INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

# SUBBYTES Y EXPANSIÓN DE LLAVE

**INTEGRANTES:**
MATA CORTÉS VALERIA **2014630305**
JUÁREZ AMPUDIA CARLOS FRANCISCO **2014630245**

**FECHA:** 06NOVIEMBRE 2015
**GRUPO:** 3CM2
**MATERIA:** CRYPTOGRAPHY
**PROFESOR:** DIAZ SANTIAGO SANDRA

## TEORÍA

# OPERADORES DE BITS

- El operador "&" corresponde a la operación lógica "AND", o en álgebra de Boole al operador "·", compara los bits uno a uno, si ambos son "1" el resultado es "1", en caso contrario "0".

- El operador "^" corresponde a la operación lógica "OR exclusivo", compara los bits uno a uno, si ambos son "1" o ambos son "0", el resultado es "0", en caso contrario "1".

- El operador "|" corresponde a la operación lógica "OR", o en álgebra de Boole al operador "+", compara los bits uno a uno, si uno de ellos es "1" el resultado es "1", en caso contrario "0".

- El operador "~", (se obtiene con la combinación de teclas ALT+126, manteniendo pulsada la tecla "ALT", se pulsan las teclas "1", "2" y "6" del teclado numérico, o bien con la combinación de teclas AltGr+4 seguido de un espacio), corresponde a la operación lógica "NOT", se trata de un operador unitario que invierte el valor de cada bit, si es "1" da como resultado un "0", y si es "0", un "1".

- El operador "<<" realiza un desplazamiento de bits a la izquierda del valor de la izquierda, introduciendo "0" por la derecha, tantas veces como indique el segundo operador; equivale a multiplicar por 2 tantas veces como indique el segundo operando.

- El operador ">>" realiza un desplazamiento de bits a la derecha del valor de la izquierda, introduciendo "0" por la izquierda, tantas veces como indique el segundo operador; equivale a dividir por 2 tantas veces como indique el segundo operando.

# VARIABLES EN HEXADECIMAL Y OCTAL

En términos generales, es común emplear el sistema decimal para la representación de números, sin embargo, en el área de programación a veces es necesario utilizar otra base para manejar datos, como es en este caso el sistema octal y el hexadecimal.

*Para asignar valores:*

- Octal: se debe anteceder un 0 al número que sea desea almacenar. Por ejemplo:

```
Int x=023;
```

- Hexadecimal: se debe anteceder 0x al número que se desea almacenar. Por ejemplo:

```
Int x=0x2C;
```

*Para imprimir valores:*

- Octal: para imprimir una variable en términos de octal se utiliza %o.c. Por ejemplo:

```
Printf("El numero es: %o",x);
```

- Hexadecimal: para imprimir una variable en términos de hexadecimal se utiliza %x. Por ejemplo:

Printf(“El numero es: %x”, y);

Es importante recordar que en la memoria de la computadora estos valores no se almacenan como octales, enteros o hexadecimales, se almacenan siempre como binario ya que es el lenguaje de la computadora.

## FUENTE


- Curso de C, C++, “Operadores de Bits”. http://c.conclase.net/curso/?cap=018
- Charles Lin, “Writing Hex and Octal values in C”. Computer Science. University of Maryland http://www.cs.umd.edu/class/sum2003/cmsc311/Notes/BitOp/hexoctal.html


# EJERCICIO

# CÓDIGO FUENTE

## LAB5.C

lab5.c | operabin.c

```c
int main(){

    int tecla,r,n,x,p;

    do{
        printf("\nOPERACIONES DENTRO DE GF\n");

        printf("\n\nMENU\n1. Multiplicacion x*2^n \n2. Determinar bit en 1 (binario)\n3. Funcion e>
        printf("Opcion: ");
        scanf("%d",&tecla);

        switch(tecla){

            case 1:
                printf("\n\nIngresa el valor de n para 2^n: ");
                scanf("%d",&n);
                printf("\nIngresa el valor de X que multiplicara a 2^n: ");
                scanf("%d",&x);
                r=corrimiento(x,n);
                printf("\n\nEl resultado es: %d\n\n",r);
                break;

            case 2:
                printf("\n\nIngresa un numero: ");
                scanf("%d",&x);
                printf("\nIngresa el bit que quieras verificar: ");
                scanf("%d",&n);
                if((bit_encendido(x,n))==1)
                    printf("\n\nEl bit esta en 1\n\n");
                else
                    printf("\n\nEl bit esta en 0\n\n");
                break;

            case 3:
                printf("\n\nIngresa un numero: ");
                scanf("%d",&x);
                printf("\nIngresa la posicion p de bits: ");
                scanf("%d",&p);
                printf("\nIngresa el numero de bits que se representaran: ");
                scanf("%d",&n);
                r=extract(x,p,n);
                printf("\n\nEl resultado es: %d\n\n",r);
                break;

            case 4:
                printf("\n\nIngresa un numero: ");
                scanf("%d",&x);
                printf("\nIngresa la posicion p de bits: ");
                scanf("%d",&p);
                printf("\nIngresa el numero de bits que se invertiran: ");
                scanf("%d",&n);
                r=invert(x,p,n);
                printf("\n\nEl resultado es: %d\n\n",r);
                break;

            case 5:
                printf("\n\nIngresa el primer numero a sumar: ");
                scanf("%d",&x);
                printf("\nIngresa el segundo numero a sumar: ");
                scanf("%d",&n);
                r=add(x,n);
                printf("\n\nEl resultado en hexadecimal es: %X\n\n",r);
                break;

            case 6:
                printf("\n\nIngresa el primer numero a multiplicar: ");
                scanf("%d",&x);
                printf("\nIngresa el segundo numero a multiplicar: ");
                scanf("%d",&p);
                printf("\nIngresa el numero modulo: ");
                scanf("%d",&n);
                setbuf(stdin,NULL);
                r=multiply(x,p,n);
                printf("\n\nEl resultado en hexadecimal es: %X\n\n",r);
                break;

            case 7:
                exit(0);
                break;

            default:
                printf("\nOpcion erronea\n\n");
                break;
        }

    }while(tecla!=7);

    exit(0);
}
```

## OPERABIN.C

lab5.c | operabin.c

```c
int corrimiento (int x, int n){ // MULTIPLICACION DE x*2^n

    int r=x<<n;
    return r;

}

int bit_encendido (int x, int i){ //REVISAR SI UN BIT ESTA EN 1

    int aux=1<<i;
    if((x&aux)==aux)
        return 1;
    else
        return 0;

}


int extract(int x, int p, int n){

    int aux= x>>p;
    int i,j=1;
    for(i=0;i<n;i++){
        j=j<<1;
        j=j|1;
    }
    return (aux&j);


}

int invert(int x, int p, int n){

    int aux= x>>p;
    int i,j=1;
    for(i=0;i<n;i++){
        j=j<<1;
        j=j|1;
    }
     j=j<<p
     return (aux^j);


}


int add(int p, int q){

    return(p^q);


}


int multiply(int a, int b, int m){

    int c,n,i,l,aux=m;
    while(aux!=0){
        aux>>=1;
        n++;
    }

    for(i=0;i<n;i++){
        if(bit_encendido(b,i)==1){
        l=a<<i;
        c=add(c,l);
        c=c^m;
    }

    return c;

}

```

# FUNCIONAMIENTO

*NOTA: La aplicación fue ejecutada en UBUNTU.*

```
OPERACIONES DENTRO DE GF


MENU
1. Multiplicacion x*2^n
2. Determinar bit en 1 (binario)
3. Funcion extract
4. Funcion invert
5. Suma GF(2^n)
6. Multiplicacion GF(2^n)
7. Salir

Opcion: 2


Ingresa un numero: 45

Ingresa el bit que quieras verificar: 7


El bit esta en 0
```

```
OPERACIONES DENTRO DE GF


MENU
1. Multiplicacion x*2^n
2. Determinar bit en 1 (binario)
3. Funcion extract
4. Funcion invert
5. Suma GF(2^n)
6. Multiplicacion GF(2^n)
7. Salir

Opcion: 3


Ingresa un numero: 35

Ingresa la posicion p de bits: 3

Ingresa el numero de bits que se representaran: 6


El resultado es: 4
```

```
OPERACIONES DENTRO DE GF


MENU
1. Multiplicacion x*2^n
2. Determinar bit en 1 (binario)
3. Funcion extract
4. Funcion invert
5. Suma GF(2^n)
6. Multiplicacion GF(2^n)
7. Salir

Opcion: 4


Ingresa un numero: 245

Ingresa la posicion p de bits: 3

Ingresa el numero de bits que se invertiran: 7


El resultado es: 2022
```

```
OPERACIONES DENTRO DE GF


MENU
1. Multiplicacion x*2^n
2. Determinar bit en 1 (binario)
3. Funcion extract
4. Funcion invert
5. Suma GF(2^n)
6. Multiplicacion GF(2^n)
7. Salir

Opcion: 5


Ingresa el primer numero a sumar: 45

Ingresa el segundo numero a sumar: 56


El resultado es: 15
```

```
OPERACIONES DENTRO DE GF


MENU
1. Multiplicacion x*2^n
2. Determinar bit en 1 (binario)
3. Funcion extract
4. Funcion invert
5. Suma GF(2^n)
6. Multiplicacion GF(2^n)
7. Salir

Opcion: 6


Ingresa el primer numero a multiplicar: 101

Ingresa el segundo numero a multiplicar: 56

Ingresa el numero modulo: 200


El resultado es: 7653
```