


计算机系统 CPU 实验报告

一、小组成员及分工占比：

王子达（20206433）： 50%

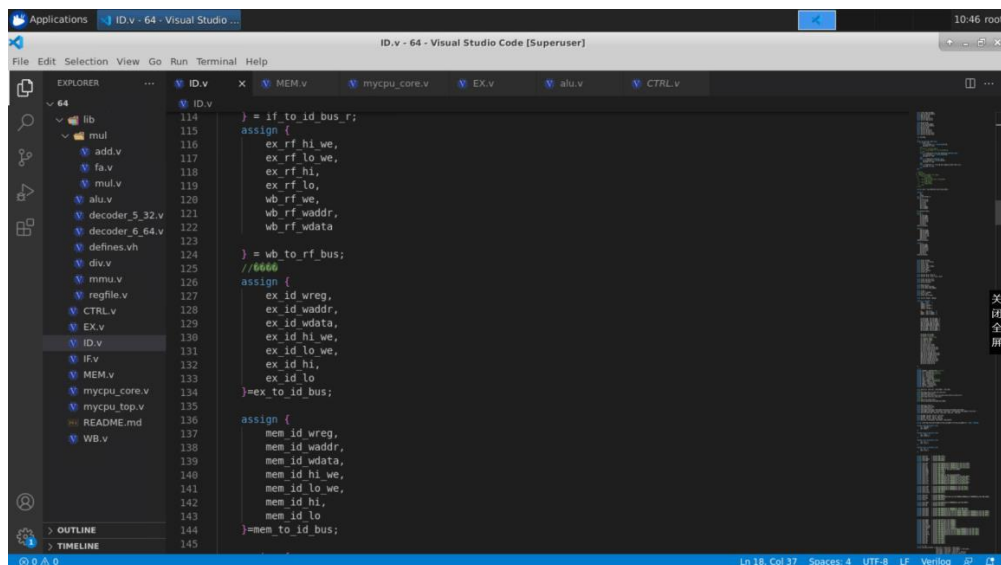
段培元（20206379）： 50%

具体的实验任务分工可参考 [github](#) 仓库上的 commit 过程

 ikun-king 抵达64号点	b938a4c 4 days ago	🕒 12 commits
SampleCPU	一号点	3 weeks ago
SampleCPU_cpu	抵达15号点	2 weeks ago
SampleCPU_cpu_36	抵达36号点	2 weeks ago
SampleCpu_51	Add files via upload	5 days ago
Sample_cpu64	抵达64号点	4 days ago
task-11.5-段培元	11.5-段培元	2 months ago
计算机系统stall以及后继指令添加	Add files via upload	3 weeks ago
README.md	Update README.md	last month

二、实验内容与体会：

最开始，是按照助教老师在腾讯会议录屏的讲解一点点在各.v、.vh 文件中先添加数据相关的指令，将 ex_to_rf_bus，mem_to_rf_bus，wb_to_rf_bus 等加好，和 id 段的 rfddata 写好。



```
114
115
116   } = if_to_id_bus_r;
117   assign (
118     ex_rf_hi_we,
119     ex_rf_lo_we,
120     ex_rf_hi,
121     ex_rf_lo,
122     wb_rf_we,
123     wb_rf_waddr,
124     wb_rf_wdata
125   ) = wb_to_rf_bus;
126   //6666
127   assign (
128     ex_id_wreg,
129     ex_id_waddr,
130     ex_id_wdata,
131     ex_id_hi_we,
132     ex_id_lo_we,
133     ex_id_hi,
134     ex_id_lo
135   ) = ex_to_id_bus;
136
137   assign (
138     mem_id_wreg,
139     mem_id_waddr,
140     mem_id_wdata,
141     mem_id_hi_we,
142     mem_id_lo_we,
143     mem_id_hi,
144     mem_id_lo
145   ) = mem_to_id_bus;
```

```

114     } = if_to_id_bus_r;
115     assign {
116         ex_rf_hi_we,
117         ex_rf_lo_we,
118         ex_rf_hi,
119         ex_rf_lo,
120         wb_rf_we,
121         wb_rf_waddr,
122         wb_rf_wdata
123     } = wb_to_rf_bus;
124     //0000
125     assign {
126         ex_id_wreg,
127         ex_id_waddr,
128         ex_id_wdata,
129         ex_id_hi_we,
130         ex_id_lo_we,
131         ex_id_hi,
132         ex_id_lo
133     } = ex_to_id_bus;
134     assign {
135         mem_id_wreg,
136         mem_id_waddr,
137         mem_id_wdata,
138         mem_id_hi_we,
139         mem_id_lo_we,
140         mem_id_hi,
141         mem_id_lo
142     } = mem_to_id_bus;
143
144
145

```

```

59     data_ram_wen,
60     sel_rf_res,
61     rf_we,
62     rf_waddr,
63     ex_result
64 } = ex_to_mem_bus_r;
65
66 //assign rf_wdata = sel_rf_res ? mem_result : ex_result;
67
68 assign rf_wdata = (data_ram_readen==4'b1111 && data_ram_en==1'b1) ? data_sram_rdata
69 : (data_ram_readen==4'b0001 && data_ram_en==1'b1 && ex_result[1:0]==2'b00) ? ({24{data_sram_rdata[7]}},data
70 : (data_ram_readen==4'b0001 && data_ram_en==1'b1 && ex_result[1:0]==2'b01) ? ({24{data_sram_rdata[15]}},data
71 : (data_ram_readen==4'b0001 && data_ram_en==1'b1 && ex_result[1:0]==2'b10) ? ({24{data_sram_rdata[23]}},data
72 : (data_ram_readen==4'b0001 && data_ram_en==1'b1 && ex_result[1:0]==2'b11) ? ({24{data_sram_rdata[31]}},data
73 : (data_ram_readen==4'b0010 && data_ram_en==1'b1 && ex_result[1:0]==2'b00) ? ({24'b0,data_sram_rdata[7:0]}},data
74 : (data_ram_readen==4'b0010 && data_ram_en==1'b1 && ex_result[1:0]==2'b01) ? ({24'b0,data_sram_rdata[15:8]}},data
75 : (data_ram_readen==4'b0010 && data_ram_en==1'b1 && ex_result[1:0]==2'b10) ? ({24'b0,data_sram_rdata[23:16]}},data
76 : (data_ram_readen==4'b0010 && data_ram_en==1'b1 && ex_result[1:0]==2'b11) ? ({24'b0,data_sram_rdata[31:24]}},data
77 : (data_ram_readen==4'b0011 && data_ram_en==1'b1 && ex_result[1:0]==2'b00) ? ({16{data_sram_rdata[15]}},data
78 : (data_ram_readen==4'b0011 && data_ram_en==1'b1 && ex_result[1:0]==2'b01) ? ({16{data_sram_rdata[31]}},data
79 : (data_ram_readen==4'b0100 && data_ram_en==1'b1 && ex_result[1:0]==2'b00) ? ({16'b0,data_sram_rdata[15:0]}},data
80 : (data_ram_readen==4'b0100 && data_ram_en==1'b1 && ex_result[1:0]==2'b10) ? ({16'b0,data_sram_rdata[31:16]}},data
81 : ex_result;
82
83
84
85
86
87 assign mem_to_wb_bus =

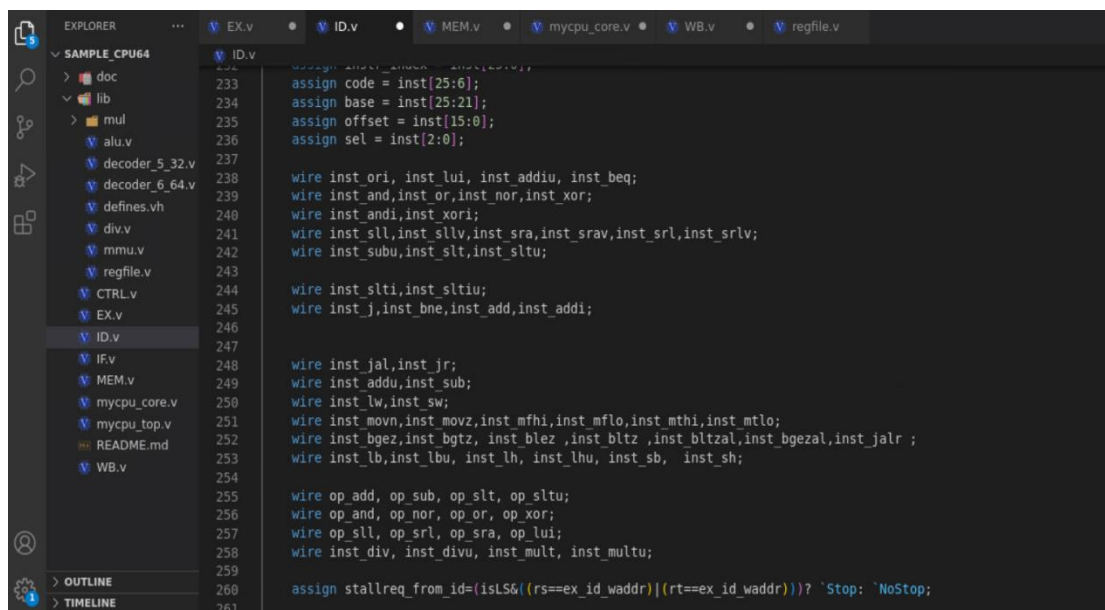
```

加完数据相关的东西之后会卡在这里：

reference: PC = 0xbfc006f8, wb_rf_wnum = 0x19,
wb_rf_wdata = 0x9fc00704

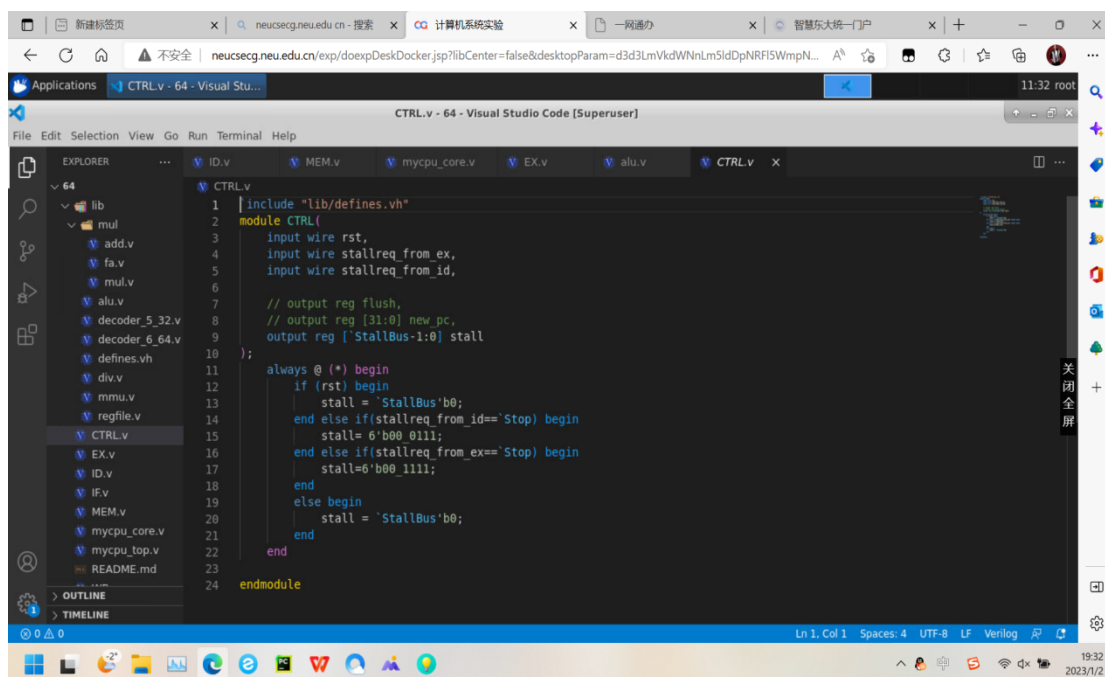
mycpu : PC = 0xbfc00714, wb_rf_wnum = 0x19, wb_rf_wdata
= 0xbfc00000

subu 指令没加，从这里开始加指令。具体如下：



```
232 assign inst_opcode = inst[25:0];
233 assign code = inst[25:6];
234 assign base = inst[25:21];
235 assign offset = inst[15:0];
236 assign sel = inst[2:0];
237
238 wire inst_ori, inst_lui, inst_addiu, inst_beq;
239 wire inst_and, inst_or, inst_nor, inst_xor;
240 wire inst_andi, inst_xori;
241 wire inst_sll, inst_sllv, inst_sra, inst_srav, inst_srl, inst_srlv;
242 wire inst_subu, inst_slt, inst_sltu;
243
244 wire inst_slti, inst_sltiu;
245 wire inst_j, inst_bne, inst_add, inst_addi;
246
247
248 wire inst_jal, inst_jr;
249 wire inst_addu, inst_sub;
250 wire inst_lw, inst_sw;
251 wire inst_movn, inst_movz, inst_mfhi, inst_mflo, inst_mthi, inst_mtlo;
252 wire inst_bgez, inst_bgtz, inst_blez, inst_bltz, inst_bltzal, inst_bgezal, inst_jalr;
253 wire inst_lb, inst_lbu, inst_lh, inst_lhu, inst_sb, inst_sh;
254
255 wire op_add, op_sub, op_slt, op_sltu;
256 wire op_and, op_nor, op_or, op_xor;
257 wire op_sll, op_srl, op_sra, op_lui;
258 wire inst_div, inst_divu, inst_mult, inst_multu;
259
260 assign stallreq_from_id = (isLS6((rs==ex_id_waddr) || (rt==ex_id_waddr))) ? `Stop : `NoStop;
261
```

当时在度过一号点前卡了一段时间的 bug，不过好在有同学和助教指导下，顺利地度过了一号点，在过完一号点之后，按照助教录屏的讲解，接下来需要添加一些 stall 指令。



```
1 include "lib/defines.vh"
2 module CTRL(
3     input wire rst,
4     input wire stallreq_from_ex,
5     input wire stallreq_from_id,
6
7     // output reg flush,
8     // output reg [31:0] new_pc,
9     output reg [31:0] stall
10 );
11
12 always @ (*) begin
13     if (rst) begin
14         stall = `StallBus'b0;
15     end else if (stallreq_from_id == `Stop) begin
16         stall = 6'b00_0111;
17     end else if (stallreq_from_ex == `Stop) begin
18         stall = 6'b00_1111;
19     end
20     else begin
21         stall = `StallBus'b0;
22     end
23 end
24 endmodule
```

```

98  .ex_to_mem_bus  (ex_to_mem_bus  ),
99  .data_sram_rdata (data_sram_rdata),
100 .mem_to_wb_bus  (mem_to_wb_bus  ),
101
102 .mem_to_id_bus  (mem_to_id_bus  )
103 );
104
105 WB u_WB(
106     .clk          (clk          ),
107     .rst          (rst          ),
108     .stall        (stall        ),
109     .mem_to_wb_bus (mem_to_wb_bus),
110     .wb_to_rf_bus  (wb_to_rf_bus ),
111     .debug_wb_pc   (debug_wb_pc  ),
112     .debug_wb_rf_wen (debug_wb_rf_wen),
113     .debug_wb_rf_wnum (debug_wb_rf_wnum),
114     .debug_wb_rf_wdata (debug_wb_rf_wdata),
115
116     .wb_to_id_bus  (wb_to_id_bus  )
117 );
118
119 CTRL u_CTRL(
120     .stallreq_from_id(stallreq_from_id),
121     .stallreq_from_ex(stallreq_from_ex),
122     .rst             (rst             ),
123     .stall           (stall           )
124 );
125
126 endmodule

```

这一部分结束之后，继续添加其他相关指令，到达了 15 号点，再接下来，是对着 A03 那个文档，查看 PC 值，记得几乎全是改的 id.v，重复性地添加指令，部分截图如下：

```

12134 bfc09e00: 39030000  ori v1,v1,0A000
12135 bfc09e6c: 39020000  xori v0,t0,0x0
12136 bfc09e70: 14430149  bne v0,v1,bfc0a398 <inst_error>
12137 bfc09e74: 00000000  nop
12138 /media/sf_nscscc2019/develop/trash/func_test_v0.03/soft/func/inst/n31_xori.S:250
12139 bfc09e78: 3c089411  lui t0,0x9411
12140 bfc09e7c: 3508ecad  ori t0,t0,0xecad
12141 bfc09e80: 3c039411  lui v1,0x9411
12142 bfc09e84: 3463ecad  ori v1,v1,0xecad
12143 bfc09e88: 39020000  xori v0,t0,0x0
12144 bfc09e8c: 14430142  bne v0,v1,bfc0a398 <inst_error>
12145 bfc09e90: 00000000  nop
12146 /media/sf_nscscc2019/develop/trash/func_test_v0.03/soft/func/inst/n31_xori.S:251
12147 bfc09e94: 3c08918b  lui t0,0x918b
12148 bfc09e98: 3508e1ef  ori t0,t0,0xe1ef
12149 bfc09e9c: 3c03918b  lui v1,0x918b
12150 bfc09ea0: 3463e1ef  ori v1,v1,0xe1ef
12151 bfc09ea4: 39020000  xori v0,t0,0x0
12152 bfc09ea8: 1443013b  bne v0,v1,bfc0a398 <inst_error>
12153 bfc09eac: 00000000  nop
12154 /media/sf_nscscc2019/develop/trash/func_test_v0.03/soft/func/inst/n31_xori.S:252
12155 bfc09eb0: 3c08b47c  lui t0,0xb47c
12156 bfc09eb4: 3508ced9  ori t0,t0,0xcd9
12157 bfc09eb8: 3c03b47c  lui v1,0xb47c
12158 bfc09ebc: 3463ced9  ori v1,v1,0xcd9
12159 bfc09ec0: 39020000  xori v0,t0,0x0
12160 bfc09ec4: 14430134  bne v0,v1,bfc0a398 <inst_error>
12161 bfc09ec8: 00000000  nop
12162 /media/sf_nscscc2019/develop/trash/func_test_v0.03/soft/func/inst/n31_xori.S:253
12163 bfc09ecc: 3c084bbe  lui t0,0x4bbe

```

```

235 assign offset = inst[15:0];
236 assign sel = inst[2:0];
237
238 wire inst_ori, inst_lui, inst_addiu, inst_beq;
239 wire inst_and, inst_or, inst_nor, inst_xor;
240 wire inst_andi, inst_xori;
241 wire inst_sll, inst_sllv, inst_sra, inst_srav, inst_srl, inst_srlv;
242 wire inst_subu, inst_slt, inst_sltu;
243
244 wire inst_slti, inst_sltiu;
245 wire inst_j, inst_bne, inst_add, inst_addi;
246
247
248 wire inst_jal, inst_jr;
249 wire inst_addu, inst_sub;
250 wire inst_lw, inst_sw;
251 wire inst_movn, inst_movz, inst_mfhi, inst_mflo, inst_mthi, inst_mtlo;
252 wire inst_bgez, inst_bgtz, inst_blez, inst_bltz, inst_bltzal, inst_bgezal, inst_jalr;
253 wire inst_lb, inst_lbu, inst_lh, inst_lhu, inst_sb, inst_sh;
254
255 wire op_add, op_sub, op_slt, op_sltu;
256 wire op_and, op_nor, op_or, op_xor;
257 wire op_sll, op_srl, op_sra, op_lui;
258 wire inst_div, inst_divu, inst_mult, inst_multu;
259
260 assign stallreq_from_id = (isLSG((rs==ex_id_waddr) || (rt==ex_id_waddr))) ? `Stop: `NoStop;
261
262 decoder_6_64 u0_decoder_6_64(
263     .in (opcode ),

```

```

290 assign inst_nor = op_d[6'b00_0000] & (sa==5'b00000) & func_d[6'b10_0111];
291 assign inst_xor = op_d[6'b00_0000] & (sa==5'b00000) & func_d[6'b10_0110];
292 assign inst_add = op_d[6'b00_0000] & func_d[6'b10_0000];
293 assign inst_addi = op_d[6'b00_1000];
294 assign inst_andi = op_d[6'b00_1100];
295 assign inst_xori = op_d[6'b00_1110];
296 assign inst_sub = op_d[6'b00_0000] & func_d[6'b10_0010];
297 assign inst_sll = op_d[6'b00_0000] & rs_d[5'b00000] & func_d[6'b00_0000];
298 assign inst_sllv = op_d[6'b00_0000] & sa==5'b00000 & func_d[6'b00_0100];
299 assign inst_sra = op_d[6'b00_0000] & rs_d[5'b00000] & func_d[6'b00_0011];
300 assign inst_srav = op_d[6'b00_0000] & sa==5'b00000 & func_d[6'b00_0111];
301 assign inst_srl = op_d[6'b00_0000] & rs_d[5'b00000] & func_d[6'b00_0010];
302 assign inst_srlv = op_d[6'b00_0000] & sa==5'b00000 & func_d[6'b00_0110];
303
304 assign inst_subu = op_d[6'b00_0000] & sa==5'b00000 & func_d[6'b10_0011];
305 assign inst_slt = op_d[6'b00_0000] & sa==5'b00000 & func_d[6'b10_1010];
306 assign inst_sltu = op_d[6'b00_0000] & sa==5'b00000 & func_d[6'b10_1011];
307 assign inst_slti = op_d[6'b00_1010];
308 assign inst_sltiu = op_d[6'b00_1011];
309
310 assign inst_jal = op_d[6'b00_0011];
311 assign inst_jr = op_d[6'b00_0000] & inst[20:11] == 10'b00000_00000 & sa==5'b00000 & func_d[6'b00_1000];
312 assign inst_bne = op_d[6'b00_0101];
313 assign inst_j = op_d[6'b00_0010];
314
315 assign inst_addu = op_d[6'b00_0000] & sa==5'b00000 & func_d[6'b10_0001];
316 assign inst_lw = op_d[6'b10_0011];
317 assign inst_sw = op_d[6'b10_1011];
318

```

```

345 // rs to reg1
346 assign sel_alu_src1[0] = inst_srlv | inst_srav | inst_sllv | inst_subu |
347     inst_xori | inst_ori | inst_addiu | inst_subu |
348     inst_andi | inst_and | inst_slt |
349     inst_sltu | inst_nor | inst_xor | inst_or |
350     inst_addu | inst_lw | inst_sw | inst_slti |
351     inst_sltiu | inst_add | inst_addi | inst_sub |
352     inst_div | inst_divu | inst_mult | inst_multu | inst_jalr |
353     inst_mthi | inst_mtlo | inst_sh | inst_sb | inst_lhu |
354     inst_lh | inst_lb | inst_lbu;
355
356 // pc to reg1
357 assign sel_alu_src1[1] = inst_jal |
358     inst_bltzal | inst_bgezal | inst_jalr;
359
360 // sa zero extend to reg1
361 assign sel_alu_src1[2] = inst_sll | inst_srl | inst_sra;
362
363 // rt to reg2
364 assign sel_alu_src2[0] = inst_srlv | inst_srav | inst_sllv | inst_subu |
365     inst_and | inst_slt | inst_sltu | inst_nor | inst_xor |
366     inst_sll | inst_srl | inst_sra | inst_or | inst_addu | inst_add | inst_sub |
367     inst_div | inst_divu | inst_mult | inst_multu | inst_sh | inst_sb | inst_lhu |
368     inst_lh | inst_lb | inst_lbu;
369
370 // imm sign extend to reg2
371 assign sel_alu_src2[1] = inst_lui | inst_addiu | inst_lw | inst_sw | inst_slti |
372     inst_sltiu | inst_addi |
373     inst_sh | inst_sh | inst_lbu | inst_lh | inst_lh | inst_lh | inst_lh;
374

```


通过运行仿真结果的 PC 值和目标 PC 值进行比对,在提供的 Test.s 文件中查找目标 PC 值,根据 PC 值对应的指令,在 A03_“系统能力培养大赛”MIPS 指令系统规范 pdf 文件里查找该指令,根据指令的不同功能进行添加代码,例如涉及到 rs、rt 寄存器的,就添加到相对应的地方(如上图),这样重复性地添加一些相关指令代码(上图中展示的加指令 slti, sltiu, j, add, addi, sub, and, andi, nor, xori, sllv, sra, srav, srl, srlv) 过后,测试到达了 36 号点。

接下来是加指令 bgez, bgtz, blez, bltz, bltzal, bgezal, jalr。测试过后可以到达过 43 号点,到这个位置:

reference: PC = 0xbfc560a0, wb_rf_wnum = 0x02, wb_rf_wdata = 0x40200000

mycpu : PC = 0xbfc560e0, wb_rf_wnum = 0x15, wb_rf_wdata = 0x40200000。这一部分稍微有意思的东西就是分支跳转指令你需要加入自己的判断方法,就是 rs_ge gt le lt_rt 那些东西。不过仍然还是比这 A03 文档继续添加。在这个过程中还学了学某些英文缩写都是什么意思: 例如 ge-greater than or equal to-大于等于, gt-greater than-大于, le-less than or equal to-小于等于, lt-less than-小于。

再接下来,加的指令有 mfhi, mflo, mthi, mtlo, div, divu, mult, multu。这一部分,继续添加指令会遇到 hilo 寄存器相关的问题,我们的理解是需要将 hilo_bus 加到很多的总线中去,比如说 ex_to_mem_bus, ex_to_rf_bus……我们在 id 段需要像那 32 个通用

寄存器一样取出 hilo 寄存器中的值，这里需要注意数据相关问题，模仿 rf_rdata 和 rdata 那样子进行处理，最后 hilo 寄存器中的值作为 hi_i, lo_i 传入 ex，在 ex 段做的事情是，如果是要将数据写入到 hilo 寄存器，如 mthi, mul, div 之类的指令，hi_o, lo_o 用来存储要写入的数据，并作为 hilo_bus 中的一部分往后传，就像 rf_wdata 一样。如果是 mfhi, mflo 之类将 hilo 寄存器中的值取出，存入到通用寄存器中，那这一部分 ex_result 将会被赋值为 hilo 寄存器中的值 (hi_i, lo_i)。具体的代码过程截图如下：

```
435 assign hi_read = inst_mfhi;
436 assign lo_read = inst_mflo;
437 assign hi_write = inst_mthi;
438 assign lo_write = inst_mtlo;
439 // regfile store enable
440 assign rf_we = inst_srlv | inst_srav | inst_sllv | inst_xori |
441 inst_or | inst_ori | inst_lui | inst_addiu | inst_subu |
442 inst_and | inst_andi | inst_slt | inst_sltu | inst_nor |
443 inst_xor | inst_sll | inst_srl | inst_sra | inst_jal |
444 inst_adduj | inst_lw | inst_slti | inst_sltiu | inst_addi | inst_sub |
445 inst_bgezal | inst_bltzal | inst_jalr | inst_mfhi | inst_mflo;
446 inst_lhu | inst_lh | inst_lbu | inst_lb;
447
448 // store in [rd]
449 assign sel_rf_dst[0] = inst_srlv | inst_srav | inst_sllv | inst_subu |
450 inst_and | inst_andi | inst_slt | inst_sltu | inst_nor | inst_xor |
451 inst_sll | inst_srl | inst_sra | inst_adduj | inst_or | inst_addi | inst_sub |
452 inst_jalr | inst_mfhi | inst_mflo;
453
454 // store in [rt]
455 assign sel_rf_dst[1] = inst_xori | inst_ori | inst_lui | inst_addiu | inst_andi |
456 inst_lw | inst_slti | inst_sltiu | inst_addi |
457 inst_lhu | inst_lh | inst_lbu | inst_lb;
458
459 // store in [31]
460 assign sel_rf_dst[2] = inst_jal |
461 inst_bltzal | inst_bgezal;
```

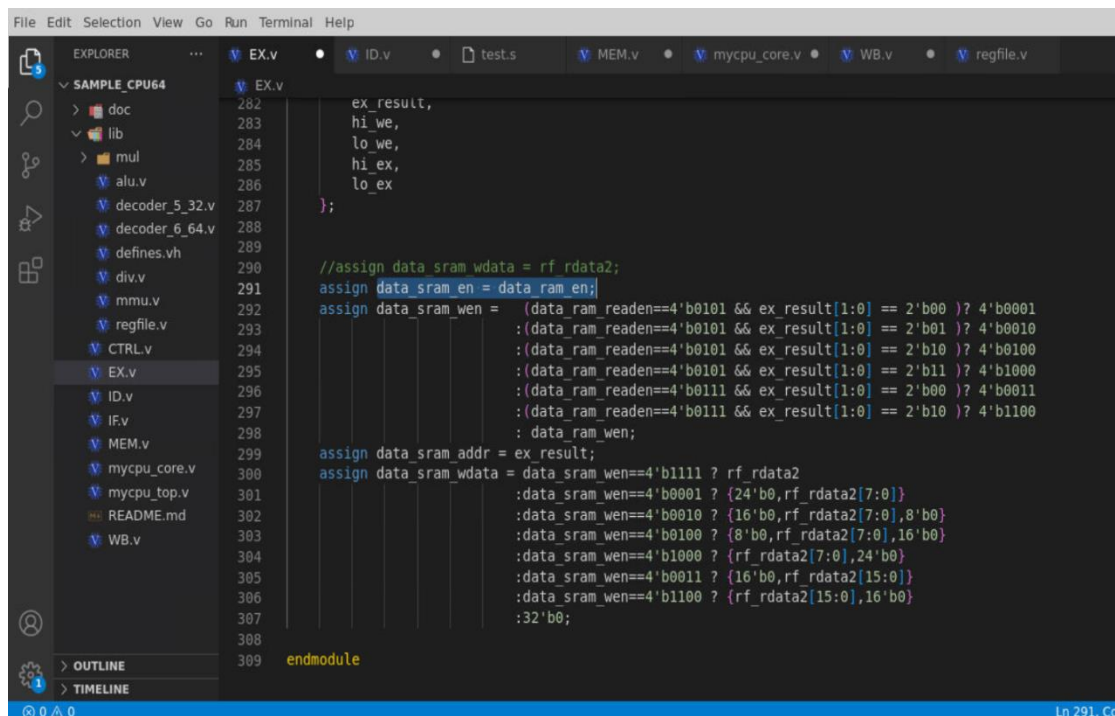
```
453 | {5(sel_rf_dst[1])} & rt
454 | {5(sel_rf_dst[2])} & 32'd31;
455
456 // 0 from alu_res ; 1 from ld_res
457 // assign sel_rf_res = inst_lw;
458 assign sel_rf_res = 1'b0;
459
460 assign id_to_ex_bus = {
461 data_ram_readen,
462 hi_write,
463 lo_write,
464 hi_read,
465 lo_read,
466 hi_out_file,
467 lo_out_file,
468 id_pc,
469 inst,
470 alu_op,
471 sel_alu_src1,
472 sel_alu_src2,
473 data_ram_en,
474 data_ram_wen,
475 rf_we,
476 rf_waddr,
477 sel_rf_res,
478 rdata1,
479 rdata2
480 };
481
```

加完 hilo 可以到达这里：

reference: PC = 0xbfc7d7dc, wb_rf_wnum = 0x15, wb_rf_wdata = 0x00000002

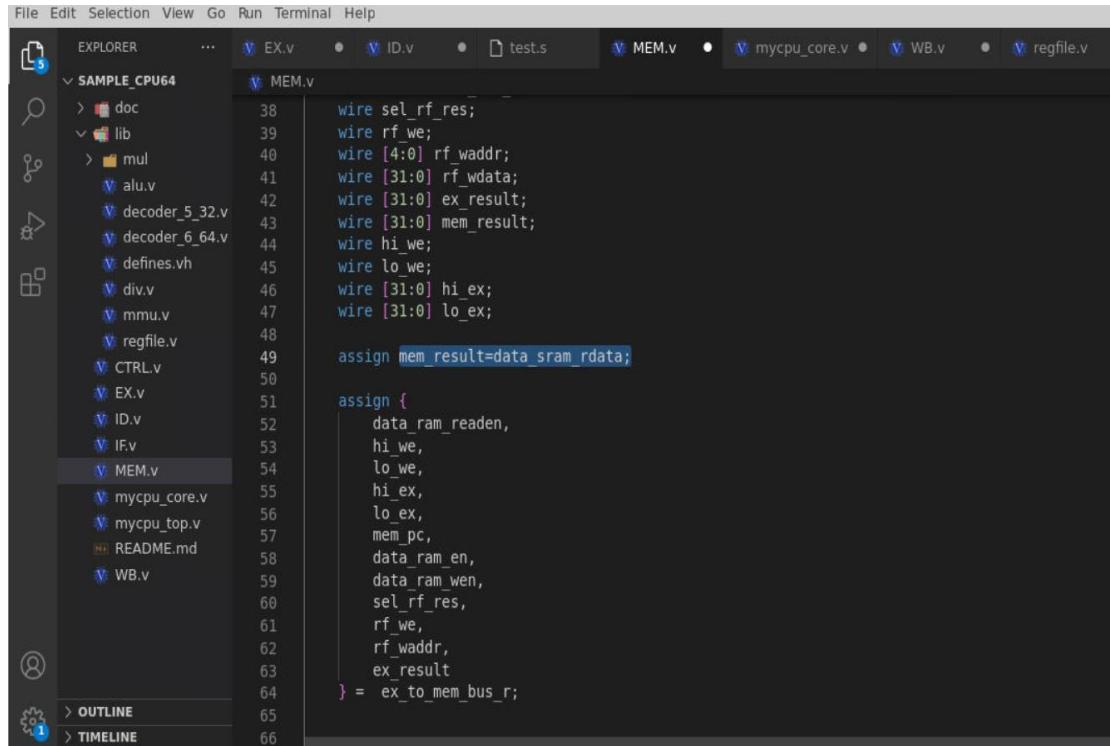
mycpu : PC = 0xbfc7d7dc, wb_rf_wnum = 0x15, wb_rf_wdata = 0x00000000, 之后在这个地方卡了很长时间, 然后也是通过询问已经通过的小组, 帮助我们解决了相关的 bug, 到达了 51 号点。

再接下来加的指令有 lb, lbu, lh, lhu, sb, sh。我们觉得在 ex 段主要就是改对于 data_sram_wen 和 data_sram_wdata 这两个, 就是按字节把通用寄存器里的值往 sram 里存, store 指令。mem 段改的是 mem_result, 目的是按字节读 data_sram_rdata 到寄存器里, load 指令。

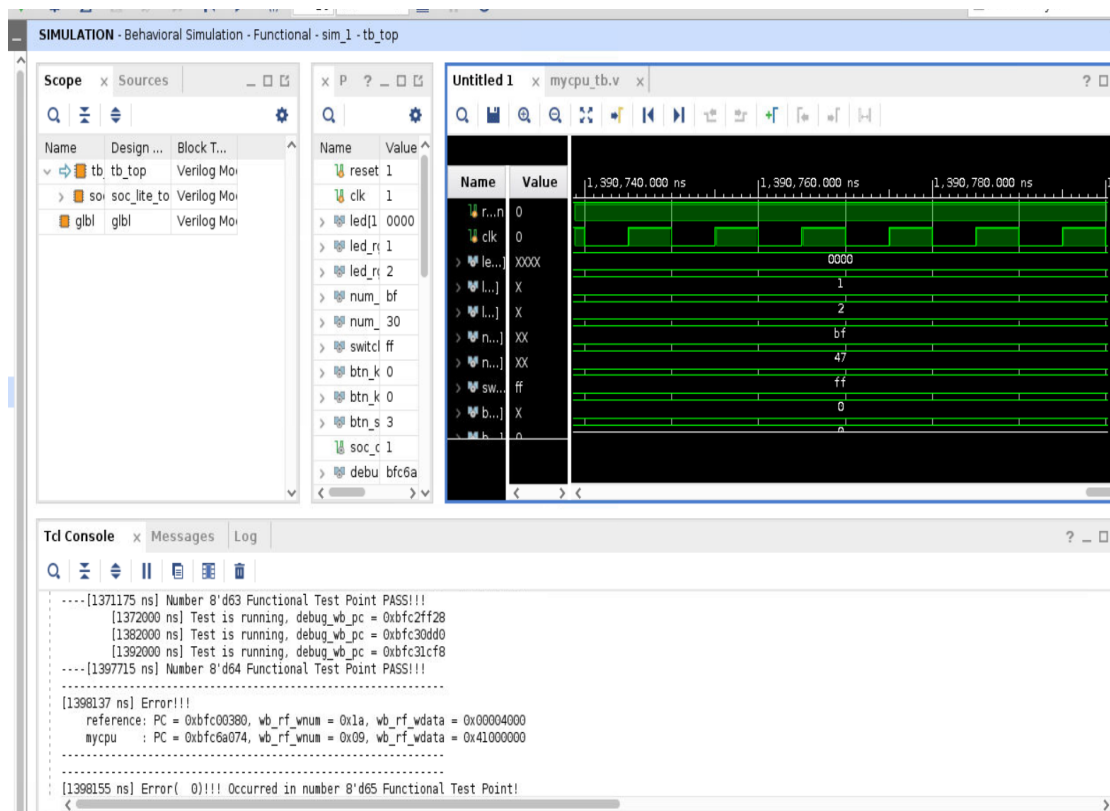


```
File Edit Selection View Go Run Terminal Help
EXPLORER
SAMPLE_CPU64
  > doc
  > lib
  > mul
    alu.v
    decoder_5_32.v
    decoder_6_64.v
    defines.vh
    div.v
    mmu.v
    regfile.v
    CTRL.v
    EX.v
    ID.v
    IF.v
    MEM.v
    mycpu_core.v
    mycpu_top.v
    README.md
    WB.v
  > OUTLINE
  > TIMELINE

EX.v
282     ex_result,
283     hi_we,
284     lo_we,
285     hi_ex,
286     lo_ex
287   );
288
289   //assign data_sram_wdata = rf_rdata2;
290   assign data_sram_en = data_ram_en;
291   assign data_sram_wen = (data_ram_readen==4'b0101 && ex_result[1:0] == 2'b00) ? 4'b0001
292   : (data_ram_readen==4'b0101 && ex_result[1:0] == 2'b01) ? 4'b0010
293   : (data_ram_readen==4'b0101 && ex_result[1:0] == 2'b10) ? 4'b0100
294   : (data_ram_readen==4'b0101 && ex_result[1:0] == 2'b11) ? 4'b1000
295   : (data_ram_readen==4'b0111 && ex_result[1:0] == 2'b00) ? 4'b0011
296   : (data_ram_readen==4'b0111 && ex_result[1:0] == 2'b10) ? 4'b1100
297   : data_ram_wen;
298
299   assign data_sram_addr = ex_result;
300   assign data_sram_wdata = data_sram_wen==4'b1111 ? rf_rdata2
301   : data_sram_wen==4'b0001 ? {24'b0,rf_rdata2[7:0]}
302   : data_sram_wen==4'b0010 ? {16'b0,rf_rdata2[7:0],8'b0}
303   : data_sram_wen==4'b0100 ? {8'b0,rf_rdata2[7:0],16'b0}
304   : data_sram_wen==4'b1000 ? {rf_rdata2[7:0],24'b0}
305   : data_sram_wen==4'b0011 ? {16'b0,rf_rdata2[15:0]}
306   : data_sram_wen==4'b1100 ? {rf_rdata2[15:0],16'b0}
307   : 32'b0;
308
309   endmodule
```

写完这些，测试便可抵达 64 号点，最终的结果展示如下：



三、实验心得：

王子达：

我认为，本次这个 **cpu** 实验过程还是比较艰难的，在最开始的时候，我认为也是最艰难的，当时分完组之后，留了第一个小任务，我记得是修改模 10 计数器，模 6 计数器那些，在这之前，我们必须先对 Verilog 语言做一些必备的简单了解，在看懂代码之后，小组商量着进行修改，并且刚开始还熟悉了 github 官网的相关知识，从注册登录到建立分支 **branch** 上传文件，后来过了一段时间，陆续开展正式的 **cpu** 实验，在最开始的时候，我觉得什么都不懂，一头雾水，后来通过助教的视频讲解知道了很多东西，并按照视频里面的代码一步一步照着在自己的 CG 平台上的 VScode 里进行编码，在度过每一个测试点的过程中也是非常地不易，不过好在每次遇到我们不会修改的 **bug**，都可以问其他的同学和助教，帮助我们顺利地度过了测试点位，调试过程中遇到的 **bug** 有太多太多，有的是语法错误，有的是指令不全导致仿真无法运行，还有一些 **bug** 可能现在也不能很好地理解，不过最终完成要求，跑完 64 号点之后，还是比较高兴的，最后我想说，虽然在本次 **cpu** 实验中我们遇到了很多问题（除了学术上的，中间还有相当长一段时间阳性发烧，被拉去酒店隔离等等），但是很感谢这次实验使我们得以锻炼，也希望以后的 **cpu** 实验可以越来越好！

段培元：

首先，整体回顾我们组做 **CPU** 实验的时候，更多感到的是不断前进的收获和喜悦，尽管我们在最初和一段时间里遇到了些麻烦和问题，

但是好在我们坚持了下来，可能我们进度有些缓慢，可正如龟兔赛跑里面的乌龟，慢但不断地前进，最终能够取得胜利，我想这正是我们做完此次实验的原因，拥有坚忍不拔、持之以恒的毅力和品质才是通往成功的必要条件。

在最开始做小任务的时候，对于一个完全陌生的语言，我们首先进行了语言的了解和学习，在学习 Verilog 之后，发现其和 C 语言很相似，这便给我带来了信心，顺利完成修改模 10 计数器的小任务后，我们对于接下来的 CPU 实验充满了信心。然而事实并非如此，我们对实验一时间无从下手，但是方法总比困难多，我们通过请教助教和同学，学习相关原理，添加了一些指令、解决数据相关问题等，在十二月伊始顺利通过一号点，之后工作便有序了起来。期间还被拉去隔离了好几天，在此期间我们也没忘记 CPU 实验，磕磕绊绊地添加指令，根据助教的录屏和自我学习，例如加 stall 等，顺利通过 15 点。在接下来的时间里，按照 PC 加指令成为主要，加完能够过 43 号点，此时收获的喜悦溢于言表，而这继续推动着我们前进。再后来，加 hilo、加 stall、根据助教的录屏敲代码，遇到 Bug 我们就通过请教同学和助教解决 Bug，然后顺利地通过了 64 号点，完成了任务。

通过此次实验我不仅学到了 CPU 的许多知识，还会运用 GitHub 来进行文件保存和交流，这对于以后的个人发展很有帮助，很幸运遇到很棒的组员，合作教会了我们一起交流成长；遇到了很棒的老师和助教，每次的课对于我们都是学习和收获，感谢您们的耐心教导，让我们受益多多，诚心祝福 CPU 实验越来越好！