Design & Implementation of a 16-Bit Single-Cycle Processor

A Project Report on the Virtual Architecture Simulator

Course: SCE 409: Computer Architecture
Instructor: Ass. Prof. Khaled Elshafey

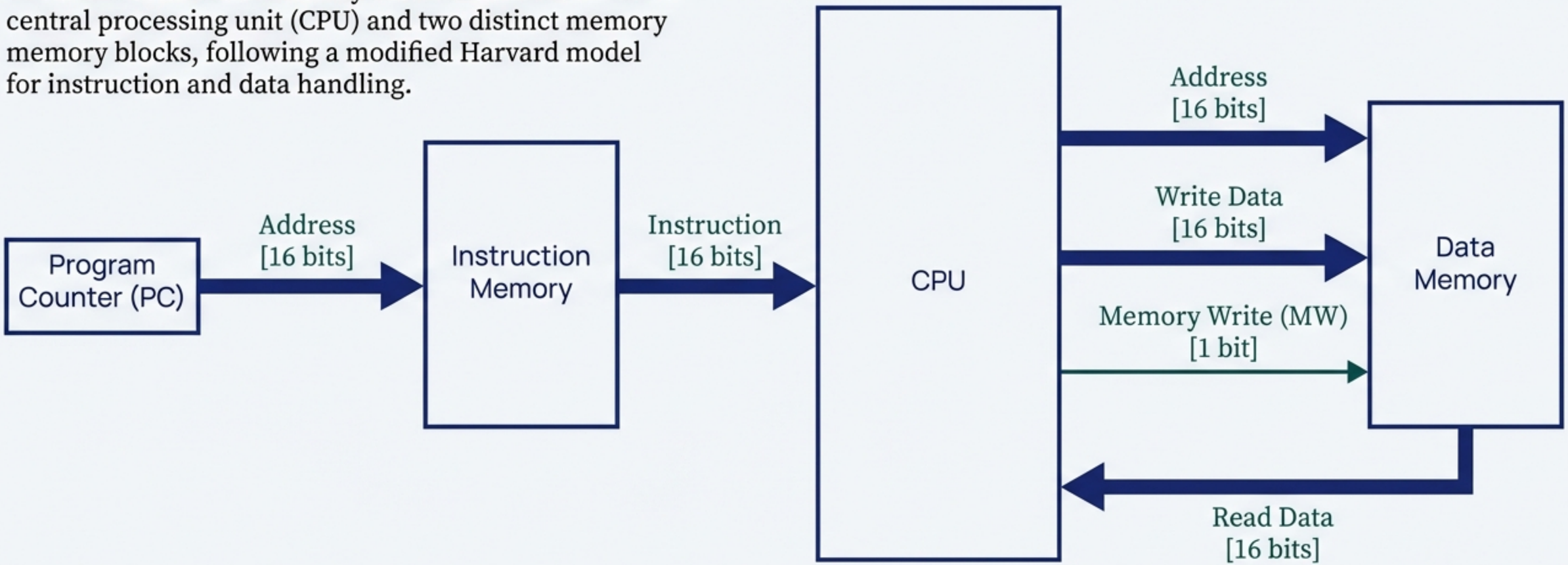
Team Members

Mostafa Khaled Mohamed Hassanin (73)
Mahmoud Mohamed Ahmed Hussein (70)
Mohamed Ramadan Saeed Mahmoud (56)
Mohamed Shaaban Abdel Maqsoud El-Shenawy (57)
Ahmed Hosny Abdel Hameed (7)

NotebookLM

System Architecture Overview: A Single-Cycle Design

Our design implements a 16-bit single-cycle computer. The architecture is defined by the interaction between a central processing unit (CPU) and two distinct memory memory blocks, following a modified Harvard model for instruction and data handling.

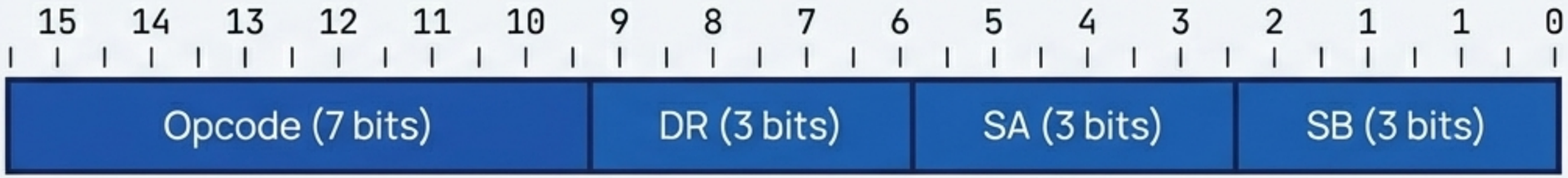


NotebookLM

Defining the Language: The Instruction Set Architecture (ISA)

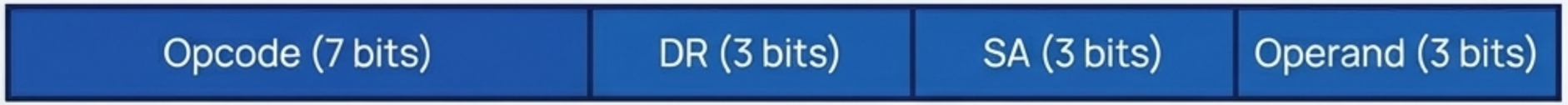
The hardware is designed to execute a specific set of instructions, categorized into three distinct formats. Each format structures the 16-bit instruction word differently to handle register-based operations, immediate values, and program flow control.

Register Addressing Mode



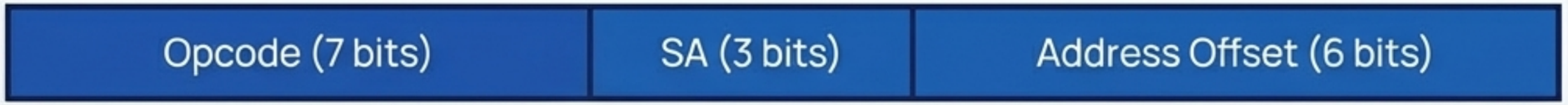
Purpose: For operations between registers (e.g., R1 ← R2 – R3)

Immediate Addressing Mode



Purpose: For operations involving a constant value (e.g., R2 ← R7 + 3)

Jump & Branch Mode

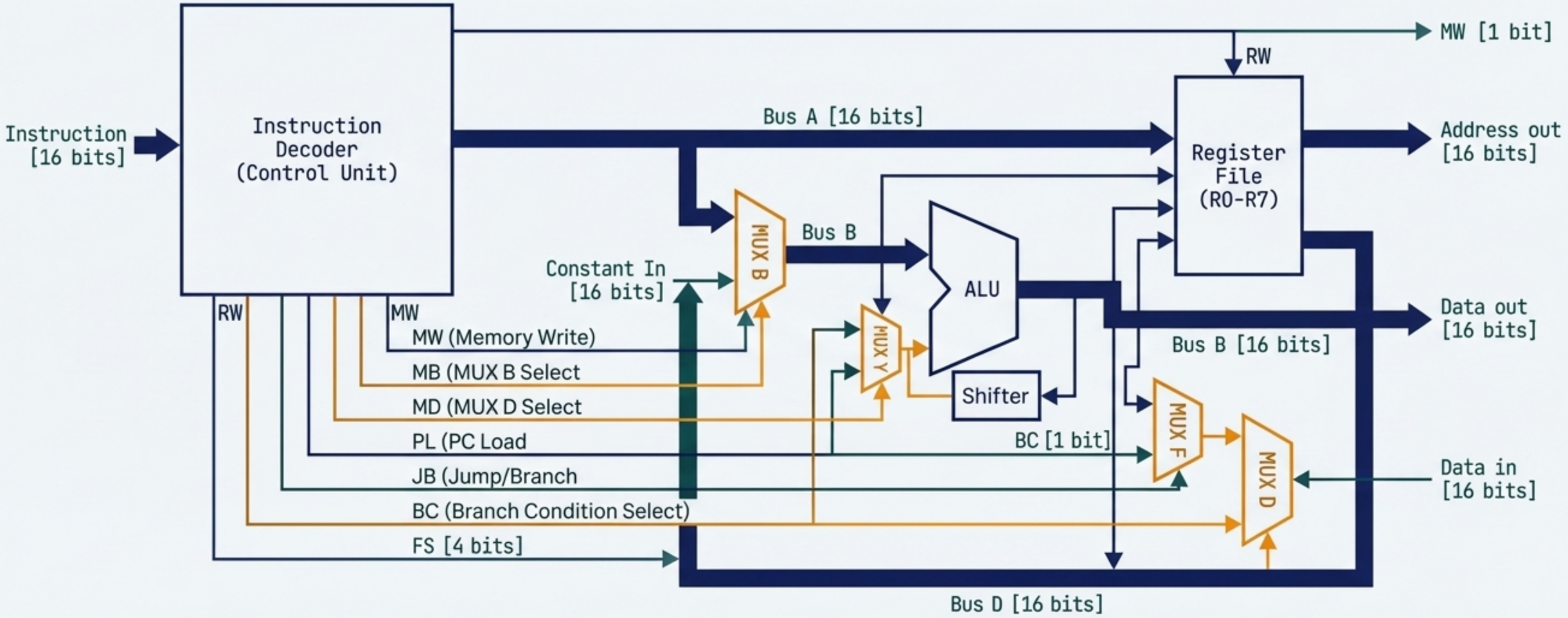


Purpose: For altering program flow, using PC-relative addressing (e.g., Branch if R6 == 0)

NotebookLM

The Processor Blueprint: Datapath and Control

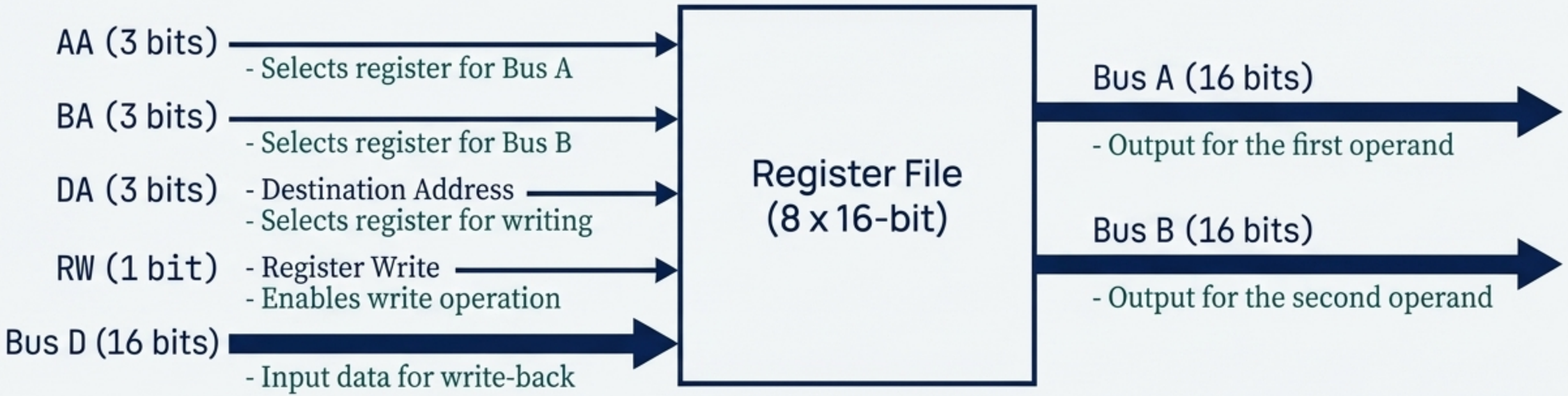
The CPU is partitioned into a Datapath, which holds and processes data, and a Control Unit, which directs the datapath's operations based on the current instruction. This diagram illustrates all major functional units and data pathways.



NotebookLM

The Datapath Core: The Register File

The Register File is a small, fast memory array of eight 16-bit registers (R0-R7) that serves as the CPU's immediate workspace. All ALU operations are performed on data held within these registers.

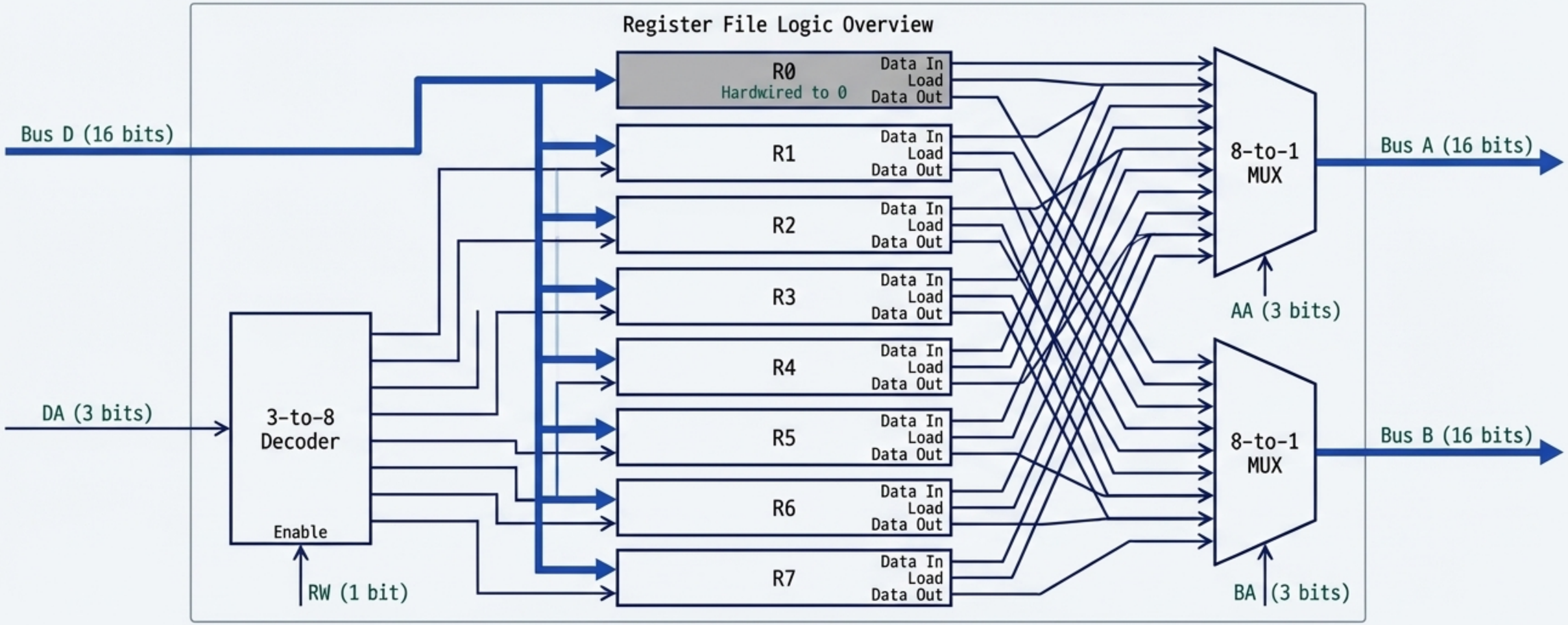


Note: Register R0 is hardwired to the value zero.

NotebookLM

Inside the Register File: Decoding and Selection Logic

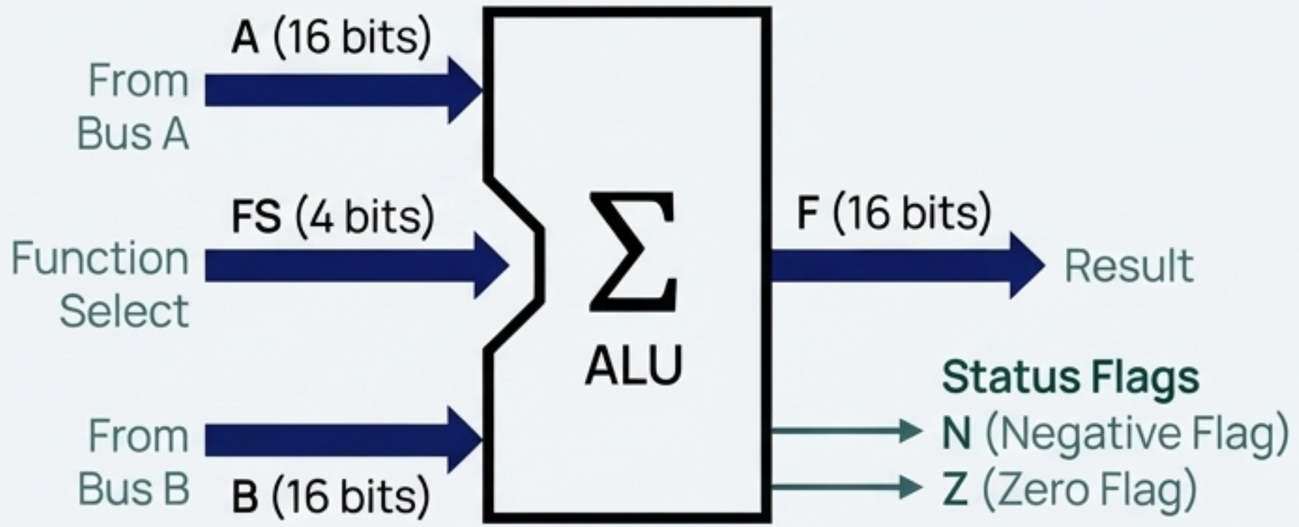
The read and write operations are implemented with standard combinational logic. Two 8-to-1 multiplexers select the registers for reading based on addresses AA and BA, while a 3-to-8 decoder enables a single register for writing based on address DA.



NotebookLM

The Processing Engine: The Arithmetic Logic Unit (ALU)

The ALU is a combinational circuit that performs all arithmetic (e.g., add, subtract) and bitwise logic (e.g., AND, OR, XOR) microoperations. Its function is determined by a control word from the Control Unit.

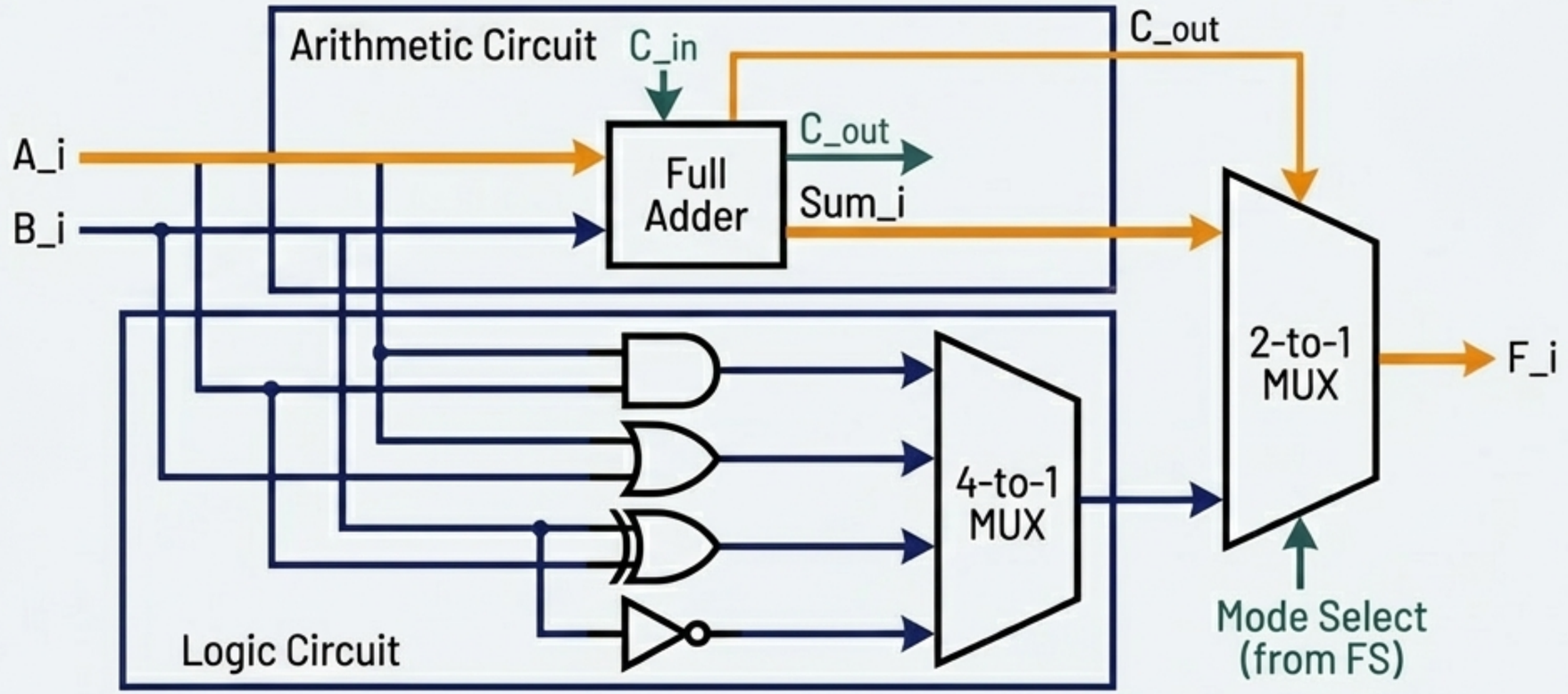


FS Code	Operation
0000	F = A ;
0010	F = A + B ;
0101	F = A AND B ;
1100	F = NOT A |

NotebookLM

Inside the ALU: Arithmetic and Logic Circuits

The ALU is composed of two parallel processing sections. The arithmetic circuit uses a series of full adders for calculations, while the logic circuit uses arrays of basic gates for bitwise operations. A final multiplexer selects the output based on the operation type.

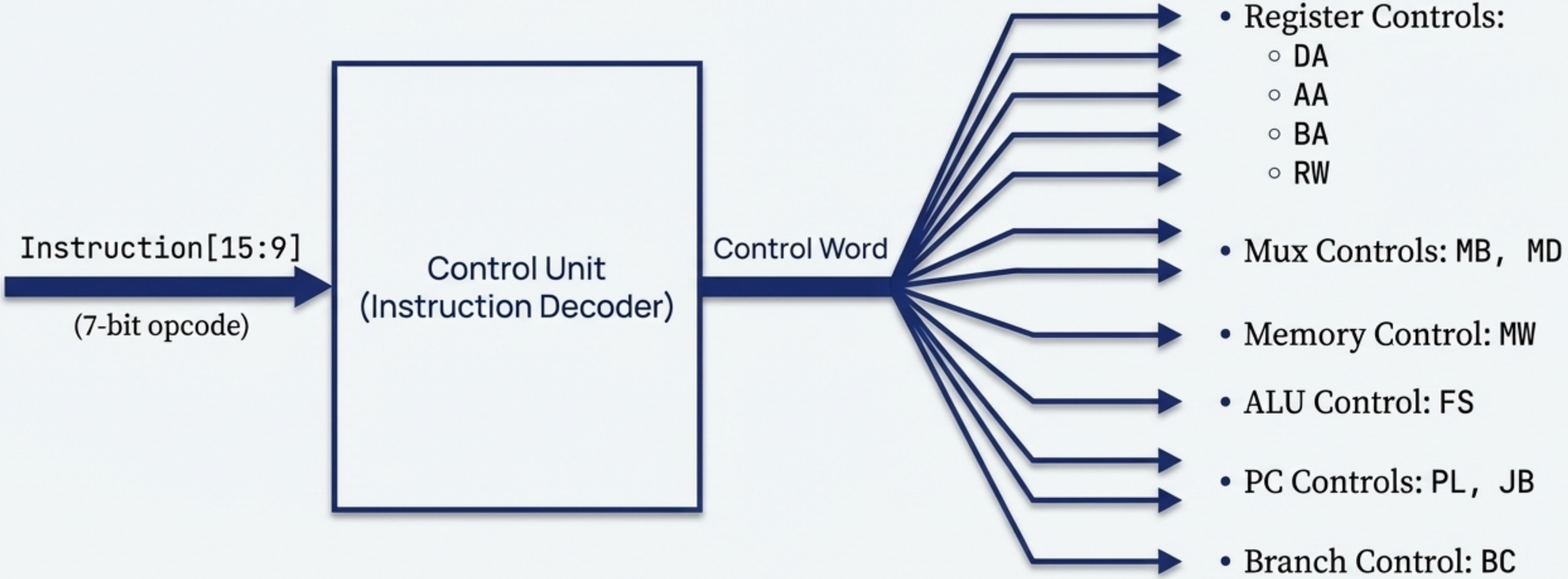


The full 16-bit ALU is constructed by replicating this 1-bit slice 16 times.

NotebookLM

The Director: The Hardwired Control Unit

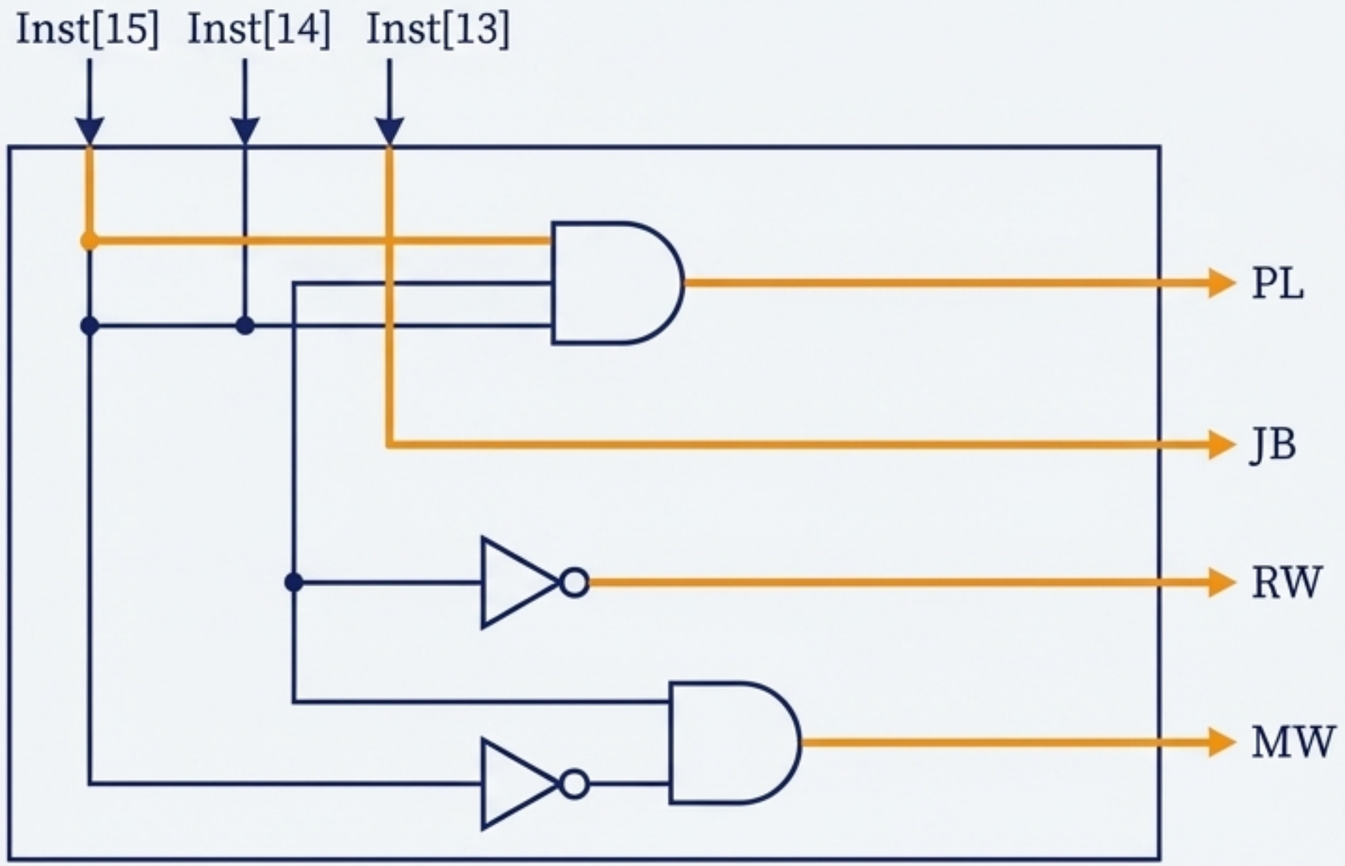
The Control Unit is the brain of the processor. It decodes the opcode from the current instruction and generates a "Control Word"—a set of binary signals that configure the datapath for the required operation. In our design, this is implemented as a combinational logic circuit (hardwired control).



NotebookLM

Inside the Control Unit: The Instruction Decoder Logic

The decoder is a combinational circuit that maps specific opcode bit patterns directly to the output control signals. Each control signal is generated by a simple logic function of the opcode bits.



This hardwired approach is fast and simple, **directly translating the instruction's intent into hardware configuration.**

NotebookLM

Formalizing Control: The Decoder's Boolean Logic

The behavior of the hardwired control unit can be precisely described by a set of Boolean equations. Each control signal is a direct function of the instruction's opcode bits.

```
 PL = Inst[15] • Inst[14]
 JB = Inst[13]
MB = Inst[15]
MD = Inst[13]
MW = Inst[14] • Inst[15]'
RW = Inst[14]'
BC = Inst[9]
DA = Inst[2:0]
AA = Inst[8:6]
BA = Inst[5:3]
```

(Selects between N and Z flags for conditional branches)

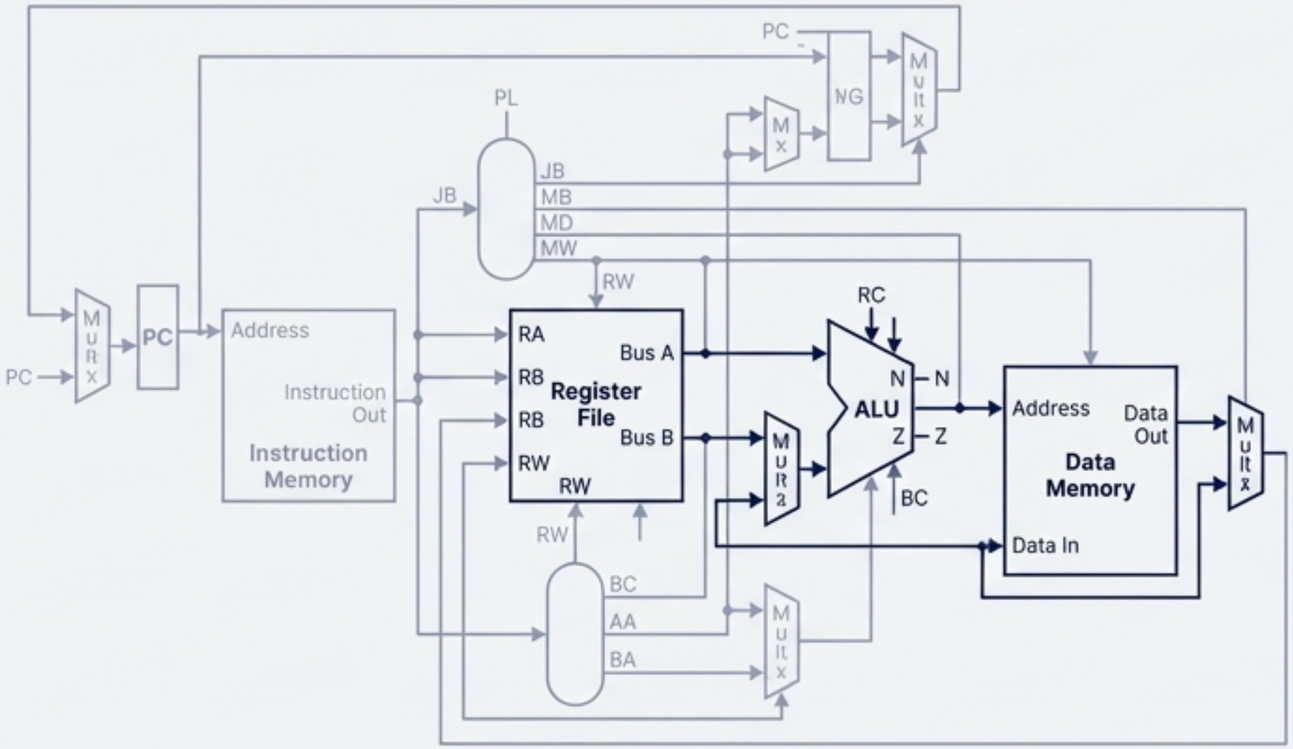
(These fields are passed directly through)

The equations demonstrate the direct mapping from instruction format to hardware control, which is the essence of a single-cycle design.

NotebookLM

System Validation: Tracing an Instruction's Lifecycle

To demonstrate the integrated functionality of the datapath and control unit, we will trace the execution of a single Register-format instruction: `ADD R3`. This operation adds the contents of R2 and R3 and stores the result in R1.



Instruction Details: `ADD R1, R2, R3`

Assembly: `ADD R1, R2, R3`

Semantic Action: `R[1] ← R[2] + R[3]`

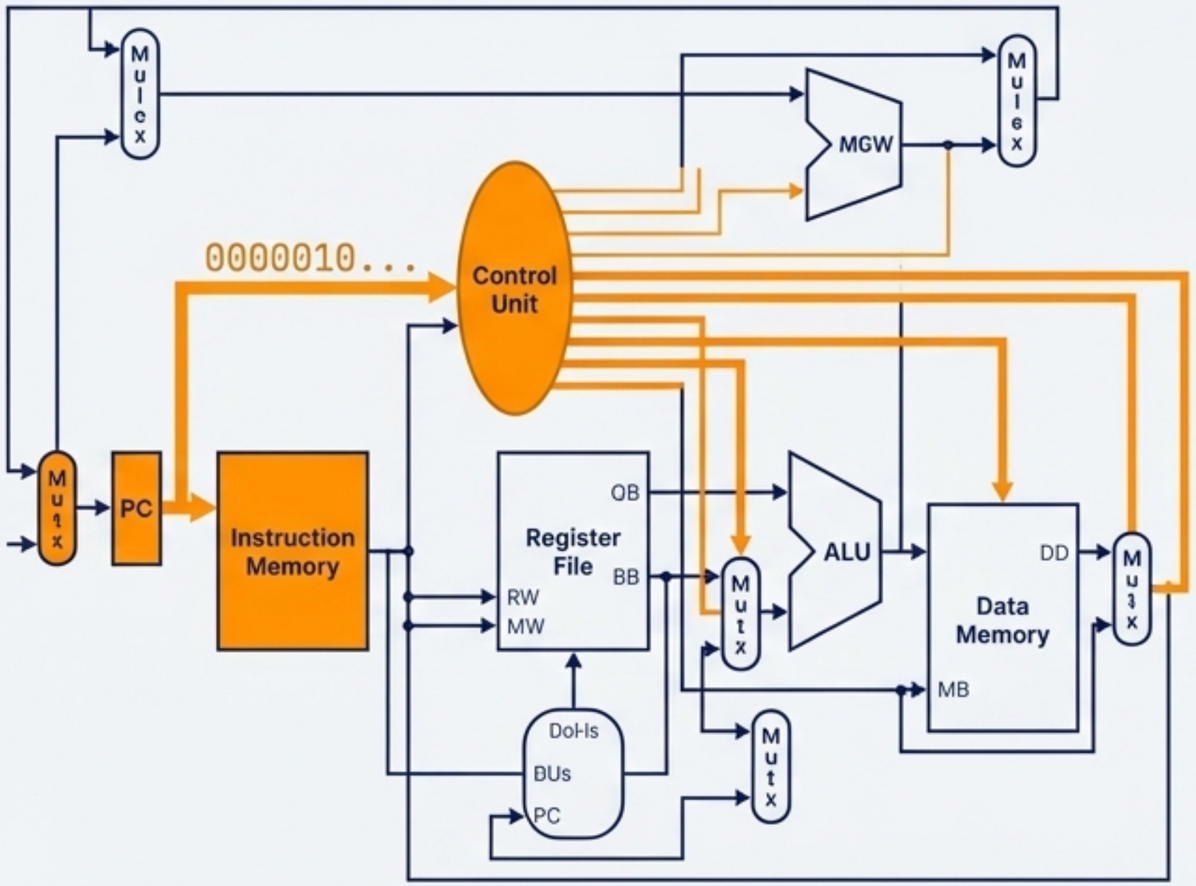
Binary Representation:

Opcode	DR	SA	SB
0000010	001	010	011

Full 16-bit Instruction: `0000 010 001 010 011`

NotebookLM

Instruction Trace Part 1: Fetch and Decode



1. Fetch Stage

The instruction is retrieved from Instruction Memory using the address in the Program Counter (PC).

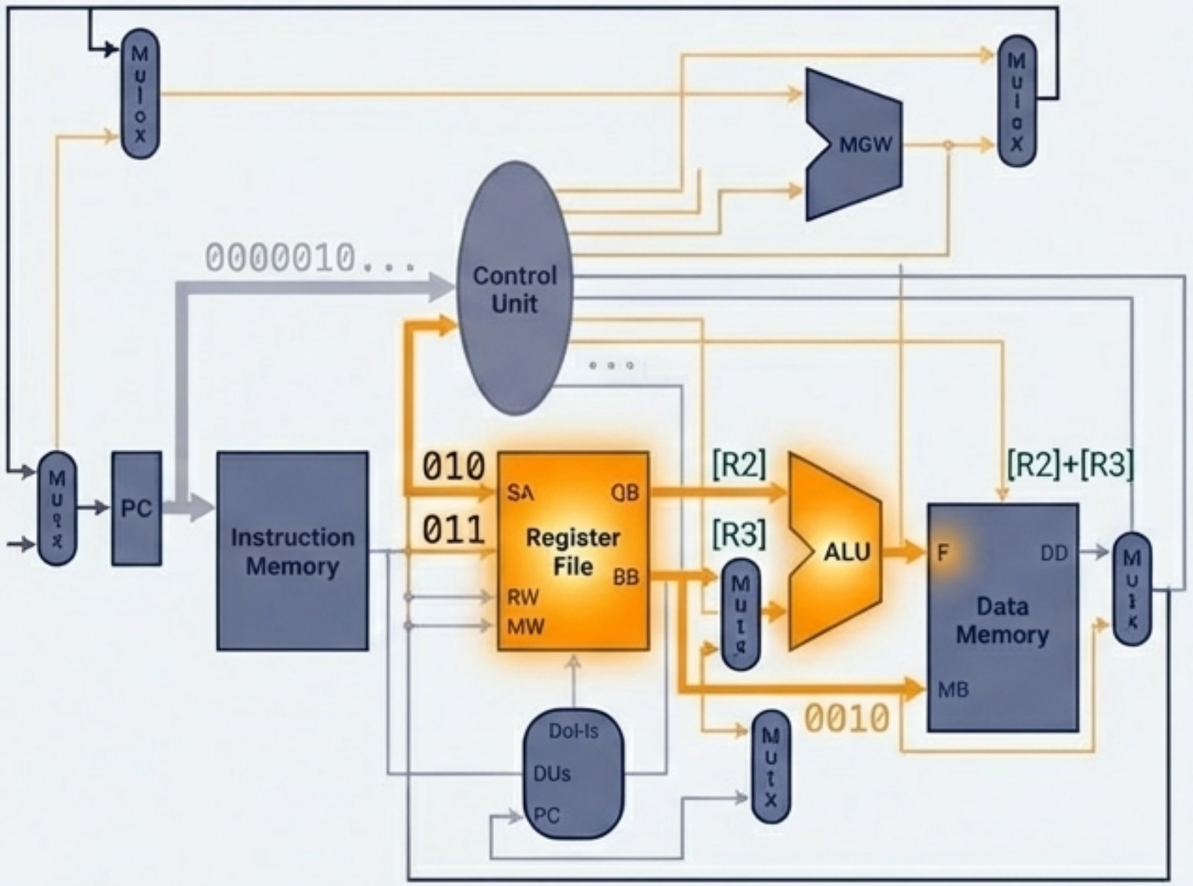
2. Decode Stage

The Control Unit decodes the instruction's opcode (`0000010`) and generates the following control signals:

Signal	Value	Reason
RW	1	An ADD writes a result to a register.
MW	0	No memory write is performed.
MB	0	The ALU operand comes from a register (Bus B).
MD	0	The write-back data comes from the ALU.
PL	0	Not a branch or jump; PC increments normally.
FS	0010	The ALU function code for ADD.

NotebookLM

Instruction Trace Part 2: Execute



3. Operand Fetch

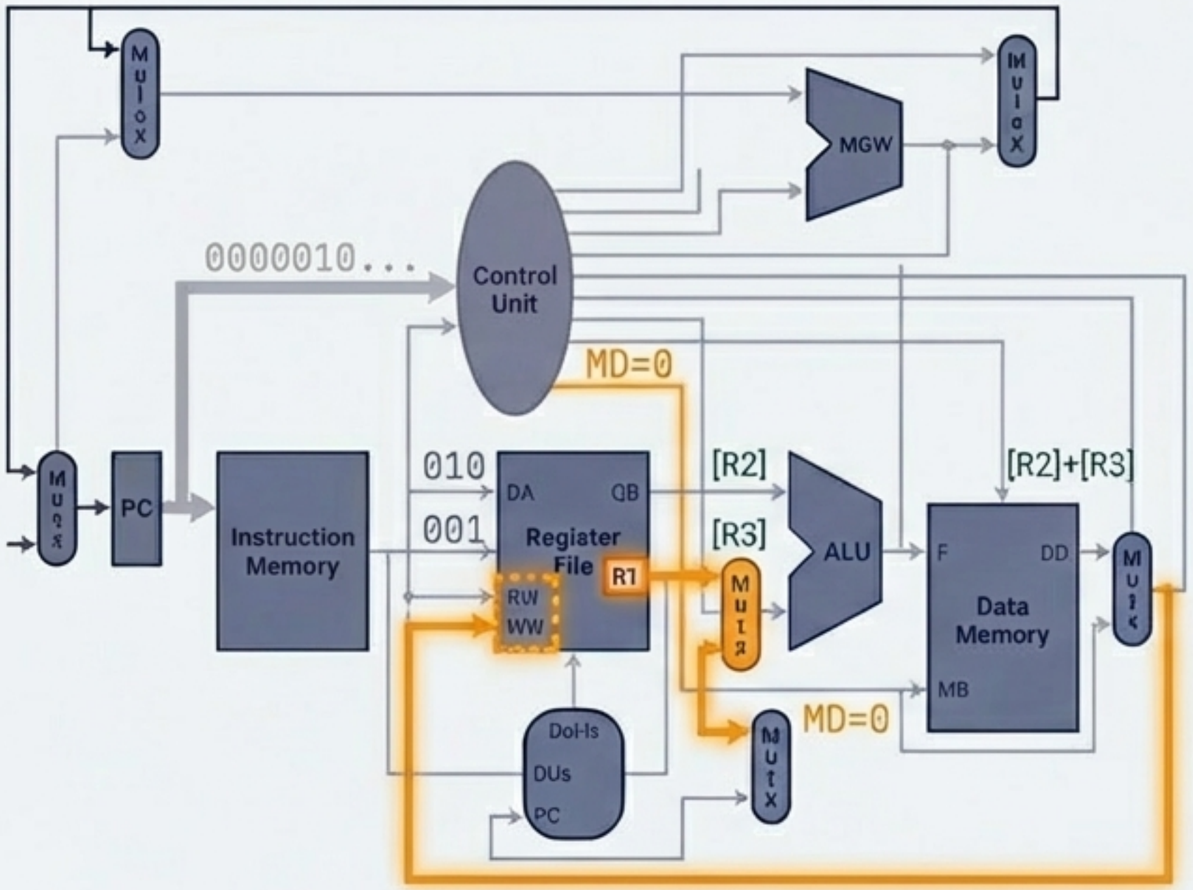
The register addresses `SA (R2)` and `BA (R3)` are sent to the Register File. The contents of these registers are read onto Bus A and Bus B.

4. ALU Operation

With the function code `FS=0010`, the ALU receives the values from Bus A and Bus B and performs an addition. The result is placed on the ALU's output.

NotebookLM

Instruction Trace Part 3: Write-Back



5. Write-Back Stage

The ALU result is routed through MUX D onto Bus D. The destination address `DA (R1)` and the `RW=1` signal instruct the Register File to write the data from Bus D into register R1, completing the operation.

The successful completion of the Fetch, Decode, Execute, and Write-Back stages for the ADD instruction validates the integrity of the single-cycle design.

NotebookLM