1.

Assume a GPU architecture that contains **10 SIMD processors**. Each SIMD instruction **has a width of 32**, and each SIMD processor contains **8 lanes** for single-precision arithmetic and load/store instructions. This means that each non-diverged SIMD instruction **can produce 32 results every 4 cycles**. Assume a kernel that has divergent branches causing, on average, **80% of threads** to be active. Assume that **70%** of all SIMD instructions executed are single-precision arithmetic, and **20%** are load/store. Because not all memory latencies are covered, assume an average SIMD instruction issue rate of **0.85**. Assume that the GPU has a clock speed of **1.5 GHz**.

a.

Compute the throughput, in GFLOP/s, for this kernel on this GPU.

b.

Assume that you have the following choices:

1. Increasing the number of single-precision lanes to **16**.
2. Increasing the number of SIMD processors to **15** (assume this change doesn't affect any other performance metrics and that the code scales to the additional processors).
3. Adding a cache that will effectively reduce memory latency by **40%**, which will increase the instruction issue rate to **0.95**.

What is the speedup in throughput for each of these improvements?

2.

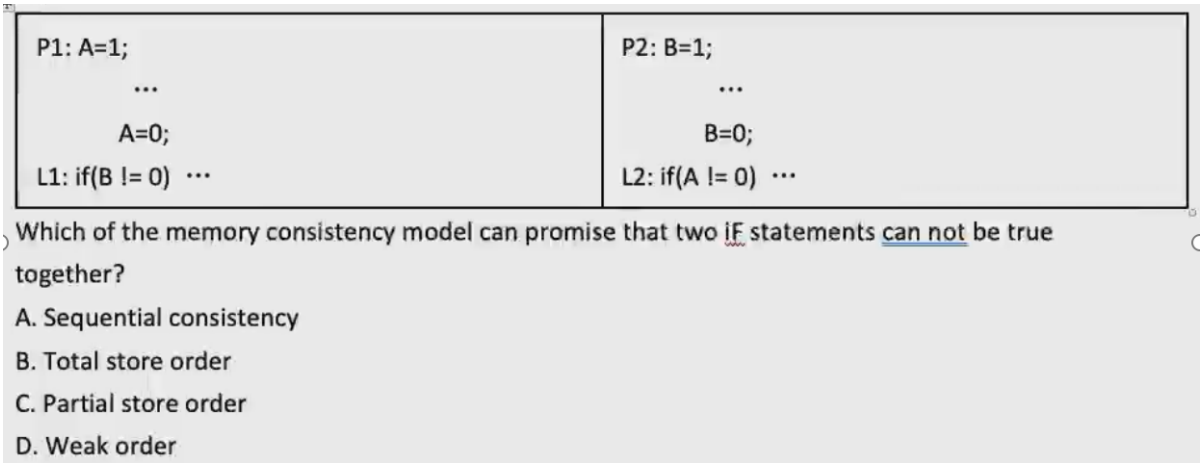
```
for (int i = 0; i < 524; i++) {
    c[i] = a[i] + b[i];
}
```

Consider the code above, arrays **a**, **b**, and **c** each have **524 8-byte-wide integer elements**.
Assume that:

- A **64-bit-wide integer add instruction** takes **1 clock cycle**.
- A **512-bit-wide vector add instruction** takes **4 clock cycles**.

Question:
How many clock cycles do we need to finish the above calculation using **512-bit-wide vector add instructions**?

3.

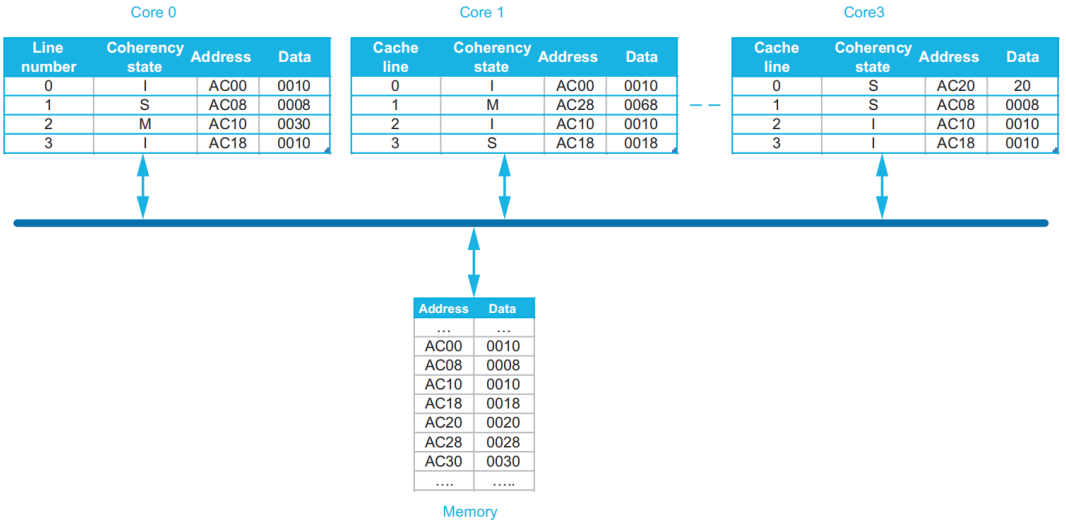


Which technique is not serving for data-level **parallelism** (DLP)?
A. SIMD
B. Dynamic Scheduling
C. Vector process
D. GPU

Which of the following techniques may get **CPI (cycle per instruction)** smaller than 1?
A. Superscalar
B. Dynamic Scheduling
C. Software pipelining
D. Speculation based on Tomasulo

4.

5.1. [10/10/10/10/10/10/10] <5.2> For each part of this exercise, the initial cache and memory state are assumed to initially have the contents shown in Figure 5.37. Each part of this exercise specifies a sequence of one or more CPU operations of the form



`Ccore#: R, <address>` for reads

and

`Ccore#: W, <address> <-- <value written>` for writes.

For example,

`C3: R, AC10 & C0: W, AC18 <-- 0018`

Read and write operations are for 1 byte at a time. Show the resulting state (i.e., coherence state, tags, and data) of the caches and memory after the actions given below. Show only the cache lines that experience some state change; for example:

`C0.L0: (I, AC20, 0001)` indicates that line 0 in core 0 assumes an "invalid" coherence state (`I`), stores `AC20` from the memory, and has data contents `0001`. Furthermore, represent any changes to the memory state as `M: <address> <- value`.

Different parts (a) through (g) do not depend on one another: assume the actions in all parts are applied to the initial cache and memory states.

a. [10] <5.2> `C0: R, AC20`

b. [10] <5.2> `C0: W, AC20 <-- 80`

c. [10] <5.2> `C3: W, AC20 <-- 80`

d. [10] <5.2> `C1: R, AC10`

e. [10] <5.2> `C0: W, AC08 <-- 48`

f. [10] <5.2> `C0: W, AC30 <-- 78`

5.

Consider the following code, which multiplies two vectors that contain single-precision complex values:

```
for (i = 0; i < 300; i++) {
    c_re[i] = a_re[i] * b_re[i] - a_im[i] * b_im[i];
    c_im[i] = a_re[i] * b_im[i] + a_im[i] * b_re[i];
}
```

Assume that the processor runs at 700 MHz and has a maximum vector length of 64.
The load/store unit has a start-up overhead of 15 cycles; the multiply unit, 8 cycles; and the add/subtract unit, 5 cycles.

a. Assuming chaining and a single memory pipeline, how many chimes are required?
How many clock cycles are required per complex result value, including start-up overhead?

b. Now assume that the processor has three memory pipelines and chaining. If there are no bank conflicts in the loop's accesses, how many clock cycles are required per result?