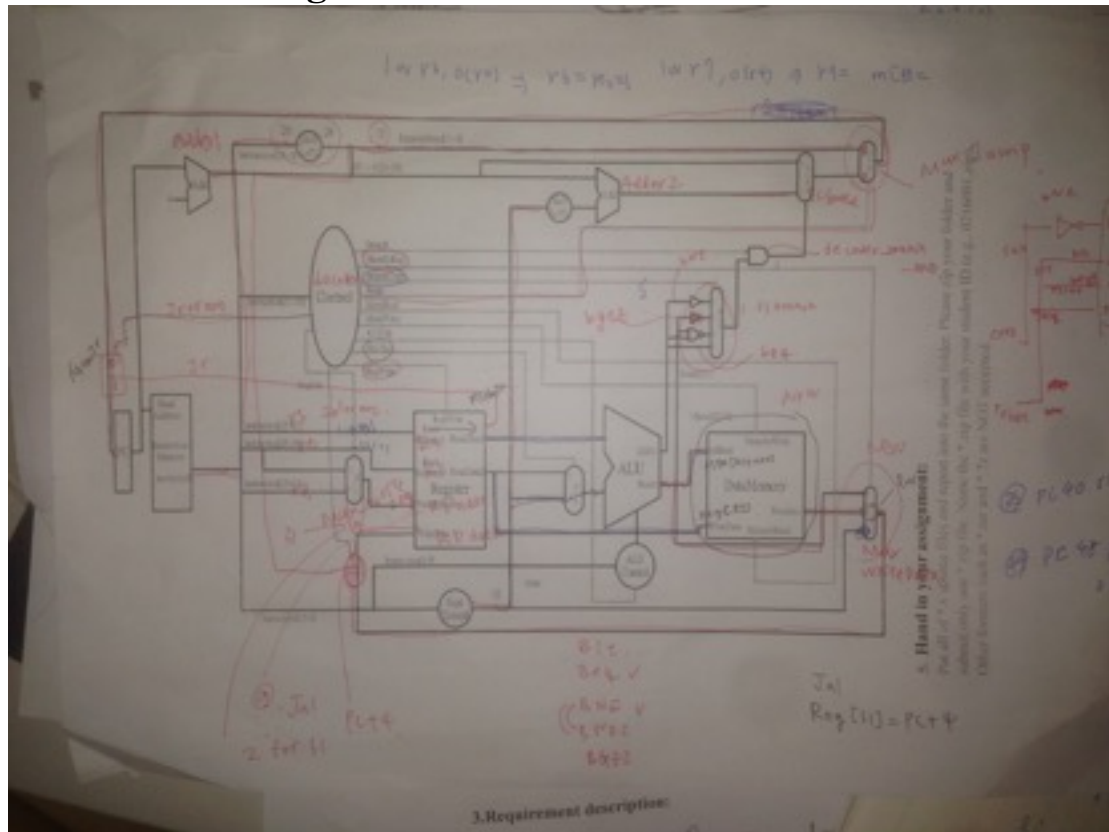


Computer Organization

Architecture diagram:



Detailed description of the implementation:

這次作業好難，好難debug。

幾乎是依照老師給的photo做最右邊datamemory出來的mux我只用了2 to 1 mux 最下面之前從sign extension 出來的wire我沒用。

另外多加了兩個mux 一個給jr 一個給jal 後來發現mux jr可以拿掉，但已經遲交，就趕快做好

test1 test2都有過 因為test3是自己打binary code就沒有測試有沒有過。

Problems encountered and solutions:

最大的麻煩就是我一直以為我lw sw 寫錯 但test1 都跑得出來，怎麼可能test2會錯。後來我一行一行看

0 add r0,r0,r0	000000 00000 00000 00000 00000 100000
4 addi a0,zero,4	001000 00000 00100 00000 000000000100 r4 = 4
8 addi t1,zero,1	001000 00000 01001 00000 000000000001 r9 = 1
12 jal fib	000011 000000000000000000000000000101 r31 = fib:
10100 20	

```

16 j      final      000010 00000000000000000000000011110 11110<<2 1111000
120
20 fib:   addi sp,sp,-12      001000 11101 11101 1111111111110100 r29=r29 -
12 = 116
24 sw ra,0(sp)      101011 11101 11111(31) 0000000000000000
m[r29=116/4=29+0] = r31
28 sw s0,4(sp)      101011 11101 10000(16) 0000000000000100
m[r29+1] = r16
32 sw s1,8(sp)      101011 11101 10001(17) 0000000000001000
m[r29+2] = r17
36 add s0,a0,zero    000000 00100 00000 10000 00000 100000 r16 = r4 +
r0
40 beq s0,zero,rel   000100 10000 00000 00000000000001100 r16 ?= r0
off=12 44+12*4 = 92
44 beq s0,t1,rel     000100 10000 01001 00000000000001011 r16 ?= r9
off=11 48+11*4 = 92
48 addi a0,s0,-1     001000 10000 00100 1111111111111111 r4 = r16 -1
52 jal fib           000011 0000000000000000000000000101 r31 =
add(56), 101<<2= 10100 =20
56 add s1,zero,v0    000000 00000 00010 10001 00000 100000 r17 = r0+r2
60 addi a0,s0,-2     001000 10000 00100 1111111111111110 r4 = r16+ (-2)
64      jal fib       000011 0000000000000000000000000101 r31 =
exitfib(68), 101<<2 = 10100 =20
68      add v0,v0,s1   000000 00010 10001 00010 00000 100000 r2 = r2+
r17
72 exitfib:lw ra,0(sp) 100011 11101 11111 000000000000000000 r31 =
mem[r29]
76      lw s0,4(sp)    100011 11101 10000 000000000000000100 r16 =
mem[r29+1]
80      lw s1,8(sp)    100011 11101 10001 00000000000001000 r17 =
mem[r29+2]
84      addi sp,sp,12   001000 11101 11101 00000000000001100 r29 = r29+12
88      jr ra          000000 11111 00000 00000 000000 01000 jr $31
      //function call結束
92 rel:   addi v0,zero,1    001000 00000 00010 00000000000000001 r2 =
r0+1;
96      j exitfib       000010 00000000000000000000000010010 10010<<2
1001000 =72
100 final:   nop

```

才發現是 jr 的opcode 也是000000 沒有辦法直接從decoder那邊去判斷 pc要吃mux Jump出來的wire (最右上角的mux)還是吃register File出來的 RS data 最後我終於看懂原來 sw ra,0(sp) 101011 11101 11111(31) 000000000000000000 m[r29=116/4=29+0] = r31 是這樣運作的。

所以Lw Sw我都没做錯，但我花了一天在debug sw lw

test1

R14 =	0, R15 =	0, R16 =	0, R17 =	0, R18 =	0, R19 =	0, R20 =	0, R21 =	0
PC =	1604							
m0 =	1, m1 =	2, m2 =	0, m3 =	0, m4 =	0, m5 =	0, m6 =	0, m7 =	0
m8 =	0, m9 =	0, m10 =	0, m11 =	0, m12 =	0, m13 =	0, m14 =	0, m15 =	0
m16 =	0, m17 =	0, m18 =	0, m19 =	0, m20 =	0, m21 =	0, m22 =	0, m23 =	0
m24 =	0, m25 =	0, m26 =	0, m27 =	0, m28 =	0, m29 =	0, m30 =	0, m31 =	0
Registers								
R0 =	0, R1 =	1, R2 =	2, R3 =	3, R4 =	4, R5 =	5, R6 =	1, R7 =	2
R8 =	4, R9 =	2, 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 =	120, R30 =	0, R31 =	0

test2

```

R24 = 0, R25 = 0, R26 = 0, R27 = 0, R28 = 0, R29 = 128, R30 = 0, R31 = 16
PC = 16
m0 = 0, m1 = 0, m2 = 0, m3 = 0, m4 = 0, m5 = 0, m6 = 0, m7 = 0
m8 = 0, m9 = 0, m10 = 0, m11 = 0, m12 = 0, m13 = 0, m14 = 0, m15 = 0
m16 = 0, m17 = 0, m18 = 0, m19 = 0, m20 = 68, m21 = 2, m22 = 1, m23 = 68
m24 = 2, m25 = 1, m26 = addr_160, m27 = 4, m28 = 3, m29 = 16, m30 = 0, m31 = 0
Registers
R0 = 0, R1 = 0, R2 = 0, R3 = 0, R4 = 0, R5 = 0, R6 = 0, R7 = 0
R8 = 0, R9 = 0, 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 = 128, R30 = 0, R31 = 16
PC = 128
m0 = 0, m1 = 0, m2 = 0, m3 = 0, m4 = 0, m5 = 0, m6 = 0, m7 = 0

```

test3

```
PC = 60;
m0 = x, m1 = 0, m2 = 0, m3 = 0, m4 = 0, m5 = 0, m6 = 0, m7 = 0;
m8 = 0, m9 = 0, m10 = 0, m11 = 0, m12 = 0, m13 = 0, m14 = 0, m15 = 0;
m16 = 0, m17 = 0, m18 = 0, m19 = 0, m20 = 0, m21 = 0, m22 = 0, m23 = 0;
m24 = 0, m25 = 0, m26 = 0, m27 = 0, m28 = 0, m29 = 0, m30 = 0, m31 = 0;

Registers
R0 = 0, R1 = 0, R2 = 0, R3 = 0, R4 = 0, R5 = 0, R6 = 0, R7 = 0;
R8 = 0, R9 = 0, R10 = 0, R11 = 0, R12 = 0, R13 = 0, R14 = 0, R15 = 0;
R16 = 0, R17 = -120, 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;
```

Lesson learnt (if any):

verilog很難debug 最好多寫幾個測試擋 一個 instruction 一個一個測