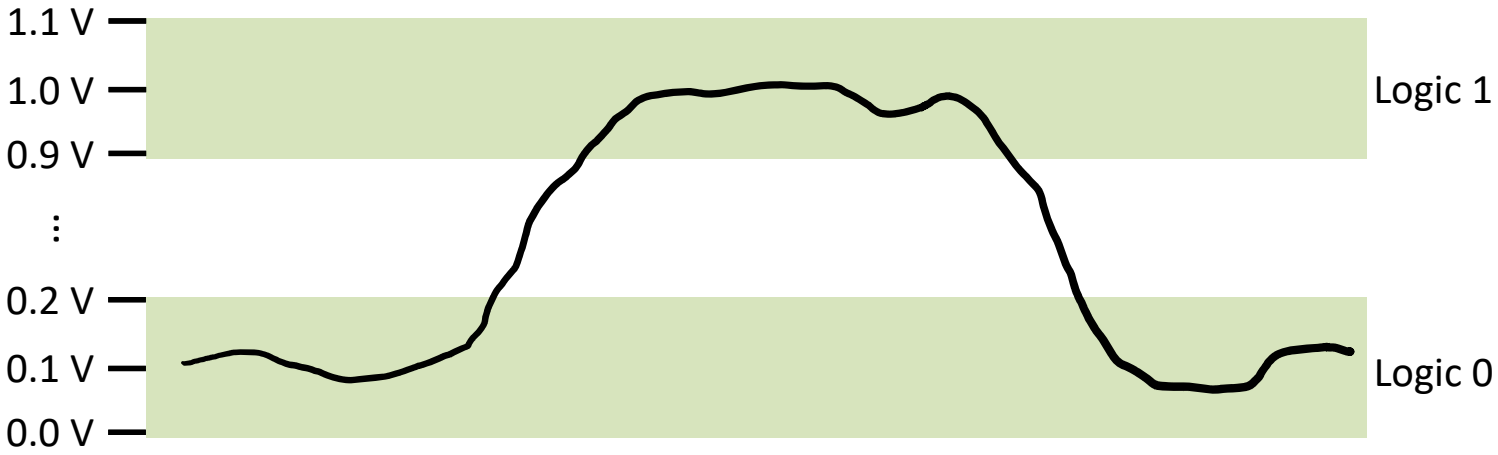
Digital Signals and Binary

- All information stored and processed by the computer is encoded as strings of bits
  - Bits are 0 and 1
- Why?
  - Bits can be stored in and retrieved from bi-stable devices
  - Bits can be reliably transmitted on noisy inaccurate wires



Successive Division

Any number D in base 10 is equivalent to an n-digit number in base R.

$$D = (a_{n-1}a_{n-2} \ldots a_1a_0)_R$$

$$= a_{n-1}R^{n-1} + a_{n-2}R^{n-2} + \ldots\ldots + a_1R^1 + a_0R^0$$

As we divide by R, each subsequent division liberates the next lower-order digit a_i from the base R representation.

$$Q_0 = \frac{D}{R} = a_{n-1}R^{n-2} + a_{n-2}R^{n-3} + \ldots\ldots\ldots + a_1R^0 \qquad \text{remainder } a_0$$

So the first division yields the quotient Q_0, and the remainder a_0 – the value of the lowest order digit.

$$Q_1 = \frac{Q_0}{R} = a_{n-1}R^{n-3} + a_{n-2}R^{n-4} + \ldots\ldots\ldots + a_2R^0 \qquad \text{remainder } a_1$$

.

.

.

$$Q_{n-1} = \frac{Q_{n-2}}{R} = 0 \qquad \text{remainder } a_{n-1}$$

If we repeat the process n times we eventually obtain a quotient of 0 and remainder of a_{n-1}, the highest order digit in base R.

Binary, Octal, and Hexidecimal

	2^1	2^3	2^4
Decimal	**Binary**	**Octal**	**Hexadecimal**
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Signed, Unsigned, Word Size and Range

	Word Size in Bits			
	8	**16**	**32**	**64**
U_{max}	255	65,535	4,294,967,295	18,446,744,073,709,551,615
S_{max}	127	32,767	2,147,483,647	9,223,372,036,854,775,807
S_{min}	-128	-32,768	-2,147,483,648	-9,223,372,036,854,775,808

- C Programming
  - Header file <limits.h> limits the values of various variable types
    - CHAR_BIT 8
    - SCHAR_MIN -128
    - SCHAR_MAX +127
    - UCHAR_MAX 255
    - …
    - ULONG_MAX 18446744073709551615

C Programming – Constants, Integers and Casting

- Constants
  - are signed integers by default
  - add the suffix “U” to make unsigned
    - 12345678U

Casting between signed and unsigned integers does not change the bit-encoding, only its interpretation

- Integers
  - are signed by default
    - int a;
    - unsigned b;
  - explicit casting can be performed between signed an unsigned values
    - a = (int) b;
    - b = (unsigned) a;
  - Implicit casting occurs when performing assignments and function calls
    - a = b;
    - b = a;

```
int foo(unsigned u)
b = foo(a);
```

C Programming – Casting Examples

- Single expressions with a mix of signed and unsigned
  - signed values are implicitly cast to unsigned
  - includes comparison operations ==, >, <, >=, and <=
- Example: K = 16 bits, U_{max} = 65,535, S_{min} = -32,768, S_{max} = +32,767

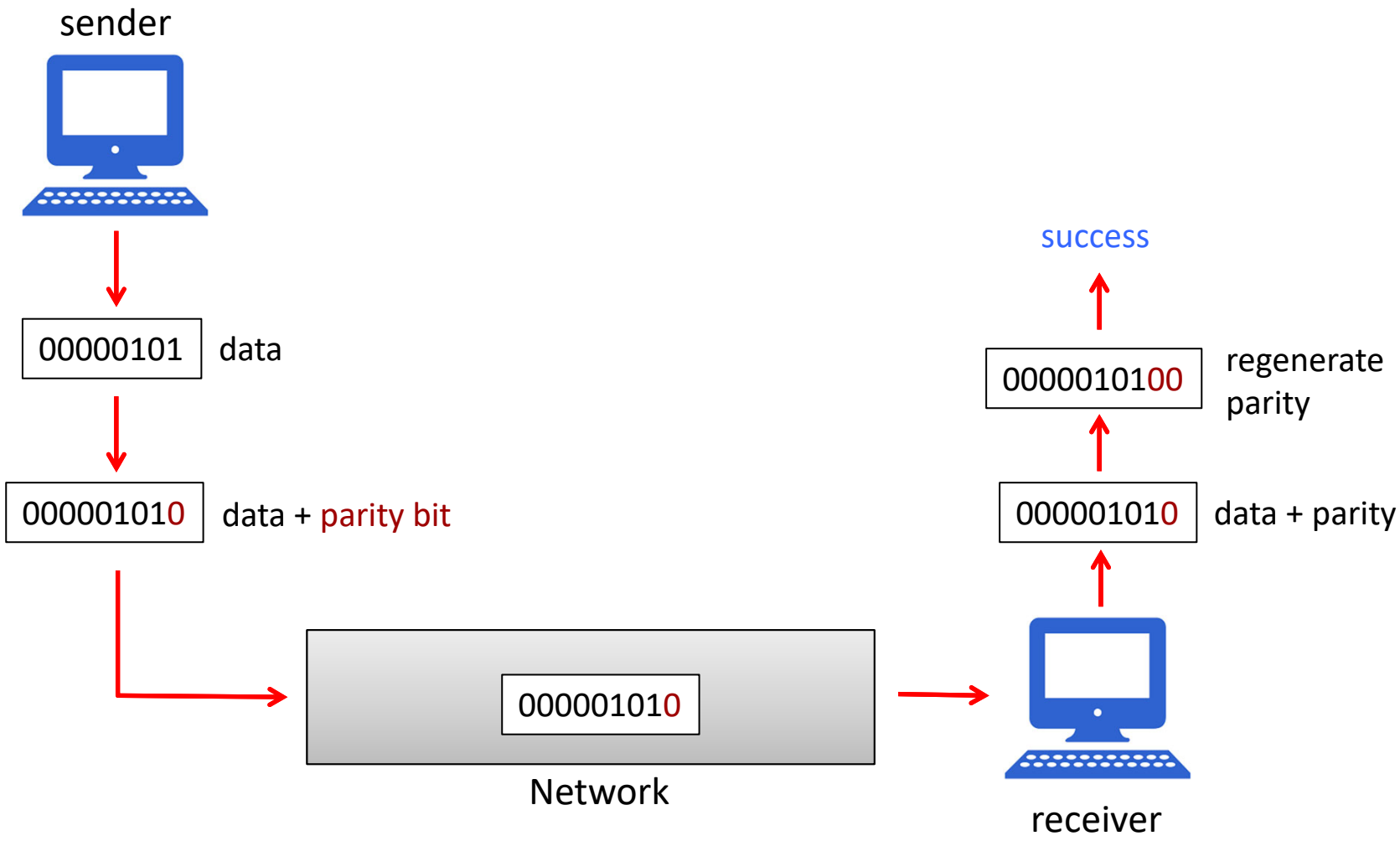
Constant1	Constant2	Relation	Evaluation
0	0U	==	unsigned
-1	0	<	signed
-1	0U	>	unsigned
32767	-32768	>	signed
32767U	-32768	<	unsigned
-1	-2	>	signed
(unsigned) -1	-2	>	unsigned
32767	32768U	<	unsigned
32767	(int) 32768U	>	signed

Parity

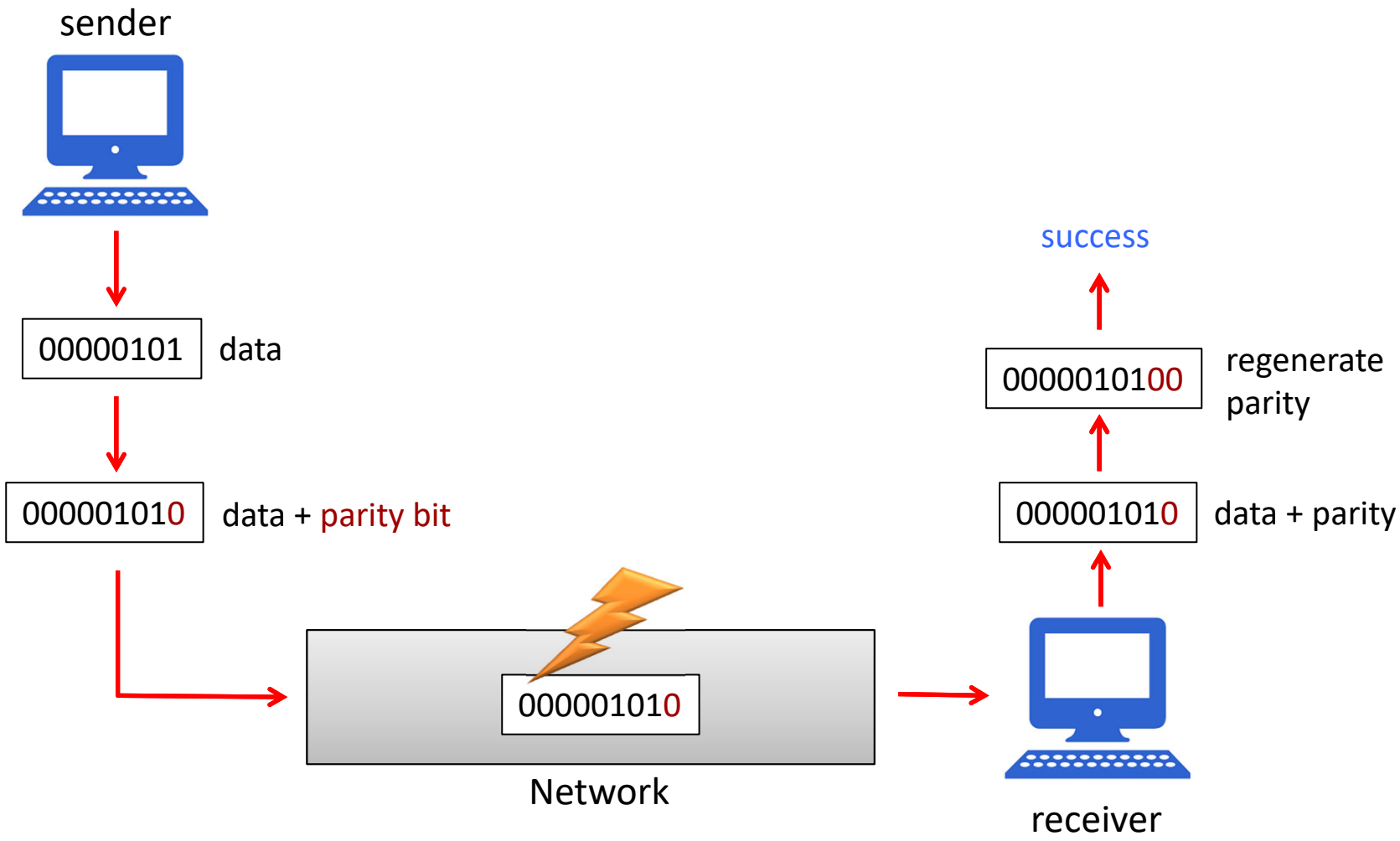
- One of the most common ways to achieve error detection
  - Extra “parity” bit included with data to make total 1’s odd or even

Original Data	Even Parity	Odd Parity
000	0	1
001	1	0
010	1	0
011	0	1
100	1	0
101	0	1
110	0	1
111	1	0

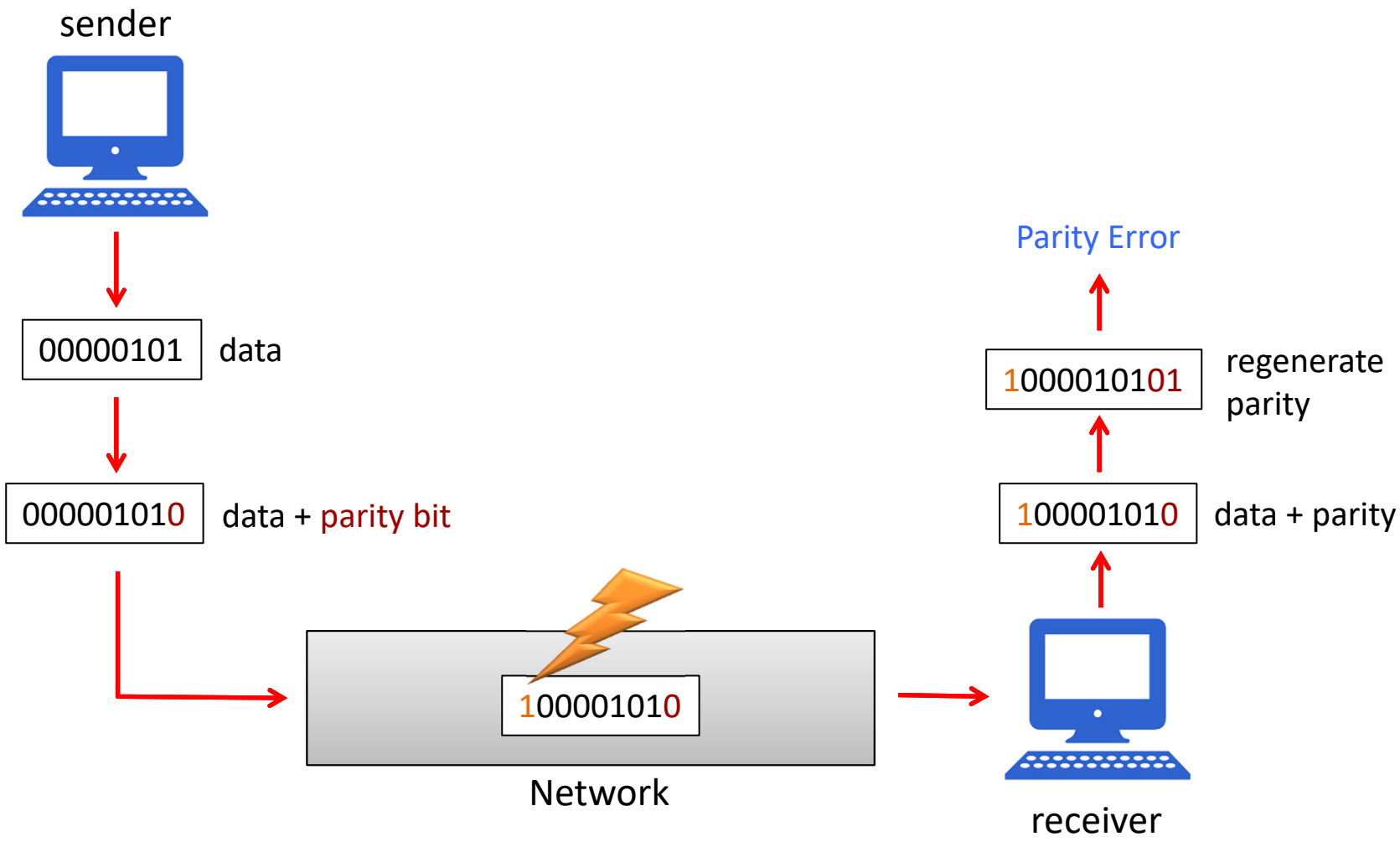
Example of Using (Even) Parity



Example of Using (Even) Parity



Example of Using (Even) Parity



American Standard Code for Information Interchange (ASCII)

Bits 4321	Bit Positions 765							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SPACE	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	‘	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	0	_	o	DEL

- 7-bit code

American Standard Code for Information Interchange (ASCII)

Bits 4321	Bit Positions 765							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SPACE	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	‘	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	0	_	o	DEL

- 7-bit code
  - 33 control

American Standard Code for Information Interchange (ASCII)

Bits 4321	Bit Positions 765							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SPACE	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	'	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

- 7-bit code
  - 33 control
  - 95 graphic

American Standard Code for Information Interchange (ASCII)

Bits 4321	Bit Positions 765							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SPACE	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	‘	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

- 7-bit code
  - 33 control
  - 95 graphic

- A = 1000001_2 = 65
- a = 1100001_2 = 97

American Standard Code for Information Interchange (ASCII)

Bits 4321	Bit Positions 765							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SPACE	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	‘	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	0	_	o	DEL

- Upper case letters

$A = 65_{10}$
$B = 66_{10}$
$C = 67_{10}$
.
.
$Z = 90_{10}$

American Standard Code for Information Interchange (ASCII)

Bits 4321	Bit Positions 765							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SPACE	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	‘	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	0	_	o	DEL

- Upper case letters

  $A = 65_{10}$
  $B = 66_{10}$
  .
  .
  .

- Lower case letters

  $a = 97_{10}$
  $b = 98_{10}$
  .
  .
  .

American Standard Code for Information Interchange (ASCII)

Bits 4321	Bit Positions 765							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SPACE	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	‘	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	0	_	o	DEL

- Conversion from character to number

Representing Strings

- A string in C is represented by an array of bytes
  - Array elements are 1-byte ASCII codes for each character
  - A null (0) byte marks the end of the array

```
char course[] = "CIS 2030";
```

C	I	S		2	0	3	0	\0
67	73	83	32	50	48	51	48	0

Summary

- Hardware directly uses binary number system
- Software uses decimal, hexadecimal, octal
  - Conversions between different bases for integer and real numbers
- Binary arithmetic
  - Rules for addition and subtraction
  - Addition fast, subtraction slow
- Signed numbers
  - Older representations include sign-magnitude, 1's-complement
  - Today's machines typically use 2's-complement
    - 2's-complement conversion
    - Addition rules
    - Fast (pencil and paper) method for computing 2's-complement
- Ranges
  - Unsigned and Signed numbers
  - Depends on word size and assumed representation

Summary

- Overflow
  - Rules for detecting signed overflow
  - Rules for detecting unsigned overflow
- Non-numeric (binary) codes
  - Parity (error detection)
  - ASCII (characters)