

The background is a solid dark blue. Overlaid on this are several thin, gold-colored lines that form abstract, angular shapes. These lines radiate from the central text box, extending towards the corners and edges of the frame, creating a sense of dynamic movement or a network of connections.

EX/MEM Data Memory
MEM/WB WriteBack

TABLE OF CONTENTS

1. OVERVIEW OF MIPS PIPELINE

Have a quick look on the MIPS pipeline and focus on components we want to implement this week

2. EX/MEM & Data Memory

Implementing EX/MEM pipeline register and data memory

3. MEM/WB & WriteBack

Implementing the final stage and pipeline register which write from memory/register to a register

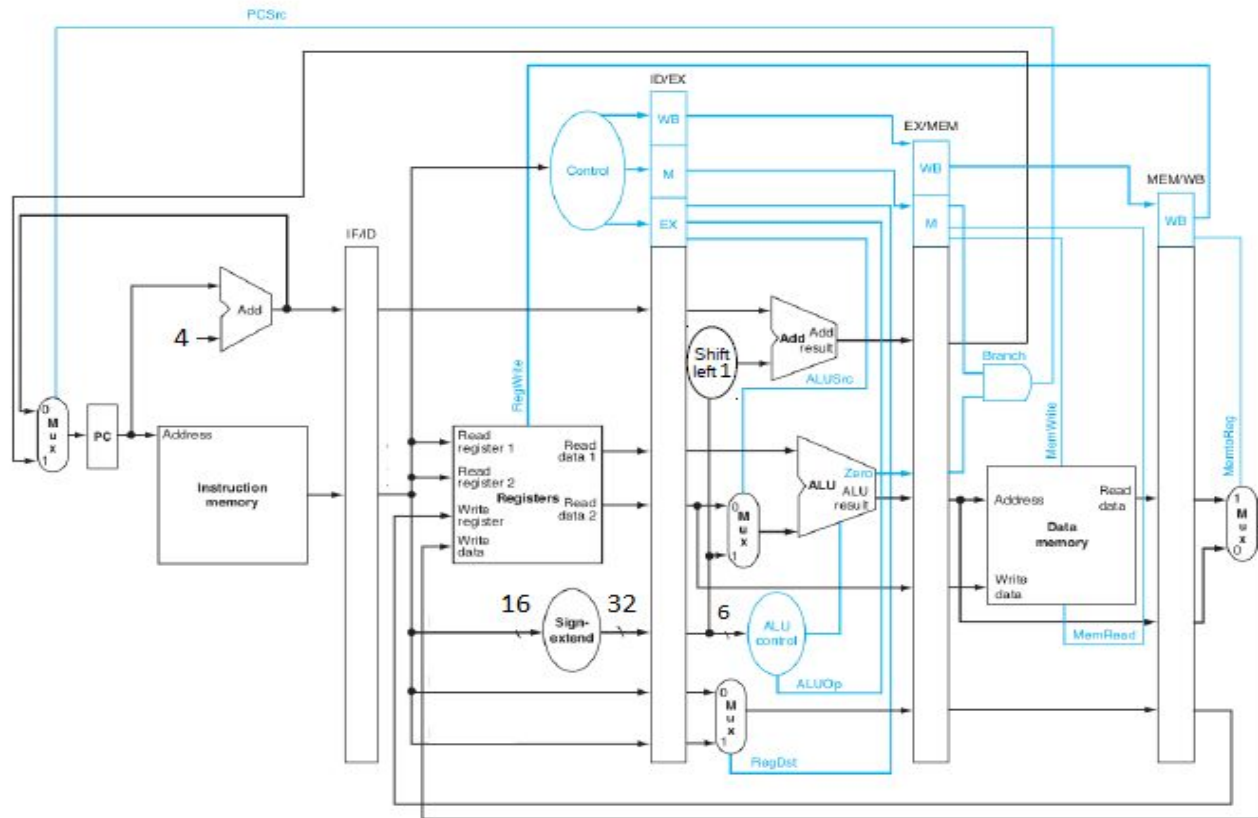
4. INTEGRATING ALL MODULES

Integrate all modules which have been implemented up to now

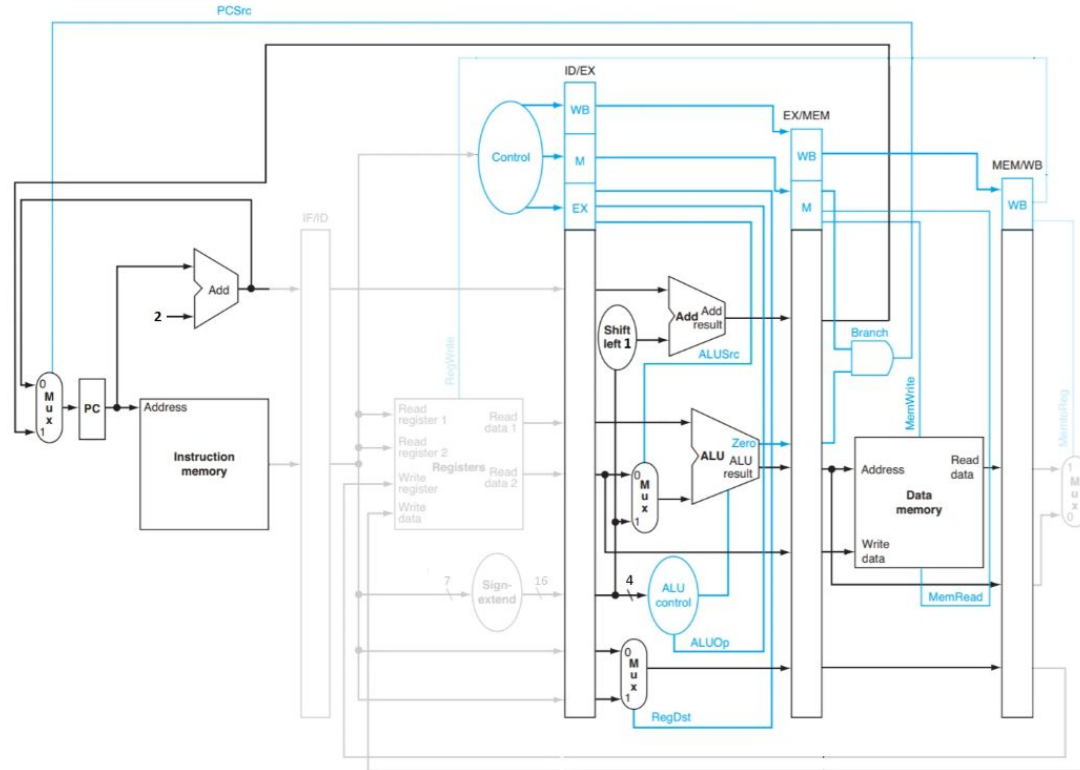
5. INPUTS & OUTPUTS

Describe inputs and outputs of required modules

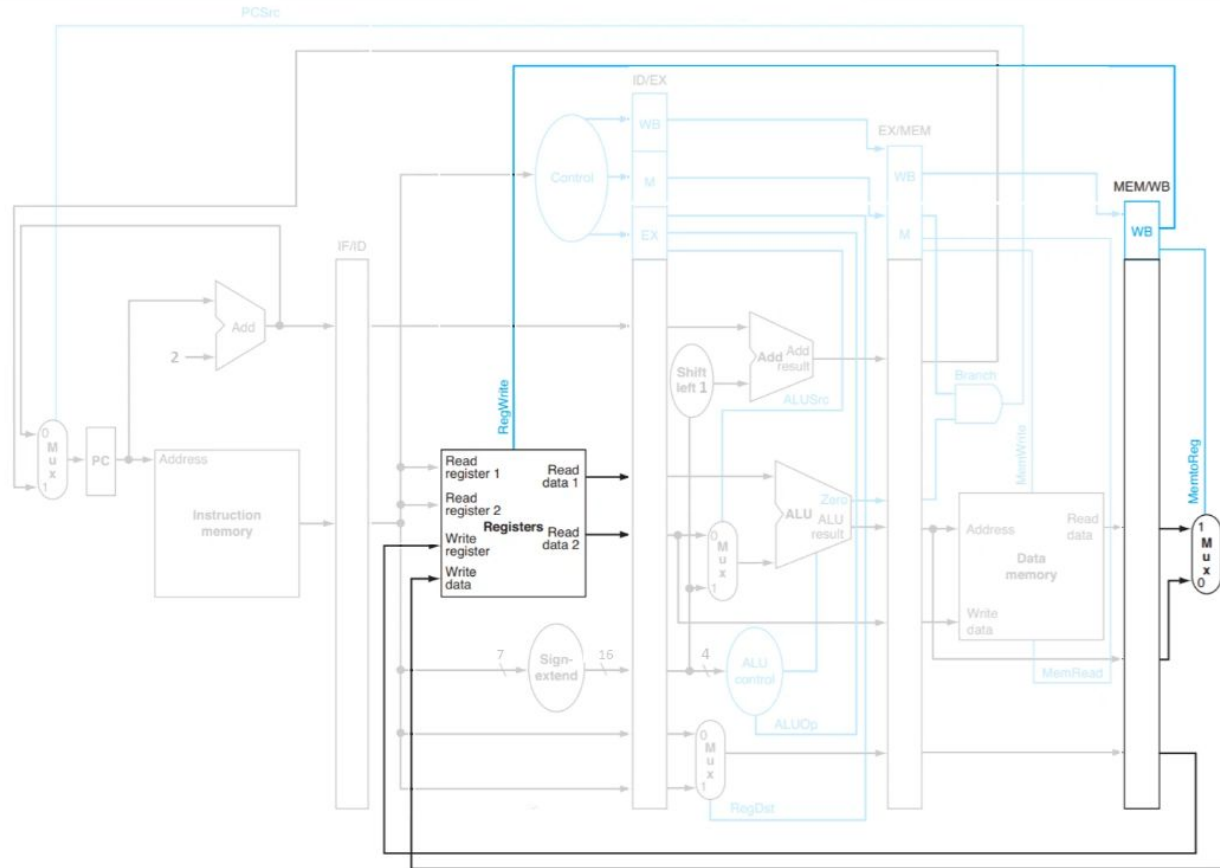
1. OVERVIEW OF MIPS PIPELINE



2. EX/MEM & Data Memory



3. MEM/WB & WriteBack



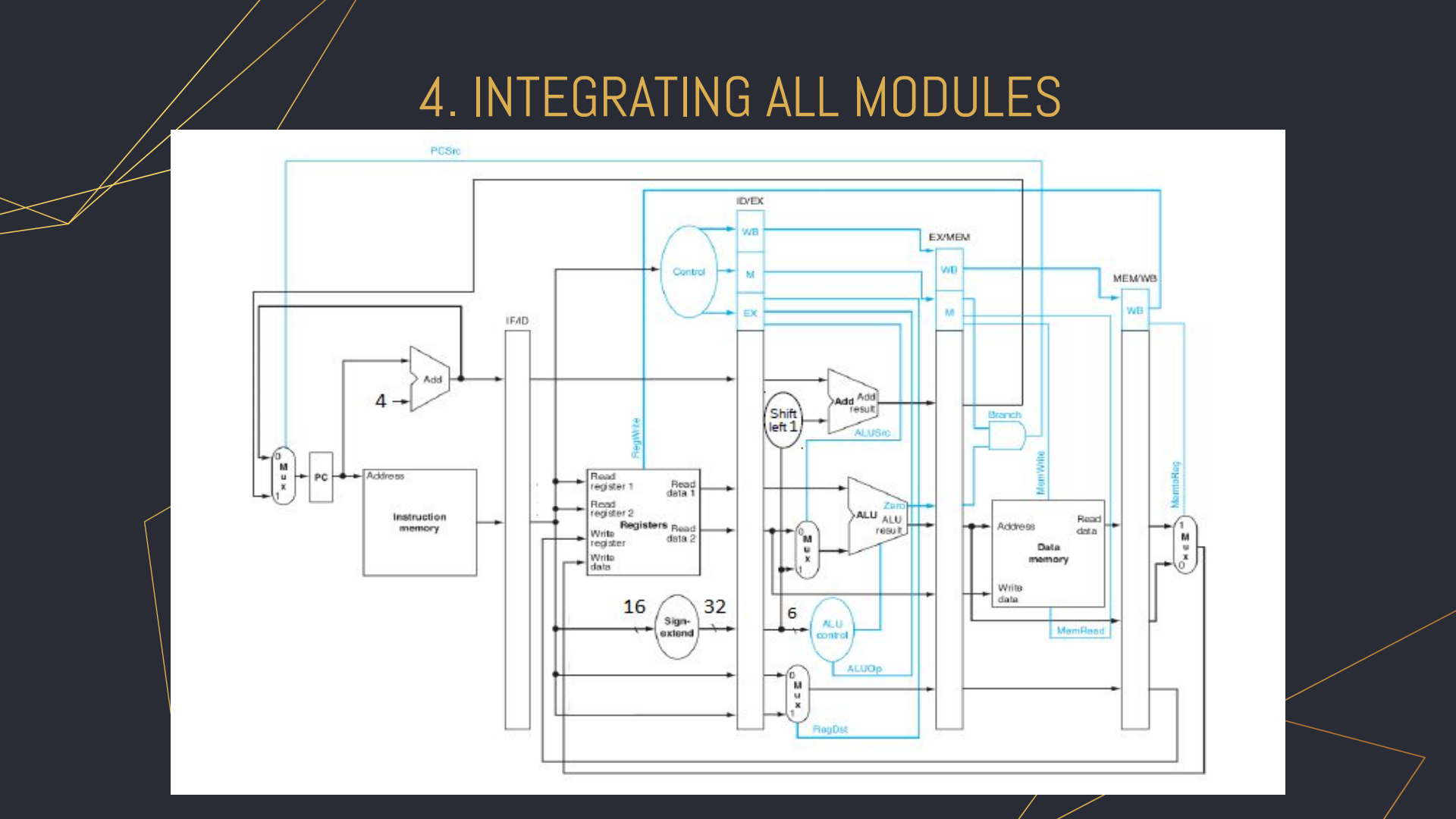
4. INTEGRATING ALL MODULES

The diagram illustrates the integration of all modules in a CPU, showing the flow of data and control signals through the stages of the pipeline: IF/ID, ID/EX, EX/MEM, and MEM/WB.

Key Components and Connections:

- IF/ID Stage:** The Program Counter (PC) is updated by adding 4 to its current value. The new PC value is used to address Instruction Memory. The instruction is then decoded into control signals (RegWrite, Branch, MemWrite, MemRead) and register indices (Rt, Rn, Rd).
- ID/EX Stage:** The Register File is accessed to read data from registers Rt and Rn. The ALU control is determined by the instruction's ALUOp field. The ALUSrc is selected between register data and an immediate value (which is sign-extended from 16 to 32 bits).
- EX/MEM Stage:** The ALU performs the operation specified by ALU control on the ALUSrc. The result is shifted left by 1 and then zero-extended to 32 bits. This result is used for branching and for writing back to the Register File (if RegWrite is enabled).
- MEM/WB Stage:** The ALU result is used to address Data Memory (if MemRead is enabled). The data from Data Memory is then written back to the Register File (if MemWrite is enabled).
- Control and Status Signals:**
 - PCSrc:** Selects between the new PC value and the ALU result for the next cycle.
 - RegWrite:** Enables writing the ALU result back to the Register File.
 - Branch:** Selects between the ALU result and the PC value for the next cycle.
 - MemWrite:** Enables writing data from the Register File to Data Memory.
 - MemRead:** Enables reading data from Data Memory.

The diagram shows the complex interconnections between these modules, ensuring that data and control signals flow correctly through the pipeline stages.



5. INPUTS & OUTPUTS

Finally, it's time to instantiate the modules and connect them together. Modules which need to be instantiated and connected are as follows:

EX/MEM Register:

- Inputs: clk(1), hit(1), branchTarget(32), zeroFlag(1), ALUResult(32), readData2(32), writeReg(5), MemRead(1), MemWrite(1), Branch(1), RegWrite(1), MemToReg(1)
- Outputs: branchTargetOut(32), zeroFlagOut(1), ALUResultOut(32), readData2Out(32), writeRegOut(5), MemReadOut(1), MemWriteOut(1), BranchOut(1), RegWriteOut(1), MemToRegOut(1), hitOut(1)

Data Memory:

- Inputs: clk(1), address(32), writeData(32), MemRead(1), MemWrite(1)
- Outputs: readData(32)

5. INPUTS & OUTPUTS (Cont'd)

MEM/WB Register:

- Inputs: clk(1), hit(1), readData(32), ALUResult(32), writeReg(5), RegWrite(1), MemtoReg(1)
- Outputs: hitOut(1), readDataOut(32), ALUResultOut(32), writeRegOut(5), RegWriteOut(1), MemtoRegOut(1)

WriteBack:

- Inputs: MemtoReg(1), readData(32), ALUResult(32)
- Outputs: writeData(32)