Beginn: 13:00 Uhr
Ort: Arbeitsgruppenraum C2.227
Anwesende: alle Projektgruppenteilnehmer außer Franz Rammig

Als Tagesordnungspunkte lagen vor:

- Änderungen des Konzepts durch vorhandensein eines „SPARC 601"-Prozessors
- Anforderungen für das P-30 Dokument

Zuerst berichtete Christof Nagel, daß Wolfram Hardt inzwischen einen „SPARC 601" Prozessor zur Verfügung hat. Dieser Prozessor besitzt sowohl die FPU-Schnittstelle, als auch eine weitere Coprozessor-Schnittstelle.
Es wurde festgelegt, daß dieser Prozessor, und somit auch das Hardware-Interface von Wolfram Hardt, auch für unser Projekt genutzt werden soll. Die Gruppe „Hardware-Interface" gab hierbei zu bedenken, daß sie schon sehr viel Arbeit an der eigenen Lösung einer Schnittstelle für eine IU mit nur einer Coprozessorschnittstelle geleistet hat, und sich nun neu in das von Wolfram Hardt entwickelte Interface-Modell einarbeiten muß, wobei auch noch nicht feststeht, wie weit sich dieses Interface überhaupt nutzen läßt. Daraufhin wurde festgelegt, daß die eigene „Huckepack"-Lösung zumindest als Alternative festgehalten werden soll, und somit auch im „P-30" Dokument beschrieben werden soll.

Als nächstes stellte die Gruppe „Betriebssystem" noch einmal ihre Überlegungen zur Coprozessor-Handhabung vor. Diese Überlegungen basieren noch auf dem Modell der „Huckepack"-Lösung für den Coprozessor, können aber größtenteils auch auf die Lösung mit dem Interface von Wolfram Hardt übertragen werden.
Es wurde davon gefordert, daß der Coprozessor keine eigenen Exceptions auslösen darf, sondern eventuelle Fehlercodes im CSR (Coprocessor Status Register) ablegt. Hiermit sind keine gesonderten Betriebssystem-Routinen erforderlich. Bei Benutzung des Interface von Wolfram Hardt kann dieses Modell übernommen werden, oder alternativ auch eine gesonderte Behandlung der Coprozessor-Exceptions vorgenommen werden, da bei dem Prozessor mit zwei verschieden Coprozessor-Schnittstellen die Herkunft lokalisiert werden kann.
Falls ein Trap ausgelöst wird, da das „FPU-Disabled-Bit" gesetzt ist, muß hierfür der Handler erweiter werden. Zu den Punkten
– FPU initialisieren
und
– Aktuellen Prozeß als FPU-benutzend markieren
muß als weiterer Punkt
– Coprozessor initialisieren
hinzugefügt werden. Dieser Fall kann bei Benutzung des „SPARC 601" nicht auftreten, da beide Coprozessoren (FPU und SFU) getrennt angeschlossen sind.
Bei der eigenen Lösung muß deshalb auch die Routine für den Kontextswitch erweitert werden. Es müßen beim Kontextswitch alle Coprozessor Register und alle FPU Register gesichert werden. (CSR/FSR und normale Register) Bei der Lösung von Wolfram Hardt muß dies prinzipiell auch geschehen, es wird aber eventuell schon durch das Betriebssystem unterstützt.

Der äußere Aufbau der SFU muß gleich dem der FPU sein. Somit kann auch die Synchronisation wie bei der FPU geregelt werden. Die Befehle werden normalerweise parallel abgearbeitet. Die Synchronisation setzt dann ein, wenn die Integer-Unit einen Befehl sendet, der kritisch werden kann. Dies sind die Befehle:

- store — Hier muß zuerst überprüft werden, ob das Ergebnis bereits berechnet ist.
- load — Es kann nur in ein freies, unbenutztes Register geschrieben werden.
- FSR/CSR sichern (für Kontextswitch) — Das FSR/CSR kann nur gesichert werden, wenn sich der Coprozessor in einem sicheren Zustand befindet.

In diesen Fällen wird das FHOLD bzw. CHOLD Signal gesendet, und die IU stoppt, bis der Coprozessor zu Ende gerechnet hat. Durch diesen Ablauf können die Assemblerbefehle zur Ansteuerung der SFU sequentiell hintereinander geschrieben werden, ohne daß eine Abschätzung der Dauer der Berechnung zwingend notwendig ist.

Dann wurden die Coprozessor-Befehle besprochen. Es wurde bemerkt, daß der notwendige Befehl um das CSR auszulesen zu den festen Befehlen gehört. Für eigene Befehle stehen 2 Blocks mit je 512 freien Befehlen, also insgesamt 1024 Befehle, zur Verfügung. Es wurde beschlossen, für die eigenen Befehle das gleiche Format zu übernehmen, das auch die anderen Befehle haben.
Für den zu benutzenden Datentyp wurde beschlossen, das „Single"-Format zu benutzen, welches 32 Bit lang ist. Dieser Typ bietet ausreichende Genauigkeit, und das Rechenwerk würde bei längeren Typen zu groß werden. Der Datentyp Single entspricht in der Programmiersprache C dem Typ „float".

Es wurde festgestellt, daß zwei Befehle auf jeden Fall notwendig sind:

- load — Um die Stützstellenwerte an die SFU zu übergeben
- store — Um die berechneten Polynomwerte im Hauptspeicher abzulegen.

Es wurde festgelegt, daß für Lade- und Speicheroperationen nach Möglichkeit das „Extended" Datenformat benutzt werden soll. Da „Extended" die vierfache Länge gegenüber „Single" besitzt, ist es hiermit möglich, beim Laden und Speichern vier Werte gleichzeitig mit einem Befehl zu übergeben. Vor allem beim Speichern erscheint dies sehr sinnvoll, da jeweils ein Polynom durch vier Werte beschrieben wird. Es muß allerdings noch abgeklärt werden, ob das Protokoll eine solche Behandlung der Werte ermöglicht, oder ob doch das normale „Single" Format benutzt werden muß.
Es wurde festgestellt, daß weder ein expliziter Start-Befehl, noch ein Ende-Signal notwendig sind. Die Gruppe „Rechenkern" sagte, daß der Rechenkern seine Berechnung in Abhängigkeit von den „valid-Flags" der einzelnen Register beginnt. Hierzu muß in Register Nr. 0 die Anzahl der verwendeten Stützstellen übergeben werden. Wenn dann bei n Stützstellen die ersten $(2n + 1)$ (x- und y-Werte der Stützstellen + Anzahl der Stützstellen) Register gültig sind, beginnt die Berechnung der Splinepolynome. Für das Auslesen der Werte braucht auch kein spezielles Signal gesetzt werden. Falls die IU versucht die Werte auszulesen, bevor die Berechnung beendet ist, wird die IU durch das CHOLD-Signal automatisch gestoppt, und die SFU kann die Rechnung zu Ende führen. Um unnötiges anhalten der IU zu vermeiden, soll aber trotzdem versucht werden, die Dauer der Berechnung, in Abhängigkeit von der Stützstellenzahl, abzuschätzen. Mit dieser Information kann dann der Assemblercode optimiert werden.
Diese Art der Implementierung ohne Start-Befehl und ohne Ende-Signal lehnt sich stark an das Interface-Konzept von Wolfram Hardt an, was ein weiterer Grund ist, weshalb diese Lösung bevorzugt wurde. Hierbei ist allerdings noch zu überprüfen, wie verläßlich dieses Interface ist, da bislang nur die Beschreibung des Interfaces vorliegt, und noch nicht abschließend hierüber geurteilt werden konnte.

Desweiteren wurde besprochen, ob die SFU eigene Exceptions verursachen kann, und wie diese behandelt werden können. Die Standard Exception-Behandlung wäre, den jeweiligen Prozeß zu suspendieren. Es war allerdings noch nicht zu klären, wie weit diese Behandlung für Exceptions, die nicht von der FPU, sondern von einem zusätzlichen Coprozessor, ausgelöst wurden, vom Kernel des Betriebssystems bereits vorgesehen ist. Als möglicher Grund für eine Exception kommen allerdings nur zwei Fehler in Betracht:

- Queue voll
- Division durch Null

Es wurde beschlossen, daß diese Fehler vom Benutzer-Interface abzufangen sind, wobei allerdings noch unklar ist, ob eventuell eine Division durch Null auftreten kann, die vorher nicht entdeckt werden kann. Da die möglichen Fehlerquellen vom Benutzer-Interface abgefangen werden, ist eine besondere Exception-Behandlung für Fehler der SFU nicht notwendig.

Die Gruppe „Rechenkern" gab noch zu bedenken, daß es, trotz der Pipeline-Struktur der gesamten SPARC-Architektur, wahrscheinlich nicht möglich sein wird, mehrere Spline-Berechnungen parallel zu bearbeiten. Es wurde deshalb beschlossen, das Benutzer-Interface in dieser Richtung nicht zu erweitern. Das bisherige Konzept beruht auf der Annahme, das jeweils eine Punktemenge behandelt wird, durch die ein Splinepolynom gelegt werden soll, und erst nach Abarbeitung dieser Punkte ein neuer Spline behandelt werden kann.