CPU 设计文档

一、数据通路

（见附表）

二、控制器

1. 主控制器

	Reg Dst	ALU Src	Memto Reg	Reg Write	Mem write	EXT Op	ALU ctr	nPC_ sel	DEXT _Op	BE Op	start	HI_ write	LO_ write	m_ or_ d
LB	00	1	001	1	0	01	0	0	2	X	0	0	0	X
LBU	00	1	001	1	0	01	0	0	1	X	0	0	0	X
LH	00	1	001	1	0	01	0	0	4	X	0	0	0	X
LHU	00	1	001	1	0	01	0	0	3	X	0	0	0	X
LW	00	1	001	1	0	01	0	0	0	X	0	0	0	X
SB	X	1	X	0	1	01	0	0	X	2	0	0	0	X
SH	X	1	X	0	1	01	0	0	X	1	0	0	0	X
SW	X	1	X	0	1	01	0	0	X	0	0	0	0	X
ADD	01	0	000	1	0	X	0	0	X	X	0	0	0	X
ADDU	01	0	000	1	0	X	0	0	X	X	0	0	0	X
SUB	01	0	000	1	0	X	1	0	X	X	0	0	0	X
SUBU	01	0	000	1	0	X	1	0	X	X	0	0	0	X
SLT	01	0	000	1	0	X	2	0	X	X	0	0	0	X
SLTU	01	0	000	1	0	X	3	0	X	X	0	0	0	X
SLL	01	0	000	1	0	X	4	0	X	X	0	0	0	X
SRL	01	0	000	1	0	X	5	0	X	X	0	0	0	X
SRA	01	0	000	1	0	X	6	0	X	X	0	0	0	X
SLLV	01	0	000	1	0	X	7	0	X	X	0	0	0	X
SRLV	01	0	000	1	0	X	8	0	X	X	0	0	0	X
SRAV	01	0	000	1	0	X	9	0	X	X	0	0	0	X
AND	01	0	000	1	0	X	10	0	X	X	0	0	0	X
OR	01	0	000	1	0	X	11	0	X	X	0	0	0	X
XOR	01	0	000	1	0	X	11	0	X	X	0	0	0	X
NOR	01	0	000	1	0	X	12	0	X	X	0	0	0	X
ADDI	00	1	000	1	0	01	0	0	X	X	0	0	0	X
ADDIU	00	1	000	1	0	01	0	0	X	X	0	0	0	X
ANDI	00	1	000	1	0	00	10	0	X	X	0	0	0	X
ORI	00	1	000	1	0	00	11	0	X	X	0	0	0	X
XORI	00	1	000	1	0	00	12	0	X	X	0	0	0	X
LUI	00	1	000	1	0	10	0	0	X	X	0	0	0	X

SLTI	00	1	000	1	0	01	2	0	X	X	0	0	0	X
SLTIU	00	1	000	1	0	01	3	0	X	X	0	0	0	X
BEQ	X	X	X	0	0	X	X	1	X	X	0	0	0	X
BNE	X	X	X	0	0	X	X	2	X	X	0	0	0	X
BLEZ	X	X	X	0	0	X	X	3	X	X	0	0	0	X
BGTZ	X	X	X	0	0	X	X	4	X	X	0	0	0	X
BLTZ	X	X	X	0	0	X	X	5	X	X	0	0	0	X
BGEZ	X	X	X	0	0	X	X	6	X	X	0	0	0	X
J	X	X	X	0	0	X	X	7	X	X	0	0	0	X
JAL	10	X	010	1	0	X	X	7	X	X	0	0	0	X
JALR	01	X	010	1	0	X	X	8	X	X	0	0	0	X
JR	X	X	X	0	0	X	X	8	X	X	0	0	0	X
MULT	X	X	X	0	0	X	X	0	X	X	1	0	0	00
MULTU	X	X	X	0	0	X	X	0	X	X	1	0	0	01
DIV	X	X	X	0	0	X	X	0	X	X	1	0	0	10
DIVU	X	X	X	0	0	X	X	0	X	X	1	0	0	11
MFHI	01	X	100	1	0	X	X	0	X	X	0	0	0	X
MFLO	01	X	011	1	0	X	X	0	X	X	0	0	0	X
MTHI	X	X	X	0	0	X	X	0	X	X	0	1	0	X
MTLO	X	X	X	0	0	X	X	0	X	X	0	0	1	X

2. 暂停机制

a. Tuse 和 Tnew

IF/ID 当前指令		
指令类型	源寄存器	Tuse
branch1	rs/rt	0
branch2	rs	0
cal_r1	rs/rt	1
cal_r2	rt	1
cal_i	rs	1
load	rs	1
store	rs	1
store	rt	2
jalr/jr	rs	0
mult(u)/div(u)	rs/rt	1
mthi/mtlo	rs	1

ID/EX(Tnew)				EX/MEM(Tnew)				MEM/WB(Tnew)			
cal_r/mfhi/mflo	cal_i	load	jal/jalr	cal_r/mfhi/mflo	cal_i	load	jal/jalr	cal_r/mfhi/mflo	cal_i	load	jal/jalr
1/rd	1/rt	2/rt	0	0/rd	0/rt	1/rt	0	0/rd	0/rt	0/rt	0

b. 构造阻塞矩阵

IF/ID 当前指令			ID/EX(Tnew)			EX/MEM(Tnew)
指令类型	源寄存器	Tuse	cal_r/mf 1/rd	cal_i 1/rt	load 2/rt	load 1/rt
branch1	rs/rt	0	暂停	暂停	暂停	暂停
branch2	rs	0	暂停	暂停	暂停	暂停
cal_r1	rs/rt	1			暂停	
cal_r2	rt	1			暂停	
cal_i	rs	1			暂停	
load	rs	1			暂停	
store	rs	1			暂停	
store	rt	2				
jalr/jr	rs	0	暂停	暂停	暂停	暂停
mult(u)/div(u)	rs/rt	1			暂停	
mthi/mtlo	rs	1			暂停	

3. 转发机制

	流水级	IF/ID		ID/EX		EX/MEM
	源寄存器	rs	rt	rs	rt	rt
	涉及指令	branch1, branch2, jalr/jr	branch1	cal_r1, cal_i, load,store,mult(u), div(u), mthi,mtlo	cal_r1, cal_r2, mult(u), div(u)	store
	转发 MUX	MFRSD	MFRTD	MFRSE	MFRTE	MFRTM
	控制信号	F_RS_D	F_RT_D	F_RS_E	F_RT_E	F_RT_M
	输入 0	RF.RD1	RF.RD2	RS@E	RT@E	RT@M
ID/EX	jal /jalr	PC8@E	PC8@E			
EX/MEM	cal_r 0/rd	AO@M	AO@M	AO@M	AO@M	
	cal_i 0/rt	AO@M	AO@M	AO@M	AO@M	
	mfhi/rd	HI@M	HI@M	HI@M	HI@M	
	mflo/rd	LO@M	LO@M	LO@M	LO@M	
	jal/jalr	PC8@M	PC8@M	PC8@M	PC8@M	
MEM/WB	cal_r 0/rd	MUX_WD	MUX_WD	MUX_WD	MUX_WD	MUX_WD
	cal_i 0/rt	MUX_WD	MUX_WD	MUX_WD	MUX_WD	MUX_WD
	mfhi/rd	MUX_WD	MUX_WD	MUX_WD	MUX_WD	MUX_WD
	mflo/rd	MUX_WD	MUX_WD	MUX_WD	MUX_WD	MUX_WD

	load 0/rt	MUX_WD	MUX_WD	MUX_WD	MUX_WD	MUX_WD
	jal/jalr	MUX_WD	MUX_WD	MUX_WD	MUX_WD	MUX_WD

三、测试程序

1、测试代码

```
ori $1,$0,-100
lui $2,100
add $1,$2,$1
sw $1,0($0)
lw $2,0($0)
addu $3,$2,$0
lw $3,0($0)
subu $4,$0,$3
lh $4,0($0)
add $5,$4,$0
lhu $5,2($0)
sub $6,$0,$5
lb $6,0($0)
and $7,$6,$3
lb $7,1($0)
or $8,$2,$7
lbu $8,2($0)
xor $9,$8,$1
lbu $9,3($0)
nor $10,$1,$9
lbu $11,3($0)
slt $12,$11,$9
lhu $13,0($0)
sltu $14,$12,$13
ori $1,$0,300
add $2,$1,$1
slt $3,$1,$2
sltu $4,$3,$0
ori $5,$0,-80
slt $6,$5,$0
ori $7,$0,-100
sltu $8,$7,$5
slt $8,$7,$5
or $9,$7,$6
xor $10,$9,$9
nor $11,$10,$9
and $12,$11,$1
```

```
sll $13,$7,5
or $14,$13,$0
srl $15,$7,5
or $16,$0,$15
sra $17,$5,5
or $18,$17,$0
ori $19,$0,7
sllv $20,$5,$19
or $21,$20,$0
srlv $22,$5,$19
or $23,$22,$19
srav $24,$5,$19
or $25,$24,$0
jal next1
nop
mult1:ori $1,$0,100
ori $2,$0,-100
mult $1,$2
mflo $3
add $3,$3,$5
jr $5
next1:addu $31,$31,$0
sltu $30,$31,$0
jalr $5,$31
nop
end1:
ori $1,$0,-100
lui $2,64
add $1,$2,$1
sw $1,0($0)
lw $2,0($0)
addi $3,$2,20
lh $3,0($0)
addiu $4,$3,-20
lhu $4,2($0)
ori $5,$4,200
lb $5,0($0)
xori $6,$5,74
lbu $6,0($0)
slti $7,$6,-20
lbu $7,0($0)
sltiu $8,$7,-20
lb $8,0($0)
slti $9,$8,0
```

```
lb $9,0($0)
sltiu $10,$9,0
lui $1,0xffff
addi $1,$1,1000
addiu $2,$1,0
andi $3,$2,0xffff
ori $4,$3,0
xori $5,$4,0xf0f0
ori $6,$0,-50
slti $7,$6,0
ori $6,$0,-90
sltiu $8,$6,0
addu $1,$1,$8
ori $1,$1,0
lw $9,0($0)
nop
slti $9,$9,0
ori $10,$0,-98
sll $11,$10,7
ori $12,$11,0
srl $13,$10,5
ori $14,$13,0
sra $15,$10,3
ori $16,$15,0
ori $17,$0,4
sllv $18,$10,$17
ori $18,$18,0
srlv $19,$10,$17
ori $19,$19,0
srav $20,$10,$17
ori $20,$20,0
jal next2
nop
mult2:ori $21,$0,100
ori $22,$0,-23
divu $21,$22
addi $21,$21,1
mflo $23
subi $23,$23,1
mfhi $24
slti $25,$24,0
jr $5
nop
next2:addiu $31,$31,0
```

```
jalr $5,$31
nop
end2:
ori $1,$0,4
sw $1,0($0)
ori $1,$0,100
sw $1,4($0)
lw $2,0($0)
lw $3,0($2)
lh $4,0($0)
lw $5,0($4)
lhu $6,0($0)
lw $7,0($6)
lb $8,0($0)
lw $9,0($8)
lbu $10,0($0)
lw $11,0($10)
ori $1,$0,4
addu $2,$1,$0
lw $3,0($2)
addi $3,$0,4
lw $4,0($3)
ori $5,$0,1
sll $5,$5,2
lw $6,0($5)
mult3:jal next3
nop
ori $1,$0,4
ori $2,$0,1
multu $1,$2
mflo $3
lw $4,0($3)
jr $5
nop
next3:jalr $5,$31
nop
end3:
ori $1,$0,4
sw $1,0($0)
ori $1,$0,0x5678
lui $2,0x1234
addu $1,$1,$2
sw $1,4($0)
lw $2,0($0)
```

```
sw $1,0($2)
lh $3,0($0)
sh $1,0($3)
lhu $4,0($0)
sh $1,2($4)
lb $5,0($0)
sb $1,0($5)
lb $6,0($0)
sb $1,1($6)
li $30,0x12345678
ori $1,$0,4
addu $2,$0,$1
sw $30,0($2)
addu $30,$30,$1
sh $30,2($0)
li $30,0x12345678
sw $30,4($0)
addi $4,$0,5
sb $30,0($4)
ori $5,$0,1
sll $5,$5,2
sw $30,4($5)
sra $29,$30,2
sw $29,8($5)
mult4:jal next4
nop
sw $5,0($0)
ori $1,$0,8
ori $2,$0,2
div $1,$2
mflo $3
sw $31,0($3)
ori $7,$0,29
ori $9,$0,30
mult $7,$9
mfhi $4
sw $4,4($3)
jr $5
nop
next4:
sw $31,0($0)
jalr $5,$31
nop
end4:
```

```
ori $1,$0,-50
sw $1,0($0)
lw $2,0($0)
sll $3,$2,20
lh $3,0($0)
srl $4,$3,5
lb $4,0($0)
sra $5,$4,5
ori $1,$0,50
sw $1,4($0)
lw $5,4($0)
sra $5,$5,4
ori $1,$0,-50
sll $2,$1,6
addiu $1,$1,0xffff
srl $1,$1,7
lui $1,0xffff
sra $1,$1,9
addiu $1,$1,200
sll $1,$1,5
ori $1,$1,1
sll $1,$1,31
sra $1,$1,20
srl $1,$1,9
jal next5
nop
mult5:
ori $1,$0,0xffff
lui $2,0xffff
mult $1,$2
mfhi $3
sll $3,$3,4
mflo $4
sra $4,$4,7
multu $1,$2
mfhi $3
srl $3,$3,5
mflo $4
sra $4,$4,6
mthi $1
mtlo $2
mfhi $1
sll $1,$1,20
mflo $2
```

```
sra $2,$2,10
jr $5
nop
next5:sll $29,$31,16
sra $28,$29,7
srl $27,$28,9
jalr $5,$31
nop
end5:
ori $1,$0,1
addiu $2,$0,20
for1:beq $1,$2,endfor1
sw $1,0($0)
addiu $1,$1,1
j for1
nop
endfor1:
ori $1,$0,3
addiu $2,$0,3
for2:bne $1,$2,endfor2
sb $2,4($2)
addiu $1,$1,1
mult $2,$2
mflo $2
j for2
nop
endfor2:
ori $2,$0,10
subu $3,$2,$0
for3:blez $3,endfor3
sw $3,0($0)
addiu $2,$2,-1
subu $3,$2,$0
j for3
nop
endfor3:
ori $2,$0,11
subu $3,$0,$2
for4:bgtz $3,endfor4
sw $3,8($0)
addiu $2,$2,-1
subu $3,$0,$2
j for4
nop
```

```
endfor4:
ori $2,$0,10
subu $3,$2,$0
for5:bltz $3,endfor5
sw $3,12($0)
addiu $2,$2,-1
subu $3,$2,$0
j for5
nop
endfor5:
ori $2,$0,11
subu $3,$0,$2
for6:bgez $3,endfor6
sw $3,0($0)
addiu $2,$2,1
subu $3,$2,$0
j for6
nop
endfor6:
ori $2,$0,11
subu $3,$0,$2
nop
endfor7:
ori $2,$1,-100
lui $3,123
mult $2,$3
mfhi $4
mflo $5
multu $2,$3
mfhi $4
mflo $5
div $2,$3
mfhi $4
mflo $5
divu $2,$3
mfhi $4
mflo $5
addi $4,$4,1
mthi $4
mfhi $5
ori $1,$0,0x1234
sw $1,0($0)
lw $2,0($0)
```

2、测试期望

```
@00003000: $ 1 <= ffff0000
@00003004: $ 1 <= ffffff9c
@00003008: $ 1 <= ffffff9c
@0000300c: $ 2 <= 00640000
@00003010: $ 1 <= 0063ff9c
@00003014: *00000000 <= 0063ff9c
@00003018: $ 2 <= 0063ff9c
@0000301c: $ 3 <= 0063ff9c
@00003020: $ 3 <= 0063ff9c
@00003024: $ 4 <= ff9c0064
@00003028: $ 4 <= ffffff9c
@0000302c: $ 5 <= ffffff9c
@00003030: $ 5 <= 00000063
@00003034: $ 6 <= ffffff9d
@00003038: $ 6 <= ffffff9c
@0000303c: $ 7 <= 0063ff9c
@00003040: $ 7 <= ffffffff
@00003044: $ 8 <= ffffffff
@00003048: $ 8 <= 00000063
@0000304c: $ 9 <= 0063ffff
@00003050: $ 9 <= 00000000
@00003054: $10 <= ff9c0063
@00003058: $11 <= 00000000
@0000305c: $12 <= 00000000
@00003060: $13 <= 0000ff9c
@00003064: $14 <= 00000001
@00003068: $ 1 <= 0000012c
@0000306c: $ 2 <= 00000258
@00003070: $ 3 <= 00000001
@00003074: $ 4 <= 00000000
@00003078: $ 1 <= ffff0000
@0000307c: $ 1 <= ffffffb0
@00003080: $ 5 <= ffffffb0
@00003084: $ 6 <= 00000001
@00003088: $ 1 <= ffff0000
@0000308c: $ 1 <= ffffff9c
@00003090: $ 7 <= ffffff9c
@00003094: $ 8 <= 00000001
@00003098: $ 8 <= 00000001
@0000309c: $ 9 <= ffffff9d
@000030a0: $10 <= 00000000
@000030a4: $11 <= 00000062
```

```
@000030a8: $12 <= 00000000
@000030ac: $13 <= fffff380
@000030b0: $14 <= fffff380
@000030b4: $15 <= 07ffffc
@000030b8: $16 <= 07ffffc
@000030bc: $17 <= fffffffd
@000030c0: $18 <= fffffffd
@000030c4: $19 <= 00000007
@000030c8: $20 <= ffffd800
@000030cc: $21 <= ffffd800
@000030d0: $22 <= 01ffffff
@000030d4: $23 <= 01ffffff
@000030d8: $24 <= ffffffff
@000030dc: $25 <= ffffffff
@000030e0: $31 <= 000030e8
@00003108: $31 <= 000030e8
@0000310c: $30 <= 00000000
@00003110: $ 5 <= 00003118
@000030e8: $ 1 <= 00000064
@000030ec: $ 1 <= ffff0000
@000030f0: $ 1 <= ffffff9c
@000030f4: $ 2 <= ffffff9c
@000030fc: $ 3 <= 00002710
@00003100: $ 3 <= 00005828
@00003108: $31 <= 000030e8
@00003118: $ 1 <= ffff0000
@0000311c: $ 1 <= ffffff9c
@00003120: $ 1 <= ffffff9c
@00003124: $ 2 <= 00400000
@00003128: $ 1 <= 003fff9c
@0000312c: *00000000 <= 003fff9c
@00003130: $ 2 <= 003fff9c
@00003134: $ 3 <= 003fffb0
@00003138: $ 3 <= ffffff9c
@0000313c: $ 4 <= ffffff88
@00003140: $ 4 <= 0000003f
@00003144: $ 5 <= 000000ff
@00003148: $ 5 <= ffffff9c
@0000314c: $ 6 <= ffffffd6
@00003150: $ 6 <= 0000009c
@00003154: $ 7 <= 00000000
@00003158: $ 7 <= 0000009c
@0000315c: $ 8 <= 00000001
@00003160: $ 8 <= ffffff9c
```

```
@00003164: $ 9 <= 00000001
@00003168: $ 9 <= ffffff9c
@0000316c: $10 <= 00000000
@00003170: $ 1 <= ffff0000
@00003174: $ 1 <= ffff03e8
@00003178: $ 2 <= ffff03e8
@0000317c: $ 3 <= 000003e8
@00003180: $ 4 <= 000003e8
@00003184: $ 5 <= 0000f318
@00003188: $ 1 <= ffff0000
@0000318c: $ 1 <= ffffffce
@00003190: $ 6 <= ffffffce
@00003194: $ 7 <= 00000001
@00003198: $ 1 <= ffff0000
@0000319c: $ 1 <= ffffffa6
@000031a0: $ 6 <= ffffffa6
@000031a4: $ 8 <= 00000000
@000031a8: $ 1 <= ffffffa6
@000031ac: $ 1 <= ffffffa6
@000031b0: $ 9 <= 003fff9c
@000031b8: $ 9 <= 00000000
@000031bc: $ 1 <= ffff0000
@000031c0: $ 1 <= ffffff9e
@000031c4: $10 <= ffffff9e
@000031c8: $11 <= ffffcf00
@000031cc: $12 <= ffffcf00
@000031d0: $13 <= 07fffffc
@000031d4: $14 <= 07fffffc
@000031d8: $15 <= fffffff3
@000031dc: $16 <= fffffff3
@000031e0: $17 <= 00000004
@000031e4: $18 <= fffff9e0
@000031e8: $18 <= fffff9e0
@000031ec: $19 <= 0ffffff9
@000031f0: $19 <= 0ffffff9
@000031f4: $20 <= fffffff9
@000031f8: $20 <= fffffff9
@000031fc: $31 <= 00003204
@00003238: $31 <= 00003204
@0000323c: $ 5 <= 00003244
@00003204: $21 <= 00000064
@00003208: $ 1 <= ffff0000
@0000320c: $ 1 <= ffffffe9
@00003210: $22 <= ffffffe9
```

```
@00003218: $21 <= 00000065
@0000321c: $23 <= 00000000
@00003220: $ 1 <= 00000001
@00003224: $23 <= ffffffff
@00003228: $24 <= 00000064
@0000322c: $25 <= 00000000
@00003244: $ 1 <= 00000004
@00003248: *00000000 <= 00000004
@0000324c: $ 1 <= 00000064
@00003250: *00000004 <= 00000064
@00003254: $ 2 <= 00000004
@00003258: $ 3 <= 00000064
@0000325c: $ 4 <= 00000004
@00003260: $ 5 <= 00000064
@00003264: $ 6 <= 00000004
@00003268: $ 7 <= 00000064
@0000326c: $ 8 <= 00000004
@00003270: $ 9 <= 00000064
@00003274: $10 <= 00000004
@00003278: $11 <= 00000064
@0000327c: $ 1 <= 00000004
@00003280: $ 2 <= 00000004
@00003284: $ 3 <= 00000064
@00003288: $ 3 <= 00000004
@0000328c: $ 4 <= 00000064
@00003290: $ 5 <= 00000001
@00003294: $ 5 <= 00000004
@00003298: $ 6 <= 00000064
@0000329c: $31 <= 000032a4
@000032c0: $ 5 <= 000032c8
@000032a4: $ 1 <= 00000004
@000032a8: $ 2 <= 00000001
@000032b0: $ 3 <= 00000004
@000032b4: $ 4 <= 00000064
@000032c8: $ 1 <= 00000004
@000032cc: *00000000 <= 00000004
@000032d0: $ 1 <= 00005678
@000032d4: $ 2 <= 12340000
@000032d8: $ 1 <= 12345678
@000032dc: *00000004 <= 12345678
@000032e0: $ 2 <= 00000004
@000032e4: *00000004 <= 12345678
@000032e8: $ 3 <= 00000004
@000032ec: *00000004 <= 12345678
```

```
@000032f0: $ 4 <= 00000004
@000032f4: *00000004 <= 56785678
@000032f8: $ 5 <= 00000004
@000032fc: *00000004 <= 56785678
@00003300: $ 6 <= 00000004
@00003304: *00000004 <= 56787878
@00003308: $ 1 <= 12340000
@0000330c: $30 <= 12345678
@00003310: $ 1 <= 00000004
@00003314: $ 2 <= 00000004
@00003318: *00000004 <= 12345678
@0000331c: $30 <= 1234567c
@00003320: *00000000 <= 567c0004
@00003324: $ 1 <= 12340000
@00003328: $30 <= 12345678
@0000332c: *00000004 <= 12345678
@00003330: $ 4 <= 00000005
@00003334: *00000004 <= 12347878
@00003338: $ 5 <= 00000001
@0000333c: $ 5 <= 00000004
@00003340: *00000008 <= 12345678
@00003344: $29 <= 048d159e
@00003348: *0000000c <= 048d159e
@0000334c: $31 <= 00003354
@00003388: *00000000 <= 00003354
@0000338c: $ 5 <= 00003394
@00003354: *00000000 <= 00003394
@00003358: $ 1 <= 00000008
@0000335c: $ 2 <= 00000002
@00003364: $ 3 <= 00000004
@00003368: *00000004 <= 00003354
@0000336c: $ 7 <= 0000001d
@00003370: $ 9 <= 0000001e
@00003378: $ 4 <= 00000000
@0000337c: *00000008 <= 00000000
@00003394: $ 1 <= ffff0000
@00003398: $ 1 <= ffffffce
@0000339c: $ 1 <= ffffffce
@000033a0: *00000000 <= ffffffce
@000033a4: $ 2 <= ffffffce
@000033a8: $ 3 <= fce00000
@000033ac: $ 3 <= ffffffce
@000033b0: $ 4 <= 07fffffe
@000033b4: $ 4 <= ffffffce
```

```
@000033b8: $ 5 <= fffffffe
@000033bc: $ 1 <= 00000032
@000033c0: *00000004 <= 00000032
@000033c4: $ 5 <= 00000032
@000033c8: $ 5 <= 00000003
@000033cc: $ 1 <= ffff0000
@000033d0: $ 1 <= ffffffce
@000033d4: $ 1 <= ffffffce
@000033d8: $ 2 <= fffff380
@000033dc: $ 1 <= 00000000
@000033e0: $ 1 <= 0000ffff
@000033e4: $ 1 <= 0001fffe
@000033e8: $ 1 <= 000003ff
@000033ec: $ 1 <= ffff0000
@000033f0: $ 1 <= ffffff80
@000033f4: $ 1 <= 00000048
@000033f8: $ 1 <= 00000900
@000033fc: $ 1 <= 00000901
@00003400: $ 1 <= 80000000
@00003404: $ 1 <= fffff800
@00003408: $ 1 <= 007ffffc
@0000340c: $31 <= 00003414
@00003464: $29 <= 34140000
@00003468: $28 <= 00682800
@0000346c: $27 <= 00003414
@00003470: $ 5 <= 00003478
@00003414: $ 1 <= 0000ffff
@00003418: $ 2 <= ffff0000
@00003420: $ 3 <= ffffffff
@00003424: $ 3 <= fffffff0
@00003428: $ 4 <= 00010000
@0000342c: $ 4 <= 00000200
@00003434: $ 3 <= 0000fffe
@00003438: $ 3 <= 000007ff
@0000343c: $ 4 <= 00010000
@00003440: $ 4 <= 00000400
@0000344c: $ 1 <= 0000ffff
@00003450: $ 1 <= fff00000
@00003454: $ 2 <= ffff0000
@00003458: $ 2 <= fffffffc0
@00003478: $ 1 <= 00000001
@0000347c: $ 2 <= 00000014
@00003484: *00000000 <= 00000001
@00003488: $ 1 <= 00000002
```

```
@00003484: *00000000 <= 00000002
@00003488: $ 1 <= 00000003
@00003484: *00000000 <= 00000003
@00003488: $ 1 <= 00000004
@00003484: *00000000 <= 00000004
@00003488: $ 1 <= 00000005
@00003484: *00000000 <= 00000005
@00003488: $ 1 <= 00000006
@00003484: *00000000 <= 00000006
@00003488: $ 1 <= 00000007
@00003484: *00000000 <= 00000007
@00003488: $ 1 <= 00000008
@00003484: *00000000 <= 00000008
@00003488: $ 1 <= 00000009
@00003484: *00000000 <= 00000009
@00003488: $ 1 <= 0000000a
@00003484: *00000000 <= 0000000a
@00003488: $ 1 <= 0000000b
@00003484: *00000000 <= 0000000b
@00003488: $ 1 <= 0000000c
@00003484: *00000000 <= 0000000c
@00003488: $ 1 <= 0000000d
@00003484: *00000000 <= 0000000d
@00003488: $ 1 <= 0000000e
@00003484: *00000000 <= 0000000e
@00003488: $ 1 <= 0000000f
@00003484: *00000000 <= 0000000f
@00003488: $ 1 <= 00000010
@00003484: *00000000 <= 00000010
@00003488: $ 1 <= 00000011
@00003484: *00000000 <= 00000011
@00003488: $ 1 <= 00000012
@00003484: *00000000 <= 00000012
@00003488: $ 1 <= 00000013
@00003484: *00000000 <= 00000013
@00003488: $ 1 <= 00000014
@00003484: *00000000 <= 00000014
@00003494: $ 1 <= 00000003
@00003498: $ 2 <= 00000003
@000034a0: *00000004 <= 03000032
@000034a4: $ 1 <= 00000004
@000034ac: $ 2 <= 00000009
@000034a0: *0000000c <= 048d099e
@000034b8: $ 2 <= 0000000a
```

```
@000034bc: $ 3 <= 0000000a
@000034c4: *00000000 <= 0000000a
@000034c8: $ 2 <= 00000009
@000034cc: $ 3 <= 00000009
@000034c4: *00000000 <= 00000009
@000034c8: $ 2 <= 00000008
@000034cc: $ 3 <= 00000008
@000034c4: *00000000 <= 00000008
@000034c8: $ 2 <= 00000007
@000034cc: $ 3 <= 00000007
@000034c4: *00000000 <= 00000007
@000034c8: $ 2 <= 00000006
@000034cc: $ 3 <= 00000006
@000034c4: *00000000 <= 00000006
@000034c8: $ 2 <= 00000005
@000034cc: $ 3 <= 00000005
@000034c4: *00000000 <= 00000005
@000034c8: $ 2 <= 00000004
@000034cc: $ 3 <= 00000004
@000034c4: *00000000 <= 00000004
@000034c8: $ 2 <= 00000003
@000034cc: $ 3 <= 00000003
@000034c4: *00000000 <= 00000003
@000034c8: $ 2 <= 00000002
@000034cc: $ 3 <= 00000002
@000034c4: *00000000 <= 00000002
@000034c8: $ 2 <= 00000001
@000034cc: $ 3 <= 00000001
@000034c4: *00000000 <= 00000001
@000034c8: $ 2 <= 00000000
@000034cc: $ 3 <= 00000000
@000034c4: *00000000 <= 00000000
@000034d8: $ 2 <= 0000000b
@000034dc: $ 3 <= fffffff5
@000034e4: *00000008 <= fffffff5
@000034e8: $ 2 <= 0000000a
@000034ec: $ 3 <= fffffff6
@000034e4: *00000008 <= fffffff6
@000034e8: $ 2 <= 00000009
@000034ec: $ 3 <= fffffff7
@000034e4: *00000008 <= fffffff7
@000034e8: $ 2 <= 00000008
@000034ec: $ 3 <= fffffff8
@000034e4: *00000008 <= fffffff8
```

```
@000034e8: $ 2 <= 00000007
@000034ec: $ 3 <= fffffff9
@000034e4: *00000008 <= fffffff9
@000034e8: $ 2 <= 00000006
@000034ec: $ 3 <= fffffffa
@000034e4: *00000008 <= fffffffa
@000034e8: $ 2 <= 00000005
@000034ec: $ 3 <= fffffffb
@000034e4: *00000008 <= fffffffb
@000034e8: $ 2 <= 00000004
@000034ec: $ 3 <= fffffffc
@000034e4: *00000008 <= fffffffc
@000034e8: $ 2 <= 00000003
@000034ec: $ 3 <= fffffffd
@000034e4: *00000008 <= fffffffd
@000034e8: $ 2 <= 00000002
@000034ec: $ 3 <= fffffffe
@000034e4: *00000008 <= fffffffe
@000034e8: $ 2 <= 00000001
@000034ec: $ 3 <= ffffffff
@000034e4: *00000008 <= ffffffff
@000034e8: $ 2 <= 00000000
@000034ec: $ 3 <= 00000000
@000034e4: *00000008 <= 00000000
@000034e8: $ 2 <= ffffffff
@000034ec: $ 3 <= 00000001
@000034e4: *00000008 <= 00000001
@000034f8: $ 2 <= 0000000a
@000034fc: $ 3 <= 0000000a
@00003504: *0000000c <= 0000000a
@00003508: $ 2 <= 00000009
@0000350c: $ 3 <= 00000009
@00003504: *0000000c <= 00000009
@00003508: $ 2 <= 00000008
@0000350c: $ 3 <= 00000008
@00003504: *0000000c <= 00000008
@00003508: $ 2 <= 00000007
@0000350c: $ 3 <= 00000007
@00003504: *0000000c <= 00000007
@00003508: $ 2 <= 00000006
@0000350c: $ 3 <= 00000006
@00003504: *0000000c <= 00000006
@00003508: $ 2 <= 00000005
@0000350c: $ 3 <= 00000005
```

```
@00003504: *0000000c <= 00000005
@00003508: $ 2 <= 00000004
@0000350c: $ 3 <= 00000004
@00003504: *0000000c <= 00000004
@00003508: $ 2 <= 00000003
@0000350c: $ 3 <= 00000003
@00003504: *0000000c <= 00000003
@00003508: $ 2 <= 00000002
@0000350c: $ 3 <= 00000002
@00003504: *0000000c <= 00000002
@00003508: $ 2 <= 00000001
@0000350c: $ 3 <= 00000001
@00003504: *0000000c <= 00000001
@00003508: $ 2 <= 00000000
@0000350c: $ 3 <= 00000000
@00003504: *0000000c <= 00000000
@00003508: $ 2 <= ffffffff
@0000350c: $ 3 <= ffffffff
@00003504: *0000000c <= ffffffff
@00003518: $ 2 <= 0000000b
@0000351c: $ 3 <= fffffff5
@00003524: *00000000 <= fffffff5
@00003528: $ 2 <= 0000000c
@0000352c: $ 3 <= 0000000c
@00003524: *00000000 <= 0000000c
@00003538: $ 2 <= 0000000b
@0000353c: $ 3 <= fffffff5
@00003544: $ 1 <= ffff0000
@00003548: $ 1 <= ffffff9c
@0000354c: $ 2 <= ffffff9c
@00003550: $ 3 <= 007b0000
@00003558: $ 4 <= ffffffff
@0000355c: $ 5 <= cff40000
@00003564: $ 4 <= 007affff
@00003568: $ 5 <= cff40000
@00003570: $ 4 <= ffffff9c
@00003574: $ 5 <= 00000000
@0000357c: $ 4 <= 0063ff9c
@00003580: $ 5 <= 00000214
@00003584: $ 4 <= 0063ff9d
@0000358c: $ 5 <= 0063ff9d
@00003590: $ 1 <= 00001234
@00003594: *00000000 <= 00001234
@00003598: $ 2 <= 00001234
```

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？

答：因为乘除运算的延迟时间很长，若整合进 ALU 会降低 CPU 执行效率。因为 HI、LO 寄存器只和几条指令有关，不是常用寄存器，因此独立出 GRF 部件。

2. 参照你对延迟槽的理解，试解释“乘除槽”。

答：“乘除槽”就是当乘除指令离开乘除部件所在的流水级后，乘除运算仍在进行，这个延迟过程就是“乘除槽”。

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?

答：因为 MEM/WB 之后的组合逻辑的延迟时间相对于 DM 的组合逻辑延迟时间很短，若将数据扩展模块放在 DM 之后会导致 M 级组合逻辑延迟时间更长；而流水线 CPU 的时钟周期频率是由延迟时间最长的那级决定的，所以放在 MEM/WB 后几乎不会影响时钟周期频率，而放在 DM 之后会降低时钟频率，影响 CPU 的执行效率。

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）

答：因为一个字符在内存中占一个字节，因此按字节访问内存可以访问字符串中的任意一个字符，而按字访问内存不能达到这样的效果。

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？

答：设计风格：设计和实现分离；

规范手段：细致的显式设计：数据通路表格、暂停转发表格、主控制器表格、数据冒险表格。

6. 你对流水线 CPU 设计风格有何见解？

答：流水线开发复杂度远高于单周期，其主要原因在于并行性导致的思考点总量及关联度急剧攀升。指令集的任何变化均会产生大量连锁反应。如果单周期、多周期是树，那么流水线就是图。因此，除了流水线方法，确保流水线开发效率与正确性的重要因素之一就是严谨的设计过程。因此设计过程必须是显式（文字、图、表、伪代码等）的，而非隐式的，这样才能确保正确并高效的完成设计调整与工程实现，并且能够回溯整个设计过程以快速定位错误。

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)

a.cal_r

冲突类型	解决办法	测试样例
R-M-RS	M 级转发	`addu $t1,$t2,$t3` `subu $t4,$t1,$t2`
R-M-RT	M 级转发	`addu $t1,$t2,$t3` `slt $t4,$t2,$t1`

R-W-RS	W 级转发	`and $t1,$t2,$t3`
		`instr 无关`
		`mult $t4,$t1,$t2`

R-W-RT	W 级转发	`sllv $t1,$t2,$t3`

		instr 无关 divu $t4,$t2,$t1
I-M-RS	M 级转发	ori $t1,$t2,1000 subu $t4,$t1,$t2
I-M-RT	M 级转发	lui $t1,100 xor $t4,$t2,$t1
I-W-RS	W 级转发	lui $t1,100 instr 无关 subu $t4,$t1,$t2

I-W-RT	W 级转发	addi $t1,$t2,100
		instr 无关
		subu $t4,$t2,$t1

LD-M-RS	暂停	lw $t1,0($t2) addu $t3,$t1,$t2
LD-M-RT	暂停	lb $t1,1($t2) addu $t3,$t2,$t1

LD-W-RS	W 级转发	lw $t1,0($t2)
		instr 无关
		addu $t3,$t1,$t2

LD-W-RT	W 级转发	lw $t1,0($t2)
		instr 无关
		addu $t3,$t2,$t1

JAL(R)-M-RS	M 级转发	jal loop addu $t2,$31,$t1
JAL(R)-M-RT	M 级转发	jalr $t0,$s0 addu $t2,$t1,$t0
JAL(R)-W-RS	W 级转发	jal loop 延迟槽 loop:addu $t2,$31,$t1
JAL(R)-W-RT	W 级转发	jalr $t0,$s0 延迟槽 addu $t2,$t1,$t0
MFHI(LO)-M-RS	M 级转发	mfhi $t0 and $t2,$t0,$t1
MFHI(LO)-M-RT	M 级转发	mflo $t0 and $t2,$t1,$t0
MFHI(LO)-W-RS	W 级转发	mfhi $t0 instr 无关 and $t2,$t0,$t1
MFHI(LO)-W-RT	W 级转发	mflo $t0 instr 无关 and $t2,$t1,$t0

b.cal_i

冲突类型	解决办法	测试样例
R-M-RS	M 级转发	addu $t2,$t1,$t3 ori $t4,$t2,100
R-W-RS	W 级转发	subu $t2,$t1,$t3 instr 无关 ori $t4,$t2,100
I-M-RS	M 级转发	addi $t1,$t0,100 ori $t2,$t1,200
I-W-RS	W 级转发	lui $t1,100 instr 无关 ori $t2,$t1,200
LD-M-RS	暂停	lh $t1,0($0) ori $t2,$t1,100
LD-W-RS	W 级转发	lbu $t1,1($0) instr 无关 ori $t2,$t1,100
JAL(R)-M-RS	M 级转发	jalr $t0,$s2 ori $t1,$t0,100
JAL(R)-W-RS	W 级转发	jal loop 延迟槽 loop:ori $t1,$31,100
MFHI(LO)-M-RS	M 级转发	mfhi $t0 slti $t1,$t0,100
MFHI(LO)-W-RS	W 级转发	mflo $t0 instr 无关 slti $t1,$t0,100

c.branch

冲突类型	解决办法	测试样例
R-E-RS	暂停	addu $t1,$t2,$t3 beq $t1,$t4,loop
R-E-RT	暂停	addu $t1,$t2,$t3 bne $t4,$t1,loop
R-M-RS	M 级转发	addu $t1,$t2,$t3 instr 无关 bgez $t1,loop
R-M-RT	M 级转发	addu $t1,$t2,$t3 instr 无关 beq $t4,$t1,loop
R-W-RS	W 级转发	addu $t1,$t2,$t3 instr 无关 instr 无关 beq $t1,$t4,loop
R-W-RT	W 级转发	addu $t1,$t2,$t3

		instr 无关 instr 无关 beq $t4,$t1,loop
I-E-RS	暂停	ori $t1,$t2,100 blez $t1,loop
I-E-RT	暂停	ori $t1,$t2,100 beq $t4,$t1,loop
I-M-RS	M 级转发	ori $t1,$t2,100 instr 无关 beq $t1,$t4,loop
I-M-RT	M 级转发	ori $t1,$t2,100 instr 无关 beq $t4,$t1,loop
I-W-RS	W 级转发	ori $t1,$t2,100 instr 无关 instr 无关 beq $t1,$t4,loop
I-W-RT	W 级转发	ori $t1,$t2,100 instr 无关 instr 无关 beq $t4,$t1,loop
LD-E-RS	暂停	lw $t1,0($t0) beq $t1,$t4,loop
LD-E-RT	暂停	lw $t1,0($t0) beq $t4,$t1,loop
LD-M-RS	暂停	lw $t1,0($t0) instr 无关 beq $t1,$t4,loop
LD-M-RT	暂停	lw $t1,0($t0) instr 无关 beq $t4,$t1,loop
LD-W-RS	W 级转发	lw $t1,0($t0) instr 无关 instr 无关 beq $t1,$t4,loop
LD-W-RT	W 级转发	lh $t1,0($t0) instr 无关 instr 无关 beq $t4,$t1,loop
JAL-M-RS	M 级转发	jal loop1 延迟槽 loop1:bne $31,$t1,loop2
JAL-M-RT	M 级转发	jal loop1 延迟槽

		loop1:beq $t1,$31,loop2
JAL-W-RS	W 级转发	jal loop1 延迟槽 loop1: instr 无关 bgez $31,loop2
JAL-W-RT	W 级转发	jal loop1 延迟槽 loop1: instr 无关 beq $t1,$31,loop2
MFHI(LO)-M-RS	M 级转发	mfhi $t0 bltz $t0,loop1
MFHI(LO)-M-RT	M 级转发	mflo $t0 bne $t1,$t0,loop1
MFHI(LO)-W-RS	W 级转发	mfhi $t0 instr 无关 bltz $t0,loop1
MFHI(LO)-W-RT	W 级转发	mflo $t0 instr 无关 bne $t1,$t0,loop1

d.load

冲突类型	解决办法	测试样例
R-M-RS	M 级转发	addu $t1,$t2,$t3 lw $s1,0($t1)
R-W-RS	W 级转发	addu $t1,$t2,$t3 instr 无关 lb $s1,2($t1)
I-M-RS	M 级转发	ori $t1,$t2,100 lbu $s1,1($t1)
I-W-RS	W 级转发	ori $t1,$t2,100 instr 无关 lh $s1,0($t1)
LD-M-RS	暂停	lh $t1,2($t2) lw $s1,0($t1)
LD-W-RS	W 级转发	lw $t1,0($t2) instr 无关 lhu $s1,0($t1)
JAL-M-RS	M 级转发	jal loop lw $t1,4($31)
JAL-W-RS	W 级转发	jal loop 延迟槽 loop:lw $t1,4($31)

MFHI(LO)-M-RS	M 级转发	mfhi $t0 lbu $t1,2($t0)
MFHI(LO)-W-RS	W 级转发	mflo $t1 instr 无关 lw $t2,-2($t1)

e.store

冲突类型	解决办法	测试样例
R-M-RS	M 级转发	addu $t1,$t2,$t3 sw $s1,0($t1)
R-W-RS	W 级转发	addu $t1,$t2,$t3 instr 无关 sh $s1,0($t1)
R-W-RT	W 级转发	addu $t1,$t2,$t3 sw $t1,0($t2)
I-M-RS	M 级转发	ori $t1,$t2,100 sb $s1,-3($t1)
I-W-RS	W 级转发	ori $t1,$t2,100 instr 无关 sw $s1,0($t1)
I-W-RT	W 级转发	ori $t1,$t2,100 sw $t1,0($t2)
LD-M-RS	暂停	lw $t1,0($t2) sb $t3,0($t1)
LD-W-RS	W 级转发	lw $t1,0($t2) instr 无关 sh $t3,0($t1)
LD-W-RT	W 级转发	lw $t1,0($t2) sw $t1,0($t5)
JAL-M-RS	M 级转发	jal loop sw $t1,0($31)
JAL-W-RS	W 级转发	jal loop 延迟槽 loop:sb $t1,0($31)
JAL-W-RT	W 级转发	jal loop sw $31,0($t1)
MFHI(LO)-M-RS	M 级转发	mfhi $t0 sb $t1,8($t0)
MFHI(LO)-W-RS	W 级转发	mflo $t0 instr 无关 sb $t1,8($t0)
MFHI(LO)-W-RT	W 级转发	mfhi $t0 sb $t0,8($t3)

f.jr/jalr

冲突类型	解决办法	测试样例
R-E-RS	暂停	addu $t1,$t2,$t3 jr $t1
R-M-RS	M 级转发	addu $t1,$t2,$t3 instr 无关 jalr $t4,$t1
R-W-RS	W 级转发	addu $t1,$t2,$t3 instr 无关 instr 无关 jr $t1
I-E-RS	暂停	ori $t1,$t2,100 jr $t1
I-M-RS	M 级转发	ori $t1,$t2,100 instr 无关 jalr $t4,$t1
I-W-RS	W 级转发	ori $t1,$t2,100 instr 无关 instr 无关 jr $t1
LD-E-RS	暂停	lw $t1,0($t2) jr $t1
LD-M-RS	暂停	lw $t1,0($t2) instr 无关 jr $t1
LD-W-RS	W 级转发	lw $t1,0($t2) instr 无关 instr 无关 jr $t1
JAL-M-RS	M 级转发	jal loop 延迟槽 loop:jr $ra
JAL-W-RS	W 级转发	jal loop 延迟槽 loop: instr 无关 jr $ra

g.mthi/mtlo

冲突类型	解决办法	测试样例
R-M-RS	M 级转发	addu $t2,$t1,$t3 mthi $t2
R-W-RS	W 级转发	subu $t2,$t1,$t3

		instr 无关 mtlo $t2
I-M-RS	M 级转发	addi $t1,$t0,100 mthi $t1
I-W-RS	W 级转发	lui $t1,100 instr 无关 mthi $t1
LD-M-RS	暂停	lh $t1,0($0) mtlo $t1
LD-W-RS	W 级转发	lbu $t1,1($0) instr 无关 mtlo $t1
JAL(R)-M-RS	M 级转发	jalr $t0,$s2 mthi $t0
JAL(R)-W-RS	W 级转发	jal loop 延迟槽 loop:mtlo $31
MFHI(LO)-M-RS	M 级转发	mfhi $t0 mthi $t0
MFHI(LO)-W-RS	W 级转发	mflo $t0 instr 无关 mtlo $t0