General instructions

- This exam has 8 assignments and 5 pages, including this one. Read all assignments carefully and completely before you begin.
- It is recommended that you use a new sheet for each assignment. Number all your sheets, and mark each sheet on top with your exam ID and the course code.
- You may answer in either English or Swedish.
- Write clearly. Unreadable text will be ignored.
- Be precise in your statements. Unprecise formulations may lead to a reduction of points.
- Motivate clearly all statements and reasoning.
- Explain calculations and solution procedures.
- The assignments are not ordered according to difficulty.
- The exam is designed for 40 points. You may thus plan about 5 minutes per point.
- Grading: U, 3, 4, 5. The preliminary threshold for passing is 20 points.

Students in international master programs and exchange students will receive ECTS grades. Due to the anonymization of written exam correction, ECTS grades will be set by the central administration where appropriate, following the Linköping university rule for translation from swedish grades (5=A, 4=B, 3=C, U=FX).
1. (6 p.) **Interrupts, processes and threads**

(a) (i) What is the purpose of the kernel mode (aka. supervisor mode, monitor mode) bit? (1p)

(ii) Where and when is it set and cleared? (1p)

(b) What is a **process control block** (PCB)? What is its purpose? And what data does it contain (at least 4 relevant items are expected)? (2p)

(c) The Cell BE, a heterogeneous multicore processor, contains one PowerPC core as master processor (running Linux) and eight slave processors (SPEs). The master processor can execute ordinary PowerPC programs and is also used to coordinate execution of the SPEs. The SPEs have basically no operating system and, as they have a different instruction set, need a separate program code with their own main function.

The master processor uses several API functions to control SPE process execution, in particular (simplified)

```
spe_ctxt_t ctx = spe_context_create( ... );
```

to prepare a new SPE process for the SPE program and return a handle `ctx` to it, and

```
spe_context_run ( ctx, ... );
```

that, when called by the master program, starts the new process on a free SPE that executes the SPE program’s main function, and blocks (i.e., program control does not return from the call) until that SPE process has terminated.

How can the master program start multiple SPE processes that run in parallel? Write C (pseudo)code for a master program that starts 8 SPE processes so that they run in parallel. Explain your code carefully. (2p)

2. (7 p.) **Synchronization**

**Money transfer between bank accounts**

A bank administers $N$ accounts, which are, for simplicity, numbered 0 to $N - 1$ and stored in an array:

```c
struct {
    float balance;  // current balance value
    unsigned int owner;  // ...
} account[N];
```

For a transfer of amount $x > 0$ from account $i$ to account $j$ (where $i \neq j$), the bank software uses the following routine:
void transfer( float x, unsigned int i, unsigned int j )
{
    if (account[i].balance >= x) { // avoid negative balance
        account[i].balance -= x;
        account[j].balance += x;
    }
    else errmsg("Transfer denied: insufficient balance");
}

Initially, this bank program was executed as a single-threaded process that was processing a batch of transfer requests one at a time, calling the transfer function above.

As the number of accounts and the frequency of transferring money grew considerably, the bank decided to migrate its software to a multiprocessor system, storing the accounts in shared memory and allowing multiple threads to process transfer requests concurrently. It is your task to make the software thread-safe.

(a) Show with a concrete example scenario that, without proper synchronization, the above code contains a race condition that can lead to wrong results. (1p)

(b) Identify the critical section(s) and write a simple, thread-safe version of the transfer function above, using an appropriate mechanism. Explain all operations carefully. (2p)

(c) For high transfer request rates, the simple synchronized solution of the previous question may not scale, even if there are enough processors available. Why? (0.5p)

Suggest a more advanced protection scheme that allows many transfers to proceed simultaneously. Show the (pseudo)code and explain all operations. (1.5p)

May your new solution lead to deadlocks? If yes, state under what condition(s) and suggest a suitable mechanism for deadlock prevention. If no, explain why your solution is deadlock-free. (1p)

(d) What would be a more appropriate abstraction (programming construct) for this kind of operation, and why? (Give the technical term and a short explanation). (1p)

3. CPU Scheduling

Given a single-CPU system and the following set of processes with arrival times (in milliseconds), expected maximum execution time (ms), and priority (1 is highest, 5 is lowest priority).

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival time</th>
<th>Execution time</th>
<th>Priority (as applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>P₂</td>
<td>3</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>P₃</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>P₄</td>
<td>6</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>P₅</td>
<td>8</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

For each of the following scheduling algorithms, create a Gantt chart (time bar diagram, starting at \( t = 0 \)) that shows when the processes will execute on the CPU. Where applicable, the time quantum will be 3 ms. Assume that a task will be eligible for scheduling...
immediately on arrival. If you need to make further assumptions, state them carefully and explain your solution. (5p)

(i) FIFO;
(ii) Round-robin;
(iii) Shortest Job First without preemption;
(iv) Priority Scheduling without preemption.
(v) Priority Scheduling with preemption.

4. (3 p.) Deadlocks

(a) There are four conditions that must hold for a deadlock to become possible. Name and describe them briefly. (2p)
(b) Why does the Banker’s algorithm for deadlock avoidance fail in systems that support hot swappable devices (such as USB sticks)? (1p)

5. (9 p.) Memory management

(a) Several 32-bit processors use three-level paging to address the large page table problem, where the page table itself is stored in main memory. (If you don’t recall three-level paging, you may answer, with reduced points, this and the following question for one- or two-level paging.)

Explain three-level paging for a 32-bit virtual address space and a memory page size of 1024 bytes. Show the structure of virtual (logical) addresses and explain (with a well commented drawing) how to compute physical memory addresses. (2.5p)

(b) As in one-level paging, a TLB can be used with three-level paging to speed up address calculations for frequently accessed pages. Given the average time $t_m = 60\text{ns}$ for a physical memory access, time $t_{TLB} = 1\text{ns}$ for a TLB access, and an assumed TLB hit rate of $0.99 = 99\%$, determine the effective memory access time in three-level paged memory. Explain your calculation carefully. (1.5p)

(c) Given a virtual memory system with 5 page frames, how many page faults occur with the Least-Recently Used replacement strategy when pages are accessed in the following order:
1, 2, 3, 4, 5, 1, 2, 3, 6, 4, 5, 1, 7, 3, 2, 4.
(Justify your answer. Just guessing the right number is not acceptable.) (1.5p)

(d) For the same access sequence, what would be the theoretical minimum number of page faults, assuming the clairvoyant optimal replacement strategy? (Justify your answer. Just guessing the right number is not acceptable.) (1.5p)

(e) Denning suggested to estimate a process’s working set size by using a window of fixed size $w$ on the sequence of addresses accessed. If this estimation is used to guide the allocation of page frames to processes, what trade-offs are inherent in choosing that window size $w$? (i.e., what are the possible advantages and disadvantages of choosing a large or small $w$, respectively?) (2p)
6. (3.5 p.) **File systems**

   (a) Does a soft link still work if the file it links to is moved to a different location on disk? Why or why not? (1p)

   (b) Describe the file allocation method *indexed allocation* and discuss its strengths and weaknesses. (1.5p)

   (c) Describe one case where the file system is *not* an appropriate abstraction for secondary storage, and explain why. (1p)

7. (3 p.) **OS Structures**

   (a) Define the term *layered operating system* carefully. (1p)

   (b) What is the main reason for structuring operating systems in a layered fashion? (0.5p)

   (c) What are the two main disadvantages of layering in operating systems? (1p)

   (d) In how far is Unix a layered operating system? (0.5p)

8. (3.5 p.) **Protection and Security**

   (a) What is a *Trojan Horse attack*? Explain the term in general and give one example scenario. (2p)

   (b) How can using *virtual machines* increase the security of a system? (1.5p)

Good luck!