实验 0 实验提交流程

实验介绍

本实验将教大家如何通过 MIPS246 网站提交部件实验和 CPU 仿真实验

实验目标

成功提交相应实验所需要的文件，并通过网站反馈得到实验是否正确。

实验步骤

部件实验提交步骤：

1. 将所需提交的文件（汇编实验提交 ASM 文件；MULT/DIV/CP0 等实验提交 V 文件）打包成 zip 格式（注意格式!!!!）。
2. 使用自己的账户登录，点击提交作业->提交计组作业。

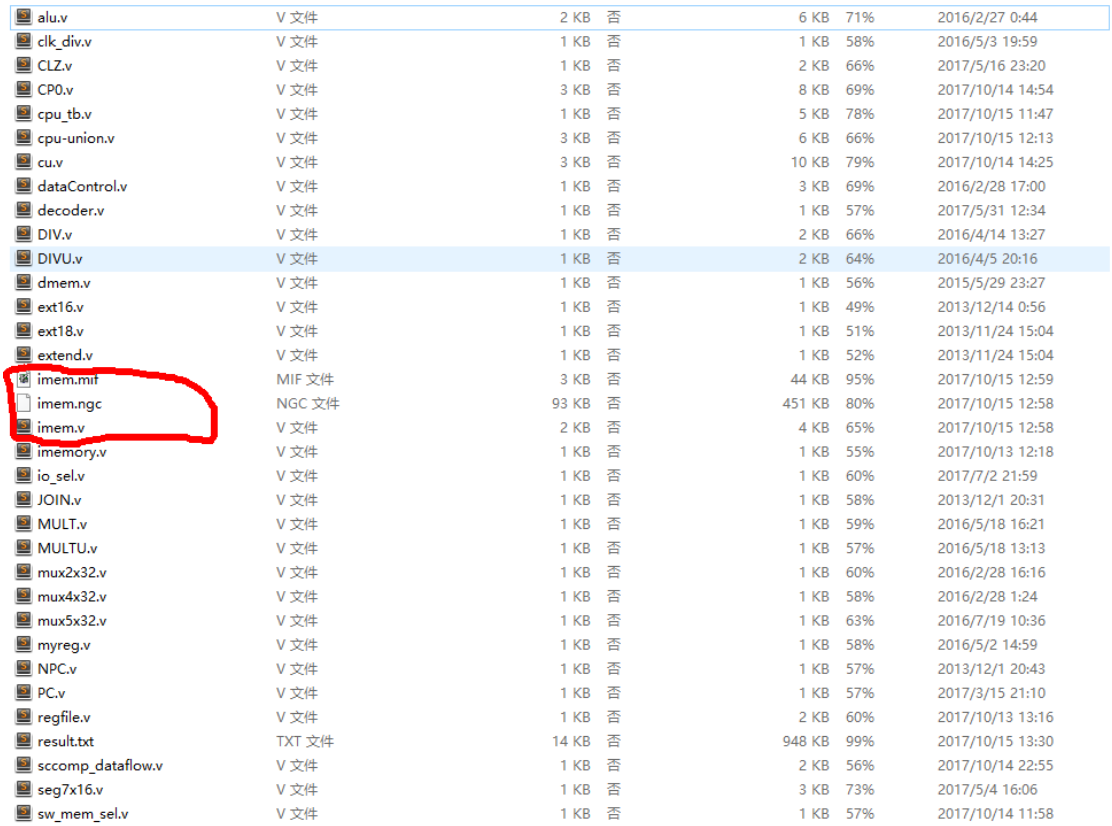
3. 选择相应实验，将出现如下页面。

注意：每次实验一定要在截止时间前提交，若超出时间或提交次数后本次作业将无法正常提交！！！

4. 对于提交的实验，网站会给出以下结果：
   A. CE 表示提交的实验在编译时出错，testlog.txt 中会有提示编译出错的信息。
   B. WA 表示实验编译通过，但结果有误，diff.txt 中会有提示错误信息，请继续测试。
   C. AC 表示实验通过检测并被接受（通过的是我们编写的 testbench，可以保证基本功能的正确性，但并不保证模块编写完全无误）。
   D. QD 表示正在排队，结果稍后会给出，切勿连续刷新页面，稍等片刻再进行刷新。

CPU 仿真工程提交：

1. 统一要求指令存储器使用 IP 核实现，提交的文件包括所有自己编写的模块（testbench 不要求提交）、IP 核相关文件（V、MIF、NGC），然后将这些文件压缩成 ZIP 格式提交。除了要求外的文件一律不能提交，若因此产生测试不通过的情况，我们将不会有补偿措施，所以要格外注意。
2. 示例说明如下：



alu.v	V 文件	2 KB	否	6 KB	71%	2016/2/27 0:44
clk_div.v	V 文件	1 KB	否	1 KB	58%	2016/5/3 19:59
CLZ.v	V 文件	1 KB	否	2 KB	66%	2017/5/16 23:20
CP0.v	V 文件	3 KB	否	8 KB	69%	2017/10/14 14:54
cpu_tb.v	V 文件	1 KB	否	5 KB	78%	2017/10/15 11:47
cpu-union.v	V 文件	3 KB	否	6 KB	66%	2017/10/15 12:13
cu.v	V 文件	3 KB	否	10 KB	79%	2017/10/14 14:25
dataControl.v	V 文件	1 KB	否	3 KB	69%	2016/2/28 17:00
decoder.v	V 文件	1 KB	否	1 KB	57%	2017/5/31 12:34
DIV.v	V 文件	1 KB	否	2 KB	66%	2016/4/14 13:27
DIVU.v	V 文件	1 KB	否	2 KB	64%	2016/4/5 20:16
dmem.v	V 文件	1 KB	否	1 KB	56%	2015/5/29 23:27
ext16.v	V 文件	1 KB	否	1 KB	49%	2013/12/14 0:56
ext18.v	V 文件	1 KB	否	1 KB	51%	2013/11/24 15:04
extend.v	V 文件	1 KB	否	1 KB	52%	2013/11/24 15:04
imem.mif	MIF 文件	3 KB	否	44 KB	95%	2017/10/15 12:59
imem.ngc	NGC 文件	93 KB	否	451 KB	80%	2017/10/15 12:58
imem.v	V 文件	2 KB	否	4 KB	65%	2017/10/15 12:58
imemory.v	V 文件	1 KB	否	1 KB	55%	2017/10/13 12:18
io_sel.v	V 文件	1 KB	否	1 KB	60%	2017/7/2 21:59
JOIN.v	V 文件	1 KB	否	1 KB	58%	2013/12/1 20:31
MULT.v	V 文件	1 KB	否	1 KB	59%	2016/5/18 16:21
MULTU.v	V 文件	1 KB	否	1 KB	57%	2016/5/18 13:13
mux2x32.v	V 文件	1 KB	否	1 KB	60%	2016/2/28 16:16
mux4x32.v	V 文件	1 KB	否	1 KB	58%	2016/2/28 1:24
mux5x32.v	V 文件	1 KB	否	1 KB	63%	2016/7/19 10:36
myreg.v	V 文件	1 KB	否	1 KB	58%	2016/5/2 14:59
NPC.v	V 文件	1 KB	否	1 KB	57%	2013/12/1 20:43
PC.v	V 文件	1 KB	否	1 KB	57%	2017/3/15 21:10
regfile.v	V 文件	1 KB	否	2 KB	60%	2017/10/13 13:16
result.txt	TXT 文件	14 KB	否	948 KB	99%	2017/10/15 13:30
sccomp_dataflow.v	V 文件	1 KB	否	2 KB	56%	2017/10/14 22:55
seg7x16.v	V 文件	1 KB	否	3 KB	73%	2017/5/4 16:06
sw_mem_sel.v	V 文件	1 KB	否	1 KB	57%	2017/10/14 11:58

红色标记中的文件是指令存储器 IP 核相关文件，其余文件为所有自己编写的文件，这里需要注意所有提交文件放在同一个路径下，最好是直接全选，然后压缩。

3. 顶层文件格式

   为了让网站能识别顶层文件，这里规定顶层文件名为 sccomp_dataflow.v，接口如下：

```
module sccomp_dataflow(
    input clk_in,
    input reset,
    output [31:0] inst,
    output [31:0] pc,
    output [31:0] addr
    );
```

4. Testbench 参考

```
module test;
  // Inputs
  reg clk_in;
  reg reset;
  // Outputs
  wire [31:0] inst;
  wire [31:0] pc;
  wire [31:0] addr;
  // Instantiate the Unit Under Test (UUT)
  sccomp_dataflow uut (
      .clk_in(clk_in),
      .reset(reset),
      .inst(inst),
      .pc(pc),
      .addr(addr)
  );
  integer file_output;
  initial begin
      //$dumpfile("mydump.txt");
      //$dumpvars(0,cpu_tb.uut.pcreg.data_out);
      file_output = $fopen("result.txt");
      // Initialize Inputs
      clk_in = 0;
      reset = 1;
      #1;
      reset = 0;
  end

  always begin
  #50;
  clk_in = ~clk_in;
  if(clk_in == 1'b1) begin
```

```
                $fdisplay(file_output, "pc: %h", test.uut.sccpu.pc_out);
                $fdisplay(file_output, "instr: %h", test.uut.sccpu.inst);
                $fdisplay(file_output, "regfile0: %h", test.uut.sccpu.cpu_ref.array_reg[0]);
                $fdisplay(file_output, "regfile1: %h", test.uut.sccpu.cpu_ref.array_reg[1]);
                $fdisplay(file_output, "regfile2: %h", test.uut.sccpu.cpu_ref.array_reg[2]);
                $fdisplay(file_output, "regfile3: %h", test.uut.sccpu.cpu_ref.array_reg[3]);
                $fdisplay(file_output, "regfile4: %h", test.uut.sccpu.cpu_ref.array_reg[4]);
                $fdisplay(file_output, "regfile5: %h", test.uut.sccpu.cpu_ref.array_reg[5]);
                $fdisplay(file_output, "regfile6: %h", test.uut.sccpu.cpu_ref.array_reg[6]);
                $fdisplay(file_output, "regfile7: %h", test.uut.sccpu.cpu_ref.array_reg[7]);
                $fdisplay(file_output, "regfile8: %h", test.uut.sccpu.cpu_ref.array_reg[8]);
                $fdisplay(file_output, "regfile9: %h", test.uut.sccpu.cpu_ref.array_reg[9]);
                $fdisplay(file_output, "regfile10: %h", test.uut.sccpu.cpu_ref.array_reg[10]);
                $fdisplay(file_output, "regfile11: %h", test.uut.sccpu.cpu_ref.array_reg[11]);
                $fdisplay(file_output, "regfile12: %h", test.uut.sccpu.cpu_ref.array_reg[12]);
                $fdisplay(file_output, "regfile13: %h", test.uut.sccpu.cpu_ref.array_reg[13]);
                $fdisplay(file_output, "regfile14: %h", test.uut.sccpu.cpu_ref.array_reg[14]);
                $fdisplay(file_output, "regfile15: %h", test.uut.sccpu.cpu_ref.array_reg[15]);
                $fdisplay(file_output, "regfile16: %h", test.uut.sccpu.cpu_ref.array_reg[16]);
                $fdisplay(file_output, "regfile17: %h", test.uut.sccpu.cpu_ref.array_reg[17]);
                $fdisplay(file_output, "regfile18: %h", test.uut.sccpu.cpu_ref.array_reg[18]);
                $fdisplay(file_output, "regfile19: %h", test.uut.sccpu.cpu_ref.array_reg[19]);
                $fdisplay(file_output, "regfile20: %h", test.uut.sccpu.cpu_ref.array_reg[20]);
                $fdisplay(file_output, "regfile21: %h", test.uut.sccpu.cpu_ref.array_reg[21]);
                $fdisplay(file_output, "regfile22: %h", test.uut.sccpu.cpu_ref.array_reg[22]);
                $fdisplay(file_output, "regfile23: %h", test.uut.sccpu.cpu_ref.array_reg[23]);
                $fdisplay(file_output, "regfile24: %h", test.uut.sccpu.cpu_ref.array_reg[24]);
                $fdisplay(file_output, "regfile25: %h", test.uut.sccpu.cpu_ref.array_reg[25]);
                $fdisplay(file_output, "regfile26: %h", test.uut.sccpu.cpu_ref.array_reg[26]);
                $fdisplay(file_output, "regfile27: %h", test.uut.sccpu.cpu_ref.array_reg[27]);
                $fdisplay(file_output, "regfile28: %h", test.uut.sccpu.cpu_ref.array_reg[28]);
                $fdisplay(file_output, "regfile29: %h", test.uut.sccpu.cpu_ref.array_reg[29]);
                $fdisplay(file_output, "regfile30: %h", test.uut.sccpu.cpu_ref.array_reg[30]);
                $fdisplay(file_output, "regfile31: %h", test.uut.sccpu.cpu_ref.array_reg[31]);
            end
        end
    endmodule
```

该 testbench 能将每个时钟周期的指令地址、指令、32 个寄存器内容打印到 result.txt 文件中，当然，如果习惯波形图的话也可以在 Modelsim 或 Vivado 的仿真波形图中看数值。注意：若实现 CPU 时，寄存器内容是靠时钟边沿触发的，那么 result.txt 文件中寄存器内容会在下一个周期更新。

5. 特殊情况

出现以下情况，请与助教联系：

A. 设计的是多周期 CPU。

B. result.txt 文件中寄存器内容更新不存在延迟现象。