Hexadecimal Reference Table

HEX	DEC	BIN	2's	1's	x16	/16
0	0	0000	0	F	0	0.0000
1	1	0001	F	E	16	0.0625
2	2	0010	E	D	32	0.1250
3	3	0011	D	C	48	0.1875
4	4	0100	C	B	64	0.2500
5	5	0101	B	A	80	0.3125
6	6	0110	A	9	96	0.3750
7	7	0111	9	8	112	0.4375
8	8	1000	8	7	128	0.5000
9	9	1001	7	6	144	0.5625
A	10	1010	6	5	160	0.6250
B	11	1011	5	4	176	0.6875
C	12	1100	4	3	192	0.7500
D	13	1101	3	2	208	0.8125
E	14	1110	2	1	224	0.8750
F	15	1111	1	0	240	0.9375

Hexadecimal Addition Table

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

Hexadecimal Multiplication Table

*	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	0	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	0	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	0	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	0	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	0	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	0	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	0	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	0	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

ACME-1021 Instruction Set

Basic Instruction Set available in Decimal or Binary Mode

- NOP
  - No Operation - do nothing and proceed to next instruction.
- JMP N
  - Set Card #0 to Value N (N is a Program Memory card number).
- SKP C
  - If Value stored on Card C is not zero, then increment value on Card #0 by one.
    - This has the effect of skipping the next instruction.
- SET C, N
  - Set Card C to the Value N
- ADD C, Na, Nb
  - Set Card C to sum of Value Na and Value Nb.
    - If a carry occurs, value stored is (Na + Nb) - 100
  - Set Card #1 to 0 if no carry and to 1 if a carry occurs.
  - If C is 1, then the carry overwrites the sum - useful for testing values
- SUB C, Na, Nb
  - Set Card C to difference of Value Na and Value Nb (Na - Nb).
    - If a borrow occurs, value stored is 100 - (Na - Nb)
  - Set Card #1 to 0 if no borrow and to 1 if a borrow occurs.
  - If C is 1, then the borrow overwrites the difference - useful for testing values
- MUL C, Na, Nb
  - Set Card C to LSB of product of Value Na and Value Nb.
  - Set Card C+1 to MSB of product of Value Na and Value Nb.
- DIV C, Na, Nb
  - Set Card C to the integer portion of the quotient of Value Na and Value Nb. (Na/Nb)
  - Set Card C+1 to the remainder.
- GET C
  - Get a single value from User and store on Card C.
- OUT C
  - Output contents of Card C.

Extended Instruction Set available in Binary Mode only

- INV C, N
  - Set Card C to bitwise inversion of Value N.
- AND C, Na, Nb
  - Set Card C to bitwise AND of Value Na and Value Nb.
- IOR C, Na, Nb
  - Set Card C to bitwise Inclusive OR of Value Na and Value Nb.
- XOR C, Na, Nb
  - Set Card C to bitwise Exclusive OR of Value Na and Value Nb.
- GETC C
  - Get a single character from the User and store its ASCII code on Card C.
- OUTC C
  - Output contents of Card C as a single ASCII character.

7-bit ASCII Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

In the above table, the upper (most significant) nibble is in the leftmost column and the lower (least significant) nibble is in the top row. For instance, the ASCII code for the letter 'G' is 0x47.

Most of the first thirty-two codes are control codes for equipment no longer in use today. As a result these codes are seldom seen in practice anymore. But a small subset of the control codes are still used.

HEX	Decimal	CODE	Name
0x00	0	NUL	Null Character
0x07	7	BEL	Bell (Buzzer)
0x08	8	BS	Backspace
0x09	9	HT	Horizontal Tab
0x0A	10	LF	Line Feed
0x0B	11	VT	Vertical Tab
0x0C	12	FF	Form Feed
0x0D	13	CR	Carriage Return
0x1B	27	ESC	Escape
0x20	32	SP	Space
0x7F	127	DEL	Delete

Precedence of C Operators

LEVEL	Description	Operator	Associativity
1	Function Expression	( )	left
	Array Expression	[ ]	
	struct indirection	->	
	struct member	.	
2	Increment / Decrement	++ --	right
	Bitwise invert (1's complement)	~	
	Logical NOT (unary NOT)	!	
	Address	&	
	Dereference	*	
	Cast	(type)	
	Unary plus (positive sign)	+	
	Unary minus (negative sign)	-	
	Size in bytes	sizeof	
3	Multiplication	*	left
	Division	/	
	Modulus	%	
4	Addition	+	
	Subtraction	-	
5	Shift Left	<<	
	Shift Right	>>	
6	Less than	<	
	Less than or equal	<=	
	Greater than	>	
	Greater than or equal	>=	
7	Equal	==	
	Not equal	!=	
8	Bitwise AND	&	
9	Bitwise XOR	^	
10	Bitwise OR	|	
11	Logical AND	&&	
12	Logical OR	||	
13	Conditional	? :	right
14	Assignment	=	
15	Comma	.	left

Format codes for printf() function family

Format Specifier: %[flags][width][.precision][modifier][type]

Required type and optional modifiers

Modifier	Type	Format	Corresponding argument must be
INTEGER TYPES			
	c	character	char, short, or int (automatically promoted to int)
	s	string	address of char (first character of null-terminated string)
	d, i	decimal	char, short, or int (automatically promoted by to int)
	o	octal	
	x	hex (a..f)	
	X	hex (A..F)	
l	d, i	decimal	long
	o	octal	
	x	hex (A..F)	
	X	hex (A..F)	
	u	decimal	unsigned char, unsigned short, unsigned int (automatically promoted to unsigned int)
l	u	decimal	unsigned long
FLOATING POINT TYPES			
	f	m.nnnnnn	float or double (automatically promoted to double) g and G use which ever format is shortest
	e	m.nnnnnne±xx	
	E	m.nnnnnnE±xx	
	g	d, e, or f	
	G	d, E, or f	
L	f	m.nnnnnn	long double g and G use which ever format is shortest
	e	m.nnnnnne±xx	
	E	m.nnnnnnE±xx	
	g	d, Le, or Lf	
	G	d, LE, or Lf	
POINTER TYPES			
	p	address	pointer (i.e., an address)
REPORT BACK			
	n	characters written so far	address of variable of type int
h	n		address of variable of type short
l	n		address of variable of type long
SPECIAL CHARACTER			
	%	%	no argument required/allowed

Optional Flags

Flag	Meaning
-	Left justify the result
+	Begin number with appropriate sign (+ or -)
space	Add space in place of an unused sign
#	type {o} : begin number with 0 (zero) to indicate octal
	type {x} : begin number with 0x to indicate hex
	type {X} : begin number with 0X to indicate hex
	type {e, E, f}: use a decimal point
	type {g, G}: use a decimal point and do not remove trailing zeros.
0 (zero)	Pad the number to the left with zeros

Format codes for scanf() function family

Conversion Code format: %[flags][width][modifier][type]

Required type and optional modifiers

Modifier	Type	Interpretation	Corresponding argument must be
INTEGER TYPES			
	c	character	address of char
	s	string	address of char
h	d	decimal integer	address of short
			address of int
l			address of long
h	o	octal integer	address of short
			address of int
l			address of long
h	x or X	hexadecimal integer	address of short
			address of int
l			address of long
h	i	nnn - decimal integer 0nnn - octal integer 0xnnn - hex integer	address of short
			address of int
l			address of long
h	u	unsigned decimal integer	address of short
			address of int
l			address of long
FLOATING POINT TYPES			
	f	floating point value - decimal or exponential format acceptable	address of float
l	e, E		address of double
L	g, G		address of long double
POINTER TYPES			
	p	address	address of a pointer
REPORT BACK			
	n	characters read so far	address of variable of type int
h			address of variable of type short
l			address of variable of type long
SPECIAL CHARACTER			
	%	match the % in the input	no argument required/allowed
MATCHING/EXCLUSION			
	[chars]	string that only has chars	address of char
	[^chars]	string that excludes chars	address of char

Optional Flags

Flag	Meaning
*	Suppress storage of scanned result (hence no corresponding address in list)

Width

- Positive integer which specifies the maximum width of the input field. Actual width may be smaller.

Characters not in a conversion code:

- whitespace - causes any whitespace to be ignored up to the next non-whitespace character.
- non-whitespace - requires an exact match to next character in input stream, which is then skipped.

Function prototypes of commonly used functions

prototype	Comments
<stdio.h>	
int printf(const char *fs [, arg, ...]);	fs = format string
int fprintf(FILE *fp,const char *fs [, arg, ...]);	
int sprintf(char *dest, const char *fs [, arg, ...]);	
int scanf(const char *fs [, arg, ...]);	fs = format string
int fscanf(FILE *fp,const char *fs [, arg, ...]);	
int sscanf(const char *src, const char *fs [, arg, ...]);	
FILE *fopen(const char *fn, const char *mode);	fn = filename, mode ex: "wt" "rb"
int fclose(FILE *fp);	returns 0 on success, EOF otherwise
<stdlib.h>	
int rand(void);	random integer [0 ,,, RAND_MAX]
void srand(unsigned int);	seed the random number generator
double sin(double radians);	argument in radians
double cos(double radians);	
double tan(double radians);	
double arcsin(double x);	return value in radians
double arccos(double x);	
double arctan(double x);	
double atan2(double y, double x);	
double exp(double x);	returns e^x
double log(double x);	returns natural logarithm of x
double log10(double radians);	returns base ten logarithm of x
double fabs(double x);	absolute value of a floating point value
<math.h>	
double sin(double radians);	argument in radians
double cos(double radians);	
double tan(double radians);	
double arcsin(double x);	returns -(PI/2) < radians < +(PI/2)
double arccos(double x);	
double arctan(double x);	
double atan2(double y, double x);	returns -PI < radians < +PI
double exp(double x);	returns e^x
double log(double x);	returns natural logarithm of x
double log10(double radians);	returns base ten logarithm of x
double pow(double x, double y);	returns x to the power of y
double fabs(double x);	absolute value of a floating point value
double ceil(double x);	smallest integer not less than x
double floor(double x);	largest integer not larger than x