Praktikum: Paralleles Programmieren

5. SIMD-Parallelisierung

Nathanael Hübbe

Wissenschaftliches Rechnen
Fachbereich Informatik
Universität Hamburg

10-5-2012

informatik
die zukunft

Flynn’s Klassifikation

SISD	MISD
SIMD	MIMD

Einige SIMD-Implementationen

- Vektorprozessoren (Cray)

- MMX (Wiederverwendung der Fließkommaregister)
- SSE (8x 128 Bit)
- SSE2 (16x 256 Bit)
- SSE3 (horizontal Instructions)
- SSSE3 (2006: Ganzvektorshift/-rotate, allgemeine Permutation)

- AltiVec (32x 128 Bit)
- VSX (64x 128 Bit)

Vektorregister

- Seit SSE/AltiVec haben SIMD-Einheiten eigene Register.
- Breite der Register bestimmt den Grad an Parallelität.
  - SSE/AltiVec: 128 Bit, SSE2: 256 Bit
- Jeder Befehl verarbeitet Teile einer bestimmten Länge parallel.
  - z. B. AltiVec: 128 Bit = 16 Bytes, 8 Halfwords oder 4 Words
  - add r3,r4,r5 ≙ vaddsbs/vaddshs/vaddsws v3,v4,v5

Befehlsklassen

- Arithmetik
  - Fixpunkt und Gleitkomma
  - Grundrechenarten
  - Zusätzlich horizontale Befehle z. B. vmsum
  - Vergleiche produzieren bool-Vektoren

Befehlsklassen

- Permutationen
  - Meist Byteweise
  - z. B. Shifts, rotates & splats
  - Allgemeine Permutation:
    Ein Register liefert Indizes (vperm)
  - Selektion:
    Die Bits eines Registers bestimmen, von welchem Register das entsprechende Ausgabebit kommt.

Befehlsklassen

- Cacheanweisungen
  - Vektoreinheiten sollen große Datenmengen schnell verarbeiten
    - $\Rightarrow$ Bus wird noch leichter zum Flaschenhals
  - Daher: Spezielle Instruktionen um
    - Prefetching zu triggern
    - Daten als nicht mehr benötigt zu markieren

Beispiel: Eine Funktion, die den Index des Maximums in einem Array positiver 32-Bit Werte bestimmt.

- Vorarbeiten

```
argmax:
    srawi. r5,r4,2 #divide by four and compare against zero
    mtctr r5
    vspltisv v0,0 #=(0,0,0,0)
    vspltisv v1,1 #=(1,1,1,1)
    vspltisv v2,2 #=(2,2,2,2)
    vspltisv v3,3 #=(3,3,3,3)
   vmrghw v0,v0,v2 #=(0,2,0,2)
   vmrghw v1,v1,v3 #=(1,3,1,3)
   vmrghw v0,v0,v1 #=(0,1,2,3)
    vspltisv v1,4 #=(4,4,4,4)
    vspltisv v2,0 #clear it
    vspltisv v3,0 #clear it
    beq endloop #in case there is nothing to do
```

informatik
die zukunft

Beispiel: Die Hauptschleife

```
loop:
    lvx v4,0,r3 #get four entries
    addi r3,r3,16 #point to the next four

    vcmpgtuw v5,v3,v4 #compare against the old max
    vmaxuw v3,v3,v4 #update the max
    vsel v2,v0,v2,v5 #remember the indices of the maxs

    vadduws v0,v0,v1 #increment the indices
    bdnz loop
endloop:
```

Beispiel: Die Nachbereitung

```
#Fuse the four results together
vsldoi v4,v3,v3,8 #rotate by two words
vsldoi v0,v2,v2,8
vcmpgtuw v5,v3,v4 #compare against the unshifted
vmaxuw v3,v3,v4 #update the max
vsel v2,v0,v2,v5 #remember the indices of the maxs

vsldoi v4,v3,v3,4 #rotate by one word
vsldoi v0,v2,v2,4
vcmpgtuw v5,v3,v4 #compare against the unshifted
vsel v2,v0,v2,v5 #remember the indices of the maxs

#Now all the words in v2 contain the result index,
#return it in r3.
stvewx v2,-4(sp)
lwz r3,-4(sp)
blr
```

informatik
die zukunft

Programmierung mit Compilerintrinsics

- Erlaubt die Verwendung von Vektoreinheiten aus Hochsprachen (also C) heraus.
- Aber:
  - 1 Assemblerbefehl $\rightarrow$ 1 Funktionsaufruf
  - Nicht Architekturunabhängig!
  - Vektordatentypen verkomplizieren die Programmierung
- $\Rightarrow$ inline asm-Statements sind im allgemeinen lesbarer!

Bewertung

- Gut für rechenintensive Aufgaben (wenn die skalare Version den Bus nicht auslasten kann)
- Parallelität ohne Nebenläufigkeitsproblematiken
- Caching wird beeinflussbar
- Kein Vorteil, wenn der Bus ohnehin der Flaschenhals ist
  - Aber die Cache Anweisungen können dann helfen
- Parallelität durch Architektur begrenzt
- Feste Registerbreite bedeutet eine Menge Mehrarbeit
- Riesige Registerfiles verteuern Kontextwechsel