

A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is light green. They are positioned diagonally, with the blue one partially covering the green one.

# Level 0x0A

Binary Tricks





# Topics

- Events
- Hacker History
- Ones and Zeros



# Ongoing Events

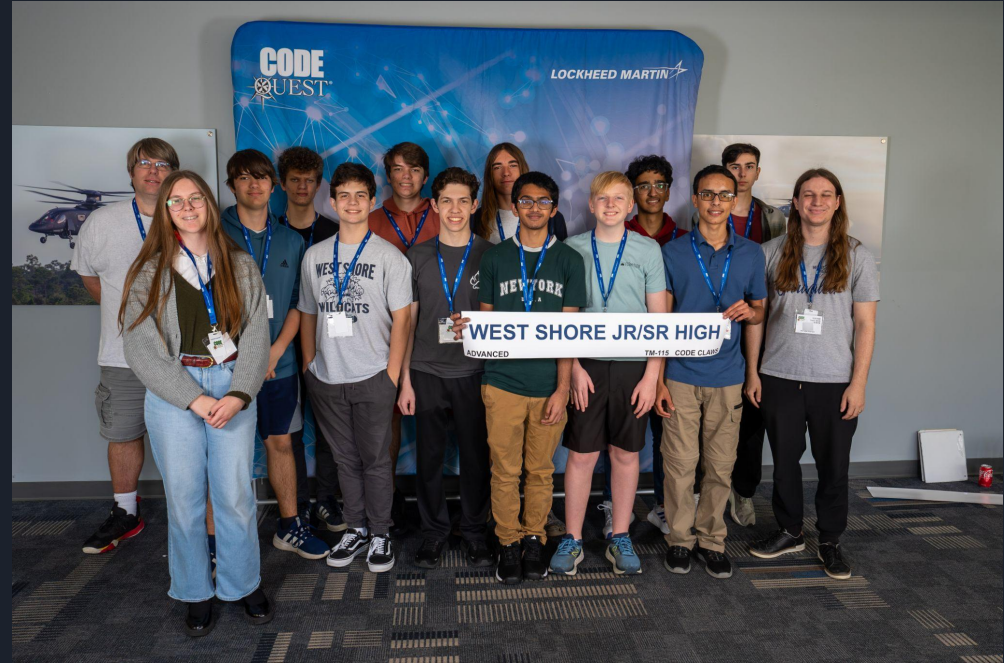
- December hacking contests
  - Advent of Cyber - [TryHackMe.com](https://tryhackme.com)
  - Sans Institute [Holiday Hack Challenge](#) - Less CTF, more game, probably more incident responder and defender focused
  - [Advent of Code](#) - Programming challenges. Used to be 25 2-parters, now reducing down to 12 days of challenges
  - [Pwn.College](#) is going to do an Advent of Pwn





# Cyber Quest and Code Quest

- Code Quest
  - Saturday, Feb 28th
  - Registration Nov 17th -
- Spring Break
  - March 23rd - 27th
- Cyber Quest
  - Saturday, March 28th
  - Registration Jan 5th -





# Code Quest

- Details
  - Saturday, February 28th, 9:00 - 1:30 ish
  - Teams are 2-3 students each, 4 teams max per school
  - 1 computer / laptop per team (one keyboard and one mouse too 😊)
  - No cell phones inside, no cameras (not even outside)
  - No buses, must provide your own transportation (they can't stay on site during the competition)
  - Breakfast and Lunch provided
- What you need to provide me
  - Team members and teams (team names)
  - Birthday (must be 11 yrs and older, middle-school officially allowed)
  - Citizenship (not related to ICE current politics, standard procedure for military contractors)
  - Return 2 Lockheed forms (Liability and Photo Release)
  - Return 1 school permission slip





# Cyber Quest

- Details:
  - Saturday, March 28th, 9:00 - 1:30 ish
  - Teams are 3-5 students each, 4 teams max per school
  - Very large monitors discouraged
  - No cell phones inside, no cameras (not even outside)
  - No buses, must provide your own transportation (they can't stay on site during the competition)
  - Breakfast and Lunch provided
- What you need to provide me
  - Team members and teams (team names)
  - Birthday (must be 14 yrs and older)
  - Citizenship (not related to ICE current politics, standard procedure for military contractors)
  - Return 2 Lockheed forms (Liability and Photo Release)
  - Return 1 school permission slip





# Robert Morris

- Created the first internet worm (not the first worm)
  - Worm = malware that replicates and spreads itself
- Father (also Robert Morris) worked at Bell Labs and NSA
- Grad school student at Cornell
- Morris Worm spread by exploiting
  - Bug in `sendmail` debug mode
  - Buffer overflow in `fingerd` (gives you info about other users)
  - Weak user passwords
- First person to ever be prosecuted by Computer Fraud and Abuse Act
  - Govt: Tens of thousands of infections, \$200-\$50,000 per computer to fix
  - 3 yrs probation, 400 hrs community service, \$10,000 fine
  - Could have been a much harsher sentence per guidelines
  - Leniency because he never intended to cause damage







# Binary Numbers

1 0 1 1 0 1 0 1	=	0xB5	=	181
0 1 1 0 1 1 0 1	=	0x6D	=	109



# Addition

- Online 6502 Assembler and Emulator
  - <https://skilldrick.github.io/easy6502/>

$$\begin{array}{r} \overset{2}{1} \overset{1}{1} \overset{1}{1} \overset{1}{1} \overset{1}{1} \overset{1}{1} \overset{1}{1} \overset{1}{1} \\ + 01101101 \\ \hline 100100010 \end{array} \begin{array}{l} = 0xB5 = 181 \\ = 0x6D = 109 \\ \Rightarrow 0x122 = 290 \end{array}$$

B-bit  
Carry flag

Assemble Run Reset Hexdump Disassemble Notes

LDA #\$b5  
ADC #\$6d

☐ Debugger  
A=\$00 X=\$00 Y=\$00  
SP=\$ff PC=\$0600  
NV-BDIZC  
00110000  
Step Jump to...

Monitor ☐ Start: \$ 0 Length: \$ ff

Preprocessing ...  
Indexing labels ...  
Found 0 labels.  
Assembling code ...  
Code assembled successfully, 4 bytes.



# Let's Try It Out

- Assemble program
- Reset CPU
- Toggle on Debugger
- Step instruction 1 instruction

LDA = Load Accumulator

- Accumulator = 0xB5

The screenshot shows a 6502 assembler/debugger interface. At the top are buttons: Assemble, Run, Reset, Hexdump, Disassemble, and Notes. The main assembly window contains the code:

```
LDA #$b5  
ADC #$6d
```

A green arrow points from the text 'Step instruction 1 instruction' to the 'Run' button. Another green arrow points from the text 'Accumulator = 0xB5' to the 'Debugger' status panel. The status panel shows:

```
Debugger  
A=$b5 X=$00 Y=$00  
SP=$ff PC=$0602  
NV-BDIZC  
10110000  
Step Jump to...
```

At the bottom, there is a 'Monitor' section with checkboxes for 'Start' (set to \$0) and 'Length' (set to \$ff). Below this is a log window showing the following text:

```
Preprocessing ...  
Indexing labels ...  
Found 0 labels.  
Assembling code ...  
Code assembled successfully, 4 bytes.
```

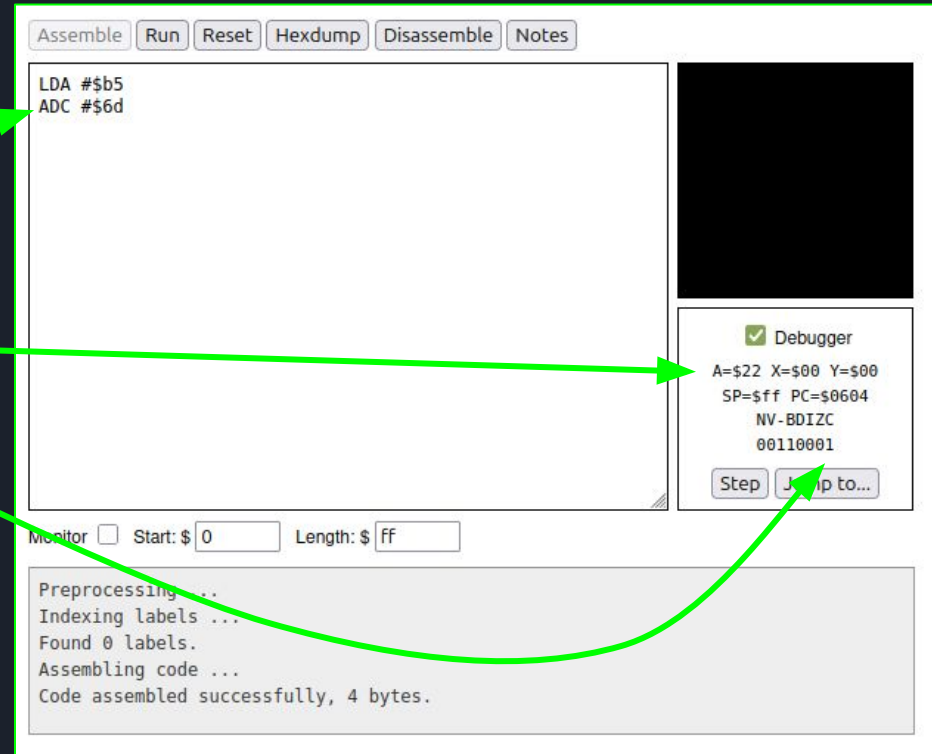


# Let's Try It Out

- Step instruction 1 more

ADC = ADd with Carry

- Accumulator = 0x22
- Carry Flag is set to 1





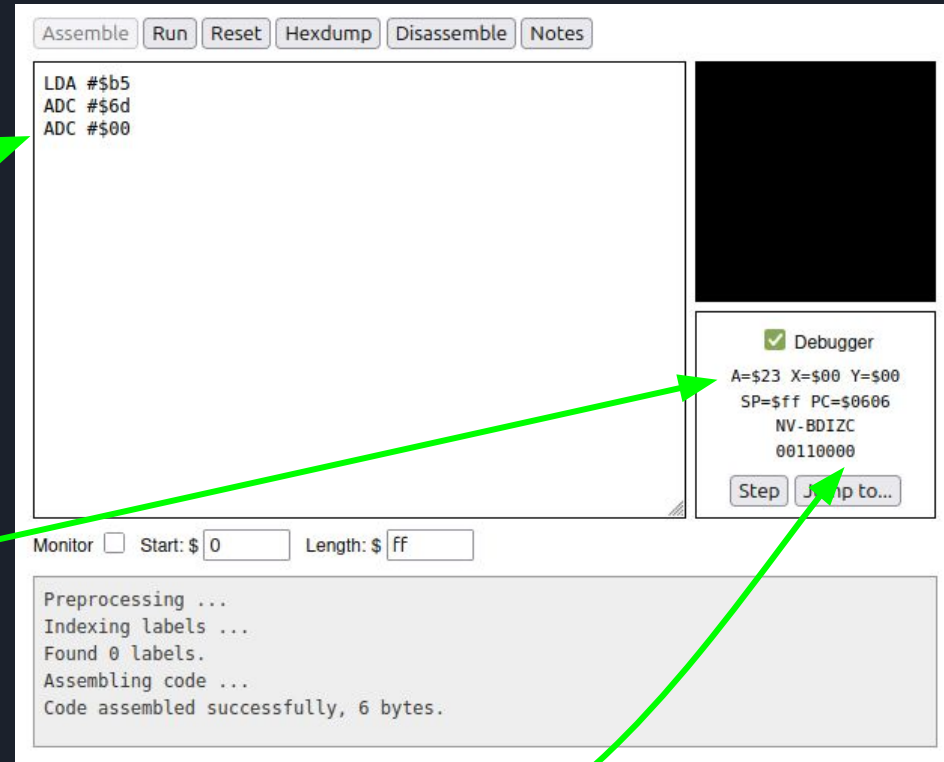
# What does “with Carry” mean?

- Added a new instruction

ADC #\$00 = ADd with Carry

- We add 0 to the accumulator...
- Plus 1 for the Carry Flag

- Accumulator is now 0x23
- Carry flag is now 0

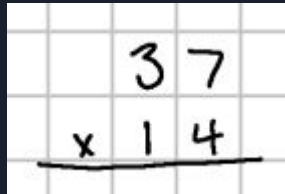






# Multiplication

- How do we do multiplication in base-10 number system?


$$\begin{array}{r} 37 \\ \times 14 \\ \hline \end{array}$$



# Multiplication

- How do we do multiplication in base-10 number system?

$$\begin{array}{r} 37 \\ \times 14 \\ \hline \end{array}$$

$$\begin{array}{r} & 2 \\ & 37 \\ \times & 14 \\ \hline 148 \\ 370 \\ \hline 518 \end{array}$$

$$\begin{array}{r} A B C D \\ \times 1234 \\ \hline \end{array}$$

- What about binary or base-16?





$$\begin{array}{r}
 0x29 \\
 \begin{array}{r}
 \phantom{0x}AB \phantom{00}CD \\
 \times \phantom{0000}34 \\
 \hline
 22BC \phantom{00}A4 \\
 + \phantom{0000}29 \\
 \hline
 22E5 \phantom{00}A4
 \end{array}
 \end{array}$$

$CD \times 34 = 29A4$   
 $AB \times 34 = 22BC$



## Sign Bit?

				0 E		
				A B	C D	
				1 2	3 4	
			<sup>1</sup> x			
		2 2		E 5	A 4	
0 C	0 6			6 A		
C	3 7			4 F	A 4	

$CD \times 12 = 0E6A$   
 $AB \times 12 = C06$

```
>>> hex(0xabcd * 0x1234)
'0xc374fa4'
```



# Negative integers

- Let's propose using the highest / top bit as a sign bit
  - 1 = negative
  - 0 = positive
- Could represent -127 to 127
- What is -0 ???

S	X	X	X	X	X	X	X				
↑	7-bits										
sign bit											
0	0	0	0	0	0	0	0	= 0x00	= 0		
0	1	1	1	1	1	1	1	= 0x7F	= 127		
1	0	0	1	0	0	1	1	= -0x13	= -35		
1	1	1	1	1	1	1	1	= -0x7F	= -127		
1	0	0	0	0	0	0	0	= -0x0	= ???		



# Negative integers

- Let's propose using the highest / top bit as a sign bit



8 ↑ sign bit	X	X	X	X	X	X	X
	7-bits						
0	0	0	0	0	0	0	$= 0_{\text{hex}} = 0$
0	1	1	1	1	1	1	$= 0_{\text{hex}}7\text{F} = 127$
1	0	0	1	0	0	1	$= 0_{\text{hex}}13 = -35$
1	1	1	1	1	1	1	$= -0_{\text{hex}}7\text{F} = -127$
1	0	0	0	0	0	0	$= -0_{\text{hex}}0 = ???$



# Two's Complement

- To invert the sign of a number
  - Invert all the bits
  - Add 1
- That top bit is kind-of a sign bit
  - If it's a 1, it's a negative number
  - But we don't have negative 0
- $-2^N$  to  $2^N-1$     -128 to 127 for 8-bit



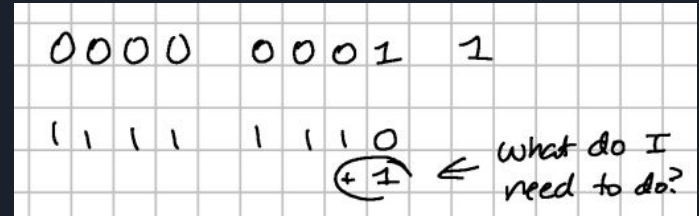
$$\begin{array}{r} 00000101 = 5 \\ \\ 11111010 \\ + 1 \\ \hline 11111011 = 0xFB = -5 \end{array}$$

$$\begin{array}{r} 01111111 = 127 \\ \\ 10000000 \\ + 1 \\ \hline 10000001 = -127 \end{array}$$



# Two's Complement

- Negative 1 is always all 1's / all F's
  - 8 bit => 0b11111111 = 0xFF
  - 16-bit -> 0b1111111111111111 = 0xFFFF
  - And so on...
- If you forget how to do a 2's complement
  - Remember -1 is all F's
  - Remember invert all the bits





# Example Math

Subtraction is essentially adding a negative number...

85  
- 42  
---  
43

$= 0x55$        $42 = 0x2a$

00101010      42

11010101  
+ 1  
---  
11010110      -42

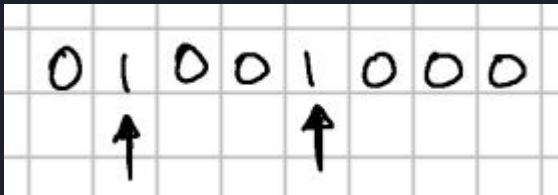
↑  
01010101       $= 0x55 = 81$   
+ 11010110       $= -42$   
---  
00101011       $\rightarrow 0x2B = 43$

throw away



# Bit Counter

- How can I count the bits that are set in a number?
  - POPCNT on x86
- You could loop through all bits (shifting a 1) and test each bit
  - Takes a lot of loops for 64-bit values



```
unsigned int popcount_shift(uint32_t x)
{
    unsigned int count = 0;

    while (x != 0) {
        // Check lowest bit (LSB)
        if (x & 1u) {
            count++;
        }

        // Shift right by 1 bit, discarding LSB
        x >>= 1;
    }

    return count;
}
```



# Try This...

- Subtract 1 from the number
- Bitwise-AND with the original number

01001000	subtract 1
$\begin{array}{r} \phantom{0}1\phantom{0}1\phantom{0}1\phantom{0}1\phantom{0}1\phantom{0}1\phantom{0}1 \\ + \phantom{0}1\phantom{0}1\phantom{0}1\phantom{0}1\phantom{0}1\phantom{0}1\phantom{0}1 \\ \hline 01000111 \end{array}$	$\leftarrow -1$
$\begin{array}{r} 01001000 \\ \& 01000111 \\ \hline 01000000 \end{array}$	and with original Number counted 1 bit

01000000	sub 1
$\begin{array}{r} \phantom{0}1\phantom{0}1\phantom{0}1\phantom{0}1\phantom{0}1\phantom{0}1\phantom{0}1 \\ \hline 00111111 \end{array}$	
$\begin{array}{r} 01000000 \\ \& 00111111 \\ \hline 00000000 \end{array}$	and with orig  2 loops 0 bits set now





# Bit Twiddling Hacks

- <https://graphics.stanford.edu/~seander/bithacks.html>
  - Full of the most clever (evil) math / algorithms you will ever see

## Counting bits set, Brian Kernighan's way

```
unsigned int v; // count the number of bits set in v
unsigned int c; // c accumulates the total bits set in v
for (c = 0; v; c++)
{
    v &= v - 1; // clear the least significant bit set
}
```





# Links

- <https://risky.biz/HTWGO1/> - About Robert Morris case
- <https://skilldrick.github.io/easy6502/>
- <https://graphics.stanford.edu/~seander/bithacks.html>