**Syzygy B100**

**Registers**

* R0: Instruction Register (IR)
  + Contains the current instruction in machine code.
* R1: Program Counter (PC)
  + Contains the current instruction number in boot mode (Harvard), or the memory address where to fetch the current instruction from in standard mode (Von Neumann).
* R2: Accumulator (AC)
  + Contains the value from a push instruction or ALU operation (last value is overwritten), and where comparisons take place.
  + If the value in this register meets the desired condition when a jump instruction is executed, R1 will be overwritten with R3's value.
* R3: Jump Address (JA)
  + Contains the address that will be written to R1 if the value in R2 meets the desired condition when a jump instruction is executed.
* R4: Peripheral I/O, bits 0-15 (P0)
  + Contains a 16-bit value, or the lower 16 bits of a 32-bit value to be read from or be written to a register of any peripheral interface.
* R5: Peripheral I/O, bits 16-31 (P1)
  + Contains the upper 16 bits of a 32-bit value to be used in combination with R4's value if the given peripheral supports 32-bit values.
* R6: ALU Input A (A)
  + Contains the operand of a unary ALU operation, or the first operand for a binary ALU operation.
* R7: ALU Input B (B)
  + Contains the second operand of a binary ALU operation. This value is ignored for unary operations.
* R8 - R15
  + General purpose registers.

**Instruction Set (machine code)**

**Key**

* Line 1: The 16-bit binary instruction:
  + A variable (a, b, c, etc.) in place of a bit or range of bits denotes a separate option or argument and is explained on or after line 3. Variables spanning multiple bits are multi-bit options, and each permutation's function is listed.
  + Bits with periods (.) are ignored by the current operation.
* Line 2: A description of what the operation does.
* Line 3+: Elaboration of the operation, including all option and argument combinations.
  + Variables referenced from line 2 are preceded with a dollar sign ($).

**Operations**

1nnn nnnn nnnn nnnn

Writes a given value to R2.

n: The value to write to R2.

0000 ..aa bbbb c...

System-specific instructions.

a: Operations

0: No-op

1: Set or clear a system flag

2: No-op

3: Halt

b: Argument 1:

if $op = 1: the flag ID to edit

c: Mode:

if $op = 1: 0 = clear, 1 = set

0001 aaaa bbbb c...

Copies the value of one register to another.

a: Register or memory address to read from.

b: Register or memory address to write to.

c: Perform bitwise negation on the value being copied before writing to the destination register.

0010 abc. .... ....

Sets the program counter to a given instruction if a given condition is met.

a: Set R1's value to the value in R3 if R2's value is less than 0.

b: Set R1's value to the value in R3 if R2's value is equal to 0.

c: Set R1's value to the value in R3 if R2's value is greater than 0.

0011 .ppp abzn irl.

ALU operations

p: Operation to perform:

0: Pass A (do not perform any operation)

1: A OR B (bitwise or)

2: A + B (addition)

3: Left shift A by B

4: A XOR B (exclusive-or)

5: Count the number of high bits in A

6: UNUSED

7: UNUSED

a: Negate input A (value of R6) before performing any operation.

b: Negate input B (value of R7) before performing any operation.

z: Use the value 0 for input B (R7 is not overwritten).

n: Negate the output of the operation.

i: If adding (p = 2), then add one to the result.

If shifting (p = 3), perform a right shift instead of a left shift.

r: If shifting (p = 3), perform a rotation instead of a shift.

l: If shifting (p = 3), perform a logical shift instead of an arithmetic shift (do not preserve the MSB (sign bit).

0100 pppp rrrr xmb.

Peripheral I/O Instructions

p: Peripheral ID to select (supports up to 16, but only 8 are currently available due to FPGA limitations).

r: Peripheral register to read/write from/to.

x: Have peripheral ID = $p execute the instruction in its R0 register.

m: Acccess mode:

0: Read the value from peripheral ID = $p's register $r and copy it to R4. Additionally, if 32-bit mode is enabled ($b = 1), the upper bits (16-31) will be copied to R5.

1: Write the value from R4 to peripheral ID = $p's register $r to R4. If 32-bit mode is enabled ($b = 1), also writes R5's value to the upper bits (16-31) of the peripheral's register.

b: Enables 32-bit read/write mode. R4 and R5 will be used in combination to form a 32-bit value, with R4's value acting as the lower bits, and R5's value acting as the upper bits.

0101 .... .... ....

UNUSED

0110 .... .... ....

UNUSED

0111 .... .... ....

UNUSED