Series 32000 Instruction Execution Times

National Semiconductor
Application Note 555
August 1988

1.0 GENERAL INFORMATION

This document provides the necessary information to calculate the instruction execution times for the NS32008, NS32016, NS32032 and NS32332 CPUs. The following assumptions are made:

- The entire instruction, with all displacements and immediate operands, is assumed to be present in the instruction queue when needed.
- Interference from instruction prefetches, which is very dependent upon the preceding instruction(s), is ignored. This assumption will tend to affect the timing estimate in an optimistic direction.
- It is assumed that all memory operand transfers are completed before the next instruction begins execution. In the case of an operand of access class rmw in memory, this is pessimistic, as the Write transfer occurs in parallel with the execution of the next instruction.
- It is assumed that there is no overlap between the fetch of an operand and the following sequences of microcode. This is pessimistic, as the fetch of Operand 1 will generally occur in parallel with the effective address calculation of Operand 2, and the fetch of Operand 2 will occur in parallel with the execution phase of the instruction.
- Where possible, the values of operands are taken into consideration when they affect instruction timing, and a range of times is given. Where this is not done, the worst case is assumed.
- Memory accesses are assumed to occur at full speed. Any wait states should be reflected in the calculations of TOPB, TOPW and TOPD.

1.1 Definitions

TEA—	The time required to calculate an operand's Effective Address. For a Register or Immediate operand, this includes the fetch of that operand.
TGET—	NS32332 only. The time required to calculate the effective address and fetch the operand.
TMMU—	NS32016 and NS32032 only. The extra clock cycle required for translation of memory addresses if an NS32082 MMU is present.
TOPB—	The time needed to read or write a memory byte.
TOPW—	The time needed to read or write a memory word.
TOPD—	The time needed to read or write a memory double-word.
TOPi—	The time needed to read or write a memory operand, where the operand size is given by the operation length of the instruction. It is always equivalent to either TOPB, TOPW or TOPD.
TCY—	Internal processing overhead, in clock cycles.
L—	Internal processing whose duration depends on the operation length. The number of clock cycles is derived by multiplying this value by the number of bytes in the operation length.
NCYC—	Number of normal bus cycles performed by the CPU to fetch or store an operand. NCYC depends on the bus width as well as the operand size and alignment.
BCYC—	NS32332 only. Number of burst bus cycles performed by the CPU to access an operand. Burst cycles can occur during the access of non-aligned operands or for operands larger than the bus size.

1.2 Equations for the NS32008, NS32016 and NS32032 CPUs

TEA— TEA values for the various addressing modes are provided in the following table.

TEA Table

Addressing Mode	TEA Value	Notes
IMMEDIATE, ABSOLUTE	4	
EXTERNAL	11 + 2 * TOPD	
MEMORY RELATIVE	7 + TOPD	
REGISTER	2	
REGISTER RELATIVE, MEMORY SPACE	5	
TOP OF STACK	4 2 3	Access Class Write Access Class Read Access Class RMW
SCALED INDEXED	TI1 + TI2	

TI1 = TEA of the basemode except:

if basemode is REGISTER then TI1 = 5

if basemode is TOP OF STACK then TI1 = 4

TI2 depends on the scale factor:

if byte indexing TI2 = 5

if word indexing TI2 = 7

if double-word indexing TI2 = 8

if quad-word indexing TI2 = 10

TMMU— If a NS32082 MMU is present then TMMU = 1
else TMMU = 0

TOPB— If operand is in a register or is immediate then TOPB = 0
else TOPB = 3 + TMMU

TOPW— If operand is in a register or is immediate then TOPW = 0
else TOPW = (4 + TMMU) * NCYC − 1

TOPD— If operand is in a register or is immediate then TOPD = 0
else TOPD = (4 + TMMU) * NCYC − 1

TOPi— If operand is in a register or is immediate then TOPi = 0
else if i = byte then TOPi = TOPB
else if i = word then TOPi = TOPW
else (i = double-word) then TOPi = TOPD

L— If i (operation length) = byte then L = 1
else if i = word then L = 2
else (i = double-word) L = 4

TCY— TCY = 1

1.3 Equations for the NS32332 CPU

TEA— TEA values for the various addressing modes are provided in the following table.

TEA Table

Addressing Mode	TEA Value	Notes
IMMEDIATE	0	
ABSOLUTE	2	
EXTERNAL	6 + 2 * TOPD	
MEMORY RELATIVE	4 + TOPD	
REGISTER	0 2	 Class Address or Mode is Index
REGISTER RELATIVE, MEMORY SPACE	2	
TOP OF STACK	3 2	Access Class Write Access Class RMW or Scaled Indexed
SCALED INDEXED	2 + TI1	

TGET— TGET values for the various addressing modes are provided in the following table.

TGET Table

Addressing Mode	TGET Value	Notes
IMMEDIATE	3 2	First Operand Second Operand
ABSOLUTE	2 + TOPi	
EXTERNAL	6 + 2 * TOPD + TOPi	
MEMORY RELATIVE	4 + TOPD + TOPi	
REGISTER	0	
REGISTER RELATIVE, MEMORY SPACE	2 + TOPi	
TOP OF STACK	1 + TOPi 2 + TOPi	Access Class Read Access Class RMW or Scaled Indexed
SCALED INDEXED	2 + TI1 + TOPi	

TI1— TEA of the base mode

TOPB— If operand is in a register or is immediate then TOPB = 0 else TOPB = 3

TOPW— If operand is in a register or is immediate then TOPW = 0

else TOPW = 3 * NCYC + 2 * BCYC

TOPD— If operand is in a register or is immediate then TOPD = 0

else TOPD = 3 * NCYC + 2 * BCYC

TOPi— If operand is in a register or is immediate then TOPi = 0

else if i = byte then TOPi = TOPB

else if i = word then TOPi = TOPW

else (i = double-word) then TOPi = TOPD

L— If i (operation length) = byte then L = 1

else if i = word then L = 2

else (i = double-word) L = 4

TCY— TCY = 1

1.4 Calculation of Total Execution Time (T_{eff})

T_{eff} is obtained by performing the following steps:

1. Find the desired instruction in the tables.
2. Calculate the values of TEA, TOPB, etc. using the numbers in the tables and the equations given on the previous sections.
3. The result derived by adding together these values is the execution time (T_{eff}) in clock cycles.

1.5 Notes on Table Use

Values in the #TEA column indicate the number of effective addresses to be calculated. If the value in this column is less than the number of general operands in the instruction, this is because one or both operands are in registers and that instruction has an optimized form which eliminates TEA for such operands.

In the L column, multiply the entry by the operation length in bytes (1, 2 or 4).

In the TCY column, special notations sometimes appear:

n1 → n2 means n1 minimum, n2 maximum

n1%n2 means that the instruction flushes the instruction queue after n1 clock cycles and nonsequentially fetches the next instruction. The value n2 indicates the number of clock cycles for the internal execution of the instruction (including n1).

The effective number of cycles (TCY) must take into account the time (T_{fetch}) required to fetch the portion of the next instruction including the basic encoding and the index bytes. This time depends on the size and alignment of this portion as well as the bus width.

If only one memory cycle is required, then:

TCY = n1 + 5 + T_{fetch} (NS32332)
TCY = n1 + 6 + T_{fetch} (NS32008, NS32016, NS32032)

If more than one memory cycle is required, then:

TCY = n1 + 4 + T_{fetch} (NS32332)
TCY = n1 + 5 + T_{fetch} (NS32008, NS32016, NS32032)

In the notes column, notations held within angle brackets < > indicate alternatives in the operand addressing modes which affect the execution time. A table entry which is affected by the operand addressing may have multiple values, corresponding to the alternatives. This addressing notations are:

<I>	Immediate
<R>	CPU Register
<M>	Memory
<F>	FPU Register, either 32 or 64 Bits
<x>	Any Addressing Mode
<ab>	a and b represent the addressing modes of operands 1 and 2 respectively. Both a and b can be any addressing mode. (e.g., <MR> means memory to CPU register).

Note: Unless otherwise specified the TCY value for immediate addressing is the same as for CPU register addressing.

1.6 Example of Table Usage for the NS32016

Calculate T_{eff} for the instruction:

CMPW R0, TOS

Operand 1 is in a register; Operand 2 is in memory. This means that we must use the table values corresponding to the <xM> case as given in the Notes column.

Only the #TEA, #TOPi and TCY columns have values assigned for the CMPi instruction. Therefore, they are the only ones that need to be calculated to find T_{eff}. The blank columns are irrelevant to this instruction.

The #TEA column contains 2 for the <xM> case. This means that effective address times have to be calculated for both operands. (For the <MR> case, the Register operand would have required no TEA time, therefore only the Memory operand TEA would have been necessary.) From the equations:

TEA (Register mode) = 2,

TEA (Top of Stack mode, access class read) = 2,

Total TEA = 2 + 2 = 4.

The #TOPi column represents potential operand transfers to or from memory. For a Compare instruction, each operand is read once, for a total of two operand transfers.

TOPi (Word, Register) = 0,
TOPi (Word, TOS) = 3 (assuming the operand aligned and no MMU)

Total TOPi = 3

TCY is the time required for internal operation within the CPU. The TCY value for this case is 3.

T_{eff} = TEA + TOPi + TCY = 4 + 3 + 3 = 10 machine cycles.

If the CPU is running at 10 MHz then a machine cycle (clock cycle) is 100 ns. Therefore, this instruction would have 10 × 100 ns, or 1.0 μs, to execute.

1.7 Example of Table Usage for the NS32332

Calculate T_{eff} for the instruction:

CBITB R0, TOS

Operand 1 is in a register; Operand 2 is in memory. This means that we must use the table values corresponding to the <xM> case as given in the Notes column (<xM> meaning "anyting to memory").

Only the #TEA, #TGET, #TOPB and TCY columns have values assigned for the CBITi instruction. Therefore, they are the only ones that need to be calculated to find T_{eff}. The blank columns are irrelevant to this instruction.

The #TEA column contains 1 for the <xM> case. (For the <MR> case, the Register operand would have required no TEA time, therefore only the Memory operand TEA would have been necessary). From the equations:

TEA (Top of Stack mode, RMW access class) = 2

The #TGET column contains 1. TGET in this case represents the time required to calculate the address of the first operand and to fetch its value. From the equations:

TGET (Register Addressing Mode) = 0

The #TOPB column represents potential operand transfers to or from memory. For the CBITB R0, TOS the second operand is first read and then is written back to memory, for a total of two operand transfers:

TOPB (BYTE, TOS) = 3
Total TOPB = 6

TCY is the time required for internal operation within the CPU. The TCY value for this case is 17.

T_{eff} = TEA + TGET + TOPB + TCY = 2 + 0 + 6 + 17 = 25 Machine Cycles.

If the CPU is running at 15 MHz then a machine cycle (clock cycle) is 66.6 ns. Therefore, this instruction would take 25 × 66.6 ns, or 1.6 μs, to execute.

2.0 EXECUTION TIMES FOR THE NS32008, NS32016 AND NS32032 CPUs

The following tables provide the execution timing information for the basic and memory management instructions as well as the floating-point instructions when the NS32081 floating-point unit is used.

The additional parameters, used in the floating-point execution timing table, are described below.

S— This parameter is related to the floating-point operand size as follows:

Standard Floating (32 bits): S = 0

Long Floating (64 bits): S = 1

Tf— The time required to transfer 32 bits of a floating-point value to or from the Floating-Point Unit.

Tf = 4 always.

Ti — The time required to transfer an integer value to or from the Floating-Point Unit.

Byte:	Ti = 2
Word:	Ti = 2
Double-Word:	Ti = 4

2.1 Basic and Memory Management Instructions: NS32008, NS32016 and NS32032

Mnemonic	#TEA	#TOPB	#TOPW	#TOPD	#TOPi	L	TCY	Notes
ABSi	2	—	—	—	2	—	9	SCR < 0
	2	—	—	—	2	—	8	SCR > 0
ACBi	1	—	—	—	2	—	16	<M> no branch
	1	—	—	—	2	—	15%20	<M> branch
	—	—	—	—	—	—	18	<R> no branch
	—	—	—	—	—	—	17%22	<R> branch
ADDi	2	—	—	—	3	—	3	<xM>
	1	—	—	—	1	—	4	<MR>
	0	—	—	—	0	—	4	<RR>
ADDCi	2	—	—	—	3	—	3	<xM>
	1	—	—	—	1	—	4	<MR>
	0	—	—	—	0	—	4	<RR>
ADDPi	2	—	—	—	3	—	16	no carry
	2	—	—	—	3	—	18	carry
ADDQi	1	—	—	—	2	—	6	<M>
	0	—	—	—	0	—	4	<R>
ADDR	2	—	—	1	—	—	2	<xM>
	1	—	—	0	—	—	3	<xR>
ADJSPi	1	—	—	—	1	—	6	
ANDi	2	—	—	—	3	—	3	<xM>
	1	—	—	—	1	—	4	<MR>
	0	—	—	—	0	—	4	<RR>
ASHi	2	1	—	—	2	—	14 ⟶ 45	
Bcond	—	—	—	—	—	—	7	no branch
	—	—	—	—	—	—	6%10	branch
BICi	2	—	—	—	3	—	3	<xM>
	1	—	—	—	1	—	4	<MR>
	0	—	—	—	0	—	4	<RR>
BICPSRB	1	1	—	—	—	—	18%22	
BICPSRW	1	—	1	—	—	—	30%34	
BISPSRB	1	1	—	—	—	—	18%22	
BISPSRW	1	—	1	—	—	—	30%34	
BPT	—	—	4	3	—	—	40	
BR	—	—	—	—	—	—	6%10	
BSR	—	—	—	1	—	—	6%16	
CASEi	1	—	—	—	1	—	4%9	
CBITi	2	2	—	—	1	—	15	<xM>
	1	0	—	—	1	—	7	<xR>
CBITIi	2	2	—	—	1	—	15	<xM>
	1	0	—	—	1	—	7	<xR>
CHECKi	2	—	—	—	3	—	7	high
	2	—	—	—	3	—	10	low
	2	—	—	—	3	—	11	ok
CMPi	2	—	—	—	2	—	3	<xM>
	1	—	—	—	1	—	3	<MR>
	0	—	—	—	0	—	3	<RR>
CMPMi	2	—	—	—	2 * n	—	9 * n + 24	n = # of elements in block

2.1 Basic and Memory Management Instructions: NS32008, NS32016 and NS32032 (Continued)

Mnemonic	#TEA	#TOPB	#TOPW	#TOPD	#TOPi	L	TCY	Notes
CMPQi	1 0	— —	— —	— —	1 0	— —	3 3	<M> <R>
CMPSi	—	—	—	—	2 * n	—	35 * n + 53	n = # of elements, not Translated
CMPST	—	n	—	—	2 * n	—	38 * n + 53	Translated
COMi	2	—	—	—	2	—	7	
CVTP	2	—	—	1	—	—	7	
CXP	—	—	3	4	—	—	16%21	
CXPD	1	—	3	3	—	—	13%18	
DEIi	2 1	— —	— —	— —	5 1	16 16	38 31	<xM> <xR>
DIA	—	—	—	—	—	—	3%7	
DIVi	2	—	—	—	3	16	58 → 68	
ENTER	—	—	—	n + 1	—	—	4 * n + 18	n = # of general registers saved
EXIT	—	—	—	n + 1	—	—	5 * n + 17	n = # of general registers restored
EXTi	2 2	— —	— —	1 —	1 1	— —	19 → 29 17 → 51	field in memory field in register
EXTSi	2	—	—	1	1	—	26 → 36	
FFSi	2	2	—	—	1	24	24 → 28	
FLAG	— —	— —	0 4	0 3	— —	— —	6 44	no trap trap
IBITi	2 1	2 0	— —	— —	1 —	— —	17 9	<xM> <xR>
INDEXi	2	—	—	—	2	16	25	
INSi	2 1	— —	— —	2 —	1 1	— —	29 → 39 28 → 96	field in memory field in register
INSSi	2	—	—	2	1	—	39 → 49	
JSR	1	—	—	1	1	—	5%15	
JUMP	1	—	—	—	—	—	2%6	
LMR	1	—	—	—	1	—	30%34	
LPRi	1	—	—	—	1	—	19 → 33	
LSHi	2	1	—	—	2	—	14 → 45	
MEIi	2	—	—	—	4	16	23	
MODi	2	—	—	—	3	16	54 → 73	
MOVi	2 1 0	— — —	— — —	— — —	2 1 0	— — —	1 3 3	<xM> <MR> <RR>
MOVMi	2	—	—	—	2 * n	—	3 * n + 20	n = # of elements in block
MOVQi	1 0	— —	— —	— —	1 0	— —	2 3	<M> <R>
MOVSi	— —	— —	— —	— —	2 * n 2 * n	— —	13 * n + 18 24 * n + 54	n = # of elements no options B, W and/or U option in effect
MOVST	—	n	—	—	2 * n	—	27 * n + 54	Translated

2.1 Basic and Memory Management Instructions: NS32008, NS32016 and NS32032 (Continued)

Mnemonic	#TEA	#TOPB	#TOPW	#TOPD	#TOPi	L	TCY	Notes
MOVSUi	2	—	—	—	2	—	33%37	
MOVUSi	2	—	—	—	2	—	33%37	
MOVXBD	2	1	—	1	—	—	6	
MOVXBW	2	1	1	—	—	—	6	
MOVXWD	2	—	1	1	—	—	6	
MOVZBD	2	1	—	1	—	—	5	
MOVZBW	2	1	1	—	—	—	5	
MOVZWD	2	—	1	1	—	—	5	
MULi	2	—	—	—	3	16	15	
NEGi	2	—	—	—	2	—	5	
NOP	—	—	—	—	—	—	3	
NOTi	2	—	—	—	2	—	5	
ORi	2	—	—	—	3	—	3	<xM>
	1	—	—	—	1	—	4	<MR>
	0	—	—	—	0	—	4	<RR>
QUOi	2	—	—	—	3	16	49 → 55	
RDVAL	1	1	—	—	—	—	21	
REMi	2	—	—	—	3	16	57 → 62	
RESTORE	—	—	—	n	—	—	5 * n + 12	n = # of general registers restored
RET	—	—	—	1	—	—	2%8	
RETI	—	1	3	3	—	—	39%45	
RETT	—	—	2	2	—	—	35%41	
ROTi	2	1	—	—	2	—	14 → 45	
RXP	—	—	1	2	—	—	2%6	
Scondi	1	—	—	—	1	—	9	No Branch
	1	—	—	—	1	—	10	Branch
SAVE	—	—	—	n	—	—	4 * n + 13	n = # of general registers saved
SBITi	2	2	—	—	1	—	15	<xM>
	1	0	—	—	1	—	7	<xR>
SBITIi	2	2	—	—	1	—	15	<xM>
	1	0	—	—	1	—	7	<xR>
SETCFG	—	—	—	—	—	—	15	
SKPSi	—	—	—	—	n	—	27 * n + 51	n = # of elements, not Translated
SKPST	—	n	—	—	n	—	30 * n + 51	Translated
SMR	1	—	—	—	1	—	25	
SPRi	1	—	—	—	1	—	21 → 27	
SUBi	2	—	—	—	3	—	3	<xM>
	1	—	—	—	1	—	4	<MR>
	0	—	—	—	0	—	4	<RR>
SUBCi	2	—	—	—	3	—	3	<xM>
	1	—	—	—	1	—	4	<MR>
	0	—	—	—	0	—	4	<RR>

2.1 Basic and Memory Management Instructions: NS32008, NS32016 and NS32032 (Continued)

Mnemonic	#TEA	#TOPB	#TOPW	#TOPD	#TOPi	L	TCY	Notes
SUBPi	2	—	—	—	3	—	16	no carry
	2	—	—	—	3	—	18	carry
SVC	—	—	4	3	—	—	40	
TBITi	2	1	—	—	1	—	14	<xM>
	1	0	—	—	1	—	4	<xR>
WAIT	—	—	—	—	—	—	6 → ?	? = until an interrupt/reset
WRVAL	1	1	—	—	—	—	21	
XORi	2	—	—	—	3	—	3	<xM>
	1	—	—	—	1	—	4	<MR>
	0	—	—	—	0	—	4	<RR>

2.2 Floating-Point Instructions: NS32008, NS32016 and NS32032 with NS32081 FPU

Mnemonic	#TEA	#TOPD	#TOPi	#Ti	#Tf	TCY	Notes
MOVf	2	2 + 2 * S	—	—	2 + 2 * S	23	<MM>
	—	—	—	—	—	27	<FF>
	1	1 + S	—	—	1 + S	23	<MF>
	—	1 + S	—	—	1 + S	27	<FM>
ADDf, SUBf	2	3 + 3 * S	—	—	3 + 3 * S	70	<MM>
	—	—	—	—	—	74	<FF>
	1	1 + S	—	—	1 + S	70	<MF>
	1	2 + 2 * S	—	—	2 + 2 * S	70	<FM>
MULf	2	3 + 3 * S	—	—	3 + 3 * S	44 + 14 * S	<MM>
	—	—	—	—	—	48 + 14 * S	<FF>
	1	1 + S	—	—	1 + S	44 + 14 * S	<MF>
	1	2 + 2 * S	—	—	2 + 2 * S	44 + 14 * S	<FM>
DIVf	2	3 + 3 * S	—	—	3 + 3 * S	85 + 30 * S	<MM>
	—	—	—	—	—	89 + 30 * S	<FF>
	1	1 + S	—	—	1 + S	85 + 30 * S	<MF>
	1	2 + 2 * S	—	—	2 + 2 * S	85 + 30 * S	<FM>
ABSf, NEGf	1	2 + 2 * S	—	—	2 + 2 * S	20	<MM>
	—	—	—	—	—	24	<FF>
	1	1 + S	—	—	1 + S	20	<MF>
	—	1 + S	—	—	1 + S	24	<FM>
CMPf	2	2 + 2 * S	—	—	2 + 2 * S	45	<MM>
	—	—	—	—	—	49	<FF>
	1	1 + S	—	—	1 + S	45	<MF>
	1	1 + S	—	—	1 + S	45	<FM>
MOVLF	1	3	—	—	3	23	<MM>
	—	—	—	—	—	27	<FF>
	1	2	—	—	2	23	<MF>
	—	1	—	—	1	27	<FM>
MOVFL	1	3	—	—	3	22	<MM>
	—	—	—	—	—	26	<FF>
	1	1	—	—	1	22	<MF>
	—	2	—	—	2	26	<FM>
MOVif	1	1 + S	1	1	1 + S	53	<MM>
	1	—	1	1	—	53	<MF>
ROUNDfi, TRUNCfi, FLOORfi	1	1 + S	1	1	1 + S	53	<MM>
	—	—	1	1	—	66	<FM>
SFSR	—	1	—	—	1	13	
LFSR	1	1	—	—	1	18	

3.0 EXECUTION TIMES FOR THE NS32332 CPU

The NS32332 execution times are provided in this section. The table for the basic and memory management instructions is similar to the one given in section 2.1. The only differences consist in the addition of the TGET parameter and the differentiation, in the MMU instructions, between the 32-bit and 16-bit slave protocols used by the NS32382 and NS32082 respectively.

The table for the floating-point instructions is significantly different from the one provided in section 2.2. This table provides only the portion of the total execution time required by the CPU to decode the instruction as well as to fetch and store the operands. The FPU execution times are totally dependent on the floating-point unit being used.

The CPU portion of the total execution time is divided into three parts:

1. Pre-Slave Execution Time.

   The time required by the CPU to decode the instruction and transfer the operands to the FPU.

2. In-Parallel-To-Slave Execution Time.

   Includes the time required to calculate the destination address.

   This time should not be included in the calculation of the total execution time, unless it is greater than the FPU execution time.

3. Post-Slave Execution Time.

   The time required to read and store the result (if any) and to complete the instruction.

Two additional parameters are also defined: Tfx and Tfy. Both of them represent the time required to transfer an operand from the CPU to the slave. The time needed to transfer a result from the slave to the CPU is included in the TCY column in the post-slave execution portion.

The reason for defining Tfx and Tfy is that the operand transfer time may be different depending upon the instruction being executed as well as whether it is the first or the second operand. Tfx and Tfy are provided in the following tables.

Tfx TABLE

Addressing Mode and Operand Size	Tfx Value	Slave Protocol
REGISTER	0	
IMM FLOAT	2 4	32-Bit 16-Bit
IMM LONG	4 8	32-Bit 16-Bit
MEM FLOAT	3 5	32-Bit 16-Bit
MEM LONG	6 + TOPD 10 + TOPD	32-Bit 16-Bit

Tfy TABLE

Addressing Mode and Operand Size	Tfx Value	Slave Protocol
REGISTER	0	
IMM FLOAT	2 4	32-Bit 16-Bit
IMM LONG	6 10	32-Bit 16-Bit
MEM FLOAT	2 4	32-Bit 16-Bit
MEM LONG	6 + TOPD 10 + TOPD	32-Bit 16-Bit

3.1 Basic and Memory Management Instruction: NS32332

Mnemonic	#TEA	#TGET	#TOPB	#TOPW	#TOPD	#TOPi	L	TCY	Notes
ABSi	1	1	—	—	—	1	—	7	<xM> SRC > 0
	1	1	—	—	—	1	—	8	<xM> SRC < 0
	—	1	—	—	—	—	—	9	<xR> SRC > 0
	—	1	—	—	—	—	—	10	<xR> SRC < 0
ACBi	—	1	—	—	—	1	—	7	<M> No Branch
	—	1	—	—	—	1	—	7%8	<M> Branch
	—	—	—	—	—	—	—	9	<R> No Branch
	—	—	—	—	—	—	—	7%10	<R> Branch
ADDi	—	2	—	—	—	1	—	4	<xM>
	—	1	—	—	—	—	—	4	<xR>
ADDCi	—	2	—	—	—	1	—	4	<xM>
	—	1	—	—	—	—	—	4	<xR>
ADDPi	—	2	—	—	—	1	—	20	No Carry
	—	2	—	—	—	1	—	21	Carry
ADDQi	—	1	—	—	—	1	—	4	<M>
	—	—	—	—	—	—	—	4	<R>
ADDR	2	—	—	—	—	1	—	1	<xM>
	2	—	—	—	—	—	—	3	<xR>
ADJSPi	—	1	—	—	—	—	—	8	
ANDi	—	2	—	—	—	1	—	4	<xM>
	—	1	—	—	—	—	—	4	<xR>
									N = Number of Shifts
ASHi	—	2	—	—	—	1	—	7	<xM> N = 0
	—	1	—	—	—	—	—	8	<xR> N = 0
	—	2	—	—	—	1	—	11 + N	<xM> N > 0
	—	1	—	—	—	—	—	12 + N	<xR> N > 0
	—	2	—	—	—	1	—	12 + |N|	<xM> N < 0
	—	1	—	—	—	—	—	13 + |N|	<xR> N < 0
Bcond	—	—	—	—	—	—	—	7	No Branch
	—	—	—	—	—	—	—	5%8	Branch
BICi	—	2	—	—	—	1	—	4	<xM>
	—	1	—	—	—	—	—	4	<xR>
BICPSRB	—	1	—	—	—	—	—	14	
BICPSRW	—	1	—	—	—	—	—	14	No PSR(U)
	—	1	—	—	—	—	—	13%16	PSR(U)
BISPSRB	—	1	—	—	—	—	—	14	
BISPSRW	—	1	—	—	—	—	—	14	No PSR(U)
	—	1	—	—	—	—	—	13%16	PSR(U)
BPT	—	—	—	4	3	—	—	31	
BR	—	—	—	—	—	—	—	2%5	
BSR	—	—	—	—	1	—	—	5%7	
CASEi	—	1	—	—	—	—	—	6%9	
CBITi	1	1	2	—	—	—	—	16	<xM>
	—	1	—	—	—	—	—	7	<xR>
CBITIi	1	1	3	—	—	—	—	17	<xM>
	—	1	—	—	—	—	—	7	<xR>
CHECKi	1	1	—	—	—	1	—	10	High
	1	1	—	—	—	2	—	19	Low
	1	1	—	—	—	2	—	20	O.K.

3.1 Basic and Memory Management Instruction: NS32332 (Continued)

Mnemonic	#TEA	#TGET	#TOPB	#TOPW	#TOPD	#TOPi	L	TCY	Notes
CMPi	—	2	—	—	—	—	—	3	<xM>, <xR>
	—	2	—	—	—	—	—	5	<xI>
									n = Number of Compares
CMPMi	2	—	—	—	—	2n	—	8 * n + 19	i = B
	2	—	—	—	—	2n	—	8 * n + 22	i = W
	2	—	—	—	—	2n	—	8 * n + 23	i = D
CMPQi	—	1	—	—	—	—	—	3	
									n = Number of Elements
CMPSi	—	—	—	—	—	2n	—	35 * n + (56 → 64)	n > 0
	—	—	—	—	—	—	—	15	n = 0 (No Elements)
									n = Number of Elements
CMPST	—	—	n	—	—	2n	—	40 * n + (57 → 65)	n > 0
	—	—	—	—	—	—	—	15	n = 0
COMi	1	1	—	—	—	1	—	6	<xM>
	—	1	—	—	—	—	—	8	<xR>
CVTP	2	—	—	—	1	—	—	4	<M>
	2	—	—	—	—	—	—	6	<R>
CXP	—	—	—	3	4	—	—	15	(Till flush Queue 9 cyc. + 2 TOPW + 2 TOPD).
CXPD	1	—	—	3	3	—	—	13	(Till flush Queue 7 cyc. + 2 TOPW + 1 TOPD).
DEIi	—	2	—	—	—	3	16	26 → 27	<xM>
	—	1	—	—	—	—	16	31 → 33	<xR>
DIA	—	—	—	—	—	—	—	2%5	
DIVi	—	2	—	—	—	1	16	49 → 56	<xM>
	—	1	—	—	—	—	16	50 → 57	<xR>
									n = Number of Registers Saved
ENTER	—	—	—	—	1	—	—	9	n = 0
	—	—	—	—	n + 1	—	—	n + 13	n > 0
									n = Number of Registers Saved
EXIT	—	—	—	—	1	—	—	5	n = 0
	—	—	—	—	n + 1	—	—	3 * n + 6	n > 0
									Field in Memory
EXTi	2	—	—	—	1	1	—	18	(Offset MOD 8) = 0
	2	—	—	—	1	1	—	20 + (Offset MOD 8)	(Offset MOD 8) > 0
									Field in Register
	1	—	—	—	—	1	—	13	(Offset MOD 32) = 0
	1	—	—	—	—	1	—	15 + (Offset MOD 32)	(Offset MOD 32) > 0
EXTSi	1	1	—	—	—	—	—	21	<xR> (Offset MOD 8) = 0
	1	1	—	—	—	—	—	23 + (Offset MOD 8)	<xR> (Offset MOD 8) > 0
	1	1	—	—	—	1	—	19	<xM> (Offset MOD 8) = 0
	1	1	—	—	—	1	—	21 + (Offset MOD 8)	<xM> (Offset MOD 8) > 0
									<xM> '1' Not Found
FFSi	—	2	—	—	—	1	—	17	Offset = 0
	—	2	—	—	—	1	—	20 + Offset	Offset > 0
									<xM> '1' Found
	—	2	—	—	—	1	—	(24 → (21 + 24* i))	Offset = 0
	—	2	—	—	—	1	—	((27 + Offset) → → (24 + 24 * i − 2 * Offset))	Offset > 0

3.1 Basic and Memory Management Instruction: NS32332 (Continued)

Mnemonic	#TEA	#TGET	#TOPB	#TOPW	#TOPD	#TOPi	L	TCY	Notes
									<xR> '1' Not Found
FFSi	—	1	—	—	—	—	—	18	Offset = 0
(Continued)	—	1	—	—	—	—		21 + Offset	Offset > 0
									<xR> '1' Found
	—	1	—	—	—	—	—	25 → (22 + 24 * i)	Offset = 0
	—	1	—	—	—	—	—	((28 + Offset) →	Offset > 0
								→ (25 + 24 * i − 2 * Offset))	
FLAG	—	—	—	—	—	—	—	6	No Trap
	—	—	—	4	3	—	—	35	Trap
IBITi	1	1	2	—	—	—	—	18	<xM>
	—	1	—	—	—	—	—	9	<xR>
INDEXi	—	2	—	—	—	—	—	83	
									Field in Memory
INSi	1	1	—	—	2	—	—	29	(Offset MOD 8) = 0
	1	1	—	—	2	—	—	35 + 2 * (Offset MOD 8)	(Offset MOD 8) > 0
									Field in Register
	—	1	—	—	—	—	—	23	(Offset MOD 32) = 0
	—	1	—	—	—	—	—	29 + 2 * (Offset MOD 32)	(Offset MOD 32) > 0
									Field In Memory
INSSi	—	2	—	—	1	—	—	29	(Offset MOD 8) = 0
	—	2	—	—	1	—	—	35 + 2 * (Offset MOD 8)	(Offset MOD 8) > 0
									Field In Register
	—	1	—	—	—	—	—	31	(Offset MOD 8) = 0
	—	1	—	—	—	—	—	37 + 2 * (Offset MOD 8)	(Offset MOD 8) > 0
JSR	1	—	—	—	1	—	—	4%6	
JUMP	1	—	—	—	—	—	—	1%4	
LMR	—	1	—	—	—	—	—	14%18	NS32082 MMU
	—	1	—	—	—	—	—	13%17	NS32382 MMU
LPRi	—	1	—	—	—	—	—	10	UPSR
	—	1	—	—	—	—	—	17%20	PSR
	—	1	—	—	—	—	—	18	No PSR
									N = Number of Shifts
LSHi	—	2	—	—	—	1	—	7	<xM> N = 0
	—	1	—	—	—	—	—	8	<xR> N = 0
	—	2	—	—	—	1	—	11 + N	<xM> N > 0
	—	1	—	—	—	—	—	12 + N	<xR> N > 0
	—	2	—	—	—	1	—	12 + |N|	<xM> N < 0
	—	1	—	—	—	—	—	13 + |N|	<xR> N < 0
MEIi	—	2	—	—	—	2	16	17	<xM>
	—	1	—	—	—	—	16	20	<xR>
MODi	—	2	—	—	—	1	16	46 → 49,	<xM> Remainder = 0
	—	2	—	—	—	1	16	56 → 63	<xM> Remainder < > 0
	—	1	—	—	—	—	16	45 → 50,	<xR> Remainder = 0
	—	1	—	—	—	—	16	57 → 64	<xR> Remainder < > 0
MOVi	1	1	—	—	—	1	—	1	<xM>
	—	1	—	—	—	—	—	3	<xR>
									n = Number of Elements in Block
MOVMi	2	—	—	—	—	2n	—	19 + 2 * n,	i = B
	2	—	—	—	—	2n	—	22 + 2 * n,	i = W
	2	—	—	—	—	2n	—	23 + 2 * n	i = D

3.1 Basic and Memory Management Instruction: NS32332 (Continued)

Mnemonic	#TEA	#TGET	#TOPB	#TOPW	#TOPD	#TOPi	L	TCY	Notes
MOVQi	1	—	—	—	—	1	—	1	<M>
	—	—	—	—	—	—	—	3	<R>
									n = Number of Elements
MOVSi	—	—	—	—	—	2n	—	12 * n + (30 → 35)	No Options
	—	—	—	—	—	2n	—	28 * n + (62 → 70)	B, W and/or
									U Options in Effect
	—	—	—	—	—	—	—	12	No Elements
									n = Number of Elements
MOVST	—	—	n	—	—	2n	—	33 * n + (63 → 71)	B, W and/or U
									Options in Effect
	—	—	—	—	—	—	—	12	No Elements
MOVSUi	2	—	—	—	—	2	—	19	
MOVUSi	2	—	—	—	—	2	—	19	
MOVXBD	1	1	—	—	1	—	—	5	<xM>
	—	1	—	—	—	—	—	7	<xR>
MOVXBW	1	1	—	1	—	—	—	5	<xM>
	—	1	—	—	—	—	—	7	<xR>
MOVXWD	1	1	—	—	1	—	—	5	<xM>
	—	1	—	—	—	—	—	7	<xR>
MOVZBD	1	1	—	—	1	—	—	4	<xM>
	—	1	—	—	—	—	—	6	<xR>
MOVZBW	1	1	—	1	—	—	—	4	<xM>
	—	1	—	—	—	—	—	6	<xR>
MOVZWD	1	1	—	—	1	—	—	4	<xM>
	—	1	—	—	—	—	—	6	<xR>
MULi	—	2	—	—	—	1	16	11	<xM>
	—	1	—	—	—	—	16	12	<xR>
NEGi	1	1	—	—	—	1	—	4	<xM>
	—	1	—	—	—	—	—	6	<xR>
NOP	—	—	—	—	—	—	—	3	
NOTi	1	1	—	—	—	1	—	4	<xM>
	—	1	—	—	—	—	—	6	<xR>
ORi	—	2	—	—	—	1	—	4	<xM>
	—	1	—	—	—	—	—	4	<xR>
QUOi	—	2	—	—	—	1	16	41 → 47	<xM>
	—	1	—	—	—	—	16	42 → 48	<xR>
RDVAL	1	—	1	—	—	—	—	30	NS32082 MMU
	1	—	1	—	—	—	—	25	NS32382 MMU
REMi	—	2	—	—	—	1	16	45 → 51	<xM>
	—	1	—	—	—	—	16	46 → 52	<xR>
									n = Number of Registers
									Restored
RESTORE	—	—	—	—	—	—	—	5	n = 0
	—	—	—	—	n	—	—	3 * n + 6	n > 0
RET	—	—	—	—	1	—	—	5%5	
RETI	—	—	1	2	2	—	—	32	Non Cascaded
	—	—	2	2	3	—	—	33	Cascaded

3.1 Basic and Memory Management Instruction: NS32332 (Continued)

Mnemonic	#TEA	#TGET	#TOPB	#TOPW	#TOPD	#TOPi	L	TCY	Notes
RETT	—	—	—	2	2	—	—	22	(Till flush Queue 14 cyc. + 2 TOPW + 1 TOPD)
ROTi									N = Number of Shifts
	—	2	—	—	—	1	—	7	<M> N = 0
	—	1	—	—	—	—	—	8	<R> N = 0
	—	2	—	—	—	1	—	11 + N	<M> N > 0
	—	1	—	—	—	—	—	12 + N	<R> N > 0
	—	2	—	—	—	1	—	12 + N	<M> N < 0
	—	1	—	—	—	—	—	13 + N	<R> N < 0
RXP	—	—	—	1	2	—	—	6	(Till flush Queue 2 + TOPD)
Scondi	1	—	—	—	—	1	—	5	<M>
	—	—	—	—	—	—	—	7	<R>
SAVE									n = Number of Registers Saved
	—	—	—	—	—	—	—	5	n = 0
	—	—	—	—	n	—	—	n + 6	n > 0
SBITi	1	1	2	—	—	—	—	16	<xM>
	—	1	—	—	—	—	—	7	<xR>
SBITIi	1	1	3	—	—	—	—	17	<xM>
	—	1	—	—	—	—	—	7	<xR>
SETCFG	—	—	—	—	—	—	—	5	
SKPSi	—	—	—	—	—	n	—	28 * n + (55 → 63)	n = Number of Elements B, W and/or U Options in Effect
	—	—	—	—	—	n	—	14	No Elements
SKPST	—	—	n	—	—	n	—	33 * n + (56 → 64)	n = Number of Elements B, W and/or U Options in Effect
	—	—	n	—	—	n	—	14	No Elements
SMR	1	—	—	—	—	1	—	18	<M> NS32082 MMU
	—	—	—	—	—	—	—	20	<R> NS32082 MMU
	1	—	—	—	—	1	—	12	<M> NS32382 MMU
	—	—	—	—	—	—	—	14	<R> NS32382 MMU
SPRi	1	—	—	—	—	1	—	7	<M> No UPSR
	1	—	—	—	—	1	—	10	<M> UPSR
	—	—	—	—	—	—	—	9	<R> No UPSR
	—	—	—	—	—	—	—	12	<R> UPSR
SUBi	—	2	—	—	—	1	—	4	<xM>
	—	1	—	—	—	—	—	4	<xR>
SUBCi	—	2	—	—	—	1	—	4	<xM>
	—	1	—	—	—	—	—	4	<xR>
SUBPi	—	2	—	—	—	1	—	20	No Carry
	—	2	—	—	—	1	—	21	Carry
SVC	—	—	—	4	3	—	—	31	
TBITi	1	1	1	—	—	—	—	14	<M>
	—	—	—	—	—	—	—	4	<R>
WAIT	—	—	—	—	—	—	—	5 → ?	? = Until an Interrupt/Reset
WRVAL	1	—	—	—	—	—	—	30	NS32082 MMU
								25	NS32382 MMU
XORi	—	2	—	—	—	1	—	4	<xM>
	—	1	—	—	—	—	—	4	<xR>

3.2 Floating-Point Instructions (CPU Portion): NS32332

Mnemonic	Pre-Slave Execution				In-Parallel-to-Slave Execution		Post-Slave Execution		Notes
	#TGET	#Tfx	#Tfy	TCY	#TEA	TCY	#TOPD	TCY	
									<xF>
MOVf	1	1	—	4	—	—	—	3	32-Bit Prot.
	1	1	—	7	—	—	—	5	16-Bit Prot.
									<xM> Float
	1	1	—	4	1	7	1	8	32-Bit Prot.
	1	1	—	7	1	7	1	12	16-Bit Prot.
									<xM> Long
	1	1	—	4	1	6	2	13	32-Bit Prot.
	1	1	—	7	1	6	2	19	16-Bit Prot.
									<xF>
ADDf,	1	1	—	4	—	—	—	3	32-Bit Prot.
SUBf,	1	1	—	7	—	—	—	5	16-Bit Prot.
MULf,									<xM> Float
DIVf	2	1	1	4	—	7	1	6	32-Bit Prot.
	2	1	1	7	—	7	1	10	16-Bit Prot.
									<xM> Long
	2	1	1	4	—	6	2	13	32-Bit Prot.
	2	1	1	7	—	6	2	19	16-Bit Prot.
									<xF>
ABSf,	1	1	—	4	—	—	—	3	32-Bit Prot.
NEGf	1	1	—	7	—	—	—	5	16-Bit Prot.
									<xM> Float
	1	1	—	4	1	7	1	6	32-Bit Prot.
	1	1	—	7	1	7	1	10	16-Bit Prot.
									<xM> Long
	1	1	—	4	1	6	2	13	32-Bit Prot.
	1	1	—	7	1	6	2	19	16-Bit Prot.
CMPf	2	1	1	4	—	7	—	12	32-Bit Prot.
	2	1	1	7	—	7	—	11	16-Bit Prot.
									<xF>
MOVLF	1	1	—	4	—	—	—	3	32-Bit Prot.
	1	1	—	7	—	—	—	5	16-Bit Prot.
									<xM>
	1	1	—	4	1	—	1	6	32-Bit Prot.
	1	1	—	7	1	—	1	10	16-Bit Prot.
									<xF>
MOVFL	1	1	—	4	—	—	—	3	32-Bit Prot.
	1	1	—	7	—	—	—	5	16-Bit Prot.
									<xM>
	1	1	—	4	1	—	2	9	32-Bit Prot.
	1	1	—	7	1	—	2	15	16-Bit Prot.

3.2 Floating-Point Instructions (CPU Portion): NS32332 (Continued)

Mnemonic	Pre-Slave Execution				In-Parallel-to-Slave Execution		Post-Slave Execution		Notes
	#TGET	#Tfx	#Tfy	TCY	#TEA	TCY	#TOPD	TCY	
									<xF>
MOVif	1	—	—	7	—	—	—	3	32-Bit Prot.
	1	—	—	10	—	—	—	5	16-Bit Prot. i = 1, 2
	1	—	—	12	—	—	—	5	i = 4
									<xM> Float
	1	—	—	7	—	—	1	8	32-Bit Prot.
	1	—	—	10	—	—	1	12	16-Bit Prot. i = 1, 2
	1	—	—	12	—	—	1	12	i = 4
									<xM> Long
	1	—	—	7	—	—	2	13	32-Bit Prot.
	1	—	—	10	—	—	2	19	16-Bit Prot. i = 1, 2
	1	—	—	12	—	—	2	19	i = 4
									<xF>
ROUNDfi,	1	1	—	4	—	1	—	8	32-Bit Prot.
TRUNCfi,	1	1	—	7	—	1	—	12	16-Bit Prot.
FLOORfi									<xM>
	1	1	—	4	1	1	1	6	32-Bit Prot.
	1	1	—	7	1	1	1	10	16-Bit Prot.
									32-Bit Prot. Only
LOGBf	1	1	—	4	—	—	—	3	<xF>
	1	1	—	4	1	7	1	6	<xM> Float
	1	1	—	4	1	6	2	13	<xM> Long
									32-Bit Prot. Only
SCALBf	1	1	—	4	—	—	—	3	<xF>
	2	1	1	4	—	7	1	6	<xM> Float
	2	1	1	4	—	6	2	13	<xM> Long
DOTf, POLYf	2	1	1	4	—	—	—	3	32-Bit Prot. Only
									<R>
SFSR	1	1	—	4	—	1	—	8	32-Bit Prot.
	1	1	—	7	—	1	—	12	16-Bit Prot.
									<M>
	1	1	—	4	1	1	1	6	32-Bit Prot.
	1	1	—	7	1	1	1	10	16-Bit Prot.
LFSR	1	—	—	7	—	—	—	3	32-Bit Prot.
	1	—	—	12	—	—	—	5	16-Bit Prot.

3.3 NS32381/NS32C081 FPU Execution Times

The FPU execution times for the NS32381 and NS32C081 are provided in this section. The numbers represent average execution times obtained by using typical operands.

Listed below are definitions of the timing terms:

EXT— EXecution Time. This is the time from the last data sent to the FPU, until the early DONE is issued. (FPU Pipe is empty).

EDD— Early Done Delta. This is the time from when the early DONE is issued until the execution of the next instruction may start.

Provided that the CPU can transfer the ID/OPCODE and any operands to the FPU during the EDD time, the average system execution time for an instruction (keeping the FPU pipe full) is: EXT + EDD

The system execution time for a single FPU instruction with FPU register destination and early DONE is: EXT plus the protocol time. (FPU pipe initially empty).

Instruction		EXT	EDD	Total
LFSR	any, reg	5	8	13
MOVF	any, reg	5	6	11
MOVL	any, reg	5	8	13
MOVif	any, reg	5	45	50
MOVFL	any, reg	9	6	15
ADDF	any, reg	11	31	42
ADDL	any, reg	11	31	42
SUBF	any, reg	11	31	42
SUBL	any, reg	11	31	42
MULF	any, reg	11	20	31
MULL	any, reg	11	27	38
DIVF	any, reg	11	45	56
DIVL	any, reg	11	59	70
POLYF	any, any	15	46	61
POLYL	any, any	15	53	68
DOTF	any, any	15	46	61
DOTL	any, any	15	53	68

The following instructions do not generate an early DONE. In this case, EXT is the time from the last data sent to the FPU, until the normal DONE is issued. (The FPU pipe is empty).

Instruction		EXT
SFSR	reg, mem	7
MOVLF	any, any	18
ROUNDfi	any, mem	46
TRUNCfi	any, mem	46
FLOORfi	any, mem	46
CMPF	any, any	17
CMPL	any, any	17
ABSf	any, any	9
NEGf	any, any	9
SCALBf	any, any	49
LOGBf	any, any	36

Lit.# 100555

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408