CPSC-240 Computer Organization and Assembly Language

Chapter 3

Data Representation

Instructor: Yitsen Ku, Ph.D.
Department of Computer Science,
California State University, Fullerton, USA

CALIFORNIA STATE UNIVERSITY FULLERTON™

Outline

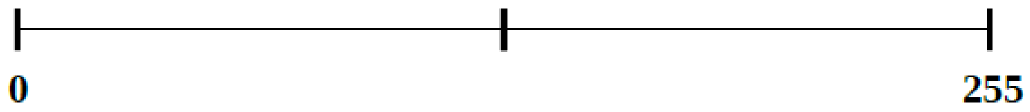
- Integer Representation
- Unsigned and Signed Addition
- Floating-point Representation
- Characters and Strings

Integer Representation

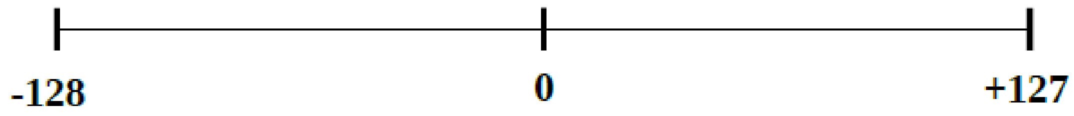
Integer Representation

Size	Size	Unsigned Range	Signed Range
Bytes (8-bits)	2^8	0 to 255	-128 to +127
Words (16-bits)	2^{16}	0 to 65,535	−32,768 to +32,767
Double-words (32-bits)	2^{32}	0 to 4,294,967,295	−2,147,483,648 to +2,147,483,647
Quadword	2^{64}	0 to 2^{64} - 1	$-(2^{63})$ to $2^{63} - 1$
Double quadword	2^{128}	0 to 2^{128} - 1	$-(2^{127})$ to $2^{127} - 1$

- The unsigned byte range:



- The signed byte range:



Two's Complement

To take the two's complement of a number (a negative value of a number):

1. take the one's complement (negate)
2. add 1 (in binary)

Ex. A signed byte representation of -15_{10} is $0xF1_{16}$

$15_{10} = 0x0F_{16} = 0000\ 1111_2$

Not $(0000\ 1111_2) = 1111\ 0000_2$;take the one's complement

$1111\ 0000_2 + 0000\ 0001_2 = 1111\ 0001_2$;two's complement

Byte Example

To find the byte size (8-bits), two's complement representation of -9 and -12.

9 (8+1)=	0000 1001
Step 1	1111 0110
Step 2	1111 0111
-9 (in hex) =	F7

12 (8+4) =	0000 1100
Step 1	1111 0011
Step 2	1111 0100
-12 (in hex) =	F4

Word Example

To find the word size (16-bits), two's complement representation of -18 and -40.

18 (16+2)=	0000 0000 0001 0010
Step 1	1111 1111 1110 1101
Step 2	1111 1111 1110 1110
-18 (in hex) =	0xFFFE

40 (32+8)=	0000 0000 0010 1000
Step 1	1111 1111 1101 0111
Step 2	1111 1111 1101 1000
-40 (in hex) =	0xFFD8

Unsigned and Signed Addition

Unsigned and Signed Byte Addition

As previously noted, the unsigned and signed representations may provide different interpretations for the final value being represented. However, the addition and subtraction operations are the same.

For example:

Unsigned addition	
241	1111 0001
+ 7	0000 0111
248	1111 1000
248 (in hex) =	F8

Signed addition	
-15	1111 0001
+ 7	0000 0111
-8	1111 1000
-8 (in hex) =	F8

Floating-point Representation

IEEE 32-bit Representation

The IEEE 754 32-bit floating-point standard is defined as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	biased exponent								fraction																						

Where ***s*** is the sign (0 => positive and 1 => negative). More formally, this can be written as;

$$\boldsymbol{N = (-1)^S \times 1.F \times 2^{E-127}}$$

IEEE 32-bit Representation

The following table provides a brief reminder of how binary handles fractional components:

	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}	
...	8	4	2	1	.	1/2	1/4	1/8	...
	0	**0**	**0**	**0**	**.**	**0**	**0**	**0**	

Ex. $100.101_2 = 4.625_{10}$

Ex. $8.125_{10} = 1000.001_2 = 1000.001_2 \times 2^0 = 1.000001_2 \times 2^3$

Ex. $0.125_{10} = 0.001_2 = 0.001_2 \times 2^0 = 1.0_2 \times 2^{-3}$

IEEE 32-bit Representation Examples

Example 1: find 32-bit floating-point representation for -7.75

- determine sign -7.75 => 1 (since negative)
- convert to binary $-7.75 = -0111.11_2$
- normalized scientific notation $= 1.1111 \times 2^2$
- compute biased exponent $2_{10} + 127_{10} = 129_{10}$
- ◦ and convert to binary $= 10000001_2$
- write components in binary:
- sign exponent mantissa (fraction)
- 1 10000001 11110000000000000000000
- convert to hex (split into groups of 4)
- 11000000111110000000000000000000
- 1100 0000 1111 1000 0000 0000 0000 0000
- C 0 F 8 0 0 0 0
- final result: $\mathbf{C0F8\ 0000_{16}}$

IEEE 32-bit Representation Examples

Example 2: find 32-bit floating-point representation for -0.125

- determine sign -0.125 => 1 (since negative)
- convert to binary $-0.125 = -0.001_2$
- normalized scientific notation $= 1.0 \times 2^{-3}$
- compute biased exponent $-3_{10} + 127_{10} = 124_{10}$
- ◦ and convert to binary $= 01111100_2$
- write components in binary:
- sign exponent mantissa (fraction)
- 1 01111100 00000000000000000000000
- convert to hex (split into groups of 4)
- 10111110000000000000000000000000
- 1011 1110 0000 0000 0000 0000 0000 0000
- B E 0 0 0 0 0 0
- final result: **BE00 0000**$_{16}$

IEEE 32-bit Representation Examples

Example 3: find the decimal value of 41440000_{16}

- convert to binary
- $0100\ 0001\ 0100\ 0100\ 0000\ 0000\ 0000\ 0000_2$
- Split into components:
- sign exponent mantissa (fraction)
- 0 10000010 $10001000000000000000000_2$
- Determine exponent $10000010_2 = 130_{10}$
- ◦ and remove bias $130_{10} - 127_{10} = 3_{10}$
- determine sign 0 => positive
- write result $+1.10001 \times 2^3 = +1100.01 =$ **+12.25**

IEEE 64-bit Representation

The IEEE 754 64-bit floating-point standard is defined as follows:

63	62		52	51		0
s	biased exponent			fraction		

The representation process is the same, however the format allows for an 11-bit biased exponent (which support large and smaller values). The 11-bit biased exponent uses a bias of ±1023.

Not a Number (NaN)

- When a value is interpreted as a floating-point value and it does not conform to the defined standard (either for 32-bit or 64-bit), then it cannot be used as a floating-point value.
- This might occur if an integer representation is treated as a floating-point representation or a floating-point arithmetic operation (add, subtract, multiply, or divide) results in a value that is too large or too small to be represented.
- The incorrect format or un-representable number is referred to as a **NaN** which is an abbreviation for *not a number*.

Characters and Strings

Characters and Strings

- In addition to numeric data, symbolic data is often required. Symbolic or non-numeric data might include an important message such as "Hello World" a common greeting for first programs. Such symbols are well understood by English language speakers.
- Computer memory is designed to store and retrieve numbers. Consequently, the symbols are represented by assigning numeric values to each symbol or character.

Character Representation

- In a computer, a character is a unit of information that corresponds to a symbol such as a letter in the alphabet. Examples of characters include letters, numerical digits, common punctuation marks (such as "." or "!"), and whitespace.
- The general concept also includes control characters, which do not correspond to symbols in a particular language, but to other information used to process text. Examples of control characters include carriage return or tab.

American Standard Code for Information Interchange

Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value
00	NUL	10	DLE	20	SP	30	0	40	@	50	P	60	`	70	p
01	SOH	11	DC1	21	!	31	1	41	A	51	Q	61	a	71	q
02	STX	12	DC2	22	"	32	2	42	B	52	R	62	b	72	r
03	ETX	13	DC3	23	#	33	3	43	C	53	S	63	c	73	s
04	EOT	14	DC4	24	$	34	4	44	D	54	T	64	d	74	t
05	ENQ	15	NAK	25	%	35	5	45	E	55	U	65	e	75	u
06	ACK	16	SYN	26	&	36	6	46	F	56	V	66	f	76	v
07	BEL	17	ETB	27	'	37	7	47	G	57	W	67	g	77	w
08	BS	18	CAN	28	(	38	8	48	H	58	X	68	h	78	x
09	HT	19	EM	29	)	39	9	49	I	59	Y	69	i	79	y
0A	LF	1A	SUB	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
0B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
0C	FF	1C	FS	2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
0D	CR	1D	GS	2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
0E	SO	1E	RS	2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
0F	SI	1F	US	2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

Unicode

- It should be noted that Unicode is a current standard that includes support for different languages.
- The Unicode Standard provides series of different encoding schemes (UTF-8, UTF-16, UTF-32, etc.) in order to provide a unique number for every character, no matter what platform, device, application or language.
- In the most common encoding scheme, UTF-8, the ASCII English text looks exactly the same in UTF-8 as it did in ASCII. Additional bytes are used for other characters as needed. Details regarding Unicode representation are not addressed in this text.

String Representation

- A string is a series of ASCII characters, typically terminated with a NULL. The NULL is a non-printable ASCII control character. Since it is not printable, it can be used to mark the end of a string.

String Representation

Ex1. the string "Hello" would be represented as follows:

Character	“H”	“e”	“l”	“l”	“o”	NULL
ASCII Value (decimal)	72	101	108	108	111	0
ASCII Value (hex)	0x48	0x65	0x6C	0x6C	0x6F	0x0

Ex2. the string "19653" would be represented as follows:

Character	“1”	“9”	“6”	“5”	“3”	NULL
ASCII Value (decimal)	49	57	54	53	51	0
ASCII Value (hex)	0x31	0x39	0x36	0x35	0x33	0x0

Thanks