سوال دوم:

Cache size: 32 bytes
Block size: 4 bytes
Associativity: 2-way set-associative

- **Capacity (C):**
  - number of data bytes in cache
- **Block size (b):**
  - bytes of data brought into cache at once
- **Number of blocks ($B = C/b$):**
  - number of blocks in cache: $B = C/b$
- **Degree of associativity (N):**
  - number of blocks in a set
- **Number of sets ($S = B/N$):**
  - each memory address maps to exactly one cache set

الف) تعداد کل بلوک های یک cache حاصل تقسیم ظرفیت بر اندازه هر بلوک است.

B = C/b → B = 32 / 4 = 8.

ب) تعداد ست ها برابر با تعداد بلوک ها تقسیم بر درجه associativity که برای این cache برابر ۲ است.

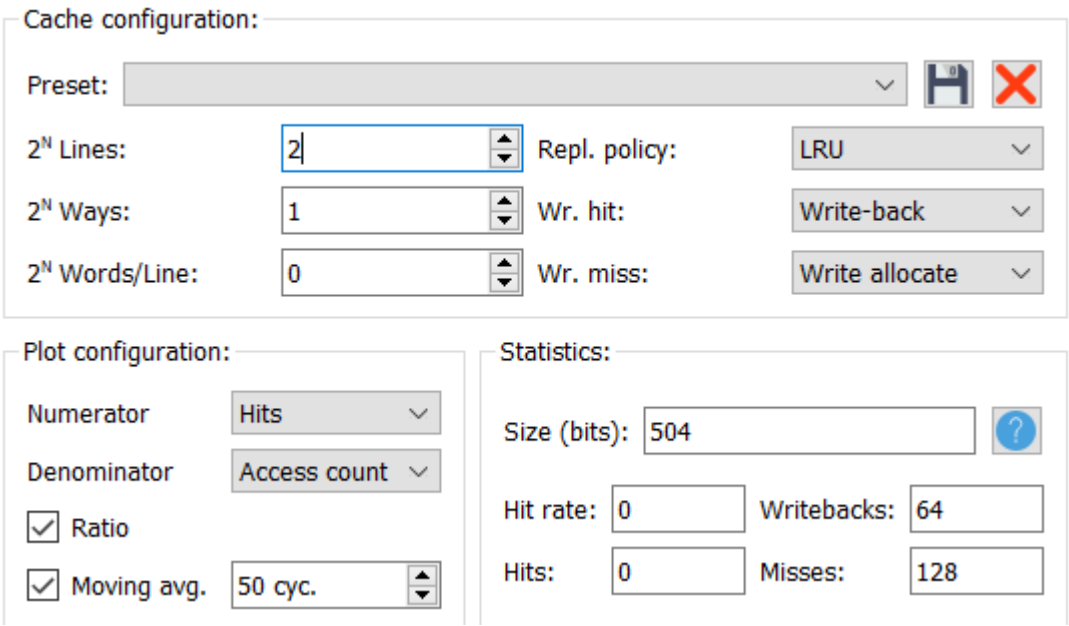
S = B/N → S = 8 / 2 = 4.

پس ۴ مجموعه (set) داریم که در هر کدام ۲ بلوک است. در هر way یک بلوک.

ج) چون تعداد ست های ما ۴ تاست پس به lg(4) = 2 بیت index نیاز خواهیم داشت . دو بیت نیز offset هست که در نتیجه 32-(2+2) = 28 بیت برای tag باقی میماند.

د) در حلقه دوم که عملیات lw انجام میشود ما در هر 64 ایتریشن miss خواهیم داشت چون هر دفعه یک آدرس جدید آورده میشود و در cache قرار داده میشود و هیچوقت دوباره استفاده نمیشود و طبیعتا miss داریم . و چون در هر way تنها یک بلوک جا دارد از spatial locality هم استفاده نمیشود .

100 % miss rate, 0% hit rate, 64 times.



```
addi s0, zero, 8 # s0 = SIZE = 8
addi t0, zero, 0 # t0 = i = 0
addi t1, zero, 0 # t1 = j = 0
addi t2, zero, 0 # t2 = sum = 0
add s1, zero, gp # s1 = head of array
for_i:
    beq t0, s0, continue
    addi t0, t0, 1 # i++
    add t1, zero, zero # j = 0
    for_j:
        beq t1, s0, for_i
        add t3, t0, t1 # t3 = i+j
        addi t4, t3, 1 # t4 = i+j+1
        sw t4, 0(s1) # array[i][j] = i+j+1
        addi s1, s1, 4 # به صورت خطی ذخیره میشه تو موری در نهایت
        addi t1, t1, 1 # j++
        j for_j
    j for_i
continue:
    add s1, zero, gp # reset the pointer to head of array
    addi t0, zero, 0 # reset i and j
    addi t1, zero, 0
    for_i_2:
    beq t0, s0, end
    addi t0, t0, 1 # i++
    add t1, zero, zero # j = 0
    for_j_2:
        beq t1, s0, for_i_2
        lw t5, 0(s1) # t5 =  array[i][j]
        addi s1, s1, 4 # مثل حلقه قبل
        add t2, t2, t5 # sum = sum + t5
        addi t1, t1, 1 # j++
        j for_j_2
    j for_i_2
end:
    addi zero, zero, 0
```