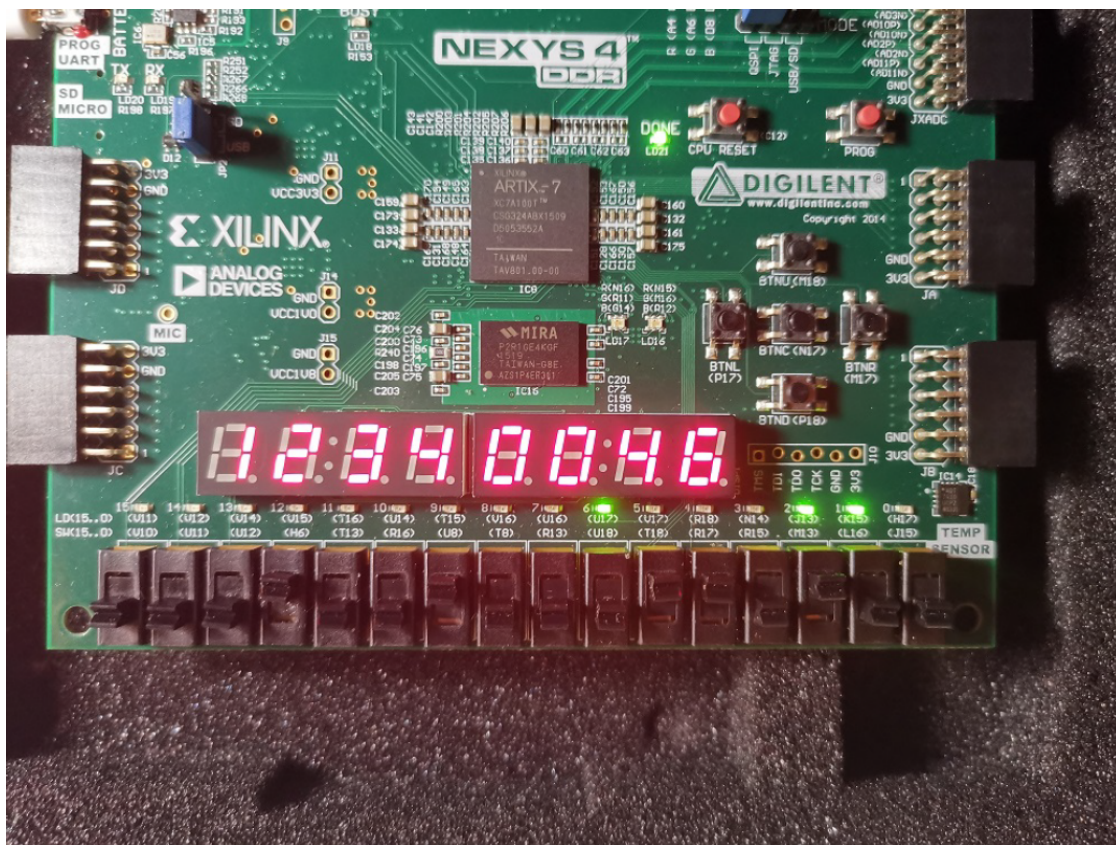











# 计算机体系结构实验 申A报告

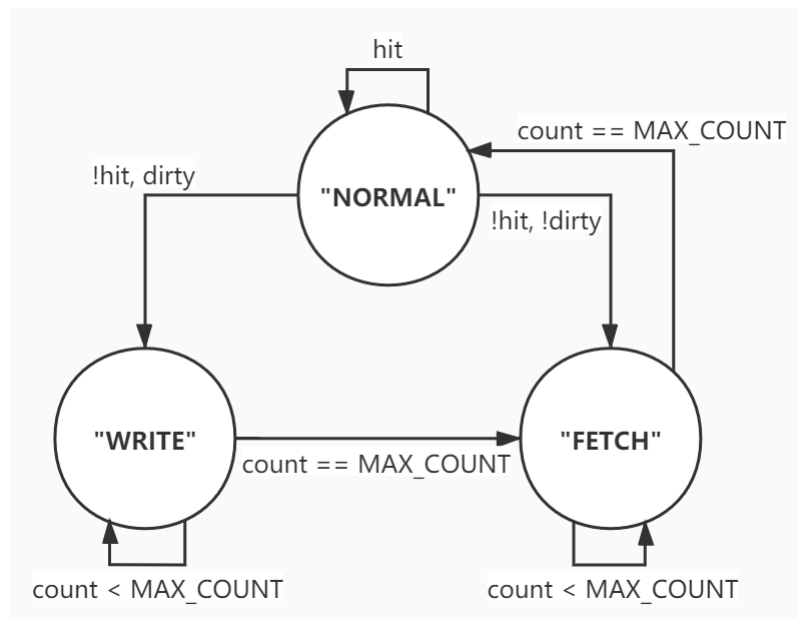
1. 受疫情影响，完成了上板验证。（见：单周期和流水线提交包中的实验录像）



2. 在课程开始前就已主动完成了课程要求的基本内容（除cache外），包括选做部分（流水线）。  
（见：github上的commit记录）

<b>Initial commit for cache</b>  jasha64 committed on 7 May
<b>Fix bug of instr slti</b>  jasha64 committed on 27 Apr
<b>Add instr sll, srl, sra, sllv, srlv, srav</b>  jasha64 committed on 3 Apr
<b>Add instr j, jr, jal, jalr</b>  jasha64 committed on 2 Apr
<b>Solved control hazard. Full hazard handling. Add instr beq, bne.</b>  jasha64 committed on 9 Mar
<b>Add stalls</b>  jasha64 committed on 8 Mar
<b>Add data forwarding. Add instr addi, andi, ori, slti, nop.</b>  jasha64 committed on 6 Mar
<b>Initial pipeline w/o hazard unit. Supporting instr R, lw, sw</b>  jasha64 committed on 5 Mar
<b>Rename files, prepared for pipeline</b>  jasha64 committed on 2 Mar

3. 在课程要求的基础上更进一步，完成了带Cache的MIPS CPU。



4. 自主设计了多笔有特点的仿真测试测资。（见：流水线实验报告和提交的文件）

```
mutual recursion.in - 记事本
文件(E) 编辑(E) 格式(O) 查看(V) 帮助(H)
# 在编译原理的语法分析中，很多文法都是递归形式给出的。
# 假设有如下文法：
# G[F]:
# F → fG | a
# G → gF
# 判断字符串 a 是否满足的程序如下：
#
# int F(int pos)
# {
#   if(a[pos] == 'a') return pos+1;
#   if(a[pos] == 'f') return G(pos+1);
#   return -1;
# }
#
# int G(int pos)
# {
#   if(a[pos] == 'g') return F(pos+1);
#   return -1;
# }
# b = F(0);
```

5. 独立发现并解决了书上代码的一处bug和老师提供的模板中的一处问题。（见：流水线实验报告）



6. 按照拔尖班的要求，新增多条指令。

```

149 6' b000000: case(funcnt) // RTYPE
150     6' b001000: controls <= 12' b0_0_0_0_0_0_0_0_1_000; // JR
151     6' b001001: controls <= 12' b1_0_0_0_0_0_0_0_1_000; // JAL
152     6' b000000: controls <= 12' b1_1_0_0_0_0_0_0_110; // SLL
153     6' b000010: controls <= 12' b1_1_1_0_0_0_0_0_110; // SRL
154     6' b000011: controls <= 12' b1_1_1_0_0_0_0_0_110; // SRA
155     default:    controls <= 12' b1_1_0_0_0_0_0_0_110; // other RTYPE instrs
156 endcase
157 6' b100011: controls <= 12' b1_0_0_1_0_0_0_1_0_000; // LW
158 6' b101011: controls <= 12' b0_0_0_1_0_0_0_1_0_000; // SW
159 6' b000100: controls <= 12' b0_0_0_0_0_1_0_0_0_001; // BEQ
160 6' b000101: controls <= 12' b0_0_0_0_0_1_0_0_0_001; // BNE
161 6' b001000: controls <= 12' b1_0_0_1_0_0_0_0_0_000; // ADDI
162 6' b001100: controls <= 12' b1_0_0_1_1_0_0_0_0_010; // ANDI
163 6' b001101: controls <= 12' b1_0_0_1_1_0_0_0_0_011; // ORI
164 6' b001010: controls <= 12' b1_0_0_1_0_0_0_0_0_100; // SLTI
165 6' b000010: controls <= 12' b0_0_0_0_0_0_0_0_1_000; // J
166 6' b000011: controls <= 12' b1_0_0_0_0_0_0_0_0_1_000; // JAL

```

7. 电路图全部自己画，从而对MIPS CPU结构有了更透彻的理解。（见：单周期、流水线实验报告）

