**计算机组成原理**

**期末大作业报告**

**学 号\_\_\_\_\_\_\_\_\_20074411\_\_\_\_\_\_\_\_\_\_\_**

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

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

**提交日期\_\_\_\_\_\_2022.6.20\_\_\_\_\_\_\_\_\_\_\_\_\_**

**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:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **sya**目录

# 总体数据通路结构设计图p3

# 二、数据通路中模块的详细描述p4

# 三、 机器指令描述p20

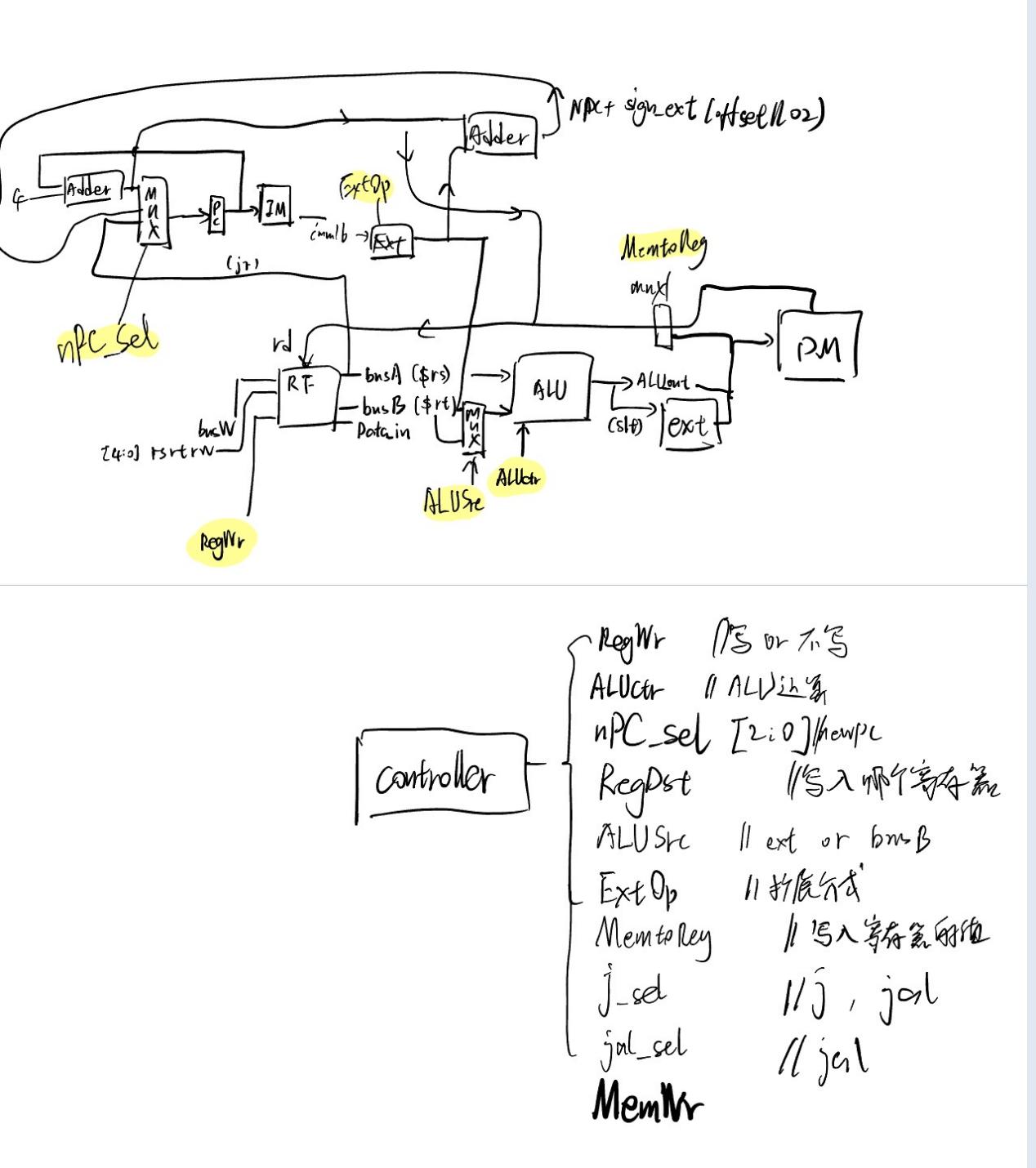
# 四、编写的测试程序p21

# 五、与测试程序相对应的测试结果截图p23

# 六、设计过程的收获、体会及总结p27

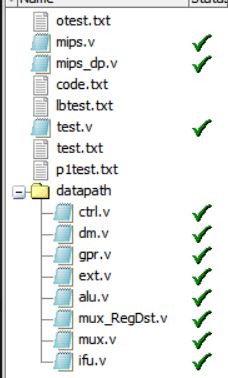
## 一、总体数据通路结构设计图

Figure1顶层设计视图



↑手绘设计思路（不包含lb的信号，lb信号通路与lw类似）

Figure2设计层次



## 二、数据通路中模块的详细描述

ALU模块：

1. 基本描述

ALU主要功能是完成计算功能。根据输入的ALUSrc和ALUctr两个控制信号选择使用进行的运算类型以及运算的两个数。在执行加法时还有额外的溢出判断，输出over信号标志溢出。

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| busA | I | 被操作数 |
| busB | I | 操作数 |
| ALUctr | I | 选择运算方式 |
| Addr[31:0] | O | 与ALU\_out相同，方便数据通路连线 |
| Zero[31:0] | O | ALU计算结果为0标志。  1：计算结果为0  0：计算结果非0 |
| over | O | ALU计算结果溢出  1：计算结果溢出  0：计算结果无溢出 |
| ALU\_out[31:0] | O | 32位MIPS指令 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 加 | ALUctr为00，输出A+B |
| 2 | 减 | ALUctr为01，输出A-B |
| 3 | 或 | ALUctr为10，输出A|| B |
| 4 | 与 | ALUctr为11，输出A&&B |
| 5 | 比大小 | ALUctr为100，如果前数小于后数输出1，否则输出0 |

IFU模块：

（1）基本描述

IFU主要功能是完成取指令和pc改变的功能。IFU内部包括PC、IM(指令存储器)以及其他相关逻辑。IFU除了能执行顺序取指令外，还能根据BEQ、j、jal、jr等指令的执行情况决定顺序取指令还是转移取指令。





（2）模块接口

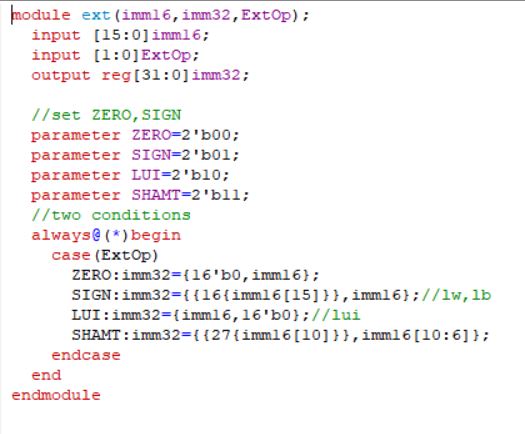
|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| nPC\_sel | I | 选择pc的下一步—newpc的值 |
| zero | I | ALU计算结果为0标志。  1：计算结果为0  0：计算结果非0 |
| j\_sel | I | 是否是j型指令的信号 |
| jValue[25:0] | I | j指令后26位address |
| jrValue[31:0] | I | 执行jr指令要跳转回的地址 |
| clk | I | 时钟信号 |
| rst | I | 复位信号。  1：复位  0：无效 |
| Jalsw[31:0] | O | Jal指令要存入寄存器的地址 |
| instruction[31:0] | O | 从im读出的32位MIPS指令 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 当复位信号reset有效时，PC被设置为0x00000000。 |
| 2 | 取指令 | 根据PC从IM中取出指令。 |
| 3 | 计算下一条指令地址 | nPC\_sel==2'b00  pcnew=pc+4  nPC\_sel==2'b01  pcnew=pc+4+extout  nPC\_sel==2'b11  Pcnew=jrValue  nPC\_sel==2'b10  zero==0  Pc=pc+4 |

EXT模块：

（1）基本描述



EXT符号拓展单元模块通过extender和多路选择器将lw、sw、beq等指令中的立即数字段的二进制补码表示的16位数实现16-bit to 32-bit拓展，包括补0补1和低位补0。

有符号数，将原有的16位数赋到32位新数的低16位，复制其符号位填满新数的高16位，实现符号拓展。

无符号数，将原有的16位数赋到32位新数的低16位，用0填满新数的高16位。

为了实现lui指令的功能，要将原有的16位数赋到32位新数的高16位，再讲低16位用0填充。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| ExtOp[1:0] | I | 立即数拓展模式选择。  00：高16位补0  01：高16位补1  10：低16位补0 |
| Imm16[15:0] | I | 16位输入 |
| Output[31:0] | O | 32位数 |

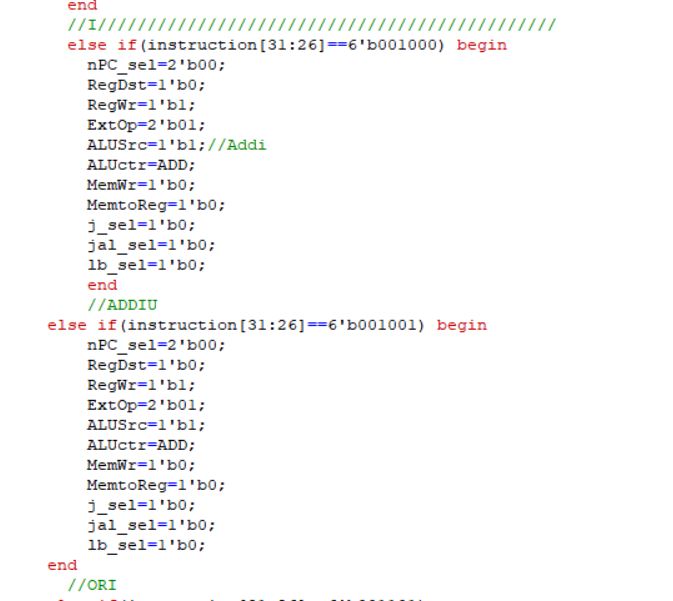
（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 高16位补0 | ExtOp为00时，输出高16位0低16位imm组成的32位数 |
| 2 | 高16位补1 | ExtOp为01时，输出高16位1低16位imm组成的32位数 |
| 3 | 低16位补0 | ExtOp为10时，输出高16位imm低16位0组成的32位数 |

控制模块：

（1）基本描述

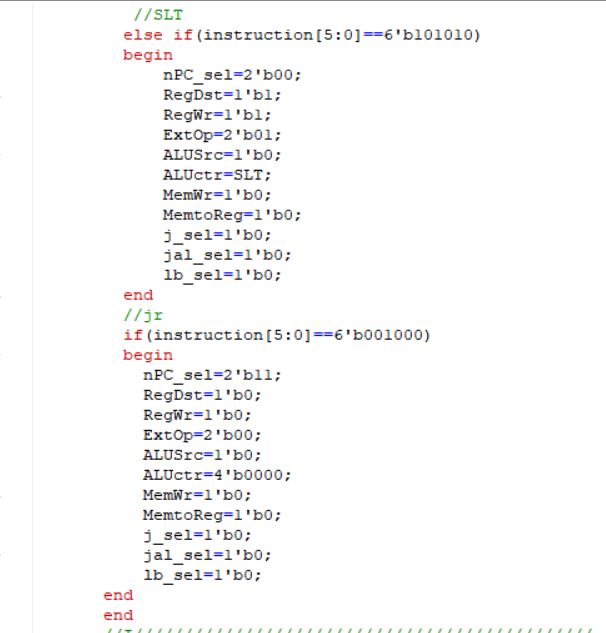
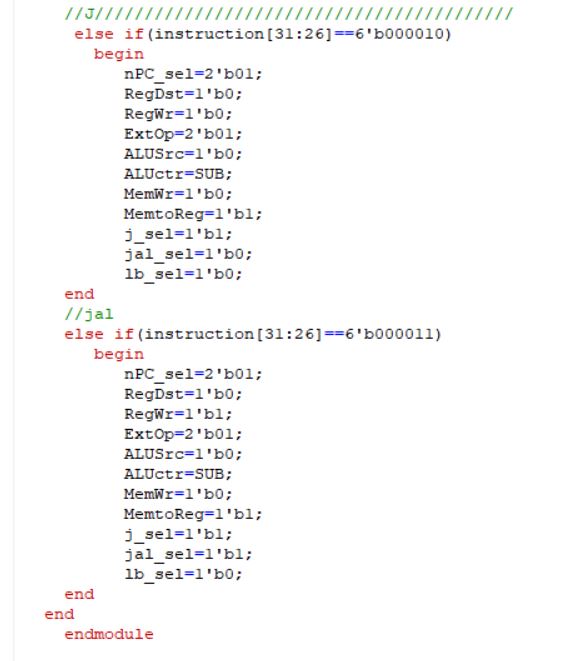












控制单元。根据指令的opcode和function内容选择产生一系列使能信号、选择信号、控制信号

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| instruction[31:0] | I | 32位指令 |
| ExtOp[1:0] | O | Ext拓展模块的控制信号，选择无符号拓展、有符号拓展和lui指令拓展 |
| nPC\_sel[1:0] | O | 选择下一条指令的地址 |
| ALUctr[3:0] | O | 选择ALU的运算方式 |
| RegDst | O | MUX\_RegDst选择器的选择信号 |
| RegWr | O | GPR寄存器堆写入使能 |
| MemtoReg | O | Dm读出数据写入寄存器堆的使能 |
| MemWr | O | 写入dm的使能 |
| ALUSrc | O | ALU第二个运算数的数据选择器的选择信号 |
| j\_sel | O | j指令的标志 |
| jal\_sel | O | Jal指令的标志 |
| lb\_sel | O | Lb指令的标志 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 译码 | 根据32位指令，将指令拆分产生并输出相应的控制信号 |

GPR模块：

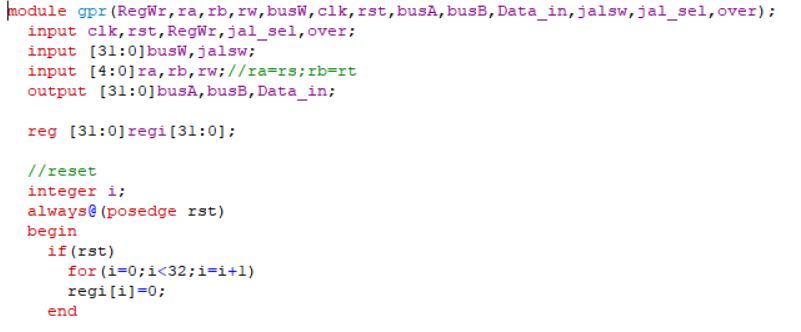
（1）基本描述

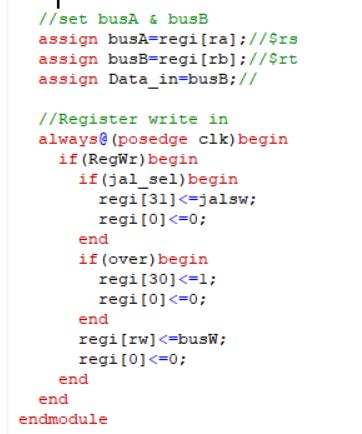
寄存器模块，两个读端口一个写端口。

寄存器写入需要写使能信号控制。写操作的输入必须在时钟边沿有效。

读操作读出以前写入的内容，而写入的内容在下一时钟周期才可读。

寄存器号的输入为5位（2^5），数据线为32位,容量1kb，可以划分为32个寄存器分别对应mips标准的寄存器。





（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| RegWr | I | 写入使能端，0不可写，1可写 |
| Jal\_sel | I | Jal指令的使能 |
| over | I | 溢出信号 |
| clk | I | 时钟信号 |
| rst | I | 为1时重置 |
| busW[31:0] | I | 要写入的数据 |
| jalsw[31:0] | I | Jal指令要存入的地址 |
| ra[4:0] | I | 选择读的地址rs |
| rb[4:0] | I | 选择读的地址rt |
| rw[4:0] | I | 写入选择端，控制写入哪个寄存器,5位控制32个 |
| busA[31:0] | O | 读出的数据1 |
| busB[31:0] | O | 读出的数据2 |
| Data\_in[31:0] | O | 等于busB，方便后续操作 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 写入 | 使能信号有效时根据选择端选择寄存器写入值 |
| 2 | 读出 | 根据选择端输入读出某寄存器的值 |

DM模块：

（1）基本描述

数据存储器，将一个大的reg分为四个存储体并行小端序储存方式。输入为写入使能、控制信号、地址和所写数据，一个输出为读出的数据。



（2）模块接口

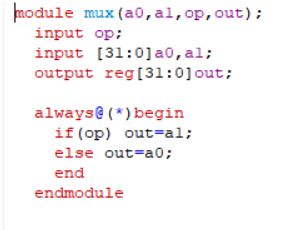
|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Addr[31:0] | I | 存入的地址，用pointer来实现按字存读数据 |
| Data\_in[31:0] | I | 要存入的数据 |
| MemWr | I | 数据存储器写入使能 |
| lb\_sel | I | Lb指令的标志 |
| clk | I | 时钟信号 |
| Data\_out[31:0] | O | 读出的数据 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 写入 | 根据使能信号和选择端将输入值写入RAM |
| 2 | 读出 | 根据选择端选择读出值 |

## MUX模块：

（1）基本描述



数据选择器

（2）模块接口

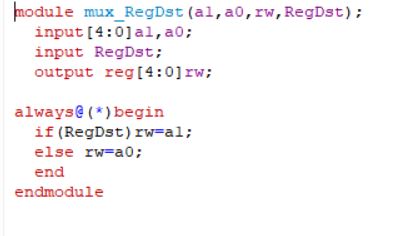
|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Op | I | 选择信号 |
| a0 | I | 第一个数据 |
| b0 | I | 第二个数据 |
| Output[31:0] | O | 被选择的信号 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择 | 选择两个输入数据中的一个作为输出 |

## MUX\_RegDst模块：

1. 基本描述



输入数据位数不同的数据选择器

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| RegDst | I | 选择信号 |
| a0 | I | 第一个数据 |
| a1 | I | 第二个数据 |
| rw[31:0] | O | 被选择的信号 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择 | 选择两个输入数据中的一个作为输出 |

## Mips模块：

（1）基本描述



顶层文件，将产生控制信号的CU控制器模块和其余模块连接起来（为了方便截图括号内信号换行处理了）

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Clk | I | 时钟信号 |
| Rst | I | 重置信号 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 连线 | 将两个大模块连接起来 |

## mips\_dp模块：

(1)基本描述



集成了除ctrl外所有用到的模块的模块，在此模块中进行连线

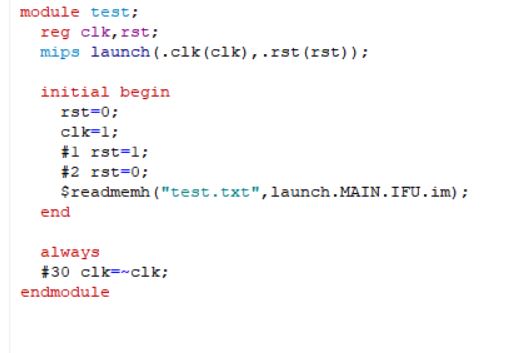
（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Clk | I | 时钟信号 |
| Rst | I | 重置信号 |
| Instruction[31:0] | O | 32位指令 |
| ExtOp[1:0] | I | Ext拓展模块的控制信号，选择无符号拓展、有符号拓展和lui指令拓展 |
| nPC\_sel[1:0] | I | 选择下一条指令的地址 |
| ALUctr[3:0] | I | 选择ALU的运算方式 |
| RegDst | I | MUX\_RegDst选择器的选择信号 |
| RegWr | I | GPR寄存器堆写入使能 |
| MemtoReg | I | Dm读出数据写入寄存器堆的使能 |
| MemWr | I | 写入dm的使能 |
| ALUSrc | I | ALU第二个运算数的数据选择器的选择信号 |
| j\_sel | I | j指令的标志 |
| jal\_sel | I | Jal指令的标志 |
| lb\_sel | I | Lb指令的标志 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 连线 | 将各个模块连接起来 |

## test：



测试用的文件，修改中间$readmemh位置的txt文件就可以转换测试用到的机器码

## 机器指令描述

|  |  |  |
| --- | --- | --- |
| 指令操作码助记符 | 操作码的机器指令代码 | 指令功能的对应关系表 |
| addu | 000000+rs+rt+rd+0000010001 | $rd = $rs op $ rt |
| subu | 000000+rs+rt+rd+00000100011 | $rd = $rs op $ rt |
| ori | 001101+rs+rt+imm | $rt = $rs op E(imm) |
| lui | 001111+00000+rt+imm | $rt=imm<<16 |
| beq | 000100+rs+rt+imm | If($rs==$rt)PC=PC+E(imm)<<2 |
| j | 000010+address | PC <- {(PC+4)[31..28],address,0,0} |
| lw | 100011+rs+rt+imm | $rt=memory[$rs+(sign-extend)] |
| lb | 100000+base+rt+offset | rt<-memory[base+offset] |
| sw | 101011+rs+rt+imm | memory[$rs+(sign-extend)]=$rt |
| addi | 001000+rs+rt+imm | $rs=$rt+imm |
| addiu | 001001+rs+rt+imm | $rt=$rs+imm(unsigned) |
| slt | 000000+rs+rt+rd+00000+101010 | Rd=(rs<rt)?1:0 |
| jal | 000011+address | $31=pc;pc={(pc+4)[31:28],addr,00} |
| jr | 000000+rs+00000+00000+00000+001000 | PC <- （$31） |

## 编写的测试程序

这里采用老师给出的test测试文件，自己编写的溢出文件，以及检查时写的lb测试文件

Test：

34100001

34110003

34080001

340cabab

3c0d000a

00102021

00082821

0c000c32

00028021

02288823

1211fffa

34080004

3c017fff

3421ffff

0001c021

27090003

270a0005

23160006

ad090000

8d0e0000

ad0a0004

8d0f0004

ad04fffc

8d12fffc

00082021

00092821

0c000c32

0148c82a

13200018

0184a02a

12800001

3c0cffff

34000001

3c13efef

3c01abab

3421cdcd

00011821

24640002

20770005

0c000c32

00024021

00082021

00092821

0c000c32

00024821

01004821

3c0a0069

11090001

1000fff4

08000c36

00851021

21801234

03e00008

201a5678

溢出：

3c017fff

34210000

00012821

3c010000

3421ffff

00a12821

20a50001

Lb：

ori $5,$0,0x00001234

sw $5,8($1)

lw $7,8($1)

lb $8,8($1)

lb $9,9($1)

机器码：

34051234

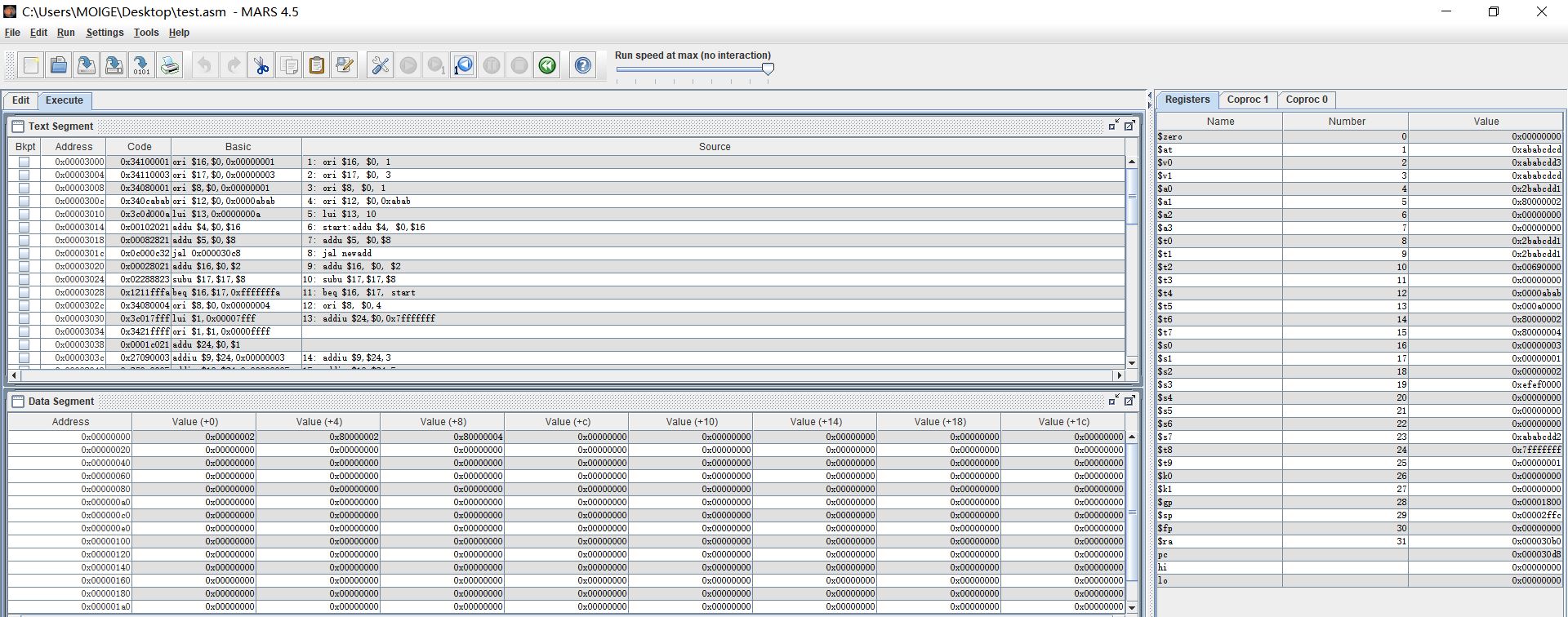
ac250008

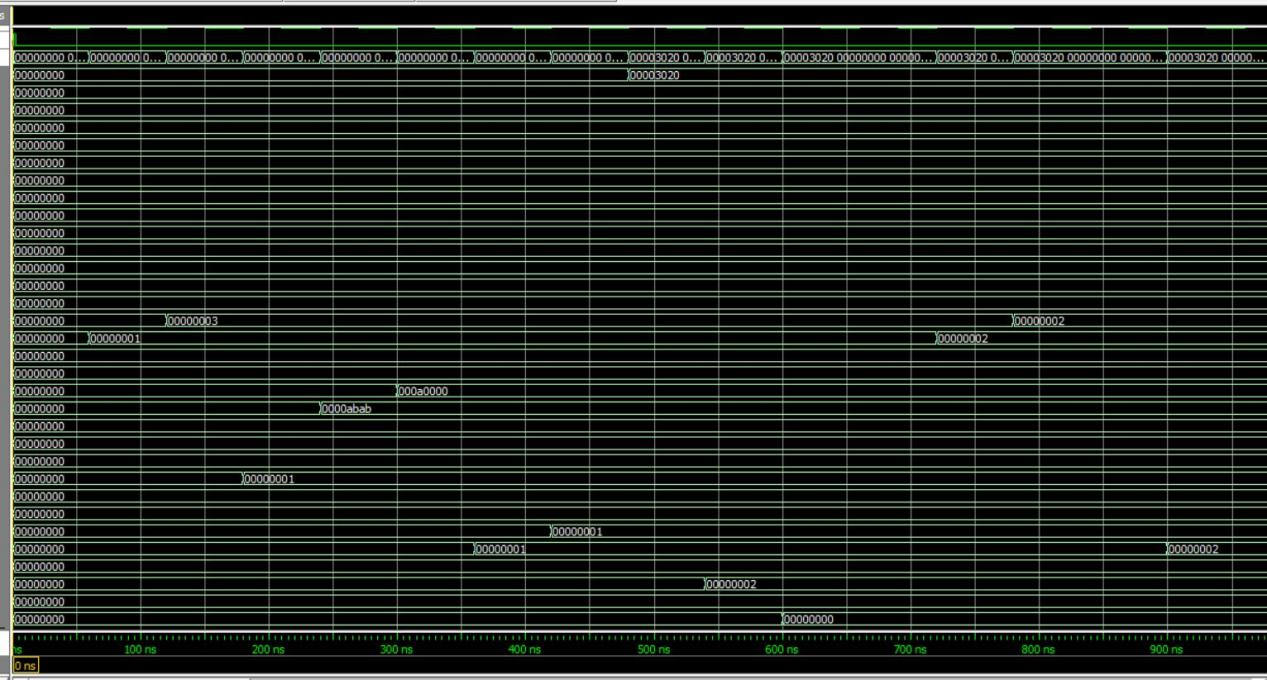
8c270008

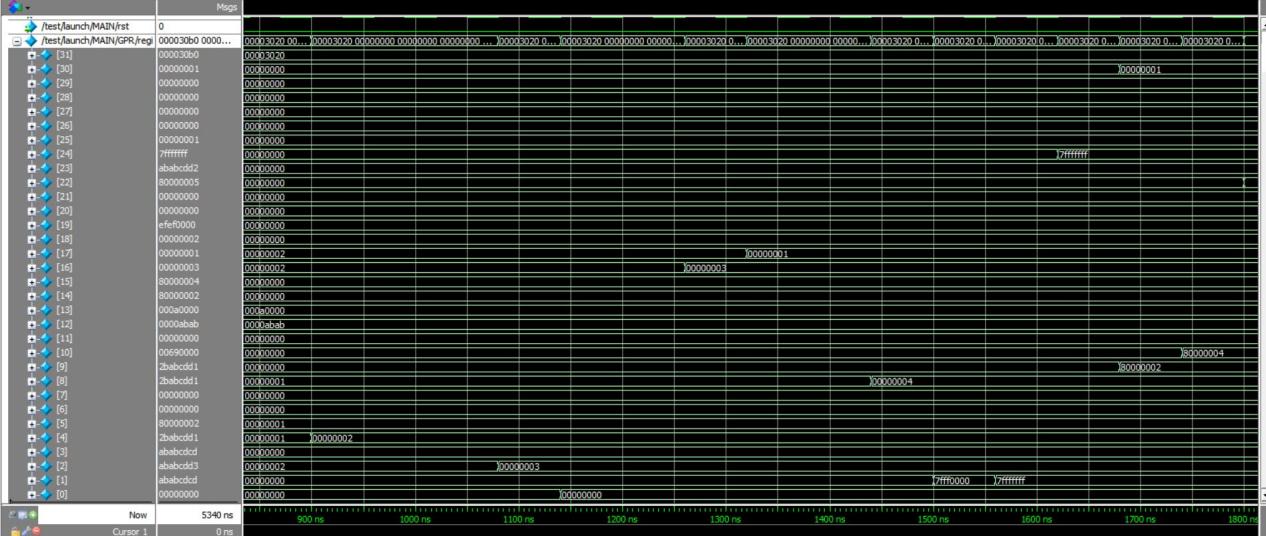
80280008

80290009

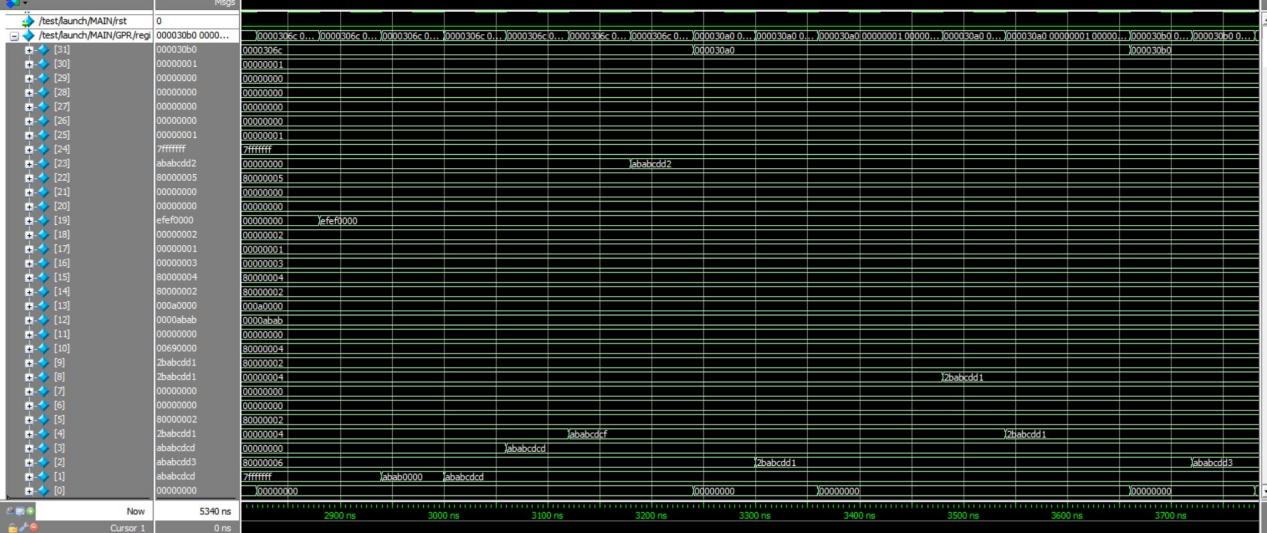
## 与测试程序相对应的测试结果截图

每条指令的波形图，其中可以看到寄存器相应的变化，与mars中的仿真结果相同

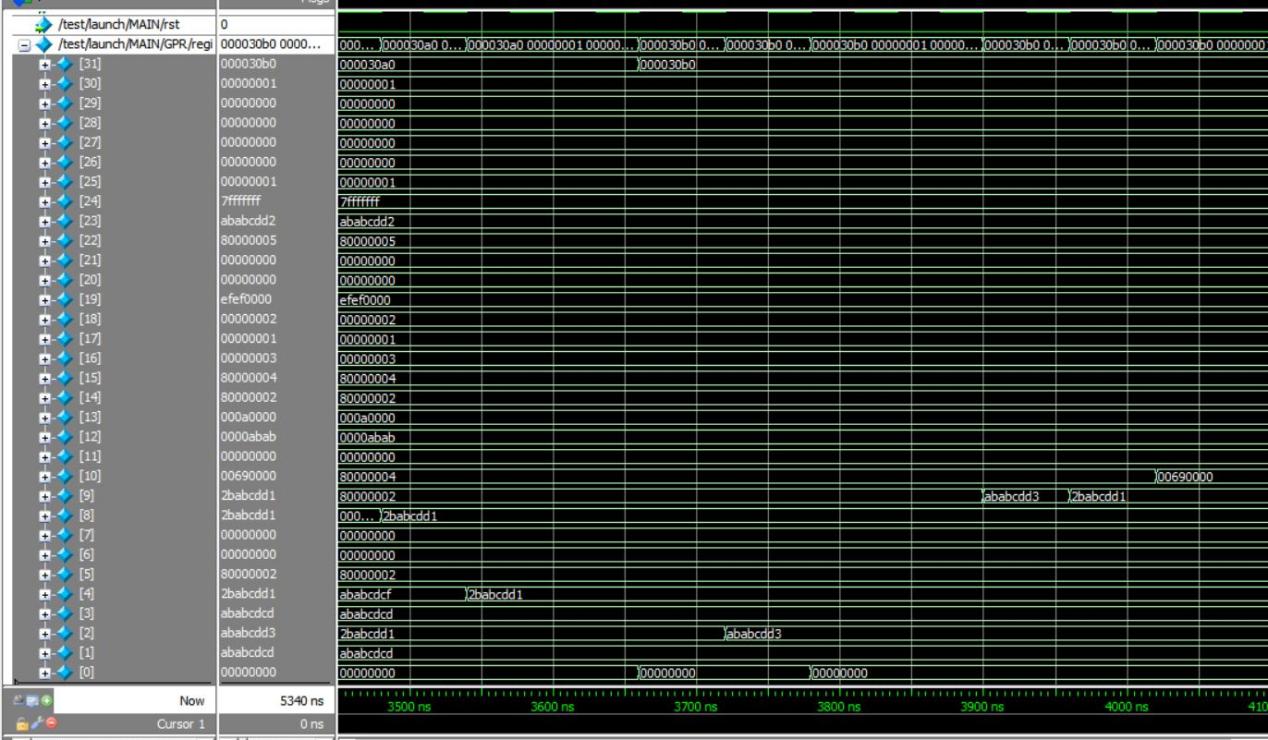


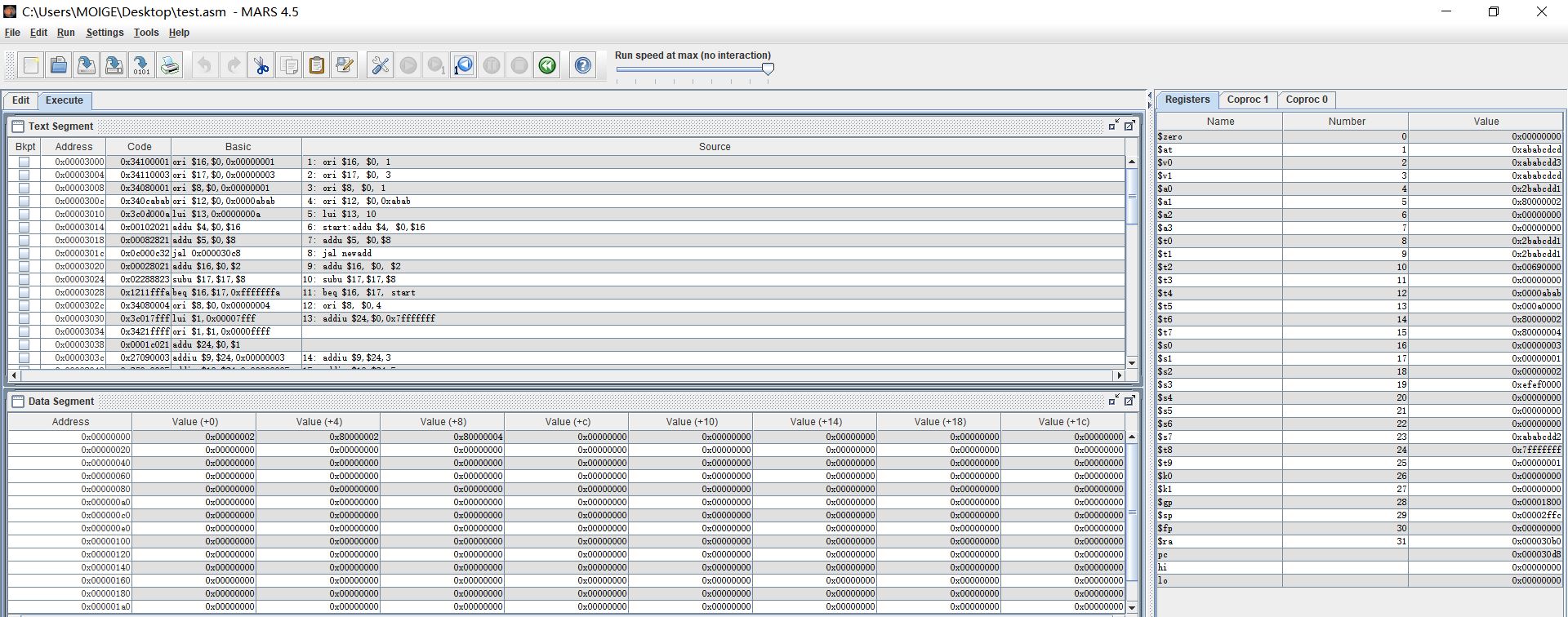


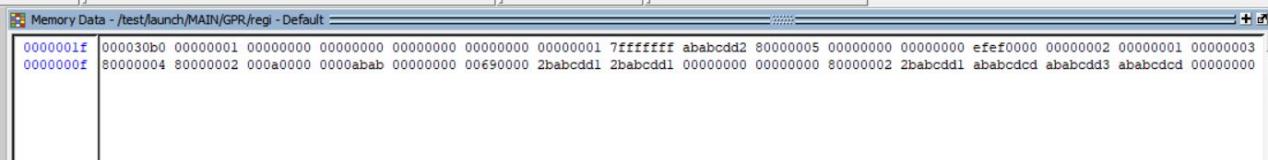


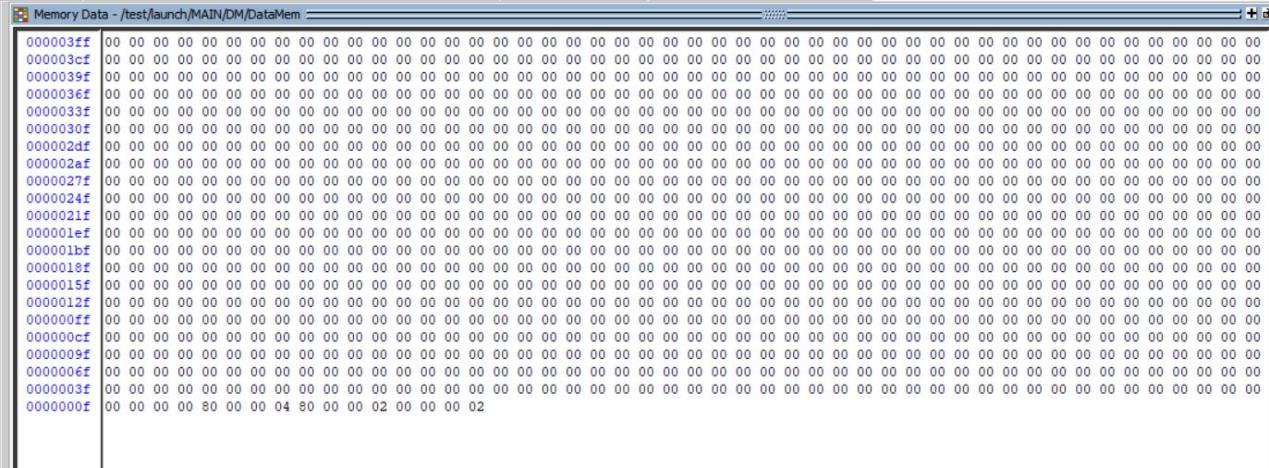




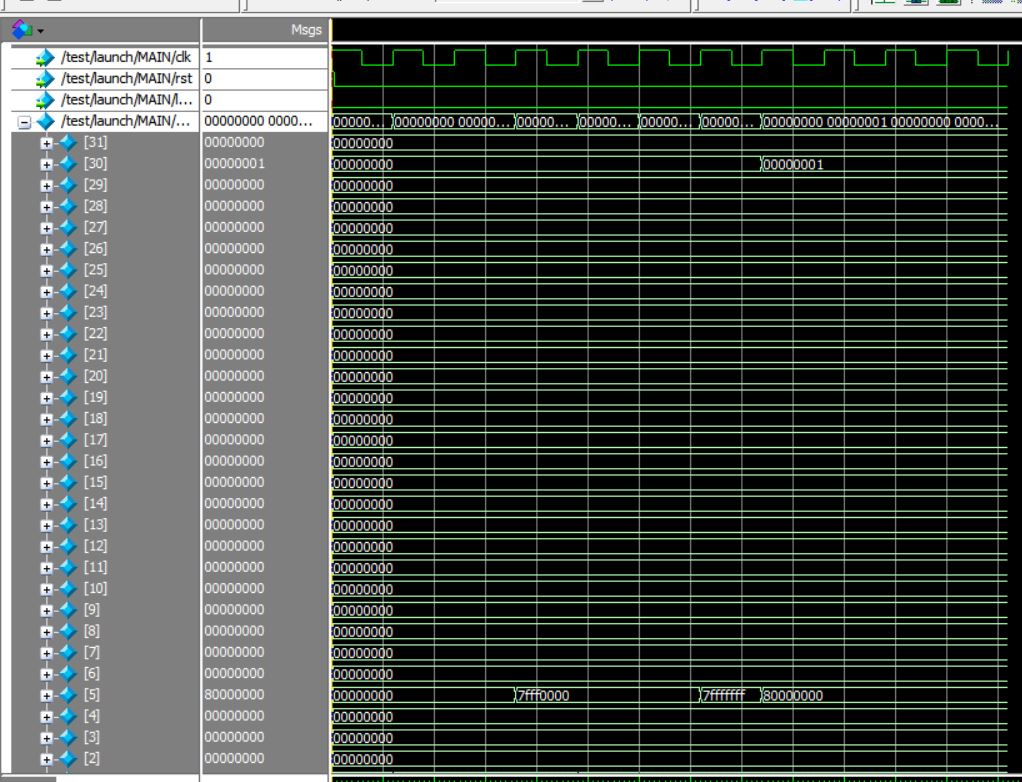


最终寄存器以及dm中结果相同

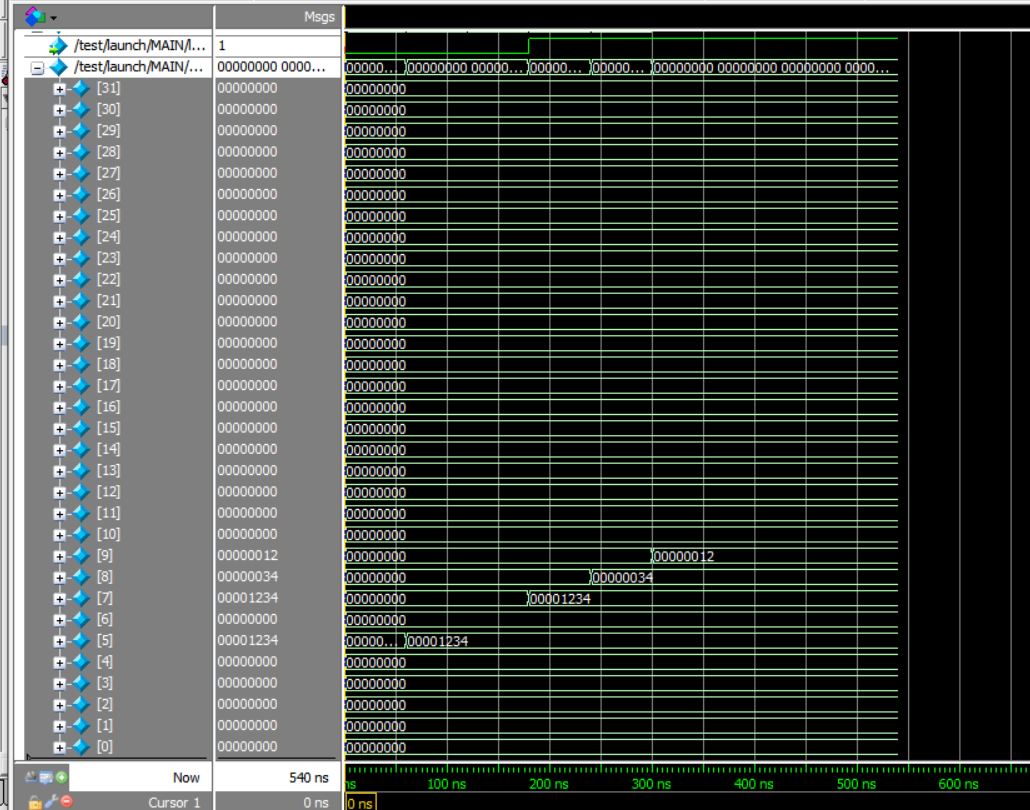




溢出的仿真



lb指令仿真



## 设计过程的收获、体会及总结

开学的时候看到计算机组成原理课既有大作业又有课设，就感觉压力会很大。现在来看虽然压力大，但是内容还是非常有趣的，自己一步步完成处理器，看到指令能运行的瞬间还是非常开心的，这促使、激励着我一直坚持学下去。

在学习写处理器的过程中，我也看到了很多别的学校的专业课内容，发现计算机这个大类专业的课程其实都是精心安排过的，一步步上课最终我们可能能够实现从操作系统到硬件设计都有深入了解，能够自己做出很多有趣的东西再将它们联系起来。这让我对我的专业有了更多的了解，也有了更多的兴趣。

虽然期末考试变为了大作业，但是我在完成之后感觉到也许对于原理课这样的课程，大作业比期末考试要更能让人学到东西，经常有人说自己再造轮子才能完全掌握知识嘛。期末考试也许可以糊弄糊弄突击一下，但是做大作业还是自己掌握了知识动脑去做效率要更高一点。在实践的过程中我们就更深刻地掌握了所学的知识。

总之，这系列的大作业给我留下了非常深刻的印象。大作业让我对计算机相关硬件、软件，相关职业，甚至类似的学科和实验课都有了更多了解，也让我明白了面对困难一定要坚持下去，这样才能成功。