Assembly Language Lecture Series:

Basic x86-64 instructions

Sensei RL Uy, College of Computer Studies,
De La Salle University, Manila, Philippines

Copyright Notice

64-bit number range (**unsigned**)

	Largest positive number (hex)	Largest positive (decimal)
8-bit	0xFF	255
16-bit	0xFFFF	65535
32-bit	0xFFFF_FFFF	4294967295
64-bit	0xFFFF_FFFF_FFFF_FFFF	18446744073709551615 (18.x * 10^{18})

64-bit number range (signed)

	Largest positive number (hex)	Largest positive (decimal)	Largest magnitude negative (hex)	Largest magnitude negative (decimal)
8-bit	0x7F	127	0x80	-128
16-bit	0x7FFF	32767	0x8000	-32768
32-bit	0x7FFF_FFFF	2147483647	0x8000_0000	-2147483648

	Largest positive number (hex)	Largest positive (decimal)
64-bit	0x7FFF_FFFF_FFFF_FFFF	9223372036854775807 (9.x * 10^{18})

	Largest magnitude negative (hex)	Largest magnitude negative (decimal)
64-bit	0x8000_0000_0000_0000	-9223372036854775808 (-9.x * 10^{18})

Basic x86-64 instructions

1. **MOV**

   Move instruction
2. **ADD**

   Addition instruction
3. **INC**

   Increment Instruction
4. **LEA**

   Load Effective Address/”ptr”
5. **NOP**

   No Operation Instruction

x86-64 Data Transfer Instructions: MOV

MOV (move instruction)

Syntax: MOV dst, src
dst ←src
dst: reg/mem
src: reg/mem/imm

Flags affected:
*none

Note: If **dst==mem64** and **src==imm, Immediate value** up to **sign-extended 32-bit** only.

x86-64 Data Transfer Instructions: MOV

MOV (move instruction)

Syntax: MOV dst, src

dst ← src

dst: reg/mem

src: reg/mem/imm

Flags affected:

*none

Example:

```
section .text
MOV RAX, 0x1357_1234_ABCD_0000
MOV RBX, 0xABCD_EF12_3456_789A
MOV AX, BX
```

What will RAX contain after execution?

x86-64 Data Transfer Instructions: MOV

MOV (move instruction)

Syntax: MOV dst, src

dst ← src

dst: reg/mem

src: reg/mem/imm

Flags affected:

*none

Example:

```
section .text
MOV RAX, 0x1357_1234_ABCD_0000
MOV RBX, 0xABCD_EF12_3456_789A
MOV AX, BX
```

What will RAX contain after execution?

RAX = 13571234ABCD789A

For readability: 1357_1234_ABCD_789A

x86-64 Data Transfer Instructions: MOV

MOV (move instruction)

Syntax: MOV dst, src
dst ← src
dst: reg/mem
src: reg/mem/imm

Flags affected:
*none

Example: set RAX to -1

- `MOV RAX, -1`
- `MOV RAX, 0xFF`
- `MOV RAX, 0xFFFF`
- `MOV RAX, 0xFFFF_FFFF`
- `MOV RAX, 0xFFFF_FFFF_FFFF_FFFF`

x86-64 Data Transfer Instructions: MOV

MOV (move instruction)

Syntax: MOV dst, src
dst ← src
dst: reg/mem
src: reg/mem/imm

Flags affected:
*none

Example: set RAX to +10

- `MOV RAX, 10`
- `MOV RAX, 0x0A`
- `MOV RAX, 0x000A`
- `MOV RAX, 0x0000_0000A`
- `MOV RAX, 0x0000_0000_0000_000A`

x86-64 Data Transfer Instructions: MOV

MOV (move instruction)

Syntax: MOV dst, src

dst ← src

dst: reg/mem

src: reg/mem/imm

Flags affected:

*none

Example: set RAX to max value

- `MOV RAX, 0x7FFF_FFFF_FFFF_FFFF; pos`
- `MOV RAX, 0x8000_0000_0000_0000; neg`

x86-64 Data Transfer Instructions: MOV

MOV (move instruction)

Syntax: MOV dst, src
dst ← src
dst: reg/mem
src: reg/mem/imm

Flags affected:
*none

Example: set 64-bit memory var1 to max value

- `MOV qword [var1],0x0000_0000_7fff_ffff ; pos`
- `MOV qword [var1], 2147483647 ; pos`
- `MOV qword [var1], 0xffff_ffff_8000_0000 ; neg`
- `MOV qword [var1], -2147483648 ; neg`

x86-64 Data Transfer Instructions: MOV

MOV (move instruction)

Syntax: MOV dst, src
dst ← src
dst: reg/mem
src: reg/mem/imm

Flags affected:
*none

Example: set 64-bit memory var1 to max value

- `MOV qword [var1],0x0000_0000_7fff_ffff ; pos`
- `MOV qword [var1], 2147483647 ; pos`
- `MOV qword [var1], 0xffff_ffff_8000_0000 ; neg`
- `MOV qword [var1], -2147483648 ; neg`

x86-64 Arithmetic Instructions: ADD

ADD (addition instruction)

Syntax: ADD dst, src

dst ← dst + src

dst: reg/mem

src: reg/mem/imm8_16_32

Flags affected:

*all status flags

Note:

1. Immediate value up to **32-bit** only
2. When an **immediate value** is used as an **operand**, it is **sign-extended** to the **length** of the **destination** operand format
3. **Negative** number in **hex** has to be sign-extended to **64-bit**

x86-64 Arithmetic Instructions: ADD

ADD (addition instruction)

Syntax: ADD dst, src

dst ← dst + src

dst: reg/mem

src: reg/mem/imm8_16_32

Flags affected:

*all status flags

Example:

```
section .data
var1 dq 0x7FFF_FFFF_FFFF_FFFE
section .text
MOV RAX, 0x01
ADD RAX, [var1]
```

1. **What will RAX contain after execution?**
2. **What will be the value of the status flags after execution?**

x86-64 Arithmetic Instructions: ADD

ADD (addition instruction)

Syntax: ADD dst, src

dst ← dst + src

dst: reg/mem

src: reg/mem/imm8_16_32

Flags affected:

*all status flags

Example:

```
section .data
var1 dq 0x7FFF_FFFF_FFFF_FFFE
section .text
MOV RAX, 0x01
ADD RAX, [var1]
```

1. What will RAX contain after execution?
2. What will be the value of the status flags after execution?

RAX = 7FFFFFFFFFFFFFFF

CF = 0

SF = 0

ZF = 0

OF = 0

PF = 1

AF = 0

For readability:

7FFF_FFFF_FFFF_FFFF

x86-64 Arithmetic Instructions: ADD

ADD (addition instruction)

Syntax: ADD dst, src

dst ← dst + src

dst: reg/mem

src: reg/mem/imm8_16_32

Flags affected:

*all status flags

Example: add RAX with -1

- `ADD RAX, -1`
- `ADD RAX, 0xFF`
- `ADD RAX, 0xFFFF`
- `ADD RAX, 0xFFFF_FFFF`
- `ADD RAX, 0xFFFF_FFFF_FFFF_FFFF`

x86-64 Arithmetic Instructions: ADD

ADD (addition instruction)

Syntax: ADD dst, src

dst ← dst + src

dst: reg/mem

src: reg/mem/imm8_16_32

Flags affected:

*all status flags

Example: add RAX with +10

- `ADD RAX, 10`
- `ADD RAX, 0x0A`
- `ADD RAX, 0x000A`
- `ADD RAX, 0x0000_0000A`
- `ADD RAX, 0x0000_0000_0000_000A`

x86-64 Arithmetic Instructions: ADD

ADD (addition instruction)

Syntax: ADD dst, src

dst ← dst + src

dst: reg/mem

src: reg/mem/imm8_16_32

Flags affected:

*all status flags

Example: add RAX with max value

- `ADD RAX, 0x0000_0000_7fff_ffff ; pos`
- `ADD RAX, 2147483647 ; pos`
- `ADD RAX, 0xffff_ffff_8000_0000 ; neg`
- `ADD RAX, -2147483648 ; neg`

x86-64 Arithmetic Instructions: INC

INC (increment instruction)

Syntax: INC dst

dst ← dst + 1

dst: reg/mem

Flags affected:

*SF, ZF, OF, PF, AF

CF – not affected

x86-64 Arithmetic Instructions: INC

INC (increment instruction)

Syntax: INC dst

dst ← dst + 1

dst: reg/mem

Flags affected:

*SF, ZF, OF, PF, AF

CF – not affected

Example:

```
section .data
var1 dq 0x7FFF_FFFF_FFFF_FFFD
section .text
INC qword[var1]
```

1. **What will memloc var1 contain after execution?**
2. **What will SF, ZF, OF, PF, AF after execution?**

x86-64 Arithmetic Instructions: INC

INC (increment instruction)

Syntax: INC dst

dst ← dst + 1

dst: reg/mem

Flags affected:

*SF, ZF, OF, PF, AF

CF – not affected

Example:

```
section .data
var1 dq 0x7FFF_FFFF_FFFF_FFFD
section .text
INC qword[var1]
```

1. What will memloc var1 contain after execution?
2. What will SF, ZF, OF, PF, AF after execution?

var1 = 7FFF_FFFF_FFFF_FFFE

SF = 0

ZF = 0

OF = 0

PF = 0

AF = 0

x86-64 Data Transfer Instructions: LEA

LEA
(Load Effective Address/”ptr”)

Syntax: LEA dst, src

dst ← effective_address(src)

dst: reg16/reg32/reg64

src: mem

Flags affected:

*none

x86-64 Data Transfer Instructions: LEA

LEA
(Load Effective Address/"ptr")

Syntax: LEA dst, src
dst ← effective_address(src)
dst: reg16/reg32/reg64
src: mem

Flags affected:
*none

Example:

```
section .data
VARX db
0x12,0x34,0x56,0x78,0x9A,0xBC,0xDE,0xF0

section .text
LEA RSI, [VARX]
MOV RBX, [RSI]
```

What will RSI contain after execution?
What will RBX contain after execution?

label	address	Memory data (byte)
	403017	F0
	403016	DE
	403015	BC
	403014	9A
	403013	78
	403012	56
	403011	34
VARX	403010	12

x86-64 Data Transfer Instructions: LEA

LEA
(Load Effective Address/”ptr”)

Syntax: LEA dst, src
dst ← effective_address(src)
dst: reg16/reg32/reg64
src: mem

Flags affected:
*none

Example:

```
section .data
VARX db
0x12,0x34,0x56,0x78,0x9A,0xBC,0xDE,0xF0

section .text
LEA RSI, [VARX]
MOV RBX, [RSI]
```

What will RSI contain after execution?
What will RBX contain after execution?

RSI = 0000000000403010
RBX = F0DEBC9A78563412

label	address	Memory data (byte)
	403017	F0
	403016	DE
	403015	BC
	403014	9A
	403013	78
	403012	56
	403011	34
VARX	403010	12

x86-64 Data Transfer Instructions: NOP

NOP
(No Operation Instruction)

Syntax: NOP

< do nothing>

Flags affected:

*none

x86-64 Data Transfer Instructions: NOP

NOP
(No Operation Instruction)

Syntax: NOP

< do nothing>

Flags affected:

*none

Example:

```
section .text
MOV RAX, 0xFFFF_FFFF_FFFF_FFFF
NOP
```

1. **What will RAX contain after execution?**

x86-64 Data Transfer Instructions: NOP

NOP
(No Operation Instruction)

Syntax: NOP

< do nothing>

Flags affected:

*none

Example:

```
section .text
MOV RAX, 0xFFFF_FFFF_FFFF_FFFF
NOP
```

1. What will RAX contain after execution?

RAX = FFFF_FFFF_FFFF_FFFF