An Apple II Build Chain for *Inform*

Michael Sternberg

Abstract
Inform is a language and compiler which creates object code that can be executed by Infocom's Z-machine. Z-machine interpreters exist for many, many computer platforms, one of which is the Apple II. This paper describes the tools and methods for targeting Inform object code for a Z-machine interpreter running on an 8-bit Apple II computer.

Contents

Acknowledgments

- Graham Nelson for creating *Inform*, as well as his work on reverse-engineering and documenting Infocom's Z-machine.
- Dave Bernazzani for creating *mInform*, a version of the Inform libraries for computers with restricted memory resources.
- Steve Nickolas (a.k.a. Usotsuki) for creating the disk-sector interleaving tools for Apple IIs: *interl* and *interlz5*.

Introduction

Historical Background

The computer fantasy simulation, *Zork*, was created by members of the Dynamic Modeling Group (DM) within MIT's Laboratory for Computer Science. The program was implemented using a LISP-like programming language invented within the group called MDL and ran on a Digital PDP-10 DECSystem-10 with 256 kilowords of memory (approximately 1152 kilobytes). Development on *Zork* began in the spring of 1977 and continued until winter 1979 having been expanded to the limits of the system's available address space.

When Infocom was founded by members of DM, it was decided its first product would be a microcomputer version of *Zork*. The target platforms were the TRS-80 (Model I) and Apple II, each with a minimum requirement of 32 kilobytes of memory. Converting *Zork* from the original mainframe environment to the resource-constrained microcomputers presented a daunting technical challenge.

Elsewhere, UCSD Pascal's P-machine proved it was possible to densely pack program instructions as bytecodes (P-code) and, at the same time, provide a means for cross-platform compatibility. Although inspirational, Pascal was ill-suited to Infocom's goals. Being a general purpose programming language meant it consumed resources for functionality not needed by *Zork*. Additionally, programming in Pascal would require a total re-write of *Zork*'s existing MDL codebase.

Infocom instead devised its own virtual machine, dubbed the Z-machine. The Z-machine was optimized for running text adventure programs on computers with limited resources.

A cross-compiler system was created to read source code syntactically similar to the original mainframe version of *Zork* and generate Z-machine bytecodes (or Z-code) as output.

And finally, a Z-machine Interpreter Program (or ZIP), would run on a host platform, such as the Apple II. The ZIP is an emulator of the imaginary Z-machine and having the job of evaluating and executing Z-code instructions.

Roadmap for this Paper

The goal of this paper is to demonstrate how to obtain and/or build the pieces needed to create a diskette-based Infocom-like text adventure for the Apple II.

To get there, the following ingredients are needed:

- An Infocom ZIP (Z-machine Interpreter Program) for the 8-bit Apple II.
- A version of the interactive fiction development tool, *Inform*, that can generate story files, or Z-code, compatible with the Apple II ZIPs
- A set of Inform library files needed for the text parser and object definitions.
- A tool called *interl*, that performs disk-sector interleaving of the Z-code (story file) and writes its output to an Apple II diskette image.

Each of these items will be discussed in turn.

1. Z-machine Interpreter Program Versions

The first versions of the Z-machine Interpreter Program were created for the Digital PDP-11, Tandy/Radio Shack TRS-80, and the Apple II to implement *Zork* I, II, and III. But over time, many different host platforms and many different games besides *Zork* would come to be supported by the Z-machine architecture and its interpreters.

1.1 ZIP Major Versions

Throughout the 1980s, some of the resource constraints found with the earliest microcomputers eased. The norm for installed memory grew from, say, 16 kilobytes, to 64 kilobytes, and later 128 kilobytes. Terminal displays grew from 40 character width to 80 characters. Meanwhile, Infocom developers were beginning to imagine new game features unsupported by the original Z-machine architecture.

Hence, the Z-machine – and transitively – the ZIPs were updated to support larger story files, improved user-interfaces, and new in-game features.

Table 1 contains a list of the major Z-machine and interpreter versions created by Infocom. The bulk of Infocom titles employed the ZIP (v3) architecture. EZIP/LZIP (v4) increased the possible story file size. XZIP (v5) added features like real-time gameplay. And YZIP (v6) added graphics.

Table 1. Major Versions of Infocom ZIPs

Version	Name	Max Story File
1-3	ZIP	128K
4	EZIP/LZIP	256K
5	XZIP	256K
6	YZIP	576K

1.2 ZIP Minor Versions (Apple II)

Between the major architectural changes to the imaginary Z-machine, there were also many minor updates and fixes to the interpreters. Infocom denoted these different versions using letters (A, B, ...).

Table 2. (Most) Minor Versions of Apple II Infocom ZIPs

Major	Minor	Example Title	Serial
2	-	Zork I	UG3AU5
3	-	Deadline	820311
3	A	Sorcerer	840131
3	B	Sea Stalker	840320
3	E	Wishbringer	850501
3	H	Spellbreaker	850916
3	K	Leather Goddesses	860711
3	M	Moonmist	861022
4	A	A Mind Forever Voyaging	850814
4	B	Trinity	860509
4	C	Bureaucracy	870212
4	H	Nord and Bert	870722
5	A	Border Zone	871008
5	E	Hitchhiker's Guide (SG)	871119
5	F	Sherlock Holmes	871214

2. Disk Organization of an Apple II Title

The disk contents of an Infocom text adventure for the Apple II is divided into three segments:

1. Boot code and a Z-machine Interpreter Program (ZIP)
2. Z-code (Story File)
3. Remaining unused space

2.1 Disk Organization (ZIP)

An Infocom game for the Apple II using a ZIP (v3) is stored on a single 16-sector disk. The first three tracks (12,288 bytes) contain the boot code and the Z-machine Interpreter Program. The fourth track and on contain the Z-code segment (story file). Since the story file can never be more than 128 kilobytes, a ZIP (v3) game will always fit on a standard 140k Apple II floppy disk (12k + 128k = 130k).

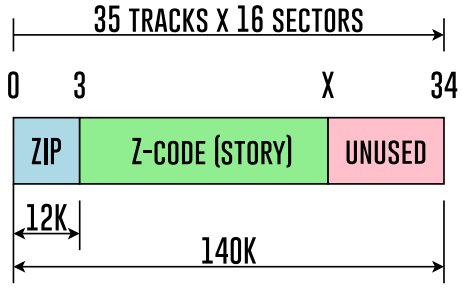


Figure 1. Disk Contents of an Apple II ZIP (v3) Title

2.2 Disk Organization (XZIP)

An Infocom game for the Apple II having an XZIP (v5) is stored on one or two disks. The first disk is formatted with 16-sectors, where the first 4 tracks contain the boot code and the Experimental Z-machine Interpreter Program (XZIP). The fifth track and on contain the Z-code segment (story file). If the story file is larger than 100,864 bytes, the remainder of the Z-code is stored on a second 18-sector disk image.

If a story file is less than 100,864 bytes, XZIP version 5A supports it as a single-disk game. XZIP versions 5E and 5F do not support single-disk games. That is, (from my experience) the story must be larger than 100,864 bytes if using XZIP 5E or 5F.

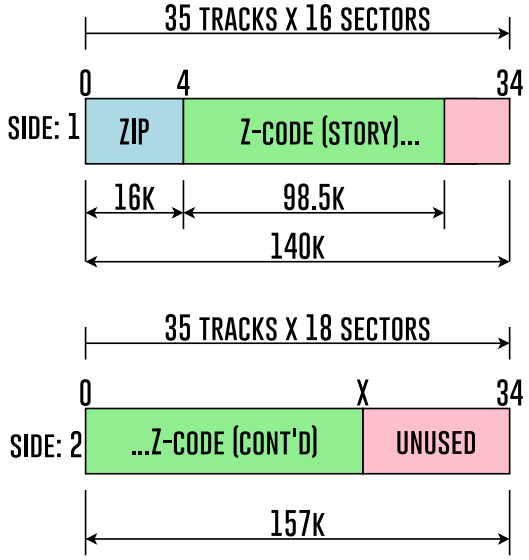


Figure 2. Disk Organization of an Apple II XZIP (v5) Title

2.3 Z-Code Sector Interleaving

For most Apple II ZIPs, the Z-code (story file) is written to the diskette's sectors that have been arranged in a non-sequential, interleaved order. This is a common performance optimization for spinning media. If done properly, the computer will complete the processing of the most recently read sector just before the next expected sector is approaching the read/write head. This works out to 6 or 7 sectors being skipped. See **Figures 3 and 4** .

To produce an Infocom ZIP-compatible disk will require writing Z-code in the sector-interleaved order expected by the ZIP. This job will be handled by the tools interl(z3) and interlz5.

There is at least one Infocom title for the Apple II that did not employ interleaving. Infocom's ZIP (v3) E interpreter (found on *Wishbringer* Release 68 / Serial 850501) does not use interleaving. This will be used in an upcoming example to quickly concatenate a ZIP (v3), Z-code (story file), and remaining unused space to assemble a bootable Apple II diskette image.

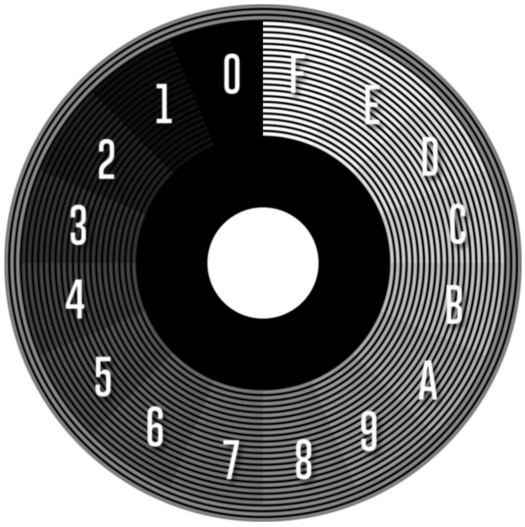


Figure 3. Non-Interleaved Z-Code

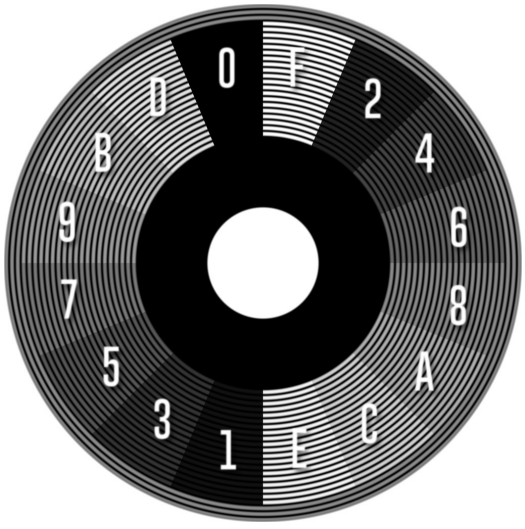


Figure 4. Interleaved Z-Code

3. ZIP: Obtaining an Apple II Z-machine Interpreter

3.1 Extracting a ZIP from an Infocom Disk Image

1. Download one or more ZIP (v3) Infocom games for the Apple II. In this example, *Wishbringer* and *The Lurking Horror* are downloaded from the Asimov archive.

```
$ wget http://mirrors.apple2.org.za/ftp.apple.asimov.net/images/games/adventure/infocom/{lurking_horror,wishbringer}.dsk
```

2. Determine the minor version of the Infocom ZIP found on the disk. Launch the games in an Apple II emulator and run the `$VERIFY` command at the parser prompt. See **Figures 5 and 6** .

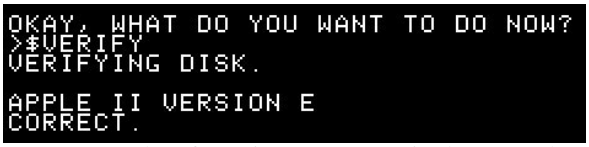


Figure 5. Results of running `$VERIFY` in the example *Wishbringer*. Notice Version `E`.

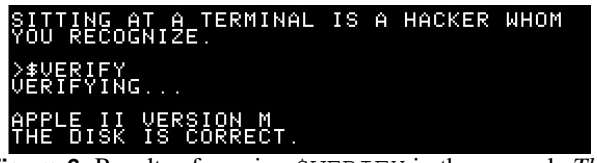


Figure 6. Results of running `$VERIFY` in the example *The Lurking Horror*. Notice Version `M`.

3. Extract the ZIP from the first three tracks (12K) from the `.dsk` file using the Unix `head` command. Choose a name that denotes the major and minor versions.

```
$ head --bytes 12288 wishbringer.dsk > info3e.bin
$ head --bytes 12288 lurking_horror.dsk > info3m.bin
```

3.2 Extracting an XZIP from an Infocom Disk Image

1. Download one or more XZIP (v5) Infocom games for the Apple II. In this example, *Beyond Zork* and *Sherlock: Riddle of the Crown Jewels* are downloaded from the Asimov archive.

```
$ wget http://mirrors.apple2.org.za/ftp.apple.asimov.net/images/games/adventure/infocom/beyond_zork/beyondzork{1.dsk,2.nib}
```

```
$ wget http://mirrors.apple2.org.za/ftp.apple.asimov.net/images/games/adventure/SherlockRiddleOfTheCrownJewels-{S1,S2-Reconstructed}.dsk
```

2. Determine the minor version of the Infocom XZIP found on the disk. Launch the games in an Apple IIe emulator (remember these games require 128K RAM) and run the `VERSION` command at the parser prompt. See **Figures 7 and 8** .

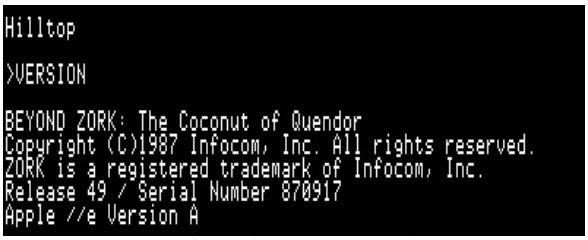


Figure 7. Results of running `VERSION` in the example *Beyond Zork*. Notice Version `A`.

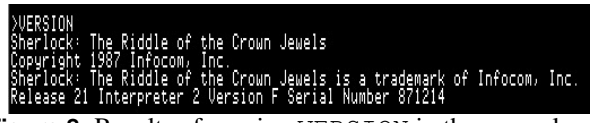


Figure 8. Results of running `VERSION` in the example *Sherlock: The Riddle of the Crown*. Notice Version `F`.

3. Extract the ZIP from the first four tracks (16K) from the side one disk image using the Unix `head` command. Choose a name that denotes the major and minor versions.

```
$ head --bytes 16384 beyondzork1.dsk > info5a.bin
$ head --bytes 16384 SherlockRiddleOfTheCrownJewels-S1.dsk > info5f.bin
```

4. Z-Code: Using *Inform* to Create Story Files

Inform is a programming language and compiler that can generate Infocom Z-code.

4.1 Inform 6.1.5

Inform 6.1.5 was the last version of the compiler with support for the creation of ZIP (v3) story files. Because our target platform is an Apple II ZIP (v3) or (v5), *Inform* 6.1.5 will be the compiler version used for the examples to follow.

4.2 Build Inform 6.1.5 from source

1. Download the *Inform* 6.1.5 source code.

```
$ wget http://www.ifarchive.org/↩
    if-archive/infocom/compilers/↩
    inform6/source/old/inform615_↩
    source.zip
```

2. Unarchive `inform615_source.zip`.

```
$ mkdir inform615
$ unzip -d inform615 inform615_↩
    source.zip
```

3. Rename source files to something the GNU C compiler expects. See **Table 3** for a complete list of the intended results.

```
$ cd inform615
$ rename 's/$/.c/' *
$ mv Relnote.c Relnote.txt
$ mv header.c header.h
```

4. Edit line `69` of `header.h`. The preprocessor directive must be set to `LINUX` in order for code specific to a GNU environment to be compiled. This is true whether using Linux, MacOS, or Windows with Cygwin/MinGW).

   Before

```
69 #define ARCHIMEDES
```

   After

```
69 #define LINUX
```

5. Compile. If successful, a new executable called `inform` will be found in the current working directory.

```
$ cc -o inform *.c
$ ls -l inform
```

6. Copy the newly-created `inform` executable to a convenient location, such as `/usr/local/bin`

```
$ sudo cp inform /usr/local/bin
```

4.3 Test Inform 6.1.5

Here, create and compile a small bit of *Inform* code and test it using a modern-day Z-machine interpreter such as *Frotz* or *Zoom*.

1. Create a text file called `hello.inf` containing a somewhat minimal Inform program.

```
[ Main i;
    for (i=0 : i<5 : i++)
    {
       print i, ": Hello!^";
    }
];
```

2. Compile `hello.inf` for Z-code for a ZIP (v3). This should create a file called `hello.z3`.

```
$ inform -v3 hello.inf
$ ls -l hello.z3
```

   or, to create Z-code for an XZIP (v5).

```
$ inform -v5 hello.inf
$ ls -l hello.z5
```

3. (Optional) Run `hello.z3` using a modern-day ZIP such as *Frotz*.

```
$ frotz hello.z3

0: Hello!
1: Hello!
2: Hello!
3: Hello!
4: Hello!
[Hit any key to exit.]
```

4. Create a bootable Apple II diskette by concatenating a ZIP (v3) E, the Z-code, and finally padding the remaining unused space with zeroes. In the end, the `.dsk` file should be 143,360 bytes. Note: ZIP (v3) E does not expect Z-code to be stored on the diskette in a sector-interleaved order.

```
$ cat info3e.bin hello.z3 > ↩
    hello.dsk
$ ls -l hello.dsk
... 13824 Jul 10 21:29 hello.dsk
$ echo $((143360 - 13824))
129536
$ head --bytes 129536 /dev/zero ↩
    >> hello.dsk
ls -l hello.dsk
... 143360 Jul 10 21:29 hello.↩
    dsk
```

Table 3. Inform 6.1.5 File Renaming

Before		After
arrays	→	arrays.c
asm	→	asm.c
bpatch	→	bpatch.c
chars	→	chars.c
directs	→	directs.c
errors	→	errors.c
expressc	→	expressc.c
expressp	→	expressp.c
files	→	files.c
inform	→	inform.c
lexer	→	lexer.c
linker	→	linker.c
memory	→	memory.c
objects	→	objects.c
states	→	states.c
header	→	header.h
symbols	→	symbols.c
syntax	→	syntax.c
tables	→	tables.c
text	→	text.c
veneer	→	veneer.c
verbs	→	verbs.c
Relnote	→	Relnote.txt

5. Boot the `hello.dsk` disk image using an Apple II emulator.

5. Z-Code: Using *Inform* Libraries

The simple Inform example in the previous section is sufficient for proving the compiler behaves as intended and that it's possible to create a bootable Apple II diskette containing a custom Inform program. But it's not interactive fiction. To build that requires a parser, a vocabulary, and a collection of objects (rooms, things, etc).

Infocom's Z-machine does not innately handle the job of parsing text. The parser is itself written in *Inform* and executed by the ZIP. A good, general-purpose parser is available as a bundle of standard Inform libraries, usually named `Parser.h`, `Grammar.h`, and `VerbLib.h`.

Many iterations of the *Inform* Libraries exist and are versioned and distributed separately from the *Inform* compiler using a major/minor notation like (6/1, 6/2, ..., 6/12). For compatibility reasons, *Inform* Libraries contemporaneous with the 6.1.5 compiler must be used. This seems to be in the range of *Inform* Libraries 6/2 or 6/3.

Old libraries can be found at the following URL:

```
http://www.ifarchive.org/indexes/if-↩
    archiveXinfocomXcompilersXinform6↩
    XlibraryXold.html
```

But first we'll examine a no-frills fork of the *Inform* Libraries called *mInform*.

5.1 mInform

mInForm is an unofficial, feature-reduced parser library for *Inform*. It was created with the goal of targeting computers with tight memory constraints.

If you are going to create a ZIP (v3) story, then you may wish to compile it using the *mInform* libraries to remain under the 128 kilobyte restriction of the ZIP (v3) interpreter. A bare Inform program compiled using *mInform* generates a 22 kilobyte story file. By comparison, the same source compiled with the Inform 6/2 libraries is 44 kilobytes.

In this example, we'll download *mInform* and compile the included example code called `minform.inf`.

1. Download *mInform* from the IF-archive.

```
$ wget http://www.ifarchive.org/↩
    if-archive/infocom/compilers/↩
    inform6/library/contributions↩
    /minform.zip
```

2. Unarchive `minform.zip` to its own subdirectory.

```
$ mkdir minform
$ unzip -d minform minform.zip
```

3. Correct filenames for Unix.

```
$ cd minform
$ mv grammar.h Grammar.h
$ mv parser.h Parser.h
$ mv verblib.h VerbLib.h
```

4. Compile the included *mInform* example.

```
$ inform -v3 minform.inf
$ ls -l minform.z3
```

5. (Optional) Run `minform.z3` using a modern-day ZIP such as `frotz`.

```
$ frotz minform.z3
```

5.2 Inform Library 6/3

In this example, we'll download and compile an *Inform* conversion of the 1976 Crowther and Woods text adventure called *Advent*. For this effort, *Inform* Library 6/3 is needed.

1. Download *Inform* 6/3 libraries from IF-archive.

```
$ wget http://www.ifarchive.org/↩
    if-archive/infocom/compilers/↩
    inform6/library/old/inform_↩
    library63.zip
```

2. Unarchive `inform_library63.zip` to its own subdirectory.

```
$ mkdir inform_library63
$ unzip -d inform_library63 ↩
    inform_library63.zip
```

3. Correct filenames.

```
$ cd inform_library63
$ rename 's/$/.h/' *
$ mv Verblib.h VerbLib.h
```

4. Download the source code for *Advent*.

```
$ wget http://www.ifarchive.org/↩
    if-archive/games/source/↩
    inform/Advent.inf
```

5. Compile `Advent.inf`

```
$ inform -v5 Advent.inf
```

6. (Optional) Run `Advent.z5` using a non-Infocom ZIP such as `frotz` or `xzip`

```
$ xzip Advent.z5
```

6. Writing to an Apple II Disk Image

6.1 Writing a ZIP (v3) Disk Image

Steve Nickolas' original *Interl* tool from 2002, which can be found at the IF-Archive as a compiled DOS/Windows executable along with its QBASIC source, provides a way to create an Apple II disk image using an Infocom ZIP (v3) and Z-code (story file). It handles the task of writing the Z-code segment in the sector interleaving order discussed earlier.

A C language port of *interl*, called *interl(z3)*, is now available. In this section, we'll download, build, and install *interl(z3)* and use it to create a ZIP (v3) disk image.

1. Download *interl(z3)*.

```
git clone https://bitbucket.org/↩
    michael_sternberg/interlz3
```

2. Build *interl(z3)*.

```
$ cd interlz3
$ make
$ ls -l interlz3
```

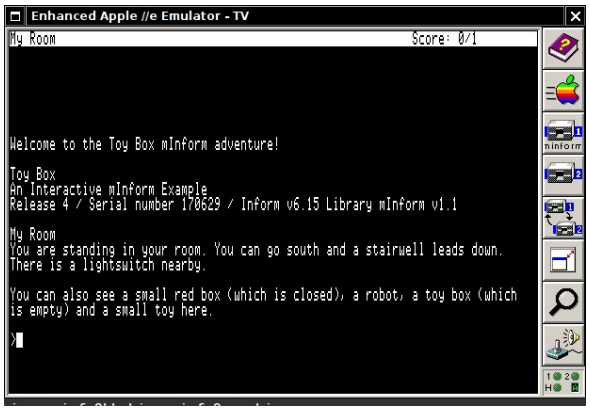
3. Install *interl(z3)* to a convenient location.

```
$ sudo cp interlz3 /usr/local/↩
    bin
```

4. Use *interlz3* to create `minform.dsk` using a ZIP (v3) and Z-code (story file) created earlier.

```
$ interlz3 info3m.bin minform.z3↩
    minform.dsk
```

5. Boot the `minform.dsk` disk image using an Apple emulator.



6.2 Writing an XZIP (v5) Disk Image

In December 2016, Steve Nickolas created an sector interleaving tool for XZIP (v5) story files, called *interlz5*. *Interlz5* creates one or two disk images depending on the size of the Z-code (story file). An XZIP interpreter and Z-code are written to the first disk in a manner similar to *interlz3*. If the story data is sufficiently large, a second, 18-sector disk is written in nibblized (`.nib`) form .

1. Download *interlz5*.

```
$ wget http://3.buric.co/interlz↩
    5-001.zip
```

2. Unarchive *interlz5*.

```
$ mkdir interlz5
$ unzip -d interlz5 interlz↩
    5-001.zip interlz5.c
```

3. Build *interlz5*.

```
$ cd interlz5
$ cc -o interlz5 interlz5.c
$ ls -l interlz5
```

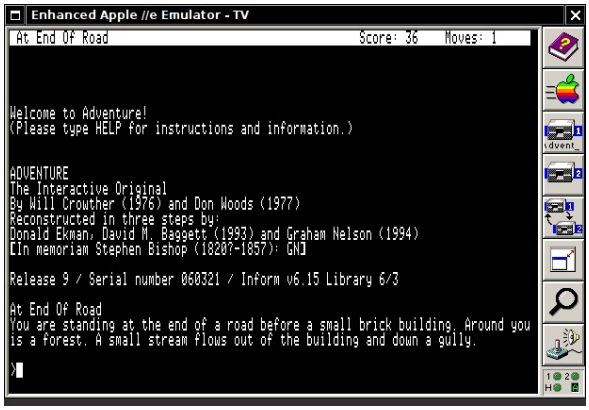
4. Install *interlz5*.

```
$ sudo cp interlz5 /usr/local/bin
```

5. Use *interlz5* to create two disks for *Advent* using an XZIP (v5) and Z-code (story file) created earlier.

```
$ interlz5 info5a.bin advent.z5 advent_s1.dsk advent_s2.nib
```

6. Boot the `advent_s1.dsk` disk image using an Apple II emulator.



7. Survey of Infocom Titles & ZIPs on Asimov

A list of Infocom titles and their corresponding release and interpreter details **Table 4** .

Some observations:

- No early versions of *Zork I* as published by Personal Software appear to exist in the archive.
- Many version M diskettes appear to be non-Infocom hacks where a ZIP and Z-code were cobbled together. Presumably this was a fan's preference to use a more mature version of the interpreter with older titles. The suspicious titles are highlighted in yellow.
- *Wishbringer* Release 68 appears as version E in the Asimov archive. However a version F with the same release number was found at `archive.org`. This is the only example of a version E interpreter. Version E does not employ sector interleaving.
- Inform published several classic titles as *Solid Gold* editions. These used XZIP (v5) interpreters. *Hitchhiker's Guide* appears to be the only one found in the Asimov archive.

Table 4. Infocom titles for the Apple II on Asimov sorted by Serial

Title	Serial	Release	Major	Minor
Zork I	UG3AU5	15	2	-
Zork II	UG3AU5	7	2	-
Deadline	820311	18	3	-
Deadline	820809	22	3	-
Zork I	820803	26	3	-
Starcross	821021	17	3	-
Suspended	830222	5	3	-
Zork III	830331	15	3	-
Planetfall	830708	20	3	-
Enchanter	830810	10	3	-
The Witness	830910	18	3	-
Infidel	830916	22	3	-
Infidel	830916	22	3	A
Deadline	831005	27	3	M
Sorcerer	840131	4	3	A
Seastalker	840320	86	3	B
Seastalker	840501	15	3	B
Suspended	840521	8	3	M
Zork I	840726	88	3	M
Zork III	840727	17	3	B
Cutthroats	840809	23	3	B
The Witness	840904	22	3	M
Zork II	840904	48	3	M
Hitchhiker's Guide	840914	47	3	B
Suspect	841005	14	3	B
Hitchhiker's Guide	841221	56	3	B
Wishbringer	850501	68	3	E
Wishbringer	850501	68	3	F
A Mind Forever Voyaging	850814	77	4	A
Spellbreaker	850916	63	3	H
Wishbringer	850920	69	3	M
Planetfall	851003	37	3	M
Ballyhoo	851218	97	3	H
Trinity	860509	11	4	B
Leather Goddesses	860711	50	3	K
Leather Goddesses	860730	59	3	M
Enchanter	860820	29	3	M
Sorcerer	860904	18	3	M
Spellbreaker	860904	87	3	M
Moonmist	861022	9	3	K
Hollywood Hijinx	861118	235	3	K
Hollywood Hijinx	861215	37	3	K
Bureaucracy	870212	86	4	C
Stationfall	870430	107	3	M
The Lurking Horror	870506	203	3	M
Bureaucracy	870602	116	4	H
Nord and Bert	870722	19	4	H
Plundered Hearts	870730	26	3	M
Beyond Zork	870917	49	5	A
The Lurking Horror	870918	221	3	M
Border Zone	871008	9	5	A
Hitchhiker's Guide (Solid Gold)	871119	31	5	E
Sherlock	871214	21	5	F

Possibly hacked ZIP (v3) M & Z-code combination
EZIP/LZIP (v4)
XZIP (v5)