國立陽明交通大學
115 學年度碩士班考試入學招生試題

科目：計算機系統(含作業系統及計算機組織)(8103)

系所班別：資訊聯招

考試日期：115 年 2 月 4 日 第 3 節

【不可使用計算機】

＊作答前請先核對試題、答案卷(試卷)與准考證之
所組別與考科是否相符！！

國立陽明交通大學 115 學年度碩士班考試入學招生試題

科目：計算機系統(含作業系統及計算機組織)(8103)　　考試日期：115 年 2 月 4 日　第 3 節

系所班別：資訊聯招　　

【不可使用計算機】＊作答前請先核對試題、答案卷(試卷)與准考證之所組別與考科是否相符！！

請用答案卡作答

一、複選題(80%)，共二十題，每題全對得 4 分。答對一個選項得 1 分(答對兩個選項得 2 分，以此類推)，答錯一個選項倒扣 2 分(答錯兩個選項倒扣 4 分，最多扣至該題 0 分為止)，整題未做答不給分。(舉例：每題有 a、b、c、d 四個選項，某題答案為 a、b，而作答時選 a、b、c，則 3 個選項正確 1 個錯誤，該題得 3-2=1 分)。

1. Regarding dynamic loading, which of the following statements are TRUE?
   (a) In dynamic loading, a routine is not loaded into the main memory until it is called; all routines are kept on disk in a relocatable load format until required.
   (b) Dynamic loading typically offers better memory-space utilization than static loading because routines that are never called (such as large, infrequently used error-handling routines) are never loaded into memory.
   (c) Dynamic loading strictly requires special support from the complex kernel process scheduling to function; it is impossible for user programs to implement this method using standard libraries alone.
   (d) Compared to dynamic loading, static loading generally results in faster program startup times for large applications because the entire program is loaded into memory at once, avoiding runtime overheads during execution.
2. Which of the following statements best distinguishes software traps from hardware interrupts in terms of their origin, execution context, and privilege transition?
   (a) Handling Timing: Hardware interrupts are typically recognized and serviced only at instruction boundaries (i.e., after the current instruction retires), whereas software traps (specifically faults like Page Faults) are recognized and serviced immediately during the execution of the instruction.
   (b) Masking Capability: The operating system can use the Interrupt Flag (e.g., IF in x86) to mask (ignore) "Divide-by-Zero" and "Page Fault" software traps to ensure atomicity within critical kernel sections.
   (c) Synchronicity: Hardware interrupts are asynchronous events triggered by external devices (independent of the CPU clock), whereas software traps are synchronous events tied directly to the execution of a specific instruction.
   (d) Event Nature: Software traps strictly indicate erroneous program behavior (e.g., illegal memory access), whereas hardware interrupts indicate normal system events; therefore, legitimate operating system service requests (System Calls) are never implemented using the trap mechanism.
3. Regarding the necessity, performance, and structure of multi-level page tables compared to single-level page tables, which of the following statements are TRUE?
   (a) A primary necessity for multi-level design in modern architectures (e.g., 64-bit) is that a linear single-level page table would require an impractically large, but not contiguous block of kernel memory, which is often impossible to allocate.
   (b) In terms of memory consumption, a multi-level page table always consumes less physical memory than a single-level page table, regardless of whether the process's virtual address space is sparse or fully populated (dense).
   (c) Regarding address-space sparsity, multi-level page tables save memory by allowing the operating system to avoid allocating inner-level page tables for invalid (unused) regions of the virtual address space.
   (d) In terms of memory access time (assuming a TLB miss), the multi-level page table structure introduces a performance overhead compared to a single-level table, as the MMU must perform multiple memory references to traverse the hierarchy (e.g.,

directory-> table -> frame).

4. Regarding the behavior of fork() and exec() system calls in a multithreaded process (specifically under POSIX standards), which of the following statements are TRUE?
   (a) When fork() is invoked by a thread in a multithreaded process, the resulting child process contains duplicates of all threads that were running in the parent process at the time of the call.
   (b) If a thread (other than the one calling fork()) holds a lock (e.g., a mutex) at the moment fork() is called, that lock will exist in the child process in a locked state, but the thread holding it will not exist in the child to unlock it, leading to potential deadlocks if the child tries to acquire this lock.
   (c) It is generally unsafe to call complex library functions (like printf or malloc) in the child process between fork() and exec() because these functions may rely on internal locks that were left in an inconsistent state during the fork.
   (d) The exec() system call allows the child process to retain the memory space and thread context of the parent process while simply resetting the Program Counter (PC) to the beginning of the main() function.
5. Regarding virtual memory management techniques, specifically Thrashing, Working-Set Model, Page Replacement Algorithms, and Copy-on-Write (COW), which of the following statements are TRUE?
   (a) A classic symptom of thrashing is low CPU utilization. This occurs because processes spend the majority of their time waiting for the paging device (I/O), which often misleads the standard OS scheduler into increasing the degree of multiprogramming, thereby worsening the thrashing.
   (b) Belady's anomaly refers to the phenomenon where increasing the number of available page frames results in an increase in the number of page faults. This anomaly can occur in the FIFO algorithm but is mathematically proven to be impossible in LRU (Least Recently Used), which belongs to the class of stack algorithms.
   (c) When fork() is executed with Copy-on-Write enabled, the parent and child processes share the same physical pages. To detect when a copy is needed, these shared pages are initially marked as read-write in the page table so that the hardware allows the write to proceed immediately without OS intervention.
   (d) The Working-Set Model prevents thrashing by estimating the current locality of each process. If the sum of the working-set sizes of all active processes exceeds the total number of available frames, the operating system selects a process to suspend (swap out) to release frames.
6. Which of following statement(s) about CPU scheduling are correct?
   (a) The turnaround time of the non-preemptive Shortest-Job-First (SJF) scheduling remains even when the arrival rate of jobs varies.
   (b) The turnaround time of the SJF is always shorter than that of the First-In-First-Out (FIFO) scheduling when all jobs arrive in the system simultaneously.
   (c) We define the response time as the time from when the job arrives in a system to the first time it is scheduled for execution. The average response time of the round robin (RR) scheduling is shorter than that of preemptive SJF scheduling.
   (d) We can periodically increase the priority of all jobs in the system to mitigate the starvation problem exhibited in Multi-Level Feedback Queue (MLFQ) scheduling.
7. The atomicity violation bug occurs when multiple operations intend to run an indivisible (atomic) unit, but are interrupted by another thread in concurrent programs. The

atomicity violation bugs result in the desired serializability among multiple memory accesses being violated, leading to lost updates. Please choose the correct atomicity violation bug code pattern(s) below.

(a)
```
static int i = 0;
static object a, b = default;
var tid1 = Task.Run( () => {
      lock(a) {i++;}
});
var tid2 = Task.Run( () => {
      lock(b) {i++;}
});
Task WhenAll (tid1, tid2); print(i);
```

(b)
```
static pthread_cond_var WorkDone =
PTHREAD_COND_INIT;
void main () {
      pthread_id tid = null;
      pthread_create(&tid, NULL, &fun,
NULL);
      pthread_wait(&WorkDone);
}
void fun() {pthread_singal(&WorkDone);}
```

(c)
```
static int i = 0;
var tid1 = Task.Run( () => {
      if (atomic_load(&i) == 0) {
            atomic_add(&i, 1);
      }
});
var tid2 = Task.Run( () => {
      if (atomic_load(&i) == 0) {
            atomic_add(&i, 1);
      }
});
Task WhenAll (tid1, tid2); print(i);
```

(d)
```
static long int i = 101;
var tid1 = Task.Run( () => {
      i++;
});
var tid2 = Task.Run( () => {
      print(i);
});
Task WhenAll (tid1, tid2);
```

8. The spin lock ensures the correctness of concurrent programs on multiple processors. Please choose the correct spin-lock implementation(s) and usage(s) including the Test-And-Set, Compare-And-Swap, Load-Linked and Store-Conditional, and Fetch-And-Add instructions. (Note that the LoadLinked() and StoreConditional() , respectively, perform Load-Linked and Store-Conditional operations).

(a)
```
int TestAndSet (int *old_ptr, int new) {
      int old = *old_ptr;
      *old_ptr = new;
      return old;
}|
```

(b)
```
int CompareAndSwap (int *ptr, int
expected, int new) {
      int original = expected;
      if (*ptr == expected)
            original = new;
      return original;
}|
```

(c)

```
void lock (lock_t *lock) {
        while (1) {
                if (LoadLinked (&lock->flag)
== 1) return;
                while (StoreConditional
(&lock->flag, 1) == 1);

        }
}
```

(d)

```
int FetchAndAdd (int*ptr) {
        int old = *ptr;
        *ptr = old + 1;
        return *ptr;
}
```

9. The Linux **cat** command opens the file for reading. On Linux, the **strace** tool can dump the trace of every system call made by the **cat** Linux command shown below. Please choose the correct description(s) for system calls used by the **cat** Linux command.

```
prompt> strace cat hello
…
open (“hello”, O_RDONLY| O_LARGEFILE)
read (3, “helloworld\n”, 4096)
write (X, “helloworld\n”, 11)
helloworld
read(3, “”, 4096)
close(3)
…
```

(a) The return value of the first open system call is 3.
(b) The return value of the first read system call is 11.
(c) The first argument (X) of the write system call represents the file descriptor and the value of X is 1.
(d) The return value of the second read system call is 0.

10. The hard drive (disk) has become the main form of persistent data storage in computer systems. In the disk, the motor spins the platters around at a constant rate. The rate of hard drive rotation is measured in Rotations Per Minute (RPM). Please choose the correct description(s) about the performance of disks below.
(a) Each rotation takes 6 milliseconds (ms) when the RPM of this disk is 10,000.
(b) When the transfer rate of a disk is 100MB/second, transferring a 1024 KB block takes 10 milliseconds (ms) to complete.
(c) The SCAN disk scheduler is invented to shorten the seek time of the disk.
(d) The SCAN disk scheduler can resolve the starvation problem is shown in the Shortest Seek Time First (SSTF) SSTF disk scheduler when a steady stream of requests is always directed to the inner track of the disk, where the head currently is positioned.

11. Which of the following statements are true?
(a) The metric of ops/sec is not the absolute performance metric for AI accelerators; instead, the metric of tokens/sec is a good indicator for LLM inference.
(b) The CPU performance metric MIPS is not a fair indicator since it does not consider the complexity of machine instructions. A machine instruction set of short and fast instructions tends to have higher MIPS.
(c) NVIDIA NVLink and AMD Infinity Fabric are used to improve the data transfer rate

between chips as an LLM model training and inference cannot be conducted within a single AI accelerating chip.

(d) Data center computing resources consistently operate at high utilization levels due to rapidly increasing demand, while utilization imbalances across data centers can be effectively mitigated through network-enabled demand redistribution.

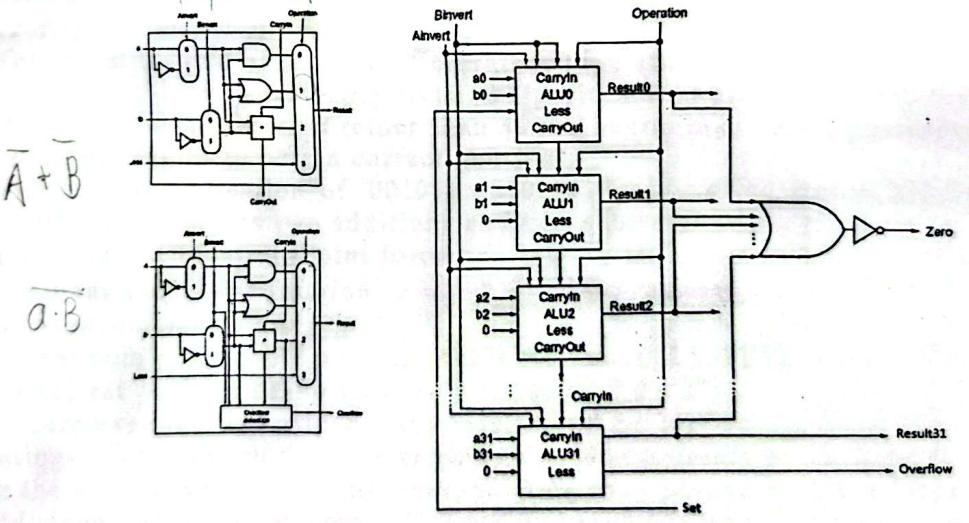
12. Which of the following statements are true?

(a) The machine instruction (add $t0, $s2, $t0) tells us two CPU design principles: I. Simplicity favours regularity, and II. Smaller is faster.

(b) Each basic block in a machine code program has a single-entry point, which may be a branch target, and a single exit point, with no branch instructions except possibly at the end. A basic block serves a purpose similar to blocks in C programs, in that it provides a unit for analyzing the lifetime of variables within the block.

(c) If the clock period of a CPU is set as n ns and the CPU supports multi-cycle instructions, comparison and branch instructions, like bge, that require 2n ns can be included in the instruction set without increasing the clock period.

(d) The key idea of lazy linkage is to use an indirection table that initially holds the address of a linker/loader resolver for a DLL routine and is later updated with the address of the loaded DLL routine, making subsequent calls more efficient.



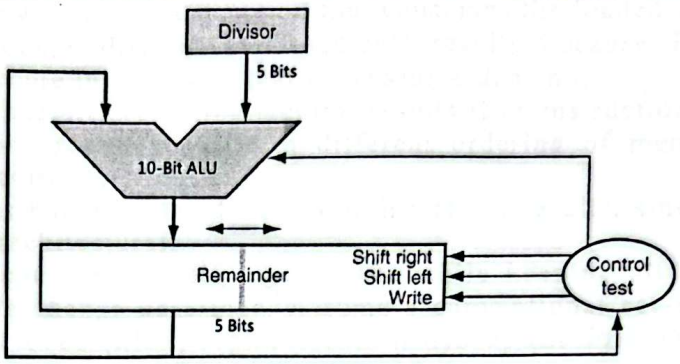
13. Consider the above ALU design. The control signals are grouped as (Ainvert, Binvert, Operation[1], Operation[0]). Which of the following statements are <u>true</u>?

(a) The control signal group (1, 1, 0, 1) performs a $\overline{a \cdot b}$ instruction.

(b) The following code sequence performs an overflow detection for unsigned addition ($t1+$t2): addu $t0, $t1, $t2; nor $t3, $t1, $zero; sltu $t3, $t3, $t2; beq $t3, $zero, Overflow.

(c) The *Set* signal is introduced to implement the *set-less-than* instruction and corresponds to the sum output of ALU31.

(d) The control signal group (0, 1, 1, 1) performs a *set-less-than* instruction.



14. Which of the following statements are **true**?
   (a) Consider the above ALU design and a multiplication of $00101_2 \times 00011_2$, the value of Remainder register is 0001111000_2 after three iterations of multiplication is conducted.
   (b) Consider the same ALU design and now a division of $01101_2 \div 00011_2$ using the restoring scheme is conducted. The value of Remainder register is 0000010001_2 after four iterations of division is conducted. Note that the shift operation is performed at the end of each iteration rather than at the beginning.
   (c) Consider the 5-bit non-restoring division algorithm that performs shift-left operation at the end of each iteration rather than at the beginning. The algorithms require 6 but not 5 iterations to obtain correct quotient.
   (d) Consider a multiplication of $00101_2 \times 01001_2$ applying the Booth algorithm. This multiplication requires two additions and two subtractions.
15. Consider a 16-bit Floating-Point format following IEEE Std 754 with one signed bit, 4-bit exponent and 11-bit fraction. $x = (-1)^s \times (1 + Fraction) \times 2^{(Exponent-Bias)}$. Which of the following statements are **true**?
   (a) The maximum positive FP number in this format is $1.1111\ 1111\ 111 \times 2^7$.
   (b) The smallest positive de-normalized number is 1.0×2^{-17}.
   (c) To ensure exponent is unsigned, the bias should be 15.
   (d) Floating-point addition requires aligning the binary points by shifting the operand with the smaller exponent; otherwise, shifting the operand with the larger exponent could make its fraction excessively large and result in overflow, while excessive shifting of the smaller exponent operand may cause its fraction to be truncated.
16. The following statements describe three pairs of pipelined processors. In each pair, the processors are identical except for the specified feature. In all cases, the processors execute the same compiled program. Ignore differences in execution time, pipeline stalls, and throughput. Consider only whether the processors may produce different architectural results.
   In Case A, processor A resolves data hazards using interlocks, while processor B uses full bypassing.
   In Case B, Processor A handles control hazards by stalling the pipeline until the branch target is known. Processor B uses a 'predict-not-taken' scheme and flushes the pipeline if the branch is taken.
   In Case C, processor A has a single memory port shared between instruction fetch and data access, while processor B has no structural hazards.

(a) Case A may produce different architectural results if a load instruction is immediately followed by an instruction that consumes the loaded value.
(b) Case B may produce different architectural results because Processor B may speculatively execute instructions that Processor A does not.
(c) Case C may produce different architectural results if an instruction fetch is delayed by a data memory access, causing a different ordering of memory accesses as observed architecturally.
(d) None of the cases will produce different architectural results, since all differences are purely microarchitectural.

17. A cache designer considers modifying a cache while keeping all parameters fixed except for the single change described. Assume a conventional set-associative cache with Least Recently Used (LRU) replacement. Which of the following statements is (are) correct?
(a) Doubling the cache line size while keeping cache capacity and associativity fixed strictly reduces the number of tag entries that must be compared on each cache access.
(b) Doubling the cache associativity while keeping cache capacity and line size fixed doubles the total number of tag entries in the cache.
(c) Doubling the capacity of a direct-mapped cache tends to reduce conflict misses but has minimal effect on compulsory misses.
(d) Doubling the cache line size tends to reduce compulsory misses by exploiting spatial locality and can increase conflict misses, while typically increasing the miss penalty.

18. A byte-addressable machine uses 36-bit virtual addresses and 30-bit physical addresses. To achieve high performance, the bits used to index a physical cache set must be obtainable without performing virtual-to-physical address translation through the TLB. The cache block size is fixed at 64 bytes.
Which of the following statements is (are) correct?
(a) Assuming a page size of 2^{13} bytes and an 8-way set-associative physically tagged cache, the maximum physical cache size that satisfies the above constraint is 32 KB.
(b) If the cache block offset is contained within the page offset, then cache indexing does not require virtual-to-physical address translation.
(c) Given a page size of 2^{13} bytes and a physical cache size of 2^{18} bytes, the minimum cache associativity required to avoid virtual-to-physical address translation during cache indexing is 32.
(d) If a 2^{15}-byte direct-mapped physical cache is used, the minimum page size that supports this cache configuration without requiring address translation for cache indexing is 2^{15} bytes.

19. Which of the following statements about branch prediction and conditional execution is (are) true?
(a) The Branch Target Buffer (BTB) is typically accessed during the instruction decode stage to enable early redirection.
(b) Indirect branches, such as those used in switch statements, are generally easier to predict than conditional branches because their targets remain fixed once the branch outcome has been resolved.
(c) The 2-bit dynamic branch prediction scheme improves prediction accuracy primarily by exploiting global branch history and correlations among nearby branches.
(d) Programs that use the CSEL instruction in ARMv8 architecture can potentially

reduce the number of conditional branch instructions by replacing control flow with conditional data selection.

20. Consider two compute systems, Machine A and Machine B, designed to accelerate linear-algebra workloads. Machine A can issue 3 multiply operations per cycle and sustains a memory bandwidth of 0.1 words per cycle. Machine B can issue 1 multiply operation per cycle and sustains a memory bandwidth of 0.5 words per cycle. Both machines have a multiplier latency of 1 cycle and operate at the same clock frequency.

Assume the following throughout this problem: instructions execute in program order; in each cycle, all multiplications whose dependencies are satisfied are issued, up to the number of available multipliers; only multiplications and loads from main memory incur cost; load latency is fully hidden once data is available; all kernels execute in steady state on large inputs; results are kept on-chip and are not written back to main memory; and at any time at most one kernel is executing on a machine.

Consider the following kernels:

- Matrix–vector multiplication of an N×N matrix with an N-element vector.
- Matrix–matrix multiplication of two N×N matrices.

Which of the following statements is (are) correct?

(a) The operational intensity of matrix–vector multiplication increases linearly with N, while the operational intensity of matrix–matrix multiplication increases quadratically with N.

(b) For sufficiently large N, matrix–vector multiplication is memory-bandwidth bound on both machines, and Machine B achieves higher performance due to its higher memory bandwidth.

(c) For sufficiently large N, matrix–matrix multiplication becomes compute-bound on Machine A, allowing it to outperform Machine B despite having lower memory bandwidth.

(d) The roofline model alone is insufficient to compare the performance of the two machines on these kernels, because instruction ordering and multiplier latency dominate performance.

二、題組(20%)，共四個題組，每一題組內所有小題全答對得 5 分，達錯任一小題或未作答得 0 分。題組 A 包含題號 21-25，題組 B 包含題號 26-28，題組 C 包含題號 29-32，題組 D 包含題號 33-35。

A. Consider a byte-addressable computer system with a 13-bit logical address, a total physical memory size of 4 KiB, and a page size of 64 Bytes. The system uses a segmented paging architecture where the logical address is divided into a Segment Number and a Segment Offset. The segment table provides a Base Address which is added to the Segment Offset to form a linear Address. This Linear Address is then translated via a single-level Page Table to a Physical Address.

Segment Table
(Segment Base Addr, Binary)

0000,0000,0000	(Idx 0)
0010,0000,0000	(Idx 1)
0001,1000,0000	(Idx 2)
0011,0000,0000	(Idx 3)
0000,1000,0000	(Idx 4)
0100,0000,0000	(Idx 5)
0101,0000,0000	(Idx 6)
0110,0000,0000	(Idx 7)

Page Table
(Frame Number, Decimal)

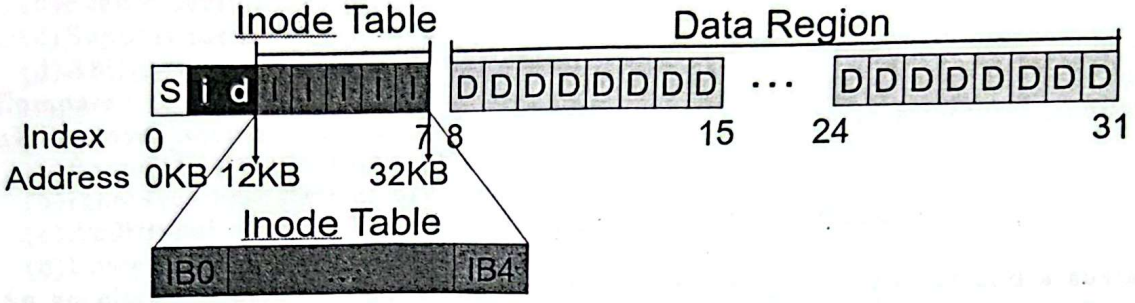
...	
12	(Idx 24)
5	(Idx 25)
33	(Idx 26)
8	(Idx 27)
...	

Given the Logical Address in hexadecimal format "16A4", and the tables below, answer the following questions.

21. What is the **Segment Index** derived from the logical address 16A4?
    (a) 3
    (b) 4
    (c) 5
    (d) 6
22. What is the **Segment Base Address** (in hexadecimal) retrieved from the Segment Table?
    (a) 0x200
    (b) 0x400
    (c) 0x500
    (d) 0x600
23. What is the calculated **Linear Address** (in hexadecimal)?
    (a) 0x2A4
    (b) 0x6A4
    (c) 0x4A4
    (d) 0x5A4
24. What is the **Virtual Page Number (VPN)** index used to look up the Page Table?
    (a) 24
    (b) 25
    (c) 26
    (d) 27
25. What is the final **Physical Address** (in hexadecimal)?
    (a) 0x864
    (b) 0x424
    (c) 0x244
    (d) 0xD44

B. Consider a file system called TFS that is divided into a series of blocks, the size of each block is 4 KB. The blocks are indexed from 0 to 31, assuming we have a tiny disk, with just 32 blocks. Let's refer to the region of the disk used for user data as the data region, which consists of the last 24 of the 32 blocks on the disk. Additionally, we allocate 5 of our 32 blocks for inode table denoted by I's in the diagram. An inode table is partitioned into multiple i-blocks indexed by i-block (IB) 0 to IB 4. An i-block stores

multiple inodes and the size of each inode is 256 bytes. Finally, the inode bitmap (i), the data bitmap (d), and the superblock (S) respectively use one 4KB block.



26. What is the maximum number of files where the TFS file system can manage?
    (a) 24
    (b) 80
    (c) 5
    (d) 16

27. The start address of the inode table on the TFS file system is 12 KB. This TFS file system uses five IBs to store the entire inode table. An IB stores multiple inodes, and the inode index starts from 0 in IB0 and is incrementally increased. The last inode is preserved in the IB4 with the index 79. Which IB preserves the inode where its index is 16?
    (a) 3
    (b) 2
    (c) 1
    (d) 0

28. The disk typically consists of a large number of addressable sectors, each of which is 512 bytes in size. To fetch the inodes where its index is 16, which sector of the disk would the file system issue a read to?
    (a) 32
    (b) 16
    (c) 8
    (d) 4

C. In recent years, general purpose processors have increasingly been supplemented, and in some cases replaced, by specialized accelerators such as GPUs and TPUs. These accelerators depart from traditional CPU designs by prioritizing high throughput computation, data level parallelism, and memory bandwidth over single thread latency and complex control flow. While such designs can deliver orders of magnitude improvements for selected workloads, they also introduce new architectural tradeoffs at the system level. The following questions examine the architectural principles that distinguish accelerators from CPUs, and the system-level consequences of these design choices.

29. Consider a multi-accelerator system executing a workload that performs frequent collective operations (e.g., all-reduce) on large tensors.
    Which interconnect property is most critical for maximizing sustained throughput?

(a) Minimum single-hop latency
(b) Peak bisection bandwidth
(c) Support for hardware cache coherence
(d) Ability to route packets out of order

30. Compared to a monolithic accelerator with the same aggregate compute resources, a chiplet-based accelerator design necessarily introduces which architectural tradeoff?
(a) Reduced arithmetic throughput per cycle
(b) Increased on-chip memory capacity
(c) Additional communication latency and bandwidth overhead
(d) Lower instruction issue width

31. An accelerator sustains a peak compute throughput of 100 TFLOP/s and a sustained memory bandwidth of 10 TB/s.
A kernel has an arithmetic intensity of 4 FLOPs per byte.
Which statement is correct?
(a) The kernel is compute-bound at 100 TFLOP/s
(b) The kernel is memory-bound at 40 TFLOP/s
(c) The kernel is control-bound due to insufficient parallelism
(d) The roofline model cannot be applied to this kernel

32. Given a fixed power and area budget for an accelerator intended primarily for large-model inference, which architectural investment most directly improves end-to-end throughput?
(a) Increasing peak floating-point units per cycle
(b) Increasing instruction window size
(c) Increasing sustained on-chip and inter-chip data bandwidth
(d) Increasing branch prediction accuracy

```
unsigned int fib (unsigned int n) {
 if (n < 2) return(n);
 else return(fib(n-1)+fib(n-2));
}
```

```
1.  fib:
2.  addi  $sp, $sp, -12
3.  sw    $ra, 0($sp)
4.  sw    $s1, 4($sp)
5.  sw    $a0, 8($sp)
6.  slti  $t0, $a0, 2
7.  beq   $t0, $0, L1
8.  addi  $v0, $a0, 0
9.  j     EXIT
10. L1:
11. addi  $a0, $a0, -1
12. jal   fib
13. addi  $s1, $v0, 0
14. addi  $a0, $a0, -1
15. jal   fib
16. add   $v0, $v0, $s1
17. EXIT:
18. lw    $ra, 0($sp)
19. lw    $a0, 8($sp)
20. lw    $s1, 4($sp)
21. addi  $sp, $sp, 12
22. jr    $ra
```

Name	$zero	$v0-$v1	$a0-$a3
Reg. no.	0	2-3	4-7

$t0-$t7	$s0-$s7	$t8-$t9	$sp
8-15	16-23	24-25	29

	Op (31:26)							
28-26 31-29	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	R-format	Bltz/gez	Jump	Jal	Beq	Bne	Blez	Bgtz
1(001)	Addi	Addiu	Slti	Sltiu	Andi	Ori	Xori	Lui
2(010)	TLB	FlPt						
3(011)								
4(100)	Lb	Lh	Lwl	Lw	Lbu	Lhu	lwr	
5(101)	Sb	Sh	Swl	Sw			Swr	
6(110)	Lwc0	Lwc1.						
7(111)	Swc0	swc1						

D. Consider the above C codes and its related 32-bit MIPS assembly codes.

33. Which statement is correct?
(a) If register *$s1* is replaced with *$t0*, register utilization can be improved without side-effects.

(b) It is not necessarily to save register *$s1* at the beginning of function, which can reduce one instruction.

(c) It is necessary to save registers *$ra* and *$a0* at the start of each function to avoid data loss in situations where a callee subsequently acts as a caller.

(d) One of the reasons why using *slti* and *beq* rather than using a single *blti* (branch less than immediate) instruction is that *blti* requires two immediate fields that cannot be accommodated within a single 32-bit MIPS instruction format.

34. Assume that the execution of each *j, jal, beq, and jr* instruction takes 3 clock cycles. As invoking the function fib(2), the total number of clock cycles to performing all *j, jal, beq, and jr* instructions is:

(a) 30
(b) 27
(c) 24
(d) 33

35. The above tables present the register numbers and the corresponding translations of assembly instructions into machine codes. The machine code of the instruction *beq $t0, $0*, L1 (Line 7) is:

(a) 000 100 01000 00000 1111 1111 1111 1110.
(b) 100 000 01000 00000 0000 0000 0000 0010.
(c) 000 100 01000 00000 0000 0000 0000 0010.
(d) 100 000 01000 00000 1111 1111 1111 1110.