MMIX als einheitlicher Modellprozessor für weite Teile des Informatik-Studiums

Axel Böttcher Martin Ruckert
Fachhochschule München
Fakultät für Informatik und Mathematik
{ab, ruckert}@cs.fhm.edu

Abstract: Wir berichten über unsere Erfahrungen mit dem Einsatz des Modellprozessors MMIX für das Studium der Informatik. Wir bewerten zunächst verschiedene Konzepte von Modellprozessoren hinsichtlich ihrer Eignung, hauptsächlich unter didaktischen Gesichtspunkten. Dann zeigen wir, wie der MMIX-Prozessor in weiten Teilen des Studiums eingesetzt werden kann: Grundlagen, Rechnerarchitektur, hardwarenahe Programmierung und Übersetzerbau.

1 Einleitung

Viele Hochschulen setzen für die Ausbildung in Informatik Modellprozessoren ein, um fundamentale Konzepte an der Schnittstelle zwischen Hard- und Software zu lehren. Die Bandbreite reicht dabei von sehr einfachen Stack-Maschinen und einfachen selbst geschriebenen Simulatoren bis hin zu komplexen Prozessoren wie MIPS und deren professionellen Simulatoren.

Wir berichten hier über die Erfahrungen mit dem MMIX-Prozessor von Donald E. Knuth ([Knu99], [Knu05]), den wir seit einigen Jahren für die Ausbildung einsetzen.

Als Grundausstattung ist ein Assembler verfügbar sowie zwei Simulatoren (ein Instruction-level Simulator und ein Mikroarchitektur-Simulator).

2 Modellprozessoren und Simulatoren

In diesem Abschnitt stellen wir einen Kriterienkatalog für Modellprozessoren und Simulatoren auf. Anhand dieses Katalogs bewerten wir anschließend verschiedene existierende Modellprozessoren.

2.1 Anforderungen aus didaktischer Sicht

Modellprozessoren für die Lehre in der Informatik müssen aus didaktischer Sicht mehrere Anforderungen erfüllen:

1. Damit die grundlegenden Konzepte (oft als ewige Wahrheiten bezeichnet) vermittelt werden können, sollten Modellprozessoren ausreichend abstrakt und möglichst frei von komplizierten Details und Sachzwängen realer Prozessoren sein. Um andererseits aber nicht den Eindruck praktischer Irrelevanz zu erwecken, sollte ein Modellprozessor konzeptionell nicht zu weit von modernen Prozessoren entfernt sein.

2. Der Einstieg darf nicht zu kompliziert sein, um den Studierenden nicht von Anfang an die Motivation zu nehmen. Es müssen sich also einfache Programme mit nur einer Hand voll Befehlen schreiben lassen, ohne allzu viel Randbedingungen beachten zu müssen. Für die Langzeitmotivation zu bieten, sowie um die Studierenden mit großem Vorwissen nicht zu langweilen, sollte es möglich sein, auch komplexere Programme für die Maschine zu erstellen.

3. Es muss wenigstens rudimentäre Unterstützung für die Fehlersuche vorhanden sein. Günstig ist es, wenn Simulatoren auch mit professionellen Werkzeugen, wie dem Gnu-Debugger zusammen arbeiten.

4. Werkzeuge zur Visualisierung der Prozessorzustände oder anderer innerer Abläufe sind sehr hilfreich. Solche Werkzeuge können sowohl im Unterricht als auch beim Selbststudium durch Anschaulichkeit das Verständnis erleichtern.

5. Wichtig ist ferner die breite Einsatzmöglichkeit eines Modellprozessors für Grundlagenvorlesungen sowie für weiterführende Themen. Im Idealfall sind dafür mehrere Simulatoren verfügbar.

6. Für den Einsatz im Unterricht ist ferner von Bedeutung, dass die Simulationssoftware gut dokumentiert ist, gepflegt wird sowie quelloffen, wartbar und gut verfügbar ist.

2.2 Bewertung

Wir wollen eine grobe Bewertung einiger prinzipiell in Frage kommender (Modell-) Prozessoren versuchen. Yurcik et al. [YWH01] geben einen guten Überblick über die verfügbaren Simulatoren. Im einzelnen ist anzumerken:

1. Unter dem Gesichtspunkt der Relevanz sind insbesondere ältere 8- und 16-Bit-Prozessoren sowie Stack-Maschinen (Kellerautomaten) auszuschließen. `MMIX` eignet sich hervorragend zur Darstellung der grundlegenden Konzepte, weil er für diesen Zweck entworfen wurde [Knu05]. Als 64-Bit-RISC-Prozessor erfüllt er auch das Kriterium der Modernität.

2. Viele reale Prozessoren oder Simulatoren für reale Prozessoren weisen komplizierte Details auf oder interagieren auf komplizierte Weise mit einem Betriebssystem, was Anfängern schwer zu vermitteln ist.

3. Debugging-Möglichkeiten sind bei den Simulatoren vorhanden. Professionelle Debugger lassen sich meist nur mit den kommerziellen Simulatoren verwenden. Der im `MMIX`-Instruction-level Simulator eingebaute Mechanismus, Progamme schrittweise auszuführen (interaktiver Modus), ist gerade ausreichend, um Fehler in selbst geschriebenen Assembler-Programmen zu finden.

4. Die Verfügbarkeit von Visualisierungen ist insgesamt als gut bis sehr gut zu bezeichnen. Viele Hochschulen haben auf diesem Gebiet viel getan und Werkzeuge sowie Visualisierungsumgebungen implementiert. Für `MMIX` gibt es mehrere Visualisierungswerkzeuge (siehe unter `www.mmix.de`), ferner Programmierwerkzeuge unter anderem die Gnu-Tools.

5. In der Arbeit von Yurcik et al. [YWH01] werden die Hörer-Kategorien „novice audience“, „intermediate audience“ sowie „advanced audience“ eingeführt. Die Kategorie der Simulatoren für intermediate audience wird ferner unterteilt in Instruction-level und Mikroarchitektur-Simulatoren. Allein diese Kategorisierung legt nahe, dass meistens für

unterschiedliche Vorlesungen verschiedene Modellprozessoren zum Einsatz kommen. Gerade hier scheint uns der Hauptvorteil von MMIX zu liegen.

6. Viele gerade sehr gute Simulatoren sind kommerzielle Produkte mit teilweise erheblichen Lizenzgebühren. Andere gute Simulatoren (z.B. MIPSSim, SimpleScalar), sind aufwändig in Installation und Konfiguration.

3 Einsatz von MMIX im Curriculum

Ein Prozessormodell mit entsprechenden Werkzeugen an der Hand zu haben, hat den Vorteil, dass die Studierenden sich nicht für jede Veranstaltung mit einem neuen Programmiermodell auseinandersetzen müssen.

In den folgenden Abschnitten beschreiben wir unsere Erfahrungen mit der Verwendung von MMIX in verschiedenen Modulen des Bachelor-Studiengangs Informatik der FH München.

3.1 Grundlagenvorlesung

In den ersten beiden Semestern behandelt die Vorlesung „IT-Systeme“ den Aufbau und die Funktionsweise von IT-Systemen von der Registerebene ausgehend bis hin zum Betriebssystem. Die Studierenden sollen dort mit der Schnittstelle zwischen Hard- und Software vertraut gemacht werden. Die Studierenden bekommen dabei auch Kenntnisse über den technischen Aufbau von Rechenanlagen vermittelt [ABR02]. Diese Themen werden im schulischen Unterricht in der Regel bewusst nicht behandelt [Hub03].

3.2 Rechnerarchitektur

Großes Potenzial entfaltet MMIX für Veranstaltungen über Rechnerarchitektur [BÖ6]. Mit einem zweiten Simulator namens mmmix (für Meta-MMIX) lässt sich MMIX als flexibel konfigurierbare superskalare Maschine simulieren.

Durch eine Konfigurationsdatei lassen sich etwa 80 Parameter einstellen. Der Meta-Simulator mmmix gibt während der Simulation viel Text aus, um den jeweils aktuellen Zustand des Prozessors zu beschreiben: ungefähr 1KByte pro Taktzyklus. Eine Simulation anhand dieser Ausgaben zu verfolgen, ist sehr mühsam. Daher haben wir eine Visualisierungsumgebung als Plugin für Eclipse geschrieben (dies steht mmix-plugin.sourceforge.net allgemein zur Verfügung). Damit werden die Textausgaben geparst und der Prozessorzustand graphisch dargestellt.

3.3 Maschinennahe Programmierung

MMIX ist ein moderner „general purpose“ Prozessor, für den nicht nur der schlichte, von Donald Knuth bereitgestellte, mmixal-Assembler zur Verfügung steht. Sowohl der GNU-Assembler als auch das GNU-Binutils Package und die GNU-Compiler Collection (GCC) [Nil01] lassen sich ohne weiteres für den Zielprozessor MMIX konfigurieren. Damit steht eine Werkzeugumgebung bereit, die es ermöglicht, auch fortgeschrittene Kurse in Assemblerprogrammierung zu unterrichten.

Im Rahmen des MMIX-Motherboard Projektes [R+04] steht inzwischen auch ein MMIX-Simulator zur Verfügung, der sich problemlos um eine Vielzahl von Devices erweitern lässt. Es handelt sich beim MMIX-Motherboard um eine offene TCP/IP basierte Bus-Architektur, die besonders die einfache Programmierung von Erweiterungen unterstützt. So lassen sich Simulatoren auch für spezielle Hardwarekonfigurationen leicht zusammenstellen.

3.4 Übersetzerbau

Durch seinen regulären Aufbau eignet sich MMIX-Code hervorragend als Zielsprache eines Übersetzers. Durch die durchgehende 3-Adress-Form der Befehle kann MMIX-Code aber auch leicht als Ersatz für konventionellen 3-Adress-Code, also als Zwischensprache verwendet werden. Dies unterstützt das intuitive Verständnis der Zwischensprache.

Bei der Arbeit am Backend kann die GNU-Compiler Collection, für die eine Portierung auf den MMIX existiert, Verwendung finden. Es bietet sich hier aber noch ein weites Betätigungsfeld für Projektarbeiten, da die vorhandene Implementierung bei weitem nicht die Möglichkeiten des MMIX-Prozessors ausschöpft.

4 Wertung

Insgesamt hat die Erfahrung über die letzten Jahre gezeigt, dass der MMIX-Prozessor durch das ganze Curriculum hindurch ein tragfähiges Prozessormodell bietet und sich aus inhaltlicher ebenso wie aus didaktischer Sicht für den Einsatz in der Lehre empfiehlt. In praktisch allen Fächern sind Synergieeffekte spürbar, da im Unterricht auf ein allen Studenten gemeinsames Grundwissen ebenso wie auf eine gemeinsame Notation und Terminologie zurückgegriffen werden kann.

Leider gibt es derzeit noch zu wenig gute Werkzeuge für MMIX und insbesondere noch kein Betriebssystem, das sich auf MMIX starten lässt. Anschluss des Gnu-Debuggers als professionellem Entwicklungswerkzeug ist gerade in Arbeit [R$^+$04].

Literatur

[ABR02] Heidi Anlauff, Axel Böttcher und Martin Ruckert. *Das MMIX-Buch – Eine praxisnahe Einführung in die Informatik*. Springer Verlag, Heidelberg, 1. Auflage, 2002.

[BÖ6] Axel Böttcher. *Rechneraufbau und Rechnerarchitektur*. Springer Verlag, Heidelberg, 1. Auflage, 2006.

[Hub03] Peter Hubwieser. *Didaktik der Informatik*. Springer Verlag, Heidelberg, 2. Auflage, 2003.

[Knu99] Donald E. Knuth. *MMIXware: A RISC Computer for the Third Millennium*. Springer-Verlag, Berlin, Heidelberg, 1. Auflage, 1999.

[Knu05] Donald E. Knuth. *The Art of Computer Programming – Vol. 1: MMIX a RISC Computer for the new Millennium. Fascicle 1*. Addison–Wesley Pub. Co., Reading, MA, 1. Auflage, 2005.

[Nil01] Hans-Peter Nilsson. Installation instructions for GCC & Co. MMIX tools. bitrange.com/mmix/install.html, 2001.

[R$^+$04] Martin Ruckert et al. The MMIX Motherboard Project. mbox.informatik.fh-muenchen.de/mmixmb/, 2004.

[YWH01] William Yurcik, Gregory S. Wolffe und Mark A. Holiday. A Survey of Simulators Used in Computer Organization/Architecture Courses. In *Proceedings of the 2001 Summer Computer Simulation Conference (SCSC 2001), Orlando FL.*, 2001.