Assembly Language Lecture Series:

x86-64 Scalar Floating-Point Instructions

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

Copyright Notice

Scalar floating-point instructions (partial)

MOVSS	MOVSD
ADDSS	ADDSD
SUBSS	SUBSD
MULSS	MULSD
DIVSS	DIVSD
RCPSS	
SQRTSS	SQRTSD
RSQRTSS	
MAXSS	MAXSD
MINSS	MINSD
UCOMISS	UCOMISD
CVTSS2SI/CVTSSD2DI	CVTSI2SS/CVTSI2SD

x86-64 Instructions : MOVSS

MOVSS (Move scalar single-precision)

Syntax: MOVSS dst, src

dst ← src

MOVSS xmmR, m32

MOVSS xmmR/m32,xmmR

Flags Affected:

*all status flags no change: if count is 0

x86-64 Instructions : MOVSS

MOVSS (Move scalar single-precision)

Syntax: MOVSS dst, src
dst ← src
MOVSS xmmR, m32
MOVSS xmmR/m32,xmmR

Example:

```
section .data
var1 dd 2.5
section .text
MOVSS XMM1, [var1]
```

1. **What will XMM1 contain after execution?**

x86-64 Instructions : MOVSS

MOVSS (Move scalar single-precision)

Syntax: MOVSS dst, src
dst ← src
MOVSS xmmR, m32
MOVSS xmmR/m32,xmmR

Example:

```
section .data
var1 dd 2.5
section .text
MOVSS XMM1, [var1]
```

1. What will XMM1 contain after execution?

XMM1 = 40200000
or
XMM1 = 2.5

x86-64 Instructions : MOVSD

MOVSD (Move scalar double-precision)

Syntax: MOVSD dst, src

dst ← src

MOVSD xmmR, m64

MOVSD xmmR/m64, xmmR

x86-64 Instructions : MOVSD

MOVSD (Move scalar double-precision)

Syntax: MOVSD dst, src

dst ← src

MOVSD xmmR, m64

MOVSD xmmR/m64, xmmR

Example:

```
section .data
var1 dq 2.5
section .text
MOVSD XMM1, [var1]
```

1. **What will XMM1 contain after execution?**

x86-64 Instructions : MOVSD

MOVSD (Move scalar double-precision)

Syntax: MOVSD dst, src
dst ← src
MOVSD xmmR, m64
MOVSD xmmR/m64, xmmR

Example:

```
section .data
var1 dq 2.5
section .text
MOVSD XMM1, [var1]
```

1. What will XMM1 contain after execution?

XMM1=4004_0000_0000_0000
or
XMM1=2.5

x86-64 Instructions : ADDSS/VADDSS

ADDSS (Add scalar single-precision)

Syntax: ADDSS dst, src
dst ← dst + src
dst:xmmR
src:xmmR/m32

VADDSS (Add scalar single-precision)

Syntax: VADDSS dst, src1, src2
dst ← src1 + src2
dst: xmmR
src1: xmmR
src2: xmmR/m32

x86-64 Instructions : ADDSS/VADDSS

ADDSS (Add scalar single-precision)

Syntax: ADDSS dst, src
dst ← dst + src
dst:xmmR
src:xmmR/m32

VADDSS (Add scalar single-precision)

Syntax: VADDSS dst, src1, src2
dst ← src1 + src2
dst: xmmR
src1: xmmR
src2: xmmR/m32

Example:

```
section .data
var1 dd 2.5
var2 dd 2.75
section .text
MOVSS XMM1, [var1]
MOVSS XMM2, [var2]
VADDSS XMM3,XMM1,XMM2
```

1. **What will XMM3 contain after execution?**

x86-64 Instructions : ADDSS/VADDSS

ADDSS (Add scalar single-precision)

Syntax: ADDSS dst, src
dst ← dst + src
dst:xmmR
src:xmmR/m32

VADDSS (Add scalar single-precision)

Syntax: VADDSS dst, src1, src2
dst ← src1 + src2
dst: xmmR
src1: xmmR
src2: xmmR/m32

Example:

```
section .data
var1 dd 2.5
var2 dd 2.75
section .text
MOVSS XMM1, [var1]
MOVSS XMM2, [var2]
VADDSS XMM3,XMM1,XMM2
```

1. What will XMM3 contain after execution?

XMM3 = 5.25

x86-64 Instructions : ADDSD/VADDSD

ADDSD (Add scalar double-precision)

Syntax: ADDSD dst, src
dst ← dst + src
dst:xmmR
src:xmmR/m64

VADDSD (Add scalar double-precision)

Syntax: VADDSD dst, src1, src2
dst ← src1 + src2
dst: xmmR
src1: xmmR
src2: xmmR/m64

x86-64 Instructions : ADDSD/VADDSD

ADDSD (Add scalar double-precision)

Syntax: ADDSD dst, src
dst ← dst + src
dst:xmmR
src:xmmR/m64

VADDSD (Add scalar double-precision)

Syntax: VADDSD dst, src1, src2
dst ← src1 + src2
dst: xmmR
src1: xmmR
src2: xmmR/m64

Example:

```
section .data
var1 dq 2.5
var2 dq 2.75
section .text
MOVSD XMM1, [var1]
VADDSD XMM2,XMM1,[var2]
```

1. **What will XMM2 contain after execution?**

x86-64 Instructions : ADDSD/VADDSD

ADDSD (Add scalar double-precision)

Syntax: ADDSD dst, src
dst ← dst + src
dst:xmmR
src:xmmR/m64

VADDSD (Add scalar double-precision)

Syntax: VADDSD dst, src1, src2
dst ← src1 + src2
dst: xmmR
src1: xmmR
src2: xmmR/m64

Example:

```
section .data
var1 dq 2.5
var2 dq 2.75
section .text
MOVSD XMM1, [var1]
VADDSD XMM2,XMM1,[var2]
```

1. What will XMM2 contain after execution?

XMM2 = 5.25

x86-64 Instructions : SUBSS/VSUBSS

SUBSS (Subtract scalar single-precision)

Syntax: SUBSS dst, src
dst ← dst - src
dst:xmmR
src:xmmR/m32

VSUBSS (Subtract scalar single-precision)

Syntax: VSUBSS dst, src1, src2
dst ← src1 - src2
dst: xmmR
src1: xmmR
src2: xmmR/m32

x86-64 Instructions : SUBSS/VSUBSS

SUBSS (Subtract scalar single-precision)

Syntax: SUBSS dst, src
dst ← dst - src
dst:xmmR
src:xmmR/m32

VSUBSS (Subtract scalar single-precision)

Syntax: VSUBSS dst, src1, src2
dst ← src1 - src2
dst: xmmR
src1: xmmR
src2: xmmR/m32

Example:

```
section .data
var1 dd 4.5
var2 dd 2.75
section .text
MOVSS XMM1, [var1]
MOVSS XMM2, [var2]
VSUBSS XMM3,XMM1,XMM2
```

1. **What will XMM3 contain after execution?**

x86-64 Instructions : SUBSS/VSUBSS

SUBSS (Subtract scalar single-precision)

Syntax: SUBSS dst, src
dst ← dst - src
dst:xmmR
src:xmmR/m32

VSUBSS (Subtract scalar single-precision)

Syntax: VSUBSS dst, src1, src2
dst ← src1 - src2
dst: xmmR
src1: xmmR
src2: xmmR/m32

Example:

```
section .data
var1 dd 4.5
var2 dd 2.75
section .text
MOVSS XMM1, [var1]
MOVSS XMM2, [var2]
VSUBSS XMM3,XMM1,XMM2
```

1. What will XMM3 contain after execution?

XMM3 = 1.75

x86-64 Instructions : SUBSD/VSUBSD

SUBSD (Subtract scalar double-precision)

Syntax: SUBSD dst, src
dst ← dst - src
dst:xmmR
src:xmmR/m64

VSUBSD (Subtract scalar double-precision)

Syntax: VSUBSD dst, src1, src2
dst ← src1 - src2
dst: xmmR
src1: xmmR
src2: xmmR/m64

x86-64 Instructions : SUBSD/VSUBSD

SUBSD (Subtract scalar double-precision)

Syntax: SUBSD dst, src
dst ← dst - src
dst:xmmR
src:xmmR/m64

VSUBSD (Subtract scalar double-precision)

Syntax: VSUBSD dst, src1, src2
dst ← src1 - src2
dst: xmmR
src1: xmmR
src2: xmmR/m64

Example:

```
section .data
var1 dq 4.5
var2 dq 2.75
section .text
MOVSD XMM1, [var1]
VSUBSD XMM2,XMM1,[var2]
```

1. **What will XMM2 contain after execution?**

x86-64 Instructions : SUBSD/VSUBSD

SUBSD (Subtract scalar double-precision)

Syntax: SUBSD dst, src
dst ← dst - src
dst:xmmR
src:xmmR/m64

VSUBSD (Subtract scalar double-precision)

Syntax: VSUBSD dst, src1, src2
dst ← src1 - src2
dst: xmmR
src1: xmmR
src2: xmmR/m64

Example:

```
section .data
var1 dq 4.5
var2 dq 2.75
section .text
MOVSD XMM1, [var1]
VSUBSD XMM2,XMM1,[var2]
```

1. What will XMM2 contain after execution?

XMM2 = 1.75

x86-64 Instructions : MULSS/VMULSS

MULSS (Multiply scalar single-precision)

Syntax: MULSS dst, src
dst ← dst * src
dst:xmmR
src:xmmR/m32

VMULSS (Multiply scalar single-precision)

Syntax:VMULSS dst, src1, src2
dst ← src1 * src2
dst: xmmR
src1: xmmR
src2: xmmR/m32

x86-64 Instructions : MULSS/VMULSS

MULSS (Multiply scalar single-precision)

Syntax: MULSS dst, src
dst ← dst * src
dst:xmmR
src:xmmR/m32

VMULSS (Multiply scalar single-precision)

Syntax:VMULSS dst, src1, src2
dst ← src1 * src2
dst: xmmR
src1: xmmR
src2: xmmR/m32

Example:

```
section .data
var1 dd 4.5
var2 dd 2.75
section .text
MOVSS XMM1, [var1]
MOVSS XMM2, [var2]
VMULSS XMM3,XMM1,XMM2
```

1. **What will XMM3 contain after execution?**

x86-64 Instructions : MULSS/VMULSS

MULSS (Multiply scalar single-precision)

Syntax: MULSS dst, src
dst ← dst * src
dst:xmmR
src:xmmR/m32

VMULSS (Multiply scalar single-precision)

Syntax:VMULSS dst, src1, src2
dst ← src1 * src2
dst: xmmR
src1: xmmR
src2: xmmR/m32

Example:

```
section .data
var1 dd 4.5
var2 dd 2.75
section .text
MOVSS XMM1, [var1]
MOVSS XMM2, [var2]
VMULSS XMM3,XMM1,XMM2
```

1. What will XMM3 contain after execution?

XMM3 = 12.375

x86-64 Instructions : MULSD/VMULSD

MULSD (Multiply scalar double-precision)

Syntax: MULSD dst, src
dst ← dst * src
dst:xmmR
src:xmmR/m64

VMULSD (Multiply scalar double-precision)

Syntax:VMULSD dst, src1, src2
dst ← src1 * src2
dst: xmmR
src1: xmmR
src2: xmmR/m64

x86-64 Instructions : MULSD/VMULSD

MULSD (Multiply scalar double-precision)

Syntax: MULSD dst, src
dst ← dst * src
dst:xmmR
src:xmmR/m64

VMULSD (Multiply scalar double-precision)

Syntax:VMULSD dst, src1, src2
dst ← src1 * src2
dst: xmmR
src1: xmmR
src2: xmmR/m64

Example:

```
section .data
var1 dq 4.5
var2 dq 2.75
section .text
MOVSD XMM1, [var1]
VMULSD XMM2,XMM1,[var2]
```

1. **What will XMM2 contain after execution?**

x86-64 Instructions : MULSD/VMULSD

MULSD (Multiply scalar double-precision)

Syntax: MULSD dst, src
dst ← dst * src
dst:xmmR
src:xmmR/m64

VMULSD (Multiply scalar double-precision)

Syntax:VMULSD dst, src1, src2
dst ← src1 * src2
dst: xmmR
src1: xmmR
src2: xmmR/m64

Example:

```
section .data
var1 dq 4.5
var2 dq 2.75
section .text
MOVSD XMM1, [var1]
VMULSD XMM2,XMM1,[var2]
```

1. What will XMM2 contain after execution?

XMM3 = 12.375

x86-64 Instructions : DIVSS/VDIVSS

DIVSS (Divide scalar single-precision)

Syntax: DIVSS dst, src
dst ← dst / src
dst:xmmR
src:xmmR/m32

VDIVSS (Divide scalar single-precision)

Syntax:VDIVSS dst, src1, src2
dst ← src1 / src2
dst: xmmR
src1: xmmR
src2: xmmR/m64

x86-64 Instructions : DIVSS/VDIVSS

DIVSS (Divide scalar single-precision)

Syntax: DIVSS dst, src
dst ← dst / src
dst:xmmR
src:xmmR/m32

VDIVSS (Divide scalar single-precision)

Syntax:VDIVSS dst, src1, src2
dst ← src1 / src2
dst: xmmR
src1: xmmR
src2: xmmR/m64

Example:

```
section .data
var1 dd 4.5
var2 dd 2.0
section .text
MOVSS XMM1, [var1]
MOVSS XMM2, [var2]
VDIVSS XMM3,XMM1,XMM2
```

1. **What will XMM3 contain after execution?**

x86-64 Instructions : DIVSS/VDIVSS

DIVSS (Divide scalar single-precision)

Syntax: DIVSS dst, src
dst ← dst / src
dst:xmmR
src:xmmR/m32

VDIVSS (Divide scalar single-precision)

Syntax:VDIVSS dst, src1, src2
dst ← src1 / src2
dst: xmmR
src1: xmmR
src2: xmmR/m64

Example:

```
section .data
var1 dd 4.5
var2 dd 2.0
section .text
MOVSS XMM1, [var1]
MOVSS XMM2, [var2]
VDIVSS XMM3,XMM1,XMM2
```

1. What will XMM3 contain after execution?

XMM3 = 2.25

x86-64 Instructions : DIVSD/VDIVSD

DIVSD (Divide scalar double-precision)

Syntax: DIVSD dst, src
dst ← dst / src
dst:xmmR
src:xmmR/m64

VDIVSD (Divide scalar double-precision)

Syntax:VDIVSD dst, src1, src2
dst ← src1 / src2
dst: xmmR
src1: xmmR
src2: xmmR/m64

x86-64 Instructions : DIVSD/VDIVSD

DIVSD (Divide scalar double-precision)

Syntax: DIVSD dst, src
dst ← dst / src
dst:xmmR
src:xmmR/m64

VDIVSD (Divide scalar double-precision)

Syntax:VDIVSD dst, src1, src2
dst ← src1 / src2
dst: xmmR
src1: xmmR
src2: xmmR/m64

Example:

```
section .data
var1 dq 4.5
var2 dq 2.0
section .text
MOVSD XMM1, [var1]
VDIVSD XMM2,XMM1,[var2]
```

1. **What will XMM2 contain after execution?**

x86-64 Instructions : DIVSD/VDIVSD

DIVSD (Divide scalar double-precision)

Syntax: DIVSD dst, src
dst ← dst / src
dst:xmmR
src:xmmR/m64

VDIVSD (Divide scalar double-precision)

Syntax:VDIVSD dst, src1, src2
dst ← src1 / src2
dst: xmmR
src1: xmmR
src2: xmmR/m64

Example:

```
section .data
var1 dq 4.5
var2 dq 2.0
section .text
MOVSD XMM1, [var1]
VDIVSD XMM2,XMM1,[var2]
```

1. What will XMM2 contain after execution?

XMM3 = 2.25

x86-64 Instructions : RCPSS

RCPSS (Reciprocal of scalar single-precision)

Syntax: RCPSS dst, src

$dst \leftarrow src^{-1}$

dst:xmmR

src:xmmR/m32

x86-64 Instructions : RCPSS

RCPSS (Reciprocal of scalar single-precision)

Syntax: RCPSS dst, src

dst ← src^{-1}

dst:xmmR

src:xmmR/m32

Example:

```
section .data
var1 dd 2.0
section .text
MOVSS XMM1, [var1]
RCPSS XMM3,XMM1
```

1. **What will XMM3 contain after execution?**

x86-64 Instructions : RCPSS

RCPSS (Reciprocal of scalar single-precision)

Syntax: RCPSS dst, src

dst ← src^{-1}

dst:xmmR

src:xmmR/m32

Example:

```
section .data
var1 dd 2.0
section .text
MOVSS XMM1, [var1]
RCPSS XMM3,XMM1
```

1. What will XMM3 contain after execution?

XMM3 = 0.5 (0.49987793)

x86-64 Instructions : SQRTSS/SQRTSD

SQRTSS (Square root of scalar single-precision)

Syntax: SQRTSS dst, src
dst ← sqrt(src)
dst:xmmR
src:xmmR/m32

SQRTSD (Square root of scalar double-precision)

Syntax: SQRTSD dst, src
dst ← sqrt(src)
dst:xmmR
src:xmmR/m64

x86-64 Instructions : SQRTSS/SQRTSD

SQRTSS (Square root of scalar single-precision)

Syntax: SQRTSS dst, src
dst ← sqrt(src)
dst:xmmR
src:xmmR/m32

SQRTSD (Square root of scalar double-precision)

Syntax: SQRTSD dst, src
dst ← sqrt(src)
dst:xmmR
src:xmmR/m64

Example:

```
section .data
var1 dd 4.0
section .text
MOVSS XMM1, [var1]
SQRTSS XMM3,XMM1
```

1. **What will XMM3 contain after execution?**

x86-64 Instructions : SQRTSS/SQRTSD

SQRTSS (Square root of scalar single-precision)

Syntax: SQRTSS dst, src
dst ← sqrt(src)
dst:xmmR
src:xmmR/m32

SQRTSD (Square root of scalar double-precision)

Syntax: SQRTSD dst, src
dst ← sqrt(src)
dst:xmmR
src:xmmR/m64

Example:

```
section .data
var1 dd 4.0
section .text
MOVSS XMM1, [var1]
SQRTSS XMM3,XMM1
```

1. What will XMM3 contain after execution?

XMM3 = 2.0

x86-64 Instructions : SQRTSS/SQRTSD

SQRTSS (Square root of scalar single-precision)

Syntax: SQRTSS dst, src
dst ← sqrt(src)
dst:xmmR
src:xmmR/m32

SQRTSD (Square root of scalar double-precision)

Syntax: SQRTSD dst, src
dst ← sqrt(src)
dst:xmmR
src:xmmR/m64

Example:

```
ection .data
var1 dq 4.0
section .text
MOVSD XMM1, [var1]
SQRTSD XMM3,XMM1
```

1. **What will XMM3 contain after execution?**

x86-64 Instructions : SQRTSS/SQRTSD

SQRTSS (Square root of scalar single-precision)

Syntax: SQRTSS dst, src
dst ← sqrt(src)
dst:xmmR
src:xmmR/m32

SQRTSD (Square root of scalar double-precision)

Syntax: SQRTSD dst, src
dst ← sqrt(src)
dst:xmmR
src:xmmR/m64

Example:

```
Section .data
var1 dq 4.0
section .text
MOVSD XMM1, [var1]
SQRTSD XMM3,XMM1
```

1. What will XMM3 contain after execution?

XMM3 = 2.0

x86-64 Instructions : RSQRTSS

RSQRTSS (Reciprocal of Square root of scalar single-precision)

Syntax: RSQRTSS dst, src

dst ← $[\text{sqrt}(\text{src})]^{-1}$

dst:xmmR

src:xmmR/m32

x86-64 Instructions : RSQRTSS

RSQRTSS (Reciprocal of Square root of scalar single-precision)

Syntax: RSQRTSS dst, src

dst ← $[sqrt(src)]^{-1}$

dst:xmmR

src:xmmR/m32

Example:

```
section .data
var1 dd 4.0
section .text
MOVSS XMM1, [var1]
RSQRTSS XMM3,XMM1
```

1. **What will XMM3 contain after execution?**

x86-64 Instructions : RSQRTSS

RSQRTSS (Reciprocal of Square root of scalar single-precision)

Syntax: RSQRTSS dst, src

dst ← $[sqrt(src)]^{-1}$

dst:xmmR

src:xmmR/m32

Example:

```
section .data
var1 dd 4.0
section .text
MOVSS XMM1, [var1]
RSQRTSS XMM3,XMM1
```

1. What will XMM3 contain after execution?

XMM3 = 0.5 (0.49987793)

x86-64 Instructions : MAXSS/VMAXSS

MAXSS (Return maximum scalar single-precision)

Syntax: MAXSS dst, src
dst ← max(dst, src)
dst:xmmR
src:xmmR/m32

VMAXSS (Return maximum scalar single-precision)

Syntax: VMAXSS dst, src1, src2
dst ← max(src1, src2)
dst:xmmR
src1: xmmR
src2:xmmR/m32

x86-64 Instructions : MAXSS/VMAXSS

MAXSS (Return maximum scalar single-precision)

Syntax: MAXSS dst, src
dst ← max(dst, src)
dst:xmmR
src:xmmR/m32

VMAXSS (Return maximum scalar single-precision)

Syntax: VMAXSS dst, src1, src2
dst ← max(src1, src2)
dst:xmmR
src1: xmmR
src2:xmmR/m32

Example:

```
section .data
var1 dd 4.5
var2 dd 2.0
section .text
MOVSS XMM1, [var1]
MOVSS XMM2, [var2]
VMAXSS XMM3,XMM1,XMM2
```

1. **What will XMM3 contain after execution?**

x86-64 Instructions : MAXSS/VMAXSS

MAXSS (Return maximum scalar single-precision)

Syntax: MAXSS dst, src
dst ← max(dst, src)
dst:xmmR
src:xmmR/m32

VMAXSS (Return maximum scalar single-precision)

Syntax: VMAXSS dst, src1, src2
dst ← max(src1, src2)
dst:xmmR
src1: xmmR
src2:xmmR/m32

Example:

```
section .data
var1 dd 4.5
var2 dd 2.0
section .text
MOVSS XMM1, [var1]
MOVSS XMM2, [var2]
VMAXSS XMM3,XMM1,XMM2
```

1. What will XMM3 contain after execution?

XMM3 = 4.5

x86-64 Instructions : MAXSD/VMAXSD

MAXSD (Return maximum scalar double-precision)

Syntax: MAXSD dst, src
dst ← max(dst, src)
dst:xmmR
src:xmmR/m64

VMAXSS (Return maximum scalar double-precision)

Syntax: VMAXSD dst, src1, src2
dst ← max(src1, src2)
dst:xmmR
src1: xmmR
src2:xmmR/m64

x86-64 Instructions : MAXSD/VMAXSD

MAXSD (Return maximum scalar double-precision)

Syntax: MAXSD dst, src
dst ← max(dst, src)
dst:xmmR
src:xmmR/m64

VMAXSS (Return maximum scalar double-precision)

Syntax: VMAXSD dst, src1, src2
dst ← max(src1, src2)
dst:xmmR
src1: xmmR
src2:xmmR/m64

Example:

```
section .data
var1 dq 4.5
var2 dq 2.0
section .text
MOVSD XMM1, [var1]
MOVSD XMM2, [var2]
VMAXSD XMM3,XMM1,XMM2
```

1. **What will XMM3 contain after execution?**

x86-64 Instructions : MAXSD/VMAXSD

MAXSD (Return maximum scalar double-precision)

Syntax: MAXSD dst, src
dst ← max(dst, src)
dst:xmmR
src:xmmR/m64

VMAXSS (Return maximum scalar double-precision)

Syntax: VMAXSD dst, src1, src2
dst ← max(src1, src2)
dst:xmmR
src1: xmmR
src2:xmmR/m64

Example:

```
section .data
var1 dq 4.5
var2 dq 2.0
section .text
MOVSD XMM1, [var1]
MOVSD XMM2, [var2]
VMAXSD XMM3,XMM1,XMM2
```

1. What will XMM3 contain after execution?

XMM3 = 4.5

x86-64 Instructions : MINSS/VMINSS

MINSS (Return minimum scalar single-precision)

Syntax: MINSS dst, src
dst ← min(dst, src)
dst:xmmR
src:xmmR/m32

VMINSS (Return minimum scalar single-precision)

Syntax: VMINSS dst, src1, src2
dst ← min(src1, src2)
dst:xmmR
src1: xmmR
src2:xmmR/m32

x86-64 Instructions : MINSS/VMINSS

MINSS (Return minimum scalar single-precision)

Syntax: MINSS dst, src
dst ← min(dst, src)
dst:xmmR
src:xmmR/m32

VMINSS (Return minimum scalar single-precision)

Syntax: VMINSS dst, src1, src2
dst ← min(src1, src2)
dst:xmmR
src1: xmmR
src2:xmmR/m32

Example:

```
section .data
var1 dd 4.5
var2 dd 2.0
section .text
MOVSS XMM1, [var1]
MOVSS XMM2, [var2]
VMINSS XMM3,XMM1,XMM2
```

1. **What will XMM3 contain after execution?**

x86-64 Instructions : MINSS/VMINSS

MINSS (Return minimum scalar single-precision)

Syntax: MINSS dst, src
dst ← min(dst, src)
dst:xmmR
src:xmmR/m32

VMINSS (Return minimum scalar single-precision)

Syntax: VMINSS dst, src1, src2
dst ← min(src1, src2)
dst:xmmR
src1: xmmR
src2:xmmR/m32

Example:

```
section .data
var1 dd 4.5
var2 dd 2.0
section .text
MOVSS XMM1, [var1]
MOVSS XMM2, [var2]
VMINSS XMM3,XMM1,XMM2
```

1. What will XMM3 contain after execution?

XMM3 = 2.0

x86-64 Instructions : MINSD/VMINSD

MINSD (Return minimum scalar double-precision)

Syntax: MINSD dst, src
dst ← min(dst, src)
dst:xmmR
src:xmmR/m64

VMINSD (Return minimum scalar double-precision)

Syntax: VMINSD dst, src1, src2
dst ← min(src1, src2)
dst:xmmR
src1: xmmR
src2:xmmR/m64

x86-64 Instructions : MINSD/VMINSD

MINSD (Return minimum scalar double-precision)

Syntax: MINSD dst, src
dst ← min(dst, src)
dst:xmmR
src:xmmR/m64

VMINSD (Return minimum scalar double-precision)

Syntax: VMINSD dst, src1, src2
dst ← min(src1, src2)
dst:xmmR
src1: xmmR
src2:xmmR/m64

Example:

```
section .data
var1 dq 4.5
var2 dq 2.0
section .text
MOVSD XMM1, [var1]
MOVSD XMM2, [var2]
VMINSD XMM3,XMM1,XMM2
```

1. **What will XMM3 contain after execution?**

x86-64 Instructions : MINSD/VMINSD

MINSD (Return minimum scalar double-precision)

Syntax: MINSD dst, src
dst ← min(dst, src)
dst:xmmR
src:xmmR/m64

VMINSD (Return minimum scalar double-precision)

Syntax: VMINSD dst, src1, src2
dst ← min(src1, src2)
dst:xmmR
src1: xmmR
src2:xmmR/m64

Example:

```
section .data
var1 dq 4.5
var2 dq 2.0
section .text
MOVSD XMM1, [var1]
MOVSD XMM2, [var2]
VMINSD XMM3,XMM1,XMM2
```

1. What will XMM3 contain after execution?

XMM3 = 2.0

x86-64 Instructions : UCOMISS

UCOMISS (Unordered compare scalar single-precision floating-point)

Syntax: UCOMISS src1,src2

ZF, PF, CF ← compare(src1, src2)

src1:xmmR

src2:xmmR/m32

Flags Affected:

- Greater than: ZF,PF,CF = 000
- Less than: ZF,PF,CF = 001
- Equal: ZF,PF,CF = 100
- Unordered: ZF,PF,CF = 111
- Unordered: if either source operand is a NaN
- OF,AF,SF are set to 0

x86-64 Instructions : UCOMISS

UCOMISS (Unordered compare scalar single-precision floating-point)

Syntax: UCOMISS src1,src2

ZF, PF, CF ← compare(src1, src2)
src1:xmmR
src2:xmmR/m32

Flags Affected:

- Greater than: ZF,PF,CF = 000
- Less than: ZF,PF,CF = 001
- Equal: ZF,PF,CF = 100
- Unordered: ZF,PF,CF = 111
- Unordered: if either source operand is a NaN
- OF,AF,SF are set to 0

Example:

```
section .data
var1 dd 4.0
var2 dd 4.5
section .text
MOVSS XMM1, [var1]
MOVSS XMM2, [var2]
UCOMISS XMM1, XMM2
```

1. **What will ZF,PF,SF contain after execution?**

x86-64 Instructions : UCOMISS

UCOMISS (Unordered compare scalar single-precision floating-point)

Syntax: UCOMISS src1,src2

ZF, PF, CF ← compare(src1, src2)
src1:xmmR
src2:xmmR/m32

Flags Affected:

- Greater than: ZF,PF,CF = 000
- Less than: ZF,PF,CF = 001
- Equal: ZF,PF,CF = 100
- Unordered: ZF,PF,CF = 111
- Unordered: if either source operand is a NaN
- OF,AF,SF are set to 0

Example:

```
section .data
var1 dd 4.0
var2 dd 4.5
section .text
MOVSS XMM1, [var1]
MOVSS XMM2, [var2]
UCOMISS XMM1, XMM2
```

1. What will ZF,PF,SF contain after execution?

ZF = 0, PF = 0, CF=1

x86-64 Instructions : UCOMISD

UCOMISS (Unordered compare scalar double-precision floating-point)

Syntax: UCOMISD src1,src2
ZF, PF, CF ← compare(src1, src2)
src1:xmmR
src2:xmmR/m64

Flags Affected:
- Greater than: ZF,PF,CF = 000
- Less than: ZF,PF,CF = 001
- Equal: ZF,PF,CF = 100
- Unordered: ZF,PF,CF = 111
- Unordered: if either source operand is a NaN
- OF,AF,SF are set to 0

x86-64 Instructions : UCOMISD

UCOMISS (Unordered compare scalar double-precision floating-point)

Syntax: UCOMISD src1,src2

ZF, PF, CF ← compare(src1, src2)
src1:xmmR
src2:xmmR/m64

Flags Affected:

- Greater than: ZF,PF,CF = 000
- Less than: ZF,PF,CF = 001
- Equal: ZF,PF,CF = 100
- Unordered: ZF,PF,CF = 111
- Unordered: if either source operand is a NaN
- OF,AF,SF are set to 0

Example:

```
section .data
var1 dq 4.0
var2 dq 4.5
section .text
MOVSD XMM1, [var1]
MOVSD XMM2, [var2]
UCOMISD XMM1, XMM2
```

1. **What will ZF,PF,SF contain after execution?**

x86-64 Instructions : UCOMISD

UCOMISS (Unordered compare scalar double-precision floating-point)

Syntax: UCOMISD src1,src2
ZF, PF, CF ← compare(src1, src2)
src1:xmmR
src2:xmmR/m64

Flags Affected:
- Greater than: ZF,PF,CF = 000
- Less than: ZF,PF,CF = 001
- Equal: ZF,PF,CF = 100
- Unordered: ZF,PF,CF = 111
- Unordered: if either source operand is a NaN
- OF,AF,SF are set to 0

Example:

```
section .data
var1 dq 4.0
var2 dq 4.5
section .text
MOVSD XMM1, [var1]
MOVSD XMM2, [var2]
UCOMISD XMM1, XMM2
```

1. What will ZF,PF,SF contain after execution?

ZF = 0, PF = 0, CF=1

x86-64 Instructions : CVTSS2SI

CVTSS2SI (convert scalar single-precision floating-point value to doubleword integer)

Syntax: CVTSS2SI dst,src
dst ← cvtss2si(src)
dst:r32_64
src:xmmR/m32

x86-64 Instructions : CVTSS2SI

CVTSS2SI (convert scalar single-precision floating-point value to doubleword integer)

Syntax: CVTSS2SI dst,src
dst ← cvtss2si(src)
dst:r32_64
src:xmmR/m32

Example:

```
section .data
var1 dd 4.5
section .text
MOVSS XMM1, [var1]
CVTSS2SI EAX, XMM1
```

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

x86-64 Instructions : CVTSS2SI

CVTSS2SI (convert scalar single-precision floating-point value to doubleword integer)

Syntax: CVTSS2SI dst,src
dst ← cvtss2si(src)
dst:r32_64
src:xmmR/m32

Example:

```
section .data
var1 dd 4.5
section .text
MOVSS XMM1, [var1]
CVTSS2SI EAX, XMM1
```

1. What will EAX contain after execution?

EAX=00000004

x86-64 Instructions : CVTSD2SI

CVTSD2SI (convert scalar double-precision floating-point value to doubleword integer)

Syntax: CVTSD2SI dst,src
dst ← cvtsd2si(src)
dst:r32_64
src:xmmR/m64

x86-64 Instructions : CVTSD2SI

CVTSD2SI (convert scalar double-precision floating-point value to doubleword integer)

Syntax: CVTSD2SI dst,src
dst ← cvtsd2si(src)
dst:r32_64
src:xmmR/m64

Example:

```
section .data
var1 dq 5.5
section .text
MOVSD XMM1, [var1]
CVTSD2SI RAX, XMM1
```

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

x86-64 Instructions : CVTSD2SI

CVTSD2SI (convert scalar double-precision floating-point value to doubleword integer)

Syntax: CVTSD2SI dst,src
dst ← cvtsd2si(src)
dst:r32_64
src:xmmR/m64

Example:

```
section .data
var1 dq 5.5
section .text
MOVSD XMM1, [var1]
CVTSD2SI RAX, XMM1
```

1. What will RAX contain after execution?

RAX=0000000000000006

x86-64 Instructions : CVTSI2SS

CVTSI2SS (convert doubleword integer to scalar single-precision floating-point value)

Syntax: CVTSI2SS dst,src
dst ← cvtsi2ss(src)
dst:xmmR
src:[r/m]_32_64

x86-64 Instructions : CVTSI2SS

CVTSI2SS (convert doubleword integer to scalar single-precision floating-point value)

Syntax: CVTSI2SS dst,src
dst ← cvtsi2ss(src)
dst:xmmR
src:[r/m]_32_64

Example:

```
section .text
MOV EAX, 4
CVTSI2SS XMM1, EAX
```

1. **What will XMM1 contain after execution?**

x86-64 Instructions : CVTSI2SS

CVTSI2SS (convert doubleword integer to scalar single-precision floating-point value)

Syntax: CVTSI2SS dst,src
dst ← cvtsi2ss(src)
dst:xmmR
src:[r/m]_32_64

Example:

```
section .text
MOV EAX, 4
CVTSI2SS XMM1, EAX
```

1. What will XMM1 contain after execution?

   XMM1=4.0

x86-64 Instructions : CVTSI2SD

CVTSI2SD (convert doubleword integer to scalar double-precision floating-point value)

Syntax: CVTSI2SD dst,src
dst ← cvtsi2sd(src)
dst:xmmR
src:[r/m]_32_64

x86-64 Instructions : CVTSI2SD

CVTSI2SD (convert doubleword integer to scalar double-precision floating-point value)

Syntax: CVTSI2SD dst,src
dst ← cvtsi2sd(src)
dst:xmmR
src:[r/m]_32_64

Example:

```
section .text
MOV RAX, 4
CVTSI2SD XMM1, RAX
```

1. **What will XMM1 contain after execution?**

x86-64 Instructions : CVTSI2SD

CVTSI2SD (convert doubleword integer to scalar double-precision floating-point value)

Syntax: CVTSI2SD dst,src
dst ← cvtsi2sd(src)
dst:xmmR
src:[r/m]_32_64

Example:

```
section .text
MOV RAX, 4
CVTSI2SD XMM1, RAX
```

1. What will XMM1 contain after execution?

XMM1=4.0