Department of Computer Science and Engineering
National Sun Yat-sen University
First Semester of 2026 PhD Qualifying Exam

Subject： Computer Architecture

1. (20% total) Assume a 32-bit, byte-addressed machine with virtual addressing. The two high-order bits of 11 are treated as **unmapped**. These addresses are only accessible by the operating system and bypass virtual address translation. The answers must be expressed as a multiple of a power of 2 or in terms of KB, MB, GB, or TB, as appropriate.

1.1 (4%) What is the maximum amount of **physical memory** this system can address?
1.2 (4%) What is the maximum amount of **virtual memory** any single process on this system can address?
1.3 (4%) How many **virtual pages** are available to each process, assuming 4KB pages?
1.4 (4%) Assume each page table entry is 4 bytes; how much memory would a single-level page table require?
1.5 (4%). Assuming a two-level page table where half the hits of the virtual page number are used to index the first level, and the other half are used to index the second level, how many second-level page tables can a process use if its total page table size is limited to 400KB?

2. (20% total) The instruction latency for a given CPU is shown in Table 1.

Table 1.

Instructions	Breakdown	Latency
load	5%	3 cycles
add	10%	5 cycles
divide	10%	8 cycles
branch	50%	2 cycles
shift left	15%	5 cycles
shift right	10%	1 cycles

2.1 (5%) Calculate the **CPI** of the given CPU?
2.2 (5%) **Variant 1:** All add instructions are replaced with corresponding subtract instructions that take the same amount of time. Calculate the **CPI** of **Variant 1**?
2.3 (5%) **Variant 2:** Remove the highest-latency instruction, and replace all those instructions with additional latency of three cycles for the lowest latency instruction in the mix. Calculate the **CPI** of **Variant 2**?
2.4 (5%) **Variant 3:** Reduce the latency for divides by a factor of four, but increase the latencies of branches by 50%. Calculate the **CPI** of **Variant 3**?

3. (20% total) An old program needs to be parallelized. Then, it can run faster on modern multicore processors. To execute the program with parallel and serial portions more efficiently, a custom heterogeneous processor needs to be designed.

- The processor has one large core, which executes code more quickly but occupies a greater area on the die, and multiple small cores, which execute code more slowly but consume less area, all sharing the same processor die.
- When the program is in its parallel portion, all of its threads execute only on small cores.
- When the program is in its serial portion, the active thread executes on the large core.
- The performance (execution speed) of a core is proportional to the square root of its area.
- Assume 16 units of die area available. A small core takes 1 unit of die area. The large core can accommodate any number of units of die area, n^2, where n is a positive integer. Area not used by the large core will be filled with smaller cores.
- The **serial portion** accounts for only 10% of the total work, while the parallel portion comprises the remaining **90%**.

3.1 (10%) What would be the speed up for the fastest possible execution of the program?

3.2 (10%) What would the same program's speedup be if all 16 units of die area were used to build a homogeneous system with 16 small cores, the serial portion ran on one of the small cores, and the parallel portion ran on all 16 small cores?

4.(20% total) Please answer the following questions.
4.1 (5%) Give two disadvantages of static code scheduling.
4.2 (5%) Give two disadvantages of dynamic code scheduling.
4.3 (5%) What are two mechanisms that dynamic code scheduling uses to mitigate the shortcomings of static code scheduling, and why does each mechanism help?
4.4 (5%) What types of dependencies can occur during out-of-order execution?

5. (20% total) Processor microarchitecture: VLIW vs. Superscalar vs. Array Processor
5.1 (4%) What do VLIW, superscalar execution, and array processing concepts have in common?
5.2 (6%) Provide two reasons why a VLIW microarchitecture is simpler than a "same-width" superscalar microarchitecture.
5.3 (4%) Provide a reason why a superscalar microarchitecture could provide higher performance than a "same-width" VLIW microarchitecture.
5.4 (3%) Provide a reason why a VLIW processor is more flexible than an array processor.
5.5 (3%) Provide a reason why an array processor is simpler than a "same-width" VLIW processor.