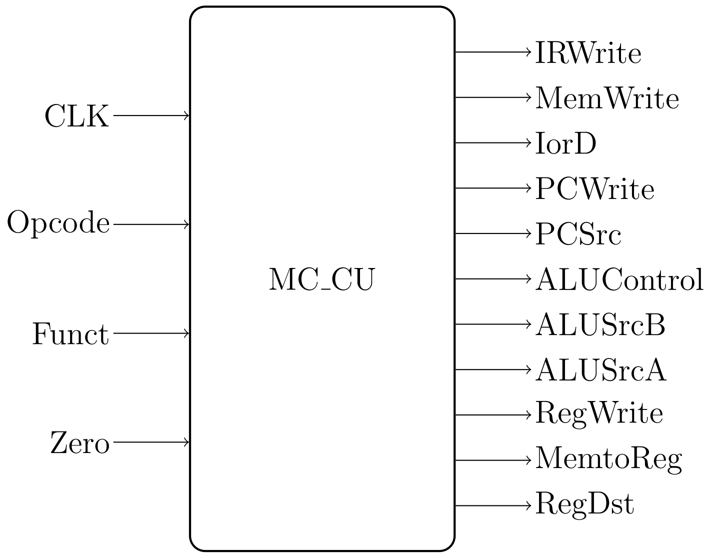
Multicycle Control Unit

Your task is to program the behavior of an entity called "MC_CU". This entity is declared in the attached file "MC_CU.vhdl" and has the following properties:

- Input: CLK with type std_logic
- Input: Opcode with type std_logic_vector of length 6
- Input: Funct with type std_logic_vector of length 6
- Input: Zero with type std_logic
- Output: PCSrc with type std_logic_vector of length 2
- Output: ALUControl with type std_logic_vector of length 3
- Output: ALUSrcB with type std_logic_vector of length 2
- Outputs: IRWrite, MemWrite, IorD, PCWrite, ALUSrcA, RegWrite, MemtoReg, RegDst with type std_logic



Do not change the file "MC_CU.vhdl".

You will have to implement the following types of instructions:

R-type:

31 26	25 21	20 16	15 11	10 6	5 0
opcode	rs	rt	rd	unused	funct

With the opcode representing the instruction, rs and rt the source registers and rd the destination register and funct representing a function for the ALU. The R-type instruction processes the two source registers in the ALU. The ALU result is saved in the destination register specified by rd.

I-type:

31 26	25 21	20 16	15 0
opcode	rs	rt	IMM

With the opcode representing the instruction, rs a source register and IMM the immediate value. The usage of rt depends on the instruction, see below for details.

Before the execution of an instruction the instruction has to be fetched and decoded. This takes two clock cycles. During the fetch clock cycle the instruction is read from the current program counter value (PC) and is saved to the instruction register. Also during the fetch clock cycle the PC register is incremented by 4 using the ALU. The next clock cycle is used to decode the instruction in the control unit. Because the instruction might be a bne instruction which might lead to a branch to a new program counter value, this new program counter value has to be calculated during the decode clock cycle. This means during decode the current PC value has to be added to the potential immediate value in the ALU. The potential immediate value has to be shifted left by 2 Bits beforehand. This makes the branching address available in the register which saves the ALU Result.

The "MC_CU" entity shall control the multicycle processor depicted in Figure 1 to perform the following instructions:

instruction	opcode	funct	type
or	000000	100101	R
bne	000101	-	I

or

This instruction performs an ALU operation with two register values and saves the result to a register. It takes two clock cycles. First the registers specified in rs and rt are loaded into the ALU and the ALU is sent the or control signal. The ALUControl signal for or can be found in Table 1. In the next clock cycle the ALU Result is saved into the register specified by rd.

bne

This instruction compares two registers and if they are not equal, branches to an address specified in the immediate value. It takes only one clock cycle, but as mentioned above, the branching address has to be prepared already while decoding. Then in the clock cycle after decoding a bne instruction, registers specified by rs and rt are compared on equality by substraction in the ALU. If the ALU operation result is zero, then it sets its Zero output to '1'. The control unit has asynchronous logic which sets the PCWrite signal to '1' in case the Zero flag is not '1' during the bne instruction. This writes a new value to the PC register, and thus performs the branching.

ALUControl	Function
010	sub
000	add
101	or

Table 1: ALUControls

To get a better understanding of the control signals, here is a description of each control signal. Use Figure 1 to see how the control signals control the data paths.

- IRWrite: When IRWrite is set to '1' then the instruction read from instruction memory gets written to the instruction register, at a rising edge of the CLK signal.
- MemWrite: When MemWrite is set to '1' then the data memory input write data is written to the data memory input address, at a rising edge of the CLK signal.
- IorD: Selects whether the data memory input address comes from the program counter register or from the ALU Result register.
- PCWrite: When PCWrite is set to '1' then the program counter register is written, at a rising edge of the clock signal.
- PCSrc: Selects whether the new program counter value shall come from the ALU, the ALU Result register or the jump address.
- ALUControl: Selects the ALU operation the ALU performs on its two input values. The controls for the available operations are listed in Table 1.
- ALUSrcB: Selects whether the ALU gets a value from the registers read data 2 output, from a constant value of 4, from the sign extended immediate value of the instruction or from the sign extended immediate value of the instruction shifted left by 2 bits.
- ALUSrcA: Selects whether the ALU gets a value from the registers read data 1 output or from the program counter register.

- RegWrite: When RegWrite is set to '1' then the registers write data input is written to the register specified by the write register input, at a rising edge of the clock signal.
- MemtoReg: Selects whether the write data for the registers comes from data memory or from the ALU Result register.
- RegDst: Selects whether the write register input for the registers comes from bits 20 – 16 of the instruction or from bits 15 – 11 of the instruction.

Consider which actions each part of the processor in Figure 1 has to take to fulfill the functions of the operations and set the control signals accordingly. This behavior has to be programmed in the attached file "MC_CU_beh.vhdl".

To turn in your solution write an email to vhdl-dis+e384@tuwien.ac.at with Subject "Result Task 8" and attach your behavior file "MC_CU_beh.vhdl"

Good Luck and May the Force be with you.

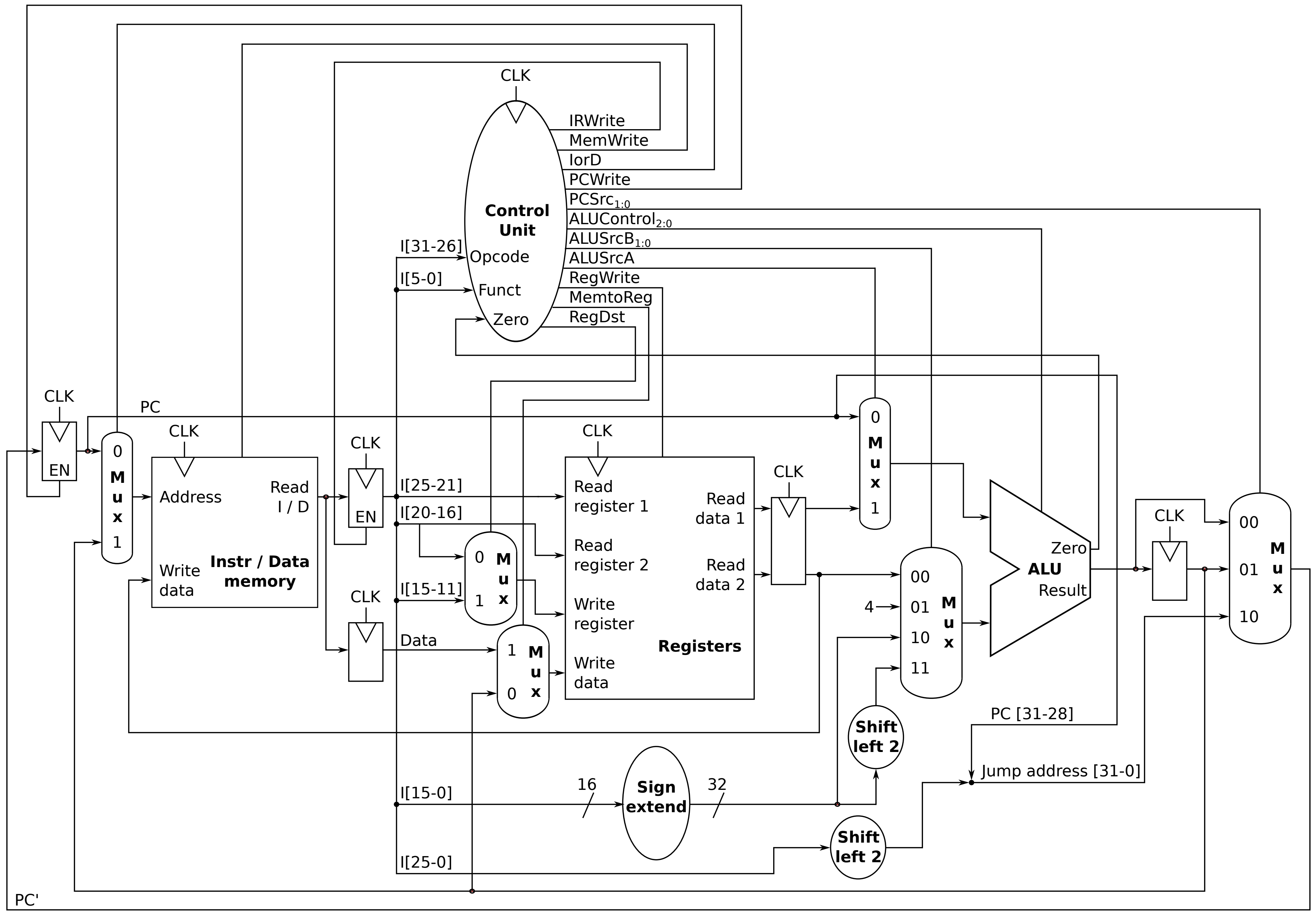


Figure 1: Multicycle processor