一、Computer Organization and Design

1. **[Acronyms]** Example: DMA →Direct Memory Access. (a) VHDL (b) SoC (c) TLB (d) RAID (e) NUMA. (Within the context of computer architecture.) (10%)
2. **[Number System, IEEE 754]** Determine the base of the number system for the following operation to be correct: 23+44+13+32 = 223. (5%) (b) Find the smallest normalized positive floating-point number in the IEEE-754 single-precision representation. (5%)
3. **[Cache]** Consider data transfers between two levels of hierarchical memory. If we logically organize all data stored in the lower level into blocks and store the most-frequently-used blocks in the upper level. Assume that the hit rate is H, the upper level latency is T_u, and the lower level latency is T_l, and the miss penalty is T_m.
   (a) What is the average memory latency? (5%)
   (b) What is the speedup by using the hierarchical memory system? (5%)
4. **[Pipelining]** (a) Assuming no hazards, a pipelined processor with *s* stages (each stage takes 1 clock cycle) can execute *n* instructions in _____ clock cycles. (2%) (b) Use the result in (a) to show that the ideal pipeline speedup equals the number of stages. (3%)
5. **[Approximation Circuits]**
   In a traditional adder design, the calculation must consider all input bits to obtain the final carry out. However, in real programs, inputs to the adder are not completely random and the *effective carry chain is much shorter* for most cases. Instead of designing for the worst-case scenario, it is possible to build a faster adder with a much shorter carry chain to *approximate* the result. Suppose we consider only the previous k inputs (look-ahead k-bits) instead of all previous input bits to estimate the *i*-th carry bit c_i , i.e.,

   $c_i=f(a_{i-1},b_{i-1}, a_{i-2},b_{i-2},\ldots, a_{i-k},b_{i-k})$ where $0<k<i+1$ and $a_j,b_j=0$ if $j<0$.

   (a) With random inputs, show that c_i will generate correct a result with a probability of

   $$1-\frac{1}{2^{k+2}}.$$(5%)

   (b) What is the probability of having a correct 'carry' result considering only *k* previous inputs for an N-bit addition? (5%)
   (c) Design a logic circuit to detect when the approximate adder will generate an incorrect carry result for the *i*-th carry bit. (5%)

二、Operating Systems

1. Suppose that two processes, *Pa* and *Pb*, are running in a *uni-processor* system. *Pa* has *three* threads. *Pb* has *two* threads. All threads in both processes are CPU-bound and they never block for I/O. The operating system uses simple *round-robin scheduling*.

   (a) (2%) Suppose that all of the threads are *user-level* threads, and that user-level threads are implemented using a single kernel thread per process. What percentage of the processor's time will be spent running *Pa*'s threads?

   (b) (2%) Suppose instead that all of the threads are *kernel* threads. What percentage of the processor's time will be spent running *Pa*'s threads?

2. Answer the following questions about deadlock.

(a) (3%) A system has three processes sharing seven units of resource R. Each process may request up to three units of R. Is there a danger of deadlock?

(b) Consider a system in which the total available memory is 48K and in which memory once allocated to a process *cannot* be preempted from that process. Three processes A, B, and C have declared in advance that the *maximum* amount of memory that they will require is 25K, 15K, and 41K words respectively. When the three processes are all in execution and using 3K, 9K, and 24K words of memory respectively, consider the following additional requests *individually*:

i) (1%) A requests 9K words more. Can this request be granted with a guarantee that deadlock will not occur? (yes or no)

ii) (1%) B requests 6K words more. Can this request be granted with a guarantee that deadlock will not occur? (yes or no)

iii) (1%) C requests 7K words more. Can this request be granted with a guarantee that deadlock will not occur? (yes or no)

(c) (2%) Explain why an OS designer needs to worry about deadlock involving physical memory but not the CPU.

(d) (2%) Does a deadlock prevention algorithm necessarily prevent starvation as well?

3. Given the following variables and their initial values as follows:
   x： integer, initial value is 0
   y： integer, initial value is 23
   m： *general semaphore*, initial value is 1
   ml： *general semaphore*, initial value is 0

For each of the following program, list values *possible* for *x* when the program terminates. Note that "**cobegin** P1; || P2; **coend**" means P1 and P2 are executed concurrently, and ":=" is the assignment operator.

(a) (2%) **cobegin** x := x +1; || x := y+1; **coend**
(b) (2%) **cobegin** P(m); x := x +1; V(m); || P(m); x := y+1; V(m); **coend**
(c) (2%) **cobegin** P(ml); x := x +1; V(m); || P(m); x := y+1; V(ml); **coend**
(d) (2%) **cobegin** V(ml); x := x +1; P(m); || V(ml); x := y+1; P(ml); **coend**

4. (a) (3%) Suppose that you wish to design a virtual memory system with the following characteristics:
   i) The size of a page table entry is 4 bytes.
   ii) Each page table must fit into a single physical frame.
   iii) The system must be able to support virtual address spaces as large as 2^{38} bytes (256 GB).

   Suppose that you decide to use a *multi-level paging scheme*. What are the respective *minimum* page sizes that your system must have if you determine to use *two levels* or *three levels* of page tables? Your answer includes two parts: one for two-level and one for three-level.

(b) (3%) For a certain page trace starting with no page in the memory, a demand-paged memory system operated under the least-recently used *(LRU) replacement policy* results in 9 and 11 *page faults* when the primary memory is of 6 and 4 pages, respectively. When the same page trace is operated under the *optimal policy*, what are the numbers of page faults may be for the same primary memory of 6 and 4 pages?

(c) (3%) When we switch from *statically* linked libraries to *dynamically* linked libraries, would there be any effects on the working set size of a process and on the working set size of all the running processes?

5. One operating system performs process scheduling by assigning *base priorities* (a high of -20 to a low of +20 with a median of 0), and making *priority adjustments*. Priority adjustments are computed in response to changing system conditions. These are added to base priorities to compute *current priorities*; the process with the *highest* current priority is dispatched first.

The priority adjustment is strongly biased in favor of process that have recently used relatively little CPU time. The scheduling algorithm "forgets" CPU usage quickly to give the benefit of the doubt to processes with changing behavior. The algorithm "forgets" 90% of recent CPU usage in $5 * n$ seconds; n is the average number of runnable processes in the last minute.

Based on the above description of the process scheduling algorithm, answer each of the following questions briefly.

(a) (2%) Are CPU-bound processes favored more when the system is heavily loaded or when it is lightly loaded? (heavily or lightly)
(b) (2%) Will an I/O-bound process receive favored or unfavored response immediately after an I/O completion? (favored or unfavored)
(c) (2%) Can CPU-bound processes suffer indefinite postponement? (yes or no)
(d) (2%) When a process's behavior changes from CPU-bound to I/O-bound, it will immediately receive favored response. True or false?
(e) (2%) When a process's behavior changes from I/O-bound to CPU-bound, it will immediately receive favored response. True or false?

6. (a) (3%) Among the first-come-first-serve (FCFS), shortest seek time next (SSTF), LOOK, SCAN, C-LOOK, C-SCAN, and N-Step SCAN disk-head scheduling policies, which are subject to *starvation* at high loads (i.e., very large number of requests arrive in a short time)?
(b) (2%) Response times are more predictable in preemptive systems than in non-preemptive systems. True or false.
(c) (2%) Describe a program that works fine with round-robin scheduling, but makes other program starving under first-come-first-serve (FCFS) scheduling.
(d) (2%) The LRU replacement scheme is *inappropriate* for managing disk cache of sequential files? True or false.