Table of Contents

Apple IIe Emulator

Select one of the following categories:

Quick Start

AppleWin runs Apple II programs from disk images, which are single files that contain the contents of an entire Apple floppy disk.

Starting an Apple program is a simple two step process:
1. Click the Drive 1 button on the toolbar and select a disk image file.
2. Click the Run button on the toolbar to boot that disk.

After booting, you may use the emulated Apple exactly as you would use a real Apple.

Of course, using an Apple is not much fun unless you have a library of Apple programs to run, so you'll probably want to get some disk images right away.   The easiest way to do that is to download images from the Internet; see the Resources section for more information. Or, if you want to learn more about creating your own disk images, see the Disks and Disk Images section.

Historical Information

The Apple II holds a unique position in the history of computing. It was the first truly general purpose personal computer, and the first widely successful one. The Apple II took the personal computer revolution from the garages of hard core hobbyists and brought it into business and into millions of homes around the country.

It was developed largely by one man, Steve Wozniak. He designed the system board, employing a number a tricks which made it easier to build but harder to program. He created a floppy drive interface, a hugely important feature at that time, during a marathon two week session in December 1977. He programmed the Apple ROM's and even wrote the first BASIC interpreter for the Apple.

From the start, the Apple II was a major success, fueling the PC revolution and launching Apple Computer Corporation as a major force in the computer industry. By 1980, Apple Computer's yearly revenues already exceeded 100 million dollars. In December of that year, the company went public, making co-founders Steve Wozniak and Steve Jobs each multi-millionaires.

Although the Apple II had originally been designed for hobbyists and home users, about 90% of them were being sold to small businesses. Apple therefore decided that the successor to the Apple II, the Apple III, should be a serious business computer. When it was released in 1980, it featured more memory, an advanced new operating system, and support for 80-column text and lowercase characters.

> *When we came out with the Apple III, the engineering staff canceled every Apple II engineering program that was ongoing, in expectation of the Apple III's success. Every single one was canceled. We really perceived that the Apple II would not last six months.*
>
> *-- Steve Wozniak*

However, the Apple III was late and suffered from poor backwards compatibility and a nearly 100% hardware failure rate. Although Apple eventually addressed these issues, they were not able overcome the Apple III's bad reputation. Apple III sales remained poor, while sales of the older Apple II continued to climb.

In 1983, Apple finally returned its attention to the Apple II series, introducing the Apple IIe. The IIe borrowed some features from the failed Apple III, including 80-column text and lowercase support. However, it was at its heart an Apple II, and retained very strong compatibility with the existing base of Apple II software. The Apple IIe was extremely successful, soon selling at twice the volume of its predecessor.

In 1984, Apple released their first portable computer, the Apple IIc. The IIc was very similar to the IIe, but came in a compact case that included the most popular peripherals, such as a disk drive and serial card, built in. It also included an enhanced CPU (the 65c02) and mouse support. However, the public did not embrace the Apple IIc, partly because it was not expandable like the IIe and partly because people incorrectly equated the small size with a lack of power.

Because the Apple IIe continued to be Apple's best seller, Apple returned focus to it in 1985, releasing the Enhanced IIe. This computer featured the same enhanced CPU as the IIc, and also included improved support for 80-column text and lowercase characters. Then, in 1987, they spruced it up with a new keyboard and some other minor hardware changes. This final IIe, called the Extended Keyboard IIe or the Platinum IIe, is the computer that AppleWin emulates.

In 1986, Apple released one more Apple II, the IIgs.   Although this computer maintained backwards compatibility with most II and IIe programs, it had a radically new architecture and feature set.   It was a 16-bit computer, unlike the previous Apple II's which were all 8-bit. It featured new graphics modes which could display thousands of different colors on the screen at once.   And it had an advanced new sound chip that could play fifteen different sounds at once.   However, partly because it was poorly marketed and partly because the world had turned its attention to the IBM PC and Apple Macintosh, the IIgs never really took off.

Disks and Disk Images

Select one of the following topics:

Introduction to Disk Images

Everyone who once used an Apple II and now uses an IBM-compatible PC has the same problem: how can you make the PC read Apple floppy disks? Unfortunately, without special hardware, you can't.

Floppy disks are analog devices, much like cassette tapes. For a computer to store digital data on a floppy disk, it must "encode" the data into an analog format. The Apple II used a method of encoding called Group Code Recording (GCR), while IBM-compatible PC's use the much more standard Modified Frequency Modulation (MFM) encoding. Since this is all done in hardware and cannot be bypassed, it is not possible for a PC program to "reprogram" the floppy drive in such a way that it could read Apple formatted floppies.

Therefore, instead of reading and writing disks directly, AppleWin uses disk *images*. A disk image is a single file, which you can store on your hard drive or on a PC floppy disk, which contains all of the data from an entire Apple disk. AppleWin treats an image exactly as if it were a real floppy disk.

Creating Disk Images

To create a new disk image, all you have to do is tell AppleWin to use an image file which doesn't already exist.   AppleWin will automatically create a new file.   Specifically, here's what you do:

1. Insert the master DOS disk and boot the emulated Apple.
2. Click on the Drive 1 toolbar button.
3. Instead of selecting a disk image from the list, type in a name for a new disk image and press enter.
4. AppleWin will ask whether you want to create a new file.   Answer yes to confirm that you do.
5. Type in a program that you want DOS to run whenever this new disk is booted.   A simple but useful program is:

   ```
   10 PRINT CHR$(4);"CATALOG"
   ```

6. Type "`INIT HELLO`" to initialize (format) the disk image.

You now have a working disk image, which you can use to save documents or other information.   If you want to fill this image with data from a real floppy disk that you have, then you need to "transfer" the disk's data.   See the Transferring Disk Images topic for more information.

Transferring Disk Images

Serial Line Transfers

The most common method of transferring disk images is through a serial line. To do this, you must connect your Apple to your PC with a serial line and null modem, then run one program on the Apple which reads data off the disk and sends it out over the serial line, and another program on the PC which collects data from the serial line and saves it to a disk image file. This system can be difficult to set up initially, but once it is working it is very fast and convenient.

There are a number of files on ftp.asimov.net which contain programs and tips to help you transfer disks in this manner. One noteworthy program is Apple Disk Transfer (adt120.zip), which can simplify the setup process by automatically installing itself on your Apple through a serial line.

Modem Transfers

If you have a modem and terminal program on both your Apple and PC, you can take advantage of that to transfer disks with very little initial setup. Here's what you do:

1. Run ShrinkIt! on the Apple to compress a disk image into a single archive file.
2. Transfer that file over the modem to your PC.
3. Run Nulib on the PC to uncompress the archive file. Nulib is available from ftp.asimov.net.

Transferring Through 3.5" Disks

One final way to transfer disk images is to copy the data onto a 3.5" disk, and then use a Macintosh to transfer the data from the 3.5" disk into a PC readable format. The advantage of this method is that it does not require a serial card or modem. However, it does involve a number of steps:

1. Run dsk2file on an Apple IIgs. This will read an entire 5 1/4" disk and save it as a single file on a 3.5" ProDOS disk.
2. Take the 3.5" disk to a Macintosh and copy the file using Apple File Exchange or the ProDOS File System Extension.
3. Format a high density 3.5" disk on a PC.
4. Take this 3.5" disk to the Macintosh and write the image file to it using Apple File Exchange or PC Exchange.

Copy Protected Disks

The process of transferring disk images is complicated by the fact that much of the software published for the Apple II was copy protected.

Software publishers have always looked for ways to prevent people from making unauthorized copies of their software.   Today, when you buy a game, it might ask you for a word from a random page of the manual, to ensure that you have purchased the game (complete with manual) and not just copied the disk.   Back in the days of the Apple II, publishers were much more direct: they simply tried to make it physically impossible to copy the disk.

Unlike the PC, the Apple II had to perform much of its disk encoding in software.   If programmers wanted to get tricky, they could bypass the operating system and do their own encoding, possibly changing the size of the sectors on the disk or the way in which the sectors were identified or stored.   This prevented standard operating systems like DOS, along with their standard copying utilities, from accessing the disk.

However, programs which were copy protected in this manner could still be copied with more sophisticated "nibble copiers", which copied each track on the disk bit for bit, rather than copying a sector at a time.   Similarly, to get a program like this to run under AppleWin, all you need to do is make a nibble image of the disk.

After nibble copiers became prevalent on the Apple, some software publishers developed tricky new ways of creating disks that even nibble copiers could not copy.   It is unlikely that such a disk could be successfully transferred into a disk image.

Disk Image Formats

Disk images can be in a number of different formats, depending on how they were created.

DOS Order Images

DOS order disk images contain the data from each sector, stored in the same order that DOS 3.3 numbers sectors.   If you run a DOS program on the Apple which reads in sectors one by one and then transfers them over a serial line to the PC, you will get a DOS order disk image.

Apple floppy disks contained 35 tracks with 16 sectors per track, for a total of 560 sectors. Each of these sectors contained 256 bytes of information, for a total of 143,360 bytes per disk.   Therefore, DOS order disk images are always at least 143,360 bytes long. Sometimes on the Internet you will see a disk image that is 143,488 or 143,616 bytes long; this is probably a DOS order image with extra header information before or after the image. In most cases, AppleWin can automatically detect this and handle it.

ProDOS Order Images

ProDOS order disk images are very similar to DOS order images, except that they contain the sectors in the order that ProDOS numbers them.   If you compress a disk with Shrinkit on an Apple, then transfer it over a modem and uncompress it on the PC, you will get a ProDOS order disk image.

Since ProDOS order disk images contain the same information as DOS order disk images, simply in a different order, they are also about 143,360 bytes long.   When you use a disk image of this size, AppleWin attempts to automatically detect whether it is in DOS order or ProDOS order by examining the contents of the disk.   If the disk was formatted with a standard operating system such as DOS or ProDOS, AppleWin will successfully detect the format.   Otherwise, it will revert to DOS order, which is by far the most common format.   To force ProDOS order, give the file an extension of ".PO".

Nibble Images

Nibble images contain all of the data on a disk; not just the data in sectors but also the sector headers and synchronization areas, all stored in the same encoded format that would be recorded on a real disk's surface.   At 232,960 bytes, nibble images are bigger than other images, but they can be useful for making images of copy protected software.

Using the Toolbar

Help
Displays the help file that you are currently reading.

Run/Reboot
Starts the emulated machine if it is not currently running, or reboots it if it is currently running.

Drive 1
Selects a disk image file for drive 1.

Drive 2
Selects a disk image file for drive 2

Transfer to File
Converts an Apple file from inside a disk image into a real file on your hard drive.

Transfer to Disk
Converts a file from your hard drive into an Apple file in a disk image.

Debug
Displays the actual assembly language instructions that the emulated machine is executing.

Configure
Allows you to customize the emulated machine, and the way the Apple's input and output devices are mapped onto your PC's input and output devices.

Using the Keyboard

The Apple //e keyboard was very similar to the PC keyboard, and most keys correspond directly between the two keyboards.   However, there were a few keys on the Apple //e that are not on the PC; these are described below.

Reset
On the Apple //e, you could usually press Control+Reset to interrupt a running program. With AppleWin you may emulate this key sequence with Ctrl+Break or Ctrl+F12.

Open Apple
The Open Apple key was first introduced in the Apple //e, and was later renamed to the Apple key.   It was similar to Ctrl and Alt on a PC, in that it was used in conjunction with other keys.   AppleWin emulates this key with the PC's left Alt key, which is in the same position as Open Apple was on the original //e.

Solid Apple
The Solid Apple key was introduced on the Apple //e and later renamed to the Option key. AppleWin emulates this key with the PC's right Alt key, which is in the same position as Solid Apple was on the original //e.

Numeric Keypad
The numeric keypad, introduced on the Extended Keyboard //e, is emulated through the PC's numeric keypad.   To enable this feature, turn on Num Lock and make sure the joystick emulation is configured to use something other than the keyboard.

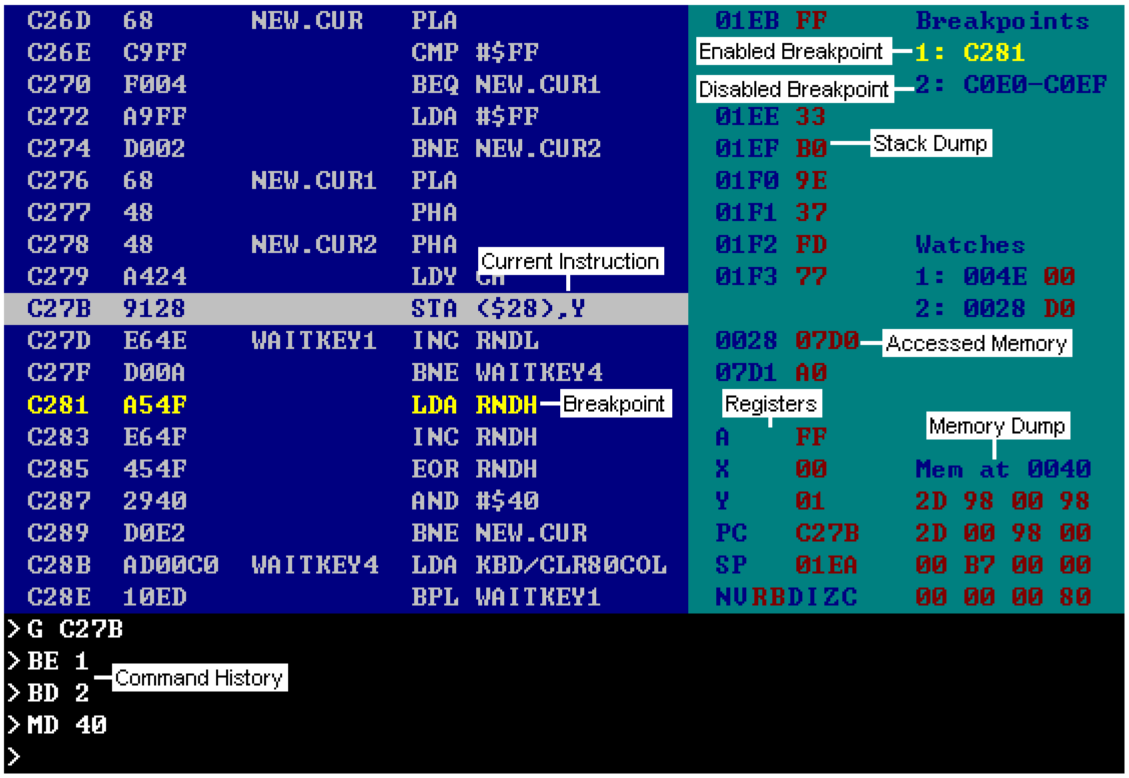
Using the Debugger

AppleWin includes a complete symbolic debugger which you can use to examine the internal workings of Apple programs.   If you're interested in writing Apple II assembly language programs or modifying existing ones, you'll find the debugger to be an invaluable aid.

For more information, select one of the following topics:

The Debugger Screen
Debugger Commands

The Debugger Screen



```
 C26D  68      NEW.CUR   PLA                 01EB FF      Breakpoints
 C26E  C9FF              CMP #$FF                         1: C281
 C270  F004              BEQ NEW.CUR1                     2: C0E0-C0EF
 C272  A9FF              LDA #$FF            01EE 33
 C274  D002              BNE NEW.CUR2        01EF B0
 C276  68      NEW.CUR1  PLA                 01F0 9E
 C277  48                PHA                 01F1 37
 C278  48      NEW.CUR2  PHA                 01F2 FD      Watches
 C279  A424              LDY CH              01F3 77      1: 004E 00
 C27B  9128              STA ($28),Y                      2: 0028 D0
 C27D  E64E    WAITKEY1  INC RNDL            0028 07D0
 C27F  D00A              BNE WAITKEY4        07D1 A0
 C281  A54F              LDA RNDH
 C283  E64F              INC RNDH            A     FF
 C285  454F              EOR RNDH            X     00     Mem at 0040
 C287  2940              AND #$40            Y     01     2D 98 00 98
 C289  D0E2              BNE NEW.CUR         PC    C27B   2D 00 98 00
 C28B  AD00C0  WAITKEY4  LDA KBD/CLR80COL    SP    01EA   00 B7 00 00
 C28E  10ED              BPL WAITKEY1        NVRBDIZC     00 00 00 80
>G C27B
>BE 1
>BD 2
>MD 40
>
```

Debugger Commands

Select a command:

BC	Breakpoint Clear
BD	Breakpoint Disable
BE	Breakpoint Enable
BP	Breakpoint Set
BW	Black and White
COL	Color
G	Go
I	Input
KEY	Feed Keystroke
MD	Memory Dump
MDC	Code Dump
ME	Memory Enter
MF	Memory Fill
O	Output
R	Set Register
Rf	Reset Flag
Sf	Set Flag
T	Trace
ZAP	Remove Instruction

Breakpoint Clear

Syntax

```
BC list
```

```
BC *
```

Description

Permanently removes one or more breakpoints by number, or all breakpoints if the wildcard (*) is used.

Example

To remove breakpoints one and two, type:

```
BC 1 2
```

Breakpoint Disable

Syntax

```
BD list
```

```
BD *
```

Description

Temporarily disables one or more breakpoints by number, or all breakpoints if the wildcard (*) is used.

Example

To temporarily disable breakpoints one and two, type:

```
BD 1 2
```

Breakpoint Enable

Syntax

```
BE list
```

```
BE *
```

Description

Enables one or more breakpoints which had previously been disabled with the Breakpoint Disable (BD) command.

Examples

To enable breakpoints one and two, type:

```
BE 1 2
```

To enable all breakpoints, type:

```
BE *
```

Breakpoint Set

Syntax

```
BP
```

BP *address*

BP *address*L*length*

Description

Sets a breakpoint on the given address or range of addresses. If the breakpoint is on a memory location, it will be triggered if the instruction at that location is about to be executed, or if the memory location is read or written to. If the breakpoint is on an I/O port, it will be triggered if the port is accessed.

After setting a breakpoint, use the Go (G) command to start running the emulator in stepping mode. Breakpoint functionality is available only in stepping mode, not in normal running mode.

Examples

To set a breakpoint at the current execution address (the address contained in the PC register) type:

```
BP
```

To set a breakpoint at address $BF00, the ProDOS Machine Language Interface, type:

```
BP BF00
```

To set a breakpoint on I/O ports $C0E0-$C0EF, trapping all disk I/O on slot 6, type:

```
BP C0E0L10
```

Black and White

Syntax

```
BW
```

Description

Changes the debugger screen to black and white mode.

Color

Syntax

```
COL
```

Description

Changes the debugger screen to color mode.

Go

Syntax

```
G

G address
```

Description

Starts running the emulator in stepping mode.   Stepping mode is slower than the normal running mode, but it allows execution to be interrupted by a triggered breakpoint, the escape key, or execution reaching the address given in the Go command.

Example

To continue execution until the program counter reaches $C27D, the address of WAITKEY1, type:

```
G C27D
```

Input

Syntax

```
I address
```

Description

Simulates reading the specified I/O port.

Example

To simulate a read of port $C083, switching the banked memory at $D000 from ROM to RAM, type:

```
I C083
```

Feed Keystroke

Syntax

```
KEY value
```

Description

Simulates pressing a key.   The given value is passed to the next program that reads the keyboard data port at $C00X.

Example

To simulate pressing the Return key, type:

```
KEY 8D
```

Memory Dump

Syntax

```
MD address
```

Description

Displays the contents of memory starting at the specified address, in hexadecimal notation.

Example

To display memory at $BF00, type:

```
MD BF00
```

Code Dump

Syntax

```
MDC address
```

Description

Displays disassembled code starting at the specified address.

Example

To display code starting at $F832, type:

```
MDC F832
```

Memory Enter

Syntax

```
ME address value(s)
```

Description

Writes the given values to memory locations starting at the specified address.

Example

To write $A9 to memory location $FBE4 and $0A to memory location $FBE5, type:

```
ME FBE4 A9 0A
```

Memory Fill

Syntax

```
MF addressLlength value
```

Description

Fills a range of memory locations with the given value.

Example

To fill memory locations $FBE4 through $FBEE with the value $EA, type:

```
MF FBE4L0B EA
```

Output

Syntax

```
O address value
```

Description

Writes the specified value to the given I/O port. If *value* is not specified, a value of zero is assumed.

Example

To write $FF to I/O port $C070, type:

```
O C070 FF
```

Set Register

Syntax

```
R register=value
```

where *register* is:

A	Accumulator
X	X index
Y	Y index
PC	Program counter
SP	Stack pointer

Description

Sets the specified register in the emulated CPU to the given value. The value is adjusted if necessary to fit the valid range of values for the specified register.

Examples

To set the value in the accumulator to $80, type:

```
R A=80
```

To set the program counter to $FA62, type:

```
R PC=FA62
```

Reset Flag

Syntax

```
Rf
```

where f is:

N	Sign flag
V	Overflow flag
R	Reserved flag
B	Break flag
D	Decimal flag
I	Interrupt flag
Z	Zero flag
C	Carry flag

Description

Clears the specified processor status flag.

Example

To clear the carry flag, type:

```
RC
```

Set Flag

Syntax

```
Sf
```

where f is:

N	Sign flag
V	Overflow flag
R	Reserved flag
B	Break flag
D	Decimal flag
I	Interrupt flag
Z	Zero flag
C	Carry flag

Description

Sets the specified processor status flag.

Example

To set the decimal flag, type:

```
SD
```

Trace

Syntax

```
T
```

```
T count
```

Description

Executes one or more instructions at the current program counter (PC) location.

Example

To execute five assembly language instructions, type:

```
T 5
```

Remove Instruction

Syntax

```
ZAP
```

Description

Removes the current instruction (the instruction to which PC points) by replacing it with one or more NOP instructions.

Resources

Select one of the following categories:

Internet Newsgroups

comp.emulators.apple2
This newsgroup is an excellent source of information about Apple II emulation, and the best place to post questions, requests, and suggestions.

comp.emulators.announce
If you are only interested in hearing announcements of new emulator products and new versions of AppleWin and other emulators, you may want to subscribe to this newsgroup instead of comp.emulators.apple2.   This newsgroup is also a good place to look for answers to frequently asked questions.

comp.sys.apple2
This newsgroup is for general discussion and questions about the Apple II series of computers.

comp.sys.apple2.programmer
This newsgroup is a good source of information about programming the Apple II series of computers.

Internet FTP Sites

Before transferring a program or disk image through FTP, make sure to configure your FTP client for binary transfer mode.   With most FTP clients you can do this by simply typing the word "binary".

ftp.asimov.net
This site is the largest Apple II emulation site, and the official release point for new versions of AppleWin.   Under the /pub/apple_II directory, you will find disk images, utilities for making your own disk images, and Apple emulators for other computers and operating systems.

Contacting the Author

To contact the author, write to:

Michael O'Brien
3 Trovita
Irvine, CA 92714