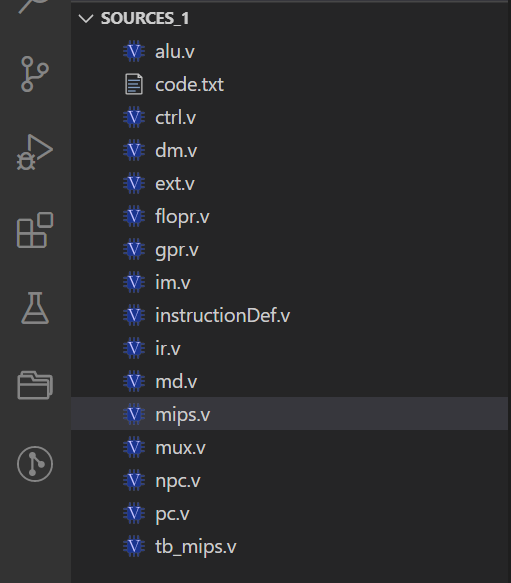
**实验报告**

姓名：孙延浩 学号：21181225

1. 整体概述

该多周期处理器采用verilog开发，支持以下多条指令：{LB,LBU,LH,LHU,LW,SB,SH,SW,ADD,ADDU,SUB,SUBU,MULT,MULTU,DIV,DIVU,SLL,SLT,SRL,SRA,SLLV,SRLV,SRAV,AND,OR,XOR,NOR,ADDI,ADDIU,ANDI,ORI,XORI,LUI,SLTI,SLTIU,BEQ,BNE,BLEZ,BGTZ,BLTZ,BGEZ,J,JAL,JALR,JR,MFHI,MFLO,MTHI,MTLO},处理器为多周期设计。

目录结构如图所示：



其中mips.v为顶层结构，tb\_mips.v为测试文件,code.txt为IM中要读入的内容，instructionDef.v为指令宏定义部分。

2.模块定义

2.1 alu

（1）基本描述

alu是该多周期处理器的运算逻辑单元，主要功能是完成需要处理的运算，并将运算结果传输给需要的部件。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| a[31:0] | I | 32位数据1 |
| b[31:0] | I | 32位数据2 |
| aluop[3:0] | I | 0000：ADD  0001：SUB  0010：ORI/OR  0011: SLT/SLTI  0100: SLL  0101: SRL  0110: SRA  0111: SLLV  1000: SRLV  1001: SRAV  1010: AND/ANDI  1011: XOR/XORI  1100: NOR  1101: DIRECTLY OUT  1110: SLTIU |
| aluout[31:0] | O | Alu输出 |
| s[4:0] | I | 移位指令相关字段表示移动位数的部分作为输入 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 对两个操作数进行加 | aluout<=a+b |
| 2 | 对两个操作数进行减 | aluout<=a-b |
| 3 | 对两个操作数进行或 | aluout<=a|b |
| 4 | 判断a是否小于b | 如果a小于b置1，否则置0 |
| 5 | 逻辑左移 | aluout<=b<<s |
| 6 | 逻辑右移 | aluout<=b>>s |
| 7 | 算术右移 | 将b算术右移s位 |
| 8 | 可变逻辑左移 | aluout<=(b<<a[4:0]) |
| 9 | 可变逻辑右移 | aluout<=(b>>a[4:0]) |
| 10 | 可变算术右移 | 将b算术右移a[4:0]位 |
| 11 | 与 | aluout<=a&b |
| 12 | 亦或 | aluout<=a^b |
| 13 | 或非 | aluout<=~(a|b) |
| 14 | 直出 | aluout<=a |
| 15 | SLTIU | 小于置1，否则置0 |

2.2 pc

（1）基本描述

PC主要功能是完成输出当前指令地址并保存下一条指令地址。复位后，PC指向0x00003000

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| rst | I | 复位信号 |
| PCwr | I | 控制是否写入PC |
| nPC[31:2] | I | nPC作为输入 |
| pcout[31:2] | O | 当前pc输出 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | rst==1，pc=0x00003000 |
| 2 | 保存nPC并输出 | 在每个clk上升沿保存nPC并输出 |

2.3 Ctrl

（1）基本描述

Ctrl部件用于通过分析当前有限状态机状态和指令决定状态机的下一状态，并根据当前状态和当前指令产生相应的控制信号

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| instr[31:0] | I | 当前指令 |
| clk | I | 时钟信号 |
| rst | I | 重置信号 |
| aluout[31:0] | I | alu的32位输出结果 |
| done | I | 用于判断乘除是否完成 |
| instr[31:0] | I | 指令码 |
| aluout[31:0] | I | alu的32位输出 |
| extop[1:0] | O | 对Extender的拓展方式进行规定 |
| aluop[1:0] | O | 控制ALU执行的运算 |
| memwrite | O | 数据存储器写使能 |
| regwrite | O | 寄存器写使能 |
| memtoreg[2:0] | O | 控制寄存器写是来自ALU结果还是DM还是PC还是乘除法部件 |
| alusrc | O | 控制是立即数参与运算还是寄存器rt |
| regdst[1:0] | O | 用于控制写入哪个寄存器（rd/rt/ra） |
| npcop | O | 用于控制PC的下一个值 |
| IRwr | O | 用于控制是否写入IR |
| PCwr | O | 用于控制是否写PC |
| mdop[2:0] | O | 用于控制乘除法部件进行的操作 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 产生信号 | 分析当前指令以及状态机状态并产生相应的控制信号 |
| 2 | 决定状态机下一状态 | 根据此时状态机的状态以及指令来决定下一状态 |

2.4 ext

（1）基本描述

ext部件用于对位数不足的数据进行位数拓展

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| offset[15:0] | I | 16位数据输入 |
| extop[1:0] | I | 00：高位不带符号拓展  01：高位带符号拓展  10：低位拓展0 |
| eout[31:0] | O | 32位数据输出 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 高位不带符号拓展 | 在16位数据前16位填0 |
| 2 | 高位带符号拓展 | 在16位数据前16位填它的符号位 |
| 3 | 低位拓展0 | 在16位数据后16位填充0 |

2.5 gpr

（1）基本描述

gpr放有32个寄存器，用于处理外界读写寄存器的请求，并给出对应的输出结果

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| a1[4:0] | I | 读寄存器地址1 |
| a2[4:0] | I | 读寄存器地址2 |
| a3[4:0] | I | 写寄存器地址 |
| rst | I | 复位 |
| wd[31:0] | I | 写入数据的输入 |
| clk | I | 时钟信号 |
| regWrite | I | 读写控制信号 |
| rd1[31:0] | O | 32位输出1 |
| rd2[31:0] | O | 32位输出2 |
| Ra[31:0] | O | 31号寄存器ra的值 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 写寄存器 | 在有写寄存器信号时往目标寄存器写入 |
| 2 | 读寄存器 | 输出目标寄存器中储存的值 |

2.6 dm

（1）基本描述

数据存储器，能读出或写入相关数据。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| din[31:0] | I | 要写入的数据 |
| addr[11:2] | I | 写入DM的地址 |
| we | I | 写使能 |
| clk | I | 时钟信号 |
| dmop | I | 用于判断dm读数据的方式  000:sw  001:后半字写入din的后16位  010:前半字写入din的后16位  011:最后一个字节写入din的后8位  100:倒数第二个字节写入din的后8位  101:第二个字节写入din的后八位  110:第一个字节写入din的后八位 |
| dout[31:0] | O | 32位数据输出 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 存储数据 | 将相关数据在写使能时写入 |
| 2 | 读取数据 | 从存储器中读出数据 |

2.7 im

（1）基本描述

指令存储器，从其中读取预存好的指令。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| addr[11:2] | I | 要读取的指令地址 |
| dout[31:0] | O | 32位指令输出 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读取指令 | 从指令寄存器读取指令 |

2.8 mux

2.8.1 aluScrMux

（1）基本描述

用于选择运算数的多路选择器，根据信号选择运算数的来源并输出

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| ain[31:0] | I | 来自立即数拓展的32位输入 |
| rt[31:0] | I | 来自rt的32位输入 |
| alusrc | I | 选择器信号：  0:rt作为多路选择器结果  1:ain作为多路选择器结果 |
| aout[31:0] | O | 多路选择器输出 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择运算数 | 根据信号选择alu的一个运算数 |

2.8.2 regDstMux

（1）基本描述

用于选择写回的目标寄存器，将目标寄存器的寄存器号输出

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| rt[4:0] | I | rt的寄存器号 |
| rd[4:0] | I | rd的寄存器号 |
| regdst[1:0] | I | 选择器信号：  00:rout<=rt  01:rout<=rd  10:rout<=31 |
| rout[4:0] | O | 输出的寄存器号 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择写回的目标寄存器 | 将选择好的目标寄存器的寄存器号输出 |

2.8.3 MemtoRegMux

（1）基本描述

用于选择写入寄存器的值的多路选择器，选出相应的值并输出

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| aluout[31:0] | I | alu的32位输出作为输入 |
| dmout[31:0] | I | dm的32位输出作为输入 |
| pcout[31:2] | I | pc的30位输出作为输入 |
| memtoreg[2:0] | I | 选择信号:  000:mout<=aluout  001:mout<=dmout  010:mout<={pcout+1,2’b00}  011:mout<=hiOut  100:mout<=loOut |
| hiOut[31:0] | I | HI寄存器的内容 |
| loOut[31:0] | I | Lo寄存器的内容 |
| mout[31:0] | O | 写入寄存器的值 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择写入寄存器的值 | 选择写入寄存器的值的多路选择器，选出相应的值并输出 |

2.8.4 dmMux

（1）基本描述

用于产生用来选择内存具体位置信号的多路选择器，根据指令输出对应的信号。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| addr[1:0] | I | 要求写入内存地址的后两位，用于辨别具体要产生哪个信号 |
| op[5:0] | I | 6位指令码 |
| dmop[2:0] | O | 输出3位dm写入时的选择信号:  001:后半字写入数据的后16位  010:前半字写入数据的后16位  011:最后一个字节写入数据的后8位  100:倒数第二个字节写入数据的后8位  101:第二个字节写入数据的后八位  110:第一个字节写入数据的后八位 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择数据写入内存的长度和位置 | 根据指令和给出的地址，判断具体最终写入的数据长度和位置，输出对应信号 |

2.8.5 rdmMux

（1）基本描述

用于处理dm数据读出结果的多路选择器，根据指令处理从dm读出的32位数据，并得到相关数据输出。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| addr[1:0] | I | 要求写入内存地址的后两位，用于辨别具体要产生哪个信号 |
| op[5:0] | I | 6位指令码 |
| din[31:0] | I | 从dm读出的32位数据 |
| rout[31:0] | O | 根据指令选择相应字段拓展出的32位结果 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 根据指令筛选dm读出的内容 | 根据指令和给出的地址，判断具体要读取哪一部分内容，并对其拓展到32位 |

2.9 npc

（1）基本描述

生成下一个pc的值并传给pc，使得pc在每一个时钟周期后指向应在的位置

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| pc[31:2] | I | 当前30位pc输入 |
| npcop[1:0] | I | 选择信号，传入多路选择器  00:npc<=pc+1  01:npc<=(beq跳转成功)?跳转位置:pc+1  10:npc<={pc[31:28],mem[25:0]}  11:npc<=rs[31:2] //jr/jalr时npc为寄存器rs内容 |
| rs[31:0] | I | 编号为rs寄存器中的内容 |
| mem[31:0] | I | 32位内存单元输入 |
| nout[31:2] | O | 下一pc值作为输出 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 判断pc转移状况 | 通过信号判断是否转移 |
| 2 | 得到下一个pc值 | 连接多路选择器后得到下一pc值并输出 |

2.10 flopr

（1）基本描述

用于处理多周期数据问题而构建的暂存器模型

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号输入 |
| rst | I | 重置信号输入 |
| fin[31:0] | I | 32位输入 |
| fout[31:0] | O | 32位输出 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 重置 | rst为1时重制暂存器 |
| 2 | 写入 | 时钟上升沿写入数据并保存 |

2.11 IR

（1）基本描述

用于暂存读出的指令，并在信号有效时写入新指令。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号输入 |
| rst | I | 重置信号输入 |
| IRwr | I | IR写入指示信号 |
| imout[31:0] | I | 读出的指令作为32位输入 |
| instr[31:0] | O | 32位指令输出 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 重置 | rst为1时重制IR寄存器 |
| 2 | 写入 | 时钟上升沿写入新的指令 |

2.12 md

（1）基本描述

用于进行与乘除法有关运算的部件，并且封装了HI,LO两个寄存器，方便写入和读出。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| mdop[2:0] | I | 用于选择要进行的操作  000:MULT  001:MULTU  010:DIV  011:DIVU  100:MTHI  101:MTLO |
| a[31:0] | I | 运算数1 |
| b[31:0] | I | 运算数2 |
| hiOut[31:0] | O | HI内容作为输出 |
| loOut[31:0] | O | LO内容作为输出 |
| done | O | 乘除法完成信号 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 进行乘除有关的运算 | 进行各种乘除运算，将结果保存在HI和LO |
| 2 | 输出HI，LO | 将HI,LO的值以hiOut,loOut的形式送出 |
| 3 | 写入HI,LO | 将rs的值写入HI,LO中 |

2.13 mips

（1）基本描述

将所有的部件通过线网连接起来，作为顶层模块完成电路构建，在rst为1时初始化线路，在时钟脉冲来临时运转整个模块。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 作为时钟脉冲控制电路的运转 |
| rst | I | 作为置位信号 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 连接所有器件 | 通过线网让电路运作起来 |
| 2 | 初始化 | 在rst为1时初始化某些值 |
| 3 | 给予时钟脉冲 | 接收到外界的时钟脉冲后传递给线路中的器件 |

2.14 tb\_mips(testbench)

（1）基本描述

测试模块，用于合理地给出clk和rst信号

（2）模块接口

无接口

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 初始化 | 最初时给出rst上升沿，对具体部件进行置位 |
| 2 | 给出脉冲 | 每隔固定时间给出时钟脉冲 |

3.测试程序原理及结果

3.1测试源代码

测试程序test.asm:

# Test File for 49 Instruction, include:

# 1. Subset 1:

# ADD/ADDU/SUB/SUBU/SLL/SRL/SRA/SLLV/SRLV/SRAV/AND/OR/XOR/NOR/

# SLT 14

# 2. Subset 2:

# ADDI/ADDIU/ANDI/ORI/XORI/LUI/SLTI/SLTIU 8

# 3. Subset 3:

# LB/LBU/LH/LHU/LW/SB/SH/SW 8

# 4. Subset 4:

# BEQ/BNE/BGEZ/BGTZ/BLEZ/BLTZ 6

# 5. Subset 5:

# J/JAL/JR/JALR 4

# 6. Subset 6:

#MULT/MULTU/DIV/DIVU/MFHI/MFLO/MTHI/MTLO 42

##################################################################

### Make sure following Settings :

# Settings -> Memory Configuration -> Compact, Data at address 0

.data

.globl array cnt

array:

.word 0:16 # array of 16 words

cnt:

.word 0 # counter of Branch

.text

##################

# Test Subset 2 #

ori $v0, $0, 0x1234

lui $v1, 0x9876

addi $a0, $v0, 0x3456

addiu $a1, $v1, -1024

xori $a2, $v0, 0xabcd

sltiu $a1, $a0, 0x34

sltiu $a1, $v0, -1

andi $a3, $a2, 0x7654

slti $t0, $v1, 0x1234

##################

# Test Subset 1 #

subu $t0, $v1, $v0

xor $t1, $t0, $v1

addu $t2, $t1, $t0

add $t2, $t2, $v0

sub $t3, $t2, $v1

nor $t4, $t3, $t2

or $t5, $t3, $t2

and $t6, $t3, $t2

slt $s3, $t5, $t4

### Test for shift

sll $t0, $t0, 3

srl $t1, $t0, 16

sra $t2, $t0, 29

addiu $t3, $0, 0x3410 # pay attention to register shift

sllv $t4, $t0, $t3

srlv $t5, $t0, $t3

srav $t6, $t0, $t3

##################

# Test Subset 3 #

addu $a0, $v0, $v1

la $sp, array

### Test for store

sw $a0, 0($sp)

sw $a0, 4($sp)

sw $a0, 8($sp)

sh $t0, 4($sp)

sh $t1, 10($sp)

sb $t2, 7($sp)

sb $t0, 9($sp)

sb $t1, 8($sp)

### Test for load

lw $t0, 0($sp)

sw $t0, 12($sp)

lh $t1, 2($sp)

sw $t1, 16($sp)

lhu $t1, 2($sp)

sw $t1, 20($sp)

lb $t2, 3($sp)

sw $t2, 24($sp)

lbu $t2, 3($sp)

sw $t2, 28($sp)

lbu $t2, 1($sp)

sw $t2, 32($sp)

##################

# Test Subset 4 #

la $sp, cnt

sw $0, 0($sp)

and $v0, $0, $t0

bne $t0, $t1, \_lb1

addiu $v0, $v0, 1

\_lb1:

bgtz $t0, \_lb2

addiu $v0, $v0, 1

\_lb2:

blez $t0, \_lb3

addiu $v0, $v0, 1

\_lb3:

bltz $t1, \_lb4

addiu $v0, $v0, 1

\_lb4:

bgez $t0, \_lb5

addiu $v0, $v0, 1

\_lb5:

beq $t1, $t2, \_lb6

addiu $v0, $v0, 1

\_lb6:

sw $v0, 0($sp)

##################

# Test Subset 5 #

la $sp, cnt

lw $v0, 0($sp)

j \_target

addu $v0, $v0, $t0

\_target:

addiu $v0, $v0, 1

jal F\_Test\_JAL

addu $a1, $a1, $v0

#################

#Test Subset 6 #

mult $t0,$t1

mfhi $s0

mflo $s1

multu $t1,$t2

mfhi $s2

mflo $s3

div $t0,$t1

mfhi $s4

mflo $s5

divu $t1,$t2

mfhi $s6

mflo $s7

mthi $t1

mtlo $t2

\_loop:

j \_loop # Dead loop

F\_Test\_JAL:

la $sp, array

sw $ra, 40($sp)

ori $v0, $v0, 0x5500

la $t1, F\_Test\_JALR

jalr $t1

la $sp, cnt

sw $v0, 0($sp)

la $sp, array

lw $ra, 40($sp)

jr $ra

F\_Test\_JALR:

la $sp, array

sw $ra, 44($sp)

ori $v0, $v0, 0x50

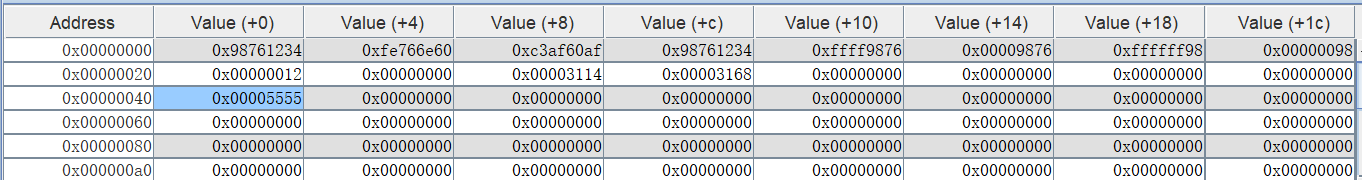
jr $ra

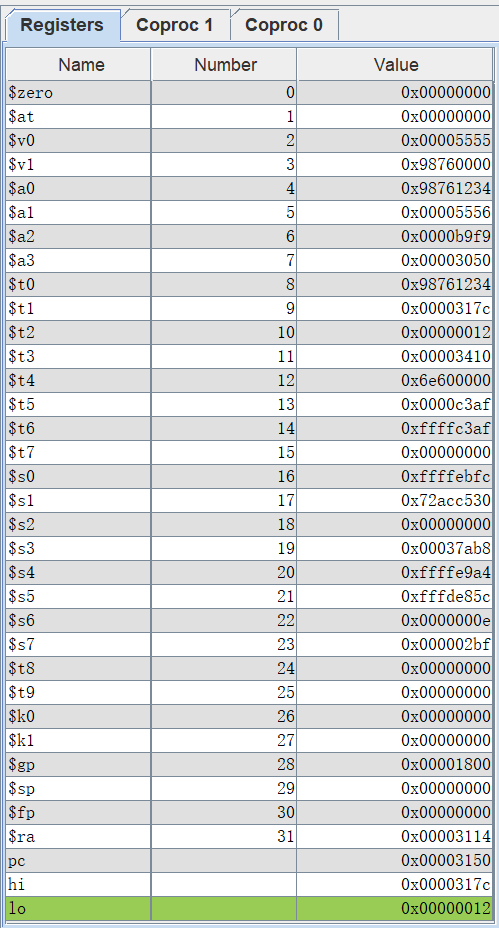
3.2测试原理

程序为project给出的42条指令测试的修改版，支持49条指令的测试，将指令分为几个subset，一一测试并对比mars和vivado跑出的结果，借以验证cpu指令的正确性。

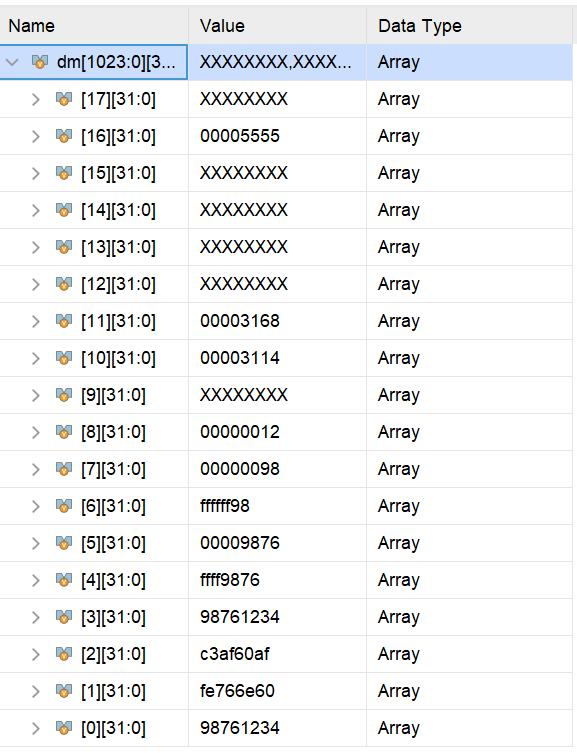
3.3测试结果（若看不清放大word比例即可）

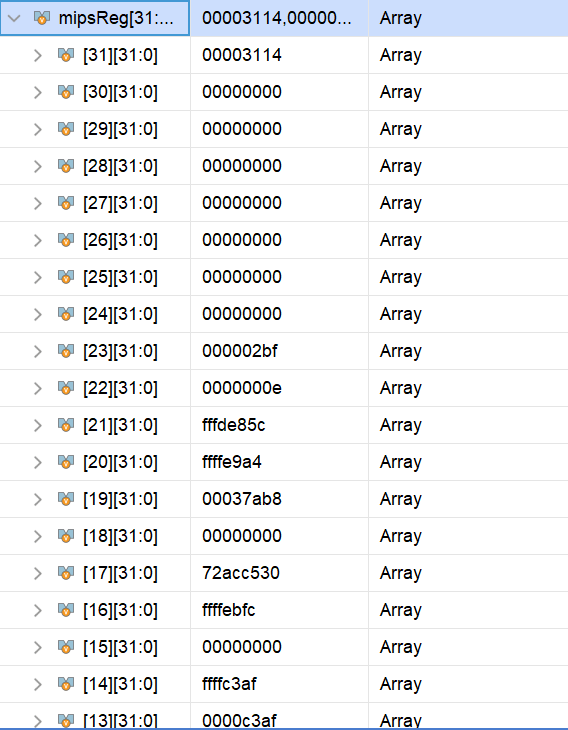
3.3.1 test.asm在mars中的运行结果：

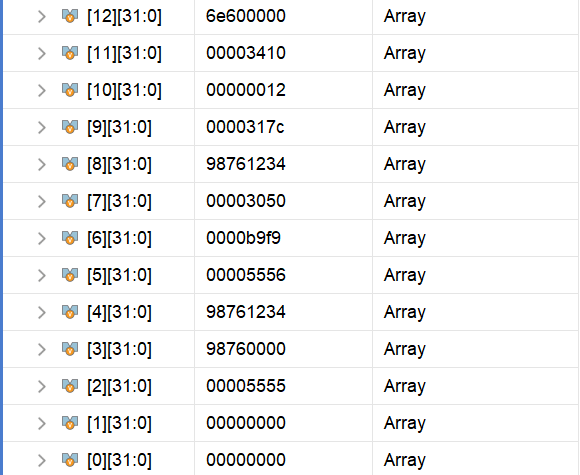




3.3.2 verilog程序模拟后的结果：









由上图所看，dm中数据完全一致，gpr除了$gp均一致。