國立成功大學

115學年度碩士班招生考試試題

編　號：138

系　所：電機資訊學院-資訊聯招

科　目：計算機組織與系統

日　期：0203

節　次：第 1 節

注　意：1.不可使用計算機
2.請於答案卷(卡)作答，於試題上作答，不予計分。

- 本試題分兩大部分，分別是**計算機組織**與**作業系統**，各佔 50 分，總分 100 分。
- **請按題號次序彙整答案，並統一於答案卷上製作如下表格作答。**

題號	答案
計算機組織 (50%)	
問題 1	(1) ? (2) ? (3) ? (4) ? (5) ?
問題 2	(1) ? (2) ? (3) ? (4) ? (5) ?
作業系統 (50%)	
選擇題	作答請依題目順序寫題號後，在題號後標出答案 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15.
簡答題 1	作答
簡答題 2	作答

計算機組織：

Total Score: 50 points (每小題 5 分)

【READ BEFORE ANSWERING: Reference Concepts】

Please read the following principles to assist your reasoning:

1. **Cache Coherence:** The MESI protocol is a foundational hardware-based cache coherence mechanism designed to maintain data consistency across the private caches of a multi-core system. It functions by assigning one of four distinct states to every cache line—Modified (M), Exclusive (E), Shared (S), or Invalid (I)—and monitoring bus traffic through a process known as "snooping." Under this protocol, if a core intends to write to a data block currently in the Shared state, it must first broadcast an Invalidate signal to all other cores to force their local copies into the Invalid state; this allows the initiating core to acquire exclusive ownership and transition to the Modified state, ensuring that no two cores can hold conflicting versions of the same data and that all processors always access the most recent values.

2. **Pipeline Overhead:** Pipelining increases **Throughput**. However, due to 'Pipeline Registers' inserted between stages, the setup/propagation time causes the total **Latency** of a 'single instruction' to increase slightly compared to a non-pipelined design.
3. **Memory Hierarchy Goals:** L1 Cache optimizes for Speed (Hit Time). L2 Cache optimizes for reducing the **Global Miss Rate** (intercepting Misses before they reach the slow DRAM). L2 is typically slower and larger than L1.
4. **Branch Limits:** In fixed-length ISAs (e.g., RISC-V), the Offset for conditional branches is limited (e.g., 12-bit, approx KB). Long-distance jumps (e.g., 1MB) require compiler workarounds (like trampolines).
5. **Spatial Locality:** Cache moves data in Blocks. If a program's access **Stride** is too large (e.g., jumping across rows in a large 2D array), it fails to utilize the data within the block, leading to a high Miss Rate.

INSTRUCTIONS:

1. Use the Reference Concepts above to assist your reasoning.
2. **WARNING:** Column B contains **'Distractors' (Incorrect Statements)**.
3. Column B has **MORE** options than Column A. Select the **BEST** match for each item in Column A.

問題 1: Core Mechanisms (25 points)

Column A: Mechanism	Column B: Characteristics & Distractors
1. Pipelining	(A) Primarily increases Throughput; single instruction Latency slightly increases due to register overhead.
2. L2 Cache	(B) Splits execution to reduce single instruction Latency to exactly 1/N.
3. Cache Block (Line)	(C) Hardware guarantees zero Data Hazards between any instructions.
4. MESI Protocol	(D) Designed to intercept L1 Misses and reduce Global Miss Rate to DRAM.
5. PC-relative Addressing	(E) Designed to be smaller than L1 to achieve higher speeds.
	(F) Uses block transfers to exploit Spatial Locality.
	(G) Optimizes access for non-contiguous structures like Linked Lists.
	(H) Requires 'Invalidate' signal before writing to Shared data.
	(I) Managed by OS; triggers Context Switch on write.
	(J) Used for conditional branches with limited range (e.g., ±4KB).
	(K) Allows jumping to any address without bit-width limits.

問題 2: Scenario Analysis (25 points)

Column A: Scenario	Column B: Behavior & Distractors
6. Convert Single-cycle to 5-stage Pipeline	(A) Total program execution time instantly becomes 1/5th.
7. Accessing 2D array via A[j][i] (Strided)	(B) Individual instruction Latency may slightly increase due to overhead.
8. Core A writes to 'Shared' (S) data	(C) Destroys Spatial Locality; Miss Rate increases significantly.
9. Branch target is 1MB away	(D) Spatial locality remains excellent regardless of access order.
10. L1 Miss but L2 Hit	(E) Can write directly since it is Shared; no need to signal.
	(F) Must acquire Bus Exclusivity and Invalidate Core B's copy.
	(G) Hardware auto-extends 12-bit offset to 32-bit.
	(H) Compiler uses 'Invert Branch + Unconditional Jump' combination.
	(I) Access latency is faster than DRAM but slower than L1.
	(J) L2 speed is identical to L1, zero latency difference.
	(K) Triggers Write-Through to HDD directly.

作業系統：

一、選擇題：(30 分，每題 2 分)

1. (2%) If a few bits in an HDD sector or NVM page are corrupted, the controller can recover the correct values using ECC only if
   (a). the bit corruption is limited to the data portion of the sector/page.
   (b). the bit corruption hasn't occurred in the ECC of the sector/page.
   (c). the bit corruption hasn't occurred in the ECC or the header of the sector/page.
   (d). number of bits corrupted is low irrespective of where the corruption occurs in the sector/page.
2. (2%) A full bootstrap program
   (a). is stored in ROM to ensure to avoid any infection from viruses.
   (b). is part of an operating system
   (c). may be infected by viruses.
   (d). runs after the operating system has been loaded in memory.
3. (2%) IBM's indexed sequential access method (ISAM)_____.
   (a). uses an index file for sequential access

(b). uses a small master index (kept in memory) that points to disk blocks of a secondary index, while the secondary index blocks point to the actual file blocks

(c). use the pointers (kept in memory) of the actual file blocks

(d). uses the pointers (kept in memory) that points to disk blocks of a master index which points to disk blocks of a secondary index, while the secondary index blocks point to the actual file blocks

4. (2%) Which of the following is true about dynamic storage allocation?

(a). Worst fit provides the best storage utilization.

(b). First fit requires less time for allocation than worst fit on average.

(c). Best fit is clearly better than first fit in terms of time and storage utilization.

(d). First fit is clearly better than best fit in terms of time and storage utilization.

5. In DMA-based I/O,

(a). cycle stealing can slow down the CPU computation, but off-loading the data-transfer work to a DMA controller generally improves the total system performance.

(b). CPU is relieved from data transfer and is interrupted after every word has been transferred to initiate the next word transfer.

(c). interrupt mechanism is not used.

(d). data is transferred to/from a single block of memory, but not to/from multiple blocks.

6. (2%) When designing an embedded I/O data acquisition path, engineers often choose between an interrupt-driven approach and polling. Which statement is correct?

(a). Under high event rates, an interrupt-driven design is always more power efficient than polling because the CPU does not repeatedly read a status register, and interrupts do not introduce meaningful context-switch overhead.

(b). In periodic polling, the average detection latency is mainly determined by the polling period. If event arrivals are uniformly distributed within each polling interval, the expected detection latency is approximately half the polling period, while the worst-case detection latency can be bounded by roughly one polling period (ignoring the polling routine's execution time).

(c). Using edge-triggered interrupts eliminates interrupt storms, so edge-triggered interrupts are always preferable to level-triggered interrupts for high-frequency events, and no additional debouncing or throttling mechanisms are necessary.

(d). With polling, increasing the polling frequency simultaneously reduces latency and CPU utilization because the system can detect events sooner and return to sleep earlier; therefore high-frequency polling is typically better for power.

7. (2%) You are implementing a very simple file system without journaling. Creating a new file involves allocating one data block, writing the data, initializing the inode, and adding a directory entry that points to that inode. Writes can be reordered by caches unless explicitly flushed. Which sequence best preserves crash consistency, meaning that after a crash you will not end up with a directory entry that references an uninitialized inode or an inode that references garbage data (you may still allow space leaks)?

(a). Persist the directory entry first, then persist the inode, then write the data block, because the name must exist before the file content is visible.

(b). Persist the inode first, then persist the directory entry, then persist the data block, because the inode can be updated later to fix any inconsistency.

(c). Persist the data block first, then persist the initialized inode that points to it, then persist the directory entry, because reachability should be created only after targets are durable.

(d). Persist the data block and inode in any order, because once the directory entry is durable it guarantees both metadata and data consistency.

8. (2%) A system has two memory tiers. Tier 0 is small and low latency. Tier 1 is large and higher latency. The operating system manages pages and can migrate them between tiers. Which policy choice most directly reduces migration thrashing (ping pong) while still capturing hot pages?

(a). Migrate any page to Tier 0 immediately on its first access in Tier 1, because first touch is the earliest indicator of hotness.

(b). Use page faults as the primary hotness signal and migrate pages only when they fault repeatedly, because faults capture the most performance critical pages.

(c). Use hardware accessed bits (and optionally dirty bits) sampled over fixed epochs, apply hysteresis with a promotion threshold and a separate demotion threshold, and enforce a minimum residency time before a page is eligible to move again.

(d). Always keep the most recently allocated pages in Tier 0, because temporal locality implies newer pages are hotter than older pages.

9. (2%) Assume a demand-paged system with 3 physical frames and an LRU replacement policy. The page reference string is:

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

How many page faults occur?

(a). 8

(b). 9

(c). 10

(d). 11

10. (2%) Consider enabling huge pages for a workload that scans large arrays with strong spatial locality. Which statement is correct?

(a). Huge pages can reduce TLB misses by increasing TLB reach, but they can increase internal fragmentation and can make some operations, such as copy on write and page migration, more expensive because the unit of management is larger.

(b). Huge pages primarily improve performance by eliminating page faults, since huge pages prevent the kernel from needing demand paging for that region.

(c). Huge pages reduce page table walk overhead only if the CPU has a larger TLB, otherwise the miss rate cannot change.

(d). Huge pages always reduce memory usage because fewer page table entries are needed, so fragmentation cannot increase.

11. (2%) On a traditional HDD (mechanical disk), which statement about common I/O schedulers and latency is correct?

(a). SCAN (elevator) can reduce average seek distance versus FCFS, but it may still cause long waits for some requests depending on direction and queueing, so fairness improves compared with SSTF but tail latency is not necessarily minimized.

(b). SSTF always provides the best tail latency because it minimizes seek distance for every request, which prevents starvation by construction.

(c). FCFS minimizes variance in response time because it preserves arrival order, which avoids reordering induced latency spikes.

(d). The same scheduler conclusions apply unchanged to NVMe SSDs because both HDDs and SSDs are dominated by seek time.

12. (2%) Which situation is most likely to increase write amplification in an SSD using an FTL with garbage collection?

(a). Large sequential writes to mostly empty media, because sequentiality maximizes valid page copying during garbage collection.

(b). Using overprovisioning, because extra spare area reduces available user space and forces the SSD to copy more data on every write.

(c). Placing hot and cold data together in the same physical blocks, because it reduces metadata overhead and lowers the frequency of garbage collection.

(d). Small random overwrites to a nearly full drive, especially without TRIM, because many blocks contain a mix of valid and invalid pages, increasing copying during garbage collection.

13. (2%) Three periodic tasks have deadlines equal to periods:

T1: C1 = 1 ms, P1 = 4 ms

T2: C2 = 1.5 ms, P2 = 6 ms

T3: C3 = 2 ms, P3 = 12 ms

Assume the standard theoretical model: independent periodic tasks, preemptive uniprocessor, deadline equals period, negligible overhead. Which statement is correct when applying utilization-based schedulability tests?

(a). The total utilization is U=1/4+1.5/6+2/12 = 2/3. Therefore the task set is schedulable under EDF (since $U \leq 1$), and it is also guaranteed schedulable under RM bound for n = 3, $U \leq 3\left(2^{\frac{1}{3}} - 1\right) \approx 0.779$.

(b). The task set is not schedulable under EDF because EDF requires $U < 1$.

(c). The task set is not guaranteed schedulable under RM because RM's utilization bound is $U \leq \ln 2 \approx 0.693$for three tasks.

(d). The task set is guaranteed schedulable under RM only if the periods are harmonic. Since 6 is not a multiple of 4, no RM guarantee can be made from utilization.

14. (2%) Which statement about priority inversion and its mitigation is correct?

(a). Priority inheritance ensures that a high-priority task can never be blocked by a lower-priority task, since the lower-priority task is immediately boosted above all others and runs to completion.

(b). Priority inversion occurs only with non-preemptive kernels; in a fully preemptive RTOS it cannot happen because the scheduler always runs the highest-priority task.

(c). Replacing mutexes with spinlocks eliminates priority inversion because spinlocks never block; they only busy-wait.

(d). Priority ceiling protocols can prevent deadlock and bound blocking time by restricting which tasks may lock a resource based on a preassigned ceiling, but they may reduce parallelism by being conservative.

15. (2%) In a system with a non-coherent CPU data cache, a device performs DMA writes into a memory buffer that the CPU later reads. Which action is correct for the CPU/driver to ensure it reads the newly DMA-written data?

(a). Before starting the DMA write, the CPU must invalidate the cache lines of the destination buffer; after DMA completion, no cache maintenance is needed because DMA writes update memory directly.

(b). The CPU should clean (write back) the cache lines after DMA completion, because writing back ensures the cache and memory match the device output.

(c). No cache maintenance is required as long as the buffer is aligned to cache-line boundaries and the DMA length is a multiple of the cache-line size.

(d). After DMA completion, the CPU must invalidate (or otherwise ensure invalidation of) the cache lines covering the buffer before reading it, because the CPU cache may still hold stale copies.

二、簡答題：(共 20 分)

1. (8%) An orphan process is a process whose parent process no longer exists, i.e., either it has finished or terminated without waiting for its child processes to terminate. Please finish the following code within the box, which can create an orphan process.

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
int main ()
{
    int pid = fork();
   if (pid > 0)
   {
    // this is parent process
   }
  else if (pid == 0)
  {
    // this is child process
  }
  return 0;
}
```

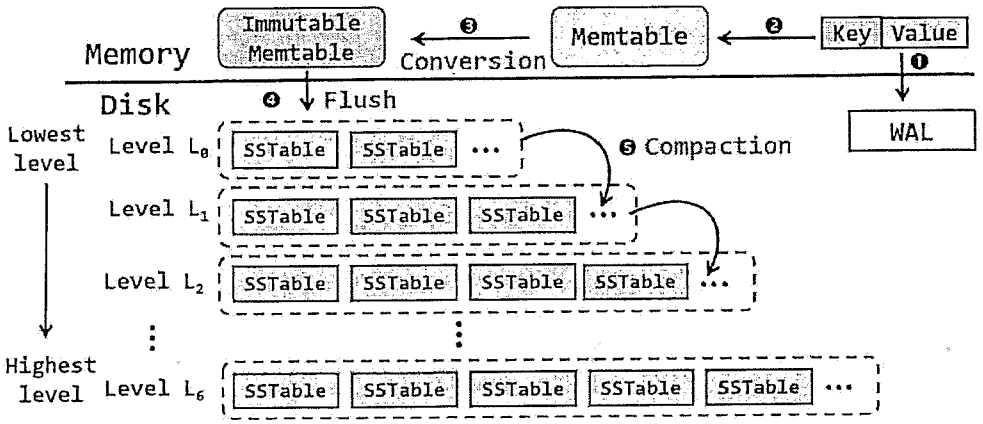
2. (12%) Read the following description about the LSM tree and answer the questions.

An LSM-tree is a disk-oriented data structure that improves write efficiency by buffering updates in memory and flushing them to disk in batches. By doing so, it turns many small random writes into fewer sequential writes, which is especially beneficial for write-heavy workloads.

LevelDB is a popular open-source LSM-tree based key-value store. New updates are first appended to a write-ahead log (WAL) for crash recovery, then inserted into an in-memory MemTable (typically a skip list). When the MemTable becomes full, it is frozen as an Immutable MemTable, and a new MemTable is created for incoming writes. The Immutable MemTable is then flushed to disk as an SSTable, where keys are stored in sorted order.

On disk, LevelDB organizes SSTables into multiple levels (L0, L1, ...). Data is first flushed into L0, where SSTables may overlap in key ranges. When L0 grows beyond a threshold, LevelDB triggers compaction: it selects one or more SSTables from level Li and merges them with overlapping SSTables in level Li+1 using a merge-sort. The merged output is written as new SSTables in Li+1. From L1 onward, SSTables are maintained to be non-overlapping within each level.

Deletes are handled using tombstones. A delete inserts a marker rather than immediately removing data. During compaction, entries covered by tombstones are dropped, which permanently removes the data. Reads search for the newest version first, checking the MemTable, then the Immutable MemTable, then SSTables from L0 down to deeper levels until the key is found or confirmed absent.



(a) (4%) Explain where write amplification comes from in an LSM-tree KV store such as LevelDB. Your answer should connect amplification to the maintenance workflow described above.

(b) (4%) Explain where read amplification comes from in an LSM-tree KV store. In particular, why can L0 increase the number of files a lookup must examine?

(c) (4%) Under what conditions does compaction become the bottleneck and lead to higher tail latency or write stalls? Describe the situation using the LSM maintenance workflow (MemTable, Immutable MemTable, flush to L0, and compaction across levels).