

## Answers

---

Q6-1.

a.

PDA: System Clock  
System Scheduler  
User interface

b.

Laser Printer: Graphics Engine  
Print Queue  
Print Engine

c.

Airplane: Fuel Injectors  
GPS  
Navigation Systems

Q6-2.

A CD player requires both periodic and aperiodic computations; periodic for digital filtering and aperiodic for user interface servicing.

Q6-3.

Using some multiple of 44.1 KHz would simplify analysis and provide adequate response. For instance, 44.1 Hz would probably work well.

Q6-4.

A cooperative context switch mechanism will require less overhead because there is no need for a scheduler.

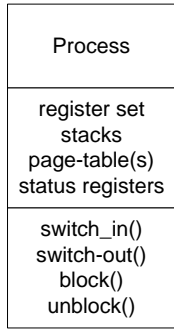
Q6-5.

A heavyweight process's activation record will contain information about the process's separate stack pointers (one for user space and one for kernel space) as well as information about the process's page directories and page tables.

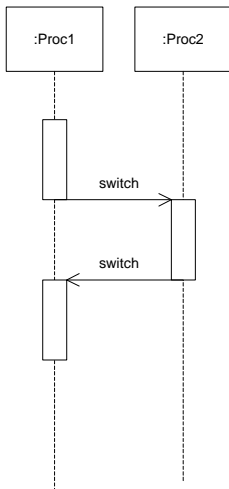
Q6-6.

Code fragment 'A' would be acceptable.

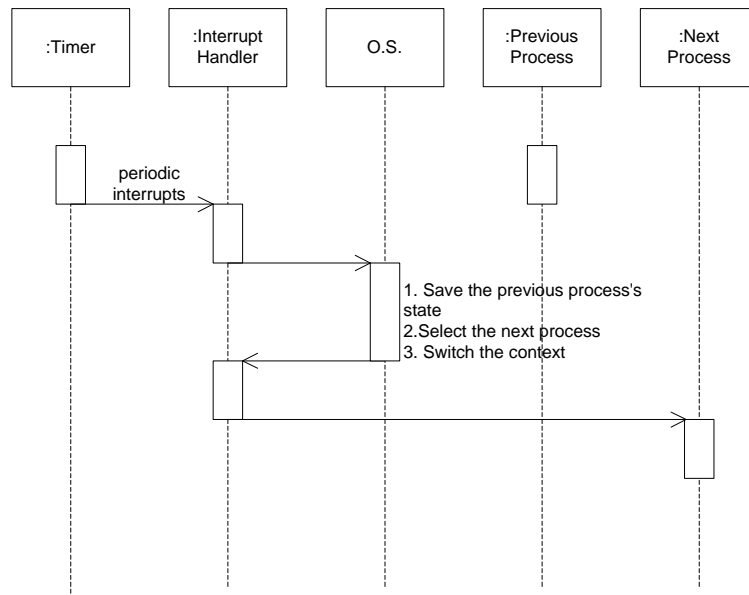
Q6-7.



Q6-8.



Q6-9.



Q6-10.

A combination of :

- 1.) System scheduler overhead
- 2.) Context switch overhead

provide the lower bound on the timer interrupt period.

Q6-11.

A combination of:

- 1.) Deadlines of the processes
- 2.) Periods of the processes
- 3.) Execution time of the processes

provide the upper bound on the timer interrupt period.

Q6-12.

Checked in, paid, #of keys given, room #, type of room, names of people in the room, credit card #, activation of phone line, state of wet bar.

Q6-13.

The ready state indicates that the process is “ready “ to run and that the process has a chance of being switched in by the scheduler. In contrast, the waiting state indicates that the process is “waiting” on some type of system event and that it

should be scheduled to run. Generally, the process is waiting on some type of data I/O.

Q6-14.

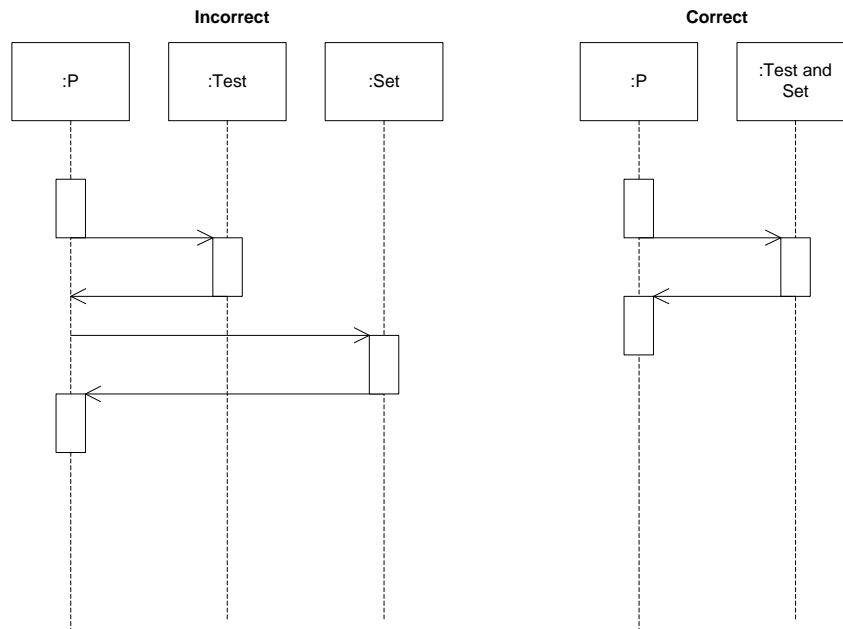
- a. Blocking IPC = disk data request
- b. Non-blocking IPC = a keyboard handler

Q6-15.

Assuming that we have one semaphore (SEMAPHORE) in the system:

```
P(){  
    while(test_and_set(SEMAPHORE) == LOCKED);  
}  
  
V(){  
    *SEMAPHORE = UNLOCKED;  
}
```

Q6-16.



Q6-17.

- 10 units (LCM of 2, 5, 10)
- 20 units (LCM of 2, 4, 5, 10)
- 60 units (LCM of 3, 4, 5, 6, 10)

Q6-18.

Currently, the LCM(200, 10, 40, 50) is 200. Hence, P2 executes 20 times, P1 executes 1 times with 2 instances, P3 executes 5 times, and P4 executes 4 times. Thus,  $20(1) + 2(4) + 5(2) + 4(6) < 200$ . Since the inequality is satisfied, we can add the second instance of P1 and still meet all deadlines.

Q6-19.

Currently, the LCM(10, 100, 20, 50, 25) = 100. Hence, in 100 unit interval we have:

$$10 \cdot P1 + 1 \cdot P1 + 5 \cdot P3 + 2 \cdot P4 + 4 \cdot P5$$

$$\rightarrow 10 + 18 + 10 + 10 + 4x = 100$$

$$4x = 52$$

$$x = 13$$

Q6-20.

- a. P1's critical instance is when P1 is just starting.
- b. P2's critical instance is when P2 is ready to go, but P1 is just starting.
- c. P3's critical instance is when P3 is ready to go and, at the same time, both P1 and P2 are ready to go.

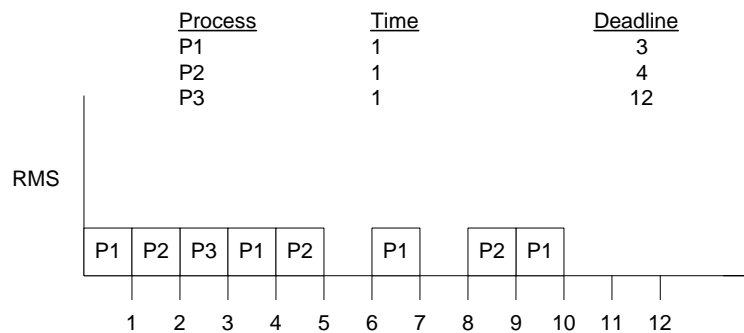
Q6-21.

$$\text{LCM}(5, 10, 25, 50) = 50$$

- a. P1 will execute 10 times over the 50 unit interval. Hence, it requires  $10(1)/50 = 20\%$
- b. P2 will execute 5 times over the 50 unit interval. Hence, it requires  $5(2)/50 = 20\%$
- c. P3 will execute 2 times over the 50 unit interval. Hence, it requires  $2(2)/50 = 8\%$
- d. P5 will execute 1 time over the 50 unit interval. Hence, it requires  $1(5)/50 = 10\%$

Q6-22.

a.

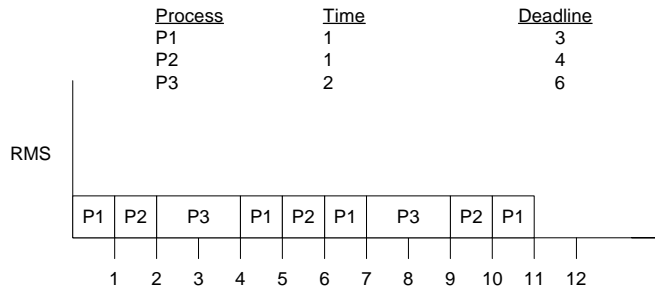


b.

In this case, the EDF schedule is the same as the RMS schedule.

Q6-23.

a.



b.

P1 has deadlines at 3,6,9,12

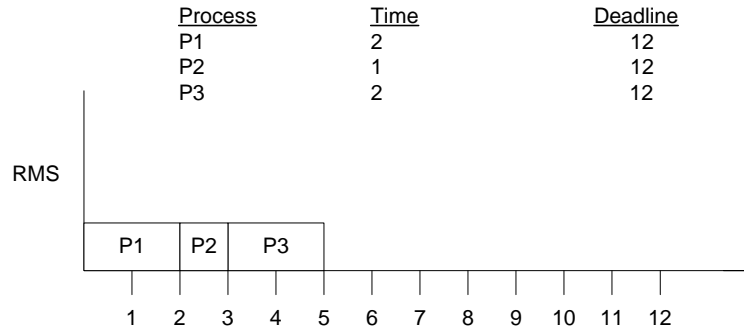
P2 has deadlines at 4,8,12

P3 has deadlines at 6,12

<i>Time</i>	<i>Running Process</i>	<i>Deadlines</i>
0	P1	P2, P3
1	P2	
2	P3	P1
3	P3	P1, P2
4	P1	
5	P2	P1, P3
6	P1	
7	P3	P2
8	P3	P2, P1
9	P2	
10	P1	
11		

Q6-24.

- a. All processes have the same deadline, so we must break ties for priority arbitrarily. If we choose the priorities as shown here, we get the same answer for parts a) and b).



If we assume that we P1 and P2/P3 can execute on different iterations, then a priority assignment that doesn't give P1 the highest priority will require two periods to complete. If everything must complete on the same period, then we must use the above priority assignment or we will violate the deadline.

- b. The priority assignment with P1 highest priority gives the shortest execution interval.

Q6-25.

- a.

P1 lasts 2 and has deadlines at: 30, 60, 90, 120  
 P2 lasts 5 and has deadlines at: 40, 80, 120  
 P3 lasts 7 and has deadlines at: 120  
 P4 lasts 5 and has deadlines at: 60, 120  
 P5 lasts 1 and has deadlines at: 15, 30, 45, 60, 75, 90, 105, 120

Time	Running Process	Status
0	P5	
1	P1	
2	P1	
3	P2	
4	P2	
5	P2	
6	P2	
7	P2	
8	P3	
9	P3	
10	P3	
11	P3	
12	P3	
13	P3	
14	P3	
15	P5	
16	P4	

17	P4
18	P4
19	P4
20	P4
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	P5
31	P1
32	P1
33	
34	
35	
36	
37	
38	
39	
40	P2
41	P2
42	P2
43	P2
44	P2
45	P5
46	
47	
48	
49	
50	
51	
52	
53	
54	
55	
56	
57	
58	
59	
60	P5
61	P1
62	P1
63	P4
64	P4
65	P4
66	P4

67	P4
68	
69	
70	
71	
72	
73	
74	
75	P5
76	
77	
78	
79	
80	P2
81	P2
82	P2
83	P2
84	P2
85	
86	
87	
88	
89	
90	P5
91	P1
92	P1
93	
94	
95	
96	
97	
98	
99	
100	
101	
102	
103	
104	
105	P5
106	
107	
108	
109	
110	
111	
112	
113	
114	
115	
116	

117  
118  
119

b.

P1 lasts 2 and has deadlines at: 30, 60, 90, 120

P2 lasts 5 and has deadlines at: 40, 80, 120

P3 lasts 7 and has deadlines at: 120

P4 lasts 5 and has deadlines at: 60, 120

P5 lasts 1 and has deadlines at: 15, 30, 45, 60, 75, 90, 105, 120

Time	Running Process	Status
0	P5	
1		switch
2	P1	
3	P1	
4		switch
5	P2	
6	P2	
7	P2	
8	P2	
9	P2	
10		switch
11		
12		
13		
14		
15	P5	
16		switch
17	P4	
18	P4	
19	P4	
20	P4	
21	P4	
22		switch
23		
24		
25		
26		
27		
28		
29		
30	P5	
31		switch
32	P1	
33	P1	
34		switch
35	P3	
36	P3	
37	P3	

38	P3	
39	P3	
40	P3	
41	P3	
42		switch
43		
44		
45	P5	
46		switch
47	P2	
48	P2	
49	P2	
50	P2	
51	P2	
52		switch
53		
54		
55		
56		
57		
58		
59		
60	P5	
61		switch
62	P1	
63	P1	
64		switch
65	P4	
66	P4	
67	P4	
68	P4	
69	P4	
70		switch
71		
72		
73		
74		
75	P5	
76		switch
77		
78		
79		
80	P2	
81	P2	
82	P2	
83	P2	
84	P2	
85		switch
86		
87		

88		
89		
90	P5	
91		switch
92	P1	
93	P1	
94		switch
95		
96		
97		
98		
99		
100		
101		
102		
103		
104		
105	P5	
106		switch
107		
108		
109		
110		
111		
112		
113		
114		
115		
116		
117		
118		
119		

Q6-26.

a.

P1 lasts 2 and has deadlines at: 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100

P2 lasts 1 and has deadlines at: 10, 20, 30, 40, 50, 60, 70, 80, 90, 100

P3 lasts 2 and has deadlines at: 20, 40, 60, 80, 100

P4 lasts 10 and has deadlines at: 50, 100

P5 lasts 7 and has deadlines at: 100

Time	
0	P1
1	P2
2	P3
3	P3
4	P4
5	P1
6	P4

7	P4
8	P4
9	P4
10	P1
11	P2
12	P4
13	P4
14	P4
15	P1
16	P4
17	P4
18	P5
19	P5
20	P1
21	P2
22	P3
23	P3
24	P4
25	P1
26	P4
27	P4
28	P4
29	P4
30	P1
31	P2
32	P4
33	P4
34	P4
35	P1
36	P4
37	P4
38	P5
39	P5
40	P1
41	P2
42	P3
43	P3
44	P5
45	P1
46	P5
47	P5
48	
49	
50	P1
51	P2
52	P4
53	P4
54	P4
55	P1
56	P4

57	P4
58	P4
59	P4
60	P1
61	P2
62	P3
63	P3
64	P4
65	P1
66	P4
67	P4
68	
69	
70	P1
71	P2
72	
73	
74	
75	P1
76	
77	
78	
79	
80	P1
81	P2
82	P3
83	P3
84	
85	P1
86	
87	
88	
89	
90	P1
91	P2
92	
93	
94	
95	P1
96	
97	
98	
99	

We have 42 context switches (we do not count a switch either into or out of idle time as a context switch).

b. Here is the EDF schedule:

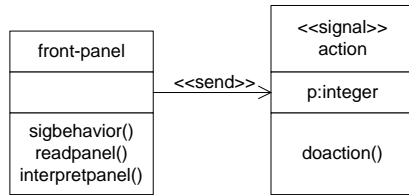
Time	Executing process	Earliest deadline
------	-------------------	-------------------

0	P1	P1
1	P2	P1
2	P3	P1
3	P3	P1
4	P4	P1
5	P1	P1, P2
6	P4	P1, P2
7	P4	P1, P2
8	P4	P1, P2
9	P4	P1, P2
10	P1	P1
11	P2	P1
12	P4	P1
13	P4	P1
14	P4	P1
15	P1	P1, P2, P3
16	P4	P1, P2, P3
17	P4	P1, P2, P3
18	P5	P1, P2, P3
19	P5	P1, P2, P3
20	P1	P1
21	P2	P1
22	P3	P1
23	P3	P1
24	P5	P1
25	P1	P1, P2
26	P5	P1, P2
27	P5	P1, P2
28	P5	P1, P2
29	P5	P1, P2
30	P1	P1
31	P2	P1
32		P1
33		P1
34		P1
35	P1	P1, P2, P3
36		P1, P2, P3
37		P1, P2, P3
38		P1, P2, P3
39		P1, P2, P3
40	P1	P1
41	P2	P1
42	P3	P1
43	P3	P1
44		P1
45	P1	P1, P4
46		P1, P4
47		P1, P4
48		P1, P4
49		P1, P4

50	P1	P1
51	P2	P1
52	P4	P1
53	P4	P1
54	P4	P1
55	P1	P1, P2, P3
56	P4	P1, P2, P3
57	P4	P1, P2, P3
58	P4	P1, P2, P3
59	P4	P1, P2, P3
60	P1	P1
61	P2	P1
62	P3	P1
63	P3	P1
64	P4	P1
65	P1	P1, P2
66	P4	P1, P2
67	P4	P1, P2
68		P1, P2
69		P1, P2
70	P1	P1
71	P2	P1
72		P1
73		P1
74		P1
75	P1	P1, P2, P3
76		P1, P2, P3
77		P1, P2, P3
78		P1, P2, P3
79		P1, P2, P3
80	P1	P1
81	P2	P1
82	P3	P1
83	P3	P1
84		P1
85	P1	P1, P2
86		P1, P2
87		P1, P2
88		P1, P2
89		P1, P2
90	P1	P1
91	P2	P1
92		P1
93		P1
94		P1
95	P1	P1, P2, P3, P4, P5
96		P1, P2, P3, P4, P5
97		P1, P2, P3, P4, P5
98		P1, P2, P3, P4, P5
99		P1, P2, P3, P4, P5



Q6-27.



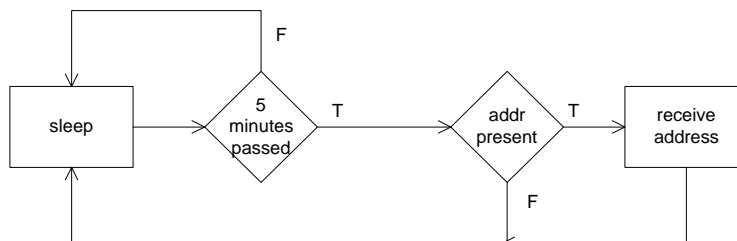
Q6-28.

- a. Shared memory. Because each successive filter must act on all of the data that is processed by each previous filter, it makes sense to have a section of shared memory that one is able to write to and the other is able to read from.
- b. Message passing. Because the action of displaying the user menus will probably occur relatively rarely, it makes sense to use a message driven interface. In this way, the user interface process will only do work in it is needed to, thus saving as much processing power and time for the decoder process.
- c. Message passing. Because the two processes will probably need to interact relatively infrequently, it makes sense to use a message driven approach.

Q6-29.

You could control the footprint through the use of cache partitioning, and by using a virtually addressed cache and controlling the virtual base address of each process.

Q6-30.



Q6-31.

You wouldn't. Because the ADPCM method encodes the difference between successive samples, and each sample of a DC signal is the same, there is no real

good way to encode with the coding alphabet  $\{-3, -2, -1, 1, 2, 3\}$ . Encoding with each successive value alternating between  $-1$  and  $1$  might be acceptable but would essential be decompressed to yield an AC signal with a very small peak-to-peak difference.