**计算机组成原理**

**期末大作业报告**

**学 号\_\_\_\_\_ \_\_\_20020203 \_\_\_\_\_\_\_\_**

**姓 名\_\_\_\_\_\_\_\_\_\_\_王思哲\_\_\_\_\_\_\_\_\_\_\_**

**指导教师\_\_\_\_\_\_\_\_\_\_\_魏坚华\_\_\_\_\_\_\_\_\_\_\_**

**提交日期\_\_\_\_\_\_2022年6月14日 \_\_\_**

**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** |  |
| **报告与大作业功能一致性** | **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

[一、 总体数据通路结构设计图 3](#_Toc106143723)

[二、 模块定义 3](#_Toc106143724)

[2.1 ALU模块 3](#_Toc106143725)

[2.1.1 基本描述 3](#_Toc106143726)

[2.1.2 模块接口 3](#_Toc106143727)

[2.1.3 功能定义 4](#_Toc106143728)

[2.2 IFU模块 4](#_Toc106143729)

[2.2.1 基本描述 4](#_Toc106143730)

[2.2.2 模块接口 5](#_Toc106143731)

[2.2.3 功能定义 5](#_Toc106143732)

[2.3 Controller模块 6](#_Toc106143733)

[2.3.1 基本描述 6](#_Toc106143734)

[2.3.2 模块接口 6](#_Toc106143735)

[2.3.3 功能定义 7](#_Toc106143736)

[2.4 GPR模块 7](#_Toc106143737)

[2.4.1 基本描述 7](#_Toc106143738)

[2.4.2 模块接口 7](#_Toc106143739)

[2.4.3 功能定义 7](#_Toc106143740)

[2.5 DM模块 8](#_Toc106143741)

[2.5.1 基本描述 8](#_Toc106143742)

[2.5.2 模块接口 8](#_Toc106143743)

[2.5.3 功能定义 8](#_Toc106143744)

[2.6 MUX模块 8](#_Toc106143745)

[2.6.1 基本描述 8](#_Toc106143746)

[2.6.2 模块接口 9](#_Toc106143747)

[2.6.3 功能定义 9](#_Toc106143748)

[2.7 EXT模块 9](#_Toc106143749)

[2.7.1 基本描述 9](#_Toc106143750)

[2.7.2 模块接口 9](#_Toc106143751)

[2.7.3 功能定义 9](#_Toc106143752)

[三、指令描述 10](#_Toc106143753)

[四、测试程序 10](#_Toc106143754)

[4.1 MIPS-Lite1指令集的测试程序 10](#_Toc106143755)

[4.2 新增指令BGEZAL的测试程序 11](#_Toc106143756)

[五、测试结果 12](#_Toc106143757)

[5.1 MIPS-Lite1指令集测试结果 12](#_Toc106143758)

[5.1.1 MARS中结果 12](#_Toc106143759)

[5.1.2 Logisim中结果 13](#_Toc106143760)

[5.2 BGEZAL指令测试结果 15](#_Toc106143761)

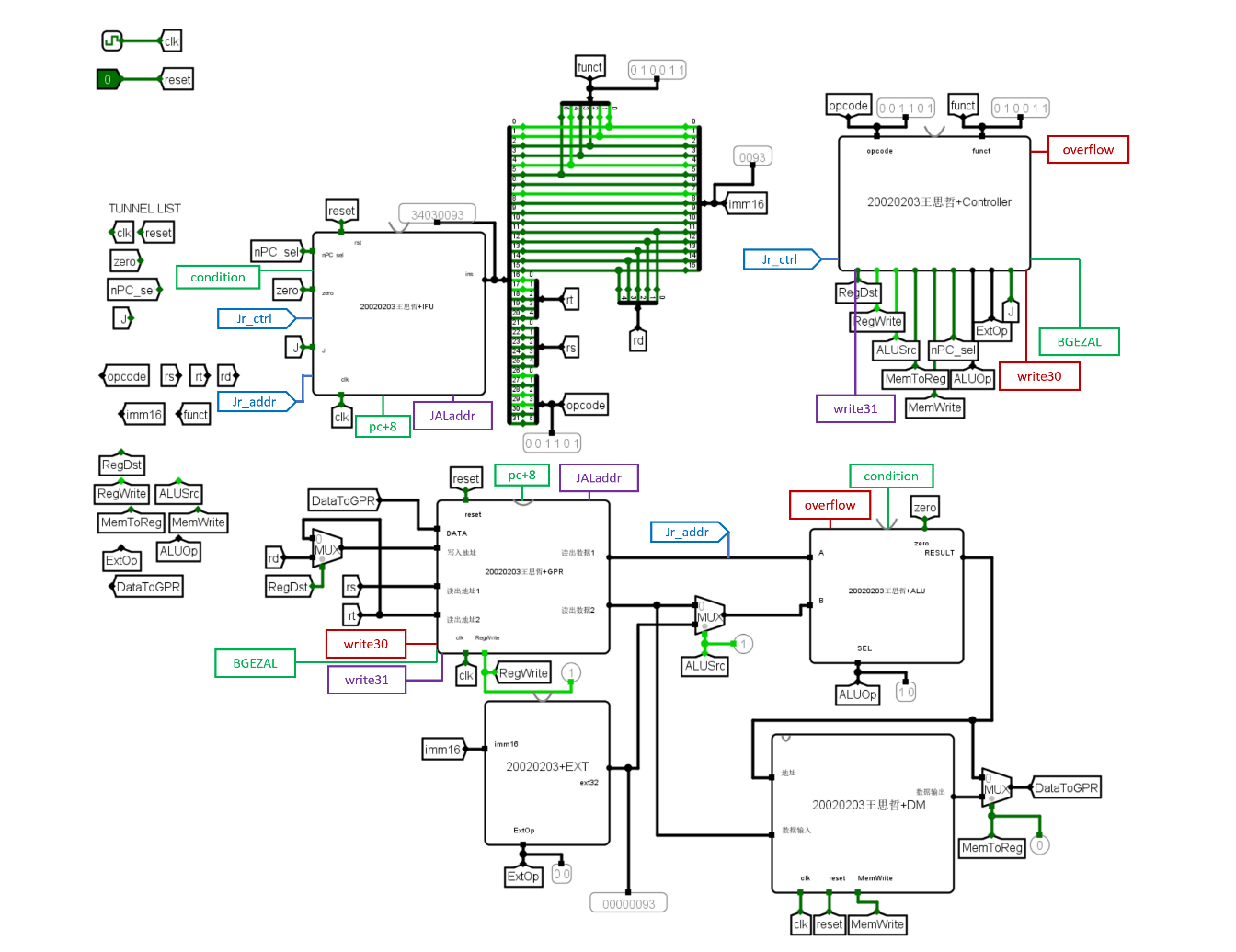
[5.2.1 MARS中结果 15](#_Toc106143762)

[5.2.2 Logisim中结果 15](#_Toc106143763)

[六、总结与心得体会 16](#_Toc106143764)

1. **总体数据通路结构设计图**

本次单周期处理器的数据通路设计图(MIPS-Lite1指令集)，是在之前的数据通路图(MIPS-Lite指令集)上进行的扩充，具体信号通路设计如下图所示。



1. **模块定义**

## 2.1 ALU模块

### 2.1.1 基本描述

ALU模块为运算模块，主要功能是根据控制信号指定的运算类型，对两个输入的32位操作数进行相应的运算（也有可能只用到了一个操作数）并输出运算结果。除了对操作数进行相应的运算外，ALU还支持溢出检测、判断结果是否为0以及判断结果是否大于0。

### 2.1.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A[31:0] | I | 第一个操作数 |
| B[31:0] | I | 第二个操作数 |
| ALUOp[2:0] | I | 控制运算类型  000：加法 001：减法 010：逻辑或  011：小于置一 100：加立即数 101：大于等于0跳转 |
| zero | O | 判断alu\_res是否为0  0：否 1：是 |
| alu\_res | O | 32位运算结果 |
| overflow | O | 判断addi指令第一个操作数加立即数（有符号）是否产生溢出  0：否 1：是 |
| condition\_jdg | O | 判断第一个操作数是否大于等于0  0：否 1：是 |

### 2.1.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 计算 | 根据运算控制信号完成相应运算：  加法：计算操作数1+操作数2  减法：计算操作数1-操作数2  逻辑或：计算操作数1 || 操作数2  小于置一：若操作数1<操作数2，结果为1，否则为0  加立即数：计算操作数1+操作数2，同时判断是否产生溢出  大于等于0跳转：比较操作数1与数字0的大小 |
| 2 | 判断 | 判断运算结果是否为0  判断运算结果是否大于等于0  判断addi运算时是否产生了溢出 |

## 2.2 IFU模块

### 2.2.1 基本描述

IFU主要完成的是取指功能。IFU内部结构关键部件有PC和指令寄存器。指令寄存器负责存储程序需要运行的所有指令，位宽8位，长度1024。PC由寄存器存储，在每次时钟沿到来时更新，更新为下一条指令的地址。

PC地址的更新分为5种情况，分别是j指令的更新、beq指令的更新、顺序执行下一条指令、jr指令的更新、bgezal指令的更新。

### 2.2.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 0：不复位 1：复位 |
| npc\_sel | I | 选择pc的下一个地址是否为beq成立时的地址 |
| zero | I | 判断alu\_res是否为0，与npc\_sel同时作用，用于选择下一个地址是否为beq成立时的地址 |
| j | I | 如果j==1，则pc更新为j指令跳转的地址 |
| jr\_ctrl | I | 如果jr\_ctrl==1，则pc更新为jr指令需要跳转的地址 |
| condition\_jdg | I | 如果condition\_jdg==1，则pc更新为bgezal需要跳转的地址 |
| insout[31:0] | O | 输出pc所指向的当前指令 |
| opcode[5:0] | O | opcode操作码 |
| rs[4:0] | O | 寄存器地址 |
| rt[4:0] | O | 寄存器地址 |
| rd[4:0] | O | 寄存器地址 |
| funct[5:0] | O | funct功能码 |
| imm16[15:0] | O | 立即数 |
| jalAddr[31:0] | O | pc+4 |
| pc\_p8[31:0] | O | pc+8 |

### 2.2.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 当复位信号有效时，PC置0 |
| 2 | 取指令 | 根据PC从IM中取出对应的指令 |
| 3 | 计算下一条指令地址 | 如果不是以下任何指令，pc🡨pc+4  如果是beq指令，且zero为1，pc🡨pc+4+(sign\_ext(insout[15:0])<<2)  如果是j指令，pc🡨{(pc+4)[31:28], insout[25:0], 2’b00}  如果是jr指令，pc🡨jrAddr  如果是bgezal指令，pc🡨pc+4+sign\_ext(offset||02) |

## 2.3 Controller模块

### 2.3.1 基本描述

根据每一条指令的opcode和funct以及alu产生的溢出信号overflow，产生与之对应的控制信号。

### 2.3.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| opcode[5:0] | I | 操作码 |
| funct[5:0] | I | 功能码 |
| overflow | I | alu加法(addi)溢出标志 0：未溢出 1：溢出 |
| RegDst | O | 选择寄存器写入地址 0：rt 1：rd |
| RegWrite | O | 寄存器写使能信号 0：无效 1：有效 |
| ALUSrc | O | 选择ALU第二个操作数来源  0：gpr第二个输出数据  1：立即数拓展结果 |
| MemtoReg | O | 选择传给寄存器的写入数据  0：alu计算结果  1：dm指定存储单元的数据 |
| MemWrite | O | 数据存储器写使能信号 0：无效 1：有效 |
| Branch | O | 是否为beq指令 0：否 1：是 |
| J | O | 当前是否为j指令，跳转信号 0：否 1：是 |
| ALUOp[2:0] | O | alu运算类型控制信号  000：加 001：减 010：逻辑或  011：小于置一slt 100：addi 101：bgezal  default：alu\_res = 0 |
| ExtOp[1:0] | O | 立即数扩展类型  00：0拓展  01：符号拓展  10：16位立即数拓展至高16位，低位补0 |
| WriteToGPR\_30 | O | 30号寄存器写入标志 0：不写入 1：写入 |
| jr\_ctrl | O | jr指令标志 0：不是jr指令 1：是jr指令 |
| write\_31 | O | 31号寄存器写入标志 0：不写入 1：写入 |
| bgezal\_31 | O | bgezal专用的31号寄存器写入标志  0：不写入 1：写入 |

### 2.3.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 产生控制信号 | 根据opcode和funct信息，产生对应的控制信号。 |

## 2.4 GPR模块

### 2.4.1 基本描述

寄存器模块，主要功能是数据的临时存储。含有32个32位寄存器，当对应编号寄存器的写使能有效的时候，可以完成数据的写入功能；任何时刻都可以完成数据的读取功能。可以将32个寄存器清零。0号寄存器不能被写入。

### 2.4.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 |
| RegWrite | I | 写使能信号(所有) 0：不能写入 1：写入 |
| WriteToGPR\_30 | I | 30号寄存器写入标志 0：不写入 1：写入 |
| writeData[31:0] | I | 写入数据 |
| writeAddr[4:0] | I | 写入地址 |
| readAddr\_1[4:0] | I | 读出数据1所在的地址 |
| readAddr\_2[4:0] | I | 读出数据2所在的地址 |
| write\_31 | I | jal指令时31号寄存器写入标志 0：不写入 1：写入 |
| bgezal\_31 | I | bgezal指令时31号寄存器写入标志0：不写入1：写入 |
| pc\_p8[31:0] | I | bgezal指令时的写入数据 |
| jalAddr[31:0] | I | jal指令时的写入数据 |
| dataOut\_1[31:0] | O | 读出数据1 |
| dataOut\_2[31:0] | O | 读出数据2 |

### 2.4.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 复位信号有效时，32个寄存器清零 |
| 2 | 写数据 | 30和31号寄存器需要在特定的使能信号有效时写入数据；其余寄存器仅需在RegWrite写使能有效时，根据写入地址将数据写入寄存器中即可。 |
| 3 | 读数据 | 根据读取数据的地址，将数据读出并输出 |

## 2.5 DM模块

### 2.5.1 基本描述

该模块主要功能为存储数据，容量为1kb，位宽8位，长度1024，采用小端序的方式存储数据。输入地址32位（只取低10位），输入数据32位，输出数据32位。

### 2.5.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 |
| MemWrite | I | 写使能信号 0：不能写入 1：写入 |
| din[31:0] | I | 被写入的数据 |
| addr[31:0] | I | 写入/读出地址 |
| dout[31:0] | O | 读出数据 |

### 2.5.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 复位信号有效时，数据存储器数据清零 |
| 2 | 写数据 | 写使能有效时，根据数据写入地址，将数据写入对应的存储器单元，采用小端序存储。 |
| 3 | 读数据 | 根据读取数据的地址，将数据读出并输出 |

## 2.6 MUX模块

### 2.6.1 基本描述

该模块实现的主要功能为数据选择器，根据选择信号，选择对应的输入数据进行输出。

定义了两种数据选择器，分别是32位二选一数据选择器、5位二选一数据选择器。

### 2.6.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| sel | I | 选择信号 |
| din\_0[4:0]/[31:0] | I | 输入数据1 |
| din\_1[4:0]/[31:0] | I | 输入数据2 |
| dout[4:0]/[31:0] | O | 输出数据 |

### 2.6.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择数据 | 根据选择信号，选择正确的输入数据并输出 |

## 2.7 EXT模块

### 2.7.1 基本描述

根据选择信号指示的拓展类型，将16位立即数扩展至32位。扩展类型包括0扩展，符号扩展以及立即数扩展至高位、低位补0三种。

### 2.7.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| imm16[15:0] | I | 16位立即数 |
| ExtOp[1:0] | I | 扩展类型选择信号  00：0扩展  01：符号扩展  10：立即数扩展至高位，低位补0 |
| ext32[31:0] | O | 扩展结果 |

### 2.7.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 扩展立即数 | 高16位0扩展、高16位符号扩展、立即数扩展至高16位并低16位补0 |

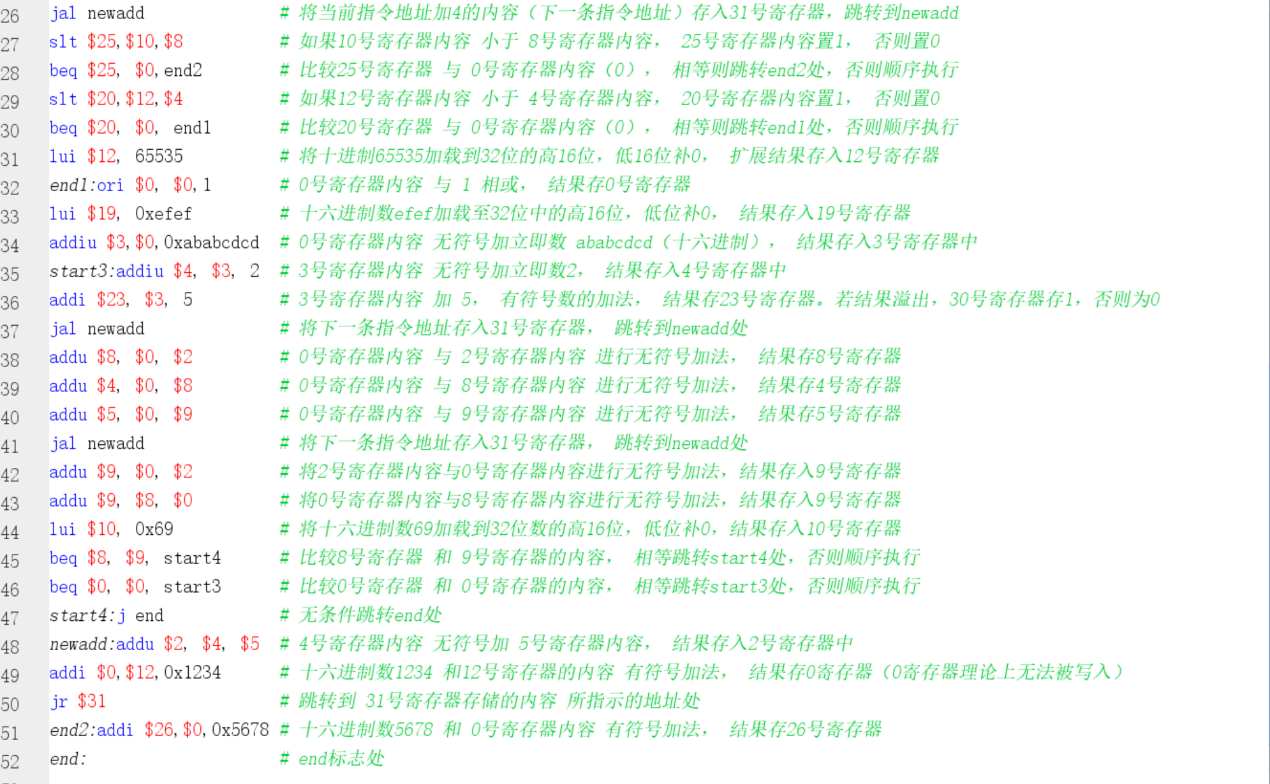
# 三、指令描述



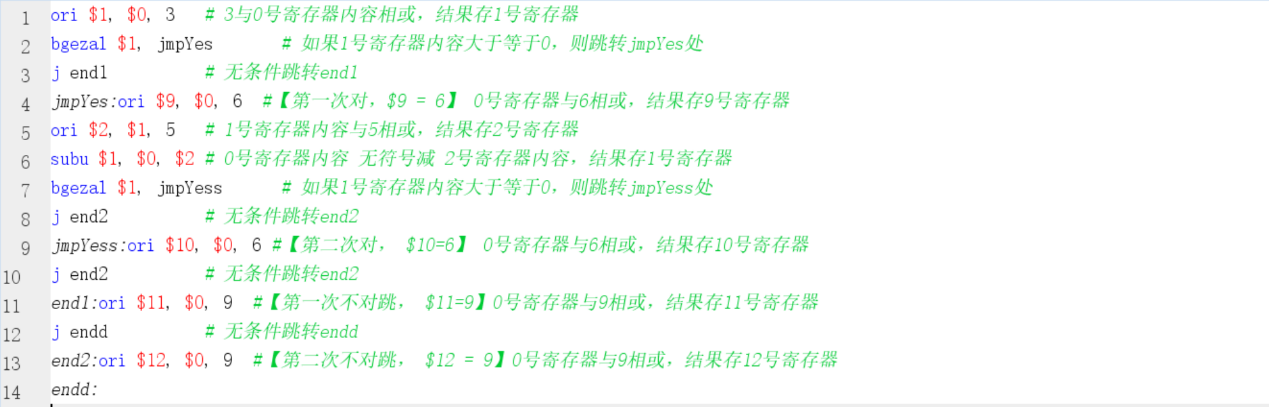
# 四、测试程序

## 4.1 MIPS-Lite1指令集的测试程序





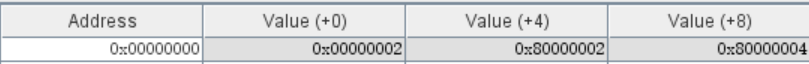
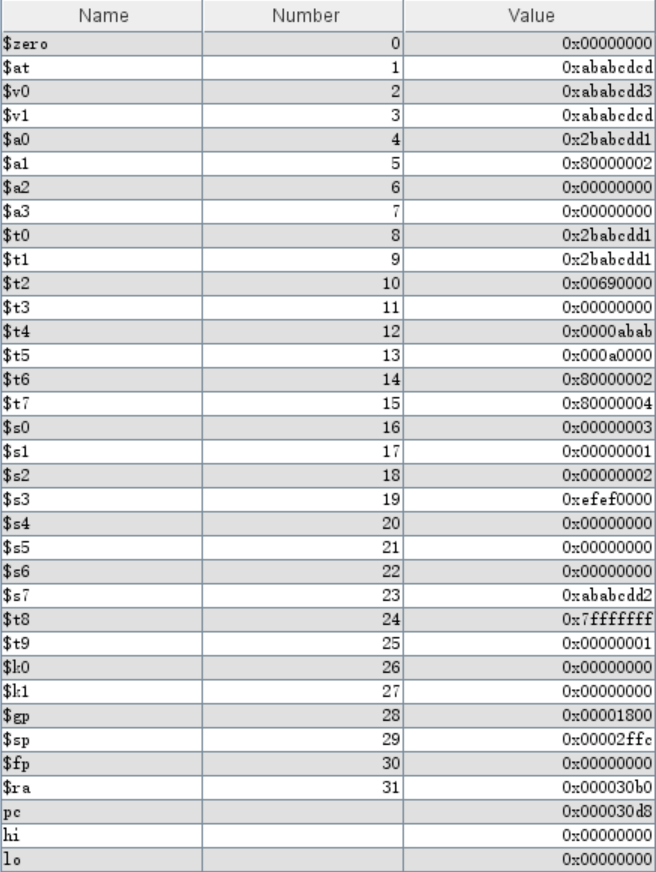
## 4.2 新增指令BGEZAL的测试程序



# 五、测试结果

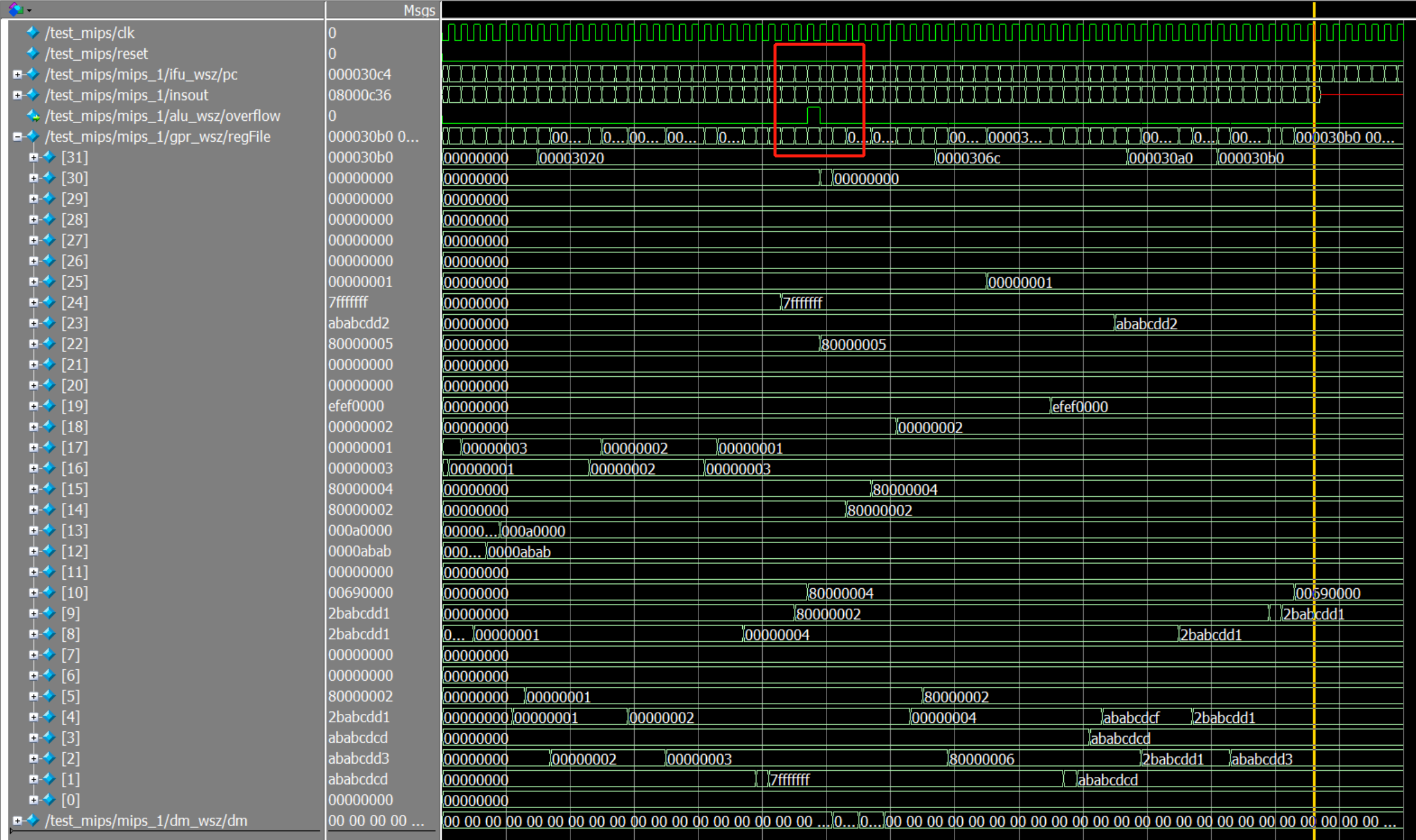
## 5.1 MIPS-Lite1指令集测试结果

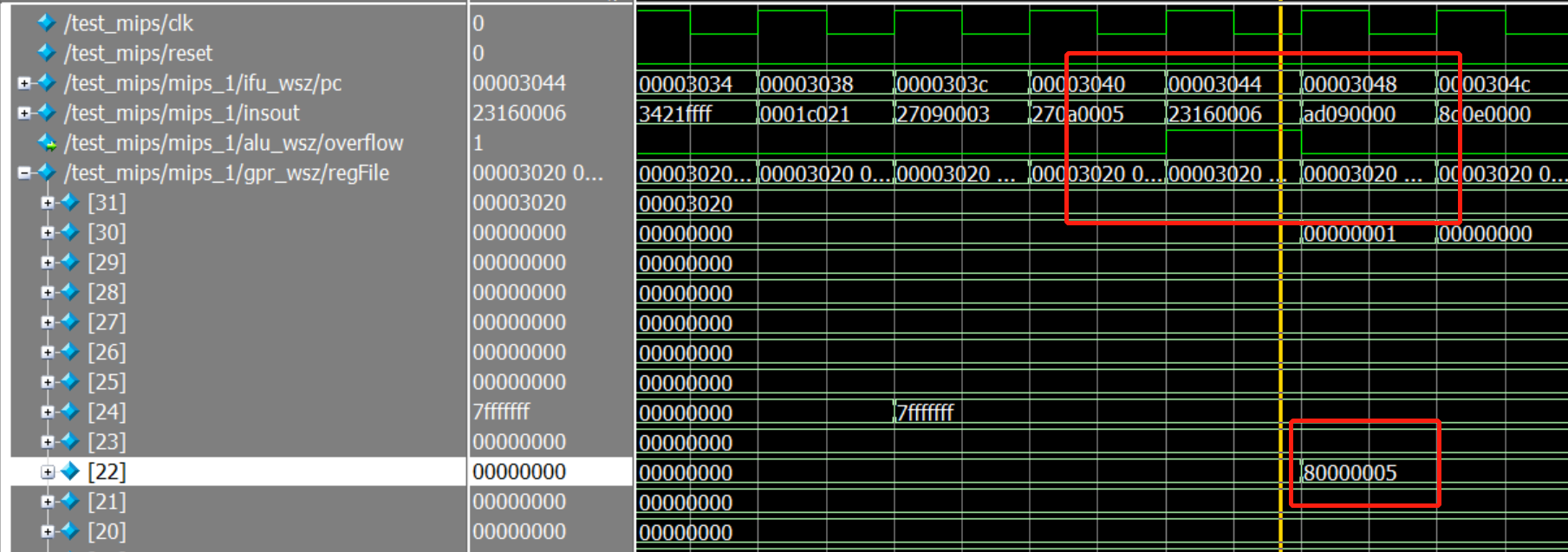
### 5.1.1 MARS中结果



### 5.1.2 Logisim中结果

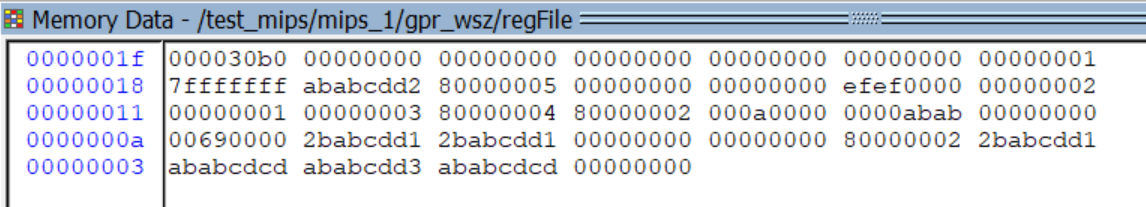
#### （1）溢出波形



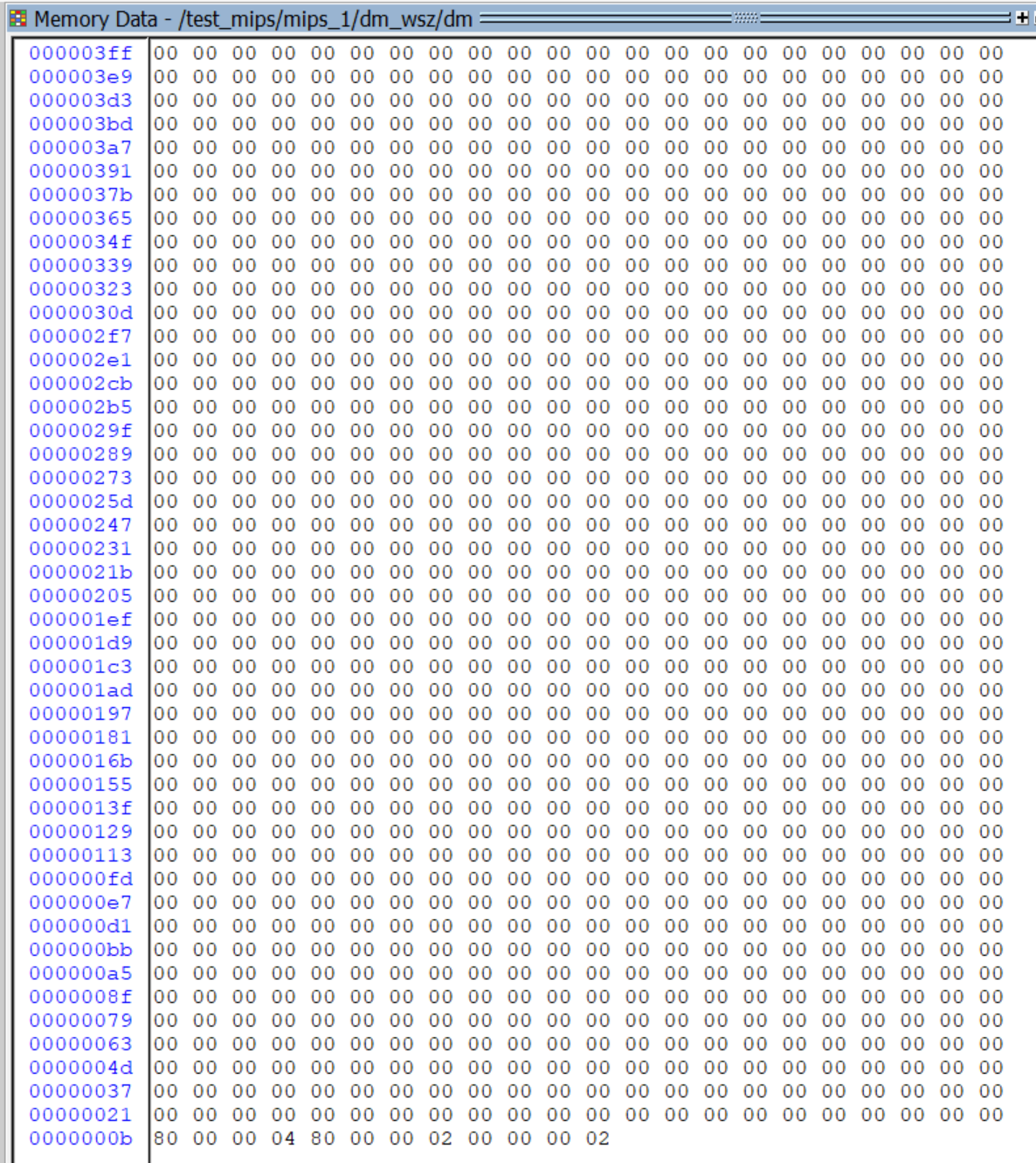


#### （2）寄存器 & 数据存储器内容

##### a. 寄存器

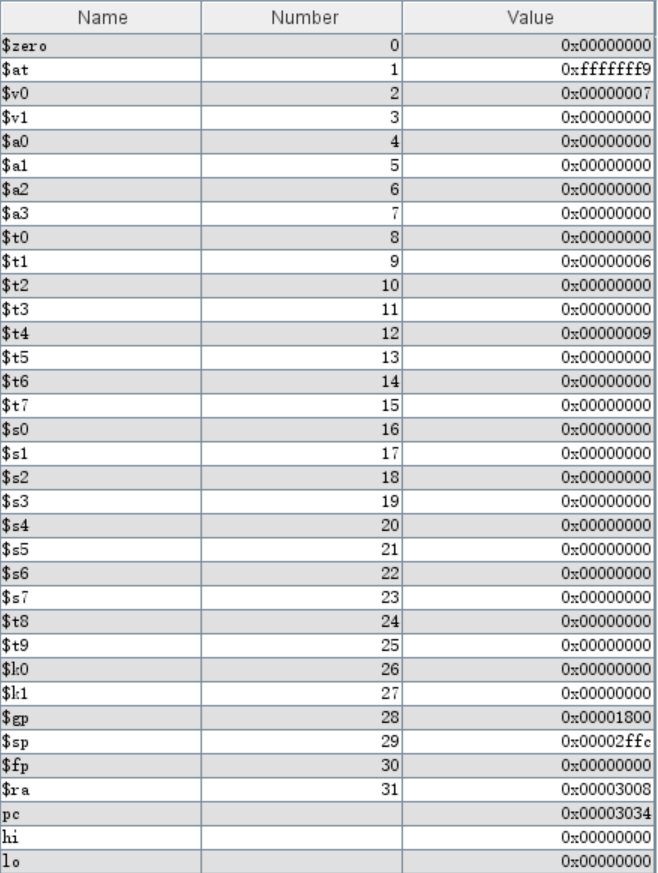


##### b. 数据存储器

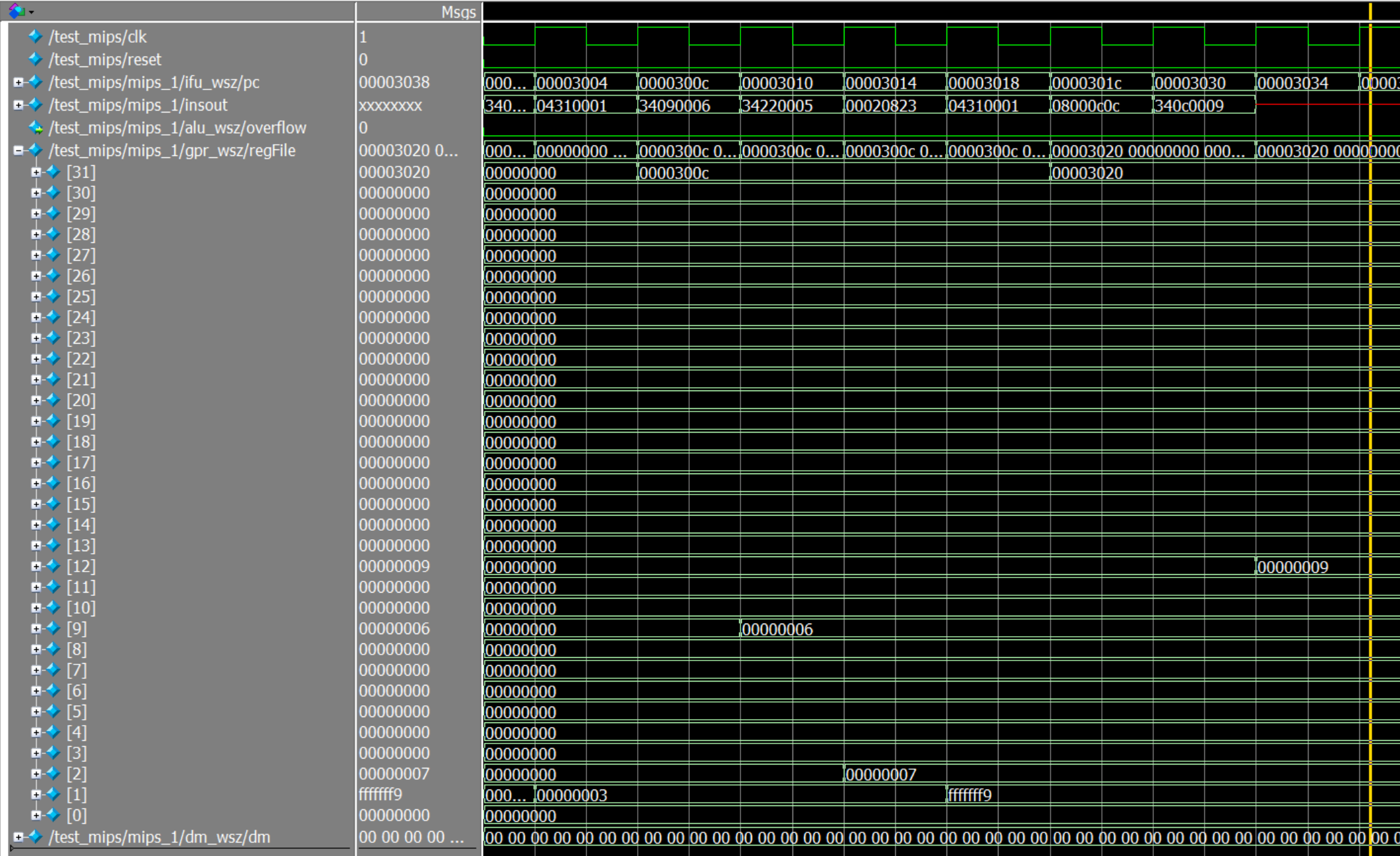


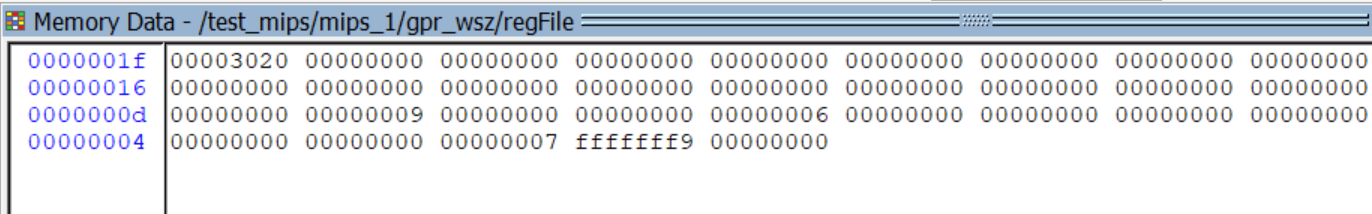
## 5.2 BGEZAL指令测试结果

### 5.2.1 MARS中结果



### 5.2.2 Logisim中结果





# 六、总结与心得体会

这次通过写verilog代码的方式，完成了单周期处理器的开发，支持MIPS-Lite1指令集，同时新添加了bgezal指令。

首先，通过这次开发，我对于开发一个设备的流程更加的清晰了。以这一次开发单周期处理器为例，在开发前，应该构思需要用到的功能，并根据需要的功能，将其模块化。在模块化的基础上，进行数据通路的设计和控制器的设计。在完成上述的设计以后，再进行verilog语言的编写，就会变得容易许多。在编写verilog时，应该注意每一个模块的输入输出的信号位宽，以及需要注意在顶层模块将各个模块连接到一起的时候，需要注意信号之间的对应问题。在本次编写中，我就曾因为模块之间的两个信号连反了，导致debug了一晚上都没有发现错误，十分的致命。这加强了我对这类错误的注意，以后在顶层模块中进行模块连接时，会更加的注意这一点。

其次，在开发过程中，我遇到的几个问题，我认为是值得总结的：

一，在完成对应指令功能的开发时，需要注意是有符号数的运算还是无符号数的运算。

二，在考虑addi的溢出问题时，是两个有符号数的加法，需要使用双符号位进行判断是否在加法过程中产生了溢出。

三，在更新pc值时，需要注意有一些指令是在pc+4的基础上更新的，这一点我在做新指令bgezal的时候忽略了，导致了pc跳转的错误。同时，bgezal指令的功能是判断当前操作数（符号扩展）是否大于等于0，也就是需要进行有符号数的判断和0之间的大小，在编写判断语句时注意需要进行的是有符号数之间的判断。

四，我发现相同类型的指令之间，其产生新pc值的逻辑有时会有些相似，如本次的bgezal指令实际上与beq指令的新pc值的产生方式是基本一致的。也就是说，其实可以进行一些变量的复用，使得程序代码更加简洁、高效。

最后，是我对此次大作业的体会。我认为通过这一次大作业，我更加深刻的理解了单周期cpu的数据通路的设计，同时对mips指令集中的代码更加的熟悉了，对于寄存器以及数据存储器中的操作变得不再害怕了。其次，我还明白了编程能力不仅仅体现在是否能使用代码完成自己设计思想的实现，更加重要的是编程的速度以及debug的能力。本次新增加指令时，我就由于改程序速度比较慢，再加上没有在规定时间内完成bug的修改，导致最后没有在规定时间内完成新增指令。这在一定程度上让我知道了需要改进的地方，我也一定会按照这个方向继续努力的。