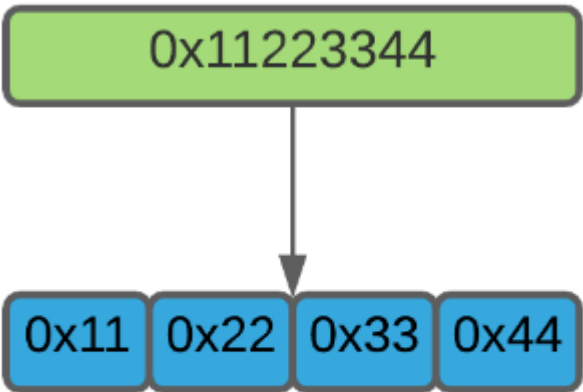


x64

Shellcode

CPU CPU:1 C

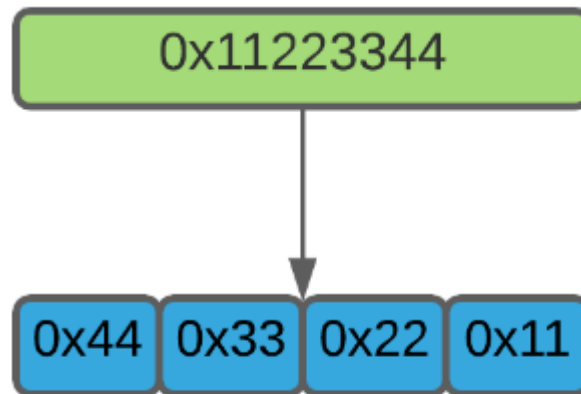
()



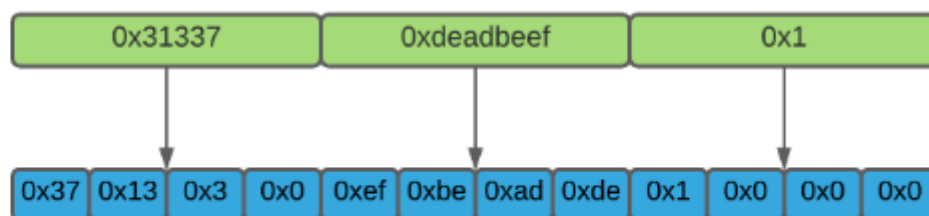
Memory Address: 0x0 0x1 0x2 0x3

0x11223344 4 0x11 0x22 0x33 0x44

0x11223344 4 0x11 0x22 0x33 0x44



Memory Address: `0x0` `0x1` `0x2` `0x3`



Memory Address: `0x0` `0x1` `0x2` `0x3` `0x4` `0x5` `0x6` `0x7` `0x8` `0x9` `0xa` `0xb`

$2^{64} - 1$ -2^{63} $2^{63} - 1$ `0`

`1` `42` `2`

```

42      0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0010 1010
      1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1101 0101
1:      1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1101 0110

```

CPU

(RAM) CPU **64** **8**

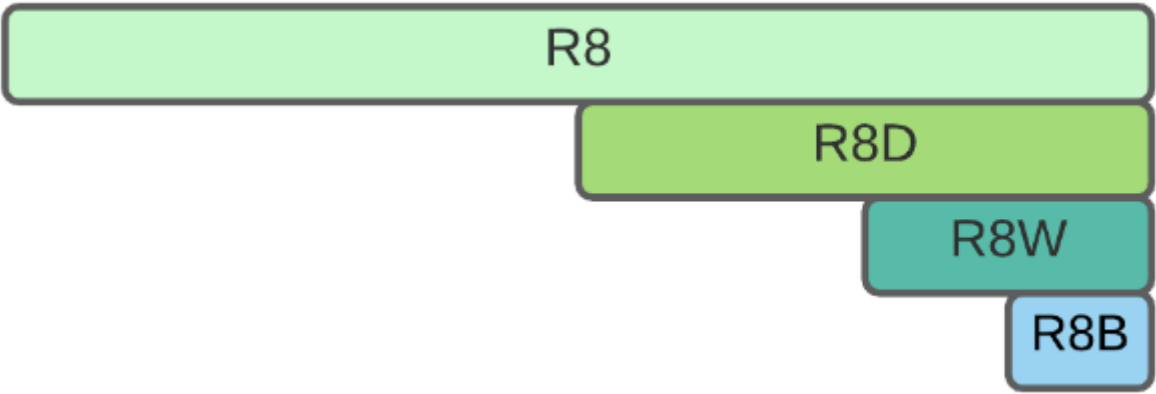
64 x64

RIP		
RAX	I/O	Windows syscall SSN
RBX		
RCX		
RDX	I/O	
RSI		
RDI		
RBP		
RSP		
R8-R15		
RFLAGS	FLAG	

x64
8
RAX
4
EAX
EAX
2
AX
AH
AL
A



R8



32 EAX 32 0 8 16

64 ???? ?	? 32 ?	? 16 ?	? 8 ?
rax	eax	ax	al
rbx	ebx	bx	bl
rcx	ecx	cx	cl
rdx	edx	dx	dl
rsi	esi	si	sil
rdi	edi	di	dil
rbp	ebp	bp	bpl
rsp	esp	sp	spl
r8	r8d	r8w	r8b
r9	r9d	r9w	r9b

64 ???? ?	32 ?	16 ?	8 ?
r10	r10d	r10w	r10b
r11	r11d	r11w	r11b
r12	r12d	r12w	r12b
r13	r13d	r13w	r13b
r14	r14d	r14w	r14b
r15	r15d	r15w	r15b

RFLAGS

0	CF	
2	PF	
6	ZF	
7	SF	
11	OF	

(LIFO)

x64 Windows

RCX RDX R8 R9

RAX R10 R11

RCX RDX R8 R9 R10 R11

RBX RBP RSI RSP R12 R13 R14 R15

RCX RDX R8 R9 R10 R11

x86_64 fastcall PUSH/POP

RSP

)

(RIP

(RSP)

RSP

RSP

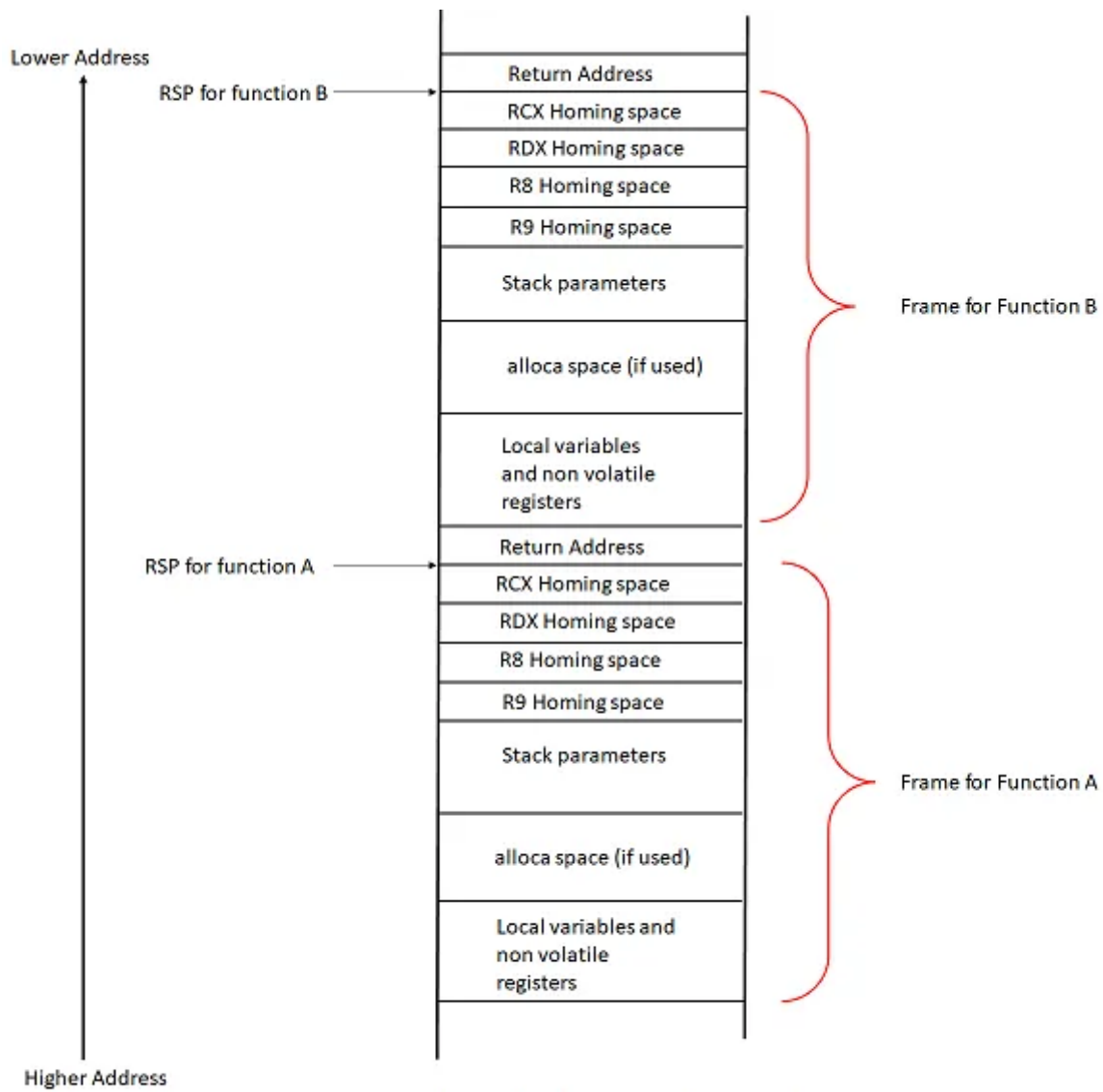
(RBP)

x64

RSP

RSP

A B



BYTE	1
WORD	2
DWORD	4
QWORD	8

Shellcode

MOV

MOV

```
MOV RAX, 1 // 1 RAX
MOV [RAX], 3 // 3 RAX
MOV RAX, RCX // RCX RAX
MOV [RDI], RAX // RAX RDI
MOV [RAX], RAX // RAX RAX
MOV RBX, [RDI + 0x10] // RDI 10 RBX
```

LEA

LEA MOV

```
LEA RBX, [RCX + 0x10] // RCX 10 RBX
MOV RBX, [RCX + 0x10] // RCX 10 RBX
LEA RAX, [RCX + 2*RAX + 0x10] // RCX + 2*EAX + 10 RAX
```

PUSH/POP

PUSH POP x64 x64 PUSH POP

```
PUSH RAX // RAX
PUSH 1 // 1
POP RAX // RAX
```

INC/DEC/ADD/SUB/MUL/DIV


```
INC RAX    // RAX  1
INC BYTE [RAX]    // RAX          1
ADD RAX, RAX    // RAX = RAX + RAX
ADD RCX, 4      // RCX = RCX + 4
ADD DWORD [RSP], RAX    // memory[RSP] = memory[RSP] + RAX
SUB RAX, RDX    // RAX = RAX - RDX
SUB RBX, 0x10    // RBX = RBX - 0x10
MUL RCX    // RDX: RAX = RAX * RCX
MUL DWORD [RDX]    // RDX: RAX = RAX * memory[RDX]
DIV RCX    // RDX: RAX/RCX=RAX···RDX
```

NEG 0x00

```
NEG RAX // RAX = -RAX
```

```
0 AND 0 = 0
0 AND 1 = 0
1 AND 0 = 0
1 AND 1 = 1

0 OR 0 = 0
0 OR 1 = 1
1 OR 0 = 1
1 OR 1 = 1
```

$$1 \text{ XOR } 1 = 0$$

NOT 1 = 0

NOT **ADD RAX, RCX**

[illegible]

```
AND RCX, 0x11 ; RCX = RCX and 0x11
```

CALL RET

jxx

JNE/JNZ / 0

JA/JNBE	/	/
JAE/JNB	/	
JB/JNAE	/	
JBE/JNA	/	

JE/JZ	/	0
JNE/JNZ		0
JG/JNLE	/	
JGE/JNL	/	
JL/JNGE	/	
JLE/JNG	/	

SAL/SAR/SHL/SHR

	SAL	SAR	1	2^n
SHL	SAL	SHR	SAR	() SAR
	2			

RAX 0xFF SHL RAX, 3

RAX
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1111 1111

SHL RAX, 3
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0111 1111 1000

SHL RAX, 56

1111 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

SHL/SHR SAL/SAR

ROL/ROR

ROL/ROR “ ”

```

┌───────────┐
1 0 0 0 1 0 0 0 |
  / / / / / / / |
0 0 0 1 0 0 0 1 └─┘

```

STOS	BYTE	WORD	DWORD	QWORD	RDI	1	RDI	1	QWC
STOS REP	CRCX	STOSRCX	1	0	REP STOS	memset			
SCAS	(AL AX EAX RAX)					DF	RDI	SCAS	RI

Revision #27
Created 1 May 2023 13:41:06 by
Updated 28 January 2024 05:09:29 by