计算机组成原理 P6 CPU 设计实验报告[1]

一、模块规格

1. F 级

F 级端口定义表

信号名	方向	描述
Clk	I	时钟信号
Reset	I	复位信号
Stop	I	暂停信号（D 级产生）
[31:0]IR_E	I	E 级 IR
[31:0]IR_M	I	M 级 IR
[15:0]imm16	I	16 位地址（D 级产生）
[25:0]imm26	I	26 位地址（D 级产生）
[31:0]rs	I	rs 寄存器回写 pc（D 级产生）
[2:0]pcsel	I	Npc 选择信号（D 级产生）
Equal	I	Beq 判断相等信号（D 级产生）
Isbeq	I	Beq 信号判断（D 级产生）
[31:0]Pc_F	O	F 级当前的 pc
[31:0]Pc4_F	O	F 级的 pc+4
[31:0]IR_F	O	F 级当前的 IR

1.1 IM

（1） 端口说明

IM 端口定义表

信号名	方向	描述
[31:0]Address	I	指令地址
[31:0]IR	O	读出指令

（2）功能定义

[1] Copyright: 张泽 18182659

IM 功能描述表

序号	功能定义	功能描述
1	加载地址	将指令的地址加载至 ROM 中
2	寻址	根据 address 找到对应指令

1.2 Npc

（2） 端口说明

npc 端口定义表

信号名	方向	描述
[31:0]Pc4	I	Pc+4 的值
[31:0]Rs	I	rs 寄存器回写 pc
[2:0]Pcsel	I	Npc 选择信号
[15:0]Imm16	I	16 位地址
[25:0]Imm26	I	26 位地址
Isbeq	I	判断 beq 指令
Stop	I	暂停信号
Clk	I	时钟信号
Reset	I	复位信号
Equal	I	Beq 判断相等指令
[31:0]Nextpc	0	下一周期的 pc 值

（2）功能定义

npc 功能描述表

序号	功能定义	功能描述

1	计算下一周期 pc	Pcsrc==0
		Nextpc <= pc4;
		Pcsrc==1
		if(isbeq && equal)
		Nextpc <= pc4 + ext16;
		else
		Nextpc <= pc4;

		pcsrc === 2 Nextpc<= ext26; pcsrc === 3 Nextpc <= rs;
2	复位	复位信号有效时，Nextpc <= 32'h3000;
3	暂停	锁死 pc，pc = pc4 - 4

2．FD 级寄存器

FD 级寄存器端口定义表

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
stop	I	暂停信号（D 级产生）(锁死 FD)
[31:0]IR	I	F 级 IR
[31:0]PC	I	F 级 PC
[31:0]PC4	I	F 级 PC+4
[31:0]IR_D	O	D 级 IR
[31:0]PC_D	O	D 级 PC
[31:0]PC4_D	O	D 级 PC4

3．D 级

D 级端口定义表

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
[31:0]IR	I	D 级当前的 IR
[31:0]IR_E	I	E 级 IR
[31:0]IR_M	I	M 级 IR
[31:0]IR_W	I	W 级 IR
[31:0]pc	I	当前的 pc

[31:0]pc4	I	当前的 pc+4
[31:0]pc4_E	I	E 级 pc+4
[31:0]high_E	I	E 级加载立即数到高位
[31:0]pc4_M	I	M 级 pc+4
[31:0]ALUout_M	I	M 级的 ALUout
[31:0]mul_WD	I	W 级的回写数据
[4:0]mul_A3	I	W 级的回写地址
RegWrite_W	I	寄存器写入控制信号（W 级产生）
[31:0]pc_W	I	W 级 pc
[31:0]IR_D	O	D 级 IR
[31:0]Data1	O	Rs 读出的数据
[31:0]Data2	O	Rt 读出的数据
[31:0]Sign_ext	O	符号扩展
[31:0]Zero_ext	O	零扩展
[31:0]high	O	加载立即数到高位
[31:0]PC_D	O	D 级 pc
[31:0]PC4_D	O	D 级 pc+4
[15:0]imm16	O	16 位立即数
[25:0]imm26	O	26 位立即数
[2:0]PCsel	O	Npc 选择信号
Equal_D	O	Beq 相等信号
Isbeq_D	O	Beq 信号
Stop	O	暂停信号

3.1 GRF

（1） 端口说明

GRF 端口定义表

信号名	方向	描述
RegWrite	I	数据写入信号
Clk	I	时钟信号
Reset	I	复位信号

WD[31:0]	I	目标寄存器写入数据
pc[31:0]	I	指令地址
A1[4:0]	I	第一个寄存器源操作数
A2[4:0]	I	第二个寄存器源操作数
A3[4:0]	I	寄存器目标操作数
data1[31:0]	O	第一个源寄存器读出数据
data2[31:0]	O	第二个源寄存器读出数据

（2） 功能定义

GRF 功能描述表

序号	功能定义	功能描述
1	从寄存器堆读数据	读出 rs(A1)、rt(A2)地址对应寄存器中的数据到 data1 和 data2
2	向寄存器堆写数据	RegWrite 信号有效时，将 WD 写入 A3 地址对应的寄存器
3	复位	Reset 信号有效时，所有寄存器数据清零。
4	打印输出	`$display("%d@%h: $%d <= %h", $time ,pc, A3,WD)`

3.2 pause

（1） 端口说明

pause 端口定义表

信号名	方向	描述
[31:0]IR	I	D 级 IR
[31:0]IR_E	I	E 级 IR
[31:0]IR_M	I	M 级 IR
stop	O	暂停信号

（2） 功能定义

GRF 功能描述表

序号	功能定义	功能描述
1	判断暂停	根据 IR、IR_E、IR_M 判断是否需要暂停

4．DE 级寄存器

DE 级寄存器端口定义表

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
[31:0]RD1	I	D 级 RD1
[31:0]RD2	I	D 级 RD2
[31:0]sign_ext	I	D 级符号扩展
[31:0]ext_0	I	D 级零扩展
[31:0]high	I	D 级加载立即数到高位
[31:0]IR	I	D 级 IR
[31:0]PC	I	D 级 PC
[31:0]PC4	I	D 级 PC+4
[31:0]RD1_E	O	E 级 RD1
[31:0]RD2_E	O	E 级 RD2
[31:0]Sign_ext_E	O	E 级符号扩展
[31:0]Ext_0_E	O	E 级零扩展
[31:0]High_E	O	E 级加载立即数到高位
[31:0]IR_E	O	E 级 IR
[31:0]PC_E	O	E 级 PC
[31:0]PC4_E	O	E 级 PC4

5．E 级

E 级端口定义表

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
[31:0]IR	I	E 级当前的 IR
[31:0]IR_M	I	M 级 IR
[31:0]IR_W	I	W 级 IR
[31:0]pc4_M	I	M 级 pc+4

[31:0]ALUout_M	I	M 级的 ALUout
[31:0]mul_WD	I	W 级的回写数据
[31:0]zeroext	I	零扩展
[31:0]signext	I	符号扩展
[31:0]high	I	加载立即数到高位
[31:0]pc	I	Pc
[31:0]pc4	I	Pc+4
[31:0]Data1	I	Data1
[31:0]Data2	I	Data2
[31:0]ALUout_E	O	E 级 alu 输出
[31:0]data2_E	O	E 级转发后的 data2
[31:0]IR_E	O	E 级 IR
[31:0]PC_E	O	E 级 pc
[31:0]PC4_E	O	E 级 pc+4

5.1 ALU

（1） 端口说明

ALU 端口定义表

信号名	方向	描述
Alu1[31:0]	I	参与 ALU 计算的第一个值
Alu2[31:0]	I	参与 ALU 计算的第二个值
ALUop[3:0]	I	ALU 功能选择信号 0000：A&B 0001：A|B 0010：A+B 0110：A-B
aluout[31:0]	O	ALU 计算的结果

（2） 功能定义

ALU 功能描述表

序号	功能定义	功能描述

1	与运算	reselt = A & B
2	或运算	reselt = A | B
3	加运算	reselt = A + B
4	减运算	reselt = A - B

5.2 alupro

（1） 端口说明

alupro 端口定义表

信号名	方向	描述
Alu1[31:0]	I	参与计算的第一个值
Alu2[31:0]	I	参与计算的第二个值
op[3:0]	I	ALU 功能选择信号 0000：非乘除相关指令 0001：mult 0010：multu 0011：div 0100：divu 0101：mtlo 0110：mthi 0111：mflo 1000：mfhi
clk	I	始终信号
Reset	I	复位信号
Start	I	乘除法计算开始信号
Out	O	LO、HI 结果输出
Alubusy	O	Alupro 模块占用信号

6. EM 级寄存器

EM 级寄存器端口定义表

信号名	方向	描述

clk	I	时钟信号
reset	I	复位信号
[31:0]ALUout	I	E 级 RD1
[31:0]data2	I	E 级 RD2
[31:0]IR	I	E 级 IR
[31:0]PC	I	E 级 PC
[31:0]PC4	I	E 级 PC+4
[31:0]ALUout_M	O	M 级 ALUout
[31:0]Data2_M	O	M 级 data2
[31:0]IR_M	O	M 级 IR
[31:0]PC_M	O	M 级 PC
[31:0]PC4_M	O	M 级 PC4

7. M 级

M 级端口定义表

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
[31:0]IR	I	M 级当前的 IR
[31:0]IR_W	I	W 级 IR
[31:0]pc	I	Pc
[31:0]pc4	I	Pc+4
[31:0]ALUout	I	Alu 的输出
[31:0]data2	I	Data2
[31:0]mul_WD	I	W 级的回写数据
[31:0]ALUout_M	O	M 级 aluout
[31:0]DMout_M	O	M 级的 DM 输出
[31:0]IR_M	O	M 级 IR
[31:0]PC_M	O	M 级 pc
[31:0]PC4_M	O	M 级 pc+4

7.1 DM

（1） 端口说明

DM 端口定义表

信号名	方向	描述
clk	I	时钟信号
Save_sel	I	写数据选择信号
reset	I	复位信号
memWrite	I	信号有效时，向内存中写入数据
memRead	I	信号有效时，从内存中读出数据
pc[31:0]	I	指令地址
Address[31:0]	I	32 位内存地址
Writedata[31:0]	I	内存写入数据
readdata[31:0]	O	从内存中读出的数据

（2） 功能定义

DM 功能描述表

序号	功能定义	功能描述
1	读数据	根据地址信号从内存中读取数据
2	写数据	根据地址信号向内存中吸入数据
3	复位	Reset 信号有效时，内存清零
4	打印输出	$display("%d@%h: *%h <= %h",$time,pc, address, writedata);

7.2 load

（1） 端口说明

load 端口定义表

信号名	方向	描述
Data	I	写寄存器数据
Address	I	写寄存器地址
Load_sel	I	加载数据选择信号
Dataout	O	写寄存器数据输出

8．MW 级寄存器

MW 级寄存器端口定义表

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
[31:0]IR	I	M 级 IR
[31:0]PC	I	M 级 PC
[31:0]PC4	I	M 级 PC+4
[31:0]ALUout	I	M 级 ALUout
[31:0]DMout	I	M 级 DMout
[31:0]IR_W	O	W 级 IR
[31:0]PC_W	O	W 级 PC
[31:0]PC4_W	O	W 级 PC4
[31:0]ALUout_W	O	W 级 ALUout
[31:0]DMout_W	O	W 级 DMout

9．W 级

W 级端口定义表

信号名	方向	描述
[31:0]IR	I	W 级当前的 IR
[31:0]pc	I	Pc
[31:0]pc4	I	Pc+4
[31:0]ALUout	I	Alu 的输出
[31:0]DMout	I	DMout
[31:0]WD_W	O	W 级回写数据
[31:0]addr_W	O	W 级回写地址
RegWrite_W	O	W 级写寄存器控制信号

二、控制器设计

Controller

（1） 端口说明

Cotroller 端口定义表

信号名	方向	描述
IR[31:0]	I	当前流水级的 IR
D_IR[31:0]	I	D 级 IR
E_IR[31:0]	I	E 级 IR
M_IR[31:0]	I	M 级 IR
W_IR[31:0]	I	W 级 IR
RegWrite	O	寄存器写入控制
MemRead	O	内存读取控制
MemWrite	O	内存写入控制
IMMsel[1:0]	O	立即数选择信号
PCsel[2:0]	O	nPC 选择信号
ALUop[3:0]	O	ALU 功能选择信号
Mul_A3 [1:0]	O	写寄存器的目标寄存器来自 rt/rd/31 操作数
Mul_WD [1:0]	O	写入寄存器数据选择信号
Z_D_rs[2:0]	O	D 级 rs 转发选择信号
Z_D_rt[2:0]	O	D 级 rt 转发选择信号
Z_E_rs[2:0]	O	E 级 rs 转发选择信号
z_E_rt[2:0]	O	E 级 rt 转发选择信号
Z_M_rt[1:0]	O	M 级 rt 转发选择信号

（2）控制信号意义

控制信号	功能定义	功能描述
RegWrite	寄存器写入控制	寄存器写入使能有效
MemRead	内存读取控制	内存读取使能有效
MemWrite	内存写入控制	内存写入使能有效
Mul_A3[1:0]	写寄存器的目标寄存器来自 rt/rd/31 操作数	0： rt 1： rd 2: 目标寄存器 31
Mul_WD [1:0]	写入寄存器数据选择信号	00：向寄存器写入 DM 读取结果 01：向寄存器写入从 ALU 计算的值

		10：向寄存器写入 pc+8
PCsel[2:0]	nPC 选择信号	00：pc+4 01：pc+4+shift2 or pc+4 10：PC31…28||shift2 11：Rdata1
ALUop[3:0]	ALU 功能选择信号	0000：A&B 0001：A|B 0010：A+B 0110：A-B
Imm_sel [1:0]	Data2-imm 选择信号	00：data2 01：sign-ext 10：0-ext 11：加载立即数到高位
Z_D_rs[2:0]	D 级 rs 转发选择信号	000：Rdata1 001：PC4@E+4 010：high@E 011：PC4@M+4 100：ALUout@M
Z_D_rt[2:0]	D 级 rt 转发选择信号	000：Rdata2 001：PC4@E+4 010：high@E 011：PC4@M+4 100：ALUout@M
Z_E_rs[2:0]	E 级 rs 转发选择信号	000：data1 001：PC4@M+4 010：ALUout@M 011：M_WD
z_E_rt[2:0]	E 级 rt 转发选择信号	000：data2 001：PC4@M+4 010：ALUout@M

		011：M_WD
Z_M_rt[1:0]	M 级 rt 转发选择信号	00：data2 01：M_WD

（3）控制信号真值表

见 excel

（4）暂停信号真值表

见 excel

（5）转发信号真值表

见 excel

三、测试代码

08000c8b	08000c20	3c108000	35ce0004	00892026	3c017777
00000000	381a0001	02042825	308f000c	00aa2826	3424b5ea
23bdfff4	381a0002	00858027	21ef0004	00800011	3c01d70a
afb00000	8fb00000	24080014	30b8000c	00004810	34251a79
afb10004	03e00008	ac040000	1880000c	00a00013	00850018
afb20008	27bd0004	ac050004	8de5fffc	00005012	00002010
00048020	23bdfffc	ac040008	8f040004	354a0001	00002012
00058821	afb00000	ac05000c	81a90001	012a001a	0c000c02
24020000	04800002	ac040010	91aaffff	00004810	00000000
12200007	00000000	ac050014	85cb0002	00005012	00023021
32320001	3c1b418a	ac040018	95ccfffe	012a4826	0c000c16
12400002	1c800002	ac05001c	00892026	00a92826	00000000
00000000	00000000	19000041	008b2026	34840001	2008308c
00501021	3b7b938a	00852026	00aa2826	0124001b	01003809
00118842	18800002	00852826	00ac2826	00004810	00000000
08000c09	00000000	0085482a	08000c71	00005012	0c000c35
00108040	0364d807	0085502b	00000000	00892026	00040022
8fb00000	04810002	012a4826	2010308c	008a2020	24040000
8fb10004	00000000	00042043	02003809	08000c46	24050000
8fb20008	009bd806	2889ffff	00000000	2108ffff	24060000
03e00008	037bd804	2caaffff	ade5fffc	8fb00000	24070000
27bd000c	8fb00000	00852023	af040004	03e00008	2008308c
23bdfffc	00e00008	00094fc0	a1a40001	27bd0004	01003809
afb00000	27bd0004	000a57c0	a1a5ffff	3c010002	00000000
00850019	23bdfffc	00892026	a5c50002	3430e6f1	1000ffff
00008012	afb00000	00aa2826	a5c4fffe	3c01000b	00000000
02068023	3c017fff	308d0003	00850018	34316d7b	
16000003	3430ffff	35ad0004	00004810	3c010007	
00000000	02042024	30ae0002	00005012	34326fa1	

四、思考题

1.为什么需要有单独的乘除法部件而不是整合进 ALU？为何需要有独立的 HI、LO 寄存器？

在 cpu 的实际工作中，乘除运算指令的执行时间要长于其他一般指令，将乘除部件整合进 ALU 可能会造成少量的乘除指令使得 cpu 整体执行效率边低的情况。将乘除模块独立出来并设置独立的 HI，LO 寄存器能大大缓解这一现象。

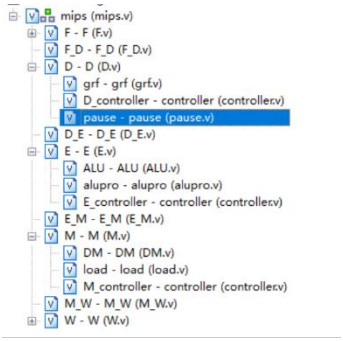
2.参照你对延迟槽的理解，试解释“乘除槽”。

在 E 级产生乘除指令堵塞信号时，D 级指令已经计入 D_E 流水线寄存器。为避免连续的乘除指令造成 bug，应在 D 级判断 D_E 级流水线寄存器中的指令是否为乘除指令，及时暂停。

3.举例说明并分析何时按字节访问内存相对于按字访问内存性能上更有优势。（Hint： 考虑 C 语言中字符串的情况）

按字存取可能读取大量无用数据。若按字访问内存 64 位 cpu 一次性访问 8 字节内存，在极端情况下可能只有 1 字节是所需数据，造成 cpu 吞吐效率低下。

4.如何概括你所设计的 CPU 的设计风格？为了对抗复杂性你采取了哪些抽象和规范手段？



多级嵌套，层次较为分明。分布式译码。

```
assign D1 = (beq || bne || bgez || bgtz || blez || bltz || jr || jalr);
assign z_D_rs = (D1 && jal_E && (rs === 31) && (rs !== 0))?1 :
                (D1 && jalr_E && (rs === rd_E) && (rs !== 0))?1 :
                (D1 && lui_E && (rs === rt_E) && (rs !== 0))?2 :
                (D1 && jal_M && (rs === 31) && (rs !== 0))?3 :
                (D1 && jalr_M && (rs === rd_M) && (rs !== 0))?3 :
                (D1 && (slti_M || sltiu_M || xori_M || andi_M || addiu_M || lui_M || ori_M)&& (rs === rt_M) && (rs !== 0))?4 :
                (D1 && (mfhi_M || mflo_M || srav_M || sra_M || srlv_M || srl_M || sllv_M || sll_M || slt_M || sltu_M || NOR_M || XOR_M || OR_M || AND_M || addu_M

assign D2 = (beq || bne);
assign z_D_rt = (D2 && jal_E && (rt === 31) && (rt !== 0))?1 :
                (D2 && jalr_E && (rt === rd_E) && (rt !== 0))?1 :
                (D2 && lui_E && (rt === rt_E) && (rt !== 0))?2 :
                (D2 && jal_M && (rt === 31) && (rt !== 0))?3 :
                (D2 && jalr_M && (rt === rd_M) && (rt !== 0))?3 :
                (D2 && (slti_M || sltiu_M || xori_M || andi_M || addiu_M || lui_M || ori_M) && (rt === rt_M) && (rt !== 0))?4 :
                (D2 && (mfhi_M || mflo_M || srav_M || sra_M || srlv_M || srl_M || sllv_M || sll_M || slt_M || sltu_M || NOR_M || XOR_M || OR_M || AND_M || addu_M
```

规划者（PLANNER）型暂停、转发控制。

为了对抗复杂性我对指令进行了一定的分类，在代码实现之前做了详细的控制信号表格和暂停转发表格。

5.你对流水线 CPU 设计风格有何见解？

“规划者”、“侦察者”型暂停转发控制在 p5 的实现上没有显著区别，但我觉得理论上“侦察者”型暂停转发控制和“标记转发法”对于 p6 这种大量的指令冲突设计有一定的优势，主要是加指令比较简便。

但是我认为设计流水线 cpu 最重要的是对自己代码的熟悉程度。“规划者”型转发控制加指令时量确实较大，改动地方较多。在进行了大量指令添加后（从 p5 加到 p6），各种指令的分类已经相当明确，后续的指令添加的思路其实比较清晰，在保证不出现非技术性错误的情况下也不难完成指令的添加。

6. 在本实验中你遇到了哪些不同指令类型组合产生的冲突？你又是如何解决的？相应的测试样例是什么样的？

IF/ID当前指令			E级寄存器			M级寄存器
指令类型	源寄存器	T_use	lhs|lw-rt|lb-rt|lh-rt|lhu-tr|lbu-rt	mfhi|mflo|srav|sra|srlv|srl|sllv|sll|slt|sltu|xor|nor|or|add|sub|addu-rd|subu_rd|and_rd	slti|sltiu|andi|xori|ori-rt|addiu|addi-rt	lw-rt|lb-rt|lh-rt|lhu-tr|lbu-rt|lhs
msub|mult|multu|div|divu|srav|srlv|sllv|slt|sltu|slti|sltiu|lw|lh|lb|lhu|lbu|sw|sh|sb|addu|subu|add|addi|addiu|sub|ori|andi|xori|and|or|xor|nor|lhs	rs	1	2	1	1	1
msub|mult|multu|div|divu|srav|sra|srlv|srl|sllv|sll|slt|sltu|addu|subu|add|sub|and|or|xor|nor	rt	1	2	1	1	1
beq|jr|jalr|bne|bgez|bgtz|blez|bltz|blezals	rs	0	2	1	1	1
beq|bne	rt	0	2	1	1	1

Instr	E				M(Irrelevent Instr Inserted)				W(Two Irrelevent Instr Inserted)			
	PC	ALU		DM	PC	ALU		DM	PC	ALU		DM
	jal,jalr	cal_r,cal_i	load_m	load	jal,jalr	cal_r,cal_i	load_m	load	jal,jalr	cal_r,cal_i	load_m	load
beq	N/A	addu $5, $4, $3 beq $5, $0, Label	mfhi $3 beq $3, $0, Label	lw $4, ($0) beq $4, $3, Label	jal N N:beq $31, $4, Label	addu $5, $4, $3 beq $5, $4, Label	mfhi $3 beq $3, $0, Label	lw $4, ($0) beq $4, $3, Label	jal N N:beq $31, $4, Label	addu $5, $4, $3 beq $5, $4, Label	mfhi $3 beq $3, $0, Label	lw $4, ($0) beq $4, $3, Label
jr,jalr	N/A	addi $31, $31, 4 jr $31	mflo $3 jalr $31, $3	lw $31, ($0) jr $31	jal N N:jr $31	addi $31, $31, 4 jr $31	mflo $3 jalr $31, $3	lw $31, ($0) jr $31	jal N N:jr $31	addi $31, $31, 4 jr $31	mflo $3 jalr $31, $3	lw $31, ($0) jr $31
cal_r,cal_i	jal Label Label:addi $4, $31, 4	ori $4, $3, 0x1234 addi $5, $4, 0x5678	mfhi $3 andi $4, $3, $2	lw $4, ($0) ori $5, $4, 0x1234	jalr $4, $5 ori $5, $4, 0x1234	ori $4, $3, 0x1234 addi $5, $4, 0x5678	mfhi $3 andi $4, $3, $2	lw $4, ($0) ori $5, $4, 0x1234	jalr $4, $5 ori $5, $4, 0x1234	ori $4, $3, 0x1234 addi $5, $4, 0x5678	mfhi $3 andi $4, $3, $2	lw $4, ($0) ori $5, $4, 0x1234
cal_m	jal N mult $31, $1	xor $3, $2, $1 mult $3, $1	mflo $3 mult $3, $1	lw $4, ($0) divu $4, $2	jal N mult $31, $1	xor $3, $2, $1 mult $3, $1	mflo $3 mult $3, $1	lw $4, ($0) divu $4, $2	jal N mult $31, $1	xor $3, $2, $1 mult $3, $1	mflo $3 mult $3, $1	lw $4, ($0) divu $4, $2
load	jal Label Label:lb $4, ($31)	ori $4, $3, 0x1234 lb $5, ($4)	mfhi $3 lw $1, ($3)	lw $4, ($0) lw $5, ($4)	jal N N:lb $5, ($31)	ori $4, $3, 0x1234 lb $5, ($4)	mfhi $3 lw $1, ($3)	lw $4, ($0) lw $5, ($4)	jal N N:lb $5, ($31)	ori $4, $3, 0x1234 lb $5, ($4)	mfhi $3 lw $1, ($3)	lw $4, ($0) lw $5, ($4)
store_m	jal N mthi $31	and $3, $2, $1 mthi $3	mfhi $3 mthi $3	lw $3, -3000($ra) mthi $3	jal N mthi $31	and $3, $2, $1 mthi $3	mfhi $3 mthi $3	lw $3, -3000($ra) mthi $3	jal N mthi $31	and $3, $2, $1 mthi $3	mfhi $3 mthi $3	lw $3, -3000($ra) mthi $3
store	jal Label Label:sb $4, ($31)	ori $4, $3, 0x1234 sb $5, ($4)	mfhi $3 sw $1, ($3)	lw $4, ($0) sw $5, ($4)	jal N N:sw $5, ($31)	ori $4, $3, 0x1234 sb $5, ($4)	mfhi $3 sw $1, ($3)	lw $4, ($0) sw $5, ($4)	jal N N:sw $5, ($31)	ori $4, $3, 0x1234 sb $5, ($4)	mfhi $3 sw $1, ($3)	lw $4, ($0) sw $5, ($4)
beq	N/A	addu $5, $4, $3 beq $4, $5, Label	mfhi $3 beq $0, $3, Label	lw $4, ($0) beq $3, $4, Label	jal N N:beq $4, $31, Label	addu $5, $4, $3 beq $4, $5, Label	mfhi $3 beq $0, $3, Label	lw $4, ($0) beq $3, $4, Label	jal N N:beq $4, $31, Label	addu $5, $4, $3 beq $4, $5, Label	mfhi $3 beq $0, $3, Label	lw $4, ($0) beq $3, $4, Label
cal_r	jal Label Label:addu $5, $4, $31	ori $4, $3, 0x1234 addu $5, $3, $4	mfhi $3 and $4, $2, $3	lw $4, ($0) addu $5, $3, $4	jalr $4, $5 addu $5, $3, $4	ori $4, $3, 0x1234 addu $5, $3, $4	mfhi $3 and $4, $2, $3	lw $4, ($0) addu $5, $3, $4	jalr $4, $5 addu $5, $3, $4	ori $4, $3, 0x1234 addu $5, $3, $4	mfhi $3 and $4, $2, $3	lw $4, ($0) addu $5, $3, $4
cal_m	jal N mult $1, $31	xori $3, $2, 0xffffffff mult $1, $3	mflo $3 div $3, $1	lw $4, ($0) multu $2, $4	jal N mult $1, $31	xori $3, $2, 0xffffffff mult $1, $3	mflo $3 div $3, $1	lw $4, ($0) multu $2, $4	jal N mult $1, $31	xori $3, $2, 0xffffffff mult $1, $3	mflo $3 div $3, $1	lw $4, ($0) multu $2, $4
store	jal Label Label:sb $31, ($0)	ori $4, $3, 0x1234 sb $4, ($0)	mflo $3 sw $3, 4($0)	lw $4, ($0) sw $4, ($0)	jal N N:sw $31, ($0)	ori $4, $3, 0x1234 sb $4, ($0)	mflo $3 sw $3, 4($0)	lw $4, ($0) sw $4, ($0)	jal N N:sw $31, ($0)	ori $4, $3, 0x1234 sb $4, ($0)	mflo $3 sw $3, 4($0)	lw $4, ($0) sw $4, ($0)