Thema, Ziele: Computerarithmetik, Monte Carlo Integration

Aufgabe 1: Inverse Square Root beim TMS320C28x

Den Digital Signal Processor (DSP) *TMS320C28x* von TI gibt es als eigenständigen Baustein. Er wird beispielsweise ebenfalls im Dualcore-Controller Concerto eingesetzt. Der C28x ist ein Floating Point DSP. Der Instruktionssatz ist im Dokument ***TMS320C28x Floating Point Unit and Instruction Set*** beschrieben. Die Instruktion EINVF32 rechnet $\frac{1}{x}$ innerhalb 2 Zyklen ungefähr auf 8 Bit Genauigkeit.

a) Studieren Sie den Befehl `EINVF32`. Wie kann mit Hilfe dieses Befehls eine Genauigkeit von 32 Bits erreicht werden?

b) Die Instruktion `EISQRTF32` funktioniert ähnlich. Sie erzeugt ebenfalls eine Genauigkeit von ungefähr 8 Bits. Wie kann mit Hilfe dieser Instruktion die Wurzel einer Zahl x, d. h. $\sqrt{x}$ erzeugt werden?

c) Der Assemblercode für die Berechnung der Wurzel sieht gemäss dem Dokument ***TMS320C28x Floating Point Unit and Instruction Set*** ungefähr so aus:

```
; Y = sqrt(X)
; Ye = Estimate(1/sqrt(X));
; Ye = Ye*(1.5 - Ye*Ye*X*0.5)
; Ye = Ye*(1.5 - Ye*Ye*X*0.5)
; Y = X*Ye
_sqrt:
                                    ; R0H = X on entry
    EISQRTF32   R1H, R0H            ; R1H = Ye = Estimate(1/sqrt(X))
    MPYF32      R2H, R0H, #0.5      ; R2H = X*0.5
    MPYF32      R3H, R1H, R1H       ; R3H = Ye*Ye
    NOP
    MPYF32      R3H, R3H, R2H       ; R3H = Ye*Ye*X*0.5
    NOP
    SUBF32      R3H, #1.5, R3H      ; R3H = 1.5 - Ye*Ye*X*0.5
    NOP
    MPYF32      R1H, R1H, R3H       ; R2H = Ye = Ye*(1.5 - Ye*Ye*X*0.5)
    NOP
    MPYF32      R3H, R1H, R2H       ; R3H = Ye*X*0.5
    NOP
    MPYF32      R3H, R1H, R3H       ; R3H = Ye*Ye*X*0.5
    NOP
    SUBF32      R3H, #1.5, R3H      ; R3H = 1.5 - Ye*Ye*X*0.5
    CMPF32      R0H, #0.0           ; Check if X == 0
    MPYF32      R1H, R1H, R3H       ; R2H = Ye = Ye*(1.5 - Ye*Ye*X*0.5)
    NOP
    MOV32       R1H, R0H, EQ        ; If X is zero, change the Ye estimate to 0
    MPYF32      R0H, R0H, R1H       ; R0H = Y = X*Ye = sqrt(X)
    LRETR
```

Die eingeflochtenen `NOP`-Instruktionen sind aufgrund der Befehlspipeline notwendig. Wenn bei einer Instruktion, die 2 Zyklen benötigt (z.B. `MPYF32`) die folgende Instruktion dasselbe Zielregister verwendet, dann muss ein `NOP` eingeflochten werden.

Wie viele Clockzyklen werden für die Berechnung der Wurzel mit dieser Methode verwendet? Zeichnen Sie dazu die zweistufige Befehlspipeline auf.

Aufgabe 2: Monte Carlo Integration

Eine spezielle Simulationsanwendung ist die Monte Carlo Integration. Die Berechnung des Integrals wird mit Hilfe eines Zufallszahlengenerators durchgeführt. In dieser Aufgabe sollen Sie die folgende Funktion von 1.0 bis 1.4 integrieren:

$$f(x) = \sqrt{(x-1)\cdot(2-x)}\cdot e^{-x^2}$$

d.h. den folgenden Integral

$$\int_{1.0}^{1.4} \sqrt{(x-1)\cdot(2-x)}\cdot e^{-x^2} dx$$

Für dieses Integral kann kaum eine geschlossene Lösung gefunden werden. Den Graphen der Funktion f(x) sehen Sie in Abbildung 1.

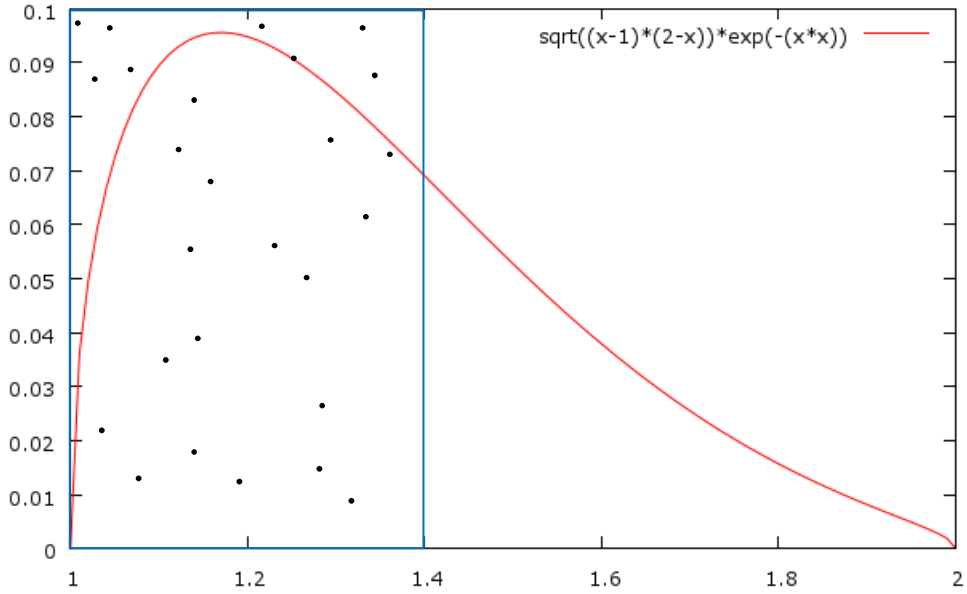

Abbildung 1: Funktionsgraph von f(x)

Aus dem Graphen ist ersichtlich, dass alle y=f(x) im betrachteten Intervall unterhalb von 0.1 liegen. Wenn nun ein Rechteck mit den Eckpunkten (1.0, 0.0) und (1.4, 0.1) angenommen wird, so kann dessen Fläche einfach berechnet werden. Ein Zufallszahlengenerator bestimmt nun zufällige Punkte (x, y) innerhalb des besagten Rechtecks. Für ein bestimmtes x kann einfach der Wert f(x) berechnet werden. Wenn der zuvor bestimmte zufällige Wert y grösser als f(x) ist, dann liegt der Punkt oberhalb der Kurve, sonst unterhalb. Bei vielen zufälligen Punkten legt das Verhältnis der Anzahl Punkte unterhalb der Kurve zur Anzahl der Punkte oberhalb der Kurve gerade das Verhältnis der beiden Teilflächen fest. So kann das Integral näherungsweise bestimmt werden.

a) Untersuchen Sie den Graphen mit Hilfe eines geeigneten Tools (z.B. Gnuplot, Matlab).

b) Implementieren Sie die Monte Carlo Integration. Beenden Sie die Integration, sobald die Änderung von einem zum nächsten Schritt eine vorgängig bestimmte Grenze (z.B. 10^{-5}) unterschreitet. Zählen Sie, wie viele Iterationsschritte dazu notwendig waren. Verwenden Sie den folgenden Funktionsprototypen:

```
// integrates the function f within [lb,hb], stops as soon as change <= epsilon
double monteCarlo(double(*f)(double x), // ptr to function to integrate
                  double lb, // lower bound
                  double hb, // higher bound
                  double height, // must be higher than max of f(x) within [lb,hb]
                  double epsilon = 1e-5, // minimal change
                  unsigned long minIterations = 10000); // minimal number of iterations
```

c) Mit der Monte Carlo Methode kann auch die Zahl π angenähert werden. Skizzieren Sie, wie Sie dabei vorgehen würden.