

6.5 FPU Clock Counts

The CPU is functionally divided into the FPU, and the integer unit. The FPU processes floating point instructions only and does so in parallel with the integer unit.

For example, when the integer unit detects a floating point instruction without memory operands, after two clock cycles the instruction passes to the FPU for execution. The integer unit continues to execute instructions while the FPU executes the floating point instruction. If

another FPU instruction is encountered, the second FPU instruction is placed in the FPU queue. Up to four FPU instructions can be queued. In the event of an FPU exception, while other FPU instructions are queued, the state of the CPU is saved to ensure recovery.

6.5.1 FPU Clock Count Table

The clock counts for the FPU instructions are listed in Table 6-19 (Page 13). The abbreviations used in this table are listed in Table 6-21.

Table 6-21. FPU Clock Count Table Abbreviations

ABBREVIATION	MEANING
n	Stack register number
TOS	Top of stack register pointed to by SSS in the status register.
ST(1)	FPU register next to TOS
ST(n)	A specific FPU register, relative to TOS
M.WI	16-bit integer operand from memory
M.SI	32-bit integer operand from memory
M.LI	64-bit integer operand from memory
M.SR	32-bit real operand from memory
M.DR	64-bit real operand from memory
M.XR	80-bit real operand from memory
M.BCD	18-digit BCD integer operand from memory
CC	FPU condition code
Env Regs	Status, Mode Control and Tag Registers, Instruction Pointer and Operand Pointer



FPU Clock Counts

Table 6-22. 6x86 FPU Instruction Set Summary

FPU INSTRUCTION	OP CODE	OPERATION	CLOCK COUNT	NOTES
F2XMI Function Evaluation 2*-1 FABS Floating Absolute Value	D9 F0 D9 E1	TOS ← 2 ^{TOS-1} TOS ← TOS	92 - 108 2	See Note 2
FADD Floating Point Add Top of Stack 80-bit Register 64-bit Real 32-bit Real FADDP Floating Point Add, Pop FIADD Floating Point Integer Add 32-bit integer 16-bit integer	DC [1100 0 n] D8 [1100 0 n] DC [mod 000 r/m] D8 [mod 000 r/m] DE [1100 0 n] DA [mod 000 r/m] DE [mod 000 r/m]	ST(n) ← ST(n) + TOS TOS ← TOS + ST(n) TOS ← TOS + M.DR TOS ← TOS + M.SR ST(n) ← ST(n) + TOS; then pop TOS TOS ← TOS + M.SI TOS ← TOS + M.WI	4 - 9 4 - 9 4 - 9 4 - 9 4 - 9 8 - 14 8 - 14	
FCHS Floating Change Sign	D9 E0	TOS ← - TOS	2	
FCLEX Clear Exceptions FNCLEX Clear Exceptions	(0B)DB E2 DB E2	Wait then Clear Exceptions Clear Exceptions	5 3	
FCOM Floating Point Compare 80-bit Register 64-bit Real 32-bit Real FCOMP Floating Point Compare, Pop 80-bit Register 64-bit Real 32-bit Real FCOMPP Floating Point Compare, Pop Two Stack Elements FICOM Floating Point Compare 32-bit integer 16-bit integer FCOMP Floating Point Compare 32-bit integer 16-bit integer	D8 [1101 0 n] DC [mod 010 r/m] D8 [mod 010 r/m] D8 [1101 1 n] DC [mod 011 r/m] D8 [mod 011 r/m] DE D9 DA [mod 010 r/m] DE [mod 010 r/m] DA [mod 011 r/m] DE [mod 011 r/m]	CC set by TOS - ST(n) CC set by TOS - M.DR CC set by TOS - M.SR CC set by TOS - ST(n); then pop TOS CC set by TOS - M.DR; then pop TOS CC set by TOS - M.SR; then pop TOS CC set by TOS - ST(1); then pop TOS and ST(1) CC set by TOS - M.WI CC set by TOS - M.SI CC set by TOS - M.WI; then pop TOS CC set by TOS - M.SI; then pop TOS	4 4 4 4 4 4 4 9 - 10 9 - 10 9 - 10 9 - 10	
FCOS Function Evaluation: Cos(x)	D9 FF	TOS ← COS(TOS)	92 - 141	See Note 1
FDECSTP Decrement Stack Pointer	D9 F6	Decrement top of stack pointer	4	
FDIV Floating Point Divide Top of Stack 80-bit Register 64-bit Real 32-bit Real FDIVP Floating Point Divide, Pop FDIVR Floating Point Divide Reversed Top of Stack 80-bit Register 64-bit Real 32-bit Real	DC [1111 1 n] D8 [1111 0 n] DC [mod 110 r/m] D8 [mod 110 r/m] DE [1111 1 n] DC [1111 0 n] D8 [1111 1 n] DC [mod 111 r/m] D8 [mod 111 r/m]	ST(n) ← ST(n) / TOS TOS ← TOS / ST(n) TOS ← TOS / M.DR TOS ← TOS / M.SR ST(n) ← ST(n) / TOS; then pop TOS TOS ← ST(n) / TOS ST(n) ← TOS / ST(n) TOS ← M.DR / TOS TOS ← M.SR / TOS	24 - 34 24 - 34 24 - 34 24 - 34 24 - 34 24 - 34 24 - 34 24 - 34 24 - 34	

Table 6-22. 6x86 FPU Instruction Set Summary (Continued)

FPU INSTRUCTION	OP CODE	OPERATION	CLOCK COUNT	NOTES
FDIVRP Floating Point Divide Reversed, Pop	DE [1111 0 n]	ST(n) ←← TOS / ST(n); then pop TOS	24 - 34	
FDIVF Floating Point Integer Divide	DA [mod 110 r/m]	TOS ←← TOS / M.SI	34 - 38	
32-bit Integer	DE [mod 110 r/m]	TOS ←← TOS / M.WI	33 - 38	
16-bit Integer				
FIDVFR Floating Point Integer Divide Reversed	DA [mod 111 r/m]	TOS ←← M.SI / TOS	34 - 38	
32-bit Integer	DE [mod 111 r/m]	TOS ←← M.WI / TOS	33 - 38	
16-bit Integer				
FFREE Free Floating Point Register	DD [1100 0 n]	TAG(n) ←← Empty	3	
FINCSTP Increment Stack Pointer	D9 F7	Increment top of stack pointer	2	
FINIT Initialize FPU	(9B)DB E3	Wait then initialize	8	
FINITI Initialize FPU	DB E3	Initialize	6	
FLD Load Data to FPU Reg.				
Top of Stack	D9 [1100 0 n]	Push ST(n) onto stack	2	
64-bit Real	DD [mod 000 r/m]	Push M.DR onto stack	2	
32-bit Real	D9 [mod 000 r/m]	Push M.SR onto stack	2	
FBLD Load Packed BCD Data to FPU Reg.	DF [mod 100 r/m]	Push M.BCD onto stack	41 - 45	
FILD Load Integer Data to FPU Reg.				
64-bit Integer	DF [mod 101 r/m]	Push M.LI onto stack	4 - 8	
32-bit Integer	DB [mod 000 r/m]	Push M.SI onto stack	4 - 6	
16-bit Integer	DF [mod 000 r/m]	Push M.WI onto stack	3 - 6	
FLDI Load Floating Const.= 1.0	D9 E8	Push 1.0 onto stack	4	
FLDCW Load FPU Mode Control Register	D9 [mod 101 r/m]	Ctl Word ←← Memory	4	
FLDENY Load FPU Environment	D9 [mod 100 r/m]	Env Regs ←← Memory	30	
FLDL2E Load Floating Const.= Log ₂ (e)	D9 EA	Push Log ₂ (e) onto stack	4	
FLDL2T Load Floating Const.= Log ₂ (10)	D9 E9	Push Log ₂ (10) onto stack	4	
FLDLG2 Load Floating Const.= Log ₁₀ (2)	D9 EC	Push Log ₁₀ (2) onto stack	4	
FLDLN2 Load Floating Const.= Ln(2)	D9 ED	Push Log _e (2) onto stack	4	
FLDP1 Load Floating Const.= π	D9 EB	Push π onto stack	4	
FLDPZ Load Floating Const.= 0.0	D9 EE	Push 0.0 onto stack	4	
FMUL Floating Point Multiply				
Top of Stack	DC [1100 1 n]	ST(n) ←← ST(n) × TOS	4 - 9	
80-bit Register	D8 [1100 1 n]	TOS ←← TOS × ST(n)	4 - 9	
64-bit Real	DC [mod 001 r/m]	TOS ←← TOS × M.DR	4 - 8	
32-bit Real	D8 [mod 001 r/m]	TOS ←← TOS × M.SR	4 - 6	
FIMULP Floating Point Multiply & Pop	DE [1100 1 n]	ST(n) ←← ST(n) × TOS; then pop TOS	4 - 9	
FIMULF Floating Point Integer Multiply				
32-bit Integer	DA [mod 001 r/m]	TOS ←← TOS × M.SI	9 - 11	
16-bit Integer	DE [mod 001 r/m]	TOS ←← TOS × M.WI	8 - 10	
FNOP No Operation	D9 D0	No Operation	2	



FPU Clock Counts

Table 6-22. 6x86 FPU Instruction Set Summary (Continued)

FPU INSTRUCTION	OP CODE	OPERATION	CLOCK COUNT	NOTES
FPATAN Function Eval: $Tan^{-1}(y/x)$	D9 F3	ST(0) ← ATAN(ST(0)/ST(1)); then pop TOS	97 - 161	See Note 3
FPREM Floating Point Remainder	D9 F8	TOS ← Rem[TOS/ST(1)]	82 - 91	
FPREMI Floating Point Remainder IEEE	D9 F5	TOS ← Rem[TOS/ST(1)]	82 - 91	
FPTAN Function Eval: $Tan(x)$	D9 F2	TOS ← TAN(TOS); then push 1.0 onto stack	117 - 129	See Note 1
FRNDINT Round to Integer	D9 FC	TOS ← Round(TOS)	10 - 20	
FRSTOR Load FPU Environment and Reg.	DD [mod 100 r/m]	Restore state.	56 - 72	
FSAVE Save FPU Environment and Reg.	(9B)DD[mod 110 r/m]	Wait then save state.	57 - 67	
FNSAVE Save FPU Environment and Reg.	DD [mod 110 r/m]	Save state.	55 - 65	
FSCALE Floating Multiply by 2^n	D9 FD	TOS ← TOS × 2 ^{ST(0)}	7 - 14	See Note 1
FSIN Function Evaluation: $Sin(x)$	D9 FE	TOS ← SIN(TOS)	76 - 140	See Note 1
FSINCOS Function Eval.: $Sin(x) & Cos(x)$	D9 FB	temp ← TOS; TOS ← SIN(temp); then push COS(temp) onto stack	145 - 161	See Note 1
FSQRT Floating Point Square Root	D9 FA	TOS ← Square Root of TOS	59 - 60	
FST Store FPU Register				
Top of Stack	DD [1101 0 n]	ST(n) ← TOS	2	
80-bit Real	DB [mod 111 r/m]	M.XR ← TOS	2	
64-bit Real	DD [mod 010 r/m]	M.DR ← TOS	2	
32-bit Real	D9 [mod 010 r/m]	M.SR ← TOS	2	
FSTP Store FPU Register, Pop				
Top of Stack	DB [1101 1 n]	ST(n) ← TOS; then pop TOS	2	
80-bit Real	DB [mod 111 r/m]	M.XR ← TOS; then pop TOS	2	
64-bit Real	DD [mod 011 r/m]	M.DR ← TOS; then pop TOS	2	
32-bit Real	D9 [mod 011 r/m]	M.SR ← TOS; then pop TOS	2	
FBSTP Store BCD Data, Pop	DF [mod 110 r/m]	M.BCD ← TOS; then pop TOS	57 - 63	
FIST Store Integer FPU Register				
32-bit Integer	DB [mod 010 r/m]	M.SI ← TOS	8 - 13	
16-bit Integer	DF [mod 010 r/m]	M.WI ← TOS	7 - 10	
FISTP Store Integer FPU Register, Pop				
64-bit Integer	DF [mod 111 r/m]	M.LI ← TOS; then pop TOS	10 - 13	
32-bit Integer	DB [mod 011 r/m]	M.SI ← TOS; then pop TOS	8 - 13	
16-bit Integer	DF [mod 011 r/m]	M.WI ← TOS; then pop TOS	7 - 10	
FSTCW Store FPU Mode Control Register	(9B)D9[mod 111 r/m]	Wait Memory ← Control Mode Register	5	
FNSTCW Store FPU Mode Control Register	D9 [mod 111 r/m]	Memory ← Control Mode Register	3	
FSTENV Store FPU Environment	(9B)D9[mod 110 r/m]	Wait Memory ← Env. Registers	14 - 24	
FNSTENV Store FPU Environment	D9 [mod 110 r/m]	Memory ← Env. Registers	12 - 22	
FSTSW Store FPU Status Register	(9B)DD[mod 111 r/m]	Wait Memory ← Status Register	6	
FNSTSW Store FPU Status Register	DD [mod 111 r/m]	Memory ← Status Register	4	
FSTSW AX Store FPU Status Register to AX	(9B)DF E0	Wait AX ← Status Register	4	
FNSTSW AX Store FPU Status Register to AX	DF E0	AX ← Status Register	2	

Table 6-22. 6x86 FPU Instruction Set Summary (Continued)

FPU INSTRUCTION	OP CODE	OPERATION	CLOCK COUNT	NOTES
FSUB <i>Floating Point Subtract</i> Top of Stack	DC [1110 1 n]	ST(n) ← ST(n) - TOS	4 - 9	
80-bit Register	D8 [1110 0 n]	TOS ← TOS - ST(n)	4 - 9	
64-bit Real	DC [mod 100 r/m]	TOS ← TOS - M.DR	4 - 9	
32-bit Real	D8 [mod 100 r/m]	TOS ← TOS - M.SR	4 - 9	
FSDBP <i>Floating Point Subtract, Pop</i>	DE [1110 1 n]	ST(n) ← ST(n) - TOS; then pop TOS	4 - 9	
FSUBR <i>Floating Point Subtract Reverse</i> Top of Stack	DC [1110 0 n]	TOS ← ST(n) - TOS	4 - 9	
80-bit Register	D8 [1110 1 n]	ST(n) ← TOS - ST(n)	4 - 9	
64-bit Real	DC [mod 101 r/m]	TOS ← M.DR - TOS	4 - 9	
32-bit Real	D8 [mod 101 r/m]	TOS ← M.SR - TOS	4 - 9	
FSUBRP <i>Floating Point Subtract Reverse, Pop</i>	DE [1110 0 n]	ST(n) ← TOS - ST(n); then pop TOS	4 - 9	
FISB <i>Floating Point Integer Subtract</i> 32-bit Integer	DA [mod 100 r/m]	TOS ← TOS - M.SI	14 - 29	
16-bit Integer	DE [mod 100 r/m]	TOS ← TOS - M.WI	14 - 27	
FISBR <i>Floating Point Integer Subtract Reverse</i> 32-bit Integer Reversed	DA [mod 101 r/m]	TOS ← M.SI - TOS	14 - 29	
16-bit Integer Reversed	DE [mod 101 r/m]	TOS ← M.WI - TOS	14 - 27	
FIST <i>Test Top of Stack</i>	D9 E4	CC set by TOS - 0.0	4	
FUCOM <i>Unordered Compare</i>	DD [1110 0 n]	CC set by TOS - ST(n)	4	
FUCOMP <i>Unordered Compare, Pop</i>	DD [1110 1 n]	CC set by TOS - ST(n); then pop TOS	4	
FUCOMPP <i>Unordered Compare, Pop two elements</i>	DA E9	CC set by TOS - ST(0); then pop TOS and ST(1)	4	
FWAIT <i>Wait</i>	9B	Wait for FPU not busy	2	
FXAM <i>Report Class of Operand</i>	D9 E5	CC ← Class of TOS	4	
FXCH <i>Exchange Register with TOS</i>	D9 [1110 1 n]	TOS ↔ ST(n) Exchange	3	
FXTRACT <i>Extract Exponent</i>	D9 F4	temp ← TOS; TOS ← exponent (temp); then push significant (temp) onto stack	11 - 16	
FLY2X <i>Function Eval. $y \times \text{Log}_2(x)$</i>	D9 F1	ST(1) ← ST(1) × Log ₂ (TOS); then pop TOS	145 - 154	
FLY2XP1 <i>Function Eval. $y \times \text{Log}_2(x+1)$</i>	D9 F9	ST(1) ← ST(1) × Log ₂ (1+TOS); then pop TOS	131 - 133	See Note 4



FPU Instruction Summary Notes

All references to TOS and ST(n) refer to stack layout prior to execution.

Values popped off the stack are discarded.

A pop from the stack increments the top of stack pointer.

A push to the stack decrements the top of stack pointer.

Note 1:

For FCOS, FSIN, FSINCOS and FPTAN, time shown is for absolute value of TOS < $3\pi/4$. Add 90 clock counts for argument reduction if outside this range.

For FCOS, clock count is 141 if TOS < $\pi/4$ and clock count is 92 if $\pi/4 < \text{TOS} < \pi/2$.

For FSIN, clock count is 81 to 82 if absolute value of TOS < $\pi/4$.

Note 2:

For F2XMI, clock count is 92 if absolute value of TOS < 0.5.

Note 3:

For FPATAN, clock count is 97 if ST(1)/TOS < $\pi/32$.

Note 4:

For FYL2XP1, clock count is 170 if TOS is out of range and regular FYL2X is called.

Note 5:

The following opcodes are reserved by Cyrix: D9D7, D9E2, D9E7, D9FC, DED8, DEDA, DEDC, DEDD, DEDE, DFEC. If a reserved opcode is executed, and unpredictable results may occur (exceptions are not generated).