

Software Pipelining

Serial Assembly Optimizer for a VLIW Processor

Submitted
by
Muhammad Salman



Supervised
by
Mr. Jehanzeb Ahmed

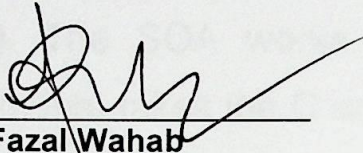
A report submitted to the department of Computer Science,
Bahria Institute of Management and Computer Sciences, Islamabad.
In partial fulfillment of requirement for the degree of BCS (Hons).

Department of Computer Sciences,
Bahria Institute of Management and Computer Sciences, Islamabad.
University of Peshawar, Peshawar.

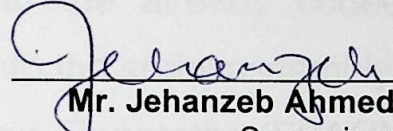
Abstract

Certificate

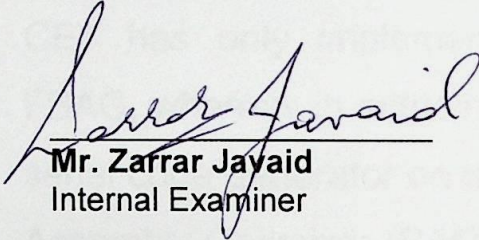
We accept the work contained in this report as a confirming to the required standard for the partial fulfillment of the degree of BCS (Hons).



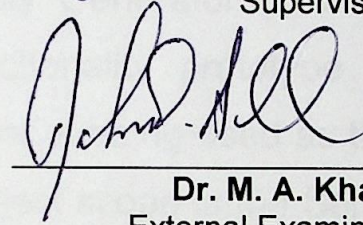
Mr. Fazal Wahab
Head of Department



Mr. Jehanzeb Ahmed
Supervisor



Mr. Zaffar Jayaid
Internal Examiner



Dr. M. A. Khan
External Examiner

Abstract

We present a *Serial Assemble Optimizer (SOA)* for the Media Engine (ME-2), being developed in Communications Enabling Technology (CET). The SOA works in collaboration with the already coded module that takes the C code and converts it into the serial assembly, referred hereon as Front-end Serial Assembly Generator (FSAG). CET has only implemented a minimal functionality prototype of FSAG, whereas in actuality the GNU C compiler is being used as the serial code generator on the backend. Our project scope is the Serial Assembly Optimizer (SAO), which takes the serial code generated by the FSAG and makes use of advanced optimization techniques to generate a parallel, and optimized code.

Acknowledgements

The final year project has been a great experience for us. During the course of its duration we have learned a lot. We have been able to apply the knowledge and skills that we have developed during our three and a half years of stay in Bahria. However, this humble effort would not have been fruitful if it were not for the guidance and support many people. We would like to thank those who, often despite of their own commitments, have taken time and effort to help us.

First of all we would like to thank Allah Almighty for his guidance, without which we would have been lost. Then we would like to thank our Project Supervisor, Mr. Jehanzeb Ahmed for his help, understanding and considerations at difficult times and of course all the people in CET who helped us in every possible way.

Table Of Contents

| | |
|--|-----------|
| Abstract | 3 |
| Acknowledgements | 4 |
| Table Of Contents | 5 |
| Section one | 9 |
| Introduction | 9 |
| I- Hand Optimization Techniques are not Scalable | 10 |
| II- Hand-Optimized Code is not Portable | 10 |
| III- Related Work | 11 |
| Chapter 1 | 12 |
| Introduction To The Dissertation | 12 |
| 1.1) Purpose of Dissertation | 12 |
| 1.2) Scope of Dissertation | 12 |
| 1.3) Layout of the Dissertation | 13 |
| Chapter 2 | 14 |
| Project Description | 14 |
| 2.1) Description | 14 |
| 2.2) Features | 15 |
| 2.2.1) Modular Approach | 15 |
| 2.2.2) Front-end Serial Assembly Generator | 15 |
| 2.2.3) Serial Assembly Optimizer | 15 |
| 2.3) Life Cycle Model | 16 |
| 2.3.1) PHASES | 16 |
| 2.3.2) FSAG | 17 |
| 2.3.3) SAO | 17 |
| Section two | 18 |
| Background Knowledge | 18 |
| Chapter 3 | 19 |
| ME-2 Architecture | 19 |
| 3.1) Introduction | 19 |
| 3.2) VLIW Architecture | 20 |
| 3.2.1) Terminology | 22 |
| 3.2.2) Principles Behind VLIWs | 23 |
| 3.2.2.1) Datapaths | 23 |
| 3.2.2.2) Pipelines | 23 |
| 3.2.2.3) Functional units | 24 |
| 3.3) ME-2 Architecture | 26 |
| 3.3.1) Central Processing Unit (CPU) | 26 |
| 3.3.2) Internal Memory | 27 |
| 3.3.3) TXP/RXP Pipeline | 27 |
| 3.4) Instruction Set Overview | 27 |
| 3.4.1) Instruction Types | 27 |
| 3.4.1.1) AGU Instructions | 28 |
| 3.4.1.2) DataPath Instructions | 28 |

| | |
|--|-----------|
| 3.4.2) Registers | 28 |
| 3.4.3) Addressing Modes | 29 |
| 3.4.4) Use of data pointer registers | 29 |
| 3.4.5) Execution Block Packet Composition | 30 |
| 3.4.6) VLIW Grouping Restrictions | 30 |
| 3.4.7) Looping restrictions | 31 |
| 3.4.8) Conditional Execution | 32 |
| 3.4.9) Latencies | 33 |
| Chapter 4 | 34 |
| Optimizations Techniques | 34 |
| 4.1) Parallelism in Programs | 34 |
| 4.1.1) Coarse-grain parallelism | 34 |
| 4.1.2) Fine-grain of Instruction Level Parallelism | 35 |
| 4.2) Types of Optimizations | 35 |
| 4.2.1) Classical Optimizations | 36 |
| 4.2.2) Superscalar Optimizations | 36 |
| 4.2.3) Multiprocessor Optimizations | 37 |
| 4.3) Dependence Analysis | 38 |
| 4.3.1) Resource Dependencies | 38 |
| 4.3.2) Control Dependencies | 38 |
| 4.3.3) Data Dependencies | 39 |
| 4.3.4) Dependence Graphs | 40 |
| 4.4) VLIW Compilers | 41 |
| 4.5) Optimization Techniques For a VLIW Compilers | 42 |
| 4.5.1) Trace Scheduling | 42 |
| 4.5.2) Software Pipelining | 42 |
| 4.5.3) Loop Unrolling | 60 |
| 4.5.4) Register Scheduling | 63 |
| Section three | 67 |
| Project Specifications | 72 |
| Chapter 5 | 73 |
| Analysis and Design Specification | 73 |
| 5.1) Environmental Model | 73 |
| 5.1.1) Statement of Purpose | 73 |
| 5.1.2) Context Diagram | 74 |
| 5.2) Data Flow Diagrams | 75 |
| 5.3) Process Specification | 77 |
| 5.3.1) Dependence Analysis (2.1) | 77 |
| 5.3.2) Loop Unrolling (2.2) | 77 |
| 5.3.3) Software Pipelining and Scheduling (2.3) | 77 |
| 5.3.4) Low-Level Optimization (1.2.3) | 78 |
| 5.3.5) Code Generation (1.2.4) | 78 |
| 5.4) Use Case Diagram | 79 |
| 5.4.1) Actor—Programmer | 80 |
| 5.4.2) Use Case Description | 80 |
| 5.4.2.1) Serial Assembly Optimization | 80 |
| 5.5) Class Relationship Collaborators | 81 |
| 5.6) Class Relationship Diagram | 85 |
| 5.7) Sequence Diagrams | 86 |
| 5.7.1) Serial Assembly Optimization | 86 |
| 5.8) Class—Attributes, Methods | 87 |

| | |
|---------------------------------------|------------|
| Section four | 96 |
| Results and Conclusion | 96 |
| I- Results | 96 |
| II- Analysis and Conclusions | 100 |
| III- Future Recommendations | 100 |
| Appendix A | 102 |
| Serial Assembly Format | 102 |
| A-1) Register Allocation | 103 |
| A-2) For Loops | 103 |
| A-3) If else and Predicated Execution | 104 |
| A-4) Auto Correlation | 104 |
| Index | 110 |
| References | 112 |

List of Figures/Table

| | |
|---|----|
| <i>Figure 3.1</i> Block diagram of an ideal VLIW and its instruction word | 21 |
| <i>Figure 3.2</i> Datapaths of a generic machine | 25 |
| <i>Figure 3.3</i> Pipelining example | 25 |
| <i>Figure 3.4</i> Looping Restrictions | 31 |
| <i>Table 3.1</i> Latencies | 33 |
| <i>Figure 4.1</i> Trace scheduling example. | 44 |
| <i>Figure 4.3.</i> Pseudo-assembly code. | 47 |
| <i>Table 4.1.</i> Single loop iteration | 47 |
| <i>Table 4.2.</i> Software pipelined schedule. | 48 |
| <i>Figure 4.5</i> Pseudo- assembly code | 61 |
| <i>Table 4.3</i> Single loop iteration | 62 |
| <i>Table 4.4.</i> Software pipelined schedule | 62 |
| <i>Figure 4.6.</i> Register renaming | 64 |
| <i>Figure 4.7.</i> A program and its dependence edge of scheduling graph | 64 |
| <i>Figure 4.8.</i> A scheduling Graph, the edge in the set E_t , and the interference graph | 65 |
| <i>Figure 4.9.</i> Example of modulo scheduling | 66 |
| <i>Figure 4.10.</i> Live in and Live out values graph | 67 |
| <i>Figure 4.11.</i> Rotating Register | 68 |
| <i>Figure 4.12.</i> The Generalized branch operation | 70 |
| <i>Figure 5.1</i> Context Diagram—Level 0 | 74 |
| <i>Figure 5.2</i> DFD (1) | 75 |
| <i>Figure 5.3</i> DFD (2) | 75 |
| <i>Figure 5.4</i> DFD (1.1) | 76 |
| <i>Figure 5.5</i> DFD (1.2) | 76 |
| <i>Figure 5.6</i> Use case Diagram | 79 |
| <i>Figure 5.7</i> Class Diagram | 85 |
| <i>Figure 5.9</i> Sequence Diagram—Serial Assembly Optimization. | 86 |