Intruction looping, an extension to conditional execution

B. Ulmann
ulmann@fafner.zdv.uni-mainz.de

February 20, 1998

Abstract

The following article describes an easy to implement but very powerful extension to simple conditional execution based program flow control as used for example in the ARM RISC processors and others.

1 Conditional execution

Some processors provide the ability of the conditional execution of instructions according to the state of one or more bits obtained from the status register. The first example of such an architecture known to the author is the Austrian computer "Mailüfterl" (cf. [4]) which was completed in 1958. On this machine the execution of every instruction could be made dependent on one out of fifteen different conditions coded in a special field contained in the instruction word.

The same scheme is used in some modern processors as for example the ARM RISC machine (cf. [2]). Like "Mailüfterl" this machine also uses a conditional execution mechanism which is controlled by one out of sixteen condition codes, including a "Don't care" condition for instructions which always have to be executed.

Since many of the conditional branches used in machine language on conventional architectures which are not employing conditional execution, are used just for skipping one or only a few instructions, conditional execution can significantly shrink the size of a program as well as the execution speed.

2 Instruction looping

Obviously the usage of conditional execution eliminates one performance bottleneck by avoiding lots of branches. But most problems do not only involve instructions to be skipped but also groups of instructions which resemble loops, and some of these loops are very small ones. A lot of these involve only one or a few instructions which are executed again and again.

The Honeywell Bull Series 60, for example, implements intructions like `RPT` and `RPD` which are used to repeat the following respectively the following two machine instruction(s) until a specific condition is fulfilled, see [1], p. 4-145ff. This scheme is employed by modern DSP-architectures like the TI TMS320C54x series, too. These processors use instructions like `RPT` and `RPTB`. The latter is used to repeat a complete block of instructions a specified number of times. Using the `RPT` command, which repeats only one instruction, avoids unnecessary instruction decoding cycles, thus significantly speeding up execution of single instruction loops like copying blocks of data etc. As soon as more than one instruction has to be repeated this speedup is lost since the affected instructions in the block have to be decoded over and over again.

In conjunction with the concept of VLIW-architectures this leads to the idea of **instruction looping** – a direct extension of conditional execution. Since these architectures combine several microinstructions in a single machine instruction they are ideally suited for the implementation of a repeat mechanism. Clearly this technique gains effect in speeding up program execution as more instructions are contained in a single instruction word.

3 An example for mixed conditional execution and instruction looping

In the following an instruction execution control field is described which can be used for coding conditional execution and instruction looping in an ARM-like architecture. On the ARM RISC processor there are eight different condition codes which leads to a three bit field for the selection of a specific condition. Another bit is used for inverting the selected value resulting in a total of 16 conditions available.

At this point an additional bit in the instruction word can be used to switch between conditional execution and instruction looping. If the latter is selected for flow con-

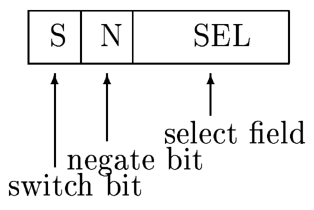

Figure 1: A hypothetic condition code field

trol of an instruction, this instruction is executed as long as the desired condition is fulfilled. So the condition code field of a processor implementing this control mechanism might look like the one shown in Fig. 1.

4 Conclusion

On architectures containing more than one instruction per instruction word, like VLIW-machines, the above described scheme of instruction looping can result in a significant speedup of program execution time.

Currently we think about a simulation of a simple VLIW-architecture containing instruction looping to be able to measure the effects of this technique on compiler generated code.

It may be concluded that instruction looping provides a very powerful tool in program flow control which can be easily implemented and promises great influences on processing speeds at the cost of only one extra bit in each instruction word if extending an conditional execution architecture.

References

[1] Honeywell Bull GmbH, *Series 60 (Level 68) Multics Processor Manual*, April 1979.

[2] Alex van Someren and Carol Atack, *The ARM RISC Chip, A Programmer's Guide*, Addison Wesley, 1994

[3] Texas Instruments, *Training and Technical Documentation CD*, TMS320C54x.

[4] H. Zemanek, *Central European Prehistory of Computing*, published in *A History of Computing in the Twentieth Century*, pp. 601...605, edited by N. Metropolis, J. Howlett, and Gian-Carlo Rota, Academic Press, 1980.