CA Final Project Report Group 22

b08901056 盧弘偉 b08901198 顏柏傑

1. CPU Architecture

先進行instruction fetch及decode，再根據不同opcode、func3、func7來決定控制訊號、輸出還有rd_data，並依需求決定ALU的模式。

2. Data Path of Instructions

- jal, jalr
  兩者實作方法相似，先進行instruction decode，若opcode符合則設rd_data為PC+4，PC_nxt則依各自規則賦值 (jal : PC-relative, jalr : Base addressing)。
- auipc
  先進行instruction decode，若opcode符合則設rd_data為PC+upper immediate，運作完成則執行下一個指令(PC_nxt = PC+4)。

3. Multi-cycle Instructions

Multi-cycle instruction的實作是採用類似作業二的FSM概念：讀到mul指令時若ALU沒有在運作(alu_running = 0)，就將alu_running_nxt與mulDiv_valid設為1；若ALU有在運作(alu_running = 1)，則會讓mulDiv_valid變回0，再依據mulDiv_ready分case，若mulDiv_ready = 0，代表答案還沒算完，此時要讓ALU繼續運作(設alu_running_nxt = 1)，PC_nxt = PC；若mulDiv_ready = 1，可以把對應的output給值，PC_nxt = PC + 4，並讓 alu_running_nxt = 0。

4. Simulation Time

- Leaf: a = 3, b = 9, c = 5, d = 17

```
Simulation complete via $finish(1) at time 275 NS + 0
```

- Perm: n = 8, r = 5

```
Simulation complete via $finish(1) at time 2695 NS + 0
```

- (Bonus) HW1: n = 11

```
Simulation complete via $finish(1) at time 2195 NS + 0
```

5. Observation

SIZE_TEXT、SIZE_DATA、SIZE_STACK影響了bonus運作的正確性，執行前兩行指令後就發生instruction fetch出錯的問題，直到上述SIZE都調到41以上才解決。

6. Register Tables

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
alu_running_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

7. Work Distribution

Main architecture : 盧弘偉

Bonus : 盧弘偉，顏柏傑

修正HW1 instructions : 顏柏傑