

Lecture 2

Bits, Bytes, and Words

Bits

- Stands for *binary digit*
- Radix – 2, therefore all representations are in 1's and 0's
- All information in the computer is stored in binary (programs, data (numbers, strings), pictures,)
- Bits are easily represented in hardware
 - o An open switch means a '0', closed switch means '1' (or vice versa)
 - o Switches can be electrical, mechanical, semiconductor, magnetic

Bytes

- A group of 8 bits is called a **byte**
- With 8 bits, there are $2^8 = 256$ different bit patterns are possible
 - o Decimal numbers from 0-255
 - o 256 ASCII symbols
- ASCII is a 7-bit code for alphanumeric and special characters
 - o Each byte can hold 1 ASCII character. The unused highest bit will be zero.
 - o ASCII 'e', for example, is 01100101_2
 - o There for $2^7 = 128$ characters

Rightmost bits

Leftmost bits

*	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	TAB	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SPC	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
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

1 bit	3 leftmost bits	4 rightmost bits
-------	-----------------	------------------

Binary	Hex
ASCII 'a' = 0 110 0001 ₂	= 0x61
ASCII '>' = 0 011 1110 ₂	= 0x3E

First 32 characters are non-printing characters

NUL (null)

SOH (start of heading)

STX (start of text)

ETX (end of text)

EOT (end of transmission) - Not the same as ETB

ENQ (enquiry)

ACK (acknowledge)

BEL (bell) - Caused teletype machines to ring a bell.
Causes a beep

BS (backspace) - Moves the cursor (or print head)
move backwards (left) one space.

TAB (horizontal tab) - Moves the cursor (or print head)
right to the next tab stop

LF (NL) (line feed, new line) - Moves the cursor (or print
head) to a new line.

VT (vertical tab)

FF (form feed) - Advances paper to the top of the next
page (if the output device is a printer).

CR (carriage return) - Moves the cursor all the way to
the left, but does not advance to the next line.

SO (shift out) - Switches output device to alternate
character set.

SI (shift in) - Switches output device back to default
character set.

DLE (data link escape)

DC1 (device control 1)

DC2 (device control 2)

DC3 (device control 3)

DC4 (device control 4)

NAK (negative acknowledge)

SYN (synchronous idle)

ETB (end of transmission block) - Not the same as EOT

CAN (cancel)

EM (end of medium)

SUB (substitute)

ESC (escape)

FS (file separator)

GS (group separator)

RS (record separator)

US (unit separator)

- With the 8th bit of the byte, additional control characters can be represented.
 - o In standard ASCII, this 8th bit is always 0
 - o These additional characters are considered extended ASCII; for these, the 8th bit is 1
 - o Platform dependent (*i.e.*, there is more than one extended ASCII coding/representation)
 - ANSI (Windows/UNIX)
 - OEM (PC/DOS)

ANSI Extended ASCII (Windows)

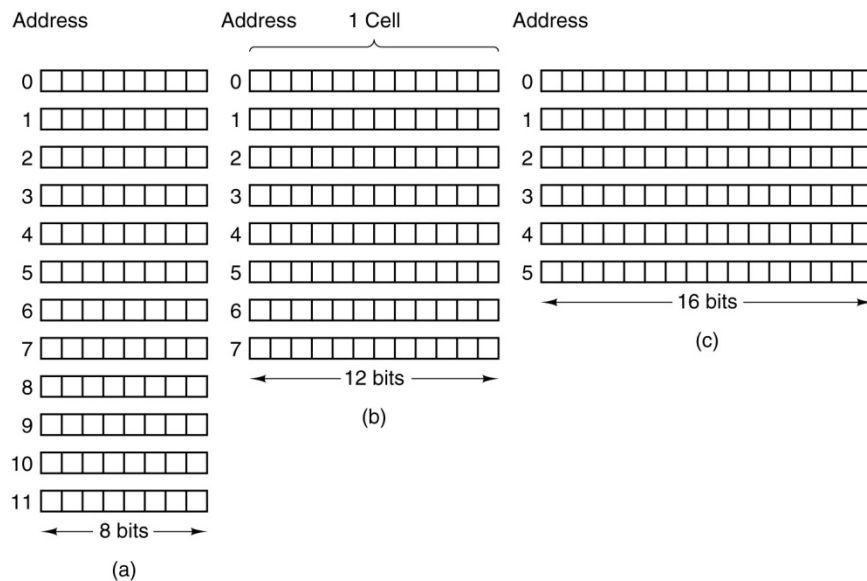
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	□	□	,	f	„	…	†	#	^	‰	Š	<	œ	□	□	□
9	□	\	/	“	”	•	-	-	™	š	>	œ	□	□	Ÿ	
A		ı	ı	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ø	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

OEM Extended ASCII

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	ç	ü	é	â	ä	à	á	ç	ê	ë	è	ï	î	ì	ñ	ø
9	é	æ	œ	ô	ö	ð	û	ù	ÿ	ö	ü	ç	£	¥	℞	f
A	á	í	ó	ú	ñ	ñ	œ	œ	¿	ı	ı	½	¾	ı	«	»
B	⌘	⌘	⌘					n	ı			ı	ı	ı	ı	ı
C	ı	ı	ı	ı	-	ı	ı	ı	ı	ı	ı	ı	ı	=	ı	ı
D	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı
E	α	β	γ	π	Σ	σ	μ	τ	ϑ	θ	Ω	δ	ω	ϖ	€	π
F	≡	±	≥	≤	ı	J	÷	≈	°	·	·	√	n	z	ı	

Words

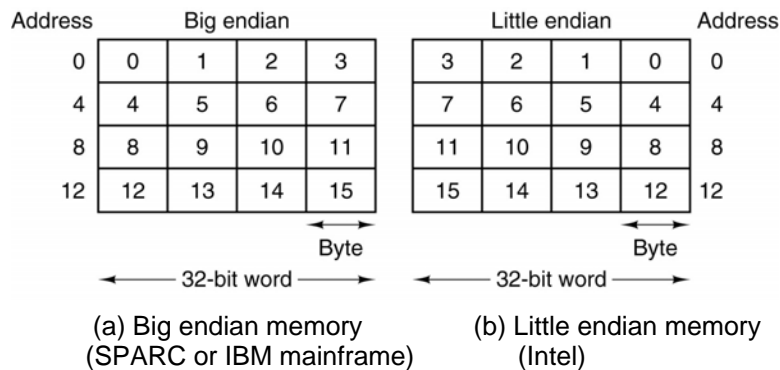
- A group of bytes is called a **word**.
- Word size is dependent on the architecture platform (*e.g.*, 32-bit machine, 64-bit machine)
- A 32-bit word contains 4 bytes; a 64-bit word contains 8 bytes
 - o In a 32-bit machine, everything is 32-bits wide (all registers, buses, numbers, *etc.*)
- The memory of the computer will deliver 1 word at a time (*i.e.*, during a read or write operation)
- The wider the word, the higher the processing power
 - o Why? More data is transmitted from memory to registers, computed upon, *etc.*, per operation.
The processor is handling more information at each step
- Memory is an array of cells.
 - o Each cell is accessible by an address of m bits
 - o Example of 96-bit memory, with cell size of (a) 8-bits, (b) 12-bits, (c) 16 bits



- o If an address is represented by m bits, the maximum number of cells (rows) addressable is 2^m
 - (a) needs a 4-bit address to represent cell locations 0-11; (c) requires 3-bits for the address
 - The number of bits in the address determines the amount of directly accessible memory and is independent of the number of bits per cell.
- o Modern machines have standardized on a 8-bit cell (1 byte), and words of 32 or 64 bits.
 - A 32-bit machine addresses 2^{32} memory cells (bytes)
 - = 4 GB [Gigabytes] = 2^{30} words of 4 bytes each.
 - A 64-bit machine addresses 2^{64} memory cells (bytes)
 - = 16 EB [Exabytes; 1 Exabyte = 2^{60} bytes] = 2^{61} words of 8 bytes each.

Byte Ordering

- Byte ordering within a word can be left-to-right or right-to left
- Big endian numbers from the left to the right (“big” or high side of the binary number)
- Little endian numbers form the right to the left (“little” or small side of the binary number)
- Naming came from Jonathan Swift’s “*Gulliver’s Travels*”, where he satirized politicians who made war over their dispute about whether eggs should be broken on the big end or the little end.



Example: Representation of the decimal number 6

- Numbers are always stored in standard binary form, right-to-left

Big endian

- Binary: 00000000 00000000 00000000 00000110
- Byte: 0 0 0 6 (the 6 is stored in byte 3, 7, 11, or 15)

Little endian

- Binary: 00000000 00000000 00000000 00000110
- Byte: 0 0 0 6 (the 6 stored is in byte 0, 4, 8, or 12)

- No problems if we only need to store numbers.
- But data stored in machines is often a mix of integers, character strings and other data.

Example: Personnel record consisting of a string (employee name), and two integers (age and department number).

- Strings are terminated with 0's to fill out the word.

Jim Smith, age 21, department 260

		Big endian				Little endian					
0	J	I	M			M	I	J	0		
4	S	M	I	T	T	I	M	S	4		
8	H	0	0	0	0	0	0	H	8		
12	0	0	0	21	0	0	0	21	12		
16	0	0	1	4	0	0	1	4	16		

(a) (b)

Why is 260 stored as 0 0 1 4 ?

Binary: 00000000 00000000 00000001 00000100

- Independently, structure is o.k.
- **But**, if data is transferred on network between machines (for example, big endian → little endian), problems arise.

		Transfer from big endian to little endian				Transfer and swap					
3		M	I	J	J	I	M		0		
7	T	I	M	S	S	M	I	T	4		
11	0	0	0	H	H	0	0	0	8		
15	21	0	0	0	0	0	0	21	12		
19	4	1	0	0	0	0	1	4	16		

(c) (d)

- Assume bytes are not reversed additionally. Byte 0 transfers to 0, 1 to 1, *etc.* (Bits inside words are **not** reversed.)
- (c) When little endian tries to print name, it is o.k. But the integers come out wrong: the integers are reversed.
- (d) Obvious solution: reverse all the bytes within a word after copy is made. But then strings print incorrectly: “MIJTIMS”, with the ‘H’ left hanging nowhere.
- No simple solution. Possible: header in front of each address indicating type of data and how many bytes.
- Lack of standardization is a major nuisance.