**电子设计自动化**

**作业五**

**姓 名 黄智越**

**学 号 23S136049**

# 实验一 利用结构体，数组进行排序

## 实验目的及方案

**实验目的：**

1. 了解结构体，数组的基本操作。
2. 了解排序的基本操作

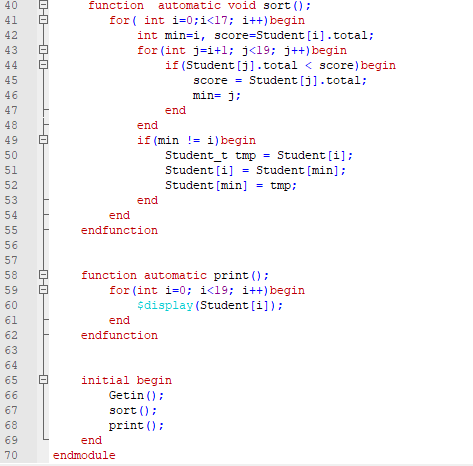
**验证方案**：

编译如下的代码

## 实验内容

1. 书写原代码





module score\_sort();

typedef struct {

string name;

string stu\_number;

int score[5:0];

int art\_tag; //Is a art OR science student?

int total;

}Student\_t;

int test[2:0] = {88,77,99};

Student\_t Student[20:0];

function automatic void Getin();

Student[0] = {"zhangsan", "20230001",{100,94,90,75,99,77}, 1,0};

Student[1] = {"lisi", "20230002",{60,68,50,60,51,80}, 0,0};

Student[2] = {"wangyu", "20230003",{91,78,88,87,99,50}, 1,0};

Student[3] = {"jack", "20230004",{50,78,67,60,80,77}, 0,0};

Student[4] = {"Rose", "20230005",{61,98,55,75,80,77}, 0,0};

Student[5] = {"Horse", "20230006",{50,87,70,75,89,77}, 1,0};

Student[6] = {"Judy", "20230007",{40,91,90,75,59,77}, 0,0};

Student[7] = {"Cristina", "20230008",{40,48,90,75,79,77}, 1,0};

Student[8] = {"Antonio", "20230009",{66,66,90,75,19,77}, 1,0};

Student[9] = {"MA\_YOYO", "20230010",{77,66,90,75,69,77}, 0,0};

Student[10] = {"Edward Yang", "20230011",{50,56,90,75,49,77}, 1,0};

Student[11] = {"Li an", "20230012",{60,87,90,75,69,77}, 1,0};

Student[12] = {"Zhangyimo", "20230013",{60,67,90,75,69,77}, 0,0};

Student[13] = {"ZHoudongyu", "20230014",{80,86,90,75,49,77}, 0,0};

Student[14] = {"Yangying", "20230015",{60,80,45,75,89,77}, 1,0};

Student[15] = {"Huangxiaoming", "20230016",{98,41,90,75,59,77}, 1,0};

Student[16] = {"Jianailiang", "20230017",{78,77,90,75,79,77}, 0,0};

Student[17] = {"Zhuneiliang", "20230018",{89,94,90,75,49,77}, 0,0};

Student[18] = {"Sheildon", "20230019",{100,100,100,100,99,99}, 1,0};

for(int i=0; i<19; i++)begin

for(int j=0; j<6; j++)begin

Student[i].total += Student[i].score[j];

end

end

endfunction

function automatic void sort();

for( int i=0;i<17; i++)begin

int min=i, score=Student[i].total;

for(int j=i+1; j<19; j++)begin

if(Student[j].total < score)begin

score = Student[j].total;

min= j;

end

end

if(min != i)begin

Student\_t tmp = Student[i];

Student[i] = Student[min];

Student[min] = tmp;

end

end

endfunction

function automatic print();

for(int i=0; i<19; i++)begin

$display(Student[i]);

end

endfunction

initial begin

Getin();

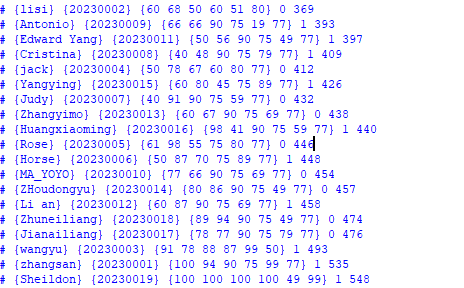
sort();

print();

end

endmodule

## 实验结果及分析



按升序对成员进行排序；

For也要用begin,end

Modelsim里display对字符串内容为中文的无法识别，打印出“??”

# 实验二 always@()的自锁现象与改进

## 实验目的及方案

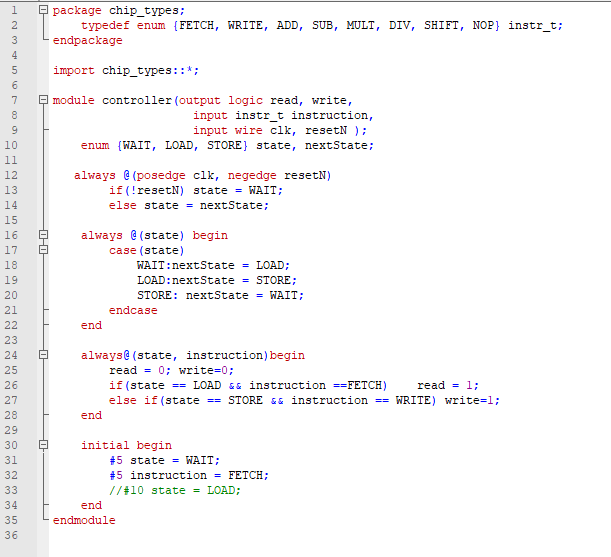
**实验目的：**

（1）always@()的自锁现象与使用always\_comb改进

**验证方案**：

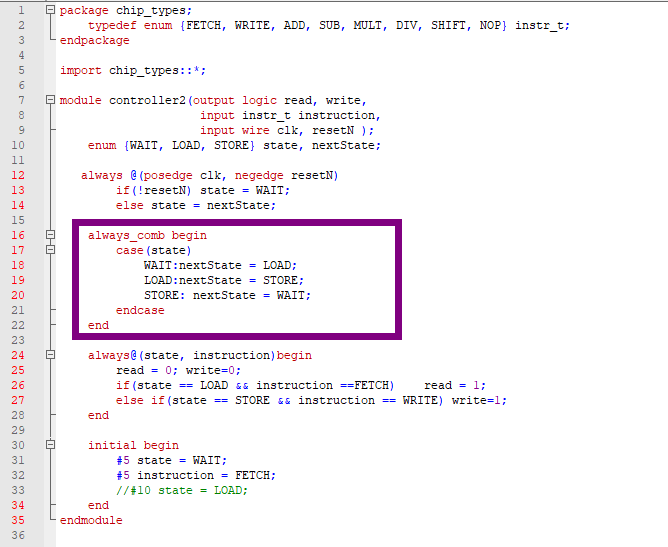
书写下列代码并编译

## 实验内容

a. 

发现always(clk, resetN), always(state)和always(state, instruction)过程块都不会被触发。

b.修改后



紫色方框部分用always\_comb替代，发现先运行always\_comb再运行initial块的内容。

## 三、实验结果及分析

a.中出现死锁是因为state默认类型为Int型，而其变量默认为0，即WAIT,所以我再给它赋值为WAIT,这说明信号没有变化，所有always(state)不能被触发；NextState默认也是WAIT，所以在时钟上升沿时state虽被赋值，但还是不变；就算增加reset信号也不能使其变化，因为reset后仍然给state赋值。

b.中always\_comb会在0时刻之前执行一次，所以nextState被赋值为LOAD；然后执行state=WAIT,如果此时外加reset信号就能变化。

# 实验三 判断==？与!=?

## 实验目的及方案

**实验目的：**

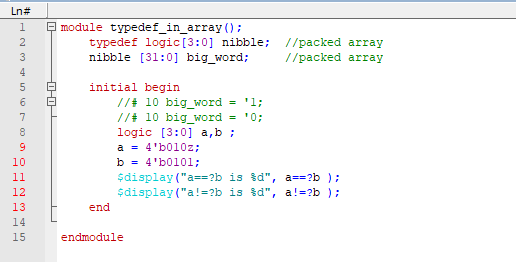
（1） 了解==？与!=?的比较结果

**验证方案**：

输入如下代码并编译

## 实验内容

代码如图所示：



结果：



## 实验结果及分析

!=? 和 ==? 的右边操作数单个的某位为数值时，而左边的操作数对应的位为x或z，此两种情况下表达式的值都为未知！

# 实验四 data[i+:4]用法

## 实验目的及方案

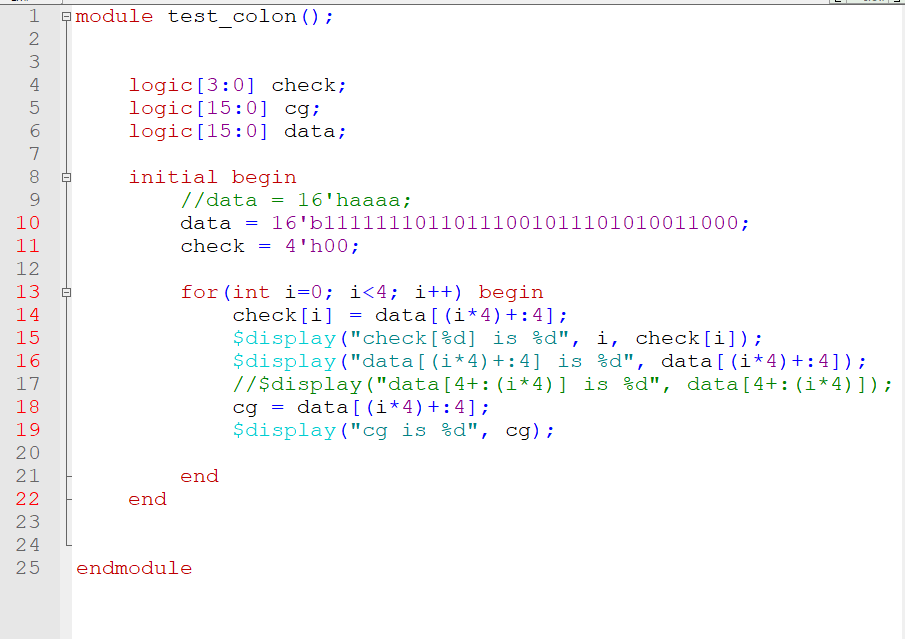
**实验目的：**

（1）熟悉data[i+:4]用法

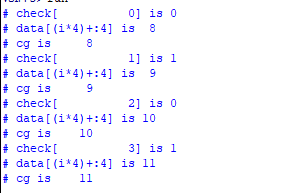
**验证方案**：

输入如下代码进行编译

## 实验内容



输出结果：



## 三、实验结果及分析

Data[i+:4] 代表从i为开始向高位取四位

# 实验四 含输入或输出的函数不能被使用的情况

## 实验目的及方案

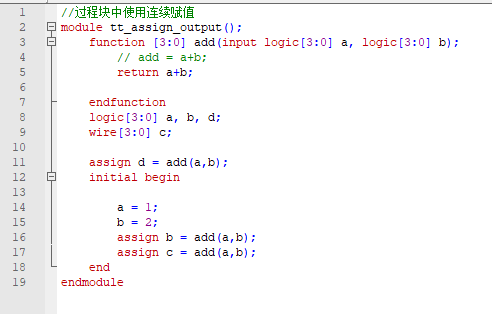
**实验目的：**

（1）验证含输入或输出的函数不能被使用的情况，如a.事件表达式；b.使用过程持续赋值的表达式；c.不在过程语句内的表达式

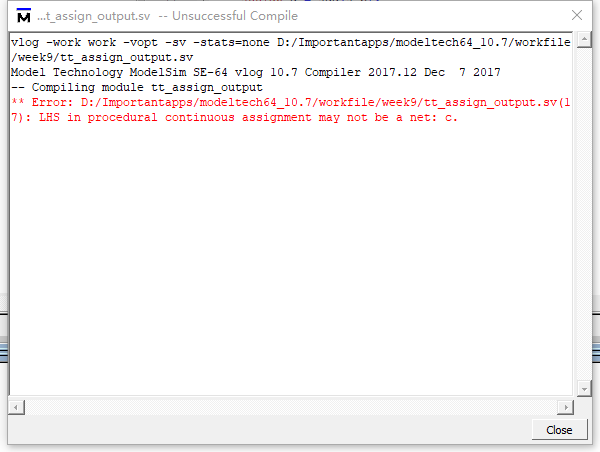
**验证方案**：

输入如下代码进行编译

## 实验内容



输出结果：



## 三、实验结果及分析

发现书上书写的条件不准确。经过实际验证发现：

b.使用过程持续赋值的左边变量为logic类型能行，但是换成wire类型不行；

c.不在过程语句内的表达式也能调用含输入或者输出的表达式

# 实验四 测试代码的等价性

## 实验目的及方案

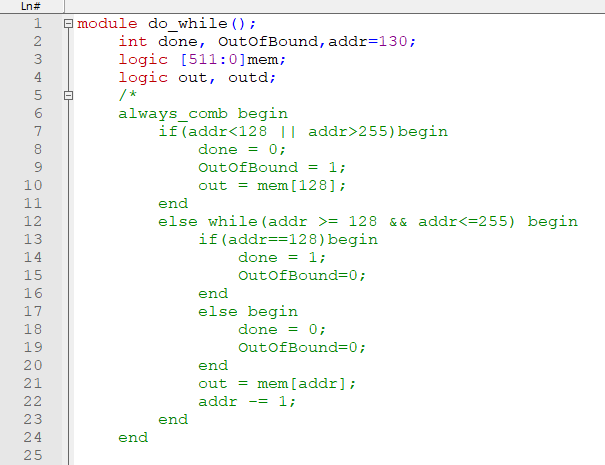
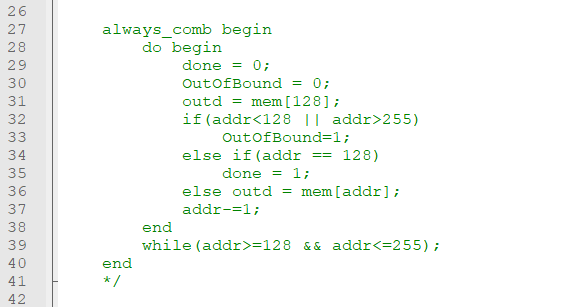
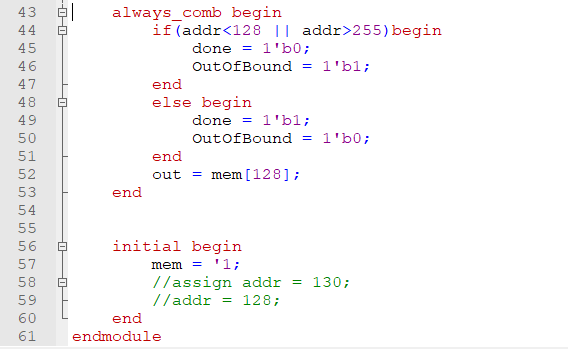
**实验目的：**

（1）测试代码的等价性

**验证方案**：

输入如下代码进行编译

## 实验内容

输出结果：

## 三、实验结果及分析

分别测试以上三个always块，发现结果都是取mem[128],第一个块和第二个宽的addr为127，第三个块没有改变addr;