1-1-1977

An interactive simulator generating system.

Ramon Tan

AN INTERACTIVE SIMULATOR GENERATING SYSTEM

by

Ramon Tan

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Computer Science

Lehigh University

1977

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

April 18, 1977
(date)

Richard J. Cichelli
Professor in Charge

Arthur E. Pitcher
Chairman of Department

ACKNOWLEDGMENTS

I wish to thank the support provided by the Software Development Group of the Mathematics Department of Lehigh University. Sincere thanks are also due Messrs. Mueller and Johnson of the Colorado State University, the original authors of the GEN systems, who provided in source form the original versions that were basis for the interactive versions. Finally, the assistance, advice and encouragement of Mr. Richard J. Cichelli as thesis adviser is gratefully acknowledged. Any errors, inaccuracies or bugs remaining in the present work are the sole responsibility of this author.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF FIGURES

A B S T R A C T

Automatic production of microprocessor assemblers and simulators has recently been realized by the ASM/GEN and SIM/GEN systems of Mueller and Johnson at the Colorado State University. A modified version of these two systems to include an interactive simulator generating capability is the subject of this paper.

Chapter I presents the rationale behind the modification of the original systems of Mueller and Johnson. The interactive and run-time debugging features available from a generated simulator of the modified versions is illustrated.

Chapter II is a detailed discussion of the generation techniques employed by these systems to produce the desired assembler or simulator. The high-level language (FORTRAN) realization of the microprocessor instruction emulation code is presented.

Chapter III is a tutorial on using the SIM/GEN system for prospective users wishing to generate a simulator. An actual and existing microprocessor, the Motorola 6800, is used as example and the necessary steps involved in the generation of a simulator for this microprocessor are covered. The CDC 6400 host machine is assumed for this actual case study.

1.1 Modified ASM/GEN and SIM/GEN

ASM/GEN and SIM/GEN are a software system comprised of a set of FORTRAN program writer modules designed to generate microcomputer assemblers and simulators. Briefly, ASM/GEN is a program that is capable of producing a complete assembler for any specified microcomputer instruction set. The resulting assembler uses 2-pass, absolute assembly and offers the usual macro facility, conditional assembly and a set of useful pseudo-directives. The counter part system, SIM/GEN, produces a simulator for any specified microcomputer, excluding I/O interfaces and timing considerations. The generated simulator executes in batch mode, but is capable of providing run-time diagnostics in the form of a machine status dump. This capability relates to the *TRACE n pseudo-directive from the assembler generated by ASM/GEN. The *TRACE pseudo-directive is used during the assembly process to denote a call for a machine status dump at a certain point in a program. An instruction is said to be TRACEd if this pseudo-directive immediately precedes it. A machine status dump, depending on the value of n, is performed before the TRACEd instruction is executed. Thereafter, the status dump continues to be performed at the end of each instruction execution cycle, until a new TRACE level is encountered.

Specifically, a TRACE level number of n = 0 disables the TRACE option. This means that no further machine status dumps are to be performed. A level of n = 1 causes the contents of the program

counter and the instruction register to be displayed. When n = 2, the level 1 displays are performed, and in addition, all registers of the simulated machine are also displayed. When n = 3, a complete machine status dump is performed. This level of tracing displays entire sections of the memory configured, the address and data bus (a basic assumption of SIM/GEN for all data transfers), time elapsed and total instruction count, where the last 8 instructions were executed, as well as the displays at the lower level options.

It is our belief that while the above-mentioned diagnostic capability is no doubt informative and useful to the debugging of the simulated program, it suffers from certain inadequacies. First of all, the programmer must determine ahead of time where his TRACE points should be inserted. This implies that at assembly time, the programmer must decide on where he wishes to obtain machine status dumps. Secondly, because the simulator executes in a batch mode, the trace points are binding at simulation time. No capability for run time control exists on the part of the programmer. Finally, excessive printouts resulting from the machine status dumps will most likely occur. This is especially true of TRACEd loops that become indefinite, or for complete status dumps (trace level 3) in which the entire memory is displayed in addition to the microprocessor state variants. More so considering the fact that the machine status dump is performed after <u>every</u> instruction execution cycle. The need to facilitate increased programmer control over the program simulated and for better

run-time debugging tools has led to a modification of the ASM/GEN and SIM/GEN generating systems.

It was first decided that if a simulator was to be truly useful to the microprocessor development cycle, programmer interaction must exist during simulation time. Hence, SIM/GEN was modified to generate a simulator that could interact with the programmer at a terminal. Next, to allow for monitoring at run-time, a new level number was introduced to the *TRACE n pseudo-directive available from the generated assembler in ASM/GEN. A level number of 4 implies a "breakpoint" in the usual understanding of the term: when an instruction is encountered during execution that was trapped by a *TRACE 4 pseudo-operation, execution is suspended and programmer gains control. At this point, the programmer may input any one of the available commands in the simulator. The available commands allow for:

- displaying any section of memory;
- displaying any microprocessor architectural component, such as the program counter, the instruction register, the address or data bus, registers and so forth;
- changing the contents of any RAM memory location;
- changing the contents of any microprocessor architectural component;
- insert or remove breakpoints, or alter the level number of a breakpoint (note, that only a level number equal to 4 will cause suspension of program execution while all other level

numbers simply cause a machine status dump and proceed with execution);

- resume or halt execution, at the next logical location or at some other desired location;

The complete list of commands that may be used by the programmer using a simulator generated from SIM/GEN appears in the next section. In view of the ability to selectively dump sections of memory, the level 3 trace was reduced to a machine status dump which included all the displays mentioned earlier, except for the display of all of memory. Lastly, all machine status displays (levels 1 - 3) would be performed only at the encountered instruction, and not at every instruction thereafter.

The net result of a modified SIM/GEN is a simulator that offers the programmer a better debugging tool in the microprocessor software development process. While retaining the machine status displays previously available from the batch simulator, a simulator generated from the modified SIM/GEN will also allow for the interactive features described. Program monitoring and control have been achieved considerably.

1.2 Available commands from generated simulator

A separator is either a blank or a comma. Values are all assumed to be hexadecimal, while register numbers decimal.

Command	-	Explanation/syntax
B	-	display address and data bus;
PC	-	display program counter;
IR	-	display instruction register;
ST	-	display hardwired stack;
SRn	-	display special register n (no separator);
GRn	-	display general register n (no separator);
T	-	display total time elapsed, no. of instructions;
H	-	display available commands;
S	-	halt simulation run;
G	-	resume simulation run at current value of program counter;
/ n	-	remove the nth breakpoint (separator required, default is n=current breakpoint encountered);
* addr n	-	insert an nth level (n between 1 and 4, default=4) *TRACE at addr;
L	-	list all breakpoints and their level numbers;
.	-	display current location;
..	-	display current location and next;
...	-	display current and next 2 locations;
D addr1 addr2	-	display memory locations addr1 to addr2, inclusive; addr2 is optional (separators required);
C arg hexval	-	change arg to the hexadecimal value hexval; arg may be any one of the ff:

AS = address bus,
DS = data bus,
PC = program counter,
SP = stack pointer,
ST = top of hardwired stack,
SRn = special register n,
GRn = general register n;

If not any of the above, arg is assumed to be a hexadecimal value denoting some memory location to be altered to contain hexval;

Sixteen (16) breakpoints is the maximum number allowed for by the generated simulator.

1.3 Example using the modified GEN systems

An assembler for the Motorola 6800 microprocessor may be generated using the set of inputs shown in Appendix A. The generation run listing is shown on the following pages. For the sake of clarity, the translation classes specified to ASM/GEN are based on the 7 different addressing modes of the Motorola 6800 microprocessor (with accumulator and implied addressing combined into class 1, and with immediate addressing broken down into 2 translation classes: class 4 for 1-byte operands, class 5 for 2-byte operands). In the last chapter, a tutorial is presented for generating the simulator. The intention of this section will be to show the reader what the generated simulator looks like.

The subroutine shown is assembled using the generated assembler and is taken from (3). It is entered with the Index Register, IX, containing the address of the most significant byte of the multiplicand. Register A contains the most significant byte of the multiplier and register B the least significant byte of the multiplier. The multiplicand and multiplier are treated as 16-bit unsigned numbers. A 16-bit product is generated in A and B. If the product is larger than 16 bits, only the least significant bits are retained.

Algorithm used is as follows:

Initial partial product (PP) = multiplier.

Repeat the following 16 times:

MULT10: Shift left, arithmetic, PP.

```
**********************************
*                                *
*    ASM/GEN:  VERSION 5.1       *
*                                *
*       GENERATOR  RUN           *
*                                *
**********************************
```

THE GENERATED ASSEMBLER HAS 7 CLASSES AND A 8 BIT WORD SIZE

OBJECT WORD SPECIFICATIONS...TRANSLATION CLASS: 1 NO. OF FIELDS: 1 NO. OF MEMORY WORDS: 1
RIGHT-MOST BIT POSITION AND WIDTH OF EACH FIELD:
1. (0, 8)

OBJECT WORD SPECIFICATIONS...TRANSLATION CLASS: 2 NO. OF FIELDS: 2 NO. OF MEMORY WORDS: 2
RIGHT-MOST BIT POSITION AND WIDTH OF EACH FIELD:
1. (8, 8) 2. (0, 8)

OBJECT WORD SPECIFICATIONS...TRANSLATION CLASS: 3 NO. OF FIELDS: 2 NO. OF MEMORY WORDS: 2
RIGHT-MOST BIT POSITION AND WIDTH OF EACH FIELD:
1. (8, 8) 2. (0, 8)

OBJECT WORD SPECIFICATIONS...TRANSLATION CLASS: 4 NO. OF FIELDS: 2 NO. OF MEMORY WORDS: 2
RIGHT-MOST BIT POSITION AND WIDTH OF EACH FIELD:
1. (8, 8) 2. (0, 8)

OBJECT WORD SPECIFICATIONS...TRANSLATION CLASS: 5 NO. OF FIELDS: 2 NO. OF MEMORY WORDS: 3
RIGHT-MOST BIT POSITION AND WIDTH OF EACH FIELD:
1. (16, 8) 2. (0,16)

OBJECT WORD SPECIFICATIONS...TRANSLATION CLASS: 6 NO. OF FIELDS: 2 NO. OF MEMORY WORDS: 2
RIGHT-MOST BIT POSITION AND WIDTH OF EACH FIELD:
1. (8, 8) 2. (0, 8)

OBJECT WORD SPECIFICATIONS...TRANSLATION CLASS: 7 NO. OF FIELDS: 2 NO. OF MEMORY WORDS: 3
RIGHT-MOST BIT POSITION AND WIDTH OF EACH FIELD:
1. (16, 8) 2. (0,16)

GENERAL MNEMONICS

NONE DEFINED

TRANSLATION CLASS 1 MNEMONICS

INX	8	ABA	1B	ASRA	47	ASRB	57
RORA	46	CBA	11	SBA	10	TAP	6
TXS	35	NEGA	40	NEGB	50	CLRA	4F

CLRB	5F	DAA	19	TBA	17	TPA	7
RTS	39	CLC	C	TAB	16	SEC	D
DECA	4A	DECB	5A	DES	34	PSHA	36
PSHB	37	TSTA	4D	TSTB	5D	RTI	3B
SEI	F	CLV	A	SEV	B	INCA	4C
INCB	5C	PULA	32	PULB	33	CLI	E
DEX	9	INS	31	SWI	3F	ASLA	48
ASLB	58	TSX	30	ROLA	49	ROLB	59
LSRA	44	LSRB	54	NOP	2	COMA	43
COMB	53	WAI	3E	RORB	56		

TRANSLATION CLASS 2 MNEMONICS

BRA	20	BEQ	27	BSR	8D	BCC	24
BCS	25	BVC	28	BLS	23	BVS	29
BGT	2E	BLT	2D	BGE	2C	BLE	2F
BNE	26	BHI	22	BMI	2B	BPL	2A

TRANSLATION CLASS 3 MNEMONICS

LDABD	D6	LDXD	DE	ORAAD	9A	ORABD	DA
CPXD	9C	SBCAD	92	SBCBD	D2	CMPAD	91
CMPBD	D1	ADCAD	99	ADCBD	D9	STAAD	97
STABD	D7	ADDAD	9B	ADDBD	DB	ANDAD	94
ANDBD	D4	STXD	DF	SUBAD	90	SUBBD	D0
BITAD	95	BITBD	D5	EORAD	98	EORBD	D8
LDSD	9E	STSD	9F	LDAAD	96		

TRANSLATION CLASS 4 MNEMONICS

SBCAI	82	SBCBI	C2	BITAI	85	BITBI	C5
ORABI	CA	LDABI	C6	EORAI	88	EORBI	C8
ORAAI	8A	LDAAI	86	ADDAI	8B	CMPAI	81
CMPBI	C1	ADCAI	89	ADCBI	C9	ADDBI	CB
ANDAI	84	ANDBI	C4	SUBAI	80	SUBBI	C0

TRANSLATION CLASS 5 MNEMONICS

LDXI	CE	CPXI	8C	LDSI	8E

TRANSLATION CLASS 6 MNEMONICS

EORAX	A8	EORBX	E8	ASRX	67	LDXX	EE
RORX	66	NEGX	60	CLRX	6F	INCX	6C
STSX	AF	DECX	6A	LDAAX	A6	LDABX	E6
TSTX	6D	JMPX	6E	ORAAX	AA	ORABX	EA
CPXX	AC	STXX	EF	JSRX	AD	ASLX	68
CMPAX	A1	CMPBX	E1	ANDAX	A9	ADCBX	E9
ROLX	69	LSRX	64	STAAX	A7	STABX	E7
ADDAX	AB	ADDBX	EB	LDSX	AE	ANDAX	A4
ANDBX	E4	SUBAX	A0	COMX	63	SBCAX	A2
SBCBX	E2	BITAX	A5	BITBX	E5		

TRANSLATION CLASS 7 MNEMONICS

STXE	FF	DECE	7A	ORABE	FA	LDAAE	96
LDABE	F6	RORE	76	STAAE	97	NEGE	70
CLRE	7F	STSE	9F	ORAAE	BA	ANDAE	99
ADCBE	F9	STABE	F7	TSTE	70	ASRE	77
SUBBE	F0	JMPE	7E	CMPAE	B1	CMPBE	F1
ROLE	79	LSRE	74	CPXE	BC	LDSE	9E
INCE	7C	ADDAE	BB	ADDBE	FB	ANDAE	B4
ANDBE	F4	JSRE	BD	SUBAE	B0	ASLE	78
SBCAE	B2	SBCBE	F2	BITAE	B5	BITBE	F5
EORAE	B8	EORBE	F8	COME	73	LDXE	FE

```
 1    0                    :
 2    0                    :     MULT16 - MULTIPLY 16 BIT UNSIGNED NUMBERS.
 3    0                    :
 4    0                    :     ENTRY .. (ACCA) = M.S.B. OF MULTIPLIER,
 5    0                    :              (ACCB) = L.S.B. OF MULTIPLIER.
 6    0                    :              (IX) = ADDRESS OF M.S.B OF MULTIPLICAND.
 7    0                    :
 8    0                    :     EXIT  .. (ACCA) = M.S.B OF 16-BIT PRODUCT,
 9    0                    :              (ACCB) = L.S.B. OF 16-BIT PRODUCT.
10    0                    :
11    0                        *GREF
12    0                        *TITLE MULT16
13   80      80               *DS     128       :1ST MEMORY SEGMENT (RAM)
14   80      80               *BEGIN  80*16
15   80       4               *TRACE  4
16   80      37        MULT:  PSHB              :KEEP ...
17   81      36               PSHA              :     MULTIPLIER
18   82      A6               LDAAX   1         :KEEP ...
19   83       1
20   84      36               PSHA
21   85      A6               LDAAX   0
22   86       0
23   87      36               PSHA              :MULTIPLICAND
24   88      86               LDAAI   16        :PUT COUNT
25   89      10
26   8A      36               PSHA
27   8B      30               TSX               :STACK NOW READY
28   8C      A6               LDAAX   3         :ACCA := MSB OF MULTIPLIER
29   8D       3
30   8E      58        MULT10: ASLB              :ARITH. LEFT SHIFT
31   8F      49               ROLA              :       THE MULTIPLIER
32   90      68               ASLX    2         :ARITH. LEFT SHIFT MULTIPLICAND,
33   91       2
34   92      69               ROLX    1         :       (M.S. BIT NOW IN -C-)
35   93       1
36   94      24               BCC     MULT20-PC-2       :BRANCH IF CARRY CLEAR
37   95       4
38   96      EB               ADDBX   4         :ADD MULTIPLIER TO
39   97       4
40   98      AB               ADDAX   3         :    PARTIAL PRODUCT
41   99       3
42   9A      6A        MULT20: DECX    0         :DECREMENT COUNTER
43   9B       0
44   9C      2E               BGT     MULT10-PC-2       :RETURN FOR MORE
45   9D      F0
46   9E      31               INS
47   9F      31               INS               :CLEAR
48   A0      31               INS               :    UP
49   A1      31               INS               :      THE
50   A2      31               INS               :        STACK
51   A3      39               RTS               :RETURN
52   A4                       *END
```

********** SYMBOLIC REFERENCE TABLE **********

SYMBOL	TYPE	VALUE
MULT10	LABEL	8E
MULT20	LABEL	9A
MULT	LABEL	80
PC	LABEL	A4

Examine next bit of multiplicand (starting with most significant bit).

If 0, branch to MULT10.

Add multiplier to PP.

Branch to MULT10.

To insure that "entry" into the subroutine is proper, a *TRACE 4 directive is inserted at the first executable instruction at assembly time (in our case of the Motorola 6800, this is at location 128 decimal = 80 hexadecimal = first ROM word). The following conditions must be satisfied:

(1) the stack pointer (SP) must point to a RAM location as the next available position on the stack;

(2) the return address must be kept on the stack with the high order byte above the low order byte (a 16-bit address);

(3) multiplier and multiplicand test values must be established using the change command;

The return address (arbitrarily we select the subroutine itself) is stored in locations 126 and 127 decimal (7E and 7F hexadecimal), so the stack pointer must point to 125 (7D hexadecimal). When execution reaches location 8E, we wish to view the status of the stack and the contents of the stack pointer (SP) and index register (SR2, special register 2 as declared to SIM/GEN). So a breakpoint is inserted. The last breakpoint will merely call for a level 2 machine status dump and resume the execution. This status dump will display all the special registers including the A and B accumulators, which were declared as special registers 0 and 1 respectively. Note that this subroutine uses

```
*TRACE 4  (DEBUG)
BREAKPOINT NO. 1 AT:       80
C SP 7D
SP
MEMORY BASED STACK POINTER=       7D
SR2
SREG 2=        0
C 1 11
D 0 1
 LOCATION  CONTENTS
        0         0
        1        11
C SR1 A3
SR0
SREG 0=        0
SR1
SREG 1=       A3
* 8E 4
BREAKPOINT INSERTED.
* 9E 2
BREAKPOINT INSERTED.
L
 NO.  BREAKPOINT  *TRACE LEV.
  1.          80       4
  2.          8E       4
  3.          9E       2
G
*TRACE 4  (DEBUG)
BREAKPOINT NO. 2 AT:       8E
SP
MEMORY BASED STACK POINTER=       78
D 77 7F
 LOCATION  CONTENTS
       77         0
       78         0
       79        10
       7A         0
       7B        11
       7C         0
       7D        A3
       7E         0
       7F         0
C 7F 80
D 7F
 LOCATION  CONTENTS
       7F        80
/
BREAKPOINT AT      8E REMOVED.
G
*TRACE 2  (PC+REGISTERS)

.PROGRAM COUNTER=       9E
LAST INSTRUCTION=       F0

(SPECIAL REGISTERS)
SREG 0=        A
SREG 1=       D3
SREG 2=       79
SREG 3=        0
SREG 4=        0
SREG 5=        0
SREG 6=        1
SREG 7=        0
SREG 8=        0
 RETURN FROM SUBROUTINE #
*TRACE 4  (DEBUG)
BREAKPOINT NO. 1 AT:       80
S
 STOP
COMMAND-
```

the memory-based stack as a work area (5 bytes following the return address).

2.1 The generating processes

This chapter is concerned with the generation process of ASM/GEN and SIM/GEN. It is best to begin the description of the generation process by directly quoting the original authors:

> The actual program to be generated was written as a "skeleton routine", that is, the program was complete except for the constants defining the target machine's architectural dimensions and particulars that are "filled-in" from the user input. Within the skeleton are "markers" indicating that user input is required to complete a missing part at the marked point. The generating process then becomes a matter of first copying the skeleton to some disk file, and then having the generator transfer the skeleton to another file, card by card, filling in the missing parts with user-provided data at the appropriate places. The latter file would contain the complete module upon termination. Diagnostics are included to aid in correction of syntax errors, however, cases of misformatted values can result in a complete module with incorrect machine specifications.

The generating process therefore consists of a single, strictly sequential pass over the associated skeleton units, inserting the appropriate FORTRAN code to the resulting file between "markers". An essential characteristic of the generators is that it is always aware of the context in which the generation process is in. This is largely dependent on the ordering of the routines within the skeleton unit. In point of fact, the program structure of the GEN systems reflects this to a very high degree.

Another characteristic is the separation of the target machine specifics (termed by the original authors as the variants) from the machine-independent aspects of the generated program (invariants). The

variants are user input at generation time while the invariants reside on the skeleton units. Only those routines within the skeleton unit that are target-machine dependent in some aspect need be "marked", signifying to the generator that some insertion (or in some cases, a skip) into the current routine is required. It may come in the form of a COMMON statement, or a DATA statement (as is frequently the case), or several lines of FORTRAN statements (typically, assignment or computed go-to statements). A dollar sign '$' is used for a marker.

2.2 The ASM/GEN generating process

There are 2 skeleton units to this generator. The first is nearly a straightforward and completely pre-written skeleton, with only the last 2 that need "filling in". The first routine is a 1-parameter subroutine (OPCODE) which in the formal view represents the semantic routine invoked by the lexical analyzer when it is sensed that the current input symbol is that of an operation code mneumonic. To be more specific, the generated subroutine OPCODE actually takes the form of a computed go-to statement in FORTRAN, each branch of which is a CALL statement to the appropriate translation class processing subroutine. Recall that the user input to the ASM/GEN system also requires a grouping scheme such that in any group (called a translation class) all instructions shall have exactly identical instruction length, operation code length, total number of operand fields, as well as operand field length. That is, an instruction may be thought of simply as being made up of a string of bit-encoded fields and generally

assumes the structure:

OPCODE OPR1 OPR2 ... OPRn

In every translation class, these fields would have identical bit-width for all instructions. Hence, all that needs to be known from the user are the OPCODE values and bit-widths for all fields. Returning to the 1-parameter subroutine OPCODE, we are now aware of its function: invoke the subroutine to process (i.e., extract operand fields and generate the bit string) an instruction, given the opcode value. What is actually passed is the position of the opcode mneumonic in the symbol table (which is, of course, built up during the generation). The computed go-to statement will actually be preceded by a statement that calculates the translation class number for the opcode in question, using the position parameter passed. There will be as many branches in the computed go-to as there are translation classes specified during generation. The subroutine for processing a particular translation class has yet to be generated of course, and this is where the other skeleton unit comes in.

The other routine remaining within the first skeleton unit requiring code insertion is the initializing routine which takes the form of a BLOCK DATA. Into this routine is inserted a series of DATA statements that initializes 2 tables: the first (collectively grouped under the COMMON /SYMTAB/ block) is the global symbol table to be used by the lexical analyzer for reserved symbol recognition and also for storing user-defined symbols during assembly; the second (grouped

under the COMMON /OPMAP/ block) is an opcode-to-translation-class-number mapping table for use by the subroutine OPCODE just mentioned.

The second skeleton unit constitutes the main body of a subroutine corresponding to a translation class processor (to reiterate, the process being extract operand fields and generate bit string). It is therefore "scanned over" as many times as there are specified translation classes. This is not to be confused with our earlier statement that the generating process is a single, strictly sequential pass over the skeleton unit. What must be observed is that if 7 translation classes were specified by the user, then seven translation class subroutines: CLAS1, CLAS2 ... CLAS7 must be generated. There is 1 parameter passed to each of these and is the opcode value corresponding to the opcode mneumonic, sensed by the lexical analyzer, which invoked the OPCODE subroutine (passing to it the opcode value's position in the /OPMAP/table), which in turn calculated the class number and invoked the corresponding CLAS subroutine.

The code inserted into this skeleton unit consists of first generating a subroutine header of the form SUBROUTINE CLASi, where i is the class number, followed by 3 DATA statements (at second insertion) specifying the instruction length, opcode and operand field bit-width and relative bit position. The remainder is completely identical in all translation classes except when the "field-swap" option is used. In this special instance, an extra 5 lines of code to perform the swap are included.

To summarize, the ASM/GEN generating process consists of the following steps:

(1) Read user inputs

(2) Transfer skeleton unit 1 until marker

(3) Do the following NCLAS times, where NCLAS is the number of translation classes that the user specified in his input:

(3.1) Position to start of skeleton unit 2

(3.2) Output a SUBROUTINE CLASi header, where i is the iteration count for this loop;

(3.3) Transfer skeleton unit 2 until marker;

(3.4) Output DATA statements to initialize the instruction length, opcode and operand field length and bit-width;

(3.5) Output field-swap code, if user specified it for this class; else skip over it;

(3.6) Transfer skeleton unit 2 until marker;

(4) Transfer skeleton unit 1 until marker;

(5) Output computed go-to statement with NCLAS branches, each branch being of the form:

```
i CALL CLASi (OPMAP (POS))
```

where i progresses from 1 to NCLAS;

(6) Output and END statement to complete the generation of the subroutine OPCODE;

(7) Transfer skeleton unit 1 until marker;

(8) Output COMMON /OPMAP/ statement to dimension this table,

based on the total number of opcodes; dimension will be 2 times opcode (1 for opcode value and 1 for class number for this opcode value);

(9) Transfer skeleton unit 1 until marker;

(10) Output DATA statement to initialize the symbol tables /OPMAP/ and /SYMTAB/;

(11) Output END statement to wrap up the BLOCK DATA;

In the outline above, "transfer" implies the line by line transfer from the skeleton unit to the resulting file, while "output" implies a formatted WRITE to the resulting file. The special case at (3.5) involves some 5 lines of FORTRAN code to accommodate the field-swap option for certain translation classes. The modified version selectively generates these lines in the sense that they are "skipped over" (skeleton unit is read, but not followed by a write to the resulting file) if the option is not chosen. This results in a shorter subroutine for the translation class under consideration. A pictorial view of the steps involved in the ASM/GEN generation is shown on the following page.

skeleton unit 1	resulting file		skeleton unit 2
PROGRAM ASSEM	PROGRAM ASSEM		. (FORTRAN
. . .	. . .	(2)	. declarations)
$	SUBROUTINE CLAS1	(3.2)	.
SUBROUTINE OPCODE	.		$
. . .	.	(3.3)	. (field swap code)
$	DATA bit-width		.
BLOCK DATA	. . .	(3.4)	$
. . .	(field-swap code)		. (generate bit-string code)
$	. . .	(3.5)	.
. . .	RETURN		
$	END	(3.6)	
DATA	. . .		
	SUBROUTINE OPCODE		
	. . .	(4)	
	GO TO (1,2, ...)		
	. . .	(5)	
	END	(6)	
	BLOCK DATA		
	. . .	(7)	
	COMMON /OPMAP/...	(8)	
	. . .	(9)	
	DATA OPMAP		
	DATA SYMTAB	(10)	
	END	(11)	

2.3 The SIM/GEN generating process

The generating process for a complete simulator involves at least 3 different steps and is quite different from ASM/GEN. Following the top-down modular approach, SIM/GEN's generation process is again best described by the original author's statements:

> The operation of all modern general-purpose digital computers is based on the repetitive sequence of FETCHing the next instruction, DECODEing it, and invoking the proper operations that EXECUTE the instruction. The initial goal of generating a simulator was divided into three subtasks: Those of generating a FETCH module, a DECODE module, and an EXECUTE module. The FETCH and DECODE are both well-defined and small enough to not have to be broken down further. EXECUTE, however, spans a wide variety of tasks and appeared to be far too extensive to be handled by a single unit.
>
> The technique to treat that EXECUTE phase was to group the instruction set into classes in which all the instructions in a particular class matched identically in their component bit structures. That is, all instructions with an opcode of length 1, operand field one of length j, operand field two of length k, etc., and which all have m operand fields would be grouped in a single class. The DECODE routine could determine which class the instruction belonged in, branch to that module, and then decoding of the operand parts could be done at the start of the module without regard to which instruction in the module was being executed. This seemed to be an effective solution and required that the user define only one module at a time, totally independent of the remainder of the system.

In the current version of SIM/GEN, these 3 subtasks have been labelled the MEMORY, DECODE and XECUTE modules, with MEMORY having the identical function of the FETCH module described above (from an earlier version). The basic philosophy behind the generating process

remains the same: transferring the skeleton unit to the resulting file, taking the appropriate action between "markers" in the form of code insertion & deletion (2.1).

In the remainder of this chapter, the term insertion will always be taken to mean the outputting of FORTRAN code to the resulting file. Insertion has already been used at the ASM/GEN generation and is always assumed to occur at the skeleton unit markers ('$' is used throughout). Deletion will be taken to mean the action of "skipping over" portions of the skeleton unit. That is, a read of the skeleton unit not followed by a write (transfer) to the resulting file. Quite often, the word 'skip' will also be used in this context. Deletion is used where the skeleton unit contains segments of code that are mutually exclusive, the segment to be chosen being dependent entirely on the user input specifications. As an example, consider the fact that one of the routines present in the skeleton unit of the MEMORY module manipulates the hardwired stack. If the simulated microprocessor does not have this option, there would be no need to include this routine in the resulting file. A skip over this segment of code is therefore necessary at the point where the stack utilities are generated.

2.4 The MEMORY module generator

Target machine dependent routines are generated by the MEMORY module generator. These routines involve memory references, data transfers to and from memory assuming a bus organization, memory addressing, stack manipulation and machine status displays. The

skeleton unit for this module generator consists of the following routines:

(1) RDMEM - function to read contents of memory at address bus to the data bus.

(2) WRMEM - subroutine to write contents of the data bus into memory specified by the address bus.

(3) ROM - function to check for memory type.

(4) VIRMEM - virtual address mapping function.

(5) MEMDMP - display memory contents subroutine.

(6) PUSH - push data onto stack routine.

(7) PULL - pull data off stack routine.

(8) GETSTK - return data off hardwired stack routine.

(9) STATUS - machine status display routine.

(10) DGREG - display general register routine.

(11) DSREG - display special register routine.

(12) DSTK - display hardwired stack routine.

(13) CHANGE - alter memory or hardware component routine.

(14) IHEX, BINSER, XOR - display utilities routines.

(15) BLOCK DATA - initializing routine.

Routines (1)-(5) deal with memory references and data transfers. The insertions performed on these routines are the memory dimensions, memory word size, memory segment type and memory segment boundaries. These come in the form of COMMON statements for the MEMORY array, which must be dimensioned to the total number of words that comprise the

simulated memory configuration, and DATA statements for the memory segment boundaries and memory type for each segment. Only FORTRAN declaratives are inserted into these 5 routines. The exception to this is the ROM function. If no Read Only Memory segments were declared, the entire body of the routine is skipped, and replaced by a single assignment statement: ROM = .FALSE.

Routines (6)-(8) are stack manipulation routines. A three-way decision is made by the generator at this point. If no stack facility is declared, these 3 routines are skipped. If a memory-based stack is declared, the first 2 are generated, but the third skipped, and if a hardwired stack is declared, all 3 routines are generated. Code insertion involves either a memory data transfer sequence, or a hard-wired stack data transfer. In the former case, the PUSH routine uses the bus structure to store data onto the stack (which is memory based) and so the following code is inserted:

```
data bus     = word to be pushed (parameter)
address bus = stack pointer
CALL WRMEM.
```

If a hardwired stack was declared, the code insertion first requires that a COMMON statement be generated to allocate a separate area of storage (named STACK) for the hardwired stack. It is naturally dimensioned to the stack depth which must be specified by the user. Then the simple assignment statement

STACK (stack pointer) = word to be pushed (parameter)

is inserted. In both cases, the decrementing of the stack pointer

following the actual push operation is part of the skeleton routine. In fact, that line of code immediately follows the marker where the PUSH code is inserted. A similar situation exists for the PULL routine.

Routines (9) to (13) involve combined insertions and deletions. Insertions are entirely the FORTRAN declaratives that dimension the register arrays, the memory array and several other counters (i.e., total number of special registers). The deletions are mainly concerned with the display function at the interactive level. For example, the body of the DSTK subroutine contains 2 segments of code: a first segment loops through the hardwired stack, converting each location to display format, and displays it. The other segment is merely a WRITE statement with a diagnostic message that no hardwired stack was declared for the simulated machine. Depending on the user declaration, the appropriate segment is skipped during the generation. A similar situation exists for the DGREG and DSREG routines. The 3 routines at (14) are print utilities, while the last (15), is used to initialize all the microprocessor's architectural properties in the form of counters, flags, and arrays. This completes our discussion of the MEMORY module generator.

2.5 The DECODE module generator

Generated by the DECODE module generator are:

(1) LOADRM - absolute loader to read load file created by the generated assembler and begin execution.

(2) FETCH, OPRDEC - instruction fetch routines.

(3) TRACE - trace manager that checks for occurrences of *TRACEd locations in the simulated program.

(4) GETKH, GETSYM, TYPSYM, EQL, GETHEX, NUMER, NUMERH, SORT, XCHANG - utilities required to interactively communicate with a user at a terminal. These routines were additions in the modified SIM/GEN and have to do with the command processing for a *TRACE 4 breakpoint.

(5) DECODE - routine to extract instruction opcode and branch to the proper execution class.

Code insertion occurs only at one point in the skeleton unit and is at the subroutine DECODE. This parallels the 2 routines in the first skeleton unit of the ASM/GEN system: OPCODE and BLOCK DATA. As a matter of fact, they are identical: the table generated by the DECODE module is also an opcode-to-execution class mapping function. This table is used to calculate the execution class (translation class equivalent of SIM/GEN) number after which the branch to the proper execution class is taken. Like subroutine OPCODE of the ASM/GEN skeleton unit, a computed go-to statement is generated, each branch of which is a call to the execution class processor (as against translation class processor in ASM/GEN). It is in the execution class processing subroutine where the simulation of the fetched instruction takes place. These subroutines are generated by the third and last module generator of the SIM/GEN system -- the XECUTE module.

Between the code insertion for the opcode table and the generated computed go-to, are also inserted several lines of FORTRAN code that first extract the opcode from the instruction register. Different lines of code will appear, depending on whether the simulated machine has fixed-size or variable-size operation codes. In any case the extract code is followed by a calculation of the execution class number (using a binary search routine BINSER, generated in the MEMORY module, on the inserted opcode table). This is then followed by the branch to the proper execution class.

2.6 The XECUTE module generator

In this last component of the SIM/GEN system are produced the emulation code for a certain execution class of instructions. As previously noted, an execution class to SIM/GEN is what a translation class is to ASM/GEN, and both names really mean the same thing. What is quite different to the generators is the processing that follows: ASM/GEN generates the necessary code to produce the bit-string for the assembled instruction, while SIM/GEN must generate the code to simulate the instruction. The latter task is certainly a more complicated one. An Instruction Definition Language (IDL) was designed to allow the user to describe a microprocessor instruction set to SIM/GEN. An IDL processor, which is part of the XECUTE module generator, translates the user's IDL statements into the equivalent FORTRAN statements. We quote from the SIM/GEN user's reference manual:

"IDL is an assembly-like language consisting of microlevel operators and architectural component operands which enable the user to emulate the functions of a microprocessor's instruction set. It can be viewed as a simulator microprogramming language in that it provides a convenient medium for specifying the microinstructions whose results define the function of the machine instruction. Its operations represent those typically found in the functional unit(s) of microcomputer Arithmetic-Logic Units in addition to simple data transfers. The operands offered are typical storage elements, some with a dedicated duty (such as the Program Counter and Stack Pointer Register) and others more flexible and general-purpose such as General Registers, Address and Data Busses and Memory."

"The power of using IDL for instruction microprogramming lies in the high degree of similarity between its notations and those found in a representative vendor microprocessor description manual. It, therefore, allows the user to readily transfer the vendor's description of the instruction to SIM/GEN which greatly simplifies the process. IDL descriptions are totally sequential, with a conditional IF construct provided for conditional execution of blocks of IDL statements. Subroutines may be defined to eliminate the need for coding redundant functions common to groups of machine instructions (e.g., status bit settings on Arithmetic Logic instructions, address computations for external referencing, etc.)."

The skeleton unit to the XECUTE module generator parallels the second skeleton unit to ASM/GEN. A difference exists at the processing level of the generators: whereas ASM/GEN produces the translation class subroutines in a single run, the XECUTE module generator of SIM/GEN can only produce one execution class subroutine per run. So if n execution classes are involved, n runs of XECUTE are necessary to complete the generation. The XECUTE processing consists of the following steps:

(1) Output SUBROUTINE CLASi header, where i is the execution

class number in question.

(2) Transfer skeleton unit until marker.

(3) Insert user-input dependent declaratives (the only ones being general and special register count, which are part of the input to XECUTE).

(4) If this execution class has no operand fields, skip until marker is encountered. Otherwise, transfer skeleton unit until marker and generate the index calculation code.

(5) Generate a computed go-to statement with m branches, if there are m instructions in this class.

(6) Process instruction definitions.

(7) Process subroutine definitions.

The code that is either skipped over or transferred to the resulting file at step (4) is the operand extraction code. An assumption here is that the generated execution class subroutines are entered only with the opcode fetched into the instruction register. So operand fields must first be extracted. This is followed by the index calculation code for the instruction to be simulated. The branch to the simulation code for that instruction is then taken by way of the computed go-to statement using the earlier calculated index. When all instructions have been processed, user-defined subroutines are checked and processed similarly by the IDL processor. Each user-defined subroutine will result in exactly 1 FORTRAN subroutine. A diagram of the XECUTE module generation is shown below:

XECUTE skeleton module	resulting file	
	SUBROUTINE CLASi	(1)
. . .	.	
(FORTRAN declarations)	.	(2)
. . .	.	
$	(FORTRAN declarations to	
	dimension register arrays)	(3)
	.	
(extract operand code)	.	
	.	(4)
	.	
	(calculate instruction index code)	
	GO TO (2, 3, ... m) index	(5)
	. (instruction emulation	
	. code generated by the	(6)
	. IDL processor)	
	END	
	(user-defined subroutines	
	in FORTRAN equivalent)	(7)

2.7 The IDL processor

Central to the simulator generating process is the generation of FORTRAN statements that emulate a microprocessor instruction. Since user input consists of IDL statements, an IDL syntax recognizer and translator is required. This section will present the generation techniques involved in the IDL processing.

The first component to the IDL processor comes in the form of the scanner for IDL operands (note, at this point, that IDL operations are recognized separately and at another higher logical level), implemented as subroutine OPSCAN in the XECUTE generator program GENXEC. When called, OPSCAN will translate an input token, assumed to be an IDL operand, into its FORTRAN form. It is worth recalling that the card by card image processing assumption simplifies the task somewhat. The following table shows the generated outputs corresponding to the available IDL operands:

IDL operand	generated output	
ABUS	ABUS	(address bus)
DBUS	DBUS	(data bus)
GREGi	GREG (i + 1)	(general register i)
GREG.i	GREG (OPR(i+1))	(general register number at operand field i)
IMMi	OPR (i + 1)	(immediate operand field i)
PC	PC	(program counter)
STACKP	STACKP	(stack pointer)
STACK	STACK (STACKP)	(top of hardwired stack)
TEMPi	TEMP (i + 1)	(temporary register i)
SREGi	SREG (i + 1)	(special register i)

There is a current character pointer into the card image being processed, and is moved forward or backward depending on the processing

stage involved. The IDL operator recognizer is simply a checker for those keywords that indicate an IDL operation. Every IDL statement begins with an IDL operator, such as MOVE, ADD, IF, etc. (see page 109, manual, for a complete list).

Consider the simple IDL statement

```
ADD   SREG0   SREG0   DBUS.
```

Assume now that the card image pointer has been left at the blank following 'ADD', so the operator has been sensed at some stage in the processing. The rest of the processing is as follows:

card pointer	IDL processor action	generated output
after 'ADD'	CALL OPSCAN	SREG(1)
after 1st 'SREG0'	emit '='	=
	CALL OPSCAN	SREG(1)
after 2nd 'SREG0'	emit '+'	+
	CALL OPSCAN	DBUS

The generated FORTRAN statement is thus

```
SREG(1) = SREG(1) + DBUS.
```

Constants are also processed by OPSCAN and converted to integer display format. The machine dependent functions like AND, OR, XOR, translate into statement function calls. In what follows, the FORTRAN implementations for each of the IDL operations are described.

CLEAR opr.

This takes the obviously simple FORTRAN statement

```
opr = 0.
```

COMONE opr1 opr2

The resulting FORTRAN statement is

```
opr1 = -opr2.
```

COMTWO oprl opr2

This has the FORTRAN statement

```
oprl = -opr2 + 1.
```

CONCAT oprl opr2 (n) opr3

This has the FORTRAN statement

```
oprl = (opr2 * (2 ** n) ) + opr3.
```

DECR opr

This has the FORTRAN statement

```
opr = opr - 1.
```

DISPLAY text

This has the FORTRAN statement

```
      WRITE (6, m)
m     FORMAT (xxH text)
```

IF oprl relop opr2

This has the FORTRAN statement

```
BOOL = oprl .relop. opr2
```

where .relop. is any one of: EQ, NE, GT, GE, LT, LE.

The subsequent IDL statements processed are of the form IF (BOOL) FORTRAN equivalent of IDL statement until an 'ENDIF' statement is encountered.

INCR opr

This has the FORTRAN statement

```
opr = opr + 1
```

MOVE oprl opr2

This has the FORTRAN statement

```
opr1 = opr2
```

PUSH opr

This has the FORTRAN statement

```
CALL  PUSH(opr)
```

PULL opr

This has the FORTRAN statement

```
CALL  PULL(opr)
```

Note that PUSH and PULL are subroutines generated by the MEMORY module generator.

READD

This has the FORTRAN statement

```
ITMP = RDMEM(X)
```

where X is some dummy argument (RDMEM is a statement function generated in MEMORY).

READI

This has the FORTRAN statements

```
ABUS = RDMEM(X)
ITMP = RDMEM(X)
```

SET opr

This has the FORTRAN statement

```
opr = 2 ** MEMSIZ - 1
```

Note that MEMSIZ is a COMMONed variable that is initialized in the BLOCK DATA generated by the MEMORY generator module.

SHLC opr1 opr2 n

This has the FORTRAN statement

```
opr1 = MOD (opr2 * (2 ** n), 2 ** MEMSIZ) + opr2/(2 ** (MEMSIZ - n) )
```

SHLL opr1 opr2 n

This has the FORTRAN statement

```
opr1 = MOD (opr2 * (2 ** n), 2 ** MEMSIZ)
```

SHRA opr1 opr2 n

This has the FORTRAN statement

```
opr1 = opr2/(2 ** n) +
   (opr2/(2**(MEMSIZ - 1))) * ((2 ** n) - 1) *
   (2 ** (MEMSIZ - n))
```

SHRL opr1 opr2 n

This has the FORTRAN statement

```
opr1 = opr2/(2 ** n)
```

WRITED

This has the FORTRAN statement

```
CALL WRMEM
```

WRITEI

This has the FORTRAN statements

```
IDUM = DBUS
ABUS = RDMEM(X)
DBUS = IDUM
CALL   WRMEM
```

In the Shift instructions, n need not be a constant; it may be any valid IDL operand. The DUMP operator becomes a CALL STATUS (3) statement, while the HALT is a STOP statement.

3.1 Introduction

This chapter will acquaint the prospective user of the SIM/GEN system with the entire simulator generating process by way of an example. A minimum configuration for the Motorola 6800 microcomputer system is chosen for this purpose. Since SIM/GEN does not take into account the I/O interface of any microcomputer system, the minimum configuration mentioned will not include such components as the Peripheral Interface Adapter or the Asynchronous Communications Interface Adapter of the Motorola 6800 family. Furthermore, certain instructions of the "interrupt" type in the instruction set of this microcomputer system will be ignored. The Motorola 6800 system to be considered will therefore consist of 128 bytes of Read/Write memory (RAM), 1024 bytes of Read Only memory (ROM), and the microprocessing unit (MPU). Frequently, reference will be made to particular pages of the SIM/GEN user's manual, so the reader is urged to have this document on hand.

To obtain a complete simulator for any microprocessor under consideration, 3 distinct and separate components of the SIM/GEN system must have been executed to successful completion. These components, called generator modules, are the MEMORY, the DECODE and the XECUTE modules. Each is associated with a user specified set of inputs, and a pre-written set of routines called a skeleton unit. SIM/GEN requires that the MEMORY and DECODE modules be run successfully at least once, while the XECUTE module must run successfully

a certain number of times depending on the number of execution classes. The details of this will be taken up later. The overall view of SIM/GEN is illustrated in Figure 1.

3.2 The Motorola 6800 microprocessor

The Motorola 6800 microprocessor (hereafter abbreviated M6800) is an 8-bit machine with 2 general purpose, 8-bit accumulators labelled ACCA (the A accumulator) and ACCB (the B accumulator). These are used to hold operands and results from arithmetic-logic operations. There are also 3 special-purpose, 16-bit registers for use by the programmer: the Program Counter (PC) contains the address of the instruction currently being executed; the Index Register (IX) is used to store data or a 16-bit memory address for the indexed mode of addressing; the Stack Pointer (SP) points to a memory location that forms the "top" of a pushdown/pop-up store. In the case of the M6800, this is an area of memory set aside by the programmer for use as a stack. Normally, this must be a random access (Read/Write) type memory. Finally, an 8-bit Condition Code register is also available for the testing of conditions resulting from the last operation. Only the low 6 bits of this 8 bit register are used, whereas the high order two are always set to ones. The testable conditions are as follows:

Bit 0 (C) - the Carry bit from bit 7 of any applicable operand result; set or cleared depending on operation.

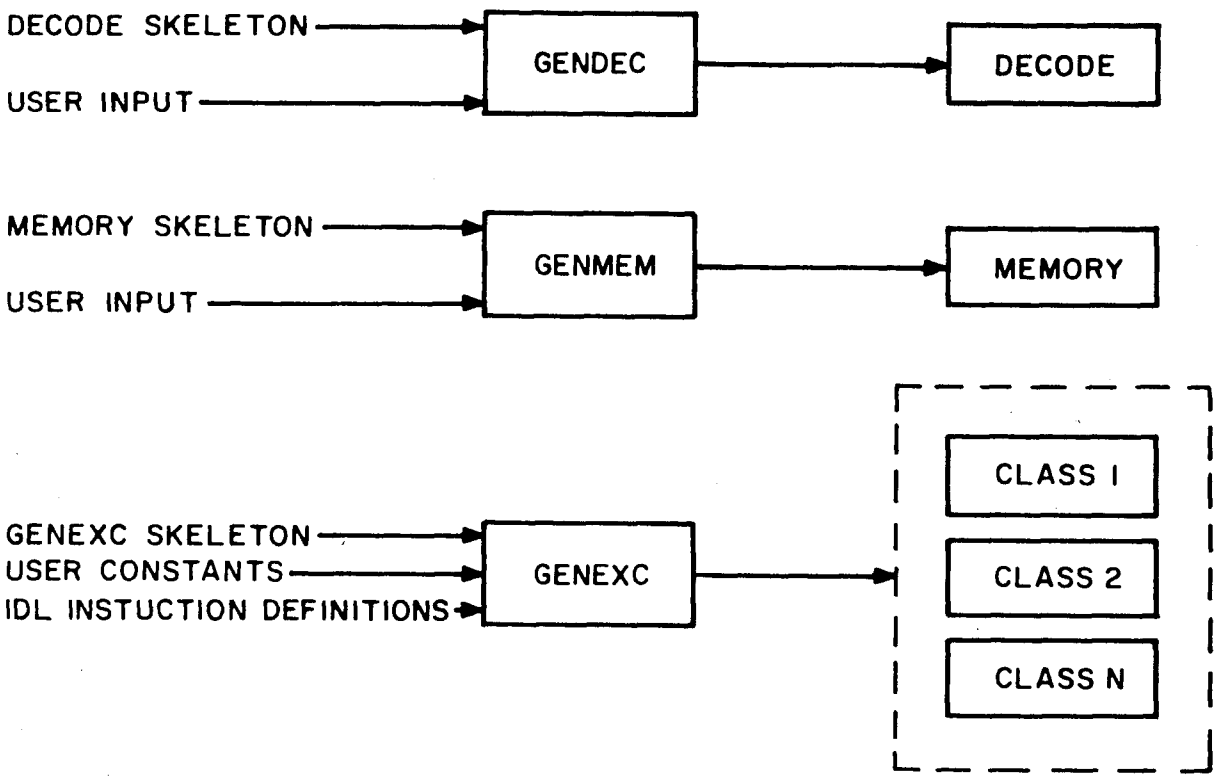


FIGURE 1.

The Simulator Generating Process (Reprinted, SIM/GEN User's Reference Manual, Version 5.3)

Bit 1 (V) - the Overflow bit; set when operation resulted in two's complement overflow, cleared otherwise.

Bit 2 (Z) - the Zero bit; set when result was zero.

Bit 3 (N) - the Negative bit;

Bit 4 (I) - the Interrupt bit; for our purposes, not much will be said of this bit since this has to do with timing.

Bit 5 (H) - the Half-carry bit from bit 3;

The setting or clearing of these bits generally depend on the type of instruction executed. Most often, arithmetic-logic type instructions affect the H, N, Z, V, & C bits. The Half-carry bit, for instance, is affected only by 3 instructions: ADDA (add accumulator with memory), ABA (add accumulators), and ADC (add accumulator with carry). The branch instructions all leave these bits unaffected. The Motorola Programming Manual contains a complete table of Boolean formulas for calculating these bits based on the operand(s) and the result. We shall have occasion to use these in a latter part of this tutorial. The architectural properties mentioned so far are summarized on Figure 2.

3.3 The Motorola 6800 Instruction Set

There are a total of 72 different instructions for the M6800, with 7 addressing modes. Included are binary and decimal arithmetic, logical, shift, rotate, load, store, branch, interrupt (ignored as far as SIM/GEN here is concerned) and stack manipulation instructions. These 7 addressing modes are somewhat arbitrary, because certain

PROGRAMMING MODEL OF THE MICROPROCESSING UNIT

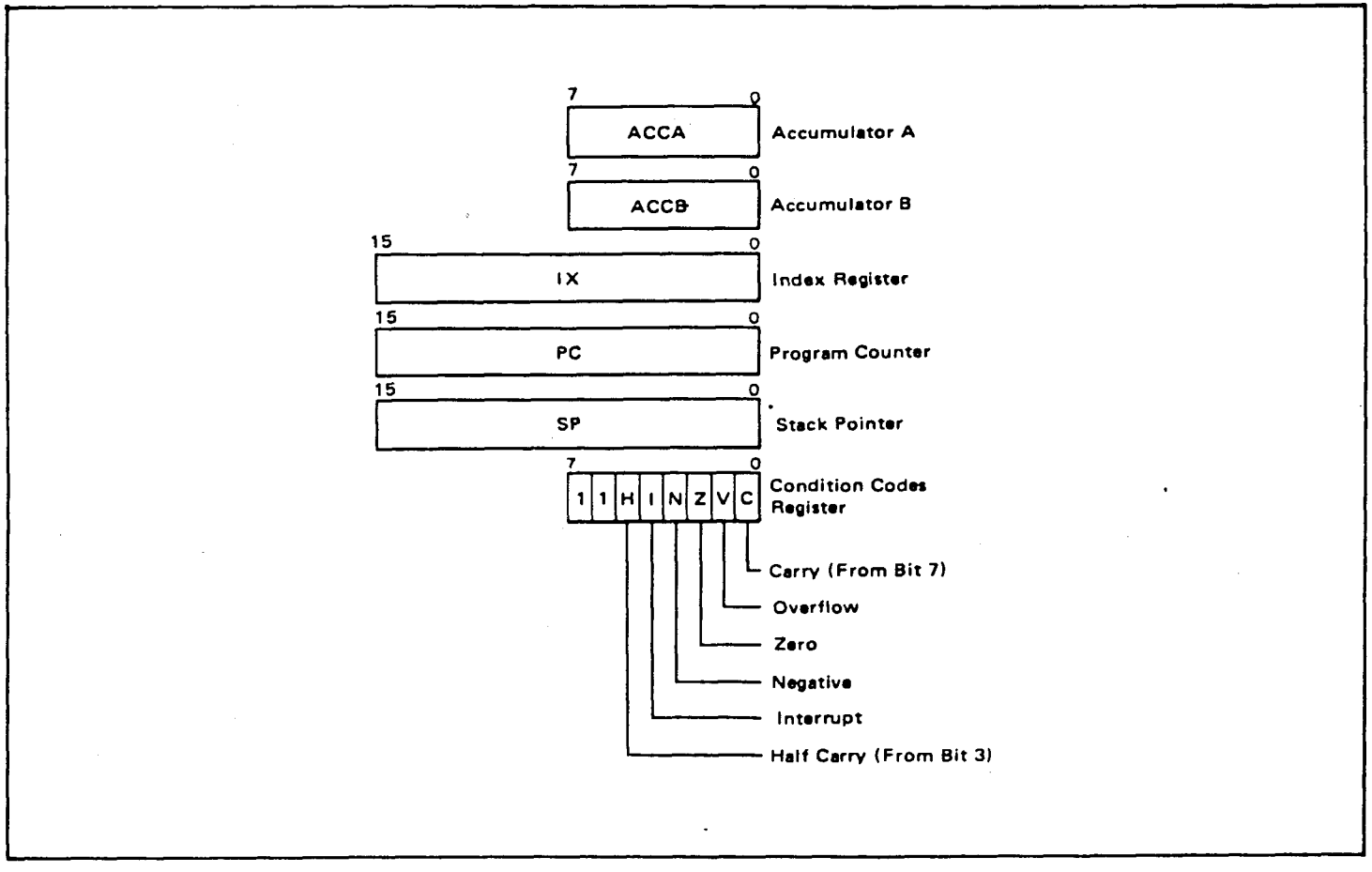


FIGURE 2. (Reprinted from M6800 Reference and Data Sheets)

instructions admit only certain addressing modes. The branch instructions, for example, admit only the relative mode and no other. The seven modes are as follows:

Accumulator addressing: In accumulator mode, either accumulator A or B is implied in the operation. These are 1-byte instructions.

Implied addressing: These are 1-byte instructions in which the operand(s) is(are) implied by opcode. These are actually similar to the Accumulator mode, except that operands other than ACCA or ACCB are implied. As an example, consider the PULA instruction (PULL data into ACCA). This is a stack manipulation instruction that will add 1 to the contents of the Stack Pointer, and load the contents of the memory location pointed to by the SP into accumulator A. Both ACCA and the stack pointer are implied in the instruction.

Immediate addressing: In this mode, the operand is contained in the second byte of the instruction, except for 3 instructions: LDS (load Stack Pointer), LDX (load Index Register) and CPX (compare Index Register). For these 3 instructions, the operand is contained in the second and third bytes of the instruction. Hence, an instruction in the immediate mode of addressing may be 2 or 3 bytes in length as noted.

Direct addressing: In direct addressing, the address of the operand is contained in the second byte of the instruction. This allows for direct addressing of the first 256 memory locations. Accordingly, enhanced execution times are achieved by storing the

most frequently used data in these locations (zero through 255). These are 2-byte instructions.

Extended addressing: When the address of the operand is greater than 255, i.e., when it is desired to address memory locations that are not among the first 256 locations, extended addressing is used. Naturally enough, these are 3-byte instructions and the address of the operand is made up of the second and third bytes of the instruction: the second byte making up the high order 8 bits and the third byte making up the low order 8 bits of the resulting 16-bit address. This represents an absolute location in memory.

Indexed addressing: In indexed addressing, the address contained in the second byte of the instruction is added to the Index Register's lowest 8 bits. The carry is then added to the high-order 8 bits of the Index Register. The result is used to address memory. Note, that this is actually adding an offset that is at most 255 in magnitude. This is an unsigned value. The Index Register is not affected when this mode of addressing is used, since the effective address resulting from the addition of the 8-bit offset is held in some temporary address register. These are also 2-byte instructions.

Relative addressing: In relative addressing, the address contained in the second byte is treated as a signed, 7-bit value. This is added to the Program Counter's lowest 8 bits, plus two. The carry or borrow is then added to the high 8 bits. This allows for addressing data within a range of -125 to +129 bytes of the present instruction.

The only instructions that admit this mode (and only this) are the branch instructions. An obvious limitation exists: if one wishes to branch on certain testable conditions (of the condition code register), one cannot directly do so if the desired location is not within the range just described. These are also 2-byte instructions.

The M6800 has fixed-size opcodes of 8 bits/opcode. Taking into account all the valid addressing modes for every instruction, there are actually 197 instructions in the instruction set of the M6800. These are summarized in Figures 3.1 through 3.4. Figure 2 is a table of the symbols used to describe the instructions on Figures 3.1-3.4. Figure 4 explains some of the special instructions. Note the condition code settings on the last column of each instruction.

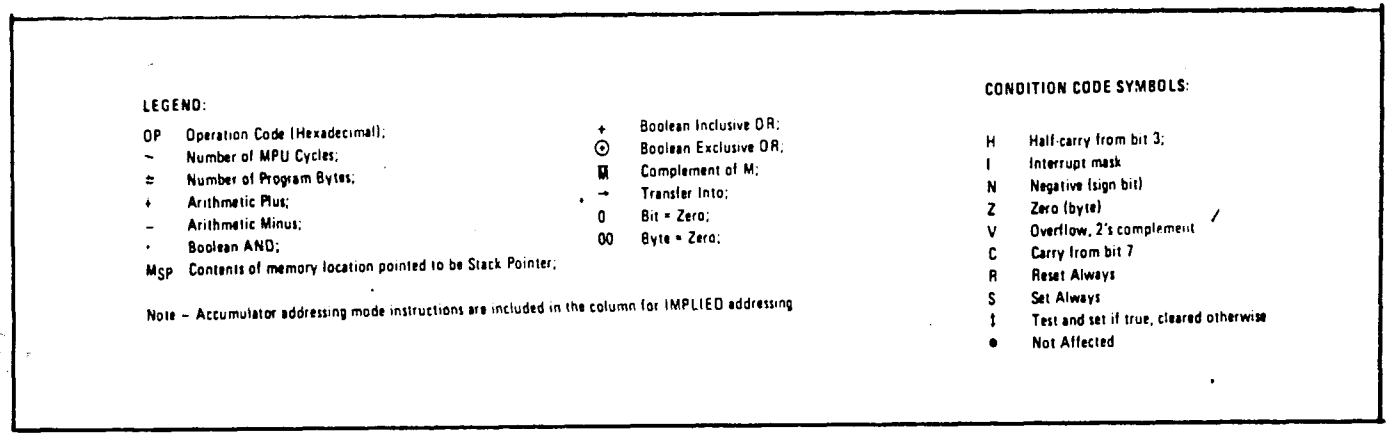
3.4 MEMORY module for the M6800

The user-required input to the MEMORY module is found on pages 16-23 of the SIM/GEN user manual. There are only 6 cards required from the user for this module, and in the case of the M6800 that we wish to generate a simulator for, our data cards will look like the following (each line represents 1 Hollerith card image):

```
0
9
8   (0,7F)   (80,47F,R)
MEM
MOTOROLA 6800
1
```

Our first card tells SIM/GEN that the M6800 has no General Registers (see page 63). In SIM/GEN usage, a general register is one capable of being addressed explicitly in an instruction. The M6800

Symbols used in FIGURES 3.1 to 3.4



LEGEND:

OP	Operation Code (Hexadecimal);
~	Number of MPU Cycles;
#	Number of Program Bytes;
+	Arithmetic Plus;
-	Arithmetic Minus;
·	Boolean AND;
M_{SP}	Contents of memory location pointed to be Stack Pointer;
+	Boolean Inclusive OR;
⊙	Boolean Exclusive OR;
M̄	Complement of M;
→	Transfer Into;
0	Bit = Zero;
00	Byte = Zero;

Note – Accumulator addressing mode instructions are included in the column for IMPLIED addressing

CONDITION CODE SYMBOLS:

H	Half-carry from bit 3;
I	Interrupt mask
N	Negative (sign bit)
Z	Zero (byte)
V	Overflow, 2's complement
C	Carry from bit 7
R	Reset Always
S	Set Always
↕	Test and set if true, cleared otherwise
•	Not Affected

FIGURE 2 (Reprinted from M6800 Reference & Data Sheets)

ACCUMULATOR AND MEMORY INSTRUCTIONS

OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED =	DIRECT OP	DIRECT ~	DIRECT =	INDEX OP	INDEX ~	INDEX =	EXTND OP	EXTND ~	EXTND =	IMPLIED OP	IMPLIED ~	IMPLIED =	BOOLEAN/ARITHMETIC OPERATION (All register labels refer to contents)	5 H	4 I	3 N	2 Z	1 V	0 C
Add	ADDA	8B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Acmltrs	ABA													1B	2	1	A + B → A	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A · M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B · M → B	•	•	↕	↕	R	•
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A · M	•	•	↕	↕	R	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B · M	•	•	↕	↕	R	•
Clear	CLR							6F	7	2	7F	6	3				00 → M	•	•	R	S	R	R
	CLRA													4F	2	1	00 → A	•	•	R	S	R	R
	CLRB													5F	2	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A − M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B − M	•	•	↕	↕	↕	↕
Compare Acmltrs	CBA													11	2	1	A − B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				$\overline{M} \rightarrow M$	•	•	↕	↕	R	S
	COMA													43	2	1	$\overline{A} \rightarrow A$	•	•	↕	↕	R	S
	COMB													53	2	1	$\overline{B} \rightarrow B$	•	•	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	7	2	70	6	3				00 − M → M	•	•	↕	↕	①	②
	NEGA													40	2	1	00 − A → A	•	•	↕	↕	①	②
	NEGB													50	2	1	00 − B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts Binary Add. of BCD Characters into BCD Format	•	•	↕	↕	↕	③
Decrement	DEC							6A	7	2	7A	6	3				M − 1 → M	•	•	↕	↕	4	•
	DECA													4A	2	1	A − 1 → A	•	•	↕	↕	4	•
	DECB													5A	2	1	B − 1 → B	•	•	↕	↕	4	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A⊕M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B⊕M → B	•	•	↕	↕	R	•
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Acmltr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	$A \rightarrow M_{SP}$, SP − 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	$B \rightarrow M_{SP}$, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	4	1	SP + 1 → SP, $M_{SP} \rightarrow A$	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, $M_{SP} \rightarrow B$	•	•	•	•	•	•
Rotate Left	ROL							69	7	2	79	6	3				M	•	•	↕	↕	⑥	↕
	ROLA													49	2	1	A	•	•	↕	↕	⑥	↕
	ROLB													59	2	1	B — C b7 b0	•	•	↕	↕	⑥	↕
Rotate Right	ROR							66	7	2	76	6	3				M	•	•	↕	↕	⑥	↕
	RORA													46	2	1	A	•	•	↕	↕	⑥	↕
	RORB													56	2	1	B — C b7 b0	•	•	↕	↕	⑥	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	•	•	↕	↕	⑥	↕
	ASLA													48	2	1	A — 0	•	•	↕	↕	⑥	↕
	ASLB													58	2	1	B — C b7 b0	•	•	↕	↕	⑥	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M	•	•	↕	↕	⑥	↕
	ASRA													47	2	1	A —	•	•	↕	↕	⑥	↕
	ASRB													57	2	1	B b7 b0 — C	•	•	↕	↕	⑥	↕
Shift Right, Logic	LSR							64	7	2	74	6	3				M	•	•	R	↕	⑥	↕
	LSRA													44	2	1	A — 0	•	•	R	↕	⑥	↕
	LSRB													54	2	1	B b7 b0 — C	•	•	R	↕	⑥	↕
Store Acmltr.	STAA				97	4	2	A7	6	2	B7	5	3				A → M	•	•	↕	↕	R	•
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A − M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B − M → B	•	•	↕	↕	↕	↕
Subtract Acmltrs.	SBA													10	2	1	A − B → A	•	•	↕	↕	↕	↕
Subtr. with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A − M − C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B − M − C → B	•	•	↕	↕	↕	↕
Transfer Acmltrs	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Test, Zero or Minus	TST							6D	7	2	7D	6	3				M − 00	•	•	↕	↕	R	R
	TSTA													4D	2	1	A − 00	•	•	↕	↕	R	R
	TSTB													5D	2	1	B − 00	•	•	↕	↕	R	R
																		H	I	N	Z	V	C

FIGURE 3.1 (Reprinted from M6800 Reference & Data Sheets)

INDEX REGISTER AND STACK MANIPULATION INSTRUCTIONS

POINTER OPERATIONS	MNEMONIC	IMMED			DIRECT			INDEX			EXTND			IMPLIED			BOOLEAN/ARITHMETIC OPERATION	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	BOOLEAN/ARITHMETIC OPERATION	H	I	N	Z	V	C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$X_H - M, X_L - (M+1)$	•	•	⑦	↕	⑦	•
Decrement Index Reg	DEX													09	4	1	$X - 1 \rightarrow X$	•	•	•	↕	•	•
Decrement Stack Pntr	DES													34	4	1	$SP - 1 \rightarrow SP$	•	•	•	•	•	•
Increment Index Reg	INX													08	4	1	$X + 1 \rightarrow X$	•	•	•	↕	•	•
Increment Stack Pntr	INS													31	4	1	$SP + 1 \rightarrow SP$	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_H, (M+1) \rightarrow X_L$	•	•	⑨	↕	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_H, (M+1) \rightarrow SP_L$	•	•	⑨	↕	R	•
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				$X_H \rightarrow M, X_L \rightarrow (M+1)$	•	•	⑨	↕	R	•
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M, SP_L \rightarrow (M+1)$	•	•	⑨	↕	R	•
Indx Reg → Stack Pntr	TXS													35	4	1	$X - 1 \rightarrow SP$	•	•	•	•	•	•
Stack Pntr → Indx Reg	TSX													30	4	1	$SP + 1 \rightarrow X$	•	•	•	•	•	•

FIGURE 3.2 (Reprinted from M6800 Reference & Data Sheets)

JUMP AND BRANCH INSTRUCTIONS

OPERATIONS	MNEMONIC	RELATIVE OP	RELATIVE ~	RELATIVE #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	IMPLIED OP	IMPLIED ~	IMPLIED #	BRANCH TEST	COND. CODE REG. 5 H	4 I	3 N	2 Z	1 V	0 C
Branch Always	BRA	20	4	2										None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	4	2										N ⊕ V = 0	•	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										Z + (N ⊕ V) = 0	•	•	•	•	•	•
Branch If Higher	BHI	22	4	2										C + Z = 0	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	4	2										Z + (N ⊕ V) = 1	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										N ⊕ V = 1	•	•	•	•	•	•
Branch If Minus	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2											•	•	•	•	•	•
Jump	JMP				6E	4	*2	7E	3	3				See Special Operations	•	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No Operation	NOP										02	2	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1		———			⑩	———	
Return From Subroutine	RTS										39	5	1		•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1	See Special Operations	•	•	•	•	•	•
Wait for Interrupt	WAI										3E	9	1		•	⑪	•	•	•	•

FIGURE 3.3 (Reprinted from Reference & Data Sheets)

CONDITION CODE REGISTER MANIPULATION INSTRUCTIONS

OPERATIONS	MNEMONIC	IMPLIED OP	~	#	BOOLEAN OPERATION	COND. CODE REG. 5 H	4 I	3 N	2 Z	1 V	0 C
Clear Carry	CLC	0C	2	1	0→C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	0→I	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	0→V	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	1→C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	1→I	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	1→V	•	•	•	•	S	•
Acmltr A→CCR	TAP	06	2	1	A→CCR	———	———	⑫	———	———	———
CCR→Acmltr A	TPA	07	2	1	CCR→A	•	•	•	•	•	•

CONDITION CODE REGISTER NOTES:

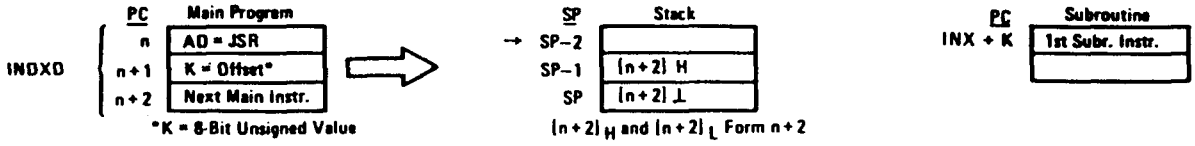
(Bit set if test is true and cleared otherwise)

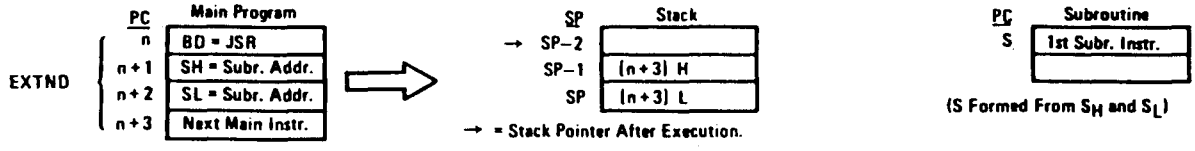
1 (Bit V) Test: Result = 10000000?
2 (Bit C) Test: Result = 00000000?
3 (Bit C) Test: Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set.)
4 (Bit V) Test: Operand = 10000000 prior to execution?
5 (Bit V) Test: Operand = 01111111 prior to execution?
6 (Bit V) Test: Set equal to result of N⊕C after shift has occurred.
7 (Bit N) Test: Sign bit of most significant (MS) byte = 1?
8 (Bit V) Test: 2's complement overflow from subtraction of MS bytes?
9 (Bit N) Test: Result less than zero? (Bit 15 = 1)
10 (All) Load Condition Code Register from Stack. (See Special Operations)
11 (Bit I) Set when interrupt occurs. If previously set, a Non-Maskable Interrupt is required to exit the wait state.
12 (All) Set according to the contents of Accumulator A.

FIGURE 3.4 (Reprinted from M6800 Reference & Data Sheets)

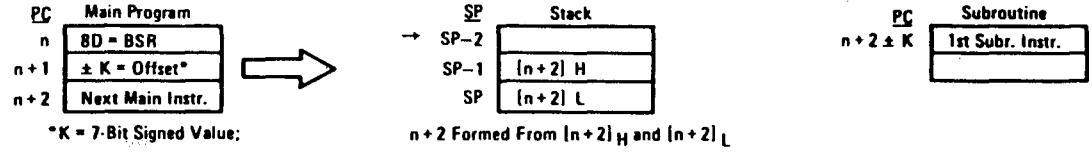
SPECIAL OPERATIONS

JSR, JUMP TO SUBROUTINE:



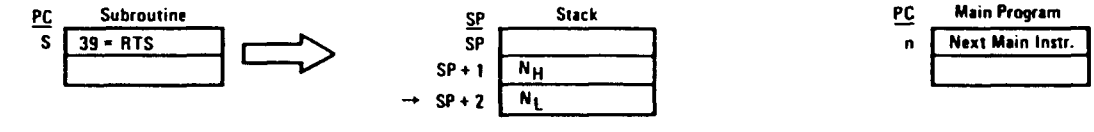


BSR, BRANCH TO SUBROUTINE:



JMP, JUMP:

RTS, RETURN FROM SUBROUTINE:



RTI, RETURN FROM INTERRUPT:

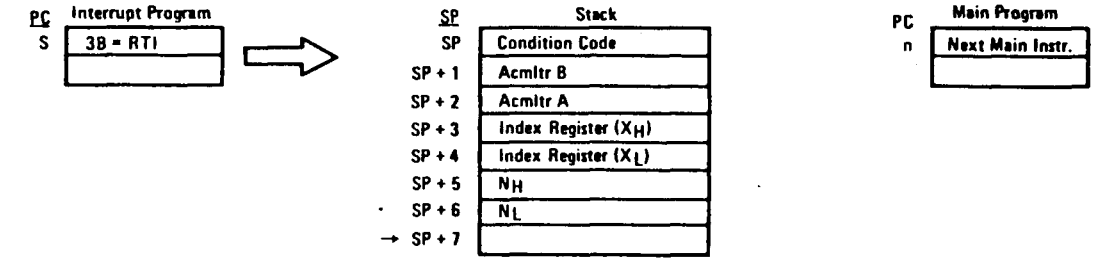


FIGURE 4 (Reprinted from Reference and Data Sheets)

has no occurrence of such a register, since the A and B accumulators are addressed solely on the basis of the opcode, and not on any extra operand field. In fact, in the accumulator mode of addressing, there are no operand fields: the one-byte opcode is sufficient. A similar case exists for those in the implied mode of addressing.

Our 9 Special Registers are assigned to SIM/GEN as follows:

SREG0 - accumulator A.

SREG1 - accumulator B.

SREG2 - Index Register.

SREG3 - Half-carry bit of the condition code register.

SREG4 - Interrupt bit of the condition code register.

SREG5 - Negative bit of the condition code register.

SREG6 - Zero bit of the condition code register.

SREG7 - Overflow bit of the condition code register.

SREG8 - Carry bit of the condition code register.

The Interrupt bit is accomodated for the sake of completeness, although we shall have no occasion to really use it for simulation. It is obvious that the condition code register was chosen <u>not</u> to be represented to SIM/GEN as a stand-alone register. This would considerably ease the instruction definitions for the XECUTE module when these bits of the condition code register would have to be set or cleared depending on the instruction. If maintained as a single Special Register to SIM/GEN, the user will face the extra burden of having to "shift-and-mask" each time a particular bit is being

considered.

The Program Counter and the Stack Pointer of the M6800 have not been declared as Special Registers since SIM/GEN has a set of operands which already include them. The names PC and STACKP have been given to registers with similar functions (see page 65).

The third data card indicates bit width of the M6800: 8 bits. The two memory segments so declared are considered the minimum for a M6800 configuration. We chose the first 128 memory locations to be a RAM type of memory, and the next 1024 to be ROM. Hence the first segment will have addresses in the range from 0 to 127 (decimal), while the next segment are in the range from 128 to 1151 (decimal). The equivalent form in hexadecimal SIM/GEN notation are shown on the third card.

The fourth data card simply indicates that our stack for the M6800 is memory-based. The fifth data card is our machine name. The sixth card indicates the number of memory words that must be fetched for every execution cycle to uniquely determine the instruction opcode. In the M6800, this is exactly 1 memory word. This completes the description of the required user input for the MEMORY module. An actual generation run for the above data set is shown on Figure 5.

3.5 DECODE module for the M6800

Pages 8-15 of the reference manual describe the user required input to the DECODE module. Our input to the DECODE module is shown

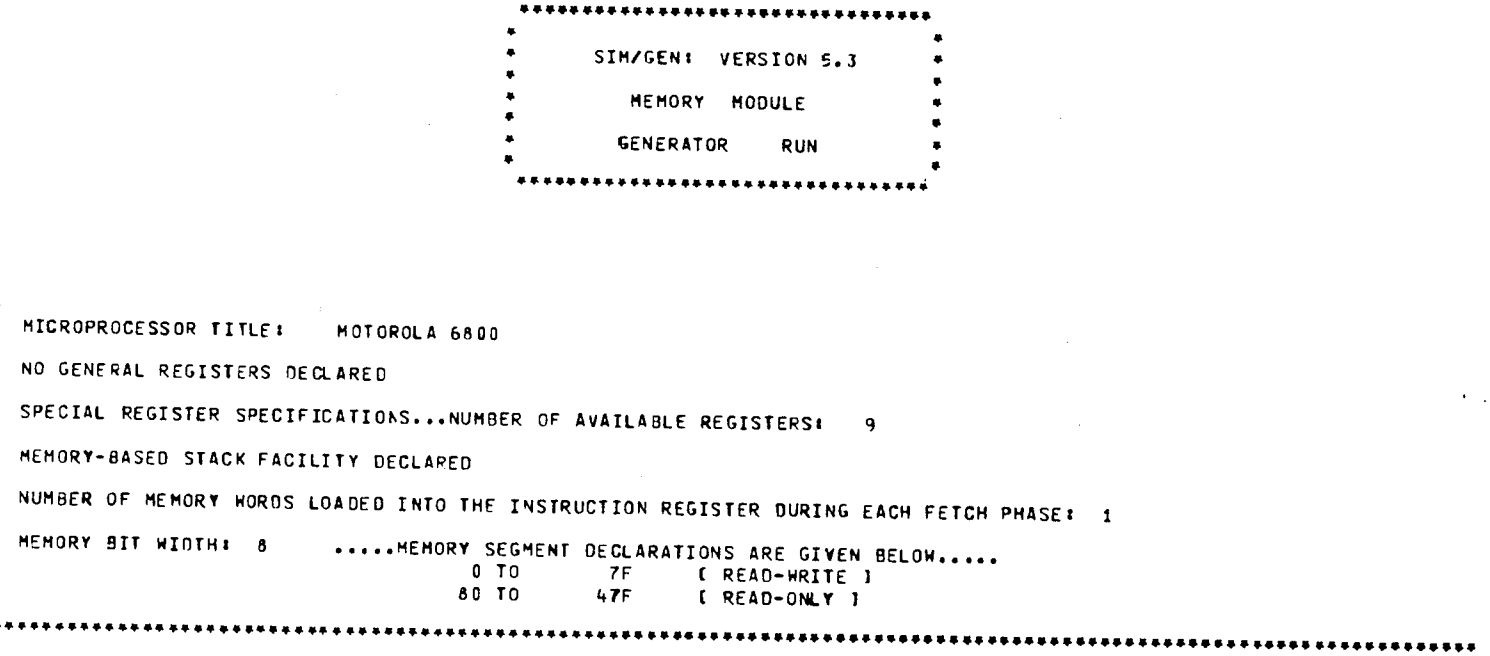


FIGURE 5 (MEMORY generation run)

on Appendix B. For uniformity, we have retained the same scheme used for specifying the translation classes in using ASM/GEN earlier in this paper. There are 7 execution classes:

Class 1: All the instructions in the accumulator and implied mode of addressing have been grouped under this class. These are simply the 1-byte instructions. There are 51 of them.

Class 2: All the instructions in the relative mode of addressing fall under this execution class. These are all the branch instructions in the instruction set (note: JSR, jump to subroutine, and JMP, unconditional jump, are not in relative mode. There is, however, nothing in SIM/GEN to prevent us from including, for instance, the JMP instruction in indexed mode in Class 2.). There are 16 such instructions.

Class 3: All instructions in the direct mode of addressing are grouped in this class. There are 27 instructions in this class.

Class 4: All 2-byte immediate mode instructions. There are 20 instructions in this execution class. Note, that the 3-byte immediate mode of addressing instructions cannot be members of this class.

Class 5: All 3-byte immediate mode instructions. There are only 3 instructions in this execution class: LDX (load IX with a 16-bit immediate operand field); CPX (compare IX with a 16-bit immediate operand field); and LDS (load Stack Pointer with a 16-bit immediate operand field).

Class 6: All indexed mode of addressing instructions. There are

40 instructions in this class.

Class 7: All extended mode of addressing instructions. There are also 40 instructions in this class and they parallel the Class 6 instructions. That is, every instruction that admits the indexed mode of addressing also admits the extended mode of addressing, and conversely.

An actual generation listing for the above classification, using the data set found in Appendix B, is shown in Figure 6 below.

3.6 Class 2 and the XECUTE module

The instructions of this execution class admit the relative mode of addressing. All leave the condition code register unaffected. Each is of the form

IF condition THEN branch to effective address,

and may be easily described to SIM/GEN using the IF statement of IDL. Each testable condition involves a check of one or more of the status bits in the condition code register. If more than 1 status bit is involved, a calculation must first be performed and the result of the calculation held in some temporary IDL operand TEMPi (i from 1 to 8, inclusive). The check is then performed against the temporary operand holding the result of calculation. The complete list of the testable conditions for each instruction is found in Figure 3.3 under the column labelled "Branch Test". The complete set of inputs to SIM/GEN for this class is shown on the following pages.

Since any instruction requires the calculation of an effective

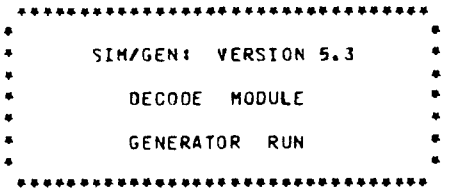


FIGURE 6 (DECODE generation run)

---------- GENERATION RUN SUMMARY ----------

THERE ARE: 7 EXECUTION CLASS MODULES

THE OPCODE BIT WIDTH IS: 8

THE LEGAL OPCODES ARE MAPPED TO THEIR EXECUTION CLASS BELOW

2	1	6	1	7	1	8	1	9	1	A	1	B	1		
C	1	D	1	E	1	F	1	10	1	11	1	16	1		
17	1	19	1	1B	1	20	2	22	2	23	2	24	2		
25	2	26	2	27	2	28	2	29	2	2A	2	2B	2		
2C	2	2D	2	2E	2	2F	2	30	1	31	1	32	1		
33	1	34	1	35	1	36	1	37	1	39	1	3B	1		
3E	1	3F	1	40	1	43	1	44	1	46	1	47	1		
48	1	49	1	4A	1	4C	1	4D	1	4F	1	50	1		
53	1	54	1	56	1	57	1	58	1	59	1	5A	1		
5C	1	5D	1	5F	1	60	6	63	6	64	6	66	6		
67	6	68	6	69	6	6A	6	6C	6	6D	6	6E	6		
6F	6	70	7	73	7	74	7	76	7	77	7	78	7		
79	7	7A	7	7C	7	7D	7	7E	7	7F	7	80	4		
81	4	82	4	84	4	85	4	86	4	88	4	89	4		
8A	4	8B	4	8C	5	8D	2	8E	5	90	3	91	3		
92	3	94	3	95	3	96	3	97	3	98	3	99	3		
9A	3	9B	3	9C	3	9E	3	9F	3	A0	6	A1	6		
A2	6	A4	6	A5	6	A6	6	A7	6	A8	6	A9	6		
AA	6	AB	6	AC	6	AD	6	AE	6	AF	6	B0	7		
B1	7	B2	7	B4	7	B5	7	B6	7	B7	7	B8	7		
B9	7	BA	7	BB	7	BC	7	BD	7	BE	7	BF	7		
C0	4	C1	4	C2	4	C4	4	C5	4	C6	4	C8	4		
C9	4	CA	4	CB	4	CE	5	D0	3	D1	3	D2	3		
D4	3	D5	3	D6	3	D7	3	D8	3	D9	3	DA	3		
DB	3	DE	3	DF	3	E0	6	E1	6	E2	6	E4	6		
E5	6	E6	6	E7	6	E8	6	E9	6	EA	6	EB	6		
EE	6	EF	6	F0	7	F1	7	F2	7	F4	7	F5	7		
F6	7	F7	7	F8	7	F9	7	FA	7	FB	7	FE	7		
FF	7														

```
    2   16    1                                                    CLS2     2
    8                                                              CLS2     3
    0                                                              CLS2     4
    9                                                              CLS2     5
BRA - BRANCH ALWAYS                                                CLS2     6
20    4.0                                                          CLS2     7
     CALL     BRNHTO   IMM1    PC.                                CLS2     8
     ENDINSTR                                                      CLS2     9
BCC - BRANCH IF CARRY CLEAR                                        CLS2    10
24    4.0                                                          CLS2    11
     IF       SREG8    EQ      0                                  CLS2    12
              CALL     BRNHTO IMM1    PC.                         CLS2    13
     ENDIF                                                         CLS2    14
     ENDINSTR                                                      CLS2    15
BCS - BRANCH IF CARRY SET                                          CLS2    16
25    4.0                                                          CLS2    17
     IF       SREG8    EQ      1                                  CLS2    18
              CALL     BRNHTO IMM1    PC.                         CLS2    19
     ENDIF                                                         CLS2    20
     ENDINSTR                                                      CLS2    21
BEQ - BRANCH IF = 0                                                CLS2    22
27    4.0                                                          CLS2    23
     IF       SREG6    EQ      1                                  CLS2    24
              CALL     BRNHTO IMM1    PC.                         CLS2    25
     ENDIF                                                         CLS2    26
     ENDINSTR                                                      CLS2    27
BGE - BRANCH IF ≥ 0                                                CLS2    28
2C    4.0                                                          CLS2    29
     XOR      TEMP2    SREG5   SREG7                              CLS2    30
     IF       TEMP2    EQ      0                                  CLS2    31
              CALL     BRNHTO IMM1    PC.                         CLS2    32
     ENDIF                                                         CLS2    33
     ENDINSTR                                                      CLS2    34
BGT - BRANCH IF > 0                                                CLS2    35
2E    4.0                                                          CLS2    36
     XOR      TEMP2    SREG5   SREG7                              CLS2    37
     OR       TEMP2    TEMP2   SREG6                              CLS2    38
     IF       TEMP2    EQ      0                                  CLS2    39
              CALL     BRNHTO IMM1    PC.                         CLS2    40
     ENDIF                                                         CLS2    41
     ENDINSTR                                                      CLS2    42
BHI - BRANCH IF HIGHER                                             CLS2    43
22    4.0                                                          CLS2    44
     OR       TEMP2    SREG8   SREG6                              CLS2    45
     IF       TEMP2    EQ      0                                  CLS2    46
              CALL     BRNHTO IMM1    PC.                         CLS2    47
     ENDIF                                                         CLS2    48
     ENDINSTR                                                      CLS2    49
BLE - BRANCH IF ≤ 0                                                CLS2    50
2F    4.0                                                          CLS2    51
     XOR      TEMP2    SREG5   SREG7                              CLS2    52
     OR       TEMP2    TEMP2   SREG6                              CLS2    53
     IF       TEMP2    EQ      1                                  CLS2    54
              CALL     BRNHTO IMM1    PC.                         CLS2    55
     ENDIF                                                         CLS2    56
     ENDINSTR                                                      CLS2    57
BLS - BRANCH IF LOWER OR SAME                                      CLS2    58
```

```
 23     4.0                                                          CLS2    59
      OR        TEMP2    SREG8  SREG6                              CLS2    60
      IF        TEMP2    EQ     1                                   CLS2    61
                CALL     BRNHTO IMM1    PC.                        CLS2    62
      ENDIF                                                        CLS2    63
      ENDINSTR                                                     CLS2    64
 BLT - BRANCH IF < 0                                               CLS2    65
 2D     4.0                                                        CLS2    66
      XOR       TEMP2    SREG5  SREG7                             CLS2    67
      IF        TEMP2    EQ     1                                 CLS2    68
                CALL     BRNHTO IMM1    PC.                       CLS2    69
      ENDIF                                                        CLS2    70
      ENDINSTR                                                     CLS2    71
 BMI - BRANCH IF MINUS                                            CLS2    72
 2B     4.0                                                        CLS2    73
      IF        SREG5    EQ     1                                 CLS2    74
                CALL     BRNHTO IMM1    PC.                       CLS2    75
      ENDIF                                                        CLS2    76
      ENDINSTR                                                     CLS2    77
 BNE - BRANCH IF NOT = 0                                          CLS2    78
 26     4.0                                                        CLS2    79
      IF        SREG6    EQ     0                                 CLS2    80
                CALL     BRNHTO IMM1    PC.                       CLS2    81
      ENDIF                                                        CLS2    82
      ENDINSTR                                                     CLS2    83
 BVC - BRANCH IF OVERFLOW CLEAR                                   CLS2    84
 28     4.0                                                        CLS2    85
      IF        SREG7    EQ     0                                 CLS2    86
                CALL     BRNHTO IMM1    PC.                       CLS2    87
      ENDIF                                                        CLS2    88
      ENDINSTR                                                     CLS2    89
 BVS - BRANCH IF OVERFLOW SET                                     CLS2    90
 29     4.0                                                        CLS2    91
      IF        SREG7    EQ     1                                 CLS2    92
                CALL     BRNHTO IMM1    PC.                       CLS2    93
      ENDIF                                                        CLS2    94
      ENDINSTR                                                     CLS2    95
 BPL - BRANCH IF PLUS                                             CLS2    96
 2A     4.0                                                        CLS2    97
      IF        SREG5    EQ     0                                 CLS2    98
                CALL     BRNHTO IMM1    PC.                       CLS2    99
      ENDIF                                                        CLS2   100
      ENDINSTR                                                     CLS2   101
 BSR - BRANCH TO SUBROUTINE                                       CLS2   102
 8D     8.0                                                        CLS2   103
      ADD       TEMP2    PC     2                                 CLS2   104
      AND       TEMP3    TEMP2  FF+16                             CLS2   105
      AND       TEMP4    TEMP2  FF00+16                           CLS2   106
      SHRL      TEMP4    TEMP4  8                                 CLS2   107
      PUSH      TEMP3                                             CLS2   108
      PUSH      TEMP4                                             CLS2   109
      CALL      BRNHTO   IMM1   PC.                               CLS2   110
      ENDINSTR                                                     CLS2   111
ENDINSDEF                                                          CLS2   112
DEFINE          BRNHTO   TEMP1  TEMP2.                           CLS2   113
      MOVE      TEMP3    TEMP1                                    CLS2   114
      AND       TEMP4    TEMP3  80+16                             CLS2   115
      IF        TEMP4    NE     0                                 CLS2   116
```

```
            CONCAT   TEMP3  FF#16  (8)    TEMP3                    CLS2   117
      ENDIF                                                        CLS2   118
      ADD      TEMP2   TEMP2  TEMP3                                CLS2   119
      AND      TEMP2   TEMP2  FFFF#16                              CLS2   120
      RETURN                                                       CLS2   121
      ENDINSTR                                                     CLS2   122
ENDCLASS                                                           CLS2   123
```

address if its testable condition is true, this common task may be factored out using the subroutine definition capability of IDL. Two operands are necessary to this task of calculating the effective address: the program counter (PC), and the first operand field of the instruction (known to SIM/GEN as IMM1). These are passed as actual parameters to the subroutine BRNHTO (CLS2.113-123), since the only operands allowed in the body of a user-defined subroutine in IDL are the temporary IDL operands TEMP1, TEMP2 ... TEMP8 (page 53). The CALL statement in IDL is then used to invoke the subroutine, with the actual parameters replacing the formal parameters in the usual understanding of a FORTRAN CALL (by reference).

The BRNHTO subroutine simulates the calculation of the effective address for the relative mode of addressing of the M6800 microprocessor: the second byte of the instruction treated as a 7-bit signed value (TEMP1) is added to the 16-bit program counter (TEMP2), plus 2. Upon entry, a check is first made to find out if the 7-bit signed offset is negative. If so, the equivalent negative number in 16-bit format is generated using the CONCAT operation (page 52) of IDL. The result is held in TEMP3 and then added to the contents of the program counter (CLS2.119). The AND operation before the RETURN ensures that only the low 16 bits are kept on the host word. If this is not done, an incorrect PC value is very likely to result. Consider the case when PC = 0080 and IMM1 = FB, both expressed in hexadecimal. Since IMM1 is a 7-bit signed value (-5 in 2's complement), it becomes FFFB,

which is the 16-bit signed value equivalent. When the addition is carried out on the host word (of 60-bits in our case), the sum is 1007B and is not correct! Only by masking out the low 16 bits is the correct answer of 7B obtained.

A second point to be made here concerns the question: why is 2 not added to the PC? This has to do with the simulator that SIM/GEN produces. It is natural to assume that during program execution (not to be confused with program simulation) the instruction currently executed has its address in the PC. In the case of this 2-byte relative mode of addressing instruction of the M6800, it is quite correct to make the analogous assumption that the program counter contains the address of the opcode (the first byte of the 2-byte instruction). This is not the case with the simulator produced by SIM/GEN. Upon entry to the code proper to simulate an instruction, the value of the PC is already 1 more than the address of the last memory word comprising the simulated instruction. That is, PC has been advanced to point to the next instruction. Adding 2 to the PC in this instance would result in an incorrect simulation for this class of instructions. To summarize, we note the following:

(1) CALLing a subroutine DEFINEd in IDL assumes that binding with the actual parameters from the calling statement is ordered as in FORTRAN (call by reference). Care must be taken, therefore, not to involve the parameters in destructive-type IDL operations (hence the MOVE statement at CLS2.114).

(2) The bit-width of the host-word is particularly important in arithmetic operations. The simulated operands have bit-widths of their own and the two must not be confused. Throughout, in this tutorial, our host machine has a 60-bit word.

(3) At the instruction emulation level, the program counter already contains the address of the next instruction in sequence. This is characteristic of the generated simulator.

(4) The M6800 microprocessor, our target machine in this tutorial, uses 2's complement arithmetic. But our host machine, the CDC-6400, is a 1's complement machine. The details of this will be taken up in a later section.

3.7 The condition code subroutines

This section will deal with the condition code settings that are affected by certain instructions as noted in Figures 3.1-3.4. The mode of addressing has nothing to do with the setting or clearing of a particular status bit of the condition code register. So these routines will be needed according to instruction, not addressing mode. Our classification scheme is by addressing mode and tends to obscure this commonality somewhat. So this section will present the subroutines that are CALLed from more than 1 execution class. We start with the following table, condensed from Figures 3.1, 3.2, 3.4:

	SREGn	Addition	Subtraction	Shift
Half-Carry	(3)	$P_3Q_3 + P_3\overline{R}_3 + \overline{R}_3Q_3$	unaffected	unaffected
Negative	(5)	R_7	R_7	R_7
Zero	(6)	all bits = 0	all bits = 0	all bits = 0
Overflow	(7)	$P_7Q_7\overline{R}_7 + \overline{P}_7\overline{Q}_7R_7$	$P_7\overline{Q}_7\overline{R}_7 + \overline{P}_7Q_7R_7$	N 0 C
Carry	(8)	$P_7Q_7 + P_7\overline{R}_7 + \overline{R}_7Q_7$	$\overline{P}_7Q_7 + Q_7R_7 + R_7\overline{P}_7$	P_0 / P_7

For the arithmetic-type operations, it is assumed that R = P + Q (addition), or R = P - Q (subtraction). The subscripts denote the bit positions of these 8-bit operands. In the usual understanding, $\overline{P}_7$ implies the complement operation on bit 7 (using a right-to-left numbering with 0 as the rightmost) of the operand P; PQ implies the logical and operation of P and Q, P + Q the logical or operation, and P ⊕ Q the exclusive or operation. The Carry bit setting for the Shift (all Rotate, Shift arithmetic or logical instructions) depends on the initial operand's left-most or right-most bit, the former if a left shift operation is involved, the latter is a right shift.

The following table contains the subroutine names defined to implement each of the necessary status bit settings:

Addition	Subtraction	Shift
CARYHA		
SIGN	SIGN	SIGN
ZERO	ZERO	ZERO
OVERFA	OVERFS	NZVCB
CARYSA	CARYSS	NZVCB

The IDL statements for these routines are contained between CLS1.318-423. In the last column, the SIGN and ZERO routines are

actually embedded in the NZVCB subroutine (CLS1.318-328).

SIGN and ZERO (CLS1.343-356) are both self-explanatory. Each is DEFINEd with 2 parameters, the first parameter being the operand whose sign bit is to be checked for a '1' (for SIGN), or, for which a zero check on all bits is wanted (for ZERO). The second parameter will always be SREG5 for SIGN and SREG6 for ZERO for all CALL's.

The CARYHA, OVERFA, OVERFS, CARYSA and CARYSS subroutines are implementations of the Boolean formulas corresponding to the first table. As is evident from the table, 3 operands are necessary to establish the correct setting for the status bits involved. The first 3 parameters of these 5 subroutines serve this purpose and will always correspond respectively to R, P, and Q in the table. Recall that we are considering R = P + Q (addition) and R = P - Q (subtraction). The fourth parameter will always be SREG3 for CARYHA, SREG7 for OVERFA and OVERFS, and SREG8 for CARYSA and CARYSS. These assumptions are necessary to build up other subroutines that combine the ones already defined. HNZVCA (CLS1.387-394) and NZVCS (CLS1.465-471) are examples of this form of build-up: this way, the condition code setting for a large number of instructions may be invoked by a single CALL statement at the instruction definition level without digressing from the IDL statement or statements for the instruction itself.

A difficulty involved in reading the subroutine definitions in IDL arises from the restriction that the only operands (apart from

constants) appearing in subroutine definitions are the IDL temporary registers TEMP1 ... TEMP8. But even at the instruction definition level, where more helpful mneumonics have been provided for such operands as PC, STACKP, IMMi, there is still a considerable degree of difficulty in remembering, say, what SREG0 stands for. For purposes of the subroutines involved in this section, we urge the substitution of TEMP1, TEMP2 and TEMP3 everywhere for R, P, and Q respectively and referring back to the first table presented. TEMP4 will depend on the particular routine as earlier noted. Thus:

CARYHA	R	P	Q	H
CARYSA	R	P	Q	C
CARYSS	R	P	Q	C
OVERFA	R	P	Q	V
OVERFS	R	P	Q	V

where R = P + Q or R = P - Q (R = Result, P = first operand, Q = second operand). At the next "higher" level, we would have:

HNZVCA	R	P	Q	H	N	Z	V	C
NZVCS	R	P	Q	N	Z	V	C	
NZVCB	R	P	N	Z	V	C	mask.	

In the above, HNZVCA will handle the R = P + Q (addition) instructions. NZVCS will handle the R = P - Q (subtract) instructions. NZVCB involves only R and P since this is the group of "shift" (CLS1.120-187) operations and so has a single operand. The last parameter, denoted by "mask", is either a mask for bit 7 (if shift

direction was left-ward) or for bit 0 (if shift direction was right-ward). The mask is applied on P, the operand.

A last subroutine is the NZR subroutine defined at CLS3.195-200. It is called at a large number of instruction definitions and involves checking the Negative and Zero status bits, and resetting the Overflow bit to 0. It is invoked in some cases where the Negative and Zero status bits are to be checked as usual, but where the Overflow bit is set or cleared in a different manner. In this latter case, the third parameter is a dummy TEMP7. The reader will observe numerous occurrences of

```
CALL   NZR   result   SREG5   SREG6   SREG7,  or
CALL   NZR   result   SREG5   SREG6   TEMP7,
```

usually with load/store accumulators instructions, accumulator and memory word operation instructions, test or transfer accumulator/memory instructions (see Figure 3.1 right-hand column).

This completes the condition code setting subroutines.

3.8 Two's complement arithmetic

In this section, the subroutine to perform 2's complement arithmetic on the 1's complement host machine will be defined. Three parameters, corresponding to the calculated difference, the minuend and the subtrahend are necessary for this purpose. The IDL subroutine will have the form

```
DEFINE   SUBTR   TEMP1   TEMP2   TEMP3. ,
```

where it is assumed that the operation

TEMP1 = TEMP2 - TEMP3

is to be carried out. In the ensuing discussion, the operands are assumed to be 8 bit operands for the purpose of simulating the M6800. But the fact that they will be simulated on the 60-bit host word must not be overlooked.

Consider first the following IDL statements to carry out the desired subtraction:

```
CLEAR   TEMP1                       . zero difference
IF      TEMP2   EQ      TEMP3
        RETURN                      . if minuend = subtrahend
ENDIF
COMTWO  TEMP1   TEMP3               . complement subtrahend
AND     TEMP1   TEMP1   255
ADD     TEMP1   TEMP2   TEMP1       . get difference
AND     TEMP1   TEMP1   255
RETURN                              . done
```

If the minuend is equal to the subtrahend, the answer is obvious. Otherwise, in the usual understanding of machine subtraction, we must add to the minuend the negative representation of the subtrahend. This is achieved by the COMTWO operation in IDL. To insure that we are working only with the low 8 bits of these 60-bit operands, a mask (= FF hexadecimal) is performed on the result to clear any garbage beyond the 8th bit. As earlier noted in section 3.6, incorrect operands may subsequently appear if this "safeguard" is not observed. The ADD operation generates the desired difference and is followed by the same safeguard. The importance of this safeguard cannot be overemphasized: consider the simple case when SUBTR is called with the following actual parameters

```
(TEMP2) = 1111 1111   (-1)
(TEMP3) = 0000 0011   (+3)
```

Upon entry, the higher-ordered bits beyond the 8th bit in these 60-bit host words are clear. The COMTWO operation, which translates into the FORTRAN statement

```
TEMP1 = -TEMP3 + 1
```

will execute as follows

```
(TEMP3) = 0 ... 0   0000 0011        . initially
        = 1 ... 1   1111 1100        . after -TEMP3
(TEMP1) = 1 ... 1   1111 1101        . after -TEMP3 + 1
```

Without the mask of the low 8 bits, the situation just before the ADD looks like

```
(TEMP2) = 0 ... 0   1111 1111
(TEMP1) = 1 ... 1   1111 1101.
```

When the ADD is executed, a carry is propagated into and out of the highest ordered bit (60th bit), causing an "end around" carry that is automatically done by the host machine, which results in:

```
(TEMP1) = 0 ... 0   1111 1101.  (-3)
```

This is an incorrect result, if it is to be interpreted as an 8-bit signed value for the M6800 machine, which in 2's complement representation is -3 (decimal). The point here is clear: at any stage during the simulation of an n-bit operand, the higher ordered (60-n) bits of the host word must always be kept clear (i.e., zeroes).

Two other checks have to be done separately. These stem from the fact that in the 1's complement machine, the number 0 has 2 different representations; all ones or zeroes. An extra effort must be

made to get around this. When, for instance, SUBTR is invoked with (TEMP3) = 0 (all zeroes), the COMTWO operation executed in the context of the host machine generates a positive 1, which then gets added to the minuend. Subtracting zero will increment the minuend. If now (TEMP3) = 1, COMTWO will result in a string of ones which is zero to the machine. In this case, subtracting 1 leaves the minuend unaffected. As it turns out, the subsequent mask should correct the situation, but the compiler involved had "avoid negative zero" code in the first place, so that at the IDL level of simulation, we must treat this as a special case.

The complete text for SUBTR is shown on lines CLS1.431-449. An equivalent for 16 bit operands which will be necessary for some M6800 instructions, is shown on lines CLS3.232-250.

3.9 XECUTE module for the M6800

Referring to Figures 3.1 and 3.2, we observe that many instructions can be emulated by a single IDL operation. For those with more than 1 mode of addressing, only the operand fetch step is different. The rest is completely identical. This property is most evidently brought out through the LDAA instruction, (load accumulator A). The 4 sets of IDL statements for this instruction in the direct, immediate, indexed, and extended mode of addressing are as follows:

```
LDAAD - LOAD ACCA FROM MEMORY, DIRECT
96      3.0
      CALL    GETOPN    IMM1     ABUS.
      MOVE    SREG0     DBUS
      CALL    NZR       SREG0    SREG5    SREG6    SREG7.
ENDINSTR

LDAAI - LOAD ACCA FROM IMMEDIATE
86      2.0
      MOVE    SREG0     IMM1
      CALL    NZR       SREG0    SREG5    SREG6    SREG7.
ENDINSTR

LDAAX - LOAD ACCA FROM MEMORY, INDEXED
A6      5.0
      CALL    GETXOP    IMM1     SREG2    ABUS.
      MOVE    SREG0     DBUS
      CALL    NZR       SREG0    SREG5    SREG6    SREG7.
ENDINSTR

LDAAE - LOAD ACCA FROM MEMORY, EXTENDED
B6      4.0
      CALL    GETEOP    IMM1     ABUS.
      MOVE    SREG0     DBUS
      CALL    NZR       SREG0    SREG5    SREG6    SREG7.
ENDINSTR
```

The immediate mode of addressing does not require a memory reference for the operand so the MOVE statement is not preceded by any CALL. GETOPN, GETXOP and GETEOP correspond to the direct, indexed and extended addressing mode routines to fetch the operand using the bus-organized assumption in SIM/GEN. The body of the subroutines should be self-explanatory: upon return the desired operand is in the data bus.

Every instruction with an immediate mode of addressing will also have the direct, indexed and extended modes of addressing. The exceptions to these are the store operations, STAA, STAB, STX, and STS. The indexed and extended modes (CLS6 and CLS7) are completely

identical following the CALL GETXOP or GETEOP, except for the JMP and JSR instructions (CLS6.305-320, CLS7.301-314). Note, that there are several instructions that deal with 16-bit operands, as in LDX, LDS, STX, STS (load, store/index register or stack pointer). The NZVLO, STOLOP and GETLOP (CLS3.201-231) are concerned with the status bit settings, storing & fetching of 16-bit operands.

The routines earlier defined concerning the status bits may now be CALLed where appropriate. In the special cases, the setting is localized, as for instance, in the CPX instruction (CLS5.6-17). The majority of the instructions CALL the ones already defined (NZR, HNZVCA, NZVCS, NZVCB). The details of the emulation code in IDL for all the instructions should be evident from the statements themselves. The remainder of this section is devoted to presenting the less obvious ones.

JSR - Jump to subroutine admits the indexed (CLS6.310-320) and extended (CLS7.305-314) modes of addressing. The operation is graphically illustrated on Figure 4. Note that in the indexed mode, the return address is 2 bytes away from the present instruction. In the extended, it is 3 bytes. Although this is to be noted, we already have the return address in the PC (program counter) upon entry to the emulation code for these instructions, because as already pointed out in section 3.6, the SIM/GEN simulator does it in the operand extract process. Adding 2 or 3 to PC would result in an incorrectly simulated return address. The return address is 16 bits, and must

be kept on the next 2 bytes of the stack. This is accomplished by the 2 AND operations and the SHRL (to right justify the high order 8 bits) operation, followed by the PUSH. Note that the low byte must be PUSHed first.

The only other instruction that requires a detailed explanation is the DAA instruction (Decimal Adjust Accumulator A). In the 8-4-2-1 BCD (Binary Coded Decimal) representation, each decimal digit is represented by its equivalent 4-bit code in binary. That is:

0 = 0000
1 = 0001
2 = 0010
3 = 0011
4 = 0100
5 = 0101
6 = 0110
7 = 0111
8 = 1000
9 = 1001

The above convention is so widely used, people hardly qualify it with the 8-4-2-1 when they say BCD. There are actually 2 more types of BCD, called the 2-4-2-1 and the excess-3 BCD codes, but we shall not be concerned with these.

In the binary addition of decimally interpreted operands, (that is, operands of the form $0 = 0_1\ 0_2$, where 0 is 8 bits, 0_1 is the high-order 4 bits and 0_2 low order 4 bits, with 0_1 and 0_2 having only values from 0 to 9) the following 3 situations may arise (consider 4 bit-operands for the moment):

(1) The sum S, of the 2 digits is such that S is in the range (0000, 1001) inclusive, so the result is correct;

(2) S is in the range (1010, 1111), in which case the result is a nonvalid BCD combination, so a correction is required;

(3) S is in the range (10000, 11001) in which case a carry is generated and the result - as read in the assumed BCD code - is incorrect.

A correction is required in some cases and none in others. The solution is to add 6 decimal to the result when a correction effort is required. Consider:

(1) 5_{10} = 0101
3_{10} = 0011
sum = 1000 = 8_{10} (correct result)

(2) 6_{10} = 0110
8_{10} = 1000
(sum=) 1110 (nonvalid BCD number)
0110 (add 6_{10})
sum= 1 0100 = 14_{10} (correct BCD result)

(3) 9_{10} = 1001
8_{10} = 1000
(sum=)1 0001 (a carry is generated and the result is incorrect)
0110 (add 6_{10})
sum= 1 0111 = 17_{10} (correct BCD result)

The DECIML (CLS1.450-463) is defined with 3 parameters that have the following meanings:

TEMP1 = a 4 bit operand that is either the high 4 bits or low 4 bits of an 8-bit M6800 byte.

TEMP2 = either SREG3 or SREG8, depending on whether the low (SREG3) or high (SREG8) 4 bits are examined;

TEMP3 = returned as a flag to indicate that a carry was generated

in the adjustment phase (i.e., in the addition of 6_{10});

To decimally adjust ACCA, (CLS1.63-75) we mask out the low and the high 4 bits into TEMP1 and TEMP2 respectively, and invoke DECIML to do the necessary correction. In adjusting the low 4 bits, DECIML is called with a second parameter of SREG3 - the Half-Carry flag of the condition code register. This is indeed important to the DECIML routine as there would be no way of finding out whether the previous add resulting in the current contents of ACCA resulted in a carry out of bit 3 (the 4th bit). The second call uses SREG8 - the Carry flag bit - for the same reason. Upon return from the first call, the result flag TEMP3, must be added to the high order 4 bits (TEMP2) first before further adjustment is to be performed.

Returning to the DECIML subroutine, we begin by clearing the return flag parameter (TEMP3) and check to see if any adjustment is required. That is, we ask if operand is greater than 9 or was a carry generated? If not, simply return. If either condition in question is satisfied, the flag is set and 6_{10} is added to TEMP1.

When both the high and low order 4 bits of ACCA have been adjusted, the CONCAT operation is used to establish the decimally adjusted ACCA. The Carry bit, if previously set, cannot be cleared by the DAA instruction. If previously cleared, it will depend on the adjustment of the high order 4.

In closing, observe the following:

(1) The "safeguard" operation (section 3.6) of masking out the low 8 or 16 bits of any arithmetic operand result is followed

throughout all execution classes.

(2) The DECR operation of IDL is avoided for the same reasons as the extra check needed in SUBTR to treat a subtrahend of 1 separately (see CLS1.76-89). Instead, an add with a negative 1 in 2's complement form is performed.

(3) The status bit setting routines are effectively grouped for sets of instructions that set/clear these bits under identical conditions. Examples are NZR, NZVCS, HNZVCA, NZVCB.

(4) The program counter will have as its contents the location of the next available instruction upon entry to the emulation code for an instruction. Instructions affecting the PC (Class 2, JSR) should take this into account.

(5) The bus organized assumption is observed at all phases of simulation. This includes the instruction fetch and operand fetch routines generated at the DECODE module generator. This also explains the state of the PC as noted in (4).

REFERENCES

(1) SIM/GEN 5.3 User's Reference Manual, University Computer Center, by Mueller & Johnson (1976).

(2) ASM/GEN 5.1 User's Reference Manual, University Computer Center, by Mueller & Johnson (1976).

(3) Designing With Microprocessors, IEEE Compcon Fall 76, Catalog no. 76CH1178-3C.

(4) The M6800 Systems Reference and Data Sheets, Motorola Semiconductor Products INC., 1975.

(5) The M6800 Microprocessor Programming Manual, Motorola Semiconductor Products INC., 1975.

(6) SIGPLAN Notices, Vol. 11, #4 April 1, 1976, Proceedings, Interface Meeting on Programming Systems in the Small Processor Environment.

(7) Introduction to Microcomputers & Microprocessors, by Barna & Porat (John Wiley, 1976).

APPENDIX A: ASM/GEN input for the M6800 microprocessor

```
    7    6
    1    1    1
    0
    8
ABA     1B
CLRA    4F
CLRB    5F
CBA     11
COMA    43
COMB    53
NEGA    40
NEGB    50
DAA     19
DECA    4A
DECB    5A
INCA    4C
INCB    5C
PSHA    36
PSHB    37
PULA    32
PULB    33
ROLA    49
ROLB    59
RORA    46
RORB    56
ASLA    48
ASLB    58
ASRA    47
ASRB    57
LSRA    44
LSRB    54
SBA     10
TAB     16
TBA     17
TSTA    4D
TSTB    5D
DEX     09
DES     34
INX     08
INS     31
TXS     35
TSX     30
NOP     02
RTI     3B
RTS     39
SWI     3F
WAI     3E
CLC     0C
CLI     0E
CLV     0A
SEC     0D
SEI     0F
SEV     0B
TAP     06
TPA     07
END
    2    2    2

    8    0
    8    8
BRA     20
BCC     24
BCS     25
BEQ     27
BGE     2C
BGT     2E
BHI     22
BLE     2F
BLS     23
BLT     2D
BMI     2B
BNE     26
BVC     28
BVS     29
BPL     2A
BSR     8D
END
    3    2    2
    8    0
    8    8
ADDAD   9B
ADDBD   DB
ADCAD   99
ADCBD   D9
ANDAD   94
ANDBD   D4
BITAD   95
BITBD   D5
CMPAD   91
CMPBD   D1
EORAD   98
EORBD   D8
LDAAD   96
LDABD   D6
ORAAD   9A
ORABD   DA
STAAD   97
STABD   D7
SUBAD   90
SUBBD   D0
SBCAD   92
SBCBD   D2
CPXD    9C
LDXD    DE
LDSD    9E
STXD    DF
STSD    9F
END
    4    2    2
    8    0
    8    8
ADDAI   8B
ADDBI   CB
ADCAI   89
ADCBI   C9
ANDAI   84

ANDBI   C4
BITAI   85
BITBI   C5
CMPAI   81
CMPBI   C1
EORAI   88
EORBI   C8
LDAAI   86
LDABI   C6
ORAAI   8A
ORABI   CA
SUBAI   80
SUBBI   C0
SBCAI   82
SBCBI   C2
END
    5    2    3
16       0
    8   16
CPXI    8C
LDXI    CE
LDSI    8E
END
    6    2    2
    8    0
    8    8
ADDAX   AB
ADDBX   EB
ADCAX   A9
ADCBX   E9
ANDAX   A4
ANDBX   E4
BITAX   A5
BITBX   E5
CLRX    6F
CMPAX   A1
CMPBX   E1
COMX    63
NEGX    60
DECX    6A
EORAX   A8
EORBX   E8
INCX    6C
LDAAX   A6
LDABX   E6
ORAAX   AA
ORABX   EA
ROLX    69
RORX    66
ASLX    68
ASRX    67
LSRX    64
STAAX   A7
STABX   E7
SUBAX   A0
SBCAX   A2
SBCBX   E2
TSTX    6D

CPXX    AC
LDXX    EE
LDSX    AE
STXX    EF
STSX    AF
JMPX    6E
JSRX    AD
END
    7    2    3
16       0
    8   16
ADDAE   BB
ADDBE   FB
ADCAE   B9
ADCBE   F9
ANDAE   B4
ANDBE   F4
BITAE   B5
BITBE   F5
CLRE    7F
CMPAE   B1
CMPBE   F1
COME    73
NEGE    70
DECE    7A
EORAE   B8
EORBE   F8
INCE    7C
LDAAE   B6
LDABE   F6
ORAAE   BA
ORABE   FA
ROLE    79
RORE    76
ASLE    78
ASRE    77
LSRE    74
STAAE   B7
STABE   F7
SUBAE   B0
SUBBE   F0
SBCAE   B2
SBCBE   F2
TSTE    7D
CPXE    BC
LDXE    FE
LDSE    BE
STXE    FF
STSE    BF
JMPE    7E
JSRE    BD
END
```

APPENDIX B: DEOCDE input for the M6800 microprocessor

```
 7       8
02       1
06.0F    1
10.11    1
16.17    1
19       1
18       1
30.37    1
39       1
38       1
3E.3F    1
40       1
43.44    1
46.4A    1
4C.40    1
4F       1
50       1
53.54    1
56.5A    1
5C.50    1
5F       1
20       2
22.2F    2
80       2
90.92    3
94.9C    3
9E.9F    3
D0.D2    3
D4.D8    3
DE.DF    3
80.82    4
84.86    4
88.88    4
C0.C2    4
C4.C6    4
C8.CB    4
8C       5
8E       5
CE       5
60       6
63.64    6
66.6A    6
6C.6F    6
A0.A2    6
A4.AF    6
E0.E2    6
E4.E8    6
EE.EF    6
70       7
73.74    7
76.7A    7
7C.7F    7
80.82    7
84.8F    7
F0.F2    7
F4.F8    7
FE.FF    7
```

APPENDIX C: XECUTE input for the M6800 microprocessor (classes 1, 3-7)

```
    1   51    0                                                                  CLS1     2
    0                                                                            CLS1     3
    9                                                                            CLS1     4
ABA - ADD ACCUMULATORS (A := A+B)                                                CLS1     5
1B    2.0                                                                        CLS1     6
      ADD     TEMP1   SREG0  SREG1                                              CLS1     7
      AND     TEMP1   TEMP1  FF+16                                              CLS1     8
      CALL    HNZVCA  TEMP1  SREG0  SREG1  SREG3  SREG5  SREG6  SREG7  SREG8.   CLS1     9
      MOVE    SREG0   TEMP1                                                      CLS1    10
      ENDINSTR                                                                   CLS1    11
CLRA - CLEAR ACCUMULATOR A                                                       CLS1    12
4F    2.0                                                                        CLS1    13
      CLEAR   SREG0                                                              CLS1    14
      CLEAR   SREG5                                                              CLS1    15
      MOVE    SREG6   1                                                          CLS1    16
      CLEAR   SREG7                                                              CLS1    17
      CLEAR   SREG8                                                              CLS1    18
      ENDINSTR                                                                   CLS1    19
CLRB - CLEAR ACCUMULATOR B                                                       CLS1    20
5F    2.0                                                                        CLS1    21
      CLEAR   SREG1                                                              CLS1    22
      CLEAR   SREG5                                                              CLS1    23
      MOVE    SREG6   1                                                          CLS1    24
      CLEAR   SREG7                                                              CLS1    25
      CLEAR   SREG8                                                              CLS1    26
      ENDINSTR                                                                   CLS1    27
CBA - COMPARE ACCUMULATORS A, B.                                                 CLS1    28
11    2.0                                                                        CLS1    29
      CALL    SUBTR   TEMP1  SREG0  SREG1.                                       CLS1    30
      CALL    NZVCS   TEMP1  SREG0  SREG1  SREG5  SREG6  SREG7  SREG8.          CLS1    31
      ENDINSTR                                                                   CLS1    32
COMA - 1*S COMPLEMENT, ACCA                                                      CLS1    33
43    2.0                                                                        CLS1    34
      COMONE  SREG0   SREG0                                                      CLS1    35
      AND     SREG0   SREG0  FF+16                                               CLS1    36
      CALL    NZR     SREG0  SREG5  SREG6  SREG7.                                CLS1    37
      MOVE    SREG8   1                                                          CLS1    38
      ENDINSTR                                                                   CLS1    39
COMB - 1*S COMPLEMENT, ACCB                                                      CLS1    40
53    2.0                                                                        CLS1    41
      COMONE  SREG1   SREG1                                                      CLS1    42
      AND     SREG1   SREG1  FF+16                                               CLS1    43
      CALL    NZR     SREG1  SREG5  SREG6  SREG7.                                CLS1    44
      MOVE    SREG8   1                                                          CLS1    45
      ENDINSTR                                                                   CLS1    46
NEGA - 2*S COMPLEMENT, ACCA                                                      CLS1    47
40    2.0                                                                        CLS1    48
      COMTWO  SREG0   SREG0                                                      CLS1    49
      AND     SREG0   SREG0  FF+16                                               CLS1    50
      CALL    NZR     SREG0  SREG5  SREG6  TEMP7.                                CLS1    51
      CALL    SETV    SREG0  SREG7  80+16.                                       CLS1    52
      CALL    SETV    SREG6  SREG8  0.                                           CLS1    53
      ENDINSTR                                                                   CLS1    54
NEGB - 2*S COMPLEMENT, ACCB                                                      CLS1    55
50    2.0                                                                        CLS1    56
      COMTWO  SREG1   SREG1                                                      CLS1    57
      AND     SREG1   SREG1  FF+16                                               CLS1    58
      CALL    NZR     SREG1  SREG5  SREG6  TEMP7.                                CLS1    59
```

```
     CALL     SETV     SREG1  SREG7  80*16.                          CLS1    60
     CALL     SETV     SREG6  SREG8  0.                              CLS1    61
     ENDINSTR                                                        CLS1    62
DAA - DECIMAL ADJUST ACCA                                            CLS1    63
19     2.0                                                           CLS1    64
     AND      TEMP1    SREG0  F*16                                   CLS1    65
     SHRL     TEMP2    SREG0  4                                      CLS1    66
     CALL     DECIML   TEMP1  SREG3  TEMP3.                          CLS1    67
     ADD      TEMP2    TEMP2  TEMP3                                  CLS1    68
     CALL     DECIML   TEMP2  SREG8  TEMP4.                          CLS1    69
     IF       TEMP4    EQ     1                                      CLS1    70
              MOVE     SREG8  1                                      CLS1    71
     ENDIF                                                           CLS1    72
     CONCAT   SREG0    TEMP2  (4)    TEMP1                          CLS1    73
     CALL     NZR      SREG0  SREG5  SREG6  TEMP7.                   CLS1    74
     ENDINSTR                                                        CLS1    75
DECA - DECREMENT ACCA                                                CLS1    76
4A     2.0                                                           CLS1    77
     CALL     SETV     SREG0  SREG7  80*16.                          CLS1    78
     ADD      SREG0    SREG0  FF*16                                  CLS1    79
     AND      SREG0    SREG0  FF*16                                  CLS1    80
     CALL     NZR      SREG0  SREG5  SREG6  TEMP7.                   CLS1    81
     ENDINSTR                                                        CLS1    82
DECB - DECREMENT ACCB                                                CLS1    83
5A     2.0                                                           CLS1    84
     CALL     SETV     SREG1  SREG7  80*16.                          CLS1    85
     ADD      SREG1    SREG1  FF*16                                  CLS1    86
     AND      SREG1    SREG1  FF*16                                  CLS1    87
     CALL     NZR      SREG1  SREG5  SREG6  TEMP7.                   CLS1    88
     ENDINSTR                                                        CLS1    89
INCA - INCREMENT ACCA                                                CLS1    90
4C     2.0                                                           CLS1    91
     CALL     SETV     SREG0  SREG7  7F*16.                          CLS1    92
     INCR     SREG0    SREG0                                         CLS1    93
     AND      SREG0    SREG0  FF*16                                  CLS1    94
     CALL     NZR      SREG0  SREG5  SREG6  TEMP7.                   CLS1    95
     ENDINSTR                                                        CLS1    96
INCB - INCREMENT ACCB                                                CLS1    97
5C     2.0                                                           CLS1    98
     CALL     SETV     SREG1  SREG7  7F*16.                          CLS1    99
     INCR     SREG1    SREG1                                         CLS1   100
     AND      SREG1    SREG1  FF*16                                  CLS1   101
     CALL     NZR      SREG1  SREG5  SREG6  TEMP7.                   CLS1   102
     ENDINSTR                                                        CLS1   103
PSHA - PUSH ACCA ONTO STACK                                          CLS1   104
36     4.0                                                           CLS1   105
     PUSH     SREG0                                                 CLS1   106
     ENDINSTR                                                        CLS1   107
PSHB - PUSH ACCB ONTO STACK                                          CLS1   108
37     4.0                                                           CLS1   109
     PUSH     SREG1                                                 CLS1   110
     ENDINSTR                                                        CLS1   111
PULA - PULL DATA ONTO ACCA FROM STACK                                CLS1   112
32     4.0                                                           CLS1   113
     PULL     SREG0                                                 CLS1   114
     ENDINSTR                                                        CLS1   115
PULB - PULL DATA ONTO ACCB FROM STACK                                CLS1   116
33     4.0                                                           CLS1   117
```

```
     PULL     SREG1                                                          CLS1   118
     ENDINSTR                                                                CLS1   119
ROLA - ROTATE LEFT, ACCA                                                     CLS1   120
49     2.0                                                                   CLS1   121
     SHLL     TEMP1    SREG0  1                                             CLS1   122
     OR       TEMP1    TEMP1  SREG8                                         CLS1   123
     CALL     NZVC8    TEMP1  SREG0  SREG5  SREG6  SREG7  SREG8  80*16.     CLS1   124
     MOVE     SREG0    TEMP1                                                CLS1   125
     ENDINSTR                                                                CLS1   126
ROLB - ROTATE LEFT, ACCB                                                     CLS1   127
59     2.0                                                                   CLS1   128
     SHLL     TEMP1    SREG1  1                                             CLS1   129
     OR       TEMP1    SREG1  SREG8                                         CLS1   130
     CALL     NZVC8    TEMP1  SREG1  SREG5  SREG6  SREG7  SREG8  80*16.     CLS1   131
     MOVE     SREG1    TEMP1                                                CLS1   132
     ENDINSTR                                                                CLS1   133
RORA - ROTATE RIGHT, ACCA                                                    CLS1   134
46     2.0                                                                   CLS1   135
     SHRL     TEMP1    SREG0  1                                             CLS1   136
     IF       SREG8    NE     0                                             CLS1   137
              OR       TEMP1  TEMP1  80*16                                  CLS1   138
     ENDIF                                                                   CLS1   139
     CALL     NZVC8    TEMP1  SREG0  SREG5  SREG6  SREG7  SREG8  1.         CLS1   140
     MOVE     SREG0    TEMP1                                                CLS1   141
     ENDINSTR                                                                CLS1   142
RORB - ROTATE RIGHT, ACCB                                                    CLS1   143
56     2.0                                                                   CLS1   144
     SHRL     TEMP1    SREG1  1                                             CLS1   145
     IF       SREG8    NE     0                                             CLS1   146
              OR       TEMP1  TEMP1  80*16                                  CLS1   147
     ENDIF                                                                   CLS1   148
     CALL     NZVC8    TEMP1  SREG1  SREG5  SREG6  SREG7  SREG8  1.         CLS1   149
     MOVE     SREG1    TEMP1                                                CLS1   150
     ENDINSTR                                                                CLS1   151
ASLA - SHIFT LEFT, ARITHMETIC, ACCA                                          CLS1   152
48     2.0                                                                   CLS1   153
     SHLL     TEMP1    SREG0  1                                             CLS1   154
     CALL     NZVC8    TEMP1  SREG0  SREG5  SREG6  SREG7  SREG8  80*16.     CLS1   155
     MOVE     SREG0    TEMP1                                                CLS1   156
     ENDINSTR                                                                CLS1   157
ASLB - SHIFT LEFT, ARITHMETIC, ACCB                                          CLS1   158
58     2.0                                                                   CLS1   159
     SHLL     TEMP1    SREG1  1                                             CLS1   160
     CALL     NZVC8    TEMP1  SREG1  SREG5  SREG6  SREG7  SREG8  80*16.     CLS1   161
     MOVE     SREG1    TEMP1                                                CLS1   162
     ENDINSTR                                                                CLS1   163
ASRA - SHIFT RIGHT, ARITHMETIC, ACCA                                         CLS1   164
47     2.0                                                                   CLS1   165
     SHRA     TEMP1    SREG0  1                                             CLS1   166
     CALL     NZVC8    TEMP1  SREG0  SREG5  SREG6  SREG7  SREG8  1.         CLS1   167
     MOVE     SREG0    TEMP1                                                CLS1   168
     ENDINSTR                                                                CLS1   169
ASRB - SHIFT RIGHT, ARITHMETIC, ACCB                                         CLS1   170
57     2.0                                                                   CLS1   171
     SHRA     TEMP1    SREG1  1                                             CLS1   172
     CALL     NZVC8    TEMP1  SREG1  SREG5  SREG6  SREG7  SREG8  1.         CLS1   173
     MOVE     SREG1    TEMP1                                                CLS1   174
     ENDINSTR                                                                CLS1   175
```

```
LSRA - LOGICAL SHIFT RIGHT, ACCA                                                CLS1   176
44     2.0                                                                      CLS1   177
       SHRL     TEMP1   SREG0  1                                               CLS1   178
       CALL     NZVCB   TEMP1  SREG0  SREG5  SREG6  SREG7  SREG8  1.           CLS1   179
       MOVE     SREG0   TEMP1                                                  CLS1   180
       ENDINSTR                                                               CLS1   181
LSRB - LOGICAL SHIFT RIGHT, ACCB                                               CLS1   182
54     2.0                                                                     CLS1   183
       SHRL     TEMP1   SREG1  1                                               CLS1   184
       CALL     NZVCB   TEMP1  SREG1  SREG5  SREG6  SREG7  SREG8  1.           CLS1   185
       MOVE     SREG1   TEMP1                                                  CLS1   186
       ENDINSTR                                                               CLS1   187
SBA - SUBTRACT ACCB FROM ACCA                                                  CLS1   188
10     2.0                                                                     CLS1   189
       CALL     SUBTR   TEMP1  SREG0  SREG1.                                   CLS1   190
       CALL     NZVCS   TEMP1  SREG0  SREG1  SREG5  SREG6  SREG7  SREG8.      CLS1   191
       MOVE     SREG0   TEMP1                                                  CLS1   192
       ENDINSTR                                                               CLS1   193
TAB - TRANSFER ACCA TO ACCB                                                    CLS1   194
16     2.0                                                                     CLS1   195
       MOVE     SREG1   SREG0                                                  CLS1   196
       CALL     NZR     SREG1  SREG5  SREG6  SREG7.                            CLS1   197
       ENDINSTR                                                               CLS1   198
TBA - TRANSFER ACCB TO ACCA                                                    CLS1   199
17     2.0                                                                     CLS1   200
       MOVE     SREG0   SREG1                                                  CLS1   201
       CALL     NZR     SREG0  SREG5  SREG6  SREG7.                            CLS1   202
       ENDINSTR                                                               CLS1   203
TSTA - TEST ACCA                                                               CLS1   204
4D     2.0                                                                     CLS1   205
       CALL     NZR     SREG0  SREG5  SREG6  SREG7.                            CLS1   206
       CLEAR    SREG8                                                         CLS1   207
       ENDINSTR                                                               CLS1   208
TSTB - TEST ACCB                                                               CLS1   209
5D     2.0                                                                     CLS1   210
       CALL     NZR     SREG1  SREG5  SREG6  SREG7.                            CLS1   211
       CLEAR    SREG8                                                         CLS1   212
       ENDINSTR                                                               CLS1   213
DEX - DECREMENT INDEX REGISTER                                                 CLS1   214
09     4.0                                                                     CLS1   215
       ADD      SREG2   SREG2  FFFF+16                                        CLS1   216
       AND      SREG2   SREG2  FFFF+16                                        CLS1   217
       CALL     ZERO    SREG2  SREG6.                                         CLS1   218
       ENDINSTR                                                               CLS1   219
DES - DECREMENT STACK POINTER                                                  CLS1   220
34     4.0                                                                     CLS1   221
       ADD      STACKP  STACKP FFFF+16                                        CLS1   222
       AND      STACKP  STACKP FFFF+16                                        CLS1   223
       ENDINSTR                                                               CLS1   224
INX -  INCREMENT INDEX REGISTER                                                CLS1   225
08     4.0                                                                     CLS1   226
       INCR     SREG2   SREG2                                                 CLS1   227
       AND      SREG2   SREG2  FFFF+16                                        CLS1   228
       CALL     ZERO    SREG2  SREG6.                                         CLS1   229
       ENDINSTR                                                               CLS1   230
INS  -   INCREMENT STACK POINTER                                               CLS1   231
31     4.0                                                                     CLS1   232
       INCR     STACKP  STACKP                                                CLS1   233
```

```
     AND     STACKP  STACKP FFFF#16                          CLS1   234
     ENDINSTR                                                CLS1   235
TXS - TRANSFER (SP := IX-1)                                  CLS1   236
35     4.0                                                   CLS1   237
     ADD     STACKP  SREG2  FFFF#16                          CLS1   238
     AND     STACKP  STACKP FFFF#16                          CLS1   239
     ENDINSTR                                                CLS1   240
TSX - TRANSFER (IX := SP+1)                                  CLS1   241
30     4.0                                                   CLS1   242
     INCR    SREG2   STACKP                                  CLS1   243
     AND     SREG2   SREG2  FFFF#16                          CLS1   244
     ENDINSTR                                                CLS1   245
NOP - NO OPERATION                                           CLS1   246
02     2.0                                                   CLS1   247
     EXIT                                                    CLS1   248
     ENDINSTR                                                CLS1   249
RTI - RETURN FROM INTERRUPT                                  CLS1   250
3B     10.0                                                  CLS1   251
     DISPLAY # RETURN FROM INTERRUPT #                       CLS1   252
     ENDINSTR                                                CLS1   253
RTS - RETURN FROM SUBROUTINE                                 CLS1   254
39     5.0                                                   CLS1   255
     DISPLAY # RETURN FROM SUBROUTINE #                      CLS1   256
     PULL    TEMP1                                           CLS1   257
     PULL    TEMP2                                           CLS1   258
     CONCAT  PC      TEMP1  (8)    TEMP2                     CLS1   259
     ENDINSTR                                                CLS1   260
SWI - SOFTWARE INTERRUPT                                     CLS1   261
3F     12.0                                                  CLS1   262
     DISPLAY # SOFTWARE INTERRUPT #                          CLS1   263
     ENDINSTR                                                CLS1   264
WAI - WAIT FOR INTERRUPT                                     CLS1   265
3E     9.0                                                   CLS1   266
     DISPLAY # WAIT FOR INTERRUPT #                          CLS1   267
     ENDINSTR                                                CLS1   268
CLC - CLEAR CARRY                                            CLS1   269
0C     2.0                                                   CLS1   270
     CLEAR   SREG8                                           CLS1   271
     ENDINSTR                                                CLS1   272
CLI - CLEAR INTERRUPT MASK                                   CLS1   273
0E     2.0                                                   CLS1   274
     CLEAR   SREG4                                           CLS1   275
     ENDINSTR                                                CLS1   276
CLV - CLEAR OVERFLOW                                         CLS1   277
0A     2.0                                                   CLS1   278
     CLEAR   SREG7                                           CLS1   279
     ENDINSTR                                                CLS1   280
SEC - SET CARRY                                              CLS1   281
0D     2.0                                                   CLS1   282
     MOVE    SREG8   1                                       CLS1   283
     ENDINSTR                                                CLS1   284
SEI  -  SET  INTERRUPT                                       CLS1   285
0F     2.0                                                   CLS1   286
     MOVE    SREG4   1                                       CLS1   287
     ENDINSTR                                                CLS1   288
SEV - SET OVERFLOW                                           CLS1   289
0B     2.0                                                   CLS1   290
     MOVE    SREG7   1                                       CLS1   291
```

```
      ENDINSTR                                                                  CLS1   292
TAP - TRANSFER ACCA TO CCR                                                      CLS1   293
06     2.0                                                                      CLS1   294
       AND  SREG8  SREG0  1                                                     CLS1   295
       AND  SREG7  SREG0  2                                                     CLS1   296
       AND  SREG6  SREG0  4                                                     CLS1   297
       AND  SREG5  SREG0  8                                                     CLS1   298
       AND  SREG4  SREG0  10*16                                                 CLS1   299
       AND  SREG3  SREG0  20*16                                                 CLS1   300
       ENDINSTR                                                                 CLS1   301
TPA - TRANSFER CCR*S TO ACCA                                                    CLS1   302
07     2.0                                                                      CLS1   303
       MOVE  SREG0  C0*16                                                       CLS1   304
       SHLC  TEMP1  SREG3  5                                                    CLS1   305
       OR    SREG0  TEMP1  SREG0                                                CLS1   306
       SHLC  TEMP1  SREG4  4                                                    CLS1   307
       OR    SREG0  TEMP1  SREG0                                                CLS1   308
       SHLC  TEMP1  SREG5  3                                                    CLS1   309
       OR    SREG0  TEMP1  SREG0                                                CLS1   310
       SHLC  TEMP1  SREG6  2                                                    CLS1   311
       OR    SREG0  TEMP1  SREG0                                                CLS1   312
       SHLC  TEMP1  SREG7  1                                                    CLS1   313
       OR    SREG0  TEMP1  SREG0                                                CLS1   314
       OR    SREG0  SREG8  SREG0                                                CLS1   315
       ENDINSTR                                                                 CLS1   316
ENDINSDEF                                                                       CLS1   317
DEFINE        NZVCB   TEMP1  TEMP2  TEMP3  TEMP4  TEMP5  TEMP6  TEMP7.         CLS1   318
      CALL    SIGN    TEMP1  TEMP3.                                            CLS1   319
      CALL    ZERO    TEMP1  TEMP4.                                            CLS1   320
      CLEAR   TEMP6                                                           CLS1   321
      AND     TEMP8   TEMP2  TEMP7                                            CLS1   322
      IF      TEMP8   NE     0                                                CLS1   323
              MOVE    TEMP6  1                                                CLS1   324
      ENDIF                                                                   CLS1   325
      XOR     TEMP5   TEMP3  TEMP6                                            CLS1   326
      RETURN                                                                  CLS1   327
      ENDINSTR                                                                CLS1   328
DEFINE        CARYHA  TEMP1  TEMP2  TEMP3  TEMP4.                             CLS1   329
      AND     TEMP5   TEMP2  TEMP3                                            CLS1   330
      COMONE  TEMP6   TEMP1                                                   CLS1   331
      AND     TEMP7   TEMP6  TEMP3                                            CLS1   332
      AND     TEMP8   TEMP6  TEMP2                                            CLS1   333
      OR      TEMP5   TEMP5  TEMP7                                            CLS1   334
      OR      TEMP5   TEMP5  TEMP8                                            CLS1   335
      AND     TEMP5   TEMP5  8*16                                             CLS1   336
      CLEAR   TEMP4                                                           CLS1   337
      IF      TEMP5   NE     0                                                CLS1   338
              MOVE    TEMP4  1                                                CLS1   339
      ENDIF                                                                   CLS1   340
      RETURN                                                                  CLS1   341
      ENDINSTR                                                                CLS1   342
DEFINE        SIGN    TEMP1  TEMP2.                                           CLS1   343
      CLEAR   TEMP2                                                           CLS1   344
      AND     TEMP3   TEMP1  80*16                                            CLS1   345
      IF      TEMP3   NE     0                                                CLS1   346
              MOVE    TEMP2  1                                                CLS1   347
      ENDIF                                                                   CLS1   348
      RETURN                                                                  CLS1   349
```

```
      ENDINSTR                                                                  CLS1   350
DEFINE        ZERO    TEMP1  TEMP2.                                           CLS1   351
      CLEAR   TEMP2                                                           CLS1   352
      IF      TEMP1   EQ     0                                                CLS1   353
              MOVE    TEMP2  1                                                CLS1   354
      ENDIF                                                                   CLS1   355
      ENDINSTR                                                                CLS1   356
DEFINE        OVERFA  TEMP1  TEMP2  TEMP3  TEMP4.                             CLS1   357
      COMONE  TEMP5   TEMP1                                                   CLS1   358
      AND     TEMP6   TEMP2  TEMP3                                            CLS1   359
      AND     TEMP6   TEMP6  TEMP5                                            CLS1   360
      COMONE  TEMP7   TEMP2                                                   CLS1   361
      COMONE  TEMP8   TEMP3                                                   CLS1   362
      AND     TEMP7   TEMP7  TEMP8                                            CLS1   363
      AND     TEMP7   TEMP7  TEMP1                                            CLS1   364
      OR      TEMP6   TEMP6  TEMP7                                            CLS1   365
      AND     TEMP6   TEMP6  80*16                                            CLS1   366
      CLEAR   TEMP4                                                           CLS1   367
      IF      TEMP6   NE     0                                                CLS1   368
              MOVE    TEMP4  1                                                CLS1   369
      ENDIF                                                                   CLS1   370
      RETURN                                                                  CLS1   371
      ENDINSTR                                                                CLS1   372
DEFINE        CARYSA  TEMP1  TEMP2  TEMP3  TEMP4.                             CLS1   373
      AND     TEMP5   TEMP2  TEMP3                                            CLS1   374
      COMONE  TEMP6   TEMP1                                                   CLS1   375
      AND     TEMP7   TEMP6  TEMP3                                            CLS1   376
      AND     TEMP8   TEMP6  TEMP2                                            CLS1   377
      OR      TEMP5   TEMP5  TEMP7                                            CLS1   378
      OR      TEMP5   TEMP5  TEMP8                                            CLS1   379
      AND     TEMP5   TEMP5  80*16                                            CLS1   380
      CLEAR   TEMP4                                                           CLS1   381
      IF      TEMP5   NE     0                                                CLS1   382
              MOVE    TEMP4  1                                                CLS1   383
      ENDIF                                                                   CLS1   384
      RETURN                                                                  CLS1   385
      ENDINSTR                                                                CLS1   386
DEFINE        HNZVCA  TEMP1  TEMP2  TEMP3  TEMP4  TEMP5  TEMP6  TEMP7  TEMP8. CLS1   387
      CALL    CARYHA  TEMP1  TEMP2  TEMP3  TEMP4.                             CLS1   388
      CALL    SIGN    TEMP1  TEMP5.                                           CLS1   389
      CALL    ZERO    TEMP1  TEMP6.                                           CLS1   390
      CALL    OVERFA  TEMP1  TEMP2  TEMP3  TEMP7.                             CLS1   391
      CALL    CARYSA  TEMP1  TEMP2  TEMP3  TEMP8.                             CLS1   392
      RETURN                                                                  CLS1   393
      ENDINSTR                                                                CLS1   394
DEFINE        OVERFS  TEMP1  TEMP2  TEMP3  TEMP4.                             CLS1   395
      COMONE  TEMP5   TEMP3                                                   CLS1   396
      COMONE  TEMP6   TEMP1                                                   CLS1   397
      AND     TEMP7   TEMP5  TEMP6                                            CLS1   398
      AND     TEMP7   TEMP7  TEMP2                                            CLS1   399
      COMONE  TEMP5   TEMP2                                                   CLS1   400
      AND     TEMP6   TEMP1  TEMP3                                            CLS1   401
      AND     TEMP6   TEMP6  TEMP5                                            CLS1   402
      OR      TEMP8   TEMP6  TEMP7                                            CLS1   403
      AND     TEMP8   TEMP8  80*16                                            CLS1   404
      CLEAR   TEMP4                                                           CLS1   405
      IF      TEMP8   NE     0                                                CLS1   406
              MOVE    TEMP4  1                                                CLS1   407
```

```
      ENDIF                                                                  CLS1   408
      RETURN                                                                 CLS1   409
      ENDINSTR                                                               CLS1   410
DEFINE        CARYSS  TEMP1  TEMP2  TEMP3  TEMP4.                            CLS1   411
      COMONE  TEMP5   TEMP2                                                  CLS1   412
      AND     TEMP6   TEMP5  TEMP1                                           CLS1   413
      AND     TEMP7   TEMP5  TEMP3                                           CLS1   414
      AND     TEMP8   TEMP1  TEMP3                                           CLS1   415
      OR      TEMP6   TEMP6  TEMP7                                           CLS1   416
      OR      TEMP6   TEMP6  TEMP8                                           CLS1   417
      AND     TEMP6   TEMP6  80+16                                           CLS1   418
      CLEAR   TEMP4                                                          CLS1   419
      IF      TEMP6   NE     0                                               CLS1   420
              MOVE    TEMP4  1                                               CLS1   421
      ENDIF                                                                  CLS1   422
      ENDINSTR                                                               CLS1   423
DEFINE        SETV    TEMP1  TEMP2  TEMP3.                                   CLS1   424
      CLEAR   TEMP2                                                          CLS1   425
      IF      TEMP1   EQ     TEMP3                                           CLS1   426
              MOVE    TEMP2  1                                               CLS1   427
      ENDIF                                                                  CLS1   428
      RETURN                                                                 CLS1   429
      ENDINSTR                                                               CLS1   430
DEFINE        SUBTR   TEMP1  TEMP2  TEMP3.                                   CLS1   431
      CLEAR   TEMP1                                                          CLS1   432
      IF      TEMP2   EQ     TEMP3                                           CLS1   433
              RETURN                                                         CLS1   434
      ENDIF                                                                  CLS1   435
      IF      TEMP3   EQ     0                                               CLS1   436
              MOVE    TEMP1  TEMP2                                           CLS1   437
              RETURN                                                         CLS1   438
      ENDIF                                                                  CLS1   439
      IF      TEMP3   EQ     1                                               CLS1   440
              ADD     TEMP1  TEMP2  FF+16                                    CLS1   441
              RETURN                                                         CLS1   442
      ENDIF                                                                  CLS1   443
      COMTWO  TEMP1  TEMP3                                                   CLS1   444
      AND     TEMP1  TEMP1  FF+16                                            CLS1   445
      ADD     TEMP1  TEMP2  TEMP1                                            CLS1   446
      AND     TEMP1  TEMP1  FF+16                                            CLS1   447
      RETURN                                                                 CLS1   448
      ENDINSTR                                                               CLS1   449
DEFINE        DECIML  TEMP1  TEMP2  TEMP3.                                   CLS1   450
      CLEAR   TEMP3                                                          CLS1   451
      IF      TEMP1   GT     9                                               CLS1   452
              MOVE    TEMP3  1                                               CLS1   453
      ENDIF                                                                  CLS1   454
      IF      TEMP2   NE     0                                               CLS1   455
              MOVE    TEMP3  1                                               CLS1   456
      ENDIF                                                                  CLS1   457
      IF      TEMP3   EQ     0                                               CLS1   458
              RETURN                                                         CLS1   459
      ENDIF                                                                  CLS1   460
      ADD     TEMP1  TEMP1  6                                                CLS1   461
      AND     TEMP1  TEMP1  F+16                                             CLS1   462
      RETURN                                                                 CLS1   463
      ENDINSTR                                                               CLS1   464
DEFINE        NZVCS   TEMP1  TEMP2  TEMP3  TEMP4  TEMP5  TEMP6  TEMP7.       CLS1   465
```

```
      CALL     SIGN     TEMP1  TEMP4.                                    CLS1   466
      CALL     ZERO     TEMP1  TEMP5.                                    CLS1   467
      CALL     OVERFS   TEMP1  TEMP2  TEMP3  TEMP6.                       CLS1   468
      CALL     CARYSS   TEMP1  TEMP2  TEMP3  TEMP7.                       CLS1   469
      RETURN                                                             CLS1   470
      ENDINSTR                                                           CLS1   471
ENDCLASS                                                                 CLS1   472
```

```
   3   27    1                                                                  CLS3     2
   8                                                                            CLS3     3
   0                                                                            CLS3     4
   9                                                                            CLS3     5
ADDAD - (ACCA := ACCA + M)                                                      CLS3     6
98    3.0                                                                       CLS3     7
      CALL     GETOPN   IMM1   ABUS.                                           CLS3     8
      ADD      TEMP1    SREG0  DBUS                                            CLS3     9
      AND      TEMP1    TEMP1  FF+16                                           CLS3    10
      CALL     HNZVCA   TEMP1  SREG0  DBUS   SREG3  SREG5  SREG6  SREG7  SREG8. CLS3    11
      MOVE     SREG0    TEMP1                                                  CLS3    12
      ENDINSTR                                                                CLS3    13
ADDBD - (ACCB := ACCB + M)                                                      CLS3    14
D8    3.0                                                                       CLS3    15
      CALL     GETOPN   IMM1   ABUS.                                           CLS3    16
      ADD      TEMP1    SREG1  DBUS                                            CLS3    17
      AND      TEMP1    TEMP1  FF+16                                           CLS3    18
      CALL     HNZVCA   TEMP1  SREG1  DBUS   SREG3  SREG5  SREG6  SREG7  SREG8. CLS3    19
      MOVE     SREG1    TEMP1                                                  CLS3    20
      ENDINSTR                                                                CLS3    21
ADCAD - (ACCA := ACCA + M + CARRY)                                              CLS3    22
99    3.0                                                                       CLS3    23
      CALL     GETOPN   IMM1   ABUS.                                           CLS3    24
      ADD      TEMP1    SREG0  DBUS                                            CLS3    25
      ADD      TEMP1    TEMP1  SREG8                                           CLS3    26
      AND      TEMP1    TEMP1  FF+16                                           CLS3    27
      CALL     HNZVCA   TEMP1  SREG0  DBUS   SREG3  SREG5  SREG6  SREG7  SREG8. CLS3    28
      MOVE     SREG0    TEMP1                                                  CLS3    29
      ENDINSTR                                                                CLS3    30
ADCBD - (ACCB := ACCB + M + CARRY)                                              CLS3    31
D9    3.0                                                                       CLS3    32
      CALL     GETOPN   IMM1   ABUS.                                           CLS3    33
      ADD      TEMP1    SREG1  DBUS                                            CLS3    34
      ADD      TEMP1    TEMP1  SREG8                                           CLS3    35
      AND      TEMP1    TEMP1  FF+16                                           CLS3    36
      CALL     HNZVCA   TEMP1  SREG1  DBUS   SREG3  SREG5  SREG6  SREG7  SREG8. CLS3    37
      MOVE     SREG1    TEMP1                                                  CLS3    38
      ENDINSTR                                                                CLS3    39
ANDAD - (ACCA := ACCA .AND. M)                                                  CLS3    40
94    3.0                                                                       CLS3    41
      CALL     GETOPN   IMM1   ABUS.                                           CLS3    42
      AND      SREG0    SREG0  DBUS.                                           CLS3    43
      CALL     NZR      SREG0  SREG5  SREG6  SREG7.                            CLS3    44
      ENDINSTR                                                                CLS3    45
ANDBD - (ACCB := ACCB .AND. M)                                                  CLS3    46
D4    3.0                                                                       CLS3    47
      CALL     GETOPN   IMM1   ABUS.                                           CLS3    48
      AND      SREG1    SREG1  DBUS                                            CLS3    49
      CALL     NZR      SREG1  SREG5  SREG6  SREG7.                            CLS3    50
      ENDINSTR                                                                CLS3    51
BITAD - (ACCA .AND. MEMORY)                                                     CLS3    52
95    3.0                                                                       CLS3    53
      CALL     GETOPN   IMM1   ABUS.                                           CLS3    54
      AND      TEMP1    SREG0  DBUS                                            CLS3    55
      CALL     NZR      TEMP1  SREG5  SREG6  SREG7.                            CLS3    56
      ENDINSTR                                                                CLS3    57
BITBD - (ACC3 .AND. MEMORY)                                                     CLS3    58
```

```
D5     3.0                                                                  CLS3    59
       CALL     GETOPN   IMM1    ABUS.                                      CLS3    60
       AND      TEMP1    SREG1   DBUS                                       CLS3    61
       CALL     NZR      TEMP1   SREG5   SREG6   SREG7.                     CLS3    62
       ENDINSTR                                                            CLS3    63
CMPAD - (COMPARE ACCA WITH MEMORY)                                         CLS3    64
91     3.0                                                                  CLS3    65
       CALL     GETOPN   IMM1    ABUS.                                      CLS3    66
       CALL     SUBTR    TEMP1   SREG0   DBUS.                            CLS3    67
       CALL     NZVCS    TEMP1   SREG0   DBUS    SREG5   SREG6   SREG7   SREG8.  CLS3    68
       ENDINSTR                                                            CLS3    69
CMPBD - (COMPARE ACCB WITH MEMORY)                                         CLS3    70
D1     3.0                                                                  CLS3    71
       CALL     GETOPN   IMM1    ABUS.                                      CLS3    72
       CALL     SUBTR    TEMP1   SREG1   DBUS.                            CLS3    73
       CALL     NZVCS    TEMP1   SREG0   DBUS    SREG5   SREG6   SREG7   SREG8.  CLS3    74
       ENDINSTR                                                            CLS3    75
EORAD - EXCLUSIVE OR ACCA W/ MEMORY                                        CLS3    76
98     3.0                                                                  CLS3    77
       CALL     GETOPN   IMM1    ABUS.                                      CLS3    78
       XOR      SREG0    SREG0   DBUS                                       CLS3    79
       CALL     NZR      SREG0   SREG5   SREG6   SREG7.                     CLS3    80
       ENDINSTR                                                            CLS3    81
EORBD - EXCLUSIVE OR ACCB W/ MEMORY                                        CLS3    82
D8     3.0                                                                  CLS3    83
       CALL     GETOPN   IMM1    ABUS.                                      CLS3    84
       XOR      SREG1    SREG1   DBUS                                       CLS3    85
       CALL     NZR      SREG1   SREG5   SREG6   SREG7.                     CLS3    86
       ENDINSTR                                                            CLS3    87
LDAAD - LOAD ACCA FROM MEMORY                                              CLS3    88
96     3.0                                                                  CLS3    89
       CALL     GETOPN   IMM1    ABUS.                                      CLS3    90
       MOVE     SREG0    DBUS                                              CLS3    91
       CALL     NZR      SREG0   SREG5   SREG6   SREG7.                     CLS3    92
       ENDINSTR                                                            CLS3    93
LDABD - LOAD ACCB FROM MEMORY                                              CLS3    94
D6     3.0                                                                  CLS3    95
       CALL     GETOPN   IMM1    ABUS.                                      CLS3    96
       MOVE     SREG1    DBUS                                              CLS3    97
       CALL     NZR      SREG1   SREG5   SREG6   SREG7.                     CLS3    98
       ENDINSTR                                                            CLS3    99
ORAAD - (ACCA := ACCA .OR. M)                                              CLS3   100
9A     3.0                                                                  CLS3   101
       CALL     GETOPN   IMM1    ABUS.                                      CLS3   102
       OR       SREG0    SREG0   DBUS                                       CLS3   103
       CALL     NZR      SREG0   SREG5   SREG6   SREG7.                     CLS3   104
       ENDINSTR                                                            CLS3   105
ORABD - (ACCB := ACCB .OR. M)                                              CLS3   106
DA     3.0                                                                  CLS3   107
       CALL     GETOPN   IMM1    ABUS.                                      CLS3   108
       OR       SREG1    SREG1   DBUS                                       CLS3   109
       CALL     NZR      SREG1   SREG5   SREG6   SREG7.                     CLS3   110
       ENDINSTR                                                            CLS3   111
STAAD - STORE ACCA INTO MEMORY                                             CLS3   112
97     4.0                                                                  CLS3   113
       MOVE     ABUS     IMM1                                              CLS3   114
       MOVE     DBUS     SREG0                                             CLS3   115
       WRITED                                                              CLS3   116
```

```
     CALL     NZR      DBUS   SREG5  SREG6  SREG7.                          CLS3   117
     ENDINSTR                                                               CLS3   118
STABD - STORE ACCB INTO MEMORY                                              CLS3   119
D7    4.0                                                                   CLS3   120
     MOVE     ABUS     IMM1                                                 CLS3   121
     MOVE     DBUS     SREG1                                                CLS3   122
     WRITED                                                                 CLS3   123
     CALL     NZR      DBUS   SREG5  SREG6  SREG7.                          CLS3   124
     ENDINSTR                                                               CLS3   125
SUBAD - (ACCA := ACCA - M)                                                  CLS3   126
90    3.0                                                                   CLS3   127
     CALL     GETOPN   IMM1   ABUS.                                         CLS3   128
     CALL     SUBTR    TEMP1  SREG0  DBUS.                                  CLS3   129
     CALL     NZVCS    TEMP1  SREG0  DBUS   SREG5  SREG6  SREG7  SREG8.    CLS3   130
     MOVE     SREG0    TEMP1                                                CLS3   131
     ENDINSTR                                                               CLS3   132
SUBBD - (ACCB := ACCB - M)                                                  CLS3   133
D0    3.0                                                                   CLS3   134
     CALL     GETOPN   IMM1   ABUS.                                         CLS3   135
     CALL     SUBTR    TEMP1  SREG1  DBUS.                                  CLS3   136
     CALL     NZVCS    TEMP1  SREG1  DBUS   SREG5  SREG6  SREG7  SREG8.    CLS3   137
     MOVE     SREG1    TEMP1                                                CLS3   138
     ENDINSTR                                                               CLS3   139
SBCAD - (ACCA:=ACCA - M - CARRY)                                            CLS3   140
92    3.0                                                                   CLS3   141
     CALL     GETOPN   IMM1   ABUS.                                         CLS3   142
     CALL     SUBTR    TEMP1  SREG0  DBUS.                                  CLS3   143
     CALL     SUBTR    TEMP2  TEMP1  SREG8.                                 CLS3   144
     CALL     NZVCS    TEMP2  SREG0  DBUS   SREG5  SREG6  SREG7  SREG8.    CLS3   145
     MOVE     SREG0    TEMP2                                                CLS3   146
     ENDINSTR                                                               CLS3   147
SBCBD - (ACCB :=ACCB - M - CARRY)                                           CLS3   148
D2    3.0                                                                   CLS3   149
     CALL     GETOPN   IMM1   ABUS.                                         CLS3   150
     CALL     SUBTR    TEMP1  SREG1  DBUS.                                  CLS3   151
     CALL     SUBTR    TEMP2  TEMP1  SREG8.                                 CLS3   152
     CALL     NZVCS    TEMP2  SREG1  DBUS   SREG5  SREG6  SREG7  SREG8.    CLS3   153
     MOVE     SREG1    TEMP2                                                CLS3   154
     ENDINSTR                                                               CLS3   155
CPXD - COMPARE IX W/ MEMORY                                                 CLS3   156
9C    4.0                                                                   CLS3   157
     CALL     GETLOP   TEMP3  IMM1   ABUS   DBUS.                          CLS3   158
     CALL     SUBTLO   TEMP4  SREG2  TEMP3.                                CLS3   159
     AND      TEMP1    TEMP4  $000+16                                       CLS3   160
     CLEAR    SREG5                                                        CLS3   161
     CLEAR    SREG7                                                        CLS3   162
     IF       TEMP1    NE     0                                            CLS3   163
              MOVE     SREG5  1                                            CLS3   164
              MOVE     SREG7  1                                            CLS3   165
     ENDIF                                                                  CLS3   166
     CALL     ZERO     TEMP4  SREG6.                                        CLS3   167
     ENDINSTR                                                               CLS3   168
LDXD - LOAD IX FROM MEMORY                                                  CLS3   169
DE    4.0                                                                   CLS3   170
     CALL     GETLOP   SREG2  IMM1   ABUS   DBUS.                          CLS3   171
     CALL     NZVLO    SREG2  SREG5  SREG6  SREG7.                          CLS3   172
     ENDINSTR                                                               CLS3   173
LDSD - LOAD STACK POINTER FROM MEMORY                                       CLS3   174
```

```
 9E      4.0                                                          CLS3   175
       CALL     GETLOP   STACKP IMM1    ABUS    DBUS.                CLS3   176
       CALL     NZVLO    STACKP SREG5   SREG6   SREG7.               CLS3   177
       ENDINSTR                                                      CLS3   178
 STXD - STORE IX INTO MEMORY                                         CLS3   179
 DF      5.0                                                         CLS3   180
       CALL     STOLOP   SREG2  IMM1    ABUS    DBUS.                CLS3   181
       CALL     NZVLO    SREG2  SREG5   SREG6   SREG7.               CLS3   182
       ENDINSTR                                                      CLS3   183
 STSD - STORE STACK POINTER INTO MEMORY                              CLS3   184
 9F      5.0                                                         CLS3   185
       CALL     STOLOP   STACKP IMM1    ABUS    DBUS.                CLS3   186
       CALL     NZVLO    STACKP SREG5   SREG6   SREG7.               CLS3   187
       ENDINSTR                                                      CLS3   188
ENDINSDEF                                                          CLS3   189
DEFINE          GETOPN   TEMP1  TEMP2.                             CLS3   190
       MOVE     TEMP2    TEMP1                                      CLS3   191
       READO                                                        CLS3   192
       RETURN                                                       CLS3   193
       ENDINSTR                                                     CLS3   194
DEFINE          NZR      TEMP1  TEMP2   TEMP3   TEMP4.              CLS3   195
       CALL     SIGN     TEMP1  TEMP2.                             CLS3   196
       CALL     ZERO     TEMP1  TEMP3.                             CLS3   197
       MOVE     TEMP4    0                                         CLS3   198
       RETURN                                                       CLS3   199
       ENDINSTR                                                     CLS3   200
DEFINE          GETLOP   TEMP1  TEMP2   TEMP3   TEMP4.              CLS3   201
       CALL     GETOPN   TEMP2  TEMP3.                             CLS3   202
       MOVE     TEMP7    TEMP4                                     CLS3   203
       INCR     TEMP5    TEMP3                                     CLS3   204
       CALL     GETOPN   TEMP5  TEMP3.                             CLS3   205
       MOVE     TEMP8    TEMP4                                     CLS3   206
       CONCAT   TEMP1    TEMP7   (8)     TEMP8                     CLS3   207
       RETURN                                                       CLS3   208
       ENDINSTR                                                     CLS3   209
DEFINE          STOLOP   TEMP1  TEMP2   TEMP3   TEMP4.              CLS3   210
       AND      TEMP6    TEMP1  FF+16                              CLS3   211
       AND      TEMP5    TEMP1  FF00+16                            CLS3   212
       SHRL     TEMP5    TEMP5  8                                  CLS3   213
       MOVE     TEMP4    TEMP5                                     CLS3   214
       MOVE     TEMP3    TEMP2                                     CLS3   215
       WRITED                                                       CLS3   216
       MOVE     TEMP4    TEMP6                                     CLS3   217
       INCR     TEMP3    TEMP3                                     CLS3   218
       WRITED                                                       CLS3   219
       RETURN                                                       CLS3   220
       ENDINSTR                                                     CLS3   221
DEFINE          NZVLO    TEMP1  TEMP2   TEMP3   TEMP4.              CLS3   222
       AND      TEMP5    TEMP1  8000+16                           CLS3   223
       CLEAR    TEMP2                                             CLS3   224
       IF       TEMP5    NE     0                                  CLS3   225
                MOVE     TEMP2  1                                  CLS3   226
       ENDIF                                                        CLS3   227
       CALL     ZERO     TEMP1  TEMP3.                             CLS3   228
       MOVE     TEMP4    0                                         CLS3   229
       RETURN                                                       CLS3   230
       ENDINSTR                                                     CLS3   231
DEFINE          SUBILO   TEMP1  TEMP2   TEMP3.                     CLS3   232
```

```
        CLEAR   TEMP1                                                   CLS3    233
        IF      TEMP2   EQ      TEMP3                                   CLS3    234
                RETURN                                                  CLS3    235
        ENDIF                                                           CLS3    236
        IF      TEMP3   EQ      0                                       CLS3    237
                MOVE    TEMP1   TEMP2                                   CLS3    238
                RETURN                                                  CLS3    239
        ENDIF                                                           CLS3    240
        IF      TEMP3   EQ      1                                       CLS3    241
                ADD     TEMP1   TEMP2   FFFF#16                         CLS3    242
                RETURN                                                  CLS3    243
        ENDIF                                                           CLS3    244
        COMTWO  TEMP1   TEMP3                                           CLS3    245
        AND     TEMP1   TEMP1   FFFF#16                                 CLS3    246
        ADD     TEMP1   TEMP2   TEMP1                                   CLS3    247
        AND     TEMP1   TEMP1   FFFF#16                                 CLS3    248
        RETURN                                                          CLS3    249
        ENDINSTR                                                        CLS3    250
ENDCLASS                                                                CLS3    251
```

```
  4    20     1                                                                    CLS4
  8                                                                                CLS4
  0                                                                                CLS4
  9                                                                                CLS4
ADDAI - (ACCA := ACCA + IMMED.)                                                    CLS4
8B     2.0                                                                         CLS4
     ADD      TEMP1    SREG0   IMM1                                                CLS4
     AND      TEMP1    TEMP1   FF+16                                               CLS4
     CALL     HNZVCA   TEMP1   SREG0   IMM1    SREG3   SREG5   SREG6   SREG7   SREG8.   CLS4
     MOVE     SREG0    TEMP1                                                       CLS4
     ENDINSTR                                                                      CLS4
ADDBI - (ACCB := ACCB + IMMED.)                                                    CLS4
CB     2.0                                                                         CLS4
     ADD      TEMP1    SREG1   IMM1                                                CLS4
     AND      TEMP1    TEMP1   FF+16                                               CLS4
     CALL     HNZVCA   TEMP1   SREG1   IMM1    SREG3   SREG5   SREG6   SREG7   SREG8.   CLS4
     MOVE     SREG1    TEMP1                                                       CLS4
     ENDINSTR                                                                      CLS4
ADCAI - (ACCA := ACCA + IMMED. + CARRY)                                            CLS4
89     2.0                                                                         CLS4
     ADD      TEMP1    SREG0   IMM1                                                CLS4
     ADD      TEMP1    TEMP1   SREG8                                               CLS4
     AND      TEMP1    TEMP1   FF+16                                               CLS4
     CALL     HNZVCA   TEMP1   SREG0   IMM1    SREG3   SREG5   SREG6   SREG7   SREG8.   CLS4
     MOVE     SREG0    TEMP1                                                       CLS4
     ENDINSTR                                                                      CLS4
ADCBI - (ACCB := ACCB + IMMED. + CARRY)                                            CLS4
C9     2.0                                                                         CLS4
     ADD      TEMP1    SREG1   IMM1                                                CLS4
     ADD      TEMP1    TEMP1   SREG8                                               CLS4
     AND      TEMP1    TEMP1   FF+16                                               CLS4
     CALL     HNZVCA   TEMP1   SREG1   IMM1    SREG3   SREG5   SREG6   SREG7   SREG8.   CLS4
     MOVE     SREG1    TEMP1                                                       CLS4
     ENDINSTR                                                                      CLS4
ANDAI - (ACCA := ACCA .AND. IMMED.)                                                CLS4
84     2.0                                                                         CLS4
     AND      SREG0    SREG0   IMM1.                                               CLS4
     CALL     NZR      SREG0   SREG5   SREG6   SREG7.                              CLS4
     ENDINSTR                                                                      CLS4
ANDBI - (ACCB := ACCB .AND. IMMED.)                                                CLS4
C4     2.0                                                                         CLS4
     AND      SREG1    SREG1   IMM1                                                CLS4
     CALL     NZR      SREG1   SREG5   SREG6   SREG7.                              CLS4
     ENDINSTR                                                                      CLS4
BITAI - (ACCA .AND. IMMED.)                                                        CLS4
85     2.0                                                                         CLS4
     AND      TEMP1    SREG0   IMM1                                                CLS4
     CALL     NZR     TEMP1   SREG5   SREG6   SREG7.                               CLS4
     ENDINSTR                                                                      CLS4
BITBI - (ACCB .AND. IMMED.)                                                        CLS4
C5     2.0                                                                         CLS4
     AND      TEMP1    SREG1   IMM1                                                CLS4
     CALL     NZR     TEMP1   SREG5   SREG6   SREG7.                               CLS4
     ENDINSTR                                                                      CLS4
CMPAI - COMPARE ACCA W/ IMMED.                                                     CLS4
81     2.0                                                                         CLS4
     CALL     SUBTR    TEMP1   SREG0   IMM1.                                       CLS4
```

```
     CALL     NZVCS   TEMP1  SREG0  IMM1   SREG5  SREG6  SREG7  SREG8.          CLS4    59
     ENDINSTR                                                                    CLS4    60
CMPBI - COMPARE ACCB W/ IMMED.                                                   CLS4    61
C1    2.0                                                                        CLS4    62
     CALL     SUBTR   TEMP1  SREG1  IMM1.                                        CLS4    63
     CALL     NZVCS   TEMP1  SREG0  IMM1   SREG5  SREG6  SREG7  SREG8.          CLS4    64
     ENDINSTR                                                                    CLS4    65
EORAI - EXCLUSIVE OR ACCA W/ IMMED.                                              CLS4    66
88    2.0                                                                        CLS4    67
     XOR      SREG0   SREG0  IMM1                                               CLS4    68
     CALL     NZR     SREG0  SREG5  SREG6  SREG7.                               CLS4    69
     ENDINSTR                                                                    CLS4    70
EORBI - EXCLUSIVE OR ACCB W/ IMMED.                                              CLS4    71
C8    2.0                                                                        CLS4    72
     XOR      SREG1   SREG1  IMM1                                               CLS4    73
     CALL     NZR     SREG1  SREG5  SREG6  SREG7.                               CLS4    74
     ENDINSTR                                                                    CLS4    75
LDAAI - LOAD ACCA FROM IMMED.                                                    CLS4    76
86    2.0                                                                        CLS4    77
     MOVE     SREG0   IMM1                                                       CLS4    78
     CALL     NZR     SREG0  SREG5  SREG6  SREG7.                               CLS4    79
     ENDINSTR                                                                    CLS4    80
LDABI - LOAD ACCB FROM IMMED.                                                    CLS4    81
C6    2.0                                                                        CLS4    82
     MOVE     SREG1   IMM1                                                       CLS4    83
     CALL     NZR     SREG1  SREG5  SREG6  SREG7.                               CLS4    84
     ENDINSTR                                                                    CLS4    85
ORAAI - (ACCA := ACCA .OR. IMMED.)                                               CLS4    86
8A    2.0                                                                        CLS4    87
     OR       SREG0   SREG0  IMM1                                               CLS4    88
     CALL     NZR     SREG0  SREG5  SREG6  SREG7.                               CLS4    89
     ENDINSTR                                                                    CLS4    90
ORABI - (ACCB := ACCB .OR. IMMED.)                                               CLS4    91
CA    2.0                                                                        CLS4    92
     OR       SREG1   SREG1  IMM1                                               CLS4    93
     CALL     NZR     SREG1  SREG5  SREG6  SREG7.                               CLS4    94
     ENDINSTR                                                                    CLS4    95
SUBAI - (ACCA := ACCA - IMMED.)                                                  CLS4    96
80    2.0                                                                        CLS4    97
     CALL     SUBTR   TEMP1  SREG0  IMM1.                                        CLS4    98
     CALL     NZVCS   TEMP1  SREG0  IMM1   SREG5  SREG6  SREG7  SREG8.          CLS4    99
     MOVE     SREG0   TEMP1                                                      CLS4   100
     ENDINSTR                                                                    CLS4   101
SUBBI - (ACCB := ACCB - IMMED.)                                                  CLS4   102
C0    2.0                                                                        CLS4   103
     CALL     SUBTR   TEMP1  SREG1  IMM1.                                        CLS4   104
     CALL     NZVCS   TEMP1  SREG1  IMM1   SREG5  SREG6  SREG7  SREG8.          CLS4   105
     MOVE     SREG1   TEMP1                                                      CLS4   106
     ENDINSTR                                                                    CLS4   107
SBCAI - (ACCA:=ACCA - IMMED. - CARRY)                                            CLS4   108
82    2.0                                                                        CLS4   109
     CALL     SUBTR   TEMP1  SREG0  IMM1.                                        CLS4   110
     CALL     SUBTR   TEMP2  TEMP1  SREG8.                                       CLS4   111
     CALL     NZVCS   TEMP2  SREG0  IMM1   SREG5  SREG6  SREG7  SREG8.          CLS4   112
     MOVE     SREG0   TEMP2                                                      CLS4   113
     ENDINSTR                                                                    CLS4   114
SBCBI - (ACCB :=ACCB - IMMED. - CARRY)                                           CLS4   115
C2    2.0                                                                        CLS4   116
```

```
      CALL     SUBTR   TEMP1  SREG1  IMM1.                                          CLS4   117
      CALL     SUBTR   TEMP2  TEMP1  SREG8.                                         CLS4   118
      CALL     NZVCS   TEMP2  SREG1  IMM1   SREG5  SREG6  SREG7  SREG8.           CLS4   119
      MOVE     SREG1   TEMP2                                                       CLS4   120
      ENDINSTR                                                                     CLS4   121
ENDINSDEF                                                                          CLS4   122
ENDCLASS                                                                           CLS4   123
```

```
   5     1     1                                                        CLS5      2
  16                                                                    CLS5      3
   0                                                                    CLS5      4
   9                                                                    CLS5      5
CPXI - COMPARE IX W/ IMMED.                                             CLS5      6
8C    3.0                                                               CLS5      7
      CALL     SUBTLO   TEMP4   SREG2   IMM1.                          CLS5      8
      AND      TEMP1    TEMP4   8000+16                                CLS5      9
      CLEAR    SREG5                                                   CLS5     10
      CLEAR    SREG7                                                   CLS5     11
      IF       TEMP1    NE      0                                      CLS5     12
               MOVE     SREG5   1                                      CLS5     13
               MOVE     SREG7   1                                      CLS5     14
      ENDIF                                                           CLS5     15
      CALL     ZERO     TEMP4   SREG6.                                 CLS5     16
      ENDINSTR                                                        CLS5     17
LDXI - LOAD IX FROM IMMED.                                              CLS5     18
CE    3.0                                                               CLS5     19
      MOVE     SREG2    IMM1.                                          CLS5     20
      CALL     NZVLO    SREG2   SREG5   SREG6   SREG7.                 CLS5     21
      ENDINSTR                                                        CLS5     22
LDSI - LOAD STACK POINTER FROM IMMED.                                   CLS5     23
8E    3.0                                                               CLS5     24
      MOVE     STACKP   IMM1                                          CLS5     25
      CALL     NZVLO    STACKP SREG5   SREG6   SREG7.                 CLS5     26
      ENDINSTR                                                        CLS5     27
ENDINSDEF                                                               CLS5     28
ENDCLASS                                                                CLS5     29
```

```
   6   40    1                                                                  CLS6     2
   8                                                                            CLS6     3
   0                                                                            CLS6     4
   9                                                                            CLS6     5
ADDAX - (ACCA := ACCA + M)                                                      CLS6     6
A9    5.0                                                                       CLS6     7
      CALL    GETXOP  IMM1   SREG2  ABUS.                                       CLS6     8
      ADD     TEMP1   SREG0  DBUS                                               CLS6     9
      AND     TEMP1   TEMP1  FF+16                                              CLS6    10
      CALL    HNZVCA  TEMP1  SREG0  DBUS   SREG3  SREG5  SREG6  SREG7  SREG8.  CLS6    11
      MOVE    SREG0   TEMP1                                                     CLS6    12
      ENDINSTR                                                                  CLS6    13
ADDBX - (ACCB := ACCB + M)                                                      CLS6    14
E8    5.0                                                                       CLS6    15
      CALL    GETXOP  IMM1   SREG2  ABUS.                                       CLS6    16
      ADD     TEMP1   SREG1  DBUS                                               CLS6    17
      AND     TEMP1   TEMP1  FF+16                                              CLS6    18
      CALL    HNZVCA  TEMP1  SREG1  DBUS   SREG3  SREG5  SREG6  SREG7  SREG8.  CLS6    19
      MOVE    SREG1   TEMP1                                                     CLS6    20
      ENDINSTR                                                                  CLS6    21
ADCAX - (ACCA := ACCA + M + CARRY)                                              CLS6    22
A9    5.0                                                                       CLS6    23
      CALL    GETXOP  IMM1   SREG2  ABUS.                                       CLS6    24
      ADD     TEMP1   SREG0  DBUS                                               CLS6    25
      ADD     TEMP1   TEMP1  SREG8                                              CLS6    26
      AND     TEMP1   TEMP1  FF+16                                              CLS6    27
      CALL    HNZVCA  TEMP1  SREG0  DBUS   SREG3  SREG5  SREG6  SREG7  SREG8.  CLS6    28
      MOVE    SREG0   TEMP1                                                     CLS6    29
      ENDINSTR                                                                  CLS6    30
ADCBX - (ACCB := ACCB + M + CARRY)                                              CLS6    31
E9    5.0                                                                       CLS6    32
      CALL    GETXOP  IMM1   SREG2  ABUS.                                       CLS6    33
      ADD     TEMP1   SREG1  DBUS                                               CLS6    34
      ADD     TEMP1   TEMP1  SREG8                                              CLS6    35
      AND     TEMP1   TEMP1  FF+16                                              CLS6    36
      CALL    HNZVCA  TEMP1  SREG1  DBUS   SREG3  SREG5  SREG6  SREG7  SREG8.  CLS6    37
      MOVE    SREG1   TEMP1                                                     CLS6    38
      ENDINSTR                                                                  CLS6    39
ANDAX - (ACCA := ACCA .AND. M)                                                  CLS6    40
A4    5.0                                                                       CLS6    41
      CALL    GETXOP  IMM1   SREG2  ABUS.                                       CLS6    42
      AND     SREG0   SREG0  DBUS.                                              CLS6    43
      CALL    NZR     SREG0  SREG5  SREG6  SREG7.                               CLS6    44
      ENDINSTR                                                                  CLS6    45
ANDBX - (ACCB := ACCB .AND. M)                                                  CLS6    46
E4    5.0                                                                       CLS6    47
      CALL    GETXOP  IMM1   SREG2  ABUS.                                       CLS6    48
      AND     SREG1   SREG1  DBUS                                               CLS6    49
      CALL    NZR     SREG1  SREG5  SREG6  SREG7.                               CLS6    50
      ENDINSTR                                                                  CLS6    51
BITAX - (ACCA .AND. MEMORY)                                                     CLS6    52
A5    5.0                                                                       CLS6    53
      CALL    GETXOP  IMM1   SREG2  ABUS.                                       CLS6    54
      AND     TEMP1   SREG0  DBUS                                               CLS6    55
      CALL    NZR     TEMP1  SREG5  SREG6  SREG7.                               CLS6    56
      ENDINSTR                                                                  CLS6    57
BITBX - (ACCB .AND. MEMORY)                                                     CLS6    58
```

```
E5    5.3                                                              CLS6    59
      CALL    GETXOP  IMM1    SREG2   ABUS.                            CLS6    60
      AND     TEMP1   SREG1   DBUS                                     CLS6    61
      CALL    NZR     TEMP1   SREG5   SREG6   SREG7.                   CLS6    62
      ENDINSTR                                                         CLS6    63
CLRX - CLEAR MEMORY                                                    CLS6    64
6F    7.0                                                              CLS6    65
      CALL    GETXOP  IMM1    SREG2   ABUS.                            CLS6    66
      CLEAR   DBUS                                                     CLS6    67
      WRITED                                                           CLS6    68
      CLEAR   SREG5                                                    CLS6    69
      MOVE    SREG6   1                                                CLS6    70
      CLEAR   SREG7                                                    CLS6    71
      CLEAR   SREG8                                                    CLS6    72
      ENDINSTR                                                         CLS6    73
CMPAX - (COMPARE ACCA WITH MEMORY)                                     CLS6    74
A1    5.0                                                              CLS6    75
      CALL    GETXOP  IMM1    SREG2   ABUS.                            CLS6    76
      CALL    SUBTR   TEMP1   SREG0   DBUS.                            CLS6    77
      CALL    NZVCS   TEMP1   SREG0   DBUS   SREG5  SREG6  SREG7  SREG8.  CLS6    78
      ENDINSTR                                                         CLS6    79
CMPBX - (COMPARE ACCB WITH MEMORY)                                     CLS6    80
E1    5.0                                                              CLS6    81
      CALL    GETXOP  IMM1    SREG2   ABUS.                            CLS6    82
      CALL    SUBTR   TEMP1   SREG1   DBUS.                            CLS6    83
      CALL    NZVCS   TEMP1   SREG0   DBUS   SREG5  SREG6  SREG7  SREG8.  CLS6    84
      ENDINSTR                                                         CLS6    85
COMX - 1'S COMPLEMENT MEMORY                                           CLS6    86
63    7.3                                                              CLS6    87
      CALL    GETXOP  IMM1    SREG2   ABUS.                            CLS6    88
      COMONE  DBUS    DBUS                                             CLS6    89
      AND     DBUS    DBUS    FF*16                                    CLS6    90
      WRITED                                                           CLS6    91
      CALL    NZR     DBUS    SREG5   SREG6  SREG7.                    CLS6    92
      MOVE    SREG8   1                                                CLS6    93
      ENDINSTR                                                         CLS6    94
NEGX - 2'S COMPLEMENT MEMORY                                           CLS6    95
60    7.0                                                              CLS6    96
      CALL    GETXOP  IMM1    SREG2   ABUS.                            CLS6    97
      COMTWO  DBUS    DBUS                                             CLS6    98
      AND     DBUS    DBUS    FF*16                                    CLS6    99
      WRITED                                                           CLS6   100
      CALL    NZR     DBUS    SREG5   SREG6  TEMP7.                    CLS6   101
      CALL    SETV    DBUS    SREG7   80*16.                           CLS6   102
      CALL    SETV    SREG6   SREG8   0.                               CLS6   103
      ENDINSTR                                                         CLS6   104
DECX - DECREMENT MEMORY                                                CLS6   105
6A    7.0                                                              CLS6   106
      CALL    GETXOP  IMM1    SREG2   ABUS.                            CLS6   107
      CALL    SETV    DBUS    SREG7   80*16.                           CLS6   108
      ADD     DBUS    DBUS    FF*16                                    CLS6   109
      AND     DBUS    DBUS    FF*16                                    CLS6   110
      WRITED                                                           CLS6   111
      CALL    NZR     DBUS    SREG5   SREG6  TEMP7.                    CLS6   112
      ENDINSTR                                                         CLS6   113
EORAX - EXCLUSIVE OR ACCA W/ MEMORY                                    CLS6   114
A8    5.0                                                              CLS6   115
      CALL    GETXOP  IMM1    SREG2   ABUS.                            CLS6   116
```

```
     XOR      SREG0    SPEG0  DBUS                                          CLS6   117
     CALL     NZR      SREG0  SREG5  SREG6  SREG7.                         CLS6   118
     ENDINSTR                                                              CLS6   119
EOR8X - EXCLUSIVE OR ACC8 W/ MEMORY                                        CLS6   120
E8    5.0                                                                  CLS6   121
     CALL     GETXOP   IMM1   SREG2  ABUS.                                 CLS6   122
     XOR      SREG1    SPEG1  DBUS                                         CLS6   123
     CALL     NZR      SREG1  SREG5  SREG6  SREG7.                         CLS6   124
     ENDINSTR                                                              CLS6   125
INCX - INCREMENT MEMORY                                                    CLS6   126
6C    7.0                                                                  CLS6   127
     CALL     GETXOP   IMM1   SREG2  ABUS.                                 CLS6   128
     CALL     SETV     DBUS   SREG7  7F+16.                                CLS6   129
     INCR     DBUS     DBUS                                                CLS6   130
     AND      DBUS     DBUS   FF+16                                        CLS6   131
     WRITED                                                                CLS6   132
     CALL     NZR      DBUS   SREG5  SREG6  TEMP7.                         CLS6   133
     ENDINSTR                                                              CLS6   134
LDAAX - LOAD ACCA FROM MEMORY                                              CLS6   135
A6    5.0                                                                  CLS6   136
     CALL     GETXOP   IMM1   SREG2  ABUS.                                 CLS6   137
     MOVE     SREG0    DBUS                                               CLS6   138
     CALL     NZR      SREG0  SREG5  SREG6  SREG7.                         CLS6   139
     ENDINSTR                                                              CLS6   140
LDA8X - LOAD ACC8 FROM MEMORY                                              CLS6   141
E6    5.0                                                                  CLS6   142
     CALL     GETXOP   IMM1   SREG2  ABUS.                                 CLS6   143
     MOVE     SREG1    DBUS                                               CLS6   144
     CALL     NZR      SREG1  SREG5  SREG6  SREG7.                         CLS6   145
     ENDINSTR                                                              CLS6   146
ORAAX - (ACCA := ACCA .OR. M)                                              CLS6   147
AA    5.0                                                                  CLS6   148
     CALL     GETXOP   IMM1   SREG2  ABUS.                                 CLS6   149
     OR       SREG0    SREG0  DBUS                                         CLS6   150
     CALL     NZR      SREG0  SREG5  SREG6  SREG7.                         CLS6   151
     ENDINSTR                                                              CLS6   152
ORA8X - (ACC8 := ACC8 .OR. M)                                              CLS6   153
EA    5.0                                                                  CLS6   154
     CALL     GETXOP   IMM1   SREG2  ABUS.                                 CLS6   155
     OR       SREG1    SREG1  DBUS                                         CLS6   156
     CALL     NZR      SREG1  SREG5  SREG6  SREG7.                         CLS6   157
     ENDINSTR                                                              CLS6   158
ROLX - ROTATE LEFT, MEMORY                                                 CLS6   159
69    7.0                                                                  CLS6   160
     CALL     GETXOP   IMM1   SREG2  ABUS.                                 CLS6   161
     SHLL     TEMP1    DBUS   1                                           CLS6   162
     OR       TEMP1    TEMP1  SREG8                                        CLS6   163
     CALL     NZVC8    TEMP1  DBUS   SREG5  SREG6  SREG7  SREG8  80+16.   CLS6   164
     MOVE     DBUS     TEMP1                                              CLS6   165
     WRITED                                                                CLS6   166
     ENDINSTR                                                              CLS6   167
RORX - ROTATE RIGHT, MEMORY                                                CLS6   168
66    7.0                                                                  CLS6   169
     CALL     GETXOP   IMM1   SREG2  ABUS.                                 CLS6   170
     SHRL     TEMP1    DBUS   1                                           CLS6   171
     IF       SREG8    NE     0                                           CLS6   172
              OR       TEMP1  TEMP1  80+16                                CLS6   173
     ENDIF                                                                 CLS6   174
```

```
     CALL     NZVCB    TEMP1   DBUS    SREG5   SREG6   SREG7   SREG8   1.          CLS6   175
     MOVE     DBUS     TEMP1                                                        CLS6   176
     WRITED                                                                        CLS6   177
     ENDINSTR                                                                      CLS6   178
ASLX - SHIFT LEFT, ARITHMETIC, MEMORY                                              CLS6   179
68     7.0                                                                         CLS6   180
     CALL     GETXOP   IMM1    SREG2   ABUS.                                      CLS6   181
     SHLL     TEMP1    DBUS    1                                                   CLS6   182
     CALL     NZVCB    TEMP1   DBUS    SREG5   SREG6   SREG7   SREG8   80*16.      CLS6   183
     MOVE     DBUS     TEMP1                                                       CLS6   184
     WRITED                                                                        CLS6   185
     ENDINSTR                                                                      CLS6   186
ASRX - SHIFT RIGHT, ARITHMETIC, MEMORY                                             CLS6   187
67     7.0                                                                         CLS6   188
     CALL     GETXOP   IMM1    SREG2   ABUS.                                      CLS6   189
     SHRA     TEMP1    DBUS    1                                                   CLS6   190
     CALL     NZVCB    TEMP1   DBUS    SREG5   SREG6   SREG7   SREG8   1.          CLS6   191
     MOVE     DBUS     TEMP1                                                       CLS6   192
     WRITED                                                                        CLS6   193
     ENDINSTR                                                                      CLS6   194
LSRX - LOGICAL SHIFT RIGHT, MEMORY                                                 CLS6   195
64     7.0                                                                         CLS6   196
     CALL     GETXOP   IMM1    SREG2   ABUS.                                      CLS6   197
     SHRL     TEMP1    DBUS    1                                                   CLS6   198
     CALL     NZVCB    TEMP1   DBUS    SREG5   SREG6   SREG7   SREG8   1.          CLS6   199
     MOVE     DBUS     TEMP1                                                       CLS6   200
     WRITED                                                                        CLS6   201
     ENDINSTR                                                                      CLS6   202
TSTX - TEST MEMORY                                                                 CLS6   203
6D     7.0                                                                         CLS6   204
     CALL     GETXOP   IMM1    SREG2   ABUS.                                      CLS6   205
     CALL     NZR      DBUS    SREG5   SREG6   SREG7.                             CLS6   206
     CLEAR    SREG8                                                               CLS6   207
     ENDINSTR                                                                      CLS6   208
STAAX - STORE ACCA INTO MEMORY                                                     CLS6   209
A7     6.0                                                                         CLS6   210
     CALL     GETXOP   IMM1    SREG2   ABUS.                                      CLS6   211
     MOVE     DBUS     SREG0                                                      CLS6   212
     WRITED                                                                        CLS6   213
     CALL     NZR      DBUS    SREG5   SREG6   SREG7.                             CLS6   214
     ENDINSTR                                                                      CLS6   215
STABX - STORE ACCB INTO MEMORY                                                     CLS6   216
E7     6.0                                                                         CLS6   217
     CALL     GETXOP   IMM1    SREG2   ABUS.                                      CLS6   218
     MOVE     DBUS     SREG1                                                      CLS6   219
     WRITED                                                                        CLS6   220
     CALL     NZR      DBUS    SREG5   SREG6   SREG7.                             CLS6   221
     ENDINSTR                                                                      CLS6   222
SUBAX - (ACCA := ACCA - M)                                                         CLS6   223
A0     5.0                                                                         CLS6   224
     CALL     GETXOP   IMM1    SREG2   ABUS.                                      CLS6   225
     CALL     SUBTR    TEMP1   SREG0   DBUS.                                      CLS6   226
     CALL     NZVCS    TEMP1   SREG0   DBUS    SREG5   SREG6   SREG7   SREG8.    CLS6   227
     MOVE     SREG0    TEMP1                                                      CLS6   228
     ENDINSTR                                                                      CLS6   229
SUBBX - (ACCB := ACCB - M)                                                         CLS6   230
E0     5.0                                                                         CLS6   231
     CALL     GETXOP   IMM1    SREG2   ABUS.                                      CLS6   232
```

```
     CALL     SUBTR    TEMP1  SREG1  DBUS.                                    CLS6   233
     CALL     NZVCS    TEMP1  SREG1  DBUS   SREG5  SREG6  SREG7  SREG8.     CLS6   234
     MOVE     SREG1    TEMP1                                                   CLS6   235
     ENDINSTR                                                                 CLS6   236
SBCAX - (ACCA:=ACCA - M - CARRY)                                              CLS6   237
A2     5.3                                                                    CLS6   238
     CALL     GETXOP   IMM1   SREG2  ABUS.                                    CLS6   239
     CALL     SUBTR    TEMP1  SREG0  DBUS.                                    CLS6   240
     CALL     SUBTR    TEMP2  TEMP1  SREG8.                                   CLS6   241
     CALL     NZVCS    TEMP2  SREG0  DBUS   SREG5  SREG6  SREG7  SREG8.     CLS6   242
     MOVE     SREG0    TEMP2                                                   CLS6   243
     ENDINSTR                                                                 CLS6   244
SBCBX - (ACCB:=ACCB - M - CARRY)                                              CLS6   245
E2     5.3                                                                    CLS6   246
     CALL     GETXOP   IMM1   SREG2  ABUS.                                    CLS6   247
     CALL     SUBTR    TEMP1  SREG1  DBUS.                                    CLS6   248
     CALL     SUBTR    TEMP2  TEMP1  SREG8.                                   CLS6   249
     CALL     NZVCS    TEMP2  SREG1  DBUS   SREG5  SREG6  SREG7  SREG8.     CLS6   250
     MOVE     SREG1    TEMP2                                                   CLS6   251
     ENDINSTR                                                                 CLS6   252
CPXX - COMPARE IX W/ MEMORY                                                   CLS6   253
AC     6.0                                                                    CLS6   254
     CALL     GETXOP   IMM1   SREG2  ABUS.                                    CLS6   255
     MOVE     TEMP1    DBUS                                                   CLS6   256
     INCR     ABUS     ABUS                                                   CLS6   257
     READD                                                                    CLS6   258
     MOVE     TEMP2    DBUS                                                   CLS6   259
     CONCAT   TEMP3    TEMP1  (8)     TEMP2                                   CLS6   260
     CALL     SUBTLO   TEMP4  SREG2   TEMP3.                                  CLS6   261
     AND      TEMP1    TEMP4  8000+16                                         CLS6   262
     CLEAR    SREG5                                                           CLS6   263
     CLEAR    SREG7                                                           CLS6   264
     IF       TEMP1    NE     0                                               CLS6   265
              MOVE     SREG5  1                                               CLS6   266
              MOVE     SREG7  1                                               CLS6   267
     ENDIF                                                                    CLS6   268
     CALL     ZERO     TEMP4  SREG6.                                          CLS6   269
     ENDINSTR                                                                 CLS6   270
LDXX - LOAD IX FROM MEMORY                                                    CLS6   271
EE     6.0                                                                    CLS6   272
     CALL     GETXOP   IMM1   SREG2  ABUS.                                    CLS6   273
     MOVE     TEMP1    DBUS                                                   CLS6   274
     INCR     ABUS     ABUS                                                   CLS6   275
     READD                                                                    CLS6   276
     MOVE     TEMP2    DBUS                                                   CLS6   277
     CONCAT   SREG2    TEMP1  (8)     TEMP2                                   CLS6   278
     CALL     NZVLO    SREG2  SREG5  SREG6  SREG7.                            CLS6   279
     ENDINSTR                                                                 CLS6   280
LDSX - LOAD STACK POINTER FROM MEMORY                                         CLS6   281
AE     6.0                                                                    CLS6   282
     CALL     GETXOP   IMM1   SREG2  ABUS.                                    CLS6   283
     MOVE     TEMP1    DBUS                                                   CLS6   284
     INCR     ABUS     ABUS                                                   CLS6   285
     READD                                                                    CLS6   286
     MOVE     TEMP2    DBUS                                                   CLS6   287
     CONCAT   STACKP   TEMP1  (8)     TEMP2                                   CLS6   288
     CALL     NZVLO    STACKP SREG5  SREG6  SREG7.                            CLS6   289
     ENDINSTR                                                                 CLS6   290
```

```
STXx - STORE IX INTO MEMORY                                                    CLS6   291
EF     7.0                                                                     CLS6   292
     ADD      ABUS     SREG2  IMM1                                            CLS6   293
     AND      ABUS     ABUS   FFFF#16                                          CLS6   294
     CALL     STOLOP   SREG2  ABUS   ABUS   DBUS.                             CLS6   295
     CALL     NZVLO    SREG2  SREG5  SREG6  SREG7.                            CLS6   296
     ENDINSTR                                                                  CLS6   297
STSX - STORE STACK POINTER INTO MEMORY                                         CLS6   298
AF     7.0                                                                     CLS6   299
     ADD      ABUS     SREG2  IMM1                                            CLS6   300
     AND      ABUS     ABUS   FFFF#16                                          CLS6   301
     CALL     STOLOP   STACKP ABUS   ABUS   DBUS.                             CLS6   302
     CALL     NZVLO    STACKP SREG5  SREG6  SREG7.                            CLS6   303
     ENDINSTR                                                                  CLS6   304
JMPX - JUMP, INDEXED                                                           CLS6   305
6E     4.0                                                                     CLS6   306
     ADD      PC       SREG2  IMM1                                            CLS6   307
     AND      PC       PC     FFFF#16                                         CLS6   308
     ENDINSTR                                                                  CLS6   309
JSRX - JUMP TO SUBROUTINE                                                      CLS6   310
AD     8.0                                                                     CLS6   311
     MOVE     TEMP1  PC                                                       CLS6   312
     AND      TEMP3    TEMP1  FF#16                                           CLS6   313
     AND      TEMP4    TEMP1  FF00#16                                         CLS6   314
     SHRL     TEMP4    TEMP4  8                                               CLS6   315
     PUSH     TEMP3                                                           CLS6   316
     PUSH     TEMP4                                                           CLS6   317
     ADD      PC       SREG2  IMM1                                            CLS6   318
     AND      PC       PC     FFFF#16                                         CLS6   319
     ENDINSTR                                                                  CLS6   320
ENDINSDEF                                                                      CLS6   321
DEFINE         GETXOP   TEMP1  TEMP2  TEMP3.                                   CLS6   322
     ADD      TEMP3    TEMP1  TEMP2                                           CLS6   323
     AND      TEMP3    TEMP3  FFFF#16                                         CLS6   324
     READD                                                                     CLS6   325
     RETURN                                                                    CLS6   326
     ENDINSTR                                                                  CLS6   327
ENDCLASS                                                                       CLS6   328
```

```
   7   40    1                                                                  CLS7     2
  16                                                                            CLS7     3
   0                                                                            CLS7     4
   9                                                                            CLS7     5
ADDAE - (ACCA := ACCA + M)                                                      CLS7     6
B8    4.0                                                                       CLS7     7
      CALL    GETEOP  IMM1   ABUS.                                              CLS7     8
      ADD     TEMP1   SREG0  DBUS                                               CLS7     9
      AND     TEMP1   TEMP1  FF+16                                              CLS7    10
      CALL    HNZVCA  TEMP1  SREG0  DBUS   SREG3  SREG5  SREG6  SREG7  SREG8.  CLS7    11
      MOVE    SREG0   TEMP1                                                     CLS7    12
      ENDINSTR                                                                  CLS7    13
ADDBE - (ACCB := ACCB + M)                                                      CLS7    14
F8    4.0                                                                       CLS7    15
      CALL    GETEOP  IMM1   ABUS.                                              CLS7    16
      ADD     TEMP1   SREG1  DBUS                                               CLS7    17
      AND     TEMP1   TEMP1  FF+16                                              CLS7    18
      CALL    HNZVCA  TEMP1  SREG1  DBUS   SREG3  SREG5  SREG6  SREG7  SREG8.  CLS7    19
      MOVE    SREG1   TEMP1                                                     CLS7    20
      ENDINSTR                                                                  CLS7    21
ADCAE - (ACCA := ACCA + M + CARRY)                                              CLS7    22
B9    4.0                                                                       CLS7    23
      CALL    GETEOP  IMM1   ABUS.                                              CLS7    24
      ADD     TEMP1   SREG0  DBUS                                               CLS7    25
      ADD     TEMP1   TEMP1  SREG8                                              CLS7    26
      AND     TEMP1   TEMP1  FF+16                                              CLS7    27
      CALL    HNZVCA  TEMP1  SREG0  DBUS   SREG3  SREG5  SREG6  SREG7  SREG8.  CLS7    28
      MOVE    SREG0   TEMP1                                                     CLS7    29
      ENDINSTR                                                                  CLS7    30
ADCBE - (ACCB := ACCB + M + CARRY)                                              CLS7    31
F9    4.0                                                                       CLS7    32
      CALL    GETEOP  IMM1   ABUS.                                              CLS7    33
      ADD     TEMP1   SREG1  DBUS                                               CLS7    34
      ADD     TEMP1   TEMP1  SREG8                                              CLS7    35
      AND     TEMP1   TEMP1  FF+16                                              CLS7    36
      CALL    HNZVCA  TEMP1  SREG1  DBUS   SREG3  SREG5  SREG6  SREG7  SREG8.  CLS7    37
      MOVE    SREG1   TEMP1                                                     CLS7    38
      ENDINSTR                                                                  CLS7    39
ANDAE - (ACCA := ACCA .AND. M)                                                  CLS7    40
B4    4.0                                                                       CLS7    41
      CALL    GETEOP  IMM1   ABUS.                                              CLS7    42
      AND     SREG0   SREG0  DBUS.                                              CLS7    43
      CALL    NZR     SREG0  SREG5  SREG6  SREG7.                               CLS7    44
      ENDINSTR                                                                  CLS7    45
ANDBE - (ACCB := ACCB .AND. M)                                                  CLS7    46
F4    4.0                                                                       CLS7    47
      CALL    GETEOP  IMM1   ABUS.                                              CLS7    48
      AND     SREG1   SREG1  DBUS                                               CLS7    49
      CALL    NZR     SREG1  SREG5  SREG6  SREG7.                               CLS7    50
      ENDINSTR                                                                  CLS7    51
BITAE - (ACCA .AND. MEMORY)                                                     CLS7    52
B5    4.0                                                                       CLS7    53
      CALL    GETEOP  IMM1   ABUS.                                              CLS7    54
      AND     TEMP1   SREG0  DBUS                                               CLS7    55
      CALL    NZR     TEMP1  SREG5  SREG6  SREG7.                               CLS7    56
      ENDINSTR                                                                  CLS7    57
BITBE - (ACCB .AND. MEMORY)                                                     CLS7    58
```

```
F5    4.0                                                                  CLS7   59
      CALL    GETEOP  IMM1    ABUS.                                        CLS7   60
      AND     TEMP1   SREG1   DBUS                                         CLS7   61
      CALL    NZR     TEMP1   SREG5  SREG6  SREG7.                        CLS7   62
      ENDINSTR                                                             CLS7   63
CLRE - CLEAR MEMORY                                                        CLS7   64
7F    6.0                                                                  CLS7   65
      CALL    GETEOP  IMM1    ABUS.                                        CLS7   66
      CLEAR   DBUS                                                         CLS7   67
      WRITED                                                               CLS7   68
      CLEAR   SREG5                                                        CLS7   69
      MOVE    SREG6   1                                                    CLS7   70
      CLEAR   SREG7                                                        CLS7   71
      CLEAR   SREG8                                                        CLS7   72
      ENDINSTR                                                             CLS7   73
CMPAE - (COMPARE ACCA WITH MEMORY)                                         CLS7   74
B1    4.0                                                                  CLS7   75
      CALL    GETEOP  IMM1    ABUS.                                        CLS7   76
      CALL    SUBTR   TEMP1   SREG0  DBUS.                                 CLS7   77
      CALL    NZVCS   TEMP1   SREG0  DBUS   SREG5  SREG6  SREG7  SREG8.   CLS7   78
      ENDINSTR                                                             CLS7   79
CMPBE - (COMPARE ACCB WITH MEMORY)                                         CLS7   80
F1    4.0                                                                  CLS7   81
      CALL    GETEOP  IMM1    ABUS.                                        CLS7   82
      CALL    SUBTR   TEMP1   SREG1  DBUS.                                 CLS7   83
      CALL    NZVCS   TEMP1   SREG0  DBUS   SREG5  SREG6  SREG7  SREG8.   CLS7   84
      ENDINSTR                                                             CLS7   85
COME - 1'S COMPLEMENT MEMORY                                               CLS7   86
73    6.0                                                                  CLS7   87
      CALL    GETEOP  IMM1    ABUS.                                        CLS7   88
      COMONE  DBUS    DBUS                                                 CLS7   89
      AND     DBUS    DBUS    FF+16                                        CLS7   90
      WRITED                                                               CLS7   91
      CALL    NZR     DBUS    SREG5  SREG6  SREG7.                        CLS7   92
      MOVE    SREG8   1                                                    CLS7   93
      ENDINSTR                                                             CLS7   94
NEGE - 2'S COMPLEMENT MEMORY                                               CLS7   95
70    6.0                                                                  CLS7   96
      CALL    GETEOP  IMM1    ABUS.                                        CLS7   97
      COMTWO  DBUS    DBUS                                                 CLS7   98
      AND     DBUS    DBUS    FF+16                                        CLS7   99
      WRITED                                                               CLS7  100
      CALL    NZR     DBUS    SREG5  SREG6  TEMP7.                        CLS7  101
      CALL    SETV    DBUS    SREG7  80+16.                                CLS7  102
      CALL    SETV    SREG6   SREG8  0.                                   CLS7  103
      ENDINSTR                                                             CLS7  104
DECE - DECREMENT MEMORY                                                    CLS7  105
7A    6.0                                                                  CLS7  106
      CALL    GETEOP  IMM1    ABUS.                                        CLS7  107
      CALL    SETV    DBUS    SREG7  80+16.                                CLS7  108
      ADD     DBUS    DBUS    FF+16                                        CLS7  109
      AND     DBUS    DBUS    FF+16                                        CLS7  110
      WRITED                                                               CLS7  111
      CALL    NZR     DBUS    SREG5  SREG6  TEMP7.                        CLS7  112
      ENDINSTR                                                             CLS7  113
EORAE - EXCLUSIVE OR ACCA W/ MEMORY                                        CLS7  114
B8    4.0                                                                  CLS7  115
      CALL    GETEOP  IMM1    ABUS.                                        CLS7  116
```

```
     XOR      SREG0    SREG0  DBUS                                          CLS7   117
     CALL     NZR      SREG0  SREG5  SREG6  SREG7.                         CLS7   118
     ENDINSTR                                                              CLS7   119
EORBE - EXCLUSIVE OR ACCB W/ MEMORY                                        CLS7   120
F8    4.0                                                                  CLS7   121
     CALL     GETEOP   IMM1   ABUS.                                         CLS7   122
     XOR      SREG1    SREG1  DBUS                                          CLS7   123
     CALL     NZR      SREG1  SREG5  SREG6  SREG7.                         CLS7   124
     ENDINSTR                                                              CLS7   125
INCE - INCREMENT MEMORY                                                    CLS7   126
7C    6.0                                                                  CLS7   127
     CALL     GETEOP   IMM1   ABUS.                                         CLS7   128
     CALL     SETV     DBUS   SREG7  7F*16.                                CLS7   129
     INCR     DBUS     DBUS                                                CLS7   130
     AND      DBUS     DBUS   FF*16                                        CLS7   131
     WRITED                                                                CLS7   132
     CALL     NZR      DBUS   SREG5  SREG6  TEMP7.                         CLS7   133
     ENDINSTR                                                              CLS7   134
LDAAE - LOAD ACCA FROM MEMORY                                              CLS7   135
B6    4.0                                                                  CLS7   136
     CALL     GETEOP   IMM1   ABUS.                                         CLS7   137
     MOVE     SREG0    DBUS                                               CLS7   138
     CALL     NZR      SREG0  SREG5  SREG6  SREG7.                         CLS7   139
     ENDINSTR                                                              CLS7   140
LDABE - LOAD ACCB FROM MEMORY                                              CLS7   141
F6    4.0                                                                  CLS7   142
     CALL     GETEOP   IMM1   ABUS.                                         CLS7   143
     MOVE     SREG1    DBUS                                               CLS7   144
     CALL     NZR      SREG1  SREG5  SREG6  SREG7.                         CLS7   145
     ENDINSTR                                                              CLS7   146
ORAAE - (ACCA := ACCA .OR. M)                                              CLS7   147
BA    4.0                                                                  CLS7   148
     CALL     GETEOP   IMM1   ABUS.                                         CLS7   149
     OR       SREG0    SREG0  DBUS                                         CLS7   150
     CALL     NZR      SREG0  SREG5  SREG6  SREG7.                         CLS7   151
     ENDINSTR                                                              CLS7   152
ORABE - (ACCB := ACCB .OR. M)                                              CLS7   153
FA    4.0                                                                  CLS7   154
     CALL     GETEOP   IMM1   ABUS.                                         CLS7   155
     OR       SREG1    SREG1  DBUS                                         CLS7   156
     CALL     NZR      SREG1  SREG5  SREG6  SREG7.                         CLS7   157
     ENDINSTR                                                              CLS7   158
ROLE - ROTATE LEFT, MEMORY                                                 CLS7   159
79      6.0                                                                CLS7   160
     CALL     GETEOP   IMM1   ABUS.                                         CLS7   161
     SHLL     TEMP1    DBUS   1                                           CLS7   162
     OR       TEMP1    TEMP1  SREG8                                       CLS7   163
     CALL     NZVC8    TEMP1  DBUS   SREG5  SREG6  SREG7  SREG8  80*16.   CLS7   164
     MOVE     DBUS     TEMP1                                              CLS7   165
     WRITED                                                                CLS7   166
     ENDINSTR                                                              CLS7   167
RORE - ROTATE RIGHT, MEMORY                                                CLS7   168
76      6.0                                                                CLS7   169
     CALL     GETEOP   IMM1   ABUS.                                         CLS7   170
     SHRL     TEMP1    DBUS   1                                           CLS7   171
     IF       SREG8    NE     0                                           CLS7   172
              OR       TEMP1  TEMP1  80*16                                CLS7   173
     ENDIF                                                                 CLS7   174
```

```
     CALL    NZVCB   TEMP1  DBUS   SREG5  SREG6  SREG7  SREG8  1.          CLS7   175
     MOVE    DBUS    TEMP1                                                 CLS7   176
     WRITED                                                                CLS7   177
     ENDINSTR                                                              CLS7   178
ASLE - SHIFT LEFT, ARITHMETIC, MEMORY                                      CLS7   179
78     6.0                                                                 CLS7   180
     CALL    GETEOP  IMM1   ABUS.                                          CLS7   181
     SHLL    TEMP1   DBUS   1                                              CLS7   182
     CALL    NZVCB   TEMP1  DBUS   SREG5  SREG6  SREG7  SREG8  80+16.      CLS7   183
     MOVE    DBUS    TEMP1                                                 CLS7   184
     WRITED                                                                CLS7   185
     ENDINSTR                                                              CLS7   186
ASRE - SHIFT RIGHT, ARITHMETIC, MEMORY                                     CLS7   187
77     6.0                                                                 CLS7   188
     CALL    GETEOP  IMM1   ABUS.                                          CLS7   189
     SHRA    TEMP1   DBUS   1                                              CLS7   190
     CALL    NZVCB   TEMP1  DBUS   SREG5  SREG6  SREG7  SREG8  1.          CLS7   191
     MOVE    DBUS    TEMP1                                                 CLS7   192
     WRITED                                                                CLS7   193
     ENDINSTR                                                              CLS7   194
LSRE - LOGICAL SHIFT RIGHT, MEMORY                                         CLS7   195
74     6.0                                                                 CLS7   196
     CALL    GETEOP  IMM1   ABUS.                                          CLS7   197
     SHRL    TEMP1   DBUS   1                                              CLS7   198
     CALL    NZVCB   TEMP1  DBUS   SREG5  SREG6  SREG7  SREG8  1.          CLS7   199
     MOVE    DBUS    TEMP1                                                 CLS7   200
     WRITED                                                                CLS7   201
     ENDINSTR                                                              CLS7   202
TSTE - TEST MEMORY                                                         CLS7   203
7D     6.0                                                                 CLS7   204
     CALL    GETEOP  IMM1   ABUS.                                          CLS7   205
     CALL    NZR     DBUS   SREG5  SREG6  SREG7.                           CLS7   206
     CLEAR   SREG8                                                         CLS7   207
     ENDINSTR                                                              CLS7   208
STAAE - STORE ACCA INTO MEMORY                                             CLS7   209
B7     6.0                                                                 CLS7   210
     CALL    GETEOP  IMM1   ABUS.                                          CLS7   211
     MOVE    DBUS    SREG0                                                 CLS7   212
     WRITED                                                                CLS7   213
     CALL    NZR     DBUS   SREG5  SREG6  SREG7.                           CLS7   214
     ENDINSTR                                                              CLS7   215
STABE - STORE ACCB INTO MEMORY                                             CLS7   216
F7     6.0                                                                 CLS7   217
     CALL    GETEOP  IMM1   ABUS.                                          CLS7   218
     MOVE    DBUS    SREG1                                                 CLS7   219
     WRITED                                                                CLS7   220
     CALL    NZR     DBUS   SREG5  SREG6  SREG7.                           CLS7   221
     ENDINSTR                                                              CLS7   222
SUBAE - (ACCA := ACCA - M)                                                 CLS7   223
B0     4.0                                                                 CLS7   224
     CALL    GETEOP  IMM1   ABUS.                                          CLS7   225
     CALL    SUBTR   TEMP1  SREG0  DBUS.                                   CLS7   226
     CALL    NZVCS   TEMP1  SREG0  DBUS   SREG5  SREG6  SREG7  SREG8.      CLS7   227
     MOVE    SREG0   TEMP1                                                 CLS7   228
     ENDINSTR                                                              CLS7   229
SUBBE - (ACCB := ACCB - M)                                                 CLS7   230
F0     4.0                                                                 CLS7   231
     CALL    GETEOP  IMM1   ABUS.                                          CLS7   232
```

```
     CALL     SUBTR    TEMP1  SREG1  DBUS.                                    CLS7   233
     CALL     NZVCS    TEMP1  SREG1  DBUS   SREG5  SREG6  SREG7  SREG8.     CLS7   234
     MOVE     SREG1    TEMP1                                                   CLS7   235
     ENDINSTR                                                                 CLS7   236
SBCAE - (ACCA:=ACCA - M - CARRY)                                              CLS7   237
B2     4.0                                                                    CLS7   238
     CALL     GETEOP   IMM1   ABUS.                                           CLS7   239
     CALL     SUBTR    TEMP1  SREG0  DBUS.                                    CLS7   240
     CALL     SUBTR    TEMP2  TEMP1  SREG8.                                   CLS7   241
     CALL     NZVCS    TEMP2  SREG0  DBUS   SREG5  SREG6  SREG7  SREG8.     CLS7   242
     MOVE     SREG0    TEMP2                                                   CLS7   243
     ENDINSTR                                                                 CLS7   244
SBCBE - (ACCB :=ACCB - M - CARRY)                                             CLS7   245
F2     4.0                                                                    CLS7   246
     CALL     GETEOP   IMM1   ABUS.                                           CLS7   247
     CALL     SUBTR    TEMP1  SREG1  DBUS.                                    CLS7   248
     CALL     SUBTR    TEMP2  TEMP1  SREG8.                                   CLS7   249
     CALL     NZVCS    TEMP2  SREG1  DBUS   SREG5  SREG6  SREG7  SREG8.     CLS7   250
     MOVE     SREG1    TEMP2                                                   CLS7   251
     ENDINSTR                                                                 CLS7   252
CPXE - COMPARE IX W/ MEMORY                                                   CLS7   253
8C     6.0                                                                    CLS7   254
     CALL     GETEOP   IMM1   ABUS.                                           CLS7   255
     MOVE     TEMP1    DBUS                                                   CLS7   256
     INCR     ABUS     ABUS                                                   CLS7   257
     READD                                                                    CLS7   258
     MOVE     TEMP2    DBUS                                                   CLS7   259
     CONCAT   TEMP3    TEMP1  (8)     TEMP2                                   CLS7   260
     CALL     SUBTLO   TEMP4  SREG2   TEMP3.                                  CLS7   261
     AND      TEMP1    TEMP4  8000+16                                         CLS7   262
     CLEAR    SREG5                                                           CLS7   263
     CLEAR    SREG7                                                           CLS7   264
     IF       TEMP1    NE     0                                               CLS7   265
              MOVE     SREG5  1                                               CLS7   266
              MOVE     SREG7  1                                               CLS7   267
     ENDIF                                                                    CLS7   268
     CALL     ZERO     TEMP4  SREG6.                                          CLS7   269
     ENDINSTR                                                                 CLS7   270
LDXE - LOAD IX FROM MEMORY                                                    CLS7   271
FE     6.0                                                                    CLS7   272
     CALL     GETEOP   IMM1   ABUS.                                           CLS7   273
     MOVE     TEMP1    DBUS                                                   CLS7   274
     INCR     ABUS     ABUS                                                   CLS7   275
     READD                                                                    CLS7   276
     MOVE     TEMP2    DBUS                                                   CLS7   277
     CONCAT   SREG2    TEMP1  (8)     TEMP2                                   CLS7   278
     CALL     NZVLO    SREG2  SREG5   SREG6  SREG7.                          CLS7   279
     ENDINSTR                                                                 CLS7   280
LDSE - LOAD STACK POINTER FROM MEMORY                                         CLS7   281
8E       6.0                                                                  CLS7   282
     CALL     GETEOP   IMM1   ABUS.                                           CLS7   283
     MOVE     TEMP1    DBUS                                                   CLS7   284
     INCR     ABUS     ABUS                                                   CLS7   285
     READD                                                                    CLS7   286
     MOVE     TEMP2    DBUS                                                   CLS7   287
     CONCAT   STACKP   TEMP1  (8)     TEMP2                                   CLS7   288
     CALL     NZVLO    STACKP SREG5   SREG6  SREG7.                          CLS7   289
     ENDINSTR                                                                 CLS7   290
```

```
STXE - STORE IX INTO MEMORY                                          CLS7   291
FF     6.0                                                           CLS7   292
     CALL     STOLOP  SREG2  IMM1   ABUS   DBUS.                    CLS7   293
     CALL     NZVLO   SREG2  SREG5  SREG6  SREG7.                   CLS7   294
     ENDINSTR                                                        CLS7   295
STSE - STORE STACK POINTER INTO MEMORY                               CLS7   296
BF     6.0                                                           CLS7   297
     CALL     STOLOP  STACKP IMM1   ABUS   DBUS.                    CLS7   298
     CALL     NZVLO   STACKP SREG5  SREG6  SREG7.                   CLS7   299
     ENDINSTR                                                        CLS7   300
JMPE - JUMP, EXTENDED                                                CLS7   301
7E     4.0                                                           CLS7   302
     MOVE     PC      IMM1                                          CLS7   303
     ENDINSTR                                                        CLS7   304
JSRE - JUMP TO SUBROUTINE                                            CLS7   305
BD     9.0                                                           CLS7   306
     MOVE     TEMP1  PC                                             CLS7   307
     AND      TEMP3  TEMP1  FF#16                                   CLS7   308
     AND      TEMP4  TEMP1  FF00#16                                 CLS7   309
     SHRL     TEMP4  TEMP4  8                                       CLS7   310
     PUSH     TEMP3                                                 CLS7   311
     PUSH     TEMP4                                                 CLS7   312
     MOVE     PC      IMM1                                          CLS7   313
     ENDINSTR                                                        CLS7   314
ENDINSDEF                                                            CLS7   315
DEFINE        GETEOP  TEMP1  TEMP2.                                  CLS7   316
     MOVE     TEMP2   TEMP1                                         CLS7   317
     READD                                                           CLS7   318
     RETURN                                                          CLS7   319
     ENDINSTR                                                        CLS7   320
ENDCLASS                                                             CLS7   321
```

BIOGRAPHY

Ramon Tan was born on January 25, 1953 to TAN Hing Liong and SIAO Muy Tie in Cebu City, Cebu, Philippines. In 1968 he entered the University of the Philippines at Diliman, Quezon City, Philippines, where he majored in mathematics. He was graduated from the same university in April, 1972 with cum laude honors in the mathematics class. Subsequently, he joined the faculty of mathematics of the De La Salle College in Manila, Philippines, where he taught until May, 1974. From July 1974 to May 1976 he was a graduate student of computer science in the Mathematics Department of Lehigh University, and also Graduate Assistant at the Lehigh University Computing Center. Since June, 1976 he has been programmer/analyst for the Allentown and Sacred Heart Computer Center in Allentown, Pennsylvania.