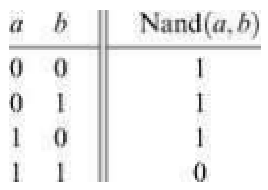
【CA】Day4

Course	Nand to Tetris
Study Date	@February 1, 2022

【Ch1】Boolean Logic

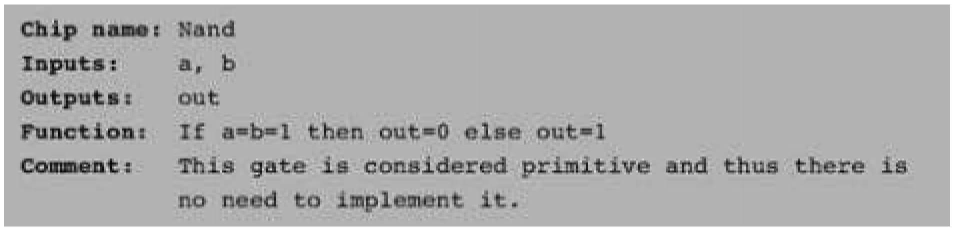
1.2.1 The Nand Gate

The Nand gate is designed to compute the following Boolean function:



a	b	Nand(a,b)
0	0	1
0	1	1
1	0	1
1	1	0

Nand API:

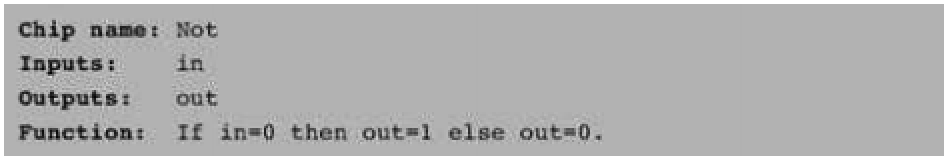


```
Chip name: Nand
Inputs:    a, b
Outputs:   out
Function:  If a=b=1 then out=0 else out=1
Comment:   This gate is considered primitive and thus there is
           no need to implement it.
```

1.2.2 Basic Logic Gates

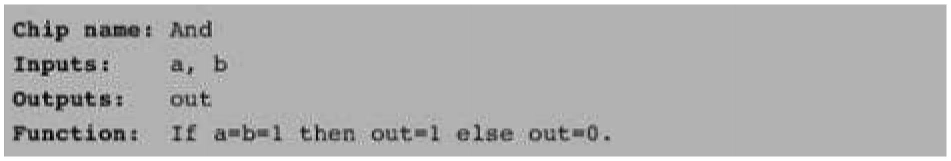
The single-input Not gate, also known as “converter,” converts its input from 0 to 1 and vice versa.

Not API:



```
Chip name: Not
Inputs:    in
Outputs:   out
Function:  If in=0 then out=1 else out=0.
```

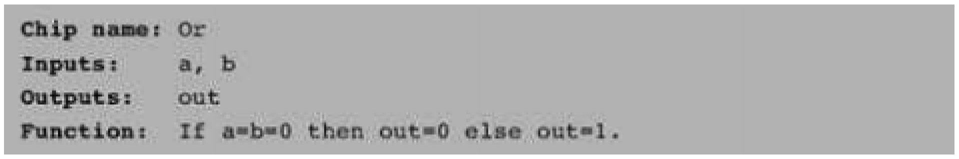
The And function returns 1 when both its inputs are 1, and 0 otherwise.
And API:



```
Chip name: And
Inputs:    a, b
Outputs:   out
Function:  If a=b=1 then out=1 else out=0.
```

The Or function returns 1 when at least one of its inputs is 1, and 0 otherwise:

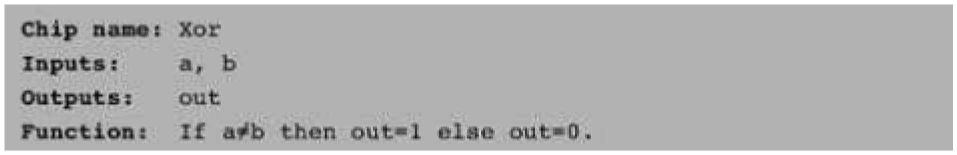
Or API:



```
Chip name: Or
Inputs:    a, b
Outputs:   out
Function:  If a=b=0 then out=0 else out=1.
```

The Xor function, also known as “exclusive or”, returns 1 when its two inputs have opposing values, and 0 otherwise.

Xor API:



```
Chip name: Xor
Inputs:    a, b
Outputs:   out
Function:  If a≠b then out=1 else out=0.
```

A multiplexor is a three-input gate that uses one of the inputs, called “selection bit,” to select and output one of the other two inputs, called “data bits”.

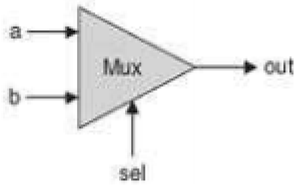
If the selection bit is 0, a is the output. Otherwise, b is the output.

Multiplexor API:

```
Chip name: Mux
Inputs:    a, b, sel
Outputs:   out
Function:  If sel=0 then out=a else out=b.
```

a	b	sel	out
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

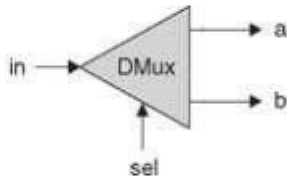
sel	out
0	a
1	b



Demultiplexor API:

```
Chip name: DMux
Inputs:    in, sel
Outputs:   a, b
Function:  If sel=0 then {a=in, b=0} else {a=0, b=in}.
```

sel	a	b
0	in	0
1	0	in



1.2.3 Multi-Bit Versions of Basic Gates

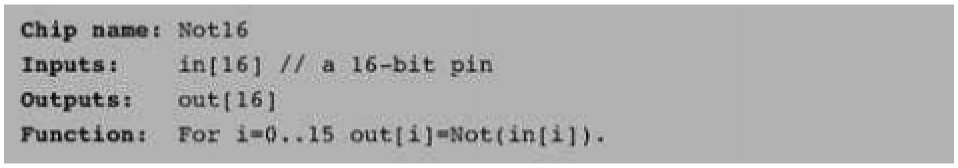
Computer hardware is typically designed to operate on multi-bit arrays called “buses.”

For example, a basic requirement of a 32-bit computer is to be able to compute(bit-wise) an And function on two given 32-bit buses.

To implement this operation, we can build an array of 32 binary And gates, each operating separately on a pair of bits. In order to enclose all this logic in one package, we can encapsulate the gates array in a single chip interface consisting of two 32-bit input buses and one 32-bit output bus.

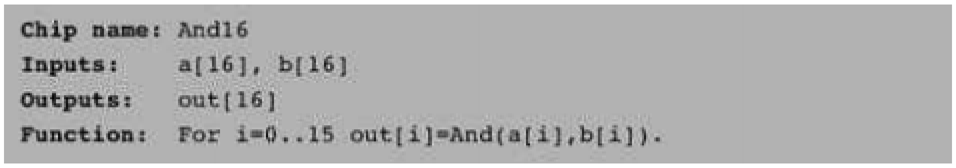
When referring to individual bits in a bus, it is common to use an array syntax. For example, to refer to individual bits in a 16-bit bus named data, we use the notation data[0], data[1], ..., data[15]

An n-bit Note gate applies the Boolean operation Not to every one of the bits in its n-bit input bus:



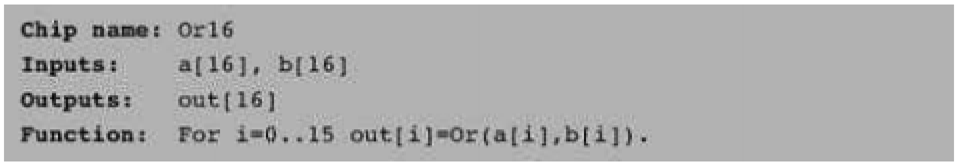
```
Chip name: Not16
Inputs:    in[16] // a 16-bit pin
Outputs:   out[16]
Function:  For i=0..15 out[i]=Not(in[i]).
```

An n-bit And gate applies the Boolean operation And to every one of the n bit-pairs arrayed in its two n-bit input buses:



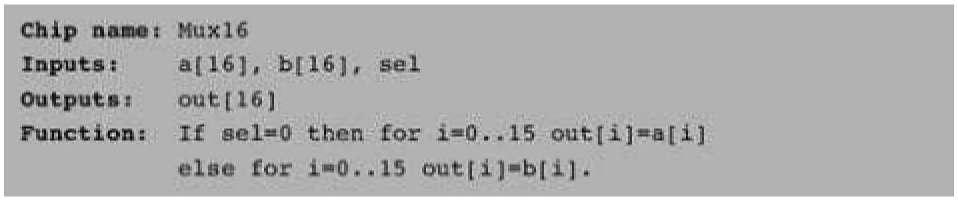
```
Chip name: And16
Inputs:    a[16], b[16]
Outputs:   out[16]
Function:  For i=0..15 out[i]=And(a[i],b[i]).
```

An n-bit Or gate applies the Boolean operation Or to every one of the n bit-pairs arrayed in its two n-bit input buses:



```
Chip name: Or16
Inputs:    a[16], b[16]
Outputs:   out[16]
Function:  For i=0..15 out[i]=Or(a[i],b[i]).
```

An n-bit multiplexor is exactly the same as the binary multiplexor described above, except that the two inputs are each n-bit wide; the selector is a single bit

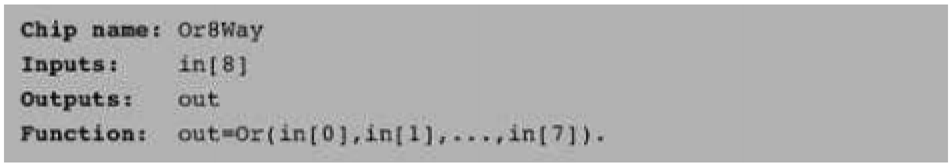


```
Chip name: Mux16
Inputs:    a[16], b[16], sel
Outputs:   out[16]
Function:  If sel=0 then for i=0..15 out[i]=a[i]
           else for i=0..15 out[i]=b[i].
```

1.2.4 Multi-Way Versions of Basic Gates

Many 2-way logic gates that accept two inputs have natural generalization to multi-way variants that accept an arbitrary number of inputs.

An n-way Or gate outputs 1 when at alest one of its n bit inputs is 1, and 0 otherwise.

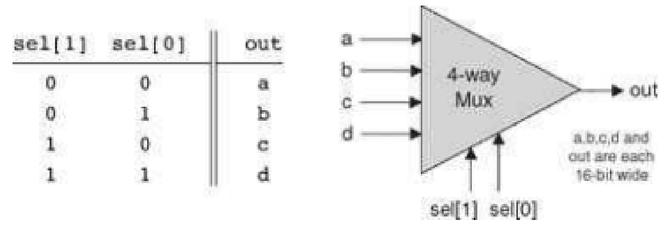


```
Chip name: Or8Way
Inputs:    in[8]
Outputs:   out
Function:  out=Or(in[0],in[1],...,in[7]).
```

Multi-Way/Multi-Bit Multiplexor

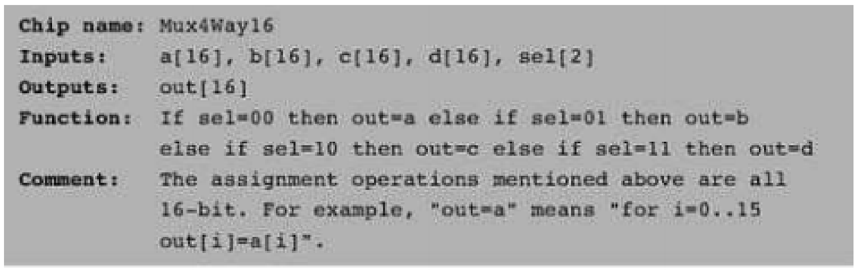
An m-way n-bit multiplexor selects one of m n-bit input buses and outputs it to a single n-bit output bus.

The selection is specified by a set of k control bits, where $k = log_2 m$. The figure below depicts a typical example.

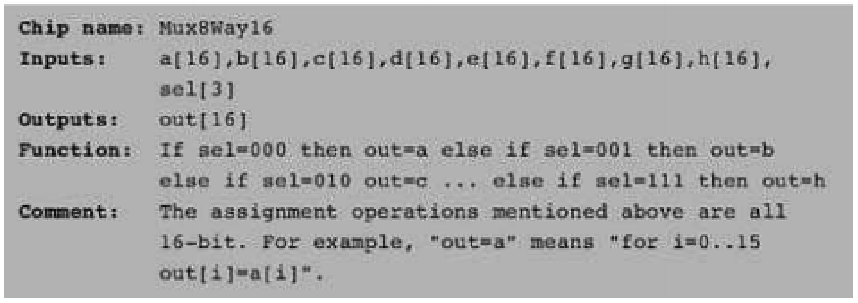


sel[1]	sel[0]	out
0	0	a
0	1	b
1	0	c
1	1	d

Figure 1.10 4-way multiplexor. The width of the input and output buses may vary.



```
Chip name: Mux4Way16
Inputs:    a[16], b[16], c[16], d[16], sel[2]
Outputs:   out[16]
Function:  If sel=00 then out=a else if sel=01 then out=b
           else if sel=10 then out=c else if sel=11 then out=d
Comment:   The assignment operations mentioned above are all
           16-bit. For example, "out=a" means "for i=0..15
           out[i]=a[i]".
```



```
Chip name: Mux8Way16
Inputs:    a[16],b[16],c[16],d[16],e[16],f[16],g[16],h[16],
           sel[3]
Outputs:   out[16]
Function:  If sel=000 then out=a else if sel=001 then out=b
           else if sel=010 out=c ... else if sel=111 then out=h
Comment:   The assignment operations mentioned above are all
           16-bit. For example, "out=a" means "for i=0..15
           out[i]=a[i]".
```

An m-way n-bit multiplexor channels a single n-bit input into one of m possible n-bit outputs. The selection is specified by a set of k control bits, where $k = log_2 m$

sel[1]	sel[0]	a	b	c	d
0	0	in	0	0	0
0	1	0	in	0	0
1	0	0	0	in	0
1	1	0	0	0	in

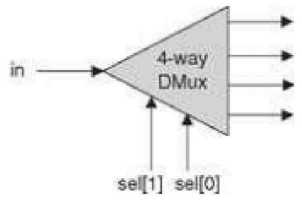


Figure 1.11 4-way demultiplexor.

```
Chip name: DMux4Way
Inputs:    in, sel[2]
Outputs:   a, b, c, d
Function:  If sel=00 then      {a=in, b=c=d=0}
           else if sel=01 then {b=in, a=c=d=0}
           else if sel=10 then {c=in, a=b=d=0}
           else if sel=11 then {d=in, a=b=c=0}.
```

```
Chip name: DMux8Way
Inputs:    in, sel[3]
Outputs:   a, b, c, d, e, f, g, h
Function:  If sel=000 then      {a=in, b=c=d=e=f=g=h=0}
           else if sel=001 then {b=in, a=c=d=e=f=g=h=0}
           else if sel=010 ...
           ...
           else if sel=111 then {h=in, a=b=c=d=e=f=g=0}.
```