流水线MIPS处理器设计

无73 欧阳良扬 2016011866

转发、阻塞与分支控制：

为了测试上面我们所提到的所有转发，阻塞与分支控制的正确性，我们设计如下汇编程序（stalltest.asm）：

.data

    in\_buff: .space 4096

.text

    la $a0,in\_buff

    addi $a1,$0,3

    sw $a1,0($a0)

    lw $a2,0($a0)

    addi $a3,$a2,1#load-use

    beq $a3,$a2,branch1#R and beq

    addi $t0,$t0,1

branch1:

    lw $a3,0($a0)

    beq $a3,$a2,branch2#lw and beq

    addi $t1,$t1,1

branch2:

    jal jump1

jump1:

    bne $a1,$ra,branch3#jal and beq

    addi $t2,$t2,1

    beq $a1,$a2,branch4

    addi $t3,$t3,1

    sw $a1,0($a0)

    lw $a2,0($a0)

    jr $a2#lw and jr

branch3:

    addi $a1,$ra,0

    jr $a1#R and jr

branch4:

    jal jump2

    addi $t4,$t4,1

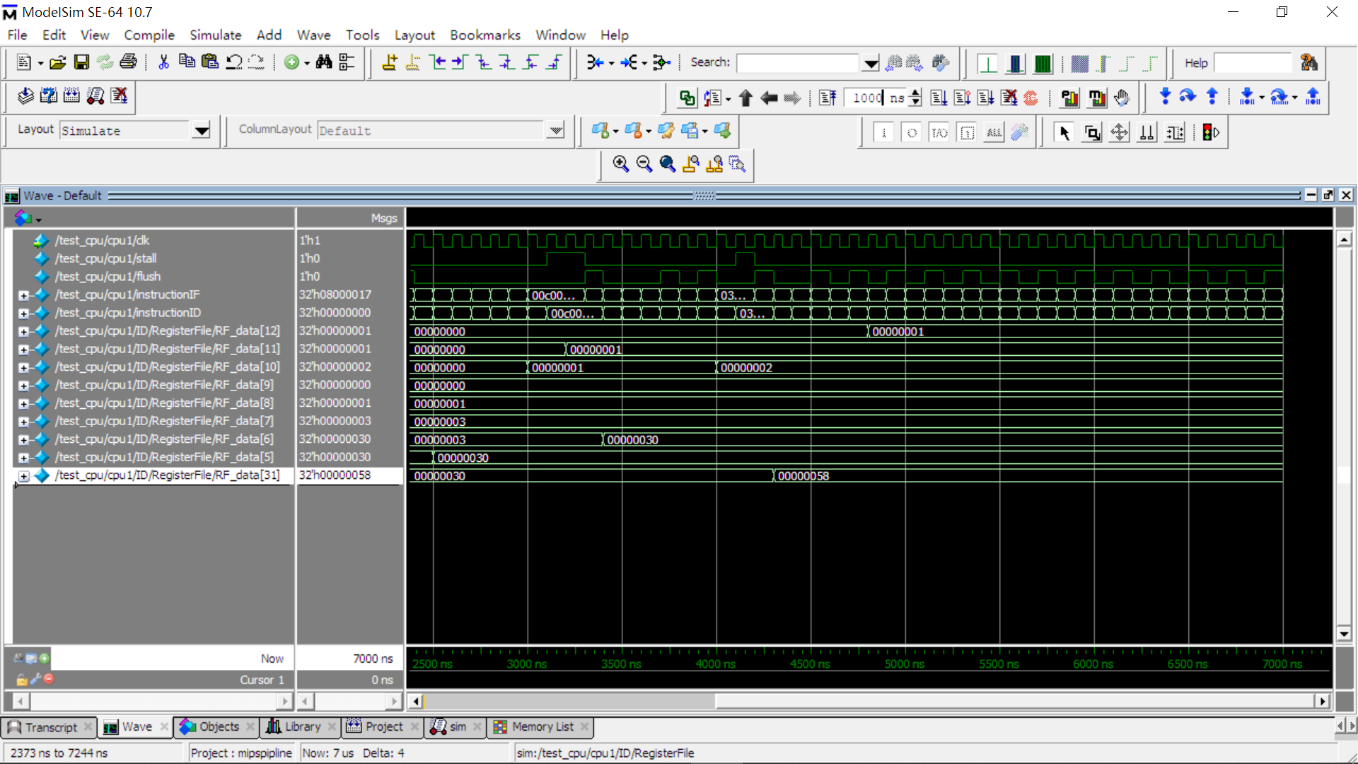
loop:

    j loop

jump2:

    jr $ra#jal and jr

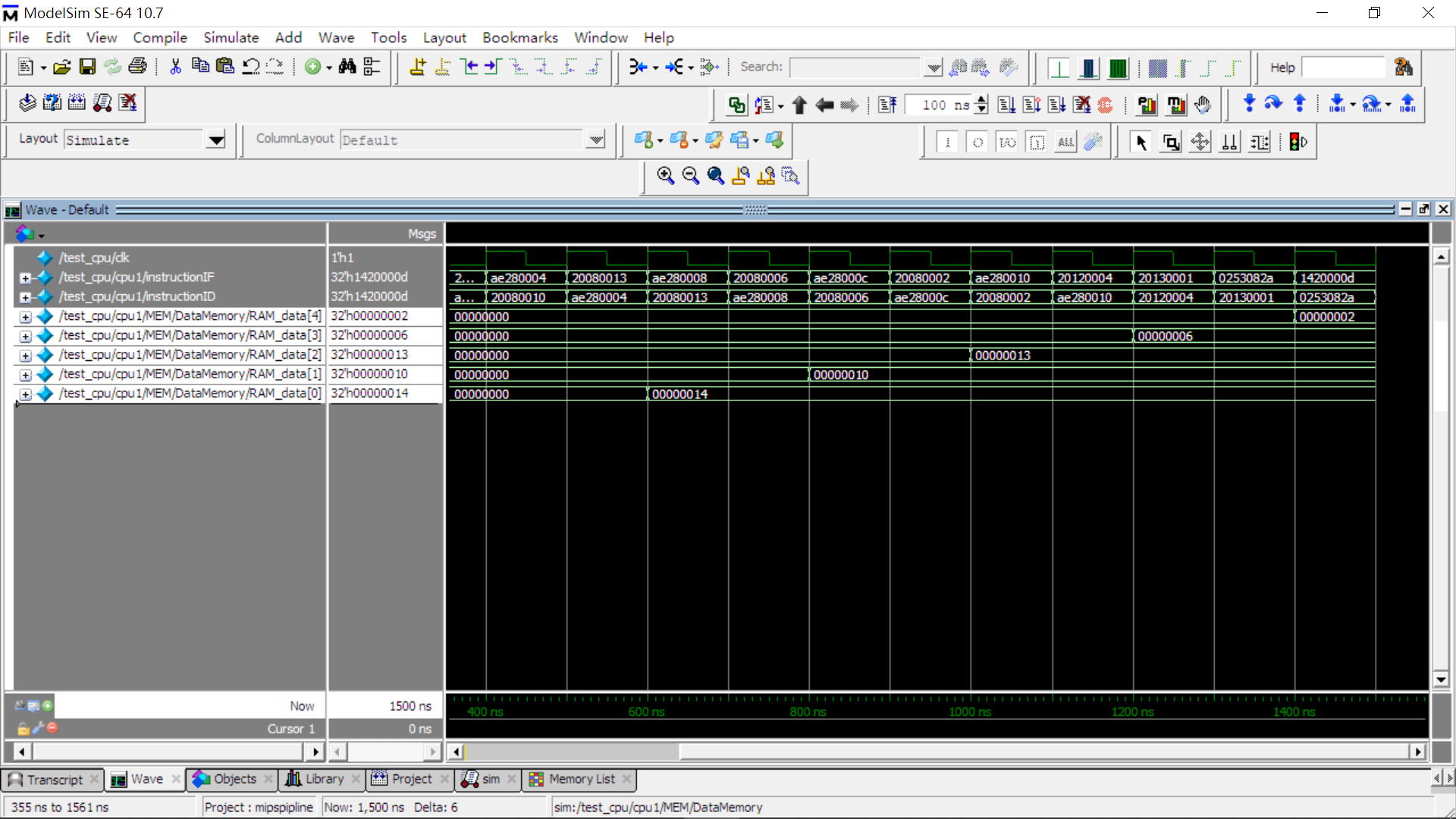
上述代码包含了所有的stall和flush情形，我们用仿真软件对这段代码的机器码（InstructionMemoryTest.v）进行了测试。为了便于观察程序运行结果，我们令$t0-$t4作为计数器统计程序是否正确地执行了各个分支。我们设计的流水线MIPS处理器得到的结果与MARS仿真结果一致如下图：



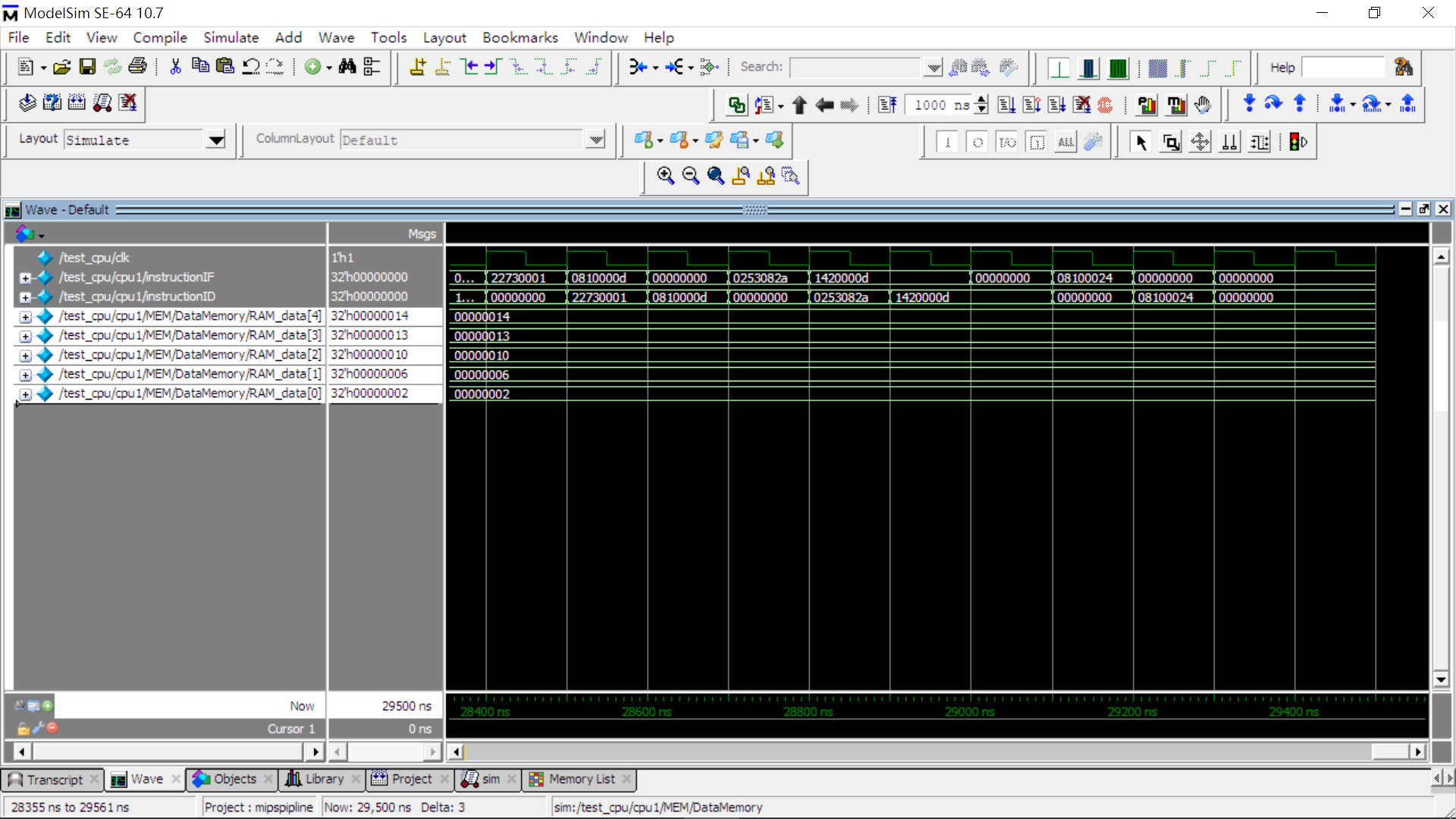
图片包含 屏幕截图

描述已自动生成

至此，我们有信心让我们的处理器完成一段复杂的、包含多个分支与跳转的排序程序。我们采用lw指令直接写入随机数以便于仿真（见sort.asm与InstructionMemorySort.v）。在仿真中我们展示5个数的排序如下图：

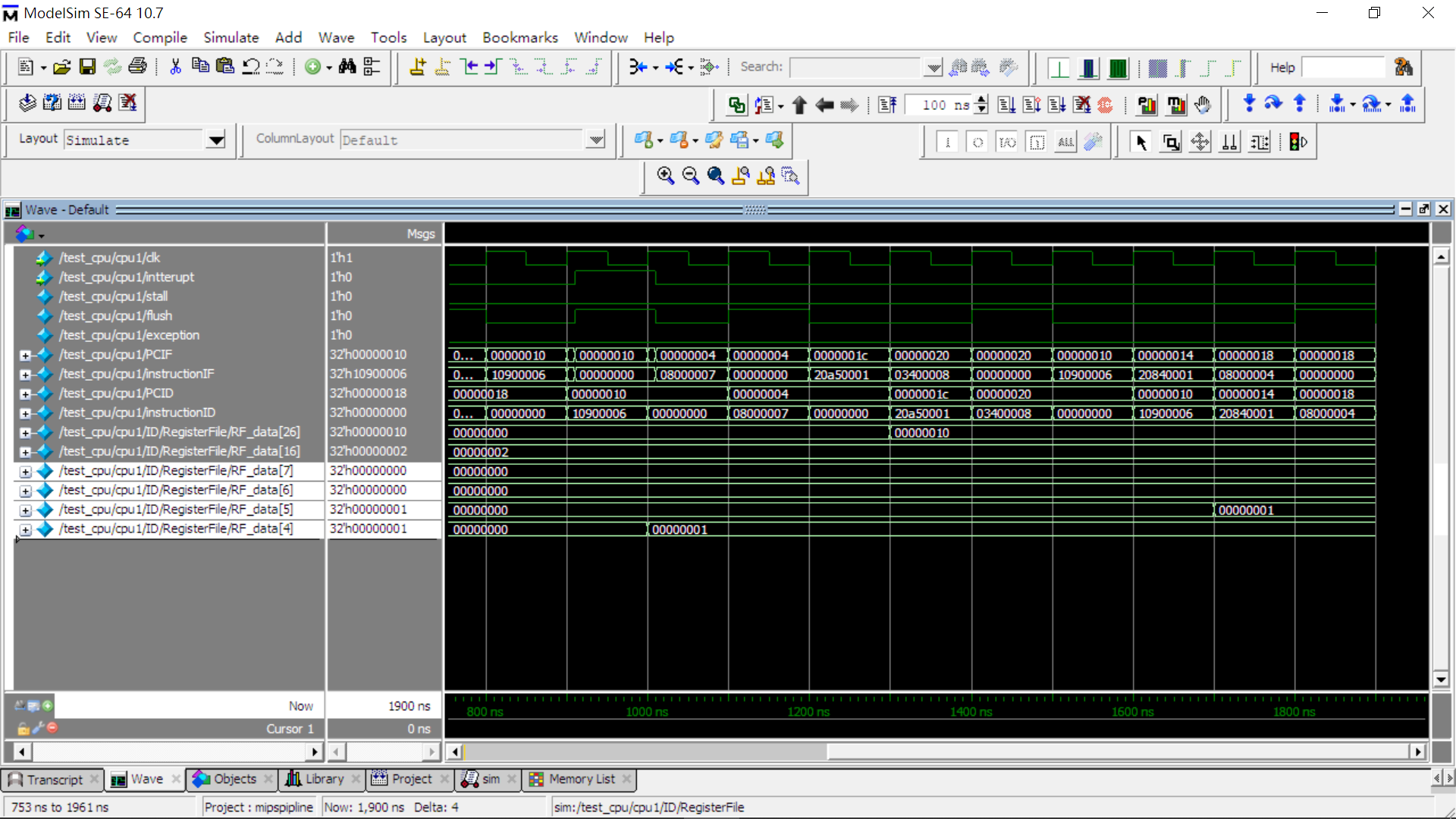


5个数据写到RAM中

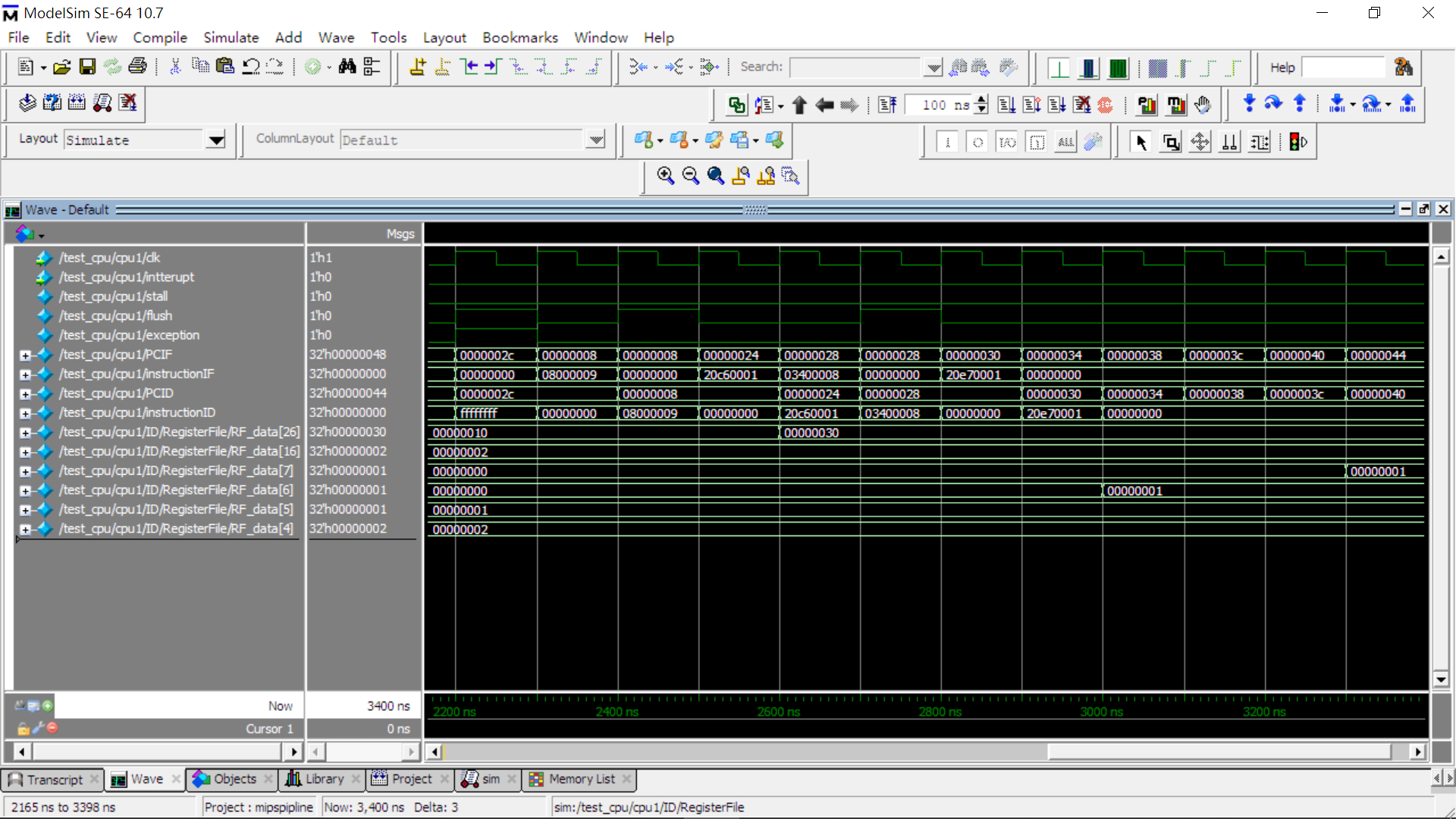


经过约280个周期排序完成

中断与异常处理：



中断



未知指令异常及最终结果

在实际验证时我们采用实验三中使用过的串口来进行数据的读写。当数据接收完毕后处理器将自动开始工作，当程序运行结束后LED将全部亮起，此时按下S4将排序后的数据发出。