MEMOREX 7100

MRX30 EMULATION AND PERFORMANCE

J.A. MILLER

7/19/72

TABLE OF CONTENTS

1. Conclusion

The Memorex 30 has, on the average, an instruction execution speed 1.08 times faster than the IBM System 3.

2. Introduction

The purpose of this discussion is to evaluate the performance of the MRX30. In addition, the evaluation provides a basis to describe how the instruction set is emulated by the 7100 microinstructions. Throughout this discussion reference is made to the memo CPU Speed of MRX50 by G. H. Leichner (Santa Clara Systems Programming Technical Memo PER 002, January 26, 1972), in which similar evaluative techniques were used. It is particularly valuable as the source for instruction mix ratios. Instruction execution times for the MRX50 were obtained from the 7200/7300 Computers Product Description manual of March 1972.

3. Evaluation of the MRX30

Since the MRX30 user instruction set is compatible with the MRX50, a comparison of execution speed with the MRX50 is the natural way to evaluate the MRX30. Once this comparison has been made, the results may be translated into the results of comparisons with other computers, notably the IBM System 3.

In what follows we shall be dealing with ratios of execution times of various instructions. If we let T(I) denote the execution time on machine I, the ratio we normally use is $R = \frac{T\ (MRX30)}{T\ (MRX50)}$. We also use a ratio R' which means the same except that the MRX50 is assumed to be in single processor mode. Let us now let S(I) be the speed of machine I. Since speed bears an inverse relation to time, we also have:

$$R = \frac{S\ (MRX50)}{S\ (MRX30)}$$

Where the R here is the same as above.

3.1 Comparison Between The MRX30 And MRX50

The MRX30 has a 400 nanosecond microinstruction cycle time and a 1.2 microsecond memory cycle time. In comparison, the MRX50 has a 100 nanosecond microinstruction cycle time and a .900 nanosecond memory cycle time. Thus, the MRX50 is 4 times the speed of the MRX30 in terms of microinstruction execution, but only 1.3 times the speed in memory operations. On this basis one would predict that the ratio R of MRX50 speed to MRX30 speed would fall between 4 and 1.3, with those operations requiring fewer memory cycles closer to 4, and those requiring more memory cycles closer to 1.3. This is true in almost all cases.

Another difference between the MRX30 and MRX50 is in the use of memory. In the MRX50 the microinstruction cycles are always locked to main memory cycles. In the MRX30, the only time the microinstruction execution becomes locked to memory timing is during a memory operation. This allows a timing on the MRX30 to be made in terms of microcycles rather than memory cycles. The result of this is that some operations, like indexing, take no time on the MRX50, but do take time on the MRX30.

The actual comparison between the MRX30 and MRX50 was done by micro-coding several instructions on the MRX30 and comparing their timing to the MRX50. The instructions chosen were ADDR, ADD, MOVR, LOD, B (ranch), BCT and MOVX. It should be noted that this subset consists of Register-Register, Memory-register and variable length memory ops. ADDR and ADD were chosen as being representative of a large class of combinatorial ops. MOVR, LOD, B and BCT were chosen because they are frequently used ops. MOVX was chosen to represent the variable length ops. Section 4 describes this comparison in detail.

The comparison showed that, for all the ops but MOVX, the ratio R was very strongly dependent upon the indirect addressing and indexing options chosen. In particular, the more indirect addressing that was done, the smaller R became. As a result of this observation, it was decided to perform the analysis both on the simple (no indexing, no indirect addressing) versions of the ops, and also on an average op obtained by a weighted average of addressing types. The weighting factors were computed by assuming: 1) 50% of the ops would have at least 1 indirect address, and 2) 50% of the ops would use indexing.

It was also decided to perform the analysis using both the normal and "single processor mode" MRX50.

From these techniques an execution time for each op for each case was obtained. The times thus obtained were averaged using a set of weights derived from operation versus frequency of execution tables. The results of these weighted averages could be said to represent the contribution of the subset of instructions to the average instruction execution time. This gave a set of times both for the MRX50 and for the MRX30 which could be used to compute a ratio for each of 4 cases. These results were:

CASE	R
Simple Addressing - Normal MRX50	2.4
Addressing Average - Normal MRX50	2.3
Simple Addressing - Single Processor MRX50	1.9
Addressing Average - Single Processor MRX50	1.9

As expected, the worst case arises for simple addressing and a normal MRX50. It is on this case that the conclusions given in Section 1 above are based. It is surprising that the result is relatively insensitive to differences between simple and average addressing. This arises for two reasons. The first is that the addressing average is weighted towards the simple ops. The second is that over half the contribution to the average instruction execution time comes in all cases from MOVX, the op that is not address type sensitive. It is worth noting at this point that Leichners memo shows that, even with decimal multiply and divide excluded, the variable length operations contribute over half to the average execution time in the MRX50.

3.2 Comparison of MRX30 with IBM System 3

Having now arrived at a figure of 2.4 for the ratio for MRX50 speed to MRX30 speed, we need to relate this figure to the IBM System 3. If we use S(I) for the speed of machine I, then we have the following relations:

$$\frac{S\ (MRX50)}{S\ (MRX30)} = 2.4$$

The published figure for the MRX50 shows:

$$\frac{S\ (MRX50)}{S\ (IBM50)} = .7$$

The following two relations represent assumptions:

$$\frac{S\ (IBM30)}{S\ (IBM50)} = .3$$

$$\frac{S\ (IBM\ S3)}{S\ (IBM\ 30)} = .9$$

We have then:

$$\frac{S\ (MRX30)}{S\ (IBM\ S3)} = \frac{S\ (MRX30)}{S\ (MRX50)} \cdot \frac{S\ (MRX50)}{S\ (IBM50)} \cdot \frac{S\ (IBM50)}{S\ (IBM30)} \cdot \frac{S\ (IBM30)}{S\ (IBM\ S3)}$$

$$= \frac{1}{2.4} \times .7 \times \frac{1}{.3} \times \frac{1}{.9} = 1.08$$

Which is the figure given above.

4. Emulation of the MRX30 Instruction Set

The starting and ending point in the emulation of any instruction is the read next instruction sequence (RNIZ, Table 1). This sequence picks up the first word of the next instruction from memory and increments the program counter to the next word. In addition it checks for a pending interrupt condition. Once the instruction is available, the RNIZ exits to a location dependent upon the 8 bit op-code of the instruction.

In what follows we shall discuss in some detail how various instructions and instruction types are emulated. In addition we will give examples of the microprogramming for these emulations, and some timing analysis based upon this microcoding.

4.1 The Basic Ops

By the term basic ops here, we mean the functions of ADD, SUB, CMP, XOR, EOR, MOV, and MVI, in conjunction with the addressing types RR, MR, Direct, Immediate, and Memory-memory.

The PIR decode branch for each of the basic ops branchs to a sequence of two subroutine calls. The first of these subroutine calls is a branch to an addressing type routine. The second call is a branch to an "operator" function that actually performs the operation.

The first (addressing type) routine takes care of getting the operands. The second (operation type) routine performs the operation on the operands, and takes care of disposition of the results, including the condition code, if any.

Thus PIRD becomes one of a number of program segments like:

```
(PIRDX)   BSR = XXXX
          BSR = XXXX
```

Where the individual routines are given in tables below:

ADDRESSING TYPE

ROUTINE TABLE

XXXX	ADDRESSING TYPE	OPERATION TABLE
ADRCT	DIRECT	1
AIMM	IMMEDIATE	1
AMR	MEMORY → REGISTER	1
ARR	REGISTER → REGISTER	1
AMM	MEMORY → MEMORY	2

OP TABLE 1

YYYY	OPERATION
OADD	ADD
OSUB	SUBTRACT
OAND	AND (Logical Product)
OIOR	EXCLUSIVE OR
OIOR	INCLUSIVE OR
OCMP	COMPARE
OMOV	MOVE
OMVI	MOVE INVERSE

OP TABLE 2 - Note these are in general alternate entries to routines listed in Op Table 1.

YYY	OPERATION
MADD	ADD
MSUB	SUBTRACT
MAND	AND (Logical Product)
MEOR	EXCLUSIVE OR
MIOR	INCLUSIVE OR
MCMP	COMPARE
MMOV	MOVE
MMVI	MOVE INVERSE

For the purposes of the timing analysis, the ARR and AMR addressing routines, and the OADD and OMOV operation routines were micro-coded. This coding is shown in Tables 2, 3, 4 and 5. Timings were obtained from this coding for all addressing variations of the instructions ADDR, MOVR, ADD and LOD. Summarys of these timings appear on Tables 6 and 7.

From these times an average time for each instruction was also computed. Each addressing variation in this average was weighed according to the following rules: 1) 50% of the instructions would use indirect address; and 2) 50% of the instructions would use indexing.

These calculations were made for the MRX50 in both the normal (MRX50 Short) and single processor mode (MRX50 Long). Tables 8 and 9 show these calculations.

4.2 Branch Operations

Emulation of the various branch operations generally consists of two program segments: 1) The execution of a branch condition test routine; 2) followed under certain conditions by the execution of a branch generation routine. It should be noted that the double subroutine call technique is not necessary since the branch generator used by a particular branch test routine is unique. The unconditional branchs will branch directly to a branch generator, since no condition test is necessary.

BRANCH GENERATORS

BGN1 - Generates a normal, post-indexed branch
BGN2 - Generates a register branch
BGN3 - Generates a pre-indexed branch
BGN4 - (Alternate entry to BGN3) Generates a non-indexed branch

BRANCH CONDITION TEST ROUTINE

RTN	USED FOR	CORR. BRANCH GENERATOR
TBA1	Branch Add 1	BGN1
TBA2	Branch Add 2	BGN1
TBOF	Branch if Bit off	BGN4
TBON	Branch if Bit on	BGN4
TBRN	Branch if Reg not zero	BGN1

TBRZ	Branch if Reg Zero	BGN1
TBCF	Branch if Condition False	BGN1
TBCT	Branch if Condition True	BGN1
TBS1	Branch Subtract One	BGN1
TBS2	Branch Subtract Two	BGN1

For the purposes of the timing analysis one branch generator (BGN1) and one branch condition test routine (TBCF) were coded. These are shown in Tables 10 and 11. From this coding timings we obtained for all addressing types for the instructions B and BCF. For BCF timings were obtained for both the branch and no branch cases. These results are presented in Table 12. In order to arrive at a single value for the branch operation, the following assumptions were made: 1) 50% of branch operations were unconditional and 50% were conditional; 2) 50% conditional branches were taken, and 50% were not. Further, for address averaging, the assumptions made in Section 4.1 were also made. The results of these averages are shown in Table 13.

4.3 Variable Length Operations

The variable length operations first pass through a variable length set up section which computes the effective operand addresses and rectifies the count fields. This initialization section is instruction independent, except that SHFK and MOVL have their own. Subsequent to this set up section, a section dependent upon the actual op is reached. This is accomplished by the same sort of double branch technique employed for the basic ops.

For the purposes of the timing analysis, a set-up section and the instruction dependent section for MOVX were microprogrammed. This coding is shown in Tables 14 and 15. For the timing, it was assumed (following Leichner) that the two operand lengths were both equal at 14.8 bytes. A summary of the timing for all addressing options is shown in Table 16. Since the times are only slightly dependent on addressing option, it was decided to use, for the MRX30, a value of 65.8 for the no address average case, and 69.0 for the address average case.

4.4 The Weighted Average Over Instruction Type

From Leichners memo a set of weights, one for each instruction, were derived that are proportional to the frequency of execution of that instruction. By using these as the weights in a weighted average we arrive at an overall figure of merit. There are 6 instances of this figure of merit. They break down into 2 cases of 3 instances each. The two cases are with and without address averaging. The 3 instances are: 1) The MRX50 running in normal mode (MRX50 Short); 2) The MRX50 running in single processor mode (MRX50 Long); and 3) the MRX30. Once the figure of merits have been developed, the ratios that were summarized above are calculated. It should be noted that, although only ADD and ADDR were evaluated, it was assumed that SUB and SBR would be the same and the weights were adjusted to include them in the average.

MRX30 EMULATION AND PERFORMANCE

APPENDIX: TABLES

```
RNI8    AMAR:S, PMR
        B = I(x'a')
        S = ADD, HOM
        PIR: MDR
        BRA: PIRD
        BRA. INT: IRUPT
```

timing = 6 cycles

Table 1 - RNI8 Microcode

ARR BRA, NIRR = ARRI
BRA = SRET
B = R2I

ARRI MAR = R2I, FMR
BRA = SRET, HOM
B = MDR

timing 3 cycles for R2 direct
5 cycles for R2 indirect

Table 2 - RR type address decode

```
AMR       AMAR=S, FMR
          S = ADD
          BRA. I2Z = AMRCP
          , HDM
          MDR → MAR, FMR
AMRCP     BRA. NR2Z = AMRCX
          , HDM
          MAR = MDR, FMR
AMRCB     BRA = SREL T, HDM
          B = MDR

AMRCX     A = R2I, HDM
          B = MDR
          MAR = ADD, FMR
          BRA = AMRCB
```

timing 9 cycles for direct with no index

10 cycles for direct with index

11 cycles for indirect with no index

12 cycles for indirect with index

Table 3 - FMR type address decode

OADD	BRA, NI1Z = OADI
	A = R1I
	R1I = ADD
OADB	A = UCR
	B = COND
	COND = INT, HDM
	BRA = RNI2
OADI	MAR = R1I, FMR
	, HDM
	A = MDR
	MDR = ADD, FMW
	BRA = OADB

timing - 7 cycles when R1 direct

11 cycles when R1 indirect

Table 4 - ADD Routine

```
OMOV    BRA. NIIZ = OMUI
        R2I = BDB
        BRA = RNIZ

OMUI    MDR = BDB
        MAR = R1I, FMW
        , HDM
        BRA = RNIZ
```

timing 3 cycles when R1 direct

6 cycles when R1 indirect

Table 5 - MOV Routine

ADD & ADDR Timing

	MRX50			MRX30			
	Cycles	Time	T'	μ-cycles	time	R	R'
RR	$M_1 + M_2$	1.7	2.1	18	7.2	4.2	3.4
2I	$2M_1$	1.8	2.2	20	8	4.4	3.6
1I	$3M_1 + M_2$	3.5	4.3	22	8.8	2.5	2.0
II	$4M_1$	3.6	4.4	24	9.6	2.6	2.2
MR	$3M_1$	2.7	3.3	24	9.6	3.6	2.9
M(X)R	$3M_1$	2.7	3.3	25	10.0	3.7	3.0
M(I)R	$4M_1$	3.6	4.4	26	10.4	2.9	2.4
M(IX)R	$4M_1$	3.6	4.4	27	10.8	3.0	2.5
MI	$5M_1$	4.5	5.5	28	11.2	2.5	2.0
M(X)I	$5M_1$	4.5	5.5	29	11.6	2.6	2.1
M(I)I	$6M_1$	5.4	6.6	30	12.0	2.2	1.8
M(IX)I	$6M_1$	5.4	6.6	31	12.4	2.3	1.9

Table 6 - ADD & ADDR Timing Summary

	MRX 50			MRX 30		R	R'
	Cycles	Time	T'	Cycles	Time		
RR	M1+M2	1.7	2.1	14	5.6	3.3	2.7
2I	2M1	1.8	2.2	16	6.4	3.6	2.9
1I	2M1+M2	2.6	3.2	17	6.8	2.6	2.1
II	3M1	2.7	3.3	19	7.6	2.8	2.3
MR	3M1	2.7	3.3	20	8.0	3.0	2.4
M(X)R	3M1	2.7	3.3	21	8.4	3.1	2.5
M(I)R	4M1	3.6	4.4	22	8.8	2.4	2.0
M(IX)R	4M1	3.6	4.4	23	9.2	2.6	2.1
MI	4M1	3.6	4.4	23	9.2	2.6	2.1
M(X)I	4M1	3.6	4.4	24	9.6	2.7	2.2
M(I)I	5M1	4.5	5.5	25	10.0	2.2	1.8
M(IX)I	5M1	4.5	5.5	26	10.4	2.3	1.9

Table 7 - MOVR & LOD Timing Summary

ADDR MRX 50 short

$1.7 \times .5 + (1.8 + 3.5 + 3.6)/6$

$= \left[1.7 + \frac{(1.8 + 3.5 + 3.6)}{3}\right]/2 = 2.3 \qquad (R = 3.5)$

ADDR MRX 50 long

$\left[2.8 + \frac{(2.2 + 4.3 + 4.4)}{3}\right]/2 = 2.9 \qquad (R' = 2.8)$

ADDR MRX 30

$\left[7.2 + \frac{(8 + 8.8 + 9.6)}{3}\right]/2 = 8$

ADD MRX50 short

$\left[2.7 + 2.7 + \frac{(3.6 + 3.6 + 4.5 + 4.5 + 5.4, 5.4)}{3}\right]/4 = 3.6 \quad (R = 2.9)$

ADD MRX 50 long

$\left[3.3 + 3.3 + \frac{(4.4 + 4.4 + 5.5 + 5.5 + 6.6 + 6.6)}{3}\right]/4 = 4.4 \quad (R' = 2.4)$

ADD MRX 30

$\left[9.6 + 10.0 + \frac{(10.4 + 10.8 + 11.2 + 11.6 + 12.0 + 12.4)}{3}\right]/4 = 10.6$

Table 8 - Address Averaging for ADD & ADDR

MOUR MRX50 short

$$= \left[1.7 + \frac{1.8+2.6+2.7}{3}\right]/2 = 2.0 \qquad (R = 3.2)$$

MOUR MRX50 Long

$$\left[2.1 + \frac{2.2+3.2+3.3}{3}\right]/2 = 2.5 \qquad (R' = 2.5)$$

MOUR MRX30

$$\left[5.6 + \frac{6.4+6.8+7.6}{3}\right]/2 = 6.3$$

LOD MRX50 short

$$\left[2.7 + 2.7 + \frac{(3.6+3.6+3.6+3.6+4.5+4.5)}{3}\right]/4 = 3.3 \qquad (R = 2.7)$$

LOD MRX50 Long

$$\left[3.3 + 3.3 + \frac{(4.4+4.4+4.4+4.4+5.5+5.5)}{3}\right]/4 = 4.0 \qquad (R' = 2.2)$$

LOD MRX30

$$\left[8.0 + 8.4 + \frac{(8.8+9.2+9.2+9.6+10.0+10.4)}{3}\right]/4 = 8.9$$

TABLE 9 Address Averaging for MOUR & LOD

```
BGN1
        MAR=S, FMR
        BRA.NIIZ=BGNA
        ,HOM
        MAR=MAR, FMR
BGNA    BRA.NRIZ=BGNX
        ,HOM
        S=MDR
        BRA=RNIZ

BGNX    A=RII, HOM
        B=MDR
        S=ADD
        BRA=RNIZ
```

timing: 6 cycles direct

7 cycles indexed

8 cycles indirect

9 cycles indirect and indexed.

Table 10 - Branch Generator BGN1

```
TBCF    B = PIR
        A = COND
        BBIT
        BRA. NUCZ = BGN1
        A = S
        B = X'2'
        S = ADD
        BRA = RNIZ
```

timing: 4 cycles if branch

8 cycles if no branch

Table 11 - Branch Condition Test TBCF

Unconditional Branch

	MRX 50 Cycles	MRX 50 Time	T'	MRX 30 Cycles	MRX 30 Time	R	R'
N	2 M,	1.8	2.2	12	4.8	2.7	2.2
X	2 M,	1.8	2.2	13	5.2	2.9	2.4
I	3 M,	2.7	3.3	14	5.6	2.1	1.7
I X	3 M,	2.7	3.3	15	6.0	2.2	1.8
		Branch on Condition (true or False)					
N - $\bar{B}$	2 M,	1.8	2.2	14	5.6	3.1	2.5
N - B	2 M,	1.8	2.2	16	6.4	3.6	2.9
X - $\bar{B}$	2 M,	1.8	2.2	14	5.6	3.1	2.5
X - B	2 M,	1.8	2.2	17	6.8	3.8	3.1
I - $\bar{B}$	3 M,	2.7	3.3	14	5.6	2.1	1.7
I - B	3 M,	2.7	3.3	18	7.2	2.7	2.2
I X - $\bar{B}$	3 M,	2.7	3.3	14	5.6	2.1	1.7
I X - B	3 M,	2.7	3.3	19	7.6	2.8	2.3

Table 12 - Branch Timing Summary

For No Address Average

MRX50 is independent of Branch type & whether or not branch was taken

MRX50 Short $t = 1.8$ $(R = 3)$

MRX50 Long $t = 2.2$ $(R' = 2.5)$

MRX30 $t = \left[4.8 + \frac{(5.6 + 6.4)}{2}\right]/2 = 5.4$

For Address Average

MRX50 Short $t = (1.8 + 2.7) = 2.25$ $(R = 2.6)$

MRX50 Long $t = (2.2 + 3.3) = 2.75$ $(R' = 2.1)$

MRX30 $t = \left[(4.8 + 5.2 + 5.6 + 6.0)/4 + (4 \times 5.6 + 6.4 + 6.8 + 7.2 + 7.6)/8\right]/2$

$= 5.85$

Note that the above average averages over:

1. Branch Type
2. Addressing type
3. Whether Branch taken or not.

Table 13 – Branch Type & Address Averaging

```
AUL     AMAR=S, FMR
        B=X'2',
        S=ADD, HOM
        BRA. NR1Z = AULX1
        OA1 = MDR
AULB1   AMAR=S, FMR
        B=X'2'
        S=ADD, HOM
        BRA.NR1Z = AULX2
        OA2 = MDR
AULB2   AMAR=S, FMR
        B=X'2'
        S=ADD, HOM
        L1=SMDR
        B=X'1'
        MAR=ADD,
        L2=SMDR
        A=OA1
        B=L1
        A=ADD
        B=X'1'
        OA1=SUB
        B=OA1
        A=SUB
        B=L2
        OA2=ADD
        A=L1
        CMP
        BRA.UCND=AUL2G
AULLG   RES=SUB
        CNT=L1
        BRA=SRET
        PIR=X'0'

AULX1   A=MDR
        B=R1I
        OA1=ADD
        BRA=AULB1
AULX2   A=MDR
        B=R2I
        OA2=ADD
        BRA=AULB2
AUL2G   B=SUB
        A=ZRO
        RES=SUB
        CNT=L2
        BRA=SRET
        PIR=X'8'
```

33 + 3 for each X + 2 for L2G

TABLE 14 - Variable Length Set-up section

```
OMOUX   B = X'1'
        CNT = CNT
        BRA. ZSB = OMXOT
OMXTP   AMAR = OA1, FMR
        OA1 = ADD, HDM
        AMAR = OA2, HMW
        OA2 = ADD
        A = CNT
        CNT = SUB, HDM
        BRA. NZSB = OMXTP
OMXOT   RES = RES
        BRA. ZSB = RUTZ
```

(this section is unimportant because the evaluation will assume that $L_1 = L_2$, so that RES = 0)

5 + 8 L

Table 15 - MOUX op-dependant CODE

MOUX Timing

Set up Only

	MRX 30 Cycles	MRX 30 Time	MRX 50 Cycles	MRX 50 Time	T'	R	R'
N	46	18.4	$6M_1+3M_2$	7.8	9.6	2.4	1.9
X	49	19.6	$6M_1+3M_2$	7.8	9.6	2.5	2.0
2X	52	20.8	$6M_1+3M_2$	7.8	9.6	2.7	2.2
L2G - N	48	19.2	$6M_1+3M_2$	7.8	9.6	2.5	2.0
L2G - X	51	20.4	$6M_1+3M_2$	7.8	9.6	2.6	2.1
L2G - 2X	54	21.6	$6M_1+3M_2$	7.8	9.6	2.8	2.3

Loop Only

$L_1 = L_2 = 14.8$

	8L	47.4	$2LM_1$	26.6	32.6	1.8	1.5

Total

N	46+8L	65.8	$6M_1+3M_2+2LM_1$	34.4	42.2	1.9	1.6
X	49+8L	67.0	"	34.4	42.2	1.9	1.6
2X	52+8L	68.2	"	34.4	42.2	2.0	1.6
L2G - N	48+8L	66.6	"	34.4	42.2	1.9	1.6
L2G - X	51+8L	67.8	"	34.4	42.2	2.0	1.6
L2G - 2X	54+8L	69.0	"	34.4	42.2	2.0	1.6

Table 16 - MOUX Timing Summary

Inst	Weight	No. Address Average						Address Average					
		MRX50 short		MRX50 long		MRX30		MRX50 short		MRX50 long		MRX30	
ADDR	.0352	1.7	.05984	2.1	.07392	7.2	.25344	2.3	.08096	2.9	.10208	8.0	.2816
ADD	.0152	2.7	.04104	3.3	.05016	9.6	.14592	3.6	.05472	4.4	.06688	10.6	.16112
MOVR	.0176	1.7	.02992	2.1	.03696	5.6	.09856	2.0	.0352	2.5	.044	6.3	.11088
LOD	.1626	2.7	.43902	3.3	.53658	8.0	1.3008	3.3	.53658	4.0	.6504	8.9	1.44714
JMP	.2324	1.8	.41832	2.2	.51128	5.4	1.25496	2.25	.5229	2.75	.6391	5.85	1.35954
MOVX	.0480	34.4	1.6512	42.2	2.0256	65.8	3.1584	34.4	1.6512	42.2	2.0256	69.0	3.312
TOTAL			2.64		3.23		6.21		2.88		3.53		6.67
R			2.4		1.9				2.3		1.9		

Table 17 - Parameters of the Weighted Average over Instruction Type