Chapter 1

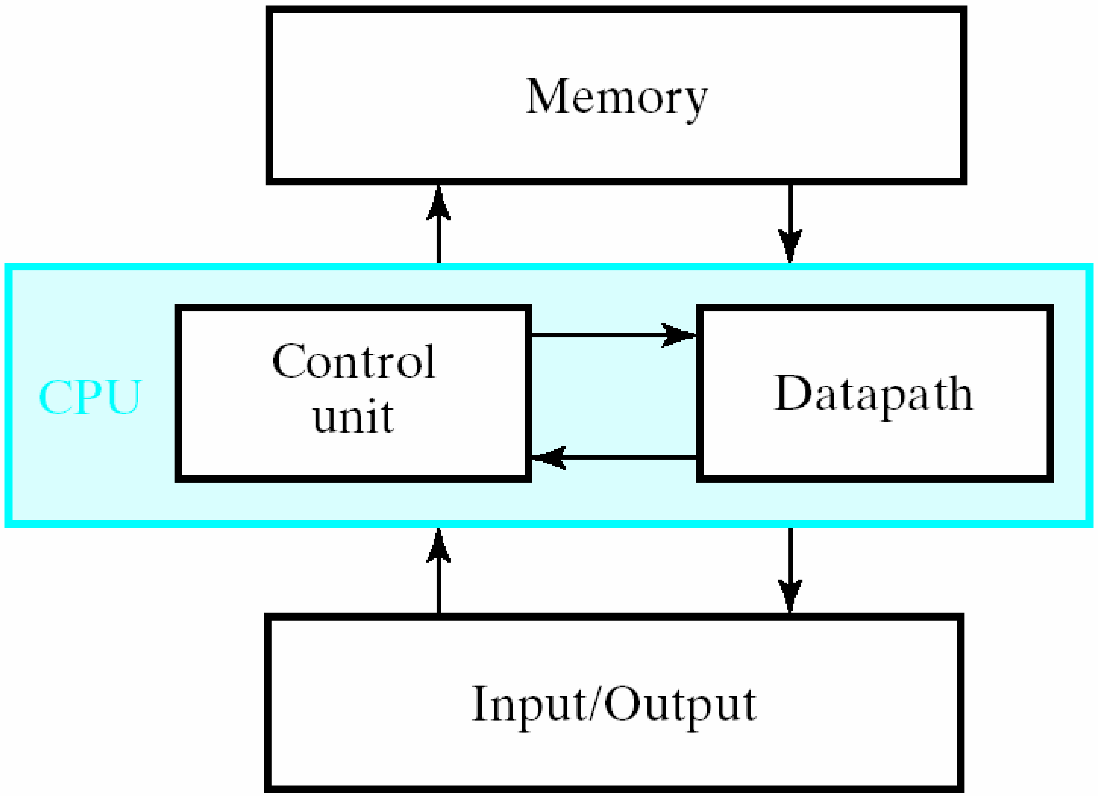
Digital Computers and Information

J.J. Shann

Chapter Overview

1-1 Digital Systems

- Digital system:
  - manipulates discrete elements of information
  - E.g.: general-purpose digital computer



Discrete Information

- Discrete information:
  - any set that is restricted to a finite # of elements
  - E.g.: 10 decimal digits,
    26 letters of the alphabet,
    52 playing cards,
    64 squares of a chessboard
  - Binary values: 2 discrete values
    - are used in most present-day electronic digital systems

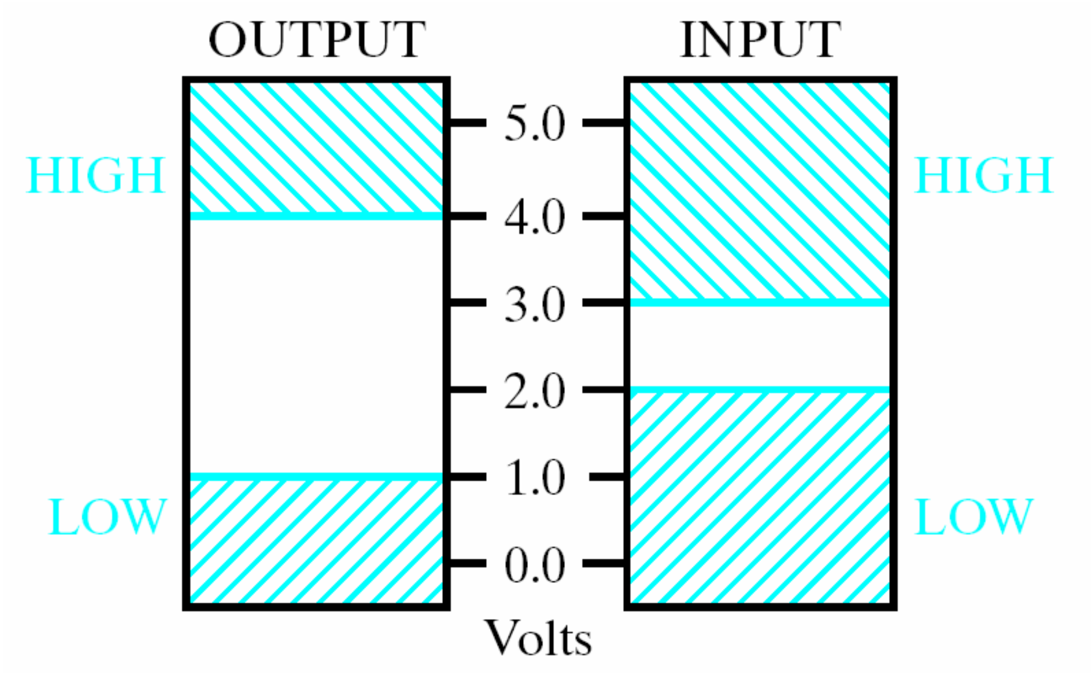
Signal

- Signal:
  - physical quantity used to represent discrete elements
  - E.g.:

CPU	Voltage
Disk	Magnetic field direction
Dynamic RAM	Electrical charge

Binary Signal

- Binary signal:
  - represents two discrete elements
  - E.g.: voltage ranges for binary signals



- Abstract representation of binary values:
  - HIGH (H), LOW (L)
  - TRUE (T), FALSE (F)
  - ON, OFF
  - 0, 1
- Why is binary used?
  - The resulting transistor ckt w/ an output that is either HIGH or LOW is simple, easy to design, and extremely reliable.

1-2 Number Systems

- Positive radix, positional number systems:
  - A number with *radix r*: a string of digits

$$\begin{matrix} r^{n-1} & r^{n-2} & \ldots & r^{1} & r^{0} & & r^{-1} & r^{-2} & \ldots & r^{-m+1} & r^{-m} \\ A_{n-1} & A_{n-2} & \ldots & A_1 & A_0 & . & A_{-1} & A_{-2} & \ldots & A_{-m+1} & A_{-m} \end{matrix}$$

$0 \leq A_i < r$ & . is the *radix point*

  - The string of digits represents the power series:

$$(\text{Number})_r = \left(\sum_{i=0}^{i=n-1} A_i \cdot r^i \right) + \left(\sum_{j=-m}^{j=-1} A_j \cdot r^j \right)$$

(Integer Portion) + (Fraction Portion)

Numbers in Different Bases

TABLE 1-2
Numbers with Different Bases

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	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

Conversion Between Bases

- Binary $\rightarrow$ Decimal:
  - expand the number into a power series w/ a base of 2 and add all the terms
  - E.g.: $\mathbf{(11010)_2 = (?)_{10}}$

$$\mathbf{(11010)_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0}$$
$$\mathbf{= (26)_{10}}$$

- Base r $\rightarrow$ Decimal: 同上

Decimal → Binary:

- Subtract the largest power of 2 that gives a positive remainder and record the power.
  Repeat, subtracting from the prior remainder and recording the power, until the remainder is zero.
  Place 1's in the positions in the binary result corresponding to the powers recorded; in all other positions place 0's.
- E.g.: $625_{10} = (1001110001)_2$

$625 - 512 = 113$	$512 = 2^9$
$113 - 64 = 49$	$64 = 2^6$
$49 - 32 = 17$	$32 = 2^5$
$17 - 16 = 1$	$16 = 2^4$
$1 - 1 = 0$	$1 = 2^0$

Decimal $\rightarrow$ Base r : Integer part + Fraction part

– Integer part:

- divide the number and all successive quotients by r and accumulate the remainders.
- E.g.: $(153)_{10} = (?)_8 = (231)_8$

$153 \div 8 = 19 \ldots \mathbf{1}$

$19 \div 8 = 2 \ldots \mathbf{3}$

$2 \div 8 = 0 \ldots \mathbf{2}$

E.g.: p.1-11

$625_{10} = (1001110001)_2$

– Fraction part:

- multiply the number and all successive fractions by r and accumulate the integers. ($\times 8 \equiv \div 8^{-1}$)
- E.g.: $(0.513)_{10} = (?)_8 = (0.406\ldots)_8$

$0.153 \times 8 = \mathbf{4}.104$

$0.104 \times 8 = \mathbf{0}.832$

$0.832 \times 8 = \mathbf{6}.656$

$\vdots \qquad \vdots$

- Binary → Octal/Hexadecimal:
  - Partition the binary number into groups of 3/4 bits each, starting from the binary point and proceeding to the left and to the right.

    The corresponding octal/hexadecimal digits is then assigned to each group.
  - E.g.: $(010\ 110\ 001\ 101\ 011\ .\ 111\ 100\ 000\ 110)_2$

    $= (\ 2\ \ 6\ \ 1\ \ 5\ \ 3\ .\ 7\ \ 4\ \ 0\ \ 6\ )_8$
- Octal/Hexadecimal → Binary:
  - Each octal/hexadecimal digit is converted to a 3/4-bit binary equivalent and extra 0's are deleted.
  - E.g.: $(673.12)_8 = (110\ 111\ 011\ .\ 001\ 01\cancel{0})_2$

1-3 Arithmetic Operations

- Arithmetic ops w/ numbers in base *r* :
  - follow the same rules as for decimal numbers.
  - Notice: When a base other than base 10 is used:
    - use only *r* allowable digits
    - perform all computations w/ base-*r* digits

Binary Addition

- E.g.: 10110 + 10111 = 101101

Carries:	1 0 1 1 0 0
Augend:	1 0 1 1 0
Addend:	+ 1 0 1 1 1
Sum:	1 0 1 1 0 1

Binary Subtraction

- E.g.: 10110 – 10011 = 00011

Borrows:	0 0 1 1 0
Augend:	1 0 1 1 0
Addend:	+ 1 0 0 1 1
Sum:	0 0 0 1 1

- E.g.: 10011 – 11110 = – 01011

Borrows:		0 0 1 1 0
Augend:	10011	1 1 1 1 0
Addend:	–11110	– 1 0 0 1 1
Sum:		– 0 1 0 1 1

Binary Multiplication

- E.g.: 10110 – 10011 = 00011

$$
\begin{array}{lr}
\text{Multiplicand:} & 1011 \\
\text{Multiplier:} & \times\quad 101 \\
\hline
 & 1011 \\
 & 0000 \\
 & 1011 \\
\hline
\text{Product:} & 110111
\end{array}
$$

Base-*r* Arithmetic Operations

- Base-*r* addition:
  - Convert each pair of digits in a column to decimal,

    add the digits in decimal, and then

    convert the result to the corresponding sum and carry in the base-*r* system.

1-4 Decimal Codes

- Binary vs. Decimal number system:
  - Binary: the most natural system for a computer
  - Decimal: people are accustomed to it
- *n*-bit binary code:
  - a group of n bits that assume up to 2^n distinct combinations of 1's and 0's
  - each combination represents one element of the set being coded
  - will have some unassigned bit combinations if the # of elements in the set is not a power of 2.
- Decimal codes:
  - represent the decimal digits by a code that contains 1's and 0's

Binary-Coded Decimal (BCD)

- Binary-coded decimal (BCD):
  - 1010 ~ 1111 are not used and have no meaning.
  - A number w/ *n* decimal digits requires 4*n* bits in BCD.
    - E.g.: $(185)_{10} = (0001\ 1000\ 0101)_{BCD}$
      $= (10111001)_2$
  - Note: BCD numbers are decimal numbers and not binary numbers.
  - Adv.: Computer input and output data are handled by people who use the decimal system.

Decimal Symbol	BCD Digit
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

BCD Addition

- BCD addition:
  - In each position, use binary arithmetic to add the digits.

    If the binary sum is greater than 1001, add 0110 to obtain the correct BCD digits sum and a carry.
  - E.g.: 448 + 489 = 937 = $(1001\ 0011\ 0111)_{BCD}$

BCD carry	1	1	
	0100	0100	1000
	+ 0100	+ 1000	+ 1001
Binary sum	1001	1101	1 0001
Add 6		+ 0110	+ 0110
BCD sum		1 0011	1 0111

1-5 Gray Codes

- Gray code:

Binary Code	Bit Changes	Gray Code	Bit Changes
000		000	
	1		1
001		001	
	2		1
010		011	
	1		1
011		010	
	3		1
100		110	
	1		1
101		111	
	2		1
110		101	
	1		1
111		100	
	3		1
000		000	

1-6 Alphanumeric Codes

- ASCII character code: 7 bits

TABLE 1-5
American Standard Code for Information Interchange (ASCII)

	$B_7B_6B_5$							
$B_4B_3B_2B_1$	000	001	010	011	100	101	110	111
0000	NULL	DLE	SP	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

Problems

Sections	Exercises
§1-1	
§1-2	1-1 ~ 1-9, 1-12 ~1-14
§1-3	1-10 ~ 1-11
§1-4	1-15 ~ 1-17
§1-5	1-18 ~ 1-19
§1-6	1-20 ~ 1-24

- Homework:

  1-7, 1-14, 1-19, 1-22, 1-24