****
TABASCO
****

APPENDIX A

UDSKRIFT AF MIK

MED SYMBOLTABEL

****

```
bs2
     0    ;****************************************
     0    ;* MIK - et styresystem til intel8080 *
     0    ;*                                      *
     0    ;* Bodil Schrøder                       *
     0    ;*                                      *
     0    ;* 2 maj 1975                           *
     0    ;****************************************
     0
     0
     0
     0    active: 0,1,1,2,2
     5            0,1,1,2,2
    10            0,1,1,2,2
    15    running: 0,0
    17    sleeping:0,0,1,1,2,2
    23
```

```
23
23    ;***************
23    ;into(chain,CD)
23    ;***************
23
23    ;before:    (B,C) = address of chain
23    ;           (D,E) = address of coroutinedescription
23
23    ;after:     (B,C) = undefined
23    ;           (D,E) = address of CD
23    ;           (H,L) = undefined
23
23    ;into inserts the CD as the last element in the chain
23
23    into:    ldax b           ;A:=length of chain
24             ana  a           ;test A
25             jz   iempty      ;jump on empty chain
28
28             inr  a           ;length:=length+1
29             stax b           ;store length
30
30             inx  b
31             inx  b
32             inx  b           ;(B,C):=last(1)
33
33             ldax b
34             mov  l,a         ;L:=last(1)
35             inx  b
36             ldax b
37             mov  h,a         ;H:=last(2)
38
38             mov  m,e
39             inx  h
40             mov  m,d         ;last.link:=CD
41
41             mov  a,d
42             stax b           ;last(2):=D
43             dcx  b
44             mov  a,e
45             stax b           ;last(1):=E
46             ret
47
47                              ;the coroutinedescription pointed at in
47                              ;(D,E) is inserted as the only element
47                              ;in the chain identified by (B,C)
47    iempty:  inr  a
48             stax b           ;chain.length:=1
49
49             inx  b           ;(B,C):=first(1)
50             mov  a,e
51             stax b
52             inx  b
53             mov  a,d
54             stax b           ;first:=CD
55
55             inx  b
56             mov  a,e
57             stax b
58             inx  b
59             mov  a,d
60             stax b           ;last:=CD
61             ret
62
```

```
62
62
62    ;***************
62    ;firstout(chain)
62    ;***************
62
62    ;before:   (B,C) = address of chain
62
62    ;after:    A=0  if the chain was not empty
62    ;          A=1  if the chain was empty
62    ;          (B,C) = undefined
62    ;          (D,E) = address of CD
62    ;          (H,L) = unchanged
62
62    ;firstout removes the first CD from the chain and puts its
62    ;address in (D,E). If the chain is empty A is set to 1.
62
62    firstout:
62            ldax b          ;A:=length of chain
63            ana  a          ;test A
64            jz   fempty     ;jump on empty chain
67
67            dcr  a
68            stax b          ;length:=length-1
69
69            inx  b          ;(B,C):-first(1)
70            ldax b
71            mov  e,a        ;E:=first(1)
72            inx  b          ;(B,C):-first(2)
73            ldax b
74            mov  d,a        ;D:=first(2)
75
75            dcx  b          ;(B,C):-first(1)
76            ldax d
77            stax b          ;first(1):-CD.link(1)
78            inx  b
79            inx  d
80            ldax d
81            stax b          ;first(2):=CD.link(2)
82            dcx  d          ;(D,E):-CD again
83
83            xra  a          ;A:=0
84            ret
85
85    fempty: inr  a          ;A:=1
86            ret
87
```

```
87
87
87   ;*********************************
87   ;common part of the central logic
87   ;*********************************
87
87   ;before this entry, interrupts have been disabled and the
87   ;       RUNNING pointer has been saved if neccessary
87   ;after  this routine a jump is made to the local program
87   ;       counter of the new RUNNING coroutine, which is
87   ;       the first in the ACTIVE chain
87   ;
87   common: lxi  b,active
90   comout: ldax b
91           ana  a             ;A:=length(activechain(priority))
92           jnz  comfound
95           inx  b
96           inx  b
97           inx  b
98           inx  b
99           inx  b             ;(B,C):=next chain
100          jmp  comout        ;ACTIVE is never empty
103  comfound:
103          call firstout      ;first CD in first not empty chain
106
106          lxi  h,running
109          mov  m,e
110          inx  h
111          mov  m,d           ;RUNNING:=next CD to run
112
112          inx  d
113          inx  d             ;(DE):=RUNNING,LPC
114          ldax d
115          mov  l,a
116          inx  d
117          ldax d
118          mov  h,a           ;(HL):=LPC
119          ei
120          pchl               ;PC:=RUNNING,LPC
121
```

```
121
121
121     ;********
121     ;release
121     ;********
121
121     ;a call of this routine enables other processes
121     ;- if any - to get processor time
121
121     release:
121             di
122             lhld running    ;(H,L) := RUNNING
125             mov  d,h
126             mov  e,l        ;(D,E):=(H,L)
127             inx  h
128             inx  h          ;(H,L):=running,LPC
129             pop  b          ;(B,C):=old program count
130             mov  m,c
131             inx  h
132             mov  m,b        ;running,LPC:=old pc
133             call intactive
136
136             jmp  common
139
```

```
139
139
139   ;***************
139   ;reactivate(CD)
139   ;***************
139
139   ;before:  (D,E) = address of CD
139
139   ;after:   (B,C) = undefined
139   ;         (D,E) = unchanged
139   ;         (H,L) = undefined
139   ;          A = 1  if activation was not allowed
139   ;          A = 0  if activation was allowed
139
139   ;if CD.status is passive,  CD.status is set to active and
139   ;the CD is inserted into the ACTIVE chain
139   ;otherwise nothing happens
139
139   reactivate:
139            di
140            mov   h,d
141            mov   l,e          ;(H,L):=CD
142            inx   h
143            inx   h
144            inx   h
145            inx   h            ;(H,L):=CD.status
146            mov   a,m
147            ana   a
148            jp    resfault     ;if status(0) = 1, status is passive:
151
151            ani   127          ;status(0):=0
153            mov   m,a
154            call  intactive
157            xra   a            ;A:=0
158            ei
159            ret
160
160   resfault:
160            mvi   a,1          ;A:=1
162            ei
163            ret
164
```

```
164
164
164   ;**********
164   ;passivate
164   ;**********
164
164   ;the address from which the coroutine is going to continue
164   ;when it is activated again - normally the LPC or the start-
164   ;address - is fetched from the stack and inserted into CD.LPC
164   ;status is set to passive
164
164   passivate:
164           di
165           lhld running    ;(H,L):=CD of calling coroutine
168           inx  h
169           inx  h          ;(H,L):=CD.LPC
170           pop  d          ;(D,E):=continuation address
171           mov  m,e
172           inx  h
173           mov  m,d        ;CD.LPC:=cont.addr.
174
174           inx  h          ;(H,L):=CD.status
175           mov  a,m
176           ori  128        ;first bit in status is set
178           mov  m,a
179
179           jmp  common
182
```

```
182
182
182   ;*******************
182   ;signal(semaphore)
182   ;*******************
182
182   ;before:  (B,C) = semaphore
182
182   ;if one or more coroutines are waiting,the first will be
182   ;activated - otherwise semaphore.sem is increased by 1
182
182   signal: di
183           inx  b
184           ldax b            ;A:=number of waiting coroutines
185           ana  a            ;test A
186           jnz  squeue
189
189                             ;no waiting coroutines:
189           dcx  b            ;(B,C):=semaphore.sem
190           ldax b
191           inr  a
192           stax b            ;sem:=sem+1
193           ret
194
194   squeue:                   ;at least one waiting coroutine:
194           call firstout     ;(D,E):=first CD from queue
197           call intactive
200           ei
201           ret
202
```

```
202
202
202   ;*****************
202   ;wait(semaphore)
202   ;*****************
202
202   ;before:  (B,C) = semaphore
202
202   ;if semaphore.sem > 0 the calling coroutine remains activated
202   ;and sem is decreased by 1, Otherwise the CD is put in the
202   ;semaphore-queue,
202
202   wait:   di                ;disable interrupts
203
203                             ;save LPC:
203           pop   d           ;(D,E):=LPC
204           lhld  running     ;(H,L):-CD of calling routine
207           inx   h
208           inx   h           ;(H,L):-CD,LPC
209           mov   m,e
210           inx   h
211           mov   m,d         ;CD,LPC:=LPC
212
212
212           ldax  b           ;A:=semaphore.sem
213           ana   a           ;test A
214           jz    wqueue      ;jump if sem=0
217
217           dcr   a           ;sem := sem - 1
218           stax  b
219           dcx   h
220           dcx   h
221           dcx   h           ;(H,L):-CD
222           xchg              ;(D,E):-CD
223           call  intactive   ;activate  CD
226           jmp   common
229
229   wqueue:                   ;the CD is inserted into the
229                             ;semaphore queue
229           inx   b           ;(B,C):-semaphore.length
230           dcx   h
231           dcx   h
232           dcx   h           ;(H,L):-CD
233           xchg              ;(D,E):-CD
234           call  into
237           jmp   common
240
```

```
240
240
240   ;*************************
240   ;send(msemaphore,message)
240   ;*************************
240
240   ;before:    (B,C) = msemaphore
240   ;           (H,L) = messagebuffer
240   ;
240   ;if a coroutine is waiting, the address of the messagebuffer
240   ;is placed in CD.mes and the coroutine is activated.
240   ;Otherwise the messagebuffer is inserted in the semaphore-
240   ;queue.
240   ;
240   ;msemaphore.sem =  1   : waiting messagebuffers
240   ;                  0   : empty queue
240   ;                 -1   : waiting coroutinedescriptions
240   send:   di
241
241           ldax b            ;A:=sem
242           inr  a
243           jz   cdwaiting    ;jump if sem = -1
246
246                             ;no waiting CD:
246           mvi  a,1          ;sem was 0 or 1, sem:=1
248           stax b
249           inx  b            ;(B,C):-msemaphore.length
250           xchg              ;(D,E):-messagebuffer
251           call into         ;into(MB,msemaphore)
254           ei
255           ret
256
256   cdwaiting:                ;at least one CD in queue
256           inx  b            ;(B,C):-msemaphore.length
257           ldax b
258           dcr  a            ;length:=lengh-1
259           jnz  s1           ;if length>0 sem remains -1
262
262           dcx  b            ;if length=0  sem:=0
263           stax b
264           inx  b
265
265   s1:     call firstout     ;(D,E):-CD from chain
268           mov  b,d
269           mov  c,e
270           inx  b
271           inx  b
272           inx  b
273           inx  b
274           inx  b            ;(B,C):-CD.mes
275           mov  a,l
276           stax b
277           inx  b
278           mov  a,h
279           stax b            ;CD.mes:=message
280           call intactive
283           ei
284           ret
285
```

```
285
285
285   ;*********************
285   ;receive(msemaphore)
285   ;*********************
285   ;
285   ;before:   (B,C) = msemaphore
285   ;
285   ;Since wait is a waitingpoint, the LPC must be saved in
285   ;calling CD.LPC,  If there is one or more messagebuffers
285   ;in queue, the first is released, and the address of the
285   ;messagebuffer is placed in calling CD.mes.
285   ;In this case the calling coroutine will remain active.
285   ;If there are no messages ready, the CD is inserted in
285   ;the semaphore-queue.
285   ;
285   ;msemaphore.sem = 1 :waiting message buffers
285   ;                 0 :empty queue
285   ;                -1 :waiting coroutines
285
285   receive:di
286           pop  d            ;get old PC from stack
287           lhld running      ;(H,L):=calling CD
290           push h            ;save  address on stack
291           inx  h
292           inx  h
293           mov  m,e
294           inx  h
295           mov  m,d          ;CD.LPC:= reactivation address
296
296           ldax b            ;A:=sem
297           dcr  a
298           jp   w2           ;jump if sem=1
301
301           mvi  a,255        ;sem:=-1
303           stax b
304           inx  b            ;(B,C):=msemaphore.length
305           pop  d            ;(D,E):=CD
306           call into         ;into(CD,msemaphorequeue)
309           jmp  common
312
312   w2:     inx  b            ;(B,C):=length
313           ldax b
314           dcr  a
315           mvi  a,1          ;if length > 1 ,sem:=1
317           jnz  w1
320           dcr  a            ;if length = 1 ,sem:=0
321
321   w1:     dcx  b            ;store sem
322           stax b
323           inx  b
324
324           call firstout     ;(D,E):=MB from queue
327           inx  h
328           inx  h            ;(H,L):=CD.mes
329           mov  m,e
330           inx  h
331           mov  m,d          ;CD.mes:=MB
332
332           pop  d            ;(D,E):=CD
333           call intactive
336           jmp  common
339
```

```
339
339
339     ;**************
339     ;intactive(CD)
339     ;**************
339     ;
339     ;before:    (D,E) = CD
339     ;
339     ;after:     (D,E) = CD
339     ;
339     ;intactive inserts the CD in activechain(CD,priority)
339
339
339
339     intactive:
339             lxi   b,active
342             mov   h,d
343             mov   l,e
344             inx   h
345             inx   h
346             inx   h
347             inx   h             ;(H,L):=CD,status=CD,priority
348             mov   a,m           ;A:=priority
349
349             mov   h,a
350             add   a
351             add   a
352             add   h             ;A:=5 * priority
353             add   c
354             mov   c,a
355             xra   a
356             adc   b
357             mov   b,a           ;(B,C):=ACTIVE + 5*priority
358
358             jmp    into         ;into(CD,activechain(priority))
361
```

```
361
361
361   ;*********************************
361   ;interruptroutine for the clock
361   ;*********************************
361   ;
361   ;All registers and status are unchanged on return.
361   ;The delays for the messages in the SLEEPING chain is
361   ;decreased by one and those messages whose delays becomes
361   ;0 are returned to the answersemaphores.
361   ;
361   iclock: push psw          ;save A and status
362           lda  sleeping+1
365           ana  a            ;A:=message count
366           jz   iclnosleep   ;jump if the chain is empty
369
369           push b
370           push d
371           push h            ;save the rest of the registers
372
372           lhld sleeping+2   ;(H,L):=first element
375           lxi  d,sleeping+2;(D,E):=sleeping.first
378   iclloop:                  ; A     = messagecount
378                             ;(D,E) = last examined
378                             ;(H,L) = next to examine
378           push psw
379           inx  h
380           inx  h            ;(H,L):=message.delay
381           mov  c,m
382           inx  h
383           mov  b,m
384           dcx  b            ;(B,C):=delay - 1
385           mov  a,b
386           add  c
387           jnz  iclstay
390           jc   iclstay      ;continue if B=C=0
393
393                             ;remove message from chain and
393           push d            ;return it to the answersemaphore
394           inx  h
395           mov  c,m
396           inx  h
397           mov  b,m          ;(B,C):=answersemaphore
398           dcx  h
399           dcx  h
400           dcx  h
401           dcx  h
402           mov  d,m
403           dcx  h            ;(H,L):=message
404           mov  e,m          ;(D,E):=next message
405           push d
406           call send         ;send(answersemaphore,message)
409
409           lxi  h,sleeping+1
412           dcr  m            ;decrease sleeping.length by 1
413           jnz  icl1
416           dcx  h
417           mvi  m,0          ;if the chain is empty sem:=0
419
419   icl1:   pop  h            ;(H,L):=next message
420           pop  d            ;(D,E):=old message
421
421           mov  a,l          ;old.link:=next message
422           stax d
423           mov  a,h
```

```
424             inx  d
425             stax d
426             dcx  d
427
427             pop  psw
428             dcr  a              ;decrease messagecount by 1
429             jnz  iclloop
432
432                                 ;no more messages and the last
432                                 ;message in the chain was removed
432             xchg                ;(H,L):=old message
433             shld  sleeping+4;sleeping,last:=old
436             jmp  iclrestore ;exit from loop
439
439                                 ;the delay is still positive:
439     iclstay:mov  m,b
440             dcx  h
441             mov  m,c            ;store delay
442
442             dcx  h
443             mov  d,m
444             dcx  h              ;(H,L):=message
445             mov  e,m            ;(D,E):=next message
446             xchg                ;(D,E) <-> (H,L)
447
447     iclcont:pop  psw
448             dcr  a              ;decrease messagecount by 1
449             jnz  iclloop        ;exit if messagecount is 0
452
452
452     iclrestore:
452             pop  h
453             pop  d
454             pop  b
455     iclnosleep:
455             pop  psw            ;restore registers and status
456             ei
457             ret
458
```

```
symbols used

    a             7             6
    b             0             6
    c             1             6
    d             2             6
    e             3             6
    h             4             6
cdwa            256             4
    k           458             7
    l             5             6
    m             6             6
iemp             47             4
acti              0             4
reac            139             4
send            240             4
firs             62             4
rele            121             4
slee             17             4
  s1            265             4
sque            194             4
femp             85             4
  sp              6             6
icl1            419             4
iclc            447             4
pass            164             4
icll            378             4
icln            455             4
iclo            361             4
runn             15             4
iclr            452             4
icls            439             4
inta            339             4
resf            160             4
into             23             4
rece            285             4
 psw              6             6
wque            229             4
wait            202             4
sign            182             4
comf            103             4
comm             87             4
como             90             4
  w1            321             4
  w2            312             4
end    92


**********************************

AFSLUTTET NORMALT
tid     :         25
linier  :        102
```