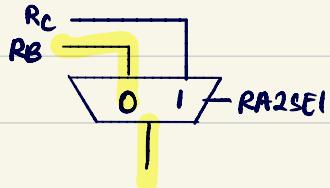


Problem: RA2SEL is always 0

↳ ST is affected



label = 0X00AC

• = 0X000

CMOVE(3, R5)

STC R5, label, R31 → ST RC Ra | C
011001 00101 11111 0000 0000 1010 1100
RB = RO

(software fix for this case)

Fix: CMOVE(3, R0)

STC R[~]X, label, R31)

ignored

if : label = 0x AABBB

not 0

ST	RC	Ra			C
011001	xxxxxx	11111			1010 1010 1011 1011
$\underbrace{\hspace{1cm}}_{RB = R21}$					

if : label = 0xFFAB ?

RB = RSI



no software fix .

Fix: CMOVE(3, R21)

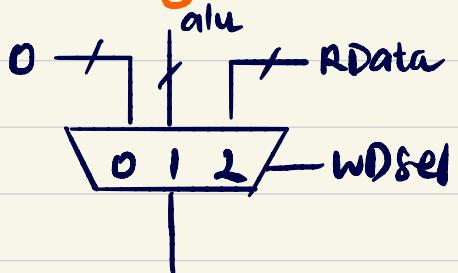
STC RX, label, R31)

ignored

Problem: WDSEL 0 mux is stuck at 0

↳ we cannot store return address automatically

which code can "check" this error?



(a) $\cdot = 0X000$

$BEQ(R31, \text{label}, LP)$

$\text{label: ADDC(RI, 4, RI)}$

$JMP(LP)$

(b) $\cdot = 0X000$

$BEQ(R31, \cdot + 4, LP)$

$JMP(LP)$

$ADDC(RI, 4, RI)$

run for 1 cycle.

Both (a) & (b) \rightarrow after BEQ , content of LP is 0 instead of 4

Problem : WDSSEL 0 mux is stuck at 0

↳ we cannot store return address automatically

$\cdot = 0x000$

BEQ(R31, label, LP)

label : ADDC(R1, 4, R1)

JMP(LP)

any software fix ?



$\cdot = 0x000$

CMOVE($\cdot + 8$, LP)

BEQ(R31, label, R31)

label : ADDC(R1, 4, R1)

JMP(LP)

Problem : BSEL | mux is always 0

all OPC operations are affected, LD, ST

Which code can check for this error?

(a) . = 0X00

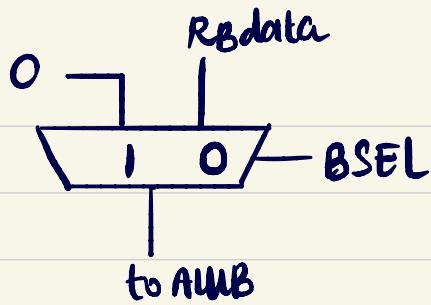
cmove (0xF000, R0) → R0 contains content of R31 instead of
mul (R0, 4, R0) 0X0000F000

(b) . = 0X00

ADD (R31, R3, R3)

JMP (R3)

} no change, can't detect error



. = 0X00
CMOVE (0XF000, R0)
MUL (R0, 4, R0)

Software fix : by loading 0XF000 as a constant
• = 0X00
LDR (constant, R0)
MUL (R0, 4, R0) → then do type I op
constant : 0XF000

(c) . = 0X00

CMOVE (constant, R1)
LD (R1, 4, R2)

constant : LONG (12)
LONG (15)

yes , will detect error.

will not LD '15' onto R2 as BSel 1 is faulty
results in '12' at R2 instead .

. = 0x00

CMOVE (constant, R1)

LD (R1, 4, R2)

constant : LONG (12)

LONG (15)

→ Software fix:

0 := 0x00

LDR (constant, R0)

LDR (index, R1)

ADD (R0, R1, R0)

LD (R0, 0, R2)

constant : LONG (12)

LONG (15)

index : LONG (4)

more tedious than original code

Can we do this in 1 cycle? (without modifying anything but ALU)

ADDMUL (R0, R1, R2) : $\text{Reg}[R2] \leftarrow \text{Reg}[R0] * \text{Reg}[R1]$
 $\text{Reg}[R0] \leftarrow \text{Reg}[R0] + \text{Reg}[R2]$

* No, we only have 1 ALU.

ADDST (R0, label, R2) : $\text{Reg}[R0] \leftarrow \text{Reg}[R0] + \text{Reg}[R2]$
 $\text{Mem}[\text{label}] \leftarrow \text{Reg}[R0]$

* No, output of ALU cannot be directly stored to memory as data

can we do this in 1 cycle? (without modifying anything but CU)

$\text{LDX } (\text{Ra}, \text{Rb}, \text{Rc}) : \quad \text{Reg}[\text{Rc}] \leftarrow \text{Mem}[\text{Reg}[\text{Ra}] + \text{Reg}[\text{Rb}]]$
 $\text{PC} \leftarrow \text{PC} + 4$

WASEL = 0

PCSEL = 0

ASEL = 0

BSEL = 0

RA2SEL = 0

WR = 0

WERF = 1

WDSEL = 2

AUIFN = "+"

← Yes. we have to implement this logic output given the opcode of LDX

any unused ones,
eg: 000001

What is the Reg transfer language for this?

Mem [PC+4+4*SXT(C)] ← Reg[RC]

Reg[RC] ← PC + 4

PC ← Reg[RA]

