Build CPU in 1 Month

Lecture 01

To-Do Lists

- Intro
  - Grammar(Verilog and SystemVerilog)
  - Architecture
  - Toolchain
- Hand on
  - Why our simulation don’t pass?
  - Add instruction
  - Run monitor program
- Course assignment
  - Advice for report

Grammar(Verilog and SystemVerilog)

- adder.v
- wire reg
- always @(*)
- always @(posedge clk)

- adder.sv
- logic
- always_comb
- always_ff @(posedge clk)
- support other data types, enum …
- especially for verification

SystemVerilog 是从 Verilog 扩展而来，也支持 Verilog 的一些语法，都属于经典的 HDL(**H**ardware **D**escription **L**anguage)

Grammar(Verilog and SystemVerilog)

- declare variable（声明变量）

```
logic     [31 : 0] a, b, c;
```

```
logic     [31 : 0] mem [1023 : 0];
```

- process block（过程块）
  - combinational logic

```
always_comb begin
    c = a + b;
end
```

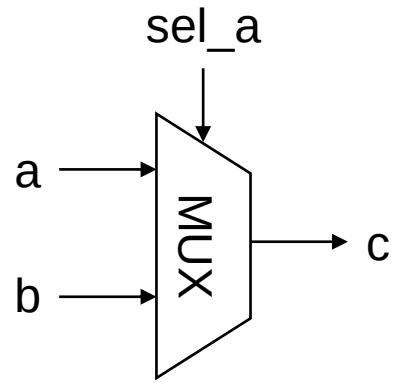
  - sequential logic

```
always_ff @(posedge clk) begin
    c <= a + b;
end
```

Grammar(Verilog and SystemVerilog)

- ternary expression（三元表达式）

```
logic                 a, b, c, sel_a;
always_comb begin
  c = sel_a ? a : b;
end
```

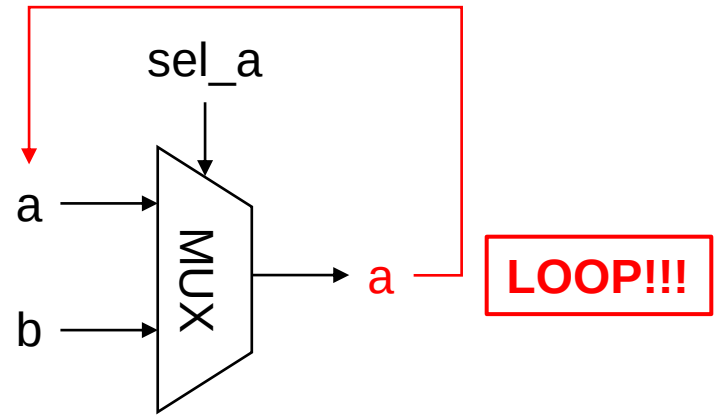


- if else

```
always_comb begin
  if (sel_a) c = a;
  else c       = b;
end
```

- case

```
always_comb begin
  case (sel_a)
    1'b1 : c = a;
    1'b0 : c = b;
    default : c = a;
  endcase
end
```

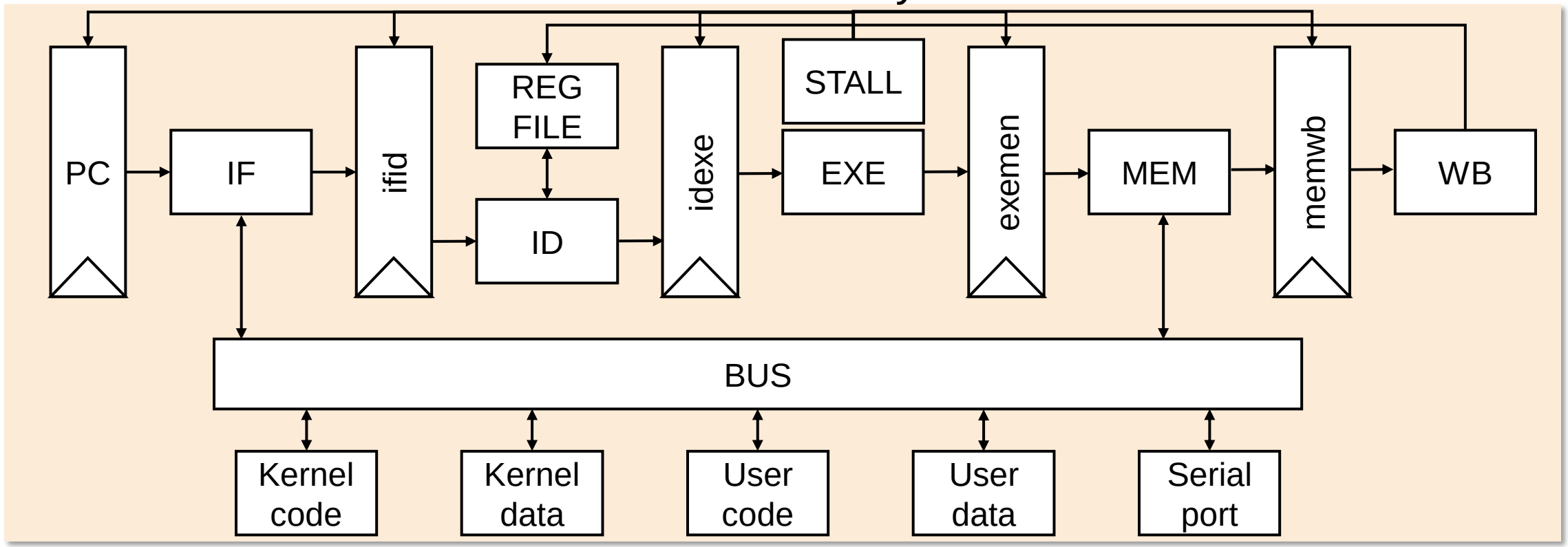


To-Do Lists

- Intro
  - Grammar(Verilog and SystemVerilog)
  - Architecture
  - Toolchain
- Hand on
  - Why our simulation don’t pass?
  - Add instruction
  - Run monitor program
- Course assignment
  - Advice for report

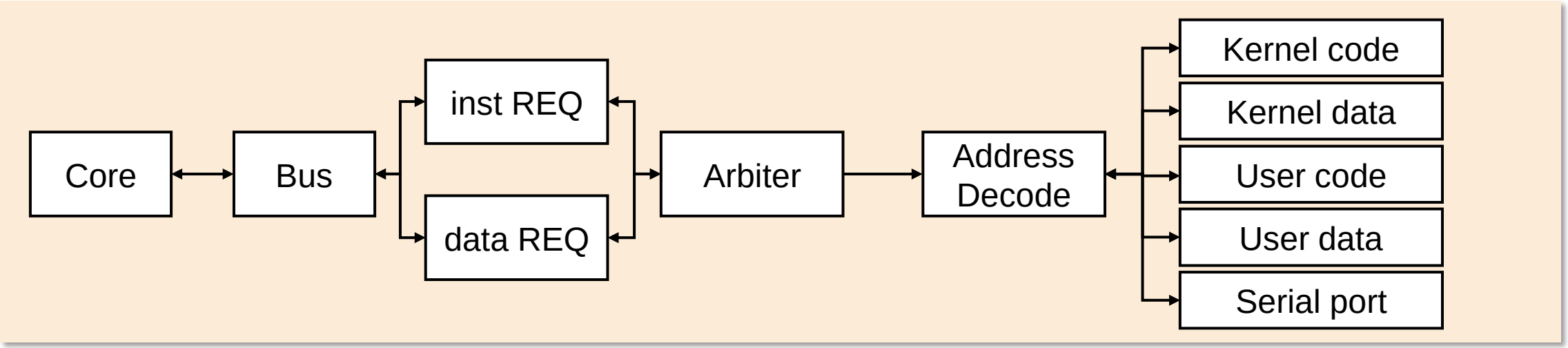
Architecture

- 5-stage pipeline CPU
  - classical micro-architecture and memory connected with bus



Architecture

- Bus（总线）
  - 定义：一组物理线和连接器的集合，计算机组件间交换数据常用的一种标准规范，处理器、内存、I/O传递信息的公用通道。总之，主机各部件通过总线连接，外设通过相应接口电路与总线连接
  - 基本功能：在多个设备间搭建一条数据通路，响应每个设备的请求



To-Do Lists

- Intro
  - Grammar(Verilog and SystemVerilog)
  - Architecture
  - Toolchain
- Hand on
  - Why our simulation don’t pass?
  - Add instruction
  - Run monitor program
- Course assignment
  - Advice for report

Toolchain

- Vivado 2019.2
  - General develop flow: Simulate HDL files, Synthesis, Implementation, Generate bitstream
- DVT Eclipse
  - get rid of Vivado useless integrated editor
- VS Code
  - a not bad choice to replace DVT but it takes some time to configure

Toolchain

This page tells you some unnecessary auxiliary tools

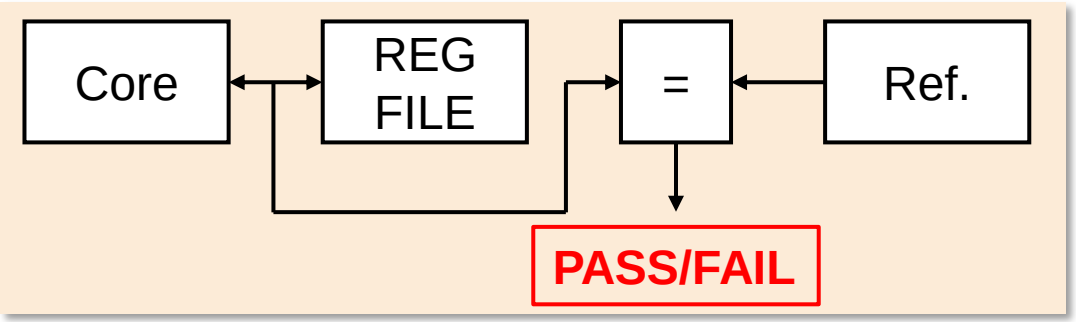
- MARS(**M**IPS **A**ssembly **R**untime **S**imulator)
  - run the code in single step
  - view all the registers' value in GUI
- QEMU(**Q**uick **EMU**lator)
  - run your own program in this fast, open source emulator
  - use this tool through command line
- MTI Bare Metal Toolchain --- compile the assembly code

To-Do Lists

- Intro
  - Grammar(Verilog and SystemVerilog)
  - Architecture
  - Toolchain
- Hand on
  - Why our simulation don’t pass?
  - Add instruction
  - Run monitor program
- Course assignment
  - Advice for report

Comparison Mechanism（比对机制）

- Q:
  - How to confirm that CPU runs correctly?
- A:
  - The aim of program is always to change the value of register(or memory).
  - Compare our CPU executing results with the GS132(reference).



Comparison Mechanism（比对机制）

- If our CPU's information is different from the reference, or when the CPU runs in a wrong loop, the simulation will stop

```
    if (  (debug_wb_pc!==ref_wb_pc) || (debug_wb_rf_wnum!==ref_wb_rf_wnum)
        ||(debug_wb_rf_wdata_v!==ref_wb_rf_wdata_v) )
    begin
        $display("--------------------------------------------------------------");
        $display("[%t] Error!!!",$time);
        $display("    reference: PC = 0x%8h, wb_rf_wnum = 0x%2h, wb_rf_wdata = 0x%8h",
                 ref_wb_pc, ref_wb_rf_wnum, ref_wb_rf_wdata_v);
        $display("    mycpu    : PC = 0x%8h, wb_rf_wnum = 0x%2h, wb_rf_wdata = 0x%8h",
                 debug_wb_pc, debug_wb_rf_wnum, debug_wb_rf_wdata_v);
        $display("--------------------------------------------------------------");
        debug_wb_err <= 1'b1;
        #40;
        $finish;
    end
end
else if(debug_wb_pc == 32'h80000008 || debug_wb_pc == 32'h8000000c) begin
        $display("--------------------------------------------------------------");
        $display("[%t] Error!!!",$time);
        $display("     CPU is in the wrong loop!!!");
        $display("--------------------------------------------------------------");
        debug_wb_err <= 1'b1;
        #40;
        $finish;
```

Run Behavioral Simulation

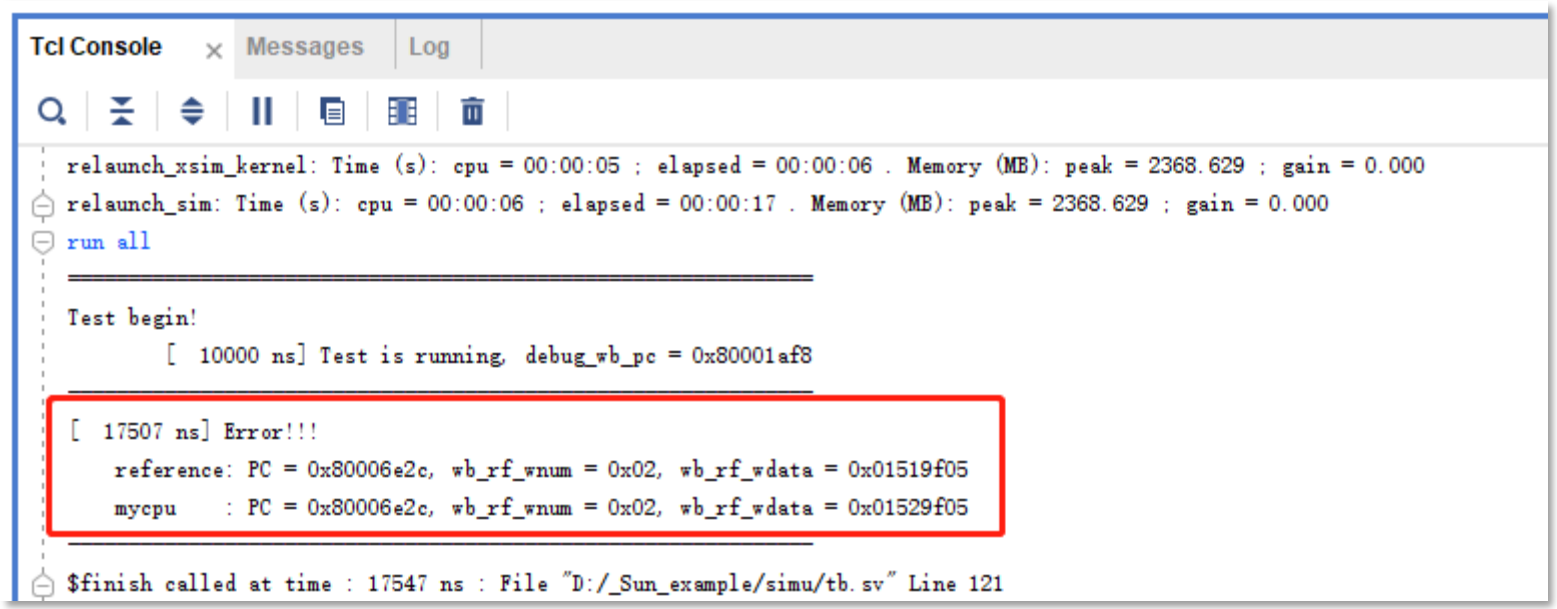
- Change the path of trace file in testbench

D:/_Sun_example/simu/tb.sv

```
1   `timescale 1ns / 1ps
2
3   `define TRACE_REF_FILE "D:/_Sun_example/simu/lab3_golden_trace.txt"
4   `define END_PC 32'h80000010
5
6   module tb_top( );
7   reg resetn;
8   reg clk_100M;
9   // reg clk_50M;
```

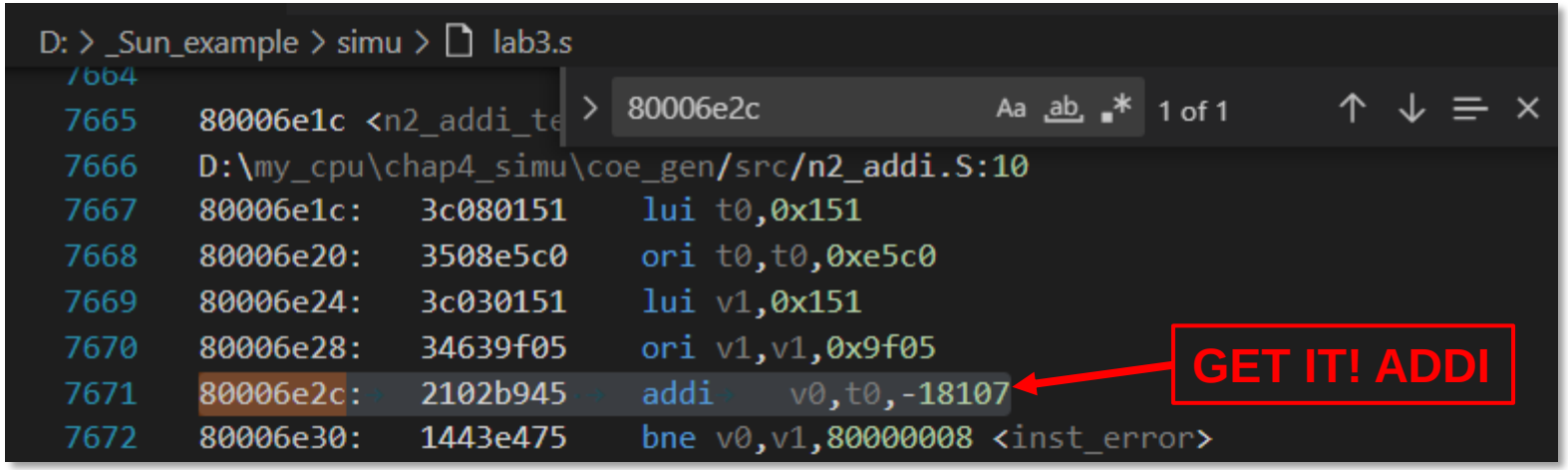
Filed To Run The Program

- Change the path of trace file in testbench



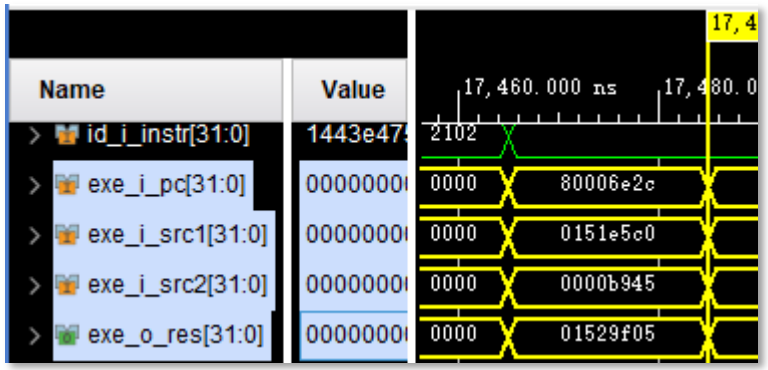
Why Our Simulation Don't Pass?

- Step 1: Check the lab3.s
  - a file translate .bin to assembly code with other information（反汇编）
  - an important file to find the bug!
- Step 2: Find the instruction according to PC

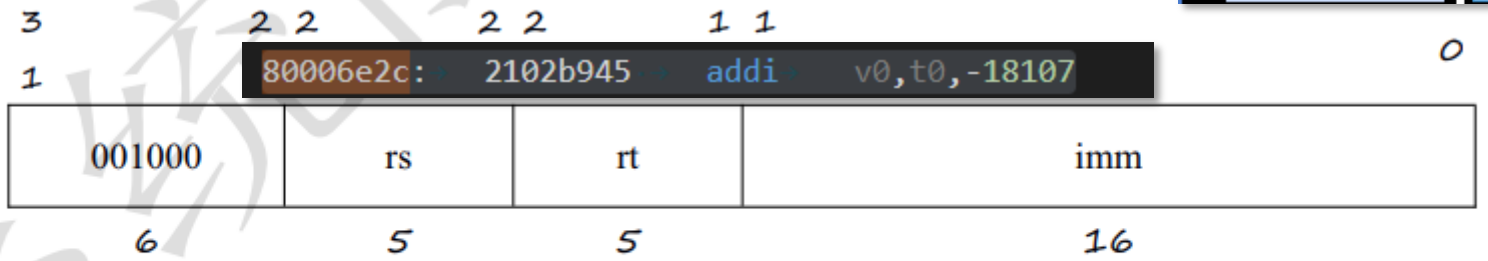


Why Our Simulation Don't Pass?

- Step 3: Check the instruction set definition
  - Find the operands in Vivado wave window



3.3.2 ADDI



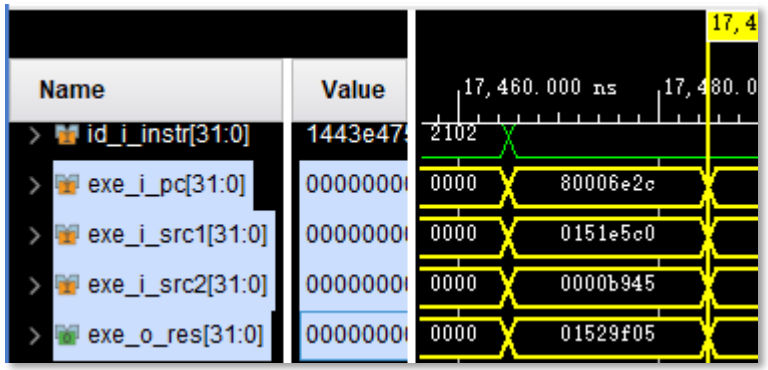
汇编格式： ADDI rt, rs, imm

功能描述： 将寄存器 rs 的值与有符号扩展至 32 位的立即数 imm 相加，结果写入 rt 寄存器中。如果产生溢出，则触发整型溢出例外（IntegerOverflow）。

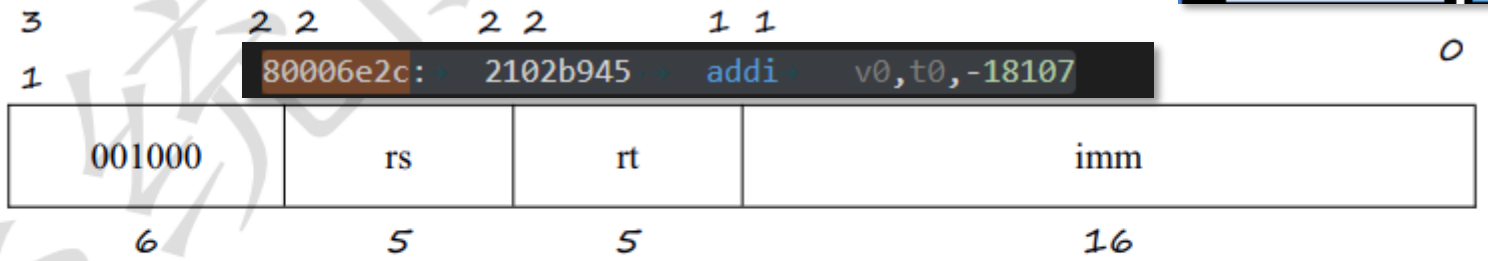
0151e5c0 + **0000b945** = **01529f05**

Why Our Simulation Don't Pass?

- Step 3: Check the instruction set definition
  - Find the operands in Vivado wave window



3.3.2 ADDI



汇编格式： ADDI rt, rs, imm

功能描述： 将寄存器 rs 的值与有符号扩展至 32 位的立即数 imm 相加，结果写入 rt 寄存器中。如果产生溢出，则触发整型溢出例外（IntegerOverflow）。

0151e5c0 + 0000b945 = 01529f05

1011_1001_0100_0101(H) = -18107(D)

Why Our Simulation Don't Pass?

- Step 4: Edit code
  - Where the exe_i_src2 is assigned?

```
always_comb begin
  src2_rt       = is_ADD || is_ADDU || is_SUB || is_SLT || is_MUL || is_AND ||
  src2_sign_imm = is_ADDIU || is_LB ||is_LW || is_SB || is_SW;
  src2_zero_imm = is_ANDI || is_XORI || is_ORI || is_ADDI;
  src2_imm      = is_LUI;
```

```
casez (1'b1)
  src2_rt : {immsel, uppersel, signext}       = 3'b000;
  src2_sign_imm : {immsel, uppersel, signext} = 3'b101;
  src2_zero_imm : {immsel, uppersel, signext} = 3'b100;
  src2_imm : {immsel, uppersel, signext}      = 3'b110;
  default : {immsel, uppersel, signext}       = 3'b000;
endcase
```

```
temp          = signext ? {{16{imm[15]}},imm} : uppersel ? (imm<<16) : {16'h0,imm};
id_o_src2     = immsel ? temp : pcsel ? id_i_pc_4 + 4 : re_rfrd2;
```

Run Simulation Again

- What is result? :)

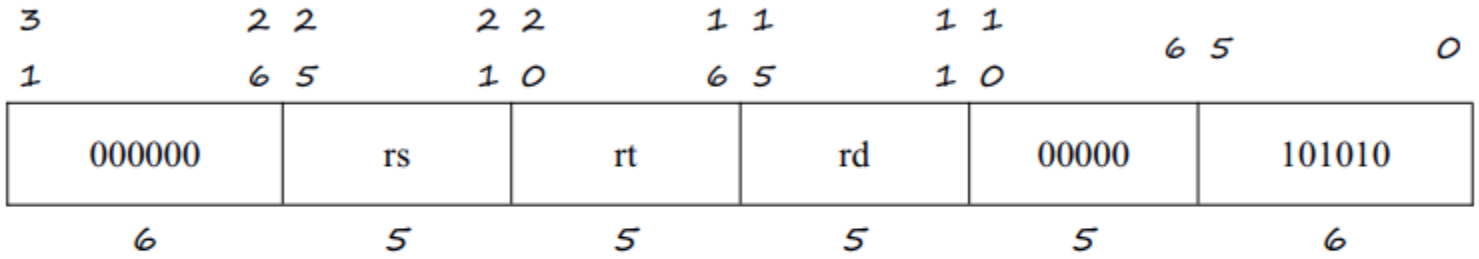
To-Do Lists

- Intro
  - Grammar(Verilog and SystemVerilog)
  - Architecture
  - Toolchain
- Hand on
  - Why our simulation don’t pass?
  - Add instruction
  - Run monitor program
- Course assignment
  - How to accomplish it

Add Instruction

- Consider SLT
  - Read the definition of this inst.

3.3.6 SLT



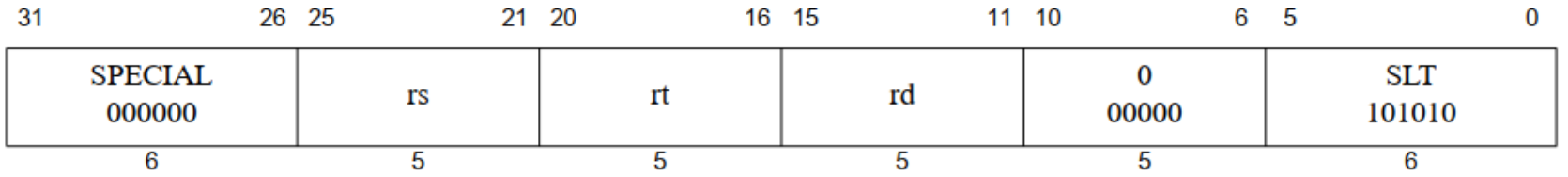
汇编格式： SLT rd, rt, rs

功能描述： 将寄存器 rs 的值与寄存器 rt 中的值进行有符号数比较，如果寄存器 rs 中的值小，则寄存器 rd 置 1；否则寄存器 rd 置 0。

Add Instruction

- Consider SLT
  - Read the definition of this inst.

Set on Less Than **SLT**



Format: `SLT rd, rs, rt` **MIPS32**

Purpose: Set on Less Than

To record the result of a less-than comparison

Add Instruction

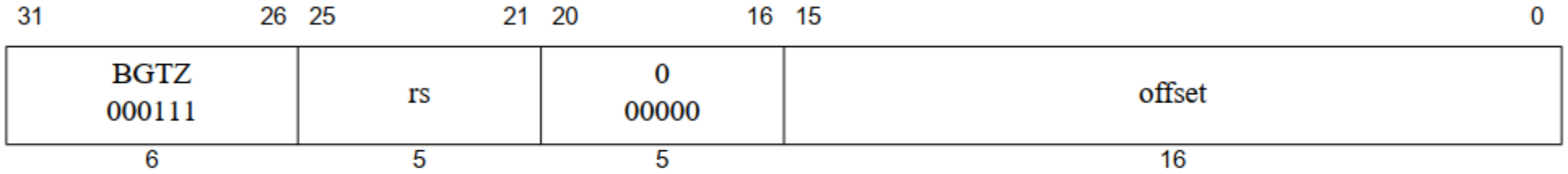
- Step 1: Decide the type of the inst.
- Step 2: Decide the regfile read address
- Step 3: Decide the data format of source operand
  - src1 is always $Rs, and needn't change format
  - src2 is from $Rt or immediate(original / 0 extend / signed extend)
- Step 4: Decide alutype
- Step 5: Decide aluop
- …

After Modification

- We have corrected ADDI and implemented SLT, it's fantastic!
  - So what about accomplish a CPU in 1 month?
- Run simulation again, see the result
  - does the CPU pass the test?

Boss In Lecture 01 --- BGTZ

Branch on Greater Than Zero **BGTZ**



Format: `BGTZ rs, offset` **MIPS32**

Purpose: Branch on Greater Than Zero

To test a GPR then do a PC-relative conditional branch

Description: `if GPR[rs] > 0 then branch`

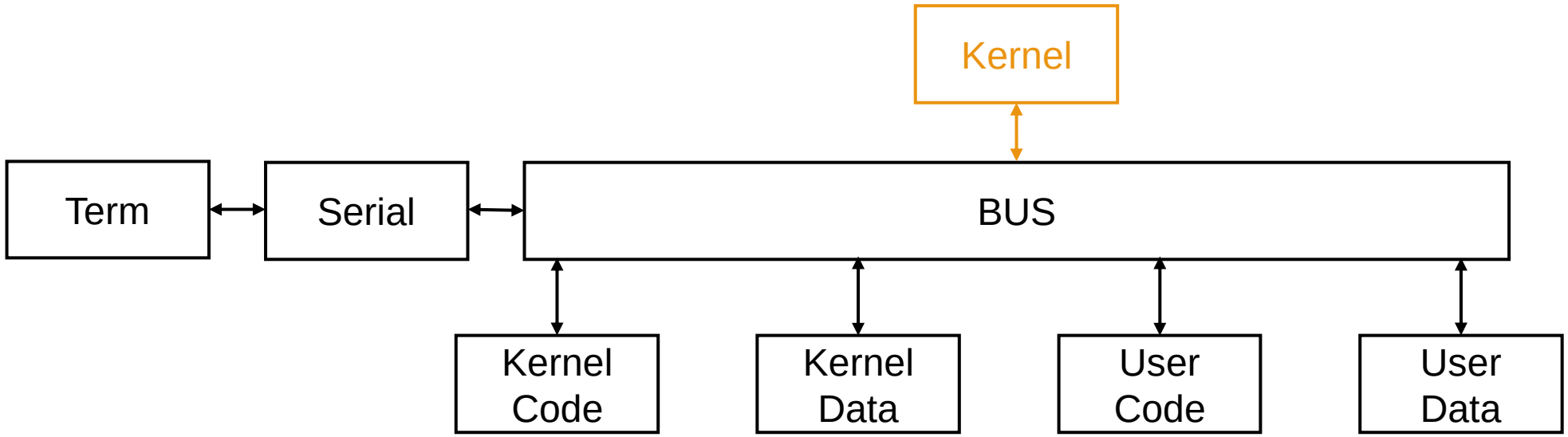
An 18-bit signed offset (the 16-bit *offset* field shifted left 2 bits) is added to the address of the instruction following the branch (not the branch itself), in the branch delay slot, to form a PC-relative effective target address.

If the contents of GPR *rs* are greater than zero (sign bit is 0 but value not zero), branch to the effective target address after the instruction in the delay slot is executed.

To-Do Lists

- Intro
  - Grammar(Verilog and SystemVerilog)
  - Architecture
  - Toolchain
- Hand on
  - Why our simulation don’t pass?
  - Add instruction
  - Run monitor program
- Course assignment
  - How to accomplish it

Run Monitor Program



- 监控程序的介绍及命令使用参考官方文档
- 监控程序的实现原理

Run Monitor Program

- To run the program, load all the memory in FPGA

虚地址空间	说明
0x80000000-0x800FFFFF	监控程序代码
0x80100000-0x80100FFF	用户代码空间
0x80400000-0x80400FFF	用户数据空间
0x807F0000-0x807FFFFF	监控程序数据
0xBFD003F8-0xBFD003FD	串口数据及状态

虚地址空间	说明
0x80000000-0x8000319B	监控程序代码
0x80100000-0x80100FFF	用户代码空间
0x80400000-0x80400FFF	用户数据空间
0x807F0000-0x807FFFFF	监控程序数据
0xBFD003F8-0xBFD003FD	串口数据及状态

To-Do Lists

- Intro
  - Grammar(Verilog and SystemVerilog)
  - Architecture
  - Toolchain
- Hand on
  - Why our simulation don’t pass?
  - Add instruction
  - Run monitor program
- Course assignment
  - Advice for report

Advice For Report

- 处理器研究的重要性
  - 芯片的研究意义
- 国内外研究现状
  - 通用处理器（RISC-V、香山、“一生一芯”、BOOM、玄铁…）
  - 图形处理器（GPU）
  - AI 处理器（寒武纪、晟腾…）
- 调研资料
  - 龙芯杯 wiki，往届优秀作品
  - 书籍
    - 《自己动手写 CPU》《计算机系统设计：基于FPGA的RISC处理器设计与实践》
    - 《计算机体系结构：量化研究方法》《计算机组成与设计：硬件/软件接口》
    - 《超标量处理设计》…

End & CU~

Lecture 01