



```
//initialize size = 11
```

```
@11
```

```
D=A
```

```
@size
```

```
M=D
```

```
//initialize i = 0
```

```
@i
```

```
M=0
```

```
//initialize arr
```

```
@1
```

```
D=A
```

```
@arr
```

```
M=D
```

```
//initialize array elements 1-11 [-5,-4,-3,-2,-1,0,1,2,3,4,5]
```

```
@5  
D=A  
@R1  
M=-D
```

```
@4  
D=A  
@R2  
M=-D
```

```
@3  
D=A  
@R3  
M=-D
```

```
@2  
D=A  
@R4  
M=-D
```

```
@1  
D=A  
@R5  
M=-1
```

```
@0  
D=A  
@R6  
M=0
```

```
@1  
D=A  
@R7  
M=1
```

```
@2  
D=A  
@R8  
M=D
```

```
@3  
D=A
```

@R9  
M=D

@4  
D=A  
@R10  
M=D

@5  
D=A  
@R11  
M=D

(LOOP)  
@i  
D=M  
@size  
D=M-D  
@END  
D;JEQ //if size - i = 0 jump to end

@i  
D=M  
@arr  
A=M+D //go to address M+D  
D=M //store location

@NEGATIVE  
D;JLE //if value is <= 0 jump to negative

(POSITIVE)  
@i  
M=M+1 //increment i  
@LOOP  
0;JMP

(NEGATIVE)  
@i  
D=M  
@arr  
A=M+D //go to address M+D  
D=M //store location  
M=-D //make value positive

```
@i  
M=M+1 //increment i  
@LOOP  
0;JMP
```

```
(END)  
@END  
0;JMP
```

Josh Clemens  
4-1-23

4 Variable Truth Table

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

K-map

	$\bar{C}\bar{D}$ 00	$\bar{C}D$ 01	$CD$ 11	$C\bar{D}$ 10	
$\bar{A}\bar{B}$ 00	0	1	1	1	$\rightarrow \bar{A}C$
$\bar{A}B$ 01	0	1	1	1	
$AB$ 11	0	1	1	0	$\rightarrow BD$
$A\bar{B}$ 10	1	0	0	1	$\downarrow A\bar{B}D$

$$F = A\bar{B}D + BD + \bar{A}D + \bar{A}C$$

Hardware Simulator (2.5) - C:\Users\JoshC\Desktop\SCHOOL\CS 220 Comp Arch - Assem Lang\midterm\MysteryMidterm\Mystery.hdl

File View Run Help

Chip Name: **Mystery** Time: 0

Input pins		Output pins	
Name	Value	Name	Value
a	1	out	1
b	1		
c	1		
d	1		

Internal pins	
Name	Value
nota	0
notb	0
notc	0
notd	0
anotb	0
w1	0
w2	1
w3	0
w4	0
out1	1
out2	0

**HDL**

```

/**
 * The Mystery Chip!
 * Good luck, friends...
 */
CHIP Mystery {
    IN a, b, c, d;
    OUT out;

    PARTS:
    //build not gates
    Not(in=a, out=nota);
    Not(in=b, out=notb);
    Not(in=c, out=notc);

```

**Script**

```

output;

set a 1,
set b 0,
set c 1,
set d 1,
eval,
output;

set a 1,
set b 1,
set c 0,
set d 0,
eval,
output;

set a 1,
set b 1,
set c 0,
set d 1,
eval,
output;

set a 1,
set b 1,
set c 1,
set d 1,
eval,
output;

```

End of script - Comparison ended successfully

```

/**
 * The Mystery Chip!
 * Good luck, friends...
 */

CHIP Mystery {
    IN a, b, c, d;
    OUT out;

    PARTS:
    //build not gates
    Not(in=a, out=nota);
    Not(in=b, out=notb);
    Not(in=c, out=notc);
    Not(in=d, out=notd);

    //first expression
    And(a=a, b=notb, out=anotb);

```

```
And(a=anotb, b=notd, out=w1);

//second
And(a=b, b=d, out=w2);

//third
And(a=nota, b=d, out=w3);

//fourth
And(a=nota, b=c, out=w4);

//combine with or gates
Or(a=w1, b=w2, out=out1);
Or(a=w3, b=w4, out=out2);
Or(a=out1, b=out2, out=out);
}
```