Binary Codes

- **Computers also use binary numbers to represent non-numeric information, such as text or graphics.**
- **Binary representations of text, (letters, textual numbers, punctuation symbols, etc.) are called codes.**
- **In a binary code, the binary number is a symbol and does not represent an actual number.**
- **A code normally cannot be "operated on" in the usual fashion – mathematical, logical, etc. That is, one can not usually add up, for example, two binary codes. It would be like attempting to add text and graphics!**

Character representation- ASCII

- ASCII (American Standard Code for Information Interchange) - **Binary Codes**
- It is the scheme used to represent characters.
- Each character is represented using 7-bit binary code.
- If 8-bits are used, the first bit is always set to 0

Numeric and Alphabetic Codes

- **ASCII code**
  - **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
  - an **alphanumeric code**
  - each character represented by a 7-bit code
    - gives 128 possible characters
    - codes defined for upper and lower-case alphabetic characters,
      digits 0 – 9, punctuation marks and various non-printing control characters (such as carriage-return and backspace)

ASCII – example

Symbol	decimal	Binary
7	55	00110111
8	56	00111000
9	57	00111001
:	58	00111010
;	59	00111011
<	60	00111100
=	61	00111101
>	62	00111110
?	63	00111111
@	64	01000000
A	65	01000001
B	66	01000010
C	67	01000011

You can use the debug command in windows to see how string of characters are represented in ASCII codes in memory

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

- Representation schemes:
  - **Top layers - Character string to character sequence**: Write each letter separately, enclosed in quotes. End string with '\0'.

    Notation: enclose strings in double quotes

    "Hello world"

    ```
    'H' 'e' 'l' 'l' 'o' ' ' 'W' 'o' 'r' 'l' 'd' '\0'
    ```

  - **Bottom layer - Character to bit-string**: Represent a character using the binary equivalent according to the ASCII table provided.

"SI"
'S' 'I' '\0'
0101001101001000 00000000

The colors are intended to help you read it; computers don't care that all the bits run together.

exercise

- Use the ASCII table to write the ASCII code for the following:
  - CIS110
  - 6=2*3
  - Write your name in hexadecimal.

Unicode - representation

- **ASCII** code can represent only $128 = 2^7$ characters.
- It only represents the English Alphabet, numeric characters, few other characters plus some control characters.
- **Unicode** is designed to represent the worldwide printable and non printable characters.
- It uses 16 bits (or more) and can represent 65536 characters (or more).
- For compatibility, the first 128 Unicode are the same as the one of the **ASCII**.

Unicode cont'd..

- Let's consider how Ethiopia's character sets are represented
- The character set is called Ethiopic
- Range: 1200-1378 (in hexadecimal)
- Example character sets

Syllables

1200	ሀ	ETHIOPIC SYLLABLE HA
1201	ሁ	ETHIOPIC SYLLABLE HU
1202	ሂ	ETHIOPIC SYLLABLE HI
1203	ሃ	ETHIOPIC SYLLABLE HAA
1204	ሄ	ETHIOPIC SYLLABLE HEE
1205	ህ	ETHIOPIC SYLLABLE HE
1206	ሆ	ETHIOPIC SYLLABLE HO
1207	ሇ	ETHIOPIC SYLLABLE HOA
1208	ለ	ETHIOPIC SYLLABLE LA
1209	ሉ	ETHIOPIC SYLLABLE LU
120A	ሊ	ETHIOPIC SYLLABLE LI
120B	ላ	ETHIOPIC SYLLABLE LAA
120C	ሌ	ETHIOPIC SYLLABLE LEE
120D	ል	ETHIOPIC SYLLABLE LE
120E	ሎ	ETHIOPIC SYLLABLE LO
120F	ሏ	ETHIOPIC SYLLABLE LWA
1210	ሐ	ETHIOPIC SYLLABLE HHA
1211	ሑ	ETHIOPIC SYLLABLE HHU
1212	ሒ	ETHIOPIC SYLLABLE HHI
1213	ሓ	ETHIOPIC SYLLABLE HHAA
1214	ሔ	ETHIOPIC SYLLABLE HHEE
1215	ሕ	ETHIOPIC SYLLABLE HHE
1216	ሖ	ETHIOPIC SYLLABLE HHO
1217	ሗ	ETHIOPIC SYLLABLE HHWA
1218	መ	ETHIOPIC SYLLABLE MA
1219	ሙ	ETHIOPIC SYLLABLE MU

1242	ቒ	ETHIOPIC SYLLABLE QI
1243	ቓ	ETHIOPIC SYLLABLE QAA
1244	ቔ	ETHIOPIC SYLLABLE QEE
1245	ቕ	ETHIOPIC SYLLABLE QE
1246	ቖ	ETHIOPIC SYLLABLE QO
1247	ቘ	ETHIOPIC SYLLABLE QOA
1248	ቚ	ETHIOPIC SYLLABLE QWA
1249		<reserved>
124A	ቜ	ETHIOPIC SYLLABLE QWI
124B	ቝ	ETHIOPIC SYLLABLE QWAA
124C	቞	ETHIOPIC SYLLABLE QWEE
124D	቟	ETHIOPIC SYLLABLE QWE
124E		<reserved>
124F		<reserved>
1250	ቐ	ETHIOPIC SYLLABLE QHA
1251	ቑ	ETHIOPIC SYLLABLE QHU
1252	ቒ	ETHIOPIC SYLLABLE QHI
1253	ቓ	ETHIOPIC SYLLABLE QHAA
1254	ቔ	ETHIOPIC SYLLABLE QHEE
1255	ቕ	ETHIOPIC SYLLABLE QHE
1256	ቖ	ETHIOPIC SYLLABLE QHO
1257		<reserved>
1258	ቘ	ETHIOPIC SYLLABLE QHWA
1259		<reserved>
125A	ቚ	ETHIOPIC SYLLABLE QHWI
125B	ቛ	ETHIOPIC SYLLABLE QHWAA
125C	ቜ	ETHIOPIC SYLLABLE QHWEE
125D	ቝ	ETHIOPIC SYLLABLE QHWE

exercise

- Use UNICODE character representation to write the following:
  - ս ու կ կ կ ս ո

Boolean Algebra

- Boolean algebra is a mathematical system for the manipulation of variables that can have one of two values.
  - In formal logic, these values are “true” and “false.”
  - In digital systems, these values are “on” and “off,” 1 and 0, or “high” and “low.”
- Boolean expressions are created by performing operations on Boolean variables.
  - Common Boolean operators include AND, OR, and NOT.

Boolean Algebra

- A Boolean operator can be completely described using a truth table.
- The truth table for the Boolean operators AND and OR are shown at the right.
- The AND operator is also known as a Boolean product. The OR operator is the Boolean sum.

X AND Y

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

X OR Y

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

Boolean Algebra

- The truth table for the Boolean NOT operator is shown at the right.
- The NOT operation is most often designated by an overbar. It is sometimes indicated by a prime mark (‘) or an “elbow” (¬).

NOT x

x	$\overline{x}$
0	1
1	0

Boolean Algebra

- A Boolean function has:
  - At least one Boolean variable,
  - At least one Boolean operator, and
  - At least one input from the set {0,1}.
- It produces an output that is also a member of the set {0,1}.

Most modern programming Languages include the **Boolean** data type.

Boolean Algebra

- The truth table for the Boolean function:

  $F(x, y, z) = x\bar{z} + y$

  is shown at the right.
- To make evaluation of the Boolean function easier, the truth table contains extra (shaded) columns to hold evaluations of subparts of the function.

$F(x, y, z) = x\bar{z} + y$

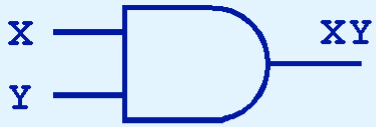
x	y	z	$\bar{z}$	$x\bar{z}$	$x\bar{z}+y$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

Logic Gates

- We have looked at Boolean functions in abstract terms.
- In this section, we see that Boolean functions are implemented in digital computer circuits called gates.
- A gate is an electronic device that produces a result based on two or more input values.
  - In reality, gates consist of one to six transistors, but digital designers think of them as a single unit.
  - Integrated circuits contain collections of gates suited to a particular purpose.

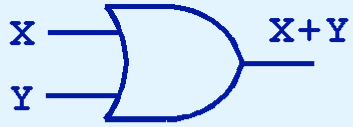
Logic Gates

- The three simplest gates are the AND, OR, and NOT gates.



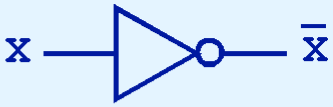
X AND Y

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1



X OR Y

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1



NOT X

X	$\overline{X}$
0	1
1	0

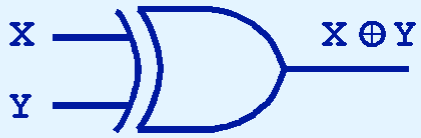
- They correspond directly to their respective Boolean operations, as you can see by their truth tables.

Logic Gates

- Another very useful gate is the exclusive OR (XOR) gate.
- The output of the XOR operation is true only when the values of the inputs differ.

X XOR Y

X	Y	X⊕Y
0	0	0
0	1	1
1	0	1
1	1	0



Note the special symbol ⊕ for the XOR operation.

Adders

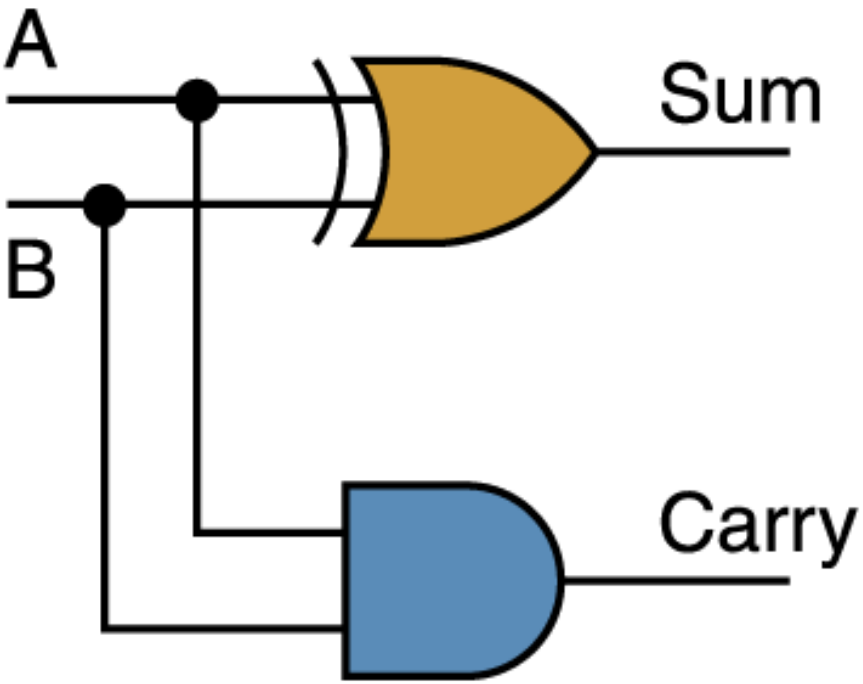
- At the digital logic level, addition is performed in binary
- Addition operations are carried out by special circuits called, appropriately, **adders**

Adders

- The result of adding two binary digits could produce a *carry value*
- Recall that 1 + 1 = 10 in base two
- A circuit that computes the sum of two bits and produces the correct carry bit is called a **half adder**

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Adders



- Circuit diagram representing a half adder
- Two Boolean expressions:

$$\text{sum} = A \oplus B$$

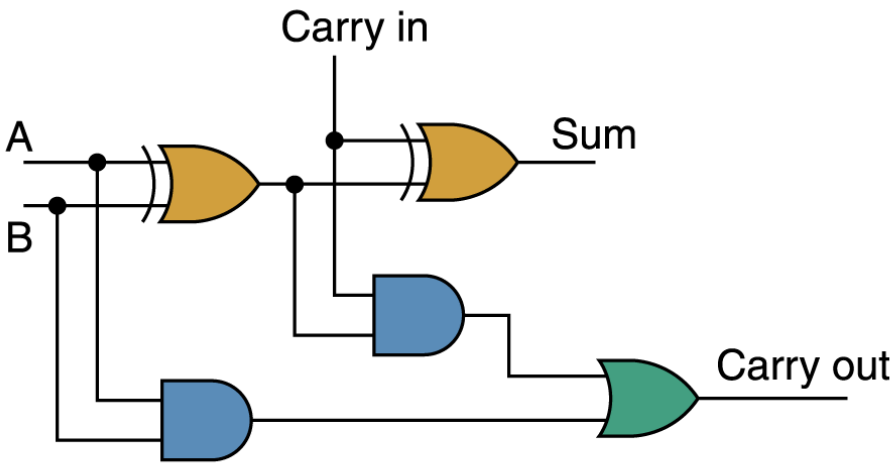
$$\text{carry} = AB$$

Q. Draw the logic circuit for the function $F(x, y, z) = x\bar{z} + y$

Adders

- A circuit called a **full adder** takes the carry-in value into account

Logic Diagram

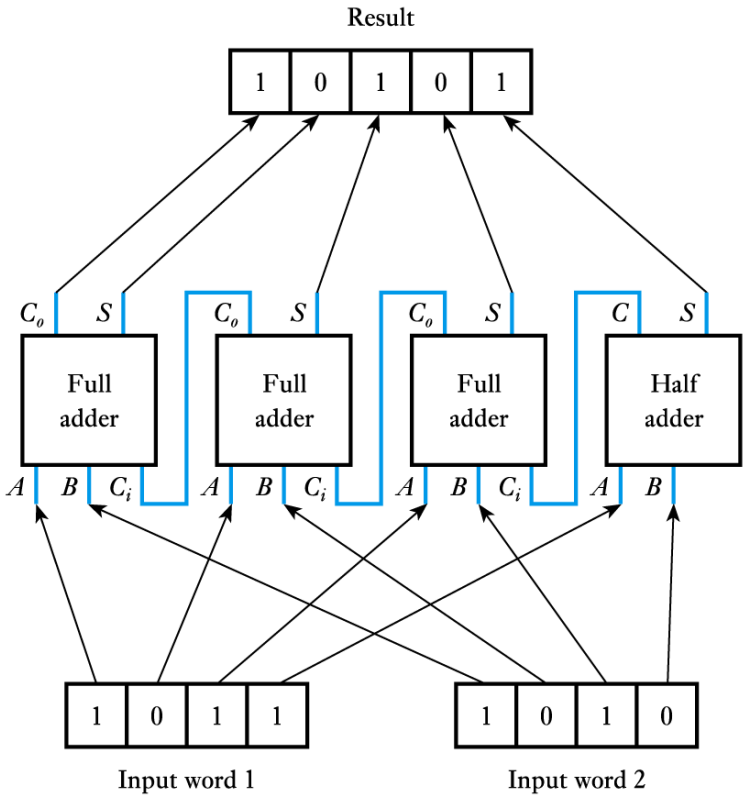


Truth Table

A	B	Carry-in	Sum	Carry-out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Full Adder

- More complex circuits can add digital words



- Similar circuits can be constructed to perform subtraction
- More complex arithmetic (such as multiplication and division) *can* be done by dedicated hardware but is more often performed using a microcomputer or complex logic device

Assignment:

- Construct a digital circuit that takes a 4-bit binary number as an input and outputs the 2’s complement of the entered binary number.