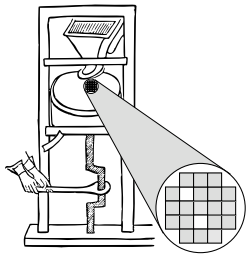
Technische Universität Darmstadt
FG Eingebettete Systeme und
ihre Anwendungen (ESA)

Prof. Dr. Andreas Koch



"Eingebettete Prozessorarchitekturen"
Aufgabe 1: VLIW
Abgabe bis zum 01.12.2006, 23:59 MET

Einleitung

In diesem ersten Übungsblatt geht es um praktische Experimente mit dem VLIW-Ansatz, speziell dem vereinfachten VEX-Prozessor. Die Grundlagen der zur Lösung benötigten Werkzeugaufrufe wurden bereits in der Vorlesung demonstriert. Vollständige Informationen finden sich in der, auch auf der Web-Seite verfügbaren, Dokumentation des VEX-Systems. Auf dieser Seite findet sich auch eine Materialsammlung zu diesem Übungsblatt. Im HOME-Bereich Ihrer Arbeitsgruppe auf unseren Rechnern sind diese Dateien bereits im Unterverzeichnis **`phase1-material/`** vorinstalliert.

Ihre Abgaben tätigen Sie fristgerecht in Form einer E-Mail an die Adresse

`epa06@esa.informatik.tu-darmstadt.de`

mit dem Betreff "Gruppe N Aufgabe 1", wobei N Ihre Gruppennummer ist. Diese E-Mail hat als Anhänge die in den Teilaufgaben geforderten Komponenten.

1 Experiment: VEX-Assembler (4 Punkte)

Schreiben Sie ein kurzes Programm in VEX-Assembler, das aus einer vorzeichenlosen 16b Zahl in Register $r0.1 in Register $r0.2 die 8b Quadratwurzel berechnet. Verwenden Sie dabei einen Algorithmus (Recherche!), der *ohne* Gleitkommazahlen und Division auskommt.

Bemühen Sie sich dabei um maximale Ausnutzung von Parallelität (möglichst viele Operationen pro Takt), beachten Sie aber:

- Zwischen der Berechnung eines Bedingungsregisters und der dieses auswertenden Sprunganweisung *muss* mindestens ein extra Takt liegen. Diesen können Sie idealerweise mit anderen nützlichen Instruktionen füllen, im Notfall müssen Sie eine **`xnop`** Anweisung einstreuen.
- Das gleiche gilt für die VEX Multiplikationsanweisungen. Auch hier muß mindestens ein extra Takt nach der Instruktion vergehen, bevor das Ergebnis vorliegt und gelesen werden kann.
- Wegen der bereits erläuterten Gutmütigkeit des Simulators werden diese Anforderungen leider nicht automatisch überprüft, auch im Fehlerfall wird das Programm korrekt simuliert. Sie müssen also diese Einschränkungen selber im Auge behalten!

Beim Verfassen des Quelltextes, z.B. als Datei **`mysqrt.s`**, können Sie sich an dem Beispiel **`intsum.s`** orientieren, das Sie auch in der Materialsammlung finden. Übersetzen Sie ihren Quelltext mit

```
vexcc -o mysqrt mysqrt.s asmlib.o
```

Das Programm ausführen können Sie dann durch den Befehl

```
./mysqrt
```

Wie in der Vorlesung demonstriert, können Sie durch die Pseudoinstruktion

```
c0 asm,0 $r0.0 = $r0.0, $r0.0
```

einen Auszug der ersten sechs Daten- und der ersten drei Bedingungsregister bekommen.

Nach der Ausführung entnehmen Sie der `ta.log.000` Datei (bzw. der aktuellen, die Zahl am Ende wird ja heraufgezählt) die Anzahl der Total, Execution und Stall Cycles sowie die IPC-Angaben (durchschnittliche Anzahl parallel ausgeführter Operationen). Geben Sie diese an und beurteilen Sie in einigen Sätzen Erläuterung anhand der width-Angaben, ob es Ihrem Programm gelingt, die maximale VEX-Parallelität von vier Operationen pro Instruktion gut auszunutzen.

Als Abgabe wird für diesen Teil der Quelltext des Programmes erwartet, der als Kommentar die im letzten Absatz geforderten Angaben und kurzen Diskussionen enthält.

2 Experiment: Compilierung für VEX (4 Punkte)

In der Materialsammlung finden Sie als `blending.c` die Quellen eines einfachen C-Programmes, das zwei Farbbilder ineinander überblenden kann. Informationen zur Benutzung finden sich in den Quellen selbst. Für die folgenden Experimente ist aus dem ganzen Quelltext nur die erste Funktion (namens `blending`) relevant, welche das eigentliche Überblenden vornimmt.

Um das Programm einfach zu halten, ist das Bildformat als XV PPM, 800x600, 8b pro Farbkomponente fest vorgegeben. Davon abweichende Eingabedaten können möglicherweise nicht korrekt gelesen werden. Zwei geeignete Beispielbilder (frei verfügbar auf `www.desktopia.com`) sind in der Materialsammlung als Dateien `image1.ppm` (Original ist Desktopia 40342.jpg) und `image2.ppm` (Desktopia 40685.jpg) enthalten.

Übersetzen Sie das Programm mit

```
vexcc -ms -H0 -o blending blending.c
```

und probieren Sie seine grundsätzliche Funktionsfähigkeit mit verschiedenen Alpha-Werten aus. Sie können die entstandenen Bilder beispielsweise mit dem Kommando `kuickshow` anzeigen lassen. Falls Sie einen tieferen Einblick in die Arbeit des Compilers nehmen wollen, finden Sie in der Datei `blending.s` den erzeugten VEX-Assembler. Darin ist der Anfang des wichtigsten Traces durch die Direktive `.trace 1` gekennzeichnet.

Löschen Sie eventuell vorhandene unnötige ta.log Dateien, und **sammeln** Sie nun einen ersten Satz Daten wie am Ende der letzten Aufgabe. Anschliessend übersetzen Sie das Programm nocheinmal neu *ohne* die `-H0` Option, Sie erlauben dem Compiler also das teilweise Entrollen von Schleifen. Dieser Vorgang wurder ja bereits in der Vorlesung anhand des cintsum-Programmes demonstriert, welches dadurch deutlich schneller wurde (mehr parallele Ausführung). **Beschreiben** Sie den Effekt, der durch das Entrollen auf das blending-Programm zu erreichen ist.

Studieren Sie nun die Sektion *Memory Disambiguation* des Abschnitts 3.2 der VEX-Dokumentation. Können Sie durch geeignete Anwendung dieser Technik die Programmlaufzeit verkürzen (anhand der Daten aus einer weiteren ta.log Datei)? **Beschreiben** Sie Ihre Vorgehensweise und Erkenntnisse!

Geben Sie für diesen Teil die oben geforderten Messdaten und Diskussionen (max. 1-2 Seiten Prosa, als ein PDF-Dokument) sowie den ggf. von Ihnen veränderten Quelltext des blending-Programmes ab.

3 Hinweise zum Thema Plagiarismus

Im Rahmen dieser Veranstaltung wird eine vorher festgelegte Arbeitsgruppe bewertet. Fremde Code-Bibliotheken außer den vom Dozenten zur Verfügung gestellten dürfen Sie *nicht* verwenden! Zusammenarbeit über Gruppengrenzen hinweg ist in Form der Diskussion von Lösungs*ideen* erlaubt. Es dürfen aber *keine* Artefakte wie Programm-Code, Dokumentationsteile (Text, Zeichnungen, Meßergebnisse) oder ähnliches ausgetauscht werden.

Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Mit der Abgabe einer Lösung (Hausaufgabe, Programmierprojekt, etc.) bestätigen Sie, dass Ihre Gruppe die alleinigen Autoren des gesamten Materials sind. Weiterführende Informationen zu diesem Thema finden Sie unter `http://www.informatik.tu-darmstadt.de/Plagiarism`.