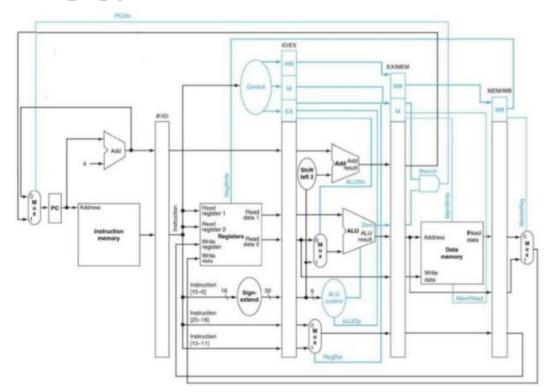
# **Computer Organization Lab4**

# **Architecture diagrams:**

CPU: 取自 CO\_Lab\_4.pdf



Pipeline Register: 數字為 bit 數

| IF/ID:   | ID/EX:          | EX/MEM:      | MEM/WB:      |  |  |
|----------|-----------------|--------------|--------------|--|--|
| 32 pc+4  | WB              | WB           | WB           |  |  |
| 32 instr | 1 RegWrite      | 1 RegWrite   | 1 RegWrite   |  |  |
|          | 1 MemtoReg      | 1 MemtoReg   | 1 MemtoReg   |  |  |
|          | M               | M            |              |  |  |
|          | 1 Branch        | 1 Branch     | 32 MemData   |  |  |
|          | 1 MemRead       | 1 MemRead    | 32 ALUResult |  |  |
|          | 1 MemWrite      | 1 MemWrite   | 5. WriteReg  |  |  |
|          | EX              |              |              |  |  |
|          | 3 ALU_op        | 32 pc_branch |              |  |  |
|          | 1 RegDst        | 1 . zero     |              |  |  |
|          | 1 ALUSrc        | 32 ALUResult |              |  |  |
|          |                 | 32 RTData    |              |  |  |
|          | 32 Pc+4         | 5. WriteReg  |              |  |  |
|          | 32 RSData       |              |              |  |  |
|          | 32 RTData       |              |              |  |  |
|          | 32 Extended     |              |              |  |  |
|          | 5. instr[20-16] |              |              |  |  |
|          | 5. instr[15-11] |              |              |  |  |

#### Hardware module analysis:

(explain how the design work and its pros and cons)

和 single-cycle CPU 相比,因為多加了 pipeline register,每個 stage 可以被分配給不同的指令,所以同時可以有多個指令在進行,不會浪費空閒的 hardware。 每個 pipeline register 中,儲存該指令在下一個 stage 所需要的 control signal 和 data

IF: 讀取當前 PC Addr 的指令,因為指令有可能是 branch 所以需要儲存 pc+4

ID: 利用 IF/ID 儲存的指令,得到 RSData、RTData 和 control signal 等

EX: 利用 ID/EX 中的 control signal 和 Data 做 ALU 的計算, 一併算出 Branch addr、得到 RD Addr (WriteReg)

MEM: 利用 EX/MEM 儲存的 RSData (Read Addr)和讀取出的 RTData,再根據 control signal 決定是否要讀/修改 memory。

且在此時確定 branch 的結果,並將計算的結果連結至 Program Counter WB: 根據 MEM/WB 儲存的 control signal 決定是否需要 Write to Reg (Memory data / ALU result)

每個 stage 都各自做不同的指令,有效利用資源,但有可能不同指令之間有 data dependency , 就會造成 hazard , 則需要更複雜的機制(forwarding 、 stall 等)去控制,但整體而言效率是比 single-cycle CPU 好很多的。

#### Finished part:

(show the screenshot of the simulation result and waveform, and explain it)

| Test1: | 和助教給的參考答案是相同的 |
|--------|---------------|
|--------|---------------|

| begin:                       |  |                                       |   |   |   |  |   |   |  |  |
|------------------------------|--|---------------------------------------|---|---|---|--|---|---|--|--|
| addi                         | \$1,\$0,3;                             |                                       |   | // a = 3                                |   |  |   |   |  |  |
| addi                         | \$2,\$0,4;                             |                                       |   | // b = 4                                |   |  |   |   |  |  |
| addi                         | \$                                     | 3,\$0,1;                              |   | // c = 1                                |   |  |   |   |  |  |
| sw                           | \$1,4(\$0);                            |                                       |   | // A[1] = 3                             |   |  |   |   |  |  |
| add                          | \$4,\$1,\$1;                           |                                       |   | // \$4 = 2*a                            |   |  |   |   |  |  |
| or                           | \$6,\$1,\$2;                           |                                       |   | $// e = a \mid b$                       |   |  |   |   |  |  |
| and                          | nd \$7,\$1,\$3;                        |                                       |   | $// f = a \delta$                       | // f = a & c                            |  |   |   |  |  |
| sub                          | b \$5,\$4,\$2;                         |                                       |   | // d = 2*                               | 'a - b                                  |  |   |   |  |  |
| slt                          | \$8,\$1,\$2;                           |                                       |   | // g = a                                | // g = a < b                            |  |   |   |  |  |
| beq                          | eq \$1,\$2,begin                       |                                       |   |   |   |  |   |   |  |  |
| lw                           | lw \$10,4(\$0);                        |                                       |   | // i = A[1]                             |   |  |   |   |  |  |
| Register                     |  |                                       |   |   |   |  |   |   |  |  |
| Register==                   |  |                                       |   |   |   |  |   |   |  |  |
| Register==:                  |  | 3, r2=                                | 4, r3=                                  | 1, r4=                                  | 6, r5=                                  | 2, r6=                                 | 7, r7= 1                                  |   |  |  |
|                              |  |                                       |   | 1, r4=<br>0, r12=                       |   | 2, r6=<br>0, r14=                      |   | 0 |  |  |
| r0=                          | 0, r1=                                 | 3, r2=                                |   |   |   |  |   | 0 |  |  |
| r0=<br>r8=                   | 0, r1=<br>1, r9=                       | 3, r2=<br>0, r10=                     | 3, r11=                                 | 0, r12=                                 | 0, r13=                                 | 0, r14=                                | 0, r15=                                   |   |  |  |
| r0=<br>r8=<br>r16=<br>r24=   | 0, r1=<br>1, r9=<br>0, r17=            | 3, r2= 0, r10= 0, r18= 0, r26=        | 3, r11=<br>0, r19=                      | 0, r12=<br>0, r20=                      | 0, r13=<br>0, r21=                      | 0, r14=<br>0, r22=                     | 0, r15=<br>0, r23=                        | 0 |  |  |
| r0=<br>r8=<br>r16=<br>r24=   | 0, r1=<br>1, r9=<br>0, r17=<br>0, r25= | 3, r2= 0, r10= 0, r18= 0, r26=        | 3, r11=<br>0, r19=                      | 0, r12=<br>0, r20=                      | 0, r13=<br>0, r21=                      | 0, r14=<br>0, r22=<br>0, r30=          | 0, r15=<br>0, r23=                        | 0 |  |  |
| r0= r8= r16= r24= Memory==== | 0, r1= 1, r9= 0, r17= 0, r25=          | 3, r2= 0, r10= 0, r18= 0, r26=        | 3, r11=<br>0, r19=<br>0, r27=<br>0, m3= | 0, r12=<br>0, r20=<br>0, r28=<br>0, m4= | 0, r13=<br>0, r21=<br>0, r29=<br>0, m5= | 0, r14=<br>0, r22=<br>0, r30=          | 0, r15=<br>0, r23=<br>0, r31=<br>0, m7= 0 | 0 |  |  |
| r0= r8= r16= r24= Memory==== | 0, r1= 1, r9= 0, r17= 0, r25= 0, m1=   | 3, r2= 0, r10= 0, r18= 0, r26= 3, m2= | 3, r11=<br>0, r19=<br>0, r27=<br>0, m3= | 0, r12= 0, r20= 0, r28= 0, m4=          | 0, r13=<br>0, r21=<br>0, r29=<br>0, m5= | 0, r14= 0, r22= 0, r30= 0, m6= 0, m14= | 0, r15=<br>0, r23=<br>0, r31=<br>0, m7= 0 | 0 |  |  |

#### Problems you met and solutions:

除了在 Pipeline Reg 的設計上想了比較久的時間,基本上都是沿用 single-cycle CPU 的設計,沒有做太多更動,因此沒有遇到太多困難。

# **Bonus (optional):**

Modified machine code:

| 0010000      | 00000000 | 00100000000 | 0000010000 |         | l1:  | addi    | \$1, \$0, 16  |         |   |   |
|--------------|----------|-------------|------------|---------|------|---------|---------------|---------|---|---|
| 0010000      | 00000010 | 0010000000  | 0001100100 |         | I10: | addi    | \$9, \$0, 100 |         |   |   |
| 0010000      | 00000000 | 110000000   | 000001000  |         | 13:  | addi    | \$3, \$0, 8   |         |   |   |
| 0010000      | 00001000 | 100000000   | 000000100  |         | 12:  | addi    | \$2, \$1, 4   |         |   |   |
| 1010110      | 00000000 | 0010000000  | 000000100  |         | 14:  | SW      | \$1,4(\$0)    |         |   |   |
| 1000110      | 00000001 | .000000000  | 000000100  |         | 15:  | lw      | \$4, 4(\$0)   |         |   |   |
| 0010000      | 00001001 | 110000000   | 000001010  |         | 18:  | addi    | \$7, \$1, 10  |         |   |   |
| 0000000      | 00011000 | 0010011000  | 0000100000 |         | 17:  | add     | \$6, \$3, \$1 |         |   |   |
| 0000000      | 00100000 | )110010100  | 0000100010 |         | 16:  | sub     | \$5, \$4, \$3 |         |   |   |
| 0000000      | 00111000 | )110100000  | 0000100100 |         | 19:  | and     | \$8, \$7, \$3 |         |   |   |
| Register==== |          |             |            | =====   |      |         |               |         |   |   |
| r0=          | 0, r1=   | 16, r2=     | 20, r3= {  | 8, r4=  | 16,  | r5=     | 8, r6= 24,    | r7= 26  |   |   |
| r8=          | 8, r9=   | 100, r10=   | 0, r11=    | 0, r12= |      | 0, r13= | 0, r14=       | 0, r15= | 0 |   |
| r16=         | 0, r17=  | 0, r18=     | 0, r19=    | 0, r20= |      | 0, r21= | 0, r22=       | 0, r23= | 0 |   |
| r24=         | 0, r25=  | 0, r26=     | 0, r27=    | 0, r28= |      | 0, r29= | 0, r30=       | 0, r31= | 0 |   |
|              |          |             |            |         |      |         |               |         |   |   |
| Memory====   |          |             |            |         |      |         |               |         |   |   |
| mO=          | O, m1=   | 16, m2=     | O, m3=     | 0, m4=  | 0    | ), m5=  | 0, m6=        | 0, m7=  | 0 |   |
| m8=          | O, m9=   | O, m10=     | O, m11=    | O, m12= |      | O, m13  | = 0, m14=     | 0, m15= | 0 |   |
| r16=         | 0, m17=  | O, m18=     | O, m19=    | 0, m2   | 0=   | O, m    | 21= 0, m22=   | 0, m23= |   | 0 |
| m24=         | 0, m25=  | 0, m26=     | 0, m27=    | O, m2   | 8=   | O, m    | 29= 0, m30=   | 0, m31= |   | 0 |
|              |          |             |            |         |      |         |               |         |   |   |

### **Summary:**

這次的 lab 讓我對 pipeline CPU 有更多的理解,bonus 的部分也讓我發現自己思考的盲點。