Vorlesung 8

Aufgabe 1

Select one slide from the lecture, research more about the topic, and report on it.

Ich habe die Folie „Loop Unrolling" gewählt.

Jede Iteration braucht typischerweise 1) Vergleich der Schleifenbedingung 2) Sprung zur nächsten Iteration und 3) Erhöhung vom Schleifenindex. Das „kostet" CPU-Zyklen. Vor allem wenn Branch Prediction falsch ausgefällt, kann das viele Zyklen kosten.

Das Ziel vom Loop Unrolling ist diese Anzahl von Zyklen zu reduzieren. Dafür kann Loop Unrolling dem CPU durch Schleifen-„Entfaltung" genug Instruktionen geben, um seine Execution Units voll auszulasten und unabhängige Operationen parallel auszuführen.

Noch dazu kann man Loop Unrolling mit SIMD kombinieren.

ABER! Loop Unrolling kann auch nachteilhaft werden – mehr unrolled Instruktionen – mehr CPU-Register werden gebraucht, wenn mehr Register gebraucht werden – müssen Werte auf den Stack ausgelagert werden (Register Spilling). Deswegen muss man Unrolling Faktor beachten.

Aufgabe 2

What do the metrics latency and throughput tell you about the performance of an intrinsic function?

Latency (Latenz) = Wie lange dauert es bis Ergebnis (einer Instruktion) berechnet wird? (In Taktzyklen)
Niedrige Latenz => bessere Performance, Ergebnis wird schnell geliefert. Und hohe Latenz kann die Performance verschlechtern, wenn die nächste Instruktion noch auf das Ergebnis warten muss.
Aber alleine sagt Latenz nicht alles aus!

Throughput (Durchsatz) = Wie oft kann eine Instruktion pro Taktzyklus ausgeführt werden?
Hoher Durchsatz => viele Berechnungen können parallel verarbeitet werden.

Aufgabe 3

Read the paper
Discuss two things you find particularly interesting

1. Ich fand interessant, dass bei einer wenig gefüllten Hash-Tabelle „klassischer“ Code sogar schneller sein kann, denn Vektorregister nur dann helfen, wenn es genug parallele Arbeit gibt. Wenn der load factor niedrig ist (z.B. 25%), dann gibt es kaum Kollisionen und die Suche ist so kurz, dass Skalarregistern schneller sind als SIMD.

2. *„For LF 70 % and with LSBLSB fingerprints, we compare 17.23 fingerprints per find on average. During these probes, we encounter 0.53 collisions per find operation. If we switch to LS-BMSB, we encounter only 0.03 collisions per find operation on average.“*

..also sprich wenn man statt nur Fingerprints aus den niederwertigsten Bits zu nehmen, sie mit höchstwertigen Bits kombiniert, reduziert sich die Anzahl der Kollisionen extrem. Ich fand interessant, wie doll die Performance nur durch Wahl des cleveren Algorithmus verbessert werden kann.