Assembly Language Lecture Series:

x86-64 Floating-Point Register

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

Copyright Notice

SIMD Register

- SIMD registers can be used **in scalar or packed format**. Can store either **integer or floating-point data type**
  - Sixteen (16) 128-bit XMM registers (XMM0-XMM15)

SIMD Packed Register

Single-Precision FP	Single-Precision FP	Single-Precision FP	Single-Precision FP
127-96	95-64	63-32	bits 31-0

Each XMM register can support up to **4 packed single-precision floating-point**

Double-Precision FP	Double-Precision FP
127-64	Bits 63-0

Each XMM register can support up to **2 packed double-precision floating-point**

SIMD Scalar Register

			Single-Precision FP
			bits 31-0

XMM register can also be used as **scalar single-precision register**.
Only the **least significant 32-bit** is used.

	Double-Precision FP
	Bits 63-0

XMM register can also be used as **scalar double-precision register**.
Only the **least significant 64-bit** is used.

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

GDB Command

w64_ScalarFP.asm | *New

```
1  %include "io64.inc"
2  section .data
3  var1 dd 4.5
4  section .text
5  global CMAIN
6  CMAIN:
7    ;write your code here
8    movss xmm1, [var1]
9    xor rax, rax
10   ret
```

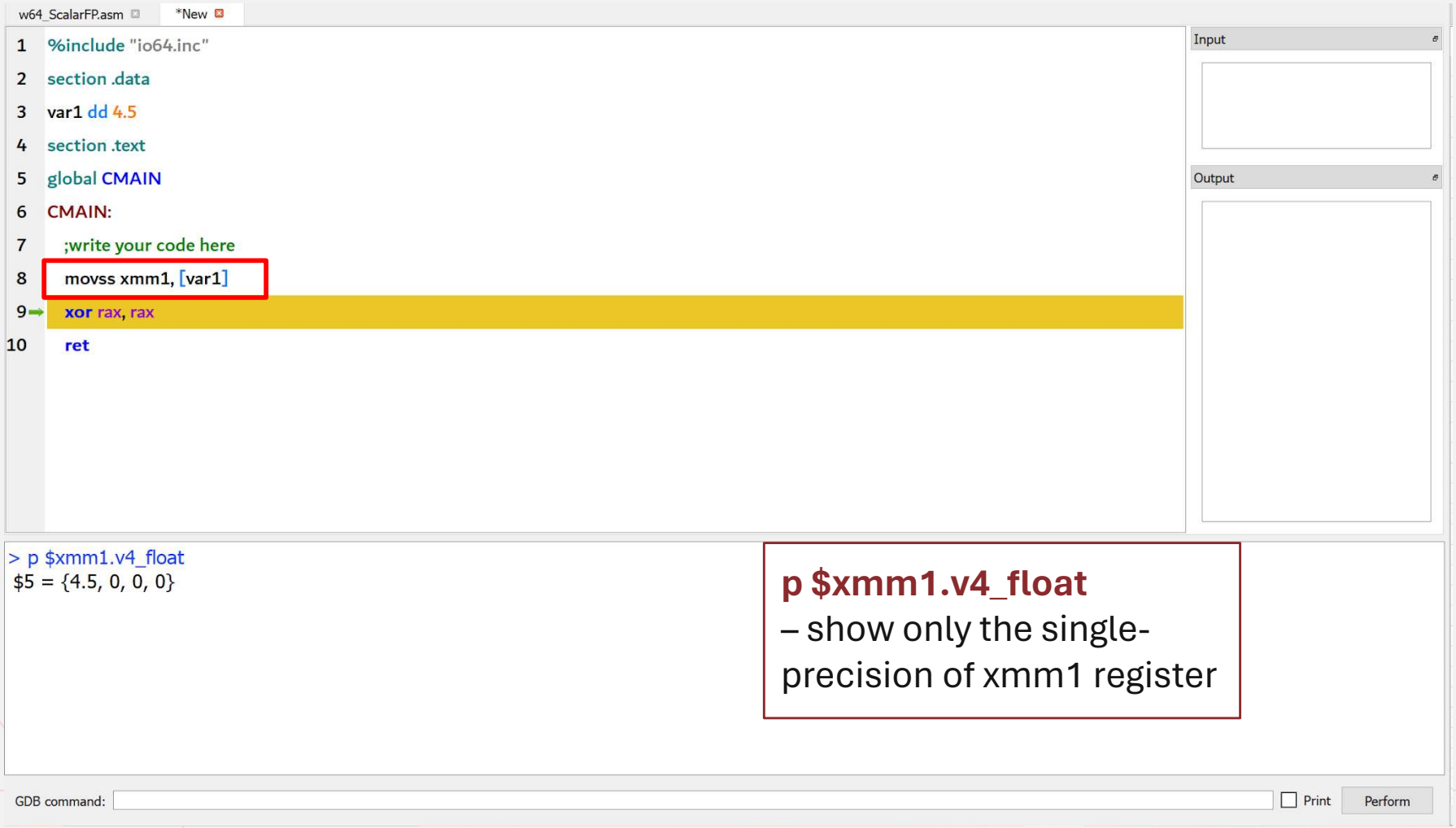
Input

Output

```
  114, 111, 103, 46, 101, 120, 101, 0}, v8_int16 = {19795, 21340, 21313,
  28749, 28530, 11879, 30821, 101}, v4_int32 = {1398558035, 1884115777,
  778530674, 6649957}, v2_int64 = {8092215645491187027, 28561348613336946},
 uint128 = 0x006578652e676f72704d5341535c4d53}
> p $xmm1
$3 = {v4_float = {4.5, 0, 0, 0}, v2_double = {5.3516153614920076e-315, 0},
 v16_int8 = {0, 0, -112, 64, 0 <repeats 12 times>}, v8_int16 = {0, 16528, 0,
  0, 0, 0, 0, 0}, v4_int32 = {1083179008, 0, 0, 0}, v2_int64 = {1083179008,
  0}, uint128 = 1083179008}
```

p $xmm1 – list all packed combinations of an XMM register

GDB Command


w64_ScalarFP.asm
*New
1 %include "io64.inc"
2 section .data
3 var1 dd 4.5
4 section .text
5 global CMAIN
6 CMAIN:
7 ;write your code here
8 movss xmm1, [var1]
9 xor rax, rax
10 ret
Input
Output
> p $xmm1.v4_float
$5 = {4.5, 0, 0, 0}
p $xmm1.v4_float
– show only the single-precision of xmm1 register
GDB command:
Print
Perform

GDB Command

```
1  %include "io64.inc"
2  section .data
3  var1 dq 4.5
4  section .text
5  global CMAIN
6  CMAIN:
7    ;write your code here
8    movsd xmm1, [var1]
9    xor rax, rax
10   ret
```

Input

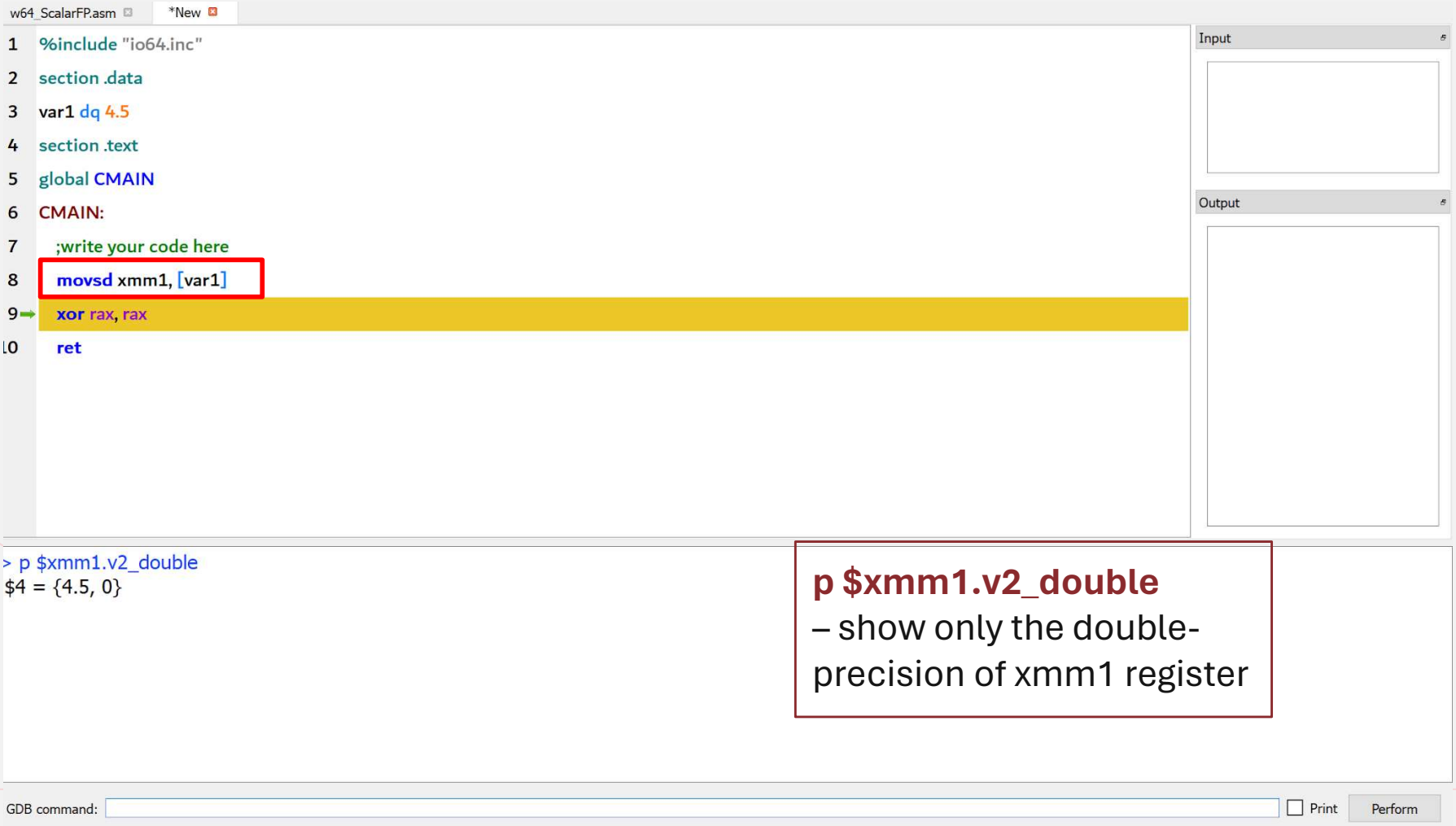
Output

```
> p $xmm1
$2 = {v4_float = {0, 2.28125, 0, 0}, v2_double = {4.5, 0}, v16_int8 = {0, 0,
  0, 0, 0, 0, 18, 64, 0, 0, 0, 0, 0, 0, 0, 0}, v8_int16 = {0, 0, 0, 16402,
  0, 0, 0, 0}, v4_int32 = {0, 1074921472, 0, 0}, v2_int64 = {
  4616752568008179712, 0}, uint128 = 4616752568008179712}
```

p $xmm1 – list all packed combinations of an XMM register

GDB command: Print Perform

GDB Command



p $xmm1.v2_double
– show only the double-precision of xmm1 register