Central Processing Unit

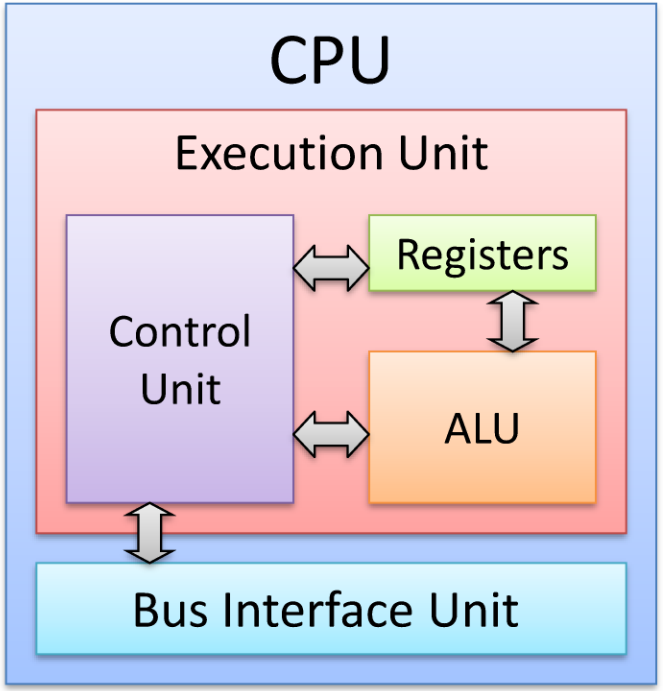
Networks and Embedded Software

Module 4.1.2

by Wolfgang Neff

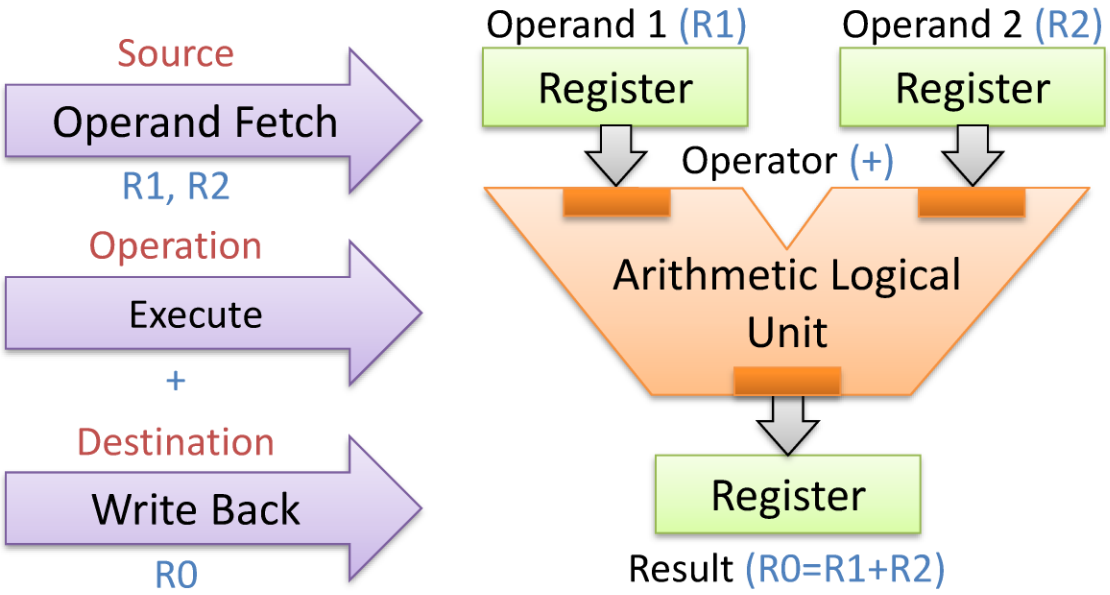
Components (1)

- Block diagram
  - Execution Unit
    - Control Unit
    - Registers
    - Arithmetic logic unit
      - ADD, SUB etc.
      - NOT, AND etc.
  - Bus Interface Unit



Components (2)

- Arithmetic logic unit

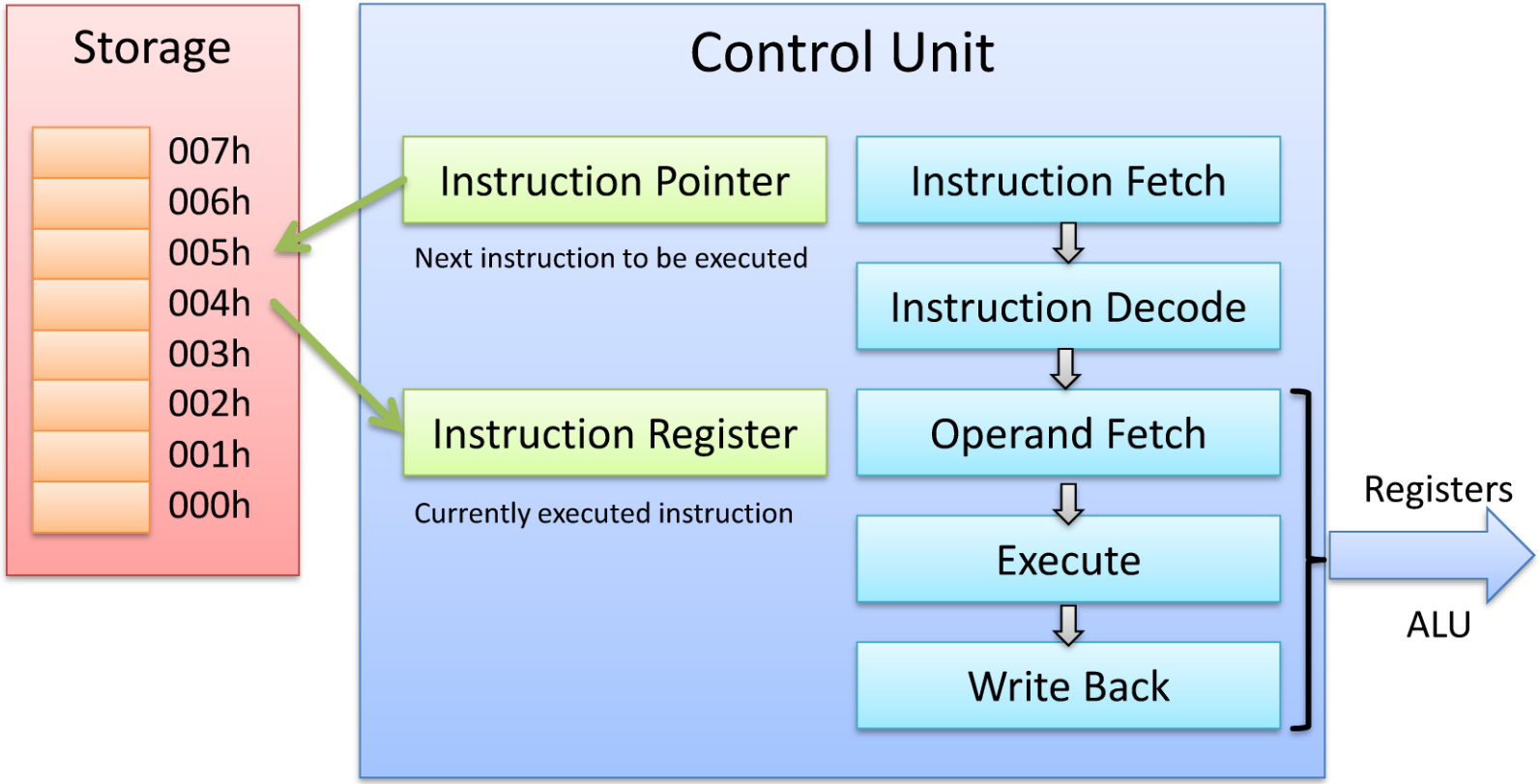


Result = Operand1 *Operator* Operand2

Destination = Source1 *Operator* Source2

Components (3)

- Structure of Control Unit



Components (4)

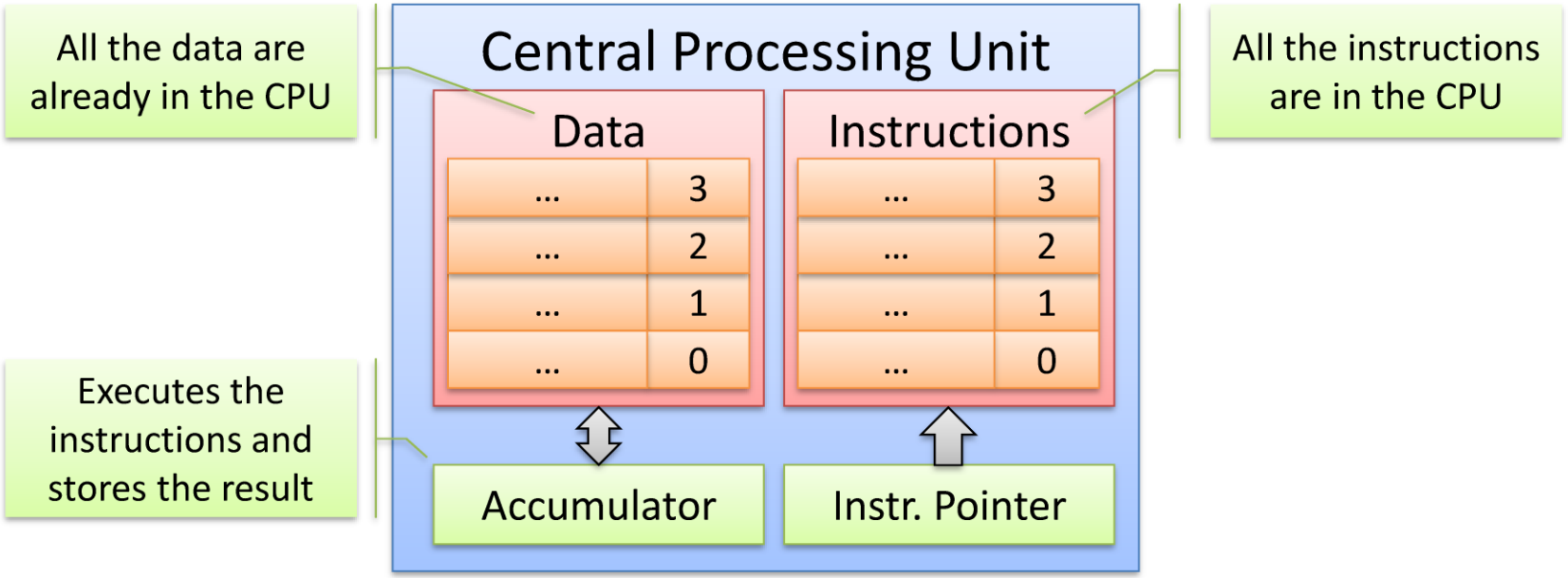
- Instruction pipeline (details: module 1.3)

Instruction	Pipeline stage						
1	IF	ID	OF	EX	WB		
2		IF	ID	OF	EX	WB	
3			IF	ID	OF	EX	WB
4				IF	ID	OF	EX
5					IF	ID	OF
Clock	**1**	**2**	**3**	**4**	**5**	**6**	**7**

Parallel execution of five instructions

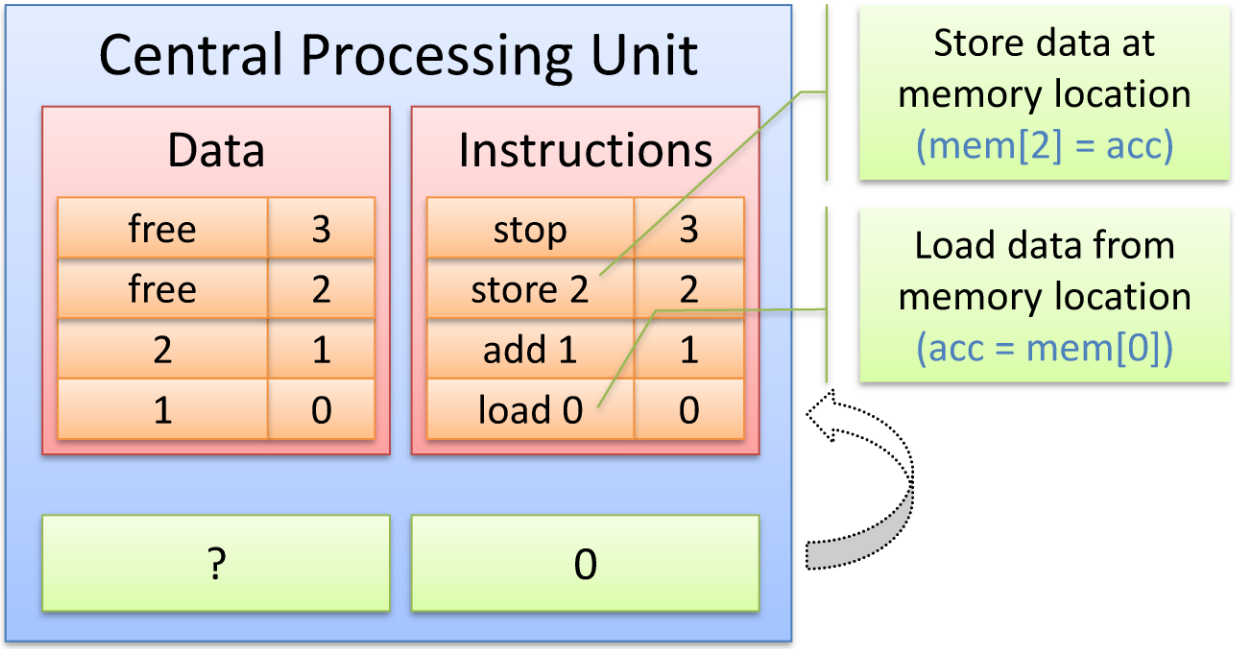
Register Machines (1)

- Theoretical model



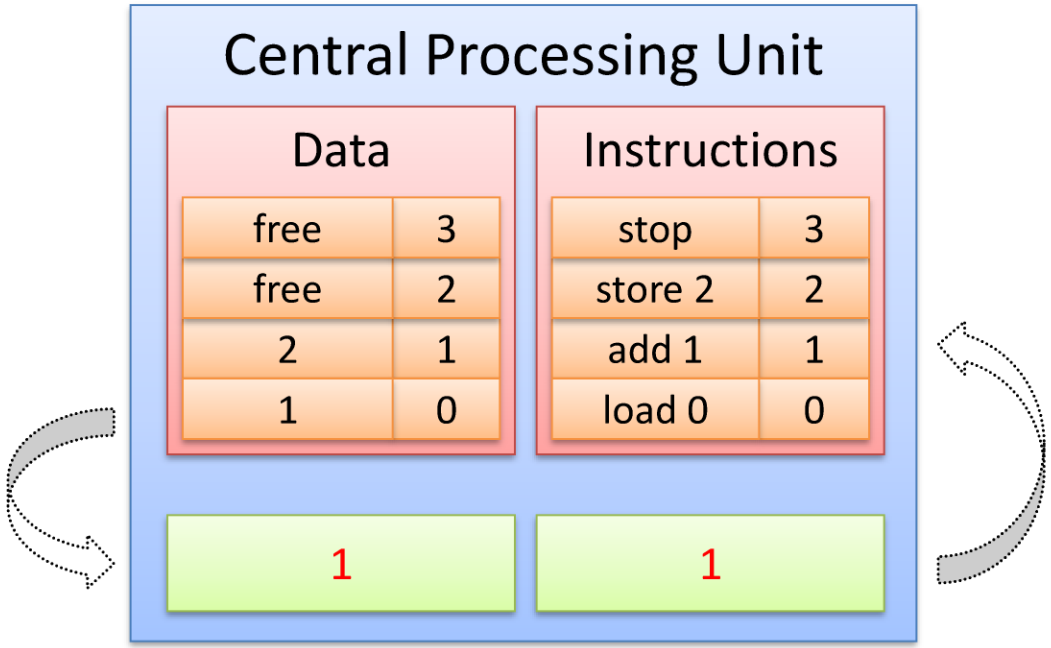
Register Machines (2)

- Program Execution



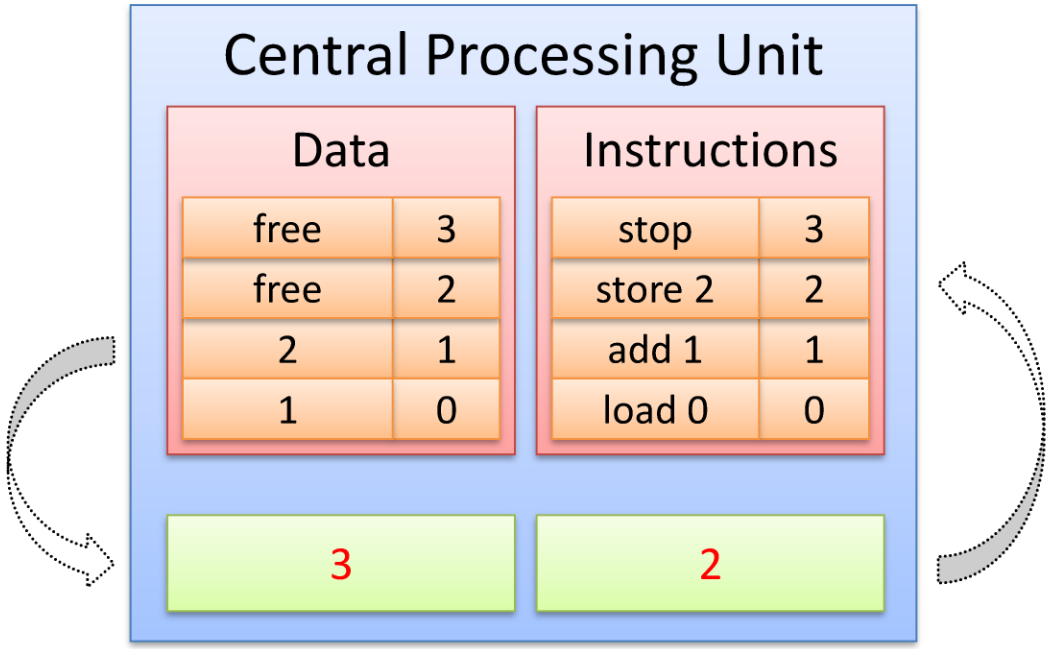
Register Machines (2)

- Program Execution (continued)



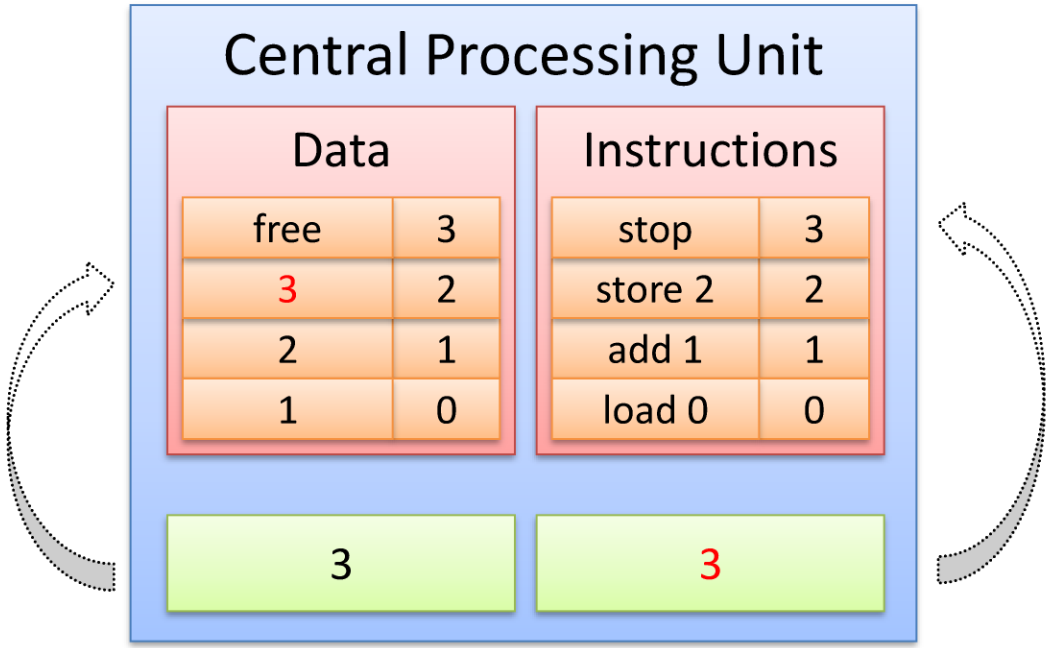
Register Machines (2)

- Program Execution (continued)



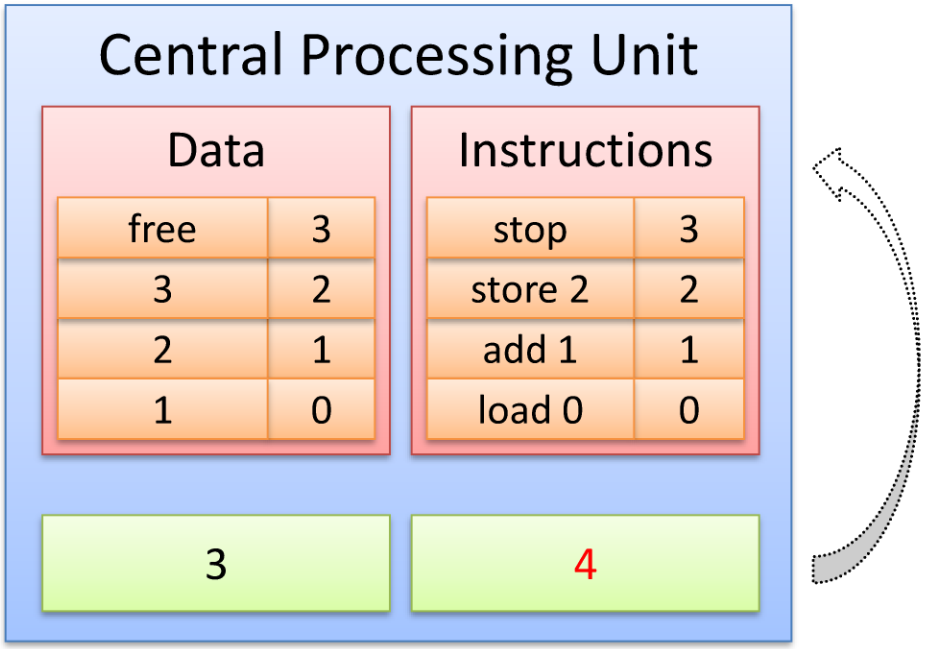
Register Machines (2)

- Program Execution (continued)



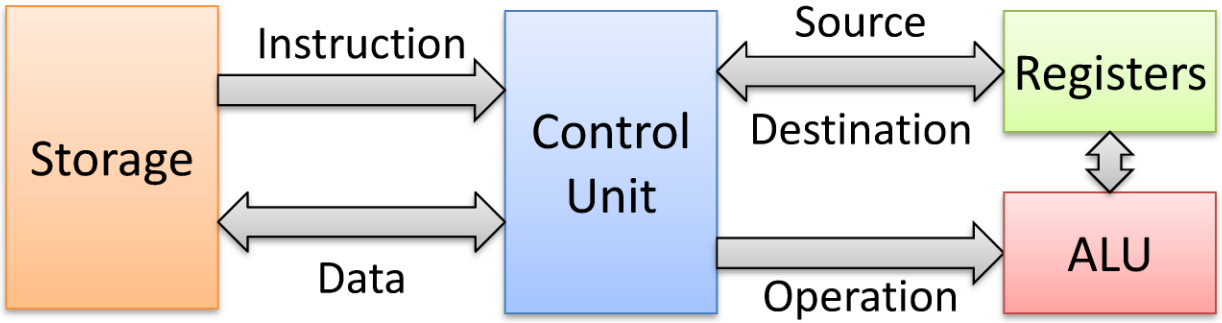
Register Machines (2)

- Program Execution (finished)



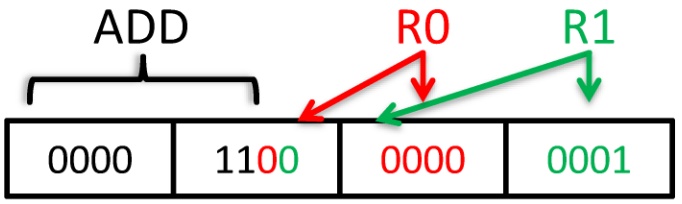
Instruction Set (1)

- Instructions
  - Statements of a program
  - Stored storage
  - Specify operation, source and destination



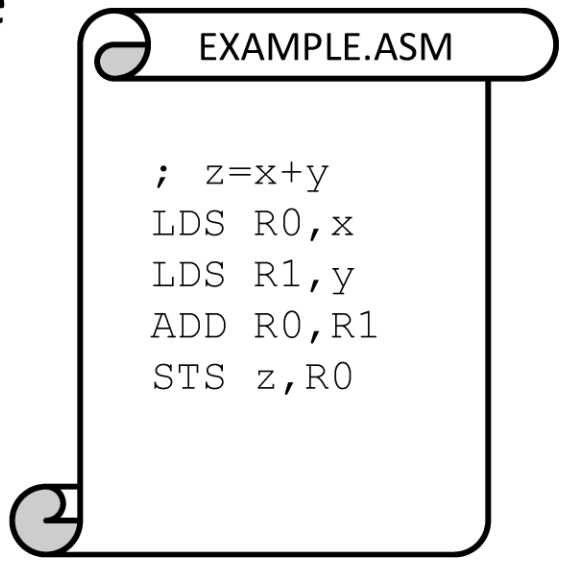
Instruction Set (2)

- Machine Language
  - Specific for every microprocessor
  - Set of instructions understood by these processors
  - Represented by numeric operation codes
  - Example:
    - Operation: ADD
    - Source: R0, R1
    - Destination: R0
    - Opcode: $0C01_{hex}$



Instruction Set (3)

- Assembly Language
  - Opcodes replaced by textual mnemonics
  - Source code is readable
  - Has to be compiled
  - Major benefits:
    - Easier to read
    - Comments allowed
    - Variables names
    - Labels



Instruction Set (4)

- Instruction Set
  - Instructions for the ALU
    - Arithmetic Instructions: *ADD*, *SUB*, *MUL*, *CP*, …
    - Logical Instructions: *AND*, *OR*, *EOR*, *COM*, …
    - Shift and Rotate Instructions: *LSL*, *LSR*, *ROL*, *ROR*, …
    - Bit Manipulation Instructions: *SBI*, *CBI*, *CLI*, *SEI*, …
  - Instructions for the Control Unit
    - Data Movement Instructions: *MOV*, *LD*, *ST*, *PUSH*, …
    - Branch Instructions: *RCALL*, *RET*, *BRcc*, …

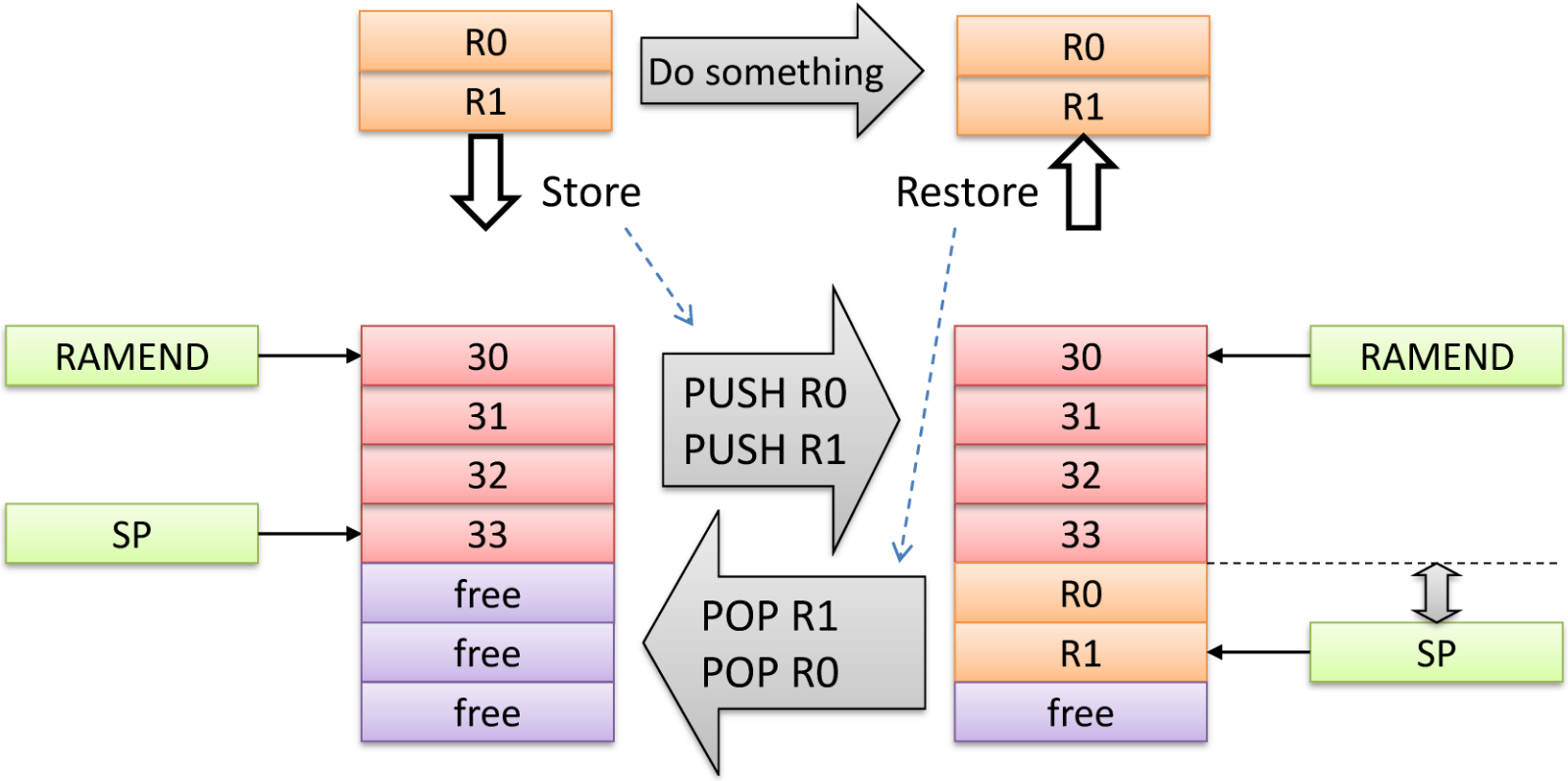
Instruction Set (5)

- Complex/Reduced Instruction Set Computing
  - CISC
    - Complex instructions
    - Many instructions
    - Few registers
    - Many cycles per instruction
    - No pipelines
    - Example
      - INTEL x86

  - RISC
    - Simple instructions
    - Few instructions
    - Many registers
    - One cycle per instruction
    - Pipelining
    - Example
      - SUN SPARC

Stack (1)

- Operating mode

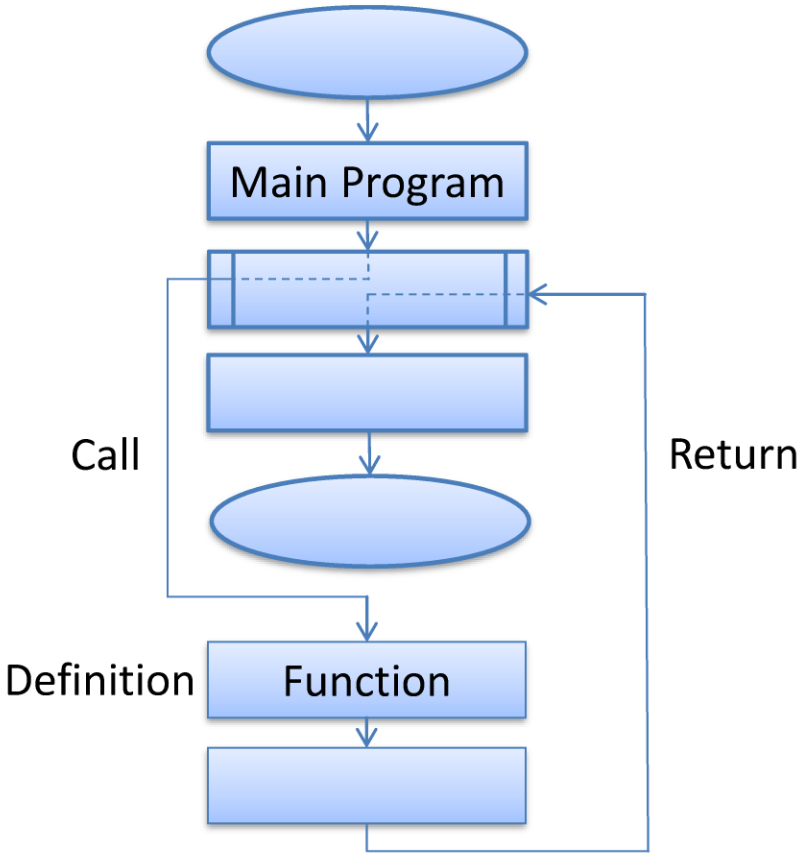


Stack (2)

- Summary
  - Used to store temporary data.
  - Is a last-in-first-out (LIFO) memory.
  - Grows from up to down.
  - Stack pointer points to the last element.
  - Two special instructions
    - PUSH: Decrement Stack pointer, put element on Stack.
    - POP: Get element from Stack, increment Stack pointer.
    - PUSH/POP: Pre-decrement / post-increment.

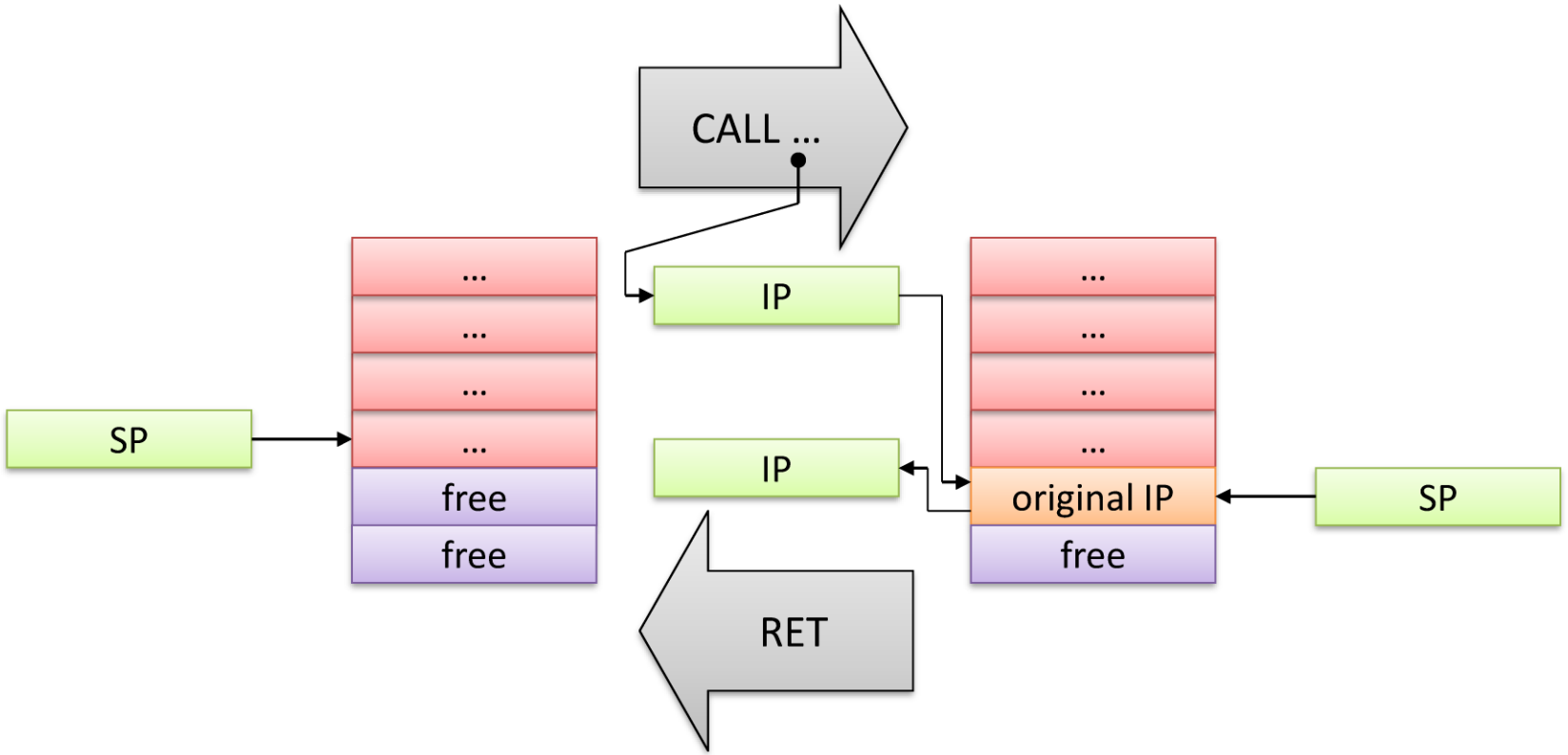
Stack (3)

- Application: functions
  - Are defined and called
  - Alter sequential control flow
  - Return to caller
    - Many callers possible
    - Return address has to be stored



Stack (4)

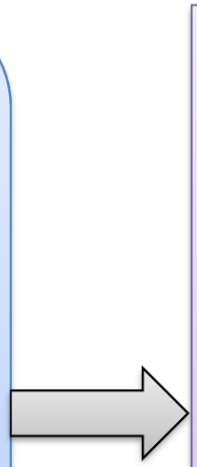
- Execution

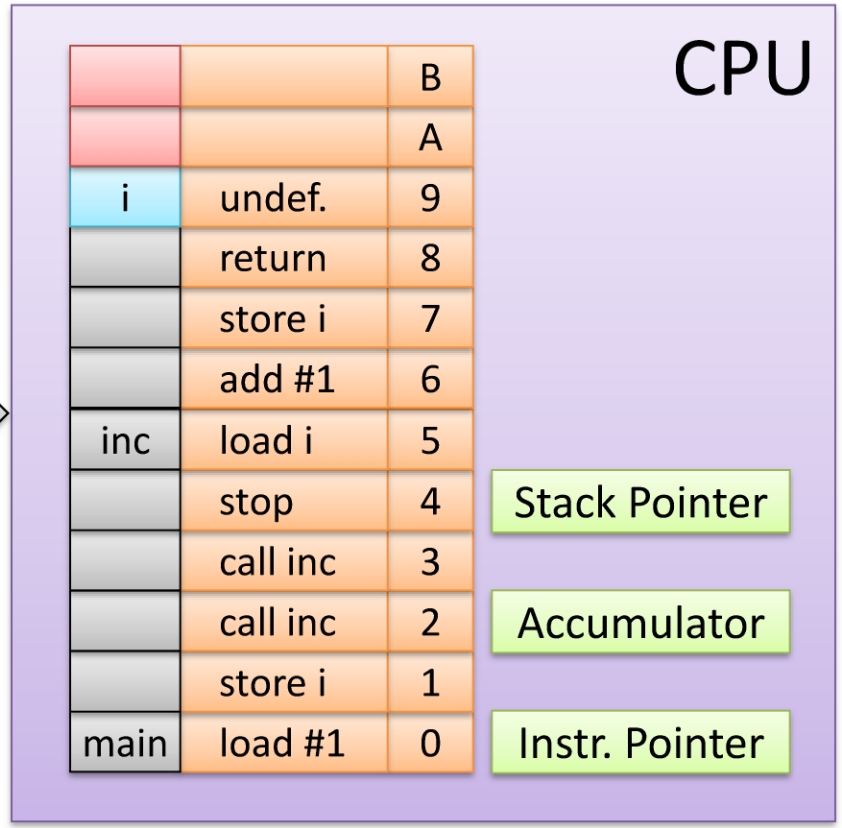


Stack (5)

- Compilation

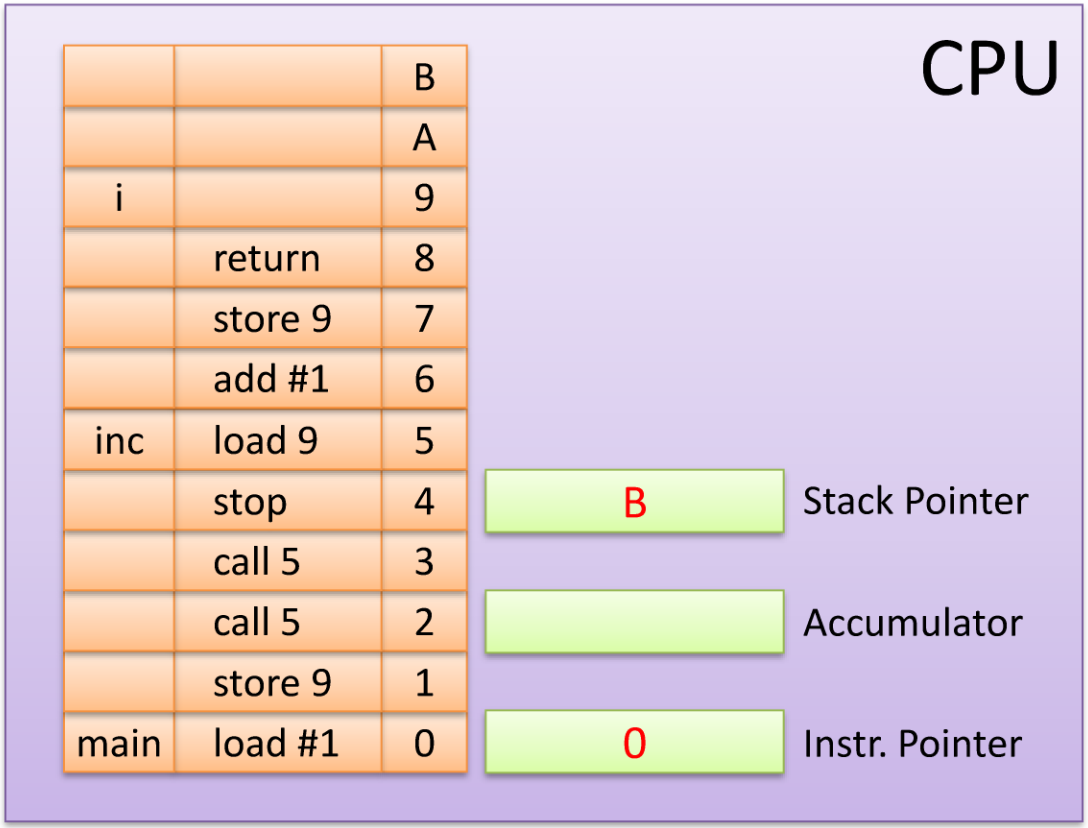
```
int i;

void inc(void)
{
   i = i+1;
}

int main(void)
{
   i = 1;
   inc();
   inc();
}
```





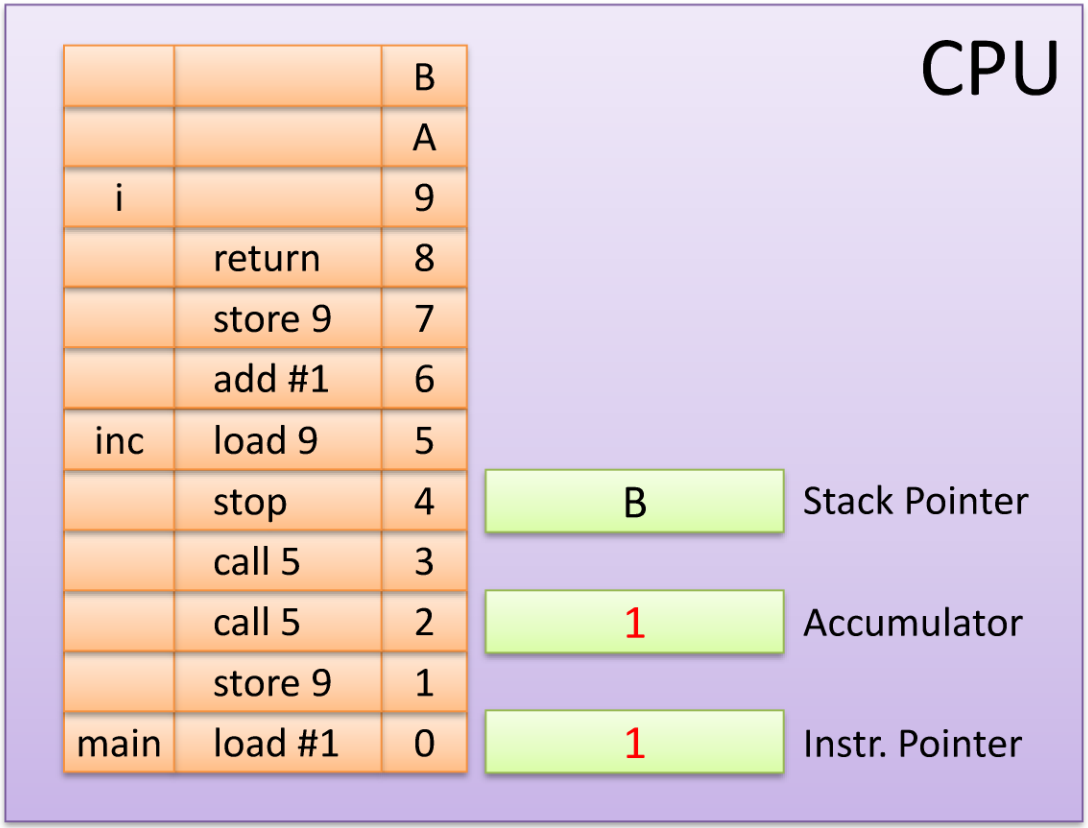
Stack (6)

- Execution



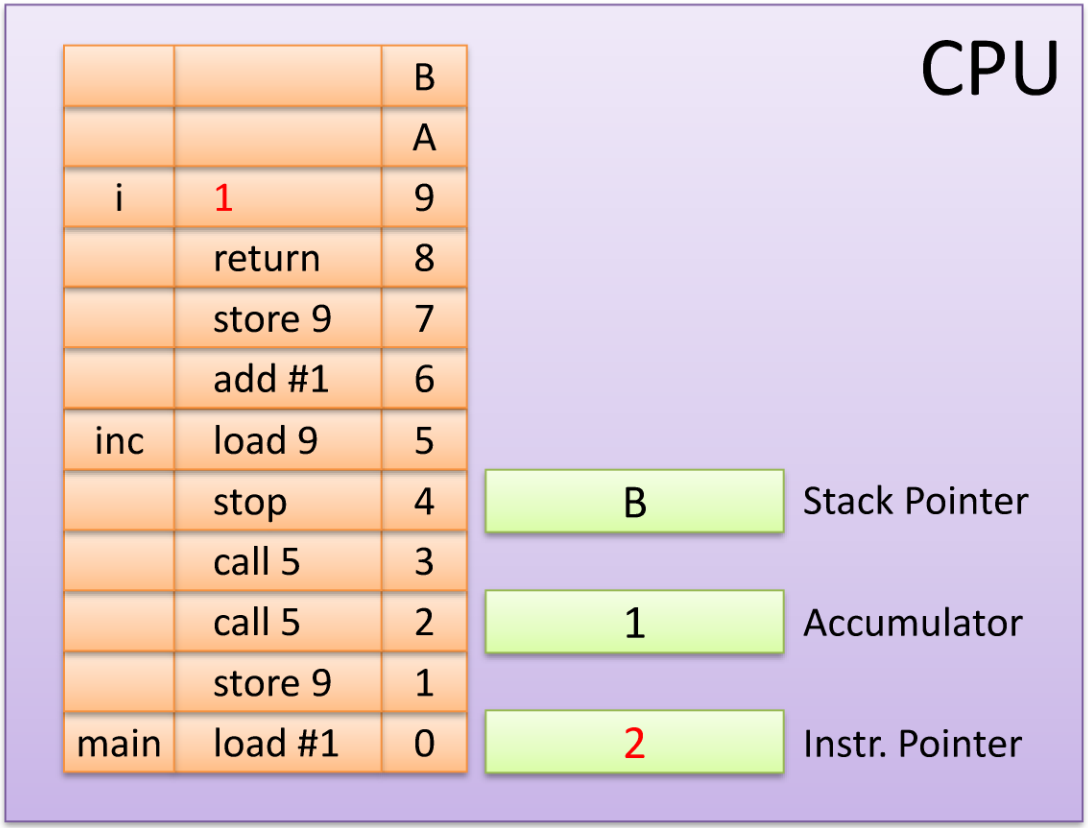
Stack (6)

- Execution (continued)



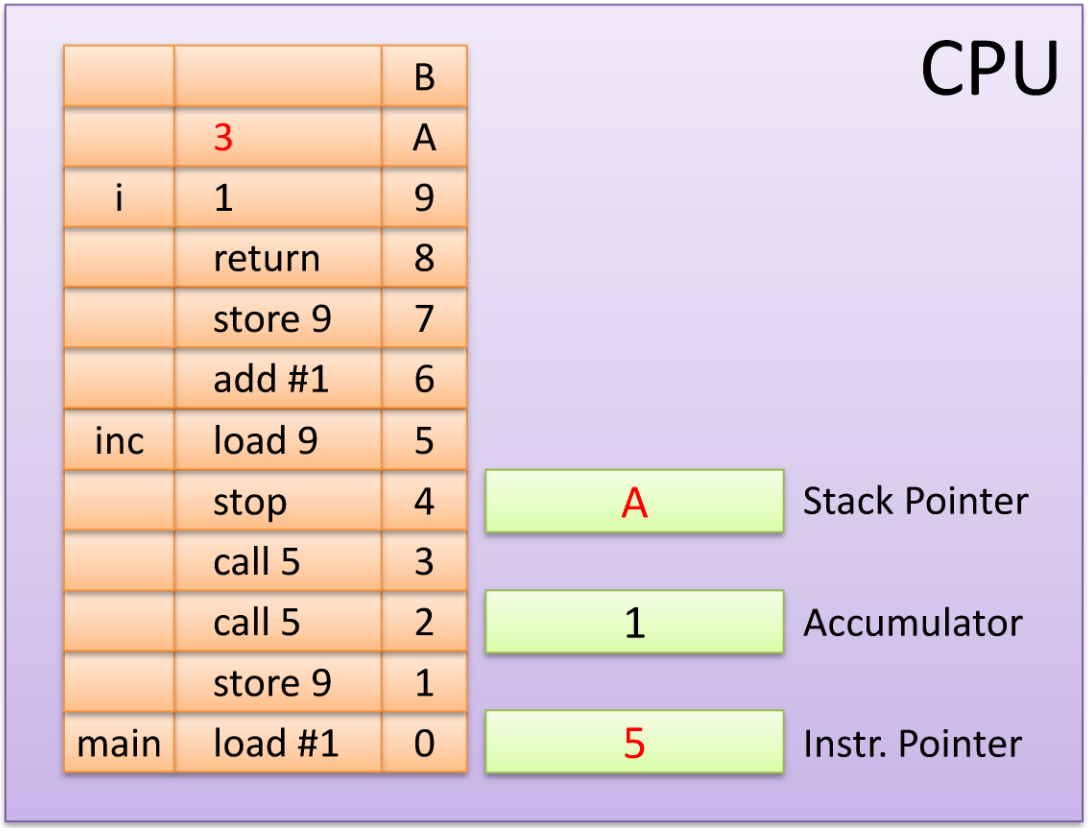
Stack (6)

- Execution (continued)



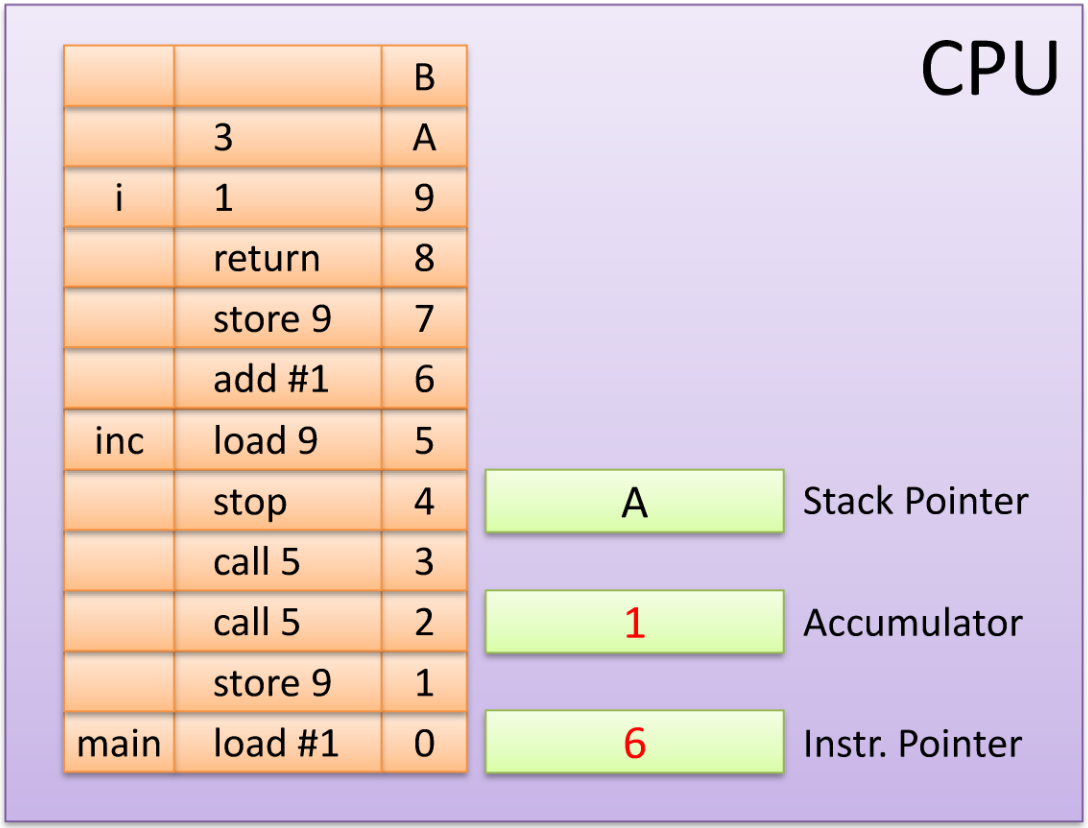
Stack (6)

- Execution (continued)



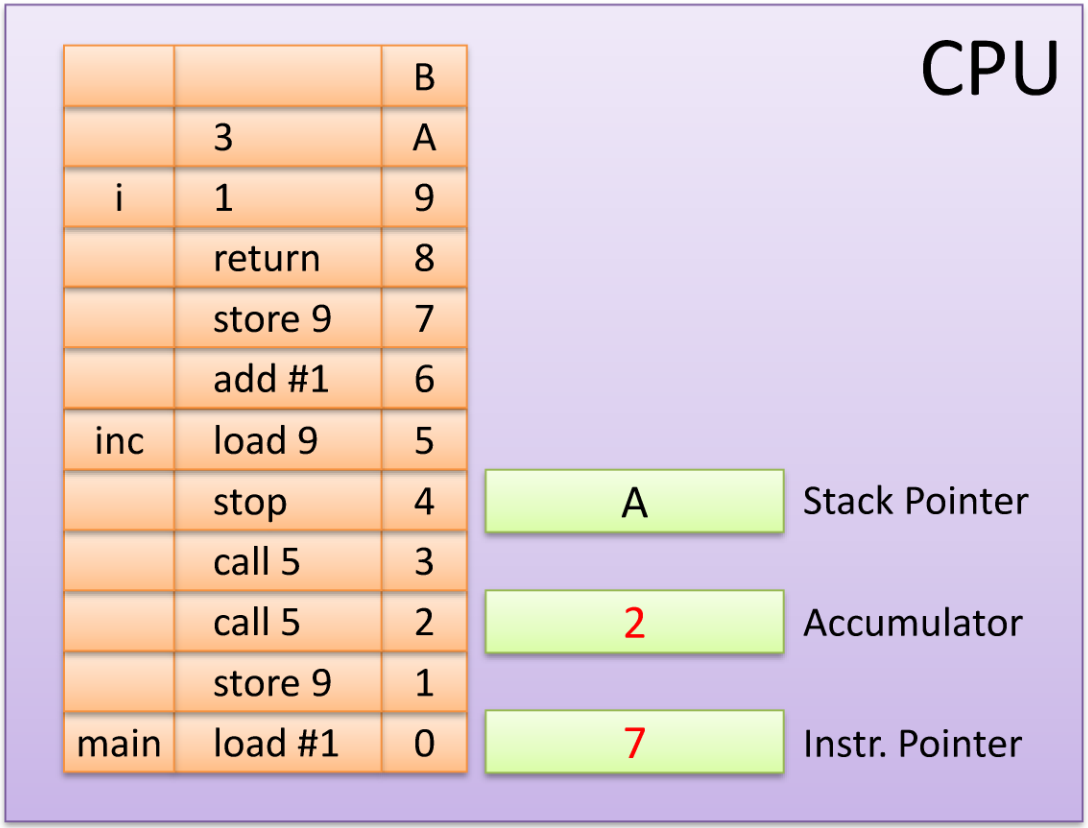
Stack (6)

- Execution (continued)



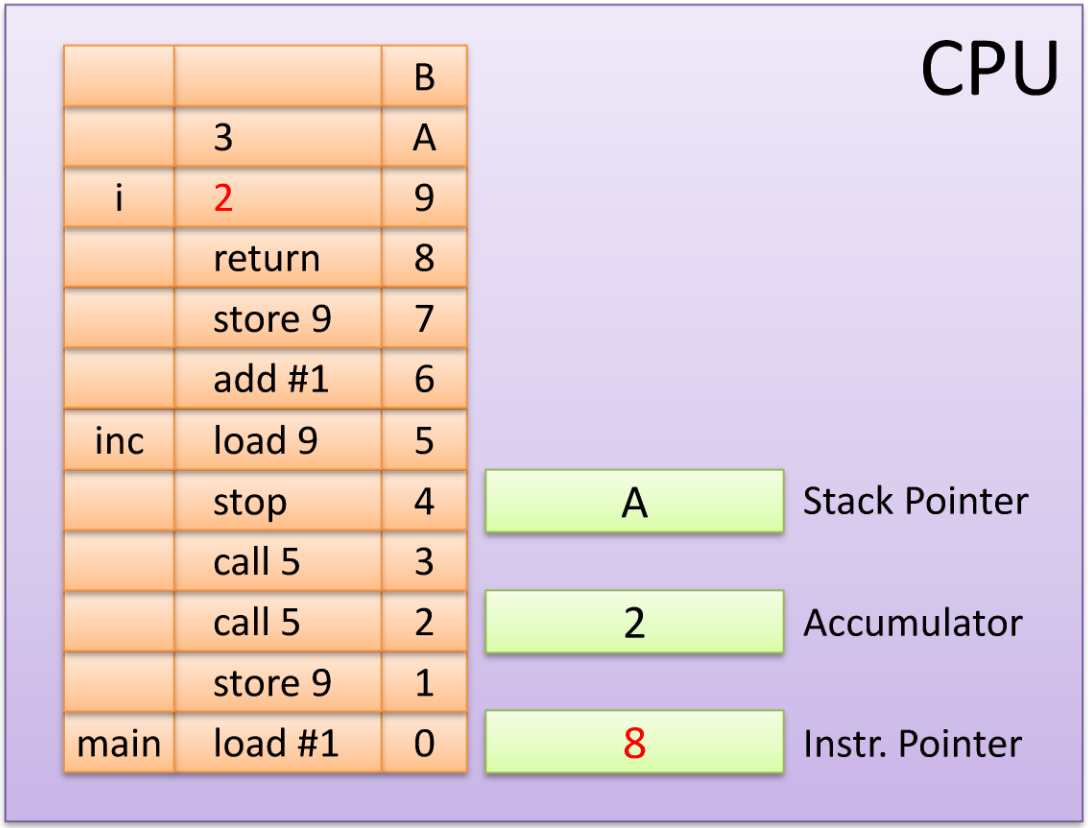
Stack (6)

- Execution (continued)



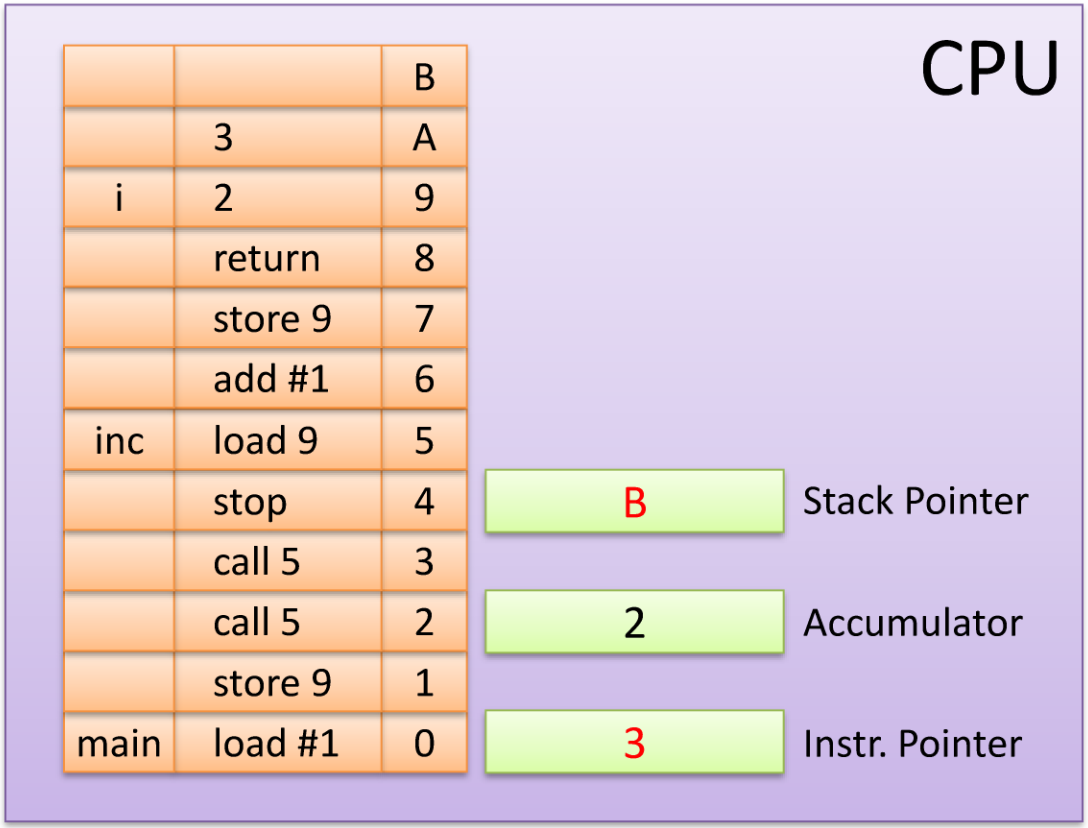
Stack (6)

- Execution (continued)



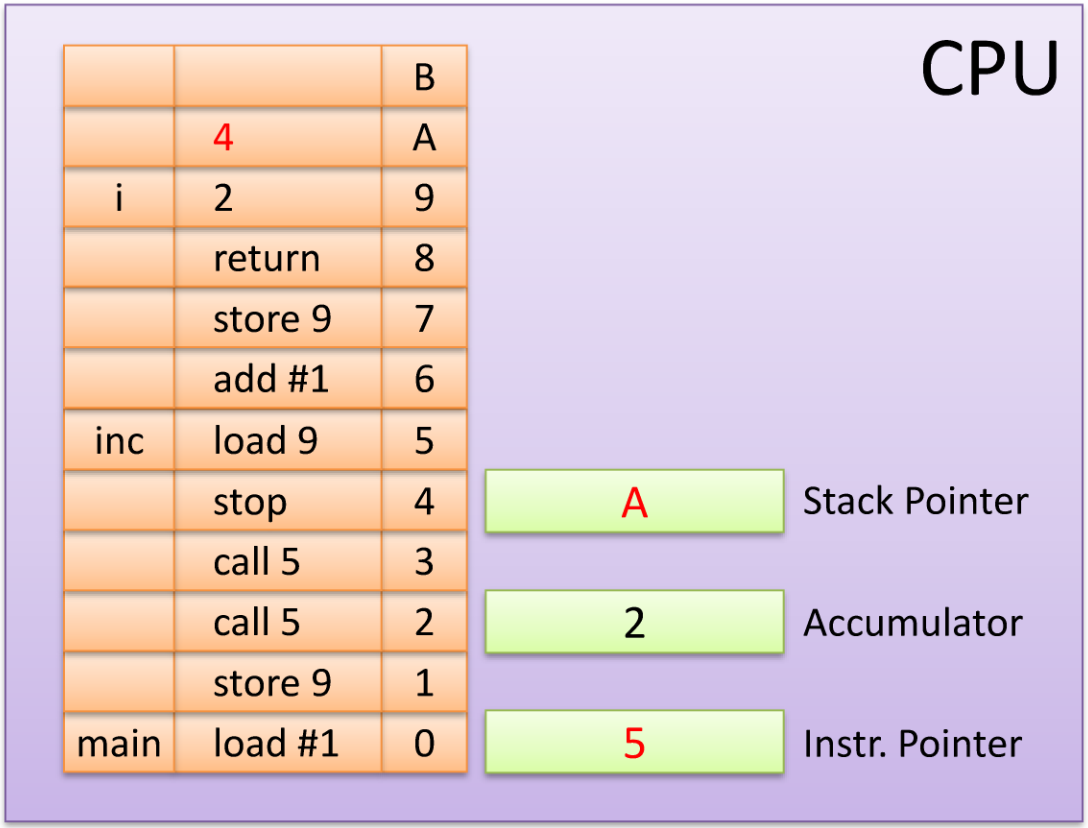
Stack (6)

- Execution (continued)



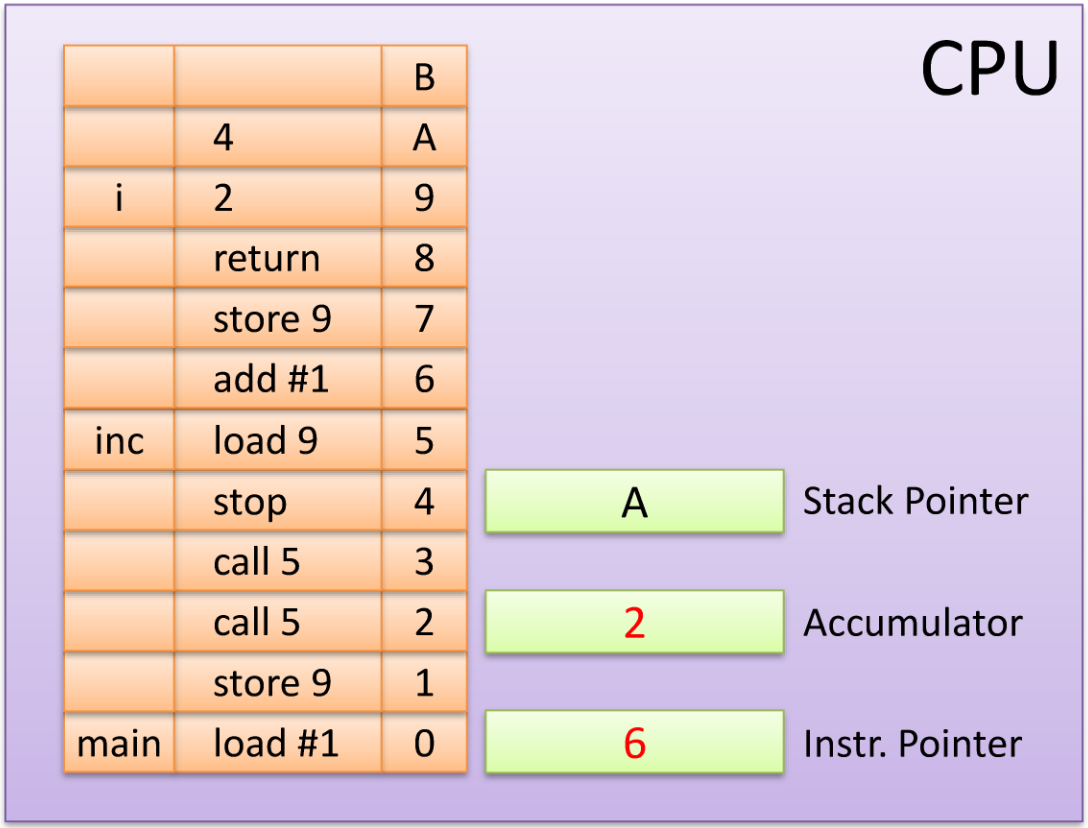
Stack (6)

- Execution (continued)



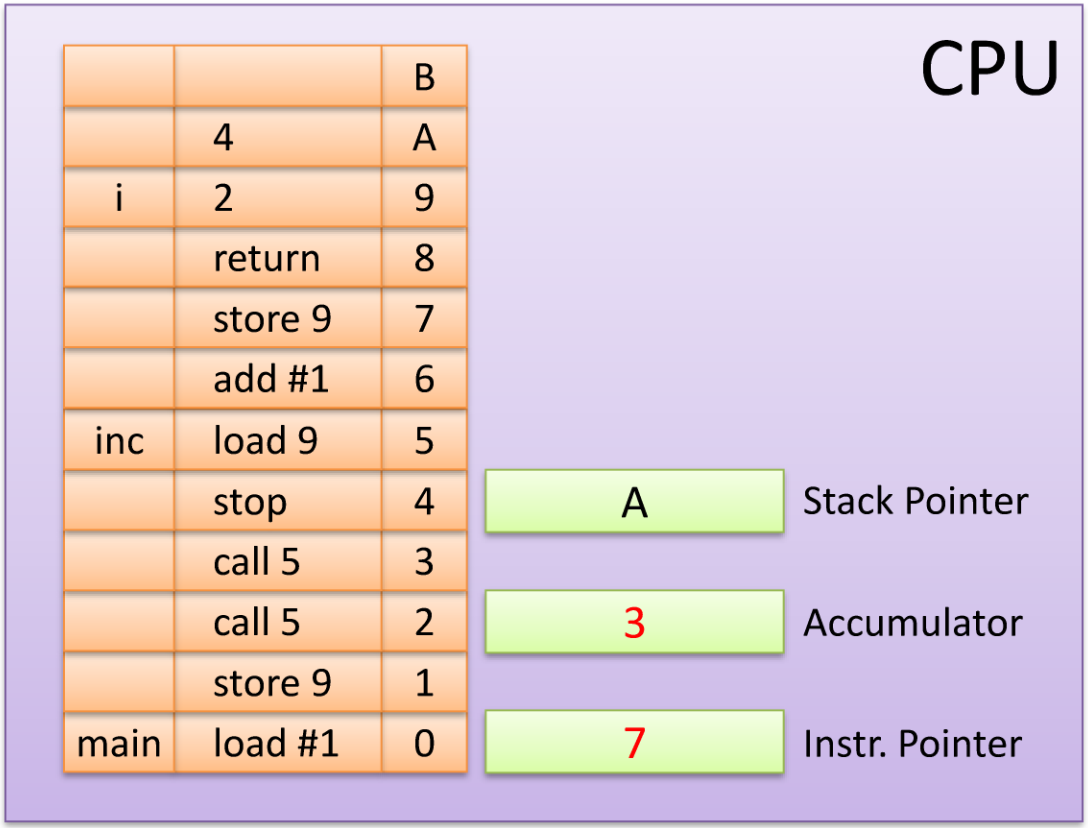
Stack (6)

- Execution (continued)



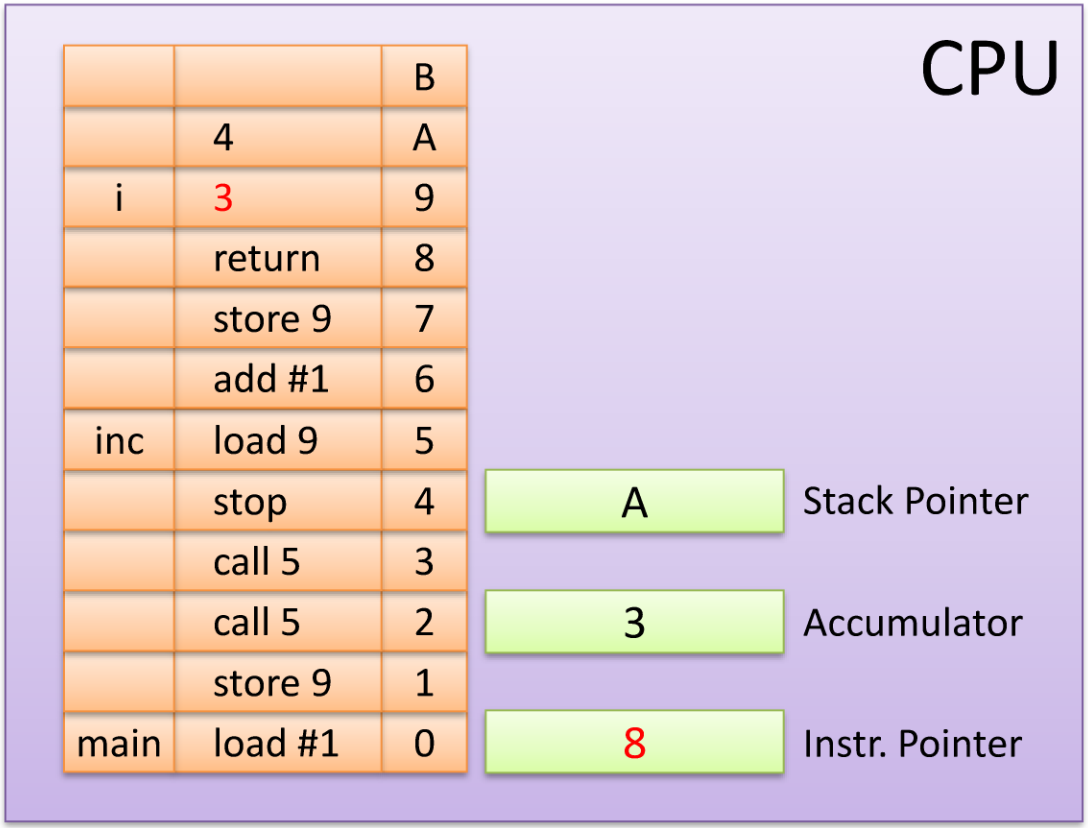
Stack (6)

- Execution (continued)



Stack (6)

- Execution (continued)



Stack (6)

- Execution (finished)