

# CS 61C Reference Card

Version 4.1.0

	Instruction	Name	Description	Type	Opcode	Funct3	Funct7
Arithmetic	<b>add</b> <b>rd rs1 rs2</b>	ADD	<b>rd = rs1 + rs2</b>	R	011 0011	000	000 0000
	<b>sub</b> <b>rd rs1 rs2</b>	SUBtract	<b>rd = rs1 - rs2</b>	R	011 0011	000	010 0000
	<b>and</b> <b>rd rs1 rs2</b>	bitwise AND	<b>rd = rs1 &amp; rs2</b>	R	011 0011	111	000 0000
	<b>or</b> <b>rd rs1 rs2</b>	bitwise OR	<b>rd = rs1   rs2</b>	R	011 0011	110	000 0000
	<b>xor</b> <b>rd rs1 rs2</b>	bitwise XOR	<b>rd = rs1 ^ rs2</b>	R	011 0011	100	000 0000
	<b>sll</b> <b>rd rs1 rs2</b>	Shift Left Logical	<b>rd = rs1 &lt;&lt; rs2</b>	R	011 0011	001	000 0000
	<b>srl</b> <b>rd rs1 rs2</b>	Shift Right Logical	<b>rd = rs1 &gt;&gt; rs2</b> (Zero-extend)	R	011 0011	101	000 0000
	<b>sra</b> <b>rd rs1 rs2</b>	Shift Right Arithmetic	<b>rd = rs1 &gt;&gt; rs2</b> (Sign-extend)	R	011 0011	101	010 0000
	<b>slt</b> <b>rd rs1 rs2</b>	Set Less Than (signed)	<b>rd = (rs1 &lt; rs2) ? 1 : 0</b>	R	011 0011	010	000 0000
	<b>sltu</b> <b>rd rs1 rs2</b>	Set Less Than (Unsigned)		R	011 0011	011	000 0000
	<b>addi</b> <b>rd rs1 imm</b>	ADD Immediate	<b>rd = rs1 + imm</b>	I	001 0011	000	
	<b>andi</b> <b>rd rs1 imm</b>	bitwise AND Immediate	<b>rd = rs1 &amp; imm</b>	I	001 0011	111	
	<b>ori</b> <b>rd rs1 imm</b>	bitwise OR Immediate	<b>rd = rs1   imm</b>	I	001 0011	110	
	<b>xori</b> <b>rd rs1 imm</b>	bitwise XOR Immediate	<b>rd = rs1 ^ imm</b>	I	001 0011	100	
	<b>slli</b> <b>rd rs1 imm</b>	Shift Left Logical Immediate	<b>rd = rs1 &lt;&lt; imm</b>	I*	001 0011	001	000 0000
	<b>srl</b> <b>rd rs1 imm</b>	Shift Right Logical Immediate	<b>rd = rs1 &gt;&gt; imm</b> (Zero-extend)	I*	001 0011	101	000 0000
	<b>srai</b> <b>rd rs1 imm</b>	Shift Right Arithmetic Immediate	<b>rd = rs1 &gt;&gt; imm</b> (Sign-extend)	I*	001 0011	101	010 0000
Memory	<b>lb</b> <b>rd imm(rs1)</b>	Load Byte	<b>rd = 1 byte of memory at address rs1 + imm, sign-extended</b>	I	000 0011	000	
	<b>lbu</b> <b>rd imm(rs1)</b>	Load Byte (Unsigned)	<b>rd = 1 byte of memory at address rs1 + imm, zero-extended</b>	I	000 0011	100	
	<b>lh</b> <b>rd imm(rs1)</b>	Load Half-word	<b>rd = 2 bytes of memory starting at address rs1 + imm, sign-extended</b>	I	000 0011	001	
	<b>lhu</b> <b>rd imm(rs1)</b>	Load Half-word (Unsigned)	<b>rd = 2 bytes of memory starting at address rs1 + imm, zero-extended</b>	I	000 0011	101	
	<b>lw</b> <b>rd imm(rs1)</b>	Load Word	<b>rd = 4 bytes of memory starting at address rs1 + imm</b>	I	000 0011	010	
	<b>sb</b> <b>rs2 imm(rs1)</b>	Store Byte	Stores least-significant byte of <b>rs2</b> at the address <b>rs1 + imm</b> in memory	S	010 0011	000	
	<b>sh</b> <b>rs2 imm(rs1)</b>	Store Half-word	Stores the 2 least-significant bytes of <b>rs2</b> starting at the address <b>rs1 + imm</b> in memory	S	010 0011	001	
	<b>sw</b> <b>rs2 imm(rs1)</b>	Store Word	Stores <b>rs2</b> starting at the address <b>rs1 + imm</b> in memory	S	010 0011	010	

	Instruction	Name	Description	Type	Opcode	Funct3
Control	<b>beq</b> <i>rs1 rs2 label</i>	Branch if Equal	<b>if</b> ( <i>rs1</i> == <i>rs2</i> ) PC = PC + <i>offset</i>	B	110 0011	000
	<b>bge</b> <i>rs1 rs2 label</i>	Branch if Greater or Equal (signed)	<b>if</b> ( <i>rs1</i> >= <i>rs2</i> ) PC = PC + <i>offset</i>	B	110 0011	101
	<b>bgeu</b> <i>rs1 rs2 label</i>	Branch if Greater or Equal (Unsigned)	PC = PC + <i>offset</i>	B	110 0011	111
	<b>blt</b> <i>rs1 rs2 label</i>	Branch if Less Than (signed)	<b>if</b> ( <i>rs1</i> < <i>rs2</i> ) PC = PC + <i>offset</i>	B	110 0011	100
	<b>bltu</b> <i>rs1 rs2 label</i>	Branch if Less Than (Unsigned)	PC = PC + <i>offset</i>	B	110 0011	110
	<b>bne</b> <i>rs1 rs2 label</i>	Branch if Not Equal	<b>if</b> ( <i>rs1</i> != <i>rs2</i> ) PC = PC + <i>offset</i>	B	110 0011	001
	<b>jal</b> <i>rd label</i>	Jump And Link	<i>rd</i> = PC + 4 PC = PC + <i>offset</i>	J	110 1111	
	<b>jalr</b> <i>rd rs1 imm</i>	Jump And Link Register	<i>rd</i> = PC + 4 PC = <i>rs1</i> + <i>imm</i>	I	110 0111	000
Other	<b>auipc</b> <i>rd immu</i>	Add Upper Immediate to PC	<i>imm</i> = <i>immu</i> << 12 <i>rd</i> = PC + <i>imm</i>	U	001 0111	
	<b>lui</b> <i>rd immu</i>	Load Upper Immediate	<i>imm</i> = <i>immu</i> << 12 <i>rd</i> = <i>imm</i>	U	011 0111	
	<b>ebreak</b>	Environment BREAK	Asks the debugger to do something ( <i>imm</i> = 0)	I	111 0011	000
	<b>ecall</b>	Environment CALL	Asks the OS to do something ( <i>imm</i> = 1)	I	111 0011	000
Ext	<b>mul</b> <i>rd rs1 rs2</i>	MULTiply (part of mul ISA extension)	<i>rd</i> = <i>rs1</i> * <i>rs2</i>	(omitted)		

#	Name	Description	#	Name	Desc
<b>x0</b>	<b>zero</b>	Constant 0	<b>x16</b>	<b>a6</b>	<b>Args</b>
<b>x1</b>	<b>ra</b>	<i>Return Address</i>	<b>x17</b>	<b>a7</b>	
<b>x2</b>	<b>sp</b>	Stack Pointer	<b>x18</b>	<b>s2</b>	<b>Saved Registers</b>
<b>x3</b>	<b>gp</b>	Global Pointer	<b>x19</b>	<b>s3</b>	
<b>x4</b>	<b>tp</b>	Thread Pointer	<b>x20</b>	<b>s4</b>	
<b>x5</b>	<b>t0</b>	<i>Temporary Registers</i>	<b>x21</b>	<b>s5</b>	
<b>x6</b>	<b>t1</b>		<b>x22</b>	<b>s6</b>	
<b>x7</b>	<b>t2</b>		<b>x23</b>	<b>s7</b>	
<b>x8</b>	<b>s0</b>	Saved Registers	<b>x24</b>	<b>s8</b>	
<b>x9</b>	<b>s1</b>		<b>x25</b>	<b>s9</b>	
<b>x10</b>	<b>a0</b>	<i>Function Arguments or Return Values</i>	<b>x26</b>	<b>s10</b>	
<b>x11</b>	<b>a1</b>		<b>x27</b>	<b>s11</b>	
<b>x12</b>	<b>a2</b>	<i>Function Arguments</i>	<b>x28</b>	<b>t3</b>	<b>Temporaries</b>
<b>x13</b>	<b>a3</b>		<b>x29</b>	<b>t4</b>	
<b>x14</b>	<b>a4</b>		<b>x30</b>	<b>t5</b>	
<b>x15</b>	<b>a5</b>		<b>x31</b>	<b>t6</b>	
<i>Caller saved registers</i>					
<i>Callee saved registers (except <b>x0</b>, <b>gp</b>, <b>tp</b>)</i>					

Immediates are sign-extended to 32 bits, except in I\* type instructions

Pseudoinstruction	Name	Description	Translation
<b>beqz</b> <i>rs1 label</i>	Branch if Equals Zero	<b>if</b> ( <i>rs1</i> == 0) PC = PC + <i>offset</i>	<b>beq</b> <i>rs1</i> <b>x0</b> <i>label</i>
<b>bnez</b> <i>rs1 label</i>	Branch if Not Equals Zero	<b>if</b> ( <i>rs1</i> != 0) PC = PC + <i>offset</i>	<b>bne</b> <i>rs1</i> <b>x0</b> <i>label</i>
<b>j</b> <i>label</i>	Jump	PC = PC + <i>offset</i>	<b>jal</b> <b>x0</b> <i>label</i>
<b>jal</b> <i>label</i>	Jump and Link	PC = PC + <i>offset</i> <i>ra</i> = PC + 4	<b>jal</b> <i>ra</i> <i>label</i>
<b>jr</b> <i>rs1</i>	Jump Register	PC = <i>rs1</i>	<b>jalr</b> <b>x0</b> <i>rs1</i> 0
<b>la</b> <i>rd label</i>	Load absolute Address	<i>rd</i> = & <i>label</i>	<b>auipc</b> , <b>addi</b>
<b>li</b> <i>rd imm</i>	Load Immediate	<i>rd</i> = <i>imm</i>	<b>lui</b> (if needed), <b>addi</b>
<b>mv</b> <i>rd rs1</i>	MoVe	<i>rd</i> = <i>rs1</i>	<b>addi</b> <i>rd</i> <i>rs1</i> 0
<b>neg</b> <i>rd rs1</i>	NEGate	<i>rd</i> = - <i>rs1</i>	<b>sub</b> <i>rd</i> <b>x0</b> <i>rs1</i>
<b>nop</b>	No OPERATION	do nothing	<b>addi</b> <b>x0</b> <b>x0</b> 0
<b>not</b> <i>rd rs1</i>	bitwise NOT	<i>rd</i> = ~ <i>rs1</i>	<b>xori</b> <i>rd</i> <i>rs1</i> -1
<b>ret</b>	RETurn	PC = <i>ra</i>	<b>jalr</b> <b>x0</b> <i>x1</i> 0

312524201915141211760

R	funct7	rs2	rs1	funct3	rd	opcode
I	imm[11:0]		rs1	funct3	rd	opcode
I*	funct7	imm[4:0]	rs1	funct3	rd	opcode
S	imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode
B	imm[12 10:5]	rs2	rs1	funct3	imm[4:1 11]	opcode
U	imm[31:12]				rd	opcode
J	imm[20 10:1 11 19:12]				rd	opcode

## Selected ASCII values

HEX	DEC	CHAR	HEX	DEC	CHAR	HEX	DEC	CHAR	HEX	DEC	CHAR	HEX	DEC	CHAR	HEX	DEC	CHAR
0x20	32	SPACE	0x30	48	0	0x40	64	@	0x50	80	P	0x60	96	`	0x70	112	p
0x21	33	!	0x31	49	1	0x41	65	A	0x51	81	Q	0x61	97	a	0x71	113	q
0x22	34	"	0x32	50	2	0x42	66	B	0x52	82	R	0x62	98	b	0x72	114	r
0x23	35	#	0x33	51	3	0x43	67	C	0x53	83	S	0x63	99	c	0x73	115	s
0x24	36	\$	0x34	52	4	0x44	68	D	0x54	84	T	0x64	100	d	0x74	116	t
0x25	37	%	0x35	53	5	0x45	69	E	0x55	85	U	0x65	101	e	0x75	117	u
0x26	38	&	0x36	54	6	0x46	70	F	0x56	86	V	0x66	102	f	0x76	118	v
0x27	39	'	0x37	55	7	0x47	71	G	0x57	87	W	0x67	103	g	0x77	119	w
0x28	40	(	0x38	56	8	0x48	72	H	0x58	88	X	0x68	104	h	0x78	120	x
0x29	41	)	0x39	57	9	0x49	73	I	0x59	89	Y	0x69	105	i	0x79	121	y
0x2A	42	*	0x3A	58	:	0x4A	74	J	0x5A	90	Z	0x6A	106	j	0x7A	122	z
0x2B	43	+	0x3B	59	;	0x4B	75	K	0x5B	91	[	0x6B	107	k	0x7B	123	{
0x2C	44	,	0x3C	60	<	0x4C	76	L	0x5C	92	\	0x6C	108	l	0x7C	124	
0x2D	45	-	0x3D	61	=	0x4D	77	M	0x5D	93	]	0x6D	109	m	0x7D	125	}
0x2E	46	.	0x3E	62	>	0x4E	78	N	0x5E	94	^	0x6E	110	n	0x7E	126	~
0x2F	47	/	0x3F	63	?	0x4F	79	O	0x5F	95	_	0x6F	111	o	0x00	0	NULL

### C Format String Specifiers

Specifier	Output
d or i	Signed decimal integer
u	Unsigned decimal integer
o	Unsigned octal
x	Unsigned hexadecimal integer, lowercase
X	Unsigned hexadecimal integer, uppercase
f	Decimal floating point, lowercase
F	Decimal floating point, uppercase
e	Scientific notation (significand/exponent), lowercase
E	Scientific notation (significand/exponent), uppercase
g	Use the shortest representation: %e or %f
G	Use the shortest representation: %E or %F
a	Hexadecimal floating point, lowercase
A	Hexadecimal floating point, uppercase
c	Character
s	String of characters
p	Pointer address

### IEEE 754 Floating Point Standard

	Sign	Exponent	Significand
Single Precision	1 bit	8 bits (bias = -127)	23 bits
Double Precision	1 bit	11 bits (bias = -1023)	52 bits
Quad Precision	1 bit	15 bits (bias = -16383)	112 bits

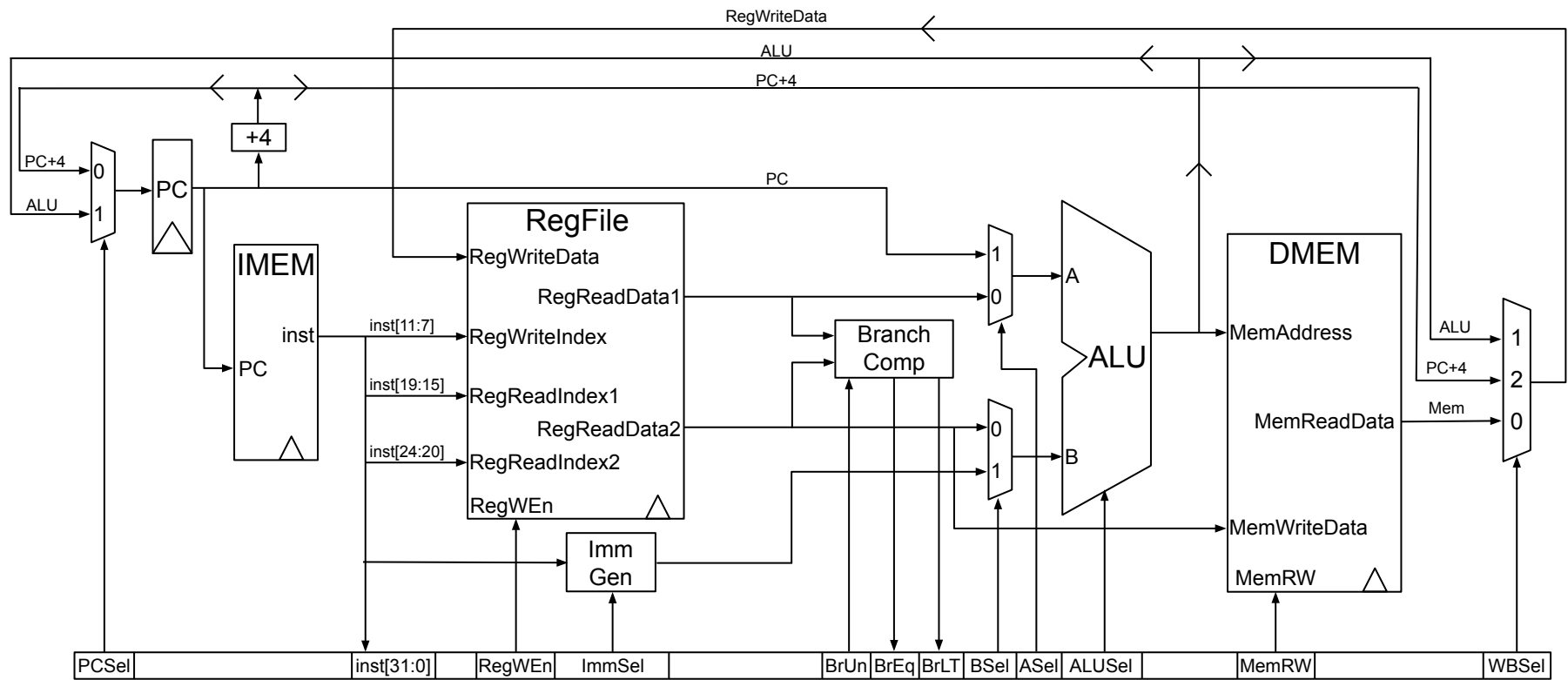
Standard exponent bias:  $-(2^{E-1}-1)$  where E is the number of exponent bits

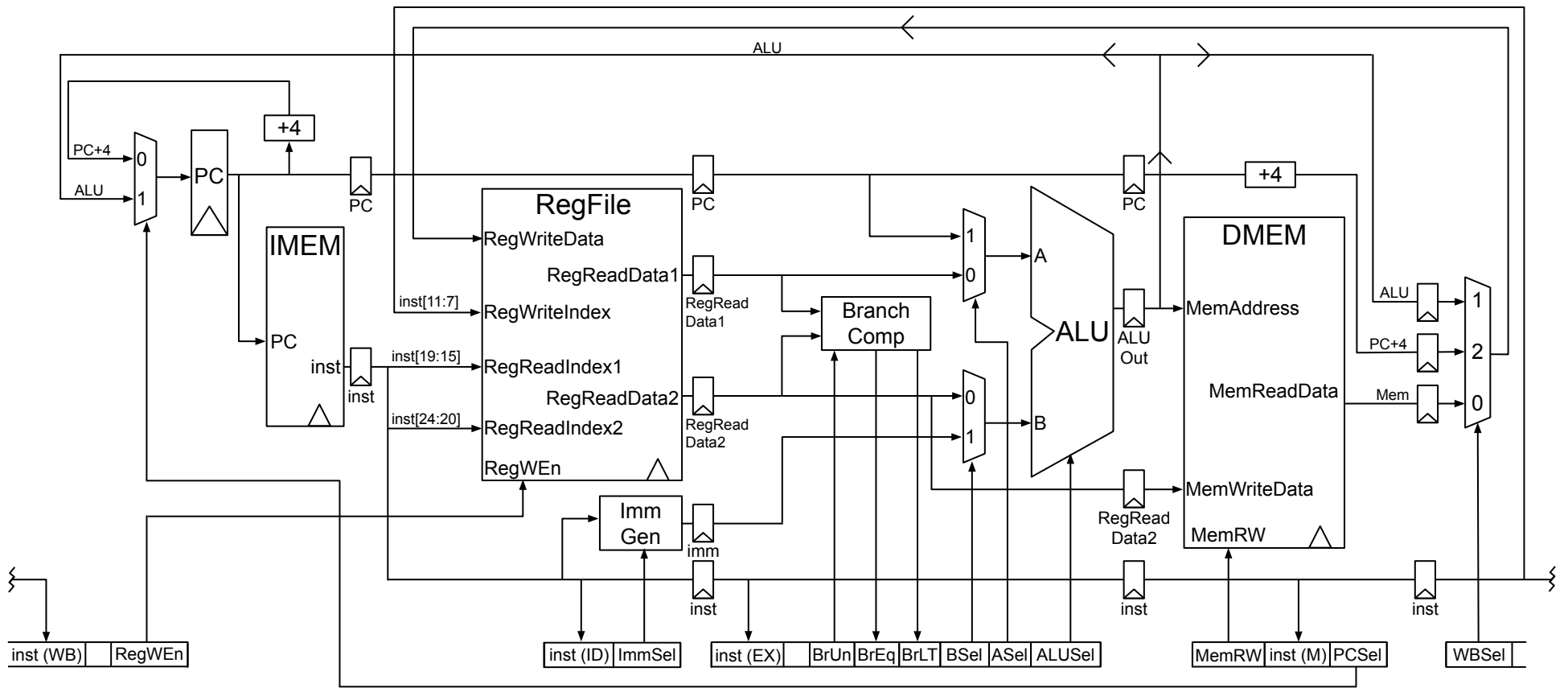
### SI Prefixes

Size	Prefix	Symbol	Size	Prefix	Symbol	Size	Prefix	Symbol
$10^{-3}$	milli-	m	$10^3$	kilo-	k	$2^{10}$	kibi-	Ki
$10^{-6}$	micro-	$\mu$	$10^6$	mega-	M	$2^{20}$	mebi-	Mi
$10^{-9}$	nano-	n	$10^9$	giga-	G	$2^{30}$	gibi-	Gi
$10^{-12}$	pico-	p	$10^{12}$	tera-	T	$2^{40}$	tebi-	Ti
$10^{-15}$	femto-	f	$10^{15}$	peta-	P	$2^{50}$	pebi-	Pi
$10^{-18}$	atto-	a	$10^{18}$	exa-	E	$2^{60}$	exbi-	Ei
$10^{-21}$	zepto-	z	$10^{21}$	zetta-	Z	$2^{70}$	zebi-	Zi
$10^{-24}$	yocto-	y	$10^{24}$	yotta-	Y	$2^{80}$	yobi-	Yi

### Laws of Boolean Algebra

$$\begin{array}{lll}
 x \cdot \bar{x} = 0 & x + \bar{x} = 1 & (xy)z = x(yz) \\
 x \cdot 0 = 0 & x + 1 = 1 & (x + y) + z = x + (y + z) \\
 x \cdot 1 = x & x + 0 = x & x(y + z) = xy + xz \\
 x \cdot x = x & x + x = x & x + yz = (x + y)(x + z) \\
 x \cdot y = y \cdot x & x + y = y + x & \overline{x \cdot y} = \bar{x} + \bar{y} \\
 xy + x = x & (x + y)x = x & \overline{(x + y)} = \bar{x} \cdot \bar{y}
 \end{array}$$





Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
Parity bit coverage	p1	✓		✓		✓		✓		✓		✓		✓		✓		✓		✓		
	p2		✓	✓			✓	✓			✓	✓			✓	✓			✓	✓		
	p4				✓	✓	✓	✓					✓	✓	✓	✓					✓	
	p8								✓	✓	✓	✓	✓	✓	✓	✓						
	p16																✓	✓	✓	✓	✓	