Even Faster Hashing on the Pentium

Antoon Bosselaers

Katholieke Universiteit Leuven, Dept. Electrical Engineering-ESAT
Kardinaal Mercierlaan 94, B-3001 Heverlee, Belgium

antoon.bosselaers@esat.kuleuven.ac.be

13 May 1997, updated 13 November 1997

Abstract. In this short note we present an improvement of about 15% over our performance figures for the MD4-family of hash functions as presented at Crypto'96. The improvement is obtained by substituting n-cycle instructions by n 1-cycle instructions, and reducing the number of instructions by means of the super-`add` instruction `lea`, thereby carefully avoiding the dreaded AGI.

In [BGV96] we presented optimized implementations of MD4, MD5, SHA-1, RIPEMD, RIPEMD-128 and RIPEMD-160 on Intel's Pentium processor. The goal of this short note is to present an improvement of about 15% over these figures. We refer to [BGV96] for an explanation of the terminology used, and to [BGV97] for a detailed critical path analysis of these algorithms.

In terms of processor pipeline stages the critical path of these new implementations is slightly longer (on the average about 16 stages). However, this lengthening allows us to substitute the single remaining 2-cycle instruction in each step by 2 single cycle instructions. This in itself doesn't reduce the total number of clock cycles (and moreover requires an additional auxiliary register), but if we can move one of these instructions partially or entirely out of the critical path by pairing it with another single cycle instruction, then the overall effect will be a reduction of the total number of clock cycles. In view of the already high percentage of **`simple`** paired instructions of the old implementations [BGV96, Table 4], this seems to be an impossible task. However, here the super-**`add`** instruction **`lea`** comes to our rescue, by allowing us to combine 2 single cycle **`add`** instructions into a single instruction taking only 1 cycle, provided the 1-cycle address generation interlock (AGI) penalty can be avoided. Miraculously, this turns out to be the case, as illustrated in Table 2 for a round 1 step of MD5, updating [BGV96, Table 3].

Table 3 is the updated version of [BGV96, Table 4]. All implementations now only use 1-cycle instructions, except for SHA-1 that uses the **`bswap`** instruction taking an additional cycle to decode due to the $\texttt{0F}_\texttt{x}$-prefix. A value for the cycles per instruction (CPI) of close to 0.5 is therefore an indication of the high percentage of **`simple`** paired instructions in the code. Table 1 gives a better idea of the resulting improvement.

Algorithm	Size (bytes)	Speed (Mbit/s) [BGV96]	this note	Factor this note-[BGV96]
MD4	1190	166.8	190.6	1.14
MD5	1713	113.7	136.2	1.20
SHA-1	4323	48.7	54.9	1.13
RIPEMD	2291	82.7	95.7	1.16
RIPEMD-128	2929	64.0	77.6	1.21
RIPEMD-160	4808	39.9	45.3	1.14

Table 1. Code size and hashing speeds of the different compression functions on a 90 MHz Pentium for our Assembly implementations of both [BGV96] and this note. The code size only refers to the improved implementations. Code and data are assumed to reside in the on-chip caches. The figures are independent of the buffer size as long as it, together with the local data, fits in the 8-Kbyte on-chip cache.

$$A := B + (A + ((B \wedge C) \vee (\overline{B} \wedge D)) + X_i + K)^{\lll s}$$

Instructions	Cycles	Instructions	Cycles	Instructions	Cycles
⋮		⋮		⋮	
`add ebx,ecx`	1	`add ebx,ecx`	1	`add ebx,ecx`	1
`mov edi,ecx`	paired	`xor edi,edx`	paired	`xor edi,edx`	paired
`xor edi,edx`	1	`add eax,X[esi]`	2	`lea eax,[eax+ebp+K]`	1
`and edi,ebx`	1	`and edi,ebx`	paired	`and edi,ebx`	paired
`xor edi,edx`	1	`add eax,K`	1	`mov ebp,[esi+4]`	1
`add eax,edi`	1	`xor edi,edx`	paired	`xor edi,edx`	paired
`add eax,X[esi]`	2	`add eax,edi`	1	`add eax,edi`	1
`add eax,K`	1	`mov edi,ebx`	paired	`mov edi,ebx`	paired
`rol eax,s`	1	`rol eax,s`	1	`rol eax,s`	1
`add eax,ebx`	1	`add eax,ebx`	1	`add eax,ebx`	1
`mov edi,ebx`	paired	`xor edi,ecx`	paired	`xor edi,ecx`	paired
⋮		⋮		⋮	

Cycles per instr.	1.00	Cycles per instr.	0.67	Cycles per instr.	0.56
V pipe use	11%	V pipe use	44%	V pipe use	44%
Paired simple instr.	25%	Paired simple instr.	100%	Paired simple instr.	100%

Table 2. Implementation of a round 1 step of MD5 on a Pentium processor. The chaining variable A, B, C, D is stored in registers `eax` through `edx`. The optimized expression $((C \oplus D) \wedge B) \oplus D$ for the multiplexer is used. The left column shows the straightforward implementation. In the middle column the instructions are rearranged in such a way that all pairable instructions are paired. In the right column the 2-cycle instruction `add eax,X[esi]` is substituted by the 2 1-cycle instructions `mov ebp,X[esi]` and `add eax,ebp`. Subsequently, the latter instruction is combined with `add eax,K` by means of the super-add instruction `lea` into a single 1-cycle instruction: `lea eax,[eax+ebp+K]`. Memory read access is as indicated if the data resides in the on-chip cache.

Algorithm	MD4	MD5	SHA-1	RMD	RMD-128	RMD-160
Instructions	417	577	1469	832	1024	1639
% instr. in V pipe	42.21	41.59	44.11	42.31	42.19	38.19
% Paired `simple` instr.	95.39	93.57	99.46	95.65	96.54	94.92
% Memory ref.'s	13.88	12.82	25.80	13.94	14.45	13.18
Cycles	241	337	837	480	592	1013
Cycles per instr.	0.58	0.58	0.57	0.58	0.58	0.62
Speed-up factor	1.73	1.71	1.76	1.73	1.73	1.62

Table 3. Performance figures on a Pentium for the improved implementations of the compression function of the 6 members of the MD4 hash function family. Both code and data are assumed to reside in the on-chip caches. All figures are independent of the processor's clock speed. The speed-up factor is with respect to a (hypothetical) execution of the same code on a non-parallel architecture under otherwise unchanged conditions.

References

[BGV96] A. Bosselaers, R. Govaerts, J. Vandewalle, "Fast hashing on the Pentium," *Advances in Cryptology, Proceedings Crypto'96, LNCS 1109*, N. Koblitz, Ed., Springer-Verlag, 1996, pp. 298–312.

[BGV97] A. Bosselaers, R. Govaerts, J. Vandewalle, "SHA: a design for parallel architectures?," *Advances in Cryptology, Proceedings Eurocrypt'97, LNCS 1233*, W. Fumy, Ed., Springer-Verlag, 1997, pp. 348–362.

[RMD160] `http://www.esat.kuleuven.ac.be/~bosselae/ripemd160.html` contains more information on the MD4 hash function family, in particular on RIPEMD-160.