

Question 2 (20 points):

1. (5 points) How many bits are used for tag, index, and offset in a 128-KB cache that is 8-way set associative with 64-byte block used in a processor with 32-bit address?

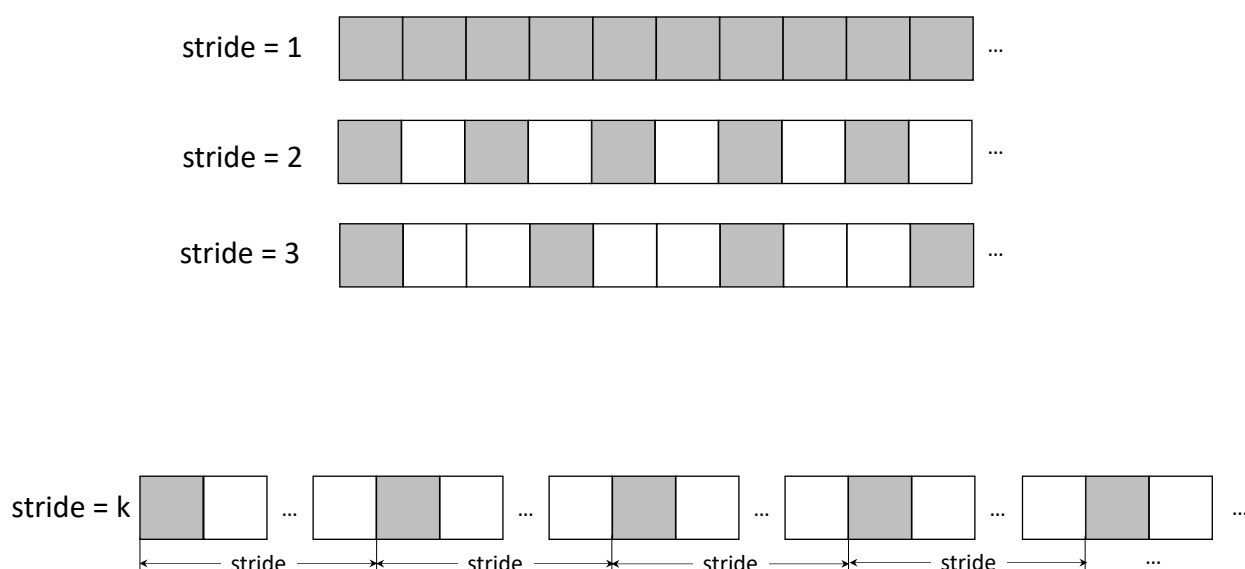


Figure 1: Illustration of strided access into a vector.

2. (5 points) A strided access into a vector consists in a program regularly skipping over elements of the vector. In the illustration shown in Figure ??, each small square represents a 32-bit integer value. A dark square represents a vector element that is accessed by a program while a white box is a vector element that is not accessed. Figure ?? illustrates accesses into a vector with different strides. Figure ?? shows the C code and the RISC-V code for the function `StridedDotProduct`. We are interested in the accesses to the data cache that occur inside the loop body. The loop index `i` and the variable `Sum` are kept in registers. Assume that in a given execution of this function, `A = 0x40000000` and `B = 0x60000000`, and `N` is very large. We will study the memory hierarchy for executions of `StridedDotProduct` with different values of `stride`. In the table below write the memory addresses, expressed in

```

2 StridedDotProduct:
3     bgtz    a3, doSum    # if N>0 goto doSum
4     mv      a0, zero     # return Sum=0
5     jalr    zero,ra,0
6 doSum:
7     mv      t0, zero     # Sum <- 0
8     mv      t1, zero     # i <- 0
9 next:
10    sll      t2, t1, 2    # t2 <- 4*i
11    add      t3, a0, t2   # t3 <- Addr(A[i])
12    lw       t4, 0(t3)    # t4 <- A[i]
13    add      t5, a1, t2   # t5 <- Addr(B[i])
14    lw       t6, 0(t5)    # t6 <- B[i]
15    mul      t7, t5, t6   # t7 <- A[i]*B[i]
16    add      t1, t1, a2   # i <- i + stride
17    blt      t1, a3, next # if i<N goto next
18    jalr     zero,ra,0

```

(a) C code

```

2 StridedDotProduct:
3     bgtz    a3, doSum    # if N>0 goto doSum
4     mv      a0, zero     # return Sum=0
5     jalr    zero,ra,0
6 doSum:
7     mv      t0, zero     # Sum <- 0
8     mv      t1, zero     # i <- 0
9 next:
10    sll      t2, t1, 4    # t2 <- 4*i
11    add      t3, a0, t2   # t3 <- Addr(A[i])
12    lw       t4, 0(t3)    # t4 <- A[i]
13    add      t5, a1, t2   # t5 <- Addr(B[i])
14    lw       t6, 0(t5)    # t6 <- B[i]
15    mul      t7, t5, t6   # t7 <- A[i]*B[i]
16    add      t1, t1, a2   # i <- i + stride
17    blt      t1, a3, next # if i<N goto next
18    jalr     zero,ra,0

```

(b) RISC-V code

Figure 2: Code for StridedDotProduct

hexadecimal, for the first eight data accesses in **StridedDotProduct** for each execution of the function with the value of **stride** specified at the top of the column.

Data Access	stride=1	stride=2	stride=4
1			
2			
3			
4			
5			
6			
7			
8			

3. (5 points) Assume that the machine where `StridedDotProduct` is executed has a 32KB data cache with 16-byte blocks. In the table below write the hit ratio for the data accesses while executing `StridedDotProduct` for the specified cache organization.

Cache Organization	stride=1	stride=2	stride=4
Direct Mapped			
Two-Way Set Associative			

4. (5 points) How would the answers to item (c) above change if the size of the cache was changed to 64KB for both possible cache organizations. Explain your answer in terms of spatial and temporal locality.