Textverarbeitung mit SIMD-Techniken

Robert Clausecker
<fuz@fuz.su>

Was sind eigentlich Strings?

Arten von Strings

Nullterminierte Strings

- Array von Zeichen gefolgt von NUL

Gezählte Strings

- Tupel aus Länge und Zeiger auf Array von Zeichen

Pascal-Strings

- Länge gefolgt von Array von Zeichen

Arten von Strings

Nullterminierte Strings

- Array von Zeichen gefolgt von NUL

Gezählte Strings

- Tupel aus Länge und Zeiger auf Array von Zeichen

~~**Pascal-Strings**~~

- ~~Länge gefolgt von Array von Zeichen~~

Übliche Aufgaben

- Strings kopieren (*strcpy*, *memcpy*)
- String-Länge bestimmen (*strlen*)
- Zeichen suchen (*strchr*, *memchr*)
- Strings vergleichen (*strcmp*, *memcmp*)
- nach Substrings suchen (*strstr*, *memmem*)
- in Tokens zerlegen (*strspn*, *strcspn*)

Übliche Aufgaben

- Strings kopieren (*einlesen*, *rausschreiben*)
- String-Länge bestimmen (*einlesen*, *vergleichen*)
- Zeichen suchen (*einlesen*, *vergleichen*)
- Strings vergleichen (*einlesen*, *vergleichen*)
- ~~nach Substrings suchen~~ (kompliziert)
- in Tokens zerlegen (*einlesen*, *Mengenvergleich*)

Was heißt das in der Praxis?

einlesen

- Zeichen für Zeichen, bis String zu Ende
- ein Ladezugriff, Vergleich, bedingter Sprung pro Zeichen

rausschreiben

- Zeichen für Zeichen, bis String zu Ende
- ein Schreibzugriff pro Zeichen

vergleichen

- Zeichen für Zeichen, bis Treffer oder String zu Ende
- ein Vergleich, bedingter Sprung pro Zeichen

Was heißt das in der Praxis?

einlesen

- Zeichen für Zeichen, bis String zu Ende
- ein Ladezugriff, Vergleich, bedingter Sprung pro Zeichen (**lahm**)

rausschreiben

- Zeichen für Zeichen, bis String zu Ende
- ein Schreibzugriff pro Zeichen (**lahm)**

vergleichen

- Zeichen für Zeichen, bis Treffer oder String zu Ende
- ein Vergleich, bedingter Sprung pro Zeichen (**lahm)**

Conclusio

Conclusio

Strings sind kacke.

Was können wir tun?

- Auf Strings verzichten (uff...)
- Spezielle Maschinenbefehle (je nach Platform)
  - Geschwindigkeit variiert stark je nach CPU-Modell
  - oft nur *memcpy*(), *memset*()
- seltsame Hacks (hm...)

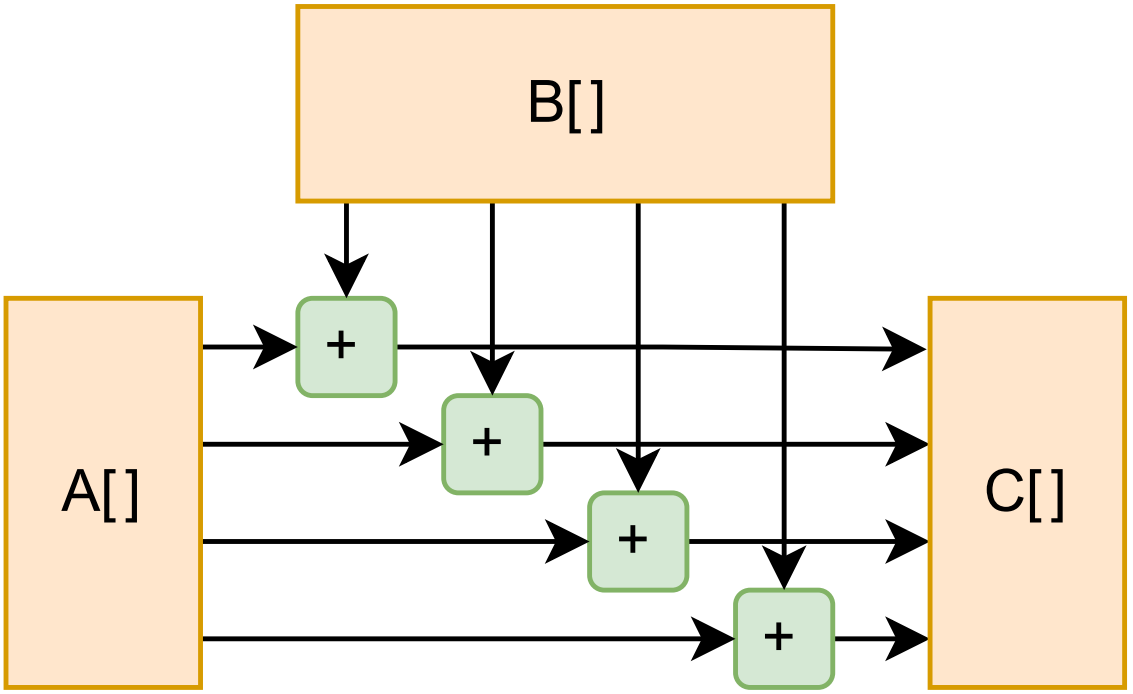
SIMD

Dein neuer bester Freund

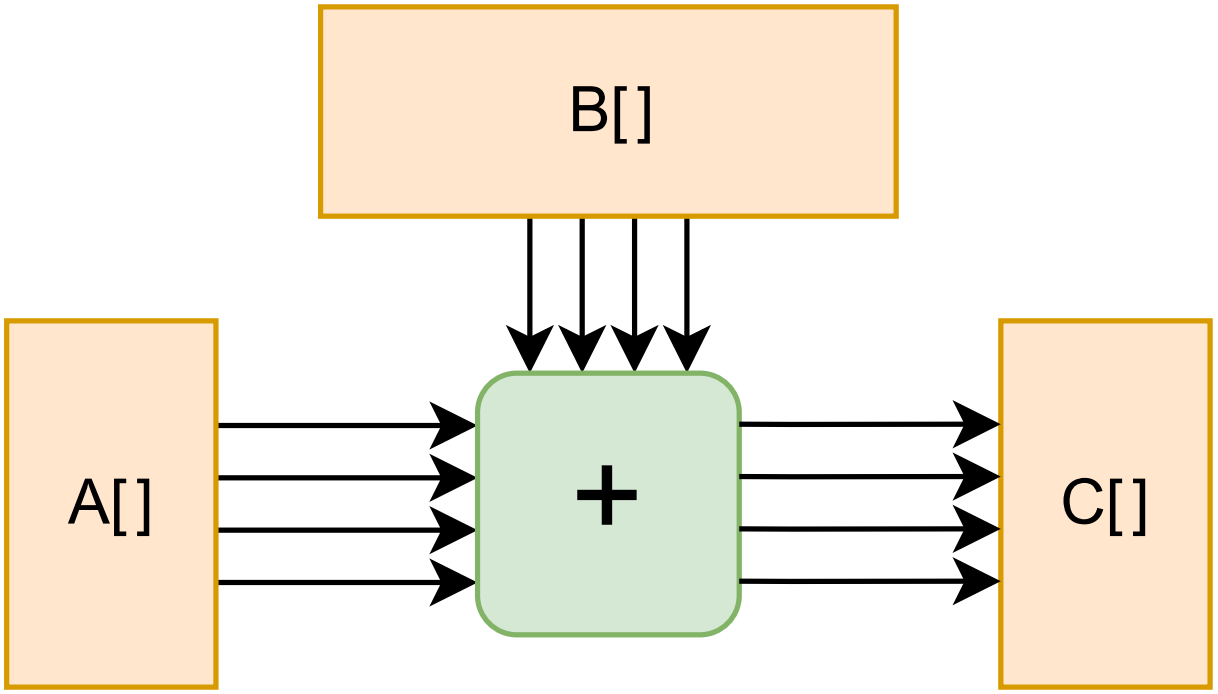
SIMD

- **S**ingle **R**egister **M**ultiple **D**ata
- *SIMD-Register*: kurze Arrays von Zahlen
- übliche Länge: 16, 32, 64 Byte
- *gleiche Operation* auf allen Elementen
- Laufzeit: genauso teuer wie Einzeloperation
- bei 16 Bytes: 16 mal so schnell wie einzeln!

Skalar vs. SIMD



Skalar vs. SIMD



Typische SIMD-Operationen

Arithmetik

- Addition, Subtraktion, Multiplikation, ...

Logik

- Vergleiche (elementweise), und, oder, xor, ...

Datentransfer

- lesen, schreiben, Masken extrahieren, ...

und vieles mehr

Strings und SIMD

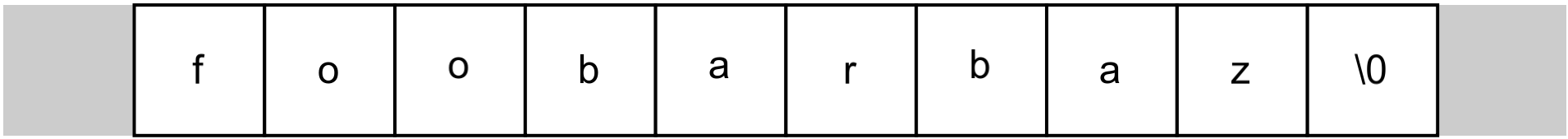
Was hilft uns das bei Strings?

Strings und SIMD

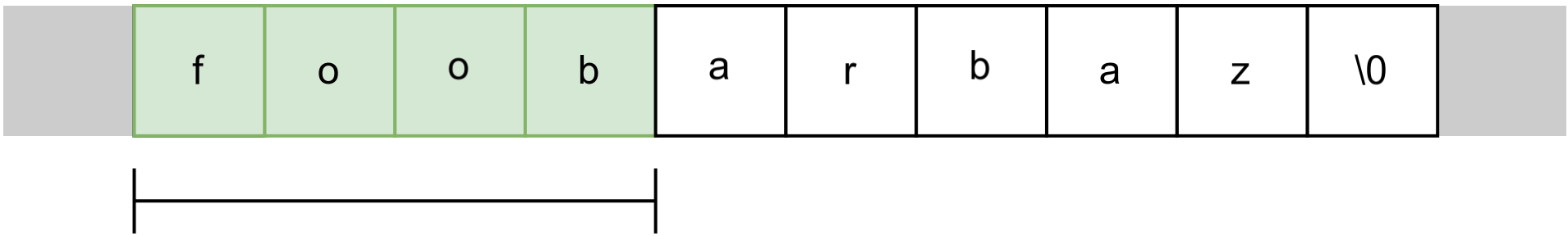
Was hilft uns das bei Strings?

- mehrere Zeichen auf einmal laden
- alle Zeichen gleichzeitig bearbeiten
- ...
- Profit?

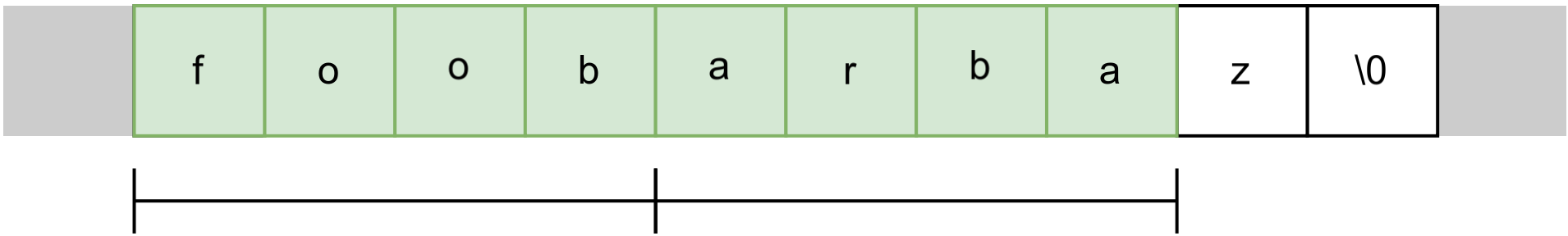
Schwierigkeiten



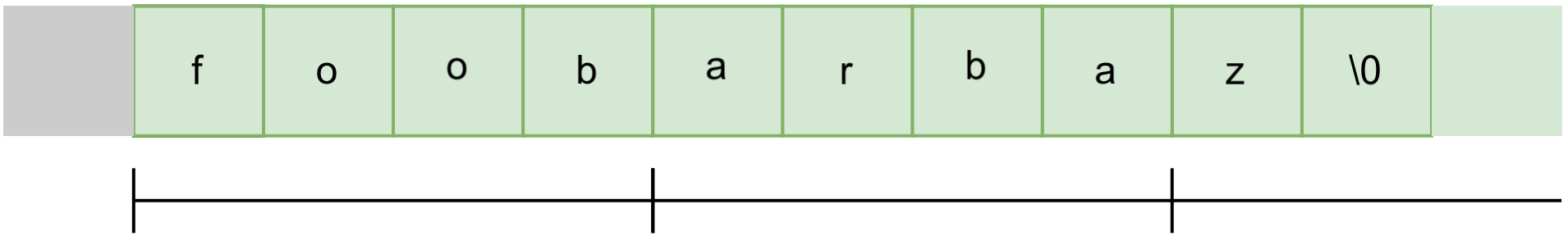
Schwierigkeiten



Schwierigkeiten



Schwierigkeiten



Schwierigkeiten

- Wenn wir nicht aufpassen, überschießen wir das Ende des Strings
- Aber bei nullterminierten Strings wissen wir nicht, wo es ist, bevor wir es gefunden haben
- Müssen wir das Ende Zeichen für Zeichen suchen?

Was können wir tun?

Was können wir tun?

String-Grenzen sind eine Fiktion

Was können wir tun?

String-Grenzen sind eine Fiktion
Lasst sie uns überwinden!

Array-Grenzen überwinden

- der Computer weiß nicht, was ein Array ist
- für ihn gibt es nur Speicher, der da ist oder nicht

Array-Grenzen überwinden

- der Computer weiß nicht, was ein Array ist
- für ihn gibt es nur Speicher, der da ist oder nicht

Ergo:

- solange wir nicht zu weit gehen passiert nichts!

Wie weit ist zu weit?

- Speicher wird in *Seiten* verwaltet
- Größe: je nach System, üblich sind 4096 Byte
- jede Seite ist entweder ganz oder garnicht zugreifbar
- feineren Speicherschutz gibt es nicht

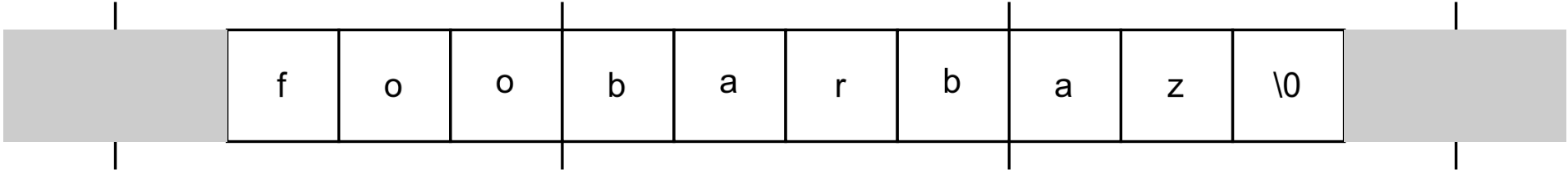
Wie weit ist zu weit?

- ist mindestens ein Byte vom String auf der Seite, dann ist die ganze Seite zugreifbar
- ausgerichtete Zugriffe überqueren nie Seitengrenzen

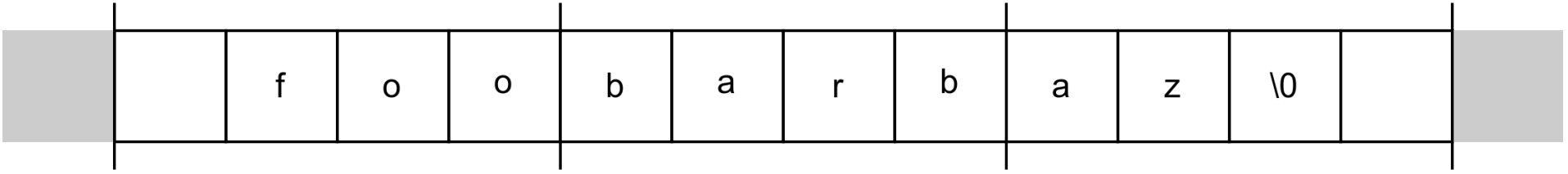
Ergo:

- wenn wir vorsichtig sind, klappt es!

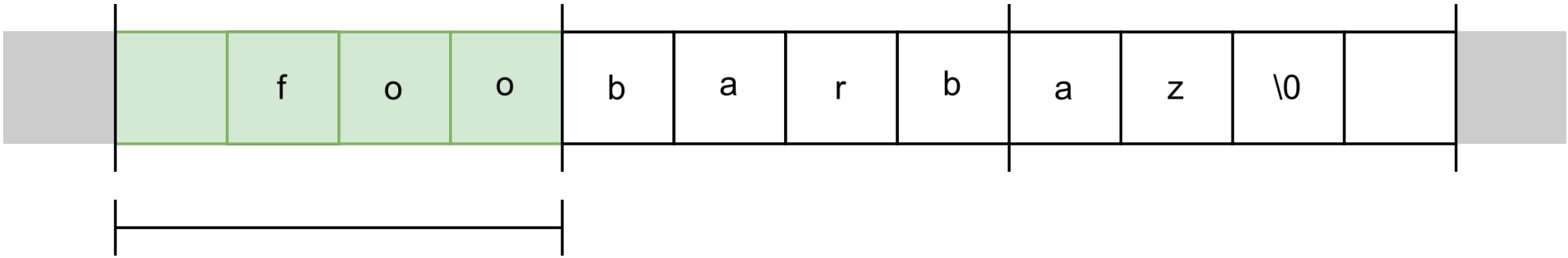
Wie sieht das dann aus?



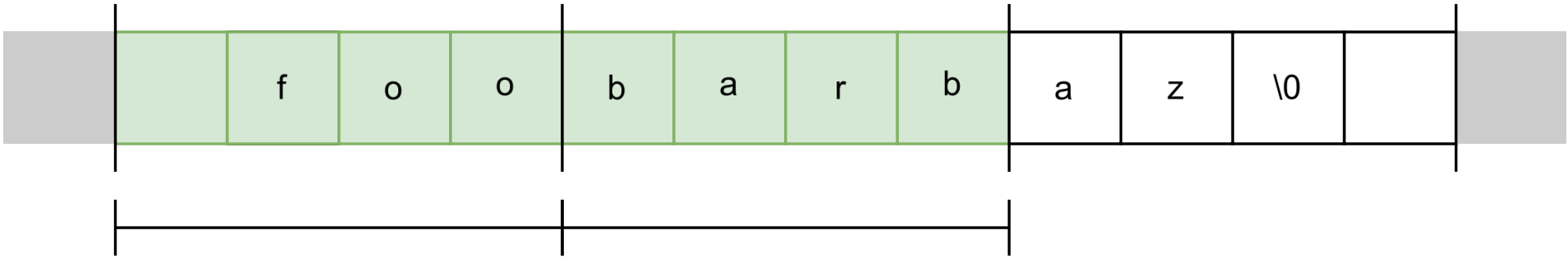
Wie sieht das dann aus?



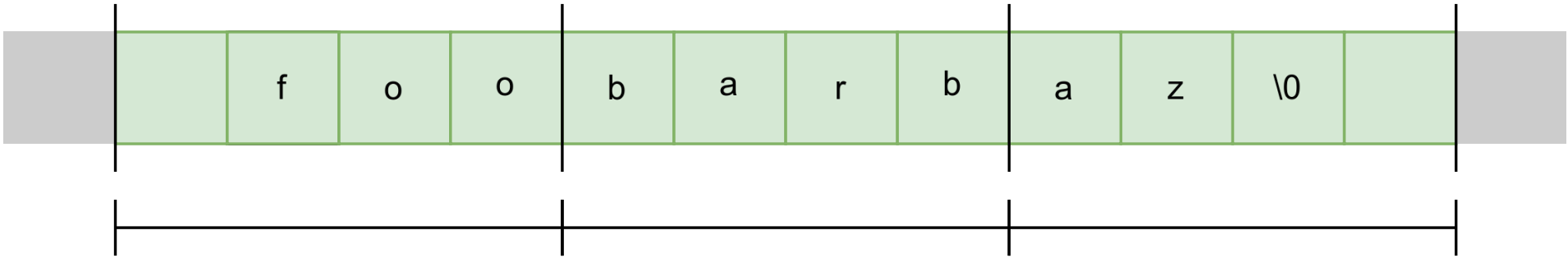
Wie sieht das dann aus?



Wie sieht das dann aus?



Wie sieht das dann aus?



Weitere Probleme

Strings rausschreiben

- Überlesen ist ok, überschreiben eher nicht

Weitere Probleme

Strings rausschreiben

- Überlesen ist ok, überschreiben eher nicht
- String in Chunks schreiben
- letzter Schreibzugriff überlappt vorletztem
- bei kurzen Strings kleinere Writes

Weitere Probleme

Strings vergleichen

- die Strings müssen nicht die gleiche Ausrichtung haben
- Ansatz von oben ist schwierig
- es gibt Tricks (kompliziert)

Weitere Probleme

Vergleich mit Zeichenmenge

- spezielle Maschinenbefehle (*pcmpistrm*, *match*)
- sonst: Algorithmus von Muła / Langdale

  http://0x80.pl/articles/simd-byte-lookup.html

Weitere Probleme

- Stringsuche
  - der Teufel steckt im Detail
  - wip

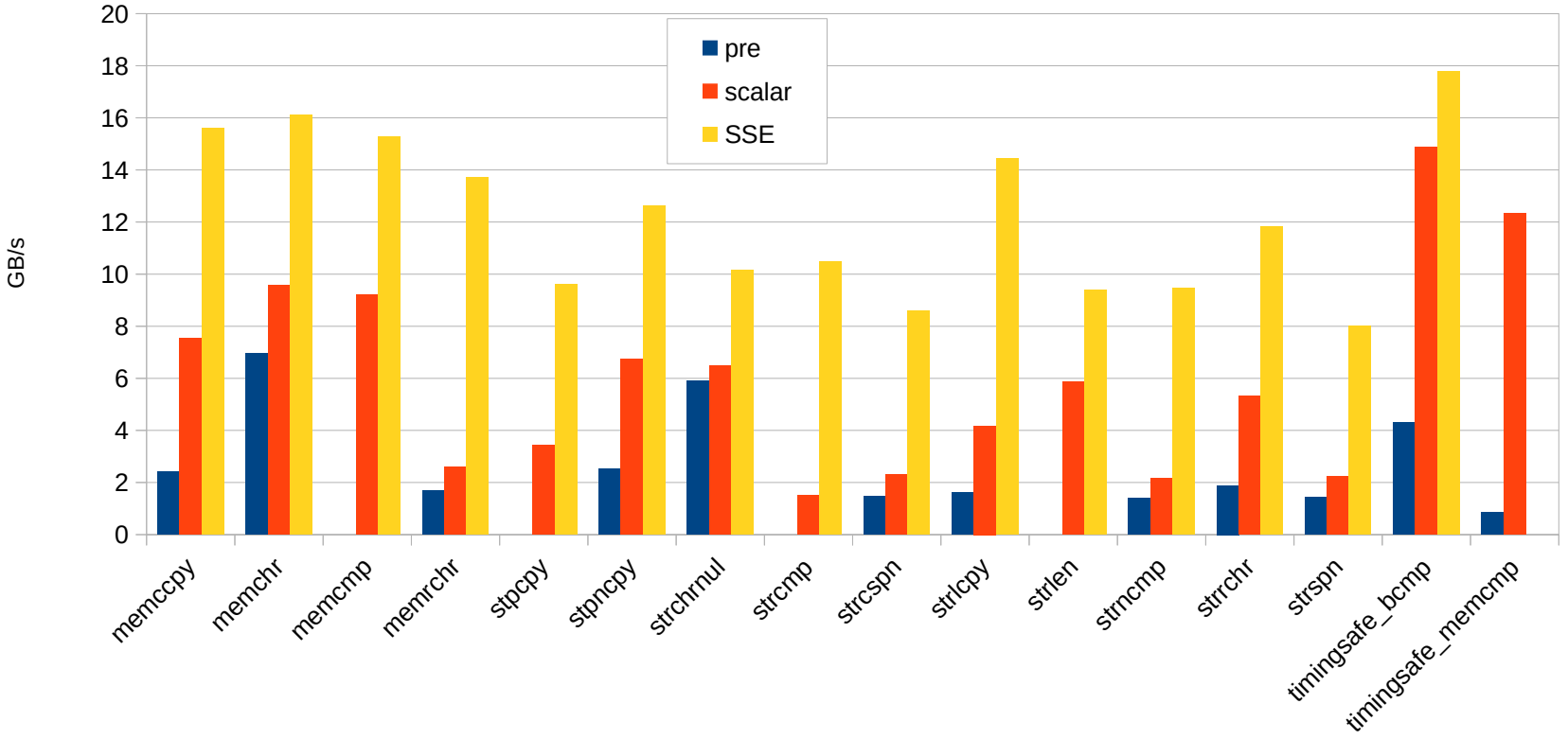
Praxiserfahrungen

- Überarbeitung der FreeBSD-libc für amd64 im Rahmen eines Auftrags der FreeBSD-Stiftung
- fast alle Funktionen aus <string.h>
- Ziel: amd64 baseline (SSE2)

Praxiserfahrungen

- Überarbeitung der FreeBSD-libc für amd64 im Rahmen eines Auftrags der FreeBSD-Stiftung
- fast alle Funktionen aus <string.h>
- Ziel: amd64 baseline (SSE2)
- durchschnittliche Verbesserung: Faktor 5,54

Ergebnisse


GB/s
20
18
16
14
12
10
8
6
4
2
0
pre
scalar
SSE
memccpy
memchr
memcmp
memrchr
stpcpy
stpncpy
strchrnul
strcmp
strcspn
strlcpy
strlen
strncmp
strrchr
strspn
timingsafe_bcmp
timingsafe_memcmp