8 Vector Processing [40 points]

Assume a vector processor that implements the following ISA:

Opcode	Operands	Latency (cycles)	Description
SET	V_{st}, #n	1	$V_{st} \leftarrow$ n (V_{st} = Vector Stride Register)
SET	V_{ln}, #n	1	$V_{ln} \leftarrow$ n (V_{ln} = Vector Length Register)
VLD	V_i, #A	100, pipelined	$V_i \leftarrow Mem[Address]$
VST	V_i, #A	100, pipelined	$Mem[Address] \leftarrow V_i$
VMUL	V_i, V_j, V_k	10, pipelined	$V_i \leftarrow V_j * V_k$
VADD	V_i, V_j, V_k	5, pipelined	$V_i \leftarrow V_j + V_k$
VDIV	V_i, V_j, V_k	20, pipelined	$V_i \leftarrow V_j / V_k$

Assume the following:

- The processor has an in-order pipeline.
- The size of a vector element is 4 bytes.
- V_{st} and V_{ln} are 10-bit registers.
- The processor *does not* support chaining between vector functional units.
- The main memory has N banks.
- Vector elements stored in consecutive memory addresses are interleaved between the memory banks. E.g., if a vector element at address A maps to bank B, a vector element at address $A + 4$ maps to bank $(B + 1)\%N$, where % is the modulo operator and N is the number of banks. N is *not necessarily* a power of two.
- The memory is byte addressable and the address space is represented using 32 bits.
- Vector elements are stored in memory in 4-byte-aligned manner.
- Each memory bank has a 4 KB row buffer.
- Each memory bank has a single read and a single write port so that a load and a store operation can be performed simultaneously.
- There are separate functional units for executing VLD and VST instructions.

(a) [5 points] What should the minimum value of N be to avoid stalls while executing a VLD or VST instruction, assuming a vector stride of 1? Explain.

100 banks (because the latency of VLD and VST instructions is 100 cycles)

(b) [8 points] What should the minimum value of N be to avoid stalls while executing a VLD or VST instruction, assuming a vector stride of 2? Explain.

101 banks.
Explanation. To avoid stalls, we need to ensure that consecutive vector elements access 100 different banks.
With a vector stride of 2, consecutive elements of a vector will map to every other bank. For example, if the first element maps to bank 0, the next element will map to bank 2, and so on.
With 100 banks, the 51st element of a vector will map to bank $100\%100 = 0$, conflicting with the first element of the vector.
Howevert, with 101 banks, the 51st element will map to bank 1, which was skipped by the previous vector elements.

Let's assume there are 102 elements in a vector and the first elements accesses bank 0. The 101 banks will be accessed in the following order:

$(0, 2, ..., 100, 102, 104, ..., 200, 202)\%101 = (0, 2, ..., 100, 1, 3, ..., 99, 0)$
We can see that none of the elements conflict in the DRAM banks. Note that, when the last vector elements accesses bank 0, the bank is already available for a new access because the 100 cycle latency of accessing the first element is overlapped by accessing the other 101 elements.

(c) [12 points] Assume:

- A machine that has a memory with as many banks as you found is part (a).
- The vector stride is set to 1.
- The value of the vector length is set to M (but we do *not* know M)

The machine executes the following program:

```
VLD V1 ← A
VLD V2 ← (A + 32768)
VADD V3 ← V1, V1
VMUL V4 ← V2, V3
VST (A + 32768*2) ← V4
```

The total number of cycles needed to complete the execution of the above program is 4306. What is M?

$M = 1000$

Explanation.

```
VLD  |-100-|--(M-1)--|
VLD                  |-100-|--(M-1)--|
VADD                  |-5-|--(M-1)--|
VMUL                                  |-10-|--(M-1)--|
VST                                                     |-100-|--(M-1)--|
```

$(M + 100 - 1) + 100 + (M - 1) + 10 + (M - 1) + 100 + (M - 1)$
$= 306 + 4 * M = 4306 \rightarrow M = 1000$ elements

(d) [15 points] If we modify the vector processor to *support chaining*, how many cycles would be required to execute the same program in part (c)? Explain.

```
VLD  |--100--|--(VLEN-1)--|
VLD                       |---100---|---(VLEN-1)---|
VADD                      |-1-|-5-|---(VLEN-1)---| (this is delayed because the processor
                                                   executes the instructions in order)
VMUL                                |-10-|---(VLEN-1)---|
VST                                      |-100-|---(VLEN-1)---|

    100 + (VLEN-1) + 100 + 10 + 100 + (VLEN-1) = 310 + 2*1000 - 2 = 2308 cycles
```