5.5 Just-In-Time-Compiler

Wie Sie vielleicht wissen, gab es in Gambas eine JIT-Komponente (JustInTime-Komponente), die auf LLVM (LowLevelVirtualMachine) - einer Programmbibliothek für den Bau von hoch optimierten Compilern, Optimierern und Laufzeitumgebungen - basierte. Der alte JIT-Compiler funktionierte nicht mehr, da neuere Versionen der LLVM nicht abwärtskompatibel sind. Daher wurde vom Minisini entschieden, einen neuen JIT-Compiler von Grund auf neu zu schreiben. Dieser übersetzt den Gambas-Code in C.

Der neue JIT-Compiler ist ab Version 3.12.0 in einer Komponente gb.jit implementiert, die bei Bedarf automatisch geladen wird. Es hat einen C-Teil, der den GambasToC_Code-Translator implementiert und einen Gambas-Teil, der sich mit der Extraktion von Gambas-Quellcode aus ausführbaren Archivdateien und der Zusammenstellung aller Übersetzungen in einer großen Datei beschäftigt.

So funktioniert der Gambas JIT-Compiler:

- Beim ersten Aufruf einer mit dem FAST-Schlüsselwort markierten Funktion wird der JIT-Compiler aufgerufen.
- Wenn noch nicht geschehen, übersetzt der JIT-Compiler alle schnell auszuführenden Methoden des aktuellen Projekts (oder der Komponente) in eine große C-Quelldatei.
- Dann wird der C-Compiler aufgerufen, um aus dieser großen C-Quelldatei eine gemeinsame Bibliothek zu erzeugen.
- Die gemeinsame Bibliothek wird dynamisch geladen.
- Die jit-kompilierte Funktion wird aufgerufen.

Diese Konstruktion hat Vorteile:

- Sie können entweder den C-Compiler gcc oder Clang (https://de.wikipedia.org/wiki/Clang) verwenden.
- Solange sich der C-Standard nicht ändert, wird der JIT-Compiler ordentlich funktionieren.

mit dem Nachteil, dass die Code-Generierung langsamer (und mit gcc noch langsamer) gegenüber dem alten JIT-Compiler ist.

5.5.1 Syntax

Die Syntax mit FAST hat sich nicht geändert: Setzen Sie das Schlüsselwort FAST an den Anfang einer Gambas-Klassendatei für eine Klasse, die alle Funktionen JIT-kompiliert statt interpretiert bereitstellen soll oder an den Anfang einer Funktionsdeklaration, so dass nur diese Funktion jit-kompiliert wird.

Es wurde mit UNSAFE ein neues Schlüsselwort eingeführt, das nur zusammen mit dem Schlüsselwort FAST verwendet wird. Wenn eine Funktion als "FAST UNSAFE" deklariert wird, dann wird sie just-in-time kompiliert, aber alle Sicherheitsüberprüfungen werden entfernt (Nullobjekt, Out-of-Array-Bounds, Division durch Null, ...)! Es erlaubt Ihnen, den Code ein wenig zu beschleunigen, wenn Sie sicher sind, dass er korrekt deklariert ist. Ansonsten erhalten Sie einen Speicher- und/oder einen Segmentierungsfehler.

5.5.2 Hinweise

- Versuchen Sie, den Einsatz des JIT-Compilers nur auf bestimmte Methoden anzuwenden, die einen Leistungsschub benötigen.
- Die größten Geschwindigkeitssteigerungen werden bei Funktionen auftreten, die viele Low-Level-Berechnungen und Kontrollabläufe verwenden, wie zum Beispiel Funktionen, die viele mathematische Operationen einsetzen.
- Wenn die Funktion jedoch hauptsächlich andere Bibliotheken aufruft, werden Sie keine große Geschwindigkeitssteigerung feststellen.
- Sie werden keinen Gewinn in der Zeichenketten-Bearbeitung erhalten.

5.5.3 Debugging

Diese Umgebungsvariablen GB_NO_JIT, GB_JIT_DEBUG, GB_JIT_CC und GB_JIT_CFLAGS steuern in der Gambas-Version 3.12.0 das Verhalten des JIT-Compilers. Dabei gilt:

- GB_NO_JIT: Setzen Sie den Wert auf 1, wenn Sie den JIT-Compiler vollständig deaktivieren möchten.
- GB_JIT_DEBUG: Setzen Sie den Wert auf 1, wenn Sie die Debugging-Meldungen des JIT-Compilers sehen möchten.
- GB_JIT_CC: Setzt den Namen des zu verwendenden Compilers. Standardmäßig wird gcc verwendet. Aber Sie können stattdessen Clang setzen oder andere C-Compiler – vorausgesetzt, dass er die gleichen Optionen wie gcc verwendet oder die Unterstützung dafür im JIT-Compiler Gambas-Teil hinzugefügt wird.
- GB_JIT_CFLAGS: Setzt den JIT-Compiler auf die Flags, die für die Kompilierung des C-Codes verwendet werden. Der Standard-Wert ist -O3.

5.5.4 Software-Test

Wenn Sie ein stabiles Gambas und den JIT-Compiler in Ihren Programmen einsetzen, dann interessiert Sie sicher, welchen Vorteil das für Ihre Programme hat. Getestete Programm-Laufzeiten finden Sie auf http://gambaswiki.org/wiki/doc/benchmark. Bei dem folgenden adaptierten Gambas-Skript wurden sehr unterschiedliche Laufzeiten erzielt:

```
[1]  #!/usr/bin/env gbs3
[2]
[3]  FAST ' Change it!
[4]  Public Sub Main()
[5]    Dim I As Integer
[6]    Dim StartTime As Float
[7]
[8]    StartTime = Timer
[9]    For I = 1 To 2 ' Runs
[10]     Print I & ". Run: " & Test(0.2) & " basic calculations"
[11]   Next
[12]   Print "-----------------------------------"
[13]   Print "Programm-Laufzeit = "; GetTime(StartTime, Timer); " Sekunden"
[14] End
[15]
[16] Private Function Test(X As Float) As Float
[17]
[18]   Dim Mu As Float = 10.0
[19]   Dim Pu, Su As Float
[20]   Dim I, J, N As Integer
[21]   Dim aPoly As New Float[100]
[22]
[23]   N = 500000
[24]   For I = 0 To N - 1
[25]     For J = 0 To 99
[26]       Mu = (Mu + 2.0) / 2.0
[27]       aPoly[J] = Mu
[28]     Next
[29]     Su = 0.0
[30]     For J = 0 To 99
[31]       Su = X * Su + aPoly[J]
[32]     Next
[33]     Pu += Su
[34]   Next
[35]   Return Pu
[36] End
[37]
[38] Private Function GetTime(StartTime As Float, EndTime As Float) As Float
[39]   Return Round(EndTime - StartTime, -3)
[40] End
```

Der Aufruf des Gambas-Skripts – gespeichert in der Datei jit_p.gbs3 – erfolgt in der Konsole:

```
hans@mint-183 ~ $ gbs3 ./jit_p.gbs3
1. Run: 1250000 basic calculations
2. Run: 1250000 basic calculations
-----------------------------------
Programm-Laufzeit = 0,755 Sekunden
```

```
hans@mint-183 ~ $ gbs3 ./jit_p.gbs3
1. Run: 1250000 basic calculations
2. Run: 1250000 basic calculations
-----------------------------------
Programm-Laufzeit = 16,688 Sekunden
```

Mit dem Schlüsselwort *FAST* ergaben sich beim Einsatz des JIT-Compilers nur 0,755 Sekunden, was einem Laufzeit-Verhältnis von etwa 22*:1* zwischen Gambas und Gambas mit JIT-Compiler entspricht!

Sie können aber auch die Schlüsselwörter 'FAST UNSAFE' einsetzen, indem Sie die obere Zeile 3 komplett löschen und die Zeile 16 so umschreiben:

```
[16] FAST UNSAFE Private Function Test(X As Float) As Float
```

Dann wird nur diese Funktion jit-kompiliert! In einem weiteren Test zeigten sich erwartungsgemäß bei den Laufzeiten die gleichen Ergebnisse.