

# 4.

## Instruction tables

### Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel and AMD CPU's

By Agner Fog. Copenhagen University College of Engineering.  
Copyright © 1996 - 2008. Last updated 2008-08-24.

#### Contents

1	Introduction .....	3
1.1	Definition of terms .....	3
1.2	Microprocessor versions tested .....	5
2	List of instruction timings for P1 and PMMX .....	6
2.1	Integer instructions (P1 and PMMX) .....	6
2.2	Floating point instructions (P1 and PMMX) .....	8
2.3	MMX instructions (PMMX) .....	10
3	List of instruction timings and $\mu$ op breakdown for PPro, P2 and P3 .....	11
3.1	Integer instructions (PPro, P2 and P3) .....	11
3.2	Floating point x87 instructions (PPro, P2 and P3) .....	14
3.3	Integer MMX instructions (P2 and P3) .....	15
3.4	Floating point XMM instructions (P3) .....	16
4	List of instruction timings and $\mu$ op breakdown for PM .....	18
4.1	Integer instructions .....	18
4.2	Floating point x87 instructions .....	22
4.3	Integer MMX and XMM instructions .....	23
4.4	Floating point XMM instructions .....	26
5	List of instruction timings and $\mu$ op breakdown for Core 2 (65nm) .....	29
5.1	Integer instructions .....	30
5.2	Floating point x87 instructions .....	34
5.3	Integer MMX and XMM instructions .....	36
5.4	Floating point XMM instructions .....	39
6	List of instruction timings and $\mu$ op breakdown for Core 2 (45nm) .....	43
6.1	Integer instructions .....	44
6.2	Floating point x87 instructions .....	48
6.3	Integer MMX and XMM instructions .....	50
6.4	Floating point XMM instructions .....	54
7	List of instruction timings and $\mu$ op breakdown for P4 .....	58
7.1	Integer instructions .....	59
7.2	Floating point x87 instructions .....	63
7.3	Integer MMX and XMM instructions .....	64
7.4	Floating point XMM instructions .....	66
8	List of instruction timings and $\mu$ op breakdown for P4E .....	68
8.1	Integer instructions .....	69
8.2	Floating point x87 instructions .....	73
8.3	Integer MMX and XMM instructions .....	75
8.4	Floating point XMM instructions .....	77
9	Instruction timings and macro-operation breakdown for AMD K7 .....	79
9.1	Integer instructions .....	79
9.2	Floating point x87 instructions .....	83
9.3	Integer MMX instructions .....	85
9.4	Floating point XMM instructions .....	86
9.5	3DNow instructions .....	87
10	Instruction timings and macro-operation breakdown for AMD K8 .....	88
10.1	Integer instructions .....	88

10.2 Floating point x87 instructions.....	92
10.3 Integer MMX and XMM instructions .....	94
10.4 Floating point XMM instructions .....	96
10.5 3DNow instructions .....	97
11 Instruction timings and macro-operation breakdown for AMD K10 .....	99
11.1 Integer instructions.....	99
11.2 Floating point x87 instructions.....	103
11.3 Integer MMX and XMM instructions .....	105
11.4 Floating point XMM instructions .....	107
11.5 3DNow instructions .....	108
12 Instruction set compatibility table.....	111
12.1 Explanation of instruction sets .....	112
13 Comparison of the different microprocessors .....	116
14 Literature.....	117

# 1 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 and AMD CPU's: An optimization guide for assembly programmers and compiler makers.
4. Instruction tables: Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel and AMD CPU's.
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 in manual 5.

The present manual contains tables of instruction latencies, throughputs and micro-operation breakdown and other tables for Intel and AMD CPU's as an appendix to the preceding manuals.

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 on the P4, P4E and Core2 microarchitectures are listed explicitly in 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.

## 1.1 Definition of terms

### Microarchitecture abbreviations

The tables in this manual are organized around the different kernel microarchitectures, not the commercial names of the microprocessors. Certain brand names are covering more than one microarchitecture, and some microarchitectures are sold under several different brand names. See manual 3: "The microarchitecture of Intel and AMD CPU's" for details.

Microprocessor name	Abbreviation
Intel Pentium (without name suffix)	P1
Intel Pentium MMX	PMMX
Intel Pentium Pro	PPro
Intel Pentium II	P2
Intel Pentium III	P3
Intel Pentium 4 (NetBurst)	P4
Intel Pentium 4 with EM64T, Pentium D, etc.	P4E
Intel Pentium M, Core Solo, Core Duo	PM
Intel Core 2	Core2
AMD Athlon	AMD K7
AMD Athlon 64, Opteron, etc., 64-bit	AMD K8
AMD Family 10h, Phenom, third generation Opteron	AMD K10

### Operands

i = immediate constant, r = any general purpose 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.

### Latency

The latency of an instruction is the delay that the instruction generates in a dependence chain. The measurement unit is clock cycles. 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 except in move, shuffle and Boolean instructions. Floating point overflow, underflow, denormal or NAN results give a similar delay.

### 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 dependence chains. 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 0.25 for ADD means that the execution units can handle 4 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.

### μops

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.

### Execution unit

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  $\mu$ op goes to can be useful for two purposes. Firstly, two  $\mu$ ops cannot execute simultaneously if they need the same execution unit. And secondly, the P4 and P4E processors have a latency of an extra clock cycle when the result of an  $\mu$ op executing in one execution unit is needed as input for an  $\mu$ op in another execution unit.

### Execution port

The execution units are clustered around a few execution ports on most Intel processors. Each  $\mu$ op passes through an execution port to get to the right execution unit. An execution port can be a bottleneck because it can only handle one  $\mu$ op at a time. Two  $\mu$ ops cannot execute simultaneously if they need the same execution port, even if they are going to different execution units.

### Backwards compatibility

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 in chapter 0 page 98. 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.

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. It is necessary to test which microprocessor the code is running on, and possibly also which operating system, before using an instruction that is not available on all processors or all operating systems.

## **1.2 Microprocessor versions tested**

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

<b>Processor name</b>	<b>Family number (hex)</b>	<b>Model number (hex)</b>	<b>Comment</b>
Intel Pentium	5	2	
Intel Pentium MMX	5	4	Stepping 4
Intel Pentium II	6	6	
Intel Pentium III	6	7	
Intel Pentium 4	F	2	Stepping 4, rev. B0
Intel Pentium 4 EM64T	F	4	Xeon. Stepping 1
Intel Pentium M	6	D	Stepping 6, rev. B1
Intel Core Duo	6	E	Not fully tested
Intel Core 2 (65 nm)	6	F	T5500, Step. 6, rev. B2
Intel Core 2 (45 nm)	6	17	E8400, Step. 6
AMD K7 Athlon	6	6	Step. 2, rev. A5
AMD K8 Opteron	F	5	Stepping A
AMD K10 Opteron	10	2	2350, step. 1

## 2 List of instruction timings for P1 and PMMX

### 2.1 Integer instructions (P1 and PMMX)

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.

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)

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
REP MOVS		12+n g)	np

SCAS		4	np
REP(N)E SCAS		9+4*n g)	np
CMPS		5	np
REP(N)E CMPS		8+4*n g)	np
BSWAP		1 a)	np
CPUID		13-16 a)	np
RDTSC		6-13 a) j)	np

#### Notes:

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

## 2.2 Floating point instructions (P1 and PMMX)

#### 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
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
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	+	2	2
FSUB(R)(P)	r/m	3	+	2	2
FMUL(P)	r/m	3	+	2	2 n)
FDIV(R)(P)	r/m	19/33/39 p)	+	38 o)	2
FCHS FABS		1	+	0	0
FCOM(P)(P) FUCOM	r/m	1	+	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
EXTRACT		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](#).

## 2.3 MMX instructions (PMMX)

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 and AMD CPU's".

### 3 List of instruction timings and $\mu$ ops breakdown for PPro, P2 and P3

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 dependence 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.

#### 3.1 Integer instructions (PPro, P2 and P3)

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				
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,l	1		1					
RCR RCL	r8,i/CL	4		4					
RCR RCL	r16/32,i/CL	3		3					
RCR RCL	m,l	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					
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		1		1					
NOP (90)				1					0.5
Long NOP (0F-1F-mod00rm)				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:**

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

### 3.2 Floating point x87 instructions (PPro, P2 and P3)

Instruction	Operands	$\mu$ ops						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	$\frac{1}{3}$ 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							
FSCALE		56							
FXTRACT		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.

### 3.3 Integer MMX instructions (P2 and P3)

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

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.

### 3.4 Floating point XMM instructions (P3)

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
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	

## 4 List of instruction timings and $\mu$ ops breakdown for PM

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 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.

**p0:** Port 0: ALU, etc.

**p1:** Port 1: ALU, jumps

**p01:** Can go to either port 0 or port 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 dependence 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 in one thread.

### 4.1 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				
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		2	1		1					
LDS LES LFS LGS LSS	m	11			8	3				
PREFETCHNTA	m	1				1				1
PREFETCHT0/1/2	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
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,l	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,l	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
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
NOP (0F 1F mod000rm)		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 and AMD CPU's".
- 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.

## 4.2 Floating point x87 instructions

Instruction	Operands	jops fused domain	jops unfused domain						Latency	Reciprocal throughput
Move instructions			p0	p1	p01	p2	p3	p4		
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	0.33 f)
FILD	m	4	3			1			5	2
FIST(P)	m	4	2				1	1	5	2
FISTTP g)	m	4	2				1	1	5	2
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
Arithmetic instructions										
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			1			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	
FSQRT		1	1					9 h)	8
FSIN FCOS		80-100	80-100					80-110	
FSINCOS		90-110	90-110					100-130	
F2XM1		ca. 20	ca. 20					ca. 45	
FYL2X		ca. 40	ca. 40					ca. 60	
FYL2XP1		ca. 55	ca. 55					ca. 65	
FPTAN		ca. 100	ca. 100					ca. 140	
FPATAN		ca. 85	ca. 85					ca. 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.

### 4.3 Integer MMX and XMM instructions

Instruction	Operands	μops fused domain	μops unfused domain						Latency	Reciprocal throughput
			p0	p1	p01	p2	p3	p4		
Move instructions										
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
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
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.

#### 4.4 Floating point XMM instructions

Instruction	Operands	jops fused domain	jops unfused domain						Latency	Reciprocal throughput
Move instructions			p0	p1	p01	p2	p3	p4		
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
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
Conversion										
CVTTPS2PD	xmm,xmm	4	2	2					3	3
CVTTPS2PD	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
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
ADDPS/D SUBPS/D	xmm,xmm	2			2				3	2
ADDPS/D SUBPS/D	xmm,m128	4			2	2			3	2
ADDSUBPS/D g)	xmm,xmm	2			2				3	2
HADDPS 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
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.

## 5 List of instruction timings and $\mu$ ops breakdown for Core 2 (65nm)

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 dependence 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.

## 5.1 Integer instructions

Instruction	Operands	jops fused domain	jops unfused domain							Latency	Reciprocal throughput
Move instructions			p015	p0	p1	p5	p2	p3	p4		
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 MOVXSD	r,r	1	1	x	x	x				1	0.33
MOVSX MOVZX	r,m	1					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		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		9
SFENCE		2						1	1		9
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
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)		4	4							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	3	3							18	12
DIV IDIV	r16	5	5							18-26 c)	12-20 c)
DIV IDIV	r32	4	4							18-42 c)	12-36 c)
DIV	r64	32	32							29-61 c)	18-37 c)
IDIV	r64	56	56							39-72 c)	28-40 c)
DIV IDIV	m8	4	3				1			18	12
DIV IDIV	m16	6	5				1			18-26 c)	12-20 c)
DIV IDIV	m32	5	4				1			18-42 c)	12-36 c)
DIV	m64	32	31				1			29-61 c)	18-37 c)
IDIV	m64	56	55				1			39-72 c)	28-40 c)
CBW CWDE CDQE		1	1	x	x	x				1	
CWD CDQ CQO		1	1	x		x				1	
Logic instructions											
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
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,l	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,l	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	10	9	x	x	x	1				5
BT	m,i	2	1	x	x	x	1				1
BTR BTS BTC	r,r/i	1	1	x	x	x				1	
BTR BTS BTC	m,r	11	8	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				2
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		7	7	x	x	x					4
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 and branch e,i)		1	1			1				0	1
Fused 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>											
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
NOP (0F 1F mod000rm)		1	1	x	x	x					1
PAUSE		3	3	x	x	x					8
CLI											
STI											
ENTER	i,0	12	10					1	1		8
ENTER	a,b										
LEAVE		3	2				1				
CPUID		46- 100									180-215
RDTSC		29									64
RDPMSR		23									54

**Notes:**

- Applies to all addressing modes
- Has an implicit LOCK prefix.
- Low values are for small results, high values for high results.
- See manual 3: "The microarchitecture of Intel and AMD CPU's" for restrictions on macro-op fusion.
- Not available in 64 bit mode.

## 5.2 Floating point x87 instructions

Instruction	Operands	jops fused domain	jops unfused domain							Unit	Latency	Reciprocal throughput
Move instructions			p015	p0	p1	p5	p2	p3	p4			
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
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	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
Arithmetic instructions												
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	
EXTRACT		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.

### 5.3 Integer MMX and XMM instructions

Instruction	Operands	jops fused domain	jops unfused domain							Unit	Latency	Reciprocal throughput
Move instructions			p015	p0	p1	p5	p2	p3	p4			
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
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
PMOVBMSKB	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
PMULHSW h)	(x)mm,(x)mm	1	1		1					int	3	1
PMULHSW 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
PASB PABSW PABSD h)	(x)mm,(x)mm	1	1	x		x				int	1	0.5
PASB 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
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
<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/RDQ	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.

## 5.4 Floating point XMM instructions

Instruction	Operands	µops fused domain	µops unfused domain							Unit	Latency	Reciprocal throughput
Move instructions			p015	p0	p1	p5	p2	p3	p4			
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
UNPCKH/LPD	xmm,m128	2	1	1			1			float		1

Conversion												
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
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.

## 6 List of instruction timings and $\mu$ ops breakdown for Core 2 (45nm)

### 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 dependence 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.

## 6.1 Integer instructions

Instruction	Operands	jops fused domain	jops unfused domain							Latency	Reciprocal throughput
Move instructions			p015	p0	p1	p5	p2	p3	p4		
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 MOVXSD	r,r	1	1	x	x	x				1	0.33
MOVSX MOVZX	r16/32,m	1					1				1
MOVSX MOVXSD	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		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
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
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	
Logic instructions											
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
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,l	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,l	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 and branch e,i)		1	1			1				0	1
Fused 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>											
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
NOP (0F 1F mod000rm)		1	1	x	x	x					1
PAUSE		3	3	x	x	x					8
CLI											
STI											
ENTER	i,0	12	10					1	1		8
ENTER	a,b										
LEAVE		3	2				1				
CPUID		53-117									53-211
RDTSC		13									32
RDPMC		23									54

**Notes:**

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

## 6.2 Floating point x87 instructions

Instruction	Operands	jops fused domain	jops unfused domain							Unit	Latency	Reciprocal throughput
Move instructions			p015	p0	p1	p5	p2	p3	p4			
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
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
Arithmetic instructions												
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	26-29	x	x	x				float	13-40	
FPREM1		28-35	28-35	x	x	x				float	18-41	
FRNDINT		17-19	17-19	x	x	x				float	10-22	
<b>Math</b>												
FSCALE		28	28	x	x	x				float	43	
EXTRACT		53-84	53-84	x	x	x				float	~170	
FSQRT		1	1	1						float	6-20	
FSIN		18-85	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	57-65	x	x	x				float	50-100	
FPTAN		19-100		x	x	x				float	40-130	
FPATAN		23-87	23-87	x	x	x				float	55-130	
<b>Other</b>												
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.

### 6.3 Integer MMX and XMM instructions

Instruction	Operands	jops fused domain	jops unfused domain							Unit	Latency	Reciprocal throughput
Move instructions			p015	p0	p1	p5	p2	p3	p4			
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
PMOVSX/ZXBW j)	xmm,xmm	1	1			1				int	1	1
PMOVSX/ZXBW j)	xmm,m64	1	1			1	1			int		1
PMOVSX/ZXBD j)	xmm,xmm	1	1			1				int	1	1
PMOVSX/ZXBD j)	xmm,m32	1	1			1	1			int		1
PMOVSX/ZXBQ j)	xmm,xmm	1	1			1				int	1	1
PMOVSX/ZXBQ j)	xmm,m16	1	1			1	1			int		1
PMOVSX/ZXWD j)	xmm,xmm	1	1			1				int	1	1
PMOVSX/ZXWD j)	xmm,m64	1	1			1	1			int		1
PMOVSX/ZXWQ j)	xmm,xmm	1	1			1				int	1	1
PMOVSX/ZXWQ j)	xmm,m32	1	1			1	1			int		1
PMOVSX/ZXDQ j)	xmm,xmm	1	1			1				int	1	1
PMOVSX/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
PBLENDDVB j)	xmm,xmm,xmm0	2	2			2				int	2	2
PBLENDDVB j)	xmm,m,xmm0	2	2			2	1			int		2
PBLENDDW j)	xmm,xmm,i	1	1			1				int	1	1
PBLENDDW 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
PMOVBMSKB	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
<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)	(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
PHADDD PHSUBD h)	(x)mm, (x)mm	3	3	1		2				int	3	2
PHADDD 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
PMULHSW h)	(x)mm,(x)mm	1	1		1					int	3	1
PMULHSW 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
PASB PABSW PABSD h)	(x)mm,(x)mm	1	1	x		x				int	1	0.5
PASB 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
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

## 6.4 Floating point XMM instructions

Instruction	Operands	jops fused domain	jops unfused domain							Unit	Latency	Reciprocal throughput
Move instructions			p015	p0	p1	p5	p2	p3	p4			
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 MOVLPD/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
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			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			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			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

Conversion												
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
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
DPSS j)	xmm,xmm,i	4	4	2	2					float	11	3
DPSS j)	xmm,m128,i	4	4	2	2		1			float		3
DPPD j)	xmm,xmm,i	4	4	x	x	x				float	9	3
DPPD j)	xmm,xmm,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.

## 7 List of instruction timings and $\mu$ op breakdown for P4

This list is measured for a Pentium 4, model 2. Timings for model 3 may be more like the values for P4E, listed in chapter 8.

### 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.

**$\mu$ 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:** The number of clock cycles from the execution of an instruction begins to the next dependent instruction can begin, if the latter 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 in manual 3: "The microarchitecture of Intel and AMD CPU's". Avoid making optimizations that rely on the latency of memory operations.

**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.

**Backwards compatibility:** Indicates the first microprocessor in the Intel 80x86 family that supported the instruction.

## 7.1 integer instructions

Instruction	Operands	μops	Microcode	Latency	Additional latency	Reciprocal throughput	Port	Execution unit	Subunit	Backwards compatibility	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	
PREFETCHCNTA	m	4	2			6				sse	
PREFETCHT0/1/2	m	4	2			6				sse	
SFENCE		4	2			40				sse	
LFENCE		4	2			38				sse2	
MFENCE		4	2			100				sse2	

Arithmetic instructions											
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	$\geq 8$		$\geq 4$				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	$\geq 9$		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		$\geq 4$				86	
NEG	r	1	0	0.5	0.5-1	0.5	0	alu0		86	
NEG	m	3	0			$\geq 3$				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	
Logic											
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	$\geq 1$	0.5-1	$\geq 1$				86	c
AND, OR, XOR	m,r	3	0	$\geq 8$		$\geq 4$				86	c
TEST	r,r	1	0	0.5	0.5-1	0.5	0	alu0		86	c
TEST	r,m	2	0	$\geq 1$	0.5-1	$\geq 1$				86	c
NOT	r	1	0	0.5	0.5-1	0.5	0	alu0		86	
NOT	m	4	0			$\geq 4$				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,l	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,l	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	
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	$\approx 4n+36$						86	
STOS		4	2	6		6				86	
REP STOS		4	2n+3	$\approx 3n+10$						86	
MOVS		4	4	6		4				86	
REP MOVS		4	$\approx 163+1.1n$			$\approx n$				86	
SCAS		4	3			6				86	
REP SCAS		4	$\approx 40+6n$			$\approx 4n$				86	
CMPS		4	5			8				86	
REP CMPS		4	$\approx 50+8n$			$\approx 4n$				86	
<b>Other</b>											
NOP (90)		1	0	0		0.25	0/1	alu0/1		86	
NOP (0F 1F mod000rm)		1	0	0		0.25	0/1	alu0/1		ppro	
PAUSE		4	2							sse2	
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.

## 7.2 Floating point x87 instructions

Instruction	Operands	$\mu$ ops	Microcode	Latency	Additional latency	Reciprocal throughput	Port	Execution unit	Subunit	Backwards compatibility	Notes
<b>Move instructions</b>											
FLD	r	1	0	6	0	1	0	mov		87	
FLD	m32/64	1	0	$\approx 7$	0	1	2	load		87	
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	$\approx 7$		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	$\approx 10$		6	2	load		87	
FILD	m32/64	2	0	$\approx 10$		1	2	load		87	
FIST	m16	3	0	$\approx 10$		2-4	0	store		87	
FIST	m32/64	2	0	$\approx 10$		2-3	0	store		87	
FISTP	m	3	0	$\approx 10$		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	
<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	
FXTRACT		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  $\mu$ ops more and 40-80 clocks more when XMM registers are disabled.

### 7.3 Integer MMX and XMM instructions

Instruction	Operands	$\mu$ ops	Microcode	Latency	Additional latency	Reciprocal throughput	Port	Execution unit	Subunit	Backwards compatibility	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	
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
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/t/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  $\mu\text{op}$  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

## 7.4 Floating point XMM instructions

Instruction	Operands	$\mu$ ops	Microcode	Latency	Additional latency	Reciprocal throughput	Port	Execution unit	Subunit	Backwards compatibility	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	
<b>Conversion</b>											
CVTTPS2PD	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/D MINPS/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.
- 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.

## 8 List of instruction timings and $\mu$ op breakdown for P4E

### 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:** The number of clock cycles from the execution of an instruction begins to the next dependent instruction can begin, if the latter 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 in manual 3: "The microarchitecture of Intel and AMD CPU's". Avoid making optimizations that rely on the latency of memory operations.

**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.

**Backwards compatibility:** Indicates the first microprocessor in the Intel 80x86 family that supported the instruction.

## 8.1 Integer instructions

Notes	Backwards compatibility	Subunit	Execution unit	Port	Reciprocal throughput	Additional latency	Latency	Microcode	μops	Operands	Instruction
<b>Move instructions</b>											
	86		alu0/1	0/1	0.25	0	1	0	1	r,r	MOV
c	86		alu0/1	0/1	0.25	0	1	0	1	r8/16/32,i	MOV
	x64		alu0/1	0/1	0.5	0		0	1	r64,i32	MOV
	x64		alu1	1	1	0		0	2	r64,i64	MOV
	86		load	2	1	0	3	0	2	r8/16,m	MOV
	86		load	2	1	0	2	0	1	r32/64,m	MOV
b,c	86		store	0	2				1	m,r	MOV
	86		store	0,3	2				2	m,i	MOV
	x64		store	0,3	2				2	m64,i32	MOV
	86				8				1	r,sr	MOV
a,q	86				27				1	sr,r/m	MOV
ℓ	x64				1				3	r,mabs	MOV
ℓ	x64				2				3	mabs,r	MOV
	sse2				2				2	m,r32	MOVNTI
c	386		alu0/1	0/1	0.25	0	1	0	1	r,r	MOVZX
c	386		alu0/1	0/1	1	0	2	0	2	r16,r8	MOVZX
	386		load	2	1	0	2	0	1	r,m	MOVZX
a,c,o	386		alu0	0	0.5	0	1	0	1	r16,r8	MOVSX
a,c,o	386		alu0	0	0.5	0	1	0	1	r32/64,r8/16	MOVSX
	386		load	2	1	0	3	0	2	r,m	MOVSX
a	x64		alu0	0	0.5	0	1	0	1	r64,r32	MOVSXD
a,e	PPro				3	0	9.5	0	3	r,r/m	CMOVcc
	86		alu0/1	0/1	1	0	2	0	3	r,r	XCHG
	86						≈ 100		2	r,m	XCHG
	86						6	0	4		XLAT
	86				2		2	0	2	r	PUSH
	186				2		2	0	2	i	PUSH
	86				2		2	0	3	m	PUSH
	86				9			3	1	sr	PUSH
	86				9			3	1		PUSHF(D/Q)
m	186				16			9	1		PUSHA(D)
	86				1	0	1	0	2	r	POP
	86				10			6	2	m	POP
	86				30			8	1	sr	POP
	86				70			8	1		POPF(D/Q)
m	186				15			16	2		POPA(D)
p	86		alu0/1	0/1	0.25			0	1	r,[m]	LEA
	86		alu0/1	0/1	0.25	0	2.5	0	1	r,[r+r/i]	LEA
	86		alu0/1	0/1	0.5	0	3.5	0	2	r,[r+r+i]	LEA
	386		alu	1	1	0	3.5	0	3	r,[r*i]	LEA
	386		alu0,1	0,1	1	0	3.5	0	2	r,[r+r*i]	LEA
	386		alu	1	1	0	3.5	0	3	r,[r+r*i+i]	LEA
n	86		int	1		0	4	0	1		LAHF
d,n	86		alu0/1	0/1		0	5	0	1		SAHF
m	86		int	1	1	0		0	2		SALC
m	86				28				2	r,m	LDS, LES, ...

LODS		1	3	8		8				86	
REP LODS		1	5n	$\approx 4n+50$						86	
STOS		1	2	8		8				86	
REP STOS		1	2.5n	$\approx 3n$						86	
MOVS		1	4	8		8				86	
REP MOVSB		9	$\approx 0.3 n$			$\approx 0.3n$				86	
REP MOVSW		1	$\approx 0.5\text{-}1.1 n$			$\approx 0.6\text{-}1.4 n$				86	
REP MOVSD		1	$\approx 1.1 n$			$\approx 1.4 n$				86	
REP MOVSQ		1	$\approx 1.1 n$			$\approx 1.4 n$				x64	
BSWAP	r	1	0	1	0	1		alu		486	
IN, OUT	r,r/i	1	52			>1000				86	
PREFETCHCNTA	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	

### Arithmetic instructions

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
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,l	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,l	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	
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	
NOP (0F 1F mod000rm)		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	
RDPMC (bit 31 = 1)		1	37			100				p5	
RDPMC (bit 31 = 0)		4	154			240				p5	
MONITOR										(sse3)	
MWAIT										(sse3)	

#### Notes:

- Add 1  $\mu$ op if source is a memory operand.
- Uses an extra  $\mu$ op (port 3) if SIB byte used.
- Add 1  $\mu$ op if source or destination, but not both, is a high 8-bit register (AH, BH, CH, DH).
- Has (false) dependence on the flags in most cases.
- Not available on PMMX
- Move accumulator to/from memory with 64 bit absolute address (opcode A0 - A3).
- Not available in 64 bit mode.
- Not available in 64 bit mode on some processors.
- MOV<sub>VSX</sub> 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.
- 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.
- 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.



## 8.2 Floating point x87 instructions

Instruction	Operands	µops	Microcode	Latency	Additional latency	Reciprocal throughput	Port	Execution unit	Subunit	Backwards compatibility	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	
FLDI		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	
FXTRACT		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	
<b>Other</b>											
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 μops when XMM registers are disabled, but the throughput is the same.

### 8.3 Integer MMX and XMM instructions

Instruction	Operands	μops	Microcode	Latency	Additional latency	Reciprocal throughput	Port	Execution unit	Subunit	Backwards compatibility	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	2	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	
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/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	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	
PMOVMASKB	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 <sub>2</sub> j
PMADDWD	r,r/m	1	0	7	1	1,2	1	fp	mul	mmx	a <sub>2</sub> j
PMULUDQ	r,r/m	1	0	7	1	1,2	1	fp	mul	sse2	a <sub>2</sub> j
PAVGB/W	r,r/m	1	0	2	1	1,2	1	mmx	alu	sse	a <sub>2</sub> j
PMIN/MAXUB	r,r/m	1	0	2	1	1,2	1	mmx	alu	sse	a <sub>2</sub> j
PMIN/MAXSW	r,r/m	1	0	2	1	1,2	1	mmx	alu	sse	a <sub>2</sub> j
PSADBW	r,r/m	1	0	4	1	1,2	1	mmx	alu	sse	a <sub>2</sub> j
<b>Logic</b>											
PAND, PANDN	r,r/m	1	0	2	1	1,2	1	mmx	alu	mmx	a <sub>2</sub> j
POR, PXOR	r,r/m	1	0	2	1	1,2	1	mmx	alu	mmx	a <sub>2</sub> j
PSLL/RLW/D/Q, PSRAW/D	r,i/r/m	1	0	2	1	1,2	1	mmx	shift	mmx	a <sub>2</sub> 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 [LDDQU](#).

## 8.4 Floating point XMM instructions

Instruction	Operands	µops	Microcode	Latency	Additional latency	Reciprocal throughput	Port	Execution unit	Subunit	Backwards compatibility	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
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
RCPPS RCPSS	r,r/m	2	0	6	1	4	1	mmx		sse	a
MAXPS/D MAXSS/D MINPS/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](#).

## 9 Instruction timings and macro-operation breakdown for AMD K7

### 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.

**Ops:** Number of macro-operations issued from instruction decoder to schedulers. Instructions with more than 2 macro-operations are vector-path instructions.

**Latency:** The number of clock cycles from the execution of an instruction begins to the next dependent instruction can begin. 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).

**Reciprocal throughput:** 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.

**Execution unit:** 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.

### 9.1 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
MOV	r16,m16	1	4	1/2	ALU, AGU	

MOV	r32,m32	1	3	1/2	AGU	segment base $\neq 0$
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		
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 address size
LEA	r32,[m]	1	2	1/3	AGU	Any address 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	
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,l	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,l	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	
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	recip. thrp.= 2 if jump
J(E)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
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 are per count
STOS		4	2	2		
REP STOS		3	1	1		values are per count
MOVS		7	3	3		
REP MOVS		4	1-4	1-4		values are per count
SCAS		5	2	2		
REP SCAS		5	2	2		values are per count
CMPS		7	6	6		
REP CMPS		6	3-4	3-4		values are per count
<b>Other</b>						
NOP (90)		1	0	1/3	ALU	
NOP (0F 1F mod000rm)		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		

## 9.2 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	
FNSTSW	m16	2		8	FMISC, ALU	
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
FIDIV(R)	m	2	12-26	9-23	FMUL,FMISC	
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			
FXTRACT		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		

### 9.3 Integer MMX instructions

Instruction	Operands	Ops	Latency	Reciprocal throughput	Execution unit	Notes
<b>Move instructions</b>						
MOVD	r32, mm	2	7	2	FMICS, 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 PACKUSWB	mm,r/m	1	2	2	FA/M	
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		
PMOVMKB	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 PADDSB/W ADDUSB/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	

## 9.4 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	
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	
ADDPS SUBPS	r,r/m	2	4	2	FADD	
MULSS	r,r/m	1	4	1	FMUL	
MULPS	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.
DIVPS	r,r/m	2	18-30	18-30	FMUL	
RCPSS	r,r/m	1	3	1	FMUL	
RCPPS	r,r/m	2	3	2	FMUL	
MAXSS MINSS	r,r/m	1	2	1	FADD	
MAXPS MINPS	r,r/m	2	2	2	FADD	
CMPccSS	r,r/m	1	2	1	FADD	
CMPccPS	r,r/m	2	2	2	FADD	
COMISS UCOMISS	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	
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		

## 9.5 3DNow instructions

These instructions are not available on Intel processors

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	

# 10 Instruction timings and macro-operation breakdown for AMD K8

## 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.

**Ops:** Number of macro-operations issued from instruction decoder to schedulers. Instructions with more than 2 macro-operations are vector-path instructions.

**Latency:** The number of clock cycles from the execution of an instruction begins to the next dependent instruction can begin. 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).

**Reciprocal throughput:** 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.

**Execution unit:** 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.

## 10.1 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
MOV	r16,m16	1	4	1/2	ALU, AGU	



MOV	r32,m32	1	3	1/2	AGU	segment base $\neq 0$
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		
MOVNTI	m,r	1		2-3	AGU	
MOVZX, MOVSX	r,r	1	1	1/3	ALU	
MOVZX, MOVSX	r,m	1	4	1/2	ALU, AGU	
MOVSXD	r64,r32	1	1	1/3	ALU	
MOVSXD	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	
PREFETCHCNTA	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	
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,l	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	
RCL, RCR	m,l	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
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 are per count
STOS		4	2	2		
REP STOS		1.5 - 2	0.5 - 1	0.5 - 1		values are per count
MOVS		7	3	3		
REP MOVS		3	1-2	1-2		values are per count
SCAS		5	2	2		
REP SCAS		5	2	2		values are per count
CMPS		2	3	3		
REP CMPS		6	2	2		values are per count
<b>Other</b>						
NOP (90)		1	0	1/3	ALU	
NOP (0F 1F mod000rm)		1	0	1/3	ALU	
ENTER	i,0	12	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		6	10	7		
RDPMSR		9	12	7		

## 10.2 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	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	Low latency immediately after FCOMI
FFREE	r	1		2	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	
FNSTSW	m16	2		8	FMISC, ALU	
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	
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		

### 10.3 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	
MOVD (MOVQ)	xmm, r64	3	9	2	FANY, ALU	
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/xmm	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		
PMOVBMSKB	r32, mm/xmm	1	2	1	FADD	
PEXTRW	r32, mm/xmm, 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 ADDUSB/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	mm,r/m	1	2	1/2	FA/M	
PCMPEQ/GT B/W/D	xmm,r/m	2	2	1	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	mm,r/m	1	3	1	FMUL	
PMADDWD	xmm,r/m	2	3	2	FMUL	
PAVGB/W	mm,r/m	1	2	1/2	FA/M	
PAVGB/W	xmm,r/m	2	2	1	FA/M	
PMIN/MAX SW/UB	mm,r/m	1	2	1/2	FA/M	
PMIN/MAX SW/UB	xmm,r/m	2	2	1	FA/M	
PSADBW	mm,r/m	1	3	1	FADD	
PSADBW	xmm,r/m	2	3	2	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	xmm,i/xmm/m	2	2	1	FA/M	
PSLLDQ, PSRLDQ	xmm,i	2	2	1	FA/M	
<b>Other</b>						
EMMS		1		1/3	FANY	

## 10.4 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	
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	
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>						
CVTTPS2PD	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	
MULSS/D	r,r/m	1	4	1	FMUL	
MULPS/D	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.
DIVPS	r,r/m	2	18-30	18-30	FMUL	
DIVSD	r,r/m	1	11-20	8-17	FMUL	
DIVPD	r,r/m	2	16-34	16-34	FMUL	
RCPSS	r,r/m	1	3	1	FMUL	



RCPPS	r,r/m	2	3	2	FMUL	
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		

## 10.5 3DNow instructions

These instructions are not available on Intel processors

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	

# 11 Instruction timings and macro-operation breakdown for AMD K10

## 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.

**Ops:** Number of macro-operations issued from instruction decoder to schedulers. Instructions with more than 2 macro-operations are vector-path instructions.

**Latency:** The number of clock cycles from the execution of an instruction begins to the next dependent instruction can begin. 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).

**Reciprocal throughput:** 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.

**Execution unit:** 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.

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

## 11.1 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.

MOV	r16,m16	1	4	1/2	ALU, AGU	Add 1 clock if code segment base $\neq 0$
MOV	r32,m32	1	3	1/2	AGU	
MOV	r64,m64	1	3	1/2	AGU	
MOV	m8,r8H	1	?	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
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	
MOVSXD	r64,r32	1	1	1/3	ALU	
MOVSXD	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
LEA	r32/64,[m]	1	1	1/3	ALU	$\leq 2$ source operands
LEA	r32/64,[m]	1	2	1/3	AGU	W. scale or 3 operands
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	
PREFETCHCNTA	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	
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 of significant bits in absolute value of dividend. See AMD software optimization guide.
DIV	r32/m32		15-46	15-46	ALU	
DIV	r64/m64		15-78	15-78	ALU	
IDIV	r16/m16		24-39	24-39	ALU	
IDIV	r32/m32		24-55	24-55	ALU	
IDIV	r64/m64		24-87	24-87	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		1/2	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,l	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,ROR	m,i /CL	1	7	1	ALU, AGU	
RCL, RCR	m,l	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	
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		
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	
NOP (0F 1F mod000rm)		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		
RDPMSR		13		5		

## 11.2 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	
FNSTSW	m16	2		9	FMISC, ALU	
FNSTCW	m16	3		2	FMISC, ALU	
FLDCW	m16	12		14	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	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		
FXTRACT		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		



### 11.3 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	
MOVD	xmm, r32	2	6	3		
MOVD	xmm,m32	1	2	1/2		
MOVD	m32,mm/xmm	1	2	1	FMISC	
MOVD (MOVQ)	r64,mm/xmm	1	3	1	FADD	Moves 64 bits. Name of instruction differs
MOVD (MOVQ)	mm,r64	2	6	3		
MOVD (MOVQ)	xmm,r64	2	6	3	FMUL, ALU	
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,mm/xmm	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		
PMOVBMSKB	r32,mm/xmm	1	3	1	FADD	
PEXTRW	r32,mm/xmm,i	2	6	1		
PINSRW	mm/xmm,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 ADDUSB/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	xmm,i/xmm/m	1	3	1/2	FA/M	
PSLLDQ, PSRLDQ	xmm,i	1	3	1/2	FA/M	
<b>Other</b>						
EMMS		1		1/3	FANY	

## 11.4 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	
MOVHLPs, 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	
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		
CVTP12PS	xmm,mm	2	7	1		
CVTP12PD	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>						
LDMXCSR	m	12	12	10		
STMXCSR	m	3	12	11		

## 11.5 3DNow instructions

These instructions are not available on Intel processors

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	



## 12 Instruction set compatibility table

The following table lists which instruction sets are supported on which microprocessors. This is intended as an aid in deciding which instruction sets to use and whether to add support for older microprocessors. The different instruction sets are explained below.

Processor	Introduction year	80186	CPUID	PPro	MMX	SSE	SSE2	SSE3	Suppl. SSE3	SSE4.1	SSE4.2	32 bit	64 bit
<b>Intel processors</b>													
8086, 8088	1978												
80186	1982	x											
80286	1982	x											
80386	1985	x											
80486	1989	x	s										
Pentium	1993	x	x									x	
Pentium Pro	1995	x	x	x								x	
Pentium MMX	1997	x	x		x							x	
Pentium II	1997	x	x	x	x							x	
Pentium III	1999	x	x	x	x	x						x	
Pentium 4	2000	x	x	x	x	x	x					x	
Pentium 4 w. EM64T	2004	x	x	x	x	x	x	x				x	x
Pentium D	2005	x	x	x	x	x	x	x				x	x
Pentium Extreme ed.	2005	x	x	x	x	x	x	x				x	x
Celeron	1998	x	x	x	x	s	s					x	
Xeon	1998	x	x	x	x	s	s					x	
Pentium M	2003	x	x	x	x	x	x					x	
Core Solo	2006	x	x	x	x	x	x	x				x	
Core Duo	2006	x	x	x	x	x	x	x				x	
Core 2, 65 nm	2006	x	x	x	x	x	x	x	x			x	x
Core 2, 45 nm	2007	x	x	x	x	x	x	x	x	x		x	x

x = supported, s = supported in some versions.

Processor	Introduction year	80186	CPUID	PPro	MMX	SSE	SSE2	SSE3	Suppl. SSE3	3DNow	3DNowE	32 bit	64 bit
<b>AMD processors</b>													
Am286	1986?	x											
Am386	1991	x										x	
Am486	1993	x	s									x	
K5	1996	x	x									x	
K6	1997	x	x		x					s		x	
Athlon	1999	x	x	x	x	s				x	x	x	
Duron	2000	x	x	x	x	x				x	x	x	

Processor	Introduction year	SSE2	SSE3	Suppl. SSE3	SSE4.1/SSE4.2	AMD SSE4A	AMD SSE5	32 bit	64 bit
<b>AMD processors</b>									
Sempron, K7	2004	s	s					x	s
Athlon 64, K8	2003	x	s					x	x
Opteron, K8	2003	x	s					x	x
K10	2007	x	x			x		x	x
?	2009?	x	x	x?		x	x	x	x

## 12.1 Explanation of instruction sets

The availability of a particular instruction set should be tested with the [CPUID](#) instruction, if available. The different instruction sets are explained below.

### x86

This is the name of the common instruction set, supported by all processors in this lineage.

### 80186

This is the first extension to the x86 instruction set. New integer instructions: [PUSH i](#), [PUSHA](#), [POPA](#), [IMUL r, r, i](#), [BOUND](#), [ENTER](#), [LEAVE](#), shifts and rotates by immediate  $\neq 1$ .

### 80286

System instructions for 16-bit protected mode.

### 80386

The eight general purpose registers are extended from 16 to 32 bits. 32-bit addressing. 32-bit protected mode. Scaled index addressing. [MOVZX](#), [MOVSX](#), [IMUL r, r](#), [SHLD](#), [SHRD](#), [BT](#), [BTR](#), [BTS](#), [BTC](#), [BSF](#), [BSR](#), [SETcc](#).

### 80486

[BSWAP](#). Later versions have [CPUID](#).

### x87

This is the floating point instruction set. Supported in 8086/8088 and later processors when a 8087 or later coprocessor is present. Some 486 processors and all processors since Pentium/K5 have built-in support for floating point instructions without the need for a coprocessor.

### 80287

[FSTSW AX](#).

### 80387

[FPREM1](#), [FSIN](#), [FCOS](#), [FSINCOS](#).

### Pentium

[RDTSC](#), [RDPMC](#).



## PPro

Conditional move ([CMOV](#), [FCMOV](#)) and fast floating point compare ([FCOMI](#)) instructions introduced in Pentium Pro. These instructions are not supported in Pentium MMX, but are supported in all processors with SSE and later.

## MMX

Integer vector instructions with packed 8, 16 and 32-bit integers in the 64-bit MMX registers [MM0](#) - [MM7](#), which are aliased upon the floating point stack registers [ST\(0\)](#) - [ST\(7\)](#).

## SSE

Single precision floating point scalar and vector instructions in the new 128-bit XMM registers [XMM0](#) - [XMM7](#). [PREFETCH](#), [SFENCE](#), [FXSAVE](#), [FXRSTOR](#), [MOVNTQ](#), [MOVNTPS](#). The use of XMM registers requires operating system support.

## SSE2

Double precision floating point scalar and vector instructions in the 128-bit XMM registers [XMM0](#) - [XMM7](#). 64-bit integer arithmetics in the MMX registers. Integer vector instructions with packed 8, 16, 32 and 64-bit integers in the XMM registers. [MOVNTI](#), [MOVNTPD](#), [PAUSE](#), [LFENCE](#), [MFENCE](#).

## SSE3

[FISTTP](#), [LDDQU](#), [MOVDDUP](#), [MOVSHDUP](#), [MOVSLDUP](#), [ADDSUBPS](#), [ADDSUPPD](#), [HADDPS](#), [HADDPD](#), [HSUBPS](#), [HSUBPD](#).

## Supplementary SSE3

[PSHUFB](#), [PHADDW](#), [PHADDSW](#), [PHADDQ](#), [PMADDUBSW](#), [PHSUBW](#), [PHSUBSW](#), [PHSUBD](#), [PSIGNB](#), [PSIGNW](#), [PSIGND](#), [PMULHRSW](#), [PABSB](#), [PABSW](#), [PABSD](#), [PALIGNR](#).

## MONITOR

The instructions [MONITOR](#) and [MWAIT](#) are available in some Intel and AMD multiprocessor CPU's with SSE3.

## SSE4.1 (Intel only)

[BLENDPD](#), [BLENDPS](#), [BLENDVPD](#), [BLENDVPS](#), [DPPD](#), [DPPS](#), [EXTRACTPS](#), [INSERTPS](#), [MOVNTDQA](#), [MPSADBW](#), [PACKUSDW](#), [PBLENDVB](#), [PBLENDW](#), [PCMPEQQ](#), [PEXTRB](#), [PEXTRD](#), [PEXTRQ](#), [PEXTRW](#), [PHMINPOSUW](#), [PINSRB](#), [PINSRD](#), [PINSRQ](#), [PMAXSB](#), [PMAXSD](#), [PMAXUD](#), [PMAXUW](#), [PMINSB](#), [PMINSW](#), [PMINUD](#), [PMINUW](#), [PMOVSXBD](#), [PMOVSXBQ](#), [PMOVSXBW](#), [PMOVSXWD](#), [PMOVSXWQ](#), [PMOVSXDQ](#), [PMOVZXBQ](#), [PMOVZXBW](#), [PMOVZXWD](#), [PMPVZXWQ](#), [PMOVZXDQ](#), [PMULDQ](#), [PMULLD](#), [PTEST](#), [ROUNDPD](#), [ROUNDPS](#), [ROUNDSD](#), [ROUNDSS](#).

## SSE4.2 (Intel only. Future)

[CRC32](#), [PCMPESTRI](#), [PCMPESTRM](#), [PCMPISTRI](#), [PCMPISTRM](#), [PCMPGTQ](#), [POPCNT](#).

## AES (Intel only. Future)

[AESDEC](#), [AESDECLAST](#), [AESENC](#), [AESENCLAST](#), [AESIMC](#), [AESKEYGENASSIST](#).

## CLMUL (Intel only. Future)

[PCLMULQDQ](#).

## AVX (Intel only. Future)

The 128-bit XMM registers are extended to 256-bit YMM registers with room for further extension in the future. The use of YMM registers requires operating system support.

Almost all previous XMM instructions now have two versions: with and without zero-extension into the full YMM register. The zero-extension versions have three operands in most cases. New instructions in AVX: [VBROADCASTSS](#), [VBROADCASTSD](#), [VEXTRACTF128](#), [VINSETRF128](#), [VLDMXCSR](#), [VMASKMOVPS](#), [VMASKMOVPD](#), [VPERMILPD](#), [VPERMIL2PD](#), [VPERMILPS](#), [VPERMIL2PS](#), [VPERM2F128](#), [VSTMXCSR](#), [VZEROALL](#), [VZERoupper](#).

### FMA (Intel only. Future)

Fused multiply and add instructions: [VFMADDPD](#), [VFMADDPS](#), [VFMADDSD](#), [VFMADDSS](#), [VFMADDSUBPD](#), [VFMADDSUBPS](#), [VFMSUBADDPD](#), [VFMSUBADDPS](#), [VFMSUBPD](#), [VFMSUBPS](#), [VFMSUBSD](#), [VFMSUBSS](#), [VFNMADDPD](#), [VFNMADDPS](#), [VFNMADDSD](#), [VFNMADDSS](#), [VFNMSUBPD](#), [VFNMSUBPS](#), [VFNMSUBSD](#), [VFNMSUBSS](#).

### 3DNow (AMD only)

Single precision floating point vector instructions in the 64-bit MMX registers. Only available on AMD processors. The 3DNow instructions are: [FEMMS](#), [PAVGUSB](#), [PF2ID](#), [PFACC](#), [PFADD](#), [PFCMPEQ/GT/GE](#), [PFMAX](#), [PFMIN](#), [PFRCP/IT1/IT2](#), [PFRSQRT/IT1](#), [PFSUB](#), [PFSUBR](#), [PI2FD](#), [PMULHRW](#), [PREFETCH/W](#).

### 3DNowE (AMD only)

[PF2IW](#), [PFNACC](#), [PFPNACC](#), [PI2FW](#), [PSWAPD](#).

### SSE4.a (AMD only)

[EXTRQ](#), [INSRTQ](#), [LZCNT](#), [MOVNTSD](#), [MOVNTSS](#), [POPCNT](#). ([POPCNT](#) shared with Intel SSE4.2).

### SSE5 (AMD only. Future)

[COMPD](#), [COMPS](#), [COMSD](#), [COMSS](#), [CVTPH2PS](#), [CVTPS2PH](#), [FMADDPD](#), [FMADDPS](#), [FMADDSD](#), [FMADDSS](#), [FMSUBPD](#), [FMSUBPS](#), [FMSUBSD](#), [FMSUBSS](#), [FNMADDPD](#), [FNMADDPS](#), [FNMADDSD](#), [FNMADDSS](#), [FNMSUBPD](#), [FNMSUBPS](#), [FNMSUBSD](#), [FNMSUBSS](#), [FRCZPD](#), [FRCZPS](#), [FRCZSD](#), [FRCZSS](#), [PCMOV](#), [PCOMB](#), [PCOMB](#), [PCOMD](#), [PCOMQ](#), [PCOMUB](#), [PCOMUD](#), [PCOMUQ](#), [PCOMUW](#), [PCOMW](#), [PERMPD](#), [PERMPS](#), [PHADDBD](#), [PHADDBQ](#), [PHADDBW](#), [PHADDDQ](#), [PHADDUBD](#), [PHADDUBQ](#), [PHADDUBW](#), [PHADDUDQ](#), [PHADDUWD](#), [PHADDUWQ](#), [PHADDWD](#), [PHADDWQ](#), [PHSUBBW](#), [PHSUBDQ](#), [PHSUBWD](#), [PMACSDQ](#), [PMACSDQH](#), [PMACSDQL](#), [PMACSSD](#), [PMACSSDQH](#), [PMACSSDQL](#), [PMACSSWD](#), [PMACSSWW](#), [PMACSWD](#), [PMACSWW](#), [PMADCSSWD](#), [PMADCSSW](#), [PPER](#), [PROTB](#), [PROTD](#), [PROTQ](#), [PROTW](#), [PSHAB](#), [PSHAD](#), [PSHAQ](#), [PSHAW](#), [PSHLB](#), [PSHLD](#), [PSHLQ](#), [PSHLW](#), [PTEST](#), [ROUNDPD](#), [ROUNDPS](#), [ROUNDSD](#), [ROUNDSS](#). The instructions [PTEST](#), [ROUNDPD](#), [ROUNDPS](#), [ROUNDSD](#), [ROUNDSS](#) are the same as in Intel SSE4.1.

### 64 bit

This instruction set is called x86-64, x64, AMD64 or EM64T. It defines a new 64-bit mode with 64-bit addressing and the following extensions: The general purpose registers are extended to 64 bits, and the number of general purpose registers is extended from eight to sixteen. The number of [XMM](#) registers is also extended from eight to sixteen, but the number of [MMX](#) and [ST](#) registers is still eight. Data can be addressed relative to the instruction pointer. There is no way to get access to these extensions in 32-bit mode.

### Instructions not available in 64 bit mode

The following instructions are not available in 64-bit mode: [PUSHA](#), [POPA](#), [BOUND](#), [INTO](#), BCD instructions: [AAA](#), [AAS](#), [DAA](#), [DAS](#), [AAD](#), [AAM](#), undocumented instructions ([SALC](#), [ICEBP](#), 82H alias for 80H opcode), [SYSENTER](#), [SYSEXIT](#), [ARPL](#). On some early Intel processors, [LAHF](#) and [SAHF](#) are not available in 64 bit mode. Increment and decrement register instructions cannot be coded in the short one-byte opcode form because these codes have been reassigned as REX prefixes.

Most instructions that involve segmentation are not available in 64 bit mode. Direct far jumps and calls are not allowed, but indirect far jumps, indirect far calls and far returns are allowed. These are used in system code for switching mode. Segment registers `DS`, `ES`, and `SS` cannot be used. `PUSH CS`, `PUSH DS`, `PUSH ES`, `PUSH SS`, `POP DS`, `POP ES`, `POP SS`, `LDS` and `LES` instructions are not allowed. `CS`, `DS`, `ES` and `SS` prefixes are allowed but ignored. The `FS` and `GS` segments and segment prefixes are available in 64 bit mode and are used for addressing thread environment blocks and processor environment blocks.

## 13 Comparison of the different microprocessors

The following table summarizes some important differences between different microprocessors.

	P1	PMMX	PPro	P2	P3	P4	P4E	PM	Core2	AMDK8	AMD K10
code cache, kb	8	16	8	16	16	≈ 64	≈ 78	32	32	64	64
code cache associativity, ways	2	4	4	4	4	4	8	8	8	2	2
data cache, kb	8	16	8	16	16	8	8-16	32	32	64	64
data cache associativity, ways	2	4	2	4	4	4	8	8	8	2	2
data cache line size	32	32	32	32	32	64	64	64	64	64	64
built-in level 2 cache, kb	0	0	0-512	0-512	128-512	128-512	256-2048	512-2048	2-6MB	512-1024	512
level 2 cache associativity, ways	0	0	4	4	8	8	8	8	16-24	16	16
level 2 cache bus size, bits	0	0	64	64	256	256	256	256	256	64	128
level 3 cache size	0	0	0	0	0	0	0	0	0	0	2 MB
branch target buffer entries	256	256	512	512	512	4096	4096	2048	2048	2048	2048
return stack buffer size	0	4	16	16	16	16	16	16	16	12	24
out of order execution	no	no	yes	yes	yes	yes	yes	yes	yes	yes	yes
branch prediction	poor	good	good	good	good	good	good	good	good	good	good
branch misprediction penalty	3-4	4-5	10-20	10-20	10-20	≥ 24	≥ 24	13	15	12	12
partial register stall	0	0	5	5	5	0	0	1-5	1-6	0	0
FMUL latency	3	3	5	5	5	6-7	7-8	5	5	4	4
IMUL latency	9	9	4	4	4	14	10-11	4	3	3	3

## 14 Literature

Intel: "Intel 64 and IA-32 Architectures Optimization Reference Manual".  
[developer.intel.com](http://developer.intel.com).

AMD: "Software Optimization Guide for AMD Family 10h Processors".  
[www.amd.com](http://www.amd.com).