

Computer Organization 2023 Lab 4

Finished part:

```
# run 50000ns
```

```
=====
```

```
(A) basic score:          75 / 75
```

```
=====
```

```
Congratulation. You pass TA's pattern
```

```
=====
```

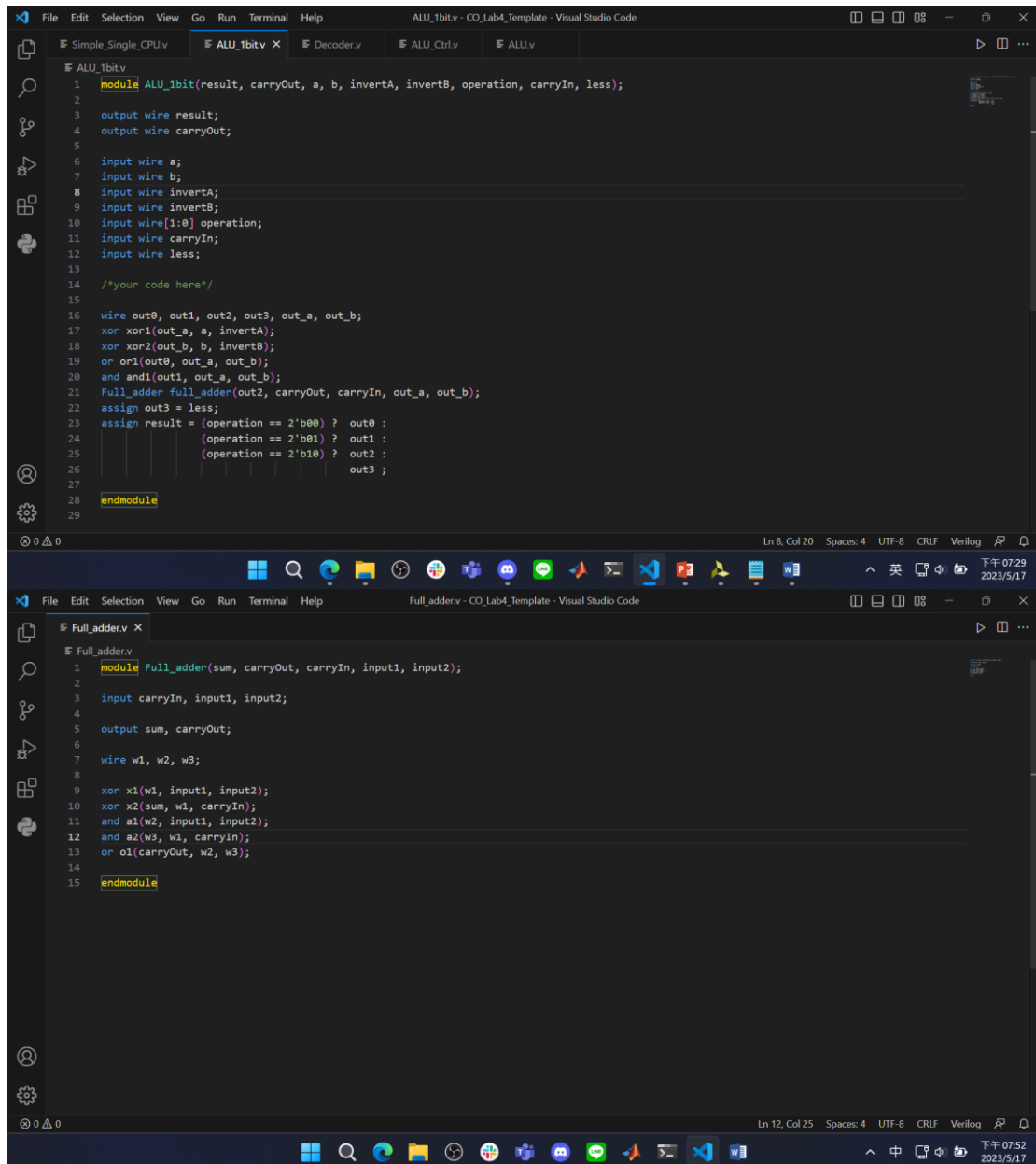
Problems you met and solutions:



這個報錯訊息是因為沒有把 Data_Memory 這個 module 命名為 DM。
改好後就沒事了。

If you create additional module, please give a short
explanation here:

因為要完成 32-bit ALU 之前，需要先定義 1-bit 的 ALU，所以我使用了在前幾次作業已經用過的自己寫的 1-bit ALU。而在 1-bit ALU 內，又定義了 Full-Adder，所以我共額外新增了這兩個檔案。



The image displays two screenshots of the Visual Studio Code editor, showing Verilog code for two modules: ALU_1bit and Full_adder.

Top Screenshot: ALU_1bit.v

```
1 module ALU_1bit(result, carryOut, a, b, invertA, invertB, operation, carryIn, less);
2
3   output wire result;
4   output wire carryOut;
5
6   input wire a;
7   input wire b;
8   input wire invertA;
9   input wire invertB;
10  input wire[1:0] operation;
11  input wire carryIn;
12  input wire less;
13
14  /*your code here*/
15
16  wire out0, out1, out2, out3, out_a, out_b;
17  xor xor1(out_a, a, invertA);
18  xor xor2(out_b, b, invertB);
19  or or1(out0, out_a, out_b);
20  and and1(out1, out_a, out_b);
21  Full_adder full_adder(out2, carryOut, carryIn, out_a, out_b);
22  assign out3 = less;
23  assign result = (operation == 2'b00) ? out0 :
24                  (operation == 2'b01) ? out1 :
25                  (operation == 2'b10) ? out2 :
26                  (operation == 2'b11) ? out3 ;
27
28  endmodule
```

Bottom Screenshot: Full_adder.v

```
1 module Full_adder(sum, carryOut, carryIn, input1, input2);
2
3   input carryIn, input1, input2;
4
5   output sum, carryOut;
6
7   wire w1, w2, w3;
8
9   xor x1(w1, input1, input2);
10  xor x2(sum, w1, carryIn);
11  and a1(w2, input1, input2);
12  and a2(w3, w1, carryIn);
13  or o1(carryOut, w2, w3);
14
15  endmodule
```

除此之外就沒有使用其他額外的.v 檔案了。