2.1 Modules Explanation

- Adder.v

  輸入兩個 32-bit 數字，將兩個數字相加後輸出一個 32-bit 數字。

- Control.v

  輸入是 7-bit 的數字，是 instrctution 的 opcode，而輸出分別有 2-bit 的 ALUOp、 1-bit 的 ALUSrc 以及 1-bit 的RegWrite。透過 opcode 來分辨指令的類型，並且將相對應的 control signal 輸出。如果 opcode 是 0010011，則是 I-type 指令，control signal 輸出 ALUOp 為 00，ALUSrc 為 1，RegWrite 為 1。如果 opcode 是 0110011，則是 R-type 指令，control signal 輸出 ALUOp 為 01，ALUSrc 為 0，RegWrite 為 1。

- ALU_Control.v

  輸入是 7-bit 的數字、3-bit 的數字以及 2-bit 的數字，是 instrctution 的 funct7 和 funct3 以及 Control 輸出的 ALUOp，而輸出是 2-bit 的 ALU_control。先透過 ALUOp 來分辨指令的類型，如果是 R-type 指令，則先透過 funct7 再透過 funct3 來分辨指令的類型，並且將相對應的 ALU_control 輸出。如果是 I-type 指令，則直接透過 funct3 來分辨指令的類型即可，並且將相對應的 ALU_control 輸出。設定 ALU_control 的數字代表的意義如下表所示。

ALI_control	**指令類型**
000	and
001	xor
010	sll
011	add
100	sub
101	mul
110	addi
111	srai

- ALU.v

  輸入是 32-bit 的數字、32-bit 的數字以及 2-bit 的數字，分別是要做運算的第一個 source (rs1)、第二個 source (rs2) 以及 ALU_control (分辨要進行甚麼類型的運算)，而輸出是 32-bit 的結果 ALU_result。先透過 ALU_control 來分辨指令的類型，將 rs1 跟 rs2 作相對應的運算後，再將 ALU_result 輸出。

- Sign_Extend.v

  輸入是 12-bit 的數字，是 instruction [31:20] 的部分，若指令是 I-type，則代表一個常數。而輸出是 32-bit 的數字，是 sign-extended 後的數字，其 [11:0] 的位子與輸入相同，而 [31:12] 的位子則是複製 20 次輸入的最高位元 [31]。

- MUX32.v

輸入是 32-bit 的數字、32-bit 的數字以及 1-bit 的數字，分別是要選擇的第一個 source (input1)、第二個 source (input2) 以及一個 control signal (sel)，而輸出是 32-bit 的數字，是選擇後的 source。如果 control signal (sel) 是 0，則將 input1 輸出，如果 control signal (sel) 是 1，則將 input2 輸出。

- CPU.v

  將所有 module 依照作業給的 datapath 串接起來。首先將 PC 的輸出分別傳給 Adder 和 Instruction_Memory，前者將 PC 的輸出加上常數 4，來指向下一個指令的位址；後者透過 PC 的輸出，找到對應位址的指令，並且將指令輸出。將 Instruction_Memory 的輸出分別拆解並傳給 Control、Registers、Sign_Extend 以及 ALU_Control。Control 依照輸入的 opcode 輸出對應的 control signal 給 Registers、MUX_ALUSrc 以及 ALU_Control。Registers 依照輸入的 register 編號，輸出 rs1 和 rs2 的值，並確認寫入的 register 編號。Sign_Extend 則是將輸入的 12-bit 數字 sign extend 成 32-bit 並輸出。而 Registers 的 rs2 和 Sign_Extend 輸出的值會經過 MUX_ALUSrc，並透過 Control 輸出的 control signal (ALU_src) 來決定要取哪個值輸出。再來 ALU 則會將 Registers 的 rs1 和 MUX_ALUSrc 的輸出進行運算，透過 ALU_Control 的輸出 (ALU_control) 來決定進行甚什運算。最後將 ALU 的輸出根據 Control 輸出的 control signal (RegWrite) 寫回 Registers。

2.2 Development Environment

- OS: WSL2 (Windows Subsystem for Linux version 2), Linux distro: Ubuntu 22.04.5 LTS
- compiler: iverilog version 11.0
- IDE: Visual Studio Code