Semantic Abstraction of IA-64 Multimedia Instructions

Xuemeng Zhang Rongcai Zhao Jianmin Pang
National Digital Switching System Engineering & Technological R&D Center
ZhengZhou HeNan China 450002
zhangxuemeng_0@163.com

Abstract

This paper presents an effective approach to semantic abstraction of IA-64 multimedia instructions, which is important to binary translation. In particular, the authors take example for multimedia instructions used in motion estimation to describe the details of generating higher-level intermediate representation from IA-64 assembly language. Finally, the originality and soundness of the approach is proved.

Key words multimedia instructions; IA-64; semantic abstraction; binary translation

1. Introduction

The new features of IA-64 multimedia instructions make it difficult to abstract the semantics of machine instructions to higher-level intermediate representations, which is important to binary translation [1, 2]. This paper presents an effective approach to the problem, improving existing technologies according to characteristics of IA-64 multimedia instruction set [3]. The techniques described here are suitable not only to binary translators but also to decompilers.

2. New features of IA-64 multimedia instructions

A new set of multimedia instructions introduced into IA-64 architecture implements SIMD (Single Instruction Multiple Data) parallelism, which obtains a significant performance improvement in multimedia applications [3]. IA-64 SIMD Extensions provides new data representation and enhanced instruction set to enhance the performance of applications.

In IA-64, an integer multimedia instruction can be 1, 2 or 4 bytes form, thus multiple subwords can be accommodated in a single register, which is 64-bit long.

3. Semantic abstraction of IA-64 multimedia instructions

UQBT (The University of Queensland Binary Translator) developed a simple and parsable Semantic Specification Language (SSL) [1, 2]. The syntax of SSL is defined in EBNF (Extended-Backus-Naur-Form), and semantics of SSL is described in natural language integrated with examples from the 80286 and SPARC architectures. However, SSL is limited in its ability to model other architectural issues.

To overcome the problem, we study the characteristics of IA-64 instruction set and semantic specification technology of UQBT. Our approach improves in two aspects: first of all, we modify and extend the syntax of SSL according to the new features of IA-64 architecture. Secondly, the semantics of some multimedia instructions is so complicated that we add new algorithms to meet the demand of semantic abstraction effectively.

3.1. Semantic abstraction

The aim of semantic abstraction of machine instructions is to automatically generate suitable code for an intermediate representation that is to be used during the further analysis stage [1], hence separating machine-independent issues from the machine-code stage. Since the intermediate representations don’t have the feature of subword parallelism, the subword parallelism of the low-level assembly instructions can be eliminated by semantic abstraction. Thus by further
analysis, the intermediate representations can be promoted to the target high-level language.

The semantics of each machine instruction is abstracted by the extended SSL, which model basic transfers of information via registers and memory locations. The syntax of SSL is extended according to the new features of IA-64 architecture. For example, we extend the EBNF of a instruction as follows:

```
instr:      lhs_def stmt_list
lhs_def:    INSTR_NAME list_parm /* single instruction */
            | INSTR_NAME DECOR list_parm /* single instruction with decorated name*/
list_parm:  list_parm ',' PARM
            | PARM
```

3.2. An optimal algorithm

As an example, we present a practical method to optimize code of the important motion estimation algorithm in real-time video MPEG-2 encoding in instruction level. Motion estimation has a great deal of parallelism that we could slice in its processing of 16*16 block matching. We replace groups of instructions by a single multimedia instruction performing the same operation:

```
MPEG2 motion estimation:  //9 instructions, 3 groups
//Do PSAD for a row, accumulate results
ld8  r32=[r22], r21
ld8  r33=[r23], r21
ld8  r34=[r24], r21 ;;
ld8  r35=[r25], r21
psad1 r32=r32, r34
psad1 r33=r33, r35 ;;
padd4 r36=r36, r32
padd4 r37=r37, r33
Br.cloop.many.spkt Psad_top ;;
```

Taking the padd4 instruction for example, we present the semantic abstraction of integer multimedia instructions using extended SSL. The padd4 is the four_byte, modulo_form of the paddn instruction. The sets of subwords from the two source operands are added in parallel, and the results are placed in the destination register. The following fragment of SSL abstracts the semantics of the psad4 instruction, which operates on two 32-bit subwords in parallel:

```
(qp) padd4 r1, r2, r3
*64* r[r1] := (r[r2]<32:63) + (r[r3]<32:63)
*32* r[tmpi] := (r[r2]<0:31) + (r[r3]<0:31)
*64* r[r1] := r[r1] << 32 + r[tmpi];
```

4. Implementation and preliminary results

We have implemented our techniques in a static binary translator from IA-64 to Alpha (I2A). We reverse the executable programs in ELF (Executable and Linking Format) binary format to the target C codes on IA-64 and the output C files are converted to machine codes on Alpha. Source binary programs are compiled using gcc 2.95.3 –O4, running Redhat Linux AS 2.1, Kernel 2.4.18. Translated codes are compiled with gcc 2.96 –O2 on Redhat Linux 7.2, Kernal 2.4.9. Our techniques are evaluated by ten benchmarks selected from the Berkeley multimedia workload [3]. The preliminary results show that the results of the translated codes running on Alpha are just the same as the results of the source programs running on IA-64. This proves the technical soundness. From our analysis, the numbers of IA-64 multimedia instructions for each application are listed in Table1.

<table>
<thead>
<tr>
<th>Kernel Name</th>
<th>Data Type</th>
<th>INT</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Block</td>
<td>8-bit(U)</td>
<td>181</td>
<td>21</td>
</tr>
<tr>
<td>Block Match</td>
<td>8-bit(U)</td>
<td>294</td>
<td>45</td>
</tr>
<tr>
<td>Clip Test and Project</td>
<td>FP</td>
<td>47</td>
<td>410</td>
</tr>
<tr>
<td>Color Space Conversion</td>
<td>8-bit(U)</td>
<td>78</td>
<td>16</td>
</tr>
<tr>
<td>DCT</td>
<td>16-bit(S)</td>
<td>110</td>
<td>20</td>
</tr>
<tr>
<td>FFT</td>
<td>FP</td>
<td>81</td>
<td>906</td>
</tr>
<tr>
<td>Inverse DCT</td>
<td>16-bit(S)</td>
<td>366</td>
<td>164</td>
</tr>
<tr>
<td>Max Value</td>
<td>32-bit(S)</td>
<td>39</td>
<td>18</td>
</tr>
<tr>
<td>Mix</td>
<td>16-bit(S)</td>
<td>820</td>
<td>268</td>
</tr>
<tr>
<td>Quantize</td>
<td>FP</td>
<td>12</td>
<td>314</td>
</tr>
</tbody>
</table>

5. Conclusions and Future work

In this paper a semantic abstraction approach has been presented. Future work includes the future development of a more compact and optimized model to reduce the codes of intermediate representation and more multimedia applications to evaluate the techniques.

6. References