Verilog流水线CPU设计文档

一、模块定义

1.IFU：

表1 功能表

模块名称	设计描述
IFU	由一个32bit*1024的ROM保存指令，输入一个合理的地址，可以输出相应位置上面的32bit指令，同时可以复位以实现CPU重启功能。

表2 输入端口

序号	功能名称	功能描述
PC_next[31:0]	下条指令地址	下一个时钟上沿到来时，传给指令寄存器
Reset	复位	异步复位，1时触发，使当前指令地址变为0x00003000
Npc_on	NPC工作信号	Npc_on = 1时，下一条指令的地址由NPC模块提供；Npc_on = 0时，下一条指令由Adder提供。
Stop	冻结信号	当stop = 1时，下一个时钟上沿到来时，PC_next不会传给指令寄存器，且指令寄存器保持不变。

表3 输出端口

序号	功能名称	功能描述
PC_now[31:0]	当前指令地址	输出以便计算下条指令
Ins[31:0]	当前指令	交给其他模块处理

2.GRF：

表4 功能表

模块名称	设计描述
GRF	符合MIPS体系结构，一共有32个通用寄存器。 由于MIPS指令定义的缘故，最多同时允许写入一个寄存器，输出两个寄存器。 可以通过Reset触发清空寄存器操作。

表5 输入端口

序号	功能名称	功能描述
RS[4:0]	RS对应的寄存器	5位二进制代码表示RS寄存器的编号，由译码器对应分配
RT[4:0]	RT对应的寄存器	5位二进制代码表示RT寄存器的编号，由译码器对应分配
Grf_reg_in[4:0]	写入对应的寄存器	5位二进制代码表示写入寄存器的编号，由译码器对应分配
Grf_data_in[31:0]	需要写入寄存器的数据	下一个时钟上沿到来时，将数据写入某一个寄存器
Grf_write	是否写入寄存器的标志	0表示当前指令不需要修改寄存器； 1表示当前指令需要写入寄存器。
Pc_now	现在PC的地址	用于display
Reset	复位	异步复位，高电平1时触发，使所有寄存器清空

表6 输出端口

序号	功能名称	功能描述
RS_Data[31:0]	RS寄存器的值	可能用作ALU的第一个输入数据
RT_Data[31:0]	RT寄存器的值	可能用作ALU的第二个输入数据

3.ALU：

表7 功能表

模块名称	设计描述
ALU	算术逻辑运算单元，应根据控制信号进行加、减、与、或等数值或逻辑运算，并将结果输出。

表8 输入端口

序号	功能名称	功能描述
A[31:0]	ALU第一个输入数据	为RS寄存器的数据或者表示基地址Base的值
B[31:0]	ALU第二个输入数据	为Rt寄存器的值，或者为扩展后的立即数
ALU_Ctr[2:0]	ALU控制信号	3bit控制信号，分别控制ALU进行不同的运算 +:000 -:001 |:010 ^:011

表9 输出端口

序号	功能名称	功能描述

Result[31:0]	ALU运算结果	无条件输出ALU当前的运算结果

4.DM：

表10 功能表

模块名称	设计描述
DM	32bit*1024的RAM组成的内存，用以存储寄存器所不能长期存放的数据。

表11 输入端口

序号	功能名称	功能描述
Address[31:0]	要写入的位置	虽然是32位，但由于RAM只有32bit*32，所以截取[6:2]位作为地址的选择。
Dm_data_in[31:0]	要写入的数据	下一个时钟上沿到来时，将Data写入Address所对应的位置上。
Pc_now[31:0]	当前指令的地址	用于display输出
Reset	复位	异步复位，高电平1时触发，使RAM中所有数据清空。
Dm_write	DM写入信号	1表示向DM中写入数据。

表12 输出端口

序号	功能名称	功能描述

Dm_data_out[31:0]	DM的输出数据	一直输出ram[address]的数据。

5.Extender

表13 功能表

模块名称	设计描述
Extender	将所有的16bit立即数输入，按照扩展信号进行符号扩展或者零扩展，变成32bit输出。

表14 输入端口

序号	功能名称	功能描述
Immediate[15:0]	16bit的立即数	16bit立即数输入
ExtendType	要扩展的类型	0表示进行零扩展； 1表示进行符号扩展

表15 输出端口

序号	功能名称	功能描述
Output[31:0]	Extender的输出数据	32bit扩展后数据输出，可供ALU作为输入使用。

7. NPC

表19 功能表

模块名称	设计描述
NPC	根据控制信号和ALU 的结果，和当前PC的值，计算下一条指令的地址

表20 输入端口

序号	功能名称	功能描述

Jr_offset [31:0]	Jr跳转的地址	数据来源于GRF的第31个寄存器
PC_Now[31:0]	当前的PC值	为J指令提供前四位，为B指令提供PC+4
Extend_imm[31:0]	32bit指令中后16bit的立即数，被扩展成32bit	提供偏移量，供B类指令使用
Ins[31:0]	当前指令	通过控制信号，对当前指令进行解析，得到计算pc_next所需的数据。
AequalsB	GRF输出Rs和Rt是否相等	给BEQ指令提供控制信号
Npc_branch	B类指令控制信号	B类指令控制信号
Npc_jump	J指令控制信号	J指令控制信号
Npc_jr	Jr指令控制信号	Jr指令控制信号

表21 输出端口

序号	功能名称	功能描述
Pc_next [31:0]	下一条PC指令地址	为IFU提供下一条指令的地址
Pc_on	跳转信号	1时PC接收来自于npc的输入，0时接收来自于Adder的输入。

8.Mux_2_32bit

表22 功能表

模块名称	设计描述
Mux_1bit	可以根据1bit控制信号，在两个输入数据间进行选择

表23 输入端口

序号	功能名称	功能描述
In0[31:0]	输入0	第一个输入数据
In1[31:0]	输入1	第二个输入数据
Slc	选择信号	选择信号

表24 输出端口

序号	功能名称	功能描述
Out[31:0]	输出	Slc = 0 输出In0 Slc = 1输出In1

8.Mux_4_5bit

表22 功能表

模块名称	设计描述
Mux_4_5bit	可以根据2bit控制信号，在4个输入数据间进行选择，主要用于Grf的寄存器选择。

表23 输入端口

序号	功能名称	功能描述
In0[4:0]	输入0	第一个输入数据
In1[4:0]	输入1	第二个输入数据
In2[4:0]	输入2	第三个输入数据
In3[4:0]	输入3	第四个输入数据
Slc	选择信号	选择信号

表24 输出端口

序号	功能名称	功能描述

Out[4:0]	输出	Slc = 0 输出In0 Slc = 1 输出In1 Slc = 2 输出In2 Slc = 3 输出In3

8.Mux_4_32bit

表22 功能表

模块名称	设计描述
Mux_4_32bit	可以根据2bit控制信号，在4个输入数据间进行选择

表23 输入端口

序号	功能名称	功能描述
In0[31:0]	输入0	第一个输入数据
In1[31:0]	输入1	第二个输入数据
In2[31:0]	输入2	第三个输入数据
In3[31:0]	输入3	第四个输入数据
Slc	选择信号	选择信号

表24 输出端口

序号	功能名称	功能描述
Out[31:0]	输出	Slc = 0 输出In0 Slc = 1 输出In1 Slc = 2 输出In2 Slc = 3 输出In3

8.Mux_8_32bit

表22 功能表

模块名称	设计描述
Mux_1bit	可以根据1bit控制信号，在两个输入数据间进行选择

表23 输入端口

序号	功能名称	功能描述
In0[31:0]	输入0	第一个输入数据
In1[31:0]	输入1	第二个输入数据
In2[31:0]	输入2	第三个输入数据
In3[31:0]	输入3	第四个输入数据
In4[31:0]	输入4	第五个输入数据
In5[31:0]	输入5	第六个输入数据
In6[31:0]	输入6	第七个输入数据
In7[31:0]	输入7	第八个输入数据
Slc	选择信号	选择信号

表24 输出端口

序号	功能名称	功能描述
Out[31:0]	输出	Slc = 0 输出In0 Slc = 1 输出In1 Slc = 2 输出In2 Slc = 3 输出In3 Slc = 4 输出In4 Slc = 5 输出In5 Slc = 6 输出In6 Slc = 7 输出In7

10.MIPS

表28 功能表

模块名称	设计描述
MIPS	最顶层模块，用于将DataPath和Controller进行连接，以实现数据交换

表29 输入端口

序号	功能名称	功能描述
Clk	时钟信号	作为与外部的接口，为DataPath提供时钟信号
Reset	重置信号	作为与外部的接口，为DataPath提供清零信号

表30 输出端口

序号	功能名称	功能描述
无输出	/	/

11.Controller

表31 功能表

模块名称	设计描述
Controller	根据当前32bit指令中的6bitOp段和func段，生成该周期内需要使用到的所有模块内部控制信号。

表32 输入端口

序号	功能名称	功能描述
ins[31:0]	当前周期的指令	根据当前周期的指令的OpCode和FuncCode进行分析，得到所有模块所需要的信号。

表33 输出端口

序号	功能名称	功能描述
Dm_write	DM读写控制信号	0读取DM 1写入DM
Grf_write	GRF写入控制信号	0不写入Reg 1写入Reg
Extender_type	立即数扩展类型	0进行0扩展 1进行符号扩展
Alu_src	ALU的第二个数据来源	0来自RT 1来自Extend被扩展后的立即数
Grf_dst[1:0]	要写入的寄存器	0表示写入RT寄存器 1表示写入RD寄存器 2表示写入31号寄存器
Npc_branch	分支控制信号	0不跳转 1分支跳转
Alu_ctr[2:0]	ALU控制信号源	详情见ALU输入端口
Npc_jump	跳转控制信号	0不跳转 1JumpOrJL
Npc_jr	Jr跳转信号	0不跳转 1Jr跳转
Grf_data_in_type	Grf输入数据种类	0alu_result 1dm_data_out 2 pc+8

设计思路：

1.思考MIPS顶层基本架构，由指令存储器IFU、控制器Controller、寄存器堆GRF、算术逻辑运算单元ALU、ALU行为控制器ALUController、扩展器Extender和存储器DM组成。

2.根据基本的指令和顶层MIPS框架，选择最基础的控制信号：

①与指令寄存器IFU相关，Beq指令的是否Branch控制信号；

②与寄存器堆GRF相关的，是否写入控制信号RegWrite、写入内容选择信号MemToReg和写入的寄存器选择信号RegDst；

③与ALU相关的，ALU行为片选信号ALU_Op和ALU数据来源选择信号ALUSrc；

④与Extender相关的，所有有立即数参与的指令都要经过Extender将16bit扩展为32bit，扩展信号ExtendType；

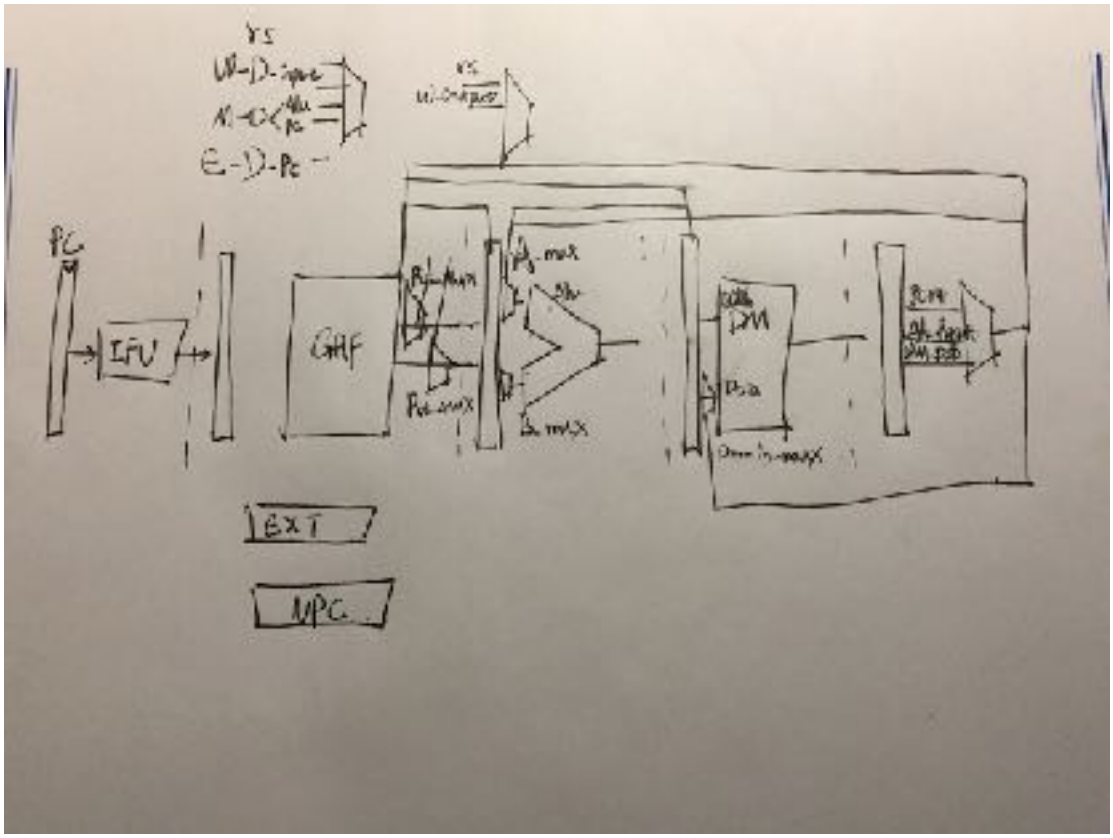
⑤与存储器DM相关的，是否写入DM 的控制信号MemReg。

3.若要新增指令，需首先考虑其是否需要使用之前的基础控制信号，若不能完全满足指令操作需求，再增加一条该指令的专属信号或可以共用的新信号。

设计流程：

1.将所有指令的6bitOpCode进行与运算，形成自己的专属标志，以判断该指令为何种指令。

2.将所有指令在其所需要的控制信号处进行或运算，以保证每个指令产生所有所需的控制信号。



12.Hazard

表31 功能表

模块名称	设计描述
Hazard	根据当前状态下各级流水线的指令，进行综合判断，是否需要进行暂停和转发。

表32 输入端口

序号	功能名称	功能描述
Ins_d[31:0]	D流水级当前周期的指令	根据当前周期的指令的OpCode和FuncCode进行分析，得到stall和转发信号。
Ins_e[31:0]	E流水级当前周期的指令	根据当前周期的指令的OpCode和FuncCode进行分析，得到stall和转发信号。

Ins_m[31:0]	M流水级当前周期的指令	根据当前周期的指令的OpCode和FuncCode进行分析，得到stall和转发信号。
Ins_w[31:0]	W流水级当前周期的指令	根据当前周期的指令的OpCode和FuncCode进行分析，得到stall和转发信号。

表33 输出端口

序号	功能名称	功能描述
stop	暂停信号	0正常工作 1熄火
Rs_data_d_to_reg_type[2:0]	D级rs数据转发	0 rs原本数据或内部转发后数据 1 E级PC 2 M级PC 3 M级ALU_Result
Rt_data_d_to_reg_type[2:0]	D级rt数据转发	0 rt原本数据或内部转发后数据 1 E级PC 2 M级PC 3 M级ALU_Result
Rs_data_e_to_alu_type[1:0]	E级rs数据转发	0 D可能转发过的rs_data 1 M级的alu_result 2 W级要写入寄存器的结果
Rt_data_e_to_alu_type[1:0]	E级rt数据转发	0 D可能转发过的rt_data 1 M级的alu_result 2 W级要写入寄存器的结果
Rt_data_m_to_dm_type	M级rt数据转发	0 E可能转发过的rt_data 1 W级要写入寄存器的结果

二、测试代码及期望结果：

```
#initial
addi $s0 $0 131071
addi $s1 $0 5
addi $s2 $0 65535
addi $s3 $0 8
lui $s5 0xABCD
add $s6 $t0 0xffff
#initial done
addu $t0 $s0 $s1
addu $t1 $s0 $t0
sllv $t2 $s0 $t0
and $t3 $s0 $t0
sw $t3 0($0)
sw $t2 4($0)
sw $t1 8($0)
srlv $t0 $s0 $s1
srlv $t1 $t0 $s0
srlv $t2 $t0 $s0
addu $t3 $t0 $s0
sw $t1 12($0)
sw $t3 16($0)
sw $t2 20($0)
xor $t0 $s0 $s1
ori $t1 $t0 59
ori $t2 $t0 21
slti $t3 $t0 36
sw $t3 24($0)
sw $t2 28($0)
sw $t1 32($0)
addi $t0 $s1 100
sllv $t0 $0 $0
lw $t1 0($t0)
lb $t2 4($t0)
lhu $t3 8($t0)
sw $t3 36($0)
sh $t2 40($0)
sw $t1 44($0)
addi $t0 $s1 100
sltiu $t0 $0 0
lh $t1 0($t0)
lhu $t2 4($t0)
lhu $t3 8($t0)
sb $t3 48($0)
sh $t2 52($0)
sh $t1 56($0)
xori $t0 $s1 997
subu $t1 $s0 $t0
srlv $t2 $s0 $t0
nor $t3 $s0 $t0
sw $t3 60($0)
sw $t2 64($0)
sw $t1 68($0)
xori $t0 $s1 631
or $t1 $t0 $s0
nor $t2 $t0 $s0
subu $t3 $t0 $s0
sw $t1 72($0)
sw $t3 76($0)
sw $t2 80($0)
ori $t0 $s1 0
ori $t1 $t0 7
andi $t2 $t0 1
slti $t3 $t0 2
sw $t2 84($0)
sw $t1 88($0)
sw $t3 92($0)
lb $t0 20($0)
```

```
sltu $t1 $s0 $t0
slt $t2 $s0 $t0
subu $t3 $s0 $t0
sw $t3 96($0)
sw $t2 100($0)
sw $t1 104($0)
lb $t0 8($0)
sllv $t1 $t0 $s0
subu $t2 $t0 $s0
xor $t3 $t0 $s0
sw $t1 108($0)
sw $t3 112($0)
sw $t2 116($0)
lbu $t0 72($0)
xori $t1 $t0 100
slti $t2 $t0 100
slti $t3 $t0 100
sw $t3 120($0)
sw $t2 124($0)
sw $t1 128($0)
addi $t0 $0 4
sw $t0 132($0)
lhu $t0 132($0)
lbu $t1 0($t0)
lh $t2 4($t0)
lh $t3 8($t0)
sh $t3 132($0)
sw $t2 136($0)
sh $t1 140($0)
nor $t0 $s0 $s1
srl $t1 $t0 0
sll $t2 $t0 1
sra $t3 $t0 3
sw $t3 144($0)
sw $t2 148($0)
sw $t1 152($0)
sltiu $t0 $s1 0
sra $t1 $s1 0
srl $t2 $s1 2
srl $t3 $s1 2
sw $t2 156($0)
sw $t1 160($0)
sw $t3 164($0)
lb $t0 0($0)
srl $t1 $t0 1
srl $t2 $t0 0
srl $t3 $t0 3
sw $t3 168($0)
sw $t2 172($0)
sw $t1 176($0)
srl $t0 $s0 0
xor $t1 $s0 $t0
addu $t2 $s0 $t0
xor $t3 $s0 $t0
sw $t3 180($0)
sw $t2 184($0)
sw $t1 188($0)
srl $t0 $s0 2
or $t1 $t0 $s0
addu $t2 $t0 $s0
subu $t3 $t0 $s0
sw $t1 192($0)
sw $t3 196($0)
sw $t2 200($0)
sll $t0 $s0 4
addiu $t1 $t0 9
xori $t2 $t0 9
sltiu $t3 $t0 1
sw $t2 204($0)
```

```
sw $t1 208($0)
sw $t3 212($0)
addi $t0 $0 4
addi $t1 $0 256
sll $t2 $t0 2
lh $t3 0($t2)
lbu $t4 4($t2)
lb $t5 8($t2)
sw $t3 216($0)
sw $t4 220($0)
sw $t5 224($0)
srl $t2 $t1 6
lb $t3 0($t2)
lw $t4 4($t2)
lh $t5 8($t2)
sw $t3 228($0)
sw $t4 232($0)
sw $t5 236($0)
srlv $t0 $s0 $s1
srav $t1 $s2 $s3
multu $t1 $t0
subu $t2 $s0 $s1
sllv $t3 $s2 $s3
or $t4 $s0 $s1
mfhi $t5
xor $t1 $t5 $s3
sllv $t6 $t5 $s1
xori $t0 $t5 -100
mult $t1 $t0
sw $t3 240($0)
sw $t4 244($0)
sw $t2 248($0)
sw $t6 252($0)
mfhi $t5
andi $t7 $t5 4
ori $t8 $t5 4
and $t9 $t5 $s1
xori $t0 $s2 4
addiu $t1 $s2 4
ori $t2 $s2 4
mult $t0 $t1
sw $t7 256($0)
sw $t8 260($0)
sw $t9 264($0)
mfhi $t5
sw $t5 268($0)
sw $t5 272($0)
sw $t5 276($0)
lui $t0 0xffff
add $t0 $t0 $s6
sw $t0 280($0)
lh $t0 280($0)
lhu $t1 280($0)
mult $t1 $t0
srav $t2 $s0 $s1
sllv $t3 $s2 $s3
srlv $t4 $s0 $s1
mflo $t5
lw $t0 280($0)
sra $t6 $t5 3
srl $t1 $t5 4
mult $t1 $t0
sw $t3 280($0)
sw $t4 284($0)
sw $t2 288($0)
sw $t6 292($0)
mflo $t5
sll $t7 $t5 4
sltiu $t8 $t5 7
```

```
or $t9 $t5 $s1
sll $t1 $s0 2
sll $t0 $s0 3
xori $t2 $s2 4
multu $t0 $t1
sw $t7 296($0)
sw $t8 300($0)
sw $t9 304($0)
mflo $t5
sw $t5 308($0)
sw $t5 312($0)
sw $t5 316($0)
slt $t0 $s0 $s1
nop
nop
mthi $t0
mfhi $t1
srlv $t0 $s0 $s1
nop
mtlo $t0
mflo $t2
and $t0 $s0 $s1
mthi $t0
mfhi $t3
sw $t1 320($0)
sw $t2 324($0)
sw $t3 328($0)
slti $t0 $s1 737
nop
nop
mthi $t0
mfhi $t1
andi $t0 $s1 860
nop
mtlo $t0
mflo $t2
xori $t0 $s1 922
mthi $t0
mfhi $t3
sw $t1 332($0)
sw $t2 336($0)
sw $t3 340($0)
sra $t0 $s0 2
nop
nop
mthi $t0
mfhi $t1
sra $t0 $s0 4
nop
mtlo $t0
mflo $t2
srl $t0 $s0 1
mthi $t0
mfhi $t3
sw $t1 344($0)
sw $t2 348($0)
sw $t3 352($0)
lb $t0 0($0)
nop
nop
mthi $t0
mfhi $t1
lw $t0 0($0)
nop
mtlo $t0
mflo $t2
lhu $t0 0($0)
mthi $t0
mfhi $t3
```

```
sw $t1 356($0)
sw $t2 360($0)
sw $t3 364($0)
addi $t0 $0 4
mthi $t0
mfhi $t5
nop
nop
lbu $t1 0($t5)
addi $t0 $0 0
mthi $t0
mfhi $t5
nop
lhu $t2 0($t5)
addi $t0 $0 8
mthi $t0
mfhi $t5
lbu $t3 0($t5)
sw $t1 368($0)
sw $t2 372($0)
sw $t3 376($0)
addu $t0 $s0 $s1
srlv $t1 $s0 $t0
srlv $t2 $s0 $t0
sltu $t3 $s0 $t0
sw $t3 380($0)
sw $t2 384($0)
sw $t1 388($0)
addu $t0 $s0 $s1
srlv $t1 $t0 $s0
nor $t2 $t0 $s0
and $t3 $t0 $s0
sw $t1 392($0)
sw $t3 396($0)
sw $t2 400($0)
or $t0 $s0 $s1
sltiu $t1 $t0 88
slti $t2 $t0 5
slti $t3 $t0 56
sw $t3 404($0)
sw $t2 408($0)
sw $t1 412($0)
addi $t0 $s1 100
xor $t0 $0 $0
lhu $t1 0($t0)
lw $t2 4($t0)
lw $t3 8($t0)
sh $t3 416($0)
sw $t2 420($0)
sw $t1 424($0)
addi $t0 $s1 100
sltiu $t0 $0 0
lhu $t1 0($t0)
lh $t2 4($t0)
lb $t3 8($t0)
sh $t3 428($0)
sb $t2 432($0)
sh $t1 436($0)
xori $t0 $s1 483
nor $t1 $s0 $t0
sltu $t2 $s0 $t0
and $t3 $s0 $t0
sw $t3 440($0)
sw $t2 444($0)
sw $t1 448($0)
xori $t0 $s1 857
addu $t1 $t0 $s0
srlv $t2 $t0 $s0
srav $t3 $t0 $s0
```

```
sw $t1 452($0)
sw $t3 456($0)
sw $t2 460($0)
ori $t0 $s1 2
slti $t1 $t0 1
sltiu $t2 $t0 2
slti $t3 $t0 5
sw $t2 464($0)
sw $t1 468($0)
sw $t3 472($0)
lhu $t0 356($0)
subu $t1 $s0 $t0
and $t2 $s0 $t0
srav $t3 $s0 $t0
sw $t3 476($0)
sw $t2 480($0)
sw $t1 484($0)
lhu $t0 388($0)
slt $t1 $t0 $s0
subu $t2 $t0 $s0
sllv $t3 $t0 $s0
sw $t1 488($0)
sw $t3 492($0)
sw $t2 496($0)
lh $t0 392($0)
xori $t1 $t0 100
slti $t2 $t0 100
addiu $t3 $t0 100
sw $t3 500($0)
sw $t2 504($0)
sw $t1 508($0)
addi $t0 $0 4
sw $t0 512($0)
lhu $t0 512($0)
lbu $t1 0($t0)
lw $t2 4($t0)
lh $t3 8($t0)
sb $t3 512($0)
sw $t2 516($0)
sb $t1 520($0)
sllv $t0 $s0 $s1
srl $t1 $t0 1
sll $t2 $t0 1
srl $t3 $t0 0
sw $t3 524($0)
sw $t2 528($0)
sw $t1 532($0)
sltiu $t0 $s1 1
sra $t1 $s1 0
sra $t2 $s1 3
sra $t3 $s1 0
sw $t2 536($0)
sw $t1 540($0)
sw $t3 544($0)
lw $t0 0($0)
sll $t1 $t0 3
sra $t2 $t0 0
sll $t3 $t0 0
sw $t3 548($0)
sw $t2 552($0)
sw $t1 556($0)
sra $t0 $s0 3
or $t1 $s0 $t0
xor $t2 $s0 $t0
sltu $t3 $s0 $t0
sw $t3 560($0)
sw $t2 564($0)
sw $t1 568($0)
sra $t0 $s0 2
```

```
sltu $t1 $t0 $s0
srav $t2 $t0 $s0
sltu $t3 $t0 $s0
sw $t1 572($0)
sw $t3 576($0)
sw $t2 580($0)
sra $t0 $s0 4
slti $t1 $t0 4
ori $t2 $t0 1
sltiu $t3 $t0 2
sw $t2 584($0)
sw $t1 588($0)
sw $t3 592($0)
addi $t0 $0 4
addi $t1 $0 256
sll $t2 $t0 2
lh $t3 0($t2)
lb $t4 4($t2)
lh $t5 8($t2)
sw $t3 596($0)
sw $t4 600($0)
sw $t5 604($0)
srl $t2 $t1 6
lb $t3 0($t2)
lw $t4 4($t2)
lb $t5 8($t2)
sw $t3 608($0)
sw $t4 612($0)
sw $t5 616($0)
sllv $t0 $s0 $s1
sllv $t1 $s2 $s3
divu $t1 $t0
and $t2 $s0 $s1
subu $t3 $s2 $s3
sltu $t4 $s0 $s1
mfhi $t5
xor $t1 $t5 $s3
or $t6 $t5 $s1
andi $t0 $t5 -100
divu $t1 $t0
sw $t3 620($0)
sw $t4 624($0)
sw $t2 628($0)
sw $t6 632($0)
mflo $t5
xori $t7 $t5 4
xori $t8 $t5 4
addu $t9 $t5 $s1
andi $t0 $s2 4
addiu $t1 $s2 4
sltiu $t2 $s2 4
divu $t0 $t1
sw $t7 636($0)
sw $t8 640($0)
sw $t9 644($0)
mflo $t5
sw $t5 648($0)
sw $t5 652($0)
sw $t5 656($0)
lui $t0 0xffff
add $t0 $t0 $s6
sw $t0 660($0)
lb $t0 660($0)
lw $t1 660($0)
multu $t1 $t0
sllv $t2 $s0 $s1
subu $t3 $s2 $s3
or $t4 $s0 $s1
mfhi $t5
```

```
lbu $t0 660($0)
sll $t6 $t5 3
sll $t1 $t5 4
multu $t1 $t0
sw $t3 660($0)
sw $t4 664($0)
sw $t2 668($0)
sw $t6 672($0)
mflo $t5
sll $t7 $t5 4
xori $t8 $t5 7
nor $t9 $t5 $s1
sll $t1 $s0 2
sll $t0 $s0 3
xori $t2 $s2 4
divu $t0 $t1
sw $t7 676($0)
sw $t8 680($0)
sw $t9 684($0)
mflo $t5
sw $t5 688($0)
sw $t5 692($0)
sw $t5 696($0)
nor $t0 $s0 $s1
nop
nop
mthi $t0
mfhi $t1
subu $t0 $s0 $s1
nop
mtlo $t0
mflo $t2
sllv $t0 $s0 $s1
mthi $t0
mfhi $t3
sw $t1 700($0)
sw $t2 704($0)
sw $t3 708($0)
xori $t0 $s1 177
nop
nop
mthi $t0
mfhi $t1
ori $t0 $s1 320
nop
mtlo $t0
mflo $t2
sltiu $t0 $s1 559
mthi $t0
mfhi $t3
sw $t1 712($0)
sw $t2 716($0)
sw $t3 720($0)
srl $t0 $s0 0
nop
nop
mthi $t0
mfhi $t1
sll $t0 $s0 2
nop
mtlo $t0
mflo $t2
sra $t0 $s0 0
mthi $t0
mfhi $t3
sw $t1 724($0)
sw $t2 728($0)
sw $t3 732($0)
lhu $t0 0($0)
```

```
nop
nop
mthi $t0
mfhi $t1
lhu $t0 0($0)
nop
mtlo $t0
mflo $t2
lb $t0 0($0)
mthi $t0
mfhi $t3
sw $t1 736($0)
sw $t2 740($0)
sw $t3 744($0)
addi $t0 $0 4
mthi $t0
mfhi $t5
nop
nop
lhu $t1 0($t5)
addi $t0 $0 0
mthi $t0
mfhi $t5
nop
lb $t2 0($t5)
addi $t0 $0 8
mthi $t0
mfhi $t5
lbu $t3 0($t5)
sw $t1 748($0)
sw $t2 752($0)
sw $t3 756($0)
lui $t0 204
and $t1 $s0 $t0
sllv $t2 $s0 $t0
xor $t3 $s0 $t0
sw $t3 760($0)
sw $t2 764($0)
sw $t1 768($0)
lui $t0 361
sltu $t1 $t0 $s0
slt $t2 $t0 $s0
or $t3 $t0 $s0
sw $t1 772($0)
sw $t3 776($0)
sw $t2 780($0)
lui $t0 662
xori $t1 $t0 2
andi $t2 $t0 17
slti $t3 $t0 1
sw $t1 784($0)
sw $t3 788($0)
sw $t2 792($0)
lui $t0 779
sw $t0 796($0)
sw $t0 800($0)
sw $t0 804($0)

and $t1 $s0 $t0
and $t2 $s0 $t0
sllv $t3 $s0 $t0
sw $t3 808($0)
sw $t2 812($0)
sw $t1 816($0)
srav $t1 $t0 $s0
xor $t2 $t0 $s0
and $t3 $t0 $s0
sw $t1 820($0)
sw $t3 824($0)
```

```
sw $t2 828($0)
xori $t1 $t0 46
addiu $t2 $t0 76
sltiu $t3 $t0 48
sw $t3 832($0)
sw $t2 836($0)
sw $t1 840($0)
sllv $t0 $0 $0
lbu $t1 0($t0)
lh $t2 4($t0)
lbu $t3 8($t0)
sw $t3 844($0)
sb $t2 848($0)
sh $t1 852($0)
sltiu $t0 $0 0
lw $t1 0($t0)
lhu $t2 4($t0)
lb $t3 8($t0)
sb $t3 856($0)
sw $t2 860($0)
sb $t1 864($0)
andi $t0 $s1 486
sw $t3 868($0)
sw $t2 872($0)
sw $t1 876($0)
andi $t0 $s1 768
sw $t1 880($0)
sw $t3 884($0)
sw $t2 888($0)
lh $t0 92($0)
sw $t3 892($0)
sw $t2 896($0)
sw $t1 900($0)
lh $t0 792($0)
sw $t1 904($0)
sw $t3 908($0)
sw $t2 912($0)
sll $t1 $t0 2
sra $t2 $t0 1
sra $t3 $t0 3
sw $t3 916($0)
sw $t2 920($0)
sw $t1 924($0)
sra $t0 $s0 4
sw $t3 928($0)
sw $t2 932($0)
sw $t1 936($0)
sra $t0 $s0 1
sw $t1 940($0)
sw $t3 944($0)
sw $t2 948($0)
or $t0 $s0 $s1
or $t1 $s2 $s3
divu $t1 $t0
mfhi $t5
andi $t0 $t5 -100
multu $t1 $t0
sw $t3 952($0)
sw $t4 956($0)
sw $t2 960($0)
sw $t6 964($0)
mfhi $t5
ori $t7 $t5 4
xori $t8 $t5 4
andi $t0 $s2 4
ori $t1 $s2 4
slti $t2 $s2 4
multu $t0 $t1
sw $t7 968($0)
```

```
sw $t8 972($0)
sw $t9 976($0)
mflo $t5
sw $t5 980($0)
sw $t5 984($0)
sw $t5 988($0)
nop
nop
mthi $t0
mfhi $t1
nop
mtlo $t0
mflo $t2
mthi $t0
mfhi $t3
sw $t1 992($0)
sw $t2 996($0)
sw $t3 1000($0)
ori $t0 $s1 832
nop
nop
mthi $t0
mfhi $t1
sltiu $t0 $s1 918
nop
mtlo $t0
mflo $t2
addiu $t0 $s1 896
mthi $t0
mfhi $t3
sw $t1 1004($0)
sw $t2 1008($0)
sw $t3 1012($0)
srl $t0 $s0 0
nop
nop
mthi $t0
mfhi $t1
srl $t0 $s0 2
nop
mtlo $t0
mflo $t2
sll $t0 $s0 1
mthi $t0
mfhi $t3
sw $t1 1016($0)
sw $t2 1020($0)
sw $t3 1024($0)
lhu $t0 0($0)
nop
nop
mthi $t0
mfhi $t1
lhu $t0 0($0)
nop
mtlo $t0
mflo $t2
lw $t0 0($0)
mthi $t0
mfhi $t3
sw $t1 1028($0)
sw $t2 1032($0)
sw $t3 1036($0)
addi $t0 $0 4
mthi $t0
mfhi $t5
nop
nop
lbu $t1 0($t5)
```

```
addi $t0 $0 0
mthi $t0
mfhi $t5
nop
lbu $t2 0($t5)
addi $t0 $0 8
mthi $t0
mfhi $t5
lbu $t3 0($t5)
sw $t1 1040($0)
sw $t2 1044($0)
sw $t3 1048($0)

and $t0 $s0 $s1
sw $t1 1152($0)
sw $t3 1156($0)
sw $t2 1160($0)

add $t0 $s0 $s1
sw $t1 1164($0)
sw $t3 1168($0)
sw $t2 1172($0)

sub $t0 $s0 $s1
sw $t1 1176($0)
sw $t3 1180($0)
sw $t2 1184($0)

sllv $t0 $s0 $s1
sw $t1 1188($0)
sw $t3 1192($0)
sw $t2 1196($0)

srlv $t0 $s0 $s1
sw $t1 1200($0)
sw $t3 1204($0)
sw $t2 1208($0)

srav $t0 $s0 $s1
sw $t1 1212($0)
sw $t3 1216($0)
sw $t2 1220($0)

xor $t0 $s0 $s1
sw $t1 1224($0)
sw $t3 1228($0)
sw $t2 1232($0)

nor $t0 $s0 $s1
sw $t1 1236($0)
sw $t3 1240($0)
sw $t2 1244($0)

slt $t0 $s0 $s1

sw $t1 1248($0)
sw $t3 1252($0)
sw $t2 1256($0)
end: j end
nop
$ 1 <= 00010000
$ 1 <= 0001ffff
$16 <= 0001ffff
$17 <= 00000005
$ 1 <= 00000000
$ 1 <= 0000ffff
$18 <= 0000ffff
$19 <= 00000008
$21 <= abcd0000
```

```
$ 1 <= 00000000
$ 1 <= 0000ffff
$22 <= 0000ffff
$ 8 <= 00020004
$ 9 <= 00040003
$10 <= 001ffff0
$11 <= 00000004
*00000000 <= 00000004
*00000004 <= 001ffff0
*00000008 <= 00040003
$ 8 <= 00000fff
$ 9 <= 00000000
$10 <= 00000000
$11 <= 00020ffe
*0000000c <= 00000000
*00000010 <= 00020ffe
*00000014 <= 00000000
$ 8 <= 0001fffa
$ 9 <= 0001fffb
$10 <= 0001ffff
$11 <= 00000000
*00000018 <= 00000000
*0000001c <= 0001ffff
*00000020 <= 0001fffb
$ 8 <= 00000069
$ 8 <= 00000000
$ 9 <= 00000004
$10 <= fffffff0
$11 <= 00000003
*00000024 <= 00000003
*00000028 <= fff0
*0000002c <= 00000004
$ 8 <= 00000069
$ 8 <= 00000000
$ 9 <= 00000004
$10 <= 0000fff0
$11 <= 00000003
*00000030 <= 03
*00000034 <= fff0
*00000038 <= 0004
$ 8 <= 000003e0
$ 9 <= 0001fc1f
$10 <= 0001ffff
$11 <= fffe0000
*0000003c <= fffe0000
*00000040 <= 0001ffff
*00000044 <= 0001fc1f
$ 8 <= 00000272
$ 9 <= 0001ffff
$10 <= fffe0000
$11 <= fffe0273
*00000048 <= 0001ffff
*0000004c <= fffe0273
*00000050 <= fffe0000
$ 8 <= 00000005
$ 9 <= 00000007
$10 <= 00000001
$11 <= 00000000
*00000054 <= 00000001
*00000058 <= 00000007
*0000005c <= 00000000
$ 8 <= 00000000
$ 9 <= 00000000
$10 <= 00000000
$11 <= 0001ffff
*00000060 <= 0001ffff
*00000064 <= 00000000
*00000068 <= 00000000
$ 8 <= 00000003
```

```
$ 9 <= 80000000
$10 <= fffe0004
$11 <= 0001fffc
*0000006c <= 80000000
*00000070 <= 0001fffc
*00000074 <= fffe0004
$ 8 <= 000000ff
$ 9 <= 0000009b
$10 <= 00000000
$11 <= 00000000
*00000078 <= 00000000
*0000007c <= 00000000
*00000080 <= 0000009b
$ 8 <= 00000004
*00000084 <= 00000004
$ 8 <= 00000004
$ 9 <= 000000f0
$10 <= 00000003
$11 <= 00000000
*00000084 <= 0000
*00000088 <= 00000003
*0000008c <= 00f0
$ 8 <= fffe0000
$ 9 <= fffe0000
$10 <= fffc0000
$11 <= fffc000
*00000090 <= fffc000
*00000094 <= fffc0000
*00000098 <= fffe0000
$ 8 <= 00000000
$ 9 <= 00000005
$10 <= 00000001
$11 <= 00000001
*0000009c <= 00000001
*000000a0 <= 00000005
*000000a4 <= 00000001
$ 8 <= 00000004
$ 9 <= 00000002
$10 <= 00000004
$11 <= 00000000
*000000a8 <= 00000000
*000000ac <= 00000004
*000000b0 <= 00000002
$ 8 <= 0001ffff
$ 9 <= 00000000
$10 <= 0003fffe
$11 <= 00000000
*000000b4 <= 00000000
*000000b8 <= 0003fffe
*000000bc <= 00000000
$ 8 <= 00007fff
$ 9 <= 0001ffff
$10 <= 00027ffe
$11 <= fffe8000
*000000c0 <= 0001ffff
*000000c4 <= fffe8000
*000000c8 <= 00027ffe
$ 8 <= 001ffff0
$ 9 <= 001ffff9
$10 <= 001ffff9
$11 <= 00000000
*000000cc <= 001ffff9
*000000d0 <= 001ffff9
*000000d4 <= 00000000
$ 8 <= 00000004
$ 9 <= 00000100
$10 <= 00000010
$11 <= 00000ffe
$12 <= 00000000
```

```
$13 <= 00000000
*000000d8 <= 00000ffe
*000000dc <= 00000000
*000000e0 <= 00000000
$10 <= 00000004
$11 <= fffffff0
$12 <= 00040003
$13 <= 00000000
*000000e4 <= fffffff0
*000000e8 <= 00040003
*000000ec <= 00000000
$ 8 <= 00000fff
$ 9 <= 000000ff
$10 <= 0001fffa
$11 <= 00ffff00
$12 <= 0001ffff
$13 <= 00000000
$ 9 <= 00000008
$14 <= 00000000
$ 1 <= ffff0000
$ 1 <= ffffff9c
$ 8 <= ffffff9c
*000000f0 <= 00ffff00
*000000f4 <= 0001ffff
*000000f8 <= 0001fffa
*000000fc <= 00000000
$13 <= ffffffff
$15 <= 00000004
$24 <= ffffffff
$25 <= 00000005
$ 8 <= 0000fffb
$ 9 <= 00010003
$10 <= 0000ffff
*00000100 <= 00000004
*00000104 <= ffffffff
*00000108 <= 00000005
$13 <= 00000000
*0000010c <= 00000000
*00000110 <= 00000000
*00000114 <= 00000000
$ 8 <= ffff0000
$ 8 <= ffffffff
*00000118 <= ffffffff
$ 8 <= ffffffff
$ 9 <= 0000ffff
$10 <= 00000fff
$11 <= 00ffff00
$12 <= 00000fff
$13 <= ffff0001
$ 8 <= ffffffff
$14 <= fffe000
$ 9 <= 0fff000
*00000118 <= 00ffff00
*0000011c <= 00000fff
*00000120 <= 00000fff
*00000124 <= fffe000
$13 <= f0001000
$15 <= 00010000
$24 <= 00000000
$25 <= f0001005
$ 9 <= 0007fffc
$ 8 <= 000ffff8
$10 <= 0000fffb
*00000128 <= 00010000
*0000012c <= 00000000
*00000130 <= f0001005
$13 <= ff800020
*00000134 <= ff800020
*00000138 <= ff800020
```

```
*0000013c <= ff800020
$ 8 <= 00000000
$ 9 <= 00000000
$ 8 <= 00000fff
$10 <= 00000fff
$ 8 <= 00000005
$11 <= 00000005
*00000140 <= 00000000
*00000144 <= 00000fff
*00000148 <= 00000005
$ 8 <= 00000001
$ 9 <= 00000001
$ 8 <= 00000004
$10 <= 00000004
$ 8 <= 0000039f
$11 <= 0000039f
*0000014c <= 00000001
*00000150 <= 00000004
*00000154 <= 0000039f
$ 8 <= 00007fff
$ 9 <= 00007fff
$ 8 <= 00001fff
$10 <= 00001fff
$ 8 <= 0000ffff
$11 <= 0000ffff
*00000158 <= 00007fff
*0000015c <= 00001fff
*00000160 <= 0000ffff
$ 8 <= 00000004
$ 9 <= 00000004
$ 8 <= 00000004
$10 <= 00000004
$ 8 <= 00000004
$11 <= 00000004
*00000164 <= 00000004
*00000168 <= 00000004
*0000016c <= 00000004
$ 8 <= 00000004
$13 <= 00000004
$ 9 <= 000000f0
$ 8 <= 00000000
$13 <= 00000000
$10 <= 00000004
$ 8 <= 00000008
$13 <= 00000008
$11 <= 00000003
*00000170 <= 000000f0
*00000174 <= 00000004
*00000178 <= 00000003
$ 8 <= 00020004
$ 9 <= 00001fff
$10 <= 00001fff
$11 <= 00000001
*0000017c <= 00000001
*00000180 <= 00001fff
*00000184 <= 00001fff
$ 8 <= 00020004
$ 9 <= 00000000
$10 <= fffc0000
$11 <= 00000004
*00000188 <= 00000000
*0000018c <= 00000004
*00000190 <= fffc0000
$ 8 <= 0001ffff
$ 9 <= 00000000
$10 <= 00000000
$11 <= 00000000
*00000194 <= 00000000
*00000198 <= 00000000
```

```
*0000019c <= 00000000
$ 8 <= 00000069
$ 8 <= 00000000
$ 9 <= 00000004
$10 <= 001ffff0
$11 <= 00040003
*000001a0 <= 0003
*000001a4 <= 001ffff0
*000001a8 <= 00000004
$ 8 <= 00000069
$ 8 <= 00000000
$ 9 <= 00000004
$10 <= fffffff0
$11 <= 00000003
*000001ac <= 0003
*000001b0 <= f0
*000001b4 <= 0004
$ 8 <= 000001e6
$ 9 <= fffe0000
$10 <= 00000000
$11 <= 000001e6
*000001b8 <= 000001e6
*000001bc <= 00000000
*000001c0 <= fffe0000
$ 8 <= 0000035c
$ 9 <= 0002035b
$10 <= 00000000
$11 <= 00000000
*000001c4 <= 0002035b
*000001c8 <= 00000000
*000001cc <= 00000000
$ 8 <= 00000007
$ 9 <= 00000000
$10 <= 00000000
$11 <= 00000000
*000001d0 <= 00000000
*000001d4 <= 00000000
*000001d8 <= 00000000
$ 8 <= 00000004
$ 9 <= 0001fffb
$10 <= 00000004
$11 <= 00001fff
*000001dc <= 00001fff
*000001e0 <= 00000004
*000001e4 <= 0001fffb
$ 8 <= 00001fff
$ 9 <= 00000001
$10 <= fffe2000
$11 <= 80000000
*000001e8 <= 00000001
*000001ec <= 80000000
*000001f0 <= fffe2000
$ 8 <= 00000000
$ 9 <= 00000064
$10 <= 00000001
$11 <= 00000064
*000001f4 <= 00000064
*000001f8 <= 00000001
*000001fc <= 00000064
$ 8 <= 00000004
*00000200 <= 00000004
$ 8 <= 00000004
$ 9 <= 000000f0
$10 <= 00040003
$11 <= 00000000
*00000200 <= 00
*00000204 <= 00040003
*00000208 <= f0
$ 8 <= 003fffe0
```

```
$ 9 <= 001ffff0
$10 <= 007fffc0
$11 <= 003fffe0
*0000020c <= 003fffe0
*00000210 <= 007fffc0
*00000214 <= 001ffff0
$ 8 <= 00000000
$ 9 <= 00000005
$10 <= 00000000
$11 <= 00000005
*00000218 <= 00000000
*0000021c <= 00000005
*00000220 <= 00000005
$ 8 <= 00000004
$ 9 <= 00000020
$10 <= 00000004
$11 <= 00000004
*00000224 <= 00000004
*00000228 <= 00000004
*0000022c <= 00000020
$ 8 <= 00003fff
$ 9 <= 0001ffff
$10 <= 0001c000
$11 <= 00000000
*00000230 <= 00000000
*00000234 <= 0001c000
*00000238 <= 0001ffff
$ 8 <= 00007fff
$ 9 <= 00000001
$10 <= 00000000
$11 <= 00000001
*0000023c <= 00000001
*00000240 <= 00000001
*00000244 <= 00000000
$ 8 <= 00001fff
$ 9 <= 00000000
$10 <= 00001fff
$11 <= 00000000
*00000248 <= 00001fff
*0000024c <= 00000000
*00000250 <= 00000000
$ 8 <= 00000004
$ 9 <= 00000100
$10 <= 00000010
$11 <= 00000ffe
$12 <= 00000000
$13 <= 00000000
*00000254 <= 00000ffe
*00000258 <= 00000000
*0000025c <= 00000000
$10 <= 00000004
$11 <= fffffff0
$12 <= 00040003
$13 <= 00000000
*00000260 <= fffffff0
*00000264 <= 00040003
*00000268 <= 00000000
$ 8 <= 003fffe0
$ 9 <= 00ffff00
$10 <= 00000005
$11 <= 0000fff7
$12 <= 00000000
$13 <= 003fff60
$ 9 <= 003fff68
$14 <= 003fff65
$ 1 <= ffff0000
$ 1 <= ffffff9c
$ 8 <= 003fff00
*0000026c <= 0000fff7
```

```
*00000270 <= 00000000
*00000274 <= 00000005
*00000278 <= 003fff65
$13 <= 00000001
$15 <= 00000005
$24 <= 00000005
$25 <= 00000006
$ 8 <= 00000004
$ 9 <= 00010003
$10 <= 00000000
*0000027c <= 00000005
*00000280 <= 00000005
*00000284 <= 00000006
$13 <= 00000000
*00000288 <= 00000000
*0000028c <= 00000000
*00000290 <= 00000000
$ 8 <= ffff0000
$ 8 <= ffffffff
*00000294 <= ffffffff
$ 8 <= ffffffff
$ 9 <= ffffffff
$10 <= 003fffe0
$11 <= 0000fff7
$12 <= 0001ffff
$13 <= fffffffe
$ 8 <= 000000ff
$14 <= fffffff0
$ 9 <= ffffffe0
*00000294 <= 0000fff7
*00000298 <= 0001ffff
*0000029c <= 003fffe0
*000002a0 <= fffffff0
$13 <= ffffe020
$15 <= fffe0200
$24 <= ffffe027
$25 <= 00001fda
$ 9 <= 0007fffc
$ 8 <= 000ffff8
$10 <= 0000fffb
*000002a4 <= fffe0200
*000002a8 <= ffffe027
*000002ac <= 00001fda
$13 <= 00000002
*000002b0 <= 00000002
*000002b4 <= 00000002
*000002b8 <= 00000002
$ 8 <= fffe0000
$ 9 <= fffe0000
$ 8 <= 0001fffa
$10 <= 0001fffa
$ 8 <= 003fffe0
$11 <= 003fffe0
*000002bc <= fffe0000
*000002c0 <= 0001fffa
*000002c4 <= 003fffe0
$ 8 <= 000000b4
$ 9 <= 000000b4
$ 8 <= 00000145
$10 <= 00000145
$ 8 <= 00000001
$11 <= 00000001
*000002c8 <= 000000b4
*000002cc <= 00000145
*000002d0 <= 00000001
$ 8 <= 0001ffff
$ 9 <= 0001ffff
$ 8 <= 0007fffc
$10 <= 0007fffc
```

```
$ 8 <= 0001ffff
$11 <= 0001ffff
*000002d4 <= 0001ffff
*000002d8 <= 0007fffc
*000002dc <= 0001ffff
$ 8 <= 00000004
$ 9 <= 00000004
$ 8 <= 00000004
$10 <= 00000004
$ 8 <= 00000004
$11 <= 00000004
*000002e0 <= 00000004
*000002e4 <= 00000004
*000002e8 <= 00000004
$ 8 <= 00000004
$13 <= 00000004
$ 9 <= 0000fff0
$ 8 <= 00000000
$13 <= 00000000
$10 <= 00000004
$ 8 <= 00000008
$13 <= 00000008
$11 <= 00000003
*000002ec <= 0000fff0
*000002f0 <= 00000004
*000002f4 <= 00000003
$ 8 <= 00cc0000
$ 9 <= 00000000
$10 <= 0001ffff
$11 <= 00cdffff
*000002f8 <= 00cdffff
*000002fc <= 0001ffff
*00000300 <= 00000000
$ 8 <= 01690000
$ 9 <= 00000000
$10 <= 00000000
$11 <= 0169ffff
*00000304 <= 00000000
*00000308 <= 0169ffff
*0000030c <= 00000000
$ 8 <= 02960000
$ 9 <= 02960002
$10 <= 00000000
$11 <= 00000000
*00000310 <= 02960002
*00000314 <= 00000000
*00000318 <= 00000000
$ 8 <= 030b0000
*0000031c <= 030b0000
*00000320 <= 030b0000
*00000324 <= 030b0000
$ 9 <= 00010000
$10 <= 00010000
$11 <= 0001ffff
*00000328 <= 0001ffff
*0000032c <= 00010000
*00000330 <= 00010000
$ 9 <= 00000000
$10 <= 030affff
$11 <= 00010000
*00000334 <= 00000000
*00000338 <= 00010000
*0000033c <= 030affff
$ 9 <= 030b002e
$10 <= 030b004c
$11 <= 00000000
*00000340 <= 00000000
*00000344 <= 030b004c
*00000348 <= 030b002e
```

```
$ 8 <= 00000000
$ 9 <= 00000004
$10 <= fffffff0
$11 <= 00000003
*0000034c <= 00000003
*00000350 <= f0
*00000354 <= 0004
$ 8 <= 00000000
$ 9 <= 00000004
$10 <= 0000fff0
$11 <= 00000003
*00000358 <= 03
*0000035c <= 0000fff0
*00000360 <= 04
$ 8 <= 00000004
*00000364 <= 00000003
*00000368 <= 0000fff0
*0000036c <= 00000004
$ 8 <= 00000000
*00000370 <= 00000004
*00000374 <= 00000003
*00000378 <= 0000fff0
$ 8 <= 00000000
*0000037c <= 00000003
*00000380 <= 0000fff0
*00000384 <= 00000004
$ 8 <= 00000000
*00000388 <= 00000004
*0000038c <= 00000003
*00000390 <= 0000fff0
$ 9 <= 00000000
$10 <= 00000000
$11 <= 00000000
*00000394 <= 00000000
*00000398 <= 00000000
*0000039c <= 00000000
$ 8 <= 00001fff
*000003a0 <= 00000000
*000003a4 <= 00000000
*000003a8 <= 00000000
$ 8 <= 0000ffff
*000003ac <= 00000000
*000003b0 <= 00000000
*000003b4 <= 00000000
$ 8 <= 0001ffff
$ 9 <= 0000ffff
$13 <= 0000ffff
$ 1 <= ffff0000
$ 1 <= ffffff9c
$ 8 <= 0000ff9c
*000003b8 <= 00000000
*000003bc <= 0001ffff
*000003c0 <= 00000000
*000003c4 <= fffffff0
$13 <= 00000000
$15 <= 00000004
$24 <= 00000004
$ 8 <= 00000004
$ 9 <= 0000ffff
$10 <= 00000000
*000003c8 <= 00000004
*000003cc <= 00000004
*000003d0 <= 00001fda
$13 <= 0003fffc
*000003d4 <= 0003fffc
*000003d8 <= 0003fffc
*000003dc <= 0003fffc
$ 9 <= 00000004
$10 <= 00000004
```

```
$11 <= 00000004
*000003e0 <= 00000004
*000003e4 <= 00000004
*000003e8 <= 00000004
$ 8 <= 00000345
$ 9 <= 00000345
$ 8 <= 00000001
$10 <= 00000001
$ 8 <= 00000385
$11 <= 00000385
*000003ec <= 00000345
*000003f0 <= 00000001
*000003f4 <= 00000385
$ 8 <= 0001ffff
$ 9 <= 0001ffff
$ 8 <= 00007fff
$10 <= 00007fff
$ 8 <= 0003fffe
$11 <= 0003fffe
*000003f8 <= 0001ffff
*000003fc <= 00007fff
*00000400 <= 0003fffe
$ 8 <= 00000004
$ 9 <= 00000004
$ 8 <= 00000004
$10 <= 00000004
$ 8 <= 00000004
$11 <= 00000004
*00000404 <= 00000004
*00000408 <= 00000004
*0000040c <= 00000004
$ 8 <= 00000004
$13 <= 00000004
$ 9 <= 000000f0
$ 8 <= 00000000
$13 <= 00000000
$10 <= 00000004
$ 8 <= 00000008
$13 <= 00000008
$11 <= 00000003
*00000410 <= 000000f0
*00000414 <= 00000004
*00000418 <= 00000003
$ 8 <= 00000005
*00000480 <= 000000f0
*00000484 <= 00000003
*00000488 <= 00000004
$ 8 <= 00020004
*0000048c <= 000000f0
*00000490 <= 00000003
*00000494 <= 00000004
$ 8 <= 0001fffa
*00000498 <= 000000f0
*0000049c <= 00000003
*000004a0 <= 00000004
$ 8 <= 003fffe0
*000004a4 <= 000000f0
*000004a8 <= 00000003
*000004ac <= 00000004
$ 8 <= 00000fff
*000004b0 <= 000000f0
*000004b4 <= 00000003
*000004b8 <= 00000004
$ 8 <= 00000fff
*000004bc <= 000000f0
*000004c0 <= 00000003
*000004c4 <= 00000004
$ 8 <= 0001fffa
*000004c8 <= 000000f0
```

```
*000004cc <= 00000003
*000004d0 <= 00000004
$ 8 <= fffe0000
*000004d4 <= 000000f0
*000004d8 <= 00000003
*000004dc <= 00000004
$ 8 <= 00000000
*000004e0 <= 000000f0
*000004e4 <= 00000003
*000004e8 <= 00000004
```

三、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？

我认为ALU只是组合电路，而乘除法部件是时序部件需要多个周期，因此最好分开来。不能占着ALU。（乘除法部件正busy的时候还能用ALU维持其他不冲突指令的运算）

2. 参照你对延迟槽的理解，试解释“乘除槽”。

我认为延迟槽是在我们已尽力最早得到跳转结果时候（ID）PC已计数到原PC+4了。下一个时钟上升沿到来时它所代表的指令就会进入IF/ID。为了不清空流水线使得流水线冒泡，利用好这一个时间片，就将跳转指令的下一条执行。

所以我认为乘除槽是指当乘除法部件在乘除法运算周期中，阻塞了乘除相关的指令，但对于不冲突不要暂停的指令继续执行。节约了时间，加大CPU的CPI

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？

不知道。。

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）

   char a[max] 按字节访问内存更好
   int b[max]按字访问内存更好

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？

我的设计是上课的工程化方法，即文中的planner型，提前划分指令类型，考虑他们要用到的寄存器来涉及转发。
我列了表格分析冲突指令所在的流水级，并把指令分成几个大类：cal_r(regreg\regshamnt\reghl)\cal_i\B\load\store\muldiv\mt\j\jal\jalr\jr
根据Tnew Tuse来设计转发和暂停

6. 你对流水线CPU设计风格有何见解？

我觉得自己写的这种方法是依靠excel表不会漏考虑情况，但转发条件的代码非常冗长。而且我觉得detector型更贴近转发的本质。

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。(非常重要)