Execution

COLLABORATORS			
	TITLE : Execution		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		June 9, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Execution

1.1 68000 Execution Times

```
                From Motorola's:
------------------------------------------------------------
M68000 8-/16-/32-BIT MICROPROCESSORS USER'S MANUAL (Section 8)
------------------------------------------------------------
Retyped by Subhuman/Epsilon
Converted to Amiga Guide by Dancing Fool/Epsilon
```


1.2 efaddr

Effective Address Calculation Times

Addressing Mode		Byte,Word	Long
	REGISTER		
Dn	Data Register Direct	0	0
An	Address Register Direct	0	0
	MEMORY		
(An)	Address Register Indirect	4	8
(An)+	Addr Reg Indirect w/Postincr	4	8
-(An)	Addr Reg Indirect w/Predecrement	6	10
(d16,An)	Addr Reg Indirect w/Displacement	8	12
(d8,An,Xn)*	Addr Register Indirect w/Index	10	14
(xxx).W	Absolute Short	8	12
(xxx).L	Absolute Long	12	16
(d8,PC)	PC Indirect with Displacement	8	12
(d16,PC,Xn)*	PC Indirect with Index	10	14
#(data)	Immediate	4	8

*The size of the index register (Xn) does not affect execution time.

1.3 moves

Move Byte and Word Instruction Execution Times

SOURCE	DESTINATION								
	Dn	An	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xn)*	(xx.W)	(xx).L
Dn	4	4	8	8	8	12	14	12	16
An	4	4	8	8	8	12	14	12	16
(An)	8	8	12	12	12	16	18	16	20
(An)+	8	8	12	12	12	16	18	16	20
-(An)	10	10	14	14	14	18	20	18	22
(d16,An)	12	12	16	16	16	20	22	20	24
(d8,An,Xn)*	14	14	18	18	18	22	24	22	26
(xxx).W	12	12	16	16	16	20	22	20	24
(xxx).L	16	16	20	20	20	24	26	24	28
(d16,PC)	12	12	16	16	16	20	22	20	24
(d8,PC,Xn)*	14	14	18	18	18	22	24	22	26
#(data)	8	8	12	12	12	16	18	16	20

*The size of the index register (Xn) does not affect execution time.

Move Long Instruction Execution Times

SOURCE	DESTINATION								
	Dn	An	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xn)*	(xx.W)	(xx).L
Dn	4	4	12	12	12	16	18	16	20
An	4	4	12	12	12	16	18	16	20
(An)	12	12	20	20	20	24	26	24	28
(An)+	12	12	20	20	20	24	26	24	28
-(An)	14	14	22	22	22	26	28	26	30
(d16,An)	16	16	24	24	24	28	30	28	32
(d8,An,Xn)*	18	18	26	26	26	30	32	30	34
(xxx).W	16	16	24	24	24	28	30	28	32
(xxx).L	20	20	28	28	28	22	34	32	36
(d,PC)	16	16	24	24	24	28	30	28	32
(d,PC,Xn)*	18	18	26	26	26	30	32	30	34
#(data)	12	12	20	20	20	24	26	24	28

*The size of the index register (Xn) does not affect execution time.

1.4 std

Standard Instruction Execution Times
An - Address register operand
Dn - Data register operand
ea - An operand specified by an effective address
M - Memory effective address operand

Instruction	Size	op<ea>,An^1	op<ea>,Dn	op Dn,<M>
ADD/ADDA	Byte,Word	8+	4+	8+
	Long	6+**	6+**	12+
AND	Byte,Word	-	4+	8+
	Long	-	6+**	12+
CMP/CMPA	Byte,Word	6+	4+	-
	Long	6+	6+	-
DIVS	-	-	158+*	-
DIVU	-	-	140+*	-
EOR	Byte,Word	-	4***	8+
	Long	-	8***	12+
MULS	-	-	70+*	-

MULU	-	-	70+*	-
OR	Byte,Word	-	4+	8+
	Long	-	6+**	12+
SUB	Byte,Word	8+	4+	8+
	Long	6+**	6+**	12+

Notes:
+ add effective address calculation time
^{1} word or long only
* indicates maximum basic value added to word effective address time.
** The base time of six clock periods is increased to eight if the effective address mode is register direct or immediate (effective address time should also be added).
*** Only available effective address mode is data register direct.

DIVS, DIVU - The divide algorithm used by the MC68000 provides less than 10% difference between the best and worst case timings.

MULS, MULU - The multiply algorithm requires 38+2n clocks where n is defined as:
MULU: n = the number of ones in the <ea>
MULS: n = concatanate the <ea> with a zero as the LSB; n is the resultant number of 10 or 01 patterns in the 17-bit source; i.e., worst case happens when the source is $5555.

1.5 immed

Immediate Instruction Execution Times
- Immediate operand
Dn - Data register operand
An - Address register operand
M - Memory operand

Instruction	Size	op #,Dn	op #,An	op #,M
ADDI	Byte,Word	8	-	12+
	Long	16	-	20+
ADDQ	Byte,Word	4	4*	8+
	Long	8	8	12+
ANDI	Byte,Word	8	-	12+
	Long	14	-	20+
CMPI	Byte,Word	8	-	8+
	Long	14	-	12+
	Byte,Word	8	-	12+

EORI	Long	16	-	20+
MOVEQ	Long	4	-	-
ORI	Byte,Word	8	-	12+
	Long	16	-	20+
SUBI	Byte,Word	8	-	12+
	Long	16	-	20+
SUBQ	Byte,Word	4	8*	8+
	Long	8	8	12+

1.6 sngl

Single Operand Instruction Execution Times

Instruction	Size	Register	Memory
CLR	Byte,Word	4	8+
	Long	6	12+
NBCD	Byte	6	8+
NEG	Byte,Word	4	8+
	Long	6	12+
NEGX	Byte,Word	4	8+
	Long	6	12+
NOT	Byte,Word	4	8+
	Long	6	12+
Scc	Byte,False	4	8+
	Byte,True	6	8+
TAS	Byte	4	14+
TST	Byte,Word	4	4+
	Long	4	4+

+ add effective address calculation time

1.7 shift

Shift/Rotate Instruction Execution Times

Instruction	Size	Register	Memory
ASR, ASL	Byte,Word	6 + 2n	8+
	Long	8 + 2n	-
LSR, LSL	Byte,Word	6 + 2n	8+
	Long	8 + 2n	-
ROR, ROL	Byte,Word	6 + 2n	8+
	Long	8 + 2n	-
ROXR, ROXL	Byte,Word	6 + 2n	8+
	Long	8 + 2n	-

+ add effective address calculation time for word operands
n is the shift count

1.8 bit

Bit Manipulation Instruction Execution Times

Instruction	Size	Dynamic		Static	
		Register	Memory	Register	Memory
BCHG	Byte	-	8+	-	12+
	Long	8*	-	12*	-
BCLR	Byte	-	8+	-	12+
	Long	10*	-	14*	-
BSET	Byte	-	8+	-	12+
	Long	8*	-	12*	-
BTST	Byte	-	4+	-	8+
	Long	6*	-	10	-

+ add effective address calculation time

* indicates maximum value; data addressing mode only

1.9 cond

Conditional Instruction Execution Times

Instruction	Displacement	Branch Taken	Branch Not Taken
Bcc	Byte	10	8
	Word	10	12
BRA	Byte	10	-
	Word	10	-
BSR	Byte	18	-
	Word	18	-
DBcc	cc true	-	12
	cc false, Count Not Expired	10	_
	cc false, Counter Expired	_	14

1.10 jmp

JMP, JSR, LEA, PEA, and MOVEM Instruction Execution Times

Instr	S	(An)	(An)+	-(An)	(d16,An)	(d8,An,Xn)+	(xxx).W	(xxx).L	(d16,PC)	(d8,PC,Xn)*
JMP	-	8	-	-	10	14	10	12	10	14
JSR	-	16	-	-	18	22	18	20	18	22
LEA	-	4	-	-	8	12	8	12	8	12
PEA	-	12	-	-	16	20	16	20	16	20

```
+-------+---+-------+-------+------+--------+-----------+-------+-------+--------+--------
|       | W | 12+4n | 12+4n |  _   | 16+4n  |   18+4n   | 16+4n | 20+4n | 16+4n  | ↩
      18+4n   |
|
                MOVEM
                 +---+-------+-------+------+--------+-----------+-------+-------+--------+--------
| M->R  | L | 12+8n | 12+8n |  _   | 16+8n  |   18+8n   | 16+8n | 20+8n | 16+8n  | ↩
      18+8n   |
+-------+---+-------+-------+------+--------+-----------+-------+-------+--------+--------
|       | W |  8+4n |   _   | 8+4n | 12+4n  |   14+4n   | 12+4n | 16+4n |   _    | ↩
        _     |
| MOVEM  ↩
   +---+-------+-------+------+--------+-----------+-------+-------+--------+-----------+
|       | L |  8+8n |   _   | 8+8n | 12+8n  |   14+8n   | 12+8n | 16+8n |   _    | ↩
        _     |
+-------+---+-------+-------+------+--------+-----------+-------+-------+--------+--------
```

n is the number of registers to move.
* The size of the index register (Xn) does not affect the instruction's execution ↩
  time.

1.11 multi

Multi-Precision Instruction Execution Times
Dn - Data register operand
M - Memory operand

Instruction	Size	op Dn,Dn	op M,M
ADDX	Byte,Word	4	18
	Long	8	30
CMPM	Byte,Word	-	12
	Long	-	20
SUBX	Byte,Word	4	18
	Long	8	30
ABCD	Byte	6	18
SBCD	Byte	6	18

1.12 misc

Miscellaneous Instruction Execution Times

Instruction	Size	Register	Memory
ANDI to CCR	Byte	20	-
ANDI to SR	Word	20	-
CHK (No Trap)	-	10+	-
EORI to CCR	Byte	20	-
EORI to SR	Word	20	-
ORI to CCR	Byte	20	-
ORI to SR	Word	20	-
MOVE from SR	-	6	8+
MOVE to CCR	-	12	12+
MOVE to SR	-	6	12+
EXG	-	6	-
EXT	Word	4	-
	Long	4	-
LINK	-	16	-
MOVE from USP	-	4	-
MOVE to USP	-	4	-
NOP	-	4	-
RESET	-	132	-
RTE	-	20	-
RTR	-	20	-
RTS	-	16	-
STOP	-	4	-
SWAP	-	4	-
TRAPV	-	4	-
UNLK	-	12	-

+ add effective address calculation time

1.13 movep

Move Peripheral Instruction Execution Times

Instruction	Size	Register->Memory	Memory->Register
MOVEP	Word	16	16
	Long	24	24

1.14 except

Exception Processing Execution Times

Exception	Periods
Address Error	50
Bus Error	50
CHK Instruction	40+
Divide by Zero	38+
Illegal Instruction	34
Interrupt	44*
Privilege Violation	34
_____	
RESET**	40
Trace	34
TRAP Instruction	34
TRAPV Instruction	34

+ add effective address calculation time
* The interrupt acknowledge cycle is assumed to take four clock periods

** indicates the time from the $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ are first sampled as negated to when instruction execution starts

1.15 tas

Note: This instruction should never be used on the Amiga as its invisiable read/write cycle can disrupt system DMA.

1.16 movem

Note: This instruction should never be used from custom chip registers. On the 68000, this instruction will actually fetch MORE than the specified number of words. Reading from write/strobe registers can be dangerous.