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

Table 6-22. **6x86 FPU Instruction Set Summary**

FPU INSTRUCTION	OP CODE	OPERATION	CLOCK COUNT	NOTES
F2XM1 *Function Evaluation 2^X-1*	D9 F0	TOS ⟵ 2^{TOS}-1	92 - 108	See Note 2
FABS *Floating Absolute Value*	D9 E1	TOS ⟵ | TOS |	2	
FADD *Floating Point Add*				
Top of Stack	DC [1100 0 n]	ST(n) ⟵ ST(n) + TOS	4 - 9	
80-bit Register	D8 [1100 0 n]	TOS ⟵ TOS + ST(n)	4 - 9	
64-bit Real	DC [mod 000 r/m]	TOS ⟵ TOS + M.DR	4 - 9	
32-bit Real	D8 [mod 000 r/m]	TOS ⟵ TOS + M.SR	4 - 9	
FADDP *Floating Point Add, Pop*	DE [1100 0 n]	ST(n) ⟵ ST(n) + TOS; then pop TOS	4 - 9	
FIADD *Floating Point Integer Add*				
32-bit integer	DA [mod 000 r/m]	TOS ⟵ TOS + M.SI	8 - 14	
16-bit integer	DE [mod 000 r/m]	TOS ⟵ TOS + M.WI	8 - 14	
FCHS *Floating Change Sign*	D9 E0	TOS ⟵ - TOS	2	
FCLEX *Clear Exceptions*	(9B)DB E2	Wait then Clear Exceptions	5	
FNCLEX *Clear Exceptions*	DB E2	Clear Exceptions	3	
FCOM *Floating Point Compare*				
80-bit Register	D8 [1101 0 n]	CC set by TOS - ST(n)	4	
64-bit Real	DC [mod 010 r/m]	CC set by TOS - M.DR	4	
32-bit Real	D8 [mod 010 r/m]	CC set by TOS - M.SR	4	
FCOMP *Floating Point Compare, Pop*				
80-bit Register	D8 [1101 1 n]	CC set by TOS - ST(n); then pop TOS	4	
64-bit Real	DC [mod 011 r/m]	CC set by TOS - M.DR; then pop TOS	4	
32-bit Real	D8 [mod 011 r/m]	CC set by TOS - M.SR; then pop TOS	4	
FCOMPP *Floating Point Compare, Pop Two Stack Elements*	DE D9	CC set by TOS - ST(1); then pop TOS and ST(1)	4	
FICOM *Floating Point Compare*				
32-bit integer	DA [mod 010 r/m]	CC set by TOS - M.WI	9 - 10	
16-bit integer	DE [mod 010 r/m]	CC set by TOS - M.SI	9 - 10	
FICOMP *Floating Point Compare*				
32-bit integer	DA [mod 011 r/m]	CC set by TOS - M.WI; then pop TOS	9 - 10	
16-bit integer	DE [mod 011 r/m]	CC set by TOS - M.SI; then pop TOS	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	DC [1111 1 n]	ST(n) ⟵ ST(n) / TOS	24 - 34	
80-bit Register	D8 [1111 0 n]	TOS ⟵ TOS / ST(n)	24 - 34	
64-bit Real	DC [mod 110 r/m]	TOS ⟵ TOS / M.DR	24 - 34	
32-bit Real	D8 [mod 110 r/m]	TOS ⟵ TOS / M.SR	24 - 34	
FDIVP *Floating Point Divide, Pop*	DE [1111 1 n]	ST(n) ⟵ ST(n) / TOS; then pop TOS	24 - 34	
FDIVR *Floating Point Divide Reversed*				
Top of Stack	DC [1111 0 n]	TOS ⟵ ST(n) / TOS	24 - 34	
80-bit Register	D8 [1111 1 n]	ST(n) ⟵ TOS / ST(n)	24 - 34	
64-bit Real	DC [mod 111 r/m]	TOS ⟵ M.DR / TOS	24 - 34	
32-bit Real	D8 [mod 111 r/m]	TOS ⟵ M.SR / TOS	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	
FIDIV *Floating Point Integer Divide*				
32-bit Integer	DA [mod 110 r/m]	TOS ⟵ TOS / M.SI	34 - 38	
16-bit Integer	DE [mod 110 r/m]	TOS ⟵ TOS / M.WI	33 - 38	
FIDIVR *Floating Point Integer Divide Reversed*				
32-bit Integer	DA [mod 111 r/m]	TOS ⟵ M.SI / TOS	34 - 38	
16-bit Integer	DE [mod 111 r/m]	TOS ⟵ M.WI / TOS	33 - 38	
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	
FNINIT *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	
FLD1 *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	
FLDENV *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	
FLDPI *Load Floating Const.= π*	D9 EB	Push π onto stack	4	
FLDZ *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	
FMULP *Floating Point Multiply* & Pop	DE [1100 1 n]	ST(n) ⟵ ST(n) × TOS; then pop TOS	4 - 9	
FIMUL *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	

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(1) ⟵ ATAN[ST(1) / TOS]; then pop TOS	97 - 161	See Note 3
FPREM *Floating Point Remainder*	D9 F8	TOS ⟵ Rem[TOS / ST(1)]	82 - 91	
FPREM1 *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$^{(ST(1))}$	7 - 14	
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 *Floating Point Subtract*				
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	
FSUBP *Floating Point Subtract, Pop*	DE [1110 1 n]	ST(n) ⟵ ST(n) - TOS; then pop TOS	4 - 9	
FSUBR *Floating Point Subtract Reverse*				
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 *Floating Point Subtract Reverse, Pop*	DE [1110 0 n]	ST(n) ⟵ TOS - ST(n); then pop TOS	4 - 9	
FISUB *Floating Point Integer Subtract*				
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	
FISUBR *Floating Point Integer Subtract Reverse*				
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	
FTST *Test Top of Stack*	D9 E4	CC set by TOS - 0.0	4	
FUCOM *Unordered Compare*	DD [1110 0 n]	CC set by TOS - ST(n)	4	
FUCOMP *Unordered Compare, Pop*	DD [1110 1 n]	CC set by TOS - ST(n); then pop TOS	4	
FUCOMPP *Unordered Compare, Pop two elements*	DA E9	CC set by TOS - ST(I); then pop TOS and ST(1)	4	
FWAIT *Wait*	9B	Wait for FPU not busy	2	
FXAM *Report Class of Operand*	D9 E5	CC ⟵ Class of TOS	4	
FXCH *Exchange Register with TOS*	D9 [1100 1 n]	TOS ⟷ ST(n) Exchange	3	
FXTRACT *Extract Exponent*	D9 F4	temp ⟵ TOS; TOS ⟵ exponent (temp); then push significant (temp) onto stack	11 - 16	
FLY2X *Function Eval. y × Log2(x)*	D9 F1	ST(1) ⟵ ST(1) × Log_2(TOS); then pop TOS	145 - 154	
FLY2XP1 *Function Eval. y × Log2(x+1)*	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 < 3π/4.
Add 90 clock counts for argument reduction if outside this range.

For FCOS, clock count is 141 if TOS < π/4 and clock count is 92 if π/4 < TOS > π/2.

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

Note 2:
For F2XM1, clock count is 92 if absolute value of TOS < 0.5.

Note 3:
For FPATAN, clock count is 97 if ST(1)/TOS < π/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, DDFC, DED8, DEDA, DEDC, DEDD, DEDE, DFFC.
If a reserved opcode is executed, and unpredictable results may occur (exceptions are not generated).

NOTICE TO CUSTOMERS: Some of the information contained in this document was obtained through a third party and IBM has not conducted independent tests of all product characteristics contained herein. The product described in this document is sold under IBM's standard warranty.

The information contained in this document is subject to change without notice. The products described in this document are NOT intended for use in implantation or other life support applications where malfunction may result in injury or death to persons. The information contained in this document does not effect or change IBM's product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of IBM or third parties. All the information contained in this document was obtained in specific environments, and is presented as an illustration. The results obtained in other operating environments may vary.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN "AS IS" BASIS. In no event will IBM be liable for any damages arising directly or indirectly from any use of the information contained in this document.

IBM Corporation
1000 River Street
Essex Junction, Vermont 05452-4299
United States of America