Advanced Operating System and Virtualization

Disassember1

Hiroaki Fukuda

Contents

- Understand the binary of 1.s
- Understand specifications of 8086
  - mod
  - r/m
  - reg
  - d
  - effective address

Binary of 1.s

Read a.out using hexdump

```
pine:asem hiroaki$ hexdump -C a.out
00000000  01 03 20 04 20 00 00 00  10 00 00 00 26 00 00 00  |.. . .......&...|
00000010  00 00 00 00 00 00 00 00  00 00 01 00 70 00 00 00  |............p...|
00000020  bb 00 00 cd 20 bb 10 00  cd 20 00 00 00 00 00 00  |.... .... ......|
00000030  01 00 04 00 01 00 06 00  00 00 20 00 00 00 00 00  |.......... .....|
00000040  01 00 01 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000050  68 65 6c 6c 6f 0a 6d 65  73 73 61 67 65 00 00 00  |hello.message...|
00000060  00 00 03 00 00 00 65 78  69 74 00 00 00 00 10 00  |......exit......|
00000070  00 00 03 00 00 00 68 65  6c 6c 6f 00 00 00 20 00  |......hello... .|
00000080  00 00 03 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000090  00 00 02 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000a0  00 00 03 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000b0  00 00 03 00 00 00 00 00  00 00 00 00 00 00 26 00  |..............&.|
000000c0  00 00 04 00 00 00                                 |......|
000000c6
```

Disassemble

```
pine:asem hiroaki$ mmvm -d a.out
0000: bb0000        mov bx, 0000
0003: cd20          int 20
0005: bb1000        mov bx, 0010
0008: cd20          int 20
000a: 0000          add [bx+si], al
000c: 0000          add [bx+si], al
000e: 0000          add [bx+si], al
```

Focus on “bb0000” → mov bx, 0000

Copy “0000” value to bx register

Using Instruction Set Summary

DATA TRANSFER

MOV = Move:

Immediate to Register

1 0 1 1 w reg	data	data if w = 1
bb	00	00

Mov reg, immediate

How to know reg and immediate? 10111011

W = 1

Reg = 011

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

mov bx, 0000

Reg = bx

Data = 0x0000

Compile 1.c and try first 3 instructions

```
pine:tests hiroaki$ /usr/local/core/bin/m2cc 1.c
"1.c", line 1: (warning) 'main' old-fashioned function definition
"1.c", line 2: (warning) implicit declaration of function write
pine:tests hiroaki$ mmvm -d a.out 2>&1 | head -n 5
0000: 31ed          xor bp, bp
0002: 89e3          mov bx, sp
0004: 8b07          mov ax, [bx]
0006: 8d5702        lea dx, [bx+2]
0009: 8d4f04        lea cx, [bx+4]
```

mod : reg : r/m : d

- Mod
  - 2 bit information to decide displacement(DISP)
- Reg
  - 3 bit information to decide register
- r/m
  - 3 bit information to decide effective address (EA)
- d
  - 1 bit information to decide the direction (from/to)

Let's try 31ed

XOR = Exclusive or:

Reg./Memory and Register to Either

0 0 1 1 0 0 d w	mod reg r/m

00110001 11101101

d = 0 w = 1 mod = 11 reg = 101 r/m = 101

if d = 1 then "to" reg; if d = 0 then "from" reg

if mod = 11 then r/m is treated as a REG field

XOR r/m, reg

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

XOR bp, bp

Let's try by hand

- 89e3
  - Why mov bx, sp

Let's try 8b07

MOV = Move:

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Register/Memory to/from Register	1 0 0 0 1 0 d w	mod reg r/m

10001011 00000111

D = 1 w = 1 mod = 00 reg = 000 r/m = 111

if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 10 then DISP = disp-high; disp-low
if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP

mov reg, r/m

mov ax, [bx]

Let's try 1.c

- Implement your own disassembler which can analyze the binary of 1.c
- Compare the result to the outout of mmvm -d