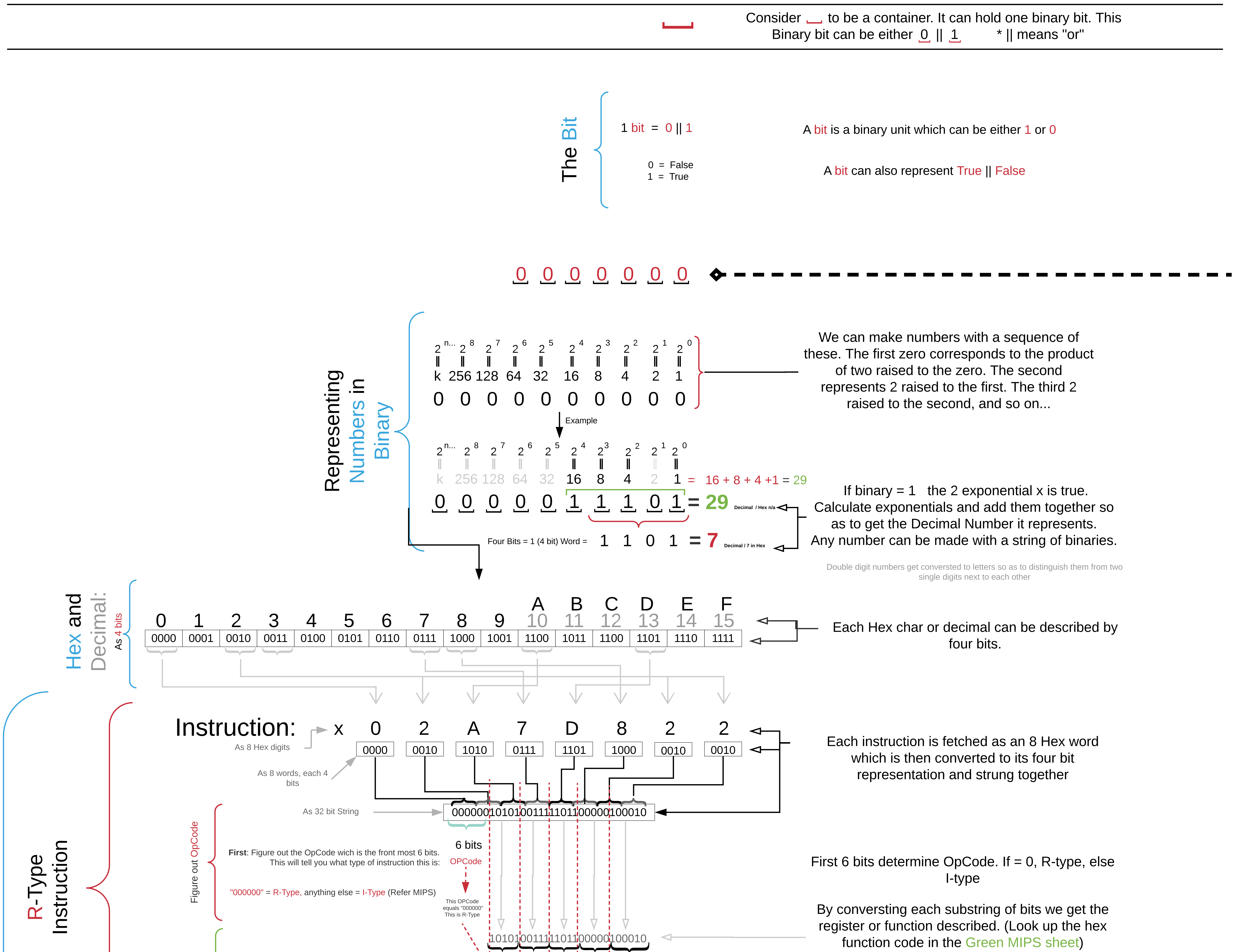


Consider ⎵ to be a container. It can hold one binary bit. This Binary bit can be either 0 || 1 * || means "or"

The Bit

1 bit = 0 || 1

0 = False
1 = True

A bit is a binary unit which can be either 1 or 0

A bit can also represent True || False

0 0 0 0 0 0 0

Representing Numbers in Binary

$2^{n...}$ 2^8 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
k 256 128 64 32 16 8 4 2 1
0 0 0 0 0 0 0 0 0 0
Example
k 256 128 64 32 16 8 4 2 1 = 16 + 8 + 4 +1 = 29
0 0 0 0 0 1 1 1 0 1 = 29
Decimal / Hex n/a
Four Bits = 1 (4 bit) Word = 1 1 0 1 = 7
Decimal / 7 in Hex

We can make numbers with a sequence of these. The first zero corresponds to the product of two raised to the zero. The second represents 2 raised to the first. The third 2 raised to the second, and so on...

If binary = 1 the 2 exponential x is true. Calculate exponentials and add them together so as to get the Decimal Number it represents. Any number can be made with a string of binaries.

Double digit numbers get conversted to letters so as to distinguish them from two single digits next to each other

Hex and Decimal:

As 4 bits

Each Hex char or decimal can be described by four bits.

R-Type Instruction

Instruction: x 0 2 A 7 D 8 2 2

As 8 Hex digits

Each instruction is fetched as an 8 Hex word which is then converted to its four bit representation and strung together

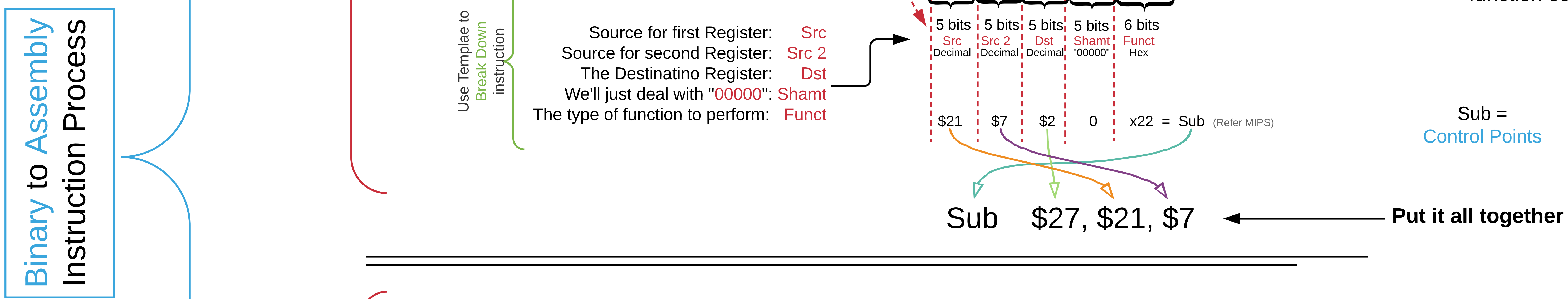
Figure out OpCode

First: Figure out the OpCode wich is the front most 6 bits. This will tell you what type of instruction this is:

"000000" = R-Type, anything else = I-Type (Refer MIPS)

First 6 bits determine OpCode. If = 0, R-type, else I-type

By conversting each substring of bits we get the register or function described. (Look up the hex function code in the Green MIPS sheet)



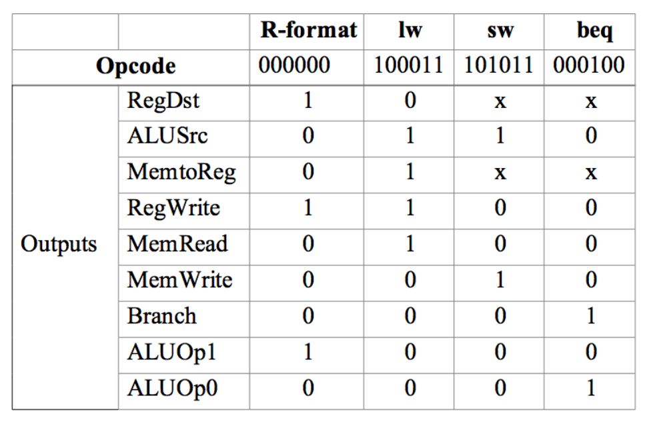
Use Templae to Break Down instruction

Source for first Register: Src
Source for second Register: Src 2
The Destinatino Register: Dst
We'll just deal with "00000": Shamt
The type of function to perform: Funct

Sub $27, $21, $7

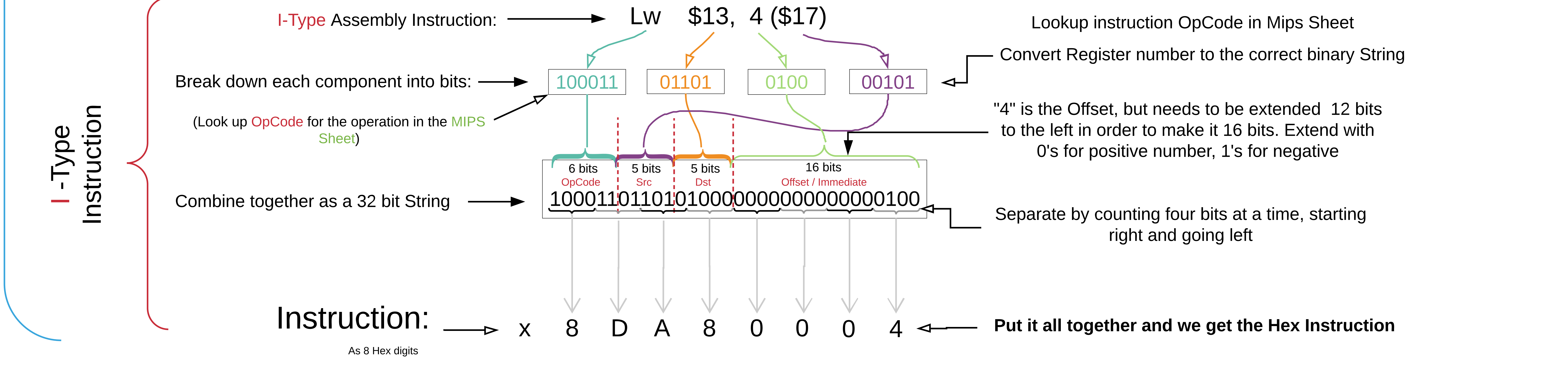
Binary to Assembly Instruction Process

Sub = Control Points



		R-format	lw	sw	beq
Opcode		000000	100011	101011	000100
Outputs	RegDst	1	0	x	x
	ALUSrc	0	1	1	0
	MemtoReg	0	1	x	x
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

Put it all together and we get the Assembly Instruction



I -Type Instruction

I-Type Assembly Instruction: Lw $13, 4 ($17)

Break down each component into bits:

(Look up OpCode for the operation in the MIPS Sheet)

Combine together as a 32 bit String

Lookup instruction OpCode in Mips Sheet

Convert Register number to the correct binary String

"4" is the Offset, but needs to be extended 12 bits to the left in order to make it 16 bits. Extend with 0's for positive number, 1's for negative

Separate by counting four bits at a time, starting right and going left

Instruction: x 8 D A 8 0 0 0 4

As 8 Hex digits

Put it all together and we get the Hex Instruction