LAB # 8
Designing Microarchitecture-III

Objective:

- To design RISC-V Microarchitecture for Store Instruction.

System Module (Hardware/Software)

- Visual Studio Code

Theoretical Background:

Introduction

Store instructions are S-type instructions. The **store word instruction**, sw, copies data from a register to memory. The register is not changed. The memory address is specified using a base/register pair. The name S-type is short for Store-type. S-type instructions use two register operands and one immediate operand. The figure below shows the S-type machine instruction format. The 32-bit instruction has five fields: op, rs1, rs2, funct3, and imm. The first four fields, op, rs1, rs2, and funct3 are like those of R-type instructions. The imm field holds the 12-bit immediate. The operation is determined by the opcode and funct3, highlighted in blue. The operands are specified in the three fields rs1, rs2, and imm. rs and imm are always used as source operands. rs2 is used as another source for store instructions.

The figure below shows an example of SW instruction represented in assembly code and machine code.

Address	Instruction	Type	Fields						Machine Language
			$imm_{11:5}$	rs2	rs1	f3	$imm_{4:0}$	op	
0x1004	sw x6, 8(x9)	S	0000000	00110	01001	010	01000	0100011	0064A423

Like I-type instructions, S-type also has 12-bit immediate. The immediate in store instructions are split into two halves: the lower bits are stored in Instr[11:7] bit and the upper bits are stored in Instr[31:25]. First, we make the 12-bit immediate by concatenating the upper and lower part of immediate and then sign extend it for further use.

Example:

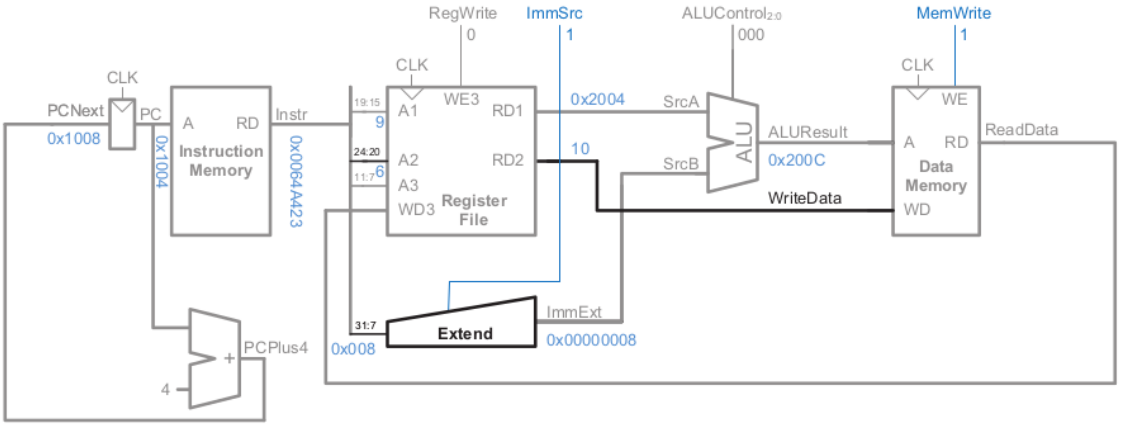
immediate = {Instr[31:25], Instr[11:7]}

Data Path for Store instruction

In this section, we will extend the data path to Store instructions. Like the load instruction, the store instruction reads a base address from port 1 of the register file and sign-extends an immediate. The ALU adds the base address to the immediate to find the memory address. All of these functions are already supported by the datapath. The store instruction also reads a second register from the register file and writes it to the data memory. The figure below shows the new connections for this function. The register is specified in the rs2 field,

Instr[24:20]. These bits of the instruction are connected to the second register file read port, A2.

The register value is read onto the RD2 port. It is connected to the write data port of the data memory. The write enable port of the data memory, WE are controlled by MemWrite. For a store instruction, MemWrite = 1, to write the data to memory; ALUControl = 000, to add the base address and offset; and RegWrite = 0, because nothing should be written to the register file. Note that data is still read from the address given to the data memory, but that this ReadData is ignored because RegWrite = 0.



Address	Instruction	Type	Fields					Machine Language
			$imm_{11:5}$	rs2	rs1	f3	$imm_{4:0}$	op	
0x1004	sw x6, 8(x9)	S	0000000	00110	01001	010	01000	0100011	0064A423

Procedure:

Open Visual Studio Code and perform the procedure mentioned in lab #01 in order to make a Verilog module and test bench.

Lab Tasks:

1. **Write a complete data path for Store Instruction in Verilog by instantiating all the module blocks. Attach the code.**

Conclusion:

What have you learnt from this lab?

Learning Outcomes:

Upon successful completion of the lab, students will be able to:

LO1: Design data path for Store Instruction in Verilog.