Instruction latencies and throughput for AMD and Intel x86 processors

Torbjörn Granlund

2014-07-20

[This report is work-in-progress. A newer version might be available here: `http://gmplib.org/~tege/x86-timing.pdf`]

In this short report we present latency and throughput data for various x86 processors. We only present data on integer operations. The data on integer MMX and SSE2 instructions is currently limited. We might present more complete data in the future, if there is enough interest.

There are several reasons for presenting this report:

1. Intel's published data were in the past incomplete and full of errors.
2. Intel did not publish any data for 64-bit operations.
3. To allow straightforward comparison of AMD and Intel pipelines.

The here presented data is the result of extensive timing tests. While we have made an effort to make sure the data is accurate, the reader is cautioned that some errors might have crept in.

1 Nomenclature and notation

L*NN* means *latency for NN-bit operation.* T*NN* means *throughput for NN-bit operation.*

The term *throughput* is used to mean *number of instructions per cycle of this type that can be sustained.* That implies that more throughput is better, which is consistent with how most people understand the term. Intel use that same term in the exact opposite meaning in their manuals.

The notation "P6 0-E", "P4 F0", etc, are used to save table header space. P4 means Pentium 4, which has family number F (in hexadecimal). Then there are model numbers 0, 1, 2, etc, where 4 and above means the core has the AMD defined 64-bit instructions. P6 means family 6, which is a big happy family with Pentium Pro, Pentium 2, Pentium 3, Pentium M, Core and Core 2. For the P6 family, model numbers (0-F in hexadecimal) and marketing names are not completely related; some method in the madness can be found in the table at the end of this document.

In some of these tables, Core 2 numbers hide under the P6 F moniker (family 6, model F).

Family and model numbers are returned by the `cpuid` instruction.

2 How data was generated

The throughput numbers here have been generated using a loop with the measured instruction repeated many times. This results in some cases in lower numbers than those claimed by the processor vendors. Our measurement method requires that the entire pipeline (cache, decode, issue, execution, complete) can sustain the indicated execution rate.

Measuring performance for immediate operands is tricky, as x86 encodes small immediate operands specially, and as different cores are optimised for different ranges.

An example of how complex this can be is that Pentium F0-F2 can sustain 3 `add r,i` per cycle for $-32768 \leq i \leq 32767$, but for larger immediate operands it can sustain only about 3/2 per cycle. Pentium F3 behaves similarly, but for $-65536 \leq i \leq 65535$. This is not related to x86 instruction encoding; the exact same encoding is used for greater immediate operands. (Operands that fit into a byte have a special encoding, though.)

3 Comments on table data

The Athlon and P6 results are somewhat orthogonal. The same cannot be said about the Pentium 4 results.

The Pentium 4 performance for 64-bit right shifts is really poor. 64-bit left shift as well as all 32-bit shift have acceptable performance. It appears that the data path from the upper 32 bits to the lower 32 bit of the ALU, is not well designed.

On Pentium 4, it is possible to reach much better 32-bit `shl/sal/shr/sal r,c` performance if dummy shift instructions with immediate counts are strategically inserted. The best measured resulting performance is about 1.4 instructions/cycle.

On AMD K8, the odd man out is `test r,i`. The reason is that the encoding of this instruction doesn't provide any short immediate version. The L1 instruction cache can provide only 16 bytes per clock, but 3 `test r,i` need 18 or 21 bytes (depending on register). The AMD K10 has a wider bus between the L1 instruction cache and decoders, and can sustain 3 insns/cycle even for these long instructions. (One might think it is more than a little silly to measure three consecutive `test` instructions, since these instructions are pointless without a branch or set-on-condition instruction. But it gives some idea of the resource usage of these instructions, and we prefer to keep the measurements orthogonal between instructions.)

The data for `adc` and `sbb` are approximate. A long chain of these instructions would have a recurrent dependency on the carry flag. Our sequences break such dependencies by regularly executing a plain `add`, relying on out-of-order execution and carry flag renaming. It is likely that slightly better performance could be achieved than what is here indicated.

Sequences with `adc` and `sbb` interleaved with `inc` or `dec` run poorly on all P6 processors (including Core processors), they result in a pipeline hiccough of about 12 cycles. That sort of problems cannot be represented in the table numbers.

4 Your feedback

Please send feedback about this report to tg at gmplib (dot) org.

		Intel P4 F4				Intel Core 2				Intel NHM				Intel SBR				Intel HWL				AMD K8-K9				AMD K10				AMD BD1				Intel Atom				VIA Nano			
		L32	T32	L64	T64	L32	T32	L64	T64	L32	T32	L64	T64	L32	T32	L64	T64	L32	T32	L64	T64	L32	T32	L64	T64	L32	T32	L64	T64	L32	T32	L64	T64	L32	T32	L64	T64	L32	T32	L64	T64
add	r,ri	1	2.5	1	2.5	1	3	1	3	1	3	1	3	1	3	1	3	1	4	1	4	1	3	1	3	1	3	1	3			1	2	1	2	1	2	1	2	1	2	1	2
sub	r,ri	1	2.5	1	2.5	1	3	1	3	1	3	1	3	1	3	1	3	1	4	1	4	1	3	1	3	1	3	1	3			1	2	1	2	1	2	1	2	1	2	1	2
and	r,r	1	2	1	2	1	3	1	3	1	3	1	3	1	3	1	3	1	4	1	4	1	3	1	3	1	3	1	3			1	2	1	2	1	2	1	2	1	2	1	2
or	r,r	1	2	1	2	1	3	1	3	1	3	1	3	1	3	1	3	1	4	1	4	1	3	1	3	1	3	1	3			1	2	1	2	1	2	1	2	1	2	1	2
xor	r,r	1	2	1	2	1	3	1	3	1	3	1	3	1	3	1	3	1	4	1	4	1	3	1	3	1	3	1	3			1	2	1	2	1	2	1	2	1	2	1	2
inc	r	1	1	1	1	1	3	1	3	1	3	1	3	1	3	1	3	1	4	1	4	1	3	1	3	1	3	1	3			1	2	1	2	1	2	1	2	1	2	1	2
dec	r	1	1	1	1	1	3	1	3	1	3	1	3	1	3	1	3	1	4	1	4	1	3	1	3	1	3	1	3			1	2	1	2	1	2	1	2	1	2	1	2
neg	r	1	2	1	2	1	3	1	3	1	3	1	3	1	3	1	3	1	4	1	4	1	3	1	3	1	3	1	3			1	2	1	2	1	2	1	2	1	2	1	2
not	r	1	1.7	1	1.7	1	3	1	3	1	3	1	3	1	3	1	3	1	4	1	4	1	3	1	3	1	3	1	3			1	2	1	2	1	2	1	2	1	2	1	2
imul	r,ri	10	1	10	1/2	3	1	5	1/2	3	1	3	1	3	1	3	1	3	1	3	1	3	1	4	1/2	3	1	4	1/2		6	1/4	5	1/2	14	1/11	4	1	5	1/2
mul	r	11	1/2	12	1/10	5	0.67	8	1/4		1/2	10^5	1/2	4	1/2	4^6	1	4	1/2	3	1	3	1	5^1	1/2	3	1	5^1	1/2		7^x	1/4	9	1/9	18	1/18	8	1/2	12	1/2
mulx	r,r,r	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-			3	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
div	r	80^2	1/34	161^2	1/151	40		116^4		26		89		26		92		28		95		39	1/39	71	1/71	45	1/45	77	1/77		76	1/76	50		191		39		157			
adc	r,ri	10	1/3	10	1/3	2	1	2	1	2	1	2	1	2^7	1	2^7	1	2^7	1	2^7	1	1	3	1	3	1	3	1	3		1	2	2	1/2	2	1/2	1	1	1	1
sbb	r,ri	10	1/3	10	1/3	2	1	2	1	2	1	2	1	2^7	1	2^7	1	2^7	1	2^7	1	1	3	1	3	1	3	1	3		1	2	2	1/2	2	1/2	1	1	1	1
shl	r,i	1	1.7	1	1.7	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	3	1	3	1	3	1	3		1	2	1	1	1	1	1	1	1	1
sal	r,i	1	1.7	1	1.7	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	3	1	3	1	3	1	3		1	2	1	1	1	1	1	1	1	1
shr	r,i	1	1.7	7	1	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	3	1	3	1	3	1	3		1	2	1	1	1	1	1	1	1	1
sar	r,i	1	1.7	7	1	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	3	1	3	1	3	1	3		1	2	1	1	1	1	1	1	1	1
shl	r,c	2	1/2	2	1/2	1	2	1	2	1	2	1	2	2	1/2	2	1/2	2	1/2	2	1/2	1	3	1	3	1	3	1	3		1	2	1	1	1	1	1	1	1	1
sal	r,c	2	1/2	2	1/2	1	2	1	2	1	2	1	2	2	1/2	2	1/2	2	1/2	2	1/2	1	3	1	3	1	3	1	3		1	2	1	1	1	1	1	1	1	1
shr	r,c	2	1/2	8	1/8	1	2	1	2	1	2	1	2	2	1/2	2	1/2	2	1/2	2	1/2	1	3	1	3	1	3	1	3		1	2	1	1	1	1	1	1	1	1
sar	r,c	2	1/2	8	1/8	1	2	1	2	1	2	1	2	2	1/2	2	1/2	2	1/2	2	1/2	1	3	1	3	1	3	1	3		1	2	1	1	1	1	1	1	1	1
shld	r,r,i	8	1/7	10	1/10	2	1	2	1	4	1	4	1	1	2	1	2	1	1	1	1	3	1/2	3	1/2	3	1/2	3	1/2		4	1/3	5	1/5	11	1/11	7	1/7	35	1/35
shrd	r,r,i	8	1/7	10	1/10	2	1	2	1	4	1	4	1	1	2	1	2	1	1	1	1	3	1/2	3	1/2	3	1/2	3	1/2		4	1/3	5	1/5	9	1/9	7	1/7	45	1/45
shld	r,r,c	9	1/8	14	1/14	2	1	2	1	4	1	4	1	2	2/3	2	2/3	4	1/2	4	1/2	3	1/3	3	1/3	3	1/3	3	1/3		4	1/4	5	1/5	10	1/10	7	1/7	35	1/35
shrd	r,r,c	9	1/8	12	1/12	2	1	2	1	4	1	4	1	2	2/3	2	2/3	4	1/2	4	1/2	3	1/3	3	1/3	3	1/3	3	1/3		4	1/4	5	1/5	9	1/9	7	1/7	45	1/45
shlx	r,r,c	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	2	1	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
shrx	r,r,c	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	2	1	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
rol	r,i	1	1	7	1/7	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1	2	1	3	1	3	1	3	1	3		1	2	1	1	1	1	1	1	1	1
ror	r,i	1	1	7	1/7	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1	2	1	3	1	3	1	3	1	3		1	2	1	1	1	1	1	1	1	1
rol	r,c	2	1/2	8	1/8	1	1	1	1	1	1	1	1	2	1/2	2	1/2	2	1/2	2	1/2	1	3	1	3	1	3	1	3		1	2	1	1	1	1	1	1	1	1
ror	r,c	2	1/2	8	1/8	1	1	1	1	1	1	1	1	2	1/2	2	1/2	2	1/2	2	1/2	1	3	1	3	1	3	1	3		1	2	1	1	1	1	1	1	1	1
cmp	r,ri	1	2.5	1	2.5										3		3		4		4	1	3	1	3	1	3	1	3		1	2	1	2	1	2	1/2	2	1	2
test	r,i	1	1.7	1	1.7										3		3		4		4	1	3	1	2.7	1	3	1	2.7		1	2	1	2	1/2	2	1	2	1/2	2	1	2
test	r,r	1	2	1	2										3		3		4		4	1	3	1	3	1	3	1	3		1	2	1	2	1	2	1/2	2	1	2
bt	r,i	8	1/8	8	1/8	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	1	3	1	3	1	3	1	3		1	2	2	1	2	1/2	2	1/2		
mov	r,r	1	2.5	1	2.5	1	3	1	3	1	3	1	3	1	3	1	3	1	4	1	4	1	3	1	3	1	3	1	3		1	2	1	2	1	1	1	1		
cmov	r,r	10	1	10	1	2	1/2	2	1/2	2	1	2	1	2	1	2	1.5	2	1	2	2	1	3	1	3	1	3	1	3		1	2	2	1/2	2	1/2	2	1	2	1
movzx	r,r	1	2.5	1	2.5	1	3	1	3	1	3	1	3	1	3	1	3	1	4	1	4	1	3	1	3	1	3	1	3		1	2	1	1	1	1	1	1	1	1
movsx	r,r	2	1.5	1	2	1	3	1	3	1	3	1	3	1	3	1	3	1	4	1	4	1	3	1	3	1	3	1	3		1	2	1	1	1	1	1	1	1	1
movsxd	r,r	-	-	1	2	-	-	1	3	-	-	1	3	-	-	1	3	-	-	1	4	-	-	1	3	-	-	1	3	-	-	1	2	-	-	1	1	-	-	1	1
bswap	r	1	2	8	0.6	4	1	4	1	1	1	3	1	1	1	2	1	1	2	2	1	1	3	1	3	1	3	1	3		1	2	1	1	1	1	1	1	1	1
lea	r,r+r			1	2.5	1	1	1	1	1	1	1	1			1	2			1	2	1	3	1	3	1	3	1	3		1	2	1	1	1	1	1	1	1	1
lea	r,r+r*s			1	1.25	1	1	1	1	1	1	1	1			1	2			1	2	2	3	2	3	2	3	2	3		2	2	1	1	1	1	1	1	1	1
lea	r,b+r+r			2	1.17	1	1	1	1	1	1	1	1			3	1			3	1	2	3	2	3	2	3	2	3		2	2	1	1	1	1	1	1	1	1
lea	r,b+r+r*s			2	0.8	1	1	1	1	1	1	1	1			3	1			3	1	2	3	2	3	2	3	2	3		2	2	1	1	1	1	1	1	1	1
bsr	r,r	16	1/2	16	1/2	2	1	2	1	3	1	3	1	3	1	3	1	3	1	3	1	11	1/10	11	1/10	4	1/3	4	1/3		4	1/4	17	1/16	17	1/16	3	1/2	3	1/2
bsf	r,r	16	1/2	16	1/2	2	1	2	1	3	1	3	1	3	1	3	1	3	1	3	1	8	1/8	9	1/9	4	1/3	4	1/3		3	1/3	16	1/16	16	1/16	3	1/2	3	1/2
lzcnt	r,r	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	3	1	3	1	-	-	-	-	2	1	2	1		3	1/2	-	-	-	-	-	-	-	-
popcnt	r,r	-	-	-	-	-	-	-	-	3	1	3	1	3	1	3	1	3	1	3	1	-	-	-	-	2	1	2	1		4	1/4	-	-	-	-	-	-	-	-

Table 1. Integer register instructions, 32-bit operations and 64-bit operations.

		Intel P4						Intel P6						AMD		AMD		AMD		Intel	
		F0-F1		F2		F3-F4		0-E		Core 2		NHM		K7		K8-K9		K10		Atom	
		L32	T32	L32	T32	L32	T32	L32	T32	L32	T32	L32	T32	L32	T32	L32	T32	L32	T32	L32	T32
add	r,ri	1/2	3	1/2	3	1	2.5	1	2	1	3	1	3	1	2.7	1	3	1	3	1	2
sub	r,ri	1/2	3	1/2	3	1	2.5	1	2	1	3	1	3	1	2.7	1	3	1	3	1	2
and	r,r	1/2	2	1/2	2	1	1.75	1	2	1	3	1	3	1	2.7	1	3	1	3	1	2
or	r,r	1/2	2	1/2	2	1	1.75	1	2	1	3	1	3	1	2.7	1	3	1	3	1	2
xor	r,r	1/2	2	1/2	2	1	1.75	1	2	1	3	1	3	1	2.7	1	3	1	3	1	2
inc	r	1/2	1.5	1/2	1.5	1	1	1	2	1	3	1	3	1	3	1	3	1	3	1	2
dec	r	1/2	1.5	1/2	1.5	1	1	1	2	1	3	1	3	1	3	1	3	1	3	1	2
neg	r	1/2	2	1/2	2	1	2	1	2	1	3	1	3	1	2.7	1	3	1	3	1	2
not	r	1/2	2	1/2	2	1	1.7	1	2	1	3	1	3	1	2.7	1	3	1	3	1	2
imul	r,ri	14	1/4	14	1/4	10	1	4	1	3	1	3	1	4	1/2	3	1	3	1	5	1/2
mul	r	14	1/10	14	1/10	11	1/2	5	1/2	5	0.67		1/2	6[3]	1/3	3	1	3	1	9	1/9
div	r	70[2]		70[2]		80[2]	1/34			40		26		39	1/39	39	1/39	45	1/45	50	
adc	r,ri	7-8	1/6	7-8	1/6	10	1/4	2	0.75	2	1	2	1	1	2	1	3	1	3	2	1/2
sbb	r,ri	7-8	1/6	7-8	1/6	10	1/4	2	0.75	2	1	2	1	1	2	1	3	1	3	2	1/2
shl	r,i	4	1	4	1	1	1.7	1	1	1	2	1	2	1	2.7	1	3	1	3	1	1
sal	r,i	4	1	4	1	1	1.7	1	1	1	2	1	2	1	2.7	1	3	1	3	1	1
shr	r,i	4	1	4	1	1	1.7	1	1	1	2	1	2	1	2.7	1	3	1	3	1	1
sar	r,i	4	1	4	1	1	1.7	1	1	1	2	1	2	1	2.7	1	3	1	3	1	1
shl	r,c	6	1/6	6	1/6	2	1/2	1	1	1	2	1	2	1	2.7	1	3	1	3	1	1
sal	r,c	6	1/6	6	1/6	2	1/2	1	1	1	2	1	2	1	2.7	1	3	1	3	1	1
shr	r,c	6	1/6	6	1/6	2	1/2	1	1	1	2	1	2	1	2.7	1	3	1	3	1	1
sar	r,c	6	1/6	6	1/6	2	1/2	1	1	1	2	1	2	1	2.7	1	3	1	3	1	1
shld	r,r,i	12	1/12	8	1/4	8	1/7	2	1/2	2	1	4	1	2	1/2	3	1/2	3	1/2	5	1/5
shrd	r,r,i	14	1/14	8	1/4	8	1/7	2	1/2	2	1	4	1	2	1/2	3	1/2	3	1/2	5	1/5
shld	r,r,c	12	1/12	7	1/4	9	1/8	2	1/2	2	1	4	1	3	1/3	3	1/3	3	1/3	5	1/5
shrd	r,r,c	14	1/14	7	1/4	9	1/8	2	1/2	2	1	4	1	3	1/3	3	1/3	3	1/3	5	1/5
rol	r,i	4	1/4	4	1/4	1	1	1	1	1	1	1	1	1	2.7	1	3	1	3	1	1
ror	r,i	4	1/4	4	1/4	1	1	1	1	1	1	1	1	1	2.7	1	3	1	3	1	1
rol	r,c	6	1/6	6	1/6	2	1/2	1	1	1	1	1	1	1	2.7	1	3	1	3	1	1
ror	r,c	6	1/6	6	1/6	2	1/2	1	1	1	1	1	1	1	2.7	1	3	1	3	1	1
cmp	r,ri	1/2	3	1/2	3	1	2.5	1	2					1	2.7	1	3	1	3	1	2
test	r,i	1/2	2	1/2	2	1	1.7	1	2					1	2.7	1	3	1	3	1	2
test	r,r	1/2	2	1/2	2	1	1.7	1	2					1	2.7	1	3	1	3	1	2
bt	r,i	6	1/6	6	1/6	8	1/8	1	1	2	1	2	1	1	3	1	3	1	3	1	1
mov	r,r	1/2	3	1/2	3	1	2.5	1	2	1	3	1	3	1	2.7	1	3	1	3	1	2
cmov	r,r	6	1	6	1	10	1	2	1/2	2	1/2	2	1	1	2.3	1	3	1	3	2	1/2
movzx	r,r	1/2	3	1/2	3	1	2.5	1	2	1	3	1	3	1	2.3	1	3	1	3	1	1
movsx	r,r	1/2	2	1/2	2	1	1.75	1	2	1	3	1	3	1	2.3	1	3	1	3	1	1
bswap	r	7	0.67	7	0.67	1	2	2	1	4	1	1	1	1	2.7	1	3	1	3	1	1
lea	r,r+r	1/2	3	1/2	3	1	2.5	1	1	1	1	1	1	2	2.3	1	3	1	3	1	1
lea	r,r+r*s	3	1	3	1	1	1.25	1	1	1	1	1	1	2	2.3	2	3	2	3	1	1
lea	r,b+r+r	1	1.5	1	1.5	2	1.17	1	1	1	1	1	1	2	3	2	3	2	3	1	1
lea	r,b+r+r*s	4	1	4	1	2	0.8	1	1	1	1	1	1	2	3	2	3	2	3	1	1
bsr	r,r	8	1/2	8	1/2	16	1/2	2	1	2	1	3	1	9	1/8	11	1/10	4	1/3	17	1/16
bsf	r,r	8	1/2	8	1/2	16	1/2	2	1	2	1	3	1	7	1/7	8	1/8	4	1/3	16	1/16

Table 2. Integer register instructions, 32-bit operations. Please note that this table lacks more recent CPUs as well as some instructions which are present in the previous table.

	Intel P4				Intel P6				AMD		AMD		AMD		Intel	
	F0-F2		F3-F4		D-E		F,17		K7		K8-K9		K10		Atom	
	L	T	L	T	L	T	L	T	L	T	L	T	L	T	L	T
paddq xmm,xmm	4	1/2	5	1/2	2	1/2	2	1	-	-	2	1	2	2	5	1/5
psubq xmm,xmm	4	1/2	5	1/2	2	1/2	2	1	-	-	2	1	2	2	5	1/5
paddd xmm,xmm	2	1/2	2	1/2	1	1	1	2	-	-	2	1	2	2	1	2
psubd xmm,xmm	2	1/2	2	1/2	1	1	1	2	-	-	2	1	2	2	1	2
pmuludq xmm,xmm	6	1/2	7	1/2	4	1/2	3	1	-	-	3	1/2	3	1	5	1/2
psllq xmm,xmm	2	1/2	2	1/2	2	1/2	1	1	-	-	2	1	3	2	5	1/5
psrlq xmm,xmm	2	1/2	2	1/2	2	1/2	1	1	-	-	2	1	3	2	5	1/5
psllq xmm,i	2	1/2	2	1/2	2	1/2	1	1	-	-	2	1	2	2	1	1
psrlq xmm,i	2	1/2	2	1/2	2	1/2	1	1	-	-	2	1	2	2	1	1
pslldq xmm,i	4	1/2	4	1/2	3	1/3	2	1	-	-	2	1	3	2	1	1
psrldq xmm,i	4	1/2	4	1/2	3	1/3	2	1	-	-	2	1	3	2	1	1
pand xmm,xmm	2	1/2	2	1/2	1	1	1	3	-	-	2	1	2	2	1	2
pandn xmm,xmm	2	1/2	2	1/2	1	1	1	3	-	-	2	1	2	2	1	2
por xmm,xmm	2	1/2	2	1/2	1	1	1	3	-	-	2	1	2	2	1	2
pxor xmm,xmm	2	1/2	2	1/2	1	1	1	3	-	-	2	1	2	2	1	2
movq xmm,xmm	2	1/2	2	1/2	1	1	1	3	-	-	2	1	2.5	3	1	2
punpckldq xmm,xmm	2	1/2	2	1/2	2	1/2	2	1/2	-	-	2	1	3	2	1	1
psadbw xmm,xmm	4	1/2	4	1/2	4	1/2	3	1	-	-	3	1/2	3	1	5	1/2
pshufd xmm,xmm,i	4	1/2	4	1/2	2	1/2	4	1	-	-	3	0.67	3	2	1	1

Table 3. Subset of integer SSE instructions.

	Intel P4				Intel P6				AMD		AMD		AMD		Intel	
	F0-F2		F3-F4		B|D		F		K7		K8-K9		K10		Atom	
	L	T	L	T	L	T	L	T	L	T	L	T	L	T	L	T
paddq mm,mm	2	1	2	1	-|2	-|1	2	1	-	-	2	2	2	2	5	1/5
psubq mm,mm	2	1	2	1	-|2	-|1	2	1	-	-	2	2	2	2	5	1/5
paddd mm,mm	2	1	2	1	1	1	1	2	2	2	2	2	2	2	1	2
psubd mm,mm	2	1	2	1	1	1	1	2	2	2	2	2	2	2	1	2
pmuludq mm,mm	6	1	7	1	-|4	-|1	3	1	-	-	3	1	3	1	4	1
pmul*w mm,mm	6	1	7	1	-|3	-|1	3	1	-	-	3	1	3	1	4	1
pmaddwd mm,mm	6	1	7	1	3	1	3	1	-	-	3	1	3	1	4	1
psllq mm,mm	2	1	2	1	1	1	1	1	2	2	2	2	2	2	5	1/5
psrlq mm,mm	2	1	2	1	1	1	1	1	2	2	2	2	2	2	5	1/5
psllq mm,i	2	1	2	1	1	1	1	1	2	2	2	2	2	2	1	1
psrlq mm,i	2	1	2	1	1	1	1	1	2	2	2	2	2	2	1	1
pand mm,mm	2	1	2	1	1	1	1	3	2	2	2	2	2	2	1	2
pandn mm,mm	2	1	2	1	1	1	1	3	2	2	2	2	2	2	1	2
por mm,mm	2	1	2	1	1	1	1	3	2	2	2	2	2	2	1	2
pxor mm,mm	2	1	2	1	1	1	1	3	2	2	2	2	2	2	1	2
movq mm,mm	6	1	7	1	1	2	1	3	2	2	2	2	2	2	1	2
punpckldq mm,mm	2	1	2	1	1	1	1	1	2	2	2	2	2	2	1	1
psadbw mm,mm	4	1	4	1	5|4	1/2|1	3	1	3	1	3	1	3	1	4	1
pshufw mm,mm,i	2	1	2	1	1	1	1	1	2	2	2	2	2	2	1	1

Table 4. Subset of integer MMX instructions.

Table remarks:

1. The latency is the indicated 5 cycles for the upper product half. The latency for the lower part is 4 cycles.
2. The latency is data dependent, the given numbers represent the worst case.
3. The latency is the indicated 6 cycles for the upper product half. The latency for the lower part is 4 cycles.
4. The latency is 77 cycles for model 0x17.
5. The latency is the indicated 10 cycles for the upper product half. The latency for the lower part is 3 cycles.
6. The latency is the indicated 4 cycles for the upper product half. The latency for the lower part is 3 cycles.
7. The latency is the indicated 2 cycles for the register result; the latency for the carry bit is just 1 cycle. The immediate operand 0 is handled specially; it gives a latency of just 1 also to the result register.

Information on some processor cores

Here we list some x86 processors, with core names, manufacturing technology, and cache sizes. **This table is obsolete.**

AMD	**Core name**	**CPUID model**	**Tech**	**L1d**	**L1i**	**L2**
Duron	Spitfire	3	180	64	64	64
Athlon	Thunderbird	4	180	64	64	256
?	?	5	180	64	64	256
Athlon	Palomino	6	180	64	64	256
Duron	Morgan	7	180	64	64	64
Athlon XP	Thoroughbred	8	130	64	64	256
Athlon XP	Thoroughbred-B	8	130	64	64	256
?	?	9	130	64	64	256
Athlon XP	Barton	10	130	64	64	512
Athlon 64	Clawhammer		130	64	64	512
Athlon 64	Sledgehammer		130	64	64	1024
Athlon 64	Newcastle		90	64	64	512
Athlon 64	Winchester		90	64	64	512
Athlon 64	Venice		90	64	64	512
Athlon 64	San Diego		90	64	64	1024
Athlon 64 X2	Manchester		90	64	64	512
Athlon 64 X2	Toledo		90	64	64	1024
Opteron	Sledgehammer		130	64	64	1024
Opteron	Denmark		90	64	64	1024
Opteron	Paris		90	64	64	1024

Intel	**Core name**	**CPUID model**	**Tech**	**L1d**	**L1i**	**L2**
Pentium 3	Katmai	67	250	16	16	512
Pentium 3	Coppermine	68	180	16	16	256
Pentium M	Banias	69	130	16	16	1024
Pentium 3	Tualatin	6B	130	16	16	512
Pentium M	Dothan	6D	90	32	32	2048
Core	Yonah	6E	65	32	32	2048
Core 2	Conroe,etc	6F	65	32	32	2048-4096
Pentium 4	Willamette	F0,F1	180	8		256
Pentium 4	Northwood	F2	130	8		512
Pentium 4	Prescott	F3,F4	90	16		1024-2048
Pentium 4	Cedar Mill	F6	65	16		2048

Change log

2005-09-06	Added numbers for Pentium M
2006-07-20	Added initial numbers for Core 2; added more core names
2007-09-02	Corrected Pentium 4 throughput for `adc` and `sbb`
2007-11-29	Corrected K7 latency for `mul`
2008-11-16	Added data for Intel Atom, AMD K10
2008-11-17	Corrected K8 throughput for `adc` and `sbb`, added some missing data points
2009-07-25	Add Core i7 numbers, corrected several Core 2 numbers
2009-07-27	Split K8-K9 and K10 into separate tables, there are small timing differences
2009-07-28	Add data for `lzcnt` and `popcnt`; cleanup formatting
2009-11-18	Add preliminary data for VIA Nano
2010-12-16	Corrections to VIA Nano data
2011-01-29	Add preliminary data for Intel Sandy Bridge
2011-02-01	Clarify Sandy Bridge adc/sbb latency
2011-03-12	Correct Sandy Bridge shld/shrd throughput

Other resources

IA-32 Intel Architecture Optimisation Reference Manual,
`http://developer.intel.com/design/pentium4/manuals/248966.htm`

Software Optimisation Guide for AMD64 Processors,
`http://amd.com/us-en/assets/content_type/white_papers_and_tech_docs/25112.PDF`

Agner Fog's x86 optimisation manuals, `http://www.agner.org/optimize/`