



# **Microprocessors & Microcontrollers**

## **: Arm Cortex M0+**

### **(Using RP2040)**


## **Lecture 1.2.1**

**Instruction Fetch, ADD and STR Instructions**

**Mouli Sankaran**

# Lecture 1.2.1: Focus

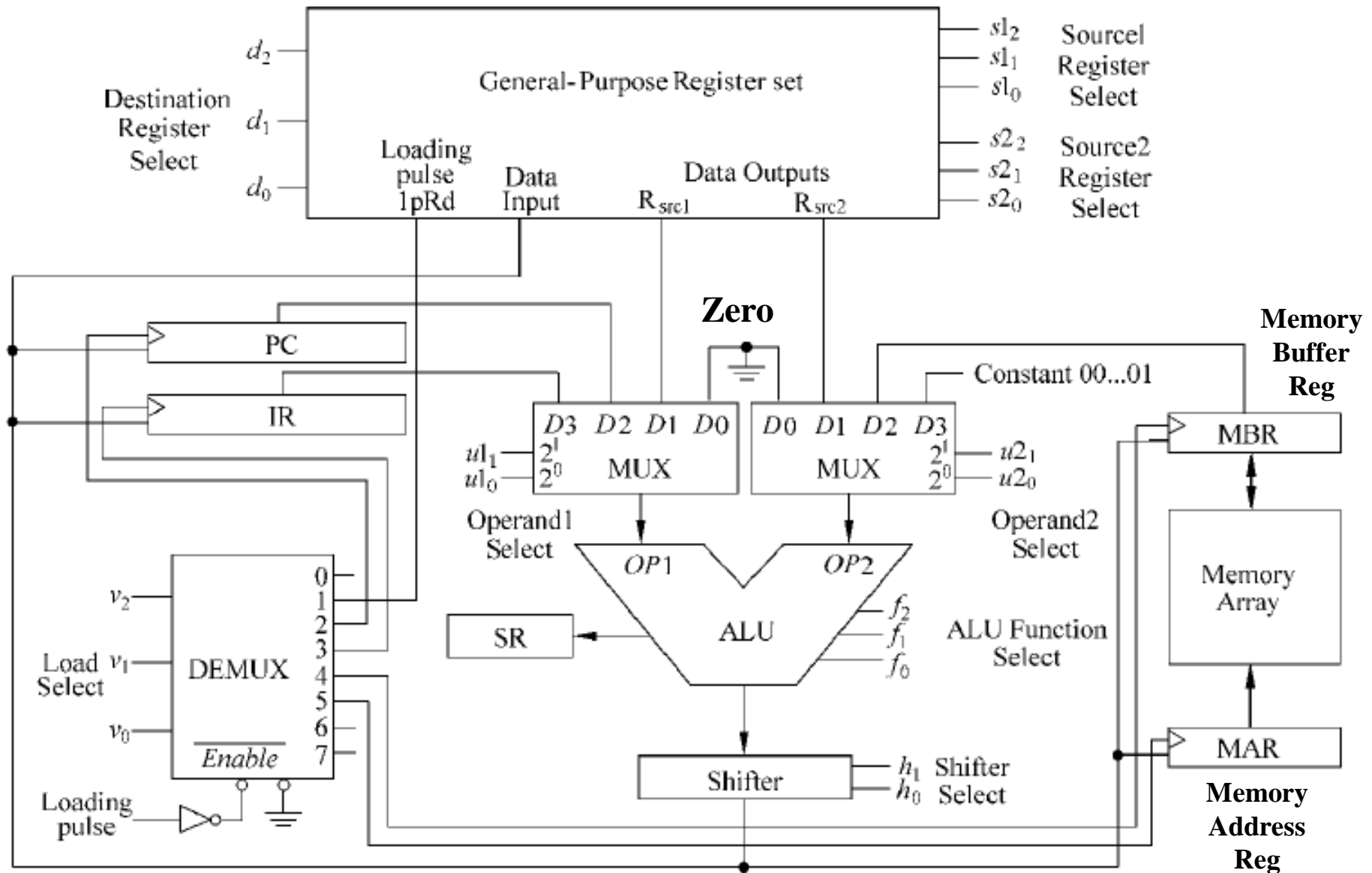
- Simple CPU Design
  - Instruction Fetch
  - Add Instruction
  - Store Instruction



# Simple CPU Design

# CPU Design: General Purpose Reg Based

Ref: Computer Organization-Principles, Analysis, and Design –By Lan Jin and Bo Hatfield, Cengage Learning



# CPU Design: General Purpose Reg Based

Ref: Computer Organization-Principles, Analysis, and Design –By Lan Jin and Bo Hatfield, Cengage Learning

## ➤ **Program Counter (PC) :**

It contains the address of an instruction to be executed next. The PC is updated by the CPU after each instruction is executed so that it always points to the next instruction to be executed. A branch or skip instruction will also modify the content of the PC.

## ➤ **Instruction Register (IR) :**

It contains the instruction most recently fetched or executed. The fetched instruction is loaded into an IR, where the opcode and operand specifier is analyzed.

## ➤ **Memory Buffer (or Data) Register (MBR or MDR) :**

It contains a word of data to be written to memory or the words most recently read. Contents of MBR are directly connected to the data bus.

## ➤ **General Purpose Registers:**

These components are general use memory storage in the CPU that can be accessed very fast (as compared to RAM); they are created from combining latches with a decoder. The latches create circuitry that can remember while the decoder creates a way for individual memory locations to be selected.

# CPU Design: General Purpose Reg Based

Ref: Computer Organization-Principles, Analysis, and Design –By Lan Jin and Bo Hatfield, Cengage Learning

MAR:

The functioning of a Memory Address Register can be explained according to the following steps:

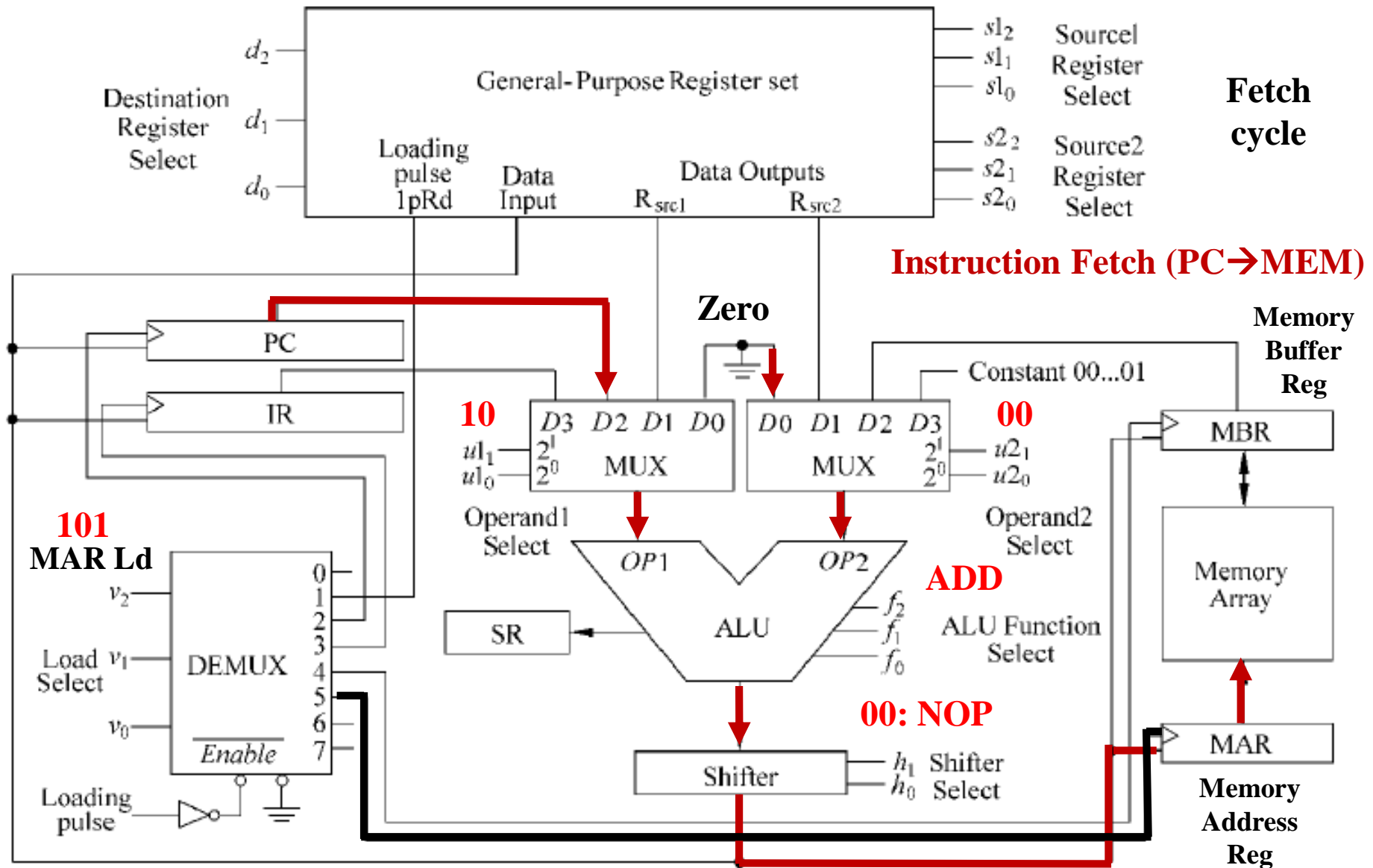
- It holds the memory location of the next piece of data or instruction that will be processed.
- The processor fetches this memory location from the MAR.
- The data or instruction at the fetched address is then read or written.
- Example: Consider the scenario of a spreadsheet application. The user selects a cell to update the content. As part of this process, the processor fetches the instruction that modifies the cell content. The MAR holds the address of this instruction in memory, which the processor then reads to execute.
- **Difference Between Program Counter and Memory Address Register**
- The **Program Counter (PC)** is a type of register that holds the memory address of the next instruction to be executed by the Central Processing Unit (CPU). Following each instruction fetch, the Program Counter increments, keeping track of the sequential execution while processing software.
- The **Memory Address Register (MAR)**, is more generalised in scope. It holds the memory addresses of data and instructions that a CPU needs to access for its next processing steps. Virtually any read or write cycles within memory can involve the MAR, making it a highly versatile component.





# Instruction Fetch

# Instruction Fetch: PC → Program mem





# Instruction Fetch: PC → Program mem

## Steps Involved in Fetch Cycle

**Fetch Cycle:** The Program Counter contains the address of the instruction that has to be next executed at the start of the fetch cycle (PC).

**Step 1:** The address in the program counter is transferred to the memory address register (MAR), which is the only register connected to the address lines of the system bus.

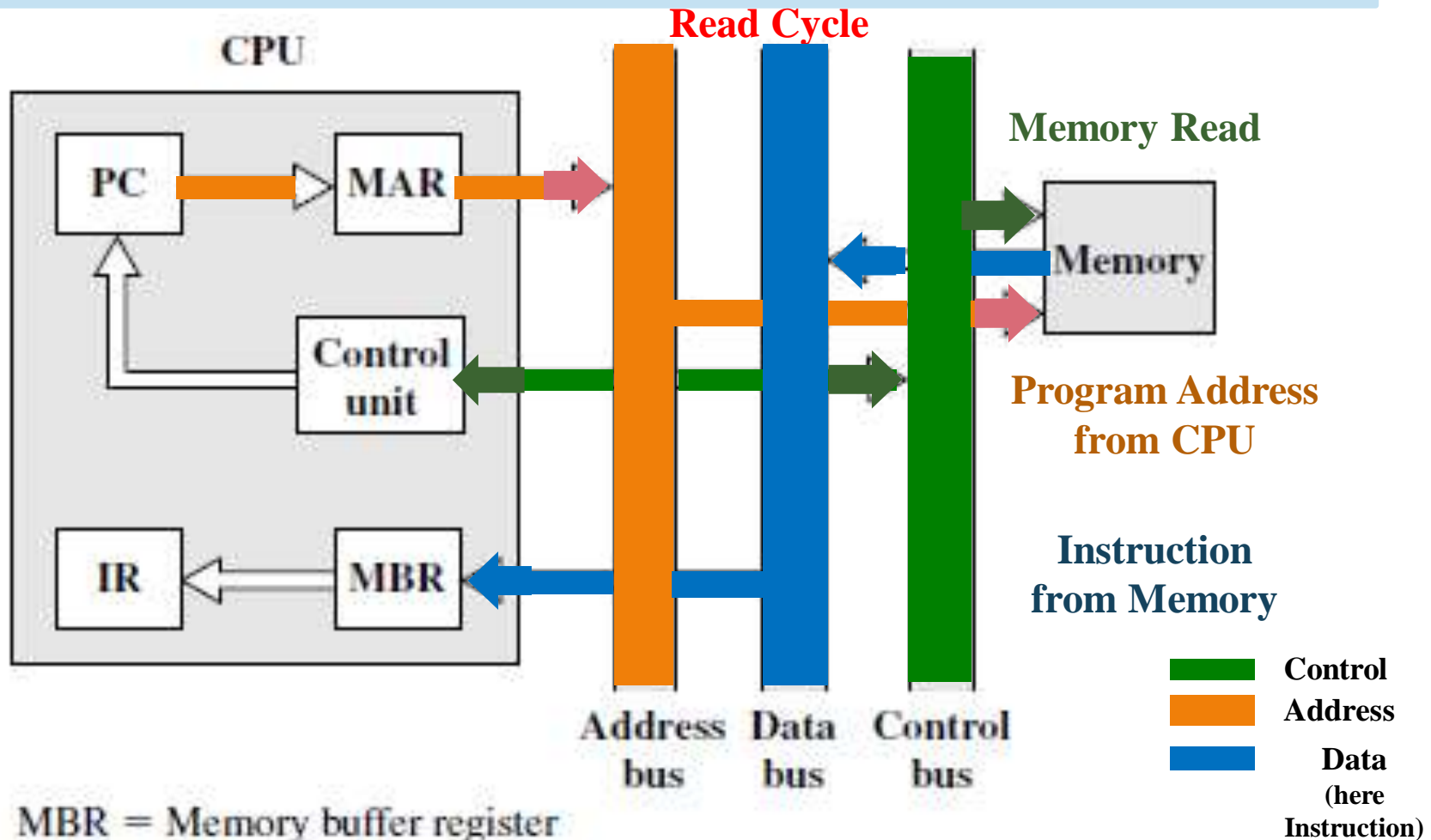
**Step 2:** The MAR address is added to the address bus. The control unit then issues a READ command on the control bus, and the result is displayed on the data bus before being copied into the memory buffer register (MBR). The program counter is increased by one to prepare for the next instruction. ( These two actions can be performed concurrently to save time.)

**Step 3:** The MBR's content is transferred to the instruction register (IR).

- Move the contents of the PC to MAR.
- MAR-specified memory location contents are to be copied to the MBR. The process increases the PC's content.
- Copy the contents of the MBR to the IR.

# Instruction Fetch – Program Mem Read

## (CPU-Memory Interactions explained)



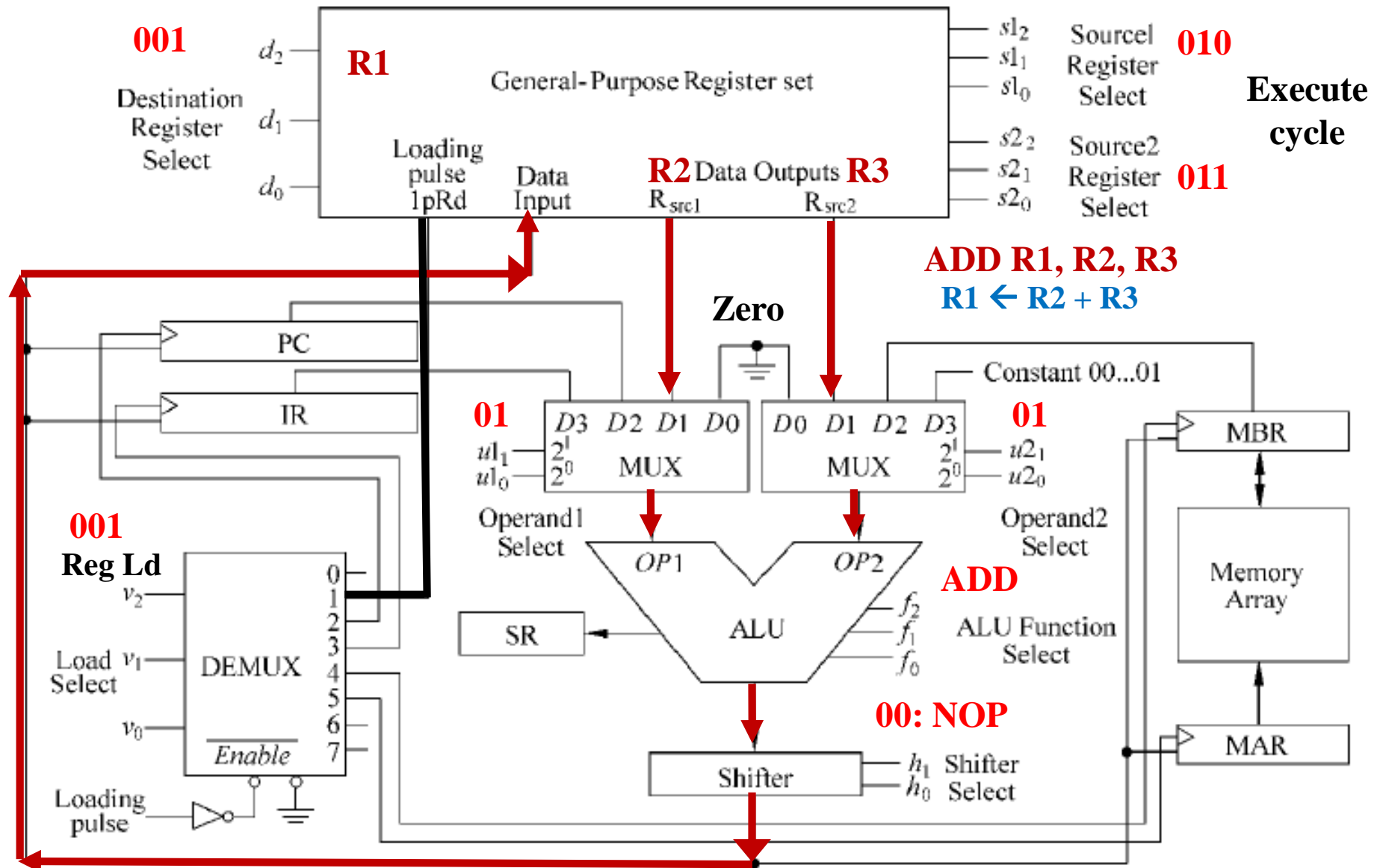
MBR = Memory buffer register  
MAR = Memory address register  
IR = Instruction register  
PC = Program counter

**Control bus** has the following **signals**:  
**RD/WR** commands **from CPU**  
**WAIT** signal from the **memory**



# ADD Instruction

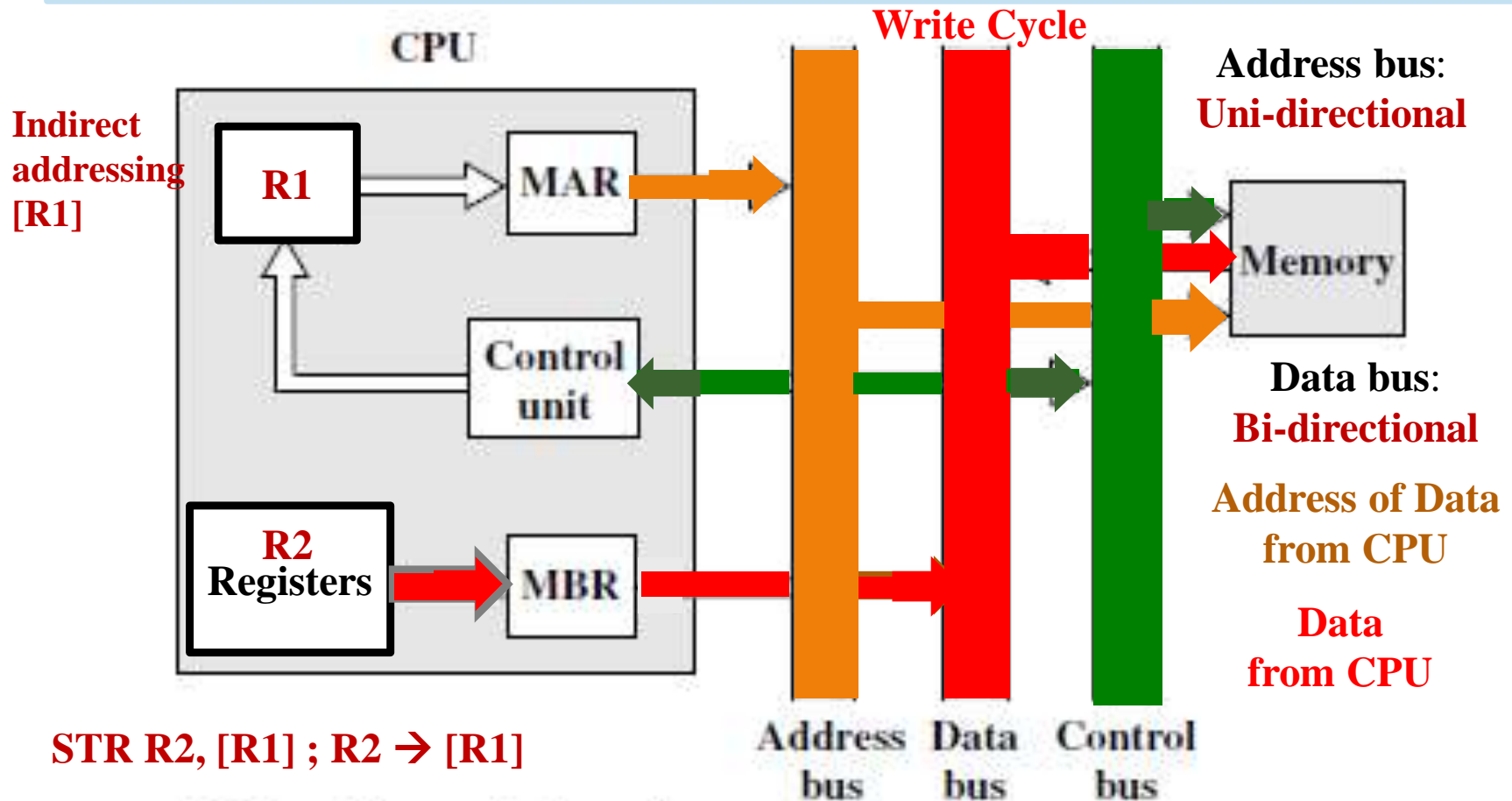
# Instruction Execution: ADD R1, R2, R3





# Store Instruction

# Store Instruction: Reg → Data Mem



MBR = Memory buffer register  
MAR = Memory address register

**Control bus** has the following **signals**:  
**RD/WR** commands **from CPU**  
**WAIT** signal from the **memory**



# Store Instruction: Reg → Data Mem

The assembly language instruction **STR R2, [R1]** is a store instruction, which is used to store the contents of register R2 into the memory location specified by the contents of register R1. Let's break down the instruction and explain how it works, including the interaction with the CPU and memory using the Memory Address Register (MAR) and Memory Buffer Register (MBR)

## **Instruction Fetch (IF):**

The CPU fetches the instruction **STR R2, [R1]** from memory.

The program counter (PC) is incremented to point to the next instruction.

## **Decode (ID):**

The CPU decodes the instruction to understand its operation.

In this case, the CPU recognizes that it is a store instruction.

## **Execution (EX):**

The CPU calculates the effective address for the memory operation. In this case, it uses the contents of register R1 as the memory address.

The Memory Address Register (MAR) is loaded with the effective address (contents of R1).

# Store Instruction: Reg → Data Mem

## **Memory Access (MEM):**

The CPU initiates a memory read operation by placing the contents of the MAR onto the address bus. The memory reads the data from the specified address (contents of R1) and loads it into the Memory Buffer Register (MBR).

## **Write Back (WB):**

The CPU stores the data from register R2 into the MBR.

The contents of R2 are placed onto the data bus.

The memory receives the data from the data bus and writes it to the address specified by the MAR (contents of R1)

**MAR (Memory Address Register):** Holds the memory address (contents of R1) during memory operations.

**MBR (Memory Buffer Register):** Holds the data read from memory (contents of the memory address specified by R1) and is later used to store the modified data before writing it back to the same memory address.

# Lecture 1.2.1: Summary

- Simple CPU Design
  - Instruction Fetch
  - Add Instruction
  - Store Instruction