MP UNIT 6
80386DX Signals, Bus Cycles and 80387 Coprocessor

Topics

- **80386DX Signals-** Pin Diagram of 80386 and description.
- **80386DX Bus Cycles-** System Clock, Bus States, Pipelined and Non-pipelined Bus Cycles.
- **80387 NDP-**

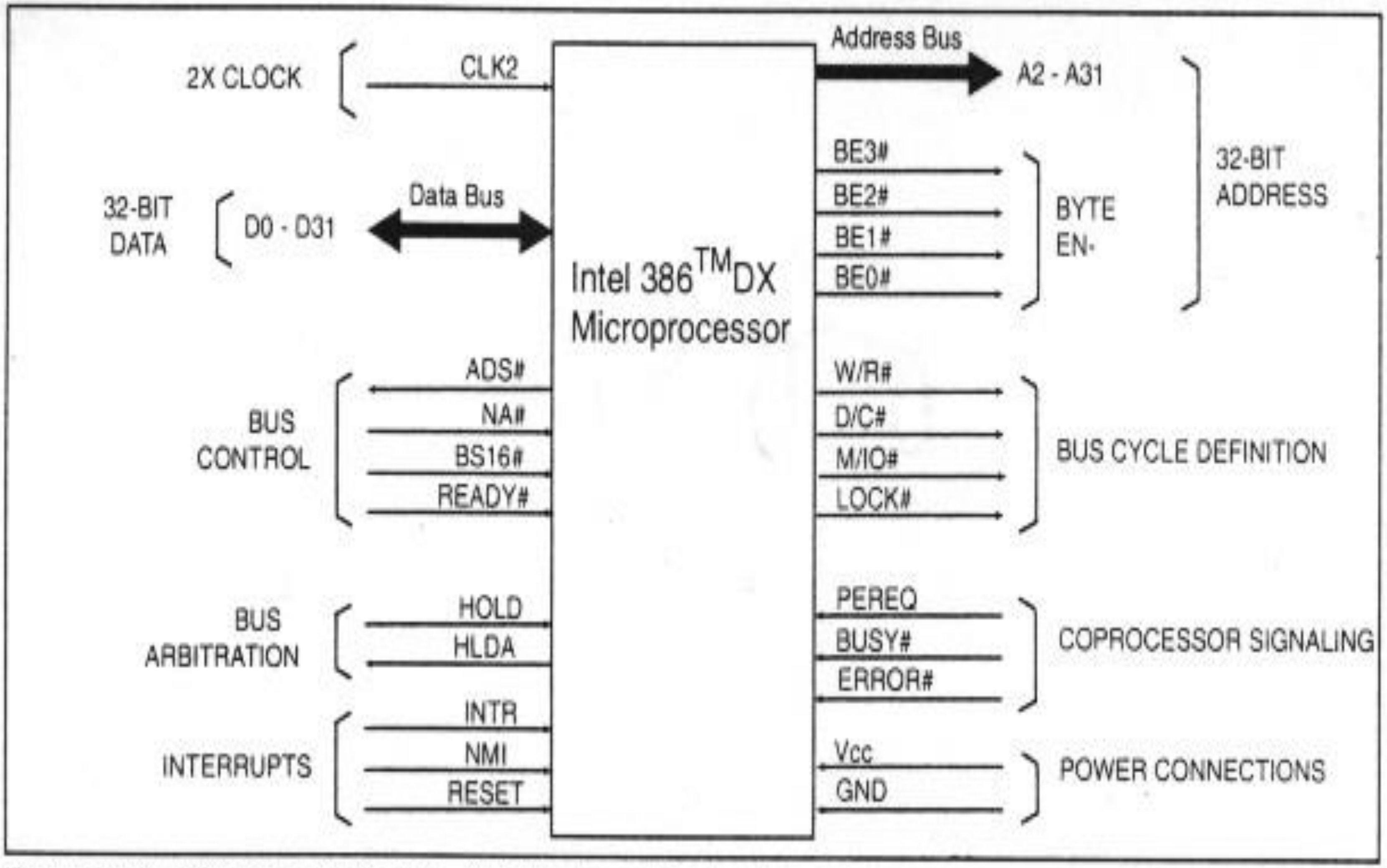


Figure 13-2. 80386 Block Diagram (# indicates active low)

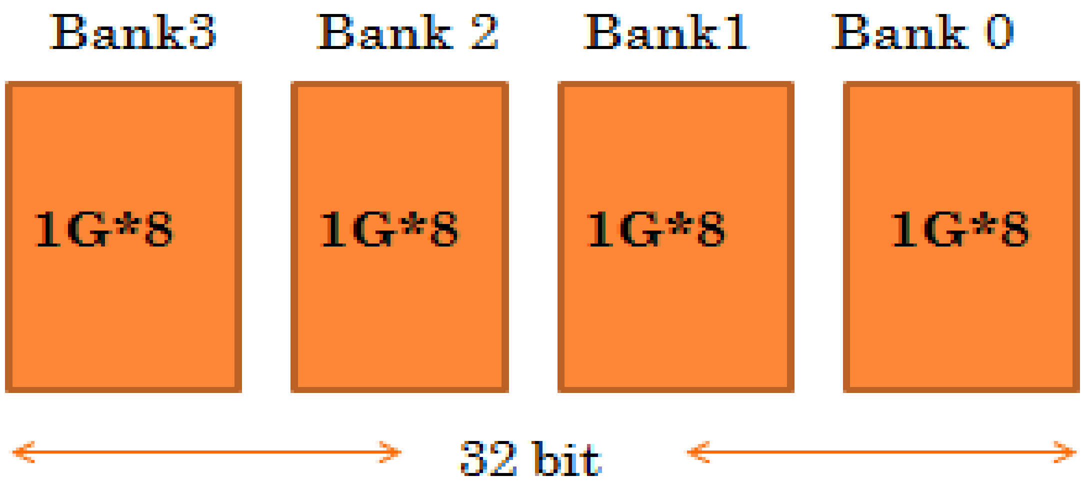
(Reprinted by permission of Intel Corporation, Copyright Intel Corp. 1992)

- **<u>CLK2:</u>** The input pin provides the basic system clock timing for the operation of 80386.

- **<u>D0 – D31:</u>** These 32 lines act as bidirectional data bus during different access cycles.

- **<u>A31 – A2:</u>** These are upper 30 bit of the 32- bit address bus.

- **<u>BE0 to BE3</u>**: (Active Low) The 32- bit data bus supported by 80386 and the memory system of 80386 can be viewed as a 4-byte wide memory access mechanism.



BE0,BE1,BE2 and BE3 are active low signals.

Bus Control

ADS#: (Address Data Strobe)

Active when **issued a valid request.**

The address status output pin indicates that the address bus and **Bus Cycle Definition Pins**(W/R#, D/C#, M/IO#, BE0# to BE3#) are carrying the respective valid signals.

BS16#: The bus size – 16 input pin **allows the interfacing of 16 bit devices with the 32 bit wide 80386 data bus.**

- **<u>READY#:</u>** The ready signals indicates to the CPU that the **previous bus cycle has been terminated** and **the bus is ready for the next cycle.** The signal is used to insert WAIT states in a bus cycle and is useful for interfacing of slow devices with CPU.

- **<u>NA#:</u>** Gives address of next instruction if pipelining is enabled. If pipelining is not enabled, this pin is high, if instruction is in waiting state.

Bus Arbitration

- **HOLD:** The Bus hold input pin enables the other bus masters to gain control of the system bus if it is asserted.

- **HLDA:** The bus hold acknowledge output indicates that a valid bus hold request has been received and the bus has been relinquished by the CPU.

Interrupts

- **INTR:** This interrupt pin is a mask-able interrupt, that can be masked using the IF of the flag register.

- **NMI:** A valid request signal at the non-mask-able interrupt request input pin internally generates a non- mask-able interrupt of type 2.

- **RESET:** A high at this input pin suspends the current operation and restart the execution from the starting location.

Co-Processor Signaling

- **BUSY#:** The busy input signal indicates to the CPU that the **coprocessor is busy** with the allocated task.

- **ERROR#:** The error input pin indicates to the CPU that the coprocessor has encountered an error while executing its instruction.

- **PEREQ: (Processor extension request)**

  Output signal indicates to the **CPU to fetch data.**

A bus cycle definition pins

- **LOCK#**: BUS LOCK is a bus cycle definition pin that indicates that system have locked system bus of other peripherals.

- **W/R#:** WRITE/READ is a bus cycle definition pin that distinguishes write cycles from read cycles.

- **D/C#:** DATA/CONTROL is a bus cycle definition pin that distinguishes data cycles, either memory or I/O, from control cycles which are: interrupt acknowledge, halt, and instruction fetching.

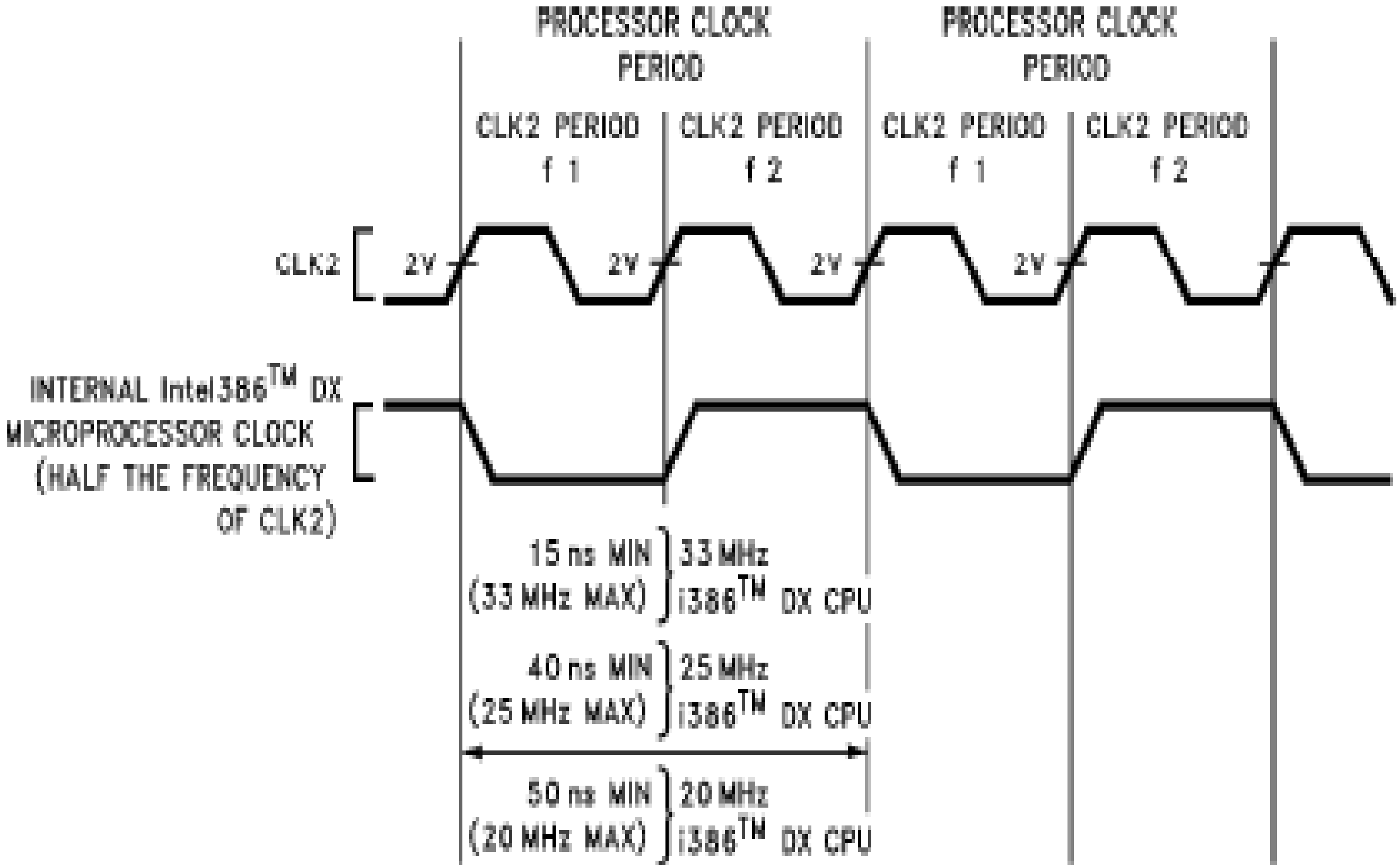
- **M/IO#:** MEMORY I/O is a bus cycle definition pin that distinguishes memory cycles from input/output cycles.

Power Connection Pins

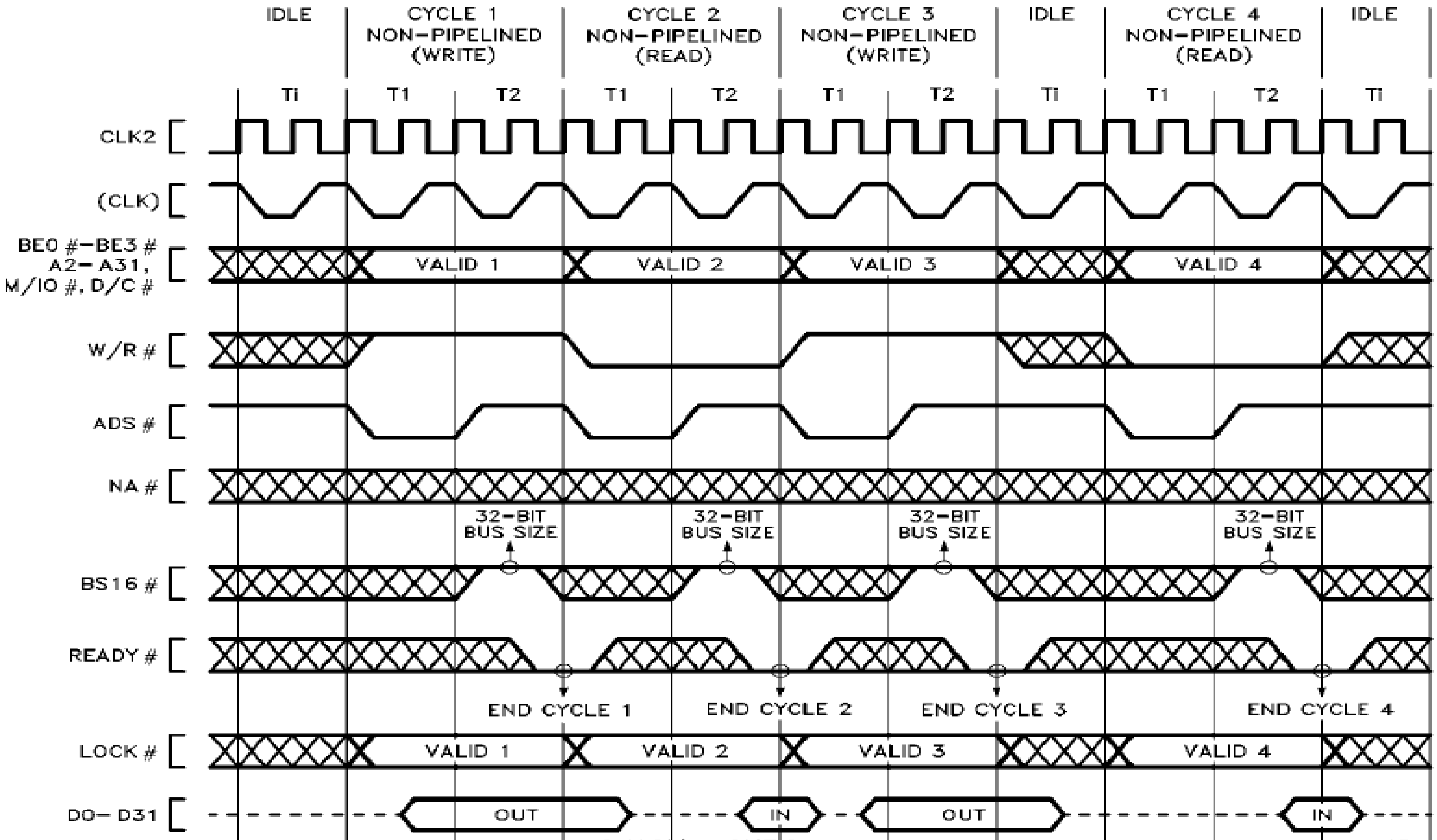
- **<u>VCC:</u>** These are system power supply lines.

- **<u>GND:</u>**

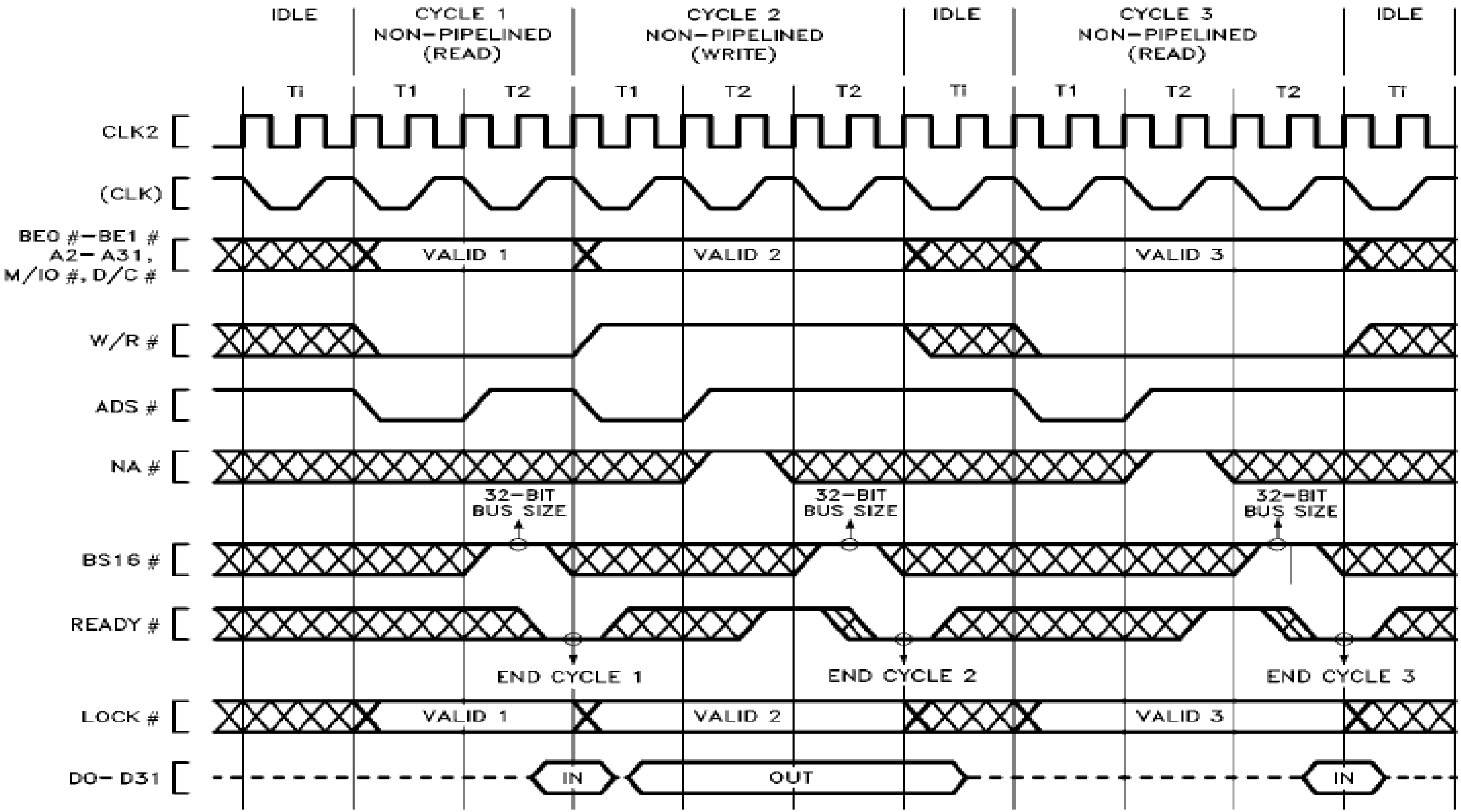
System Clock



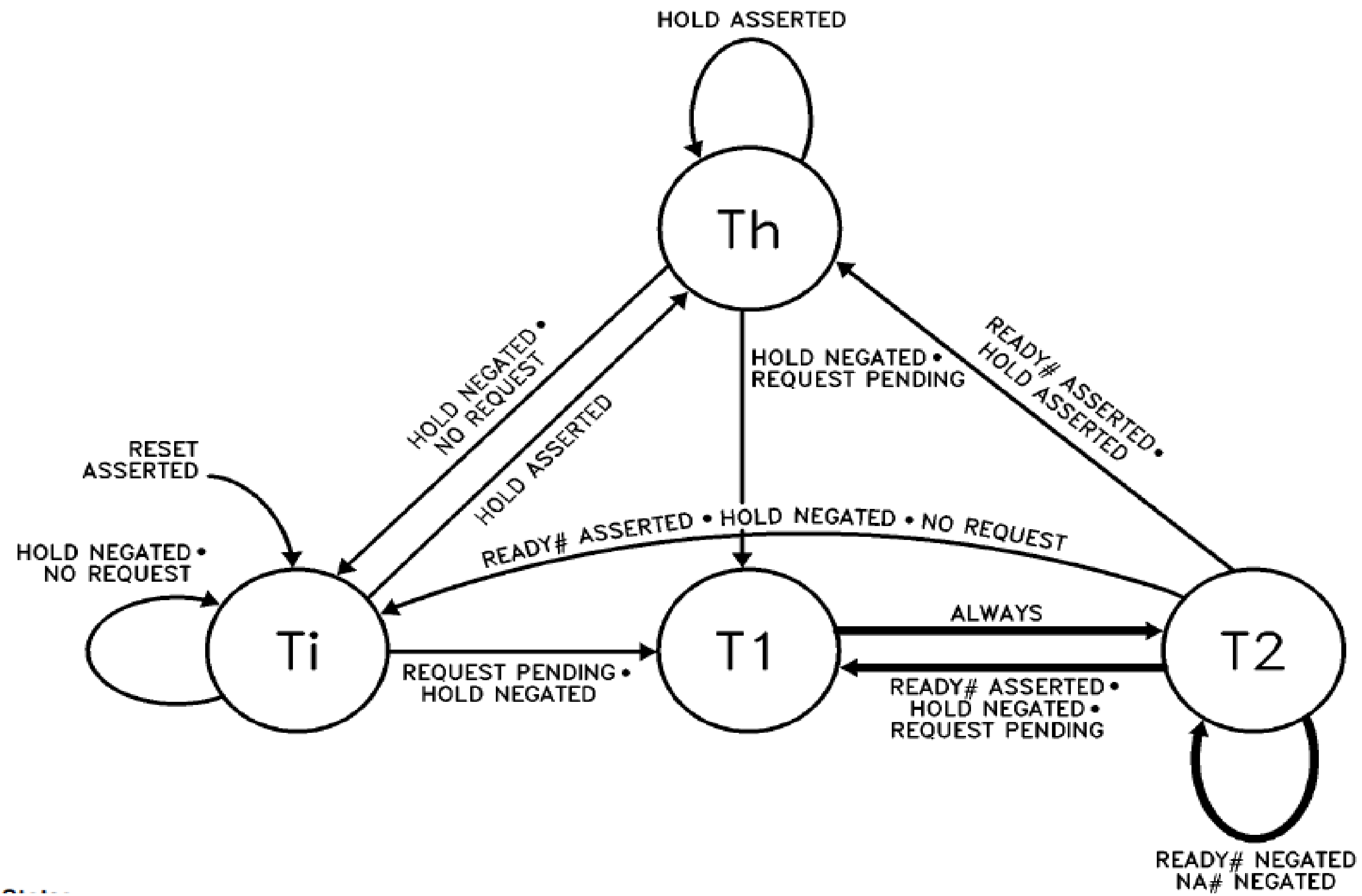
Non-pipelined read & write cycles (No wait states)



Non-pipelined read & write cycles (With wait states)



Bus States (Using non-pipelined address)



- **T1:** First clock of a non-pipelined bus cycle (Intel386 DX drives new address and asserts ADS#)

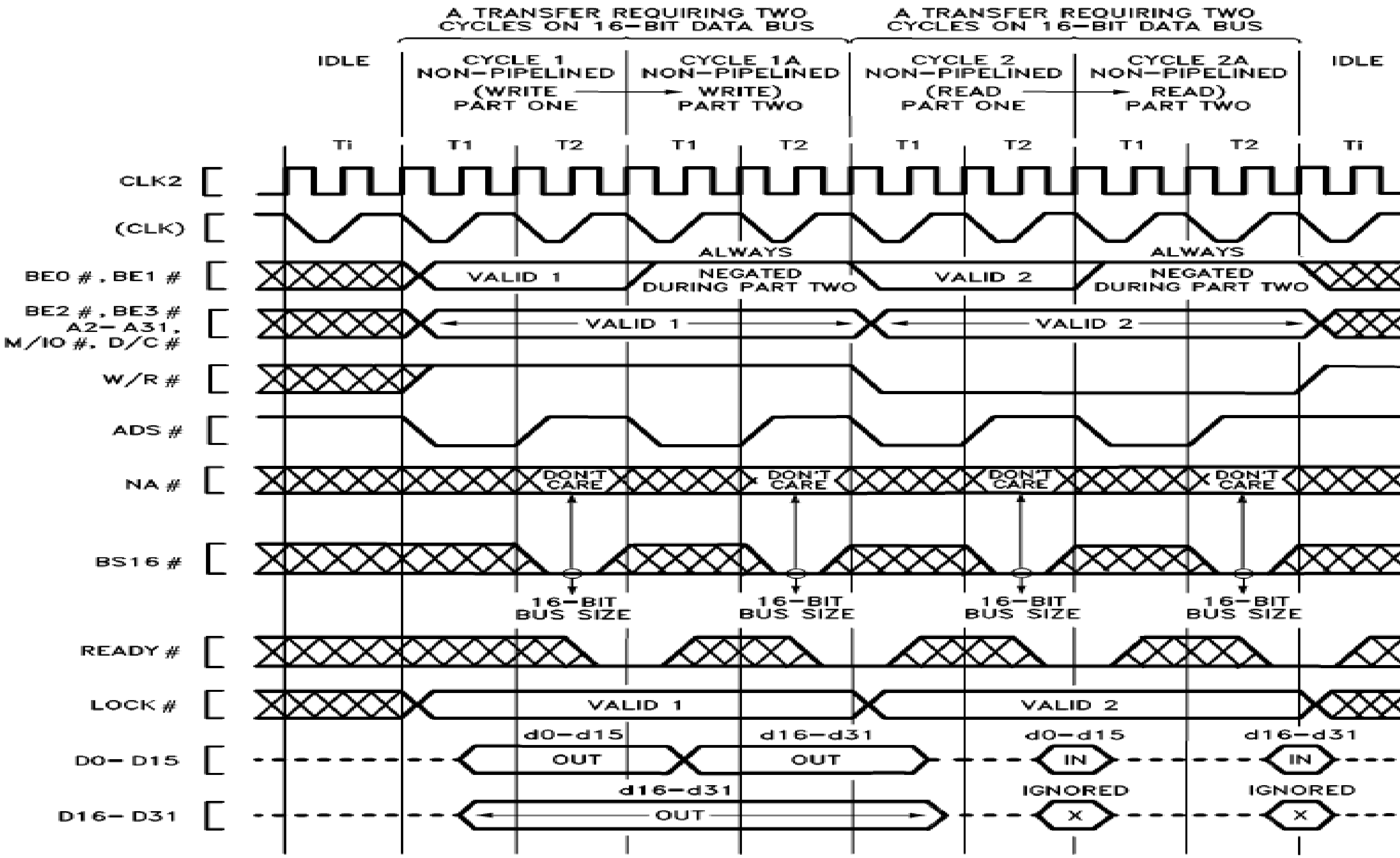
- **T2:** subsequent clocks of a bus cycle when NA# has not been sampled asserted in the current bus cycle

- **Ti:** idle state

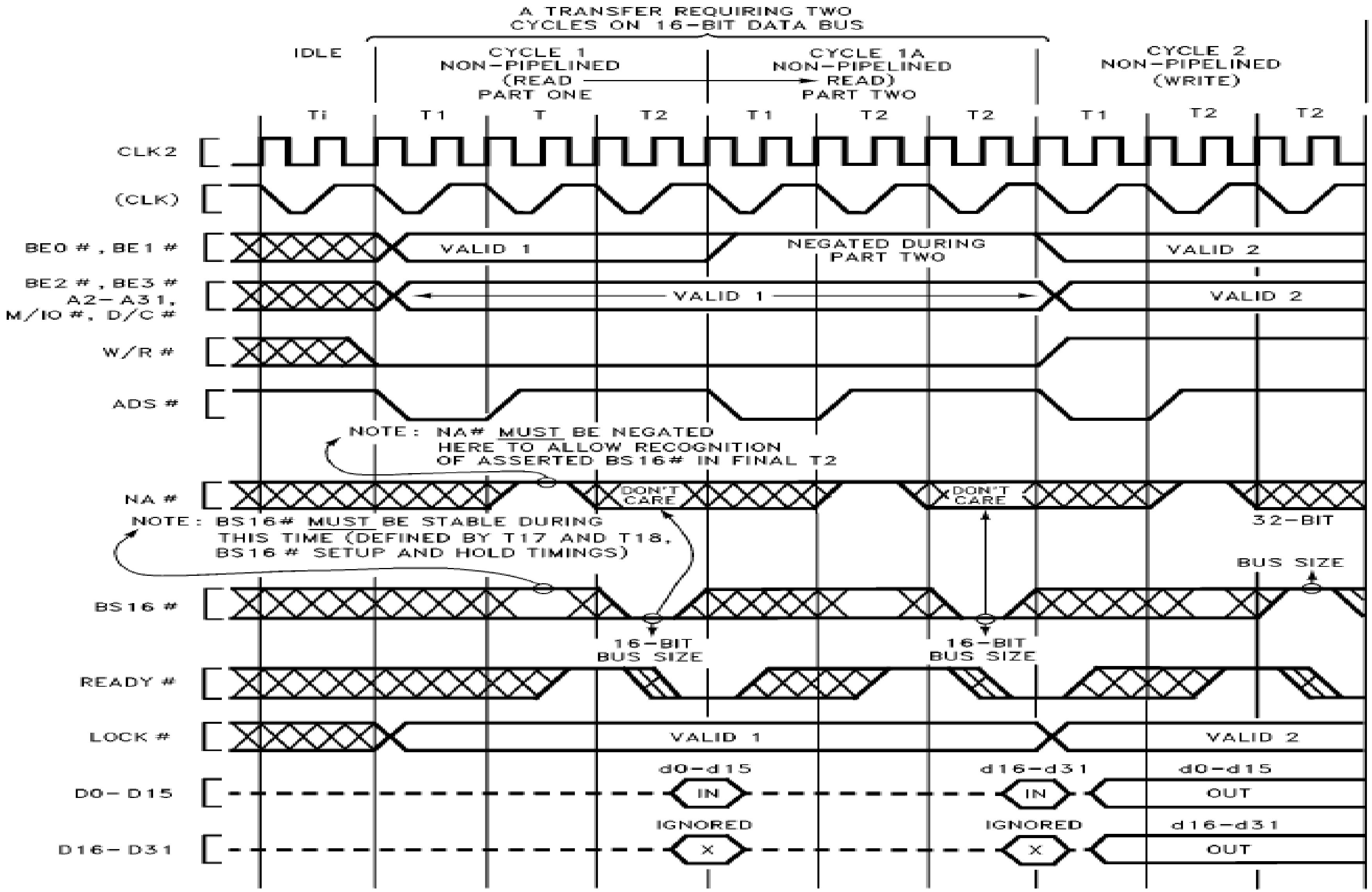
- **Th:** Hold acknowledge state (Intel386 DX asserts HLDA)

- The fastest bus cycle consists of two states: T1 and T2.

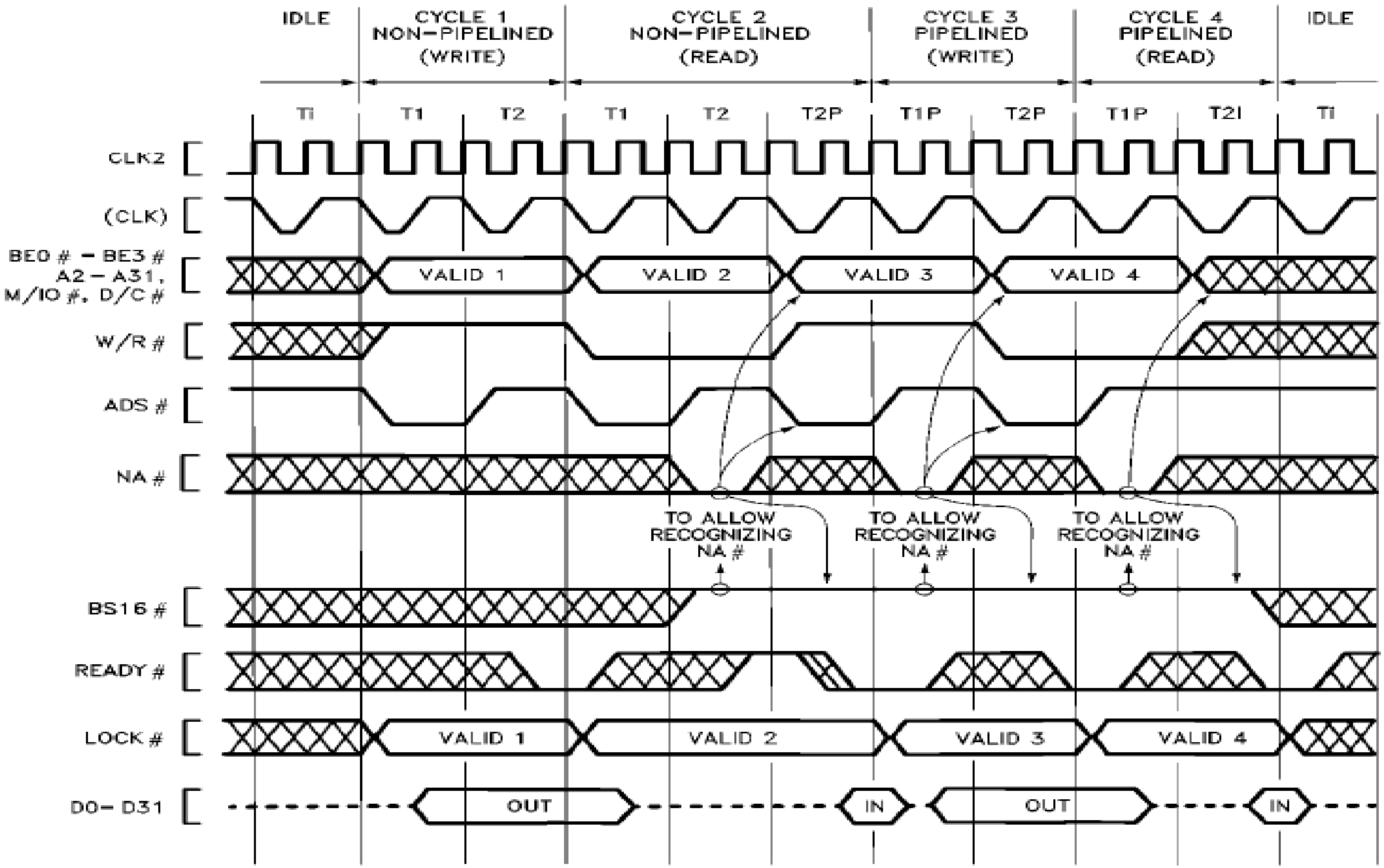
Asserting BS16#: No wait states



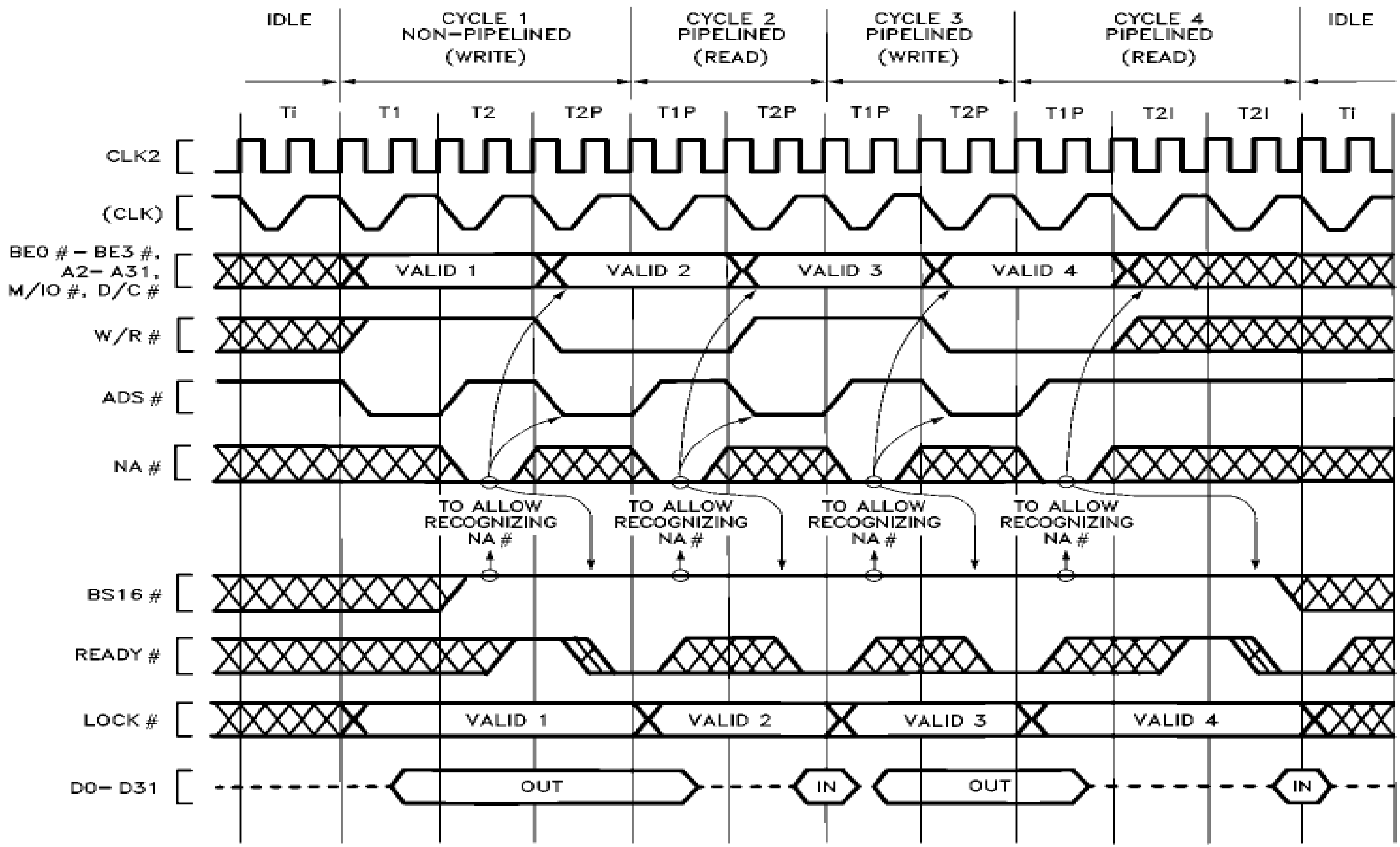
Asserting BS16#: Wait states

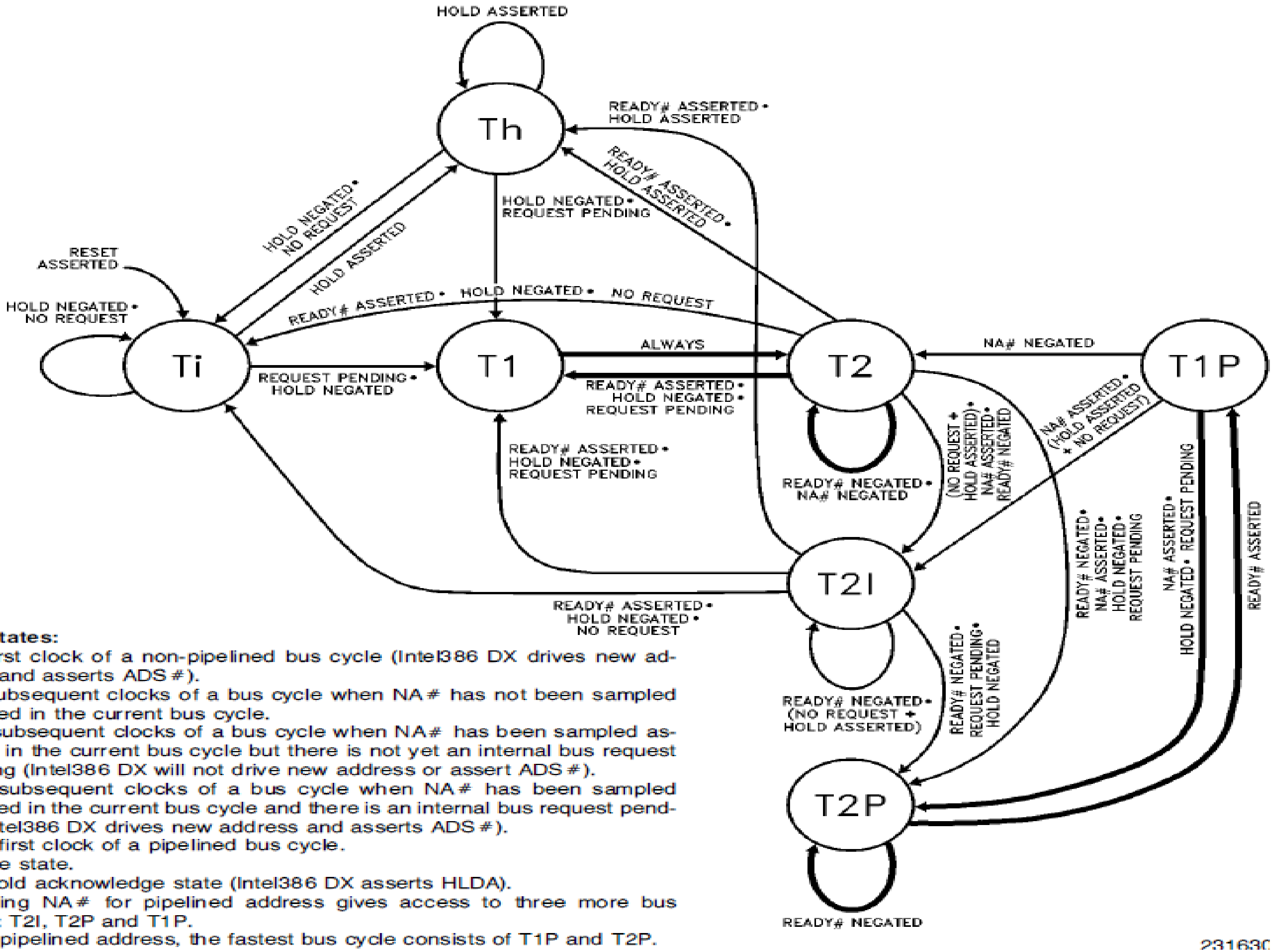


Transitioning to pipelined address



Fast Transitioning to pipelined address





Bus States:

T1—first clock of a non-pipelined bus cycle (Intel386 DX drives new address and asserts ADS#).

T2—subsequent clocks of a bus cycle when NA# has not been sampled asserted in the current bus cycle.

T2I—subsequent clocks of a bus cycle when NA# has been sampled asserted in the current bus cycle but there is not yet an internal bus request pending (Intel386 DX will not drive new address or assert ADS#).

T2P—subsequent clocks of a bus cycle when NA# has been sampled asserted in the current bus cycle and there is an internal bus request pending (Intel386 DX drives new address and asserts ADS#).

T1P—first clock of a pipelined bus cycle.

Ti—idle state.

Th—hold acknowledge state (Intel386 DX asserts HLDA).

Asserting NA# for pipelined address gives access to three more bus states: T2I, T2P and T1P.

Using pipelined address, the fastest bus cycle consists of T1P and T2P.

1. Processor Vs. Co-processor

- Difference between Processor and Co-processor

   i. **Functions and Features**
   ii. **Instruction Set**
   iii. **Register Organization**

Processor

- Processor performs operations on Integer numbers.
- Integer numbers are usually stored in Hexadecimal format with Packed BCD representation.
- Can perform Arithmetic and Logical Operations.
- Can handle data types: 16,32, and 64 bit integers.

Co-processor

- Co-Processor performs operations on both Floating Point numbers and Integer numbers.
- Floating Point numbers are represented using IEEE standards.
- Can perform upto 68 additional arithmetic, trigonometric, exponential, & logarithmic instructions.
- Can handle data types: 16,32, and 64 bit integers; 32,64, and 80 bit floating-point real numbers; and up to 18-digit (BCD) operands.

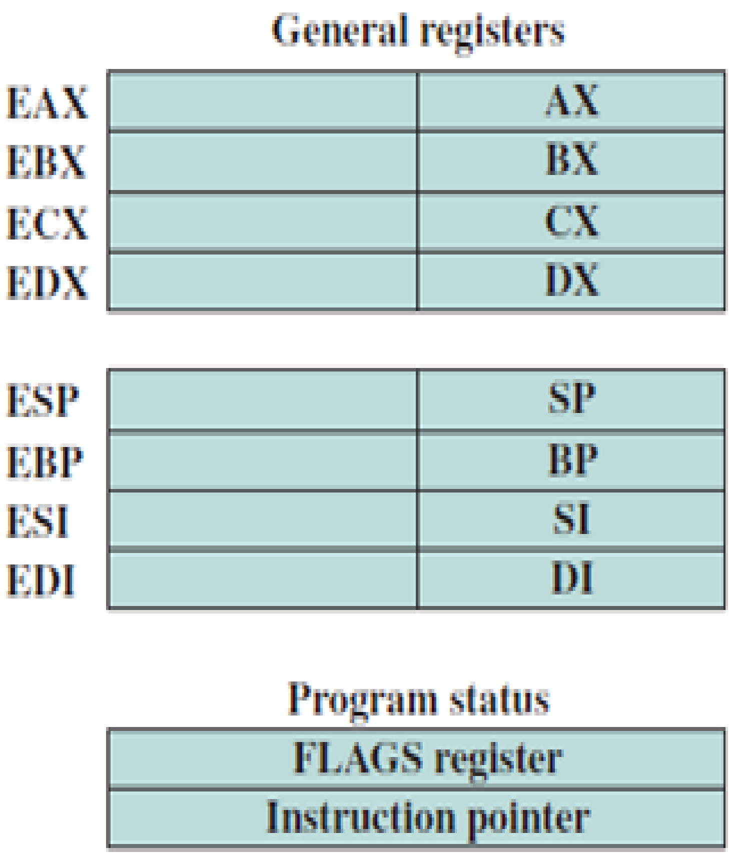
Instruction Set

- As 80386 and 80387 are completely different processors, they have different
  - **Instruction Set**
  - **Bandwidth**
  - **Clock speed**

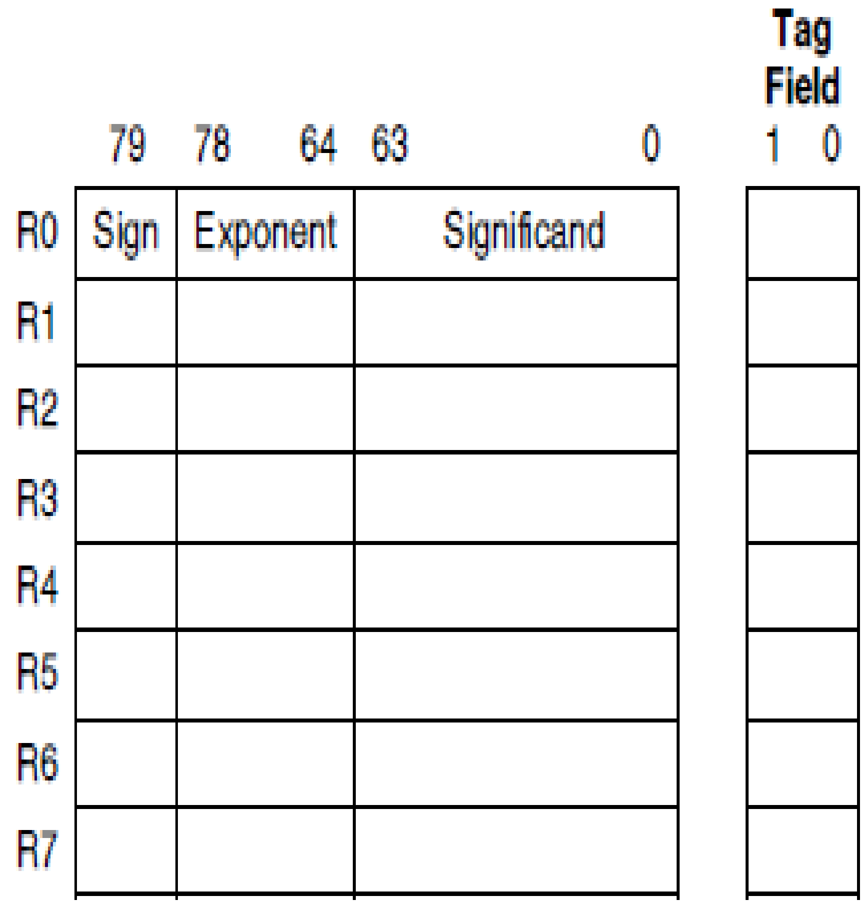
- For **processor instructions, Set of Operands are expected**, where actually operations are performed.

- But for **Coprocessor instructions, Many of instructions do not have operands**. In this case by default operations will be performed on TOP value.

80386 Register Organization



80387 Register Organization



80387: NDP

- Control Register bits for Coprocessor support
- 80387 Register Stack
- Data Types
- Load and Store Instructions
- Trigonometric and Transcendental Instructions
- Interfacing signals of 80386DX with 80387.

Features: 80387

- High performance **80-Bit Internal Architecture**.
- **Implements IEEE standard** for Binary floating-point arithmetic.
- Expands Intel386DX CPU **data types to include 32-, 64-, 80-bit floating point, 32-bit, 64-bit integers and 80-bit BCD operands.**
- Extends Intel386DX CPU **instruction set to include Trigonometric, Logarithmic, Exponential and Arithmetic instructions for all data types.**

- **Support transcendental operations** for SINE, COSINE, TANGENT, ARCTANGENT and LOGARITHM.
- Built-in **Exception handling**.
- **Operates independently in all modes** of 80386.
- **Eight 80-bit Numeric registers.**
- Available in **68-pin PGA package.**
- One version supports **16MHz-33MHz.**

Register use of 8087

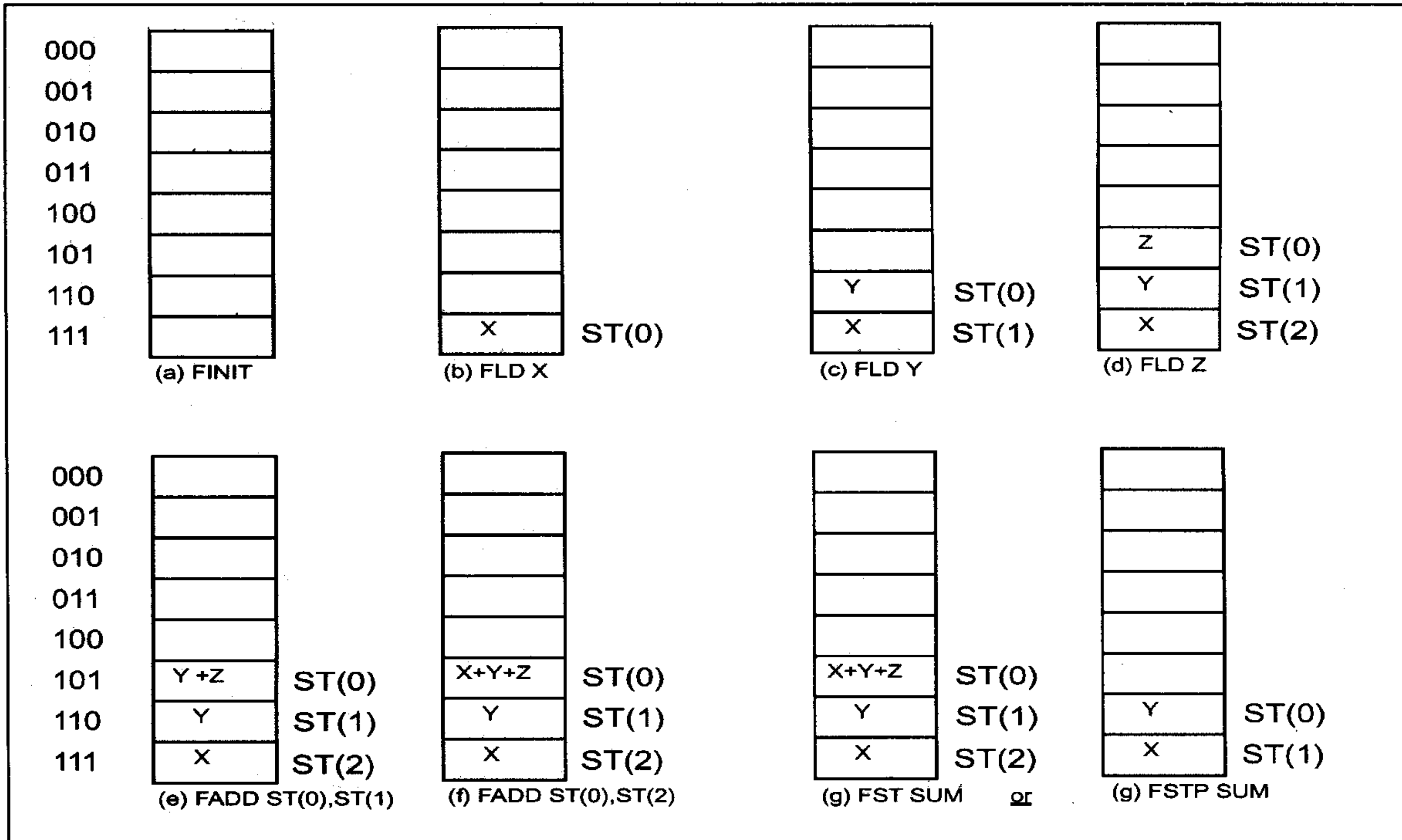


Figure 20-2. Stack Diagram for Example 20-5

Register Set

- **Data registers:** Eight 80-bit registers.

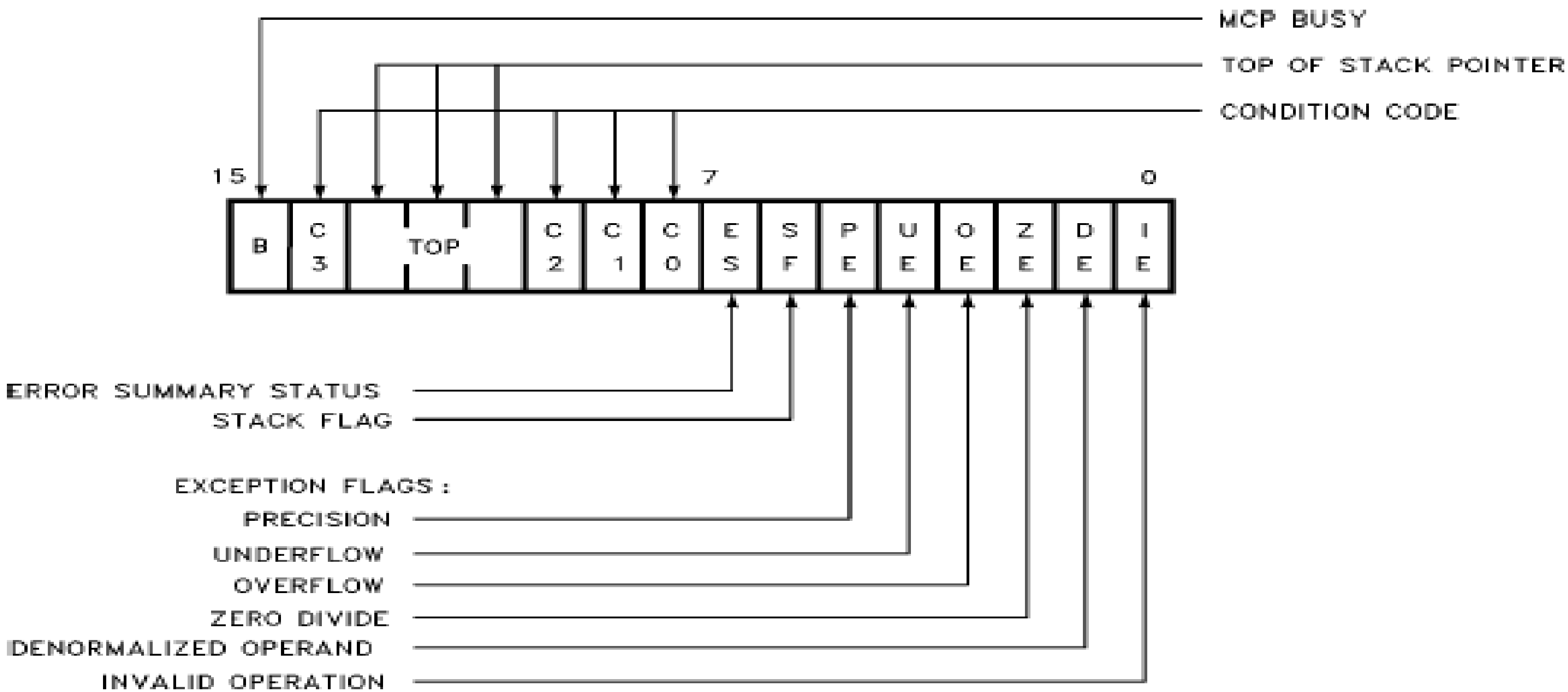
- **Tag Word:** The tag word marks the content of each numeric data register, two bits for each data register.

- **Status word:** The 16-bit status word reflects the overall state of the MCP (Math Coprocessor).

- **Instruction and Data pointers:** Two pointer registers allows identification of the failing numeric instruction which supply the address of failing numeric instruction and the address of its numeric memory operand.

- **Control Word:** Several processing options are selected by loading a control word from memory into the control register.

MCP Status Word



ES is set if any unmasked exception bit is set; cleared otherwise.
See Table 2.2 for interpretation of condition code.
TOP values:
000 = Register 0 is Top of Stack
001 = Register 1 is Top of Stack
•
•
•
111 = Register 7 is Top of Stack
For definitions of exceptions, refer to the section entitled "Exception Handling"

- **Bits 13 –11 (TOP)** point to the Intel387 DX MCP register that is the **Current Top-of-stack.**

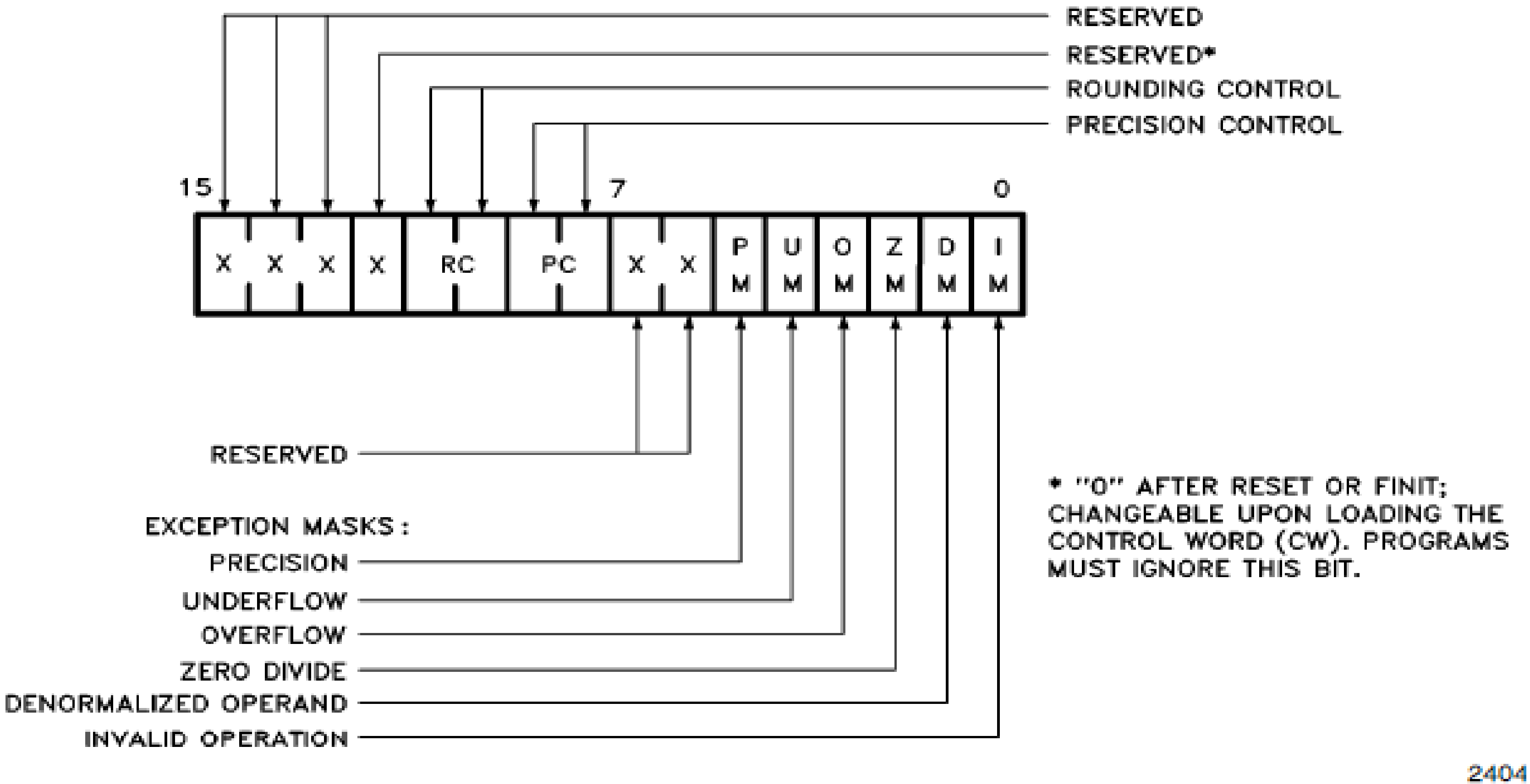
- The four **Numeric Condition Code Bits (C3–C0)** are similar to the flags in a CPU; instructions that perform arithmetic operations update these bits to reflect the outcome.

- Bit 7 is the **Error Summary (ES) Status Bit**. This bit is **set if any unmasked exception bit is set**; it is clear otherwise. If this bit is set, the ERROR# signal is asserted

- Bit 6 is the **Stack Flag (SF).** This bit is used to **distinguish invalid operations due to stack overflow or underflow** from other kinds of invalid operations. When SF is set, bit 9 (C1) distinguishes between stack overflow (C1 = 1) and underflow (C1 = 0).

- Figure shows the **six Exception Flags in bits 5 – 0** of the status word. Bits 5 –0 are set to indicate that the MCP has detected an exception while executing an instruction.

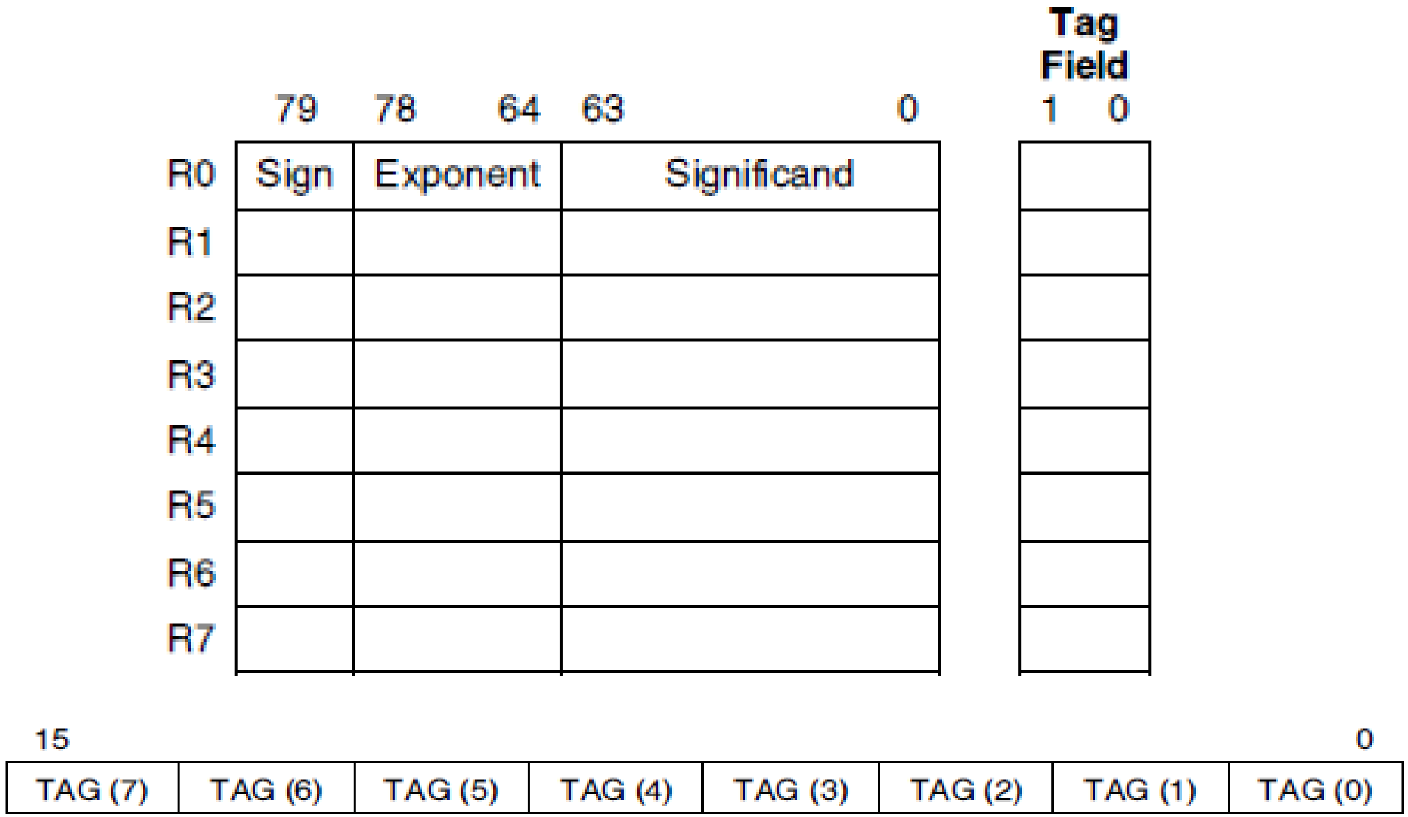
Control register bits for coprocessor support



Precision Control
- 00—24 bits (single precision)
- 01—(reserved)
- 10—53 bits (double precision)
- 11—64 bits (extended precision)

Rounding Control
- 00—Round to nearest or even
- 01—Round down (toward $-\infty$)
- 10—Round up (toward $+\infty$)
- 11—Chop (truncate toward zero)

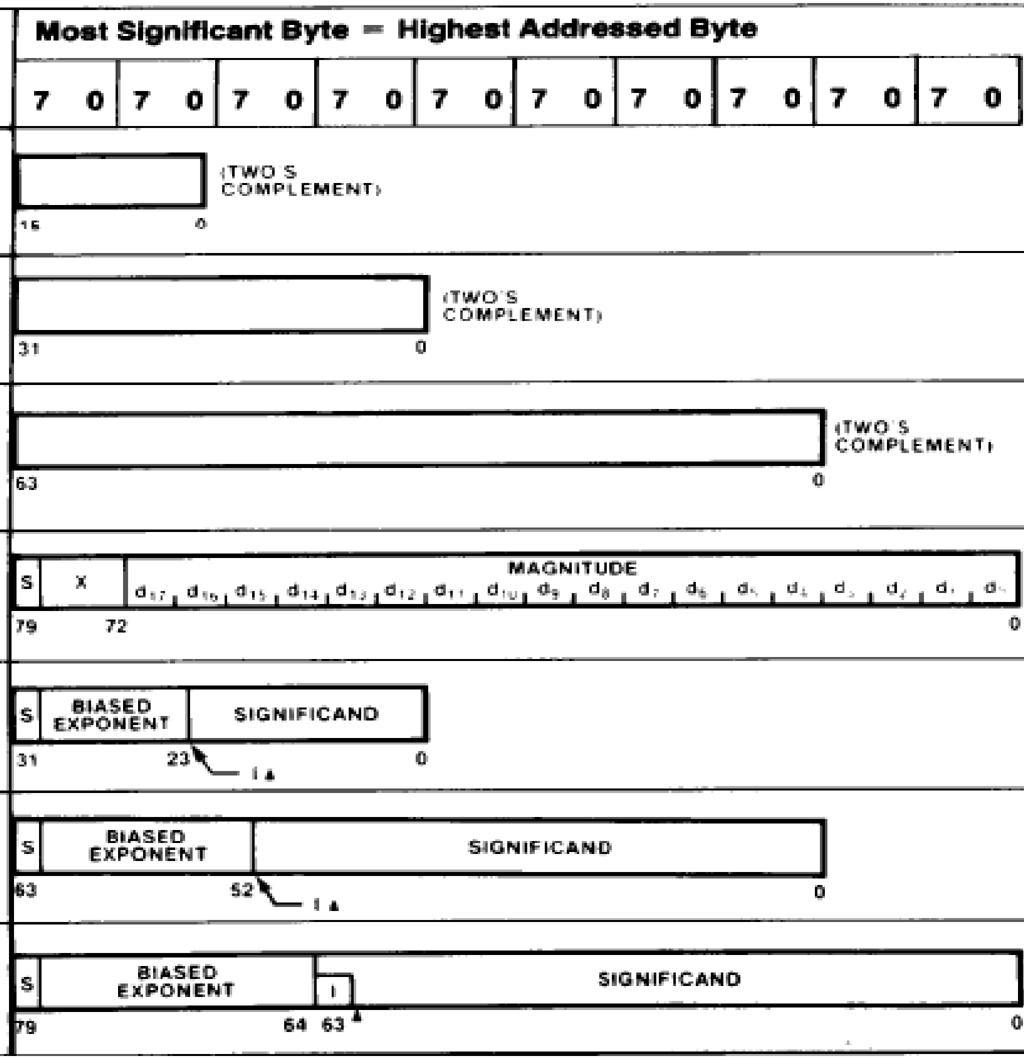
80387 Register Stack



TAG VALUES:
00 = Valid
01 = Zero
10 = QNaN, SNaN, Infinity, Denormal and Unsupported Formats
11 = Empty

Data Types

Data Formats	Range	Precision
Word Integer	$\pm 10^4$	16 Bits
Short Integer	$\pm 10^9$	32 Bits
Long Integer	$\pm 10^{18}$	64 Bits
Packed BCD	$\pm 10^{\pm 18}$	18 Digits
Single Precision	$\pm 10^{\pm 38}$	24 Bits
Double Precision	$\pm 10^{\pm 308}$	53 Bits
Extended Precision	$\pm 10^{\pm 4932}$	64 Bits



Instruction Set

1. Data transfer instructions
2. Non-transcendental instructions
3. Comparison instructions
4. Transcendental instructions
5. Constant instructions
6. Processor Control instructions

- **FINIT: (Initialize Floating Point Unit)**
  - Initialize FPU after checking for pending unmasked floating-point exceptions.
  - Syntax: FINIT (no operand)

1. Data transfer instructions

Real Transfers	
FLD FST FSTP FXCH	Load Real Store real Store real and pop Exchange registers
Integer Transfers	
FILD FIST FISTP	Integer load Integer store Integer store and pop
Packed Decimal Transfers	
FBLD FBSTP	Packed decimal (BCD) load Packed decimal (BCD) store and pop

Mnemonic	Description
FLD m32fp	Push m32fp onto the FPU register stack.
FLD m64fp	Push m64fp onto the FPU register stack.
FLD m80fp	Push m80fp onto the FPU register stack.
FLD ST(i)	Push ST(i) onto the FPU register stack.

Instruction	64-Bit Mode	Compat/Leg Mode	Description
FST *m32fp*	Valid	Valid	Copy ST(0) to *m32fp*.
FST *m64fp*	Valid	Valid	Copy ST(0) to *m64fp*.
FST ST(i)	Valid	Valid	Copy ST(0) to ST(i).
FSTP *m32fp*	Valid	Valid	Copy ST(0) to *m32fp* and pop register stack.
FSTP *m64fp*	Valid	Valid	Copy ST(0) to *m64fp* and pop register stack.
FSTP *m80fp*	Valid	Valid	Copy ST(0) to *m80fp* and pop register stack.
FSTP ST(i)	Valid	Valid	Copy ST(0) to ST(i) and pop register stack.

Instruction	Description
FBSTP m80bcd	Store ST(0) in m80bcd and pop ST(0).

- **FLD:** Push one of seven commonly used constants (in double extended-precision floating-point format) onto the FPU register stack. **It sets TOP to ST0**.

Mnemonic	Description
FLD1	Push 1 to ST0
FLDL2T	Push log_2(10) to ST0
FLDL2E	Push log_2(e) to ST0
FLDPI	Push PI to ST0
FLDLG2	Push log_10(2) to ST0
FLDLN2	Push log_e(2) to ST0
FLDZ	Push 0 to ST0

- **FBST/ FBSTP: Store BCD Integer / Pop**
  - Converts the value in the ST(0) register to an 18-digit packed BCD integer, stores the result in the destination operand, and pops the register stack (For FBSTP).

Instruction	64-Bit Mode
FBST m80bcd	Store ST(0) in m80bcd
FBSTP m80bcd	Store ST(0) in m80bcd and pop ST(0).

FBSTP Instruction

If ST0: (1234567890ABCDEF4321) in IEEE precision format.

Memory location	Data
1000000A h	
10000009 h	12
10000008 h	34
10000007 h	56
10000006 h	78
10000005 h	90
10000004 h	AB
10000003 h	CD
10000002 h	EF
10000001 h	43
10000000 h	21

2. Non-transcendental instructions

Operation	Instructions
Addition	FADD, FADDP, FIADD
Subtraction	FSUB, FSUBP, FISUB, FSUBR (Reverse Subtract), FSUBRP, FISUBR
Multiplication	FMUL, FMULP, FIMUL
Division	FDIV, FDIVP, FIDIV, FDIVR (Reverse Division), FDIVRP, FIDIVR
Other operations	FSQRT, FSCALE, FPREM (Partial Reminder), FPREM1, FRNDINT (Round to Integer), FXTRACT (Extract Exponent and Mantissa), FABS (Absolute Value), FCHS (Change Sign)

Mnemonic	Description
FSUB m32fp	Subtract m32fp from ST(0) and store result in ST(0).
FSUB m64fp	Subtract m64fp from ST(0) and store result in ST(0).
FSUB ST(0), ST(i)	Subtract ST(i) from ST(0) and store result in ST(0).
FSUB ST(i), ST(0)	Subtract ST(0) from ST(i) and store result in ST(i).
FSUBP ST(i), ST(0)	Subtract ST(0) from ST(i), store result in ST(i), and pop register stack.
FSUBP	Subtract ST(0) from ST(1), store result in ST(1), and pop register stack.
FISUB m32int	Subtract m32int from ST(0) and store result in ST(0).
FISUB m16int	Subtract m16int from ST(0) and store result in ST(0).

Mnemonic	Description
FSUBR m32fp	Subtract ST(0) from m32fp and store result in ST(0).
FSUBR m64fp	Subtract ST(0) from m64fp and store result in ST(0).
FSUBR ST(0), ST(i)	Subtract ST(0) from ST(i) and store result in ST(0).
FSUBR ST(i), ST(0)	Subtract ST(i) from ST(0) and store result in ST(i).
FSUBRP ST(i), ST(0)	Subtract ST(i) from ST(0), store result in ST(i), and pop register stack.
FSUBRP	Subtract ST(1) from ST(0), store result in ST(1), and pop register stack.
FISUBR m32int	Subtract ST(0) from m32int and store result in ST(0).
FISUBR m16int	Subtract ST(0) from m16int and store result in ST(0).

Mnemonic	Description
FMUL m32fp	Multiply ST(0) by m32fp and store result in ST(0)
FMUL m64fp	Multiply ST(0) by m64fp and store result in ST(0)
FMUL ST(0), ST(i)	Multiply ST(0) by ST(i) and store result in ST(0)
FMUL ST(i), ST(0)	Multiply ST(i) by ST(0) and store result in ST(i)
FMULP ST(i), ST(0)	Multiply ST(i) by ST(0), store result in ST(i), and pop the register stack
FMULP	Multiply ST(1) by ST(0), store result in ST(1), and pop the register stack
FIMUL m32int	Multiply ST(0) by m32int and store result in ST(0)
FIMUL m16int	Multiply ST(0) by m16int and store result in ST(0)

Mnemonic	Description
FDIV m32fp	Divide ST(0) by m32fp and store result in ST(0).
FDIV m64fp	Divide ST(0) by m64fp and store result in ST(0).
FDIV ST(0), ST(i)	Divide ST(0) by ST(i) and store result in ST(0).
FDIV ST(i), ST(0)	Divide ST(i) by ST(0) and store result in ST(i).
FDIVP ST(i), ST(0)	Divide ST(i) by ST(0), store result in ST(i), and pop the register stack.
FDIVP	Divide ST(1) by ST(0), store result in ST(1), and pop the register stack.
FIDIV m32int	Divide ST(0) by m32int and store result in ST(0).
FIDIV m16int	Divide ST(0) by m64int and store result in ST(0).

Mnemonic	Description
FDIVR m32fp	Divide m32fp by ST(0) and store result in ST(0)
FDIVR m64fp	Divide m64fp by ST(0) and store result in ST(0)
FDIVR ST(0), ST(i)	Divide ST(i) by ST(0) and store result in ST(0)
FDIVR ST(i), ST(0)	Divide ST(0) by ST(i) and store result in ST(i)
FDIVRP ST(i), ST(0)	Divide ST(0) by ST(i), store result in ST(i), and pop the register stack
FDIVRP	Divide ST(0) by ST(1), store result in ST(1), and pop the register stack
FIDIVR m32int	Divide m32int by ST(0) and store result in ST(0)
FIDIVR m16int	Divide m16int by ST(0) and store result in ST(0)

Mnemonic	Description
FSQRT	Computes square root of ST(0) and stores the result in ST(0).

Mnemonic	Description
FSCALE	Scale ST(0) by ST(1).

Mnemonic	Description
FPREM	Replace ST(0) with the remainder obtained from dividing ST(0) by ST(1).

Mnemonic	Description
FPREM1	Replace ST(0) with the IEEE remainder obtained from dividing ST(0) by ST(1).

Mnemonic	Description
FRNDINT	Round ST(0) to an integer.

Mnemonic	Description
FXTRACT	Separate value in ST(0) into exponent and mantissa, store exponent in ST(0), and push the mantissa onto the register stack.

Mnemonic	Description
FABS	Replace ST with its absolute value.

Mnemonic	Description
FCHS	Complements sign of ST(0)

3. Comparison Instructions

FCOM	Compare real
FCOMP	Compare real and pop
FCOMPP	Compare real and pop twice
FICOM	Integer compare
FICOMP	Integer compare and pop
FTST	Test
FUCOM	Unordered compare real
FUCOMP	Unordered compare real and pop
FUCOMPP	Unordered compare real and pop twice
FXAM	Examine

Table 4-6. Condition Code Resulting from Comparisons

Order	C3 (ZF)	C2 (PF)	C0 (CF)	80386 Conditional Branch
ST > Operand	0	0	0	JA
ST < Operand	0	0	1	JB
ST = Operand	1	0	0	JE
Unordered	1	1	1	JP

Mnemonic	Description
FCOM m32fp	Compare ST(0) with m32fp.
FCOM m64fp	Compare ST(0) with m64fp.
FCOM ST(i)	Compare ST(0) with ST(i).
FCOM	Compare ST(0) with ST(1).
FCOMP m32fp	Compare ST(0) with m32fp and pop register stack.
FCOMP m64fp	Compare ST(0) with m64fp and pop register stack.
FCOMP ST(i)	Compare ST(0) with ST(i) and pop register stack.
FCOMP	Compare ST(0) with ST(1) and pop register stack.
FCOMPP	Compare ST(0) with ST(1) and pop register stack twice.

Mnemonic	Description
FTST	Compare ST(0) with 0.0.

Mnemonic	Description
FUCOM ST(i)	Compare ST(0) with ST(i).
FUCOM	Compare ST(0) with ST(1).
FUCOMP ST(i)	Compare ST(0) with ST(i) and pop register stack.
FUCOMP	Compare ST(0) with ST(1) and pop register stack.
FUCOMPP	Compare ST(0) with ST(1) and pop register stack twice.

Mnemonic	Description
FXAM	Classify value or number in ST(0).

FXAM Results

Class	C3	C2	C0
Unsupported	0	0	0
NaN	0	0	1
Normal finite number	0	1	0
Infinity	0	1	1
Zero	1	0	0
Empty	1	0	1
Denormal number	1	1	0

4. Transcendental instructions

FSIN	Sine
FCOS	Cosine
FSINCOS	Sine and cosine
FPTAN	Tangent of ST
FPATAN	Arctangent of ST(1)/ST
F2XM1	$2^X - 1$
FYL2X	$Y \bullet \log_2 X$; Y is ST(1), X is ST
FYL2XP1	$Y \bullet \log_2(X + 1)$; Y is ST(1), X is ST

5. Constant Instructions

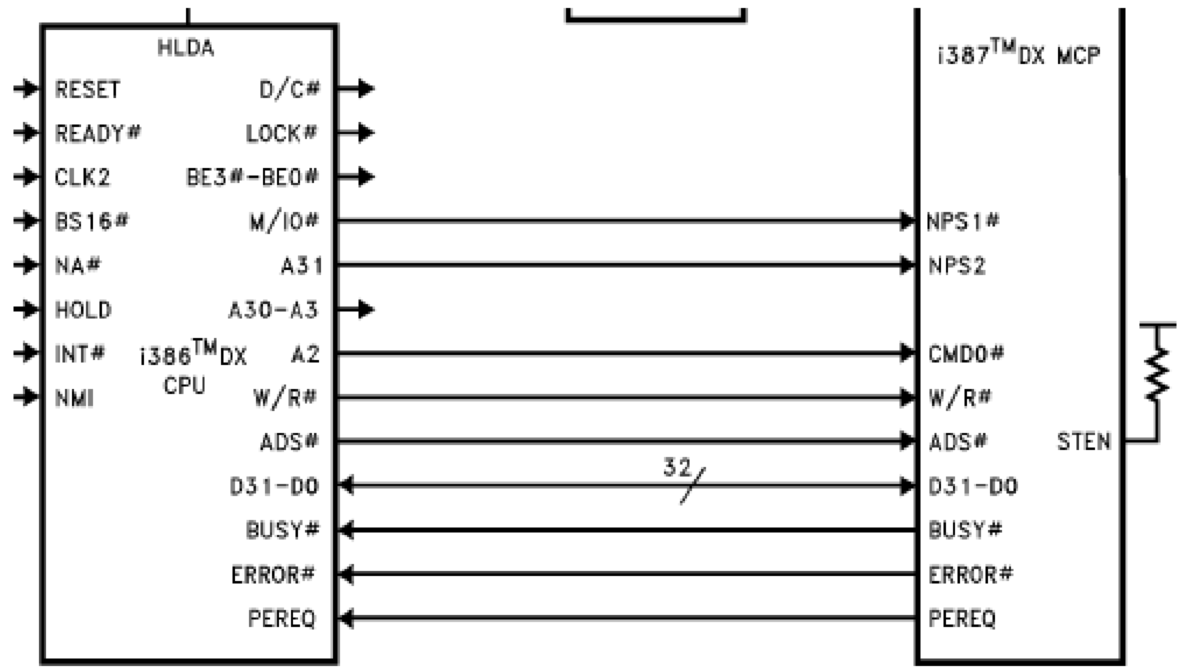
FLDZ	Load + 0.0
FLD1	Load + 1.0
FLDPI	Load π
FLDL2T	Load $\log_2 10$
FLDL2E	Load $\log_2 e$
FLDLG2	Load $\log_{10} 2$
FLDLN2	Load $\log_e 2$

6. Processor Control Instructions

FINIT/FNINIT	Initialize processor
FLDCW	Load control word
FSTCW/FNSTCW	Store control word
FSTSW/FNSTSW	Store status word
FSTSW AX/FNSTSW AX	Store status word to AX
FCLEX/FNCLEX	Clear exceptions
FSTENV/FNSTENV	Store environment
FLDENV	Load environment
FSAVE/FNSAVE	Save state
FRSTOR	Restore state
FINCSTP	Increment stack pointer
FDECSTP	Decrement stack pointer
FFREE	Free register
FNOP	No operation
FWAIT	CPU Wait

Interfacing Signals of 80386DX with 80387

80386 Pin	80387 Pin
M/IO#	NPS1#
A31	NPS2
A2	CMD0#
W/R#	W/R#
ADS#	ADS#
D31-D0	D31-D0
BUSY#	BUSY#
ERROR#	ERROR#
PEREQ	PEREQ



80387 Pin Description

RESETIN	Immediate termination of present operation
$\overline{\text{BUSY}}$	Output indicates coprocessor is executing a command.
$\overline{\text{ERROR}}$	Output indicates an unmasked error condition has occurred.
PEREQ	If output is high, ready to transfer data. If output is low, data is being transferred to or from / PEACK
CMD0	Command line for control of 80387 operations
$\overline{\text{NPS1}}$ & NPS2	Inputs indicates 80386 is performing an ESC instruction. No data transfer unless these lines are selected.
$\overline{\text{READY}}$	End of bus cycle signals this input pin. Bus ready.
$\overline{\text{READYO}}$	Write cycles last 2 clocks, read cycles last 3 clocks, and then terminates. Can drive $\overline{\text{READY}}$.
HLDA	Input informs coprocessor of 80386 bus control.
STEN	Status enable, serves as a chip select.
V_{SS}	Systems ground connection.
V_{CC}	Systems +5 volt power supply.