**Assignment 5 – Question 1**

Table 1: Setting %rax, or %eax, return val to 0

|  |  |
| --- | --- |
| **x86-64 instruction**  **(represented here as an assembly instruction as opposed to a machine instruction)** | **The size (in bytes) of the corresponding x86-64 machine instruction** |
| xorq %rax, %rax | 3 bytes |
| xorl %eax, %eax | **2 bytes (memory efficient)** |
| movq $0, %rax | 7 bytes |
| movl $0, %eax | 5 bytes |
| subl %eax, %eax | **2 bytes (memory efficient)** |
| imull $0, %eax | 3 bytes |
| andl $0, %eax | 3 bytes |

Table 2: incrementing %eax, return val by 1

|  |  |
| --- | --- |
| **x86-64 instruction**  **(represented here as an assembly instruction as opposed to a machine instruction)** | **The size (in bytes) of the corresponding x86-64 machine instruction** |
| addl $1, %eax | 3 bytes |
| leal 1(%eax), %eax | 4 bytes |
| incl %eax | **2 bytes (memory efficient)** |
| subl $-1, %eax | 3 bytes |

Table 3: Adding 8 to %rax, (return val)

|  |  |
| --- | --- |
| **x86-64 instruction**  **(represented here as an assembly instruction as opposed to a machine instruction)** | **The size (in bytes) of the corresponding x86-64 machine instruction** |
| addl $8, %eax | **3 bytes (memory efficient)** |
| leal 8(%eax), %eax | 4 bytes |

Table 4

|  |  |
| --- | --- |
| **x86-64 instruction**  **(represented here as an assembly instruction as opposed to a machine instruction)** | **The size (in bytes) of the corresponding x86-64 machine instruction** |
| subq $8, %rsp | 4 bytes |
| movq %rdi, (%rsp) | 4 bytes |
| pushq %rdi | **1 byte (memory efficient)** |

Table 5

|  |  |
| --- | --- |
| **x86-64 instruction**  **(represented here as an assembly instruction as opposed to a machine instruction)** | **The size (in bytes) of the corresponding x86-64 machine instruction** |
| movq (%rsp), %rsi | 4 bytes |
| addq $8, %rsp | 4 bytes |
| popq %rsi | **1 byte (memory efficient)** |

**Assembly Code**

xorq %rax, %rax

xorl %eax, %eax

movq $0, %rax

movl $0, %eax

subl %eax, %eax

imull $0, %eax

andl $0, %eax

addl $1, %eax

leal 1(%eax), %eax

incl %eax

subl $-1, %eax

addl $8, %eax

leal 8(%eax), %eax

subq $8, %rsp

movq %rdi, (%rsp)

pushq %rdi

movq (%rsp), %rsi

addq $8, %rsp

popq %rsi

**Disassembled code**

main.o: file format elf64-x86-64

Disassembly of section .text:

0000000000000000 <.text>:

0: 48 31 c0 xor %rax,%rax

3: 31 c0 xor %eax,%eax

5: 48 c7 c0 00 00 00 00 mov $0x0,%rax

c: b8 00 00 00 00 mov $0x0,%eax

11: 29 c0 sub %eax,%eax

13: 6b c0 00 imul $0x0,%eax,%eax

16: 83 e0 00 and $0x0,%eax

19: 83 c0 01 add $0x1,%eax

1c: 67 8d 40 01 lea 0x1(%eax),%eax

20: ff c0 inc %eax

22: 83 e8 ff sub $0xffffffff,%eax

25: 83 c0 08 add $0x8,%eax

28: 67 8d 40 08 lea 0x8(%eax),%eax

2c: 48 83 ec 08 sub $0x8,%rsp

30: 48 89 3c 24 mov %rdi,(%rsp)

34: 57 push %rdi

35: 48 8b 34 24 mov (%rsp),%rsi

39: 48 83 c4 08 add $0x8,%rsp

3d: 5e pop %rsi

Disassembly of section .debug\_line:

0000000000000000 <.debug\_line>:

0: 55 push %rbp

1: 00 00 add %al,(%rax)

3: 00 05 00 08 00 2a add %al,0x2a000800(%rip) # 0x2a000809

9: 00 00 add %al,(%rax)

b: 00 01 add %al,(%rcx)

d: 01 01 add %eax,(%rcx)

f: fb sti

10: 0e (bad)

11: 0d 00 01 01 01 or $0x1010100,%eax

16: 01 00 add %eax,(%rax)

18: 00 00 add %al,(%rax)

1a: 01 00 add %eax,(%rax)

1c: 00 01 add %al,(%rcx)

1e: 01 01 add %eax,(%rcx)

20: 1f (bad)

21: 01 00 add %eax,(%rax)

23: 00 00 add %al,(%rax)

25: 00 02 add %al,(%rdx)

27: 01 1f add %ebx,(%rdi)

29: 02 0f add (%rdi),%cl

2b: 02 00 add (%rax),%al

...

35: 00 00 add %al,(%rax)

37: 09 02 or %eax,(%rdx)

...

41: 01 3d 2f 75 59 2f add %edi,0x2f59752f(%rip) # 0x2f597576

47: 3d 3f 3d 4b 2f cmp $0x2f4b3d3f,%eax

4c: 3e 3d 4c 4b 4b 22 ds cmp $0x224b4b4c,%eax

52: 4b rex.WXB

53: 4b 02 01 rex.WXB add (%r9),%al

56: 00 01 add %al,(%rcx)

58: 01 .byte 0x1

Disassembly of section .debug\_line\_str:

0000000000000000 <.debug\_line\_str>:

0: 2f (bad)

1: 68 6f 6d 65 2f push $0x2f656d6f

6: 73 68 jae 0x70

8: 61 (bad)

9: 33 34 30 xor (%rax,%rsi,1),%esi

c: 2f (bad)

d: 73 66 jae 0x75

f: 75 68 jne 0x79

11: 6f outsl %ds:(%rsi),(%dx)

12: 6d insl (%dx),%es:(%rdi)

13: 65 2f gs (bad)

15: 43 rex.XB

16: 4d 50 rex.WRB push %r8

18: 54 push %rsp

19: 32 39 xor (%rcx),%bh

1b: 35 2f 41 73 73 xor $0x7373412f,%eax

20: 69 67 6e 6d 65 6e 74 imul $0x746e656d,0x6e(%rdi),%esp

27: 73 2f jae 0x58

29: 41 73 73 rex.B jae 0x9f

2c: 6e outsb %ds:(%rsi),(%dx)

2d: 35 2d 66 69 6c xor $0x6c69662d,%eax

32: 65 73 00 gs jae 0x35

35: 6d insl (%dx),%es:(%rdi)

36: 61 (bad)

37: 69 6e 2e 73 00 6d 61 imul $0x616d0073,0x2e(%rsi),%ebp

3e: 69 .byte 0x69

3f: 6e outsb %ds:(%rsi),(%dx)

40: 2e 73 00 jae,pn 0x43

Disassembly of section .debug\_info:

0000000000000000 <.debug\_info>:

0: 24 00 and $0x0,%al

2: 00 00 add %al,(%rax)

4: 05 00 01 08 00 add $0x80100,%eax

9: 00 00 add %al,(%rax)

b: 00 01 add %al,(%rcx)

...

19: 3e 00 00 ds add %al,(%rax)

...

24: 00 00 add %al,(%rax)

26: 01 .byte 0x1

27: 80 .byte 0x80

Disassembly of section .debug\_abbrev:

0000000000000000 <.debug\_abbrev>:

0: 01 11 add %edx,(%rcx)

2: 00 10 add %dl,(%rax)

4: 17 (bad)

5: 11 01 adc %eax,(%rcx)

7: 12 0f adc (%rdi),%cl

9: 03 0e add (%rsi),%ecx

b: 1b 0e sbb (%rsi),%ecx

d: 25 0e 13 05 00 and $0x5130e,%eax

...

Disassembly of section .debug\_aranges:

0000000000000000 <.debug\_aranges>:

0: 2c 00 sub $0x0,%al

2: 00 00 add %al,(%rax)

4: 02 00 add (%rax),%al

6: 00 00 add %al,(%rax)

8: 00 00 add %al,(%rax)

a: 08 00 or %al,(%rax)

...

18: 3e 00 00 ds add %al,(%rax)

...

Disassembly of section .debug\_str:

0000000000000000 <.debug\_str>:

0: 6d insl (%dx),%es:(%rdi)

1: 61 (bad)

2: 69 6e 2e 73 00 2f 68 imul $0x682f0073,0x2e(%rsi),%ebp

9: 6f outsl %ds:(%rsi),(%dx)

a: 6d insl (%dx),%es:(%rdi)

b: 65 2f gs (bad)

d: 73 68 jae 0x77

f: 61 (bad)

10: 33 34 30 xor (%rax,%rsi,1),%esi

13: 2f (bad)

14: 73 66 jae 0x7c

16: 75 68 jne 0x80

18: 6f outsl %ds:(%rsi),(%dx)

19: 6d insl (%dx),%es:(%rdi)

1a: 65 2f gs (bad)

1c: 43 rex.XB

1d: 4d 50 rex.WRB push %r8

1f: 54 push %rsp

20: 32 39 xor (%rcx),%bh

22: 35 2f 41 73 73 xor $0x7373412f,%eax

27: 69 67 6e 6d 65 6e 74 imul $0x746e656d,0x6e(%rdi),%esp

2e: 73 2f jae 0x5f

30: 41 73 73 rex.B jae 0xa6

33: 6e outsb %ds:(%rsi),(%dx)

34: 35 2d 66 69 6c xor $0x6c69662d,%eax

39: 65 73 00 gs jae 0x3c

3c: 47 rex.RXB

3d: 4e 55 rex.WRX push %rbp

3f: 20 41 53 and %al,0x53(%rcx)

42: 20 32 and %dh,(%rdx)

44: 2e 33 38 cs xor (%rax),%edi

...