ett segment i en annan process ej kan exekveras 'samtidigt' . De kritiska regionerna delar gemensamma resurser (data, in/ut-portar ..) som bara kan hanteras av en process i taget.

Odelbar region är ett programsegment som inte får avbrytas av någon annan process.

Ömsesidig uteslutning: Garanterar att endast en av flera kritiska regioner som hanterar gemensam resurs/data exekveras 'samtidigt'.

c)

```
PROCESS nr_1 {

        waitsem(S1);
          if n>0{
                tal=buf[n];
                n=n-1;
          }
        signalsem(S1)
          ….
PROCESS nr_2{

        waitsem(S1);
          if n<MAX{
                n=n+1;
                buf[n]=intal;
          }
        signalsem(S1);
```

Uppgift 4 (2p)

a) Garanterar ej ömsesidig uteslutning; Beskrivning se Dekker 2 sid 45 i läroboken

b) **Deadlock:** När två eller flera processer låser varandra från vidare exekvering genom att de till exempelvis korsvis väntar på semafor som den andra processen har tagit.

Svält: När en process av någon anledning får vänta betydligt längre än vad som avses för att få CPU-tid i syfte att exekvera vidare.

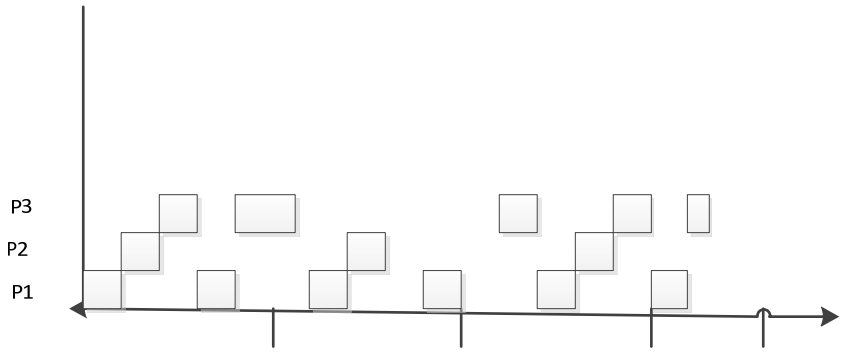
Uppgift 5 (3p)

a)
Lämpligen justeras periodtiderna till : 6, 12 , 18.

Före justering :
LCM(6,14,18)=LCM(LCM(6,14),18)=LCM((6*14)/2),18)=42*18/6=126

LCM(6,12,18)=LCM(LCM(6,12),18)=LCM((6*12)/6),18)=12*18/6=36

b) Statiskt schema :



Uppgift 6 (3p)

a)
Svarstid för Process 4:

$$R_4^1 = 16 + \left\lceil \frac{16}{68} \right\rceil \bullet 12 + \left\lceil \frac{16}{34} \right\rceil \bullet 6 + \left\lceil \frac{16}{18} \right\rceil \bullet 4 = 38$$

$$R_4^1 = 16 + \left\lceil \frac{38}{68} \right\rceil \bullet 12 + \left\lceil \frac{38}{34} \right\rceil \bullet 6 + \left\lceil \frac{38}{18} \right\rceil \bullet 4 = 52$$

Osv till $R_4 = 52$

P3:

$$R_3^1 = 12 + \left\lceil \frac{12}{34} \right\rceil \bullet 6 + \left\lceil \frac{12}{18} \right\rceil \bullet 4 = 22$$

Osv till $R_3=26$
…..
$R_2=10$ och $R_1=4$

Alla R_i är mindre än motsvarande deadline vilket gör att processerna är schemaläggningsbara.

b)
Semaforanvändningen ger att P3 och P2 båda får en blockeringsfaktor b=5 ms.

R_3 beräknas enligt a) men med b=5 i uttrycket.
$R_3=31$ ms vilket gör att deadline för processen överskrides.

Uppgift 7 (2p)

Beräkningar enligt metoden för godtycklig deadline ger för en start följande första uttryck:

$$w(0) = 3 + \left\lceil \frac{3+1}{25} \right\rceil \bullet 6 + \left\lceil \frac{3}{20} \right\rceil \bullet 5 = 14$$

Ytterligare en iteration ger w(0) = 14 dvs konvergens . Svarstiden för en start är större än periodtiden varför man testar med ytterligare en start (q=1) vilket ger första iterationens uttryck enligt nedan:

$$w(1) = 2 \bullet 3 + \left\lceil \frac{14+1}{25} \right\rceil \bullet 6 + \left\lceil \frac{14}{20} \right\rceil \bullet 5 = 17$$

Ytterligare en iterationer ger svaret 17 dvs svarstiden är 17.
Svarstiden 17 ms är större än 2 x periodtiden (=16) vilket gör att man måste testa ytterligare en start dvs q=2.

$$w(2) = 3 \bullet 3 + \left\lceil \frac{9+1}{25} \right\rceil \bullet 6 + \left\lceil \frac{9}{20} \right\rceil \bullet 5 = 20$$

Ytterligare iterationer ger konvergens för svarstiden 20 ms . Svarstiden 20 ms är mindre än 3xperiodtiden (=24 ms) vilket gör att man kan sluta räkna på fler starter.

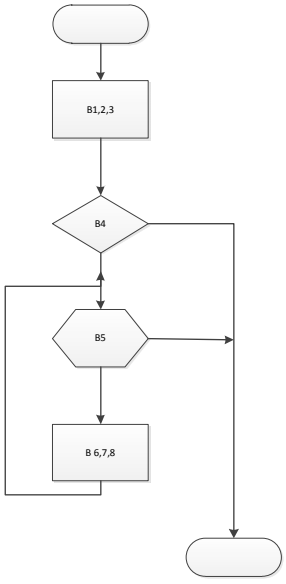
Man räknar nu ut varje enskild starts svarstid och bland dem finns den maximala.

R(0)=w(0)= 14 (ms)
R(1)=w(1)-1*p=17-8= 9 (ms)
R(2)=w(2)-2*p=20-16= 4 (ms)

Svar : R_3max = 14 ms .

Uppgift 8 (2p) / Justerad version

a) Programflödesgraf enligt vidstående fig.



b) BBmin = BB1-3min+BB4min = 35

BBmax = (BB1-4)max + 3*(BB5-8max)+BB5 =223

Uppgift 9 (3p):

a)

```
REGTYP* add_first(REGTYP* temp, int data){
// Funktion som lägger till ett element först i den länkade listan samt
// lägger in talet data i elementets fält tal.

        REGTYPE *newitem;
        newitem=(REGTYP*) malloc(sizeof(REGTYP));
        newitem->tal=data;
        newitem->next=temp;
        newitem->prev=NULL;
        temp->prev=newitem;
        return newitem

}
```

b)

```
head=add_first(head, intal);
```
