AUFGABE 5: STATISCHE ANALYSE NICHT-FUNKTIONALER EIGENSCHAFTEN

In dieser Aufgabe sollen Sie die Analyse-Werkzeuge StackAnalyzer und aiT aus der schon bekannten a3 Analyse-Suite von AbsInth nutzen, um den Stackverbrauch und die maximale Laufzeit von Funktionen aus den vorhergehenden Aufgaben zu bestimmen.

Auch bei dieser Aufgabe ist der Arbeitsaufwand wieder relativ klein, weswegen nur eine Woche für die Bearbeitung vorgesehen ist.

Aufgabenstellung

1. *Compiler:* Im Unterschied zu Astree verarbeiten StackAnalyzer und aiT keinen C-Quellcode, sondern ausführbare Programme. Leider stehen uns beide Werkzeuge nicht für die Intel-Architektur zur Verfügung sondern nur für die Tricore-Prozessor-Familie. Deswegen müssen Sie für diese Aufgabe einen anderen Compiler verwenden. Dies erreichen Sie, indem Sie für den Aufruf von `cmake` die `CC`- und `CXX`-Umgebungsvariablen entsprechend anpassen:

```
CC=/proj/i4ciao/tools/trigcc346-sijesche/bin/tricore-gcc
CXX=/proj/i4ciao/tools/trigcc346-sijesche/bin/tricore-g++
cmake ....
```

2. *Annotationen:* Machen Sie sich mit den zur Verfügung gestellten Handbuch vertraut. Verschaffen Sie sich einen groben Überblick über die verfügbaren Annotationen.

3. *Konfiguration:* Starten sie die Werkzeug-Suite mit dem Befehl `/proj/i4ezs/tools/a3_tricore-b240070/bin/a3tricore`. Falls Sie nach einer Lizenzdatei gefragt werden, geben Sie folgenden Pfad an:

```
/proj/i4ezs/tools/a3_tricoreb240070/share/a3_tricore/share/license.dat
```

Laden Sie im *Configuration*-Panel *TriCore* die im Vorgabeverzeichnis befindliche Maschinenbeschreibungsdatei `TC1796.msf`. Machen Sie sich mit den verfügbaren Konfigurationsoptionen vertraut und wenden Sie diejenigen an, von denen Sie glauben, dass sie notwendig sind oder Ihr Analyseergebnis positiv beeinflussen. Beachten Sie insbesondere, dass AIS-Annotationen im Quellcode standardmäßig nicht von der Analyse beachtet werden. Die kann durch entsprechende Konfiguration aber geändert werden. Speichern Sie Ihr Projekt in Ihrem git-Repository ab und stellen Sie es unter Versionsverwaltung.

4. *Prioritätswarteschlange:* Kompilieren Sie nun die in Aufgabe 3 von Ihnen geschriebene Filteranwendung für den Tricore. Falls *nicht kritische* Teile wie z.B. Sensorstubs nicht kompilierbar sind, kommentieren Sie diese bitte aus. Analysieren Sie nun sowohl den maximalen Stackverbrauch als auch die maximale Laufzeit der `main()` Funktion und interessanter Module wie z.B. die Filterfunktion. *Woran scheitern die Analysen? Wie können Sie den Werkzeugen dabei helfen das Programm trotzdem zu analysieren? Notieren Sie sich die nötigen Anpassungen der Implementation oder Annotationen. Vergleichen Sie die Ergebnisse für unterschiedliche Compileroptimierungen. Gerade* `-O0` *und* `-Os` *sollten hier interessant sein.*

5. *Rekursion* Gehen Sie für die Funktionen aus Aufgabe 4 analog vor. Vergleichen Sie insbesondere den analysierten Stackverbrauch mit dem von Ihnen gemessenen. *Sind beide Werte aufgrund der unterschiedlichen Architekturen vergleichbar? Wie verhält sich Ihre Messung im Verhältnis? Welche Anpassungen und Annotationen sind hier nötig?*

Hinweise

- Bearbeitung: Gruppe mit je zwei bis drei Teilnehmern.
- Abgabezeit: die Woche vom 22.06.2015 bis zum 29.06.2015

- Fragen bitte an i4ezs@lists.informatik.uni-erlangen.de