

IF-Bedingung bisher

```
1 // if (a[i] != b[i])
2 //   a[i] = 0;
3   mov   ecx, DWORD PTR [rsi]
4   cmp   DWORD PTR [rdi], ecx
5   je    .Lskip
6   mov   BYTE PTR [rdi], 0
7 .Lskip:
8   ...
```

- ▶ Ergebnis von CMP im Flag-Register
- ▶ Abhängig davon wird MOV übersprungen
- ▶ Probleme für SIMD:
- ▶ Kein Flag-Register für jedes Vektor-Element
- ▶ Instruktion kann nur für alle Elemente übersprungen werden

Maskierung

- ▶ Vergleiche mit SIMD funktionieren anders
- ▶ Bedingung bereits im Vergleich
- ▶ Ergebnis nicht in Flag-Register, sondern XMM-Register
- ▶ Ergebnis als Bitmaske
- ▶ Falls Bedingung erfüllt, alle Bits in Element auf 1 gesetzt
- ▶ Falls Bedingung nicht erfüllt, alle Bits in Element auf 0 gesetzt

Vergleichsbefehle

- ▶ PCMPEQB xmm1, xmm2/m128
 - ▶ Vergleicht Bytes von xmm1 mit xmm2/m128 auf Gleichheit
 - ▶ Ergebnis als Bitmaske in xmm1
- ▶ PCMPEQW xmm1, xmm2/m128
- ▶ PCMPEQD xmm1, xmm2/m128
- ▶ PCMPEQQ xmm1, xmm2/m128

Beispiel

```
pcmpeqd xmm0, xmm1
```

xmm0	15	20	1024	0xffffffff
	=	=	=	=
xmm1	15	100	1024	33

Ergebnis in xmm0

0xffffffff	0	0xffffffff	0
------------	---	------------	---

Quiz

Welche der Elemente in res stimmen mit dem Ergebnis von pcmpeqd xmm0, xmm1 überein?

xmm0	0	20	0x15	1
	=	=	=	=
xmm1	0	20	15	-1
res	0xffffffff	1	0	0xffffffff
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Vergleichsbefehle

- ▶ PCMPGTB xmm1, xmm2/m128
- ▶ PCMPGTW xmm1, xmm2/m128
- ▶ PCMPGTD xmm1, xmm2/m128
- ▶ PCMPGTQ xmm1, xmm2/m128

Quiz

Welche der Elemente in res stimmen mit dem Ergebnis von `pcmpgtd` `xmm0`, `xmm1` überein?

<code>xmm0</code>	2	0x10	103	0xffffffff
	>	>	>	>
<code>xmm1</code>	2	10	-1	1
<code>res</code>	0	1	0xffffffff	0
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Quiz

Wie kann man eine Bitmaske erstellen die prüft ob die 16-Bit großen Elemente von xmm0 kleiner sind als die von xmm1?

`pcmpgtw xmm1, xmm0`

`pcmpltw xmm0, xmm1`

`pcmpgtb xmm1, xmm0`

Maske anwenden

- ▶ Problem: Instruktion kann nicht für einzelne Elemente übersprungen werden
- ▶ Bitmaske auf Eingabe von Instruktion anwenden
 - ▶ Instruktion hat nur einen Effekt falls die Bedingung erfüllt ist

Maske anwenden

```
1 // if (a[i] != b[i])
2 //   a[i] = 0;
3
4 // xmm0=a[i]
5 movdqa  xmm0, xmmword ptr [rdi]
6 // xmm1=b[i]
7 movdqa  xmm1, xmmword ptr [rsi]
8 pcmpeqd xmm1, xmm0
9 pand    xmm0, xmm1
10 movdqa  xmmword ptr [rdi], xmm0
```

- ▶ Die nächsten 4 Werte in xmm-Register laden
- ▶ Mit pcmpeqd Bitmaske erstellen
- ▶ Bitmaske überschreibt xmm1
- ▶ Falls Bedingung nicht erfüllt:
 - ▶ PAND setzt Element auf 0
- ▶ Falls Bedingung erfüllt:
 - ▶ Element bleibt unverändert

Quiz

Wie kann man eine Bitmaske erstellen die prüft ob die 32-Bit großen Elemente von xmm0 kleiner oder gleich sind wie die von xmm1?

```
pcmpged xmm1, xmm0
```

```
pcmpgtd xmm1, xmm0  
pcmpeqd xmm0, xmm1  
por     xmm0, xmm1
```

```
movdqu  xmm2, xmm1  
pcmpgtd xmm1, xmm0  
pcmpeqd xmm0, xmm2  
por     xmm0, xmm1
```

Quiz

Wie kann man eine Bitmaske erstellen die prüft ob die **vorzeichenlosen** 8-Bit Elemente von xmm0 kleiner sind als die von xmm1?

zuerst alle Elemente beider Register mit -1 multiplizieren,
dann: `pcmpgtb xmm0, xmm1`

zuerst 128 auf alle Elemente beider Register addieren,
dann: `pcmpgtb xmm1, xmm0`

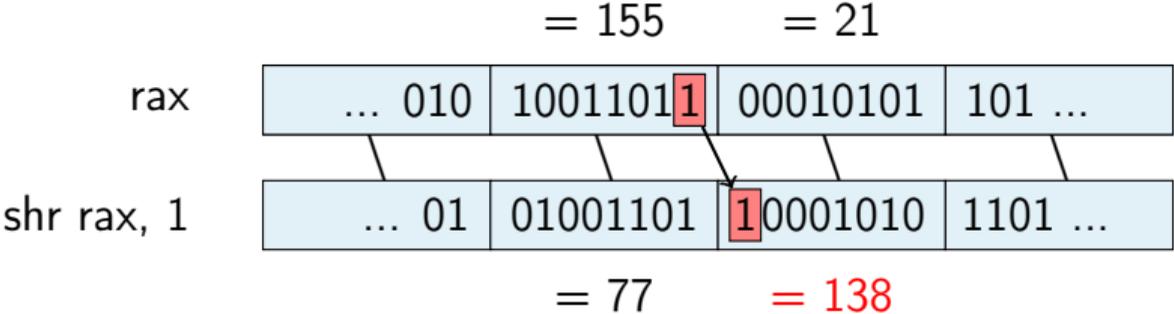
zuerst 128 von allen Elemente beider Register abziehen,
dann: `pcmpgtb xmm1, xmm0`

SIMD mit General Purpose Registern

- ▶ SIMD: Single Instruction, Multiple Data
- ▶ Bis jetzt mit Hilfe von SSE und AVX
- ▶ SIMD geht auch mit General Purpose Registern
- ▶ Manuell durchsetzen dass ein Element keinen Einfluss auf ein anderes hat

SIMD mit General Purpose Registern

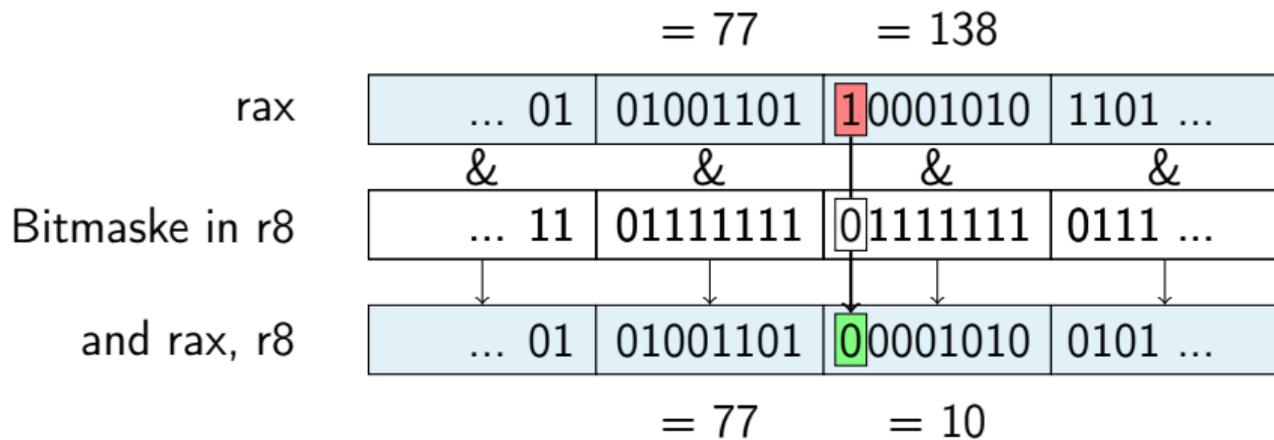
- ▶ Beispiel: Elemente von `uint8_t` Array durch 2 teilen
- ▶ 8 Elemente in `rax` laden, Division durch 2 mit Bitshift



- ▶ Problem: Niedrigstes Bit von einem Element wird ins nächste geschoben

SIMD mit General Purpose Registern

- ▶ Höchsten Bits sollen auf 0 gesetzt werden



Quiz

Wie kann man die Division durch 2 für einen Vorzeichenbehafteten `int8_t` Array anpassen?

(32 Bit Register werden hier für Immediates verwendet, geht aber auch mit 64 Bit)

```
mov eax, [rdi]
sar eax, 1
and eax, 0x7f7f7f7f
```

```
mov eax, [rdi]
shr eax, 1
and eax, 0x7f7f7f7f
mov edx, [rdi]
and edx, 0x80808080
or  eax, edx
```

```
mov eax, [rdi]
sar eax, 1
and eax, 0xff7f7f7f
mov edx, [rdi]
and edx, 0x00808080
or  eax, edx
```

Wann ist SIMD sinnvoll?

- ▶ Große Datenmengen
 - ▶ Bildbearbeitung
 - ▶ Matrixoperationen
- ▶ Gleiche oder ähnliche Instruktionen auf Elementen
- ▶ Daten müssen geschickt im Speicher liegen
 - ▶ z.B. Linked Lists schlechter als Arrays
- ▶ Keine anderen Funktionen in Schleife aufrufen
 - ▶ z.B. `a[i] = foo(i);`
- ▶ Auf loop carried dependences aufpassen
 - ▶ z.B. `a[i] += a[i-1];`

Quiz

Welche der folgenden Programme können vektorisiert werden?

```
a[i] += b[i-1];
```

```
a[i] += a[i-40];
```

```
a[i] += b[i-1];  
b[i-1] = a[i];
```