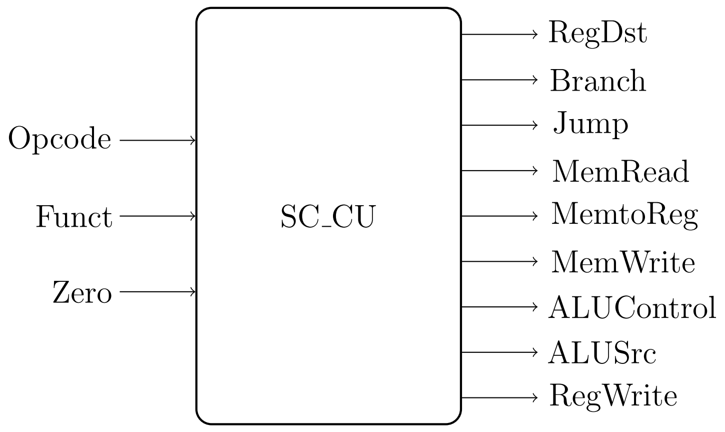
Single Cycle Control Unit

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

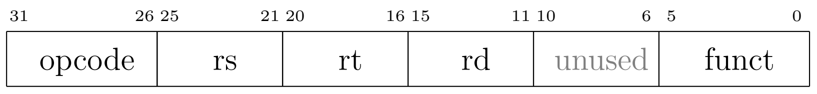
- Input: Opcode with type std_logic_vector of length 6
- Input: Funct with type std_logic_vector of length 6
- Input: Zero with type std_logic
- Output: ALUControl with type std_logic_vector of length 3
- Output: Control signals RegDst, Branch, Jump, MemRead, MemtoReg, MemWrite, ALUSrc and RegWrite all with type std_logic



Do not change the file “SC_CU.vhdl”.

You will have to implement the following types of instructions:

R-type:



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.

The "SC_CU" entity shall control the single cycle processor depicted in Figure 1 to perform the following instructions:

instruction	opcode	funct	zero	type
bne	000101	-	0/1	I
sub	000000	100010	-	R
beq	000100	-	0/1	I
xor	000000	100110	-	R

The sub and XOR instruction uses the ALU to process two register values. The result is stored in the destination register. See Table 1 to find the respective control signal for the ALU. The bne instruction initiates a branch to a new PC value only if the ALU flags its two input register values as unequal. The input registers are specified in rs and rt. The new PC value is calculated by adding the PC value and the immediate value of the instruction. Before adding the immediate value it is extendend from 16 to 32 bits and shifted to the left by two bits. The ALU checks for equality by subtracting its input values and setting the Zero flag on equality. The beq instruction initiates a branch to a new PC value only if the ALU flags its two input register values as equal. The input registers are specified in rs and rt. The new PC value is calculated by adding the PC value and the immediate value of the instruction. Before adding the immediate value it is extendend from 16 to 32 bits and shiftet to the left by two bits. The ALU checks for equality by subtracting its input values and setting the Zero flag on equality.

ALUControl	Function
110	xor
010	sub

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.

- RegDst: Selects whether the register destination comes from the bits 20 – 16 or bits 15 – 11 of the instruction. In other words, this control signal selects if the destination register comes from rt or rd.

- Branch: Has an effect on the source for the next program counter value. If the branch condition is met it will be ‘1’. If the current instruction is not a branch instruction it will always be set to ‘0’.
- Jump: Has an effect on the source for the next program counter value. If the current instruction is not a jump instruction it will always be set to ‘0’.
- MemRead: When the MemRead signal is set to ‘1’, then the data memory outputs the data specified by the read address input.
- MemtoReg: Selects whether the register’s write data input will come from the data memory or the ALU output.
- MemWrite: When the MemWrite signal is set to ‘1’, then the data memory writes its write data to the write 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.
- ALUSrc: Selects whether the ALU gets a value from the register’s read data 2 output, or from the sign extended immediate value of the instruction.
- RegWrite: When the RegWrite signal is set to ‘1’, then the register writes the write data to the write register.

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 “SC_CU_beh.vhdl”.

To turn in your solution write an email to vhdl-dis+e384@tuwien.ac.at with Subject “Result Task 7” and attach your file “SC_CU_beh.vhdl”.

Good Luck and May the Force be with you.

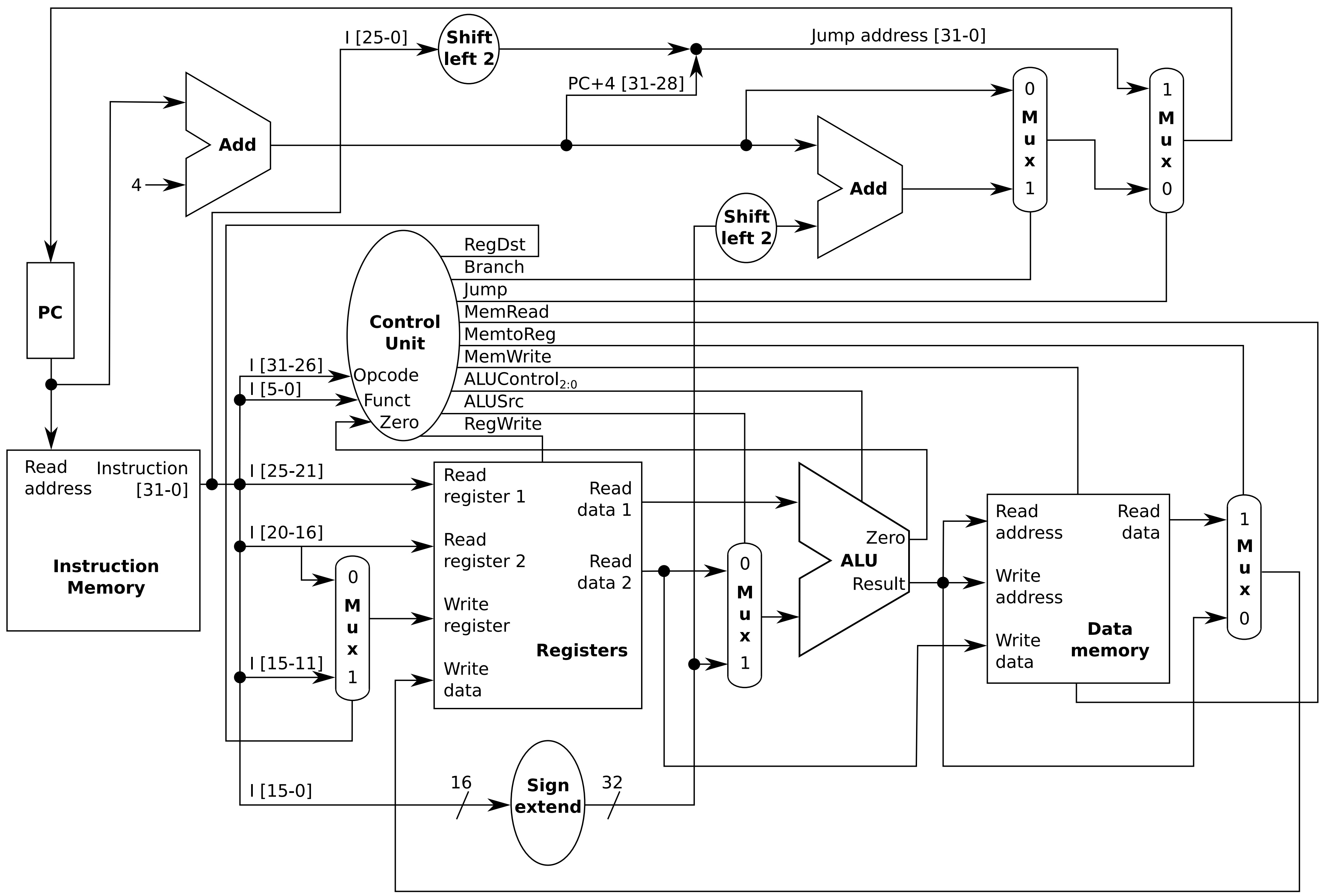


Figure 1: Single cycle processor