Coding like its 1977:
Ruby on the Apple][

Colin Fulton (they) – **Duo Security**

Twitter: @PeterQuines
Github: JustColin
Email: JustColin@gmail.com

CODING LIKE IT'S 1977:

RUBY ON THE APPLE][

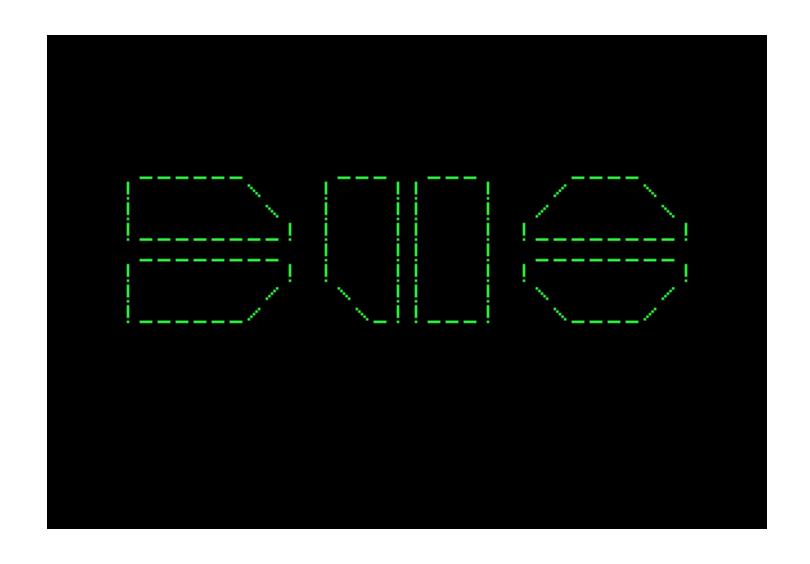COLIN FULTON (THEY) - DUO SECURITY

TWITTER: @PETERQUINES
GITHUB: JUSTCOLIN
EMAIL: JUSTCOLIN@GMAIL.COM

IS THAT WHAT I THINK IT IS?

DID YOU SERIOUSLY FLY HERE WITH A
30 YEAR OLD COMPUTER?

COULDN'T YOU JUST USE AN EMULATOR?

CPU THAT MADE THE HOME COMPUTER: 6502

- 8-BIT (16-BIT ADDRESSES)

- 3 REGISTERS

- FIRST 256 BYTE OF MEMORY HAVE
  SPECIAL (FASTER) INSTRUCTIONS

- 56 INSTRUCTIONS (SORT OF)
  - NO MULTIPLY OR DIVIDE OR SUBTRACT
  - ROTATE RIGHT WAS ORIGINALLY BROKEN

- FAST AND CHEAP (1/6TH THE PRICE)

WRITE CODE LIKE IT'S 1977:

HELLO WORLD: WOZ STYLE

STEP 1: PLAN THE CODE

1. INITIALIZE ONE REGISTER WITH THE STRING LENGTH

2. LOAD A CHARACTER OF THE STRING FROM MEMORY INTO ANOTHER REGISTER

3. STORE THE CHARACTER IN VIDEO MEMORY

4. DECREMENT THE STRING LENGTH REGISTER

5. IF IT ISN'T ZERO, GO BACK TO STEP 2

6. END OF PROGRAM

STEP 2: WRITE THE MACHINE CODE

BASIC: DEMOCRATIZING PROGRAMMING

RUBY WOULD BE SO MUCH BETTER

LET'S COMPILE AND RUN CRUBY!
- CRUBY EXPECTS THINGS...
    LIKE A FILE SYSTEM
    AND UNICODE
        AND ASCII
- CRUBY BINARY IS OVER 3MB IN SIZE!

MRUBY TO THE RESCUE!
- STILL TOO BIG
- C IS TOO HIGH LEVEL

IS ALL HOPE LOST?

```
 /|          nRuby
/ |
/__|
|___|
     FOR THE
     APPLE ][
```

THE JOY OF RUBY, BUT REALLY SMALL
        REALLY REALLY SMALL

   YOU CAN HAVE 256 OBJECTS!

      WRITTEN IN ASSEMBLY

WHY NRUBY? AN "N" IS HALF AN "M"

WILL IT RUN RAILS?

NOPE!

ENOUGH TALKING

SHOW US A DEMO!

A PEEK UNDER THE HOOD

ASSEMBLY?!

LESS SCARY THAN YOU MAY THINK:

- ASSEMBLY DOES VERY LITTLE

- DO TEST DRIVEN DEVELOPMENT (TDD)

LABEL	MNEMONIC	ARGUMENT
	LDY	#12
LOOP	LDA	TEXT, Y
	STA	$0633, Y
	DEY	
	BEQ	LOOP

MODERN APPLE][DEVELOPMENT:

- VIRTUAL][EMULATOR
- MERLIN32 ASSEMBLER
  - BASED ON VINTAGE APPLE][ASSEMBLER
  - CC65 IS ANOTHER OPTION
- APPLECOMMANDER TO CREATE DISK IMAGES
- ADTPRO TO TRANSFER DISK IMAGES TO AN APPLE][USING A SERIAL CABLE

HOW DOES NRUBY MANAGE MEMORY?

```
               MANAGING MEMORY:


                    PAGE 0       PAGE 1      . . .
               --------------------------------------
BYTE  00-0F   |  [         ]  [         ]   . . .
BYTE  10-1F   |  [         ]  [         ]   . . .
BYTE  20-2F   |  [         ]  [         ]   . . .
BYTE  30-3F   |  [         ]  [         ]   . . .
BYTE  40-4F   |  [         ]  [         ]   . . .
BYTE  50-5F   |  [         ]  [         ]   . . .
BYTE  60-6F   |  [         ]  [         ]   . . .
BYTE  70-7F   |  [         ]  [         ]   . . .
BYTE  80-8F   |  [         ]  [         ]   . . .
BYTE  90-9F   |  [         ]  [         ]   . . .
BYTE  A0-AF   |  [         ]  [         ]   . . .
BYTE  B0-BF   |  [         ]  [         ]   . . .
BYTE  C0-CF   |  [         ]  [         ]   . . .
BYTE  D0-DF   |  [         ]  [         ]   . . .
BYTE  E0-EF   |  [         ]  [         ]   . . .
BYTE  F0-FF   |  [         ]  [         ]   . . .
```

```
              MANAGING MEMORY:


                   PAGE 0        PAGE 1       . . .
              -------------------------------------
BYTE  00-0F  |   [    00    ]  [    10    ]    . . .
BYTE  10-1F  |   [    01    ]  [    11    ]    . . .
BYTE  20-2F  |   [    02    ]  [    12    ]    . . .
BYTE  30-3F  |   [    03    ]  [    13    ]    . . .
BYTE  40-4F  |   [    04    ]  [    14    ]    . . .
BYTE  50-5F  |   [    05    ]  [    15    ]    . . .
BYTE  60-6F  |   [    06    ]  [    16    ]    . . .
BYTE  70-7F  |   [    07    ]  [    17    ]    . . .
BYTE  80-8F  |   [    08    ]  [    18    ]    . . .
BYTE  90-9F  |   [    09    ]  [    19    ]    . . .
BYTE  A0-AF  |   [    0A    ]  [    1A    ]    . . .
BYTE  B0-BF  |   [    0B    ]  [    1B    ]    . . .
BYTE  C0-CF  |   [    0C    ]  [    1C    ]    . . .
BYTE  D0-DF  |   [    0D    ]  [    1D    ]    . . .
BYTE  E0-EF  |   [    0E    ]  [    1E    ]    . . .
BYTE  F0-FF  |   [    0F    ]  [    1F    ]    . . .
```

CALCULATING SLOT ADDRESS:

LET'S SAY WE HAVE AN SLOT ID OF CF
SLOT CF STARTS ON BYTE F0 OF PAGE 0C

```
SLOT_TO_ADDR        TAX
                    ASL
                    ASL
                    ASL
                    ASL
                    STA         SLOT_ADDR_BYTE
                    TXA
                    LSR
                    LSR
                    LSR
                    LSR
                    CLC
                    ADC         #>OBJ_SPACE
                    STA         SLOT_ADDR_PAGE
                    RTS
```

```
                WE CAN GET CLEVER:


    PAGE 0         PAGE 1         PAGE 2         ...
 ----------------------------------------------------
   [ 00 ]          . . .          . . .          . . .
   [ 01 ]          . . .          . . .          . . .
   [ 02 ]          . . .          . . .          . . .
   [ 03 ]          . . .          . . .          . . .
   [ 04 ]          . . .          . . .          . . .
   [ 05 ]          . . .          . . .          . . .
   [ 06 ]          . . .          . . .          . . .
   [ 07 ]          . . .          . . .          . . .
   [ 08 ]          . . .          . . .          . . .
   [ 09 ]          . . .          . . .          . . .
   [ 0A ]          . . .          . . .          . . .
   [ 0B ]          . . .          . . .          . . .
   [ 0C ]          . . .          . . .          . . .
   [ 0D ]          . . .          . . .          . . .
   [ 0E ]          . . .          . . .          . . .
   [ 0F ]          . . .          . . .          . . .
```

WE CAN GET CLEVER:

      PAGE 0   PAGE 1   PAGE 2   ...

  [00   ]   [01   ]   [02   ]   ...

WE CAN GET CLEVER:

```
    PAGE 0          PAGE 1          PAGE 2          . . .
------------------------------------------------------------
    [   00   ]      [   01   ]      [   02   ]       . . .
    [   10   ]      [   11   ]      [   12   ]       . . .
    [   20   ]      [   21   ]      [   22   ]       . . .
    [   30   ]      [   31   ]      [   32   ]       . . .
    [   40   ]      [   41   ]      [   42   ]       . . .
    [   50   ]      [   51   ]      [   52   ]       . . .
    [   60   ]      [   61   ]      [   62   ]       . . .
    [   70   ]      [   71   ]      [   72   ]       . . .
    [   80   ]      [   81   ]      [   82   ]       . . .
    [   90   ]      [   91   ]      [   92   ]       . . .
    [   A0   ]      [   A1   ]      [   A2   ]       . . .
    [   B0   ]      [   B1   ]      [   B2   ]       . . .
    [   C0   ]      [   C1   ]      [   C2   ]       . . .
    [   D0   ]      [   D1   ]      [   D2   ]       . . .
    [   E0   ]      [   E1   ]      [   E2   ]       . . .
    [   F0   ]      [   F1   ]      [   F2   ]       . . .
```

```
        NEW SLOT ADDRESS ALGORITHM:

IN THE OLD ALGORYTHM CF TRANSLATES TO
PAGE 0C
BYTE F0

IN THE NEW ALGORYTHM CF TRANSLATES TO
PAGE 0F
BYTE C0

SLOT_TO_ADDR        TAX
                    AND         #$F0
                    STA         SLOT_ADDR_BYTE
                    TXA
                    AND         #$0F
                    CLC
                    ADC         #>OBJ_SPACE
                    STA         SLOT_ADDR_PAGE
                    RTS
```

BEFORE:

```
SLOT_TO_ADDR      TAX
                  ASL
                  ASL
                  ASL
                  ASL
                  STA      SLOT_ADDR_BYTE
                  TXA
                  LSR
                  LSR
                  LSR
                  LSR
                  CLC
                  ADC      #>OBJ_SPACE
                  STA      SLOT_ADDR_PAGE
                  RTS
```

```
                    AFTER:

SLOT_TO_ADDR       TAX
                   AND      #$F0
                   STA      SLOT_ADDR_BYTE
                   TXA
                   AND      #$0F
                   CLC
                   ADC      #>OBJ_SPACE
                   STA      SLOT_ADDR_PAGE
                   RTS
```

WHAT'S NEXT FOR NRUBY?

- FINISH NRUBY V1.0

- RELEASE IT ON GITHUB

- OTHER PROCESSORS, LIKE THE MSP430

- OTHER COMPUTERS RUNNING THE 6502
  LIKE THE NES
  (2KB OF RAM AND 40KB ROM)

WHAT'S NEXT FOR YOU?

- WANT TO LEARN MORE?
  "REVERSE ENGINEERING THE MOS 6502"
  "ULTIMATE COMMODORE 64 TALK"
  AND LOOK UP THE "DEMOSCENE"

- WANT TO SOLVE FUN PUZZLES WITH
  ASSEMBLY PROGRAMMING?
  OR "TIS-100" FROM ZACHTRONICS

- TRY RETRO PROGRAMMING FOR THE
  COMMODORE 64 OR APPLE II

- WANT TO PROGRAM RETRO GAMES?
  TRY THE ATARI 2600 OR THE NES

THANK YOU!

COLIN FULTON (THEY) - DUO SECURITY

    TWITTER: @PETERQUINES
    GITHUB: JUSTCOLIN
    EMAIL: JUSTCOLIN@GMAIL.COM