B08901027 吳育丞
B08901048 陳宥辰
B08901019 吳冠緯

我們的 CPU 架構主要有 Memory, RegFile, PC, ALU, ImmGen, FSM, mulDiv，前兩者助教已經完成，然後 mulDiv 也在作業二完成，所以我們主要設計的部分在於 PC, ALU, ImmGen, FSM，然後這些部分我們都是寫在 Chip 的 module 裡面。以上的單元都是利用從 Memory 讀進來的 instructions 進行控制，我們按照講義將 op code 命名為 Control，可以區分 branch, auipc, jal, jalr, load, store, I-type, R-type 的 instructions，然後把 funct code 當作 ALU_control 的 input，可以再區分出 R-type 裡面的 add, sub, mul 以及 I-type 裡面的 addi, slti, slli, srli。至於 instructions 裡面的其他部分我們會分別送到 RegFile 和 ImmGen 中得出正確的 rs1/rs2 data 和 immediate。最後我們只要根據 Control 和 ALU_cotrol 的數值讓 ALU 進行適當的運算，同時讓 PC 得出下個 instruction 的 address，最後再視情況將 ALU, PC 或 Memory 讀出的數值存入 RegFile 的 rd data 或是 Memory 即可。

auipc: 讀出 instruction，將 rd_address 送進 RegFile，imm 的部分送入 ImmGen 得出正確的 immediate，然後再利用 Control(op code & funct code)控制 ALU 得出 PC+immediate 的數值，最後存入 RegFile 的 rd data。

jal: 讀出 instruction，將 rd_address 送進 RegFile，imm 的部分送入 ImmGen 得出正確的 immediate，然後再利用 Control(op code & funct code)控制 ALU 得出 PC+immediate 的數值，最後存入 PC_nxt。此外，還需要把 PC+4 的數值存入 RegFile 的 rd data。

jalr: 讀出 instruction，將 rs1/rd_address 送進 RegFile 得到 rs1_data，imm 的部分送入 ImmGen 得出正確的 immediate，然後再利用 Control(op code & funct code)控制 ALU 得出 rs1_data+immediate 的數值，最後存入 PC_nxt。此外，還需要把 PC+4 的數值存入 RegFile 的 rd data。

slli: 讀出 instruction，將 rs1/rd_address 送進 RegFile 得到 rs1_data，imm 的部分送入 ImmGen 得出正確的 immediate，然後再利用 Control(op code & funct code)控制 ALU 得出 rs1_data<<immediate 的數值，最後存入 RegFile 的 rd_data。

srli: 讀出 instruction，將 rs1/rd_address 送進 RegFile 得到 rs1_data，imm 的部分送入 ImmGen 得出正確的 immediate，然後再利用 Control(op code & funct code)控制 ALU 得出 rs1_data>>immediate 的數值，最後存入 RegFile 的 rd_data。

針對 multi-cycles instruction(mul)，我們設計了 FSM 來協助控制。在其他 single-cycle operation 以及 mul 的第一個 cycle，state 定義為 Single，而當 Control 和 ALU_control 的值顯示出目前的 instruction 為 mul，state 會改變成 Multi，此時 valid 會變成 1，將 rs1/rs2_data 送進 mulDiv 進行運算，而 valid 在第二個 cycle 開始就會變回 0。在 mulDiv 運算的過程中 PC 會維持不變，state 會維持在 Multi，直到 mulDiv 的 ready 變成 1 代表計算完成，此時會把 rd_address 送入 RegFile，regWrite 變為 1，然後把 mulDiv 的結果存入 RegFile 的 rd_data，最後把 state 改回 Single 再繼續執行後面的 instructions。

Simulation Time(Default):

Leaf: 255 NS

Fact: 3875 NS

HW1: 645 NS

由結果可見 Fact 因為使用 multi-cycles operation(mul)所以明顯需要更多時間。

Register table:

Register Name	Type	Width	Bus	MB	AR	AS	SR	SS	ST
PC_reg	Flip-flop	31	Y	N	Y	N	N	N	N
PC_reg	Flip-flop	1	N	N	N	Y	N	N	N
state_reg	Flip-flop	1	N	N	Y	N	N	N	N

Register Name	Type	Width	Bus	MB	AR	AS	SR	SS	ST
mem_reg	Flip-flop	995	Y	N	Y	N	N	N	N
mem_reg	Flip-flop	29	Y	N	N	Y	N	N	N

Register Name	Type	Width	Bus	MB	AR	AS	SR	SS	ST
alu_in_reg	Flip-flop	32	Y	N	Y	N	N	N	N
shreg_reg	Flip-flop	64	Y	N	Y	N	N	N	N
state_reg	Flip-flop	3	Y	N	Y	N	N	N	N
counter_reg	Flip-flop	5	Y	N	Y	N	N	N	N

Distribution:

B08901027(吳育丞): Control, ImmGen, PC, ALU, FSM, Linking wires and regs, Bonus, report

B08901048(陳宥辰): Control, ImmGen, PC, Bonus, nWave-debugging, changing test patterns, report

B08901019(吳冠緯): ALU, Linking wires and regs, changing test patterns, report