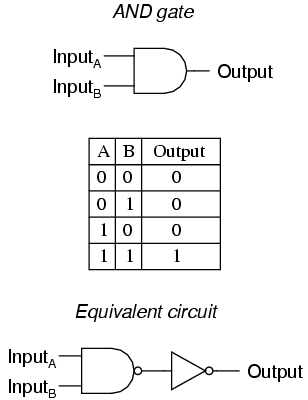
Bit Manipulation:

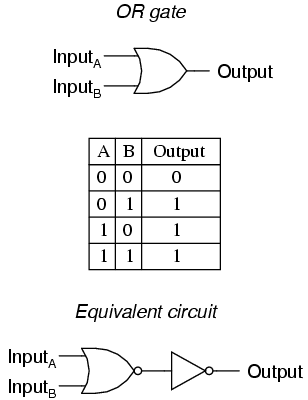
Boolean Gates ---------------------------------------------------------

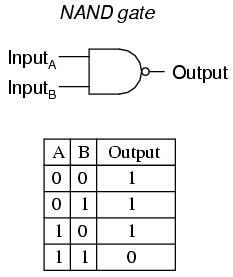
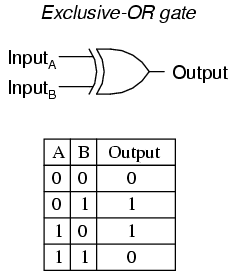
\*1 = True, 0 = false

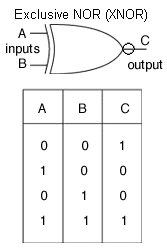
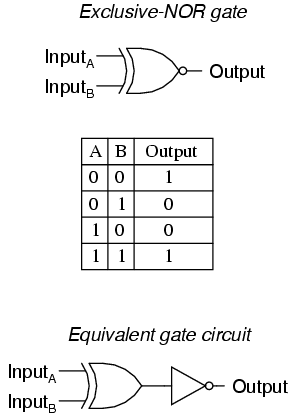
Logic Gates:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | AND | OR | XOR | NAND | NOR | XNOR |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |









Practical App. of Bool. Gates Ex. ---------------------------------

[Hex] [Binary]

AND:

0xABCEDF74 1010 1011 1100 1101 1110 1111 0111 0100

0x987654AB 1001 1000 0111 1110 0100 0100 1010 1011

========================================================================

0x88444420 1000 1000 0100 0100 0100 0100 0010 0000

OR:

0xABCEDF74 1010 1011 1100 1101 1110 1111 0111 0100

0x987654AB 1001 1000 0111 1110 0100 0100 1010 1011

========================================================================

0xBBFFFFFF 1011 1011 1111 1111 1111 1111 1111 1111

XOR:

0xABCEDF74 1010 1011 1100 1101 1110 1111 0111 0100

0x987654AB 1001 1000 0111 1110 0100 0100 1010 1011

========================================================================

0x33B3ABDF 0011 0011 1011 0011 1010 1011 1101 1111

NAND:

0xABCEDF74 1010 1011 1100 1101 1110 1111 0111 0100

0x987654AB 1001 1000 0111 1110 0100 0100 1010 1011

========================================================================

0x88444420 0111 0111 1011 0011 1011 1011 1101 1111

NOR/XNOR:

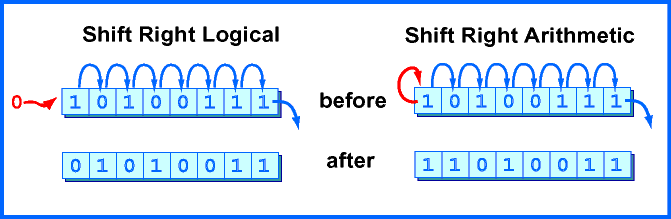
0xABCEDF74 1010 1011 1100 1101 1110 1111 0111 0100

0x987654AB 1001 1000 0111 1110 0100 0100 1010 1011

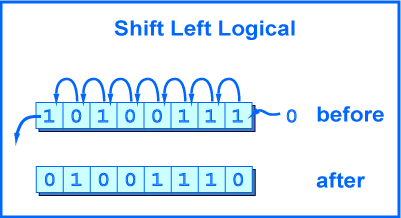
========================================================================

0xBBFFFFFF 1000 1100 0100 1100 0101 0100 0010 0000

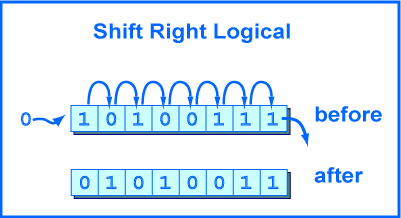
Bit Shifting ---------------------------------------------------------



Logical Shift Left



Logical Shift Right



Bit Shifting Example -------------------------------------------

C++ Code to manipulate bit organization

===============================

#include <iostream>

using namespace std;

Int main(){

bool x[10]={false,false,false,false,true,true,true,true}; //bool =1 byte (4 bits)

bool y[4]={false,true,false,true,true,true,true,false};

for(int i =0; i < 8; i++)

{

cout <<y[i];

}

//Print out 0s and 1s

}

//shifting the bit

// 010 shift to the left and put it in

int v;

for(int i =0; i < 8; i++){

int bit =y[i]; << i // LSL in ARM

v= v | bit // OR Statement

cout << b << ‘\n’;

}

cout << v << “/n’;

}

Assembly code equivalent

=====================

.global \_Z3setPji

\_Z3setPji:

@R0 = address of the bits array

@R1 = position to set

push {R2,R3,R4,R5}

MOV R2,R1

LSR R2,#3 @pos/32 (in words, so really pos/8)

ADD R3,R0,R2 @bits[pos/32]

LDR R4, [R3]

MOV R5, #1

AND R1, #0x1F

LSL R5, R1 @R5 is 1 shift left by bit pos 1000000

OR R4, R5

STR R4,[R3] @write it back to memory

POP {R2,R3,R4,R5}

BX LR