P5 流水线处理器（Verilog）设计文档

1. 数据通路设计

表1 数据通路模块定义

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| datapath.v | module datapath(  input Clk,  input Reset,  ); |

* 1. IFU：内部包括PC，IM和NPC共三个子模块。

表2 IFU端口定义

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| Clk | I | 全局时钟信号（由顶层统一接入时钟） |
| Reset | I | 复位有效时，将PC清零 |
| Br[15:0] | I | Beq指令中的偏移量 |
| Jr[31:0] | I | Jr指令偏移量 |
| J\_JAL[25:0] | I | Jal指令偏移量 |
| NPC\_ctrl[1:0] | I | 控制NPC模块输出下一条指令在IM中的地址 |
| PC4 | O | 输出PC+4的值 |
| ins[31:0] | O | 输出当前指令 |

表3 IFU功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能定义 | 功能描述 |
| 1 | 加载指令 | 初始化时通过加载raw文件将指令读取到IM中 |
| 2 | 读出指令 | 通过PC输出的地址输出IM中相应指令 |
| 3 | 计算NPC（下一条指令地址） | 用当前地址PC和相关信号计算NPC |

注：PC起始地址为0x00003000；

IM容量为2048字且ROM寻址方式为字地址故而取PC[12:2]

表4 IFU子模块接口及定义

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| pc.v | module pc(  input Clk, //时钟信号  input Reset, //复位信号  input [31:0] Addr, //输入的下一条指令地址  output reg [31:0] Out //输出当前指令地址  ); |
| PC\_MUX.v | module PC\_MUX(  input [31:0] BR,  input [31:0] JR,  input [31:0] J\_JAL,  input [31:0] PC4,  input [1:0] NPC\_ctrl,  output [31:0] NPC  ); |
| im.v | module im(  input [12:2] Addr, //当前指令地址  input Clk,  output [31:0] Out //当前指令  ); |

* 1. NPC计算单元

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| npc.v | module npc(  input [31:0] PC4, //当前指令地址  input [15:0] BrOffset,  input [31:0] JrOffset,  input [25:0] JalOffset,  output [31:0] BR,  output [31:0] JR,  output [31:0] J\_JAL,  output [31:0] PC8  ); |

* 1. CMP（分支比较器）

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| CMP.v | module CMP(  input [31:0] D1,  input [31:0] D2,  output Eq,  output Neq  ); |

* 1. GRF（通用寄存器组）

表5 GRF模块定义

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| grf.v | module grf(  input [4:0] A1,  input [4:0] A2,  input [4:0] A3,  input [31:0] WD,  output [31:0] RD1,  output [31:0] RD2,  input Clk,  input RegWr,  input Reset  ); |

表6 GRF端口定义

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| A1[4:0] | I | 准备进行读取的第一个寄存器编号 |
| A2[4:0] | I | 准备进行读取的第二个寄存器编号 |
| A3[4:0] | I | 准备写入的寄存器编号 |
| WD[31:0] | I | 准备写入寄存器的32位数据 |
| RD1[31:0] | O | 读取的第一个寄存器中的数据 |
| RD2[31:0] | O | 读取的第二个寄存器中的数据 |
| RegWr | I | 寄存器写入使能 |
| Clk | I | 全局时钟信号 |
| Reset | I | 复位有效时，将所有寄存器清零 |

表7 GRF功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能定义 | 功能描述 |
| 1 | 读取数据 | 输出指定编号（A1,A2）寄存器的内容 |
| 2 | 写入数据 | 将来自于WD的数据写入指定编号（A3）寄存器 |
| 3 | 存储数据 | 将数据保存在寄存器当中即使断电也不会清零 |

注：寄存器总数为 32 个；

0 号寄存器的值保持为 0。

* 1. ALU（算术逻辑单元）:

表8 ALU模块接口定义

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| alu.v | module alu(  input [31:0] A,  input [31:0] B,  output [31:0] Out,  input [4:0] ALUOp,  input [4:0] sa  ); |

表9 ALU端口定义

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| A[31:0] | I | ALU进行运算的第一个操作数 |
| B[31:0] | I | ALU进行运算的第二个操作数 |
| Eq | O | 判断两个输入是否相等，相等则输出1 |
| ALUop[4:0] | I | ALU功能选择信号，决定输出哪一个运算结果 |
| Out[31:0] | O | 运算结果输出 |
| sa[4:0] | I | sll,srl偏移量 |

表10 ALUop信号定义

|  |  |
| --- | --- |
| ALUop | Func |
| 00000 | 加法 |
| 00001 | 减法 |
| 00010 | 比较大小，当且仅当A<B时输出1 |
| 00011 | 按位或运算 |
| 00100 | B<<sa |
| 00101 | B>>sa |
| 00110 | AND |
| 00111 | XOR |
| 01000 | 算术右移 |
| 01001 | 算数可变右移 |
| 01010 | 逻辑可变左移 |
| 01011 | 逻辑可变右移 |
| 01100 | NOR |
| 01101 | 无符号比较 |

表11 ALU功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能定义 | 功能描述 |
| 1 | 加 | 执行A+B |
| 2 | 减 | 执行A-B |
| 3 | OR | 按位计算A or B |
| 4 | 比较大小 | 当且仅当A <B 输出1（或无符号比较） |
| 5 | 逻辑（可变）左/右移 | 根据sa或rs将B逻辑左/右移相应位数 |
| 6 | AND | 按位与 |
| 7 | XOR | 按位异或 |
| 8 | SRA/V | 算数（可变）右移 |
| 9 | NOR | NOR |

* 1. MDU（乘除法支持模块）

表12 MDU模块接口定义

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| MDU.v | module MDU(  input [31:0] A,  input [31:0] B,  input Reset,  input Clk,  input [3:0] MDU\_ctrl,  input start,  output busy,  output [31:0] HI,  output [31:0] LO  ); |

表13 MDU端口定义

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| A[31:0] | I | MDU进行运算的第一个操作数 |
| B[31:0] | I | MDU进行运算的第二个操作数 |
| Reset | I | 重置清零HI，LO |
| Clk | I | 时钟信号 |
| MDU\_ctrl | I | 执行指令选择 |
| start | I | 乘除开始 |
| busy | O | 乘除占用信号 |
| HI | O | 存储乘法高位和除法余数 |
| LO | O | 存储乘法低位和除法的商 |

表13 MDU功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能定义 | 功能描述 |
| 1 | mult | 有符号乘法 |
| 2 | multu | 无符号乘法 |
| 3 | div | 有符号除法 |
| 4 | divu | 无符号除法 |
| 5 | mfhi | 读取HI寄存器 |
| 6 | mflo | 读取LO寄存器 |
| 7 | mthi | 写入HI寄存器 |
| 8 | mtlo | 写入LO寄存器 |

* 1. DM（数据存储器）

表14 DM模块接口定义

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| dm.v | module dm(  input Clk,  input MemWr,  input [31:0] WD,  input [31:0] Addr,  input Reset,  input [3:0] BE,  output [31:0] RD  ); |

表15 DM端口定义

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| WD[31:0] | I | 将写入内存的数据 |
| Addr[31:0] | I | 准备进行读写的内存地址 |
| MemWr | I | 内存写使能 |
| RD[31:0] | O | 输出指定地址中存放的数据 |
| Clk | I | 全局时钟信号 |
| Reset | I | 复位有效时，将Mem清零 |
| BE | I | 字节位写使能 |

表16 DM功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能定义 | 功能描述 |
| 1 | 读取 | 按照指定地址输出存储的数据 |
| 2 | 写入 | 将输入数据写入指定地址内存中 |
| 3 | 按字节读写 | 可以按照指定字节读写内容 |

注：内存起始地址：0x00000000。

RAM大小共计2048字

* 1. BL（W级数据扩展模块）

表17 BL模块接口定义

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| BL.v | module BL(  input [31:0] ins,  input [31:0] Din,  output [31:0] Dout  ); |

注：根据指令，按照相对应的字节处理MemRdData从而实现lblh等

* 1. BE（字节写使能控制）

表18 BE模块接口定义

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| BE.v | module BE(  input [1:0] Addr,  input [31:0] ins,  output [3:0] BE  ); |

注：根据指令产生相应的字节写使能信号。

* 1. EXT：

表19 EXT模块接口定义

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| ext.v | module ext(  input ExtOp,  input [15:0] In,  output [31:0] Out  ); |

表20 端口定义

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| In[15:0] | I | 输入16位带扩展数字 |
| ExtOp | I | 扩展功能选择 |
| Out[31:0] | O | 输出扩展结果 |

表21 ExtOp信号定义

|  |  |
| --- | --- |
| ExtOp | Func |
| 0 | 零扩展 |
| 1 | 符号扩展 |

表22 EXT功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能定义 | 功能描述 |
| 1 | 零扩展 | 将输入数据扩展为32位并用0填充高16位 |
| 2 | 符号扩展 | 将输入数据扩展为32位并用原数据符号位填充高16位 |

* 1. SH16（16位移位器）

表23 SH16模块接口定义

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| sl16.v | module sl16(  input [15:0] in,  output [31:0] out  ); |

表24 SH16端口定义

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| In[15:0] | I | 输入十六位待移位数据 |
| SH[31:0] | O | 移位结果 |

表25 SH16功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能定义 | 功能描述 |
| 1 | 16位左移 | 将输入数据零扩展为32位后左移指定位数 |

* 1. MUX（多路选择器）

表26 MUX模块接口定义

|  |  |  |
| --- | --- | --- |
| 文件 | 模块 | 模块接口定义 |
| mux.v | mux2\_5 | module mux2\_5( //5位2选1  input [4:0] d1, //输入数据  input [4:0] d2,  input src, //输出选择  output [4:0] out //输出  ); |
| mux4\_32 | module mux4\_32( //32位4选1  input [1:0] src,  input [31:0] d1,  input [31:0] d2,  input [31:0] d3,  input [31:0] d4,  output reg [31:0] out  ); |
| mux2\_32 | module mux2\_32( //32位2选1  input src,  input [31:0] d1,  input [31:0] d2,  output [31:0] out  ); |

1. 控制器设计

（暂停和转发的详细逻辑请看思考题部分）

|  |  |
| --- | --- |
| 文件 | 模块端口定义 |
| stall.v | module stall(  input [31:0] IR\_D,  input [31:0] IR\_E,  input [31:0] IR\_M,  input Cal\_r\_D,  input Cal\_i\_D,  input beq\_D,  input jr\_D,  input jal\_D,  input ld\_D,  input st\_D,  input Cal\_r\_E,  input Cal\_i\_E,  input beq\_E,  input jr\_E,  input jal\_E,  input ld\_E,  input st\_E,  input sl\_E,  input sl\_D,  input ld\_M,  output stall  ); |

|  |  |
| --- | --- |
| 文件 | 模块端口定义 |
| forward.v | module forward(  input [4:0] WrReg\_M,  input RegWr\_M,  input generated\_M,  input [4:0] WrReg\_W,  input RegWr\_W,  input generated\_W,  input [4:0] ReReg1\_E,  input [4:0] ReReg2\_E,  input [4:0] ReReg1\_D,  input [4:0] ReReg2\_D,  input [4:0] ReReg2\_M,  output [1:0] ForwardRSD,  output [1:0] ForwardRTD,  output [1:0] ForwardRSE,  output [1:0] ForwardRTE,  output [1:0] ForwardRTM,  output [1:0] ForwardRTE\_ALUb  ); |

表23 Controler模块接口定义

|  |  |
| --- | --- |
| 文件 | 模块端口定义 |
| control.v | module controla(  input [31:0] ins,  input Eq,  input Neq,  input lz,  input Ltz,  input Lez,  input Gtz,  input Gez,  input [1:0] stage,  input [31:0] ALUout,  input [31:0] Shift,  input [31:0] PC8,  output [3:0] MDU\_ctrl,  output start,  output mdu\_out\_ctrl,  output Uout\_ctrl,  output [1:0] NPC\_ctrl,  output RegDst,  output RegDst\_2,  output [1:0] grf\_WDsrc,  output RegWr,  output EXT\_ZS,  output ALUsrc,  output [4:0] ALUOp,  output MemWr,  output Cal\_r,  output Cal\_i,  output ld,  output st,  output beq,  output j,  output jal,  output jr,  output sl,  output lui,  output md,  output mf,  output mt,  output [31:0] WrData\_M,  output generated,  output [4:0] WrReg,  output [4:0] ReReg1,  output [4:0] ReReg2  ); |

* 1. 为具有多个输入的模块添加选择信号确定输入来源，并根据指令做出真值表确定其他命令信号

表24 控制信号

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FUNC | 100001 | 001000 | 100011 | N/A | | | | | |
| OP | 000000 | 000000 | 000000 | 001101 | 100011 | 101011 | 000100 | 001111 | 000011 |
| 控制  信号 | addu  add | jr | subu  sub | ori | lw | sw | beq  bne | lui | jal |
| RegDst | 0 | 0 | 0 | 1 | 1 | X | X | 1 | 0 |
| RegDst\_2 | 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | 1 |
| grf\_WDsrc | 00 | X | 00 | 00 | 01 | X | X | 10 | 11 |
| RegWr | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| EXT\_ZS | X | X | X | 0 | 1 | 1 | 0 | X | X |
| ALUsrc | 0 | 0 | 0 | 1 | 1 | 1 | 0 | X | X |
| ALUOp | 000 | X | 001 | 011 | 000 | 000 | 010 | X | X |
| MemWr | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| NPC\_ctrl | 00 | 10 | 00 | 00 | 00 | 00 | 01 | 00 | 11 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| FUNC | N/A | | 000000 | 000010 | 100100 | 100101 | 100110 |
| OP | 000000 | 000000 | 000000 | 001101 | 100011 | 101011 | 000100 | |
| 控制  信号 | addi | addiu | sll | srl | AND | OR | XOR | |
| RegDst | 1 | 1 | 0 | 0 | 0 | 0 | 0 | |
| RegDst\_2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| grf\_WDsrc | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| RegWr | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| EXT\_ZS | 1 | 1 | 0 | 0 | 0 | 0 | 0 | |
| ALUsrc | 1 | 1 | 0 | 0 | 0 | 0 | 0 | |
| ALUOp | 000 | 000 | 100 | 101 | 110 | 110 | 110 | |
| MemWr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| NPC\_ctrl | 00 | 10 | 00 | 00 | 00 | 00 | 00 | |

* 1. 具体参数

表25 控制器端口定义

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| ins[31:0] | I | 当前指令 |
| Eq | I | 判断beq条件 |
| Neq | I | 判断bne条件 |
| stage | I | 当前流水级 |
| RegDst | O | 控制数据存储在GRF中的位置，0取rd字段作为地址，1取rt字段 |
| RegDst\_2 | O | 控制数据存储在GRF中的位置，0取RegDst对应输出，1取$31输出 |
| WDsrc[1:0] | O | GRF写入数据来源 |
| ALUsrc | O | ALU操作数来源，0为RD2，1为EXT |
| EXT\_ZS | O | 选择扩展器进行零扩展（0）或者符号扩展（1） |
| RegWr | O | 寄存器堆写使能 |
| MemWr | O | 内存写使能 |
| ALUop[2:0] | O | ALU功能选择信号 |
| NPC\_ctrl[1:0] | O | 控制下一条指令地址 |
| start | O | 乘除法开始 |
| mdu\_out\_ctrl | O | 控制HILO\_mux |
| Uout\_ctrl | O | 控制AO\_M\_mux |
| Cal\_r | O | R型计算类指令 |
| Cal\_i | O | I型计算类指令 |
| ld | O | load类 |
| st | O | store类 |
| beq | O | branch类 |
| jr | O | jr类 |
| jal | O | jal类 |
| sl | O | sl类 |
| mf | O | mf类 |
| md | O | md类 |
| mt | O | mt类 |
| generated | O | 确认当前流水级已经生成了写入数据 |
| WrReg[4:0] | O | 当前指令准备写的寄存器 |
| ReReg1[4:0] | O | 当前指令需要的寄存器编号 |
| ReReg2[4:0] | O | 当前指令需要的寄存器编号 |

表26 GRF写入数据来源控制信号

|  |  |
| --- | --- |
| WDsrc | Src |
| 00 | ALU计算结果 |
| 01 | Mem（内存） |
| 10 | SH（移位器） |
| 11 | PC+4 |

表27 ALU控制信号

|  |  |
| --- | --- |
| ALUop | Func |
| 00000 | 加法 |
| 00001 | 减法 |
| 00010 | 比较大小，当且仅当A<B时输出1 |
| 00011 | 按位或运算 |
| 00100 | B<<sa |
| 00101 | B>>sa |
| 00110 | AND |
| 00111 | XOR |
| 01000 | 算术右移 |
| 01001 | 算数可变右移 |
| 01010 | 逻辑可变左移 |
| 01011 | 逻辑可变右移 |
| 01100 | NOR |
| 01101 | 无符号比较 |

表28 NPC控制信号

|  |  |
| --- | --- |
| NPC\_ctrl | Src |
| 00 | PC+4 |
| 01 | 输出beq类指令对应地址 |
| 10 | Jr地址 |
| 11 | Jal\_J地址 |

1. 测试程序

Case1:

lui $s0, 1

ori $s0 $s0 65535

ori $s1 $0 5

ori $s2 $0 65535

ori $s3 $0 8

lui $s5 0xABCD

ori $s6 $t0 0xffff

#initial done

subu $t0 $s0 $s1

addu $t1 $s0 $t0

addu $t2 $s0 $t0

addu $t3 $s0 $t0

sw $t3 0($0)

sw $t2 4($0)

sw $t1 8($0)

subu $t0 $s0 $s2

addu $t1 $t0 $s0

addu $t2 $t0 $s0

subu $t3 $t0 $s0

sw $t1 12($0)

sw $t3 16($0)

sw $t2 20($0)

subu $t0 $s0 $s3

ori $t1 $t0 36

ori $t2 $t0 21

ori $t3 $t0 4

sw $t3 24($0)

sw $t2 28($0)

sw $t1 32($0)

ori $t0 $s1 100

addu $t0 $0 $s3

lw $t1 0($t0)

lw $t2 4($t0)

lw $t3 8($t0)

sw $t3 36($0)

sw $t2 40($0)

sw $t1 44($0)

ori $t0 $s1 100

ori $t0 $0 0

lw $t1 0($t0)

lw $t2 4($t0)

lw $t3 8($t0)

sw $t3 48($0)

sw $t2 52($0)

sw $t1 56($0)

ori $t0 $s1 178

addu $t1 $s0 $t0

subu $t2 $s0 $t0

addu $t3 $s0 $t0

sw $t3 60($0)

sw $t2 64($0)

sw $t1 68($0)

ori $t0 $s1 971

subu $t1 $t0 $s0

addu $t2 $t0 $s0

subu $t3 $t0 $s0

sw $t1 72($0)

sw $t3 76($0)

sw $t2 80($0)

ori $t0 $s1 6

ori $t1 $t0 9

ori $t2 $t0 6

ori $t3 $t0 2

sw $t2 84($0)

sw $t1 88($0)

sw $t3 92($0)

lw $t0 44($0)

subu $t1 $s0 $t0

subu $t2 $s0 $t0

addu $t3 $s0 $t0

sw $t3 96($0)

sw $t2 100($0)

sw $t1 104($0)

lw $t0 16($0)

addu $t1 $t0 $s0

subu $t2 $t0 $s0

subu $t3 $t0 $s0

sw $t1 108($0)

sw $t3 112($0)

sw $t2 116($0)

lw $t0 84($0)

ori $t1 $t0 100

ori $t2 $t0 100

ori $t3 $t0 100

sw $t3 120($0)

sw $t2 124($0)

sw $t1 128($0)

ori $t0 $0 4

sw $t0 132($0)

lw $t0 132($0)

lw $t1 0($t0)

lw $t2 4($t0)

lw $t3 8($t0)

sw $t3 132($0)

sw $t2 136($0)

sw $t1 140($0)

subu $t0 $s0 $s1

addu $t1 $s0 $t0

addu $t2 $s0 $t0

addu $t3 $s0 $t0

sw $t3 144($0)

sw $t2 148($0)

sw $t1 152($0)

subu $t0 $s0 $s1

addu $t1 $t0 $s0

subu $t2 $t0 $s0

subu $t3 $t0 $s0

sw $t1 156($0)

sw $t3 160($0)

sw $t2 164($0)

addu $t0 $s0 $s1

ori $t1 $t0 21

ori $t2 $t0 81

ori $t3 $t0 0

sw $t3 168($0)

sw $t2 172($0)

sw $t1 176($0)

ori $t0 $s1 100

addu $t0 $0 $0

lw $t1 0($t0)

lw $t2 4($t0)

lw $t3 8($t0)

sw $t3 180($0)

sw $t2 184($0)

sw $t1 188($0)

ori $t0 $s1 100

ori $t0 $0 0

lw $t1 0($t0)

lw $t2 4($t0)

lw $t3 8($t0)

sw $t3 192($0)

sw $t2 196($0)

sw $t1 200($0)

ori $t0 $s1 357

subu $t1 $s0 $t0

addu $t2 $s0 $t0

addu $t3 $s0 $t0

sw $t3 204($0)

sw $t2 208($0)

sw $t1 212($0)

ori $t0 $s1 316

subu $t1 $t0 $s0

subu $t2 $t0 $s0

addu $t3 $t0 $s0

sw $t1 216($0)

sw $t3 220($0)

sw $t2 224($0)

ori $t0 $s1 4

ori $t1 $t0 9

ori $t2 $t0 7

ori $t3 $t0 2

sw $t2 228($0)

sw $t1 232($0)

sw $t3 236($0)

lw $t0 52($0)

addu $t1 $s0 $t0

addu $t2 $s0 $t0

subu $t3 $s0 $t0

sw $t3 240($0)

sw $t2 244($0)

sw $t1 248($0)

lw $t0 240($0)

addu $t1 $t0 $s0

subu $t2 $t0 $s0

addu $t3 $t0 $s0

sw $t1 252($0)

sw $t3 256($0)

sw $t2 260($0)

lw $t0 180($0)

ori $t1 $t0 100

ori $t2 $t0 100

ori $t3 $t0 100

sw $t3 264($0)

sw $t2 268($0)

sw $t1 272($0)

ori $t0 $0 4

sw $t0 276($0)

lw $t0 276($0)

lw $t1 0($t0)

lw $t2 4($t0)

lw $t3 8($t0)

sw $t3 276($0)

sw $t2 280($0)

sw $t1 284($0)

addu $t0 $s0 $s1

addu $t1 $s0 $t0

subu $t2 $s0 $t0

subu $t3 $s0 $t0

sw $t3 288($0)

sw $t2 292($0)

sw $t1 296($0)

subu $t0 $s0 $s1

subu $t1 $t0 $s0

addu $t2 $t0 $s0

addu $t3 $t0 $s0

sw $t1 300($0)

sw $t3 304($0)

sw $t2 308($0)

addu $t0 $s0 $s1

ori $t1 $t0 10

ori $t2 $t0 83

ori $t3 $t0 39

sw $t3 312($0)

sw $t2 316($0)

sw $t1 320($0)

ori $t0 $s1 100

addu $t0 $0 $0

lw $t1 0($t0)

lw $t2 4($t0)

lw $t3 8($t0)

sw $t3 324($0)

sw $t2 328($0)

sw $t1 332($0)

ori $t0 $s1 100

ori $t0 $0 0

lw $t1 0($t0)

lw $t2 4($t0)

lw $t3 8($t0)

sw $t3 336($0)

sw $t2 340($0)

sw $t1 344($0)

ori $t0 $s1 831

addu $t1 $s0 $t0

addu $t2 $s0 $t0

subu $t3 $s0 $t0

sw $t3 348($0)

sw $t2 352($0)

sw $t1 356($0)

ori $t0 $s1 947

subu $t1 $t0 $s0

addu $t2 $t0 $s0

subu $t3 $t0 $s0

sw $t1 360($0)

sw $t3 364($0)

sw $t2 368($0)

ori $t0 $s1 7

ori $t1 $t0 9

ori $t2 $t0 0

ori $t3 $t0 0

sw $t2 372($0)

sw $t1 376($0)

sw $t3 380($0)

lw $t0 136($0)

subu $t1 $s0 $t0

addu $t2 $s0 $t0

addu $t3 $s0 $t0

sw $t3 384($0)

sw $t2 388($0)

sw $t1 392($0)

lw $t0 344($0)

addu $t1 $t0 $s0

addu $t2 $t0 $s0

addu $t3 $t0 $s0

sw $t1 396($0)

sw $t3 400($0)

sw $t2 404($0)

lw $t0 92($0)

ori $t1 $t0 100

ori $t2 $t0 100

ori $t3 $t0 100

sw $t3 408($0)

sw $t2 412($0)

sw $t1 416($0)

ori $t0 $0 4

sw $t0 420($0)

lw $t0 420($0)

lw $t1 0($t0)

lw $t2 4($t0)

lw $t3 8($t0)

sw $t3 420($0)

sw $t2 424($0)

sw $t1 428($0)

subu $t0 $s0 $s1

addu $t1 $s0 $t0

subu $t2 $s0 $t0

addu $t3 $s0 $t0

sw $t3 432($0)

sw $t2 436($0)

sw $t1 440($0)

addu $t0 $s0 $s1

addu $t1 $t0 $s0

subu $t2 $t0 $s0

addu $t3 $t0 $s0

sw $t1 444($0)

sw $t3 448($0)

sw $t2 452($0)

subu $t0 $s0 $s1

ori $t1 $t0 3

ori $t2 $t0 79

ori $t3 $t0 51

sw $t3 456($0)

sw $t2 460($0)

sw $t1 464($0)

ori $t0 $s1 100

addu $t0 $0 $0

lw $t1 0($t0)

lw $t2 4($t0)

lw $t3 8($t0)

sw $t3 468($0)

sw $t2 472($0)

sw $t1 476($0)

ori $t0 $s1 100

ori $t0 $0 0

lw $t1 0($t0)

lw $t2 4($t0)

lw $t3 8($t0)

sw $t3 480($0)

sw $t2 484($0)

sw $t1 488($0)

ori $t0 $s1 148

addu $t1 $s0 $t0

subu $t2 $s0 $t0

subu $t3 $s0 $t0

sw $t3 492($0)

sw $t2 496($0)

sw $t1 500($0)

ori $t0 $s1 205

subu $t1 $t0 $s0

subu $t2 $t0 $s0

subu $t3 $t0 $s0

sw $t1 504($0)

sw $t3 508($0)

sw $t2 512($0)

ori $t0 $s1 3

ori $t1 $t0 4

ori $t2 $t0 3

ori $t3 $t0 3

sw $t2 516($0)

sw $t1 520($0)

sw $t3 524($0)

lw $t0 48($0)

addu $t1 $s0 $t0

subu $t2 $s0 $t0

addu $t3 $s0 $t0

sw $t3 528($0)

sw $t2 532($0)

sw $t1 536($0)

lw $t0 312($0)

subu $t1 $t0 $s0

addu $t2 $t0 $s0

addu $t3 $t0 $s0

sw $t1 540($0)

sw $t3 544($0)

sw $t2 548($0)

lw $t0 532($0)

ori $t1 $t0 100

ori $t2 $t0 100

ori $t3 $t0 100

sw $t3 552($0)

sw $t2 556($0)

sw $t1 560($0)

ori $t0 $0 4

sw $t0 564($0)

lw $t0 564($0)

lw $t1 0($t0)

lw $t2 4($t0)

lw $t3 8($t0)

sw $t3 564($0)

sw $t2 568($0)

sw $t1 572($0)

addu $t0 $s0 $s1

subu $t1 $s0 $t0

subu $t2 $s0 $t0

subu $t3 $s0 $t0

sw $t3 576($0)

sw $t2 580($0)

sw $t1 584($0)

subu $t0 $s0 $s1

subu $t1 $t0 $s0

addu $t2 $t0 $s0

subu $t3 $t0 $s0

sw $t1 588($0)

sw $t3 592($0)

sw $t2 596($0)

subu $t0 $s0 $s1

ori $t1 $t0 82

ori $t2 $t0 43

ori $t3 $t0 98

sw $t3 600($0)

sw $t2 604($0)

sw $t1 608($0)

ori $t0 $s1 100

addu $t0 $0 $0

lw $t1 0($t0)

lw $t2 4($t0)

lw $t3 8($t0)

sw $t3 612($0)

sw $t2 616($0)

sw $t1 620($0)

ori $t0 $s1 100

ori $t0 $0 0

lw $t1 0($t0)

lw $t2 4($t0)

lw $t3 8($t0)

sw $t3 624($0)

sw $t2 628($0)

sw $t1 632($0)

ori $t0 $s1 739

addu $t1 $s0 $t0

subu $t2 $s0 $t0

addu $t3 $s0 $t0

sw $t3 636($0)

sw $t2 640($0)

sw $t1 644($0)

ori $t0 $s1 904

subu $t1 $t0 $s0

addu $t2 $t0 $s0

addu $t3 $t0 $s0

sw $t1 648($0)

sw $t3 652($0)

sw $t2 656($0)

ori $t0 $s1 1

ori $t1 $t0 0

ori $t2 $t0 9

ori $t3 $t0 9

sw $t2 660($0)

sw $t1 664($0)

sw $t3 668($0)

lw $t0 624($0)

addu $t1 $s0 $t0

subu $t2 $s0 $t0

subu $t3 $s0 $t0

sw $t3 672($0)

sw $t2 676($0)

sw $t1 680($0)

lw $t0 412($0)

subu $t1 $t0 $s0

subu $t2 $t0 $s0

addu $t3 $t0 $s0

sw $t1 684($0)

sw $t3 688($0)

sw $t2 692($0)

lw $t0 404($0)

ori $t1 $t0 100

ori $t2 $t0 100

ori $t3 $t0 100

sw $t3 696($0)

sw $t2 700($0)

sw $t1 704($0)

ori $t0 $0 4

sw $t0 708($0)

lw $t0 708($0)

lw $t1 0($t0)

lw $t2 4($t0)

lw $t3 8($t0)

sw $t3 708($0)

sw $t2 712($0)

sw $t1 716($0)

lui $t0 224

subu $t1 $s0 $t0

subu $t2 $s0 $t0

addu $t3 $s0 $t0

sw $t3 720($0)

sw $t2 724($0)

sw $t1 728($0)

lui $t0 469

subu $t1 $t0 $s0

addu $t2 $t0 $s0

addu $t3 $t0 $s0

sw $t1 732($0)

sw $t3 736($0)

sw $t2 740($0)

lui $t0 121

ori $t1 $t0 55

ori $t2 $t0 42

ori $t3 $t0 24

sw $t1 744($0)

sw $t3 748($0)

sw $t2 752($0)

lui $t0 686

sw $t0 756($0)

sw $t0 760($0)

sw $t0 764($0)

测试程序期望结果是：

