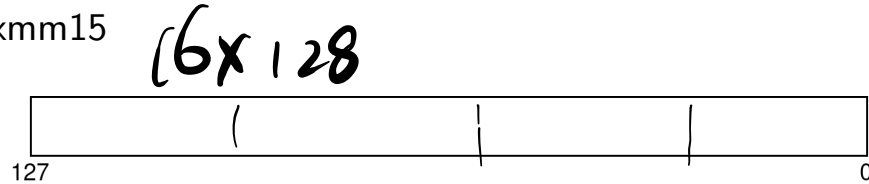


# SSE-Register

## SSE 指令集

- ▶ 16 weitere Register

- ▶ xmm0 bis xmm15



- ▶ Für skalare Berechnungen sind nur die unteren 32 bzw. 64 Bit von Relevanz

初電

- ▶ xmm-Registern verwendbar für Floating-Point Berechnungen

# Konstanten

0

- ▶ XOR-Instruktionen für xmm-Register

- ▶ `pxor dst, src`

- ▶ Floating-Point Konstanten können aus dem Speicher (z.B. `.rodata`) geladen werden

- ▶ `movss dst, src`

- ▶ Beispiel: `movss xmm0, [rip + .Lconstx]`

xmm15

- ▶ Moves zwischen General-Purpose und xmm-Registern möglich

- ▶ `movd / movq`

- ▶ Keine Konvertierung!

- ▶ Gut für Bitmanipulationen

是内存: [32, 12, 7] auf 0-gesetzt  
↑  
MOVSS  
是寄存器: 只改前31位  
不为 [32, 12, 7]

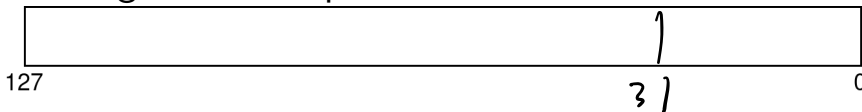
# Arithmetik – Teil 1

- ▶ Namenskonvention bei Instruktionen:
  - ▶ 'ss' → Scalar Single (Precision)
  - ▶ 'sd' → Scalar Double (Precision)

# Arithmetik – Teil 2

- ▶ `addss dest, src` – Addition zweier Floating-Point Werte

- ▶ `src`: Register oder Speicher



- ▶ Untere 32 bit für Addition mit `addss`

- ▶ `subss dest, src` – Subtraktion zweier Floating-Point Werte

- ▶ Analog zu `addss`

*mov ss  
add ss  
sub ss  
mul ss  
div ss*

## Arithmetik – Teil 3

- ▶ `divss dest, src` – Division mit zwei Floating-Point Werten
  - ▶ Unterschied zu `div`: Operanden werden nicht implizit gefolgert, explizite Angabe
- ▶ `mulss dest, src` – Multiplikation mit zwei Floating-Point Werten
  - ▶ Analog zu `divss`
- ▶ Konstanter Divisor: Multiplikation mit Kehrwert bevorzugen

# Vergleiche

- ▶ ucomiss op1, op2 - Skalarer Vergleich zweier Floating-Point Werte
- ▶ Flags gesetzt in Abhängigkeit des Ergebnisses
  - ▶ Ermöglicht Sprünge mittels jCC
  - ▶ Aber: Condition Codes für vorzeichenlose Vergleiche

```
1 cmpFloat:
2     ucomiss xmm1, xmm0 I 先字: Comp a.i
3     # jp .Lunordered // xmm0 or xmm1 NaN
4     jbe .Llessor ← jl // xmm1 < xmm0
5     jae .Lgreater ← jg // xmm1 > xmm0
6     jbe .Lequal ← jbe // xmm1 == xmm0
```

- ▶ ucomiss behandelt Vergleiche mit NaN gesondert
  - ▶ Überprüfbar mit jp bzw. jnp

# Codebeispiel

```
1 #include <stdio.h>
2
3 extern float func(float x);
4
5 int main(int argc, char** argv) {
6     float res = func(2.0);
7     printf("Result: %f\n", res);
8 }
```


```
1 .intel_syntax noprefix
2 .global func
3 .text
4 func:
5     mov r8, 1
6     cvtsi2ss xmm1, r8
7     divss xmm1, xmm0
8     movss xmm0, xmm1
9     ret
```

$\div = \div$

$\left\{ \begin{array}{l} \text{Cvt si 2ss} \quad 64 \text{ signed} \rightarrow \text{float} \rightarrow 128\text{bit 浮点} \\ \text{cvt ss 2si} \quad 32 \text{ signed} \rightarrow 64 \text{ 整数} \end{array} \right.$

# Erweiterte Calling Convention - Teil 1

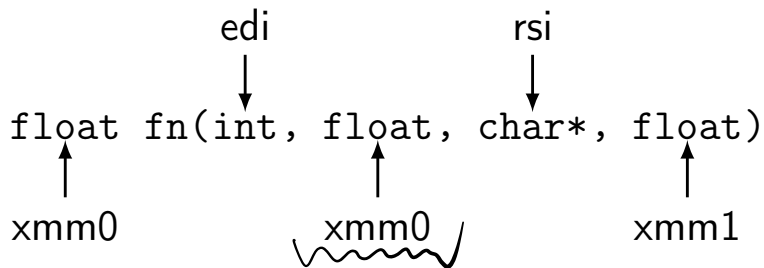


- ▶ Registerknappheit motiviert Erweiterung der CC
  - ▶ Floating-Point Rückgabewert: `xmm0`
  - ▶ Floating-Point Argumente: `xmm0` bis `xmm7` (weitere auf Stack)
  - ▶ Wichtig: Alle Register sind caller-saved/temporär
- 



# Erweiterte Calling Convention - Teil 2

- ▶ Kombinationen von FP und Int/Ptr Argumenten
  - ▶ Separate Durchnummerierung der Register



```
1  ex:
2      mov rdi, 1
3      movss xmm0, [rip + .Lconstx]
4      mov rsi, 2
5      movss xmm1, [rip + .Lconsty]
6      call fn
7      <...>
```

## Quiz: movss

Kreuzen Sie alle richtigen Antworten in Bezug auf movss an.



Wenn der zweite Operand ein Speicherzugriff ist, werden die Bits [32;127] des xmm-Registers auf Null gesetzt



Wenn der zweite Operand ein Register ist, dann werden nur die Bits [0;31] des Ziel-Registers geändert



movss benötigt immer mindestens einen Speicheroperanden



movss erlaubt es, Single-Precision Floating-Point Werte zwischen Registern bzw. Registern und dem Speicher zu bewegen

Int-extended

Quiz: cvtsi2ss

Kreuzen Sie alle richtigen Antworten in Bezug auf cvtsi2ss an.

☐

Für diese Instruktion gibt es kein Gegenstück für Double-Precision Floats

☒

Die Instruktion konvertiert eine Ganzzahl in einen Single-Precision Floating-Point Wert

☐

Die Instruktion konvertiert eine Ganzzahl in einen Double-Precision Floating-Point Wert

☒

Die Komponente 'si' steht für Single Integer

32 → 64

## Quiz: cvtss2si

Kreuzen Sie alle richtigen Antworten in Bezug auf cvtss2si an.

☒

Der Floating-Point Wert kann aus dem Speicher bezogen werden

☒

Die Instruktion konvertiert einen Single-Precision Floating-Point Wert in eine Ganzzahl

☐

Es findet keine Rundung bei der Konvertierung statt

☐

Der erste Operand ist immer xmm1

## Quiz: func

Was berechnet die Funktion 'func'?



$$\frac{x}{8}$$



$$\frac{1}{x}$$



$$x^2$$

func:

```
mov r8, 1
ctvsi2ss xmm1, r8
divss xmm1, xmm0
movss xmm0, xmm1
ret
```

## Quiz: Calling Convention

In welchem SSE-Register liegt der Rückgabewert?

☐

xmm3

☐

xmm16

☒

xmm0

func:

mov r8, 1

ctvsi2ss xmm1, r8

divss xmm1, xmm0

movss xmm0, xmm1

ret