1. (a) In multi-programming systems, I/O operations are going in parallel with the CPU computations, thus, I/O interrupt is needed at the completion of I/O. It also needs Hardware interrupts for memory problems or disk problems. A timer interrupt is also essential for round-robin type of scheduling in time sharing systems.

(b) In any multi-programming systems, I/O operations are going in parallel with the CPU computations, thus, I/O interrupt is needed at the completion of I/O. It also needs Hardware interrupts for memory problems or disk problems. Timer interrupt is not needed in multi-programming batch systems.

(c) No interrupt is needed in a pure batch system because the processes are being executed sequentially. The system waits when using memory or disk so no need to have interrupts either.

(d) Personal computers are the same as Multi-programming time sharing systems. It actually handles multiple tasks from the user as well as the system in a time shared manner.

(e) The system is dedicated to the pressure computation, nothing else. So, there is no need for interrupt.

(f) In a real time system that handles multiple events, the events are prioritized and interrupts are needed for high priority events to interrupt low priority tasks.

2. (a) a = 13, b = 26 from child
    a = 7, b = 4 from parent

(b) A B C D E (a=7, b=4)
    A B D C E (a= 27, b=26)
    A B D E C (a= 7, b=14)
    A D B C E (a= 29, b=26)
    A D B E C (a= 7, b=14)
    A D E B C (a= 12, b=24)
    D A B C E (a= 29, b=26)
    D A B E C (a= 7, b=14)
    D A E B C (a= 12, b=24)
    D E A B C (a= 13, b=26)

3. (a) Addresses   Contents
    0x00001018  0x0100E308
    0x0000101C  0x0300E30C
0x00001020  0x0200E310
(1st byte: opcode (e.g., 0x01), remaining 3 bytes are address of data)

0x0000E308  0x00000014 ; (a=20=0x14)
0x0000E30C  0x00000088 ; (b=136=0x88)
0x0000E310  0x00000000 ; (c=0=0x00, or it can be anything)

(b)

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC → MAR</td>
<td>0x00001018</td>
</tr>
<tr>
<td>M → MBR</td>
<td>0x0100E308</td>
</tr>
<tr>
<td>MBR → IR</td>
<td>0x0100E308</td>
</tr>
<tr>
<td>IR (operand) → MAR</td>
<td>0x0000E308</td>
</tr>
<tr>
<td>M → MBR</td>
<td>0x00000014</td>
</tr>
<tr>
<td>MBR → AC</td>
<td>0x00000014</td>
</tr>
<tr>
<td>PC+4 → PC</td>
<td>0x0000101C</td>
</tr>
<tr>
<td>PC → MAR</td>
<td>0x0000101C</td>
</tr>
<tr>
<td>M → MBR</td>
<td>0x0300E30C</td>
</tr>
<tr>
<td>MBR → IR</td>
<td>0x0300E30C</td>
</tr>
<tr>
<td>IR (operand) → MAR</td>
<td>0x0000E30C</td>
</tr>
<tr>
<td>M → MBR</td>
<td>0x00000088</td>
</tr>
<tr>
<td>MBR + AC → AC</td>
<td>0x0000009C</td>
</tr>
<tr>
<td>PC+4 → PC</td>
<td>0x00001020</td>
</tr>
<tr>
<td>PC → MAR</td>
<td>0x00001020</td>
</tr>
<tr>
<td>M → MBR</td>
<td>0x0200E310</td>
</tr>
<tr>
<td>MBR → IR</td>
<td>0x0200E310</td>
</tr>
<tr>
<td>IR (operand) → MAR</td>
<td>0x0000E310</td>
</tr>
<tr>
<td>AC → MBR</td>
<td>0x0000009C</td>
</tr>
<tr>
<td>MBR → M</td>
<td>0x0000009C</td>
</tr>
<tr>
<td>PC+4 → PC</td>
<td>0x00001024</td>
</tr>
</tbody>
</table>

4.

At time 22:
- P1: blocked for I/O
- P3: blocked for I/O
- P5: ready/running
- P7: blocked for I/O
- P8: ready/running

At time 37
- P1: ready/running
- P3: ready/running
- P5: blocked suspend
- P7: blocked for I/O
- P8: ready/running

At time 47
5.
-- Event: user runs the program → ready
Program P;
{
  -- Event: scheduled to run → running (context switch) (program running in user mode)
     read (x);
  -- Event: I/O statement → blocked
     Before blocked, switch to kernel mode to handle read
     Then, switch to execute another process (context switch)
  -- Event: scheduled to run → running (context switch)
     compute factorial of x;
     print (x);
  -- Event: I/O statement → blocked
     Before blocked, switch to kernel mode to handle print
     Then, switch to execute another process (context switch)
  -- Event: scheduled to run → running (context switch)
     pid = fork ();
  -- Event: system call → Kernel trap, switch to kernel mode to handle fork (context switch)
     OS creates the child process, put it in ready queue
     Switch back to run parent process in user mode

     if (pid == 0)
       -- Event: scheduled to run child process → running (context switch)
         {   read (y);
       -- Event: I/O statement → blocked
         Before blocked, switch to kernel mode to handle read
         Then, switch to execute another process (context switch)

       -- Event: scheduled to run → running (context switch)
         compute factorial of y;
         print (y);
       -- Event: I/O statement → blocked
         Before blocked, switch to kernel mode to handle print
         Then, switch to execute another process (context switch)

       -- Event: scheduled to run → running (context switch)
         Child process terminates, switch to another process
     }
}
else /* parent process */
  wait (pid);

-- Event: system call → blocked
Kernel trap, switch to kernel mode to handle wait
OS puts parent process in waiting queue
Switch to execute another process (context switch)

  -- while waiting, child process’s “read (y)” is done
  CPU switches to interrupt handler and put child process in ready state (partial context switch)
  Note: CPU is running another process, not the parent process

  -- while waiting, child process’s “print (y)” is done
  CPU switches to interrupt handler and put child process in ready state (partial context switch)

} Program Q;
{
  initialize matrix A using random number generator;
  -- I/O interrupt: P’s “read (x)” operation is done
  CPU switches to interrupt handler and put P in ready state (partial context switch)
  CPU continue to execute the original running process

  initialize matrix B using random number generator;
  -- I/O interrupt: P’s “print (x)” operation is done
  CPU switches to interrupt handler and put P in ready state (partial context switch)
  CPU continue to execute the original running process

  create a thread to compute A*B;
  Nothing, since it is a user level thread
}

6.

<table>
<thead>
<tr>
<th>Job Number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrival Time</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>Service Time</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

For (a) Shortest job first, non-preemptive
<table>
<thead>
<tr>
<th>Job #</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrival time</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>Finish time</td>
<td>6</td>
<td>8</td>
<td>14</td>
<td>10</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>Response time</td>
<td>6</td>
<td>7</td>
<td>10</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Service time</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Average response time = \( \frac{6+7+10+3+4+5}{6} = \frac{35}{6} = 5.83 \)
Average (response time/service time) = \( \frac{1+7/2+10/4+3/2+4/4+5/2}{6} = 2 \)

For (b) Highest remaining response ratio first, preemptive

<table>
<thead>
<tr>
<th>Time</th>
<th>Job 1</th>
<th>Job 2</th>
<th>Job 3</th>
<th>Job 4</th>
<th>Job 5</th>
<th>Job 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \frac{0+5}{5} = 1 )</td>
<td>( \frac{0+2}{2} = 1 )</td>
<td>Can choose either, choose 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>( \frac{0+2}{2} = 1 )</td>
<td>( \frac{3+2}{2} = 2.5 )</td>
<td>Job 3 = ( \frac{0+4}{4} = 1 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>( \frac{2+2}{2} = 2 )</td>
<td>Job 3 = ( \frac{2+4}{4} = 1.5 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Job 1 = ( \frac{2+1}{1} = 3 )</td>
<td>Job 3 = ( \frac{3+4}{4} = 1.75 )</td>
<td>Job 4 = 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Job 3 = ( \frac{4+4}{4} = 2 )</td>
<td>Job 4 = ( \frac{1+2}{2} = 1.5 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>only Job 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>only Job 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Job 5 = 1; Job 6 = 1; Can choose either, choose 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Job #</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrival time</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>Finish time</td>
<td>8</td>
<td>6</td>
<td>12</td>
<td>14</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>Response time</td>
<td>8</td>
<td>5</td>
<td>8</td>
<td>7</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Service time</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Average response time = \( \frac{8+5+8+7+4+5}{6} = \frac{37}{6} = 6.17 \)
Average (response time/service time) = \( \frac{8/6+5/2+2+7/2+1+5/2}{6} = 2.14 \)
For (c) Round robin (time quantum = 1), assume that jobs terminate after new arrivals

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td></td>
<td>J1</td>
<td>J2</td>
<td>J1</td>
<td>J3</td>
<td>J1</td>
<td>J3</td>
<td>J4,J1</td>
<td>J3,J4</td>
<td>J1,J3</td>
<td>J4,J1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Job # | 1 | 2 | 3 | 4 | 5 | 6
---|---|---|---|---|---|---
Arrival time | 0 | 1 | 4 | 7 | 17 | 18
Finish time | 12 | 4 | 14 | 13 | 23 | 21
Response time | 12 | 3 | 10 | 6 | 6 | 3
Service time | 6 | 2 | 4 | 2 | 4 | 2

Average response time = (12+3+10+6+6+3)/6 = 40/6 = 6.67
Average (response time/service time) = (2+3/2+10/4+3+6/4+3/2)/6 = 12/6 = 2

For (d) Multilevel feedback queue (number of queue levels = 4, time quantum = 1)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
<tr>
<td>Q2</td>
<td>1,2</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q3</td>
<td>1,1,1</td>
<td>1,3</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q4</td>
<td>1,1,1</td>
<td>1,3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Job # | 1 | 2 | 3 | 4 | 5 | 6
---|---|---|---|---|---|---
Arrival time | 0 | 1 | 4 | 7 | 17 | 18
Finish time | 14 | 4 | 12 | 9 | 23 | 21
Response time | 14 | 3 | 8 | 2 | 6 | 3
Service time | 6 | 2 | 4 | 2 | 4 | 2

Average response time = (14+3+8+2+6+3)/6 = 36/6 = 6
Average (response time/service time) = \( \frac{7/3+3/2+2+1+3/2+3/2}{6} = 1.58 \)

7. There will be a feasible schedule to allow the execution of all jobs. More formally, we can use rate monotonic theorem to prove the schedulability of the tasks. If \( U < n \left( \frac{2^{1/n}}{n} - 1 \right) \), then the jobs are schedulable, where \( U \) is CPU utilization and \( n \) is the number of jobs. \( U = \frac{2}{10} + \frac{1}{6} + \frac{2}{12} + \frac{1.5}{9} = 0.7 < n \left( \frac{2^{1/n}}{n} - 1 \right) = 0.7568 \). So, the jobs are schedulable.

LCD for period of time = 180 sec, all jobs can be scheduled as follows
Job 1: # of jobs that can be serviced = \( \frac{180}{10} = 18 \)
Job 2: # of jobs that can be serviced = \( \frac{180}{6} = 30 \)
Job 3: # of jobs that can be serviced = \( \frac{180}{12} = 15 \)
Job 4: # of jobs that can be serviced = \( \frac{180}{9} = 20 \)
System throughput = \( \frac{(18+30+15+20)}{180} = \frac{83}{180} = 0.46 \)

In fact, if you take any period to compute this, it is fine.

8. At time 0: P enters the system, \( U_0 = 0, C_0 = 0, P_1 = 60 \)
   At time 30: \( U_1 = 0 \Rightarrow C_1 = 0, P_2 = 60 \)
   At time 70: \( U_2 = 40, C_1 = 0 \Rightarrow C_2 = 20, P_3 = 70 \) // next two time slices are 60ms each
   At time 130: \( U_3 = 0, C_2 = 20 \Rightarrow C_3 = 10 \)
   At time 190: \( U_4 = 0, C_3 = 10 \Rightarrow C_4 = 5 \)
   At time 210: \( U_5 = 20, C_4 = 5 \Rightarrow C_5 = 12, P_6 = 66 \)
9.

lock(lck1);
lock(lck2);
P1: A; B; unlock(lck1); lock(lck2); C;
P2: lock(lck1); D; unlock(lck1); unlock(lck2); E;

10.
P1: set blocked[1] = true;
P1: check and find turn is not 1 ( while (turn != id) do )
P1: check and find blocked[0] = false, exit inner loop ( while (blocked[1–id]) do; )
P0: set blocked[0] = true;
P0: check and find turn is 0, exit outer loop ( while (turn != id) do )
P0: enter critical section
P1: set turn = 1;
P1: check and find turn is 1, exit outer loop ( while (turn != id) do )
P1: enter critical section