HECTOR 1600
PROGRAMMER'S GUIDE

Chapter 1: HECTOR 1600 Programmer's Model

This chapter contains a general description of the HECTOR 1600 microprocessor. HECTOR's branching modes, addressing modes, and timing data are described and listed. The following symbols are used in these descriptions:

REG	Register direct addressing
IND	Register indirect addressing
INC	Register indirect with post increment addressing
NDX	Indexed addressing
CC	Condition Codes
BM	Branch Mode

Overview:

The HECTOR 1600 is a 16-bit microprocessor built in a 4-month timespan by five graduate students and one faculty advisor. Due to the short design cycle and engineering inexperience, HECTOR was built as simply as possible. However, the simple design not only enabled the machine to be built by the required deadline, but also caused the instruction set to become orthogonal, powerful, and easy-to-use.

Programmer's Model:

The programmer's model for HECTOR is shown below.

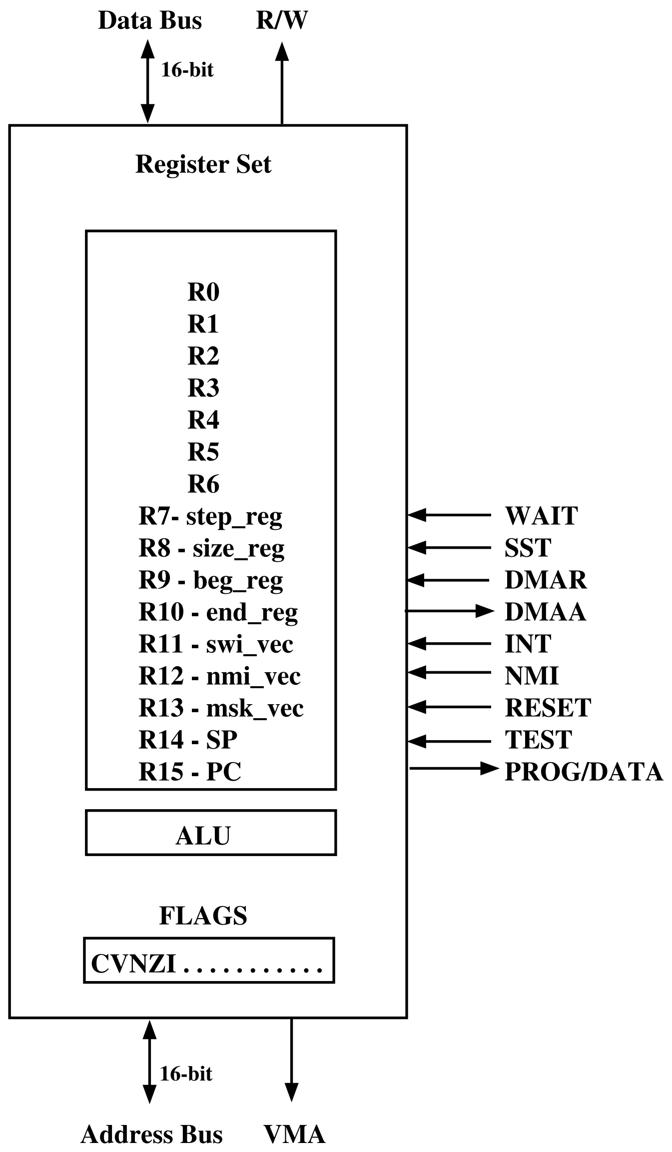

Figure 1: Programmer's Model

HECTOR interfaces to the world through a 16-bit bi-directional data bus and a 16-bit uni-directional address bus. Additionally, there are pins available for interrupts, reset, self-test, single-stepping, wait-states, direct memory access, and program/data memory space separation. For the most part, we will consider HECTOR to be a processor which can access 16-bit words from a 64K-word memory.

Internally, HECTOR contains a 16-bit Arithmetic Logic Unit (ALU), five ALU status flags, and sixteen 16-bit registers. The ALU is capable of performing the following functions in conjunction with the register set:

1) Add
2) Add with carry
3) Subtract
4) Subtract with carry
5) Bit-wise logical AND
6) Bit-wise logical OR
7) Bit-wise logical XOR
8) Bit-wise logical NOT
9) Increment
10) Decrement
11) Arithmetic shift right
12) Arithmetic shift left
13) Logical rotate right
14) Logical rotate left
15) Byte-swapping
16) Two's-complement negation
17) Loading the ALU flags
18) Storing the ALU flags
19) Clearing

The flags associated with the ALU are:

1) Carry flag = C
2) Overflow flag = V
3) Negative flag = N
4) Zero flag = Z
5) Interrupt enable flag = I

These five flags operate in the same manner as those found on common microprocessors such as the Motorola 6800.

The register file contains seven general-purpose registers (R0-R6) and nine registers with pre-assigned functions (R7-R15). These are described below:

R7	step_reg	=	Size of search list structure
R8	size_reg	=	Size of memory exchange block
R9	beg_reg	=	Pointer to beginning of search list
R10	end_reg	=	Pointer to end of search list
R11	swi_vec	=	Pointer to software interrupt routine
R12	nmi_vec	=	Pointer to non-maskable interrupt routine
R13	msk_vec	=	Pointer to maskable interrupt routine
R14	SP	=	Stack Pointer
R15	PC	=	Program Counter

Important: It should be emphasized that while R7-R15 have pre-assigned functions, they can be used in the same way as the general-purpose registers (R0-R6). Also, if you decide not to use certain capabilities of the processor, then the registers associated with that function can be thought of as general-purpose registers. For example, if non-maskable interrupts will not be used in a particular application, then nmi_vec (R12) becomes a general purpose register. The register codes are listed in Table 1.

Code	Register Name/Function
0000	R0
0001	R1
0010	R2
0011	R3
0100	R4
0101	R5
0110	R6
0111	R7 - step_reg
1000	R8 - size_reg
1001	R9 - beg_reg
1010	R10 - end_reg
1011	R11 - swi_vic
1100	R12 - nmi_vec
1101	R13 - msk_vec
1110	R14 - SP
1111	R15 - PC

Table 1: Register Coding Summary

Addressing Modes:

HECTOR supports four fundamental addressing modes: register direct, register indirect, indirect with post-increment, and indexed. Each is briefly described below.

Register direct	-	The operand is the contents of one of the 16 general registers.
Register Indirect	-	The operand is the contents of the memory location pointed to by one of the 16 general registers.
Indirect Post-inc.	-	Same as Register Indirect, but the contents of the general register is incremented after it is used to access the operand in memory.

Indexed - The operand is the contents of the memory location pointed to by the sum of the contents of one of the 16 general registers and a 16-bit displacement constant specified in an extension word following the operand word.

Table 2 summarizes the coding and assembler conventions for these four addressing modes:

Code	Mode	Asm. Syntax	Example
00	Register Direct	Rn	MOVE R0, R1
01	Register Indirect	[Rn]	MOVE [R1], [R7]
10	Indirect Post-Inc.	[Rn+]	CLR [R10+]
11	Indexed	[Rn, d16]	TEST [R1, 03ffeh]

Table 2: HECTOR Addressing Modes

Branching Modes:

HECTOR supports four modes of specifying the control transfer address for branch (BRA) and jump-to-subroutine (JSR) instructions. These modes are described below:

Register Direct - Control is transferred to the location pointed to by one of the 16 general registers.

Absolute - Control is transferred to the location pointed to by the contents of a 16-bit extension word following the operand word.

Relative - Control is transferred to the location pointed to by the sum of the contents of the Program Counter (PC / R15) and a 16-bit extension word following the operand word.

Indexed - Control is transferred to the location pointed to by the contents of the memory location whose address is the sum of the contents of one of the 16 general registers and a 16-bit extension word following the operand word.

Table 3 summarizes the coding and assembler conventions for the HECTOR branch/jump modes.

Code	Mode	Asm. Syntax	Example
00	Register Direct	Rn	BRA R1
01	Absolute	@label	JSR @entry_point
10	Relative	label	JSR entry_point
11	Indexed	[Rn, label]	BRA [R5, jmp_tbl]

Table 3: HECTOR Branching Modes

All HECTOR branch (BRA) and jump-to-subroutine (JSR) instructions are conditional. Whether or not control is transferred to the address computed by the branch mode (BM) is determined by one of sixteen combinatorial functions of the condition flag bits N, V, C, and Z. The condition selected is specified by a 4-bit code listed in Table 4. The assembler accepts the mnemonic codes listed in the rightmost column. These codes follow the branch mode field and are separated from it by a comma. For example, the instruction

BRA @entry_point, .ge

causes control to be transferred to the location *entry_point* if the function $N \oplus V$ evaluates to 0. Note that there are two special conditions, 1 = 0, which always evaluates false, and 1 = 1, which always evalutates true. The latter is useful for unconditional branches and subroutines, while the former essentially generates *no-op* instructions.

CC	Name	Condition	M6800 Equiv.	Assembler
0000	overflow clear	$V = 0$	BVC	.VC
0001	plus	$N = 0$	BPL	.PL
0010	greater or equal	$N \oplus V = 0$	BGE	.GE
0011	nop	$1 = 0$	BRN	.F
0100	less or equal	$(N \oplus V) \cup Z = 1$	BLE	.LE
0101	not equal	$Z = 0$	BNE	.NE
0110	low or same	$C \cup Z = 1$	BLS	.LS
0111	carry clear	$C = 0$	BCC	.CC
1000	overflow set	$V = 1$	BVS	.VS
1001	minus	$N = 1$	BMI	.MI
1010	less than	$N \oplus V = 1$	BLT	.LT
1011	branch always	$1 = 1$	BRA	.T
1100	greater than	$(N \oplus V) \cup Z = 0$	BGT	.GT
1101	equal	$Z = 1$	BEQ	.EQ
1110	high	$C \cup Z = 0$	BHI	.HI
1111	carry set	$C = 1$	BCS	.CS

Table 4: Branching Condition Codes

Timing:

The following tables lists the HECTOR 1600 instruction execution times in clock cycles according the *instruction group*. The instruction group number can be found in chapter 2. For Groups 1, 3, and 8, the columns are the destination address mode and the rows are the source address mode. For Groups 4 and 5, the first number represents clock cycles required if the jump or branch is performed (condition is met) and the second number is if execution continues with the following instruction (condition is not met, so no branch or jump).

DM \ SM	REG	IND	INC	NDX
REG	2	3	4	6
IND	4	5	6	8
INC	4	5	6	8
NDX	6	7	8	10

GROUP 1

REG	IND	INC	NDX
2	4	4	6

GROUP 2

DM \ SM	REG	IND	INC	NDX
REG	2	3	4	6
IND	3	4	5	7
INC	4	5	6	8
NDX	6	7	8	10

GROUP 3

REG	IND	INC	NDX
3/3	3/3	4/3	4/3

GROUP 4

REG	IND	INC	NDX
5/3	6/3	6/3	7/3

GROUP 5

REG	IND	INC	NDX
2	4	4	6

GROUP 6

REG	IND	INC	NDX
2	3	3	5

GROUP 7

DM \ SM	REG	IND	INC	NDX
REG	2	2	3	5
IND	2	3	4	6
INC	2	3	4	6
NDX	5	6	7	9

GROUP 8

REG	IND	INC	NDX
2	3	4	6

GROUP 9

REG	IND	INC	NDX
2	3	3	5

GROUP 10

REG	IND	INC	NDX
2	3	4	6

GROUP 11

REG	IND	INC	NDX
3	4	5	7

GROUP 12

GROUP 13: 2 cycles

GROUP 14: 6 cycles

GROUP 15: 8 cycles

GROUP 16: (2 + 8 * size_reg) cycles

GROUP 17: (3 + 6 * (# of items examined)) cycles

Chapter 2: HECTOR 1600 Instruction Set

This chapter contains the descriptions of the HECTOR 1600 instruction set. The following symbols are used in the descriptions:

SM	Source operand addressing mode
DM	Destination operand addressing mode
Src	Register used as the source operand or addressing
Dst	Register used as the destination operand or addressing
CC	Condition Codes used in instruction execution branching
PC	Program Counter register
SP	Stack Pointer register
C	Carry Flag
I	Interrupt Enable Flag
N	Negative Flag
V	Overflow Flag
Z	Zero Flag

ADD - Group 1

Operation: Source + Destination → Destinatinon

Assembler Syntax: ADD Src, Dst

Description: Adds the source operand with the destination operand and stores the result in the destination location.

Condition Codes:

C	Set if a carry is generated. Otherwise cleared.
N	Set if the most significant bit of the result (negative value) is 1. Otherwise cleared.
I	Not affected.
V	Set if an overflow occurs. Otherwise cleared.
Z	Set if the result is zero. Otherwise cleared.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
0	0	0	0	SM	DM	Src	Dst

ADDC - Group 1 - add with carry

Operation: Source + Destination + Carry → Destinatinon

Assembler Syntax: ADD Src, Dst

Description: Adds the value of the Carry Flag and the source operand to the destination operand and stores the result in the destination location.

Condition Codes:

C	Set if a carry is generated. Otherwize cleared.
N	Set if the most significant bit of the result (negative value) is 1. Otherwize cleared.
I	Not affected.
V	Set if an overflow occurs. Otherwise cleared.
Z	Set if the result is zero. Otherwise cleared.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
0	0	0	1	SM	DM	Src	Dst

AND - Group 1

Operation: Source $\cap$ Destination $\rightarrow$ Destinatinon

Assembler Syntax: AND Src, Dst

Description: Logically AND's the source operand with the destination operand and stores the result in the destination location.

Condition Codes:

C	Not affected.
N	Set if the most significant bit of the result (negative value) is 1. Otherwise cleared.
I	Not affected.
V	Not affected.
Z	Set if the result is zero. Otherwise cleared.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
0	0	1	1	SM	DM	Src	Dst

BRA - Group 4 - branch

Operation: Evaluate CC
If condition is true
target address $\rightarrow$ PC
Else
no operation

Assembler Syntax: BRA Src, CC

Description: Evaluates the condition. If the result is true (1), then the branch is taken; else, the next sequential instruction is executed.

Condition Codes: No flags are affected.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	1	0	0	BM	00	Src	CC

BTST - Group 3 - bit test

Operation: Source $\cap$ Destination

Assembler Syntax: BTST Src, Dst

Description: Logically AND's the source operand with the destination operand and discards the result.

Condition Codes:

C	Not affected.
N	Set if the most significant bit of the result (negative value) is 1. Otherwise cleared.
I	Not affected.
V	Not affected.
Z	Set if the result is zero. Otherwise cleared.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	0	1	1	SM	DM	Src	Dst

CLC - Group 13 - clear carry

Operation: 0 $\longrightarrow$ Carry Flag

Assembler Syntax: CLC

Description: Sets the Carry bit flag to zero.

Condition Codes:

C	Cleared; Set to zero.
N	Not affected.
I	Not affected.
V	Not affected.
Z	Not affected.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	1	1	1	00	01	- - - -	- - - -

CLI - Group 13 - clear interrupt

Operation: 0 → Interrupt Flag

Assembler Syntax: CLI

Description: Clear the interrupt enable flag.

Condition Codes:

C	Not affected.
N	Not affected.
I	Cleared; Set to zero.
V	Not affected.
Z	Not affected.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	1	1	1	00	11	- - - -	- - - -

CLR - Group 7 - clear

Operation: 0 → Source

Assembler Syntax: CLR Src

Description: Sets the the source operand/location to zero.

Condition Codes:

C	Not affected.
N	Cleared. Set to zero.
I	Not affected.
V	Not affected.
Z	Cleared; Set to zero.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	1	0	0	SM	11	Src	Src

CMP - Group 3 - compare

Operation: Source - Destination

Assembler Syntax: CMP Src, Dst

Description: Subtracts the value of the source operand from the destination operand using two's complement arithmetic to set the flags. The result is discarded.

Condition Codes:

C	Set if a carry is generated. Otherwize cleared.
N	Set if the most significant bit of the result (negative value) is 1. Otherwize cleared.
I	Not affected.
V	Set if an overflow occurs. Otherwise cleared.
Z	Set if the result is zero. Otherwise cleared.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	0	1	0	SM	DM	Src	Src

DEC - Group 2 - decrement

Operation: Source - 1 $\longrightarrow$ Source

Assembler Syntax: DEC Src

Description: Decrements the value of the source operand and stores the result in the source location.

Condition Codes:

C	Set if a carry is generated. Otherwize cleared.
N	Set if the most significant bit of the result (negative value) is 1. Otherwize cleared.
I	Not affected.
V	Set if an overflow occurs. Otherwise cleared.
Z	Set if the result is zero. Otherwise cleared.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	0	0	0	SM	11	Src	Src

INC - Group 2 - increment

Operation: Source + 1 $\longrightarrow$ Source

Assembler Syntax: INC Src

Description: Increments the value of the source operand and stores the result in the source location.

Condition Codes:

C	Set if a carry is generated. Otherwize cleared.
N	Set if the most significant bit of the result (negative value) is 1. Otherwize cleared.
I	Not affected.
V	Set if an overflow occurs. Otherwise cleared.
Z	Set if the result is zero. Otherwise cleared.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	0	0	0	SM	10	Src	Src

JSR - Group 5 - jump to subroutine

Operation:

Evaluate CC
If condition is true
  SP - 1 $\longrightarrow$ SP
  PC $\longrightarrow$ (SP)
  target address $\longrightarrow$ PC
Else
  no operation

Assembler Syntax: JSR Src, CC

Description: Evaluates the condition. If the result is true (1), then the address of the instruction immediately following the JSR instruction is pushed onto the system stack. Program execution continues at the target address specified in the Source location. If the result is false (0), then no operation.

Condition Codes: No flags are affected.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	1	0	0	BM	01	Src	CC

LDF - Group 11 - load flags

Operation: Source $\longrightarrow$ Flags

Assembler Syntax: LDF Src

Description: Loads the Flags register from the source location.

Condition Codes:

C	Set to source value.
N	Set to source value.
I	Set to source value.
V	Set to source value.
Z	Set to source value.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	1	1	0	SM	10	Src	Src

MOVE - Group 8

Operation: Source $\longrightarrow$ Destinatinon

Assembler Syntax: MOVE Src, Dst

Description: Moves the 16-bit data word from the source address location to the destination location.

Condition Codes: No flags are affected.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	1	0	1	SM	DM	Src	Dst

NEG - Group 2 - negate

Operation: 0 - Source → Source

Assembler Syntax: NEG Src

Description: Subtracts the source operand from zero and stores the result in the source location.

Condition Codes:

C	Set if a carry is generated. Otherwize cleared.
N	Set if the most significant bit of the result (negative value) is 1. Otherwize cleared.
I	Not affected.
V	Set if an overflow occurs. Otherwise cleared.
Z	Set if the result is zero. Otherwise cleared.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	0	0	0	SM	01	Src	Src

NOT - Group 2

Operation: ~ Source → Source

Assembler Syntax: NOT Src

Description: Performs the logical NOT on the source operand and stores the result in the source location.

Condition Codes:

C	Not affected.
N	Set if the most significant bit of the result (negative value) is 1. Otherwize cleared.
I	Not affected.
V	Not affected.
Z	Set if the result is zero. Otherwise cleared.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	0	0	0	SM	00	Src	Src

OR - Group 1

Operation: Source $\cup$ Destination $\rightarrow$ Destination

Assembler Syntax: OR Src, Dst

Description: Logically OR's the source operand with the destination operand and stores the result in the destination location.

Condition Codes:

C	Not affected.
N	Set if the most significant bit of the result (negative value) is 1. Otherwize cleared.
I	Not affected.
V	Not affected.
Z	Set if the result is zero. Otherwise cleared.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
0	1	0	1	SM	DM	Src	Dst

PUSH - Group 12

Operation: Destination - 1 $\rightarrow$ Destination
Source $\rightarrow$ (Destination)

Assembler Syntax: PUSH Src, Dst

Description: Decrements the destination location value (address) and puts the the source operand into the address specificied by the destination.

Condition Codes: No flags are affected.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	1	1	0	SM	11	Src	Dst

ROL - Group 2 - rotate left

Operation: Source rotated one bit left through Carry → Source

Assembler Syntax: ROL Src

Description: Rotates the Source one bit to the left through the Carry Flag. The Carry Flag is shifted into the LSB, and the MSB is shifted out into the Carry Flag.

Condition Codes:

C	Set to the most significant bit of the source.
N	Set if the most significant bit of the result (negative value) is 1. Otherwize cleared.
I	Not affected.
V	Not affected.
Z	Set if the result is zero. Otherwise cleared.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	0	0	1	SM	01	Src	Src

ROR - Group 2 - rotate right

Operation: Source rotated one bit right through Carry → Source

Assembler Syntax: ROR Src

Description: Rotates the Source one bit to the right through the Carry Flag. The Carry Flag is shifted into the MSB, and the LSB is shifted out into the Carry Flag.

Condition Codes:

C	Set to the least significant bit of the source.
N	Set if the most significant bit of the result (negative value) is 1. Otherwize cleared.
I	Not affected.
V	Not affected.
Z	Set if the result is zero. Otherwise cleared.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	0	0	1	SM	11	Src	Src

RTI - Group 14 - return from interrupt

Operation: (SP) → PC
SP + 1 → SP
(SP) → Flags
SP + 1 → SP

Assembler Syntax: RTI

Description: Returns from a SWI (Software Interrupt) instruction call. The contents of the stack location are put into the PC; this is the address at which to continue execution. The stack is incremented and the condition flags are restored to their previous values (before the interrupt) with the contents of the stack location.

Condition Codes: All flags are affected.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	1	1	1	01	00	- - - -	- - - -

SEC - Group 13 - set carry

Operation: 1 → Carry Flag

Assembler Syntax: SEC

Description: Sets the Carry bit flag to one.

Condition Codes:

C	Set to one.
N	Not affected.
I	Not affected.
V	Not affected.
Z	Not affected.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	1	1	1	00	00	- - - -	- - - -

SEI - Group 13 - set interrupt

Operation: 1 $\rightarrow$ Interrupt Flag

Assembler Syntax: SEI

Description: Set the interrupt enable flag to one.

Condition Codes:

C	Not affected.
N	Not affected.
I	Set to one.
V	Not affected.
Z	Not affected.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	1	1	1	00	10	- - - -	- - - -

SHL - Group 2 - shift left

Operation: Source shifted one bit $\rightarrow$ Source

Assembler Syntax: SHL Src

Description: Shifts the Source one bit to the left. The most significant bit is shifted to the Carry Flag and a zero is shifted into the least significant bit.

Condition Codes:

C	Set to the most significant bit of the source.
N	Set if the most significant bit of the result (negative value) is 1. Otherwize cleared.
I	Not affected.
V	Not affected.
Z	Set if the result is zero. Otherwise cleared.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	0	0	1	SM	00	Src	Src

SHR - Group 2 - shift right

Operation: Source shifted one bit → Source

Assembler Syntax: SHR Src

Description: Shifts the Source one bit to the right. The least significant bit is shifted to the Carry Flag and the most significant bit is shifted back into itself, implementing a signed divide-by-2.

Condition Codes:

C	Set to the least significant bit of the source.
N	Set if the most significant bit of the result (negative value) is 1. Otherwize cleared.
I	Not affected.
V	Not affected.
Z	Set if the result is zero. Otherwise cleared.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	0	0	1	SM	10	Src	Src

STF - Group 10 - store flags

Operation: Flags → Store

Assembler Syntax: STF Dst

Description: Stores the Flags register into the destination location.

Condition Codes: No flags are affected.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	1	1	0	SM	01	Dst	Dst

STOP

Operation: Stop program execution.

Assembler Syntax: STOP

Description: Stops the execution of the program and returns control to the simulator.

Condition Codes: No flags are affected.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
0	1	1	1	- - - -	- - - -	- - - -	- - - -

SUB - Group 1

Operation: Destination - Source $\longrightarrow$ Destinatinon

Assembler Syntax: SUB Src, Dst

Description: Subtracts the source operand from the destination operand and stores the result in the destination location.

Condition Codes:

C	Set if a carry is generated. Otherwise cleared.
N	Set if the most significant bit of the result (negative value) is 1. Otherwise cleared.
I	Not affected.
V	Set if an overflow occurs. Otherwise cleared.
Z	Set if the result is zero. Otherwise cleared.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
0	0	1	0	SM	DM	Src	Dst

SUBC - Group 1 - subtract with carry

Operation: Destination - Source - Carry Flag $\longrightarrow$ Destinatinon

Assembler Syntax: SUBC Src, Dst

Description: Subtracts the source operand and Carry Flag bit from the destination operand and stores the result in the destination location.

Condition Codes:

C	Set if a carry is generated. Otherwise cleared.
N	Set if the most significant bit of the result (negative value) is 1. Otherwise cleared.
I	Not affected.
V	Set if an overflow occurs. Otherwise cleared.
Z	Set if the result is zero. Otherwise cleared.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
0	1	0	0	SM	DM	Src	Dst

SWI - Group 15 - software interrupt

Operation:

SP - 1 $\longrightarrow$ SP
Flags $\longrightarrow$ (SP)
SP -1 $\longrightarrow$ SP
PC $\longrightarrow$ (SP)
SWIREG $\longrightarrow$ PC

Assembler Syntax: SWI

Description: The Condition Flags and PC are saved to the stack. Then the address in the Software Interrupt Register (SWIREG) is put into the PC. Execution is transferred to the location in the PC.

Condition Codes: No flags are affected.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	1	1	1	01	01	- - - -	- - - -

TEST - Group 9

Operation: Src $\longrightarrow$ T1

Assembler Syntax: TEST Src

Description: Stores the contents of the source register into the first temporary register.

Condition Codes:

C	Not affected.
N	Set if the most significant bit of the result (negative value) is 1. Otherwise cleared.
I	Not affected.
V	Not affected.
Z	Set if the result is zero. Otherwise cleared.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
1	1	1	0	SM	00	Src	Src

XOR - Group 1

Operation: Source + Destination $\longrightarrow$ Destination

Assembler Syntax: XOR Src, Dst

Description: Logically Exclusive OR's the source operand with the destination operand and stores the result in the destination location.

Condition Codes:

C	Not affected.
N	Set if the most significant bit of the result (negative value) is 1. Otherwize cleared.
I	Not affected.
V	Not affected.
Z	Set if the result is zero. Otherwise cleared.

Instruction Format:

15	14	13	12	11-10	9-8	7-4	3-0
0	1	1	0	SM	DM	Src	Dst