P5.2 实验报告

14061075 修闽珂

思考1：

Sw 和lw转发的数据相关，可能包括的情况有1. Lw $1 ?(); sw ? ($1)这种情况下,将内存读出的数据转发到ALU中进行地址的相加，得到要存入的地址。2. lw $1 ?(); nop; sw ? ($1)这种情况下将寄存器写入的数据转发到ALU进行地址的相加。3. Sw ? ($2); lw ? ($2) 这种情况是将一个寄存器中的内容转存到$2寄存器中的值（或者与立即数的和）所对应的存储地址中，紧接着从这个地址中取数放进寄存器。这种情况没有数据冒险。

思考2：

指令大致分为这几种类别，第一种是j类指令，包含j和jal；第二种是R有Rd的指令，addu， subu，用r表示；第三种是R没有Rd的类型，也就是立即数的类型，用i表示，包括ori, lui；第四种是分支指令，用b表示，包含beq指令；第五种是jr，在这里做特殊研究;第六种是m型，涉及存储器的存取，lw和sw。

测试样例所要包含的类型至少有如下几种：

r – r, rs; 882+8+

r- r,rt;

r – I, rs;

r – I rt;

I – r, rt;

l – r, rs;

I – I, rs;

r- b, rs;

r-b rt

i-b rs

i-b rt

lw-b rs

lw-b rt

r – sw, rs

I – sw, rs

R – sw rt

I – sw rt

R – jr

I –jr

Lw – jr

在每一个类别里，每一种指令都要组合出现。当前指令位于ID时，前面一条会引起中途的指令可能位于EX，MEM或者WB阶段，这里又有三种情况。所以共有（4+8+8+2+8+2+8+5）\*3 = 180种可能的情况。

测试程序，（原理附在注释中）,预计结果在图中，左图为寄存器，右图为存储器

1.jump

nop

addi $5 $5 12328

j start

addi $1 $1 1#$1 = 1

addi $2 $2 2#$2 = 2

jal second

addi $3 $3 3#$3 = 3

lui $4 4#$4 = 262144

j third

addu $5 $2 $1#$5 = 3

subu $6 $4 $3#$6 = 262141

addu $7 $2 $4#$7 = 262146

subu $8 $4 $5#$8 = 262141

ori $9 $8 9#$9 = 262141

j fourth

sw $1 ($0)#@0 = 1

sw $2 4($0)#@4 = 2

sw $3 8($0)#@8 = 3

sw $4 12($0)#@12 = 262144

sw $5 16($0)#@16 = 3

j end

sw $7 20($0)#@20 = 262146

sw $8 24($0)#@24 = 262141

#testing jr

start: addi $1 $0 12304#$1 = 12304

jr $1#to line 5

second: nop

jr $31#to line 8

sw $5 8($0)#save $5 to @8

third: lw $2 8($0)

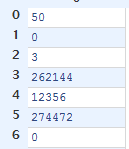
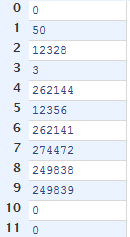
jr $2#load instruction and jr

fourth: addi $1 $0 50#$1 = 50

addu $5 $5 $1#$5 = 12356

jr $5#to line 19

end: nop



2.alu\_sw\_lw

addi $1 $1 12

addi $2 $2 24

addi $3 $3 15

#testings of ALU instruments, they are very same, so gathered as one kind

subu $1 $2 $3

addu $4 $1 $2# r-m-rs mem, rs

subu $1 $2 $3

addu $4 $2 $1#r-m-rt, mem, rt

addu $1 $2 $3

addu $5 $3 $2

subu $6 $1 $2#r-w-rs

addu $1 $3 $4

ori $6 $2 20

subu $7 $5 $6#r-w-rt

addi $8 $6 20

addi $9 $8 0

beq $8 $9 kengdie

addu $1 $2 $1

addi $2 $3 1515462135

kengdie: ori $6 $6 1#ALU-beq, rt, mem

addu $2 $3 $4

addi $10 $2 0

beq $10 $2 heheda

nop

subu $2 $3 $1

heheda: lui $6 12#ALU-beq, rs, mem

sw $4 ($16)

addi $11 $0 12

lw $1 4($11)#alu-lw, mem, rt

lw $2 ($16)

subu $3 $2 $1

lw $4 ($16)

addu $5 $1 $4#lw-alu, mem, rt & rs

addu $9 $0 $12

nop

lw $1 4($11)#alu-lw wb rt

addu $8 $9 $2

addi $4 $0 28

sw $8 ($4)#alu-sw mem rt

addu $8 $9 $2

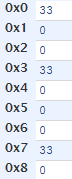
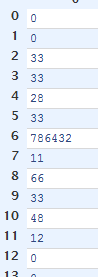
sw $8 ($4)#alu-sw mem, rs

addi $11 $0 12

sw $3 ($11)

lw $9 ($11)

addu $8 $9 $3#lw-alu mem rs



3. random

addi $1 $0 10#$1 = 10

addi $2 $0 20#$2 = 20

sw $2 6($1)#@16 = $12

addi $2 $0 25#$2 = 25

addi $1 $0 15#$1 = 15

sw $2 9($1)#@24 = $2

lw $3 9($1)#$3 = @24

beq $2 $3 TheWitcher3#beq

addu $4 $2 $2 #$4 = $2 + $2

addi $3 $2 3

addi $6 $1 12

TheWitcher3: addu $5 $4 $2#branch

subu $5 $2 $3#$5 = $2 - $3

jal MetalGearSolidV #j and l

addu $5 $3 $4#5 = $3 + 44

ori $2 $5 13

nop

addi $31 $0 0

bgtz $2 FallOut4#bgtz is in ID but the ori appears in the WB

addi $6 $1 8#$6 = $1 ori 8

addu $7 $6 $2

FallOut4: lui $8 4

sw $8 32($0)#the result of $8 store in @32

MetalGearSolidV: ori $4 $5 20

addi $8 $3 30#

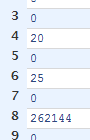
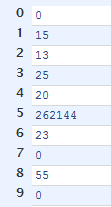
lw $5 32($0)

jr $31

nop

addi $6 $0 12336

jr $6#



4. sw\_lw

addi $1 $1 1#$1 = 1

addi $2 $2 2#$2 = 2

addi $3 $3 3#$3 = 3

lui $4 4#$4 = 262144

addu $5 $2 $1#$5 = 3

subu $6 $4 $3#$6 = 262141

addu $7 $2 $4#$7 = 262146

subu $8 $4 $5#$8 = 262141

ori $9 $8 9#$9 = 262141

sw $1 ($0)#@0 = 1

sw $2 4($0)#@4 = 2

sw $3 8($0)#@8 = 3

sw $4 12($0)#@12 = 262144

sw $5 16($0)#@16 = 3

sw $7 20($0)#@20 = 262146

sw $8 24($0)#@24 = 262141

#testing begin

lw $10 9($3)#$10 = 262144

sw $10 28($0)#@28 = 262144

lw $11 5($3)# $11 = 3

sw $8 21($11)#@24 = 262141

lw $10 9($3)#$10 = 262144

addi $10 $10 -261486#$10 = 658

sw $10 32($0)#@32 = 658

addi $8 $0 15#$8 = 15

lw $11 5($3)#$11 = 3

nop#

sw $8 21($11)#@24 = 15

sw $8 -7($8)#@8 = 15

lw $12 8($0)#$12 = 15

addu $8 $8 $2

sw $8 1($11)#@4 = 17

nop

lw $12 4($0)#$12 = 17

