Verify a Pipelined Picoblaze

Aalap Khanolkar Kathyayani Neerudi
Nikolay Nikolov

ECE 571, Fall 2022
Final Project

Portland State University Maseeh College of Engineering and Computer Science

Date: December 9, 2022

Contents

1 Introduction:

This project aimed to teach us how to design a validation plan, identify critical components in a design, and test each component with a set of SystemVerilog assertions.

The project's end goal was to validate a design with bugs and identify the bugs or at least what components in the design were not performing accurately.

Overall, our team's validation captured some of the bugs or at least units directly impacted by the bugs. However, because of the limited time, we needed to be able to write enough assertions, which could have provided us with enough information to identify where precisely the bugs were.

For example, while we could detect that the ALU was not doing ADD and SUB accurately, we needed more assertions to identify where exactly the bug was.

Moreover, we should have focused on coverage more. On the other hand, we had a well-written assembly file, which exercised the pipeline enough to fire our assertions and capture irregular behaviors in the pipeline during the execution.

Ideally, our coverage should have included three components:

1. Code Coverage (structural) needs to be 100 percent.
2. Functional Coverage that needs to be designed to cover functionality (i.e., intent) of the entire design and must completely cover the design.
3. Temporal domain coverage (using the SVA' cover' feature) must be carefully designed to cover all necessary temporal domain conditions of the design comprehensively. [1]

Initially, we wanted to approach the validation from the perspective of black-box testing.

Formal verification can be called Black Box verification with Black Box observability. The main idea behind this kind of testing is applying vectors/transactions at the primary input of the 'block' without caring for what is in the block (Black Box verification) while observing the block's behavior only at the primary outputs (black-box observability).

Assertions(SystemVerilog), on the other hand, allow doing black box verification with white box (internal to the block) observability. [1]

2 Design Description:

Our design consisted of a .psm file (assembly) where we wrote instructions that should trigger particular functionalities based on the PicoBlaze spec. We created the assembly code primarily by reviewing the PicoBlaze specifications and identifying key behaviors.[3][4]

We have provided the assembly code as a reference in the next couple of pages. We wrote our code in a self-contained manner. While we have included comments in the assembly code, we also wanted to ensure that if we change one variable, we do not need to go back and change all the comments. Moreover, while professors often love well-commented code, the issue we run into is that when more people edit code simultaneously, it is easy to end up with irrelevant comments that everyone is afraid to remove.

```
;-----------------------------------------------------
; alu_test.psm - test alu
;
; Author   : Niko Nikolov
; Modified : 12-2-2022
;
; Description :
; Alu operations
;
;-----------------------------------------------------
; Rename Registers
; NAMEREG sX, <name>
;-----------------------------------------------------
NAMEREG s1, ram_data
NAMEREG s2, ram_address
NAMEREG s3, add_reg
NAMEREG s4, sub_reg
NAMEREG s5, xor_reg
NAMEREG s6, and_reg
NAMEREG s7, or_reg
NAMEREG s8, sl0_reg
NAMEREG s9, sl1_reg
NAMEREG s9, sr1_reg
NAMEREG sA, carry
NAMEREG sB, test_reg
NAMEREG sC, compare_reg
NAMEREG sD, rotate_reg

; Constnants
; -----------------------------------------------------
CONSTANT initial_value  , 00                        ; Initialize to 0
CONSTANT ram_locations  , 40                        ; There are 64 locationss
; -----------------------------------------------------
ADDRESS 000 ; Program always start at reset vector 0
; -----------------------------------------------------
; -----------------------------------------------------
; Arithmentic Instructions
; if the resulting sum is either
; 0 or 256 (register sX is zero with CARRY set), then the ZERO flag is set.
add_zero_set_zero_:
    LOAD add_reg            , 00                    ; Initialize to 0
    ADD  add_reg            , 00                    ; Add 0 to result
    LOAD s0  , s0 ; NOP
    JUMP add_256_set_zero_;
;-----------------------------------------------------
add_256_set_zero_:
    LOAD add_reg            , 00                    ; Initialize to 0
    ADD  add_reg            , 100                   ; Add 256 (0x100) to
```

```
    result
    LOAD s0  , s0 ; NOP
    JUMP add_257_set_carry_addcy_;
;-----------------------------------------------------
add_257_set_carry_addcy_:
    LOAD    add_reg         , FF                        ; Add FF 255 to result
    ADD add_reg             , 1                         ;
    LOAD s0  , s0 ; NOP
    JUMP add_addcy_
;-----------------------------------------------------
add_addcy_:
    LOAD    add_reg         , FF                        ; Add 255 to result
    ADDCY   add_reg         , 01                        ; Next clock cycle Carry
    should be 1
    LOAD s0  , s0 ; NOP
    JUMP sub_zero_from_zero_set_zero_;
;-----------------------------------------------------
; The resulting operation affects both the CARRY and ZERO
; flags. If the resulting difference is less than 0, then the CARRY flag is set. If
     the resulting
; difference is 0 or -256, then the ZERO flag is set.
; The SUBCY instruction is a subtract operation with borrow. If the CARRY flag is
    set, then
; SUBCY subtracts an additional one from the resulting difference.
;-----------------------------------------------------
sub_zero_from_zero_set_zero_:
    LOAD sub_reg            , 00                    ; Initialize to 0
    SUB  sub_reg            , 00                    ;
    JUMP sub_one_from_zero_set_carry_;
;-----------------------------------------------------
sub_one_from_zero_set_carry_:
    LOAD sub_reg            , 00                     ; Initialize to 0
    SUB  sub_reg            , 01                     ; Next clock cycle Carry
    should 1
    JUMP sub_one_from_previous_subcy_                ;
;-----------------------------------------------------
; The SUBCY instruction is a subtract operation with borrow. If the CARRY flag is
    set, then
; SUBCY subtracts an additional one from the resulting difference.
;-----------------------------------------------------
sub_one_from_previous_subcy_:
    LOAD sub_reg            , 00                     ; Initialize to 0
    SUBCY  sub_reg          , 01                      ; Carry should have been
    asserted above hence sub 2 (carry)
    JUMP clear_carry_bit_;
;-----------------------------------------------------
clear_carry_bit_:
    AND sub_reg         , sub_reg                    ; Clear carry bit
   JUMP set_carry_                                   ;
;-----------------------------------------------------
set_carry_:
    LOAD carry          , 00                         ; Set Carry
    COMPARE carry       , 01                         ; Reset Zero flag
    JUMP gen_parity_test_                            ;

;-----------------------------------------------------
gen_parity_test_:
    LOAD test_reg           , 05                     ; 00000101
    TEST test_reg           , 04                     ; Carry = 1, ZERO = 0
    JUMP gen_all_bits_parity_                        ;
;-----------------------------------------------------
gen_all_bits_parity_:
    TEST s0 , FF ; Include all bit in parity gen
    JUMP compare_zero_set_;
;-----------------------------------------------------
; The COMPARE instruction performs an 8-bit subtraction of two operands but only
    affects
; the ZERO and CARRY flags
;-----------------------------------------------------
compare_zero_set_:
    LOAD compare_reg         , 00                    ; Set to 0
    COMPARE compare_reg      , 00                    ; Set the Zero flag
    LOAD s0  , s0 ; NOP
```

```
    COMPARE compare_reg      , 01                        ; Set Carry Flag
    JUMP sl0_                                            ;

; -------------------------------------------------------
; All shift instructions affect the
; CARRY and ZERO flags.
; -------------------------------------------------------
; The SL0 sX instruction shift the contents of register sX
; left by one bit position.
sl0_:
    LOAD sl0_reg             , FF                        ;
    SL0 sl0_reg;
    JUMP sl1_ ;
;-------------------------------------------------------
; The SL1 and SR1 shift instructions are similar to SL0 and
; SR0 except that the empty bit
; location is filled with a 1
;-------------------------------------------------------
sl1_:
    LOAD sl1_reg             , FF                        ;
    SL1 sl1_reg;
    JUMP sr1_ ;
;-------------------------------------------------------
sr1_:
    LOAD    sr1_reg          , 01                        ; Load 1  and shift rign
    SR1     sr1_reg;
    JUMP rotate_left_ ;
;-------------------------------------------------------
; The rotate instructions, shown in Table 3-5, rotate the
; contents of the specified register left
; or right. The RL sX instruction shifts the contents of register sX
; left with the most-significant bit, bit 7,
; feeding the least-significant bit, bit 0
;-------------------------------------------------------
rotate_left_:
    LOAD rotate_reg         , FF                         ;
    RL rotate_reg                                        ; Rotate left
    JUMP rotate_right_                                   ; rotate
;-------------------------------------------------------
rotate_right_:
    LOAD rotate_reg         , FF                         ;
    RR rotate_reg                                        ; Rotate Right
    JUMP init_ ;
;-------------------------------------------------------
init_:
    LOAD ram_address        , ram_locations              ; Initialize the top
    ram addr
    LOAD ram_data           , initial_value              ; Initialize ram data
    JUMP loop_                                           ;
;-------------------------------------------------------
loop_:
    LOAD s0                 , s0                         ; NOP
    SUB ram_address         , 01                         ;
    STORE ram_address       , (ram_address)              ;
    JUMP NZ                 , loop_                      ;

end_:
    LOAD s0 , s0                                         ; NOP
    JUMP
    RETURN
```

However, the assembly file alone would have been useless. Therefore, we had to use the KCPSM3Asm Assembler to apply the stimulus to our design. Compiling the assembly code with the assembler provided us with a .mem file. The produced mem file is included below as a reference.

As a side note, we decided to include some of our files in our report to make it easier to review instead of having to toggle between a text

editor and a PDF reader.

```
/* Symbol Table */
// add_256_set_zero_ = LABEL: 4
// add_257_set_carry_addcy_ = LABEL: 8
// add_addcy_ = LABEL: 12
// add_reg = REGISTER: 3
// add_zero_set_zero_ = LABEL: 0
// and_reg = REGISTER: 6
// carry = REGISTER: 10
// clear_carry_bit_ = LABEL: 25
// compare_reg = REGISTER: 12
// compare_zero_set_ = LABEL: 35
// end_ = LABEL: 62
// gen_all_bits_parity_ = LABEL: 33
// gen_parity_test_ = LABEL: 30
// init_ = LABEL: 55
// initial_value = CONSTANT: 0
// loop_ = LABEL: 58
// or_reg = REGISTER: 7
// ram_address = REGISTER: 2
// ram_data = REGISTER: 1
// ram_locations = CONSTANT: 64
// rotate_left_ = LABEL: 49
// rotate_reg = REGISTER: 13
// rotate_right_ = LABEL: 52
// s0 = REGISTER: 0
// s1 = REGISTER: 1
// s2 = REGISTER: 2
// s3 = REGISTER: 3
// s4 = REGISTER: 4
// s5 = REGISTER: 5
// s6 = REGISTER: 6
// s7 = REGISTER: 7
// s8 = REGISTER: 8
// s9 = REGISTER: 9
// sA = REGISTER: 10
// sB = REGISTER: 11
// sC = REGISTER: 12
// sD = REGISTER: 13
// sE = REGISTER: 14
// sF = REGISTER: 15
// set_carry_ = LABEL: 27
// sl0_ = LABEL: 40
// sl0_reg = REGISTER: 8
// sl1_ = LABEL: 43
// sl1_reg = REGISTER: 9
// sr1_ = LABEL: 46
// sr1_reg = REGISTER: 9
// sub_one_from_previous_subcy_ = LABEL: 22
// sub_one_from_zero_set_carry_ = LABEL: 19
// sub_reg = REGISTER: 4
// sub_zero_from_zero_set_zero_ = LABEL: 16
// test_reg = REGISTER: 11
// xor_reg = REGISTER: 5

/* Program Code */
// #1: ;-----------------------------------------------------------
// #2: ; alu_test.psm - test alu
// #3: ;
// #4: ; Author   : Niko Nikolov
// #5: ; Modified : 12-2-2022
// #6: ;
// #7: ; Description :
// #8: ; Alu operations
// #9: ;
// #10: ;-----------------------------------------------------------
// #11: ; Rename Registers
// #12: ; NAMEREG sX, <name>
// #13: ;-----------------------------------------------------------
// #14: NAMEREG(s1,ram_data)
// #15: NAMEREG(s2,ram_address)
// #16: NAMEREG(s3,add_reg)
// #17: NAMEREG(s4,sub_reg)
```

```
// #18: NAMEREG(s5,xor_reg)
// #19: NAMEREG(s6,and_reg)
// #20: NAMEREG(s7,or_reg)
// #21: NAMEREG(s8,sl0_reg)
// #22: NAMEREG(s9,sl1_reg)
// #23: NAMEREG(s9,sr1_reg)
// #24: NAMEREG(sA,carry)
// #25: NAMEREG(sB,test_reg)
// #26: NAMEREG(sC,compare_reg)
// #27: NAMEREG(sD,rotate_reg)
// #29: ; Constnants
// #30: ; -----------------------------------------------------
// #31: CONSTANT(initial_value,0) ; Initialize to 0
// #32: CONSTANT(ram_locations,64) ; There are 64 locationss
// #33: ; -----------------------------------------------------
@000 // #34: ADDRESS(0) ; Program always start at reset vector 0
// #35: ; -----------------------------------------------------
// #36: ; -----------------------------------------------------
// #37: ; Arithmentic Instructions
// #38: ; if the resulting sum is either
// #39: ; 0 or 256 (register sX is zero with CARRY set), then the ZERO flag is set.
// @000 #40: [add_zero_set_zero_]
00300 // @000 #41: LOAD(add_reg,0) ; Initialize to 0
18300 // @001 #42: ADD(add_reg,0) ; Add 0 to result
01000 // @002 #43: LOAD(s0,s0) ; NOP
34004 // @003 #44: JUMP(add_256_set_zero_) ;
// #45: ;-----------------------------------------------------
// @004 #46: [add_256_set_zero_]
00300 // @004 #47: LOAD(add_reg,0) ; Initialize to 0
18300 // @005 #48: ADD(add_reg,256) ; Add 256 (0x100) to result
01000 // @006 #49: LOAD(s0,s0) ; NOP
34008 // @007 #50: JUMP(add_257_set_carry_addcy_) ;
// #51: ;-----------------------------------------------------
// @008 #52: [add_257_set_carry_addcy_]
003ff // @008 #53: LOAD(add_reg,FF) ; Add FF 255 to result
18301 // @009 #54: ADD(add_reg,1) ;
01000 // @00a #55: LOAD(s0,s0) ; NOP
3400c // @00b #56: JUMP(add_addcy_)
// #57: ;-----------------------------------------------------
// @00c #58: [add_addcy_]
003ff // @00c #59: LOAD(add_reg,FF) ; Add 255 to result
1a301 // @00d #60: ADDCY(add_reg,1) ; Next clock cycle Carry should be 1
01000 // @00e #61: LOAD(s0,s0) ; NOP
34010 // @00f #62: JUMP(sub_zero_from_zero_set_zero_) ;
// #63: ;-----------------------------------------------------
// #64: ; The resulting operation affects both the CARRY and ZERO
// #65: ; flags. If the resulting difference is less than 0, then the CARRY flag is
     set. If the resulting
// #66: ; difference is 0 or -256, then the ZERO flag is set.
// #67: ; The SUBCY instruction is a subtract operation with borrow. If the CARRY
    flag is set, then
// #68: ; SUBCY subtracts an additional one from the resulting difference.
// #69: ;-----------------------------------------------------
// @010 #70: [sub_zero_from_zero_set_zero_]
00400 // @010 #71: LOAD(sub_reg,0) ; Initialize to 0
1c400 // @011 #72: SUB(sub_reg,0) ;
34013 // @012 #73: JUMP(sub_one_from_zero_set_carry_) ;
// #74: ;-----------------------------------------------------
// @013 #75: [sub_one_from_zero_set_carry_]
00400 // @013 #76: LOAD(sub_reg,0) ; Initialize to 0
1c401 // @014 #77: SUB(sub_reg,1) ; Next clock cycle Carry should 1
34016 // @015 #78: JUMP(sub_one_from_previous_subcy_) ;
// #79: ;-----------------------------------------------------
// #80: ; The SUBCY instruction is a subtract operation with borrow. If the CARRY
    flag is set, then
// #81: ; SUBCY subtracts an additional one from the resulting difference.
// #82: ;-----------------------------------------------------
// @016 #83: [sub_one_from_previous_subcy_]
00400 // @016 #84: LOAD(sub_reg,0) ; Initialize to 0
1e401 // @017 #85: SUBCY(sub_reg,1) ; Carry should have been asserted above hence
    sub 2 (carry)
34019 // @018 #86: JUMP(clear_carry_bit_) ;
// #87: ;-----------------------------------------------------
```

```
// @019 #88: [clear_carry_bit_]
0b440 // @019 #89: AND(sub_reg,sub_reg) ; Clear carry bit
3401b // @01a #90: JUMP(set_carry_) ;
// #91: ;-----------------------------------------------------
// @01b #92: [set_carry_]
00a00 // @01b #93: LOAD(carry,0) ; Set Carry
14a01 // @01c #94: COMPARE(carry,1) ; Reset Zero flag
3401e // @01d #95: JUMP(gen_parity_test_) ;
// #97: ;-----------------------------------------------------
// @01e #98: [gen_parity_test_]
00b05 // @01e #99: LOAD(test_reg,5) ; 00000101
12b04 // @01f #100: TEST(test_reg,4) ; Carry = 1, ZERO = 0
34021 // @020 #101: JUMP(gen_all_bits_parity_) ;
// #102: ;-----------------------------------------------------
// @021 #103: [gen_all_bits_parity_]
120ff // @021 #104: TEST(s0,FF) ; Include all bit in parity gen
34023 // @022 #105: JUMP(compare_zero_set_) ;
// #106: ;-----------------------------------------------------
// #107: ; The COMPARE instruction performs an 8-bit subtraction of two operands
    but only affects
// #108: ; the ZERO and CARRY flags
// #109: ;-----------------------------------------------------
// @023 #110: [compare_zero_set_]
00c00 // @023 #111: LOAD(compare_reg,0) ; Set to 0
14c00 // @024 #112: COMPARE(compare_reg,0) ; Set the Zero flag
01000 // @025 #113: LOAD(s0,s0) ; NOP
14c01 // @026 #114: COMPARE(compare_reg,1) ; Set Carry Flag
34028 // @027 #115: JUMP(sl0_) ;
// #117: ; -----------------------------------------------------
// #118: ; All shift instructions affect the
// #119: ; CARRY and ZERO flags.
// #120: ; -----------------------------------------------------
// #121: ; The SL0 sX instruction shift the contents of register sX
// #122: ; left by one bit position.
// @028 #123: [sl0_]
008ff // @028 #124: LOAD(sl0_reg,FF) ;
20806 // @029 #125: SL0(sl0_reg) ;
3402b // @02a #126: JUMP(sl1_) ;
// #127: ;-----------------------------------------------------
// #128: ; The SL1 and SR1 shift instructions are similar to SL0 and
// #129: ; SR0 except that the empty bit
// #130: ; location is filled with a 1
// #131: ;-----------------------------------------------------
// @02b #132: [sl1_]
009ff // @02b #133: LOAD(sl1_reg,FF) ;
20907 // @02c #134: SL1(sl1_reg) ;
3402e // @02d #135: JUMP(sr1_) ;
// #136: ;-----------------------------------------------------
// @02e #137: [sr1_]
00901 // @02e #138: LOAD(sr1_reg,1) ; Load 1  and shift rign
2090f // @02f #139: SR1(sr1_reg) ;
34031 // @030 #140: JUMP(rotate_left_) ;
// #141: ;-----------------------------------------------------
// #142: ; The rotate instructions, shown in Table 3-5, rotate the
// #143: ; contents of the specified register left
// #144: ; or right. The RL sX instruction shifts the contents of register sX
// #145: ; left with the most-significant bit, bit 7,
// #146: ; feeding the least-significant bit, bit 0
// #147: ;-----------------------------------------------------
// @031 #148: [rotate_left_]
00dff // @031 #149: LOAD(rotate_reg,FF) ;
20d02 // @032 #150: RL(rotate_reg) ; Rotate left
34034 // @033 #151: JUMP(rotate_right_) ; rotate
// #152: ;-----------------------------------------------------
// @034 #153: [rotate_right_]
00dff // @034 #154: LOAD(rotate_reg,FF) ;
20d0c // @035 #155: RR(rotate_reg) ; Rotate Right
34037 // @036 #156: JUMP(init_) ;
// #157: ;-----------------------------------------------------
// @037 #158: [init_]
00240 // @037 #159: LOAD(ram_address,ram_locations) ; Initialize the top ram addr
00100 // @038 #160: LOAD(ram_data,initial_value) ; Initialize ram data
3403a // @039 #161: JUMP(loop_) ;
```

```
214 // #162: ;-----------------------------------------------------
215 // @03a #163: [loop_]
216 01000 // @03a #164: LOAD(s0,s0) ; NOP
217 1c201 // @03b #165: SUB(ram_address,1) ;
218 2f220 // @03c #166: STORE(ram_address,ram_address) ;
219 3543a // @03d #167: JUMP(NZ,loop_) ;
220 // @03e #169: [end_]
221 01000 // @03e #170: LOAD(s0,s0) ; NOP
222 2a000 // @03f #171: RETURN
```

Once we had the assembly file and the memory formatted file, the next step was to test the stimulus with the provided RojoBlaze pipeline.

Furthermore, when we had everything in place, we examined and compared the logs between the pipeline we knew had no bugs with the provided encrypted pipeline. Uncovering differences between the two execution logs was the first significant step to capturing different behaviors. The most noticeable difference was that storing any value to the 64-byte scratchpad failed in the bugged version, which generally would have succeeded.

However, we still needed to determine the reason for the failure. To proceed further, we started writing assertions for a few basic functionalities we knew should work to affect the pipeline to meet some basic behaviors.

Hence, we started to look for clues in each respective domain each team member had. For example, Aalap tested if the instruction decode worked correctly, while Kathyayani aimed at validating the Program Counter, and Niko tested ALU's addition and subtraction.

We began designing our tests targeting higher-level behaviors.

The motivation behind our intent was isolating the broken units in the pipeline. Thus, once we knew what parts of the pipeline were not functioning correctly, we planned to drill down to each component and validate if each input/output of the unit complied with the expected timing and functional constraints outlined in the PicoBlaze.

Overall, we spent two-thirds of our time digging into the PicoBlaze specs and reading how to use all the tools available.

Since PicoBlaze is an older project, we needed help finding any additional tools we could have used to make our workflow more efficient. For example, we tried installing a PicoBlaze IDE. However, the IDE was slow and, for the most part, not valid. It did not provide any additional capabilities we could have by using a text editor and the java utility to assemble our psm files.

The most helpful were the files provided by Roy, especially those created by Miles and Seth prior. Those files gave us enough clues about what we had to do to complete the project.

Below we are providing our assertions in the "Assertions" section.

3 Coverage:

Despite our best efforts, we needed more time to focus on coverage. Based on QuestaSim's coverage report, we achieved 33% coverage with 16 assertions. Although, despite the low range, we captured most of the defects or infected modules because we had a good strategy in place. Sometimes, it is not about quantity.

4 Assertions:

```
/* Assertions */

//boundary check - when write enable is on then check the address is within 6 bit boundary
always @(posedge clk) begin
  assert property(dut.scratch_write_enable |-> (dut.scratch_address >= 6'd0 && dut.scratch_address <=6'd63))
  else
    $display("\n\nError scratchpad address boundary issue address = %d\n\n", dut.scratch_address);

end

always @(posedge clk) begin
  assert property(dut.instruction[17:13] !== 5'h01 &&
                    dut.instruction[17:13] !== 5'h04 &&
                    dut.instruction[17:13] !== 5'h08 &&
                    dut.instruction[17:13] !== 5'h0b &&
                    dut.instruction[17:13] !== 5'h11 &&
                    dut.instruction[17:13] !== 5'h12 &&
                    dut.instruction[17:13] !== 5'h13 &&
                    dut.instruction[17:13] !== 5'h14 &&
                    dut.instruction[17:13] !== 5'h19 &&
                    dut.instruction[17:13] !== 5'h1b &&
                    dut.instruction[17:13] !== 5'h1d &&
                    dut.instruction[17:13] !== 5'h1f)
  else
    $display(
        "\n\nError instruction[17:13] = %x, should be inside opcode_t\n\n", dut.instruction[17:13]
    );
end


// PC should jump to the target address
property pc_jump;
  @(posedge clk) disable iff (rst) (((dut.instruction[17:13] === 5'h1a) || (dut.instruction[17:13] === 5'h18)) && (dut.enable_PC === 1'b1) && (dut.conditional_match)) |=> ((dut.program_counter) === $past(
    dut.idu_code_address
  ))
endproperty
assert property (pc_jump)
else
  $display(
      "\n\n Error OP: %s , PC is [%d], address is [%d] \n\n",
      $past(
          opcode
      ),
      (dut.program_counter),
      $past(
```

```
            dut.idu_code_address, 1
        )
    );

// to check if the opcode matches with idu_operation
property opcode_check;
    @(posedge clk) disable iff (rst) (dut.instruction[17:13] === dut.idu_operation)
endproperty
assert property (opcode_check)
else
    $display(
        "\n\n Error OP: %s , inst is [%d] operation is [%d] \n\n",
        $past(
            opcode
        ),
        $past(
            dut.instruction[17:13]
        ),
        $past(
            dut.idu_operation
        )
    );

// to check if PC is zero when reset is asserted
property reset_check;
    @(posedge clk) ((rst) |=> ##2 (dut.program_counter) === RESET_VECTOR)
endproperty
assert property (reset_check)
else
    $display(
        "\n\n Error: PC is [%d] RESET_VECTOR is [%d] \n\n", $past(dut.program_counter
    ), (RESET_VECTOR)
    );

//check if PC rollover
property rollover_check;
    @(posedge clk) disable iff (rst) ((dut.instruction[17:13] === 5'h1a) && (dut.
    program_counter === 10'hFFFFFFFF)
    |-> ##2 (dut.program_counter === 10'h0))
endproperty
assert property (rollover_check)
else
    $display(
        "\n\n Error: PC is [%d] Instruction is [%s] \n\n", dut.program_counter, dut.
    instruction[17:13]
    );

/* If ADD result is 0 then zero flag is asserted */
//-------------------------------------------------------------------------
property zero_flag;
    @(clk) disable iff (rst) ((dut.instruction[17:13] === 5'h0c) && (dut.alu.result
    === 2'h0)) |-> (##[1:5] dut.zero)
endproperty
assert property (zero_flag)
else
    $display(
        "\n\n ALU Result was 0; Error OP: %s , Zero flag was not set [ %h ], FAIL \n\
    n",
        opcode,
        dut.zero
    );
//-------------------------------------------------------------------------
/* If ADD result is 0 then carry flag is asserted */
property carry_flag;
    @(posedge clk) disable iff (rst) ((dut.instruction[17:13] === 5'h0c) && (dut.alu.
    result > 2'hff)) |-> (##1 dut.alu.carry_out);
endproperty
assert property (carry_flag)
else
    $display(
        "\n\n ALU was Result > FF ; Error OP: %s , ALU Carry Out was not set [ %h ],
    FAIL \n\n",
        opcode,
```

```
112       dut.alu.carry_out
113   );
114 //------------------------------------------------------------------------
115 /* If ADDCY result is 0 then carry flag is asserted */
116 property addcy_carry_flag;
117   @(posedge clk) disable iff (rst) ((dut.instruction[17:13] === 5'h0d) && (dut.alu.
      result > 2'hff)) |-> (##1 dut.alu.carry_out);
118 endproperty
119 assert property (addcy_carry_flag)
120 else
121   $display(
122       "\n\n ALU Result > FF; Error OP: %s , ALU Carry Out was not set [ %h ], FAIL
      \n\n",
123       opcode,
124       dut.alu.carry_out
125   );
126 //------------------------------------------------------------------------
127 /* If SUB 0 minus 0 will set the Zero flag */
128 property sub_zero_flag;
129   @(posedge clk) disable iff (rst) ((dut.instruction[17:13] === 5'h0e) && (dut.alu.
      result < 2'h0)) |-> (##1 dut.zero);
130 endproperty
131 assert property (sub_zero_flag)
132 else
133   $display(
134       "\n\n ALU Result < 0; Error OP: %s , Zero Flag was not set [ %h ], FAIL \n\n"
      ,
135       opcode,
136       dut.zero
137   );
138 //------------------------------------------------------------------------
139 /* If SUB result is negative set carry */
140 property sub_carry_flag;
141   @(posedge clk) disable iff (rst) ((dut.instruction[17:13] === 5'h0e) && (dut.alu.
      result < 2'h0)) |-> (##1 dut.alu.carry_out);
142 endproperty
143 assert property (sub_carry_flag)
144 else
145   $display(
146       "\n\n ALU Result < 0; Error OP: %s , ALU Carry Out was not set [ %h ], FAIL \
      n\n",
147       opcode,
148       dut.alu.carry_out
149   );
150 //------------------------------------------------------------------------
151 /* If SUBCY result is negative set carry */
152 property subcy_carry_flag;
153   @(posedge clk) disable iff (rst) ((dut.instruction[17:13] === 5'h0f) && (dut.alu.
      result < 2'h0)) |-> (##1 dut.alu.carry_out);
154 endproperty
155 assert property (subcy_carry_flag)
156 else
157   $display(
158       "\n\n Subcy ALU Result < 0; Error OP: %s , Zero Flag was not set [ %h ], FAIL
      \n\n",
159       opcode,
160       dut.alu.carry_out
161   );
162 //------------------------------------------------------------------------
163 /* If we JUMP and ZERO flag is set, if the next instruction is AND sX, sX */
164 /* Carry bit should be cleared  with AND sX, sX */
165 //------------------------------------------------------------------------
166 property clear_carry_flag;
167   @(posedge clk) disable iff (rst) ((dut.instruction[17:13] === 5'h05) && (dut.
      instruction[11:8] === dut.instruction[7:4]) && (dut.alu.carry_out ) ) |=> (##1
      (dut.alu.carry_out === 0));
168 endproperty
169 assert property (clear_carry_flag)
170 else
171   $display(
172       "\n\n AND sX, sX should clear Carry Flag; Error OP: %s , Carry Flag should
      has de-asserted but it is [ %h ], FAIL \n\n",
173       opcode,
```

```
      dut.alu.carry_out
  );

//------------------------------------------------------------------------------
/* If we COMPARE 0,0 we should NOT set the Carry Flag */
//------------------------------------------------------------------------------
property set_carry_flag;
  @(posedge clk) disable iff (rst) ((dut.instruction[17:13] === 5'h0a) && (dut.
    instruction[11:8] === dut.instruction[7:4]) ) |=> (##1 (dut.alu.carry_out ===
    0) && dut.zero);
endproperty
assert property (set_carry_flag)
else
  $display(
      "\n\n COMPARE sX, sX should set Zero flag but not Carry Flag; Error OP: %s ,
    Carry Flag should has de-asserted but it is [ %h ], FAIL \n\n",
      opcode,
      dut.alu.carry_out
  );

//------------------------------------------------------------------------------
/* When shiftbit then op1 + op2 == result */
//------------------------------------------------------------------------------

property test_ADD_SUB;
  @(negedge clk) disable iff (rst)  ( ((dut.alu.operand_a !== 2'hx) && (dut.alu.
    operand_a)) && ( (dut.alu.operand_b !== 2'hx) && dut.alu.operand_b)

    |-> (((dut.alu.operand_a + dut.alu.operand_b ) === dut.alu.addsub_result[7:0])
    || (dut.alu.operand_a - dut.alu.operand_b ) === dut.alu.addsub_result[7:0]) );
endproperty
assert property (test_ADD_SUB)
else
  $display(
      "\n\n Error: OP1 %h +/- OP2 %h should NOT equal %h, you should ignore the
    error if OP1 == FF && OP2 == 1 \n\n",
      dut.alu.operand_a,
      dut.alu.operand_b,
      dut.alu.addsub_result[7:0]
  );


/* Since we decrement by 1 to write to scrachpad OP1 and Result from previous OP
    should be at most 1 less 8 */
property test_STORE;
  @(negedge clk) disable iff (rst) ((dut.instruction[17:13] === 5'h17) && (dut.
    instruction[11:8] === 2'h2 && dut.instruction[7:4] )
   |-> dut.alu.operand_a === (dut.alu.result + 2'h01)) ;
endproperty
assert property (test_STORE)
else
  $display(
      "\n\n Error: STORE after SUB S2,01 operand_a [OP1] is %h the next result
    should be  %h , but is %h, FAIL \n\n",
      dut.alu.operand_a,
      (dut.alu.operand_a - 2'h01),
      dut.alu.result

  );
```

5 Unit Level Testing Plans:

5.1 Instruction Fetch

Team Member	Design Unit	Test Name
Kathyayani	IF	reset_check: Checks if PC = zero when reset signal is asserted
Kathyayani	IF	rollover_check: Checks if PC rollover is working
Kathyayani	IF	pc_jump: Checks if PC is jumping to the target address

5.2 Instruction Decode

Team Member	Design Unit	Test Name
Aalap	ID	Check if ZERO flag is asserted
Aalap	ID	Check if CARRY flag is asserted
Aalap	ID	Validating IDU instructions

5.3 Instruction Execute

Team Member	Design Unit	Test Name
Niko	IE	Validate ALU operations such as ADD and SUB
Niko	IE	Validate the ZERO flag is asserted correctly
Niko	IE	Validate the CARRY flag is asserted correctly
Niko	IE	Validate the shift_bit is asserted correctly
Niko	IE	Validate write and read strobe
Niko	IE	Validate scratch write enable
Niko	IE	Validate register write enable
Niko	IE	Validate ALU operand_a and operand_b are correctly asserted

5.4 Memory Access

Team Member	Design Unit	Test Name
Aalap	MA	To ensure correct memory access, we set an assertion to match the Scratchpad boundary.
Aalap	MA	Check Scratch Write gets asserted after STORE
Aalap	MA	Check Register Write gets asserted during STORE
Aalap	MA	Check WR Strobe gets asserted when writing to Scratch Register

6 Decoded Bugs

1. ALU not setting the Zero Flag
2. ALU not performing correctly ADD and SUB
3. ALU not setting the Carry Flag correctly
4. STORE not storing the data in the scratchpad due to decrementing incorrectly and ALU wrong Address calculation (later, we have explained it was due to the stack pointer)

7 Challenges faced

1. PicoBlaze architecture and its functioning were difficult to understand.
2. Eliminating false positives from the assertions was also a crucial step.
3. Coordinating remotely
4. Given the limited amount of time, it wasn't easy to decide what we should test first
5. CAT infrastructure goes down or slows down often.

8 Learnings

In this project, we learned how to work as a team. Moreover, all of us had a different backgrounds. For example, Aalap's experience is in Machine / Deep learning. Hence, we all came from backgrounds that did not entirely relate to RTL design. Moreover, all of us already had busy schedules due to internships and work, leaving us little time to explore SVA further.

Overall, we all learned what it takes to write SVAs. In addition, we realized that SystemVerilog is not C. While the code can make tons of sense, and in the case of assertions, even when we had correctly written our assertions, details like clock edge, timing, signal propagation, and overall hardware behavior made the testing challenging.

1. We got to work collectively in the GitHub classroom version control environment.
2. Working together remotely, exchanging ideas, and collaborating helped all three of us to get better insights into the PicoBlaze, writing the test cases and assertions.
3. Timing is everything when it comes to digital design and testing

9 Transcripts

9.1 No Bugs Transcript

```
# vsim -c work.alt_rojo_tb work.blockram work.blockram_sv_unit work.disassembler work.disassembler_sv_unit work.kcpsm_register_sv_unit
work.kcpsm_rojo_sv_unit work.kcpsmx work.kcpsmx3_idu_sv_unit work.kcpsmx3_inc work.kcpsmx_alu work.kcpsmx_alu_sv_unit work.kcpsmx_idu
work.kcpsmx_register work.kcpsmx_scratch work.kcpsmx_scratch_sv_unit work.kcpsmx_stack work.kcpsmx_stack_sv_unit work.rojoblaze_defs
work.tb_testprogs_sv_unit
# Start time: 14:18:39 on Dec 06,2022
# ** Note: (vsim-3812) Design is being optimized...
# //  Questa Intel Starter FPGA Edition-64
# //  Version 2022.1 linux_x86_64 Jan 29 2022
# //
# //  Copyright 1991-2022 Mentor Graphics Corporation
# //  All Rights Reserved.
# //
# //  QuestaSim and its associated documentation contain trade
# //  secrets and commercial or financial information that are the property of
# //  Mentor Graphics Corporation and are privileged, confidential,
# //  and exempt from disclosure under the Freedom of Information Act,
# //  5 U.S.C. Section 552. Furthermore, this information
# //  is prohibited from disclosure under the Trade Secrets Act,
# //  18 U.S.C. Section 1905.
# //
# Loading sv_std.std
# Loading work.kcpsmx3_inc(fast)
# Loading work.rojoblaze_defs(fast)
# Loading work.disassembler_sv_unit(fast)
# Loading work.kcpsmx_scratch_sv_unit(fast)
# Loading work.kcpsmx_stack_sv_unit(fast)
# Loading work.kcpsmx_alu_sv_unit(fast)
# Loading work.kcpsm_register_sv_unit(fast)
# Loading work.kcpsmx3_idu_sv_unit(fast)
# Loading work.blockram_sv_unit(fast)
# Loading work.kcpsm_rojo_sv_unit(fast)
# Loading work.tb_testprogs_sv_unit(fast)
# Loading work.alt_rojo_tb(fast)
# Loading work.blockram(fast)
# Loading work.disassembler(fast)
# Loading work.kcpsmx(fast)
# Loading work.kcpsmx_alu(fast)
# Loading work.kcpsmx_idu(fast)
# Loading work.kcpsmx_register(fast)
# Loading work.kcpsmx_scratch(fast)
# Loading work.kcpsmx_stack(fast)
# ----------------------------------------------------------------------------------------------------------------------------------
# [T: 0 ] || OP: Invalid Instruction || ZERO: x || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP:  || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: x || RG_WR_EN: x || WR_SRB: x || ZR_CR_WR_EN: x x
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++
# Running test file alu_test.mem
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++
# [T: 5 ] || OP: Invalid Instruction || ZERO: x || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP:  || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# [T: 15 ] || OP: Invalid Instruction || ZERO: 0 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP:  || REG_DATA: 0 || ADD_SUB: 00 || OP1: 00 || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ----------------------------------------------------------------------------------------------------------------------------------
# [T: 75 ] || OP: LOAD s3,00  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP:  || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0
|| RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ----------------------------------------------------------------------------------------------------------------------------------
# [T: 85 ] || OP: ADD  s3,00  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: xx || OP1: xx || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
```

```
# ------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 95 ] || OP: LOAD s0,s0  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: 00 || OP1: 00 || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
# ------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 105 ] || OP: JUMP 004  || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 115 ] || OP: LOAD s3,00  || ZERO: 1 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: RR_SLX || REG_DATA: 4 || ADD_SUB: xx || OP1: xx || OP2: 04 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# [T: 125 ] || OP: LOAD s3,00  || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: 00 || OP1: 00 || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 135 ] || OP: ADD  s3,00  || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: 00 || OP1: 00 || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 145 ] || OP: LOAD s0,s0  || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: 00 || OP1: 00 || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
# ------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 155 ] || OP: JUMP 008  || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 165 ] || OP: LOAD s3,FF  || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 8 || ADD_SUB: xx || OP1: xx || OP2: 08 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# [T: 175 ] || OP: LOAD s3,FF  || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: ff || OP1: 00 || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 185 ] || OP: ADD  s3,01  || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: fe || OP1: ff || OP2: ff || SCR_WR_EN:
0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 195 ] || OP: LOAD s0,s0  || ZERO: 1 || CARRY: 1 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: 00 || OP1: ff || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
# ------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 205 ] || OP: JUMP 00C  || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 215 ] || OP: LOAD s3,FF  || ZERO: 1 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: RR_SLX || REG_DATA: c || ADD_SUB: xx || OP1: xx || OP2: 0c ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# [T: 225 ] || OP: LOAD s3,FF  || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: ff || OP1: 00 || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 235 ] || OP: ADDCYs3,01  || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: fe || OP1: ff || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 245 ] || OP: LOAD s0,s0  || ZERO: 1 || CARRY: 1 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 1 || ADD_SUB: 01 || OP1: ff || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
#  Error: OP1 ff +/- OP2 01 should NOT equal 01, you should ignore the error if OP1 == FF && OP2 == 1
#
#
# ------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 255 ] || OP: JUMP 010  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 265 ] || OP: LOAD s4,00  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 10 || ADD_SUB: xx || OP1: xx || OP2: 10 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# [T: 275 ] || OP: LOAD s4,00  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: xx || OP1: xx || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ------------------------------------------------------------------------------------------------------------------------------------------------
```

[T: 285] || OP: SUB s4,00 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: 00 || OP1: 00 || OP2: 00 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 295] || OP: JUMP 013 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: 00 || OP1: 00 || OP2: 00 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 305] || OP: LOAD s4,00 || ZERO: 1 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: RL_SRX || REG_DATA: 13 || ADD_SUB: xx || OP1: xx || OP2: 13 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 315] || OP: LOAD s4,00 || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: 00 || OP1: 00 || OP2: 00 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 325] || OP: SUB s4,01 || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: 00 || OP1: 00 || OP2: 00 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 335] || OP: JUMP 016 || ZERO: 1 || CARRY: 1 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: ff || ADD_SUB: ff || OP1: 00 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 345] || OP: LOAD s4,00 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SC || REG_DATA: 16 || ADD_SUB: xx || OP1: xx || OP2: 16 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 355] || OP: LOAD s4,00 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: ff || OP1: ff || OP2: 00 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 365] || OP: SUBCYs4,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: 00 || OP1: 00 || OP2: 00 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 375] || OP: JUMP 019 || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: fe || ADD_SUB: fe || OP1: 00 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 385] || OP: AND s4,s4 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 19 || ADD_SUB: xx || OP1: xx || OP2: 19 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 395] || OP: AND s4,s4 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: fe || ADD_SUB: fc || OP1: fe || OP2: fe || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 405] || OP: JUMP 01B || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: fe || ADD_SUB: fc || OP1: fe || OP2: fe || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 415] || OP: LOAD sA,00 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: RL_SRX || REG_DATA: 1b || ADD_SUB: xx || OP1: xx || OP2: 1b || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 425] || OP: LOAD sA,00 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: xx || OP1: xx || OP2: 00 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 435] || OP: COMPARE sA,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: 00 || OP1: 00 || OP2: 00 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 445] || OP: JUMP 01E || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: ff || ADD_SUB: ff || OP1: 00 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 455] || OP: LOAD sB,05 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SC || REG_DATA: 1e || ADD_SUB: xx || OP1: xx || OP2: 1e || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 465] || OP: LOAD sB,05 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: RR_SLX || REG_DATA: 5 || ADD_SUB: xx || OP1: xx || OP2: 05 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 475] || OP: TEST sB,04 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RR_SLX || REG_DATA: 5 || ADD_SUB: 0a || OP1: 05 || OP2: 05 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 485] || OP: JUMP 021 || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 1 || SHIFT_OP: RR_SLX || REG_DATA: 4 || ADD_SUB: 09 || OP1: 05 || OP2: 04 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 495] || OP: TEST s0,FF || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 21 || ADD_SUB: xx || OP1: xx || OP2: 21 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 505] || OP: TEST s0,FF || ZERO: 0 || CARRY: x || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: ff || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 515] || OP: JUMP 023 || ZERO: x || CARRY: x || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: ff || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 525] || OP: LOAD sC,00 || ZERO: x || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: RL_SRX || REG_DATA: 23 || ADD_SUB: xx || OP1: xx || OP2: 23 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 535] || OP: LOAD sC,00 || ZERO: x || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: xx || OP1: xx || OP2: 00 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 545] || OP: COMPARE sC,00 || ZERO: x || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: 00 || OP1: 00 || OP2: 00 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 555] || OP: LOAD s0,s0 || ZERO: x || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: 00 || OP1: 00 || OP2: 00 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 565] || OP: COMPARE sC,01 || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 575] || OP: JUMP 028 || ZERO: 1 || CARRY: 1 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: ff || ADD_SUB: ff || OP1: 00 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 585] || OP: LOAD s8,FF || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 28 || ADD_SUB: xx || OP1: xx || OP2: 28 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 595] || OP: LOAD s8,FF || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: xx || OP1: xx || OP2: ff || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 605] || OP: SL0 s8 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: fe || OP1: ff || OP2: ff || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 615] || OP: JUMP 02B || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 0 || SHIFT_OP: SC || REG_DATA: fe || ADD_SUB: 05 || OP1: ff || OP2: 06 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 625] || OP: LOAD s9,FF || ZERO: 0 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: RL_SRX || REG_DATA: 2b || ADD_SUB: xx || OP1: xx || OP2: 2b || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 635] || OP: LOAD s9,FF || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: xx || OP1: xx || OP2: ff || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 645] || OP: SL1 s9 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: fe || OP1: ff || OP2: ff || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 655] || OP: JUMP 02E || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: 06 || OP1: ff || OP2: 07 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 665] || OP: LOAD s9,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SC || REG_DATA: 2e || ADD_SUB: xx || OP1: xx || OP2: 2e || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 675] || OP: LOAD s9,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 1 || ADD_SUB: 00 || OP1: ff || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 685] || OP: SR1 s9 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 1 || ADD_SUB: 02 || OP1: 01 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 695] || OP: JUMP 031 || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: 80 || ADD_SUB: 10 || OP1: 01 || OP2: 0f || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 705] || OP: LOAD sD,FF || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 31 || ADD_SUB: xx || OP1: xx || OP2: 31 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 715] || OP: LOAD sD,FF || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: xx || OP1: xx || OP2: ff || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

--
[T: 725] || OP: RL sD || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: fe || OP1: ff || OP2: ff || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 735] || OP: JUMP 034 || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: ff || ADD_SUB: 01 || OP1: ff || OP2: 02 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
--
[T: 745] || OP: LOAD sD,FF || ZERO: 0 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: RR_SLX || REG_DATA: 34 || ADD_SUB: xx || OP1: xx || OP2: 34 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 755] || OP: LOAD sD,FF || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: fe || OP1: ff || OP2: ff || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 765] || OP: RR sD || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: fe || OP1: ff || OP2: ff || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 775] || OP: JUMP 037 || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 1 || SHIFT_OP: RR_SLX || REG_DATA: ff || ADD_SUB: 0b || OP1: ff || OP2: 0c || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
--
[T: 785] || OP: LOAD s2,40 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: 37 || ADD_SUB: xx || OP1: xx || OP2: 37 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 795] || OP: LOAD s2,40 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 40 || ADD_SUB: xx || OP1: xx || OP2: 40 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 805] || OP: LOAD s1,00 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 40 || ADD_SUB: 80 || OP1: 40 || OP2: 40 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 815] || OP: JUMP 03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: xx || OP1: xx || OP2: 00 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 825] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: RL_SRX || REG_DATA: 3a || ADD_SUB: xx || OP1: xx || OP2: 3a || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 835] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 845] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 855] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 3f || ADD_SUB: 3f || OP1: 40 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
--
[T: 865] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 3f || ADD_SUB: 7e || OP1: 3f || OP2: 3f || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 875] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 885] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 895] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 905] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 3e || ADD_SUB: 3e || OP1: 3f || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
--
[T: 915] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 3e || ADD_SUB: 7c || OP1: 3e || OP2: 3e || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 925] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 935] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||

SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 945] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 955] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 3d || ADD_SUB: 3d || OP1: 3e || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 965] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 3d || ADD_SUB: 7a || OP1: 3d || OP2: 3d ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 975] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 985] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 995] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1005] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 3c || ADD_SUB: 3c || OP1: 3d || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1015] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 3c || ADD_SUB: 78 || OP1: 3c || OP2: 3c ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1025] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1035] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1045] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1055] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 3b || ADD_SUB: 3b || OP1: 3c || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1065] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 3b || ADD_SUB: 76 || OP1: 3b || OP2: 3b ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1075] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1085] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1095] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1105] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 3a || ADD_SUB: 3a || OP1: 3b || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1115] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 3a || ADD_SUB: 74 || OP1: 3a || OP2: 3a ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1125] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1135] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1145] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1155] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 39 || ADD_SUB: 39 || OP1: 3a || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1165] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 39 || ADD_SUB: 72 || OP1: 39 || OP2: 39 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1175] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1185] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1195] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1205] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 38 || ADD_SUB: 38 || OP1: 39 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1215] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 38 || ADD_SUB: 70 || OP1: 38 || OP2: 38 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1225] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1235] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1245] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1255] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 37 || ADD_SUB: 37 || OP1: 38 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1265] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 37 || ADD_SUB: 6e || OP1: 37 || OP2: 37 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1275] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1285] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1295] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1305] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 36 || ADD_SUB: 36 || OP1: 37 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1315] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 36 || ADD_SUB: 6c || OP1: 36 || OP2: 36 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1325] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1335] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1345] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1355] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 35 || ADD_SUB: 35 || OP1: 36 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1365] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 35 || ADD_SUB: 6a || OP1: 35 || OP2: 35 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1375] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1385] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1395] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1405] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 34 || ADD_SUB: 34 || OP1: 35 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1415] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 34 || ADD_SUB: 68 || OP1: 34 || OP2: 34 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1425] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1435] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1445] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1455] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 33 || ADD_SUB: 33 || OP1: 34 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1465] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 33 || ADD_SUB: 66 || OP1: 33 || OP2: 33 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1475] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1485] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1495] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1505] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 32 || ADD_SUB: 32 || OP1: 33 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1515] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 32 || ADD_SUB: 64 || OP1: 32 || OP2: 32 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1525] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1535] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1545] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1555] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 31 || ADD_SUB: 31 || OP1: 32 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1565] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 31 || ADD_SUB: 62 || OP1: 31 || OP2: 31 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1575] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||

SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1585] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1595] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1605] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 30 || ADD_SUB: 30 || OP1: 31 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1615] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 30 || ADD_SUB: 60 || OP1: 30 || OP2: 30 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1625] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1635] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1645] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1655] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 2f || ADD_SUB: 2f || OP1: 30 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1665] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 2f || ADD_SUB: 5e || OP1: 2f || OP2: 2f ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1675] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1685] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1695] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1705] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 2e || ADD_SUB: 2e || OP1: 2f || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1715] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 2e || ADD_SUB: 5c || OP1: 2e || OP2: 2e ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1725] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1735] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1745] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1755] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 2d || ADD_SUB: 2d || OP1: 2e || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1765] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 2d || ADD_SUB: 5a || OP1: 2d || OP2: 2d ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1775] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1785] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1795] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1805] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 2c || ADD_SUB: 2c || OP1: 2d || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1815] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 2c || ADD_SUB: 58 || OP1: 2c || OP2: 2c || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1825] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1835] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1845] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1855] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 2b || ADD_SUB: 2b || OP1: 2c || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1865] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 2b || ADD_SUB: 56 || OP1: 2b || OP2: 2b || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1875] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1885] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1895] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1905] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 2a || ADD_SUB: 2a || OP1: 2b || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1915] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 2a || ADD_SUB: 54 || OP1: 2a || OP2: 2a || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1925] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1935] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1945] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1955] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 29 || ADD_SUB: 29 || OP1: 2a || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 1965] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 29 || ADD_SUB: 52 || OP1: 29 || OP2: 29 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1975] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 1985] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1995] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2005] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 28 || ADD_SUB: 28 || OP1: 29 || OP2: 01 ||

SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 2015] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 28 || ADD_SUB: 50 || OP1: 28 || OP2: 28 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2025] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2035] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2045] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2055] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 27 || ADD_SUB: 27 || OP1: 28 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 2065] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 27 || ADD_SUB: 4e || OP1: 27 || OP2: 27 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2075] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2085] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2095] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2105] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 26 || ADD_SUB: 26 || OP1: 27 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 2115] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 26 || ADD_SUB: 4c || OP1: 26 || OP2: 26 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2125] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2135] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2145] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2155] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 25 || ADD_SUB: 25 || OP1: 26 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 2165] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 25 || ADD_SUB: 4a || OP1: 25 || OP2: 25 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2175] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2185] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2195] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2205] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 24 || ADD_SUB: 24 || OP1: 25 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 2215] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 24 || ADD_SUB: 48 || OP1: 24 || OP2: 24 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

--
[T: 2225] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2235] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 2245] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 2255] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 23 || ADD_SUB: 23 || OP1: 24 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
--
[T: 2265] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 23 || ADD_SUB: 46 || OP1: 23 || OP2: 23 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 2275] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2285] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 2295] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 2305] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 22 || ADD_SUB: 22 || OP1: 23 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
--
[T: 2315] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 22 || ADD_SUB: 44 || OP1: 22 || OP2: 22 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 2325] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2335] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 2345] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 2355] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 21 || ADD_SUB: 21 || OP1: 22 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
--
[T: 2365] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 21 || ADD_SUB: 42 || OP1: 21 || OP2: 21 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 2375] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2385] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 2395] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 2405] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 20 || ADD_SUB: 20 || OP1: 21 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
--
[T: 2415] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 20 || ADD_SUB: 40 || OP1: 20 || OP2: 20 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 2425] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2435] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||

SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2445] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2455] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 1f || ADD_SUB: 1f || OP1: 20 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 2465] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 1f || ADD_SUB: 3e || OP1: 1f || OP2: 1f ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2475] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2485] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2495] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2505] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 1e || ADD_SUB: 1e || OP1: 1f || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 2515] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 1e || ADD_SUB: 3c || OP1: 1e || OP2: 1e ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2525] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2535] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2545] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2555] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 1d || ADD_SUB: 1d || OP1: 1e || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 2565] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 1d || ADD_SUB: 3a || OP1: 1d || OP2: 1d ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2575] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2585] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2595] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2605] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 1c || ADD_SUB: 1c || OP1: 1d || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 2615] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 1c || ADD_SUB: 38 || OP1: 1c || OP2: 1c ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2625] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2635] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2645] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2655] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 1b || ADD_SUB: 1b || OP1: 1c || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 2665] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 1b || ADD_SUB: 36 || OP1: 1b || OP2: 1b || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2675] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2685] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2695] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2705] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 1a || ADD_SUB: 1a || OP1: 1b || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 2715] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 1a || ADD_SUB: 34 || OP1: 1a || OP2: 1a || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2725] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2735] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2745] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2755] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 19 || ADD_SUB: 19 || OP1: 1a || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 2765] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 19 || ADD_SUB: 32 || OP1: 19 || OP2: 19 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2775] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2785] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2795] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2805] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 18 || ADD_SUB: 18 || OP1: 19 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 2815] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 18 || ADD_SUB: 30 || OP1: 18 || OP2: 18 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2825] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2835] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2845] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2855] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 17 || ADD_SUB: 17 || OP1: 18 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 2865] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 17 || ADD_SUB: 2e || OP1: 17 || OP2: 17 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2875] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2885] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2895] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2905] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 16 || ADD_SUB: 16 || OP1: 17 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 2915] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 16 || ADD_SUB: 2c || OP1: 16 || OP2: 16 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2925] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2935] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2945] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2955] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 15 || ADD_SUB: 15 || OP1: 16 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 2965] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 15 || ADD_SUB: 2a || OP1: 15 || OP2: 15 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2975] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 2985] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 2995] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3005] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 14 || ADD_SUB: 14 || OP1: 15 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 3015] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 14 || ADD_SUB: 28 || OP1: 14 || OP2: 14 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3025] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3035] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3045] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3055] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 13 || ADD_SUB: 13 || OP1: 14 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 3065] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 13 || ADD_SUB: 26 || OP1: 13 || OP2: 13 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3075] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||

SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3085] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3095] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3105] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 12 || ADD_SUB: 12 || OP1: 13 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 3115] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 12 || ADD_SUB: 24 || OP1: 12 || OP2: 12 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3125] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3135] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3145] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3155] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 11 || ADD_SUB: 11 || OP1: 12 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 3165] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 11 || ADD_SUB: 22 || OP1: 11 || OP2: 11 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3175] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3185] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3195] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3205] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 10 || ADD_SUB: 10 || OP1: 11 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 3215] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 10 || ADD_SUB: 20 || OP1: 10 || OP2: 10 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3225] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3235] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3245] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3255] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: f || ADD_SUB: 0f || OP1: 10 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 3265] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: f || ADD_SUB: 1e || OP1: 0f || OP2: 0f ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3275] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3285] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3295] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3305] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: e || ADD_SUB: 0e || OP1: 0f || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 3315] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: e || ADD_SUB: 1c || OP1: 0e || OP2: 0e || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3325] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3335] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3345] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3355] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: d || ADD_SUB: 0d || OP1: 0e || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 3365] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: d || ADD_SUB: 1a || OP1: 0d || OP2: 0d || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3375] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3385] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3395] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3405] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: c || ADD_SUB: 0c || OP1: 0d || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 3415] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: c || ADD_SUB: 18 || OP1: 0c || OP2: 0c || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3425] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3435] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3445] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3455] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: b || ADD_SUB: 0b || OP1: 0c || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 3465] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: b || ADD_SUB: 16 || OP1: 0b || OP2: 0b || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3475] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3485] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3495] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3505] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: a || ADD_SUB: 0a || OP1: 0b || OP2: 01 ||

SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 3515] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: a || ADD_SUB: 14 || OP1: 0a || OP2: 0a ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3525] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3535] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3545] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3555] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 9 || ADD_SUB: 09 || OP1: 0a || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 3565] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 9 || ADD_SUB: 12 || OP1: 09 || OP2: 09 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3575] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3585] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3595] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3605] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 8 || ADD_SUB: 08 || OP1: 09 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 3615] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 8 || ADD_SUB: 10 || OP1: 08 || OP2: 08 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3625] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3635] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3645] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3655] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 7 || ADD_SUB: 07 || OP1: 08 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 3665] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 7 || ADD_SUB: 0e || OP1: 07 || OP2: 07 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3675] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3685] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3695] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3705] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 6 || ADD_SUB: 06 || OP1: 07 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 3715] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 6 || ADD_SUB: 0c || OP1: 06 || OP2: 06 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

--
[T: 3725] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3735] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 3745] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 3755] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 5 || ADD_SUB: 05 || OP1: 06 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
--
[T: 3765] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 5 || ADD_SUB: 0a || OP1: 05 || OP2: 05 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 3775] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3785] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 3795] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 3805] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 4 || ADD_SUB: 04 || OP1: 05 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
--
[T: 3815] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 4 || ADD_SUB: 08 || OP1: 04 || OP2: 04 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 3825] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3835] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 3845] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 3855] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 3 || ADD_SUB: 03 || OP1: 04 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
--
[T: 3865] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 3 || ADD_SUB: 06 || OP1: 03 || OP2: 03 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 3875] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3885] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 3895] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 3905] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 2 || ADD_SUB: 02 || OP1: 03 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
--
[T: 3915] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 2 || ADD_SUB: 04 || OP1: 02 || OP2: 02 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 3925] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3935] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||

SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3945] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3955] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 1 || ADD_SUB: 01 || OP1: 02 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 3965] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 1 || ADD_SUB: 02 || OP1: 01 || OP2: 01 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3975] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 3985] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 3995] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 4005] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: 00 || OP1: 01 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 4015] || OP: JUMP NZ,03A || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: 00 || OP1: 00 || OP2: 00 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 4025] || OP: LOAD s0,s0 || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: 1 || ADD_SUB: ff || OP1: fe || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 4035] || OP: RETURN || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 4045] || OP: Invalid Instruction || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: xx || OP1: xx || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 4055] || OP: Invalid Instruction || ZERO: 1 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
Test Program concluded.
++
Scratch Pad Mem Values:
++
RAM[0] = 00

RAM[1] = 01

RAM[2] = 02

RAM[3] = 03

RAM[4] = 04

RAM[5] = 05

RAM[6] = 06

RAM[7] = 07

RAM[8] = 08

RAM[9] = 09

RAM[10] = 0a

RAM[11] = 0b

RAM[12] = 0c

RAM[13] = 0d

RAM[14] = 0e

RAM[15] = 0f

RAM[16] = 10

RAM[17] = 11

RAM[18] = 12

RAM[19] = 13

RAM[20] = 14

RAM[21] = 15

RAM[22] = 16

RAM[23] = 17

RAM[24] = 18

RAM[25] = 19

RAM[26] = 1a

RAM[27] = 1b

RAM[28] = 1c

RAM[29] = 1d

RAM[30] = 1e

RAM[31] = 1f

RAM[32] = 20

RAM[33] = 21

RAM[34] = 22

RAM[35] = 23

RAM[36] = 24

RAM[37] = 25

```
# RAM[38] = 26
# ----------------------------------------------------------------------------
# RAM[39] = 27
# ----------------------------------------------------------------------------
# RAM[40] = 28
# ----------------------------------------------------------------------------
# RAM[41] = 29
# ----------------------------------------------------------------------------
# RAM[42] = 2a
# ----------------------------------------------------------------------------
# RAM[43] = 2b
# ----------------------------------------------------------------------------
# RAM[44] = 2c
# ----------------------------------------------------------------------------
# RAM[45] = 2d
# ----------------------------------------------------------------------------
# RAM[46] = 2e
# ----------------------------------------------------------------------------
# RAM[47] = 2f
# ----------------------------------------------------------------------------
# RAM[48] = 30
# ----------------------------------------------------------------------------
# RAM[49] = 31
# ----------------------------------------------------------------------------
# RAM[50] = 32
# ----------------------------------------------------------------------------
# RAM[51] = 33
# ----------------------------------------------------------------------------
# RAM[52] = 34
# ----------------------------------------------------------------------------
# RAM[53] = 35
# ----------------------------------------------------------------------------
# RAM[54] = 36
# ----------------------------------------------------------------------------
# RAM[55] = 37
# ----------------------------------------------------------------------------
# RAM[56] = 38
# ----------------------------------------------------------------------------
# RAM[57] = 39
# ----------------------------------------------------------------------------
# RAM[58] = 3a
# ----------------------------------------------------------------------------
# RAM[59] = 3b
# ----------------------------------------------------------------------------
# RAM[60] = 3c
# ----------------------------------------------------------------------------
# RAM[61] = 3d
# ----------------------------------------------------------------------------
# RAM[62] = 3e
# ----------------------------------------------------------------------------
# RAM[63] = 3f
# ----------------------------------------------------------------------------
#
================================================================================
=========
# ** Note: $stop    : /home/niko/Downloads/fnlproj-niko-kathy-aalap/team-demo/no_bug_pipelined_picoblaze_hdl/tb_testprogs.sv(349)
#    Time: 6440 ns  Iteration: 1  Instance: /alt_rojo_tb
# Break at /home/niko/Downloads/fnlproj-niko-kathy-aalap/team-demo/no_bug_pipelined_picoblaze_hdl/tb_testprogs.sv line 349
```

```
# Stopped at /home/niko/Downloads/fnlproj-niko-kathy-aalap/team-demo/no_bug_pipelined_picoblaze_hdl/tb_testprogs.sv line 349
# End time: 14:19:43 on Dec 06,2022, Elapsed time: 0:01:04
# Errors: 0, Warnings: 0
```

9.2 With Bugs Transcript

```
# vsim -c work.alt_rojo_tb work.blockram work.blockram_sv_unit work.disassembler work.disassembler_sv_unit work.kcpsm_register_sv_unit
work.kcpsm_rojo_sv_unit work.kcpsmx work.kcpsmx3_idu_sv_unit work.kcpsmx3_inc work.kcpsmx_alu work.kcpsmx_alu_sv_unit work.kcpsmx_idu
work.kcpsmx_register work.kcpsmx_scratch work.kcpsmx_scratch_sv_unit work.kcpsmx_stack work.kcpsmx_stack_sv_unit work.rojoblaze_defs
work.tb_testprogs_sv_unit
# Start time: 14:18:39 on Dec 06,2022
# ** Note: (vsim-3812) Design is being optimized...
# //  Questa Intel Starter FPGA Edition-64
# //  Version 2022.1 linux_x86_64 Jan 29 2022
# //
# //  Copyright 1991-2022 Mentor Graphics Corporation
# //  All Rights Reserved.
# //
# //  QuestaSim and its associated documentation contain trade
# //  secrets and commercial or financial information that are the property of
# //  Mentor Graphics Corporation and are privileged, confidential,
# //  and exempt from disclosure under the Freedom of Information Act,
# //  5 U.S.C. Section 552. Furthermore, this information
# //  is prohibited from disclosure under the Trade Secrets Act,
# //  18 U.S.C. Section 1905.
# //
# Loading sv_std.std
# Loading work.kcpsmx3_inc(fast)
# Loading work.rojoblaze_defs(fast)
# Loading work.disassembler_sv_unit(fast)
# Loading work.kcpsmx_scratch_sv_unit(fast)
# Loading work.kcpsmx_stack_sv_unit(fast)
# Loading work.kcpsmx_alu_sv_unit(fast)
# Loading work.kcpsm_register_sv_unit(fast)
# Loading work.kcpsmx3_idu_sv_unit(fast)
# Loading work.blockram_sv_unit(fast)
# Loading work.kcpsm_rojo_sv_unit(fast)
# Loading work.tb_testprogs_sv_unit(fast)
# Loading work.alt_rojo_tb(fast)
# Loading work.blockram(fast)
# Loading work.disassembler(fast)
# Loading work.kcpsmx(fast)
# Loading work.kcpsmx_alu(fast)
# Loading work.kcpsmx_idu(fast)
# Loading work.kcpsmx_register(fast)
# Loading work.kcpsmx_scratch(fast)
# Loading work.kcpsmx_stack(fast)
# ----------------------------------------------------------------------------------------------------------------------------------
# [T: 0 ] || OP: Invalid Instruction || ZERO: x || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP:  || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: x || RG_WR_EN: x || WR_SRB: x || ZR_CR_WR_EN: x x
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++
# Running test file alu_test.mem
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++
# [T: 5 ] || OP: Invalid Instruction || ZERO: x || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP:  || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# [T: 15 ] || OP: Invalid Instruction || ZERO: 0 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP:  || REG_DATA: 0 || ADD_SUB: 00 || OP1: 00 || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ----------------------------------------------------------------------------------------------------------------------------------
# [T: 75 ] || OP: LOAD s3,00  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP:  || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0
|| RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ----------------------------------------------------------------------------------------------------------------------------------
# [T: 85 ] || OP: ADD  s3,00  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: xx || OP1: xx || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
```

```
# ------------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 95 ] || OP: LOAD s0,s0  || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: ff || ADD_SUB: ff || OP1: 00 || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
# ------------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 105 ] || OP: JUMP 004  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ------------------------------------------------------------------------------------------------------------------------------------------------------
#
#
#  ALU Result was 0; Error OP:          LOAD s3,00  , Zero flag was not set [ 0 ], FAIL
#
#
# [T: 115 ] || OP: LOAD s3,00  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: RR_SLX || REG_DATA: 4 || ADD_SUB: xx || OP1: xx || OP2: 04 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
#  ALU Result was 0; Error OP:          LOAD s3,00  , Zero flag was not set [ 0 ], FAIL
#
#
# [T: 125 ] || OP: LOAD s3,00  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: fe || OP1: ff || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ------------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 135 ] || OP: ADD  s3,00  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: ff || OP1: 00 || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ------------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 145 ] || OP: LOAD s0,s0  || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: ff || ADD_SUB: ff || OP1: 00 || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
# ------------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 155 ] || OP: JUMP 008  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ------------------------------------------------------------------------------------------------------------------------------------------------------
#
#
#  ALU Result was 0; Error OP:          LOAD s3,FF  , Zero flag was not set [ 0 ], FAIL
#
#
# [T: 165 ] || OP: LOAD s3,FF  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 8 || ADD_SUB: xx || OP1: xx || OP2: 08 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
#  ALU Result was 0; Error OP:          LOAD s3,FF  , Zero flag was not set [ 0 ], FAIL
#
#
# [T: 175 ] || OP: LOAD s3,FF  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: fd || OP1: ff || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
#  Error: OP1 ff +/- OP2 ff should NOT equal fd, you should ignore the error if OP1 == FF && OP2 == 1
#
#
# ------------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 185 ] || OP: ADD  s3,01  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: fd || OP1: ff || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
#  Error: OP1 ff +/- OP2 ff should NOT equal fd, you should ignore the error if OP1 == FF && OP2 == 1
#
#
# ------------------------------------------------------------------------------------------------------------------------------------------------------
```

[T: 195] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: ff || ADD_SUB: ff || OP1: ff || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 ff +/- OP2 01 should NOT equal ff, you should ignore the error if OP1 == FF && OP2 == 1
#
#

#
#
ALU was Result > FF ; Error OP: JUMP 00C , ALU Carry Out was not set [0], FAIL
#
#
[T: 205] || OP: JUMP 00C || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 215] || OP: LOAD s3,FF || ZERO: 0 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: RR_SLX || REG_DATA: c || ADD_SUB: xx || OP1: xx || OP2: 0c || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 225] || OP: LOAD s3,FF || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: fd || OP1: ff || OP2: ff || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 ff +/- OP2 ff should NOT equal fd, you should ignore the error if OP1 == FF && OP2 == 1
#
#

[T: 235] || OP: ADDCYs3,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: fd || OP1: ff || OP2: ff || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 ff +/- OP2 ff should NOT equal fd, you should ignore the error if OP1 == FF && OP2 == 1
#
#

[T: 245] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: ff || ADD_SUB: ff || OP1: ff || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 ff +/- OP2 01 should NOT equal ff, you should ignore the error if OP1 == FF && OP2 == 1
#
#

#
#
ALU Result > FF; Error OP: JUMP 010 , ALU Carry Out was not set [0], FAIL
#
#
[T: 255] || OP: JUMP 010 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 265] || OP: LOAD s4,00 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 10 || ADD_SUB: xx || OP1: xx || OP2: 10 || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 275] || OP: LOAD s4,00 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: xx || OP1: xx || OP2: 00 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 285] || OP: SUB s4,00 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: ff || OP1: 00 || OP2: 00 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 295] || OP: JUMP 013 || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: fd || ADD_SUB: fd || OP1: 00 || OP2: 00 ||

SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 305] || OP: LOAD s4,00 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: RL_SRX || REG_DATA: 13 || ADD_SUB: xx || OP1: xx || OP2: 13 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 315] || OP: LOAD s4,00 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: fc || OP1: fd || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 325] || OP: SUB s4,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: ff || OP1: 00 || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 335] || OP: JUMP 016 || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: fc || ADD_SUB: fc || OP1: 00 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 345] || OP: LOAD s4,00 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SC || REG_DATA: 16 || ADD_SUB: xx || OP1: xx || OP2: 16 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 355] || OP: LOAD s4,00 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: fb || OP1: fc || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 365] || OP: SUBCYs4,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: ff || OP1: 00 || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 375] || OP: JUMP 019 || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: fd || ADD_SUB: fd || OP1: 00 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 385] || OP: AND s4,s4 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 19 || ADD_SUB: xx || OP1: xx || OP2: 19 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 395] || OP: AND s4,s4 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: fd || ADD_SUB: f9 || OP1: fd || OP2: fd ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 fd +/- OP2 fd should NOT equal f9, you should ignore the error if OP1 == FF && OP2 == 1
#
#

[T: 405] || OP: JUMP 01B || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: fd || ADD_SUB: f9 || OP1: fd || OP2: fd ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 fd +/- OP2 fd should NOT equal f9, you should ignore the error if OP1 == FF && OP2 == 1
#
#

[T: 415] || OP: LOAD sA,00 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: RL_SRX || REG_DATA: 1b || ADD_SUB: xx || OP1: xx || OP2: 1b ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 425] || OP: LOAD sA,00 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: xx || OP1: xx || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 435] || OP: COMPARE sA,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: ff || OP1: 00 || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 445] || OP: JUMP 01E || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: fc || ADD_SUB: fc || OP1: 00 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0

[T: 455] || OP: LOAD sB,05 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SC || REG_DATA: 1e || ADD_SUB: xx || OP1: xx || OP2: 1e ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 465] || OP: LOAD sB,05 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: RR_SLX || REG_DATA: 5 || ADD_SUB: xx || OP1: xx || OP2: 05 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 475] || OP: TEST sB,04 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RR_SLX || REG_DATA: 5 || ADD_SUB: 09 || OP1: 05 || OP2: 05 ||

SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 05 +/- OP2 05 should NOT equal 09, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--
[T: 485] || OP: JUMP 021 || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 1 || SHIFT_OP: RR_SLX || REG_DATA: 4 || ADD_SUB: 08 || OP1: 05 || OP2: 04 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 05 +/- OP2 04 should NOT equal 08, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--
[T: 495] || OP: TEST s0,FF || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 21 || ADD_SUB: xx || OP1: xx || OP2: 21 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 505] || OP: TEST s0,FF || ZERO: 0 || CARRY: x || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
--
[T: 515] || OP: JUMP 023 || ZERO: x || CARRY: x || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: ff || SCR_WR_EN:
0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
--
[T: 525] || OP: LOAD sC,00 || ZERO: x || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: RL_SRX || REG_DATA: 23 || ADD_SUB: xx || OP1: xx || OP2: 23 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 535] || OP: LOAD sC,00 || ZERO: x || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: xx || OP1: xx || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 545] || OP: COMPARE sC,00 || ZERO: x || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: ff || OP1: 00 || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 555] || OP: LOAD s0,s0 || ZERO: x || CARRY: 1 || SHIFT_BIT: x || SHIFT_OP: SA || REG_DATA: fd || ADD_SUB: fd || OP1: 00 || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
--
[T: 565] || OP: COMPARE sC,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 575] || OP: JUMP 028 || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: fc || ADD_SUB: fc || OP1: 00 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
--
[T: 585] || OP: LOAD s8,FF || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 28 || ADD_SUB: xx || OP1: xx || OP2: 28 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 595] || OP: LOAD s8,FF || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: xx || OP1: xx || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 605] || OP: SL0 s8 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: fd || OP1: ff || OP2: ff || SCR_WR_EN: 0 ||
RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 ff +/- OP2 ff should NOT equal fd, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--
[T: 615] || OP: JUMP 02B || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 0 || SHIFT_OP: SC || REG_DATA: fe || ADD_SUB: 04 || OP1: ff || OP2: 06 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 ff +/- OP2 06 should NOT equal 04, you should ignore the error if OP1 == FF && OP2 == 1
#

```
#
# ------------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 625 ] || OP: LOAD s9,FF  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: RL_SRX || REG_DATA: 2b || ADD_SUB: xx || OP1: xx || OP2: 2b ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# [T: 635 ] || OP: LOAD s9,FF  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: xx || OP1: xx || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ------------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 645 ] || OP: SL1 s9  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: fd || OP1: ff || OP2: ff || SCR_WR_EN: 0 ||
RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
#  Error: OP1 ff +/- OP2 ff should NOT equal fd, you should ignore the error if OP1 == FF && OP2 == 1
#
#
# ------------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 655 ] || OP: JUMP 02E  || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: 05 || OP1: ff || OP2: 07 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
#  Error: OP1 ff +/- OP2 07 should NOT equal 05, you should ignore the error if OP1 == FF && OP2 == 1
#
#
# ------------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 665 ] || OP: LOAD s9,01  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SC || REG_DATA: 2e || ADD_SUB: xx || OP1: xx || OP2: 2e ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# [T: 675 ] || OP: LOAD s9,01  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 1 || ADD_SUB: ff || OP1: ff || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
#  Error: OP1 ff +/- OP2 01 should NOT equal ff, you should ignore the error if OP1 == FF && OP2 == 1
#
#
# ------------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 685 ] || OP: SR1 s9  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 1 || ADD_SUB: 01 || OP1: 01 || OP2: 01 || SCR_WR_EN: 0
|| RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
#  Error: OP1 01 +/- OP2 01 should NOT equal 01, you should ignore the error if OP1 == FF && OP2 == 1
#
#
# ------------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 695 ] || OP: JUMP 031  || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: 80 || ADD_SUB: 0f || OP1: 01 || OP2: 0f ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
#  Error: OP1 01 +/- OP2 0f should NOT equal 0f, you should ignore the error if OP1 == FF && OP2 == 1
#
#
# ------------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 705 ] || OP: LOAD sD,FF  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 31 || ADD_SUB: xx || OP1: xx || OP2: 31 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# [T: 715 ] || OP: LOAD sD,FF  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: xx || OP1: xx || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ------------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 725 ] || OP: RL sD  || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: fd || OP1: ff || OP2: ff || SCR_WR_EN: 0 ||
RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
#  Error: OP1 ff +/- OP2 ff should NOT equal fd, you should ignore the error if OP1 == FF && OP2 == 1
```

#
#
--
[T: 735] || OP: JUMP 034 || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: ff || ADD_SUB: 00 || OP1: ff || OP2: 02 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 ff +/- OP2 02 should NOT equal 00, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--
[T: 745] || OP: LOAD sD,FF || ZERO: 0 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: RR_SLX || REG_DATA: 34 || ADD_SUB: xx || OP1: xx || OP2: 34 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 755] || OP: LOAD sD,FF || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: fd || OP1: ff || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 ff +/- OP2 ff should NOT equal fd, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--
[T: 765] || OP: RR sD || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: ff || ADD_SUB: fd || OP1: ff || OP2: ff || SCR_WR_EN: 0 ||
RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 ff +/- OP2 ff should NOT equal fd, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--
[T: 775] || OP: JUMP 037 || ZERO: 0 || CARRY: 1 || SHIFT_BIT: 1 || SHIFT_OP: RR_SLX || REG_DATA: ff || ADD_SUB: 0a || OP1: ff || OP2: 0c ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 ff +/- OP2 0c should NOT equal 0a, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--
[T: 785] || OP: LOAD s2,40 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SC || REG_DATA: 37 || ADD_SUB: xx || OP1: xx || OP2: 37 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 795] || OP: LOAD s2,40 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 40 || ADD_SUB: xx || OP1: xx || OP2: 40 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 805] || OP: LOAD s1,00 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 40 || ADD_SUB: 7f || OP1: 40 || OP2: 40 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 40 +/- OP2 40 should NOT equal 7f, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--
[T: 815] || OP: JUMP 03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: xx || OP1: xx || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 825] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP: RL_SRX || REG_DATA: 3a || ADD_SUB: xx || OP1: xx || OP2: 3a ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
[T: 835] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 845] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||

SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 855] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: SA || REG_DATA: 3c || ADD_SUB: 3c || OP1: 40 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 40 +/- OP2 01 should NOT equal 3c, you should ignore the error if OP1 == FF && OP2 == 1
#
#
#
#
Error: STORE after SUB S2,01 operand_a [OP1] is 40 the next result should be 3f , but is 3c, FAIL
#
#

[T: 865] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 3c || ADD_SUB: 77 || OP1: 3c || OP2: 3c ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 3c +/- OP2 3c should NOT equal 77, you should ignore the error if OP1 == FF && OP2 == 1
#
#

[T: 875] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: ff || ADD_SUB: fb || OP1: fd || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 fd +/- OP2 ff should NOT equal fb, you should ignore the error if OP1 == FF && OP2 == 1
#
#
[T: 885] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 895] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 905] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 38 || ADD_SUB: 38 || OP1: 3c || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 3c +/- OP2 01 should NOT equal 38, you should ignore the error if OP1 == FF && OP2 == 1
#
#
#
#
Error: STORE after SUB S2,01 operand_a [OP1] is 3c the next result should be 3b , but is 38, FAIL
#
#

[T: 915] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 38 || ADD_SUB: 6f || OP1: 38 || OP2: 38 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 38 +/- OP2 38 should NOT equal 6f, you should ignore the error if OP1 == FF && OP2 == 1
#
#

[T: 925] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: ff || ADD_SUB: fb || OP1: fd || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

#
#
Error: OP1 fd +/- OP2 ff should NOT equal fb, you should ignore the error if OP1 == FF && OP2 == 1
#
#
[T: 935] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 945] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 955] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 34 || ADD_SUB: 34 || OP1: 38 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 38 +/- OP2 01 should NOT equal 34, you should ignore the error if OP1 == FF && OP2 == 1
#
#
#
#
Error: STORE after SUB S2,01 operand_a [OP1] is 38 the next result should be 37 , but is 34, FAIL
#
#
--
[T: 965] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 34 || ADD_SUB: 67 || OP1: 34 || OP2: 34 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 34 +/- OP2 34 should NOT equal 67, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--
[T: 975] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: ff || ADD_SUB: fb || OP1: fd || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 fd +/- OP2 ff should NOT equal fb, you should ignore the error if OP1 == FF && OP2 == 1
#
#
[T: 985] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 995] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1005] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 30 || ADD_SUB: 30 || OP1: 34 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 34 +/- OP2 01 should NOT equal 30, you should ignore the error if OP1 == FF && OP2 == 1
#
#
#
#
Error: STORE after SUB S2,01 operand_a [OP1] is 34 the next result should be 33 , but is 30, FAIL
#
#
--
[T: 1015] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 30 || ADD_SUB: 5f || OP1: 30 || OP2: 30 ||

SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 30 +/- OP2 30 should NOT equal 5f, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--
[T: 1025] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: ff || ADD_SUB: fb || OP1: fd || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 fd +/- OP2 ff should NOT equal fb, you should ignore the error if OP1 == FF && OP2 == 1
#
#
[T: 1035] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1045] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1055] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 2c || ADD_SUB: 2c || OP1: 30 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 30 +/- OP2 01 should NOT equal 2c, you should ignore the error if OP1 == FF && OP2 == 1
#
#
#
#
Error: STORE after SUB S2,01 operand_a [OP1] is 30 the next result should be 2f , but is 2c, FAIL
#
#
--
[T: 1065] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 2c || ADD_SUB: 57 || OP1: 2c || OP2: 2c ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 2c +/- OP2 2c should NOT equal 57, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--
[T: 1075] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: ff || ADD_SUB: fb || OP1: fd || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 fd +/- OP2 ff should NOT equal fb, you should ignore the error if OP1 == FF && OP2 == 1
#
#
[T: 1085] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1095] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1105] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 28 || ADD_SUB: 28 || OP1: 2c || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 2c +/- OP2 01 should NOT equal 28, you should ignore the error if OP1 == FF && OP2 == 1

#
#
#
#
Error: STORE after SUB S2,01 operand_a [OP1] is 2c the next result should be 2b , but is 28, FAIL
#
#
--
[T: 1115] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 28 || ADD_SUB: 4f || OP1: 28 || OP2: 28 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 28 +/- OP2 28 should NOT equal 4f, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--
[T: 1125] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: ff || ADD_SUB: fb || OP1: fd || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 fd +/- OP2 ff should NOT equal fb, you should ignore the error if OP1 == FF && OP2 == 1
#
#
[T: 1135] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1145] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1155] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 24 || ADD_SUB: 24 || OP1: 28 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 28 +/- OP2 01 should NOT equal 24, you should ignore the error if OP1 == FF && OP2 == 1
#
#
#
#
Error: STORE after SUB S2,01 operand_a [OP1] is 28 the next result should be 27 , but is 24, FAIL
#
#
--
[T: 1165] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 24 || ADD_SUB: 47 || OP1: 24 || OP2: 24 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 24 +/- OP2 24 should NOT equal 47, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--
[T: 1175] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: ff || ADD_SUB: fb || OP1: fd || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 fd +/- OP2 ff should NOT equal fb, you should ignore the error if OP1 == FF && OP2 == 1
#
#
[T: 1185] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

--
[T: 1195] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1205] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 20 || ADD_SUB: 20 || OP1: 24 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 24 +/- OP2 01 should NOT equal 20, you should ignore the error if OP1 == FF && OP2 == 1
#
#
#
#
Error: STORE after SUB S2,01 operand_a [OP1] is 24 the next result should be 23 , but is 20, FAIL
#
#
--
[T: 1215] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 20 || ADD_SUB: 3f || OP1: 20 || OP2: 20 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 20 +/- OP2 20 should NOT equal 3f, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--
[T: 1225] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: ff || ADD_SUB: fb || OP1: fd || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 fd +/- OP2 ff should NOT equal fb, you should ignore the error if OP1 == FF && OP2 == 1
#
#
[T: 1235] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1245] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1255] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 1c || ADD_SUB: 1c || OP1: 20 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 20 +/- OP2 01 should NOT equal 1c, you should ignore the error if OP1 == FF && OP2 == 1
#
#
#
#
Error: STORE after SUB S2,01 operand_a [OP1] is 20 the next result should be 1f , but is 1c, FAIL
#
#
--
[T: 1265] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 1c || ADD_SUB: 37 || OP1: 1c || OP2: 1c ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 1c +/- OP2 1c should NOT equal 37, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--

[T: 1275] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: ff || ADD_SUB: fb || OP1: fd || OP2: ff || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 fd +/- OP2 ff should NOT equal fb, you should ignore the error if OP1 == FF && OP2 == 1
#
#
[T: 1285] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1295] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1305] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 18 || ADD_SUB: 18 || OP1: 1c || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 1c +/- OP2 01 should NOT equal 18, you should ignore the error if OP1 == FF && OP2 == 1
#
#
#
#
Error: STORE after SUB S2,01 operand_a [OP1] is 1c the next result should be 1b , but is 18, FAIL
#
#
--
[T: 1315] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 18 || ADD_SUB: 2f || OP1: 18 || OP2: 18 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 18 +/- OP2 18 should NOT equal 2f, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--
[T: 1325] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: ff || ADD_SUB: fb || OP1: fd || OP2: ff || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 fd +/- OP2 ff should NOT equal fb, you should ignore the error if OP1 == FF && OP2 == 1
#
#
[T: 1335] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1345] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1355] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 14 || ADD_SUB: 14 || OP1: 18 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 18 +/- OP2 01 should NOT equal 14, you should ignore the error if OP1 == FF && OP2 == 1
#
#
#
#
Error: STORE after SUB S2,01 operand_a [OP1] is 18 the next result should be 17 , but is 14, FAIL
#
#

--
[T: 1365] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 14 || ADD_SUB: 27 || OP1: 14 || OP2: 14 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 14 +/- OP2 14 should NOT equal 27, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--
[T: 1375] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: ff || ADD_SUB: fb || OP1: fd || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 fd +/- OP2 ff should NOT equal fb, you should ignore the error if OP1 == FF && OP2 == 1
#
#
[T: 1385] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1395] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1405] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 10 || ADD_SUB: 10 || OP1: 14 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 14 +/- OP2 01 should NOT equal 10, you should ignore the error if OP1 == FF && OP2 == 1
#
#
#
#
Error: STORE after SUB S2,01 operand_a [OP1] is 14 the next result should be 13 , but is 10, FAIL
#
#
--
[T: 1415] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 10 || ADD_SUB: 1f || OP1: 10 || OP2: 10 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 10 +/- OP2 10 should NOT equal 1f, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--
[T: 1425] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: ff || ADD_SUB: fb || OP1: fd || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 fd +/- OP2 ff should NOT equal fb, you should ignore the error if OP1 == FF && OP2 == 1
#
#
[T: 1435] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1445] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1455] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: c || ADD_SUB: 0c || OP1: 10 || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#

#
Error: OP1 10 +/- OP2 01 should NOT equal 0c, you should ignore the error if OP1 == FF && OP2 == 1
#
#
#
#
Error: STORE after SUB S2,01 operand_a [OP1] is 10 the next result should be 0f , but is 0c, FAIL
#
#
--
[T: 1465] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: c || ADD_SUB: 17 || OP1: 0c || OP2: 0c ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 0c +/- OP2 0c should NOT equal 17, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--
[T: 1475] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: ff || ADD_SUB: fb || OP1: fd || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 fd +/- OP2 ff should NOT equal fb, you should ignore the error if OP1 == FF && OP2 == 1
#
#
[T: 1485] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1495] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
--
[T: 1505] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 8 || ADD_SUB: 08 || OP1: 0c || OP2: 01 ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 0c +/- OP2 01 should NOT equal 08, you should ignore the error if OP1 == FF && OP2 == 1
#
#
#
#
Error: STORE after SUB S2,01 operand_a [OP1] is 0c the next result should be 0b , but is 08, FAIL
#
#
--
[T: 1515] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 8 || ADD_SUB: 0f || OP1: 08 || OP2: 08 ||
SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 08 +/- OP2 08 should NOT equal 0f, you should ignore the error if OP1 == FF && OP2 == 1
#
#
--
[T: 1525] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: ff || ADD_SUB: fb || OP1: fd || OP2: ff ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 fd +/- OP2 ff should NOT equal fb, you should ignore the error if OP1 == FF && OP2 == 1
#
#

[T: 1535] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1545] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1555] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 4 || ADD_SUB: 04 || OP1: 08 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 08 +/- OP2 01 should NOT equal 04, you should ignore the error if OP1 == FF && OP2 == 1
#
#
#
#
Error: STORE after SUB S2,01 operand_a [OP1] is 08 the next result should be 07 , but is 04, FAIL
#
#

[T: 1565] || OP: JUMP NZ,03A || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 4 || ADD_SUB: 07 || OP1: 04 || OP2: 04 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 04 +/- OP2 04 should NOT equal 07, you should ignore the error if OP1 == FF && OP2 == 1
#
#

[T: 1575] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: ff || ADD_SUB: fb || OP1: fd || OP2: ff || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
Error: OP1 fd +/- OP2 ff should NOT equal fb, you should ignore the error if OP1 == FF && OP2 == 1
#
#
[T: 1585] || OP: LOAD s0,s0 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1595] || OP: SUB s2,01 || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1605] || OP: STORE s2,(s2) || ZERO: 0 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: 00 || OP1: 04 || OP2: 01 || SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 1 0
#
#
Error: OP1 04 +/- OP2 01 should NOT equal 00, you should ignore the error if OP1 == FF && OP2 == 1
#
#
#
#
Error: STORE after SUB S2,01 operand_a [OP1] is 04 the next result should be 03 , but is 00, FAIL
#
#

[T: 1615] || OP: JUMP NZ,03A || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: ff || OP1: 00 || OP2: 00 || SCR_WR_EN: 1 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0

[T: 1625] || OP: LOAD s0,s0 || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 1 || SHIFT_OP: RL_SRX || REG_DATA: ff || ADD_SUB: fb || OP1: fd || OP2: ff || SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#

```
#
#  Error: OP1 fd +/- OP2 ff should NOT equal fb, you should ignore the error if OP1 == FF && OP2 == 1
#
#
# ------------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 1635 ] || OP: RETURN  || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 1 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# ------------------------------------------------------------------------------------------------------------------------------------------------------
# [T: 1645 ] || OP: Invalid Instruction || ZERO: 1 || CARRY: 0 || SHIFT_BIT: 0 || SHIFT_OP: SA || REG_DATA: 0 || ADD_SUB: xx || OP1: xx || OP2: 00 ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
# [T: 1655 ] || OP: Invalid Instruction || ZERO: 1 || CARRY: 0 || SHIFT_BIT: x || SHIFT_OP:  || REG_DATA: x || ADD_SUB: xx || OP1: xx || OP2: xx ||
SCR_WR_EN: 0 || RG_WR_EN: 0 || WR_SRB: 0 || ZR_CR_WR_EN: 0 0
#
#
#  ----------------------------------------
#  FAILED To Write to Scratchpad
#
#
# ** Note: $stop    : /home/niko/Downloads/fnlproj-niko-kathy-aalap/team-demo/with_bug_pipelined_picoblaze_hdl/tb_testprogs.sv(349)
#    Time: 6050 ns  Iteration: 0  Instance: /alt_rojo_tb
# Break at /home/niko/Downloads/fnlproj-niko-kathy-aalap/team-demo/with_bug_pipelined_picoblaze_hdl/tb_testprogs.sv line 349
# Stopped at /home/niko/Downloads/fnlproj-niko-kathy-aalap/team-demo/with_bug_pipelined_picoblaze_hdl/tb_testprogs.sv line 349
```

References

[1] Sutherland, S. (2017). RTL Modeling with SystemVerilog for Simulation and Synthesis using SystemVerilog for ASIC and FPGA design. Sutherland HDL, Inc.

[2] Mehta, A. (2016). SystemVerilog Assertions and Functional Coverage Guide to Language, Methodology and Applications. Springer International Publishing AG Switzerland.

[3] Taraate, V. (2020). SystemVerilog for Hardware Description RTL Design and Verification. Springer.

[4] Chapman, K. (2003). Picoblaze KCPSM3 8-bit Micro Controller for Spartan-3, Virtex-II and Virtex-IIPRO. Xilinx Ltd.

[5] Herriman, A.,& Blanchard, E. (2006). PacoBlaze. Xilinx Ltd. Chapman, K. (2005). PicoBlaze 8-bit Embedded Microcontroller User Guide for Spartan-3, Virtex-II, and Virtex-II Pro FPGAs. Xilinx.

List of Figures