

# 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  $\leftarrow$  0;  
    k  $\leftarrow$  1;  
    while (k  $\leq$  n) do  
        Sum_SF  $\leftarrow$  Sum_SF + A[k];  
        k  $\leftarrow$  k + 1;  
    endwhile  
    Sum  $\leftarrow$  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  $A = [2, 5, 10, 3, 12, 24]$

## Algorithm Array-Sum

**Array-Sum(A, n);**

**begin**

**Sum\_SF  $\leftarrow$  0;**

**k  $\leftarrow$  1;**

**while (k  $\leq$  n) do**

**Sum\_SF  $\leftarrow$  Sum\_SF + A[k];**

**k  $\leftarrow$  k + 1;**

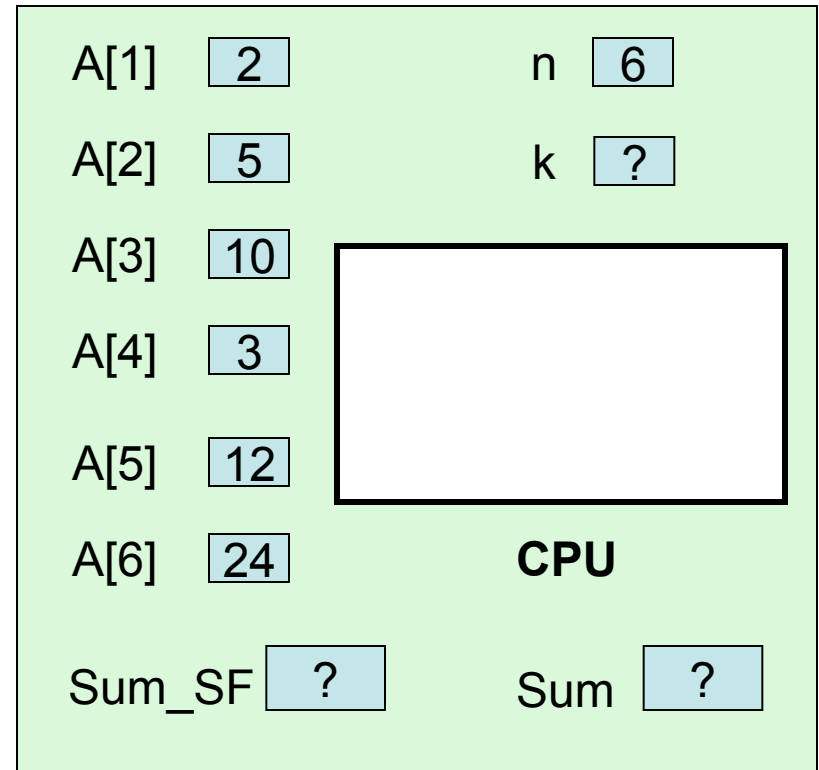
**endwhile**

**Sum  $\leftarrow$  Sum\_SF;**

**Print "Sum is", Sum;**

**end;**

## *Model of Computer: Initial State*



**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  $\leftarrow$  0;**

**k  $\leftarrow$  1;**

**while (k  $\leq$  n) do**

**Sum\_SF  $\leftarrow$  Sum\_SF + A[k];**

**k  $\leftarrow$  k + 1;**

**endwhile**

**Sum  $\leftarrow$  Sum\_SF;**

**Print "Sum is", Sum;**

**end;**

*Start of execution...*

A[1] 2

n 6

A[2] 5

k ?

A[3] 10

A[4] 3

A[5] 12

A[6] 24

Sum\_SF ?

**CPU**

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;**

*executing...*

A[1] 2

n 6

A[2] 5

k ?

A[3] 10

A[4] 3

A[5] 12

A[6] 24

**Sum\_SF ← 0;**

**CPU**

**Sum\_SF** 0

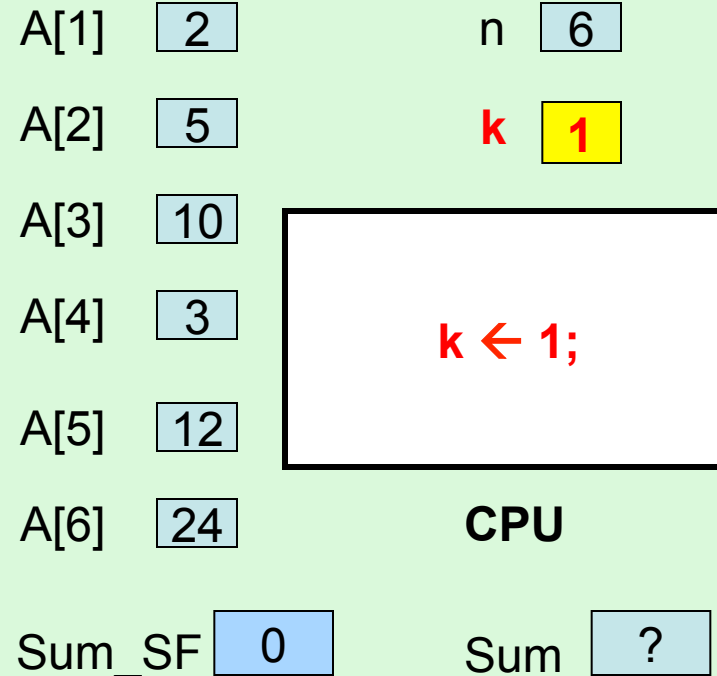
Sum ?

**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*

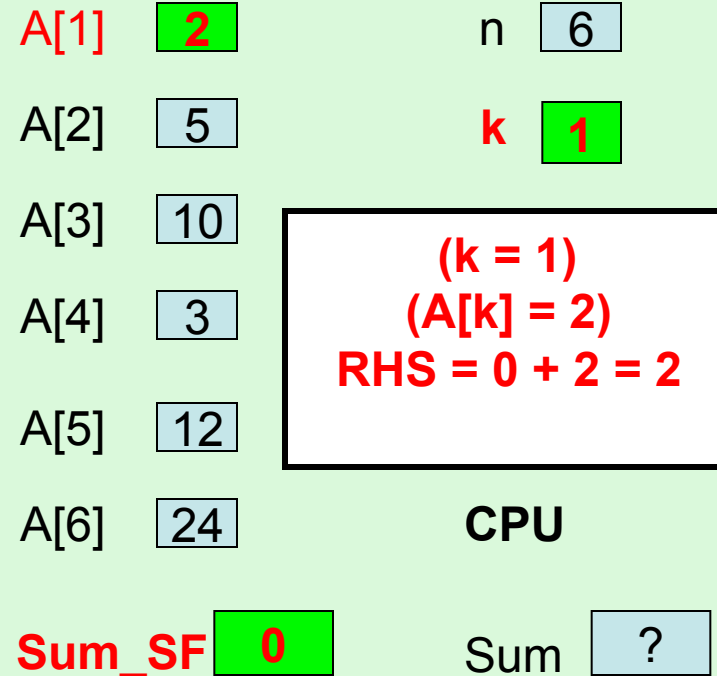
A[1]	2	n	6
A[2]	5	k	1
A[3]	10	Is (k ≤ n)? Is (1 ≤ 6)? True	
A[4]	3		
A[5]	12	CPU	
A[6]	24		
Sum_SF	0	Sum	?

**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*



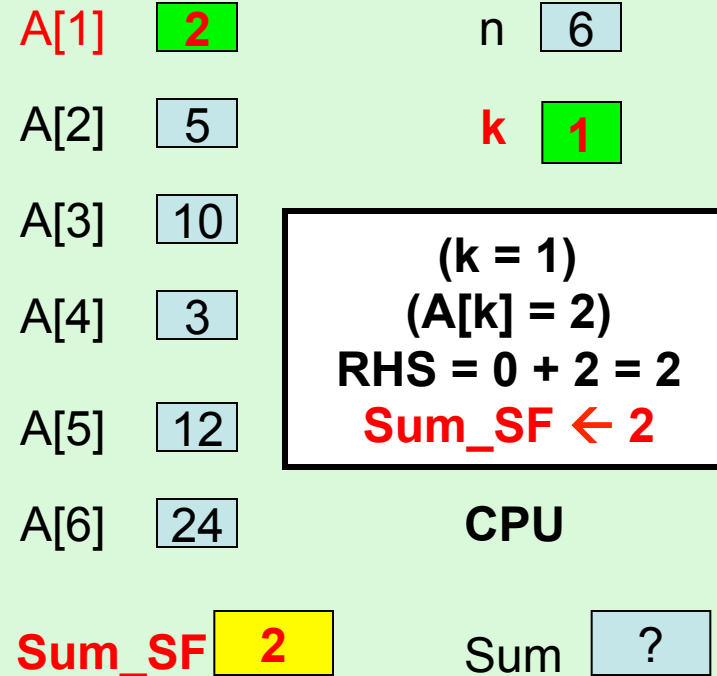
Calculate value of expression  
on the right-hand-side;  
Note: In A[k], k is used as an  
index (subscript) to the array A



# 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  $\leftarrow$  0;  
    k  $\leftarrow$  1;  
    while (k  $\leq$  n) do  
        Sum_SF  $\leftarrow$  Sum_SF + A[k];  
        k  $\leftarrow$  k + 1;  
    endwhile  
    Sum  $\leftarrow$  Sum_SF;  
    Print "Sum is", Sum;  
end;
```

*inside body of loop*


A[1]	2	n	6
A[2]	5	k	2
A[3]	10		
A[4]	3		
A[5]	12		
A[6]	24		
Sum_SF	2	CPU	
		Sum	?

**k  $\leftarrow$  k + 1;**  
**k  $\leftarrow$  1 + 1;**

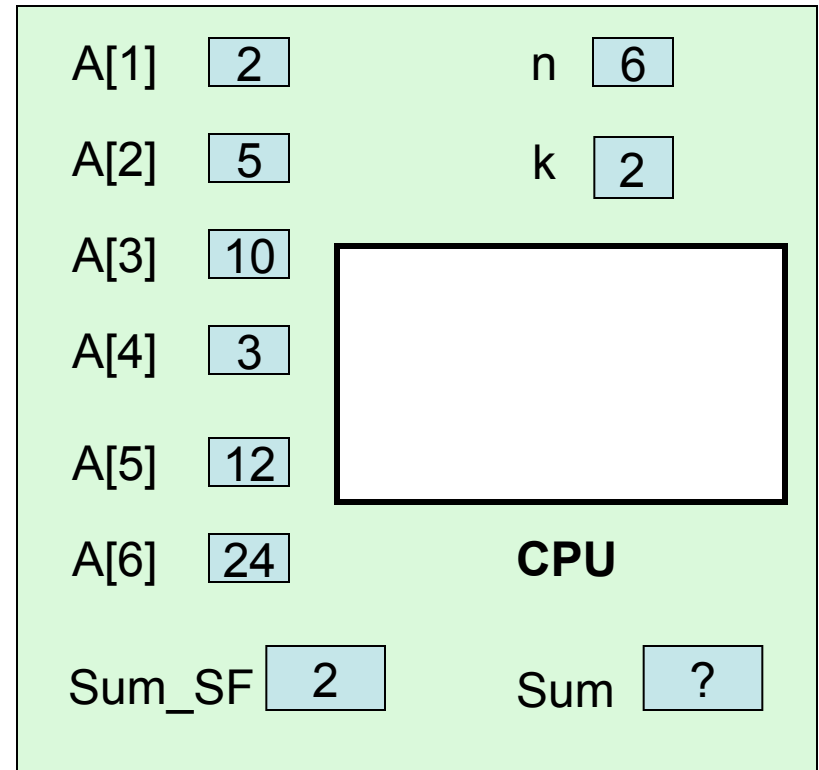
**This increments value of k;**

# 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;
```



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

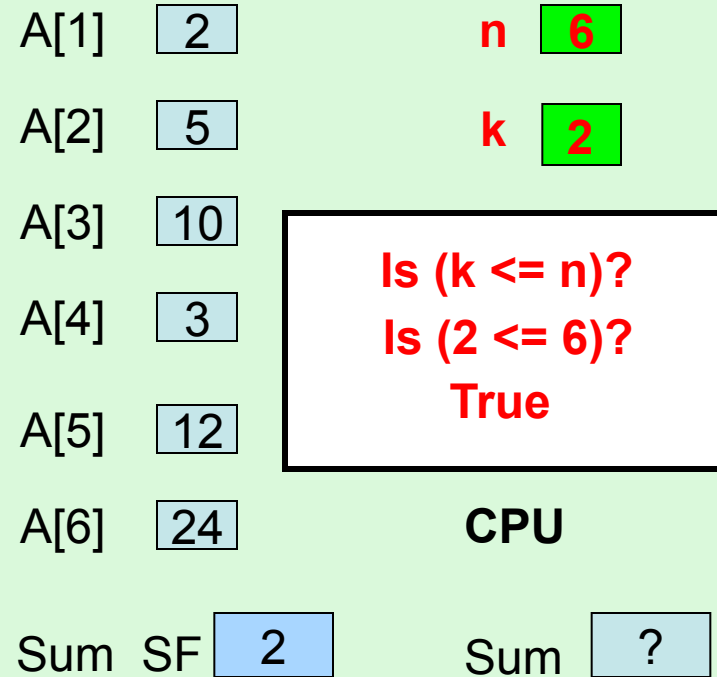


Reached end of the loop body.  
**Must loop back and re-test the loop condition.**  
**(We will skip this in future loops)**

## 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...*

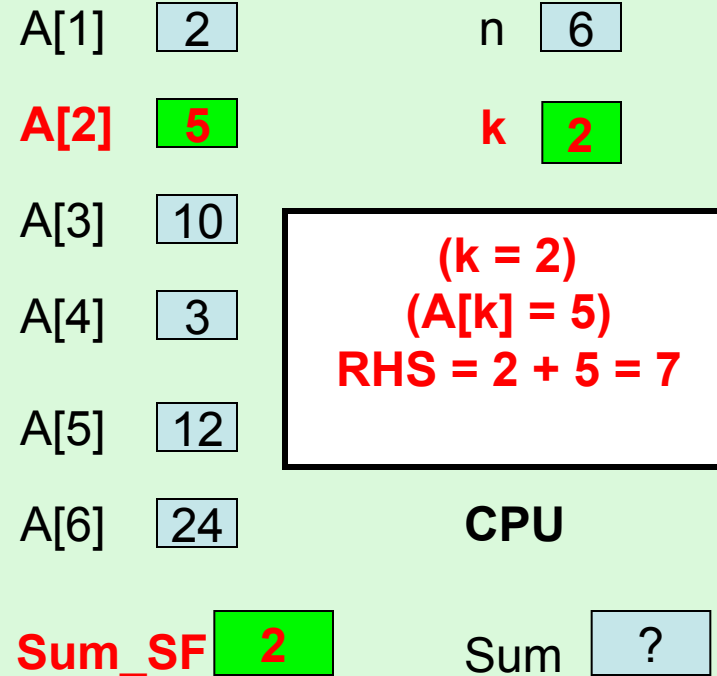


Testing the loop condition again with new value of k.  
(This is call a pre-test loop, where testing is done before entering the loop.)

# 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*

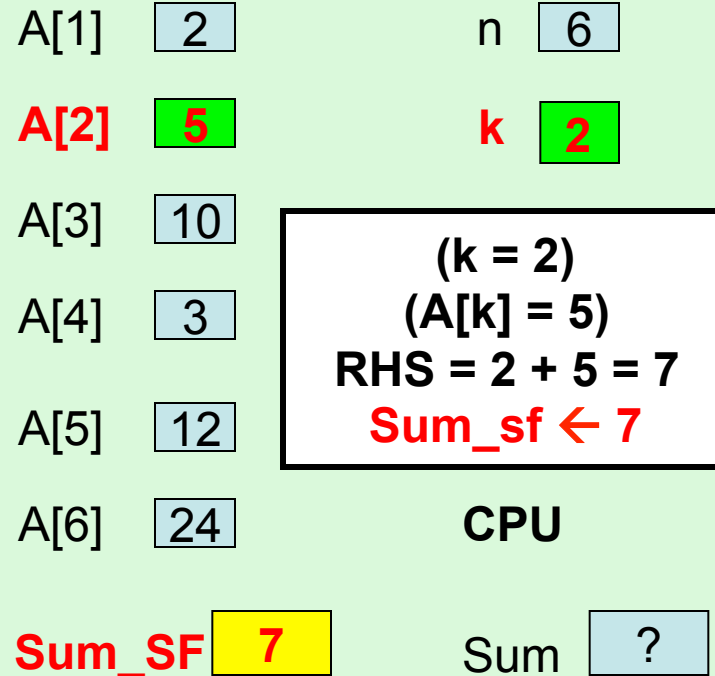


Notice that A[k] now gets the second number in the list;

# 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*

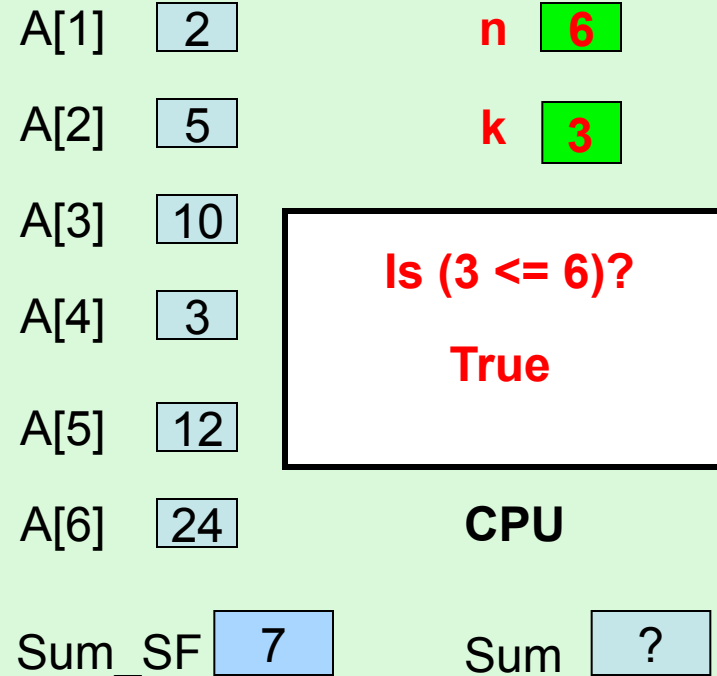
A[1]	2	n	6
A[2]	5	k	3
A[3]	10	<b>k ← k + 1; k ← 2 + 1;</b>	
A[4]	3		
A[5]	12	CPU	
A[6]	24	Sum_SF	7
		Sum	?

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*

A[1]	2	n	6
A[2]	5	k	3
A[3]	10	<b>(k = 3) (A[k] = 10) Rhs = 7 + 10 Sum_SF ← 17</b>	
A[4]	3		
A[5]	12	<b>CPU</b>	
A[6]	24	Sum_SF	17
		Sum	?

Now A[k] gets the third element  
Add A[3] to Sum\_sf;

# Algorithm Array-Sum

```
Array-Sum(A, n);  
begin  
    Sum_SF  $\leftarrow$  0;  
    k  $\leftarrow$  1;  
    while (k  $\leq$  n) do  
        Sum_SF  $\leftarrow$  Sum_SF + A[k];  
        k  $\leftarrow$  k + 1;  
    endwhile  
    Sum  $\leftarrow$  Sum_SF;  
    Print "Sum is", Sum;  
end;
```

*inside body of loop*

A[1]	2	n	6
A[2]	5	k	4
A[3]	10		
A[4]	3		
A[5]	12		
A[6]	24		
Sum_SF	17	CPU	
		Sum	?

**k  $\leftarrow$  k + 1;**  
**k  $\leftarrow$  3 + 1;**

increment 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...*

A[1]	2	n	6
A[2]	5	k	4
A[3]	10	Is (4 ≤ 6)? True	
A[4]	3		
A[5]	12	CPU	
A[6]	24	Sum_SF	17
		Sum	?

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*

A[1]	2	n	6
A[2]	5	k	4
A[3]	10		
A[4]	3		
A[5]	12		
A[6]	24		
Sum_SF	20	CPU	Sum ?

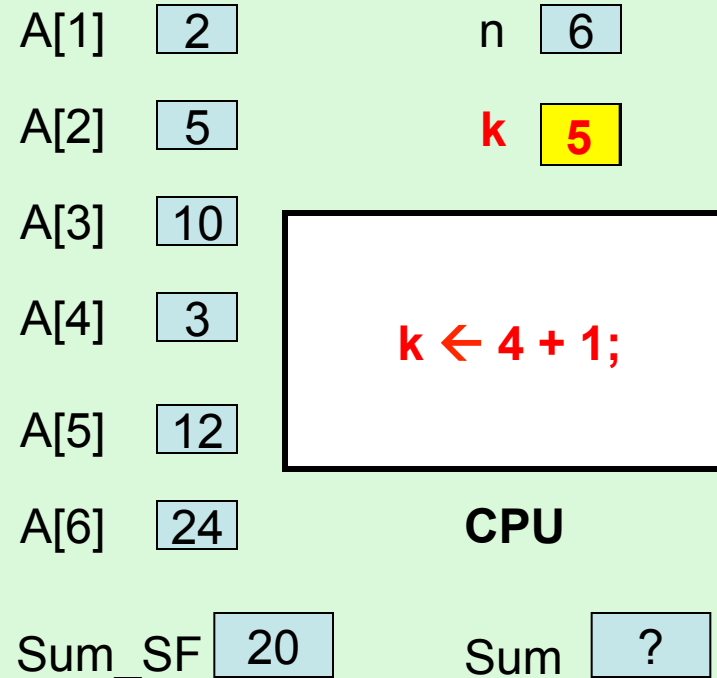
(k = 4)  
(A[k] = 3)  
Rhs = 17 + 3  
Sum\_SF ← 20

Add A[4] 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*



increment 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...*

A[1]	2	n	6
A[2]	5	k	5
A[3]	10	Is (5 ≤ 6)? True	
A[4]	3		
A[5]	12	CPU	
A[6]	24		
Sum_SF	20	Sum	?

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*

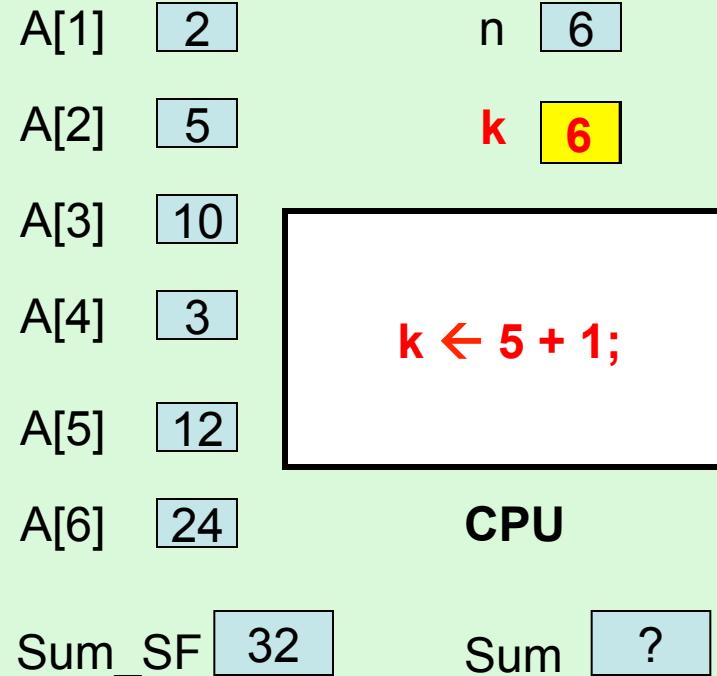
A[1]	2	n	6
A[2]	5	k	5
A[3]	10	<b>(k = 5) (A[k] = 12) Rhs = 20 + 12 Sum_SF ← 32</b>	
A[4]	3		
A[5]	12	CPU	
A[6]	24	Sum_SF	32
		Sum	?

Add A[5] 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*



increment 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...*

A[1]	2	n	6
A[2]	5	k	6
A[3]	10	Is (6 ≤ 6)? True	
A[4]	3		
A[5]	12	CPU	
A[6]	24	Sum	
Sum_SF	32	Sum	?

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*

A[1]	2	n	6
A[2]	5	k	6
A[3]	10		
A[4]	3		
A[5]	12		
A[6]	24		
Sum_SF	56	CPU	Sum
			?

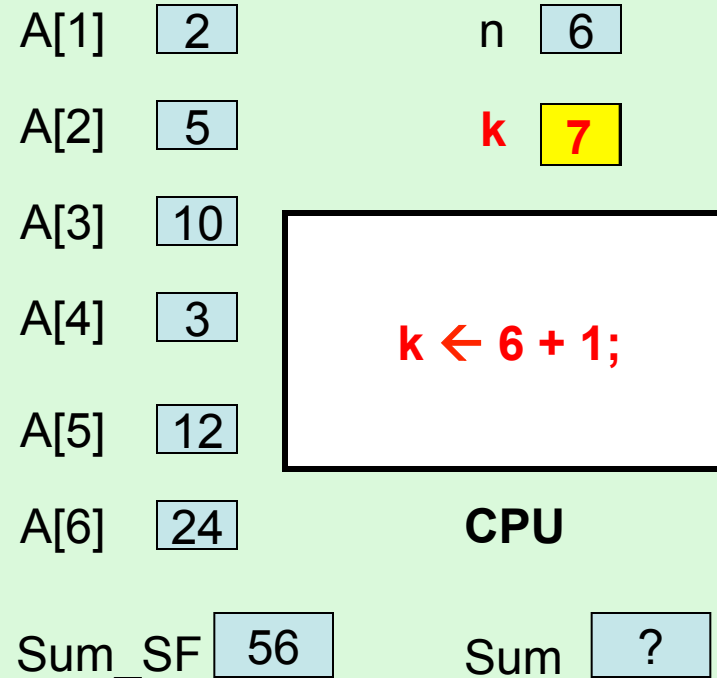
(k = 6)  
(A[k] = 24)  
Rhs = 32 + 24  
Sum\_SF ← 56

Add A[6] 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*



increment 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...*

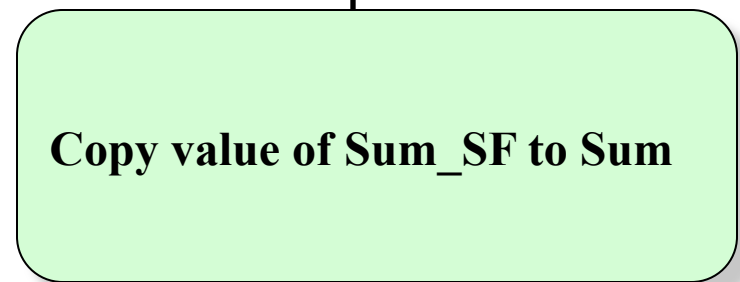
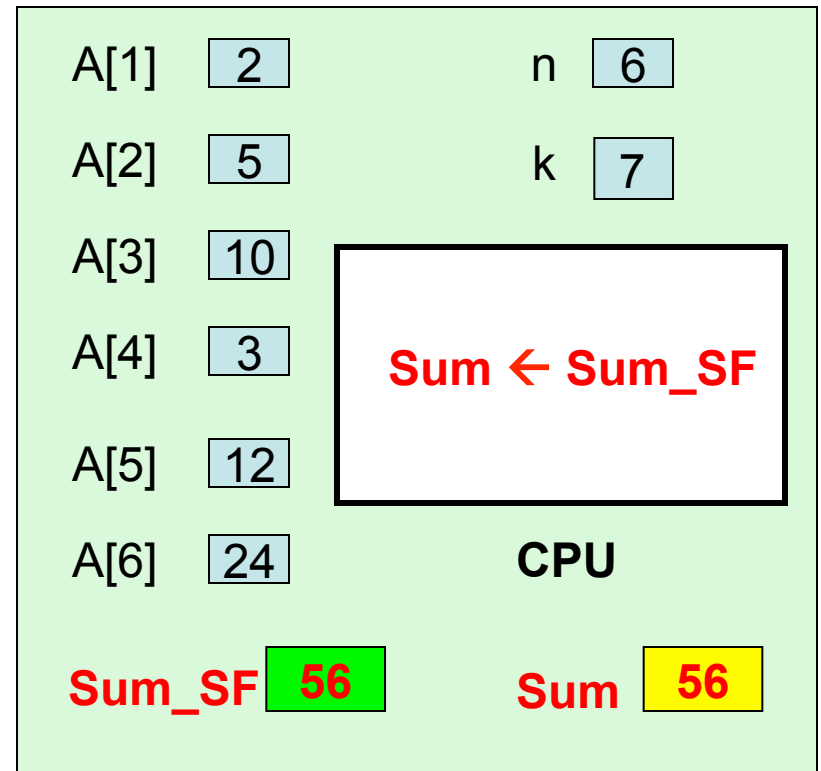
A[1]	2	n	6
A[2]	5	k	7
A[3]	10	Is (7 ≤ 6)? False	
A[4]	3		
A[5]	12	CPU	
A[6]	24	Sum_SF	56
		Sum	?

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...*



# 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...*

A[1]	2	n	6
A[2]	5	k	7
A[3]	10		
A[4]	3		
A[5]	12		
A[6]	24		
Sum_SF	56	CPU	Sum 56

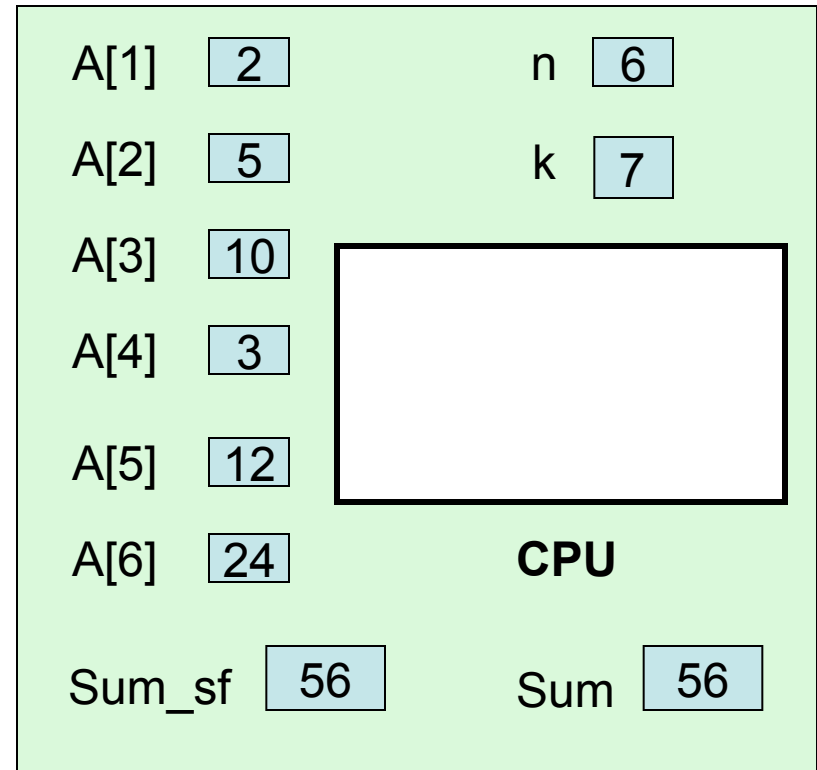
**Output of Algorithm Sum:**

**Sum is 56**

# Algorithm Array-Sum

```
Array-Sum(A, n);  
begin  
    Sum_SF  $\leftarrow$  0;  
    k  $\leftarrow$  1;  
    while (k  $\leq$  n) do  
        Sum_SF  $\leftarrow$  Sum_SF + A[k];  
        k  $\leftarrow$  k + 1;  
    endwhile  
    Sum  $\leftarrow$  Sum_SF;  
    Print "Sum is", Sum;  
end;
```

*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;
```

### 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.