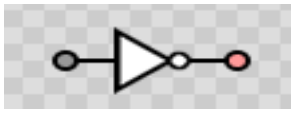
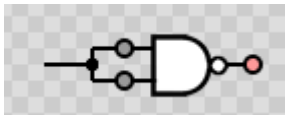
1. **¿Cuál es el objetivo de este proyecto con sus palabras y describa qué debe hacer para desarrollarlo?**

El objetivo de esta El objetivo principal de este proyecto consiste en que comprendamos las compuertas lógicas mediante el uso de otras compuertas, utilizando como punto de partida la compuerta NAND, para lograr este objetivo utilizaremos el software nand2tetris mediante el cual haciendo uso de códigos y a partir de la compuerta NAND simularemos las siguientes compuertas: Not, And, Or, Xor, Mux, Dmux, Not16, And16, Or16, Mux16, Or8way, Mux4way16, Mux8way16, Dmux4way Y Dmux8way

2. **En el sitio web respectivo para esta práctica (no olvide identificarlo correctamente), mencione los principales elementos de entrega de la práctica: códigos realizados (bien explicados), resultados relevantes obtenidos y descripción de los mismos**

COMPUERTA NOT

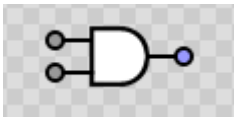
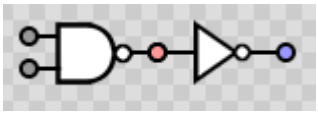




```
CHIP Not {

   IN in;

   OUT out;

   PARTS:

  Nand(a=in,b=in,out=out);

}
```

La compuerta Not básicamente funciona negando cualquier entrada que se le asigne, entonces para representarla usamos una compuerta NAND de una entrada que nos daría un comportamiento similar.

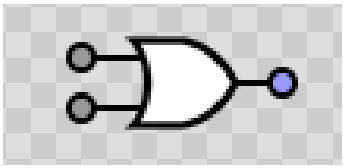
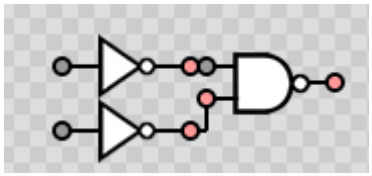
COMPUERTA AND

```
CHIP And {
   IN a, b;
   OUT out;
   PARTS:
   Nand (a=a, b=b, out=out1);
         Not(in=out1, out=out);
}
```

el funcionamiento de la compuerta and es la multiplicación de sus entradas, la cual podemos simulador usando una compuerta NAND y una NOT

COMPUERTA OR

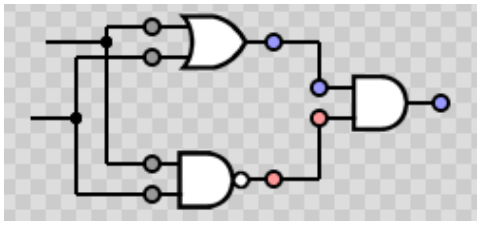



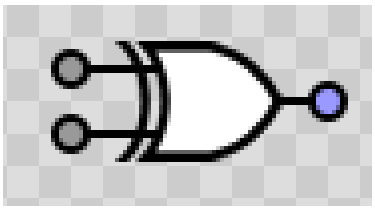
OR

```
CHIP Or {
   IN a, b;
   OUT out;
      PARTS:
      Not(in = a, out = not0);
      Not(in = b, out = not1);
      Nand(a = not0, b = not1, out = out);
}
```

Esta compuerta tendrá su salida en 1 cuando alguna de las 2 entradas es 1 ya que según la lógica booleana la compuerta nos arroja de salida la suma de las 2 entradas, otra forma de representarla seria usando dos compuertas NOT y una compuerta NAND.

COMPUERTA XOR

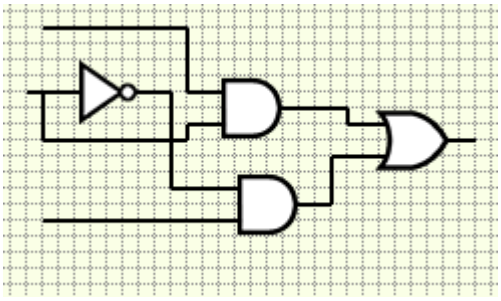


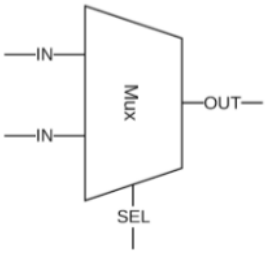


```
CHIP Xor {
    IN a, b;
    OUT out;
    PARTS:
    Nand(a=a, b=b, out=x);
    Or(a=a, b=b, out=y);
    And(a=x, b=y, out=out);
}
```

La compuerta XOR básicamente funciona como una compuerta OR pero negando las salidas, la podemos representar usando una compuerta NAND, una compuerta OR y una compuerta AND.

COMPUERTA MUX





CHIP Mux {

IN a, b, sel;

OUT out;

PARTS:

Not(in=sel, out=sel1);

And(a=a, b=sel1, out=x);

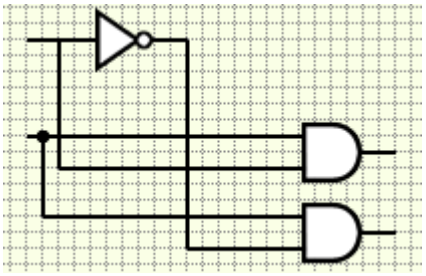
And(a=b, b=sel, out=y);

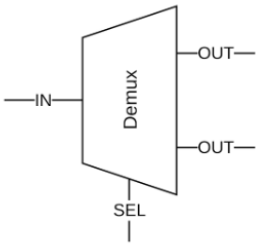
Or(a=x, b=y, out=out);

}

El multiplexor está conformado por dos entradas (a,b) y un selector, si el selector es cero la salida tendrá el mismo valor que la entrada a, si el selector es 1 la salida tendrá el mismo valor que la entrada b, podemos representar el multiplexor con una compuerta NAND, OR y una AND.

COMPUERTA DEMUX

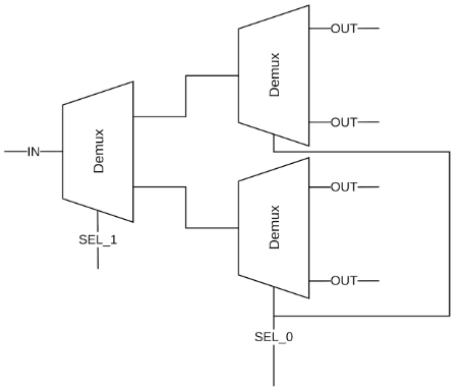


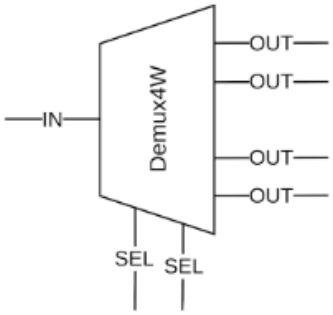



```
CHIP DMux {
   IN in, sel;
   OUT a, b;
   PARTS:
Not(in=sel, out=sel1);
   And(a=in, b=sel1, out=a);
   And(a=in, b=sel, out=b);
}
```

El demultiplexor es pocas palabras que expresen su lógica se basa en el selector, el cual dependiendo si toma un 1 o un 0 afecta las salidas de toda su función. Por ejemplo, cuando el selector toma un nivel bajo todo aquello que salga de la variable a será 0 y en la variable b dependerá del nivel de entrada. Esto aplica viceversa mente, y para esto usamos un par de compuertas AND y una compuerta NOT que dirigen la lógica.

COMPUERTA DEMUX4WAY





PARTS:

CHIP DMux4Way {

IN in, sel[2];

OUT a, b, c, d;

PARTS:

DMux(in=in,sel=sel[1],a=x,b=y);

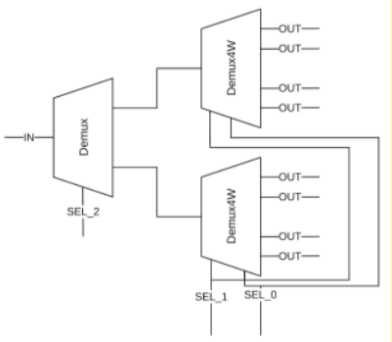
DMux(in=x,sel=sel[0],a=a,b=b);

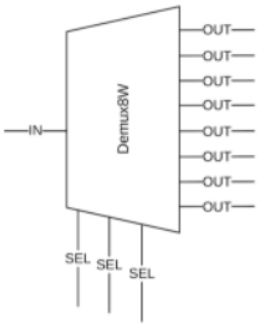
DMux(in=y,sel=sel[0],a=c,b=d);

}

Esta compuerta tiene 4 salidas, utilizara dos selectores, inicialmente usamos una compuerta demux de dos salidas a las cuales en sus salidas colocamos otros 2 demux con dos selectores y así obtendremos las 4 salidas.

COMPUERTA DEMUX8WAY





CHIP DMux8Way {

IN in, sel[3];

OUT a, b, c, d, e, f, g, h;

PARTS:

DMux(in=in,sel=sel[2],a=s,b=t);

DMux(in=s,sel=sel[1],a=x,b=w);

DMux(in=t,sel=sel[1],a=y,b=z);

DMux(in=x,sel=sel[0],a=a,b=b);

DMux(in=w,sel=sel[0],a=c,b=d);

DMux(in=y,sel=sel[0],a=e,b=f);

DMux(in=z,sel=sel[0],a=g,b=h);

}

Esta compuerta utilizara 3 selectores, inicialmente utilizamos un demux de 2 salidas con su selector, luego utilizamos 2 demux4way las cuales contienen 2 selectores para así poder representar la demux8way que tiene 8 salidas y 3 selectores