MIPS单周期处理器设计文档

1 除控制器与冲突处理单元外的模块设计

1.1 IFU（取指令单元）

1.1.1 PC（程序计数器）

表1.1 PC信号说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | PC\_input [31:0] | I | 在时钟上升沿到来时向PC中写入程序要执行的下一条指令的地址 |
| 2 | clk | I | 时钟信号，在上升沿到来时触发PC中的寄存器 |
| 3 | reset | I | 同步复位信号，高电平有效 |
| 4 | en | I | 使能信号，高电平有效 |
| 5 | PC\_output [31:0] | O | 输出从IM要取得的指令的地址 |

1.1.2 IM（指令存储器）

表1.2 IM信号说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | PC [31:0] | I | 将PC的输出信号PC\_output作为要读取的指令的地址输入到IM |
| 2 | IR [31:0] | O | 输出要当前时钟周期要执行的指令 |

1.1.3 NPC

表1.3 NPC信号说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | PC4 [31:0] | I | 输入当前PC+4的值 |
| 2 | I26 [25:0] | I | 输入当前指令的低26位 |
| 3 | isBranch | I | 输入isBranch信号，判断当前指令是否为branch类指令 |
| 4 | NPC\_output [31:0] | O | 输出NPC的值作为PC的下一个选择之一 |

1.2 GRF（通用寄存器组）

表1.4 GRF信号说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | A1 [4:0] | I | 根据输入的5位地址指定32个寄存器中的一个，读取其存储的值并输出到RD1 |
| 2 | A2 [4:0] | I | 根据输入的5位地址指定32个寄存器中的一个，读取其存储的值并输出到RD2 |
| 3 | A3 [4:0] | I | 根据输入的5位地址指定32个寄存器中的一个，将WD输入的32位数值写入指定的寄存器 |
| 4 | WD [31:0] | I | 输入要写入GRF的数值 |
| 5 | clk | I | 时钟信号，时钟上升沿触发 |
| 6 | reset | I | 同步复位信号，高电平有效 |
| 7 | WE | I | 写使能信号，当其为1时将WD输入的数值写入指定寄存器，否则不写入GRF |
| 8 | RD1 | O | 输出A1的地址指向的寄存器存储的值 |
| 9 | RD2 | O | 输出A2的地址指向的寄存器存储的值 |

1.3 ALU（算术逻辑单元）

表1.5 ALU信号说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | A [31:0] | I | 输入传入ALU的第一个32位数A |
| 2 | B [31:0] | I | 输入传入ALU的第一个32位数B |
| 3 | ALUOp [3:0] | I | 输入两位二进制数，用于控制ALU进行的运算类型，输入为00时执行A+B，输入为01时执行A-B，输入为10时执行A|B |
| 4 | Shamt [4:0] | I | 指令的shamt部分，用于移位相关指令 |
| 5 | Result [31:0] | O | 输出ALU的运算结果 |

1.4 DM（数据存储器）

表1.6 DM信号说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | Addr [31:0] | I | 输入32位地址，此地址指向DM中的要读取或写入数据的位置 |
| 2 | WD [31:0] | I | 输入要写入DM的32位数据 |
| 3 | WE | I | 写使能信号，如果输入为1，则向DM中写入WD输入的数据，否则不写入 |
| 4 | reset | I | 同步复位，高电平有效 |
| 5 | clk | I | 时钟信号，时钟上升沿触发 |
| 6 | BEOp [1:0] | I | 字节使能的判断信号，用于判断存储指令的指令类型。 |
| 7 | BE [3:0] | I | 字节使能，在写入信号有效时会对字节使能有效的字节进行写入。 |
| 8 | RD [31:0] | O | 输出从DM中读取的数据 |

1.5 BE\_Control（字节使能产生模块）

表1.7 BE\_Control信号说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | Addr [31:0] | I | 输入32位地址，此地址指向DM中的要读取或写入数据的位置 |
| 2 | BEOp [1:0] | I | 由控制器产生，用于识别不同类型的存储指令 |
| 3 | BE [3:0] | O | 输出字节使能信号 |

1.6 DM\_EXT（DM读取数据扩展模块）

表1.8 DM\_EXT信号说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | Addr [31:0] | I | 输入32位地址，此地址指向DM中的要读取或写入数据的位置 |
| 2 | Din [31:0] | I | 输入从DM中读出的待扩展的数值 |
| 3 | LOp [1:0] | I | 由控制器产生，用于识别读取指令的指令类型 |
| 4 | DMEXTOp | I | 用于区分符号扩展与无符号扩展 |
| 5 | Dout [31:0] | O | 输出扩展后的结果 |

1.6 EXT

表1.9 EXT信号说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | I16 [15:0] | I | 输入16位数据 |
| 2 | EXTOp [1:0] | I | 如果输入为00，进行符号扩展；如果输入为01，进行无符号扩展；如果输入为10，将输入的数据加载到32位中的高16位，同时在低16位补0 |
| 3 | EXT\_output [31:0] | O | 输出扩展后的数据 |

1.7 CMP

表1.10 CMP信号说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | D1 [31:0] | I | 输入要比较的第一个数 |
| 2 | D2 [31:0] | I | 输入要比较的第二个数 |
| 3 | isEqual | O | 如果两个数相等则输出1，不相等则输出0 |
| 4 | isEqual0 | O | 如果D1=0则输出1，不相等则输出0 |
| 5 | Great0 | O | 如果D1>0则输出1，不相等则输出0 |
| 6 | Less0 | O | 如果D1<0则输出1，不相等则输出0 |

1.8 流水线寄存器

1.8.1 FD\_reg（IF/ID级流水线寄存器）

表1.11 FD\_reg信号说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | IR\_F [31:0] | I | 输入IF阶段的指令 |
| 2 | PC4\_F [31:0] | I | 输入IF阶段PC+4的值 |
| 3 | PC8\_F [31:0] | I | 输入IF阶段PC+8的值 |
| 4 | clk | I | 时钟信号 |
| 5 | en | I | 使能信号，高电平有效 |
| 6 | reset | I | 同步复位信号 |
| 7 | IR\_D [31:0] | O | 输出ID阶段的指令 |
| 8 | PC4\_D [31:0] | O | 输出ID阶段PC+4的值 |
| 9 | PC8\_D [31:0] | O | 输出ID阶段PC+8的值 |

1.8.2 DE\_reg（ID/EX级流水线寄存器）

表1.12 DE\_reg信号说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | IR\_D [31:0] | I | 输入ID阶段的指令 |
| 2 | RS\_D [31:0] | I | 输入ID阶段从寄存器堆读出的GPR[rs]的值 |
| 3 | RT\_D [31:0] | I | 输入ID阶段从寄存器堆读出的GPR[rt]的值 |
| 4 | PC8\_D [31:0] | I | 输入ID阶段PC+8的值 |
| 5 | EXT\_D [31:0] | I | 输入ID阶段EXT部件输出的值 |
| 6 | clk | I | 时钟信号 |
| 7 | reset | I | 同步复位信号 |
| 8 | IR\_D [31:0] | O | 输出EX阶段的指令 |
| 9 | RS\_D [31:0] | O | 输出EX阶段GPR[rs]的值 |
| 10 | RT\_D [31:0] | O | 输出EX阶段GPR[rt]的值 |
| 11 | PC8\_D [31:0] | O | 输出EX阶段PC+8的值 |
| 12 | EXT\_D [31:0] | O | 输出EX阶段EXT部件输出的值 |

1.8.3 EM\_reg（EX/MEM级流水线寄存器）

表1.13 EM\_reg信号说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | IR\_E [31:0] | I | 输入EX阶段的指令 |
| 2 | AO\_E [31:0] | I | 输入EX阶段ALU的计算结果 |
| 3 | RT\_E [31:0] | I | 输入EX阶段的GPR[rt]的值 |
| 4 | PC8\_E [31:0] | I | 输入EX阶段PC+8的值 |
| 5 | clk | I | 时钟信号 |
| 6 | reset | I | 同步复位信号 |
| 7 | IR\_M[31:0] | O | 输出MEM阶段的指令 |
| 8 | AO\_M [31:0] | O | 输出MEM阶段GPR[rs]的值 |
| 9 | RT\_M [31:0] | O | 输出MEM阶段GPR[rt]的值 |
| 10 | PC8\_M [31:0] | O | 输出MEM阶段PC+8的值 |

1.8.4 MW\_reg（MEM/WB级流水线寄存器）

表1.14 MW\_reg信号说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | IR\_M [31:0] | I | 输入MEM阶段的指令 |
| 2 | AO\_M [31:0] | I | 输入MEM阶段ALU的计算结果 |
| 3 | DM\_M [31:0] | I | 输入MEM阶段从DM读取的值 |
| 4 | PC8\_M [31:0] | I | 输入MEM阶段PC+8的值 |
| 5 | clk | I | 时钟信号 |
| 6 | reset | I | 同步复位信号 |
| 7 | IR\_W [31:0] | O | 输出WB阶段的指令 |
| 8 | AO\_W [31:0] | O | 输出WB阶段ALU的结果 |
| 9 | DM\_W [31:0] | O | 输出WB阶段DM的值 |
| 10 | PC8\_W [31:0] | O | 输出WB阶段PC+8的值 |

（与转发相关的元件请见“P6\_相关表格.xlsx”）

1.9 MD（乘除模块）

表1.15 MD信号说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | D1 [31:0] | I | 输入的第一个数 |
| 2 | D2 [31:0] | I | 输入的第二个数 |
| 3 | Use\_MD | I | 用于判断是否要使用MD模块 |
| 4 | Start | I | 乘除法运算触发信号 |
| 5 | CW [1:0] | I | 用于对乘除模块内的功能进行选择 |
| 6 | isSigned\_MD | I | 判断是否进行有符号运算 |
| 7 | Clk | I | 时钟信号 |
| 8 | Reset | I | 复位信号 |
| 9 | Busy | O | 输出当前是否在进行乘除运算 |
| 10 | HI\_Output | O | 输出HI寄存器的值 |
| 11 | LO\_Output | O | 输出LO寄存器的值 |

2 控制器设计

2.1 主控单元设计

（主控单元真值表请见“P6\_相关表格.xlsx”）

2.2 EXT控制单元设计

2.2.1 输入

由主控单元产生的2位EXPOp信号和指令中包含的16位立即数。

2.2.2 EXT控制单元设计

表2.4 EXT控制信号说明

|  |  |  |
| --- | --- | --- |
| 序号 | EXTOp | 扩展方式 |
| 1 | 00 | 将立即数符号扩展为32位数 |
| 2 | 01 | 将立即数无符号扩展为32位数 |
| 3 | 10 | 将立即数加载到32位中的高16位，同时在低16位补零生成32位数 |

2.3 MD模块CW控制信号

表2.5 CW控制信号说明

|  |  |  |
| --- | --- | --- |
| 序号 | CW | 扩展方式 |
| 1 | 00 | 进行乘法运算 |
| 2 | 01 | 进行除法运算 |
| 3 | 10 | 写入HI寄存器 |
| 4 | 11 | 写入LO寄存器 |

3 冲突处理单元设计

冲突处理单元的设计请见附表“P6\_相关表格.xlsx”。

4 测试程序

测试程序分为两部分，一部分是针对于冲突的测试，另一部分是完整情况的测试，关于冲突的测试请见附表“P6\_相关表格.xlsx”，完整的测试如下。

4.1 基本情况测试1(ori lui addu subu sw lw beq j jal jalr jr)

ori $a0 $0 123

ori $a1 $a0 456

lui $a2 123

lui $a3 0xffff

ori $a3 $a3 0xffff

addu $s0 $a0 $a2

addu $s1 $a0 $a3

addu $s2 $a3 $a3

subu $s0 $a0 $a2

subu $s1 $a0 $a3

subu $s2 $a3 $a3

ori $t0 $0 0x0000

sw $a0 0($t0)

sw $a1 4($t0)

sw $a2 8($t0)

sw $a3 12($t0)

sw $s0 16($t0)

sw $s1 20($t0)

sw $s2 24($t0)

lw $a0 0($t0)

lw $a1 12($t0)

sw $a0 28($t0)

sw $a1 32($t0)

ori $a0 $0 1

ori $a1 $0 2

ori $a2 $0 1

beq $a0 $a1 mark1

nop

beq $a0 $a2 mark2

nop

mark1: sw $a0 36($t0)

mark2: sw $a1 40($t0)

jal mark3

nop

ori $s2 $0 2

jal mark4

nop

lui $s7 0xabcd

j end

mark3:

ori $s1 $0 1

jalr $t9 $ra

nop

mark4:

jr $ra

end:

nop

实际的测试期望为：

@00003000: $ 4 <= 0000007b

@00003004: $ 5 <= 000001fb

@00003008: $ 6 <= 007b0000

@0000300c: $ 7 <= ffff0000

@00003010: $ 7 <= ffffffff

@00003014: $16 <= 007b007b

@00003018: $17 <= 0000007a

@0000301c: $18 <= fffffffe

@00003020: $16 <= ff85007b

@00003024: $17 <= 0000007c

@00003028: $18 <= 00000000

@0000302c: $ 8 <= 00000000

@00003030: \*00000000 <= 0000007b

@00003034: \*00000004 <= 000001fb

@00003038: \*00000008 <= 007b0000

@0000303c: \*0000000c <= ffffffff

@00003040: \*00000010 <= ff85007b

@00003044: \*00000014 <= 0000007c

@00003048: \*00000018 <= 00000000

@0000304c: $ 4 <= 0000007b

@00003050: $ 5 <= ffffffff

@00003054: \*0000001c <= 0000007b

@00003058: \*00000020 <= ffffffff

@0000305c: $ 4 <= 00000001

@00003060: $ 5 <= 00000002

@00003064: $ 6 <= 00000001

@0000307c: \*00000028 <= 00000002

@00003080: $31 <= 00003088

@0000309c: $17 <= 00000001

@000030a0: $25 <= 000030a8

@00003088: $18 <= 00000002

@0000308c: $31 <= 00003094

@00003094: $23 <= abcd0000

@0000309c: $17 <= 00000001

4.2 基本情况测试2(beq bne bgtz bgez bltz blez)

ori $t0 $0 0

ori $t1 $0 1

lui $t2 0xffff

ori $t2 $t2 0xffff

bgez $t2 mark1

ori $s0 $0 1

bgez $t0 mark1

nop

ori $s0 $0 2

mark1:

blez $t1 mark2

ori $s1 $0 1

blez $t2 mark2

nop

ori $s1 $0 2

mark2:

bgtz $t0 mark3

ori $s2 $0 1

bgtz $t1 mark3

nop

ori $s2 $0 2

mark3:

bltz $t0 mark4

ori $s3 $0 1

bltz $t2 mark4

nop

ori $s3 $0 2

mark4:

测试期望：

@00003000: $ 8 <= 00000000

@00003004: $ 9 <= 00000001

@00003008: $10 <= ffff0000

@0000300c: $10 <= ffffffff

@00003014: $16 <= 00000001

@00003028: $17 <= 00000001

@0000303c: $18 <= 00000001

@00003050: $19 <= 00000001

4.3 基本情况测试3(beq bne bgtz bgez bltz blez offset为负)

ori $t0 $0 0x000a

ori $t1 $0 0x0009

mark1:

addi $t1 $t1 1

beq $t0 $t1 mark1

nop

ori $t1 $0 0x0008

mark2:

addi $t1 $t1 1

bne $t0 $t1 mark2

nop

lui $t1 0xffff

ori $t1 $t1 0xfffe

mark3:

addi $t1 $t1 1

bltz $t1 mark3

nop

lui $t1 0xffff

ori $t1 $t1 0xfffe

mark4:

addi $t1 $t1 1

blez $t1 mark4

nop

ori $t1 $0 2

mark5:

addi $t1 $t1 -1

bgtz $t1 mark5

nop

ori $t1 $0 2

mark6:

addi $t1 $t1 -1

bgez $t1 mark6

nop

测试期望：

@00003000: $ 8 <= 0000000a

@00003004: $ 9 <= 00000009

@00003008: $ 9 <= 0000000a

@00003008: $ 9 <= 0000000b

@00003014: $ 9 <= 00000008

@00003018: $ 9 <= 00000009

@00003018: $ 9 <= 0000000a

@00003024: $ 9 <= ffff0000

@00003028: $ 9 <= fffffffe

@0000302c: $ 9 <= ffffffff

@0000302c: $ 9 <= 00000000

@00003038: $ 9 <= ffff0000

@0000303c: $ 9 <= fffffffe

@00003040: $ 9 <= ffffffff

@00003040: $ 9 <= 00000000

@00003040: $ 9 <= 00000001

@0000304c: $ 9 <= 00000002

@00003050: $ 9 <= 00000001

@00003050: $ 9 <= 00000000

@0000305c: $ 9 <= 00000002

@00003060: $ 9 <= 00000001

@00003060: $ 9 <= 00000000

@00003060: $ 9 <= ffffffff

4.4 基本情况测试4(logic shift compute)

ori $t0 $0 12

andi $t1 $t0 10

or $s0 $t0 $t1

and $s1 $s0 $t1

nor $s2 $s0 $s1

xori $s3 $s1 9

xor $s4 $s2 $s3

sll $s5 $t0 2

ori $t2 $0 3

sllv $s5 $s3 $t2

srl $s5 $s5 2

ori $t2 $0 1

srlv $s5 $s3 $t2

lui $t9 0xffff

ori $t9 $t9 0x0010

sra $t8 $t9 4

ori $t2 $0 16

srav $t7 $t9 $t2

add $t6 $t0 $t1

addi $t6 $t6 1

addiu $t6 $t6 1

sub $t6 $t6 $t7

subu $t6 $t6 1

测试期望：期望输出少了且给错了

@00003000: $ 9 <= 00000000

@00003004: $16 <= 00000000

@00003008: $17 <= 00000000

@0000300c: $18 <= ffffffff

@00003010: $19 <= 00000009

@00003014: $20 <= fffffff6

@00003018: $21 <= 00000000

@0000301c: $10 <= 00000003

@00003020: $21 <= 00000048

@00003024: $21 <= 00000012

@00003028: $10 <= 00000001

@0000302c: $21 <= 00000004

@00003030: $25 <= ffff0000

@00003034: $25 <= ffff0010

@00003038: $24 <= fffff001

@0000303c: $10 <= 00000010

@00003040: $15 <= ffffffff

@00003044: $14 <= 00000000

@00003048: $14 <= 00000001

@0000304c: $14 <= 00000002

@00003050: $14 <= 00000003

@00003054: $ 1 <= 00000000

@00003058: $ 1 <= 00000001

@0000305c: $14 <= 00000002

4.5 基本情况测试 5(mult-div module)

ori $t0 $0 2

ori $t1 $0 -2

mult $t0 $t1

mflo $t2

mult $t2 $t2

div $t2 $t2

mflo $t3

ori $s0 $0 -9

ori $s1 $0 2

div $s0 $s1

mflo $s6

mfhi $s7

divu $s0 $s1

mflo $s0

sb $s0 3($0)

ori $t9 $0 0xabcd

sh $t9 0($0)

lhu $s0 0($0)

lh $s0 0($0)

mthi $s0

测试期望：

@00003000: $ 8 <= 00000002

@00003004: $ 1 <= ffff0000

@00003008: $ 1 <= fffffffe

@0000300c: $ 9 <= fffffffe

@00003014: $10 <= fffffffc

@00003020: $11 <= 00000001

@00003024: $ 1 <= ffff0000

@00003028: $ 1 <= fffffff7

@0000302c: $16 <= fffffff7

@00003030: $17 <= 00000002

@00003038: $22 <= fffffffc

@0000303c: $23 <= ffffffff

@00003044: $16 <= 7ffffffb

@00003048: \*00000003 <= fb000000

@0000304c: $25 <= 0000abcd

@00003050: \*00000000 <= fb00abcd

@00003054: $16 <= 0000abcd

@00003058: $16 <= ffffabcd

4.6 基本情况测试 6(mult-div module 2)

ori $t0 $0 -6

ori $t1 $0 -2

ori $t2 $0 3

ori $t3 $0 -4

ori $t4 $0 5

div $t0 $t1

mfhi $s6

mflo $s7

div $t0 $t2

mfhi $s6

mflo $s7

div $t0 $t3

mfhi $s6

mflo $s7

div $t0 $t4

mfhi $s6

mflo $s7

mult $t0 $t1

mflo $s6

mult $t0 $t2

mflo $s6

mfhi $s7

测试期望：

@00003000: $ 1 <= ffff0000

@00003004: $ 1 <= fffffffa

@00003008: $ 8 <= fffffffa

@0000300c: $ 1 <= ffff0000

@00003010: $ 1 <= fffffffe

@00003014: $ 9 <= fffffffe

@00003018: $10 <= 00000003

@0000301c: $ 1 <= ffff0000

@00003020: $ 1 <= fffffffc

@00003024: $11 <= fffffffc

@00003028: $12 <= 00000005

@00003030: $22 <= 00000000

@00003034: $23 <= 00000003

@0000303c: $22 <= 00000000

@00003040: $23 <= fffffffe

@00003048: $22 <= fffffffe

@0000304c: $23 <= 00000001

@00003054: $22 <= ffffffff

@00003058: $23 <= ffffffff

@00003060: $22 <= 0000000c

@00003068: $22 <= ffffffee

@0000306c: $23 <= ffffffff

4.7 基本情况测试 7（load/store）

lui $t0 0xabcd

ori $t0 0x1234

sw $t0 0($0)

lh $t1 2($0)

lhu $t2 2($0)

lb $t2 0($0)

lbu $t2 0($0)

lb $t3 1($0)

lw $t4 0($0)

lui $s0 0xcdef

ori $s0 0x2345

sh $s0 4($0)

sb $s0 6($0)

sb $s0 7($0)

测试期望：

@00003000: $ 8 <= abcd0000

@00003004: $ 8 <= abcd1234

@00003008: \*00000000 <= abcd1234

@0000300c: $ 9 <= ffffabcd

@00003010: $10 <= 0000abcd

@00003014: $10 <= 00000034

@00003018: $10 <= 00000034

@0000301c: $11 <= 00000012

@00003020: $12 <= abcd1234

@00003024: $16 <= cdef0000

@00003028: $16 <= cdef2345

@0000302c: \*00000004 <= 00002345

@00003030: \*00000004 <= 00452345

@00003034: \*00000004 <= 45452345

4.8 综合测试 1

#test compute (add...)

ori $t0 $0 2

ori $t1 $0 8

add $t2 $t0 $t1

addi $t2 $t2 -3

addu $t2 $t2 $t2

addi $t2 $t2 4

addiu $t2 $t2 -9

addiu $t2 $t2 3

sub $t3 $t2 $t1

subu $t3 $t3 $t2

sub $t4 $t3 $t2

#test logic(or...)

or $t5 $t4 $t3

ori $t2 $t5 0xabcd

and $t5 $t5 $t2

andi $t3 $t2 0xa123

xor $t6 $t4 $t5

xori $t7 $t6 0x0a0b

lui $t8 0xfbfb

ori $t8 0xeffe

nor $t8 $t8 $t7

#test shift(sll...)

sll $t0 $t8 4

sllv $t9 $t0 $t1

srl $t1 $t9 2

ori $t2 $0 4

srlv $t1 $t1 $t2

sra $t3 $t4 1

lui $t3 0xf000

srav $t3 $t3 $t2

#test set bit(slt...)

ori $t0 $0 1

ori $t1 $0 2

lui $t2 0xffff

ori $t2 0xffff

slt $s0 $t0 $t1

slti $s1 $t1 -9

slt $s2 $t1 $t2

sltu $s3 $t1 $t2

slt $s4 $s2 $s3

sltu $s5 $s4 $s3

sltiu $s6 $s5 -12

addi $t3 $t2 -3

sltu $s7 $t2 $t3

slti $s0 $t2 4

sltiu $s1 $t2 4

#test branch(beq...)

ori $t0 $0 1

ori $t1 $0 1

lui $t2 0xffff

ori $t2 $t2 0xffff

beq $t0 $t1 mark1\_1

addi $t0 $t0 -1

mark1\_2:

lui $s0 0xabcd

mark1\_1:

bne $t0 $t2 mark1\_2

addi $t0 $t0 -1

beq $t0 $t2 mark1\_1

addi $t0 $t0 1

bltz $t0 mark2\_1

addi $t2 $t2 -8

ori $s3 $0 0xaaaa

mark2\_1:

lui $t3 0xffff

ori $t3 0xfffc

mark2\_2:

blez $t3 mark2\_2

addi $t3 $t3 1

bgez $t3 mark3

nop

mark3:

addiu $t3 $t3 -1

bgez $t3 mark3

ori $s7 0xabcd

addi $t3 $t3 9

bgtz $t3 mark4

mark4:

addiu $t3 $t3 -3

bgtz $t3 mark4

nop

#test jump(j...)

j jump1

ori $t0 $0 1

lui $s0 0xffff

jump1:

ori $t1 $0 2

sllv $t2 $t0 $t1

jal jump2

lui $t3 0xffff

ori $s0 0xaaaa

ori $s1 0xbbbb

jalr $t7 $t8

ori $s2 0xcccc

jump2:

ori $t3 $t3 0xffff

or $t9 $ra $0

jalr $t8 $t9

addu $t9 $t9 4

jal jump3

nop

j end\_jump

nop

jump3:

jr $ra

addi $t0 $ra 4

end\_jump:

#test mult\_div

ori $t0 $0 4

ori $t1 $0 6

lui $t2 0xffff

ori $t2 0xffff

addi $t3 $t2 -2

mult $t0 $t1

mult $t1 $t2

mflo $s0

mfhi $s1

div $t0 $t1

div $t1 $t0

mflo $s2

mfhi $s3

mtlo $s3

mthi $s2

mflo $s4

mult $t0 $t3

mflo $s4

mult $t2 $t3

mflo $s4

mult $t3 $t1

mflo $s4

div $t1 $t3

mflo $s5

mfhi $s6

div $t3 $t1

mflo $s5

mfhi $s6

ori $t0 $0 12

lui $t1 0xffff

mthi $t0

mtlo $t1

mflo $s0

mfhi $s1

测试期望：

@00003000: $ 8 <= 00000002

@00003004: $ 9 <= 00000008

@00003008: $10 <= 0000000a

@0000300c: $10 <= 00000007

@00003010: $10 <= 0000000e

@00003014: $10 <= 00000012

@00003018: $10 <= 00000009

@0000301c: $10 <= 0000000c

@00003020: $11 <= 00000004

@00003024: $11 <= fffffff8

@00003028: $12 <= ffffffec

@0000302c: $13 <= fffffffc

@00003030: $10 <= fffffffd

@00003034: $13 <= fffffffc

@00003038: $11 <= 0000a121

@0000303c: $14 <= 00000010

@00003040: $15 <= 00000a1b

@00003044: $24 <= fbfb0000

@00003048: $24 <= fbfbeffe

@0000304c: $24 <= 04041000

@00003050: $ 8 <= 40410000

@00003054: $25 <= 41000000

@00003058: $ 9 <= 10400000

@0000305c: $10 <= 00000004

@00003060: $ 9 <= 01040000

@00003064: $11 <= fffffff6

@00003068: $11 <= f0000000

@0000306c: $11 <= ff000000

@00003070: $ 8 <= 00000001

@00003074: $ 9 <= 00000002

@00003078: $10 <= ffff0000

@0000307c: $10 <= ffffffff

@00003080: $16 <= 00000001

@00003084: $17 <= 00000000

@00003088: $18 <= 00000000

@0000308c: $19 <= 00000001

@00003090: $20 <= 00000001

@00003094: $21 <= 00000000

@00003098: $22 <= 00000001

@0000309c: $11 <= fffffffc

@000030a0: $23 <= 00000000

@000030a4: $16 <= 00000001

@000030a8: $17 <= 00000000

@000030ac: $ 8 <= 00000001

@000030b0: $ 9 <= 00000001

@000030b4: $10 <= ffff0000

@000030b8: $10 <= ffffffff

@000030c0: $ 8 <= 00000000

@000030cc: $ 8 <= ffffffff

@000030c4: $16 <= abcd0000

@000030cc: $ 8 <= fffffffe

@000030d4: $ 8 <= ffffffff

@000030dc: $10 <= fffffff7

@000030e4: $11 <= ffff0000

@000030e8: $11 <= fffffffc

@000030f0: $11 <= fffffffd

@000030f0: $11 <= fffffffe

@000030f0: $11 <= ffffffff

@000030f0: $11 <= 00000000

@000030f0: $11 <= 00000001

@000030f0: $11 <= 00000002

@000030fc: $11 <= 00000001

@00003104: $23 <= 0000abcd

@000030fc: $11 <= 00000000

@00003104: $23 <= 0000abcd

@000030fc: $11 <= ffffffff

@00003104: $23 <= 0000abcd

@00003108: $11 <= 00000008

@00003110: $11 <= 00000005

@00003110: $11 <= 00000002

@00003110: $11 <= ffffffff

@00003120: $ 8 <= 00000001

@00003128: $ 9 <= 00000002

@0000312c: $10 <= 00000004

@00003130: $31 <= 00003138

@00003134: $11 <= ffff0000

@00003148: $11 <= ffffffff

@0000314c: $25 <= 00003138

@00003150: $24 <= 00003158

@00003154: $ 1 <= 00000000

@00003138: $16 <= abcdaaaa

@0000313c: $17 <= 0000bbbb

@00003140: $15 <= 00003148

@00003144: $18 <= 0000cccc

@00003158: $ 1 <= 00000004

@0000315c: $25 <= 0000313c

@00003160: $31 <= 00003168

@00003174: $ 8 <= 0000316c

@00003178: $ 8 <= 00000004

@0000317c: $ 9 <= 00000006

@00003180: $10 <= ffff0000

@00003184: $10 <= ffffffff

@00003188: $11 <= fffffffd

@00003194: $16 <= fffffffa

@00003198: $17 <= ffffffff

@000031a4: $18 <= 00000001

@000031a8: $19 <= 00000002

@000031b4: $20 <= 00000002

@000031bc: $20 <= fffffff4

@000031c4: $20 <= 00000003

@000031cc: $20 <= ffffffee

@000031d4: $21 <= fffffffe

@000031d8: $22 <= 00000000

@000031e0: $21 <= 00000000

@000031e4: $22 <= fffffffd

@000031e8: $ 8 <= 0000000c

@000031ec: $ 9 <= ffff0000

@000031f8: $16 <= ffff0000

@000031fc: $17 <= 0000000c

4.9 综合测试 2

#mix mult-div load-store branch jump

ori $t0 $0 1

ori $t1 $0 2

lui $t2 0xffff

ori $t2 $0 0xffff

mult $t0 $t1

mflo $t3

beq $t3 $t1 mark1

lui $s0 0xaaaa

mark2:

jal mark3

lui $s2 0xcccc

mflo $s6

mfhi $s7

jr $t8

lui $t8 0xabcd

mark1:

div $t1 $t2

mflo $t3

bne $t3 $t0 mark2

lui $s1 0xbbbb

mark3:

ori $v0 0x0011

lui $v1 0xffff

ori $v1 $v1 0xffff

or $t9 $ra $0

jalr $t8 $t9

multu $v0 $v1

mult $s0 $s1

mflo $s6

mfhi $s7

multu $s0 $s1

mflo $s6

mfhi $s7

mtlo $s7

mthi $s6

bgtz $v0 mark4

div $v0 $v1

ori $t4 $0 0xcccc

mark4:

mflo $s6

mfhi $s7

mark5:

addi $s6 $s6 1

bltz $s6 mark5

lui $t9 0xffff

blez $s6 mark5

addiu $s6 $s6 0xffff

jal mark6

multu $s6 $s7

lui $s3 0xbcda

lui $s3 0x4321

mfhi $s4

mflo $s5

mthi $s5

mtlo $s4

addiu $t6 $t6 -4

ori $s6 $s6 0xabcd

lui $s7 0x1a2b

jr $t6

nop

mark6:

or $t7 $ra $0

mfhi $t5

mflo $t4

addiu $t7 $t7 4

jalr $t6 $t7

div $t6 $t7

mfhi $t0

mflo $t1

bgez $t0 mark7

mtlo $t0

mark7b:

mfhi $s4

mflo $s5

sw $s4 4($0)

sw $s5 8($0)

lh $s5 8($0)

lhu $s6 8($0)

lb $s4 5($0)

lbu $s3 6($0)

mult $s3 $s6

mfhi $t0

mflo $t1

j mark8

nop

mark7:

mflo $t1

sb $t1 0($0)

sh $t1 2($0)

sh $t2 2($0)

sb $t1 3($0)

lw $t2 0($0)

divu $t2 $t1

bne $t1 $t2 mark7b

nop

mark8:

lui $t0 0xa1b2

ori $t0 $t0 0x3c4d

sw $t0 12($0)

lb $t1 12($0)

lbu $t2 14($0)

lh $t3 14($0)

lhu $t4 14($0)

bgez $t4 mark9

nop

mark9b:

mfhi $t8

mflo $t9

jal mark10

addi $ra $ra 4

jr $s6

nop

mark9:

bltz $t3 mark9b

mult $t1 $t4

mfhi $v0

mflo $v1

mark10:

addiu $ra $ra -4

or $s7 $ra $0

jalr $s6 $s7

addiu $s6 $s6 4

lui $s5 0xaaaa

lui $s4 0xbbbb

测试期望：

@00003000: $ 8 <= 00000001

@00003004: $ 9 <= 00000002

@00003008: $10 <= ffff0000

@0000300c: $10 <= 0000ffff

@00003014: $11 <= 00000002

@0000301c: $16 <= aaaa0000

@0000303c: $11 <= 00000000

@00003044: $17 <= bbbb0000

@00003020: $31 <= 00003028

@00003024: $18 <= cccc0000

@00003048: $ 2 <= 00000011

@0000304c: $ 3 <= ffff0000

@00003050: $ 3 <= ffffffff

@00003054: $25 <= 00003028

@00003058: $24 <= 00003060

@00003028: $22 <= ffffffef

@0000302c: $23 <= 00000010

@00003034: $24 <= abcd0000

@00003064: $22 <= 00000000

@00003068: $23 <= 16c1d82e

@00003070: $22 <= 00000000

@00003074: $23 <= 7d26d82e

@0000308c: $22 <= ffffffef

@00003090: $23 <= 00000000

@00003094: $22 <= fffffff0

@0000309c: $25 <= ffff0000

@00003094: $22 <= fffffff1

@0000309c: $25 <= ffff0000

@00003094: $22 <= fffffff2

@0000309c: $25 <= ffff0000

@00003094: $22 <= fffffff3

@0000309c: $25 <= ffff0000

@00003094: $22 <= fffffff4

@0000309c: $25 <= ffff0000

@00003094: $22 <= fffffff5

@0000309c: $25 <= ffff0000

@00003094: $22 <= fffffff6

@0000309c: $25 <= ffff0000

@00003094: $22 <= fffffff7

@0000309c: $25 <= ffff0000

@00003094: $22 <= fffffff8

@0000309c: $25 <= ffff0000

@00003094: $22 <= fffffff9

@0000309c: $25 <= ffff0000

@00003094: $22 <= fffffffa

@0000309c: $25 <= ffff0000

@00003094: $22 <= fffffffb

@0000309c: $25 <= ffff0000

@00003094: $22 <= fffffffc

@0000309c: $25 <= ffff0000

@00003094: $22 <= fffffffd

@0000309c: $25 <= ffff0000

@00003094: $22 <= fffffffe

@0000309c: $25 <= ffff0000

@00003094: $22 <= ffffffff

@0000309c: $25 <= ffff0000

@00003094: $22 <= 00000000

@0000309c: $25 <= ffff0000

@000030a4: $ 1 <= 00000000

@00003094: $22 <= 00000001

@0000309c: $25 <= ffff0000

@000030a4: $ 1 <= 00000000

@000030a8: $ 1 <= 0000ffff

@000030ac: $22 <= 00010000

@000030b0: $31 <= 000030b8

@000030e4: $15 <= 000030b8

@000030e8: $13 <= 00000000

@000030ec: $12 <= 00000000

@000030f0: $15 <= 000030bc

@000030f4: $14 <= 000030fc

@000030bc: $19 <= 43210000

@000030c0: $20 <= 00000040

@000030c4: $21 <= 00000001

@000030d0: $14 <= 000030f8

@000030d4: $22 <= 0001abcd

@000030d8: $23 <= 1a2b0000

@000030fc: $ 8 <= 0000003c

@00003100: $ 9 <= 00000001

@00003140: $ 9 <= 0000003c

@00003144: \*00000000 <= 0000003c

@00003148: \*00000000 <= 003c003c

@0000314c: \*00000000 <= ffff003c

@00003150: \*00000000 <= 3cff003c

@00003154: $10 <= 3cff003c

@0000310c: $20 <= 00000000

@00003110: $21 <= 01044001

@00003114: \*00000004 <= 00000000

@00003118: \*00000008 <= 01044001

@0000311c: $21 <= 00004001

@00003120: $22 <= 00004001

@00003124: $20 <= 00000000

@00003128: $19 <= 00000000

@00003130: $ 8 <= 00000000

@00003134: $ 9 <= 00000000

@00003164: $ 8 <= a1b20000

@00003168: $ 8 <= a1b23c4d

@0000316c: \*0000000c <= a1b23c4d

@00003170: $ 9 <= 0000004d

@00003174: $10 <= 000000b2

@00003178: $11 <= ffffa1b2

@0000317c: $12 <= 0000a1b2

@00003188: $24 <= 00000000

@0000318c: $25 <= 0030a28a

@00003190: $31 <= 00003198

@00003194: $31 <= 0000319c

@000031b0: $31 <= 00003198

@000031b4: $23 <= 00003198

@000031b8: $22 <= 000031c0

@000031bc: $22 <= 000031c4

@000031c4: $20 <= bbbb0000

4.10 综合测试 3

# basic compute

ori $t0 $0 1

ori $t1 $0 2

addu $t2 $t0 $t1

add $t3 $t2 $t1

lui $t4 0xffff

ori $t4 $t4 0xfff0

addu $t5 $t3 $t4

add $t6 $t5 $t4

lui $t0 0x7fff

ori $t0 $t0 0x1234

addiu $t0 $t0 0xffff

addi $t0 $t0 0xffff

add $t7 $t0 $t6

lui $t8 0xaaaa

addu $t9 $t7 $t8

addiu $t9 $t9 0xffffaaaa

addiu $t9 $t9 -1

addi $t9 $t9 -1

subu $s0 $t9 $t7

subu $s1 $t7 $t9

subu $s2 $s1 $s0

lui $t0 0xffff

ori $t0 $t0 0xfff0

ori $t1 $0 10

sub $s3 $t1 $t0

sub $s4 $t0 $t1

sub $s3 $s4 $s3

# load store

addi $s7 $0 8

addi $t4 $t4 -1

sw $t4 -8($s7)

lb $t0 -8($s7)

addi $t5 $t5 2

sb $t5 -4($s7)

lbu $t1 -7($s7)

lh $t2 -4($s7)

subu $t6 $t6 $s7

sb $t6 -3($s7)

lhu $t3 -4($s7)

sh $t7 -2($s7)

lw $t0 -4($s7)

ori $s0 $s0 1

sw $s0 0($s7)

ori $s1 $s1 2

sh $s1 4($s7)

lw $t1 0($s7)

addi $s2 $s2 -2

sb $s2 6($s7)

lh $t3 6($s7)

sub $s3 $s3 $s7

sb $s3 7($s7)

lbu $t0 7($s7)

# logic

lui $t0 0xabcd

lui $t1 0xcdef

ori $t0 $t0 0x1234

ori $t1 $t1 0x5678

and $t2 $t0 $t1

or $t3 $t2 $t1

and $t4 $t3 $t2

or $t5 $t4 $t3

andi $t6 $t5 0xa8b6

ori $t7 $t6 0x12b4

xor $t8 $t7 $t6

xor $t9 $t8 $t7

xori $t0 $t9 0x7823

andi $t1 $t0 0xffff

nor $t2 $t1 $t0

nor $t3 $t2 $t1

xori $t4 $t3 0xabcd

nor $t5 $t4 $t3

#branch

ori $t0 $0 1

ori $t1 $0 2

lui $t2 0xffff

ori $t2 $t2 0xffff

sub $t2 $t2 $t0

sub $t2 $t2 $t0

bne $t0 $t1 mark1

lui $t4 0xffff

mark1\_back:

ori $s0 $0 1

beq $t3 $t4 mark2

addi $t3 $t3 1

mark2\_back:

bltz $t2 mark3

addi $t2 $t2 1

mark3\_back:

addi $t2 $t2 1

blez $t2 mark4

subu $t2 $t2 $t1

mark4\_back:

bgtz $t2 mark5

add $t2 $t2 $t1

mark5\_back:

bgez $t2 mark6

addi $t2 $t2 1

mark6\_back:

j end

nop

mark1:

ori $s0 $0 2

bne $t4 $0 mark1\_back

lui $t3 0xffff

mark2:

addi $t4 $t4 1

beq $t3 $t4 mark2\_back

nop

mark3:

sub $t2 $t2 $t0

bltz $t2 mark3\_back

nop

mark4:

blez $t2 mark4\_back

addi $t2 $t2 112

mark5:

bgtz $t2 mark5\_back

sub $t2 $t2 $t1

mark6:

bgez $0 mark6\_back

end:

#shift

lui $t0 0xfff0

ori $t0 $t0 0xabcd

ori $t1 $0 3

sll $t2 $t0 2

sllv $t2 $t2 $t1

srl $t2 $t2 3

addi $t1 $t1 -1

srlv $t2 $t2 $t1

addi $t1 $t1 3

srlv $t2 $t2 $t1

lui $t3 0x8000

or $t2 $t3 $t2

sra $t2 $t2 4

addi $t1 $t1 -3

srav $t2 $t2 $t1

lui $t3 0x7fff

ori $t3 $t3 0xffff

and $t2 $t2 $t3

sra $t2 $t2 2

srav $t2 $t2 $t1

#jump

j mark7

ori $s0 $0 0xffff

mark7\_back:

jal mark8

nop

ori $s0 $0 0xaaaa

addiu $s3 $s3 8

jr $s3

nop

ori $s0 $0 0xbbbb

j mark9

nop

mark7:

j mark7\_back

addu $s1 $s0 $s0

mark8:

addiu $s2 $ra 16

jalr $s3 $s2

lui $s7 0xffff

mark9:

jr $ra

nop

jal mark10

lui $s6 0xffff

mark10:

addi $s5 $ra 16

jalr $s4 $s5

lui $t7 0xbbbb

lui $t9 0xcccc

j end\_2

ori $s7 0xffff

lui $t9 0xffff

lui $t8 0xaaaa

jr $s4

end\_2:

#mult\_div

ori $t0 $0 3

ori $t1 $0 4

lui $t2 0xffff

ori $t2 $t2 0xfff0

lui $t3 0x7000

ori $t3 $t3 0xffff

lui $t4 0xabcd

ori $t4 $t4 0xefef

mult $t0 $t1

mflo $s0

mthi $t4

mfhi $t4

mult $t2 $t3

mfhi $s1

mflo $s2

multu $t3 $t2

mfhi $s1

mflo $s2

or $s3 $s1 $0

mthi $s2

mtlo $s3

multu $t2 $t4

mfhi $s3

mflo $s4

div $t0 $t1

mfhi $s0

mflo $s1

div $t2 $t0

mfhi $s2

mflo $s3

divu $t2 $t0

mfhi $s2

mflo $s3

div $t3 $t4

mfhi $s2

mflo $s3

mthi $0

mtlo $0

mflo $s3

mfhi $s2

divu $t4 $t3

mfhi $s0

mflo $s1

#setbit

ori $t0 $0 2

ori $t1 $0 5

lui $t2 0x7000

lui $t3 0xffff

ori $t3 $t3 0xffff

slt $s0 $t0 $t3

slt $s0 $t3 $t0

slt $s0 $t3 $t3

slti $s1 $t2 0x7fff

slti $s1 $t3 0

slti $s1 $t3 -1

sltu $s2 $t0 $t3

sltu $s2 $t3 $t2

sltiu $s3 $t1 0x0fff

sltiu $s3 $t3 1

测试期望：

@00003000: $ 8 <= 00000001

@00003004: $ 9 <= 00000002

@00003008: $10 <= 00000003

@0000300c: $11 <= 00000005

@00003010: $12 <= ffff0000

@00003014: $12 <= fffffff0

@00003018: $13 <= fffffff5

@0000301c: $14 <= ffffffe5

@00003020: $ 8 <= 7fff0000

@00003024: $ 8 <= 7fff1234

@00003028: $ 1 <= 00000000

@0000302c: $ 1 <= 0000ffff

@00003030: $ 8 <= 80001233

@00003034: $ 1 <= 00000000

@00003038: $ 1 <= 0000ffff

@0000303c: $ 8 <= 80011232

@00003040: $15 <= 80011217

@00003044: $24 <= aaaa0000

@00003048: $25 <= 2aab1217

@0000304c: $25 <= 2aaabcc1

@00003050: $25 <= 2aaabcc0

@00003054: $25 <= 2aaabcbf

@00003058: $16 <= aaa9aaa8

@0000305c: $17 <= 55565558

@00003060: $18 <= aaacaab0

@00003064: $ 8 <= ffff0000

@00003068: $ 8 <= fffffff0

@0000306c: $ 9 <= 0000000a

@00003070: $19 <= 0000001a

@00003074: $20 <= ffffffe6

@00003078: $19 <= ffffffcc

@0000307c: $23 <= 00000008

@00003080: $12 <= ffffffef

@00003084: \*00000000 <= ffffffef

@00003088: $ 8 <= ffffffef

@0000308c: $13 <= fffffff7

@00003090: \*00000004 <= 000000f7

@00003094: $ 9 <= 000000ff

@00003098: $10 <= 000000f7

@0000309c: $14 <= ffffffdd

@000030a0: \*00000004 <= 0000ddf7

@000030a4: $11 <= 0000ddf7

@000030a8: \*00000004 <= 1217ddf7

@000030ac: $ 8 <= 1217ddf7

@000030b0: $16 <= aaa9aaa9

@000030b4: \*00000008 <= aaa9aaa9

@000030b8: $17 <= 5556555a

@000030bc: \*0000000c <= 0000555a

@000030c0: $ 9 <= aaa9aaa9

@000030c4: $18 <= aaacaaae

@000030c8: \*0000000c <= 00ae555a

@000030cc: $11 <= 000000ae

@000030d0: $19 <= ffffffc4

@000030d4: \*0000000c <= c4ae555a

@000030d8: $ 8 <= 000000c4

@000030dc: $ 8 <= abcd0000

@000030e0: $ 9 <= cdef0000

@000030e4: $ 8 <= abcd1234

@000030e8: $ 9 <= cdef5678

@000030ec: $10 <= 89cd1230

@000030f0: $11 <= cdef5678

@000030f4: $12 <= 89cd1230

@000030f8: $13 <= cdef5678

@000030fc: $14 <= 00000030

@00003100: $15 <= 000012b4

@00003104: $24 <= 00001284

@00003108: $25 <= 00000030

@0000310c: $ 8 <= 00007813

@00003110: $ 9 <= 00007813

@00003114: $10 <= ffff87ec

@00003118: $11 <= 00000000

@0000311c: $12 <= 0000abcd

@00003120: $13 <= ffff5432

@00003124: $ 8 <= 00000001

@00003128: $ 9 <= 00000002

@0000312c: $10 <= ffff0000

@00003130: $10 <= ffffffff

@00003134: $10 <= fffffffe

@00003138: $10 <= fffffffd

@00003140: $12 <= ffff0000

@0000317c: $16 <= 00000002

@00003184: $11 <= ffff0000

@00003144: $16 <= 00000001

@0000314c: $11 <= ffff0001

@00003188: $12 <= ffff0001

@00003154: $10 <= fffffffe

@00003194: $10 <= fffffffd

@00003158: $10 <= fffffffe

@00003160: $10 <= fffffffc

@000031a4: $10 <= 0000006c

@00003168: $10 <= 0000006e

@000031ac: $10 <= 0000006c

@00003170: $10 <= 0000006d

@000031b4: $ 8 <= fff00000

@000031b4: $ 8 <= fff00000

@000031b8: $ 8 <= fff0abcd

@000031bc: $ 9 <= 00000003

@000031c0: $10 <= ffc2af34

@000031c4: $10 <= fe1579a0

@000031c8: $10 <= 1fc2af34

@000031cc: $ 9 <= 00000002

@000031d0: $10 <= 07f0abcd

@000031d4: $ 9 <= 00000005

@000031d8: $10 <= 003f855e

@000031dc: $11 <= 80000000

@000031e0: $10 <= 803f855e

@000031e4: $10 <= f803f855

@000031e8: $ 9 <= 00000002

@000031ec: $10 <= fe00fe15

@000031f0: $11 <= 7fff0000

@000031f4: $11 <= 7fffffff

@000031f8: $10 <= 7e00fe15

@000031fc: $10 <= 1f803f85

@00003200: $10 <= 07e00fe1

@00003208: $16 <= 0000ffff

@00003234: $17 <= 0001fffe

@0000320c: $31 <= 00003214

@00003238: $18 <= 00003224

@0000323c: $19 <= 00003244

@00003240: $23 <= ffff0000

@00003224: $16 <= 0000bbbb

@00003214: $16 <= 0000aaaa

@00003218: $19 <= 0000324c

@0000324c: $31 <= 00003254

@00003250: $22 <= ffff0000

@00003254: $21 <= 00003264

@00003258: $20 <= 00003260

@0000325c: $15 <= bbbb0000

@00003268: $23 <= ffffffff

@00003278: $ 8 <= 00000003

@0000327c: $ 9 <= 00000004

@00003280: $10 <= ffff0000

@00003284: $10 <= fffffff0

@00003288: $11 <= 70000000

@0000328c: $11 <= 7000ffff

@00003290: $12 <= abcd0000

@00003294: $12 <= abcdefef

@0000329c: $16 <= 0000000c

@000032a4: $12 <= abcdefef

@000032ac: $17 <= fffffff8

@000032b0: $18 <= fff00010

@000032b8: $17 <= 7000fff7

@000032bc: $18 <= fff00010

@000032c0: $19 <= 7000fff7

@000032d0: $19 <= abcdefe4

@000032d4: $20 <= 43210110

@000032dc: $16 <= 00000003

@000032e0: $17 <= 00000000

@000032e8: $18 <= ffffffff

@000032ec: $19 <= fffffffb

@000032f4: $18 <= 00000000

@000032f8: $19 <= 55555550

@00003300: $18 <= 1bceefee

@00003304: $19 <= ffffffff

@00003310: $19 <= 00000000

@00003314: $18 <= 00000000

@0000331c: $16 <= 3bcceff0

@00003320: $17 <= 00000001

@00003324: $ 8 <= 00000002

@00003328: $ 9 <= 00000005

@0000332c: $10 <= 70000000

@00003330: $11 <= ffff0000

@00003334: $11 <= ffffffff

@00003338: $16 <= 00000000

@0000333c: $16 <= 00000001

@00003340: $16 <= 00000000

@00003344: $17 <= 00000000

@00003348: $17 <= 00000001

@0000334c: $17 <= 00000000

@00003350: $18 <= 00000001

@00003354: $18 <= 00000000

@00003358: $19 <= 00000001

@0000335c: $19 <= 00000000

4.11 综合测试 4

.data

mem: .space 400

.text

ori $t0 $0 6

ori $t1 $0 8

lui $t2 0xffff

ori $t2 $t2 0xffff

addi $t3 $t2 -2

lui $t4 0xfabc

lui $t5 0x6543

ori $t4 $t4 0xffff

ori $t5 $t5 0xabcd

blez $t2 mark1

mult $t0 $t1

mark1\_back:

lui $s0 0xaaaa

mfhi $t9

mflo $t8

sw $t8 0($0)

sh $t9 2($0)

sb $t9 3($0)

sb $t9 0($0)

lhu $t8 2($0)

lbu $t8 0($0)

lh $t9 0($0)

lb $t8 3($0)

mult $t9 $t8

div $t8 $t9

multu $t2 $t4

mfhi $s7

mflo $s6

mtlo $t9

mthi $t8

mfhi $s7

mflo $s6

jal mark2

addi $t8 $t8 2

ori $s4 $0 0xf0

sw $t8 -16($s4)

lb $t8 -15($s4)

jr $t7

mfhi $t7

mark1:

mfhi $t9

mflo $t8

mthi $t8

slt $s0 $t4 $t2

bne $s0 $0 mark1\_back

div $t4 $t5

mark2:

addi $t6 $ra -4

jalr $t7 $t6

multu $t2 $t4

mflo $t8

addu $t8 $t7 $t8

addi $t8 $t8 -1

bltz $t8 mark3

addi $t8 $t8 1

beq $t8 $0 mark4

mult $t7 $t9

mark4:

mthi $t1

mfhi $s7

mflo $s6

jr $t1

mark3:

ori $t0 $0 0x30cc

jalr $t1 $t0

addi $t1 $t1 4

lui $s0 0xbbbb

lui $s0 0xcccc #30f8

ori $s1 0x3104

jalr $s2 $s1

addi $s2 $s1 8

lui $s3 0xaaaa

lui $s4 0xbbbb

bgtz $s3 mark5

sh $s3 4090($0)

addi $s3 $s3 -1

jr $s2

xor $s3 $s3 $s4

mark5:

mthi $s3

mtlo $s4

mult $s3 $s4

mflo $t2

bgez $t2 mark6

lui $s7 0xffff

mark6:

or $t2 $s7 $0

slt $t3 $t2 $0

sltu $t3 $t2 $0

xori $t3 $t2 0x1bdc

slti $t4 $t3 -3

sltiu $t4 $t3 3

ori $t0 $0 1

ori $t1 $0 3 #3158

ori $t2 $0 0x3168

jalr $s7 $t2

addi $t1 $t1 -1

beq $t0 $t1 mark7

nop

jr $s7

addi $s7 $s7 -8

mark7:

divu $s7 $s6

mfhi $s7

mflo $s6

sh $s7 3332($0)

ori $t0 $0 4096

sb $s6 -1600($t0)

测试期望：

@00003000: $ 8 <= 00000006

@00003004: $ 9 <= 00000008

@00003008: $10 <= ffff0000

@0000300c: $10 <= ffffffff

@00003010: $11 <= fffffffd

@00003014: $12 <= fabc0000

@00003018: $13 <= 65430000

@0000301c: $12 <= fabcffff

@00003020: $13 <= 6543abcd

@00003098: $25 <= 00000000

@0000309c: $24 <= 00000030

@000030a4: $16 <= 00000001

@0000302c: $16 <= aaaa0000

@00003030: $25 <= fabcffff

@00003034: $24 <= 00000000

@00003038: \*00000000 <= 00000000

@0000303c: \*00000000 <= ffff0000

@00003040: \*00000000 <= ffff0000

@00003044: \*00000000 <= ffff00ff

@00003048: $24 <= 0000ffff

@0000304c: $24 <= 000000ff

@00003050: $25 <= 000000ff

@00003054: $24 <= ffffffff

@00003064: $23 <= fabcfffe

@00003068: $22 <= 05430001

@00003074: $23 <= ffffffff

@00003078: $22 <= 000000ff

@0000307c: $31 <= 00003084

@00003080: $24 <= 00000001

@000030b0: $14 <= 00003080

@000030b4: $15 <= 000030bc

@00003080: $24 <= 00000003

@00003084: $20 <= 000000f0

@00003088: \*000000e0 <= 00000003

@0000308c: $24 <= 00000000

@00003094: $15 <= fabcfffe

@000030bc: $24 <= 05430001

@000030c0: $24 <= ffffffff

@000030c4: $24 <= fffffffe

@000030cc: $24 <= ffffffff

@000030e8: $ 8 <= 000030cc

@000030ec: $ 9 <= 000030f4

@000030f0: $ 9 <= 000030f8

@000030cc: $24 <= 00000000

@000030dc: $23 <= 000030f8

@000030e0: $22 <= c242fe02

@000030e8: $ 8 <= 000030cc

@000030f8: $16 <= cccc0000

@000030fc: $17 <= 00003104

@00003100: $18 <= 00003108

@00003104: $18 <= 0000310c

@00003104: $18 <= 0000310c

@00003108: $19 <= aaaa0000

@0000310c: $20 <= bbbb0000

@00003114: \*00000ff8 <= 00000000

@00003118: $19 <= aaa9ffff

@00003120: $19 <= 1112ffff

@0000310c: $20 <= bbbb0000

@00003114: \*00000ff8 <= ffff0000

@00003130: $10 <= 44450000

@00003138: $23 <= ffff0000

@0000313c: $10 <= ffff0000

@00003140: $11 <= 00000001

@00003144: $11 <= 00000000

@00003148: $11 <= ffff1bdc

@0000314c: $12 <= 00000001

@00003150: $12 <= 00000000

@00003154: $ 8 <= 00000001

@00003158: $ 9 <= 00000003

@0000315c: $10 <= 00003168

@00003160: $23 <= 00003168

@00003164: $ 9 <= 00000002

@00003174: $23 <= 00003160

@00003174: $23 <= 00003158

@00003160: $23 <= 00003168

@00003164: $ 9 <= 00000001

@0000317c: $23 <= 00003168

@00003180: $22 <= 00000000

@00003184: \*00000d04 <= 00003168

@00003188: $ 8 <= 00001000

@0000318c: \*000009c0 <= 00000000

5 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？

之所以不整合进ALU是因为乘除法运算需要模拟实际的运算延时，而ALU中的其他运算均不模拟延时，在进行乘除法运算的同时，其他指令在与乘除法运算不冲突的情况下仍然可以正常执行，如果都整合到ALU的话会导致ALU输入输出信号更多，使这一过程的控制非常不便。

之所以要有独立的乘除法结果的寄存器而不使用通用寄存器，是因为通用寄存器的每一个都可以被任意读写，容易造成对乘除法结果的错误修改。同时，要想读乘除法结果时由于没有指定存入的寄存器，导致读不方便。而且，还会为流水线的设计造成不便，因为计算出结果并写回会会与其他指令冲突，进而还要设计相应的暂停机制。

2.参照你对延迟槽的理解，试解释“乘除槽”。

MIPS的延迟槽正是为流水线的结构而设计的。在实验中考虑的事分支延迟槽，在执行分支指令时，只有在进入了ID级才可以知道下一条指令的地址，如果没有延迟槽的设计，那么就要使流水线暂停一个周期，等待下一条指令的地址。设计了延迟槽后，可以在下一条指令的地址计算出之前就先执行下一条指令，一方面能够当不进行跳转时节省时间，另一方面即使是跳转也可以在这个等待周期中执行一条不会造成影响的指令，提高效率。

这里的“乘除槽”便是一种类比。在乘除结果计算出来之前，如果当前指令不对乘除模块的寄存器读写，那么就相当于上文“不会造成影响的指令”，可以继续执行而不必等待，节省时间。

3.为何上文文末提到的lb等指令使用的数据扩展模块应在 MEM/WB 之后，而不能在 DM 之后?

在实际情况中，DM的写入是比读取要快的，因为写入一般是先写入到cache中，然后由cache向主存储器写入，即只要写入了cache，后续的过程就不再CPU时钟周期的考虑范围之内了，但是读取的时候，从寄存器，到cache，最坏的情况是到主存储器中读。因此平均情况下，DM的写入比读取要快。因此，如果把数据扩展模块放到DM后，本来读取就有较大的延时，再加上拓展模块组合逻辑的延时，势必会增加CPU时钟周期，如果放在MEM/WB后，由于DM写入时间相对较短，而流水线CPU时钟周期取决于耗时最长的操作，这样即使加上拓展模块组合逻辑的延时，也很可能不会增加CPU时钟周期。

4.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）

比如在处理字符串的情况，由于一个字符串的大小是一字节，如果只能按字进行访问的话，要想实现一条指令读取字符串中的，则在存储时一个字符必须占用一个字的空间，会造成巨大的空间浪费，而如果一个自己一个字符进行存储，按字访问无法一次取一个字符，在进行类似于C语言中的字符串操作时则会更加不便。而按字节访问则会更加方便。

5. 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？

我认为我的设计风格更偏向于第二种“规划者”型。虽然这样确实会造成代码量的增加，但是我会在设计前用Excel列出转发和暂停的分析表格，只要前期考虑清楚，然后仔细写相关代码，这样在改动时借助相关表格一目了然。

关于对抗复杂性的抽象手段，一，为了防止不同流水级的信号重名，我会在不同流水级信号的后面加上下划线以及流水级的缩写；二，使用宏定义，在开始就用宏定义定义相关指令的相关部分，避免在后期因写01出现细小错误；三，在设计转发和暂停时会按类别设置，后期，如果遇到同类的指令，只要在开始时增加对这个类型的“解释”，而不必修改后面的控制逻辑。

6. 你对流水线CPU设计风格有何见解？

在CPU前期设计过程中，除了数据通路，最重要的就是对转发与暂停的分析，当实际设计时，可以使用教程中的工程化方法。但是我认为，在兼顾工程化方法，通过将不同指令归类简化转发、暂停机制的实际操作过程时，应当兼顾不同指令的不同之处，比如branch类型，beq要使用rs和rt对应的寄存器，但是bgtz这样的指令并不会使用rt，可以另外增加一个类型，以减少暂停。

在实际操作过程中，关于我自己的代码书写，我会选择使用一些更长的描述名称来使设计更加容易理解，而且由于流水线有不同的流水级，我会在每一级的变量后加上这一级的后缀，同时统一把变量按不同级别分看写在数据通路的顶端，同时，以注释的形式去区分不同的流水段，以方便操作。

7. 在本实验中你遇到了哪些不同指令组合产生的冲突？你又是如何解决的？相应的测试样例是什么样的？请有条理的罗列出来。

请见“P6\_相关表格.xlsx”，在其中以列表形式列出。