Microarchitecture

Topics

- Introduction
- Performance Analysis
- Single-Cycle Processor
- Multicycle Processor
- Pipelined Processor
- Exceptions
- Advanced Microarchitecture

Introduction

- Microarchitecture: how to implement an architecture in hardware
- Processor:
  - Datapath: functional blocks
  - Control: control signals

Application Software	programs
Operating Systems	device drivers
Architecture	instructions registers
Micro-architecture	datapaths controllers
Logic	adders memories
Digital Circuits	AND gates NOT gates
Analog Circuits	amplifiers filters
Devices	transistors diodes
Physics	electrons

Space and Time Metrics

- Two important metrics for any program
  - Space: How much memory does the program code and data require? (Memory footprint)
  - Time: What is the execution time for the program?
- Different design methodologies
  - CISC
  - RISC
- Memory footprint and execution time are not necessarily correlated

What determines execution time?

- **Execution time = ($\sum$ CPI$_j$) * clock cycle time**, where $1 \leq j \leq n$
- **Execution time = n * CPI$_{Avg}$ * clock cycle time**, where n is the number of instructions (executed not static instruction count)

Instruction Frequency

- *Static* instruction frequency refers to number of times a particular instruction occurs in compiled code.
  - Impacts memory footprint
  - If a particular instruction appears a lot in a program, can try to optimize amount of space it occupies by clever instruction encoding techniques in the instruction format.
- *Dynamic* instruction frequency refers to number of times a particular instruction is executed when program is run.
  - Impacts execution time of program
  - If dynamic frequency of an instruction is high then can try to make enhancements to datapath and control to ensure that CPI taken for its execution is minimized.

Benchmarks

- **Benchmarks** are a set of programs that are representative of the workload for a processor.
- The key difficulty is to be sure that the benchmark program selected really are representative.
- A radical new design is hard to benchmark because there may not yet be a compiler or much code.

SPECint2006

12 programs for quantifying performance of processors on integer programs

Intel Core 2 Duo
E6850 (3 GHz)

Program name	Description	Time in seconds
400.perlbench	Applications in Perl	510
401.bzip2	Data compression	602
403.gcc	C Compiler	382
429.mcf	Optimization	328
445.gobmk	Game based on AI	548
456.hmmer	Gene sequencing	593
458.sjeng	Chess based on AI	679
462.libquantum	Quantum computing	422
464.h264ref	Video compression	708
471.omnetpp	Discrete event simulation	362
473.astar	Path-finding algorithm	466
483.xalancbmk	XML processing	302

Processor Performance

- Program execution time

Execution Time = (# instructions)(cycles/instruction)(seconds/cycle)

- Definitions:
  - Cycles/instruction = CPI
  - Seconds/cycle = clock period
  - 1/CPI = Instructions/cycle = IPC
- Challenge is to satisfy constraints of:
  - Cost
  - Power
  - Performance

Increasing the Processor Performance

- **Execution time = n * CPI_{Avg} * clock cycle time**
- Reduction in the number of executed instructions
- Datapath organization leading to lower CPI
- Increasing clock speed

Speedup

- Assume a base case execution time of 10 sec.
- Assume an improved case execution time of 5 sec.
- Percent improvement = (base-new)/base
- Percent improvement = (base-new)/new
- Percent improvement = (10-5)/5 = 100%
- Speedup = base/new
- Speedup = 10/5 = 2
- Speedup is preferred by advertising copy writers

Recall: What is Computer Architecture?

- **Computer Architecture:**

The science and art of designing, selecting, and interconnecting hardware components and designing the hardware/software interface to create a computing system that meets functional, performance, energy consumption, cost, and other specific goals.

- **Traditional definition:**

The term architecture is used here to describe the attributes of a system as seen by the programmer, i.e., the conceptual structure and functional behavior as distinct fror dataflow and controls, the logic design, implementation." Gene Amdahl, IBM 1964

Dr. Amdahl holding a 100gate LSI air-cooled chip. On his desk is a circuit board with the chips on it. This circuit board was for an Amdahl 470 V/6 (photograph dated March 1973).

Amdahl’s Law

Amdahl's law, named after computer architect Gene Amdahl, is used to find the maximum expected improvement to an overall system when only part of the system is improved.

Amdhal's law can be interpreted more technically, but in simplest terms it means that it is the algorithm that decides the speedup not the number of processors.

Wikipedia

Amdahl's Law

- f: Parallelizable fraction of a program
- P: Number of processors

$$\text{Speedup} = \frac{1}{1 - f + \frac{f}{P}}$$

- Maximum speedup limited by serial portion:

  Serial bottleneck

Increasing the Throughput of the Processor

- Don’t focus on trying to speedup individual instructions
- Instead focus on throughput i.e. number of instructions executed per unit time

Microarchitecture

- Multiple implementations for a single architecture:
  - Single-cycle
    - Each instruction executes in a single cycle
  - Multicycle
    - Each instruction is broken up into a series of shorter steps
  - Pipelined
    - Each instruction is broken up into a series of steps
    - Multiple instructions execute at once.

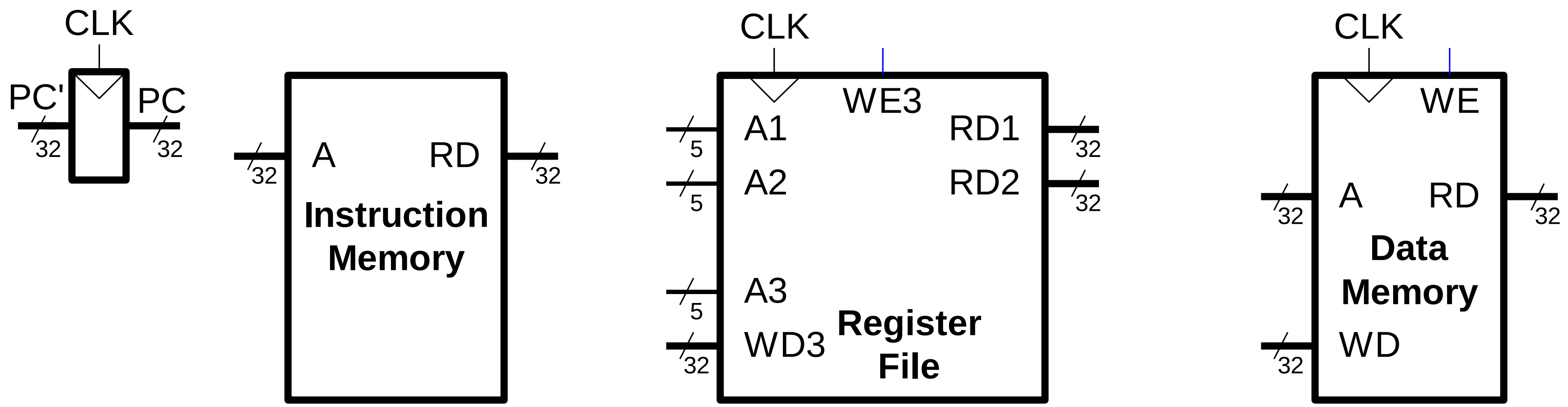
MIPS Processor

- We consider a subset of MIPS instructions:
  - R-type instructions: `and`, `or`, `add`, `sub`, `slt`
  - Memory instructions: `lw`, `sw`
  - Branch instructions: `beq`
- Later consider adding `addi` and `j`

Architectural State

- Determines everything about a processor:
  - `PC`
  - 32 registers
  - Memory

MIPS State Elements

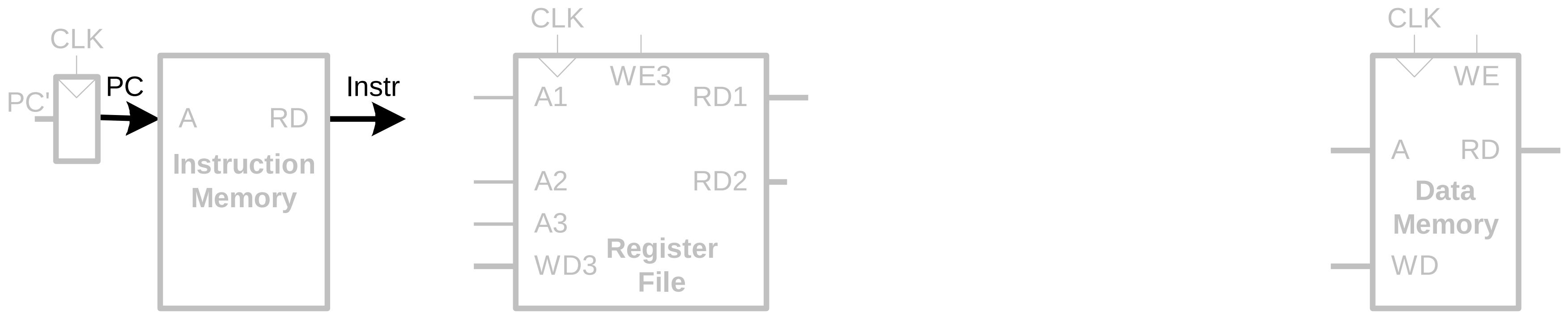


Single-Cycle MIPS Processor

- Datapath
- Control

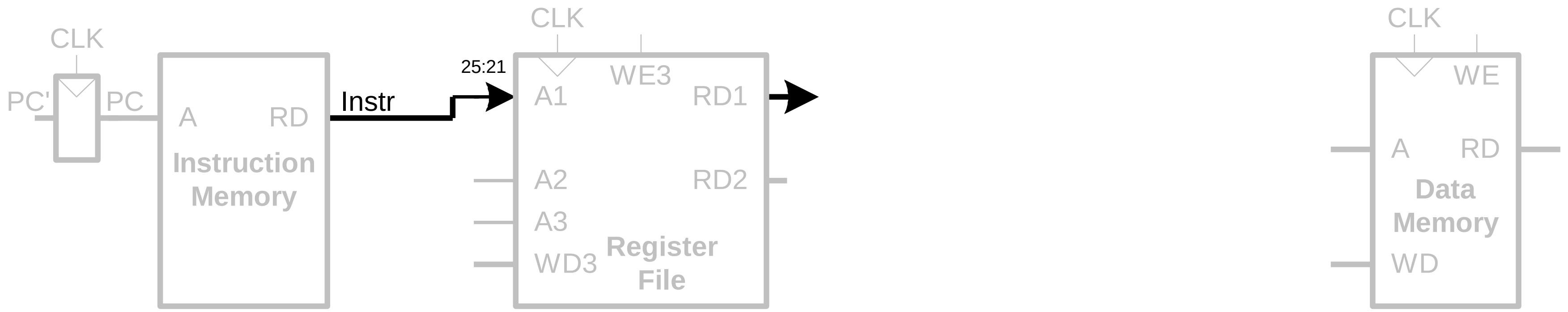
Single-Cycle Datapath: `lw` fetch

- First consider executing `lw`
- **STEP 1:** Fetch instruction



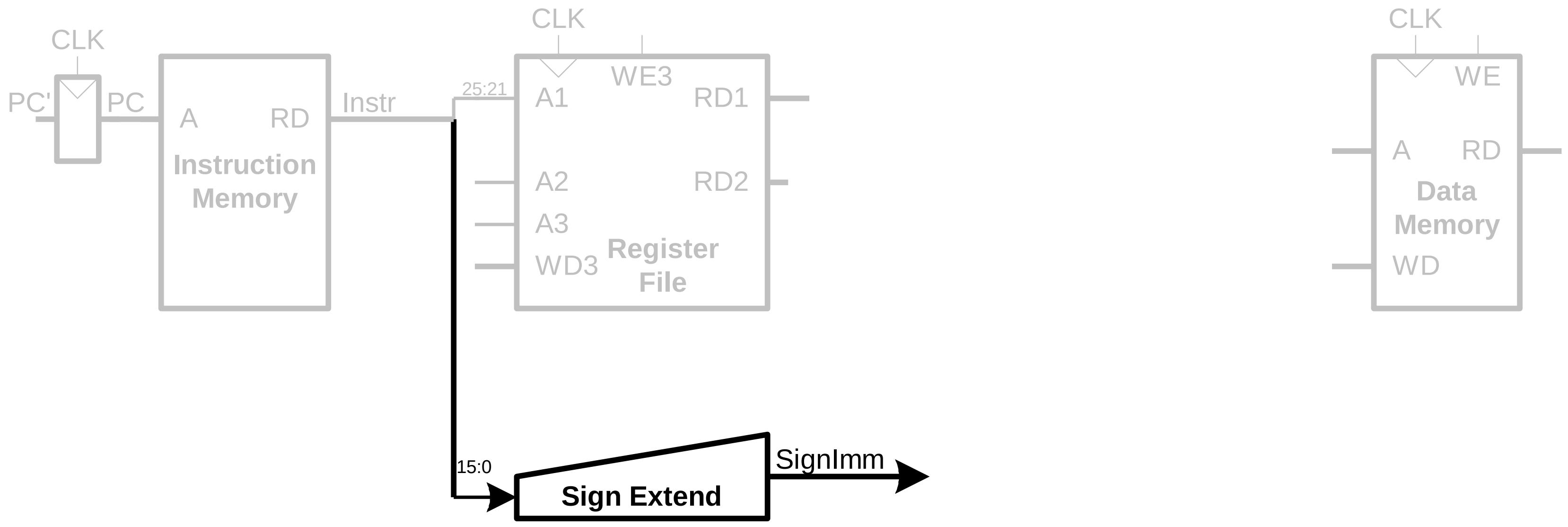
Single-Cycle Datapath: `lw` register read

- **STEP 2:** Read source operands from register file



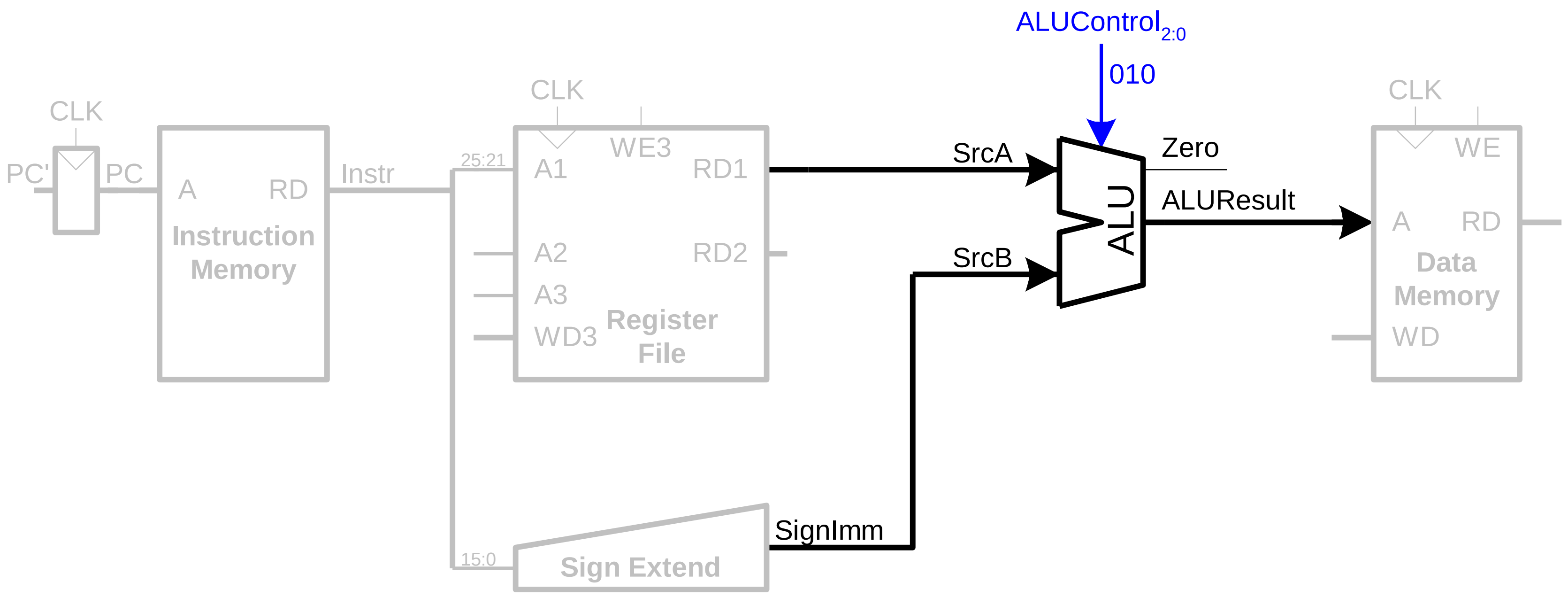
Single-Cycle Datapath: `lw` immediate

- **STEP 3:** Sign-extend the immediate



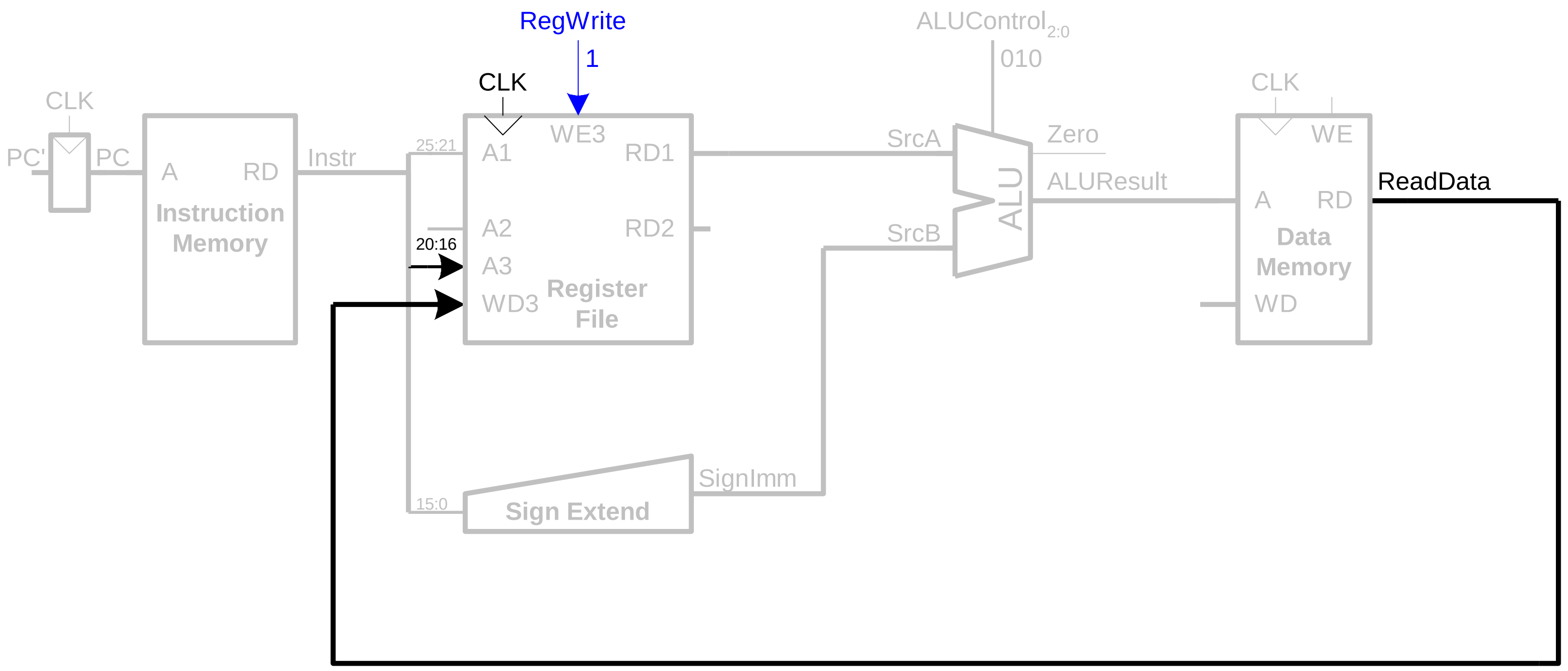
Single-Cycle Datapath: `lw` address

- **STEP 4:** Compute the memory address



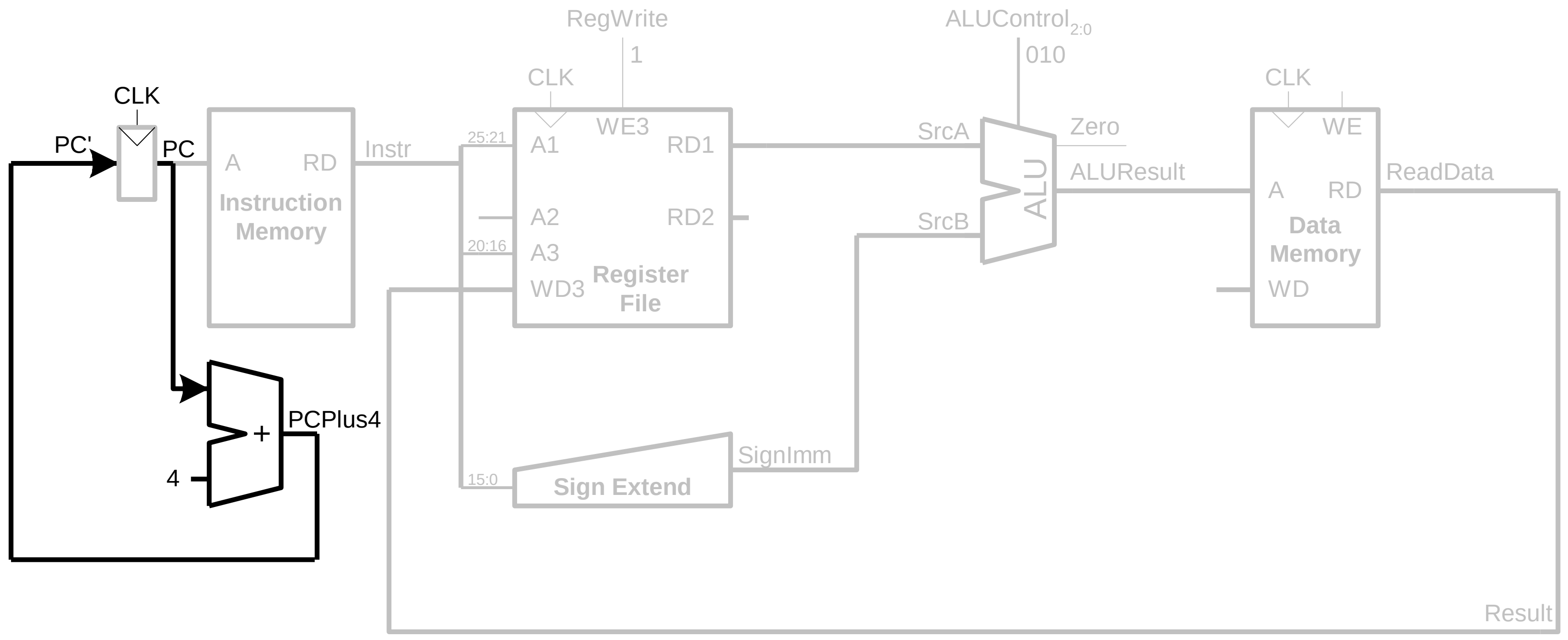
Single-Cycle Datapath: `lw` memory read

- **STEP 5:** Read data from memory and write it back to register file



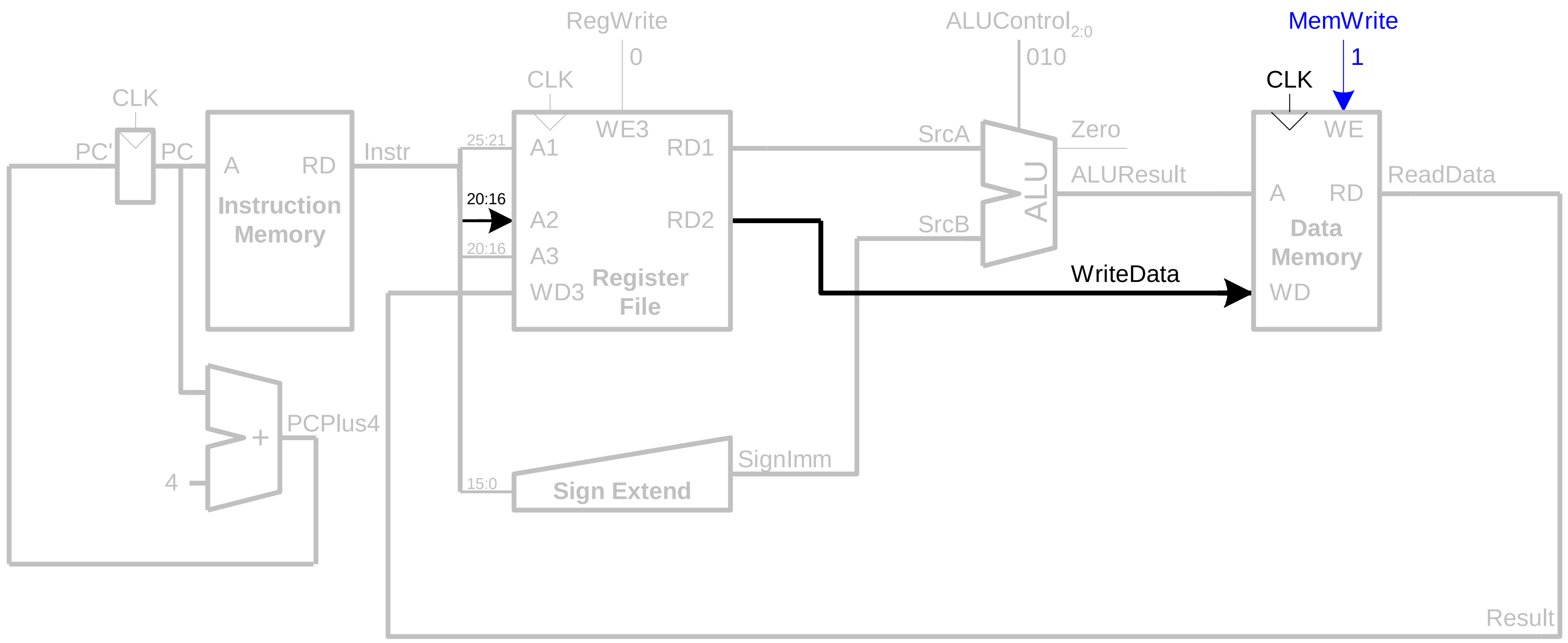
Single-Cycle Datapath: `lw` PC increment

- **STEP 6:** Determine the address of the next instruction



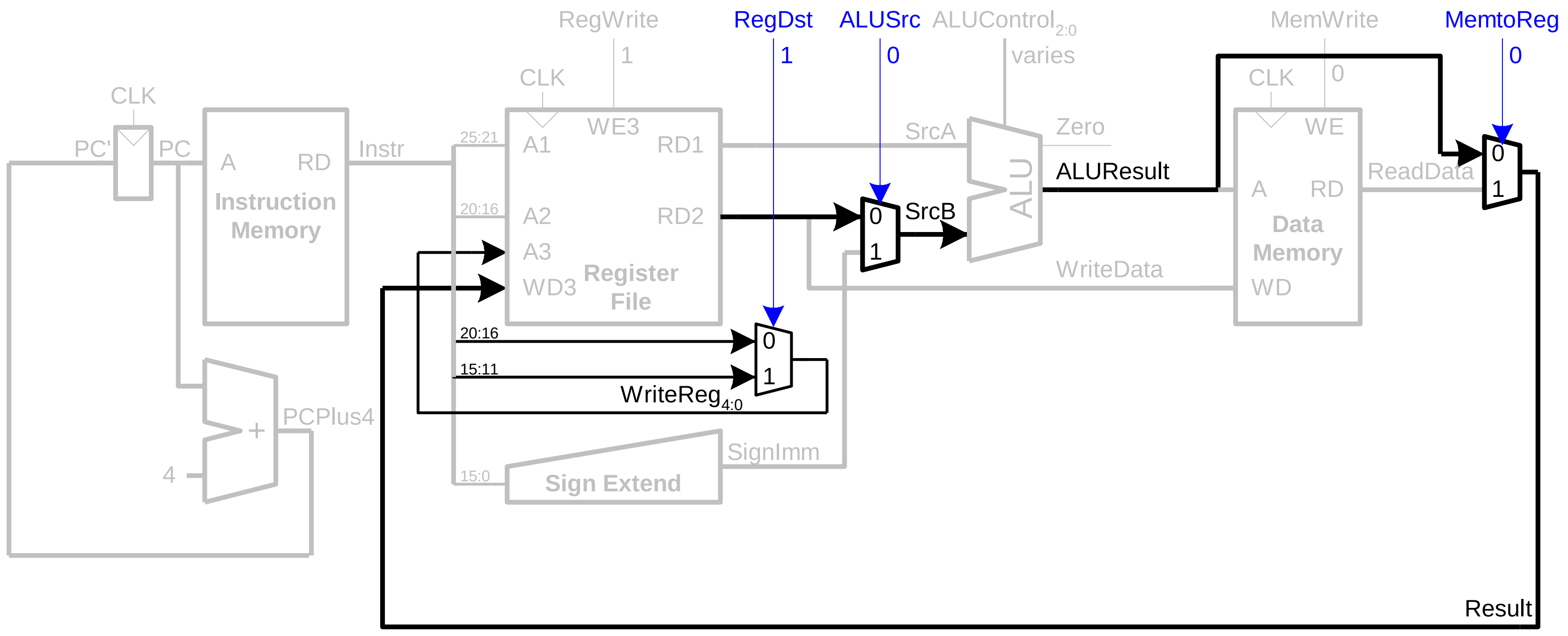
Single-Cycle Datapath: `sw`

- Write data in `rt` to memory



Single-Cycle Datapath: R-type instructions

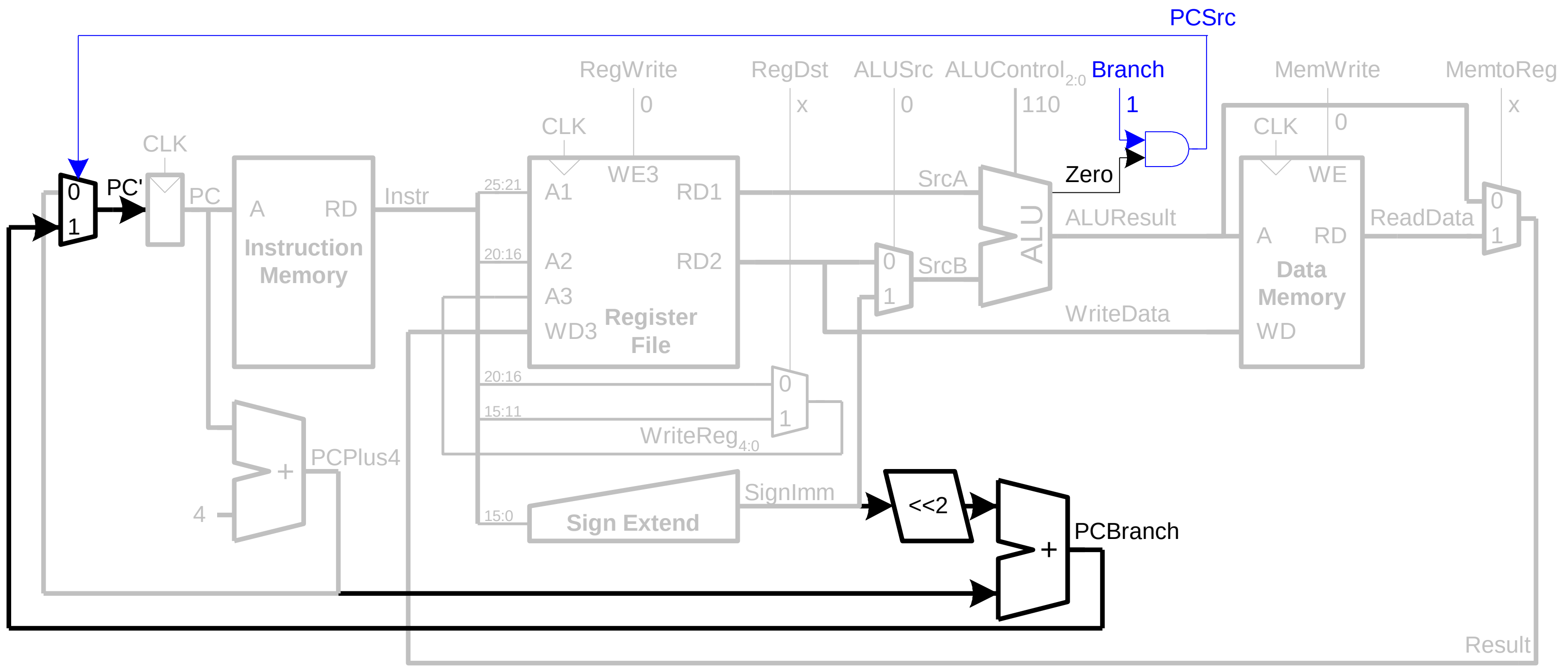
- Read from `rs` and `rt`
- Write *ALUResult* to register file
- Write to `rd` (instead of `rt`)



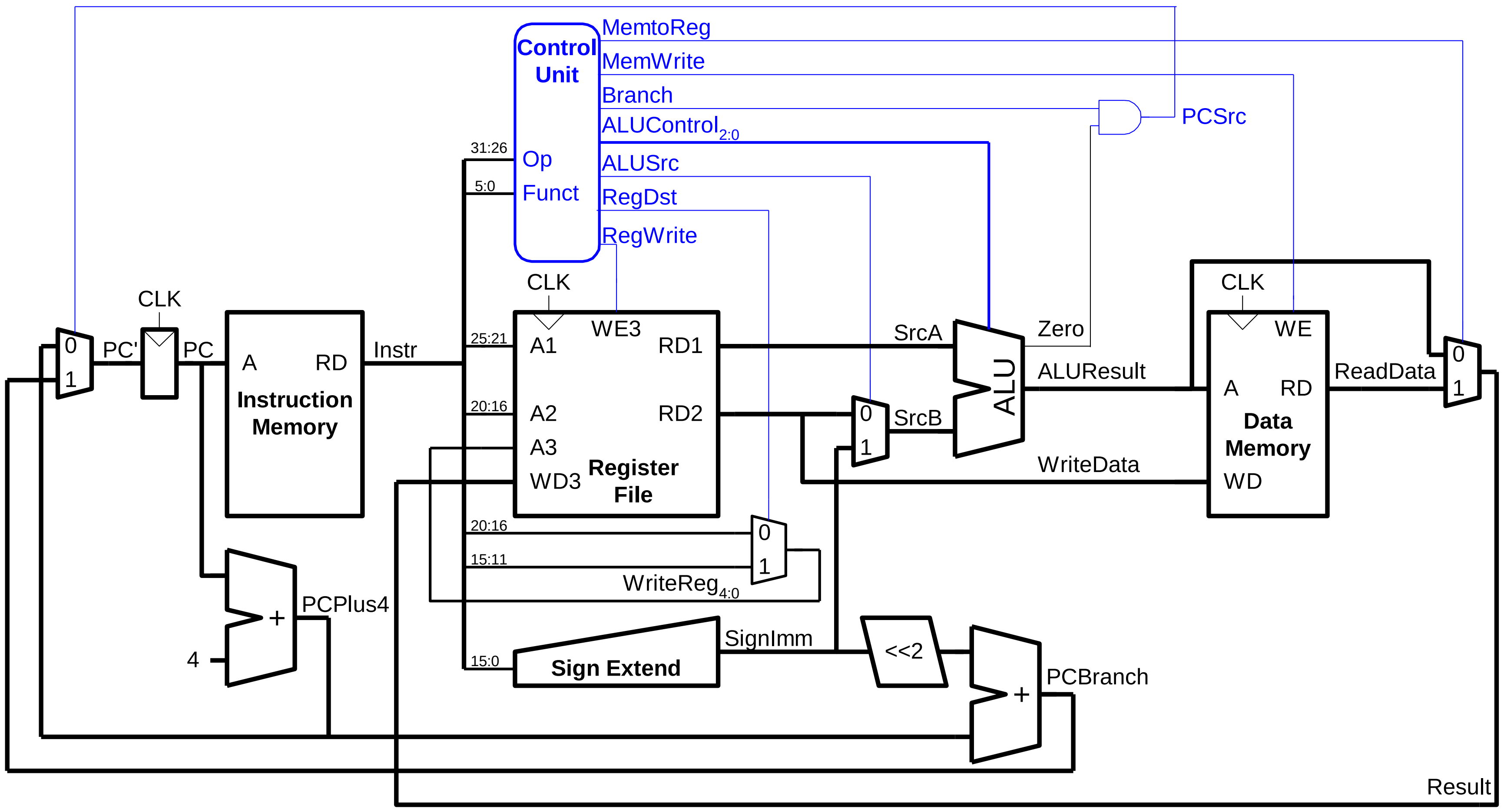
Single-Cycle Datapath: `beq`

- Determine whether values in `rs` and `rt` are equal
- Calculate branch target address:

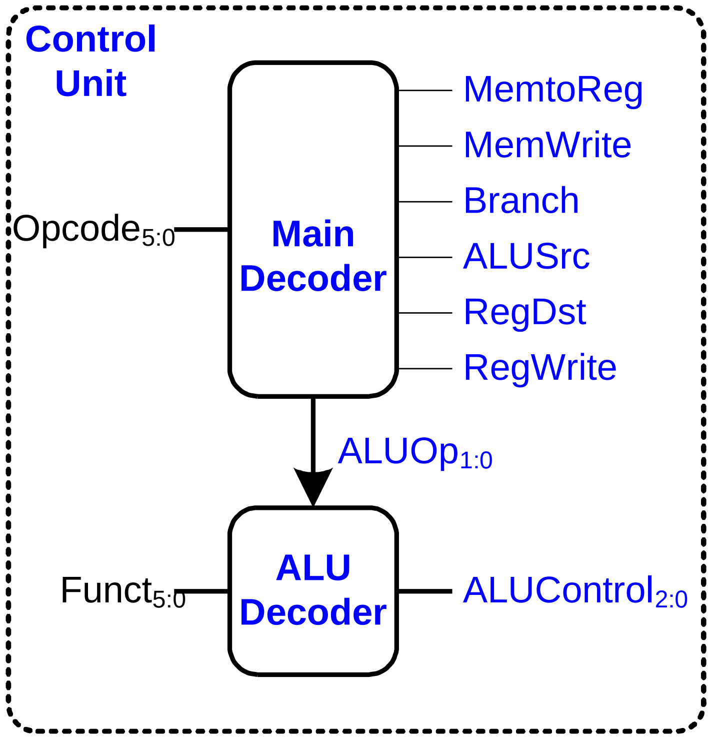
  BTA = (sign-extended immediate << 2) + (PC+4)



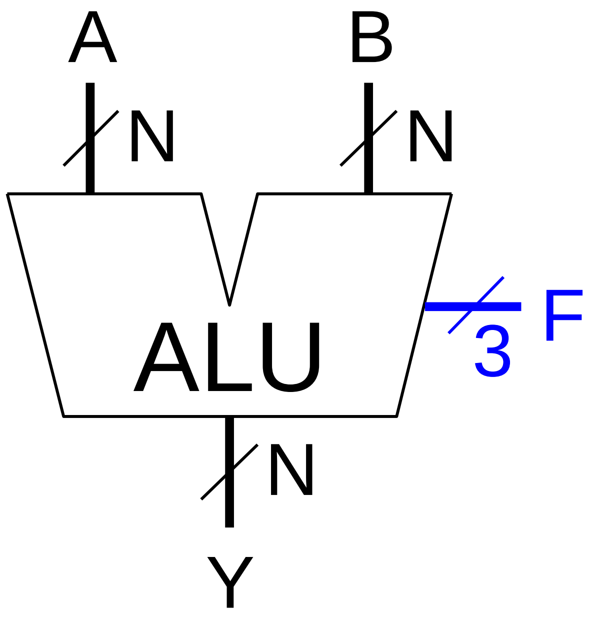
Complete Single-Cycle Processor



Control Unit

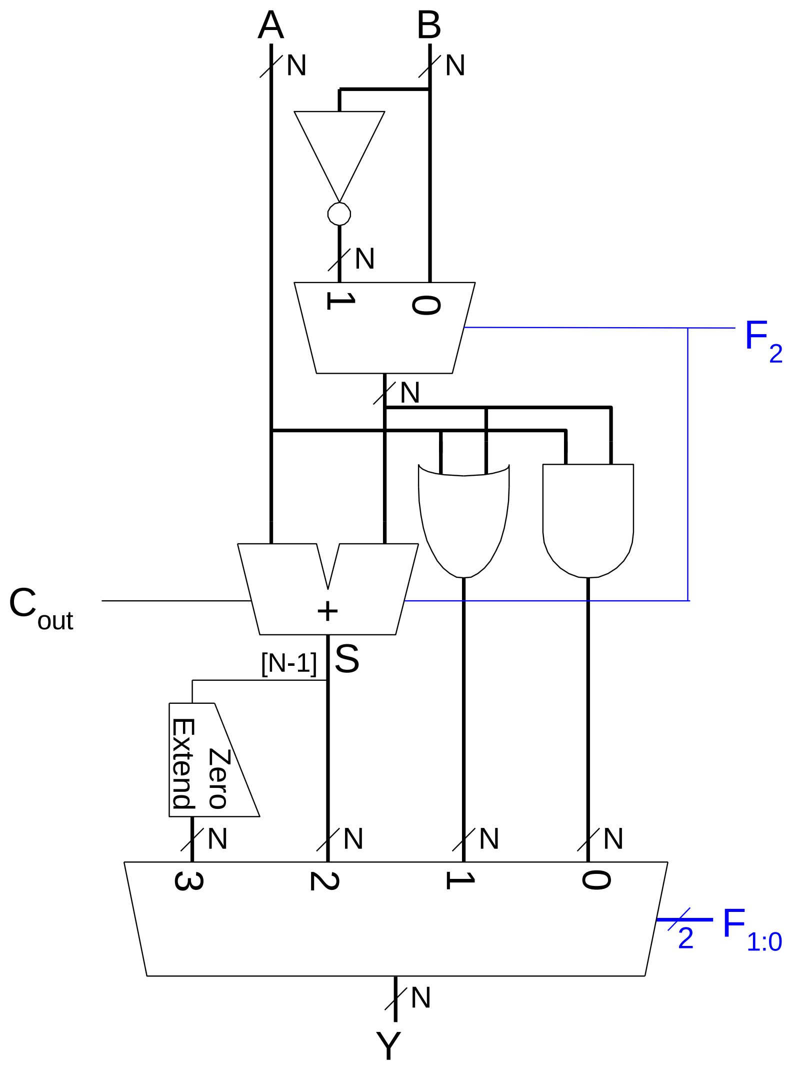


Review: ALU



$F_{2:0}$	Function
000	A & B
001	A | B
010	A + B
011	not used
100	A & ~B
101	A | ~B
110	A - B
111	SLT

Review: ALU



Control Unit: ALU Decoder

$ALUOp_{1:0}$	Meaning
00	Add
01	Subtract
10	Look at Funct
11	Not Used

$ALUOp_{1:0}$	Funct	$ALUControl_{2:0}$
00	X	010 (Add)
X1	X	110 (Subtract)
1X	100000 (`add`)	010 (Add)
1X	100010 (`sub`)	110 (Subtract)
1X	100100 (`and`)	000 (And)
1X	100101 (`or`)	001 (Or)
1X	101010 (`slt`)	111 (SLT)