Q1

A **CPU** has a 32 KB direct mapped cache with 128 byte-block size. Suppose A is two dimensional array of size 512×512 with elements that occupy $8 - bytes$ each. Consider the following two C code segments, $P1$ and $P2$.

$P1$:

```
for (i=0; i<512; i++)
{
    for (j=0; j<512; j++)
    {
        x +=A[i] [j];
    }
}
```

P2:

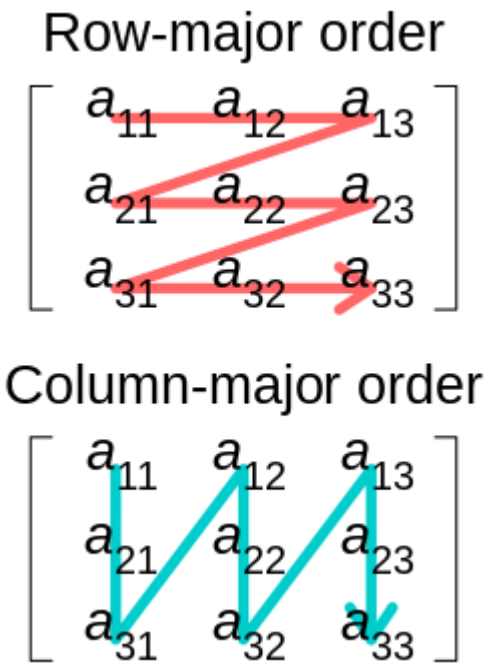
```
for (i=0; i<512; i++)
{
    for (j=0; j<512; j++)
    {
        x +=A[j] [i];
    }
}
```

$P1$ and $P2$ are executed independently with the same initial state, namely, the array A is not in the cache and i, j, x are in registers. Let the number of cache misses experienced by $P1$ be $M1$ and that for $P2$ be $M2$.

The value of the ratio $\frac{M_1}{M_2}$:

Kindly Note that C stores in Row Major Order.

The difference between the orders lies in which elements of an array are contiguous in memory. In row-major order, the consecutive elements of a row reside next to each other



Q2

In a C program, an array is declared as float $A[2048]$. Each array element is 4 Bytes in size, and the starting address of the array is $0x00000000$. This program is run on a computer that has a direct mapped data cache of size 8 Kbytes, with block (line) size of 16 Bytes.

A. Which elements of the array conflict with element $A[0]$ in the data cache? Justify your answer briefly.
B. If the program accesses the elements of this array one by one in reverse order i.e., starting with the last element and ending with the first element, how many data cache misses would occur? Justify your answer briefly. Assume that the data cache is initially empty and that no other data or instruction accesses are to be considered.

Q3

Consider a $2-$way set associative cache with 256 blocks and uses LRU replacement. Initially the cache is empty. Conflict misses are those misses which occur due to the contention of multiple blocks for the same cache set. Compulsory misses occur due to first time access to the block. The following sequence of access to memory blocks :

$$\{0, 128, 256, 128, 0, 128, 256, 128, 1, 129, 257, 129, 1, 129, 257, 129\}$$

is repeated 10 times. The number of *conflict misses* experienced by the cache is _____ .

Questions on Pipeline SPLIT Phase isnt considered

Q4

Instruction execution in a processor is divided into 5 stages, *Instruction Fetch* (IF), *Instruction Decode* (ID), *Operand fetch* (OF), *Execute* (EX), and *Write Back* (WB). These stages take **5, 4, 20, 10** and **3 nanoseconds (ns)** respectively. A pipelined implementation of the processor requires buffering between each pair of consecutive stages with a delay of **2 ns**. Two pipelined implementation of the processor are contemplated:

i. a naive pipeline implementation (NP) with 5 stages and
ii. an efficient pipeline (EP) where the OF stage is divided into stages OF1 and OF2 with execution times of **12 ns** and **8 ns** respectively.

The speedup (correct to two decimal places) achieved by EP over NP in executing 20 independent instructions with no hazards is _____ .

Q5

A pipelined processor uses a 4-stage instruction pipeline with the following stages: Instruction fetch (IF), Instruction decode (ID), Execute (EX) and Writeback (WB). The arithmetic operations as well as the load and store operations are carried out in the EX stage. The sequence of instructions corresponding to the statement $X = (S - R * (P + Q))/T$ is given below. The values of variables P, Q, R, S and T are available in the registers $R0, R1, R2, R3$ and $R4$ respectively, before the execution of the instruction sequence.

ADD	$R5, R0, R1$	; $R5 \leftarrow$ R0 + R1
MUL	$R6, R2, R5$	; $R6 \leftarrow$ R2 * R5
SUB	$R5, R3, R6$	; $R5 \leftarrow$ R3 -R6
DIV	$R6, R5, R4$	; $R6 \leftarrow$ R5/R4
STORE	$R6, X$	; $X \leftarrow$ R6

The IF, ID and WB stages take 1 clock cycle each. The EX stage takes 1 clock cycle each for the ADD, SUB and STORE operations, and 3 clock cycles each for MUL and DIV operations. Operand forwarding from the EX stage to the ID stage is used. The number of clock cycles required to complete the sequence of instructions is

A. 10
B. 12
C. 14
D. 16

Q6

Consider the given C-code and its corresponding assembly code, with a few operands U1-U4 being unknown. Some useful information as well as the semantics of each unique assembly instruction is annotated as inline comments in the code. The memory is byte-addressable.

```
//C-code
int a[10], b[10], i;
// int is 32-bit
for (i=0; i<10;i++)
    a[i] = b[i] * 8;
```

```
;assembly-code (; indicates comments)
;r1-r5 are 32-bit integer registers
;initialize r1=0, r2=10
;initialize r3, r4 with base address of a, b

 L01: jeq r1, r2, end  ;if(r1==r2) goto end
 L02: lw  r5, 0(r4)    ;r5 <- Memory[r4+0]
 L03: shl r5, r5, U1   ;r5 <- r5 << U1
 L04: sw  r5, 0(r3)    ;Memory[r3+0] <- r5
 L05: add r3, r3, U2   ;r3 <- r3+U2
 L06: add r4, r4, U3
 L07: add r1, r1, 1
 L08: jmp U4           ;goto U4
 L09: end
```

Find Value of U1,U2,U3,U4
PS
U1.U2,U3 Would be Integer
U4 Would be Label
