P6　Verilog 流水线 CPU 设计文档

一、 整体结构

本文档所描述的处理器为 32 位五级流水线处理器，采用 Verilog HDL 实现。该处理器支持的指令集为 MIPS-C3={LB、LBU、LH、LHU、LW、SB、SH、SW、ADD、ADDU、SUB、 SUBU、 MULT、MULTU、DIV、DIVU、SLL、SRL、SRA、SLLV、SRLV、SRAV、AND、OR、XOR、NOR、ADDI、ADDIU、ANDI、ORI、XORI、LUI、SLT、SLTI、SLTIU、SLTU、BEQ、BNE、BLEZ、BGTZ、BLTZ、BGEZ、J、JAL、JALR、JR、MFHI、MFLO、MTHI、MTLO}，且支持延迟槽。

该处理器采用模块化和层次化设计，包含 Controller（控制器）、PC（程序计数器）、IM（指令存储器）、Adder（加法器）、NPC（下一条指令地址计算单元）、GRF（通用寄存器组）、CMP（比较单元）、ALU（算术逻辑单元）、MDU（乘除单元）、DM（数据存储器）、EXT（位扩展器）等基本部件。处理器顶层有效驱动信号有时钟信号 clk 和复位信号 reset。

二、 基本模块规格

1. PC（程序计数器）

1) 基本描述

PC 存储当前指令地址，并在时钟上升沿更新值。PC 的容量为 32bit*4096。

2) 端口说明

表 1 PC 端口说明

信号名	方向	描述
Clk	I	时钟信号。
Reset	I	复位信号（高电平有效）。
En	I	写使能信号（高电平有效）。
In[31:0]	I	输入下一个时钟上升沿 PC 要写入的值。
Out[31:0]	O	输出当前 PC 的值。

3) 功能定义

表 2 PC 功能定义

序号	功能名称	功能描述
1	同步复位	当时钟上升沿到来时，若复位信号有效，PC 被设置为 0x00003000。
2	输出当前指令地址	存储并由 Out 输出当前指令地址。
3	更新 PC	当时钟上升沿到来时，若写使能信号有效且复位信号无效，则更新 PC 为 In 的值。

2. IM（指令存储器）

1) 基本描述

IM 存储 CPU 要执行的指令，并输出输入地址所对应的指令。IM 的容量为 32bit*1024。

2) 端口说明

表 3 IM 端口说明

信号名	方向	描述
Addr[31:0]	I	输入指令地址。
Instr[31:0]	O	输出 Addr 所对应的指令。

3) 功能定义

表 4 IM 功能定义

序号	功能名称	功能描述
1	输出当前指令	由 Instr 输出 Addr 所对应的指令。

3. Adder（加法器）

1) 基本描述

Adder 输入当前 PC 的值，输出 PC+4 的值。

2) 端口说明

表 5 Adder 端口说明

信号名	方向	描述
In[31:0]	I	数据输入信号，输入当前 PC 的值。
Out[31:0]	O	数据输出信号，输出 PC+4 的值。

3) 功能定义

表 6 Adder 功能定义

序号	功能名称	功能描述
1	PC+4	由 Out 输出 PC+4 的值。

4. NPC（下一条指令地址计算单元）

1) 基本描述

NPC 根据当前指令地址和相应的控制信号计算出下一条指令地址，并输出 PC+8 的值。

2) 端口说明

表 7 NPC 端口说明

信号名	方向	描述
NPCOp	I	指定 NPC 要执行的操作： 0：B 型指令； 1：J 型指令。
PC4[31:0]	I	输入 PC+4。
Imm26[25:0]	I	输入 26 位立即数，用于计算分支或跳转后 PC 的值。
Out[31:0]	O	输出计算出的下一条指令地址。
PC8[31:0]	O	输出 PC+8。

3) 功能定义

表 8 NPC 功能定义

序号	功能名称	功能描述
1	计算下一条指令地址	当 NPCOp 为 0 时，Out 输出 PC4+SignExt(Imm26[15:0]||0^2)； 当 NPCOp 为 1 时，Out 输出(PC4)[31:28]||Imm26||0^2。
2	输出 PC+8	由 PC8 输出 PC+8 的值。

5. GRF（通用寄存器组）

1) 基本描述

GRF 内部包括 32 个具有复位功能的寄存器。其中，0 号寄存器的值始终保持为 0，其他寄存器初始值均为 0。GRF 提供同时读取 2 个寄存器和写入 1 个寄存器的功能，并且支持内部转发。

2) 端口说明

表 9 GRF 端口说明

信号名	方向	描述
Clk	I	时钟信号。
Reset	I	复位信号（高电平有效）。
A1[4:0]	I	地址输入信号 1，将其对应寄存器中存储的数据输出至 RD1。
A2[4:0]	I	地址输入信号 2，将其对应寄存器中存储的数据输出至 RD2。
A3[4:0]	I	地址输入信号 3，指定写入操作所对应的寄存器。
WD[31:0]	I	数据输入信号，即要写入寄存器中的数据。
RD1[31:0]	O	数据输出信号，输出 A1 对应寄存器中的 32 位数据。
RD2[31:0]	O	数据输出信号，输出 A2 对应寄存器中的 32 位数据。

注：在 Verilog 实现中增加了 WPC[31:0]输入，用于在线测试时输出 PC 的值。

3) 功能定义

表 10 GRF 功能定义

序号	功能名称	功能描述
1	同步复位	当时钟上升沿到来时，若复位信号有效，GRF 中的每一个寄存器都被设置为 0x00000000。
2	读取数据	读取 A1 和 A2 所对应寄存器中的数据至 RD1 和 RD2（当同一个寄存器同时被写入和读取时，读取的值为写入的值）。
3	写入数据	当时钟上升沿到来时，如果复位信号无效，就将 WD 写入 A3 所对应的寄存器中。

6. CMP（比较单元）

1) 基本描述

CMP 对输入的两个操作数提供相等比较功能，并对输入的第一个操作数提供零比较功能，输出比较结果。

2) 端口说明

表 11 CMP 端口说明

信号名	方向	描述
A[31:0]	I	数据输入信号，输入 CMP 的第一个操作数。
B[31:0]	I	数据输入信号，输入 CMP 的第二个操作数。
Equal	O	相等标志信号（高电平有效），标志两操作数是否相等。
LTZ	O	小于 0 标志信号（高电平有效），标志 A 是否小于 0。
EQZ	O	等于 0 标志信号（高电平有效），标志 A 是否等于 0。

3) 功能定义

表 12 CMP 功能定义

序号	功能名称	功能描述

1	相等比较运算	若 A=B，则 Equal 信号有效；否则无效。
2	零比较运算	若 A<0，则 LTZ 信号有效；否则无效。 若 A=0，则 EQZ 信号有效；否则无效。

7. ALU（算术逻辑单元）

1) 基本描述

ALU 对输入的两个操作数提供 32 位加、减、或、与、或非、异或和移位运算以及小于置位功能，输出运算结果。

2) 端口说明

表 13 ALU 端口说明

信号名	方向	描述
A[31:0]	I	数据输入信号，输入 ALU 的第一个操作数。
B[31:0]	I	数据输入信号，输入 ALU 的第二个操作数。
ALUOp[3:0]	I	指定 ALU 所要进行的操作： 0000：A+B； 0001：A-B； 0010：A|B； 0011：A&B； 0100：~(A|B)； 0101：A^B； 0110：B<<A[4:0]； 0111：B>>A[4:0]； 1000：B>>>A[4:0]； 1001：(A<B)?1:0； 1010：((0|A)<(0|B))?1:0。
Result[31:0]	O	数据输出信号，输出 ALU 的计算结果。

3) 功能定义

表 14 ALU 功能定义

序号	功能名称	功能描述
1	加法	当 ALUOp 为 0000 时，Result 输出 A+B 的值。
2	减法	当 ALUOp 为 0001 时，Result 输出 A-B 的值。
3	或运算	当 ALUOp 为 0010 时，Result 输出 A|B 的值。
4	与运算	当 ALUOp 为 0011 时，Result 输出 A&B 的值。
5	或非运算	当 ALUOp 为 0100 时，Result 输出~(A|B)的值。
6	异或运算	当 ALUOp 为 0101 时，Result 输出 A^B 的值。
7	逻辑左移运算	当 ALUOp 为 0110 时，Result 输出 B<<A[4:0]的值。
8	逻辑右移运算	当 ALUOp 为 0111 时，Result 输出 B>>A[4:0]的值。
9	算术右移运算	当 ALUOp 为 1000 时，Result 输出 B>>>A[4:0]的值。
10	小于比较运算	当 ALUOp 为 1001 时，若 A<B，则 Result 输出 1；否则 Result 输出 0。
11	无符号小于比较运算	当 ALUOp 为 1010 时，若(0|A)<(0|B)，则 Result 输出 1；否则 Result 输出 0。

8. MDU（乘除单元）

1) 基本描述

MDU 用于计算乘除法，内置 HI 和 LO 两个寄存器用于保存计算结果，具有启动信号和忙标记。

2) 端口说明

表 15 MDU 端口说明

信号名	方向	描述
Clk	I	时钟信号。
Reset	I	复位信号（高电平有效）。
Start	I	开始计算信号（高电平有效）。
MDUOp[1:0]	I	指定操作： 00：无符号乘法； 01：有符号乘法； 10：无符号除法； 11：有符号除法。
HIWrite	I	HI 寄存器写使能（高电平有效）。
LOWrite	I	LO 寄存器写使能（高电平有效）。
A[31:0]	I	数据输入信号，输入第一个操作数。
B[31:0]	I	数据输入信号，输入第二个操作数。
Busy	O	忙标记（高电平有效）。
HI[31:0]	O	数据输出信号，输出 HI 寄存器的数据。
LO[31:0]	O	数据输出信号，输出 LO 寄存器的数据。

3) 功能定义

表 16 MDU 功能定义

序号	功能名称	功能描述
1	同步复位	当时钟上升沿到来时，若复位信号有效，HI 和 LO 都被设置为 0x00000000。
2	无符号乘法	复位信号无效，Start 信号有效且 Op 信号为 00 后的第一个时钟上升沿后开始计算 A×B（无符号），5 个周期后将 64 位结果的高低半部分分别存入 HI 和 LO 寄存器。
3	有符号乘法	复位信号无效，Start 信号有效且 Op 信号为 01 后的第一个时钟上升沿后开始计算 A×B，5 个周期后将 64 位结果的高低半部分分别存入 HI 和 LO 寄存器。
4	无符号除法	复位信号无效，Start 信号有效且 Op 信号为 10 后的第一个时钟上升沿后开始计算 A÷B（无符号），10 个周期后将余数和商分别存入 HI 和 LO 寄存器。
5	有符号除法	复位信号无效，Start 信号有效且 Op 信号为 11 后的第一个时钟上升沿后开始计算 A÷B，10 个周期后将余数和商分别存入 HI 和 LO 寄存器。
6	存入 HI	当时钟上升沿到来时，若 HIWrite 信号有效且复位信号无效，则将 A 存入 HI。
7	存入 LO	当时钟上升沿到来时，若 LOWrite 信号有效且复位信号无效，则将 A 存入 LO。

9. DM（数据存储器）

1) 基本描述

DM 用于存储数据，其容量为 32bit*4096，起始地址为 0x00000000。DM 支持同步复位功能，并且数据读取和写入端口分离。

2) 端口说明

表 17 DM 端口说明

信号名	方向	描述
Clk	I	时钟信号。
Reset	I	复位信号（高电平有效）。
Addr[31:0]	I	地址信号，指定要操作的存储单元的地址。
WD[31:0]	I	数据输入信号，输入要写入到 Addr 所对应的存储单元的数据。
MemWrite	I	写使能信号（高电平有效）。
OpWidth[1:0]	I	指定操作位宽： 00：Word； 01：Half； 10：Byte。
LoadSigned	I	指定是否进行有符号读取（高电平有效）。
RD[31:0]	O	数据输出信号，输出 Addr 所对应的存储单元的数据。

注：在 Verilog 实现中增加了 WPC[31:0]输入，用于在线测试时输出 PC 的值。

3) 功能定义

表 18 DM 功能定义

序号	功能名称	功能描述
1	同步复位	当时钟上升沿到来时，若复位信号有效，DM 中的每一个存储单元都被设置为 0x00000000。
2	读取	RD 根据 OpWidth 和 LoadSigned 信号输出 Addr 所对应的存储单元的数据。
3	写入	当时钟上升沿到来时，如果 MemWrite 有效且复位信号无效，就根据 OpWidth 信号将 WD 写入 Addr 所对应的存储单元中。

10. EXT（位扩展器）

1) 基本描述

EXT 用于将输入的 16 位立即数根据操作信号扩展成 32 位并输出。

2) 端口说明

表 19 EXT 端口说明

信号名	方向	描述
Imm16[15:0]	I	数据输入信号，输入要进行扩展的数据。
ExtOp[1:0]	I	符号扩展信号（高电平有效）。
Imm32[31:0]	O	数据输出信号，输出扩展后的数据。

3) 功能定义

表 20 EXT 功能定义

序号	功能名称	功能描述
1	无符号扩展	当 ExtOp 为 00 时，将 Imm16 无符号扩展至 32 位并输出至 Imm32。
2	符号扩展	当 ExtOp 为 01 时，将 Imm16 符号扩展至 32 位并输出至 Imm32。
3	左移 16 位	当 ExtOp 为 10 时，将 Imm16 左移 16 位并输出至 Imm32。

三、 数据通路设计

见 Excel 表格。

四、 数据通路参考示意图

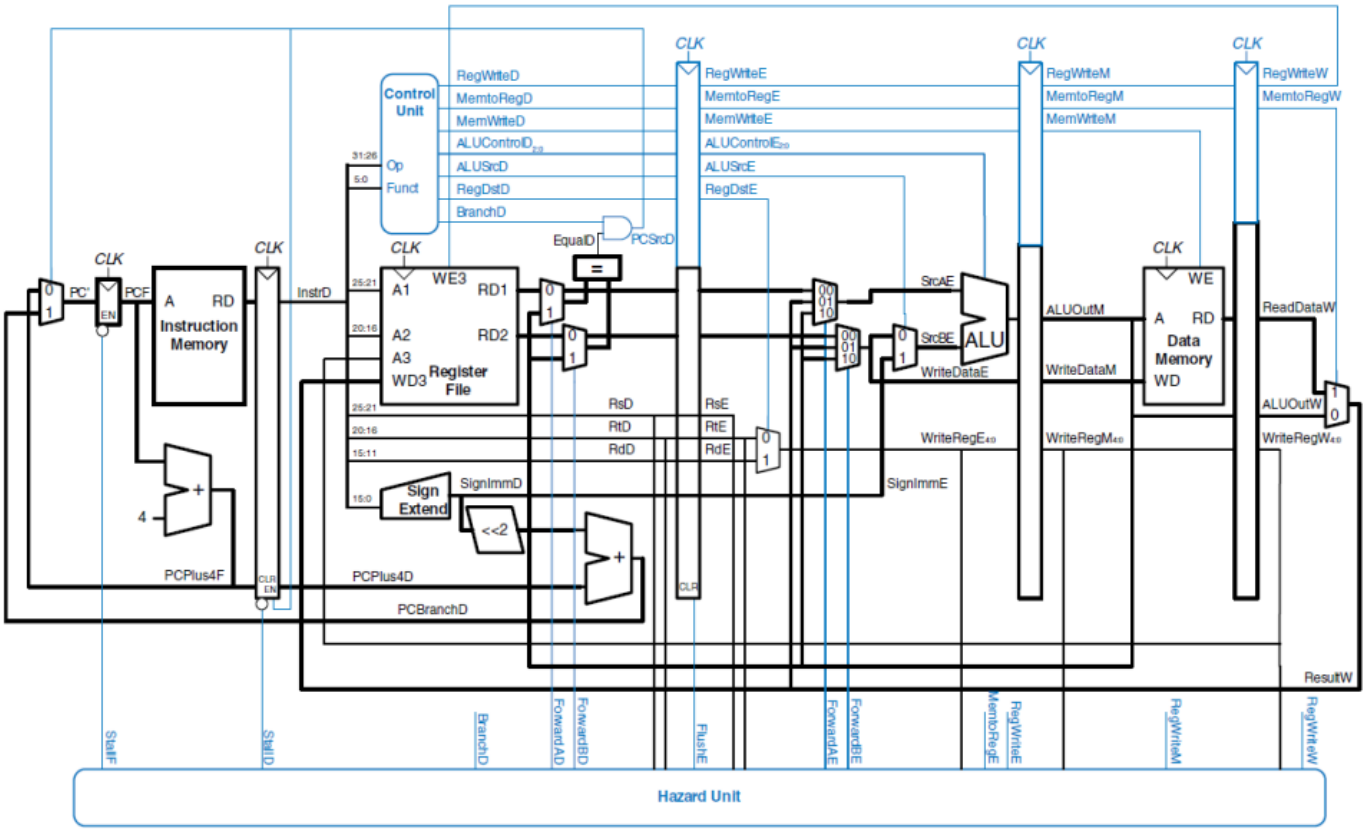

Figure 7.58 Pipelined processor with full hazard handling

五、 控制器（Controller）设计

1. 基本描述

控制器分为主控制器和冒险处理单元。主控制器通过输入的 Op 和 Funct 信号以及 CMP 产生的比较信号产生数据通路所需的控制信号，采用分布式译码，实例化 3 个；冒险处理单元负责为几个转发位点提供数据转发并通过检测无法由转发解决的数据冒险来插入暂停。

2. D 级主控制器真值表

表 21 D 级主控制器真值表

指令(Op/Funct)	NPCOp	ExtOp[1:0]	PCSrc[1:0]	A3Sel[1:0]	GenD	MD	D1Use	D2Use
addu (000000/100001)	x	xx	00	10	0	0	0	0
subu (000000/100011)	x	xx	00	10	0	0	0	0
ori (001101)	x	00	00	11	0	0	0	0
lw (100011)	x	01	00	11	0	0	0	0
sw (101011)	x	01	00	00	0	0	0	0
beq (000100)	0	01	CMP.Equal?01:00	00	0	0	1	1
lui	x	10	00	11	0	0	0	0

(001111)								
j (000010)	1	xx	01	00	0	0	0	0
jal (000011)	1	xx	01	01	1	0	0	0
jr (000000/001000)	x	xx	10	00	0	0	1	0
lb (100000)	x	01	00	11	0	0	0	0
lbu (100100)	x	01	00	11	0	0	0	0
lh (100001)	x	01	00	11	0	0	0	0
lhu (100101)	x	01	00	11	0	0	0	0
sb (101000)	x	01	00	00	0	0	0	0
sh (101001)	x	01	00	00	0	0	0	0
add (000000/100000)	x	xx	00	10	0	0	0	0
sub (000000/100010)	x	xx	00	10	0	0	0	0
mult (000000/011000)	x	xx	00	00	0	1	0	0
multu (000000/011001)	x	xx	00	00	0	1	0	0
div (000000/011010)	x	xx	00	00	0	1	0	0
divu (000000/011011)	x	xx	00	00	0	1	0	0
sll (000000/000000)	x	xx	00	10	0	0	0	0
srl (000000/000010)	x	xx	00	10	0	0	0	0
sra (000000/000011)	x	xx	00	10	0	0	0	0
sllv (000000/000100)	x	xx	00	10	0	0	0	0
srlv (000000/000110)	x	xx	00	10	0	0	0	0
srav (000000/000111)	x	xx	00	10	0	0	0	0
and (000000/100100)	x	xx	00	10	0	0	0	0
or (000000/100101)	x	xx	00	10	0	0	0	0
xor	x	xx	00	10	0	0	0	0

(000000/100110)								
nor (000000/100111)	x	xx	00	10	0	0	0	0
addi (001000)	x	01	00	11	0	0	0	0
addiu (001001)	x	01	00	11	0	0	0	0
andi (001100)	x	00	00	11	0	0	0	0
xori (001110)	x	00	00	11	0	0	0	0
slt (000000/101010)	x	xx	00	10	0	0	0	0
slti (001010)	x	01	00	11	0	0	0	0
sltiu (001011)	x	01	00	11	0	0	0	0
sltu (000000/101011)	x	xx	00	10	0	0	0	0
bne (000101)	0	01	!CMP.Equal?01:00	00	0	0	1	1
blez (000110)	0	01	(CMP.LTZ||CMP.EQZ)?01:00	00	0	0	1	0
bgtz (000111)	0	01	(!CMP.LTZ&&!CMP.EQZ)?01:00	00	0	0	1	0
bltz (000001/Instr[20:16]=00000)	0	01	CMP.LTZ?01:00	00	0	0	1	0
bgez (000001/Instr[20:16]=00001)	0	01	!CMP.LTZ?01:00	00	0	0	1	0
jalr (000000/001001)	x	xx	10	10	1	0	1	0
mfhi (000000/010000)	x	xx	00	10	0	1	0	0
mflo (000000/010010)	x	xx	00	10	0	1	0	0
mthi (000000/010001)	x	xx	00	00	0	1	0	0
mtlo (000000/010011)	x	xx	00	00	0	1	0	0

3. E 级主控制器真值表

表 22 E 级主控制器真值表

指令(Op/Funct)	ALUOp[3:0]	ALUSrcA	ALUSrcB	Start	MDUOp[1:0]	HIWrite	LOWrite	GenE[1:0]	E1Use	E2Use
addu (000000/100001)	0000	0	0	0	xx	0	0	01	1	1

subu (000000/100011)	0001	0	0	0	xx	0	0	01	1	1
ori (001101)	0010	0	1	0	xx	0	0	01	1	0
lw (100011)	0000	0	1	0	xx	0	0	00	1	0
sw (101011)	0000	0	1	0	xx	0	0	00	1	0
beq (000100)	xxxx	x	x	0	xx	0	0	00	0	0
lui (001111)	0000	0	1	0	xx	0	0	01	0	0
j (000010)	xxxx	x	x	0	xx	0	0	00	0	0
jal (000011)	xxxx	x	x	0	xx	0	0	00	0	0
jr (000000/001000)	xxxx	x	x	0	xx	0	0	00	0	0
lb (100000)	0000	0	1	0	xx	0	0	00	1	0
lbu (100100)	0000	0	1	0	xx	0	0	00	1	0
lh (100001)	0000	0	1	0	xx	0	0	00	1	0
lhu (100101)	0000	0	1	0	xx	0	0	00	1	0
sb (101000)	0000	0	1	0	xx	0	0	00	1	0
sh (101001)	0000	0	1	0	xx	0	0	00	1	0
add (000000/100000)	0000	0	0	0	xx	0	0	01	1	1
sub (000000/100010)	0001	0	0	0	xx	0	0	01	1	1
mult (000000/011000)	xxxx	x	x	1	01	0	0	00	1	1
multu (000000/011001)	xxxx	x	x	1	00	0	0	00	1	1
div (000000/011010)	xxxx	x	x	1	11	0	0	00	1	1
divu (000000/011011)	xxxx	x	x	1	10	0	0	00	1	1
sll (000000/000000)	0110	1	0	0	xx	0	0	01	0	1
srl (000000/000010)	0111	1	0	0	xx	0	0	01	0	1
sra (000000/000011)	1000	1	0	0	xx	0	0	01	0	1

sllv (000000/000100)	0110	0	0	0	xx	0	0	01	1	1
srlv (000000/000110)	0111	0	0	0	xx	0	0	01	1	1
srav (000000/000111)	1000	0	0	0	xx	0	0	01	1	1
and (000000/100100)	0011	0	0	0	xx	0	0	01	1	1
or (000000/100101)	0010	0	0	0	xx	0	0	01	1	1
xor (000000/100110)	0101	0	0	0	xx	0	0	01	1	1
nor (000000/100111)	0100	0	0	0	xx	0	0	01	1	1
addi (001000)	0000	0	1	0	xx	0	0	01	1	0
addiu (001001)	0000	0	1	0	xx	0	0	01	1	0
andi (001100)	0011	0	1	0	xx	0	0	01	1	0
xori (001110)	0101	0	1	0	xx	0	0	01	1	0
slt (000000/101010)	1001	0	0	0	xx	0	0	01	1	1
slti (001010)	1001	0	1	0	xx	0	0	01	1	0
sltiu (001011)	1010	0	1	0	xx	0	0	01	1	0
sltu (000000/101011)	1010	0	0	0	xx	0	0	01	1	1
bne (000101)	xxxx	x	x	0	xx	0	0	00	0	0
blez (000110)	xxxx	x	x	0	xx	0	0	00	0	0
bgtz (000111)	xxxx	x	x	0	xx	0	0	00	0	0
bltz (000001/Instr[20:16]= 00000)	xxxx	x	x	0	xx	0	0	00	0	0
bgez (000001/Instr[20:16]= 00001)	xxxx	x	x	0	xx	0	0	00	0	0
jalr (000000/001001)	xxxx	x	x	0	xx	0	0	00	0	0
mfhi (000000/010000)	xxxx	x	x	0	xx	0	0	10	0	0
mflo (000000/010010)	xxxx	x	x	0	xx	0	0	11	0	0

mthi (000000/010001)	xxxx	x	x	0	xx	1	0	00	1	0
mtlo (000000/010011)	xxxx	x	x	0	xx	0	1	00	1	0

4. M 级主控制器真值表

表 23 M 级主控制器真值表

指令(Op/Funct)	MemWrite	OpWidth[1:0]	LoadSigned	GenM	M2Use
addu (000000/100001)	0	xx	x	0	0
subu (000000/100011)	0	xx	x	0	0
ori (001101)	0	xx	x	0	0
lw (100011)	0	00	x	1	0
sw (101011)	1	00	x	0	1
beq (000100)	0	xx	x	0	0
lui (001111)	0	xx	x	0	0
j (000010)	0	xx	x	0	0
jal (000011)	0	xx	x	0	0
jr (000000/001000)	0	xx	x	0	0
lb (100000)	0	10	1	1	0
lbu (100100)	0	10	0	1	0
lh (100001)	0	01	1	1	0
lhu (100101)	0	01	0	1	0
sb (101000)	1	10	x	0	1
sh (101001)	1	01	x	0	1
add (000000/100000)	0	xx	x	0	0
sub (000000/100010)	0	xx	x	0	0
mult (000000/011000)	0	xx	x	0	0
multu (000000/011001)	0	xx	x	0	0

div (000000/011010)	0	xx	x	0	0
divu (000000/011011)	0	xx	x	0	0
sll (000000/000000)	0	xx	x	0	0
srl (000000/000010)	0	xx	x	0	0
sra (000000/000011)	0	xx	x	0	0
sllv (000000/000100)	0	xx	x	0	0
srlv (000000/000110)	0	xx	x	0	0
srav (000000/000111)	0	xx	x	0	0
and (000000/100100)	0	xx	x	0	0
or (000000/100101)	0	xx	x	0	0
xor (000000/100110)	0	xx	x	0	0
nor (000000/100111)	0	xx	x	0	0
addi (001000)	0	xx	x	0	0
addiu (001001)	0	xx	x	0	0
andi (001100)	0	xx	x	0	0
xori (001110)	0	xx	x	0	0
slt (000000/101010)	0	xx	x	0	0
slti (001010)	0	xx	x	0	0
sltiu (001011)	0	xx	x	0	0
sltu (000000/101011)	0	xx	x	0	0
bne (000101)	0	xx	x	0	0
blez (000110)	0	xx	x	0	0
bgtz (000111)	0	xx	x	0	0
bltz (000001/Instr[20:16]=00000)	0	xx	x	0	0

bgez (000001/Instr[20:16]=00001)	0	xx	x	0	0
jalr (000000/001001)	0	xx	x	0	0
mfhi (000000/010000)	0	xx	x	0	0
mflo (000000/010010)	0	xx	x	0	0
mthi (000000/010001)	0	xx	x	0	0
mtlo (000000/010011)	0	xx	x	0	0

5. 暂停策略

采用标记转发法，当需求寄存器的值尚未算出时暂停，具体策略如下：

```
assign StallD = (D1Use && A1D == A3E && A3E != 0 && WDE === 32'bz) ||
                (D1Use && A1D == A3M && A3M != 0 && WDM === 32'bz && !(A1D
== A3E && A3E != 0 && WDE !== 32'bz)) ||
                (D2Use && A2D == A3E && A3E != 0 && WDE === 32'bz) ||
                (D2Use && A2D == A3M && A3M != 0 && WDM === 32'bz && !(A2D
== A3E && A3E != 0 && WDE !== 32'bz)) ||
                ((Start || Busy) && MD);
assign StallE = (E1Use && A1E == A3M && A3M != 0 && WDM === 32'bz) ||
                (E2Use && A2E == A3M && A3M != 0 && WDM === 32'bz);

   assign PCEn = ~(StallD || StallE);
   assign DRegEn = ~(StallD || StallE);
   assign ERegEn = ~StallE;
   assign ERegFlush = StallD;
   assign MRegFlush = StallE;
```

6. 转发策略

采用标记转发法，为每一个需求者增加转发，具体策略如下：

```
assign ForwardD1 = (A1D == A3E && A3E != 0) ? WDE :
                   (A1D == A3M && A3M != 0) ? WDM :
                   RD1D;
   assign ForwardD2 = (A2D == A3E && A3E != 0) ? WDE :
                   (A2D == A3M && A3M != 0) ? WDM :
```

```
                        RD2D;
    assign ForwardE1 = (A1E == A3M && A3M != 0) ? WDM :
                        (A1E == A3W && A3W != 0) ? WDW :
                        (A1E == A3T && A3T != 0) ? WDT :
                        RD1E;
    assign ForwardE2 = (A2E == A3M && A3M != 0) ? WDM :
                        (A2E == A3W && A3W != 0) ? WDW :
                        (A2E == A3T && A3T != 0) ? WDT :
                        RD2E;
    assign ForwardM2 = (A2M == A3W && A3W != 0) ? WDW :
                        RD2M;
```

六、 CPU 测试

1. 测试程序

```
lui $t0, 0x1234
ori $t0, $0, 0x5678
addi $s0, $0, 16
sb $t0, 1($0)
lb $t4, -15($s0)
lbu $t5, -15($s0)
sb $t0, -16($s0)
lb $t6, 0($0)
lbu $t7, 0($0)
sh $t0, 2($0)
lh $t4, -14($s0)
lhu $t5, -14($s0)
sh $t0, -14($s0)
lh $t6, 2($0)
lhu $t7, 2($0)
sw $t0, 4($0)
lw $t1, -12($s0)
sw $t0, -8($s0)
lw $t2, 8($0)
```

```
addu $t3, $t1, $t0
subu $t4, $t0, $t1
add $t7, $t1, $t0
sub $t8, $t0, $t1
li $a0, -1
mult $t0, $a0
mfhi $t3
multu $t0, $a0
mflo $t4
li $a0, -3
li $a1, 2
div $a0, $a1
mfhi $s1
mflo $s2
divu $a0, $a1
mfhi $s1
mflo $s2
mthi $t0
mtlo $t1
mfhi $s3
mflo $s4
sll $s1, $s1, 2
srl $s2, $s2, 3
sra $a0, $a0, 4
li $t5, 4
li $t6, 3
li $t7, 2
sllv $s1, $s1, $t5
srlv $s2, $s2, $t6
srav $a0, $a0, $t7
and $s3, $s2, $s1
or $s3, $s2, $s1
xor $s3, $s2, $s1
```

```
nor $s3, $s2, $s1
addiu $s3, $s3, -1
andi $s3, $s2, 0x1010
xori $s3, $s2, 0x1010
slt $s4, $t0, $t1
slti $s4, $t0, -1
sltiu $s4, $t0, -1
li $a0, -1
sltu $s4, $s4, $a0

LabelEQ:
beq $t2, $0, SkipEQ
lui $t5, 1
lui $s3, 1
beq $t0, $0, LabelEQ
nop
beq $t1, $t2, SkipEQ
nop
lui $t6, 1
SkipEQ:
nop

LabelNE:
bne $t2, $t1, SkipNE
lui $t5, 2
lui $s3, 2
bne $t1, $t2, LabelNE
nop
bne $t1, $0, SkipNE
nop
lui $t6, 2
SkipNE:
nop
```

```
li $s5, -1
li $s6, 1

LabelLEZ:
blez $s5, SkipLEZ
li $t5, 1
li $t6, 1
SkipLEZ:
blez $s6, LabelLEZ
nop

LabelLTZ:
bltz $s5, SkipLTZ
li $t5, 2
li $t6, 2
SkipLTZ:
bltz $s6, LabelLTZ
nop

LabelGTZ:
bgtz $s6, SkipGTZ
li $t5, 1
li $t6, 1
SkipGTZ:
bgtz $s5, LabelGTZ
nop

LabelGEZ:
bgez $s6, SkipGEZ
li $t5, 2
li $t6, 2
SkipGEZ:
```

```
bgez $s5, LabelGEZ
nop

jal Funct
lui $s4, 64
ori $t9, $0, 0x3054
la $a0, End
jalr $ra, $a0
lui $s5, 256

Funct:
ori $t8, $0, 16
jr $ra
lui $s6, 1024

Target:
lui $s7, 1027
j Cal_r
nop

End:
j Target
ori $s2, $0, 129

Cal_r:
addu $t2, $t1, $t0
subu $t3, $t2, $t1
addu $t2, $t1, $t0
subu $t3, $t1, $t2
addu $t2, $t1, $t0
ori $s0, $s0, 10
subu $t3, $t2, $t1
addu $t2, $t1, $t0
```

```
ori $s0, $s0, 1
subu $t3, $t1, $t2
lui $t2, 129
subu $t3, $t2, $t1
lui $t4, 129
subu $t3, $t1, $t4
lui $t2, 127
addu $s1, $s2, $s3
subu $t3, $t2, $t1
lui $t4, 127
subu $s1, $s2, $s3
subu $t3, $t1, $t4
lw $t4, 0($0)
addu $t3, $t4, $t1
lw $t5,4($0)
addu $t3, $t2, $t5
lw $t4, 0($0)
subu $s1, $s2, $s3
addu $t3, $t4, $t1
lw $t5,4($0)
subu $s1, $s2, $s3
addu $t3, $t2, $t5
jal Label1
addu $s4, $ra, $0
Label1:
jal Label2
addu $s5, $0, $ra
Label2:
jal Label3
nop
Label3:
addu $s4, $ra, $0
jal Label4
```

```
nop
Label4:
addu $s5, $0, $ra
la $a0, Label50
jalr $ra, $a0
addu $s4, $ra, $0
Label50:
la $a0, Label51
jalr $ra, $a0
addu $s5, $0, $ra
Label51:
la $a0, Label52
jalr $ra, $a0
nop
Label52:
addu $s4, $ra, $0
la $a0, Label53
jalr $ra, $a0
nop
Label53:
addu $s5, $0, $ra
mthi $t0
mfhi $s4
addu $s5, $s4, $t0
mtlo $t1
mflo $s5
addu $s4, $t0, $s5
mthi $t0
mfhi $s4
mult $t0, $s4
addu $s5, $s4, $t0
mtlo $t1
mflo $s5
```

```
mult $t1, $s5
addu $s4, $t0, $s5
sll $s4, $t0, 5
subu $s5, $s4, $t1
sll $s4, $t1, 4
subu $s5, $t0, $s4
sll $s4, $t0, 5
srl $s5, $t1, 4
subu $s5, $s4, $t1
sll $s4, $t1, 4
sra $s5, $t0, 3
subu $s5, $t0, $s4

Cal_i:
addu $t2, $t1, $t0
ori $t3, $t2, 31
addu $t2, $t1, $t3
ori $s0, $s0, 10
ori $t3, $t2, 127
lui $t2, 129
ori $t3, $t2, 1
lui $t2, 127
addu $s1, $s2, $s3
ori $t3, $t2, 6
lw $t4, 0($0)
ori $t3, $t4, 98
lw $t5, 4($0)
subu $s1, $s2, $s3
ori $t3, $t5, 101
jal Label5
ori $s4, $ra, 6
Label5:
jal Label6
```

```
nop
Label6:
ori $s4, $ra, 9
la $a0, Label54
jalr $ra, $a0
addi $s4, $ra, 11
Label54:
la $a0, Label55
jalr $ra, $a0
nop
Label55:
addi $s4, $ra, 22
mthi $t0
mfhi $s4
addi $s5, $s4, 33
mthi $t0
mfhi $s4
mult $t0, $s4
addi $s5, $s4, 44
sll $s4, $t0, 5
ori $s5, $s4, 321
sll $s4, $t0, 5
srl $s5, $t1, 4
ori $s5, $s4, 123

Load:
ori $t1, $0, 2
ori $t2, $0, 2
addu $t3, $t2, $t1
lw $t4, 0($t3)
ori $t2, $0, 4
addu $t3, $t2, $0
addu $s0, $s1, $s2
```

```
lw $t4, 0($t3)
ori $t2, $0, 4
lw $t5, 0($t2)
ori $t2, $0, 4
addu $s2, $s1, $s0
lw $t5, 0($t2)
ori $t3, $0, 8
sw $t3, 0($t3)
lw $t4, 0($t3)
lw $t5, 0($t4)
lw $t6, 0($t3)
addu $s1, $s0, $s2
lw $t5, 0($t6)
mthi $t3
mfhi $s4
lw $s5, 0($s4)
mthi $t3
mfhi $s5
div $t3, $s5
lw $s4, 0($s5)
sra $s4, $t3, 1
lw $s5, 0($s4)
sra $s5, $t3, 1
sll $t3, $t3, 1
lw $s4, 0($s5)
```

Store:

```
ori $t1, $0, 4
ori $t2, $0, 8
addu $t3, $t2, $t1
sw $t3, 0($t3)
ori $t2, $0, 48
addu $t3, $t2, $0
```

```
addu $s0, $s1, $s2
sw $t4, 0($t3)
ori $t2, $0, 40
sw $t5, 0($t2)
ori $t2, $0, 32
addu $s2, $s1, $s0
sw $t5, 0($t2)
ori $t3, $0, 80
sw $t3, 0($t3)
lw $t4, 0($t3)
sw $t5, 0($t4)
lw $t6, 0($t3)
addu $s1, $s0, $s2
sw $t5, 0($t6)
li $t3, 100
mthi $t3
mfhi $s4
sw $s5, 0($s4)
li $t3, 104
mthi $t3
mfhi $s4
mult $t3, $s4
sw $s5, 0($s4)
li $t3, 70
sll $s4, $t3, 2
sw $s5, 0($s4)
li $t3, 71
sll $s4, $t3, 2
sra $s5, $s4, 3
sw $s5, 0($s4)
ori $t1, $0, 4
ori $t2, $0, 8
addu $t3, $t2, $t1
```

```
sw $t3, 0($t3)
ori $t2, $0, 84
sw $t2, 0($t2)
ori $t3, $0, 8
lw $t4, 0($t3)
sw $t4, 4($t3)
jal Label56
sw $ra, 200($t3)
Label56:
la $t3, Label57
jalr $ra, $t3
sw $ra, 260($0)
Label57:
mfhi $t3
sw $t3, 264($0)
sll $t3, $t3, 2
sw $t3, 268($0)

Branch:
addu $t1, $t2, $t3
addu $t4, $t2, $t3
beq $t4, $t1, Label11
nop
addu $s1, $s2, $s3
Label11:
addu $t2, $t1, $t3
addu $t4, $t1, $t3
beq $t2, $t4, Label12
nop
addu $s1, $s2, $s3
Label12:
addu $t1, $t2, $t3
addu $t4, $t2, $t3
```

```
beq $t1, $t4, Label13
nop
addu $s1, $s2, $s3
Label13:
addu $t2, $t1, $t3
addu $t4, $t1, $t3
beq $t4, $t2, Label14
nop
addu $s1, $s2, $s3
Label14:
addu $t1, $t2, $t3
addu $t4, $t2, $t3
addu $s1, $s2, $s3
beq $t1, $t4, Label15
nop
addu $s1, $s2, $s3
Label15:
addu $t2, $t1, $t3
addu $t4, $t1, $t3
addu $s1, $s2, $s3
beq $t4, $t2, Label16
nop
addu $s1, $s2, $s3
Label16:
ori $t1, $0, 1
ori $t2, $0, 1
beq $t2, $t1, Label17
nop
addu $s1, $s2, $s3
Label17:
ori $t1, $0, 2
ori $t2, $0, 2
beq $t1, $t2, Label18
```

```
nop
addu $s1, $s2, $s3
Label18:
ori $t1, $0, 3
ori $t2, $0, 3
addu $s1, $s2, $s3
beq $t1, $t2, Label19
nop
addu $s1, $s2, $s3
Label19:
ori $t1, $0, 4
ori $t2, $0, 4
addu $s1, $s2, $s3
beq $t2, $t1, Label20
nop
addu $s1, $s2, $s3
Label20:
ori $t3, $0, 20
sw $s0, 0($t3)
lw $t2, 0($t3)
beq $t2, $s0, Label21
nop
addu $s1, $s2, $s3
Label21:
ori $t4, $0, 24
sw $s0, 0($t4)
lw $t1, 0($t4)
beq $s0, $t1, Label22
nop
addu $s1, $s2, $s3
Label22:
ori $t3, $0, 28
sw $s0, 0($t3)
```

```
lw $t2, 0($t3)
addu $s1, $s2, $s3
beq $t2, $s0, Label23
nop
addu $s1, $s2, $s3
Label23:
ori $t4, $0, 32
sw $s0, 0($t4)
lw $t1, 0($t4)
addu $s1, $s2, $s3
beq $s0, $t1, Label24
nop
addu $s1, $s2, $s3
Label24:
ori $t3, $0, 36
sw $s0, 0($t3)
lw $t2, 0($t3)
addu $s1, $s2, $s3
nop
beq $t2, $s0, Label25
nop
addu $s1, $s2, $s3
Label25:
ori $t4, $0, 44
sw $s0, 0($t4)
lw $t1, 0($t4)
addu $s1, $s2, $s3
nop
beq $s0, $t1, Label26
nop
addu $s1, $s2, $s3
Label26:
jal Label27
```

```
addu $t1, $0, $ra
Label27:
beq $ra, $t1, Label28
nop
addu $s1, $s2, $s3
Label28:
jal Label29
addu $t1, $0, $ra
Label29:
beq $t1, $ra, Label30
nop
addu $s1, $s2, $s3
Label30:
jal Label31
nop
Label31:
addu $t1, $0, $ra
beq $ra, $t1, Label32
nop
addu $s1, $s2, $s3
Label32:
jal Label33
nop
Label33:
addu $t1, $0, $ra
beq $t1, $ra, Label34
nop
addu $s1, $s2, $s3
Label34:
la $a0, Label58
jalr $ra, $a0
addu $t1, $0, $ra
Label58:
```

```
beq $ra, $t1, Label59
nop
addu $s1, $s2, $s3
Label59:
la $a0, Label60
jalr $ra, $a0
addu $t1, $0, $ra
Label60:
beq $t1, $ra, Label61
nop
addu $s1, $s2, $s3
Label61:
la $a0, Label62
jalr $ra, $a0
nop
Label62:
addu $t1, $0, $ra
beq $ra, $t1, Label63
nop
addu $s1, $s2, $s3
Label63:
la $a0, Label64
jalr $ra, $a0
nop
Label64:
addu $t1, $0, $ra
beq $t1, $ra, Label65
nop
addu $s1, $s2, $s3
Label65:
mthi $t0
mfhi $s0
beq $s0, $t0, Label66
```

```
nop
addu $s1, $s2, $s3
Label66:
mthi $t1
mfhi $s0
beq $t1, $s0, Label67
nop
addu $s1, $s2, $s3
Label67:
mthi $t0
mfhi $s0
nop
beq $s0, $t0, Label68
nop
addu $s1, $s2, $s3
Label68:
mthi $t1
mfhi $s0
nop
beq $t1, $s0, Label69
nop
addu $s1, $s2, $s3
Label69:
mthi $t0
mfhi $s0
nop
beq $s0, $t0, Label70
nop
addu $s1, $s2, $s3
Label70:
mthi $t1
mfhi $s0
nop
```

```
beq $t1, $s0, Label71
nop
addu $s1, $s2, $s3
Label71:
li $a0, 4
li $a1, 2
sll $a1, $a1, 1
beq $a1, $a0, Label72
nop
addu $s2, $s1, $s3
Label72:
li $a0, 8
li $a1, 16
sra $a1, $a1, 1
beq $a0, $a1, Label73
nop
addu $s1, $s2, $s3
Label73:
li $a0, 4
li $a1, 2
sll $a1, $a1, 1
nop
beq $a1, $a0, Label74
nop
addu $s2, $s1, $s3
Label74:
li $a0, 8
li $a1, 16
sra $a1, $a1, 1
nop
beq $a0, $a1, Label75
nop
addu $s1, $s2, $s3
```

```
Label75:
li $a0, 4
li $a1, 2
sll $a1, $a1, 1
nop
nop
beq $a1, $a0, Label76
nop
addu $s2, $s1, $s3
Label76:
li $a0, 8
li $a1, 16
sra $a1, $a1, 1
nop
nop
beq $a0, $a1, Label77
nop
addu $s1, $s2, $s3
Label77:

Jr:
jal Label35
ori $t2, $0, 12
Label35:
addu $t1, $t2, $ra
jr $t1
nop
jal Label36
ori $t2, $0, 16
Label36:
addu $t1, $t2, $ra
nop
jr $t1
```

```
nop
jal Label37
ori $t2, $0, 20
Label37:
addu $t1, $t2, $ra
nop
nop
jr $t1
nop
jal Label38
ori $t2, $0, 16
Label38:
addu $t1, $t2, $ra
ori $t4, $t1, 0
jr $t4
nop
jal Label39
ori $t2, $0, 20
Label39:
addu $t1, $t2, $ra
ori $t4, $t1, 0
nop
jr $t4
nop
jal Label40
ori $t2, $0, 24
Label40:
addu $t1, $t2, $ra
ori $t4, $t1, 0
nop
nop
jr $t4
nop
```

```
jal Label41
ori $t2, $0, 20
Label41:
addu $t1, $t2, $ra
sw $t1, 0($t2)
lw $t3, 0($t2)
jr $t3
nop
jal Label42
ori $t2, $0, 24
Label42:
addu $t1, $t2, $ra
sw $t1, 0($t2)
lw $t3, 0($t2)
nop
jr $t3
nop
jal Label43
ori $t2, $0, 28
Label43:
addu $t1, $t2, $ra
sw $t1, 0($t2)
lw $t3, 0($t2)
nop
nop
jr $t3
nop
jal Label44
nop
j Label45
nop
Label44:
jr $ra
```

```
nop
Label45:
jal Label47
nop
j Label48
nop
Label47:
addu $s1, $s2, $s3
jr $ra
nop
Label48:
la $a0, Label78
jalr $ra, $a0
nop
j Label79
nop
Label78:
jr $ra
nop
Label79:
la $a0, Label80
jalr $ra, $a0
nop
j Label81
nop
Label80:
addu $s2, $s1, $s3
jr $ra
nop
Label81:
la $a0, Label82
mthi $a0
mfhi $ra
```

```
jr $ra
nop
Label82:
la $a0, Label83
mthi $a0
mfhi $ra
sll $a0, $a0, 1
jr $ra
nop
Label83:
la $a0, Label84
mthi $a0
mfhi $ra
sll $a0, $a0, 1
nop
jr $ra
nop
Label84:

Shift:
addu $t0, $t1, $t2
sll $t0, $t0, 1
subu $t3, $t4, $t5
xori $s0, $s0, 0x1111
srl $t3, $t0, 1
addi $s0, $s0, 125
srl $s1, $s0, 2
addi $s0, $s0, 127
andi $s1, $s1, 0x1010
srl $s1, $s0, 2
lw $a0, 0($0)
sll $a0, $a0, 2
lw $a0, 4($0)
```

```
sll $a0, $a0, 3
jal Label85
sll $ra, $ra, 3
Label85:
jal Label86
nop
Label86:
sll $ra, $ra, 3
la $a0, Label87
jalr $ra, $a0
sll $ra, $ra, 3
Label87:
la $a0, Label88
jalr $ra, $a0
nop
Label88:
sll $ra, $ra, 2
mfhi $a0
sra $a0, $a0, 2
mfhi $a1
sll $a1, $a1, 4
srl $a1, $a1, 2
sll $a1, $a1, 2
sll $s0, $s0, 3
srav $a1, $a1, $a1
srl $s0, $s0, 4
```

2. 期望结果

@00003000: $ 8 <= 12340000

@00003004: $ 8 <= 00005678

@00003008: $16 <= 00000010

@0000300c: *00000000 <= 00007800

@00003010: $12 <= 00000078

@00003014: $13 <= 00000078

@00003018: *00000000 <= 00007878

@0000301c: $14 <= 00000078

@00003020: $15 <= 00000078

@00003024: *00000000 <= 56787878

@00003028: $12 <= 00005678

@0000302c: $13 <= 00005678

@00003030: *00000000 <= 56787878

@00003034: $14 <= 00005678

@00003038: $15 <= 00005678

@0000303c: *00000004 <= 00005678

@00003040: $ 9 <= 00005678

@00003044: *00000008 <= 00005678

@00003048: $10 <= 00005678

@0000304c: $11 <= 0000acf0

@00003050: $12 <= 00000000

@00003054: $15 <= 0000acf0

@00003058: $24 <= 00000000

@0000305c: $ 4 <= ffffffff

@00003064: $11 <= ffffffff

@0000306c: $12 <= ffffa988

@00003070: $ 4 <= fffffffd

@00003074: $ 5 <= 00000002

@0000307c: $17 <= ffffffff

@00003080: $18 <= ffffffff

@00003088: $17 <= 00000001

@0000308c: $18 <= 7ffffffe

@00003098: $19 <= 00005678

@0000309c: $20 <= 00005678

@000030a0: $17 <= 00000004

@000030a4: $18 <= 0fffffff

@000030a8: $ 4 <= ffffffff

@000030ac: $13 <= 00000004

@000030b0: $14 <= 00000003

@000030b4: $15 <= 00000002

@000030b8: $17 <= 00000040

@000030bc: $18 <= 01ffffff

@000030c0: $ 4 <= ffffffff

@000030c4: $19 <= 00000040

@000030c8: $19 <= 01ffffff

@000030cc: $19 <= 01ffffbf

@000030d0: $19 <= fe000000

@000030d4: $19 <= fdffffff

@000030d8: $19 <= 00001010

@000030dc: $19 <= 01ffefef

@000030e0: $20 <= 00000000

@000030e4: $20 <= 00000000

@000030e8: $20 <= 00000001

@000030ec: $ 4 <= ffffffff

@000030f0: $20 <= 00000001

@000030f8: $13 <= 00010000

@000030fc: $19 <= 00010000

@0000311c: $13 <= 00020000

@00003120: $19 <= 00020000

@0000313c: $21 <= ffffffff

@00003140: $22 <= 00000001

@00003148: $13 <= 00000001

@0000315c: $13 <= 00000002

@00003170: $13 <= 00000001

@00003184: $13 <= 00000002

@00003194: $31 <= 0000319c

@00003198: $20 <= 00400000

@000031ac: $24 <= 00000010

@000031b4: $22 <= 04000000

@0000319c: $25 <= 00003054

@000031a0: $ 4 <= 000031c4

@000031a4: $31 <= 000031ac

@000031a8: $21 <= 01000000

@000031c8: $18 <= 00000081

@000031b8: $23 <= 04030000

@000031cc: $10 <= 0000acf0

@000031d0: $11 <= 00005678

@000031d4: $10 <= 0000acf0

@000031d8: $11 <= ffffa988

@000031dc: $10 <= 0000acf0

@000031e0: $16 <= 0000001a

@000031e4: $11 <= 00005678

@000031e8: $10 <= 0000acf0

@000031ec: $16 <= 0000001b

@000031f0: $11 <= ffffa988

@000031f4: $10 <= 00810000

@000031f8: $11 <= 0080a988

@000031fc: $12 <= 00810000

@00003200: $11 <= ff7f5678

@00003204: $10 <= 007f0000

@00003208: $17 <= 00020081

@0000320c: $11 <= 007ea988

@00003210: $12 <= 007f0000

@00003214: $17 <= fffe0081

@00003218: $11 <= ff815678

@0000321c: $12 <= 56787878

@00003220: $11 <= 5678cef0

@00003224: $13 <= 00005678

@00003228: $11 <= 007f5678

@0000322c: $12 <= 56787878

@00003230: $17 <= fffe0081

@00003234: $11 <= 5678cef0

@00003238: $13 <= 00005678

@0000323c: $17 <= fffe0081

@00003240: $11 <= 007f5678

@00003244: $31 <= 0000324c

@00003248: $20 <= 0000324c

@0000324c: $31 <= 00003254

@00003250: $21 <= 00003254

@00003254: $31 <= 0000325c

@0000325c: $20 <= 0000325c

@00003260: $31 <= 00003268

@00003268: $21 <= 00003268

@0000326c: $ 4 <= 00003278

@00003270: $31 <= 00003278

@00003274: $20 <= 00003278

@00003278: $ 4 <= 00003284

@0000327c: $31 <= 00003284

@00003280: $21 <= 00003284

@00003284: $ 4 <= 00003290

@00003288: $31 <= 00003290

@00003290: $20 <= 00003290

@00003294: $ 4 <= 000032a0

@00003298: $31 <= 000032a0

@000032a0: $21 <= 000032a0

@000032a8: $20 <= 00005678

@000032ac: $21 <= 0000acf0

@000032b4: $21 <= 00005678

@000032b8: $20 <= 0000acf0

@000032c0: $20 <= 00005678

@000032c8: $21 <= 0000acf0

@000032d0: $21 <= 00005678

@000032d8: $20 <= 0000acf0

@000032dc: $20 <= 000acf00

@000032e0: $21 <= 000a7888

@000032e4: $20 <= 00056780

@000032e8: $21 <= fffaeef8

@000032ec: $20 <= 000acf00

@000032f0: $21 <= 00000567

@000032f4: $21 <= 000a7888

@000032f8: $20 <= 00056780

@000032fc: $21 <= 00000acf

@00003300: $21 <= fffaeef8

@00003304: $10 <= 0000acf0

@00003308: $11 <= 0000acff

@0000330c: $10 <= 00010377

@00003310: $16 <= 0000001b

@00003314: $11 <= 0001037f

@00003318: $10 <= 00810000

@0000331c: $11 <= 00810001

@00003320: $10 <= 007f0000

@00003324: $17 <= 00020081

@00003328: $11 <= 007f0006

@0000332c: $12 <= 56787878

@00003330: $11 <= 5678787a

@00003334: $13 <= 00005678

@00003338: $17 <= fffe0081

@0000333c: $11 <= 0000567d

@00003340: $31 <= 00003348

@00003344: $20 <= 0000334e

@00003348: $31 <= 00003350

@00003350: $20 <= 00003359

@00003354: $ 4 <= 00003360

@00003358: $31 <= 00003360

@0000335c: $20 <= 0000336b

@00003360: $ 4 <= 0000336c

@00003364: $31 <= 0000336c

@0000336c: $20 <= 00003382

@00003374: $20 <= 00005678

@00003378: $21 <= 00005699

@00003380: $20 <= 00005678

@00003388: $21 <= 000056a4

@0000338c: $20 <= 000acf00

@00003390: $21 <= 000acf41

@00003394: $20 <= 000acf00

@00003398: $21 <= 00000567

@0000339c: $21 <= 000acf7b

@000033a0: $ 9 <= 00000002

@000033a4: $10 <= 00000002

@000033a8: $11 <= 00000004

@000033ac: $12 <= 00005678

@000033b0: $10 <= 00000004

@000033b4: $11 <= 00000004

@000033b8: $16 <= fffe0102

@000033bc: $12 <= 00005678

@000033c0: $10 <= 00000004

@000033c4: $13 <= 00005678

@000033c8: $10 <= 00000004

@000033cc: $18 <= fffc0183

@000033d0: $13 <= 00005678

@000033d4: $11 <= 00000008

@000033d8: *00000008 <= 00000008

@000033dc: $12 <= 00000008

@000033e0: $13 <= 00000008

@000033e4: $14 <= 00000008

@000033e8: $17 <= fffa0285

@000033ec: $13 <= 00000008

@000033f4: $20 <= 00000008

@000033f8: $21 <= 00000008

@00003400: $21 <= 00000008

@00003408: $20 <= 00000008

@0000340c: $20 <= 00000004

@00003410: $21 <= 00005678

@00003414: $21 <= 00000004

@00003418: $11 <= 00000010

@0000341c: $20 <= 00005678

@00003420: $ 9 <= 00000004

@00003424: $10 <= 00000008

@00003428: $11 <= 0000000c

@0000342c: *0000000c <= 0000000c

@00003430: $10 <= 00000030

@00003434: $11 <= 00000030

@00003438: $16 <= fff60408

@0000343c: *00000030 <= 00000008

@00003440: $10 <= 00000028

@00003444: *00000028 <= 00000008

@00003448: $10 <= 00000020

@0000344c: $18 <= fff0068d

@00003450: *00000020 <= 00000008

@00003454: $11 <= 00000050

@00003458: *00000050 <= 00000050

@0000345c: $12 <= 00000050

@00003460: *00000050 <= 00000008

@00003464: $14 <= 00000008

@00003468: $17 <= ffe60a95

@0000346c: *00000008 <= 00000008

@00003470: $11 <= 00000064

@00003478: $20 <= 00000064

@0000347c: *00000064 <= 00000004

@00003480: $11 <= 00000068

@00003488: $20 <= 00000068

@00003490: *00000068 <= 00000004

@00003494: $11 <= 00000046

@00003498: $20 <= 00000118

@0000349c: *00000118 <= 00000004

@000034a0: $11 <= 00000047

@000034a4: $20 <= 0000011c

@000034a8: $21 <= 00000023

@000034ac: *0000011c <= 00000023

@000034b0: $ 9 <= 00000004

@000034b4: $10 <= 00000008

@000034b8: $11 <= 0000000c

@000034bc: *0000000c <= 0000000c

@000034c0: $10 <= 00000054

@000034c4: *00000054 <= 00000054

@000034c8: $11 <= 00000008

@000034cc: $12 <= 00000008

@000034d0: *0000000c <= 00000008

@000034d4: $31 <= 000034dc

@000034d8: *000000d0 <= 000034dc

@000034dc: $11 <= 000034e8

@000034e0: $31 <= 000034e8

@000034e4: *00000104 <= 000034e8

@000034e8: $11 <= 00000000

@000034ec: *00000108 <= 00000000

@000034f0: $11 <= 00000000

@000034f4: *0000010c <= 00000000

@000034f8: $ 9 <= 00000054

@000034fc: $12 <= 00000054

@0000350c: $10 <= 00000054

@00003510: $12 <= 00000054

@00003520: $ 9 <= 00000054

@00003524: $12 <= 00000054

@00003534: $10 <= 00000054

@00003538: $12 <= 00000054

@00003548: $ 9 <= 00000054

@0000354c: $12 <= 00000054

@00003550: $17 <= fff2068d

@00003560: $10 <= 00000054

@00003564: $12 <= 00000054

@00003568: $17 <= fff2068d

@00003578: $ 9 <= 00000001

@0000357c: $10 <= 00000001

@0000358c: $ 9 <= 00000002

@00003590: $10 <= 00000002

@000035a0: $ 9 <= 00000003

@000035a4: $10 <= 00000003

@000035a8: $17 <= fff2068d

@000035b8: $ 9 <= 00000004

@000035bc: $10 <= 00000004

@000035c0: $17 <= fff2068d

@000035d0: $11 <= 00000014

@000035d4: *00000014 <= fff60408

@000035d8: $10 <= fff60408

@000035e8: $12 <= 00000018

@000035ec: *00000018 <= fff60408

@000035f0: $ 9 <= fff60408

@00003600: $11 <= 0000001c

@00003604: *0000001c <= fff60408

@00003608: $10 <= fff60408

@0000360c: $17 <= fff2068d

@0000361c: $12 <= 00000020

@00003620: *00000020 <= fff60408

@00003624: $ 9 <= fff60408

@00003628: $17 <= fff2068d

@00003638: $11 <= 00000024

@0000363c: *00000024 <= fff60408

@00003640: $10 <= fff60408

@00003644: $17 <= fff2068d

@00003658: $12 <= 0000002c

@0000365c: *0000002c <= fff60408

@00003660: $ 9 <= fff60408

@00003664: $17 <= fff2068d

@00003678: $31 <= 00003680

@0000367c: $ 9 <= 00003680

@0000368c: $31 <= 00003694

@00003690: $ 9 <= 00003694

@000036a0: $31 <= 000036a8

@000036a8: $ 9 <= 000036a8

@000036b8: $31 <= 000036c0

@000036c0: $ 9 <= 000036c0

@000036d0: $ 4 <= 000036dc

@000036d4: $31 <= 000036dc

@000036d8: $ 9 <= 000036dc

@000036e8: $ 4 <= 000036f4

@000036ec: $31 <= 000036f4

@000036f0: $ 9 <= 000036f4

@00003700: $ 4 <= 0000370c

@00003704: $31 <= 0000370c

@0000370c: $ 9 <= 0000370c

@0000371c: $ 4 <= 00003728

@00003720: $31 <= 00003728

@00003728: $ 9 <= 00003728

@0000373c: $16 <= 00005678

@00003750: $16 <= 00003728

@00003764: $16 <= 00005678

@0000377c: $16 <= 00003728

@00003794: $16 <= 00005678

@000037ac: $16 <= 00003728

@000037c0: $ 4 <= 00000004

@000037c4: $ 5 <= 00000002

@000037c8: $ 5 <= 00000004

@000037d8: $ 4 <= 00000008

@000037dc: $ 5 <= 00000010

@000037e0: $ 5 <= 00000008

@000037f0: $ 4 <= 00000004

@000037f4: $ 5 <= 00000002

@000037f8: $ 5 <= 00000004

@0000380c: $ 4 <= 00000008

@00003810: $ 5 <= 00000010

@00003814: $ 5 <= 00000008

@00003828: $ 4 <= 00000004

@0000382c: $ 5 <= 00000002

@00003830: $ 5 <= 00000004

@00003848: $ 4 <= 00000008

@0000384c: $ 5 <= 00000010

@00003850: $ 5 <= 00000008

@00003868: $31 <= 00003870

@0000386c: $10 <= 0000000c

@00003870: $ 9 <= 0000387c

@0000387c: $31 <= 00003884

@00003880: $10 <= 00000010

@00003884: $ 9 <= 00003894

@00003894: $31 <= 0000389c

@00003898: $10 <= 00000014

@0000389c: $ 9 <= 000038b0

@000038b0: $31 <= 000038b8

@000038b4: $10 <= 00000010

@000038b8: $ 9 <= 000038c8

@000038bc: $12 <= 000038c8

@000038c8: $31 <= 000038d0

@000038cc: $10 <= 00000014

@000038d0: $ 9 <= 000038e4

@000038d4: $12 <= 000038e4

@000038e4: $31 <= 000038ec

@000038e8: $10 <= 00000018

@000038ec: $ 9 <= 00003904

@000038f0: $12 <= 00003904

@00003904: $31 <= 0000390c

@00003908: $10 <= 00000014

@0000390c: $ 9 <= 00003920

@00003910: *00000014 <= 00003920

@00003914: $11 <= 00003920

@00003920: $31 <= 00003928

@00003924: $10 <= 00000018

@00003928: $ 9 <= 00003940

@0000392c: *00000018 <= 00003940

@00003930: $11 <= 00003940

@00003940: $31 <= 00003948

@00003944: $10 <= 0000001c

@00003948: $ 9 <= 00003964

@0000394c: *0000001c <= 00003964

@00003950: $11 <= 00003964

@00003964: $31 <= 0000396c

@0000397c: $31 <= 00003984

@0000398c: $17 <= fff2068d

@00003998: $ 4 <= 000039ac

@0000399c: $31 <= 000039a4

@000039b4: $ 4 <= 000039c8

@000039b8: $31 <= 000039c0

@000039c8: $18 <= fff4068d

@000039d4: $ 4 <= 000039e8

@000039dc: $31 <= 000039e8

@000039e8: $ 4 <= 00003a00

@000039f0: $31 <= 00003a00

@000039f4: $ 4 <= 00007400

@00003a00: $ 4 <= 00003a1c

@00003a08: $31 <= 00003a1c

@00003a0c: $ 4 <= 00007438

@00003a1c: $ 8 <= 00003980

@00003a20: $ 8 <= 00007300

@00003a24: $11 <= 000038fc

@00003a28: $16 <= 00002639

@00003a2c: $11 <= 00003980

@00003a30: $16 <= 000026b6

@00003a34: $17 <= 000009ad

@00003a38: $16 <= 00002735

@00003a3c: $17 <= 00000000

@00003a40: $17 <= 000009cd

@00003a44: $ 4 <= 56787878

@00003a48: $ 4 <= 59e1e1e0

@00003a4c: $ 4 <= 00005678

@00003a50: $ 4 <= 0002b3c0

@00003a54: $31 <= 00003a5c

@00003a58: $31 <= 0001d2e0

@00003a5c: $31 <= 00003a64

@00003a64: $31 <= 0001d320

@00003a68: $ 4 <= 00003a74

@00003a6c: $31 <= 00003a74

@00003a70: $31 <= 0001d3a0

@00003a74: $ 4 <= 00003a80

@00003a78: $31 <= 00003a80

@00003a80: $31 <= 0000ea00

@00003a84: $ 4 <= 00003a1c

@00003a88: $ 4 <= 00000e87

@00003a8c: $ 5 <= 00003a1c

@00003a90: $ 5 <= 0003a1c0

@00003a94: $ 5 <= 0000e870

@00003a98: $ 5 <= 0003a1c0

@00003a9c: $16 <= 000139a8

@00003aa0: $ 5 <= 0003a1c0

@00003aa4: $16 <= 0000139a

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？

   乘除法计算的延迟较高，为防止其成为关键路径延长整个 CPU 的时钟周期，需要有单独的乘除法部件。由于乘除法部件的计算并不在一个时钟周期内完成，无法通过流水线寄存器传递，需要有独立的 HI、LO 寄存器。

2. 参照你对延迟槽的理解，试解释“乘除槽”。

   “乘除槽”就是指乘除指令后续的几条指令的位置里只能执行与乘除法和 HI、LO 寄存器无关的其他指令，否则只能暂停。

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）

   C 语言中 char 数组每个元素位宽为 1 个字节，此时按字节访问内存性能上更有优势。

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？

   设计风格可以概括为侦测者（Detector）型。命名规范化，使用宏定义，模块化和层次化设计，高内聚低耦合的设计原则等。

5. 你对流水线 CPU 设计风格有何见解？

   我认为应当采用 Detector 型这种易于快速添加指令的 CPU 设计，以便快速通过课上测试。

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？

cal_r 型指令：

表 24 cal_r 型指令冲突分析

用例编号	测试类型	解决方法	测试序列
1	cal_r-M-rs	F	addu $t2, $t1, $t0 subu $t3, $t2, $t1
2	cal_r-M-rt	F	addu $t2, $t1, $t0 subu $t3, $t1, $t2

3	cal_r-W-rs	F	addu $t2, $t1, $t0
			ori $s0, $s0, 10
			subu $t3, $t2, $t1

4	cal_r-W-rt	F	addu $t2, $t1, $t0
			ori $s0, $s0, 1
			subu $t3, $t1, $t2

5	cal_i-M-rs	F	lui $t2, 129 subu $t3, $t2, $t1
6	cal_i-M-rt	F	lui $t4, 129 subu $t3, $t1, $t4

7	cal_i-W-rs	F	lui $t2, 127
			addu $s1, $s2, $s3
			subu $t3, $t2, $t1

8	cal_i-W-rt	F	lui $t4, 127
			subu $s1, $s2, $s3
			subu $t3, $t1, $t4

9	load-M-rs	S	lw $t4, 0($0) addu $t3, $t4, $t1
10	load-M-rt	S	lw $t5,4($0) addu $t3, $t2, $t5

11	load-W-rs	， F	lw $t4, 0($0)
			subu $s1, $s2, $s3
			addu $t3, $t4, $t1

12	load-W-rt	F	lw $t5,4($0)
			subu $s1, $s2, $s3
			addu $t3, $t2, $t5

13	jal-M-rs	F	jal Label1 addu $s4, $ra, $0 Label1:
14	jal-M-rt	F	jal Label2 addu $s5, $0, $ra Label2:
15	jal-W-rs	F	jal Label3 nop Label3: addu $s4, $ra, $0
16	jal-W-rt	F	jal Label4 nop Label4: addu $s5, $0, $ra
17	jalr-M-rs	F	la $a0, Label50 jalr $ra, $a0 addu $s4, $ra, $0 Label50:
18	jalr-M-rt	F	la $a0, Label51 jalr $ra, $a0 addu $s5, $0, $ra Label51:
19	jalr-W-rs	F	la $a0, Label52 jalr $ra, $a0 nop Label52: addu $s4, $ra, $0
20	jalr-W-rt	F	la $a0, Label53 jalr $ra, $a0 nop Label53: addu $s5, $0, $ra
21	mfhl-M-rs	F	mthi $t0 mfhi $s4 addu $s5, $s4, $t0

22	mfhl-M-rt	F	mtlo $t1 mflo $s5 addu $s4, $t0, $s5

23	mfhl-W-rs	F	mthi $t0
			mfhi $s4
			mult $t0, $s4
			addu $s5, $s4, $t0

24	mfhl-W-rt	F	mtlo $t1
			mflo $s5
			mult $t1, $s5
			addu $s4, $t0, $s5

25	shift-M-rs	F	sll $s4, $t0, 5 subu $s5, $s4, $t1
26	shift-M-rt	F	sll $s4, $t1, 4 subu $s5, $t0, $s4
27	shift-W-rs	F	sll $s4, $t0, 5 srl $s5, $t1, 4 subu $s5, $s4, $t1
28	shift-W-rt	F	sll $s4, $t1, 4 sra $s5, $t0, 3 subu $s5, $t0, $s4

cal_i 型指令：

表 25 cal_i 型指令冲突分析

用例编号	测试类型	解决方法	测试序列
1	cal_r-M-rs	F	addu $t2, $t1, $t0 ori $t3, $t2, 31
2	cal_r-W-rs	F	addu $t2, $t1, $t3 ori $s0, $s0, 10 ori $t3, $t2, 127
3	cal_i-M-rs	F	lui $t2, 129 ori $t3, $t2, 1
4	cal_i-W-rs	F	lui $t2, 127 addu $s1, $s2, $s3 ori $t3, $t2, 6
5	load-M-rs	S	lw $t4, 0($0) ori $t3, $t4, 98

6	load-W-rs	F	lw $t5, 4($0)
			subu $s1, $s2, $s3
			ori $t3, $t5, 101

7	jal-M-rs	F	jal Label5 ori $s4, $ra, 6 Label5:
8	jal-W-rs	F	jal Label6 nop Label6: ori $s4, $ra, 9

9	jalr-M-rs	F	la $a0, Label54 jalr $ra, $a0 addi $s4, $ra, 11 Label54:
10	jalr-W-rs	F	la $a0, Label55 jalr $ra, $a0 nop Label55: addi $s4, $ra, 22
11	mfhl-M-rs	F	mthi $t0 mfhi $s4 addi $s5, $s4, 33

12	mfhl-W-rs	F	mthi $t0
			mfhi $s4
			mult $t0, $s4
			addi $s5, $s4, 44

13	shift-M-rs	F	sll $s4, $t0, 5 ori $s5, $s4, 321
14	shift-W-rs	F	sll $s4, $t0, 5 srl $s5, $t1, 4 ori $s5, $s4, 123

load 型指令：

表 26 load 型指令冲突分析

用例编号	测试类型	解决方法	测试序列

1	cal_r-M-rs	F	ori $t1, $0, 2
			ori $t2, $0, 2
			addu $t3, $t2, $t1
			lw $t4, 0($t3)

2	cal_r-W-rs	F	ori $t2, $0, 4
			addu $t3, $t2, $0
			addu $s0, $s1, $s2
			lw $t4, 0($t3)

3	cal_i-M-rs	F	ori $t2, $0, 4 lw $t5, 0($t2)

4	cal_i-W-rs	F	ori $t2, $0, 4
			addu $s2, $s1, $s0
			lw $t5, 0($t2)

5	load-M-rs	S	ori $t3, $0, 8
			sw $t3, 0($t3)
			lw $t4, 0($t3)
			lw $t5, 0($t4)

6	load-W-rs	F	lw $t6, 0($t3)
			addu $s1, $s0, $s2
			lw $t5, 0($t6)

7	mfhl-M-rs	F	mthi $t3 mfhi $s4 lw $s5, 0($s4)

8	mfhl-W-rs	F	mthi $t3 mfhi $s5 div $t3, $s5 lw $s4, 0($s5)
9	shift-M-rs	F	sra $s4, $t3, 1 lw $s5, 0($s4)
10	shift-W-rs	F	sra $s5, $t3, 1 sll $t3, $t3, 1 lw $s4, 0($s5)

store 型指令：

表 27 store 型指令冲突分析

用例编号	测试类型	解决方法	测试序列

1	cal_r-M-rs	F	ori $t1, $0, 4
			ori $t2, $0, 8
			addu $t3, $t2, $t1
			sw $t3, 0($t3)

2	cal_r-W-rs	F	ori $t2, $0, 48
			addu $t3, $t2, $0
			addu $s0, $s1, $s2
			sw $t4, 0($t3)

3	cal_i-M-rs	F	ori $t2, $0, 40 sw $t5, 0($t2)

4	cal_i-W-rs	F	ori $t2, $0, 32
			addu $s2, $s1, $s0
			sw $t5, 0($t2)

5	load-M-rs	S	ori $t3, $0, 80
			sw $t3, 0($t3)
			lw $t4, 0($t3)
			sw $t5, 0($t4)

6	load-W-rs	F	lw $t6, 0($t3)
			addu $s1, $s0, $s2
			sw $t5, 0($t6)

7	mfhl-M-rs	F	li $t3, 100 mthi $t3 mfhi $s4 sw $s5, 0($s4)

8	mfhl-W-rs	F	li $t3, 104
			mthi $t3
			mfhi $s4
			mult $t3, $s4
			sw $s5, 0($s4)

9	shift-M-rs	F	li $t3, 70 sll $s4, $t3, 2 sw $s5, 0($s4)
10	shift-W-rs	F	li $t3, 71 sll $s4, $t3, 2 sra $s5, $s4, 3 sw $s5, 0($s4)

11	cal_r-W-rt	F	ori $t1, $0, 4
			ori $t2, $0, 8
			addu $t3, $t2, $t1
			sw $t3, 0($t3)

12	cal_i-W-rt	F	ori $t2, $0, 84 sw $t2, 0($t2)

13	load-W-rt	F	ori $t3, $0, 8
			lw $t4, 0($t3)
			sw $t4, 4($t3)

14	jal-W-rt	F	jal Label56 sw $ra, 200($t3) Label56:
15	jalr-W-rt	F	la $t3, Label57 jalr $ra, $t3 sw $ra, 260($0) Label57:
16	mfhl-W-rt	F	mfhi $t3 sw $t3, 264($0)
17	shift-W-rt	F	sll $t3, $t3, 2 sw $t3, 268($0)

branch 型指令：

表 28 branch 型指令冲突分析

用例编号	测试类型	解决方法	测试序列

1	cal_r-E-rs	S	addu $t1, $t2, $t3
			addu $t4, $t2, $t3
			beq $t4, $t1, Label11
			nop
			addu $s1, $s2, $s3
			Label11:

2	cal_r-E-rt	S	addu $t2, $t1, $t3
			addu $t4, $t1, $t3
			beq $t2, $t4, Label12
			nop
			addu $s1, $s2, $s3
			Label12:

3	cal_r-M-rs	F	addu $t1, $t2, $t3
			addu $t4, $t2, $t3
			beq $t1, $t4, Label13
			nop
			addu $s1, $s2, $s3
			Label13:

4	cal_r-M-rt	F	addu $t2, $t1, $t3
			addu $t4, $t1, $t3
			beq $t4, $t2, Label14
			nop
			addu $s1, $s2, $s3
			Label14:

5	cal_r-W-rs	F	addu $t1, $t2, $t3
			addu $t4, $t2, $t3
			addu $s1, $s2, $s3
			beq $t1, $t4, Label15
			nop
			addu $s1, $s2, $s3
			Label15:

6	cal_r-W-rt	F	addu $t2, $t1, $t3
			addu $t4, $t1, $t3
			addu $s1, $s2, $s3
			beq $t4, $t2, Label16
			nop
			addu $s1, $s2, $s3
			Label16:

7	cal_i-E-rs	S	ori $t1, $0, 1 ori $t2, $0, 1 beq $t2, $t1, Label17 nop addu $s1, $s2, $s3 Label17:
8	cal_i-E-rt	S	ori $t1, $0, 2 ori $t2, $0, 2 beq $t1, $t2, Label18 nop addu $s1, $s2, $s3 Label18:
9	cal_i-M-rs	F	同 8
10	cal_i-M-rt	F	同 7
11	cal_i-W-rs	F	ori $t1, $0, 3 ori $t2, $0, 3 addu $s1, $s2, $s3 beq $t1, $t2, Label19 nop addu $s1, $s2, $s3 Label19:
12	cal_i-W-rt	F	ori $t1, $0, 4 ori $t2, $0, 4 addu $s1, $s2, $s3 beq $t2, $t1, Label20 nop addu $s1, $s2, $s3 Label20:

13	load-E-rs	S	ori $t3, $0, 20
			sw $s0, 0($t3)
			lw $t2, 0($t3)
			beq $t2, $s0, Label21
			nop
			addu $s1, $s2, $s3
			Label21:

14	load-E-rt	S	ori $t4, $0, 24
			sw $s0, 0($t4)
			lw $t1, 0($t4)
			beq $s0, $t1, Label22
			nop
			addu $s1, $s2, $s3
			Label22:

15	load-M-rs	F	ori $t3, $0, 28
			sw $s0, 0($t3)
			lw $t2, 0($t3)
			addu $s1, $s2, $s3
			beq $t2, $s0, Label23
			nop
			addu $s1, $s2, $s3
			Label23:

16	load-M-rt	S	ori $t4, $0, 32
			sw $s0, 0($t4)
			lw $t1, 0($t4)
			addu $s1, $s2, $s3
			beq $s0, $t1, Label24
			nop
			addu $s1, $s2, $s3
			Label24:

17	load-W-rs	F	ori $t3, $0, 36
			sw $s0, 0($t3)
			lw $t2, 0($t3)
			addu $s1, $s2, $s3
			nop
			beq $t2, $s0, Label25
			nop
			addu $s1, $s2, $s3
			Label25:

18	load-W-rt	F	ori $t4, $0, 44
			sw $s0, 0($t4)
			lw $t1, 0($t4)
			addu $s1, $s2, $s3
			nop
			beq $s0, $t1, Label26
			nop
			addu $s1, $s2, $s3
			Label26:

19	jal-M-rs	F	jal Label27
			addu $t1, $0, $ra
			Label27:
			beq $ra, $t1, Label28
			nop
			addu $s1, $s2, $s3
			Label28:

20	jal-M-rt	F	jal Label29
			addu $t1, $0, $ra
			Label29:
			beq $t1, $ra, Label30
			nop
			addu $s1, $s2, $s3
			Label30:

21	jal-W-rs	F	jal Label31
			nop
			Label31:
			addu $t1, $0, $ra
			beq $ra, $t1, Label32
			nop
			addu $s1, $s2, $s3
			Label32:

22	jal-W-rt	F	jal Label33
			nop
			Label33:
			addu $t1, $0, $ra
			beq $t1, $ra, Label34
			nop
			addu $s1, $s2, $s3
			Label34:

23	jalr-M-rs	F	la $a0, Label58
			jalr $ra, $a0
			addu $t1, $0, $ra
			Label58:
			beq $ra, $t1, Label59
			nop
			addu $s1, $s2, $s3
			Label59:

24	jalr-M-rt	F	la $a0, Label60
			jalr $ra, $a0
			addu $t1, $0, $ra
			Label60:
			beq $t1, $ra, Label61
			nop
			addu $s1, $s2, $s3
			Label61:

25	jalr-W-rs	F	la $a0, Label62
			jalr $ra, $a0
			nop
			Label62:
			addu $t1, $0, $ra
			beq $ra, $t1, Label63
			nop
			addu $s1, $s2, $s3
			Label63:

26	jalr-W-rt	F	la $a0, Label64
			jalr $ra, $a0
			nop
			Label64:
			addu $t1, $0, $ra
			beq $t1, $ra, Label65
			nop
			addu $s1, $s2, $s3
			Label65:

27	mfhl-E-rs	S	mthi $t0 mfhi $s0 beq $s0, $t0, Label66 nop addu $s1, $s2, $s3 Label66:
28	mfhl-E-rt	S	mthi $t1 mfhi $s0 beq $t1, $s0, Label67 nop addu $s1, $s2, $s3 Label67:
29	mfhl-M-rs	F	mthi $t0 mfhi $s0 nop beq $s0, $t0, Label68 nop addu $s1, $s2, $s3 Label68:
30	mfhl-M-rt	F	mthi $t1 mfhi $s0 nop beq $t1, $s0, Label69 nop addu $s1, $s2, $s3 Label69:
31	mfhl-W-rs	F	mthi $t0 mfhi $s0 nop beq $s0, $t0, Label70 nop addu $s1, $s2, $s3 Label70:
32	mfhl-W-rt	F	mthi $t1 mfhi $s0 nop beq $t1, $s0, Label71 nop addu $s1, $s2, $s3 Label71:

33	shift-E-rs	S	li $a0, 4 li $a1, 2 sll $a1, $a1, 1 beq $a1, $a0, Label72 nop addu $s2, $s1, $s3 Label72:
34	shift-E-rt	S	li $a0, 8 li $a1, 16 sra $a1, $a1, 1 beq $a0, $a1, Label73 nop addu $s1, $s2, $s3 Label73:
35	shift-M-rs	F	li $a0, 4 li $a1, 2 sll $a1, $a1, 1 nop beq $a1, $a0, Label74 nop addu $s2, $s1, $s3 Label74:
36	shift-M-rt	F	li $a0, 8 li $a1, 16 sra $a1, $a1, 1 nop beq $a0, $a1, Label75 nop addu $s1, $s2, $s3 Label75:
37	shift-W-rs	F	li $a0, 4 li $a1, 2 sll $a1, $a1, 1 nop nop beq $a1, $a0, Label76 nop addu $s2, $s1, $s3 Label76:
38	shift-W-rt	F	li $a0, 8 li $a1, 16 sra $a1, $a1, 1 nop nop beq $a0, $a1, Label77 nop addu $s1, $s2, $s3 Label77:

jr 型指令：

表 29 jr 型指令冲突分析

用例编号	测试类型	解决方法	测试序列
1	cal_r-E-rs	S	jal Label35 ori $t2, $0, 12 Label35: addu $t1, $t2, $ra jr $t1 nop
2	cal_r-M-rs	F	jal Label36 ori $t2, $0, 16 Label36: addu $t1, $t2, $ra nop jr $t1 nop
3	cal_r-W-rs	F	jal Label37 ori $t2, $0, 20 Label37: addu $t1, $t2, $ra nop nop jr $t1 nop
4	cal_i-E-rs	S	jal Label38 ori $t2, $0, 16 Label38: addu $t1, $t2, $ra ori $t4, $t1, 0 jr $t4 nop
5	cal_i-M-rs	F	jal Label39 ori $t2, $0, 20 Label39: addu $t1, $t2, $ra ori $t4, $t1, 0 nop jr $t4 nop
6	cal_i-W-rs	F	jal Label40 ori $t2, $0, 24 Label40: addu $t1, $t2, $ra ori $t4, $t1, 0 nop nop jr $t4 nop

7	load-E-rs	S	jal Label41
			ori $t2, $0, 20
			Label41:
			addu $t1, $t2, $ra
			sw $t1, 0($t2)
			lw $t3, 0($t2)
			jr $t3
			nop

8	load-M-rs	F	jal Label42
			ori $t2, $0, 24
			Label42:
			addu $t1, $t2, $ra
			sw $t1, 0($t2)
			lw $t3, 0($t2)
			nop
			jr $t3
			nop

9	load-W-rs	F	jal Label43
			ori $t2, $0, 28
			Label43:
			addu $t1, $t2, $ra
			sw $t1, 0($t2)
			lw $t3, 0($t2)
			nop
			nop
			jr $t3
			nop

10	jal-M-rs	F	jal Label44 nop j Label45 nop Label44: jr $ra nop Label45:
11	jal-W-rs	F	jal Label47 nop j Label48 nop Label47: addu $s1, $s2, $s3 jr $ra nop Label48:

12	jalr-M-rs	F	la $a0, Label78 jalr $ra, $a0 nop j Label79 nop Label78: jr $ra nop Label79:
13	jalr-W-rs	F	la $a0, Label80 jalr $ra, $a0 nop j Label81 nop Label80: addu $s2, $s1, $s3 jr $ra nop Label81:
14	mfhl-E-rs	S	la $a0, Label82 mthi $a0 mfhi $ra jr $ra nop Label82:
15	mfhl-M-rs	F	la $a0, Label83 mthi $a0 mfhi $ra sll $a0, $a0, 1 jr $ra nop Label83:
16	mfhl-W-rs	F	la $a0, Label84 mthi $a0 mfhi $ra sll $a0, $a0, 1 nop jr $ra nop Label84:

jalr 型指令：冲突同 jr 型指令。

md 型指令：冲突同 cal_r 型指令。

mthl 型指令：冲突同 cal_i 型指令。

shift 型指令：

表 30 shift 型指令冲突分析

用例编号	测试类型	解决方法	测试序列

1	cal_r-M-rs	F	addu $t0, $t1, $t2 sll $t0, $t0, 1
2	cal_r-W-rs	F	subu $t3, $t4, $t5 xori $s0, $s0, 0x1111 srl $t3, $t0, 1
3	cal_i-M-rs	F	addi $s0, $s0, 125 srl $s1, $s0, 2
4	cal_i-W-rs	F	addi $s0, $s0, 127 andi $s1, $s1, 0x1010 srl $s1, $s0, 2
5	load-M-rs	S	lw $a0, 0($0) sll $a0, $a0, 2
6	load-W-rs	F	lw $a0, 4($0) sll $a0, $a0, 3
7	jal-M-rs	F	jal Label85 sll $ra, $ra, 3 Label85:
8	jal-W-rs	F	jal Label86 nop Label86: sll $ra, $ra, 3
9	jalr-M-rs	F	la $a0, Label87 jalr $ra, $a0 sll $ra, $ra, 3 Label87:
10	jalr-W-rs	F	la $a0, Label88 jalr $ra, $a0 nop Label88: sll $ra, $ra, 2
11	mfhl-M-rs	F	mfhi $a0 sra $a0, $a0, 2
12	mfhl-W-rs	F	mfhi $a1 sll $a1, $a1, 4
13	shift-M-rs	F	srl $a1, $a1, 2 sll $a1, $a1, 2
14	shift-W-rs	F	sll $s0, $s0, 3 srav $a1, $a1, $a1 srl $s0, $s0, 4