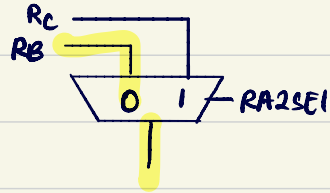
Problem: **RA2SEL is always 0**

↳ ST is affected



label = 0X00AC

. = 0X000

CMOVE(3, R5)

ST(R5, label, R31) → ST Rc Ra C

011001 00101 11111 0000 0000 1010 1100

RB = R0

(Software fix for this case)

Fix: CMOVE (3, R0)

STC Rx, label, R31)

ignored

if : label = 0x AABB

not 0

ST	Rc	Ra		C		
011 001	xxxxx	11111	1010	1010	1011	1011

$R_B = R21$

if : label = 0xFFAB ?

$R_B = R31$

↓

no software fix.

Fix: CMOVE (3, R21)

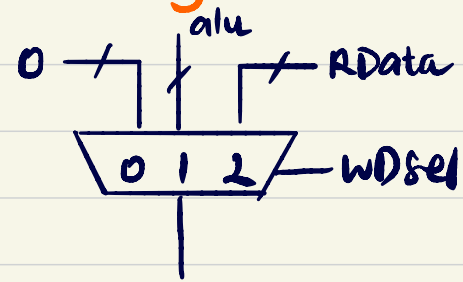
STC Rx, label, R31)

ignored

Problem: WDSEL 0 mux is stuck at 0

↳ we cannot store return address automatically

which code can "check" this error?



(a)
```
. = 0x000
BEQ(R31, label, LP)
label: ADDC(R1, 4, R1)
       JMP(LP)
```

(b)
```
. = 0x000
BEQ(R31, .+4, LP)
JMP(LP)
ADDC(R1, 4, R1)
```

run for 1 cycle.

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

Problem: WDSEL 0 mux is stuck at 0

↳ we cannot store return address automatically

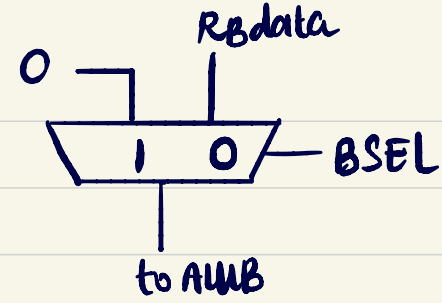
```
. = 0x000
BEQ(R31, label, LP)
label: ADDC(R1, 4, R1)
       JMP(LP)
```

any software fix?

```
. = 0x000
CMOVE(.+8, LP)
BEQ(R31, label, R31)
label: ADDC(R1, 4, R1)
       JMP(LP)
```

Problem: **BSEL 1 mux is always 0**

all OPC operations are affected, LD, ST



Which code can check for this error?

CMOVE is macro for:
ADDC (R31, C, R0)

(a) .=0X00
CMOVE (0XF000, R0) → R0 contains content of R31 instead of 0X0000F000
MULC(R0, 4, R0)

(b) .=0X00
ADD (R31, R3, R3)
JMP (R3)
} no change, cannot detect error

```
.=0X00
CMOVE (0XF000,R0)
MULC(R0, 4, R0)
```

→

Software fix: by loading 0XF000 as a constant

```
.=0X00
LDR( constant1, R0)
LDR( constant2, R1)
MUL (R1, R0, R0) → then do type 1 OP
constant1: 0XF000
constant2: LONG4)
```

(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 INSTRUCTION CMOVE (constant, R1)
– in 32 bits – at R2 instead.

```
. = 0x00
CMOVE ( constant, R1)
LD ( R1, 4, R2)
constant : LONG (12)
           LONG (15)
```

→ Software fix:

```
0 .= 0x00
  LDR (constant-address, R0)
  LDR  (index , R1)
  ADD (R0, R1, R0)
  LD (R0 , 0 , R2)
constant : LONG (12)
           LONG (15)
index : LONG (4)
constant-address: constant
```

more tedious than original code

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

ADDMUL (R0, R1, R2): Reg[R2] ← Reg[R0] * Reg[R1]
Reg[R0] ← Reg[R0] + Reg[R2]

* No, we only have 1 ALU.

ADDST (R0, label, R2): Reg[R0] ← Reg[R0] + Reg[R2]
Mem[label] ← Reg[R0]

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

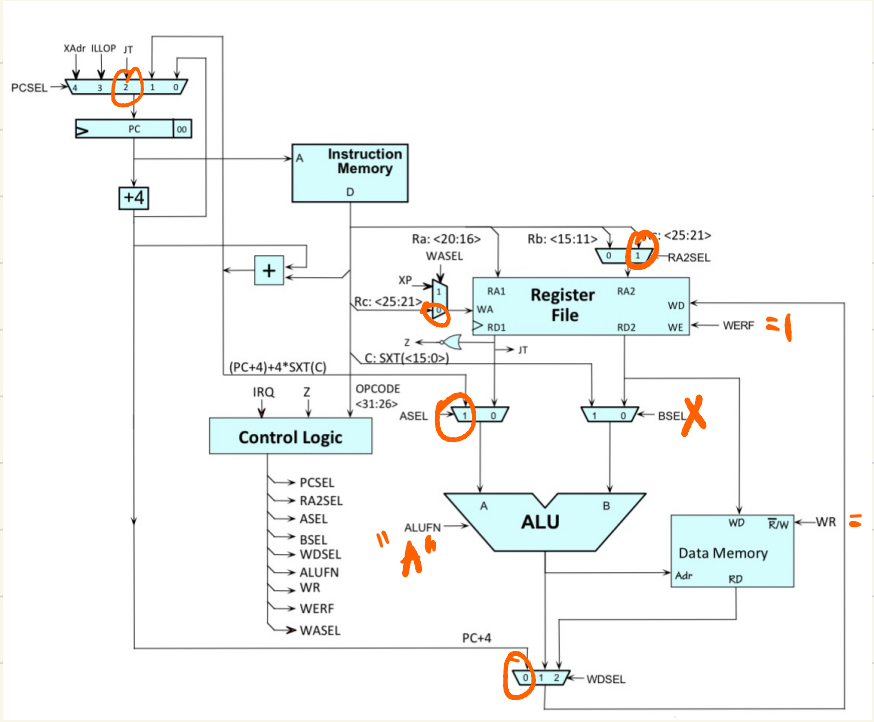
can we do this in 1 cycle? (without modifying anything but CU)

LDX (Ra, Rb, Rc) : Reg[Rc] ← Mem[Reg[Ra] + Reg[Rb]]
PC ← PC+4

WASEL = 0
PCSEL = 0
ASEL = 0
BSEL = 0
RA2SEL = 0
WR = 0
WERF = 1
WDSEL = 2
ALUFN = "+"

← 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]