**Faculty of Computing**



**Computer Architecture**

**Spring 2025**

**LAB # 5**

**Instructor**

Ayesha Akram

TF, Faculty of Computing,

Riphah International University, Islamabad

**Learning Objectives:**

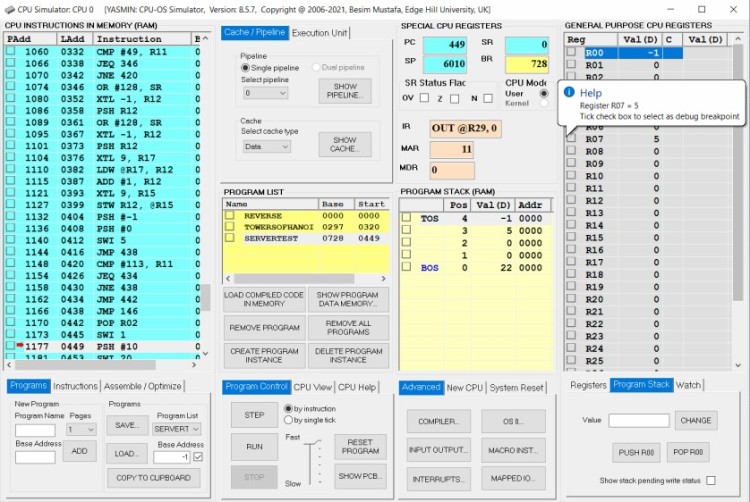
* Understand the basics of CPU Sim software
* Learn how to set up and configure the environment
* Explore basic CPU architecture components
* Develop and test simple assembly programs
* Analyze the Fetch-Decode-Execute cycle through simulations

**Essential Tools in Our Lab:**

* **Computer System:** The main machine
* **CPU Sim**: CPU Sim is a tool used for simulating simple CPU architectures, helping students understand processor design, instruction execution, and debugging.

**Introduction to CPU Sim**

CPU Sim is an educational software used for simulating the functioning of a CPU at the microarchitecture level. It allows to design and test simple CPU architectures, write assembly language programs, and understand instruction execution cycles.



**Key Features of CPU Sim**

1. **Custom CPU Design:** Users can define their own CPU components, such as registers, memory, and ALU.
2. **Instruction Set Simulation:** Allows defining and testing assembly language instructions.
3. **Memory Management:** Simulates memory read/write operations and addressing modes.
4. **Execution Cycle Analysis:** Visualizes fetch, decode, and execute cycles for each instruction.
5. **Step-by-Step Debugging:** Helps students analyze instruction execution and troubleshoot errors.

**Computer Architecture**

* **Instruction Cycle Understanding:** Demonstrates the role of the control unit, ALU, and registers.
* **Pipeline Processing:** Helps visualize how pipelining works in instruction execution.
* **Memory Hierarchy Simulation:** Illustrates how registers, cache, and RAM interact.
* **Microprogramming:** Allows experimentation with microinstructions for CPU control.

# ****Download and Install CPU Sim****

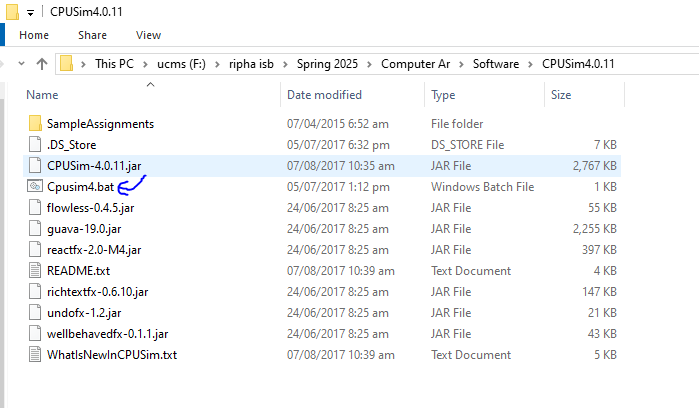
1. Download the JRE compatible with your operating system.

<https://centrexmanager.frontier.com/webstart/webstartHelp.jsp>



1. Visit the official CPU Sim website.

<https://www.cs.colby.edu/djskrien/CPUSim/>

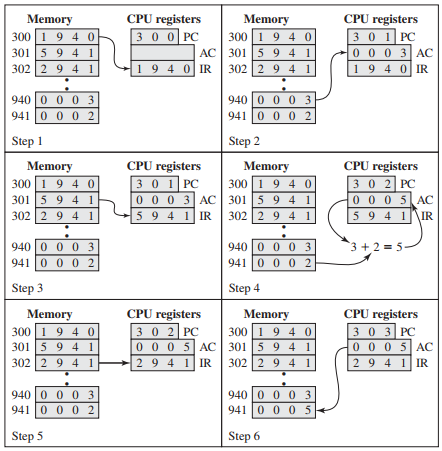


## **Introduction:**

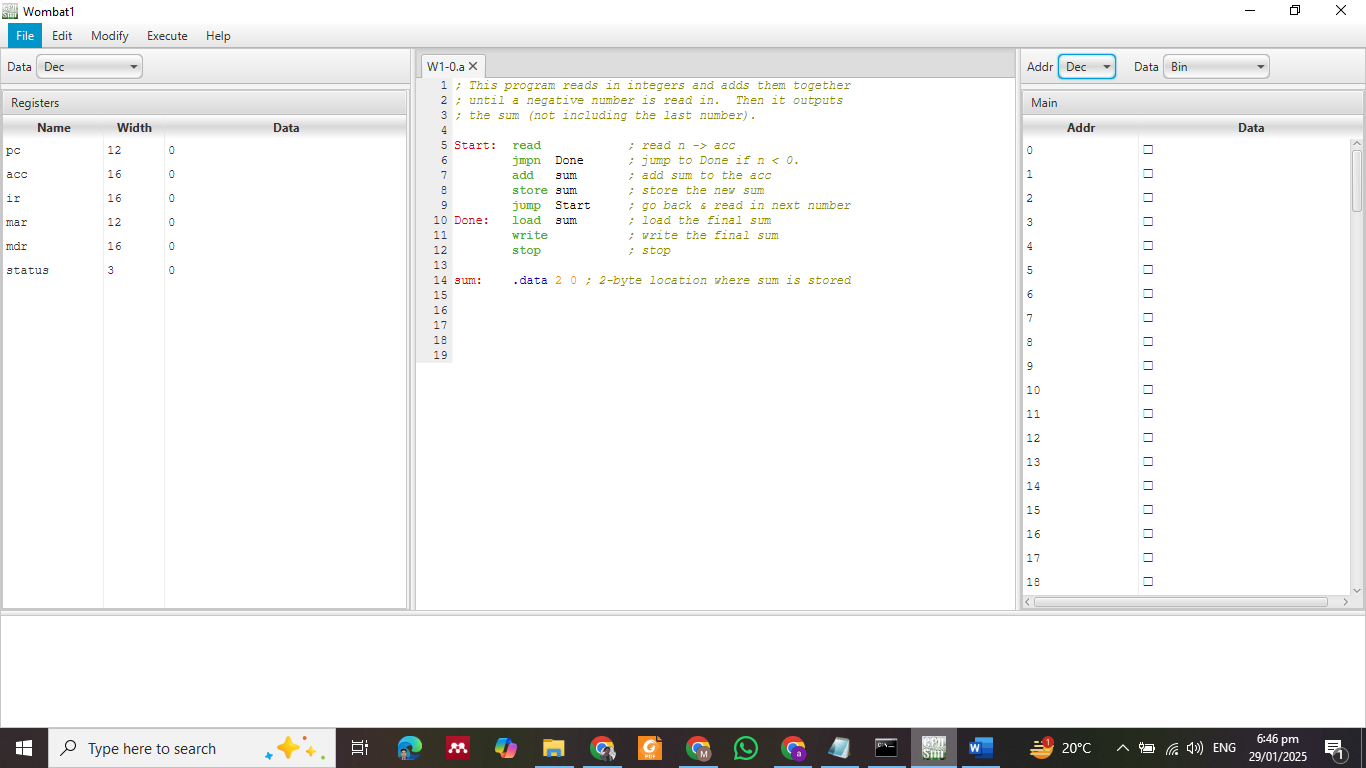
CPU SIM is a software tool used to simulate a basic CPU architecture, helping students understand how instructions are fetched and executed.

In this lab, we will:

1. Define registers required for execution.
2. Implement the Fetch Cycle.



1. Click on the batch file and the window of CPU Sim is open.



The screen shows a CPU simulation interface with the following components:

1. **Top Menu Bar**: Options like File, Edit, Execute, and Help.
2. **Registers Panel (Left)**: Lists registers (pc, ir, mar, etc.) with their bit width and current values (all 0).
3. **Code Editor (Center)**: Displays an assembly program.
4. **Memory Panel (Right)**: Shows memory addresses and data.
5. **Program Explanation (Bottom)**: Comments describe the program logic and storage of the final sum.

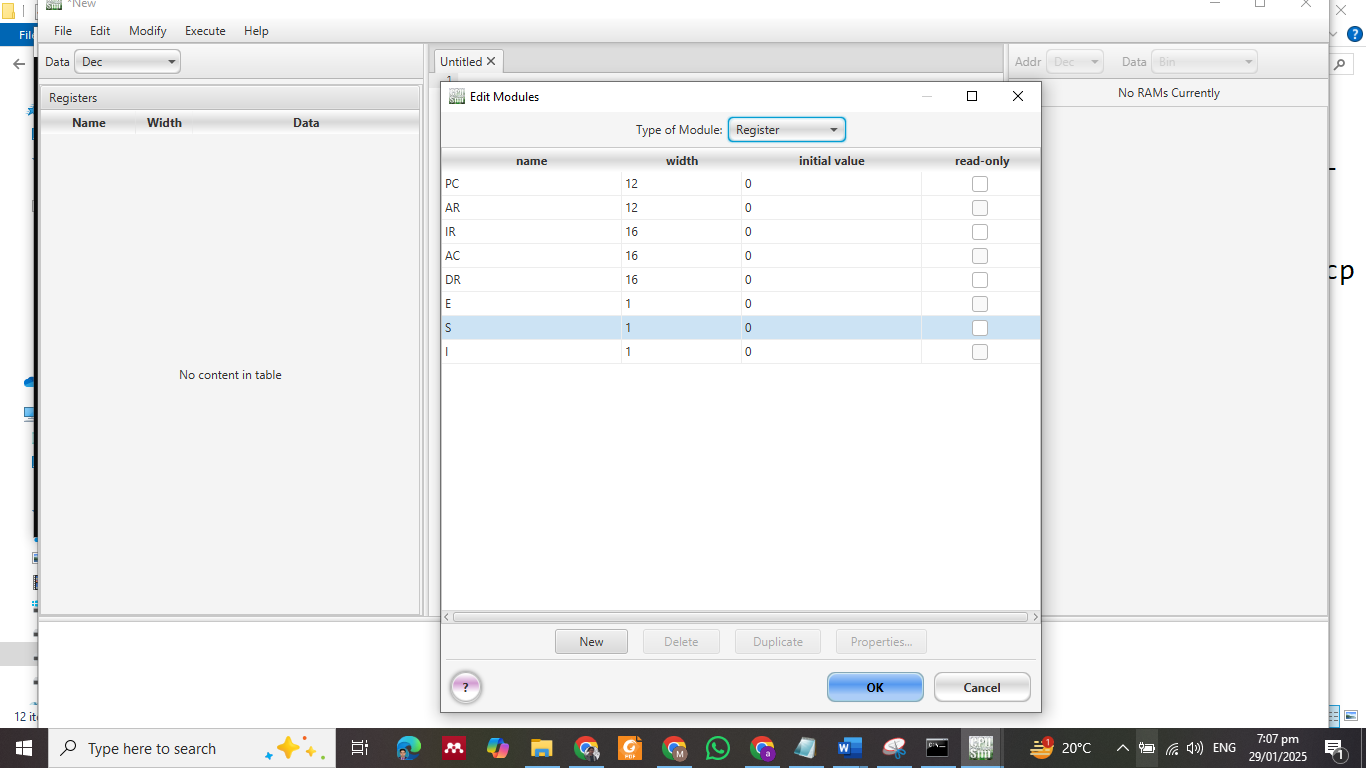
This setup is for writing, testing, and debugging assembly programs.

## **Registers Used in CPU SIM:**

1. **Program Counter (PC)** – Stores the address of the next instruction (**12-bit**).
2. **Address Register (AR)** – Stores the memory address being accessed (**12-bit**).
3. **Instruction Register (IR)** – Holds the fetched instruction (**16-bit**).
4. **Accumulator Register (AC)** – Used for arithmetic operations (**16-bit**).
5. **Data Register (DR)** – Temporarily holds data being processed (**16-bit**).
6. **Temporary Register (TR)** – Optional register for intermediate calculations (**16-bit**).

A close-up of a register

AI-generated content may be incorrect.



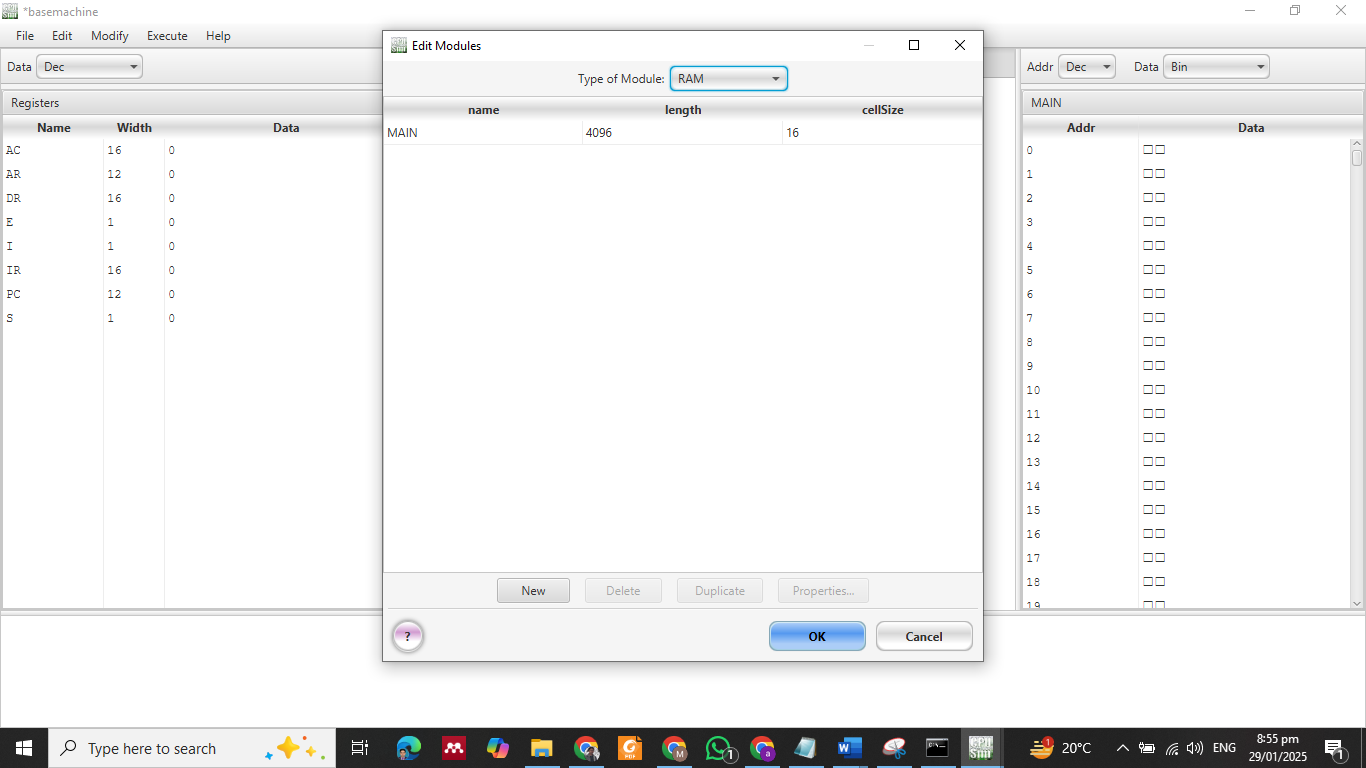
1. **Condition Registers:**
   * **E (Carry Bit)** – Used for carry operations.
   * **S (Status Register)** – Used for halt operations.
   * **I (Indexing Bit)** – Specifies direct or indirect addressing.

A screenshot of a computer

Description automatically generated

## **Setting Up Memory**

1. Open the **Hardware Module > Register > RAM**.
2. Create a new memory module:
   * **Size:** 4096 words
   * **Cell Size:** 16 bits
   * **Type:** RAM
   * **Format:** Hexadecimal
3. Save the configuration with **.cpu** extension.



A screenshot of a computer

Description automatically generated

***Machine based on the following architecture***

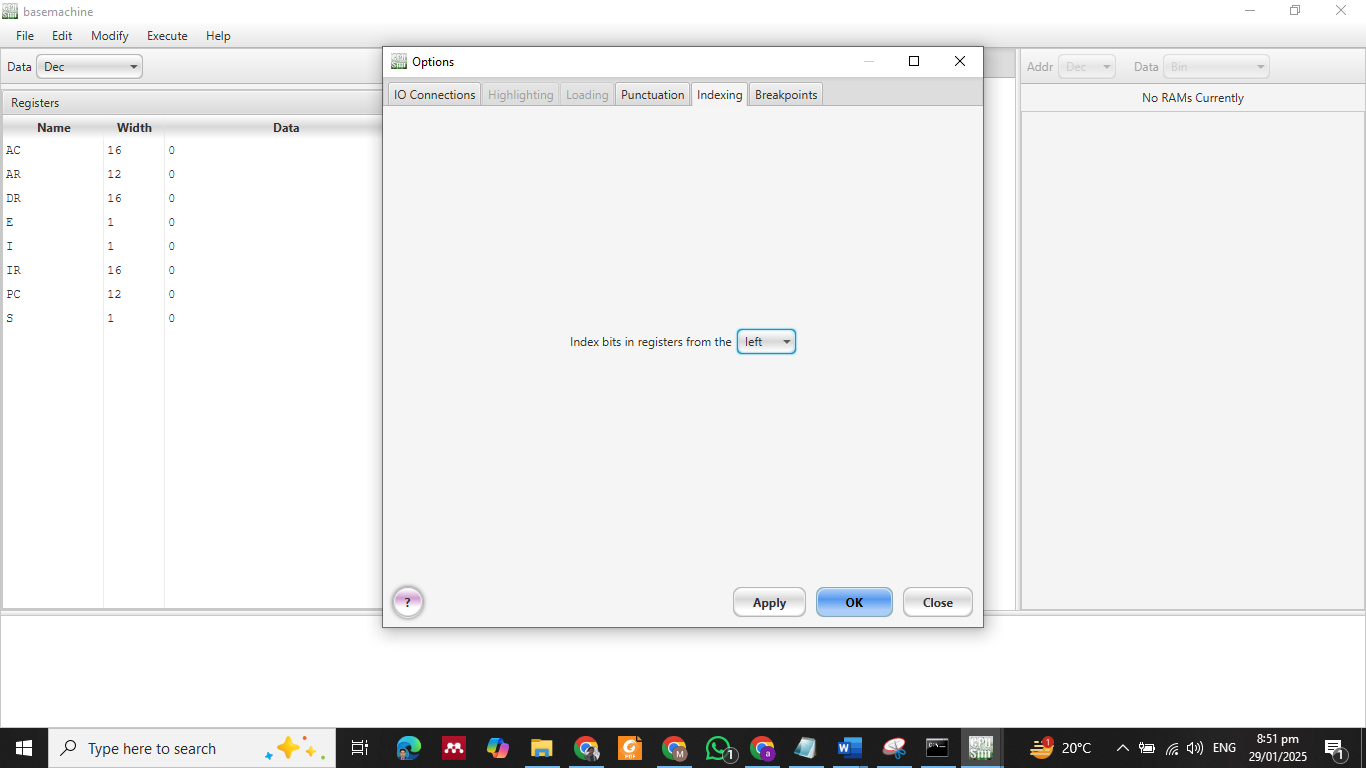
A screen shot of a computer

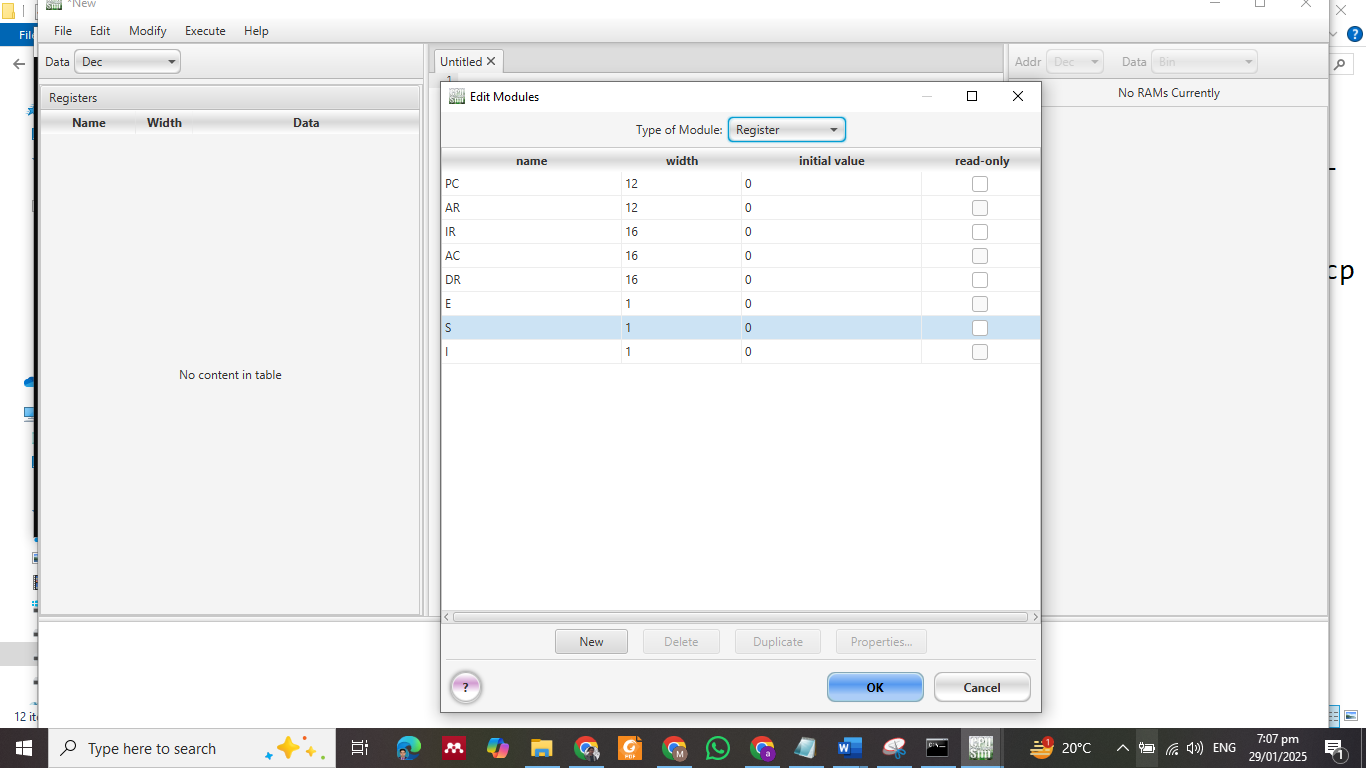
Description automatically generated

Figure 1 machine based on the following architecture

* Set the indexing from the left side.

Execute and then click on option and select indexing set it to left.





# Fetch Cycle:

The Fetch Cycle involves the following steps:

|  |  |
| --- | --- |
| **1** | AR <- PC |
| **2** | IR <- Main [AR] |
| **3** | PC - INCR |
| **4** | AR <- IR (4-15) |
| **5** | DECODE – IR |

1. Transfer the address from **PC to AR**.

A screenshot of a computer

Description automatically generated

1. Read the instruction from memory into **IR**.

A screenshot of a computer

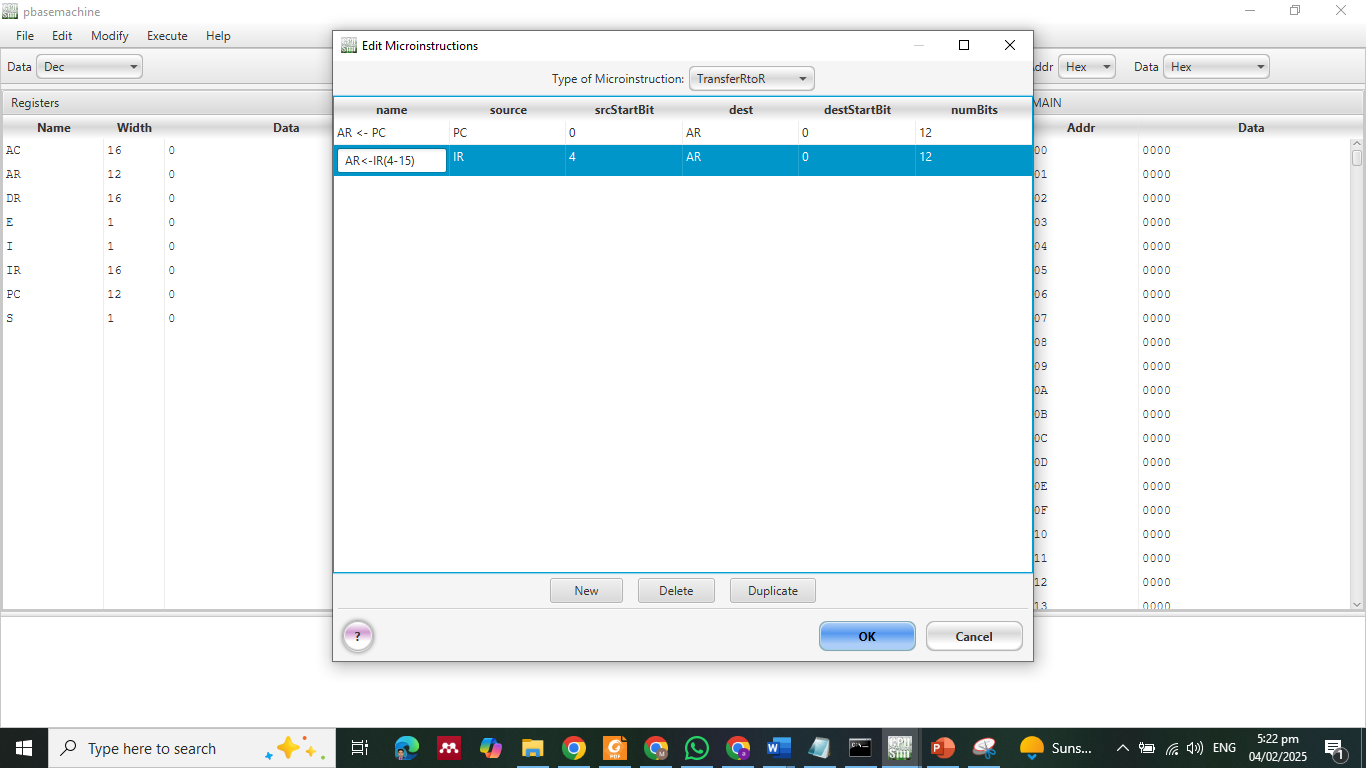
Description automatically generated

1. Increment **PC** to point to the next instruction.

A screenshot of a computer

Description automatically generated

1. Extract the address part of the instruction and transfer it to **AR**.

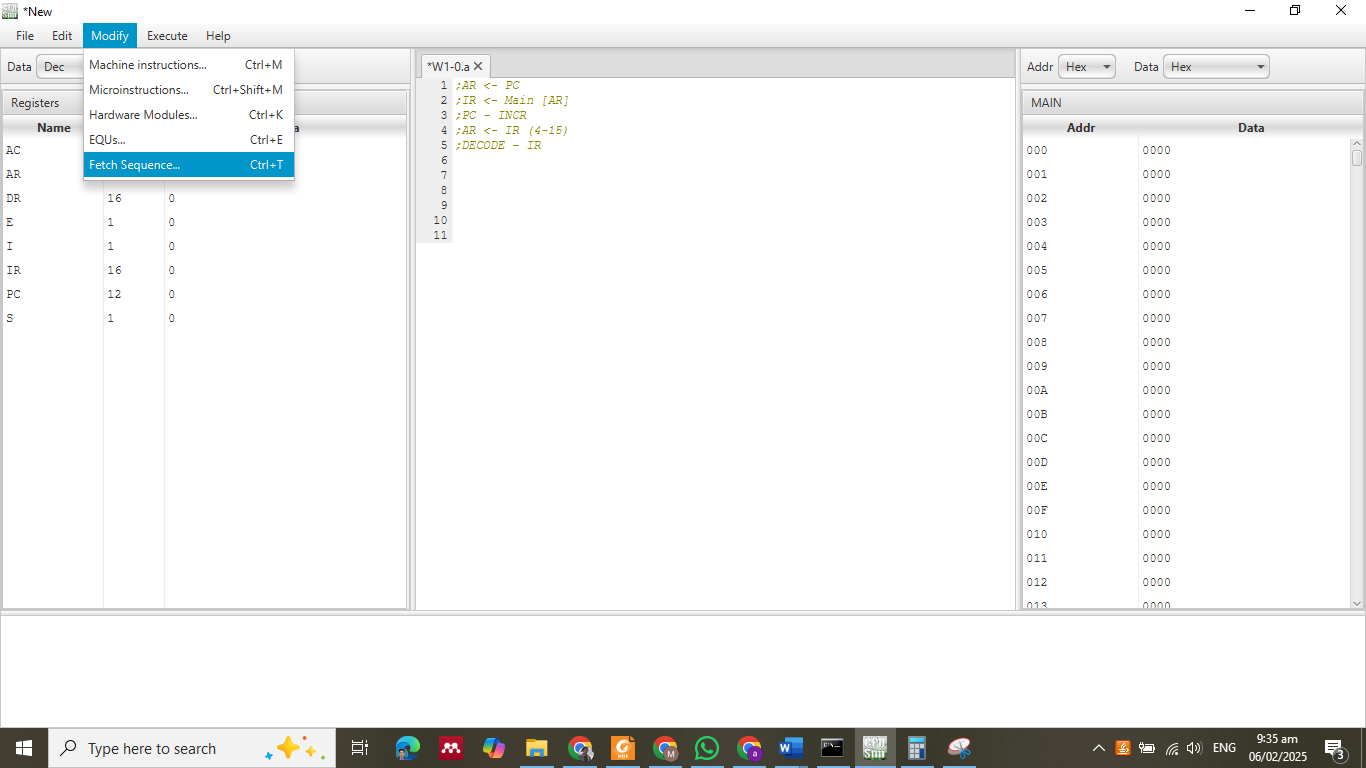


1. Decode the instruction for execution.

A screenshot of a computer

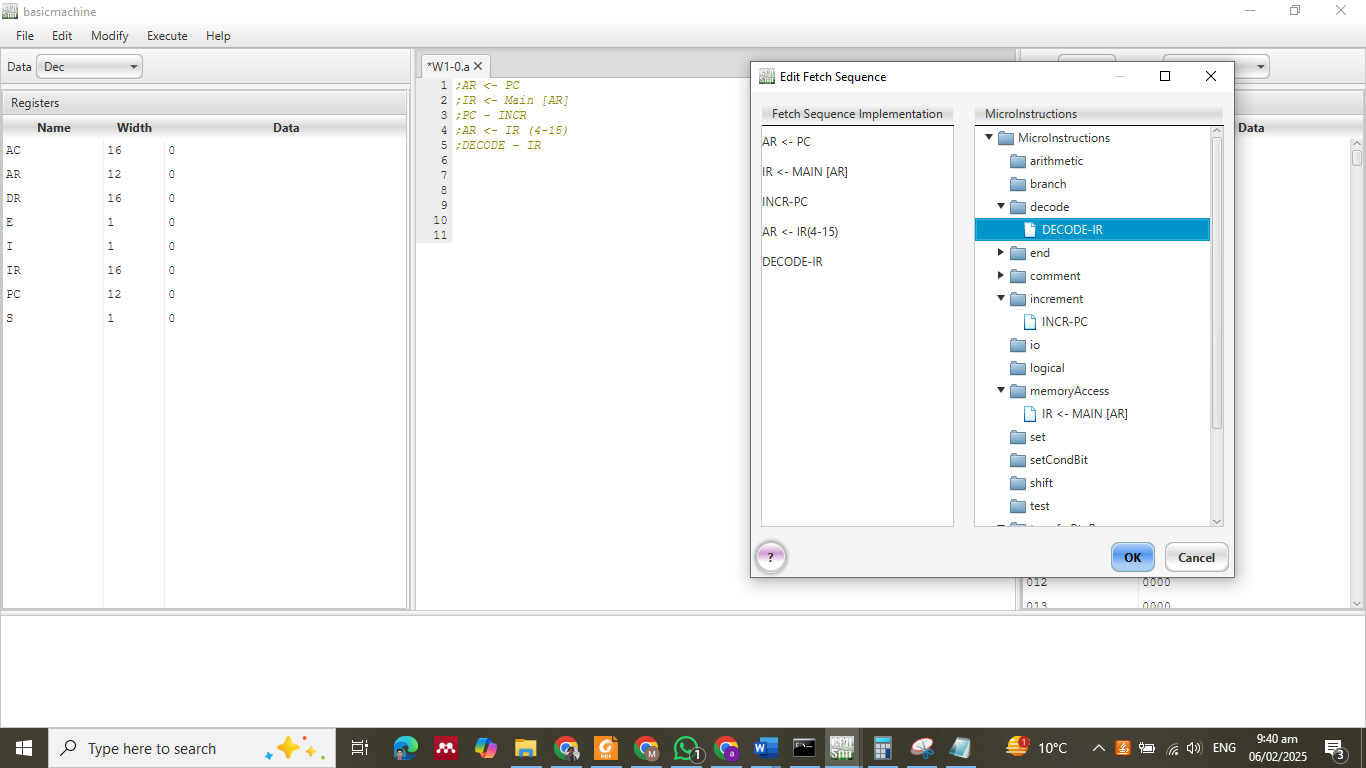
Description automatically generated

# Fetch Execution



A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Lab Tasks**

1. Write a **detailed explanation** of how the Fetch-Decode-Execute cycle works.
2. Use a simple instruction as an example and describe each step.
3. Explain the role of **PC, AR, IR, AC and DR in your own words.**
4. What is the function of the Arithmetic Logic Unit (ALU**)** in CPU operations?

How does ALU interact with registers and memory?

1. Create a new base machine and change the bit width of a register (e.g., make AC 8-bit instead of 16-bit)