Sweet 16 for MAC/65 Atari 8bit#

General Information

Author: Steve "Woz" Wozniak
Assembler: Mac/65
Published: 1977

see also Original Sweet 16

This is a port of the Sweet 16 interpreter to the 8bit ATARI.

Diffs#

- The ATARI uses the zero page registers at $E0-$FF instead of $00-$1F
- the "save" and "restore" routines are not in ROM, but in the source

Mac/65 Source

```
1010 ***************************
1020 *                           *
1030 * APPLE II PSEUDO MACHINE *
1040 *        INTERPRETER       *
1050 *                           *
1060 * COPYRIGHT (C) 1977       *
1070 * APPLE COMPUTER INC.    *
1080 *                           *
1090 * STEVE WOZNIAK              *
1100 *                           *
1110 ***************************
1120 * TITLE: SWEET 16 INTERPRETER
1130 ;
1131      *=  $7889
1132 ;
1140 R0L =  $E0
1150 R0H =  $E1
1160 R14H =  $FD
1170 R15L =  $FE
1180 R15H =  $FF
1190 ;
1220 ;
1230 ; PRESERVE 6502 REG CONTENT
1240 ;
1250 SW16 JSR SAVE
1260 ;
1270 ; INIT SWEET16 PC
1280 ; FROM RETURN ADDRESS
1290 ;
1300 SW16A PLA
1310      STA R15L
1320      PLA
1330      STA R15H
1340 ;
1350 ; INTERPRET AND EXECUTE
1360 ; ONE SWEET16 INSTRUCTION
1370 ;
1380 SW16B JSR SW16C
```

```
1390      JMP SW16B
1400 ;
1410 ; INCREMENT SWEET16 PC
1420 ; FOR FETCH
1430 ;
1440 SW16C INC R15L
1450      BNE SW16D
1460      INC R15H
1470 ;
1480 ; COMMON HIG BZTE FOR ALL
1490 ; ROUTINES, PUSH ON STACK FOR
1500 ; RTS
1510 SW16D LDA  # >SET
1520      PHA
1530 ;
1540 ; FETCH INSTRUCTION
1550 ;
1560      LDY #$00
1570      LDA (R15L),Y
1580 ;
1590 ; MASK REGISTER SPECIFICATION
1600 ;
1610      AND #$0F
1620 ;
1630 ; DOUBLE FOR TWO BYTE REGISTERS
1640 ;
1650      ASL A
1660 ;
1670 ; TO X REGISTER FOR INDEXING
1680 ;
1690      TAX
1700      LSR A
1710 ;
1720 ; NOW HAVE OPCODE
1730 ;
1740      EOR (R15L),Y
1750 ;
1760 ; IF ZERO THEN NO REG OPCODE
1770 ;
1780      BEQ TOBR
1790 ;
1800 ; INDICATE PRIOR RESULT REG
1810 ;
1820      STX R14H
1830 ;
1840 ; OPCODE * 2 TO LSB'S
1850 ;
1860      LSR A
1870      LSR A
1880      LSR A
1890 ;
1900 ; TO Y REGISTER FOR INDEXING
1910 ;
1920      TAY
1930 ;
1940 ; LOW ORDER ADR BYTE ON STACK
1950 ;
1960      LDA OPTBL-2,Y
1970      PHA
```

```
1980 ;
1990 ; GOTO REG-OPCODE ROUTINE
2000 ;
2010      RTS
2020 ;
2030 ; INCREMENT PC
2040 ;
2050 TOBR
2051      INC R15L
2060      BNE TOBR2
2070      INC R15H
2080 ;
2090 ; LOW ORDER ADR BYTE ONTO
2100 ; STACK FOR NON-REG OPCODE
2110 ;
2120 TOBR2 LDA BRTBL,X
2130      PHA
2140 ;
2150 ; PRIOR RESULT REG INDEX
2160 ;
2170      LDA R14H
2180 ;
2190 ; PREPARE CARRY FOR BC. BNC.
2200 ;
2210      LSR A
2220 ;
2230 ; GOTO NON-REG OPCODE ROUTINE
2240 ;
2250      RTS
2260 ;
2270 ; RETURN TO 6502 MODE ROUTINE
2280 ;
2290 ; POP RETURN ADDRESS
2300 ;
2310 RTNZ PLA
2320      PLA
2330 ;
2340 ; RESTORE 6502 REG CONTENT
2350 ;
2360      JSR RESTORE
2370 ;
2380 ; RETURN TO 6502 CODE VIA PC
2390 ;
2400      JMP (R15L)
2410 ;
2420 ; SET REGISTER ROUTINE
2430 ;
2440 ; GET HIGH BYTE OF CONSTANT
2450 ;
2460 SETZ LDA (R15L),Y
2470      STA R0H,X
2480      DEY
2490 ;
2500 ; GET LOW BYTE OF CONSTANT
2510 ;
2520      LDA (R15L),Y
2530      STA R0L,X
2540 ;
2550 ; Y REG CONTAINS 1
```

```
2560 ;
2570      TYA
2580 ;
2590 ; ADD 2 TO PC
2600 ;
2610      SEC
2620      ADC R15L
2630      STA R15L
2640      BCC SET2
2650      INC R15H
2660 SET2 RTS
2670 ;
2680 ; OPCODE TABLE
2690 ;
2700 OPTBL .BYTE  <SET-1 ; 1X
2710 BRTBL .BYTE  <RTN-1 ; 0
2720      .BYTE  <LD-1 ; 2X
2730      .BYTE  <BR-1 ; 1
2740      .BYTE  <ST-1 ; 3X
2750      .BYTE  <BNC-1 ; 2
2760      .BYTE  <LDAT-1 ; 4X
2770      .BYTE  <BC-1 ; 3
2780      .BYTE  <STAT-1 ; 5X
2790      .BYTE  <BP-1 ; 4
2800      .BYTE  <LDDAT-1 ; 6X
2810      .BYTE  <BM-1 ; 5
2820      .BYTE  <STDAT-1 ; 7X
2830      .BYTE  <BZ-1 ; 6
2840      .BYTE  <POP-1 ; 8X
2850      .BYTE  <BNZ-1 ; 7
2860      .BYTE  <STPAT-1 ; 9X
2870      .BYTE  <BM1-1 ; 8
2880      .BYTE  <ADD-1 ; AX
2890      .BYTE  <BNM1-1 ; 9
2900      .BYTE  <SUB-1 ; BX
2910      .BYTE  <BK-1 ; A
2920      .BYTE  <POPD-1 ; CX
2930      .BYTE  <RS-1 ; B
2940      .BYTE  <CPR-1 ; DX
2950      .BYTE  <BS-1 ; C
2960      .BYTE  <INR-1 ; EX
2970      .BYTE  <NUL-1 ; D
2980      .BYTE  <DCR-1 ; FX
2990      .BYTE  <NUL-1 ; E
3000      .BYTE  <NUL-1 ; UNUSED
3010      .BYTE  <NUL-1 ; F
3020 ;
3030 ; THE FOLLOWING CODE MUST
3040 ; BE CONTAINED IN A SINGLE PAGE
3050 ;
3060 SET
3061      JMP SETZ   ; ALWAYS
3070 LD
3071      LDA R0L,X
3080 BK  =  LD+1
3090      STA R0L
3100 ;
3110 ; MOV RX TO R0
3120 ;
```

```
3130      LDA R0H,X
3140      STA R0H
3150      RTS
3160 ;
3170 ST  LDA R0L
3180 ;
3190 ; MOV R0 TO RX
3200 ;
3210      STA R0L,X
3220      LDA R0H
3230      STA R0H,X
3240      RTS
3250 ;
3260 ; STORE BYTE INDEIRECT
3270 ;
3280 STAT LDA R0L
3290 STAT2 STA (R0L,X)
3300      LDY #$00
3310 ;
3320 ; INDICATE R0 IS RESULT NEG
3330 ;
3340 STAT3 STY R14H
3350 INR INC R0L,X
3360      BNE INR2   ; INC RX
3370      INC R0H,X
3380 INR2 RTS
3390 ;
3400 ; LOAD INDIRECT (RX) TO R0
3410 ;
3420 LDAT LDA (R0L,X)
3430      STA R0L
3440 ;
3450 ; ZERO HIGH ORDER R0 BYTE
3460 ;
3470      LDY #$00
3480      STY R0H
3490      BEQ STAT3 ; ALWAYS
3500 ;
3510 ; HIGH ORDER BYTE = 0
3520 ;
3530 POP LDY #$00
3540      BEQ POP2   ; ALWAYS
3550 POPD JSR DCR    ; DECR RX
3560 ;
3570 ; POP HIGH ORDER BYTE @RX
3580 ; AND SAV IN Y REG
3590 ;
3600      LDA (R0L,X)
3610      TAY
3620 POP2 JSR DCR    ; DECR RX
3630 ;
3640 ; LOW ORDER BYTE TO R0
3650 ;
3660      LDA (R0L,X)
3670      STA R0L
3680      STY R0H
3690 ;
3700 ; INDICATE R0 AS LAST RES. REG
3710 ;
```

```
3720 POP3 LDY #$00
3730      STY R14H
3740      RTS
3750 ;
3760 ; LOW ORDER BYTE TO R0, INC RX
3770 ;
3780 LDDAT JSR LDAT
3790 ;
3800 ; HIGH ORDER BYTE TO R0
3810      LDA (R0L,X)
3820      STA R0H
3830      JMP INR     ; INC RX
3840 ;
3850 ; STORE INDIRECT LOW ORDER
3860 ; BYTE AND INC RX THEN
3870 ; STORE HIGH  ORDER BYTE,
3880 ; INC RX AND RETURN
3890 ;
3900 STDAT JSR LDAT
3910      LDA R0H
3920      STA (R0L,X)
3930      JMP INR
3940 STPAT JSR DCR  ; DEC RX
3950      LDA R0L
3960 ;
3970 ; STORE R0 LOW BYTE @RX
3980 ;
3990      STA (R0L,X)
4000 ;
4010 ; INDICATE R0 AS LAST RES REG
4020 ;
4030      JMP POP3
4040 ;
4050 DCR LDA R0L,X
4060      BNE DCR2   ; DEC RX
4070      DEC R0H,X
4080 DCR2 DEC R0L,X
4090      RTS
4100 ;
4110 ; RESULT TO R0
4120 ;
4130 SUB LDY #$00
4140 ;
4150 ; NOTE Y REG = 13 * 2 FOR CPR
4160 ;
4170 CPR SEC
4180      LDA R0L
4190      SBC R0L,X
4200      STA R0L,Y ; RY=R0-RX
4210      LDA R0H
4220      SBC R0H,X
4230 SUB2 STA R0H,Y
4240 ;
4250 ; LAST RESULT REG * 2
4260 ;
4270      TYA
4280 ;
4290 ; CARRY TO LSB
4300 ;
```

```
4310      ADC #$00
4320      STA R14H
4330      RTS
4340 ;
4350 ADD LDA R0L
4360      ADC R0L,X
4370      STA R0L     ; R0=RX+R0
4380      LDA R0H
4390      ADC R0H,X
4400 ;
4410 ; R0 FOR RESULT
4420 ;
4430      LDY #$00
4440 ; FINISH ADD
4450      BEQ SUB2
4460 ;
4470 ; NOTE X REG IS 12 * 2 !
4480 ;
4490 BS  LDA R15L
4500 ;
4510 ; PUSH LOW PC BYTE VIA R12
4520 ;
4530      JSR STAT2
4540      LDA R15H
4550 ;
4560 ; PUSH HIGH PC BYTE
4570 ;
4580      JSR STAT2
4590 BR  CLC
4600 BNC BCS BNC2    ; NO CARRY TEST
4610 ;
4620 ; DISPLACEMENT BYTE
4630 ;
4640 BR1 LDA (R15L),Y
4650      BPL BR2
4660      DEY
4670 ;
4680 ; ADD TO PC
4690 ;
4700 BR2 ADC R15L
4710      STA R15L
4720      TYA
4730      ADC R15H
4740      STA R15H
4750 BNC2 RTS
4760 BC  BCS BR
4770      RTS
4780 ;
4790 ; DOUBLE RESULT REG INDEX
4800 ;
4810 BP  ASL A
4820 ;
4830 ; TO X REG FOR INDEX
4840 ;
4850      TAX
4860 ; TEST FOR PLUS, BRANCH IF SO
4870 ;
4880      LDA R0H,X
4890      BPL BR1
```

```
4900      RTS
4910 ;
4920 ; DOUBLE RESULT REG INDEX
4930 ;
4940 BM  ASL A
4950      TAX
4960 ;
4970 ; TEST FOR MINUS
4980 ;
4990      LDA R0H,X
5000      BMI BR1
5010      RTS
5020 ;
5030 ; DOUBLE RESULT REG INDEX
5040 ;
5050 BZ  ASL A
5060      TAX
5070 ;
5080 ; TEST FOR ZERO (BOTH BYTES)
5090 ;
5100      LDA R0L,X
5110      ORA R0H,X
5120      BEQ BR1
5130      RTS
5140 ;
5150 ; DOUBLE RESULT REG INDEX
5160 ;
5170 BNZ ASL A
5180      TAX
5190 ;
5200 ; TEST FOR NON-ZERO (BOTH)
5210 ;
5220      LDA R0L,X
5230      ORA R0H,X
5240      BNE BR1
5250      RTS
5260 ;
5270 ; DOUBLE RESULT REG INDEX
5280 ;
5290 BM1 ASL A
5300      TAX
5310 ;
5320 ; TEST FOR $FF (-1) BOTH BYTES
5330 ;
5340      LDA R0L,X
5350      AND R0H,X
5360      EOR #$FF
5370      BEQ BR1
5380      RTS
5390 ;
5400 ; DOUBLE RESULT REG
5410 ;
5420 BNM1 ASL A
5430      TAX
5440 ;
5450 ; TEST FOR NO $FF
5460 ;
5470      LDA R0L,X
5480      AND R0H,X
```

```
5490      EOR #$FF
5500      BNE BR1
5510 NUL RTS
5520 ; 12*2 FOR R12 AS STACK POINTER
5530 ;
5540 RS  LDX #$18
5550 ;
5560 ; DECR STACK POINTER
5570 ;
5580      JSR DCR
5590 ;
5600 ; POP HIGHRETURN ADDRESS TO PC
5610 ;
5620      LDA (R0L,X)
5630      STA R15H
5640 ;
5650 ; SAME WITH LOW ORDER BYTE
5660 ;
5670      JSR DCR
5680      LDA (R0L,X)
5690      STA R15L
5700      RTS
5710 RTN JMP RTNZ
5720 ;------------------------------
5730 SAVE
5740      STA ACC
5750      STX XREG
5760      STY YREG
5770      PHP
5780      PLA
5790      STA STATUS
5800      CLD
5810      RTS
5820 ;------------------------------
5830 RESTORE LDA STATUS
5840      PHA
5850      LDA ACC
5860      LDX XREG
5870      LDY YREG
5880      PLP
5890      RTS
5900 ;------------------------------
5910 ACC .BYTE 0
5920 XREG .BYTE 0
5930 YREG .BYTE 0
5940 STATUS .BYTE 0

5950 ;------------------------------
5960 TEST
5970      JSR SW16A
5980      .BYTE $11,$00,$F0 ; SET R1
5990      .BYTE $12,$00,$40 ; SET R2
6000      .BYTE $13,$00,$02 ; SET R3
6010      .BYTE $41 ; LD @R1
6020      .BYTE $52 ; ST @R2
6030      .BYTE $F3 ; DCR R3
6040      .BYTE $07,$FB ; BNZ -4
6050      .BYTE $00 ; RTN
6060      RTS
```

```
6070 ;------------------------------
```