Animation of Algorithm

Goal:

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

Algorithm: Sum 1-to-5 (find sum from 1 to 5)

(Note: Similar to Sum 1-to-100, but shorter!!)

Observe carefully:

- **sequential operations/statements**
- **conditional statements,**
- **iterative statements,**

Simulating an *Algorithm*

0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *)
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Let's ***animate*** the execution of this simple algorithm.

Simulating an *Algorithm*

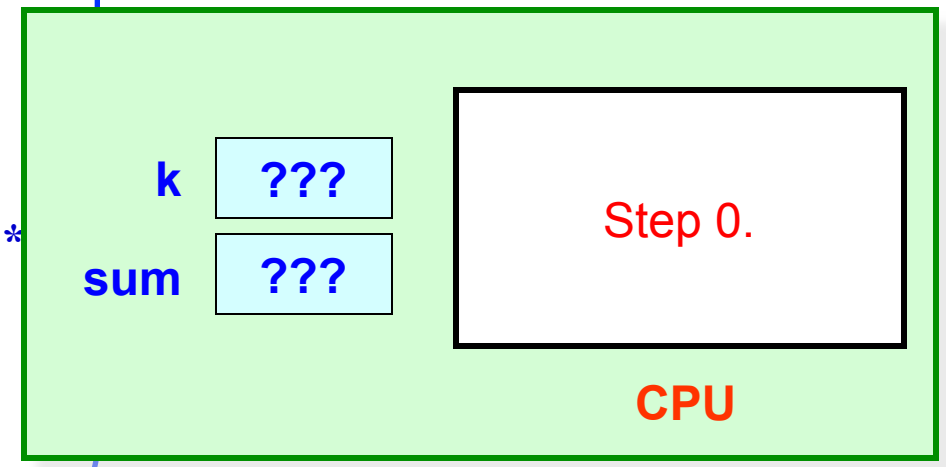
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Initial state of the algorithm



Our abstract model of the computer

Simulating an *Algorithm*

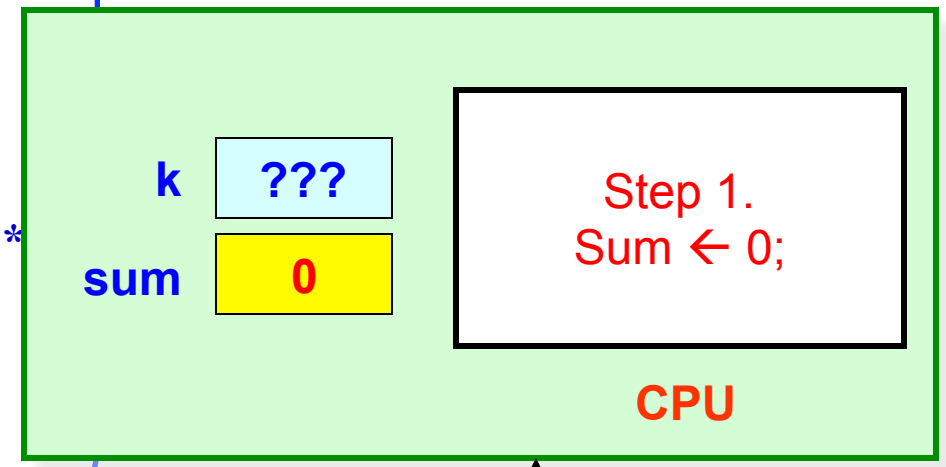
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Start of execution, at Step 1.



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

Simulating an *Algorithm*

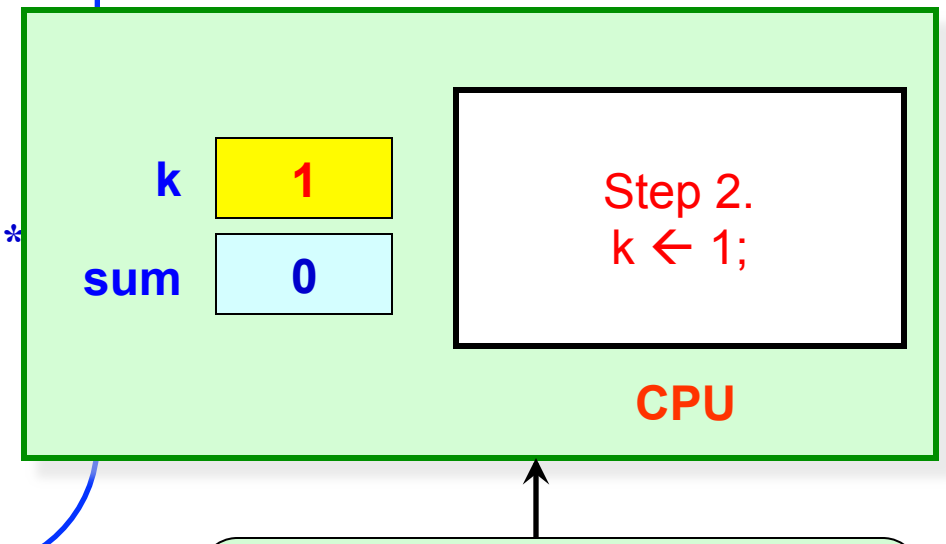
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 2.



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

Simulating an *Algorithm*

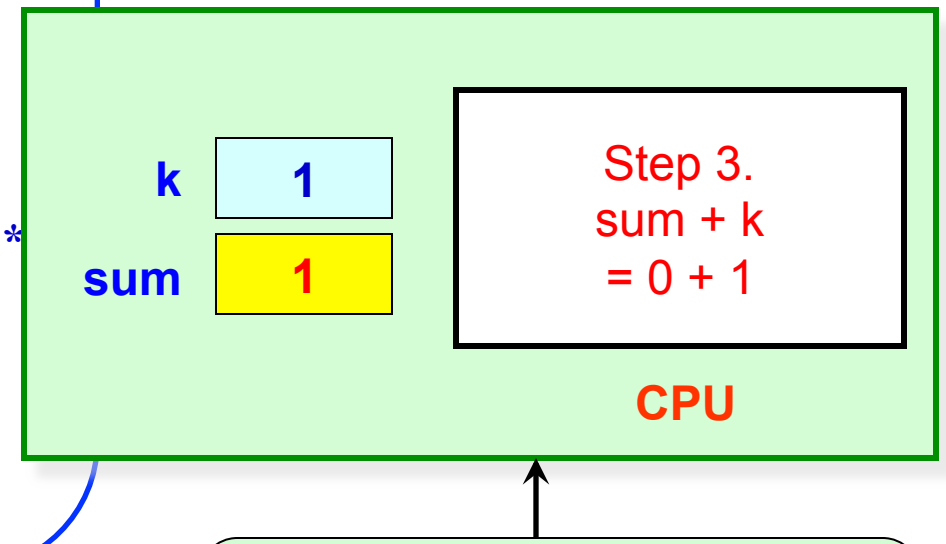
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 3.
start of "body-of-loop"



Assignment statement;
The new value of sum is stored;
the old value is gone.

Simulating an *Algorithm*

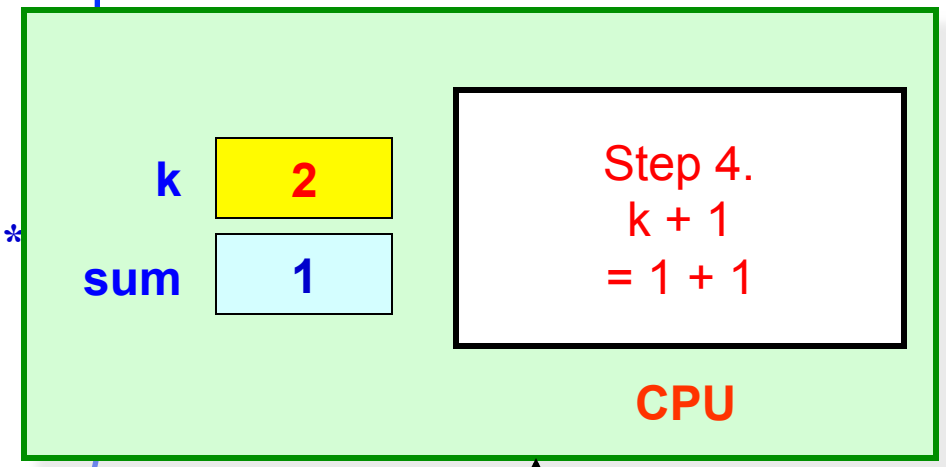
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 4. inside "body-of-loop"



Assignment statement;
The new value of k is stored;
the old value is gone.

Simulating an *Algorithm*

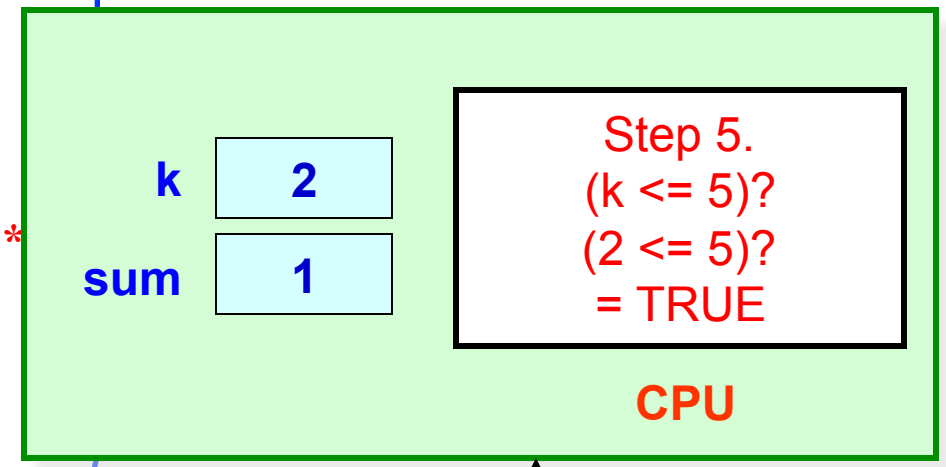
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 5.
loop-test



Condition check:
evaluate (k <= 5)?
TRUE → execute Step 6 next.

Simulating an *Algorithm*

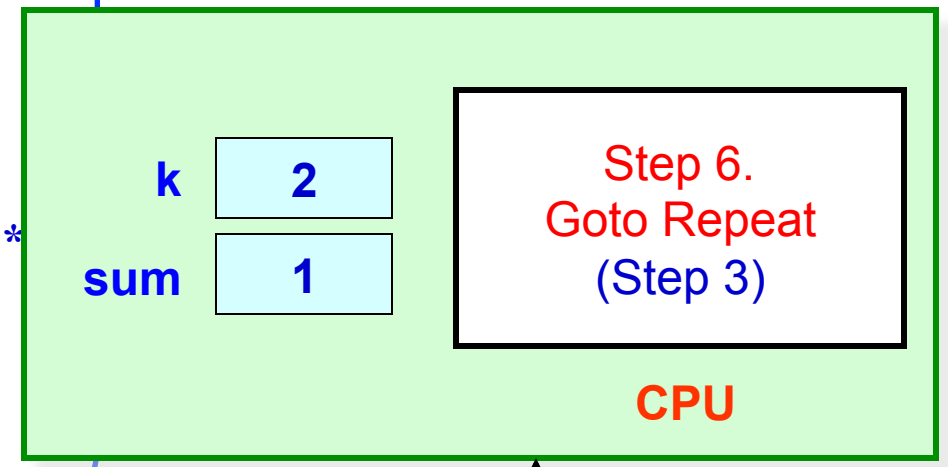
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 6.



goto "Repeat" means to get algorithm to continue at the step labelled "repeat".

Simulating an *Algorithm*

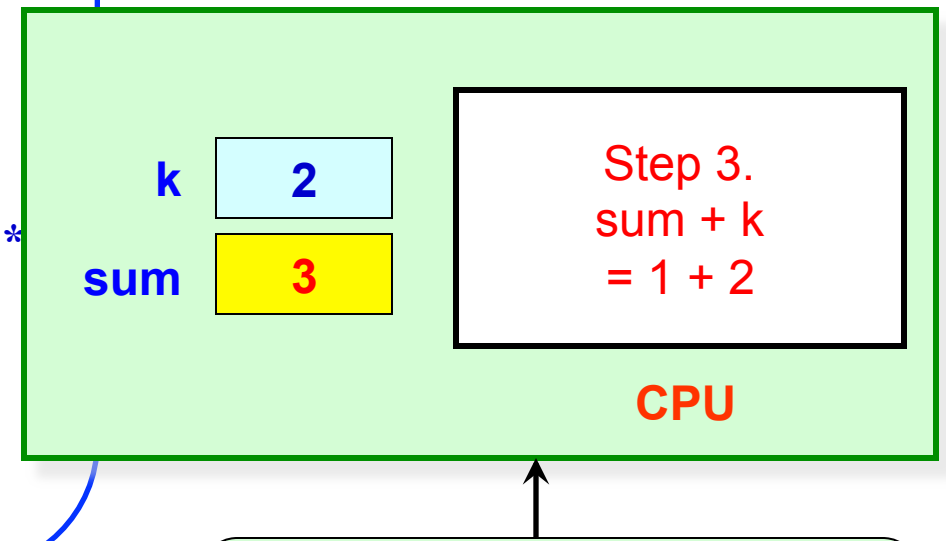
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 3.
2nd round of loop-body



Add 2 to sum;
The new value of sum is stored;
the old value is gone.

Simulating an *Algorithm*

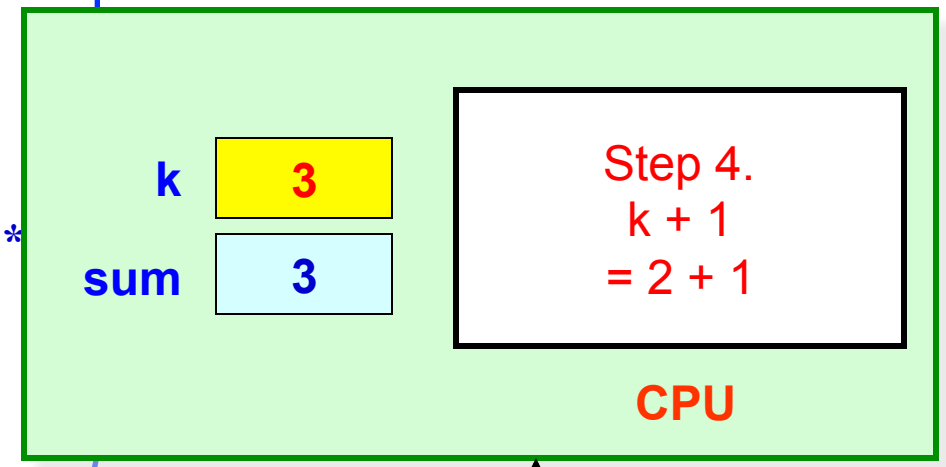
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 4.
2nd round of loop-body



Increment k;
The new value of k is stored;
the old value is gone.

Simulating an *Algorithm*

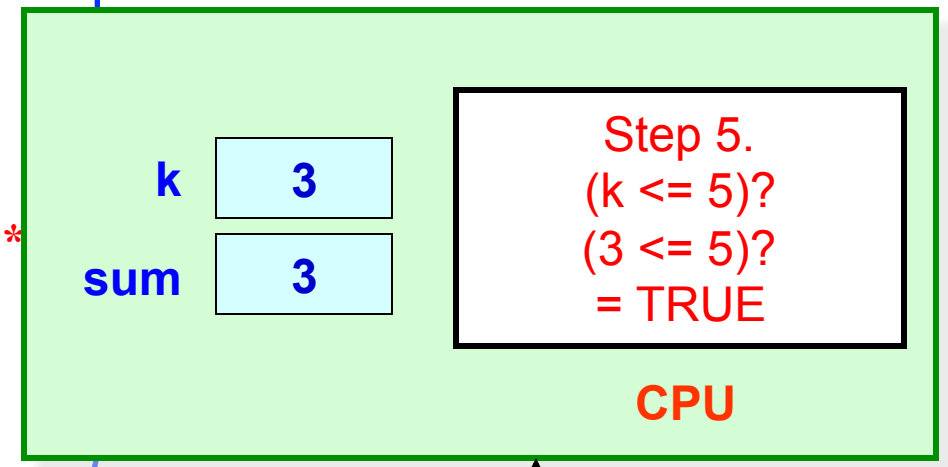
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 5.
2nd loop-test



Condition check:
evaluate (k <= 5)?
TRUE → execute Step 6 next.

Simulating an *Algorithm*

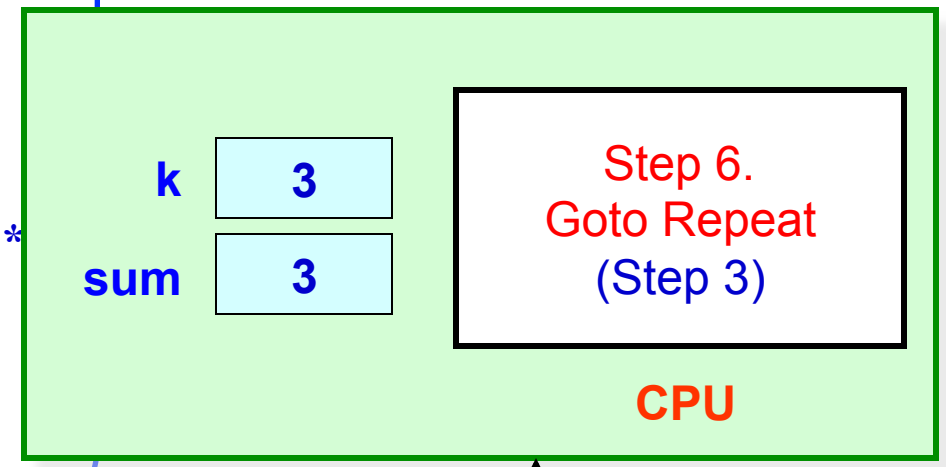
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 6.
2nd round



Goto Step 3 and
Execute the loop-body again.

Simulating an *Algorithm*

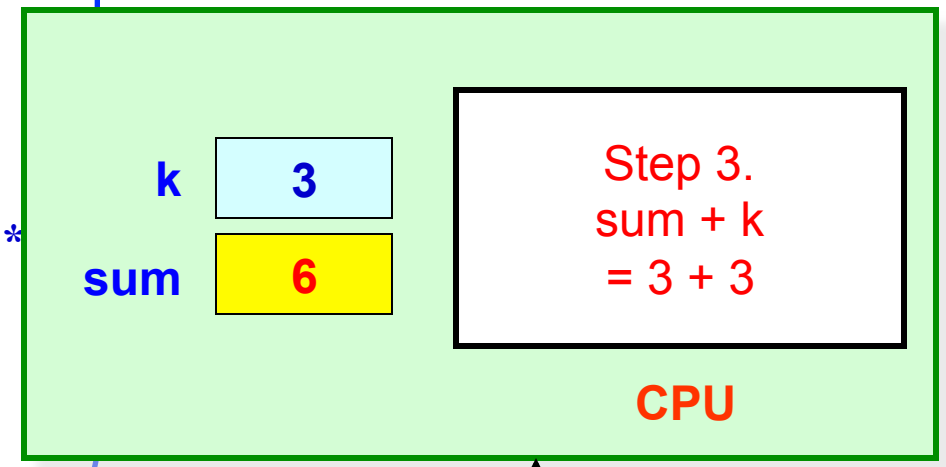
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 3.
3rd round of loop-body



Add 3 to sum;
The new value of sum is stored;
the old value is gone.

Simulating an *Algorithm*

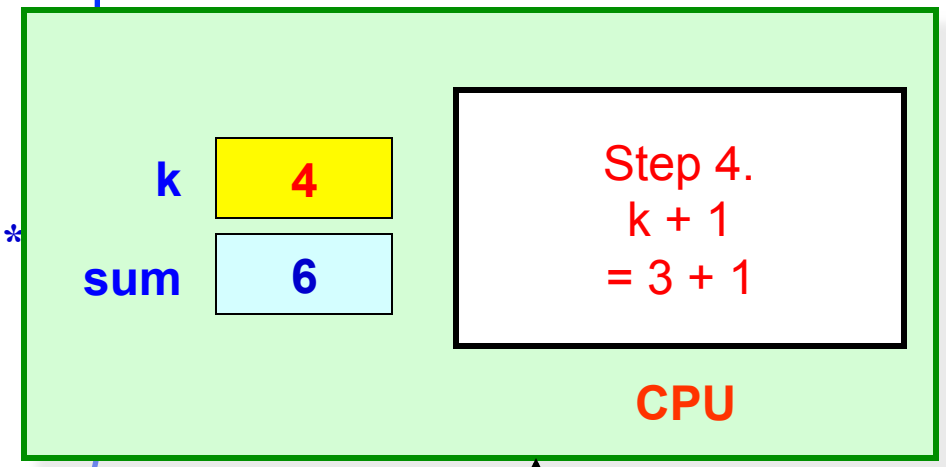
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 4.
3rd round of loop-body



Increment k;
The new value of k is stored;
the old value is gone.

Simulating an *Algorithm*

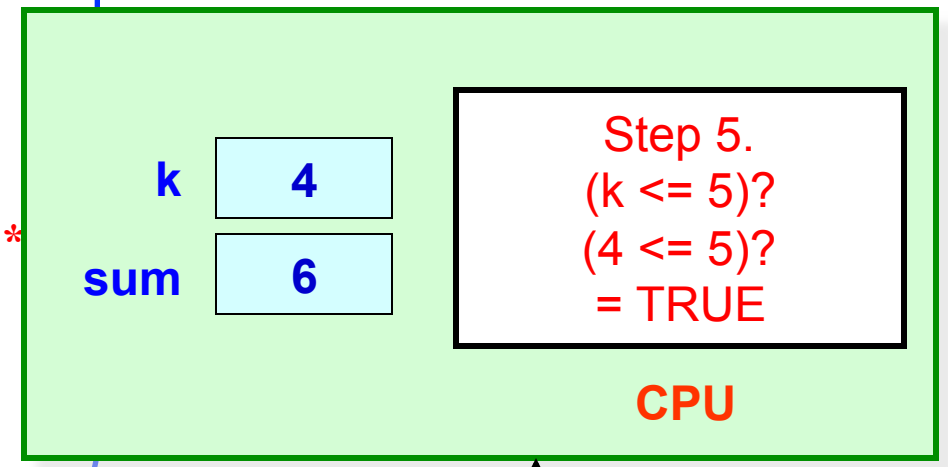
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 5.
3rd loop-test



Condition check:
evaluate (k <= 5)?
TRUE → execute Step 6 next.

Simulating an *Algorithm*

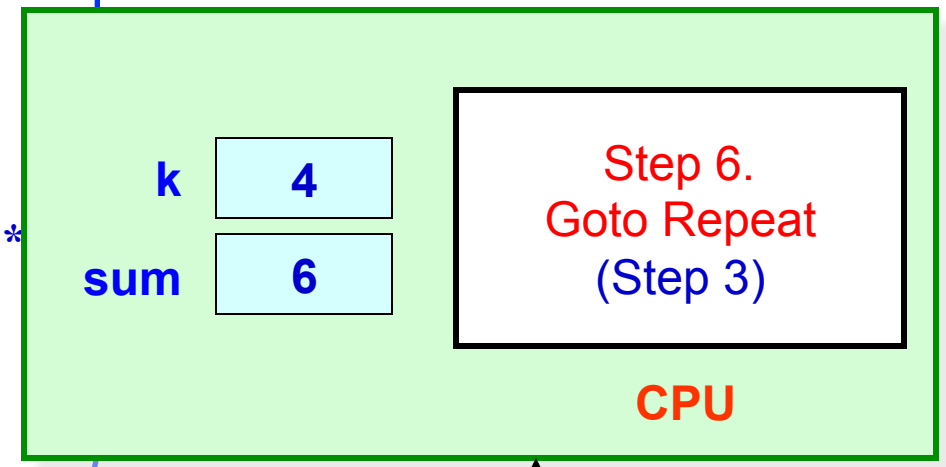
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 6.
3rd round



Goto Step 3 and
Execute the loop-body again.

Simulating an *Algorithm*

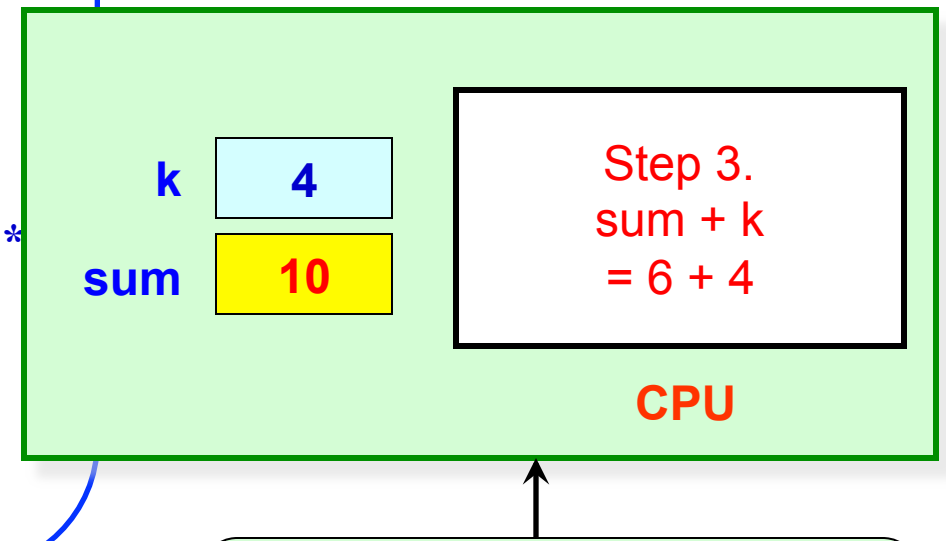
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 3.
4th round of loop-body



Add 4 to sum;
The new value of sum is stored;
the old value is gone.

Simulating an *Algorithm*

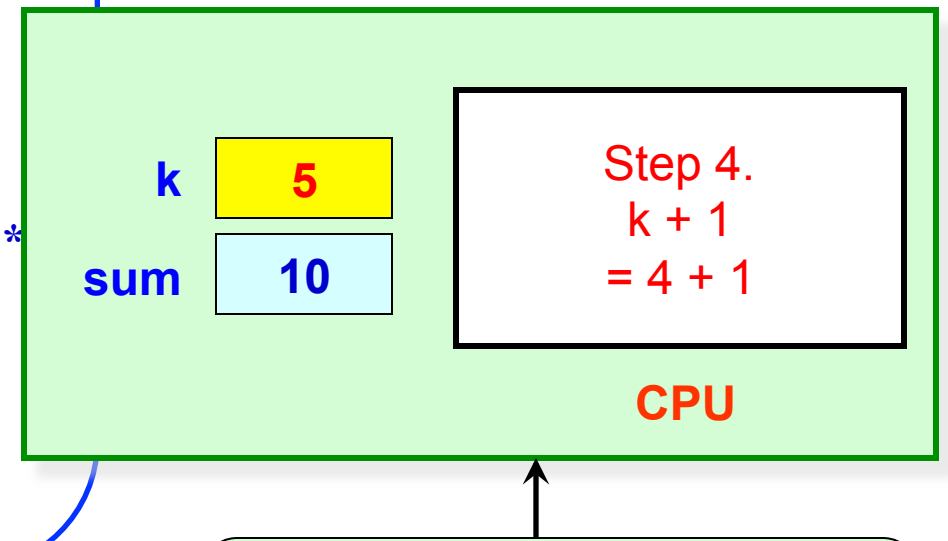
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 4.
4th round of loop-body



Increment k;
The new value of k is stored;
the old value is gone.

Simulating an *Algorithm*

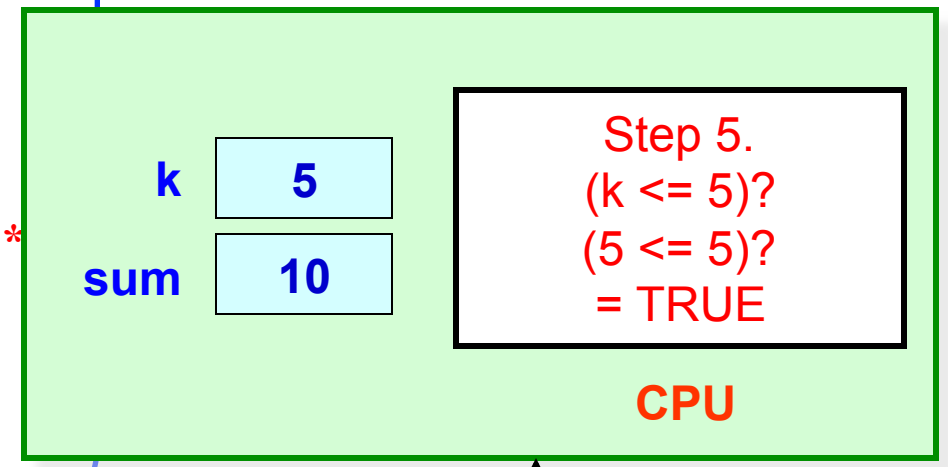
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 5.
4th loop-test



Condition check:
evaluate (k <= 5)?
TRUE → execute Step 6 next.

Simulating an *Algorithm*

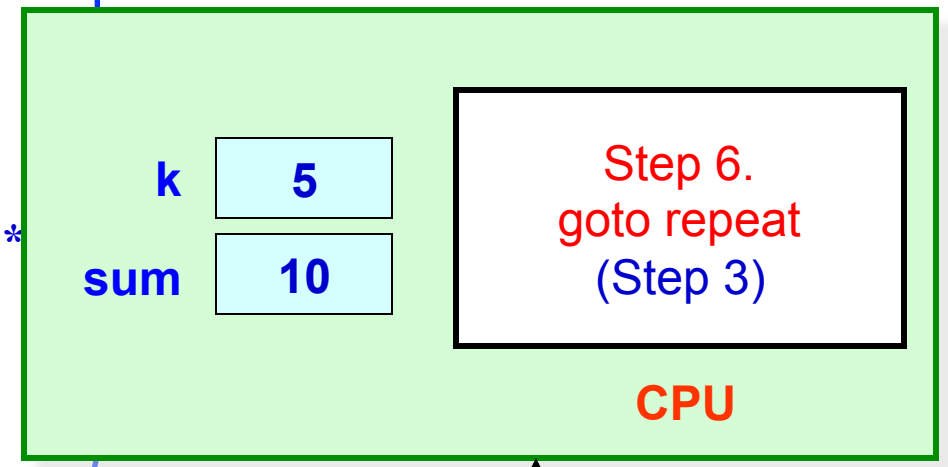
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 6.
4th round



Goto Step 3 and
Execute the loop-body again.

Simulating an *Algorithm*

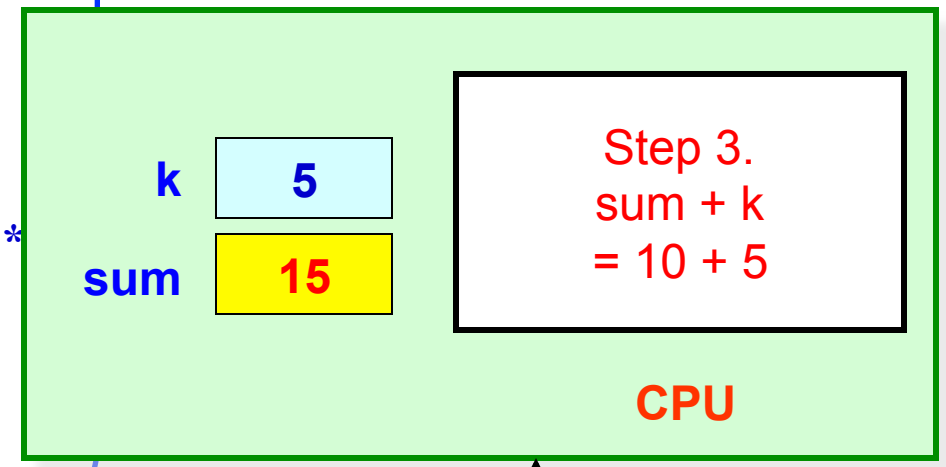
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 3.
5th round of loop-body



Add 5 to sum;
The new value of sum is stored;
the old value is gone.

Simulating an *Algorithm*

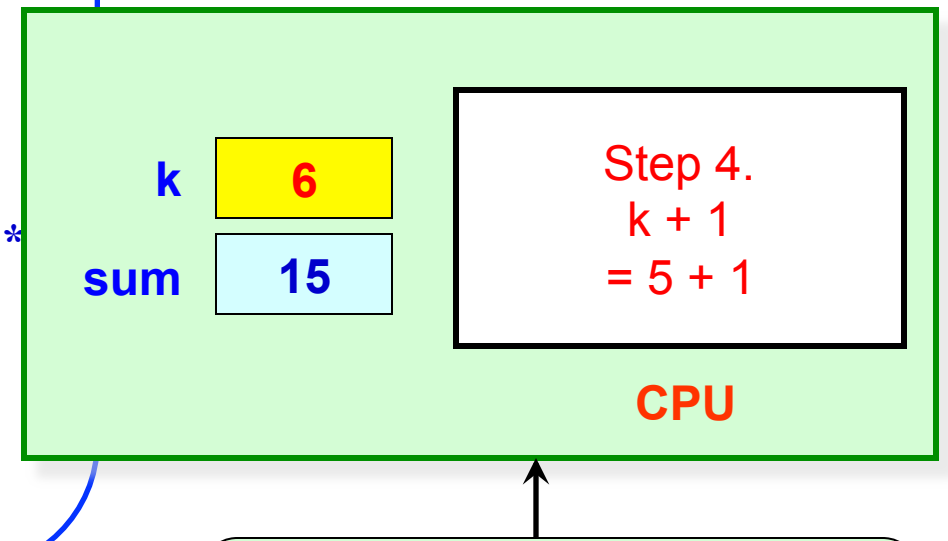
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 4.
5th round of loop-body



Increment k;
The new value of k is stored;
the old value is gone.

Simulating an *Algorithm*

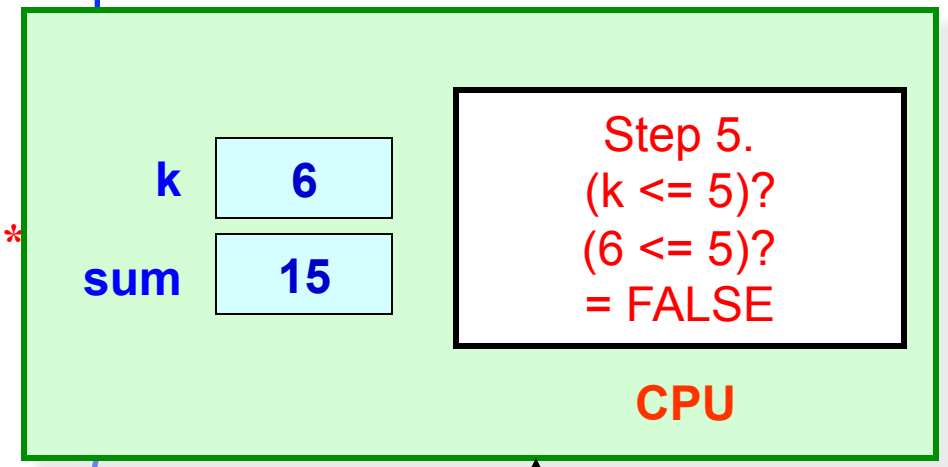
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 5.
5th loop-test



Condition check:
evaluate (k <= 5)?
FALSE → execute Step 7 next.

Simulating an *Algorithm*

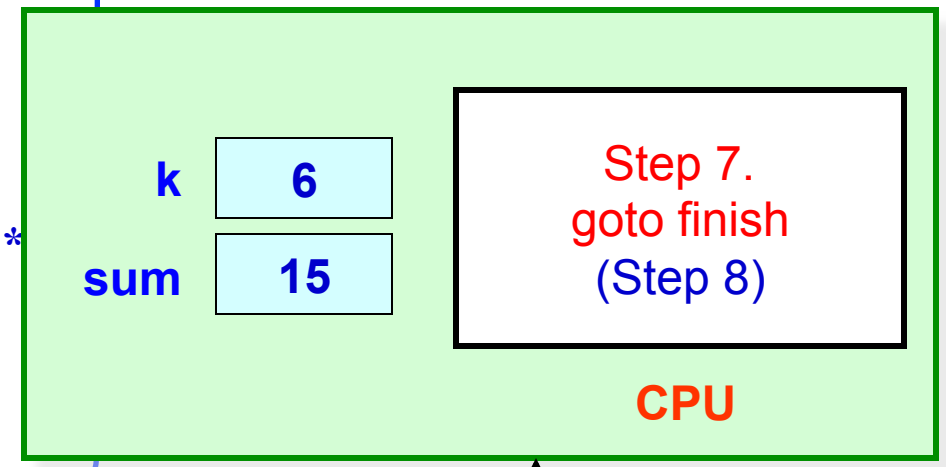
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

finish: print out the value of sum

Executing Step 7.
& exit the iterative loop



Goto finish (Step 8)
(exit the iterative loop!)

Simulating an *Algorithm*

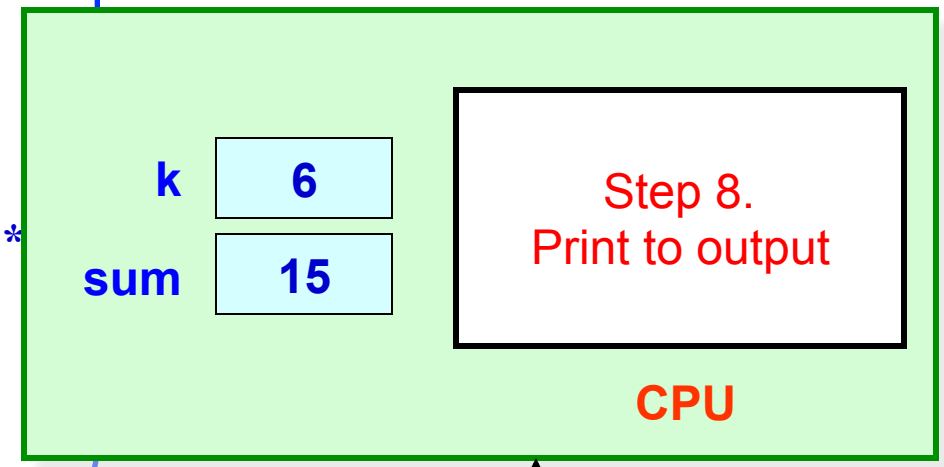
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 5) (* means < or = *
6. then Goto Step 3. Repeat
7. else Goto Finish

finish: print out the value of sum

Executing Step 8.
print output and END



Output of Algorithm:

15

Print statement;
print to output the value of sum

Summary

- **Summary of Steps:**
  - 1, 2, (3,4,5,6), (3,4,5,6), (3,4,5,6), (3,4,5,6), (3,4,5,7), 8
- **Note the sequential execution, except for**
  - Conditional statements
  - Goto statements
  - iterative statements
- **Questions:**
  - Where is the "loop-body"?
  - How many iteration of the loop-body?
  - How many times is the loop-test done?

Recurring Principle

RP5: "The Power of Iteration" (aka Recursion)

ALGORITHM Sum-1-to-100;

```
        sum ← 0
        k ← 1
Repeat: add k to sum
        add 1 to k
        If (k <= 100)
           then Goto Repeat
           else Goto Finish
Finish: print out the value of sum
```

If you are new to algorithm, go through the algorithm animation SLOWLY. Make sure you master it.

Explore further (DIY)

- We did Sum 1-to-5 (instead of Sum 1-to-100)
- DIY: Simulate the execution for the original algorithm for Sum 1-to-100?
- (Use the following "ending"-slides to help you.)

Simulating an *Algorithm*

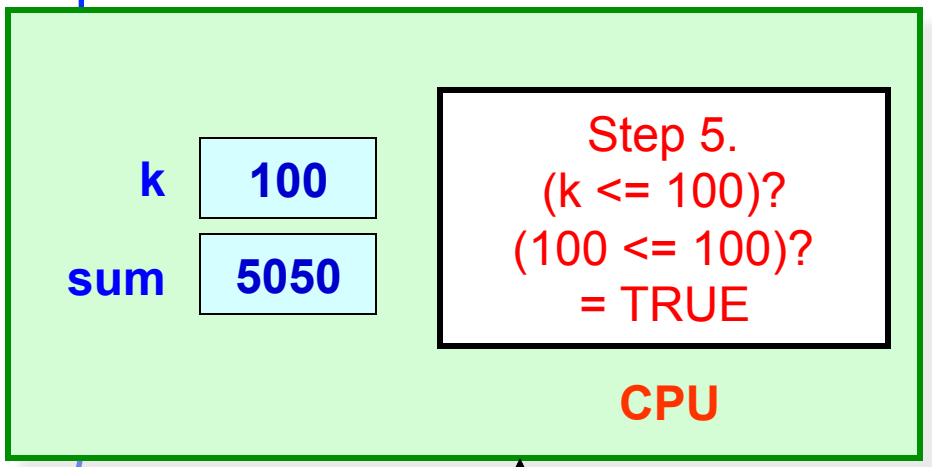
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-100;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 100)
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 5.
99th loop-test



Condition check:
evaluate (k <= 100)?
TRUE → execute Step 6 next.

Simulating an *Algorithm*

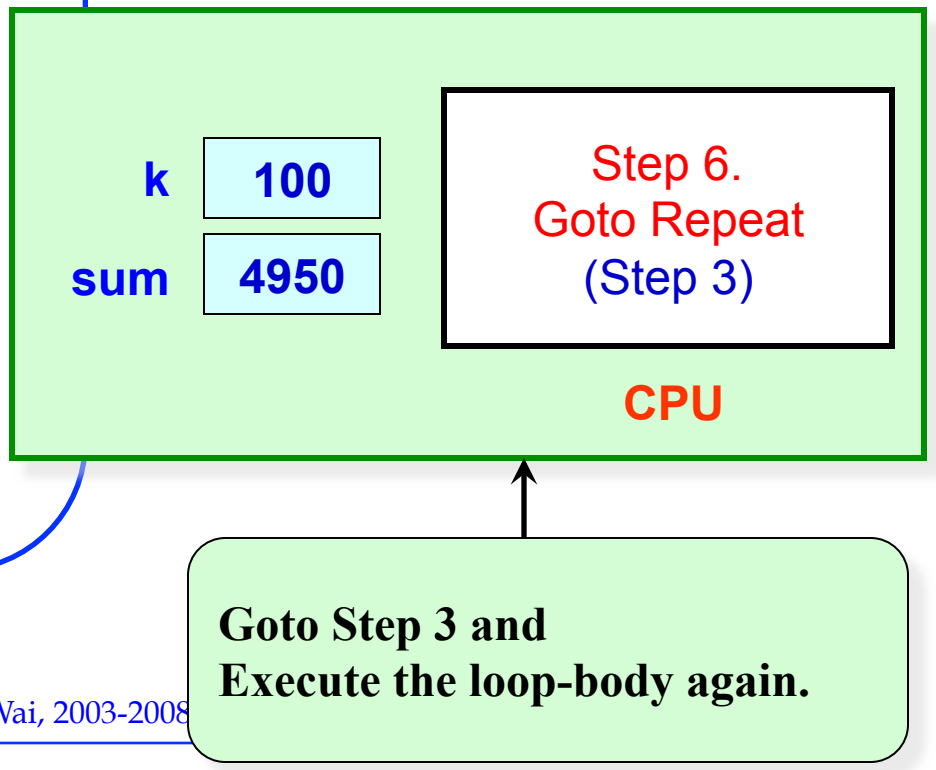
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-5;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 100)
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 6.
99th round



Simulating an *Algorithm*

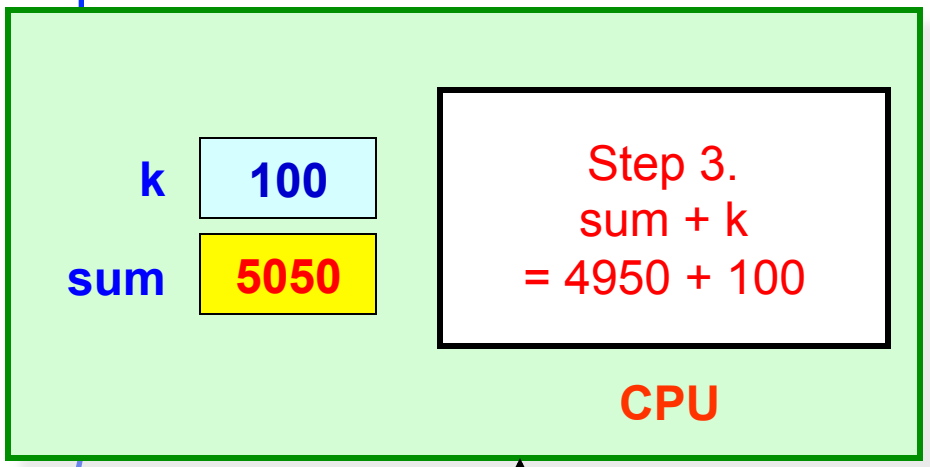
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-100;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 100)
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 3.
100th round of loop-body



Add 100 to sum;
The new value of sum is stored;
the old value is gone.

Simulating an *Algorithm*

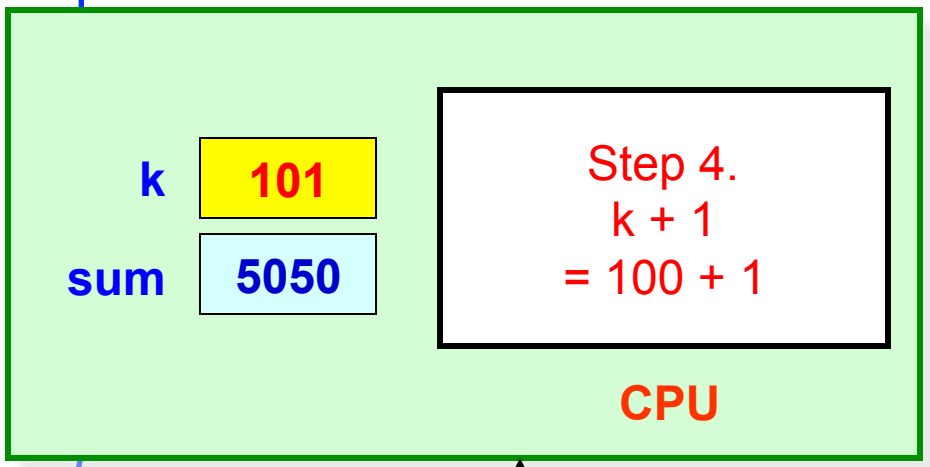
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-100;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 100)
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 4.
100^{th} round of loop-body



Increment k;
The new value of k is stored;
the old value is gone.

Simulating an *Algorithm*

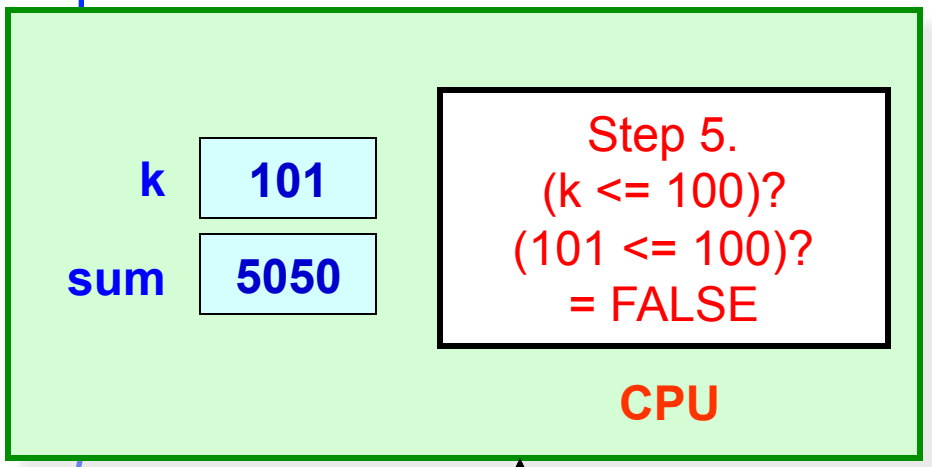
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-100;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 100)
6. then Goto Step 3. Repeat
7. else Goto Finish

Finish: print out the value of sum

Executing Step 5.
100th loop-test



Condition check:
evaluate (k <= 100)?
FALSE → execute Step 7 next.

Simulating an *Algorithm*

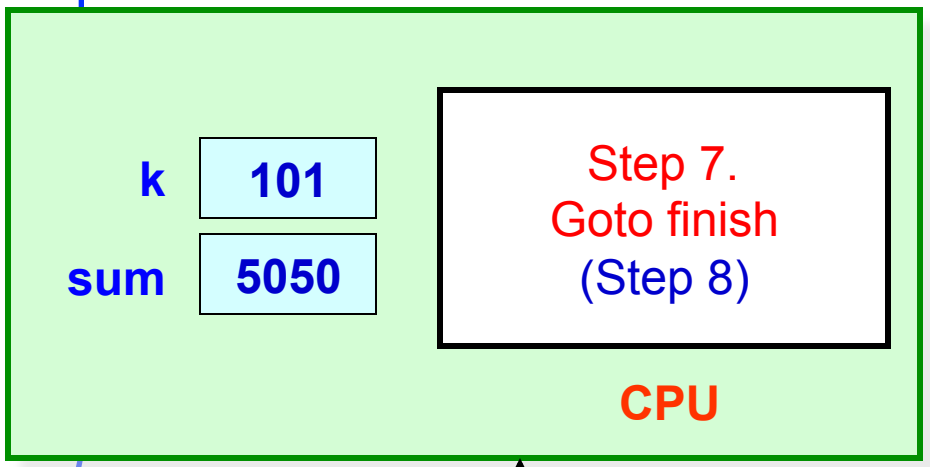
0+1=1; 1+2=3; 3+3=6; 6+4=10; 10+5=15;

ALGORITHM Sum-1-to-100;

1. sum ← 0
2. k ← 1
3. Repeat: add k to sum
4. add 1 to k
5. if (k <= 100)
6. then Goto Step 3. Repeat
7. else Goto Finish

finish: print out the value of sum

Executing Step 7.
& exit the iterative loop



Goto finish (Step 8)
(exit the iterative loop!)