複選題 Multiple Choices with at Least One Correction Choice

- 至少有一個選項為正確答案。At least one choice is a correct answer.
- 每一個選項分別計分。Each choice is graded individually.
- 不倒扣。No penalty for an incorrect answer.
- 整題空白，則該題零分。Zero point for not selecting any choice.

※ 注意：請用 2B 鉛筆作答於答案卡，並先詳閱答案卡上之「畫記說明」。

Q1 (5pts) Regarding system design, modern operating systems are often designed for specific scenarios such as smartphones, artificial intelligence servers, and safety-critical systems (e.g., avionics, vehicles, medical devices). Select correct description(s).

(A) Energy efficiency is an important design goal for smartphones.
(B) Efficient GPU management is an important design goal for artificial intelligence servers.
(C) High memory bandwidth is an important design goal for artificial intelligence servers.
(D) Improving average-case performance is more important than improving worst-case performance for safety-critical systems.
(E) Nondeterministic scheduling is usually preferred for safety-critical systems.

Q2 (5pts) Regarding processes and threads, select correct description(s).

(A) Threads belonging to the same process share the same stack section in the process address space.
(B) Threads belonging to the same process share the same data section in the process address space.
(C) Threads belonging to the same process share the same program counter.
(D) Context switching between threads is usually faster than that between processes.
(E) With synchronous threading, the parent thread creates a child and resumes its execution.

Q3 (5pts) Regarding virtual memory, select correct description(s).

(A) Under pure demand paging, a process may begin execution without any page loaded in physical memory and immediately incur a page fault on its first instruction fetch.
(B) The optimal page replacement algorithm cannot be implemented in practice as it requires future knowledge of page references.
(C) A page replacement algorithm that satisfies the stack property cannot suffer from the Belady's anomaly.
(D) If the total demand for frames, computed as the summation of all processes' working-set sizes, exceeds the number of available physical frames, the system may enter a thrashing state.
(E) Compared to global page replacement, local page replacement generally provides more predictable per-process performance at the cost of lower overall system throughput.

Q4 (5pts) Regarding security, select correct description(s).

(A) User-kernel mode separation helps prevent user programs from directly accessing hardware resources.
(B) Address space isolation helps prevent processes from unauthorized memory access by other processes.
(C) Mandatory access control allows users to freely change file permissions for their own files.
(D) Sandboxing imposes an irremovable set of restrictions on a process right after the execution of the main() function of the process.
(E) Encrypting data at rest can help protect user privacy.

見背面

Q5 (5pts) Regarding scheduling and neglecting context-switching time, given a set of processes, select correct description(s).

(A) If the processes arrive at the same time, the optimal nonpreemptive scheduling for minimizing average waiting time is the Shortest-Job-First (SJF) scheduling.
(B) If the processes arrive at different times, the optimal nonpreemptive scheduling for minimizing average waiting time is the Shortest-Job-First (SJF) scheduling.
(C) If the processes arrive at different times, the optimal preemptive scheduling for minimizing average waiting time is the Shortest-Remaining-Time-First (SRTF) scheduling.
(D) If the processes have deadlines and arrive at different times, the optimal nonpreemptive scheduling for schedulability is the Earliest-Deadline-First (EDF) scheduling.
(E) If the processes have deadlines and arrive at different times, the optimal preemptive scheduling for schedulability is the Earliest-Deadline-First (EDF) scheduling.

Q6 (5pts) Regarding synchronization, P_i and P_j are two processes sharing the critical section with the following assumptions:

- The "load" and "store" machine-language instructions are atomic.
- The two processes share "turn" to indicate whose turn.

Given the following two approaches:

Approach 1 for Process P_i	Approach 2 for Process P_i

```
while (true){
    turn = i;
    while (turn == j)
        ;
    /* critical section */
    turn = j;
    /* remainder section */
}
```
	```
	while (true){

	    while (turn == j)
	        ;
	    /* critical section */
	    turn = j;
	    /* remainder section */
	}
	```

Note: exchange i and j for P_j	Note: exchange i and j for P_j

Select correct description(s).

(A) Approach 1 satisfies the requirement of mutual exclusion.
(B) Approach 1 satisfies the requirement of bounded waiting.
(C) Approach 2 satisfies the requirement of mutual exclusion.
(D) Approach 2 satisfies the requirement of progress.
(E) Approach 2 satisfies the requirement of bounded waiting.

Q7 (5pts) Regarding a general computer architecture and ISA design principles, select correct description(s).

(A) Increasing pipeline depth improves performance primarily by decreasing the CPI.
(B) Removing byte load and store instructions from the ISA can potentially increase the code sizes of most applications.
(C) Transitioning from a load/store ISA to a memory-register ISA (allowing memory operands in arithmetic instructions) typically increases instruction count.
(D) Increasing the number of architectural registers can reduce instruction count and thus improve overall performance.
(E) Incorporating complex addressing modes into ISA reduces instruction count but complicates hardware implementations, leading to an increase in the clock cycle time.

接次頁

Q8 (5pts) Regarding cache conflict patterns, consider the following C code executing on a system with a 32 KB direct-mapped L1 data cache, 64-byte cache lines, and 4-byte integers:

```
int A[8192], B[8192];   // Arrays allocated contiguously, cache-line aligned
for (int i = 0; i < 8192; i++) {
    A[i] = A[i] + B[i];
}
```

Select correct description(s).

(A) Every access to B[i] results in a cache miss due to conflict with A[i].
(B) Changing the cache to 2-way set-associative with the same total size would eliminate all conflict misses.
(C) Adding padding of 8192 integers between arrays A and B would eliminate conflict misses.
(D) Loop interchange would improve cache performance for this code.
(E) Prefetching B[i+16] in each iteration would eliminate all cache misses.

Q9 (10pts) Regarding LLM inference, quantization, and memory hierarchy, consider running inference for a large language model on a CPU system with the following specifications:

- L1 cache: 32 KB per core, 4 cycles latency
- L2 cache: 256 KB per core, 12 cycles latency
- L3 cache: 32 MB shared, 40 cycles latency
- Main memory: 64 GB DDR5, 200 cycles latency, 50 GB/s bandwidth
- The model has 7 billion parameters

The following code performs a single matrix-vector multiplication for one transformer layer:

```
// Weight matrix W: 4096 x 4096, activation vector x: 4096
// FP16: 2 bytes per element, INT8: 1 byte per element, INT4: 0.5 bytes per element
void matvec(void *W, float *x, float *y, int N, int precision) {
    for (int i = 0; i < N; i++) {
        float sum = 0;
        for (int j = 0; j < N; j++) {
            float w = dequantize(W, i, j, precision);
            sum += w * x[j];
        }
        y[i] = sum;
    }
}
```

Select correct description(s).

(A) With FP16 weights, the 4096×4096 weight matrix requires 32 MB and fits comfortably in L3 cache with room for activations and other data.
(B) Quantizing from FP16 to INT4 reduces memory bandwidth requirements by 4× for weight loading.
(C) The activation vector x (4096 floats = 16 KB) can remain in L1 cache throughout the computation if accessed with proper loop ordering.
(D) For a memory-bound workload, switching from FP16 to INT8 would approximately double the achievable tokens per second.
(E) The 7B parameter model in INT4 format requires approximately 3.5 GB, making it possible to fit entirely in L3 cache.

見背面

Q10 (10pts) Regarding CPU-accelerator data orchestration for LLM inference, consider a system designed for LLM inference with the following specifications:

- Hardware Configuration:
  - CPU: 128 GB DDR5 memory, 50 GB/s memory bandwidth, 2 TFLOPS INT8 throughput
  - NPU (Neural Processing Unit): 8 GB on-chip SRAM, 1 TB/s internal bandwidth, 400 TFLOPS INT8 throughput
  - Interconnect: PCIe 5.0 x16, 50 GB/s effective transfer bandwidth
- Model Characteristics (70B parameter LLM, INT8 quantization):
  - Total model weights: 70 GB
  - Number of transformer layers: 80
  - Weight size per layer: 875 MB
  - Computation operations per layer: 140 billion INT8 operations

Since the NPU delivers 200× higher computation throughput than the CPU, running inference on the CPU alone would be impractically slow (35 seconds per token versus 175 ms on NPU). The system must use the NPU for all matrix computations. However, the NPU's 8 GB SRAM cannot hold the 70 GB model, requiring layer weights to be streamed from CPU memory during inference.

The following code generates a single output token:

```
void generate_one_token() {
    // Process all 80 layers sequentially
    for (int L = 0; L < 80; L++) {

        // Transfer: CPU memory -> NPU SRAM (875 MB per layer)
        pcie_transfer_to_npu(weight[L], 875 MB);

        // Compute: 140 billion INT8 operations at 400 TFLOPS
        npu_compute_layer(L);
    }
    sample_output_token();
}
```

- Calculated Latencies Per Layer:
  - PCIe transmission latency: 875 MB ÷ 50 GB/s = 17.5 ms
  - NPU computation latency: 140×10^9 ops ÷ 400×10^{12} ops/s = 0.35 ms

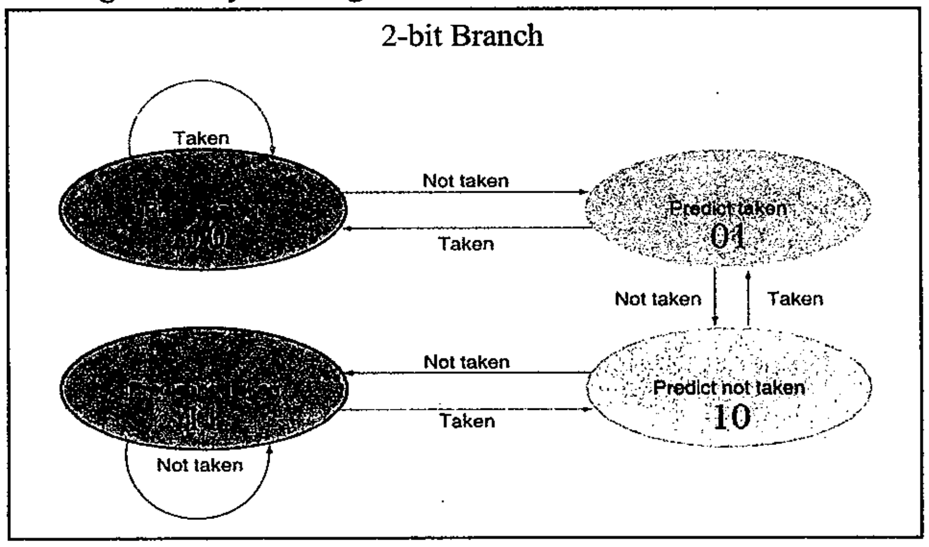
Select correct description(s).

(A) The total PCIe transmission latency per token (1.4 seconds) is 50× longer than the total NPU computation latency (28 ms), making this system severely PCIe-bandwidth-bound.
(B) Quantizing from INT8 to INT4 would halve the weight size to 35 GB, reducing per-token latency from approximately 1.4 seconds to approximately 0.7 seconds.
(C) If the NPU had 70 GB of SRAM instead of 8 GB, all weights would remain resident and per-token latency would drop to approximately 28 ms.
(D) Running inference entirely on the CPU would achieve lower latency than using the NPU because it eliminates PCIe transmission overhead.
(E) Doubling the PCIe bandwidth to 100 GB/s would reduce total per-token latency by exactly 50%.

接次頁

Q11 (10pts) You are analyzing a RISC-V 6-stage pipeline consisting of IF (Fetch), ID (Decode), EX (Execute), MEM1&MEM2 (Memory), and WB (Write-Back) stages. The pipeline utilizes a split-phase register file, where writes occur in the first half of the clock cycle and reads in the second. A 2-bit branch predictor is used, and a branch instruction is resolved in the EX stage. The processor implements a branch target buffer. Therefore, when a branch is predicted "taken", the target address is also predicted. Assume this loop executes 10 iterations. Consider the following assembly code segment:

```
Loop:
    lw     x3, 0(x11)
    lw     x4, 0(x12)
    add    x5, x3, x4
    sw     x5, 0(x11)
    addi   x11, x11, 4
    addi   x12, x12, 4
    bne    x11, x13, loop
```



Assume the predictor state is initialized to "Predict Non-Taken, 11", and the loop is executed multiple times. The execution cycles of an iteration are measured as the distance between the start of its Instruction Fetch (IF) stage and the start of the next iteration's IF stage (or the beginning of the first instruction that exits the loop). For example, if the i-*th* iteration begins its IF stage at cycle 12 and the (i+1)-*th* iteration begins at cycle 25, the total execution time for the i-*th* iteration is 13 cycles. Select correct description(s).

(A) With full hardware forwarding, the second iteration of the first loop executes in a total of 11 cycles
(B) With full hardware forwarding, the second iteration of the first loop executes in a total of 9 cycles.
(C) Without hardware forwarding, the last iteration of the first loop executes in a total of 18 cycles.
(D) Without hardware forwarding, the last iteration of the first loop executes in a total of 21 cycles.
(E) The prediction accuracy of the bne instruction is 90% except for the first loop.

Q12 (10pts) You are evaluating the energy efficiency of two RISC-V dual-core processors running a specific workload. When only one core is active, the second core is power-gated (i.e., no power consumption). Once the workload completes, both cores are power-gated.

- **Workload**: 10^9 instructions, of which 75% are parallelizable (i.e., can be split on two cores without incurring overheads) and the remaining 25% are strictly serial.
- **Processor A**: Two cores, Frequency: 3.0 GHz, Voltage: 1.1V, Avg. CPI (per core) = 1.2, Capacitance = 1.2 nF, Leakage Power = 500mW (total for 2 cores)
- **Processor B**: Two cores, Frequency: 2.0 GHz, Voltage: 0.9V, Avg. CPI (per core) = 1, Capacitance = 1.0 nF, Leakage Power = 400mW (total for 2 cores)
- **Dynamic power** = C x V^2 x F

Select correct description(s).

(A) In both Processor A and Processor B, 25% of the total execution time is executed in serial mode.
(B) Processor A consumes the same leakage energy as Processor B.
(C) $2 <$ Energy(A) / Energy(B) ≤ 2.3
(D) If the metric of energy efficiency is defined as Energy x Delay, Processor A is more energy efficient than Processor B.
(E) Although Processor A operates at a higher frequency than Processor B, the target workload runs faster on Processor B due to its lower average CPI.

見背面

Q13 (10pts) Regarding file allocation, a block size is 4,096 bytes, and a pointer to a block is 8 bytes. We will allocate an N-byte file to an unlimited disk space, where using a file allocation table is not allowed.

- The contiguous allocation allocates W blocks for the file.
- The linked allocation without clustering allocates X blocks for the file and the corresponding pointers.
- The linked allocation with clustering and 2 blocks as a cluster allocates Y blocks for the file and the corresponding pointers.
- The indexed allocation, where the linked allocation without clustering is used for the indexes, allocates Z blocks for the file and the corresponding pointers.

Select correct description(s).

(A) When N is approaching infinity, $W / X = 512 / 513$.
(B) When N is approaching infinity, $W / Y = 1{,}024 / 1{,}025$.
(C) When N is approaching infinity, $W / Z = 512 / 513$.
(D) When N is approaching infinity, $X > Y$.
(E) When N is approaching infinity, $X = Z$.

Q14 (10pts) Regarding system design and scheduling, consider the following system and requirements:

- There is one CPU executing tasks. The CPU cannot execute multiple tasks at the same time, and the task execution is nonpreemptive.
- There are two queues of tasks: $s_1, s_2, \ldots, s_m$ and $t_1, t_2, \ldots, t_n$. The task execution must follow the order of each queue, meaning that s_i must be executed before s_j if $i < j$, and t_i must be executed before t_j if $i < j$.
- Each task has its arrival time: $A_1, A_2, \ldots, A_m$ and $B_1, B_2, \ldots, B_n$, representing the earliest time that the CPU can execute the task.
- Each task has its execution time on the CPU: $E_1, E_2, \ldots, E_m$ and $F_1, F_2, \ldots, F_n$, representing the time from the start to the completion of the execution of the task.
- The execution between two tasks has a switching time, representing the required time switching from the completion of the execution of one task to the start of the execution of the other task. The switching time of the tasks from the same queue is W, and the switching time of the tasks from different queues is W*.
- The goal of system design is to decide the start time of each task so that the requirements above are satisfied and the completion time of the last-completed task (which is either s_m or t_n) is minimized.

Given the following problem instance with $m = n = 7$, $W = 1$, and $W^* = 3$:

Task	s_1	s_2	s_3	s_4	s_5	s_6	s_7
Arrival Time (A_i)	0	25	50	100	107	108	141
Execution Time (E_i)	16	8	24	8	10	5	4
Task	t_1	t_2	t_3	t_4	t_5	t_6	t_7
Arrival Time (B_i)	0	25	50	75	101	108	142
Execution Time (F_i)	8	16	4	4	6	4	16

The minimized objective is $100X + 10Y + Z$, where X, Y, Z are integers and $0 \le X, Y, Z \le 9$. Select correct description(s).

(A) There are multiple optimal solutions to this problem instance (some tasks can have different start times that lead to the minimized objective).
(B) There exists a polynomial time (to m and n) algorithm which solves the problem optimally.
(C) $X + Y + Z = 12$.
(D) $Y^2 + Z^2 < 50$.
(E) $X^2 + 2Y + 4Z > 30$.

試題隨卷繳回