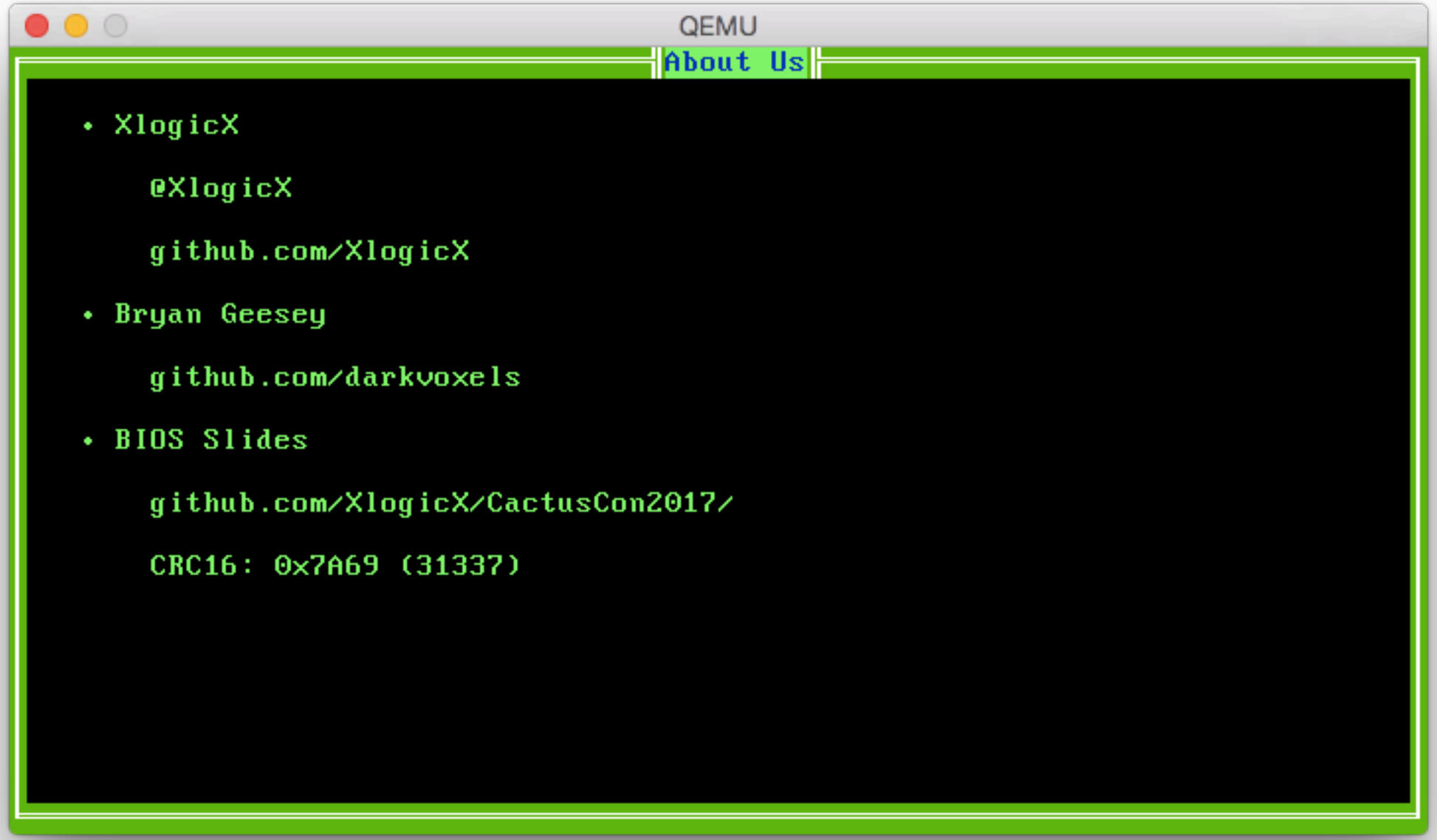
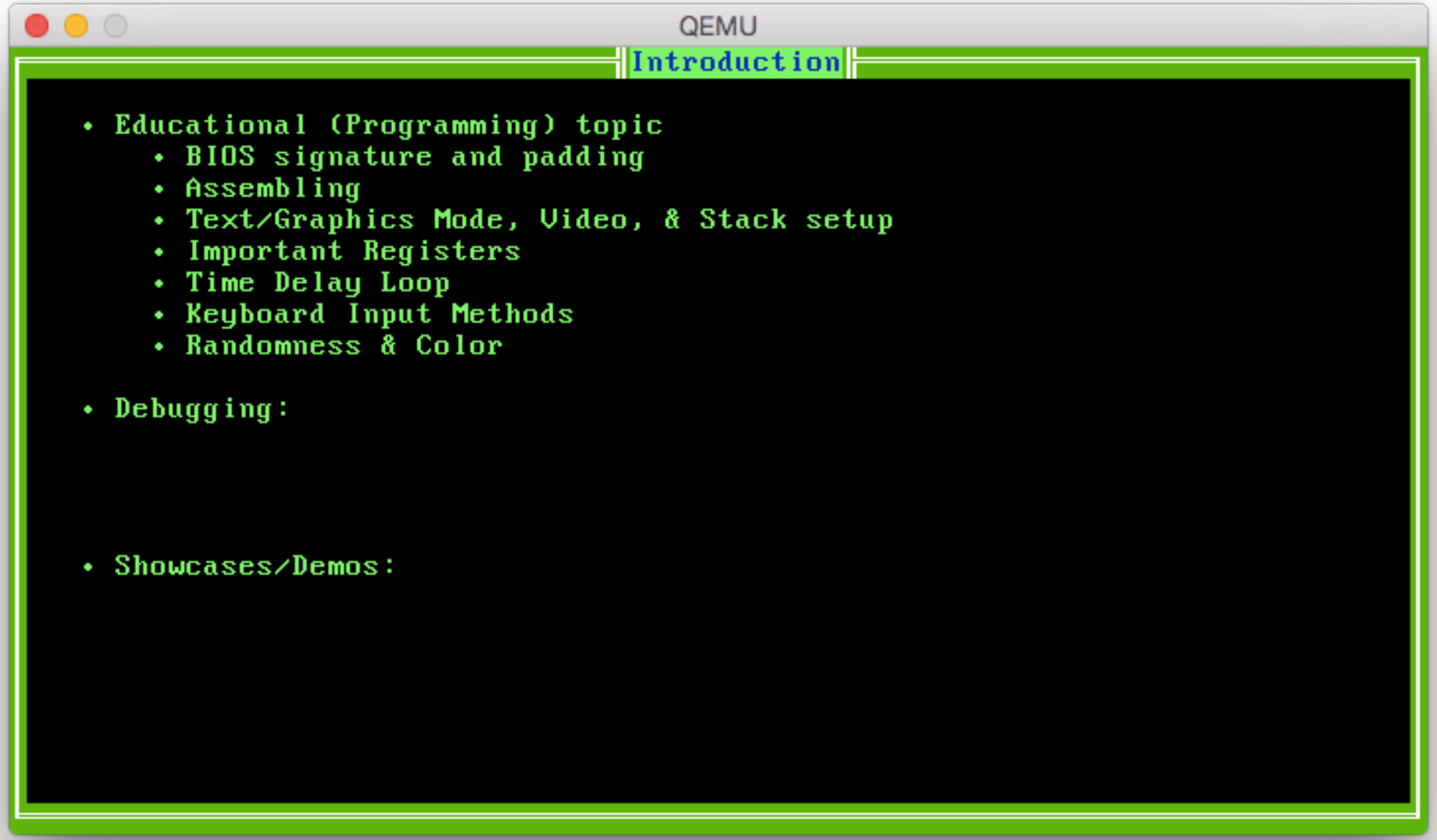
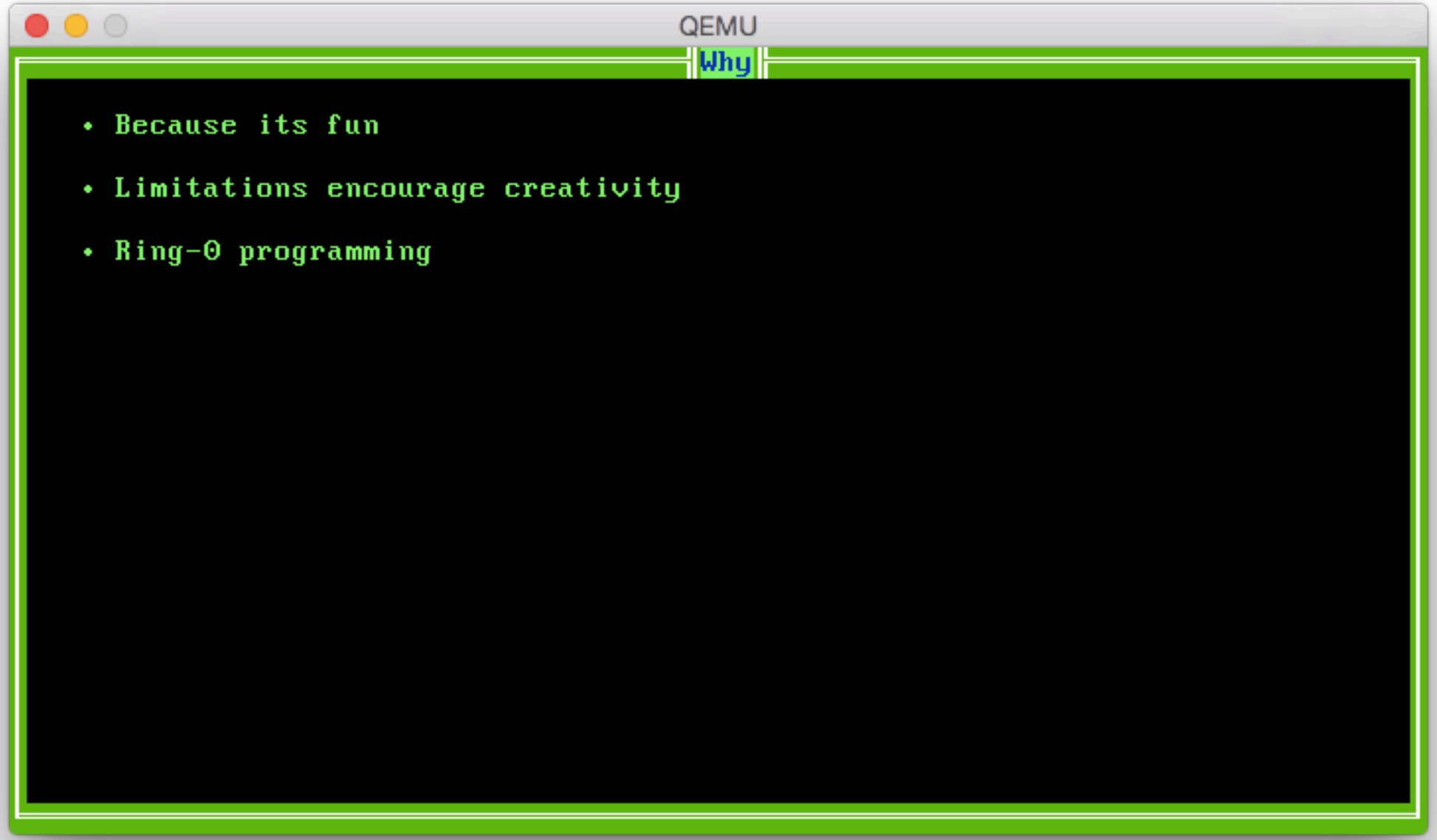
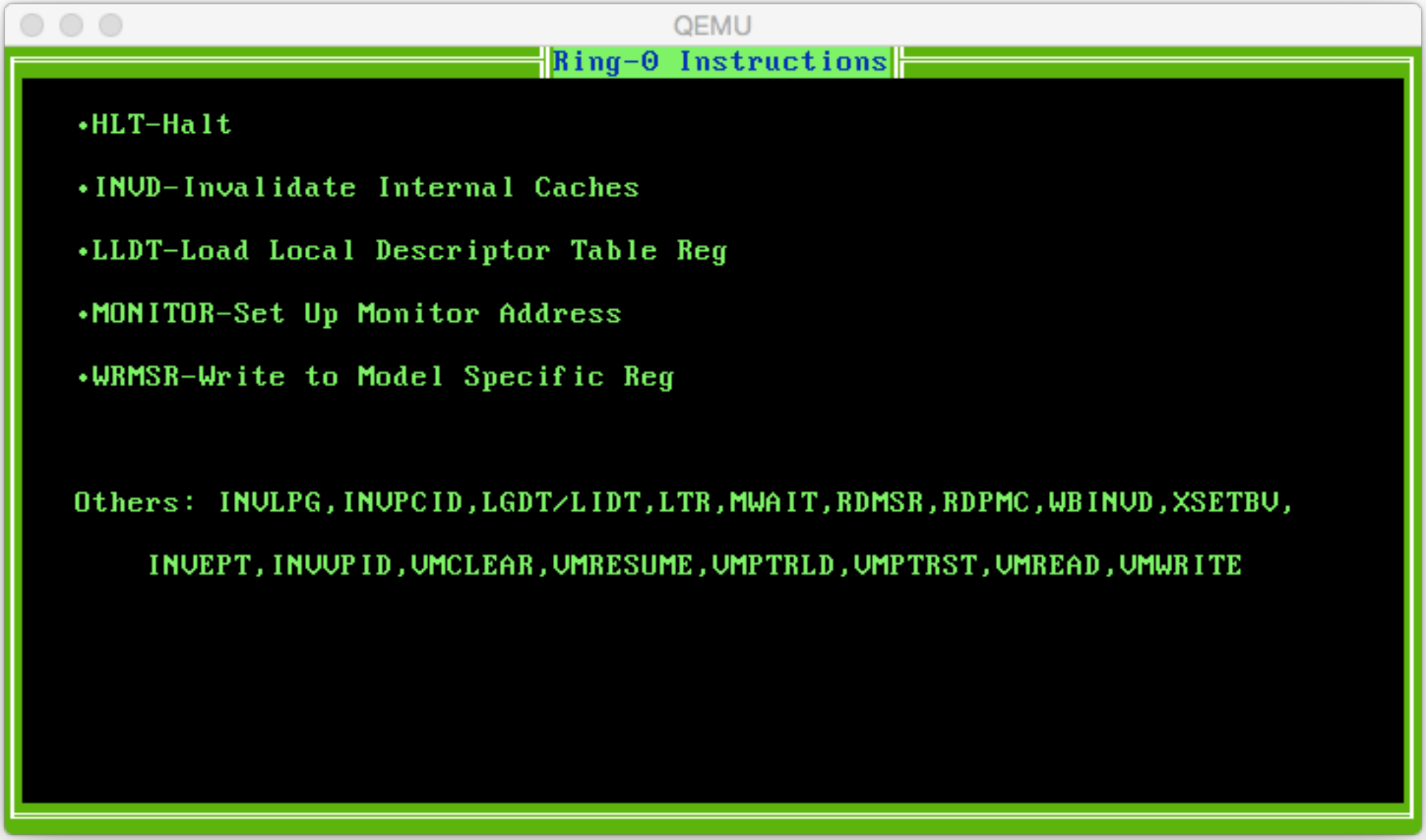
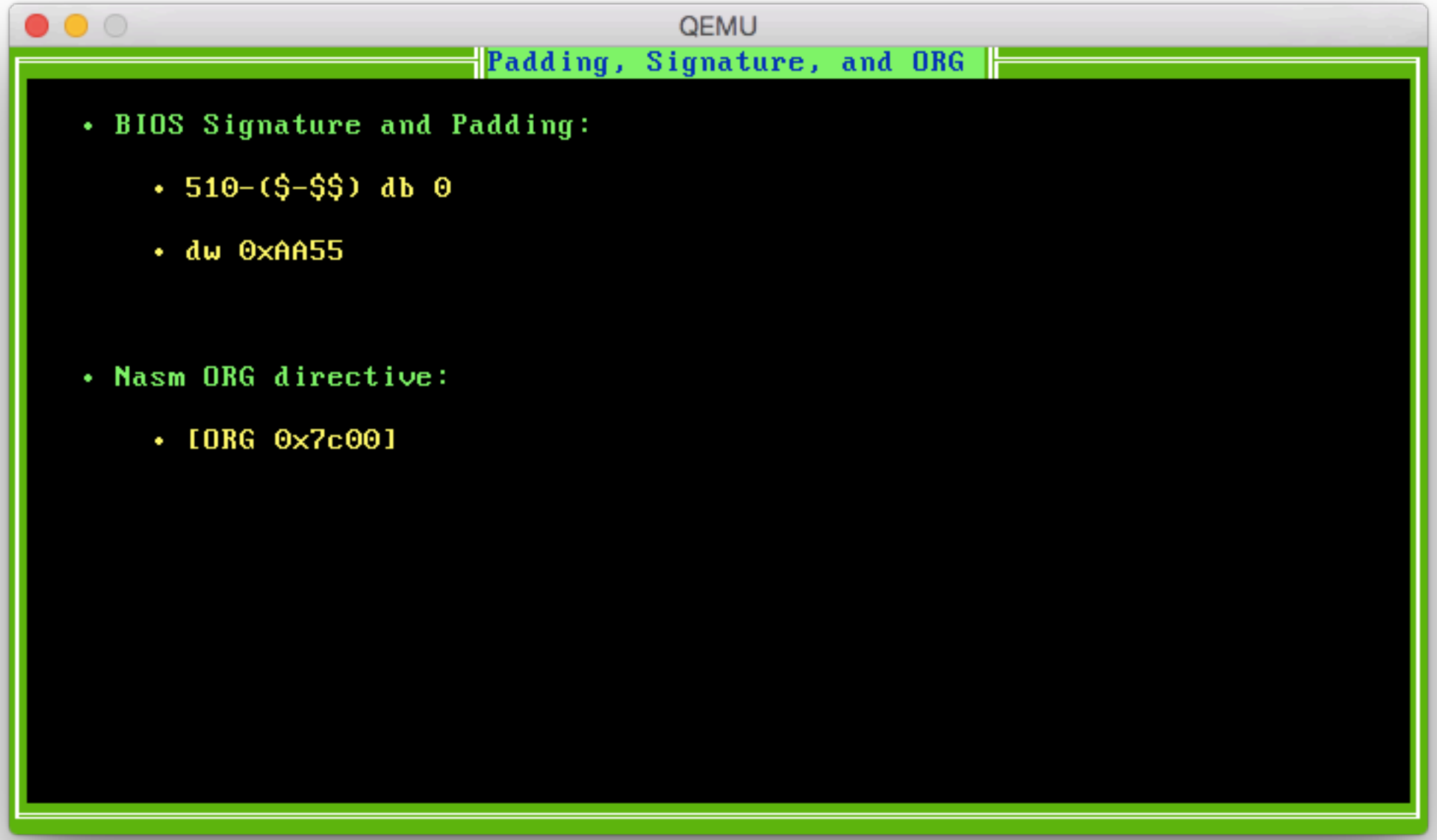

QEMU
About Us
• XlogicX
@XlogicX
github.com/XlogicX
• Bryan Geesey
github.com/darkvoxels
• BIOS Slides
github.com/XlogicX/CactusCon2017/
CRC16: 0x7A69 (31337)

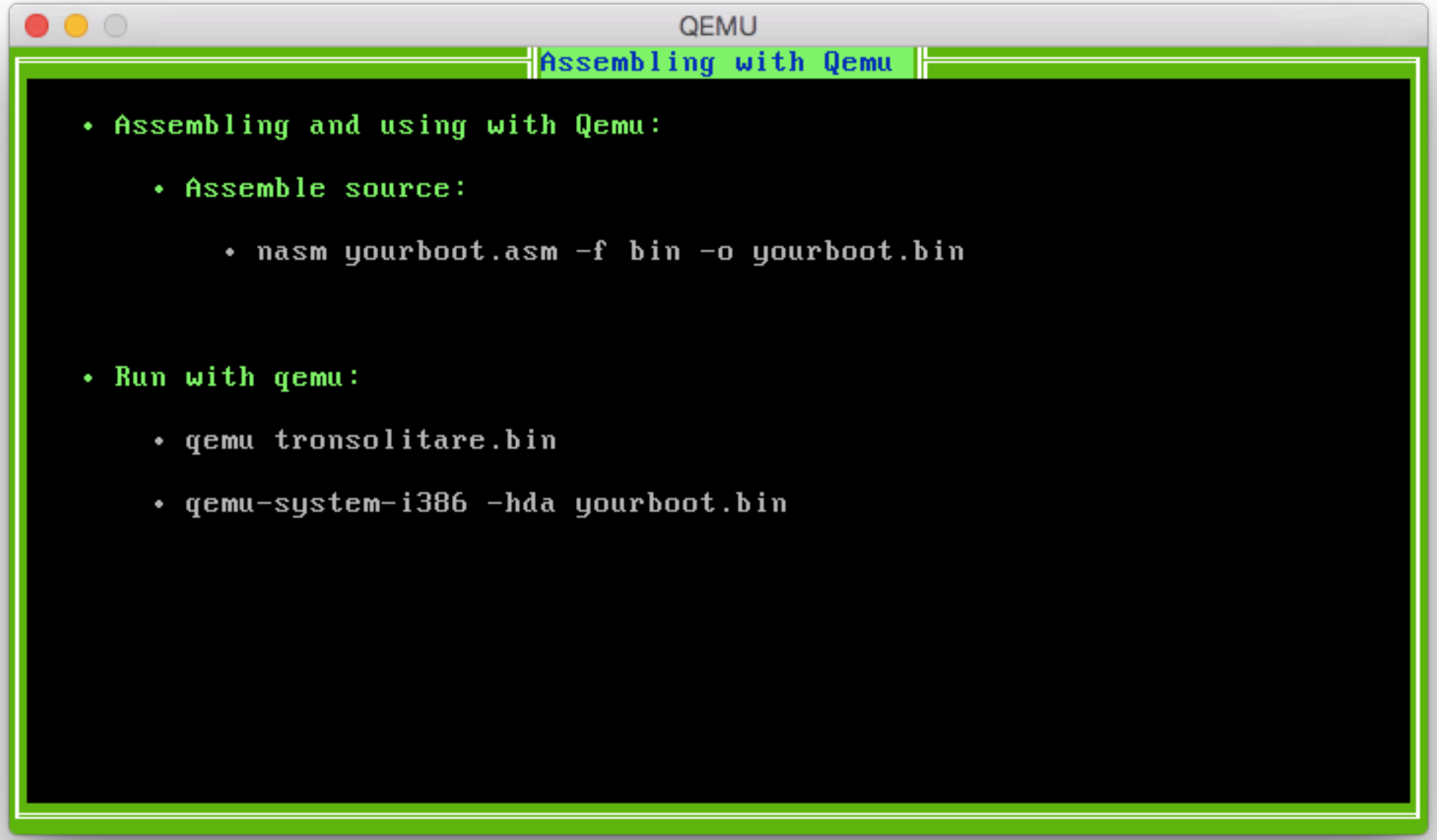

QEMU
CACTUS
NEON


QEMU
Introduction
• Educational (Programming) topic
• BIOS signature and padding
• Assembling
• Text/Graphics Mode, Video, & Stack setup
• Important Registers
• Time Delay Loop
• Keyboard Input Methods
• Randomness & Color
• Debugging:
• Showcases/Demos:

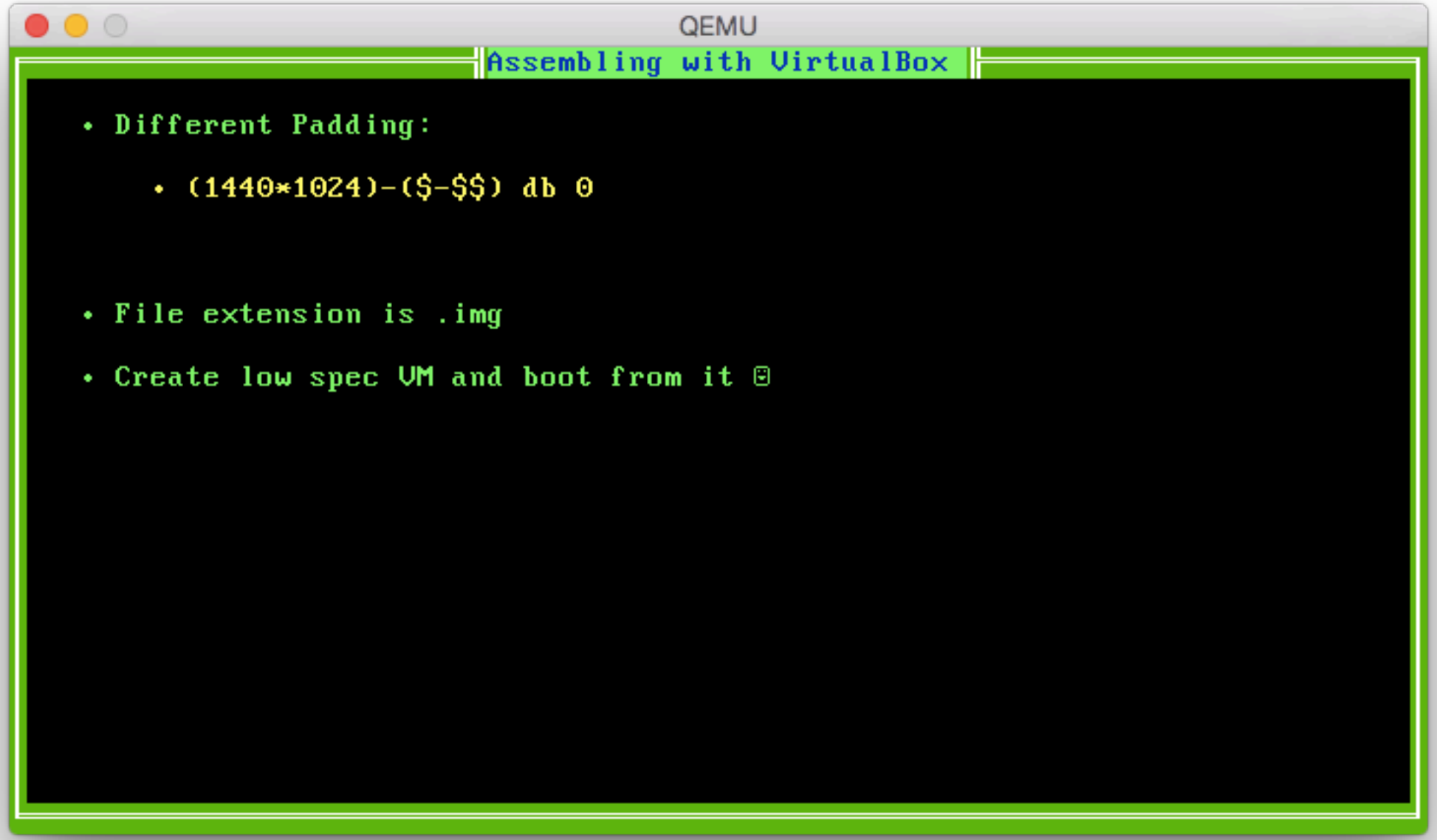

QEMU
Why
• Because its fun
• Limitations encourage creativity
• Ring-0 programming

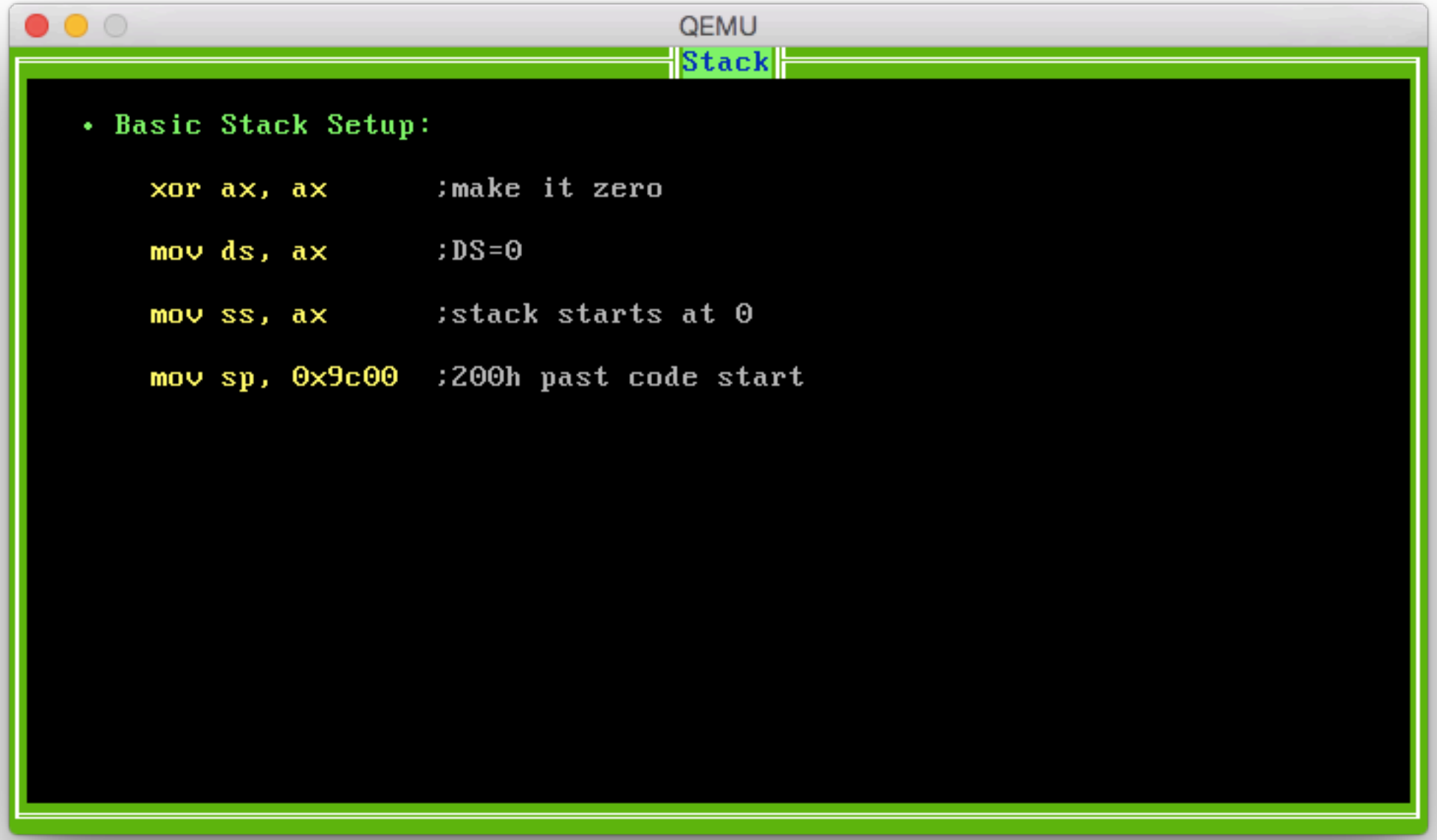

QEMU
Ring-0 Instructions
•HLT-Halt
•INVD-Invalidate Internal Caches
•LLDT-Load Local Descriptor Table Reg
•MONITOR-Set Up Monitor Address
•WRMSR-Write to Model Specific Reg
Others: INVLPG,INVPCID,LGDT/LIDT,LTR,MWAIT,RDMSR,RDPMC,WBINVD,XSETBV,
INVEPT,INVVPID,VMCLEAR,VMRESUME,VMPTRLD,VMPTRST,VMREAD,VMWRITE

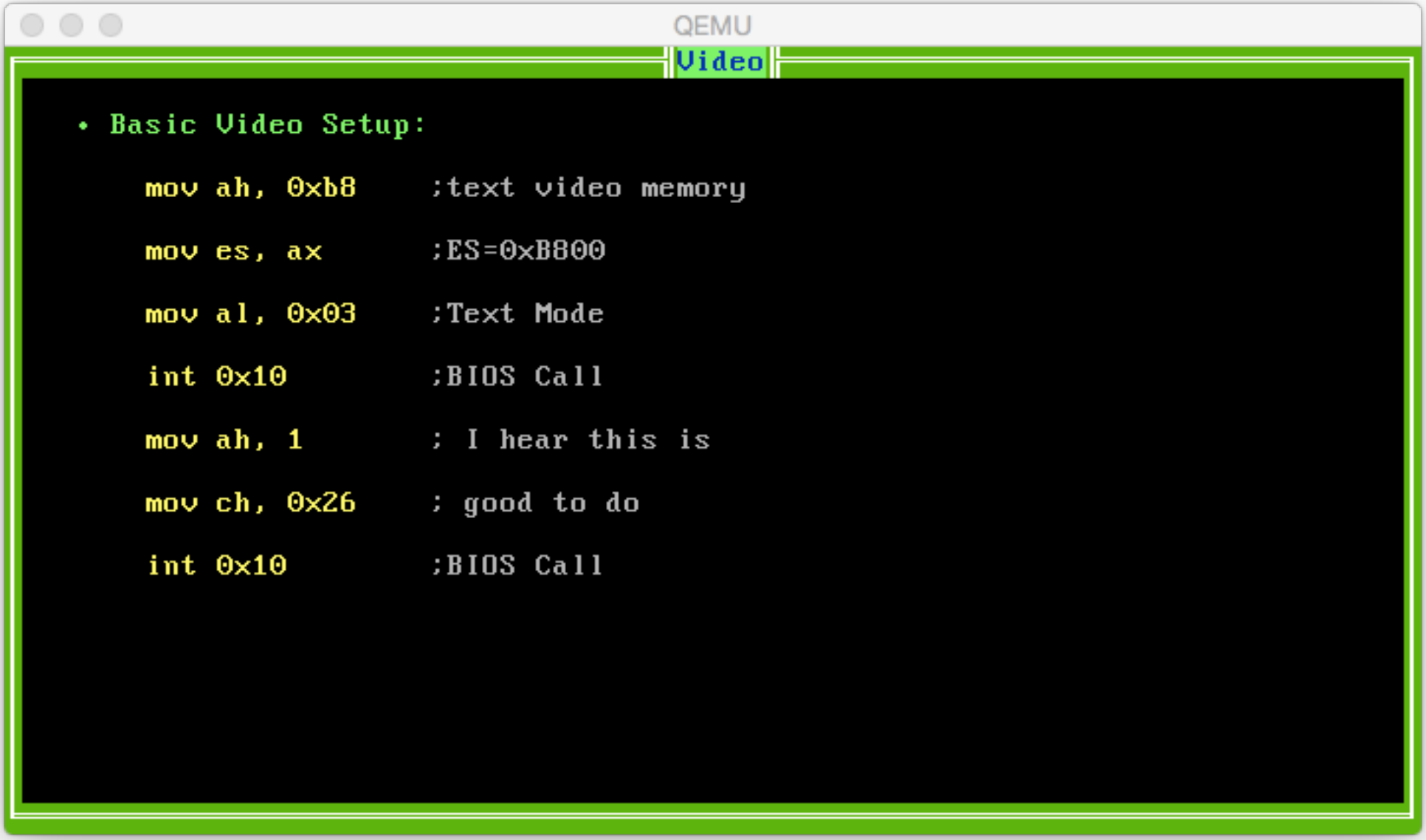

QEMU
Padding, Signature, and ORG
• BIOS Signature and Padding:
• 510-($-$$) db 0
• dw 0xAA55
• Nasm ORG directive:
• [ORG 0x7c00]

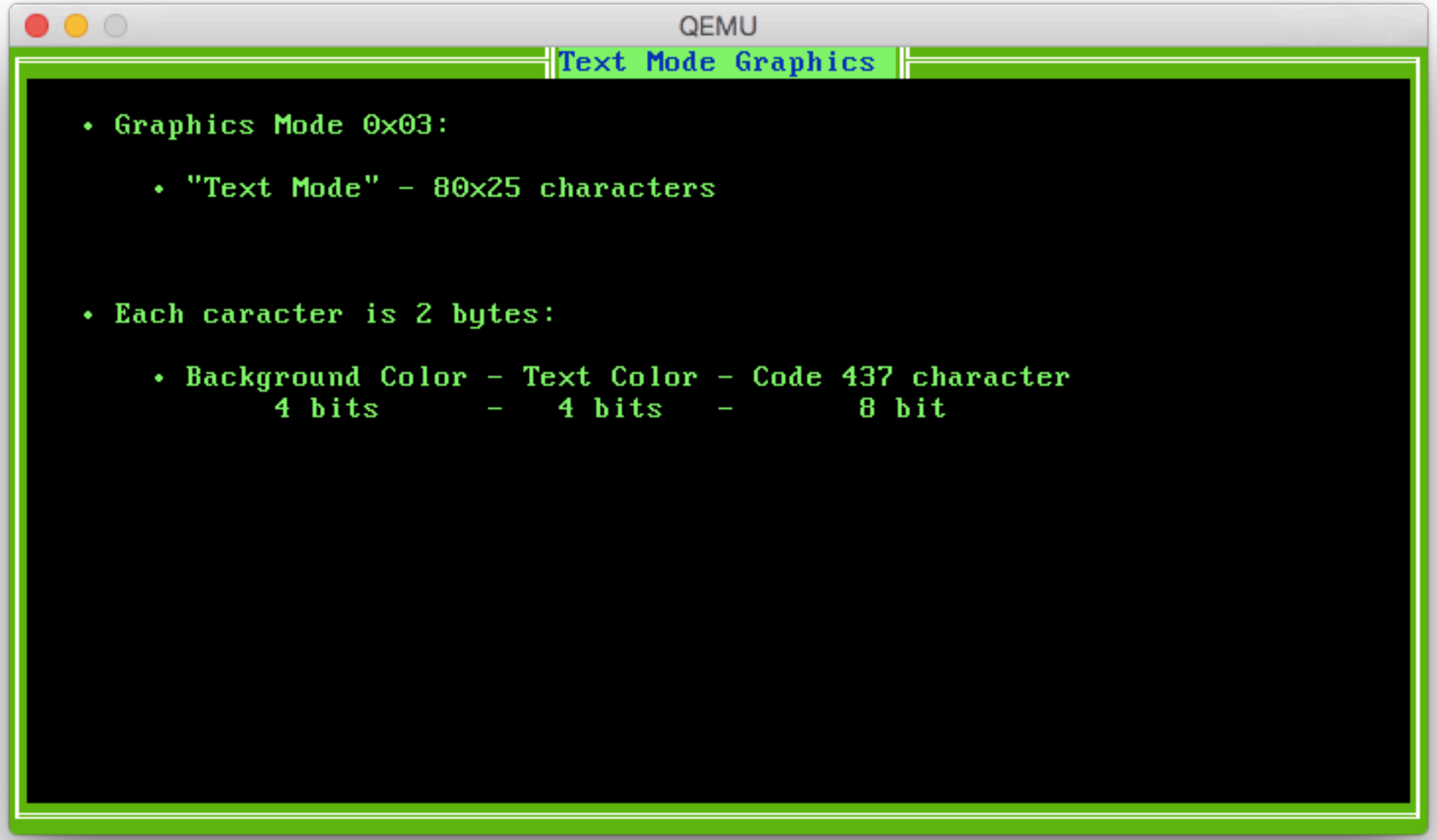

QEMU
Assembling with Qemu
• Assembling and using with Qemu:
• Assemble source:
• nasm yourboot.asm -f bin -o yourboot.bin
• Run with qemu:
• qemu tronsolitare.bin
• qemu-system-i386 -hda yourboot.bin

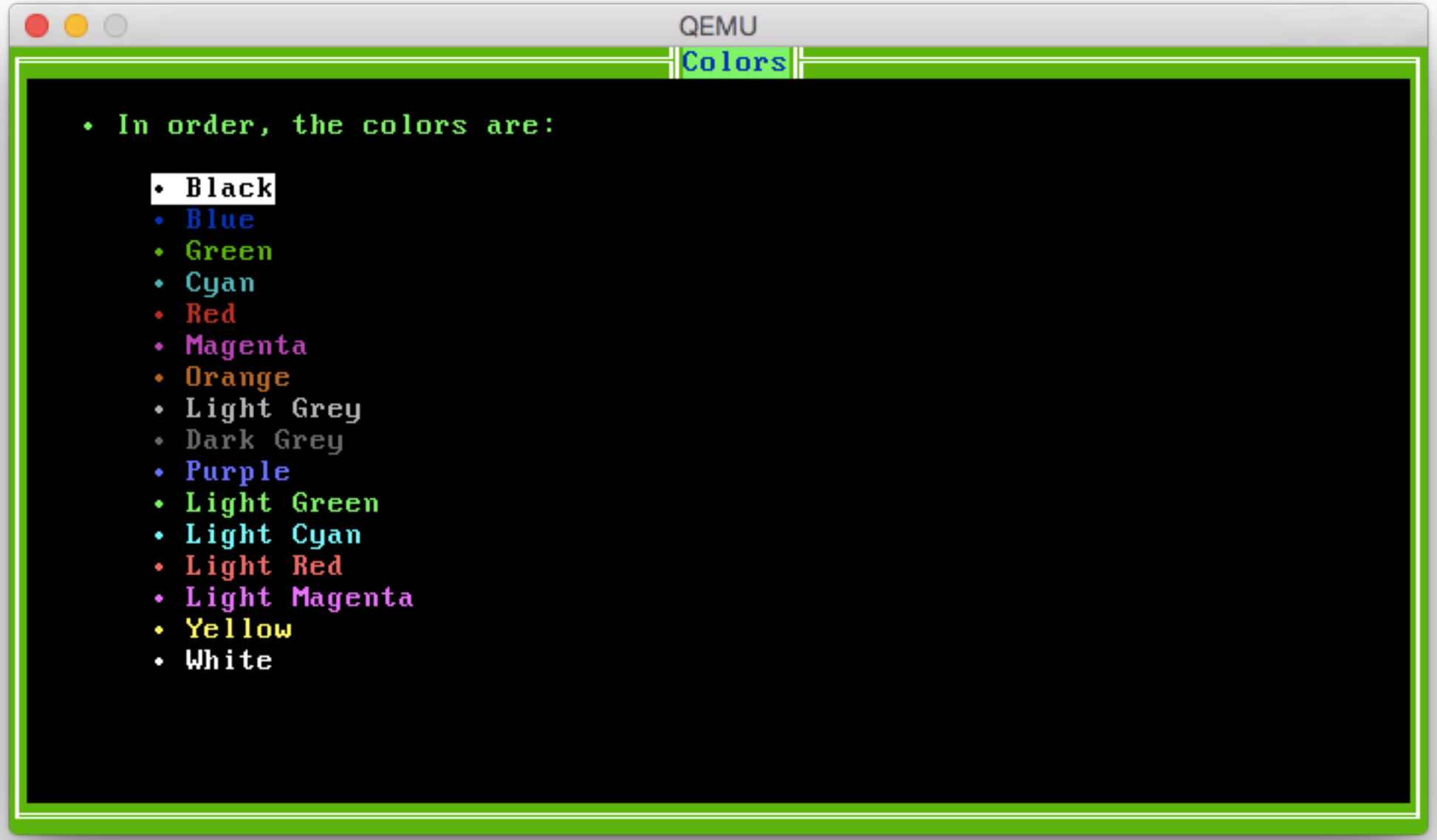
QEMU

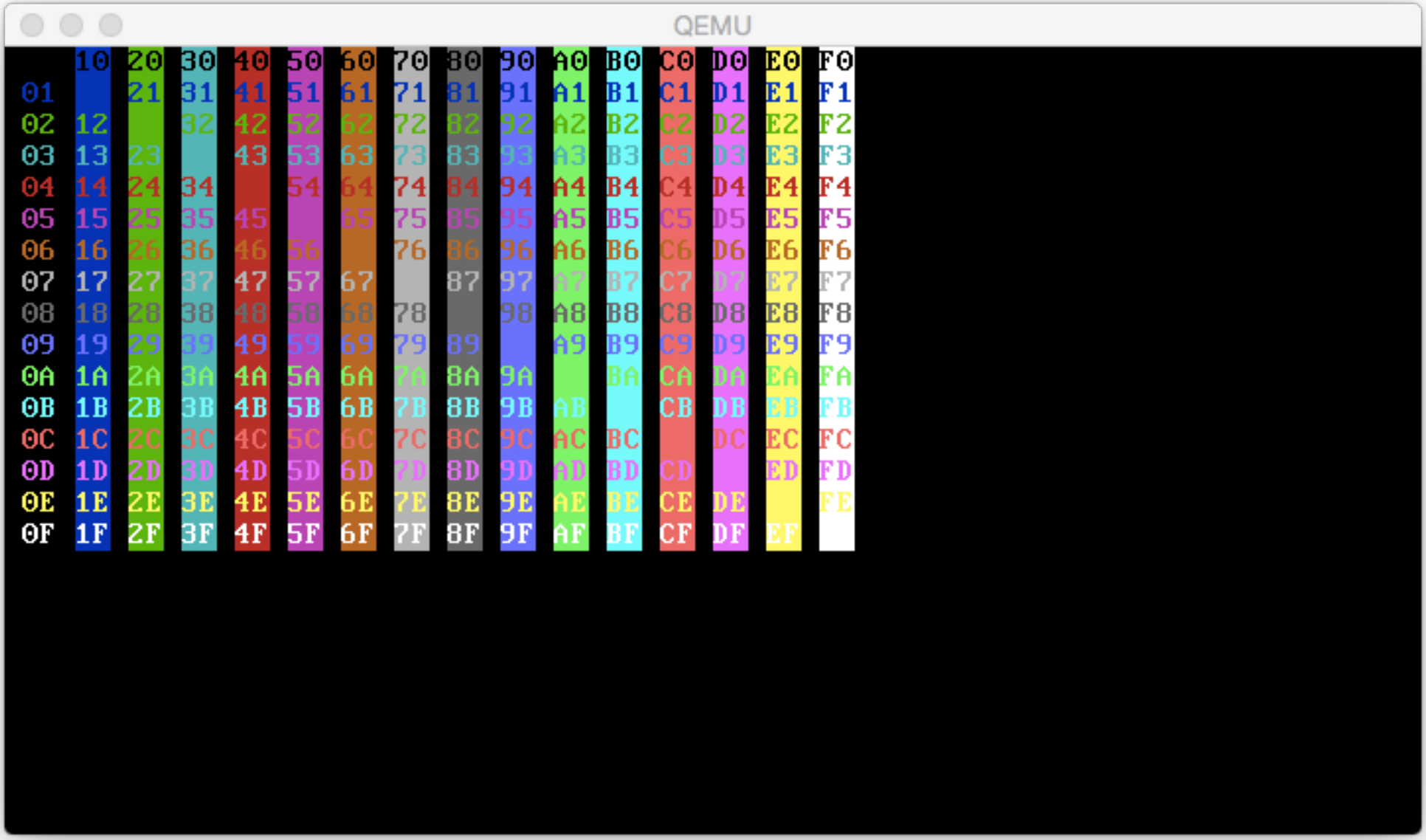

QEMU
Assembling with VirtualBox
• Different Padding:
• (1440*1024)-($-$$) db 0
• File extension is .img
• Create low spec VM and boot from it


QEMU
Stack
• Basic Stack Setup:
xor ax, ax ;make it zero
mov ds, ax ;DS=0
mov ss, ax ;stack starts at 0
mov sp, 0x9c00 ;200h past code start

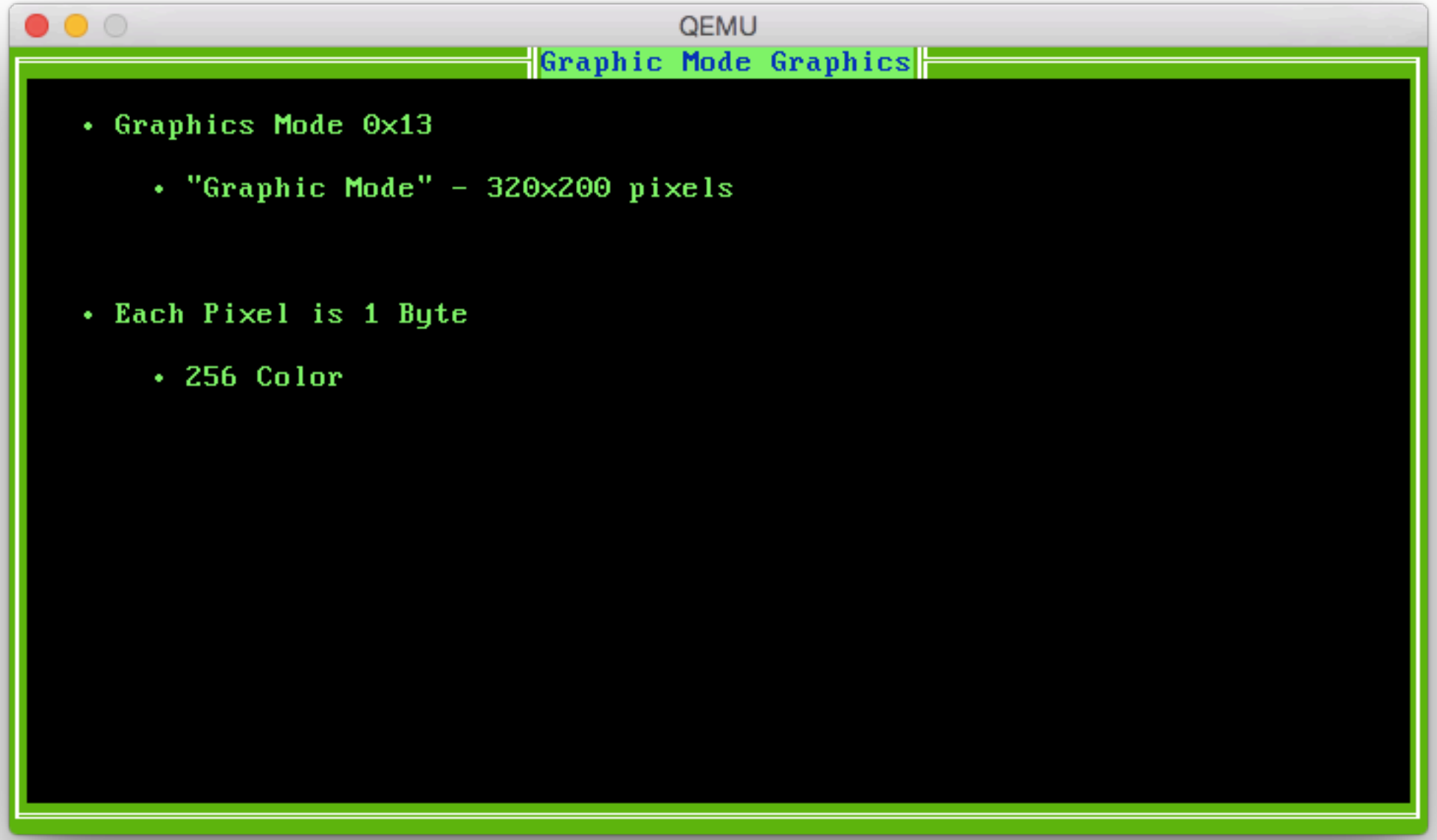

QEMU
Video
• Basic Video Setup:
mov ah, 0xb8 ;text video memory
mov es, ax ;ES=0xB800
mov al, 0x03 ;Text Mode
int 0x10 ;BIOS Call
mov ah, 1 ; I hear this is
mov ch, 0x26 ; good to do
int 0x10 ;BIOS Call

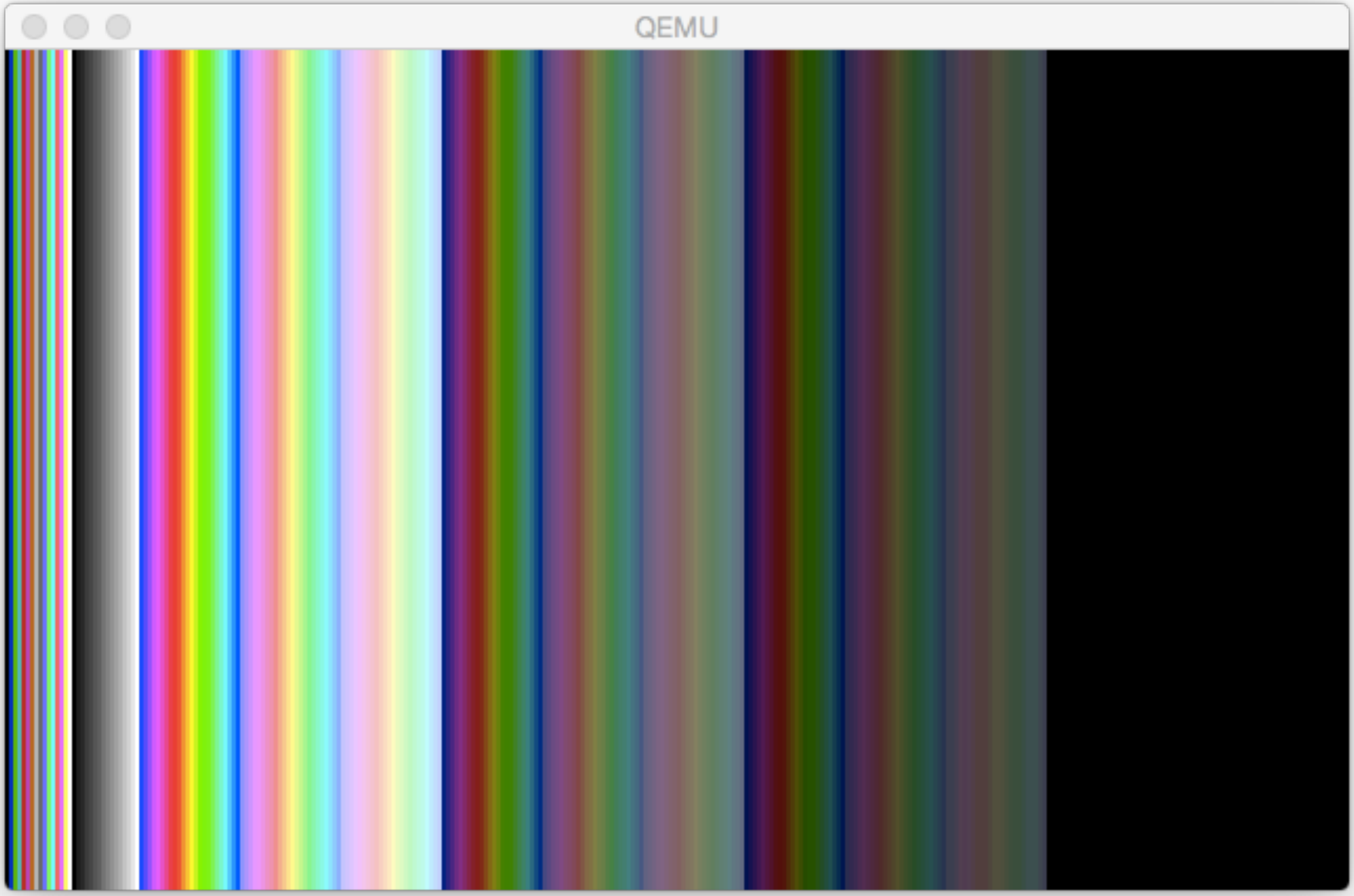

QEMU
Text Mode Graphics
• Graphics Mode 0x03:
• "Text Mode" - 80x25 characters
• Each caracter is 2 bytes:
• Background Color - Text Color - Code 437 character
4 bits - 4 bits - 8 bit

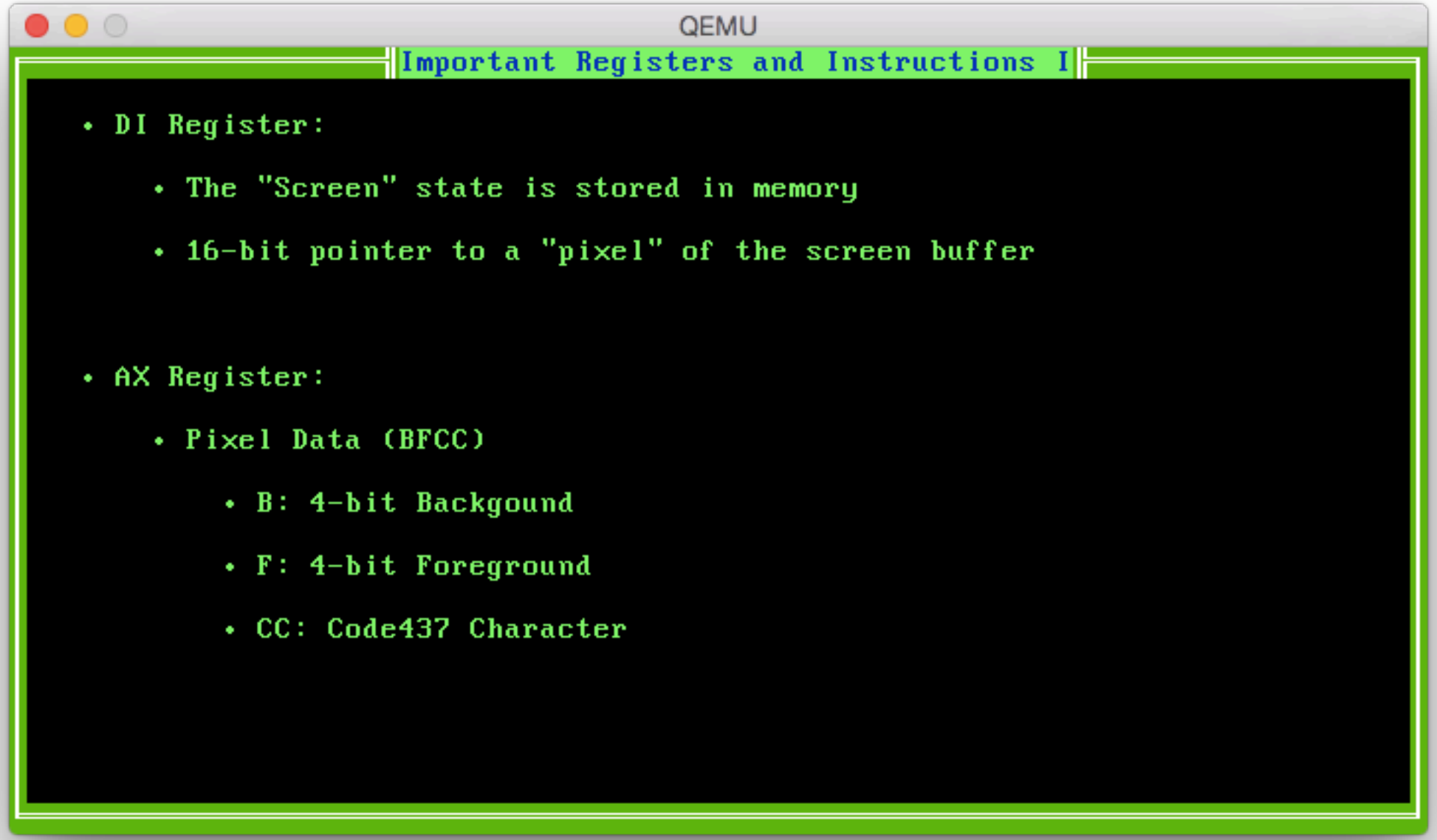

QEMU
Colors
• In order, the colors are:
• Black
• Blue
• Green
• Cyan
• Red
• Magenta
• Orange
• Light Grey
• Dark Grey
• Purple
• Light Green
• Light Cyan
• Light Red
• Light Magenta
• Yellow
• White


QEMU
10 20 30 40 50 60 70 80 90 A0 B0 C0 D0 E0 F0
01 21 31 41 51 61 71 81 91 A1 B1 C1 D1 E1 F1
02 12 32 42 52 62 72 82 92 A2 B2 C2 D2 E2 F2
03 13 23 43 53 63 73 83 93 A3 B3 C3 D3 E3 F3
04 14 24 34 54 64 74 84 94 A4 B4 C4 D4 E4 F4
05 15 25 35 45 65 75 85 95 A5 B5 C5 D5 E5 F5
06 16 26 36 46 56 76 86 96 A6 B6 C6 D6 E6 F6
07 17 27 37 47 57 67 87 97 A7 B7 C7 D7 E7 F7
08 18 28 38 48 58 68 78 98 A8 B8 C8 D8 E8 F8
09 19 29 39 49 59 69 79 89 A9 B9 C9 D9 E9 F9
0A 1A 2A 3A 4A 5A 6A 7A 8A 9A BA CA DA EA FA
0B 1B 2B 3B 4B 5B 6B 7B 8B 9B AB CB DB EB FB
0C 1C 2C 3C 4C 5C 6C 7C 8C 9C AC BC DC EC FC
0D 1D 2D 3D 4D 5D 6D 7D 8D 9D AD BD CD ED FD
0E 1E 2E 3E 4E 5E 6E 7E 8E 9E AE BE CE DE FE
0F 1F 2F 3F 4F 5F 6F 7F 8F 9F AF BF CF DF EF


QEMU
Graphic
• Graphic Mode Setup:
mov ah, 0xA0 ;Pixel Memory
mov es, ax ;ES=0xA000
mov al, 0x13 ; Graphic Mode
int 0x10 ;BIOS Call
mov ah, 1 ; I hear this is
mov ch, 0x26 ; good to do
int 0x10 ;BIOS Call


QEMU
Graphic Mode Graphics
• Graphics Mode 0x13
• "Graphic Mode" - 320x200 pixels
• Each Pixel is 1 Byte
• 256 Color

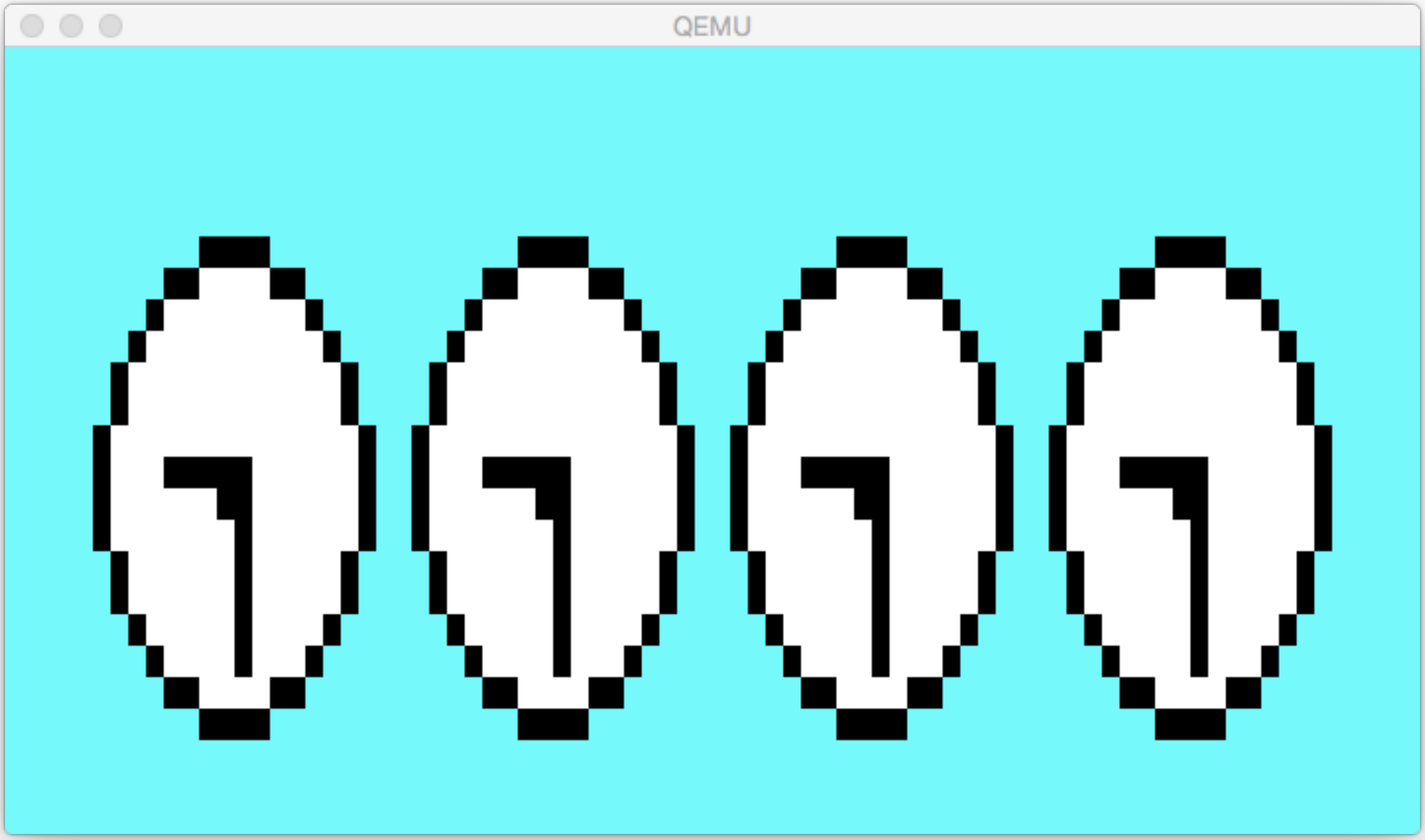

QEMU

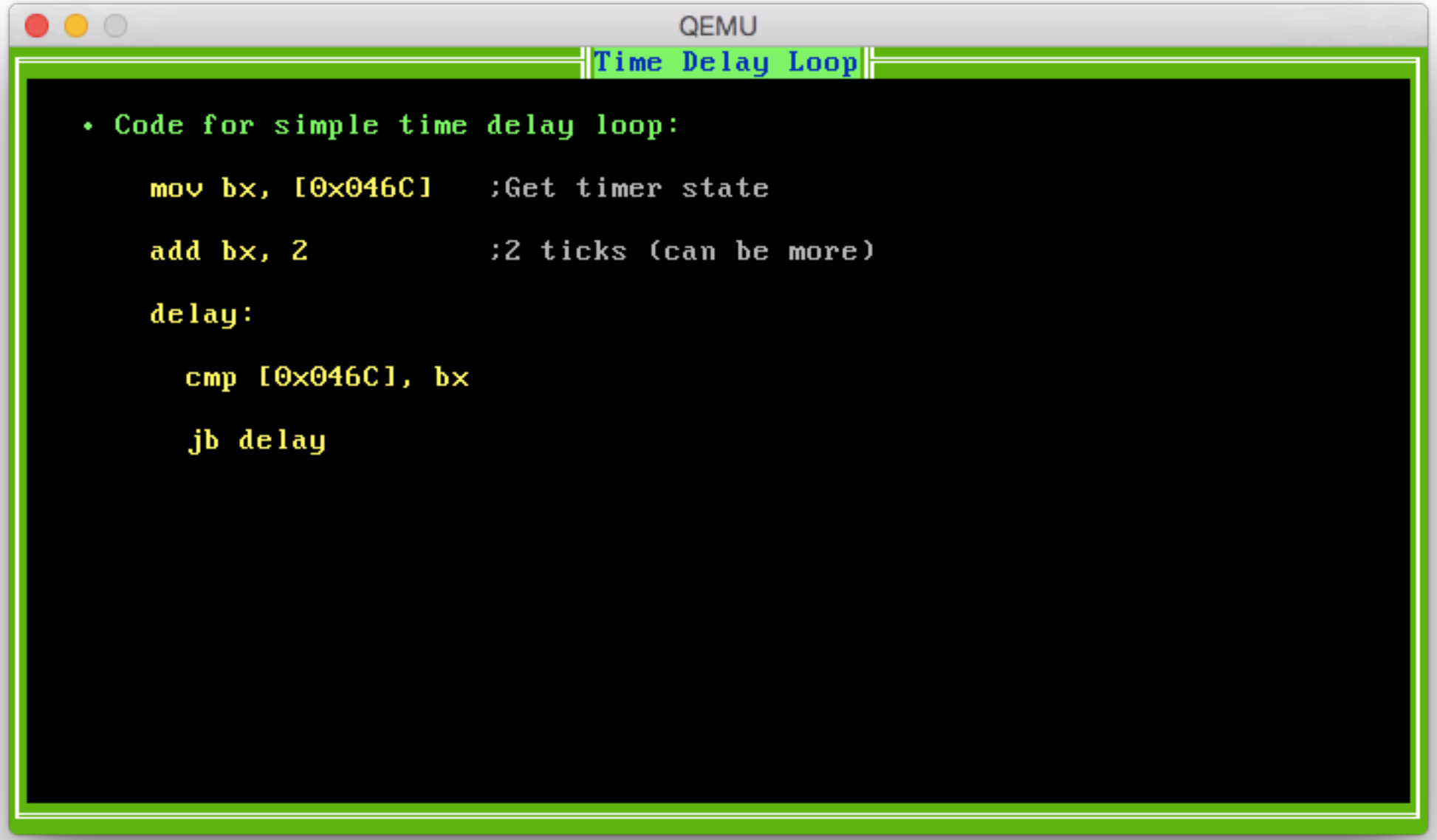

QEMU
Important Registers and Instructions I
• DI Register:
• The "Screen" state is stored in memory
• 16-bit pointer to a "pixel" of the screen buffer
• AX Register:
• Pixel Data (BFCC)
• B: 4-bit Backgound
• F: 4-bit Foreground
• CC: Code437 Character

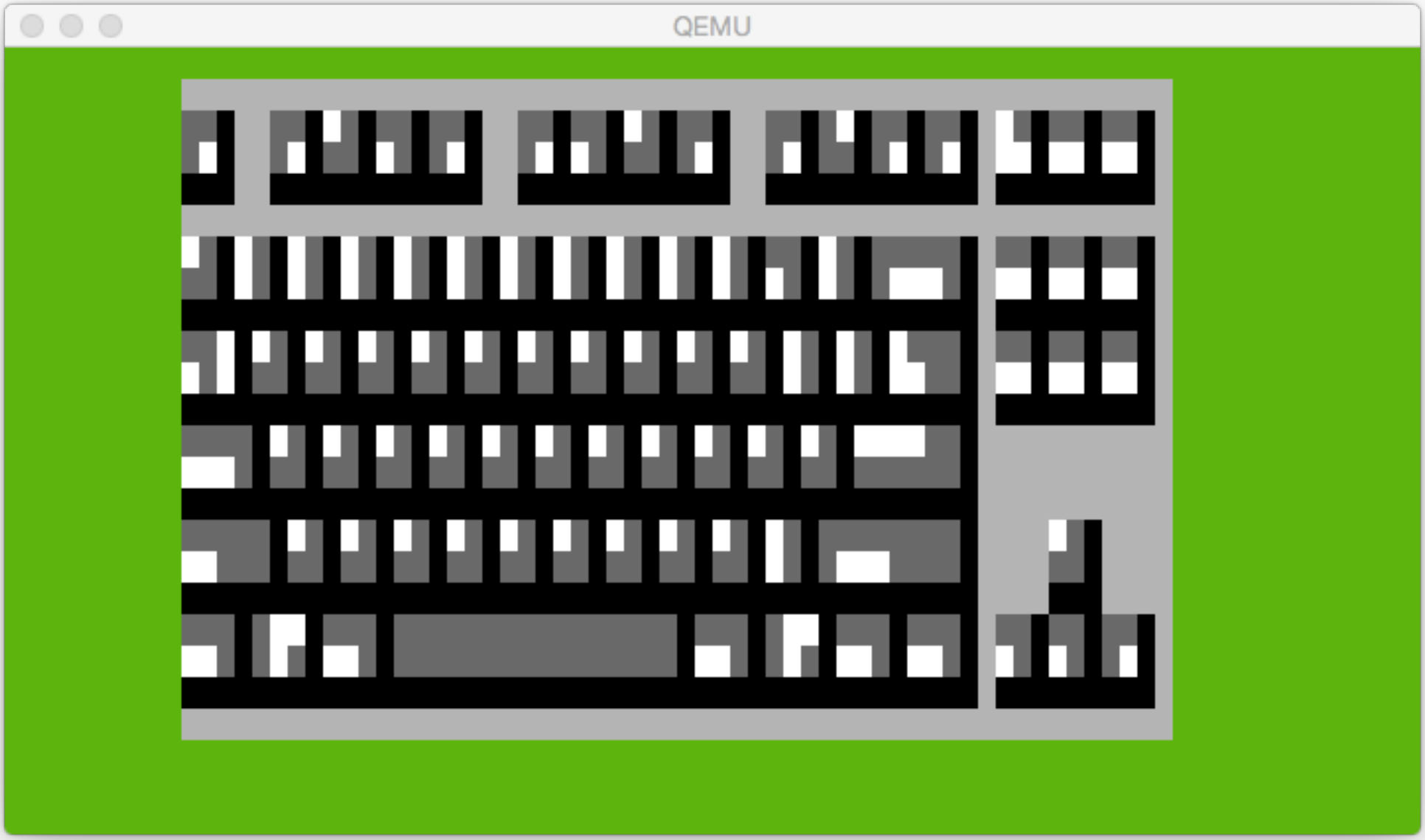
QEMU

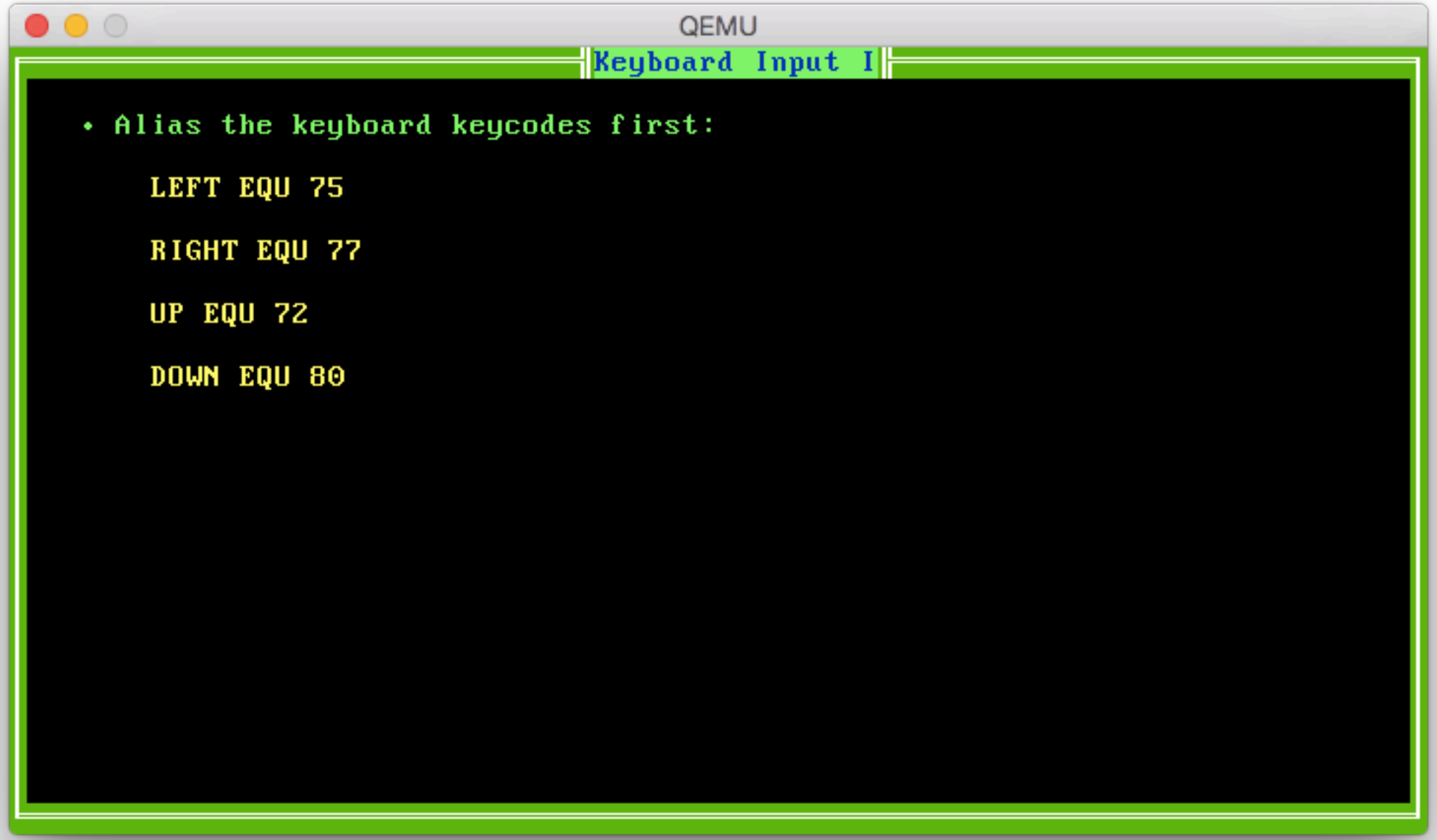
Important Registers and Instructions II

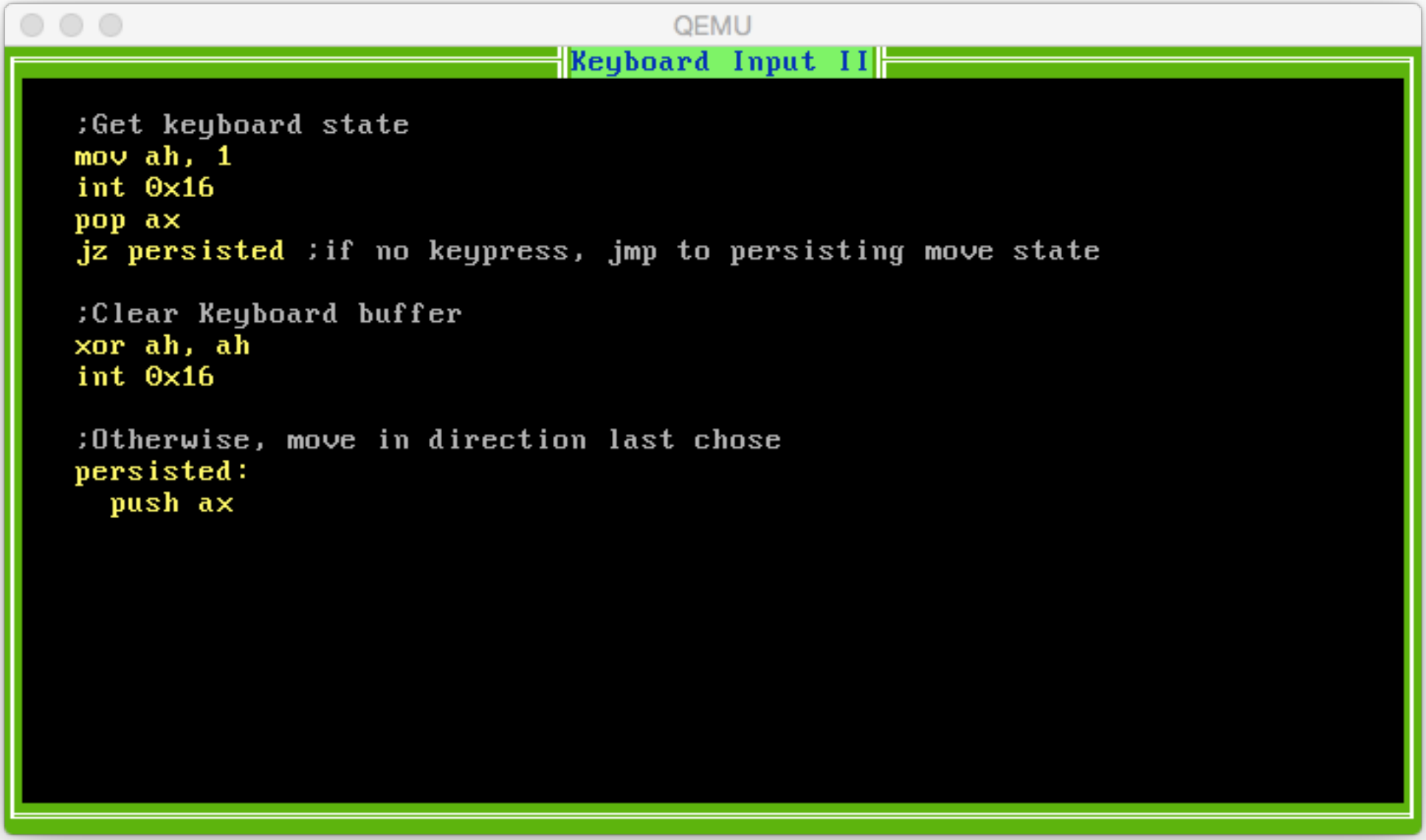
- CX Register:
  - Counter used in conjunction with STOSW
- STOSW Instruction:
  - Puts AX data into DI (Video Mem) and Decrements CX
- SCASW Instruction:
  - Increment DI with the least machine code
- CBW Instruction:
  - Zero out AH in 1 byte

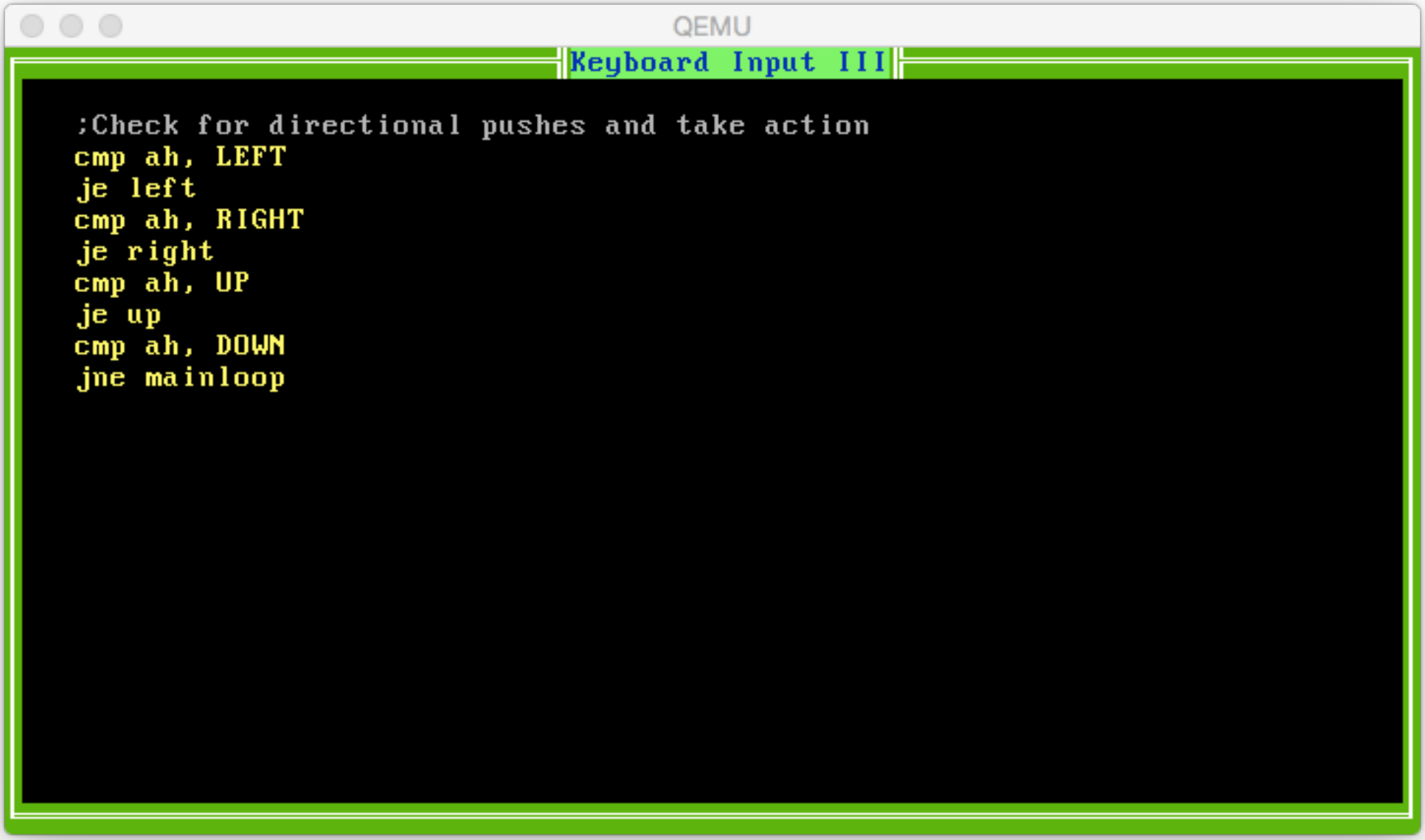

QEMU

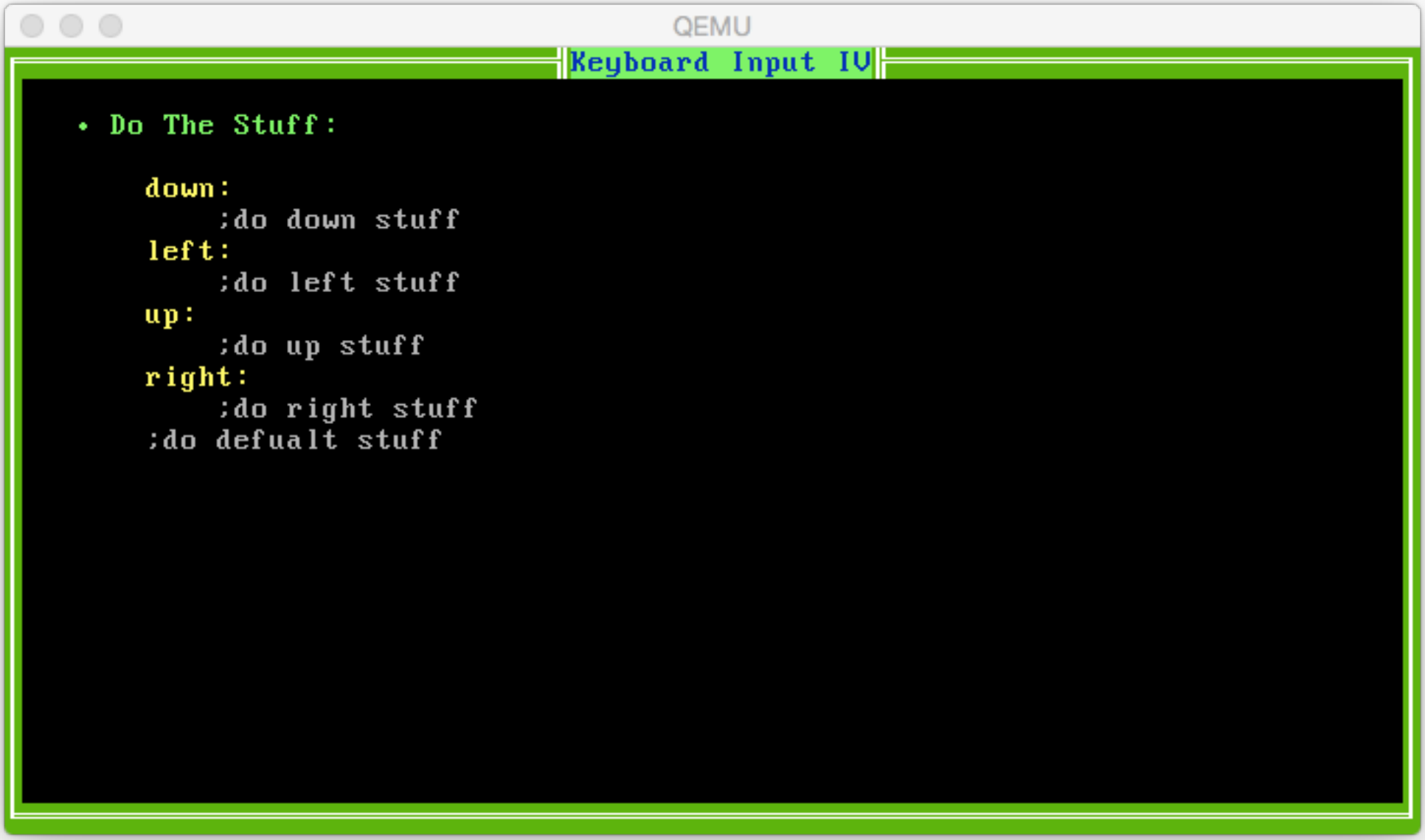

QEMU
Time Delay Loop
• Code for simple time delay loop:
mov bx, [0x046C] ;Get timer state
add bx, 2 ;2 ticks (can be more)
delay:
cmp [0x046C], bx
jb delay

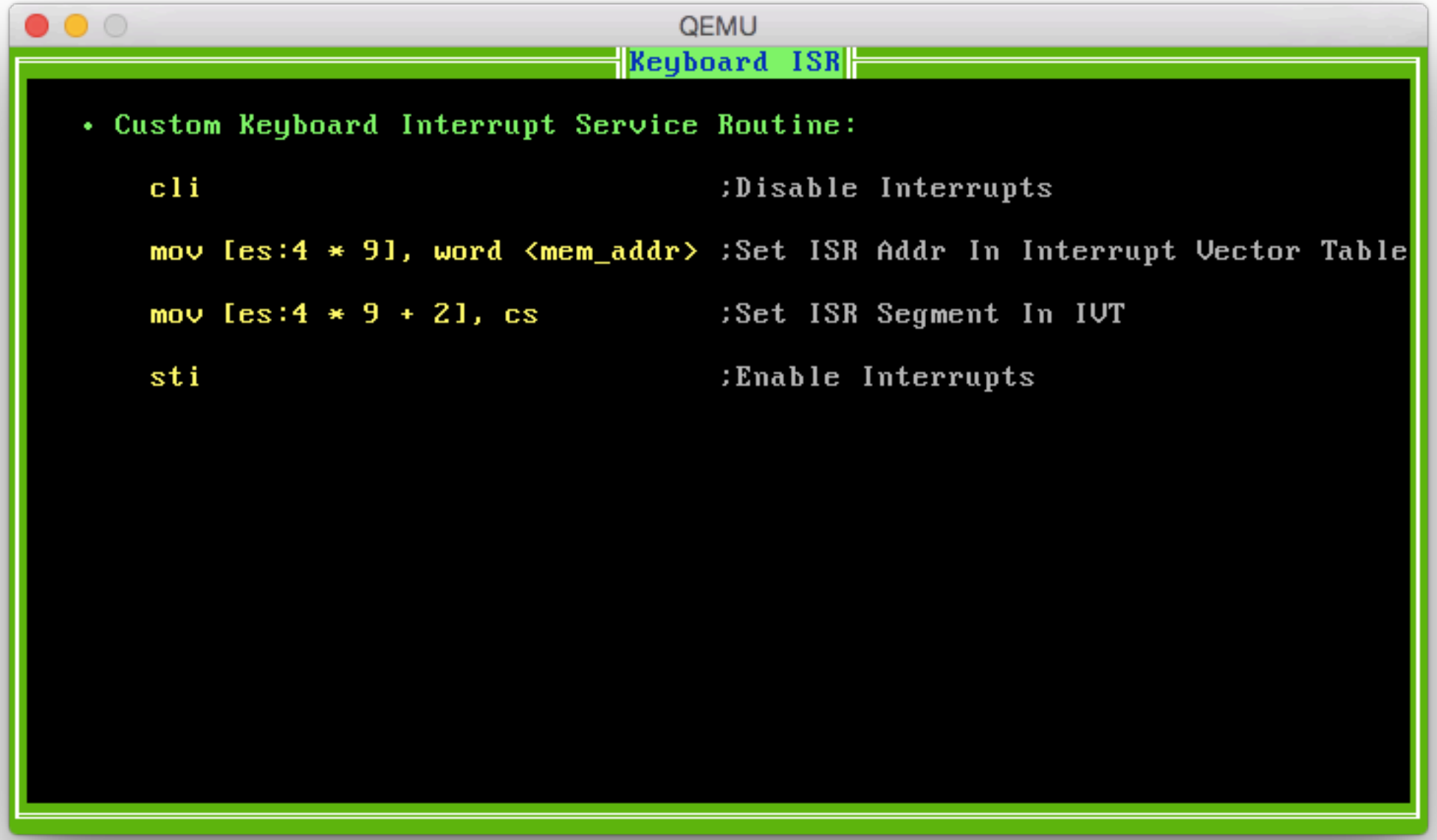

QEMU

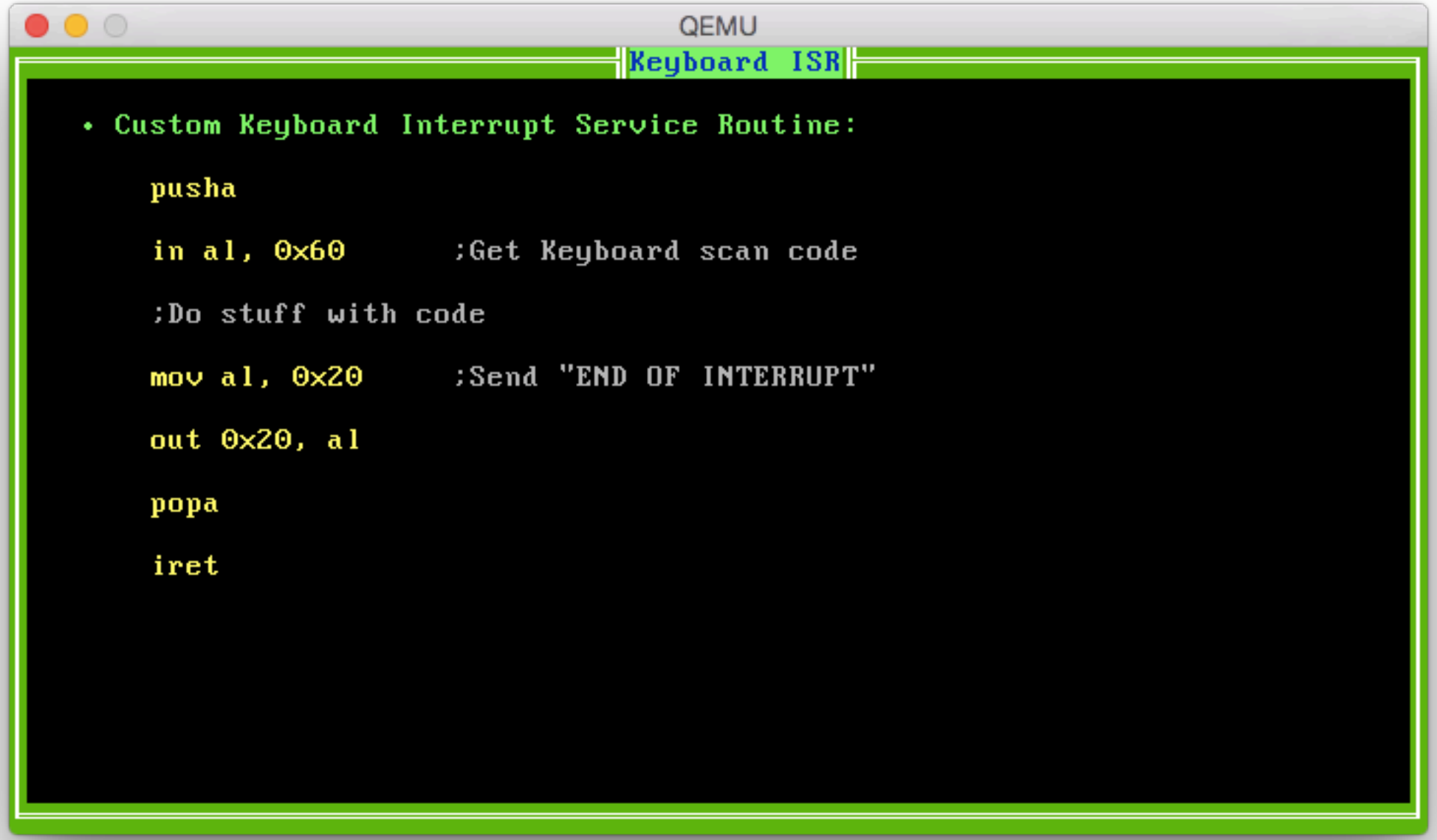

QEMU
Keyboard Input I
• Alias the keyboard keycodes first:
LEFT EQU 75
RIGHT EQU 77
UP EQU 72
DOWN EQU 80

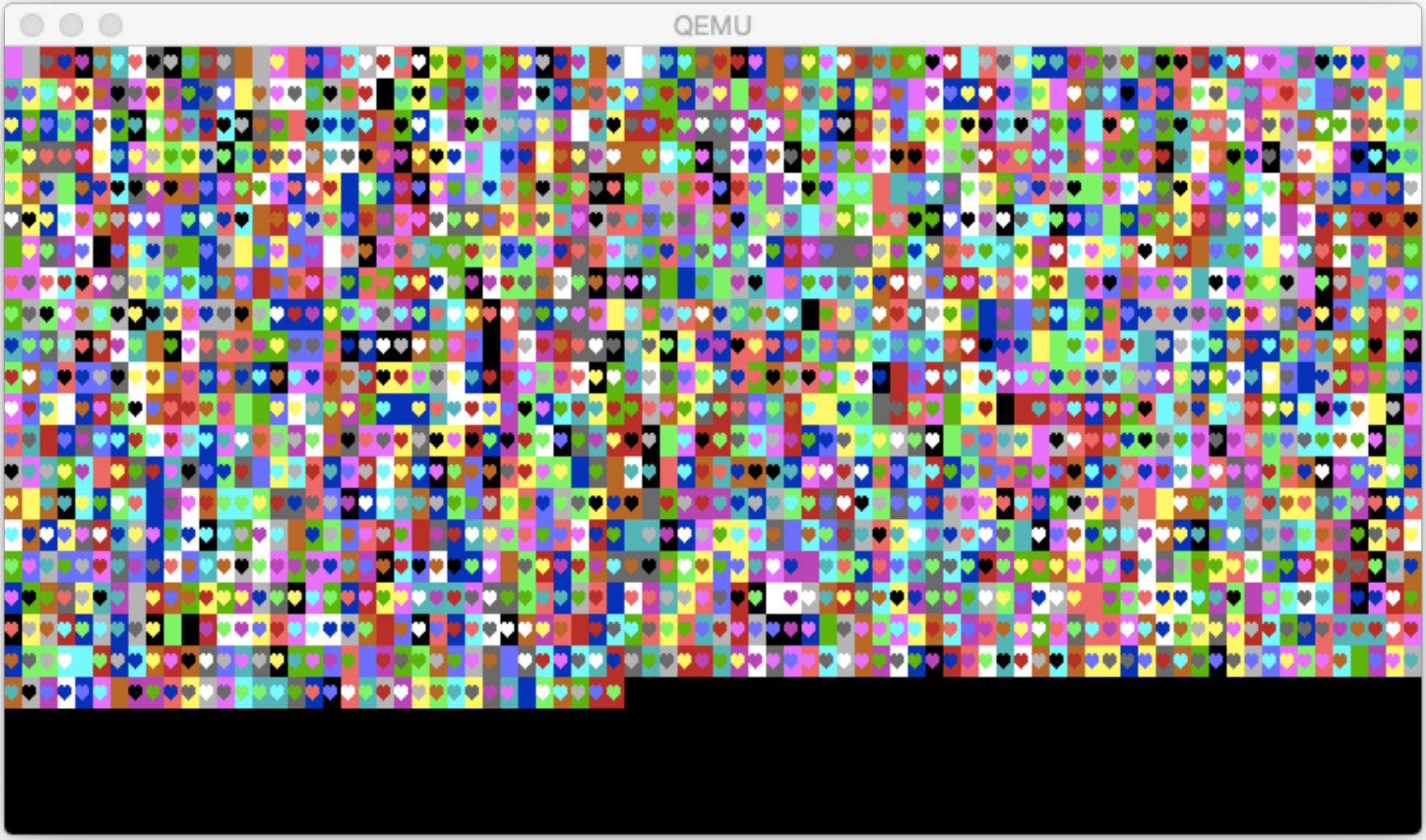

QEMU
Keyboard Input II
;Get keyboard state
mov ah, 1
int 0x16
pop ax
jz persisted ;if no keypress, jmp to persisting move state
;Clear Keyboard buffer
xor ah, ah
int 0x16
;Otherwise, move in direction last chose
persisted:
push ax

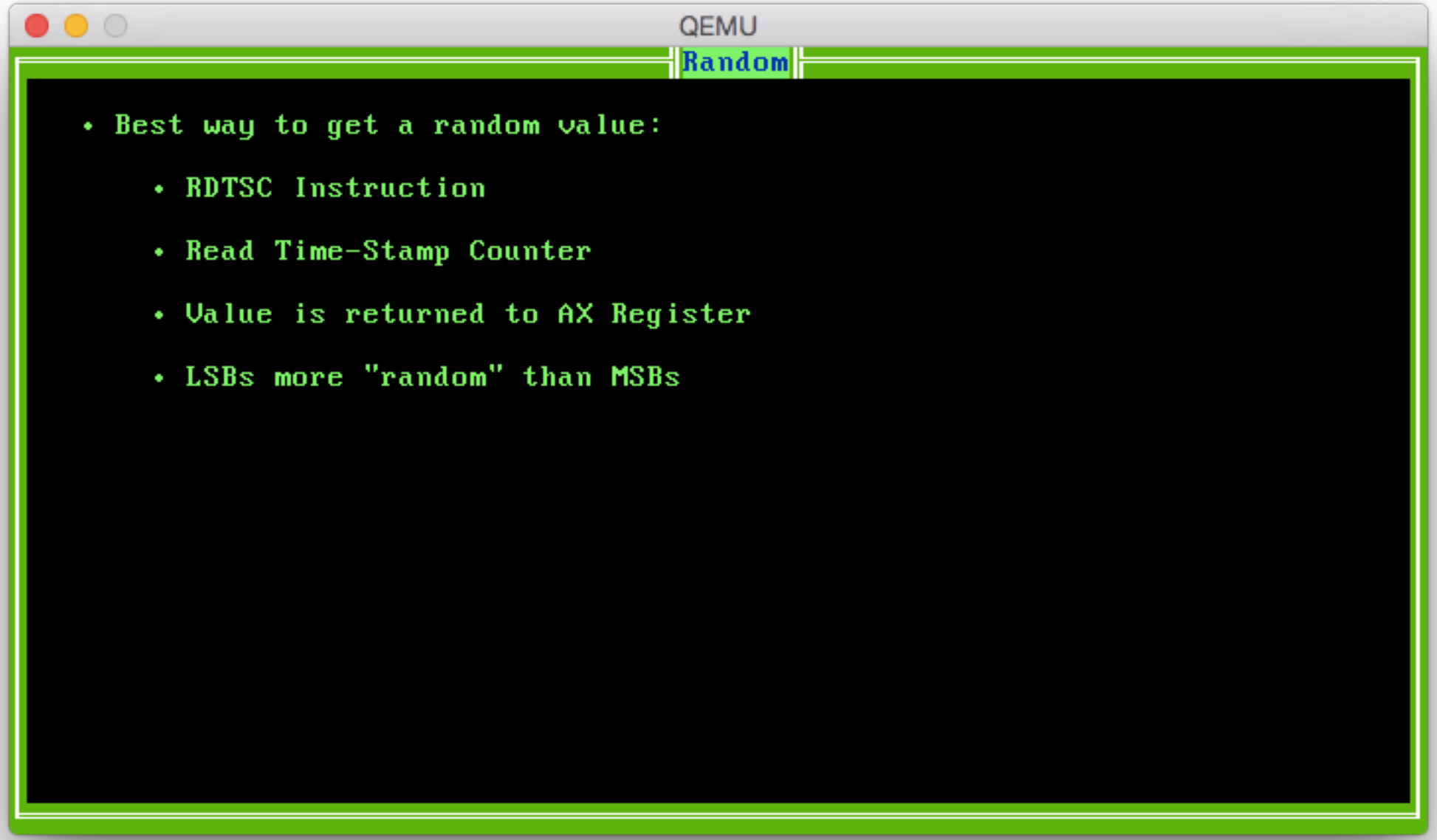

QEMU
Keyboard Input III
;Check for directional pushes and take action
cmp ah, LEFT
je left
cmp ah, RIGHT
je right
cmp ah, UP
je up
cmp ah, DOWN
jne mainloop

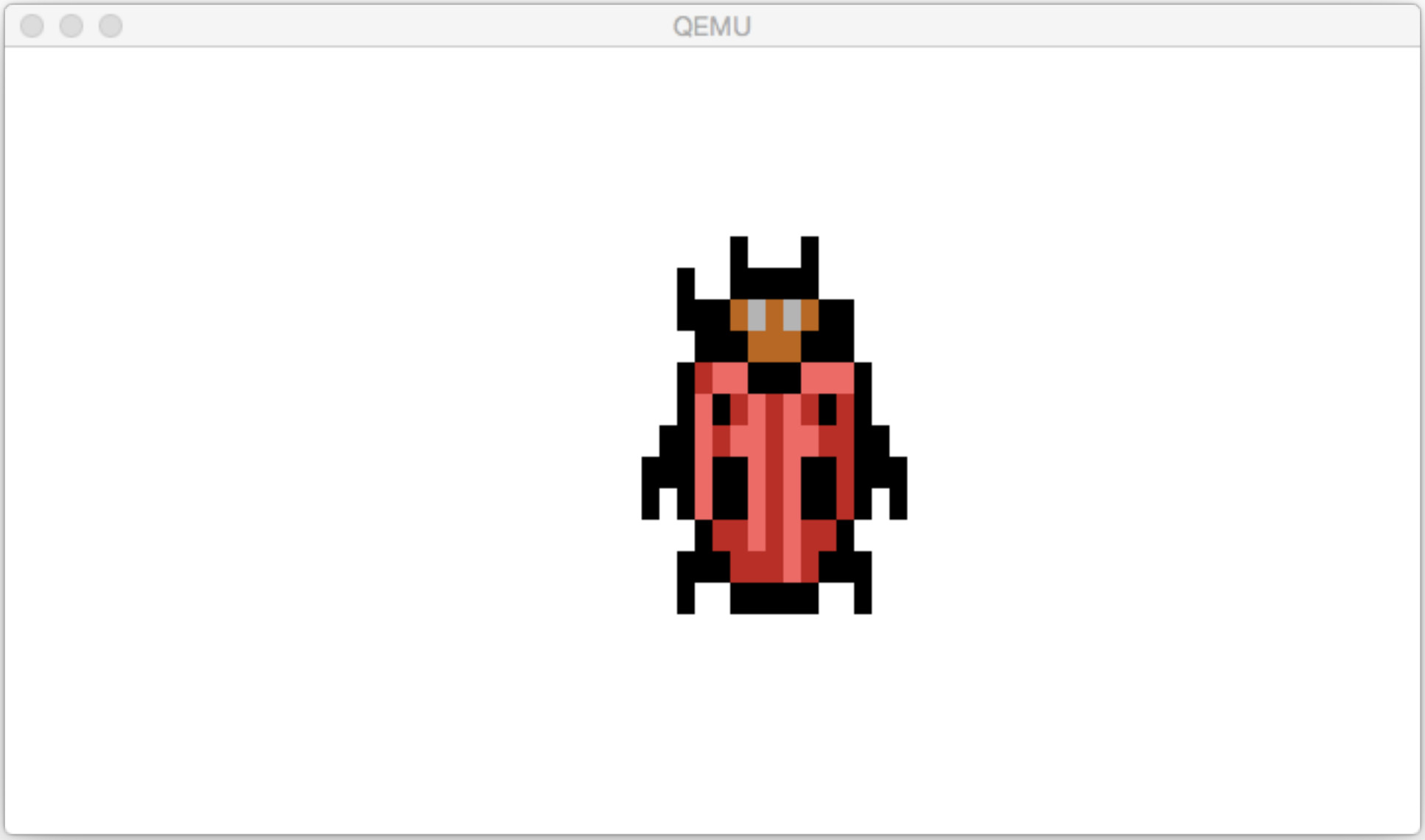

QEMU
Keyboard Input IV
• Do The Stuff:
down:
;do down stuff
left:
;do left stuff
up:
;do up stuff
right:
;do right stuff
;do defualt stuff

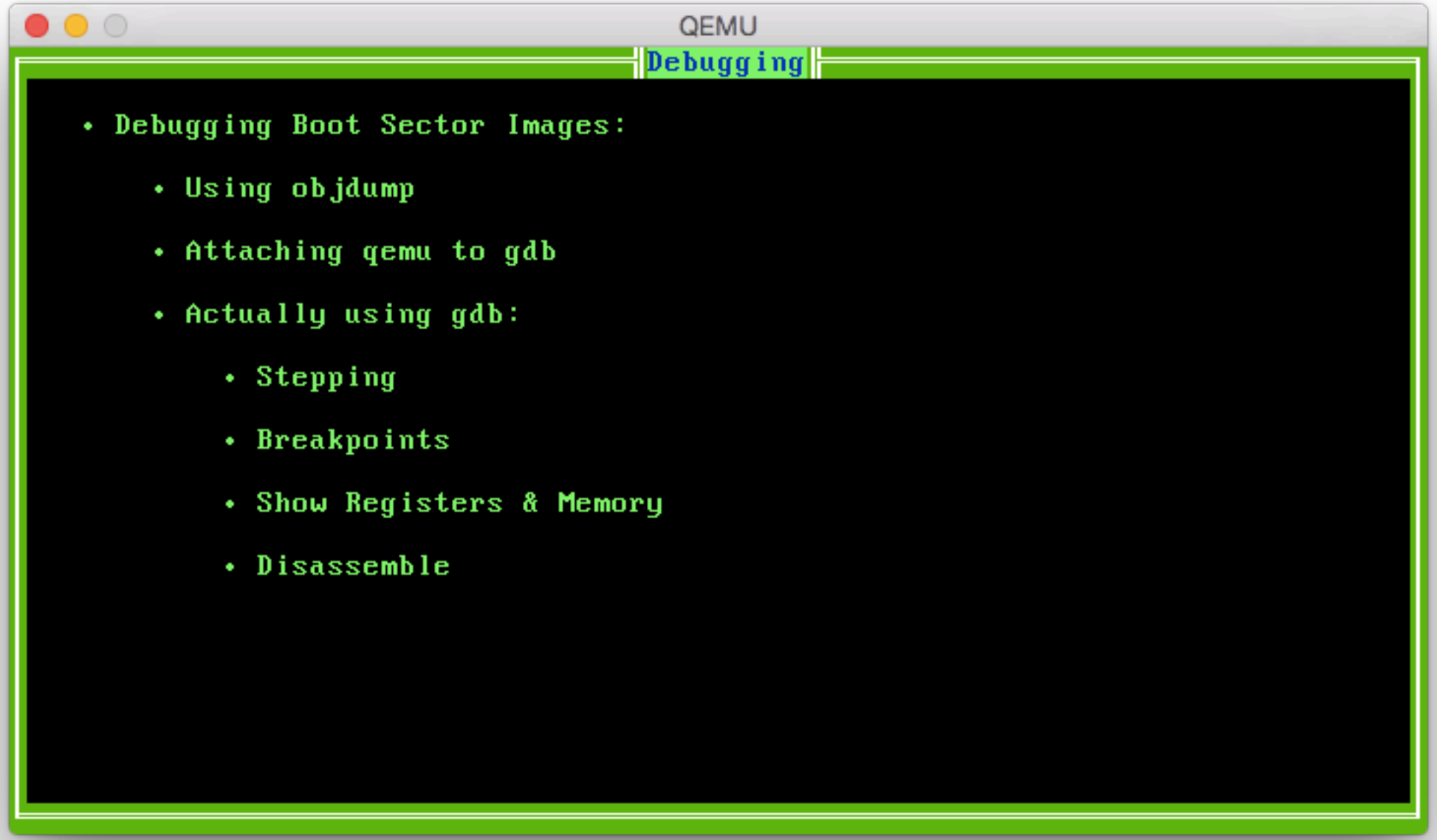

QEMU
Keyboard ISR
• Custom Keyboard Interrupt Service Routine:
cli ;Disable Interrupts
mov [es:4 * 9], word <mem_addr> ;Set ISR Addr In Interrupt Vector Table
mov [es:4 * 9 + 2], cs ;Set ISR Segment In IVT
sti ;Enable Interrupts

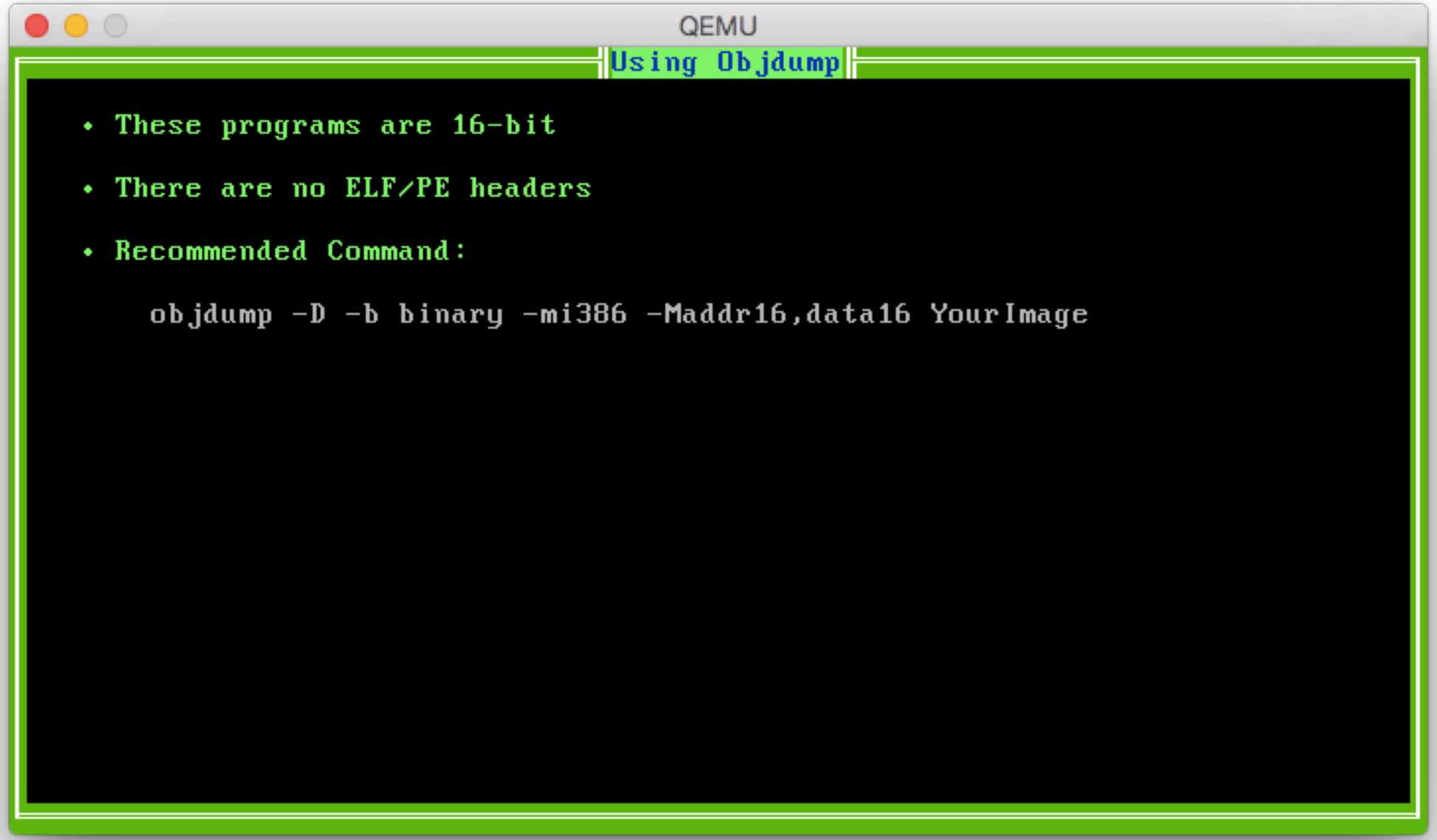

QEMU
Keyboard ISR
• Custom Keyboard Interrupt Service Routine:
pusha
in al, 0x60 ;Get Keyboard scan code
;Do stuff with code
mov al, 0x20 ;Send "END OF INTERRUPT"
out 0x20, al
popa
iret

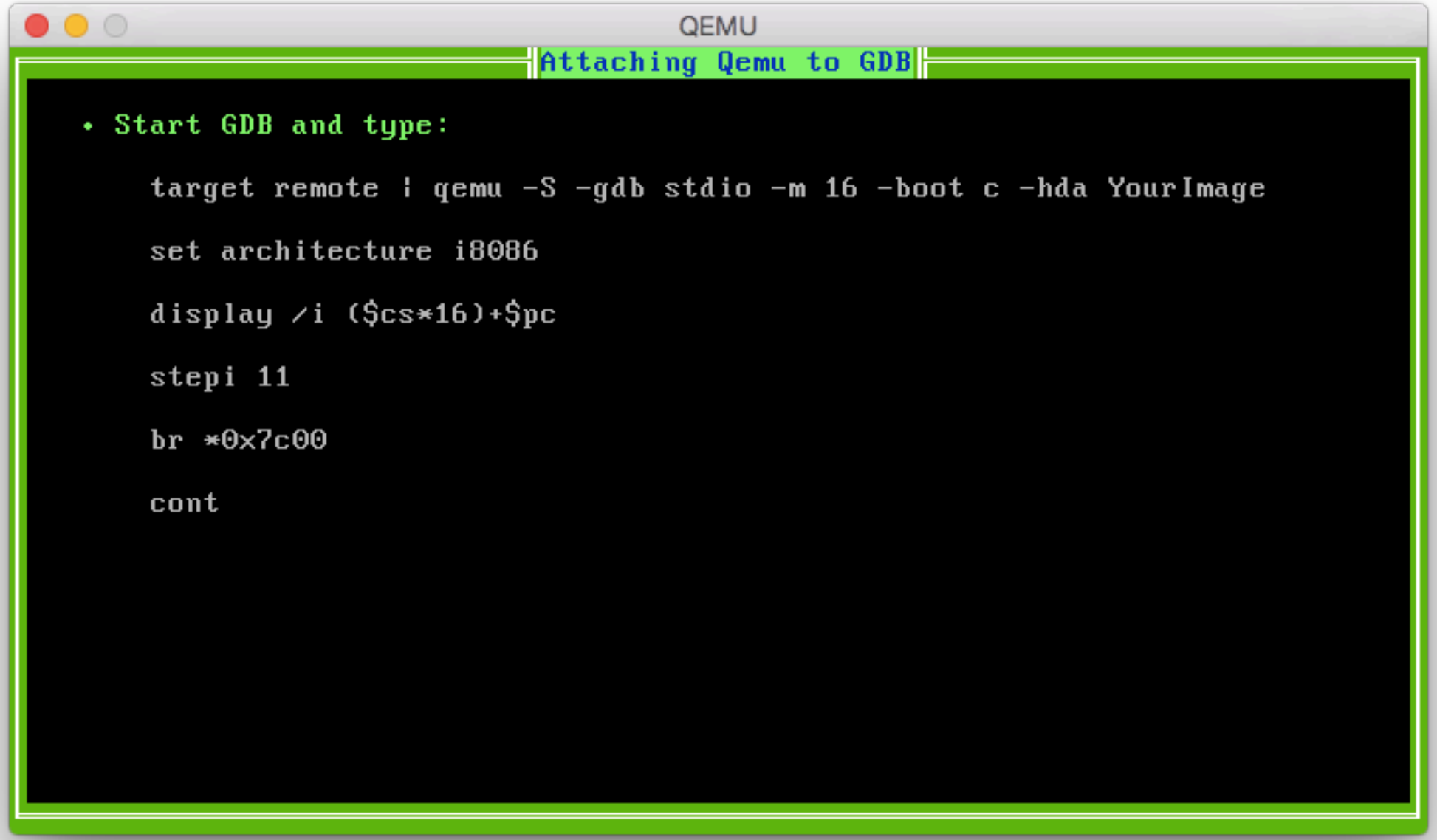

QEMU


QEMU
Random
• Best way to get a random value:
• RDTSC Instruction
• Read Time-Stamp Counter
• Value is returned to AX Register
• LSBs more "random" than MSBs

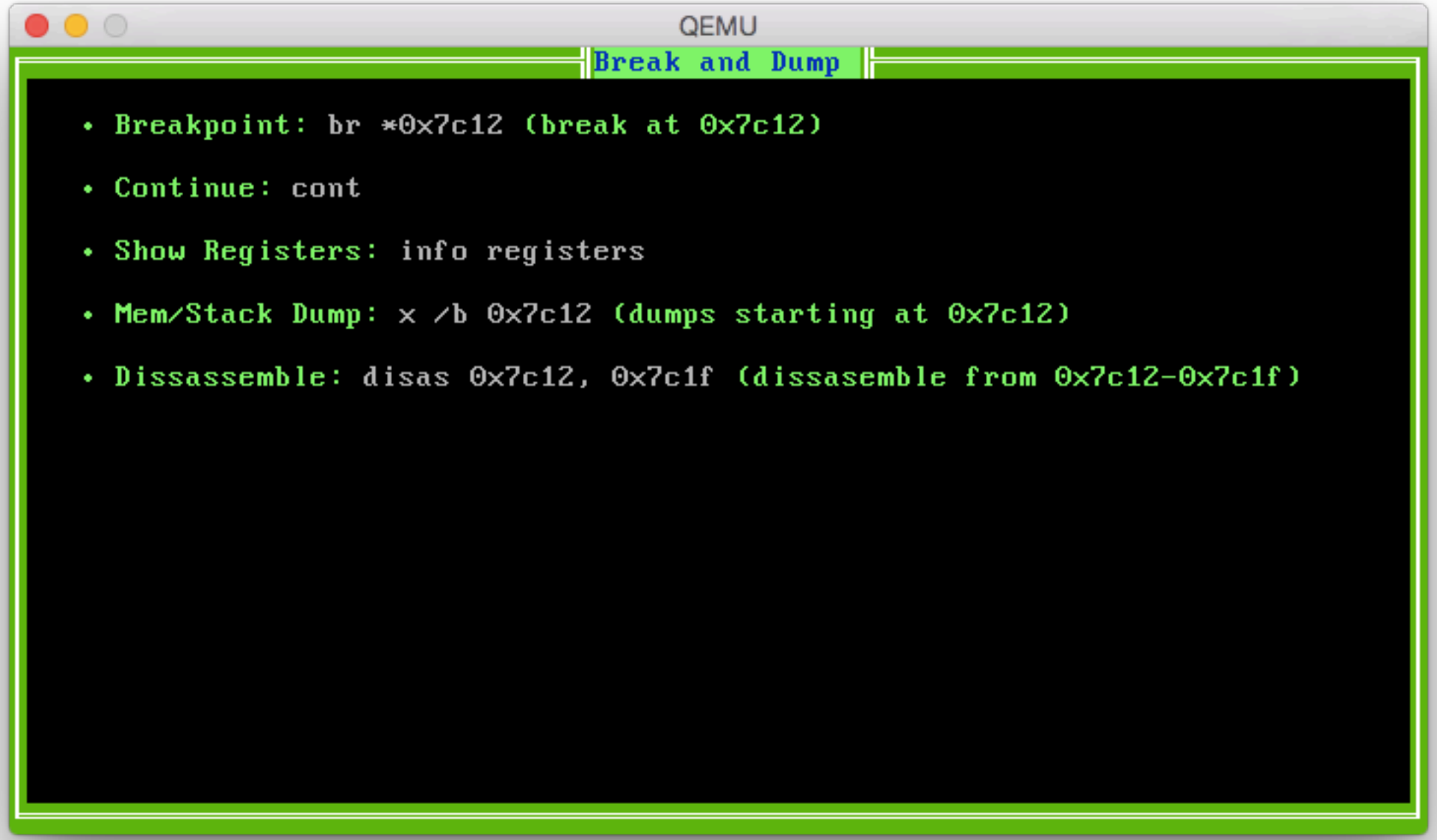

QEMU


QEMU
Debugging
• Debugging Boot Sector Images:
• Using objdump
• Attaching qemu to gdb
• Actually using gdb:
• Stepping
• Breakpoints
• Show Registers & Memory
• Disassemble

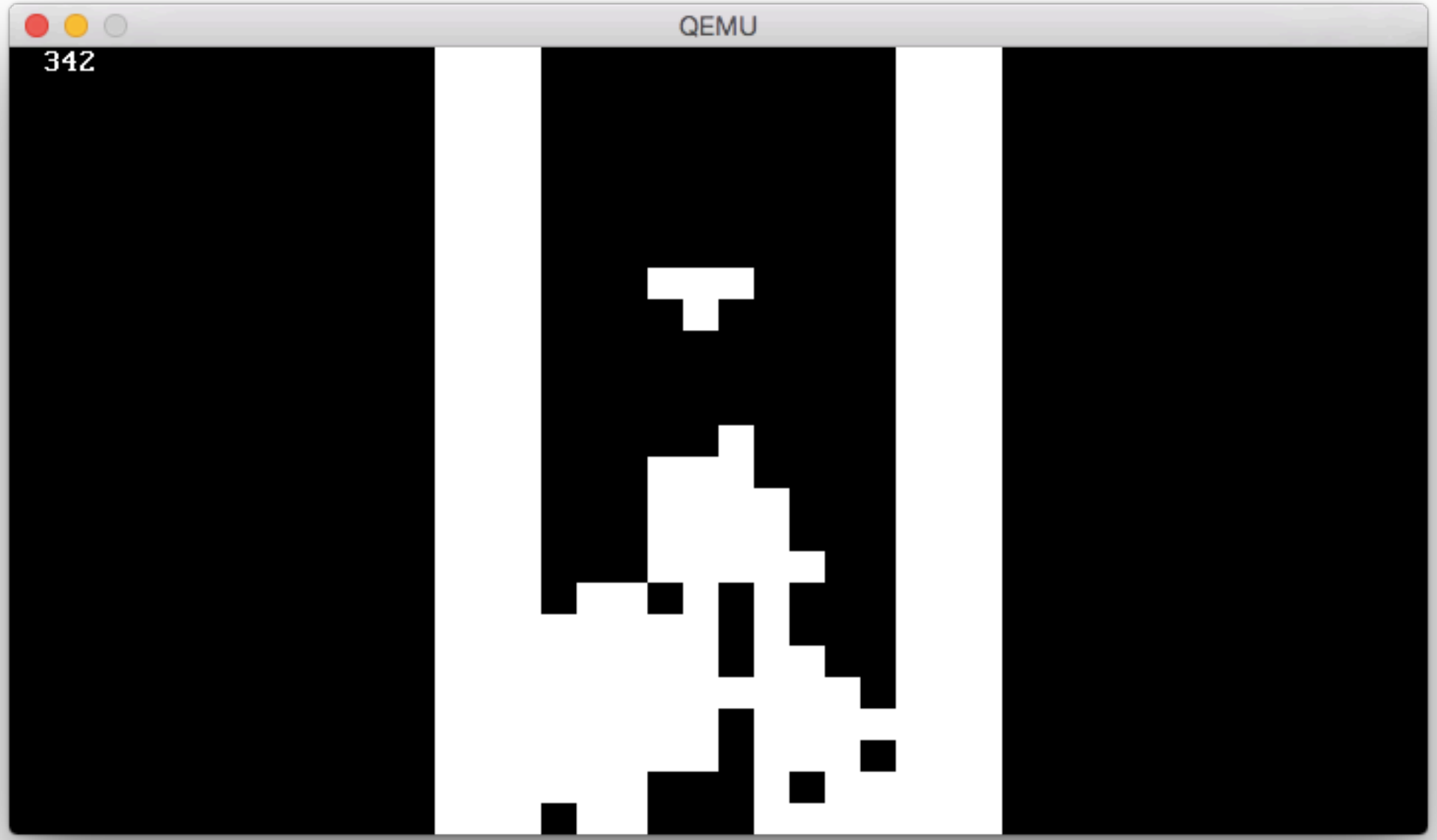

QEMU
Using Objdump
• These programs are 16-bit
• There are no ELF/PE headers
• Recommended Command:
objdump -D -b binary -mi386 -Maddr16,data16 YourImage

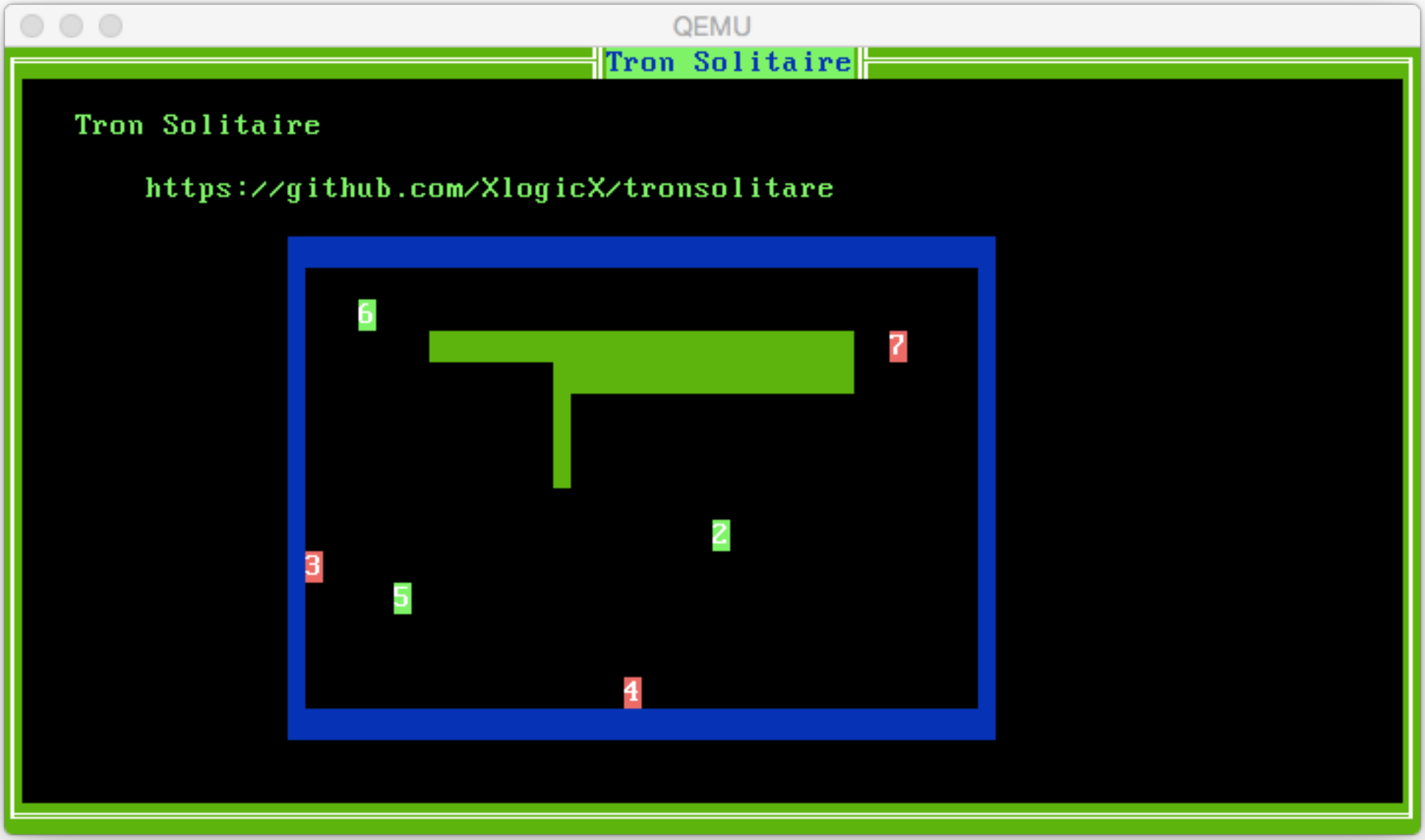

QEMU
Attaching Qemu to GDB
• Start GDB and type:
target remote | qemu -S -gdb stdio -m 16 -boot c -hda YourImage
set architecture i8086
display /i ($cs*16)+$pc
stepi 11
br *0x7c00
cont

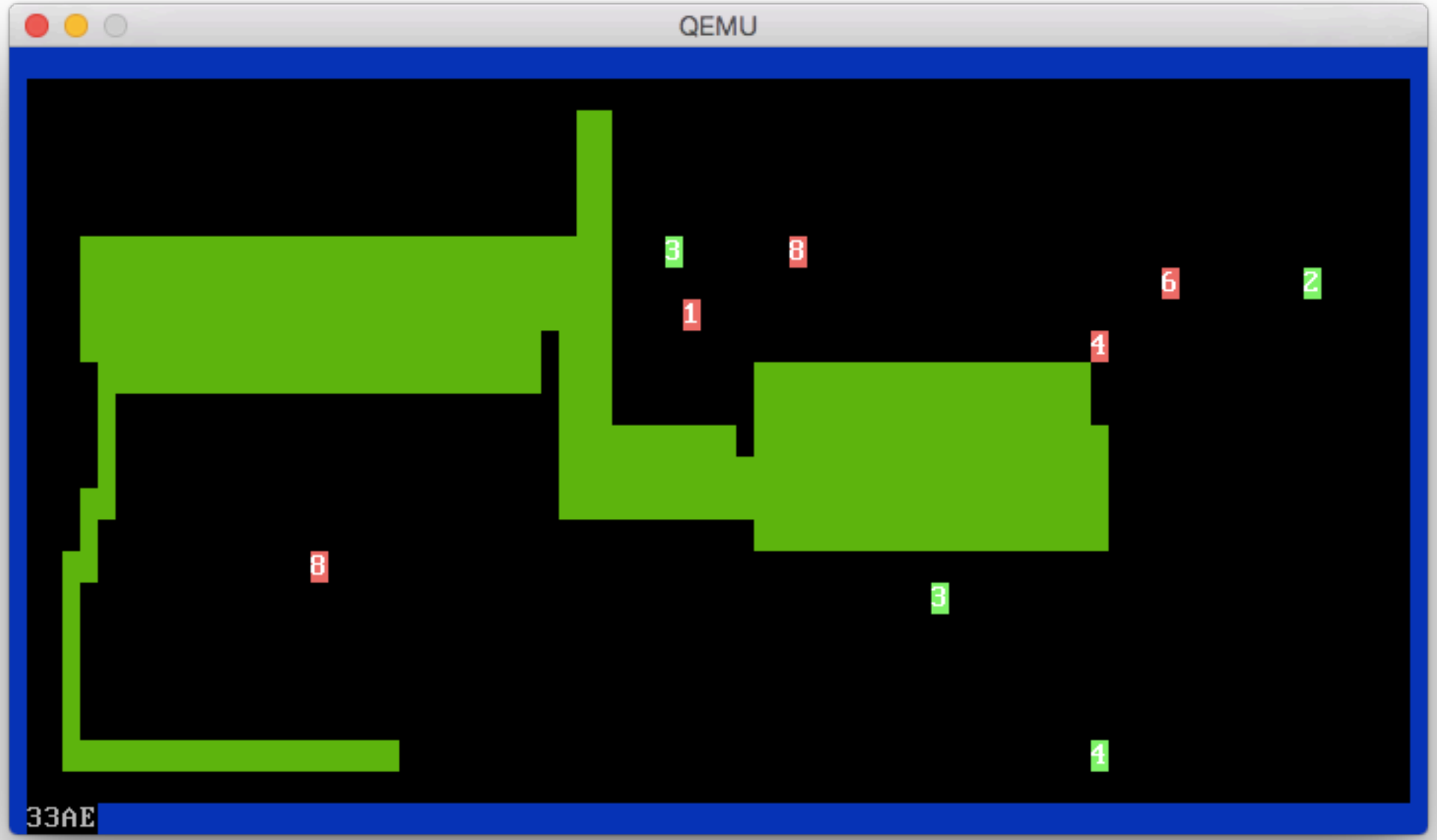

QEMU
Stepping in GDB
• Single Step: step
• Step n times: step 7
• "Step Over" INT 0x10
• Note address of the INT 0x10
• Add 2 to this number
• Set your breakpoint to that address
• Then continue

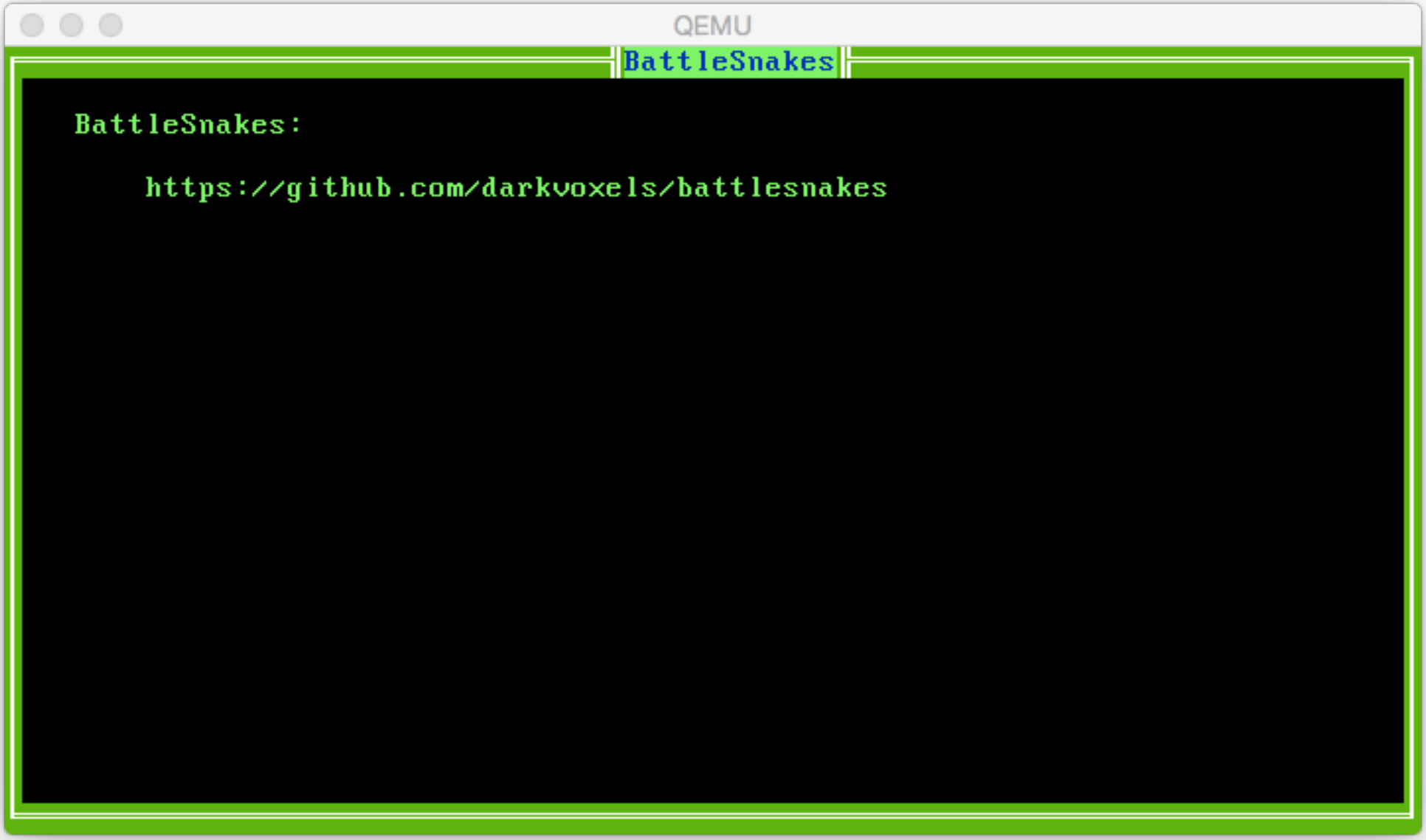

QEMU
Break and Dump
• Breakpoint: br *0x7c12 (break at 0x7c12)
• Continue: cont
• Show Registers: info registers
• Mem/Stack Dump: x /b 0x7c12 (dumps starting at 0x7c12)
• Dissassemble: disas 0x7c12, 0x7c1f (dissasemble from 0x7c12-0x7c1f)

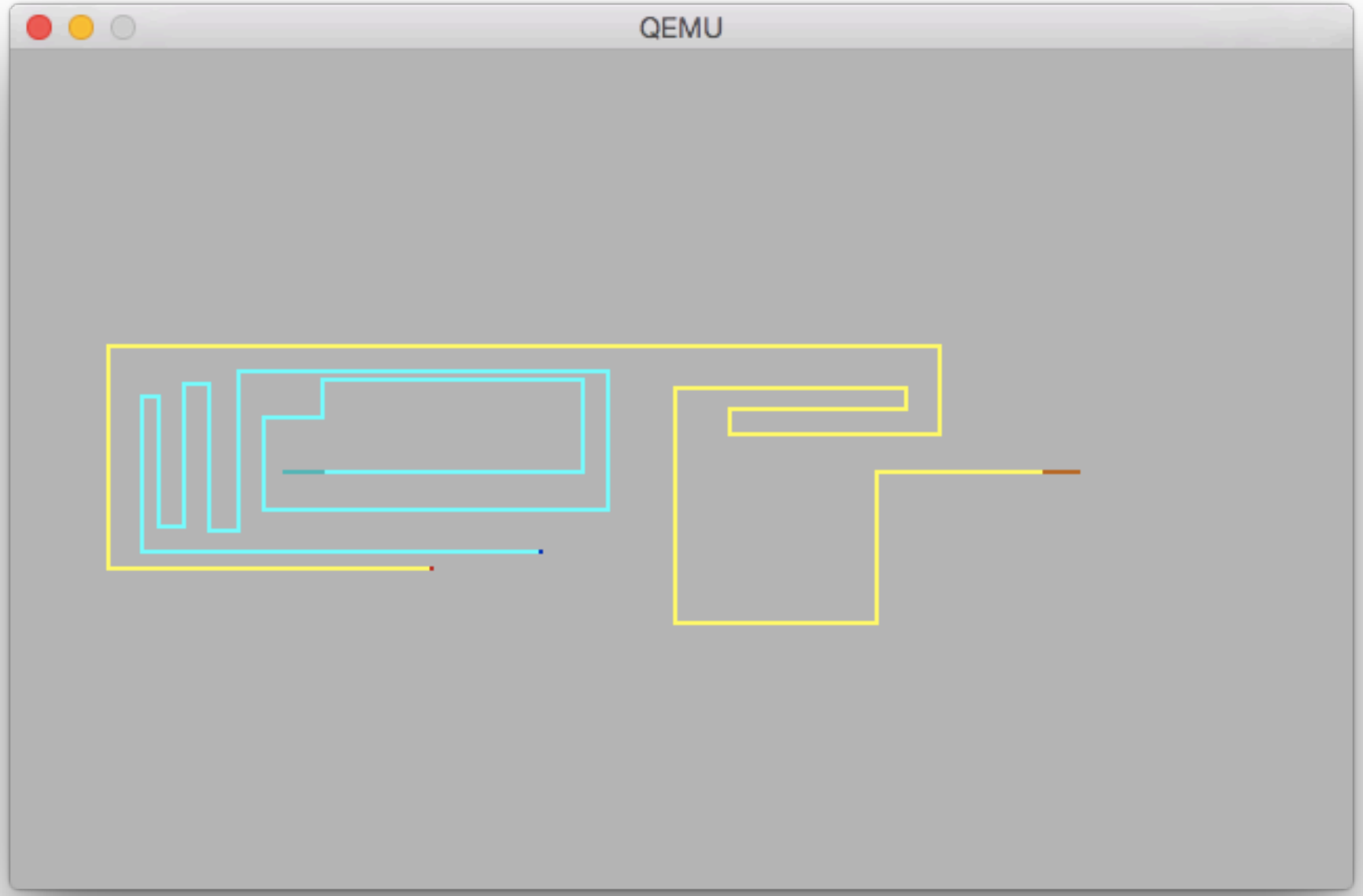

QEMU
Tetranglix
•Tetranglix
•https://github.com/Shikhin/tetranglix

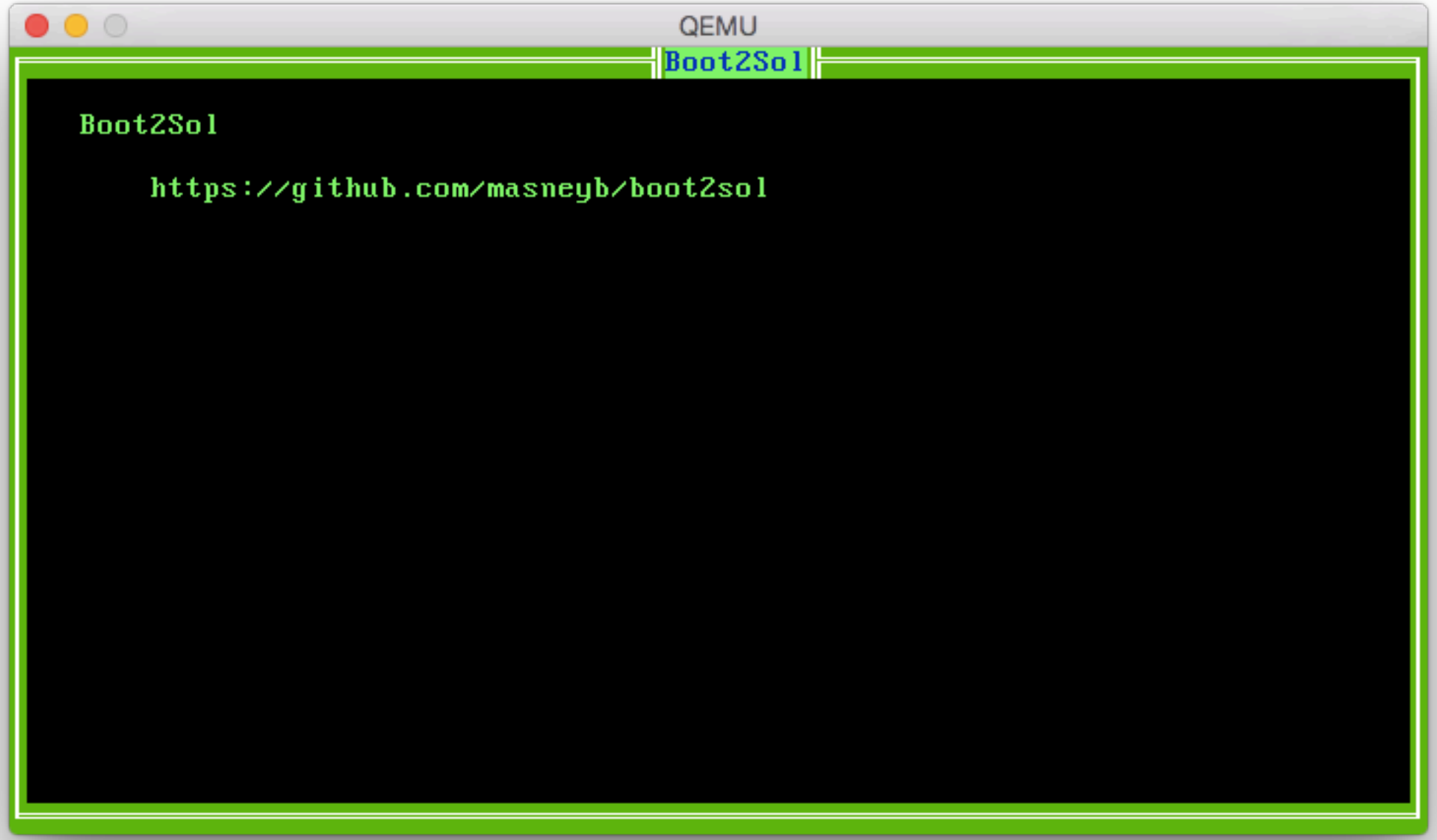

QEMU
342

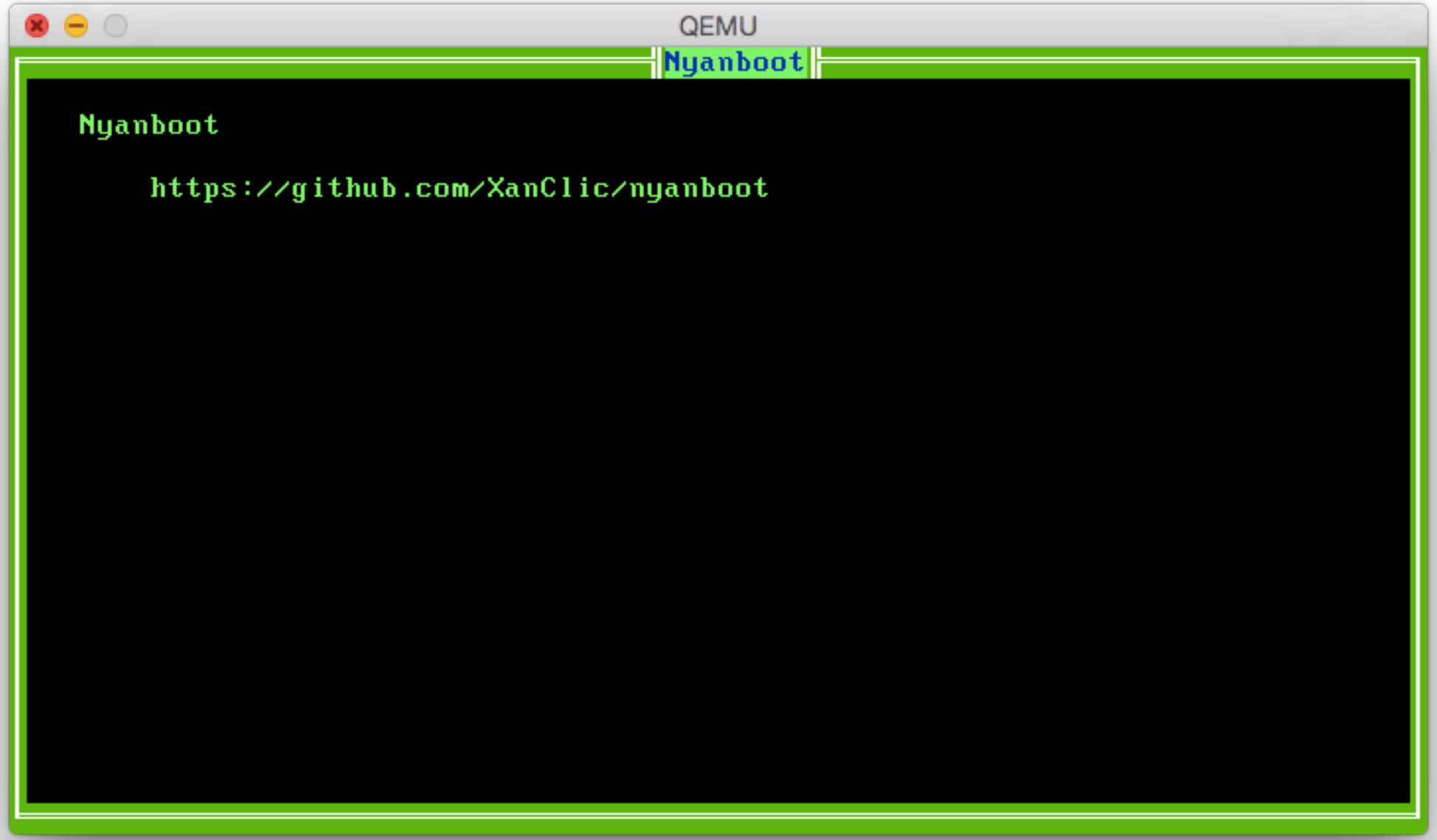

QEMU
Tron Solitaire
Tron Solitaire
https://github.com/XlogicX/tronsolitare
6
7
2
3
5
4

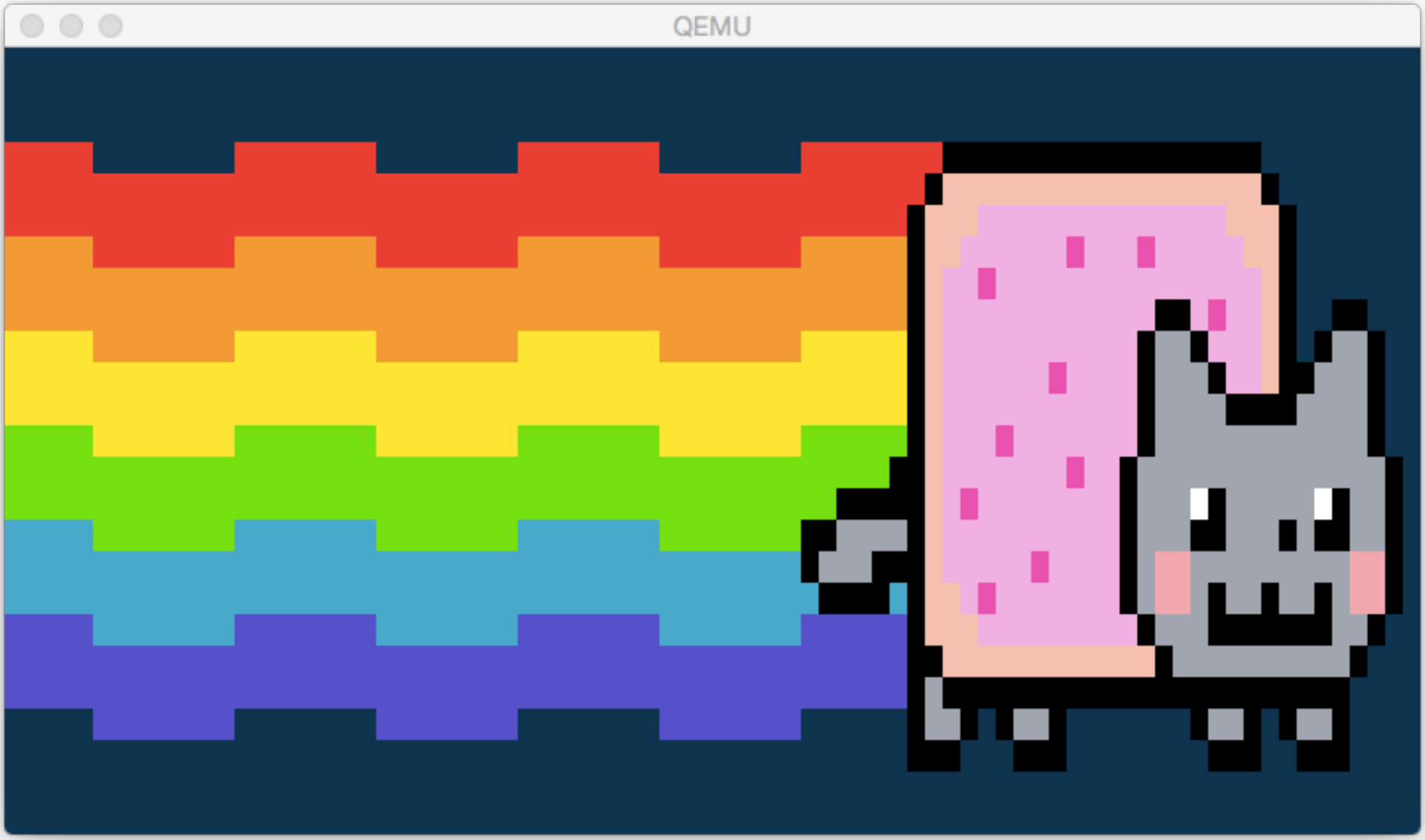

QEMU
3
8
6
2
1
4
8
3
4
33AE

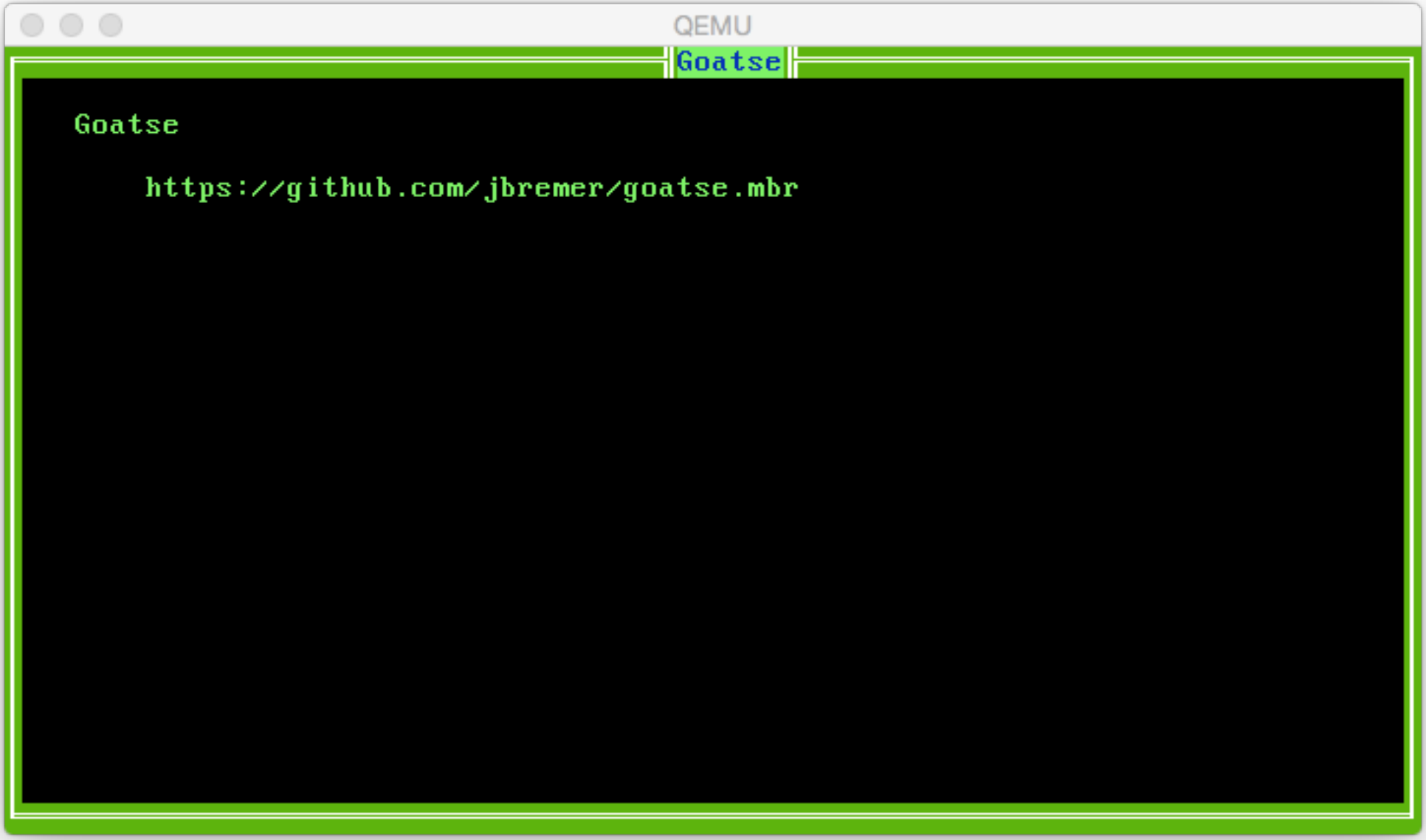

QEMU
BattleSnakes
BattleSnakes:
https://github.com/darkvoxels/battlesnakes

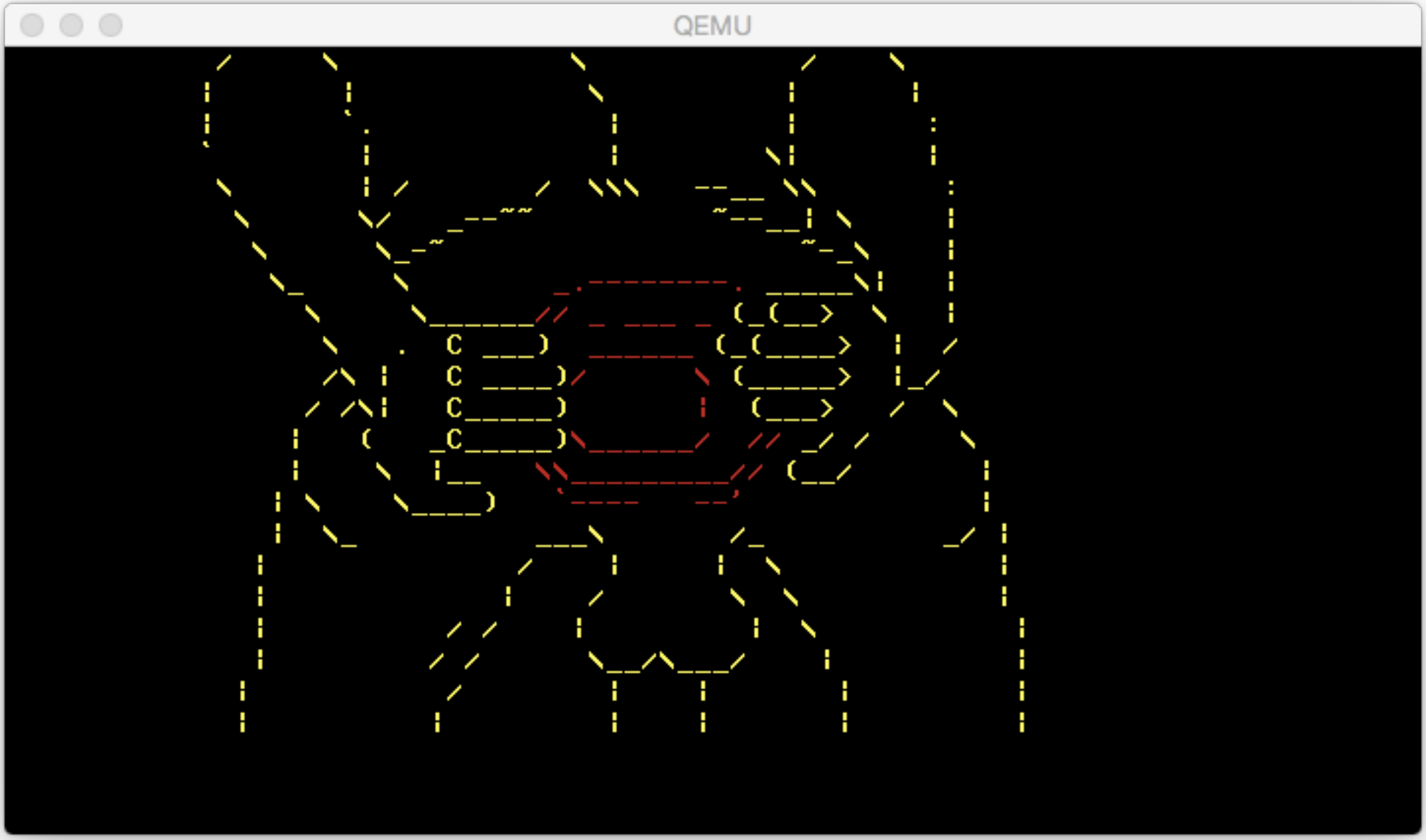

QEMU

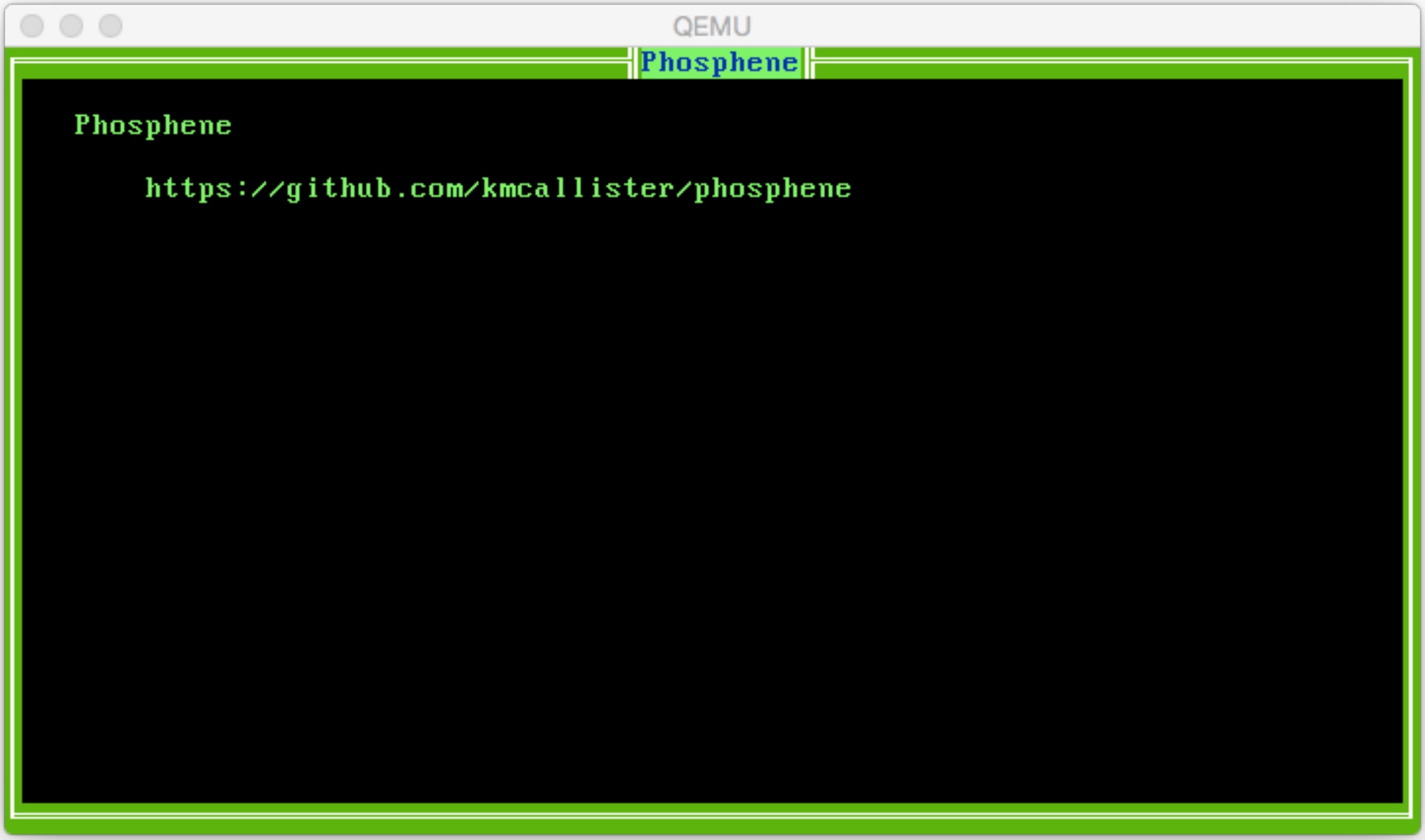

QEMU
Boot2Sol
Boot2Sol
https://github.com/masneyb/boot2sol

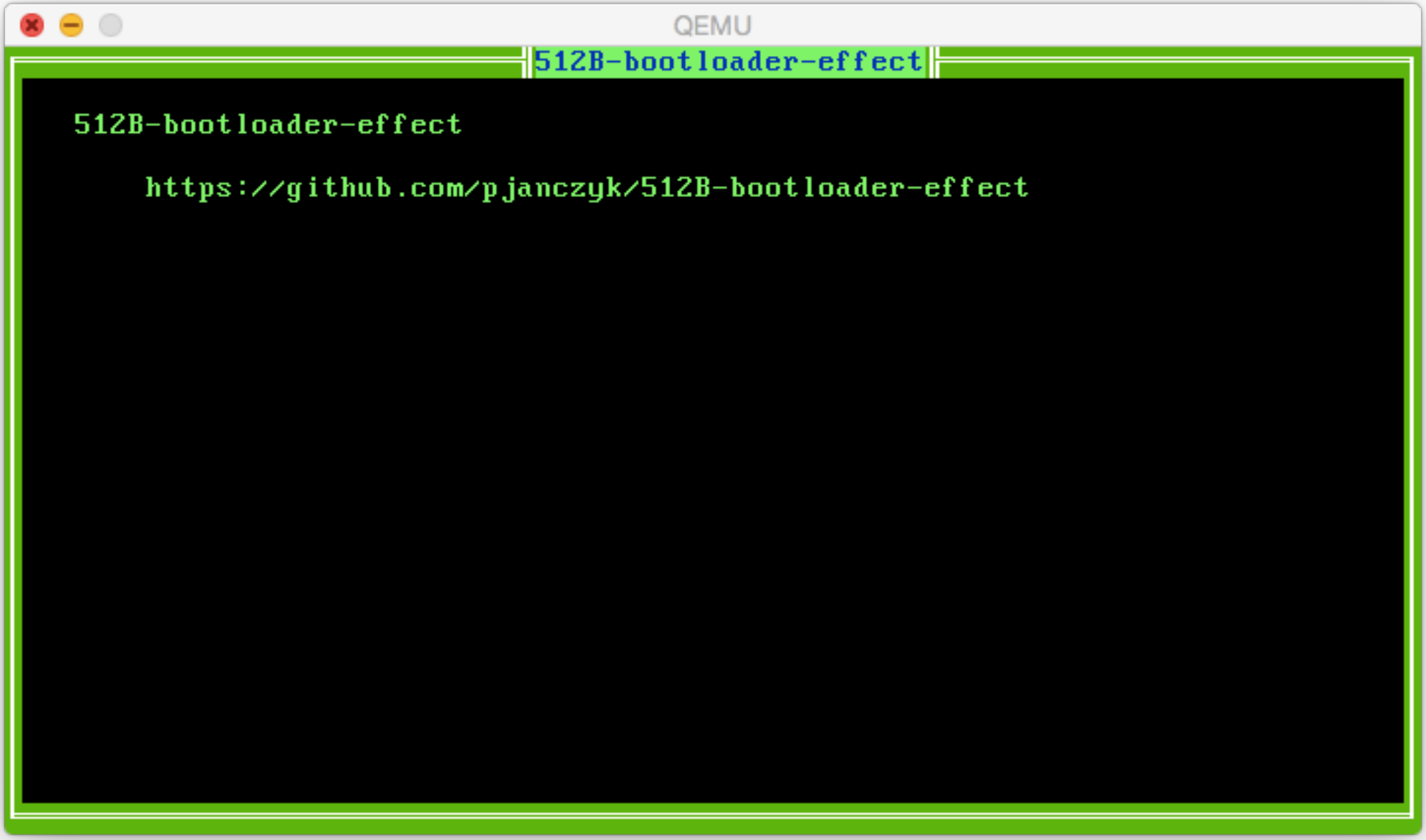

QEMU
Nyanboot
Nyanboot
https://github.com/XanClic/nyanboot

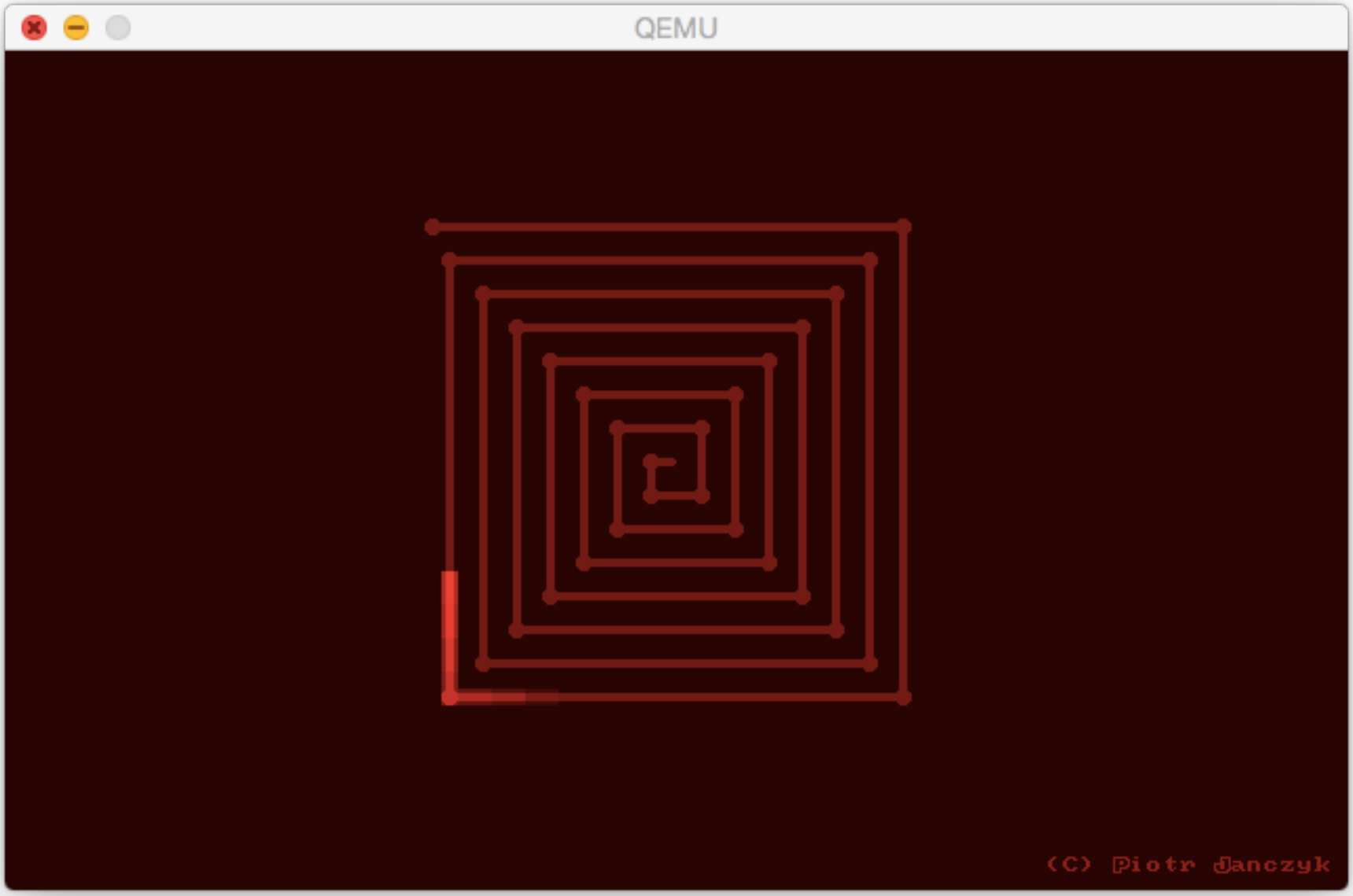

QEMU


QEMU
Goatse
Goatse
https://github.com/jbremer/goatse.mbr


QEMU

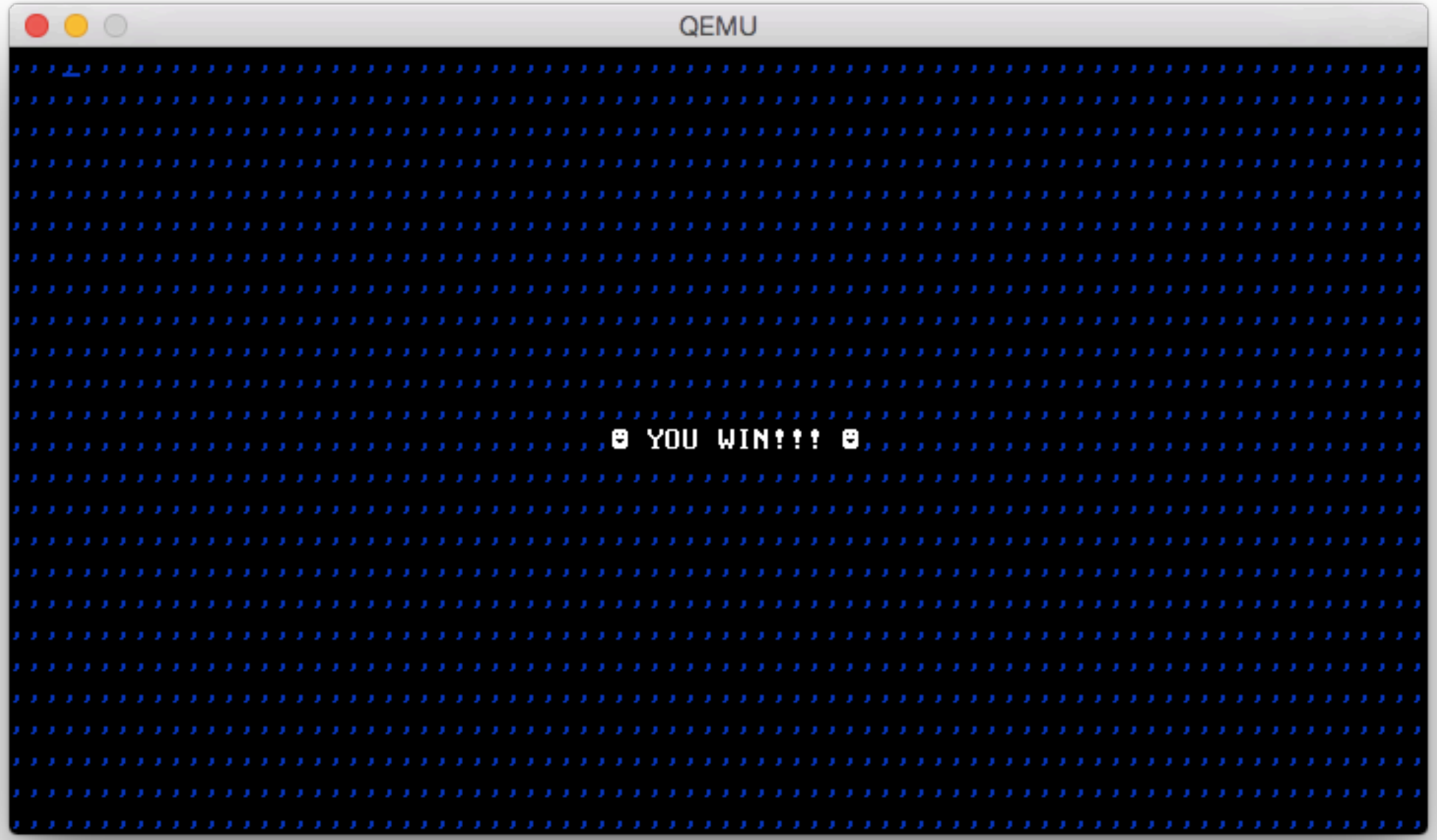

QEMU
Phosphene
Phosphene
https://github.com/kmcallister/phosphene


QEMU
512B-bootloader-effect
512B-bootloader-effect
https://github.com/pjanczyk/512B-bootloader-effect


QEMU
(C) Piotr Janczyk

QEMU

```
>
B0 03 CD 10 31 C0 8E D8 8E D0 BC 00 9C B4 B8 8E C0 80 3E 67 7D 42 0F 84 01
01 C7 06 64 7D 01 00 C6 06 68 7D 00 B9 D0 07 B8 20 0F 31 FF F3 AB 31 FF B8
3E 0F AB B8 20 0F AB B4 02 30 FF 8B 16 64 7D CD 10 B4 01 CD 16 30 E4 CD 16
3C 64 74 06 3C 69 74 60 EB EE BB 00 7C BF A0 00 C6 06 63 7D 18 31 C0 8A 07
C0 E8 04 67 8A 80 53 7D 00 00 B4 0F AB 31 C0 8A 07 24 0F 67 8A 80 53 7D 00
00 B4 0F AB B0 20 AB 43 81 FB 00 7E 74 17 80 3E 63 7D 00 74 06 FE 0E 63 7D
EB C8 83 C7 0A C6 06 63 7D 18 EB BE B4 01 CD 16 30 E4 CD 16 80 FC 1C 0F 84
60 FF EB EF B4 01 CD 16 30 E4 CD 16 80 FC 1C 74 26 80 3E 68 7D 02 74 EC 8B
1E 68 7D 88 87 66 7D FE 06 68 7D B4 0F AB FF 06 64 7D B4 02 30 FF 8B 16 64
7D CD 10 EB CD 80 2E 66 7D 30 80 3E 66 7D 10 7F 02 EB 05 80 2E 66 7D 07 80
2E 67 7D 30 80 3E 67 7D 10 7F 02 EB 07 80 2E 67 7D 07 EB 00 A0 66 7D C0 E0
04 08 06 67 7D E9 F6 FE BA 00 01 42 8B 1E 6C 04 83 C3 02 39 1E 6C 04 75 FA
89 D0 31 FF B9 D0 07 F3 AB BF C4 07 B0 0F BE 45 7D B1 0E A4 AA E2 FC EB D9
02 20 59 4F 55 20 57 49 4E 21 21 21 20 02 30 31 32 33 34 35 36 37 38 39 41
42 43 44 45 46 13 01 00 06 69 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 55 AA
```


QEMU
YOU WIN!!!