Digging in the Dirt

Edward Marczak

marczak@radiotope.com

@marczak

Digging in the Dirt

Edward Marczak

marczak@radiotope.com

@marczak

Tycho

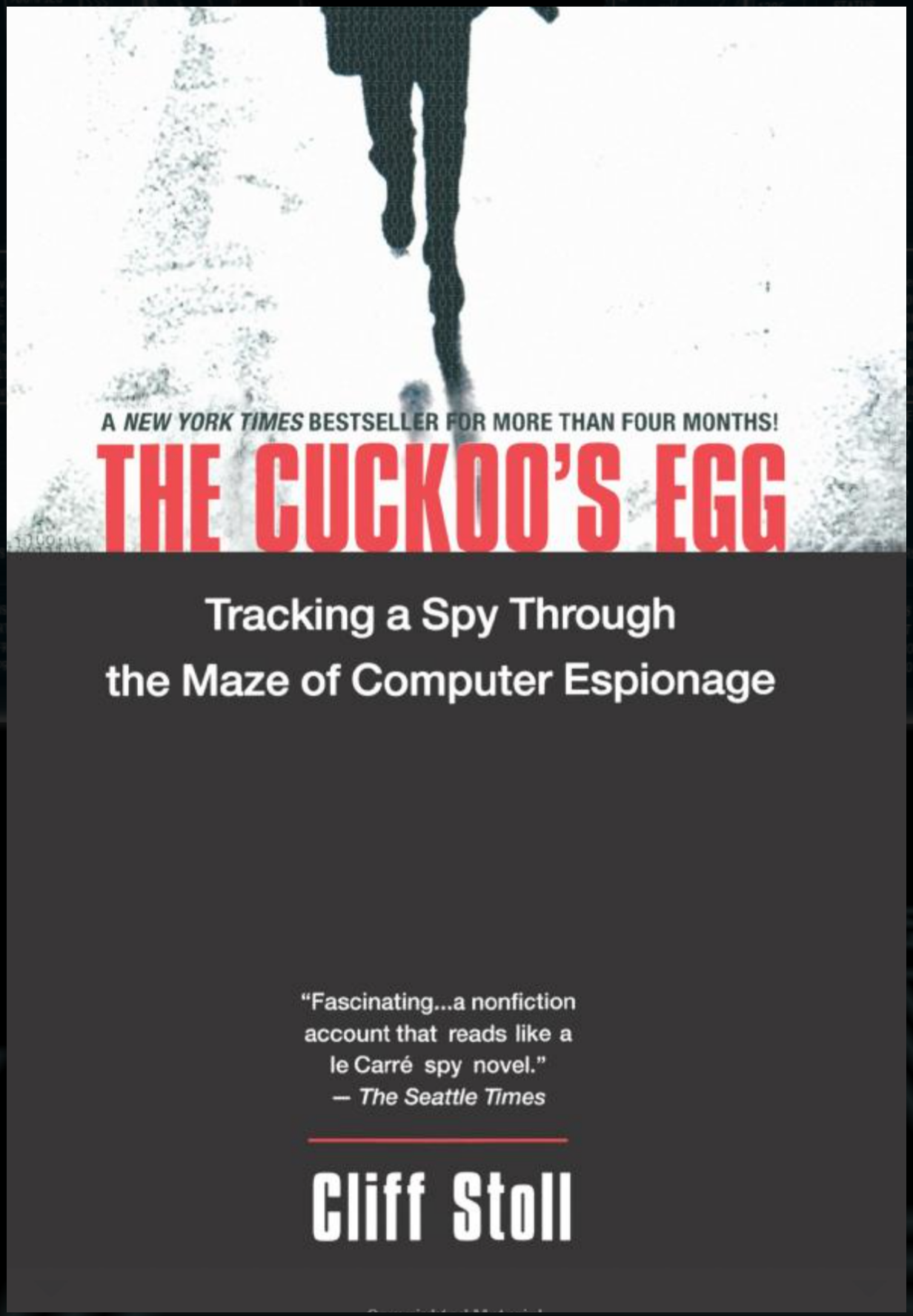
TYCHO _ØØ5917

traxIV<40c<40c> 48c<48c>

Tycho

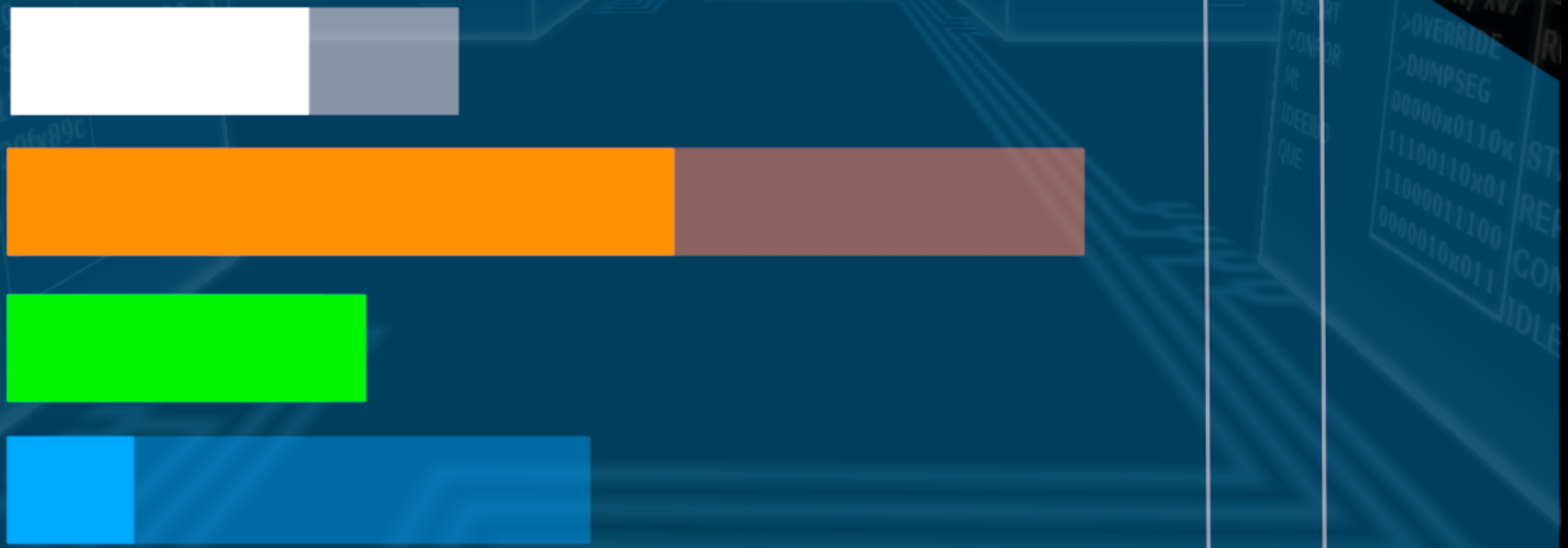
https://goo.gl/FFuvkn

2


A NEW YORK TIMES BESTSELLER FOR MORE THAN FOUR MONTHS!
THE CUCKOO'S EGG
Tracking a Spy Through
the Maze of Computer Espionage
"Fascinating...a nonfiction account that reads like a le Carré spy novel."
— The Seattle Times
Cliff Stoll

I thought what I'd do was, I'd pretend I was one of those dea

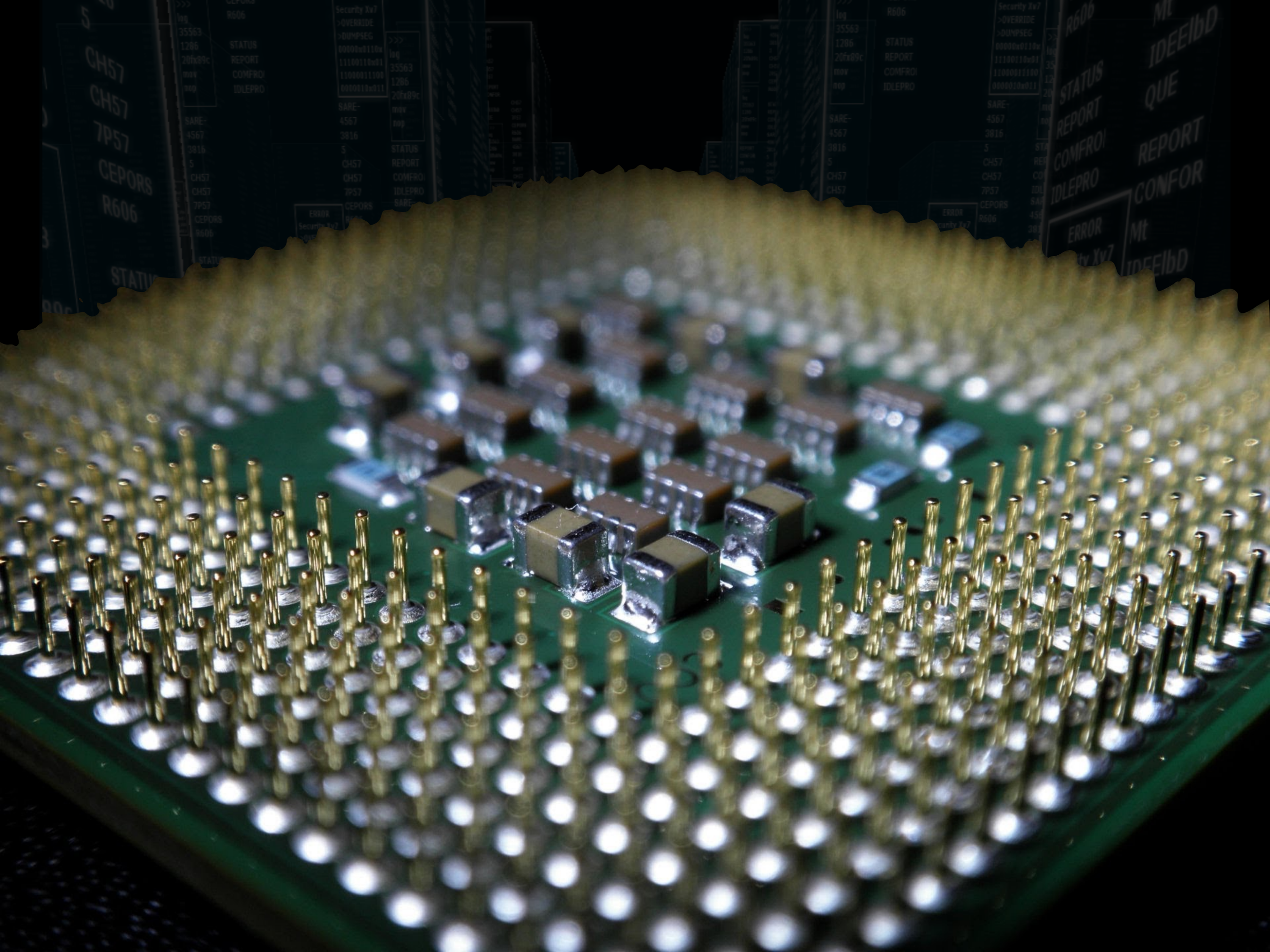
Computer Science

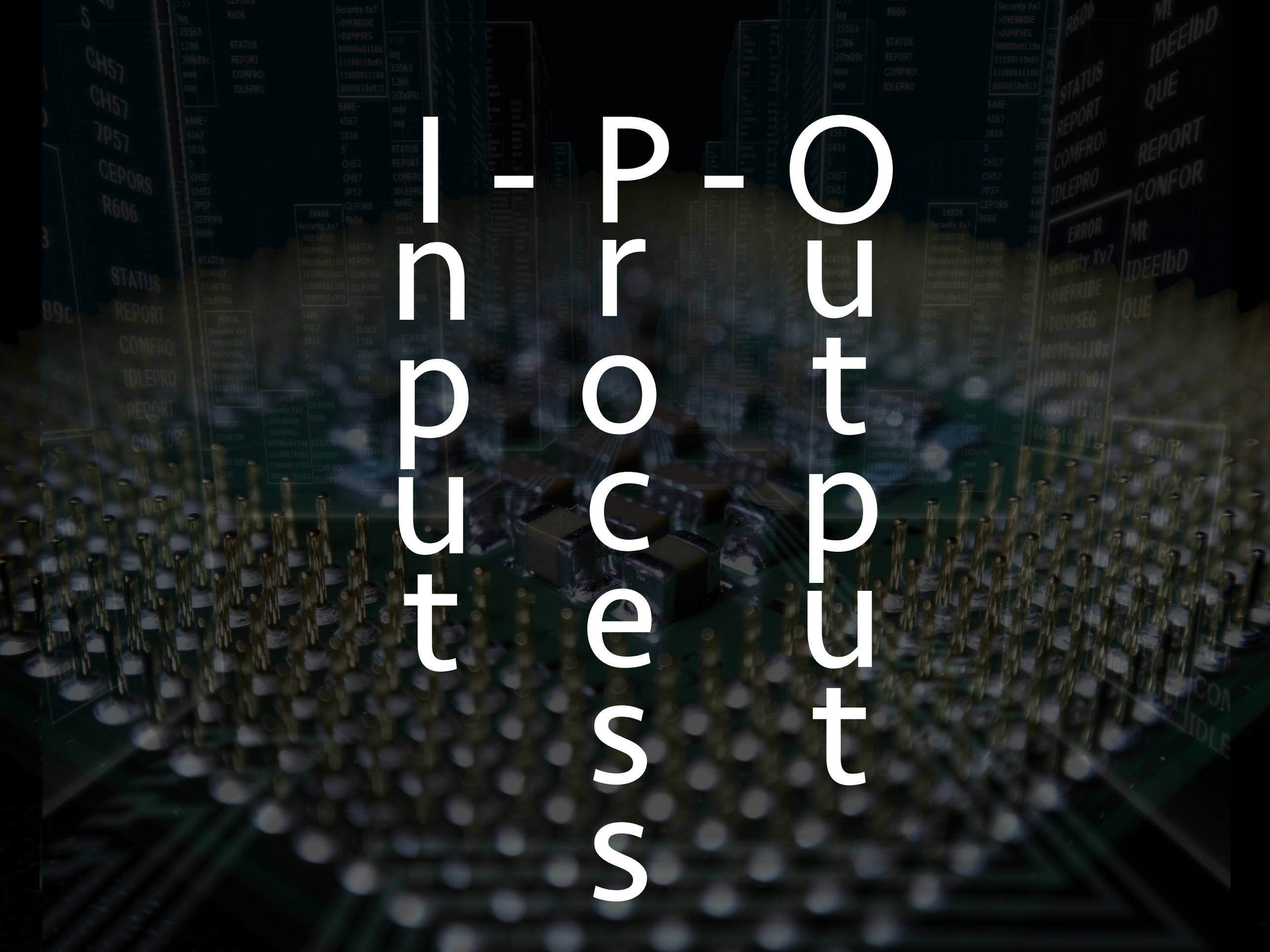


THINKING LARGE.
BOOK WHOLE
DRAGON.

That hunger for knowledge in saturated stomach is an ideal combination.

Architecture




I-P-O
input process output

MAX

Storage

CPU

0	ax
1	bx
2	cx
3	dx
4	si
5	di
6	bp
7	sp

Data

Instructions

I/O

ALU

0

```
MOV    76,AX
ADD    1,AX,BX
MOV    BX,#1000
```

```
MOV   12,AX
_BEGIN:
ADD   1,AX
CMP   AX,24
JE    #300
JMP   _BEGIN
```

Mnemonic	Opcode
add	0000
sub	0001
mov	0010
test	0011

Register	Code
AX	00
BX	01
CX	10
DX	11

0010 0000110000000000

0000 0000000010001

0010 0100000011111101000

mov

0010 0000110000000000

0000 0000000010001

0010 0100000011111101000

12

0010 0000110000000000

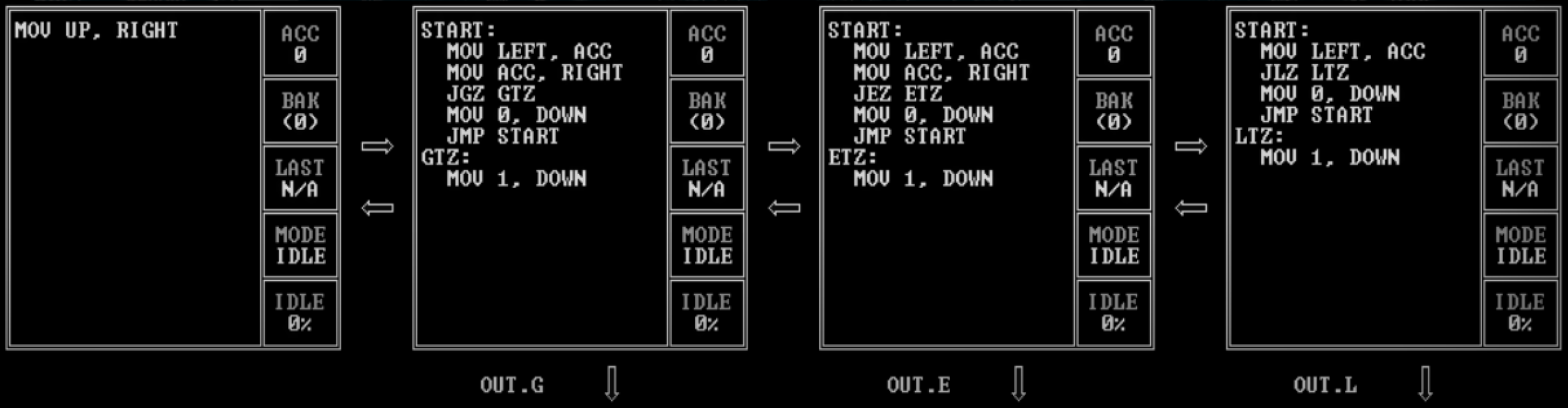
0000 0000000010001

0010 0100000011111101000

ax

```
0010 000011000000000000
0000 000000010001
0010 010000001111101000
```

2


MOV UP, RIGHT
ACC 0
BAK (0)
LAST N/A
MODE IDLE
IDLE 0%
START:
MOV LEFT, ACC
MOV ACC, RIGHT
JGZ GTZ
MOV 0, DOWN
JMP START
GTZ:
MOV 1, DOWN
ACC 0
BAK (0)
LAST N/A
MODE IDLE
IDLE 0%
OUT.G
START:
MOV LEFT, ACC
MOV ACC, RIGHT
JEZ ETZ
MOV 0, DOWN
JMP START
ETZ:
MOV 1, DOWN
ACC 0
BAK (0)
LAST N/A
MODE IDLE
IDLE 0%
OUT.E
START:
MOV LEFT, ACC
JLZ LTZ
MOV 0, DOWN
JMP START
LTZ:
MOV 1, DOWN
ACC 0
BAK (0)
LAST N/A
MODE IDLE
IDLE 0%
OUT.L

ACC
0

BAK
<0>

LAST
N/A

MODE
IDLE

IDLE
0%

```
START:
  MOV LEFT, ACC
  MOV ACC, RIGHT
  JGZ GTZ
  MOV 0, DOWN
  JMP START
GTZ:
  MOV 1, DOWN
```

ACC
0

BAK
<0>

LAST
N/A

MODE
IDLE

IDLE
0%

OUT.G

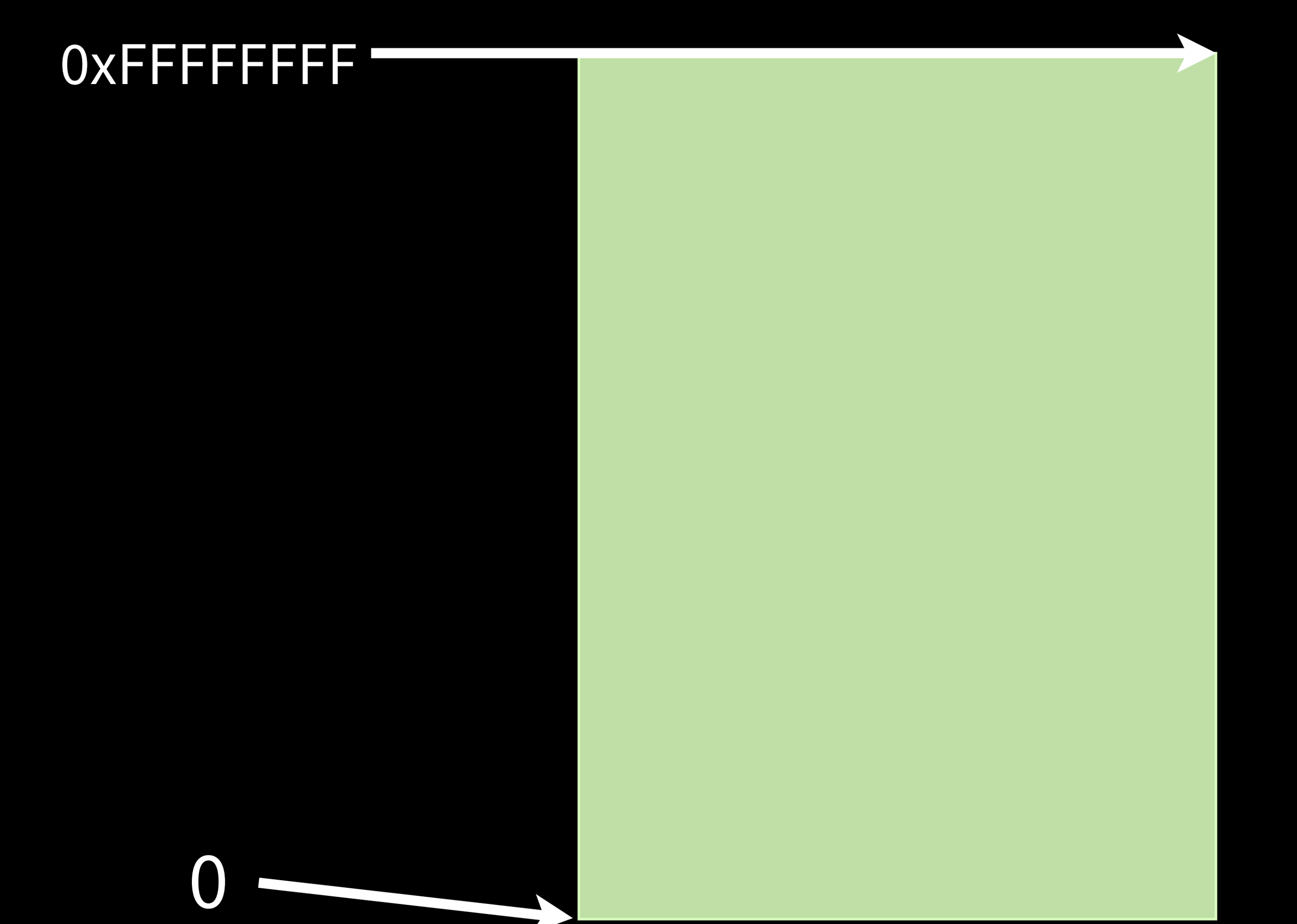
```
 0 |
 1 ClearText()
 2
 3 Print("---")
 4 Print("Enter command followed by Enter.")
 5 Print("'h' for help")
 6 Print("---")
 7 Print("")
 8
 9 loop
10  var command = Input("> ")
11  Dispatch(command)
12 end
13
14 void Dispatch(string command)
15  if command == "h" or command == "H" or command == "help"
16          Print("boot - Boot from floppy disk")
17    Print("net - connect to computer")
18          Print("info - System information")
19      else if command == "boot"
20          #BootFromFloppy() # enable if the computer has a driv
21      else if command == "net"
22          var target = Input("Target: ")
23          var id = Connect(target)
24          Print("Connection id: " + id)
25      else if command == "info"
26          Info()
27  else
28   Print("Can't understand command '" + command + "'")
29          Print("Enter 'h' for help")
30  end
31 end
```

```
Suggestions: Count Range GetIndexes Remove RemoveAll HasIndex
Append Round Int Mod Info Random GetUser Name Time Width
Height GetHour GetMinute Sleep Quit CharToInt IntToChar
ClearText Print PlaySound Pitch Sin Cos Print$ Input Line Text
Rect DisplayGraphics Color Repeat HSVtoRGB RGBtoHSV Lines
GetConnections Slurp Connect GetMemories ClearMemories
HasMemory SaveMemory LoadMemory IsKeyPressed HasFloppy
LoadData ClearData SaveData BootFromFloppy
```

COMPILE

UNDO

RESET


0xFFFFFFFF
0

Here be dragons

OS

Here be dragons

OS

Accessible RAM

Here be dragons

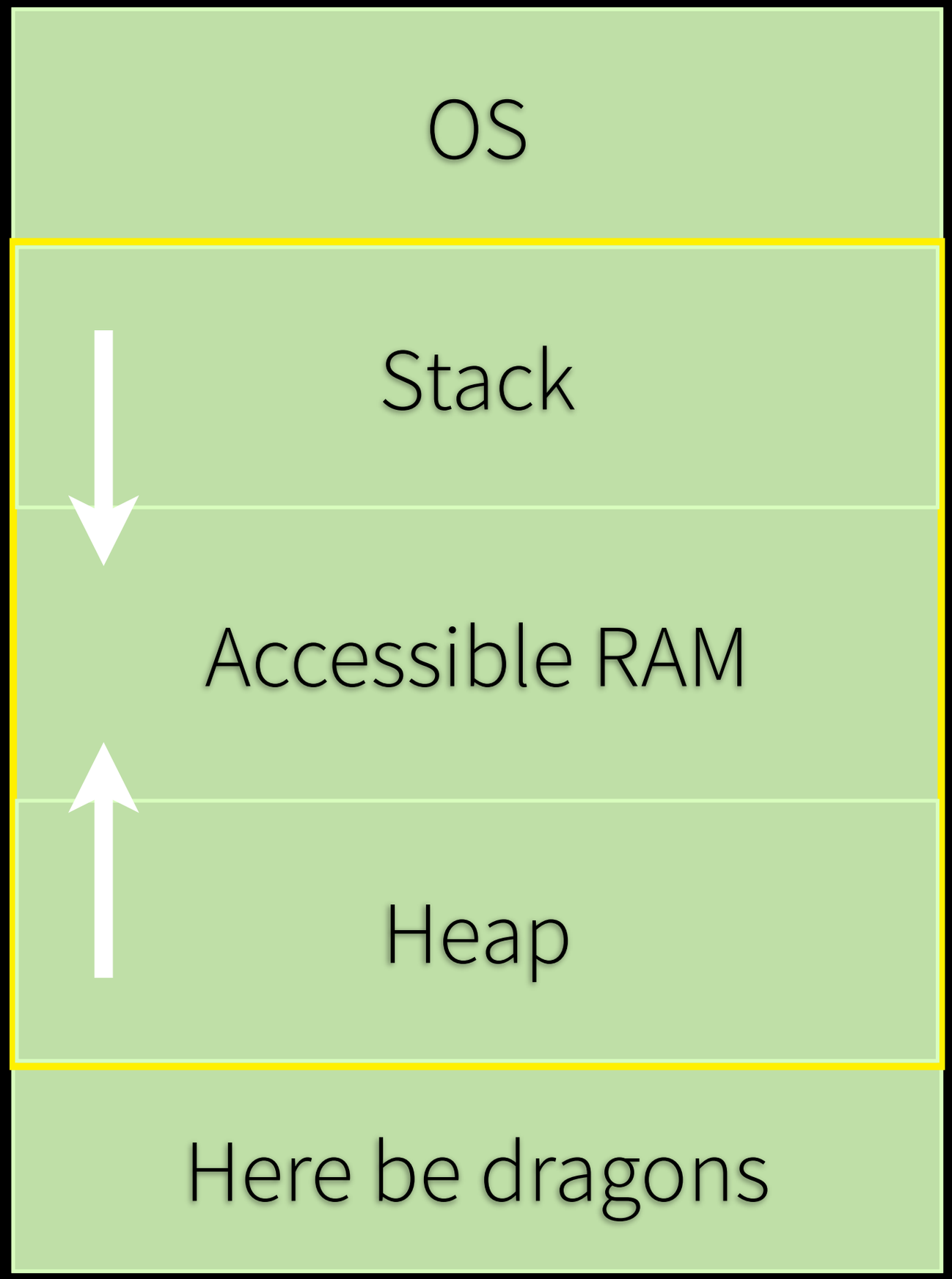
OS

Stack

Accessible RAM

Heap

Here be dragons


OS
Stack
Accessible RAM
Heap
Here be dragons

Stack
env, argv, argc, vars for main()

Available for stack and heap growth

Heap

.data

.text

Microsoft Word Info

Microsoft Word
Microsoft Word 4.0A

Kind : application program
Size : 671K on disk (684,134 bytes used)

Where : MacintoshHD : Word 4.0 :

Created : Wed, Oct 4, 1989, 5:27 PM
Modified : Wed, May 9, 1990, 5:17 AM
Version : 4.0A, © 1987-1989 Microsoft Corporation

Comments :

☐ **Locked**

Memory
Suggested size : 512 K
Current size : 1024 K

```c
int add(int x, int y) {
   int sum = 0;
   sum = x + y;
   return sum;
}
```

```c
int main(int argc, char *argv[]) {
   int x = 0;
   x = add(4,5);
   printf("The sum is %d.\n", x);
}
```

```
(lldb) bt
* thread #1: tid = 0x17d04f, 0x0000000100000f00
stacktrace`add, queue = 'com.apple.main-thread', stop
reason = instruction step into
  * frame #0: 0x0000000100000f00 stacktrace`add
    frame #1: 0x0000000100000f49 stacktrace`main + 41
    frame #2: 0x00007fff8de905c9 libdyld.dylib`start + 1
    frame #3: 0x00007fff8de905c9 libdyld.dylib`start + 1
```

```
int add(int x, int y) {
   int sum = 0;
   x = fuzzit(x);
   sum = x + y;
   return sum;
}

int main(int argc, char *argv[]) {
   int x = 0;
   x = add(4,5);
   printf("The sum is %d.\n", x);
}
```

```
(lldb) bt
* thread #1: tid = 0x17bfff, 0x0000000100000ec0
stacktrace`fuzzit, queue = 'com.apple.main-thread', stop
reason = instruction step into
  * frame #0: 0x0000000100000ec0 stacktrace`fuzzit
    frame #1: 0x0000000100000f06 stacktrace`add + 38
    frame #2: 0x0000000100000f49 stacktrace`main + 41
    frame #3: 0x00007fff8de905c9 libdyld.dylib`start + 1
    frame #4: 0x00007fff8de905c9 libdyld.dylib`start + 1
```

```c
int add(int x, int y) {
   int sum = 0;
   sum = x + y;
   return sum;
}
```

```c
int main(int argc, char *argv[]) {
   int x = 0;
   x = add(4,5);
   printf("The sum is %d.\n", x);
}
```

Calling Conventions

GPRs

ax, bx, cx, dx

GPRs

ax, bx, cx, dx

ah, al

Specific Registers

bp, ip, sp

“flags”

32-bit, 64-bit
ax - 16 bit

32-bit, 64-bit

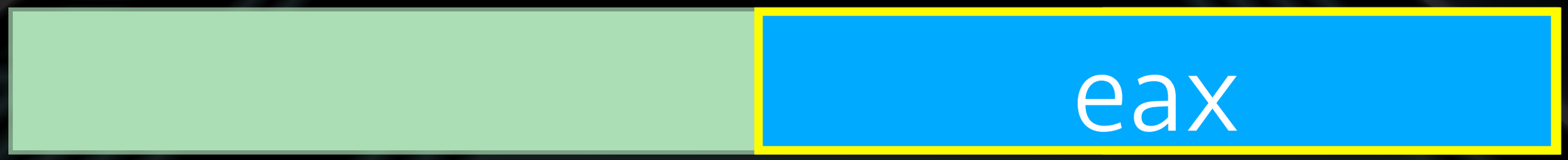
ax - 16 bit

32-bit, 64-bit
ax - 16 bit

32-bit, 64-bit

ax - 16 bit

eax - 32 bit



32-bit, 64-bit

ax - 16 bit

eax - 32 bit

rax - 64 bit

Calling Conventions

RDI, RSI, RDX, RCX, R8, R9

XMM0, XMM1, XMM2, XMM3, XMM4, XMM5, XMM6, XMM7

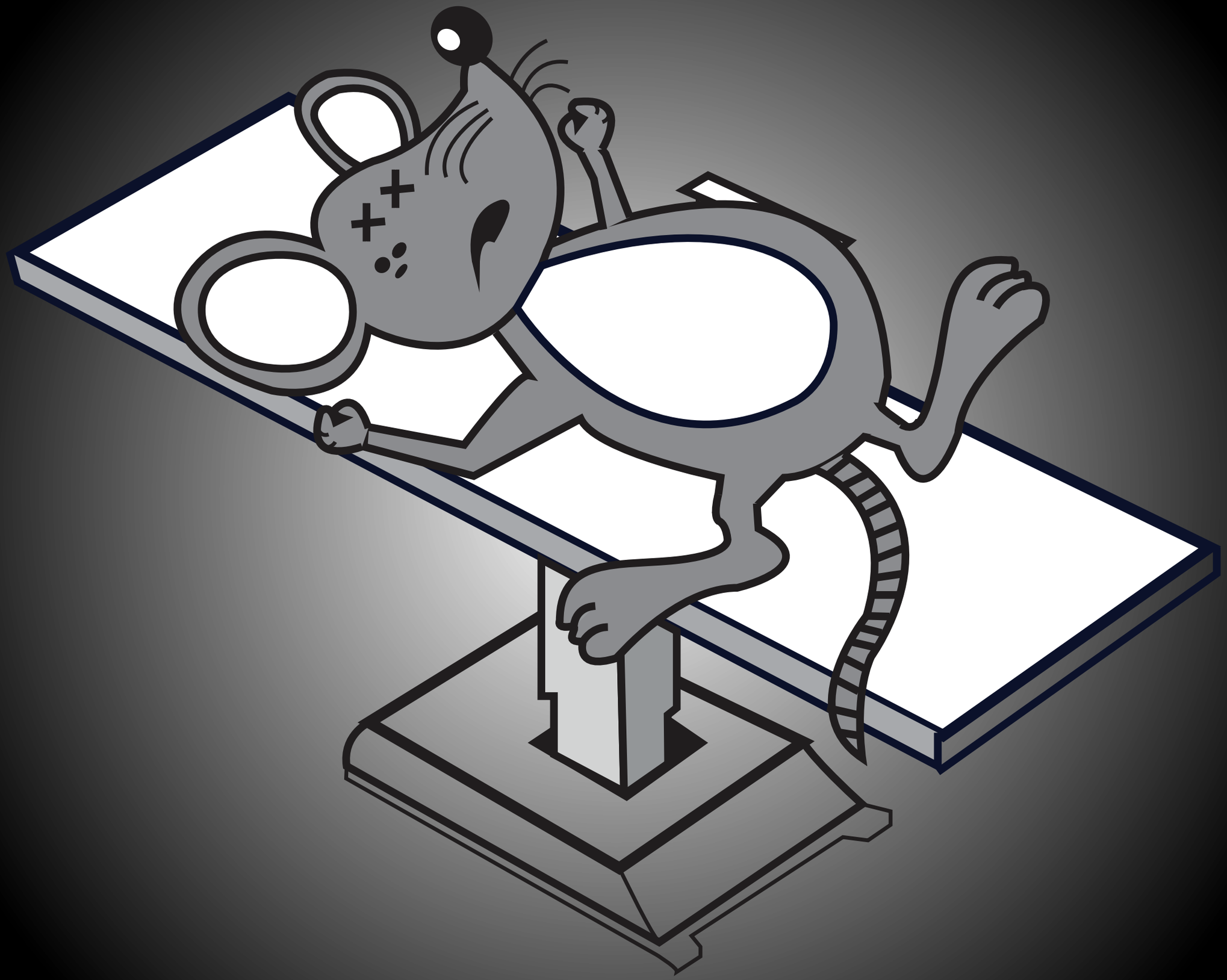
Calling Conventions

RBP - “Callee save”

RAX - Return value

Examining code
aka
Reverse Engineering
aka
RE

2



Check ‘yo self

Check ‘yo self
Before you wreck ‘yo self

Cos I'm bad for your health, I come real stealth
Droppin bombs on ya moms, kill car alarms
So come on and chickity-check yo self before you wreck yo self

Check ‘yo self
Before you wreck ‘yo self
‘cause malware is bad for your health

otool

XCODE

class-dump

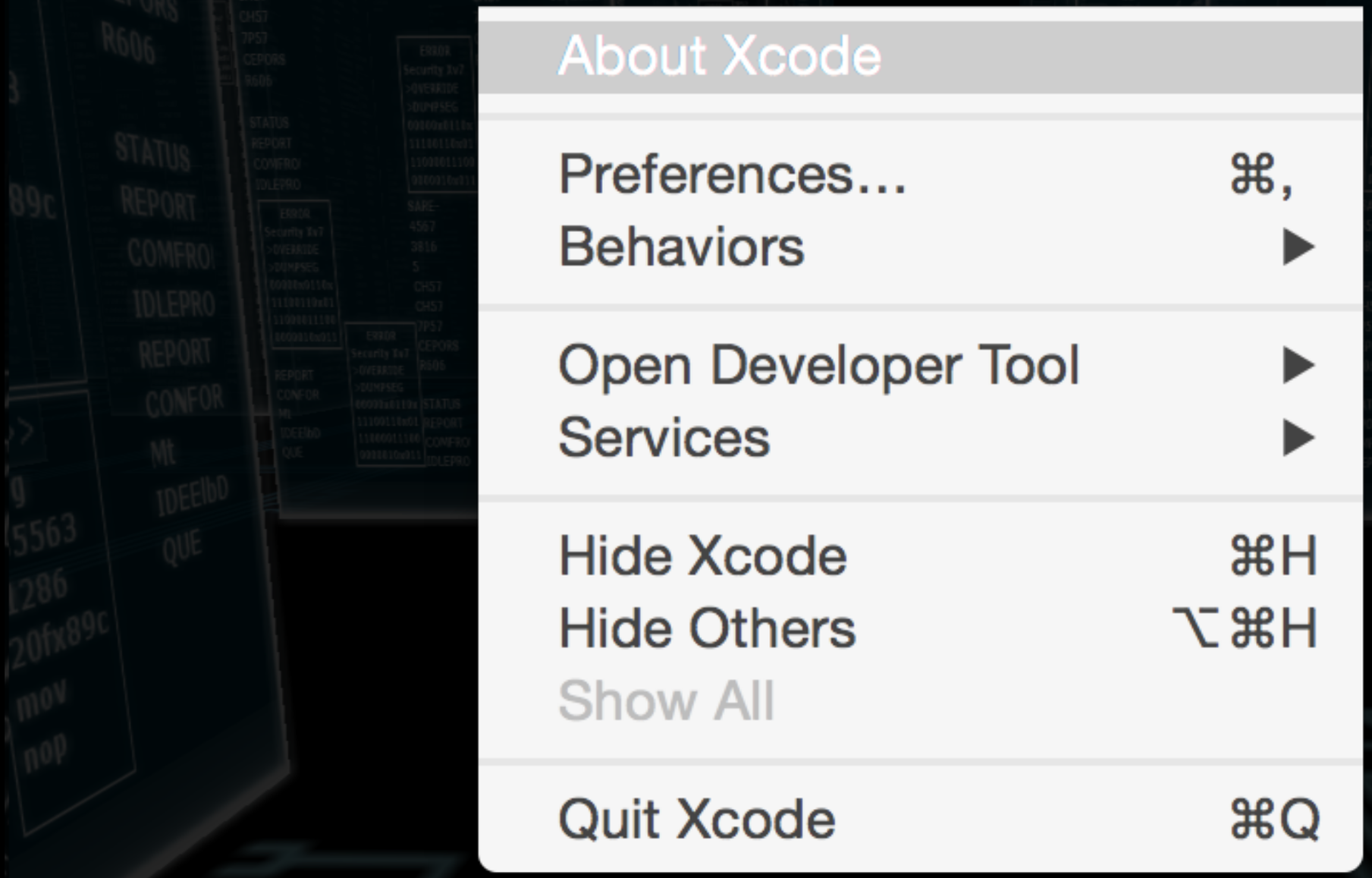
http://stevenygard.com/projects/class-dump/

strings

xxd

HexFiend

http://ridiculousfish.com/hexfiend/



http://ridiculousfish.com/hexfiend/

Peter Ammon (HexFiend)
Copyright © 2005-2009, Peter Ammon. All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

http://ridiculousfish.com/hexfiend/

lldb

XCODE

Hopper

http://hopperapp.com

csfde

createmobileaccount

Digging in the Dirt

Edward Marczak

marczak@radiotope.com

@marczak