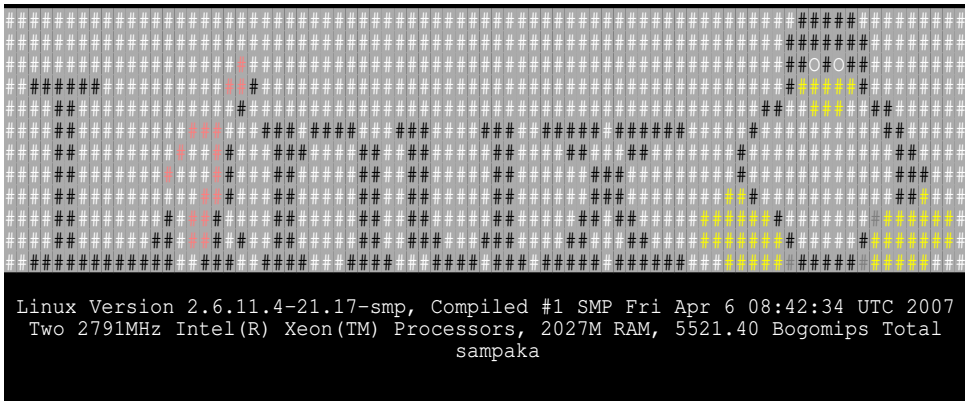
Size Coding on the Apple II

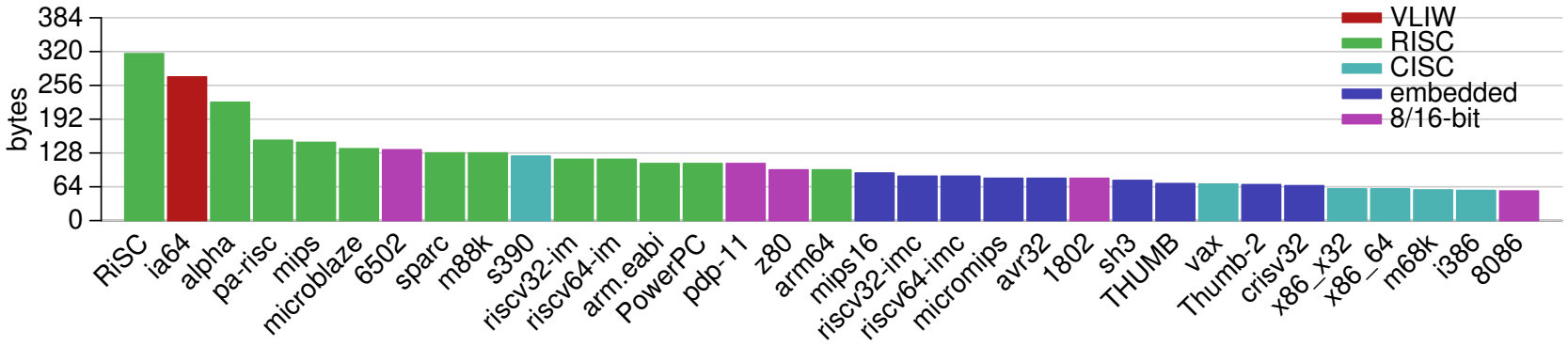
Vince "DEATER" Weaver

vince@deater.net

Lovebyte — 14 March 2021

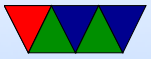
Sizecoding on 30+ Architectures





LZSS decompress size-coded for 30+ architectures

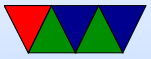
`http://www.deater.net/weave/vmwprod/asm/ll/`



Apple II Background

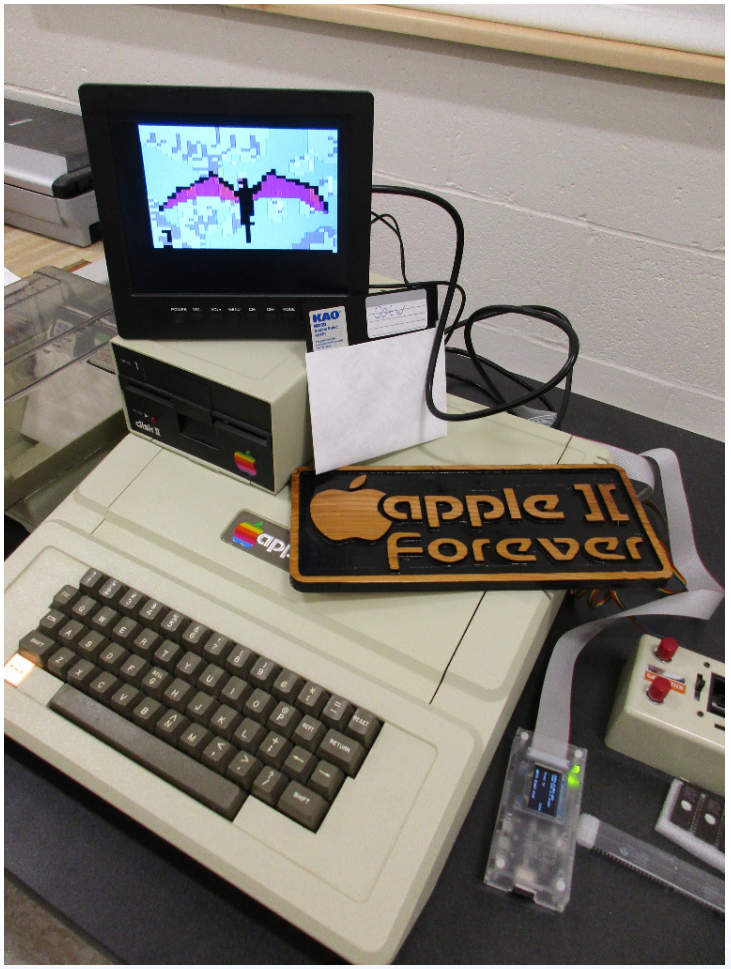
Apple II (1977)

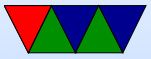
- 1MHz 6502
- 4k-48k RAM
- Discrete 7400 series logic
- cassette
- Bitbang Speaker
- 40x24 text mode
- 40x48 15 color lo-res
- 140x192 6-color hi-res
- need 16k for hi-res graphics
- Integer BASIC in ROM



Apple II+ (1979)

- 1MHz 6502
- 48k RAM
- 140k $5\frac{1}{4}$" floppy
- Applesoft (Microsoft BASIC) in ROM





Apple IIe (1983)

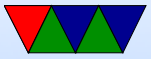
- 1MHz 6502
- 64k-128k RAM
- Lowercase
- 80 column card standard
- If 128k:
  - Double lo-res
    80x48 15 color
  - Double hi-res
    140x192 15-color

Apple IIe Enhanced/Platinum (1985/1987)

- 1MHz 65C02
  adds new, compact opcodes
- 128k RAM
- Mouse-text

Apple IIc (1984)

- 1MHz 65C02
- 128k
- Mostly compatible
- Also a IIc+ with built in accelerator



Apple IIgs (1986)

- 16-bit 2.8MHz 65C816
- 256k - 1M RAM
- Advanced graphics
  Super hi-res
  320x200, 640x200
- Advanced sound
  Ensoniq ES5503
  by same guy who made SID

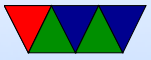
Standard 6502 Optimizations

- Count loops backward
- BIT trick
- Data and Code in zero page
- Use X and Y registers
- See if A/X/Y/flags useful after function call
- Self-modifying code
- Give up and ask qkumba

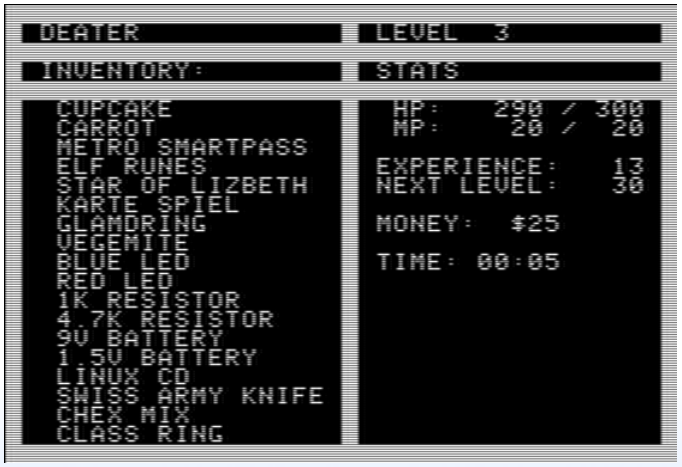
Apple II Specific Optimizations

Apple II+ Memory Map

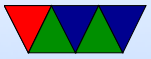
$0000	. . .	$00FF	Zero Page
$0100	. . .	$01FF	Stack
$0200	. . .	$03FF	Mostly free (input buffer, vectors)
$0400	. . .	$07FF	Lo-res/Text Page 1
$0800	. . .	$0BFF	Lo-res/Text Page 2 (BASIC programs load here)
$0C00	. . .	$1FFF	Free
$2000	. . .	$3FFF	Hi-res Page1
$4000	. . .	$5FFF	Hi-res Page2
$6000	. . .	$95FF	Free
$9600	. . .	$BFFF	DOS 3.3 and buffers
$C000	. . .	$CFFF	I/O, soft-switches
$D000	. . .	$F7FF	Applesoft ROM (can be bankswitched on later models)
$F800	. . .	$FFFF	Monitor ROM



Text/Lo-res – 1k at $400



- Lo-res
  - 40x48, 15 colors
    two greys identical
  - Optional 4 lines text
- Text
  - 40x24, no color
  - Normal, flash, inverse
  - II/II+ only uppercase
  - No box/line chars
  - No custom chars

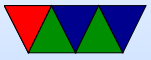


Text/Lo-res Memory Layout

0	$400	. . .	$427
1	$480	. . .	$4A7
2	$500	. . .	$527
3	$580	. . .	$5A7
...			
7	$780	. . .	$7A7
8	$428	. . .	$44F
9	$4A8	. . .	$4CF
10	$528	. . .	$54F
...			
15	$7A8	. . .	$7CF
16	$450	. . .	$477
17	$4D0	. . .	$4F7
18	$550	. . .	$577
...			
23	$7D0	. . .	$7F7

- Note: non-linear
- Avoid screen holes
- Text mode: write ASCII byte
  - Bit 7 set: plain
  - Bit 6 set: flash
  - High two bits clear: inverse
- Lo-res mode: byte two colors
  - Top pixel = low nibble
  - Bottom pixel = high nibble

Colors 5 and 10 (grey) are same*



Text/Graphics – Soft Switches

- Configure Apple II graphics with "soft switches"
- These are memory addresses you read (or sometimes write) to set/clear machine state.
- Takes 3 bytes to set (with BIT, LDA, or STA)

```
SET_GR      =   $C050 ; Enable graphics mode
SET_TEXT    =   $C051 ; Enable text mode
FULLGR      =   $C052 ; Graphics full screen
TEXTGR      =   $C053 ; Graphics with 4-lines text
PAGE1       =   $C054 ; Display PAGE 1
PAGE2       =   $C055 ; Display PAGE 2
LORES       =   $C056 ; Enable LORES graphics
HIRES       =   $C057 ; Enable HIRES graphics
```

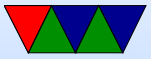

Text/Lo-res – Init Graphics Sample code

- To enable Lo-res manually

```
bit SET_GR   ; $C050 3 bytes
bit LORES    ; $C056 3 bytes
bit FULLGR   ; $C052 3 bytes
bit PAGE1    ; $C054 3 bytes
```

- Instead you could call into ROM, Apple II has some well-defined and stable entry points.

```
jsr SETGR   ; $FB40 3 bytes (same as Applesoft GR)
            ;               set Lo-res graphics, Page1
            ;               split text/graphics, clear to black
```

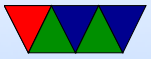



Text/Lores – Plotting Points Fast

```
    lda YPOS              ; load y-coordinate
    and #$FE              ; make even
    tay                   ; put in Y register
    lda gr_offsets,Y      ; get address from lookup
    sta GBASL
    lda gr_offsets+1,Y
    sta GBASH             ; if page-flipping, should add $0/$4

    lda COLOR             ; get color (note: 40x24 faster and smaller!)
                          ; for 40x48 need to load, mask, logical-or
    ldy XPOS              ; load x-coordinate
    sta (GBASL),Y
```

```
gr_offsets:  .word $400,$480,$500,$580,$600,$680,$700,$780
             .word $428,$4a8,$528,$5a8,$628,$6a8,$728,$7a8
             .word $450,$4d0,$550,$5d0,$650,$6d0,$750,$7d0
```

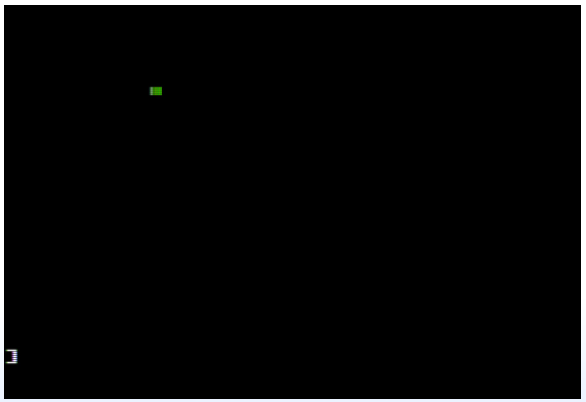



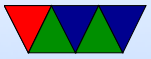
Text/Lo-res – Plotting Points Small (ROM)

- Plot light green point at 10,10

```
lda #$CC     ; load color hi/lo (light green here)
sta COLOR    ; store to zero page $30
ldy #10
lda #10
jsr PLOT     ; $F800 plots at screen location in Y, A
```

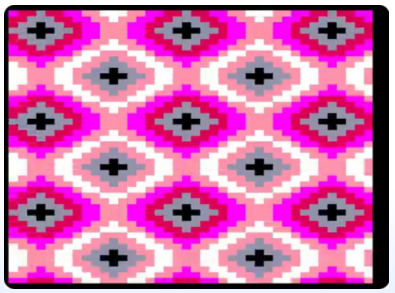
- Slow, does math to calculate address with each point

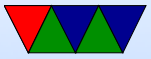




Text/Lo-res – Speed/Size Compromise

- If drawing on same line, no need to-recalc address
- Call `GBASCALC` which will set up `GBASL`/`GBASH`
- (Calling `PLOT` once will also do this)
- Need to be sure `MASK` (ZP $2E) is set to $0F/$F0 for odd/even lines
- Now call `PLOT1` and will plot at current Y-coord, with X-coord in Y register





Text/Lores – Other Routines

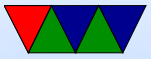
- `HLIN` – horizontal line from `Y` to `$2C` on line `A`
- `VLIN` – vertical line from `A` to `$2D` in column `Y`
- `SCRN` – get color of pixel at `Y`, `A`
- `SETCOL` – set color to `A*17` (top and bottom same)

Finding ROM Routines

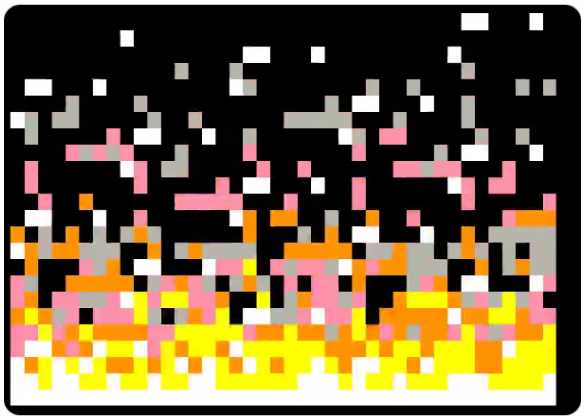
- Apple manuals
- *Apple II Monitors Peeled* book
- ROM disassemblies online:
  - Applesoft:

    `http://www.txbobsc.com/scsc/scdocumentor/`
  - Monitor ROM:

    `https://6502disassembly.com/a2-rom/AutoF8ROM.html`

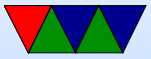


Text/Lo-res – Unusual use of ROM routines



64B Doom Flame Demo

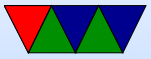
- Algorithm involves scrolling everything up a line
- Instead of open-coding, take advantage of fact text/lo-res are same, use the ROM `SCROLL` routine
- Saved 16 bytes



Page Flipping

- Makes for smooth animations (draw offscreen)
- Unfortunately ROM routines not PAGE aware
- Generally takes 20 bytes

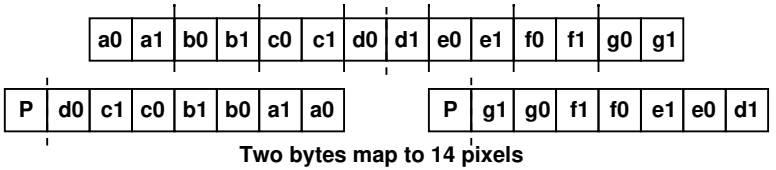
```
    ldx     #0                  ; x already 0
    lda     draw_page_smc+1     ; DRAW_PAGE
    beq     done_page
    inx
done_page:
    ldy     PAGE0,X             ; set display page to PAGE1 or PAGE2
    eor     #$4                 ; flip draw page between $400/$800
    sta     draw_page_smc+1     ; DRAW_PAGE
```

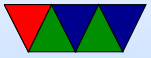
Hi-Res Graphics

line	x=0		x=279
0	$2000	. . .	$2027
1	$2400	. . .	$2427
2	$2800	. . .	$2827
3	$2C00	. . .	$2C27
4	$3000	. . .	$3027
5	$3400	. . .	$3427
6	$3800	. . .	$3827
7	$3C00	. . .	$3C27
8	$2080	. . .	$20A7
9	$2480	. . .	$24A7
...			
15	$3C80	. . .	$3CA7
64	$2028	. . .	$204F
...			
...			
191	$3FD0	. . .	$3FF7

- Even worse than lo-res
- 280x192 on mono monitor
- Color at 140x192 from NTSC artifacts
- 2 bytes for 7 color pixels (div by 7?)



- 00=black 01=color1 10=color2 11=white
- adjacent 00/11 always black/white, fringing
- high bit 1/2 bit shift, change pal, clash

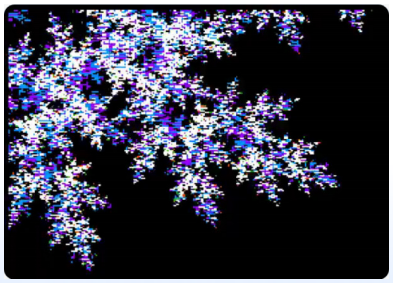


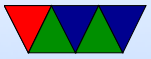
ROM hi-res routines

HGR	$F3E2	set hires/mixed/page1/clear to 0
HGR2	$F3D8	set hires/full/page2/clear to 0
HCLR	$F3F2	clear page in $E6 to 0
BKGND	$F3F6	clear page in $E6 to last color plotted
HPOSN	$F411	move to Xcoord (Y,X) Ycoord (A)
HPLOT0	$F457	plot point at (Y,X), (A)
HGLIN	$F53A	draw line to (A,X), (Y)
HLINRL	$F530	draw relative (A,X), (Y)

Zero page:

$E6	HGR.PAGE	current draw page
$26/$27	GBASL/GBASH	current line address
$E4	HGR.COLOR	color pattern





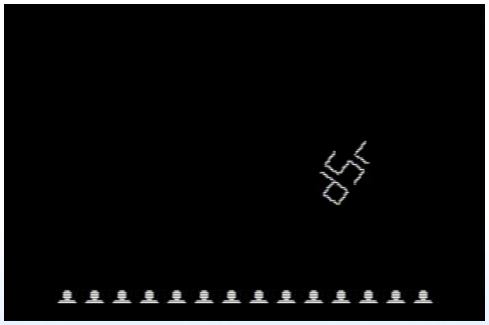
Shape Tables – In-ROM Vector Drawing

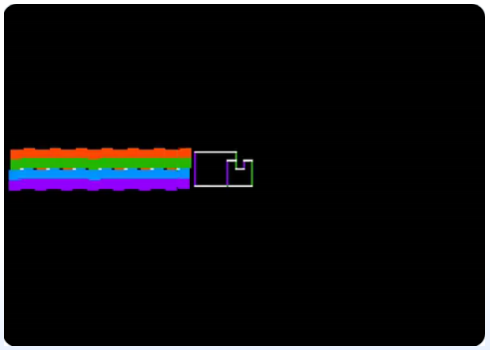
- Rotate, scale, also collision detection
- DRAW/XDRAW at current position (call HPOSN)

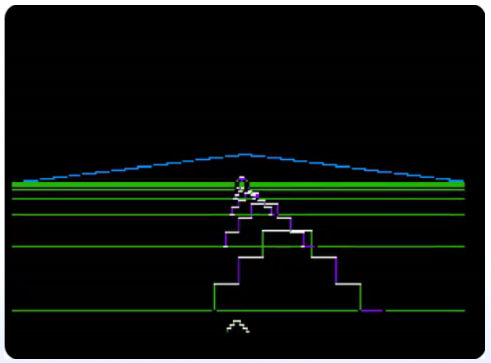
DRAW1	$F605	draw shape SHAPE1/SHAPE2 with rotation=A
XDRAW1	$F661	xor-shape SHAPE1/SHAPE2 with rotation=A

HGR.SHAPE1/HGR.SHAPE2=$1A/$1B, HGR.SCALE=$E7

- Format is packed bytes, draw and move: UP/DN/LT/RT, nodraw and move: NUP/NDN/NLT/NRT

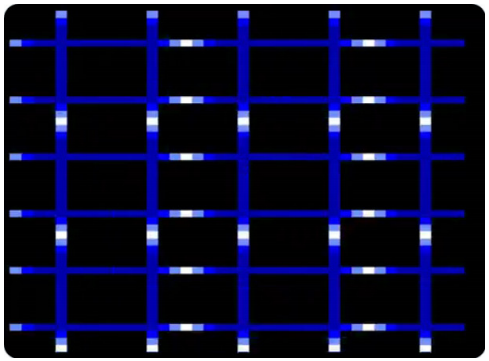


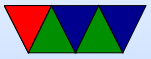




Hi-res – Unusual use of ROM routines

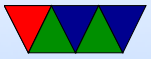
- HGR/HGR2 to set graphics mode then drop to lo-res (can be fewer bytes depending on what mode you want)
- Can use HCLR and BKGND to clear/set memory at $4000 or $6000





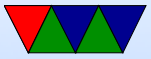
Random Numbers

- ROM Applesoft routine is floating point (and not very random)
- Can make relatively compact (13-byte?) 8-bit linear-feedback PRNG in 6502 asm
- If you need really small, just index into the ROM



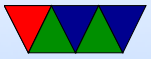
Math: sine/cosine/sqrt

- Applesoft ROM has routines, but complex to use
- Also not really all that fast
- Hard to use them compactly
- Probably best to use lookup tables
- ROM has a table of cos(90*X/16 DEGREES)*$100 - 1 at $F5BA



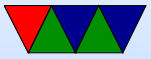
Delay

- No timers on Apple II
- Can call `WAIT` at `$FCA8`
- Delays $\frac{(26+27A+5A^2)}{2}$ microseconds



Reading the Keyboard

- Read value at $C000
  - If positive (bit 7 not set) no keypress
  - If negative, bottom 7 bits is ASCII
- Need to clear keyboard strobe by reading $C010 before next access
- ASCII only, no raw mode
- Only key-down event, no key-up
- Older machines manual REPT, newer auto-repeat



Sound

- Plain speaker at $C030
  - No timer or hardware
  - Access address for click
  - Anything more than clicks or groans need cycle-count
- Mockingboard Expansion
  - AY-3-8910 chips behind 6522 I/O
  - Includes timers
  - Hard to sizecode

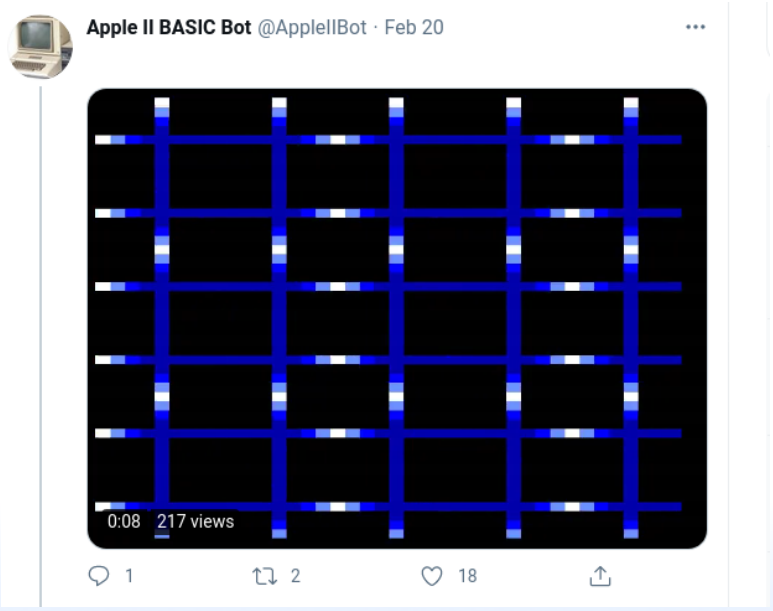
Applesoft BASIC Twitterbot

- `https://appleiibot.com/`
- Run up to 280 chars of BASIC
- Can get 144 byte machine language payload

Replying to @AppleIIBot

{B11}1FORI=0TO140:POKE875+I,4*PEEK(2125+I)-
192+(PEEK(2266+I/3)-35)/4^(I-INT(I/3)*3):NEXT
2&",clZ48jl;=^X-QW<?WX>2(02TQV-?V+>b.i_,?b(Y1f
/:2W0>bd^Xa,k0j_E_B1D^0M9R802,1QePBU3T0XlO;
8=mY8J0N9X5U:21oY_03_2K
/ZSf02X8Q<83nR4iJXb4T9W:JoKI04G0S$/E>3CM;
3C;3S4*+Z6#'`QK=K/%+6*06*)WGTG$%#-Ua._

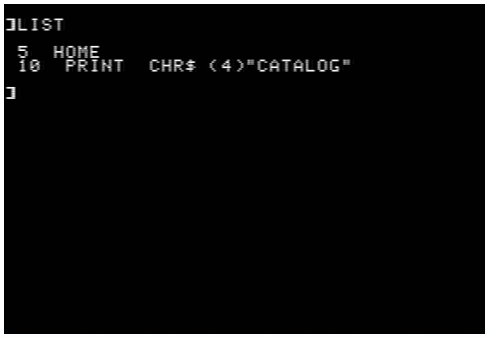
9:53 AM · Feb 20, 2021 · Twitter Web App

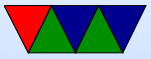


Compo Issues – Autorun

- Write executable to disk image
- Have `HELLO` BASIC program that runs at boot

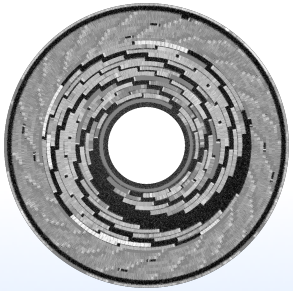
```
10 PRINT CHR$(4)"BRUN FILENAME"
```

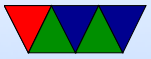




Compo Issues – Header Size

- Machine language programs have 4 bytes of header
  - 16-bit (little endian) address to load at
  - 16-bit (little endian) file size
- These are not loaded into memory
  In fact, should be filesystem metadata
- Most compos let you not count these

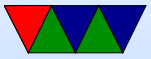




Compo Issues – Returning to DOS/BASIC

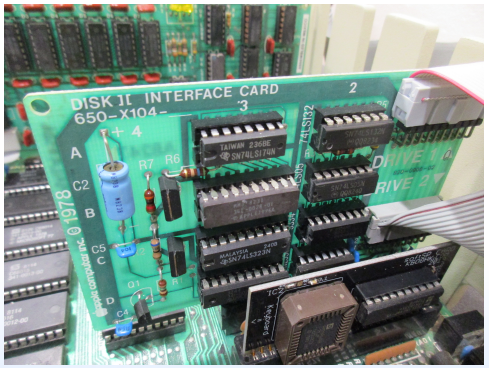
- Larger (256 byte) entries expected to exit
- This is tricky on Apple II
- Try not to stomp on DOS parts of Zero Page
- Try not to stomp on IRQ Vectors at $3D0-$3FF
- Avoid memory above $9600
- Hope for the best and:

```
JMP $3D0
```

Bootsector Demos

- 140k disk – DOS3.3 – 35 tracks, 16 sectors, 256B each
- Track0/Sector0 loaded by disk firmware
  (Note: Disk2 much faster than cassette or C64-1541)
- Loaded to $800 and jumps to $801
- First byte (at $800) is usually 1, means how many sectors to load.
- Turn off drive motor with `LDA $C088,X` where X is slot*16



Double-Lores

- 80x48 15-color mode on Apple IIe and newer
- Complicated to program (ROM routines not much help)
  - Lots of soft-switches, bank-switched memory
  - Horizontal bytes alternate between AUX/MAIN RAM
  - Page-flipping is difficult

Murder Manor by the Crow Cousins

Double Softswitches

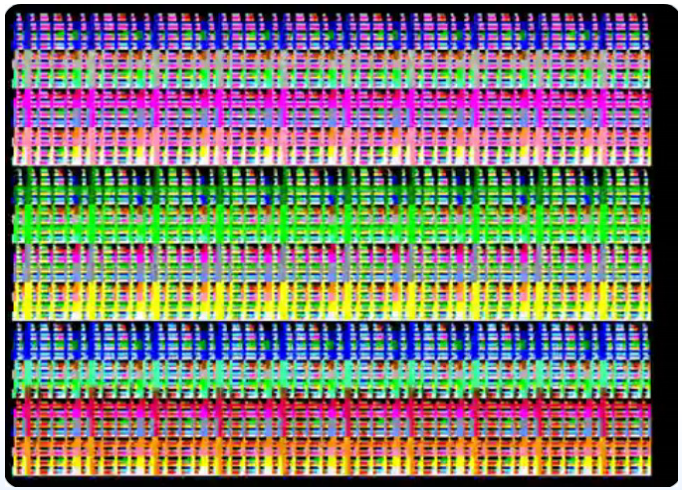
- Bank switch 64k/64k
  Problem, need stack / ZP / code in both locations
- If just doing graphics, have mode where PAGE1/PAGE2 swaps in AUX/MEM so can do double-modes w/o full bank switch
- Note, some of these AUX switches require writes, or even double writes

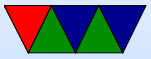
```
sta 80STOREOFF      ; $C000 page2 switches page1/page2
sta 80STOREON       ; $C001 page2 switches main/aux video
sta 80COLON         ; $C00D 80 column/double-res mode
sta ALTCHARSETON    ; $C00F Enable mouse text
sta AN3             ; $C05E set double graphics
```


Double-Hires

- 140x192 15-color mode on Apple IIe and newer
- Even more complicated to program
  - Page-flipping is difficult
  - Colors 4-bits (no palette) but 7-bit shifted/split across banks
  - ROM routines again not much help

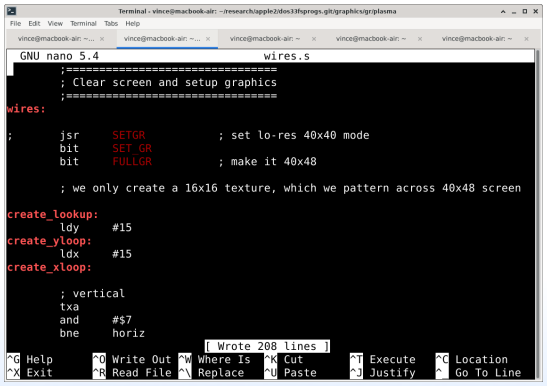
Image from Bill Buckels BMP2DHR

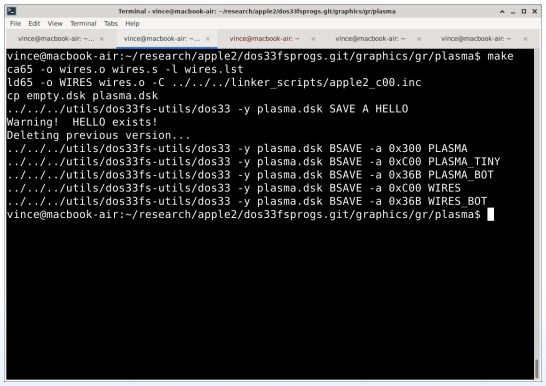


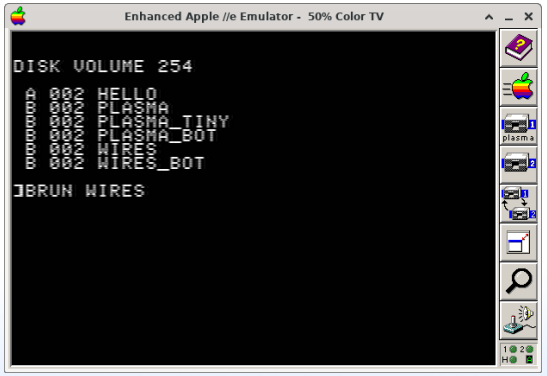


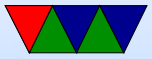
My Coding Setup

- Develop on Linux, nano editor
- Use the ca65 assembler from cc65
- Custom dos33fs tool to save to disk image
- Test in AppleWin emulator (under wine)
- Copy to USB/SD adapter on actual hardware









Questions?

vince@deater.net

More Info and Sourcecode

http://www.deater.net/weave/vmwprod/demos/sizecoding.html

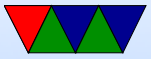


Image Credits

- 1977 Apple II, image by FozzTexx on Wikipedia, CC BY-SA 4.0
- Apple IIe, All About Apple museum official web site via Wikipedia, CC BY-SA 3.0
- Apple IIc, image by Bilby, via Wikipedia, CC BY-SA 3.0
- Apple IIgs, About Apple museum official web site via Wikipedia, CC BY-SA 2.5
- Apple IIe speaker, Apple Rescue of Denver