REPORT

Branch Not Equal Instruction (bne)

Assembly code was renewed with a new opcode definition

char *op_bne = "00001010"

For this instruction the control unit module was changed accordingly with,

OPCODE = 8'b00001010

ALUOP = 3'b001 (Same opcode as the add/sub instruction)

BRANCH_NOTEQUAL – control signal for further use in updating the program counter.

Inside the ALU unit, by calculating the difference between values in the two input registers RESULT is calculated with a #2 time delay which is already assigned to the add/sub instruction)

Then ZERO output is updated with the new RESULT value. Afterwards inside the CPU unit according to ZERO,BRANCH_NOTEQUAL signals the new_PC value is assigned with,

new_PC = PC+4 + EXTENDED_VAL (how many instructions to branch forward)

Finally PC updates with the coming Positive clockedge and skip to the branched instruction.

Logical Left and Right Shift (sll,srl)

new opcode definitions used

char *op_sll = "00001011"

char *op_srl = "00001100"

Control unit module was added new aluop signals.

OPCODE = 8'b00001011	ALUOP = 3'b101	for left shift
OPCODE = 8'b00001100	ALUOP= 3'b110	for right shift

When the 32 bit instruction is fetched from the instruction register immediate value (which is the shift amount) is given to the ALU input(DATA2) through the IMMEDMUX_SEL.
The register value which need to be shifted left or right is given to the other ALU input(DATA1) through the REGISTER_FILE.

Inside the ALU unit according to the ALUOP the shift type is selected(left or right). Then with a #1 time delay(which is given for the shifting process) RESULT is calculated by shifting according to the given amount. (concatenate method is used)

As the final step in the coming positive clockedge the shifted value is written into the destination register given in the instruction.

- The added new instructions Run within a single clock cycle since the new added delays are not exceeding the previously implemented instruction delays.