Solutions for CS 354 Final, Spring 2013

P1(a) 10 pts

A delta list for future timer events stores time values of events relative
to the preceding element in the list. For example, events A, B, C with time values 5msec, 27msec, 40msec would be stored relative time values 5msec, 22msec, 13msec.
5 pts

The main benefit is constant update overhead: when a clock interrupt handler (say invoked every 1msec) updates time stamps of all future events,
only the time value of the first element needs to be updated since all other elements are relative.
5 pts

P1(b) 10 pts

Assuming two processes X and Y, and binary semaphores s and t: X has s,
X wants t, Y has t, Y wants s. The resultant 4-node resource graph has a cycle.
4 pts

Deadlock detection is tantamount to cycle detection in a directed graph,
hence linear overhead.
3 pts

Modern kernel adopt the Ostrich approach: ignore the problem and let users/apps in user space handle it. This approach is viable since the impact of a deadlock is limited to the deadlocked processes.
3 pts

P1(c) 10 pts

In external fragmentation, there is, in total, sufficient memory to satisfy a dynamic memory allocation request, however, the free memory is fragmented so that contiguous memory allocation is not possible.
4 pts

Indirection that maps virtual (or logical) addresses to physical addresses
achieves defragmentation, but the translation must be done with hardware
support since software indirection would incur too much overhead.
3 pts

An L1 cache stores frequently accessed instructions/data in small but fast memory that is addressed by a CPU using virtual addresses. Since virtual addresses are used, assuming a cache hit, there is no need to perform additional hardware assisted address translation with the help of TLB.
3 pts

P1(d) 10 pts

Accessing a byte in a file in Xinu FS requires linear overhead to search
through the linked list index blocks. Doing so in a UNIX FS requires logarithmic overhead due to the tree structure of the inode pointers comprised of indirect, doubly-indirect, and triply-indirect pointers.
5 pts

By the 90/10 rule, most files in real-world file systems are small. UNIX
FS deals with this property by allocating 10 direct pointers that reference file data up to 5KB assuming 512B block size.
5 pts

P2(a) 36 pts

A page fault arises when a page addressed by a CPU is not found in RAM, hence must be read in from swap space on secondary memory (e.g., disk).
6 pts

The rationale for responding to page faults via interrupt handlers as part of a kernel is: fetching the missing page from swap space incurs significant slow-down (e.g., disk is about 1000 times slower than RAM).
Hence context switching out the process that page faulted so that a CPU may be utilized by other processes while disk I/O is on-going is necessary to not idle the CPU. Since context switching entails scheduler intervention, this task is well-suited for kernel execution.
6 pts

P2(b) 12 pts

One form of overhead comes from saving the context of a process to be switched out, and restoring the previously checkpointed context of a process to be switched in.
6 pts

Another form of overhead comes from having to flush the contents of caches (L1, TLB, L2) to deal with aliasing, which then incurs significant
overhead due to initial cache misses when populating the caches with

instructions/data/TLB entries referenced by the context switched in process.
6 pts

P2(c) 12 pts

The two design choices for achieving isolation/protection were: execute
the user supplied callback function 1) after switching back to user mode
from kernel mode, and 2) switching to the context of the process that registered the callback function.
6 pts

The lower half of a kernel which handles interrupt related processing, is either executed while borrowing the context of the current process or as a separate kernel thread. In both cases, the lower half which acts as a writer (or reader) is just another process. Hence, the coordination required to achieve device I/O is again between a pair of processes as in IPC.
6 pts

P3(a) 12 pts

The upper half handles system calls and the lower half responds to interrupts.
3 pts

The top half of the lower half handles the fast part of interrupt handling
which pertains to interacting with a hardware device (controller) such as copying data from/to device buffer to kernel memory (RAM). This component needs to be carried out with other interrupts disabled which is feasible to do since the top half involves only a few instructions.
3 pts

The bottom half handles the slow part of interrupt handling such as processing Internet packets that involve many instructions. This slow component of interrupt handling runs with interrupts enabled.
3 pts

The two design options for bottom half are: 1) a kernel function call that borrows the context of the current process, or 2) a kernel thread that executes the kernel function call as a separate process.
The first option incurs lower overhead but bottom half cannot make blocking calls. The opposite holds for the second option.
3 pts

P3(b) 12 pts

A 2-level page table consists of a first-level table with 2^10 entries.
Each of the entries points to a second-level table which has 2^10 entries,
or it points to NULL. Since many app processes use far less memory (i.e., page) than 2^32 (i.e., 2^20 pages), most of the first-level table entries will be NULL. This saves significant space.
6 pts

With only 5 pages needed, only 5 entries in the first-level table will point to second-level page tables. Since all tables have 2^10 entries, a total of 6 tables (the first-level table and 5 second-level tables) are used. 6 x 2^10 is a much smaller size than 2^20 needed in a 1-level page table..
6 pts

Bonus

Two exceptions: pageout kernel process and bottom half of lower half implemented as kernel thread.
6 pts

The main rationale is that context borrowing incurs less overhead than context switching between processes/threads. Also, if kernel chores are
implemented as threads, IPC is needed for coordination which incurs additional overhead. Since speed/efficiency is a major concern in kernel
design, context borrowing using function libraries has dominated.