

# 4

## Instruction tables

### Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD and VIA CPUs

By Agner Fog. Copenhagen University College of Engineering.

Copyright © 1996 - 2012. Last updated 2012-02-29.

#### Introduction

This is the fourth in a series of five manuals:

1. Optimizing software in C++: An optimization guide for Windows, Linux and Mac platforms.
2. Optimizing subroutines in assembly language: An optimization guide for x86 platforms.
3. The microarchitecture of Intel, AMD and VIA CPUs: An optimization guide for assembly programmers and compiler makers.
4. Instruction tables: Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD and VIA CPUs.
5. Calling conventions for different C++ compilers and operating systems.

The latest versions of these manuals are always available from [www.agner.org/optimize](http://www.agner.org/optimize). Copyright conditions are listed below.

The present manual contains tables of instruction latencies, throughputs and micro-operation breakdown and other tables for x86 family microprocessors from Intel, AMD and VIA.

The figures in the instruction tables represent the results of my measurements rather than the official values published by microprocessor vendors. Some values in my tables are higher or lower than the values published elsewhere. The discrepancies can be explained by the following factors:

- My figures are experimental values while figures published by microprocessor vendors may be based on theory or simulations.
- My figures are obtained with a particular test method under particular conditions. It is possible that different values can be obtained under other conditions.
- Some latencies are difficult or impossible to measure accurately, especially for memory access and type conversions that cannot be chained.
- Latencies for moving data from one execution unit to another are listed explicitly in some of my tables while they are included in the general latencies in some tables published by Intel.

Most values are the same in all microprocessor modes (real, virtual, protected, 16-bit, 32-bit, 64-bit). Values for far calls and interrupts may be different in different modes. Call gates have not been tested.

Instructions with a LOCK prefix have a long latency that depends on cache organization and possibly RAM speed. If there are multiple processors or cores or direct memory access (DMA) devices then all locked instructions will lock a cache line for exclusive access, which may involve RAM access. A LOCK prefix typically costs more than a hundred clock cycles, even on single-processor systems. This also applies to the XCHG instruction with a memory operand.

## Introduction

If any text in the pdf version of this manual is unreadable, then please refer to the spreadsheet version.

### **Copyright notice**

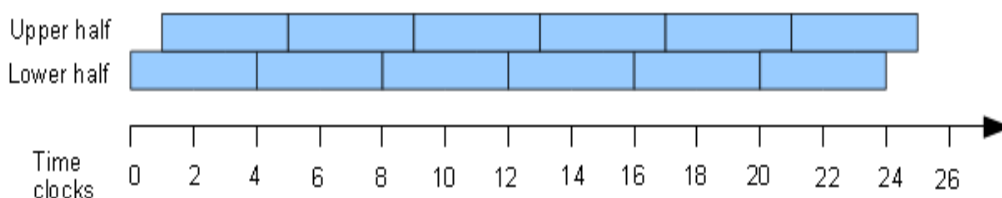
This series of five manuals is copyrighted by Agner Fog. Public distribution and mirroring is not allowed. Non-public distribution to a limited audience for educational purposes is allowed. The code examples in these manuals can be used without restrictions. A GNU Free Documentation License shall automatically come into force when I die. See [www.gnu.org/copyleft/fdl.html](http://www.gnu.org/copyleft/fdl.html)

## Definition of terms

**Operands** Operands can be different types of registers, memory, or immediate constants. Abbreviations used in the tables are: i = immediate constant, r = any general purpose register, r32 = 32-bit register, etc., mm = 64 bit mmx register, x or xmm = 128 bit xmm register, y = 256 bit ymm register, sr = segment register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc.

**Latency** The latency of an instruction is the delay that the instruction generates in a dependency chain. The measurement unit is clock cycles. Where the clock frequency is varied dynamically, the figures refer to the core clock frequency. The numbers listed are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity may increase the latencies by possibly more than 100 clock cycles on many processors, except in move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results may give a similar delay. A missing value in the table means that the value has not been measured or that it cannot be measured in a meaningful way.

Some processors have a pipelined execution unit that is smaller than the largest register size so that different parts of the operand are calculated at different times. Assume, for example, that we have a long dependency chain of 128-bit vector instructions running in a fully pipelined 64-bit execution unit with a latency of 4. The lower 64 bits of each operation will be calculated at times 0, 4, 8, 12, 16, etc. And the upper 64 bits of each operation will be calculated at times 1, 5, 9, 13, 17, etc. as shown in the figure below. If we look at one 128-bit instruction in isolation, the latency will be 5. But if we look at a long chain of 128-bit instructions, the total latency will be 4 clock cycles per instruction plus one extra clock cycle in the end. The latency in this case is listed as 4 in the tables because this is the value it adds to a dependency chain.



**Reciprocal throughput** The throughput is the maximum number of instructions of the same kind that can be executed per clock cycle when the operands of each instruction are independent of the preceding instructions. The values listed are the reciprocals of the throughputs, i.e. the average number of clock cycles per instruction when the instructions are not part of a limiting dependency chain. For example, a reciprocal throughput of 2 for FMUL means that a new FMUL instruction can start executing 2 clock cycles after a previous FMUL. A reciprocal throughput of 0.33 for ADD means that the execution units can handle 3 integer additions per clock cycle.

The reason for listing the reciprocal values is that this makes comparisons between latency and throughput easier. The reciprocal throughput is also called issue latency. The values listed are for a single thread or a single core. A missing value in the table means that the value has not been measured.

## Definition of terms

<b>μops</b>	Uop or μop is an abbreviation for micro-operation. Processors with out-of-order cores are capable of splitting complex instructions into μops. For example, a read-modify-instruction may be split into a read-μop and a modify-μop. The number of μops that an instruction generates is important when certain bottlenecks in the pipeline limit the number of μops per clock cycle.
<b>Execution unit</b>	The execution core of a microprocessor has several execution units. Each execution unit can handle a particular category of μops, for example floating point additions. The information about which execution unit a particular μop goes to can be useful for two purposes. Firstly, two μops cannot execute simultaneously if they need the same execution unit. And secondly, some processors have a latency of an extra clock cycle when the result of a μop executing in one execution unit is needed as input for a μop in another execution unit.
<b>Execution port</b>	The execution units are clustered around a few execution ports on most Intel processors. Each μop passes through an execution port to get to the right execution unit. An execution port can be a bottleneck because it can handle only one μop at a time. Two μops cannot execute simultaneously if they need the same execution port, even if they are going to different execution units.
<b>Instruction set</b>	<p>This indicates which instruction set an instruction belongs to. The instruction is only available in processors that support this instruction set. The different instruction sets are listed at the end of this manual. Availability in processors prior to 80386 does not apply for 32-bit and 64-bit operands. Availability in the MMX instruction set does not apply to 128-bit packed integer instructions, which require SSE2. Availability in the SSE instruction set does not apply to double precision floating point instructions, which require SSE2.</p> <p>32-bit instructions are available in 80386 and later. 64-bit instructions in general purpose registers are available only under 64-bit operating systems. Instructions that use XMM registers (SSE and later) are only available under operating systems that support this register set. Instructions that use YMM registers (AVX and later) are only available under operating systems that support this register set.</p>

## How the values were measured

The values in the tables are measured with the use of my own test programs, which are available from [www.agner.org/optimize/testp.zip](http://www.agner.org/optimize/testp.zip)

The time unit for all measurements is CPU clock cycles. It is attempted to obtain the highest clock frequency if the clock frequency is varying with the workload. Many Intel processors have a performance counter named "core clock cycles". This counter gives measurements that are independent of the varying clock frequency. Where no "core clock cycles" counter is available, the "time stamp counter" is used (RDTSC instruction). In cases where this gives inconsistent results (e.g. in AMD Bobcat) it is necessary to make the processor boost the clock frequency by executing a large number of instructions (> 1 million) or turn off the power-saving feature in the BIOS setup.

Instruction throughputs are measured with a long sequence of instructions of the same kind, where subsequent instructions use different registers in order to avoid dependence of each instruction on the previous one. The input registers are cleared in the cases where it is impossible to use different registers. The test code is carefully constructed in each case to make sure that no other bottleneck is limiting the throughput than the one that is being measured.

Instruction latencies are measured in a long dependency chain of identical instructions where the output of each instruction is needed as input for the next instruction.

The sequence of instructions should be long, but not so long that it doesn't fit into the level-1 code cache. A typical length is 100 instructions of the same type. This sequence is repeated in a loop if a larger number of instructions is desired.

## Definition of terms

It is not possible to measure the latency of a memory read or write instruction with software methods. It is only possible to measure the combined latency of a memory write followed by a memory read from the same address. What is measured here is not actually the cache access time, because in most cases the microprocessor is smart enough to make a "store forwarding" directly from the write unit to the read unit rather than waiting for the data to go to the cache and back again. The latency of this store forwarding process is arbitrarily divided into a write latency and a read latency in the tables. But in fact, the only value that makes sense to performance optimization is the sum of the write time and the read time.

A similar problem occurs where the input and the output of an instruction use different types of registers. For example, the MOVD instruction can transfer data between general purpose registers and XMM vector registers. The value that can be measured is the combined latency of data transfer from one type of registers to another type and back again ( $A \rightarrow B \rightarrow A$ ). The division of this latency between the  $A \rightarrow B$  latency and the  $B \rightarrow A$  latency is sometimes obvious, sometimes based on guesswork,  $\mu\text{op}$  counts, indirect evidence, or triangular sequences such as  $A \rightarrow B \rightarrow \text{Memory} \rightarrow A$ . In many cases, however, the division of the total latency between  $A \rightarrow B$  latency and  $B \rightarrow A$  latency is arbitrary. However, what cannot be measured cannot matter for performance optimization. What counts is the sum of the  $A \rightarrow B$  latency and the  $B \rightarrow A$  latency, not the individual terms.

The  $\mu\text{op}$  counts are usually measured with the use of the performance monitor counters (PMCs) that are built into modern microprocessors. The PMCs for VIA processors are undocumented, and the interpretation of these PMCs is based on experimentation.

The execution ports and execution units that are used by each instruction or  $\mu\text{op}$  are detected in different ways depending on the particular microprocessor. Some microprocessors have PMCs that can give this information directly. In other cases it is necessary to obtain this information indirectly by testing whether a particular instruction or  $\mu\text{op}$  can execute simultaneously with another instruction/ $\mu\text{op}$  that is known to go to a particular execution port or execution unit. On some processors, there is a delay for transmitting data from one execution unit (or cluster of execution units) to another. This delay can be used for detecting whether two different instructions/ $\mu\text{ops}$  are using the same or different execution units.

## Microprocessor versions tested

The tables in this manual are based on testing of the following microprocessors

Processor name	Microarchitecture Code name	Family number (hex)	Model number (hex)	Comment
AMD K7 Athlon		6	6	Step. 2, rev. A5
AMD K8 Opteron		F	5	Stepping A
AMD K10 Opteron		10	2	2350, step. 1
AMD Bulldozer	Bulldozer, Zambezi	15	1	FX-6100, step 2
AMD Bobcat	Bobcat	14	1	E350, step. 0
Intel Pentium		5	2	
Intel Pentium MMX	P5	5	4	Stepping 4
Intel Pentium II	P6	6	6	
Intel Pentium III	P6	6	7	
Intel Pentium 4	Netburst	F	2	Stepping 4, rev. B0
Intel Pentium 4 EM64T	Netburst, Prescott	F	4	Xeon. Stepping 1
Intel Pentium M	Dothan	6	D	Stepping 6, rev. B1
Intel Core Duo	Yonah	6	E	Not fully tested
Intel Core 2 (65 nm)	Merom	6	F	T5500, Step. 6, rev. B2
Intel Core 2 (45 nm)	Wolfdale	6	17	E8400, Step. 6
Intel Core i7	Nehalem	6	1A	i7-920, Step. 5, rev. D0
Intel Core i5	Sandy Bridge	6	2A	i5-2500, Step 7
Intel Atom 330	Diamondville	6	1C	Step. 2
VIA Nano L2200		6	F	Step. 2
VIA Nano L3050	Isaiah	6	F	Step. 8 (prerel. sample)

**AMD K7****List of instruction timings and macro-operation breakdown***Explanation of column headings:*

<b>Instruction:</b>	Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc.
<b>Operands:</b>	i = immediate constant, r = any register, r32 = 32-bit register, etc., mm = 64 bit mmx register, xmm = 128 bit xmm register, sr = segment register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc.
<b>Ops:</b>	Number of macro-operations issued from instruction decoder to schedulers. Instructions with more than 2 macro-operations use microcode.
<b>Latency:</b>	This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's, infinity and exceptions increase the delays. The latency listed does not include the memory operand where the operand is listed as register or memory (r/m).
<b>Reciprocal throughput:</b>	This is also called issue latency. This value indicates the average number of clock cycles from the execution of an instruction begins to a subsequent independent instruction of the same kind can begin to execute. A value of 1/3 indicates that the execution units can handle 3 instructions per clock cycle in one thread. However, the throughput may be limited by other bottlenecks in the pipeline.
<b>Execution unit:</b>	Indicates which execution unit is used for the macro-operations. ALU means any of the three integer ALU's. ALU0_1 means that ALU0 and ALU1 are both used. AGU means any of the three integer address generation units. FADD means floating point adder unit. FMUL means floating point multiplier unit. FMISC means floating point store and miscellaneous unit. FA/M means FADD or FMUL is used. FANY means any of the three floating point units can be used. Two macro-operations can execute simultaneously if they go to different execution units.

**Integer instructions**

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution unit	Notes
<b>Move instructions</b>						
MOV	r,r	1	1	1/3	ALU	
MOV	r,i	1	1	1/3	ALU	
MOV	r8,m8	1	4	1/2	ALU, AGU	Any addr. mode. Add 1 clk if code segment base ≠ 0
MOV	r16,m16	1	4	1/2	ALU, AGU	do.
MOV	r32,m32	1	3	1/2	AGU	do.
MOV	m8,r8H	1	8	1/2	AGU	AH, BH, CH, DH
MOV	m8,r8L	1	2	1/2	AGU	Any other 8-bit register
MOV	m16/32,r	1	2	1/2	AGU	Any addressing mode
MOV	m,i	1	2	1/2	AGU	
MOV	r,sr	1	2	1		

AMD K7

MOV	sr,r/m	6	9-13	8		
MOVZX, MOVSX	r,r	1	1	1/3	ALU	
MOVZX, MOVSX	r,m	1	4	1/2	ALU, AGU	
CMOVcc	r,r	1	1	1/3	ALU	
CMOVcc	r,m	1		1/2	ALU, AGU	
XCHG	r,r	3	2	1	ALU	
XCHG	r,m	3	16	16	ALU, AGU	Timing depends on hw
XLAT		2	5		ALU, AGU	
PUSH	r	1		1	ALU, AGU	
PUSH	i	1		1	ALU, AGU	
PUSH	m	2		1	ALU, AGU	
PUSH	sr	2		1	ALU, AGU	
PUSHF(D)		1		1	ALU, AGU	
PUSHA(D)		9		4	ALU, AGU	
POP	r	2		1	ALU, AGU	
POP	m	3		1	ALU, AGU	
POP	DS/ES/FS/GS	6		10	ALU, AGU	
POP	SS	9		18	ALU, AGU	
POPF(D)		2		1	ALU, AGU	
POPA(D)		9		4	ALU, AGU	
LEA	r16,[m]	2	3	1	AGU	Any addr. size
LEA	r32,[m]	1	2	1/3	AGU	Any addr. size
LAHF		4	3	2	ALU	
SAHF		2	2	2	ALU	
SALC		1	1	1	ALU	
LDS, LES, ...	r,m	10		9		
BSWAP	r	1	1	1/3	ALU	
<b>Arithmetic instructions</b>						
ADD, SUB	r,r/i	1	1	1/3	ALU	
ADD, SUB	r,m	1	1	1/2	ALU, AGU	
ADD, SUB	m,r	1	7	2.5	ALU, AGU	
ADC, SBB	r,r/i	1	1	1/3	ALU	
ADC, SBB	r,m	1	1	1/2	ALU, AGU	
ADC, SBB	m,r/i	1	7	2.5	ALU, AGU	
CMP	r,r/i	1	1	1/3	ALU	
CMP	r,m	1		1/2	ALU, AGU	
INC, DEC, NEG	r	1	1	1/3	ALU	
INC, DEC, NEG	m	1	7	3	ALU, AGU	
AAA, AAS		9	5	5	ALU	
DAA		12	6	6	ALU	
DAS		16	7	7	ALU	
AAD		4	5		ALU0	
AAM		31	13		ALU	
MUL, IMUL	r8/m8	3	3	2	ALU0	
MUL, IMUL	r16/m16	3	3	2	ALU0_1	latency ax=3, dx=4
MUL, IMUL	r32/m32	3	4	3	ALU0_1	
IMUL	r16,r16/m16	2	3	2	ALU0	



AMD K7

IMUL	r32,r32/m32	2	4	2.5	ALU0
IMUL	r16,(r16),i	2	4	1	ALU0
IMUL	r32,(r32),i	2	5	2	ALU0
IMUL	r16,m16,i	3		2	ALU0
IMUL	r32,m32,i	3		2	ALU0
DIV	r8/m8	32	24	23	ALU
DIV	r16/m16	47	24	23	ALU
DIV	r32/m32	79	40	40	ALU
IDIV	r8	41	17	17	ALU
IDIV	r16	56	25	25	ALU
IDIV	r32	88	41	41	ALU
IDIV	m8	42	17	17	ALU
IDIV	m16	57	25	25	ALU
IDIV	m32	89	41	41	ALU
CBW, CWDE		1	1	1/3	ALU
CWD, CDQ		1	1	1/3	ALU
<b>Logic instructions</b>					
AND, OR, XOR	r,r	1	1	1/3	ALU
AND, OR, XOR	r,m	1	1	1/2	ALU, AGU
AND, OR, XOR	m,r	1	7	2.5	ALU, AGU
TEST	r,r	1	1	1/3	ALU
TEST	r,m	1	1	1/2	ALU, AGU
NOT	r	1	1	1/3	ALU
NOT	m	1	7	2.5	ALU, AGU
SHL, SHR, SAR	r,i/CL	1	1	1/3	ALU
ROL, ROR	r,i/CL	1	1	1/3	ALU
RCL, RCR	r,1	1	1	1/3	ALU
RCL	r,i	9	4	4	ALU
RCR	r,i	7	3	3	ALU
RCL	r,CL	9	3	3	ALU
RCR	r,CL	7	3	3	ALU
SHL,SHR,SAR,ROL,ROR	m,i /CL	1	7	3	ALU, AGU
RCL, RCR	m,1	1	7	4	ALU, AGU
RCL	m,i	10	5	4	ALU, AGU
RCR	m,i	9	8	4	ALU, AGU
RCL	m,CL	9	6	4	ALU, AGU
RCR	m,CL	8	7	3	ALU, AGU
SHLD, SHRD	r,r,i	6	4	2	ALU
SHLD, SHRD	r,r,cl	7	4	3	ALU
SHLD, SHRD	m,r,i/CL	8	7	3	ALU, AGU
BT	r,r/i	1	1	1/3	ALU
BT	m,i	1		1/2	ALU, AGU
BT	m,r	5		2	ALU, AGU
BTC, BTR, BTS	r,r/i	2	2	1	ALU
BTC	m,i	5	7	2	ALU, AGU
BTR, BTS	m,i	4	7	2	ALU, AGU
BTC, BTR, BTS	m,r	8	6	3	ALU, AGU
BSF	r,r	19	7	7	ALU
BSR	r,r	23	9	9	ALU

AMD K7

BSF	r,m	20	8	8	ALU, AGU	
BSR	r,m	23	10	10	ALU, AGU	
SETcc	r	1	1	1/3	ALU	
SETcc	m	1		1/2	ALU, AGU	
CLC, STC		1		1/3	ALU	
CMC		1	1	1/3	ALU	
CLD		2		1	ALU	
STD		3		2	ALU	
<b>Control transfer instructions</b>						
JMP	short/near	1		2	ALU	
JMP	far	16-20	23-32			low values = real mode
JMP	r	1		2	ALU	
JMP	m(near)	1		2	ALU, AGU	
JMP	m(far)	17-21	25-33			low values = real mode
Jcc	short/near	1		1/3 - 2	ALU	rcp. t.= 2 if jump
J(E)CXZ	short	2		1/3 - 2	ALU	rcp. t.= 2 if jump
LOOP	short	7	3-4	3-4	ALU	
CALL	near	3	2	2	ALU	
CALL	far	16-22	23-32			low values = real mode
CALL	r	4	3	3	ALU	
CALL	m(near)	5	3	3	ALU, AGU	
CALL	m(far)	16-22	24-33			low values = real mode
RETN		2	3	3	ALU	
RETN	i	2	3	3	ALU	
RETF		15-23	24-35			low values = real mode
RETF	i	15-24	24-35			low values = real mode
IRET		32	81			real mode
INT	i	33	42			real mode
BOUND	m	6		2		values are for no jump
INTO		2		2		values are for no jump
<b>String instructions</b>						
LODS		4	2	2		
REP LODS		5	2	2		values per count
STOS		4	2	2		
REP STOS		3	1	1		values per count
MOVS		7	3	3		
REP MOVS		4	1-4	1-4		values per count
SCAS		5	2	2		
REP SCAS		5	2	2		values per count
CMPS		7	6	6		
REP CMPS		6	3-4	3-4		values per count

AMD K7

<b>Other</b>						
NOP (90)		1	0	1/3	ALU	
Long NOP (0F 1F)		1	0	1/3	ALU	
ENTER		i,0	12	12	12	
LEAVE		3		3		3 ops, 5 clk if 16 bit
CLI		8-9		5		
STI		16-17		27		
CPUID		19-28	44-74			
RDTSC		5		11		
RDPMC		9		11		

**Floating point x87 instructions**

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution unit	Notes
<b>Move instructions</b>						
FLD	r	1	2	1/2	FA/M	
FLD	m32/64	1	4	1/2	FANY	
FLD	m80	7	16	4		
FBLD	m80	30	41	39		
FST(P)	r	1	2	1/2	FA/M	
FST(P)	m32/64	1	3	1	FMISC	
FSTP	m80	10	7	5		
FBSTP	m80	260		188		
FXCH	r	1	0	0.4		
FILD	m	1	9	1	FMISC	
FIST(P)	m	1	7	1	FMISC, FA/M	
FLDZ, FLD1		1		1	FMISC	
FCMOVcc	st0,r	9	6	5	FMISC, FA/M	Low latency immediately after FCOMI
FFREE	r	1		1/3	FANY	
FINCSTP, FDECSTP		1	0	1/3	FANY	
FNSTSW	AX	2	6-12	12	FMISC, ALU	Low latency immediately after FCOM FTST
FSTSW	AX	3	6-12	12	FMISC, ALU	do.
FNSTSW	m16	2		8	FMISC, ALU	do.
FNSTCW	m16	3		1	FMISC, ALU	
FLDCW	m16	14		42	FMISC, ALU	faster if unchanged
<b>Arithmetic instructions</b>						
FADD(P),FSUB(R)(P)	r/m	1	4	1	FADD	
FIADD,FISUB(R)	m	2	4	1-2	FADD,FMISC	
FMUL(P)	r/m	1	4	1	FMUL	
FIMUL	m	2	4	2	FMUL,FMISC	
FDIV(R)(P)	r/m	1	11-25	8-22	FMUL	Low values are for round divisors

AMD K7

FIDIV(R)	m	2	12-26	9-23	FMUL,FMISC	do.
FABS, FCHS		1	2	1	FMUL	
FCOM(P), FUCOM(P)	r/m	1	2	1	FADD	
FCOMPP, FUCOMPP		1	2	1	FADD	
FCOMI(P)	r	1	3	1	FADD	
FICOM(P)	m	2		1	FADD, FMISC	
FTST		1	2	1	FADD	
FXAM		2		2	FMISC, ALU	
FRNDINT		5	10	3		
FPREM		1	7-10	8	FMUL	
FPREM1		1	8-11	8	FMUL	
<b>Math</b>						
FSQRT		1	35	12	FMUL	
FSIN		44	90-100			
FCOS		51	90-100			
FSINCOS		76	100-150			
FPTAN		46	100-200			
FPATAN		72	160-170			
FSCALE		5	8			
FEXTRACT		7	11			
F2XM1		8	27			
FYL2X		49	126			
FYL2XP1		63	147			
<b>Other</b>						
FNOP		1	0	1/3	FANY	
(F)WAIT		1	0	1/3	ALU	
FNCLEX		7		24	FMISC	
FNINIT		25		92	FMISC	
FNSAVE		76		147		
FRSTOR		65		120		
FXSAVE		44		59		
FXRSTOR		85		87		

**Integer MMX instructions**

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution unit	Notes
<b>Move instructions</b>						
MOVD	r32, mm	2	7	2	FMISC, ALU	
MOVD	mm, r32	2	9	2	FANY, ALU	
MOVD	mm,m32	1		1/2	FANY	
MOVD	m32, r	1		1	FMISC	
MOVQ	mm,mm	1	2	1/2	FA/M	
MOVQ	mm,m64	1		1/2	FANY	
MOVQ	m64,mm	1		1	FMISC	
MOVNTQ	m,mm	1		2	FMISC	
PACKSSWB/DW PACK-USWB	mm,r/m	1	2	2	FA/M	

AMD K7

PUNPCKH/LBW/WD	mm,r/m	1	2	2	FA/M
PSHUFW	mm,mm,i	1	2	1/2	FA/M
MASKMOVQ	mm,mm	32		24	
PMOVMSKB	r32,mm	3		3	FADD
PEXTRW	r32,mm,i	2	5	2	FMISC, ALU
PINSRW	mm,r32,i	2	12	2	FA/M
<b>Arithmetic instructions</b>					
PADDB/W/D PADDWB/W					
PADDUSB/W					
PSUBB/W/D PSUBSB/W					
PSUBUSB/W					
	mm,r/m	1	2	1/2	FA/M
PCMPEQ/GT B/W/D	mm,r/m	1	2	1/2	FA/M
PMULLW PMULHW					
PMULHUW	mm,r/m	1	3	1	FMUL
PMADDWD	mm,r/m	1	3	1	FMUL
PAVGB/W	mm,r/m	1	2	1/2	FA/M
PMIN/MAX SW/UB	mm,r/m	1	2	1/2	FA/M
PSADBW	mm,r/m	1	3	1	FADD
<b>Logic</b>					
PAND PANDN POR					
PXOR	mm,r/m	1	2	1/2	FA/M
PSLL/RLW/D/Q					
PSRAW/D	mm,i/mm/m	1	2	1/2	FA/M
<b>Other</b>					
EMMS		1		1/3	FANY

**Floating point XMM instructions**

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution unit	Notes
<b>Move instructions</b>						
MOVAPS	r,r	2	2	1	FA/M	
MOVAPS	r,m	2		2	FMISC	
MOVAPS	m,r	2		2	FMISC	
MOVUPS	r,r	2	2	1	FA/M	
MOVUPS	r,m	5		2		
MOVUPS	m,r	5		2		
MOVSS	r,r	1	2	1	FA/M	
MOVSS	r,m	2	4	1	FANY FMISC	
MOVSS	m,r	1	3	1	FMISC	
MOVHPS, MOVLHPS	r,r	1	2	1/2	FA/M	
MOVHPS, MOVLPS	r,m	1		1/2	FMISC	
MOVHPS, MOVLPS	m,r	1		1	FMISC	
MOVNTPS	m,r	2		4	FMISC	
MOVMSKPS	r32,r	3		2	FADD	
SHUFPS	r,r/m,i	3	3	3	FMUL	

AMD K7

UNPCK H/L PS	r,r/m	2	3	3	FMUL		
<b>Conversion</b>							
CVTPI2PS	xmm,mm	1	4		FMISC		
CVT(T)PS2PI	mm,xmm	1	6		FMISC		
CVTSI2SS	xmm,r32	4		10	FMISC		
CVT(T)SS2SI	r32,xmm	2		3	FMISC		
<b>Arithmetic</b>							
ADDSS SUBSS	r,r/m	1	4	1	FADD		
ADDPSS SUBPSS	r,r/m	2	4	2	FADD		
MULSS	r,r/m	1	4	1	FMUL		
MULPSS	r,r/m	2	4	2	FMUL		
DIVSS	r,r/m	1	11-16	8-13	FMUL	Low values are for round divisors, e.g. powers of 2. do.	
DIVPSS	r,r/m	2	18-30	18-30	FMUL		
RCPSS	r,r/m	1	3	1	FMUL		
RCPPSS	r,r/m	2	3	2	FMUL		
MAXSS MINSS	r,r/m	1	2	1	FADD		
MAXPSS MINPSS	r,r/m	2	2	2	FADD		
CMPccSS	r,r/m	1	2	1	FADD		
CMPccPSS	r,r/m	2	2	2	FADD		
COMISS UCOMISS	r,r/m	1	2	1	FADD		
<b>Logic</b>							
ANDPSS/D ANDNPSS/D ORPSS/D XORPSS/D	r,r/m	2	2	2	FMUL		
<b>Math</b>							
SQRTSS	r,r/m	1	19	16	FMUL		
SQRTPSS	r,r/m	2	36	36	FMUL		
RSQRTSS	r,r/m	1	3	1	FMUL		
RSQRTPSS	r,r/m	2	3	2	FMUL		
<b>Other</b>							
LDMXCSR	m	8		9			
STMXCSR	m	3		10			

**3DNow instructions (obsolete)**

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution unit	Notes
<b>Move and convert instructions</b>						
PREFETCH(W)	m	1		1/2	AGU	
PF2ID	mm,mm	1	5	1	FMISC	
PI2FD	mm,mm	1	5	1	FMISC	
PF2IW	mm,mm	1	5	1	FMISC	3DNow E
PI2FW	mm,mm	1	5	1	FMISC	3DNow E
PSWAPD	mm,mm	1	2	1/2	FA/M	3DNow E

AMD K7

<b>Integer instructions</b>						
PAVGUSB	mm,mm	1	2	1/2	FAM	
PMULHRW	mm,mm	1	3	1	FMUL	
<b>Floating point instructions</b>						
PFADD/SUB/SUBR	mm,mm	1	4	1	FADD	
PFCMPEQ/GE/GT	mm,mm	1	2	1	FADD	
PFMAX/MIN	mm,mm	1	2	1	FADD	
PFMUL	mm,mm	1	4	1	FMUL	
PFACC	mm,mm	1	4	1	FADD	
PFNACC, PFPNACC	mm,mm	1	4	1	FADD	3DNow E
PFRCP	mm,mm	1	3	1	FMUL	
PFRCPIT1/2	mm,mm	1	4	1	FMUL	
PFRSQRT	mm,mm	1	3	1	FMUL	
PFRSQIT1	mm,mm	1	4	1	FMUL	
<b>Other</b>						
FEMMS	mm,mm	1		1/3	FANY	

## AMD K8

### List of instruction timings and macro-operation breakdown

#### Explanation of column headings:

<b>Instruction:</b>	Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc.
<b>Operands:</b>	i = immediate constant, r = any register, r32 = 32-bit register, etc., mm = 64 bit mmx register, xmm = 128 bit xmm register, sr = segment register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc.
<b>Ops:</b>	Number of macro-operations issued from instruction decoder to schedulers. Instructions with more than 2 macro-operations use microcode.
<b>Latency:</b>	This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's, infinity and exceptions increase the delays. The latency listed does not include the memory operand where the operand is listed as register or memory (r/m).
<b>Reciprocal throughput:</b>	This is also called issue latency. This value indicates the average number of clock cycles from the execution of an instruction begins to a subsequent independent instruction of the same kind can begin to execute. A value of 1/3 indicates that the execution units can handle 3 instructions per clock cycle in one thread. However, the throughput may be limited by other bottlenecks in the pipeline.
<b>Execution unit:</b>	Indicates which execution unit is used for the macro-operations. ALU means any of the three integer ALU's. ALU0_1 means that ALU0 and ALU1 are both used. AGU means any of the three integer address generation units. FADD means floating point adder unit. FMUL means floating point multiplier unit. FMISC means floating point store and miscellaneous unit. FA/M means FADD or FMUL is used. FANY means any of the three floating point units can be used. Two macro-operations can execute simultaneously if they go to different execution units.

### Integer instructions

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution unit	Notes
<b>Move instructions</b>						
MOV	r,r	1	1	1/3	ALU	
MOV	r,i	1	1	1/3	ALU	
MOV	r8,m8	1	4	1/2	ALU, AGU	Any addressing mode. Add 1 clock if code segment base $\neq 0$
MOV	r16,m16	1	4	1/2	ALU, AGU	
MOV	r32,m32	1	3	1/2	AGU	
MOV	r64,m64	1	3	1/2	AGU	
MOV	m8,r8H	1	8	1/2	AGU	AH, BH, CH, DH
MOV	m8,r8L	1	3	1/2	AGU	Any other 8-bit register
MOV	m16/32/64,r	1	3	1/2	AGU	Any addressing mode
MOV	m,i	1	3	1/2	AGU	
MOV	m64,i32	1	3	1/2	AGU	
MOV	r,sr	1	2	1/2-1		
MOV	sr,r/m	6	9-13	8		



## K8

MOVNTI	m,r	1		2-3	AGU	
MOVZX, MOVZX	r,r	1	1	1/3	ALU	
MOVZX, MOVZX	r,m	1	4	1/2	ALU, AGU	
MOVSLD	r64,r32	1	1	1/3	ALU	
MOVSLD	r64,m32	1		1/2	ALU, AGU	
CMOVCc	r,r	1	1	1/3	ALU	
CMOVCc	r,m	1		1/2	ALU, AGU	
XCHG	r,r	3	2	1	ALU	
XCHG	r,m	3	16	16	ALU, AGU	Timing depends on hw
XLAT		2	5		ALU, AGU	
PUSH	r	1	1	1	ALU, AGU	
PUSH	i	1	1	1	ALU, AGU	
PUSH	m	2	1	1	ALU, AGU	
PUSH	sr	2	1	1	ALU, AGU	
PUSHF(D/Q)		5	2	2	ALU, AGU	
PUSHA(D)		9	4	4	ALU, AGU	
POP	r	2	1	1	ALU, AGU	
POP	m	3	1	1	ALU, AGU	
POP	DS/ES/FS/GS	4-6	8	8	ALU, AGU	
POP	SS	7-9	28	28	ALU, AGU	
POPF(D/Q)		25	10	10	ALU, AGU	
POPA(D)		9	4	4	ALU, AGU	
LEA	r16,[m]	2	3	1	AGU	Any address size
LEA	r32,[m]	1	2	1/3	AGU	Any address size
LEA	r64,[m]	1	2	1/3	AGU	Any address size
LAHF		4	3	2	ALU	
SAHF		1	1	1/3	ALU	
SALC		1	1	1/3	ALU	
LDS, LES, ...	r,m	10		9		
BSWAP	r	1	1	1/3	ALU	
PREFETCHNTA	m	1		1/2	AGU	
PREFETCHT0/1/2	m	1		1/2	AGU	
SFENCE		6		8		
LFENCE		1		5		
MFENCE		7		16		
IN	r,i/DX	270				
OUT	i/DX,r	300				
<b>Arithmetic instructions</b>						
ADD, SUB	r,r/i	1	1	1/3	ALU	
ADD, SUB	r,m	1	1	1/2	ALU, AGU	
ADD, SUB	m,r	1	7	2.5	ALU, AGU	
ADC, SBB	r,r/i	1	1	1/3	ALU	
ADC, SBB	r,m	1	1	1/2	ALU, AGU	
ADC, SBB	m,r/i	1	7	2.5	ALU, AGU	
CMP	r,r/i	1	1	1/3	ALU	
CMP	r,m	1		1/2	ALU, AGU	
INC, DEC, NEG	r	1	1	1/3	ALU	
INC, DEC, NEG	m	1	7	3	ALU, AGU	

## K8

AAA, AAS		9	5	5	ALU	
DAA		12	6	6	ALU	
DAS		16	7	7	ALU	
AAD		4	5		ALU0	
AAM		31	13		ALU	
MUL, IMUL	r8/m8	1	3	1	ALU0	
MUL, IMUL	r16/m16	3	3-4	2	ALU0_1	latency ax=3, dx=4
MUL, IMUL	r32/m32	2	3	1	ALU0_1	
MUL, IMUL	r64/m64	2	4-5	2	ALU0_1	latency rax=4, rdx=5
IMUL	r16,r16/m16	1	3	1	ALU0	
IMUL	r32,r32/m32	1	3	1	ALU0	
IMUL	r64,r64/m64	1	4	2	ALU0_1	
IMUL	r16,(r16),i	2	4	1	ALU0	
IMUL	r32,(r32),i	1	3	1	ALU0	
IMUL	r64,(r64),i	1	4	2	ALU0	
IMUL	r16,m16,i	3		2	ALU0	
IMUL	r32,m32,i	3		2	ALU0	
IMUL	r64,m64,i	3		2	ALU0_1	
DIV	r8/m8	31	15	15	ALU	
DIV	r16/m16	46	23	23	ALU	
DIV	r32/m32	78	39	39	ALU	
DIV	r64/m64	143	71	71	ALU	
IDIV	r8	40	17	17	ALU	
IDIV	r16	55	25	25	ALU	
IDIV	r32	87	41	41	ALU	
IDIV	r64	152	73	73	ALU	
IDIV	m8	41	17	17	ALU	
IDIV	m16	56	25	25	ALU	
IDIV	m32	88	41	41	ALU	
IDIV	m64	153	73	73	ALU	
CBW, CWDE, CDQE		1	1	1/3	ALU	
CWD, CDQ, CQO		1	1	1/3	ALU	
<b>Logic instructions</b>						
AND, OR, XOR	r,r	1	1	1/3	ALU	
AND, OR, XOR	r,m	1	1	1/2	ALU, AGU	
AND, OR, XOR	m,r	1	7	2.5	ALU, AGU	
TEST	r,r	1	1	1/3	ALU	
TEST	r,m	1	1	1/2	ALU, AGU	
NOT	r	1	1	1/3	ALU	
NOT	m	1	7	2.5	ALU, AGU	
SHL, SHR, SAR	r,i/CL	1	1	1/3	ALU	
ROL, ROR	r,i/CL	1	1	1/3	ALU	
RCL, RCR	r,1	1	1	1/3	ALU	
RCL	r,i	9	3	3	ALU	
RCR	r,i	7	3	3	ALU	
RCL	r,CL	9	4	4	ALU	
RCR	r,CL	7	3	3	ALU	
SHL,SHR,SAR,ROL,ROR	m,i /CL	1	7	3	ALU, AGU	

## K8

RCL, RCR	m,1	1	7	4	ALU, AGU	
RCL	m,i	10	9	4	ALU, AGU	
RCR	m,i	9	8	4	ALU, AGU	
RCL	m,CL	9	7	4	ALU, AGU	
RCR	m,CL	8	8	3	ALU, AGU	
SHLD, SHRD	r,r,i	6	3	3	ALU	
SHLD, SHRD	r,r,cl	7	3	3	ALU	
SHLD, SHRD	m,r,i/CL	8	6	3	ALU, AGU	
BT	r,r/i	1	1	1/3	ALU	
BT	m,i	1		1/2	ALU, AGU	
BT	m,r	5		2	ALU, AGU	
BTC, BTR, BTS	r,r/i	2	2	1	ALU	
BTC	m,i	5	7	2	ALU, AGU	
BTR, BTS	m,i	4	7	2	ALU, AGU	
BTC	m,r	8	5	5	ALU, AGU	
BTR, BTS	m,r	8	8	3	ALU, AGU	
BSF	r16/32,r	21	8	8	ALU	
BSF	r64,r	22	9	9	ALU	
BSR	r,r	28	10	10	ALU	
BSF	r16,m	20	8	8	ALU, AGU	
BSF	r32,m	22	9	9	ALU, AGU	
BSF	r64,m	25	10	10	ALU, AGU	
BSR	r,m	28	10	10	ALU, AGU	
SETcc	r	1	1	1/3	ALU	
SETcc	m	1		1/2	ALU, AGU	
CLC, STC		1		1/3	ALU	
CMC		1	1	1/3	ALU	
CLD		1		1/3	ALU	
STD		2		1/3	ALU	
<b>Control transfer instructions</b>						
JMP	short/near	1		2	ALU	
JMP	far	16-20	23-32			low values = real mode
JMP	r	1		2	ALU	
JMP	m(near)	1		2	ALU, AGU	
JMP	m(far)	17-21	25-33			low values = real mode
Jcc	short/near	1		1/3 - 2	ALU	recip. thrp.= 2 if jump
J(E/R)CXZ	short	2		1/3 - 2	ALU	recip. thrp.= 2 if jump
LOOP	short	7	3-4	3-4	ALU	
CALL	near	3	2	2	ALU	
CALL	far	16-22	23-32			low values = real mode
CALL	r	4	3	3	ALU	
CALL	m(near)	5	3	3	ALU, AGU	
CALL	m(far)	16-22	24-33			low values = real mode
RETN		2	3	3	ALU	
RETN	i	2	3	3	ALU	
RETF		15-23	24-35			low values = real mode
RETF	i	15-24	24-35			low values = real mode
IRET		32	81			real mode
INT	i	33	42			real mode

## K8

BOUND	m	6		2		values are for no jump
INTO		2		2		values are for no jump
<b>String instructions</b>						
LODS		4	2	2		values are per count
REP LODS		5	2	2		
STOS		4	2	2		
REP STOS		1.5 - 2	0.5 - 1	0.5 - 1		values are per count
MOVS		7	3	3		values are per count
REP MOVS		3	1-2	1-2		
SCAS		5	2	2		
REP SCAS		5	2	2		values are per count
CMPS		2	3	3		values are per count
REP CMPS		6	2	2		
<b>Other</b>						
NOP (90)		1	0	1/3	ALU	3 ops, 5 clk if 16 bit
Long NOP (0F 1F)		1	0	1/3	ALU	
ENTER		i,0	12	12	12	
LEAVE		2		3		
CLI		8-9		5		
STI		16-17		27		
CPUID		22-50	47-164			
RDTSC		6	10	7		
RDPMC		9	12	7		

## Floating point x87 instructions

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution unit	Notes
<b>Move instructions</b>						
FLD	r	1	2	1/2	FA/M	Low latency immediately after FCOMI
FLD	m32/64	1	4	1/2	FANY	
FLD	m80	7	16	4		
FBLD	m80	30	41	39		
FST(P)	r	1	2	1/2	FA/M	
FST(P)	m32/64	1	3	1	FMISC	
FSTP	m80	10	7	5		
FBSTP	m80	260	173	160		
FXCH	r	1	0	0.4		
FILD	m	1	9	1	FMISC	
FIST(P)	m	1	7	1	FMISC, FA/M	
FLDZ, FLD1		1		1	FMISC	
FCMOVcc	st0,r	9	4-15	4	FMISC, FA/M	
FFREE	r	1		2	FANY	
FINCSTP, FDECSTP		1	0	1/3	FANY	
FNSTSW	AX	2	6-12	12	FMISC, ALU	
FSTSW	AX	3	6-12	12	FMISC, ALU	do.

## K8

FNSTSW	m16	2		8	FMISC, ALU	do.
FNSTCW	m16	3		1	FMISC, ALU	
FLDCW	m16	18		50	FMISC, ALU	faster if unchanged
<b>Arithmetic instructions</b>						
FADD(P),FSUB(R)(P)	r/m	1	4	1	FADD	
FIADD,FISUB(R)	m	2	4	1-2	FADD,FMISC	
FMUL(P)	r/m	1	4	1	FMUL	
FIMUL	m	2	4	2	FMUL,FMISC	
FDIV(R)(P)	r/m	1	11-25	8-22	FMUL	Low values are for round divisors
FIDIV(R)	m	2	12-26	9-23	FMUL,FMISC	do.
FABS, FCHS		1	2	1	FMUL	
FCOM(P), FUCOM(P)	r/m	1	2	1	FADD	
FCOMPP, FUCOMPP		1	2	1	FADD	
FCOMI(P)	r	1	3	1	FADD	
FICOM(P)	m	2		1	FADD, FMISC	
FTST		1	2	1	FADD	
FXAM		2		1	FMISC, ALU	
FRNDINT		5	10	3		
FPREM		1	7-10	8	FMUL	
FPREM1		1	8-11	8	FMUL	
<b>Math</b>						
FSQRT		1	27	12	FMUL	
FLDPI, etc.		1		1	FMISC	
FSIN		66	140-190			
FCOS		73	150-190			
FSINCOS		98	170-200			
FPTAN		67	150-180			
FPATAN		97	217			
FSCALE		5	8			
FXTRACT		7	12	7		
F2XM1		53	126			
FYL2X		72	179			
FYL2XP1		75	175			
<b>Other</b>						
FNOP		1	0	1/3	FANY	
(F)WAIT		1	0	1/3	ALU	
FNCLEX		8		27	FMISC	
FNINIT		26		100	FMISC	
FNSAVE		77		171		
FRSTOR		70		136		
FXSAVE		61		56		
FXRSTOR		101		95		

**Integer MMX and XMM instructions**

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution unit	Notes
<b>Move instructions</b>						
MOVD	r32, mm	2	4	2	FMISC, ALU	
MOVD	mm, r32	2	9	2	FANY, ALU	
MOVD	mm,m32	1		1/2	FANY	
MOVD	r32, xmm	3	2	2	FMISC, ALU	
MOVD	xmm, r32	3	3	2		
MOVD	xmm,m32	2		1	FANY	
MOVD	m32, r	1		1	FMISC	
MOVD (MOVQ)	r64,mm/xmm	2	4	2	FMISC, ALU	Moves 64 bits.Name of instruction differs
MOVD (MOVQ)	mm,r64	2	9	2	FANY, ALU	do.
MOVD (MOVQ)	xmm,r64	3	9	2	FANY, ALU	do.
MOVQ	mm,mm	1	2	1/2	FA/M	
MOVQ	xmm,xmm	2	2	1	FA/M, FMISC	
MOVQ	mm,m64	1		1/2	FANY	
MOVQ	xmm,m64	2		1	FANY, FMISC	
MOVQ	m64,mm/x	1		1	FMISC	
MOVDQA	xmm,xmm	2	2	1	FA/M	
MOVDQA	xmm,m	2		2	FMISC	
MOVDQA	m,xmm	2		2	FMISC	
MOVDQU	xmm,m	4		2		
MOVDQU	m,xmm	5		2		
MOVDQ2Q	mm,xmm	1	2	1/2	FA/M	
MOVQ2DQ	xmm,mm	2	2	1	FA/M, FMISC	
MOVNTQ	m,mm	1		2	FMISC	
MOVNTDQ	m,xmm	2		3	FMISC	
PACKSSWB/DW						
PACKUSWB	mm,r/m	1	2	2	FA/M	
PACKSSWB/DW						
PACKUSWB	xmm,r/m	3	3	2	FA/M	
PUNPCKH/LBW/WD/DQ	mm,r/m	1	2	2	FA/M	
PUNPCKH/LBW/WD/DQ	xmm,r/m	2	2	2	FA/M	
PUNPCKHQDQ	xmm,r/m	2	2	1	FA/M	
PUNPCKLQDQ	xmm,r/m	1	2	1/2	FA/M	
PSHUFD	xmm,xmm,i	3	3	1.5	FA/M	
PSHUFW	mm,mm,i	1	2	1/2	FA/M	
PSHUFL/HW	xmm,xmm,i	2	2	1	FA/M	
MASKMOVQ	mm,mm	32		13		
MASKMOVDQU	xmm,xmm	64		26		
PMOVMASKB	r32,mm/xmm	1	2	1	FADD	
PEXTRW	r32,mm/x,i	2	5	2	FMISC, ALU	
PINSRW	mm,r32,i	2	12	2	FA/M	
PINSRW	xmm,r32,i	3	12	3	FA/M	
<b>Arithmetic instructions</b>						

PADDB/W/D/Q PADDSB/W PADDUSB/W PSUBB/W/D/Q PSUBSB/W PSUBUSB/W	mm,r/m	1	2	1/2	FA/M
PADDB/W/D/Q PADDSB/W ADDUSB/W PSUBB/W/D/Q PSUBSB/W PSUBUSB/W	xmm,r/m	2	2	1	FA/M
PCMPEQ/GT B/W/D PCMPEQ/GT B/W/D	mm,r/m xmm,r/m	1 2	2 2	1/2 1	FA/M FA/M
PMULLW PMULHW PMULHUW PMULUDQ	mm,r/m	1	3	1	FMUL
PMULLW PMULHW PMULHUW PMULUDQ	xmm,r/m	2	3	2	FMUL
PMADDWD PMADDWD	mm,r/m xmm,r/m	1 2	3 3	1 2	FMUL FMUL
PAVGB/W PAVGB/W	mm,r/m xmm,r/m	1 2	2 2	1/2 1	FA/M FA/M
PMIN/MAX SW/UB PMIN/MAX SW/UB	mm,r/m xmm,r/m	1 2	2 2	1/2 1	FA/M FA/M
PSADBW PSADBW	mm,r/m xmm,r/m	1 2	3 3	1 2	FADD FADD
<b>Logic</b>					
PAND PANDN POR PXOR	mm,r/m	1	2	1/2	FA/M
PAND PANDN POR PXOR	xmm,r/m	2	2	1	FA/M
PSLL/RL W/D/Q PSRAW/D	mm,i/mm/m	1	2	1/2	FA/M
PSLL/RL W/D/Q PSRAW/D	x,i/x/m	2	2	1	FA/M
PSLLDQ, PSRLDQ	xmm,i	2	2	1	FA/M
<b>Other</b>					
EMMS		1		1/3	FANY

### Floating point XMM instructions

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution unit	Notes
<b>Move instructions</b>						
MOVAPS/D	r,r	2	2	1	FA/M	
MOVAPS/D	r,m	2		2	FMISC	
MOVAPS/D	m,r	2		2	FMISC	
MOVUPS/D	r,r	2	2	1	FA/M	

## K8

MOVUPS/D	r,m	4		2		
MOVUPS/D	m,r	5		2		
MOVSS/D	r,r	1	2	1	FA/M	
MOVSS/D	r,m	2	4	1	FANY FMISC	
MOVSS/D	m,r	1	3	1	FMISC	
MOVHLPs, MOVLHPS	r,r	1	2	1/2	FA/M	
MOVHPS/D, MOVLPS/D	r,m	1		1	FMISC	
MOVHPS/D, MOVLPS/D	m,r	1		1	FMISC	
MOVDDUP	r,r	2	2	1		SSE3
MOVSH/LDUP	r,r	2	2	2		SSE3
MOVNTPS/D	m,r	2		3	FMISC	
MOVMSKPS/D	r32,r	1	8	1	FADD	
SHUFPS/D	r,r/m,i	3	3	2	FMUL	
UNPCK H/L PS/D	r,r/m	2	3	3	FMUL	
<b>Conversion</b>						
CVTSP2PD	r,r/m	2	4	2	FMISC	
CVTPD2PS	r,r/m	4	8	3	FMISC	
CVTSD2SS	r,r/m	3	8	8	FMISC	
CVTSS2SD	r,r/m	1	2	1	FMISC	
CVTDQ2PS	r,r/m	2	5	2	FMISC	
CVTDQ2PD	r,r/m	2	5	2	FMISC	
CVT(T)PS2DQ	r,r/m	2	5	2	FMISC	
CVT(T)PD2DQ	r,r/m	4	8	3	FMISC	
CVTPI2PS	xmm,mm	1	4	1	FMISC	
CVTPI2PD	xmm,mm	2	5	2	FMISC	
CVT(T)PS2PI	mm,xmm	1	6	1	FMISC	
CVT(T)PD2PI	mm,xmm	3	8	2	FMISC	
CVTSI2SS	xmm,r32	3	14	2	FMISC	
CVTSI2SD	xmm,r32	2	12	2	FMISC	
CVT(T)SD2SI	r32,xmm	2	10	2	FMISC	
CVT(T)SS2SI	r32,xmm	2	9	2	FMISC	
<b>Arithmetic</b>						
ADDSS/D SUBSS/D	r,r/m	1	4	1	FADD	
ADDPS/D SUBPS/D	r,r/m	2	4	2	FADD	
HADDPS/D						
HSUBPS/D	r,r/m	2	4	2	FADD	SSE3
MULSS/D	r,r/m	1	4	1	FMUL	
MULPS/D	r,r/m	2	4	2	FMUL	
Low values are for round divisors, e.g. powers of 2.						
DIVSS	r,r/m	1	11-16	8-13	FMUL	
DIVPS	r,r/m	2	18-30	18-30	FMUL	do.
DIVSD	r,r/m	1	11-20	8-17	FMUL	do.
DIVPD	r,r/m	2	16-34	16-34	FMUL	do.
RCPSS	r,r/m	1	3	1	FMUL	
RCPSS	r,r/m	2	3	2	FMUL	



## K8

MAXSS/D MINSS/D	r,r/m	1	2	1	FADD
MAXPS/D MINPS/D	r,r/m	2	2	2	FADD
CMPccSS/D	r,r/m	1	2	1	FADD
CMPccPS/D	r,r/m	2	2	2	FADD
COMISS/D					
UCOMISS/D	r,r/m	1	2	1	FADD
<b>Logic</b>					
ANDPS/D ANDNPS/D					
ORPS/D XORPS/D	r,r/m	2	2	2	FMUL
<b>Math</b>					
SQRTSS	r,r/m	1	19	16	FMUL
SQRTPS	r,r/m	2	36	36	FMUL
SQRTSD	r,r/m	1	27	24	FMUL
SQRTPD	r,r/m	2	48	48	FMUL
RSQRTSS	r,r/m	1	3	1	FMUL
RSQRTPS	r,r/m	2	3	2	FMUL
<b>Other</b>					
LDMXCSR	m	8		9	
STMXCSR	m	3		10	

## AMD K10

### List of instruction timings and macro-operation breakdown

#### Explanation of column headings:

<b>Instruction:</b>	Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc.
<b>Operands:</b>	i = immediate constant, r = any register, r32 = 32-bit register, etc., mm = 64 bit mmx register, xmm = 128 bit xmm register, sr = segment register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc.
<b>Ops:</b>	Number of macro-operations issued from instruction decoder to schedulers. Instructions with more than 2 macro-operations use microcode.
<b>Latency:</b>	This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's, infinity and exceptions increase the delays. The latency listed does not include the memory operand where the operand is listed as register or memory (r/m).
<b>Reciprocal throughput:</b>	This is also called issue latency. This value indicates the average number of clock cycles from the execution of an instruction begins to a subsequent independent instruction of the same kind can begin to execute. A value of 1/3 indicates that the execution units can handle 3 instructions per clock cycle in one thread. However, the throughput may be limited by other bottlenecks in the pipeline.
<b>Execution unit:</b>	Indicates which execution unit is used for the macro-operations. ALU means any of the three integer ALU's. ALU0_1 means that ALU0 and ALU1 are both used. AGU means any of the three integer address generation units. FADD means floating point adder unit. FMUL means floating point multiplier unit. FMISC means floating point store and miscellaneous unit. FA/M means FADD or FMUL is used. FANY means any of the three floating point units can be used. Two macro-operations can execute simultaneously if they go to different execution units.

### Integer instructions

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution unit	Notes
<b>Move instructions</b>						
MOV	r,r	1	1	1/3	ALU	
MOV	r,i	1	1	1/3	ALU	
MOV	r8,m8	1	4	1/2	ALU, AGU	Any addressing mode. Add 1 clock if code segment base $\neq 0$
MOV	r16,m16	1	4	1/2	ALU, AGU	
MOV	r32,m32	1	3	1/2	AGU	
MOV	r64,m64	1	3	1/2	AGU	
MOV	m8,r8H	1	8	1/2	AGU	AH, BH, CH, DH
MOV	m8,r8L	1	3	1/2	AGU	Any other 8-bit register
MOV	m16/32/64,r	1	3	1/2	AGU	Any addressing mode
MOV	m,i	1	3	1/2	AGU	
MOV	m64,i32	1	3	1/2	AGU	
MOV	r,sr	1	3-4	1/2		
MOV	sr,r/m	6	8-26	8		from AMD manual

## K10

MOVNTI	m,r	1		1	AGU	
MOVZX, MOVSX	r,r	1	1	1/3	ALU	
MOVZX, MOVSX	r,m	1	4	1/2	ALU, AGU	
MOVSLD	r64,r32	1	1	1/3	ALU	
MOVSLD	r64,m32	1	4	1/2	ALU, AGU	
CMOVCc	r,r	1	1	1/3	ALU	
CMOVCc	r,m	1	4	1/2	ALU, AGU	
XCHG	r,r	2	1	1	ALU	
XCHG	r,m	2	21	19	ALU, AGU	Timing depends on hw
XLAT		2	5	5	ALU, AGU	
PUSH	r	1		1/2	ALU, AGU	
PUSH	i	1		1/2	ALU, AGU	
PUSH	m	2		1	ALU, AGU	
PUSH	sr	2		1	ALU, AGU	
PUSHF(D/Q)		9		3	ALU, AGU	
PUSHA(D)		9	6	6	ALU, AGU	
POP	r	1		1/2	ALU, AGU	
POP	m	3	3	1	ALU, AGU	
POP	DS/ES/FS/GS	6	10	8	ALU, AGU	
POP	SS	10	26	16	ALU, AGU	
POPF(D/Q)		28	16	11	ALU, AGU	
POPA(D)		9	6	6	ALU, AGU	
LEA	r16,[m]	2	3	1	ALU, AGU	Any address size ≤ 2 source operands
LEA	r32/64,[m]	1	1	1/3	ALU	W. scale or 3 opr.
LEA	r32/64,[m]	1	2	1/3	AGU	
LAHF		4	3	2	ALU	
SAHF		1	1	1/3	ALU	
SALC		1	1	1	ALU	
LDS, LES, ...	r,m	10		10		
BSWAP	r	1	1	1/3	ALU	
PREFETCHNTA	m	1		1/2	AGU	
PREFETCHT0/1/2	m	1		1/2	AGU	
SFENCE		6		8		
LFENCE		1		1		
MFENCE		4		33		
IN	r,i/DX	~270				
OUT	i/DX,r	~300				
<b>Arithmetic instructions</b>						
ADD, SUB	r,r/i	1	1	1/3	ALU	
ADD, SUB	r,m	1		1/2	ALU, AGU	
ADD, SUB	m,r	1	4	1	ALU, AGU	
ADC, SBB	r,r/i	1	1	1/3	ALU	
ADC, SBB	r,m	1		1/2	ALU, AGU	
ADC, SBB	m,r/i	1	4	1	ALU, AGU	
CMP	r,r/i	1	1	1/3	ALU	
CMP	r,m	1		1/2	ALU, AGU	
INC, DEC, NEG	r	1	1	1/3	ALU	
INC, DEC, NEG	m	1	7	2	ALU, AGU	

K10

AAA, AAS		9	5	5	ALU	
DAA		12	6	6	ALU	
DAS		16	7	7	ALU	
AAD		4	5	5	ALU0	
AAM		30	13	13	ALU	
MUL, IMUL	r8/m8	1	3	1	ALU0	
MUL, IMUL	r16/m16	3	3	2	ALU0_1	latency ax=3, dx=4
MUL, IMUL	r32/m32	2	3	1	ALU0_1	
MUL, IMUL	r64/m64	2	4	2	ALU0_1	latency rax=4, rdx=5
IMUL	r16,r16/m16	1	3	1	ALU0	
IMUL	r32,r32/m32	1	3	1	ALU0	
IMUL	r64,r64/m64	1	4	2	ALU0_1	
IMUL	r16,(r16),i	2	4	1	ALU0	
IMUL	r32,(r32),i	1	3	1	ALU0	
IMUL	r64,(r64),i	1	4	2	ALU0	
IMUL	r16,m16,i	3		2	ALU0	
IMUL	r32,m32,i	3		2	ALU0	
IMUL	r64,m64,i	3		2	ALU0_1	
DIV	r8/m8		17	17	ALU	
IDIV	r8		19	19	ALU	
IDIV	m8		22	22	ALU	
DIV	r16/m16		15-30	15-30	ALU	Depends on number
DIV	r32/m32		15-46	15-46	ALU	of significant bits in
DIV	r64/m64		15-78	15-78	ALU	absolute value of
IDIV	r16/m16		24-39	24-39	ALU	dividend. See AMD
IDIV	r32/m32		24-55	24-55	ALU	software optimiza-
IDIV	r64/m64		24-87	24-87	ALU	tion guide.
CBW, CWDE, CDQE		1	1	1/3	ALU	
CWD, CDQ, CQO		1	1	1/3	ALU	
<b>Logic instructions</b>						
AND, OR, XOR	r,r	1	1	1/3	ALU	
AND, OR, XOR	r,m	1		1/2	ALU, AGU	
AND, OR, XOR	m,r	1	4	1	ALU, AGU	
TEST	r,r	1	1	1/3	ALU	
TEST	r,m	1		1/2	ALU, AGU	
NOT	r	1	1	1/3	ALU	
NOT	m	1	7	1	ALU, AGU	
SHL, SHR, SAR	r,i/CL	1	1	1/3	ALU	
ROL, ROR	r,i/CL	1	1	1/3	ALU	
RCL, RCR	r,1	1	1	1	ALU	
RCL	r,i	9	3	3	ALU	
RCR	r,i	7	3	3	ALU	
RCL	r,CL	9	4	4	ALU	
RCR	r,CL	7	3	3	ALU	
SHL,SHR,SAR,ROL,RC	m,i /CL	1	7	1	ALU, AGU	
RCL, RCR	m,1	1	7	1	ALU, AGU	
RCL	m,i	10	7	5	ALU, AGU	
RCR	m,i	9	7	6	ALU, AGU	
RCL	m,CL	9	8	6	ALU, AGU	

## K10

RCR	m,CL	8	7	5	ALU, AGU	
SHLD, SHRD	r,r,i	6	3	2	ALU	
SHLD, SHRD	r,r,cl	7	3	3	ALU	
SHLD, SHRD	m,r,i/CL	8	7.5	6	ALU, AGU	
BT	r,r/i	1	1	1/3	ALU	
BT	m,i	1		1/2	ALU, AGU	
BT	m,r	5	7	2	ALU, AGU	
BTC, BTR, BTS	r,r/i	2	2	1/3	ALU	
BTC	m,i	5	9	1.5	ALU, AGU	
BTR, BTS	m,i	4	9	1.5	ALU, AGU	
BTC	m,r	8	8	10	ALU, AGU	
BTR, BTS	m,r	8	8	7	ALU, AGU	
BSF	r,r	6	4	3	ALU	
BSR	r,r	7	4	3	ALU	
BSF	r,m	7	7	3	ALU, AGU	
BSR	r,m	8	7	3	ALU, AGU	
POPCNT	r,r/m	1	2	1	ALU	SSE4.A / SSE4.2
LZCNT	r,r/m	1	2	1	ALU	SSE4.A, AMD only
SETcc	r	1	1	1/3	ALU	
SETcc	m	1		1/2	ALU, AGU	
CLC, STC		1		1/3	ALU	
CMC		1	1	1/3	ALU	
CLD		1		1/3	ALU	
STD		2		2/3	ALU	
<b>Control transfer instructions</b>						
JMP	short/near	1		2	ALU	
JMP	far	16-20	23-32			low values = real mode
JMP	r	1		2	ALU	
JMP	m(near)	1		2	ALU, AGU	
JMP	m(far)	17-21	25-33			low values = real mode
Jcc	short/near	1		1/3 - 2	ALU	recip. thrp.= 2 if jump
J(E/R)CXZ	short	2		2/3 - 2	ALU	recip. thrp.= 2 if jump
LOOP	short	7		3	ALU	
CALL	near	3	2	2	ALU	
CALL	far	16-22	23-32			low values = real mode
CALL	r	4	3	3	ALU	
CALL	m(near)	5	3	3	ALU, AGU	
CALL	m(far)	16-22	24-33			low values = real mode
RETN		2	3	3	ALU	
RETN	i	2	3	3	ALU	
RETF		15-23	24-35			low values = real mode
RETF	i	15-24	24-35			low values = real mode
IRET		32	81			real mode
INT	i	33	42			real mode
BOUND	m	6		2		values are for no jump
INTO		2		2		values are for no jump
<b>String instructions</b>						
LODS		4	2	2		

K10

REP LODS		5	2	2		values are per count
STOS		4	2	2		
REP STOS		2	1	1		values are per count
MOVS		7	3	3		
REP MOVS		3	1	1		values are per count
SCAS		5	2	2		
REP SCAS		5	2	2		values are per count
CMPS		7	3	3		
REP CMPS		3	1	1		values are per count
<b>Other</b>						
NOP (90)		1	0	1/3	ALU	
Long NOP (0F 1F)		1	0	1/3	ALU	
ENTER		i,0	12		12	
LEAVE		2		3		3 ops, 5 clk if 16 bit
CLI		8-9		5		
STI		16-17		27		
CPUID		22-50	47-164			
RDTSC		30		67		
RDPMC		13		5		

**Floating point x87 instructions**

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution unit	Notes
<b>Move instructions</b>						
FLD	r	1	2	1/2	FA/M	
FLD	m32/64	1	4	1/2	FANY	
FLD	m80	7	13	4		
FBLD	m80	20	94	30		
FST(P)	r	1	2	1/2	FA/M	
FST(P)	m32/64	1	2	1	FMISC	
FSTP	m80	10	8	7		
FBSTP	m80	218	167	163		
FXCH	r	1	0	1/3		
FILD	m	1	6	1	FMISC	
FIST(P)	m	1	4	1	FMISC	
FLDZ, FLD1		1		1	FMISC	
FCMOVcc	st0,r	9			FMISC, FA/M	Low latency immediately after FCOMI
FFREE	r	1		1/3	FANY	
FINCSTP, FDECSTP		1	0	1/3	FANY	
FNSTSW	AX	2		16	FMISC, ALU	Low latency immediately after FCOM FTST
FSTSW	AX	3		14	FMISC, ALU	do.
FNSTSW	m16	2		9	FMISC, ALU	do.
FNSTCW	m16	3		2	FMISC, ALU	
FLDCW	m16	12		14	FMISC, ALU	faster if unchanged
<b>Arithmetic instructions</b>						

K10

FADD(P),FSUB(R)(P)	r/m	1	4	1	FADD
FIADD,FISUB(R)	m	2		4	FADD,FMISC
FMUL(P)	r/m	1	4	1	FMUL
FIMUL	m	2		4	FMUL,FMISC
FDIV(R)(P)	r/m	1	?	24	FMUL
FIDIV(R)	m	2	31	24	FMUL,FMISC
FABS, FCHS		1	2	2	FMUL
FCOM(P), FUCOM(P)	r/m	1		1	FADD
FCOMPP, FUCOMPP		1		1	FADD
FCOMI(P)	r	1		1	FADD
FICOM(P)	m	2		1	FADD, FMISC
FTST		1		1	FADD
FXAM		2		1	FMISC, ALU
FRNDINT		6		37	
FPREM		1		7	FMUL
FPREM1		1		7	FMUL
<b>Math</b>					
FSQRT		1	35	35	FMUL
FLDPI, etc.		1		1	FMISC
FSIN		45	~51?		
FCOS		51	~90?		
FSINCOS		76	~125?		
FPTAN		45	~119		
FPATAN		9	151?	45?	
FSCALE		5	9	29	
EXTRACT		11	9	41	
F2XM1		8	65	30?	
FYL2X		8	13	30?	
FYL2XP1		12	114	44?	
<b>Other</b>					
FNOP		1	0	1/3	FANY
(F)WAIT		1	0	1/3	ALU
FNCLEX		8		28	FMISC
FNINIT		26		103	FMISC
FNSAVE	m	77	162	149	
FRSTOR	m	70	133	149	
FXSAVE	m	61	63	58	
FXRSTOR	m	85	89	79	

**Integer MMX and XMM instructions**

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution unit	Notes
<b>Move instructions</b>						
MOVD	r32, mm	1	3	1	FADD	
MOVD	mm, r32	2	6	3		
MOVD	mm,m32	1	4	1/2	FANY	
MOVD	r32, xmm	1	3	1	FADD	

## K10

MOVD	xmm, r32	2	6	3		
MOVD	xmm,m32	1	2	1/2		
MOVD	m32,mm/x	1	2	1	FMISC	
						Moves 64 bits.
MOVD (MOVQ)	r64,(x)mm	1	3	1	FADD	Name of instruction differs
MOVD (MOVQ)	mm,r64	2	6	3		do.
MOVD (MOVQ)	xmm,r64	2	6	3	FMUL, ALU	do.
MOVQ	mm,mm	1	2	1/2	FA/M	
MOVQ	xmm,xmm	1	2.5	1/3	FANY	
MOVQ	mm,m64	1	4	1/2	FANY	
MOVQ	xmm,m64	1	2	1/2	?	
MOVQ	m64,(x)mm	1	2	1	FMISC	
MOVDQA	xmm,xmm	1	2.5	1/3	FANY	
MOVDQA	xmm,m	1	2	1/2	?	
MOVDQA	m,xmm	2	2	1	FMUL,FMISC	
MOVDQU	xmm,m	1	2	1/2		
MOVDQU	m,xmm	3	3	2		
MOVDQ2Q	mm,xmm	1	2	1/3	FANY	
MOVQ2DQ	xmm,mm	1	2	1/3	FANY	
MOVNTQ	m,mm	1		1	FMISC	
MOVNTDQ	m,xmm	2		1	FMUL,FMISC	
PACKSSWB/DW						
PACKUSWB	mm,r/m	1	2	1/2	FA/M	
PACKSSWB/DW						
PACKUSWB	xmm,r/m	1	3	1/2	FA/M	
PUNPCKH/LBW/WD/DQ	mm,r/m	1	2	1/2	FA/M	
PUNPCKH/LBW/WD/DQ	xmm,r/m	1	3	1/2	FA/M	
PUNPCKHQDQ	xmm,r/m	1	3	1/2	FA/M	
PUNPCKLQDQ	xmm,r/m	1	3	1/2	FA/M	
PSHUFD	xmm,xmm,i	1	3	1/2	FA/M	
PSHUFW	mm,mm,i	1	2	1/2	FA/M	
PSHUFL/HW	xmm,xmm,i	1	2	1/2	FA/M	
MASKMOVQ	mm,mm	32		13		
MASKMOVDQU	xmm,xmm	64		24		
PMOVMASKB	r32,mm/xmm	1	3	1	FADD	
PEXTRW	r32,(x)mm,i	2	6	1		
PINSRW	(x)mm,r32,i	2	9	3	FA/M	
INSERTQ	xmm,xmm	3	6	2	FA/M	SSE4.A, AMD only
INSERTQ	xmm,xmm,i,i	3	6	2	FA/M	SSE4.A, AMD only
EXTRQ	xmm,xmm	1	2	1/2	FA/M	SSE4.A, AMD only
EXTRQ	xmm,xmm,i,i	1	2	1/2	FA/M	SSE4.A, AMD only
<b>Arithmetic instructions</b>						



PADDB/W/D/Q						
PADDSB/W						
PADDUSB/W						
PSUBB/W/D/Q						
PSUBSB/W						
PSUBUSB/W	mm/xmm,r/m	1	2	1/2	FA/M	
PCMPEQ/GT B/W/D	mm/xmm,r/m	1	2	1/2	FA/M	
PMULLW PMULHW						
PMULHUW						
PMULUDQ	mm/xmm,r/m	1	3	1	FMUL	
PMADDWD	mm/xmm,r/m	1	3	1	FMUL	
PAVGB/W	mm/xmm,r/m	1	2	1/2	FA/M	
PMIN/MAX SW/UB	mm/xmm,r/m	1	2	1/2	FA/M	
PSADBW	mm/xmm,r/m	1	3	1	FADD	
<b>Logic</b>						
PAND PANDN POR						
PXOR	mm/xmm,r/m	1	2	1/2	FA/M	
PSLL/RL W/D/Q						
PSRAW/D	mm,i/mm/m	1	2	1/2	FA/M	
PSLL/RL W/D/Q						
PSRAW/D	x,i(x)mm	1	3	1/2	FA/M	
PSLLDQ, PSRLDQ	xmm,i	1	3	1/2	FA/M	
<b>Other</b>						
EMMS		1		1/3	FANY	

### Floating point XMM instructions

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution unit	Notes
<b>Move instructions</b>						
MOVAPS/D	r,r	1	2.5	1/2	FANY	
MOVAPS/D	r,m	1	2	1/2	?	
MOVAPS/D	m,r	2	2	1	FMUL,FMISC	
MOVUPS/D	r,r	1	2.5	1/2	FANY	
MOVUPS/D	r,m	1	2	1/2	?	
MOVUPS/D	m,r	3	3	2	FMISC	
MOVSS/D	r,r	1	2	1/2	FA/M	
MOVSS/D	r,m	1	2	1/2	?	
MOVSS/D	m,r	1	2	1	FMISC	
MOVHPS, MOVLHPS	r,r	1	3	1/2	FA/M	
MOVHPS/D, MOVLPS/D	r,m	1	4	1/2	FA/M	
MOVHPS/D, MOVLPS/D	m,r	1		1	FMISC	
MOVNTPS/D	m,r	2		3	FMUL,FMISC	
MOVNTSS/D	m,r	1		1	FMISC	SSE4.A, AMD only
MOVMSKPS/D	r32,r	1	3	1	FADD	
SHUFPS/D	r,r/m,i	1	3	1/2	FA/M	

## K10

UNPCK H/L PS/D	r,r/m	1	3	1/2	FA/M
<b>Conversion</b>					
CVTPS2PD	r,r/m	1	2	1	FMISC
CVTPD2PS	r,r/m	2	7	1	
CVTSD2SS	r,r/m	3	8	2	
CVTSS2SD	r,r/m	3	7	2	
CVTDQ2PS	r,r/m	1	4	1	FMISC
CVTDQ2PD	r,r/m	1	4	1	FMISC
CVT(T)PS2DQ	r,r/m	1	4	1	FMISC
CVT(T)PD2DQ	r,r/m	2	7	1	
CVTPI2PS	xmm,mm	2	7	1	
CVTPI2PD	xmm,mm	1	4	1	FMISC
CVT(T)PS2PI	mm,xmm	1	4	1	FMISC
CVT(T)PD2PI	mm,xmm	2	7	1	
CVTSI2SS	xmm,r32	3	14	3	
CVTSI2SD	xmm,r32	3	14	3	
CVT(T)SD2SI	r32,xmm	2	8	1	FADD,FMISC
CVT(T)SS2SI	r32,xmm	2	8	1	FADD,FMISC
<b>Arithmetic</b>					
ADDSS/D SUBSS/D	r,r/m	1	4	1	FADD
ADDPS/D SUBPS/D	r,r/m	1	4	1	FADD
MULSS/D	r,r/m	1	4	1	FMUL
MULPS/D	r,r/m	1	4	1	FMUL
DIVSS	r,r/m	1	16	13	FMUL
DIVPS	r,r/m	1	18	15	FMUL
DIVSD	r,r/m	1	20	17	FMUL
DIVPD	r,r/m	1	20	17	FMUL
RCPSS RCPPS	r,r/m	1	3	1	FMUL
MAXSS/D MINSS/D	r,r/m	1	2	1	FADD
MAXPS/D MINPS/D	r,r/m	1	2	1	FADD
CMPccSS/D	r,r/m	1	2	1	FADD
CMPccPS/D	r,r/m	1	2	1	FADD
COMISS/D					
UCOMISS/D	r,r/m	1		1	FADD
<b>Logic</b>					
ANDPS/D ANDNPS/D					
ORPS/D XORPS/D	r,r/m	1	2	1/2	FA/M
<b>Math</b>					
SQRTSS	r,r/m	1	19	16	FMUL
SQRTPS	r,r/m	1	21	18	FMUL
SQRTSD	r,r/m	1	27	24	FMUL
SQRTPD	r,r/m	1	27	24	FMUL
RSQRTSS	r,r/m	1	3	1	FMUL
RSQRTPS	r,r/m	1	3	1	FMUL
<b>Other</b>					

K10

LDMXCSR	m	12	12	10	
STMXCSR	m	3	12	11	

**3DNow instructions (obsolete)**

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution unit	Notes
<b>Move and convert instructions</b>						
PREFETCH(W)	m	1		1/2	AGU	
PF2ID	mm,mm	1	5	1	FMISC	
PI2FD	mm,mm	1	5	1	FMISC	
PF2IW	mm,mm	1	5	1	FMISC	3DNow extension
PI2FW	mm,mm	1	5	1	FMISC	3DNow extension
PSWAPD	mm,mm	1	2	1/2	FA/M	3DNow extension
<b>Integer instructions</b>						
PAVGUSB	mm,mm	1	2	1/2	FA/M	
PMULHRW	mm,mm	1	3	1	FMUL	
<b>Floating point instructions</b>						
PFADD/SUB/SUBR	mm,mm	1	4	1	FADD	
PFCMPEQ/GE/GT	mm,mm	1	2	1	FADD	
PFMAX/MIN	mm,mm	1	2	1	FADD	
PFMUL	mm,mm	1	4	1	FMUL	
PFACC	mm,mm	1	4	1	FADD	
PFNACC, PFPNACC	mm,mm	1	4	1	FADD	3DNow extension
PFRCP	mm,mm	1	3	1	FMUL	
PFRCPIT1/2	mm,mm	1	4	1	FMUL	
PFRSQRT	mm,mm	1	3	1	FMUL	
PFRSQIT1	mm,mm	1	4	1	FMUL	
<b>Other</b>						
FEMMS	mm,mm	1		1/3	FANY	

Thank you to Xucheng Tang for doing the measurements on the K10.

## AMD Bulldozer

### List of instruction timings and macro-operation breakdown

#### Explanation of column headings:

<b>Instruction:</b>	Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc.
<b>Operands:</b>	i = immediate constant, r = any register, r32 = 32-bit register, etc., mm = 64 bit mmx register, x = 128 bit xmm register, y = 256 bit ymm register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc.
<b>Ops:</b>	Number of macro-operations issued from instruction decoder to schedulers. Instructions with more than 2 macro-operations use microcode.
<b>Latency:</b>	This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's, infinity and exceptions increase the delays. The latency listed does not include the memory operand where the listing for register and memory operand are joined (r/m).
<b>Reciprocal throughput:</b>	This is also called issue latency. This value indicates the average number of clock cycles from the execution of an instruction begins to a subsequent independent instruction of the same kind can begin to execute. A value of 1/3 indicates that the execution units can handle 3 instructions per clock cycle in one thread. However, the throughput may be limited by other bottlenecks in the pipeline.
<b>Execution pipe:</b>	Indicates which execution pipe or unit is used for the macro-operations: Integer pipes: EX0: integer ALU, division EX1: integer ALU, multiplication, jump EX01: can use either EX0 or EX1 AG01: address generation unit 0 or 1 Floating point and vector pipes: P0: floating point add, mul, div, convert, shuffle, shift P1: floating point add, mul, div, shuffle, shift P2: move, integer add, boolean P3: move, integer add, boolean, store P01: can use either P0 or P1 P23: can use either P2 or P3 Two macro-operations can execute simultaneously if they go to different execution pipes
<b>Domain:</b>	Tells which execution unit domain is used: ivec: integer vector execution unit. fp: floating point execution unit. fma: floating point multiply/add subunit. inherit: the output operand inherits the domain of the input operand. ivec/fma means the input goes to the ivec domain and the output comes from the fma domain. There is an additional latency of 1 clock cycle if the output of an ivec instruction goes to the input of a fp or fma instruction, and when the output of a fp or fma instruction goes to the input of an ivec or store instruction. There is no latency between the fp and fma units. All other latencies after memory load and before memory store instructions are included in the latency counts. An fma instruction has a latency of 5 if the output goes to another fma instruction, 6 if the output goes to an fp instruction, and 6+1 if the output goes to an ivec or store instruction.

Bulldozer

**Integer instructions**

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution pipes	Notes
<b>Move instructions</b>						
MOV	r,r	1	1	0.5	EX01	
MOV	r,i	1	1	0.5	EX01	
MOV	r,m	1	4	0.5	AG01	all addr. modes
MOV	m,r	1	4	1	EX01 AG01	all addr. modes
MOV	m,i	1		1		
MOVNTI	m,r	1	5	2		
MOVZX, MOVSX	r,r	1	1	0.5	EX01	
MOVSX	r,m	1	5	0.5	EX01	
MOVZX	r,m	1	4	0.5	EX01	
MOVSXD	r64,r32	1	1	0.5	EX01	
MOVSXD	r64,m32	1	5	0.5	EX01	
CMOVcc	r,r	1	1	0.5	EX01	
CMOVcc	r,m	1		0.5	EX01	
XCHG	r,r	2	1	1	EX01	
XCHG	r,m	2	~50	~50	EX01	Timing depends on hw
XLAT		2	6	2		
PUSH	r	1		1		
PUSH	i	1		1		
PUSH	m	2		1.5		
PUSHF(D/Q)		8		4		
PUSHA(D)		9		9		
POP	r	1		1		
POP	m	2		1		
POPF(D/Q)		34		19		
POPA(D)		14		8		
LEA	r16,[m]	2	2-3		EX01	any addr. size
LEA	r32,[m]	1	2-3		EX01	16 bit addr. size
LEA	r32/64,[m]	1	2	0.5	EX01	scale factor > 1 or 3 operands
LEA	r32/64,[m]	1	1	0.5	EX01	all other cases
LAHF		4	3	2		
SAHF		2	2	1		
SALC		1	1	1		
BSWAP	r	1	1	0.5	EX01	
PREFETCHNTA	m	1		0.5		
PREFETCHT0/1/2	m	1		0.5		
SFENCE		6		89		
LFENCE		1		0.25		
MFENCE		6		89		
<b>Arithmetic instructions</b>						
ADD, SUB	r,r	1	1	0.5	EX01	
ADD, SUB	r,i	1	1	0.5	EX01	
ADD, SUB	r,m	1		0.5	EX01	

Bulldozer

ADD, SUB	m,r	1	7-8	1	EX01
ADD, SUB	m,i	1	7-8	1	EX01
ADC, SBB	r,r	1	1		EX01
ADC, SBB	r,i	1	1		EX01
ADC, SBB	r,m	1	1	1	EX01
ADC, SBB	m,r	1	9	1	EX01
ADC, SBB	m,i	1	9	1	EX01
CMP	r,r	1	1	0.5	EX01
CMP	r,i	1	1	0.5	EX01
CMP	r,m	1		0.5	EX01
INC, DEC, NEG	r	1	1	0.5	EX01
INC, DEC, NEG	m	1	7-8	1	EX01
AAA, AAS		10	6		
DAA		16	9		
DAS		20	10		
AAD		4	6		
AAM		9	20	20	
MUL, IMUL	r8/m8	1	4	2	EX1
MUL, IMUL	r16/m16	2	4	2	EX1
MUL, IMUL	r32/m32	1	4	2	EX1
MUL, IMUL	r64/m64	1	6	4	EX1
IMUL	r16,r16/m16	1	4	2	EX1
IMUL	r32,r32/m32	1	4	2	EX1
IMUL	r64,r64/m64	1	6	4	EX1
IMUL	r16,(r16),i	2	5	2	EX1
IMUL	r32,(r32),i	1	4	2	EX1
IMUL	r64,(r64),i	1	6	4	EX1
IMUL	r16,m16,i	2		2	EX1
IMUL	r32,m32,i	2		2	EX1
IMUL	r64,m64,i	2		4	EX1
DIV	r8/m8	14	20	20	EX0
DIV	r16/m16	18	15-27	15-28	EX0
DIV	r32/m32	16	16-43	16-43	EX0
DIV	r64/m64	16	16-75	16-75	EX0
IDIV	r8/m8	33	23	20	EX0
IDIV	r16/m16	36	23-33	20-27	EX0
IDIV	r32/m32	36	22-48	20-43	EX0
IDIV	r64/m64	36	22-79	20-75	EX0
CBW, CWDE, CDQE		1	1		EX01
CDQ, CQO		1	1	0.5	EX01
CWD		2	1	1	EX01
<b>Logic instructions</b>					
AND, OR, XOR	r,r	1	1	0.5	EX01
AND, OR, XOR	r,i	1	1	0.5	EX01
AND, OR, XOR	r,m	1		0.5	EX01
AND, OR, XOR	m,r	1	7-8	1	EX01
AND, OR, XOR	m,i	1	7-8	1	EX01
TEST	r,r	1	1	0.5	EX01
TEST	r,i	1	1	0.5	EX01

Bulldozer

TEST	m,r	1		0.5	EX01	
TEST	m,i	1		0.5	EX01	
NOT	r	1		0.5	EX01	
NOT	m	1	7	1	EX01	
SHL, SHR, SAR	r,i/CL	1	1	0.5	EX01	
ROL, ROR	r,i/CL	1	1	0.5	EX01	
RCL	r,1	1	1		EX01	
RCL	r,i	16	8		EX01	
RCL	r,cl	17	9		EX01	
RCR	r,1	1	1		EX01	
RCR	r,i	15	8		EX01	
RCR	r,cl	16	8		EX01	
SHLD, SHRD	r,r,i	6	3	3	EX01	
SHLD, SHRD	r,r,cl	7	4	3.5	EX01	
SHLD, SHRD	m,r,i/CL	8		3.5	EX01	
BT	r,r/i	1	1	0.5	EX01	
BT	m,i	1		0.5	EX01	
BT	m,r	7		3.5	EX01	
BTC, BTR, BTS	r,r/i	2	2	1	EX01	
BTC, BTR, BTS	m,i	4		2	EX01	
BTC, BTR, BTS	m,r	10		5	EX01	
BSF	r,r	6	3	3	EX01	
BSF	r,m	8	4	4	EX01	
BSR	r,r	7	4	4	EX01	
BSR	r,m	9		5	EX01	
LZCNT	r,r	1	2	2	EX0	SSE4.A
POPCNT	r,r/m	1	4	2	EX1	SSE4.2
SETcc	r	1	1	0.5	EX01	
SETcc	m	1		1	EX01	
CLC, STC		1		0.5	EX01	
CMC		1	1		EX01	
CLD		2		3		
STD		2		4		
<b>Control transfer instructions</b>						
JMP	short/near	1		2	EX1	
JMP	r	1		2	EX1	
JMP	m	1		2	EX1	
Jcc	short/near	1		1-2	EX1	2 if jumping
fused CMP+Jcc	short/near	1		1-2	EX1	2 if jumping
J(E/R)CXZ	short	1		1-2	EX1	2 if jumping
LOOP	short	1		1-2	EX1	2 if jumping
LOOPE LOOPNE	short	1		1-2	EX1	2 if jumping
CALL	near	2		2	EX1	
CALL	r	2		2	EX1	
CALL	m	3		2	EX1	
RET		1		2	EX1	
RET	i	4		2-3	EX1	
BOUND	m	11		5		for no jump
INTO		4		24		for no jump

Bulldozer

<b>String instructions</b>						
LODS		3		3		
REP LODS		6n		3n		
STOS		3		3		
REP STOS		2n		2n		small n
REP STOS		3 per 16B		3 per 16B		best case
MOVS		5		3		
REP MOVS		2n		2n		small n
REP MOVS		4 per 16B		3 per 16B		best case
SCAS		3		3		
REP SCAS		7n		4n		
CMPS		6		3		
REP CMPS		9n		4n		
<b>Synchronization</b>						
LOCK ADD	m,r	1	~55			
XADD	m,r	4	10			
LOCK XADD	m,r	4	~51			
CMPXCHG	m8,r8	5	15			
LOCK CMPXCHG	m8,r8	5	~51			
CMPXCHG	m,r16/32/64	6	14			
LOCK CMPXCHG	m,r16/32/64	6	~52			
CMPXCHG8B	m64	18	15			
LOCK CMPXCHG8B	m64	18	~53			
CMPXCHG16B	m64	22	52			
LOCK CMPXCHG16B	m64	22	~94			
<b>Other</b>						
NOP (90)		1		0.25	none	
Long NOP (0F 1F)		1		0.25	none	
PAUSE		40		43		
ENTER	a,0	13		22		
ENTER	a,b	11+5b		16+4b		
LEAVE		2		4		
CPUID		37-63		112-280		
RDTSC		36		42		
RDPMSR		22		30		
CRC32	r32,r8	3	3	2		
CRC32	r32,r16	5	5	5		
CRC32	r32,r32	5	6	6		
XGETBV		4		31		

**Floating point x87 instructions**

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution pipes	Domain, notes
<b>Move instructions</b>						
FLD	r	1	2	0.5	P01	fp
FLD	m32/64	1	8	1		fp



Bulldozer

FLD	m80	8	14	4		fp
FBLD	m80	60	61	40	P0 P1 P2 P3	fp
FST(P)	r	1	2	0.5	P01	fp
FST(P)	m32/64	2	8	1		fp
FSTP	m80	13	9	20		fp
FBSTP	m80	239	240	244	P0 P1 F3	fp
FXCH	r	1	0	0.5	P01	inherit
FILD	m	1	12	1	F3	fp
FIST(P)	m	2	8	1	P0 F3	fp
FLDZ, FLD1		1		0.5	P01	fp
FCMOVcc	st0,r	8	3	3	P0 P1 F3	fp
FFREE	r	1		0.25	none	
FINCSTP, FDECSTP		1	0	0.25	none	inherit
FNSTSW	AX	4	~13	22	P0 P2 P3	
FNSTSW	m16	3	~13	19	P0 P2 P3	
FLDCW	m16	1		3		
FNSTCW	m16	3		2		
<b>Arithmetic instructions</b>						
FADD(P),FSUB(R)(P)	r/m	1	5-6	1	P01	fma
FIADD,FISUB(R)	m	2		2	P01	fma
FMUL(P)	r/m	1	5-6	1	P01	fma
FIMUL	m	2		2	P01	fma
FDIV(R)(P)	r	1	10-42	5-18	P01	fp
FDIV(R)	m	2			P01	fp
FIDIV(R)	m	2			P01	fp
FABS, FCHS		1	2	0.5	P01	fp
FCOM(P), FUCOM(P)	r/m	1		0.5	P01	fp
FCOMPP, FUCOMPP		1		0.5	P01	fp
FCOMI(P)	r	2	2	1	P0 P1 F3	fp
FICOM(P)	m	2		1	P01	fp
FTST		1		0.5	P01	fp
FXAM		1	~20	0.5	P01	fp
FRNDINT		1	4	1	P0	fp
FPREM		1	19-62		P0	fp
FPREM1		1	19-65		P0	fp
<b>Math</b>						
FSQRT		1	10-53		P01	
FLDPI, etc.		1		0.5	P01	
FSIN		10-162	65-210	65-210	P0 P1 P3	
FCOS		160-170	~160	~160	P0 P1 P3	
FSINCOS		12-166	95-160	95-160	P0 P1 P3	
FPTAN		11-190	95-245	95-245	P0 P1 P3	
FPATAN		10-355	60-440	60-440	P0 P1 P3	
FSCALE		8	52		P0 P1 P3	
FEXTRACT		12	10	5	P0 P1 P3	
F2XM1		10	64-71		P0 P1 P3	
FYL2X		10-175	60-290		P0 P1 P3	
FYL2XP1		10-175	60-320		P0 P1 P3	

Bulldozer

<b>Other</b>					
FNOP		1		0.25	none
(F)WAIT		1		0.25	none
FNCLEX		18		57	P0
FNINIT		31		170	P0
FNSAVE	m864	103	300	300	P0 P1 P2 P3
FRSTOR	m864	76	312	312	P0 P3

**Integer MMX and XMM instructions**

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution pipes	Notes
<b>Move instructions</b>						
MOVD	r32/64, mm/x	1	8	1		
MOVD	mm/x, r32/64	2	10	1		
MOVD	mm/x,m32	1	6	0.5		
MOVD	m32,mm/x	1	5	1		
MOVQ	mm/x,mm/x	1	2	0.5	P23	
MOVQ	mm/x,m64	1	6	0.5		
MOVQ	m64,mm/x	1	5	1	P3	
MOVDQA	xmm,xmm	1	0	0.25	none	inherit
MOVDQA	xmm,m	1	6	0.5		
MOVDQA	m,xmm	1	5	1	P3	
VMOVDQA	ymm,ymm	2	2	0.5	P23	
VMOVDQA	ymm,m256	2	6	1		
VMOVDQA	m256,ymm	4	5	3	P3	
MOVDQU	xmm,xmm	1	0	0.25	none	inherit
MOVDQU	xmm,m	1	6	0.5		
MOVDQU	m,xmm	1	5	1	P3	
LDDQU	xmm,m	1	6	0.5		
VMOVDQU	ymm,m256	2	6	1-2		
VMOVDQU	m256,ymm	8	6	10	P2 P3	
MOVDQ2Q	mm,xmm	1	2	0.5	P23	
MOVQ2DQ	xmm,mm	1	2	0.5	P23	
MOVNTQ	m,mm	1	6	2	P3	
MOVNTDQ	m,xmm	1	6	2	P3	
MOVNTDQA	xmm,m	1	6	0.5		
PACKSSWB/DW	(x)mm,r/m	1	2	1	P1	
PACKUSWB	(x)mm,r/m	1	2	1	P1	
PUNPCKH/LBW/WD/DQ	(x)mm,r/m	1	2	1	P1	
PUNPCKHQDQ	xmm,r/m	1	2	1	P1	
PUNPCKLQDQ	xmm,r/m	1	2	1	P1	
PSHUFB	(x)mm,r/m	1	3	1	P1	
PSHUFD	xmm,xmm,i	1	2	1	P1	
PSHUFW	mm,mm,i	1	2	1	P1	
PSHUFL/HW	xmm,xmm,i	1	2	1	P1	
PALIGNR	(x)mm,r/m,i	1	2	1	P1	
PBLENDW	xmm,r/m	1	2	0.5	P23	SSE4.1

Bulldozer

MASKMOVQ	mm,mm	31	38	37	P3	
MASKMOVDQU	xmm,xmm	64	48	61	P1 P3	
PMOVMASKB	r32,mm/x	2	10	1	P1 P3	
PEXTRB/W/D/Q	r,x/mm,i	2	10	1	P1 P3	AVX
PINSRB/W/D/Q	x/mm,r,i	2	12	2	P1	
PMOVSBW/BD/BQ/ WD/WQ/DQ	xmm,xmm	1	2	1	P1	SSE4.1
PMOVZXBW/BD/BQ/ WD/WQ/DQ	xmm,xmm	1	2	1	P1	SSE4.1
<b>Arithmetic instructions</b>						
PADDB/W/D/Q/SB/SW /USB/USW	(x)mm,r/m	1	2	0.5	P23	
PSUBB/W/D/Q/SB/SW /USB/USW	(x)mm,r/m	1	2	0.5	P23	
PCMPEQ/GT B/W/D	(x)mm,r/m	1	2	0.5	P23	
PMULLW PMULHW PMULHUW PMULUDQ	(x)mm,r/m	1	4	1	P0	
PMULLD	xmm,r/m	1	5	2	P0	SSE4.1
PMULDQ	xmm,r/m	1	4	1	P0	SSE4.1
PMULHRSW	(x)mm,r/m	1	4	1	P0	SSSE3
PMADDWD	(x)mm,r/m	1	4	1	P0	
PMADDUBSW	(x)mm,r/m	1	4	1	P0	
PAVGB/W	(x)mm,r/m	1	2	0.5	P23	
PMIN/MAX SB/SW/ SD UB/UW/UD	(x)mm,r/m	1	2	0.5	P23	
PHMINPOSUW	xmm,r/m	2	4	1	P1 P23	SSE4.1
PABSB/W/D	(x)mm,r/m	1	2	0.5	P23	SSSE3
PSIGNB/W/D	(x)mm,r/m	1	2	0.5	P23	SSSE3
PSADBW	(x)mm,r/m	2	4	1	P23	
MPSADBW	x,x,i	8	8	4	P1 P23	SSE4.1
<b>Logic</b>						
PAND PANDN POR PXOR	(x)mm,r/m	1	2	0.5	P23	
PSLL/RL W/D/Q PSRAW/D	(x)mm,r/m	1	3	1	P1	
PSLL/RL W/D/Q PSRAW/D	(x)mm,i	1	2	1	P1	
PSLLDQ, PSRLDQ	xmm,i	1	2	1	P1	
PTEST	xmm,r/m	2		1	P1 P3	SSE4.1
<b>String instructions</b>						
PCMPESTRI	x,x,i	27	17	10	P1 P2 P3	SSE4.2
PCMPESTRM	x,x,i	27	10	10	P1 P2 P3	SSE4.2
PCMPISTRI	x,x,i	7	14	3	P1 P2 P3	SSE4.2
PCMPISTRM	x,x,i	7	7	4	P1 P2 P3	SSE4.2
<b>Encryption</b>						
PCLMULQDQ	xmm,r/m	5	12	7	P1	pclmul

Bulldozer

AESDEC	x,x	2	5	2	P01	aes
AESDECLAST	x,x	2	5	2	P01	aes
AESENC	x,x	2	5	2	P01	aes
AESENCLAST	x,x	2	5	2	P01	aes
AESIMC	x,x	1	5	1	P0	aes
AESKEYGENASSIST	x,x,i	1	5	1	P0	aes
<b>Other</b>						
EMMS		1		0.25		

**Floating point XMM and YMM instructions**

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution pipes	Domain, notes
<b>Move instructions</b>						
MOVAPS/D						
MOVUPS/D	x,x	1	0	0.25	none	inherit
VMOVAPS/D	y,y	2	2	0.5	P23	ivec
MOVAPS/D						
MOVUPS/D	x,m128	1	6	0.5		
VMOVAPS/D						
VMOVUPS/D	y,m256	2	6	1-2		
MOVAPS/D						
MOVUPS/D	m128,x	1	5	1	P3	
VMOVAPS/D	m256,y	4	5	3	P3	
VMOVUPS/D	m256,y	8	6	10	P2 P3	
MOVSS/D	x,x	1	2	0.5	P01	fp
MOVSS/D	x,m32/64	1	6	0.5		
MOVSS/D	m32/64,x	1	5	1		
MOVHPS/D						
MOVLPS/D	x,m64	1	7	1		
MOVHPS/D	m64,x	2	8	1	P1 P3	
MOVLPS/D	m64,x	1	7	1	P3	
MOVLHPS MOVHLPS	x,x	1	2	1	P1	ivec
MOVMSKPS/D	r32,x	2	10	1	P1 P3	
VMOVMSKPS/D	r32,y					
MOVNTPS/D	m128,x	1	6	2	P3	
VMOVNTPS/D	m256,y					
SHUFPS/D	x,x/m,i	1	2	1	P1	ivec
VSHUFPS/D	y,y,y/m,i	2	2	2	P1	ivec
VPERMILPS/PD	x,x,x/m	1	3	1	P1	ivec
VPERMILPS/PD	y,y,y/m	2	3	2	P1	ivec
VPERMILPS/PD	x,x/m,i	1	2	1	P1	ivec
VPERMILPS/PD	y,y/m,i	2	2	2	P1	ivec
VPERM2F128	y,y,y,i	8	4	3	P23	ivec
VPERM2F128	y,y,m,i	10		4	P23	ivec
BLENDPS/PD	x,x/m,i	1	2	0.5	P23	ivec
VBLENDPS/PD	y,y,y/m,i	2	2	1	P23	ivec
BLENDVPS/PD	x,x/m,xmm0	1	2	1	P1	ivec
VBLENDVPS/PD	y,y,y/m,y	2	2	2	P1	ivec

Bulldozer

MOVDDUP	x,x	1	2	1	P1	ivec
MOVDDUP	x,m64	1		0.5		
VMOVDDUP	y,y	2	2	2	P1	ivec
VMOVDDUP	y,m256	2		1		
VBROADCASTSS	x,m32	1	6	0.5		
VBROADCASTSS	y,m32	2	6	0.5	P23	
VBROADCASTSD	y,m64	2	6	0.5	P23	
VBROADCASTF128	y,m128	2	6	0.5	P23	
MOVSH/LDUP	x,x	1	2	1	P1	ivec
MOVSH/LDUP	x,m128	1		0.5		
VMOVSH/LDUP	y,y	2	2	2	P1	ivec
VMOVSH/LDUP	y,m256	1		1		
UNPCKH/LPS/D	x,x/m	1	2	1	P1	ivec
VUNPCKH/LPS/D	y,y,y/m	2	2	2	P1	ivec
EXTRACTPS	r32,x,i	2	10	1	P1 P3	
EXTRACTPS	m32,x,i	2	14	1	P1 P3	
VEXTRACTF128	x,y,i	1	2	1	P23	ivec
VEXTRACTF128	m128,y,i	2	7	1	P23	
INSERTPS	x,x,i	1	2	1	P1	
INSERTPS	x,m32,i	1		1	P1	
VINSERTF128	y,y,x,i	2	2	1	P23	ivec
VINSERTF128	y,y,m128,i	2	9	1	P23	
VMASKMOVPS/D	x,x,m128	1	9	0.5	P01	
VMASKMOVPS/D	y,y,m256	2	9	1	P01	
VMASKMOVPS/D	m128,x,x	18	22	7	P0 P1 P2 P3	
VMASKMOVPS/D	m256,y,y	34	25	13	P0 P1 P2 P3	
<b>Conversion</b>						
CVTPD2PS	x,x	2	7	1	P01	fp
VCVTPD2PS	x,y	4	7	2	P01	fp
CVTPS2PD	x,x	2	7	1	P01	fp
VCVTPS2PD	y,x	4	7	2	P01	fp
CVTSD2SS	x,x	1	4	1	P0	fp
CVTSS2SD	x,x	1	4	1	P0	fp
CVTDQ2PS	x,x	1	4	1	P0	fp
VCVTDQ2PS	y,y	2	4	2	P0	fp
CVT(T) PS2DQ	x,x	1	4	1	P0	fp
VCVT(T) PS2DQ	y,y	2	4	2	P0	fp
CVTDQ2PD	x,x	2	7	1	P01	fp
VCVTDQ2PD	y,x	4	8	2	P01	fp
CVT(T)PD2DQ	x,x	2	7	1	P01	fp
VCVT(T)PD2DQ	x,y	4	7	2	P01	fp
CVTPI2PS	x,mm	1	4	1	P0	fp
CVT(T)PS2PI	mm,x	1	4	1	P0	fp
CVTPI2PD	x,mm	2	7	1	P0 P1	fp
CVT(T) PD2PI	mm,x	2	7	1	P0 P1	fp
CVTSI2SS	x,r32	2	14	1	P0	fp
CVT(T)SS2SI	r32,x	2	13	1	P0	fp
CVTSI2SD	x,r32/64	2	14	1	P0	fp
CVT(T)SD2SI	r32/64,x	2	13	1	P0	fp

Bulldozer

<b>Arithmetic</b>						
ADDSS/D SUBSS/D	x,x/m	1	5-6	0.5	P01	fma
ADDPSS/D SUBPSS/D	x,x/m	1	5-6	0.5	P01	fma
VADDPSS/D VSUBPSS/D	y,y,y/m	2	5-6	1	P01	fma
ADDSUBPSS/D	x,x/m	1	5-6	0.5	P01	fma
VADDSUBPSS/D	y,y,y/m	2	5-6	1	P01	fma
HADDPSS/D						
HSUBPSS/D	x,x	3	10	2	P01 P1	ivec/fma
HADDPSS/D						
HSUBPSS/D	x,m128	4		2	P01 P1	ivec/fma
VHADDPSS/D						
VHSUBPSS/D	y,y,y	8	10	4	P01 P1	ivec/fma
VHADDPSS/D						
VHSUBPSS/D	y,y,m	10		4	P01 P1	ivec/fma
MULSS MULSD	x,x/m	1	5-6	0.5	P01	fma
MULPSS MULPD	x,x/m	1	5-6	0.5	P01	fma
VMULPSS VMULPD	y,y,y/m	2	5-6	1	P01	fma
DIVSS DIVPSS	x,x/m	1	9-24	4.5-9.5	P01	fp
VDIVPSS	y,y,y/m	2	9-24	9-19	P01	fp
DIVSD DIVPD	x,x/m	1	9-27	4.5-11	P01	fp
VDIVPD	y,y,y/m	2	9-27	9-22	P01	fp
RCPSS/PS	x,x/m	1	5	1	P01	fp
VRCPPS	y,y/m	2	5	2	P01	fp
COMPSS/D						
COMPSS/D	x,x/m	1	2	0.5	P01	fp
VCOMPSS/D	y,y,y/m	2	2	1	P01	fp
COMISS/D						
UCOMISS/D	x,x/m	2		1	P01 P3	fp
MAXSS/SD/PS/PD						
MINSS/SD/PS/PD	x,x/m	1	2	0.5	P01	fp
VMAXPSS/D VMINPSS/D	y,y,y/m	2	2	1	P01	fp
ROUNDSS/SD/PS/PD	x,x,m,i	1	4	1	P0	fp
VROUNDSS/SD/PS/PD	y,y,m,i	2	4	2	P0	fp
DPSS	x,x,i	16	25	6	P01 P23	fma
DPSS	x,m128,i	18		7	P01 P23	fma
VDPSS	y,y,y,i	25	27	13	P01 P3	fma
VDPSS	y,m256,i	29		13	P01 P3	fma
DPPD	x,x,i	15	15	5	P01 P23	fma
DPPD	x,m128,i	17		6	P01 P23	fma
<b>Math</b>						
SQRTSS/PS	x,x/m	1	14-15	4.5-12	P01	fp
VSQRTPS	y,y/m	2	14-15	9-24	P01	fp
SQRTSD/PD	x,x/m	1	24-26	4.5-16.5	P01	fp
VSQRTPD	y,y/m	2	24-26	9-33	P01	fp
RSQRTSS/PS	x,x/m	1	5	1	P01	fp
VRSQRTPS	y,y/m	2	5	2	P01	fp

Bulldozer

Bulldozer						
<b>Logic</b>						
AND/ANDN/OR/XORPS/PD	x,x/m	1	2	0.5	P23	ivec
VAND/ANDN/OR/XORPS/PD	y,y,y/m	2	2	1	P23	ivec
<b>Other</b>						
VZEROUPPER		9		4		32 bit mode
VZEROUPPER		16		5		64 bit mode
VZEROALL		17		6	P2 P3	32 bit mode
VZEROALL		32		10	P2 P3	64 bit mode
LDMXCSR	m32	1	10	4	P0 P3	
STMXCSR	m32	2	19	19	P0 P3	
FXSAVE	m4096	67	136	136	P0 P1 P2 P3	
FXRSTOR	m4096	116	176	176	P0 P1 P2 P3	
XSAVE	m	122	196	196	P0 P1 P2 P3	
XRSTOR	m	177	250	250	P0 P1 P2 P3	

**AMD-specific instructions**

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution pipes	Notes
<b>3DNow instructions</b>						
PREFETCH/W	m	1		0.5		
<b>SSE4A instructions</b>						
LZCNT	r,r	2	2	2		
POPCNT	r16/32,r16/32	1	4	2		
POPCNT	r64,r64	1	4	4		
EXTRQ	x,i,i	1	3	1	P1	
EXTRQ	x,x	1	3	1	P1	
INSERTQ	x,x,i,i	1	3	1	P1	
INSERTQ	x,x	1	3	1	P1	
MOVNTSS/SD	m,x	1		4	P3	
<b>XOP instructions</b>						
VFRCZSS/SD/PS/PD	x,x	2	10	2	P01	
VFRCZSS/SD/PS/PD	x,m	3	10	2	P01	
VPCMOV	x,x,x,x/m	1	2	1	P1	
VPCMOV	y,y,y,y/m	2	2	2	P1	
VPPERM	x,x,x,x/m	1	2	1	P1	
VPCOMB/W/D/Q	x,x,x/m,i	1	2	0.5	P23	latency 0 if i=6, 7
VPCOMUB/W/D/Q	x,x,x/m,i	1	2	0.5	P23	latency 0 if i=6, 7
VPHADDBW/BD/BQ/WD/WQ/DQ	x,x/m	1	2	0.5	P23	
VPHADDUBW/BD/BQ/WD/WQ/DQ	x,x/m	1	2	0.5	P23	
VPHSUBBW/WD/DQ	x,x/m	1	2	0.5	P23	
VPMACSWW/WD	x,x,x/m,x	1	4	1	P0	

Bulldozer

VPMACSSDD	x,x,x/m,x	1	5	2	P0	
VPMACSSDQH/L	x,x,x/m,x	1	4	1	P0	
VPMACSSWW/WD	x,x,x/m,x	1	4	1	P0	
VPMACSSDD	x,x,x/m,x	1	5	2	P0	
VPMACSSDQH/L	x,x,x/m,x	1	4	1	P0	
VPMADCSWD	x,x,x/m,x	1	4	1	P0	
VPMADCSSWD	x,x,x/m,x	1	4	1	P0	
VPROTB/W/D/Q	x,x,x/m	1	3	1	P1	
VPROTB/W/D/Q	x,x,i	1	2	1	P1	
VPSHAB/W/D/Q	x,x,x/m	1	3	1	P1	
VPSHLB/W/D/Q	x,x,x/m	1	3	1	P1	
<b>FMA4 instructions</b>						
VFMASS/SD	x,x,x,x/m	1	5-6	0.5	P01	fma
VFMASSSPS/PD	x,x,x,x/m	1	5-6	0.5	P01	fma
VFMASSSPS/PD	y,y,y,y/m	2	5-6	1	P01	fma
VFMSUBSS/SD	x,x,x,x/m	1	5-6	0.5	P01	fma
VFMSUBSSPS/PD	x,x,x,x/m	1	5-6	0.5	P01	fma
VFMSUBSSPS/PD	y,y,y,y/m	2	5-6	1	P01	fma
VFMASSSUBPS/PD	x,x,x,x/m	1	5-6	0.5	P01	fma
VFMASSSUBPS/PD	y,y,y,y/m	2	5-6	1	P01	fma
VFMSUBADDPS/PD	x,x,x,x/m	1	5-6	0.5	P01	fma
VFMSUBADDPS/PD	y,y,y,y/m	2	5-6	1	P01	fma
VFNMADDSS/SD	x,x,x,x/m	1	5-6	0.5	P01	fma
VFNMADDSSPS/PD	x,x,x,x/m	1	5-6	0.5	P01	fma
VFNMADDSSPS/PD	y,y,y,y/m	2	5-6	1	P01	fma
VFNMSSUBSS/SD	x,x,x,x/m	1	5-6	0.5	P01	fma
VFNMSSUBSSPS/PD	x,x,x,x/m	1	5-6	0.5	P01	fma
VFNMSSUBSSPS/PD	y,y,y,y/m	2	5-6	1	P01	fma



## AMD Bobcat

### List of instruction timings and macro-operation breakdown

#### Explanation of column headings:

<b>Instruction:</b>	Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc.
<b>Operands:</b>	i = immediate constant, r = any register, r32 = 32-bit register, etc., mm = 64 bit mmx register, xmm = 128 bit xmm register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc.
<b>Ops:</b>	Number of micro-operations issued from instruction decoder to schedulers. Instructions with more than 2 micro-operations are micro-coded.
<b>Latency:</b>	This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's, infinity and exceptions increase the delays. The latencies listed do not include memory operands where the operand is listed as register or memory (r/m).  The clock frequency varies dynamically, which makes it difficult to measure latencies. The values listed are measured after the execution of millions of similar instructions, assuming that this will make the processor boost the clock frequency to the highest possible value.
<b>Reciprocal throughput:</b>	This is also called issue latency. This value indicates the average number of clock cycles from the execution of an instruction begins to a subsequent independent instruction of the same kind can begin to execute. A value of 1/2 indicates that the execution units can handle 2 instructions per clock cycle in one thread. However, the throughput may be limited by other bottlenecks in the pipeline.
<b>Execution pipe:</b>	Indicates which execution pipe is used for the micro-operations. I0 means integer pipe 0. I0/1 means integer pipe 0 or 1. FP0 means floating point pipe 0 (ADD). FP1 means floating point pipe 1 (MUL). FP0/1 means either one of the two floating point pipes. Two micro-operations can execute simultaneously if they go to different execution pipes.

#### Integer instructions

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution pipe	Notes
<b>Move instructions</b>						
MOV	r,r	1	1	1/2	I0/1	
MOV	r,i	1		1/2	I0/1	
MOV	r,m	1	4	1	AGU	Any addressing mode
MOV	m,r	1	4	1	AGU	Any addressing mode
MOV	m8,r8H	1	7	1	AGU	AH, BH, CH, DH
MOV	m,i	1		1	AGU	
MOVNTI	m,r	1	6	1	AGU	
MOVZX, MOVZX	r,r	1	1	1/2	I0/1	
MOVZX, MOVZX	r,m	1	5	1		
MOVSXD	r64,r32	1	1	1/2		
MOVSXD	r64,m32	1	5	1		
CMOVcc	r,r	1	1	1/2	I0/1	
CMOVcc	r,m	1		1		
XCHG	r,r	2	1	1	I0/1	

## Bobcat

XCHG	r,m	3	20			Timing depends on hw
XLAT		2	5			
PUSH	r	1		1		
PUSH	i	1		1		
PUSH	m	3		2		
PUSHF(D/Q)		9		6		
PUSHA(D)		9		9		
POP	r	1		1		
POP	m	4		4		
POPF(D/Q)		29		22		
POPA(D)		9		8		
LEA	r16,[m]	2	3	2	I0	Any address size
LEA	r32/64,[m]	1	1	1/2	I0/1	no scale, no offset
LEA	r32/64,[m]	1	2-4	1	I0	w. scale or offset
LEA	r64,[m]	1		1/2	I0/1	RIP relative
LAHF		4	4	2		
SAHF		1	1	1/2	I0/1	
SALC		1	1			
BSWAP	r	1	1	1/2	I0/1	
PREFETCHNTA	m	1		1	AGU	
PREFETCHT0/1/2	m	1		1	AGU	
PREFETCH	m	1		1	AGU	AMD only
SFENCE		4		~45	AGU	
LFENCE		1		1	AGU	
MFENCE		4		~45	AGU	
<b>Arithmetic instructions</b>						
ADD, SUB	r,r/i	1	1	1/2	I0/1	
ADD, SUB	r,m	1		1		
ADD, SUB	m,r	1		1		
ADC, SBB	r,r/i	1	1	1	I0/1	
ADC, SBB	r,m	1		1		
ADC, SBB	m,r/i	1	6-7			
CMP	r,r/i	1	1	1/2	I0/1	
CMP	r,m	1		1		
INC, DEC, NEG	r	1	1	1/2	I0/1	
INC, DEC, NEG	m	1	6			
AAA		9	5			
AAS		9	10			
DAA		12	7			
DAS		16	8			
AAD		4	5			
AAM		33	23	23		
MUL, IMUL	r8/m8	1	3	1	I0	
MUL, IMUL	r16/m16	3	3-5		I0	latency ax=3, dx=5
MUL, IMUL	r32/m32	2	3-4	2	I0	latency eax=3, edx=4
MUL, IMUL	r64/m64	2	6-7		I0	latency rax=6, rdx=7
IMUL	r16,r16/m16	1	3	1	I0	
IMUL	r32,r32/m32	1	3	1	I0	
IMUL	r64,r64/m64	1	6	4	I0	

## Bobcat

IMUL	r16,(r16),i	2	4	3	10
IMUL	r32,(r32),i	1	3	1	10
IMUL	r64,(r64),i	1	7	4	10
DIV	r8/m8	1	27	27	10
DIV	r16/m16	1	33	33	10
DIV	r32/m32	1	49	49	10
DIV	r64/m64	1	81	81	10
IDIV	r8/m8	1	29	29	10
IDIV	r16/m16	1	37	37	10
IDIV	r32/m32	1	55	55	10
IDIV	r64/m64	1	81	81	10
CBW, CWDE, CDQE		1	1		10/1
CWD, CDQ, CQO		1	1		10/1
<b>Logic instructions</b>					
AND, OR, XOR	r,r	1	1	1/2	10/1
AND, OR, XOR	r,m	1		1	
AND, OR, XOR	m,r	1		1	
TEST	r,r	1	1	1/2	10/1
TEST	r,m	1		1	
NOT	r	1	1	1/2	10/1
NOT	m	1		1	
SHL, SHR, SAR	r,i/CL	1	1	1/2	10/1
ROL, ROR	r,i/CL	1	1	1/2	10/1
RCL, RCR	r,1	1	1	1	10/1
RCL	r,i	9	5	5	
RCR	r,i	7	4	4	
RCL	r,CL	9	6	5	
RCR	r,CL	9	5	4	
SHL,SHR,SAR,ROL, ROR	m,i /CL	1	7	1	
RCL, RCR	m,1	1	7	1	
RCL	m,i	10		~15	
RCR	m,i	9	18	~14	
RCL	m,CL	9		15	
RCR	m,CL	8		15	
SHLD, SHRD	r,r,i	6	3	3	
SHLD, SHRD	r,r,cl	7	4	4	
SHLD, SHRD	m,r,i/CL	8	18	15	
BT	r,r/i	1		1/2	
BT	m,i	1		1	
BT	m,r	5		3	
BTC, BTR, BTS	r,r/i	2	2	1	
BTC	m,i	5		15	
BTR, BTS	m,i	4-5		15	
BTC	m,r	8	16	13	
BTR, BTS	m,r	8	15	15	
BSF, BSR	r,r	11	6	6	
BSF, BSR	r,m	11		6	
POPCNT	r,r/m	9	12	5	SSE4.A/SSE4.2

Bobcat

LZCNT	r,r/m	8	5			SSE4.A, AMD only
SETcc	r	1	1	1/2		
SETcc	m	1		1		
CLC, STC		1		1/2	I0/1	
CMC		1	1	1/2	I0/1	
CLD		1		1	I0	
STD		2		2	I0,I1	
<b>Control transfer instructions</b>						
JMP	short/near	1		2		
JMP	r	1		2		
JMP	m(near)	1		2		
Jcc	short/near	1		1/2 - 2		recip. thrp.= 2 if jump
J(E/R)CXZ	short	2		1 - 2		recip. thrp.= 2 if jump
LOOP	short	8		4		
CALL	near	2		2		
CALL	r	2		2		
CALL	m(near)	5		2		
RET		1		~3		
RET	i	4		~4		
BOUND	m	8		4		values are for no jump
INTO		4		2		values are for no jump
<b>String instructions</b>						
LODS		4		~3		
REP LODS		5		~3		values are per count
STOS		4		2		
REP STOS		2				best case 6-7 Byte/clock
MOVS		7		5		
REP MOVS		2				best case 5 Byte/clock
SCAS		5		3		
REP SCAS		6		3		values are per count
CMPS		7		4		
REP CMPS		6		3		values are per count
<b>Other</b>						
NOP (90)		1	0	1/2	I0/1	
Long NOP (0F 1F)		1	0	1/2	I0/1	
PAUSE		6		6		
ENTER	i,0		12		36	
ENTER	a,b		10+6b		34+6b	
LEAVE		2		3		32 bit mode
CPUID		30-52	70-830			
RDTSC		26		87		
RDPMSR		14		8		

**Floating point x87 instructions**

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution pipe	Notes
-------------	----------	-----	---------	-----------------------	----------------	-------

Bobcat

<b>Move instructions</b>					
FLD	r	1	2	1/2	FP0/1
FLD	m32/64	1	6	1	FP0/1
FLD	m80	7	14	5	
FBLD	m80	21	30	35	
FST(P)	r	1	2	1/2	FP0/1
FST(P)	m32/64	1	6	1	FP1
FSTP	m80	16	19	9	
FBSTP	m80	217	177	180	
FXCH	r	1	0	1	FP1
FILD	m	1	9	1	FP1
FIST(T)(P)	m	1	6	1	
FLDZ, FLD1		1		1	FP1
FCMOVcc	st0,r	12	7	7	FP0/1
FFREE	r	1		1	FP1
FINCSTP, FDECSTP		1	1	1	FP1
FNSTSW	AX	2	~20	10	FP1
FNSTSW	m16	2	~20	10	FP1
FNSTCW	m16	3		2	FP0
FLDCW	m16	12		10	FP1
<b>Arithmetic instructions</b>					
FADD(P),FSUB(R)(P)	r	1	3	1	FP0
FADD(P),FSUB(R)(P)	m	1	3	1	FP0
FIADD,FISUB(R)	m	2		3	FP0,FP1
FMUL(P)	r	1	5	3	FP1
FMUL(P)	m	1	5	3	FP1
FIMUL	m	2			FP1
FDIV(R)(P)	r	1	19	19	FP1
FDIV(R)(P)	m	1		19	FP1
FIDIV(R)	m	2		19	FP1
FABS, FCHS		1	2	2	FP1
FCOM(P), FUCOM(P)	r	1		1	FP0
FCOM(P), FUCOM(P)	m	1		1	FP0
FCOMPP, FUCOMPP		1		1	FP0
FCOMI(P)	r	1	2	2	FP0
FICOM(P)	m	2		1	FP0, FP1
FTST		1		1	FP0
FXAM		2		2	FP1
FRNDINT		5	11		FP0, FP1
FPREM		1	11-16		FP1
FPREM1		1	11-19		FP1
<b>Math</b>					
FSQRT		1	31		FP1
FLDPI, etc.		1		1	FP0
FSIN		4-44	27-105	27-105	FP0, FP1
FCOS		11-51	51-94	51-94	FP0, FP1
FSINCOS		11-75	48-110	48-110	FP0, FP1
FPTAN		~45	~113	~113	FP0, FP1

Bobcat

FPATAN		9-75	49-163	49-163	FP0, FP1
FSCALE		5	8		FP0, FP1
FXTRACT		7	9		FP0, FP1
F2XM1		30-56	~60		FP0, FP1
FYL2X		8	29		FP0, FP1
FYL2XP1		12	44		FP0, FP1
<b>Other</b>					
FNOP		1	0	1/2	FP0, FP1
(F)WAIT		1	0	1/2	ALU
FNCLEX		9		30	FP0, FP1
FNINIT		26		78	FP0, FP1
FNSAVE	m	85		163	FP0, FP1
FRSTOR	m	80		123	FP0, FP1
FXSAVE	m	71		105	FP0, FP1
FXRSTOR	m	111		118	FP0, FP1

**Integer MMX and XMM instructions**

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution pipe	Notes
<b>Move instructions</b>						
MOVD	r32, mm	1	7	1	FP0	
MOVD	mm, r32	1	7	3	FP0/1	
MOVD	mm,m32	1	5	1	FP0/1	
MOVD	r32, xmm	1	6	1	FP0	
MOVD	xmm, r32	3	6	3	FP1	
MOVD	xmm,m32	2	5	1	FP1	
MOVD	m32,(x)mm	1	6	2	FP1	
MOVD (MOVQ)	r64,(x)mm	1	7	1	FP0	Moves 64 bits.Name of instruction differs
MOVD (MOVQ)	mm,r64	2	7	3	FP0/1	do.
MOVD (MOVQ)	xmm,r64	3	7	3	FP0/1	do.
MOVQ	mm,mm	1	1	1/2	FP0/1	
MOVQ	xmm,xmm	2	1	1	FP0/1	
MOVQ	mm,m64	1	5	1	FP0/1	
MOVQ	xmm,m64	2	5	1	FP1	
MOVQ	m64,(x)mm	1	6	2	FP1	
MOVDQA	xmm,xmm	2	1	1	FP0/1	
MOVDQA	xmm,m	2	6	2	AGU	
MOVDQA	m,xmm	2	6	3	FP1	
MOVDQU, LDDQU	xmm,m	2	6-9	2-5.5	AGU	
MOVDQU	m,xmm	2	6-9	3-6	FP1	
MOVDQ2Q	mm,xmm	1	1	1/2	FP0/1	
MOVQ2DQ	xmm,mm	2	1	1	FP0/1	
MOVNTQ	m,mm	1	13	1.5	FP1	
MOVNTDQ	m,xmm	2	13	3	FP1	
PACKSSWB/DW						
PACKUSWB	mm,r/m	1	1	1/2	FP0/1	
PACKSSWB/DW						
PACKUSWB	xmm,r/m	3	2	2	FP0/1	

## Bobcat

PUNPCKH/LBW/WD/DQ	mm,r/m	1	1	1/2		
PUNPCKH/LBW/WD/DQ	xmm,r/m	2	1	1		
PUNPCKHQDQ	xmm,r/m	2	1	1	FP0, FP1	
PUNPCKLQDQ	xmm,r/m	1	1	1/2	FP0/1	
PSHUFB	mm,mm	1	2	1	FP0/1	Suppl. SSE3
PSHUFB	xmm,xmm	6	3	3	FP0/1	Suppl. SSE3
PSHUFD	xmm,xmm,i	3	2	2	FP0/1	
PSHUFW	mm,mm,i	1	1	1/2	FP0/1	
PSHUFL/HW	xmm,xmm,i	2	2	2	FP0/1	
PALIGNR	xmm,xmm,i	20	19	12	FP0/1	Suppl. SSE3
MASKMOVQ	mm,mm	32	146-1400	130-1170	FP0, FP1	
MASKMOVDQU	xmm,xmm	64	279-3000	260-2300	FP0, FP1	
PMOVMSKB	r32,(x)mm	1	8	2	FP0	
PEXTRW	r32,(x)mm,i	2	12	2	FP0, FP1	
PINSRW	mm,r32,i	2	10	6	FP0/1	
PINSRW	xmm,r32,i	3	10		FP0/1	
INSERTQ	xmm,xmm	3	3-4	3	FP0, FP1	SSE4.A, AMD only
INSERTQ	xmm,xmm,i,i	3	3-4	3	FP0, FP1	SSE4.A, AMD only
EXTRQ	xmm,xmm	1	1	1	FP0/1	SSE4.A, AMD only
EXTRQ	xmm,xmm,i,i	1	2	2	FP0/1	SSE4.A, AMD only
<b>Arithmetic instructions</b>						
PADDB/W/D/Q						
PADDSB/W						
PADDUSB/W						
PSUBB/W/D/Q						
PSUBSB/W						
PSUBUSB/W	mm,r/m	1	1	1/2	FP0/1	
PADDB/W/D/Q						
PADDSB/W						
ADDUSB/W						
PSUBB/W/D/Q						
PSUBSB/W						
PSUBUSB/W	xmm,r/m	2	1	1	FP0/1	
PHADD/SUBW/SW/D	mm,r/m	1	1	1/2	FP0/1	Suppl. SSE3
PHADD/SUBW/SW/D	xmm,r/m	2	4	1	FP0/1	Suppl. SSE3
PCMPEQ/GT B/W/D	mm,r/m	1	1	1/2	FP0/1	
PCMPEQ/GT B/W/D	xmm,r/m	2	1	1	FP0/1	
PMULLW PMULHW						
PMULHUW						
PMULUDQ	mm,r/m	1	2	1	FP0	
PMULLW PMULHW						
PMULHUW						
PMULUDQ	xmm,r/m	2	2	2	FP0	
PMULHRSW	mm,r/m	1	2	1	FP0	Suppl. SSE3
PMULHRSW	xmm,r/m	2	2	2	FP0	Suppl. SSE3
PMADDWD	mm,r/m	1	2	1	FP0	
PMADDWD	xmm,r/m	2	2	2	FP0	
PMADDUBSW	mm,r/m	1	2	1	FP0	Suppl. SSE3
PMADDUBSW	xmm,r/m	2	2	2	FP0	Suppl. SSE3

Bobcat

PAVGB/W	mm,r/m	1	1	1/2	FP0/1	
PAVGB/W	xmm,r/m	2	1	1	FP0/1	
PMIN/MAX SW/UB	mm,r/m	1	1	1/2	FP0/1	
PMIN/MAX SW/UB	xmm,r/m	2	1	1	FP0/1	
PABSB/W/D	mm,r/m	1	1	1/2	FP0/1	Suppl. SSE3
PABSB/W/D	xmm,r/m	2	1	1	FP0/1	Suppl. SSE3
PSIGNB/W/D	mm,r/m	1	1	1/2	FP0/1	Suppl. SSE3
PSIGNB/W/D	xmm,r/m	2	1	1	FP0/1	Suppl. SSE3
PSADBW	mm,r/m	1	2	2	FP0	
PSADBW	xmm,r/m	2	2	2	FP0, FP1	
<b>Logic</b>						
PAND PANDN POR PXOR	mm,r/m	1	1	1/2	FP0/1	
PAND PANDN POR PXOR	xmm,r/m	2	1	1	FP0/1	
PSLL/RL W/D/Q PSRAW/D	mm,i/mm/m	1	1	1	FP0/1	
PSLL/RL W/D/Q PSRAW/D	xmm,i/xmm/m	2	1	1	FP0/1	
PSLLDQ, PSRLDQ	xmm,i	2	1	1	FP0/1	
<b>Other</b>						
EMMS		1		1/2	FP0/1	

**Floating point XMM instructions**

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution pipe	Notes
<b>Move instructions</b>						
MOVAPS/D	r,r	2	1	1	FP0/1	
MOVAPS/D	r,m	2	6	2	AGU	
MOVAPS/D	m,r	2	6	3	FP1	
MOVUPS/D	r,r	2	1	1	FP0/1	
MOVUPS/D	r,m	2	6-9	2-6	AGU	
MOVUPS/D	m,r	2	6-9	3-6	FP1	
MOVSS/D	r,r	1	1	1/2	FP0/1	
MOVSS/D	r,m	2	6	2	FP1	
MOVSS/D	m,r	1	5	2	FP1	
MOVHLPS, MOVLHPS	r,r	1	1	1/2	FP0/1	
MOVHPS/D, MOVLPS/D	r,m	1	6	2	AGU	
MOVHPS/D, MOVLPS/D	m,r	1	5	3	FP1	
MOVNTPS/D	m,r	2	12	3	FP1	
MOVNTPS/D	m,r	1	12	2	FP1	SSE4.A, AMD only
MOVDDUP	r,r	2	2	1	FP0/1	SSE3
MOVDDUP	r,m64	2	7	2	FP0/1	SSE3
MOVSHDUP, MOVSLDUP	r,r	2	1	1	FP0/1	



Bobcat

MOVSHDUP, MOVSLDUP	r,m	2	12	3	AGU	
MOVMSKPS/D	r32,r	1	~6	2	FP0	
SHUFPS/D	r,r/m,i	3	2	2	FP0/1	
UNPCK H/L PS/D	r,r/m	2	1	1	FP0/1	
<b>Conversion</b>						
CVTSP2PD	r,r/m	2	5	2	FP1	
CVTPD2PS	r,r/m	4	5	3	FP0, FP1	
CVTSD2SS	r,r/m	3	5	3	FP0, FP1	
CVTSS2SD	r,r/m	1	4	1	FP1	
CVTDQ2PS	r,r/m	2	4	4	FP1	
CVTDQ2PD	r,r/m	2	5	2	FP1	
CVT(T)PS2DQ	r,r/m	2	4	4	FP1	
CVT(T)PD2DQ	r,r/m	4	6	3	FP0, FP1	
CVTPI2PS	xmm,mm	1	4	2	FP1	
CVTPI2PD	xmm,mm	2	5	2	FP1	
CVT(T)PS2PI	mm,xmm	1	4	1	FP1	
CVT(T)PD2PI	mm,xmm	3	6	2	FP0, FP1	
CVTSI2SS	xmm,r32	3	12	3	FP0, FP1	
CVTSI2SD	xmm,r32	2	11	3	FP1	
CVT(T)SS2SI	r32,xmm	2	12	1	FP0, FP1	
CVT(T)SD2SI	r32,xmm	2	11	1	FP0, FP1	
<b>Arithmetic</b>						
ADDSS/D SUBSS/D	r,r/m	1	3	1	FP0	
ADDPS/D SUBPS/D	r,r/m	2	3	2	FP0	
ADDSUBPS/D	r,r/m	2	3	2	FP0	SSE3
HADDPS/D						
HSUBPS/D	r,r/m	2	3	2	FP0	SSE3
MULSS	r,r/m	1	2	1	FP1	
MULSD	r,r/m	1	4	2	FP1	
MULPS	r,r/m	2	2	2	FP1	
MULPD	r,r/m	2	4	4	FP1	
DIVSS	r,r/m	1	13	13	FP1	
DIVPS	r,r/m	2	38	38	FP1	
DIVSD	r,r/m	1	17	17	FP1	
DIVPD	r,r/m	2	34	34	FP1	
RCPSS	r,r/m	1	3	1	FP1	
RCPPS	r,r/m	2	3	2	FP1	
MAXSS/D MINSS/D	r,r/m	1	2	1	FP0	
MAXPS/D MINPS/D	r,r/m	2	2	2	FP0	
CMPccSS/D	r,r/m	1	2	1	FP0	
CMPccPS/D	r,r/m	2	2	2	FP0	
COMISS/D						
UCOMISS/D	r,r/m	1		1	FP0	
<b>Logic</b>						
ANDPS/D ANDNPS/D						
ORPS/D XORPS/D	r,r/m	2	1	1	FP0/1	

Bobcat

<b>Math</b>						
SQRTSS	r,r/m	1	14	14	FP1	
SQRTPS	r,r/m	2	48	48	FP1	
SQRTSD	r,r/m	1	24	24	FP1	
SQRTPD	r,r/m	2	48	48	FP1	
RSQRTSS	r,r/m	1	3	1	FP1	
RSQRTPS	r,r/m	2	3	2	FP1	
<b>Other</b>						
LDMXCSR	m	12		10	FP0, FP1	
STMXCSR	m	3		11	FP0, FP1	

**Intel Pentium and Pentium MMX****List of instruction timings***Explanation of column headings:*

**Operands** r = register, accum = al, ax or eax, m = memory, i = immediate data, sr = segment register, m32 = 32 bit memory operand, etc.

**Clock cycles** The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably.

**Pairability** u = pairable in u-pipe, v = pairable in v-pipe, uv = pairable in either pipe, np = not pairable.

**Integer instructions (Pentium and Pentium MMX)**

Instruction	Operands	Clock cycles	Pairability
NOP		1	uv
MOV	r/m, r/m/i	1	uv
MOV	r/m, sr	1	np
MOV	sr, r/m	>= 2 b)	np
MOV	m, accum	1	uv h)
XCHG	(E)AX, r	2	np
XCHG	r, r	3	np
XCHG	r, m	>15	np
XLAT		4	np
PUSH	r/i	1	uv
POP	r	1	uv
PUSH	m	2	np
POP	m	3	np
PUSH	sr	1 b)	np
POP	sr	>= 3 b)	np
PUSHF		3-5	np
POPF		4-6	np
PUSHA POPA		5-9 i)	np
PUSHAD POPAD		5	np
LAHF SAHF		2	np
MOVSX MOVZX	r, r/m	3 a)	np
LEA	r, m	1	uv
LDS LES LFS LGS LSS	m	4 c)	np
ADD SUB AND OR XOR	r, r/i	1	uv
ADD SUB AND OR XOR	r, m	2	uv
ADD SUB AND OR XOR	m, r/i	3	uv
ADC SBB	r, r/i	1	u
ADC SBB	r, m	2	u
ADC SBB	m, r/i	3	u
CMP	r, r/i	1	uv
CMP	m, r/i	2	uv
TEST	r, r	1	uv
TEST	m, r	2	uv
TEST	r, i	1	f)

## Intel Pentium

TEST	m, i	2	np
INC DEC	r	1	uv
INC DEC	m	3	uv
NEG NOT	r/m	1/3	np
MUL IMUL	r8/r16/m8/m16	11	np
MUL IMUL	all other versions	9 d)	np
DIV	r8/m8	17	np
DIV	r16/m16	25	np
DIV	r32/m32	41	np
IDIV	r8/m8	22	np
IDIV	r16/m16	30	np
IDIV	r32/m32	46	np
CBW CWDE		3	np
CWD CDQ		2	np
SHR SHL SAR SAL	r, i	1	u
SHR SHL SAR SAL	m, i	3	u
SHR SHL SAR SAL	r/m, CL	4/5	np
ROR ROL RCR RCL	r/m, 1	1/3	u
ROR ROL	r/m, i(><1)	1/3	np
ROR ROL	r/m, CL	4/5	np
RCR RCL	r/m, i(><1)	8/10	np
RCR RCL	r/m, CL	7/9	np
SHLD SHRD	r, i/CL	4 a)	np
SHLD SHRD	m, i/CL	5 a)	np
BT	r, r/i	4 a)	np
BT	m, i	4 a)	np
BT	m, i	9 a)	np
BTR BTS BTC	r, r/i	7 a)	np
BTR BTS BTC	m, i	8 a)	np
BTR BTS BTC	m, r	14 a)	np
BSF BSR	r, r/m	7-73 a)	np
SETcc	r/m	1/2 a)	np
JMP CALL	short/near	1 e)	v
JMP CALL	far	>= 3 e)	np
conditional jump	short/near	1/4/5/6 e)	v
CALL JMP	r/m	2/5 e	np
RETN		2/5 e	np
RETN	i	3/6 e)	np
RETF		4/7 e)	np
RETF	i	5/8 e)	np
J(E)CXZ	short	4-11 e)	np
LOOP	short	5-10 e)	np
BOUND	r, m	8	np
CLC STC CMC CLD STD		2	np
CLI STI		6-9	np
LODS		2	np
REP LODS		7+3*n g)	np
STOS		3	np
REP STOS		10+n g)	np
MOVS		4	np

Intel Pentium

REP MOVSB		12+n g)	np
SCASB		4	np
REP(N)E SCASB		9+4*n g)	np
CMPSB		5	np
REP(N)E CMPSB		8+4*n g)	np
BSWAP	r	1 a)	np
CPUID		13-16 a)	np
RDTSC		6-13 a) j)	np

**Notes:**

- a This instruction has a 0FH prefix which takes one clock cycle extra to decode on a P1 unless preceded by a multi-cycle instruction.
- b versions with FS and GS have a 0FH prefix. see note a.
- c versions with SS, FS, and GS have a 0FH prefix. see note a.
- d versions with two operands and no immediate have a 0FH prefix, see note a.
- e high values are for mispredicted jumps/branches.
- f only pairable if register is AL, AX or EAX.
- g add one clock cycle for decoding the repeat prefix unless preceded by a multi-cycle instruction (such as CLD).
- h pairs as if it were writing to the accumulator.
- i 9 if SP divisible by 4 (imperfect pairing).
- j on P1: 6 in privileged or real mode; 11 in non-privileged; error in virtual mode. On PMMX: 8 and 13 clocks respectively.

**Floating point instructions (Pentium and Pentium MMX)**

*Explanation of column headings*

- Operands** r = register, m = memory, m32 = 32-bit memory operand, etc.
- Clock cycles** The numbers are minimum values. Cache misses, misalignment, denormal operands, and exceptions may increase the clock counts considerably.
- Pairability** + = pairable with FXCH, np = not pairable with FXCH.
- i-ov** Overlap with integer instructions. i-ov = 4 means that the last four clock cycles can overlap with subsequent integer instructions.
- fp-ov** Overlap with floating point instructions. fp-ov = 2 means that the last two clock cycles can overlap with subsequent floating point instructions. (WAIT is considered a floating point instruction here)

Instruction	Operand	Clock cycles	Pairability	i-ov	fp-ov
FLD	r/m32/m64	1	0	0	0
FLD	m80	3	np	0	0
FBLD	m80	48-58	np	0	0
FST(P)	r	1	np	0	0
FST(P)	m32/m64	2 m)	np	0	0
FST(P)	m80	3 m)	np	0	0
FBSTP	m80	148-154	np	0	0
FILD	m	3	np	2	2

Intel Pentium

FIST(P)	m	6	np	0	0
FLDZ FLD1		2	np	0	0
FLDPI FLDL2E etc.		5 s)	np	2	2
FNSTSW	AX/m16	6 q)	np	0	0
FLDCW	m16	8	np	0	0
FNSTCW	m16	2	np	0	0
FADD(P)	r/m	3	0	2	2
FSUB(R)(P)	r/m	3	0	2	2
FMUL(P)	r/m	3	0	2	2 n)
FDIV(R)(P)	r/m	19/33/39 p)	0	38 o)	2
FCHS FABS		1	0	0	0
FCOM(P)(P) FUCOM	r/m	1	0	0	0
FIADD FISUB(R)	m	6	np	2	2
FIMUL	m	6	np	2	2
FIDIV(R)	m	22/36/42 p)	np	38 o)	2
FICOM	m	4	np	0	0
FTST		1	np	0	0
FXAM		17-21	np	4	0
FPREM		16-64	np	2	2
FPREM1		20-70	np	2	2
FRNDINT		9-20	np	0	0
FSCALE		20-32	np	5	0
FEXTRACT		12-66	np	0	0
FSQRT		70	np	69 o)	2
FSIN FCOS		65-100 r)	np	2	2
FSINCOS		89-112 r)	np	2	2
F2XM1		53-59 r)	np	2	2
FYL2X		103 r)	np	2	2
FYL2XP1		105 r)	np	2	2
FPTAN		120-147 r)	np	36 o)	0
FPATAN		112-134 r)	np	2	2
FNOP		1	np	0	0
FXCH	r	1	np	0	0
FINCSTP FDECSTP		2	np	0	0
FFREE	r	2	np	0	0
FNCLEX		6-9	np	0	0
FNINIT		12-22	np	0	0
FNSAVE	m	124-300	np	0	0
FRSTOR	m	70-95	np	0	0
WAIT		1	np	0	0

**Notes:**

- m The value to store is needed one clock cycle in advance.
- n 1 if the overlapping instruction is also an FMUL.
- o Cannot overlap integer multiplication instructions.
- p FDIV takes 19, 33, or 39 clock cycles for 24, 53, and 64 bit precision respectively. FIDIV takes 3 clocks more. The precision is defined by bit 8-9 of the floating point control word.
- q The first 4 clock cycles can overlap with preceding integer instructions.
- r Clock counts are typical. Trivial cases may be faster, extreme cases may be slower.

s May be up to 3 clocks more when output needed for FST, FCHS, or FABS.

### **MMX instructions (Pentium MMX)**

A list of MMX instruction timings is not needed because they all take one clock cycle, except the MMX multiply instructions which take 3. MMX multiply instructions can be pipelined to yield a throughput of one multiplication per clock cycle.

The EMMS instruction takes only one clock cycle, but the first floating point instruction after an EMMS takes approximately 58 clocks extra, and the first MMX instruction after a floating point instruction takes approximately 38 clocks extra. There is no penalty for an MMX instruction after EMMS on the PMMX.

There is no penalty for using a memory operand in an MMX instruction because the MMX arithmetic unit is one step later in the pipeline than the load unit. But the penalty comes when you store data from an MMX register to memory or to a 32-bit register: The data have to be ready one clock cycle in advance. This is analogous to the floating point store instructions.

All MMX instructions except EMMS are pairable in either pipe. Pairing rules for MMX instructions are described in manual 3: "The microarchitecture of Intel, AMD and VIA CPUs".

## **Intel Pentium II and Pentium III**

### List of instruction timings and $\mu$ op breakdown

#### Explanation of column headings:

- Operands:** i = immediate data, r = register, mm = 64 bit mmx register, xmm = 128 bit xmm register, sr = segment register, m = memory, m32 = 32-bit memory operand, etc.
- $\mu$ ops:** The number of  $\mu$ ops that the instruction generates for each execution port.
- p0:** Port 0: ALU, etc.
- p1:** Port 1: ALU, jumps
- p01:** Instructions that can go to either port 0 or 1, whichever is vacant first.
- p2:** Port 2: load data, etc.
- p3:** Port 3: address generation for store
- p4:** Port 4: store data
- Latency:** This is the delay that the instruction generates in a dependency chain. (This is not the same as the time spent in the execution unit. Values may be inaccurate in situations where they cannot be measured exactly, especially with memory operands). The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays by 50-150 clocks, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay.
- Reciprocal throughput:** The average number of clock cycles per instruction for a series of independent instructions of the same kind.

### Integer instructions (Pentium Pro, Pentium II and Pentium III)

Instruction	Operands	$\mu$ ops						Latency	Reciprocal throughput
		p0	p1	p01	p2	p3	p4		
MOV	r,r/i			1					
MOV	r,m				1				
MOV	m,r/i					1	1		
MOV	r,sr			1					
MOV	m,sr			1		1	1		
MOV	sr,r	8						5	
MOV	sr,m	7			1			8	
MOVSX MOVZX	r,r			1					
MOVSX MOVZX	r,m				1				
CMOVcc	r,r	1		1					
CMOVcc	r,m	1		1	1				
XCHG	r,r			3					
XCHG	r,m			4	1	1	1	high b)	
XLAT				1	1				
PUSH	r/i			1		1	1		
POP	r			1	1				
POP	(E)SP			2	1				
PUSH	m			1	1	1	1		
POP	m			5	1	1	1		
PUSH	sr			2		1	1		
POP	sr			8	1				



## Pentium II and III

PUSHF(D)		3		11		1	1		
POPF(D)		10		6	1				
PUSHA(D)				2		8	8		
POPA(D)				2	8				
LAHF SAHF				1					
LEA	r,m	1						1 c)	
LDS LES LFS LGS									
LSS	m			8	3				
ADD SUB AND OR XOR	r,r/i			1					
ADD SUB AND OR XOR	r,m			1	1				
ADD SUB AND OR XOR	m,r/i			1	1	1	1		
ADC SBB	r,r/i			2					
ADC SBB	r,m			2	1				
ADC SBB	m,r/i			3	1	1	1		
CMP TEST	r,r/i			1					
CMP TEST	m,r/i			1	1				
INC DEC NEG NOT	r			1					
INC DEC NEG NOT	m			1	1	1	1		
AAA AAS DAA DAS			1						
AAD		1		2				4	
AAM		1	1	2				15	
IMUL	r,(r),(i)	1						4	1
IMUL	(r),m	1			1			4	1
DIV IDIV	r8	2		1				19	12
DIV IDIV	r16	3		1				23	21
DIV IDIV	r32	3		1				39	37
DIV IDIV	m8	2		1	1			19	12
DIV IDIV	m16	2		1	1			23	21
DIV IDIV	m32	2		1	1			39	37
CBW CWDE				1					
CWD CDQ		1							
SHR SHL SAR ROR									
ROL	r,i/CL	1							
SHR SHL SAR ROR									
ROL	m,i/CL	1			1	1	1		
RCR RCL	r,1	1		1					
RCR RCL	r8,i/CL	4		4					
RCR RCL	r16/32,i/CL	3		3					
RCR RCL	m,1	1		2	1	1	1		
RCR RCL	m8,i/CL	4		3	1	1	1		
RCR RCL	m16/32,i/CL	4		2	1	1	1		
SHLD SHRD	r,r,i/CL	2							
SHLD SHRD	m,r,i/CL	2		1	1	1	1		
BT	r,r/i			1					
BT	m,r/i	1		6	1				
BTR BTS BTC	r,r/i			1					
BTR BTS BTC	m,r/i	1		6	1	1	1		
BSF BSR	r,r		1	1					
BSF BSR	r,m		1	1	1				
SETcc	r			1					

Pentium II and III

SETcc	m			1		1	1		
JMP	short/near		1						2
JMP	far	21			1				
JMP	r		1						2
JMP	m(near)		1		1				2
JMP	m(far)	21			2				
conditional jump	short/near		1						2
CALL	near		1	1		1	1		2
CALL	far	28			1	2	2		
CALL	r		1	2		1	1		2
CALL	m(near)		1	4	1	1	1		2
CALL	m(far)	28			2	2	2		
RETN			1	2	1				2
RETN	i		1	3	1				2
RETF		23			3				
RETF	i	23			3				
J(E)CXZ	short		1	1					
LOOP	short	2	1	8					
LOOP(N)E	short	2	1	8					
ENTER	i,0			12		1	1		
ENTER	a,b	ca.	18	+4b		b-1	2b		
LEAVE				2	1				
BOUND	r,m	7		6	2				
CLC STC CMC				1					
CLD STD				4					
CLI		9							
STI		17							
INTO				5					
LODS					2				
REP LODS			10+6n						
STOS					1	1	1		
REP STOS			ca. 5n		a)				
MOVS				1	3	1	1		
REP MOVS			ca. 6n		a)				
SCAS				1	2				
REP(N)E SCAS			12+7n						
CMPS				4	2				
REP(N)E CMPS			12+9n						
BSWAP	r	1		1					
NOP (90)				1					0.5
Long NOP (0F 1F)				1					1
CPUID		23-48							
RDTSC		31							
IN		18						>300	
OUT		18						>300	
PREFETCHNTA d)	m				1				
PREFETCHT0/1/2 d)	m				1				
SFENCE d)						1	1		6

Notes

Pentium II and III

- a) Faster under certain conditions: see manual 3: "The microarchitecture of Intel, AMD and VIA CPUs".
- b) Has an implicit LOCK prefix.
- c) 3 if constant without base or index register
- d) P3 only.

**Floating point x87 instructions (Pentium Pro, II and III)**

Instruction	Operands	μops						Latency	Reciprocal throughput
		p0	p1	p01	p2	p3	p4		
FLD	r	1							
FLD	m32/64				1			1	
FLD	m80	2			2				
FBLD	m80	38			2				
FST(P)	r	1							
FST(P)	m32/m64					1	1	1	
FSTP	m80	2				2	2		
FBSTP	m80	165				2	2		
FXCH	r							0	1/3 f)
FILD	m	3			1			5	
FIST(P)	m	2				1	1	5	
FLDZ		1							
FLD1 FLDPI FLDL2E etc.		2							
FCMOVcc	r	2						2	
FNSTSW	AX	3						7	
FNSTSW	m16	1				1	1		
FLDCW	m16	1		1	1			10	
FNSTCW	m16	1				1	1		
FADD(P) FSUB(R)(P)	r	1						3	1
FADD(P) FSUB(R)(P)	m	1			1			3-4	1
FMUL(P)	r	1						5	2 g)
FMUL(P)	m	1			1			5-6	2 g)
FDIV(R)(P)	r	1						38 h)	37
FDIV(R)(P)	m	1			1			38 h)	37
FABS		1							
FCHS		3						2	
FCOM(P) FUCOM	r	1						1	
FCOM(P) FUCOM	m	1			1			1	
FCOMPP FUCOMPP		1		1				1	
FCOMI(P) FUCOMI(P)	r	1						1	
FCOMI(P) FUCOMI(P)	m	1			1			1	
FIADD FISUB(R)	m	6			1				
FIMUL	m	6			1				
FIDIV(R)	m	6			1				
FICOM(P)	m	6			1				
FTST		1						1	
FXAM		1						2	
FPREM		23							
FPREM1		33							
FRNDINT		30							

Pentium II and III

FSCALE		56						
EXTRACT		15						
FSQRT		1					69	e,i)
FSIN FCOS		17-97				27-103	e)	
FSINCOS		18-110				29-130	e)	
F2XM1		17-48				66	e)	
FYL2X		36-54				103	e)	
FYL2XP1		31-53				98-107	e)	
FPTAN		21-102				13-143	e)	
FPATAN		25-86				44-143	e)	
FNOP		1						
FINCSTP FDECSTP		1						
FFREE	r	1						
FFREEP	r	2						
FNCLEX				3				
FNINIT		13						
FNSAVE		141						
FRSTOR		72						
WAIT				2				

**Notes:**

- e) Not pipelined
- f) FXCH generates 1  $\mu$ op that is resolved by register renaming without going to any port.
- g) FMUL uses the same circuitry as integer multiplication. Therefore, the combined throughput of mixed floating point and integer multiplications is 1 FMUL + 1 IMUL per 3 clock cycles.
- h) FDIV latency depends on precision specified in control word: 64 bits precision gives latency 38, 53 bits precision gives latency 32, 24 bits precision gives latency 18. Division by a power of 2 takes 9 clocks. Reciprocal throughput is 1/(latency-1).
- i) Faster for lower precision.

**Integer MMX instructions (Pentium II and Pentium III)**

Instruction	Operands	$\mu$ ops						Latency	Reciprocal throughput
		p0	p1	p01	p2	p3	p4		
MOVD MOVQ	r,r			1				1	0.5
MOVD MOVQ	mm,m32/64				1				1
MOVD MOVQ	m32/64,mm					1	1		1
PADD PSUB PCMP	mm,mm			1				1	0.5
PADD PSUB PCMP	mm,m64			1	1				1
PMUL PMADD	mm,mm	1						3	1
PMUL PMADD	mm,m64	1			1			3	1
PAND(N) POR PXOR	mm,mm			1				1	0.5
PAND(N) POR PXOR	mm,m64			1	1				1
PSRA PSRL PSLL	mm,mm/i		1					1	1
PSRA PSRL PSLL	mm,m64		1		1				1
PACK PUNPCK	mm,mm		1					1	1
PACK PUNPCK	mm,m64		1		1				1
EMMS		11						6 k)	
MASKMOVQ d)	mm,mm			1		1	1	2-8	2 - 30

Pentium II and III

PMOVMSKB d)	r32,mm		1					1	1
MOVNTQ d)	m64,mm					1	1		1 - 30
PSHUFW d)	mm,mm,i		1					1	1
PSHUFW d)	mm,m64,i		1		1			2	1
PEXTRW d)	r32,mm,i		1	1				2	1
PINSRW d)	mm,r32,i		1					1	1
PINSRW d)	mm,m16,i		1		1			2	1
PAVGB PAVGW d)	mm,mm			1				1	0.5
PAVGB PAVGW d)	mm,m64			1	1			2	1
PMIN/MAXUB/SW d)	mm,mm			1				1	0.5
PMIN/MAXUB/SW d)	mm,m64			1	1			2	1
PMULHUW d)	mm,mm	1						3	1
PMULHUW d)	mm,m64	1			1			4	1
PSADBW d)	mm,mm	2		1				5	2
PSADBW d)	mm,m64	2		1	1			6	2

Notes:

- d) P3 only.
- k) The delay can be hidden by inserting other instructions between EMMS and any subsequent floating point instruction.

Floating point XMM instructions (Pentium III)

Instruction	Operands	μops						Latency	Reciprocal throughput
		p0	p1	p01	p2	p3	p4		
MOVAPS	xmm,xmm			2				1	1
MOVAPS	xmm,m128				2			2	2
MOVAPS	m128,xmm					2	2	3	2
MOVUPS	xmm,m128				4			2	4
MOVUPS	m128,xmm		1			4	4	3	4
MOVSS	xmm,xmm			1				1	1
MOVSS	xmm,m32			1	1			1	1
MOVSS	m32,xmm					1	1	1	1
MOVHPS MOVLPS	xmm,m64			1				1	1
MOVHPS MOVLPS	m64,xmm					1	1	1	1
MOVLHPS MOVHLPS	xmm,xmm			1				1	1
MOVMSKPS	r32,xmm	1						1	1
MOVNTPS	m128,xmm					2	2		2 - 15
CVTPI2PS	xmm,mm		2					3	1
CVTPI2PS	xmm,m64		2		1			4	2
CVT(T)PS2PI	mm,xmm		2					3	1
CVTPS2PI	mm,m128		1		2			4	1
CVTSI2SS	xmm,r32		2		1			4	2
CVTSI2SS	xmm,m32		2		2			5	2
CVT(T)SS2SI	r32,xmm		1		1			3	1
CVTSS2SI	r32,m128		1		2			4	2
ADDPS SUBPS	xmm,xmm		2					3	2
ADDPS SUBPS	xmm,m128		2		2			3	2
ADDSS SUBSS	xmm,xmm		1					3	1
ADDSS SUBSS	xmm,m32		1		1			3	1

Pentium II and III

MULPS	xmm,xmm	2					4	2
MULPS	xmm,m128	2			2		4	2
MULSS	xmm,xmm	1					4	1
MULSS	xmm,m32	1			1		4	1
DIVPS	xmm,xmm	2					48	34
DIVPS	xmm,m128	2			2		48	34
DIVSS	xmm,xmm	1					18	17
DIVSS	xmm,m32	1			1		18	17
AND(N)PS ORPS XORPS	xmm,xmm		2				2	2
AND(N)PS ORPS XORPS	xmm,m128		2		2		2	2
MAXPS MINPS	xmm,xmm		2				3	2
MAXPS MINPS	xmm,m128		2		2		3	2
MAXSS MINSS	xmm,xmm		1				3	1
MAXSS MINSS	xmm,m32		1		1		3	1
CMPccPS	xmm,xmm		2				3	2
CMPccPS	xmm,m128		2		2		3	2
CMPccSS	xmm,xmm		1				3	1
CMPccSS	xmm,m32		1		1		3	1
COMISS UCOMISS	xmm,xmm		1				1	1
COMISS UCOMISS	xmm,m32		1		1		1	1
SQRTPS	xmm,xmm	2					56	56
SQRTPS	xmm,m128	2			2		57	56
SQRTSS	xmm,xmm	2					30	28
SQRTSS	xmm,m32	2			1		31	28
RSQRTPS	xmm,xmm	2					2	2
RSQRTPS	xmm,m128	2			2		3	2
RSQRTSS	xmm,xmm	1					1	1
RSQRTSS	xmm,m32	1			1		2	1
RCPPS	xmm,xmm	2					2	2
RCPPS	xmm,m128	2			2		3	2
RCPSS	xmm,xmm	1					1	1
RCPSS	xmm,m32	1			1		2	1
SHUFPS	xmm,xmm,i		2	1			2	2
SHUFPS	xmm,m128,i		2		2		2	2
UNPCKHPS UNPCKLPS	xmm,xmm		2	2			3	2
UNPCKHPS UNPCKLPS	xmm,m128		2		2		3	2
LDMXCSR	m32	11					15	15
STMXCSR	m32	6					7	9
FXSAVE	m4096	116					62	
FXRSTOR	m4096	89					68	

## Intel Pentium M, Core Solo and Core Duo

### List of instruction timings and $\mu$ op breakdown

#### Explanation of column headings:

<b>Operands:</b>	i = immediate data, r = register, mm = 64 bit mmx register, xmm = 128 bit xmm register, sr = segment register, m = memory, m32 = 32-bit memory operand, etc.
<b><math>\mu</math>ops fused domain:</b>	The number of $\mu$ ops at the decode, rename, allocate and retirement stages in the pipeline. Fused $\mu$ ops count as one.
<b><math>\mu</math>ops unfused domain:</b>	The number of $\mu$ ops for each execution port. Fused $\mu$ ops count as two.
<b>p0:</b>	Port 0: ALU, etc.
<b>p1:</b>	Port 1: ALU, jumps
<b>p01:</b>	Instructions that can go to either port 0 or 1, whichever is vacant first.
<b>p2:</b>	Port 2: load data, etc.
<b>p3:</b>	Port 3: address generation for store
<b>p4:</b>	Port 4: store data
<b>Latency:</b>	This is the delay that the instruction generates in a dependency chain. (This is not the same as the time spent in the execution unit. Values may be inaccurate in situations where they cannot be measured exactly, especially with memory operands). The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays by 50-150 clocks, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay.
<b>Reciprocal throughput:</b>	The average number of clock cycles per instruction for a series of independent instructions of the same kind.

### Integer instructions

Instruction	Operands	$\mu$ ops fused domain	$\mu$ ops unfused domain						Latency	Reciprocal throughput
			p0	p1	p01	p2	p3	p4		
<b>Move instructions</b>										
MOV	r,r/i	1			1					0.5
MOV	r,m	1				1				1
MOV	m,r	1					1	1		1
MOV	m,i	2					1	1		1
MOV	r,sr	1			1					
MOV	m,sr	2			1		1	1		
MOV	sr,r	8	8						5	
MOV	sr,m	8	7		1				8	
MOVNTI	m,r32	2					1	1		2
MOVSX MOVZX	r,r	1			1				1	0.5
MOVSX MOVZX	r,m	1				1				1
CMOVcc	r,r	2	1		1				2	1.5
CMOVcc	r,m	2	1		1	1				

Pentium M

XCHG	r,r	3			3				2	1.5
XCHG	r,m	7			4	1	1	1	high b)	
XLAT		2			1	1				1
PUSH	r	1					1	1	1	1
PUSH	i	2					1	1	1	1
PUSH	m	2				1	1	1	2	1
PUSH	sr	2			1		1	1		
PUSHF(D)		16	3		11		1	1		6
PUSHA(D)		18			2		8	8	8	8
POP	r	1				1				
POP	(E)SP	3			2	1				
POP	m	2				1	1	1	2	1
POP	sr	10			9	1				
POPF(D)		17	10		6	1				16
POPA(D)		10			2	8			7	7
LAHF SAHF		1			1				1	1
SALC		2	1	1						1
LEA	r,m	1	1						1	1
BSWAP	r	2	1		1					
LDS LES LFS LGS LSS	m	11			8	3				
PREFETCHNTA	m	1				1				1
PREFETCH0/1/2	m	1				1				1
SFENCE/LFENCE/MFENCE		2					1	1		6
IN			18						>300	
OUT			18						>300	
<b>Arithmetic instructions</b>										
ADD SUB	r,r/i	1			1				1	0.5
ADD SUB	r,m	1			1	1			2	1
ADD SUB	m,r/i	3			1	1	1	1		1
ADC SBB	r,r/i	2		1	1				2	2
ADC SBB	r,m	2		1	1	1				
ADC SBB	m,r/i	7			4	1	1	1		
CMP	r,r/i	1			1				1	0.5
CMP	m,r	1			1	1			1	1
CMP	m,i	2			1	1				1
INC DEC NEG NOT	r	1			1				1	0.5
INC DEC NEG NOT	m	3			1	1	1	1		
AAA AAS DAA DAS		1		1						
AAD		3	1		2				2	
AAM		4	1	1	2				15	
MUL IMUL	r8	1	1						4	1
MUL IMUL	r16/r32	3	3						5	1
IMUL	r,r	1	1						4	1
IMUL	r,r,i	1	1						4	1
MUL IMUL	m8	1	1			1			4	1
MUL IMUL	m16/m32	3	3			1			5	1
IMUL	r,m	1	1			1			4	1
IMUL	r,m,i	2	1			1			4	1
DIV IDIV	r8	5	4		1				15-16 c)	12



Pentium M

DIV IDIV	r16	4	3		1				15-24 c)	12-20 c)
DIV IDIV	r32	4	3		1				15-39 c)	12-20 c)
DIV IDIV	m8	6	4		1	1			15-16 c)	12
DIV IDIV	m16	5	3		1	1			15-24 c)	12-20 c)
DIV IDIV	m32	5	3		1	1			15-39 c)	12-20 c)
CBW CWDE		1		1					1	1
CWD CDQ		1		1					1	1
<b>Logic instructions</b>										
AND OR XOR	r,r/i	1			1				1	0.5
AND OR XOR	r,m	1			1	1			2	1
AND OR XOR	m,r/i	3			1	1	1	1		1
TEST	r,r/i	1			1				1	0.5
TEST	m,r	1			1	1			1	1
TEST	m,i	2			1	1				1
SHR SHL SAR ROR ROL	r,i/CL	1	1						1	1
SHR SHL SAR ROR ROL	m,i/CL	3	1			1	1	1		
RCR RCL	r,1	2	1		1				2	2
RCR	r8,i/CL	9	5		4				11	
RCL	r8,i/CL	8	4		4				10	
RCR RCL	r16/32,i/CL	6	3		3				9	9
RCR RCL	m,1	7	2		2	1	1	1		
RCR	m8,i/CL	12	6		3	1	1	1		
RCL	m8,i/CL	11	5		3	1	1	1		
RCR RCL	m16/32,i/CL	10	5		2	1	1	1		
SHLD SHRD	r,r,i/CL	2	2						2	2
SHLD SHRD	m,r,i/CL	4	1		1	1	1	1		
BT	r,r/i	1		1					1	1
BT	m,r	8			7	1				
BT	m,i	2		1		1				
BTR BTS BTC	r,r/i	1		1						
BTR BTS BTC	m,r	10			7	1	1	1	6	
BTR BTS BTC	m,i	3		1		1	1	1		
BSF BSR	r,r	2		1	1					
BSF BSR	r,m	2		1	1	1				
SETcc	r	1		1						
SETcc	m	2		1			1	1		
CLC STC CMC		1		1						1
CLD STD		4			4					7
<b>Control transfer instructions</b>										
JMP	short/near	1		1						1
JMP	far	22	21			1				28
JMP	r	1		1						1
JMP	m(near)	2		1		1				2
JMP	m(far)	25	23			2				31
conditional jump	short/near	1		1						1
J(E)CXZ	short	2		1	1					1
LOOP	short	11	2	1	8					6
LOOP(N)E	short	11	2	1	8					6

Pentium M

CALL	near	4		1	1		1	1		2
CALL	far	32	27			1	2	2		27
CALL	r	4		1	2		1	1		9
CALL	m(near)	4		1		1	1	1		2
CALL	m(far)	35	29			2	2	2		30
RETN		2		1	2	1				2
RETN	i	3		1	1	1				2
RETF		27	24			3				30
RETF	i	27	24			3				30
BOUND	r,m	15	7		6	2				8
INTO		5			5					4
<b>String instructions</b>										
LODS		2				2				4
REP LODS		6n			10+6n					0.5
STOS		3				1	1	1		1
REP STOS		5n			ca. 5n a)					0.7
MOVS		6			1	3	1	1		0.7
REP MOVS		6n			ca. 6n a)					0.5
SCAS		3			1	2				1.3
REP(N)E SCAS		7n			12+7n					0.6
CMPS		6			4	2				0.7
REP(N)E CMPS		9n			12+9n					0.5
<b>Other</b>										
NOP (90)		1			1					0.5
Long NOP (0F 1F)		1			1					1
PAUSE		2			2					
CLI			9							
STI			17							
ENTER	i,0	12			10		1	1		
ENTER	a,b		ca.	18	+4b		b-1	2b		
LEAVE		3			2	1				
CPUID		38-59	38-59						ca. 130	
RDTSC		13	13							42

**Notes:**

- a) Faster under certain conditions: see manual 3: "The microarchitecture of Intel, AMD and VIA CPUs".
- b) Has an implicit LOCK prefix.
- c) High values are typical, low values are for round divisors. Core Solo/Duo is more efficient than Pentium M in cases with round values that allow an early-out algorithm.

**Floating point x87 instructions**

Instruction	Operands	μops fused domain	μops unfused domain						Latency	Recip- rocal through put
			p0	p1	p01	p2	p3	p4		
<b>Move instructions</b>										

Pentium M

FLD	r	1	1					1	
FLD	m32/64	1				1		1	
FLD	m80	4	2			2			
FBLD	m80	40	38			2			
FST(P)	r	1	1						
FST(P)	m32/m64	1					1	1	1
FSTP	m80	6	2				2	2	3
FBSTP	m80	169	165				2	2	167
FXCH	r	1							0.33 f)
FILD	m	4	3			1		5	2
FIST(P)	m	4	2				1	1	5
FISTTP g)	m	4	2				1	1	5
FLDZ		1	1						
FLD1 FLDPI FLDL2E etc.		2	2						
FCMOVcc	r	2	2					2	
FNSTSW	AX	3	3					7	3
FNSTSW	m16	2	1				1	1	
FLDCW	m16	3	1		1	1			19
FNSTCW	m16	3	1				1	1	3
FINCSTP FDECSTP		1	1					1	
FFREE	r	1	1						1
FFREEP	r	2	2						2
FNSAVE		142	142						131
FRSTOR		72	72						91
<b>Arithmetic instructions</b>									
FADD(P) FSUB(R)(P)	r	1				1		3	1
FADD(P) FSUB(R)(P)	m	1				1	1	3	1
FMUL(P)	r	1	1					5	2
FMUL(P)	m	1	1				1	5	2
FDIV(R)(P)	r	1	1					9-38 c)	8-37 c)
FDIV(R)(P)	m	1	1				1	9-38 c)	8-37 c)
FABS		1	1					1	1
FCHS		1	1					1	1
FCOM(P) FUCOM	r	1			1			1	1
FCOM(P) FUCOM	m	1			1	1		1	1
FCOMPP FUCOMPP		2			1	1		1	1
FCOMI(P) FUCOMI(P)	r	1			1			1	1
FIADD FISUB(R)	m	6	3		1	1	1	3	3
FIMUL	m	6	5				1	5	3
FIDIV(R)	m	6	5					9-38 c)	8-37 c)
FICOM(P)	m	6	3		2		1		4
FTST		1			1				1
FXAM		1			1				1
FPREM FPREM1		26	26					37	
FRNDINT		15	15					19	
<b>Math</b>									
FSCALE		28	28					43	
FXTRACT		15				15		9	

Pentium M

FSQRT	1	1					9 h)	8
FSIN FCOS	80-100	80-100					80-110	
FSINCOS	90-110	90-110					100-130	
F2XM1	~ 20	~20					~45	
FYL2X	~ 40	~40					~60	
FYL2XP1	~ 55	~55					~65	
FPTAN	~ 100	~100					~140	
FPATAN	~ 85	~85					~140	
<b>Other</b>								
FNOP	1	1						1
WAIT	2		1	1				1
FNCLEX	3	3						13
FNINIT	14	14						27

Notes:

- c) High values are typical, low values are for low precision or round divisors.
- f) FXCH generates 1  $\mu$ op that is resolved by register renaming without going to any port.
- g) SSE3 instruction only available on Core Solo and Core Duo.

Integer MMX and XMM instructions

Instruction	Operands	$\mu$ ops fused domain	$\mu$ ops unfused domain						Latency	Recip- recal through put
			p0	p1	p01	p2	p3	p4		
<b>Move instructions</b>										
MOVD	r32,mm	1			1				1	0.5
MOVD	mm,r32	1			1				1	0.5
MOVD	mm,m32	1				1				1
MOVD	m32,mm	1					1	1		1
MOVD	r32,xmm	1		1					1	1
MOVD	xmm,r32	2			2					1
MOVD	xmm,m32	2			1	1				1
MOVD	m32, xmm	1					1	1		1
MOVQ	mm,mm	1			1					0.5
MOVQ	mm,m64	1				1				1
MOVQ	m64,mm	1					1	1		1
MOVQ	xmm,xmm	2			2				1	1
MOVQ	xmm,m64	2			1	1				1
MOVQ	m64, xmm	1					1	1		1
MOVDQA	xmm, xmm	2			2				1	1
MOVDQA	xmm, m128	2				2				2
MOVDQA	m128, xmm	2					2	2		2
MOVDQU	xmm, m128	4			2	2				2-10
MOVDQU	m128, xmm	8			5-6		2-3	2-3		4-20
LDDQU g)	xmm, m128	4								2
MOVDQ2Q	mm, xmm	1		1					1	1
MOVQ2DQ	xmm,mm	2		1	1				1	1
MOVNTQ	m64,mm	1					1	1		2

Pentium M

MOVNTDQ	m128,xmm	4					2	2		3
PACKSSWB/DW										
PACKUSWB	mm,mm	1	1						1	1
PACKSSWB/DW										
PACKUSWB	mm,m64	1	1			1			1	1
PACKSSWB/DW										
PACKUSWB	xmm,xmm	3	2	1					2	2
PACKSSWB/DW										
PACKUSWB	xmm,m128	4	1	1		2			2	2
PUNPCKH/LBW/WD/DQ	mm,mm	1	1						1	1
PUNPCKH/LBW/WD/DQ	mm,m64	1	1			1				1
PUNPCKH/LBW/WD/DQ	xmm,xmm	2	2						2	2
PUNPCKH/LBW/WD/DQ	xmm,m128	3	1			2				2
PUNPCKHQDQ	xmm,xmm	2		1	1				1	1
PUNPCKHQDQ	xmm, m128	3		1		2				1
PUNPCKLQDQ	xmm,xmm	1		1					1	1
PUNPCKLQDQ	xmm, m128	1				1				1
PSHUFW	mm,mm,i	1	1						1	1
PSHUFW	mm,m64,i	2	1			1				1
PSHUFD	xmm,xmm,i	3	2	1					2	2
PSHUFD	xmm,m128,i	4	1	1		2				2
PSHUFL/HW	xmm,xmm,i	2	1	1						1
PSHUFL/HW	xmm, m128,i	3		1		2				1
MASKMOVQ	mm,mm	3			1		1	1		
MASKMOVDQU	xmm,xmm	8		1			2	2		
PMOVMSKB	r32,mm	1	1						1	1
PMOVMSKB	r32,xmm	1	1	j)					1	1
PEXTRW	r32,mm,i	2	1	1					2	1
PEXTRW	r32,xmm,i	4	2	2					3	2
PINSRW	mm,r32,i	1	1						1	1
PINSRW	xmm,r32,i	2	2						1	2
<b>Arithmetic instructions</b>										
PADD/SUB(U)(S)B/W/D	mm,mm	1				1			1	0.5
PADD/SUB(U)(S)B/W/D	mm,m64	1				1	1			1
PADD/SUB(U)(S)B/W/D	xmm,xmm	2				2			1	1
PADD/SUB(U)(S)B/W/D	xmm,m128	4				2	2			2
PADDQ PSUBQ	mm,mm	2				2			2	1
PADDQ PSUBQ	mm,m64	2				2	1			1
PADDQ PSUBQ	xmm,xmm	4				4			2	2
PADDQ PSUBQ	xmm,m128	6				4	2			2
PCMPEQ/GTB/W/D	mm,mm	1				1			1	0.5
PCMPEQ/GTB/W/D	mm,m64	1				1	1			1
PCMPEQ/GTB/W/D	xmm,xmm	2				2			1	1
PCMPEQ/GTB/W/D	xmm,m128	2				2	2			2
PMULL/HW PMULHUW	mm,mm	1				1			3	1
PMULL/HW PMULHUW	mm,m64	1				1	1		3	1
PMULL/HW PMULHUW	xmm,xmm	2				2			3	2
PMULL/HW PMULHUW	xmm,m128	4				2	2		3	2
PMULUDQ	mm,mm	1	1						4	1
PMULUDQ	mm,m64	1	1				1		4	1

Pentium M

PMULUDQ	xmm,xmm	2	2					4	2
PMULUDQ	xmm,m128	4	2			2		4	2
PMADDWD	mm,mm	1			1			3	1
PMADDWD	mm,m64	1			1	1		3	1
PMADDWD	xmm,xmm	2			2			3	2
PMADDWD	xmm,m128	4			2	2		3	2
PAVGB/W	mm,mm	1			1			1	0.5
PAVGB/W	mm,m64	1			1	1			1
PAVGB/W	xmm,xmm	2			2			1	1
PAVGB/W	xmm,m128	4			2	2			2
PMIN/MAXUB/SW	mm,mm	1			1			1	0.5
PMIN/MAXUB/SW	mm,m64	1			1	1			1
PMIN/MAXUB/SW	xmm,xmm	2			2			1	1
PMIN/MAXUB/SW	xmm,m128	4			2	2			2
PSADBW	mm,mm	2			2			4	1
PSADBW	mm,m64	2			2	1		4	1
PSADBW	xmm,xmm	4			4			4	2
PSADBW	xmm,m128	6			4	2		4	2
<b>Logic instructions</b>									
PAND(N) POR PXOR	mm,mm	1			1			1	0.5
PAND(N) POR PXOR	mm,m64	1			1	1			1
PAND(N) POR PXOR	xmm,xmm	2			2			1	1
PAND(N) POR PXOR	xmm,m128	4			2	2			2
PSLL/RL/RAW/D/Q	mm,mm/i	1	1					1	1
PSLL/RL/RAW/D/Q	mm,m64	1	1			1			1
PSLL/RL/RAW/D/Q	xmm,i	2	2					2	2
PSLL/RL/RAW/D/Q	xmm,xmm	3	2	1				2	2
PSLL/RL/RAW/D/Q	xmm,m128	3		1		2			2
PSLL/RLDQ	xmm,i	4	3	1				3	3
<b>Other</b>									
EMMS		11			11			6 k)	6

**Notes:**

- g) SSE3 instruction only available on Core Solo and Core Duo.
- j) Also uses some execution units under port 1.
- k) You may hide the delay by inserting other instructions between EMMS and any subsequent floating point instruction.

**Floating point XMM instructions**

Instruction	Operands	μops fused domain	μops unfused domain						Latency	Recip- rocal through put
			p0	p1	p01	p2	p3	p4		
<b>Move instructions</b>										
MOVAPS/D	xmm,xmm	2			2				1	1
MOVAPS/D	xmm,m128	2				2			2	2
MOVAPS/D	m128,xmm	2					2	2	3	2
MOVUPS/D	xmm,m128	4				4			2	2

Pentium M

MOVUPS/D	m128,xmm	8			4		2	2	3	4
MOVSS/D	xmm,xmm	1		1					1	1
MOVSS/D	xmm,m32/64	2		1		1			1	1
MOVSS/D	m32/64,xmm	1					1	1	1	1
MOVHPS/D MOVLPS/D	xmm,m64	1		1		1			1	1
MOVHPS/D MOVLPS/D	m64,xmm	1					1	1	1	1
MOVLHPS MOVHLPS	xmm,xmm	1		1					1	1
MOVMSKPS/D	r32,xmm	1	1	j)					2	1
MOVNTPS/D	m128,xmm	2					2	2		3
SHUFPS/D	xmm,xmm,i	3	2	1					2	2
SHUFPS/D	xmm,m128,i	4	1	1		2				2
MOVDDUP g)	xmm,xmm	2							1	1
MOVSH/LDUP g)	xmm,xmm	2							2	2
MOVSH/LDUP g)	xmm,m128	4								
UNPCKH/LPS	xmm,xmm	4	2	2					3-4	5
UNPCKH/LPS	xmm,m128	4		2		2				5
UNPCKH/LPD	xmm,xmm	2		1	1				1	1
UNPCKH/LPD	xmm,m128	3		1	1	1				1
<b>Conversion</b>										
CVTPS2PD	xmm,xmm	4	2	2					3	3
CVTPS2PD	xmm,m64	4	1	2		1				3
CVTPD2PS	xmm,xmm	4	3	1					4	3
CVTPD2PS	xmm,m128	6	3	1		2				3
CVTSD2SS	xmm,xmm	2			2				4	2
CVTSD2SS	xmm,m64	3			2	1				2
CVTSS2SD	xmm,xmm	2	2						2	2
CVTSS2SD	xmm,m64	3	2			1				2
CVTDQ2PS	xmm,xmm	2			2				3	2
CVTDQ2PS	xmm,m128	4			2	2				2
CVT(T) PS2DQ	xmm,xmm	2			2				3	2
CVT(T) PS2DQ	xmm,m128	4			2	2				2
CVTDQ2PD	xmm,xmm	4			4				4	2
CVTDQ2PD	xmm,m64	5			4	1				2
CVT(T)PD2DQ	xmm,xmm	4			4				4	3
CVT(T)PD2DQ	xmm,m128	6			4	2				3
CVTPI2PS	xmm,mm	1		1					3	1
CVTPI2PS	xmm,m64	2		1		1				1
CVT(T)PS2PI	mm,xmm	1		1					3	1
CVT(T)PS2PI	mm,m128	2		1		1				1
CVTPI2PD	xmm,mm	4	2	2					5	2
CVTPI2PD	xmm,m64	5	2	2		1				2
CVT(T) PD2PI	mm,xmm	3			3				4	2
CVT(T) PD2PI	mm,m128	5			3	2				2
CVTSI2SS	xmm,r32	2	1	1					4	1
CVT(T)SS2SI	r32,xmm	2		1	1				4	1
CVT(T)SS2SI	r32,m32	3		1	1	1				1
CVTSI2SD	xmm,r32	2	1	1					4	1
CVTSI2SD	xmm,m32	3	1	1		1				1
CVT(T)SD2SI	r32,xmm	2		1	1				4	1

## Pentium M

CVT(T)SD2SI	r32,m64	3		1	1	1			1
<b>Arithmetic</b>									
ADDSS/D SUBSS/D	xmm,xmm	1			1			3	1
ADDSS/D SUBSS/D	xmm,m32/64	2			1	1		3	1
ADDPSS/D SUBPS/D	xmm,xmm	2			2			3	2
ADDPSS/D SUBPS/D	xmm,m128	4			2	2		3	2
ADDSUBPS/D g)	xmm,xmm	2			2			3	2
HADDPSS HSUBPS g)	xmm,xmm	6?			?			7	4
HADDPD HSUBPD g)	xmm,xmm	3			3			4	2
MULSS	xmm,xmm	1	1					4	1
MULSD	xmm,xmm	1	1					5	2
MULSS	xmm,m32	2	1			1		4	1
MULSD	xmm,m64	2	1			1		5	2
MULPS	xmm,xmm	2	2					4	2
MULPD	xmm,xmm	2	2					5	4
MULPS	xmm,m128	4	2			2		4	2
MULPD	xmm,m128	4	2			2		5	4
DIVSS	xmm,xmm	1	1					9-18 c)	8-17 c)
DIVSD	xmm,xmm	1	1					9-32 c)	8-31 c)
DIVSS	xmm,m32	2	1			1		9-18 c)	8-17 c)
DIVSD	xmm,m64	2	1			1		9-32 c)	8-31 c)
DIVPS	xmm,xmm	2	2					16-34 c)	16-34 c)
DIVPD	xmm,xmm	2	2					16-62 c)	16-62 c)
DIVPS	xmm,m128	4	2			2		16-34 c)	16-34 c)
DIVPD	xmm,m128	4	2			2		16-62 c)	16-62 c)
CMPccSS/D	xmm,xmm	1			1			3	1
CMPccSS/D	xmm,m32/64	2			1	1			1
CMPccPS/D	xmm,xmm	2			2			3	2
CMPccPS/D	xmm,m128	4			2	2			2
COMISS/D UCOMISS/D	xmm,xmm	1		1					1
COMISS/D UCOMISS/D	xmm,m32/64	2		1		1			1
MAXSS/D MINSS/D	xmm,xmm	1			1			3	1
MAXSS/D MINSS/D	xmm,m32/64	2			1	1		3	1
MAXPS/D MINPS/D	xmm,xmm	2			2			3	2
MAXPS/D MINPS/D	xmm,m128	4			2	2		3	2
RCPSS	xmm,xmm	1		1				3	1
RCPSS	xmm,m32	2		1		1			1
RCPPS	xmm,xmm	2		2				3	2
RCPPS	xmm,m128	4		2		2			2
<b>Math</b>									
SQRTSS	xmm,xmm	2	2					6-30	4-28
SQRTSS	xmm,m32	3	2			1			4-28
SQRTSD	xmm,xmm	1	1					5-58	4-57
SQRTSD	xmm,m64	2	1			1			4-57
SQRTPS	xmm,xmm	2	2					8-56	16-55
SQRTPD	xmm,xmm	2	2					16-114	16-114
SQRTPS	xmm,m128	4	2			2			16-55
SQRTPD	xmm,m128	4	2			2			16-114



Pentium M

RSQRTSS	xmm,xmm	1		1				3	1
RSQRTSS	xmm,m32	2		1		1			1
RSQRTPS	xmm,xmm	2		3				3	2
RSQRTPS	xmm,m128	4		2		2			2
<b>Logic</b>									
AND/ANDN/OR/XORPS/D	xmm,xmm	2				2		1	1
AND/ANDN/OR/XORPS/D	xmm,m128	4				2	2		1
<b>Other</b>									
LDMXCSR	m32	9	9						20
STMXCSR	m32	6	6						12
FXSAVE	m4096	118	32				43	43	63
FXRSTOR	m4096	87	43			44			72

**Notes:**

- c) High values are typical, low values are for round divisors.
- g) SSE3 instruction only available on Core Solo and Core Duo.
- j) Also uses some execution units under port 1.

## Intel Core 2 (Merom, 65nm)

### List of instruction timings and $\mu$ op breakdown

#### Explanation of column headings:

**Operands:** i = immediate data, r = register, mm = 64 bit mmx register, xmm = 128 bit xmm register, (x)mm = mmx or xmm register, sr = segment register, m = memory, m32 = 32-bit memory operand, etc.

**$\mu$ ops fused domain:** The number of  $\mu$ ops at the decode, rename, allocate and retirement stages in the pipeline. Fused  $\mu$ ops count as one.

**$\mu$ ops unfused domain:** The number of  $\mu$ ops for each execution port. Fused  $\mu$ ops count as two. Fused macro-ops count as one. The instruction has  $\mu$ op fusion if the sum of the numbers listed under p015 + p2 + p3 + p4 exceeds the number listed under  $\mu$ ops fused domain. An x under p0, p1 or p5 means that at least one of the  $\mu$ ops listed under p015 can optionally go to this port. For example, a 1 under p015 and an x under p0 and p5 means one  $\mu$ op which can go to either port 0 or port 5, whichever is vacant first. A value listed under p015 but nothing under p0, p1 and p5 means that it is not known which of the three ports these  $\mu$ ops go to.

**p015:** The total number of  $\mu$ ops going to port 0, 1 and 5.

**p0:** The number of  $\mu$ ops going to port 0 (execution units).

**p1:** The number of  $\mu$ ops going to port 1 (execution units).

**p5:** The number of  $\mu$ ops going to port 5 (execution units).

**p2:** The number of  $\mu$ ops going to port 2 (memory read).

**p3:** The number of  $\mu$ ops going to port 3 (memory write address).

**p4:** The number of  $\mu$ ops going to port 4 (memory write data).

**Unit:** Tells which execution unit cluster is used. An additional delay of 1 clock cycle is generated if a register written by a  $\mu$ op in the integer unit (int) is read by a  $\mu$ op in the floating point unit (float) or vice versa. flt→int means that an instruction with multiple  $\mu$ ops receive the input in the float unit and delivers the output in the int unit. Delays for moving data between different units are included under latency when they are unavoidable. For example, movd eax,xmm0 has an extra 1 clock delay for moving from the XMM-integer unit to the general purpose integer unit. This is included under latency because it occurs regardless of which instruction comes next. Nothing listed under unit means that additional delays are either unlikely to occur or unavoidable and therefore included in the latency figure.

**Latency:** This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays very much, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay. The time unit used is core clock cycles, not the reference clock cycles given by the time stamp counter.

**Reciprocal throughput:** The average number of core clock cycles per instruction for a series of independent instructions of the same kind in the same thread.

### Integer instructions

Instruction	Operands	$\mu$ ops fused domain	$\mu$ ops unfused domain							Unit	Latency	Reciprocal throughput
			p015	p0	p1	p5	p2	p3	p4			

Merom

<b>Move instructions</b>													
MOV	r,r/i	1	1	x	x	x					int	1	0.33
MOV a)	r,m	1					1				int	2	1
MOV a)	m,r	1						1	1		int	3	1
MOV	m,i	1						1	1		int	3	1
MOV	r,sr	1					1				int		1
MOV	m,sr	2					1	1	1		int		1
MOV	sr,r	8	4	x	x	x	4				int		16
MOV	sr,m	8	3	x		x	5				int		16
MOVNTI	m,r	2						1	1		int		2
MOVSX MOVZX													
MOVSXD	r,r	1	1	x	x	x					int	1	0.33
MOVSX MOVZX	r,m	1					1				int		1
CMOVcc	r,r	2	2	x	x	x					int	2	1
CMOVcc	r,m	2	2	x	x	x	1				int		
XCHG	r,r	3	3	x	x	x					int	2	2
XCHG	r,m	7					1	1	1		int	high b)	
XLAT		2	1				1				int	4	1
PUSH	r	1						1	1		int	3	1
PUSH	i	1						1	1		int		1
PUSH	m	2					1	1	1		int		1
PUSH	sr	2	1					1	1		int		1
PUSHF(D/Q)		17	15	x	x	x		1	1		int		7
PUSHA(D) i)		18	9					1	8		int		8
POP	r	1					1				int	2	1
POP	(E/R)SP	4	3				1				int		
POP	m	2					1	1	1		int		1.5
POP	sr	10	9				1				int		17
POPF(D/Q)		24	23	x	x	x	1				int	20	
POPA(D) i)		10	2				8				int		7
LAHF SAHF		1	1	x	x	x					int	1	0.33
SALC i)		2	2	x	x	x					int	4	1
LEA a)	r,m	1	1	1							int	1	1
BSWAP	r	2	2	1		1					int	4	1
LDS LES LFS LGS LSS	m	11	11				1				int		17
PREFETCHNTA	m	1					1				int		1
PREFETCHT0/1/2	m	1					1				int		1
LFENCE		2						1	1		int		8
MFENCE		2						1	1		int		9
SFENCE		2						1	1		int		9
CLFLUSH	m8	4	2	x	x	x		1	1		int	240	117
IN											int		
OUT											int		
<b>Arithmetic instructions</b>													
ADD SUB	r,r/i	1	1	x	x	x					int	1	0.33
ADD SUB	r,m	1	1	x	x	x	1				int		1
ADD SUB	m,r/i	2	1	x	x	x	1	1	1		int	6	1
ADC SBB	r,r/i	2	2	x	x	x					int	2	2
ADC SBB	r,m	2	2	x	x	x	1				int	2	2

## Merom

ADC SBB	m,r/i	4	3	x	x	x	1	1	1	int	7	
CMP	r,r/i	1	1	x	x	x				int	1	0.33
CMP	m,r/i	1	1	x	x	x	1			int	1	1
INC DEC NEG NOT	r	1	1	x	x	x				int	1	0.33
INC DEC NEG NOT	m	3	1	x	x	x	1	1	1	int	6	1
AAA AAS DAA DAS i)		1	1		1					int		1
AAD i)		3	3	x	x	x				int		1
AAM i)		4	4							int	17	
MUL IMUL	r8	1	1		1					int	3	1
MUL IMUL	r16	3	3	x	x	x				int	5	1.5
MUL IMUL	r32	3	3	x	x	x				int	5	1.5
MUL IMUL	r64	3	3	x	x	x				int	7	4
IMUL	r16,r16	1	1		1					int	3	1
IMUL	r32,r32	1	1		1					int	3	1
IMUL	r64,r64	1	1	1						int	5	2
IMUL	r16,r16,i	1	1		1					int	3	1
IMUL	r32,r32,i	1	1		1					int	3	1
IMUL	r64,r64,i	1	1	1						int	5	2
MUL IMUL	m8	1	1		1		1			int	3	1
MUL IMUL	m16	3	3	x	x	x	1			int	5	1.5
MUL IMUL	m32	3	3	x	x	x	1			int	5	1.5
MUL IMUL	m64	3	2	2			1			int	7	4
IMUL	r16,m16	1	1		1		1			int	3	1
IMUL	r32,m32	1	1		1		1			int	3	1
IMUL	r64,m64	1	1	1			1			int	5	2
IMUL	r16,m16,i	1	1		1		1			int		2
IMUL	r32,m32,i	1	1		1		1			int		1
IMUL	r64,m64,i	1	1	1			1			int		2
DIV IDIV	r8	3	3							int	18	12
DIV IDIV	r16	5	5							int	18-26	12-20 c)
DIV IDIV	r32	4	4							int	18-42	12-36 c)
DIV	r64	32	32							int	29-61	18-37 c)
IDIV	r64	56	56							int	39-72	28-40 c)
DIV IDIV	m8	4	3				1			int	18	12
DIV IDIV	m16	6	5				1			int	18-26	12-20 c)
DIV IDIV	m32	5	4				1			int	18-42	12-36 c)
DIV	m64	32	31				1			int	29-61	18-37 c)
IDIV	m64	56	55				1			int	39-72	28-40 c)
CBW CWDE CDQE		1	1	x	x	x				int	1	
CWD CDQ CQO		1	1	x		x				int	1	
<b>Logic instructions</b>												
AND OR XOR	r,r/i	1	1	x	x	x				int	1	0.33
AND OR XOR	r,m	1	1	x	x	x	1			int		1
AND OR XOR	m,r/i	2	1	x	x	x	1	1	1	int	6	1
TEST	r,r/i	1	1	x	x	x				int	1	0.33
TEST	m,r/i	1	1	x	x	x	1			int		1
SHR SHL SAR	r,i/cl	1	1	x		x				int	1	0.5
SHR SHL SAR	m,i/cl	3	2	x		x	1	1	1	int	6	1
ROR ROL	r,i/cl	1	1	x		x				int	1	1

Merom

ROR ROL	m,i/cl	3	2	x		x	1	1	1	int	6	1
RCR RCL	r,1	2	2	x	x	x				int	2	2
RCR	r8,i/cl	9	9	x	x	x				int	12	
RCL	r8,i/cl	8	8	x	x	x				int	11	
RCR RCL	r16/32/64,i/cl	6	6	x	x	x				int	11	
RCR RCL	m,1	4	3	x	x	x	1	1	1	int	7	
RCR	m8,i/cl	12	9	x	x	x	1	1	1	int	14	
RCL	m8,i/cl	11	8	x	x	x	1	1	1	int	13	
RCR RCL	m16/32/64,i/cl	10	7	x	x	x	1	1	1	int	13	
SHLD SHRD	r,r,i/cl	2	2	x	x	x				int	2	1
SHLD SHRD	m,r,i/cl	3	2	x	x	x	1	1	1	int	7	
BT	r,r/i	1	1	x	x	x				int	1	1
BT	m,r	10	9	x	x	x	1			int		5
BT	m,i	2	1	x	x	x	1			int		1
BTR BTS BTC	r,r/i	1	1	x	x	x				int	1	
BTR BTS BTC	m,r	11	8	x	x	x	1	1	1	int	5	
BTR BTS BTC	m,i	3	1	x	x	x	1	1	1	int	6	
BSF BSR	r,r	2	2	x	1	x				int	2	1
BSF BSR	r,m	2	2	x	1	x	1			int		2
SETcc	r	1	1	x	x	x				int	1	1
SETcc	m	2	1	x	x	x		1	1	int		1
CLC STC CMC		1	1	x	x	x				int	1	0.33
CLD		7	7	x	x	x				int		4
STD		6	6	x	x	x				int		14
<b>Control transfer instructions</b>												
JMP	short/near	1	1			1				int	0	1-2
JMP i)	far	30	30							int		76
JMP	r	1	1			1				int	0	1-2
JMP	m(near)	1	1			1	1			int	0	1-2
JMP	m(far)	31	29				2			int		68
Conditional jump	short/near	1	1			1				int	0	1
Fused compare/test and branch	e,i)	1	1			1				int	0	1
J(E/R)CXZ	short	2	2	x	x	1				int		1-2
LOOP	short	11	11	x	x	x				int		5
LOOP(N)E	short	11	11	x	x	x				int		5
CALL	near	3	2	x	x	x		1	1	int		2
CALL i)	far	43	43							int		75
CALL	r	3	2					1	1	int		2
CALL	m(near)	4	3				1	1	1	int		2
CALL	m(far)	44	42				2			int		75
RETN		1				1	1			int		2
RETN	i	3	1			1	1			int		2
RETF		32	30				2			int		78
RETF	i	32	30				2			int		78
BOUND i)	r,m	15	13				2			int		8
INTO i)		5	5							int		3
<b>String instructions</b>												
LODS		3	2				1			int		1

Merom

REP LODS		4+7n - 14+6n								int	1+5n - 21+3n	
STOS		4	2						1	1	int	1
REP STOS		8+5n - 20+1.2n								int	7+2n - 0.55n	
MOVS		8	5								int	
		1	1	1		5					int	
REP MOVS		7+7n - 13+n								int	1+3n - 0.63n	
SCAS		4	3					1			int	1
REP(N)E SCAS		7+8n - 17+7n								int	3+8n - 23+6n	
CMPS		7	5					2			int	3
REP(N)E CMPS		7+10n - 7+9n								int	2+7n - 22+5n	
Other												
NOP (90)		1	1	x	x	x					int	0.33
Long NOP (0F 1F)		1	1	x	x	x					int	1
PAUSE		3	3	x	x	x					int	8
ENTER	i,0	12	10						1	1	int	8
ENTER	a,b										int	
LEAVE		3	2					1			int	
CPUID		46-100								int	180-215	
RDTSC		29								int	64	
RDPMSR		23								int	54	

Notes:

- a) Applies to all addressing modes
- b) Has an implicit LOCK prefix.
- c) Low values are for small results, high values for high results.
- e) See manual 3: "The microarchitecture of Intel, AMD and VIA CPUs" for restrictions on macro-op fusion.
- i) Not available in 64 bit mode.

Floating point x87 instructions

Instruction	Operands	μops fused domain	μops unfused domain								Unit	Latency	Reciprocal throughput
			p015	p0	p1	p5	p2	p3	p4				
<b>Move instructions</b>													
FLD	r	1	1	1							float	1	1
FLD	m32/64	1	1				1				float	3	1
FLD	m80	4	2	2			2				float	4	3
FBLD	m80	40	38				2				float	45	20
FST(P)	r	1	1	1							float	1	1
FST(P)	m32/m64	1						1	1		float	3	1
FSTP	m80	7	3	x	x	x		2	2		float	4	5
FBSTP	m80	170	166	x	x	x		2	2		float	164	166
FXCH	r	1	0 f)								float	0	1
FILD	m	1	1	1			1				float	6	1
FIST	m	2	1		1			1	1		float	6	1
FISTP	m	3	1		1			1	1		float	6	1
FISTTP g)	m	3	1		1			1	1		float	6	1
FLDZ		1	1	1							float		1
FLD1		2	2	1	1						float		2

Merom

FLDPI FLDL2E etc.		2	2	2				float		2
FCMOVcc	r	2	2	2				float	2	2
FNSTSW	AX	1	1	1				float		1
FNSTSW	m16	2	1	1		1	1	float		2
FLDCW	m16	2	1			1		float		10
FNSTCW	m16	3	1				1	float		8
FINCSTP FDECSTP		1	1	1				float	1	1
FFREE(P)	r	2	2	2				float		2
FNSAVE	m	142						float	184	192
FRSTOR	m	78						float	169	177
<b>Arithmetic instructions</b>										
FADD(P) FSUB(R)(P)	r	1	1		1			float	3	1
FADD(P) FSUB(R)(P)	m	1	1		1	1		float		1
FMUL(P)	r	1	1	1				float	5	2
FMUL(P)	m	1	1	1		1		float		2
FDIV(R)(P)	r	1	1	1				float	6-38 d)	5-37 d)
FDIV(R)(P)	m	1	1	1		1		float		5-37 d)
FABS		1	1	1				float	1	1
FCHS		1	1	1				float	1	1
FCOM(P) FUCOM	r	1	1		1			float		1
FCOM(P) FUCOM	m	1	1		1	1		float		1
FCOMPP FUCOMPP		2	2	1	1			float		
FCOMI(P) FUCOMI(P)	r	1	1		1			float		1
FIADD FISUB(R)	m	2	2	1	1	1		float		2
FIMUL	m	2	2	2		1		float		2
FIDIV(R)	m	2	2	2		1		float		5-37 d)
FICOM(P)	m	2	2	1	1	1		float		2
FTST		1	1		1			float		1
FXAM		1	1		1			float		1
FPREM FPREM1		21-27	21-27					float	16-56	
FRNDINT		7-15	7-15					float	22-29	
<b>Math</b>										
FSCALE		27	27					float	41	
FEXTRACT		82	82					float	170	
FSQRT		1	1					float	6-69	
FSIN FCOS		~96	~96					float	~96	
FSINCOS		~100	~100					float	~115	
F2XM1		~19	~19					float	~45	
FYL2X FYL2XP1		~53	~53					float	~96	
FPTAN		~98	~98					float	~136	
FPATAN		~70	~70					float	~119	
<b>Other</b>										
FNOP		1	1	1				float		1
WAIT		2	2					float		1
FNCLEX		4	4					float		15
FNINIT		15	15					float		63

Notes:

- d) Round divisors or low precision give low values.  
 f) Resolved by register renaming. Generates no  $\mu$ ops in the unfused domain.  
 g) SSE3 instruction set.

### Integer MMX and XMM instructions

Instruction	Operands	$\mu$ ops fused domain	$\mu$ ops unfused domain							Unit	Latency	Reciprocal throughput
			p015	p0	p1	p5	p2	p3	p4			
<b>Move instructions</b>												
MOVD k)	r32/64,(x)mm	1	1	x	x	x				int	2	0.33
MOVD k)	m32/64,(x)mm	1						1	1		3	1
MOVD k)	(x)mm,r32/64	1	1	x		x				int	2	0.5
MOVD k)	(x)mm,m32/64	1					1			int	2	1
MOVQ	(x)mm, (x)mm	1	1	x	x	x				int	1	0.33
MOVQ	(x)mm,m64	1					1			int	2	1
MOVQ	m64, (x)mm	1						1	1		3	1
MOVDQA	xmm, xmm	1	1	x	x	x				int	1	0.33
MOVDQA	xmm, m128	1					1			int	2	1
MOVDQA	m128, xmm	1						1	1		3	1
MOVDQU	m128, xmm	9	4	x	x	x	1	2	2		3-8	4
MOVDQU	xmm, m128	4	2	x		x	2			int	2-8	2
LDDQU g)	xmm, m128	4	2	x		x	2			int	2-8	2
MOVDQ2Q	mm, xmm	1	1	x	x	x				int	1	0.33
MOVQ2DQ	xmm,mm	1	1	x	x	x				int	1	0.33
MOVNTQ	m64,mm	1						1	1			2
MOVNTDQ	m128,xmm	1						1	1			2
PACKSSWB/DW PACKUSWB	mm,mm	1	1	1						int	1	1
PACKSSWB/DW PACKUSWB	mm,m64	1	1	1			1			int		1
PACKSSWB/DW PACKUSWB	xmm,xmm	3	3							flt→int	3	2
PACKSSWB/DW PACKUSWB	xmm,m128	4	3				1			int		2
PUNPCKH/LBW/WD/DQ	mm,mm	1	1	1						int	1	1
PUNPCKH/LBW/WD/DQ	mm,m64	1	1	1			1			int		1
PUNPCKH/LBW/WD/DQ	xmm,xmm	3	3							flt→int	3	2
PUNPCKH/LBW/WD/DQ	xmm,m128	4	3				1			int		2
PUNPCKH/LQDQ	xmm,xmm	1	1							int	1	1
PUNPCKH/LQDQ	xmm, m128	2	1				1			int		1
PSHUFB h)	mm,mm	1	1			1				int	1	1
PSHUFB h)	mm,m64	2	1			1	1			int		1
PSHUFB h)	xmm,xmm	4	4							int	3	2
PSHUFB h)	xmm,m128	5	4				1			int		2
PSHUFW	mm,mm,i	1	1			1				int	1	1
PSHUFW	mm,m64,i	2	1			1	1			int		1
PSHUFD	xmm,xmm,i	2	2	x	x	1				flt→int	3	1
PSHUFD	xmm,m128,i	3	2	x	x	1	1			int		1
PSHUFL/HW	xmm,xmm,i	1	1			1				int	1	1
PSHUFL/HW	xmm, m128,i	2	1			1	1			int		1
PALIGNR h)	mm,mm,i	2	2	x	x	x				int	2	1
PALIGNR h)	mm,m64,i	2	2	x	x	x	1			int		1
PALIGNR h)	xmm,xmm,i	2	2	x	x	x				int	2	1



Merom

PALIGNR h)	xmm,m128,i	2	2	x	x	x	1			int		1
MASKMOVQ	mm,mm	4								int		2-5
MASKMOVDQU	xmm,xmm	10								int		6-10
PMOVMSKB	r32,(x)mm	1	1	1						int	2	1
PEXTRW	r32,mm,i	2	2							int	3	1
PEXTRW	r32,xmm,i	3	3							int	5	1
PINSRW	mm,r32,i	1	1				1			int	2	1
PINSRW	mm,m16,i	2	1				1	1		int		1
PINSRW	xmm,r32,i	3	3	x	x	x				int	6	1.5
PINSRW	xmm,m16,i	4	3	x	x	x	1			int		1.5
<b>Arithmetic instructions</b>												
PADD/SUB(U)(S)B/W/D	(x)mm,(x)mm	1	1	x		x				int	1	0.5
PADD/SUB(U)(S)B/W/D	(x)mm,m	1	1	x		x	1			int		1
PADDQ PSUBQ	(x)mm,(x)mm	2	2	x		x				int	2	1
PADDQ PSUBQ	(x)mm,m	2	2	x		x	1			int		1
PHADD(S)W												
PHSUB(S)W h)	mm,mm	5	5							int	5	4
PHADD(S)W												
PHSUB(S)W h)	mm,m64	6	5				1			int		4
PHADD(S)W												
PHSUB(S)W h)	xmm,xmm	7	7							int	6	4
PHADD(S)W												
PHSUB(S)W h)	xmm,m128	8	7				1			int		4
PHADDD PHSUBD h)	mm,mm	3	3							int	3	2
PHADDD PHSUBD h)	mm,m64	4	3				1			int		2
PHADDD PHSUBD h)	xmm,xmm	5	5							int	5	3
PHADDD PHSUBD h)	xmm,m128	6	5				1			int		3
PCMPEQ/GTB/W/D	(x)mm,(x)mm	1	1	x		x				int	1	0.5
PCMPEQ/GTB/W/D	(x)mm,m	1	1	x		x	1			int		1
PMULL/HW PMULHUW	(x)mm,(x)mm	1	1			1				int	3	1
PMULL/HW PMULHUW	(x)mm,m	1	1			1	1			int		1
PMULHRW h)	(x)mm,(x)mm	1	1			1				int	3	1
PMULHRW h)	(x)mm,m	1	1			1	1			int		1
PMULUDQ	(x)mm,(x)mm	1	1			1				int	3	1
PMULUDQ	(x)mm,m	1	1			1	1			int		1
PMADDWD	(x)mm,(x)mm	1	1			1				int	3	1
PMADDWD	(x)mm,m	1	1			1	1			int		1
PMADDUBSW h)	(x)mm,(x)mm	1	1			1				int	3	1
PMADDUBSW h)	(x)mm,m	1	1			1	1			int		1
PAVGB/W	(x)mm,(x)mm	1	1	x		x				int	1	0.5
PAVGB/W	(x)mm,m	1	1	x		x	1			int		1
PMIN/MAXUB/SW	(x)mm,(x)mm	1	1	x		x				int	1	0.5
PMIN/MAXUB/SW	(x)mm,m	1	1	x		x	1			int		1
PABSB PABSW PABSD	(x)mm,(x)mm	1	1	x		x				int	1	0.5
h)	(x)mm,m	1	1	x		x	1			int		1
PSIGNB PSIGNW	(x)mm,(x)mm	1	1	x		x				int	1	0.5
PSIGND h)	(x)mm,m	1	1	x		x	1			int		1
PSADBW	(x)mm,(x)mm	1	1			1				int	3	1
PSADBW	(x)mm,m	1	1			1	1			int		1

Merom

<b>Logic instructions</b>													
PAND(N) POR PXOR	(x)mm,(x)mm	1	1	x	x	x					int	1	0.33
PAND(N) POR PXOR	(x)mm,m	1	1	x	x	x	1				int		1
PSLL/RL/RAW/D/Q	mm,mm/i	1	1	1							int	1	1
PSLL/RL/RAW/D/Q	mm,m64	1	1	1			1				int		1
PSLL/RL/RAW/D/Q	xmm,i	1	1	1							int	1	1
PSLL/RL/RAW/D/Q	xmm,xmm	2	2	x	x						int	2	1
PSLL/RL/RAW/D/Q	xmm,m128	3	2	x	x		1				int		1
PSLL/RLDQ	xmm,i	2	2	x	x						int	2	1
<b>Other</b>													
EMMS		11	11	x	x	x					float		6

Notes:

- g) SSE3 instruction set.
- h) Supplementary SSE3 instruction set.
- k) MASM uses the name MOVD rather than MOVQ for this instruction even when moving 64 bits.

**Floating point XMM instructions**

Instruction	Operands	µops fused domain	µops unfused domain							Unit	Latency	Reciprocal throughput	
			p015	p0	p1	p5	p2	p3	p4				
<b>Move instructions</b>													
MOVAPS/D	xmm,xmm	1	1	x	x	x					int	1	0.33
MOVAPS/D	xmm,m128	1					1				int	2	1
MOVAPS/D	m128,xmm	1						1	1			3	1
MOVUPS/D	xmm,m128	4	2	1		1	2				int	2-4	2
MOVUPS/D	m128,xmm	9	4	x	x	x	1	2	2			3-4	4
MOVSS/D	xmm,xmm	1	1	x	x	x					int	1	0.33
MOVSS/D	xmm,m32/64	1					1				int	2	1
MOVSS/D	m32/64,xmm	1						1	1			3	1
MOVHPS/D MOVLPS/D	xmm,m64	2	1			1	1				int	3	1
MOVHPS/D	m64,xmm	2	1	1				1	1			5	1
MOVLPS/D	m64,xmm	1						1	1			3	1
MOVLHPS MOVHLPS	xmm,xmm	1	1	1							float	1	1
MOVMSKPS/D	r32,xmm	1	1	1							float	1	1
MOVNTPS/D	m128,xmm	1						1	1				2-3
SHUFPS	xmm,xmm,i	3	3		3						flt→int	3	2
SHUFPS	xmm,m128,i	4	3		3		1				flt→int		2
SHUFPD	xmm,xmm,i	1	1	1							float	1	1
SHUFPD	xmm,m128,i	2	1	1			1				float		1
MOVDDUP g)	xmm,xmm	1	1	1							int	1	1
MOVDDUP g)	xmm,m64	2	1	1			1				int		1
MOVSH/LDUP g)	xmm,xmm	1	1			1					int	1	1
MOVSH/LDUP g)	xmm,m128	2	1			1	1				int		1
UNPCKH/LPS	xmm,xmm	3	3		3						flt→int	3	2
UNPCKH/LPS	xmm,m128	4	3		3		1				int		2
UNPCKH/LPD	xmm,xmm	1	1	1							float	1	1

## Merom

UNPCKH/LPD	xmm,m128	2	1	1		1		float		1
<b>Conversion</b>										
CVTPD2PS	xmm,xmm	2	2					float	4	1
CVTPD2PS	xmm,m128	2	2			1		float		1
CVTSD2SS	xmm,xmm	2	2					float	4	1
CVTSD2SS	xmm,m64	2	2			1		float		1
CVTPS2PD	xmm,xmm	2	2	2				float	2	2
CVTPS2PD	xmm,m64	2	2	2		1		float		2
CVTSS2SD	xmm,xmm	2	2					float	2	2
CVTSS2SD	xmm,m32	2	2	2		1		float		2
CVTDQ2PS	xmm,xmm	1	1		1			float	3	1
CVTDQ2PS	xmm,m128	1	1		1	1		float		1
CVT(T) PS2DQ	xmm,xmm	1	1		1			float	3	1
CVT(T) PS2DQ	xmm,m128	1	1		1	1		float		1
CVTDQ2PD	xmm,xmm	2	2	1	1			float	4	1
CVTDQ2PD	xmm,m64	3	2			1		float		1
CVT(T)PD2DQ	xmm,xmm	2	2					float	4	1
CVT(T)PD2DQ	xmm,m128	2	2			1		float		1
CVTPI2PS	xmm,mm	1	1		1			float	3	3
CVTPI2PS	xmm,m64	1	1		1	1		float		3
CVT(T)PS2PI	mm,xmm	1	1		1			float	3	1
CVT(T)PS2PI	mm,m128	1	1		1	1		float		1
CVTPI2PD	xmm,mm	2	2	1	1			float	4	1
CVTPI2PD	xmm,m64	2	2	1	1	1		float		1
CVT(T) PD2PI	mm,xmm	2	2	1	1			float	4	1
CVT(T) PD2PI	mm,m128	2	2	1	1	1		float		1
CVTSI2SS	xmm,r32	1	1		1			float	4	3
CVTSI2SS	xmm,m32	1	1		1	1		float		3
CVT(T)SS2SI	r32,xmm	1	1		1			float	3	1
CVT(T)SS2SI	r32,m32	1	1		1	1		float		1
CVTSI2SD	xmm,r32	2	2	1	1			float	4	3
CVTSI2SD	xmm,m32	2	1		1	1		float		3
CVT(T)SD2SI	r32,xmm	1	1		1			float	3	1
CVT(T)SD2SI	r32,m64	1	1		1	1		float		1
<b>Arithmetic</b>										
ADDSS/D SUBSS/D	xmm,xmm	1	1		1			float	3	1
ADDSS/D SUBSS/D	xmm,m32/64	1	1		1	1		float		1
ADDPS/D SUBPS/D	xmm,xmm	1	1		1			float	3	1
ADDPS/D SUBPS/D	xmm,m128	1	1		1	1		float		1
ADDSUBPS/D g)	xmm,xmm	1	1		1			float	3	1
ADDSUBPS/D g)	xmm,m128	1	1		1	1		float		1
HADDPS HSUBPS g)	xmm,xmm	6	6					float	9	3
HADDPS HSUBPS g)	xmm,m128	7	6			1		float		3
HADDPD HSUBPD g)	xmm,xmm	3	3					float	5	2
HADDPD HSUBPD g)	xmm,m128	4	3			1		float		2
MULSS	xmm,xmm	1	1	1				float	4	1
MULSS	xmm,m32	1	1	1		1		float		1
MULSD	xmm,xmm	1	1	1				float	5	1

Merom

MULSD	xmm,m64	1	1	1			1			float		1
MULPS	xmm,xmm	1	1	1						float	4	1
MULPS	xmm,m128	1	1	1			1			float		1
MULPD	xmm,xmm	1	1	1						float	5	1
MULPD	xmm,m128	1	1	1			1			float		1
DIVSS	xmm,xmm	1	1	1						float	6-18 d)	5-17 d)
DIVSS	xmm,m32	1	1	1			1			float		5-17 d)
DIVSD	xmm,xmm	1	1	1						float	6-32 d)	5-31 d)
DIVSD	xmm,m64	1	1	1			1			float		5-31 d)
DIVPS	xmm,xmm	1	1	1						float	6-18 d)	5-17 d)
DIVPS	xmm,m128	1	1	1			1			float		5-17 d)
DIVPD	xmm,xmm	1	1	1						float	6-32 d)	5-31 d)
DIVPD	xmm,m128	1	1	1			1			float		5-31 d)
RCPSS/PS	xmm,xmm	1	1		1					float	3	2
RCPSS/PS	xmm,m	1	1		1		1			float		2
CMPccSS/D	xmm,xmm	1	1		1					float	3	1
CMPccSS/D	xmm,m32/64	1	1		1		1			float		1
CMPccPS/D	xmm,xmm	1	1		1					float	3	1
CMPccPS/D	xmm,m128	1	1		1		1			float		1
COMISS/D UCOMISS/D	xmm,xmm	1	1		1					float	3	1
COMISS/D UCOMISS/D	xmm,m32/64	1	1		1		1			float		1
MAXSS/D MINSS/D	xmm,xmm	1	1		1					float	3	1
MAXSS/D MINSS/D	xmm,m32/64	1	1		1		1			float		1
MAXPS/D MINPS/D	xmm,xmm	1	1		1					float	3	1
MAXPS/D MINPS/D	xmm,m128	1	1		1		1			float		1
<b>Math</b>												
SQRTSS/PS	xmm,xmm	1	1	1						float	6-29	6-29
SQRTSS/PS	xmm,m	2	1	1			1			float		6-29
SQRTSD/PD	xmm,xmm	1	1	1						float	6-58	6-58
SQRTSD/PD	xmm,m	2	1	1			1			float		6-58
RSQRTSS/PS	xmm,xmm	1	1		1					float	3	2
RSQRTSS/PS	xmm,m	1	1		1		1			float		2
<b>Logic</b>												
AND/ANDN/OR/XORPS/D	xmm,xmm	1	1	x	x	x				int	1	0.33
AND/ANDN/OR/XORPS/D	xmm,m128	1	1	x	x	x	1			int		1
<b>Other</b>												
LDMXCSR	m32	14	13				1					42
STMXCSR	m32	6	4					1	1			19
FXSAVE	m4096	141									145	145
FXRSTOR	m4096	119									164	164

**Notes:**

- d) Round divisors give low values.
- g) SSE3 instruction set.

## Intel Core 2 (Wolfdale, 45nm)

### List of instruction timings and $\mu$ op breakdown

*Explanation of column headings:*

**Operands:** i = immediate data, r = register, mm = 64 bit mmx register, xmm = 128 bit xmm register, (x)mm = mmx or xmm register, sr = segment register, m = memory, m32 = 32-bit memory operand, etc.

**$\mu$ ops fused domain:** The number of  $\mu$ ops at the decode, rename, allocate and retirement stages in the pipeline. Fused  $\mu$ ops count as one.

**$\mu$ ops unfused domain:** The number of  $\mu$ ops for each execution port. Fused  $\mu$ ops count as two. Fused macro-ops count as one. The instruction has  $\mu$ op fusion if the sum of the numbers listed under p015 + p2 + p3 + p4 exceeds the number listed under  $\mu$ ops fused domain. An x under p0, p1 or p5 means that at least one of the  $\mu$ ops listed under p015 can optionally go to this port. For example, a 1 under p015 and an x under p0 and p5 means one  $\mu$ op which can go to either port 0 or port 5, whichever is vacant first. A value listed under p015 but nothing under p0, p1 and p5 means that it is not known which of the three ports these  $\mu$ ops go to.

**p015:** The total number of  $\mu$ ops going to port 0, 1 and 5.

**p0:** The number of  $\mu$ ops going to port 0 (execution units).

**p1:** The number of  $\mu$ ops going to port 1 (execution units).

**p5:** The number of  $\mu$ ops going to port 5 (execution units).

**p2:** The number of  $\mu$ ops going to port 2 (memory read).

**p3:** The number of  $\mu$ ops going to port 3 (memory write address).

**p4:** The number of  $\mu$ ops going to port 4 (memory write data).

**Unit:** Tells which execution unit cluster is used. An additional delay of 1 clock cycle is generated if a register written by a  $\mu$ op in the integer unit (int) is read by a  $\mu$ op in the floating point unit (float) or vice versa. flt→int means that an instruction with multiple  $\mu$ ops receive the input in the float unit and delivers the output in the int unit. Delays for moving data between different units are included under latency when they are unavoidable. For example, movd eax,xmm0 has an extra 1 clock delay for moving from the XMM-integer unit to the general purpose integer unit. This is included under latency because it occurs regardless of which instruction comes next. Nothing listed under unit means that additional delays are either unlikely to occur or unavoidable and therefore included in the latency figure.

**Latency:** This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays very much, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay. The time unit used is core clock cycles, not the reference clock cycles given by the time stamp counter.

**Reciprocal throughput:** The average number of core clock cycles per instruction for a series of independent instructions of the same kind in the same thread.

### Integer instructions

Instruction	Operands	$\mu$ ops fused domain	$\mu$ ops unfused domain							Unit	Latency	Reciprocal throughput
			p015	p0	p1	p5	p2	p3	p4			

Wolfdale

Move instructions												
MOV	r,r/i	1	1	x	x	x					1	0.33
MOV a)	r,m	1					1				2	1
MOV a)	m,r	1						1	1		3	1
MOV	m,i	1						1	1		3	1
MOV	r,sr	1					1					1
MOV	m,sr	2					1	1	1			1
MOV	sr,r	8	4	x	x	x	4					16
MOV	sr,m	8	3	x		x	5					16
MOVNTI	m,r	2						1	1			2
MOVSX MOVZX												
MOVSXD	r,r	1	1	x	x	x					1	0.33
MOVSX MOVZX	r16/32,m	1					1					1
MOVSX MOVSXD	r64,m	2	1	x	x	x	1					1
CMOVcc	r,r	2	2	x	x	x					2	1
CMOVcc	r,m	2	2	x	x	x	1					
XCHG	r,r	3	3	x	x	x					2	2
XCHG	r,m	7					1	1	1	high b)		
XLAT		2	1				1				4	1
PUSH	r	1						1	1		3	1
PUSH	i	1						1	1			1
PUSH	m	2					1	1	1			1
PUSH	sr	2	1					1	1			1
PUSHF(D/Q)		17	15	x	x	x		1	1			7
PUSHA(D) i)		18	9					1	8			8
POP	r	1					1				2	1
POP	(E/R)SP	4	3				1					
POP	m	2					1	1	1			1.5
POP	sr	10	9				1					17
POPF(D/Q)		24	23	x	x	x	1				20	
POPA(D) i)		10	2				8					7
LAHF SAHF		1	1	x	x	x					1	0.33
SALC i)		2	2	x	x	x					4	1
LEA a)	r,m	1	1	1							1	1
BSWAP	r	2	2	1		1					4	1
LDS LES LFS LGS LSS	m	11	11				1					17
PREFETCHNTA	m	1					1					1
PREFETCHT0/1/2	m	1					1					1
LFENCE		2						1	1			8
MFENCE		2						1	1			6
SFENCE		2						1	1			9
CLFLUSH	m8	4	2	1		1		1	1		120	90
IN												
OUT												
Arithmetic instructions												
ADD SUB	r,r/i	1	1	x	x	x					1	0.33
ADD SUB	r,m	1	1	x	x	x	1					1
ADD SUB	m,r/i	2	1	x	x	x	1	1	1		6	1
ADC SBB	r,r/i	2	2	x	x	x					2	2

Wolfdale

ADC SBB	r,m	2	2	x	x	x	1			2	2
ADC SBB	m,r/i	4	3	x	x	x	1	1	1	7	
CMP	r,r/i	1	1	x	x	x				1	0.33
CMP	m,r/i	1	1	x	x	x	1			1	1
INC DEC NEG NOT	r	1	1	x	x	x				1	0.33
INC DEC NEG NOT	m	3	1	x	x	x	1	1	1	6	1
AAA AAS DAA DAS i)		1	1			1					1
AAD i)		3	3	x	x	x					1
AAM i)		5	5	x	x	x				17	
MUL IMUL	r8	1	1			1				3	1
MUL IMUL	r16	3	3	x	x	x				5	1.5
MUL IMUL	r32	3	3	x	x	x				5	1.5
MUL IMUL	r64	3	3	x	x	x				7	4
IMUL	r16,r16	1	1			1				3	1
IMUL	r32,r32	1	1			1				3	1
IMUL	r64,r64	1	1	1						5	2
IMUL	r16,r16,i	1	1			1				3	1
IMUL	r32,r32,i	1	1			1				3	1
IMUL	r64,r64,i	1	1	1						5	2
MUL IMUL	m8	1	1			1		1		3	1
MUL IMUL	m16	3	3	x	x	x	1			5	1.5
MUL IMUL	m32	3	3	x	x	x	1			5	1.5
MUL IMUL	m64	3	2	2				1		7	4
IMUL	r16,m16	1	1			1		1		3	1
IMUL	r32,m32	1	1			1		1		3	1
IMUL	r64,m64	1	1	1				1		5	2
IMUL	r16,m16,i	1	1			1		1			2
IMUL	r32,m32,i	1	1			1		1			1
IMUL	r64,m64,i	1	1	1				1			2
DIV IDIV	r8	4	4	1	2	1				9-18 c)	
DIV IDIV	r16	7	7	x	x	x				14-22 c)	
DIV IDIV	r32	7	7	2	3	2				14-23 c)	
DIV	r64	32-38	32-38	9	10	13				18-57 c)	
IDIV	r64	56-62	56-62	x	x	x				34-88 c)	
DIV IDIV	m8	4	3	1	2			1		9-18	
DIV IDIV	m16	7	6	2	3	2		1		14-22 c)	
DIV IDIV	m32	7	6	x	x	x		1		14-23 c)	
DIV	m64	32	31	x	x	x		1		34-88 c)	
IDIV	m64	56	55	x	x	x		1		39-72 c)	
CBW CWDE CDQE		1	1	x	x	x				1	
CWD CDQ CQO		1	1	x		x				1	
<b>Logic instructions</b>											
AND OR XOR	r,r/i	1	1	x	x	x				1	0.33
AND OR XOR	r,m	1	1	x	x	x	1				1
AND OR XOR	m,r/i	2	1	x	x	x	1	1	1	6	1
TEST	r,r/i	1	1	x	x	x				1	0.33
TEST	m,r/i	1	1	x	x	x	1				1
SHR SHL SAR	r,i/cl	1	1	x		x				1	0.5
SHR SHL SAR	m,i/cl	3	2	x		x	1	1	1	6	1

Wolfdale

ROR ROL	r,i/cl	1	1	x		x					1	1
ROR ROL	m,i/cl	3	2	x		x	1	1	1		6	1
RCR RCL	r,1	2	2	x	x	x					2	2
RCR	r8,i/cl	9	9	x	x	x					12	
RCL	r8,i/cl	8	8	x	x	x					11	
RCR RCL	r16/32/64,i/cl	6	6	x	x	x					11	
RCR RCL	m,1	4	3	x	x	x	1	1	1		7	
RCR	m8,i/cl	12	9	x	x	x	1	1	1		14	
RCL	m8,i/cl	11	8	x	x	x	1	1	1		13	
RCR RCL	m16/32/64,i/cl	10	7	x	x	x	1	1	1		13	
SHLD SHRD	r,r,i/cl	2	2	x	x	x					2	1
SHLD SHRD	m,r,i/cl	3	2	x	x	x	1	1	1		7	
BT	r,r/i	1	1	x	x	x					1	1
BT	m,r	9	8	x	x	x	1					4
BT	m,i	3	2	x	x	x	1					1
BTR BTS BTC	r,r/i	1	1	x	x	x					1	
BTR BTS BTC	m,r	10	7	x	x	x	1	1	1		5	
BTR BTS BTC	m,i	3	1	x	x	x	1	1	1		6	
BSF BSR	r,r	2	2	x	1	x					2	1
BSF BSR	r,m	2	2	x	1	x	1					1
SETcc	r	1	1	x	x	x					1	1
SETcc	m	2	1	x	x	x		1	1			1
CLC STC CMC		1	1	x	x	x					1	0.33
CLD		6	6	x	x	x						3
STD		6	6	x	x	x						14
<b>Control transfer instructions</b>												
JMP	short/near	1	1				1				0	1-2
JMP i)	far	30	30									76
JMP	r	1	1				1				0	1-2
JMP	m(near)	1	1				1	1			0	1-2
JMP	m(far)	31	29					2				68
Conditional jump	short/near	1	1				1				0	1
Fused compare/test and branch	e,i)	1	1				1				0	1
J(E/R)CXZ	short	2	2	x	x	1						1-2
LOOP	short	11	11	x	x	x						5
LOOP(N)E	short	11	11	x	x	x						5
CALL	near	3	2	x	x	x			1	1		2
CALL i)	far	43	43									75
CALL	r	3	2						1	1		2
CALL	m(near)	4	3					1	1	1		2
CALL	m(far)	44	42					2				75
RETN		1					1	1				2
RETN	i	3	1				1	1				2
RETF		32	30					2				78
RETF	i	32	30					2				78
BOUND i)	r,m	15	13					2				8
INTO i)		5	5									3
<b>String instructions</b>												



Wolfdale

LODS		3	2				1					1
REP LODS		4+7n-14+6n									1+5n-21+3n	
STOS		4	2					1	1			1
REP STOS		8+5n-20+1.2n									7+2n-0.55n	
MOVS		8	5									
		1	1	1		5						
REP MOVS		7+7n-13+n									1+3n-0.63n	
SCAS		4	3				1					1
REP(N)E SCAS		7+8n-17+7n									3+8n-23+6n	
CMPS		7	5				2					3
REP(N)E CMPS		7+10n-7+9n									2+7n-22+5n	
<b>Other</b>												
NOP (90)		1	1	x	x	x						0.33
Long NOP (0F 1F)		1	1	x	x	x						1
PAUSE		3	3	x	x	x						8
ENTER	i,0	12	10						1	1		8
ENTER	a,b											
LEAVE		3	2				1					
CPUID		53-117									53-211	
RDTSC		13									32	
RDPMSR		23									54	

**Notes:**

- a) Applies to all addressing modes
- b) Has an implicit LOCK prefix.
- c) Low values are for small results, high values for high results. The reciprocal throughput is only slightly less than the latency.
- e) See manual 3: "The microarchitecture of Intel, AMD and VIA CPUs" for restrictions on macro-op fusion.
- i) Not available in 64 bit mode.

**Floating point x87 instructions**

Instruction	Operands	µops fused domain	µops unfused domain								Unit	Latency	Reciprocal throughput
			p015	p0	p1	p5	p2	p3	p4				
<b>Move instructions</b>													
FLD	r	1	1	1							float	1	1
FLD	m32/64	1	1					1			float	3	1
FLD	m80	4	2	2				2			float	4	3
FBLD	m80	40	38	x	x	x		2			float	45	20
FST(P)	r	1	1	1							float	1	1
FST(P)	m32/m64	1							1	1	float	3	1
FSTP	m80	7	3	x	x	x		2	2		float	4	5
FBSTP	m80	171	167	x	x	x		2	2		float	164	166
FXCH	r	1	0 f)								float	0	1
FILD	m	1	1		1			1			float	6	1
FIST	m	2	1		1				1	1	float	6	1
FISTP	m	3	1		1				1	1	float	6	1

Wolfdale

FISTTP g)	m	3	1		1			1	1	float	6	1
FLDZ		1	1	1						float		1
FLD1		2	2	1	1					float		2
FLDPI FLDL2E etc.		2	2		2					float		2
FCMOVcc	r	2	2	2						float	2	2
FNSTSW	AX	1	1	1						float		1
FNSTSW	m16	2	1	1				1	1	float		2
FLDCW	m16	2	1				1			float		10
FNSTCW	m16	3	1			1		1	1	float		8
FINCSTP FDECSTP		1	1	1						float	1	1
FFREE(P)	r	2	2	x	x	x				float		2
FNSAVE	m	141	95	x	x	x	7	23	23	float		142
FRSTOR	m	78	51	x	x	x	27			float		177
<b>Arithmetic instructions</b>												
FADD(P) FSUB(R)(P)	r	1	1		1					float	3	1
FADD(P) FSUB(R)(P)	m	1	1		1		1			float		1
FMUL(P)	r	1	1	1						float	5	2
FMUL(P)	m	1	1	1			1			float		2
FDIV(R)(P)	r	1	1	1						float	6-21 d)	5-20 d)
FDIV(R)(P)	m	1	1	1			1			float	6-21 d)	5-20 d)
FABS		1	1	1						float	1	1
FCHS		1	1	1						float	1	1
FCOM(P) FUCOM	r	1	1		1					float		1
FCOM(P) FUCOM	m	1	1		1		1			float		1
FCOMPP FUCOMPP		2	2	1	1					float		
FCOMI(P) FUCOMI(P)	r	1	1		1					float		1
FIADD FISUB(R)	m	2	2		2		1			float	3	2
FIMUL	m	2	2	1	1		1			float	5	2
FIDIV(R)	m	2	2	1	1		1			float	6-21	5-20 d)
FICOM(P)	m	2	2		2		1			float		2
FTST		1	1		1					float		1
FXAM		1	1		1					float		1
FPREM		26-29			x	x	x			float	13-40	
FPREM1		28-35			x	x	x			float	18-41	
FRNDINT		17-19			x	x	x			float	10-22	
<b>Math</b>												
FSCALE		28	28		x	x	x			float	43	
FEXTRACT		53-84			x	x	x			float	~170	
FSQRT		1	1	1						float	6-20	
FSIN		18-85			x	x	x			float	32-85	
FCOS		76-100			x	x	x			float	70-100	
FSINCOS		18-105			x	x	x			float	38-107	
F2XM1		19	19		x	x	x			float	45	
FYL2X FYL2XP1		57-65			x	x	x			float	50-100	
FPTAN		19-100			x	x	x			float	40-130	
FPATAN		23-87			x	x	x			float	55-130	

Wolfdale

Other												
FNOP		1	1	1							float	1
WAIT		2	2	x	x	x					float	1
FNCLEX		4	4		x	x					float	15
FNINIT		15	15	x	x	x					float	63

Notes:

- d) Round divisors or low precision give low values.
- f) Resolved by register renaming. Generates no  $\mu$ ops in the unfused domain.
- g) SSE3 instruction set.

Integer MMX and XMM instructions

Instruction	Operands	$\mu$ ops fused domain	$\mu$ ops unfused domain							Unit	Latency	Reciprocal throughput	
			p015	p0	p1	p5	p2	p3	p4				
<b>Move instructions</b>													
MOVD k)	r32/64,(x)mm	1	1	x	x	x					int	2	0.33
MOVD k)	m32/64,(x)mm	1							1	1		3	1
MOVD k)	(x)mm,r32/64	1	1	x		x					int	2	0.5
MOVD k)	(x)mm,m32/64	1					1				int	2	1
MOVQ	(x)mm, (x)mm	1	1	x	x	x					int	1	0.33
MOVQ	(x)mm,m64	1					1				int	2	1
MOVQ	m64, (x)mm	1						1	1			3	1
MOVDQA	xmm, xmm	1	1	x	x	x					int	1	0.33
MOVDQA	xmm, m128	1					1				int	2	1
MOVDQA	m128, xmm	1						1	1			3	1
MOVDQU	m128, xmm	9	4	x	x	x	1	2	2			3-8	4
MOVDQU	xmm, m128	4	2	x		x	2				int	2-8	2
LDDQU g)	xmm, m128	4	2	x		x	2				int	2-8	2
MOVDQ2Q	mm, xmm	1	1	x	x	x					int	1	0.33
MOVQ2DQ	xmm,mm	1	1	x	x	x					int	1	0.33
MOVNTQ	m64,mm	1						1	1				2
MOVNTDQ	m128,xmm	1						1	1				2
MOVNTDQA j)	xmm, m128	1					1					2	1
PACKSSWB/DW PACKUSWB	mm,mm	1	1	1							int	1	1
PACKSSWB/DW PACKUSWB	mm,m64	1	1	1			1				int		1
PACKSSWB/DW PACKUSWB	xmm,xmm	1	1			1					int	1	1
PACKSSWB/DW PACKUSWB	xmm,m128	1	1			1	1				int		1
PACKUSDW j)	xmm,xmm	1	1			1					int	1	1
PACKUSDW j)	xmm,m	1	1			1	1				int		1
PUNPCKH/LBW/WD/DQ	mm,mm	1	1	1							int	1	1
PUNPCKH/LBW/WD/DQ	mm,m64	1	1	1			1				int		1
PUNPCKH/LBW/WD/DQ	xmm,xmm	1	1			1					int	1	1
PUNPCKH/LBW/WD/DQ	xmm,m128	1	1			1	1				int		1
PUNPCKH/LQDQ	xmm,xmm	1	1			1					int	1	1
PUNPCKH/LQDQ	xmm, m128	2	1			1	1				int		1

Wolfdale

PMOVXSX/ZXBW j)	xmm,xmm	1	1			1				int	1	1
PMOVXSX/ZXBW j)	xmm,m64	1	1			1	1			int		1
PMOVXSX/ZXBD j)	xmm,xmm	1	1			1				int	1	1
PMOVXSX/ZXBD j)	xmm,m32	1	1			1	1			int		1
PMOVXSX/ZXBQ j)	xmm,xmm	1	1			1				int	1	1
PMOVXSX/ZXBQ j)	xmm,m16	1	1			1	1			int		1
PMOVXSX/ZXWD j)	xmm,xmm	1	1			1				int	1	1
PMOVXSX/ZXWD j)	xmm,m64	1	1			1	1			int		1
PMOVXSX/ZXWQ j)	xmm,xmm	1	1			1				int	1	1
PMOVXSX/ZXWQ j)	xmm,m32	1	1			1	1			int		1
PMOVXSX/ZXDQ j)	xmm,xmm	1	1			1				int	1	1
PMOVXSX/ZXDQ j)	xmm,m64	1	1			1	1			int		1
PSHUFB h)	mm,mm	1	1			1				int	1	1
PSHUFB h)	mm,m64	2	1			1	1			int		1
PSHUFB h)	xmm,xmm	1	1			1				int	1	1
PSHUFB h)	xmm,m128	1	1			1	1			int		1
PSHUFW	mm,mm,i	1	1			1				int	1	1
PSHUFW	mm,m64,i	2	1			1	1			int		1
PSHUFD	xmm,xmm,i	1	1			1				int	1	1
PSHUFD	xmm,m128,i	2	1			1	1			int		1
PSHUFL/HW	xmm,xmm,i	1	1			1				int	1	1
PSHUFL/HW	xmm, m128,i	2	1			1	1			int		1
PALIGNR h)	mm,mm,i	2	2			2				int	2	1
PALIGNR h)	mm,m64,i	3	2			3	1			int		1
PALIGNR h)	xmm,xmm,i	1	1			1				int	1	1
PALIGNR h)	xmm,m128,i	1	1			1	1			int		1
PBLENDVB j)	x,x,xmm0	2	2			2				int	2	2
PBLENDVB j)	x,m,xmm0	2	2			2	1			int		2
PBLENDW j)	xmm,xmm,i	1	1			1				int	1	1
PBLENDW j)	xmm,m,i	1	1			1	1			int		1
MASKMOVQ	mm,mm	4	1	1			1	1	1	int		2-5
MASKMOVDQU	xmm,xmm	10	4	1		3	2	2	3	int		6-10
PMOVMSKB	r32,(x)mm	1	1	1						int	2	1
PEXTRB j)	r32,xmm,i	2	2	x	x	x				int	3	1
PEXTRB j)	m8,xmm,i	2	2	x	x	x				int	3	1
PEXTRW	r32,(x)mm,i	2	2	x	x	x	1			int	3	1
PEXTRW j)	m16,(x)mm,i	2	2			1		1	1	int		1
PEXTRD j)	r32,xmm,i	2	2	x	x	x				int	3	1
PEXTRD j)	m32,xmm,i	2	1			1		1	1	int		1
PEXTRQ j,m)	r64,xmm,i	2	2	x	x	x				int	3	1
PEXTRQ j,m)	m64,xmm,i	2	1			1		1	1	int		1
PINSRB j)	xmm,r32,i	1	1			1				int	1	1
PINSRB j)	xmm,m8,i	2	1			1	1			int		1
PINSRW	(x)mm,r32,i	1	1			1				int	2	1
PINSRW	(x)mm,m16,i	2	1			1	1			int		1
PINSRD j)	xmm,r32,i	1	1			1				int	1	1
PINSRD j)	xmm,m32,i	2	1			1	1			int		1
PINSRQ j,m)	xmm,r64,i	1	1			1				int	1	1
PINSRQ j,m)	xmm,m64,i	2	1			1	1			int		1

Wolfdale

Arithmetic instructions											
PADD/SUB(U)(S)B/W/D	(x)mm, (x)mm	1	1	x	x				int	1	0.5
PADD/SUB(U)(S)B/W/D	(x)mm,m	1	1	x	x	1			int		1
PADDQ PSUBQ	(x)mm, (x)mm	2	2	x	x				int	2	1
PADDQ PSUBQ	(x)mm,m	2	2	x	x	1			int		1
PHADD(S)W											
PHSUB(S)W h)	(x)mm, (x)mm	3	3	1	2				int	3	2
PHADD(S)W											
PHSUB(S)W h)	(x)mm,m64	4	3	1	2	1			int		2
PHADD PHSUBD h)	(x)mm, (x)mm	3	3	1	2				int	3	2
PHADD PHSUBD h)	(x)mm,m64	4	3	1	2	1			int		2
PCMPEQ/GTB/W/D	(x)mm,(x)mm	1	1	x	x				int	1	0.5
PCMPEQ/GTB/W/D	(x)mm,m	1	1	x	x	1			int		1
PCMPEQQ j)	xmm,xmm	1	1			1			int	1	1
PCMPEQQ j)	xmm,m128	1	1			1	1		int		1
PMULL/HW PMULHUW	(x)mm,(x)mm	1	1		1				int	3	1
PMULL/HW PMULHUW	(x)mm,m	1	1		1		1		int		1
PMULHRSW h)	(x)mm,(x)mm	1	1		1				int	3	1
PMULHRSW h)	(x)mm,m	1	1		1		1		int		1
PMULLD j)	xmm,xmm	4	4		2	2			int	5	2
PMULLD j)	xmm,m128	6	5	1	2	2	1		int	5	4
PMULDQ j)	xmm,xmm	1	1		1				int	3	1
PMULDQ j)	xmm,m128	1	1		1		1		int		1
PMULUDQ	(x)mm,(x)mm	1	1		1				int	3	1
PMULUDQ	(x)mm,m	1	1		1		1		int		1
PMADDWD	(x)mm,(x)mm	1	1		1				int	3	1
PMADDWD	(x)mm,m	1	1		1		1		int		1
PMADDUBSW h)	(x)mm,(x)mm	1	1		1				int	3	1
PMADDUBSW h)	(x)mm,m	1	1		1		1		int		1
PAVGB/W	(x)mm,(x)mm	1	1	x	x				int	1	0.5
PAVGB/W	(x)mm,m	1	1	x	x	1			int		1
PMIN/MAXSB j)	xmm,xmm	1	1	1					int	1	1
PMIN/MAXSB j)	xmm,m128	1	1	1			1		int		1
PMIN/MAXUB	(x)mm,(x)mm	1	1	x	x				int	1	0.5
PMIN/MAXUB	(x)mm,m	1	1	x	x	1			int		1
PMIN/MAXSW	(x)mm,(x)mm	1	1	x	x				int	1	0.5
PMIN/MAXSW	(x)mm,m	1	1	x	x	1			int		1
PMIN/MAXUW j)	xmm,xmm	1	1	1					int	1	1
PMIN/MAXUW j)	xmm,m	1	1				1		int		1
PMIN/MAXSD j)	xmm,xmm	1	1	1					int	1	1
PMIN/MAXSD j)	xmm,m128	1	1	1			1		int		1
PMIN/MAXUD j)	xmm,xmm	1	1	1					int	1	1
PMIN/MAXUD j)	xmm,m128	1	1	1			1		int		1
PHMINPOSUW j)	xmm,xmm	4	4			4			int	4	4
PHMINPOSUW j)	xmm,m128	4	4			4	1		int		4
PABSB PABSW PABSD h)	(x)mm,(x)mm	1	1	x	x				int	1	0.5
PABSB PABSW PABSD h)	(x)mm,m	1	1	x	x	1			int		1
PSIGNB PSIGNW											
PSIGND h)	(x)mm,(x)mm	1	1	x	x				int	1	0.5

Wolfdale

PSIGNB PSIGNW PSIGND h)	(x)mm,m	1	1	x		x	1			int		1
PSADBW	(x)mm,(x)mm	1	1		1					int	3	1
PSADBW	(x)mm,m	1	1		1		1			int		1
MPSADBW j)	xmm,xmm,i	3	3		1	2				int	5	2
MPSADBW j)	xmm,m,i	4	3		1	2	1			int		2
<b>Logic instructions</b>												
PAND(N) POR PXOR	(x)mm,(x)mm	1	1	x	x	x				int	1	0.33
PAND(N) POR PXOR	(x)mm,m	1	1	x	x	x	1			int		1
PTEST j)	xmm,xmm	2	2	1	x	x				int	1	1
PTEST j)	xmm,m128	2	2	1	x	x	1			int		1
PSLL/RL/RAW/D/Q	mm,mm/i	1	1	1						int	1	1
PSLL/RL/RAW/D/Q	mm,m64	1	1	1			1			int		1
PSLL/RL/RAW/D/Q	xmm,i	1	1	1						int	1	1
PSLL/RL/RAW/D/Q	xmm,xmm	2	2	x		x				int	2	1
PSLL/RL/RAW/D/Q	xmm,m128	3	2	x		x	1			int		1
PSLL/RLDQ	xmm,i	1	1	x		x				int	1	1
<b>Other</b>												
EMMS		11	11	x	x	x				float		6

**Notes:**

- g) SSE3 instruction set.
- h) Supplementary SSE3 instruction set.
- j) SSE4.1 instruction set
- k) MASM uses the name MOVD rather than MOVQ for this instruction even when moving 64 bits
- m) Only available in 64 bit mode

**Floating point XMM instructions**

Instruction	Operands	µops fused domain	µops unfused domain								Unit	Latency	Reciprocal throughput
			p015	p0	p1	p5	p2	p3	p4				
<b>Move instructions</b>													
MOVAPS/D	xmm,xmm	1	1	x	x	x				int	1	0.33	
MOVAPS/D	xmm,m128	1					1			int	2	1	
MOVAPS/D	m128,xmm	1						1	1		3	1	
MOVUPS/D	xmm,m128	4	2	1		1	2			int	2-4	2	
MOVUPS/D	m128,xmm	9	4	x	x	x	1	2	2		3-4	4	
MOVSS/D	xmm,xmm	1	1	x	x	x				int	1	0.33	
MOVSS/D	xmm,m32/64	1					1			int	2	1	
MOVSS/D	m32/64,xmm	1						1	1		3	1	
MOVHPS/D MOVLPS/D	xmm,m64	2	1			1	1			int	3	1	
MOVHPS/D	m64,xmm	2	1	1				1	1		5	1	
MOVLPS/D	m64,xmm	1						1	1		3	1	
MOVLHPS MOVHLPS	xmm,xmm	1	1	1						float	1	1	
MOVMSKPS/D	r32,xmm	1	1	1						float	1	1	
MOVNTPS/D	m128,xmm	1						1	1			2-3	
SHUFPS	xmm,xmm,i	1	1			1				int	1	1	

Wolfdale

SHUFPS	xmm,m128,i	2	1			1	1			int		1
SHUFPD	xmm,xmm,i	1	1	1						float	1	1
SHUFPD	xmm,m128,i	2	1	1						float		1
BLENDPS/PD j)	xmm,xmm,i	1	1			1				int	1	1
BLENDPS/PD j)	xmm,m128,i	1	1			1	1			int		1
BLENDVPS/PD j)	xmm,xmm,xmm0	2	2			2				int	2	2
BLENDVPS/PD j)	xmm,m,xmm0	2	2			2	1			int		2
MOVDDUP g)	xmm,xmm	1	1	1						int	1	1
MOVDDUP g)	xmm,m64	2	1	1						int		1
MOVSH/LDUP g)	xmm,xmm	1	1			1				int	1	1
MOVSH/LDUP g)	xmm,m128	2	1			1	1			int		1
UNPCKH/LPS	xmm,xmm	1	1			1				int	1	1
UNPCKH/LPS	xmm,m128	1	1			1	1			int		1
UNPCKH/LPD	xmm,xmm	1	1	1						float	1	1
UNPCKH/LPD	xmm,m128	2	1	1						float		1
EXTRACTPS j)	r32,xmm,i	2	2	x	x	x				int	4	1
EXTRACTPS j)	m32,xmm,i	2	1			1		1	1	int		1
INSERTPS j)	xmm,xmm,i	1	1			1				int	1	1
INSERTPS j)	xmm,m32,i	2	1			1	1			int		1
<b>Conversion</b>												
CVTPD2PS	xmm,xmm	2	2	1	1					float	4	1
CVTPD2PS	xmm,m128	2	2	1	1				1	float		1
CVTSD2SS	xmm,xmm	2	2	1	1					float	4	1
CVTSD2SS	xmm,m64	2	2	1	1				1	float		1
CVTPS2PD	xmm,xmm	2	2	2						float	2	2
CVTPS2PD	xmm,m64	2	2	2					1	float		2
CVTSS2SD	xmm,xmm	2	2	2						float	2	2
CVTSS2SD	xmm,m32	2	2	2					1	float		2
CVTDQ2PS	xmm,xmm	1	1		1					float	3	1
CVTDQ2PS	xmm,m128	1	1		1				1	float		1
CVT(T) PS2DQ	xmm,xmm	1	1		1					float	3	1
CVT(T) PS2DQ	xmm,m128	1	1		1				1	float		1
CVTDQ2PD	xmm,xmm	2	2	1	1					float	4	1
CVTDQ2PD	xmm,m64	2	2	1	1				1	float		1
CVT(T)PD2DQ	xmm,xmm	2	2	1	1					float	4	1
CVT(T)PD2DQ	xmm,m128	2	2	1	1				1	float		1
CVTPI2PS	xmm,mm	1	1		1					float	3	3
CVTPI2PS	xmm,m64	1	1		1				1	float		3
CVT(T)PS2PI	mm,xmm	1	1		1					float	3	1
CVT(T)PS2PI	mm,m128	1	1		1				1	float		1
CVTPI2PD	xmm,mm	2	2	1	1					float	4	1
CVTPI2PD	xmm,m64	2	2	1	1				1	float		1
CVT(T) PD2PI	mm,xmm	2	2	1	1					float	4	1
CVT(T) PD2PI	mm,m128	2	2	1	1				1	float		1
CVTSI2SS	xmm,r32	1	1		1					float	4	3
CVTSI2SS	xmm,m32	1	1		1				1	float		3
CVT(T)SS2SI	r32,xmm	1	1		1					float	3	1
CVT(T)SS2SI	r32,m32	1	1		1				1	float		1
CVTSI2SD	xmm,r32	2	2	1	1					float	4	3

Wolfdale

CVTSI2SD	xmm,m32	2	1		1		1		float		3
CVT(T)SD2SI	r32,xmm	1	1		1				float	3	1
CVT(T)SD2SI	r32,m64	1	1		1		1		float		1
<b>Arithmetic</b>											
ADDSS/D SUBSS/D	xmm,xmm	1	1		1				float	3	1
ADDSS/D SUBSS/D	xmm,m32/64	1	1		1		1		float		1
ADDPS/D SUBPS/D	xmm,xmm	1	1		1				float	3	1
ADDPS/D SUBPS/D	xmm,m128	1	1		1		1		float		1
ADDSUBPS/D g)	xmm,xmm	1	1		1				float	3	1
ADDSUBPS/D g)	xmm,m128	1	1		1		1		float		1
HADDPS HSUBPS g)	xmm,xmm	3	3		1	2			float	7	3
HADDPS HSUBPS g)	xmm,m128	4	3		1	2	1		float		3
HADDPD HSUBPD g)	xmm,xmm	3	3	x	x	x			float	6	1.5
HADDPD HSUBPD g)	xmm,m128	4	3	x	x	x	1		float		1.5
MULSS	xmm,xmm	1	1	1					float	4	1
MULSS	xmm,m32	1	1	1			1		float		1
MULSD	xmm,xmm	1	1	1					float	5	1
MULSD	xmm,m64	1	1	1			1		float		1
MULPS	xmm,xmm	1	1	1					float	4	1
MULPS	xmm,m128	1	1	1			1		float		1
MULPD	xmm,xmm	1	1	1					float	5	1
MULPD	xmm,m128	1	1	1			1		float		1
DIVSS	xmm,xmm	1	1	1					float	6-13 d)	5-12 d)
DIVSS	xmm,m32	1	1	1			1		float		5-12 d)
DIVSD	xmm,xmm	1	1	1					float	6-21 d)	5-20 d)
DIVSD	xmm,m64	1	1	1			1		float		5-20 d)
DIVPS	xmm,xmm	1	1	1					float	6-13 d)	5-12 d)
DIVPS	xmm,m128	1	1	1			1		float		5-12 d)
DIVPD	xmm,xmm	1	1	1					float	6-21 d)	5-20 d)
DIVPD	xmm,m128	1	1	1			1		float		5-20 d)
RCPSS/PS	xmm,xmm	1	1		1				float	3	2
RCPSS/PS	xmm,m	1	1		1		1		float		2
CMPccSS/D	xmm,xmm	1	1		1				float	3	1
CMPccSS/D	xmm,m32/64	1	1		1		1		float		1
CMPccPS/D	xmm,xmm	1	1		1				float	3	1
CMPccPS/D	xmm,m128	1	1		1		1		float		1
COMISS/D UCOMISS/D	xmm,xmm	1	1		1				float	3	1
COMISS/D UCOMISS/D	xmm,m32/64	1	1		1		1		float		1
MAXSS/D MINSS/D	xmm,xmm	1	1		1				float	3	1
MAXSS/D MINSS/D	xmm,m32/64	1	1		1		1		float		1
MAXPS/D MINPS/D	xmm,xmm	1	1		1				float	3	1
MAXPS/D MINPS/D	xmm,m128	1	1		1		1		float		1
ROUNDSS/D j)	xmm,xmm,i	1	1		1				float	3	1
ROUNDSS/D j)	xmm,m128,i	1	1		1		1		float		1
ROUNDPS/D j)	xmm,xmm,i	1	1		1				float	3	1
ROUNDPS/D j)	xmm,m128,i	1	1		1		1		float		1
DPPS j)	xmm,xmm,i	4	4	2	2				float	11	3
DPPS j)	xmm,m128,i	4	4	2	2		1		float		3
DPPD j)	xmm,xmm,i	4	4	x	x	x			float	9	3



Wolfdale

DPPD j)	xmm,m128,i	4	4	x	x	x	1			float		3
<b>Math</b>												
SQRTSS/PS	xmm,xmm	1	1	1						float	6-13	5-12
SQRTSS/PS	xmm,m	2	1	1			1			float		5-12
SQRTSD/PD	xmm,xmm	1	1	1						float	6-20	5-19
SQRTSD/PD	xmm,m	2	1	1			1			float		5-19
RSQRTSS/PS	xmm,xmm	1	1			1				float	3	2
RSQRTSS/PS	xmm,m	1	1			1	1			float		2
<b>Logic</b>												
AND/ANDN/OR/XORPS/D	xmm,xmm	1	1	x	x	x				int	1	0.33
AND/ANDN/OR/XORPS/D	xmm,m128	1	1	x	x	x	1			int		1
<b>Other</b>												
LDMXCSR	m32	13	12	x	x	x	1					38
STMXCSR	m32	10	8	x	x	x		1	1			20
FXSAVE	m4096	151	67	x	x	x	8	38	38			145
FXRSTOR	m4096	121	74	x	x	x	47					150

**Notes:**

- d) Round divisors give low values.
- g) SSE3 instruction set.

**Intel Nehalem****List of instruction timings and  $\mu$ op breakdown***Explanation of column headings:*

**Operands:** i = immediate data, r = register, mm = 64 bit mmx register, xmm = 128 bit xmm register, (x)mm = mmx or xmm register, sr = segment register, m = memory, m32 = 32-bit memory operand, etc.

**$\mu$ ops fused domain:** The number of  $\mu$ ops at the decode, rename, allocate and retirement stages in the pipeline. Fused  $\mu$ ops count as one.

**$\mu$ ops unfused domain:** The number of  $\mu$ ops for each execution port. Fused  $\mu$ ops count as two. Fused macro-ops count as one. The instruction has  $\mu$ op fusion if the sum of the numbers listed under p015 + p2 + p3 + p4 exceeds the number listed under  $\mu$ ops fused domain. An x under p0, p1 or p5 means that at least one of the  $\mu$ ops listed under p015 can optionally go to this port. For example, a 1 under p015 and an x under p0 and p5 means one  $\mu$ op which can go to either port 0 or port 5, whichever is vacant first. A value listed under p015 but nothing under p0, p1 and p5 means that it is not known which of the three ports these  $\mu$ ops go to.

**p015:** The total number of  $\mu$ ops going to port 0, 1 and 5.

**p0:** The number of  $\mu$ ops going to port 0 (execution units).

**p1:** The number of  $\mu$ ops going to port 1 (execution units).

**p5:** The number of  $\mu$ ops going to port 5 (execution units).

**p2:** The number of  $\mu$ ops going to port 2 (memory read).

**p3:** The number of  $\mu$ ops going to port 3 (memory write address).

**p4:** The number of  $\mu$ ops going to port 4 (memory write data).

**Domain:** Tells which execution unit domain is used: "int" = integer unit (general purpose registers), "ivec" = integer vector unit (SIMD), "fp" = floating point unit (XMM and x87 floating point). An additional "bypass delay" is generated if a register written by a  $\mu$ op in one domain is read by a  $\mu$ op in another domain. The bypass delay is 1 clock cycle between the "int" and "ivec" units, and 2 clock cycles between the "int" and "fp", and between the "ivec" and "fp" units.

The bypass delay is indicated under latency only where it is unavoidable because either the source operand or the destination operand is in an unnatural domain such as a general purpose register (e.g. eax) in the "ivec" domain. For example, the PEXTRW instruction executes in the "int" domain. The source operand is an xmm register and the destination operand is a general purpose register. The latency for this instruction is indicated as 2+1, where 2 is the latency of the instruction itself and 1 is the bypass delay, assuming that the xmm operand is most likely to come from the "ivec" domain. If the xmm operand comes from the "fp" domain then the bypass delay will be 2 rather than one. The flags register can also have a bypass delay. For example, the COMISS instruction (floating point compare) executes in the "fp" domain and returns the result in the integer flags. Almost all instructions that read these flags execute in the "int" domain. Here the latency is indicated as 1+2, where 1 is the latency of the instruction itself and 2 is the bypass delay from the "fp" domain to the "int" domain.

The bypass delay from the memory read unit to any other unit and from any unit to the memory write unit are included in the latency figures in the table. Where the domain is not listed, the bypass delays are either unlikely to occur or unavoidable and therefore included in the latency figure.

## Nehalem

**Latency:** This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays very much, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay. The time unit used is core clock cycles, not the reference clock cycles given by the time stamp counter.

**Reciprocal throughput:** The average number of core clock cycles per instruction for a series of independent instructions of the same kind in the same thread.

### Integer instructions

Instruction	Operands	μops fused domain	μops unfused domain							Do-main	Laten-cy	Reци-procal through-put
			p015	p0	p1	p5	p2	p3	p4			
<b>Move instructions</b>												
MOV	r,r/i	1	1	x	x	x				int	1	0.33
MOV a)	r,m	1					1			int	2	1
MOV a)	m,r	1						1	1	int	3	1
MOV	m,i	1						1	1	int	3	1
MOV	r,sr	1					1			int		1
MOV	m,sr	2					1	1	1	int		1
MOV	sr,r	6	3	x	x	x	3			int		13
MOV	sr,m	6	2	x		x	4			int		14
MOVNTI	m,r	2						1	1	int	~270	1
MOVSX MOVZX												
MOVSXD	r,r	1	1	x	x	x				int	1	0.33
MOVSX MOVZX												
MOVSXD	r,m	1					1			int		1
CMOVcc	r,r	2	2	x	x	x				int	2	1
CMOVcc	r,m	2	2	x	x	x	1			int		1
XCHG	r,r	3	3	x	x	x				int	2	2
XCHG	r,m	7					1	1	1	int	20 b)	
XLAT		2	1				1			int	5	1
PUSH	r	1						1	1	int	3	1
PUSH	i	1						1	1	int		1
PUSH	m	2					1	1	1	int		1
PUSH	sr	2	1					1	1	int		1
PUSHF(D/Q)		3	2	x	x	x		1	1	int		1
PUSHA(D) i)		18	2	x	1	x		8	8	int		8
POP	r	1					1			int	2	1
POP	(E/R)SP	3	2	x	1	x	1			int		5
POP	m	2					1	1	1	int		1
POP	sr	7	2				5			int		15
POPF(D/Q)		8	7	x	x	x	1			int		14
POPA(D) i)		10	2				8			int		8
LAHF SAHF		1	1	x	x	x				int	1	0.33
SALC i)		2	2	x	x	x				int	4	1
LEA a)	r,m	1	1		1					int	1	1
BSWAP	r32	1	1		1					int	1	1

Nehalem

BSWAP	r64	1	1		1						int	3	1
LDS LES LFS LGS LSS	m	9	3	x	x	x	6				int		15
PREFETCHNTA	m	1					1				int		1
PREFETCHT0/1/2	m	1					1				int		1
LFENCE		2						1	1		int		9
MFENCE		3	1	x	x	x		1	1		int		23
SFENCE		2						1	1		int		5
<b>Arithmetic instructions</b>													
ADD SUB	r,r/i	1	1	x	x	x					int	1	0.33
ADD SUB	r,m	1	1	x	x	x	1				int		1
ADD SUB	m,r/i	2	1	x	x	x	1	1	1		int	6	1
ADC SBB	r,r/i	2	2	x	x	x					int	2	2
ADC SBB	r,m	2	2	x	x	x	1				int	2	2
ADC SBB	m,r/i	4	3	x	x	x	1	1	1		int	7	
CMP	r,r/i	1	1	x	x	x					int	1	0.33
CMP	m,r/i	1	1	x	x	x	1				int	1	1
INC DEC NEG NOT	r	1	1	x	x	x					int	1	0.33
INC DEC NEG NOT	m	3	1	x	x	x	1	1	1		int	6	1
AAA AAS DAA DAS i)		1	1		1						int	3	1
AAD i)		3	3	x	x	x					int	15	2
AAM i)		5	5	x	x	x					int	20	7
MUL IMUL	r8	1	1		1						int	3	1
MUL IMUL	r16	3	3	x	x	x					int	5	2
MUL IMUL	r32	3	3	x	x	x					int	5	2
MUL IMUL	r64	3	3	x	x	x					int	3	2
IMUL	r16,r16	1	1		1						int	3	1
IMUL	r32,r32	1	1		1						int	3	1
IMUL	r64,r64	1	1	1							int	3	1
IMUL	r16,r16,i	1	1		1						int	3	1
IMUL	r32,r32,i	1	1		1						int	3	1
IMUL	r64,r64,i	1	1	1							int	3	2
MUL IMUL	m8	1	1		1		1				int	3	1
MUL IMUL	m16	3	3	x	x	x	1				int	5	2
MUL IMUL	m32	3	3	x	x	x	1				int	5	2
MUL IMUL	m64	3	2	2			1				int	3	2
IMUL	r16,m16	1	1		1		1				int	3	1
IMUL	r32,m32	1	1		1		1				int	3	1
IMUL	r64,m64	1	1	1			1				int	3	1
IMUL	r16,m16,i	1	1		1		1				int		1
IMUL	r32,m32,i	1	1		1		1				int		1
IMUL	r64,m64,i	1	1	1			1				int		1
DIV c)	r8	4	4	1	2	1					int	11-21	7-11
DIV c)	r16	6	6	x	4	x					int	17-22	7-12
DIV c)	r32	6	6	x	3	x					int	17-28	7-17
DIV c)	r64	~40		x	x	x					int	28-90	19-69
IDIV c)	r8	4	4	1	2	1					int	10-22	7-11
IDIV c)	r16	8	8	x	5	x					int	18-23	7-12
IDIV c)	r32	7	7	x	3	x					int	17-28	7-17
IDIV c)	r64	~60		x	x	x					int	37-100	26-86

Nehalem

CBW CWDE CDQE		1	1	x	x	x				int	1	1
CWD CDQ CQO		1	1	x		x				int	1	1
POPCNT ℓ)	r,r	1	1		1					int	3	1
POPCNT ℓ)	r,m	1	1		1		1			int		1
CRC32 ℓ)	r,r	1	1		1					int	3	1
CRC32 ℓ)	r,m	1	1		1		1			int		1
<b>Logic instructions</b>												
AND OR XOR	r,r/i	1	1	x	x	x				int	1	0.33
AND OR XOR	r,m	1	1	x	x	x	1			int		1
AND OR XOR	m,r/i	2	1	x	x	x	1	1	1	int	6	1
TEST	r,r/i	1	1	x	x	x				int	1	0.33
TEST	m,r/i	1	1	x	x	x	1			int		1
SHR SHL SAR	r,i/cl	1	1	x		x				int	1	0.5
SHR SHL SAR	m,i/cl	3	2	x		x	1	1	1	int	6	1
ROR ROL	r,i/cl	1	1	x		x				int	1	1
ROR ROL	m,i/cl	3	2	x		x	1	1	1	int	6	1
RCR RCL	r,1	2	2	x	x	x				int	2	2
RCR	r8,i/cl	9	9	x	x	x				int	13	
RCL	r8,i/cl	8	8	x	x	x				int	11	
RCR RCL	r16/32/64,i/cl	6	6	x	x	x				int	12-13	12-13
RCR RCL	m,1	4	3	x	x	x	1	1	1	int	7	
RCR	m8,i/cl	12	9	x	x	x	1	1	1	int	16	
RCL	m8,i/cl	11	8	x	x	x	1	1	1	int	14	
RCR RCL	m16/32/64,i/cl	10	7	x	x	x	1	1	1	int	15	
SHLD	r,r,i/cl	2	2	x	x	x				int	3	1
SHLD	m,r,i/cl	3	2	x	x	x	1	1	1	int	8	
SHRD	r,r,i/cl	2	2	x	x	x				int	4	1
SHRD	m,r,i/cl	3	2	x	x	x	1	1	1	int	9	
BT	r,r/i	1	1	x		x				int	1	1
BT	m,r	9	8	x		x	1			int		5
BT	m,i	2	2	x		x	1			int		1
BTR BTS BTC	r,r/i	1	1	x		x				int	1	1
BTR BTS BTC	m,r	10	7	x	x	x	1	1	1	int	6	
BTR BTS BTC	m,i	3	3	x		x	1	1	1	int	6	
BSF BSR	r,r	1	1		1					int	3	1
BSF BSR	r,m	2	1		1		1			int	3	1
SETcc	r	1	1	x		x				int	1	1
SETcc	m	2	1	x	x	x		1	1	int		1
CLC STC CMC		1	1	x	x	x				int	1	0.33
CLD		2	2	x	x	x				int		4
STD		2	2	x	x	x				int		5
<b>Control transfer instructions</b>												
JMP	short/near	1	1			1				int	0	2
JMP i)	far	31	31							int		67
JMP	r	1	1			1				int	0	2
JMP	m(near)	1	1			1	1			int	0	2
JMP	m(far)	31	31				11			int		73
Conditional jump	short/near	1	1			1				int	0	2

### Nehalem

Fused compare/test and branch e)		1	1			1					int	0	2
J(E/R)CXZ	short	2	2	x	x	1					int		2
LOOP	short	6	6	x	x	x					int		4
LOOP(N)E	short	11	11	x	x	x					int		7
CALL	near	2	2			1		1	1		int		2
CALL i)	far	46	46					9			int		74
CALL	r	3	2			1			1	1	int		2
CALL	m(near)	4	3			1	1		1	1	int		2
CALL	m(far)	47	47					1			int		79
RETN		1	1			1	1				int		2
RETN	i	3	2			1	1				int		2
RETF		39	39								int		120
RETF	i	40	40								int		124
BOUND i)	r,m	15	13					2			int		7
INTO i)		4	4								int		5
<b>String instructions</b>													
LODS		2	1	x	x	x	1				int		1
REP LODS		11+4n									int	40+12n	
STOS		3	1	x	x	x			1	1	int		1
REP STOS	small n	60+n									int	12+n	
REP STOS	large n	2.5/16 bytes									int	1 clk / 16 bytes	
MOVS		5	2	x	x	x	1	1	1		int		4
REP MOVS	small n	13+6n									int	12+n	
REP MOVS	large n	2/16 bytes									int	1 clk / 16 bytes	
SCAS		3	2	x	x	x	1				int		1
REP SCAS		37+6n									int	40+2n	
CMPS		5	3	x	x	x	2				int		4
REP CMPS		65+8n									int	42+2n	
<b>Other</b>													
NOP (90)		1	1	x	x	x					int		0.33
Long NOP (0F 1F)		1	1	x	x	x					int		1
PAUSE		5	5	x	x	x					int		9
ENTER	a,0	11	9	x	x	x	1	1	1		int		8
ENTER	a,b	34+7b									int	79+5b	
LEAVE		3	3					1			int		5
CPUID		25-100									int	~200	~200
RDTSC		22									int		24
RDPNC		28									int		40-60

**Notes:**

- a) Applies to all addressing modes
- b) Has an implicit LOCK prefix.
- c) Low values are for small results, high values for high results.
- e) See manual 3: "The microarchitecture of Intel, AMD and VIA CPUs" for restrictions on macro-op fusion.
- i) Not available in 64 bit mode.
- ℓ) SSE4.2 instruction set.

## Floating point x87 instructions

Instruction	Operands	µops fused domain	µops unfused domain							Do-main	Laten-cy	Reci-procal through-put	
			p015	p0	p1	p5	p2	p3	p4				
<b>Move instructions</b>													
FLD	r	1	1	1							float	1	1
FLD	m32/64	1	1					1			float	3	1
FLD	m80	4	2	1	1			2			float	4	2
FBLD	m80	41	38	x	x	x		3			float	45	20
FST(P)	r	1	1	1							float	1	1
FST(P)	m32/m64	1							1	1	float	4	1
FSTP	m80	7	3	x	x	x			2	2	float	5	5
FBSTP	m80	208	204	x	x	x			2	2	float	242	245
FXCH	r	1	0 f)								float	0	1
FILD	m	1	1		1			1			float	6	1
FIST(P)	m	3	1		1				1	1	float	7	1
FISTTP g)	m	3	1		1				1	1	float	7	1
FLDZ		1	1	1							float		1
FLD1		2	2	1	1						float		2
FLDPI FLDL2E etc.		2	2		2						float		2
FCMOVcc	r	2	2	2							float	2+2	2
FNSTSW	AX	2	2								float		1
FNSTSW	m16	3	2						1	1	float		2
FLDCW	m16	2	1					1			float	7	31
FNSTCW	m16	2	1	1					1	1	float	5	1
FINCSTP FDECSTP		1	1	1							float	1	1
FFREE(P)	r	2	2	x	x	x					float		4
FNSAVE	m	143	89	x	x	x	8	23	23		float	178	178
FRSTOR	m	79	52	x	x	x	27				float	156	156
<b>Arithmetic instructions</b>													
FADD(P) FSUB(R)(P)	r	1	1		1						float	3	1
FADD(P) FSUB(R)(P)	m	1	1		1			1			float		1
FMUL(P)	r	1	1	1							float	5	1
FMUL(P)	m	1	1	1				1			float		1
FDIV(R)(P)	r	1	1	1							float	7-27 d)	7-27 d)
FDIV(R)(P)	m	1	1	1				1			float	7-27 d)	7-27 d)
FABS		1	1	1							float	1	1
FCHS		1	1	1							float	1	1
FCOM(P) FUCOM	r	1	1		1						float		1
FCOM(P) FUCOM	m	1	1		1			1			float		1
FCOMPP FUCOMPP		2	2	1	1						float		1
FCOMI(P) FUCOMI(P)	r	1	1		1						float		1
FIADD FISUB(R)	m	2	2		2			1			float	3	2
FIMUL	m	2	2	1	1			1			float	5	2
FIDIV(R)	m	2	2	1	1			1			float	7-27 d)	7-27 d)
FICOM(P)	m	2	2		2			1			float		1
FTST		1	1		1						float		1
FXAM		1	1		1						float		1

### Nehalem

FPREM	25	25	x	x	x				float	14	
FPREM1	35	35	x	x	x				float	19	
FRNDINT	17	17	x	x	x				float	22	
<b>Math</b>											
FSCALE	24	24	x	x	x				float	12	
FEXTRACT	17	17	x	x	x				float	13	
FSQRT	1	1	1						float	~27	
FSIN	~100	~100	x	x	x				float	40-100	
FCOS	~100	~100	x	x	x				float	40-100	
FSINCOS	~100	~100	x	x	x				float	~110	
F2XM1	19	19	x	x	x				float	58	
FYL2X FYL2XP1	~55	~55	x	x	x				float	~80	
FPTAN	~100	~100	x	x	x				float	~115	
FPATAN	~82	~82	x	x	x				float	~120	
<b>Other</b>											
FNOP	1	1	1						float		1
WAIT	2	2	x	x	x				float		1
FNCLEX	3	3		x	x				float		17
FNINIT	~190	~190	x	x	x				float		77

**Notes:**

- d) Round divisors or low precision give low values.
- f) Resolved by register renaming. Generates no  $\mu$ ops in the unfused domain.
- g) SSE3 instruction set.

### Integer MMX and XMM instructions

Instruction	Operands	$\mu$ ops fused do-main	$\mu$ ops unfused domain							Do-main	Latency	Reciprocal throughput
			p015	p0	p1	p5	p2	p3	p4			
<b>Move instructions</b>												
MOVD k)	r32/64,(x)mm	1	1	x	x	x				int	1+1	0.33
MOVD k)	m32/64,(x)mm	1						1	1		3	1
MOVD k)	(x)mm,r32/64	1	1	x	x	x				ivec	1+1	0.33
MOVD k)	(x)mm,m32/64	1					1				2	1
MOVQ	(x)mm, (x)mm	1	1	x	x	x				ivec	1	0.33
MOVQ	(x)mm,m64	1					1				2	1
MOVQ	m64, (x)mm	1						1	1		3	1
MOVDQA	xmm, xmm	1	1	x	x	x				ivec	1	0.33
MOVDQA	xmm, m128	1					1				2	1
MOVDQA	m128, xmm	1						1	1		3	1
MOVDQU	xmm, m128	1	1				1				2	1
MOVDQU	m128, xmm	1	1					1	1		3	1
LDDQU g)	xmm, m128	1	1				1				2	1
MOVDQ2Q	mm, xmm	1	1	x	x	x				ivec	1	0.33
MOVQ2DQ	xmm,mm	1	1	x	x	x				ivec	1	0.33
MOVNTQ	m64,mm	1						1	1		~270	2
MOVNTDQ	m128,xmm	1						1	1		~270	2



Nehalem

MOVNTDQA j)	xmm, m128	1					1				2	1
PACKSSWB/DW PACK- USWB	mm,mm	1	1		1					ivec	1	1
PACKSSWB/DW PACK- USWB	mm,m64	1	1		1		1					2
PACKSSWB/DW PACK- USWB	xmm,xmm	1	1	x		x				ivec	1	0.5
PACKSSWB/DW PACK- USWB	xmm,m128	1	1	x		x	1					2
PACKUSDW j)	xmm,xmm	1	1	x		x				ivec	1	2
PACKUSDW j)	xmm,m	1	1	x		x	1					2
PUNPCKH/LBW/WD/DQ	(x)mm, (x)mm	1	1	x		x				ivec	1	0.5
PUNPCKH/LBW/WD/DQ	(x)mm,m	1	1	x		x	1					2
PUNPCKH/LQDQ	xmm,xmm	1	1	x		x				ivec	1	0.5
PUNPCKH/LQDQ	xmm, m128	2	1	x		x	1					1
PMOVSX/ZXBW j)	xmm,xmm	1	1	x		x				ivec	1	1
PMOVSX/ZXBW j)	xmm,m64	1	1	x		x	1					2
PMOVSX/ZXBD j)	xmm,xmm	1	1	x		x				ivec	1	1
PMOVSX/ZXBD j)	xmm,m32	1	1	x		x	1					2
PMOVSX/ZXBQ j)	xmm,xmm	1	1	x		x				ivec	1	1
PMOVSX/ZXBQ j)	xmm,m16	1	1	x		x	1					2
PMOVSX/ZXWD j)	xmm,xmm	1	1	x		x				ivec	1	1
PMOVSX/ZXWD j)	xmm,m64	1	1	x		x	1					2
PMOVSX/ZXWQ j)	xmm,xmm	1	1	x		x				ivec	1	1
PMOVSX/ZXWQ j)	xmm,m32	1	1	x		x	1					2
PMOVSX/ZXDQ j)	xmm,xmm	1	1	x		x				ivec	1	1
PMOVSX/ZXDQ j)	xmm,m64	1	1	x		x	1					2
PSHUFB h)	(x)mm, (x)mm	1	1	x		x				ivec	1	0.5
PSHUFB h)	(x)mm,m	2	1	x		x	1					1
PSHUFW	mm,mm,i	1	1	x		x				ivec	1	0.5
PSHUFW	mm,m64,i	2	1	x		x	1					1
PSHUFD	xmm,xmm,i	1	1	x		x				ivec	1	0.5
PSHUFD	xmm,m128,i	2	1	x		x	1					1
PSHUFL/HW	xmm,xmm,i	1	1	x		x				ivec	1	0.5
PSHUFL/HW	xmm, m128,i	2	1	x		x	1					1
PALIGNR h)	(x)mm,(x)mm,i	1	1	x		x				ivec	1	1
PALIGNR h)	(x)mm,m,i	2	1	x		x	1					1
PBLENDVB j)	x,x,xmm0	2	2	1		1				ivec	2	1
PBLENDVB j)	xmm,m,xmm0	3	2	1		1	1					1
PBLENDW j)	xmm,xmm,i	1	1	x		x				ivec	1	0.5
PBLENDW j)	xmm,m,i	2	1	x		x	1					1
MASKMOVQ	mm,mm	4	1	1			1	1	1	ivec		2
MASKMOVDQU	xmm,xmm	10	4	x	x	x	2	2	x	ivec		7
PMOVMASKB	r32,(x)mm	1	1	1						float	2+2	1
PEXTRB j)	r32,xmm,i	2	2	x	x	x				ivec	2+1	1
PEXTRB j)	m8,xmm,i	2	2	x		x						1
PEXTRW	r32,(x)mm,i	2	2	x	x	x				ivec	2+1	1
PEXTRW j)	m16,(x)mm,i	2	2	x		x		1	1			1
PEXTRD j)	r32,xmm,i	2	2	x	x	x				ivec	2+1	1
PEXTRD j)	m32,xmm,i	2	1	x		x		1	1			1
PEXTRQ j,m)	r64,xmm,i	2	2	x	x	x				ivec	2+1	1

Nehalem

PEXTRQ j,m)	m64,xmm,i	2	1	x	x		1	1			1
PINSRB j)	xmm,r32,i	1	1	x	x				ivec	1+1	1
PINSRB j)	xmm,m8,i	2	1	x	x	1					1
PINSRW	(x)mm,r32,i	1	1	x	x				ivec	1+1	1
PINSRW	(x)mm,m16,i	2	1	x	x	1					1
PINSRD j)	xmm,r32,i	1	1	x	x				ivec	1+1	1
PINSRD j)	xmm,m32,i	2	1	x	x	1					1
PINSRQ j,m)	xmm,r64,i	1	1	x	x				ivec	1+1	1
PINSRQ j,m)	xmm,m64,i	2	1	x	x	1					1
<b>Arithmetic instructions</b>											
PADD/SUB(U) (S)B/W/D/Q	(x)mm, (x)mm	1	1	x	x				ivec	1	0.5
PADD/SUB(U) (S)B/W/D/Q	(x)mm,m	1	1	x	x	1					2
PHADD/SUB(S)W/D h)	(x)mm, (x)mm	3	3	x	x				ivec	3	1.5
PHADD/SUB(S)W/D h)	(x)mm,m64	4	3	x	x	1					3
PCMPEQ/GTB/W/D	(x)mm,(x)mm	1	1	x	x				ivec	1	0.5
PCMPEQ/GTB/W/D	(x)mm,m	1	1	x	x	1					2
PCMPEQQ j)	xmm,xmm	1	1	x	x				ivec	1	0.5
PCMPEQQ j)	xmm,m128	1	1	x	x	1					2
PCMPGTQ †)	xmm,xmm	1	1		1				ivec	3	1
PCMPGTQ †)	xmm,m128	1	1		1	1					1
PMULL/HW PMULHUW	(x)mm,(x)mm	1	1		1				ivec	3	1
PMULL/HW PMULHUW	(x)mm,m	1	1		1	1					1
PMULHRSW h)	(x)mm,(x)mm	1	1		1				ivec	3	1
PMULHRSW h)	(x)mm,m	1	1		1	1					1
PMULLD j)	xmm,xmm	2	2		2				ivec	6	2
PMULLD j)	xmm,m128	3	2		2	1					
PMULDQ j)	xmm,xmm	1	1		1				ivec	3	1
PMULDQ j)	xmm,m128	1	1		1	1					1
PMULUDQ	(x)mm,(x)mm	1	1		1				ivec	3	1
PMULUDQ	(x)mm,m	1	1		1	1					1
PMADDWD	(x)mm,(x)mm	1	1		1				ivec	3	1
PMADDWD	(x)mm,m	1	1		1	1					1
PMADDUBSW h)	(x)mm,(x)mm	1	1		1				ivec	3	1
PMADDUBSW h)	(x)mm,m	1	1		1	1					1
PAVGB/W	(x)mm,(x)mm	1	1	x	x				ivec	1	0.5
PAVGB/W	(x)mm,m	1	1	x	x	1					1
PMIN/MAXSB j)	xmm,xmm	1	1	x	x				ivec	1	1
PMIN/MAXSB j)	xmm,m128	1	1	x	x	1					2
PMIN/MAXUB	(x)mm,(x)mm	1	1	x	x				ivec	1	0.5
PMIN/MAXUB	(x)mm,m	1	1	x	x	1					2
PMIN/MAXSW	(x)mm,(x)mm	1	1	x	x				ivec	1	0.5
PMIN/MAXSW	(x)mm,m	1	1	x	x	1					2
PMIN/MAXUW j)	xmm,xmm	1	1	x	x				ivec	1	1
PMIN/MAXUW j)	xmm,m	1	1	x	x	1					2
PMIN/MAXU/SD j)	xmm,xmm	1	1	x	x				ivec	1	1
PMIN/MAXU/SD j)	xmm,m128	1	1	x	x	1					2
PHMINPOSUW j)	xmm,xmm	1	1		1				ivec	3	1

Nehalem

PHMINPOSUW j)	xmm,m128	1	1		1	1					3
PABSB PABSW PABSD h)	(x)mm,(x)mm	1	1	x		x			ivec	1	0.5
PABSB PABSW PABSD h)	(x)mm,m	1	1	x		x	1				1
PSIGNB PSIGNW PSIGND h)	(x)mm,(x)mm	1	1	x		x			ivec	1	0.5
PSIGNB PSIGNW PSIGND h)	(x)mm,m	1	1	x		x	1				2
PSADBW	(x)mm,(x)mm	1	1		1				ivec	3	1
PSADBW	(x)mm,m	1	1		1		1				3
MPSADBW j)	xmm,xmm,i	3	3	x	x	x			ivec	5	1
MPSADBW j)	xmm,m,i	4	3	x	x	x	1				2
PCLMULQDQ n)	xmm,xmm,i									12	8
AESDEC, AESDECLAST, AESENC, AESENCLAST n)	xmm,xmm									~5	~2
AESIMC n)	xmm,xmm									~5	~2
AESKEYGENASSIST n)	xmm,xmm,i									~5	~2
<b>Logic instructions</b>											
PAND(N) POR PXOR	(x)mm,(x)mm	1	1	x	x	x			ivec	1	0.33
PAND(N) POR PXOR	(x)mm,m	1	1	x	x	x	1				1
PTEST j)	xmm,xmm	2	2	x	x	x			ivec	3	1
PTEST j)	xmm,m128	2	2	x	x	x	1				1
PSLL/RL/RAW/D/Q	mm,mm/i	1	1		1				ivec	1	1
PSLL/RL/RAW/D/Q	mm,m64	1	1		1		1				2
PSLL/RL/RAW/D/Q	xmm,i	1	1		1				ivec	1	1
PSLL/RL/RAW/D/Q	xmm,xmm	2	2	x	1	x			ivec	2	2
PSLL/RL/RAW/D/Q	xmm,m128	3	2	x	1	x	1				1
PSLL/RLDQ	xmm,i	1	1	x		x			ivec	1	1
<b>String instructions</b>											
PCMPSTRM ℓ)	xmm,xmm,i	8	8	x	x	x			ivec	14	5
PCMPSTRM ℓ)	xmm,m128,i	9	8	x	x	x	1		ivec	14	6
PCMPSTRM ℓ)	xmm,xmm,i	9	9	x	x	x			ivec	7	6
PCMPSTRM ℓ)	xmm,m128,i	10	10	x	x	x	1		ivec	7	6
PCMPSTRM ℓ)	xmm,xmm,i	3	3	x	x	x			ivec	8	2
PCMPSTRM ℓ)	xmm,m128,i	4	4	x	x	x	1		ivec	8	2
PCMPSTRM ℓ)	xmm,xmm,i	4	4	x	x	x			ivec	7	2
PCMPSTRM ℓ)	xmm,m128,i	6	5	x	x	x	1		ivec	7	5
<b>Other</b>											
EMMS		11	11	x	x	x			float		6

**Notes:**

- g) SSE3 instruction set.
- h) Supplementary SSE3 instruction set.
- j) SSE4.1 instruction set
- k) MASM uses the name MOVD rather than MOVQ for this instruction even when moving 64 bits

- ℓ) SSE4.2 instruction set  
 m) Only available in 64 bit mode  
 n) Only available on newer models

### Floating point XMM instructions

Instruction	Operands	μops fused do-main	μops unfused domain								Do-main	Laten-cy	Reci-procal through-put
			p015	p0	p1	p5	p2	p3	p4				
<b>Move instructions</b>													
MOVAPS/D	xmm,xmm	1	1			1					float	1	1
MOVAPS/D	xmm,m128	1					1					2	1
MOVAPS/D	m128,xmm	1						1	1			3	1
MOVUPS/D	xmm,m128	1					1					2	1-4
MOVUPS/D	m128,xmm	1						1	1			3	1-3
MOVSS/D	xmm,xmm	1	1			1						1	1
MOVSS/D	xmm,m32/64	1					1					2	1
MOVSS/D	m32/64,xmm	1						1	1			3	1
MOVHPS/D MOVLPS/D	xmm,m64	2	1			1	1					3	2
MOVH/LPS/D	m64,xmm	2	1			1		1	1			5	1
MOVLHPS MOVHLPS	xmm,xmm	1	1			1					float	1	1
MOVMSKPS/D	r32,xmm	1	1	1							float	1+2	1
MOVNTPS/D	m128,xmm	1						1	1			~270	2
SHUFPS/D	xmm,xmm,i	1	1			1					float	1	1
SHUFPS/D	xmm,m128,i	2	1			1	1				float		1
BLENDPS/PD j)	xmm,xmm,i	1	1			1					float	1	1
BLENDPS/PD j)	xmm,m128,i	2	1			1	1				float		1
BLENDVPS/PD j)	x,x,xmm0	2	2			2					float	2	2
BLENDVPS/PD j)	xmm,m,xmm0	3	2			2	1				float		2
MOVDDUP g)	xmm,xmm	1	1			1					float	1	1
MOVDDUP g)	xmm,m64	1					1					2	1
MOVSH/LDUP g)	xmm,xmm	1	1			1					float	1	1
MOVSH/LDUP g)	xmm,m128	1					1						1
UNPCKH/LPS/D	xmm,xmm	1	1			1					float	1	1
UNPCKH/LPS/D	xmm,m128	1	1			1	1				float		1
EXTRACTPS j)	r32,xmm,i	1	1			1					float	1+2	1
EXTRACTPS j)	m32,xmm,i	2	1			1		1	1				1
INSERTPS j)	xmm,xmm,i	1	1			1					float	1	1
INSERTPS j)	xmm,m32,i	3	1			2	1				float		2
<b>Conversion</b>													
CVTPD2PS	xmm,xmm	2	2			1	1				float	4	1
CVTPD2PS	xmm,m128	2	2			1		1			float		1
CVTSD2SS	xmm,xmm	2	2			1	1				float	4	1
CVTSD2SS	xmm,m64	2	2			1		1			float		1
CVTSS2PD	xmm,xmm	2	2	1		1					float	2	1
CVTSS2PD	xmm,m64	2	2	1		1	1				float		1
CVTSS2SD	xmm,xmm	1	1	1							float	1	1
CVTSS2SD	xmm,m32	1	1	1				1			float		2

Nehalem

CVTDQ2PS	xmm,xmm	1	1	1			float	3+2	1
CVTDQ2PS	xmm,m128	1	1	1	1		float		1
CVT(T) PS2DQ	xmm,xmm	1	1	1			float	3+2	1
CVT(T) PS2DQ	xmm,m128	1	1	1	1		float		1
CVTDQ2PD	xmm,xmm	2	2	1	1		float	4+2	1
CVTDQ2PD	xmm,m64	2	2	1	1	1	float		1
CVT(T)PD2DQ	xmm,xmm	2	2	1	1		float	4+2	1
CVT(T)PD2DQ	xmm,m128	2	2	1	1	1	float		1
CVTPI2PS	xmm,mm	1	1	1			float	3+2	3
CVTPI2PS	xmm,m64	1	1	1	1		float		3
CVT(T)PS2PI	mm,xmm	1	1	1			float	3+2	1
CVT(T)PS2PI	mm,m128	1	1	1	1		float		1
CVTPI2PD	xmm,mm	2	2	1	1		ivec/float	6	1
CVTPI2PD	xmm,m64	2	2	1	1	1			1
CVT(T) PD2PI	mm,xmm	2	2	x	1	x	float/ivec	6	1
CVT(T) PD2PI	mm,m128	2	2	x	1	x			1
CVTSI2SS	xmm,r32	1	1	1			float	3+2	3
CVTSI2SS	xmm,m32	1	1	1	1		float		3
CVT(T)SS2SI	r32,xmm	1	1	1			float	3+2	1
CVT(T)SS2SI	r32,m32	1	1	1	1		float		1
CVTSI2SD	xmm,r32	2	2	1	1		float	4+2	3
CVTSI2SD	xmm,m32	2	1	1	1		float		3
CVT(T)SD2SI	r32,xmm	1	1	1			float	3+2	1
CVT(T)SD2SI	r32,m64	1	1	1	1		float		1
<b>Arithmetic</b>									
ADDSS/D SUBSS/D	xmm,xmm	1	1	1			float	3	1
ADDSS/D SUBSS/D	xmm,m32/64	1	1	1	1		float		1
ADDPS/D SUBPS/D	xmm,xmm	1	1	1			float	3	1
ADDPS/D SUBPS/D	xmm,m128	1	1	1	1		float		1
ADDSUBPS/D g)	xmm,xmm	1	1	1			float	3	1
ADDSUBPS/D g)	xmm,m128	1	1	1	1		float		1
HADDPS HSUBPS g)	xmm,xmm	3	3	1	2		float	5	2
HADDPS HSUBPS g)	xmm,m128	4	3	1	2	1	float		2
HADDPD HSUBPD g)	xmm,xmm	3	3	1	2		float	3	2
HADDPD HSUBPD g)	xmm,m128	4	3	1	2	1	float		2
MULSS MULPS	xmm,xmm	1	1	1			float	4	1
MULSS MULPS	xmm,m	1	1	1		1	float		1
MULSD MULPD	xmm,xmm	1	1	1			float	5	1
MULSD MULPD	xmm,m	1	1	1		1	float		1
DIVSS DIVPS	xmm,xmm	1	1	1			float	7-14	7-14
DIVSS DIVPS	xmm,m	1	1	1		1	float		7-14
DIVSD DIVPD	xmm,xmm	1	1	1			float	7-22	7-22
DIVSD DIVPD	xmm,m	1	1	1		1	float		7-22
RCPSS/PS	xmm,xmm	1	1	1			float	3	2
RCPSS/PS	xmm,m	1	1	1	1		float		2
CMPccSS/D CMPccPS/D	xmm,xmm	1	1	1			float	3	1
CMPccSS/D CMPccPS/D	xmm,m	2	1	1	1		float		1

Nehalem

COMISS/D UCOMISS/D	xmm,xmm	1	1		1					float	1+2	1
COMISS/D UCOMISS/D	xmm,m32/64	1	1		1		1			float		1
MAXSS/D MINSS/D	xmm,xmm	1	1		1					float	3	1
MAXSS/D MINSS/D	xmm,m32/64	1	1		1		1			float		1
MAXPS/D MINPS/D	xmm,xmm	1	1		1					float	3	1
MAXPS/D MINPS/D	xmm,m128	1	1		1		1			float		1
ROUNDSS/D ROUNDPS/D j)	xmm,xmm,i	1	1		1					float	3	1
ROUNDSS/D ROUNDPS/D j)	xmm,m128,i	2	1		1		1			float		1
DPPS j)	xmm,xmm,i	4	4	1	2	1				float	11	2
DPPS j)	xmm,m128,i	6	5	x	x	x	1			float		
DPPD j)	xmm,xmm,i	3	3	x	x	x				float	9	1
DPPD j)	xmm,m128,i	4	3	x	x	x	1			float		3
<b>Math</b>												
SQRTSS/PS	xmm,xmm	1	1	1						float	7-18	7-18
SQRTSS/PS	xmm,m	2	1	1			1			float		7-18
SQRTSD/PD	xmm,xmm	1	1	1						float	7-32	7-32
SQRTSD/PD	xmm,m	2	1	1			1			float		7-32
RSQRTSS/PS	xmm,xmm	1	1		1					float	3	2
RSQRTSS/PS	xmm,m	1	1		1		1			float		2
<b>Logic</b>												
AND/ANDN/OR/XORPS/D	xmm,xmm	1	1				1			float	1	1
AND/ANDN/OR/XORPS/D	xmm,m128	1	1				1	1		float		1
<b>Other</b>												
LDMXCSR	m32	6	6	x	x	x	1					5
STMXCSR	m32	2	1				1		1	1		1
FXSAVE	m4096	141	141	x	x	x	5	38	38		90	90
FXRSTOR	m4096	112	90	x	x	x	42					100

**Notes:**

g) SSE3 instruction set.

## Intel Sandy Bridge

### List of instruction timings and $\mu$ op breakdown

#### Explanation of column headings:

**Operands:** i = immediate data, r = register, mm = 64 bit mmx register, x = 128 bit xmm register, (x)mm = mmx or xmm register, y = 256 bit ymm register, same = same register for both operands. m = memory operand, m32 = 32-bit memory operand, etc.

**$\mu$ ops fused domain:** The number of  $\mu$ ops at the decode, rename, allocate and retirement stages in the pipeline. Fused  $\mu$ ops count as one.

**$\mu$ ops unfused domain:** The number of  $\mu$ ops for each execution port. Fused  $\mu$ ops count as two. Fused macro-ops count as one. The instruction has  $\mu$ op fusion if the sum of the numbers listed under p015 + p23 + p4 exceeds the number listed under  $\mu$ ops fused domain. A number indicated as 1+ under a read or write port means a 256-bit read or write operation using two clock cycles for handling 128 bits each cycle. The port cannot receive another read or write  $\mu$ op in the second clock cycle, but a read port can receive an address-calculation  $\mu$ op in the second clock cycle. An x under p0, p1 or p5 means that at least one of the  $\mu$ ops listed under p015 can optionally go to this port. For example, a 1 under p015 and an x under p0 and p5 means one  $\mu$ op which can go to either port 0 or port 5, whichever is vacant first. A value listed under p015 but nothing under p0, p1 and p5 means that it is not known which of the three ports these  $\mu$ ops go to.

**p015:** The total number of  $\mu$ ops going to port 0, 1 and 5.

**p0:** The number of  $\mu$ ops going to port 0 (execution units).

**p1:** The number of  $\mu$ ops going to port 1 (execution units).

**p5:** The number of  $\mu$ ops going to port 5 (execution units).

**p23:** The number of  $\mu$ ops going to port 2 or 3 (memory read or address calculation).

**p4:** The number of  $\mu$ ops going to port 4 (memory write data).

**Latency:** This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Where hyperthreading is enabled, the use of the same execution units in the other thread leads to inferior performance. Denormal numbers, NAN's and infinity do not increase the latency. The time unit used is core clock cycles, not the reference clock cycles given by the time stamp counter.

**Reciprocal throughput:** The average number of core clock cycles per instruction for a series of independent instructions of the same kind in the same thread.

The latencies and throughputs listed below for addition and multiplication using full size YMM registers are obtained only after a warm-up period of a thousand instructions or more. The latencies may be one or two clock cycles longer and the reciprocal throughputs double the values for shorter sequences of code. There is no warm-up effect when vectors are 128 bits wide or less.

### Integer instructions

Instruction	Operands	$\mu$ ops fused domain	$\mu$ ops unfused domain						Latency	Reciprocal throughput	Comments
			p015	p0	p1	p5	p23	p4			
Move instructions											

## Sandy Bridge

MOV	r,r/i	1	1	x	x	x			1	0.33	
MOV	r,m	1					1		2	0.5	all addressing modes
MOV	m,r	1					1	1	3	1	
MOV	m,i	1					1	1		1	
MOVNTI	m,r	2					1	1	~350	1	
MOVSX MOVZX	r,r	1	1	x	x	x			1	0.33	
MOVSXD											
MOVSX MOVZX	r,m	1					1			0.5	
MOVSXD											
CMOVcc	r,r	2	2	x	x	x			2	1	
CMOVcc	r,m	2	2	x	x	x	1			1	
XCHG	r,r	3	3	x	x	x			2	1	
XCHG	r,m	8					2	1	25		implicit lock
XLAT		3	2				1		7	1	
PUSH	r	1					1	1	3	1	
PUSH	i	1					1	1		1	
PUSH	m	2					2	1		1	
PUSHF(D/Q)		3	2	x	x	x	1	1		1	
PUSHA(D)		16	0				8	8		8	not 64 bit
POP	r	1					1		2	0.5	
POP	(E/R)SP	1	0				1			0.5	
POP	m	2					2	1		1	
POPF(D/Q)		9	8	x	x	x	1			18	
POPA(D)		18	10				8			9	not 64 bit
LAHF SAHF		1	1						1	1	
SALC		3	3						1	1	not 64 bit
LEA	r,m	1	1	1		1			1	0.5	simple
LEA	r,m	1	1		1				3	1	complex or rip relative
BSWAP	r32	1	1		1				1	1	
BSWAP	r64	2	2		2				2	1	
PREFETCHNTA	m	1					1			0.5	
PREFETCHT0/1/2	m	1					1			0.5	
LFENCE		2					1	1		4	
MFENCE		3	1				1	1		33	
SFENCE		2					1	1		6	
<b>Arithmetic instructions</b>											
ADD SUB	r,r/i	1	1	x	x	x			1	0.33	
ADD SUB	r,m	1	1	x	x	x	1			0.5	
ADD SUB	m,r/i	2	1	x	x	x	2	1	6	1	
SUB	r,same	1	0						0	0.25	
ADC SBB	r,r/i	2	2	x	x	x			2	1	
ADC SBB	r,m	2	2	x	x	x	1		2	1	
ADC SBB	m,r/i	4	3	x	x	x	2	1	7	1.5	
CMP	r,r/i	1	1	x	x	x			1	0.33	
CMP	m,r/i	1	1	x	x	x	1		1	0.5	



Sandy Bridge

INC DEC NEG NOT	r	1	1	x	x	x			1	0.33	
INC DEC NEG NOT	m	3	1	x	x	x	2	1	6	2	
AAA AAS		2	2						4		not 64 bit
DAA DAS		3	3						4		not 64 bit
AAD		3	3						2		not 64 bit
AAM		8	8						20	11	not 64 bit
MUL IMUL	r8	1	1		1				3	1	
MUL IMUL	r16	4	4						4	2	
MUL IMUL	r32	3	3						4	2	
MUL IMUL	r64	2	2						3	1	
IMUL	r,r	1	1		1				3	1	
IMUL	r16,r16,i	2	2						4	1	
IMUL	r32,r32,i	1	1		1				3	1	
IMUL	r64,r64,i	1	1		1				3	1	
MUL IMUL	m8	1	1		1		1		3	1	
MUL IMUL	m16	4	3				1			2	
MUL IMUL	m32	3	2				1			2	
MUL IMUL	m64	2	1				1			2	
IMUL	r,m	1	1		1		1			1	
IMUL	r16,m16,i	2	2				1			1	
IMUL	r32,m32,i	1	1		1		1			1	
IMUL	r64,m64,i	1	1		1		1			1	
DIV	r8	10	10						20-24	11-14	
DIV	r16	11	11						21-25	11-14	
DIV	r32	10	10						20-28	11-18	
DIV	r64	34-56							30-94	22-76	
IDIV	r8	10	10						21-24	11-14	
IDIV	r16	10	10						21-25	11-14	
IDIV	r32	9	9						20-27	11-18	
IDIV	r64	59-138							40-103	25-84	
CBW		1	1						1	0.5	
CWDE		1	1			1			1	1	
CDQE		1	1						1	0.5	
CWD		2	2						1	1	
CDQ		1	1						1	1	
CQO		1	1						1	0.5	
POPCNT	r,r	1	1		1				3	1	SSE4.2
POPCNT	r,m	1	1		1		1			1	SSE4.2
CRC32	r,r	1	1		1				3	1	SSE4.2
CRC32	r,m	1	1		1		1			1	SSE4.2
<b>Logic instructions</b>											
AND OR XOR	r,r/i	1	1	x	x	x			1	0.33	
AND OR XOR	r,m	1	1	x	x	x	1			0.5	
AND OR XOR	m,r/i	2	1	x	x	x	2	1	6	1	
XOR	r,same	1	0						0	0.25	
TEST	r,r/i	1	1	x	x	x			1	0.33	
TEST	m,r/i	1	1	x	x	x	1			0.5	
SHR SHL SAR	r,i	1	1	x		x			1	0.5	

Sandy Bridge

SHR SHL SAR	m,i	3	1				2	1	1	2
SHR SHL SAR	r,cl	3	3						2	2
SHR SHL SAR	m,cl	5	3				2	1		4
ROR ROL	r,i	1	1						1	1
ROR ROL	m,i	4	3				2	1		2
ROR ROL	r,cl	3	3						2	2
ROR ROL	m,cl	5	3				2	1		4
RCR	r8,1	high							high	high
RCR	r16/32/64,1	3	3						2	2
RCR	r,i	8	8						5	5
RCR	m,i	11	7							6
RCR	r,cl	8	8						5	5
RCR	m,cl	11	7							6
RCL	r,1	3	3						2	2
RCL	r,i	8	8						6	6
RCL	m,i	11	7							6
RCL	r,cl	8	8						6	6
RCL	m,cl	11	7							6
SHRD SHLD	r,r,i	1	1							0.5
SHRD SHLD	m,r,i	3					2	1		2
SHRD SHLD	r,r,cl	4	4						2	2
SHRD SHLD	m,r,cl	5	3				2	1		4
BT	r,r/i	1	1						1	0.5
BT	m,r	10	8				1			5
BT	m,i	2	1				1			0.5
BTR BTS BTC	r,r/i	1	1						1	0.5
BTR BTS BTC	m,r	11	7				2	1		5
BTR BTS BTC	m,i	3	1				2	1		2
BSF BSR	r,r	1	1						3	1
BSF BSR	r,m	1	1		1		1			1
SETcc	r	1	1	x		x			1	0.5
SETcc	m	2	1	x		x	1	1		1
CLC		1	0							0.25
STC CMC		1	1	x	x	x			1	0.33
CLD STD		3	3							4
<b>Control transfer instructions</b>										
JMP	short/near	1	1			1			0	2
JMP	r	1	1			1			0	2
JMP	m	1	1			1	1		0	2
Conditional jump	short/near	1	1			1			0	1-2
Fused arithmetic and branch		1	1			1			0	1-2
J(E/R)CXZ	short	2	2	x	x	1				2-4
LOOP	short	7	7							5
LOOP(N)E	short	11	11							5
CALL	near	3	2			1	1	1		2
CALL	r	2	1			1	1	1		2

fastest if not jumping

Sandy Bridge

CALL	m	3	2			1	2	1		2	
RET		2	2			1	1			2	
RET	i	3	2			1	1			2	
BOUND	r,m	15	13							7	not 64 bit
INTO		4	4							6	not 64 bit
<b>String instructions</b>											
LODS		3	2				1			1	
REP LODS		5n+12							~2n		
STOS		3	1				1	1		1	
REP STOS		2n							n		worst case
REP STOS		1.5/16B							1/16B		best case
MOVS		5								4	
REP MOVS		2n							1.5 n		worst case
REP MOVS		3/16B							1/16B		best case
SCAS		3								1	
REP SCAS		6n+47							2n+45		
CMPS		5								4	
REP CMPS		8n+80							2n+80		
<b>Other</b>											
NOP (90)		1	0							0.25	
Long NOP (0F 1F)		1	0							0.25	decode only 1 per clk
PAUSE		7	7							11	
ENTER	a,0	12	10				2	1		8	
ENTER	a,b	49+6b							84+3b		
LEAVE		3	3				1			7	
CPUID		31-75							100-250		
RDTSC		21								28	
RDPMSR		35								42	

**Floating point x87 instructions**

Instruction	Operands	μops fused domain	μops unfused domain						Latency	Reciprocal throughput	Comments
			p015	p0	p1	p5	p23	p4			
<b>Move instructions</b>											
FLD	r	1	1	1					1	1	
FLD	m32/64	1	1				1		3	1	
FLD	m80	4	2	1	1		2		4	2	
FBLD	m80	43	40				3		45	21	
FST(P)	r	1	1	1					1	1	
FST(P)	m32/m64	1					1	1	4	1	
FSTP	m80	7	3				2	2	5	5	

Sandy Bridge

FBSTP	m80	246						252	
FXCH	r	1	0				0	0.5	
FILD	m	1	1	1	1		6	1	
FIST(P)	m	3	1	1	1	1	7	2	
FISTTP	m	3	1	1	1	1	7	2	SSE3
FLDZ		1	1	1				2	
FLD1		2	2	1	1			2	
FLDPI FLDL2E etc.		2	2		2			2	
FCMOVcc	r	3	3				3	2	
FNSTSW	AX	2	2				2	1	
FNSTSW	m16	2	1			1	1	1	
FLDCW	m16	3	2			1	8		
FNSTCW	m16	2	1	1		1	1	1	
FINCSTP FDECSTP		1	1	1			1	1	
FFREE(P)	r	1	1					1	
FNSAVE	m	143						166	
FRSTOR	m	90						165	
<b>Arithmetic instructions</b>									
FADD(P) FSUB(R)(P)	r	1	1		1		3	1	
FADD(P) FSUB(R)(P)	m	2	2		1	1		1	
FMUL(P)	r	1	1	1			5	1	
FMUL(P)	m	1	1	1		1		1	
FDIV(R)(P)	r	1	1	1			10-24	10-24	
FDIV(R)(P)	m	1	1	1		1		10-24	
FABS		1	1	1			1	1	
FCHS		1	1	1			1	1	
FCOM(P) FUCOM	r	1	1		1		3	1	
FCOM(P) FUCOM	m	1	1		1	1		1	
FCOMPP FUCOMPP		2	2	1	1			1	
FCOMI(P) FUCOMI(P)	r	3	3		1		4	1	
FIADD FISUB(R)	m	2	2		2	1		1	
FIMUL	m	2	2	1	1	1		1	
FIDIV(R)	m	2	2	1	1	1			
FICOM(P)	m	2	2		2	1		2	
FTST		1	1		1			1	
FXAM		2	2		1			2	
FPREM		28	28				21	21	
FPREM1		41-87					26-50	26-50	
FRNDINT		17	17				22		
<b>Math</b>									
FSCALE		27	27				12		
FEXTRACT		17	17				10		
FSQRT		1	1	1			10-24		
FSIN		64-100					47-100		
FCOS		20-110					47-115		
FSINCOS		20-110					43-123		
F2XM1		53-118					61-69		
FYL2X		454	454				724		

Sandy Bridge

FYL2XP1		464	464						726	
FPTAN		102	102						130	
FPATAN		28-91							93-146	
<b>Other</b>										
FNOP		1	1	1					1	
WAIT		2	2						1	
FNCLEX		5	5						22	
FNINIT		26	26						81	

**Integer MMX and XMM instructions**

Instruction	Operands	μops fused do-main	μops unfused domain						Latency	Reci-procal through-put	Com-ments
			p015	p0	p1	p5	p23	p4			
<b>Move instructions</b>											
MOVD	r32/64,(x)mm	1	1	x	x	x			1	0.33	
MOVD	m32/64,(x)mm	1					1	1	3	1	
MOVD	(x)mm,r32/64	1	1	x	x	x			1	0.33	
MOVD	(x)mm,m32/64	1					1		3	0.5	
MOVQ	(x)mm,(x)mm	1	1	x	x	x			1	0.33	
MOVQ	(x)mm,m64	1					1		1	0.5	
MOVQ	m64, (x)mm	1					1	1	3	1	
MOVDQA	x,x	1	1	x	x	x			1	0.33	
MOVDQA	x, m128	1					1		3	0.5	
MOVDQA	m128, x	1					1	1	3	1	
MOVDQU	x, m128	1	1				1		3	0.5	
MOVDQU	m128, x	1	1				1	1	3	1	
LDDQU	x, m128	1	1				1		3	0.5	SSE3
MOVDQ2Q	mm, x	2	2						1	1	
MOVQ2DQ	x,mm	1	1						1	0.33	
MOVNTQ	m64,mm	1					1	1	~300	1	
MOVNTDQ	m128,x	1					1	1	~300		
MOVNTDQA	x, m128	1					1			0.5	SSE4.1
PACKSSWB/DW PACK-USWB	mm,mm	1	1						1	1	
PACKSSWB/DW PACK-USWB	mm,m64	1	1		1		1				
PACKSSWB/DW PACK-USWB	x,x	1	1	x		x			1	0.5	
PACKSSWB/DW PACK-USWB	x,m128	1	1	x		x	1			0.5	
PACKUSDW	x,x	1	1	x		x			1	0.5	SSE4.1
PACKUSDW	x,m	1	1	x		x	1			0.5	SSE4.1
PUNPCKH/LBW/WD/DQ	(x)mm,(x)mm	1	1	x		x			1	0.5	
PUNPCKH/LBW/WD/DQ	(x)mm,m	1	1	x		x	1			0.5	
PUNPCKH/LQDQ	x,x	1	1	x		x			1	0.5	
PUNPCKH/LQDQ	x, m128	2	1	x		x	1			0.5	
PMOVSX/ZXBW	x,x	1	1	x		x			1	0.5	SSE4.1
PMOVSX/ZXBW	x,m64	1	1	x		x	1			0.5	SSE4.1

Sandy Bridge

PMOVSX/ZXBD	x,x	1	1	x	x			1	0.5	SSE4.1
PMOVSX/ZXBD	x,m32	1	1	x	x	1			0.5	SSE4.1
PMOVSX/ZXBQ	x,x	1	1	x	x			1	0.5	SSE4.1
PMOVSX/ZXBQ	x,m16	1	1	x	x	1			0.5	SSE4.1
PMOVSX/ZXWD	x,x	1	1	x	x			1	0.5	SSE4.1
PMOVSX/ZXWD	x,m64	1	1	x	x	1			0.5	SSE4.1
PMOVSX/ZXWQ	x,x	1	1	x	x			1	0.5	SSE4.1
PMOVSX/ZXWQ	x,m32	1	1	x	x	1			0.5	SSE4.1
PMOVSX/ZXDQ	x,x	1	1	x	x			1	0.5	SSE4.1
PMOVSX/ZXDQ	x,m64	1	1	x	x	1			0.5	SSE4.1
PSHUFB	(x)mm,(x)mm	1	1	x	x			1	0.5	SSSE3
PSHUFB	(x)mm,m	2	1	x	x	1			0.5	SSSE3
PSHUFW	mm,mm,i	1	1	x	x			1	0.5	
PSHUFW	mm,m64,i	2	1	x	x	1			0.5	
PSHUFD	xmm,x,i	1	1	x	x			1	0.5	
PSHUFD	x,m128,i	2	1	x	x	1			0.5	
PSHUFL/HW	x,x,i	1	1	x	x			1	0.5	
PSHUFL/HW	x, m128,i	2	1	x	x	1			0.5	
PALIGNR	(x)mm,(x)mm,i	1	1	x	x			1	0.5	SSSE3
PALIGNR	(x)mm,m,i	2	1	x	x	1			0.5	SSSE3
PBLENDVB	x,x,xmm0	2	2	1	1			2	1	SSE4.1
PBLENDVB	x,m,xmm0	3	2	1	1	1			1	SSE4.1
PBLENDW	x,x,i	1	1	x	x			1	0.5	SSE4.1
PBLENDW	x,m,i	2	1	x	x	1			0.5	SSE4.1
MASKMOVQ	mm,mm	4	1	1			2	1	1	
MASKMOVDQU	x,x	10	4				4	x	6	
PMOVMASKB	r32,(x)mm	1	1	1					2	1
PEXTRB	r32,x,i	2	2	x	x	x			2	1
PEXTRB	m8,x,i	2	1	x	x	1	1		1	SSE4.1
PEXTRW	r32,(x)mm,i	2	2	x	x				2	1
PEXTRW	m16,(x)mm,i	2	1	x	x	1	1		2	SSE4.1
PEXTRD	r32,x,i	2	2	x	x	x			2	1
PEXTRD	m32,x,i	3	2	x	x	1	1		1	SSE4.1
PEXTRQ	r64,x,i	2	2	x	x	x			2	1
PEXTRQ	m64,x,i	3	2	x	x	1	1		1	SSE4.1, 64b
PINSRB	x,r32,i	2	2	x	x				2	1
PINSRB	x,m8,i	2	1	x	x	1			0.5	SSE4.1
PINSRW	(x)mm,r32,i	2	2	x	x				2	1
PINSRW	(x)mm,m16,i	2	1	x	x	1			0.5	
PINSRD	x,r32,i	2	2	x	x				2	1
PINSRD	x,m32,i	2	1	x	x	1			0.5	SSE4.1
PINSRQ	x,r64,i	2	2	x	x				2	1
PINSRQ	x,m64,i	2	1	x	x	1			0.5	SSE4.1, 64 b
<b>Arithmetic instructions</b>										
PADD/SUB(U,S)B/W/D/Q	(x)mm, (x)mm	1	1	x	x				1	0.5
PADD/SUB(U,S)B/W/D/Q	(x)mm,m	1	1	x	x	1				0.5
PHADD/SUB(S)W/D	(x)mm, (x)mm	3	3	x	x				2	1.5
PHADD/SUB(S)W/D	(x)mm,m64	4	3	x	x	1				1.5
PCMPEQ/GTB/W/D	(x)mm,(x)mm	1	1	x	x				1	0.5

Sandy Bridge

PCMPEQ/GTB/W/D	(x)mm,m	1	1	x		x	1		0.5	
PCMPEQQ	x,x	1	1	x		x		1	0.5	SSE4.1
PCMPEQQ	x,m128	1	1	x		x	1		0.5	SSE4.1
PCMPGTQ	x,x	1	1	1				5	1	SSE4.2
PCMPGTQ	x,m128	1	1	1			1		1	SSE4.2
PSUBxx, PCMPGTx	x,same	1	0					0	0.25	
PCMPEQx	x,same	1	1					0	0.5	
PMULL/HW PMULHUW	(x)mm,(x)mm	1	1		1			5	1	
PMULL/HW PMULHUW	(x)mm,m	1	1		1	1			1	
PMULHRWSW	(x)mm,(x)mm	1	1	1				5	1	SSSE3
PMULHRWSW	(x)mm,m	1	1	1		1			1	SSSE3
PMULLD	x,x	1	1	1				5	1	SSE4.1
PMULLD	x,m128	2	1	1		1			1	SSE4.1
PMULDQ	x,x	1	1	1				5	1	SSE4.1
PMULDQ	x,m128	1	1	1		1			1	SSE4.1
PMULUDQ	(x)mm,(x)mm	1	1	1				5	1	
PMULUDQ	(x)mm,m	1	1	1		1			1	
PMADDWD	(x)mm,(x)mm	1	1	1				5	1	
PMADDWD	(x)mm,m	1	1	1		1			1	
PMADDUBSW	(x)mm,(x)mm	1	1	1				5	1	SSSE3
PMADDUBSW	(x)mm,m	1	1	1		1			1	SSSE3
PAVGB/W	(x)mm,(x)mm	1	1	x		x		1	0.5	
PAVGB/W	(x)mm,m	1	1	x		x	1		0.5	
PMIN/MAXSB	x,x	1	1	x		x		1	0.5	SSE4.1
PMIN/MAXSB	x,m128	1	1	x		x	1		0.5	SSE4.1
PMIN/MAXUB	(x)mm,(x)mm	1	1	x		x		1	0.5	
PMIN/MAXUB	(x)mm,m	1	1	x		x	1		0.5	
PMIN/MAXSW	(x)mm,(x)mm	1	1	x		x		1	0.5	
PMIN/MAXSW	(x)mm,m	1	1	x		x	1		0.5	
PMIN/MAXUW	x,x	1	1	x		x		1	0.5	SSE4.1
PMIN/MAXUW	x,m	1	1	x		x	1		0.5	SSE4.1
PMIN/MAXU/SD	x,x	1	1	x		x		1	0.5	SSE4.1
PMIN/MAXU/SD	x,m128	1	1	x		x	1		0.5	SSE4.1
PHMINPOSUW	x,x	1	1	1				5	1	SSE4.1
PHMINPOSUW	x,m128	1	1	1			1		1	SSE4.1
PABSB/W/D	(x)mm,(x)mm	1	1	x		x		1	0.5	SSSE3
PABSB/W/D	(x)mm,m	1	1	x		x	1		0.5	SSSE3
PSIGNB/W/D	(x)mm,(x)mm	1	1	x		x		1	0.5	SSSE3
PSIGNB/W/D	(x)mm,m	1	1	x		x	1		0.5	SSSE3
PSADBW	(x)mm,(x)mm	1	1	1				5	1	
PSADBW	(x)mm,m	1	1	1			1		1	
MPSADBW	x,x,i	3	3					6	1	SSE4.1
MPSADBW	x,m,i	4	3				1		1	SSE4.1
PCLMULQDQ	x,x,i	18	18					14	8	only in some processors
AESDEC, AESDECLAST, AESENC, AESENCLAST	x,x	2	2					8	4	do.

Sandy Bridge

AESIMC	x,x	2	2						2	2	do.
AESKEYGENASSIST	x,x,i	11	11						8	8	do.
<b>Logic instructions</b>											
PAND(N) POR PXOR	(x)mm,(x)mm	1	1	x	x	x			1	0.33	
PAND(N) POR PXOR	(x)mm,m	1	1	x	x	x	1			0.5	
PXOR	x,same	1	0						0	0.25	
PTEST	x,x	1	1						1	1	SSE4.1
PTEST	x,m128	1	1				1			1	SSE4.1
PSLL/RL/RAW/D/Q	mm,mm/i	1	1		1				1	1	
PSLL/RL/RAW/D/Q	mm,m64	1	1		1		1			2	
PSLL/RL/RAW/D/Q	xmm,i	1	1		1				1	1	
PSLL/RL/RAW/D/Q	x,x	2	2						2	1	
PSLL/RL/RAW/D/Q	x,m128	3	2				1			1	
PSLL/RDQ	x,i	1	1						1	1	
<b>String instructions</b>											
PCMPSTRM	x,x,i	8	8						4	4	SSE4.2
PCMPSTRM	x,m128,i	8	7				1			4	SSE4.2
PCMPSTRM	x,x,i	8	8						11-12	4	SSE4.2
PCMPSTRM	x,m128,i	8	7				1			4	SSE4.2
PCMPSTRM	x,x,i	3	3						3	3	SSE4.2
PCMPSTRM	x,m128,i	4	3				1			3	SSE4.2
PCMPSTRM	x,x,i	3	3						11	3	SSE4.2
PCMPSTRM	x,m128,i	4	3				1			3	SSE4.2
<b>Other</b>											
EMMS		31	31							18	

Floating point XMM and YMM instructions

Instruction	Operands	μops fused domain	μops unfused domain						Latency	Reciprocal throughput	Comments
			p015	p0	p1	p5	p23	p4			
<b>Move instructions</b>											
MOVAPS/D	x,x	1	1				1		1	1	
VMOVAPS/D	y,y	1	1				1		1	1	AVX
MOVAPS/D MOVUPS/D	x,m128	1						1	3	0.5	
VMOVAPS/D	y,m256	1						1+	4	1	AVX
MOVAPS/D MOVUPS/D	m128,x	1						1	1	3	1
VMOVAPS/D	m256,y	1						1	1+	3	1
VMOVUPS/D											AVX
MOVSS/D	x,x	1	1				1		1	1	
MOVSS/D	x,m32/64	1						1	3	0.5	
MOVSS/D	m32/64,x	1						1	1	3	1
MOVHPS/D MOVLPS/D	x,m64	1	1				1	1	3	1	
MOVH/LPS/D	m64,x	1	1				1	1	1	3	1
MOVLHPS MOVHLPS	x,x	1	1				1		1	1	



## Sandy Bridge

MOVMSKPS/D	r32,x	1	1	1				2	1	
VMOVMSKPS/D	r32,y	1	1	1				2	1	
MOVNTPS/D	m128,x	1				1	1	~300	1	
VMOVNTPS/D	m256,y	1				1	4	~300	25	AVX
SHUFPS/D	x,x,i	1	1		1			1	1	
SHUFPS/D	x,m128,i	2	1		1	1			1	
VSHUFPS/D	y,y,y,i	1	1		1			1	1	AVX
VSHUFPS/D	y, y,m256,i	2	1		1	1+			1	AVX
VPERMILPS/PD	x,x,x/i	1	1		1			1	1	AVX
VPERMILPS/PD	y,y,y/i	1	1		1			1	1	AVX
VPERMILPS/PD	x,x,m	2	1		1	1			1	AVX
VPERMILPS/PD	y,y,m	2	1		1	1+			1	AVX
VPERMILPS/PD	x,m,i	2	1		1	1			1	AVX
VPERMILPS/PD	y,m,i	2	1		1	1+			1	AVX
VPERM2F128	y,y,y,i	1	1		1			2	1	AVX
VPERM2F128	y,y,m,i	2	1		1	1+			1	AVX
BLENDPS/PD	x,x,i	1	1		1			1	0.5	SSE4.1
BLENDPS/PD	x,m128,i	2	1		1	1			0.5	SSE4.1
VBLENDPS/PD	y,y,i	1	1		1			1	1	AVX
VBLENDPS/PD	y,m256,i	2	1		1	1+			1	AVX
BLENDVPS/PD	x,x,xmm0	2	2		2			2	1	SSE4.1
BLENDVPS/PD	x,m,xmm0	3	2		2	1			1	SSE4.1
VBLENDVPS/PD	y,y,y,y	2	2		2			2	1	AVX
VBLENDVPS/PD	y,y,m,y	3	2		2	1+			1	AVX
MOVDDUP	x,x	1	1		1			1	1	SSE3
MOVDDUP	x,m64	1				1		3	0.5	SSE3
VMOVDDUP	y,y	1	1		1			1	1	AVX
VMOVDDUP	y,m256	1				1+		3	1	AVX
VBROADCASTSS	x,m32	1				1			1	AVX
VBROADCASTSS	y,m32	2	1		1	1			1	AVX
VBROADCASTSD	y,m64	2	1		1	1			1	AVX
VBROADCASTF128	y,m128	2	1		1	1			1	AVX
MOVSH/LDUP	x,x	1	1		1			1	1	SSE3
MOVSH/LDUP	x,m128	1				1		3	0.5	SSE3
VMOVSH/LDUP	y,y	1	1		1			1	1	AVX
VMOVSH/LDUP	y,m256	1				1+		4	1	AVX
UNPCKH/LPS/D	x,x	1	1		1			1	1	SSE3
UNPCKH/LPS/D	x,m128	1	1		1	1			1	SSE3
VUNPCKH/LPS/D	y,y,y	1	1		1			1	1	AVX
VUNPCKH/LPS/D	y,y,m256	1	1		1	1+			1	AVX
EXTRACTPS	r32,x,i	2	2		1			2	1	SSE4.1
EXTRACTPS	m32,x,i	3	2		1	1	1		1	SSE4.1
VEXTRACTF128	x,y,i	1	1		1			2	1	AVX
VEXTRACTF128	m128,y,i	2	1			1	1		1	AVX
INSERTPS	x,x,i	1	1		1			1	1	SSE4.1
INSERTPS	x,m32,i	2	1		1	1			1	SSE4.1
VINSERTF128	y,y,x,i	1	1		1			2	1	AVX
VINSERTF128	y,y,m128,i	2	1		1	1			1	AVX
VMASKMOVPS/D	x,x,m128	3	2			1			1	AVX
VMASKMOVPS/D	y,y,m256	3	2			1+			1	AVX

Sandy Bridge

VMASKMOVPS/D	m128,x,x	4	2				1	1		1	AVX	
VMASKMOVPS/D	m256,y,y	4	2				1	1+		2	AVX	
<b>Conversion</b>												
CVTPD2PS	x,x	2	2		1	1				3	1	
CVTPD2PS	x,m128	2	2		1		1				1	
VCVTPD2PS	x,y	2	2		1	1				4	1	AVX
VCVTPD2PS	x,m256	2	2		1			1+			1	AVX
CVTSD2SS	x,x	2	2		1	1				3	1	
CVTSD2SS	x,m64	2	2		1			1			1	
CVTPS2PD	x,x	2	2	1		1				3	1	
CVTPS2PD	x,m64	2	2	1		1	1				1	
VCVTPS2PD	y,x	2	2	1		1				4	1	AVX
VCVTPS2PD	y,m128	3	3				1	1			1	AVX
CVTSS2SD	x,x	2	2	1						3	1	
CVTSS2SD	x,m32	2	1	1				1			1	
CVTDQ2PS	x,x	1	1		1					3	1	
CVTDQ2PS	x,m128	1	1		1			1			1	
VCVTDQ2PS	y,y	1	1		1					3	1	AVX
VCVTDQ2PS	y,m256	1	1		1			1+			1	AVX
CVT(T) PS2DQ	x,x	1	1		1					3	1	
CVT(T) PS2DQ	x,m128	1	1		1			1			1	
VCVT(T) PS2DQ	y,y	1	1		1					3	1	AVX
VCVT(T) PS2DQ	y,m256	1	1		1			1+			1	AVX
CVTDQ2PD	x,x	2	2		1	1				4	1	
CVTDQ2PD	x,m64	2	2		1	1	1				1	
VCVTDQ2PD	y,x	2	2		1	1				5	1	AVX
VCVTDQ2PD	y,m128	3	2		1	1	1				1	AVX
CVT(T)PD2DQ	x,x	2	2		1	1				4	1	
CVT(T)PD2DQ	x,m128	2	2		1	1	1				1	
VCVT(T)PD2DQ	x,y	2	2		1	1				5	1	AVX
VCVT(T)PD2DQ	x,m256	2	2		1	1	1+				1	AVX
CVTPI2PS	x,mm	1	1		1					4	2	
CVTPI2PS	x,m64	1	1		1			1			2	
CVT(T)PS2PI	mm,x	2	2		1					4	1	
CVT(T)PS2PI	mm,m128	2	1		1			1			1	
CVTPI2PD	x,mm	2	2		1	1				4	1	
CVTPI2PD	x,m64	2	2		1	1	1				1	
CVT(T) PD2PI	mm,x	2	2							4	1	
CVT(T) PD2PI	mm,m128	2	2					1			1	
CVTSI2SS	x,r32	2	2		1					4	1.5	
CVTSI2SS	x,m32	1	1		1			1			1.5	
CVT(T)SS2SI	r32,x	2	2		1					4	1	
CVT(T)SS2SI	r32,m32	2	2		1			1			1	
CVTSI2SD	x,r32	2	2	1	1					4	1.5	
CVTSI2SD	x,m32	1	1		1			1			1.5	
CVT(T)SD2SI	r32,x	2	2		1					4	1	
CVT(T)SD2SI	r32,m64	2	2		1			1			1	
<b>Arithmetic</b>												

## Sandy Bridge

ADDSS/D SUBSS/D	x,x	1	1	1			3	1	
ADDSS/D SUBSS/D	x,m32/64	1	1	1	1			1	
ADDPSS/D SUBPSS/D	x,x	1	1	1			3	1	
ADDPSS/D SUBPSS/D	x,m128	1	1	1	1			1	
VADDPSS/D VSUBPSS/D	y,y,y	1	1	1			3	1	AVX
VADDPSS/D VSUBPSS/D	y,y,m256	1	1	1	1+			1	AVX
ADDSUBPSS/D	x,x	1	1	1			3	1	SSE3
ADDSUBPSS/D	x,m128	1	1	1	1			1	SSE3
VADDSUBPSS/D	y,y,y	1	1	1			3	1	AVX
VADDSUBPSS/D	y,y,m256	1	1	1	1+			1	AVX
HADDPSS/D HSUBPSS/D	x,x	3	3	1	2		5	2	SSE3
HADDPSS/D HSUBPSS/D	x,m128	4	3	1	2	1		2	SSE3
VHADDPSS/D									
VHSUBPSS/D	y,y,y	3	3	1	2		5	2	AVX
VHADDPSS/D									
VHSUBPSS/D	y,y,m256	4	3	1	2	1+		2	AVX
MULSS MULPS	x,x	1	1	1			5	1	
MULSS MULPS	x,m	1	1	1		1		1	
VMULPS	y,y,y	1	1	1			5	1	AVX
VMULPS	y,y,m256	1	1	1	1+			1	AVX
MULSD MULPD	x,x	1	1	1			5	1	
MULSD MULPD	x,m	1	1	1		1		1	
VMULPD	y,y,y	1	1	1			5	1	AVX
VMULPD	y,y,m256	1	1	1	1+			1	AVX
DIVSS DIVPS	x,x	1	1	1			10-14	10-14	
DIVSS DIVPS	x,m	1	1	1		1		10-14	
VDIVPS	y,y,y	3	3	2	1		21-29	20-28	AVX
VDIVPS	y,y,m256	4	3	2	1	1+		20-28	AVX
DIVSD DIVPD	x,x	1	1	1			10-22	10-22	
DIVSD DIVPD	x,m	1	1	1		1		10-22	
VDIVPD	y,y,y	3	3	2	1		21-45	20-44	AVX
VDIVPD	y,y,m256	4	3	2	1+			20-44	AVX
RCPSS/PS	x,x	1	1	1			5	1	
RCPSS/PS	x,m128	1	1	1		1		1	
VRCPPS	y,y	2	3				7	2	AVX
VRCPPS	y,m256	4	3			1+		2	AVX
CMPccSS/D CMPccPS/D									
CMPccSS/D CMPccPS/D	x,x	1	1	1			3	1	
CMPccSS/D CMPccPS/D									
CMPccSS/D CMPccPS/D	x,m128	2	1	1	1			1	
VCMPccPS/D	y,y,y	1	1	1			3	1	AVX
VCMPccPS/D	y,y,m256	2	1	1	1+			1	AVX
COMISS/D UCOMISS/D	x,x	2	2				2	1	
COMISS/D UCOMISS/D	x,m32/64	2	2	1	1			1	
MAXSS/D MINSS/D	x,x	1	1	1			3	1	
MAXSS/D MINSS/D	x,m32/64	1	1	1	1			1	
MAXPS/D MINPS/D	x,x	1	1	1			3	1	
MAXPS/D MINPS/D	x,m128	1	1	1	1			1	
VMAXPS/D VMINPS/D	y,y,y	1	1	1			3	1	AVX
VMAXPS/D VMINPS/D	y,y,m256	1	1	1	1+			1	AVX
ROUNDSS/SD/PS/PD	x,x,i	1	1	1			3	1	SSE4.1

Sandy Bridge

ROUNDSS/SD/PS/PD	x,m128,i	2	1		1	1			1	SSE4.1
VROUNDSS/SD/PS/PD	y,y,i	1	1		1			3	1	AVX
VROUNDSS/SD/PS/PD	y,m256,i	2	1		1	1+			1	AVX
DPPS	x,x,i	4	4	1	2	1		12	2	SSE4.1
DPPS	x,m128,i	6	5						4	SSE4.1
VDPPS	y,y,y,i	4	4					12	2	AVX
VDPPS	y,m256,i	6	5						4	AVX
DPPD	x,x,i	3	3					9	2	SSE4.1
DPPD	x,m128,i	4	3						2	SSE4.1
<b>Math</b>										
SQRTSS/PS	x,x	1	1	1				10-14	10-14	
SQRTSS/PS	x,m128	1	1	1					10-14	
VSQRTPS	y,y	3	3						21-28	AVX
VSQRTPS	y,m256	4	3						21-28	AVX
SQRTSD/PD	x,x	1	1	1				10-21	10-21	
SQRTSD/PD	x,m128	2	1	1					10-21	
VSQRTPD	y,y	3	3					21-43	21-43	AVX
VSQRTPD	y,m256	4	3						21-43	AVX
RSQRTSS/PS	x,x	1	1	1				5	1	
RSQRTSS/PS	x,m128	1	1	1					1	
VRSQRTPS	y,y	3	3					7	2	AVX
VRSQRTPS	y,m256	4	3						2	AVX
<b>Logic</b>										
AND/ANDN/OR/XORPS/PD	x,x	1	1			1		1	1	
AND/ANDN/OR/XORPS/PD	x,m128	1	1			1	1		1	
VAND/ANDN/OR/XORPS/PD	y,y,y	1	1			1		1	1	AVX
VAND/ANDN/OR/XORPS/PD	y,y,m256	1	1			1	1+		1	AVX
(V)XORPS/PD	x/y,x/y,same	1	0					0	0.25	
<b>Other</b>										
VZEROUPPER		4						2	1	AVX
VZEROALL		12							11	AVX, 32 bit
VZEROALL		20							9	AVX, 64 bit
LDMXCSR	m32	3	3				1		3	
STMXCSR	m32	3	3			1	1	1	1	
VSTMXCSR	m32	3	3			1	1	1	1	AVX
FXSAVE	m4096	130							68	
FXRSTOR	m4096	116							72	
XSAVEOPT	m	100-161						60-500		

## Intel Pentium 4

### List of instruction timings and $\mu$ op breakdown

This list is measured for a Pentium 4, model 2. Timings for model 3 may be more like the values for P4E, listed on the next sheet

#### Explanation of column headings:

<b>Instruction:</b>	Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc.
<b>Operands:</b>	i = immediate constant, r = any register, r32 = 32-bit register, etc., mm = 64 bit mmx register, xmm = 128 bit xmm register, sr = segment register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc.
<b><math>\mu</math>ops:</b>	Number of $\mu$ ops issued from instruction decoder and stored in trace cache.
<b>Microcode:</b>	Number of additional $\mu$ ops issued from microcode ROM.
<b>Latency:</b>	This is the delay that the instruction generates in a dependency chain if the next dependent instruction starts in the same execution unit. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's, infinity and exceptions increase the delays. The latency of moves to and from memory cannot be measured accurately because of the problem with memory intermediates explained above under "How the values were measured".
<b>Additional latency:</b>	This number is added to the latency if the next dependent instruction is in a different execution unit. There is no additional latency between ALU0 and ALU1.
<b>Reciprocal throughput:</b>	This is also called issue latency. This value indicates the number of clock cycles from the execution of an instruction begins to a subsequent independent instruction can begin to execute in the same execution subunit. A value of 0.25 indicates 4 instructions per clock cycle in one thread.
<b>Port:</b>	The port through which each $\mu$ op goes to an execution unit. Two independent $\mu$ ops can start to execute simultaneously only if they are going through different ports.
<b>Execution unit:</b>	Use this information to determine additional latency. When an instruction with more than one $\mu$ op uses more than one execution unit, only the first and the last execution unit is listed.
<b>Execution subunit:</b>	Throughput measures apply only to instructions executing in the same subunit.
<b>Instruction set</b>	Indicates the compatibility of an instruction with other 80x86 family microprocessors. The instruction can execute on microprocessors that support the instruction set indicated.

### Integer instructions

---

Pentium 4

Instruction	Operands	µops	Microcode	Latency	Additional latency	Reciprocal through-put	Port	Execution unit	Subunit	Instruction set	Notes
<b>Move instructions</b>											
MOV	r,r	1	0	0.5	0.5-1	0.25	0/1	alu0/1		86	c
MOV	r,i	1	0	0.5	0.5-1	0.25	0/1	alu0/1		86	
MOV	r32,m	1	0	2	0	1	2	load		86	
MOV	r8/16,m	2	0	3	0	1	2	load		86	
MOV	m,r	1	0	1		2	0	store		86	b, c
MOV	m,i	3	0			2	0,3	store		86	
MOV	r,sr	4	2			6				86	
MOV	sr,r/m	4	4	12	0	14				86	a, q
MOVNTI	m,r32	2	0			≈33				sse2	
MOVZX	r,r	1	0	0.5	0.5-1	0.25	0/1	alu0/1		386	c
MOVZX	r,m	1	0	2	0	1	2	load		386	
MOVSX	r,r	1	0	0.5	0.5-1	0.5	0	alu0		386	c
MOVSX	r,m	2	0	3	0.5-1	1	2,0			386	
CMOVcc	r,r/m	3	0	6	0	3				ppro	a, e
XCHG	r,r	3	0	1.5	0.5-1	1	0/1	alu0/1		86	
XCHG	r,m	4	8	>100						86	
XLAT		4	0	3						86	
PUSH	r	2	0	1		2				86	
PUSH	i	2	0	1		2				186	
PUSH	m	3	0			2				86	
PUSH	sr	4	4			7				86	
PUSHF(D)		4	4			10				86	
PUSHA(D)		4	10			19				186	
POP	r	2	0	1	0	1				86	
POP	m	4	8			14				86	
POP	sr	4	5			13				86	
POPF(D)		4	8			52				86	
POPA(D)		4	16			14				186	
LEA	r,[r+r/i]	1	0	0.5	0.5-1	0.25	0/1	alu0/1		86	
LEA	r,[r+r+i]	2	0	1	0.5-1	0.5	0/1	alu0/1		86	
LEA	r,[r*i]	3	0	4	0.5-1	1	1	int,alu		386	
LEA	r,[r+r*i]	2	0	4	0.5-1	1	1	int,alu		386	
LEA	r,[r+r*i+i]	3	0	4	0.5-1	1	1	int,alu		386	
LAHF		1	0	4	0	4	1	int		86	
SAHF		1	0	0.5	0.5-1	0.5	0/1	alu0/1		86	d
SALC		3	0	5	0	1	1	int		86	
LDS, LES, ...	r,m	4	7			15				86	
BSWAP	r	3	0	7	0	2		int,alu		486	
IN, OUT	r,r/i	8	64			>1000			86		
PREFETCHNTA	m	4	2			6				sse	
PREFETCHT0/1/2	m	4	2			6				sse	

## Pentium 4

SFENCE		4	2			40				sse	
LFENCE		4	2			38				sse2	
MFENCE		4	2			100				sse2	
<b>Arithmetic instructions</b>											
ADD, SUB	r,r	1	0	0.5	0.5-1	0.25	0/1	alu0/1		86	c
ADD, SUB	r,m	2	0	1	0.5-1	1				86	c
ADD, SUB	m,r	3	0	≥ 8		≥ 4				86	c
ADC, SBB	r,r	4	4	6	0	6	1	int,alu		86	
ADC, SBB	r,i	3	0	6	0	6	1	int,alu		86	
ADC, SBB	r,m	4	6	8	0	8	1	int,alu		86	
ADC, SBB	m,r	4	7	≥ 9		8				86	
CMP	r,r	1	0	0.5	0.5-1	0.25	0/1	alu0/1		86	c
CMP	r,m	2	0	1	0.5-1	1				86	c
INC, DEC	r	2	0	0.5	0.5-1	0.5	0/1	alu0/1		86	
INC, DEC	m	4	0	4		≥ 4				86	
NEG	r	1	0	0.5	0.5-1	0.5	0	alu0		86	
NEG	m	3	0			≥ 3				86	
AAA, AAS		4	27	90						86	
DAA, DAS		4	57	100						86	
AAD		4	10	22			1	int	fpmul	86	
AAM		4	22	56			1	int	fpdiv	86	
MUL, IMUL	r8/32	4	6	16	0	8	1	int	fpmul	86	
MUL, IMUL	r16	4	7	17	0	8	1	int	fpmul	86	
MUL, IMUL	m8/32	4	7-8	16	0	8	1	int	fpmul	86	
MUL, IMUL	m16	4	10	16	0	8	1	int	fpmul	86	
IMUL	r32,r	4	0	14	0	4.5	1	int	fpmul	386	
IMUL	r32,(r),i	4	0	14	0	4.5	1	int	fpmul	386	
IMUL	r16,r	4	5	16	0	9	1	int	fpmul	386	
IMUL	r16,r,i	4	5	15	0	8	1	int	fpmul	186	
IMUL	r16,m16	4	7	15	0	10	1	int	fpmul	386	
IMUL	r32,m32	4	0	14	0	8	1	int	fpmul	386	
IMUL	r,m,i	4	7	14	0	10	1	int	fpmul	186	
DIV	r8/m8	4	20	61	0	24	1	int	fpdiv	86	a
DIV	r16/m16	4	18	53	0	23	1	int	fpdiv	86	a
DIV	r32/m32	4	21	50	0	23	1	int	fpdiv	386	
IDIV	r8/m8	4	24	61	0	24	1	int	fpdiv	86	a
IDIV	r16/m16	4	22	53	0	23	1	int	fpdiv	86	a
IDIV	r32/m32	4	20	50	0	23	1	int	fpdiv	386	a
CBW		2	0	1	0.5-1	1	0	alu0		86	
CWD, CDQ		2	0	1	0.5-1	0.5	0/1	alu0/1		86	
CWDE		1	0	0.5	0.5-1	0.5	0	alu0		386	
<b>Logic instructions</b>											
AND, OR, XOR	r,r	1	0	0.5	0.5-1	0.5	0	alu0		86	c
AND, OR, XOR	r,m	2	0	≥ 1	0.5-1	≥ 1				86	c
AND, OR, XOR	m,r	3	0	≥ 8		≥ 4				86	c
TEST	r,r	1	0	0.5	0.5-1	0.5	0	alu0		86	c
TEST	r,m	2	0	≥ 1	0.5-1	≥ 1				86	c
NOT	r	1	0	0.5	0.5-1	0.5	0	alu0		86	

Pentium 4

NOT	m	4	0			≥ 4				86	
SHL, SHR, SAR	r,i	1	0	4	1	1	1	int	mmxsh	186	
SHL, SHR, SAR	r,CL	2	0	6	0	1	1	int	mmxsh	86	d
ROL, ROR	r,i	1	0	4	1	1	1	int	mmxsh	186	d
ROL, ROR	r,CL	2	0	6	0	1	1	int	mmxsh	86	d
RCL, RCR	r,1	1	0	4	1	1	1	int	mmxsh	86	d
RCL, RCR	r,i	4	15	16	0	15	1	int	mmxsh	186	d
RCL, RCR	r,CL	4	15	16	0	14	1	int	mmxsh	86	d
SHL,SHR,SAR,ROL, ROR	m,i/CL	4	7-8	10	0	10	1	int	mmxsh	86	d
RCL, RCR	m,1	4	7	10	0	10	1	int	mmxsh	86	d
RCL, RCR	m,i/CL	4	18	18-28		14	1	int	mmxsh	86	d
SHLD, SHRD	r,r,i/CL	4	14	14	0	14	1	int	mmxsh	386	
SHLD, SHRD	m,r,i/CL	4	18	14	0	14	1	int	mmxsh	386	
BT	r,i	3	0	4	0	2	1	int	mmxsh	386	d
BT	r,r	2	0	4	0	1	1	int	mmxsh	386	d
BT	m,i	4	0	4	0	2	1	int	mmxsh	386	d
BT	m,r	4	12	12	0	12	1	int	mmxsh	386	d
BTR, BTS, BTC	r,i	3	0	6	0	2	1	int	mmxsh	386	
BTR, BTS, BTC	r,r	2	0	6	0	4	1	int	mmxsh	386	
BTR, BTS, BTC	m,i	4	7	18	0	8	1	int	mmxsh	386	
BTR, BTS, BTC	m,r	4	15	14	0	14	1	int	mmxsh	386	
BSF, BSR	r,r	2	0	4	0	2	1	int	mmxsh	386	
BSF, BSR	r,m	3	0	4	0	3	1	int	mmxsh	386	
SETcc	r	3	0	5	0	1	1	int		386	
SETcc	m	4	0	5	0	3	1	int		386	
CLC, STC		3	0	10	0	2				86	d
CMC		3	0	10	0	2				86	
CLD		4	7	52	0	52				86	
STD		4	5	48	0	48				86	
CLI		4	5	35		35				86	
STI		4	12	43		43				86	
<b>Control transfer instructions</b>											
JMP	short/near	1	0	0	0	1	0	alu0	branch	86	
JMP	far	4	28	118		118	0			86	
JMP	r	3	0	4		4	0	alu0	branch	86	
JMP	m(near)	3	0	4		4	0	alu0	branch	86	
JMP	m(far)	4	31	11		11	0			86	
Jcc	short/near	1	0	0		2-4	0	alu0	branch	86	
J(E)CXZ	short	4	4	0		2-4	0	alu0	branch	86	
LOOP	short	4	4	0		2-4	0	alu0	branch	86	
CALL	near	3	0	2		2	0	alu0	branch	86	
CALL	far	4	34				0			86	
CALL	r	4	4	8			0	alu0	branch	86	
CALL	m(near)	4	4	9			0	alu0	branch	86	
CALL	m(far)	4	38				0			86	
RETN		4	0	2			0	alu0	branch	86	
RETN	i	4	0	2			0	alu0	branch	86	
RETF		4	33	11			0			86	



Pentium 4

RETf	i	4	33	11			0			86
IRET		4	48	24			0			86
ENTER	i,0	4	12	26		26				186
ENTER	i,n	4	45+24n			128+16n			186	
LEAVE		4	0	3		3				186
BOUND	m	4	14	14		14				186
INTO		4	5	18		18				86
INT	i	4	84	644						86
<b>String instructions</b>										
LODS		4	3	6		6				86
REP LODS		4	5n	≈ 4n+36				86		
STOS		4	2	6		6				86
REP STOS		4	2n+3	≈ 3n+10				86		
MOVS		4	4	6		4				86
REP MOVS		4	≈ 163+1.1n					86		
SCAS		4	3			6				86
REP SCAS		4	≈ 40+6n	≈ 4n					86	
CMPS		4	5			8				86
REP CMPS		4	≈ 50+8n	≈ 4n					86	
<b>Other</b>										
NOP (90)		1	0	0		0.25	0/1	alu0/1		86
Long NOP (0F 1F)		1	0	0		0.25	0/1	alu0/1		ppro sse2
PAUSE		4	2							
CPUID		4	39-81		200-500			p5		
RDTSC		4	7			80				p5

**Notes:**

- a) Add 1  $\mu$ op if source is a memory operand.
- b) Uses an extra  $\mu$ op (port 3) if SIB byte used. A SIB byte is needed if the memory operand has more than one pointer register, or a scaled index, or ESP is used as base pointer.
- c) Add 1  $\mu$ op if source or destination, but not both, is a high 8-bit register (AH, BH, CH, DH).
- d) Has (false) dependence on the flags in most cases.
- e) Not available on PMMX
- q) Latency is 12 in 16-bit real or virtual mode, 24 in 32-bit protected mode.

**Floating point x87 instructions**

Instruction	Operands	$\mu$ ops	Microcode	Latency	Additional latency	Reciprocal through-put	Port	Execution unit	Subunit	Instruction set	Notes
<b>Move instructions</b>											
FLD	r	1	0	6	0	1	0	mov		87	
FLD	m32/64	1	0	≈ 7	0	1	2	load		87	

Pentium 4

FLD	m80	3	4			6	2	load		87	
FBLD	m80	3	75			90	2	load		87	
FST(P)	r	1	0	6	0	1	0	mov		87	
FST(P)	m32/64	2	0	≈ 7		2-3	0	store		87	
FSTP	m80	3	8			8	0	store		87	
FBSTP	m80	3	311			400	0	store		87	
FXCH	r	1	0	0	0	1	0	mov		87	
FILD	m16	3	3	≈ 10		6	2	load		87	
FILD	m32/64	2	0	≈ 10		1	2	load		87	
FIST	m16	3	0	≈ 10		2-4	0	store		87	
FIST	m32/64	2	0	≈ 10		2-3	0	store		87	
FISTP	m	3	0	≈ 10		2-4	0	store		87	
FLDZ		1	0			2	0	mov		87	
FLD1		2	0			2	0	mov		87	
FCMOVcc	st0,r	4	0	2-4	1	4	1	fp		PPro	e
FFREE	r	3	0			4	0	mov		87	
FINCSTP, FDECSTP		1	0	0	0	1	0	mov		87	
FNSTSW	AX	4	0	11	0	3	1			287	
FSTSW	AX	6	0	11	0	3	1			287	
FNSTSW	m16	4	4			6	0			87	
FNSTCW	m16	4	4			6	0			87	
FLDCW	m16	4	7	(3)		(8)	0,2			87	f
<b>Arithmetic instructions</b>											
FADD(P),FSUB(R)(P)	r	1	0	5	1	1	1	fp	add	87	
FADD,FSUB(R)	m	2	0	5	1	1	1	fp	add	87	
FIADD,FISUB(R)	m16	3	4	6	0	6	1	fp	add	87	
FIADD,FISUB(R)	m32	3	0	5	1	2	1	fp	add	87	
FMUL(P)	r	1	0	7	1	2	1	fp	mul	87	
FMUL	m	2	0	7	1	2	1	fp	mul	87	
FIMUL	m16	3	4	7	1	6	1	fp	mul	87	
FIMUL	m32	3	0	7	1	2	1	fp	mul	87	
FDIV(R)(P)	r	1	0	43	0	43	1	fp	div	87	g, h
FDIV(R)	m	2	0	43	0	43	1	fp	div	87	g, h
FIDIV(R)	m16	3	4	43	0	43	1	fp	div	87	g, h
FIDIV(R)	m32	3	0	43	0	43	1	fp	div	87	g, h
FABS		1	0	2	1	1	1	fp	misc	87	
FCHS		1	0	2	1	1	1	fp	misc	87	
FCOM(P), FUCOM(P)	r	1	0	2	0	1	1	fp	misc	87	
FCOM(P)	m	2	0	2	0	1	1	fp	misc	87	
FCOMPP, FUCOMPP		2	0	2	0	1	1	fp	misc	87	
FCOMI(P)	r	3	0	10	0	3	0,1	fp	misc	PPro	
FICOM(P)	m16	4	4			6	1	fp	misc	87	
FICOM(P)	m32	3	0	2	0	2	1,2	fp	misc	87	
FTST		1	0	2	0	1	1	fp	misc	87	
FXAM		1	0	2	0	1	1	fp	misc	87	
FRNDINT		3	15	23	0	15	0,1			87	
FPREM		6	84	212			1	fp		87	
FPREM1		6	84	212			1	fp		387	

Pentium 4

Pentium 4											
<b>Math</b>											
FSQRT	1	0	43	0	43	1	fp	div	87	g, h	
FLDPI, etc.	2	0			3	1	fp		87		
FSIN	6	≈150	≈180		≈170	1	fp		387		
FCOS	6	≈175	≈207		≈207	1	fp		387		
FSINCOS	7	≈178	≈216		≈211	1	fp		387		
FPTAN	6	≈160	≈230		≈200	1	fp		87		
FPATAN	3	92	≈187		≈153	1	fp		87		
FSCALE	3	24	57		66	1	fp		87		
FEXTRACT	3	15	20		20	1	fp		87		
F2XM1	3	45	≈165		63	1	fp		87		
FYL2X	3	60	≈200		90	1	fp		87		
FYL2XP1	11	134	≈242		≈220	1	fp		87		
<b>Other</b>											
FNOP	1	0	1	0	1	0		mov	87		
(F)WAIT	2	0	0	0	1	0		mov	87		
FNCLEX	4	4			96	1			87		
FNINIT	6	29			172				87		
FNSAVE	4	174	456		420	0,1			87		
FRSTOR	4	96	528		532				87		
FXSAVE	4	69	132		96				sse	i	
FXRSTOR	4	94	208		208				sse	i	

**Notes:**

- e) Not available on PMMX
- f) The latency for FLDCW is 3 when the new value loaded is the same as the value of the control word before the preceding FLDCW, i.e. when alternating between the same two values. In all other cases, the latency and reciprocal throughput is 143.
- g) Latency and reciprocal throughput depend on the precision setting in the F.P. control word. Single precision: 23, double precision: 38, long double precision (default): 43.
- h) Throughput of FP-MUL unit is reduced during the use of the FP-DIV unit.
- i) Takes 6 μops more and 40-80 clocks more when XMM registers are disabled.

**Integer MMX and XMM instructions**

Instruction	Operands	μops	Microcode	Latency	Additional latency	Reciprocal throughput	Port	Execution unit	Subunit	Instruction set	Notes
<b>Move instructions</b>											
MOVD	r32, mm	2	0	5	1	1	0	fp		mmx	
MOVD	mm, r32	2	0	2	0	2	1	mmx	alu	mmx	
MOVD	mm,m32	1	0	≈ 8	0	1	2	load		mmx	
MOVD	r32, xmm	2	0	10	1	2	0	fp		sse2	
MOVD	xmm, r32	2	0	6	1	2	1	mmx	shift	sse2	

## Pentium 4

MOVD	xmm,m32	1	0	≈ 8	0	1	2	load		sse2	
MOVD	m32, r	2	0	≈ 8		2	0,1			mmx	
MOVQ	mm,mm	1	0	6	0	1	0	mov		mmx	
MOVQ	xmm,xmm	1	0	2	1	2	1	mmx	shift	sse2	
MOVQ	r,m64	1	0	≈ 8		1	2	load		mmx	
MOVQ	m64,r	2	0	≈ 8		2	0	mov		mmx	
MOVDQA	xmm,xmm	1	0	6	0	1	0	mov		sse2	
MOVDQA	xmm,m	1	0	≈ 8		1	2	load		sse2	
MOVDQA	m,xmm	2	0	≈ 8		2	0	mov		sse2	
MOVDQU	xmm,m	4	0			2	2	load		sse2	k
MOVDQU	m,xmm	4	6			2	0	mov		sse2	k
MOVDQ2Q	mm,xmm	3	0	8	1	2	0,1	mov-mmx	sse2		
MOVQ2DQ	xmm,mm	2	0	8	1	2	0,1	mov-mmx	sse2		
MOVNTQ	m,mm	3	0			75	0	mov		sse	
MOVNTDQ	m,xmm	2	0			18	0	mov		sse2	
PACKSSWB/DW											
PACKUSWB	mm,r/m	1	0	2	1	1	1	mmx	shift	mmx	a
PACKSSWB/DW											
PACKUSWB	xmm,r/m	1	0	4	1	2	1	mmx	shift	mmx	a
PUNPCKH/LBW/WD/DQ	mm,r/m	1	0	2	1	1	1	mmx	shift	mmx	a
PUNPCKHBW/WD/DQ/QDQ	xmm,r/m	1	0	4	1	2	1	mmx	shift	sse2	a
PUNPCKLBW/WD/DQ/QDQ	xmm,r/m	1	0	2	1	2	1	mmx	shift	sse2	a
PSHUFD	xmm,xmm,i	1	0	4	1	2	1	mmx	shift	sse2	
PSHUFL/HW	xmm,xmm,i	1	0	2	1	2	1	mmx	shift	sse2	
PSHUFW	mm,mm,i	1	0	2	1	1	1	mmx	shift	mmx	
MASKMOVQ	mm,mm	4	4			7	0	mov		sse	
MASKMOVDQU	xmm,xmm	4	6			10	0	mov		sse2	
PMOVMSKB	r32,r	2	0	7	1	3	0,1	mmx-alu0	sse		
PEXTRW	r32,mm,i	3	0	8	1	2	1	mmx-int	sse		
PEXTRW	r32,xmm,i	3	0	9	1	2	1	mmx-int	sse2		
PINSRW	mm,r32,i	2	0	3	1	2	1	int-mmx	sse		
PINSRW	xmm,r32,i	2	0	4	1	2	1	int-mmx	sse2		
<b>Arithmetic instructions</b>											
PADDB/W/D											
PADD(U)SB/W	r,r/m	1	0	2	1	1,2	1	mmx	alu	mmx	a,j
PSUBB/W/D											
PSUB(U)SB/W	r,r/m	1	0	2	1	1,2	1	mmx	alu	mmx	a,j
PADDQ, PSUBQ	mm,r/m	1	0	2	1	1	1	mmx	alu	sse2	a
PADDQ, PSUBQ	xmm,r/m	1	0	4	1	2	1	fp	add	sse2	a
PCMPEQB/W/D											
PCMPGTB/W/D	r,r/m	1	0	2	1	1,2	1	mmx	alu	mmx	a,j
PMULLW PMULHW	r,r/m	1	0	6	1	1,2	1	fp	mul	mmx	a,j
PMULHUW	r,r/m	1	0	6	1	1,2	1	fp	mul	sse	a,j
PMADDWD	r,r/m	1	0	6	1	1,2	1	fp	mul	mmx	a,j
PMULUDQ	r,r/m	1	0	6	1	1,2	1	fp	mul	sse2	a,j
PAVGB/W	r,r/m	1	0	2	1	1,2	1	mmx	alu	sse	a,j
PMIN/MAXUB	r,r/m	1	0	2	1	1,2	1	mmx	alu	sse	a,j

Pentium 4

PMIN/MAXSW	r,r/m	1	0	2	1	1,2	1	mmx	alu	sse	a,j
PSADBW	r,r/m	1	0	4	1	1,2	1	mmx	alu	sse	a,j
<b>Logic</b>											
PAND, PANDN	r,r/m	1	0	2	1	1,2	1	mmx	alu	mmx	a,j
POR, PXOR	r,r/m	1	0	2	1	1,2	1	mmx	alu	mmx	a,j
PSLL/RLW/D/Q, PSRAW/D	r,i/r/m	1	0	2	1	1,2	1	mmx	shift	mmx	a,j
PSLLDQ, PSRLDQ	xmm,i	1	0	4	1	2	1	mmx	shift	sse2	a
<b>Other</b>											
EMMS		4	11	12		12	0			mmx	

**Notes:**

- a) Add 1  $\mu$ op if source is a memory operand.
- j) Reciprocal throughput is 1 for 64 bit operands, and 2 for 128 bit operands.
- k) It may be advantageous to replace this instruction by two 64-bit moves

**Floating point XMM instructions**

Instruction	Operands	$\mu$ ops	Microcode	Latency	Additional latency	Reciprocal throughput	Port	Execution unit	Subunit	Instruction set	Notes
<b>Move instructions</b>											
MOVAPS/D	r,r	1	0	6	0	1	0	mov		sse	
MOVAPS/D	r,m	1	0	$\approx 7$	0	1	2			sse	
MOVAPS/D	m,r	2	0	$\approx 7$		2	0			sse	
MOVUPS/D	r,r	1	0	6	0	1	0	mov		sse	
MOVUPS/D	r,m	4	0			2	2			sse	k
MOVUPS/D	m,r	4	6			8	0			sse	k
MOVSS	r,r	1	0	2	0	2	1	mmx	shift	sse	
MOVSD	r,r	1	0	2	1	2	1	mmx	shift	sse	
MOVSS, MOVSD	r,m	1	0	$\approx 7$	0	1	2			sse	
MOVSS, MOVSD	m,r	2	0			2	0			sse	
MOVHLPS	r,r	1	0	4	0	2	1	mmx	shift	sse	
MOVLHPS	r,r	1	0	2	0	2	1	mmx	shift	sse	
MOVHPS/D, MOVLPS/D	r,m	3	0			4	2			sse	
MOVHPS/D, MOVLPS/D	m,r	2	0			2	0			sse	
MOVNTPS/D	m,r	2	0			4	0			sse/2	
MOVMSKPS/D	r32,r	2	0	6	1	3	1	fp		sse	
SHUFPS/D	r,r/m,i	1	0	4	1	2	1	mmx	shift	sse	
UNPCKHPS/D	r,r/m	1	0	4	1	2	1	mmx	shift	sse	
UNPCKLPS/D	r,r/m	1	0	2	1	2	1	mmx	shift	sse	

Pentium 4

<b>Conversion</b>											
CVTSP2PD	r,r/m	4	0	7	1	4	1	mmx	shift	sse2	a
CVTPD2PS	r,r/m	2	0	10	1	2	1	fp-mmx	sse2	a	
CVTSD2SS	r,r/m	4	0	14	1	6	1	mmx	shift	sse2	a
CVTSS2SD	r,r/m	4	0	10	1	6	1	mmx	shift	sse2	a
CVTDQ2PS	r,r/m	1	0	4	1	2	1	fp		sse2	a
CVTDQ2PD	r,r/m	3	0	9	1	4	1	mmx-fp	sse2	a	
CVT(T)PS2DQ	r,r/m	1	0	4	1	2	1	fp		sse2	a
CVT(T)PD2DQ	r,r/m	2	0	9	1	2	1	fp-mmx	sse2	a	
CVTPI2PS	xmm,mm	4	0	10	1	4	1	mmx		sse	a
CVTPI2PD	xmm,mm	4	0	11	1	5	1	fp-mmx	sse2	a	
CVT(T)PS2PI	mm,xmm	3	0	7	0	2	0,1	fp-mmx	sse	a	
CVT(T)PD2PI	mm,xmm	3	0	11	1	3	0,1	fp-mmx	sse2	a	
CVTSI2SS	xmm,r32	3	0	10	1	3	1	fp-mmx	sse	a	
CVTSI2SD	xmm,r32	4	0	15	1	6	1	fp-mmx	sse2	a	
CVT(T)SD2SI	r32,xmm	2	0	8	1	2.5	1	fp		sse2	a
CVT(T)SS2SI	r32,xmm	2	0	8	1	2.5	1	fp		sse	a
<b>Arithmetic</b>											
ADDPs/D ADDSS/D	r,r/m	1	0	4	1	2	1	fp	add	sse	a
SUBPs/D SUBSS/D	r,r/m	1	0	4	1	2	1	fp	add	sse	a
MULPs/D MULSS/D	r,r/m	1	0	6	1	2	1	fp	mul	sse	a
DIVSS	r,r/m	1	0	23	0	23	1	fp	div	sse	a,h
DIVPS	r,r/m	1	0	39	0	39	1	fp	div	sse	a,h
DIVSD	r,r/m	1	0	38	0	38	1	fp	div	sse2	a,h
DIVPD	r,r/m	1	0	69	0	69	1	fp	div	sse2	a,h
RCPpS RCPSS	r,r/m	2	0	4	1	4	1	mmx		sse	a
MAXPs/D MAXSS/DMINPs/D MINSS/D	r,r/m	1	0	4	1	2	1	fp	add	sse	a
CMPccPs/D CMPccSS/D	r,r/m	1	0	4	1	2	1	fp	add	sse	a
COMISS/D UCOMISS/D	r,r/m	2	0	6	1	3	1	fp	add	sse	a
<b>Logic</b>											
ANDPs/D ANDNPs/D ORPs/D XORPs/D	r,r/m	1	0	2	1	2	1	mmx	alu	sse	a
<b>Math</b>											
SQRTSS	r,r/m	1	0	23	0	23	1	fp	div	sse	a,h
SQRTPS	r,r/m	1	0	39	0	39	1	fp	div	sse	a,h
SQRTSD	r,r/m	1	0	38	0	38	1	fp	div	sse2	a,h
SQRTPD	r,r/m	1	0	69	0	69	1	fp	div	sse2	a,h
RSQRTSS	r,r/m	2	0	4	1	3	1	mmx		sse	a
RSQRTPS	r,r/m	2	0	4	1	4	1	mmx		sse	a
<b>Other</b>											
LDMXCSR	m	4	8	98		100	1			sse	
STMXCSR	m	4	4			6	1			sse	

**Notes:**

a) Add 1  $\mu$ op if source is a memory operand.

## Pentium 4

- h)** Throughput of FP-MUL unit is reduced during the use of the FP-DIV unit.
- k)** It may be advantageous to replace this instruction by two 64-bit moves.

## **Intel Pentium 4 w. EM64T (Prescott)**

### List of instruction timings and $\mu$ op breakdown

*Explanation of column headings:*

- Instruction:** Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc.
- Operands:** i = immediate constant, r = any register, r32 = 32-bit register, etc., mm = 64 bit mmx register, xmm = 128 bit xmm register, sr = segment register, m = any memory operand including indirect operands, m64 means 64-bit memory operand, etc., mabs = memory operand with 64-bit absolute address.
- $\mu$ ops:** Number of  $\mu$ ops issued from instruction decoder and stored in trace cache.
- Microcode:** Number of additional  $\mu$ ops issued from microcode ROM.
- Latency:** This is the delay that the instruction generates in a dependency chain if the next dependent instruction starts in the same execution unit. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's, infinity and exceptions increase the delays. The latency of moves to and from memory cannot be measured accurately because of the problem with memory intermediates explained above under "How the values were measured".
- Additional latency:** This number is added to the latency if the next dependent instruction is in a different execution unit. There is no additional latency between ALU0 and ALU1.
- Reciprocal throughput:** This is also called issue latency. This value indicates the number of clock cycles from the execution of an instruction begins to a subsequent independent instruction can begin to execute in the same execution subunit. A value of 0.25 indicates 4 instructions per clock cycle in one thread.
- Port:** The port through which each  $\mu$ op goes to an execution unit. Two independent  $\mu$ ops can start to execute simultaneously only if they are going through different ports.
- Execution unit:** Use this information to determine additional latency. When an instruction with more than one  $\mu$ op uses more than one execution unit, only the first and the last execution unit is listed.
- Execution subunit:** Throughput measures apply only to instructions executing in the same subunit.
- Instruction set** Indicates the compatibility of an instruction with other 80x86 family microprocessors. The instruction can execute on microprocessors that support the instruction set indicated.

### Integer instructions

Instruction	Operands	$\mu$ ops	Microcode	Latency	Additional latency	Reciprocal throughput	Port	Execution unit	Subunit	Instruction set	Notes
<b>Move instructions</b>											
MOV	r,r	1	0	1	0	0.25	0/1	alu0/1		86	c
MOV	r8/16/32,i	1	0	1	0	0.25	0/1	alu0/1		86	
MOV	r64,i32	1	0		0	0.5	0/1	alu0/1		x64	



Prescott

MOV	r64,i64	2	0		0	1	1	alu1	x64	
MOV	r8/16,m	2	0	3	0	1	2	load	86	
MOV	r32/64,m	1	0	2	0	1	2	load	86	
MOV	m,r	1	0			2	0	store	86	b,c
MOV	m,i	2	0			2	0,3	store	86	
MOV	m64,i32	2	0			2	0,3	store	x64	
MOV	r,sr	1	2			8			86	
MOV	sr,r/m	1	8			27			86	a,q
MOV	r,mabs	3	0			1			x64	l
MOV	mabs,r	3	0			2			x64	l
MOVNTI	m,r32	2	0			2			sse2	
MOVZX	r,r	1	0	1	0	0.25	0/1	alu0/1	386	c
MOVZX	r16,r8	2	0	2	0	1	0/1	alu0/1	386	c
MOVZX	r,m	1	0	2	0	1	2	load	386	
MOVSX	r16,r8	2	0	2	0	1	0	alu0	386	a,c,o
MOVSX	r32/64,r8/16	1	0	1	0	0.5	0	alu0	386	a,c,o
MOVSX	r,m	2	0	3	0	1	2	load	386	
MOVSLD	r64,r32	1	0	1	0	0.5	0	alu0	x64	a
CMOVCc	r,r/m	3	0	9.5	0	3			PPro	a,e
XCHG	r,r	3	0	2	0	1	0/1	alu0/1	86	
XCHG	r,m	2	6	≈100					86	
XLAT		4	0	6					86	
PUSH	r	2	0	2		2			86	
PUSH	i	2	0	2		2			186	
PUSH	m	3	0	2		2			86	
PUSH	sr	1	3			9			86	
PUSHF(D/Q)		1	3			9			86	
PUSHA(D)		1	9			16			186	m
POP	r	2	0	1	0	1			86	
POP	m	2	6			10			86	
POP	sr	1	8			30			86	
POPF(D/Q)		1	8			70			86	
POPA(D)		2	16			15			186	m
LEA	r,[m]	1	0			0.25	0/1	alu0/1	86	p
LEA	r,[r+r/i]	1	0	2.5	0	0.25	0/1	alu0/1	86	
LEA	r,[r+r+i]	2	0	3.5	0	0.5	0/1	alu0/1	86	
LEA	r,[r*i]	3	0	3.5	0	1	1	alu	386	
LEA	r,[r+r*i]	2	0	3.5	0	1	0,1	alu0,1	386	
LEA	r,[r+r*i+i]	3	0	3.5	0	1	1	alu	386	
LAHF		1	0	4	0		1	int	86	n
SAHF		1	0	5	0		0/1	alu0/1	86	d,n
SALC		2	0		0	1	1	int	86	m
LDS, LES, ...	r,m	2	10			28			86	m
LODS		1	3	8		8			86	
REP LODS		1	5n	≈ 4n+50					86	
STOS		1	2	8		8			86	
REP STOS		1	2.5n	≈ 3n					86	
MOVS		1	4	8		8			86	
REP MOVSB		9	≈.3n	≈.3n					86	
REP MOVSW		1	≈.5-1.1n	≈.6-1.4n					86	

Prescott

REP MOVSD		1	≈1.1n	≈ 1.4 n				86			
REP MOVSQ		1	≈1.1n	≈ 1.4 n				x64			
BSWAP	r	1	0	1	0	1		alu		486	
IN, OUT	r,r/i	1	52			>1000			86		
PREFETCHNTA	m	1	0			1				sse	
PREFETCHT0/1/2	m	1	0			1				sse	
SFENCE		1	2			50				sse	
LFENCE		1	2			50				sse2	
MFENCE		1	4			124				sse2	
<b>Arithmetic instructions</b>											
ADD, SUB	r,r	1	0	1	0	0.25	0/1	alu0/1		86	c
ADD, SUB	r,m	2	0	1	0	1				86	c
ADD, SUB	m,r	3	0	5		2				86	c
ADC, SBB	r,r/i	3	0	10	0	10	1	int,alu		86	
ADC, SBB	r,m	2	5	10	0	10	1	int,alu		86	
ADC, SBB	m,r	2	6	20		10				86	
ADC, SBB	m,i	3	5	22		10				86	
CMP	r,r	1	0	1	0	0.25	0/1	alu0/1		86	c
CMP	r,m	2	0	1	0	1				86	c
INC, DEC	r	2	0	1	0	0.5	0/1	alu0/1		86	
INC, DEC	m	4	0	5		3				86	
NEG	r	1	0	1	0	0.5	0	alu0		86	
NEG	m	3	0	5		3				86	
AAA, AAS		1	10	26						86	m
DAA, DAS		1	16	29						86	m
AAD		2	5	13			1	int	mul	86	m
AAM		2	17	71			1	int	fpdiv	86	m
MUL, IMUL	r8	1	0	10	0		1	int	mul	86	
MUL, IMUL	r16	4	0	11	0		1	int	mul	86	
MUL, IMUL	r32	3	0	11	0		1	int	mul	86	
MUL, IMUL	r64	1	5	11	0		1	int	mul	x64	
MUL, IMUL	m8	2	0	10	0		1	int	mul	86	
MUL, IMUL	m16	2	5	11	0		1	int	mul	86	
MUL, IMUL	m32	3	0	11	0		1	int	mul	86	
MUL, IMUL	m64	2	6	11	0		1	int	mul	x64	
IMUL	r16,r16	1	0	10	0	2.5	1	int	mul	386	
IMUL	r16,r16,i	2	0	11	0	2.5	1	int	mul	186	
IMUL	r32,r32	1	0	10	0	2.5	1	int	mul	386	
IMUL	r32,(r32),i	1	0	10	0	2.5	1	int	mul	386	
IMUL	r64,r64	1	0	10	0	2.5	1	int	mul	x64	
IMUL	r64,(r64),i	1	0	10	0	2.5	1	int	mul	x64	
IMUL	r16,m16	2	0	10	0	2.5	1	int	mul	386	
IMUL	r32,m32	2	0	10	0	2.5	1	int	mul	386	
IMUL	r64,m64	2	0	10	0	2.5	1	int	mul	x64	
IMUL	r,m,i	3	0	10	0	1-2.5	1	int	mul	186	
DIV	r8/m8	1	20	74	0	34	1	int	fpdiv	86	a
DIV	r16/m16	1	19	73	0	34	1	int	fpdiv	86	a
DIV	r32/m32	1	21	76	0	34	1	int	fpdiv	386	a
DIV	r64/m64	1	31	63	0	52	1	int	fpdiv	x64	a

Prescott

IDIV	r8/m8	1	21	76	0	34	1	int	fpdiv	86	a
IDIV	r16/m16	1	19	79	0	34	1	int	fpdiv	86	a
IDIV	r32/m32	1	19	79	0	34	1	int	fpdiv	386	a
IDIV	r64/m64	1	58	96	0	91	1	int	fpdiv	x64	a
CBW		2	0	2	0	1	0	alu0		86	
CWD		2	0	2	0	1	0/1	alu0/1		86	
CDQ		1	0	1	0	1	0/1	alu0/1		386	
CQO		1	0	7	0	1	0/1	alu0/1		x64	
CWDE		2	0	2	0	1	0/1	alu0/1		386	
CDQE		1	0	1	0	1	0/1	alu0/1		x64	
SCAS		1	3		0	8				86	
REP SCAS		1	$\approx 54+6n$		$\approx 4n$				86		
CMPS		1	5			10				86	
REP CMPS		1	$\approx 81+8n$		$\approx 5n$				86		
<b>Logic</b>											
AND, OR, XOR	r,r	1	0	1	0	0.5	0	alu0		86	c
AND, OR, XOR	r,m	2	0	1	0	1				86	c
AND, OR, XOR	m,r	3	0	5		2				86	c
TEST	r,r	1	0	1	0	0.5	0	alu0		86	c
TEST	r,m	2	0	1	0	1				86	c
NOT	r	1	0	1	0	0.5	0	alu0		86	
NOT	m	3	0	5		2				86	
SHL	r,i	1	0	1	0	0.5	1	alu1		186	
SHR, SAR	r8/16/32,i	1	0	1	0	0.5	1	alu1		186	
SHR, SAR	r64,i	1	0	7	0	2	1	alu1		x64	
SHL	r,CL	2	0	2	0	2	1	alu1		86	
SHR, SAR	r8/16/32,CL	2	0	2	0	2	1	alu1		86	
SHR, SAR	r64,CL	2	0	8	0		1	alu1		x64	
ROL, ROR	r8/16/32,i	1	0	1	0	1	1	alu1		186	d
ROL, ROR	r64,i	1	0	7	0	7	1	alu1		x64	d
ROL, ROR	r8/16/32,CL	2	0	2	0	2	1	alu1		86	d
ROL, ROR	r64,CL	2	0	8	0	8	1	alu1		x64	d
RCL, RCR	r,1	1	0	7	0	7	1	alu1		86	d
RCL	r,i	2	11	31	0	31	1	alu1		186	d
RCR	r,i	2	11	25	0	25	1	alu1		186	d
RCL	r,CL	1	11	31	0	31	1	alu1		86	d
RCR	r,CL	1	11	25	0	25	1	alu1		86	d
SHL, SHR, SAR	m8/16/32,i	3	6	10	0		1	alu1		86	
ROL, ROR	m8/16/32,i	3	6	10	0		1	alu1		86	d
SHL, SHR, SAR	m8/16/32,cl	2	6	10	0		1	alu1		86	
ROL, ROR	m8/16/32,cl	2	6	10	0		1	alu1		86	d
RCL, RCR	m8/16/32,1	2	5	27	0	27	1	alu1		86	d
RCL, RCR	m8/16/32,i	3	13	38	0	38	1	alu1		86	d
RCL, RCR	m8/16/32,cl	2	13	37	0	37	1	alu1		86	d
SHLD, SHRD	r8/16/32,r,i	3	0	8	0	7	1	alu1		386	
SHLD	r64,r64,i	4	5	10	0		1	alu1		x64	
SHRD	r64,r64,i	3	7	10	0		1	alu1		x64	
SHLD, SHRD	r8/16/32,r,cl	4	0	9	0	8	1	alu1		386	
SHLD	r64,r64,cl	4	5	14	0		1	alu1		x64	

Prescott

SHRD	r64,r64,cl	3	8	12	0		1	alu1		x64	
SHLD, SHRD	m,r,i	3	8	20	0	10	1	alu1		386	
SHLD, SHRD	m,r,CL	2	8	20	0	10	1	alu1		386	
BT	r,i	1	0	8	0	8	1	alu1		386	d
BT	r,r	2	0	9	0	9	1	alu1		386	d
BT	m,i	3	0	8	0	8	1	alu1		386	d
BT	m,r	2	7	10	0	10	1	alu1		386	d
BTR, BTS, BTC	r,i	1	0	8	0	8	1	alu1		386	
BTR, BTS, BTC	r,r	2	0	9	0	9	1	alu1		386	
BTR, BTS, BTC	m,i	3	6	28	0	10	1	alu1		386	
BTR, BTS, BTC	m,r	2	10	14	0	14	1	alu1		386	
BSF, BSR	r,r/m	2	0	16	0	4	1	alu1		386	
SETcc	r	2	0	9	0	1	1	int		386	
SETcc	m	3	0	9	0	2	1	int		386	
CLC, STC		2	0		0	8				86	d
CMC		3	0	15	0					86	
CLD, STD		1	8		0	53				86	
<b>Control transfer instructions</b>											
JMP	short/near	1	0	0	0	1	0	alu0	branch	86	
JMP	far	2	25			154	0			86	m
JMP	r	3	0			15	0	alu0	branch	86	
JMP	m(near)	3	0			10	0	alu0	branch	86	
JMP	m(far)	2	28			157	0			86	
Jcc	short/near	1	0			2-4	0	alu0	branch	86	
J(E)CXZ	short	4	0			4	0	alu0	branch	86	
LOOP	short	4	0			4	0	alu0	branch	86	
CALL	near	3	0			7	0	alu0	branch	86	
CALL	far	3	29			160	0			86	m
CALL	r	4	0			7	0	alu0	branch	86	
CALL	m(near)	4	0			9	0	alu0	branch	86	
CALL	m(far)	2	32			160	0			86	
RETN		4	0			7	0	alu0	branch	86	
RETN	i	4	0			7	0	alu0	branch	86	
RETF		1	30			160	0			86	
RETF	i	2	30			160	0			86	
IRET		1	49			325	0			86	
BOUND	m	2	11			12				186	m
INT	i	2	67			470				86	
INTO		1	4			26				86	m
<b>Other</b>											
NOP (90)		1	0	0		0.25	0/1	alu0/1		86	
Long NOP (0F 1F)		1	0	0		0.25	0/1	alu0/1		ppro	
PAUSE		1	2			50				sse2	
LEAVE		4	0	5		5				186	
CLI		1	5			52				86	
STI		1	11			64				86	
CPUID		1	49-90			300-500		p5			
RDTSC		1	12			100				p5	

Prescott

RDPMC (bit 31 = 1)	1	37			100				p5	
RDPMC (bit 31 = 0)	4	154			240				p5	
MONITOR										(sse3)
MWAIT										(sse3)

**Notes:**

- a) Add 1  $\mu$ op if source is a memory operand.
- b) Uses an extra  $\mu$ op (port 3) if SIB byte used.
- c) Add 1  $\mu$ op if source or destination, but not both, is a high 8-bit register (AH, BH, CH, DH).
- d) Has (false) dependence on the flags in most cases.
- e) Not available on PMMX
- l) Move accumulator to/from memory with 64 bit absolute address (opcode A0 - A3).
- m) Not available in 64 bit mode.
- n) Not available in 64 bit mode on some processors.
- o) MOVSX uses an extra  $\mu$ op if the destination register is smaller than the biggest register size available. Use a 32 bit destination register in 16 bit and 32 bit mode, and a 64 bit destination register in 64 bit mode for optimal performance.
- p) LEA with a direct memory operand has 1  $\mu$ op and a reciprocal throughput of 0.25. This also applies if there is a RIP-relative address in 64-bit mode. A sign-extended 32-bit direct memory operand in 64-bit mode without RIP-relative address takes 2  $\mu$ ops because of the SIB byte. The throughput is 1 in this case. You may use a MOV instead.
- q) These values are measured in 32-bit mode. In 16-bit real mode there is 1 microcode  $\mu$ op and a reciprocal throughput of 17.

**Floating point x87 instructions**

Instruction	Operands	$\mu$ ops	Microcode	Latency	Additional latency	Port	Reciprocal throughput	Execution unit	Subunit	Instruction set	Notes
<b>Move instructions</b>											
FLD	r	1	0	7	0	1	0	mov		87	
FLD	m32/64	1	0		0	1	2	load		87	
FLD	m80	3	3			8	2	load		87	
FBLD	m80	3	74			90	2	load		87	
FST(P)	r	1	0	7	0	1	0	mov		87	
FST(P)	m32/64	2	0	7		2	0	store		87	
FSTP	m80	3	6			10	0	store		87	
FBSTP	m80	3	311			400	0	store		87	
FXCH	r	1	0	0	0	1	0	mov		87	
FILD	m16	3	2			8	2	load		87	
FILD	m32/64	2	0			2	2	load		87	
FIST(P)	m	3	0			2.5	0	store		87	
FISTTP	m	3	0			2.5	0	store		sse3	
FLDZ		1	0			2	0	mov		87	

Prescott

FLD1		2	0			2	0	mov		87	
FCMOVcc	st0,r	4	0	5	1	4	1	fp		PPro	e
FFREE	r	3	0			3	0	mov		87	
FINCSTP, FDECSTP		1	0	0	0	1	0	mov		87	
FNSTSW	AX	4	0		0	3	1			287	
FSTSW	AX	6	0		0	3	1			287	
FNSTSW	m16	2	3			8	0			87	
FNSTCW	m16	4	0			3	0			87	
FLDCW	m16	3	6			10	0,2			87	f
<b>Arithmetic instructions</b>											
FADD(P),FSUB(R)(P)	r	1	0	6	1	1	1	fp	add	87	
FADD,FSUB(R)	m	2	0	6	1	1	1	fp	add	87	
FIADD,FISUB(R)	m16	3	3	7	1	6	1	fp	add	87	
FIADD,FISUB(R)	m32	3	0	6	1	2	1	fp	add	87	
FMUL(P)	r	1	0	8	1	2	1	fp	mul	87	
FMUL	m	2	0	8	1	2	1	fp	mul	87	
FIMUL	m16	3	3	8	1	8	1	fp	mul	87	
FIMUL	m32	3	0	8	1	3	1	fp	mul	87	
FDIV(R)(P)	r	1	0	45	1	45	1	fp	div	87	g,h
FDIV(R)	m	2	0	45	1	45	1	fp	div	87	g,h
FIDIV(R)	m16	3	3	45	1	45	1	fp	div	87	g,h
FIDIV(R)	m32	3	3	45	1	45	1	fp	div	87	g,h
FABS		1	0	3	1	1	1	fp	misc	87	
FCHS		1	0	3	1	1	1	fp	misc	87	
FCOM(P), FUCOM(P)	r	1	0	3	0	1	1	fp	misc	87	
FCOM(P)	m	2	0	3	0	1	1	fp	misc	87	
FCOMPP, FUCOMPP		2	0	3	0	1	1	fp	misc	87	
FCOMI(P)	r	3	0			3	0,1	fp	misc	PPro	
FICOM(P)	m16	3	3			8	1	fp	misc	87	
FICOM(P)	m32	3	0			2	1,2	fp	misc	87	
FTST		1	0			1	1	fp	misc	87	
FXAM		1	0			1	1	fp	misc	87	
FRNDINT		3	14	28	1	16	0,1			87	
FPREM		8	86	220	1		1	fp		87	
FPREM1		9	92	220	1		1	fp		387	
<b>Math</b>											
FSQRT		1	0	45	1	45	1	fp	div	87	g,h
FLDPI, etc.		2	0			2	1	fp		87	
FSIN, FCOS		3	≈100	≈200		≈200	1	fp		387	
FSINCOS		5	≈150	≈200		≈200	1	fp		387	
FPTAN		8	≈170	≈270		≈270	1	fp		87	
FPATAN		4	97	≈250		≈250	1	fp		87	
FSCALE		3	25	96			1	fp		87	
FEXTRACT		4	16	27			1	fp		87	
F2XM1		3	190	≈270			1	fp		87	
FYL2X		3	63	≈170			1	fp		87	
FYL2XP1		3	58	≈170			1	fp		87	

Prescott

Other											
FNOP	1	0	1	0	1	0			mov	87	
(F)WAIT	2	0	0	0	1	0			mov	87	
FNCLEX	1	4			120	1				87	
FNINIT	1	30			200					87	
FNSAVE	2	181	500				0,1			87	
FRSTOR	2	96	570							87	
FXSAVE	2	121			160					sse	i
FXRSTOR	2	118			244					sse	i

**Notes:**

- e) Not available on PMMX
- f) The latency for FLDCW is 3 when the new value loaded is the same as the value of the control word before the preceding FLDCW, i.e. when alternating between the same two values. In all other cases, the latency and reciprocal throughput is > 100.
- g) Latency and reciprocal throughput depend on the precision setting in the F.P. control word. Single precision: 32, double precision: 40, long double precision (default): 45.
- h) Throughput of FP-MUL unit is reduced during the use of the FP-DIV unit.
- i) Takes fewer microcode  $\mu$ ops when XMM registers are disabled, but the throughput is the same.

**Integer MMX and XMM instructions**

Instruction	Operands	$\mu$ ops	Microcode	Latency	Additional latency	Reciprocal throughput	Port	Execution unit	Subunit	Instruction set	Notes
<b>Move instructions</b>											
MOVD	r32, mm	2	0	6	1	1	0	fp		mmx	
MOVD	mm, r32	1	0	3	1	1	1	mmx	alu	mmx	
MOVD	mm,m32	1	0			1	2	load		mmx	
MOVD	r32, xmm	1	0	7	1	1	0	fp		sse2	
MOVD	xmm, r32	2	0	4	1	2	1	mmx	shift	sse2	
MOVD	xmm,m32	1	0			1	2	load		sse2	
MOVD	m32, r	2	0			2	0,1			mmx	
MOVQ	mm,mm	1	0	7	0	1	0	mov		mmx	
MOVQ	xmm,xmm	1	0	2	1	2	1	mmx	shift	sse2	
MOVQ	r,m64	1	0			1	2	load		mmx	
MOVQ	m64,r	2	0			2	0	mov		mmx	
MOVDQA	xmm,xmm	1	0	7	0	1	0	mov		sse2	
MOVDQA	xmm,m	1	0			1	2	load		sse2	
MOVDQA	m,xmm	2	0			2	0	mov		sse2	
MOVDQU	xmm,m	4	0				23	load		sse2	k
MOVDQU	m,xmm	4	2			8	0	mov		sse2	k
LDDQU	xmm,m	4	0			2.5	2	load		sse3	
MOVDQ2Q	mm,xmm	3	0	10	1	2	0,1	mov-mmx	sse2		

Prescott

MOVQ2DQ	xmm,mm	2	0	10	1	2	0,1	mov-mmx	sse2		
MOVNTQ	m,mm	3	0			4	0	mov		sse	
MOVNTDQ	m,xmm	2	0			4	0	mov		sse2	
MOVDDUP	xmm,xmm	1	0	2	1	2	1	mmx	shift	sse3	
MOVSHDUP											
MOVSLDUP	xmm,xmm	1	0	4	1	2	1	mmx	shift	sse3	
PACKSSWB/DW											
PACKUSWB	mm,r/m	1	0	2	1	2	1	mmx	shift	mmx	a
PACKSSWB/DW											
PACKUSWB	xmm,r/m	1	0	4	1	4	1	mmx	shift	mmx	a
PUNPCKH/LBW/WD/ DQ	mm,r/m	1	0	2	1	2	1	mmx	shift	mmx	a
PUNPCKHBW/WD/DQ/ QDQ	xmm,r/m	1	0	4	1	4	1	mmx	shift	sse2	a
PUNPCKLBW/WD/DQ/Q DQ	xmm,r/m	1	0	2	1	2	1	mmx	shift	sse2	a
PSHUFD	xmm,xmm,i	1	0	4	1	2	1	mmx	shift	sse2	
PSHUFL/HW	xmm,xmm,i	1	0	2	1	2	1	mmx	shift	sse	
PSHUFW	mm,mm,i	1	0	2	1	1	1	mmx	shift	sse	
MASKMOVQ	mm,mm	1	4			10	0	mov		sse	
MASKMOVDQU	xmm,xmm	1	6			12	0	mov		sse2	
PMOVMSKB	r32,r	2	0	7		3	0,1	mmx-alu0	sse		
PEXTRW	r32,mm,i	2	0	7		2	1	mmx-int	sse		
PEXTRW	r32,xmm,i	2	0	7		3	1	mmx-int	sse2		
PINSRW	r,r32,i	2	0	4		2	1	int-mmx	sse		
<b>Arithmetic instructions</b>											
PADDB/W/D											
PADD(U)SB/W	r,r/m	1	0	2	1	1,2	1	mmx	alu	mmx	a,j
PSUBB/W/D											
PSUB(U)SB/W	r,r/m	1	0	2	1	1,2	1	mmx	alu	mmx	a,j
PADDQ, PSUBQ	mm,r/m	1	0	2	1	1	1	mmx	alu	sse2	a
PADDQ, PSUBQ	xmm,r/m	1	0	5	1	2	1	fp	add	sse2	a
PCMPEQB/W/D											
PCMPGTB/W/D	r,r/m	1	0	2	1	1,2	1	mmx	alu	mmx	a,j
PMULLW PMULHW	r,r/m	1	0	7	1	1,2	1	fp	mul	mmx	a,j
PMULHUW	r,r/m	1	0	7	1	1,2	1	fp	mul	sse	a,j
PMADDWD	r,r/m	1	0	7	1	1,2	1	fp	mul	mmx	a,j
PMULUDQ	r,r/m	1	0	7	1	1,2	1	fp	mul	sse2	a,j
PAVGB/W	r,r/m	1	0	2	1	1,2	1	mmx	alu	sse	a,j
PMIN/MAXUB	r,r/m	1	0	2	1	1,2	1	mmx	alu	sse	a,j
PMIN/MAXSW	r,r/m	1	0	2	1	1,2	1	mmx	alu	sse	a,j
PSADBW	r,r/m	1	0	4	1	1,2	1	mmx	alu	sse	a,j
<b>Logic</b>											
PAND, PANDN	r,r/m	1	0	2	1	1,2	1	mmx	alu	mmx	a,j
POR, PXOR	r,r/m	1	0	2	1	1,2	1	mmx	alu	mmx	a,j
PSLL/RLW/D/Q, PSRAW/D	r,i/r/m	1	0	2	1	1,2	1	mmx	shift	mmx	a,j
PSLLDQ, PSRLDQ	xmm,i	1	0	4	1	2	1	mmx	shift	sse2	



<b>Other</b>										
EMMS		10	10			12	0			mmx

**Notes:**

- a) Add 1  $\mu$ op if source is a memory operand.
- j) Reciprocal throughput is 1 for 64 bit operands, and 2 for 128 bit operands.
- k) It may be advantageous to replace this instruction by two 64-bit moves or LD-DQU.

**Floating point XMM instructions**

Instruction	Operands	$\mu$ ops	Microcode	Latency	Additional latency	Reciprocal throughput	Port	Execution unit	Subunit	Instruction set	Notes
<b>Move instructions</b>											
MOVAPS/D	r,r	1	0	7	0	1	0	mov		sse	
MOVAPS/D	r,m	1	0		0	1	2			sse	
MOVAPS/D	m,r	2	0			2	0			sse	
MOVUPS/D	r,r	1	0	7	0	1	0	mov		sse	
MOVUPS/D	r,m	4	0			2	2			sse	k
MOVUPS/D	m,r	4	2			8	0			sse	k
MOVSS	r,r	1	0	2	1	2	1	mmx	shift	sse	
MOVSD	r,r	1	0	4	1	2	1	mmx	shift	sse	
MOVSS, MOVSD	r,m	1	0		0	1	2			sse	
MOVSS, MOVSD	m,r	2	0			2	0			sse	
MOVHLPS	r,r	1	0	4	1	2	1	mmx	shift	sse	
MOVLHPS	r,r	1	0	2	1	2	1	mmx	shift	sse	
MOVHPS/D, MOVLPS/D	r,m	2	0			2	2			sse	
MOVHPS/D, MOVLPS/D	m,r	2	0			2	0			sse	
MOVSH/LDUP	r,r	1	0	4	1	2	1			sse3	
MOVDDUP	r,r	1	0	2	1	2	1			sse3	
MOVNTPS/D	m,r	2	0			4	0			sse	
MOVMSKPS/D	r32,r	2	0	5	1	3	1	fp		sse	
SHUFPS/D	r,r/m,i	1	0	4	1	2	1	mmx	shift	sse	
UNPCKHPS/D	r,r/m	2	0	4	1	2	1	mmx	shift	sse	
UNPCKLPS/D	r,r/m	1	0	2	1	2	1	mmx	shift	sse	
<b>Conversion</b>											
CVTTPS2PD	r,r/m	1	0	4	1	4	1	mmx	shift	sse2	a
CVTPD2PS	r,r/m	2	0	10	1	2	1	fp-mmx	sse2	a	
CVTSD2SS	r,r/m	3	0	14	1	6	1	mmx	shift	sse2	a
CVTSS2SD	r,r/m	2	0	8	1	6	1	mmx	shift	sse2	a
CVTDQ2PS	r,r/m	1	0	5	1	2	1	fp		sse2	a
CVTDQ2PD	r,r/m	3	0	10	1	4	1	mmx-fp	sse2	a	
CVT(T)PS2DQ	r,r/m	1	0	5	1	2	1	fp		sse2	a
CVT(T)PD2DQ	r,r/m	2	0	11	1	2	1	fp-mmx	sse2	a	
CVTPI2PS	xmm,mm	4	0	12	1	6	1	mmx		sse	a

Prescott

CVTPI2PD	xmm,mm	4	0	12	1	5	1	fp-mmx	sse2	a	
CVT(T)PS2PI	mm,xmm	3	0	8	0	2	0,1	fp-mmx	sse	a	
CVT(T)PD2PI	mm,xmm	4	0	12	1	3	0,1	fp-mmx	sse2	a	
CVTSI2SS	xmm,r32	3	0	20	1	4	1	fp-mmx	sse	a	
CVTSI2SD	xmm,r32	4	0	20	1	5	1	fp-mmx	sse2	a	
CVT(T)SD2SI	r32,xmm	2	0	12	1	4	1	fp		sse2	a
CVT(T)SS2SI	r32,xmm	2	0	17	1	4	1	fp		sse	a
<b>Arithmetic</b>											
ADDPS/D ADDSS/D	r,r/m	1	0	5	1	2	1	fp	add	sse	a
SUBPS/D SUBSS/D	r,r/m	1	0	5	1	2	1	fp	add	sse	a
ADDSUBPS/D	r,r/m	1	0	5	1	2	1	fp	add	sse3	a
HADDPS/D HSUBPS/D	r,r/m	3	0	13	1	5-6	1	fp	add	sse3	a
MULPS/D MULSS/D	r,r/m	1	0	7	1	2	1	fp	mul	sse	a
DIVSS	r,r/m	1	0	32	1	23	1	fp	div	sse	a,h
DIVPS	r,r/m	1	0	41	1	41	1	fp	div	sse	a,h
DIVSD	r,r/m	1	0	40	1	40	1	fp	div	sse2	a,h
DIVPD	r,r/m	1	0	71	1	71	1	fp	div	sse2	a,h
RCPSS/RCPSS	r,r/m	2	0	6	1	4	1	mmx		sse	a
MAXPS/D MAXSS/DMINPS/D MINSS/D	r,r/m	1	0	5	1	2	1	fp	add	sse	a
CMPccPS/D CMPccSS/D	r,r/m	1	0	5	1	2	1	fp	add	sse	a
COMISS/D UCOMISS/D	r,r/m	2	0	6	1	3	1	fp	add	sse	a
<b>Logic</b>											
ANDPS/D ANDNPS/D ORPS/D XORPS/D	r,r/m	1	0	2	1	2	1	mmx	alu	sse	a
<b>Math</b>											
SQRTSS	r,r/m	1	0	32	1	32	1	fp	div	sse	a,h
SQRTPS	r,r/m	1	0	41	1	41	1	fp	div	sse	a,h
SQRTSD	r,r/m	1	0	40	1	40	1	fp	div	sse2	a,h
SQRTPD	r,r/m	1	0	71	1	71	1	fp	div	sse2	a,h
RSQRTSS	r,r/m	2	0	5	1	3	1	mmx		sse	a
RSQRTPS	r,r/m	2	0	6	1	4	1	mmx		sse	a
<b>Other</b>											
LDMXCSR	m	2	11			13	1			sse	
STMXCSR	m	3	0			3	1			sse	

**Notes:**

- a) Add 1  $\mu$ op if source is a memory operand.
- h) Throughput of FP-MUL unit is reduced during the use of the FP-DIV unit.
- k) It may be advantageous to replace this instruction by two 64-bit moves or LDDQU.

**Intel Atom****List of instruction timings and  $\mu$ ops breakdown***Explanation of column headings:*

<b>Instruction:</b>	Instruction name. cc means any condition code. For example, Jcc can be JB, JNE, etc.
<b>Operands:</b>	i = immediate data, r = register, mm = 64 bit mmx register, xmm = 128 bit xmm register, (x)mm = mmx or xmm register, sr = segment register, m = memory, m32 = 32-bit memory operand, etc.
<b><math>\mu</math>ops:</b>	The number of $\mu$ ops from the decoder or ROM.
<b>Unit:</b>	Tells which execution unit is used. Instructions that use the same unit cannot execute simultaneously. ALU0 and ALU1 means integer unit 0 or 1, respectively. ALU0/1 means that either unit can be used. ALU0+1 means that both units are used. Mem means memory in/out unit. FP0 means floating point unit 0 (includes multiply, divide and other SIMD instructions). FP1 means floating point unit 1 (adder). MUL means multiplier, shared between FP and integer units. DIV means divider, shared between FP and integer units. np means not pairable: Cannot execute simultaneously with any other instruction.
<b>Latency:</b>	This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays very much, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay.
<b>Reciprocal throughput:</b>	The average number of clock cycles per instruction for a series of independent instructions of the same kind in the same thread.

**Integer instructions**

	<b>Operands</b>	<b><math>\mu</math>ops</b>	<b>Unit</b>	<b>Latency</b>	<b>Reciprocal throughput</b>	<b>Remarks</b>
<b>Move instructions</b>						
MOV	r,r	1	ALU0/1	1	1/2	
MOV	r,i	1	ALU0/1	1	1/2	
MOV	r,m	1	ALU0, Mem	1-3	1	All addr. modes
MOV	m,r	1	ALU0, Mem	1	1	All addr. modes
MOV	m,i	1	ALU0, Mem		1	
MOV	r,sr	1		1	1	
MOV	m,sr	2			5	
MOV	sr,r	7			21	
MOV	sr,m	8			26	
MOVNTI	m,r	1	ALU0, Mem		2.5	
MOVSX MOVZX MOVSD	r,r/m	1	ALU0	1	1	
CMOVcc	r,r	1	ALU0+1	2	2	

## Atom

CMOVcc	r,m	1			3	
XCHG	r,r	3		6	6	
XCHG	r,m	4		6	6	Implicit lock
XLAT		3		6	6	
PUSH	r	1	np	1	1	
PUSH	i	1	np		1	
PUSH	m	2			5	
PUSH	sr	3			6	
PUSHF(D/Q)		14			12	
PUSHA(D)		9			11	Not in x64 mode
POP	r	1	np	1	1	
POP	(E/R)SP	1	np	1	1	
POP	m	3			6	
POP	sr	7			31	
POPF(D/Q)		19			28	
POPA(D)		16			12	Not in x64 mode
LAHF		1	ALU0+1	2	2	
SAHF		1	ALU0/1	1	1/2	
SALC		2		7	5	Not in x64 mode 4 clock latency on input register
LEA	r,m	1	AGU1	1-4	1	
BSWAP	r	1	ALU0	1	1	
LDS LES LFS LGS LSS	m	10		30	30	
PREFETCHNTA	m	1	Mem		1	
PREFETCHT0/1/2	m	1	Mem		1	
LFENCE		1			1/2	
MFENCE		1			1	
SFENCE		1			1	
<b>Arithmetic instructions</b>						
ADD SUB	r,r/i	1	ALU0/1	1	1/2	
ADD SUB	r,m	1	ALU0/1, Mem		1	
ADD SUB	m,r/i	1		2	1	
ADC SBB	r,r/i	1		2	2	
ADC SBB	r,m	1		2	2	
ADC SBB	m,r/i	1		2	2	
CMP	r,r/i	1	ALU0/1	1	1/2	
CMP	m,r/i	1			1	
INC DEC NEG NOT	r	1	ALU0/1	1	1/2	
INC DEC NEG NOT	m	1		1		
AAA		13		16		Not in x64 mode
AAS		13		12		Not in x64 mode
DAA		20		20		Not in x64 mode
DAS		21		25		Not in x64 mode
AAD		4		7		Not in x64 mode
AAM		10		24		Not in x64 mode
MUL IMUL	r8	3	ALU0, Mul	7	7	
MUL IMUL	r16	4	ALU0, Mul	6	6	
MUL IMUL	r32	3	ALU0, Mul	6	6	
MUL IMUL	r64	8	ALU0, Mul	14	14	

## Atom

IMUL	r16,r16	2	ALU0, Mul	6	5	
IMUL	r32,r32	1	ALU0, Mul	5	2	
IMUL	r64,r64	6	ALU0, Mul	13	11	
IMUL	r16,r16,i	2	ALU0, Mul	5	5	
IMUL	r32,r32,i	1	ALU0, Mul	5	2	
IMUL	r64,r64,i	7	ALU0, Mul	14	14	
MUL IMUL	m8	3	ALU0, Mul	6		
MUL IMUL	m16	5	ALU0, Mul	7		
MUL IMUL	m32	4	ALU0, Mul	7		
MUL IMUL	m64	8	ALU0, Mul	14		
DIV	r/m8	9	ALU0, Div	22	22	
DIV	r/m16	12	ALU0, Div	33	33	
DIV	r/m32	12	ALU0, Div	49	49	
DIV	r/m 64	38	ALU0, Div	183	183	
IDIV	r/m8	26	ALU0, Div	38	38	
IDIV	r/m16	29	ALU0, Div	45	45	
IDIV	r/m32	29	ALU0, Div	61	61	
IDIV	r/m64	60	ALU0, Div	207	207	
CBW		2	ALU0	5		
CWDE		1	ALU0	1		
CDQE		1	ALU0	1		
CWD		2	ALU0	5		
CDQ		1	ALU0	1		
CQO		1	ALU0	1		
<b>Logic instructions</b>						
AND OR XOR	r,r/i	1	ALU0/1	1	1/2	
AND OR XOR	r,m	1	ALU0/1, Mem		1	
AND OR XOR	m,r/i	1	ALU0/1, Mem	1	1	
TEST	r,r/i	1	ALU0/1	1	1/2	
TEST	m,r/i	1	ALU0/1, Mem		1	
SHR SHL SAR	r,i/cl	1	ALU0	1	1	
SHR SHL SAR	m,i/cl	1	ALU0	1	1	
ROR ROL	r,i/cl	1	ALU0	1	1	
ROR ROL	m,i/cl	1	ALU0	1	1	
RCR	r,1	5	ALU0	7		
RCL	r,1	2	ALU0	1		
RCR	r/m,i/cl	12-17	ALU0	12-15		
RCL	r/m,i/cl	14-20	ALU0	14-18		
SHLD	r16,r16,i	10	ALU0	10		1-2 more if mem
SHLD	r32,r32,i	2	ALU0	5		1-2 more if mem
SHLD	r64,r64,i	10	ALU0	11		1-2 more if mem
SHLD	r16,r16,cl	9	ALU0	9		1-2 more if mem
SHLD	r32,r32,cl	2	ALU0	5		1-2 more if mem
SHLD	r64,r64,cl	9	ALU0	10		1-2 more if mem
SHRD	r16,r16,i	8	ALU0	8		1-2 more if mem
SHRD	r32,r32,i	2	ALU0	5		1-2 more if mem
SHRD	r64,r64,i	10	ALU0	9		1-2 more if mem
SHRD	r16,r16,cl	7	ALU0	8		1-2 more if mem
SHRD	r32,r32,cl	2	ALU0	5		1-2 more if mem

## Atom

SHRD	r64,r64,cl	9	ALU0	9		1-2 more if mem
BT	r,r/i	1	ALU1	1	1	
BT	m,r	9		10		
BT	m,i	2		5		
BTR BTS BTC	r,r/i	1	ALU1	1	1	
BTR BTS BTC	m,r	10	ALU1	11		
BTR BTS BTC	m,i	3	ALU1	6		
BSF BSR	r,r/m	10		16		
SETcc	r	1	ALU0+1	2	2	
SETcc	m	2			5	
CLC STC		1	ALU0/1		1/2	
CMC		1		2	2	
CLD		5			7	
STD		6			25	
<b>Control transfer instructions</b>						
JMP	short/near	1	ALU1		2	
JMP	far	29			66	Not in x64 mode
JMP	r	1			4	
JMP	m(near)	2			7	
JMP	m(far)	30			78	
Conditional jump	short/near	1	ALU1		2	
J(E/R)CXZ	short	3			7	
LOOP	short	8			8	
LOOP(N)E	short	8			8	
CALL	near	1			3	
CALL	far	37			65	Not in x64 mode
CALL	r	1			18	
CALL	m(near)	2			20	
CALL	m(far)	38			64	
RETN		1	np		6	
RETN	i	1	np		6	
RETF		36			80	
RETF	i	36			80	
BOUND	r,m	11			10	Not in x64 mode
INTO		4			6	Not in x64 mode
<b>String instructions</b>						
LODS		3		6		
REP LODS		5n+11		3n+50		
STOS		2		5		
REP STOS		3n+10		2n+4		
MOVS		4		6		
REP MOVS		4n+11		2n - 4n		fastest for high n
SCAS		3		6		
REP SCAS		5n+16		3n+60		
CMPS		5		7		
REP CMPS		6n+16		4n+40		
Other						

## Atom

NOP (90)		1	ALU0/1		1/2
Long NOP (0F 1F)		1	ALU0/1		1/2
PAUSE		5		24	
ENTER	a,0	14		23	
ENTER	a,b	20+6b			
LEAVE		4			6
CPUID		40-80		100-170	
RDTSC		16		29	
RDPMSR		24		48	

## Floating point x87 instructions

	Operands	$\mu$ ops	Unit	Latency	Reciprocal	Remarks
<b>Move instructions</b>						
FLD	r	1		1	1	
FLD	m32/m64	1		3	1	
FLD	m80	4		9	10	
FBLD	m80	52		92	92	
FST(P)	r	1		1	1	
FST(P)	m32/m64	3		7	9	
FSTP	m80	8		12	13	
FBSTP	m80	189		221	221	
FXCH	r	1		1	1	
FILD	m	1		7	6	
FIST(P)	m	3		11	9	
FISTTP	m	3		11	9	SSE3
FLDZ		1			1	
FLD1		2			8	
FLDPI FLDL2E etc.		2			10	
FCMOVcc	r	3		9	9	
FNSTSW	AX	4			10	
FNSTSW	m16	4			10	
FLDCW	m16	2			8	
FNSTCW	m16	3			9	
FINCSTP FDECSTP		1		1	1	
FFREE(P)		1			1	
FNSAVE	m	166		321	321	
FRSTOR	m	83		177	177	
<b>Arithmetic instructions</b>						
FADD(P) FSUB(R)(P)	r/m	1		5	1	
FMUL(P)	r/m	1	Mul	5	2	
FDIV(R)(P)	r/m	1	Div	71	71	
FABS		1		1	1	
FCHS		1		1	1	
FCOM(P) FUCOM	r/m	1		1	1	
FCOMPP FUCOMPP		1		1	1	
FCOMI(P) FUCOMI(P)	r	5			10	
FIADD FISUB(R)	m	3			9	

## Atom

FIMUL	m	3	Mul		9
FIDIV(R)	m	3	Div		73
FICOM(P)	m	3			9
FTST		1		1	1
FXAM		1		1	1
FPREM		26		~110	
FPREM1		37		~130	
FRNDINT		19		48	
<b>Math</b>					
FSCALE		30		56	
FEXTRACT		15		24	
FSQRT		1	Div	71	
FSIN FCOS		9		~260	
FSINCOS		112		~260	
F2XM1		25		~100	
FYL2X FYL2XP1		63		~220	
FPTAN		100		~300	
FPATAN		91		~300	
<b>Other</b>					
FNOP		1			1
WAIT		2		5	5
FNCLEX		4			26
FNINIT		23		74	

## Integer MMX and XMM instructions

	Operands	$\mu$ ops	Unit	Latency	Reciprocal	Remarks
<b>Move instructions</b>						<b>through-</b>
MOVD	r32/64,(x)mm	1		4	2	
MOVD	m32/64,(x)mm	1	Mem	5	1	
MOVD	(x)mm,r32/64	1		3	1	
MOVD	(x)mm,m32/64	1	Mem	4	1	
MOVQ	(x)mm, (x)mm	1	FP0/1	1	1/2	
MOVQ	(x)mm,m64	1	Mem	4	1	
MOVQ	m64, (x)mm	1	Mem	5	1	
MOVDQA	xmm, xmm	1	FP0/1	1	1/2	
MOVDQA	xmm, m128	1	Mem	4	1	
MOVDQA	m128, xmm	1	Mem	5	1	
MOVDQU	m128, xmm	3	Mem	6	6	
MOVDQU	xmm, m128	4	Mem	6	6	
LDDQU	xmm, m128	4	Mem	6	6	
MOVDQ2Q	mm, xmm	1		1	1	
MOVQ2DQ	xmm,mm	1		1	1	
MOVNTQ	m64,mm	1	Mem	~400	1	
MOVNTDQ	m128,xmm	1	Mem	~450	3	



## Atom

PACKSSWB/DW					
PACKUSWB	(x)mm, (x)mm	1	FP0	1	1
PUNPCKH/LBW/WD/DQ	(x)mm, (x)mm	1	FP0	1	1
PUNPCKH/LQDQ	(x)mm, (x)mm	1	FP0	1	1
PSHUFB	mm,mm	1	FP0	1	1
PSHUFB	xmm,xmm	4		6	6
PSHUFW	mm,mm,i	1	FP0	1	1
PSHUFL/HW	xmm,xmm,i	1	FP0	1	1
PSHUFD	xmm,xmm,i	1	FP0	1	1
PALIGNR	xmm, xmm,i	1	FP0	1	1
MASKMOVQ	mm,mm	1	Mem		2
MASKMOVDQU	xmm,xmm	2	Mem		7
PMOVMSKB	r32,(x)mm	1		4	2
PINSRW	(x)mm,r32,i	1		3	1
PEXTRW	r32,(x)mm,i	2		5	5
<b>Arithmetic instructions</b>					
PADD/SUB(U)(S)B/W/D	(x)mm, (x)mm	1	FP0/1	1	1/2
PADDQ PSUBQ	(x)mm, (x)mm	2		5	5
PHADD(S)W PHSUB(S)W	(x)mm, (x)mm	7		8	8
PHADD PHSUBD	(x)mm, (x)mm	3		6	
PCMPEQ/GTB/W/D	(x)mm,(x)mm	1	FP0/1	1	1/2
PMULL/HW PMULHUW	mm,mm	1	FP0, Mul	4	1
PMULL/HW PMULHUW	xmm,xmm	1	FP0, Mul	5	2
PMULHRSW	mm,mm	1	FP0, Mul	4	1
PMULHRSW	xmm,xmm	1	FP0, Mul	5	2
PMULUDQ	mm,mm	1	FP0, Mul	4	1
PMULUDQ	xmm,xmm	1	FP0, Mul	5	2
PMADDWD	mm,mm	1	FP0, Mul	4	1
PMADDWD	xmm,xmm	1	FP0, Mul	5	2
PMADDUBSW	mm,mm	1	FP0, Mul	4	1
PMADDUBSW	xmm,xmm	1	FP0, Mul	5	2
PSADBW	mm,mm	1	FP0, Mul	4	1
PSADBW	xmm,xmm	1	FP0, Mul	5	2
PAVGB/W	(x)mm,(x)mm	1	FP0/1	1	1/2
PMIN/MAXUB	(x)mm,(x)mm	1	FP0/1	1	1/2
PMIN/MAXSW	(x)mm,(x)mm	1	FP0/1	1	1/2
PABSB PABSW PABSD	(x)mm,(x)mm	1	FP0/1	1	1/2
PSIGNB PSIGNW PSIGND	(x)mm,(x)mm	1	FP0/1	1	1/2
<b>Logic instructions</b>					
PAND(N) POR PXOR	(x)mm,(x)mm	1	FP0/1	1	1/2
PSLL/RL/RAW/D/Q	(x)mm,(x)mm	2	FP0	5	5
PSLL/RL/RAW/D/Q	(x)xmm,i	1	FP0	1	1
PSLL/RDQ	xmm,i	1	FP0	1	1
<b>Other</b>					
EMMS		9			9

## Floating point XMM instructions

	Operands	μops	Unit	Latency	Reciprocal	Remarks
<b>Move instructions</b>						
MOVAPS/D	xmm,xmm	1	FP0/1	1	1/2	
MOVAPS/D	xmm,m128	1	Mem	4	1	
MOVAPS/D	m128,xmm	1	Mem	5	1	
MOVUPS/D	xmm,m128	4	Mem	6	6	
MOVUPS/D	m128,xmm	3	Mem	6	6	
MOVSS/D	xmm,xmm	1	FP0/1	1	1/2	
MOVSS/D	xmm,m32/64	1	Mem	4	1	
MOVSS/D	m32/64,xmm	1	Mem	5	1	
MOVHPS/D MOVLPS/D	xmm,m64	1	Mem	5	1	
MOVHPS/D	m64,xmm	1	Mem	4	1	
MOVLPS/D	m64,xmm	1	Mem	4	1	
MOVLHPS MOVHLPS	xmm,xmm	1	FP0	1	1	
MOVMSKPS/D	r32,xmm	1		4	2	
MOVNTPS/D	m128,xmm	1	Mem	~500	3	
SHUFPS	xmm,xmm,i	1	FP0	1	1	
SHUFPD	xmm,xmm,i	1	FP0	1	1	
MOVDDUP	xmm,xmm	1	FP0	1	1	
MOVSH/LDUP	xmm,xmm	1	FP0	1	1	
UNPCKH/LPS	xmm,xmm	1	FP0	1	1	
UNPCKH/LPD	xmm,xmm	1	FP0	1		
<b>Conversion</b>						
CVTPD2PS	xmm,xmm	4		11	11	
CVTSD2SS	xmm,xmm	3		10	10	
CVTPS2PD	xmm,xmm	4		7	6	
CVTSS2SD	xmm,xmm	3		6	6	
CVTDQ2PS	xmm,xmm	3		6	6	
CVT(T) PS2DQ	xmm,xmm	3		6	6	
CVTDQ2PD	xmm,xmm	3		7	6	
CVT(T)PD2DQ	xmm,xmm	3		6	6	
CVTPI2PS	xmm,mm	1		6	5	
CVT(T)PS2PI	mm,xmm	1		4	1	
CVTPI2PD	xmm,mm	3		7	6	
CVT(T) PD2PI	mm,xmm	4		7	7	
CVTSI2SS	xmm,r32	3		7	6	
CVT(T)SS2SI	r32,xmm	3		10	8	
CVTSI2SD	xmm,r32	3		8	6	
CVT(T)SD2SI	r32,xmm	3		10	8	
<b>Arithmetic</b>						
ADDSS SUBSS	xmm,xmm	1	FP1	5	1	
ADDSD SUBSD	xmm,xmm	1	FP1	5	1	
ADDPSS SUBPSS	xmm,xmm	1	FP1	5	1	
ADDPD SUBPD	xmm,xmm	3	FP1	6	6	
ADDSUBPS	xmm,xmm	1	FP1	5	1	
ADDSUBPD	xmm,xmm	3	FP1	6	6	

Atom

HADDPS HSUBPS	xmm,xmm	5	FP0+1	8	7
HADDPD HSUBPD	xmm,xmm	5	FP0+1	8	7
MULSS	xmm,xmm	1	FP0, Mul	4	1
MULSD	xmm,xmm	1	FP0, Mul	5	2
MULPS	xmm,xmm	1	FP0, Mul	5	2
MULPD	xmm,xmm	6	FP0, Mul	9	9
DIVSS	xmm,xmm	3	FP0, Div	31	31
DIVSD	xmm,xmm	3	FP0, Div	60	60
DIVPS	xmm,xmm	6	FP0, Div	64	64
DIVPD	xmm,xmm	6	FP0, Div	122	122
RCPSS	xmm,xmm	1		4	1
RCPPS	xmm,xmm	5		9	8
CMPccSS/D	xmm,xmm	1	FP0	5	1
CMPccPS/D	xmm,xmm	3	FP0	6	6
COMISS/D UCOMISS/D	xmm,xmm	4	FP0	9	9
MAXSS/D MINSS/D	xmm,xmm	1	FP0	5	1
MAXPS/D MINPS/D	xmm,xmm	3	FP0	6	6
<b>Math</b>					
SQRTSS	xmm,xmm	3	FP0, Div	31	31
SQRTPS	xmm,xmm	5	FP0, Div	63	63
SQRTSD	xmm,xmm	3	FP0, Div	60	60
SQRTPD	xmm,xmm	5	FP0, Div	121	121
RSQRTSS	xmm,xmm	1	FP0	4	1
RSQRTPS	xmm,xmm	5	FP0	9	8
<b>Logic</b>					
ANDPS/D	xmm,xmm	1	FP0/1	1	1/2
ANDNPS/D	xmm,xmm	1	FP0/1	1	1/2
ORPS/D	xmm,xmm	1	FP0/1	1	1/2
XORPS/D	xmm,xmm	1	FP0/1	1	1/2
<b>Other</b>					
LDMXCSR	m32	4		5	6
STMXCSR	m32	4		14	15
FXSAVE	m4096	121		142	144
FXRSTOR	m4096	116		149	150

## VIA Nano 2000 series

### List of instruction timings and $\mu$ op breakdown

#### Explanation of column headings:

**Operands:** i = immediate data, r = register, mm = 64 bit mmx register, xmm = 128 bit xmm register, (x)mm = mmx or xmm register, sr = segment register, m = memory, m32 = 32-bit memory operand, etc.

**$\mu$ ops:** The number of micro-operations from the decoder or ROM. Note that the VIA Nano 2000 processor has no reliable performance monitor counter for  $\mu$ ops. Therefore the number of  $\mu$ ops cannot be determined except in simple cases.

**Port:** Tells which execution port or unit is used. Instructions that use the same port cannot execute simultaneously.

**I1:** Integer add, Boolean, shift, etc.

**I2:** Integer add, Boolean, move, jump.

**I12:** Can use either I1 or I2, whichever is vacant first.

**MA:** Multiply, divide and square root on all operand types.

**MB:** Various Integer and floating point SIMD operations.

**MBfadd:** Floating point addition subunit under MB.

**SA:** Memory store address.

**ST:** Memory store.

**LD:** Memory load.

**Latency:** This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays very much, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay.

**Note:** There is an additional latency for moving data from one unit or subunit to another. A table of these latencies is given in manual 3: "The microarchitecture of Intel, AMD and VIA CPUs". These additional latencies are not included in the listings below where the source and destination operands are of the same type.

**Reciprocal throughput:** The average number of clock cycles per instruction for a series of independent instructions of the same kind in the same thread.

### Integer instructions

	Operands	$\mu$ ops	Port	Latency	Reciprocal throughput	Remarks
<b>Move instructions</b>						
MOV	r,r	1	I2	1	1	Latency 4 on pointer register
MOV	r,i	1	I2	1	1	
MOV	r,m	1	LD	2	1	
MOV	m,r	1	SA, ST	2	1.5	
MOV	m,i	1	SA, ST		1.5	
MOV	r,sr				1	
MOV	m,sr				2	
MOV	sr,r			20	20	
MOV	sr,m			20	20	
MOVNTI	m,r		SA, ST	2	1.5	

VIA Nano 2000

MOVSX MOVSD						
MOVZX	r,r	1	I2	1	1	
MOVSX MOVSD	r,m	2	LD, I2	3	1	
MOVZX	r,m	1	LD	2	1	
CMOVcc	r,r	2	I1, I2	2	1	
CMOVcc	r,m		LD, I1	5	2	
XCHG	r,r	3	I2	3	3	
XCHG	r,m			20	20	Implicit lock
XLAT	m			6		
PUSH	r		SA, ST		1-2	
PUSH	i		SA, ST		1-2	
PUSH	m		Ld, SA, ST		2	
PUSH	sr				17	
PUSHF(D/Q)				8	8	
PUSHA(D)					15	Not in x64 mode
POP	r		LD		1.25	
POP	(E/R)SP				4	
POP	m				5	
POP	sr				20	
POPF(D/Q)				9	9	
POPA(D)					12	Not in x64 mode
LAHF		1	I1	1	1	
SAHF		1	I1	1	1	
SALC				9	6	Not in x64 mode
LEA	r,m	1	SA	1	1	3 clock latency on input register
BSWAP	r	1	I2	1	1	
LDS LES LFS LGS LSS	m			30	30	
PREFETCHNTA	m		LD		1-2	
PREFETCH0/1/2	m		LD		1-2	
LFENCE					14	
MFENCE					14	
SFENCE					14	
<b>Arithmetic instructions</b>						
ADD SUB	r,r/i	1	I12	1	1/2	
ADD SUB	r,m	2	LD I12		1	
ADD SUB	m,r/i	3	LD I12 SA ST	5	2	
ADC SBB	r,r/i	1	I1	1	1	
ADC SBB	r,m	2	LD I1		1	
ADC SBB	m,r/i	3	LD I1 SA ST	5	2	
CMP	r,r/i	1	I12	1	1/2	
CMP	m,r/i	2	LD I12		1	
INC DEC NEG NOT	r	1	I12	1	1/2	
INC DEC NEG NOT	m	3	LD I12 SA ST	5		
AAA					37	Not in x64 mode
AAS					37	Not in x64 mode
DAA					22	Not in x64 mode
DAS					24	Not in x64 mode

VIA Nano 2000

AAD					23	Not in x64 mode
AAM					30	Not in x64 mode
MUL IMUL	r8		MA	7-9		Extra latency to other ports
MUL IMUL	r16		MA	7-9		do.
MUL IMUL	r32		MA	7-9		do.
MUL IMUL	r64		MA	8-10		do.
IMUL	r16,r16		MA	4-6	1	do.
IMUL	r32,r32		MA	4-6	1	do.
IMUL	r64,r64		MA	5-7	2	do.
IMUL	r16,r16,i		MA	4-6	1	do.
IMUL	r32,r32,i		MA	4-6	1	do.
IMUL	r64,r64,i		MA	5-7	2	do.
DIV	r8		MA	26	26	do.
DIV	r16		MA	27-35	27-35	do.
DIV	r32		MA	25-41	25-41	do.
DIV	r64		MA	148-183	148-183	do.
IDIV	r8		MA	26	26	do.
IDIV	r16		MA	27-35	27-35	do.
IDIV	r32		MA	23-39	23-39	do.
IDIV	r64		MA	187-222	187-222	do.
CBW CWDE CDQE		1	I1	1	1	
CWD CDQ CQO		1	I1	1	1	
<b>Logic instructions</b>						
AND OR XOR	r,r/i	1	I12	1	1/2	
AND OR XOR	r,m	2	LD I12		1	
AND OR XOR	m,r/i	3	LD I12 SA ST	5	2	
TEST	r,r/i	1	I12	1	1/2	
TEST	m,r/i	2	LD I12		1	
SHR SHL SAR	r,i/cl	1	I1	1	1	
ROR ROL	r,i/cl	1	I1	1	1	
RCR RCL	r,1	1	I1	1	1	
RCR RCL	r,i/cl		I1	28+3n	28+3n	
SHLD SHRD	r16,r16,i		I1	11	11	
SHLD SHRD	r32,r32,i		I1	7	7	
SHLD	r64,r64,i		I1	33	33	
SHRD	r64,r64,i		I1	43	43	
SHLD SHRD	r16,r16,cl		I1	11	11	
SHLD SHRD	r32,r32,cl		I1	7	7	
SHLD	r64,r64,cl		I1	33	33	
SHRD	r64,r64,cl		I1	43	43	
BT	r,r/i	1	I1	1	1	
BT	m,r		I1		8	
BT	m,i	2	I1		1	
BTR BTS BTC	r,r/i	2	I1	2	2	
BTR BTS BTC	m,r		I1	10	10	
BTR BTS BTC	m,i		I1	8	8	
BSF BSR	r,r		I1	3	2	
SETcc	r		I1	2	1	

SETcc	m				1	
CLC STC CMC			11	3	3	
CLD STD				3	3	
<b>Control transfer instructions</b>						
JMP	short/near	1	12	3	3	8 if >2 jumps in 16 bytes block
JMP	far			58		Not in x64 mode
JMP	r		12	3	3	8 if >2 jumps in 16 bytes block
JMP	m(near)			3	3	do.
JMP	m(far)			55		
Conditional jump	short/near			1-3-8	1-3-8	1 if not jumping. 3 if jumping. 8 if >2 jumps in 16 bytes block
J(E/R)CXZ	short			1-3-8	1-3-8	do.
LOOP	short			1-3-8	1-3-8	do.
LOOP(N)E	short			25	25	
CALL	near			3	3	8 if >2 jumps in 16 bytes block
CALL	far			72	72	Not in x64 mode
CALL	r			3	3	8 if >2 jumps in 16 bytes block
CALL	m(near)			4	3	do.
CALL	m(far)			72	72	
RETN				3	3	8 if >2 jumps in 16 bytes block
RETN	i			3	3	do.
RETF				39	39	
RETF	i			39	39	
BOUND	r,m				13	Not in x64 mode
INTO					7	Not in x64 mode
<b>String instructions</b>						
LODSB/W/D/Q					1	
REP LODSB/W/D/Q					3n+22	
STOSB/W/D/Q					1-2	
REP STOSB/W/D/Q					Small: 2n+2, Big: 6 bytes per clock	
MOVSB/W/D/Q					2	
REP MOVSB/W/D/Q					Small: 2n+45, Big: 6 bytes per clock	
SCASB/W/D/Q					1	
REP SCASB					2.2n	
REP SCASW/D/Q					Small: 2n+50	
CMPSB/W/D/Q					Big: 5 bytes per clock	

VIA Nano 2000

REP CMPSB/W/D/Q					2.4n+24	
<b>Other</b>						
NOP (90)		1	All		1	Blocks all ports
Long NOP (0F 1F)		1	I12		1/2	
PAUSE					25	
ENTER	a,0				23	
ENTER	a,b				52+5b	
LEAVE				4	4	
CPUID				53-173		
RDTSC					39	
RDPMC				40	40	

**Floating point x87 instructions**

	Operands	μops	Port and Unit	Latency	Reciprocal through-put	Remarks
<b>Move instructions</b>						
FLD	r	1	MB	1	1	
FLD	m32/m64	2	LD MB	4	1	
FLD	m80	2	LD MB	4	1	
FBLD	m80			54	54	
FST(P)	r	1	MB	1	1	
FST(P)	m32/m64	3	MB SA ST	5	1-2	
FSTP	m80	3	MB SA ST	5	1-2	
FBSTP	m80			125	125	
FXCH	r	1	I2	0	1	
FILD	m16			7		
FILD	m32			5		
FILD	m64			5		
FIST(T)(P)	m16			6		
FIST(T)(P)	m32			5		
FIST(T)(P)	m64			5		
FLDZ FLD1		1	MB		1	
FLDPI FLDL2E etc.					10	
FCMOVcc	r			2	2	
FNSTSW	AX				5	
FNSTSW	m16				3	
FLDCW	m16			13	13	
FNSTCW	m16				2	
FINCSTP FDECSTP		1	I2	0	1	
FFREE(P)		1	MB		1	
FNSAVE	m			321	321	
FRSTOR	m			195	195	
<b>Arithmetic instructions</b>						
FADD(P) FSUB(R)(P)	r/m	1	MB	2	1	Lower precision: Lat: 4, Thr: 2
FMUL(P)	r/m	1	MA	4	2	
FDIV(R)(P)	r/m		MA	15-42	15-42	



VIA Nano 2000

FABS		1	MB	1	1
FCHS		1	MB	1	1
FCOM(P) FUCOM	r/m	1	MB		1
FCOMPP FUCOMPP		1	MB		1
FCOMI(P) FUCOMI(P)	r	1	MB		1
FIADD FISUB(R)	m		MB		2
FIMUL	m				4
FIDIV(R)	m				42
FICOM(P)	m	1			2
FTST		1	MB		1
FXAM					41
FPREM				151-171	
FPREM1				106-155	
FRNDINT				29	
<b>Math</b>					
FSCALE				39	
FEXTRACT				36-57	
FSQRT				73	
FSIN FCOS				51-159	
FSINCOS				270-360	
F2XM1				50-200	
FYL2X				~60	
FYL2XP1				~170	
FPTAN				300-370	
FPATAN				~170	
<b>Other</b>					
FNOP		1	MB		1
WAIT		1	I12	0	1/2
FNCLEX					57
FNINIT					85

**Integer MMX and XMM instructions**

	Operands	µops	Port and Unit	Latency	Reciprocal throughput	Remarks
<b>Move instructions</b>						
MOVD	r32/64,(x)mm	1		3	1	
MOVD	m32/64,(x)mm	1	SA ST	2-3	1-2	
MOVD	(x)mm,r32/64			4	1	
MOVD	(x)mm,m32/64	1	LD	2-3	1	
MOVQ	(x)mm, (x)mm	1	MB	1	1	
MOVQ	(x)mm,m64	1	LD	2-3	1	
MOVQ	m64, (x)mm	1	SA ST	2-3	1-2	
MOVDQA	xmm, xmm	1	MB	1	1	
MOVDQA	xmm, m128	1	LD	2-3	1	
MOVDQA	m128, xmm	1	SA ST	2-3	1-2	
MOVDQU	m128, xmm	1	SA ST	2-3	1-2	
MOVDQU	xmm, m128	1	LD	2-3	1	

VIA Nano 2000

LDDQU	xmm, m128	1	LD	2-3	1
MOVDQ2Q	mm, xmm	1	MB	1	1
MOVQ2DQ	xmm,mm	1	MB	1	1
MOVNTQ	m64,mm	3		~300	2
MOVNTDQ	m128,xmm	3		~300	2
PACKSSWB/DW					
PACKUSWB	(x)mm, (x)mm	1	MB	1	1
PUNPCKH/LBW/WD/DQ	(x)mm, (x)mm	1	MB	1	1
PUNPCKH/LQDQ	(x)mm, (x)mm	1	MB	1	1
PSHUFB	(x)mm,(x)mm	1	MB	1	1
PSHUFW	mm,mm,i	1	MB	1	1
PSHUFL/HW	xmm,xmm,i	1	MB	1	1
PSHUFD	xmm,xmm,i	1	MB	1	1
PALIGNR	xmm, xmm,i	1	MB	1	1
MASKMOVQ	mm,mm				1-3
MASKMOVDQU	xmm,xmm				1-3
PMOVMSKB	r32,(x)mm			3	1
PEXTRW	r32 ,(x)mm,i			3	1
PINSRW	(x)mm,r32,i			9	9
<b>Arithmetic instructions</b>					
PADD/SUB(U)(S)B/W/D	(x)mm, (x)mm	1	MB	1	1
PADDQ PSUBQ	(x)mm, (x)mm	1	MB	1	1
PHADD(S)W					
PHSUB(S)W	(x)mm, (x)mm	3	MB	3	3
PHADDD PHSUBD	(x)mm, (x)mm	3	MB	3	3
PCMPEQ/GTB/W/D	(x)mm,(x)mm	1	MB	1	1
PMULL/HW PMULHUW	(x)mm,(x)mm	1	MA	3	1
PMULHSW	(x)mm,(x)mm	1	MA	3	1
PMULUDQ	(x)mm,(x)mm	1	MA	3	1
PMADDWD	(x)mm,(x)mm			4	2
PMADDUBSW	(x)mm,(x)mm			10	8
PSADBW	(x)mm,(x)mm		MB	2	1
PAVGB/W	(x)mm,(x)mm	1	MB	1	1
PMIN/MAXUB	(x)mm,(x)mm	1	MB	1	1
PMIN/MAXSW	(x)mm,(x)mm	1	MB	1	1
PABSB PABSW PABSD	(x)mm,(x)mm	1	MB	1	1
PSIGNB PSIGNW					
PSIGND	(x)mm,(x)mm	1	MB	1	1
<b>Logic instructions</b>					
PAND(N) POR PXOR	(x)mm,(x)mm	1	MB	1	1
PSLL/RL/RAW/D/Q	(x)mm,(x)mm	1	MB	1	1
PSLL/RL/RAW/D/Q	(x)xmm,i	1	MB	1	1
PSLL/RLDQ	xmm,i	1	MB	1	1
<b>Other</b>					
EMMS		1	MB		1

**Floating point XMM instructions**

	Operands	μops	Port and Unit	Latency	Reciprocal throughput	Remarks
<b>Move instructions</b>						
MOVAPS/D	xmm,xmm	1	MB	1	1	
MOVAPS/D	xmm,m128	1	LD	2-3	1	
MOVAPS/D	m128,xmm	1	SA ST	2-3	1-2	
MOVUPS/D	xmm,m128	1	LD	2-3	1	
MOVUPS/D	m128,xmm	1	SA ST	2-3	1-2	
MOVSS/D	xmm,xmm	1	MB	1	1	
MOVSS/D	xmm,m32/64	1	LD	2-3	1	
MOVSS/D	m32/64,xmm	1	SA ST	2-3	1-2	
MOVHPS/D	xmm,m64			6	1	
MOVLPS/D	xmm,m64			6	1	
MOVHPS/D	m64,xmm			6	1-2	
MOVLPS/D	m64,xmm			2	1-2	
MOVLHPS MOVHLPS	xmm,xmm	1	MB	1	1	
MOVMSKPS/D	r32,xmm			3	1	
MOVNTPS/D	m128,xmm			~300	2.5	
SHUFPS	xmm,xmm,i	1	MB	1	1	
SHUFPD	xmm,xmm,i	1	MB	1	1	
MOVDDUP	xmm,xmm	1	MB	1	1	
MOVSH/LDUP	xmm,xmm	1	MB	1	1	
UNPCKH/LPS	xmm,xmm	1	MB	1	1	
UNPCKH/LPD	xmm,xmm	1	MB	1	1	
<b>Conversion</b>						
CVTPD2PS	xmm,xmm			3-4		
CVTSD2SS	xmm,xmm			15		
CVTPS2PD	xmm,xmm			3-4		
CVTSS2SD	xmm,xmm			15		
CVTDQ2PS	xmm,xmm			3		
CVT(T) PS2DQ	xmm,xmm			2		
CVTDQ2PD	xmm,xmm			4		
CVT(T)PD2DQ	xmm,xmm			3		
CVTPI2PS	xmm,mm			4		
CVT(T)PS2PI	mm,xmm			3		
CVTPI2PD	xmm,mm			4		
CVT(T) PD2PI	mm,xmm			3		
CVTSI2SS	xmm,r32			5		
CVT(T)SS2SI	r32,xmm			4		
CVTSI2SD	xmm,r32			5		
CVT(T)SD2SI	r32,xmm			4		
<b>Arithmetic</b>						
ADDSS SUBSS	xmm,xmm	1	MBfadd	2-3	1	
ADDSD SUBSD	xmm,xmm	1	MBfadd	2-3	1	
ADDPS SUBPS	xmm,xmm	1	MBfadd	2-3	1	

VIA Nano 2000

ADDPD SUBPD	xmm,xmm	1	MBfadd	2-3	1
ADDSUBPS	xmm,xmm	1	MBfadd	2-3	1
ADDSUBPD	xmm,xmm	1	MBfadd	2-3	1
HADDPS HSUBPS	xmm,xmm		MBfadd	5	3
HADDPD HSUBPD	xmm,xmm		MBfadd	5	3
MULSS	xmm,xmm	1	MA	3	1
MULSD	xmm,xmm	1	MA	4	2
MULPS	xmm,xmm		MA	3	1
MULPD	xmm,xmm		MA	4	2
DIVSS	xmm,xmm		MA	15-22	15-22
DIVSD	xmm,xmm		MA	15-36	15-36
DIVPS	xmm,xmm		MA	42-82	42-82
DIVPD	xmm,xmm		MA	24-70	24-70
RCPSS	xmm,xmm			5	5
RCPPS	xmm,xmm			14	11
CMPccSS/D	xmm,xmm	1	MBfadd	2	1
CMPccPS/D	xmm,xmm	1	MBfadd	2	1
COMISS/D UCOMISS/D	xmm,xmm			3	1
MAXSS/D MINSS/D	xmm,xmm	1	MBfadd	2	1
MAXPS/D MINPS/D	xmm,xmm	1	MBfadd	2	1
<b>Math</b>					
SQRTSS	xmm,xmm		MA	33	33
SQRTPS	xmm,xmm		MA	126	126
SQRTSD	xmm,xmm		MA	62	62
SQRTPD	xmm,xmm		MA	122	122
RSQRTSS	xmm,xmm			5	5
RSQRTPS	xmm,xmm			14	11
<b>Logic</b>					
ANDPS/D	xmm,xmm	1	MB	1	1
ANDNPS/D	xmm,xmm	1	MB	1	1
ORPS/D	xmm,xmm	1	MB	1	1
XORPS/D	xmm,xmm	1	MB	1	1
<b>Other</b>					
LDMXCSR	m32			45	29
STMXCSR	m32			13	13
FXSAVE	m4096			208	208
FXRSTOR	m4096			232	232

**VIA-specific instructions**

Instruction	Conditions	Clock cycles, approximately
XSTORE	Data available	160-400 clock giving 8 bytes
XSTORE	No data available	50-80 clock giving 0 bytes
REP XSTORE	Quality factor = 0	4800 clock per 8 bytes
REP XSTORE	Quality factor > 0	19200 clock per 8 bytes
REP XCRYPTECB	128 bits key	44 clock per 16 bytes

VIA Nano 2000

REP XCRYPTECB	192 bits key	46 clock per 16 bytes
REP XCRYPTECB	256 bits key	48 clock per 16 bytes
REP XCRYPTCBC	128 bits key	54 clock per 16 bytes
REP XCRYPTCBC	192 bits key	59 clock per 16 bytes
REP XCRYPTCBC	256 bits key	63 clock per 16 bytes
REP XCRYPTCTR	128 bits key	43 clock per 16 bytes
REP XCRYPTCTR	192 bits key	46 clock per 16 bytes
REP XCRYPTCTR	256 bits key	48 clock per 16 bytes
REP XCRYPTCFB	128 bits key	54 clock per 16 bytes
REP XCRYPTCFB	192 bits key	59 clock per 16 bytes
REP XCRYPTCFB	256 bits key	63 clock per 16 bytes
REP XCRYPTOFB	128 bits key	54 clock per 16 bytes
REP XCRYPTOFB	192 bits key	59 clock per 16 bytes
REP XCRYPTOFB	256 bits key	63 clock per 16 bytes
REP XSHA1		3 clock per byte
REP XSHA256		4 clock per byte

## VIA Nano 3000 series

### List of instruction timings and $\mu$ op breakdown

*Explanation of column headings:*

**Operands:** i = immediate data, r = register, mm = 64 bit mmx register, xmm = 128 bit xmm register, (x)mm = mmx or xmm register, sr = segment register, m = memory, m32 = 32-bit memory operand, etc.

**$\mu$ ops:** The number of micro-operations from the decoder or ROM. Note that the VIA Nano 3000 processor has no reliable performance monitor counter for  $\mu$ ops. Therefore the number of  $\mu$ ops cannot be determined except in simple cases.

**Port:** Tells which execution port or unit is used. Instructions that use the same port cannot execute simultaneously.

**I1:** Integer add, Boolean, shift, etc.

**I2:** Integer add, Boolean, move, jump.

**I12:** Can use either I1 or I2, whichever is vacant first.

**MA:** Multiply, divide and square root on all operand types.

**MB:** Various Integer and floating point SIMD operations.

**MBfadd:** Floating point addition subunit under MB.

**SA:** Memory store address.

**ST:** Memory store.

**LD:** Memory load.

**Latency:** This is the delay that the instruction generates in a dependency chain. The numbers are minimum values. Cache misses, misalignment, and exceptions may increase the clock counts considerably. Floating point operands are presumed to be normal numbers. Denormal numbers, NAN's and infinity increase the delays very much, except in XMM move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay.

**Note:** There is an additional latency for moving data from one unit or subunit to another. A table of these latencies is given in manual 3: "The microarchitecture of Intel, AMD and VIA CPUs". These additional latencies are not included in the listings below where the source and destination operands are of the same type.

**Reciprocal throughput:** The average number of clock cycles per instruction for a series of independent instructions of the same kind in the same thread.

### Integer instructions

	Operands	$\mu$ ops	Port	Latency	Reciprocal throughput	Remarks
<b>Move instructions</b>						
MOV	r,r	1	I2	1	1	Latency 4 on pointer register
MOV	r,i	1	I12	1	1/2	
MOV	r,m	1	LD	2	1	
MOV	m,r	1	SA, ST	2	1.5	
MOV	m,i	1	SA, ST		1.5	
MOV	r,sr		I12		1/2	
MOV	m,sr				1.5	
MOV	sr,r			20	20	
MOV	sr,m			20	20	
MOVNTI	m,r		SA, ST	2	1.5	

Nano 3000

MOVSX MOVZX	r,r	1	I12	1	1/2	
MOVSXD	r64,r32	1		1	1	
MOVSX MOVSXD	r,m	2	LD, I12	3	1	
MOVZX	r,m	1	LD	2	1	
CMOVcc	r,r	1	I12	1	1/2	
CMOVcc	r,m		LD, I12	5	1	
XCHG	r,r	3	I12	3	1.5	
XCHG	r,m			18	18	Implicit lock
XLAT	m	3	LD, I1	6	2	
PUSH	r	1	SA, ST		1-2	
PUSH	i	1	SA, ST		1-2	
PUSH	m		LD, SA, ST		2	
PUSH	sr				6	
PUSHF(D/Q)		3		2	2	
PUSHA(D)		9			15	Not in x64 mode
POP	r	2	LD		1.25	
POP	(E/R)SP				4	
POP	m	3			2	
POP	sr				11	
POPF(D/Q)		3			1	
POPA(D)		16			12	Not in x64 mode
LAHF		1	I1	1	1	
SAHF		1	I1	1	1	
SALC		2		10	6	Not in x64 mode
LEA	r,m	1	SA	1	1	Extra latency to other ports
BSWAP	r	1	I2	1	1	
LDS LES LFS LGS LSS	m	12		28	28	
PREFETCHNTA	m	1	LD		1	
PREFETCHT0/1/2	m	1	LD		1	
LFENCE MFENCE SFENCE					15	
<b>Arithmetic instructions</b>						
ADD SUB	r,r/i	1	I12	1	1/2	
ADD SUB	r,m	2	LD I12		1	
ADD SUB	m,r/i	3	LD I12 SA ST	5	2	
ADC SBB	r,r/i	1	I1	1	1	
ADC SBB	r,m	2	LD I1		1	
ADC SBB	m,r/i	3	LD I1 SA ST	5	2	
CMP	r,r/i	1	I12	1	1/2	
CMP	m,r/i	2	LD I12		1	
INC DEC NEG NOT	r	1	I12	1	1/2	
INC DEC NEG NOT	m	3	LD I12 SA ST	5		
AAA		12			37	Not in x64 mode
AAS		12			22	Not in x64 mode
DAA		14			22	Not in x64 mode
DAS		14			24	Not in x64 mode
AAD		7			24	Not in x64 mode

Nano 3000

AAM		13			31	Not in x64 mode
MUL IMUL	r8	1	I2	2		
MUL IMUL	r16	3	I2	3		
MUL IMUL	r32	3	I2	3		
MUL IMUL	r64	3	MA	8	8	Extra latency to other ports
IMUL	r16,r16	1	I2	2	1	
IMUL	r32,r32	1	I2	2	1	
IMUL	r64,r64	1	MA	5	2	Extra latency to other ports
IMUL	r16,r16,i	1	I2	2	1	
IMUL	r32,r32,i	1	I2	2	1	
IMUL	r64,r64,i	1	MA	5	2	Extra latency to other ports
DIV	r8		MA	22-24	22-24	
DIV	r16		MA	24-28	24-28	
DIV	r32		MA	22-30	22-30	
DIV	r64		MA	145-162	145-162	
IDIV	r8		MA	21-24	21-24	
IDIV	r16		MA	24-28	24-28	
IDIV	r32		MA	18-26	18-26	
IDIV	r64		MA	182-200	182-200	
CBW CWDE CDQE		1	I2	1	1	
CWD CDQ CQO		1	I2	1	1	
<b>Logic instructions</b>						
AND OR XOR	r,r/i	1	I12	1	1/2	
AND OR XOR	r,m	2	LD I12		1	
AND OR XOR	m,r/i	3	LD I12 SA ST	5	2	
TEST	r,r/i	1	I12	1	1/2	
TEST	m,r/i	2	LD I12		1	
SHR SHL SAR	r,i/cl	1	I12	1	1/2	
ROR ROL	r,i/cl	1	I1	1	1	
RCR RCL	r,1	1	I1	1	1	
RCR RCL	r,i/cl	5+2n	I1	28+3n	28+3n	
SHLD SHRD	r16,r16,i/cl	2	I1	2	2	
SHLD SHRD	r32,r32,i/cl	2	I1	2	2	
SHLD	r64,r64,i/cl	16	I1	32	32	
SHRD	r64,r64,i/cl	23	I1	42	42	
BT	r,r/i	1	I1	1	1	
BT	m,r	6	I1		8	
BT	m,i	2	I1		1	
BTR BTS BTC	r,r/i	2	I1	2	2	
BTR BTS BTC	m,r	8	I1	10	10	
BTR BTS BTC	m,i	5	I1	8	8	
BSF BSR	r,r	2	I1	2	2	
SETcc	r8	1	I1	1	1	
SETcc	m	2			2	
CLC STC CMC		3	I1	3	3	
CLD STD		3	I1	3	3	



<b>Control transfer instructions</b>						
JMP	short/near	1	12	3	3	8 if >2 jumps in 16 bytes block Not in x64 mode
JMP	far	14			50	
JMP	r	2	12	3	3	8 if >2 jumps in 16 bytes block do.
JMP	m(near)	2		3	3	
JMP	m(far)	17			42	
Conditional jump	short/near	1	12	1-3-8	1-3-8	1 if not jumping. 3 if jumping. 8 if >2 jumps in 16 bytes block
J(E/R)CXZ	short	2		1-3-8	1-3-8	
LOOP	short	2		1-3-8	1-3-8	
LOOP(N)E	short	5		24	24	
CALL	near	2		3	3	8 if >2 jumps in 16 bytes block Not in x64 mode
CALL	far	17			58	
CALL	r	2		3	3	8 if >2 jumps in 16 bytes block do.
CALL	m(near)	3		4	3	
CALL	m(far)	19			54	
RETN		3		3	3	8 if >2 jumps in 16 bytes block do.
RETN	i	4		3	3	
RETF		20			49	Not in x64 mode
RETF	i	20			49	
BOUND	r,m	9			13	
INTO		3			7	Not in x64 mode
<b>String instructions</b>						
LODSB/W/D/Q		2			1	Small: n+40, Big: 6-7 bytes/clock
REP LODSB/W/D/Q		3n			3n+27	
STOSB/W/D/Q		1			1-2	
REP STOSB/W/D/Q						Small: 2n+20, Big: 6-7 bytes/clock
MOVSB/W/D/Q		3			2	
REP MOVSB/W/D/Q						Small: 2n+31, Big: 5 bytes/clock
SCASB/W/D/Q		3			1	
REP SCASB					2.4n	
REP SCASW/D/Q						Small: 2n+30, Big: 5 bytes/clock
CMPSB/W/D/Q		5			6	
REP CMPSB/W/D/Q					2.2n+30	

Nano 3000

<b>Other</b>						
NOP (90)		0-1	I12	0	1/2	Sometimes fused
long NOP (0F 1F)		0-1	I12	0	1/2	
PAUSE		2			6	
ENTER	a,0	10			21	
ENTER	a,b				52+5b	
LEAVE		3		2	2	
CPUID				55-146		
RDTSC					37	
RDPMC					40	

**Floating point x87 instructions**

	Operands	μops	Port	Latency	Reciprocal through-put	Remarks
<b>Move instructions</b>						
FLD	r	1	MB	1	1	
FLD	m32/m64	2	LD MB	4	1	
FLD	m80	2	LD MB	4	1	
FBLD	m80	36		54	54	
FST(P)	r	1	MB	1	1	
FST(P)	m32/m64	3	MB SA ST	5	1-2	
FSTP	m80	3	MB SA ST	5	1-2	
FBSTP	m80	80		125	125	
FXCH	r	1	I2	0	1	
FILD	m16	3		7		
FILD	m32	2		5		
FILD	m64	2		5		
FIST(T)(P)	m16	3		6		
FIST(T)(P)	m32	3		5		
FIST(T)(P)	m64	3		5		
FLDZ FLD1		1	MB		1	
FLDPI FLDL2E etc.		3			10	
FCMOVcc	r	1	MB	2	2	
FNSTSW	AX	1			1	
FNSTSW	m16	3			2	
FLDCW	m16	5			8	
FNSTCW	m16	3			2	
FINCSTP FDECSTP		1	I2	0	1	
FFREE(P)		1	MB		1	
FNSAVE	m	122		319	319	
FRSTOR	m	115		196	196	
<b>Arithmetic instructions</b>						
FADD(P) FSUB(R)(P)	r/m	1	MB	2	1	
FMUL(P)	r/m	1	MA	4	2	
FDIV(R)(P)	r/m		MA	14-23	14-23	
FABS		1	MB	1	1	
FCHS		1	MB	1	1	
FCOM(P) FUCOM	r/m	1	MB		1	

Nano 3000

FCOMPP FUCOMPP		1	MB		1		
FCOMI(P) FUCOMI(P)	r	1	MB	2	1		
FIADD FISUB(R)	m	3	MB		2		
FIMUL	m	3			4		
FIDIV(R)	m	3			16		
FICOM(P)	m	3			2		
FTST		1	MB	2	1		
FXAM		15		38	38		
FPREM				~130			
FPREM1				~130			
FRNDINT		11		27			
<b>Math</b>							
FSCALE		22		37			
FEXTRACT		13		57			
FSQRT				73		Less at lower precision	
FSIN FCOS				~150			
FSINCOS				270-360			
F2XM1				50-200			
FYL2X				~50			
FYL2XP1				~50			
FPTAN				300-370			
FPATAN				~180			
<b>Other</b>							
FNOP		1	MB		1		
WAIT		1	I12	0	1/2		
FNCLEX					59		
FNINIT					84		

**Integer MMX and XMM instructions**

	Operands	μops	Port	Latency	Reciprocal through-put	Remarks
<b>Move instructions</b>						
MOVD	r32/64,(x)mm	1	MB	3	1	
MOVD	m32/64,(x)mm	1	SA ST	2	1-2	
MOVD	(x)mm,r32/64	1	I2	4	1	
MOVD	(x)mm,m32/64	1	LD	2	1	
MOVQ	(x)mm, (x)mm	1	MB	1	1	
MOVQ	(x)mm,m64	1	LD	2	1	
MOVQ	m64, (x)mm	1	SA ST	2	1-2	
MOVDQA	xmm, xmm	1	MB	1	1	
MOVDQA	xmm, m128	1	LD	2	1	
MOVDQA	m128, xmm	1	SA ST	2	1-2	
MOVDQU	m128, xmm	1	SA ST	2	1-2	
MOVDQU	xmm, m128	1	LD	2	1	
LDDQU	xmm, m128	1	LD	2	1	
MOVDQ2Q	mm, xmm	1	MB	1	1	

Nano 3000

MOVQ2DQ	xmm,mm	1	MB	1	1
MOVNTQ	m64,mm	2		~360	2
MOVNTDQ	m128,xmm	2		~360	2
MOVNTDQA	xmm,m128	1		2	1
PACKSSWB/DW					
PACKUSWB	(x)mm, (x)mm	1	MB	1	1
PACKUSDW	xmm,xmm	1	MB	1	1
PUNPCKH/LBW/WD/DQ	(x)mm, (x)mm	1	MB	1	1
PUNPCKH/LQDQ	(x)mm, (x)mm	1	MB	1	1
PSHUFB	(x)mm,(x)mm	1	MB	1	1
PSHUFW	mm,mm,i	1	MB	1	1
PSHUFL/HW	xmm,xmm,i	1	MB	1	1
PSHUFD	xmm,xmm,i	1	MB	1	1
PBLENDVB	x,x,xmm0	1	MB	2	2
PBLENDW	xmm,xmm,i	1	MB	1	1
PALIGNR	xmm, xmm,i	1	MB	1	1
MASKMOVQ	mm,mm				1-2
MASKMOVDQU	xmm,xmm				1-2
PMOVMSKB	r32,(x)mm			3	1
PEXTRW	r32 ,(x)mm,i	1	MB	3	1
PEXTRB/D/Q	r32/64 ,xmm,i	1	MB	3	1
PINSRW	(x)mm,r32,i	2	MB	5	1
PINSRB/D/Q	xmm,r32/64,i	2	MB	5	1
PMOVSX/ZXBW/BD/ BQ/WD/WQ/DQ	xmm,xmm	1	MB	1	1
<b>Arithmetic instructions</b>					
PADD/SUB(U)(S)B/W/D	(x)mm, (x)mm	1	MB	1	1
PADDQ PSUBQ	(x)mm, (x)mm	1	MB	1	1
PHADD(S)W					
PHSUB(S)W	(x)mm, (x)mm	3	MB	3	3
PHADD PHSUBD	(x)mm, (x)mm	3	MB	3	3
PCMPEQ/GTB/W/D	(x)mm,(x)mm	1	MB	1	1
PCMPEQQ	xmm,xmm	1	MB	1	1
PMULL/HW PMULHUW	(x)mm,(x)mm	1	MA	3	1
PMULHRSW	(x)mm,(x)mm	1	MA	3	1
PMULLD	xmm,xmm	1	MA	3	1
PMULUDQ	(x)mm,(x)mm	1	MA	3	1
PMULDQ	xmm,xmm	1	MA	3	1
PMADDWD	(x)mm,(x)mm	1	MA	4	2
PMADDUBSW	(x)mm,(x)mm	7		10	8
PSADBW	(x)mm,(x)mm	1	MB	2	1
MPSADBW	xmm,xmm,i	1	MB	2	1
PAVGB/W	(x)mm,(x)mm	1	MB	1	1
PMIN/MAXSW	(x)mm,(x)mm	1	MB	1	1
PMIN/MAXUB	(x)mm,(x)mm	1	MB	1	1
PMIN/MAXSB/D	xmm,xmm	1	MB	1	1
PMIN/MAXUW/D	xmm,xmm	1	MB	1	1
PHMINPOSUW	xmm,xmm	1	MB	2	1

Nano 3000

PABSB PABSW PABSD	(x)mm,(x)mm	1	MB	1	1
PSIGNB PSIGNW PSIGND	(x)mm,(x)mm	1	MB	1	1
<b>Logic instructions</b>					
PAND(N) POR PXOR	(x)mm,(x)mm	1	MB	1	1
PTEST	xmm,xmm	1	MB	3	1
PSLL/RL/RAW/D/Q	(x)mm,(x)mm	1	MB	1	1
PSLL/RL/RAW/D/Q	(x)xmm,i	1	MB	1	1
PSLL/RLDQ	xmm,i	1	MB	1	1
<b>Other</b>					
EMMS		1	MB		1

**Floating point XMM instructions**

	Operands	μops	Port	Latency	Reciprocal throughput	Remarks
<b>Move instructions</b>						
MOVAPS/D	xmm,xmm	1	MB	1	1	
MOVAPS/D	xmm,m128	1	LD	2	1	
MOVAPS/D	m128,xmm	1	SA ST	2	1	
MOVUPS/D	xmm,m128	1	LD	2	1	
MOVUPS/D	m128,xmm	2	SA ST	2	1	
MOVSS/D	xmm,xmm	1	MB	1	1	
MOVSS/D	xmm,m32/64	1	LD	2-3	1	
MOVSS/D	m32/64,xmm	2	SA ST	2-3	1-2	
MOVHPS/D	xmm,m64	2		6	1	
MOVLPS/D	xmm,m64	2		6	1	
MOVHPS/D	m64,xmm	3		6	1-2	
MOVLPS/D	m64,xmm	1		2	1-2	
MOVLHPS MOVHLPS	xmm,xmm	1		1	1	
MOVMSKPS/D	r32,xmm			3	1	
MOVNTPS/D	m128,xmm	2		~360	1-2	
SHUFPS	xmm,xmm,i	1	MB	1	1	
SHUFPD	xmm,xmm,i	1	MB	1	1	
MOVDDUP	xmm,xmm	1	MB	1	1	
MOVSH/LDUP	xmm,xmm	1	MB	1	1	
UNPCKH/LPS	xmm,xmm	1	MB	1	1	
UNPCKH/LPD	xmm,xmm	1	MB	1	1	
<b>Conversion</b>						
CVTPD2PS	xmm,xmm	2		5	2	
CVTSD2SS	xmm,xmm	1		2		
CVTSS2PD	xmm,xmm	2		5	1	
CVTSS2SD	xmm,xmm	1		2		
CVTDQ2PS	xmm,xmm	1	MB	3	1	
CVT(T) PS2DQ	xmm,xmm	1		2	1	
CVTDQ2PD	xmm,xmm	2		5	1	

Nano 3000

CVT(T)PD2DQ	xmm,xmm			4	2
CVTPI2PS	xmm,mm	2		5	2
CVT(T)PS2PI	mm,xmm	1		4	1
CVTPI2PD	xmm,mm	2		4	1
CVT(T) PD2PI	mm,xmm	2		4	2
CVTSI2SS	xmm,r32	2		5	
CVT(T)SS2SI	r32,xmm	1		4	1
CVTSI2SD	xmm,r32	2		5	
CVT(T)SD2SI	r32,xmm	1		4	1
<b>Arithmetic</b>					
ADDSS SUBSS	xmm,xmm	1	MBfadd	2	1
ADDSD SUBSD	xmm,xmm	1	MBfadd	2	1
ADDPS SUBPS	xmm,xmm	1	MBfadd	2	1
ADDPD SUBPD	xmm,xmm	1	MBfadd	2	1
ADDSUBPS	xmm,xmm	1	MBfadd	2	1
ADDSUBPD	xmm,xmm	1	MBfadd	2	1
HADDPS HSUBPS	xmm,xmm	3	MBfadd	5	3
HADDPD HSUBPD	xmm,xmm	3	MBfadd	5	3
MULSS	xmm,xmm	1	MA	3	1
MULSD	xmm,xmm	1	MA	4	2
MULPS	xmm,xmm	1	MA	3	1
MULPD	xmm,xmm	1	MA	4	2
DIVSS	xmm,xmm	1	MA	13	13
DIVSD	xmm,xmm	1	MA	13-20	13-20
DIVPS	xmm,xmm	1	MA	24	24
DIVPD	xmm,xmm	1	MA	21-38	21-38
RCPSS	xmm,xmm	1	MA	5	5
RCPPS	xmm,xmm	3	MA	14	11
CMPccSS/D	xmm,xmm	1	MBfadd	2	1
CMPccPS/D	xmm,xmm	1	MBfadd	2	1
COMISS/D UCOMISS/D	xmm,xmm	1	MBfadd	3	1
MAXSS/D MINSS/D	xmm,xmm	1	MBfadd	2	1
MAXPS/D MINPS/D	xmm,xmm	1	MBfadd	2	1
<b>Math</b>					
SQRTSS	xmm,xmm	1	MA	33	33
SQRTPS	xmm,xmm	1	MA	64	64
SQRTSD	xmm,xmm	1	MA	62	62
SQRTPD	xmm,xmm	1	MA	122	122
RSQRTSS	xmm,xmm	1		5	5
RSQRTPS	xmm,xmm	3		14	11
<b>Logic</b>					
ANDPS/D	xmm,xmm	1	MB	1	1
ANDNPS/D	xmm,xmm	1	MB	1	1
ORPS/D	xmm,xmm	1	MB	1	1
XORPS/D	xmm,xmm	1	MB	1	1

Nano 3000

Other					
LDMXCSR	m32				31
STMXCSR	m32				13
FXSAVE	m4096				97
FXRSTOR	m4096				201

**VIA-specific instructions**

Instruction	Conditions	Clock cycles, approximately
XSTORE	Data available	160-400 clock giving 8 bytes
XSTORE	No data available	50-80 clock giving 0 bytes
REP XSTORE	Quality factor = 0	1300 clock per 8 bytes
REP XSTORE	Quality factor > 0	5455 clock per 8 bytes
REP XCRYPTECB	128 bits key	15 clock per 16 bytes
REP XCRYPTECB	192 bits key	17 clock per 16 bytes
REP XCRYPTECB	256 bits key	18 clock per 16 bytes
REP XCRYPTCBC	128 bits key	29 clock per 16 bytes
REP XCRYPTCBC	192 bits key	33 clock per 16 bytes
REP XCRYPTCBC	256 bits key	37 clock per 16 bytes
REP XCRYPTCTR	128 bits key	23 clock per 16 bytes
REP XCRYPTCTR	192 bits key	26 clock per 16 bytes
REP XCRYPTCTR	256 bits key	27 clock per 16 bytes
REP XCRYPTCFB	128 bits key	29 clock per 16 bytes
REP XCRYPTCFB	192 bits key	33 clock per 16 bytes
REP XCRYPTCFB	256 bits key	37 clock per 16 bytes
REP XCRYPTOFB	128 bits key	29 clock per 16 bytes
REP XCRYPTOFB	192 bits key	33 clock per 16 bytes
REP XCRYPTOFB	256 bits key	37 clock per 16 bytes
REP XSHA1		5 clock per byte
REP XSHA256		5 clock per byte