Problem 1. Operating System Issues (5 points)

The Yield() SVC can be used in user-mode programs on a time-sharing system to give up the remainder of their current time slice. The kernel implementation of the Yield simply calls the kernel Scheduler() routine to choose another process to execute. When the yielding process is next scheduled, user-mode execution resumes with the instruction following the Yield() SVC.

Complete the code for the handler for a new SVC, YieldN(), which expects a numeric value, N, in the user’s R0 and behaves as if the user program had contained N consecutive Yield() SVCs. When execution resumes following the completion YieldN(), R0 should contain 0.

```c
YieldN_h() {
    if (User.Regs[0] > 0) {
        User.Regs[0] -= 1;     // decrement count
        User.Regs[XP] -= 4;    // arrange to re-execute SVC
        Scheduler();           // yield to next process
    } else {
        User.Regs[0] = 0;     // [optional] final value for R0
    }
}
```
Problem 2. Virtual Memory (7 points)

The micro-Beta has a 12-bit virtual address, an 11-bit physical address and uses a page size of 256 (\(= 2^8\)) bytes. The micro-Beta has been running for a while and at the current time the page map has the contents shown on the right.

(A) (1 point) Assuming each page map entry contains the usual dirty (D) and resident (R) bits, what is the total size of the page map in bits?

\[\text{Size of page map (bits): } 80\]

(B) (2 points) The following instruction, located at virtual address 0x0BA, is about to be executed.

\[\text{LD(R31, 0x2C8, R0)}\]

When the instruction is executed, what main memory locations are accessed by the instruction fetch and then the memory access initiated by the LD? Use the page map shown to the right. Assume the LRU page is virtual page 0xE.

**Physical address for instruction fetch:** 0x2BA

**Physical address for data read by LD instruction:** 0x4C8

(C) (4 points). A few instructions later, the following instruction, located at virtual address 0x0CC, is executed:

\[\text{ST(BP, -4, SP)} \quad // \text{current value of SP = 0x604}\]

Please mark up the page map to show its contents after the ST has been executed. Use the page map shown to the right. Assume the LRU page is virtual page 0xE.

Remember to show any changes to the dirty and resident control bits as well as updates to the physical page numbers. If an entry in the page map no longer matters, please indicate that by replacing it with “— 0 —” for the D, R and PPN entries.

**Show updated contents of page map**

\[0x0CC \rightarrow \text{VPN 0} \rightarrow \text{PPN 2}\]
\[0x604 - 4 = 0x600 \rightarrow \text{VPN 6} \rightarrow \text{page fault}\]
reusing LRU page (VPN 0xE)
Problem 3. Interrupts and Real Time (10 points)

Ben Bitdiddle has designed a wrist device called the BenBit to measure how long you’ve been tooling away without getting up and moving around. The BenBit runs a real-time operating system supporting three tasks whose handlers are executed in response to periodic requests. Each task has a period (time between requests), a service time (time it takes to run its handler), and a deadline (maximum time allowed to elapse between request and completion of the handler).

<table>
<thead>
<tr>
<th>Task</th>
<th>Period (ms)</th>
<th>Service time (ms)</th>
<th>Deadline (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check Accelerometer (CA)</td>
<td>80</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>Update Display (UD)</td>
<td>200</td>
<td>?</td>
<td>200</td>
</tr>
<tr>
<td>Determine heart rate (DHR)</td>
<td>60</td>
<td>10</td>
<td>50</td>
</tr>
</tbody>
</table>

Ben is trying to figure out whether to use a weak or strong priority system to manage task execution. For each of the questions below, please fill the answers for both types of priority system. For both the weak and strong priority system assume the task priority is CA > DHR > UD, i.e., CA has the highest priority and UD the lowest. If a calculation requires the service time for UD, use your answer for part (A).

<table>
<thead>
<tr>
<th></th>
<th>Using Weak Priorities</th>
<th>Using Strong Priorities</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A) What is the maximum service time for UD handler that still allows all deadlines to be met (in ms)?</td>
<td>20ms otherwise CA will miss deadline</td>
<td>100ms = 200 – 3<em>20 – 4</em>10</td>
</tr>
<tr>
<td>(B). What fraction of the time will the processor spend idle? (%)</td>
<td>48.33% =1-(20/80)-(20/200)-(10/60))</td>
<td>8.33% =1-(20/80)-(100/200)-(10/60)</td>
</tr>
<tr>
<td>(C) What is the worst-case completion time for the CA task (in ms)?</td>
<td>40ms = UD_{ST} + CA_{ST}</td>
<td>20ms = CA_{ST}</td>
</tr>
<tr>
<td>(D) What is the worst-case completion time for the UD task (in ms)?</td>
<td>50ms = DHR_{ST} + CA_{ST} + UD_{ST}</td>
<td>200ms = 3<em>CA_{ST} + 4</em>DHR_{ST} + UD_{ST}</td>
</tr>
<tr>
<td>(E) What is the worst-case completion time for the DHR task (in ms)?</td>
<td>50ms = UD_{ST} + CA_{ST} + DHR_{ST}</td>
<td>30ms = CA_{ST} + DHR_{ST}</td>
</tr>
</tbody>
</table>
Problem 4. Semaphores (8 points)

Schro Dinger has a company that produces pairs of entangled particles, which are then packaged and sent to manufacturers of quantum computers. Since it’s a complicated process, there are multiple machines that produce particle pairs; each machine runs the Producer code shown below.

The completed particle pairs are placed in the particle buffer, where they take up 2 of the buffer locations. There’s a single packaging machine that takes a particle pair from the particle buffer and prepares it for shipment; the packing machine runs the Consumer code shown below.

To prevent any violations of the boundary conditions the following rules must be followed:

1. A production machine can only place a particle pair in the buffer if there are two spaces available.
2. The particle pair must be stored in consecutive buffer locations, i.e., a particle from some other production machine can’t appear between the particles that make up the pair.
3. The capacity of the buffer (100 particles, or 50 particle pairs) can’t be exceeded.
4. The packaging machine breaks if it accesses the buffer and finds it empty – it should only proceed when there are at least two particles in the buffer.

Schro has heard of semaphores but is unsure how to use them to ensure the rules are followed.

- Please insert the appropriate semaphores, WAITs, and SIGNALs into the Producer and Consumer code to ensure correct operation and to prevent deadlock.
- Be sure to indicate initial values for any semaphores you use.
- Remember: **there are multiple producers and a single consumer**!
- For full credit, use a minimum number of semaphores and don’t introduce unnecessary precedence constraints.

<table>
<thead>
<tr>
<th>Shared Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>particle buffer[100]; // holds 100 particles</td>
</tr>
<tr>
<td>Semaphores and initial values: P=0, S=50, M=1</td>
</tr>
</tbody>
</table>

```
### Producer
PLoop:

Produce pair P1, P2

WAIT(S); WAIT(M)
Place P1 in buffer

Place P2 in buffer

SIGNAL(M); SIGNAL(P)
Go to PLoop

### Consumer
CLoop:

WAIT(P)
Fetch P1 from buffer

Fetch P2 from buffer

SIGNAL(S)
Package and ship

Go to CLoop
```

P = # of particle pairs in buffer, enforces rule 4
S = # of pair spaces in buffer, enforces rules 1 and 3
M = mutual exclusion lock, enforces rule 2 (when there are multiple producers)