VXA: A Virtual Architecture for Durable Compressed Archives

Bryan Ford
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

http://pdos.csail.mit.edu/~baford/vxa/

The Ubiquity of Data Compression

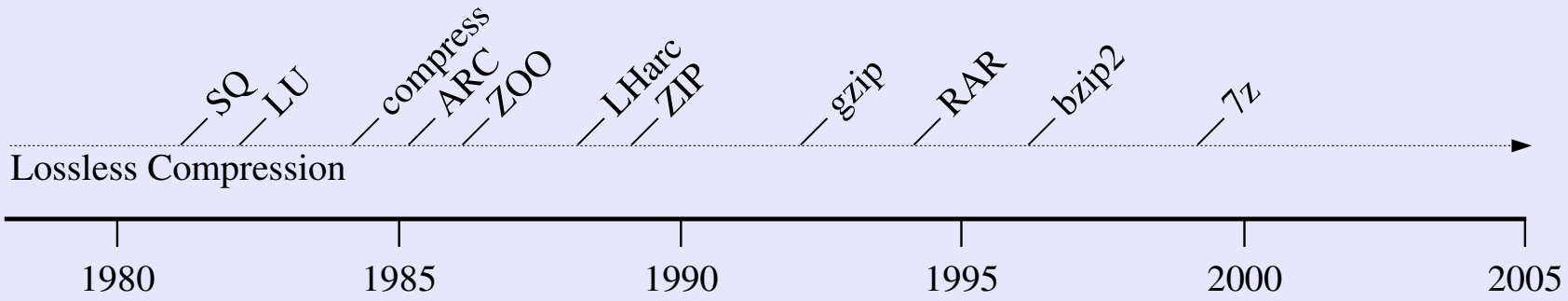
Everything is compressed these days

- **Archive/Backup/Distribution:** ZIP, tar.gz, ...
- **Multimedia streams:** mp3, ogg, wmv, ...
- **Office documents:** XML-in-ZIP
- **Digital cameras:** JPEG, proprietary RAW, ...
- **Video camcorders:** DV, MPEG-2, ...

Compressed Data Formats

Observation #1:

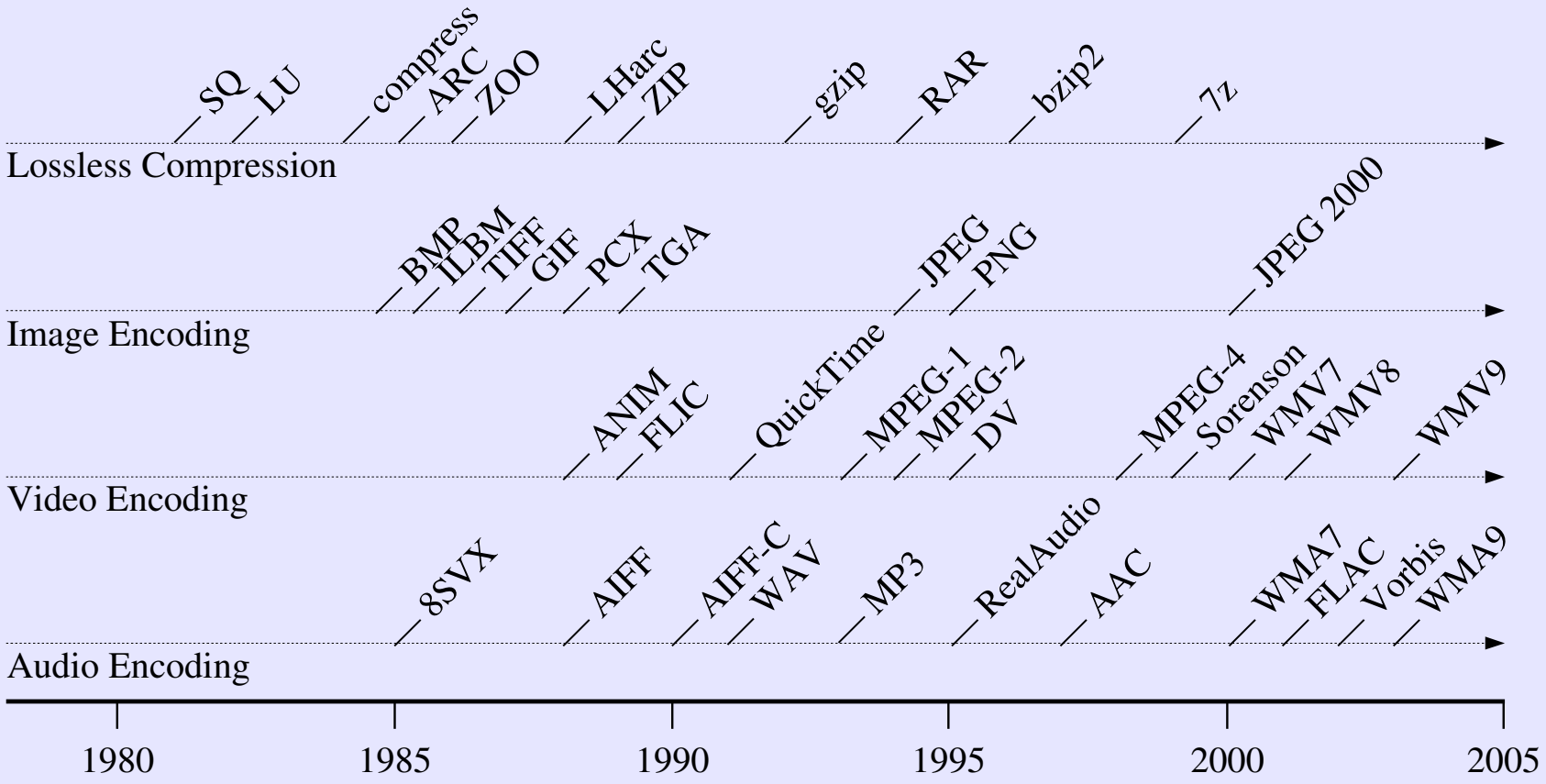
Data compression formats evolve rapidly



Compressed Data Formats

Observation #1:

Data compression formats evolve rapidly



Compressed Data Formats

Observation #1:

Data compression formats evolve rapidly

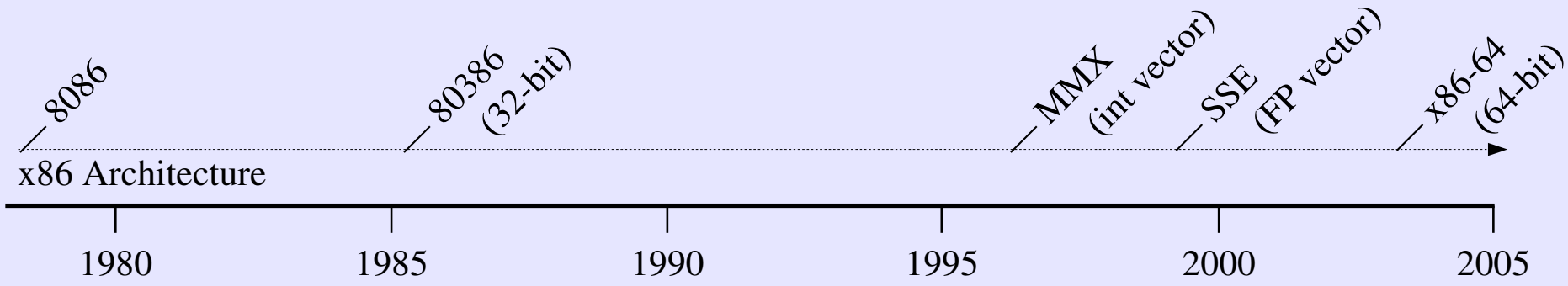
Problems:

- *Inconvenient:*
  each new algorithm requires decoder install/upgrade
- *Impedes data portability:*
  data unusable on systems without supported decoder
- *Threatens long-term data usability:*
  old decoders may not run on new operating systems

Archiving Compressed Data

Observation #2:

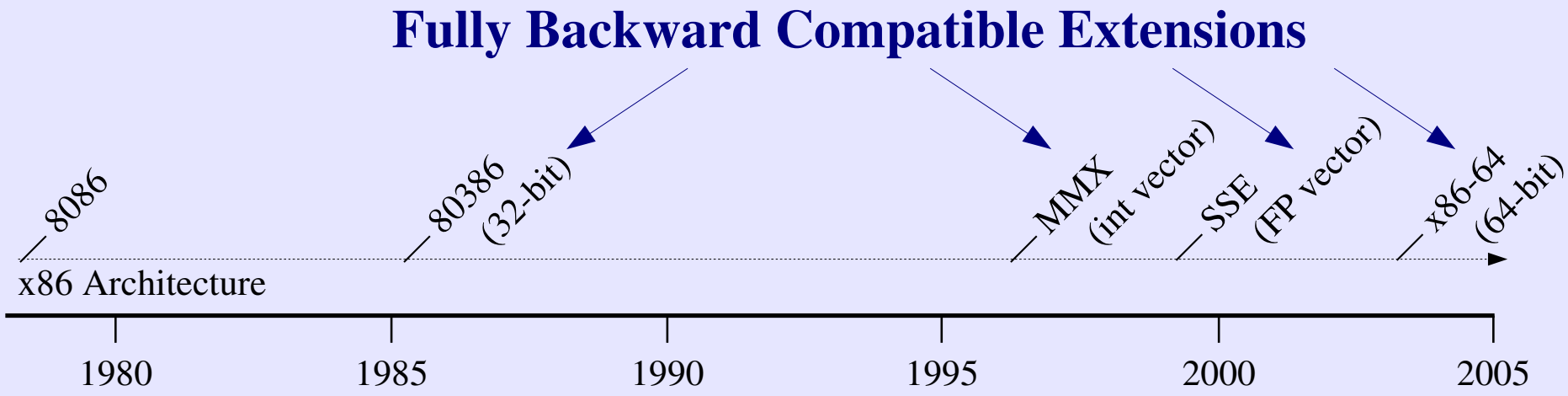
Processor architectures evolve more conservatively



Archiving Compressed Data

Observation #2:

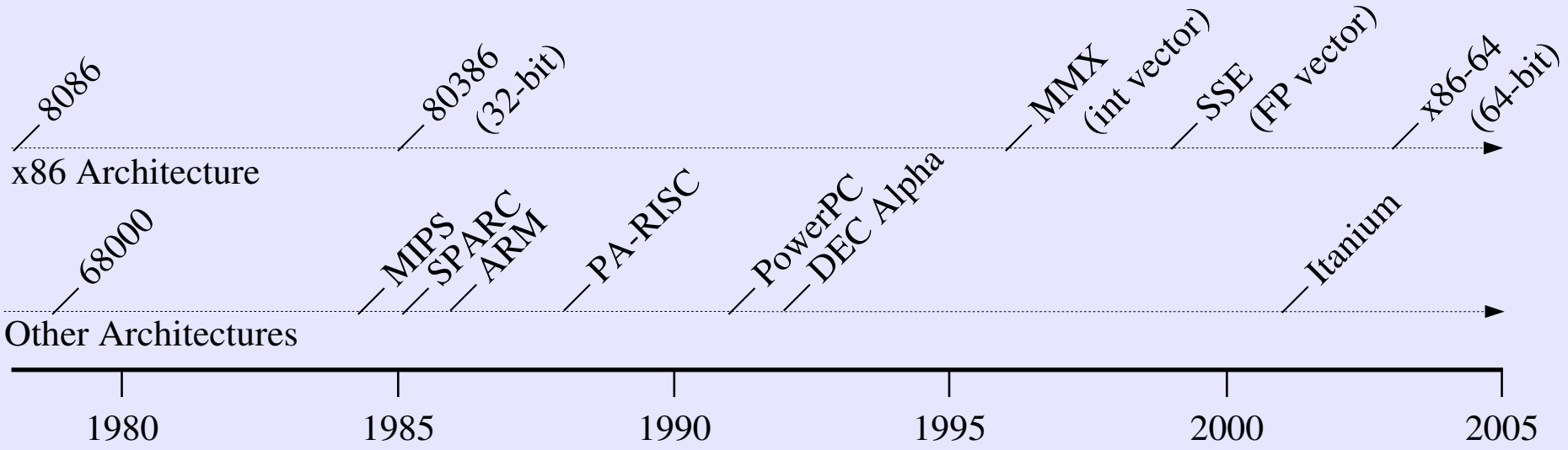
Processor architectures evolve more conservatively



Archiving Compressed Data

Observation #2:

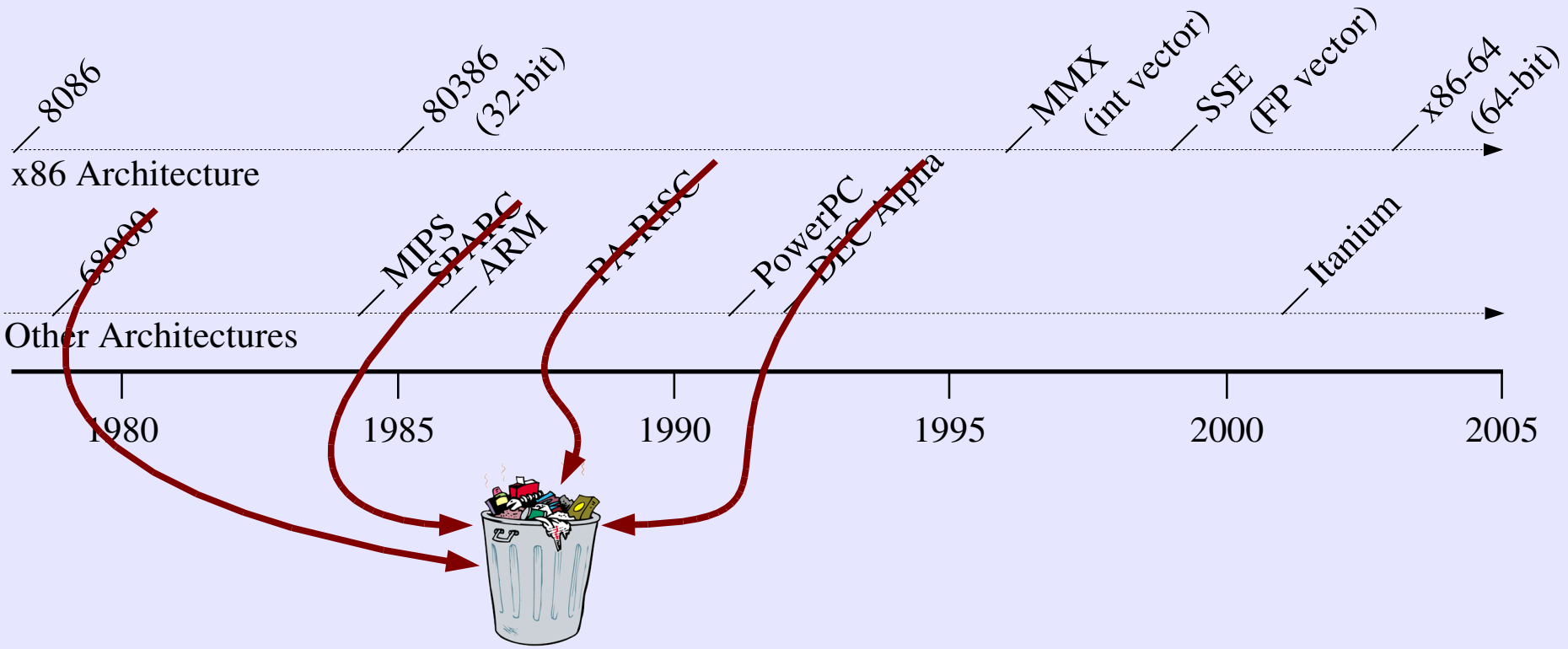
Processor architectures evolve more conservatively



Archiving Compressed Data

Observation #2:

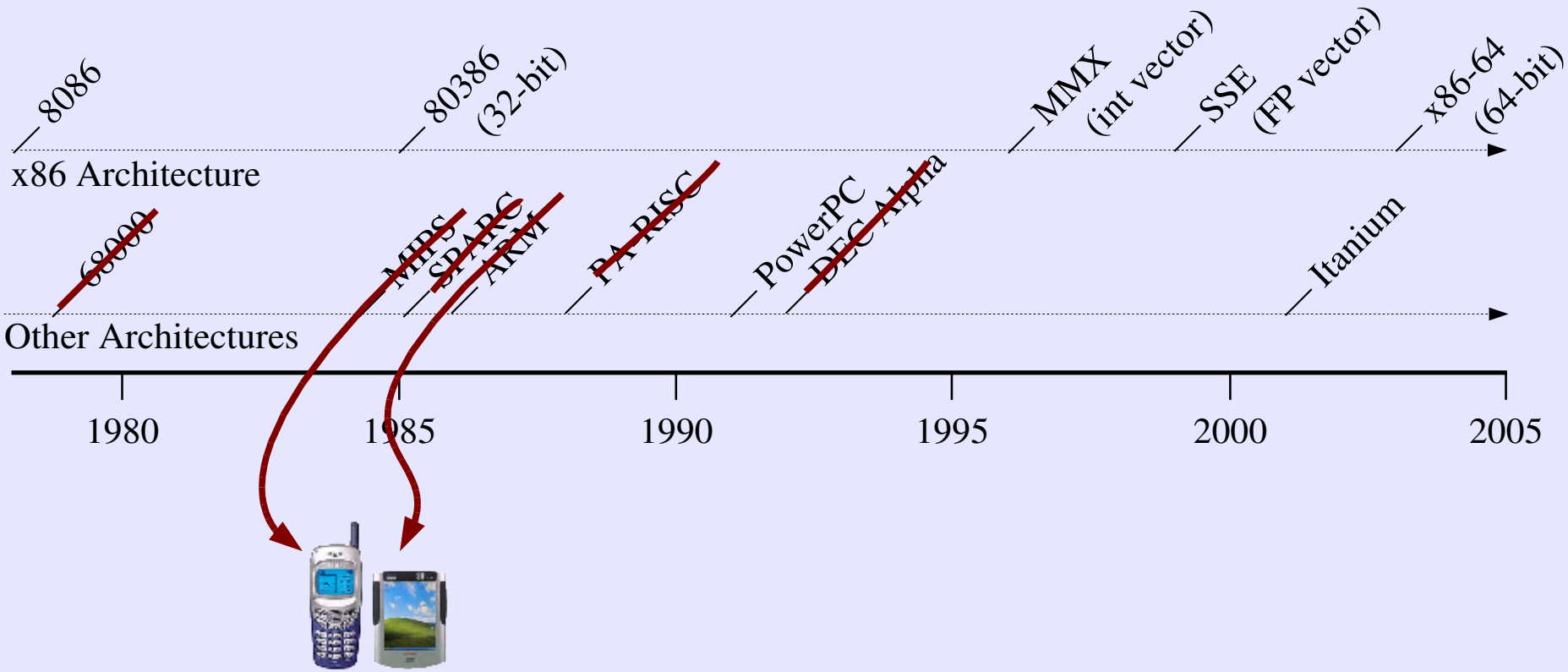
Processor architectures evolve more conservatively



Archiving Compressed Data

Observation #2:

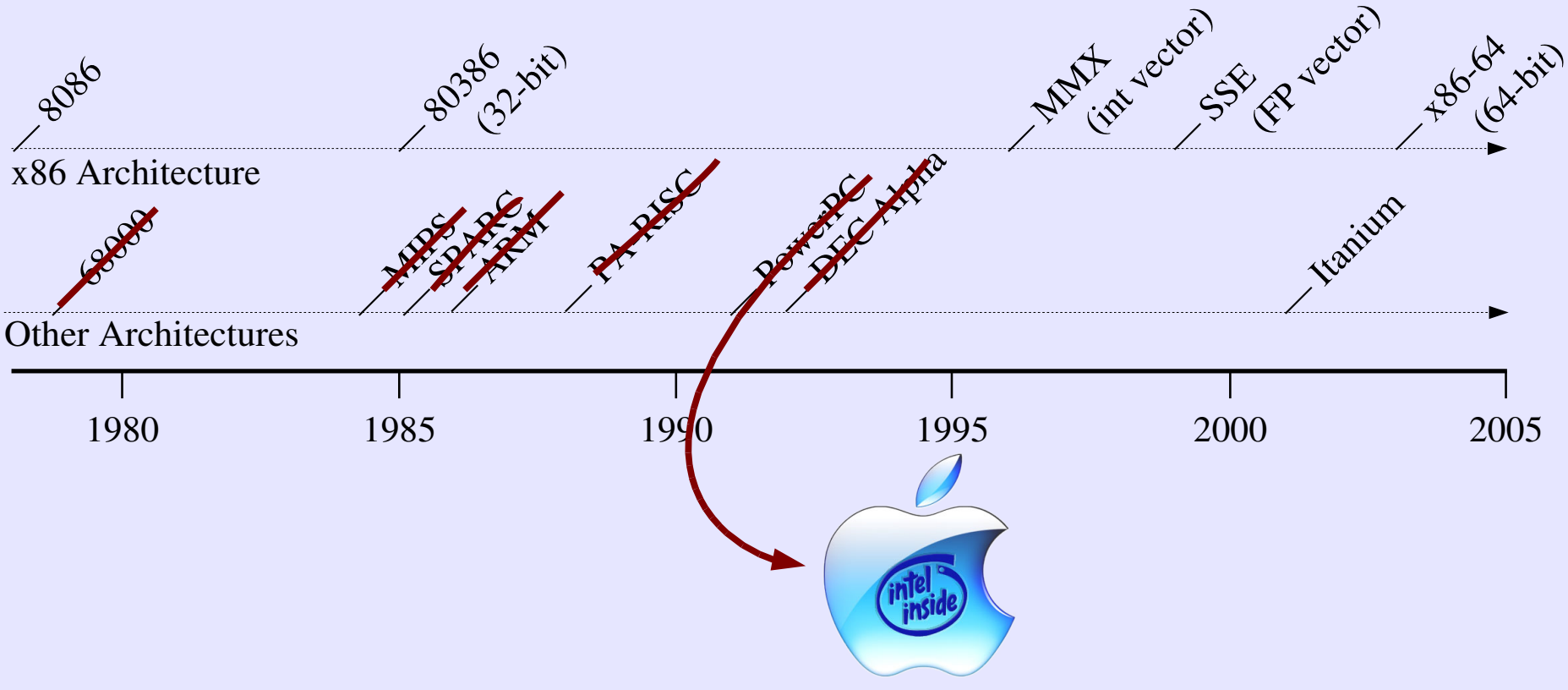
Processor architectures evolve more conservatively



Archiving Compressed Data

Observation #2:

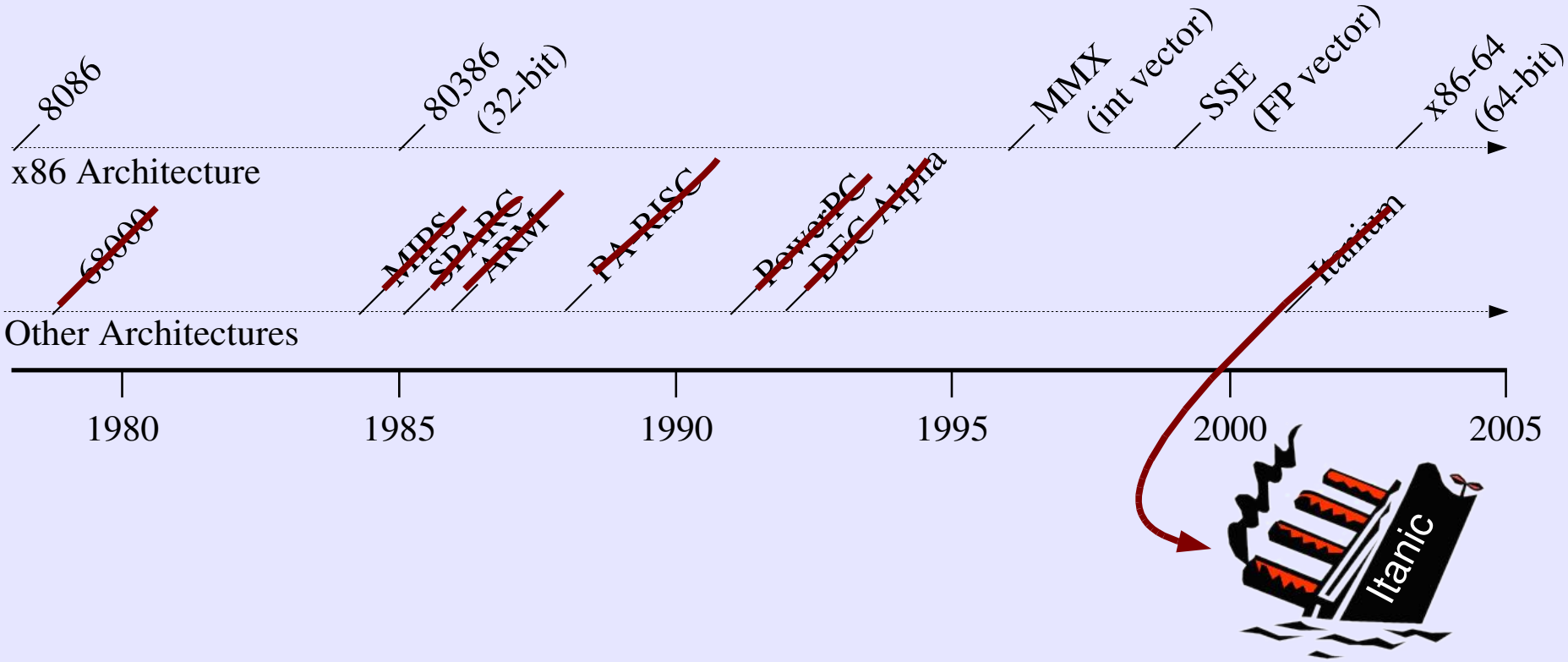
Processor architectures evolve more conservatively



Archiving Compressed Data

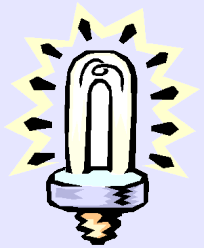
Observation #2:

Processor architectures evolve more conservatively



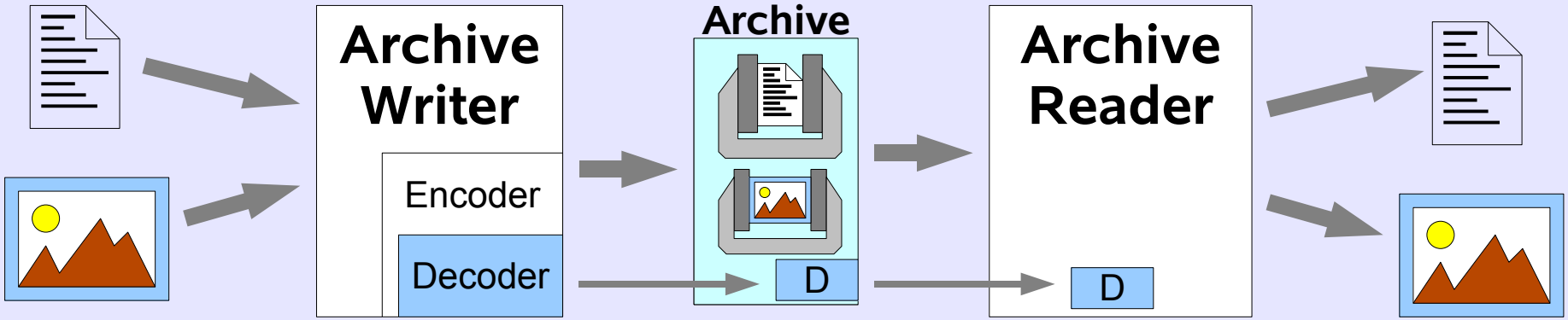
VXA: Virtual Executable Archives

Observation 1+2: Instruction formats are historically more durable than compressed data formats



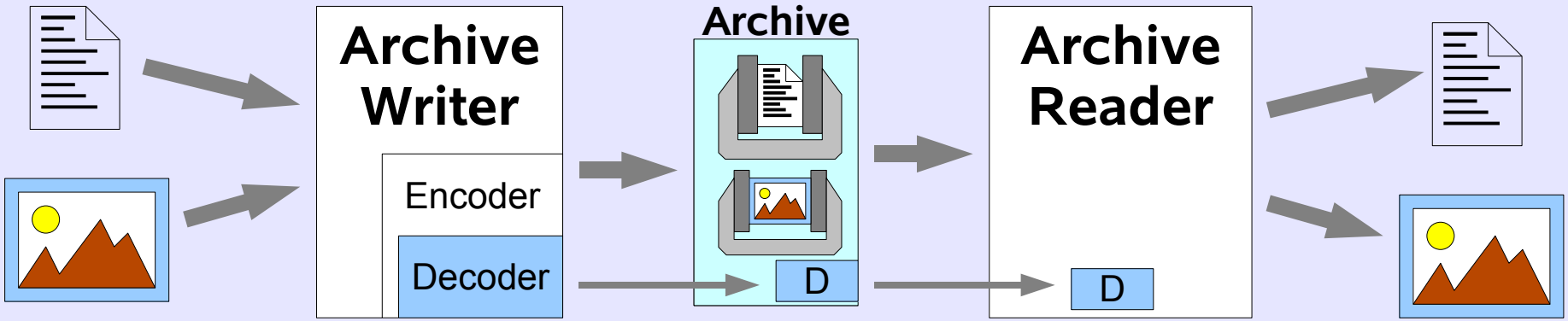
Make archive *self-extracting* (data + executable decoder)

To extract data, archive reader runs embedded decoder



Goals of VXA

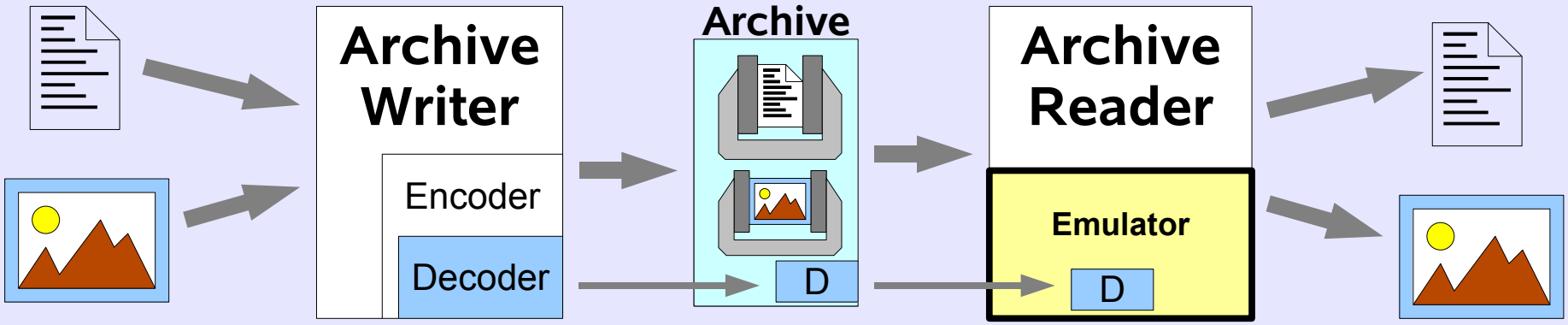
Make self-extracting archives...



Goals of VXA

Make self-extracting archives...

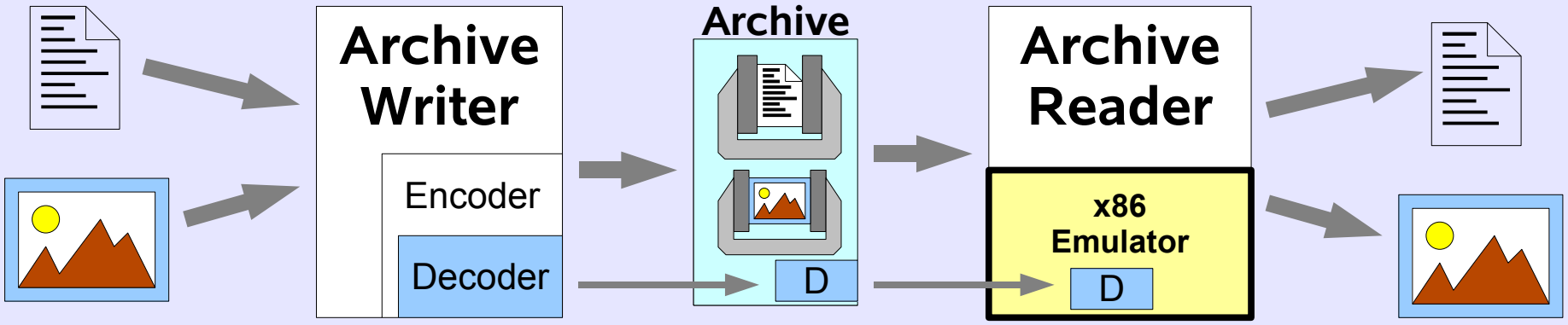
1. **Safe:** malicious decoders can't compromise host
2. **Future-proof:** simple, well-defined architecture [Lorie]



Goals of VXA

Make self-extracting archives...

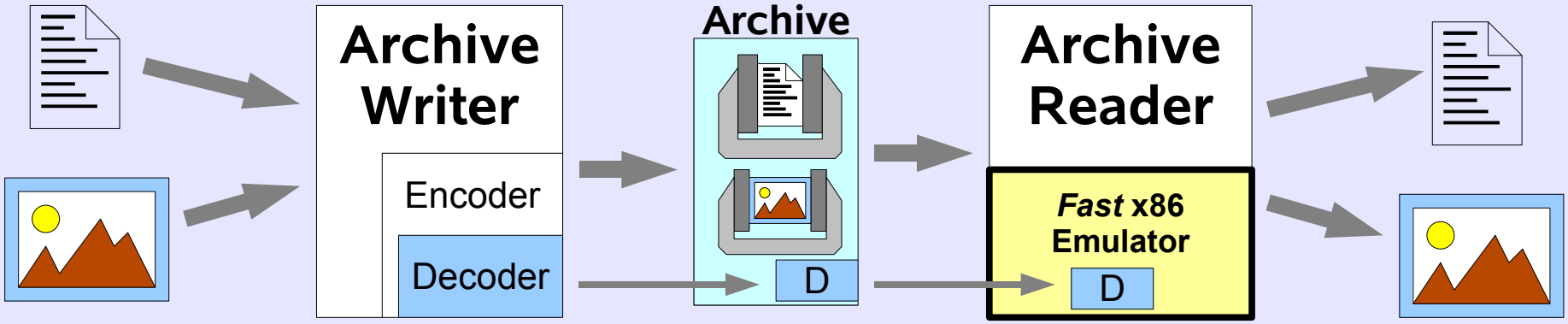
1. **Safe:** malicious decoders can't compromise host
2. **Future-proof:** simple, well-defined architecture [Lorie]
3. **Easy:** allow reuse of existing code, languages, tools



Goals of VXA

Make self-extracting archives...

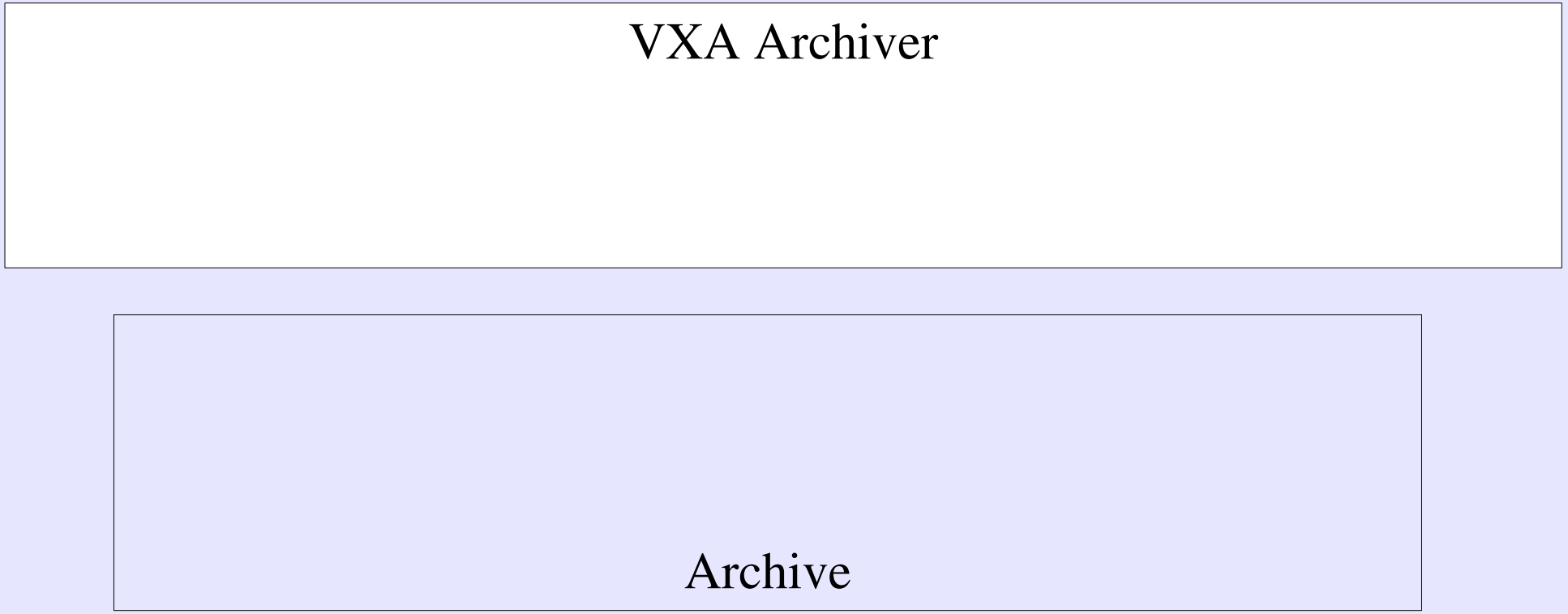
1. **Safe:** malicious decoders can't compromise host
2. **Future-proof:** simple, well-defined architecture [Lorie]
3. **Easy:** allow reuse of existing code, languages, tools
4. **Efficient:** practical for *short term* data packaging too



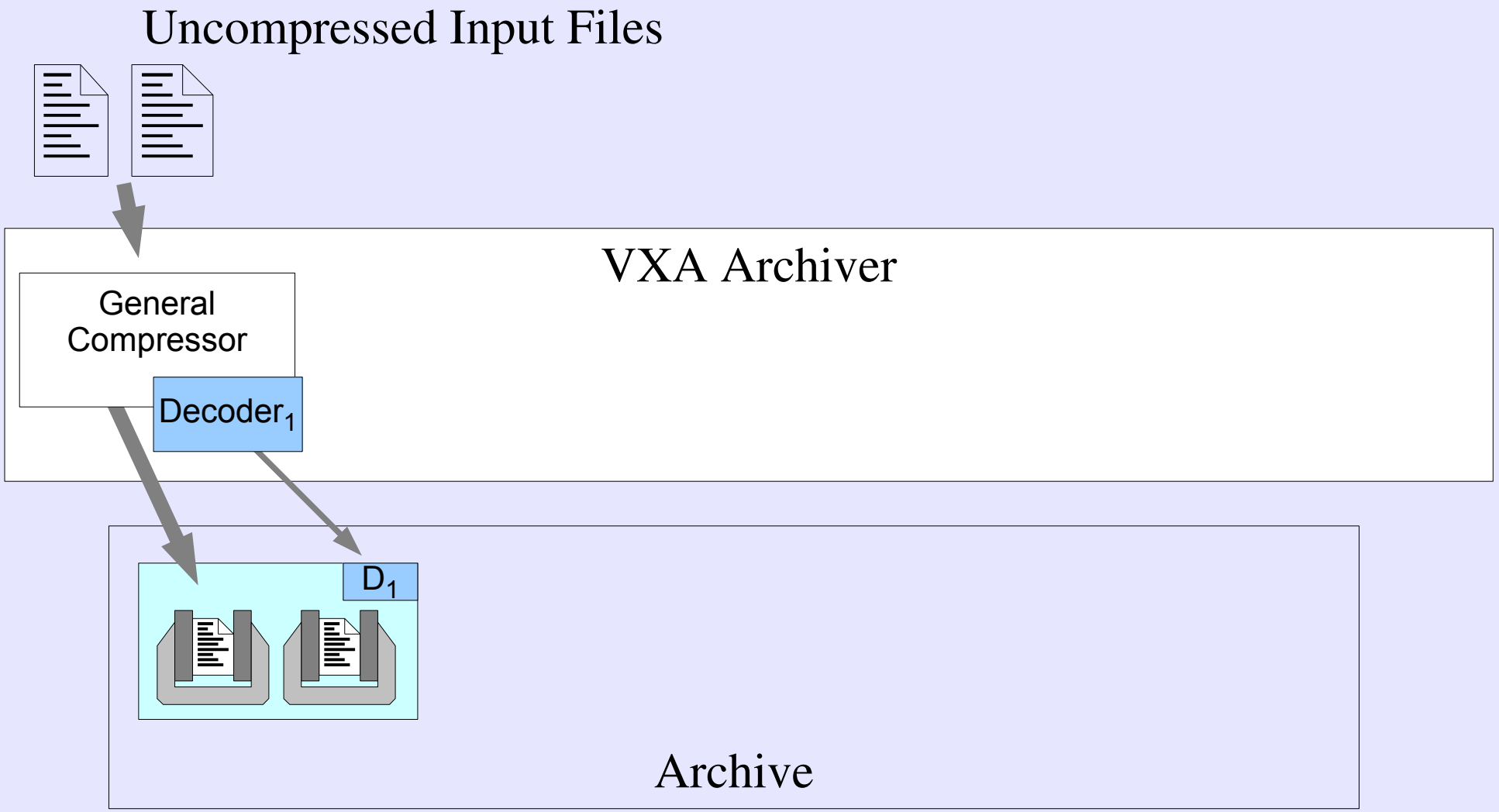
Outline

- Archiver Operation
- vxZIP Archive Format
- Decoder Architecture
- Emulator Design & Implementation
- Evaluation (performance, storage overhead)
- Conclusion

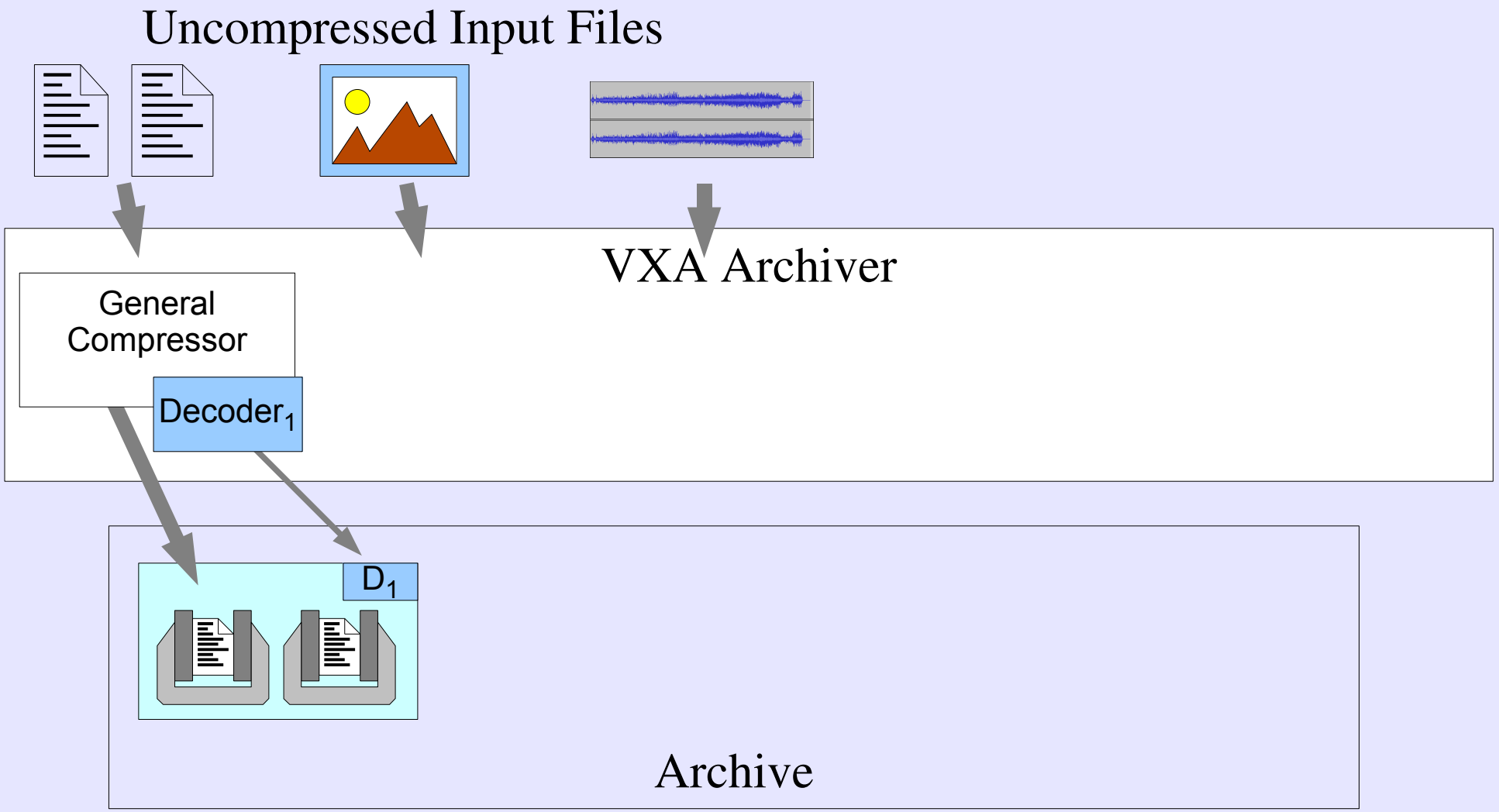
Archive Writer Operation



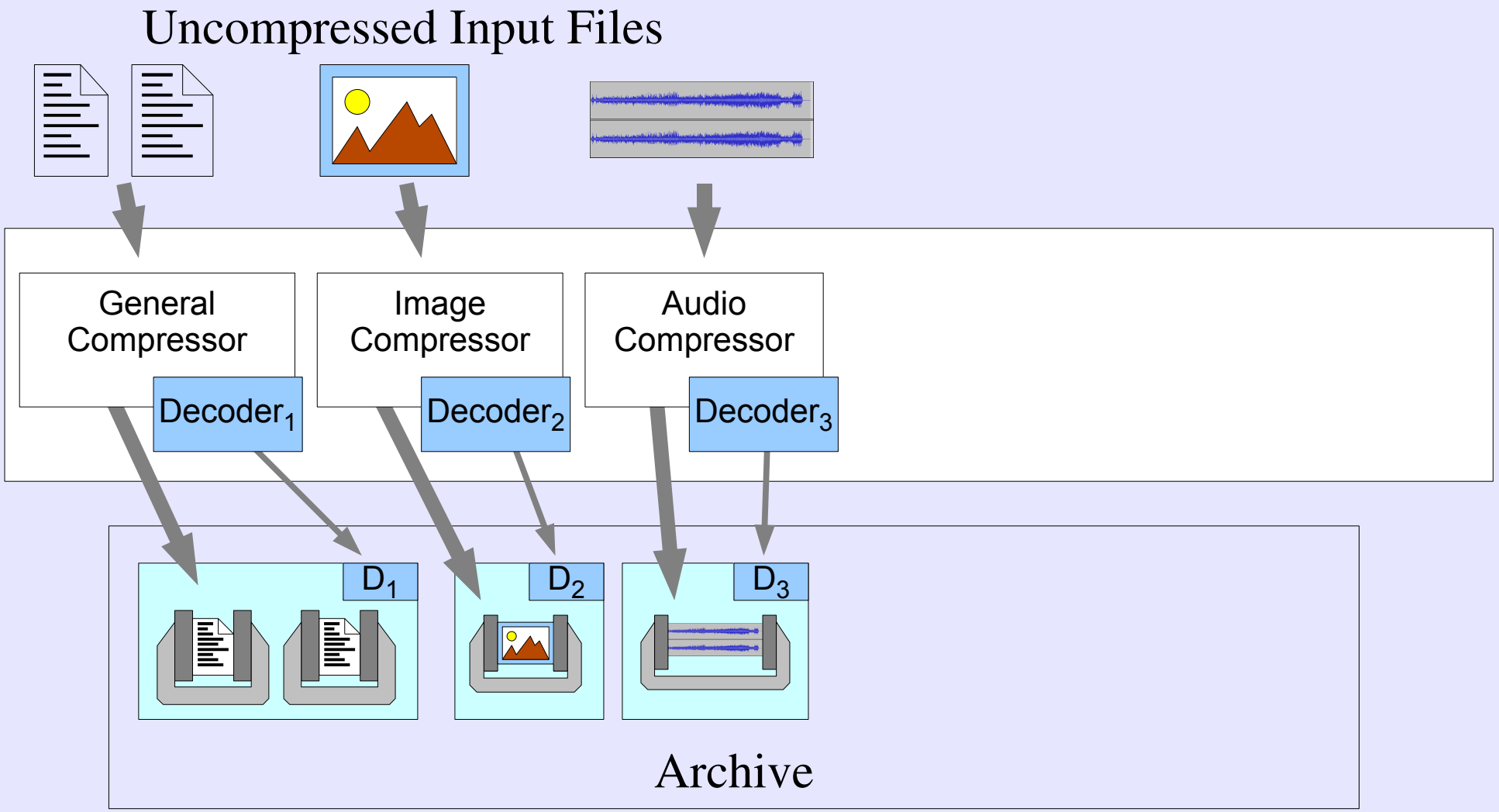
Archive Writer Operation



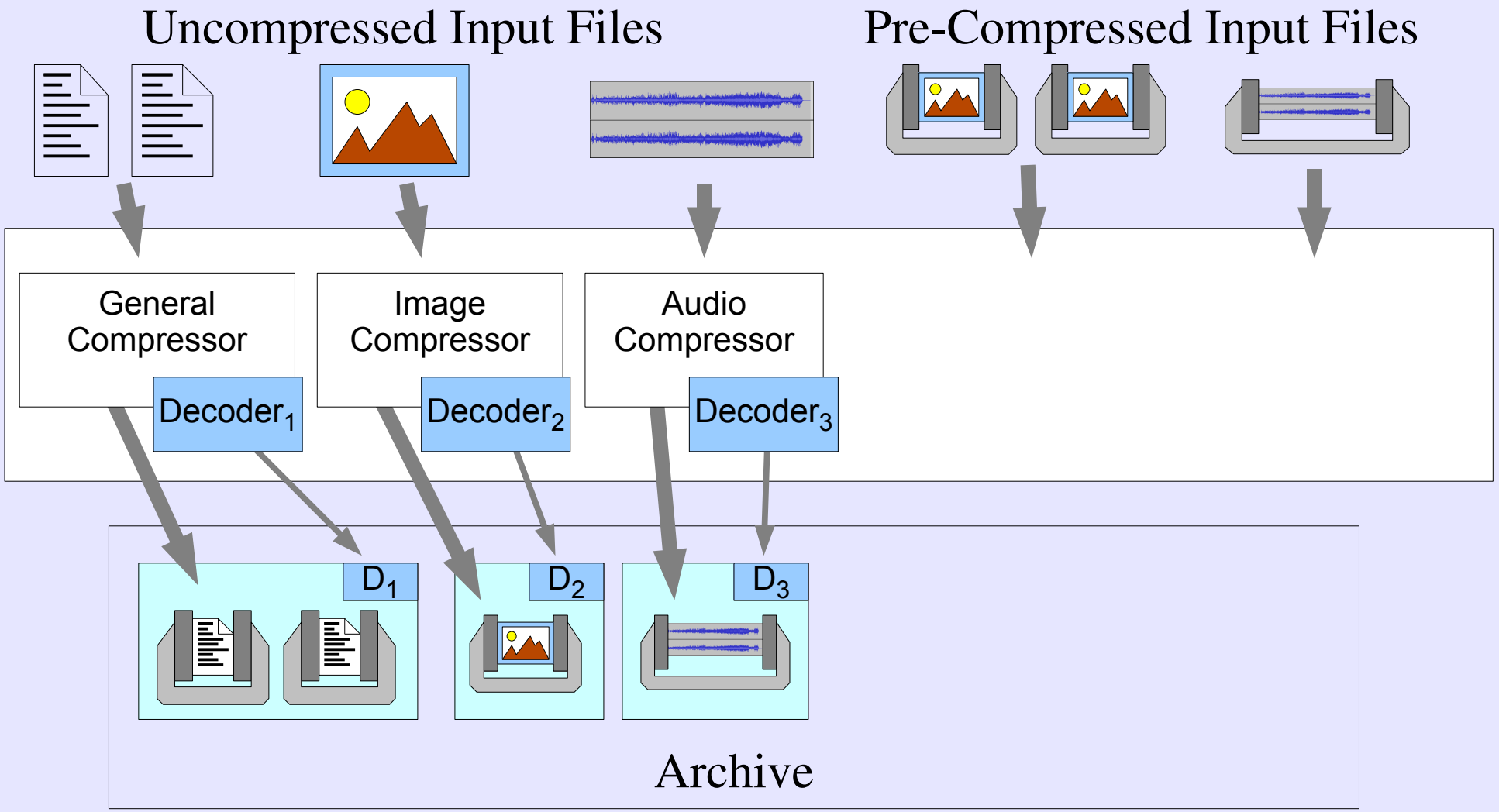
Archive Writer Operation



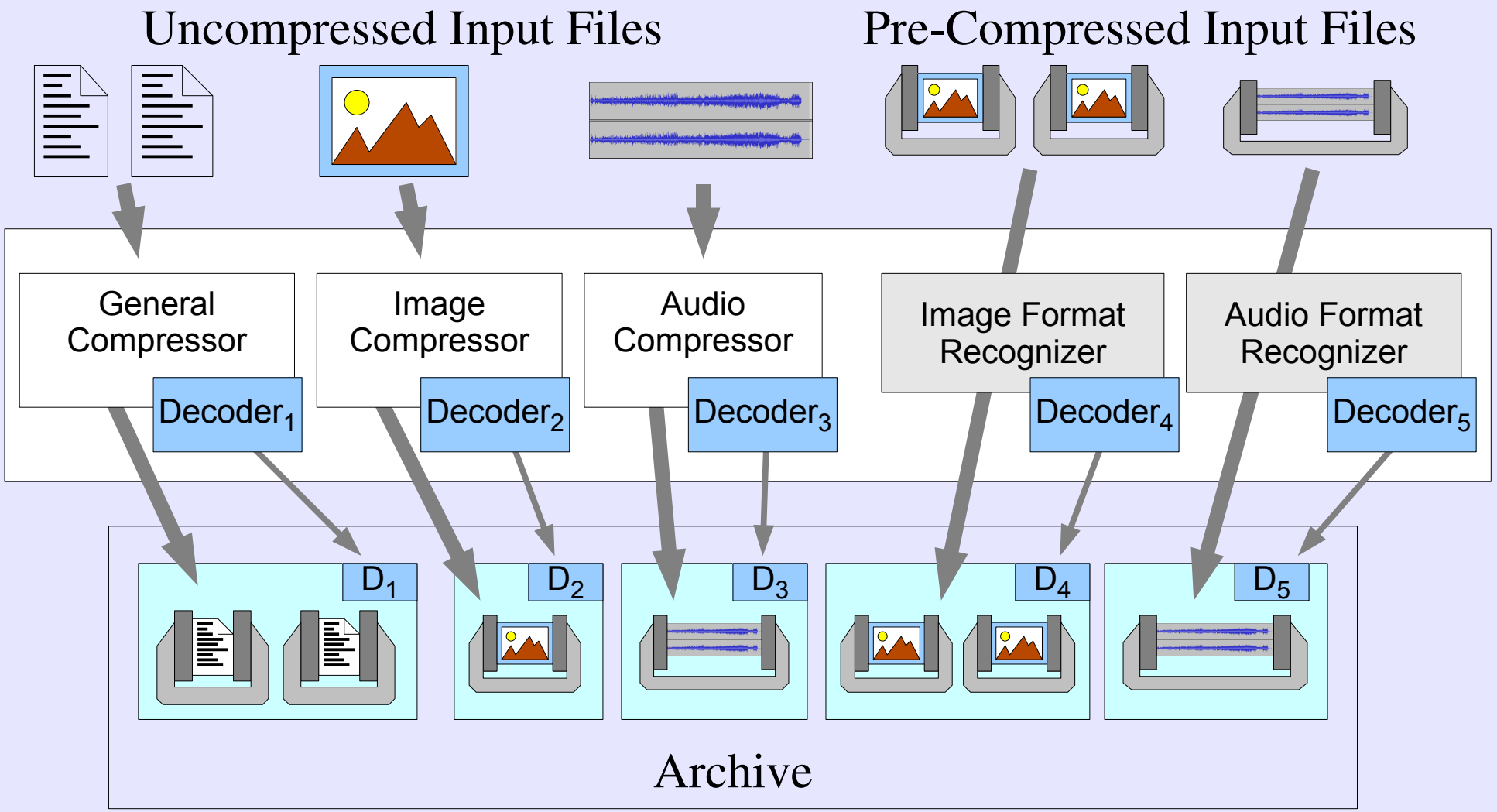
Archive Writer Operation



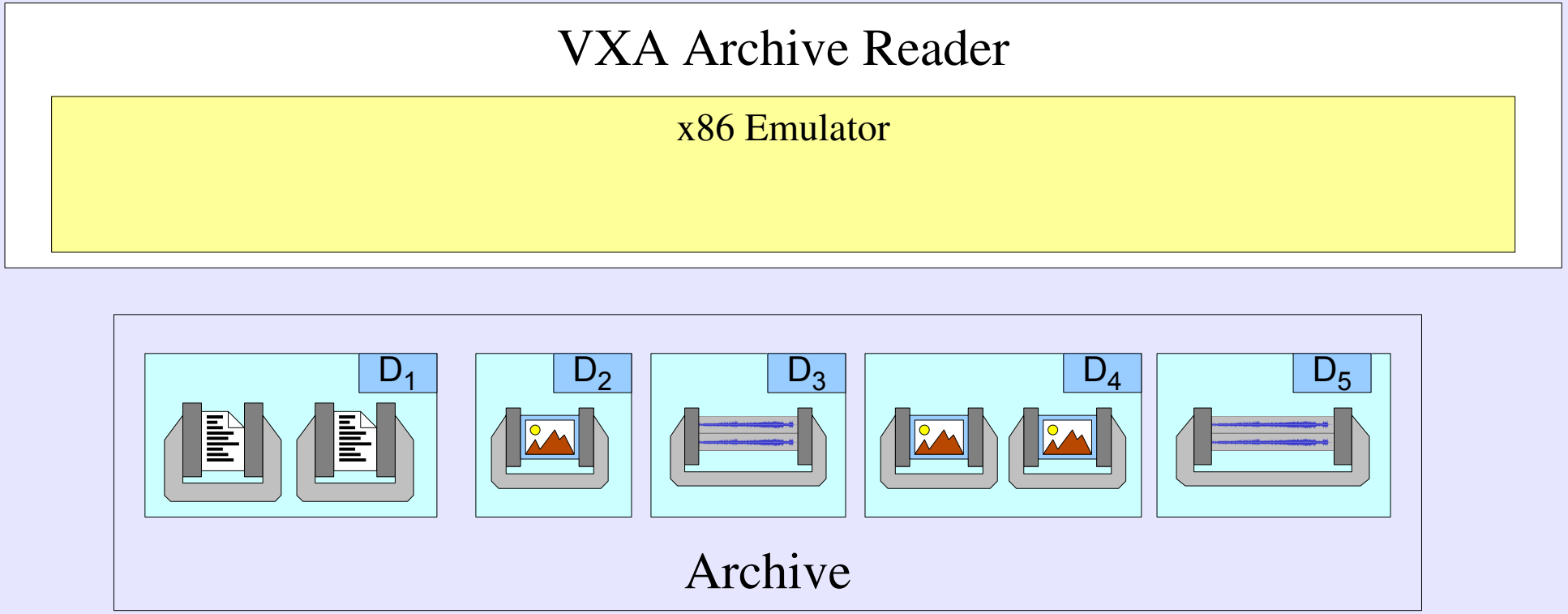
Archive Writer Operation



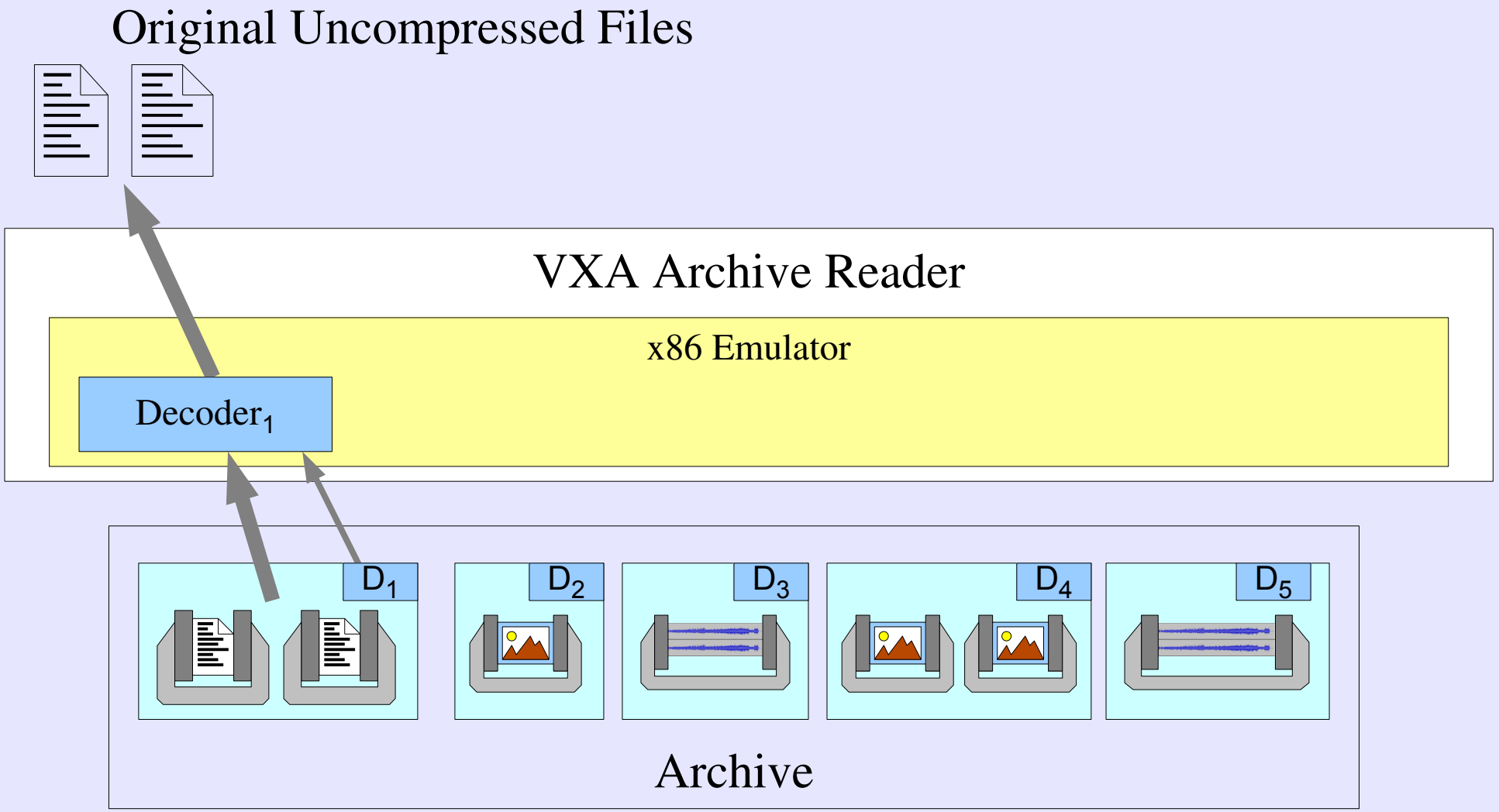
Archive Writer Operation



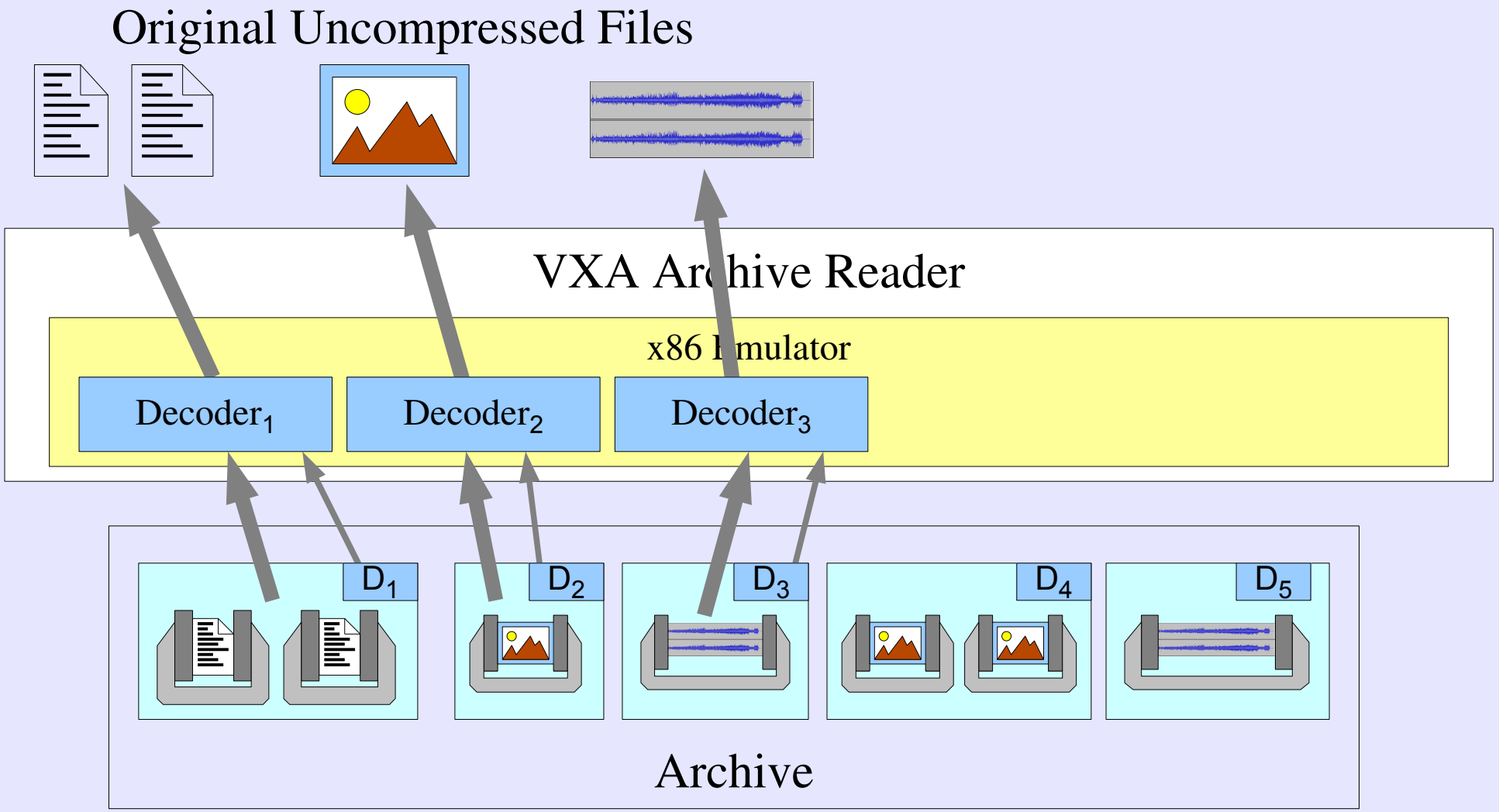
Archive Reader Operation



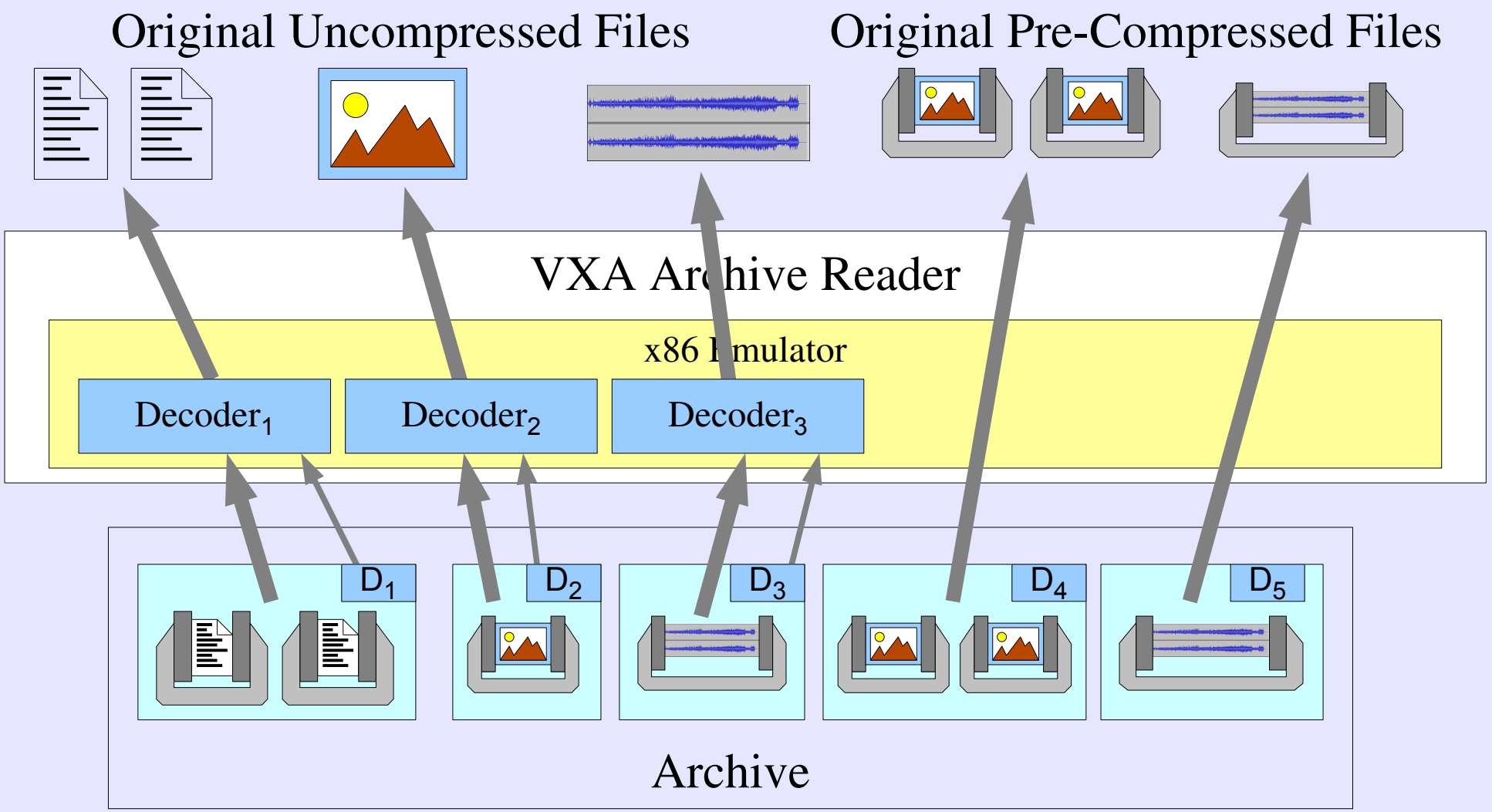
Archive Reader Operation



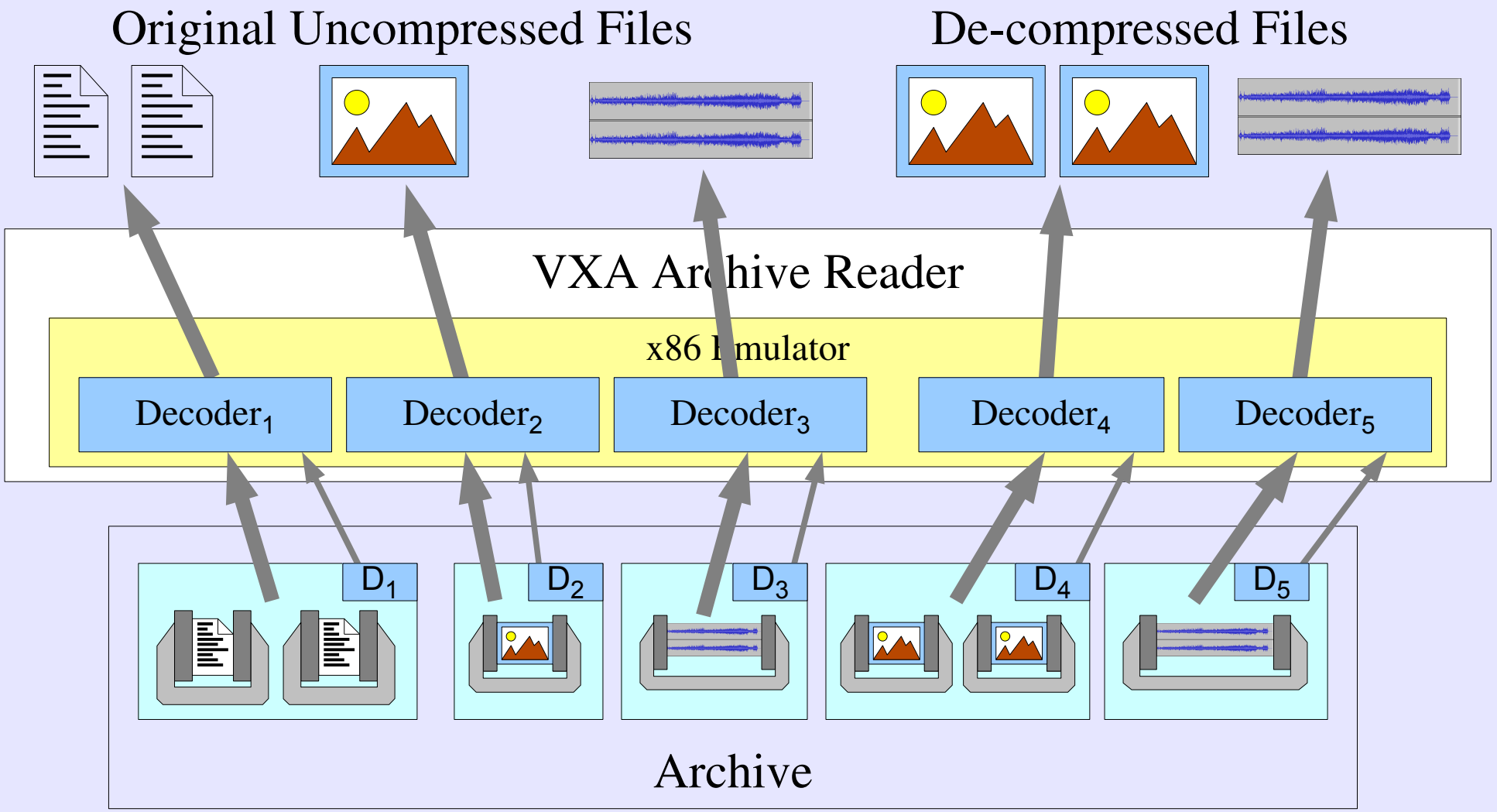
Archive Reader Operation



Archive Reader Operation

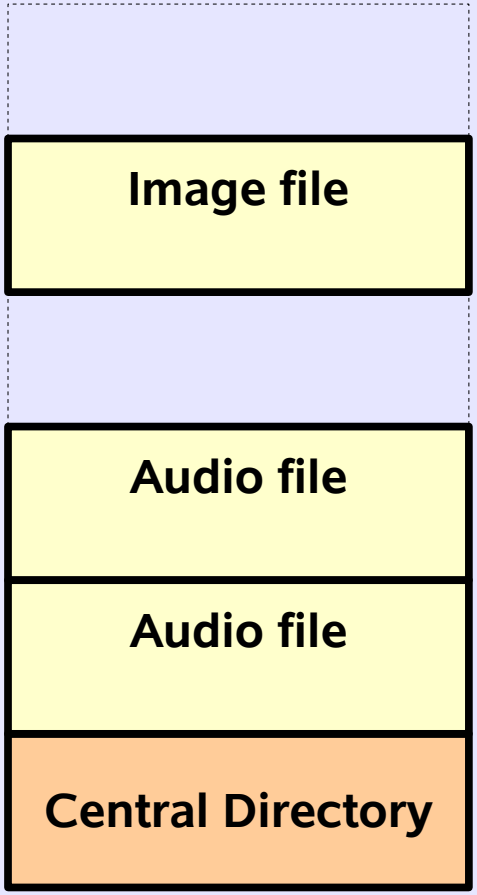


Archive Reader Operation



vxZIP Archive Format

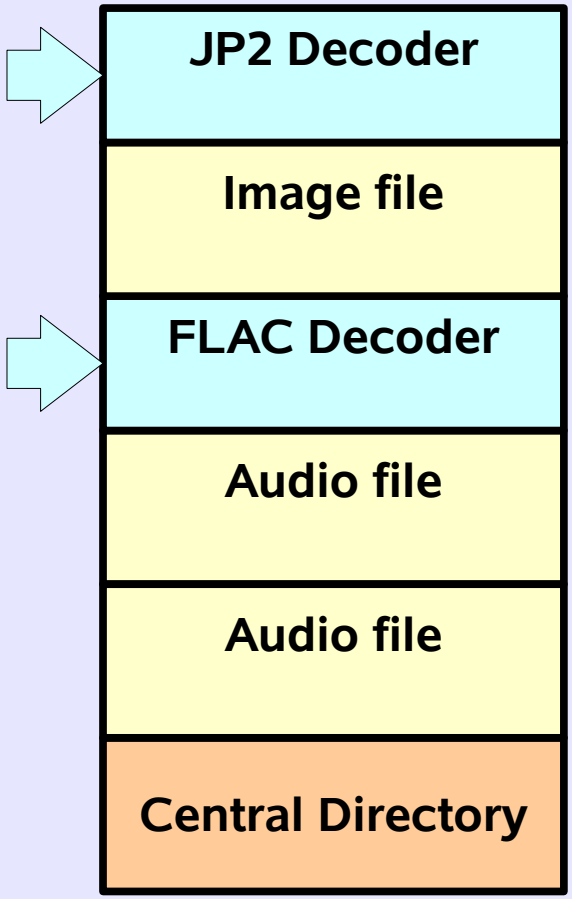
- Backward compatible with legacy ZIP format



vxZIP Archive

vxZIP Archive Format

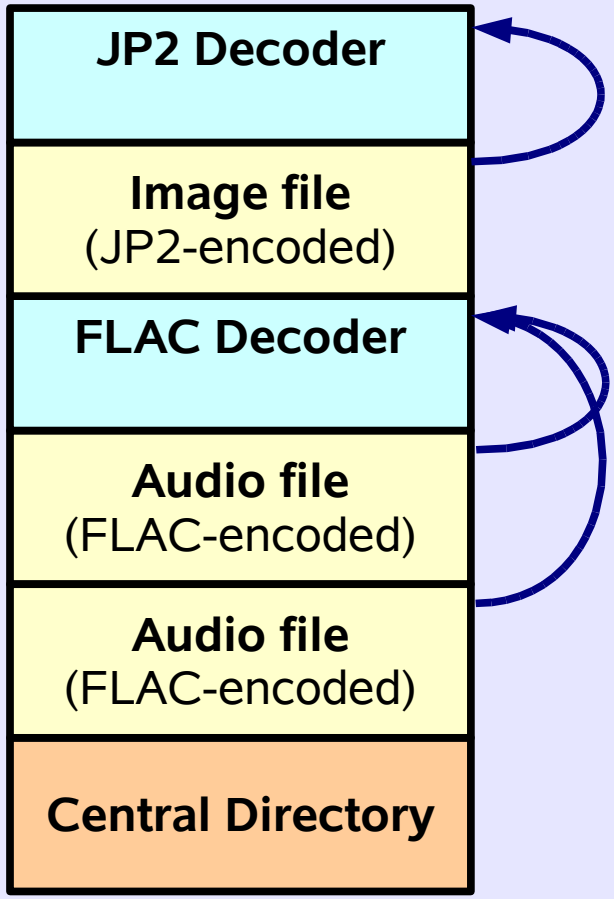
- Backward compatible with legacy ZIP format
- Decoders intermixed with archived files



vxZIP Archive

vxZIP Archive Format

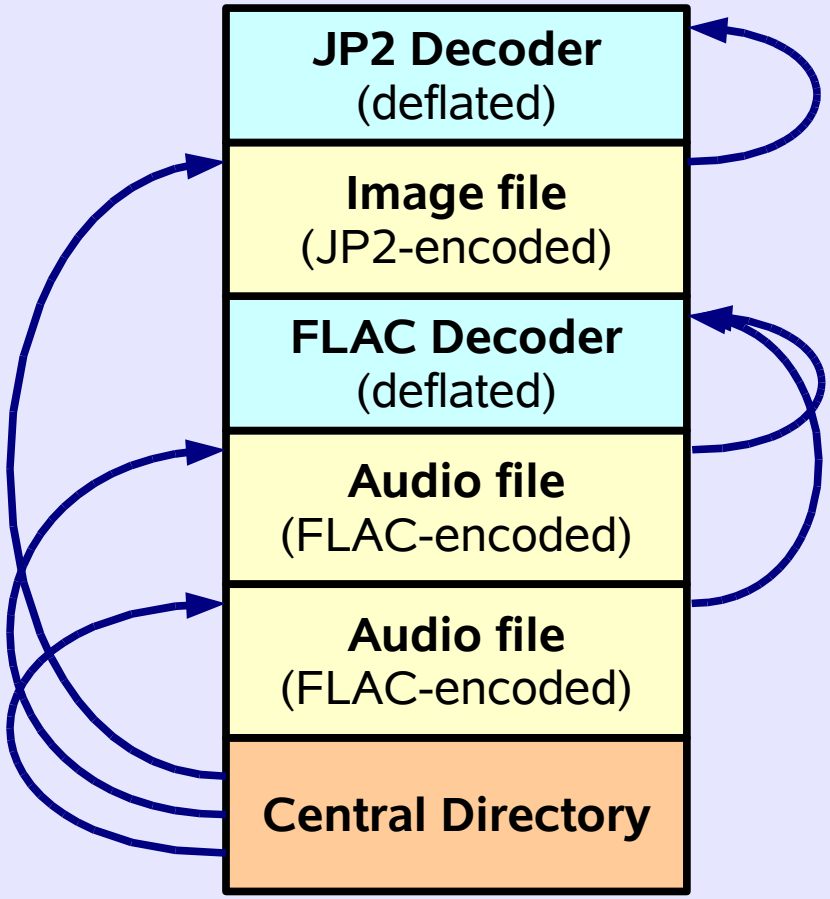
- Backward compatible with legacy ZIP format
- Decoders intermixed with archived files
- Archived files have new extension header pointing to decoder



vxZIP Archive

vxZIP Archive Format

- Backward compatible with legacy ZIP format
- Decoders intermixed with archived files
- Archived files have new extension header pointing to decoder
- Decoders are hidden, “deflated” (gzip)



vxZIP Archive

vxZIP Decoder Architecture

- Decoders are ELF executables for x86-32
  - Can be written in any language, safe or unsafe
  - Compiled using ordinary tools (GCC)
- Decoders have access to five “system calls”:
  - *read stdin, write stdout, malloc, next file, exit*
- Decoders **cannot**:
  - open files, windows, devices, network connections, ...
  - get system info: user name, current time, OS type, ...

Decoders Ported So Far

(using existing implementations in C, mostly unmodified)

General-purpose (lossless):

- **zlib:** Classic gzip/deflate algorithm
- **bzip2:** Burrows-Wheeler algorithm

Still image codecs:

- **jpeg:** Classic lossy image compression scheme
- **jp2:** JPEG 2000 wavelet-based algorithm, lossy or lossless

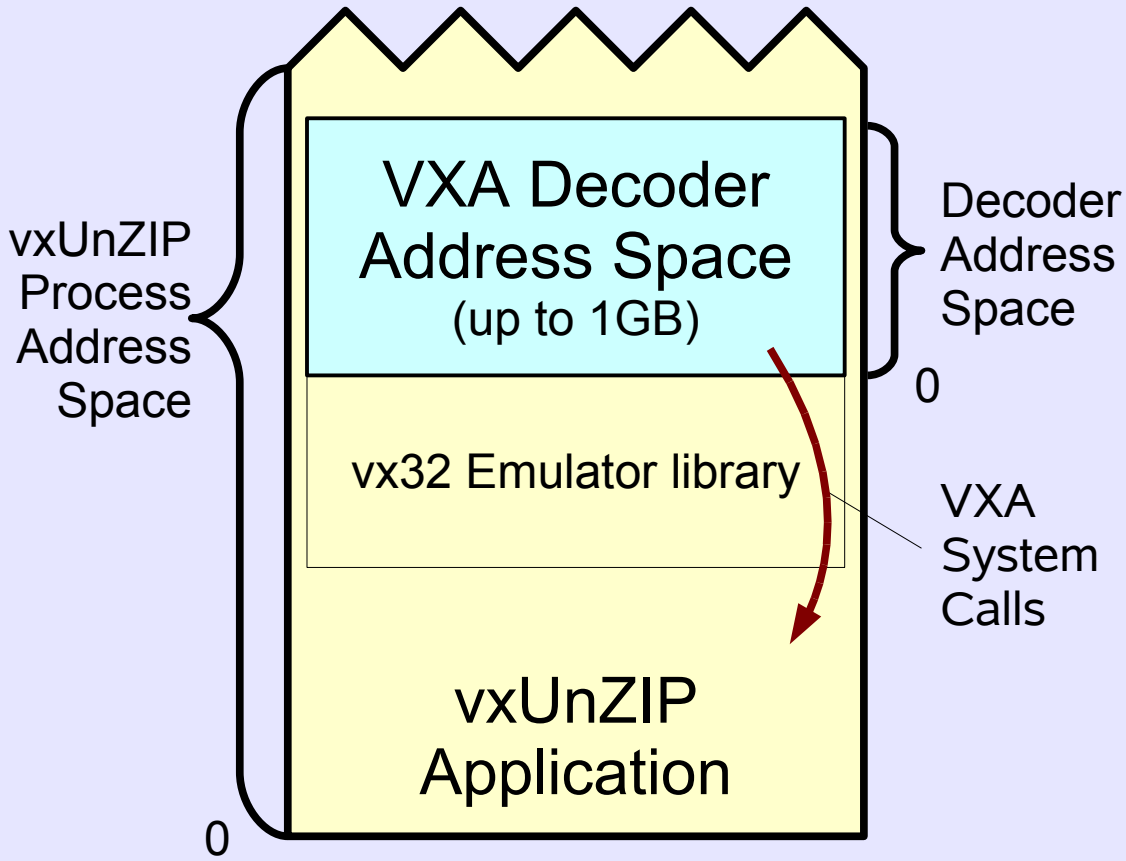
Audio codecs:

- **flac:** Free Lossless Audio Codec
- **vorbis:** Standard lossy audio codec for Ogg streams

vx32 Emulator Architecture

Runs in vxUnZIP process

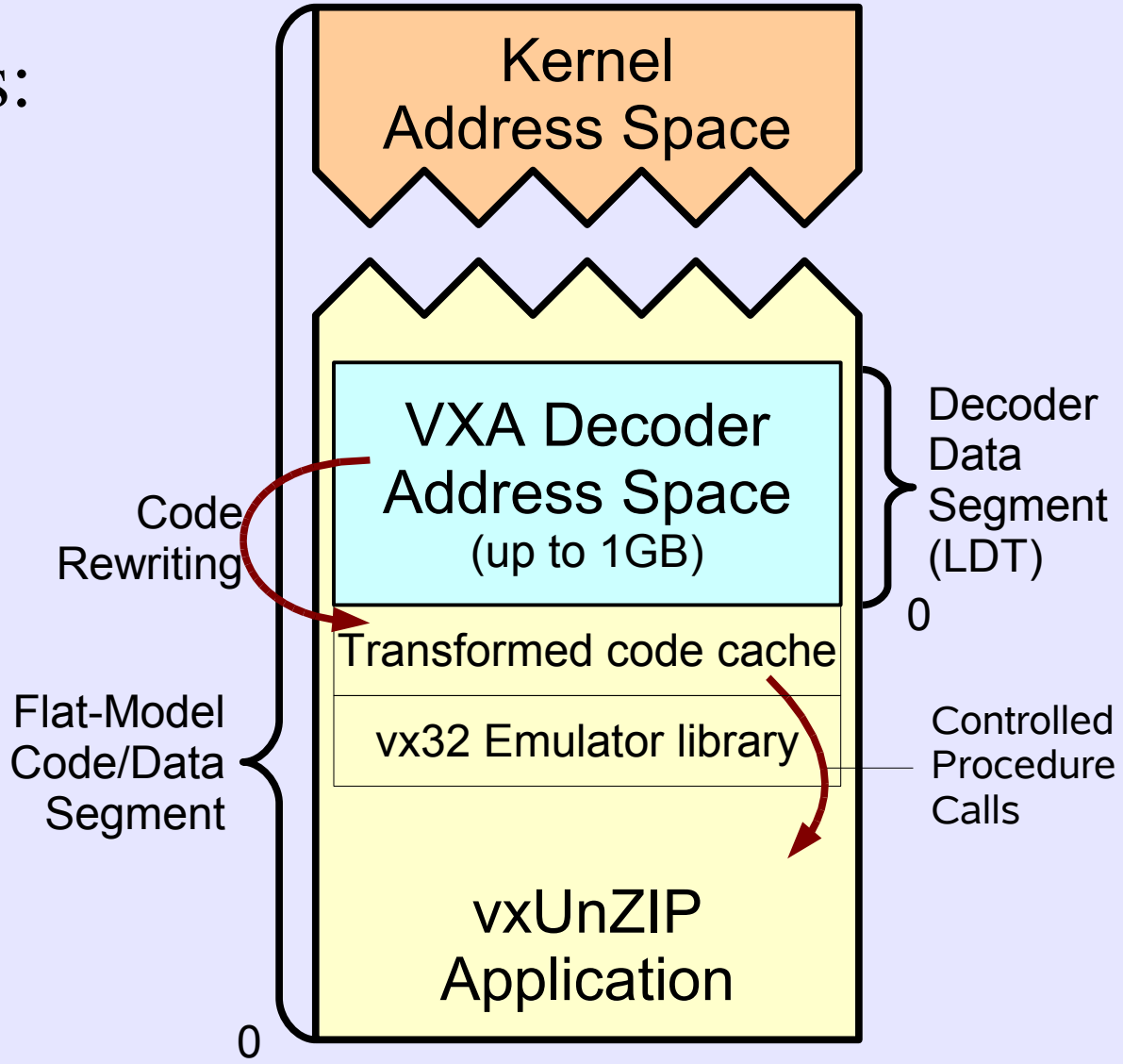
- Loads decoder into address space sandbox
- Restricts decoder's memory accesses to sandbox
- Dispatches decoder's VXA “system calls” to vxUnZIP (*not* to host OS!)



vx32 Emulator Implementation

On x86-{32/64} hosts:

- Secure fault isolation [Wahbe]
- Data sandboxing via custom LDT segments
- Code sandboxing via instruction rewriting [Sites, Nethercote]
- No privileges or kernel extensions



vx32 Emulator Implementation

On other host architectures:

- Portable but slow “fallback” instruction interpreter (mostly done)
- Fast x86-to-PowerPC binary translator (in progress)
- Hopefully more in the future

Emulator implemented as generic library

- Can be used for other sandboxing applications

Evaluation

Two issues to address:

- Performance overhead of emulated decoders
  - not important for long-term archival storage, but...
  - *very* important for common short-term uses of archives: backups, software distribution, structured documents, ...
- Storage overhead of archived decoders

Performance Test Method

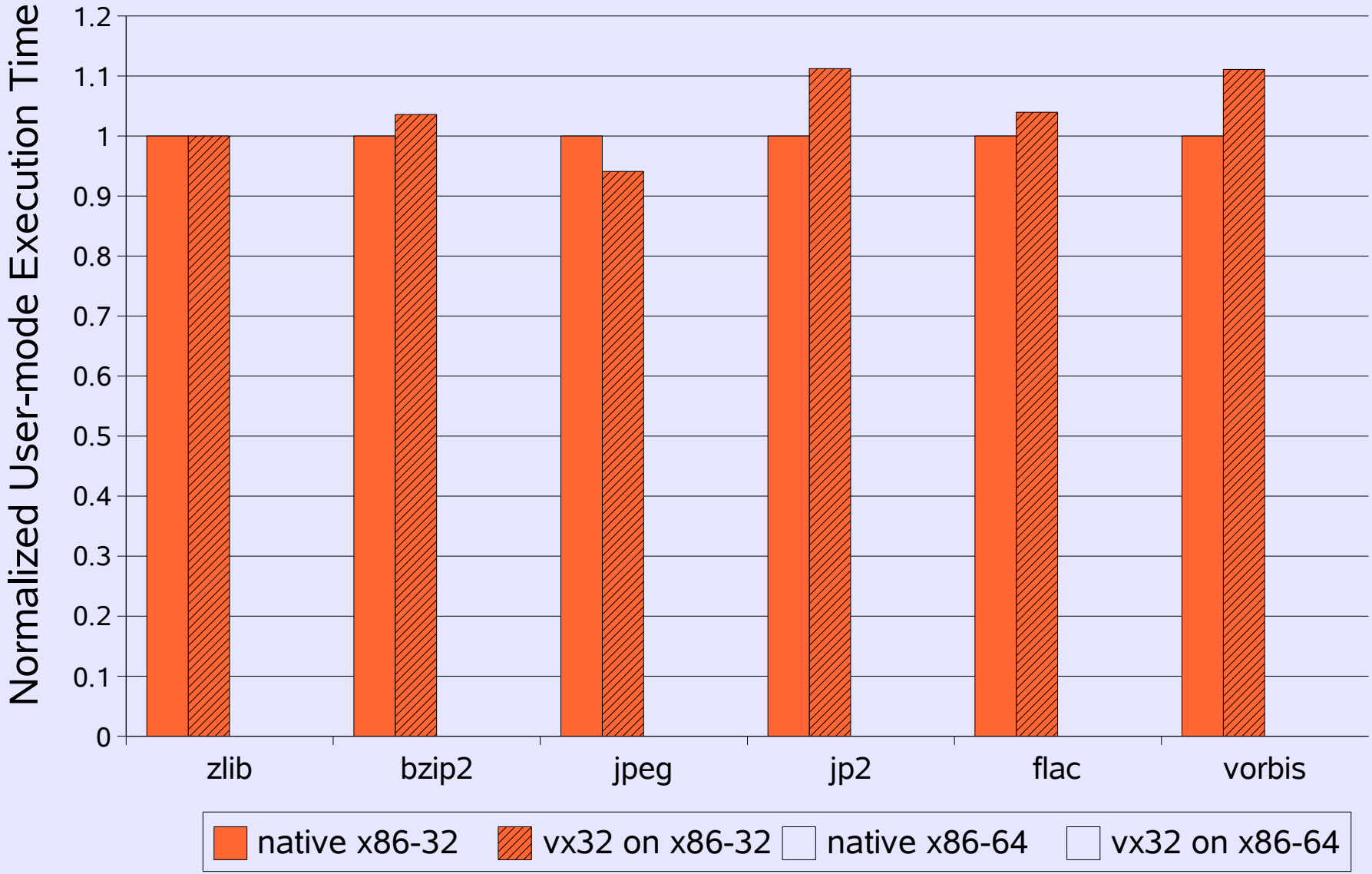
Run 6 ported decoders on appropriate data sets

- Athlon 64 3000+ PC running SuSE Linux 9.3
- Measure *user-mode CPU time* (not wall-clock time)

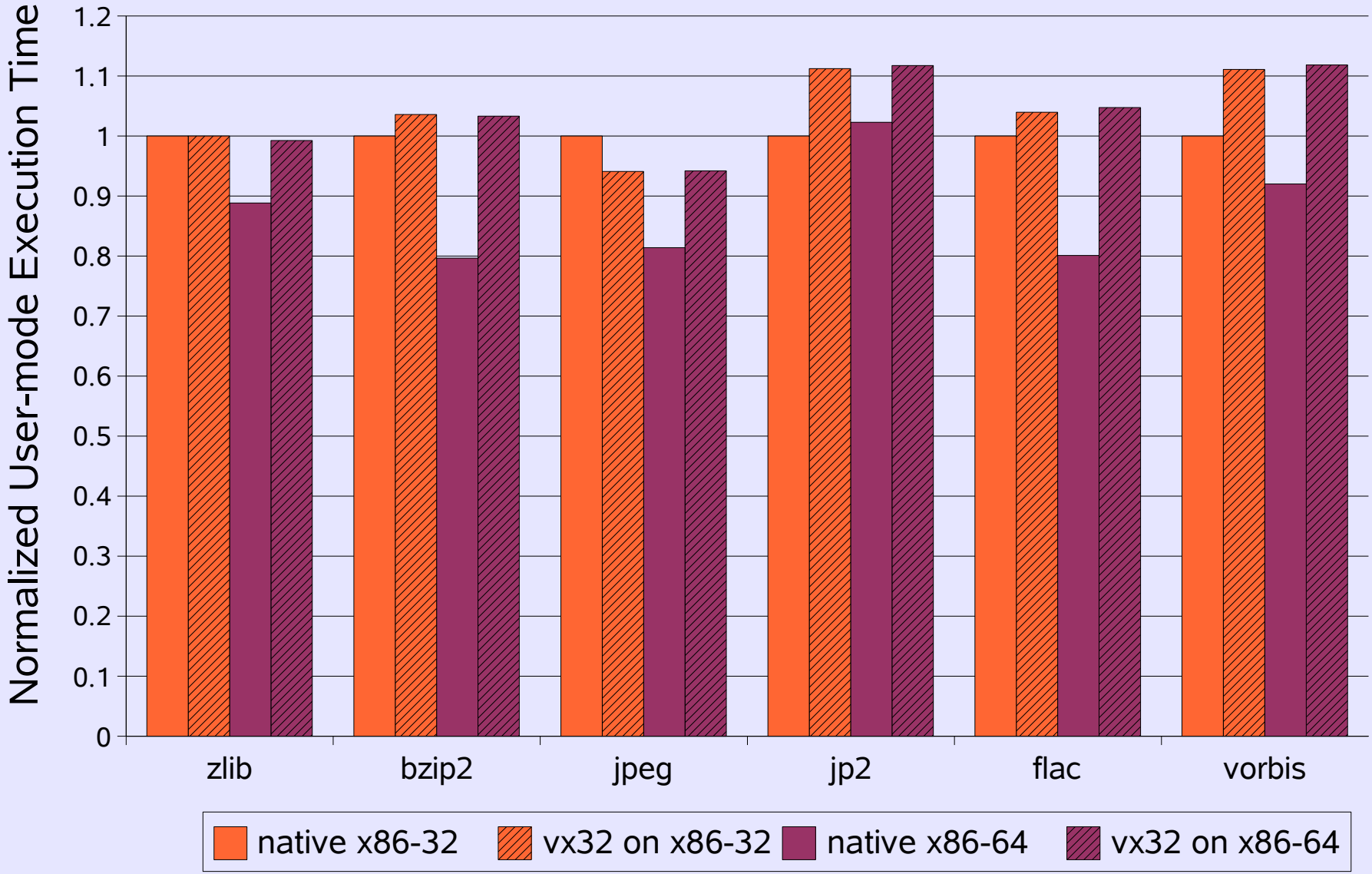
Compare:

- Emulated vs native execution
- Running on x86-32 vs x86-64 host environment

Performance Overhead



Performance Overhead



Storage Overhead

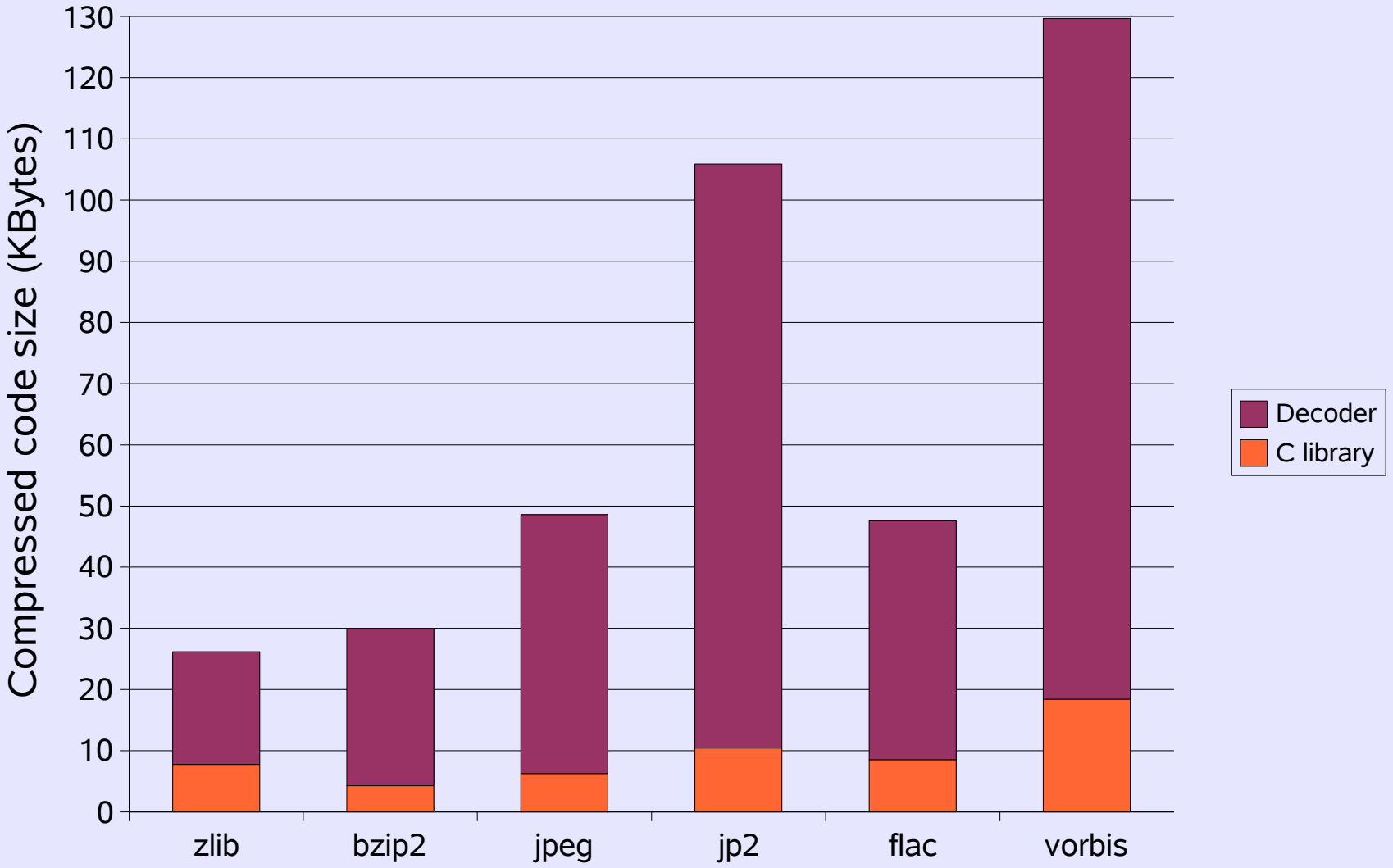
Archiver stores only one copy of each decoder

- Storage cost amortized over all files of same type
- Relative overhead depends on size of archive

Therefore, measure only *absolute* decoder size

(compressed, as stored in archive)

Storage Overhead



Conclusion

VXA makes self-extracting archives...

- **Safe:** decoders fully sandboxed
- **Future-proof:** simple, OS-independent environment
- **Easy:** re-use existing decoders, languages, tools
- **Efficient:** ≤ 11% slowdown vs native x86-32

Available at: http://pdos.csail.mit.edu/~baford/vxa/