Division durch (Quasi-)Konstanten

M. Anton Ertl
TU Wien

Division und Multiplikation

- Divisionsbefehle sind teuer
  Ganzzahlige Division auf Skylake:
  Latenz: 35–42 Zyklen
  Durchsatz: 1 Division pro 21–24 Zyklen

- Multiplikation ist billiger
  Ganzzahlige Multiplikation auf Skylake:
  Latenz: 4 Zyklen
  Durchsatz: 1 Multiplikation pro Zyklus

- $n/d \to n(1/d)$
  für konstante d
  für schleifeninvariante d

- ganze Zahlen richtig runden

Zweistufige Division: Benutzung

```
staged/-size ( -- u )
u/-stage1m   ( u1 addr -- )
u/-stage2m   ( u2 addr -- u2/u1 )

: array/ ( addr u u1 -- )
  {: | reci[ staged/-size ] :}
  reci[ u/-stage1m
  cells bounds u+do
    i @ reci[ u/-stage2m i !
  1 cells +loop ;
```

Zweistufige Division: Implementierung

```
\ http://git.savannah.gnu.org/cgit/gforth.git/tree/stagediv.fs

: u/-stage1m {: udivisor addr -- :}
    udivisor 2 u< -24 and throw
    udivisor addr staged/-divisor !
    0 1 udivisor um/mod addr staged/-inverse-hi !
    udivisor 1- swap udivisor um/mod addr staged/-inverse ! drop ;

\ als Primitive implementiert, hier die Forth-Variante
: u/-stage2m {: udividend addr -- uquotient :}
    udividend addr staged/-inverse @ um* nip 0
    udividend addr staged/-inverse-hi @ um* d+ nip ;
```

Zweistufige Division: Performance

auf Skylake (Core i5-6600K) in Zyklen

normal	zweist.	
	228.9	`u/stage1`
	158.9	`/fstage1`
41.3	16.0	`u/`
39.9	19.7	`umod`
44.0	25.2	`u/mod`
48.7	16.9	`/f`
47.9	20.5	`modf`
53.0	24.5	`/modf`

Division durch Konstante: Beispiele

```
: foo 3 u/ ;
see foo
: foo
  $7FA70F918640 u/-stage2m ; ok

: bar 4 u/ ;
see bar
: bar
  2 rshift ; ok
```

Division durch Konstante: Implementierung

```
: lit/, {: divisor xt: stage1 xt: stage2 -- :}
    next-section staged/-size small-allot previous-section {: addr :}
    divisor addr stage1 ]] addr stage2 [[ ;

: opt-u/ ( xt -- )
    lits# 1 = if
        lits> dup 0<> if
            dup pow2? if
                ctz ]] literal rshift [[ drop exit then
            ['] u/-stage1m ['] u/-stage2m lit/, drop exit then
        >lits then
    fold2-1 ;
' opt-u/ optimizes u/
```

Zusammenfassung

- Multiplikation viel billiger als Division
- Dividieren durch Multiplikation mit Reziprokwert sinnvoll bei quasi-konstanten Divisoren
- Zweistufige Division
- Optimierung der Division durch Konstante