Christoph Regli

µOBERON

A Development System for Mcs51 Microcontrollers

Diploma Thesis 1996/97
Swiss Federal Institute of Technology
Supervised by Erwin Oertli, Hans Eberle

µOBERON
A Development System for Mcs51 Microcontrollers
Christoph Regli

Institute for Computer Systems
Department of Computer Science
Swiss Federal Institute of Technology, Zurich

Erwin Oertli, Prof. Hans Eberle

Acknowledgements

I would like to express my gratitude to Professor Hans Eberle who gave me the opportunity to do this interesting work. It was a great challenge which continually led to new hurdles that had to be taken. Erwin Oertli who attended me was always a great help and didn't hesitate to contribute new ideas and suggestions. It was exciting to discuss with him about what should be implemented and where the focal points of my interest should be, and his enthusiasm for simple but smart and clever solutions is very infectious. His knowledge about compiler construction seems to be inexhaustible. Jonas Kurth and Pascal Peng deserve special thanks for proofreading this text. They pointed out numerous grammatical and stylistic mistakes, and their corrections and suggestions led to an improved readability.

Finally I am greatly indebted to my parents who enabled me to do my studies, and to whom this work is dedicated.

Kurzfassung

Als aussergewöhnlich vielseitig einsetzbare Bauteile gewinnen Mikrocontroller immer mehr an Bedeutung. Verglichen mit mechanischen oder elektrischen Systemen erhöhen sie Funktionalität und Zuverlässigkeit und reduzieren gleichzeitig Grösse und Kosten. Zudem trägt die Wiederverwendbarkeit von Programm- und Bauteilen zusätzlich zur Reduktion der Entwicklungszeit und Kosten bei.

Mikrocontroller werden heutzutage aus Gründen der Geschwindigkeit häufig in Assembler und nicht in einer Hochsprache programmiert. In Assembler erstellte Programme weisen meist eine höhere Codedichte auf, was einen geringeren Speicherbedarf und eine kürzere Ausführungszeit zur Folge hat. Auf der anderen Seite bietet eine Hochsprache mehr Komfort und Effizienz beim Entwickeln von Programmen. Modulares Programmieren und Wiederverwendbarkeit von Programmteilen sind nur zwei Stichworte in diesem Zusammenhang.

µOberon ist ein Versuch, die Geschwindigkeit von Assembler mit der Modularität von Hochsprachen zu vereinen. Mit dem Inline-Assembler ist es möglich, neben den Anweisungen in der Hochsprache direkt auf Maschinenebene zu programmieren, wo alle Register des Mikrocontrollers direkt zugänglich sind.

Die Hauptschwierigkeit während der Entwicklung von µOberon lag darin, dass der Instruktionssatz der Mcs51-Mikrocontroller keine Befehle mit indirekter Adressierung mit Offset anbietet. Andere Crosscompiler berechnen daher die Zieladresse bei jedem Variablenzugriff, so dass wie gebräuchlich die lokalen Variablen auf dem Stack alloziert werden können. µOberon hingegen führt mit der statischen Speicherzuteilung einen neuen Ansatz ein. Die absoluten Adressen der lokalen Variablen werden hier statisch während dem Linken zugeteilt, was zu besserem Code führt, der sich selbst mit Assemblerprogrammen vergleichen lässt.

Abstract

Providing general purpose solutions, microcontrollers are becoming increasingly important in the world of electronic systems. Compared to mechanical or simple analog electrical control systems they dramatically improve functionality and reliability, while reducing size and cost. Furthermore the capability of reusing software and hardware components reduces overall design-in time and cost.

Nowadays microcontrollers usually are programmed in assembler and not in a high-level language for efficiency reasons. As a rule, programs written in assembler language have a higher code density, what results in a lower memory consumption and a shorter execution time. On the other side a high-level language provides more comfort and higher efficiency during program development. Modular programming and code reusing are only two points in that context.

µOberon is a an approach to unite the speed of assembler programs and the modularity of high-level programs. It allows high-level as well as low-level programming, i.e. all microcontroller resources are easily accessible. This implied the need of an inline assembler, and µOberon offers a comfortable embedding of assembler sections into the language.

The main problem while developing µOberon was the fact that Mcs51 microcontrollers don't provide indirect-offset addressing. Therefore other cross compilers calculate the effective address on each variable access, so local variables may be located on the stack and addressed relative to a frame pointer as usual. µOberon on the other hand introduces a new approach, the so-called Static Memory Assignment, where the absolute addresses of local variables are assigned statically while linking. This leads to better code that even competes with assembler programs.

Table of Contents

Acknowledgements i

Kurzfassung ii

Abstract iii

Table of Contents v

1 Introduction 1
1.1 Motivation 1
1.2 Goal 1
1.3 Used Terms 1
1.4 Outline 4

2 Mcs51 Microcontrollers 5
2.1 Overview 5
2.2 Programming Model 6
2.3 Problems for Compiler Construction 7

3 Project µOberon 8
3.1 Aims 8
3.2 T Diagrams 9
3.3 Module Overview 10
3.4 Register Management 11
3.5 Sets 12
3.6 Numbers 13
3.7 Fixup Chains 14
3.8 CASE Statement 15
3.9 Dynamic Memory 17
3.10 Procedure Variables 17
3.11 Inline Assembler 18
3.12 Static Memory Assignment 18
3.13 The Linker 23
3.14 The Loader 25

4 Using µOberon 26
4.1 Development Process 26
4.2 The µOberon Control Panel 26
4.3 The µOberon Options Panel 27

5 Conclusion 30
5.1 Summary 30
5.2 Benchmark 30
5.3 Outlook 31

Epilogue 34
Computer Scientists and Transparent Programming 34

Appendix A: The Implemented Language µOberon 35
A.1 Differences Between Oberon and µOberon 35
A.2 Interrupt Service Procedures 35
A.3 Trap Procedure 36

A.4 The Inline Assembler 36
A.5 EBNF of μOberon 37
A.6 Predefined Procedures 40
A.7 The Module SYSTEM 41
A.8 Trap Numbers 42
A.9 ASCII Character Set 42

Appendix B: Mcs51 Hardware Overview **43**
B.1 Mcs51 Microcontroller Family 43
B.2 Architecture 44
B.3 Code Memory 46
B.4 Data Memory 47
B.5 Special Function Registers 48

Appendix C: Mcs51 Instruction Set Summary **58**
C.1 Addressing Modes 58
C.2 Instruction Set 59

Appendix D: File Formats **65**
D.1 μOberon Object File Format 65
D.2 μOberon Symbol File Format 66
D.3 Intel-Hex File Format 66
D.4 μOberon Options File Format 67

Appendix E: Data Structures and Module Interfaces **68**
E.1 Data Structures 68
E.2 Module Interfaces 68

Appendix F: Bibliography **74**

Appendix G: Index **75**