Digital Logic Design

Hajar Falahati

Department of Computer Engineering
IRAN University of Science and Technology

hfalahati@iust.ac.ir

Signed Numbers Representation

n-1	n-2	n-3	.....	2	1	0	.	-1	-2	....	-m

Signed bit

Magnitude representation

- Let $N = (a_{n-1} \ldots a_0)_2$
  - If $N \geq 0$, it is represented by $(0a_{n-1} \ldots a_0)_2$
  - If $N < 0$, it is represented by $[0a_{n-1} \ldots a_0]_2$
    - $[N]_2 = 2^n - (N)_2$

- Diminished radix complement $[N]_{r-1}$
  - $[N]_{r-1} = r^n - (N)_r - 1$

- One's complement (r = 2):
  - $[N]_{2-1} = 2^n - (N)_2 - 1$

Overflow Condition

- Presenting numbers using two's complement number system:
  - Addition: Add two numbers.
  - Subtraction: Add two's complement of the subtrahend to the minuend.
    - **Carry bit is discarded**
  - Overflow is detected as the Table.

Case	Carry	Sign Bit	Condition	Overflow ?
B + C	0 0	0 1	$B + C \leq 2^{n-1} - 1$ $B + C > 2^{n-1} - 1$	No Yes
B - C	1 0	0 1	$B \leq C$ $B > C$	No No
-B - C	1 1	1 0	$-(B + C) \geq -2^{n-1}$ $-(B + C) < -2^{n-1}$	No Yes

Outline

- Digital codes

Computer Codes

Computer Codes

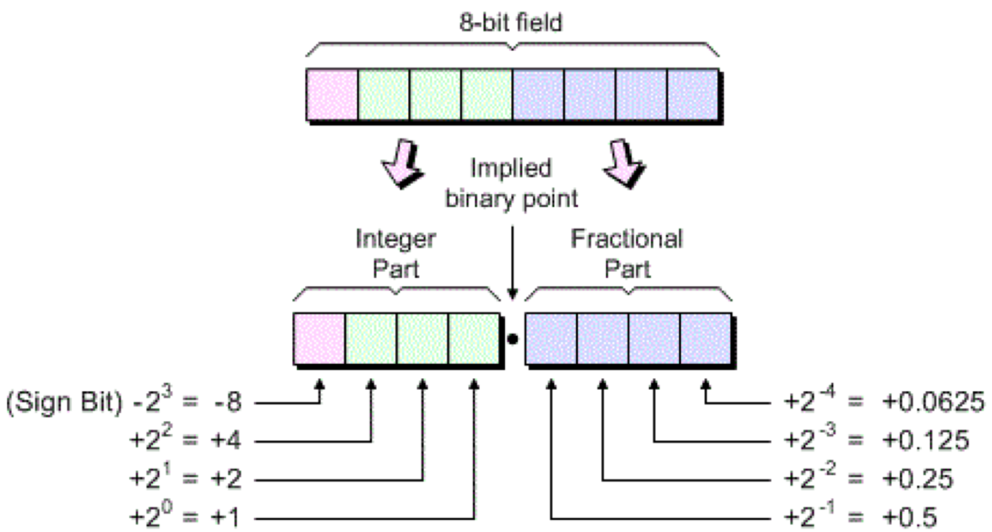
- Code
  - Systematic use of a given set of symbols
  - => for representing information
- Numeric Codes
  - To represent numbers
    - Fixed-point numbers
    - Floating-point number

Traffic lights, 3-bit code

Color	3-bit code
Red	000
Orange	001
Yellow	010
Green	011
Blue	100
Indigo	101
Violet	110

Computer Codes (cont'd)

- Fixed-point Numbers
  - Used for signed integers or integer fractions
  - Sign magnitude, two's complement, or one's complement systems are used.
  - Integer: (Sign bit) + (Magnitude) + (Implied radix point)
  - Fraction: (Sign bit) + (Implied radix point) + (Magnitude)



Computer Codes (cont'd)

- Excess or Biased Representation
  - **Excess-K** representation of a code C:
    - Add **K** to each code word C.
  - Used for exponents of floating-point numbers
- Example
  - Excess-8 representation of 4-bit two's complement code

TABLE 1.8 EXCESS-8 CODE

Decimal	Two's Complement	Excess-8
+7	0111	1111
+6	0110	1110
+5	0101	1101
+4	0100	1100
+3	0011	1011
+2	0010	1010
+1	0001	1001
0	0000	1000
-1	1111	0111
-2	1110	0110
-3	1101	0101
-4	1100	0100
-5	1011	0011
-6	1010	0010
-7	1001	0001
-8	1000	0000

Characters and Other Codes

- To represent information as strings of alpha-numeric characters
  - Binary coded decimal (BCD)
  - ASCII
  - Gray code
  - Error detection codes
  - Error correction codes
  - Hamming code

Binary Coded Decimal (BCD)

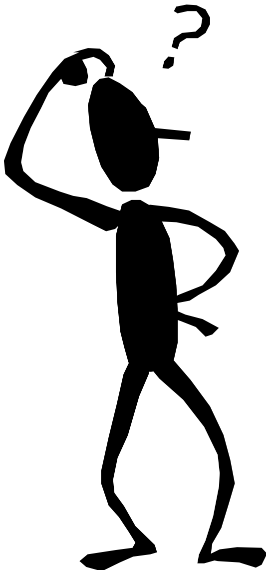
- Used to represent the decimal digits 0 - 9
- 4 bits are used.
- Each bit position has a weight associated with it
  - Weighted code
  - 8, 4, 2, and 1 from MSB to LSB (called 8-4-2-1 code)
- BCD Codes:
  - 0: 0000
  - 1: 0001
  - 2: 0010
  - 3: 0011
  - 4: 0100
  - 5: 0101
  - 6: 0110
  - 7: 0111
  - 8: 1000
  - 9: 1001

BCD in Applications!

- Used to encode numbers for output to numerical displays
- Used in processors that perform decimal arithmetic
- Example
  - $(9750)_{10} = (1001, 0111, 0101, 0000)_{BCD}$
  - $(9750)_{10} = (1001011101010000)_{BCD}$

BCD: Sample

- $(9)_{10} = (?)_{BCD}$
- $(29)_{10} = ()_{BCD}$
- $(129)_{10} = ()_{BCD}$
- $(1029)_{10} = ()_{BCD}$

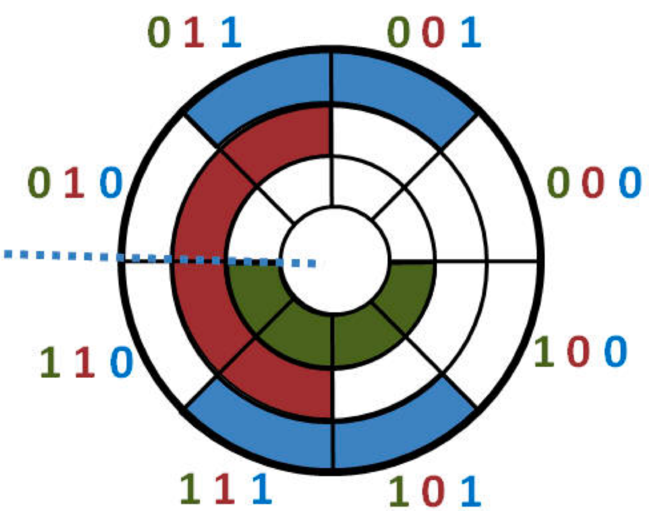


BCD: Sample (cont'd)

- $(9)_{10} = (1001)_{BCD}$
- $(29)_{10} = (0010\ 1001)_{BCD} = (00101001)_{BCD}$
- $(129)_{10} = (0001\ 0010\ 1001)_{BCD} = (000100101001)_{BCD}$
- $(1029)_{10} = (0001\ 0000\ 0010\ 1001)_{BCD} = (0001000000101001)_{BCD}$

Gray Code

- Gray code:
  - A cyclic code with the property
  - Cyclic code: a circular shifting of a code word produces another code word
  - Property: two consecutive code words differ in only 1 bit
    - Distance between the two code words is 1



$Q_2Q_1Q_0$
000
001
011
010
110
111
101
100

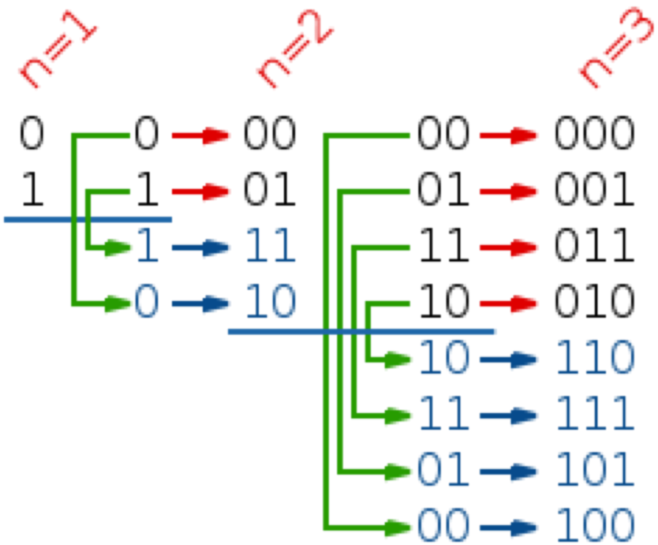
Gray Code

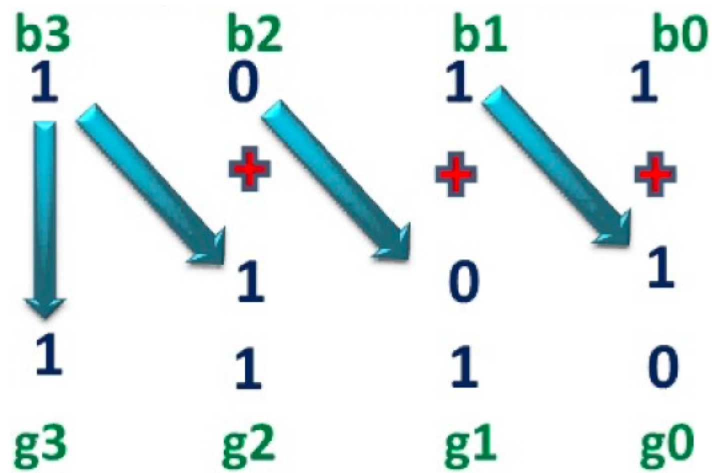
- Bit i in gray code
  - Compare bit i and i+1 in binary code
  - Similar ==> Gray(i) = 0
  - Otherwise ==> Gray(i) =1

Digit	Binary	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100

Digit	Binary	Gray Code
0	1000	1100
1	1001	1101
2	1010	1111
3	1011	1110
4	1100	1010
5	1101	1011
6	1110	1001
7	1111	1000

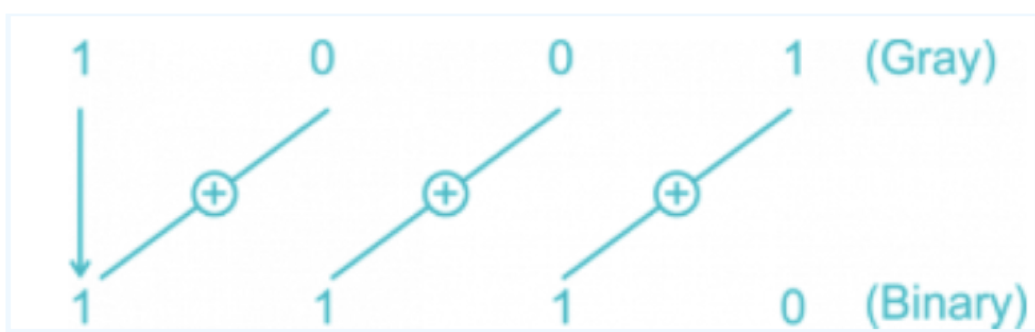
Gray Code Conversion





$A \oplus B$

A	B	$Y = AB' + A'B$
0	0	0
0	1	1
1	0	1
1	1	0



Iran University of Science & Technology
IUST

ASCII

- American Standard Code for Information Interchange
  - Most widely used character code
  - 7-bit ASCII code
  - Eighth bit is often used for error detection (parity bit)

$c_6c_5c_4$

$c_3c_2c_1c_0$

	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS IS4	,	<	L	\	l	|
D	CR	GS IS3	-	=	M	]	m	}
E	SO	RS IS2	.	>	N	^	n	~
F	SI	US IS1	/	?	O	_	o	DEL

ASCII (cont'd)

- Control Character
  - First 32 characters of ASCII table are used for control
    - 00 – 1F
  - One control character appears at end of ASCII table
    - DEL

$c_6c_5c_4$

$c_3c_2c_1c_0$

	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS IS4	,	<	L	\	l	|
D	CR	GS IS3	-	=	M	]	m	}
E	SO	RS IS2	.	>	N	^	n	~
F	SI	US IS1	/	?	O	_	o	DEL

ASCII (cont'd)

- Example
  - ASCII code representation of the word ***Digital***

Digital	
Character	Binary Code
D	
i	
g	
i	
t	
a	
l	

$c_6c_5c_4$

$c_3c_2c_1c_0$

	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS IS4	,	<	L	\	l	|
D	CR	GS IS3	-	=	M	]	m	}
E	SO	RS IS2	.	>	N	^	n	~
F	SI	US IS1	/	?	O	_	o	DEL

Iran University of Science & Technology
IUST

ASCII (cont'd)

- Example
  - ASCII code representation of the word ***Digital***

Digital	
Character	Binary Code
D	100 0100
i	110 1001
g	110 0111
i	110 1001
t	111 0100
a	110 0001
l	110 1100

$c_3c_2c_1c_0$

$c_6c_5c_4$	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS IS4	,	<	L	\	l	|
D	CR	GS IS3	-	=	M	]	m	}
E	SO	RS IS2	.	>	N	^	n	~
F	SI	US IS1	/	?	O	_	o	DEL

Character Codes

- Standard ASCII
  - 7-bit character codes (0 – 127)
- Extended ASCII
  - 8-bit character codes (0 – 255)
- Unicode
  - 16-bit character codes (0 – 65,535)
- Unicode standard represents a universal character set
  - Defines codes for characters used in all major languages
  - Used in Windows-XP: each character is encoded as 16 bits
- UTF-8: variable-length encoding used in HTML
  - Encodes all Unicode characters
  - Uses 1 byte for ASCII, but multiple bytes for other characters

Codes

Decimal digits	BCD (8421)	Excess-3	84-2-1	2421	Bi-quinary (5043210)
0	0000	0011	0000	0000	0100001
1	0001	0100	0111	0001	0100010
2	0010	0101	0110	0010	0100100
3	0011	0110	0101	0011	0101000
4	0100	0111	0100	0100	0110000
5	0101	1000	1011	1011	1000001
6	0110	1001	1010	1100	1000010
7	0111	1010	1001	1101	1000100
8	1000	1011	1000	1110	1001000
9	1001	1100	1111	1111	1010000

Floating Point

Floating Point Numbers

- $N = M \times r^E$ (1.13)
  - M (mantissa or significand)
    - Significant digits of N
  - E (exponent or characteristic)
    - An integer exponent
- $N = \pm (a_{n-1} \ldots a_0 . a_{-1} \ldots a_{-m})_r$
  - Presented by: $N = \pm (.a_{n-1} \ldots a_{-m})_r \times r^n$

Floating Point Numbers

- $N = M \times r^E$ (1.13)
  - M (mantissa or significand)
    - Significant digits of N
  - E (exponent or characteristic)
    - An integer exponent

- M
  - Usually represented in sign magnitude
  - $M = (S_M.a_{n-1} \ldots a_{-m})_{rsm}$ (1.14)
  - $(.a_{n-1} \ldots a_{-m})_r$ represents the magnitude
  - $M = (-1)^{S_M} \times (.a_{n-1} \ldots a_{-m})_r$ (0: positive, 1: negative) (1.15)

Floating Point Numbers

- $N = M \times r^E$ (1.13)
  - M (mantissa or significand)
    - Significant digits of N
  - E (exponent or characteristic)
    - An integer exponent
- $M = (S_M.a_{n-1} \ldots a_{-m})_{rsm} = (-1)^{S_M} \times (.a_{n-1} \ldots a_{-m})_r$ (1.14), (1.15)
- E
  - $-2^{e-1} \leq E \leq 2^{e-1}$
  - Usually coded in **excess-K** two's complement.
  - **K** : called bias
    - Usually selected to be 2^{e-1} (**e is the number of bits**).
  - Biased E is:
    - $0 \leq E + 2^{e-1} \leq 2^e$
  - Excess-K form of E is written as (1.16)
    - $E = (b_{e-1}, b_{e-2} \ldots b_0)_{excess\text{-}K}$
    - b_{e-1} is the sign bit.

Floating Point Numbers

- $N = M \times r^E$ (1.13)
  - M (mantissa or significand)
    - Significant digits of N
  - E (exponent or characteristic)
    - An integer exponent
- $M = (S_M.a_{n-1} \dots a_{-m})_{rsm} = (-1)^{S_M} \times (.a_{n-1} \dots a_{-m})_r$ (1.14), (1.15)
- $E = (b_{e-1}, b_{e-2} \dots b_0)_{excess\text{-}K}$ (1.16)
- Combining Eqs. (1.14) and (1.16), we have
  - $N = (S_M b_{e-1} b_{e-2} \dots b_0 a_{n-1} \dots a_{-m})_r$ (1.17)
  - $N = (-1)^{S_M} \times (.a_{n-1} \dots a_{-m})_r \times r^{(b_{e-1} b_{e-2} \dots b_0) - 2^{e-1}}$ (1.18)
- Number 0 is represented by an all-zero word.

Floating Point Numbers (cont'd)

- Multiple representations of a given number:
  - $N = M \times r^E$ (1.19)
  - $= (M \div r) \times r^{E+1}$ (1.20)
  - $= (M \times r) \times r^{E-1}$ (1.21)

- Example: $M = +(1101.0101)_2$
  - $M = +(1101.0101)_2$

    $= (0.11010101)_2 \times 2^4$

    $= (0.011010101)_2 \times 2^5$

    $= (0.0011010101)_2 \times 2^6$

    …

Floating Point Numbers (cont’d)

- **Normalization**
  - A unique representation
  - Mantissa has a nonzero value in its MSD position.
  - Example:
    - $M = +(1101.0101)_2$
    - Normal representation: $(0.11010101)_2 \times 2^4$

Floating Point Numbers (cont'd)

- Floating-point number formats
  - Typical single-precision format
  - $N = (S_M b_{e-1} b_{e-2} \ldots b_0 a_{n-1} \ldots a_{-m})_r$ $\quad (-1)^{S_M} \times (.a_{n-1} \ldots a_{-m})_r \times r^{(b_{e-1} b_{e-2} \ldots b_0) - 2^{e-1}}$

S_M	Exponent E	Mantissa M

Sign of mantissa

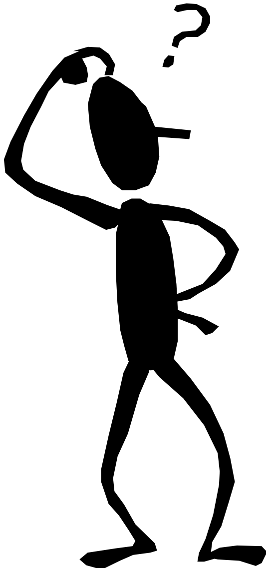
  - Typical extended-precision format

S_M	Exponent E	Mantissa M (most significant part)

Mantissa M (least significant part)

Floating Point Numbers (cont'd)

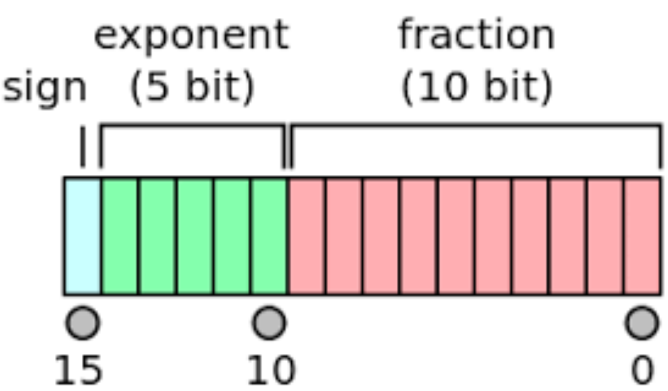
- $N = (101101.101)_2$
- Assumption
  - Normalized sign magnitude fraction is used for M
  - Excess-16 two's complement is used for E.



Floating Point Numbers (cont'd)

- $N = (101101.101)_2$
  - $N = (101101.101)_2 = (0.101101101)_2 \times 2^6$
  - $M = +(0.1011011010)_2 = (0.1011011010)_{2sm}$
  - $E = +(6)_{10} = +(0110)_2 = (00110)_{2cns}$
    - Add the bias $16 = (10000)_2$ to E
    - $E = 00110 + 10000 = 10110$
    - $E = (1, 0110)_{excess\text{-}16}$
  - Combining M and E

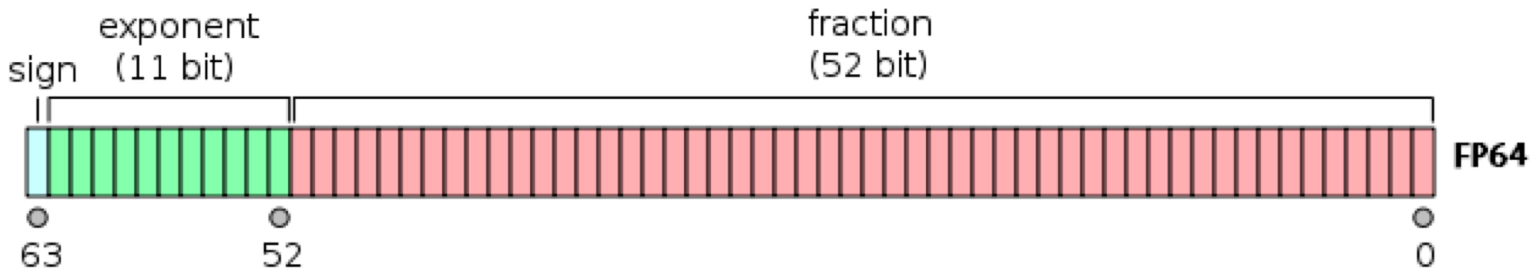
  $N = (0, 10110, 1011011010)_{fp}$

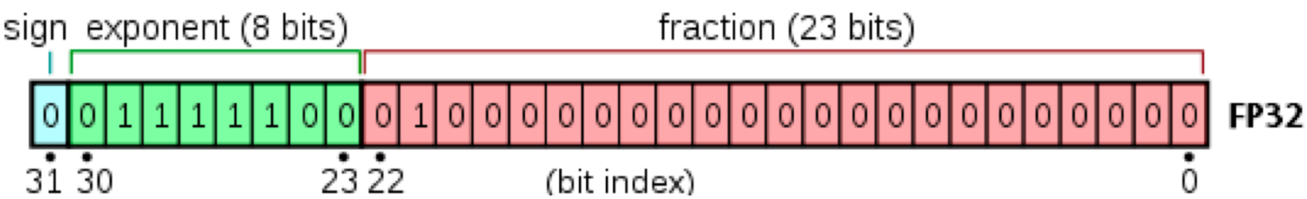


Floating Point Numbers (cont’d)

System/ Format	Total bits	Significand bits	Exponent bits	Exponent bias	Mantissa coding
IEEE Std. 754-1985:					Sign/Mag: (radix 2):
Single Precision	32	23 (+1)	8	127	$1 \leq \lvert M \rvert < 2$
Double Precision	64	52 (+1)	11	1023	$1 \leq \lvert M \rvert < 2$
IBM System/360:					Sign/Mag (radix 16):
Single Precision	32	24	7	64	$1/16 \leq \lvert M \rvert < 1$
Double Precision	64	56	7	64	$1/16 \leq \lvert M \rvert < 1$
DEC VAX 11/780:					Sign/Mag (radix 2):
F Format	32	23 (+1)	8	128	$1/2 \leq \lvert M \rvert < 1$
D Format	64	55 (+1)	8	128	$1/2 \leq \lvert M \rvert < 1$
G Format	64	52 (+1)	11	1024	$1/2 \leq \lvert M \rvert < 1$
CDC Cyber 70:	60	48	11	1024	1’s Complement (radix 2) $1 \leq \lvert M \rvert < 2^{48}$

IEEE 754





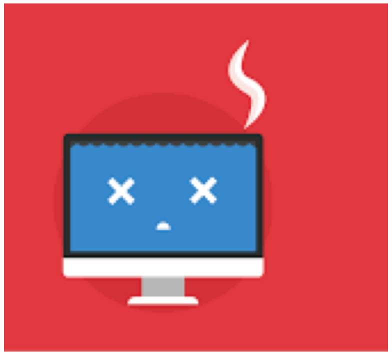
IEEE 754 Special Number Representation

Single Precision		Double Precision		Number Represented
Exponent	Significand	Exponent	Significand	
0	0	0	0	0
0	nonzero	0	nonzero	Denormalized number
1 to 254	anything	1 to 2046	anything	Floating Point Number
255	0	2047	0	Infinity
255	nonzero	2047	nonzero	NaN (Not A Number)

Error Correction/Detection Codes

Error

- Error
  - An incorrect value in one or more bits
- Single error
  - An incorrect value in only one bit
- Multiple error
  - One or more bits are incorrect
- Error sources
  - Hardware failures
  - External interference (noise)
  - Other unwanted events.



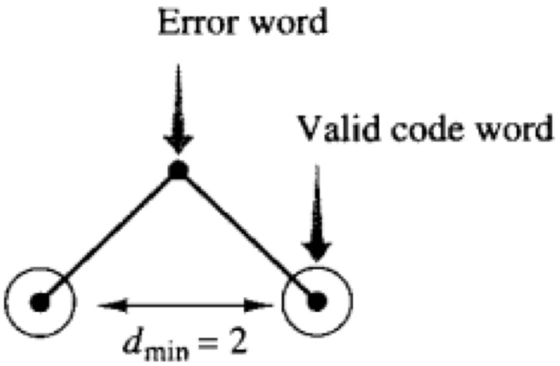
Error Detection/Correction Codes

- Error detection/correction code
  - Encode information in such a way that a particular class of errors can be detected and/or corrected.
- Let I and J be n-bit binary information words
  - w(I): number of 1's in I (weight)
  - d(I, J): number of bit positions in which I and J differ (distance)
- Example: I = (01101100) and J = (11000100)
  - w(I) = 4 and w(J) = 3
  - d(I, J) = 3

0	1	1	0	1	1	0	0
1	1	0	0	0	1	0	0
↑		↑		↑			

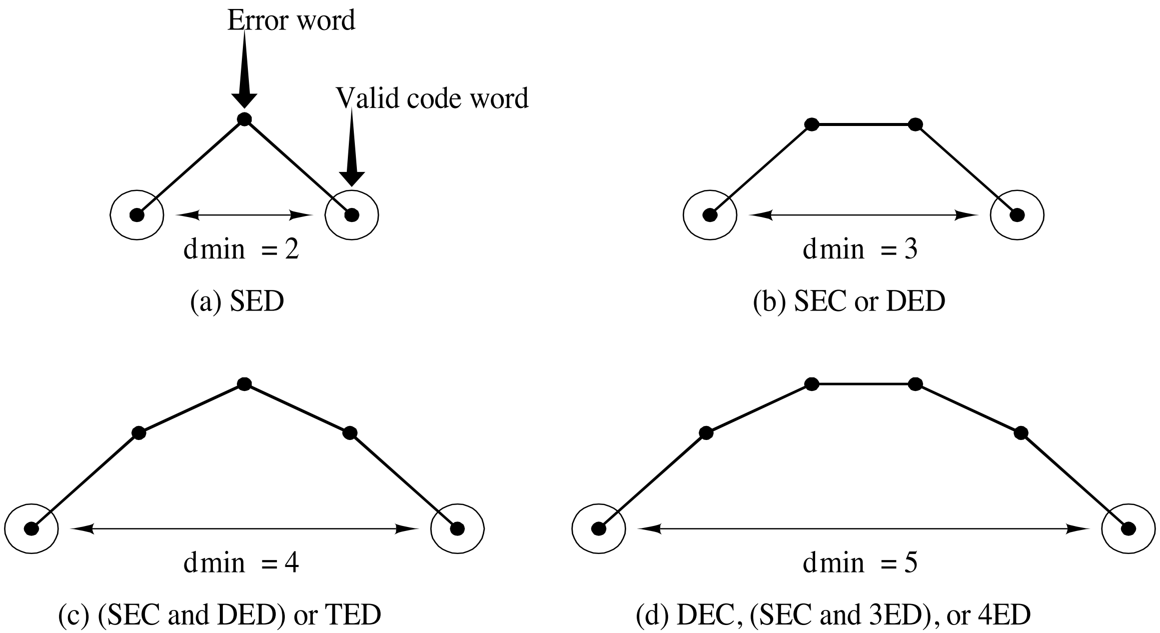
Error Detection/Correction Codes (cont'd)

- Minimum distance, d_{min}, of a code C
  - For any two code words I and J in C, $d(I, J) \geq d_{min}$
  - Determines the properties of error correction and detection of a code



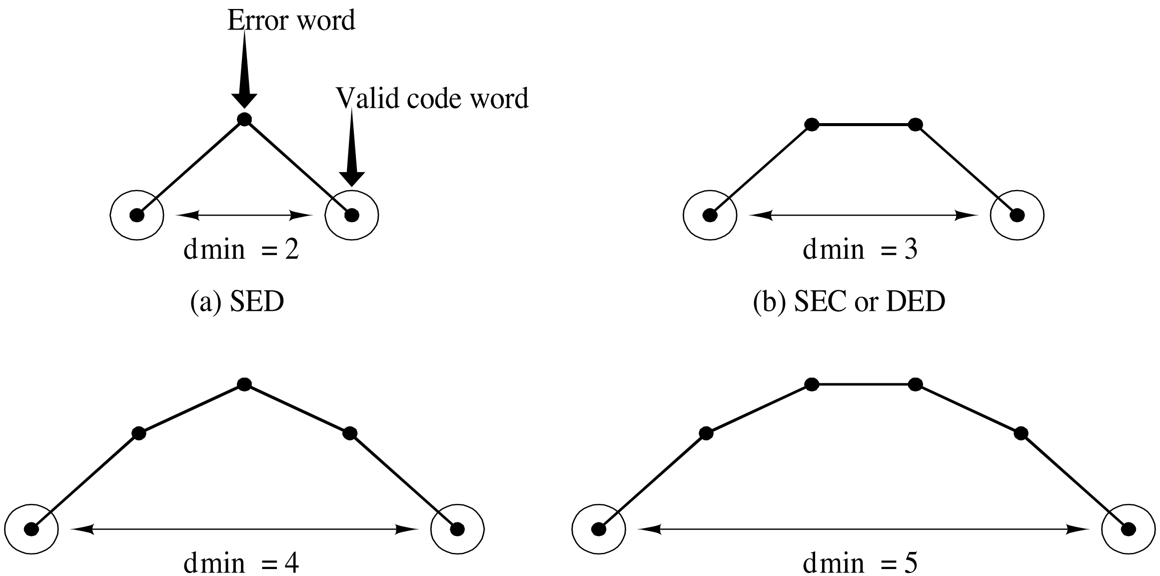
Error Detection/Correction Codes (cont'd)

- Relationship between d_{min} of code words and detection/correction ability
  - A code provides *t* error correction
  - Plus *s* error detection of *s* additional errors if and only if.
    - $2t + s + 1 \leq d_{min}$ (1.25)



Error Detection/Correction Codes (cont'd)

- Example
  - Single-error detection (SED): $s = 1$, $t = 0$, $d_{min} = 2$.
  - Single-error correction (SEC): $s = 0$, $t = 1$, $d_{min} = 3$.
  - Single-error correction and double-error detection (SEC/DED)
    - $s = t = 1$, $d_{min} = 4$.



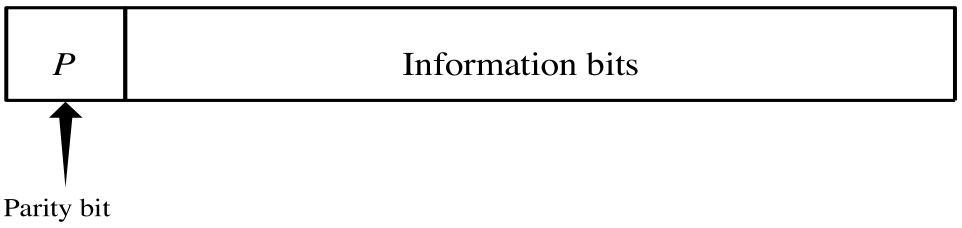
(a) SED (b) SEC or DED (c) (SEC and DED) or TED (d) DEC, (SEC and 3ED), or 4ED

Error Detection/Correction Codes (cont’d)

- Simple Parity code
- Two-out-of-Five code
- Hamming code

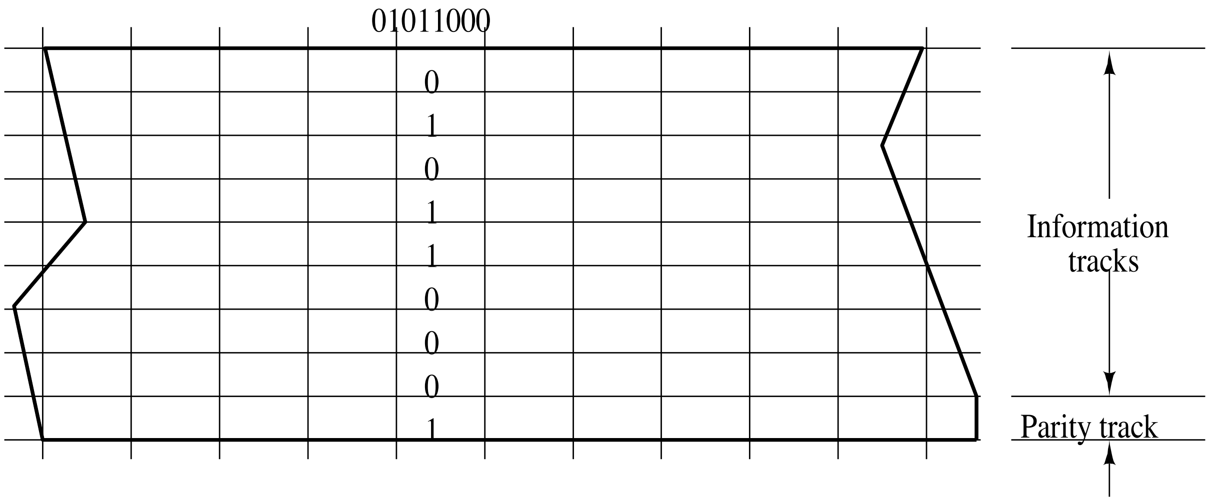
Parity

- Simple Parity Code
  - Concatenate (|) a parity bit, *P*, to each code word of *C*.
  - Odd-parity code: w(P|C) is odd.
  - Even-parity code: w(P|C) is even.



Parity (cont'd)

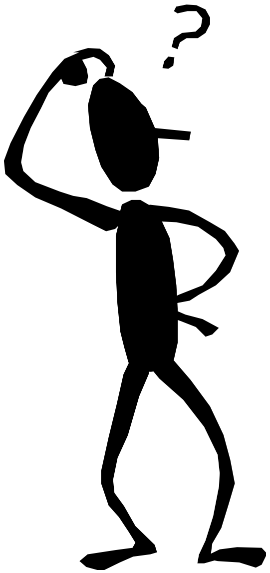
- Sample
  - Parity coding on magnetic tape:



Parity (cont'd)

- Produce odd parity code for ASCII code of character 0,X,=, BEL

Character	ASCII Code
0	0110000
X	1011000
=	0111100
BEL	0000111



Parity (cont'd)

- Produce odd parity code for ASCII code of character 0,X,=, BEL

Character	ASCII Code	Odd-parity Code
0	0110000	10110000
X	1011000	01011000
=	0111100	1111100
BEL	0000111	00000111

Parity (cont’d)

- Error detection
  - Check whether a code word has the correct parity
- Does parity have error detection ability?

Parity (cont'd)

- Error detection
  - Check whether a code word has the correct parity
- Does parity have error detection ability?
  - Yes,
  - Single-error detection code (d_{min} = 2).

Two-out-of-Five Code

- Two-out-of-Five code
  - Each code word has exactly two 1’s and three 0’s.
- Error detection
  - Counting the number of ones
  - If number of ones is not exactly equal to 2
    - ➔ error
  - Detects single and multiple errors in adjacent bits.

Digit	Two-out-of-Five Code
0	00011
1	00101
2	01001
3	10001
4	00110
5	01010
6	10010
7	01100
8	10100
9	11000

Hamming Codes

- Richard Hamming, 1950
- An extension of simple parity codes with multiple parity or check bits
- Each check bit
  - Is defined over (or covers) a subset of the information bits.
- Subsets overlap
  - Each information bit is in at least two subsets.
- Error detection/correction ability
  - Number of check bits
  - How check bits are defined
  - d_{min} : weight of the minimum-weight nonzero code word.

Hamming Codes (cont'd)

- Hamming Code 1 (Table 1.14)
  - A code word consists of 4 information bits and 3 check bits:

    $c = (i_3\ i_2\ i_1\ i_0\ c_2\ c_1\ c_0)$
  - Each check bit covers:

$$c_2: i_3, i_2, i_1$$

$$c_1: i_3, i_2, i_0$$

$$c_0: i_3, i_1, i_0$$

  - $d_{min} = 3$
  - Single error correction code.

Hamming Codes (cont'd)

- m =8
- r?

$$(m + r + 1) \le 2^r$$

Bit position:	1	2	3	4	5	6	7	8	9	10	11	12
	P_1	P_2	1	P_4	1	0	0	P_8	0	1	0	0

$P_1 = \text{XOR of bits } (3, 5, 7, 9, 11) = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 0$

$P_2 = \text{XOR of bits } (3, 5, 7, 10, 11) = 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 = 0$

$P_4 = \text{XOR of bits } (5, 6, 7, 12) = 1 \oplus 0 \oplus 0 \oplus 0 = 1$

$P_8 = \text{XOR of bits } (9, 10, 11, 12) = 0 \oplus 1 \oplus 0 \oplus 0 = 1$

$C_1 = \text{XOR of bits } (1, 3, 5, 7, 9, 11)$

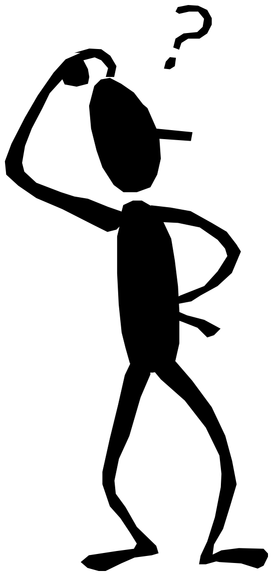
$C_2 = \text{XOR of bits } (2, 3, 6, 7, 10, 11)$

$C_4 = \text{XOR of bits } (4, 5, 6, 7, 12)$

$C_8 = \text{XOR of bits } (8, 9, 10, 11, 12)$

Sample 1

- Send this data in haming code
- 10101001



Sample 1 (cont'd)

- Find r?

$$10101001 \qquad (m + r + 1) \leq 2^r$$

$$r = 1 \quad \therefore \quad (8 + 1 + 1) \leq 2^1 \qquad 10 \leq 2$$

$$r = 2 \quad \therefore \quad (8 + 2 + 1) \leq 2^2 \qquad 11 \leq 4$$

$$r = 3 \quad \therefore \quad (8 + 3 + 1) \leq 2^3 \qquad 12 \leq 8$$

$$r = 4 \quad \therefore \quad (8 + 4 + 1) \leq 2^4 \qquad 13 \leq 16$$

Sample 1 (cont'd)

- Determine P_i?

1	2	3	4	5	6	7	8	9	10	11	12
?	?	1	?	0	1	0	?	1	0	0	1

$3=1+2 = 2^0+2^1$

$5=1+4 = 2^0+2^2$

$6=2+4 = 2^1+2^2$

$7=1+2+4 = 2^0+2^1+2^2$

$9=1+8 = 2^0+2^3$

$10=2+8 = 2^1+2^3$

$11=1+2+8 = 2^0+2^1+2^3$

$12=4+8 = 2^2+2^3$

Sample 1 (cont'd)

- Determine P_i?

1	2	3	4	5	6	7	8	9	10	11	12
?	?	1	?	0	1	0	?	1	0	0	1

- P_1 = xor (3,5,7,9,11) = xor (1,0,0,1,0) = 0
- P_2 = xor (3,6,7,10,11) = xor (1,1,0,0,0) = 0
- P_4 = xor (5,6,7,12) = xor (0,1,0,1) = 0
- P_8 = xor (9,10,11,12) = xor (1,0,0,1) = 0

1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	0	0	1	0	0	1	0	0	1

Sample 1 (cont'd)

- Destination

1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	0	0	1	0	0	1	0	0	1

- C_1 = xor (1,3,5,7,9,11) = xor (0,1,0,0,1,0) = 0
- C_2 = xor (2,3,6,7,10,11) = xor (0,1,1,0,0,0) = 0
- C_4 = xor (4,5,6,7,12) = xor (0,0,1,0,1) = 0
- C_8 = xor (8,9,10,11,12) = xor (0,1,0,0,1) = 0

Sample 1 (cont'd)

- Destination

1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	1	0	0	1	0	0	1

- C_1 = xor (1,3,5,7,9,11) = xor (0,0,0,0,1,0) = 1
- C_2 = xor (2,3,6,7,10,11) = xor (0,0,1,0,0,0) = 1
- C_4 = xor (4,5,6,7,12) = xor (0,0,1,0,1) = 0
- C_8 = xor (8,9,10,11,12) = xor (0,1,0,0,1) = 0

Thank You