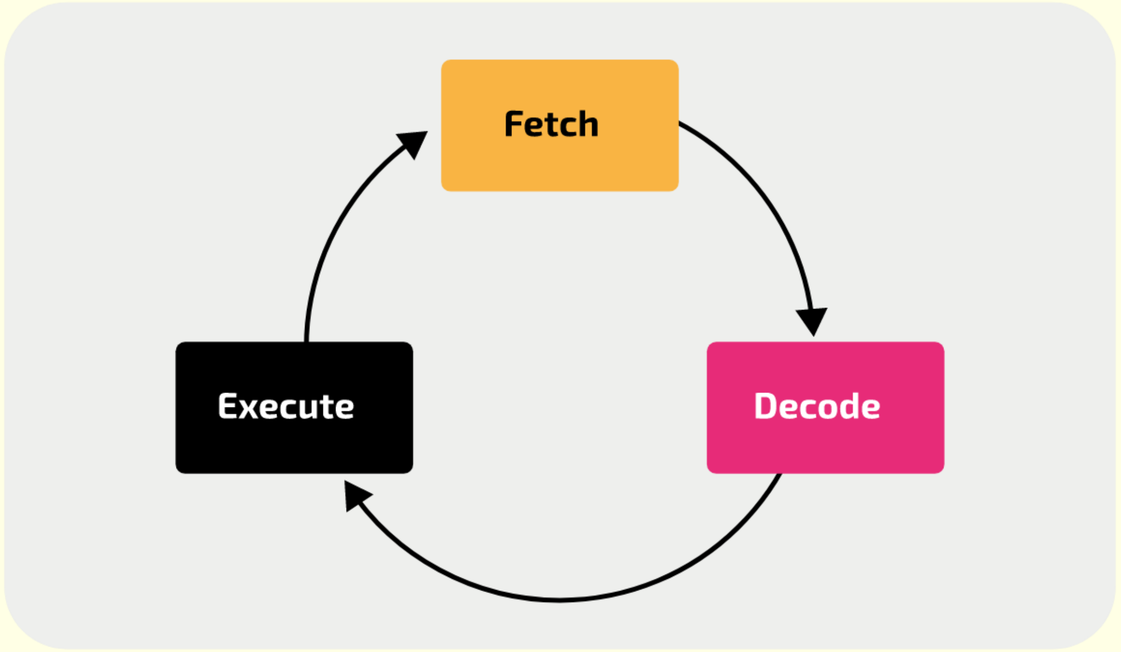
Learning Aims

Describe the Fetch-Execute cycle and the role of the registers and buses:

Learning Aims

- Be able to describe the **<u>FDE cycle</u>**



Fetch next instruction from memory
Decode instruction
Execute the instruction

Key Terms

Fetch
Decode
Execute
Data
Instructions
Memory Address

Buses (Address, data and control bus)
Contents
Copies
Read/Write
Carries data

MAR
MDR
CIR
Program counter
Accumulator

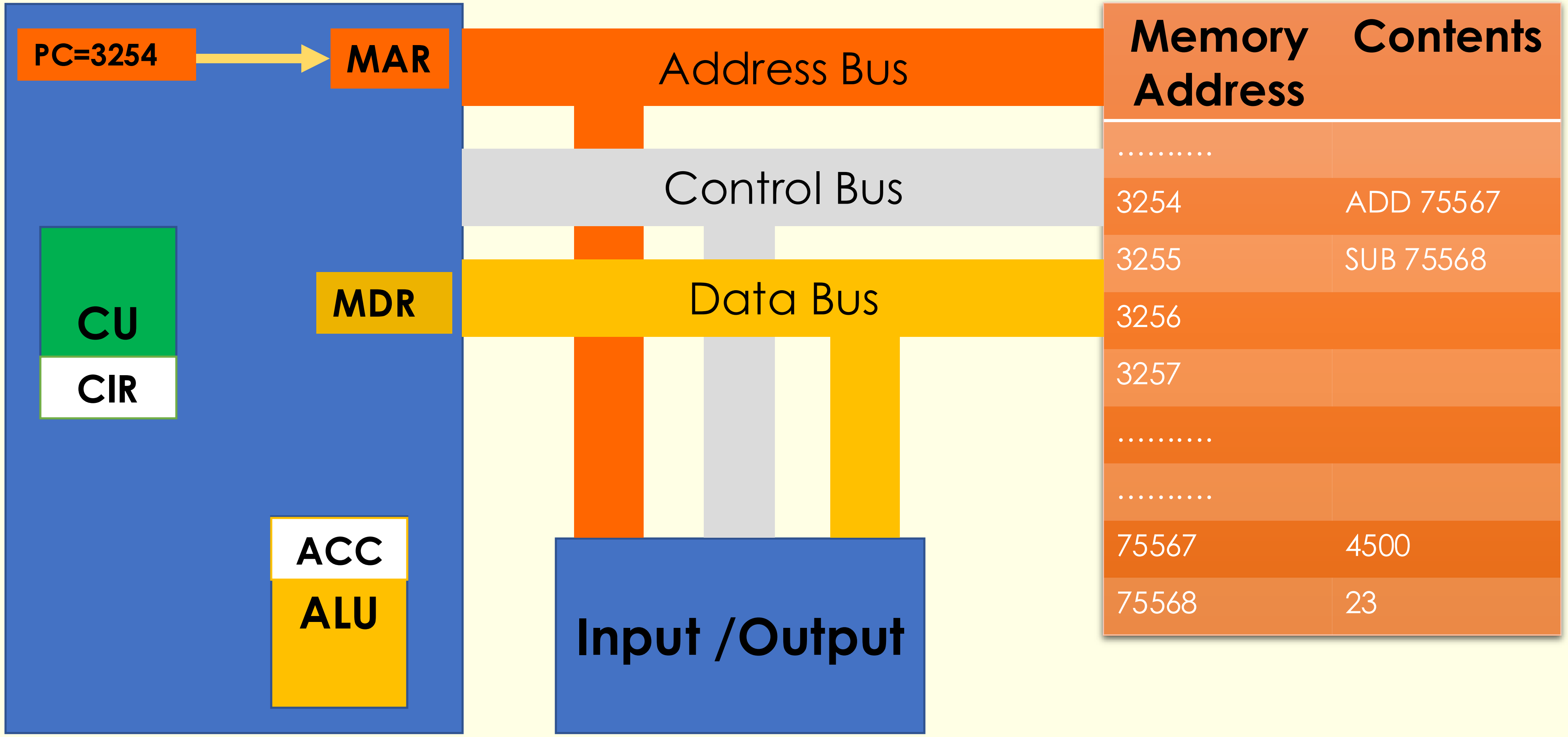
Racap: Registers and the Fetch-Decode-Execute cycle

The registers you should know about include:

- **Program Counter (PC)** - this holds the address of the next instruction to be fetched and executed.
- **Current Instruction Register (CIR)** - this holds the current instruction being executed.
- **Memory Address Register (MAR)** - this holds the RAM address you want to read to or write from.
- **Memory Data Register (MDR)** - this holds the data you have read from RAM or want to write to RAM.
- **Accumulator** - hold the data being worked on and the results of arithmetic and logical operations.
- **Interrupt Register** - this holds details about whether an interrupt has happened.

Example F-D-E

Using registers to execute an instruction in a program.
Consider the following situation:



Memory Address	Contents
……….	
3254	ADD 75567
3255	SUB 75568
3256	
3257	
……….	
……….	
75567	4500
75568	23

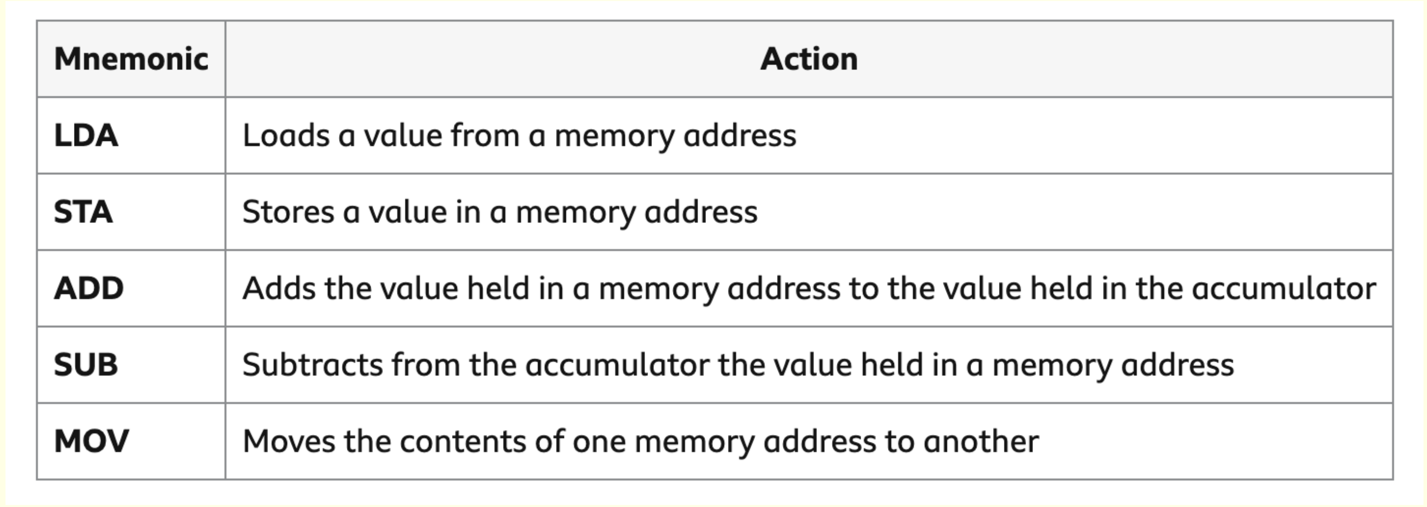
Note in the above that we have not used binary either for the RAM address or the contents, to make things easier to understand! In reality everything in RAM would be stored as 0's and 1's

Before we begin let's understand what instructions look like in memory

Memory Address	Contents
....	
3254	ADD 75567
3255	SUB 75568
3256	
3257	
....	
....	
75567	4500
75568	23

Processors have a specific **instruction set**.

Here are some examples of the operations a CPU can perform.



Mnemonic	Action
LDA	Loads a value from a memory address
STA	Stores a value in a memory address
ADD	Adds the value held in a memory address to the value held in the accumulator
SUB	Subtracts from the accumulator the value held in a memory address
MOV	Moves the contents of one memory address to another

Address bus and word sizes

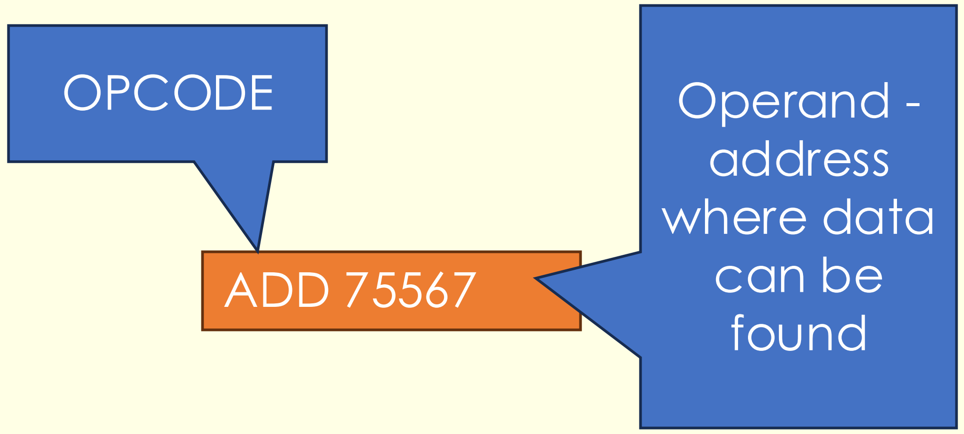
Memory is divided up internally into units called **words**.

A word is a fixed size group of digits, typically 16, 32 or 64 bits, which is handled as a unit by the processor, and different types of processor have different **word sizes**.

Each word in memory has its own specific address.

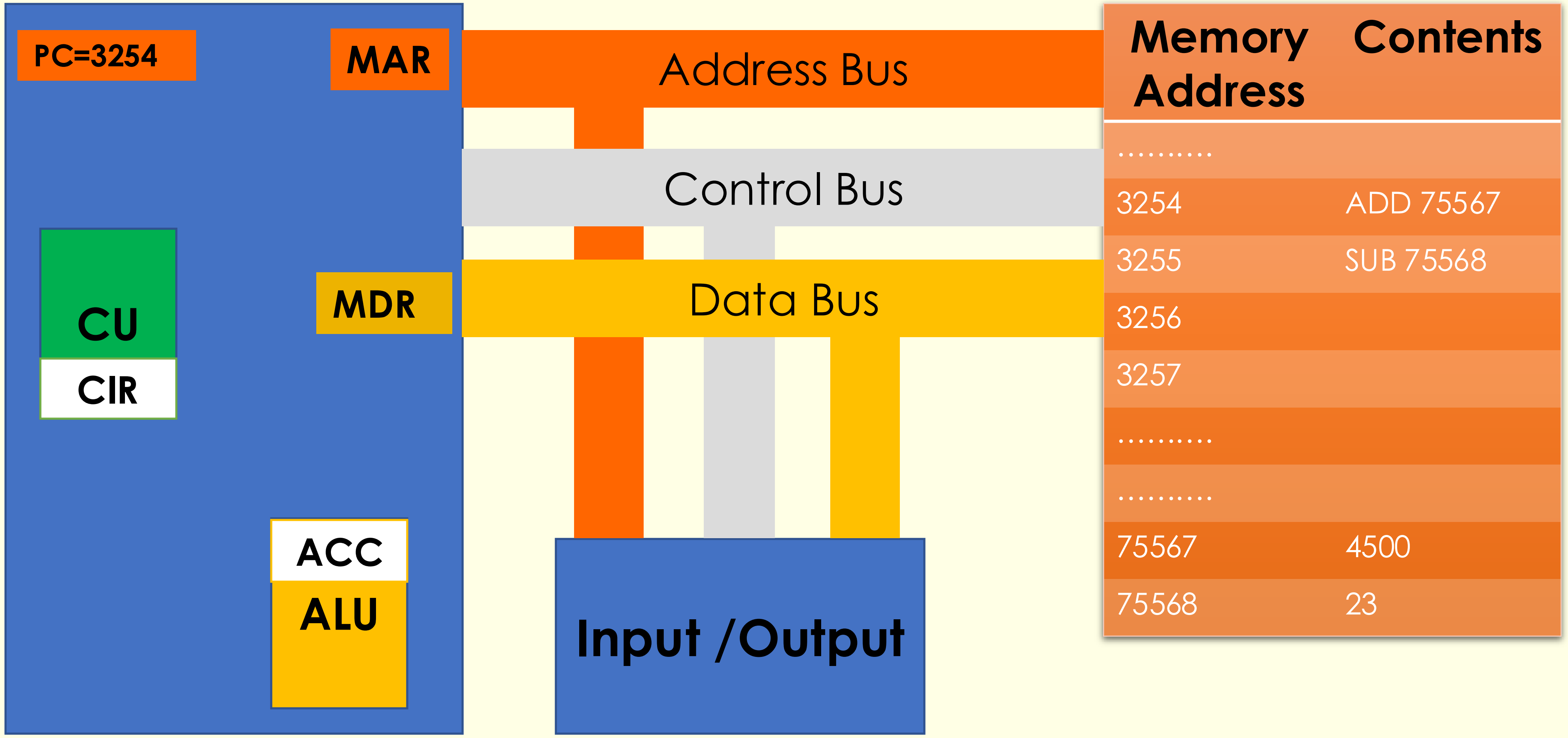
The **address bus** transmits the memory addresses of words that are used as **operands** in program instructions, so that the data can be retrieved and sent back to the processor.

When an instruction has been performed and the result is to be stored at a particular memory location, it is transmitted via the data bus.



Steps in the fetch process

1. The CPU reads the contents of the Program Counter to find the address of the next instruction to be fetched, decoded and executed.

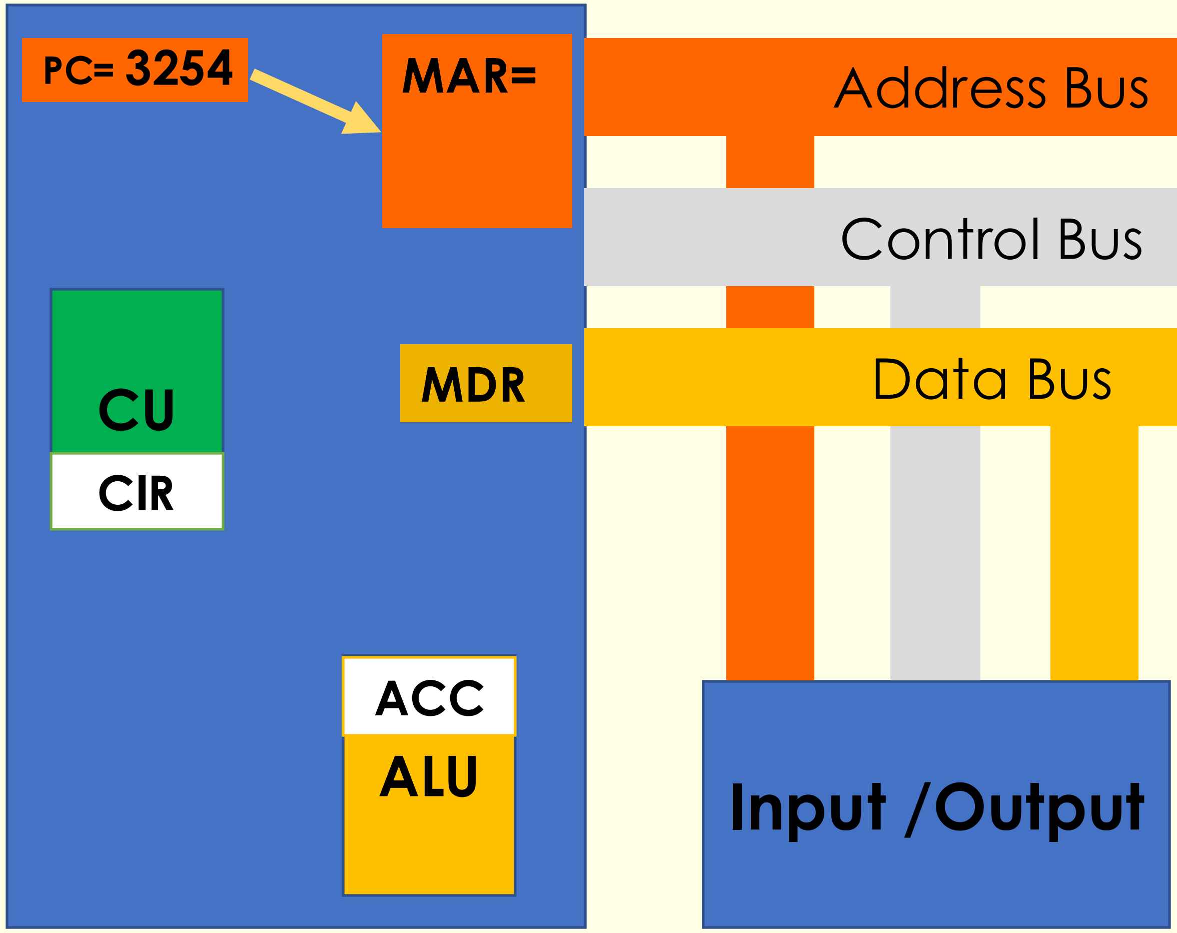


Memory Address	Contents
....	
3254	ADD 75567
3255	SUB 75568
3256	
3257	
....	
....	
75567	4500
75568	23

Steps in the fetch process

The CPU reads the contents of the Program Counter to find the address of the next instruction to be fetched, decoded and executed.

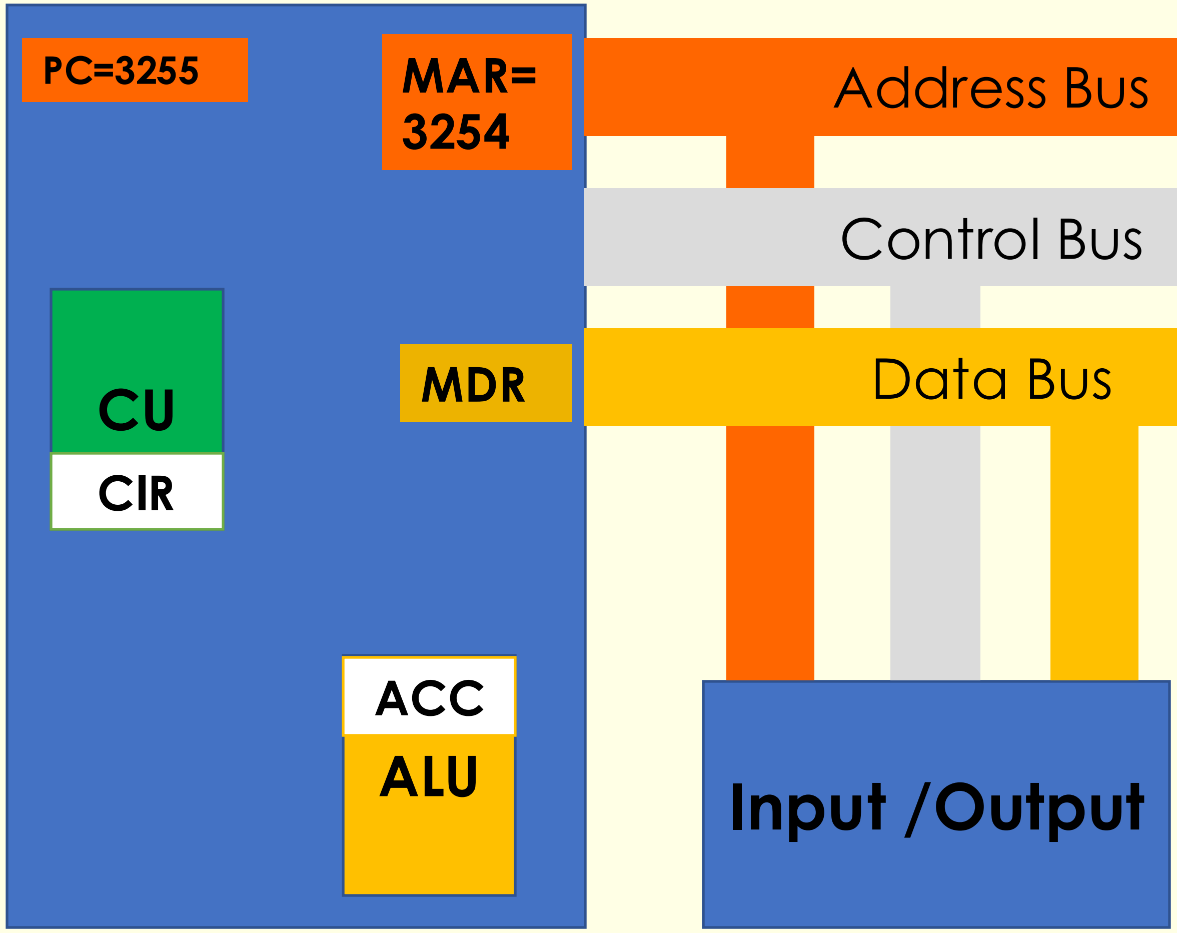
2. The contents from the PC are then copied to the MAR.



Memory Address	Contents
....	
3254	ADD 75567
3255	SUB 75568
3256	
3257	
....	
....	
75567	4500
75568	23

Steps in the fetch process

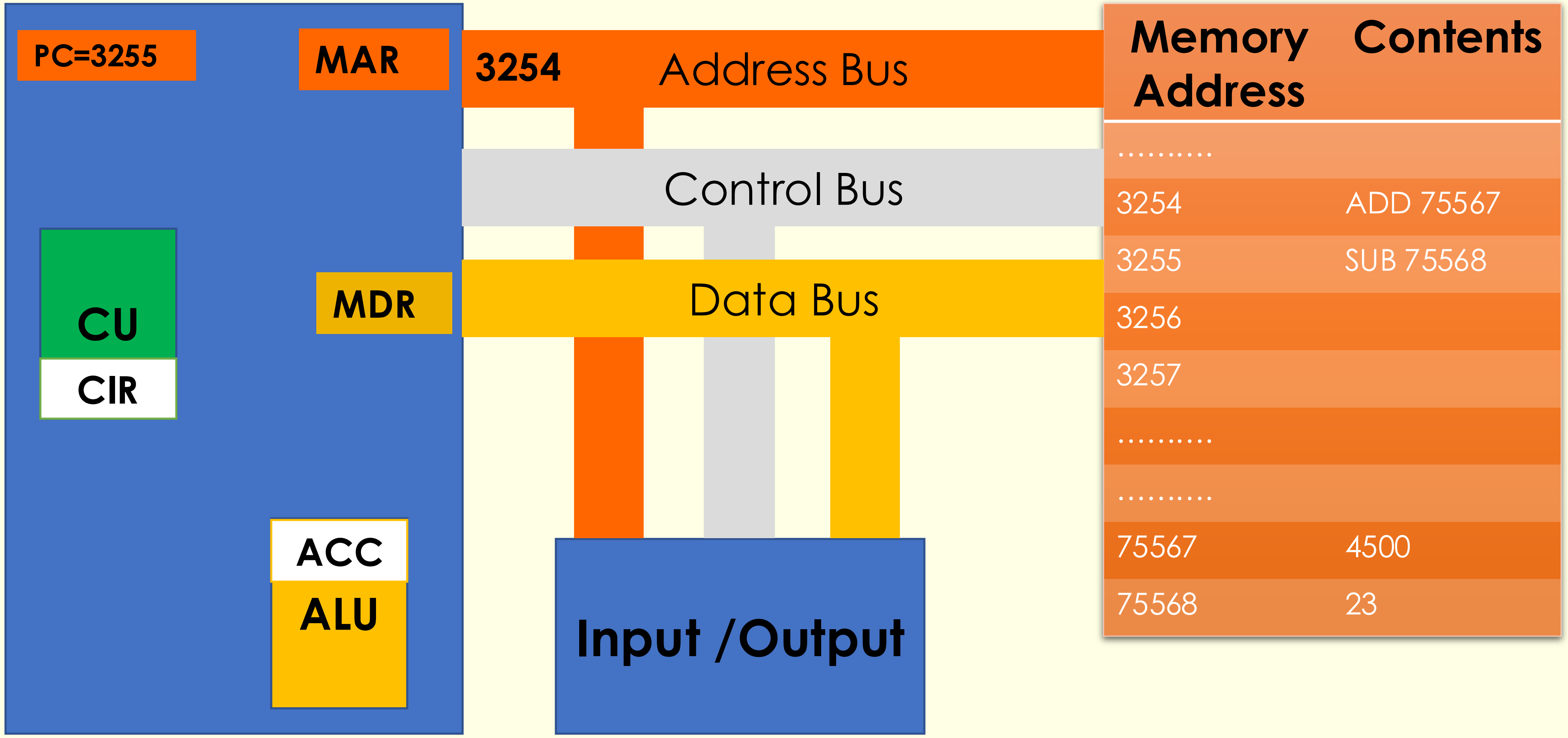
3. As soon as it is read, the PC increments. PC = PC + 1, or 3255



Memory Address	Contents
……….	
3254	ADD 75567
3255	SUB 75568
3256	
3257	
……….	
……….	
75567	4500
75568	23

Steps in the fetch process

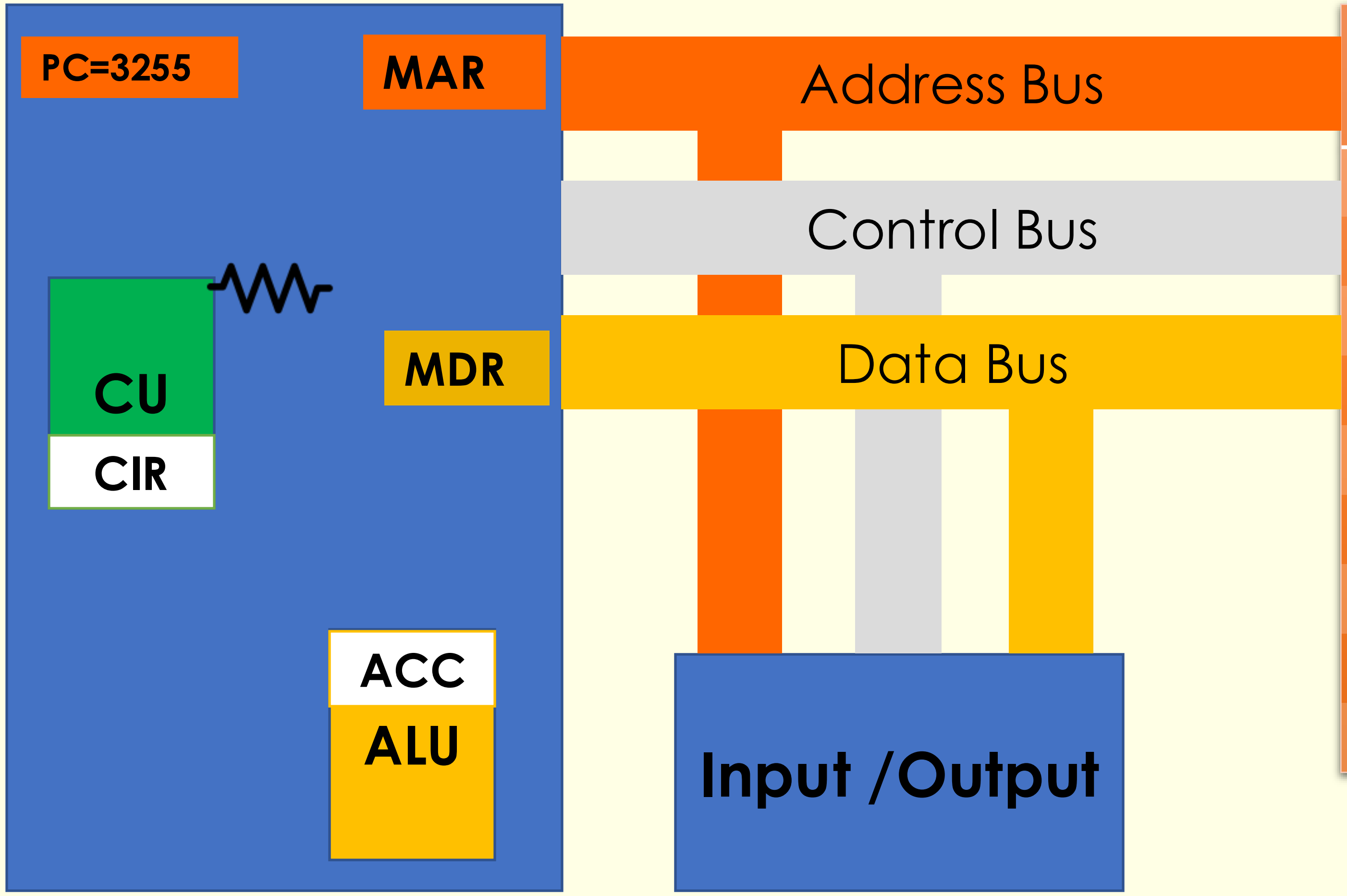
4. The address in the MAR is sent across the **address bus** to locate the contents in RAM



Memory Address	Contents
....	
3254	ADD 75567
3255	SUB 75568
3256	
3257	
....	
....	
75567	4500
75568	23

Steps in the fetch process

5. The control unit sends a **read signal** across the control bus to **read** from the memory address

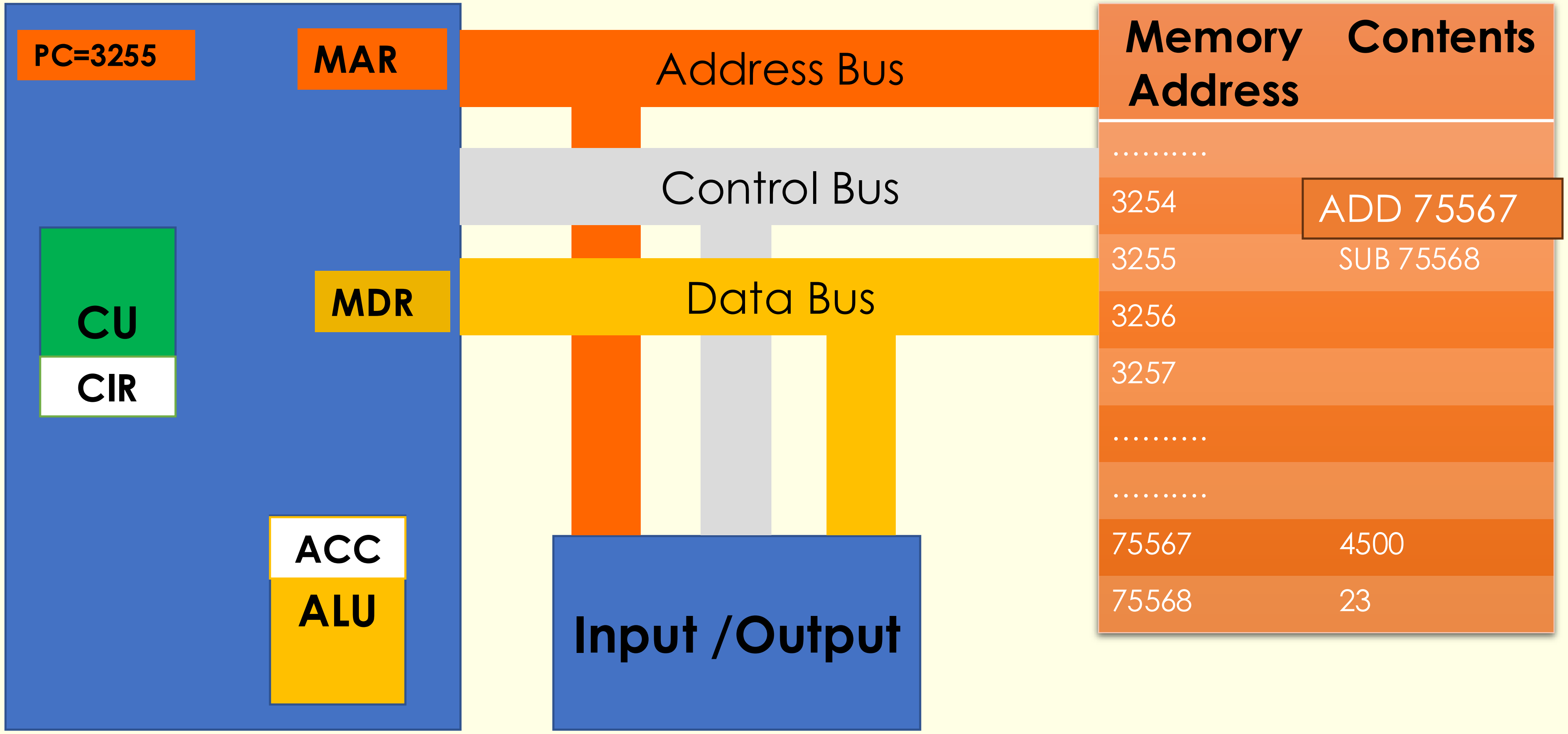


Memory Address	Contents
....	
3254	ADD 75567
3255	SUB 75568
3256	
3257	
....	
....	
75567	4500
75568	23

Steps in the fetch process

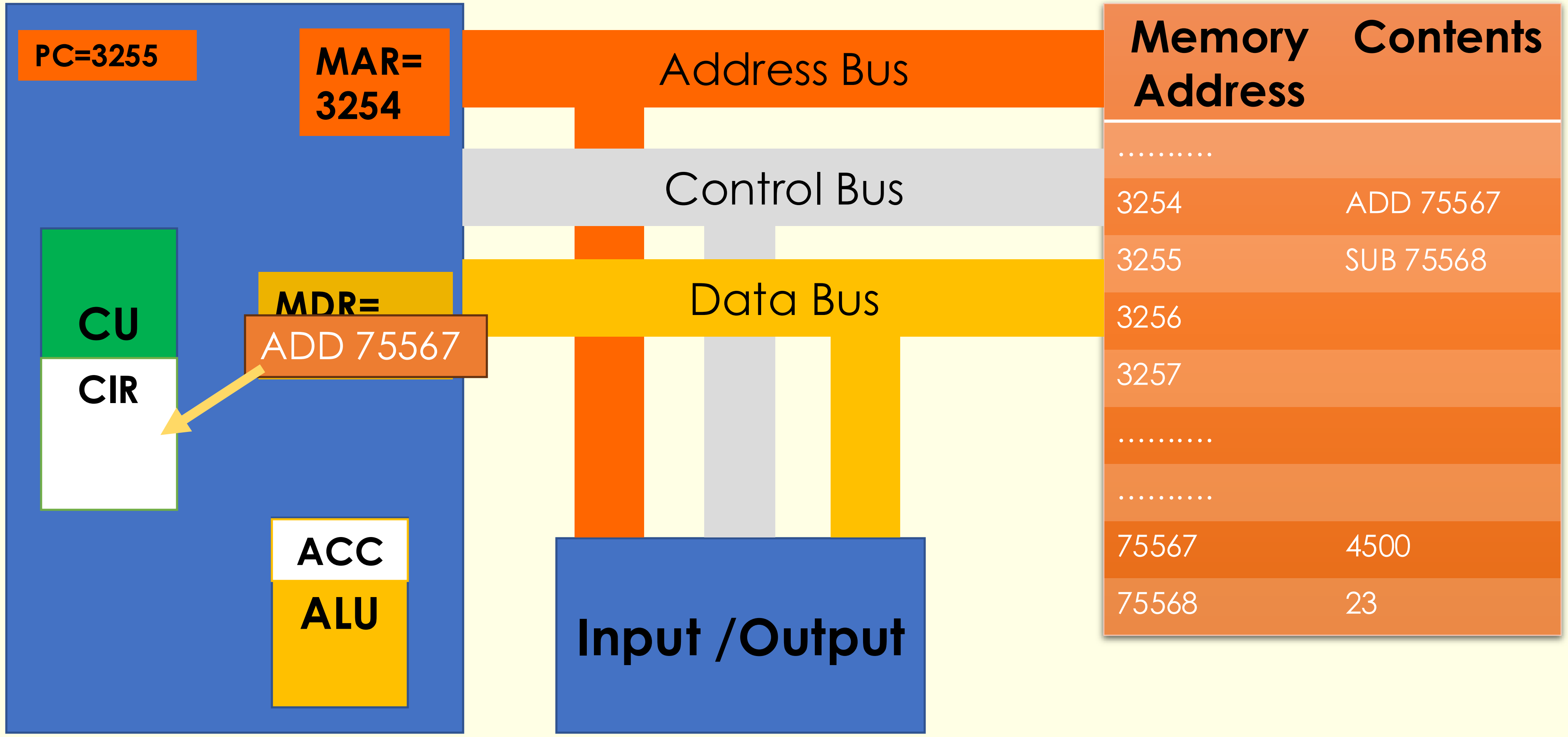
6. The contents of this address are copied to the MDR via the data bus

7. The MDR now holds the instruction that must be executed.



Steps in the fetch process

8. The instruction in the MDR is then copied to the CIR, as we will often need to use the MDR again to complete the execution of an instruction.



Memory Address	Contents
....	
3254	ADD 75567
3255	SUB 75568
3256	
3257	
....	
....	
75567	4500
75568	23

Summary

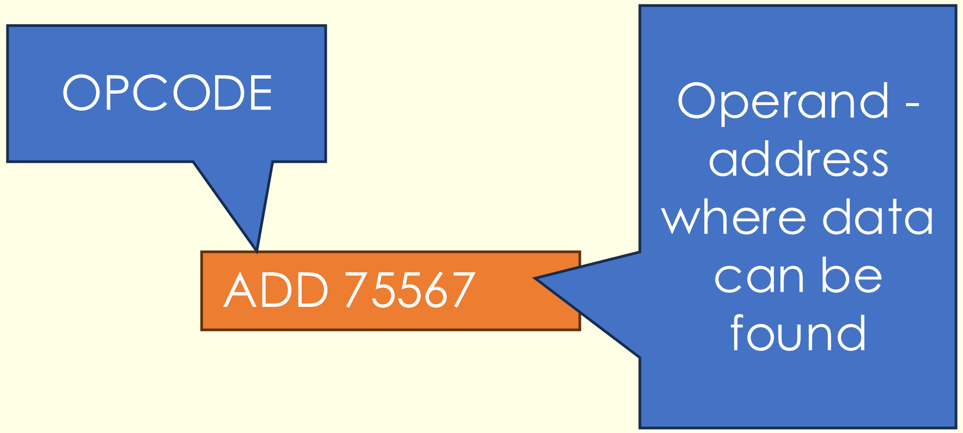
Steps in the fetch process

1. The CPU reads the contents of the Program Counter to find the address of the next instruction to be fetched, decoded and executed.
2. As soon as it is read, the PC increments.
3. The contents from the PC are then copied into the MAR.
4. The address in the MAR is sent across the address bus to locate the contents in RAM
5. The control unit sends a read signal across the control bus to read from the memory address
6. The contents of this address are copied to the MDR via the data bus
7. The MDR now holds the instruction that must be executed.
8. The instruction in the MDR is then copied to the CIR, as we will often need to use the MDR again to complete the execution of an instruction.

Key words:
Contents
Copied
SEND/TRANSFERRED

DECODE Stage

- The control unit decodes the instruction
- The contents of the CIR are divided.
- Part of the instruction might be an operation (like ADD) and part of the instruction might be data,
- The ADD part is known as the **OPCODE** and the data part is known as the **OPERAND**.
- The operator (ADD) is decoded by the Control Unit in the CPU, so it knows what it has to do (ADD in our case).
- The operand 75567 is put back on the MAR.
- The contents of 75567 is then found in RAM and put on the MDR.



Control bus

The control bus is a **bi-directional bus**, meaning that signals can be carried in both directions.

The control bus is used to send control signals that manage and coordinate the activities of the system's components.

Control signals include:

- **Memory Write:** causes data on the data bus to be written into the addressed location
- **Memory Read:** causes data from the addressed location to be placed on the data bus
- **Interrupt request:** indicates that a device is requesting access to the CPU
- **Clock:** used to synchronise operations

EXECUTE

The instruction can now be executed.

This could be:

- Load data from memory
- Write data to memory
- Do a calculation or logic operation using the ALU
- Change the address in the PC
- Halt the program

For instance:

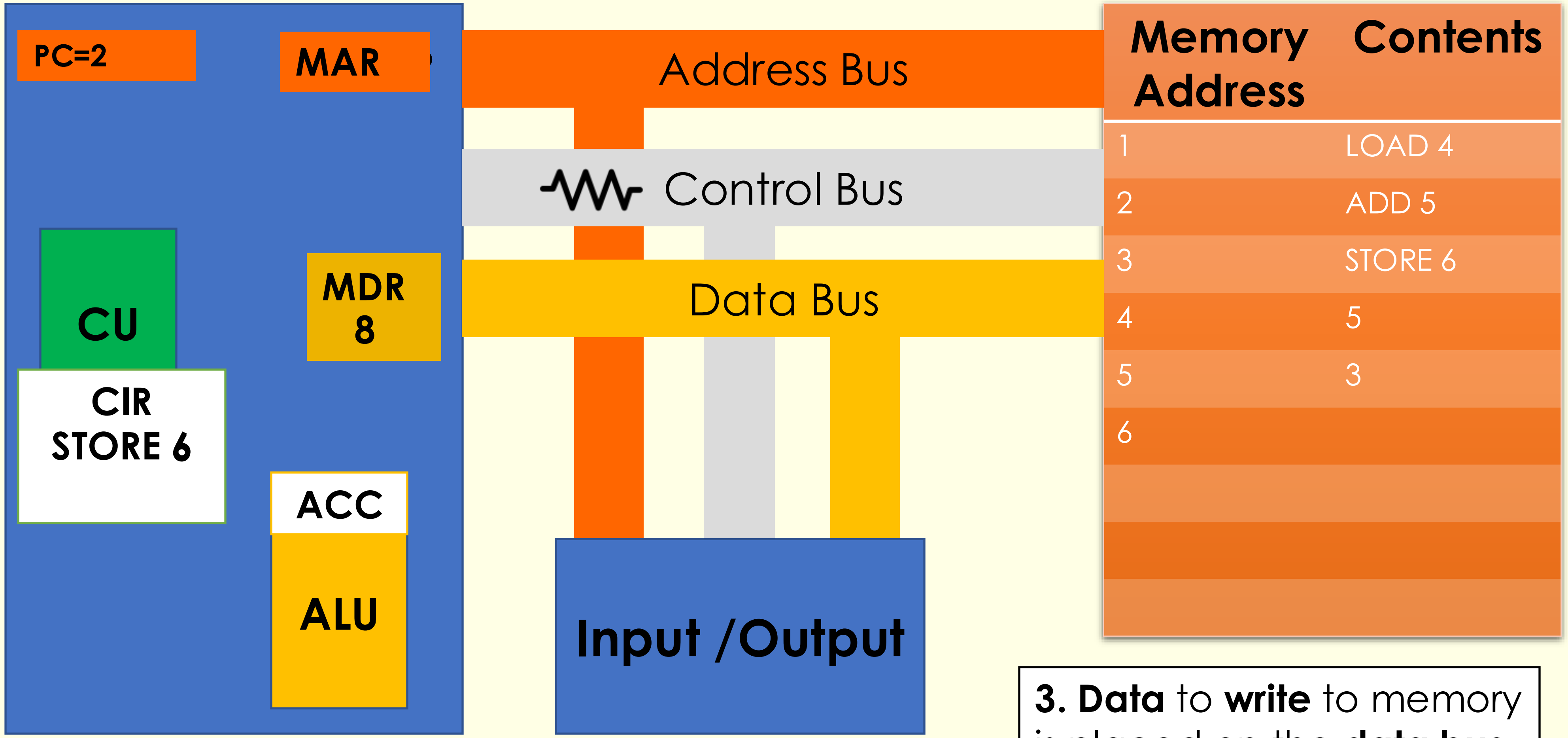
- if the instruction is for a memory location to be read from or written to (that is, LDA or STA), then the address stored within the instruction will be loaded into the MAR.
- In the case of STA, the data stored in the ACC is sent to memory.
- In the case of LDA, the data is loaded from memory into the ACC.
- If the instruction is to carry out a calculation (that is, ADD or SUB) then the contents of the MDR and ACC are sent to the ALU and the result sent back to the ACC.

Executing **STORE 5** instruction

STORE 5 is an instruction used to write data to memory.

1. Control unit sends a signal on the **control bus** to start a **write** operation

2. Address of memory (to store data) is placed on the **address bus**



Memory Address	Contents
1	LOAD 4
2	ADD 5
3	STORE 6
4	5
5	3
6	

3. Data to **write** to memory is placed on the **data bus**

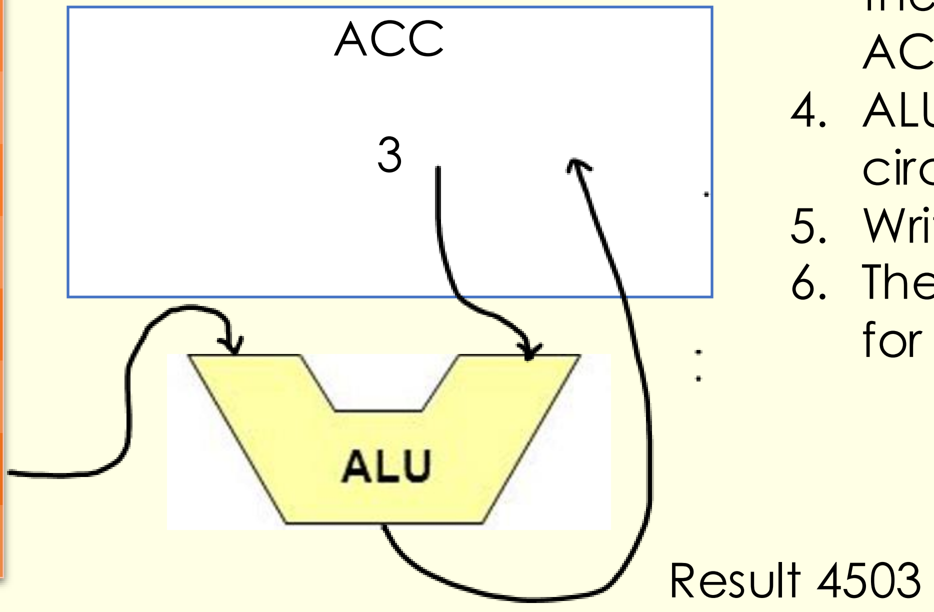
Example: ADD instruction

Arithmetic and logical instructions are carried out using the Accumulator(s) in a CPU.

Signals are sent out to different parts of the CPU to execute the instruction ADD.

This will result in adding 4500 to whatever is in the Accumulator, and then over-writing the contents of the Accumulator with the result of the addition.

Memory Address	Contents
....	
3254	ADD 75567
3255	SUB 75568
3256	
3257	
....	
....	
75567	4500
75568	23



1. Assume a value of 3 has already been loaded into the accumulator
2. Now **ADD 75567** is read, this will get the value 4500 from memory.
3. The value 4500 is loaded into the ALU and the value 3 is loaded into the ALU from the ACC
4. ALU performs the ADD (using an adder circuit)
5. Writes the result back into the ACC
6. Then it can be output or stored in memory for later use.