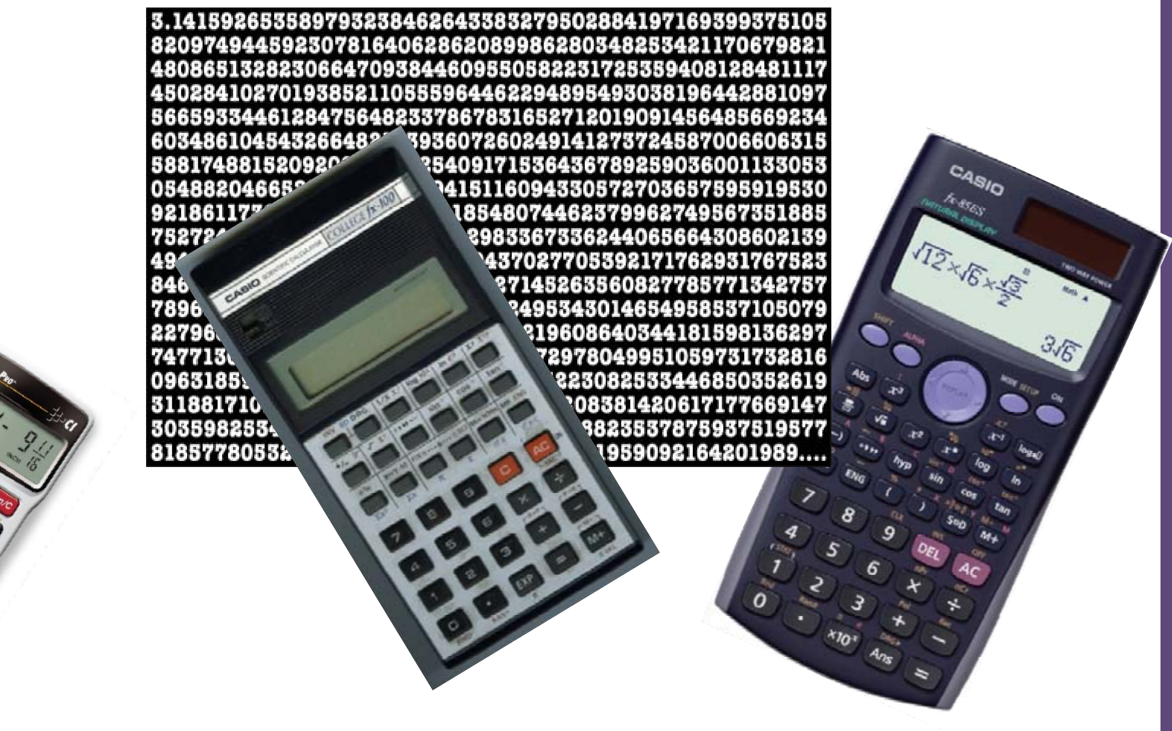
Dr./Nermien Ismail Schools

★ N.I.A.S ★



Title: Talking with Machines--Binary Conversions

Introductory Activity:

Purpose: A motivational puzzle that demonstrates the place value of digits in the decimal number system.

Find the number.

1 It is a three-digit whole number.
2 It is divisible by 5.
3 It is an even number.
4 Each of its digits is different.
5 Its tens digit is greater than its ones digit.
6 Its hundreds digit is greater than its tens digit.
7 It is less than 400.
8 It is divisible by three.
9 It has only one odd digit.
Answer: 210

Decimal Number System

Decimal Number System -- A number system, having a base of 10. Digits are 0 through 9. It is the most popular system in use. Also referred to as the Arabic number system.

Place Value by Position

Digit name by position Thousands Hundreds Tens Ones

Exponential digit value by position 10^3 10^2 10^1 10^0

We can demonstrate the value of each position(*place value*) by analyzing a sample decimal number.

Example: $3210_{10} = 3 \times 10^3 + 2 \times 10^2 + 1 \times 10^1 + 0 \times 10^0 = 3000 + 200 + 10 + 0 = 3210_{10}$

Before moving onto the binary system you may want to peak the students interest by demonstrating a math magic trick based on the binary system. It works as follows:

1) Give students the worksheet full of numbers categorized in sections A through E. 2) Ask them to pick a number between 1 and 30. 3) Ask them to find which boxes it appears in (it could be in more than one) and give the corresponding letters. 4) Use the following place value system to get a number for each letter. A - 16 B - 8 C - 4 D - 2 E - 1 5) Add up the numbers and you will have their number.

Example: If a student chose the number 25 it will appear in section A,B, and E 16 + 8 + 1 will give the number chosen (25).

Binary Number System

Binary Number System --A number system having a base of two. Binary digits are digits 0 and 1.

The binary number system is much simpler than the decimal system. It is used because it is very compatible with digital electronic circuits that are either on or off. The two binary digits of 0 and 1 are used to represent this occurrence. Binary digits are sometimes called bits, which evolved from the first and last two letters of the two words *bi* nary digi*ts*..

Place Value by Position

Digit name by position Eights Fours Twos Ones

Exponential digit value by position 2^3 2^2 2^1 2^0

We can demonstrate the value of each position(*place value*) by analyzing a sample binary number.

Example: 11012 = $1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 8 + 4 + 0 + 1 = 13_{10}$

This example illustrates one way to convert a binary number to its equivalent decimal value.

Chalkboard examples:

Try converting the following binary numbers to decimal.

1110_2 1111_2 1010_2

Answers: 14_{10}, 15_{10}, 10_{10}

Octal Number System

Octal Number System -- A number system, having a base of 8. Digits are 0 through 7.

Normally, data is fed to a computer in some system other than binary. This is because entering data in binary is time - consuming and prone to error. Also binary numbers are difficult to remember and express in words. The octal system is more closely related to binary than to decimal. Due to the large values of each digits position we will not name them below (digit name by position).

Place Value by Position

Exponential digit value by position 8^3 8^2 8^1 8^0

We can demonstrate the value of each position(*place value*) by analyzing a sample decimal number.

Example: $32108 = 3 \times 8^3 + 2 \times 8^2 + 1 \times 8^1 + 0 \times 8^0 = 3 \times 512 + 2 \times 64 + 1 \times 8 + 0 \times 1 = 1536 + 128$
$+ 8 + 0 = 1672_{10}$

This example illustrates one way to convert an octal number to its equivalent decimal value.

Chalkboard Examples:

Try converting the following octal numbers to decimal.

301_8 417_8 1036_8

Answers: 193_{10}, 271_{10}, 542_{10}

Hexadecimal Number System

Hexadecimal Number System -- A number system, having a base of 16. Convenient for representing 4 bit numbers. Digits are 0 through 9 and A through F.

The hexadecimal system was named from the prefix *hexa* which stands for six and *decimal* which implies ten. The first ten digits are the same as the decimal system (0-9). However, for the decimal numbers 11-15, the hexadecimal number system uses the letters A - F. Since many personal computers today use a 16-bit *assembly language*, the hexadecimal system has become very popular. The hexadecimal system is also more closely related to binary than decimal. Because of the large values for each digits position we will not name them below (digit name by position).

<u>Place Value by Position</u>

Exponential digit value by position 16^3 16^2 16^1 16^0

We can demonstrate the value of each position(*place value*) by analyzing a sample decimal number.

Example: $321016 = 3 \times 16^3 + 2 \times 16^2 + 1 \times 16^1 + 0 \times 16^0$
$= 3 \times 4096 + 2 \times 256 + 1 \times 16 + 0 \times 1$
$=12288 + 512 + 16 + 0$
$= 12816_{10}$

This example illustrates one way to convert a hexadecimal number to its equivalent decimal value.

Chalkboard Examples:

Try converting the following hexadecimal numbers to decimal.

$3C1_{16}$ $41A_{16}$ $1F06_{16}$

Answers: 961_{10}, 1050_{10}, 7942_{10}

Converting Number Systems to Binary

Digital integrated circuits handle digital information using switching circuits. These simple circuits made up of diodes, transistors, and resistors can perform the basic Boolean logic functions. Boolean logic uses the binary number system. Therefore it is important to understand how to convert number systems into binary.

Converting Decimal to Binary

Method: Successive division by two

1) Divide the decimal number by two.2) Use the remainder from this division to fill in the ones place value. 3) Continue to divide by two on each resulting quotient. 4) Each remainder fills the next higher place value. 5) The procedure is over when you have a division of zero.

Example: 15310 is converted as follows.

153 / 2 = 76 remainder 1 ==> 1s place
76 / 2 = 38 remainder 0 ==> 2s place
38 / 2 = 19 remainder 0 ==> 4s place
19 / 2 = 9 remainder 1 ==> 8s place
9 / 2 = 4 remainder 1 ==> 16s place
4 / 2 = 2 remainder 0 ==> 32s place
2 / 2 = 1 remainder 0 ==> 64s place
1 / 2 = 0 remainder 1 ==> 128s place
0 / 2 = 0

Result: $(153)_{10} = (10011001)_2$

Chalkboard Examples: Try converting the following decimal numbers to binary. 127_{10} 93_{10} 80_{10} Answers:

1111111_2, 1011101_2, 1010000_2

Converting Octal to Binary

Octal is more closely related to binary than is decimal. Therefore it is important to understand how to convert octal to binary.

Method: 1) Convert each octal digit to its three-digit binary equivalent.
2) Record it from left to right starting with the first 1.

Example: 3258 ==> 3 2 5
==> 011 010 101 (binary equivalent)

Result: 11010101_2

Chalkboard examples: Try converting the following octal numbers to binary.

100_8 427_8 702_8

Answers: 1000000_2, 100010111_2, 111000010_2

Converting Hexadecimal to Binary

Hexadecimal is also more closely related to binary than is decimal. Again, the relationship permits easy conversion between the systems. Therefore it is important to understand how to convert hexadecimal to binary.

Method: 1) Convert each hexadecimal digit to its four-digit binary equivalent. 2) Record it from left to right starting with the first 1.

Example: $A25_{16}$ ==> A 2 5 ==> 1010 0010 0101 (binary equivalent)

Result: 101000100101_2

Chalkboard examples: Try converting the following hexadecimal numbers to binary. 9F116 $D03_{16}$ $52C_{16}$

Answers: 100111110001_2, 110100000011_2, 10100101100_2

At this time the students should complete the worksheet entitled Converting to Binary. After numbers in decimal, octal, and hexadecimal are converted to binary the ASCII code chart can be used to find the corresponding letter. The secret message will read: Math controls the world. For example:

77_{10} will be converted to 1001101_2 which corresponds to the letter M on the ASCII code chart.

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
0	0	NUL	16	10	DLE	32	20	(space)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|

77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	

Free auto converter available at http://www.cut-the-knot.org/binary.shtml

The Binary System

Table of Contents

Basic Concepts Behind the Binary System

To understand binary numbers, begin by recalling elementary school math. When we first learned about numbers, we were taught that, in the decimal system, things are organized into columns:

```
    H | T | O
    1 | 9 | 3
```

such that "H" is the hundreds column, "T" is the tens column, and "O" is the ones column. So the number "193" is 1-hundreds plus 9-tens plus 3-ones.

Years later, we learned that the ones column meant 10^0, the tens column meant 10^1, the hundreds column 10^2 and so on, such that

```
      10^2|10^1|10^0
        1 |  9 |  3
```

the number 193 is really {(1*10^2)+(9*10^1)+(3*10^0)}.

As you know, the decimal system uses the digits 0-9 to represent numbers. If we wanted to put a larger number in column 10^n (e.g., 10), we would have to multiply 10*10^n, which would give 10^(n+1), and be carried a column to the left. For example, putting ten in the 10^0 column is impossible, so we put a 1 in the 10^1 column, and a 0 in the 10^0 column, thus using two columns. Twelve would be 12*10^0, or 10^0(10+2), or 10^1+2*10^0, which also uses an additional column to the left (12).

The binary system works under the exact same principles as the decimal system, only it operates in base 2 rather than base 10. In other words, instead of columns being

```
       10^2|10^1|10^0
```

they are

```
        2^2|2^1|2^0
```

Instead of using the digits 0-9, we only use 0-1 (again, if we used anything larger it would be like multiplying 2*2^n and getting 2^n+1, which would not fit in the 2^n column. Therefore, it would shift you one column to the left. For example, "3" in binary cannot be put into one column. The first column we fill is the right-most column, which is 2^0, or 1. Since 3>1, we need to use an extra column to the left, and indicate it as "11" in binary (1*2^1) + (1*2^0).

Examples: What would the binary number 1011 be in decimal notation?

Answers are available at the end of the sheet

Try converting these numbers from binary to decimal:

- 10
- 111
- 10101
- 11110

Remember:

```
2^4| 2^3| 2^2| 2^1| 2^0
   |    |    |  1  |  0
   |    | 1  |  1  |  1
1  | 0  | 1  |  0  |  1
1  | 1  | 1  |  1  |  0
```

Binary Addition

Consider the addition of decimal numbers:

```
  23
 +48
 ___
```

We begin by adding 3+8=11. Since 11 is greater than 10, a one is put into the 10's column (carried), and a 1 is recorded in the one's column of the sum. Next, add {(2+4) +1} (the one is from the carry)=7, which is put in the 10's column of the sum. Thus, the answer is 71.

Binary addition works on the same principle, but the numerals are different. Begin with one-bit binary addition:

```
   0    0    1
  +0   +1   +0
 ___  ___  ___
   0    1    1
```

1+1 carries us into the next column. In decimal form, 1+1=2. In binary, any digit higher than 1 puts us a column to the left (as would 10 in decimal notation). The decimal number "2" is written in binary notation as "10" (1*2^1)+(0*2^0). Record the 0 in the ones column, and carry the 1 to the twos column to get an answer of "10." In our vertical notation,

```
   1
  +1
 ___
  10
```

The process is the same for multiple-bit binary numbers:

```
   1010
  +1111
 ______
```

- Step one:
  Column 2^0: 0+1=1.
  Record the 1.
  Temporary Result: 1; Carry: 0
- Step two:
  Column 2^1: 1+1=10.
  Record the 0, carry the 1.
  Temporary Result: 01; Carry: 1
- Step three:
  Column 2^2: 1+0=1 Add 1 from carry: 1+1=10.
  Record the 0, carry the 1.
  Temporary Result: 001; Carry: 1
- Step four:
  Column 2^3: 1+1=10. Add 1 from carry: 10+1=11.
  Record the 11.
  Final result: 11001

Alternately:
```
    11   (carry)
    1010
   +1111
  ______
   11001
```
Always remember

- 0+0=0
- 1+0=1
- 1+1=10

Try a few examples of binary addition:
```
       111      101      111
      +110     +111     +111
    ______   ______   ______
```

Binary Multiplication

Multiplication in the binary system works the same way as in the decimal system:

- 1*1=1
- 1*0=0
- 0*1=0

```
   101
  * 11
  ____
   101
  1010
 _____
  1111
```
Note that multiplying by two is extremely easy. To multiply by two, just add a 0 on the end.

Binary Division

Follow the same rules as in decimal division. For the sake of simplicity, throw away the remainder.
For Example: 111011/11

```
      10011 r 10
    _______
  11)111011
    -11
     ______
       101
       -11
     ______
        101
         11
     ______
         10
```

Decimal to Binary

Converting from decimal to binary notation is slightly more difficult conceptually, but can easily be done once you know how through the use of algorithms. Begin by thinking of a few examples. We can easily see that the number 3= 2+1. and that this is equivalent to (1*2^1)+(1*2^0). This translates into putting a "1" in the 2^1 column and a "1" in the 2^0 column, to get "11". Almost as intuitive is the number 5: it is obviously 4+1, which is the same as saying [(2*2) +1], or 2^2+1. This can also be written as [(1*2^2)+(1*2^0)]. Looking at this in columns,

```
        2^2 | 2^1 | 2^0
         1     0     1
```

or 101.

What we're doing here is finding the largest power of two within the number (2^2=4 is the largest power of 2 in 5), subtracting that from the number (5-4=1), and finding the largest power of 2 in the remainder (2^0=1 is the largest power of 2 in 1). Then we just put this into columns. This process continues until we have a remainder of 0. Let's take a look at how it works. We know that:

```
2^0=1
2^1=2
2^2=4
2^3=8
2^4=16
2^5=32
2^6=64
2^7=128
```

and so on. To convert the decimal number 75 to binary, we would find the largest power of 2 less than 75, which is 64. Thus, we would put a 1 in the 2^6 column, and subtract 64 from 75, giving us 11. The largest power of 2 in 11 is 8, or 2^3. Put 1 in the 2^3 column, and 0 in 2^4 and 2^5. Subtract 8 from 11 to get 3. Put 1 in the 2^1 column, 0 in 2^2, and subtract 2 from 3. We're left with 1, which goes in 2^0, and we subtract one to get zero. Thus, our number is 1001011.

Answers

What would the binary number 1011 be in decimal notation?

```
1011=(1*2^3)+(0*2^2)+(1*2^1)+(1*2^0)
    = (1*8) + (0*4) + (1*2) + (1*1)
= 11 (in decimal notation)
```

Try converting these numbers from binary to decimal:

```
10=(1*2^1) + (0*2^0) = 2+0 = 2
111 = (1*2^2) + (1*2^1) + (1*2^0) = 4+2+1=7
10101= (1*2^4) + (0*2^3) + (1*2^2) + (0*2^1) + (1*2^0)=16+0+4+0+1=21
11110= (1*2^4) + (1*2^3) + (1*2^2) + (1*2^1) + (0*2^0)=16+8+4+2+0=30
```

Try a few examples of binary addition:

```
                      1                1
          111            111             111
         +110                  +110            +110
        ______               ______          _____
               1                       01          1101

          1                        11             1
  101            101    101
        +111                 +111            +111
        _____                ____            _____
              0                    00            1100

              1                   1                      1
     111    111            111
           +111                +111                 +111
          _____               _____                _____
         0                    10                  1110
```

Using the regular algorithm for binary adition, add (5+12), (-5+12), (-12+-5), and (12+-12) in each system. Then convert back to decimal numbers.

Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin
0	0	000	00000000	16	10	020	00010000	32	20	040	00100000	48	30	060	00110000
1	1	001	00000001	17	11	021	00010001	33	21	041	00100001	49	31	061	00110001
2	2	002	00000010	18	12	022	00010010	34	22	042	00100010	50	32	062	00110010
3	3	003	00000011	19	13	023	00010011	35	23	043	00100011	51	33	063	00110011
4	4	004	00000100	20	14	024	00010100	36	24	044	00100100	52	34	064	00110100
5	5	005	00000101	21	15	025	00010101	37	25	045	00100101	53	35	065	00110101
6	6	006	00000110	22	16	026	00010110	38	26	046	00100110	54	36	066	00110110
7	7	007	00000111	23	17	027	00010111	39	27	047	00100111	55	37	067	00110111
8	8	010	00001000	24	18	030	00011000	40	28	050	00101000	56	38	070	00111000
9	9	011	00001001	25	19	031	00011001	41	29	051	00101001	57	39	071	00111001
10	A	012	00001010	26	1A	032	00011010	42	2A	052	00101010	58	3A	072	00111010
11	B	013	00001011	27	1B	033	00011011	43	2B	053	00101011	59	3B	073	00111011
12	C	014	00001100	28	1C	034	00011100	44	2C	054	00101100	60	3C	074	00111100
13	D	015	00001101	29	1D	035	00011101	45	2D	055	00101101	61	3D	075	00111101
14	E	016	00001110	30	1E	036	00011110	46	2E	056	00101110	62	3E	076	00111110
15	F	017	00001111	31	1F	037	00011111	47	2F	057	00101111	63	3F	077	00111111

Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin
64	40	100	01000000	80	50	120	01010000	96	60	140	01100000	112	70	160	01110000
65	41	101	01000001	81	51	121	01010001	97	61	141	01100001	113	71	161	01110001
66	42	102	01000010	82	52	122	01010010	98	62	142	01100010	114	72	162	01110010
67	43	103	01000011	83	53	123	01010011	99	63	143	01100011	115	73	163	01110011
68	44	104	01000100	84	54	124	01010100	100	64	144	01100100	116	74	164	01110100
69	45	105	01000101	85	55	125	01010101	101	65	145	01100101	117	75	165	01110101
70	46	106	01000110	86	56	126	01010110	102	66	146	01100110	118	76	166	01110110
71	47	107	01000111	87	57	127	01010111	103	67	147	01100111	119	77	167	01110111
72	48	110	01001000	88	58	130	01011000	104	68	150	01101000	120	78	170	01111000
73	49	111	01001001	89	59	131	01011001	105	69	151	01101001	121	79	171	01111001
74	4A	112	01001010	90	5A	132	01011010	106	6A	152	01101010	122	7A	172	01111010
75	4B	113	01001011	91	5B	133	01011011	107	6B	153	01101011	123	7B	173	01111011
76	4C	114	01001100	92	5C	134	01011100	108	6C	154	01101100	124	7C	174	01111100
77	4D	115	01001101	93	5D	135	01011101	109	6D	155	01101101	125	7D	175	01111101
78	4E	116	01001110	94	5E	136	01011110	110	6E	156	01101110	126	7E	176	01111110
79	4F	117	01001111	95	5F	137	01011111	111	6F	157	01101111	127	7F	177	01111111

Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin
128	80	200	10000000	144	90	220	10010000	160	A0	240	10100000	176	B0	260	10110000
129	81	201	10000001	145	91	221	10010001	161	A1	241	10100001	177	B1	261	10110001
130	82	202	10000010	146	92	222	10010010	162	A2	242	10100010	178	B2	262	10110010
131	83	203	10000011	147	93	223	10010011	163	A3	243	10100011	179	B3	263	10110011
132	84	204	10000100	148	94	224	10010100	164	A4	244	10100100	180	B4	264	10110100
133	85	205	10000101	149	95	225	10010101	165	A5	245	10100101	181	B5	265	10110101
134	86	206	10000110	150	96	226	10010110	166	A6	246	10100110	182	B6	266	10110110
135	87	207	10000111	151	97	227	10010111	167	A7	247	10100111	183	B7	267	10110111
136	88	210	10001000	152	98	230	10011000	168	A8	250	10101000	184	B8	270	10111000
137	89	211	10001001	153	99	231	10011001	169	A9	251	10101001	185	B9	271	10111001
138	8A	212	10001010	154	9A	232	10011010	170	AA	252	10101010	186	BA	272	10111010
139	8B	213	10001011	155	9B	233	10011011	171	AB	253	10101011	187	BB	273	10111011
140	8C	214	10001100	156	9C	234	10011100	172	AC	254	10101100	188	BC	274	10111100
141	8D	215	10001101	157	9D	235	10011101	173	AD	255	10101101	189	BD	275	10111101
142	8E	216	10001110	158	9E	236	10011110	174	AE	256	10101110	190	BE	276	10111110
143	8F	217	10001111	159	9F	237	10011111	175	AF	257	10101111	191	BF	277	10111111

Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin
192	C0	300	11000000	208	D0	320	11010000	224	E0	340	11100000	240	F0	360	11110000
193	C1	301	11000001	209	D1	321	11010001	225	E1	341	11100001	241	F1	361	11110001
194	C2	302	11000010	210	D2	322	11010010	226	E2	342	11100010	242	F2	362	11110010
195	C3	303	11000011	211	D3	323	11010011	227	E3	343	11100011	243	F3	363	11110011
196	C4	304	11000100	212	D4	324	11010100	228	E4	344	11100100	244	F4	364	11110100
197	C5	305	11000101	213	D5	325	11010101	229	E5	345	11100101	245	F5	365	11110101
198	C6	306	11000110	214	D6	326	11010110	230	E6	346	11100110	246	F6	366	11110110
199	C7	307	11000111	215	D7	327	11010111	231	E7	347	11100111	247	F7	367	11110111
200	C8	310	11001000	216	D8	330	11011000	232	E8	350	11101000	248	F8	370	11111000
201	C9	311	11001001	217	D9	331	11011001	233	E9	351	11101001	249	F9	371	11111001
202	CA	312	11001010	218	DA	332	11011010	234	EA	352	11101010	250	FA	372	11111010
203	CB	313	11001011	219	DB	333	11011011	235	EB	353	11101011	251	FB	373	11111011
204	CC	314	11001100	220	DC	334	11011100	236	EC	354	11101100	252	FC	374	11111100
205	CD	315	11001101	221	DD	335	11011101	237	ED	355	11101101	253	FD	375	11111101

206	CE	316	11001110	222	DE	336	11011110	238	EE	356	11101110	254	FE	376	11111110
207	CF	317	11001111	223	DF	337	11011111	239	EF	357	11101111	255	FF	377	11111111

Introduction to Logic Gates

This web site provides a brief description of logic gates and defines a few of the common logic gates found in simple digital circuits.

The navigation bar provides links to pages where you can view the symbol, truth table, and animation of particular logic gates and a sample circuit.

Digital Logic

Keep in mind that computers work on an electrical flow where a high voltage is considered a 1 and a low voltage is considered a 0. Using these highs and lows, data are represented. Electronic circuits must be designed to manipulate these positive and negative pulses into meaningful logic.

Logic gates are the building blocks of digital circuits. Combinations of logic gates form circuits designed with specific tasks in mind. For example, logic gates are combined to form circuits to add binary numbers (adders), set and reset bits of memory (flip flops), multiplex multiple inputs, etc.

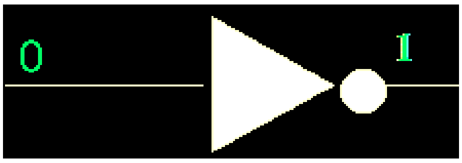
Not gate

The NOT gate is also known as an inverter, simply because it changes the input to its opposite (inverts it). The NOT gate accepts only one input and the output is the opposite of the input. In other words, a low-voltage input (0) is converted to a high-voltage output (1). It's that simple!

Select an input value from the pull-down selector above and view the NOT gate in action.

A common way of using the NOT gate is to simply attach the circle to the front of another gate. This simplifies the circuit drawing and simply says: "Invert the output from this gate."

For example, the combination of an AND and NOT gate is shown in the following picture:

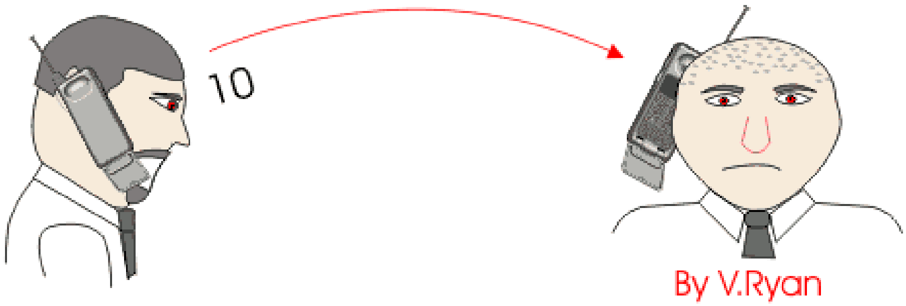


Free animated solver available at:
http://isweb.redwoods.cc.ca.us/INSTRUCT/CalderwoodD/diglogic/index.htm

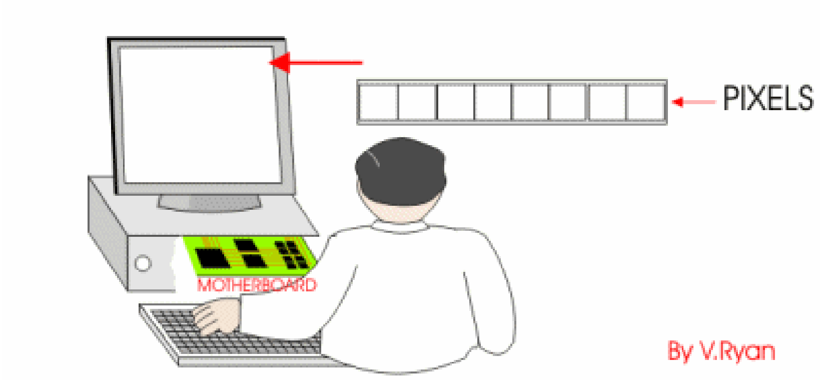
DIGITAL ELECTRONICS - LOGIC CIRCUITS - 1

Most modern electronic devices such as mobile telephones and computers depend on digital electronics. In fact, most electronics about the home and in industry depend on digital electronics to work.
Digital electronics normally based on ‘logic circuits’. These circuits depend on pulses of electricity to make the circuit work. For instance, if current is present - this is represented as ‘1’. If current is not present, this is represented as ‘0’. Digital electronics is based on a series of 1s and 0s.

A good example of a digital electronic system is a mobile phone. As you speak into the phone, the digital electronic circuits it contains converts your voice into a series of electronic pulses (or 1s and 0s). These are transmitted and the receiving mobile phone then converts the digital pulses back into your voice. Digital circuits are used because they are efficient and work well, also, digital signals are easier to transmit than actual sound (for example a persons voice).



The various parts of a computer communicate through the use of electronic pulses (1s and 0s). Consequently digital logic circuits are ideal for the internal electronics. The main part of the computer is the motherboard. This is a complex piece of electronics that processes all the important data. For instance, when word processing, it is very important to display letters and words on the monitor. The motherboard generates the individual letters on the monitor by sending a series of 1s and 0s to the screen.

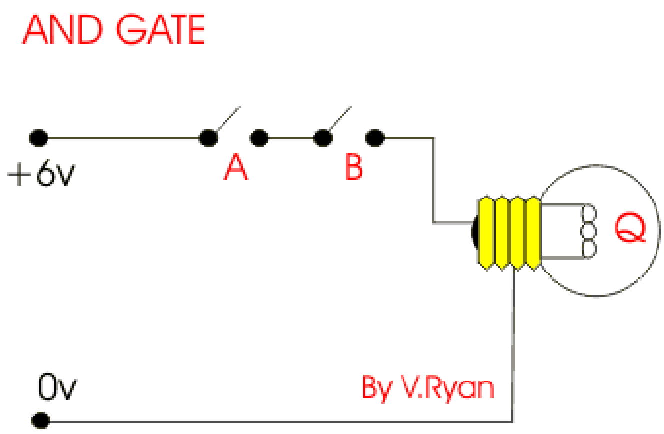


When the computer operator presses the letter 'H' on the keyboard, the motherboard converts this into a digital signal composed of 1s and 0s. The 'H' in the form of 1s and 0s is displayed on the monitor.
When you word-process a paragraph of writing all the letters/words are displayed on the monitor in a similar way. In reality the letters are not composed of 1s and 0s but as black or white pixels.

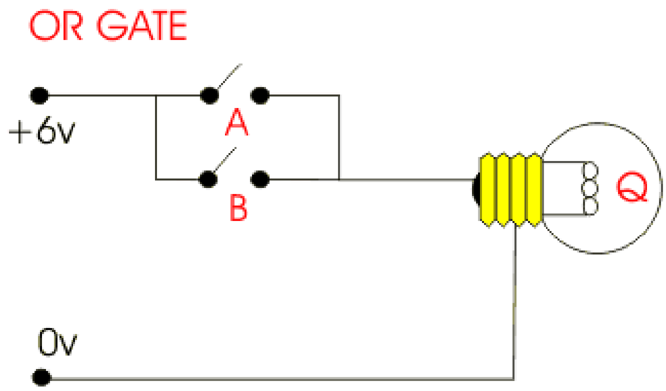
QUESTION:
Look closely at a computer monitor. The pixels are very small but you may be able to see them especially if you use a magnifying glass. If you are looking at a colour picture, the pixels will be in different colours, not only black and white.

DIGITAL ELECTRONICS - LOGIC CIRCUITS - 2

LOGIC circuits are normally composed of 'gates'. A combination of gates make up a circuit and some digital circuits can be extremely complex. It is the logic gates that produce pulses of electrical current (1s and 0s). At school level, digital logic circuits are relatively simple. Below are simple drawings that help explain the two most popular logic gates - the AND gate and the OR gate.



The simplified AND gate shown above has two inputs, switch A and switch B. The bulb Q will only light if both switches are closed. This will allow current to flow through the bulb, illuminating the filament.



The simplified OR gate shown above has two inputs, switch A and switch B. The bulb Q will light if either switch A or B are closed. This will allow current to flow through the bulb, illuminating the filament.

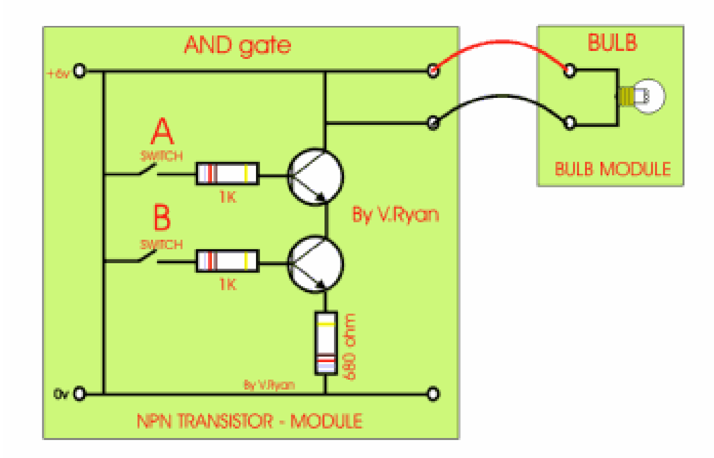
When the bulb lights this represents a ‘1’ as current is running through the filament. If current is not running through the filament the bulb will not light and this represents a ‘0’ (zero).

THE ROLE OF TRANSISTORS

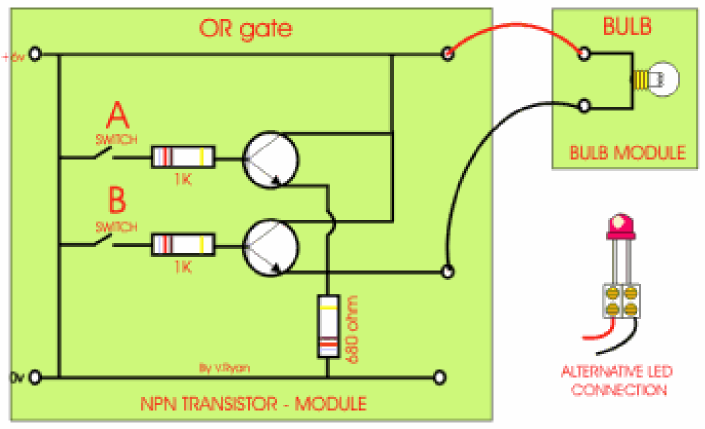
Transistors are vital for digital circuits to work. These components are used as very fast switches in digital logic circuits. Transistors are normally so small that hundreds of thousands fit on one processing chip on a computer motherboard. The types of transistors used in school projects are normally large enough to fit on the end of a small finger. However, the way they switched on and off is the same. When a transistor is switched on it produces a ‘1’ and when it is switched off it produces a ‘0’.

Transistors in the circuit of a computer microprocessor can switch on and off thousands of times per second. Without the invention of the transistor, computer processing power would be very limited and slow.

Two basic examples of simple transistor driven logic (AND / OR) circuits are shown below.



This is an AND gate circuit and it can be made quite easily. The example shown is built from a modular electronics kit. Both switches ‘A’ and ‘B’ must be pressed together for the bulb to light.
If you construct this circuit, you may need to alter the value of the resistors. This will depend on the type of transistors used and whether to bulb or an LED is used.



This is an OR gate circuit. Either switch ‘A’ or ‘B’ must be pressed for the bulb to light. The switches do not have to be pressed together.

QUESTIONS:
1. Explain how an AND gate works. Use a circuit diagram to help explain your answer.
2. Explain how an OR gate works. Use a circuit diagram to help explain your answer.
3. Build a simple logic circuit using a breadboard and available components. You may wish to build one of the circuits shown above.

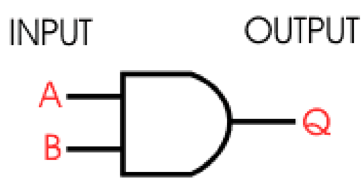
DIGITAL ELECTRONICS - BASIC LOGIC TABLES

A range of logic gates exist and they are represented as symbols, each with its own truth table (sometimes called a logic table). Gates have inputs and produce outputs and these are in the form of 1s and 0s. Remember, a 1 represents an input or output of electrical current. Each truth table clearly shows the ‘state’ of inputs and outputs at any one time.
Study the symbols and tables below. You will soon find that they can be combined to design interesting logic circuits.

The AND gate will only output current (produce a 1 at Q) if both logic states at inputs A and B change to 1.

AND gate

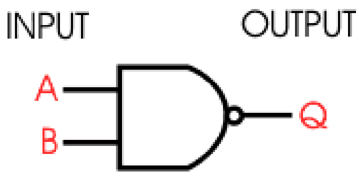
A	B	Q
0	0	0



The NAND gate has the opposite outputs to the AND gate. How does the NAND gate symbol differ to the AND gate?

NAND gate

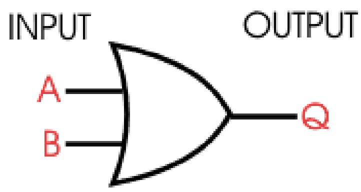
A	B	Q
0	0	1



The OR gate will output current at Q if either of the logic states of inputs A and B change to1.

OR gate

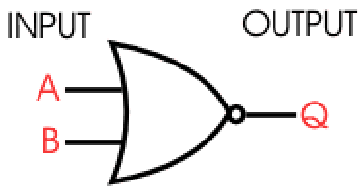
A	B	Q
0	0	0



The NOR gate has the opposite outputs to the OR gate. How does the NOR gate symbol differ to the OR gate?

NOR gate

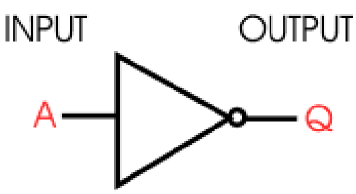
A	B	Q
0	0	1



The INVERTER gate reverses input. For example, if the input is 1 then the output is 0.
This is a very useful gate especially when designing logic circuits.

INVERTER gate

A	Q
0	1



QUESTIONS:
1. Draw each of the logic gates shown above and explain how each gate works.
2. Learn and remember each of the logic tables.

ALTERNATIVE REPRESENTATIONS OF LOGIC GATES

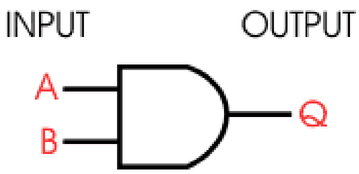
A ‘1’ means that current is present. For instance, if current is present at an output of a gate then this is represented as a ‘1’. Instead of placing a ‘1’ at the output other terms can be applied - high, true, on or up - all mean that current is present.

A ‘0’ means that current is not present. For instance, if current is not present at an output of a gate then this is represented as a ‘0’. Instead of placing a'0’ at the output other terms can be applied - low, false, off or low - all mean that current is not present.

Alternative ways of representing the AND gate are written below.

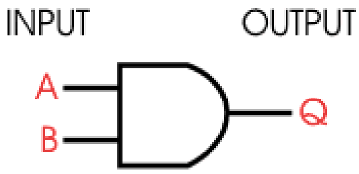
AND gate

A	B	Q
LOW	LOW	LOW



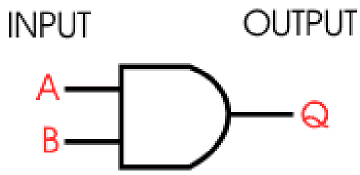
AND gate

A	B	Q
OFF	OFF	OFF



AND gate

A	B	Q
FALSE	FALSE	FALSE

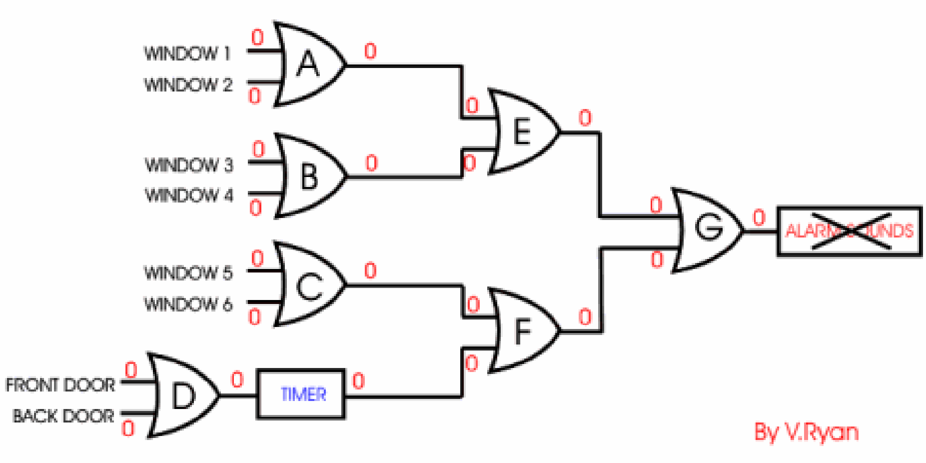


QUESTION:
Write the OR truth table using alternative terms other than ‘1s’ and ‘0s’.

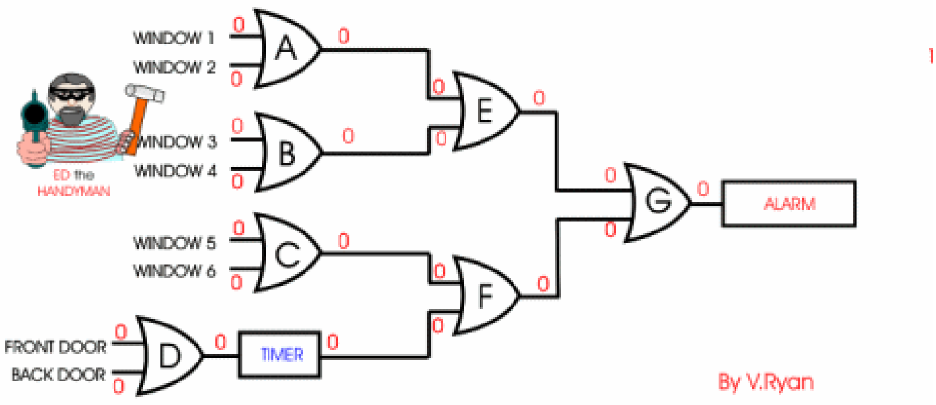
EXAMPLE LOGIC CIRCUITS - 1

Below is the logic circuit for a simple house alarm. The alarm protects the front and back doors and six windows. Once the alarm is set if any of the doors or windows are opened the alarm will sound. OR gates have been used. The TIMER allows the house owner to enter the house by either the front or back door and turn of the alarm system before the alarm sounds. The inputs for each of the gates representing the doors and windows can be connected to a vast range of sensors (eg. movement and magnetic sensors).

On the circuit below the input states of each of the sensors are '0' (false, low, off). This means that they have not detected an intruder. As a result the alarm does not sound.



The situation changes when local thug, Ed the Handyman forces window 3 open. Notice how the logic state of the input of GATE B changes from false to true. The output state of gate 'B' changes to true, followed by the INPUT of gate 'E' and its output. The input and output of gate 'G' also change to true. This train of events leads to the alarm sounding. Because OR gates have been used, it only takes one input to change to true at the windows or doors to activate the alarm.

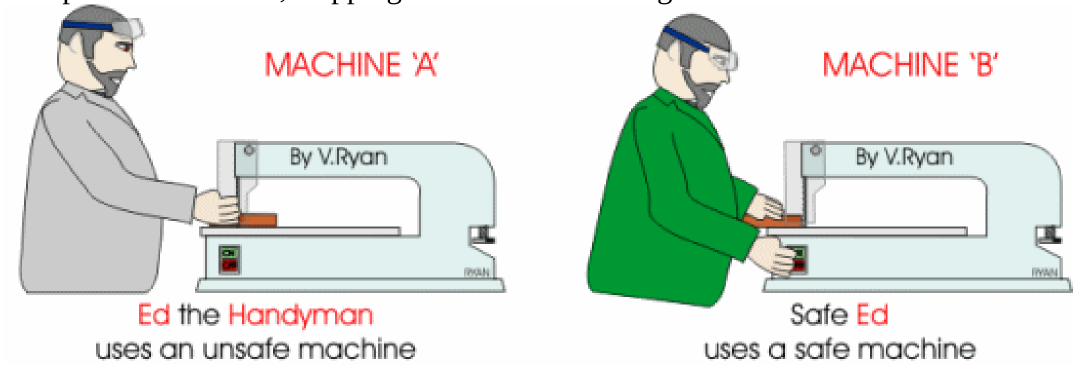


EXAMPLE LOGIC CIRCUITS – 2

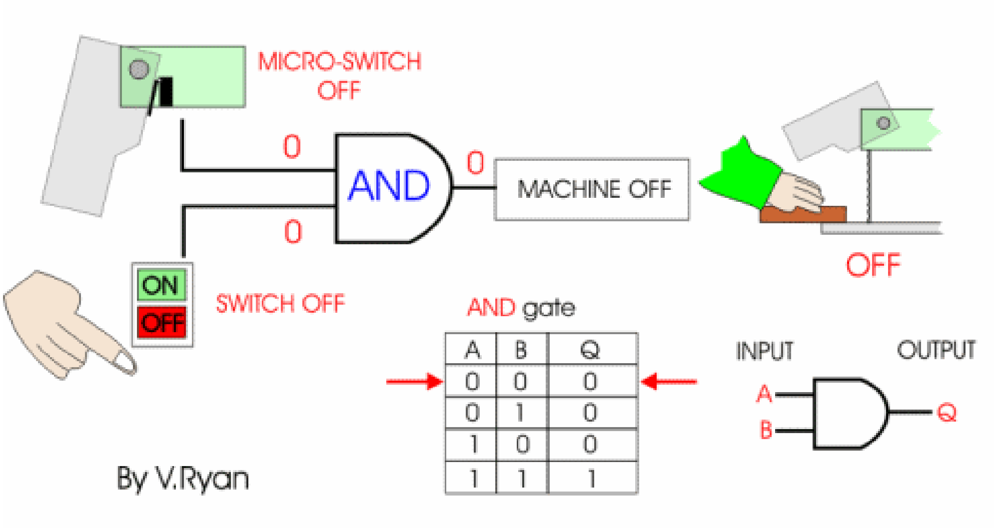
In manufacturing industry safe use of machines is very important. All machines should be set up in such a way that it is impossible for the machine operator to have an accident.

Machine 'A' is unsafe because it can be turned on and used when the guard is out of position, especially if it is operated by a machinist such as Ed the Handyman (website cartoon character). This means that the operator's hands could be seriously injured by the dangerous blade as it cuts the material.

Alternatively, machine ‘B’ has been fitted with a logic circuit. It is designed to ensure that the guard must be in the correct position and the ‘ON’ switch is pressed simultaneously, before the machine will work. This means that the operator must keep his/her spare hand on the switch or electrical power will be cut, stopping the machine working.



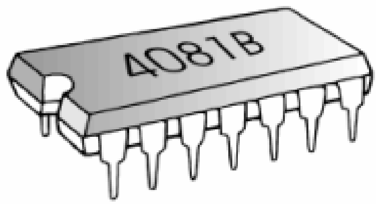
The animation below shows what happens when the micro-switch has been switched 'ON' as the guard is in the correct position. This means that the logic states of both inputs are 1 (true, on, high, up), consequently the output logic state is 1 (true, on, high, up) and the machine works.
Remember, for the AND gate to output 1 both inputs must be 1.



QUESTION:
Draw a series of logic circuits that clearly show the logic states of inputs and outputs as the guard is put in the correct position and the 'ON' switch pressed.

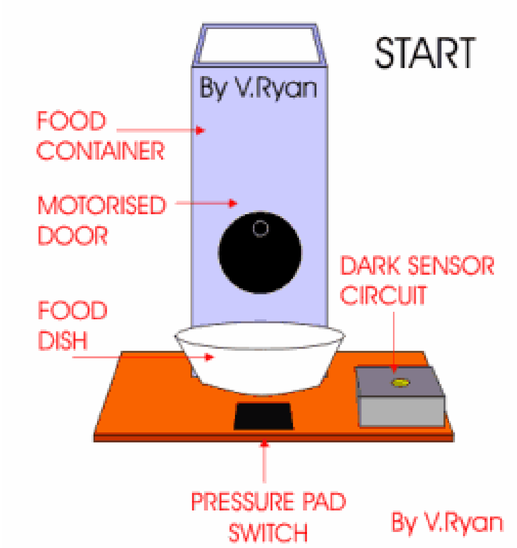
THE 4081B LOGIC CIRCUIT

Logic gates are usually electronic circuits (based on an integrated circuit) and they are used to make simple decisions. A good example of this type of circuit is based on the 4081B integrated circuit (IC) which can be used effectively in school projects.

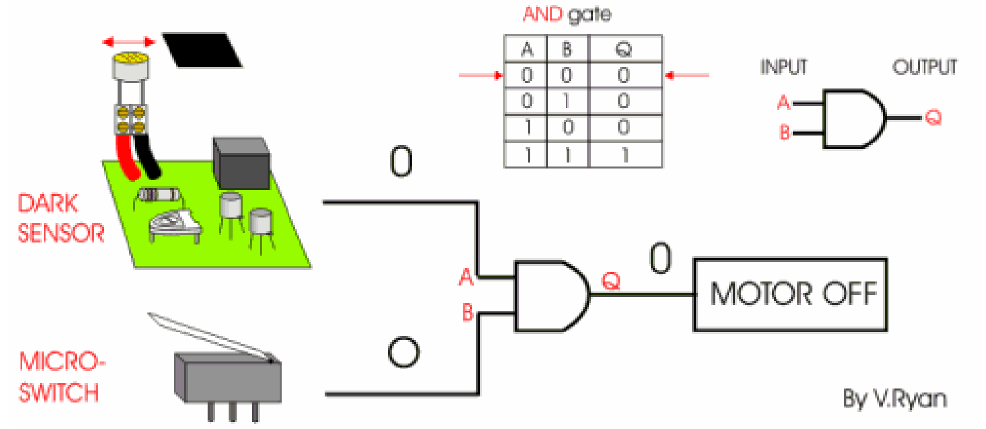


For example, a dog owner wants to build an automatic animal feeder to work at night and when his dog presses a switch (pressure pad). This type of device would automatically feed the dog when the owner is asleep. A diagram of a simple prototype design is shown opposite. The 4081B integrated circuit will detect when the two switches are activated, one by the dog and the other as darkness falls - the motor allows food to be released from a tube. If only one switch is activated, food will not be released.

The logic diagram is shown below. A micro-switch (pressure pad) is used as one input device and a dark sensing circuit as the other. The AND gate has two inputs. If both are activated - the dark sensor and the micro-switch - the logic state of the output changes to high and the motor releases food to the hungry dog.



The diagram below shows that the micro-switch has not been pressed and that it is daylight, consequently the motor is off.



QUESTION:
Draw the diagram of the sensors and logic circuit above but it must clearly show the motor working. Show and explain the logic states of inputs and outputs.

Worksheet: Logic Gates

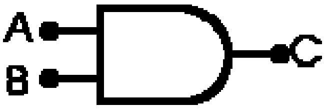
The worksheet on Binary Arithmetic With Switches showed that simple mechanical switches can carry out arithmetic. While switches are easy to understand, and were actually used in some early computers, they are, however, slow. Even though switches are no longer used, computers are still built with these and similar functions. There have been many technologies used in computers to replace switches, including vacuum tubes, transistors and different generations of microchips. However, the basic functions have remained constant. Because the technology and construction details change while the function remains constant, it is very useful to have representation or symbols for the functions, independent of the technology used to implement the function. This set of representations or symbols is called *logic gates*. The *logic* part because they represent classical logical relationships, and *gates* because they can steer signals to different parts of a large circuit. Logic gates are drawn with

- a shape representing the function of the gate
- two input lines on the left-hand side with circles representing contacts, and letters to identify the input
- one output line on the right-hand side, with a circle to represent the contact, and a letter to identify the output

The logical function is made explicit with a truth table. In the truth table, 0 represents false and 1 represents true.

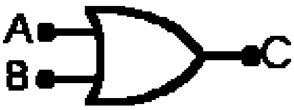
Here are the most important examples. The name of the gate is its logical function that relates the inputs to the outputs. Take an AND gate, for example. Its output is true (1) if input A is true AND input A is true.

A	B	C
0	0	
0	1	
1	0	
1	1	



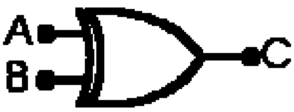
1. AND gate. So named because the output is true if Input A is true and Input B is true. In the switch worksheet, this was the series connection. This function carries out binary multiplication.

A	B	C
0	0	
0	1	
1	0	
1	1	



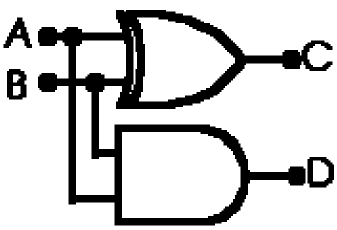
2. OR gate. This is the normal *inclusive* or; the output is true if A is true, or if B is true, or if both are true. Inclusive means that the case where both inputs are true is included in making the output true.

A	B	C
0	0	
0	1	
1	0	
1	1	



3. Exclusive or, or XOR (pronounced "zor"); the output is true if A is true, or if B is true, but not if both are true. This function carries out binary addition, except for the carry in the case of 1 + 1.

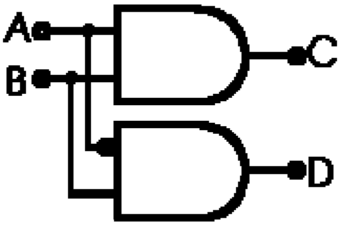
A	B	C	D
0	0		
0	1		
1	0		
1	1		



4. Combining the XOR and AND gates carries out binary addition with the carry.

To illustrate the steering capabilities of gates, we need one more feature, *inversion*. Any signal line (an input or output) can be *inverted* by placing a circle on its connection to the body of the logic gate. The circle inverts the truth of that input. This means that it changes true to false or false to true, before it gets used by the gate itself.

A	B	C	D
0	0		
0	1		
1	0		
1	1		



5. Using the inversion function, the circuit to the right steers input B to output C if input A = 1, or to output D if A = 0.