I. Kernel, Syscall and Scheduling [25 points]

(a) What data does the kernel stack of process save? [3 points]

(b) Explain the difference between processes and kernel-level threads. What actions does context switching between two different processes require but NOT needed for context switching between two threads of a same process. [4 points]

(c) Explain what happens when a parent process terminates before its children. [4 points]

(d) Does an exception (trap) always cause a user process to be terminated during its execution? If so, why? If not, provide an example. [4 points]

(e) Consider we have four jobs with (arrival-time, runtime) pair as (0, 1000), (100, 500), (400, 200), (0, 600). Compute the average response time and turnaround time for SJF, STCF, and RR with 100 time unit as quantum (time slice). Here response time means the time between when a job arrives and when it gets first scheduled, while turnaround time means the time between when job arrival time and when it finishes. [6 points]

(f) Explain why the original multi-level feedback queue scheduler may cause some processes to suffer from starvation. Provide a solution to this problem. [4 points]

II. Memory Virtualization [25 points]

(a) Name two advantages and one disadvantage of paging against segmentation. [5 points]

(b) Suppose we have a multi-level page table with 8KB page size and 8-byte PTE size. How many layers should this page table have to support a 56-bit virtual address space. How many page table entries does a node at each level have? [5 points]

(c) Consider a virtual memory system with 4 physical pages. A program, whose virtual address space contains 8 pages called A-H, accesses memory pages in the following order: CAHCBGDFHDAFCGDEH. How many page faults are incurred if the LRU page replacement policy is used? [5 points]

(d) Describe one pathological case where the LRU page replacement policy is strictly no better than choosing a random page to evict. [5 points]

(e) In the following code, why might LOOP 1 run much slower than LOOP 2? What is the cause of this slowness and where is the time being spent? [5 points]

```
int main(int argc, char* argv[]) {
    int* a = (int*) malloc(10000 * sizeof(int));

    // LOOP1
    for (size_t i = 0; i < 10000; ++i) a[i] = i;

    // LOOP2
    int sum = 0;
    for (size_t i = 0; i < 10000; ++i) sum += a[i];

    free(a);

    return sum;
}
```

III. Concurrency [25 points]

(a) What is the difference between spin locks and blocking locks? [5 points]

(b) Explain why the following code does not always print out 2000000. (Illustrate at least one case with scheduling timeline) [10 points]

```
int i = 0;

void* run(void* _) {
    for (int j = 0; j < 1000000; j++) i++;
}

void main() {
    pthread_t t1, t2;
    pthread_create(&t1, NULL, run, NULL);
    pthread_create(&t2, NULL, run, NULL);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    printf("%d\n", i);
}
```

(c) Consider a producer/consumer problem with a bounded queue where multiple producers and consumers can operate concurrently. Below is an attempt to solve this problem using condition variables. Identify the bug in the code below and try to fix it. [10 points]

```
void produce(int input) {
    mutex_lock(&m);
    while (queue->is_full()) cond_wait(&cond, &m);
    queue->push(input);
    cond_notify_one(&cond);
    mutex_unlock(&m);
}

int consume() {
    mutex_lock(&m);
    while (queue->is_empty()) cond_wait(&cond, &m);
    int res = queue->pop();
    cond_notify_one(&cond);
    mutex_unlock(&m);
    return res;
}
```

IV. File System [25 points]

For a file system with 4KB blocks and 8 byte block numbers, suppose each inode contains 12 direct block numbers, 1 single indirect block number, and 1 double indirect block number.

(a) How many block numbers can one single indirect block contain? [5 points]

(b) Consider a file in this file system containing 32MB of data. How many total indirect blocks (single and double) will be part of the file's metadata? [5 points]

(c) What is the maximum file size to the nearest megabyte (MB) supported by this file system? [5 points]

(d) Explain how RAID-5 can tolerate the failure of a whole disk and how it can recover the data on a newly installed disk in place of the faulty one. [5 points]

(e) Explain the purpose of journaling in a file system. [5 points]

V. Performance Analysis [20 points]

(a) An image processing task takes 40% of the overall processing time of an application. This task is now accelerated in a GPU. What is the maximum speedup that can be expected for the entire application through this acceleration? What are constraints that will limit approaching this maximum speedup (list at least 3)? [10 points]

(b) Most processors are designed to have adjustable voltage, so a 15% reduction in voltage may result in a 15% reduction in frequency. What would be the impacts on dynamic energy and on dynamic power? [5 points]

(c) What are the implications for power consumption when moving from a write-through policy to a write-back policy? [5 points]

VI. Cache/Memory System [20 points]

(a) Suppose we have a processor with a base CPI of 1.0, assuming all references hit in the primary cache, and a clock rate of 5 GHz. Assume a main memory access time of 100ns, including all the miss handling. Suppose the hit rate per instruction at the primary cache is 98%. How much faster will the processor be if we add a secondary cache that has a 6ns access time for either a hit or a miss and is large enough to reduce the miss rate to main memory to 0.5% per instruction? [5 points]

(b) Assume a cache containing 4K total blocks, a four-word block size, and a 32-bit address, find the total number of sets and the total number of tag bits for caches that are direct-mapped, two-way and four-way associative and fully-associative. [5 points]

(c) Show an example of a sequence of cache accesses that will increase cache misses when moving from direct mapped to fully-associative caches (when keeping cache size constant and assuming LRU). [5 points]

(d) We have a virtual memory system that has a physically-indexed, physically tagged cache. A memory reference can encounter three different types of misses: a TLB miss, a page fault and a cache miss. Consider all possible combination of these three events with one or more occurring. For each possibility, state whether this event can actually occur and under what circumstances. [5 points]

VII. Cache/Memory System [20 points]

A "second chance cache" (SCC) is a hardware cache designed to decrease conflict misses and improve hit latency for direct-mapped L1 caches. It is employed at the refill path of an L1 data cache, such that any cache line (block) which gets evicted from the cache is cached in the SCC. In the case of a miss in L1, the SCC cache is looked up (in some implementations, the SCC and L1 cache are looked up in parallel but we omit that possibility in this question). If the resulting access is a hit in the SCC, the line is transferred to L1.

(a) Explain the rationale behind the SCC proposal. Why can one expect it to be useful? Give a sample data access sequence illustrating the performance difference between an L1 cache and an L1+SCC combination. [4 points]

(b) One designer suggests to employ a direct-mapped L1 cache of (A+B) KB, instead of employing a direct-mapped L1 cache of A KB and an SCC of B KB. Are you in favor of this suggestion? Why? Or why not? [4 points]

(c) Give a sample data access sequence that is not good for the SCC. Explain clearly why it is not good. [4 points]

(d) Do you think an SCC should be direct-mapped or set-associative? Why? [4 points]

(e) Do you think an SCC can be used along with an L2 or L3 cache as well? Explain. [4 points]

VIII. Pipelining [20 points]

(a) [4 points]

CYCLE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
QUX: lw $2, 0($4)	F	D	E	M	W											
lw $3, 0($5)		F	D	E	M	W										
addu $2, $2, $3			F	d	d	d	D	E	M	W						
sw $2, 0($4)				f	f	f	F	D	E	M	W					
lw $4, 4($4)								F	D	E	M	W				
addi $5, $5, 4									F	D	E	M	W			
bne $4, $0, QUX										F	d	d	D	E	M	W

Consider the above ***correct*** schedule for some FDEMW 5-stage scalar pipeline with **unspecified** support for <u>forwarding</u>, <u>branch resolution</u>, and <u>hazard detection</u>. Lowercase letters indicate instruction stalls, and excepting the sw and bne instructions, the leftmost register number is the destination specifier (sw and bne do not define registers).

Which of the following options best matches the observed behavior of the above schedule?

1. Hazard detection in D, Branch Resolution in D, Only W->M forwarding supported
2. Hazard detection in E, Branch Resolution in E, Forwarding from W to E and M
3. Hazard detection in E, Branch resolution in D, All possible forwarding is present
4. Hazard detection in D, Branch resolution in E, Forwarding from W and M to M and E
5. None of the above

(b) [8 points]

Consider the below sequence of instructions executed on a 6-stage F|D|E|M1|M2|W pipeline with branch resolution and hazard detection in D with all possible forwarding paths supported. Assume that the branch is predicted not-taken, but is actually taken.

CYCLE	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**
FOO: lw $3, 4($5)	**F**																		
lw $2, 4($4)																			
addu $3, $2, $3																			
lw $4, 8($4)																			
sw $3, 0($2)																			
addi $5, $5, 4																			
bne $4, $0, FOO																			
add $2, $5, $0																			

A) Complete the schedule through the writeback of the post-branch LW instruction. Use the first row to depict the second fetched instance of the LW at FOO, as well as the first.

Either copy-paste the above table directly into your answer field, or, for slightly better formatting, use the table insertion button to initiate a table and then copy paste the table data into the existing table in the answer field. Use capital letters for completed stages and lower case letters for stalled stages.

B) List **all** instances of forwarding that occur in your schedule in the following format (cycle#, op#:stage -> op#:stage) e.g. (2, 1:D->2:F) would [nonsensically] indicate that the LW at FOO forwarded from its decode stage to the fetch stage of the addi during cycle 2.
Include W->D, but do not include W->d, in your forwarding list. (Note that forwarding can span iterations)

(c) [8 points]

Consider moving from an FDEMW pipeline to an F1|F2|H|D|E1|E2|M1|M2|W 9-stage pipeline. Assume that the FDEMW pipeline has hazard detection and branch resolution in D and all possible forwarding paths while the second (9-stage) has hazard detection (and decode) in H and branch resolution (and register read) in D and all possible forwarding paths. Assume an "always not taken" branch predictor.

If your program consists entirely of ALU, Load, Store and Branch instructions

A1) Give a pair of instructions that causes the longest possible data hazard stall in the 9-stage pipeline.

A2) How many cycles is the above stall?

B) If a representative code sequence is:

```
CONT:
lw      $t1, 0($a0)
add     $s0, $s0, $t1
lw      $a0, 4($s0)
beq     $t1, $0, SKIP
sw      $s0, 0($a1)
SKIP:   bne $s0, $0, CONT
```

Below is a schedule for the loop when the branch to SKIP is A) not taken and B) correctly predicted:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
CONT: lw $t1, 0($a0)	F1	F2	H	D	E1	E2	M1	M2	W									
add $s0, $s0, $t1		F1	F2	h	h	h	H	D	E1	E2	M1	M2	W					
lw $a0, 4($s0)			F1	f2	f2	f2	F2	h	H	D	E1	E2	M1	M2	W			
beq $t1, $0, SKIP				f1	f1	f1	F1	f2	F2	H	D	E1	E2	M1	M2	W		
sw $s0, 0($a1)								f1	F1	F2	H	D	E1	E2	M1	M2	W	
SKIP:bne $s0, $0, CONT										F1	F2	H	D	E1	E2	M1	M2	W

Below is a schedule where the branch to SKIP is A) taken and B) correctly predicted:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
CONT: lw $t1, 0($a0)	F1	F2	H	D	E1	E2	M1	M2	W									
add $s0, $s0, $t1		F1	F2	h	h	h	H	D	E1	E2	M1	M2	W					
lw $a0, 4($s0)			F1	f2	f2	f2	F2	h	H	D	E1	E2	M1	M2	W			
beq $t1, $0, SKIP				f1	f1	f1	F1	f2	F2	H	D	E1	E2	M1	M2	W		
SKIP:bne $s0, $0, CONT								f1	F1	F2	H	D	E1	E2	M1	M2	W	

If the branch to SKIP is only taken on the first iteration and the branch to CONT is taken 999 times in a row and not taken on the 1000th time, what is the average CPI for the first 300 loop iterations for the 9-stage pipeline?

C) What would the CPI be for the 9-stage pipeline if we added a 1-bit bimodal branch predictor, initially in state N=0 (for the branch to SKIP) and T=1 (for the branch to CONT), both branches have been encountered before and an entry is present for each in the BTB, and this loop executes 1000 times (again, as in B, 999 taken, then 1 not taken to CONT, 1 Taken and then 999 not taken to SKIP)?

IX. Parallelism [20 points]

Two important techniques have emerged recently for achieving high performance in processors. One is speculative execution, where instructions – or sequences of instructions – are executed before all the information needed to commit the instruction has been nailed down. The other is simultaneous multithreading (SMT), where the processor can issue instructions from multiple threads (or processes), potentially in the same cycle.

(a) Using a sequence of instructions in a MIPS-like assembly, explain how each technique works. [5 points]

(b) What is the fundamental problem with aggressive speculation? [5 points]

(c) What are the advantages and drawbacks of each technique (speculation and SMT) with respect to the other? [5 points]

(d) Do these approaches capture different aspects of/opportunities for parallelism, or are they different ways at getting at the same results? If both were implemented in a single system, would you expect their projected improvement to be *additive*? Why or why not? [5 points]