

Instruction latencies and throughput for AMD and Intel x86 processors

Torbjörn Granlund, Swox AB

November 29, 2007

Copyright Swox AB, 2005, 2006, 2007.

Verbatim copying and distribution of this entire article is permitted in any medium, provided this notice is preserved.

[This report is work-in-progress. A newer version might be available for download here: <http://swox.com/doc/x86-timing.pdf>]

In this short report we present latencies 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 is incomplete and full of errors.
2. Intel does 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 performed at Swox. While we have made an effort to make sure the data is accurate, the reader is cautioned that some errors might have crept in. The data is surely less riddled by errors than Intel's manuals, though.

Nomenclature and notation

LNN means *latency for NN-bit operation*. *TNN* 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 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.

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 optimized for different ranges. Our numbers are for the fastest operands for each core, not necessarily the smallest.

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.)

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/sar 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 data cache can provide only 16 bytes per clock, but 3 `test r, i` need 18 or 21 bytes (depending on register).

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 "Core" 2, they result in a pipeline hiccup of about 12 cycles. That sort of problems cannot be represented in the table numbers.

Your feedback

Please send feedback about this report to tg at swox (dot) com.

		Intel P4 F4				Intel P6 F				AMD K8			
		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
sub	r,ri	1	2.5	1	2.5	1	3	1	3	1	3	1	3
and	r,r	1	2	1	2	1	3	1	3	1	3	1	3
or	r,r	1	2	1	2	1	3	1	3	1	3	1	3
xor	r,r	1	2	1	2	1	3	1	3	1	3	1	3
inc	r	1	1	1	1	1	3	1	3	1	3	1	3
dec	r	1	1	1	1	1	3	1	3	1	3	1	3
neg	r	1	2	1	2	1	3	1	3	1	3	1	3
not	r	1	1.7	1	1.7	1	3	1	3	1	3	1	3
imul	r,ri	10	1	10	0.5	3	1	5	0.5	3	1	4	0.5
mul	r	11	0.5	12	1/10	5	0.67	8	1/4	3	1	5 ¹	0.5
div	r	80 ²	1/34	161 ²	1/151					39	1/39	71	1/71
adc	r,ri	10	1/3	10	1/3	2	1	2	1	1	2.2	1	2.3
sbb	r,ri	10	1/3	10	1/3	2	1	2	1	1	2.2	1	2.3
shl	r,i	1	1.7	1	1.7	1	2	1	2	1	3	1	3
sal	r,i	1	1.7	1	1.7	1	2	1	2	1	3	1	3
shr	r,i	1	1.7	7	1	1	2	1	2	1	3	1	3
sar	r,i	1	1.7	7	1	1	2	1	2	1	3	1	3
shl	r,c	2	0.5	2	0.5	1	2	1	2	1	3	1	3
sal	r,c	2	0.5	2	0.5	1	2	1	2	1	3	1	3
shr	r,c	2	0.5	8	1/8	1	2	1	2	1	3	1	3
sar	r,c	2	0.5	8	1/8	1	2	1	2	1	3	1	3
shld	r,r,i	8	1/7	10	1/10	2	1	2	1	3	0.5	3	0.5
shrd	r,r,i	8	1/7	10	1/10	2	1	2	1	3	0.5	3	0.5
shld	r,r,c	9	1/8	14	1/14	2	1	2	1	3	1/3	3	1/3
shrd	r,r,c	9	1/8	12	1/12	2	1	2	1	3	1/3	3	1/3
rol	r,i	1	1	7	1/7	1	1	1	1	1	3	1	3
ror	r,i	1	1	7	1/7	1	1	1	1	1	3	1	3
rol	r,c	2	0.5	8	1/8	1	1	1	1	1	3	1	3
ror	r,c	2	0.5	8	1/8	1	1	1	1	1	3	1	3
cmp	r,ri	1	2.5	1	2.5	1	3	1	3	1	3	1	3
test	r,i	1	1.7	1	1.7	1	3	1	2.7	1	3	1	2.7
test	r,r	1	2	1	2	1	3	1	3	1	3	1	3
bt	r,i	8	1/8	8	1/8	1	1	1	1	1	3	1	3
mov	r,r	1	2.5	1	2.5	1	3	1	3	1	3	1	3
cmov	r,r	10	1	10	1	1	1	1	1	1	3	1	3
movzx	r,r	1	2.5	1	2.5	1	3	1	3	1	3	1	3
movsx	r,r	2	1.5	1	2	1	3	1	3	1	3	1	3
movsxd	r,r	-	-	1	2	1				-	-	1	3
bswap	r	1	2	8	0.6	4	1	4	1	1	3	1	3
lea	r,r+r			1	2.5	1	1	1	1	2	3	2	3
lea	r,r+r*s			1	1.25	1	1	1	1	2	3	2	3
lea	r,b+r+r			2	1.17	1	1	1	1	2	3	2	3
lea	r,b+r+r*s			2	0.8	1	1	1	1	2	3	2	3
bsr	r,r	16	0.5	16	0.5	1	1	1	1	11	1/10	11	1/10
bsf	r,r	16	0.5	16	0.5	1	1	1	1	8	1/8	9	1/9

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

	Intel P4						Intel P6				AMD		AMD	
	F0-F1		F2		F3-F4		0-E		F		K7		K8	
	L32	T32	L32	T32	L32	T32	L32	T32	L32	T32	L32	T32	L32	T32
add r,ri	0.5	3	0.5	3	1	2.5	1	2	1	3	1	2.7	1	3
sub r,ri	0.5	3	0.5	3	1	2.5	1	2	1	3	1	2.7	1	3
and r,r	0.5	2	0.5	2	1	1.75	1	2	1	3	1	2.7	1	3
or r,r	0.5	2	0.5	2	1	1.75	1	2	1	3	1	2.7	1	3
xor r,r	0.5	2	0.5	2	1	1.75	1	2	1	3	1	2.7	1	3
inc r	0.5	1.5	0.5	1.5	1	1	1	2	1	3	1	3	1	3
dec r	0.5	1.5	0.5	1.5	1	1	1	2	1	3	1	3	1	3
neg r	0.5	2	0.5	2	1	2	1	2	1	3	1	2.7	1	3
not r	0.5	2	0.5	2	1	1.7	1	2	1	3	1	2.7	1	3
imul r,ri	14	1/4	14	1/4	10	1	4	1	3	1	4	0.5	3	1
mul r	14	1/10	14	1/10	11	0.5	5	0.5	5	0.67	6 ³	1/3	3	1
div r	70 ²		70 ²		80 ²	1/34					39	1/39	39	1/39
adc r,ri	7-8	1/6	7-8	1/6	10	1/4	2	0.75	2		1	2	1	2.2
sbb r,ri	7-8	1/6	7-8	1/6	10	1/4	2	0.75	2		1	2	1	2.2
shl r,i	4	1	4	1	1	1.7	1	1	1	2	1	2.7	1	3
sal r,i	4	1	4	1	1	1.7	1	1	1	2	1	2.7	1	3
shr r,i	4	1	4	1	1	1.7	1	1	1	2	1	2.7	1	3
sar r,i	4	1	4	1	1	1.7	1	1	1	2	1	2.7	1	3
shl r,c	6	1/6	6	1/6	2	0.5	1	1	1	2	1	2.7	1	3
sal r,c	6	1/6	6	1/6	2	0.5	1	1	1	2	1	2.7	1	3
shr r,c	6	1/6	6	1/6	2	0.5	1	1	1	2	1	2.7	1	3
sar r,c	6	1/6	6	1/6	2	0.5	1	1	1	2	1	2.7	1	3
shld r,r,i	12	1/12	8	1/4	8	1/7	2	0.5	2	1	2	0.5	3	0.5
shrd r,r,i	14	1/14	8	1/4	8	1/7	2	0.5	2	1	2	0.5	3	0.5
shld r,r,c	12	1/12	7	1/4	9	1/8	2	0.5	2	1	3	1/3	3	1/3
shrd r,r,c	14	1/14	7	1/4	9	1/8	2	0.5	2	1	3	1/3	3	1/3
rol r,i	4	1/4	4	1/4	1	1	1	1	1	1	1	2.7	1	3
ror r,i	4	1/4	4	1/4	1	1	1	1	1	1	1	2.7	1	3
rol r,c	6	1/6	6	1/6	2	0.5	1	1	1	1	1	2.7	1	3
ror r,c	6	1/6	6	1/6	2	0.5	1	1	1	1	1	2.7	1	3
cmp r,ri	0.5	3	0.5	3	1	2.5	1	2	1	3	1	2.7	1	3
test r,i	0.5	2	0.5	2	1	1.7	1	2	1	3	1	2.7	1	3
test r,r	0.5	2	0.5	2	1	1.7	1	2	1	3	1	2.7	1	3
bt r,i	6	1/6	6	1/6	8	1/8	1	1	1	1	1	3	1	3
mov r,r	0.5	3	0.5	3	1	2.5	1	2	1	3	1	2.7	1	3
cmov r,r	6	1	6	1	10	1	2	0.5	1	1	1	2.3	1	3
movzx r,r	0.5	3	0.5	3	1	2.5	1	2	1	3	1	2.3	1	3
movsx r,r	0.5	2	0.5	2	1	1.75	1	2	1	3	1	2.3	1	3
bswap r	7	0.67	7	0.67	1	2	2	1	4	1	1	2.7	1	3
lea r,r+r	0.5	3	0.5	3	1	2.5	1	1	1	1	2	2.3	2	3
lea r,r+r*s	3	1	3	1	1	1.25	1	1	1	1	2	2.3	2	3
lea r,b+r+r	1	1.5	1	1.5	2	1.17	1	1	1	1	2	3	2	3
lea r,b+r+r*s	4	1	4	1	2	0.8	1	1	1	1	2	3	2	3
bsr r,r	8	0.5	8	0.5	16	0.5	2	1	1	1	9	1/8	11	1/10
bsf r,r	8	0.5	8	0.5	16	0.5	2	1	1	1	7	1/7	8	1/8

Table 2. Integer register instructions, 32-bit operations.

		Intel P4				Intel P6				AMD		AMD	
		4 F0-F2		4 F3-F4		D-E		F		K7		K8	
		L	T	L	T	L	T	L	T	L	T	L	T
paddq	xmm, xmm	4	0.5	5	0.5	2	0.5	2	1	-	-	2	1
psubq	xmm, xmm	4	0.5	5	0.5	2	0.5	2	1	-	-	2	1
padd	xmm, xmm	2	0.5	2	0.5	1	1	1	2	-	-	2	1
psubd	xmm, xmm	2	0.5	2	0.5	1	1	1	2	-	-	2	1
pmuludq	xmm, xmm	6	0.5	7	0.5	4	0.5	3	1	-	-	3	0.5
psllq	xmm, xmm	2	0.5	2	0.5	2	0.5	1	1	-	-	2	1
psrlq	xmm, xmm	2	0.5	2	0.5	2	0.5	1	1	-	-	2	1
pslldq	xmm, i	4	0.5	4	0.5	3	1/3	2	1	-	-	2	1
psrldq	xmm, i	4	0.5	4	0.5	3	1/3	2	1	-	-	2	1
pand	xmm, xmm	2	0.5	2	0.5	1	1	1	3	-	-	2	1
pandn	xmm, xmm	2	0.5	2	0.5	1	1	1	3	-	-	2	1
por	xmm, xmm	2	0.5	2	0.5	1	1	1	3	-	-	2	1
pxor	xmm, xmm	2	0.5	2	0.5	1	1	1	3	-	-	2	1
movq	xmm, xmm	2	0.5	2	0.5	1	1	1	3	-	-	2	1
punpckldq	xmm, xmm	2	0.5	2	0.5	2	0.5	2	0.5	-	-	2	1
psadbw	xmm, xmm	4	0.5	4	0.5	4	0.5	3	1	-	-	3	0.5
pshufd	xmm, xmm, i	4	0.5	4	0.5	2	0.5	4	1	-	-	3	0.67

Table 3. Subset of integer SSE instructions.

		Intel P4				Intel P6				AMD		AMD	
		4 F0-F2		4 F3-F4		B#D		F		K7		K8	
		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
psubq	mm, mm	2	1	2	1	-#2	-#1	2	1	-	-	2	2
padd	mm, mm	2	1	2	1	1	1	1	2	2	2	2	2
psubd	mm, mm	2	1	2	1	1	1	1	2	2	2	2	2
pmuludq	mm, mm	6	1	7	1	-#4	-#1	3	1	-	-	3	1
pmul*w	mm, mm	6	1	7	1	-#3	-#1	3	1	-	-	3	1
pmaddwd	mm, mm	6	1	7	1	3	1	3	1	-	-	3	1
psllq	mm, mm	2	1	2	1	1	1	1	1	2	2	2	2
psrlq	mm, mm	2	1	2	1	1	1	1	1	2	2	2	2
pand	mm, mm	2	1	2	1	1	1	1	3	2	2	2	2
pandn	mm, mm	2	1	2	1	1	1	1	3	2	2	2	2
por	mm, mm	2	1	2	1	1	1	1	3	2	2	2	2
pxor	mm, mm	2	1	2	1	1	1	1	3	2	2	2	2
movq	mm, mm	6	1	7	1	1	2	1	3	2	2	2	2
punpckldq	mm, mm	2	1	2	1	1	1	1	1	2	2	2	2
psadbw	mm, mm	4	1	4	1	5#4	0.5#1	3	1	3	1	3	1
pshufw	mm, mm, i	2	1	2	1	1	1	1	1	2	2	2	2

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.

Information on some processor cores

Here we list some x86 processors, with core names, manufacturing technology, and cache sizes.

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`

References

IA-32 Intel Architecture Optimization Reference Manual,

<http://developer.intel.com/design/pentium4/manuals/248966.htm>

Software Optimization Guide for AMD64 Processors,

http://amd.com/us-en/assets/content_type/white_papers_and_tech_docs/25112.PDF