**计算机组成原理**

**课程设计报告**

**学 号\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_**

**姓 名\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_**

**指导教师\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_**

**提交日期\_\_\_\_\_\_\_\_\_\_\_2020.7\_\_\_\_\_\_\_\_\_\_\_**

**成绩评价表**

|  |  |  |
| --- | --- | --- |
| **报告内容** | **报告结构** | **报告最终成绩** |
| **□丰富正确**  **□基本正确**  **□有一些问题**  **□问题很大** | **□完全符合要求**  **□基本符合要求**  **□有比较多的缺陷**  **□完全不符合要求** |  |
| **报告与Project功能一致性** | **报告图表** | **总体评价** |
| **□完全一致**  **□基本一致**  **□基本不一致** | **□符合规范**  **□基本符合规范**  **□有一些错误**  **□完全不正确** |  |

**教师签字:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

目录

**Project11**

一、总体数据通路结构设计图 3

二、模块描述 3

三、指令描述 8

四、测试程序 13

五、modelsim测试 15

六、收获体会 16

**Project217**

一、总体数据通路结构设计图 17

二、模块描述 17

三、指令描述 24

四、测试程序 29

五、modelsim测试 31

六、收获体会 35

**Project336**

一、总体数据通路结构设计图 36

二、模块描述 36

三、指令描述 43

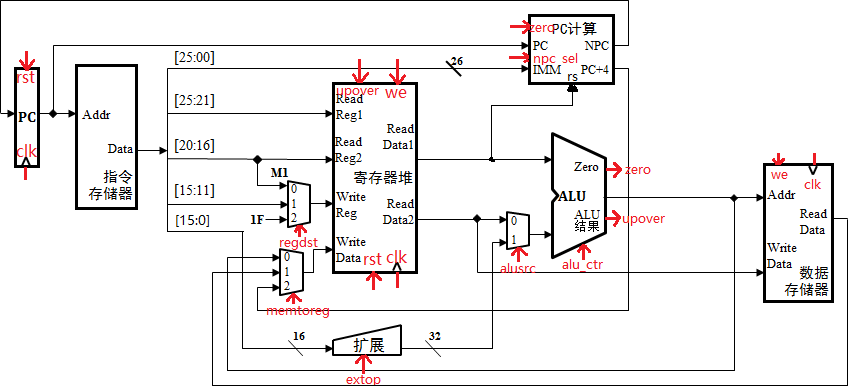
四、测试程序 44

五、modelsim测试 45

六、收获体会 47

**Project1**

一、总体数据通路结构设计图



二、模块描述

1. ctrl（控制模块）

（1）基本描述

控制模块主要功能是根据指令中func码字段和opcode码字段分析出当 前指令类型，进而给出不同的控制信号。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| instr[31:0] | I | 接收当前指令 |
| alu\_ctr  [1:0] | O | alu功能选择信号  00：ALU执行加法  01：ALU执行减法  10：ALU执行或运算  11：a>b输出0;a<b输出1 |
| ext\_op  [1:0] | O | ext功能选择信号  00：0扩展  01：符号扩展  10：低位补零扩展 |
| npc\_sel  [1:0] | O | 当前指令识别信号  00：一般指令  01：beq指令  10：j指令  11：jr指令 |
| memwrite | O | dm写入控制信号  0：不写入  1：写入 |
| regweite | O | gpr(寄存器组)写入控制信号  0：不写入  1：写入 |
| memtoreg  [1:0] | O | gpr写入数据选择信号  00：写入ALU运算后的结果  01：写入DM提供的内容  10:写入PC+4的结果 |
| alusrc | O | alu的inputB选择信号  0：ReadData2中的数据传入ALU  1：扩展后的32位立即数写入ALU |
| regdst  [1:0] | O | gpr的writereg写入数据选择信号  00：rt写入writereg  01：rd写入writereg  10：31写入writereg |

（3）功能定义

根据指令中func码字段和opcode码字段分析出当前指令类型，进而给 出不同的控制信号。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | alusrc | regwrite | memwrite | memtoreg | regdst | extop | npc\_sel | alu\_ctr |
| addu | 0 | 1 | 0 | 00 | 01 | 01 | 00 | 00 |
| subu | 0 | 1 | 0 | 00 | 01 | 01 | 00 | 01 |
| ori | 1 | 1 | 0 | 00 | 00 | 00 | 00 | 10 |
| lw | 1 | 1 | 0 | 01 | 00 | 01 | 00 | 00 |
| sw | 1 | 0 | 1 | 00 | 00 | 01 | 00 | 00 |
| beq | 0 | 0 | 0 | 00 | 00 | 01 | 01 | 01 |
| lui | 1 | 1 | 0 | 00 | 00 | 10 | 00 | 10 |
| j | 1 | 0 | 0 | 00 | 00 | 01 | 10 | 00 |
| addi | 1 | 1 | 0 | 00 | 00 | 01 | 00 | 00 |
| addiu | 1 | 1 | 0 | 00 | 00 | 01 | 00 | 00 |
| slt | 0 | 1 | 0 | 00 | 01 | 01 | 00 | 11 |
| jal | 1 | 1 | 0 | 10 | 10 | 01 | 10 | 00 |
| jr | 0 | 0 | 0 | 00 | 00 | 01 | 11 | 00 |
| jalr | 1 | 1 | 0 | 10 | 01 | 01 | 11 | 00 |

2、im

（1）基本描述

im的功能是储存指令，根据不同的输入地址，输出相应的指令。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| addr  [11:0] | I | 当前指令的地址（低12位） |
| dout  [31:0] | O | 输出32位指令 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 输出指令 | 根据不同的输入地址，输出相应的指令 |

3、pc

（1）基本描述

pc储存当前指令的地址，第一条指令地址为0x0000\_3000

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号  0：无效  1：有效 |
| npc[31:0] | I | 从npc模块接收下一条指令地址 |
| pc[31:] | O | 输出当前指令地址 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 复位 | 当复位信号有效时，PC被设置为0x0000\_3000。 |
| 2 | 取指令 | 根据npc,输出指令 |

4、npc

（1）基本描述

根据控制信号npc\_sel、zero，确定下一条指令的地址。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| npc\_sel  [1:0] | I | 判断当前指令  00：一般指令  01：beq指令  10：j指令、jal指令  11：jr指令 |
| zero | I | ALU计算结果为0标志。  0：计算结果非0  1：计算结果为0 |
| imme  [25:0] | I | 从指令寄存器中获得部分跳转地址 |
| pc[31:0] | I | 从pc中获得部分跳转地址 |
| rs[31:0] | I | 从alu结果中获得跳转地址 |
| npc[31:0] | O | 下一条指令地址 |
| pc\_4 | O | pc+4的地址 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 一般跳转 | npc=pc+4 |
| 2 | beq跳转 | npc= pc\_4+{{14{offset[15]}},offset,2'b0} |
| 3 | j跳转 | npc= {pc[31:28],imme,2'b0} |
| 4 | jr跳转 | npc=rs |

5、alu

（1）基本描述

运算器，根据ALUctr给出的不同信号，调用adder/substract/ori/slt子模块对 接收到的两个32位数据加/减/或/比较运算，将结果输出。

（2）模块端口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| a | I | 被操作数 |
| b | I | 操作数 |
| alu\_ctr  [1:0] | I | 功能选择信号  00：加运算  01：减运算  10：或运算  11：比较运算 |
| output  [31:0] | O | 运算结果 |
| zero | O | 零标志位  0：output不为0  1：output为0 |
| upover | O | 加法溢出标志  0：不溢出  1：溢出 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 加 | a+b |
| 2 | 减 | a-b |
| 3 | 或 | a|b |
| 4 | 比较 | a>b:0;a<b:1 |

6、gpr

（1）基本描述

GPR包含32个寄存器，有两个读端口，一个写端口，一个写控制信号，两个 读寄存器地址信息，一个写寄存器地址信息，复位信号，溢出信号和时钟信号。 两个读端口可以随时读出当前地址对应的寄存器内容，写端口写入数据需要 RegWrite信号为1。寄存器号的输入都是5位的，数据线为32位。

（2）模块端口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| we  (regwrite) | I | 写入控制信号  0：不写入  1：写入 |
| rst | I | 复位信号  0：不复位  1：复位 |
| upover | I | 加法溢出信号  0：不溢出  1：溢出 |
| readreg1  [4:0] | I | 读寄存器地址1 |
| readreg2  [4:0] | I | 读寄存器地址2 |
| writereg  [4:0] | I | 写寄存器地址 |
| writedata  [31:0] | I | 写寄存器内容 |
| readdata1  [31:0] | O | 读寄存器内容1 |
| readdata2  [31:0] | O | 读寄存器内容2 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 读寄存器 | 根据readreg1、readreg2地址读取寄存器内容到readdata1、readdata2 |
| 2 | 写寄存器 | 根据writeref地址将writedata数据写入寄存器 |
| 3 | 溢出置零 | 加法溢出时，第30号寄存器第0位置1 |
| 4 | 复位 | 28号寄存器置0000\_1800  29号寄存器置0000\_2ffc  其余置0 |

7、dm

（1）基本描述

DM为内存单元，有一个地址输入端口，一个写入数据端口，一个读出数据端 口和一个写控制信号。写控制信号控制将数据写入对应地址或者将数据从对应 地址读出。

（2）模块端口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| we | I | 写入控制信号  0：不写入  1：写入 |
| addr[11:0] | I | 内存地址 |
| din[31:0] | I | 写内存内容 |
| dout[31:0] | O | 读内存内容 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 读内存 | 根据addr地址读取内存内容到dout |
| 2 | 写内存 | 根据addr地址将din数据写入内存 |

8、ext

（1）基本描述

ext为扩展器，有一个16位数据写入端口，一个控制信号和一个32位 数据输出端口。根据功能选择信号的不同将输入数据进行0扩展/符号扩展/低 位补0扩展。

（2）模块端口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| din[15:0] | I | 输入16位数据 |
| ext\_op  [1:0] | I | 功能选择信号  00：0扩展  01：符号扩展  10：低位补零扩展 |
| dout[31:0] | O | 输出32位数据 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 0扩展 | 高16位补0，低16位为input数据 |
| 2 | 符号扩展 | 高16位符号扩展，低16位为input数据 |
| 3 | 低位补0扩展 | 高16位补为input数据，低16为补0 |

9、mux

（1）基本描述

多选一选择器，内有32位3选1，31位2选1，5位3选1三个模块

（2）模块端口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| a0 | I | 数据1 |
| a1 | I | 数据2 |
| a2 | I | 数据3 |
| ch | I | 数据选择信号 |
| out | O | 数据输出 |

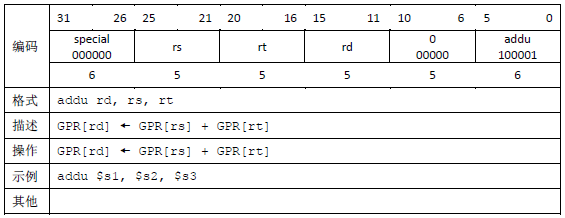
（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 数据选择 | 根据ch，从若干个输入数据中选择一个输出 |

三、指令描述

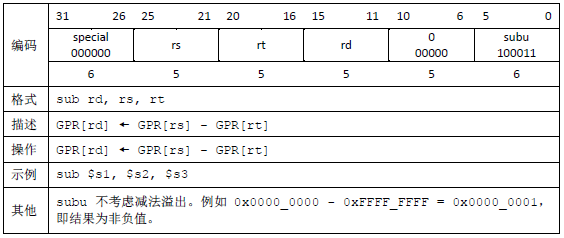
1. addu

功能：无符号加



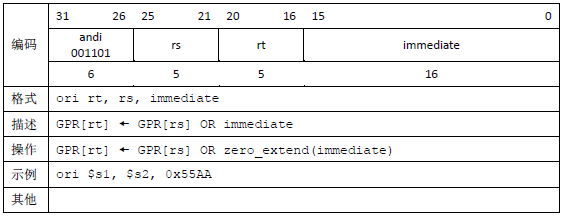
2、subu

功能：无符号减



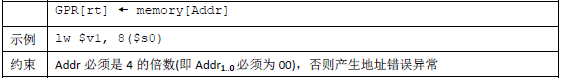
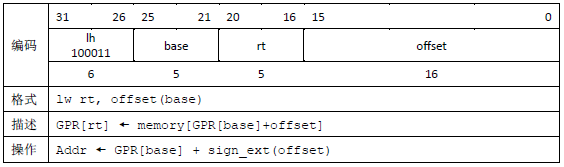
3、ori

功能：或立即数



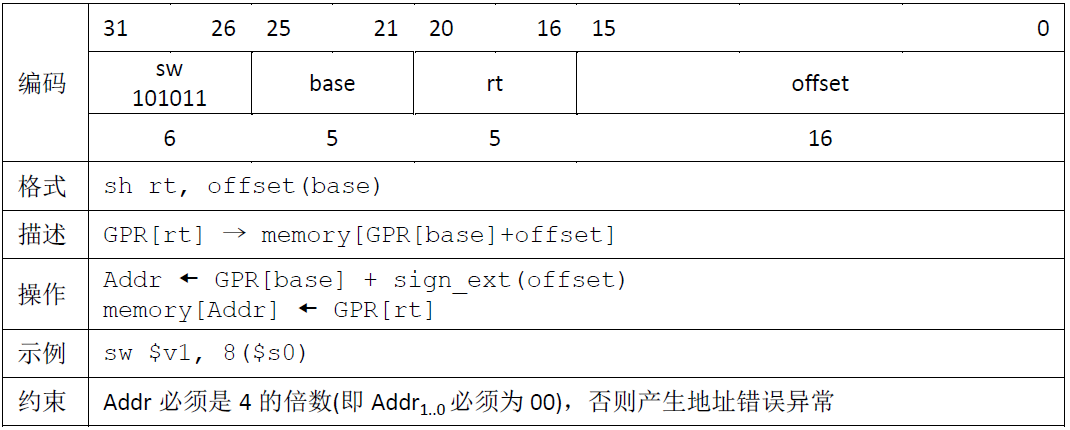
4、lw

功能：加载字



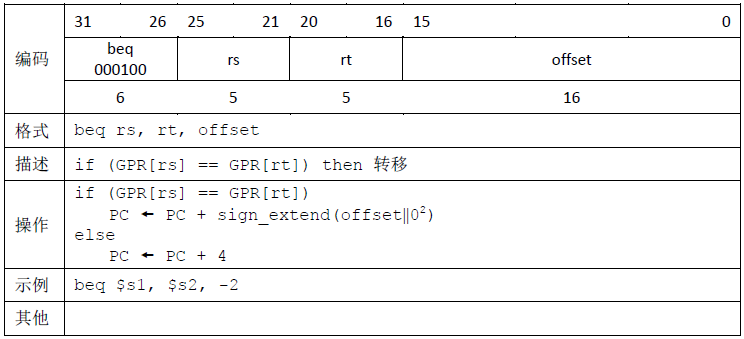
5、sw

功能：储存字



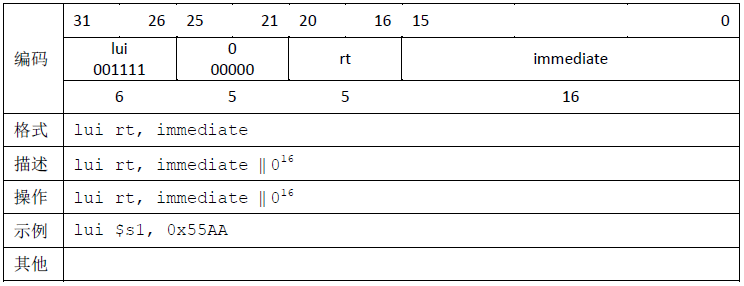
6、beq

功能：相等时转移



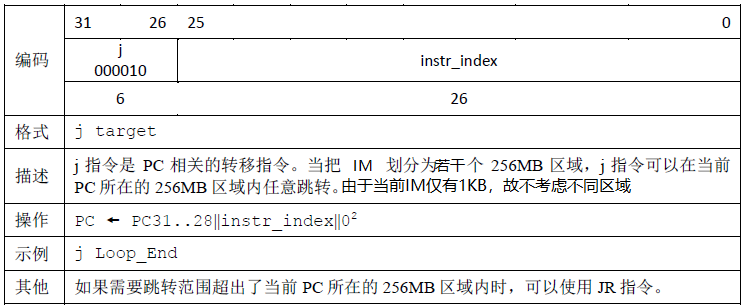
7、lui

功能：立即数加载至高位，低位补0



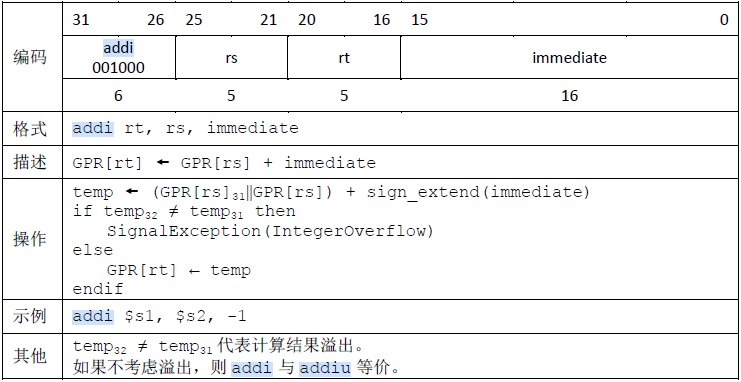
8、j

功能：跳转



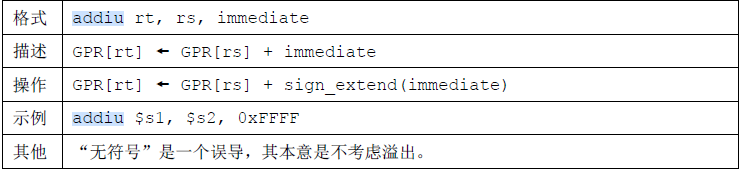
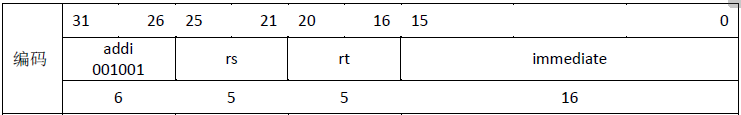
9、addi

功能：符号加立即数



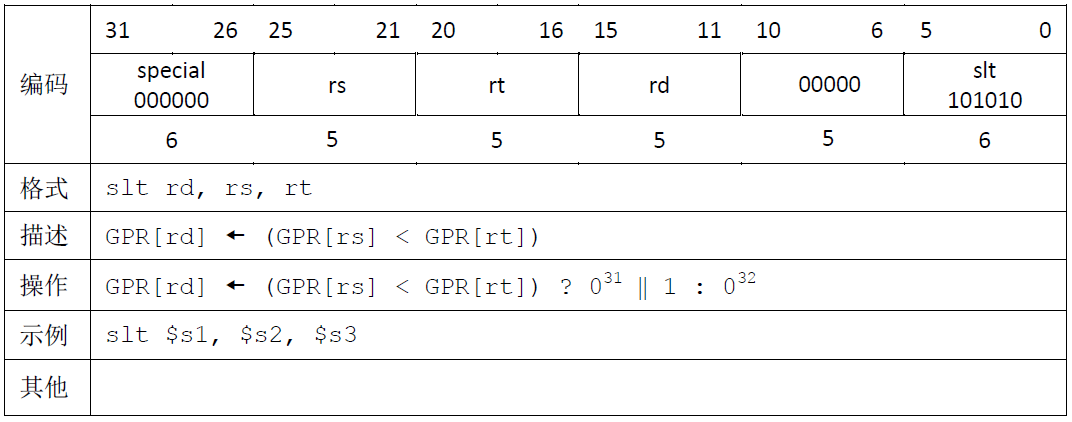
10、addiu

功能：无符号加立即数



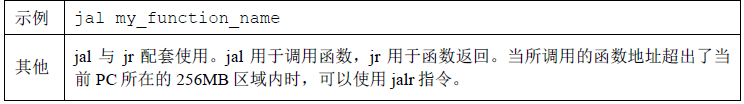
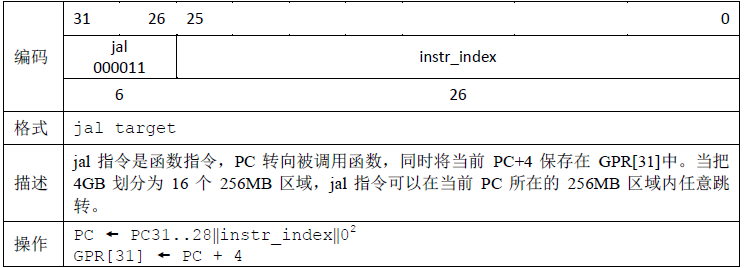
11、slt

功能：小于立即数置1



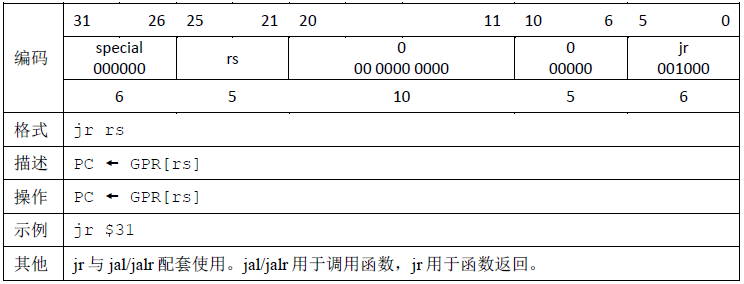
12、jal

功能：跳转并链接



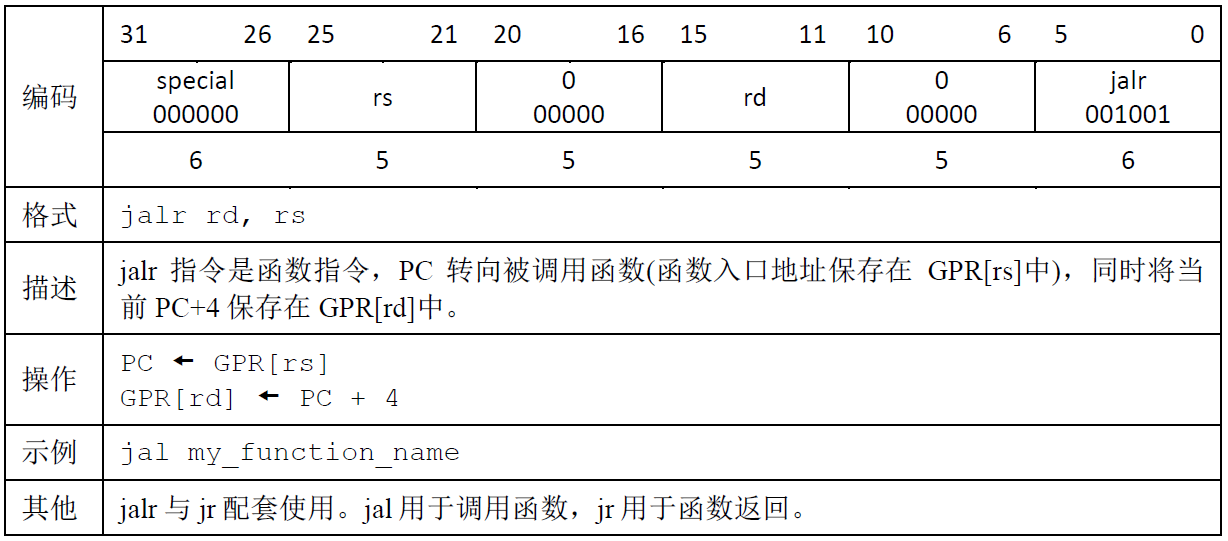
13、jr

功能：跳转至寄存器



14、jalr

功能：跳转并链接



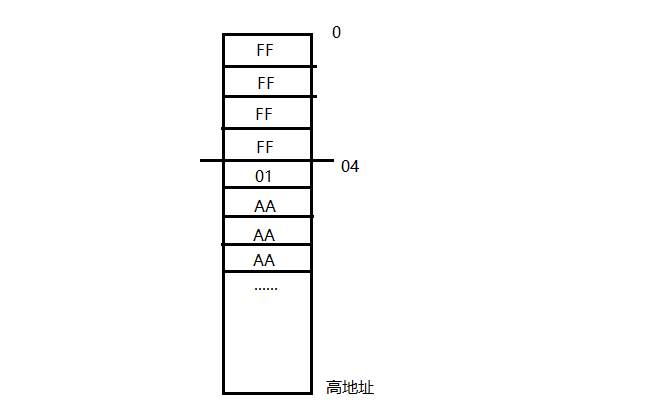
四、测试程序

1、测试前数据准备

（1）寄存器组数据

$s0=0x0000 0010

$s1=0x0000 0001

 $s2=0x0000 1100

$s3=0x0000 0FF2

$s4=0x0000 0004

$s5=0x7FFF FFFE

（2）内存单元数据（字地址）

00：0xFFFF FFFF

04：0XAAAA AA01

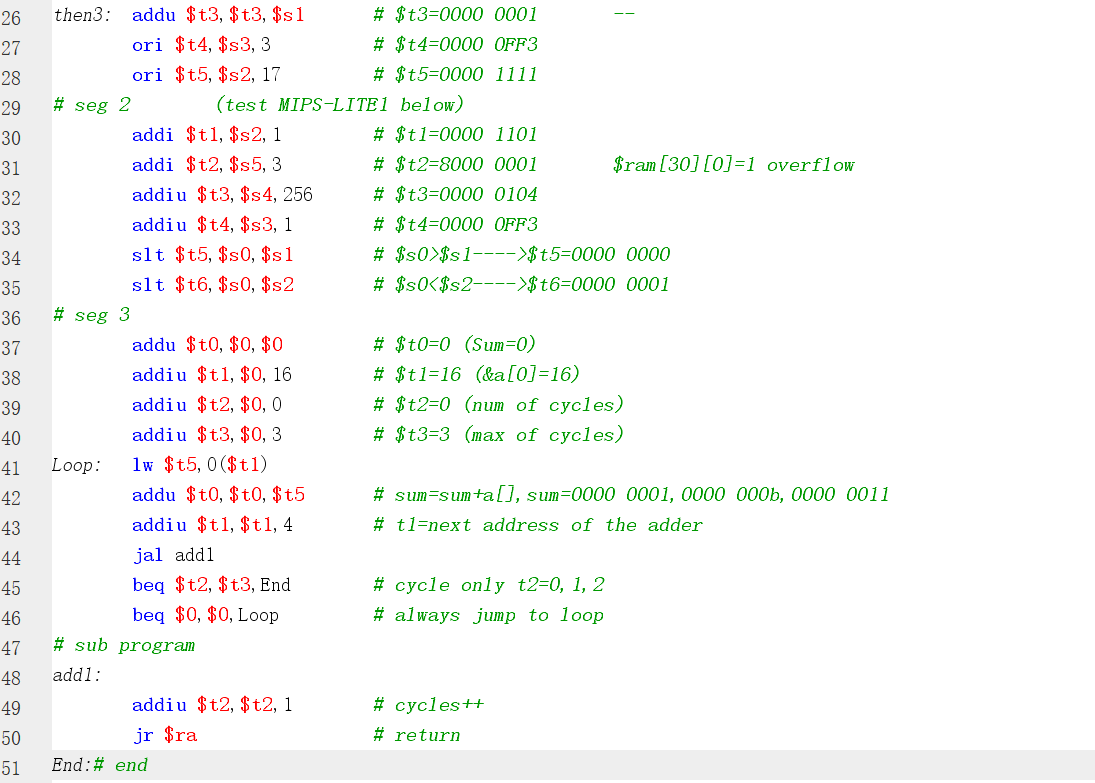
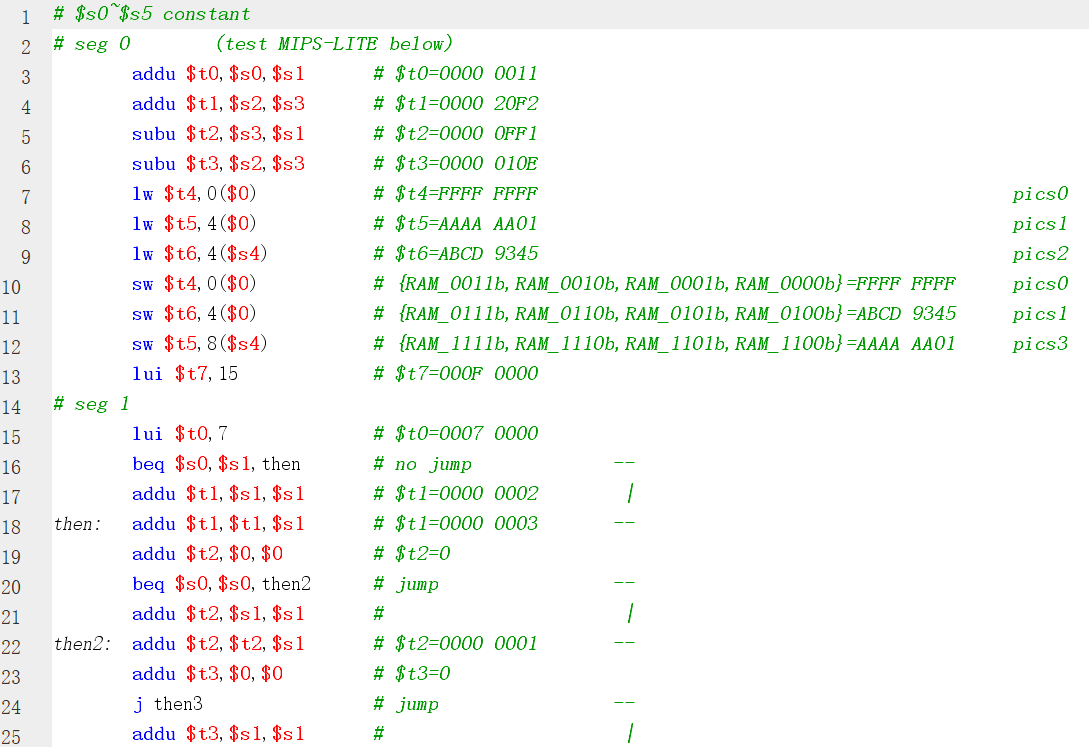
08：0XABCD 9345

16: 0X0000 0001

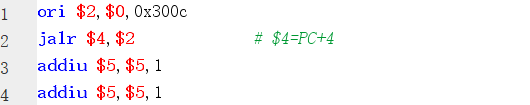
20: 0X0000 000A

24: 0X0000 0006

2、测试程序

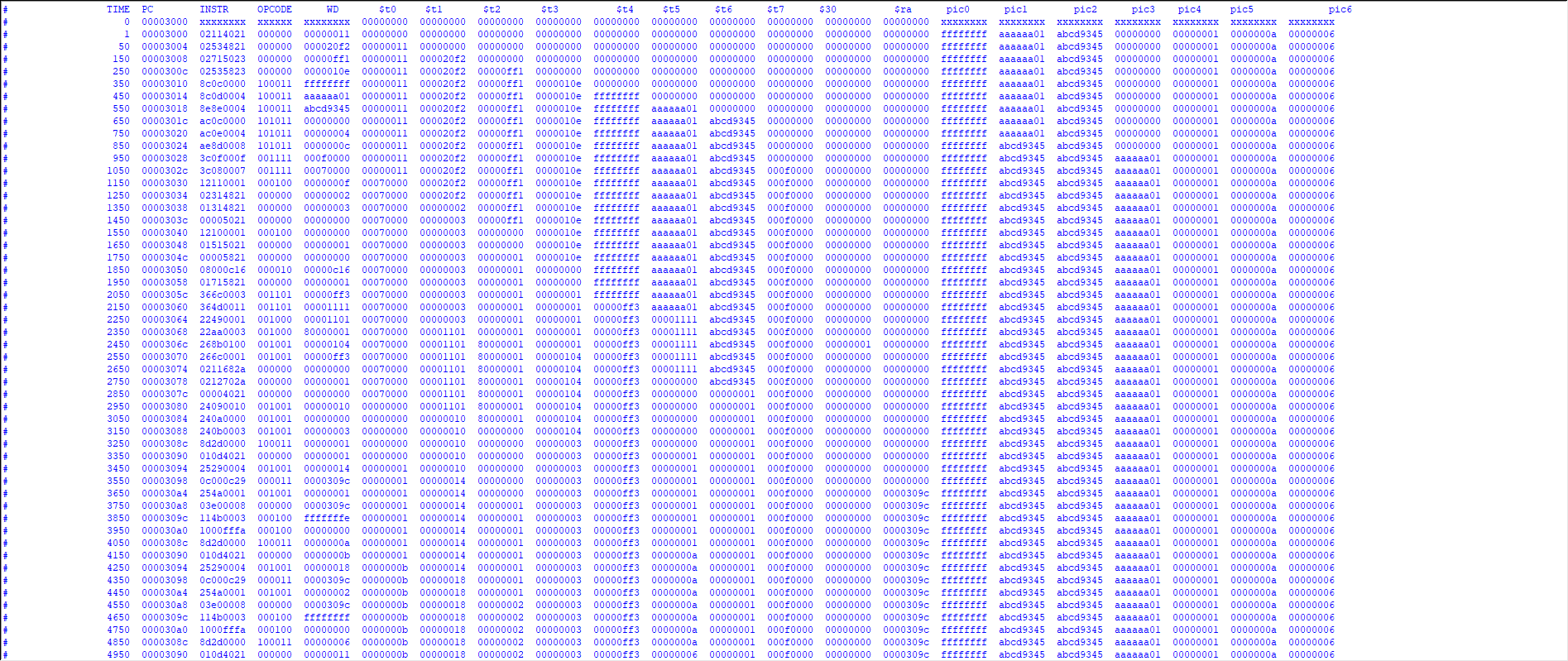
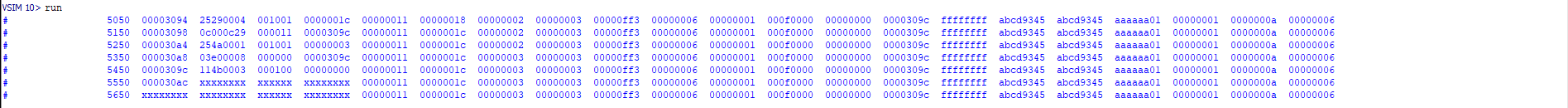


3、新增测试程序



五、modelsim测试

1、测试结果

 运行前将test1\_code.txt读入IM，test1\_dm.txt读入DM，test1\_gpr.txt读入GPR。

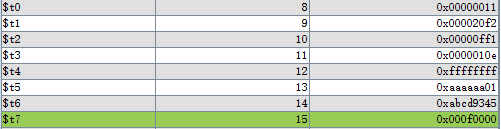
测试选择每次运行一条指令后输出时间（TIME），指令地址（PC），指令（INSTR）， 操作码（OPCODE），寄存器写入内容（WD），$t0~$t7号寄存器，$30寄存器，$ra 寄存器，内存地址第0~6个字内容（命名为pics0~pics6）。其余寄存器及内存单 元未使用，故未输出。

2、检验

在测试代码中，每个seg仅对同一个寄存器/内存单元操作一次，因此可以在每个 seg段最后检验。

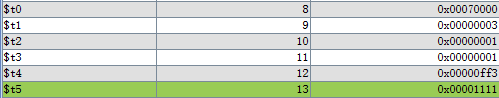
（1）seg0段末：

在TIME=1150时，代码执行完lui,$t7,15，对$t0~$t7、内存单元分别进行验证， 下图分别为Mars寄存器、Mars内存单元、modelsim仿真截图，验证一致。



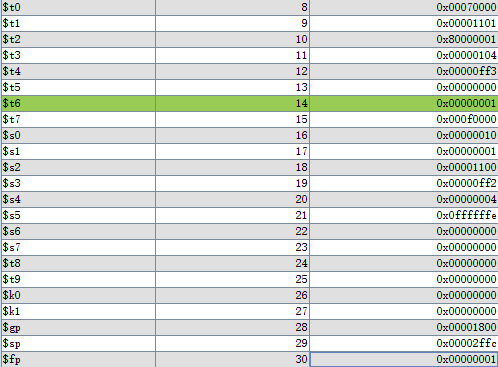
（2）seg1段末：

在TIME=2250时，代码执行完ori $t5,$s2,17，对$t0~$t7分别进行验证， 下图分别为Mars寄存器、modelsim仿真截图，验证一致。



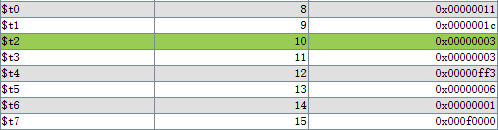
（3）seg2段末：

在TIME=2850时，代码执行完slt $t6,$s0,$s2，对$t0~$t7,$30分别进行验证， 下图分别为Mars寄存器、modelsim仿真截图，验证一致。（由于addi $t2,$s5,3 指令运算溢出，因此$30第0位被置1一次）



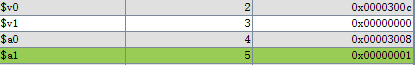
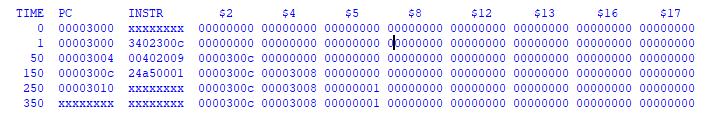
（4）seg3至程序末尾：

在TIME=5450时，代码全部执行完，对$t0进行验证。下图分别为Mars寄存 器、modelsim仿真截图，验证一致。（一共循环3次，$t0的值为内存单元 pics4~pics6之和）



3、新增测试程序的检验

新增测试程序一共四条，执行顺序为3000，3004，300c，对$2，$4，$5进行验证。下图分别为Mars寄存器、modelsim仿真截图，验证一致。

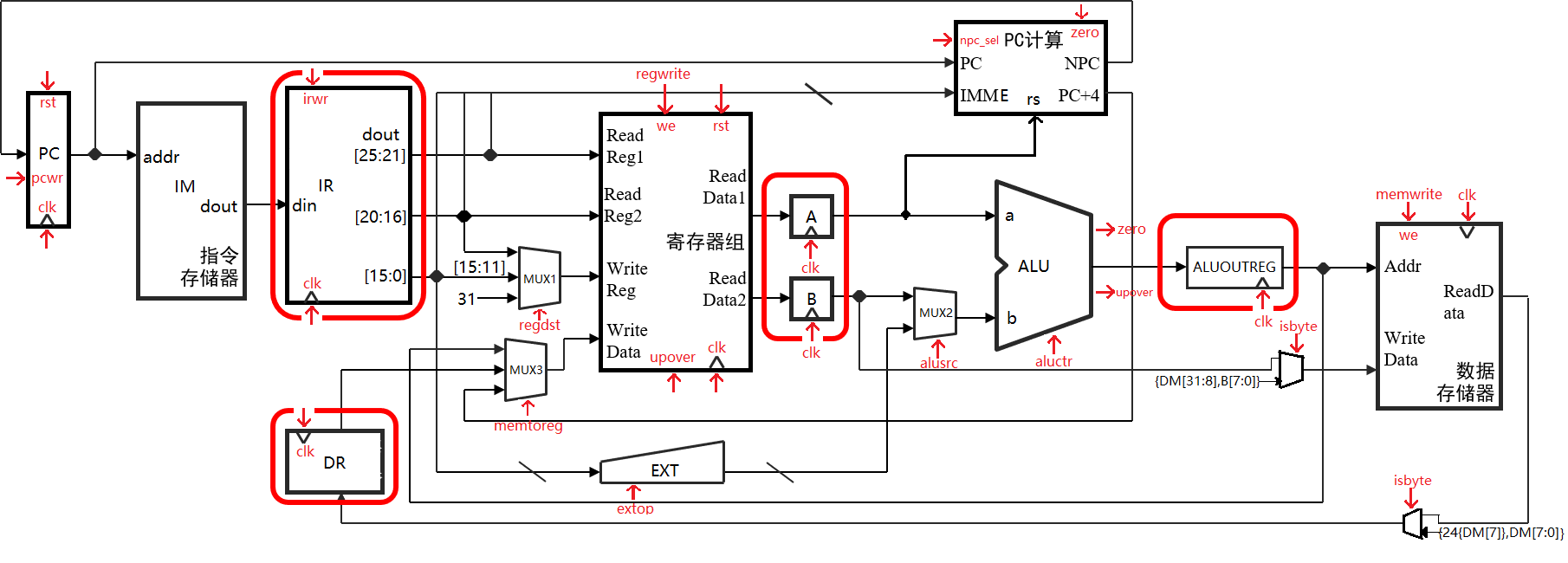


六、收获体会

这次课设Project1加深了我对计算机组成原理的理解，对控制器、内存单元、指令存储单元、寄存器等CPU内部结构及其工作原理有了更深的理解。在我一次次遇到问题、毫无思绪的时候，在老师的帮助下，查看教材、网上搜索资料最终使我获得了启发，最终耗时6天完成。除此之外，我对modelsim仿真、mars软件应用更加熟悉，为Project2、Project3打下基础。

**Project2**

一、总体数据通路结构设计图



二、模块描述

1. ctrl（控制模块）

（1）基本描述

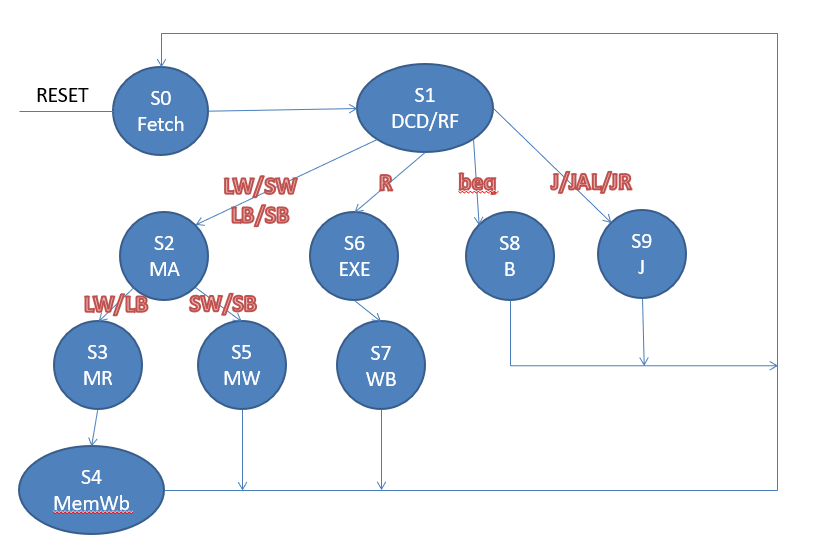
控制模块主要功能是根据指令中func码字段和opcode码字段分析出当 前指令类型，结合当前状态，进而给出不同的控制信号。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| ret | I | 复位信号 |
| instr[31:0] | I | 接收当前指令 |
| zero | I | ALU中输入两个相等信号标志 |
| alu\_ctr  [1:0] | O | alu功能选择信号  00：ALU执行加法  01：ALU执行减法  10：ALU执行或运算  11：a>b输出0;a<b输出1 |
| ext\_op  [1:0] | O | ext功能选择信号  00：0扩展  01：符号扩展  10：低位补零扩展 |
| npc\_sel  [1:0] | O | 当前指令识别信号  00：一般指令  01：beq指令  10：j指令  11：jr指令 |
| memwrite | O | dm写入控制信号  0：不写入  1：写入 |
| regweite | O | gpr(寄存器组)写入控制信号  0：不写入  1：写入 |
| memtoreg  [1:0] | O | gpr写入数据选择信号  00：写入ALUOUTreg的结果  01：写入DR提供的内容  10:写入PC+4的结果 |
| alusrc | O | alu的inputB选择信号  0：寄存器B中的数据传入ALU  1：扩展后的32位立即数写入ALU |
| regdst  [1:0] | O | gpr的writereg写入数据选择信号  00：rt写入writereg  01：rd写入writereg  10：31写入writereg |
| pcwr | I | PC写入控制信号  0：不写入  1：写入 |
| irwr | I | IR写入控制信号  0：不写入  1：写入 |
| isbyte | I | 字节命令信号  0：非lb/sb命令  1：lb/sb命令 |
| neg | I | 寄存器为负标志  0：非负  1：负数 |

（3）功能定义

根据指令中func码字段和opcode码字段分析出当前指令类型，结合状态机， 进而给出不同的控制信号。



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | s0 | s1 | s2 | s3 | s4 | s5 | s6 | s7 | s8 | s9 |
| pcwr | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | zero==1  neg==1 | 1 |
| npc\_sel | 00 | xx | xx | xx | xx | xx | xx | xx | 01 | jr:11  jal:10  j:10 |
| irwr | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| regwrite | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | bltzal | jal:1 |
| memwrite | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| aluctr | xx | xx | 00 | xx | xx | xx | 注1 | xx | xx | xx |
| regdst | xx | xx | xx | xx | 00 | xx | xx | 注2 | 10 | jal:10 |
| memtoreg | xx | xx | xx | xx | 01 | xx | xx | 00 | 10 | jal:10 |
| extop | xx | xx | 01 | xx | xx | xx | 注3 | xx | xx | xx |
| alusrc | 0 | 0 | 1 | 0 | 0 | 0 | 注4 | 0 | 0 | 0 |
| isbyte | 0 | 0 | 0 | lb:1  lw:0 | 0 | sb:1  sw:0 | 0 | 0 | 0 | 0 |

注1：slt: 11，ori || lui: 10，subu: 01，addu ||addiu || addi: 00

注2：addi || addiu || ori || lui：00，addu || subu || slt：01

注3：lui: 10，ori: 00，addi || addiu: 01

注4：addi || addiu || ori || lui：1，addu || subu || slt：0

2、im

（1）基本描述

im的功能是储存指令，根据不同的输入地址，输出相应的指令至IR。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| addr  [11:0] | I | 当前指令的地址（低12位） |
| dout  [31:0] | O | 输出32位指令 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 输出指令 | 根据不同的输入地址，输出相应的指令至IR |

3、pc

（1）基本描述

pc储存当前指令的地址，第一条指令地址为0x0000\_3000

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号  0：无效  1：有效 |
| pcwr | I | 写有效信号  0：无效  1：有效 |
| npc[31:0] | I | 从npc模块接收下一条指令地址 |
| pc[31:] | O | 输出当前指令地址 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 复位 | 当复位信号有效时，PC被设置为0x0000\_3000。 |
| 2 | 取指令 | 根据npc,输出指令 |
| 3 | 写入指令 | 在pcwr有效时，将npc写入pc |

4、ir

（1）基本描述

ir为指令寄存器，储存当前指令内容，每当状态机为s0时，写入信号有效， 在下一个时钟沿内容被刷新。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| irwr | I | 写有效信号  0：无效  1：有效 |
| din[31:0] | I | 写入指令寄存器内容 |
| dout[31:0] | O | 当前指令寄存器内容 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 储存指令 | 每当状态机为s0时，在下一个时钟沿内容被刷新，  否则始终保存当前指令 |

5、npc

（1）基本描述

根据控制信号npc\_sel、zero，确定下一条指令的地址。

（2）模块接口

|  |  |  |  |
| --- | --- | --- | --- |
| 信号名 | | 方向 | 描述 |
| npc\_sel  [1:0] | | I | 判断当前指令  00：一般指令  01：beq指令  10：j指令、jal指令  11：jr指令 |
| zero | | I | ALU输入端口内容相同标志。  0：内容不同  1：内容相同 |
| imme[25:0] | | I | 从指令寄存器中获得部分跳转地址 |
| pc[31:0] | | I | 从pc中获得部分跳转地址 |
| rs[31:0] | | I | 从寄存器A中获得跳转地址 |
| npc[31:0] | | O | 下一条实际执行的指令地址 |
| pc\_4[31:0] | | O | 当前执行指令下一条的地址（pc） |
| neg | I | | 寄存器为负标志  0：非负  1：负数 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | R跳转 | npc=pc+4 |
| 2 | beq跳转 | (npc\_sel==BEQ&&zero)? pc\_4+{{14{offset[15]}},offset,2'b0} (npc\_sel==BEQ&&neg)? pc\_4+{{14{offset[15]}},offset,2'b0} |
| 3 | j跳转 | npc= {pc[31:28],imme,2'b0} |
| 4 | jr跳转 | npc=rs |

6、alu

（1）基本描述

运算器，根据ALUctr给出的不同信号，调用adder/substract/ori/slt子模块对 接收到的两个32位数据加/减/或/比较运算，将结果输出。

（2）模块端口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| a | I | 被操作数 |
| b | I | 操作数 |
| alu\_ctr  [1:0] | I | 功能选择信号  00：加运算  01：减运算  10：或运算  11：比较运算 |
| output  [31:0] | O | 运算结果 |
| zero | O | 零标志位  0：output不为0  1：output为0 |
| upover | O | 加法溢出标志  0：不溢出  1：溢出 |
| neg | O | 寄存器为负标志  0：非负  1：负数 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 加 | a+b |
| 2 | 减 | a-b |
| 3 | 或 | a|b |
| 4 | 比较 | a>b:0;a<b:1 |

7、gpr

（1）基本描述

GPR包含32个寄存器，有两个读端口，一个写端口，一个写控制信号，两个 读寄存器地址信息，一个写寄存器地址信息，复位信号，溢出信号和时钟信号。 两个读端口可以随时读出当前地址对应的寄存器内容，写端口写入数据需要 we(regwrite)信号为1。寄存器号的输入都是5位的，数据线为32位。

（2）模块端口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| we  (regwrite) | I | 写入控制信号  0：不写入  1：写入 |
| rst | I | 复位信号  0：不复位  1：复位 |
| upover | I | 加法溢出信号  0：不溢出  1：溢出 |
| readreg1  [4:0] | I | 读寄存器地址1 |
| readreg2  [4:0] | I | 读寄存器地址2 |
| writereg  [4:0] | I | 写寄存器地址 |
| writedata  [31:0] | I | 写寄存器内容 |
| readdata1  [31:0] | O | 读寄存器内容1 |
| readdata2  [31:0] | O | 读寄存器内容2 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 读寄存器 | 根据readreg1、readreg2地址读取寄存器内容到readdata1、readdata2 |
| 2 | 写寄存器 | 根据writeref地址将writedata数据写入寄存器 |
| 3 | 溢出置零 | 加法溢出时，第30号寄存器第0位置1 |
| 4 | 复位 | 28号寄存器置0000\_1800  29号寄存器置0000\_2ffc  其余置0 |

8、dm

（1）基本描述

DM为内存单元，有一个地址输入端口，一个写入数据端口，一个读出数据端 口和一个写控制信号。写控制信号控制将数据写入对应地址或者将数据从对应 地址读出。

（2）模块端口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| we  （memwrite） | I | 写入控制信号  0：不写入  1：写入 |
| addr[11:0] | I | 内存地址 |
| din[31:0] | I | 写内存内容 |
| dout[31:0] | O | 读内存内容 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 读内存 | 根据addr地址读取内存内容到dout |
| 2 | 写内存 | 根据addr地址将din数据写入内存 |

9、ext

（1）基本描述

ext为扩展器，有一个16位数据写入端口，一个控制信号和一个32位 数据输出端口。根据功能选择信号的不同将输入数据进行0扩展/符号扩展/低 位补0扩展。

（2）模块端口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| din[15:0] | I | 输入16位数据 |
| ext\_op  [1:0] | I | 功能选择信号  00：0扩展  01：符号扩展  10：低位补零扩展 |
| dout[31:0] | O | 输出32位数据 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 0扩展 | 高16位补0，低16位为input数据 |
| 2 | 符号扩展 | 高16位符号扩展，低16位为input数据 |
| 3 | 低位补0扩展 | 高16位补为input数据，低16为补0 |

10、mux

（1）基本描述

多选一选择器，内有32位3选1，31位2选1，5位3选1三个模块。

（2）模块端口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| a0 | I | 数据1 |
| a1 | I | 数据2 |
| a2 | I | 数据3 |
| ch | I | 数据选择信号 |
| out | O | 数据输出 |

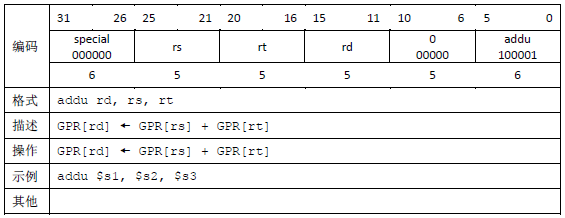
（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 数据选择 | 根据ch，从若干个输入数据中选择一个输出 |

三、指令描述

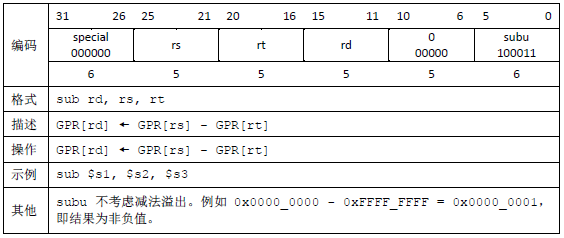
1. addu

功能：无符号加



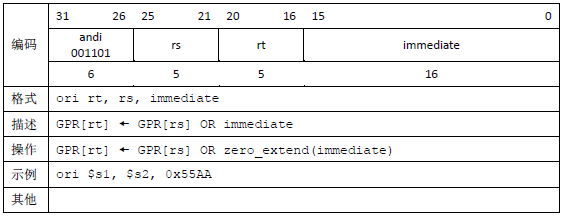
2、subu

功能：无符号减



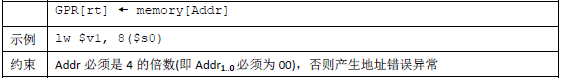
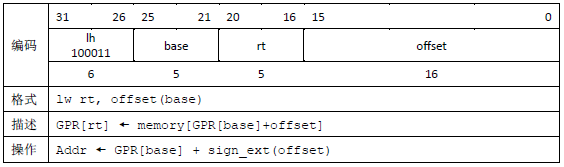
3、ori

功能：或立即数



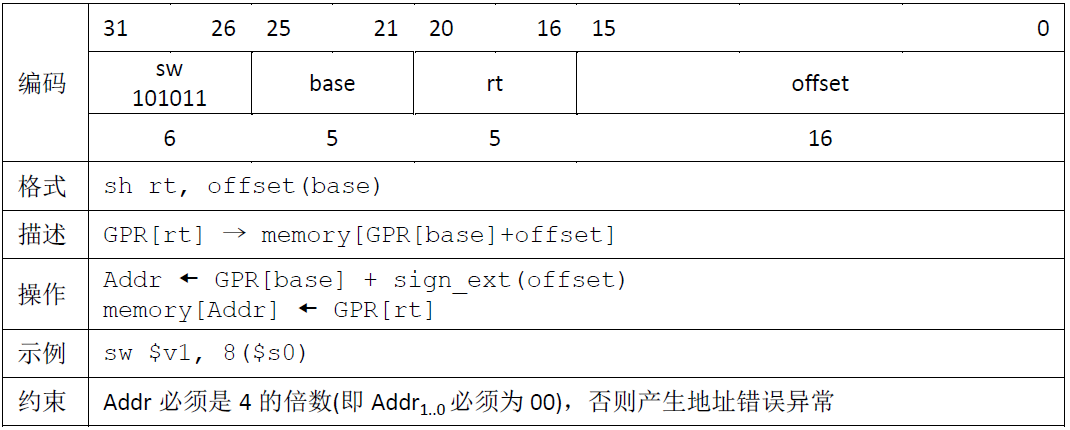
4、lw

功能：加载字



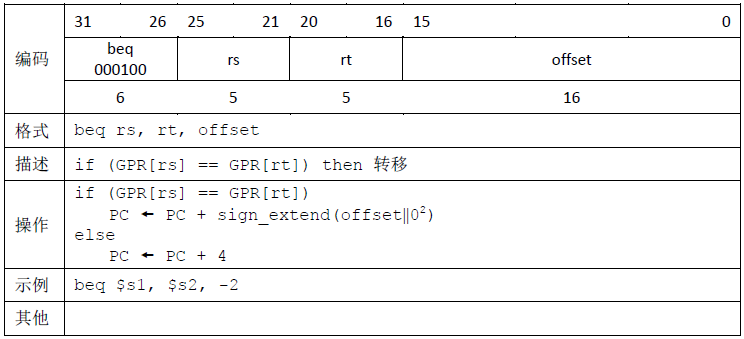
5、sw

功能：储存字



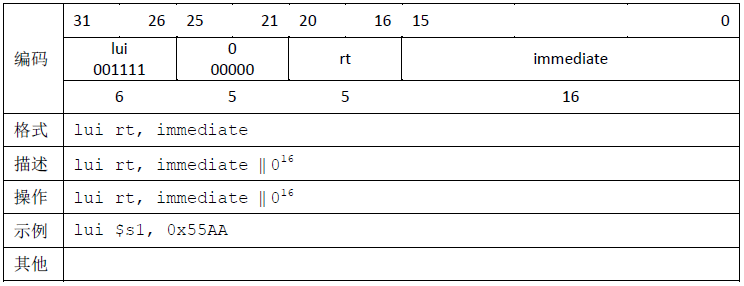
6、beq

功能：相等时转移



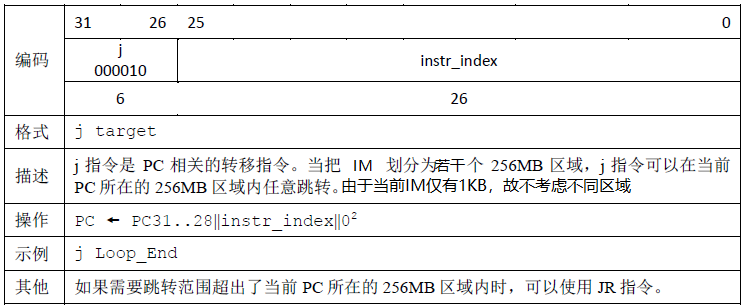
7、lui

功能：立即数加载至高位，低位补0



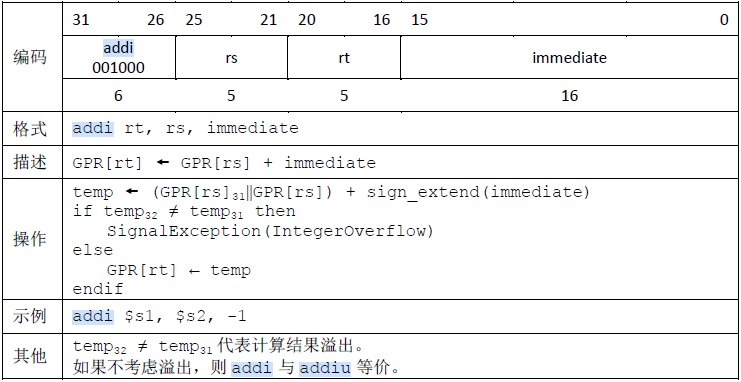
8、j

功能：跳转



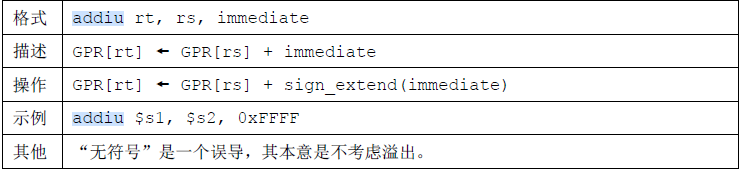
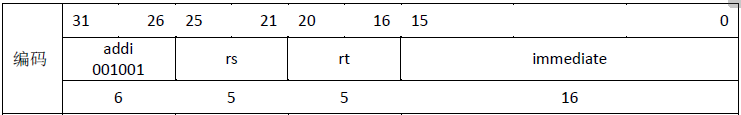
9、addi

功能：符号加立即数



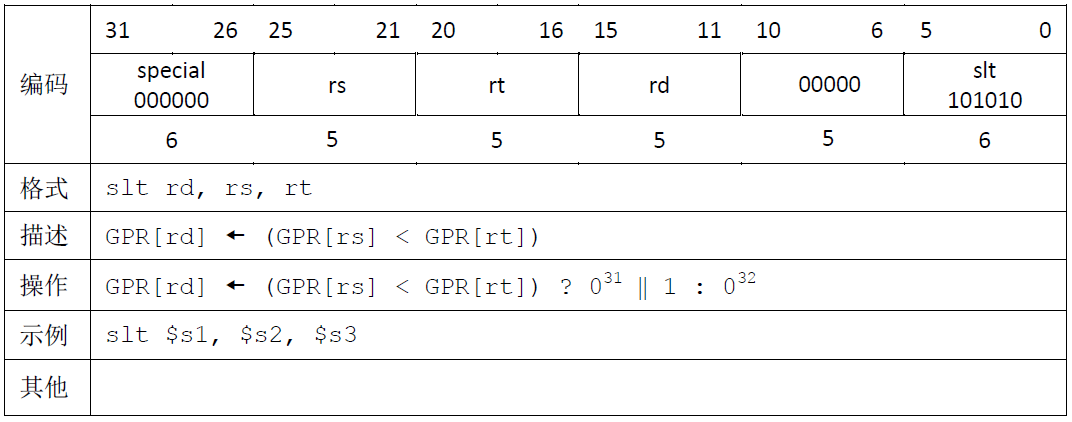
10、addiu

功能：无符号加立即数



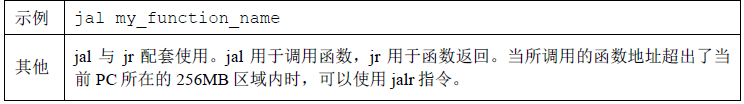
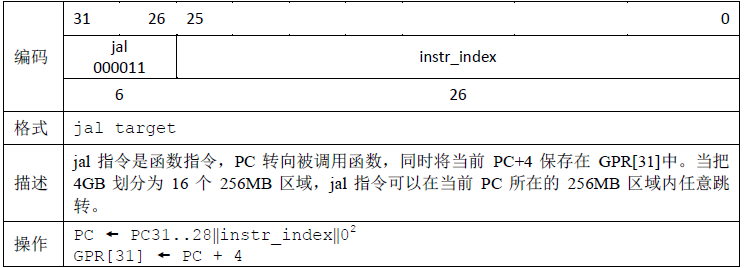
11、slt

功能：小于立即数置1



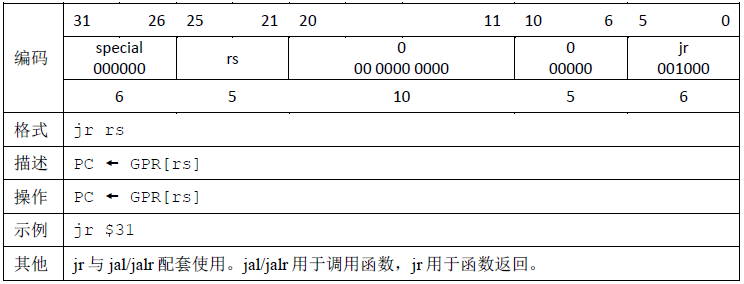
12、jal

功能：跳转并链接



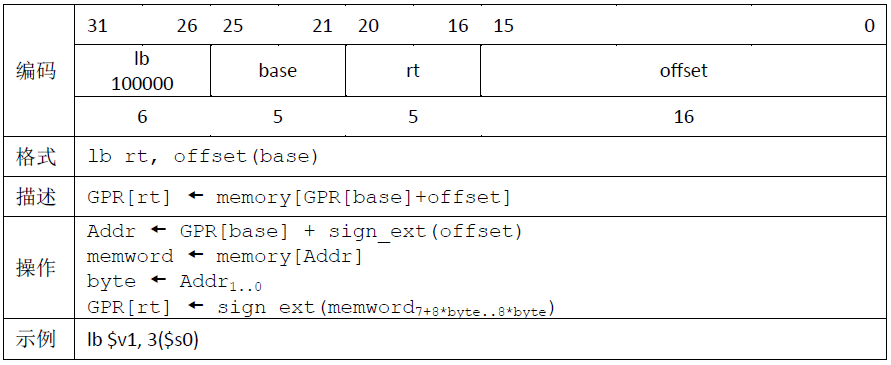
13、jr

功能：跳转至寄存器



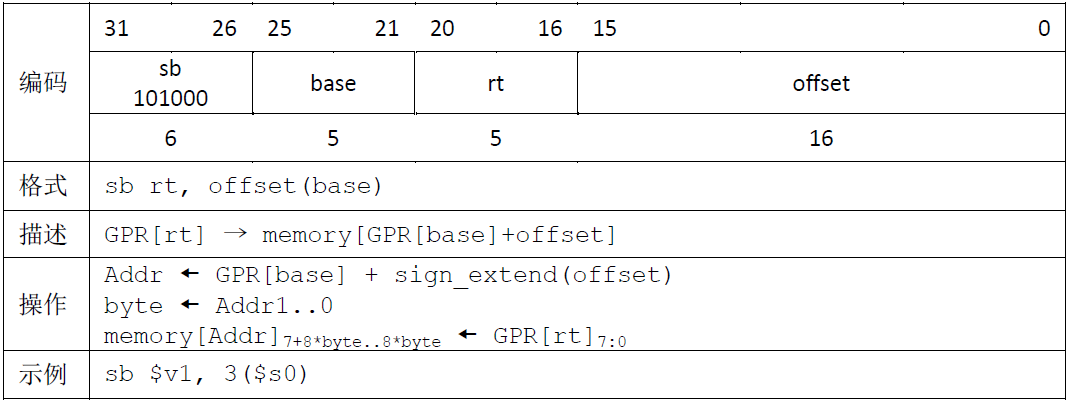
14、lb

功能：加载字节



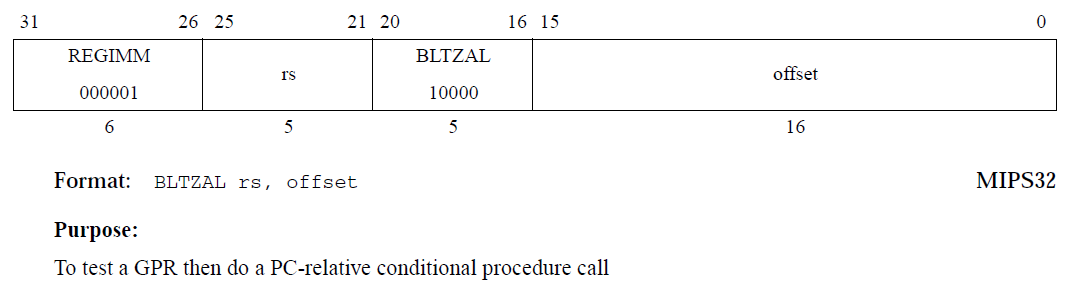
15、sb

功能：储存字节



16、bltzal

功能：小于跳转并链接



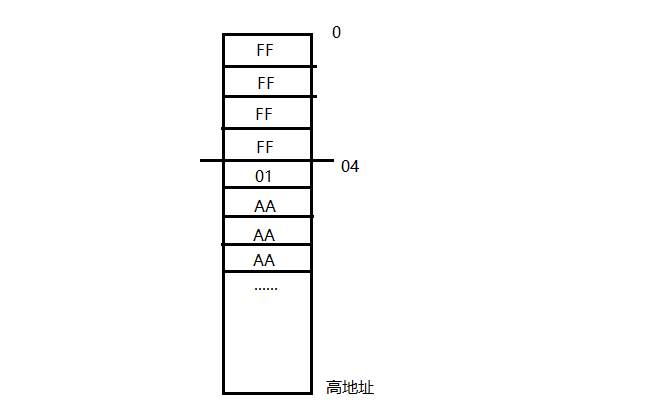
四、测试程序

1、测试前数据准备

（1）寄存器组数据

$s0=0x0000 0010

$s1=0x0000 0001

 $s2=0x0000 1100

$s3=0x0000 0FF2

$s4=0x0000 0004

$s5=0x7FFF FFFE

（2）内存单元数据（字地址）

00：0xFFFF FFFF

04：0XAAAA AA01

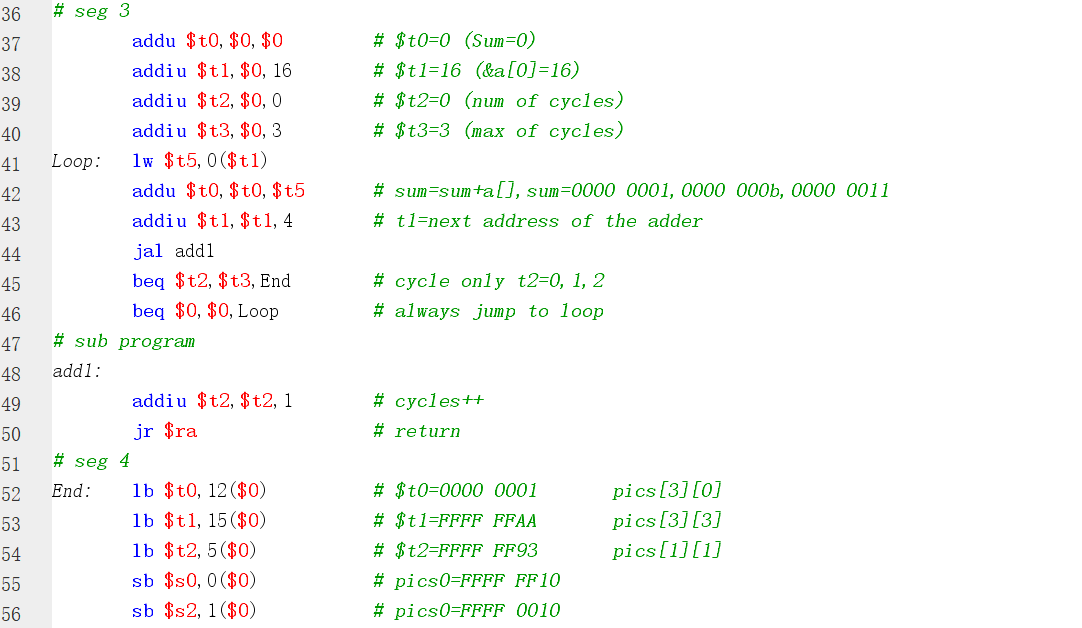
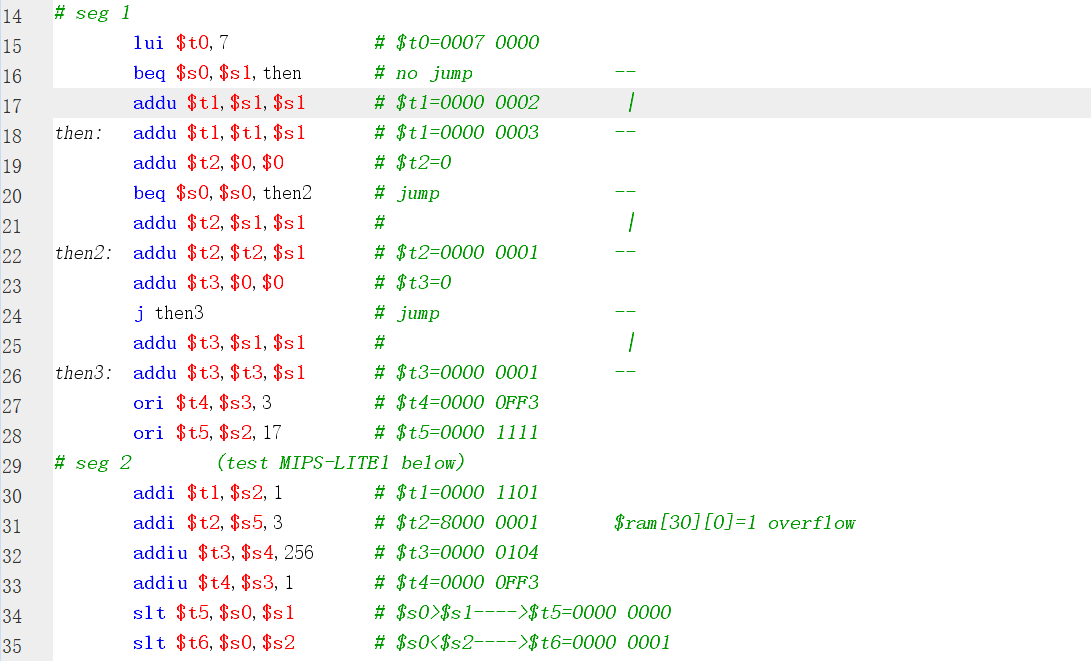
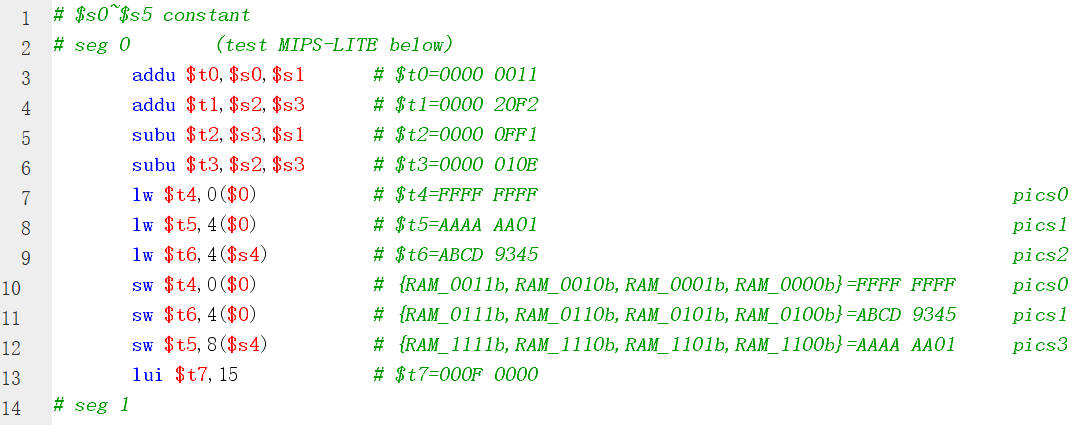
08：0XABCD 9345

16: 0X0000 0001

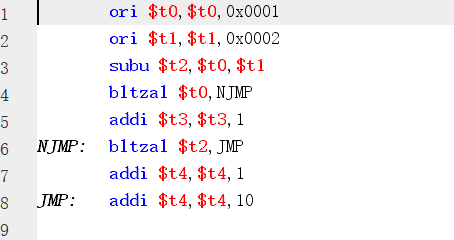
20: 0X0000 000A

24: 0X0000 0006

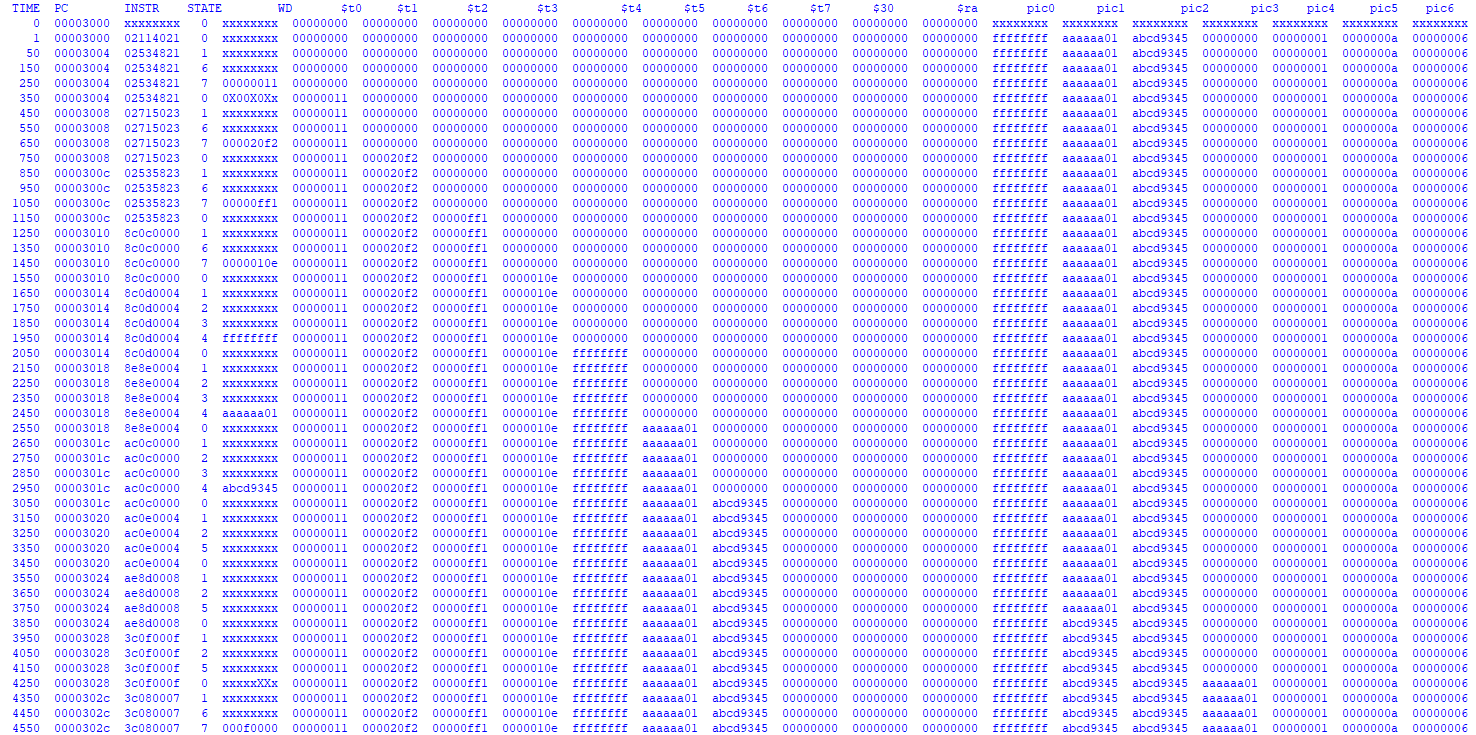
2、测试程序

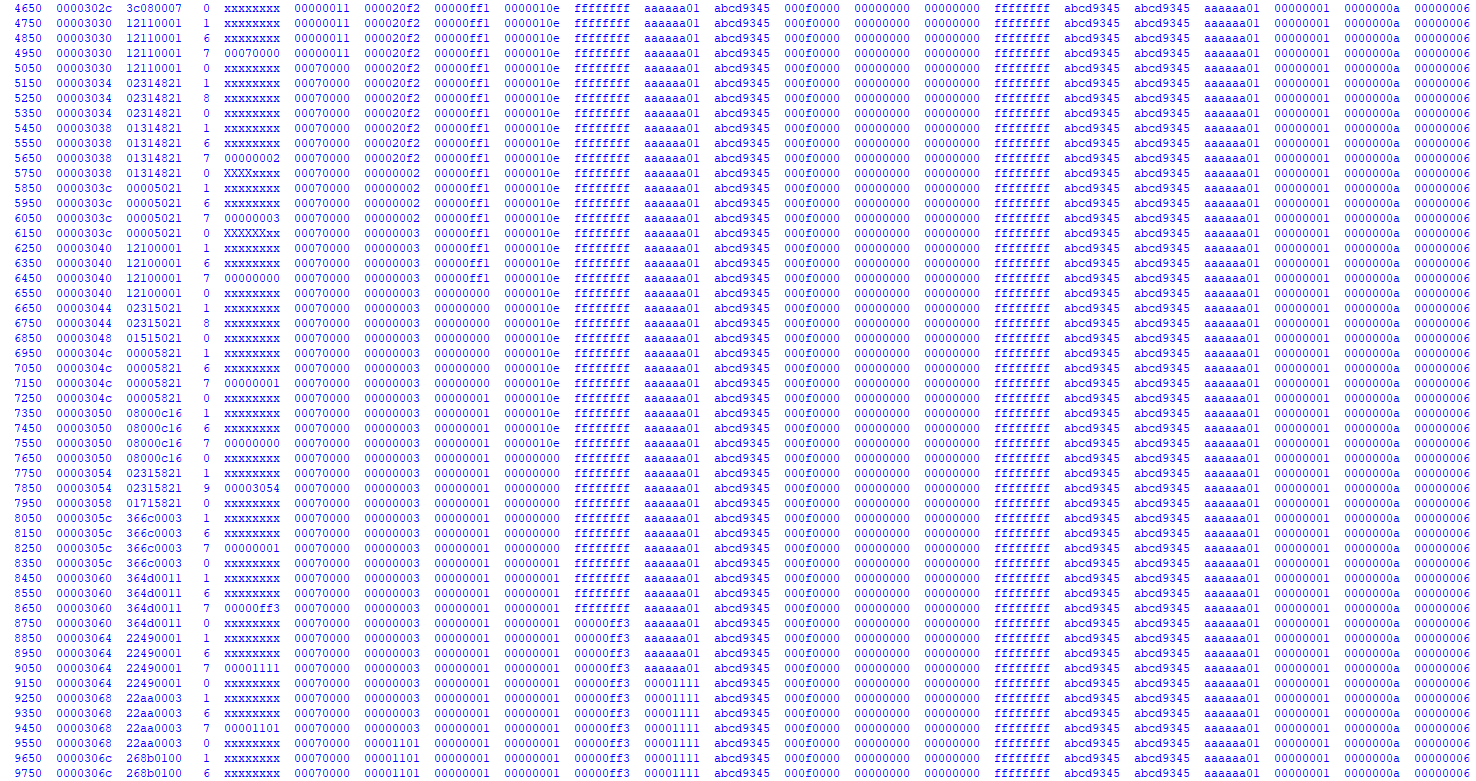
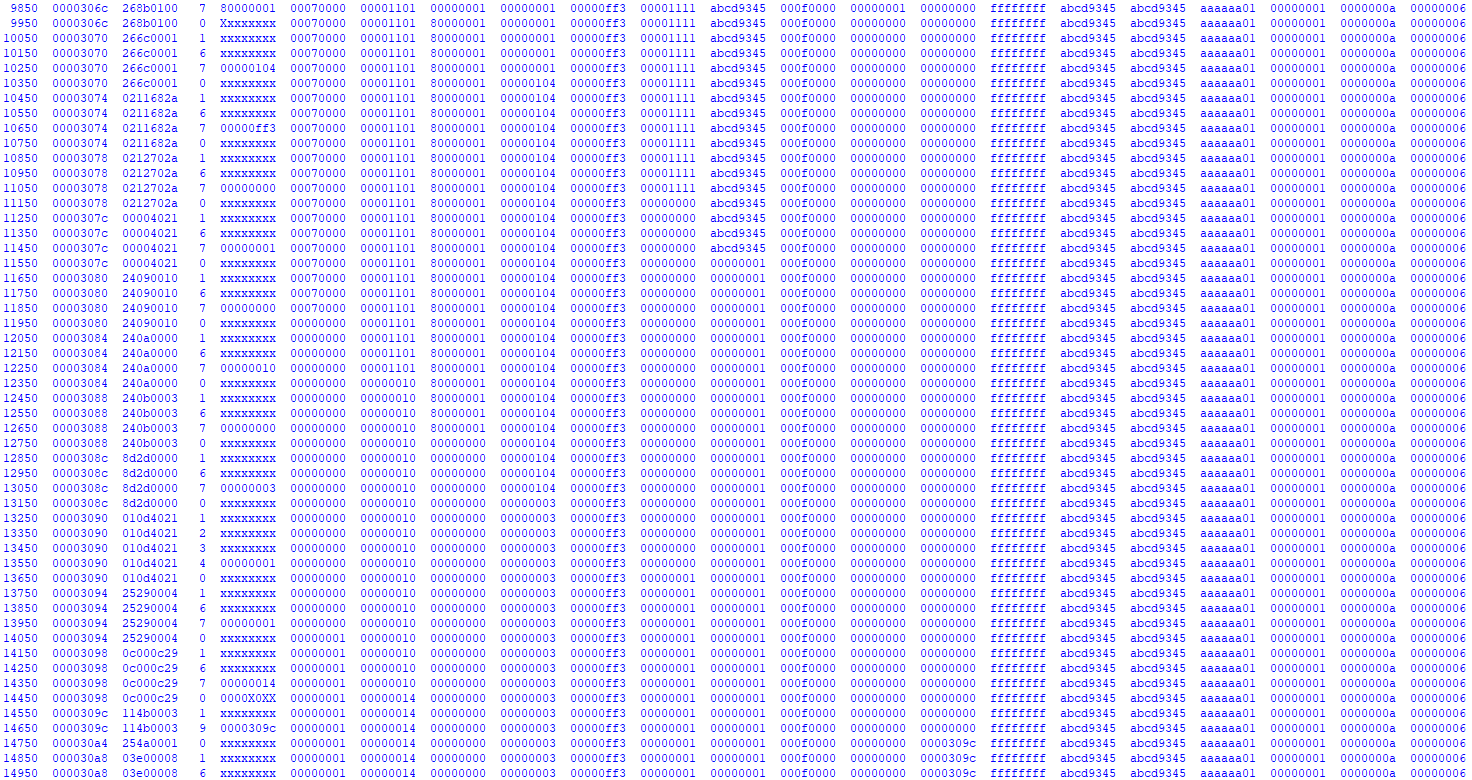


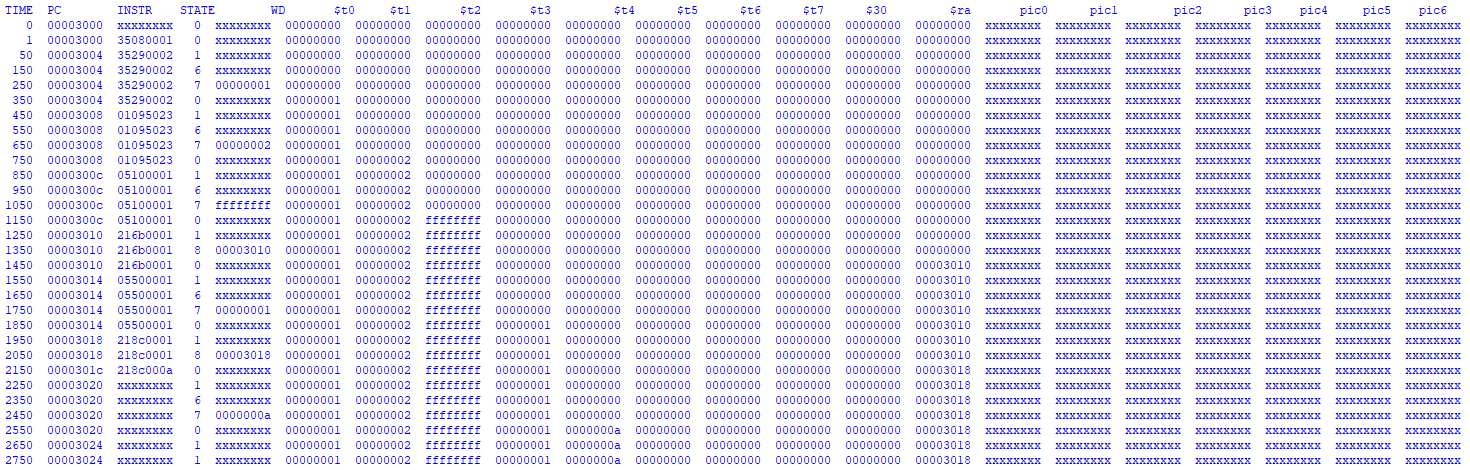
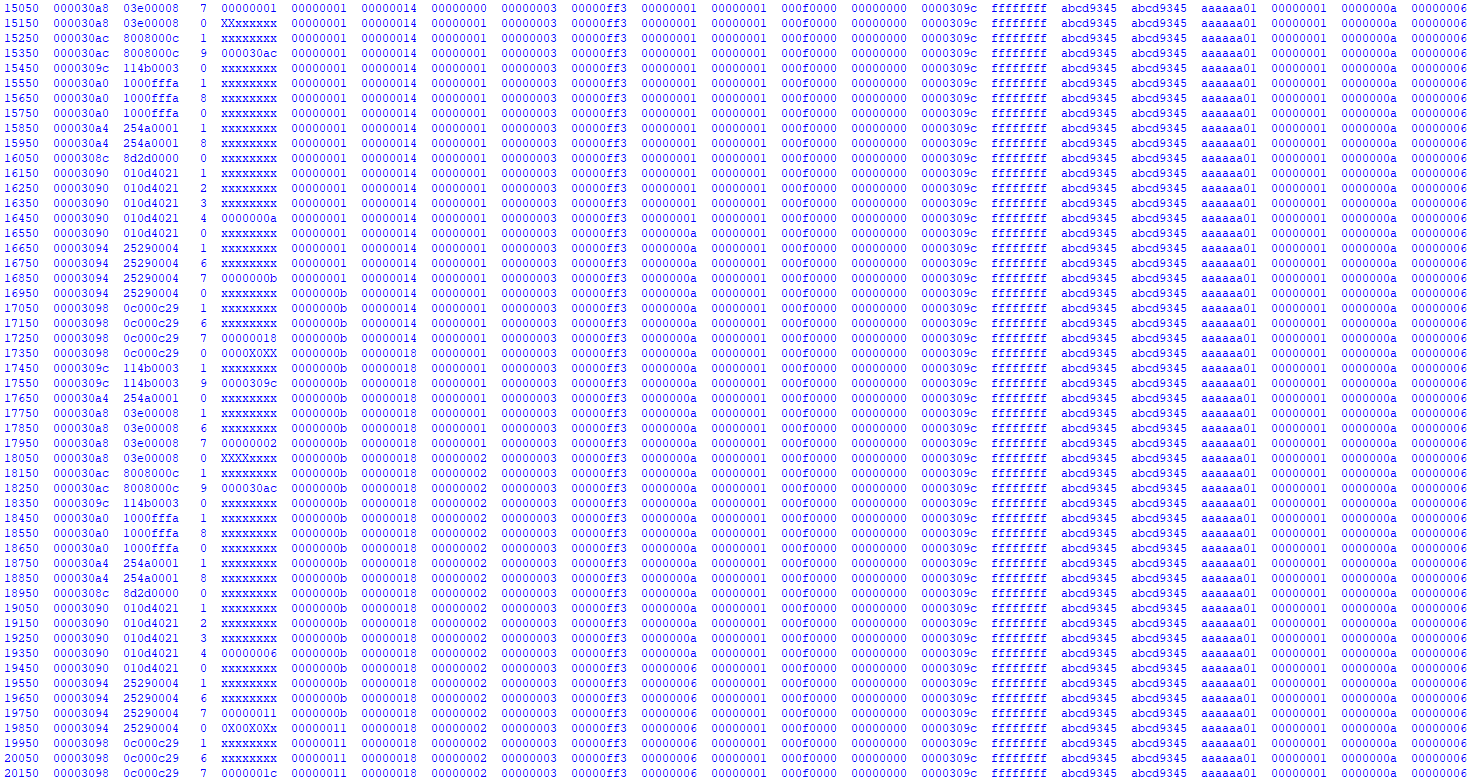
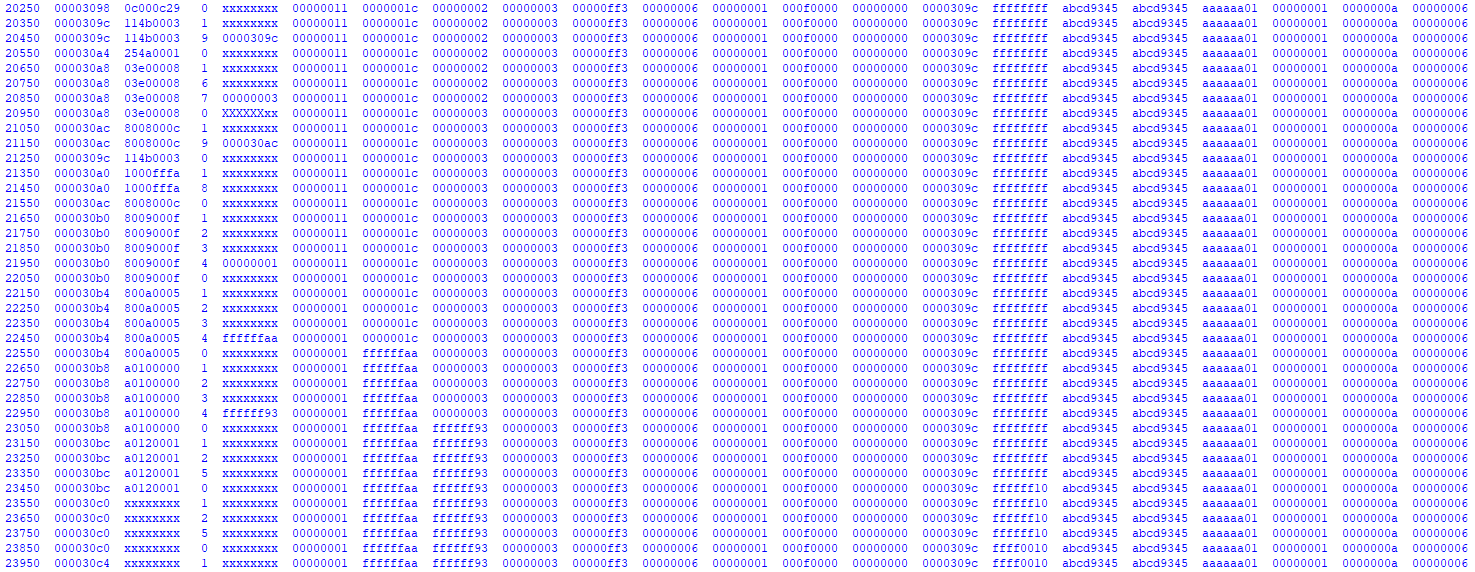
1. 新指令测试程序



五、modelsim测试

 1、测试结果



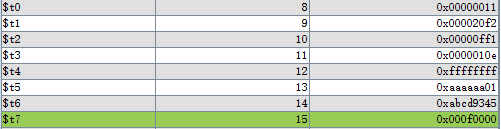
检查新指令时测试：

2、检验

在测试代码中，每个seg仅对同一个寄存器/内存单元操作一次，因此可以在每个 seg段最后检验。

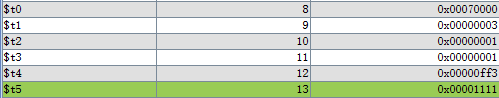
（1）seg0段末：

在TIME=4650时，代码执行完lui,$t7,15，对$t0~$t7、内存单元分别进行验证， 下图分别为Mars寄存器、Mars内存单元、modelsim仿真截图，验证一致。



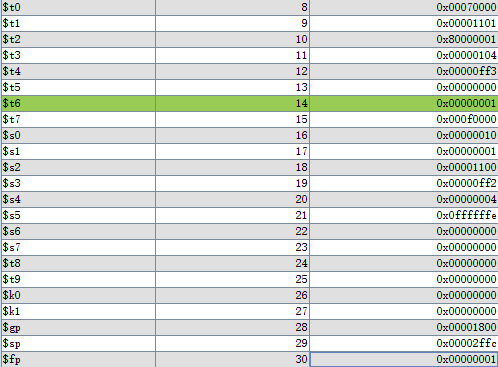
（2）seg1段末：

在TIME=9150时，代码执行完ori $t5,$s2,17，对$t0~$t7分别进行验证， 下图分别为Mars寄存器、modelsim仿真截图，验证一致。



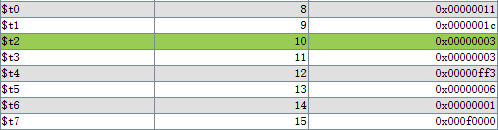
（3）seg2段末：

在TIME=11550时，代码执行完slt $t6,$s0,$s2，对$t0~$t7,$30分别进行验证， 下图分别为Mars寄存器、modelsim仿真截图，验证一致。（由于addi $t2,$s5,3 指令运算溢出，因此$30第0位被置1一次）



（4）seg3段末：

在TIME=21550时，代码执行完beq $0,$0,Loop，跳出循环，对$t0进行验证。 下图分别为Mars寄存器、modelsim仿真截图，验证一致。（一共循环3次， $t0的值为内存单元pics4~pics6之和）



（5）seg4段末（程序结束）

在TIME=23850，代码全部执行完，对$t0~$t2，pics0进行验证。下图分别为 Mars寄存器、Mars内存单元、modelsim仿真截图，验证一致。



新指令检验：

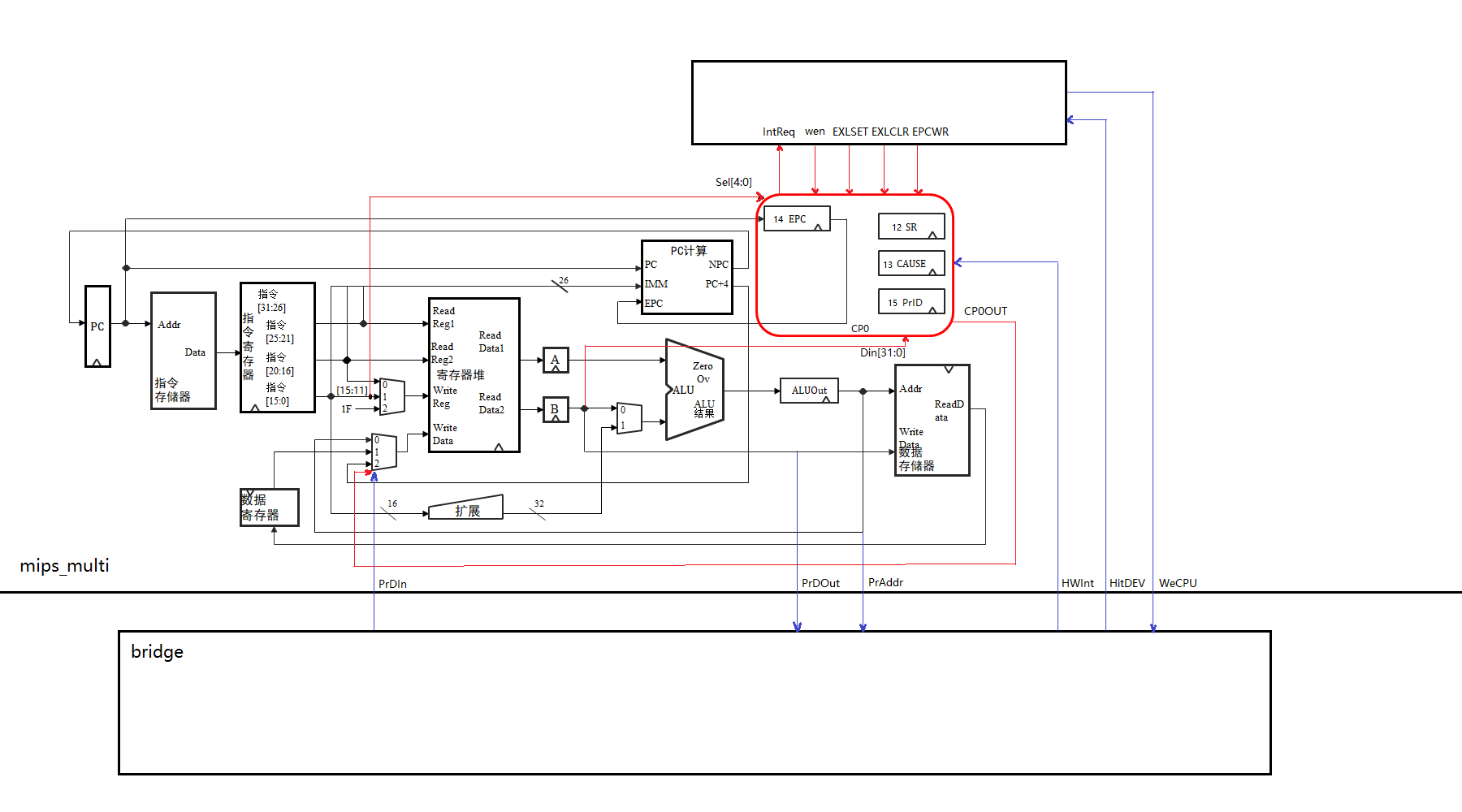


六、收获体会

这次课设Project2加深了我对计算机组成原理的理解，对状态机和多周期有了更深的理解。这项作业耗时3天完成。除此之外，我对modelsim仿真、mars软件应用更加熟悉，为Project3打下基础。

**Project3**

1. 总体数据通路结构设计图



1. 模块描述（新增）
2. 1、ctrl（控制模块）

（1）基本描述

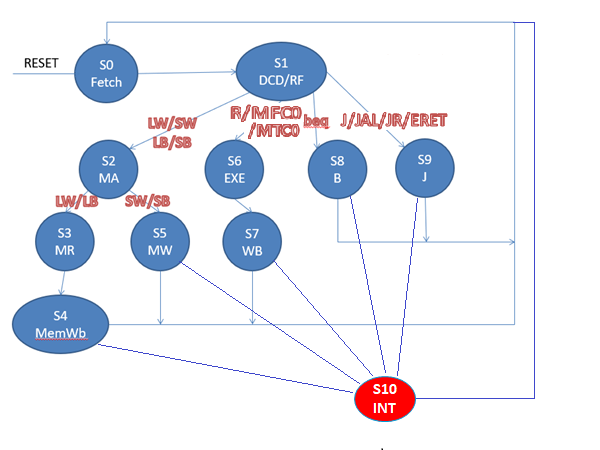
控制模块主要功能是根据指令中func码字段和opcode码字段分析出当 前指令类型，结合当前状态，进而给出不同的控制信号。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| ret | I | 复位信号 |
| instr[31:0] | I | 接收当前指令 |
| zero | I | ALU中输入两个相等信号标志 |
| alu\_ctr  [1:0] | O | alu功能选择信号  00：ALU执行加法  01：ALU执行减法  10：ALU执行或运算  11：a>b输出0;a<b输出1 |
| ext\_op  [1:0] | O | ext功能选择信号  00：0扩展  01：符号扩展  10：低位补零扩展 |
| npc\_sel  [1:0] | O | 当前指令识别信号  00：一般指令  01：beq指令  10：j指令  11：jr指令 |
| memwrite | O | dm写入控制信号  0：不写入  1：写入 |
| regweite | O | gpr(寄存器组)写入控制信号  0：不写入  1：写入 |
| memtoreg  [1:0] | O | gpr写入数据选择信号  00：写入ALUOUTreg的结果  01：写入DR提供的内容  10:写入PC+4的结果 |
| alusrc | O | alu的inputB选择信号  0：寄存器B中的数据传入ALU  1：扩展后的32位立即数写入ALU |
| regdst  [1:0] | O | gpr的writereg写入数据选择信号  00：rt写入writereg  01：rd写入writereg  10：31写入writereg |
| pcwr | I | PC写入控制信号  0：不写入  1：写入 |
| irwr | I | IR写入控制信号  0：不写入  1：写入 |
| isbyte | I | 字节命令信号  0：非lb/sb命令  1：lb/sb命令 |
| intreq | I | 中断请求信号  0：非中断  1：中断 |
| hitdev | I | 桥命中外设信号  0：未命中外设  1：命中外设 |
| epcwr | O | CP0中EPC寄存器写信号  0：不写入  1：从PC写入 |
| wen | O | CP0中寄存器写信号  0：不写入  1：从Din写入寄存器 |
| exlset | O | 中断嵌套置1信号  0：exl不置1  1：exl置1 |
| exlclr | O | 中断嵌套置0信号  0：exl不置0  1：exl置0 |
| fsm[3:0] | O | 输出当前状态 |

（3）功能定义

根据指令中func码字段和opcode码字段分析出当前指令类型，结合状态机， 进而给出不同的控制信号。



|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | s0 | s1 | s2 | s3 | s4 | s5 | s6 | s7 | s8 | s9 | s10 |
| pcwr | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | zero==1 | 1 | 1 |
| npc\_sel | 000 | xx | xx | xx | xx | xx | xx | xx | 001 | jr:011  jal:010  j:010  eret101 | 100 |
| irwr | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| regwrite | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | jal:1 | 0 |
| memwrite | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| aluctr | xx | xx | 00 | 00 | 00 | 00 | 注1 | xx | xx | xx | xx |
| regdst | xx | xx | xx | xx | 00 | xx | xx | 注2 | xx | jal:10 | xx |
| memtoreg | xx | xx | xx | xx | 注6 | xx | xx | 注5 | xx | jal:010 | xx |
| extop | xx | xx | 01 | 01 | 01 | 01 | 注3 | xx | xx | xx | xx |
| alusrc | 0 | 0 | 1 | 1 | 1 | 1 | 注4 | 0 | 0 | 0 | 0 |
| isbyte | 0 | 0 | 0 | lb:1  lw:0 | 0 | sb:1  sw:0 | 0 | 0 | 0 | 0 | 0 |

注1：slt: 11，ori || lui: 10，subu: 01，addu ||addiu || addi: 00

注2：addi || addiu || ori || lui || mfc0：00，addu || subu || slt：01

注3：lui: 10，ori: 00，addi || addiu: 01

注4：addi || addiu || ori || lui：1，addu || subu || slt：0

注5：仅当mfc0=1时为011，其余000

注6：hitdev=0时为001（为1时判断是否为lw，若是，S2~S4皆为100）

2、im（同Project2）

3、pc（同Project2）

4、ir（同Project2）

5、npc

（1）基本描述

根据控制信号npc\_sel、zero，确定下一条指令的地址。

（2）模块接口

|  |  |  |  |
| --- | --- | --- | --- |
| 信号名 | | 方向 | 描述 |
| npc\_sel  [1:0] | | I | 判断当前指令  00：一般指令  01：beq指令  10：j指令、jal指令  11：jr指令 |
| zero | | I | ALU输入端口内容相同标志。  0：内容不同  1：内容相同 |
| imme[25:0] | | I | 从指令寄存器中获得部分跳转地址 |
| pc[31:0] | | I | 从pc中获得部分跳转地址 |
| rs[31:0] | | I | 从寄存器A中获得跳转地址 |
| npc[31:0] | | O | 下一条实际执行的指令地址 |
| pc\_4[31:0] | | O | 当前执行指令下一条的地址（pc） |
| epc[31:2] | I | | 从CP0中获取EPC值 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | R跳转 | npc=pc+4 |
| 2 | beq跳转 | (npc\_sel==BEQ&&zero)? pc\_4+{{14{offset[15]}},offset,2'b0} |
| 3 | j跳转 | npc= {pc[31:28],imme,2'b0} |
| 4 | jr跳转 | npc=rs |
| 5 | 中断 | (npc\_sel==3'b100)?32'h0000\_4180 |
| 6 | 中断返回 | (npc\_sel==3'b101)?{epc,2'b00} |

6、alu（同Project2）

7、gpr（同Project2）

8、dm（同Project2）

9、ext（同Project2）

10、mux（同Project2）

11、cp0

（1）基本描述

协处理器，含有EPC、SR、CAUSE、PrID四个寄存器，存储产生异常或中断的指令的PC值，产生异常或中断的原因，提供屏蔽信息。中断产生、中断返回、指令MFC0、MTC0需要该模块。

（2）模块接口

|  |  |  |  |
| --- | --- | --- | --- |
| 信号名 | | 方向 | 描述 |
| clk | | I | 时钟信号 |
| reset | | I | 复位信号 |
| IntReq | O | | 中断请求信号  0：非中断  1：中断 |
| EPCWr | I | | EPC寄存器写信号  0：不写入  1：从PC写入 |
| Wen | I | | CP0中寄存器写信号  0：不写入  1：从Din写入寄存器 |
| EXLSet | I | | 中断嵌套置1信号  0：exl不置1  1：exl置1 |
| EXLClr | O | | 中断嵌套置0信号  0：exl不置0  1：exl置0 |
| pc[31:2] | | I | pc输入 |
| Din[31:0] | | I | 数据输入 |
| HWInt[5:0] | | I | 桥中断请求识别信号 |
| Sel[4:0] | | I | 寄存器选择信号 |
| epc[29:0] | | O | epc寄存器输出 |
| DOut[31:0] | | O | 寄存器输出 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 储存PC | 进入中断前储存PC+4 |
| 2 | 储存Prid | 储存个性化信息 |
| 3 | 屏蔽控制 | SR寄存器控制屏蔽、中断嵌套、全局中断 |
| 4 | 记录中断 | CAUSE寄存器记录当前哪些硬件中断有效 |

12、bridge

（1）基本描述

桥，完成地址、数据转换，控制信号的产生。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| WeCPU | I | 外设可写信号 |
| IRQ | I | 中断请求信号  0：不中断  1：请求中断 |
| PrAddr[29:0] | I | MIPS地址输入 |
| PrWD | I | MIPS数据输入 |
| DEV0\_RD[31:0] | I | 外设0数据输入 |
| DEV1\_RD[31:0] | I | 外设1数据输入 |
| DEV2\_RD[31:0] | I | 外设2数据输入 |
| DEV\_Addr[1:0] | O | 外设内寄存器选择 |
| PrRD[31:0] | O | 数据输出至MIPS |
| DEV\_WD[31:0] | O | 外设写入数据 |
| WeDEV0 | O | 外设0写信号使能  0：无效  1：有效 |
| WeDEV1 | O | 外设1写信号使能  0：无效  1：有效 |
| WeDEV2 | O | 外设2写信号使能  0：无效  1：有效 |
| HitDEV | O | 命中外设信号  0：未命中  1：命中 |
| HWInt[5:0] | O | 桥中断请求识别信号 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 写入连接 | 连接MIPS和外设写入数据 |
| 2 | 读取连接 | 选择外设并将其数据与MIPS连接 |
| 3 | 使能控制 | 根据地址选设置相应外设使能信号 |
| 4 | 请求连接 | 连接MIPS和外设中断请求信号 |
| 5 | 命中外设 | 根据地址判断是否命中外设并输入至MIPS |
| 6 | 外设寄存器 | 选择相应外设的寄存器 |

13、mips\_multi

（1）基本描述

MIPS处理器模块。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 |
| HitDEV | I | 命中外设信号  0：未命中  1：命中 |
| PrDIn[31:0] | I | 来自桥的数据输入 |
| HWInt[5:0] | I | 桥中断请求识别信号 |
| PrAddr[29:0] | O | 地址输出 |
| PrDOut[31:0] | O | 数据输出 |
| WeCPU | O | 外设写信号  0：无效  1：有效 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 连接模块 | 连接CP0等各个MIPS模块 |

14、tc

（1）基本描述

计时器模块，根据不同的计数模式，在计数为0后，计数器或者自动装填初值并重新倒计数，或者保持在0值直至初值寄存器再次被装载。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| CLK\_I | I | 时钟信号 |
| RST\_I | I | 复位信号 |
| ADD\_I[3:2] | I | 地址输入 |
| WE\_I | I | 写使能 |
| DAT\_I[31:0] | I | 32 位数据输入 |
| DAT\_O[32:0] | O | 32 位数据输出 |
| IRQ | O | 中断请求 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 模式0 | 当计数器倒计数为 0 后，计数器停止计数。  当初值寄存器再次被外部写入后，  初值寄存器值再次被加载至计数器，计数器重新启动倒计数。 |
| 2 | 模式1 | 当计数器倒计数为 0 后，初值寄存器值被自动加载至计数器，  计数器继续倒计数。 |

15、indev

（1）基本描述

输入设备。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 |
| WeDEV0 | I | 写入使能信号（无效） |
| DEV\_Addr[1:0] | I | 外设寄存器选择信号（无效） |
| indata[31:0] | I | 输入信号 |
| outdata[31:0] | O | 输出信号 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 输入 | 将indata输入至outdata |

16、outdev

（1）基本描述

输出设备。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 |
| WeDEV1 | I | 写入使能信号 |
| DEV\_Addr[1:0] | I | 外设寄存器选择信号 |
| indata[31:0] | I | 输入信号 |
| outdata[31:0] | O | 输出信号 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 输出 | 输入outdata |
| 2 | 改变输出 | outdata=indata |

三、指令描述

1. addu（与Project2相同）

2、subu（与Project2相同）

3、ori（与Project2相同）

4、lw（与Project2相同）

5、sw（与Project2相同）

6、beq（与Project2相同）

7、lui（与Project2相同）

8、j（与Project2相同）

9、addi（与Project2相同）

10、addiu（与Project2相同）

11、slt（与Project2相同）

12、jal（与Project2相同）

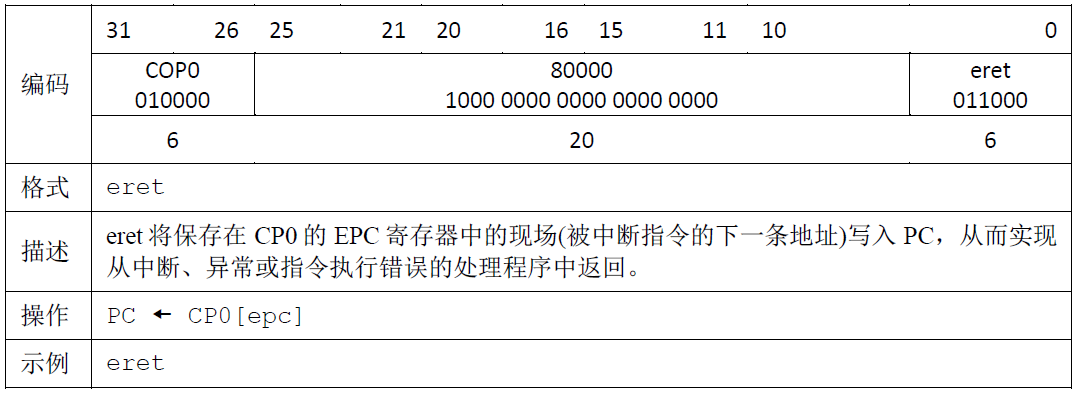
13、jr（与Project2相同）

14、lb（与Project2相同）

15、sb（与Project2相同）

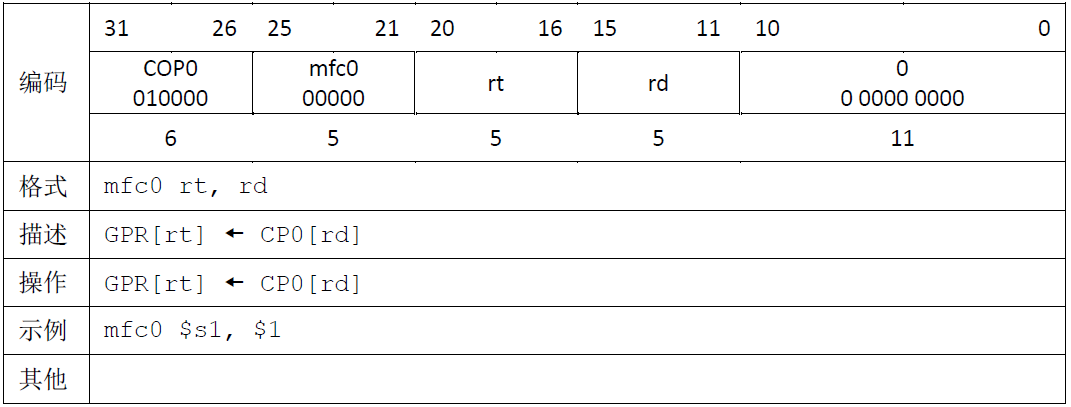
16、eret

功能：异常返回



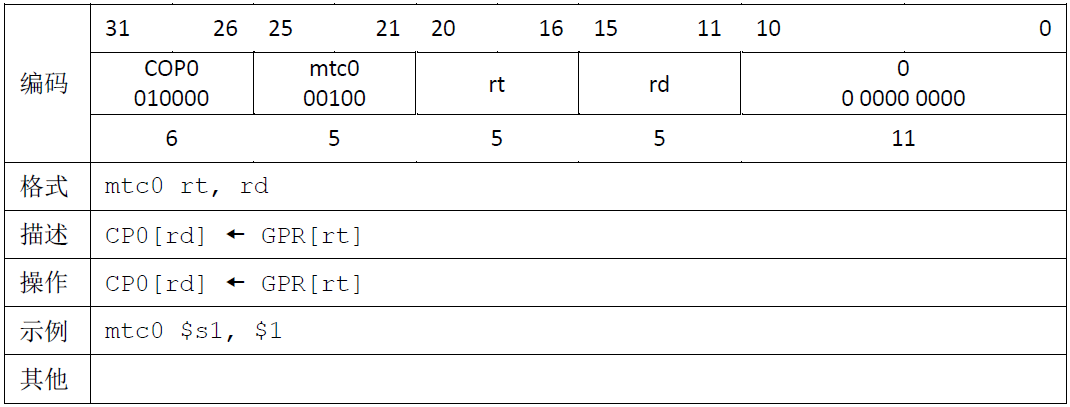
17、MFC0

功能：读CP0寄存器

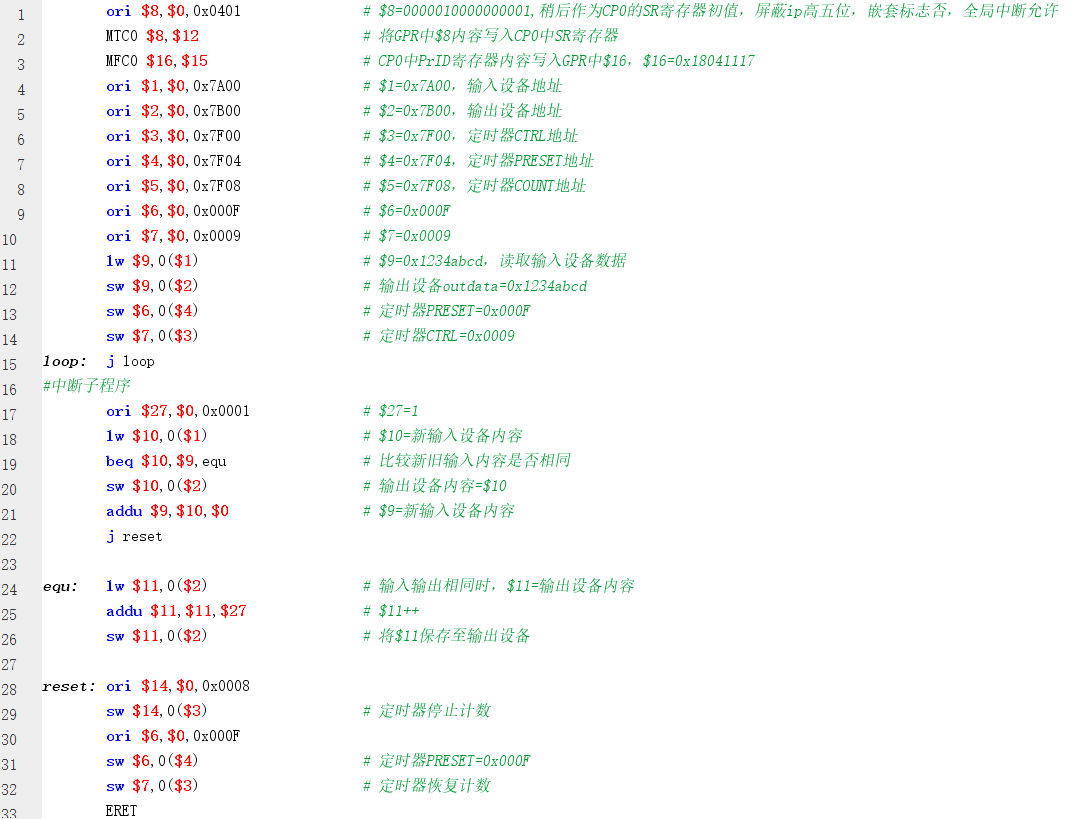


18、MTC0

功能：写CP0寄存器

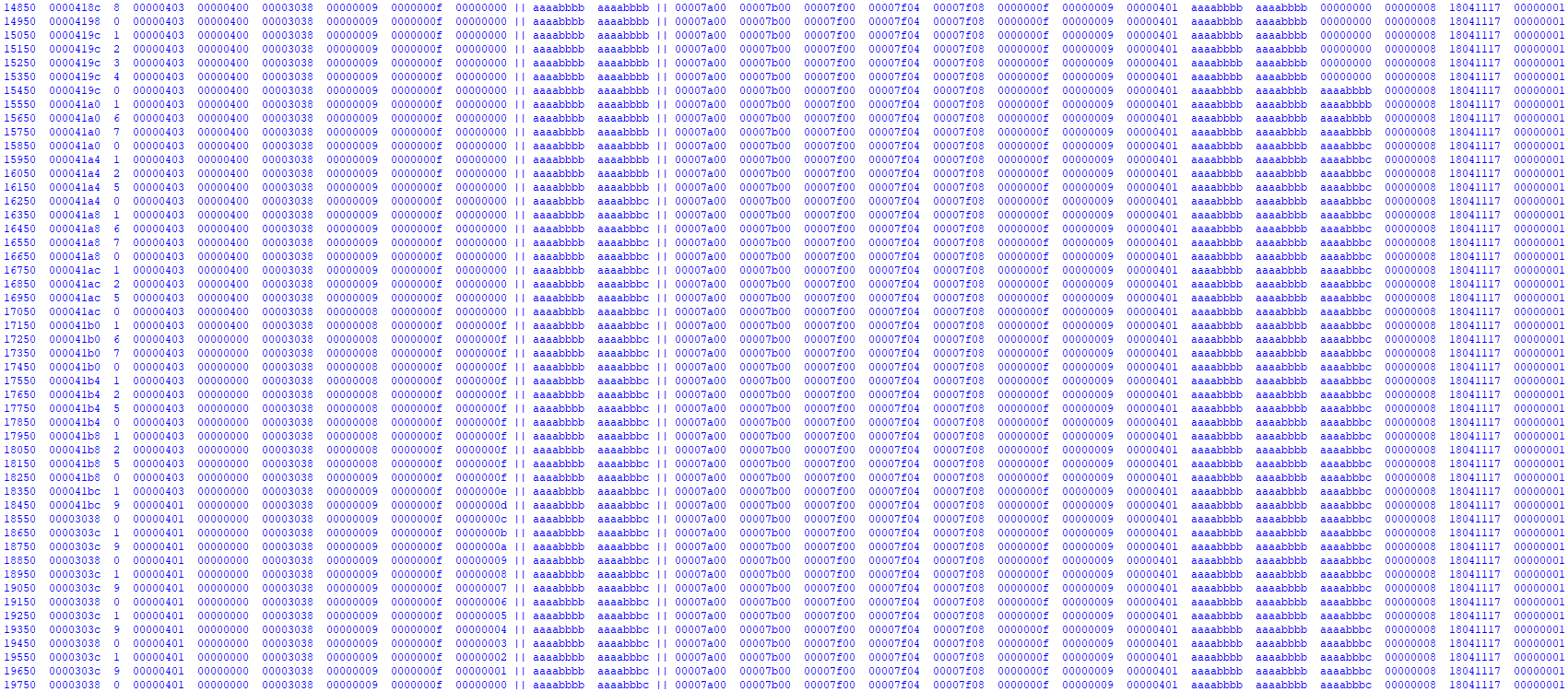
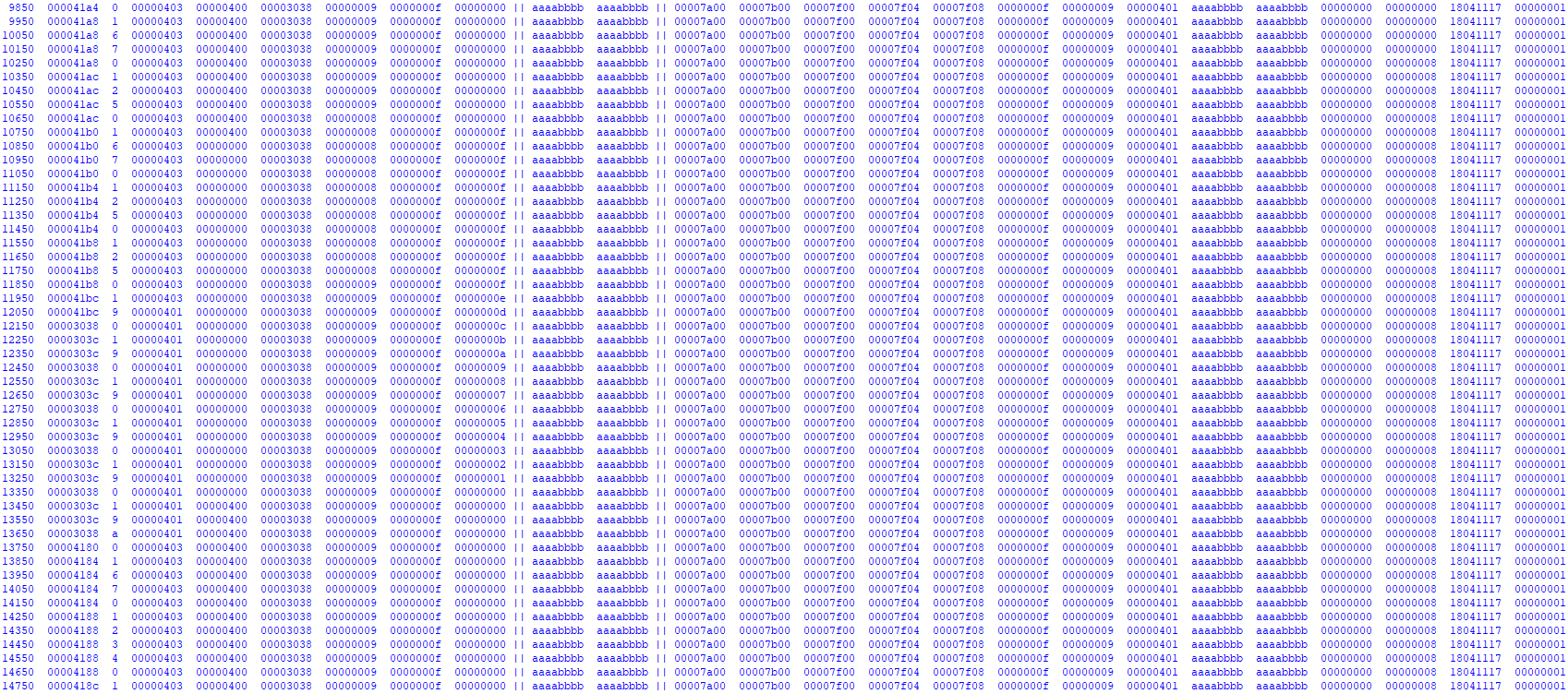
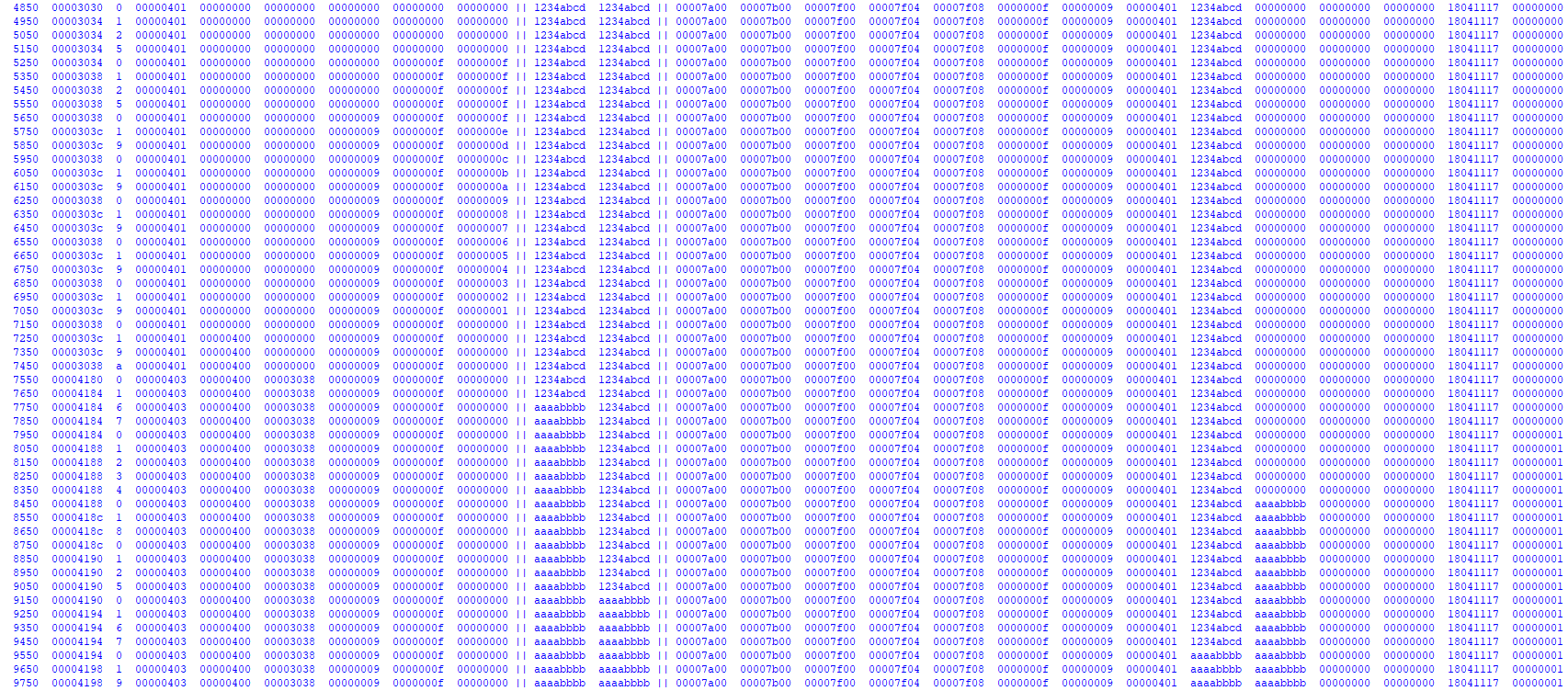
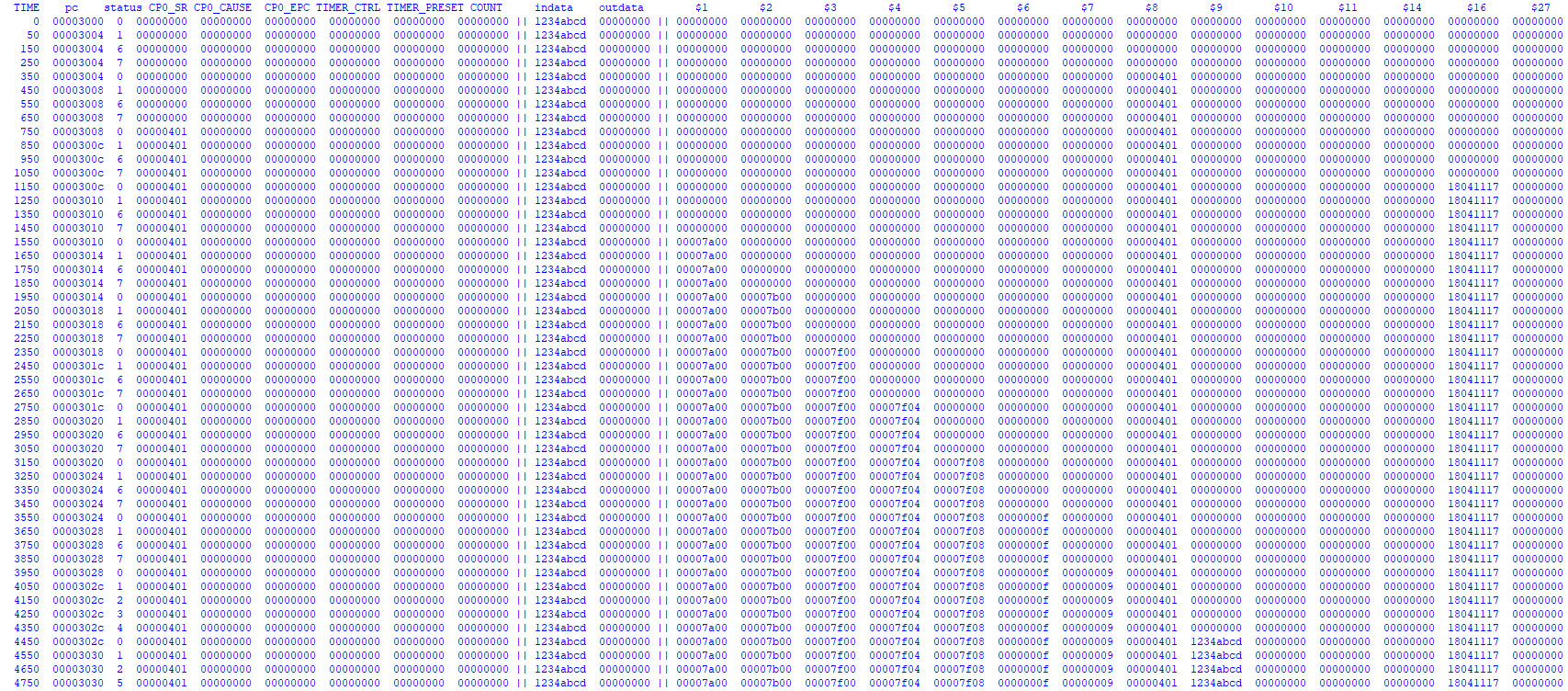


四、测试程序



五、modelsim测试

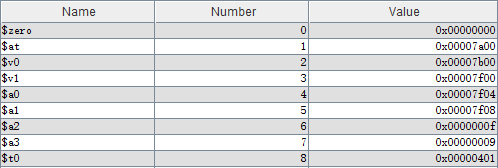
1、测试结果



1. 检验

|  |  |  |
| --- | --- | --- |
| MARS行数 | Modelsim时间 | 状态 |
| 14 | t=5650 | 计时器开始计时 |
| 16 | t=7550 | 进入中断，输入值indata改变，此次中断测试输入不等于输出情况 |
| 33 | t=12150 | 中断返回 |
| 16 | t=13750 | 进入中断，此次中断测试输入等于输出情况 |
| 33 | t=18550 | 中断返回，可结束测试 |

MARS运行结果：



Mars结果与Modelsim输出结果一致

六、收获体会

这次课设Project3难度较大，在多周期MIPS的基础上增加了CP0，桥和输入输出设备。最开始的几天毫无头绪，后来在老师的启发下，和同学的讨论中逐渐有了思路。P3不仅使我对MIPS中断有了一定的了解，也培养了我面对困难毫无头绪的情况下坚持思考，不轻言放弃的习惯。耗时6天做完P3时内心的喜悦和小小的成就感更给予了我面对新问题时更强大的内心。最终作为整个报告的结尾，我要感谢老师的辛苦付出，在这两周的小学期里无论是工作时间还是休息时间，老师都非常负责的为我们答疑解惑，谢谢老师，老师辛苦了！