AUFGABE 2: HALLO ZEIT

In dieser Übungsaufgaben werden Sie sich mit verschiedenen Möglichkeiten befassen, maximale Ausführungszeiten abzuschätzen. Ziel dieser Aufgabe ist es, ein Gefühl für die Grenzen dieser verschiedenen Herangehensweisen zu bekommen.

Aufgabenstellung

1. *Zeitmessung mit der libEZS:* Um die zeitlichen Abläufe im System messen zu können, muss zunächst die `libEZS` erweitert werden. Einen hardwareunabhängigen Zähler haben wir bereits vorgegeben. Sie können auf diesen mittels der Funktion `ezs_counter_get()` zugreifen und den aktuellen Wert in Ticks auslesen. *Implementieren Sie nun die Funktionen* `ezs_watch_start()` *und* `ezs_watch_stop()` *in* `ezs_stopwatch.c`. Damit mehrere Zeitmessungen parallel und über Funktionsgrenzen hinweg erfolgen können, erhalten beide Funktionen einen Zeiger auf den Zustand der Messung durch den Aufrufer. Die Zustandsvariable muss in der Anwendung später global[1] angelegt werden. Die Funktion `ezs_watch_stop()` liefert das Ergebnis der Messung in Form von Zeitgeber-Ticks zurück. Die Dauer eines solchen Ticks ist prinzipiell hardwareabhängig, der von uns bereitgestellte Zähler hat ein Raster von 13,3 ns.

☞ `libEZS/...`

☞ `ezs_counter.h`

Wir haben für Sie bereits die Funktionen `heapsort()` und `quicksort()` implementiert. *Von welchen Faktoren ist die Ausführungszeit dieser Funktionen abhängig?* Verwenden Sie die von Ihnen implementierte Stopuhr um die *Worst Case Execution Time (WCET)* beider Funktionen abzuschätzen. Visualisieren Sie sich hierbei auch das Histogramm der gemessenen Zeitwerte. Die Zeitwerte können Sie hierzu auf die serielle Konsole ausgeben. Wenn Sie sich diese mit Hilfe von `cutecom` anschauen, können Sie die Messwerte direkt in eine Datei speichern. Diese Daten können dann beispielsweise in `qtiplot` plotten lassen (rechte Maustaste auf den Kopf der Spalte mit den Werten, Menüpunkt „Plot“, „Statistical Graphs“, „Histogram“).

☞ `print_measurement()`

Versuchen Sie eine möglichst lange Laufzeit der Funktion `heapsort()` zu erzwingen: Die Gruppe, die die längste Laufzeit für ein Eingabearray der Länge 1024 demonstrieren kann, erhält eine Belohnung.
Wie groß sind der Mittelwert und dessen Standardfehler bei dieser Eingabe? Wie groß ist die Spanne zwischen gemessenem Minimal- und Maximalwert? Welche Bedeutung haben diese Wert im Bezug auf die WCET?

2. *Zeitmessung mit dem Oszilloskop:* Häufig steht auf in Echtzeitsystemen verwendeter Hardware kein Zeitgeber zur Verfügung, der so genau arbeitet und ein

[1] Gültigkeit von Variablen in C, Folie 37ff: `http://www4.cs.fau.de/Lehre/SS13/V_SP1/Folien/SP-02a-A4.pdf`

so feines Raster hat wie der des Tricores. Aus diesem Grund sollen Sie in dieser Teilaufgabe lernen Ausführungszeiten mit Hilfe des Oszilloskops zu ermitteln.
Die Funktion `ezs_gpio_set()` erlaubt es Ihnen den auf der Experimentierplatine herausgeführten Pin des Tricore als GPIO zu verwenden. *Wie können Sie mit Hilfe des GPIO Ausführungszeiten messen? Welchen Ausgang der Experimentierplatine müssen Sie hierzu abgreifen?* Führen Sie nun die Messungen der vorangegangenen Teilaufgabe noch einmal mit dem Oszilloskop durch. *Wie genau stimmen diese mit den Messungen Ihrer Stopuhr überein?*
Welche Probleme sehen Sie sowohl bei der WCET-Abschätzung durch den Zeitgeber als auch bei der durch das Oszilloskop?

3. *Werkzeuggestützte WCET-Ermittlung:* In dieser Teilaufgabe sollen Sie die WCET mit Hilfe eines Werkzeugs zur Codeanalyse bestimmen. Im Rahmen dieser Veranstaltung kommt hierbei der *aiT* der Firma *absint* zum Einsatz. Laden Sie Ihr Programm in aiT und analysieren Sie die WCET der Funktion `heapsort()`. *Wieso scheitert die Analyse? Wie können Sie aiT dazu bringen für diese Funktion eine Abschätzung zu ermitteln?*
Stellen Sie aiT so ein, dass eine Analyse gelingt und führen Sie die Analyse durch. Wiederholen Sie die Analyse mit ausgeschaltetem Instruktionscache. *Was beobachten Sie?*
Für welche Arten von Echtzeitsystemen ist die werkzeuggestützte WCET-Bestimmung gut geeignet? Für welche weniger gut? Inwiefern ist das Ausführungsmodell von eCos für die werkzeuggestützte WCET-Bestimmung problematisch? Haben Sie eine Idee, wie ein günstigerer Ansatz aussehen könnte?

4. *Analyse von quicksort:* Versuchen Sie nun die Funktion `quicksort()` ebenfalls zu analysieren. *Wieso scheitert die Analyse diesmal? Wieso wird im Echtzeitbereich für gewöhnlich Heapsort, nicht aber Quicksort verwendet?*

5. *WCET-Analyse-freundliche Entwurfsmuster:* Betrachten Sie nun die Funktion `sample_job()`. *Inwiefern ist das in dieser Funktion verwendete Entwurfsmuster für eine WCET-Analyse schlecht geeignet?*

Die Erweiterten Übungsaufgaben sind nur für Teilnehmer verpflichtend, die das 7,5-ECTS-Modul belegen. **Wir werden Sie natürlich auch dann bei der Bearbeitung unterstützen, wenn Sie diese Teilaufgaben freiwillig bearbeiten.**

Erweiterte Aufgabe

1. *Behebung fehlgeschlagener Analysen:* Annotieren Sie nun die Funktion `quicksort()` so, dass aiT eine möglichst genaue Abschätzung der WCET liefert.

Hinweise

- Bearbeitung: Gruppe mit je drei Teilnehmern.
- Abgabezeit: 10.11.2014
- Fragen bitte an i4ezs@lists.cs.fau.de