המרכז
לחינוך סייבר
CYBER EDUCATION CENTER

Assembly for Reverse Engineering

Bit Operations & Arithmetic Instructions

Barak Gonen

Bit Operation

המרכז לחינוך סייבר
CYBER EDUCATION CENTER

- AND
- OR
- NOT
- XOR
- NEG
- SHR
- SHL

AND

המרכז
לחינוך סייבר
CYBER EDUCATION CENTER

```
mov     eax, 0xFFFFFF07
mov     ebx, 0x00000005
and     eax, ebx

call    print_eax
```

Riddle

המרכז
לחינוך סייבר
CYBER EDUCATION CENTER

How can we tell if a number is even / odd?

… Can be divided by 4 without reminder?

OR

המרכז
לחינוך סייבר
CYBER EDUCATION CENTER

```
mov     eax, 0xFFFFFF07
mov     ebx, 0x000000FF
or      eax, ebx

call    print_eax
```

Riddle

המרכז
לחינוך סייבר
CYBER EDUCATION CENTER

How can we set the leftmost bit in EAX to 1?

NOT

המרכז
לחינוך סייבר
CYBER EDUCATION CENTER

```
mov     eax, 0xFFFFFF07
not     eax

call    print_eax
```

XOR

המרכז לחינוך סייבר
CYBER EDUCATION CENTER

```
mov     eax, 0xFFFFFF07
mov     ebx, 0x000000FF
xor     eax, ebx

call    print_eax
```

NEG

Find out on your own ☺

SHL

המרכז
לחינוך סייבר
CYBER EDUCATION CENTER

```
call    read_hex
shl     eax, 4

call    print_eax
```

SHR

המרכז
לחינוך סייבר
CYBER EDUCATION CENTER

```
call    read_hex
shr     eax, 1

call    print_eax
```

Arithmetic Instructions

המרכז לחינוך סייבר
CYBER EDUCATION CENTER

- INC
- DEC
- ADD
- SUB
- MUL / IMUL
- DIV / IDIV
- LEA (not exactly arithmetic instruction, but useful)

MUL

המרכז לחינוך סייבר
CYBER EDUCATION CENTER

Takes only 1 operand – the other is always EAX*

Result is stored in EAX

```
call    read_hex
mov     ebx, eax
call    read_hex
mul     ebx
call    print_eax
```

* When 32 bit reg. is used

MUL – cont.

המרכז לחינוך סייבר
CYBER EDUCATION CENTER

Isn’t the output unexpected?

```
call    read_hex
mov     ebx, eax
call    read_hex
mov     edx, 0xFFFFFFFF
mul     ebx
call    print_eax
mov     eax, edx
call    print_eax
```

MUL – cont.

המרכז לחינוך סייבר
CYBER EDUCATION CENTER

```
mul     bl      ; 8 bit reg * al. Result -> ax
mul     bx      ; 16 bit reg * ax. Result -> eax
mul     ebx     ; 32 bit reg * eax. Result -> edx:eax
```

Think

המרכז לחינוך סייבר
CYBER EDUCATION CENTER

How much is 00000011b times 11100001b?

- 00000011b is 3
- 10000001b is either 225 or -31
- So what is the result?

MUL - IMUL

המרכז לחינוך סייבר
CYBER EDUCATION CENTER

```
xor     eax, eax
mov     al, 00000011b
mov     bl, 11100001b
mul     bl
call    print_eax

xor     eax, eax
mov     al, 00000011b
mov     bl, 11100001b
imul    bl
call    print_eax
```

DIV

המרכז
לחינוך סייבר
CYBER EDUCATION CENTER

```
call    read_hex
mov     ebx, eax
call    read_hex
xor     eax, ebx
xor     ebx, eax
xor     eax, ebx
div     bl
call    print_eax
```

DIV - IDIV

המרכז לחינוך סייבר
CYBER EDUCATION CENTER

```
xor     eax, eax
mov     ax, 0x08CA
mov     bl, 11100001b
div     bl
call    print_eax

xor     eax, eax
mov     ax, 0x00C8
mov     bl, 11111110b
idiv    bl
call    print_eax
```

LEA

המרכז לחינוך סייבר
CYBER EDUCATION CENTER

Load Effective Address

Actually enables several calculations in one instruction

Calculates equations of the form “ register*a + register + b “, where a = 1, 2, 4, or 8

```
mov     ebx, 5
mov     ecx, 1
lea     eax, [ebx*8+ecx+5]
call    print_eax
```