CITY UNIVERSITY OF HONG KONG

Register Allocation for Embedded Systems

Submitted to
Department of Computer Science
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

by

Huang Yazhi

September 2012

黄亞志
Abstract

Compilers play an important part in the system performance improvement. A compilation process is to translate source code from a high-level programming language to a lower level language. For embedded systems, the system resource such as the number of registers, cache size, and memory size are very limited. Therefore, an efficient compilation can particularly have great effect on the system performance of these systems. Register allocation, which is a key compilation step, aims at multiplexing a large number of target program variables onto a small number of physical registers. A register allocator tries to keep as many operands as possible in registers to maximize the execution speed of programs. For embedded systems often with limited number of registers, it is particularly important to conduct efficient register allocation for the system performance improvement.

Embedded systems are usually designed for specific application and these applications vary widely. Various application of embedded systems have difference requirement in compilation. Therefore, the compilation solutions for embedded systems should be determined according to the actual need of their application. For register allocation activity in the compilation process, an identical solution can have positive or negative effect on the system performance of different embedded systems with different design purpose. As a result, register allocation also should be considered carefully in different embedded systems, depending on their applications and actual requirements. In this thesis, we study the different requirements for different embedded systems. We propose system-specific register allocation approach to meet the need of each type of embedded system.
First, we study the schedule length minimization for loop programs in embedded systems. Our study is based on loop scheduling for schedule length reduction. A cooperative register allocation and loop scheduling approach is proposed that combines the phase ordering problem of register allocation and loop scheduling. Starting with an aggressive schedule, we show that a program’s schedule length can effectively reduced by considering register allocation and loop scheduling.

Second, we study the application of real-time embedded systems with clustered VLIW architecture. On one hand, worst-case execution time (WCET) is an important real-time constraint for real-time systems. On the other, the phase ordering problem of register allocation, instruction scheduling, and cluster assignment should be carefully considered in systems with clustered VLIW architecture such that the effort on WCET reduction is effective. For this type of machine, we propose a WCET-aware re-scheduling register allocation technique to achieve WCET minimization. The effects of register allocation, scheduling, and cluster assignment on the quality of generated code are taken into account for WCET minimization. We show that starting with an aggressive initial scheduling and an initial cluster assignment solution for virtual registers, a program’s WCET can be effectively reduced by integrating these three activities into a single phase.

Third, we study the application of real-time cyber-physical systems (CPSs). There are two challenges for CPSs. First, CPSs often include a number of sensor nodes. Update of preloaded code on remote sensor nodes powered by batteries is extremely energy-consuming. The code update issue in the energy sensitive CPS must be carefully considered. Second, CPSs are often real-time embedded systems so that WCET is an important concern in real-time system design. While existing works only consider one of these two challenges at a time, we propose a compiler level optimization, namely join WCET and update conscious compilation technique to jointly consider WCET and code update for CPSs such that a balanced solution with minimal WCET and minimal
code update can be achieved.

Finally, we study the application of non-volatile memories for embedded systems. Non-volatile memories are good candidates for DRAM replacement as main memory in embedded systems and they have many desirable characteristics. However, we need to address two challenges. First, the lifetime of some of the non-volatile memories is limited by the number of erase operations. Second, read and write operations have asymmetric speed or power consumption in non-volatile memory. To process these two problems, we propose register allocation technique with re-computation to reduce the number of store instructions, which will in turn reduce write activities on non-volatile memory and extend the lifetime.

To summarize, we show that for different application of different embedded systems, an effective register allocation solution can have large effect on the system performance. By specifically considering their requirements during register allocation, a significant performance improvement can be achieved.
## Contents

Contents vi

List of Tables xi

List of Figures xiii

Nomenclature xvi

1 Introduction 1
   1.1 Compiler and Compilation .................................. 1
   1.2 Register Allocation ........................................ 2
   1.3 Embedded Systems .......................................... 5
       1.3.1 Real-time Embedded Systems with Clustered VLIW Architecture ........................................ 7
       1.3.2 Real-time Cyber-physical Systems .................... 9
       1.3.3 Non-volatile Memory in Embedded Systems .......... 10
   1.4 Outline of the Thesis ...................................... 11

2 Minimizing Schedule Length via Cooperative Register Allocation and Loop Scheduling for Embedded Systems 13
   2.1 Introduction of a Loop Program .......................... 13
   2.2 Background .................................................. 15
       2.2.1 Traditional Register Allocation and Loop Scheduling .................................................. 15
       2.2.2 The Phase Ordering Problem between Register Allocation and Scheduling .......................... 17
## CONTENTS

2.2.3 Related Works with Loop Scheduling for Minimizing Overall Schedule Length .................................. 18

2.3 Problem Analysis .................................. 19

2.3.1 Problem Formulation ............................ 19

2.3.2 Live Range Splitting ............................ 21

2.3.3 Motivational Example ............................ 22

2.4 Algorithm .................................. 22

2.5 Experimental results ............................ 28

2.6 Summary .................................. 29

3 WCET-aware Re-scheduling Register Allocation for Real-time Embedded Systems with Clustered VLIW Architecture ................................................. 30

3.1 Challenges of WCET reduction in real-time embedded systems with clustered VLIW architecture .............................. 30

3.2 Background Information regarding the Phase Ordering Problem of Register Allocation, Instruction Scheduling, and Cluster Assignment .............................. 32

3.3 Overview .................................. 33

3.3.1 Overview of the Proposed WCET-aware Rescheduling Register Allocation Technique .................. 33

3.3.2 Motivational Example ............................ 35

3.4 Problem Analysis .................................. 37

3.4.1 Strategy for WCET-based Register Allocation and CFG Node Selection .................................. 37

3.4.2 Strategy for Local and Global Variable Allocation .... 39

3.4.3 Strategy for Block-Based Instruction Re-scheduling and Cluster Assignment .............................. 41

3.4.3.1 Maintaining the Dependency Relationship of Instructions during Re-scheduling .......... 42

3.4.3.2 Formulating the Cost Model .................... 43

3.4.3.3 Re-scheduling the Instructions and the Corresponding Move Operations .................... 44

3.5 Algorithm .................................. 45

3.6 Experimental Results ............................ 46
CONTENTS

3.6.1 Experimental Environment .................................. 46
3.6.2 Worst-case Execution Time .................................. 47
3.6.3 Average-case Execution Time ................................. 48
3.6.4 Parameter Sensitivity Analysis with $\alpha$ and $\beta$ .......... 49
3.7 Summary .......................................................... 50

4 Joint WCET and Update Conscious Compilation for Cyber-physical Systems 53
4.1 Challenges for Real-time Cyber-physical Systems ............ 53
4.2 Background and Motivational Example ........................ 56
4.2.1 Conventional Re-programming Technique and the Update
        -Conscious Compilation Technique .......................... 56
4.2.2 Traditional WCET-aware Register Allocation Technique .. 57
4.2.3 Motivational Example ......................................... 58
4.3 Problem Formulation .............................................. 61
4.3.1 Problem Description ........................................... 61
4.3.2 Problem Formulation ........................................... 62
4.4 Problem Analysis ................................................... 63
4.4.1 Overview of the Proposed Technique ....................... 64
4.4.2 Strategy for CFG Node Selection ............................. 65
4.4.3 A Model for CFG Node Selection ............................. 67
4.5 Algorithm .......................................................... 69
4.5.1 Algorithm for Problem$^a$ ..................................... 69
4.5.2 Algorithm for Problem$^b$ ..................................... 70
4.6 Experimental results .............................................. 71
4.6.1 WCET Versus Code Similarity ................................. 71
4.6.2 Code Difference among Three Different Approaches ..... 72
4.6.3 WCET Benefit of WUCC from UCC ............................ 73
4.6.4 Relative WCET Increments under Given Code Similarity
        Requirements ................................................... 74
4.7 Summary .......................................................... 75
5 The Application of Non-volatile Memory for Embedded Systems 76

5.1 Challenges for Non-volatile Memory .............................. 76
5.2 Problem Formulation .............................................. 78
  5.2.1 Overview of Re-computing Approach ...................... 79
  5.2.2 Register Allocation for Write Activity Reduction .......... 80
  5.2.3 Motivating Example ........................................ 83
5.3 Problem Analysis .................................................. 85
  5.3.1 Re-computing Variables ................................... 85
  5.3.2 Live Range Splitting of Variables with Long Live Ranges
       before Graph Coloring ...................................... 87
  5.3.3 Potential Spill Selection Strategy in Programs with Multi-
       ple Basic Blocks during Graph Coloring .................. 87
  5.3.4 Re-computation Delay Reduction on Systems with Multiple
       Functional Units ............................................ 89
5.4 Algorithm ........................................................ 90
5.5 Experimental Results ............................................. 92
  5.5.1 Write Reduction under the Proposed Approach ............ 92
  5.5.2 The Overall Scheduling Length of the Benchmark Execu-
       tion under the Proposed Approach ........................ 95
  5.5.3 Impact of the Number of Functional Units and Available
       Registers on the Number of Spills .......................... 99
5.6 Summary .......................................................... 100

6 Conclusions ......................................................... 102
  6.1 Contributions .................................................. 102
  6.2 Future Work ..................................................... 104

Bibliography ........................................................ 105