# Animation of Algorithm

**Goal:**

To understand an algorithm by animating its execution, step-by-step.

**Algorithm:** Array-Sum (find sum of n numbers)

**Observe** carefully:

sequential instruction

conditional statements,

repetitive statements,

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

Let's ***animate*** the execution of this algorithm on the input:

**Input to Algo. Array-Sum:**

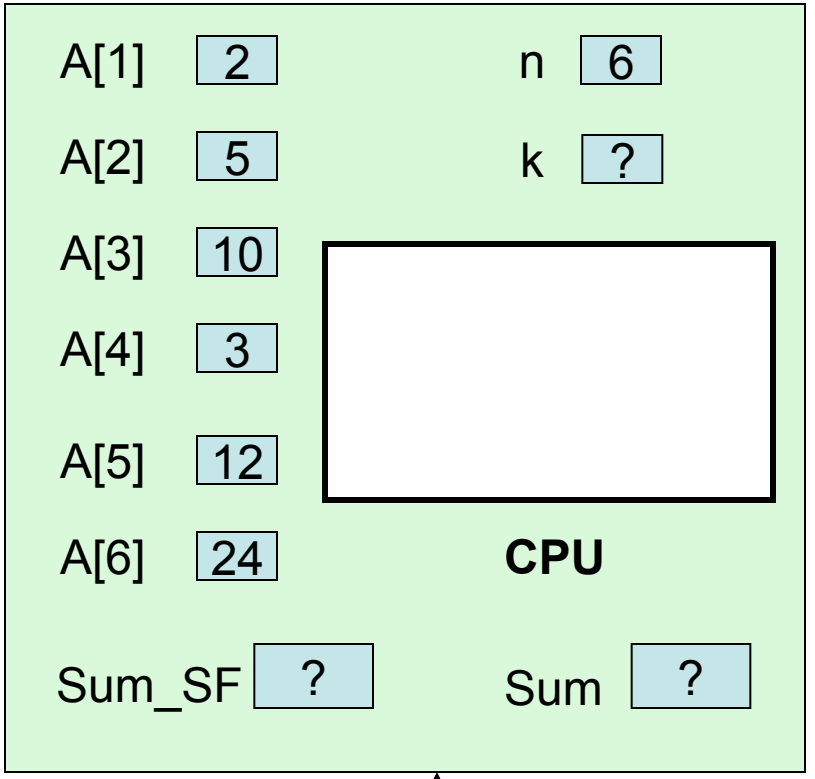
$n = 6$

Array $A$ = [2, 5, 10, 3, 12, 24]

# Model of Computer: Initial State

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

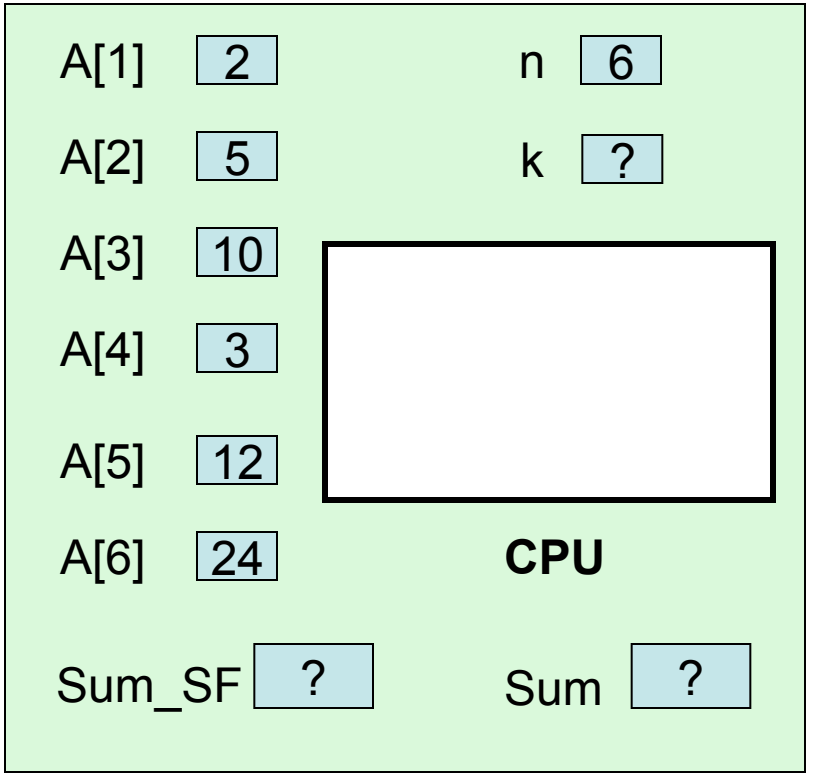



We assume that the values of n and the array A[1..6] has been read into the computer.

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

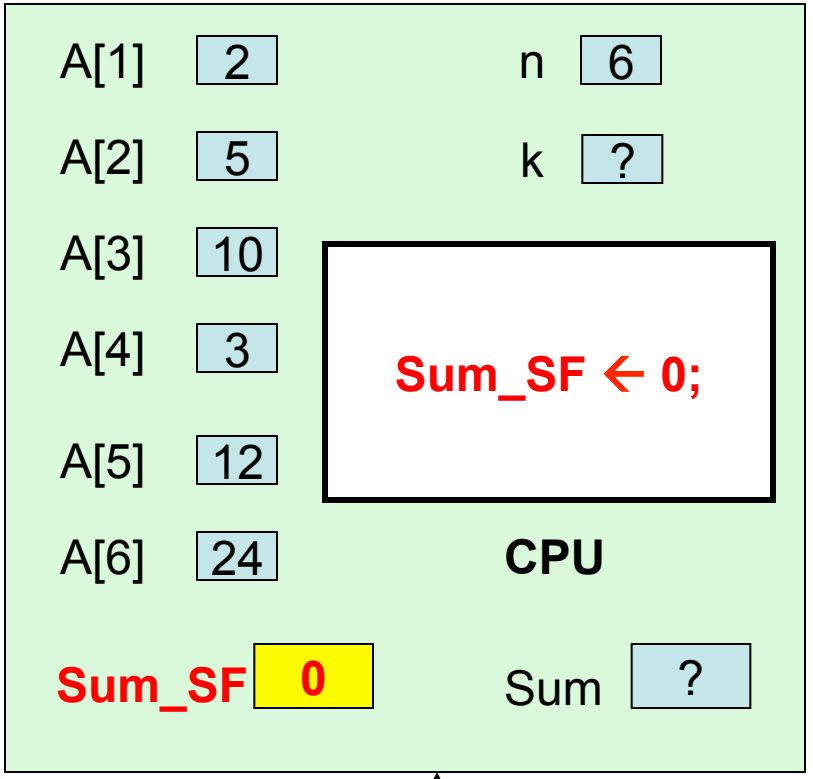
***Start of execution…***



## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

*executing…*

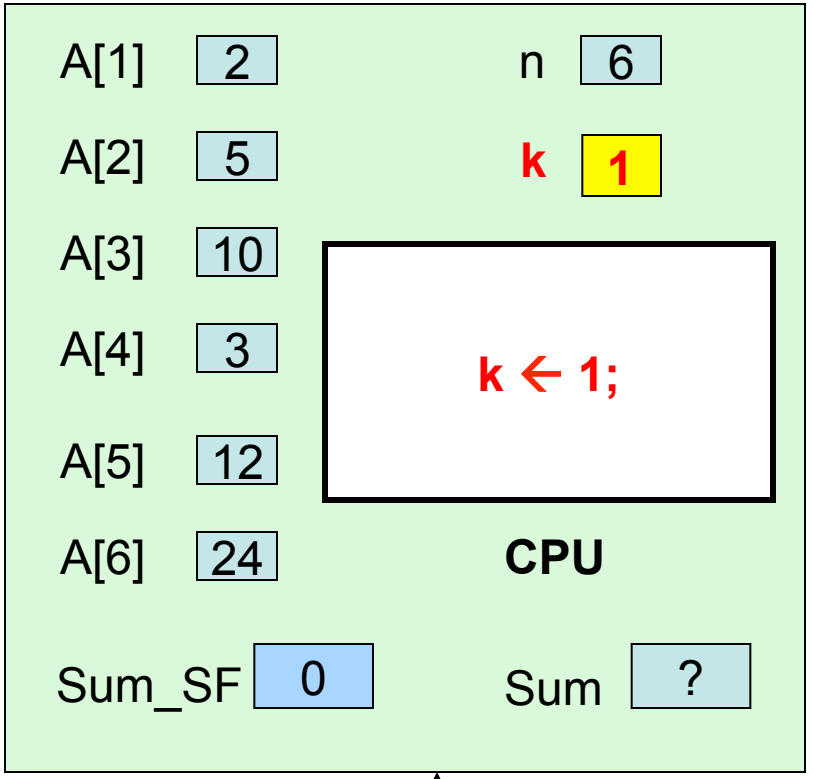



**Assignment statement;**
**The value of 0 is stored in the storage box called Sum_sf.**

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

*executing…*

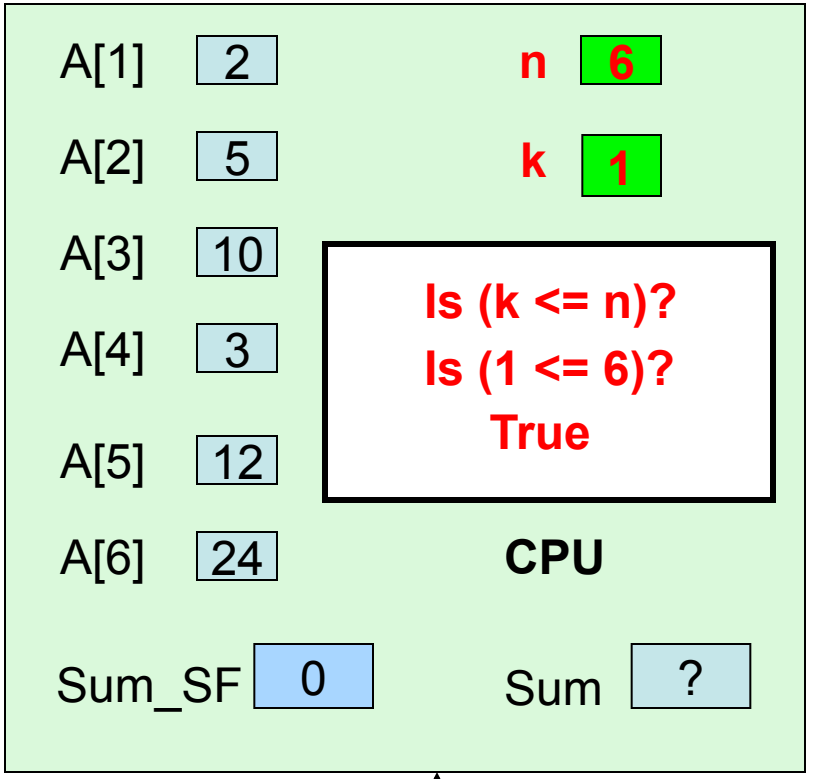



**Assignment statement;**
**The value of 1 is stored in the storage box called k.**

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

*executing beginning of loop*

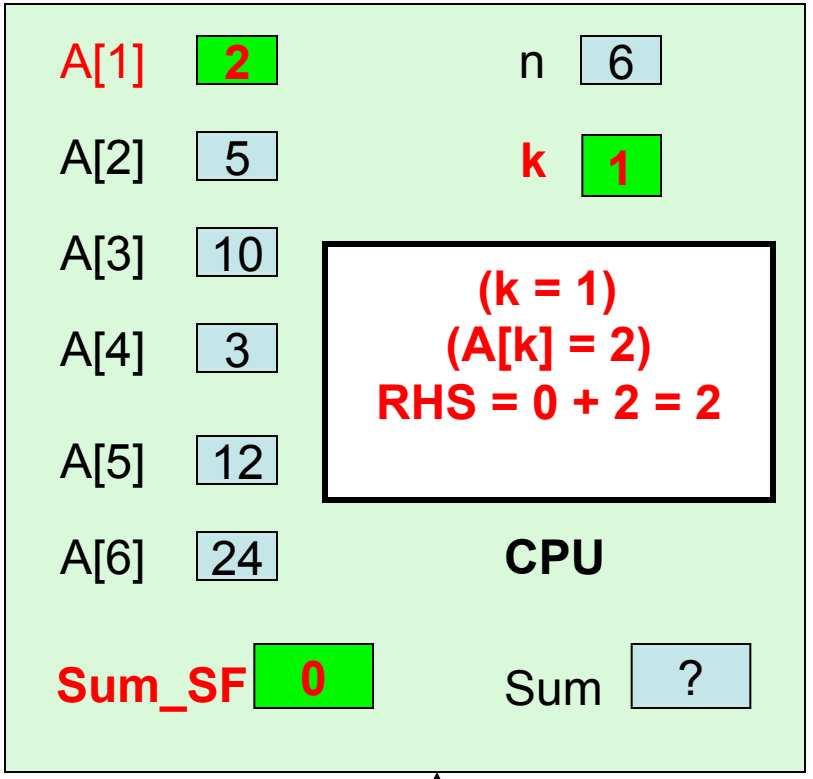



**Conditional: (k <= n)**
**Reads value of k and n;**
**Check outcome of (k <= n),**
**Condition is True, execute loop;**
**Value of k, n are unchanged.**

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

*inside body of loop*

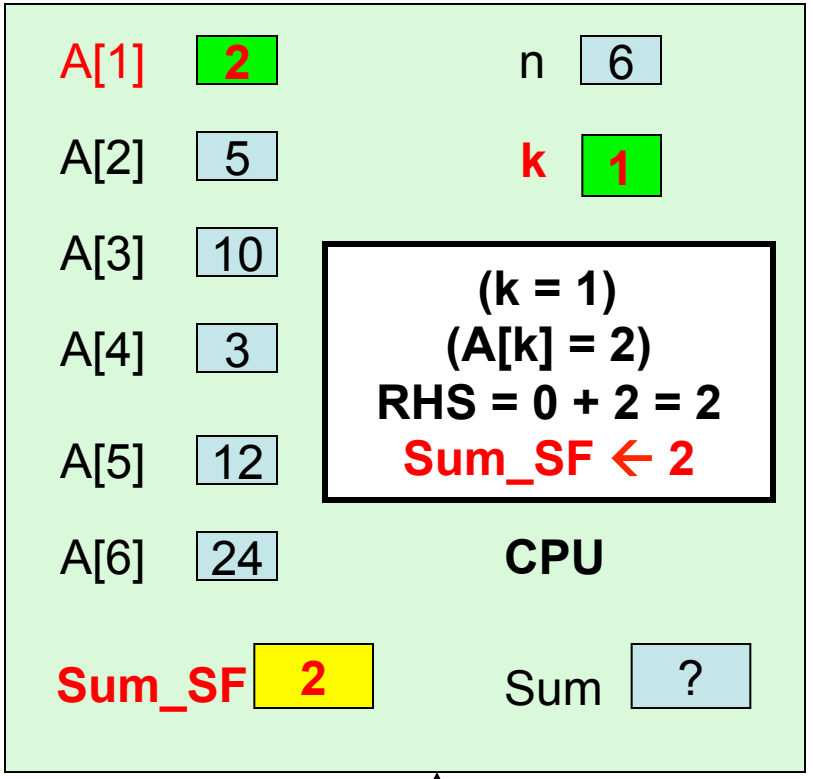



**Calculate value of expression on the right-hand-side;**
**Note: In A[k], k is used as an index (subscript) to the array A**

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

*inside body of loop*

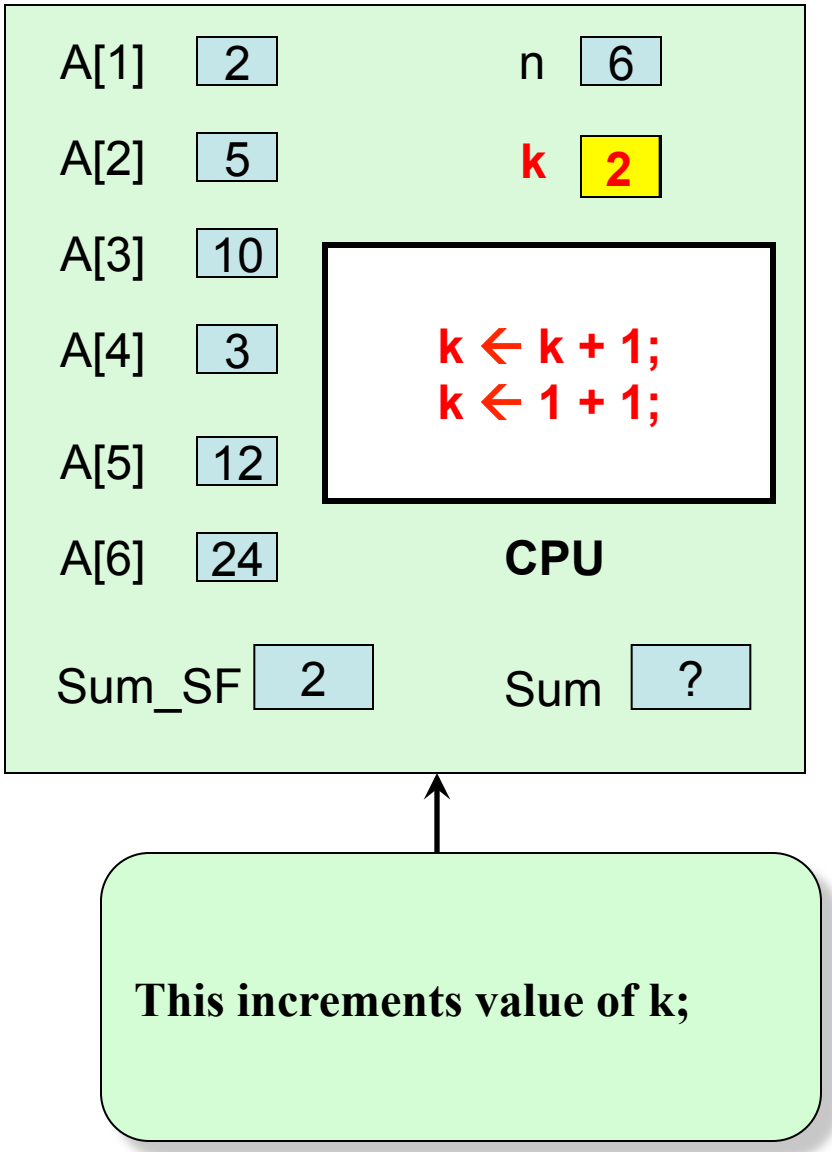



**Calculate value of expression on the right-hand-side;**
**Assign result to variable on left;**

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

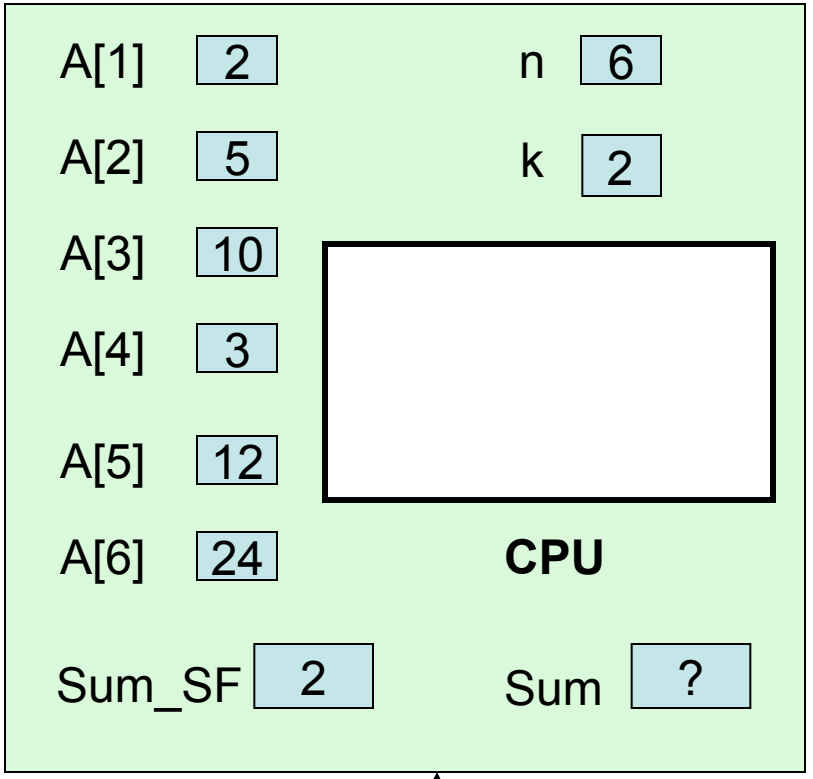
*inside body of loop*



## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

*End of body of loop, loop back...*

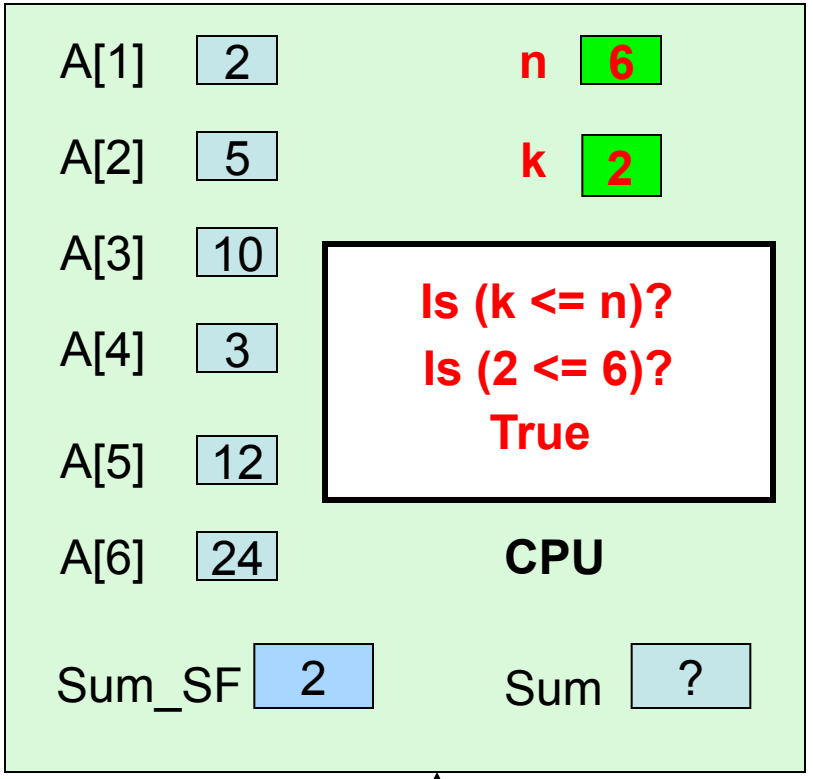



**Reached end of the loop body.**
**Must loop back and re-test the loop condition.**
**(We will skip this in future loops)**

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

## *Test loop condition again…*

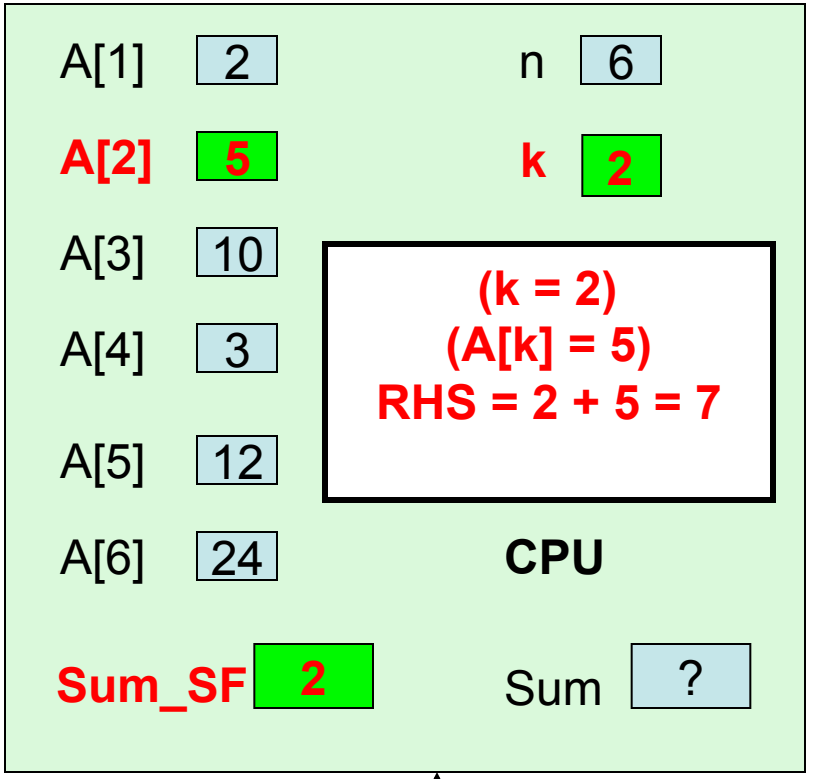



**Testing the loop condition again with new value of k.**
**(This is call a pre-test loop, where testing is done before entering the loop.)**

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

***inside body of loop***

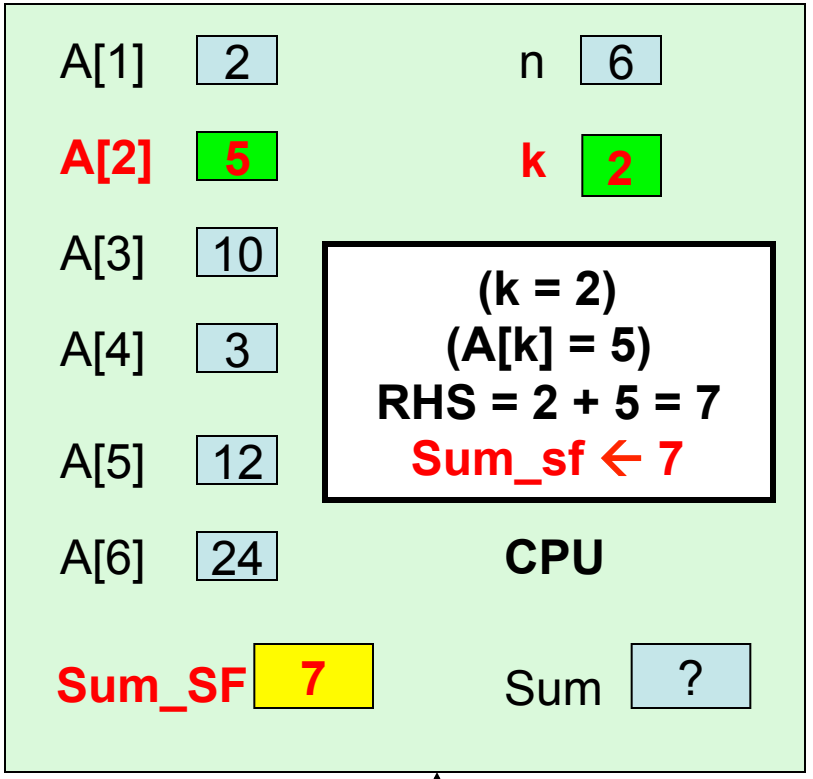



**Notice that A[k] now gets the second number in the list;**

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

*inside body of loop*

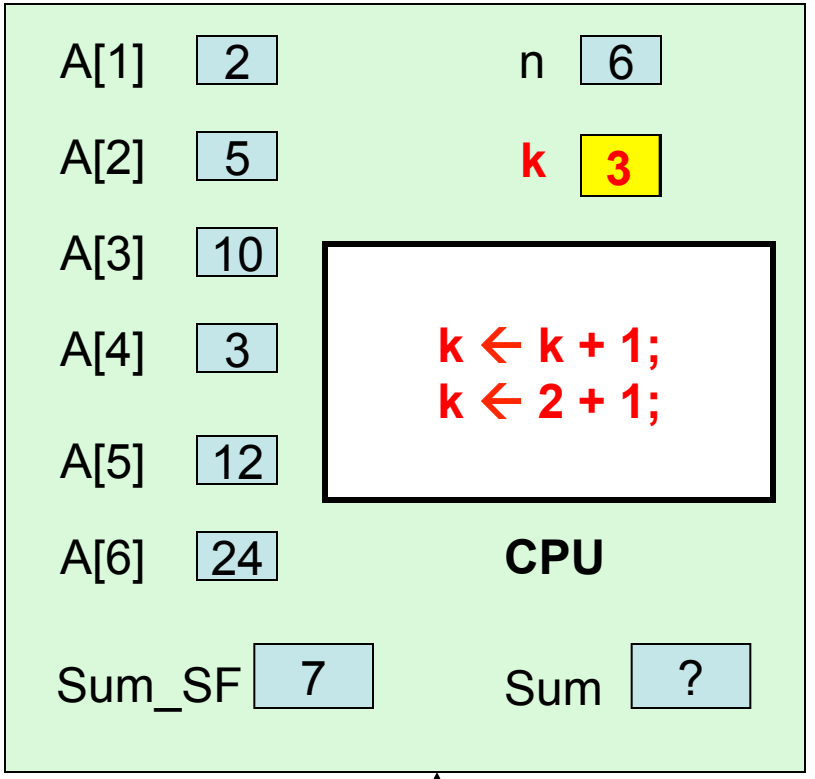



Notice that A[k] now gets the second number in the list; This number (5) is added to Sum_SF

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

***inside body of loop***

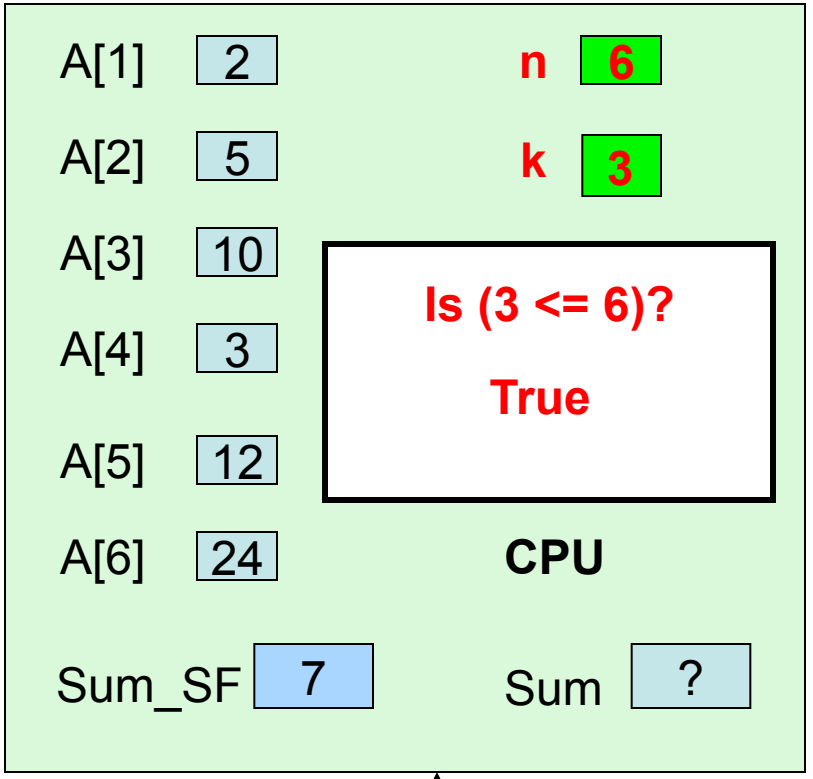



**This increments value of k;**

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

## *Test loop condition again…*

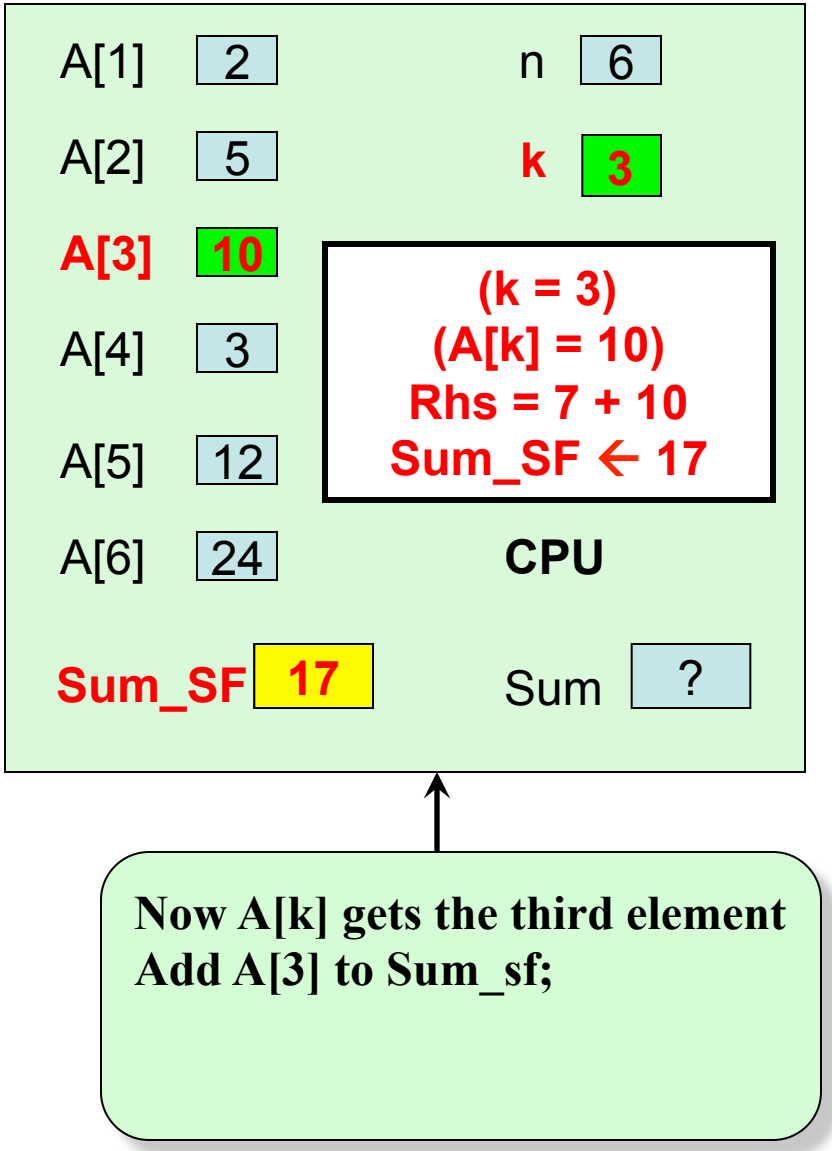



**Testing the loop condition again with new value of k.**

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

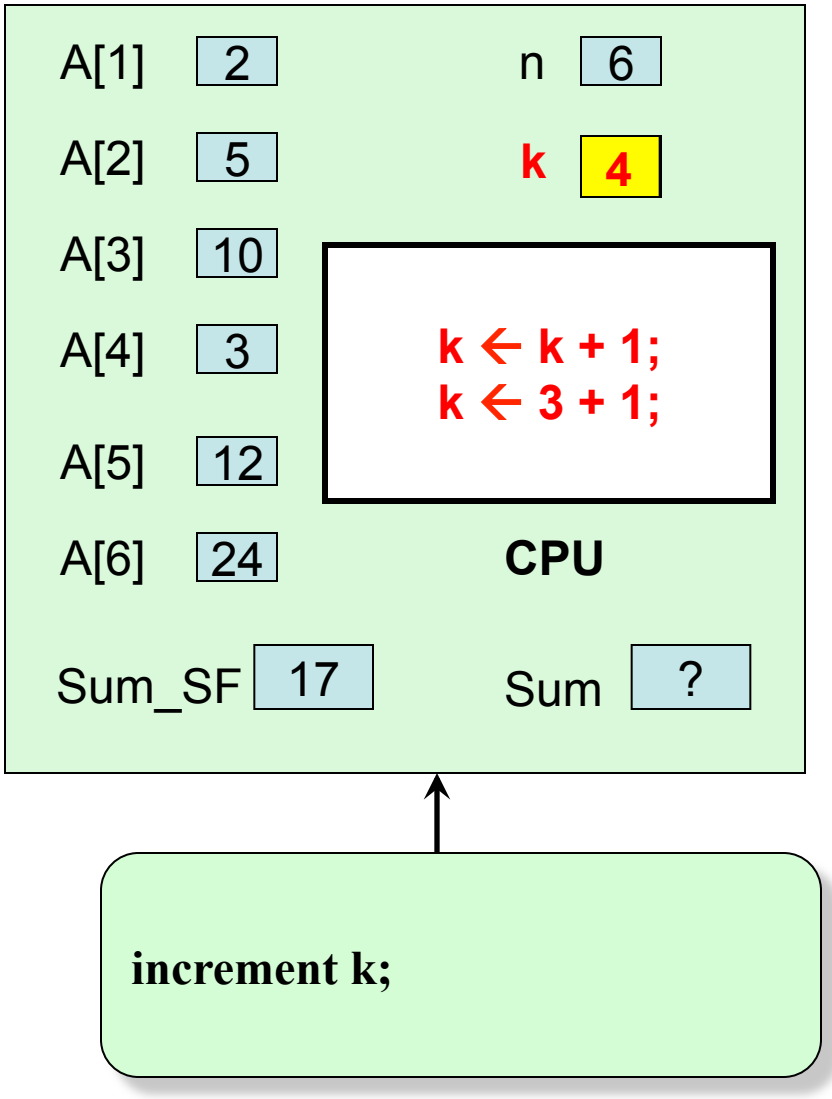
*inside body of loop*



## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

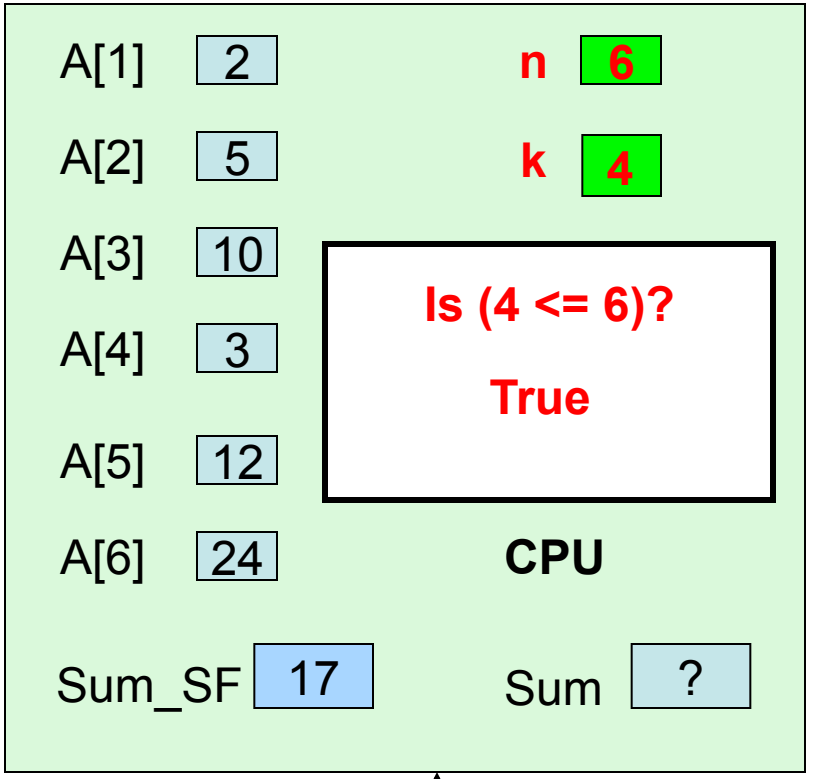
*inside body of loop*



## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

## *Test loop condition again…*

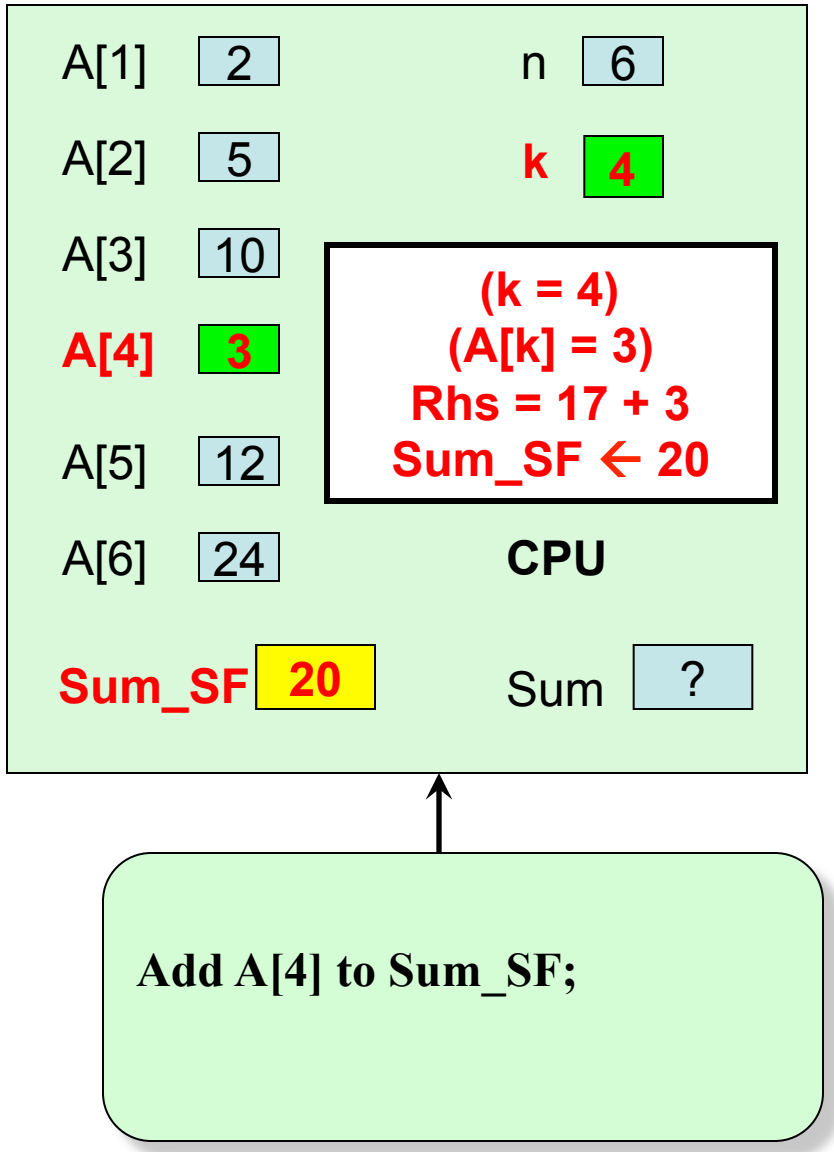



**Testing the while loop condition again with new value of k.**

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

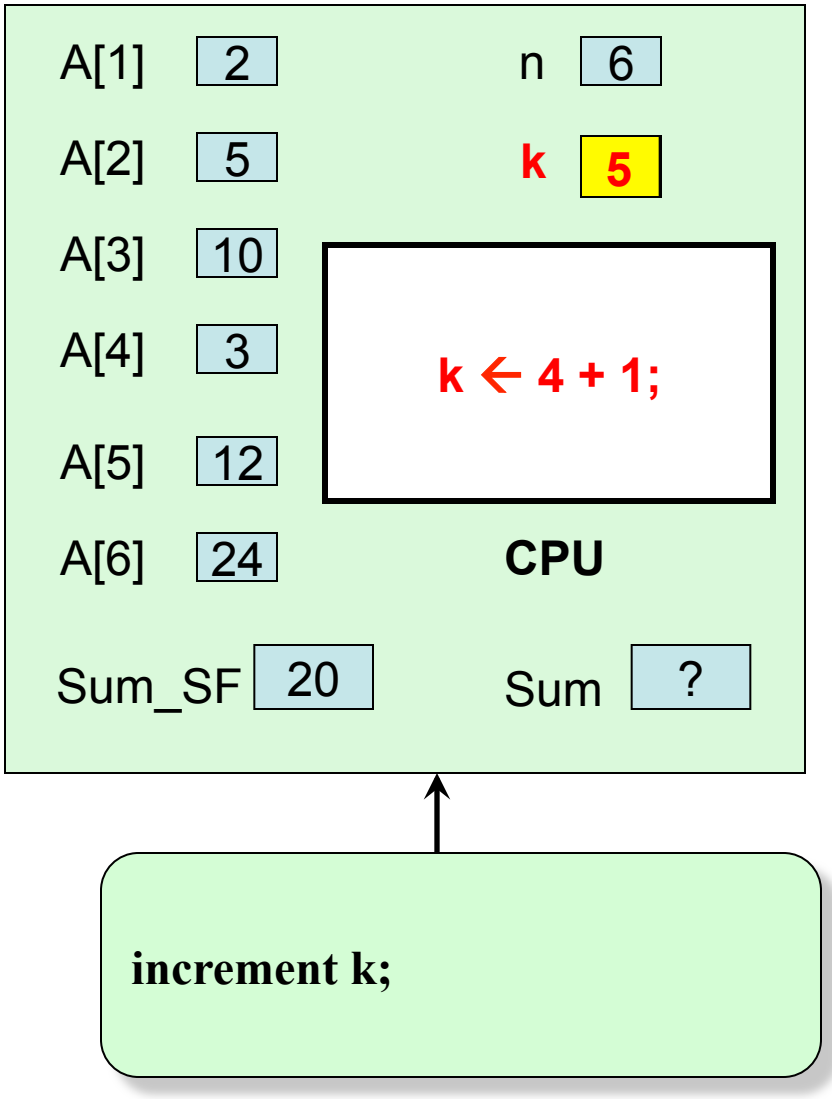
*inside body of loop*



## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

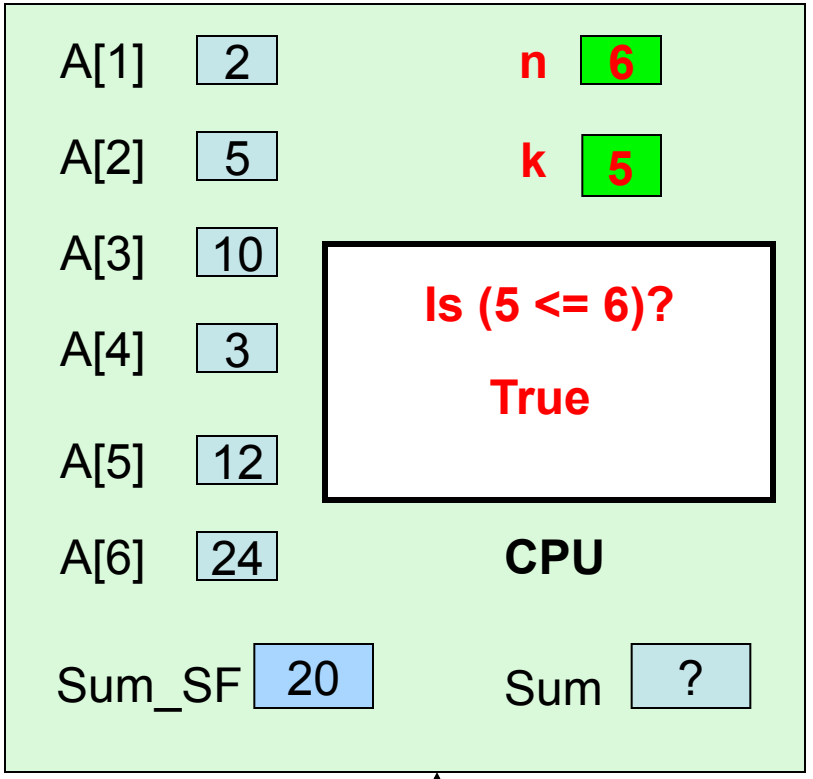
*inside body of loop*



## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

## *Test loop condition again…*

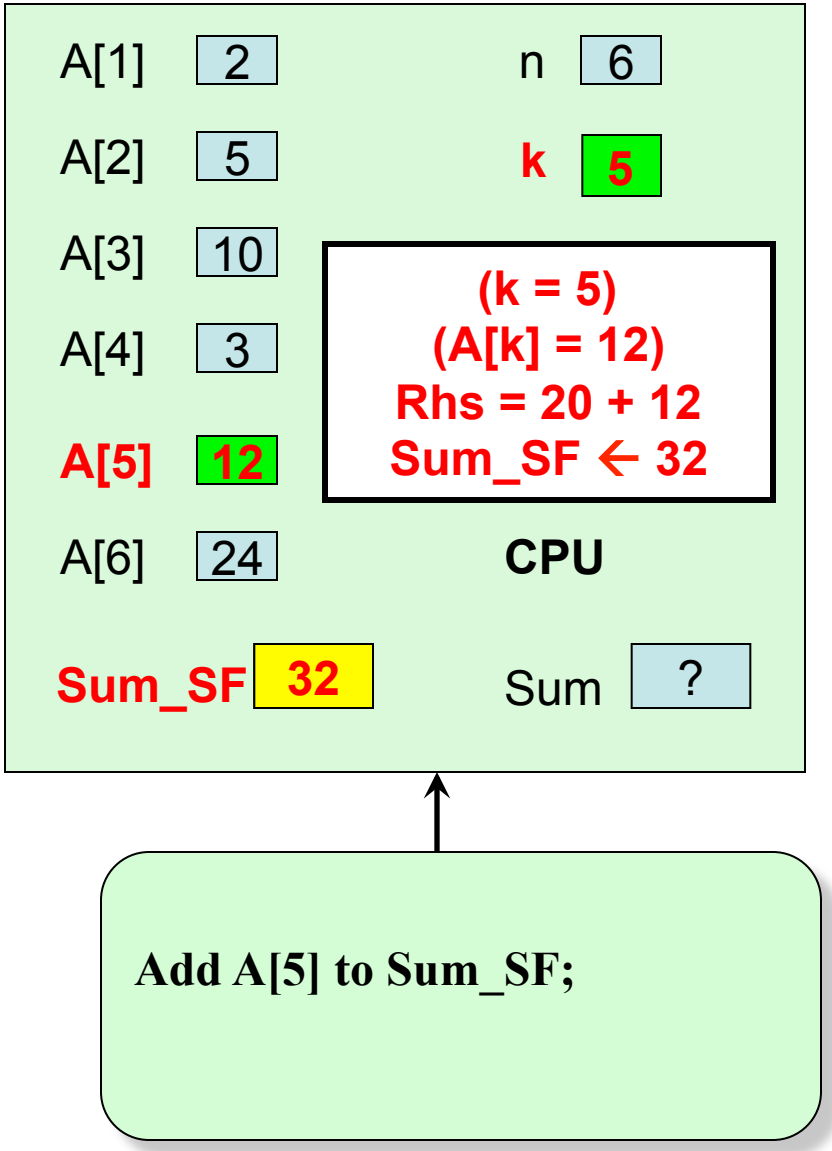



**Testing the while loop condition again with new value of k.**

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

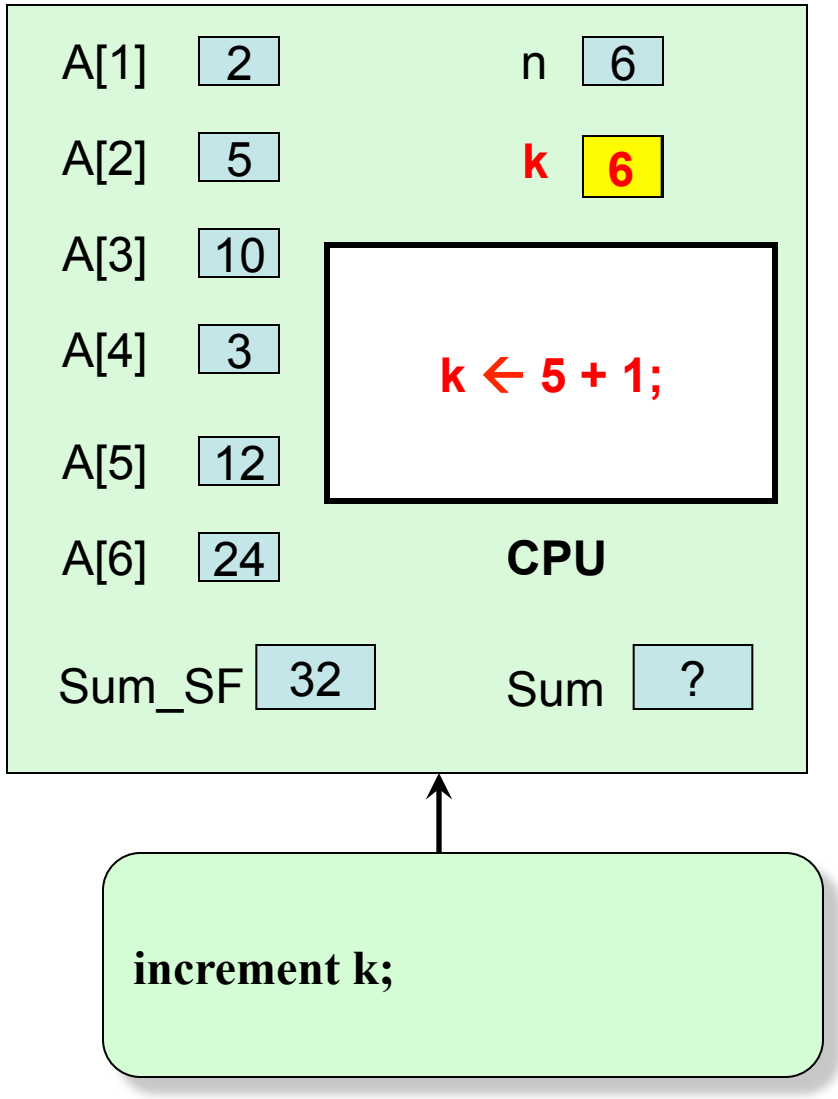
*inside body of loop*



## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

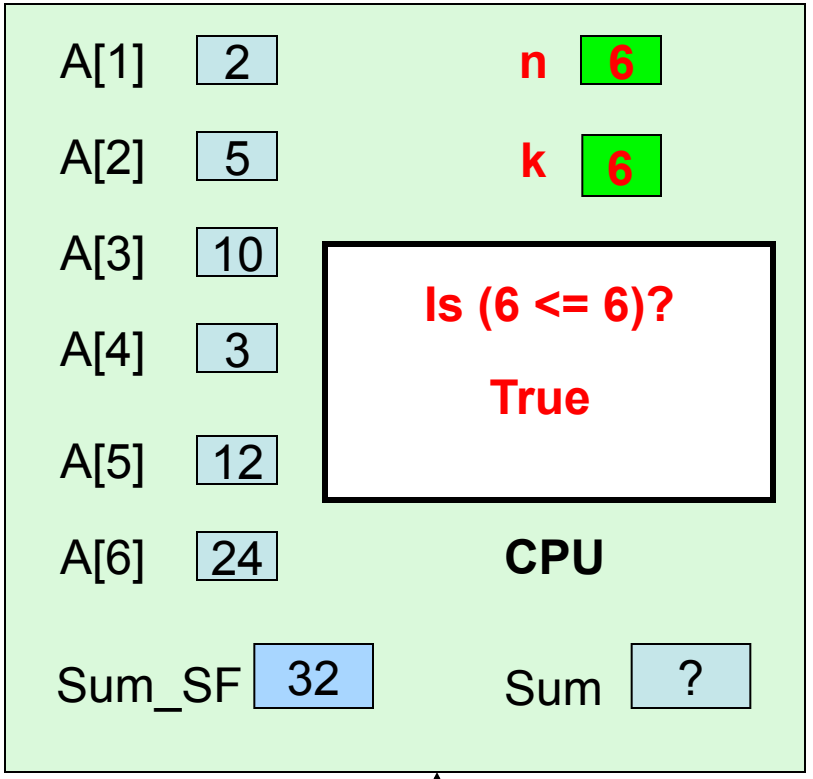
*inside body of loop*



## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

## *Test loop condition again…*

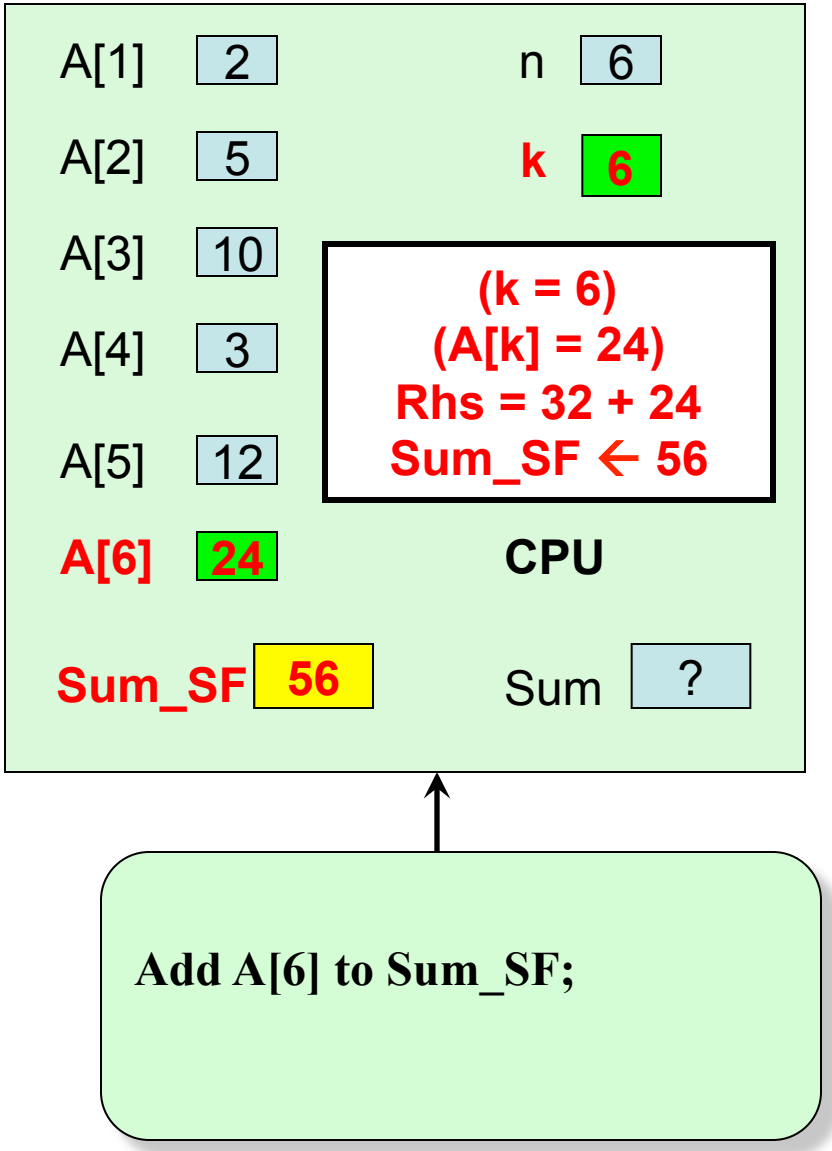



**Testing the while loop condition again with new value of k.**

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

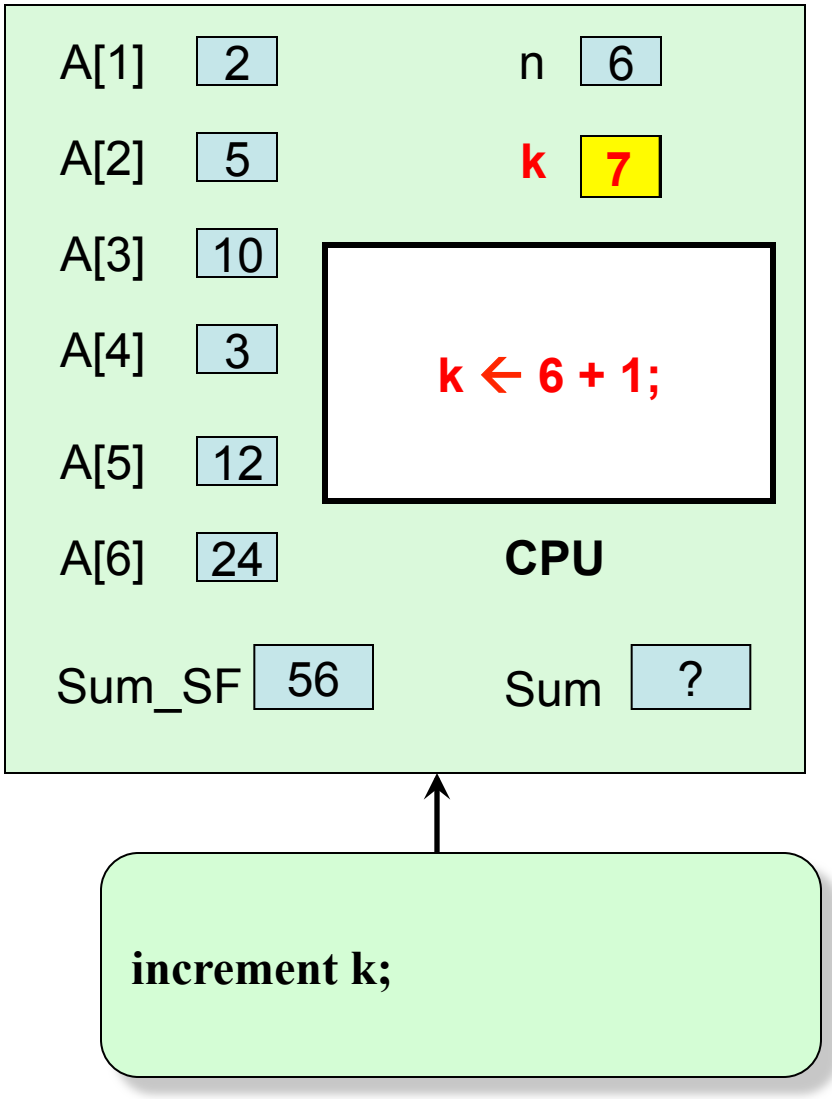
*inside body of loop*



## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

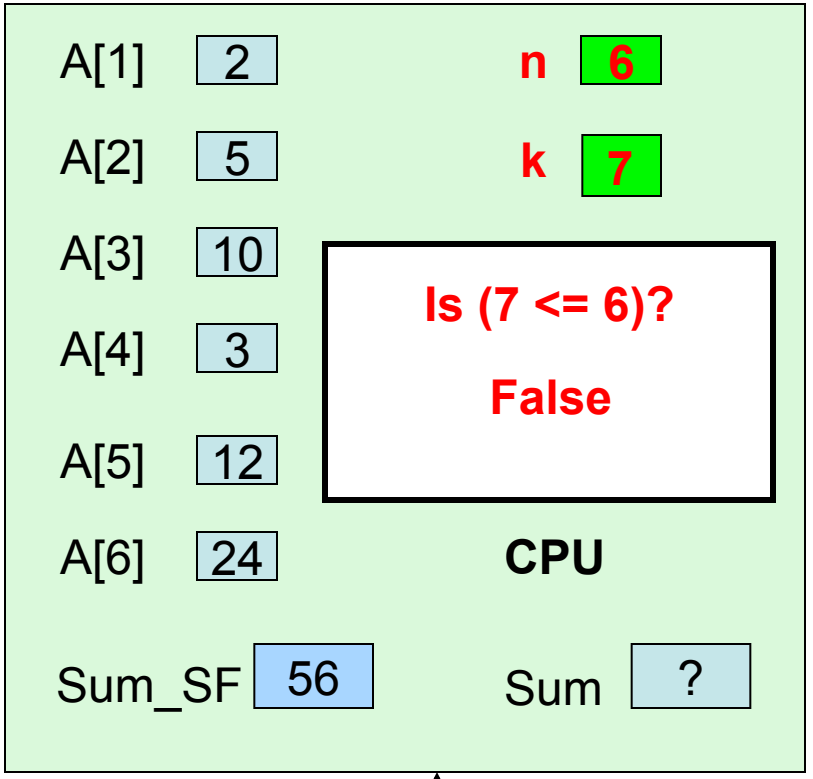
*inside body of loop*



## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

## *Test loop condition again…*

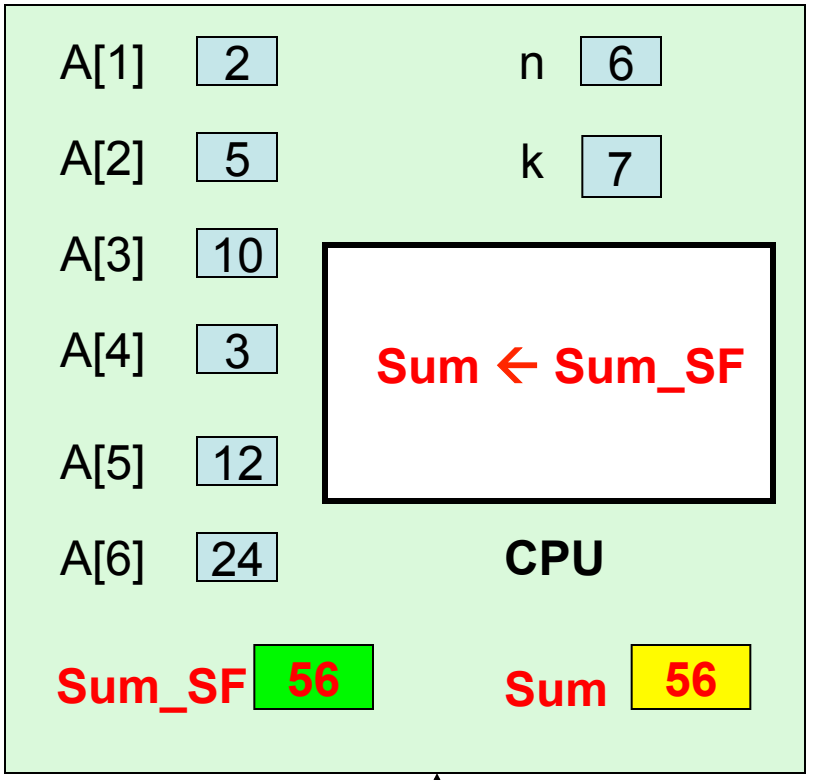



**The loop condition fails (is false);**
**Skip the body of loop.**
**Go to *the statement after* the while loop;**

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

***Jump out of loop, to next stmt…***

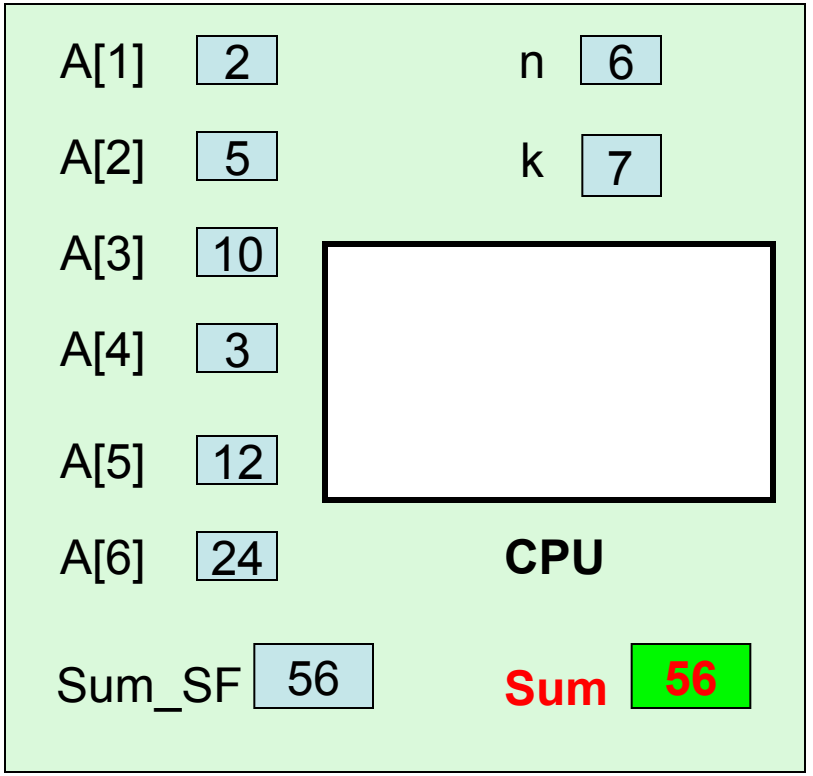



Copy value of Sum_SF to Sum

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

## *Print statement…*



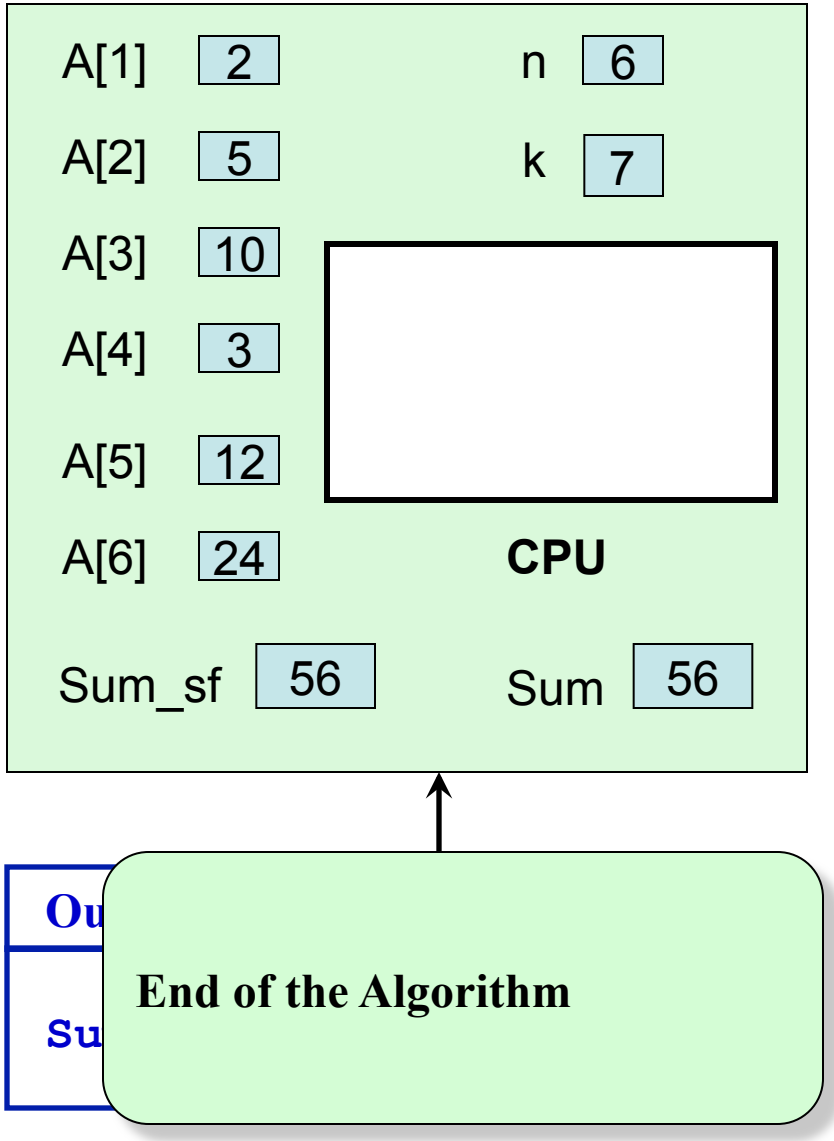

**Output of Algrithm Sum:**

```
Sum is 56
```

# End of Algorithm…

## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```



## Algorithm Array-Sum

```
Array-Sum(A, n);
begin
    Sum_SF ← 0;
    k ← 1;
    while (k <= n) do
        Sum_SF ← Sum_SF + A[k];
        k ← k + 1;
    endwhile
    Sum ← Sum_SF;
    Print "Sum is", Sum;
end;
```

**Your Homework:**

On your own, execute algorithm Sum on this input.

**Input to Algorithm Sum:**

$n = 4$

Array $A$ = [4, 7, 3, 9, 2, 20, 10]

# Summary

## Review:

A. Did you understand the following concepts:

1. variables (storage boxes)
2. arrays (a collection of contiguous variables)
3. array index or subscript

B. Can you follow the execution of
sequential instruction,
the loop pre-test,
repetitive statements?

# Self Assessment

**Self Test 1:** (based on the example)

1. How many times was the loop executed?
2. How many times was the pre-loop test executed?
3. What is the value of *k* at the end of the algorithm?

**Self Test 2:** (for a general value of *n*)

1. How many times was the loop executed?
2. How many times was the pre-loop test executed?
3. What is the value of *k* at the end of the algorithm?

**Self Test 3:** (Modifying the algorithm *slightly* to …)

1. compute the average of the *n* numbers in array *A*.
2. the "sum-of-squares" of the *n* numbers.