Estimation of Assembly Programs Execution Time

- For a CPU running at a constant clock rate:

  clock rate = 1 / clock cycle time

- Every machine or assembly instruction takes one or more clock cycles to complete.
- The total time an assembly program requires to run is given by:

Execution time = Total number of cycles X Clock cycle time
= Instruction count X cycles per instruction X clock cycle time
= Instruction count X cycles per instruction / clock rate

Example:

For a CPU running at 8MHZ is executing a program with a total of 100 000 instructions. Assuming that each instruction takes 10 clock cycles to complete:

Execution time = 100 000 X 10 / 8 000 000 = 0.125 seconds

68000 Cycles Per Instruction

- For the 68000, the number of clock cycles required by an instruction to execute depends on:
  - The exact type of the instruction in question.
  - Addressing modes used for both the instruction source and destination if applicable.
  - Size of operands:
    - Byte- and word-sized operands usually take the same number of cycles to process.
    - Instructions with long word-sized operands take more cycles than that with byte/word-sized operands.
- For a 68000 running at 8MHZ all instruction times are rounded to multiples of four (i.e. 10 becomes 12 etc.).

68000 Instructions Summary

Instr	Description	Instr	Description
ABCD	Add decimal with extend	MOVE	Move source to destination
ADD	Add	MULS	Signed multiply
AND	Logical AND	MULU	Unsigned multiply
ASL	Arithmetic shift left	NBCD	Negate Decimal with Extend
ASR	Arithmetic shift right	NEG	Negate
Bcc	Branch conditionally	NOP	No operation
BCHG	Bit test and change	NOT	Ones complement
BCLR	Bit test and clear	OR	Logical OR
BRA	Branch always	PEA	Push effective address on stack
BSET	Bit test and set	RESET	Reset External devices
BSR	Branch to subroutine	ROL	Rotate left without extend
BTST	Bit test	ROR	Rotate right without extend
CHK	Check register against bounds	ROXL	Rotate left with extend
CLR	Clear operand	ROXR	Rotate right with extend
CMP	Compare	RTD	Return and deallocate
DBcc	Decrement and branch conditionally	RTE	Return from exception
DIVS	Signed divide	RTR	Return and restore
DIVU	Unsigned divide	RTS	Return from subroutine
EOR	Exclusive OR	SBCD	Subtract decimal with extend
EXG	Exchange registers	Scc	Set conditional
EXT	Sign extend	STOP	Stop
JMP	Jump	SUB	Subtract
JSR	Jump to subroutine	SWAP	Swap data register halves
LEA	Load Effective Address	TAS	Test and set operand
LINK	Link stack	TRAP	Trap
LSL	Logical shift left	TRAPV	Trap on overflow
LSR	Logical shift right	TST	Test

68000 Cycles For MOVE Instructions

Operand Size	Addressing Mode								
.b.w/.l	dn	an	(an)	(an)+	-(an)	d(an)	d(an,dn)	abs.s	abs.l
dn	4/4	4/4	8/12	8/12	8/14	12/16	14/18	12/16	16/20
an	4/4	4/4	8/12	8/12	8/14	12/16	14/18	12/16	16/20
(an)	8/12	8/12	12/20	12/20	12/20	16/24	18/26	16/24	20/28
(an)+	8/12	8/12	12/20	12/20	12/20	16/24	18/26	16/24	20/28
-(an)	10/14	10/14	14/22	14/22	14/22	18/26	20/28	18/26	22/30
d(an)	12/16	12/16	16/24	16/24	16/24	20/28	22/30	20/28	24/32
d(an,dn)	14/18	14/18	18/26	18/26	18/26	22/30	24/32	22/30	26/34
abs.s	12/16	12/16	16/24	16/24	16/24	20/28	22/30	20/28	24/32
abs.l	16/20	16/20	20/28	20/28	20/28	24/32	26/34	24/32	28/36
d(pc)	12/16	12/16	16/24	16/24	16/24	20/28	22/30	20/28	24/32
d(pc,dn)	14/18	14/18	18/26	18/26	18/26	22/30	24/32	22/30	26/34
Immediate	8/12	8/12	12/20	12/20	12/20	16/24	18/26	16/24	20/28

Clock Cycles

Time to Calculate Effective Addresses

Addressing Mode

Operand Size	(an)	(an)+	-(an)	d(an)	d(an,dn)
.b.w/.l	4/8	4/8	6/10	8/12	10/14

Addressing Mode

Operand Size	abs.s	abs.l	d(pc)	d(pc,dn)	Imm
.b.w/.l	8/12	12/16	8/12	10/14	4/8

The time taken to calculate the effective address must be added to instructions that affect a memory address.

68000 Cycles For Standard Instructions

Operand Size	Addressing Mode		
.b.w/.l	ea,an	ea,dn	dn,mem
add	8/6(8)	4/6(8)	8/12
and	-	4/6(8)	8/12
cmp	6/6	4/6	-
divs	-	158max	-
divu	-	140max	-
eor	-	4/8	8/12
muls	-	70max	-
mulu	-	70max	-
or	-	4/6(8)	8/12
sub	8/6(8)	4/6(8)	8/12

Clock Cycles

(8) time if effective address is direct

Add effective address times from above for mem addresses

Cycles For Immediate Instructions

Operand Size	Addressing Mode		
.b.w/.l	#,dn	#,an	#,mem
addi	8/16	-	12/20
addq	4/8	8/8	8/12
andi	8/16	-	12/20
cmpi	8/14	8/14	8/12
eori	8/16	-	12/20
moveq	4	-	-
ori	8/16	-	12/20
subi	8/16	-	12/20
subq	4/8	8/8	8/12

Clock Cycles

Moveq.l only

nbcd+tas.b only

scc false/true

Add effective address times from above for mem addresses

Cycles for Single-Operand Instructions

Operand Size | Addressing Mode

.b.w/.l	#,dn	#,an	#,mem
clr	4/6	4/6	8/12
nbcd	6	6	8
neg	4/6	4/6	8/12
negx	4/6	4/6	8/12
not	4/6	4/6	8/12
scc	4/6	4/6	8/8
tas	4	4	10
tst	4/4	4/4	4/4

Clock Cycles

Add effective address times from above for mem addresses

Cycles for Shift/Rotate Instructions

Operand Size	Addressing Mode		
.b.w/.l	dn	an	mem
asr,asl	6/8	6/8	8
lsr,lsl	6/8	6/8	8
ror,rol	6/8	6/8	8
roxr,roxl	6/8	6/8	8

Clock Cycles

Memory is byte only
For register add 2x the shift count

Misc. Instructions

Addressing Mode

	(an)	(an)+	-(an)	d(an)	d(an,dn)	abs.s	abs.l	d(pc)	d(pc,dn)
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
movem	t=4								
m>r	12	12	-	16	18	16	20	16	18
movem	t=5								
r>m	8	-	8	12	14	12	16	-	-
movem	add t x number of registers for .w								
movem	add 2t x number of registers for .l								

Clock Cycles

Cycles for Bit Manipulation Instructions

Addressing Mode

Operand Size .b/.l	register .l only	memory .b only
bchg	8/12	8/12
bclr	10/14	8/12
bset	8/12	8/12
btst	6/10	4/8

Clock Cycles

Cycles To Process Exceptions

Address Error	50
Bus Error	50
Interrupt	44
Illegal Instr.	34
Privilege Viol.	34
Trace	34

Cycles for Other Instructions

Operand Size .b.w/.l	dn,dn	m,m	Addressing Mode
addx	4/8	18/30	
cmpm	-	12/20	
subx	4/8	18/30	
abcd	6	18	.b only
sbcd	6	18	.b only
Bcc	.b/.w	10/10	8/12
bra	.b/.w	10/10	-
bsr	.b/.w	18/18	-
DBcc	t/f	10	12/14
chk	-	40 max	8
trap	-	34	-
trapv	-	34	4

Clock Cycles

Add effective address times from above for mem addresses

Cycles for Other Instructions

```
reg<>mem
movep   .w/.l   16/24
```

Addressing Mode

	Reg	Mem
andi to ccr	20	-
andi to sr	20	-
eori to ccr	20	-
eori to sr	20	-
exg	6	-
ext	4	-
link	18	-
move to ccr	12	12
move to sr	12	12
move from sr	6	8
move to usp	4	-

Addressing Mode

	Reg
move from usp	4
nop	4
ori to ccr	20
ori to sr	20
reset	132
rte	20
rtr	20
rts	16
stop	4
swap	4
unlk	12

Clock Cycles

Timing Example 1

	Instruction		Clock Cycles
RANDOM	ADDI.B	#17,D0	8
	LSL.B	#3,D0	12
	NOT.B	D0	4
	RTS		16
	Total Cycles needed:		40 cycles

For a 68000 running at 8MHZ:

Clock cycle = 125 nsec

Execution time = 40 X 125 nsec = 5 μs = 5 x 10^{-6} second

Timing Example 2

			Clock Cycles	
	Instruction		Overhead	Loop
	MOVE.B	#255,D0	8	
READ	ADD.W	(A0)+,D1		8
	SUBQ.B	#1,D0		4
	BNE	READ		10

Total Cycles Needed = 8 + 255 (8 + 4 + 10)

= 8 + 255 x 22

= 5618 cycles

Execution time for 8MHZ 68000 = 5618 x 125 nsec

= 0.00070225 Seconds = .702 msec

Timing Example 3

- **TOBIN converts a four-digit BCD number in the lower word of D0 into a binary number returned in D2**

	Instructions		Clock Cycles overhead	outer loop	inner loop
TOBIN	CLR.L	D2	6		
	MOVEQ	#3,D6	4		
NEXTDIGIT	MOVEQ	#3,D5		4	
	CLR.W	D1		4	
GETNUM	LSL.W	#1,D0			8
	ROXL.W	#1,D1			8
	DBRA	D5,GETNUM			10
	MULU	#10,D2		42	
	ADD.W	D1,D2		4	
	DBRA	D6,NEXTDIGIT		10	
	RTS		16		

Total Clock cycles = overhead + ((inner loop cycles x 4) + outer loop cycles) x 4
= 26 + ((26 x 4) + 64) x 4
= 26 + 168 x 4 = 698 cycles
= 698 x 125 nsec = 87.25 ms
or over 11 400 BCD numbers converted to binary every second.