Microarchitecture Design

# Introduction

AgriCore is a 32-bit, single-issue, in-order processor targeting low-power agricultural applications such as image and sensor data processing. It supports RISC-V-like instruction formats (R, I, S, B, J) with floating-point, vector/SIMD, and string extensions. Design goals include small area, low latency for ML/NLP inference, and minimal power usage.

# Incremental Datapath Design

Each instruction format (R, I, S, B/J, FP, Vector/String) adds incremental datapath components: - R-type: ALU + RegFile + WriteBack MUX. - I-type: ALU + ImmGen + D-Cache (for loads). - S-type: ALU + ImmGen + D-Cache (write). - B/J-type: Comparator + Branch target adder + PC MUX. - FP: FPU + FP RegFile. - Vector/String: SIMD lanes and string buffer units. All are later unified into a

5-stage pipeline (IF, ID, EX, MEM, WB).

# Unified Single-Cycle Datapath

Combines PC, instruction/data caches, ALU, FPU, Vector, and String units. Control signals (RegWrite, MemRead, MemWrite, ALUSrc, etc.) coordinate data flow across the pipeline stages.

# Control Unit Design

Control signals generated from opcode/funct fields: RegWrite, MemRead, MemWrite, Branch, Jump, FPUEnable, VectorEnable, StringEnable. Multi-cycle FPU and Vector ops are managed via FSM sequencing. Static not-taken branch prediction minimizes control hazard cost.

# Pipeline and Hazards

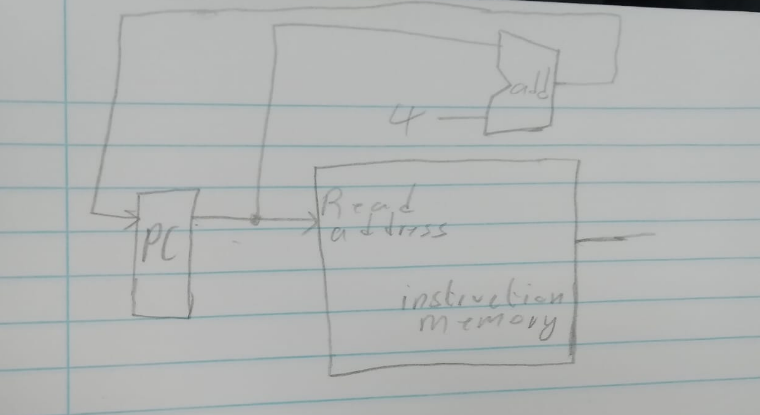
Pipeline: IF–ID–EX–MEM–WB. Forwarding resolves most RAW hazards. Load-use hazards insert single-cycle stalls. Branch mispredictions flush IF/ID. Multi-cycle FPU/Vector units assert busy/stall signals when active.

# Memory Hierarchy

8KB I-Cache (direct-mapped), 16KB D-Cache (2-way), both 1-cycle hit, ~10-cycle miss. Vector memory supports gather/scatter buffering. No coherence required (single core). Cache bypass for image-streaming operations.

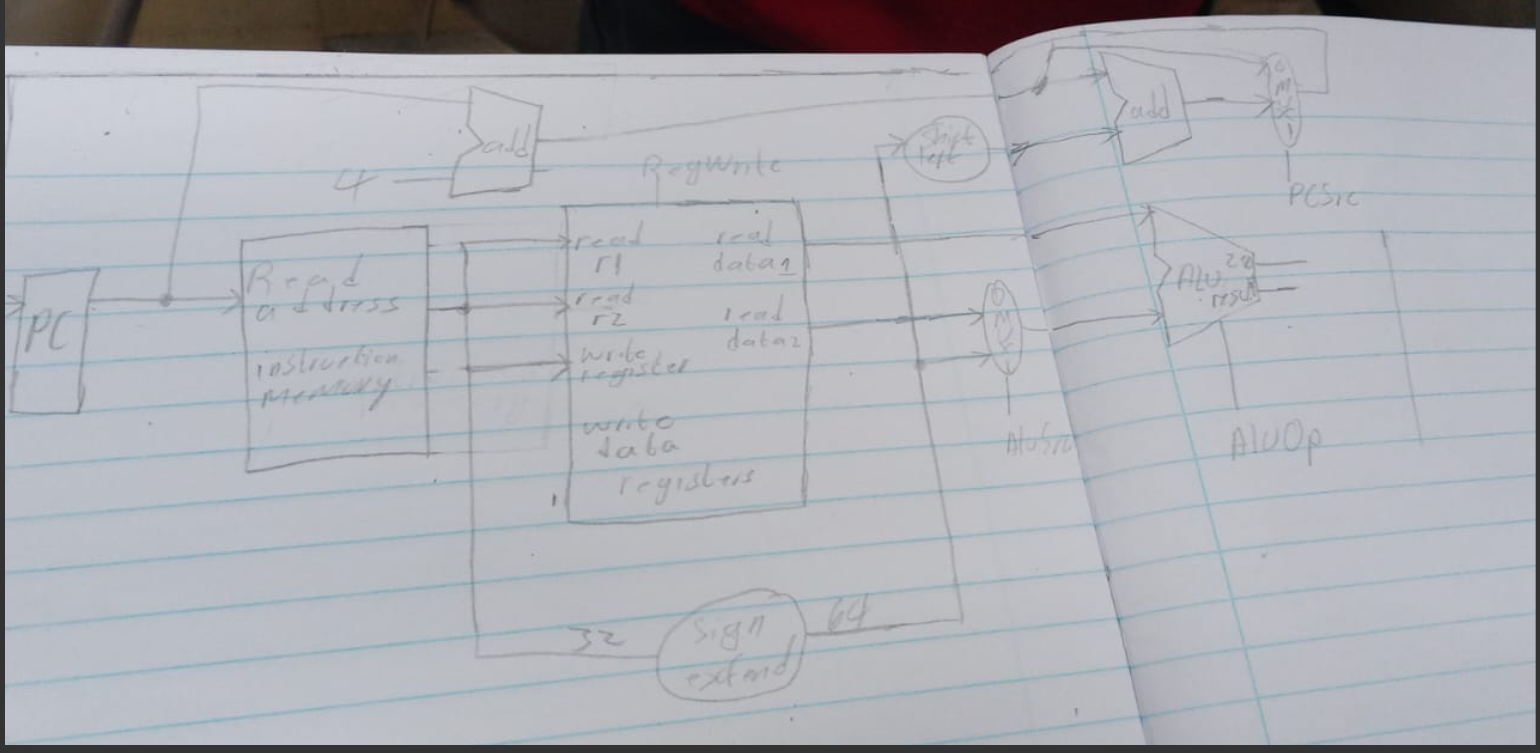
Now Our Diagrams and their explanation to get what they do:

The First Diagram: Instruction Fetch



Here the PC fetches the address of the current instruction from the Memory and then points to the next address.

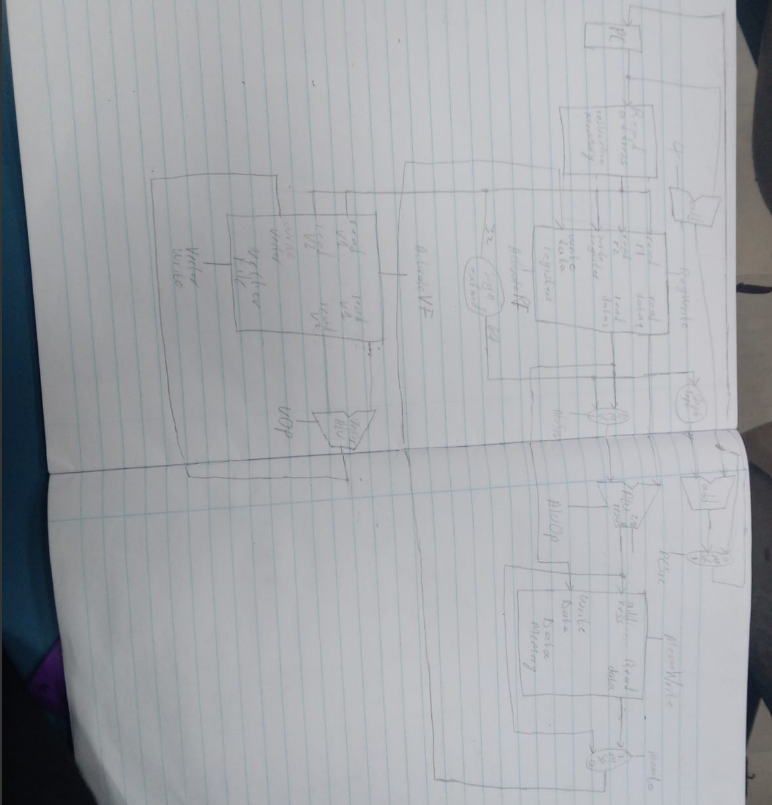
Second Diagram: R-type, I-Type and Branch



R-type (Resgister type), they are responsible for the making the adding, Subtracting and multiplying.

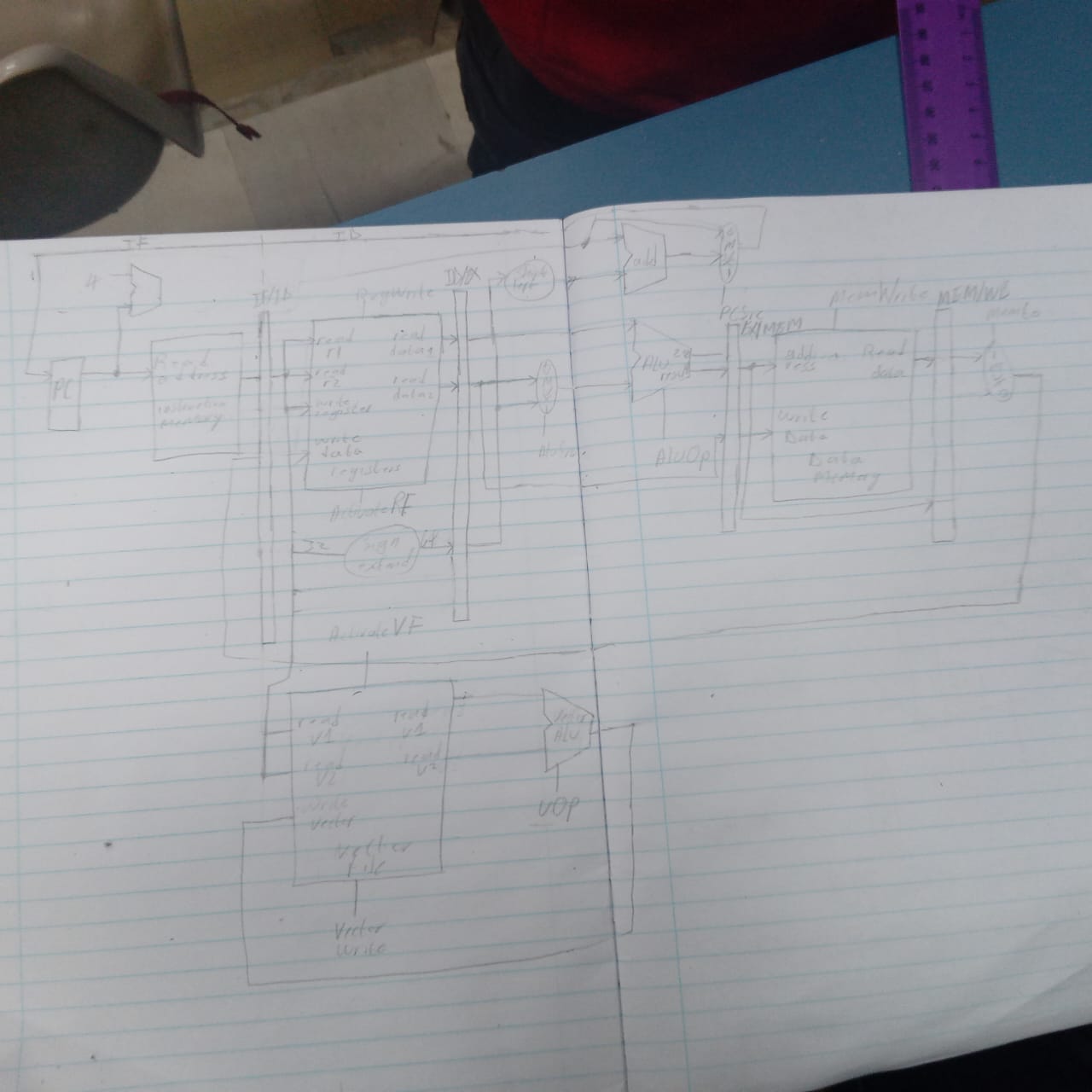
I-type , this one is responsible for taking the value from the memory and then sign extend it.

Third Diagram:



This diagram is able to do the vector operation designate by vector operation control line (VOP), i.e. addition , subtraction and multiplication. It even helps in running the matrix operations.

Forth Diagram: Now the Pipeline Implementation:



The pipelining helps in reducing time for each instruction to complete and the instructions do not take a whole clock cycle to complete.

# Conclusion

AgriCore, structured per CS3520 template, achieves energy-efficient performance for

domain-specific workloads. It balances simplicity and speed with a 5-stage pipeline, small caches, and vector/string accelerators for ML and NLP in agriculture.

**Control Unit Truth Table**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction Class** | **RegWrite** | **MemRead** | **MemWrite** | **ALUSrc** | **Branch** | **Jump** | **FPU** | **Vector** | **String** |
| R-type | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| I-type | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Load | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Store | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| Branch | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| JAL/JALR | 1 | 0 | 0 | X | 0 | 1 | 0 | 0 | 0 |
| FPU | 1 | 0 | 0 | X | 0 | 0 | 1 | 0 | 0 |
| Vector | 1 | 0/1 | 0/1 | X | 0 | 0 | 0 | 1 | 0 |
| String | 1 | 0/1 | 0 | X | 0 | 0 | 0 | 0 | 1 |