An Instruction Set Architecture Based Code Compression Scheme for Embedded Processors

Sreejith K Menon Priti Shankar
Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560012, India.

We propose a general purpose code compression scheme for embedded systems, based on the instruction set architecture and report results on the Intel StrongARM, a low-cost, low-power RISC architecture and TI TMS320C62x, a widely used VLIW architecture. Fast decompression techniques are explored to improve the decompression overhead of the compression scheme. Compression ratios ranging from 68% to 75% were obtained for TMS320C62x and 69% to 78% for the StrongARM processor.

The basic idea of the compression scheme is to divide the instructions into different logical classes and to build multiple dictionaries for them. The size and the number of multiple dictionaries are fixed for a given processor and are determined by the *partitioning algorithm* which works over the instruction set architecture supplied as input. Frequently occurring unique instruction segments are inserted into the dictionaries and the instructions are encoded as pointers to the respective entries. An *opcode,* which helps in fast decompression, is attached to an instruction segment to identify its logical class and the dictionary to be accessed.

The compression results can be improved slightly by incorporating segments that are at a small Hamming distance from the dictionary entries. However, a major improvement in the compression results can be obtained with *Multilevel dictionaries,* where dictionaries of frequently used instruction classes are divided into two unequal parts - a small primary dictionary section and a secondary dictionary section. The main advantage is that highly frequent entries can be represented using very small pointers.

Decompression steps consist of 1. Identifying the logical class of the instructions and the corresponding dictionaries, 2. Sending the offset to the respective dictionaries and 3. Retrieving the original instructions. The decompression speed can be improved by decoding the instructions in a block in parallel. Starting addresses of the different instructions have to be determined at the initial stage. This can be achieved by dividing the instructions into two sections - an *opcode section* and a *segment section*. The size of the opcode section is fixed as the size of the block and the size of the opcode are fixed for a processor, and this helps in the parallel decompression of instructions within a block.

References

1. Prakash J, Sandeep C, Y N Srikant, Priti Shankar. A Simple and Fast Scheme for Code Compression for VLIW Processors *Data Compression Conference, 2003.*

2. Montserrat Ros and Peter Sutton. Code Compression Based on Operand-Factorization for VLIW Processors. *Data Compression Conference, 2004.*

Proceedings of the 2005 Data Compression Conference (DCC'05)
1068-0314/05 $ 20.00 **IEEE**