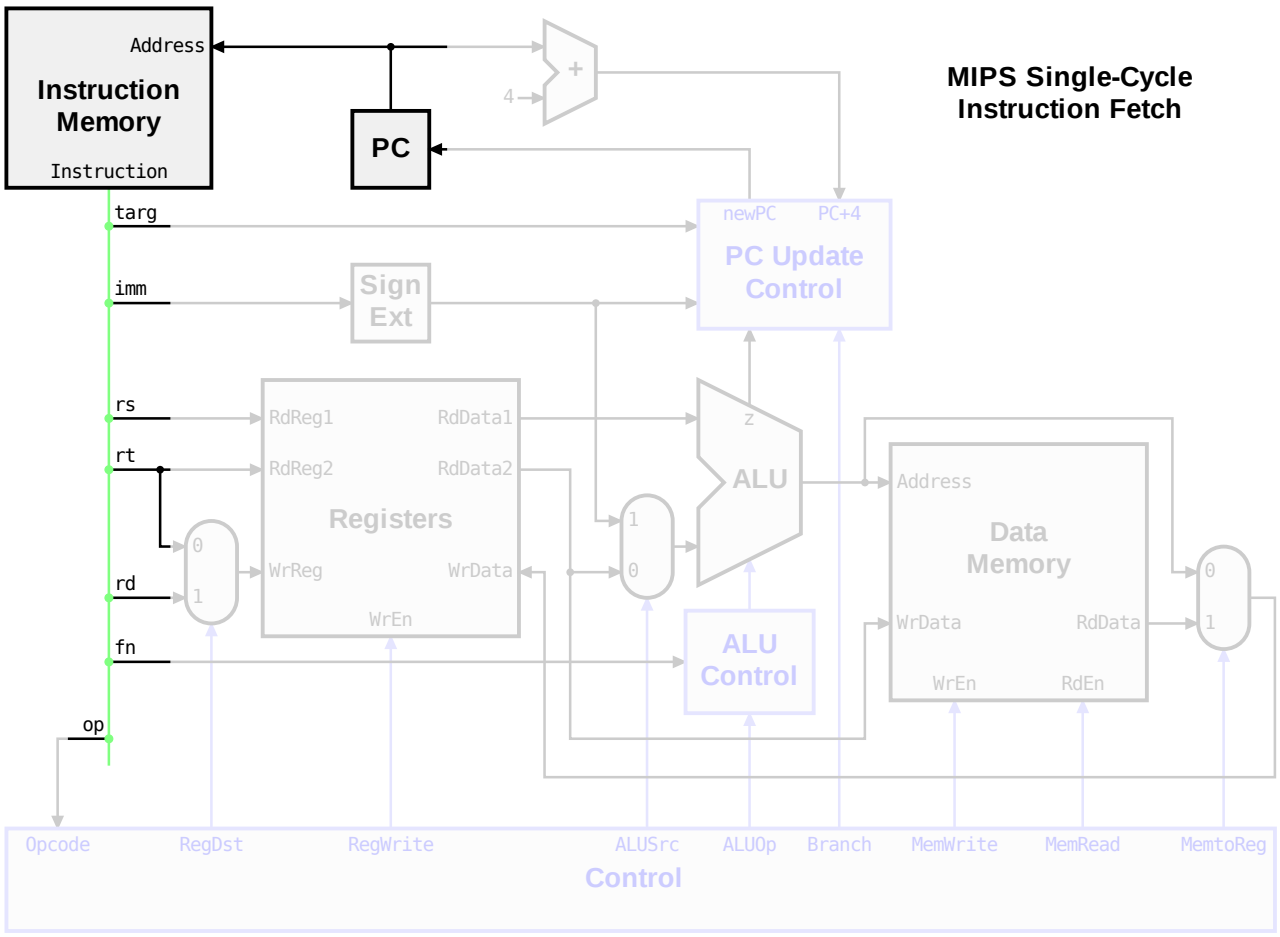
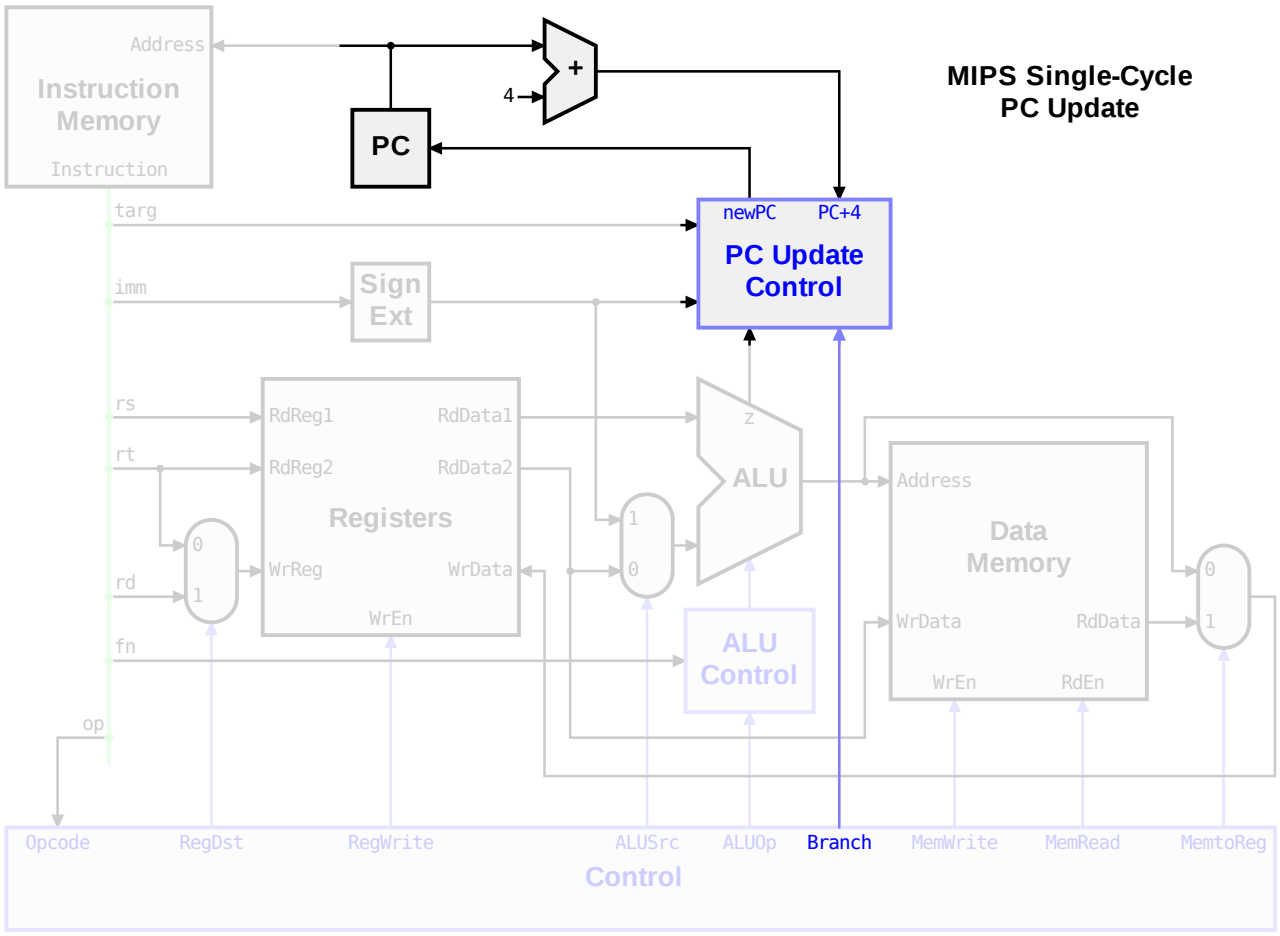


Instruction Fetch

The contents of the program counter (PC) are used as an address sent to instruction memory. Fields from the fetched instruction are sent wherever they are needed. Instruction fetch is the same for all instructions.

Control Signals

Since instruction fetch is the same for all instructions it requires no control signals.



PC Update

The PC gets a new value selected from the following.

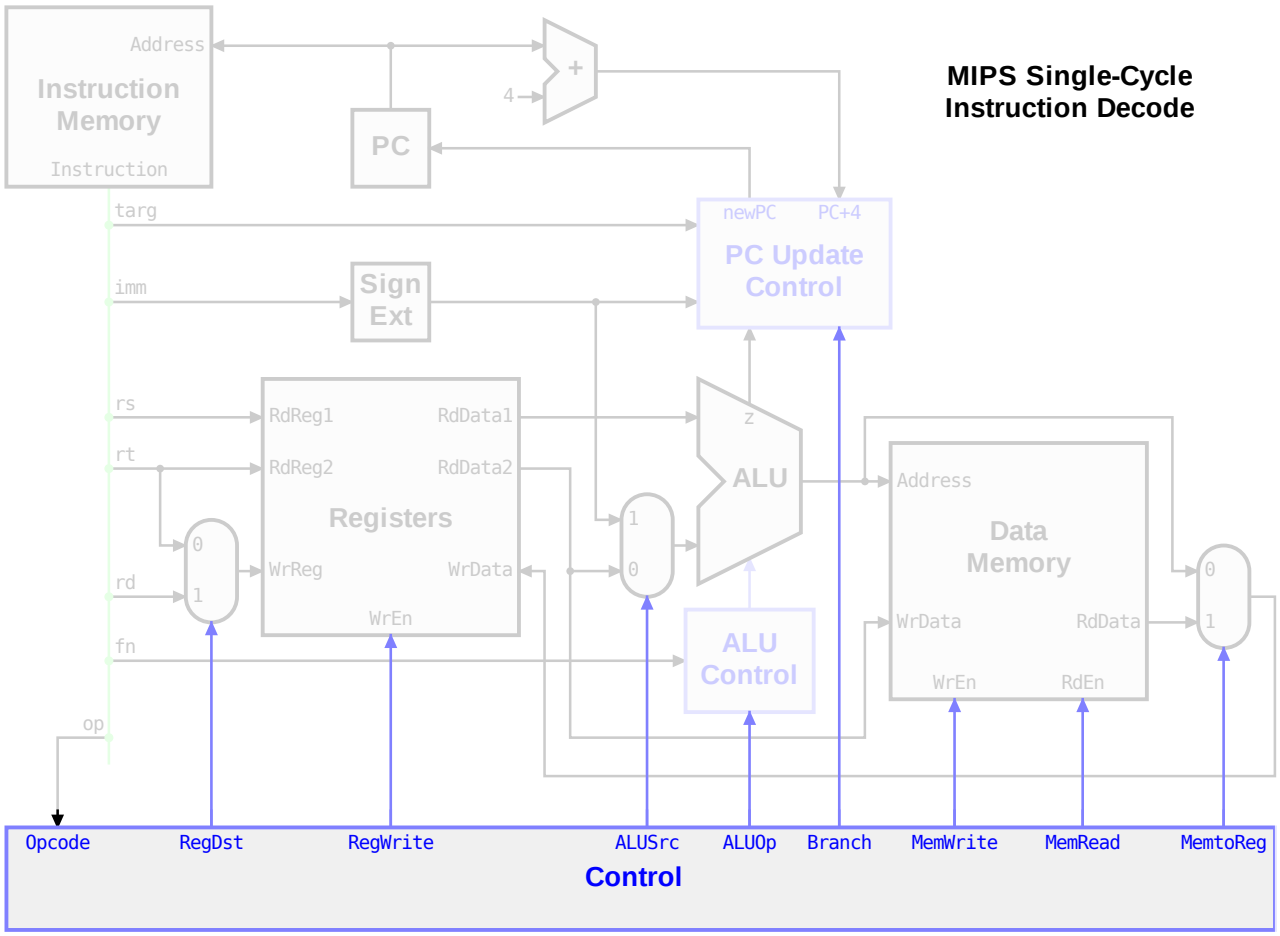
- PC + 4 (most instructions)
- Branch target address (branch instructions)
- Jump target address from the instruction (j and jal instructions)
- Jump target address from a register (jr and jalr instructions)
- Interrupt address (syscall, external interrupts, and exceptions)

Control Signals

- **Branch** — Asserted for branch instructions to select the branch target address as the next instruction address.
- **Jump** — Asserted for jump instructions to select the jump target address as

The Jump control signal is not shown in the MIPS single-cycle implementation diagram. It can be considered to be a part of the Branch control signal.

The MIPS single-cycle implementation diagram and control signals need to be modified to deal with register jump target addresses or interrupt addresses.



Instruction Decode

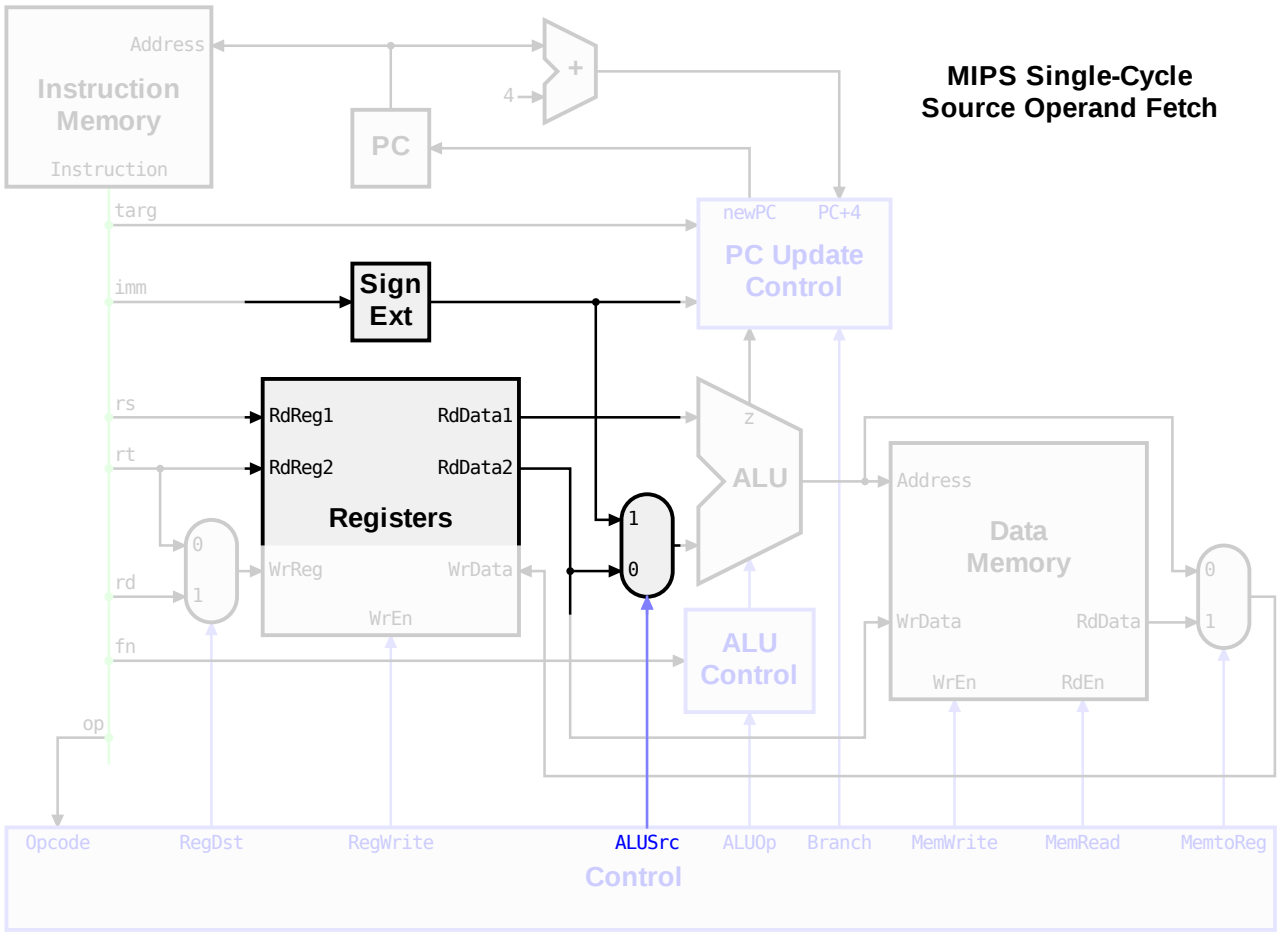
Instruction decoding produces controls signals for the datapath and memory. The inputs to control circuitry are the opcode and function fields of the instruction. It generates the following kinds of control signals.

- read and write control signals for data memory
- write control signals for registers
- multiplexer controls for routing data through the datapath
- control signals to select an appropriate ALU operation

Instruction decode is the same for all instructions.

Control Signals

Since instruction decode is the same for all instructions it requires no control signals.



Source Operand Fetch

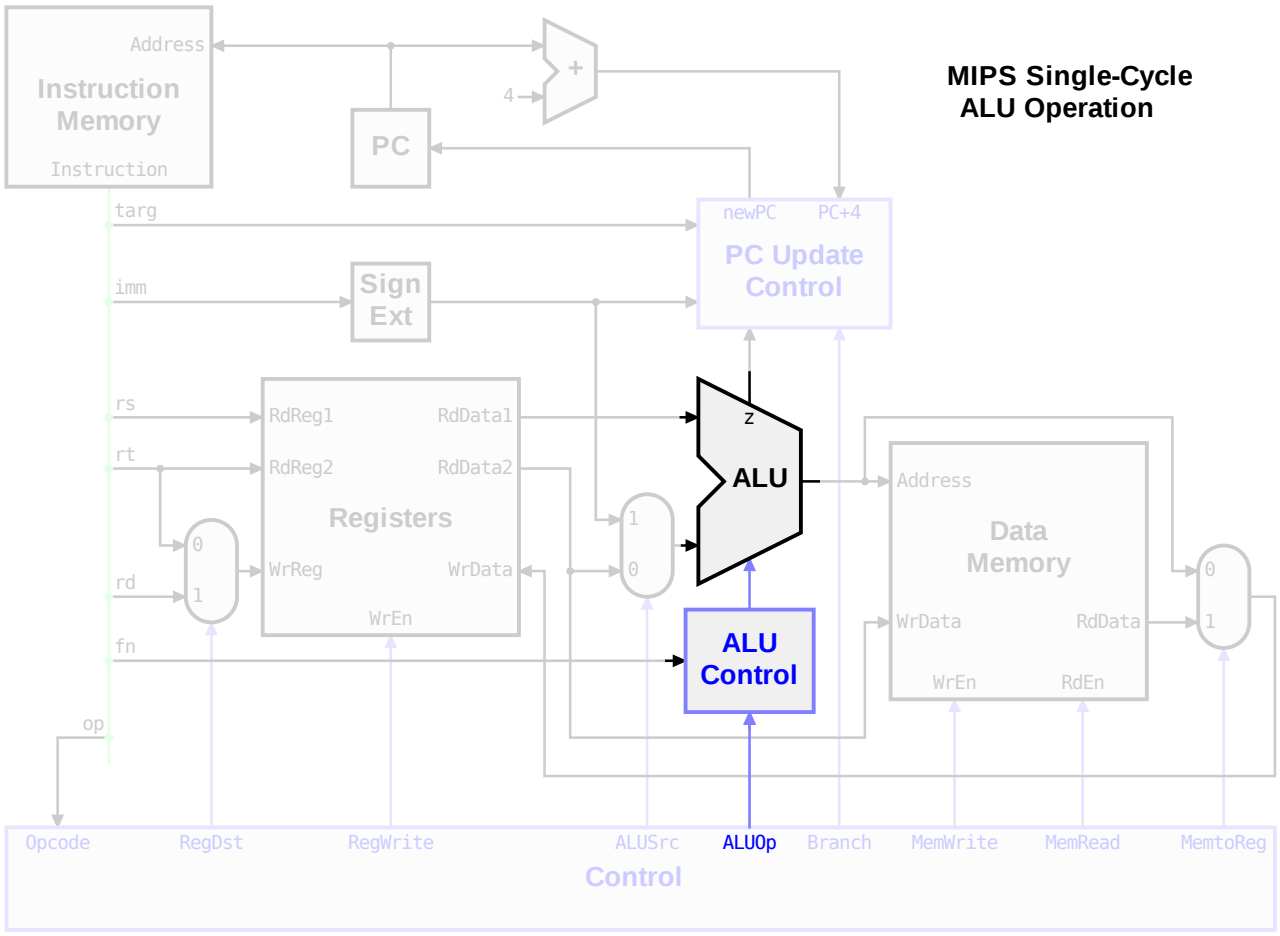
The ALU is designed to combine two source operands to produce a result. The source operand fetch activity fetches the two source operands. One source operand is always the register specified by the rs instruction field. The other is selected from the following.

- The register specified by the rt instruction field
- The sign-extended immediate instruction field
- The zero-extended immediate instruction field

Control Signals

- **ALUSrc** — Selects the second ALU input from either rt or the sign-extended immediate field of the instruction. To deal with the `sltiu` instruction the ALUSrc multiplexer needs an additional input: the zero-extended immediate field. An added value is needed for the ALUSrc control signal to select this input.

The MIPS single-cycle implementation diagram and control signals need to be modified to deal with the zero-extended immediate instruction field. It is only used in a few unsigned integer instructions.



ALU Operation

The ALU can be used in three different ways:

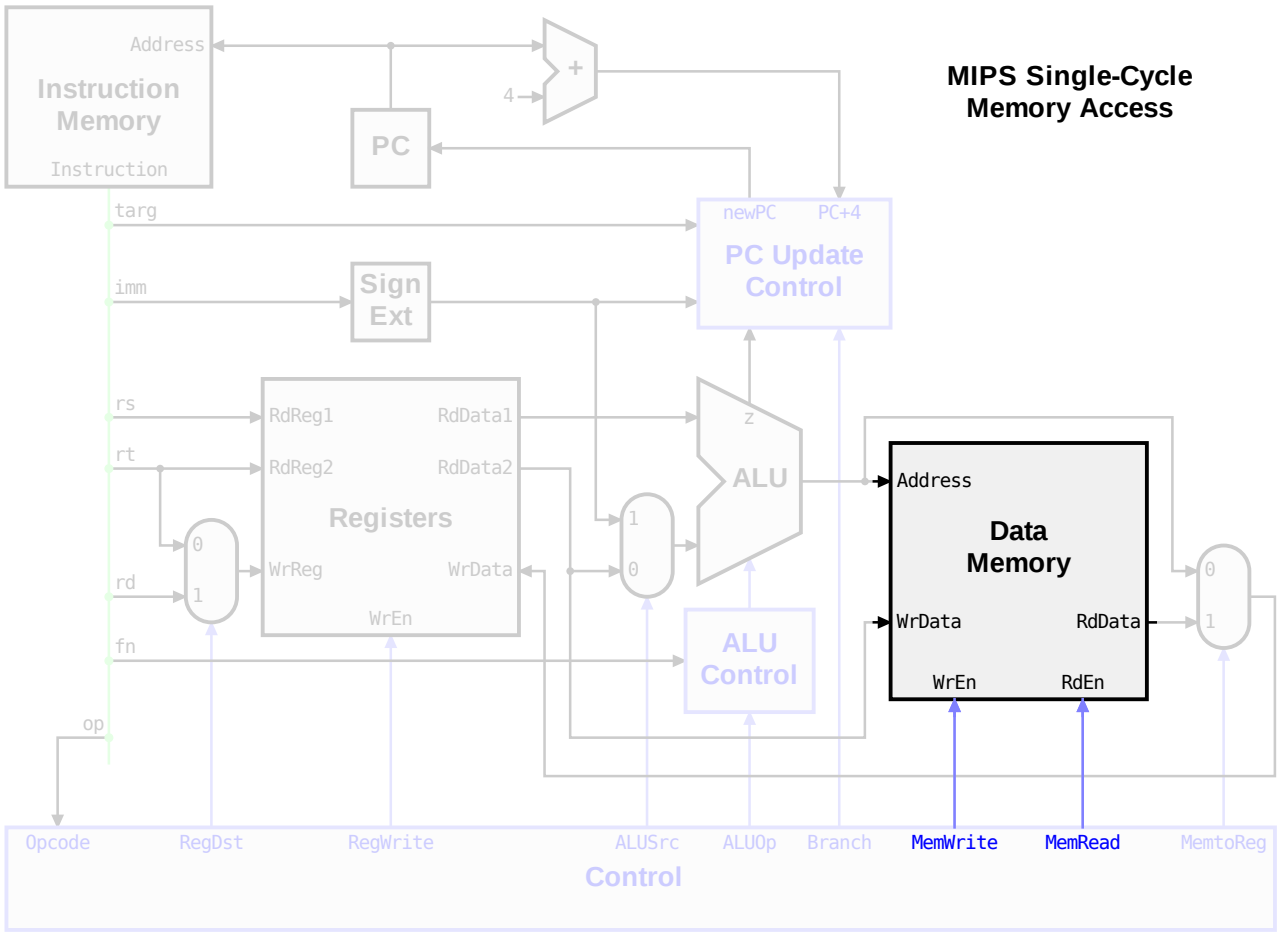
- It performs the arithmetic or logical operation specified by the instruction mnemonic. This is how it is used for all arithmetic and logical instructions.
- It does a subtraction in order to compare two numbers. This is how it is used for branches and compare instructions.
- It calculates a memory address by adding a register and the sign-extended immediate field. This is how it is used for memory loads and stores.

The main control block only decodes the opcode bits of the instruction. When these bits are not all 0 the ALUOp signal from the main control block specifies the ALU operation and this signal is passed to the ALU without modification. When the opcode bits are 000000 it indicates that the instruction is an R-type instruction. Then the function (fn) bits specify the operation performed by the ALU. The ALUOp signal is then just a special code that indicates that the ALU Control block should determine the ALU operation from the function bits.

Control Signals

- **ALUOp** — Determines the operation performed by the ALU. It has three values: "add", "subtract", or "decoded from function field". This signal is sent to the ALU Control circuitry. If ALUOp specifies "add" or "subtract", the ALU Control circuitry sends appropriate control signals for an add or subtract operation to the ALU. If ALUOp specifies "decoded from function field" then the ALU Control circuitry uses the function field to determine the control signal sent to the ALU.

The MIPS single-cycle implementation diagram and control signals need to be modified to deal with immediate instructions such as `ori`.

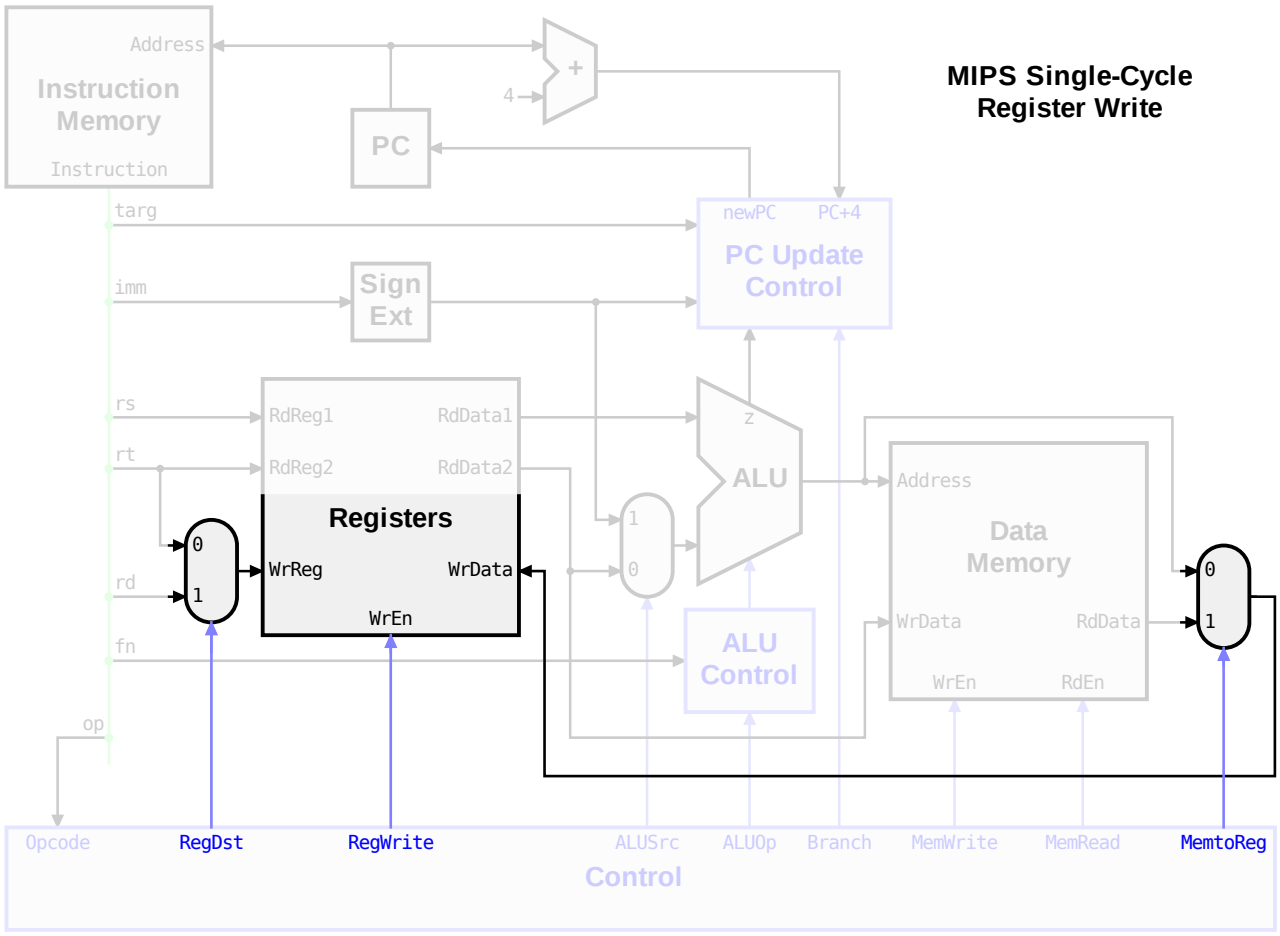


Memory Access

A read or write control signal is sent to memory. The result from the ALU is used as an address. For a store, the value written to memory comes from the register specified by the rt field of the instruction.

Control Signals

- **MemRead** — Asserted for load instructions, tells memory to do a read.
- **MemWrite** — Asserted for store instructions, tells memory to do a write.



Register Write

Some instructions, such as branches, jumps, and stores, do not write to a register. For the instructions that do write to a register, the destination register can be one of the following.

- The register specified by the rd field (R-type instructions)
- The register specified by the rt field (I-type instructions)
- $ra (jal instruction)

The value to be written to the register can come from the following places.

- The ALU (arithmetic and logical instructions)
- Memory (load instructions)
- The incremented PC (jal and jalr instructions)

Control Signals

- **RegWrite** — Asserted if a result needs to be written to a register.
- **RegDst** — Selects the destination register as either rd (R-type instructions) or rt (I-type instructions).
- **MemtoReg** — Selects the source value for the register write as either the ALU result or memory.

The MIPS single-cycle implementation diagram and control signals need to be modified to deal with writing $ra or the incremented PC to a register.