Optimization of x265 encoder using ARM SVE

AOKI Ryosuke
ryosuke-aoki@uec.ac.jp
The University of Electro-Communications
Tokyo, Japan

MURAO Hirokazu (Advisor)
murao@cs.uec.ac.jp
The University of Electro-Communications
Tokyo, Japan

ABSTRACT

We optimized several heavily used functions in x265, an open source implementation of H.265/HEVC, using SVE instructions. The result showed that our implementation reduced the number of instructions executed up to 52% compared to the code generated by GCC from the original C++ source code. The implementation also scales better with the vector length than the original code.

In our poster, implementation details will be illustrated, and the detailed result of benchmark for every function will be shown.

KEYWORDS

H.265/HEVC, Scalable Vector Extension, x265

1 INTRODUCTION

Scalable Vector Extension (SVE) [4] is a vector extension for ARMv8-A, a 64bit CPU architecture developed by ARM Ltd. The most notable feature of SVE is its scalable vector registers. SVE does not specify the exact length of these registers, but let the implementation choose the length, from 128 bits up to 2048 bits. SVE adopts the vector-length agnostic (VLA) programming model, making it possible to run programs on every SVE platform with different vector length, without the need of recompiling. In this April, ARM announced SVE2, a new extension based on SVE, which has new instruction to vectorize DSP and multimedia SIMD codes [3]. SVE and its unique VLA programming model are expected to be deployed in various fields, not limited to supercomputing.

H.265/HEVC is a video codec developed and standardized in 2013. H.265/HEVC achieved higher compression efficiency compared to its predecessor H.264/AVC, at the expense of significantly higher computational cost.

2 OPTIMIZING X265 FOR SVE

x265 [2] is an open source implementation of H.265/HEVC encoder. It is mostly written in C++, but several frequently used subroutines, called "primitives", are implemented in assembly. From our profiling of x265 on an AArch64 machine, we found out 40% executed instructions of the total encoding process came from only 10 functions. From these functions we chose computationally intensive functions to optimize, namely for sum of absolute transformed differences (SATD), interpolation, and DCT.

Our implementation is based on SVE's VLA programming model, and the performance is expected to scale with the vector lengths equipped with platforms actually used. GCC doesn't vectorize functions mentioned above, so we reimplemented them in assembly code. In our implementation of DCT32 (processes 32×32 16bit interger data), 2 lines in the input data is processed using contiguous 128 bit part of SVE registers. As the SVE longer gets longer, the more lines are processed in one iteration. For example on a 2048bit SVE machine, DCT32 can be computed only one iteration. SATD and interpolation functions are optimized in the simillar way.

3 EVALUATION

We executed and analyzed the encoder using Arm Instruction Emulator (ArmIE) [1]. We used a short test clip to count the whole numbers of instructions executed in encoding. The result (Figure 1) shows that our implementation reduces up to 52% of instructions, compared with the original version. Also we observed that the number of executed instructions decrease as the vector gets longer.

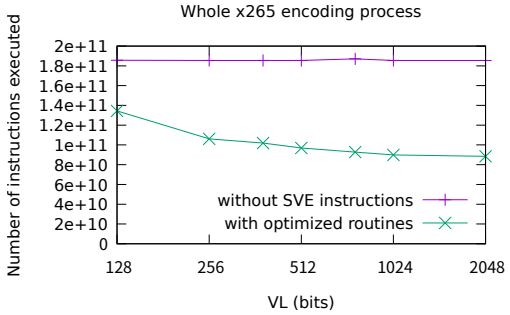

Figure 1: Total number of instructions executed during test clip encoding.

4 CONCLUSION

We optimized several heavily used functions in x265 using SVE instructions, and achieved the reduction of the number of instructions executed. The executed instruction count also scales better with vector length. Using instructions introduced in SVE2, like pairwise operations, might reduce more instructions. Investigation of SVE2 and the comparison with our result are left for future study.

5 ACKNOWLEDGMENTS

We would like to thank Mitsuhisa Sato, Jinpil Lee, and other researchers of RIKEN for accepting the author as an intern and giving advice on writing and optimizing programs for SVE.

REFERENCES

[1] Arm Limited. [n. d.]. Arm Instruction Emulator | Analyzing SVE programs – Arm Developer. https://developer.arm.com/tools-and-software/server-and-hpc/arm-architecture-tools/arm-instruction-emulator/analyzing-sve-programs
[2] MulticoreWare Inc. [n. d.]. x265 HEVC Encoder / H.265 Video Codec. http://x265.org/
[3] N. Stephens. 2019. New Technologies in the Arm Architecture.
[4] N. Stephens, S. Biles, M. Boettcher, J. Eapen, M. Eyole, G. Gabrielli, M. Horsnell, G. Magklis, A. Martinez, N. Premillieu, A. Reid, A. Rico, and P. Walker. 2017. The ARM Scalable Vector Extension. *IEEE Micro* 37, 2 (March 2017), 26–39. https://doi.org/10.1109/MM.2017.35