**Veriloghdl 单周期，多周期处理器开发，mips微系统开发**

目录

[Project1 veriloghdl完成单周期处理器开发 4](#_Toc139318164)

[一，总体数据通路设计图 4](#_Toc139318165)

[二，各模块定义 5](#_Toc139318166)

[Pc模块 5](#_Toc139318167)

[Im模块 5](#_Toc139318168)

[Splitter模块 6](#_Toc139318169)

[Gpr模块 6](#_Toc139318170)

[Ext模块 9](#_Toc139318171)

[Npc模块 9](#_Toc139318172)

[Alu模块 10](#_Toc139318173)

[Dm模块 12](#_Toc139318174)

[Controller模块 13](#_Toc139318175)

[三，指令描述 15](#_Toc139318176)

[四，测试程序 19](#_Toc139318177)

[1，主要测试程序 19](#_Toc139318178)

[2，额外指令bltzal的测试程序 20](#_Toc139318179)

[五，测试结果 20](#_Toc139318180)

[1， mars结果 20](#_Toc139318181)

[2，modelsim仿真波形 22](#_Toc139318182)

[六，总结与心得体会 22](#_Toc139318183)

[Project2，veriloghdl多周期处理器开发 23](#_Toc139318184)

[一， 总体数据结构通路 23](#_Toc139318185)

[二，模块定义 24](#_Toc139318186)

[pc模块 24](#_Toc139318187)

[Controller模块 25](#_Toc139318188)

[Ir模块 27](#_Toc139318189)

[Datareg模块 28](#_Toc139318190)

[Rega/regb模块 28](#_Toc139318191)

[Sb模块 28](#_Toc139318192)

[Gpr模块 29](#_Toc139318193)

[二， 指令描述 30](#_Toc139318194)

[四，测试程序 32](#_Toc139318195)

[主要测试程序 32](#_Toc139318196)

[五，测试结果 33](#_Toc139318197)

[Mars结果 33](#_Toc139318198)

[Modelsim仿真波形 34](#_Toc139318199)

[六，总结与心得体会 35](#_Toc139318200)

[Project3 VerilogHDL完成MIPS微系统开发(支持设备与中断) 36](#_Toc139318201)

[一，总体数据通路设计 36](#_Toc139318202)

[二，模块定义 36](#_Toc139318203)

[Controller模块 37](#_Toc139318204)

[Npc模块 38](#_Toc139318205)

[Cp0模块 38](#_Toc139318206)

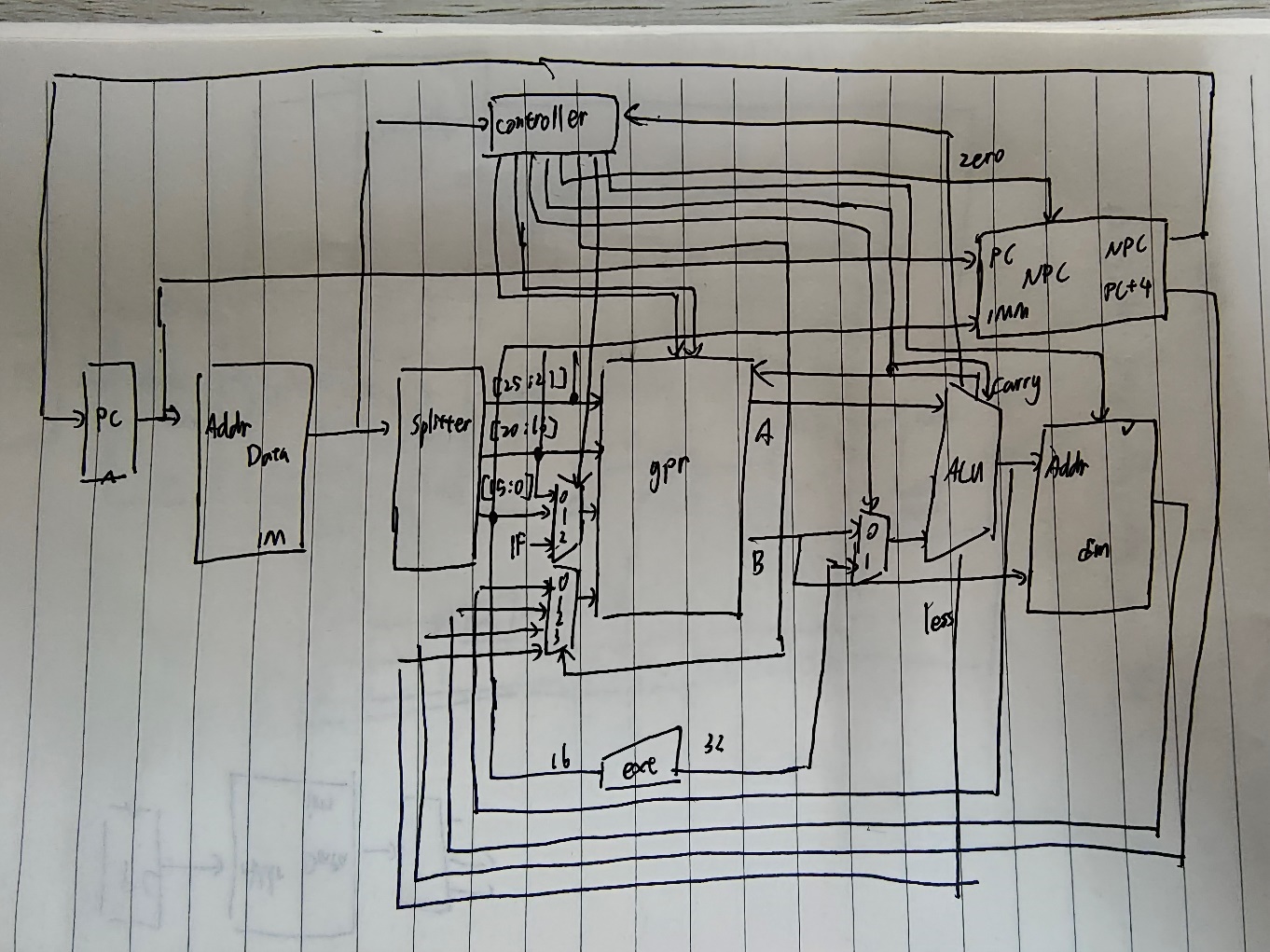
[Bridge模块 38](#_Toc139318207)

[Dev\_output模块 38](#_Toc139318208)

[Timer模块 38](#_Toc139318209)

# Project1 veriloghdl完成单周期处理器开发

## 一，总体数据通路设计图



参照ppt内容以及要求完成的指令，设计如图所示的总体数据通路。

指令集{addu，subu，ori，lw，sw，beq，lui，addi，addiu，slt，jal，j，jr}

需求分析：addu，subu需要将rs，rt地址的寄存器内容取出，分别进行加减计算，回写进rd地址的寄存器；ori指令需要将rs寄存器内容与符号扩展后的16位立即数相或，回写到rt地址寄存器；lw，sw需要访问存储器，地址为rs寄存器内容加符号扩展后的16位立即数，将rt寄存器写入读取内容或将地址存入rt寄存器内容；beq和bltzal指令将立即数符号扩展后低两位补零后与pc+4相加算出跳转地址；lui将16位立即数高位扩展后存入rt地址的寄存器；addi和addiu将rs寄存器内容和符号扩展后的立即数相加并存进rt地址的寄存器，若addi指令溢出则rt存0，且30号寄存器存1代表溢出；j和jal将pc+4的高四位与26位立即数拼接，低两位补零形成跳转地址，且jal指令将pc+4存在31号寄存器；jr指令读取rs寄存器内容并将其作为下一地址。

数据通路设计：

存储器：im存储指令，dm存储数据

寄存器组：32个32位寄存器形成的寄存器组，按照rs，rt内容读取内容，rt或rd写入内容

Ifu：pc和npc，npc负责计算下一条指令地址，pc把地址传给im

扩展器：按要求处理立即数

运算单元：按要求进行加，减，或等操作

控制器：

译码并发出正确信号将数据通路调整成每条指令所需的状态。

## 二，各模块定义

### Pc模块

1，基本描述

Pc负责接收npc发来的下一条指令地址，并在每个时钟信号上升沿把其输出给指令存储器。在收到复位信号rst后将发出的地址重置为32‘h000300。

2，模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [31:0]pcin | i | 输入npc模块的输出，下一条指令的地址 |
| clk | i | 时钟信号 |
| rst | i | 复位信号 |
| [9:0]pcout | o | 下一条指令的地址 |
| [31:0]pcout\_31 | o | 返回给npc用于计算pc+4的地址 |

3，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 输出下一条指令地址 | 当时钟信号上升沿，输出下一条指令的地址给im用于执行下一条指令 |
| 2 | 返回当前地址给npc | 将当前地址返回npc，用于计算pc+4 |
| 3 | 复位 | 当rst信号有效时，将当前地址重置为32’h00003000 |

### Im模块

1，基本描述

Im是指令存储器，用于读取存储在txt文件中的16进制指令并输出给controller和分线器splitter。

2，模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [9:0]addr | i | 指令的地址 |
| [31:0]dout | o | 根据输入地址读取出的指令 |

3，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 存储指令 | 使用$readmemh读取txt文件中存储的指令 |
| 2 | 输出指令 | 根据pc输出的地址取出对应的32位指令输出给controller与分线器splitter |

### Splitter模块

1，基本描述

接收im输出的32位指令并将其按照各指令码的分割方式分解成不同位宽并输出给其他元件使用。首先，r型指令需要rs，rt，rd，于是将32位指令的[25:21]，[20:16]，[15:11]位分别作为rs，rt，rd输出给gpr；其次，i型指令在需要rs，rt的基础上需要低16位构成立即数，于是将[15:0]输出作为立即数输出给ext；最后，部分跳转指令需要除了opcode外的低26位计算下一条指令地址，于是输出[25:0]给npc。

2，模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [31:0]data | I | 由pc传来的32位指令 |
| [4:0]rs | O | 指令[25:21]位，用于传给gpr做rs位位选 |
| [4:0]rt | O | 指令[20:16]位，用于传给gpr做rt位位选 |
| [4:0]rd | O | 指令[15:11]位，用于传给gpr做rd位位选 |
| [15:0]imm | O | 用于传给ext做扩展后作为alu第二操作数的备选 |
| [25:0]imm1 | O | 用于传给npc作跳转指令地址的计算 |

3，功能定义

|  |  |
| --- | --- |
| 序号 | 功能名称 |
| 1 | 将32位指令分割并传给其他元件 |

### Gpr模块

1，基本描述

寄存器组，拥有32个32位寄存器，可以根据位选信号和时钟信号选择读取和输出指定寄存器的信息，以及根据写使能的开关将指定信息写入指定寄存器。

2，模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| en | I | Gpr写使能，由controller传来 |
| rst | I | 复位信号 |
| [4:0]ra | I | 位选信号，用于指定第一个输出信号输出的寄存器的序号，连接由splitter传来的rs |
| [4:0]rb | I | 位选信号，用于指定第二个输出信号输出的寄存器的序号，连接由splitter传来的rt |
| [4:0]wr1 | I | 位选信号，由splitter传来。用于指定回写信息写入的寄存器的序号，连接由splitter传来的rd |
| [4:0]wr2 | I | 位选信号，由splitter传来。用于指定回写信息写入的寄存器的序号，连接由splitter传来的rd |
| [1:0]regdst | I | 用于选定写入地址，由controller传来。  00：写入wr1号寄存器  01：写入wr2号寄存器  10：写入31号寄存器。 |
| less | I | Slt指令专用信号，由alu传来。  1：a小于b，写回信号内容设置为32'h00000001  0：a不小于b，写回信号内容设置为32'h00000000 |
| carry | I | Addi溢出标志。配合addi信号使用。  1：加法计算溢出  0：加法计算没有溢出  由alu传来。若二者同时有效，则说明addi指令溢出，将30号寄存器的值设置为1. |
| addi | I | Addi信号标志，配合carry信号使用。  1：当前指令是addi指令 0：当前指令不是addi指令由controller传来。若二者同时有效，则说明addi指令溢出，将30号寄存器的值设置为1. |
| bltzal | I | Bltzal指令标志，由controller发出。  1：当前指令是bltzal指令  0：当前指令不是bltzal指令 |
| bltzal\_0 | I | bltzal\_0指令是否跳转标志。  1：rs寄存器内值小于0，满足跳转要求  0：rs寄存器内值不小于0，不满足跳转要求 |
| [31:0]aluout | I | 由alu传回的计算结果 |
| [31:0]dm\_out | I | 由dm传回的访存数据 |
| [31:0]pc\_4 | I | 由npc传回的pc+4 |
| [1:0]write\_sel | I | 决定写回数据的信号，由controller传来。  00：alu计算结果  01：dm访存结果  10：pc+4  11：slt指令专用，若less为1则将32'h00000001写回，否则将32'h00000000写回 |
| [31:0]busa | O | Rs寄存器的值 |
| [31:0]busb | O | Rt寄存器的值 |

3，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择地址 | 根据选择信号选择读取和写入的寄存器地址 |
| 2 | 选择数据 | 根据选择信号选择写入寄存器的数据 |
| 3 | 输出数据 | 读出对应地址寄存器的存储数据并输出 |
| 4 | 处理溢出 | 根据当前指令是否为addi以及溢出标志是否有效来处理addi的溢出 |
| 5 | 处理slt | 根据当前指令是否为slt以及小于标志是否有效来决定写入数据 |
| 6 | 处理bltzal | 根据当前指令是否为bltzal以及小于标志是否有效来决定是否写入 |

### Ext模块

1，基本描述

立即数扩展模块，可根据指令需求将16位立即数进行零扩展，符号扩展及高位扩展。

2，模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [15:0]din | I | 16位立即数，由splitter传来 |
| [1:0]extop | I | 扩展方式选择信号  00：零扩展  01：符号扩展  10：高位扩展 |
| [31:0]dout | O | 输出扩展完后的32位结果 |

3，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 扩展立即数 | 将传进来的16位立即数根据扩展信号选择的扩展方式扩展为32位并输出。 |

### Npc模块

1，基本描述

计算下一条指令的地址并输出给pc。

2，模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [25:0]imm1 | I | 26位立即数，用于在部分跳转指令中组成下个地址用 |
| [31:0]pc | I | 当前指令地址，由pc传来  用于计算pc+4 |
| [31:0]gpr | I | gpr中寄存器存储的内容。 |
| [1:0]pc\_sel | I | 输出选择信号  00：输出pc+4  01：beq指令和bltzal指令用，若跳转标志有效则输出跳转后地址，跳转信号无效就输出pc+4  10：j，jal指令用，输出将pc+4，imm1拼接后的地址  11：jr指令用，输出gpr中寄存器中存储的地址。 |
| zero | I | 两操作数相等信号  0：不相等，不跳转  1：相等，跳转  用于beq指令判断是否跳转 |
| bltzal | I | Bltzal指令标志信号  0：当前指令不是bltzal指令  1：当前指令是bltzal指令 |
| bltzal\_0 | I | Bltzal指令跳转标志信号  0：不满足跳转条件，不跳转  1：满足跳转条件，跳转 |
| [31:0]npc | O | 下一条指令地址，输出给pc |
| [31:0]pc\_4 | O | Pc+4的值，输出给gpr |

3，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 计算地址 | 根据指令需求计算下一条指令地址并输出给pc |
| 2 | 输出pc+4 | 将pc+4的值输出给gpr，用于在jal和bltzal指令保存下一条指令地址。 |

### Alu模块

1，基本描述

计算模块，将传进来的第一操作数与第二操作数按照需求进行加减或运算并输出。

2，模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [31:0]ina | I | Gpr输出数据1的值，来自gpr |
| [31:0]inb1 | I | Gpr输出数据2的值，来自gpr |
| [31:0]imm | I | 扩展后的16位立即数，来自ext |
| alusrc | I | 第二操作数选择信号  0：inb1  1：imm  来自controller |
| addi | I | Addi指令标志信号  0：当前指令不是addi信号  1：当前指令是addi信号  来自controller |
| [1:0]aluctr | I | 运算方式选择信号  00：加法  01：减法  10：或运算  11：判断是否满足bltzal跳转条件  由controller传来 |
| [31:0]out\_32 | O | 计算结果，传给gpr做写回信息 |
| [9:0]out\_10 | O | 计算结果低10位，传给dm做地址 |
| zero | O | Beq跳转标志信号，  0：第一操作数和第二操作数不相等  1：第一操作数和第二操作数相等 |
| carry | O | Addi溢出信号  0：addi未溢出  1：addi溢出  当溢出时，alu计算结果变为32，h00000000。 |
| less | O | Slt小于信号  0：第一操作数不小于第二操作数  1：第一操作数小于第二操作数 |
| bltzal\_0 | O | Bltzal跳转信号  0：第一操作数不小于0，不跳转  0：第一操作数小于0，跳转  传给gpr和npc |

3，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 数据选择 | 根据controller传来的信号选择第二操作数 |
| 2 | 运算并输出 | 根据选择的运算类型计算两操作数并输出 |
| 3 | 判断beq是否跳转 | 根据第一操作数和第二操作数的减法结果判断beq是否跳转。若两数减法结果为零说明两数相等，相等标志置1，否则置0. |
| 4 | 判断addi是否溢出 | 根据第一操作数和第二操作数的加法结果判断是否溢出。  先判断操作数正负，若两个操作数符号位都为0而运算结果符号位为1说明溢出；若两个操作数符号位都为1而运算结果符号位为0说明溢出。上述两种情况溢出标志置1，否则保持为0。 |
| 5 | 判断slt是否小于 | 根据第一操作数和第二操作数的减法运算结果的符号位判断是否小于，符号位为1则小于，符号位为0则不小于 |
| 6 | 判断bltzal是否跳转 | 根据第一操作数的符号位判断，若符号位为1则说明第一操作数小于零，标志置1，否则置0. |

### Dm模块

1，基本描述

存储器模块，包含1024个8位存储单元，用于存储和读取数据。

2，模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [9:0]addr | I | 访问地址，由alu传来 |
| [31:0] din | I | 写入数据，来源是gpr的第二操作数 |
| we | I | 写使能 |
| clk | I | 时钟信号 |
| [31:0] dout | O | 读出数据 |

3，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读取数据 | 读出地址中存储的一个字的数据并输出 |
| 2 | 写入数据 | 将传输进来的一个字的数据写入指定地址位置 |

### Controller模块

1，基本描述

Controller模块是根据im传来的指令进行译码，并根据指令需要向各元件发出控制信号构建正确数据通路的元件。

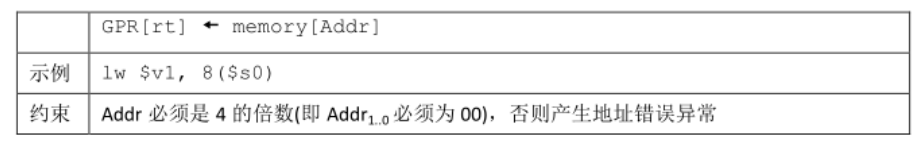
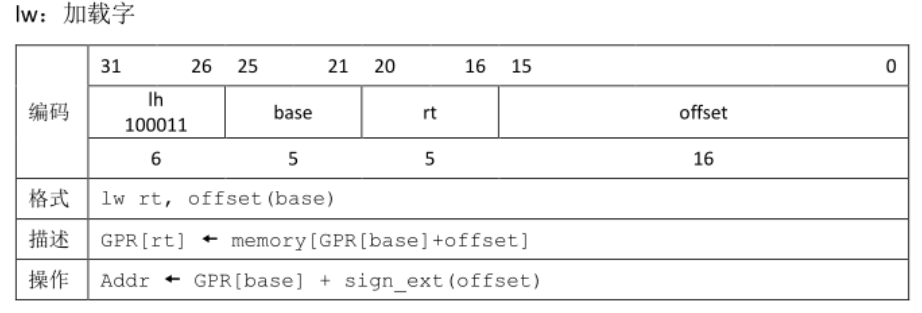
2，模块接口

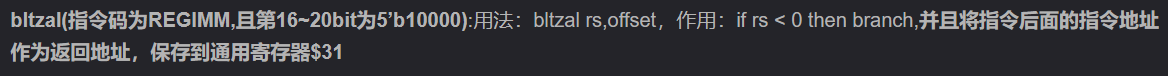
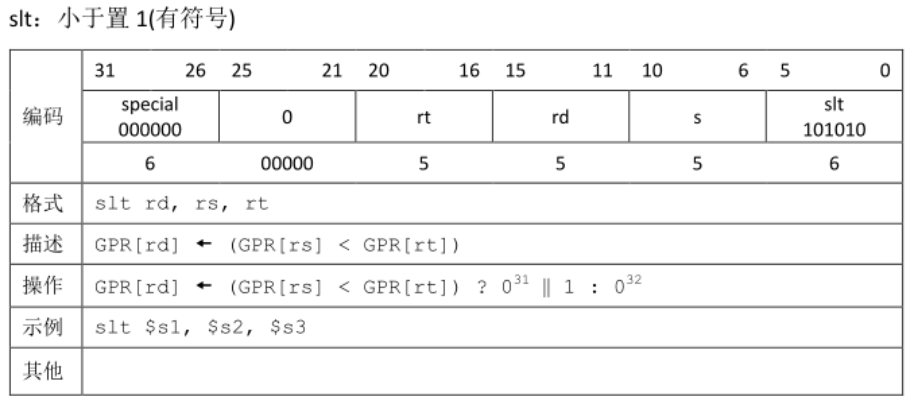
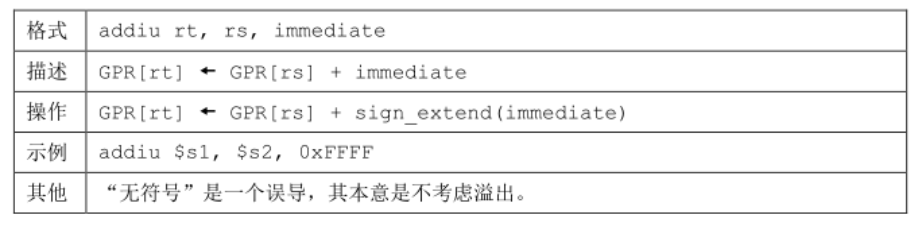
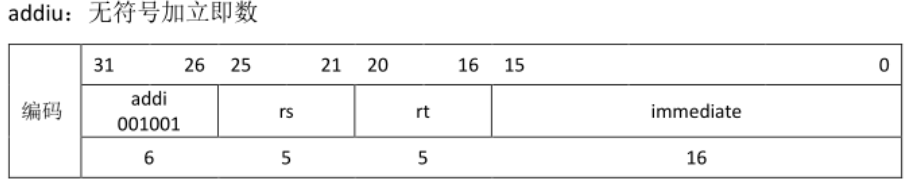
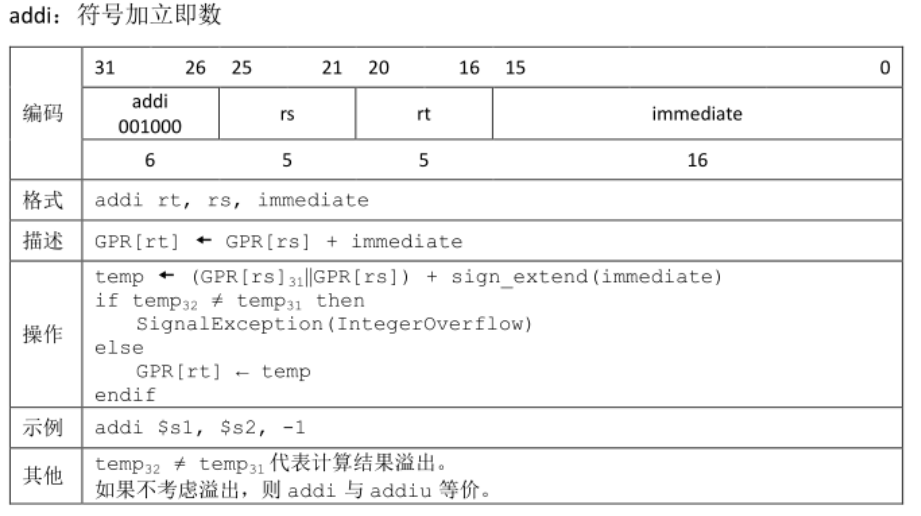
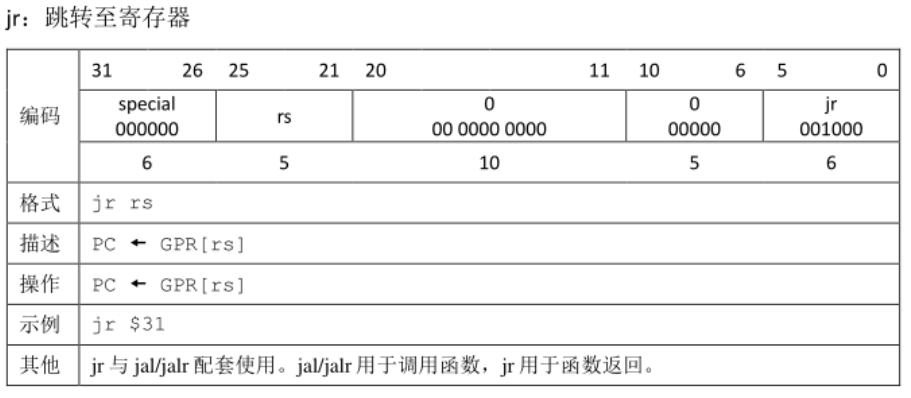
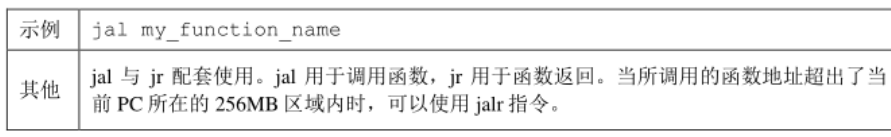
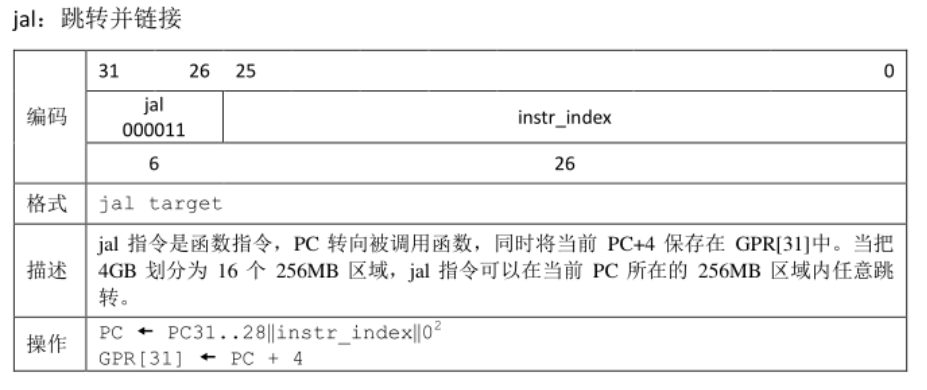
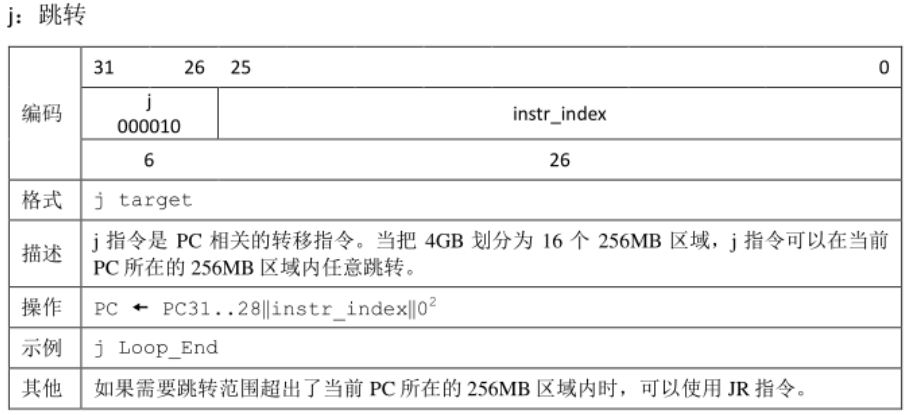
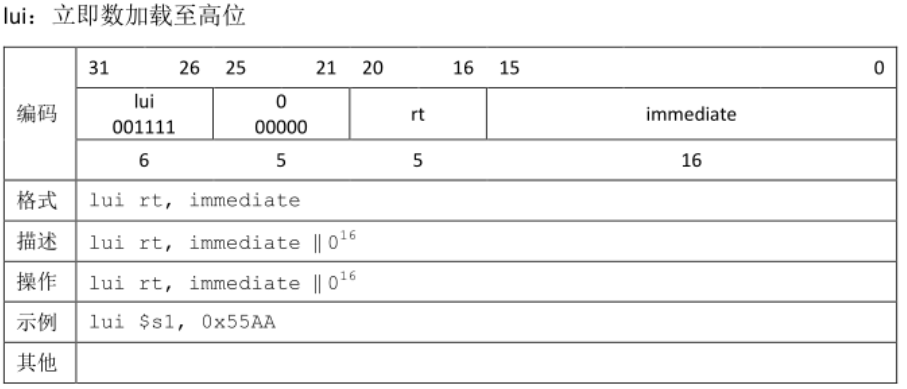
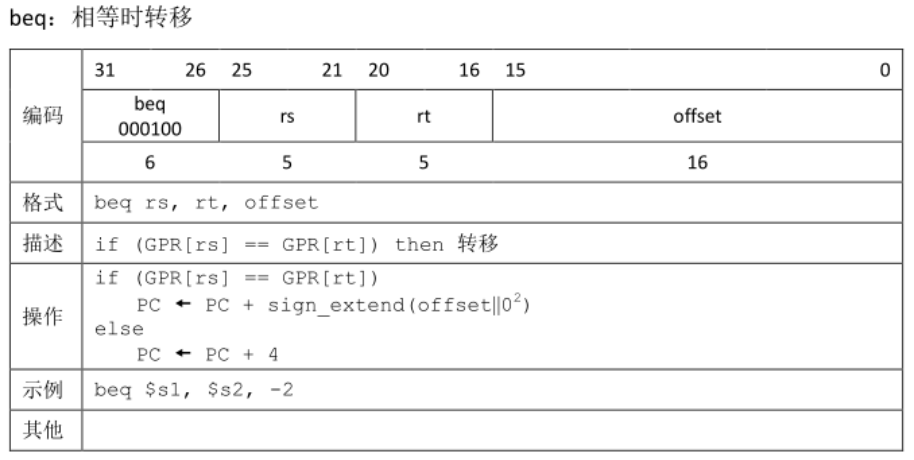
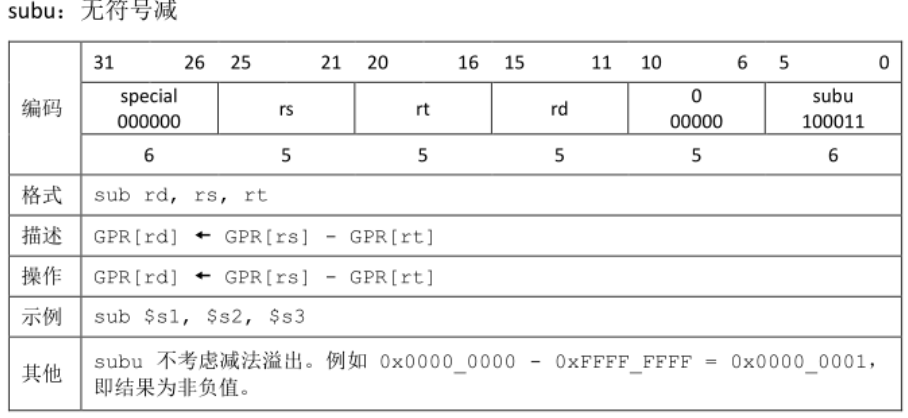
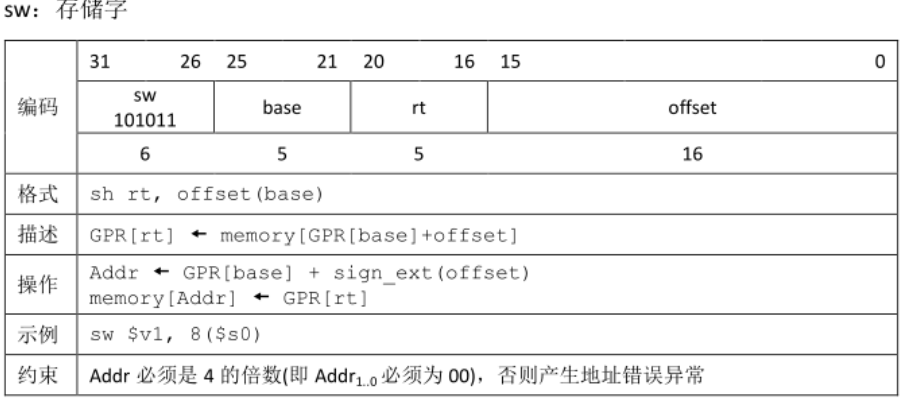
|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [31:0]din | I | 32位指令，由im传输而来，用于译码。 |
| [1:0]regdst | O | 用于选定gpr写入地址，传给gpr。  00：写入wr1号寄存器  01：写入wr2号寄存器  10：写入31号寄存器。 |
| memwr | O | Dm写使能，传给dm。 |
| [1:0]write\_sel | O | 决定gpr写回数据选谁的信号，传给gpr。  00：alu计算结果  01：dm访存结果  10：pc+4  11：slt指令专用，若less为1则将32'h00000001写回，否则将32'h00000000写回 |
| [1:0]pc\_sel | O | npc输出选择信号，传给npc  00：输出pc+4  01：beq指令和bltzal指令用，若跳转标志有效则输出跳转后地址，跳转信号无效就输出pc+4  10：j，jal指令用，输出将pc+4，imm1拼接后的地址  11：jr指令用，输出gpr中寄存器中存储的地址。 |
| [1:0]aluctr | O | alu运算方式选择信号  00：加法  01：减法  10：或运算  11：判断是否满足bltzal跳转条件  传给alu |
| alusrc | O | alu第二操作数选择信号  0：inb1  1：imm  传给alu |
| [1:0]extop | O | 扩展方式选择信号  00：零扩展  01：符号扩展  10：高位扩展  传给ext |
| addi | O | 判断当前指令是否为addi  0不是  1是  传给gpr与alu |
| en | O | Gpr写使能  传给gpr |
| bltzal | O | 判断当前指令是否为bltzal  0不是  1是  传给gpr和npc |

3，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能定义 | 功能描述 |
| 1 | 译码 | 将指令高六位和低六位取出，判断当前指令种类 |
| 2 | 产生控制信号 | 根据译码结果选择适当控制信号发给各元件，构建正确数据通路 |

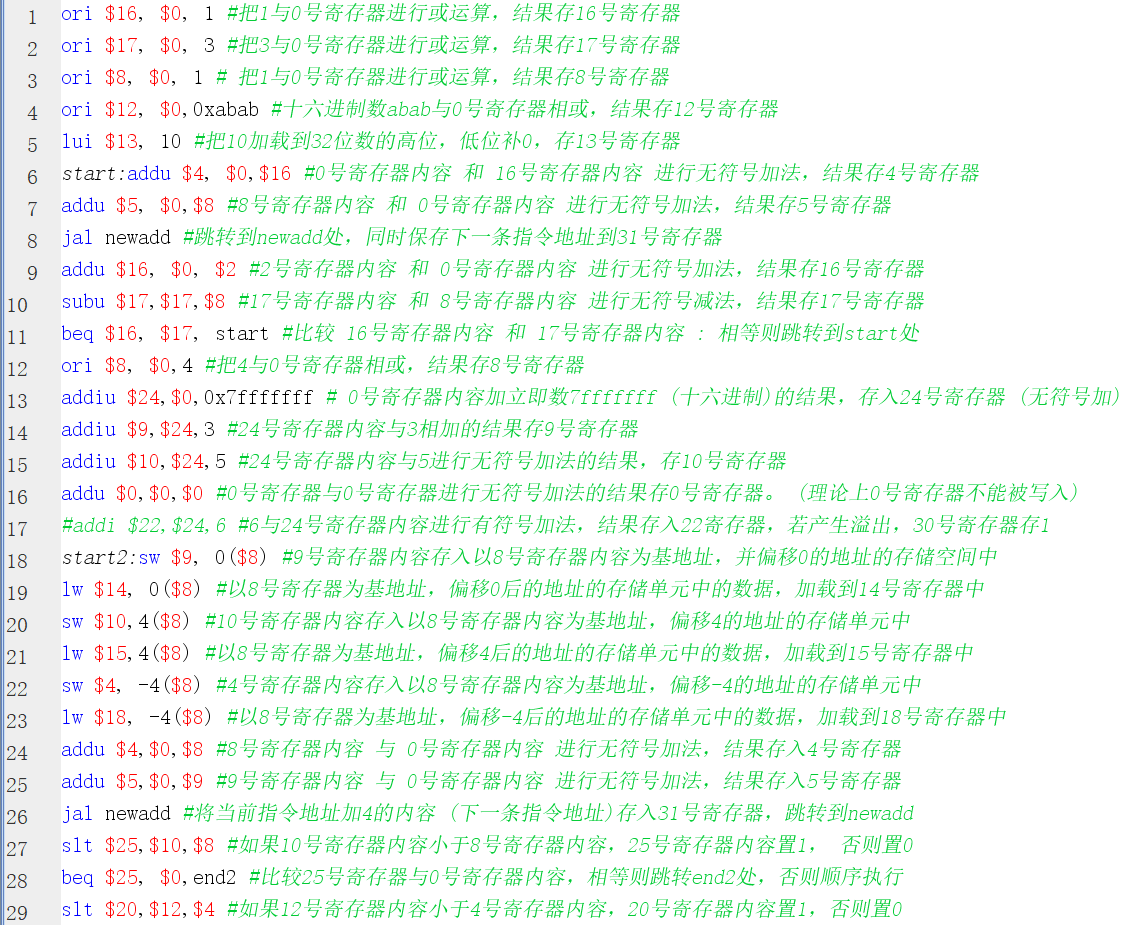
## 三，指令描述

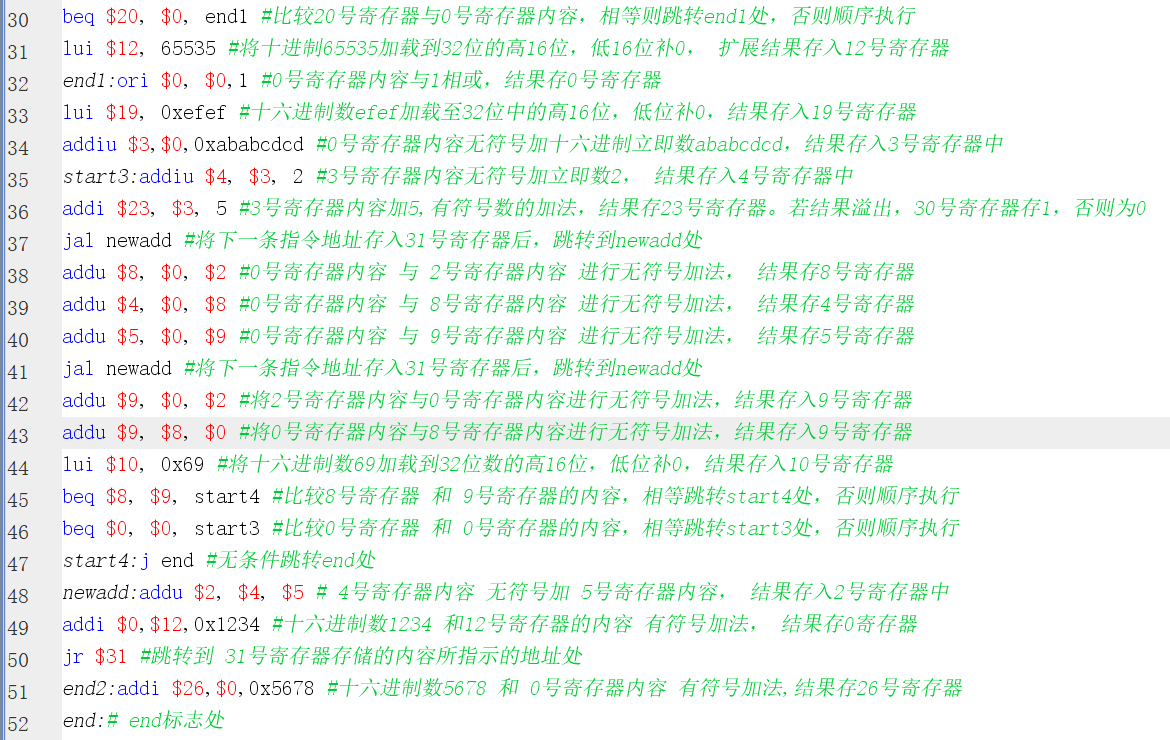




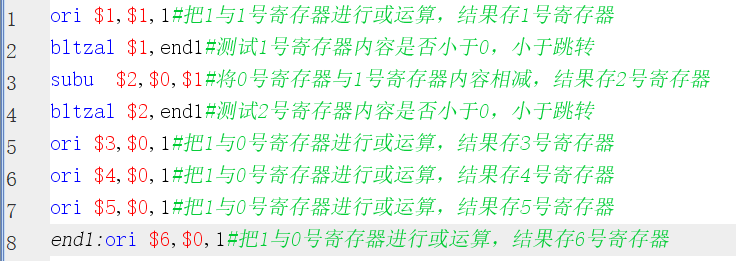
## 四，测试程序

### 1，主要测试程序





### 2，额外指令bltzal的测试程序

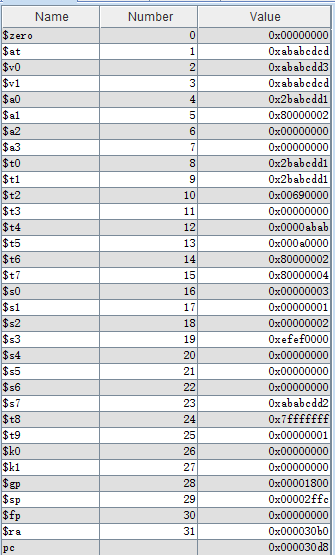


## 五，测试结果

### mars结果

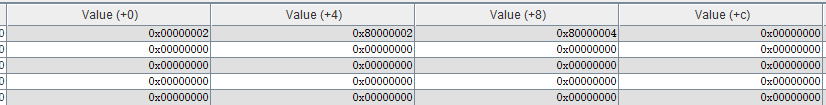
1. 主要测试程序

寄存器存储内容：



应注意的是，mars中28和29号寄存器内容与程序运行无关，故在modelsim仿真中28和29号寄存器内容应是0。

存储器存储内容：



由于测试程序只在前三个字有写入内容，故后面部分省略。

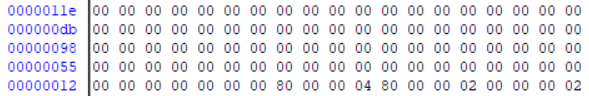
### 2，modelsim仿真波形



以图中黄线所在指令为例，指令地址是00003044，指令码是23160006，即addi $22,$24,6；addi标志置1；立即数为16‘h0006，第一操作数是7fffffff，扩展后得出第二操作数为00000006，加法结果溢出，溢出位置1，alu输出变为0；npc为00003048。

Modelsim寄存器与存储器结果





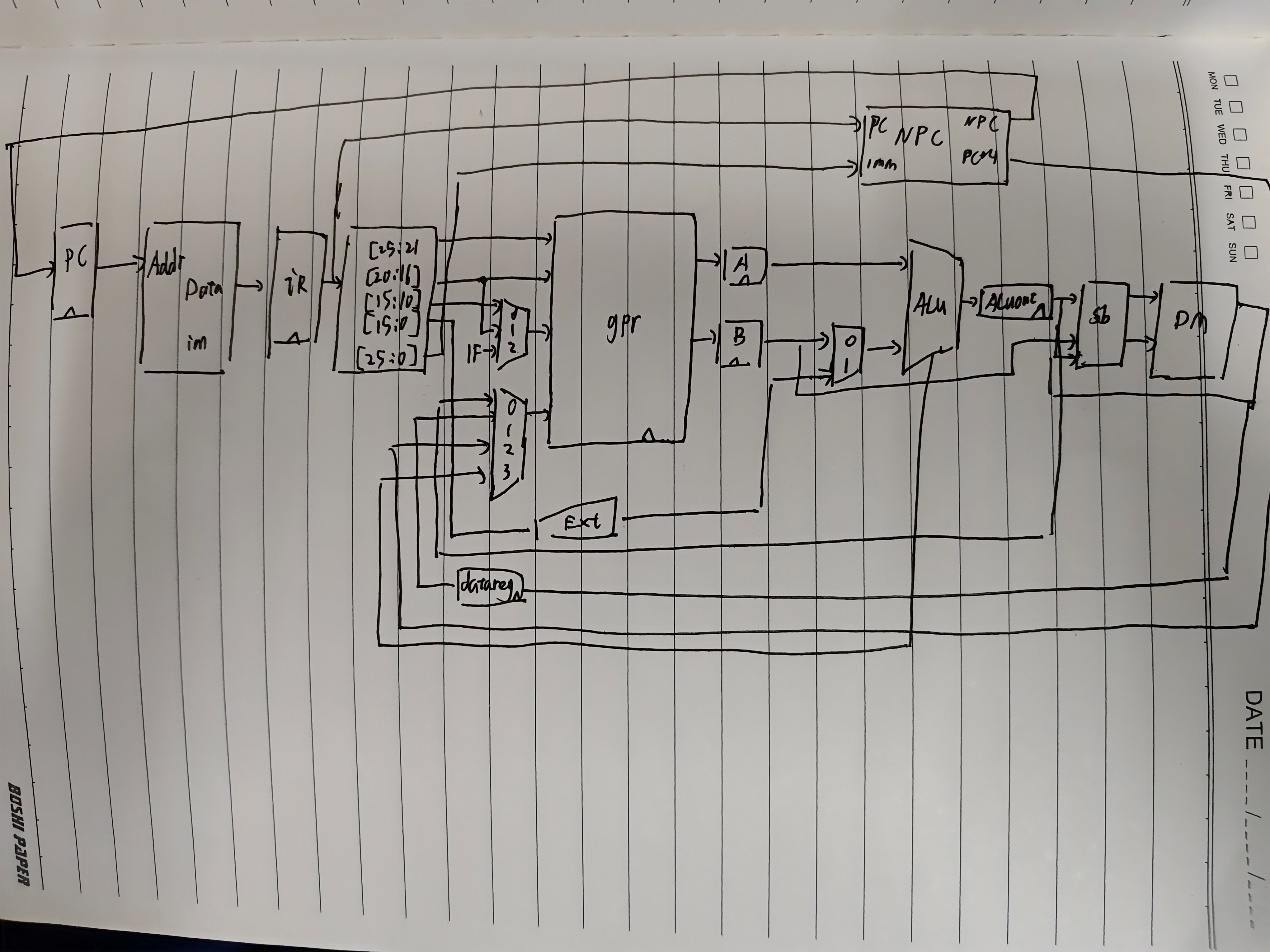
与mars仿真结果对比得出，指令全部正确执行。

## 六，总结与心得体会

对我个人而言，总是经过实践而学得的知识更容易理解。在计组课上，我对于第五单元部分只能说是一知半解，直到自己亲手写大作业和课设时才会注意到那些平时自己忽略的问题，和本来就不到位的理解。在面对问题并解决问题的过程中，我真正从到到尾捋清了思路，彻底理解了每条指令的执行过程，同时也对cpu的每个元件与其应负责的功能产生了清晰明了的认识。

# Project2，veriloghdl多周期处理器开发

## 总体数据结构通路



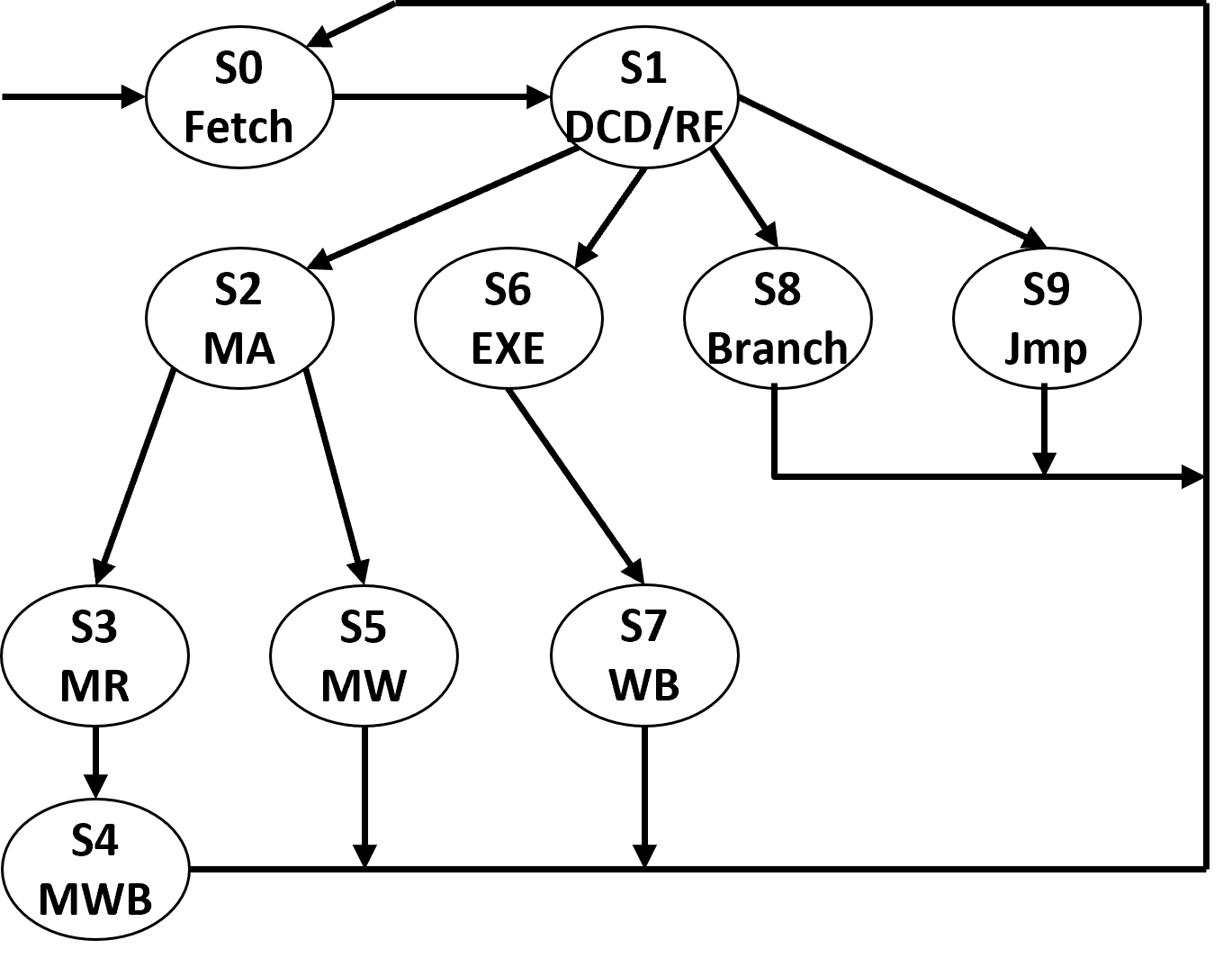
多周期在单周期的基础上将指令执行阶段分为1，取指，2，译码+寄存器操作，3，alu执行，4，访存，5，回写，五个阶段，不同指令的执行阶段不同，多周期cpu会根据指令的要求跳过不需要的阶段，以此减少指令耗费时间，提高性能。

实现方法：将整个数据回路分割为五个部分，在im与controller和gpr间添加指令寄存器；为gpr的输出各添加一个寄存器；为alu输出添加一个寄存器；为dm回写数据添加一个寄存器。

指令集{addu，subu，ori，lw，sw，beq，lui，addi，addiu，slt，jal，j，jr，lb，sb}

新增指令：lb，sb需要访问存储器，地址为rs寄存器内容加符号扩展后的16位立即数，将rt寄存器的一个字节的数据写入读取内容或将地址的一个字节的数据存入rt寄存器；为了支持sb指令，我为dm添加了sb模块。

多周期分析：上述所有指令都需要经历1取指和2译码、寄存器操作部分。Lw，sw，lb，sb指令需要寄存器操作，分为一组；其中lw和lb需要经历完整的执行，访存，写回阶段；sw和sb需要经历执行和访存，不需要写回；Addu、subu、ori、lui、addi、addiu、slt指令需要经历执行和写回阶段；beq、jr需要经历执行阶段；j和jal需要执行阶段，其中jal的改写npc与将pc+4写入寄存器是在同一阶段。完成以上分析，就可以按照不同指令的不同阶段进行状态机的设计。额外指令我完成了将controller状态机的九个状态合并成五个状态的任务。

九个状态的状态机如图所示。

## 二，模块定义

本模块只叙述相比project1新增的部分。

### pc模块

1，基本描述

Pc负责接收npc发来的下一条指令地址，并在每个写信号打开的时钟信号上升沿将输入信号赋值给输出信号。

2，模块新增接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Pcen | i | Pc模块的写使能，在取指阶段打开 |

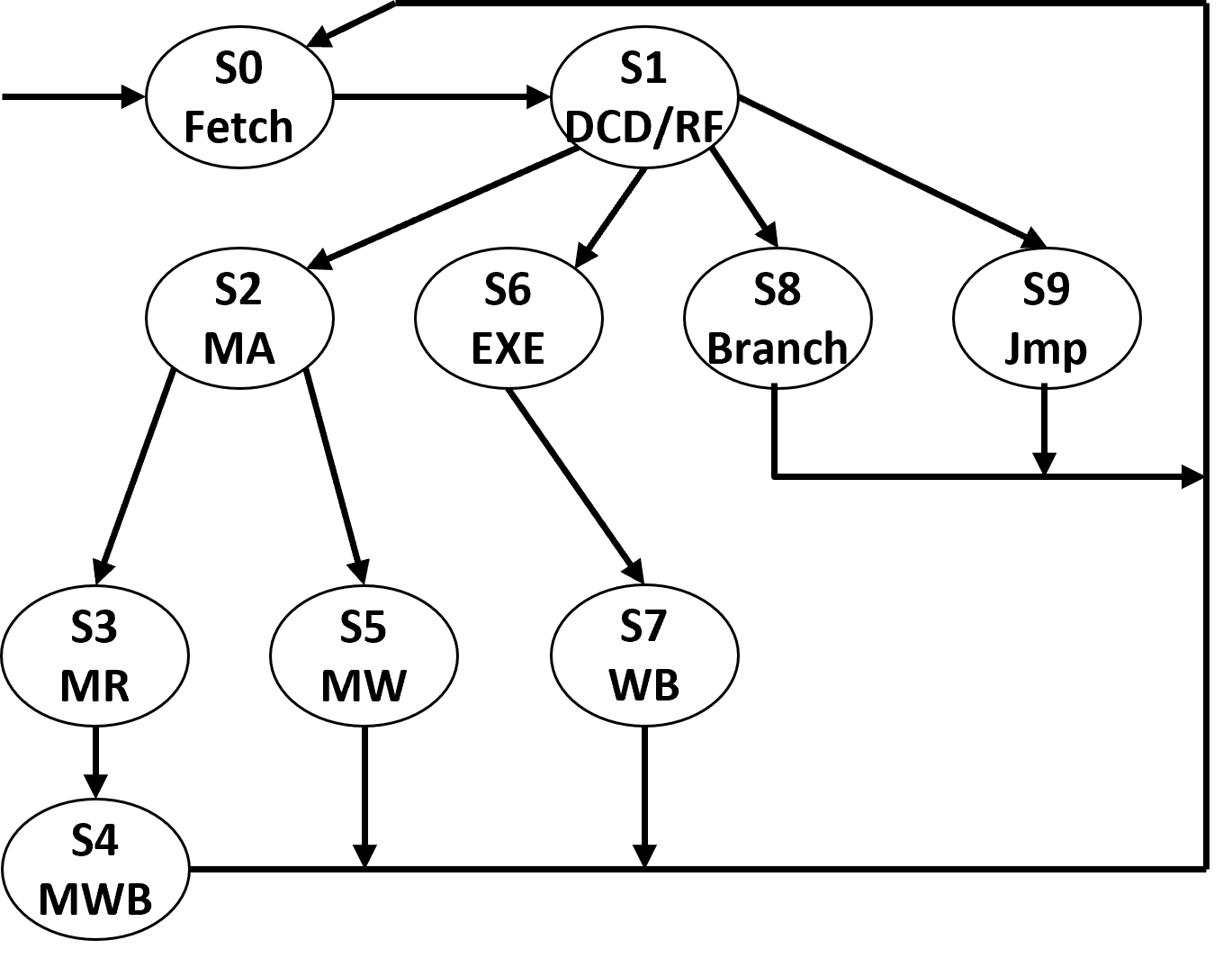
3，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 存储指令 | 在上一条指令取指阶段完成后，pc的值就更新为下一条指令的地址，一直将其储存到本次指令的取指阶段完成，再换成存储下一条指令地址，以此类推。 |
| 2 | 返回当前地址给npc | 将当前地址返回npc，用于计算pc+4 |
| 3 | 复位 | 当rst信号有效时，将当前地址重置为32’h00003000 |
| 4 | 输出下一条指令地址 | 当时钟信号上升沿且pc写使能有效，输出下一条指令的地址给im用于执行下一条指令 |

### Controller模块

1，基本描述

Controller模块是根据im传来的指令进行译码，并根据指令需要向各元件发出控制信号构建正确数据通路的元件。完成译码功能。



如图设计状态机。所有指令都需要经历s0取指和s1译码、寄存器操作部分。Lw，sw，lb，sb指令需要寄存器操作，走s2；其中lw和lb需要经历完整的执行，访存，写回阶段，故走s3，s4，最后回到s0；sw和sb需要经历执行和访存，不需要写回，故走s5，最后回到s0；Addu、subu、ori、lui、addi、addiu、slt指令需要经历执行和写回阶段，故走s6，s7，最后回到s0；beq、jr需要经历执行阶段，故走s8，最后回到s0；j和jal需要执行阶段，走s9，最后回到s0。其中jal的改写npc与将pc+4写入寄存器是在同一阶段。

额外功能将九个状态机合成五个，以下是我的思路：

lw和lb需要经历完整的执行，访存，写回阶段，故走s2执行，s3访存，s4写回，最后回到s0；sw和sb需要经历执行和访存，不需要写回，走s3执行，s4访存，然后回到s0；Addu、subu、ori、lui、addi、addiu、slt指令需要经历执行和写回阶段，走s3执行，s4写回；beq、jr、j和jal都只需要一个阶段，故直接走s4，然后回到s0。

Controller译码经过如下几个阶段：1，译码阶段根据高六位和低六位判断当前是什么指令，

2，根据当前指令决定状态机走向 3，配合状态机走向决定各控制信号。

1. 模块新增接口

九个状态版本：

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| pcwr | O | Pc写信号  1：在s0阶段、在jal和j指令的s9阶段，在jr的s8阶段，在beq的相等标志生效的s8阶段有效。  0：其余阶段均无效。 |
| irwr | O | Ir写使能，  1：s0阶段  0：其余所有情况均无效 |
| lb1 | O | Lb指令标志  0：当前指令不是lb指令  1：当前指令是lb指令 |
| sb1 | O | sb指令标志  0：当前指令不是sb指令  1：当前指令是sb指令 |
| [3:0]s | O | 输出当前状态机前一个状态 |
| [3:0]ns | O | 输出当前状态机状态 |
| En | O | Gpr写使能，回写阶段打开  1：Addu、subu、ori、lui、addi、addiu、slt指令的s7阶段，lw和lb指令的s4阶段，jal的s9阶段。 |
| Memwr | O | Dm写使能  1：Sw，sb的s5阶段  0：其他阶段 |
| pc\_sel[0] | O | Pc下个地址计算方式选择信号的低位  1：beq和jr指令的不是s0的阶段  0：其他情况 |
| pc\_sel[1] | O | Pc下个地址计算方式选择信号的高位  1：j和jal指令的不是s0的阶段  0：其他情况 |

五个状态版本：

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| pcwr | O | Pc写信号  1：在s0阶段、在jal和j指令的s4阶段，在jr的s4阶段，在beq的相等标志生效的s4阶段有效。  0：其余阶段均无效。 |
| irwr | O | Ir写使能，  1：s0阶段  0：其余所有情况均无效 |
| lb1 | O | Lb指令标志  0：当前指令不是lb指令  1：当前指令是lb指令 |
| sb1 | O | sb指令标志  0：当前指令不是sb指令  1：当前指令是sb指令 |
| [3:0]s | O | 输出当前状态机前一个状态 |
| [3:0]ns | O | 输出当前状态机状态 |
| En | O | Gpr写使能，回写阶段打开  1：Addu、subu、ori、lui、addi、addiu、slt指令的s7阶段，lw和lb指令的s4阶段，jal的s9阶段。 |
| Memwr | O | Dm写使能  1：Sw，sb的s4阶段  0：其他阶段 |
| pc\_sel[0] | O | Pc下个地址计算方式选择信号的低位  1：beq和jr指令的不是s0的阶段  0：其他情况 |
| pc\_sel[1] | O | Pc下个地址计算方式选择信号的高位  1：j和jal指令的不是s0的阶段  0：其他情况 |

3，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能定义 | 功能描述 |
| 1 | 译码 | 将指令高六位和低六位取出，判断当前指令种类 |
| 2 | 产生控制信号 | 根据译码结果和当前状态选择适当控制信号发给各元件，构建正确数据通路 |
| 3 | 构建状态机 | 按设计好的状态机构建状态，并根据不同的指令确定下一个状态。 |

### Ir模块

1，基本描述

指令寄存器，将im传来的指令寄存，并在每个s1开始阶段输出变化为本次执行的指令，并保持到下条指令的s1阶段变化为输出下条指令，以此类推。负责取指阶段。分割数据通路。

2，模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| en | I | 写使能  1：允许写入  0：不允许写入 |
| [31:0]in | I | 输入的指令，由im传来。 |
| [31:0]out | O | 输出指令 |

3，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 寄存指令 | 除了每条指令s0阶段都不许写入，在每条指令s1阶段更新指令，是取指阶段必要的元件。 |

### Datareg模块

1，基本描述

Dm输出的寄存器，负责分割数据通路并寄存dm输出，在正确的阶段释放dm输出。在写回阶段用到。

2，模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| [31:0] in | I | Dm输出数据 |
| [31:0] dout | O | 输出dm输出数据 |

3，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 寄存数据 | 寄存dm输出数据，在正确的阶段将其输出 |

### Rega/regb模块

1，基本描述

aReg & bReg模块实现的是同样的功能，切分数据通路并寄存两操作数。当两个输出数据写入两个模块的寄存器时，代表译码阶段结束。因为两者一模一样，这里只展示rega

2，模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| [31:0]busa | I | 第一操作数输出，来自gpr |
| [31:0]ra\_out | O | 寄存器输出，在时钟信号上升沿更新。 |

3，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 寄存数值 | 将gpr处理完成的两个数据分别寄存，当下个时钟信号上升沿将寄存器值更改为输入值。 |

### Sb模块

1，基本描述

为了支持sb指令字节操作功能添加的模块。将alu输出的地址和breg输出的数据处理后再传给dm。因为dm只能处理4的倍数的地址，故在本模块截取低两位，并把地址传给dm并读取dm对应地址的输出，并根据截取的低两位地址决定将接受的字节数据插入到哪个位置。把插入完的数据输回dm。

2，模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [31:0]dm\_out | I | 根据输入给dm的地址得到的Dm的数据，用于sb指令操作。 |
| [31:0]busb | I | Breg输出的数据，若不是sb指令则原封不动输出给dm，若是sb指令则将其低八位与接收到的dm数据拼接。 |
| [9:0]addr | I | Aluout传来的10位地址，若不是sb指令则原封不动输出给dm，若是sb指令则将其低两位归零输出给dm。 |
| sb | I | Sb指令标志信号  1：当前指令是sb指令  0：当前指令不是sb指令 |
| [31:0]dm\_in | O | 输出给dm的数据，若不是sb指令则将busb原封不动输出给dm，若是sb指令则将busb低八位与接收到的dm数据拼接再输出。 |
| [9:0]addr\_out | O | 输出给dm的地址。 |

3，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 处理地址 | 若是sb指令则将地址低两位归零输出给dm |
| 2 | 处理数据 | 若是sb指令则将输入数据低八位与dm输出数据拼接再输出给dm |
| 3 | 支持sb指令 | 若是sb指令，根据传入地址低两位决定将待写入字节数据写入那个位置。 |

### Gpr模块

1，基本描述

寄存器组，拥有32个32位寄存器，可以根据位选信号和时钟信号选择读取和输出指定寄存器的信息，以及根据写使能的开关将指定信息写入指定寄存器。

2，模块新增接口

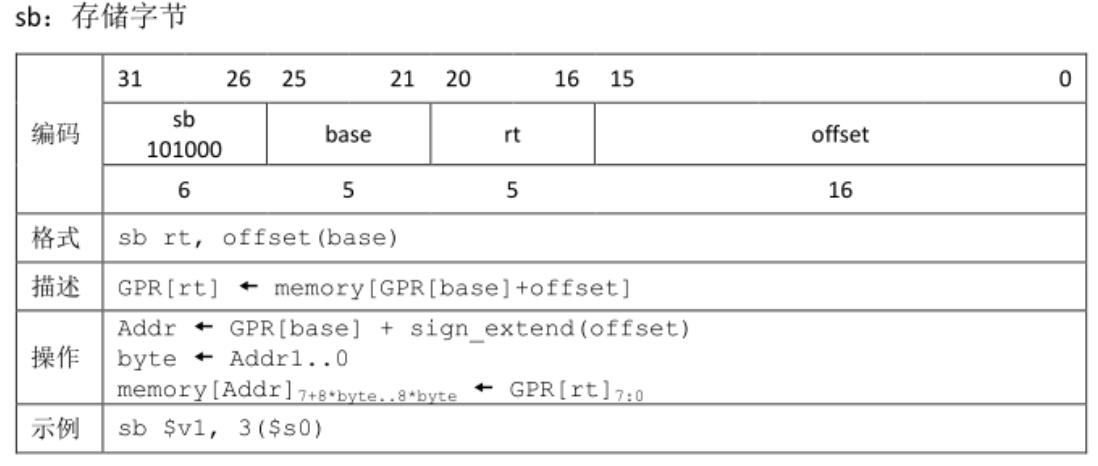
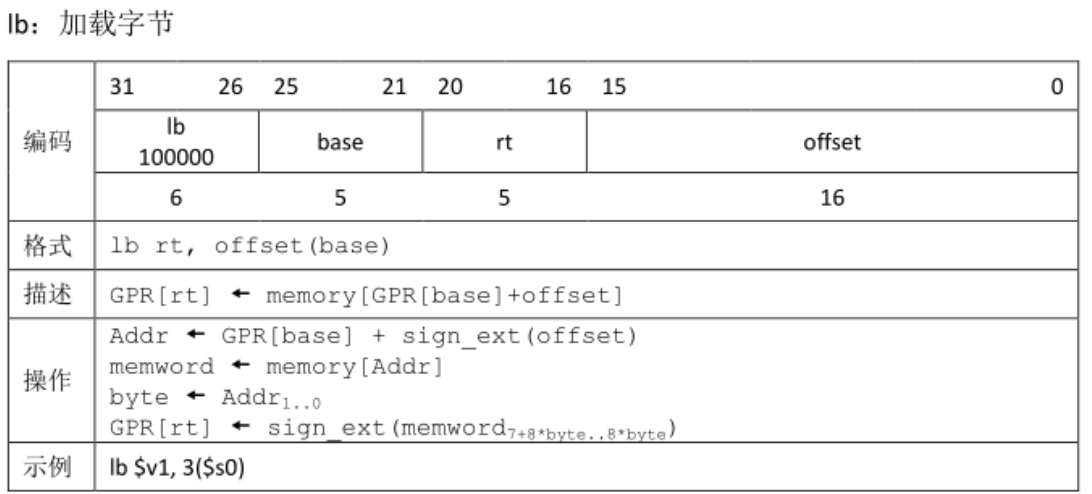
|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| lb | I | Lb指令标志信号  1：当前指令是lb指令  0：当前指令不是lb指令 |

3，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择地址 | 根据选择信号选择读取和写入的寄存器地址 |
| 2 | 选择数据 | 根据选择信号选择写入寄存器的数据 |
| 3 | 输出数据 | 读出对应地址寄存器的存储数据并输出 |
| 4 | 处理溢出 | 根据当前指令是否为addi以及溢出标志是否有效来处理addi的溢出 |
| 5 | 处理slt | 根据当前指令是否为slt以及小于标志是否有效来决定写入数据 |
| 6 | 处理bltzal | 根据当前指令是否为bltzal以及小于标志是否有效来决定是否写入 |
| 7 | 处理lb指令 | 依据当前指令是否为lb决定dm寄存器输出数据的写入方式。若是，则将传入数据低八位符号扩展后再写入。若不是，则正常写入。 |

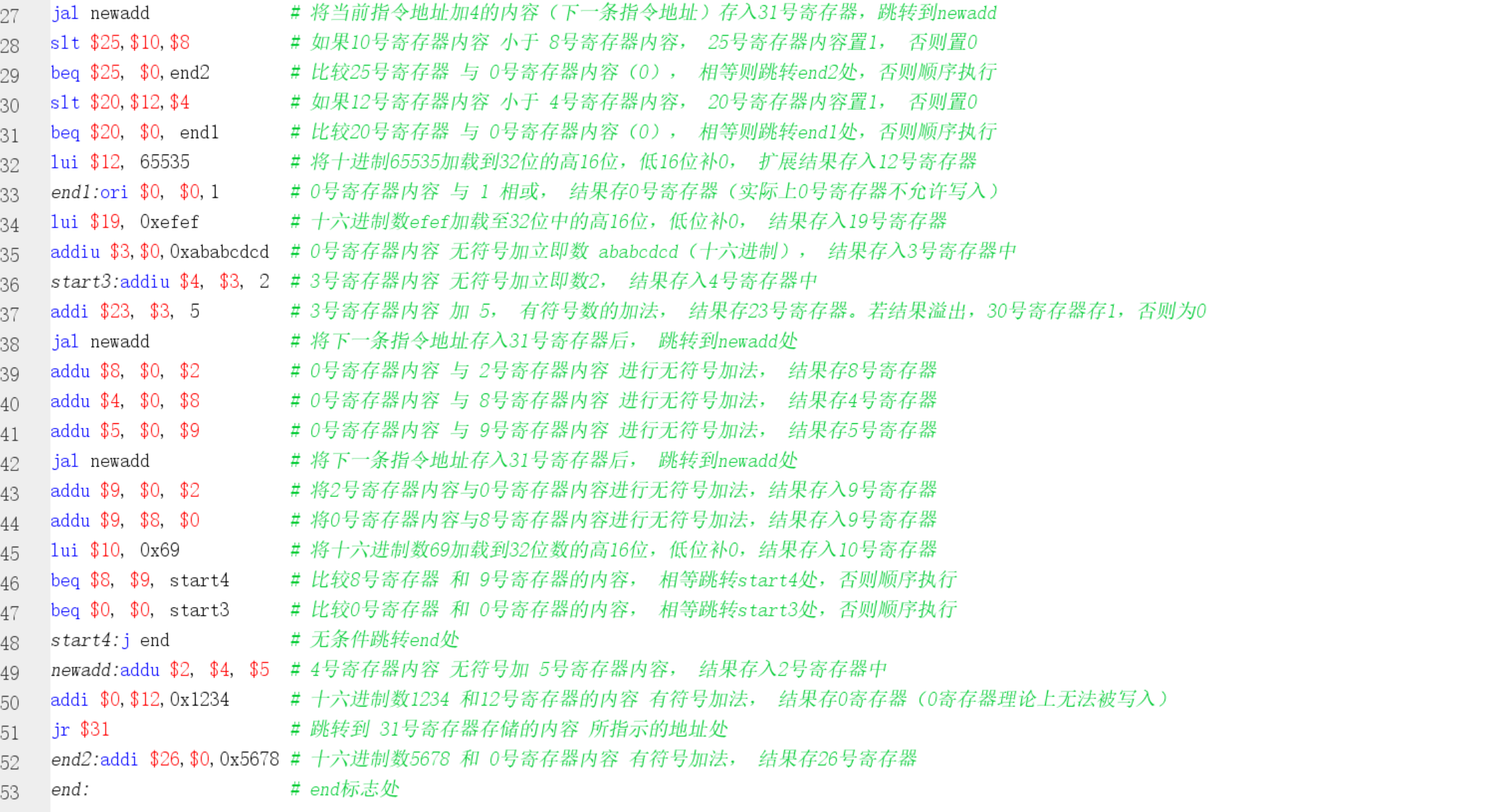
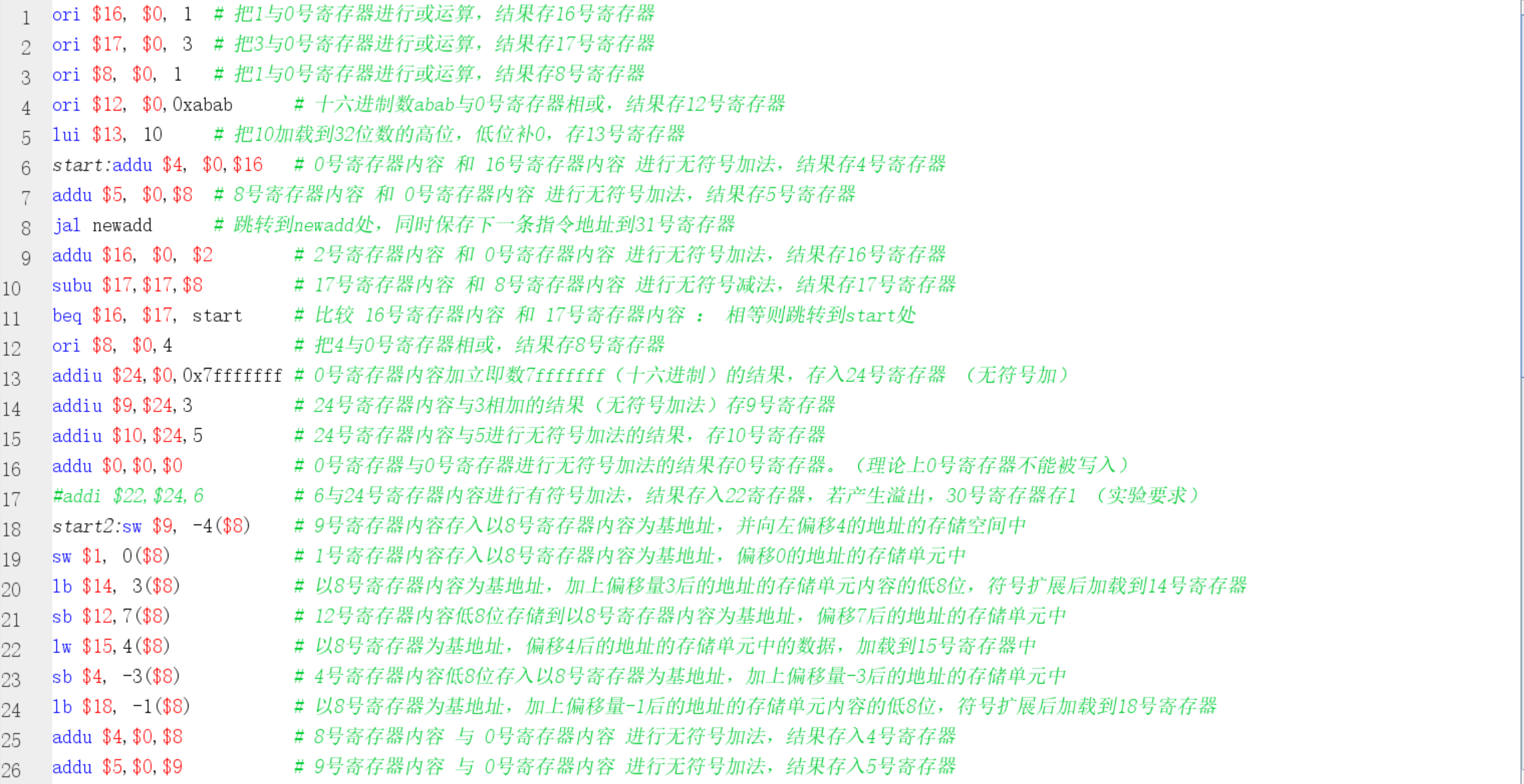
## 指令描述

以下是project2新增的两条指令：



## 四，测试程序

### 主要测试程序



## 五，测试结果

### Mars结果

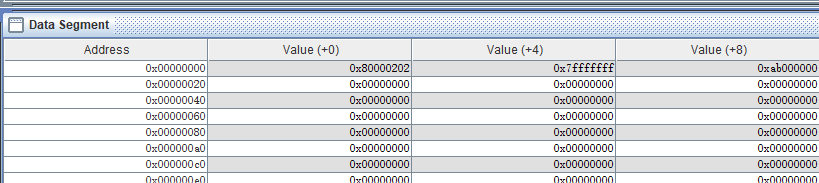
1，主要测试程序

寄存器存储内容：

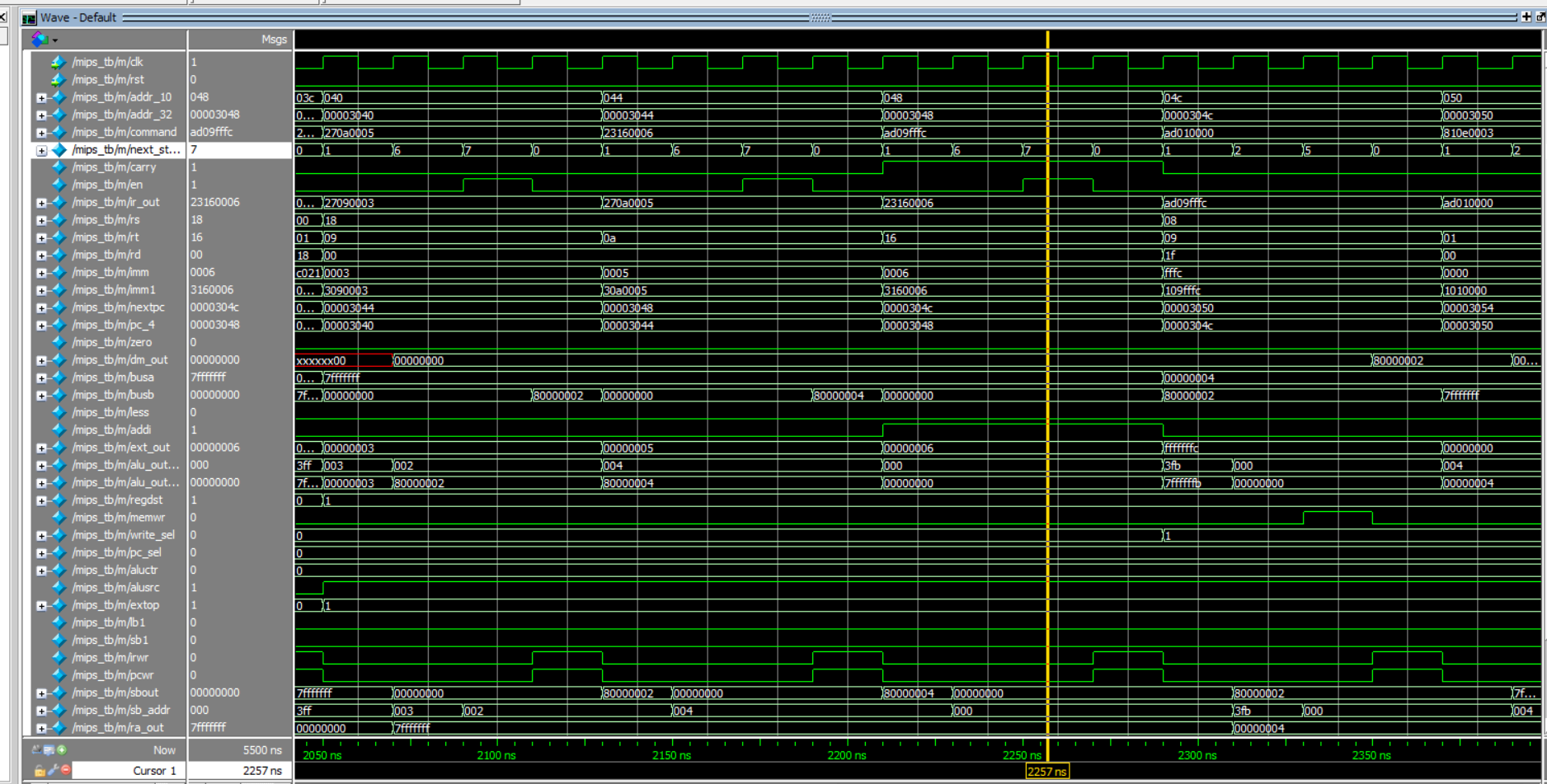


应注意的是，mars中28和29号寄存器内容与程序运行无关，故在modelsim仿真中28和29号寄存器内容应是0。

存储器存储内容：



### Modelsim仿真波形



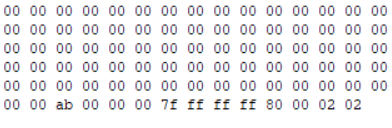
以图中黄线所在指令为例，指令地址是00003044，指令码是23160006，即addi $22,$24,6；可以看出，本指令经历0，1，6，7四个阶段。（Tips：值得注意的是s0阶段显示的地址才是本次指令真正的地址，pc会在s1阶段就将输出值更新为pc+4，一直保持到下一条指令s1阶段到来前。）

从s0阶段开始看，s0阶段pc和ir写使能打开；s1阶段pc更新成pc+4，ir更新成本次执行的指令，controller译码并发送对应信号打开数据通路，构成addi需要的数据通路。（值得注意的是进位信号之所以在s1阶段就有效是因为rega寄存器还保持着上一条指令的输出，而立即数已经经过扩展并传到alu里参与运算，二者相加正好溢出，故此处溢出并不是设计有问题。）

S6执行阶段，ab寄存器更新输出值，alu加法计算并使溢出信号有效，aluout等待更新。

S7回写阶段，aluout输出值更新，gpr写使能打开，将alu计算结果回写进gpr指定寄存器。





Modelsim寄存器和存储器值

对比可知，仿真结果正确。

## 六，总结与心得体会

Project2增强了我对不同指令的各个阶段的理解，通过合理设计状态机，分配信号，我对于将指令分成多个阶段进行处理这一思想有了更为深刻的认识，也正是因为这一次课设，我把每一条指令需要的指令阶段以及控制信号的何时产生，都理的更加顺畅了，在自己脑子里构建了更加清晰的认知。在完全学懂了多周期后，我自然地体会到了多周期对于指令执行时间的节省与高效率，对多周期的优点有了深刻的认知。同时这又是对project1知识的一次补完。

本project问答测验时我因为紧张等原因出现重大失误，幸好老师通情达理给予我第二次机会证明自己，同时鼓励我要锻炼自己的心态，口头表达能力，要全方位发展才能达到自己应有的成绩。我深受感动，同时也为自己的失态反思，这份经验一定可以在未来对我产生帮助。

# Project3 VerilogHDL完成MIPS微系统开发(支持设备与中断)

## 一，总体数据通路设计



MIPS微系统的数据通路主要分为两个部分，一个是多周期处理器的数据通路，另一个是CPU外部的bridge以及和外设的连接。

多周期处理器的数据通路，在P2的基础上增加了CP0协处理器及其相应数据通路。为gpr的写入数据增加了一个

CPU外部设备主要有三个：0号设备定时器，1号写入设备，2号输出设备。

系统桥将外设与CPU内部连接起来，起到联系的作用。

通路中画出了数据通路以及中断信号的传输路径。

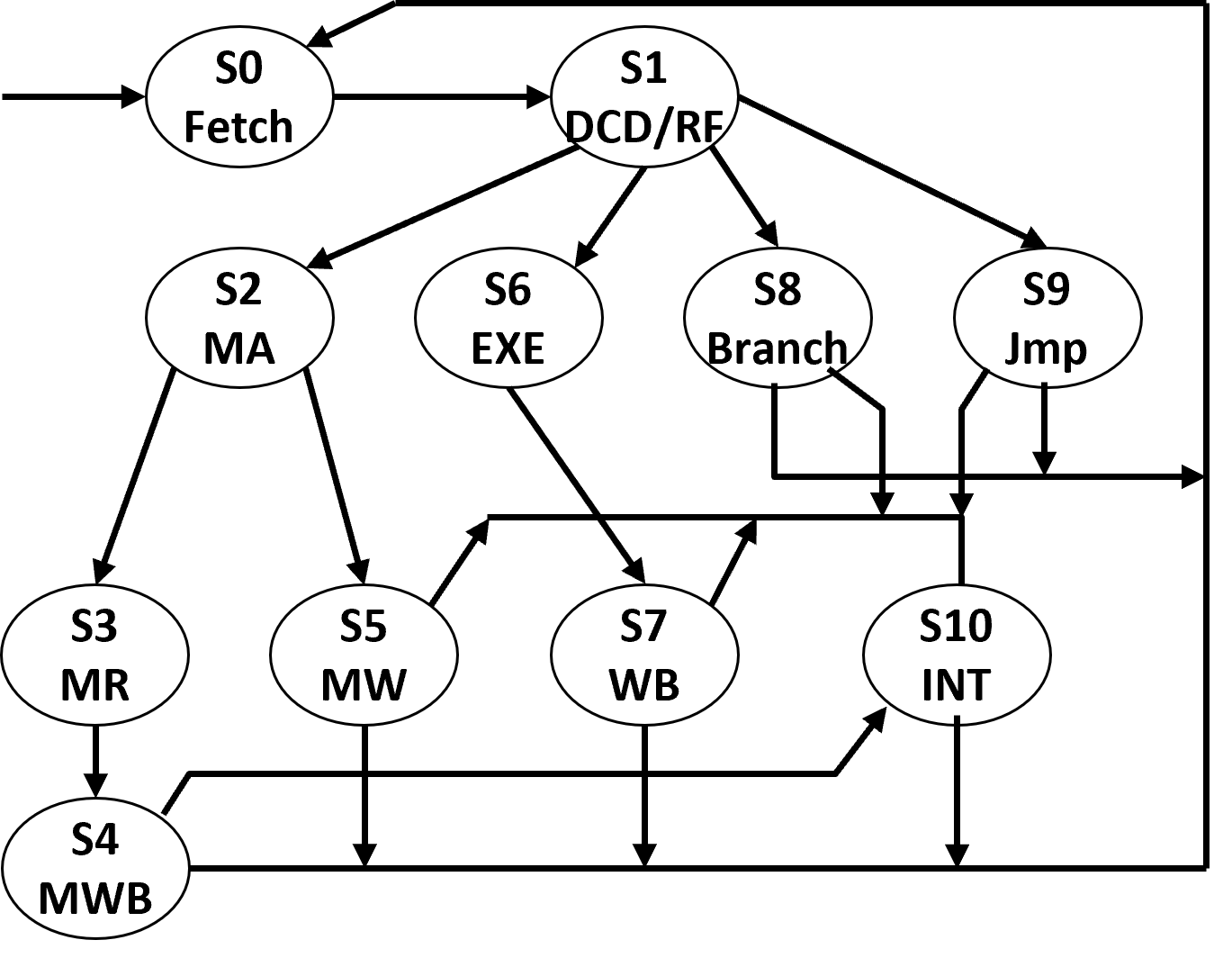
## 二，模块定义

本模块只叙述相比project2新增的部分。

### Controller模块

1，基本描述

Controller模块是根据im传来的指令进行译码，并根据指令需要向各元件发出控制信号构建正确数据通路的元件。完成译码功能。



如图设计状态机。状态机新增加了一个S10状态，他是附于每一条指令的最后一个阶段之后的。如果在指令的最后一个阶段，并且intreq信号有效的话，就会进入S10状态，中断状态；否则就忽略掉S10状态，直接跳回S0即可。

对于MTC0和MFC0两条指令，他们的指令功能同sw/sb/lw/lb是相类似的，都是读/写功能，故将他们两个与上述四个指令分在同一个周期S2。到了S2阶段后，MFC0是读cp0寄存器并写入gpr寄存器中，与lw/lb指令类似，有回写阶段，故分到S3读阶段（MFC0是读cp0的寄存器）然后S4回写阶段。MTC0是写cp0寄存器，这与sw/sb指令功能类似，故分到S5阶段进行写入（这里写的是cp0的寄存器）。

对于ERET指令，其功能与j/jal类似，进行无条件跳转（形成地址），故分到S9阶段，在S9阶段完成跳转地址的更新。

Controller译码经过如下几个阶段：1，译码阶段根据高六位和低六位判断当前是什么指令，

2，根据当前指令决定状态机走向 3，配合状态机走向决定各控制信号。

2，模块新增接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| intreq | I | 中断请求信号  1：中断请求  0：无中断请求 |
| cp0\_en | O | Cp0写使能，  1：写使能有效  0：写使能无效 |
| bridge\_en | O | bridge写使能，  1：写使能有效  0：写使能无效 |
| exlset | O | exl标志信号，  1：已经进入中断响应，不可再接受其他中断请求  0：不在中断响应，可接收其他中断请求。 |
| exlclr | O | Exl清零信号  1：Exl清零  0：Exl不清零 |
| intpc | O | pc跳转中断地址标志信号  1：跳转  0：不跳转 |

3，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能定义 | 功能描述 |
| 1 | 译码 | 将指令高六位和低六位取出，判断当前指令种类 |
| 2 | 产生控制信号 | 根据译码结果和当前状态选择适当控制信号发给各元件，构建正确数据通路 |
| 3 | 构建状态机 | 按设计好的状态机构建状态，并根据不同的指令确定下一个状态。 |

### Npc模块

1，基本描述

计算下一条指令的地址并输出给pc。增加intreq，epc，eret信号用于支持中断指令。

当intreq信号有效，则跳转中断地址32'h00004180；若eret有效，则地址跳回epc。

2，模块新增接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| intreq | I | 由controller发来，中断信号  1：中断  0：不中断 |
| [31:0]epc | I | 由cp0传来，中断产生时的pc地址的pc+4，用于eret指令跳转出中断使用。 |
| eret | I | Eret指令标志。  1：当前指令是eret指令  0：当前指令不是eret指令 |

3，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 计算地址 | 根据指令需求计算下一条指令地址并输出给pc |
| 2 | 输出pc+4 | 将pc+4的值输出给gpr，用于在jal和bltzal指令保存下一条指令地址。 |
| 3 | 中断跳转 | 中断信号有效时跳转到中断地址 |
| 4 | 中断返回 | 中断返回信号有效时跳转回epc |

### Cp0模块

一，基本描述

cp0模块是协处理器，主要用来处理异常。本次cp0模块中包含4个寄存器：sr、cause、epc和prid。sr寄存器的功能是用[15:10]位存储屏蔽位im，第零位存储全局中断使能信号ie以及第一位存储防止再中断的标志位exl；cause寄存器用[15:10]位不断的锁存中断产生的原因；epc寄存器用于存储下一条指令地址；prid寄存器是存储个性化签名。需要注意的是，cause寄存器只读。

二，模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [31:0]pc | I | Epc输出 |
| [31:0]busb | I | 写入数据 |
| [7:2]hwint | I | 6路中断信号，因为本次只考虑外界给的中断，故只有最低位可能为1。 |
| [4:0]sel | I | 读写地址 |
| wen | I | 寄存器写使能 |
| exlset | I | 中断开始信号，完成epc寄存器写入以及exl置位  1：中断  0：无中断 |
| exlclr | I | eret信号，开exl中断  1：清零，可再次接收中断  0：不清零 |
| clk | I | 时钟信号 |
| rst | I | 充值信号 |
| intreq | O | 根据屏蔽位im以及标志位ie、exl产生的中断信号 |
| [31:0]epc | O |  |
| [31:0]dout | O |  |

三，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 写数据 | 根据数据输入和写入地址，写入数据（使能有效时） |
| 2 | 读数据 | 根据地址输入，读数据并输出 |
| 3 | 产生中断信号 | 根据6路hwint中断信号产生1位中断信号intreq |
| 4 | 记录异常原因 | cause寄存器不断锁存HWInt信号 |

### Bridge模块

一，基本描述

Bridge模块的基本功能是连接cpu与外设的通讯桥梁，传递彼此之间的地址、数据以及自身生成的使能信号。三大功能为，地址转换、读数据、写数据。同时还是定时器产生的IRQ信号的传递桥梁。写使能配合地址选出访问的外设编号，共同判断操作的是哪个设备。

外设1输入设备，不需要写使能，只读。外设0是timer，外设2是输出设备。

二，模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [31:0]praddr | I | 外设地址 |
| [31:0]prwd | I | 外设写数据 |
| [31:0]dev0\_rd | I | 外设0读数据 |
| [31:0]dev1\_rd | I | 外设1读数据 |
| [31:0]dev2\_rd | I | 外设2读数据 |
| wecpu | I | 外设写使能，需要搭配选择信号使用 |
| Irq | I | 计时器产生的中断请求信号 |
| [31:0]prrd | O | 外设读的数据（选择后的） |
| [31:0]devwd | O | 外设写数据 |
| [3:2]devaddr | O | 外设地址，因为只需要两位故直接截取两位输出 |
| wedev0 | O | 外设0写使能 |
| wedev2 | O | 外设2写使能 |
| [5:0]HWInt | O | 6路中断信号 |

三，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 传递读出的外设数据 | 将从外设读的数据传给gpr进行写入 |
| 2 | 传递要写给外设的数据 | 将dm读出数据传给外设进行写入 |
| 3 | 外设选择，使能信号生成 | 根据地址和译码信号，产生使能信号 |
| 4 | 汇总中断信号并传递 | 计时器irq传给桥，桥汇总到6位中断信号，传给cpu内部 |

### Dev\_output模块

一，基本描述

outputDev为外设输出模块，包含两个寄存器：初值寄存器以及当前值寄存器。模块具有输入输出功能：输入时，在时钟上升沿且写使能有效时，将数据写入地址指示的寄存器中；输出时，根据地址，选择正确的寄存器的值进行输出。若复位信号有效，两寄存器值清零。

两个寄存器的目的是，便于在后续秒计数时，判断当前输入与初值寄存器的值是否相同。如果相同，即输入没变化，则自加1。否则将两个寄存器更新为新的输入值，然后在下一次中断时再进行自加1。

二，模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Rst | I | 复位信号 |
| clk | I | 时钟信号 |
| en | I | 写使能信号 |
| [3:2]addr | I | 读/写地址，也可以理解为寄存器选择信号。 |
| [31:0]din | I | 写入数据 |
| [31:0]dout | O | 读出数据 |

三，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读数据 | 根据addr读出指定寄存器数据。 |
| 2 | 写数据 | 根据addr，在写使能有效且时钟上升沿时，写入数据到寄存器中。 |

### Timer模块

一，基本描述

Timer模块主要由三个寄存器组成：控制寄存器、初值寄存器以及计数器。主要的功能是，当计数器为0时，产生中断请求信号IRQ。

寄存器的功能：控制寄存器主要负责的是管理是否允许中断、计数模式选择以及是否允许计数。第三位决定是否允许中断，第二，第一位决定采用模式，第零位决定是否允许计数。初值寄存器用于存放初值，保持不变。计数器用于在时钟沿来临时进行自减，完成计数。

当计数器值为0时，根据不同的模式执行不同的操作。当为模式0时，则一直保持0不变，直到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；当为模式1时，则自动加载初值寄存器的值到计数器，开始新一轮的倒计数。

二，模块接口

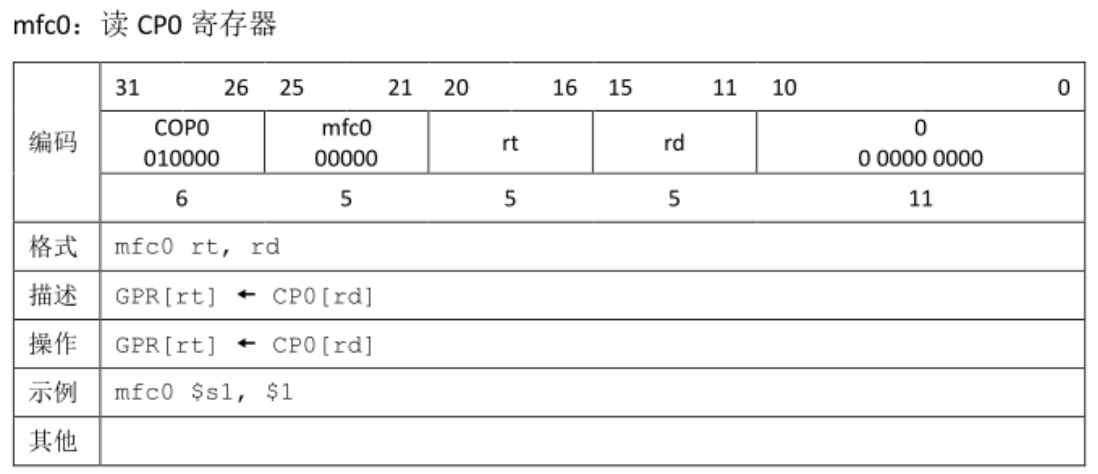
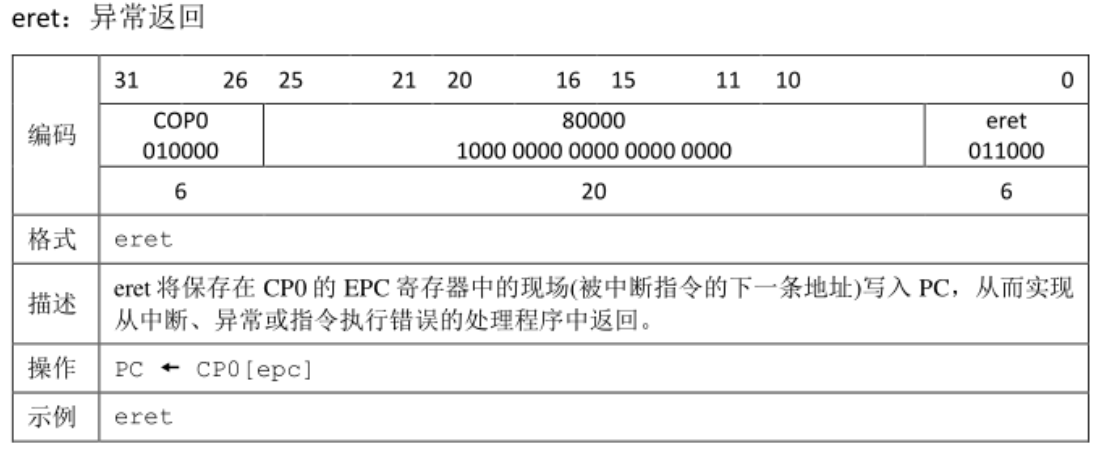
|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| CLK\_I | I | 时钟信号 |
| RST\_I | I | 复位信号 |
| [3:2]ADDR\_I | I | 读/写地址,三个寄存器，两位地址信号即可。且低两位永远为0，故不用考虑。 |
| WE\_I | I | 写使能信号  1：允许写  0：不允许写 |
| [31:0]DAT\_I | I | 写入数据 |
| [31:0]DAT\_O | O | 读出数据 |
| IRQ | O | 中断请求信号，count为0时产生  1：请求中断  0：不请求中断 |

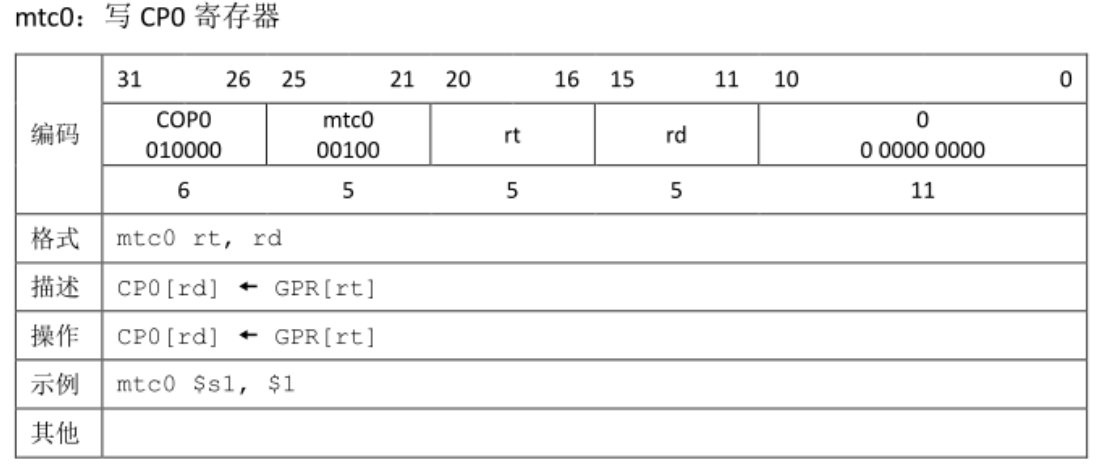
三，功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 计数 | 根据控制寄存器内容，在不同的模式下完成计数（允许计数时）。 |
| 2 | 产生中断请求信号 | 当计数器值为0时，产生中断请求信号IRQ。 |

## 指令描述

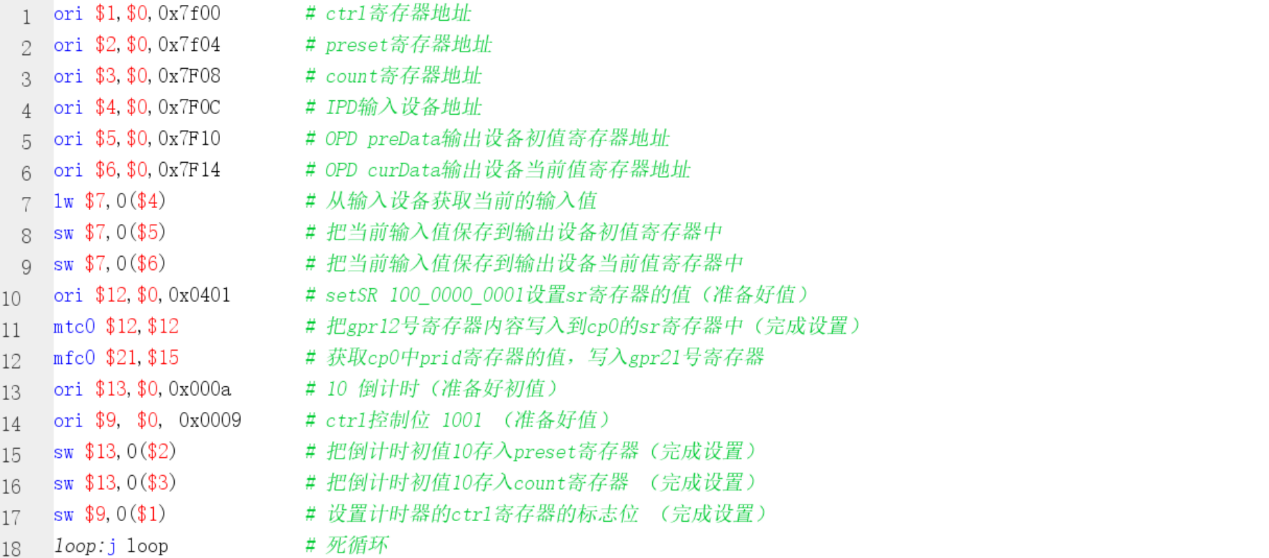
以下是新增加的三条指令：



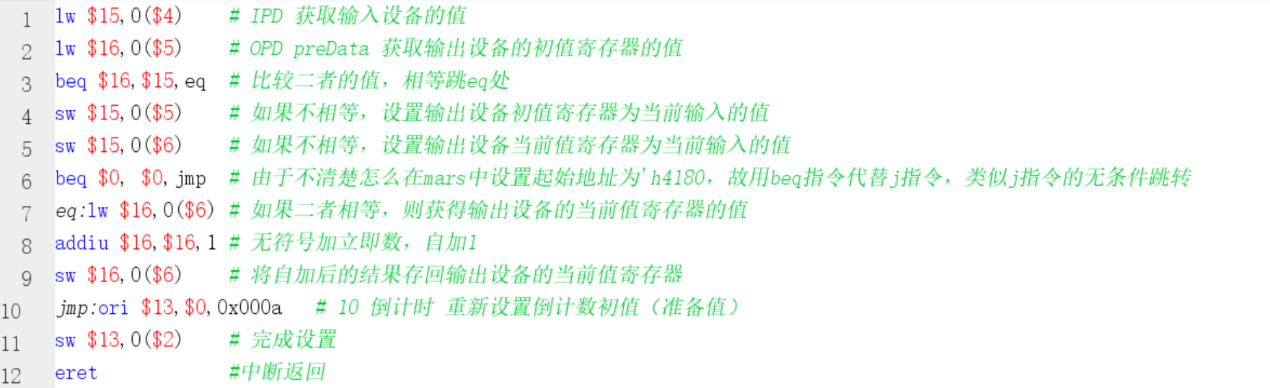


## 测试程序

主程序：

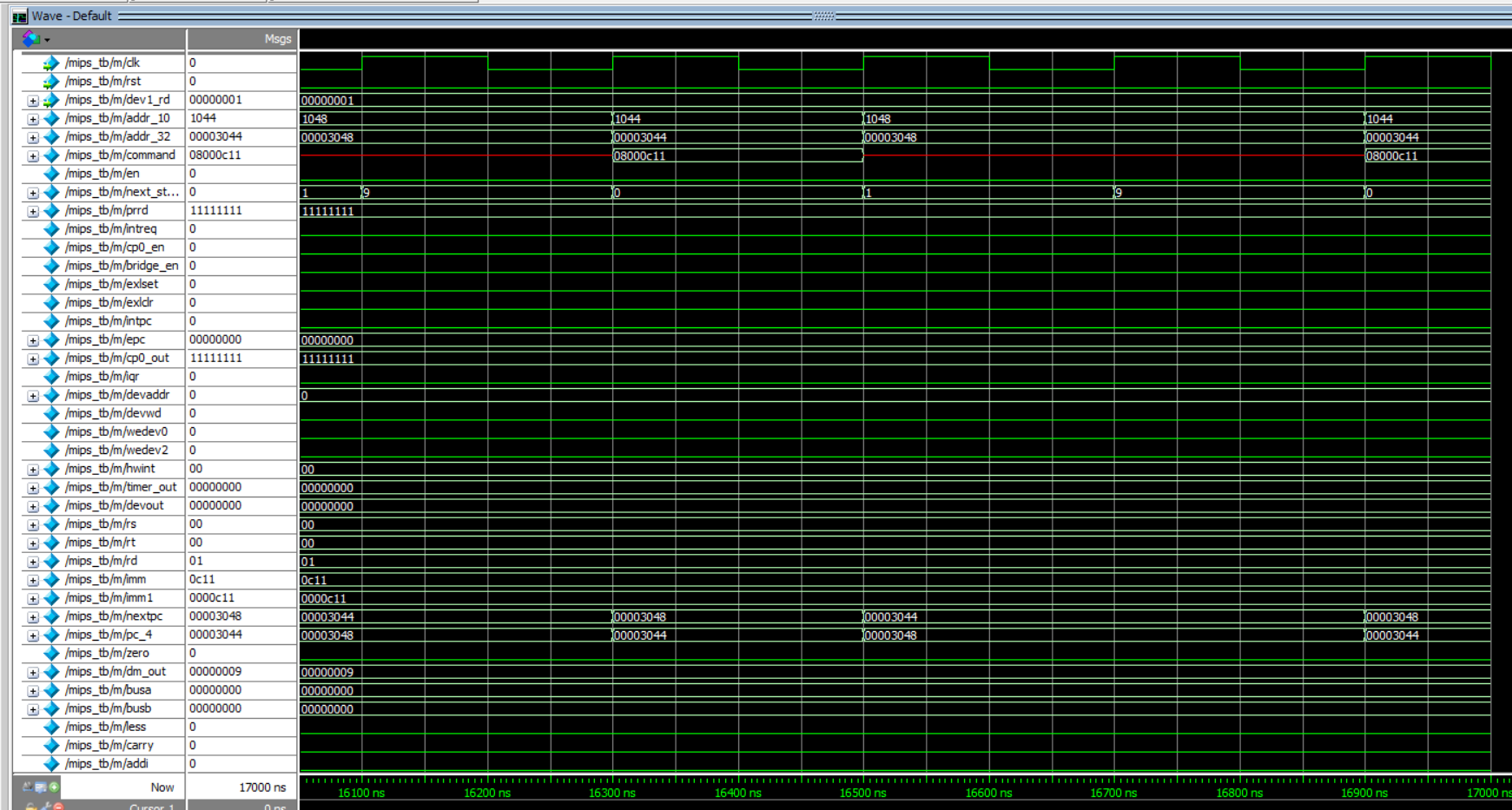


中断子程序：



## 测试结果

### Modelsim波形仿真



数据通路连接完成，但中断指令并未正确执行

## 总结与心得体会

通过完成MIPS微系统的开发，我不仅仅对cpu内部的数据通路、信号控制有了更上一层的了解，更关键的是，我对于cpu内部是如何与cpu外部进行通讯的有了很全面的了解。

在开发过程中，印象比较深刻的就是中断请求信号的传输过程：从外设定时器发出，经历系统桥，到协处理器，再到控制器，最后再到npc模块。其中比较关键的是intreq信号与状态机的配合使用，以及在多个中断需要处理时，使用屏蔽位来管理他们的优先级这一思想，都使我受益匪浅。

这次的MIPS微系统开发，主要解决的问题是与外设的通讯、交互功能。解决问题的主要思想是，合理设计控制信号的通路、合理设计状态机、合理设计系统桥，并且需要给外设分配符合存储空间规定的外设地址，以此来达到设计要求。虽然程序调试并未成功，但理论知识我已全部掌握，相信一定能在将来为我提供帮助。