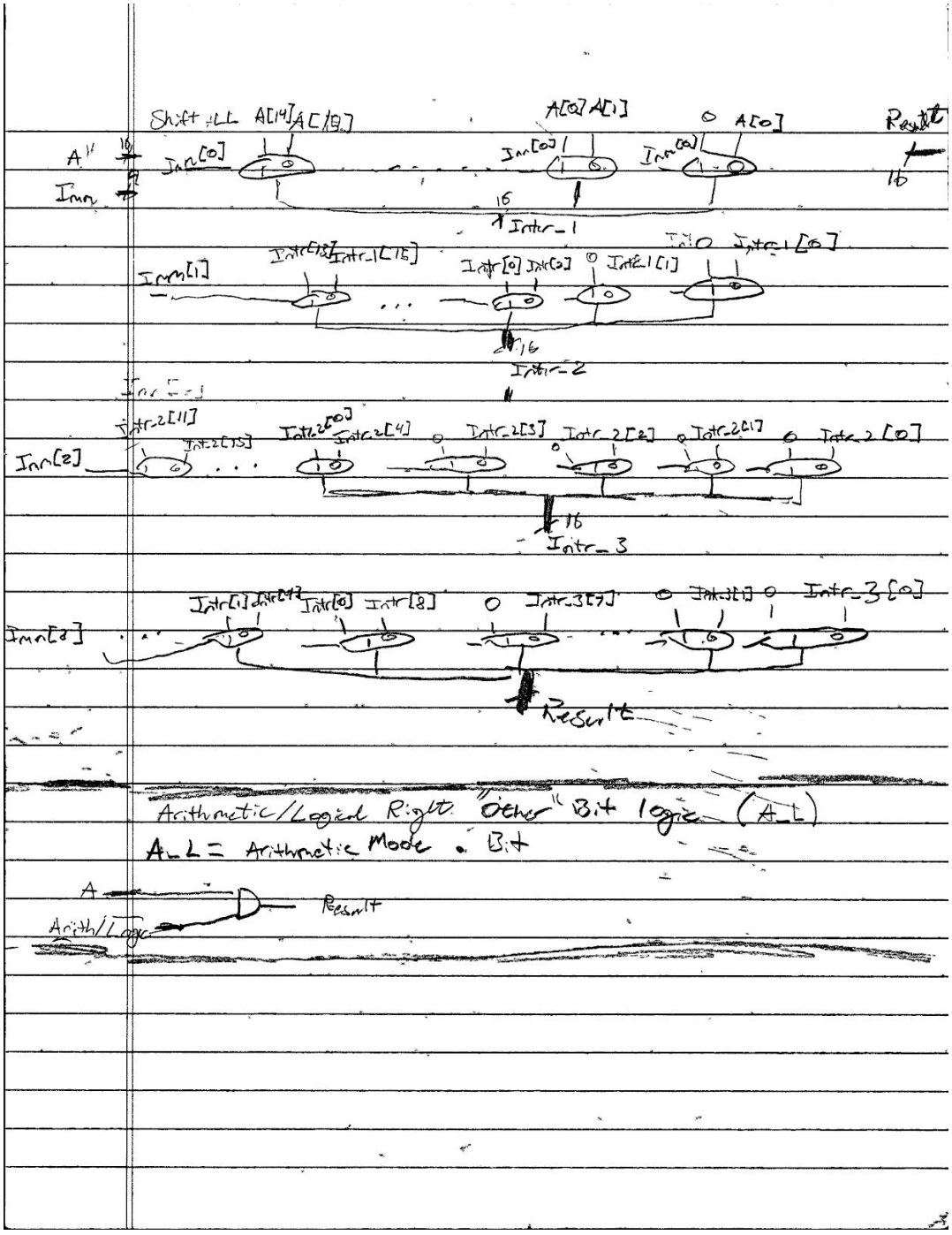
SEAN WILSON
10 - 23 - 15
CS 552

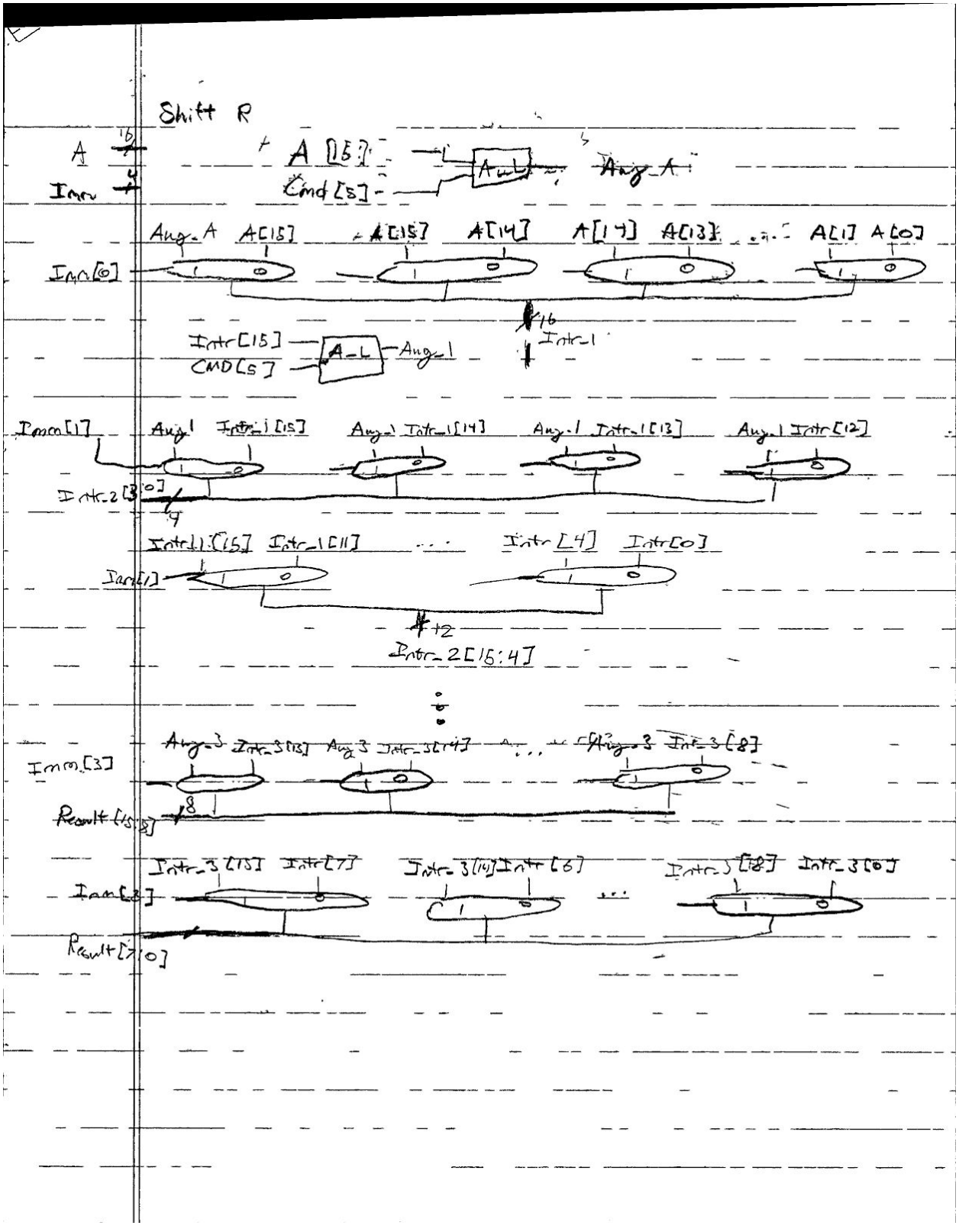
Project Progress Report

Folder Hierarchy:
Project

- **compiled_tests: Contains the compiled testbench for all test programs**
  - **test_progs_hex: Contains the hex format instruction mem progs**
- **Report: Contains this and screenshots from other tests.**
- **wisc_15_processor:**
  - makefile - type make to create a.out of cpu_tb
  - control.v - processes incoming instruction to produce control signals
  - cpu_tb.v - Used to build and test cpu.v
  - instr_logic.v - used to handle logic to create next cycle's pc
  - rf_pipelined.v - register file
  - instr_mem.v - instruction memory
  - data_mem.v -data memory
  - **alu: Contains all the makings of the alu**
    - alu.v - The top level of the alu, muxes out shift, add, sub, paddsb, lb
    - au.v - The arithmetic unit: Handles add, sub, paddsb
    - full_adder.v - A full adder used in au.v
    - shifter.v - Handles all versions of shifting
  - **tbs: Contains various testbenches used throughout production**

SCHEMATICS ORDERING:
Shift LL
Shift RL/RA
AU
Overflow Detection, and sum bit masking
ALU Top Level
CPU Top Level
Control Signals

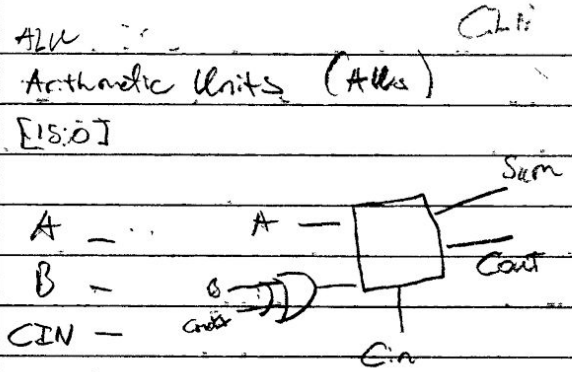

Shift LL
A[14] A[15]
A[0] A[1]
A[0]
Result
A
Inn
Inn[0]
16
Intr_1
Inn[1]
Intr_2
Inn[2]
Intr_3
Inn[3]
Result
Arithmetic/Logical Right "Other" Bit logic (A_L)
A_L = Arithmetic Mode . Bit
A
Result
Arith/Logic

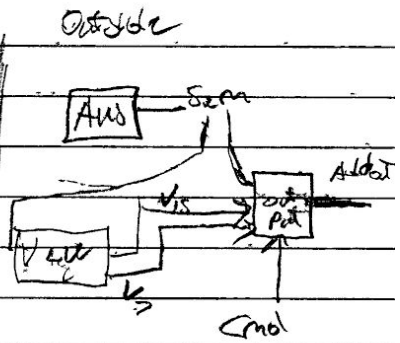

Shift R
A
Imm
A [15:]
Cmd [5]
ALU
Aug_A
Aug_A A[15] A[15] A[14] A[13] A[1] A[0]
Imm[0]
16
Intr_1
Intr[15]
CMD[5]
A-L
Aug_1
Imm[1]
Aug_1 Intr_1[15] Aug_1 Intr_1[14] Aug_1 Intr_1[13] Aug_1 Intr[12]
Intr_2[3:0]
4
Intr_1[15] Intr_1[11] Intr[4] Intr[0]
Imm[1]
12
Intr_2[15:4]
Aug_3 Intr_3[15] Aug_3 Intr_3[14] Aug_3 Intr_3[8]
Imm[3]
8
Result[15:8]
Intr_3[15] Intr[7] Intr_3[14] Intr[6] Intr_3[8] Intr_3[0]
Imm[2]
Result[7:0]

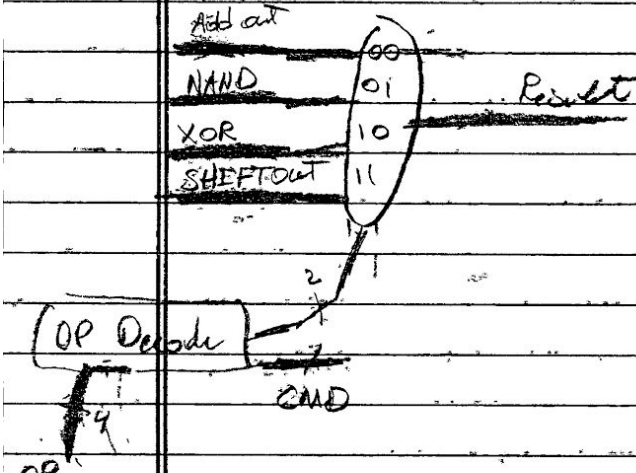
ALU

Arithmetic Units (ALUs)

[15:0]







CIn for ALU [8]

Cout [7] → CIN [8]

CMD [8]

Signals PADDSB

CIN for ALU [0]

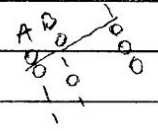
CI → CIN [0]

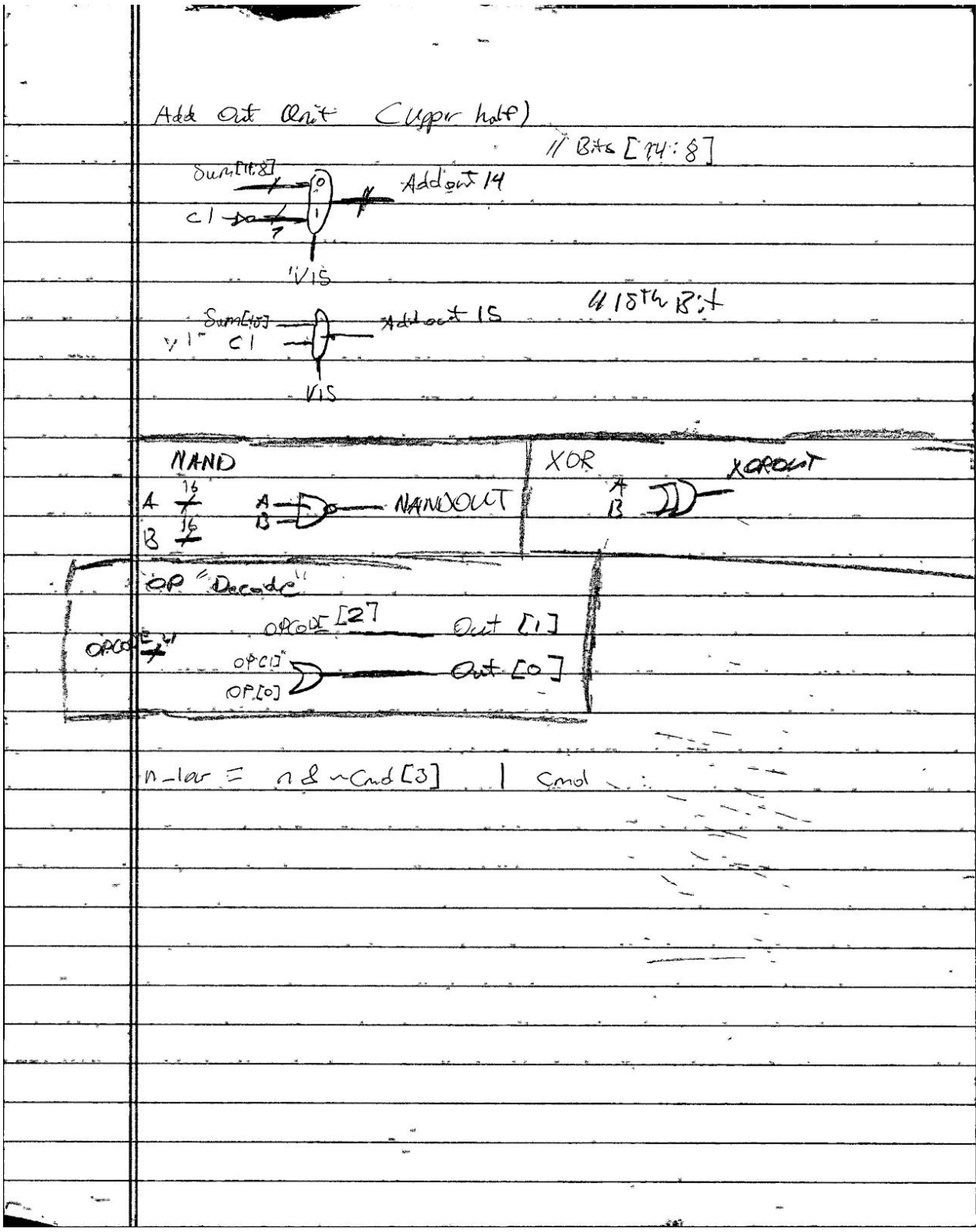
Logic:

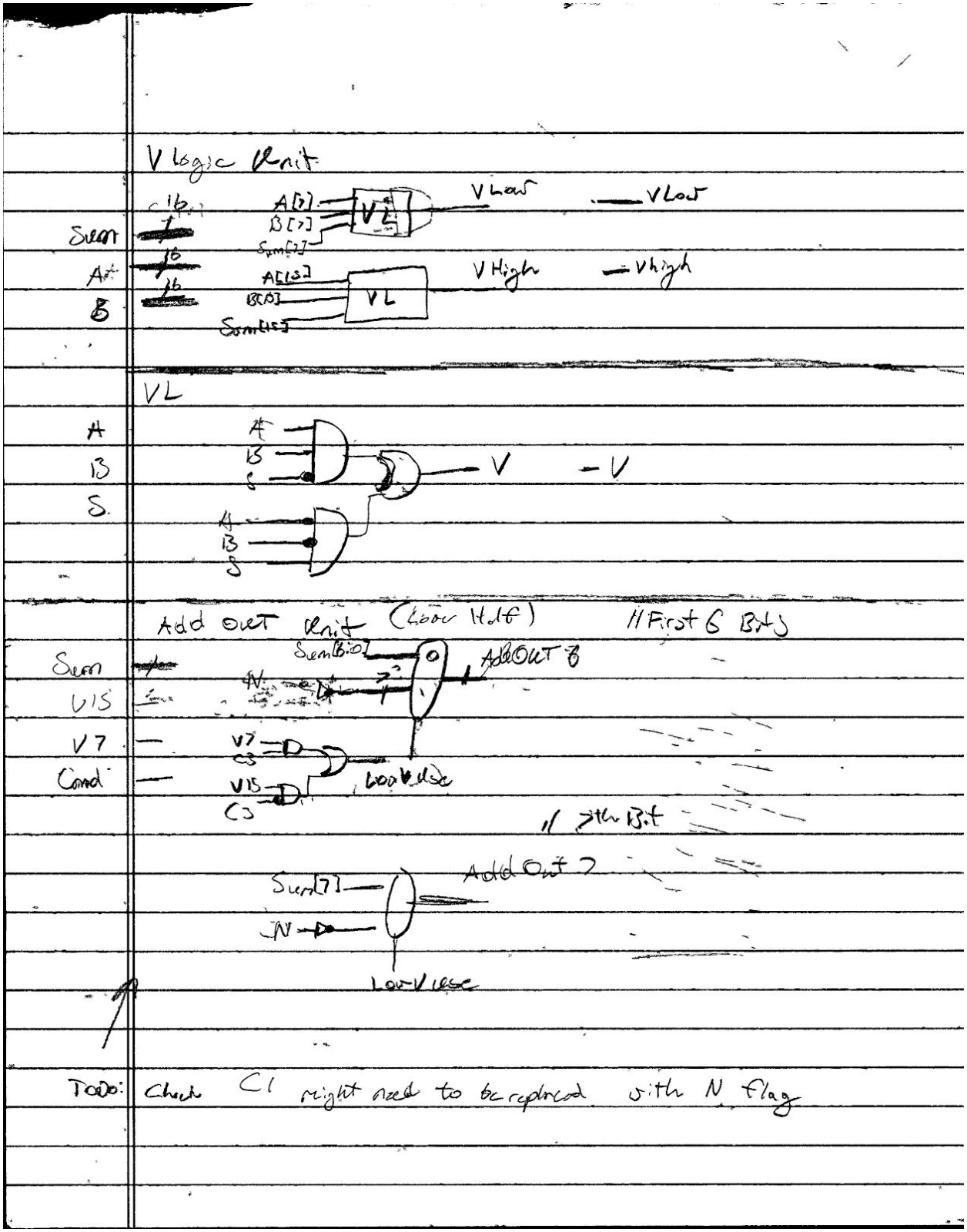
$V = \bar{a}_{n-1} b_{n-1} S_{n-1} + a_{n-1} b_{n-1} \bar{S}_{n-1}$

$N = \bar{V} \cdot S_{n-1} + V \cdot C_{n-1}$

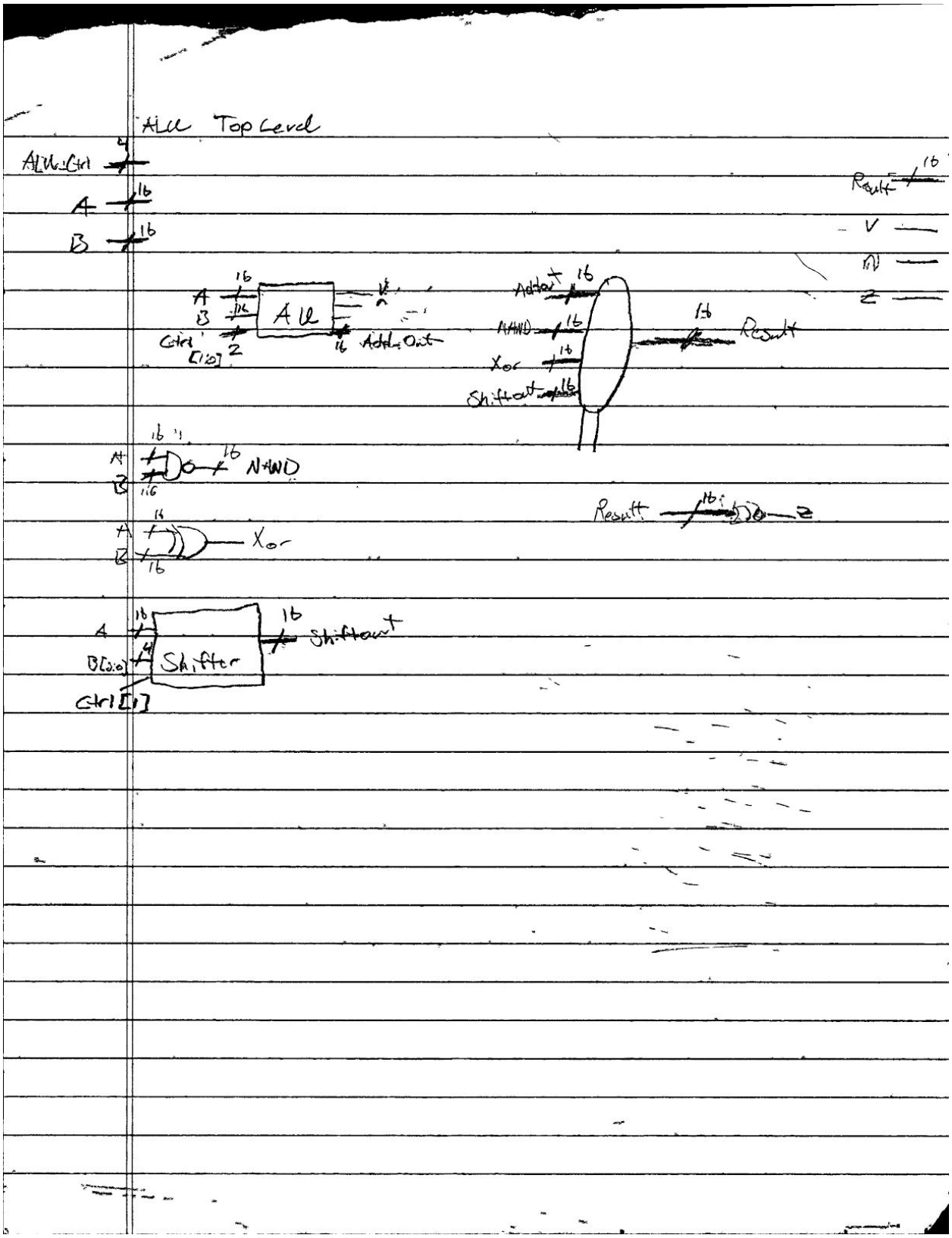
Z = ORC Bits [0:15]




Add Out Unit (Upper half)
11 Bits [14:8]
Sum[14:8]
Add out 14
c1
1
VIS
4 15th Bit
Sum[15]
Add out 15
"1" c1
VIS
NAND
A 16
B 16
A
B
NANDOUT
XOR
A
B
XOROUT
OP "Decode"
OPCode [2]
Out [1]
OPCode
OP[1]
OP[0]
Out [0]
n_lor = n & ~Cmd[3] | Cmd

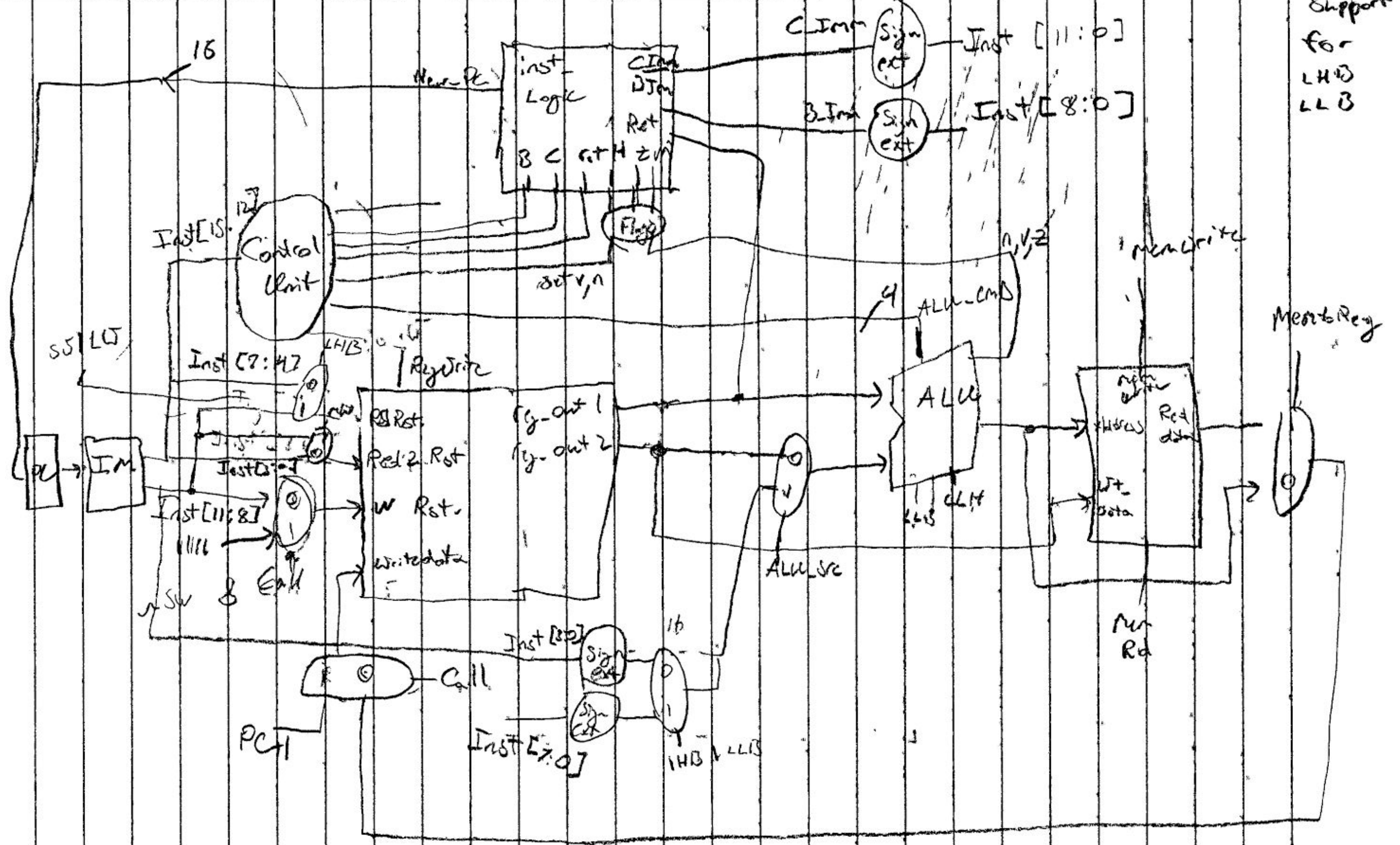


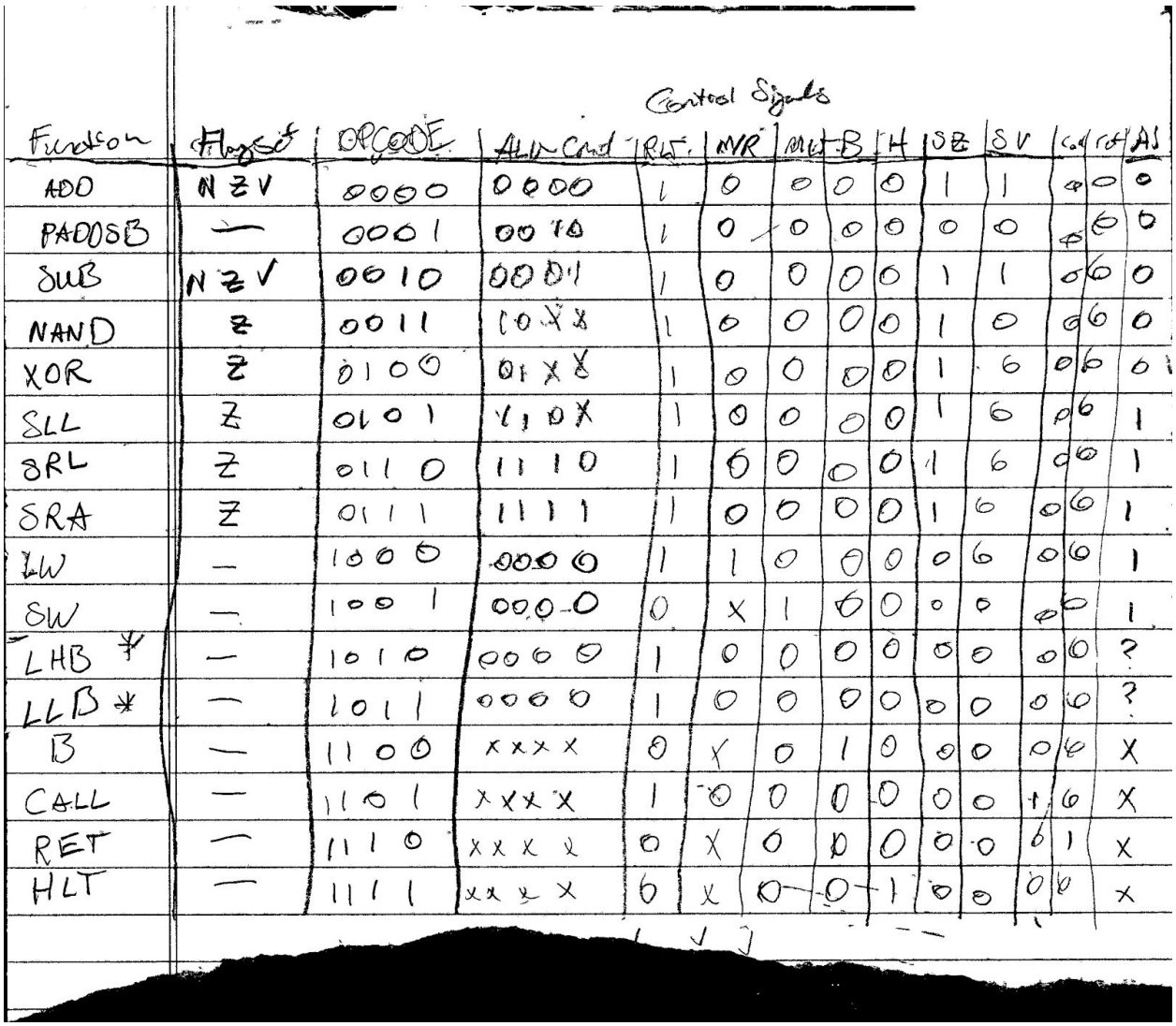
Todo: Check C1 might need to be replaced with N flag


ALU Top Level
ALU_Ctrl
4
A
16
B
16
Result
16
V
N
Z
ALU
Add_Out
Ctrl[1:0]
2
Adder
NAND
Xor
Shift_out
NAND
Xor
Shifter
B[3:0]
Ctrl[2]

Processor Datapath

TODO: Add support for LHB LLB





Control Signals

Function	Flags	OPCODE	ALU Cmd	RW	MR	MW	MB	IH	SB	SV	cond	rel	AJ
ADD	N Z V	0000	0000	1	0	0	0	0	1	1	0	0	0
PADDSB	—	0001	0010	1	0	0	0	0	0	0	0	0	0
SUB	N Z V	0010	0001	1	0	0	0	0	1	1	0	0	0
NAND	Z	0011	0XX0	1	0	0	0	0	1	0	0	0	0
XOR	Z	0100	01XX	1	0	0	0	0	1	0	0	0	0
SLL	Z	0101	1101	1	0	0	0	0	1	0	0	0	1
SRL	Z	0110	1110	1	0	0	0	0	1	0	0	0	1
SRA	Z	0111	1111	1	0	0	0	0	1	0	0	0	1
LW	—	1000	0000	1	1	0	0	0	0	0	0	0	1
SW	—	1001	0000	0	X	1	0	0	0	0	0	0	1
LHB *	—	1010	0000	1	0	0	0	0	0	0	0	0	?
LLB *	—	1011	0000	1	0	0	0	0	0	0	0	0	?
B	—	1100	XXXX	0	X	0	1	0	0	0	0	0	X
CALL	—	1101	XXXX	1	0	0	0	0	0	0	1	0	X
RET	—	1110	XXXX	0	X	0	0	0	0	0	0	1	X
HLT	—	1111	XXXX	0	X	0	0	1	0	0	0	0	X

SCREEN CAPTURE OF “BasicOp” Machine Code

Terminal

```
swilson@laptop ~/school/cs552/proj/compiled_tests/test_progs_hex $ ls
BasicOp.hex  Branch.hex  Control.hex  DataDependence.hex  instr.hex  Loop.hex  LwStall.hex
swilson@laptop ~/school/cs552/proj/compiled_tests/test_progs_hex $ cat BasicOp.hex
@0000 B100
@0001 A101
@0002 B2FF
@0003 A27F
@0004 2011
@0005 0312
@0006 3412
@0007 4512
@0008 1601
@0009 B180
@000A A101
@000B 1712
@000C 7844
@000D 6944
@000E 5A44
@000F 9A05
@0010 8B05
@0011 D003 //CALL FUNC
@0012 CE02 //B Uncond PASS
@0013 B1FF
@0014 A1FF
@0015 F000 //PASS:
@0016 B1AA //FUNC:
@0017 A1AA
@0018 E0F0
@0019 B1FF //FAIL:
@001A A1FF
@001B F000

//    LLB R1, 0x00
//    LHB R1, 0x01        # R1=0x0100
//    LLB R2, 0xFF
//    LHB R2, 0x7F        # R2=0x7FFF
//    SUB R0, R1, R1      # R0=0x0000
//    ADD R3, R1, R2      # R3=0x7FFF
//
//    NAND R4, R1, R2     # R4=0xFEFF
//    XOR  R5, R1, R2     # R5=0x7EFF
//
//    PADDSB R6, R0, R1   # R6=0x0100
//    LLB R1, 0x80        # R1=0xFF80
//    LHB R1, 0x01        # R1=0x0180
//    PADDSB R7, R1, R2   # R7=0x7F80
//
//    SRA R8, R4, 0x4     # R8=0xFFEF
//    SRL R9, R4, 0x4     # R9=0x0FEF
//    SLL R10, R4, 0x4    # R10=0xEFF0
//
//    SW  R10, R0, 0x5    # mem[5]=0xEFF0
//    LW  R11, R0, 0x5    # R11=0xEFF0
//
//    CALL FUNC
//
//    B UNCOND, PASS
```

SCREEN CAPTURE OF THE STATE OF THE SIMULATION AFTER COMPLETION OF “BasicOp”