Compilers

Arthur Hoskey, Ph.D.
Farmingdale State College
Computer Systems Department

Today’s Lecture

- Computer Architecture and Assembler Overview

<u>Von Neumann Architecture</u>

- Standard architecture for most computers today.

- John von Neumann developed it in the late 1940’s.

Major guidelines for Von Neumann Architecture:

- Memory holds both data and programs.
- Memory is addressed linearly.
- Memory is addressed by the location number without regard to the data contained within.

Von Neumann Architecture

Von Neumann Architecture

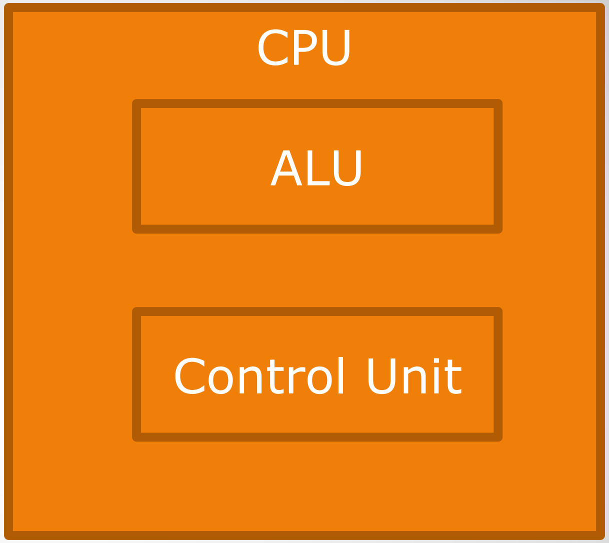
Von Neumann also defined functional organization of a computer to be made up of the following:

- Control unit – Executes instructions.
- Arithmetic/Logic unit (ALU) – Performs arithmetic and logical calculations.
- Memory (RAM)

CPU = Control Unit + ALU

CPU

CPU =
Control Unit
+ ALU



CPU

ALU

Control Unit

Note: There are some details that are left out, but this is the basic setup.

CPU

Control Unit

- Controls and interprets the execution of instructions.

- Follows a sequence of actions that correspond to the fetch-execute instruction cycle.
  - Get instruction from memory
  - Move data and addresses from one part of the CPU to another.

CPU – Control Unit

Arithmetic Logic Unit (ALU)

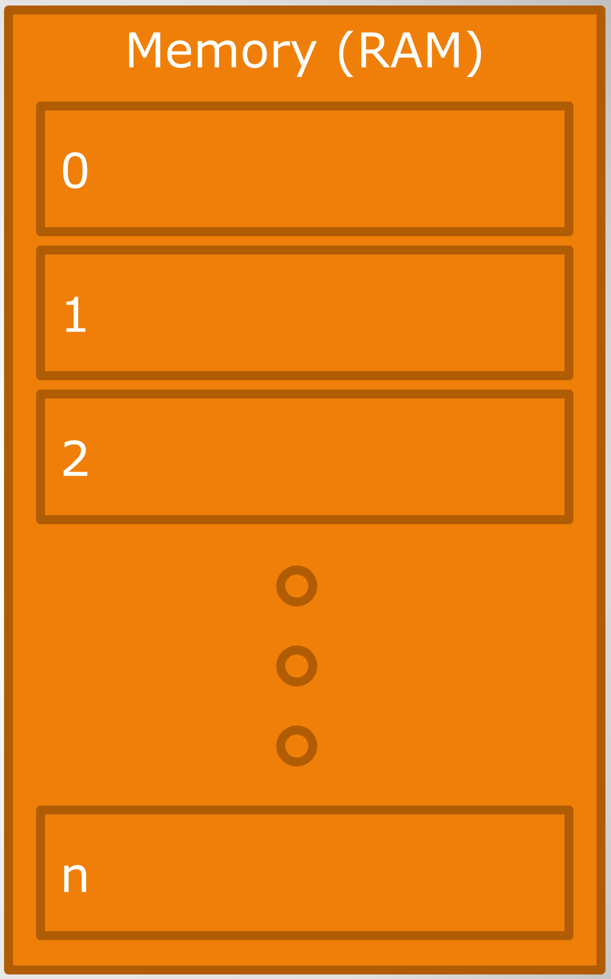
- Calculations take place here.

- Works as follows:
  - Data gets moved into the ALU (into ALU temporary storage).
  - Calculations are performed.
  - Result data is moved out of the ALU to register(s).

CPU – Arithmetic Logic Unit

Memory

Memory is linear and starts from address 0



Memory (RAM)

0

1

2

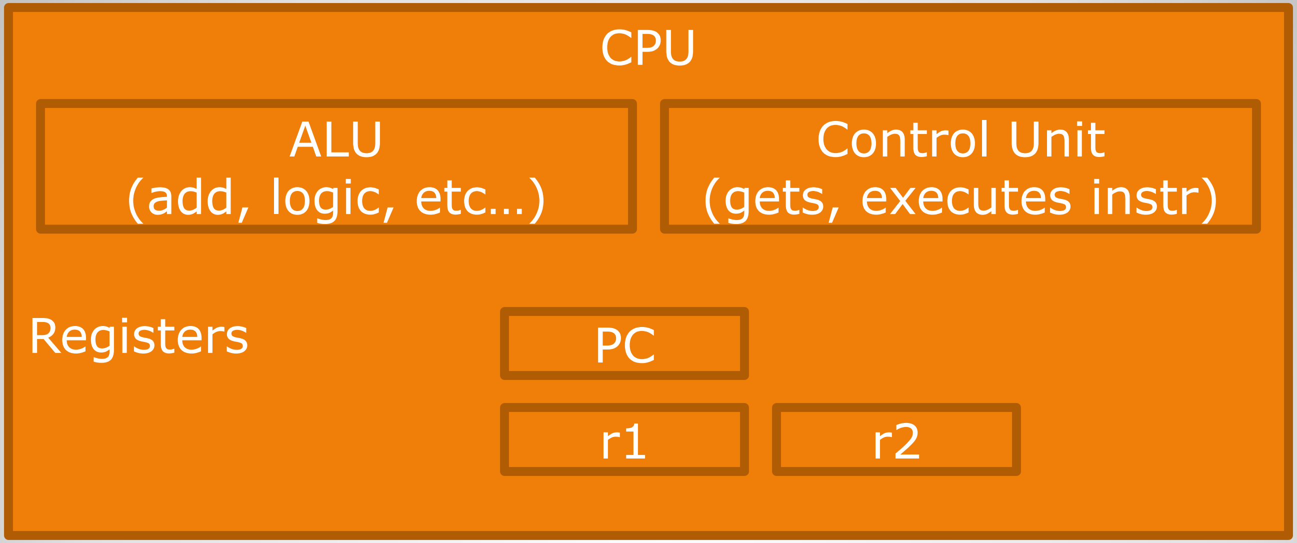
n

Register

- Single permanent storage location within the CPU.
- Each register usually has a defined purpose (dependent on the particular CPU).
- For example:
  - Program counter register (PC).
  - Holds the address of the current instruction being processed.
- General registers can be used for anything. We will use r1, r2, etc. to refer to general registers.

Register

CPU Showing Some Registers



CPU

ALU
(add, logic, etc…)

Control Unit
(gets, executes instr)

Registers

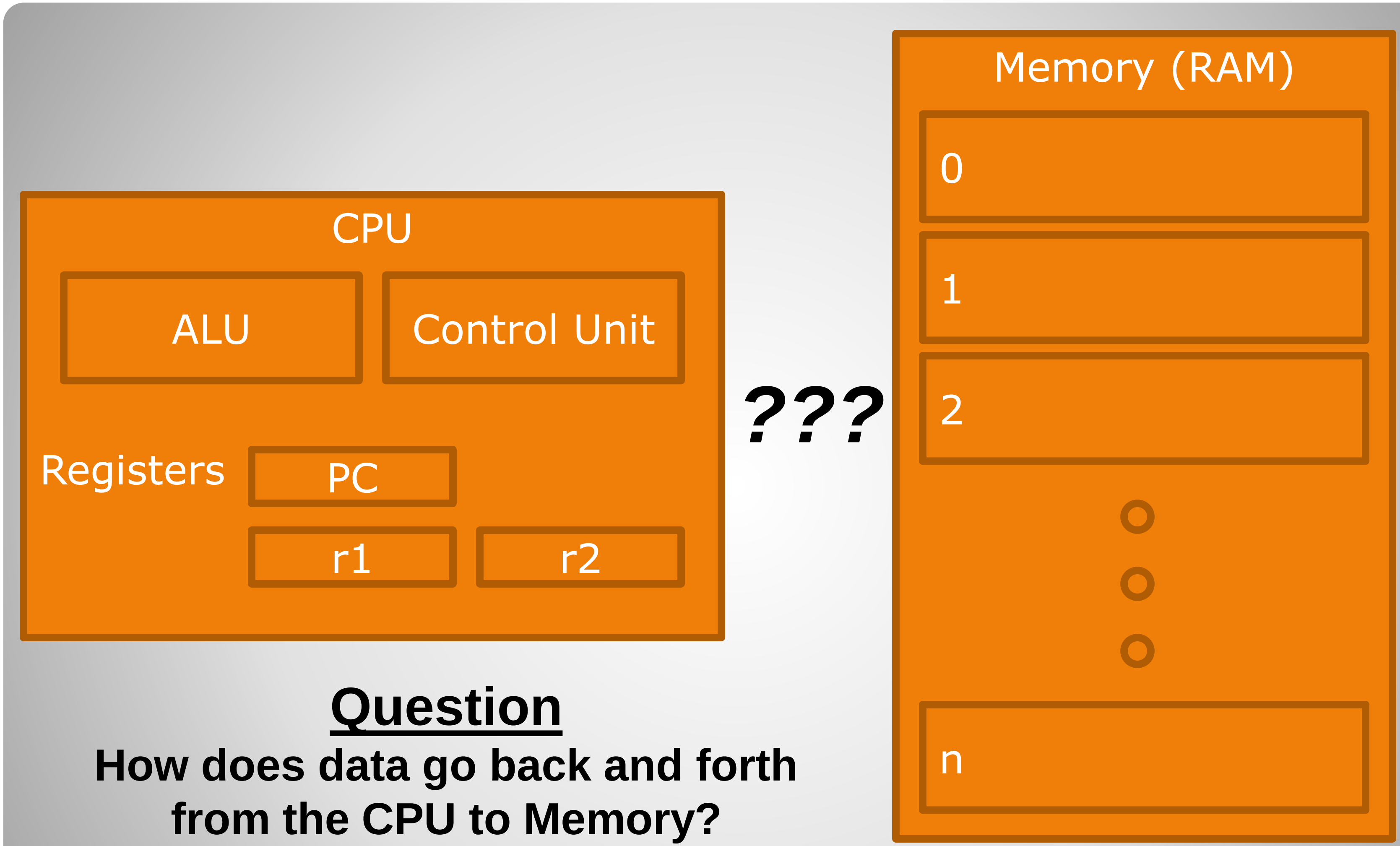
PC

r1

r2

- PC – Program Counter. Address of the current instruction
- r1 – General purpose register.
- r2 – General purpose register.
- Note: CPUs differ on the number of registers they contain as well as the names of those registers.

CPU Showing Some Registers

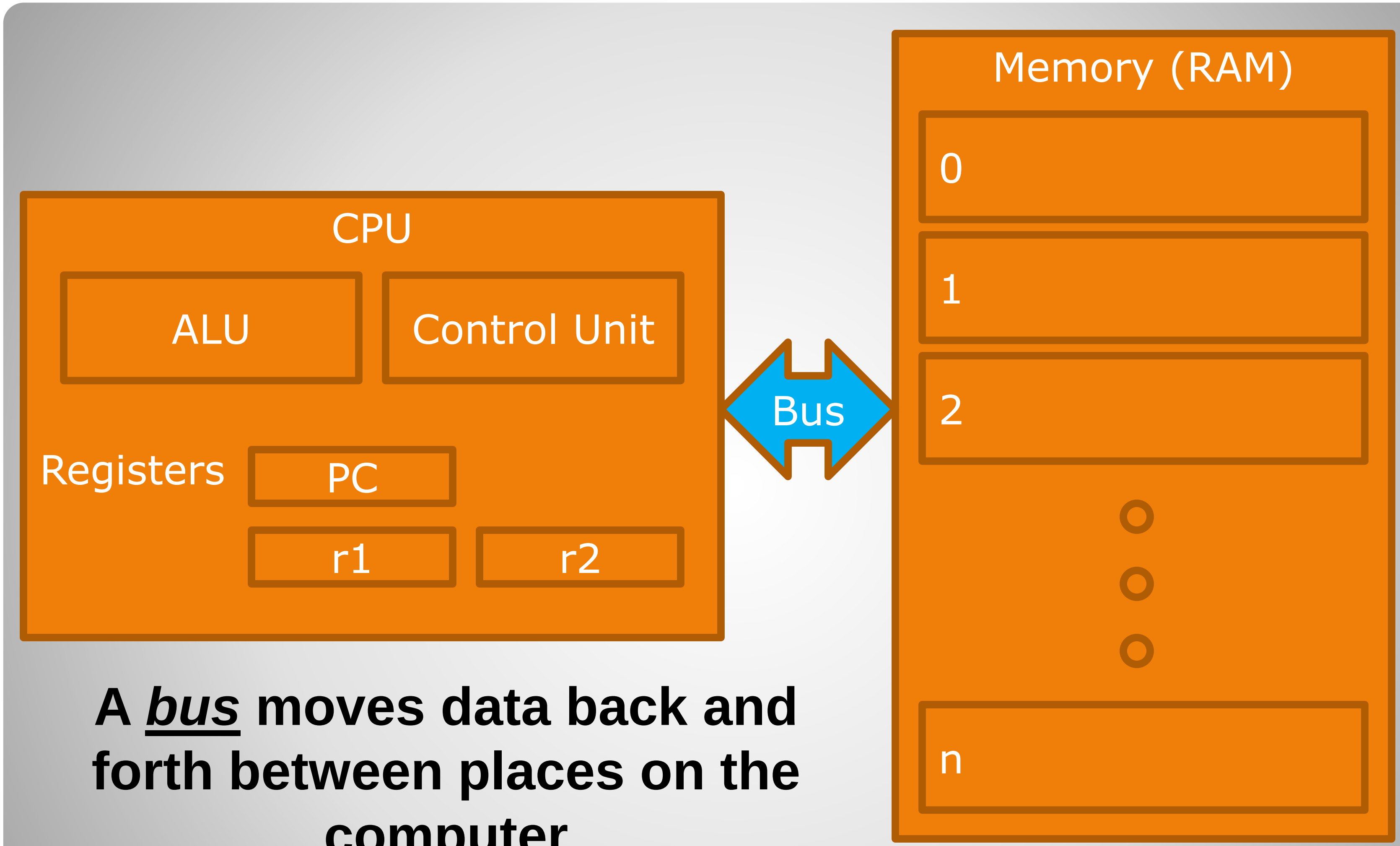


Question

How does data go back and forth from the CPU to Memory?

Von Neumann Architecture

Von Neumann Architecture



A *bus* moves data back and forth between places on the computer

Bus

- Bus – A group of electrical conductors suitable for carrying computer signals from one location to another.

- The bus is part of the motherboard.

- Used to move "data" around the computer.

Bus

- Now we will go over what happens when some assembly instructions run.

Assembly Language

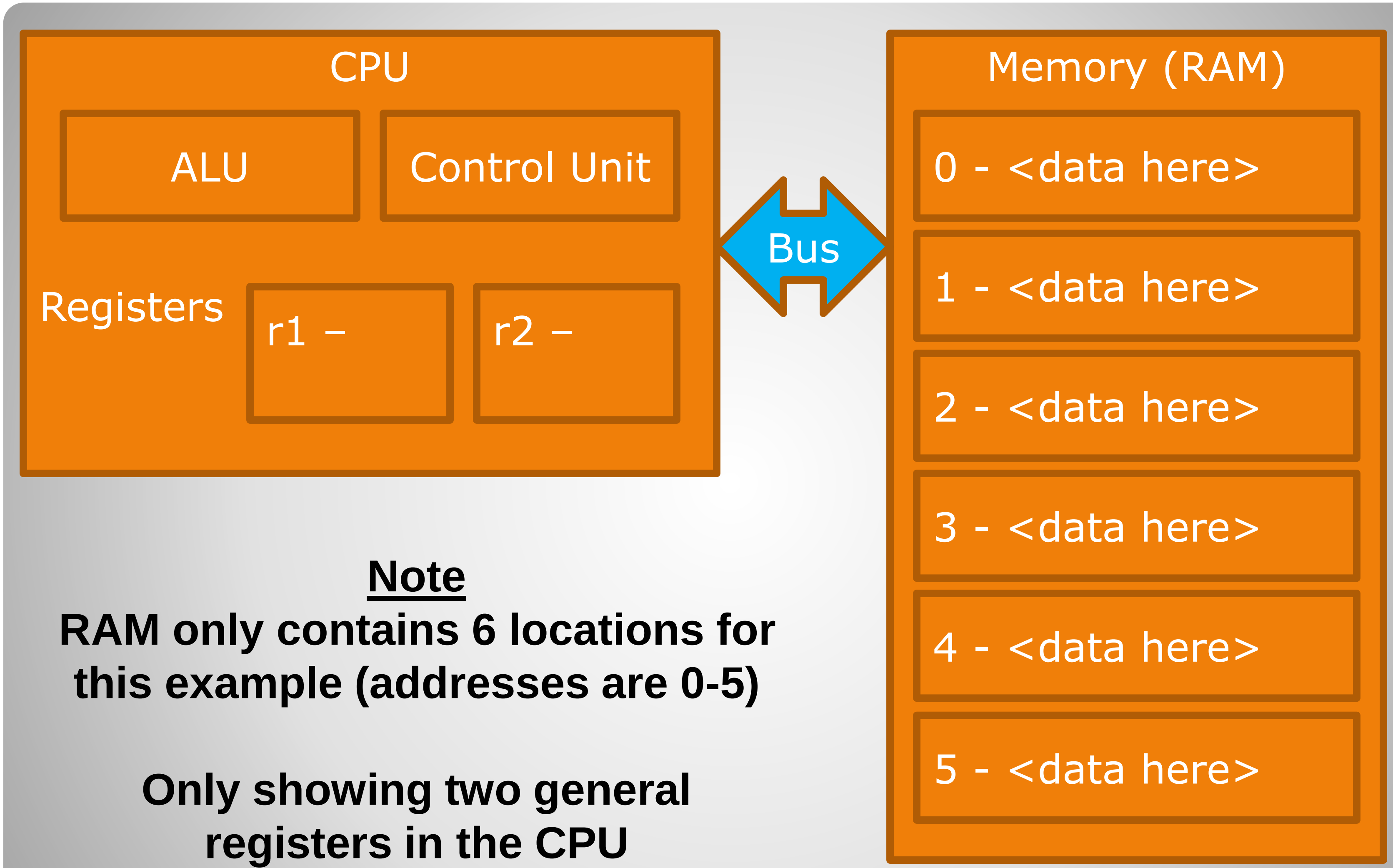
Load

- Loads a piece of data from memory into a register.
- General format of load:
  - load <register>, <memory location>

- Here is a load instruction that will get data from memory address 3 and put it in register r2:
  - load r2, 3

Load Instruction

System Diagram



CPU

ALU

Control Unit

Registers

r1 –

r2 –

Bus

Memory (RAM)

0 - <data here>

1 - <data here>

2 - <data here>

3 - <data here>

4 - <data here>

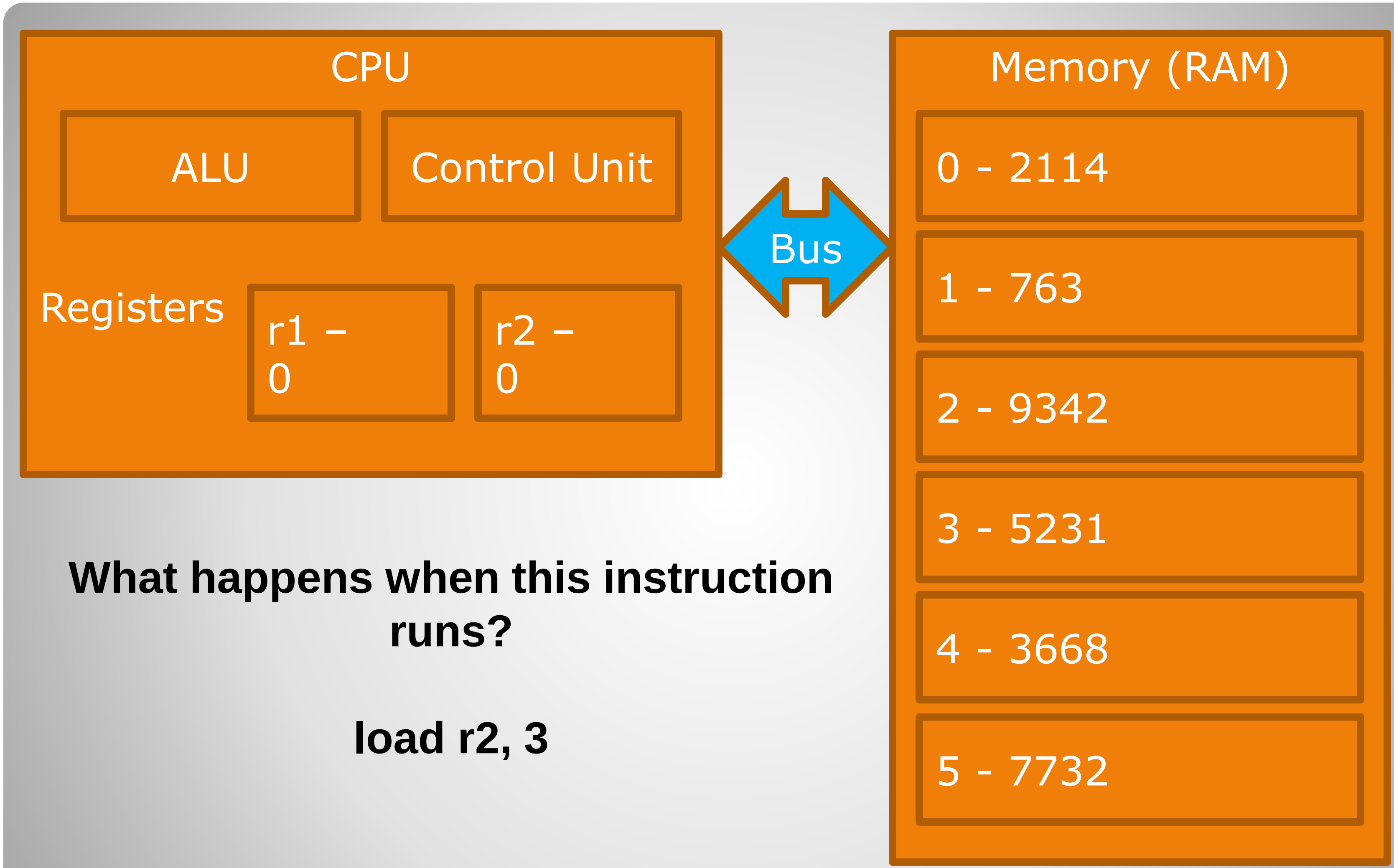
5 - <data here>

<u>Note</u>

RAM only contains 6 locations for this example (addresses are 0-5)

Only showing two general registers in the CPU

Load Instruction



CPU

ALU

Control Unit

Registers

r1 – 0

r2 – 0

Bus

Memory (RAM)

0 - 2114

1 - 763

2 - 9342

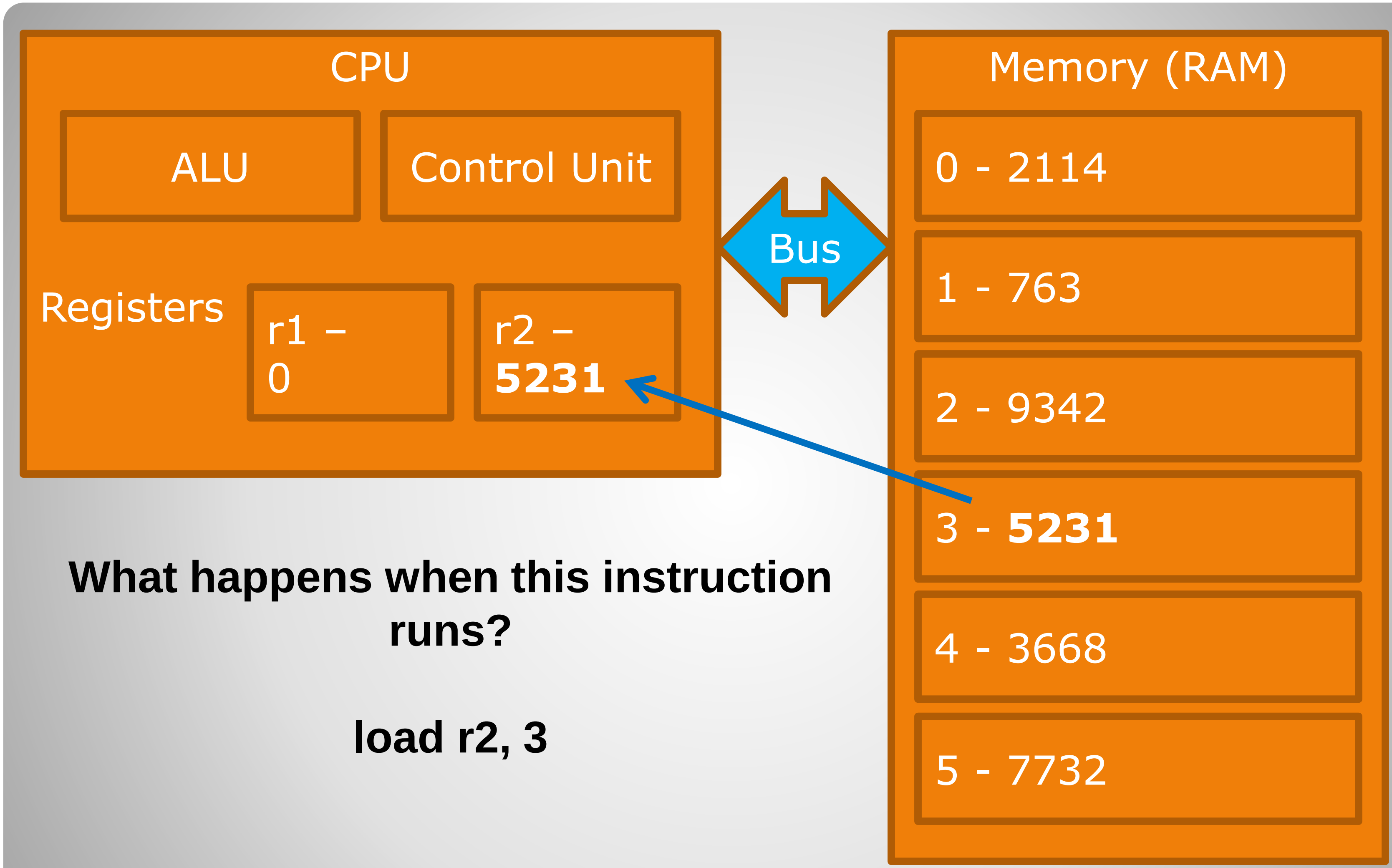
3 - 5231

4 - 3668

5 - 7732

What happens when this instruction runs?

load r2, 3



What happens when this instruction runs?

load r2, 3

Load Instruction

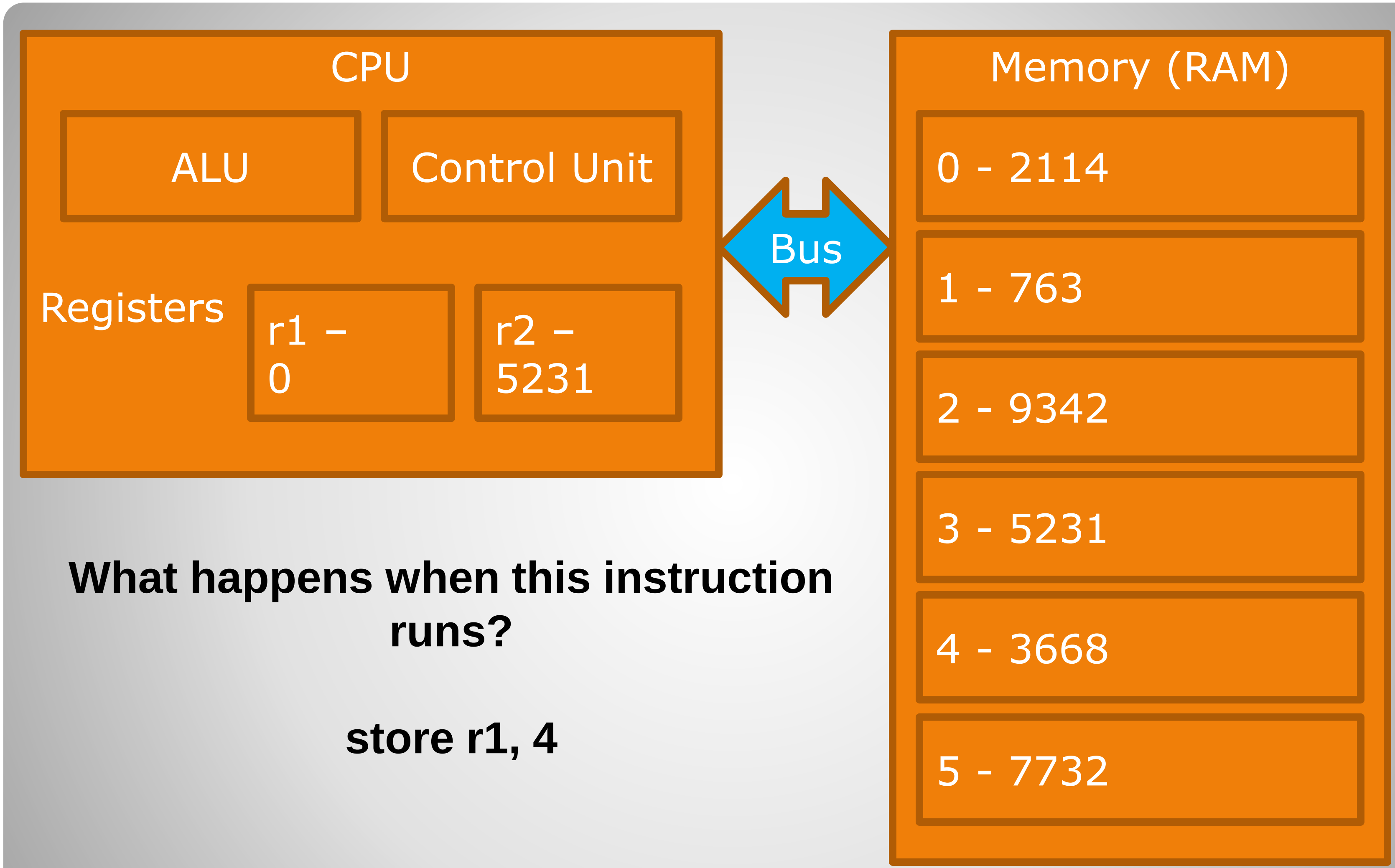
Store

- Stores a piece of data from a register into memory.
- General format of load:
  - store <register>, <memory location>

- Here is a load instruction that will get data from register r1 and put it in memory address 4:
  - store r1, 4

Store Instruction

Store Instruction



CPU

ALU

Control Unit

Registers

r1 – 0

r2 – 5231

Bus

Memory (RAM)

0 - 2114

1 - 763

2 - 9342

3 - 5231

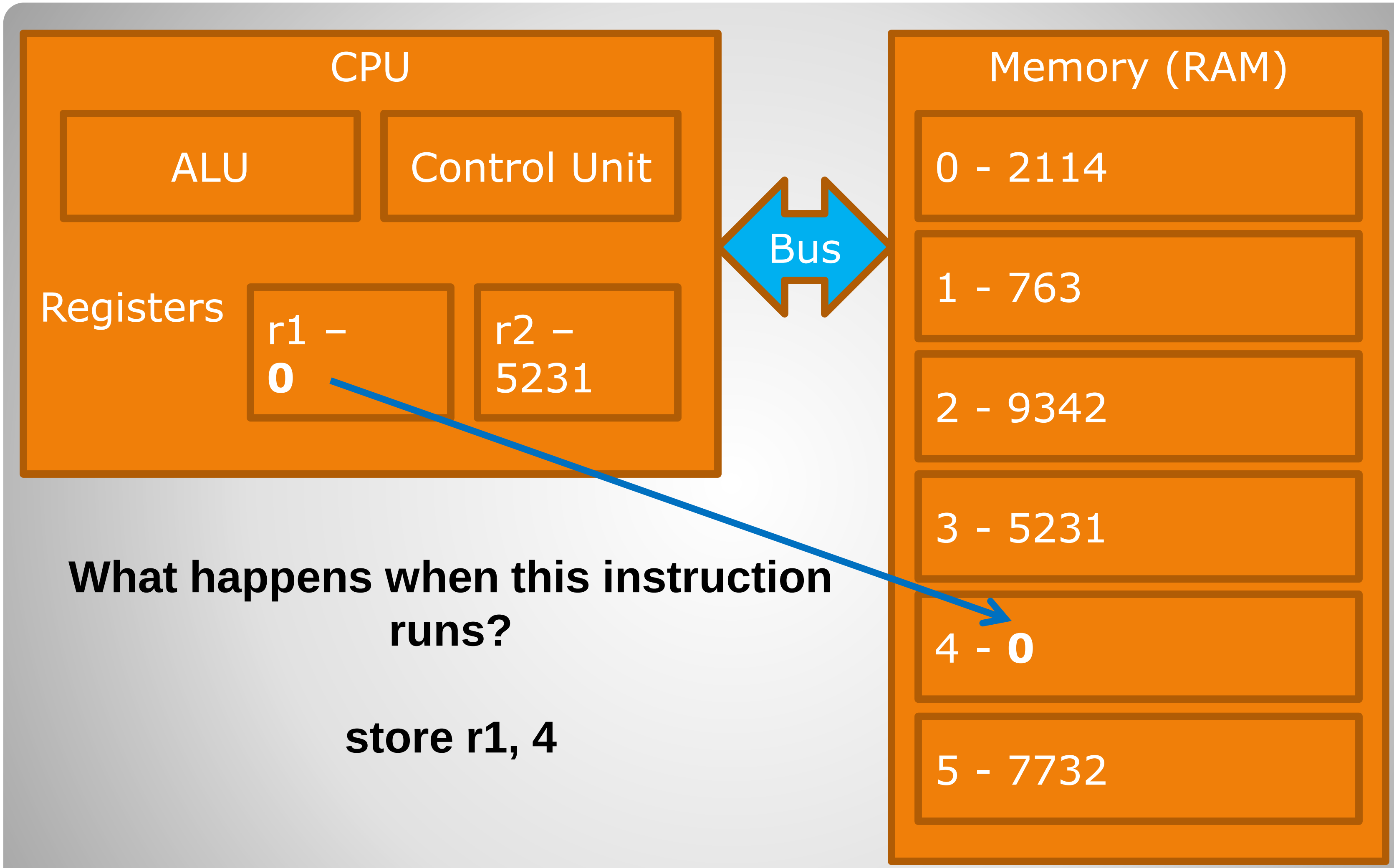
4 - 3668

5 - 7732

What happens when this instruction runs?

store r1, 4

Store Instruction



CPU

ALU

Control Unit

Registers

r1 – **0**

r2 – 5231

Bus

Memory (RAM)

0 - 2114

1 - 763

2 - 9342

3 - 5231

4 - **0**

5 - 7732

What happens when this instruction runs?

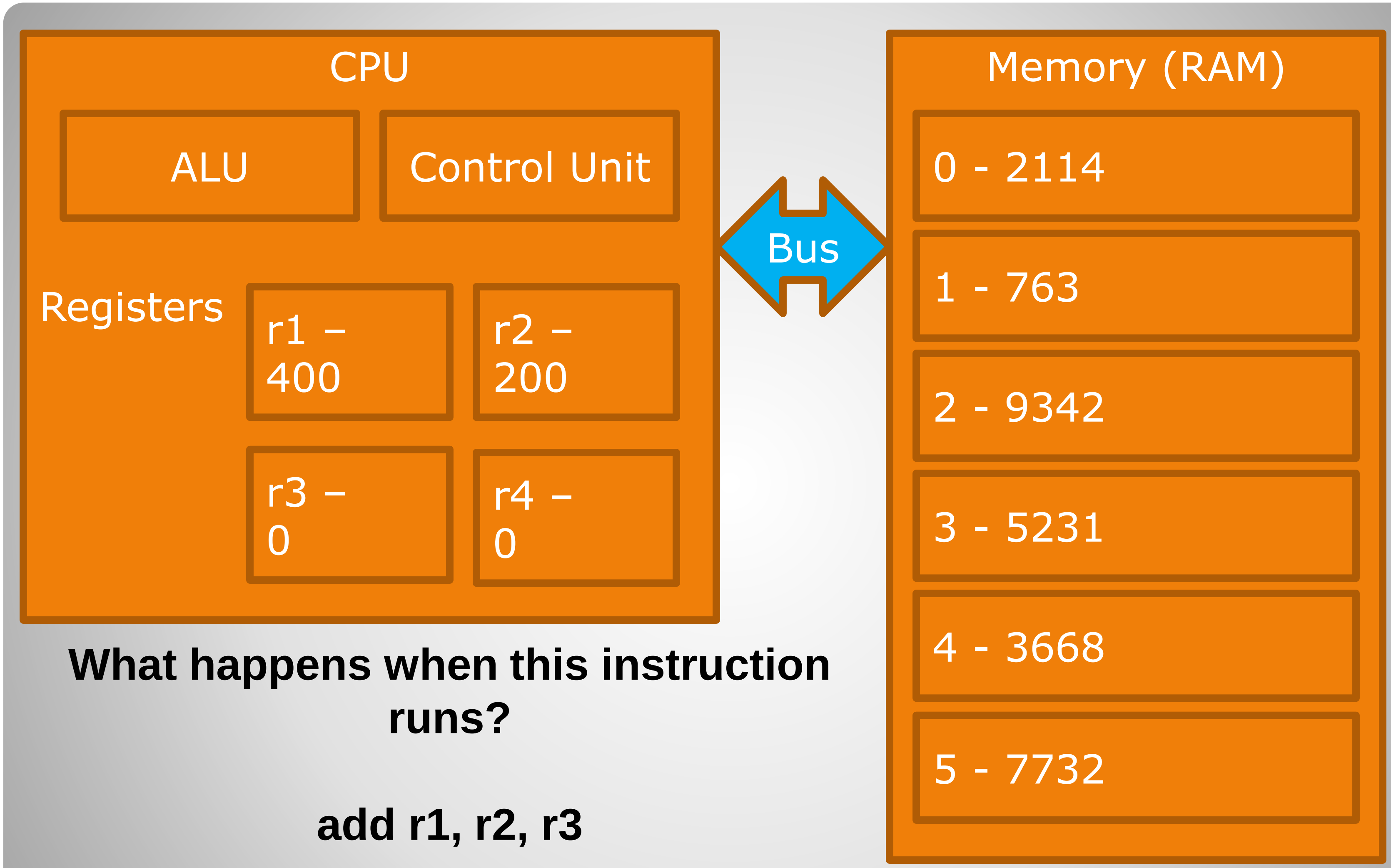
store r1, 4

Add

- Adds data from two registers and stores the result in a register.
- General format of load:
  - add <register>, <register>, <register>
- Here is an add instruction that will get data from registers r1 and r2 and put the result in register r3:
  - add r1, r2, r3

Add Instruction

Add Instruction



CPU

ALU

Control Unit

Registers

r1 – 400

r2 – 200

r3 – 0

r4 – 0

Bus

Memory (RAM)

0 - 2114

1 - 763

2 - 9342

3 - 5231

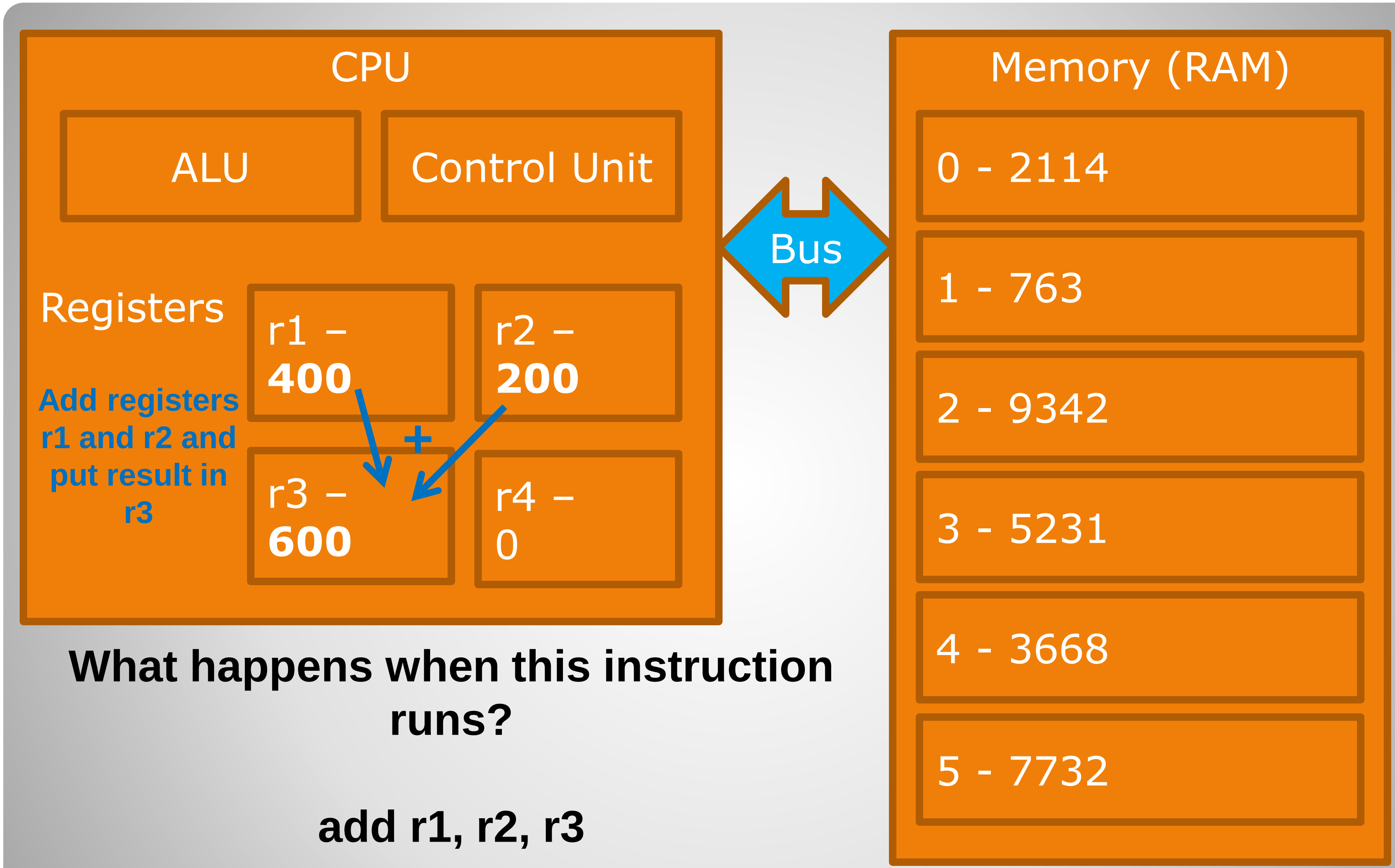
4 - 3668

5 - 7732

What happens when this instruction runs?

add r1, r2, r3

Add Instruction



CPU

ALU

Control Unit

Registers

r1 – **400**

r2 – **200**

r3 – **600**

r4 – 0

+

Add registers r1 and r2 and put result in r3

Bus

Memory (RAM)

0 - 2114

1 - 763

2 - 9342

3 - 5231

4 - 3668

5 - 7732

What happens when this instruction runs?

add r1, r2, r3

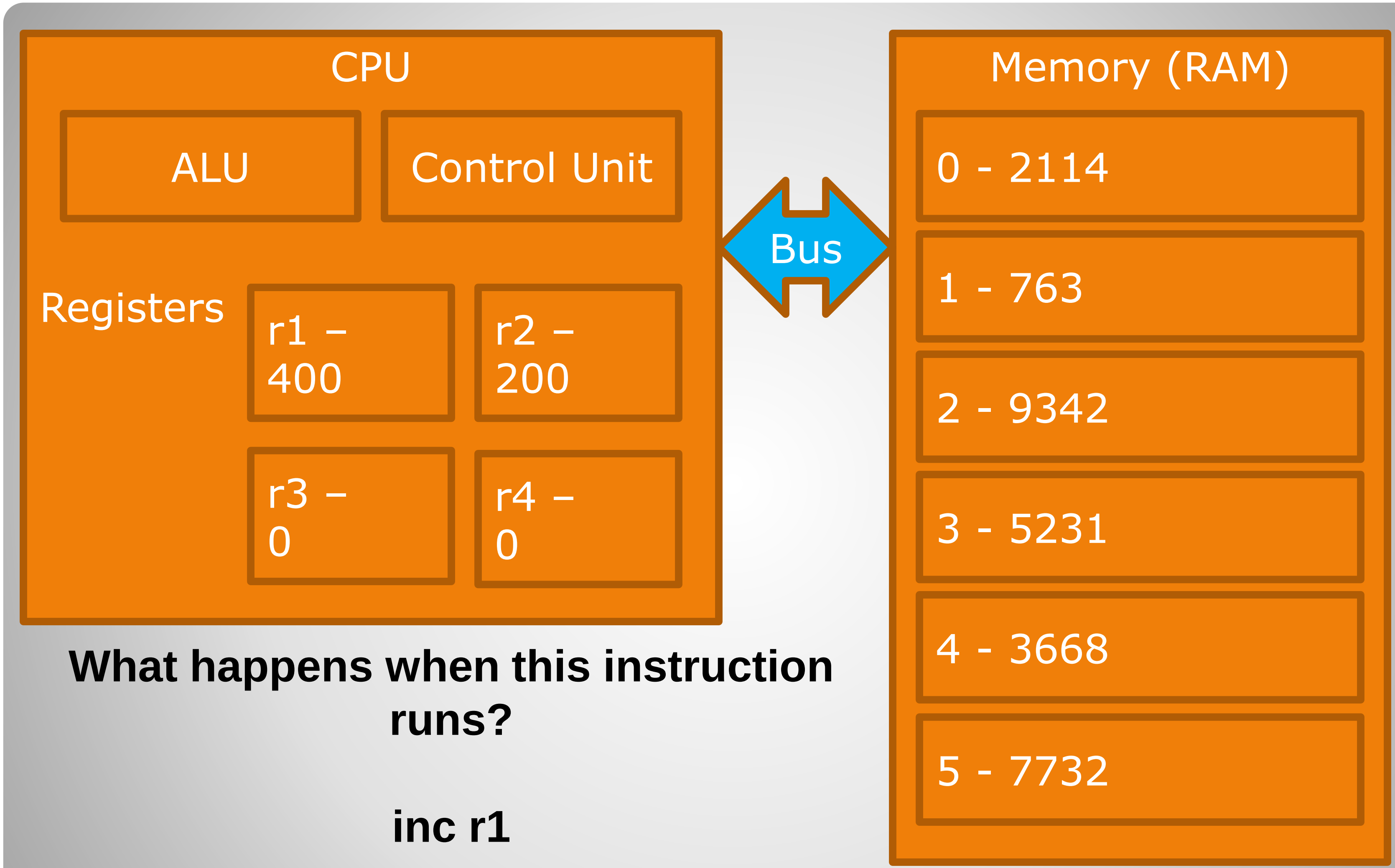
Inc Instruction

Inc

- Increments the value in a register (add 1 to the value in the register).
- General format of load:
  - inc <register>

- Here is an increment instruction for register r1:
  - inc r1

Inc Instruction



CPU

ALU

Control Unit

Registers

r1 – 400

r2 – 200

r3 – 0

r4 – 0

Bus

Memory (RAM)

0 - 2114

1 - 763

2 - 9342

3 - 5231

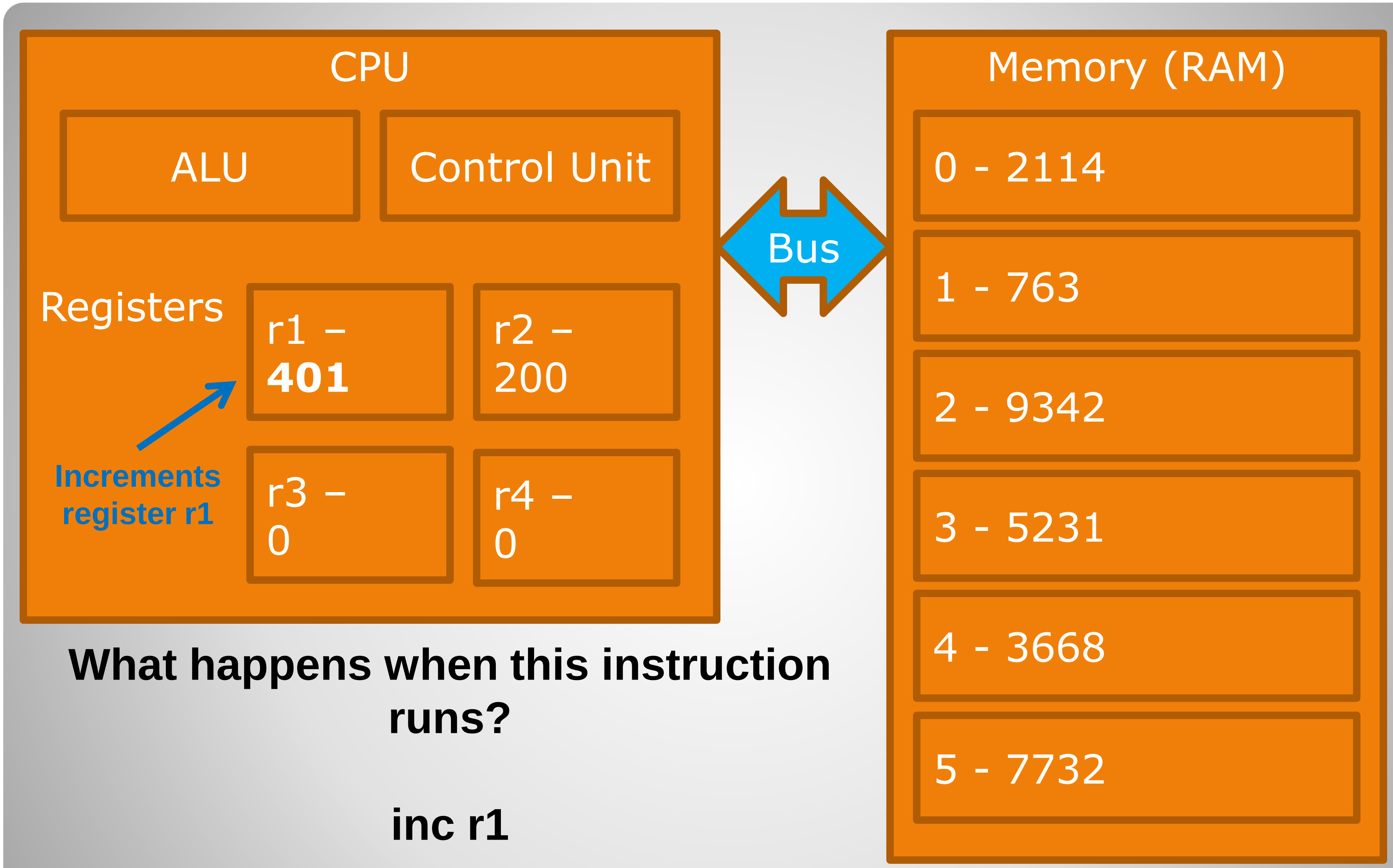
4 - 3668

5 - 7732

What happens when this instruction runs?

inc r1

Inc Instruction



CPU

ALU

Control Unit

Registers

r1 – **401**

r2 – 200

r3 – 0

r4 – 0

Increments register r1

Bus

Memory (RAM)

0 - 2114

1 - 763

2 - 9342

3 - 5231

4 - 3668

5 - 7732

What happens when this instruction runs?

inc r1

- **End of Slides**

End of Slides