**Syzygy B100**

**Registers**

* R0: Comparisons and ALU output
* R1: Jump Address
* R2: External peripheral output
* R3: External peripheral input
* R4: ALU status register
* R5: Dynamic instruction register (used for eval)
* R6 - R15 are general-purpose

**Instruction Set (operations)**

* push - Sets R0 to a given value.
* load - Loads a value from a register or memory address into the ALU's A, B, or both.
* loadn - Loads a negated value from a register or memory address into the ALU's A, B, or both.
* copy - Copies a value from a register or memory address to another register or memory address.
* pass - Sets R0 to A.
* neg - Sets R0 to !A.
* inc - Sets R0 to A + 1.
* dec - Sets R0 to A - 1.
* add - Sets R0 to A + B.
* addn - Sets R0 to !(A + B).
* iadd - Sets R0 to A + B + 1.
* addd - Sets R0 to A + B - 1.
* iaddn - Sets R0 to !(A + B + 1).
* adddn - Sets R0 to !(A + B - 1).
* and - Sets R0 to A & B.
* andn - Sets R0 to !(A & B).
* iand - Sets R0 to ((A + 1) & B).
* andd - Sets R0 to (A & B) - 1.
* iandn - Sets R0 to !((A + 1) & B).
* anddn - Sets R0 to !((A & B) - 1).
* xor - Sets R0 to A XOR B.
* xorn - Sets R0 to !(A XOR B)
* ixor - Sets R0 to (A+1) XOR B.
* xord - Sets R0 to (A XOR B) - 1.
* ixorn - Sets R0 to !((A+1) XOR B).
* xordn - Sets R0 to !(A XOR B) - 1.
* ls - Sets R0 to A << B.
* lsn - Sets R0 to !(A << B).
* ils - Sets R0 to (A + 1) << B.
* lsd - Sets R0 to (A << B) - 1.
* ilsn - Sets R0 to !((A + 1) << B).
* lsdn - Sets R0 to !((A << B) - 1).
* lr - Sets R0 to A <~ B.
* lrn - Sets R0 to !(A <~ B).
* ilr - Sets R0 to (A + 1) <~ B).
* lrd - Sets R0 to (A <~ B) - 1
* ilrn - Sets R0 to !((A + 1) <~ B)
* lrdn - Sets R0 to !((A <~ B) - 1).
* las - Sets R0 to A <<< B.
* lasn - Sets R0 to !(A <<< B).
* ilas - Sets R0 to (A + 1) <<< B.
* lasd - Sets R0 to (A <<< B) - 1.
* ilasn - Sets R0 to !((A + 1) <<< B).
* lasdn - Sets R0 to !((A <<< B) - 1).
* lar - Sets R0 to A <<~ B.
* larn - Sets R0 to !(A <<~ B).
* ilar - Sets R0 to (A + 1) <<~ B.
* lard - Sets R0 to (A <<~ B) - 1.
* ilarn - Sets R0 to !(A + 1) <<~ B)).
* lardn - Sets R0 to !(A <<~ B) - 1).
* rs - Sets R0 to A >> B.
* rsn - Sets R0 to !(A >> B).
* irs - Sets R0 to (A + 1) >> B.
* rsd - Sets R0 to (A >> B) - 1.
* irsn - Sets R0 to !((A + 1) >> B).
* rsdn - Sets R0 to !((A >> B) - 1).
* rr - Sets R0 to A ~> B.
* rrn - Sets R0 to !(A ~> B).
* irr - Sets R0 to (A + 1) ~> B).
* rrd - Sets R0 to (A ~> B) - 1
* irrn - Sets R0 to !((A + 1) ~> B)
* rrdn - Sets R0 to !((A ~> B) - 1).
* ras - Sets R0 to A >>> B.
* rasn - Sets R0 to !(A >>> B).
* iras - Sets R0 to (A + 1) >>> B.
* rasd - Sets R0 to (A >>> B) - 1.
* irasn - Sets R0 to !((A + 1) >>> B).
* rasdn - Sets R0 to !((A >>> B) - 1).
* rar - Sets R0 to A ~>> B.
* rarn - Sets R0 to !(A ~>> B).
* irar - Sets R0 to (A + 1) ~>> B.
* rard - Sets R0 to (A ~>> B) - 1.
* irarn - Sets R0 to !(A + 1) ~>> B)).
* rardn - Sets R0 to !(A ~>> B) - 1).
* jmp - Sets the current instruction to R1's value.
* jeq - Sets the current instruction to R1's value if R0 = 0.
* jne - Sets the current instruction to R1's value if R0 != 0.
* jlt - Sets the current instruction to R1's value if R0 < 0.
* jgt - Sets the current instruction to R1's value if R0 > 0.
* jle - Sets the current instruction to R1's value if R0 <= 0.
* jge - Sets the current instruction to R1's value if R0 >= 0.
* sdrd - Reads two bytes from the SD card using R2 as the offset.
* sdwd - Writes two bytes to the SD card using R2 as the offset.
* mrd - Reads two bytes from memory using R2 as the offset.
* mwd - Writes two bytes to memory using R2 as the offset.

**Instruction Set (machine code)**

1nnn nnnn nnnn nnnn

push <n>

Sets R0 to $n.

0000 aa\_\_ \_\_\_\_ \_\_\_\_

sys op

a: Operations

0: No-op

1: Eval R5's value as an instruction and execute it

2: No-op

3: Halt

0001 aaaa bbbb cd\_\_

load a, b

a: Register number that has the value or memory address to write to the ALU's A-side.

b: Register number that has the value or memory address to write to the ALU's B-side.

c: Use register $a as a memory pointer when loading the value in the corresponding memory address into the ALU.

d: Perform a bitwise negation on the value loaded.

0010 aaaa bbbb cd\_\_

copy a, b

a: Register or memory address to read from.

b: Register or memory address to write to.

c: Use the read register as a memory pointer to read directly from memory.

d: Use the write register as a memory pointer to write directly into memory.

0011 aaab cdef g\_\_\_

alu op

a: Operation to perform:

0: Pass A

1: A + B

2: A & B

3: A XOR B

4: A << B

5: A >> B

6: \_\_\_

7: \_\_\_

b: Increment A before the operation is performed.

c: \_\_\_\_

d: Negate the output.

e: Decrement the output.

f: If shifting, rotate bits.

g: If shifting, perform an arithmetic shift.

0100 abc\_ \_\_\_\_ \_\_\_\_

jump

a: Jump to R1's value if R0's value is < 0.

b: Jump to R1's value if R0's value is = 0.

c: Jump to R1's value if R0's value is > 0.

0101 aaaa bc\_\_ \_\_\_\_

a: Peripheral ID to select.

b: Data direction:

0: Read

1: Write

c: Peripheral register select:

0: Selects the state register if reading, or the op register if writing.

1: Selects the data register.

0110 \_\_\_\_ \_\_\_\_ \_\_\_\_

No-op

0111 \_\_\_\_ \_\_\_\_ \_\_\_\_

No-op