Data Types

Data Types

Bit: 0, 1

Bit String: sequence of bits of a particular length
- 4 bits is a nibble
- 8 bits is a byte
- 16 bits is a half-word (VAX: word)
- 32 bits is a word (VAX: long word)

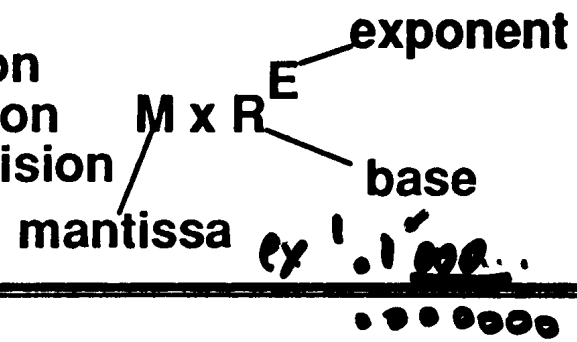
Character:
- ASCII 7 bit code
- EBCDIC 8 bit code

Decimal:
- digits 0-9 encoded as 0000b thru 1001b
- two decimal digits packed per 8 bit byte

Integers:

Sign & Magnitude:	0X vs. 1X	Positive #'s same in all
1's Complement:	0X vs. 1(~X)	First 2 have two zeros
2's Complement:	0X vs. (1's comp) + 1	Last one usually chosen

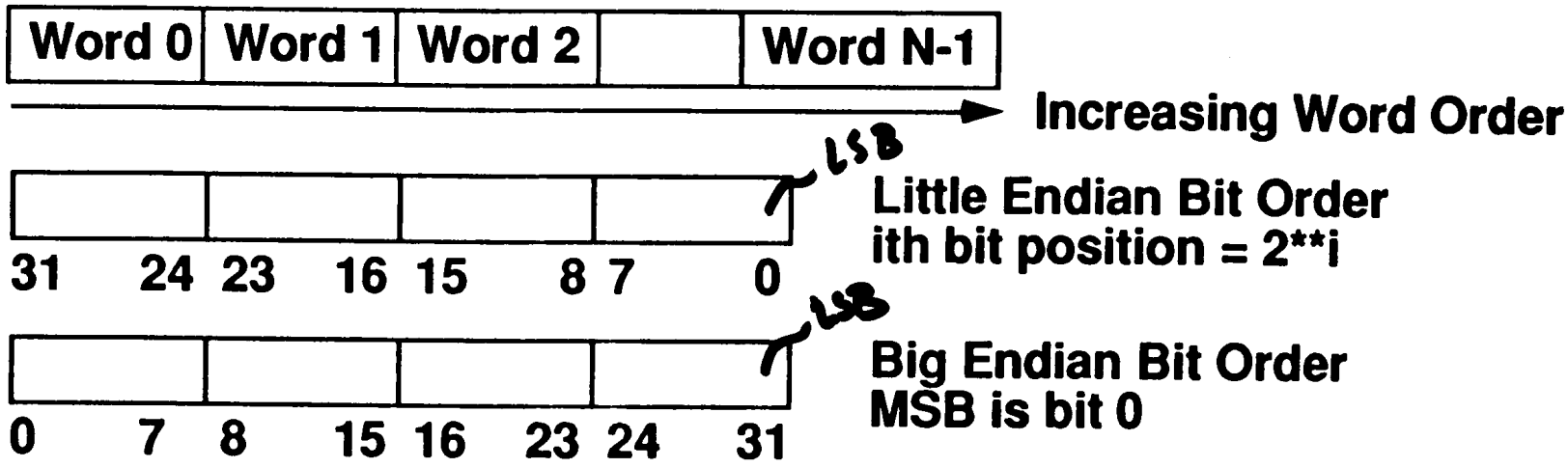
Floating Point:
- Single Precision
- Double Precision
- Extended Precision



How many +/- #'s?
Where is decimal pt?
How are +/- exponents represented?

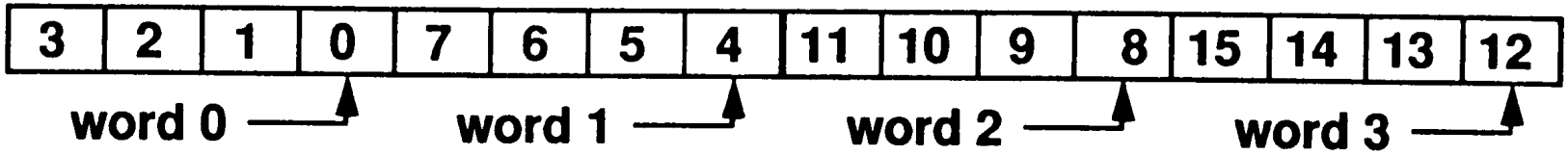
Word and Byte Orderings

Big Endians vs. Little Endians

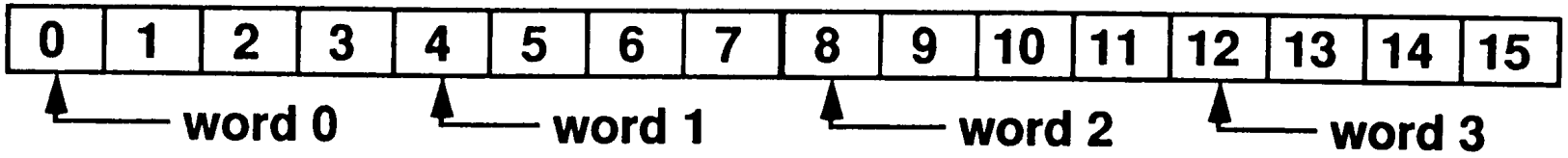


Assume Little Endian *Bit Ordering* and Byte Addressed Machine:

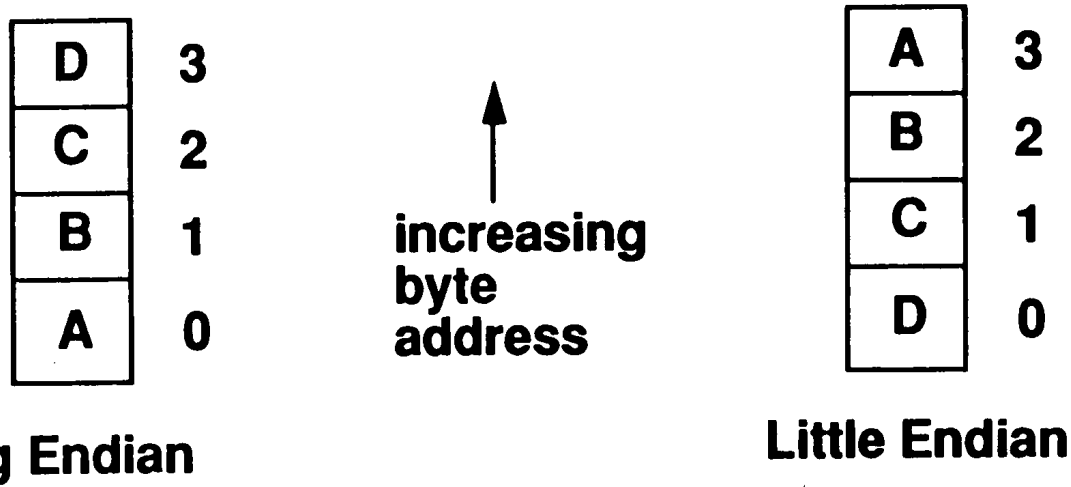
Little Endian Byte Ordering: VAX



Big Endian Byte Ordering: IBM, Motorola 68000



Byte Swap Problem



Big Endian

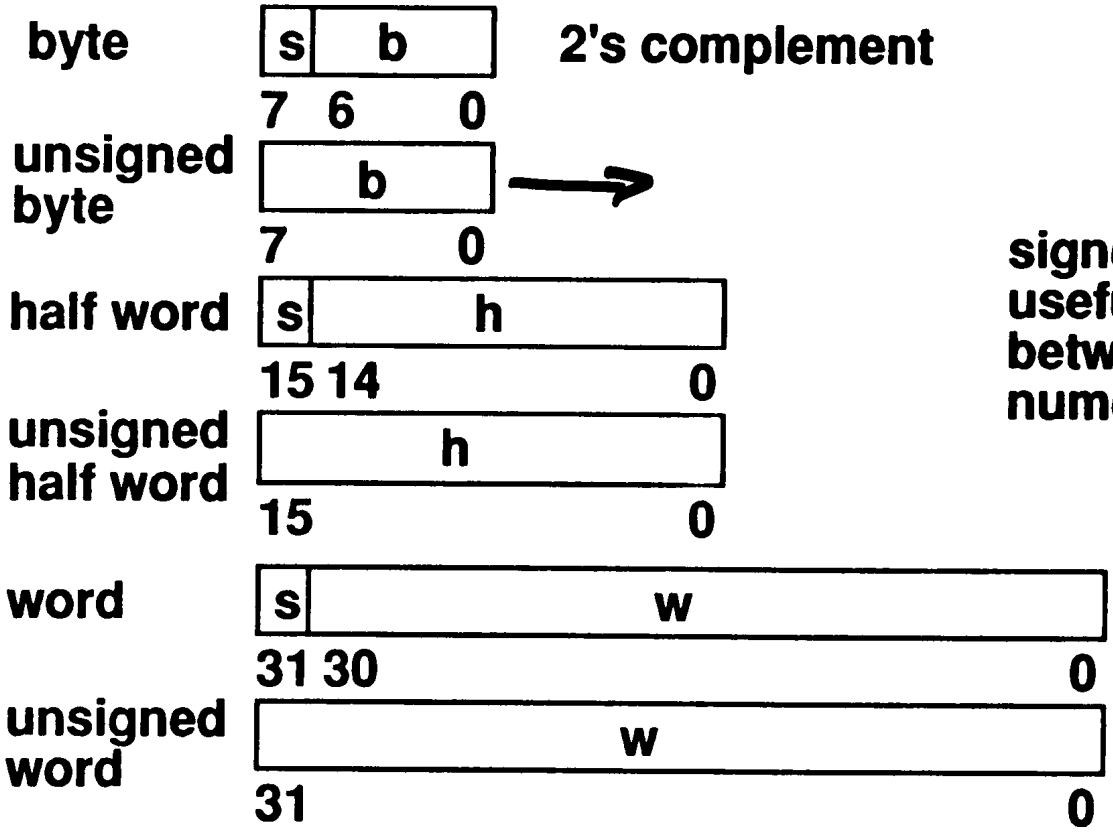
Little Endian

When words are transfered between Big Endian and Little Endian machines, you must permute the bytes to successfully copy the data

Each system is self-consistent, but causes problems when they need communicate!

MIPS Operands

MIPS has little endian bit ordering and configurable byte ordering during hardware reset (big/little endian)



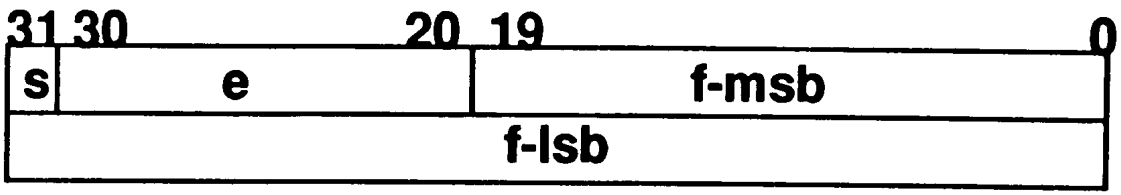
signed vs. unsigned useful to distinguish between character and numeric comparisons

MIPS can also access triple bytes

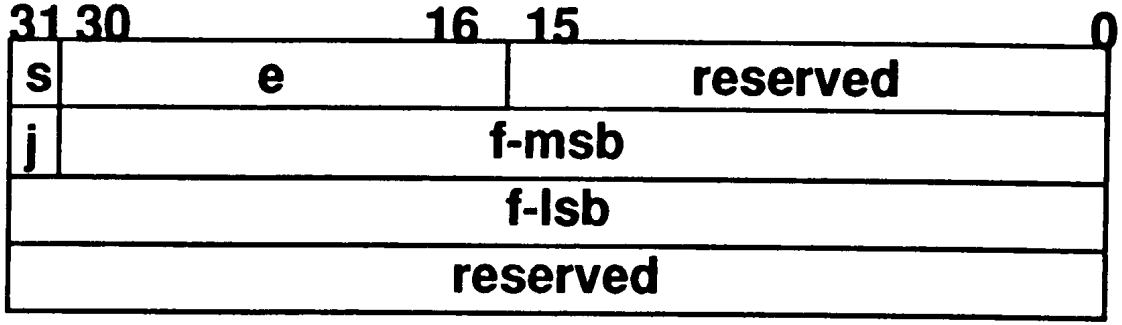
FP Operands

single precision

double precision



extended precision



Single Precision:
1 sign bit
8 exponent bits
23 fraction bits

$(-1)^{s} * 2^{e-127} * 1.f$

Double Precision:
1 sign bit
11 exponent bits
52 fraction bits

$(-1)^{s} * 2^{e-1023} * 1.f$

Extended Precision:
1 sign bit
15 exponent bits
1 integer bit
63 fraction bits

$(-1)^{s} * 2^{e-16383} * i.f$

TABLE 4.4
Decimal and hexadecimal values for the ASCII table.

dec	hex	char	dec	hex	char	dec	hex	char	dec	hex	char
0	0	nul	32	20	sp	64	40	@	96	60	‘
1	1	soh	33	21	!	65	41	A	97	61	a
2	2	stx	34	22	”	66	42	B	98	62	b
3	3	etx	35	23	#	67	43	C	99	63	c
4	4	eot	36	24	$	68	44	D	100	64	d
5	5	enq	37	25	%	69	45	E	101	65	e
6	6	ack	38	26	&	70	46	F	102	66	f
7	7	bel	39	27	’	71	47	G	103	67	g
8	8	bs	40	28	(	72	48	H	104	68	h
9	9	ht	41	29	)	73	49	I	105	69	i
10	a	nl	42	2a	*	74	4a	J	106	6a	j
11	b	vt	43	2b	+	75	4b	K	107	6b	k
12	c	np	44	2c	,	76	4c	L	108	6c	l
13	d	cr	45	2d	–	77	4d	M	109	6d	m
14	e	so	46	2e	.	78	4e	N	110	6e	n
15	f	si	47	2f	/	79	4f	O	111	6f	o
16	10	dle	48	30	0	80	50	P	112	70	p
17	11	dc1	49	31	1	81	51	Q	113	71	q
18	12	dc2	50	32	2	82	52	R	114	72	r
19	13	dc3	51	33	3	83	53	S	115	73	s
20	14	dc4	52	34	4	84	54	T	116	74	t
21	15	nak	53	35	5	85	55	U	117	75	u
22	16	syn	54	36	6	86	56	V	118	76	v
23	17	etb	55	37	7	87	57	W	119	77	w
24	18	can	56	38	8	88	58	X	120	78	x
25	19	em	57	39	9	89	59	Y	121	79	y
26	1a	sub	58	3a	:	90	5a	Z	122	7a	z
27	1b	esc	59	3b	;	91	5b	[	123	7b	{
28	1c	fs	60	3c	<	92	5c	\	124	7c	|
29	1d	gs	61	3d	=	93	5d	]	125	7d	}
30	1e	rs	62	3e	>	94	5e	^	126	7e	~
31	1f	us	63	3f	?	95	5f	_	127	7f	del

TABLE 4.1
Sample of binary representations of positive integers.

Decimal representation	Binary representation	
	8-bit binary	32-bit binary
0	0000 0000	0000 0000 0000 0000 0000 0000 0000 0000
1	0000 0001	0000 0000 0000 0000 0000 0000 0000 0001
2	0000 0010	0000 0000 0000 0000 0000 0000 0000 0010
3	0000 0011	0000 0000 0000 0000 0000 0000 0000 0011
4	0000 0100	0000 0000 0000 0000 0000 0000 0000 0100
5	0000 0101	0000 0000 0000 0000 0000 0000 0000 0101
6	0000 0110	0000 0000 0000 0000 0000 0000 0000 0110
7	0000 0111	0000 0000 0000 0000 0000 0000 0000 0111
8	0000 1000	0000 0000 0000 0000 0000 0000 0000 1000
.	.	.
.	.	.
.	.	.
36	0010 0100	0000 0000 0000 0000 0000 0000 0010 0100
37	0010 0101	0000 0000 0000 0000 0000 0000 0010 0101
38	0010 0110	0000 0000 0000 0000 0000 0000 0010 0110
.	.	.
.	.	.
.	.	.

sign	magnitude
X	XXXXXXX

a. The format of an 8-bit sign magnitude number.

bit pattern	decimal value
00000011	3
10000111	−7
11111111	−127
00000000	0
10000000	−0

b. Samples of 8-bit sign magnitude numbers.

FIGURE 4.2
Sign magnitude examples.

bit pattern	decimal value
00000011	3
11111100	−3
00011111	31
11100000	−31
00000000	+0
11111111	−0
00000001	1
11111110	−1

FIGURE 4.3.
One's complement examples.

bit pattern	decimal value
00000011	3
11111100	−4
11111101	−3
00011111	31
11100000	−32
11100001	−31
00000000	0
11111111	−1

FIGURE 4.4
Two's complement examples.

010001	(17)	1101000	(−24)
↓		↓	
101110	all bits inverted	0010111	all bits inverted
+ 1		+ 1	
101111	(−17)	0011000	(24)

FIGURE 4.5
Examples of finding an additive inverse in two's complement representation.

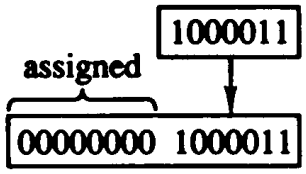
TABLE 4.2

Decimal values for the various integer representations.

	Interpretation					
Bit pattern	**unsigned**	**sign mag**	**two's comp**	**one's comp**	**biased-127**	**biased-128**
0000 0000	0	0	0	0	−127	−128
0000 0001	1	1	1	1	−126	−127
0000 0010	2	2	2	2	−125	−126
0000 0011	3	3	3	3	−124	−125
0000 0100	4	4	4	4	−123	−124
0000 0101	5	5	5	5	−122	−123
0000 0110	6	6	6	6	−121	−122
0000 0000	7	7	7	7	−120	−121
0000 1000	8	8	8	8	−119	−120
0000 1001	9	9	9	9	−118	−119
0000 1010	10	10	10	10	−117	−118
0000 1011	11	11	11	11	−116	−117
0000 1100	12	12	12	12	−115	−116
0000 1101	13	13	13	13	−114	−115
0000 1110	14	14	14	14	−113	−114
0001 1111	15	15	15	15	−112	−113
0001 0000	16	16	16	16	−111	−112
0001 0001	17	17	17	17	−110	−111
0001 0010	18	18	18	18	−109	−110
0001 0011	19	19	19	19	−108	−109
0001 0100	20	20	20	20	−107	−108
0001 0101	21	21	21	21	−106	−107
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
0111 1010	122	122	122	122	−5	−6
0111 1011	123	123	123	123	−4	−5
0111 1100	124	124	124	124	−3	−4
0111 1101	125	125	125	125	−2	−3
0111 1110	126	126	126	126	−1	−2
0111 1111	127	127	127	127	0	−1
1000 0000	128	0	−128	−127	1	0
1000 0001	129	−1	−127	−126	2	1
1000 0010	130	−2	−126	−125	3	2
1000 0011	131	−3	−125	−124	4	3
1000 0100	132	−4	−124	−123	5	4
1000 0101	133	−5	−123	−122	6	5
1000 0110	134	−6	−122	−121	7	6
1000 0111	135	−7	−121	−120	8	7
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
1111 1011	251	−123	−5	−4	124	123
1111 1100	252	−124	−4	−3	125	124
1111 1101	253	−125	−3	−2	126	125
1111 1110	254	−126	−2	−1	127	126
1111 1111	255	−127	−1	0	128	127

TABLE 4.3
Mathematical interpretation of common integer representations. The $(n + 1)$-bit sequence $b_n b_{n-1} b_{n-2} \cdots b_2 b_1 b_0$ represents the integer I.

Format	Range	Value
Unsigned	$0 \le I < 2^{n+1}$	$I = \sum_{i=0}^{n} b_i \cdot 2^i$
Sign magnitude	$-2^n < I < 2^n$	$I = (-1)^{b_n} \sum_{i=0}^{n-1} b_i \cdot 2^i$
One's complement	$-2^n < I < 2^n$	$I = \sum_{i=0}^{n-1} b_i \cdot 2^i - b_n(2^n - 1)$
Two's complement	$-2^n \le I < 2^n$	$I = \sum_{i=0}^{n-1} b_i \cdot 2^i - b_n \cdot 2^n$
Bias-B	$-B \le I < 2^{n+1} - B$	$I = 1 \sum_{i=0}^{n} b_i \cdot 2^i - B$



Sign magnitude integers are easy to expand. The sign bit of the smaller representation is placed into the sign bit (most significant bit) of the larger representation. The magnitude is placed into the least significant portion of the larger representation. The remaining bits are assigned to be zeros.