Syzygy A100CPU

# >Description<

The Syzygy A100 is a CPU featuring variable clock speed (in Minecraft only), and a relatively advanced ALU, capable of pre and post-calculation operations. The ALU is capable of all basic arithmetic and logical operations, including a single-step square root extractor, a hardware integrated random number generator, a bit counter, and an identity/magnitude comparator for all conditional expressions.Taking all combinations into consideration, the ALU is capable of over 400 different operations. The Syzygy A100 is also capable of conditional branching.

# > Specifications <

Integer limits: -32768 to 32767 (16-bit)

Instruction word size: 16-bit

Program memory addresses: 64 (6-bit)

CPU memory addresses: 16 (4-bit)

Clock speed: Variable

Inputs: ???

Outputs: 4x 5-digit 7-segment displays

Dimensions: ???

# >Pseudocode Abbreviations <

$0,$1…$f CPU memory addresses (lowercase hex)

$A,$B ALU input memory (uppercase A or B)

iA,iB… Input panel index (uppercase A to D)

$X ALU accumulator (uppercase X)

$N Condition flag (uppercase N)

@23 Program memory index (number 0 to 63)

#2 Display panel index (number 0 to 3)

# >Instruction Set<

SYS - System controls

0000 xxaa bbbb xxxx

a: 2-bit operation flag

00: Do nothing

01: UNUSED

10: Wait untilbutton press.

11: Stop system clock

STOR - Writes from input to memory address.

0001 xxaa bbbb xxxx

a: 2-bit input panel selection.

b: 4-bit memory address to write to.

COPY – Copies a value from memory to a different address.

0010 aaaa bbbb xxcd

a: 4-bit address to read. Ignored if #c is 1.

b: 4-bit address to write. Ignored if #d is 1.

c: 1-bit ALUaccumulatorread flag. Ignored if #d=1 or #a!=0.

d: 1-bit ALU accumulatorwrite flag. Ignored if #c=1 or #b!=0.

CLR - Clears a memory address.

0011 aaaa xxbbxxxc

a: 4-bit address to clear. Ignored if #b is not 0.

b: 2-bit component memory address:

00: Nothing

01: ALU accumulator

10: ALU Register

11: Conditional flag

c: 1-bit register select:

0: A

1: B

ALU - ALU operations. Result is wrote to ALU memory.

0100 aaaabbbcddee

a: 4-bit operation to perform.

0000: PASS A

0001: SUM A + B

0010: OR A | B

0011: AND A & B

0100: XOR A ^ B

0101: SHFT (shift)

0110: SQUR (square)

0111: SQRT (square root)

1000: BCNT (bit counter)

1001: RAND (random number)

1010: IFEQ A = B

1011: INEQ A != B

1100: IFLT A < B

1101: IFGT A > B

1110: LTEQ A <= B

1111: GTEQ A >= B

b: Unlessthe operation is a shift, b is the 3-bit pre-operation control

0000: Nothing

0001: Negate A

0010: Negate B

0011: Increment A

0100: Increment B

0101: Decrement A

0110: Decrement B

0111: 2’s compliment B

If the operation is a shift, b is the 4-bit shift amount

c: 1-bit shift direction (ignored if #a != 1001)

0: Left

1: Right

d: 2-bit shift type

00: Logical (No rotation)

01: Arithmetic

10: Logical (With rotation)

e: 2-bit post-operation control

00: Nothing

01: Negate

10: Increment

11: Decrement

JUMP – Set program counter to specified index if condition flag is 1.

0101 xxaa aaaa xxxx

a: 6-bit instruction location to jump to.

LOAD– Sets the specified ALU register to a value in memory.

0110 aaaa xxxb xxxx

a: 4-bit memory address to read.

b: 1-bit register select.

0: A

1: B

DISP – Displays the value in memory.

0111 aaaa bcdd xxxx

a: 4-bit memory address to read.

b: 1-bit flag to clear the screen.

c: 1-bit flag to ignore the sign bit

(1100 0000 would display as 192 instead of -64)

d: 2-bit display index to use.

EXP0 - Custom Expansion 1. (The expansion will need its own decoder for the

1000 ???? ???? ???? arguments.)

EXP1 - Custom Expansion 2.

1001 ???? ???? ????

...

EXPF - Custom Expansion 16.

1111 ???? ???? ????

# > Example Programs <

### Addition

Index Raw Binary Hex SZG-Code

00 0001 0000 0000 0000 10 00 STOR i0,$0

01 0001 0001 0001 0000 11 10 STOR i1,$1

02 0110 0000 0000 0000 60 00 LOAD $0,$A

03 0110 0001 0001 0000 61 10 LOAD $1,$B

04 0100 0010 0000 0000 42 00 SUM

05 0010 0000 0010 0010 20 22 COPY $X,$2

06 0111 0010 0000 0000 72 00 DISP $2,0

07 0000 0011 0000 0000 03 00 SYS TERM

### Multiplication

Similar to division

### Division w/ Remainder

Index Raw Binary Hex SZG-Code

00 0001 0000 0000 0000 10 00 STOR i0,$0

01 0001 0001 0001 0000 11 10 STOR i1,$1

02 0110 0000 0000 0000 60 00 LOAD $0,$A

03 0110 0001 0001 0000 61 10 LOAD $1,$B

04 0100 0010 0111 0000 42 70 COMP $B !SUM

05 0011 0000 0010 0000 30 20 CLR $A

06 0011 0000 0010 0001 30 21 CLR $B

07 0011 0000 0000 0000 30 00 CLR $0

08 0010 0000 0000 0010 20 02 COPY $X,$0

09 0011 0000 0001 0000 30 10 CLR $X

0A 0110 0010 0000 0000 62 00 LOAD $0,$A

0B 0100 0001 0000 1000 41 08 PASS $A INC

0C 0011 0000 0010 0000 30 20 CLR $A

0D 0011 0010 0000 0000 32 00 CLR $2

0E 0010 0000 0010 0010 20 22 COPY $X,$2

0F 0011 0000 0001 0000 30 10 CLR $X

10 0110 0000 0000 0000 60 00 LOAD $0,$A

11 0110 0001 0001 0000 61 10 LOAD $1,$B

12 0011 0000 0011 0000 30 30 CLR $N

13 0100 1111 0000 0000 4F 00 GTEQ

14 0101 0000 0100 0000 50 40 JUMP @04

15 0111 0010 0000 0000 72 00 DISP $2,0

16 0111 0000 0001 0000 70 10 DISP $0,1

17 0000 0011 0000 0000 03 00 SYS TERM

### Input + Random Number

Generate random number

Add to user input

### Fibonacci Sequence (0, 1, 1, 2, 3, 5, 8, 13, 21, 34 …)

Initialize 0 and 1 to $0 and $1

Load to A and B

Sum, copy $X to

Add 0 + 1, Return 1.

Add 1 + 1, Return 2.

Add 1 + 2, Return 3.

Add 2 + 3, Return 5.

Etc