

The NEBULA Computer

(A General Manual)

Part 1:

Hardware Systems

Department of Computer Science

Oregon State University  
Corvallis, Oregon 97331

August 1976

## TABLE OF CONTENTS

GENERAL INFORMATION	1
Memory	1-1
Map	2-2
Peripheral Units	1-3
PROCESSOR	2
Registers	2-1
Processor States	2-4
Data Formats	2-6
Instruction Formats	2-7
Effective Address Calculation	2-8
Shift Instructions	2-11
I/O Instructions	2-12
Partial & Double-Word Addressing	2-14
INPUT/OUTPUT	3
Single Character I/O	3-1
Interrupt System	3-3
INSTRUCTIONS	4
Explanation of Notations	4-1
Register Specifying	4-2
Floating Point	4-6
Miscellaneous Arithmetic	4-10
Shifts	4-12
Partial & Multi-Word	4-16
Conditional Jumps & Skips (except I/O)	4-18
Machine Status	4-20
I/O Instructions	4-23
CONSOLE	5
Power	5-1
Bootstrap	5-3
F-States	5-3
Control Switches	5-4
LIST OF INSTRUCTIONS	6-1
I/O Instructions	6-3
Mnemonic Code Table	6-4
COMPUTER CENTER PUBLICATIONS	7-1

The NEBULA is a general purpose, serial computer, designed, constructed, and maintained by OSU students. NEBULA had its beginnings in 1963 in a graduate seminar whose purpose was to investigate the possibility of building a low-cost digital computer to serve as a modern complement to the ALWAC III-E. Some members of the seminar carried on the design and construction, which in their later stages were funded by grants from the Office of Naval Research.

The goals of the research project were to study the use of glass delay lines as a main computer memory, and to design an inexpensive machine with some of the characteristic capabilities of larger machines but which could be easily modified. These goals have for the most part been realized in the machine as it now stands.

#### The Memory

NEBULA was originally constructed using 100uS glass delay line memory. There were two types of memory constructed, a 4096 word random-access memory for general use and a 2048 word experimental associative, or "content-addressable" memory (CAM).

These memory systems have been supplanted with a 32K x 36 core memory in the form of an IBM 7302 core storage unit. The memory was originally part of a 7030 "STRETCH" computer, and is sometimes referred to as the Stretch Memory.

The 7302 has a cycle time of 2.18uS, with a read access of approximately 1uS. The memory controller has eight memory

ports, of which the NEBULA processor uses one. The memory cycles are allocated on a priority basis, with the next cycle allotted to the highest priority request pending.

Although 36 data bits are available from the memory NEBULA uses only 34. The bits of a NEBULA word are numbered right to left, 0--33. Each word consists of the spare bit (bit 0), a 32-bit data word (bits 1 through 32), and the parity bit (bit 33).

The parity bit actually has nothing to do with parity except that original plans to put parity on the memory were never completed. The parity bit remains useful as a tag bit, like the spare bit.

## Peripheral Units

NEBULA has a wide range of peripheral hardware for input and output.

Teletypes:           one model 33 ASR, one model 33 KSR  
                      both with full duplex interfaces.

Line Printer:       Potter model 3502  
                      350 lines per minute, 132 columns

Paper Tape:         Digitronics Model B5000 tape reader  
                      300 characters per second, 5-7-8 channels.

Tally model P120 paper tape punch  
120 characters per second, 5-7-8 channels

Drum:               Vermont Research Corp. fixed head drum  
                      32768 words (256, 128-word sectors)  
                      36 mS per sector transfer time

Floppy Disk:        Memorex model 651 flexible disk 65536 words  
                      64 tracks, 8 sectors/track, 128 words/sector  
                      maximum access time (land head, seek to track,  
                      latency to sector) approx. 650 mS.

## The Processor

The NEBULA processor contains eight flip-flop registers and eight memory locations which are used as index registers. Three of the registers act as general arithmetic registers except for certain operations; the others have specialized functions. All hardware registers are treated as part of memory, except the instruction register.

<u>Memory Address</u>	<u>Register</u>		<u>Description</u>
00	Z	(34 bits)	Zeros Register
01	U	(34 bits)	Universal Accumulator
02	X	(34 bits)	Extension of U Register
03			Unused
04	C	(32 bits)	
		bits 17-32	Program Counter
		bits 1-16	Flag Register
05	E	(8 bits)	Input/Output Buffer
06	V	(34 bits)	Operand Register
07	W	(34 bits)	Ones Register
	IR	(32 bits)	Instruction Register
		0 (bits 1-8)	Operation Code
		M (bits 9-16)	Modifier Code
		D (bits 17-32)	Address Field

The lower half of the C register forms a 16-bit flag register. Some of these bits are used to indicate hardware status conditions, such as overflow, etc. The remaining bits are available for

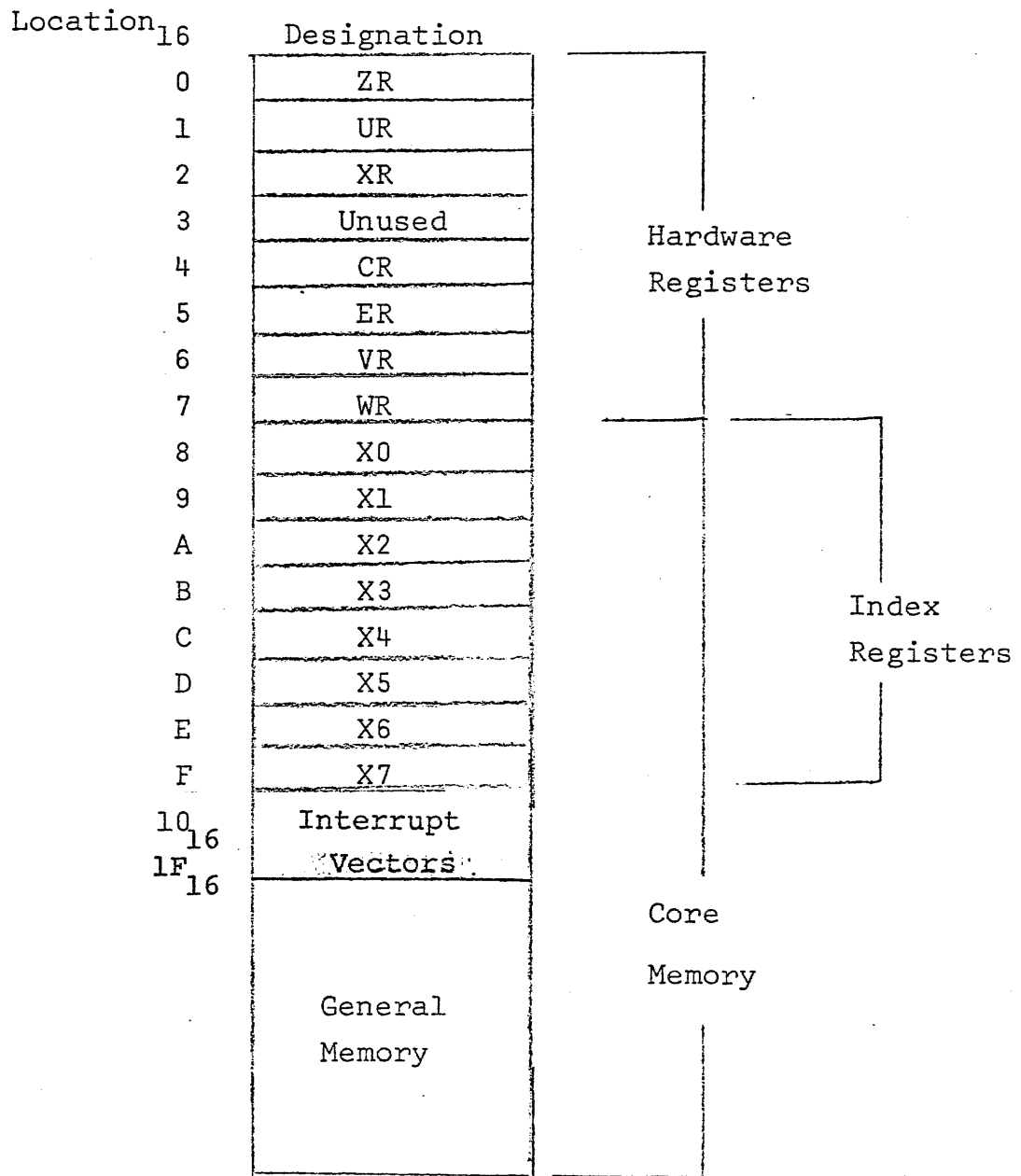


Figure 2-0

Memory Map

general programming use. There are instructions to store, set, reset and restore these flags for a variety of circumstances. The table following gives the hardware bit assignments.

#### C Register

Bit	Mask	Meaning
1	0001	COMPARE FLAGS (SET BY CMr INSTRUCTIONS)
2	0002	
3	0004	Arithmetic Overflow
4	0008	Drum Error (read parity or write protec
5	0010	----
6	0020	----
7	0040	Arithmetic shift (lost precision)
8	0080	Set by TSB
9	0100	----
10	0200	----
11	0400	----
12	0800	----
13	1000	----
14	2000	----
15	4000	----
16	8000	----

#### The Mythical Y register

This register ought to occupy location 3 in the registers. The hardware register was never built, and references to location 3 are equivalent to referencing the Z register. Opcodes that normally would deal with the Y register are undefined, but



reserved. The associated Mnemonics are also reserved, and appear on some instruction charts.

### Processor States

Instruction execution is done in a variable number of phases, called F-states. The F-states are numbered  $F_0 - F_9$ , and each take exactly one word time. The actions for each F-state are given below:

$F_0$	instruction fetch
$F_1$	index modification
$F_2$	indirect addressing
$F_3$	most simple instructions complete execution
$F_4$	double length operand instructions complete execution
$F_5$	} execution phases for more complex operations (FLOATING MULTIPLY, etc)
$F_6$	
$F_7$	
$F_8$	
$F_9$	

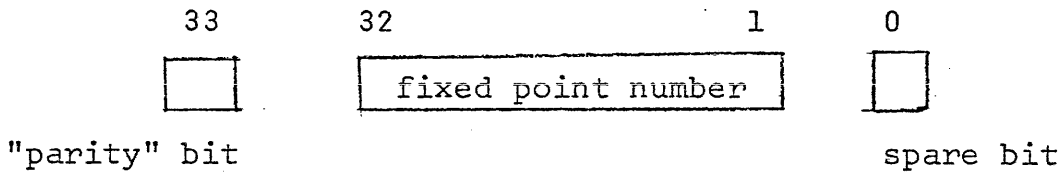
Not all F-states need be entered, nor do they have to be executed in order. An instruction with no indirect addressing or indexing will go directly from  $F_0$  to  $F_3$ , for example, while a multi-level indirectioned and indexed instruction might oscillate between  $F_1$  and  $F_2$  several times. (For explanation of indirect addressing and indexing see Effective Address Calculation, p. 2-8,9)



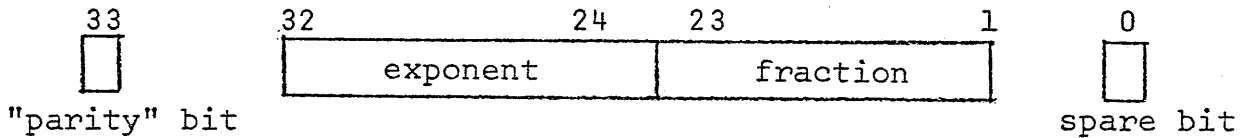
## Data Formats

A positive fixed-point number is represented in the NEBULA as a 32-bit word. The negative fixed point numbers are represented in 2's-complement form (See Fig. 2-1).

Floating-point numbers are represented in one word. Bits 1-23 are used for the fraction while bits 24-31 form the exponent. The exponent is represented in excess 128 ( $128 + \text{exponent}$ ). The negative for a floating-point number is represented as the 2's complement of its positive form computed as if it were a fixed-point integer. Fixed-point and floating-point zero are the same.



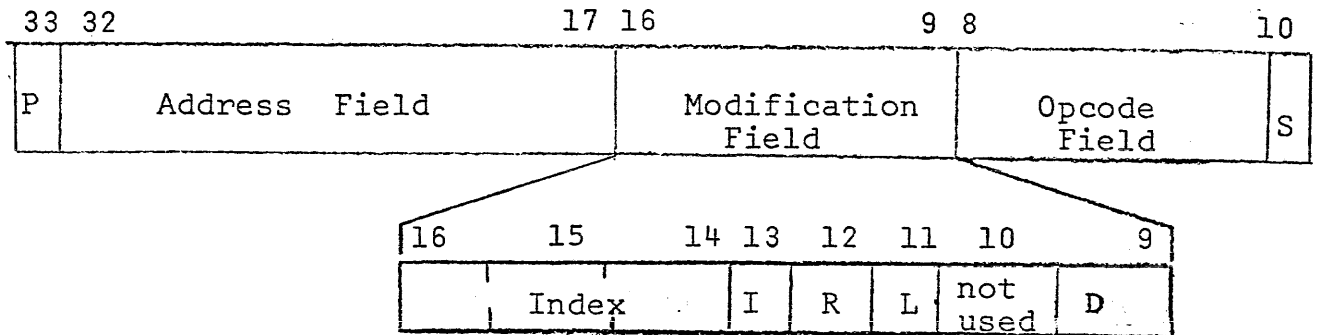
### a. Fixed Point Format



### b. Floating Point Format

Figure 2-1 NEBULA Data Formats

## Instruction Formats



Most NEBULA instructions have the format shown above, which sacrifices compactness for simplicity, generality, and versatility. Note that full modification is available for all instructions. The parity and spare bits are ignored.

The register-specifying instructions can select any one of the eight hardware registers or, in one case, one of eight index registers. These instructions are distinguished by having bit five of their operation code a one. Bits six through eight select the register the operation is to be performed on. Bits one through four specify the operation.

The register non-specifying commands either do not operate with the registers or are assigned to operate on one specific register or set of registers.

## Effective Address Calculation

NEBULA'S modifier field can specify direct, single, or multi-level indirect addressing, as well as index modification. In addition, either the contents of the final address calculated (the "effective address"), or that address itself can be specified as the operand for most instructions.

### Indexing:

Three modifier field bits (14-16) select one of seven index registers in memory locations 09-0F<sub>16</sub>. If the field is zero, no indexing takes place. The contents of the upper half of the index word are added to the instruction address.

### Indirect and Deferred:

Two bits of the modifier field (the "I" and "D" bits) determine if indirect addressing takes place. There are three types of indirect addressing:

- 1) Normal, multilevel indirect - The instruction address plus the index (if any) is used to fetch a new address, index, and indirect bit. The new index, if any, is added to the new address, and the indirect bit is checked. The process repeats until the indirect bit comes up zero.  
(I=1, D=0).
- 2) Deferred left - Similar to multi-level indirect, but only one level of indirect is performed. The effective address is the left half of the word addressed, regardless of the indirect bit or index field in that word. (I=1, D=1).
- 3) Deferred right - the same as deferred left, except the right half of the word addressed becomes the effective address.  
(I=0, D=1).

EFFECTIVE  
ADDRESS  
CALCULATION

F<sub>0</sub>: Fetch instruction  
Op-code    0 register  
Modifier   M register  
Address    D register

Is index  
field zero?

F<sub>1</sub>: Indexing  
Add bits 17-32 of  
specified index to  
D register

Defer or  
indirect bit  
set?

F<sub>2</sub>: Indirect  
Perform indirect  
OR defer as per  
I and D bits:  
I=1, D=0: Fetch new  
address field, index  
field and I bit from  
word addressed by D  
register  
I=1, D=1: fetch new  
address field from  
left half of word  
addressed by D register  
I=0, D=1: fetch new  
address field from  
right half of word  
addressed by D register

FINISH F<sub>3</sub>: first execution state  
FINISH F<sub>4</sub>: second execution state  
FINISH F<sub>q</sub>: last execution state

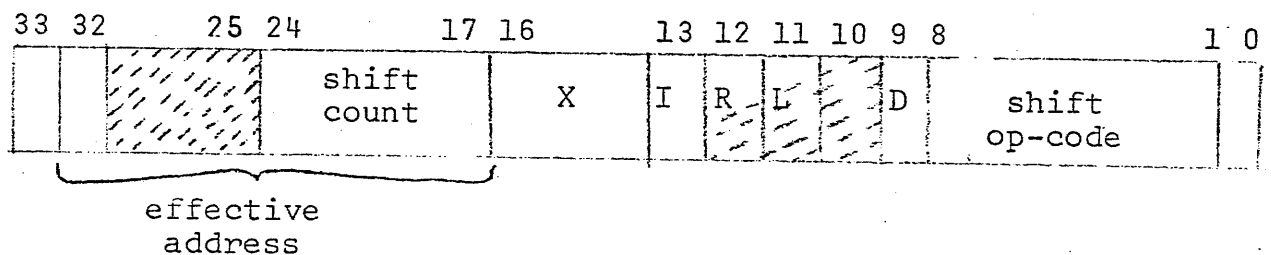
### Immediate Modes:

The left and right immediate bits of the modifier determine the final operand for most instructions. If both the immediate bits are zero, the operand is the contents of the effective address. If the right immediate bit is one, the operand is the effective address, 16-bits, right-justified. If the left immediate bit is a one, the effective address is the left 16-bits of the operand, with the right half of the word taken as zero. If both immediate bits are ones, the effective address forms both the left and right halves of the operand. With any immediate operand, the spare and parity bits are zero.

## Shift Instruction Format

There are three types of shift instructions used on NEBULA: the register specifying commands for a logical shift and for a modified shift and the register non-specifying commands for double length shifts. The double length logical shift instructions and the register-specifying logical shift instructions have the same format. The modified shift instruction format differs only in the significance of bit 32 of the instruction word. For the modified shift, bit 32 indicates whether the shift is a circular shift (bit 32 equals 1) or an arithmetic shift (bit 32 equals zero). For the logical shifts, bit 32 indicates the direction of shift: bit 32 equals 1 for a left shift and 0 for a right shift.

Bits 17 through 24 of the effective address contain the number of places to be shifted. Immediate mode has no effect.





## Input-Output Instruction Format

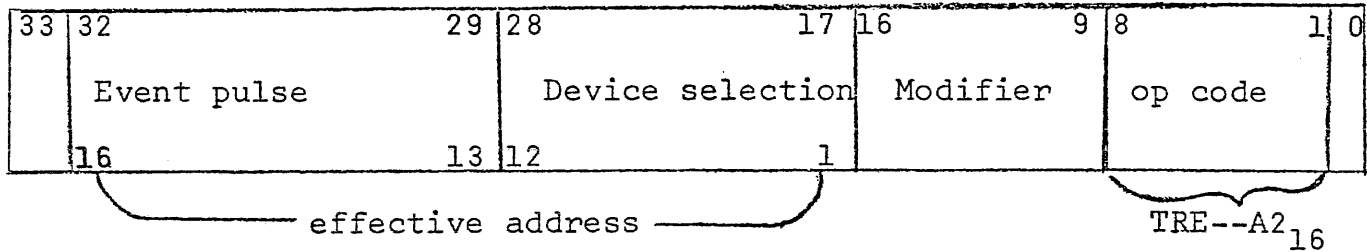


Figure 2-2 I/O Instruction Format

Most of the input-output commands have the same operation code, TRE ("Transfer Register") or A2<sub>16</sub>. Bits 1 through 12 of the effective address each select one of 12 different input-output devices. Bits 13 through 16 control the event pulses for the transfer of information or the control of the input-output device selected. The event pulses occur sequentially: EV<sub>1</sub> occurs at bit time i-1.

High speed devices, such as the disk, transfer directly into memory and operate simultaneously with the processor. There are five instructions that drive these devices: LIAR (AA), RIOTS (AC), IDIOT (AB), SENSE(AD), and SENSN (AE).

LIAR is used to select a device and transmit operation instructions. This includes sector and track selection, etc., and interrupt selection by condition. IDIOT transfers an address and count to the last device selected with a LIAR instruction, and initiates the transfer. Both LIAR and IDIOT skip the next instruction if the information was accepted by the device. If the device is busy, or if no device is selected or exists, the information is Rejected and the instruction does not skip.

Current device status may be obtained with the RIOTS, SENSE and SENSN instructions. RIOTS reads the current status from the last selected device and stores it into the effective address. SENSE compares the operand with the status and skips if their AND is non-zero. SENSN is similar to SENSE, but skips if the AND is zero.

## Partial and Double Word Addressing

NEBULA also has the ability to address words as 8-bit bytes (character addressing), 16-bit bytes (half-word addressing) and as 64-bit bytes (double-word addressing) for loading and storing.

A character address is in the form  $4 * \text{word address} + \text{character position}$ . The character positions are numbered 0-3 from right to left as shown in figure 2- . .

Half-word addressing follows a similar pattern. A half-word address is in the form  $2 * \text{word address} + \text{half-word position}$ . The half-word positions are numbered 0 and 1 from right as shown in figure 2-3.

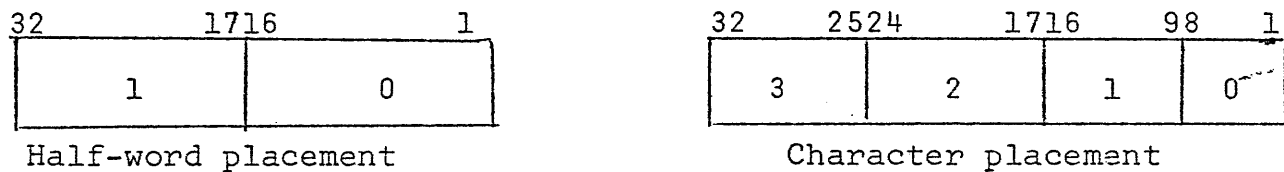


Figure 2-3 Half-word and Character Placement

Character addressing is performed only with the E register, half-word addressing is performed only with the U register, and double-word addressing is performed only with the U and X registers.



## INPUT/OUTPUT

### Single Character I/O

All single-character I/O on NEBULA is done with the TRE (transfer E Register) instructions. The lower 12 bits of the effective address select the device, while the top 4 bits select 4 possible "event times" to be sent to the device interface. All I/O is done through the E register.

### Teletypes

Instructions for the two teletypes are the same with the exception of device selection. Characters may be printed on both units at the same time, but characters received from the keyboards must be received separately.

### Keyboard:

TTY 1      device 5 (010)  
TTY 2      device 1 (001)

### Event times:

EV 1	<u>Skip on flag</u>	SKKF2	100100A2
		SKKF	101000A2
EV 2	Clear E register		
EV 3	Transfer character to E register		
EV 4	<u>Clear Keyboard flag</u>	CKF2	800100A2
		CKF	801000A2
EV's 2,3, and 4 may be combined			
	<u>Teletype In</u>	TTI2	E00100A2
		TTI	E01000A2

### Teletype Printer:

TTY 1      device 6 (002)  
TTY 2      device 2 (020)

### Event times:

EV 1	<u>Skip on flag</u>	SPRF2	100200A2
		SPRF	102000A2

EV 2	<u>Clear flag</u>	CPRF2	2G0200A2
		CPRF	202000A2
EV 3	Not used		
EV 4	Send character to TTY		
EV's 2,3, and 4 may be combined			
	<u>Teletype Out</u>	TT02	E00200A2
		TTO	E02000A2

Paper Tape Reader                      Device 3 (004)

Event times:

EV 1	<u>Skip on Flag</u>	SKRF	100400A2
EV 2	<u>Clear E register and clear flag</u>		
EV 3	Or C(E) with tape reader buffer then initiate forward tape motion		
EV 4	Or C(E) with tape reader buffer then initiate backward tape motion		

EV's 2 and 3, or 2 and 4 may be combined

	<u>Read Forward</u>	RDF	600400A2
	<u>Read Backward</u>	RDB	A00400A2

Paper Tape Punch                      Device 4 (008)

Event times:

EV 1	<u>Skip on Flag</u>	SKPF	100800A2
EV 2	<u>Clear Flag</u>	CPF	200800A2
EV 3	Not Used		
EV 4	Punch character		

EV's 2, 3, and 4 may be combined

	<u>Punch Character</u>	PCH	E00800A2
--	------------------------	-----	----------

Line Printer                              Device 9 (100)

Event times:

EV 1	<u>Skip if Buffer Ready</u>	SKBR	110000A2
EV 2	<u>Transfer C(E) to printer buffer</u>	LPLB	210000A2
EV 3	<u>Initiate Print</u>	LPPR	410000A2
EV 4	<u>Clear Printer Buffer</u>	LPCL	810000A2

Any event times may be combined.

NOTES:

DC power to line printer turned off results in a constant ready signal.

Line printer flag = buffer ready AND paper not moving.  
The line printer buffers 128 character lines. LPLB transfers the character to the next sequential position in the buffer. Buffer overflow automatically initiates a print cycle, buffer will become un-ready until cycle has completed.

Clearing the printer buffer (LPCL) automatically sets top-of-form condition. Line printer will go to top-of-form at the completion of the next print cycle.

The printer buffer should not be cleared until the completion of the print cycle.

Interrupt System

The processor recognizes interrupts on 16 priority levels. When an interrupt is recognized, the processor completes the execution of the current instruction, then stores the contents of the C-register in the interrupt vector corresponding to the interrupt level and at the same time loading the C-register with the former contents of that vector. This effects saving the C-register and transferring control to the interrupt handling routine. Interrupt vectors reside in memory locations  $10_{16}$  through  $IF_{16}$ , with location  $10_{16}$  corresponding to level zero.

Individual interrupt levels are controlled by the 16-bit Interrupt Mask Register (IMR). An interrupt may occur on a given level only if the bit for that level in the IMR is a one. The IMR may be set and reset by the SIM and RIM instructions.

Overall control of the interrupts is held by the Interrupt Enable Toggle (ENINT). This flip-flop must be set to a one before any interrupt can occur. ENINT is reset as soon as an interrupt occurs, and should be set to one just before exiting an interrupt routine. The

execution of a SIT instruction is delayed one instruction time to allow time to exit the service routine.

Interrupt service routines should always properly restore the vector location before exiting. This is most easily done by executing an EXC in the location just before the service routine entry or re-entry point.

#### Interrupt Vector Locations

Level	Location <sub>16</sub>	IMR Mask	Device
0	10	0001	TTY 2 Keyboard
1	11	0002	TTY 2 Printer
2	12	0004	Paper Tape Reader
3	13	0010	Paper Tape Punch
4	14	0010	TTY 1 Keyboard
5	15	0020	TTY 1 Printer
6	16	0040	unassigned
7	17	0080	unassigned
8	18	0100	Line Printer
9	19	0200	unassigned
10	1A	0400	unassigned
11	1B	0800	unassigned
12	1C	1000	Manual Interrupt
13	1D	2000	Internal Interrupt
14	1E	4000	Real Time Clock
15	1F	8000	Executive Interrupt



## THE NEBULA INSTRUCTION SET

Each instruction is described in two ways: in pseudo algorithmic language and by a brief paragraph. The execution time in word times is given in parenthesis to the left of the instruction mnemonic. Instruction fetch, indexing, and each level of indirect addressing each take an additional word time.

The table below gives the symbols used in the pseudo-algorithmic description of the instructions.

Symbol	Description
A	effective address
$A_i$	value of bit i of effective address
C( )	contents of
C( ) i-j	contents of bits i to j of
M	memory word
R	specified register
	absolute value
∧	and
∨	or
⊕	exclusive or
=	equal
≠	not equal
:=	set equal to
/	divide by
*	multiplied by
+	plus
-	minus
>	greater than
<	less than
$\overline{C( )}$	one's complement of the contents of
I	specified index register
∃	there exists
∀	for all
:::	swap

## Register-Specifying Instructions

<u>Register Zero Jumps</u>	(0)	UZJ,EQ	030
		,NE	430
The left and right immediate bits		,GE	830
select the condition to test for		,LT	C30
jumping. The conditions selectable		XZJ,EQ	050
are: equal (EQ), not equal(NE),		,NE	450
greater or equal (GE), and less than		,GE	850
(LT). If the register, compared		,LT	C50
with zero, meets the condition		EZJ,EQ	0B0
specified by the instruction, control		,NE	4B0
is transferred to the effective		,GE	8B0
address. Otherwise this instruction		,LT	CB0
acts as a NOP. In either case the		VZJ,EQ	0D0
execution time is zero.		,NE	4D0
		,GE	8D0
		,LT	CD0
<u>Load Register</u>	(1)	LDU	31
		LDX	51
C(R):=C(A)		LDC	91
Contents of the effective address are		LDE	B1
loaded into the register specified.		LDV	D1
<u>Store Register</u>	(1)	STU	32
		STX	52
C(A):=C(R)		STC	92
Contents of the specified register are		STE	B2
stored into the effective address.		STV	D2

<u>Add to Register</u>	(1)	ADU	33
$C(R) := C(R) + C(A)$		ADX	53
		ADC	93
		ADE	B3
		ADV	D3

The contents of the effective address are added to the contents of the specified register. Overflow sets  $C_3$ .

<u>Subtract from Register</u>		<u>Menmonic Code</u>	<u>Hex Code</u>
$C(R) := C(R) - C(A)$	(1)	SBU	34
		SBX	54
		SBC	94
		SBE	B4
		SBV	D4

Contents of the effective address are subtracted from the contents of the specified register. Overflow will set  $C_3$ .

<u>Or to Register</u>			
$C(R) := C(R) \vee C(A)$	(1)	ORU	35
		ORX	55
		ORE	B5
		ORV	D5

The logical sum of the contents of the effective address and the contents of the specified register is placed in the specified register.

<u>Exclusive Or to Register</u>			
$C(R) := C(R) \oplus C(A)$	(1)	EOU	36
		EOX	56
		EOC	96
		EOE	B6
		EOV	D6

The modulo-two sum of the contents of the effective address and of the specified register is placed in the specified register.

	Mnemonic Code	Hex Code
<u>And to Register</u>		
$C(R) := C(R) \wedge C(A)$	(1) ANU	37
The logical product of the contents of the effective address and the contents of the specified register is placed in the specified register.	ANX	57
	ANE	B7
	ANV	D7
<u>Compare Register</u>		
if $C(R) > C(A)$ then $C(C)_2 := 0 \wedge C(C)_1 := 1$	(1) CMZ	18
if $C(R) = C(A)$ then $C(C)_2 := 0 \wedge C(C)_1 := 0$	CMU	38
if $C(R) < C(A)$ then $C(C)_2 := 1 \wedge C(C)_1 := 0$	CMX	58
Contents of the effective address are compared with the contents of the specified register. $C_1$ and $C_2$ indicate the result.	CME	B8
	CMV	D8
	CMW	F8
<u>Test Bit in Register</u>		
if $C(R)_A = 1$ then $C(C)_{17-32} := C(C)_{17-32} + 1$	(1) TSU	3B
If the bit in the specified register designated by the effective address is 1, the control counter is incremented by 1 causing a skip. The effective address is used mode 64. Testing "bits" 34-63 results in no skip (always zero).	TSX	5B
	TSC	9B
	TSE	BB
	TSV	DB
<u>Exchange Register with Memory</u>		
$C(R) := C(A)$	(1) EXU	3C
The contents of the specified register are exchanged with the contents of the effective address.	EXX	5C
	EXC	9C
	EXE	BC
	EXV	DC

Absolute Value Register

Mnemonic Code	Hex Code
------------------	-------------

C(R):= C(R)

(1)	AVU	3D
	AVX	5D
	AVV	DD

Contents of the specified register are replaced by the absolute value of the contents of the specified register. Immediate mode has no effect. The effective address is ignored. Overflow will set C<sub>3</sub>.

Negate Register

C(R):=-C(R)

(1)	NGU	3E
	NGX	5E
	NGE	BE
	NGV	DE

Contents of the specified register are replaced by the two's complement of the specified register. Immediate mode has no effect. The effective address is ignored. Overflow will set C<sub>3</sub>.

## FLOATING POINT

### Floating Point Add

<u>Mnemonic</u>	<u>Hex</u>
<u>Code</u>	<u>Code</u>

$C(U) := C(U) + C(A)$	(5)	FAD	43
-----------------------	-----	-----	----

if exponent  $> 2^{127}$  then  $C(C)_3 := 1$

else if exponent  $< 2^{-128}$  then  $C(C)_3 := 1$

The contents of the effective address are added to the contents of the U register. The contents of the E and V registers are destroyed. The left immediate mode treats the effective address as an operand appearing in the left half of a 32-bit word with the right half set to zero. Exponent underflow or overflow sets  $C_3$ . The E register is clobbered.

### Floating Point Subtract

$C(U) := C(U) - C(A)$	(5)	FSB	44
-----------------------	-----	-----	----

if exponent  $> 2^{127}$  then  $C(C)_3 := 1$

else if exponent  $< 2^{-128}$  then  $C(C)_3 := 1$

Contents of the effective address are subtracted from the contents of the U register. The contents of the E and V registers are destroyed. The left immediate mode treats the effective address as an operand appearing in the left half of a 32-bit word with the right half set to zero. Exponent overflow or underflow sets  $C_3$ . The E register is clobbered.

### Floating Point Multiply

<u>Mnemonic Code</u>	<u>Hex Code</u>
----------------------	-----------------

$C(U) := C(U) * C(A)$   
if exponent  $> 2^{127}$  then  $C(C)_3 := 1$   
else if exponent  $< 2^{-128}$  then  $C(C)_3 := 1$

(37)	FMP	45
------	-----	----

Contents of the U register are multiplied by the contents of the effective address. The contents of the E, X, and V registers are destroyed. The left immediate mode treats the effective address as an operand appearing in the left half of a 32-bit word with the right half set to zero. Exponent overflow or underflow sets  $C_3$ . The E register is clobbered.

### Floating Point Divide

if divisor = 0 then  $C(C)_3 := 1$   
 $C(U) := C(U) / C(A)$   
if exponent  $> 2^{127}$  then  $C(C)_3 := 1$   
else if exponent  $< 2^{-128}$  then  $C(C)_3 := 1$

(38-40)	FDV	46
---------	-----	----

Contents of the U register are divided by the contents of the effective address. The contents of the E, X, and V registers are destroyed. The left immediate mode treats the effective address as an operand appearing in the left half of a 32-bit word with the right half set to zero. Exponent overflow or underflow sets  $C_3$ . Also an attempt to divide by 0 sets  $C_3$ . The E register is clobbered.

<u>Pack</u>	Mnemonic Code	Hex Code
For k=1 step 1 until A $C(U)_{32} := C(U)_{32}$ $C(U)_i := C(U)_{i+1} / i=1, \dots, 31$ if $C(X)_{32} = 0$ then $C(X)_{31} := C(U)_1$ else $C(X)_{32} := \overline{C(U)_1}$ $C(X)_i := C(X)_{i+1} / i=1, \dots, 29$	(2-3) PCK	40----80

This command packs an exponent from the U register into the X register between the sign bit  $X_{32}$  and the rest of the fraction. If the quantity in X is negative, the one's complement of U is packed into X. The effective address A determines the number of bits in U that are used as an exponent. Immediate mode has no effect.

<u>Unpack</u>	Mnemonic Code	Hex Code
For k=1 step 1 until A $C(X)_1 := 0$ $C(X)_{i+1} := C(X)_i / i=1, \dots, 30$ if $X_{32} = 0$ then $C(U)_1 := C(X)_{31}$ else $C(U)_1 := \overline{C(X)_{31}}$ $C(U)_{i+1} := C(U)_i / i=1, \dots, 31$	(2-3) UPK	CO----80

This command unpacks an exponent from the X register into the U register. If the quantity in the X register is negative, the one's complement of the exponent is unpacked and put in the U register.



## Unpack (continued)

The effective address A determines the number of bits in the U register that are used as an exponent. The immediate mode has no effect.

## Normalize

if  $C(U)C(X)_{1-31} = 0$  then no operation (5) NRM 83

$C(X)_{32} := 0$

$C(U)C(X)_{1-31} := C(U)C(X) * 2^t$

(where t is such that

$2^{62} < C(U)C(X)_{1-31} < 2^{63}$ )

$C(V) := -t$

The contents of the U register and bits 1-31 of the X register are shifted left (or right for the special case of -1) until bits 32 and 31 of the U register are not equal.  $X_{32}$  does not take part in the shift. The two's complement of the number of shifts is placed in the V register. The E register is clobbered. Immediate mode has no effect.

MISCELLANEOUS ARITHMETICFixed Point Multiply

$C(U)$  and  $C(X)_{1-31} := C(U) * C(A)$  (33)

$C(V) := C(A); C(X)_{32} := 0$

if  $C(U) * C(A) = 1$  then  $C(C)_3 := 1$

Contents of U register and bits 1 to 31 of the X register are set equal to the product of the contents of the U register with the contents of the effective address. Bit 32 of the X register is set to 0. Overflow sets  $C_3$ .

Mnemonic Code	Hex Code
---------------	----------

MUL	25
-----	----

Fixed Point Divide

if divisor=0 then  $C(C)_3 := 1$  (34-37) DIV 26

$C(X) := \left[ C(U) C(X)_{1-31} \right] / C(A)$

$C(U) := \text{Remainder}; C(V) := C(A)$

if overflow then  $C(C)_3 := 1$

Contents of the U register together with bits 1 to 31 of the X register are divided by the contents of the effective address. Overflow turns on  $C_3$ . Note: Sign bit(s) must be in U register.

Increment and Skip on Zero

$C(A)_{17-32} := C(A)_{17-32} + 1$  (2) ISZ 21

if  $C(A)_{17-32} = 0$  then  $C(C)_{17-32} := C(C)_{17-32} + 1$

The contents of the address field of the effective address are incremented by 1. If the contents of the effective address is now zero, the next instruction is skipped.

Decrement and Skip on Zero

Mnemonic Code	Hex Code
---------------	----------

$C(A)_{17-32} := C(A)_{17-32} - 1$	(2) DSZ	22
if $C(A)_{17-32} = 0$ then $C(C)_{17-32} := C(C)_{17-32} + 1$		

The contents of the address field of the effective address are decremented by 1. If the contents of the effective address is now zero, the next instruction is skipped.

Increment Index Register and Skip

if $C(M)_4 = 0$ then $C(I)_{17-32} := C(I)_{17-32} + A$	(2) IX0	1F
else $C(I)_{17-32} := A$	IX1	3F
	IX2	5F
	IX3	7F
if $C(I)_{17-32} = 0$ then $C(C)_{17-32} := C(C)_{17-32} + 1$	IX4	9F
	IX5	BF
	IX6	DF
	IX7	FF

In the non-immediate mode the effective address is added to bits 17 through 32 of the specified index register.

In the left immediate mode the effective address is loaded into bits 17-32 of the specified index register. In either case if the result in C(I) is zero, the next instruction is skipped.

Load Index Register

$C(I)_{17-32} := A$	(2) LX0	81F
if $C(I)_{17-32} = 0$ , then	LX1	83F
	LX2	85F
$C(C)_{17-32} := C(C)_{17-32} + 1$	LX3	87F
	LX4	89F
	LX5	8BF
	LX6	8DF
	LX7	8FF

The effective address is loaded into bits 17 through 32 of the specified index register. If this causes the contents of the specified index register to be zero, the next instruction is skipped.

## SHIFT INSTRUCTIONS

### Shift Right

For  $i=1$  step 1 until  $A \bmod 2^8$  (2)  
 $C(R)_k := C(R)_{k+1} / k=1, \dots, 31$   
 $C(R)_{32} := 0$

<u>Mnemonic Code</u>	<u>Hex Code</u>
SRU	00----39
SRX	00----59
SRE	00----B9
SRV	00----D9

Contents of the specified register are shifted right the number of places specified by the effective address (modulo  $2^8$ ). Zeros are shifted in from the left. Immediate mode has no effect.

### Shift Left

For  $i=1$  step 1 until  $A \bmod 2^8$  (2)  
 $C(R)_k := C(R)_{k-1} / k=2, \dots, 32$   
 $C(R)_1 := 0$

SLU	80----39
SLX	80----59
SLV	80----D9

Contents of specified register are shifted left the number of places specified by the effective address (modulo  $2^8$ ). Zeros are shifted in from the right. Immediate mode has no effect.

### Shift Arithmetic

$C(R) := 2^{-A \bmod 2^8} * C(R)$  if  $C(R)_1 - (A \bmod 2^8) \neq 0$  (2)  
 $C(R)_{32} = 1$  then  $C(C)_7 := 1$

<u>Mnemonic Code</u>	<u>Hex Code</u>
SAU	00----3A
SAX	00----5A
SAE	00----BA
SAV	00----DA

Contents of the specified register are shifted right the number of bits specified by the effective address (modulo  $2^8$ ). If one bits are shifted off the right end and the contents of the specified register are less than zero,  $C_7$  is set to one. Bit 31 copies bit 32 spreading the sign. Immediate mode has no effect.

### Shift Circularly

For  $i=1$  step 1 until  $A \bmod 2^8$  (2)  
 $C(R)_K := C(R)_{K+1}$  /for  $K=1, \dots, 31$   
 $C(R)_{32} := C(R)_1$

SCU	80----3A
SCX	80----5A
SCE	80----BA
SCV	80----DA

Contents of the specified register are rotated right the number of places specified by the effective address (modulo  $2^8$ ).

Shift Right Double

Mnemonic Code	Hex Code
------------------	-------------

For k=1 step 1 until A mod  $2^8$

(2-3)

SRD

00----81

$C(U)_{32} := 0$

$C(U)_i := C(U)_{i+1} / i=1, \dots, 31$

$C(X)_{32} := C(U)_1$

$C(X)_i := C(X)_{i+1} / i=1, \dots, 31$

The contents of the U register and the contents of the X register are shifted right, as a double length number, the number of places specified by the effective address (modulo  $2^8$ ). Zeros are shifted in from the left. Immediate mode has no effect.

Shift Left Double

For k=1 step 1 until A mod  $2^8$

(2-3)

SLD

80----81

$C(U)_{i+1} := C(U)_i / i=1, \dots, 31$

$C(U)_1 := C(X)_{32}$

$C(X)_{i+1} := C(X)_i / i=1, \dots, 31$

$C(X)_1 := 0$

Contents of U register and contents of X register are shifted left, as a double length number, the number of places specified by the effective address (modulo  $2^8$ ). Zeros are shifted in from the right. Immediate mode has no effect.

Shift Right Arithmetic

$$C(U)C(X)_{1-31} := C(U)C(X)_{1-31} * 2^{-A} \text{ mod } 2^8$$

if  $C(X)_1 = 1 \wedge C(U) < 0$  during

shift then  $C(C)_7 := 1$

$$C(X)_{32} := 0$$

	<u>Mnemonic Code</u>	<u>Hex Code</u>
(2-3)	SRA	00----80

Contents of U register and contents of bits 1 to 31 of the X register are shifted right, as a double length number, the number of places specified by the effective address (modulo  $2^8$ ). Bit 32 of the X register is set to zero. If the sign of the U register is negative and one bits are lost,  $C_7$  is set to one. Immediate mode has no effect.

Shift Left Arithmetic

$$C(U)C(X)_{1-31} := C(U)C(X)_{1-31} * 2^A \text{ mod } 2^8$$

if  $C(U)_{32} \neq C(U)_{31}$  during shift

then  $C(C)_3 := 1$

$$C(X)_{32} := 0$$

(2-3)	SLA	80----80
-------	-----	----------

Contents of the U register and contents of bits 1 to 31 of the X register are shifted left, as a double length number, the number of places specified by the effective address (modulo  $2^8$ ). Bit 32 of the X register is set to zero. If the sign of the number changes during the shift,  $C_3$  is set to one. The immediate mode has no effect.

PARTIAL AND MULTI-WORD INSTRUCTIONS

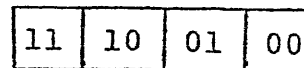
Load Character to E Register

$C(E) := C(A)$  .... memory addressed as eight bit words

Character specified by the effective address is loaded into the E register. The immediate mode has no effect.

Mnemonic Code	Hex Code
---------------	----------

(2) LCH C0



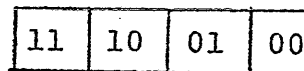
Character Placement

Store Character from E Register

$C(A) := C(E)$  .... memory addressed as eight bit words

Contents of the E register are stored in the character specified by the effective address. The immediate mode has no effect. The command will not store into locations  $0000_{16}$  to  $7_{16}$ .

(2) SCH C1



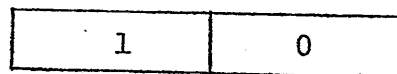
Character Placement

Load Half Word to U Register

$C(U)_{1-16} := C(A)$  memory addressed as 16-bit words  
 $C(U)_{17-32} := 0$

The half word specified by the effective address is loaded into the right half of the U register. The left half of the U register is set to zero. The immediate mode has no effect.

(2) LHW C2



Half word placement



Store Half Word from U Register

Mnemonic Code	Hex Code
---------------	----------

C(A) := C(U)<sub>1-16</sub> memory addressed as 16-bit words. (2) SHW C3

The contents of bits 1 to 16 of the U register are stored in the half word specified by the effective address.



Half-word placement

The immediate mode has no effect. This command will not store into locations 0000<sub>16</sub> to 0007<sub>16</sub>.

Load Double Length

C(U) := C(A) (2) LDD 88  
C(X) := C(A+1)

The U register is loaded with the contents of the effective address. The X register is loaded with the contents of the effective address plus one. Immediate mode is undefined.

Store Double Length (2) STD 84

C(A) := C(U)  
C(A+1) := C(X)

Contents of the U register are stored at the location specified by the effective address, and the contents of the X register are stored at the location specified by the effective address plus one. The immediate mode has no effect. Command will not store to locations 0000<sub>16</sub> to 0007<sub>16</sub>.

Exchange Double Length

(2)

EXD

8C

C(A):::C(U)

C(X):::C(A+1)

This instruction swaps the contents of the effective address with the contents of the U register, then swaps the contents of the effective address plus one with the contents of the X register. Operation with locations 0-7 is undefined, as is immediate mode.

CONDITIONAL JUMPS AND SKIPS (EXCEPT I/O)

Skip if True (1) SKT 62

if  $\exists i(C(C_i) \wedge A_i) = 1$  then

$C(C)_{17-32} := C(C)_{17-32} + 1$

If any bits of  $C_{1-16}$  masked by the effective address are ones, the control counter is incremented by one, causing a skip.

Immediate mode has no effect.

Skip if False (1) SKF 63

if  $\forall i(C(C)_{17i} \wedge A_i) = 0$  then

$C(C)_{17-32} := C(C)_{17-32} + 1$

If all bits  $C_{1-16}$  masked by the effective address are zero, the control counter is incremented by 1, causing a skip.

Immediate mode has no effect.

Skipping on results of compares:

Compare results are set by the CM instruction.

See REGISTER SPECIFYING INSTRUCTIONS--Compare Memory with Register.

The following instructions are defined for convenience in testing the results of compare operations and other hardware status. They are special cases of SKT or SKF.

<u>Skip on Equal</u>	SKE	30063
<u>Skip on Not Equal</u>	SKNE	30062
<u>Skip on Greater</u>	SKG	10062
<u>Skip on Less</u>	SKL	20062
<u>Skip on Greater or Equal</u>	SKGE	20063
<u>Skip on Less or Equal</u>	SKLE	10063
<u>Skip on Overflow</u>	SKOV	40062
<u>Skip on No Overflow</u>	SKNO	40063

Conditional Jump Instructions:

The following instructions test the bottom three bits of the C register (overflow and compare flags) if the stated condition is met, control is transferred to the effective address. The left and right immediate bits specify the condition.

<u>Jump if Equal</u>	(1) R = Memory	JEQ	010
<u>Jump if Not Equal</u>	R ≠ Memory	JNE	410
<u>Jump if Greater</u>	R > Memory	JGT	490
<u>Jump if Greater or Equal</u>	R ≥ Memory	JGE	890
<u>Jump if Less</u>	R < Memory	JLT	C90
<u>Jump if Less or Equal</u>	R ≤ Memory	JLE	090
<u>Jump if Overflow</u>	Overflow	JOV	C10
<u>Jump if No Overflow</u>	No Overflow	JNO	810

<u>Select Jumps</u>	(1)	SJ1	460
		SJ2	860
if $SJS_i = 1$		SJ3	C60

then  $C(C)_{17-32} := A$

$(i = 2 * M_4 + M_3)$

If the select jump switch on the console specified by bits 3 and 4 of the modifier field (the immediate bits) is on, control is transferred to the effective address. If neither immediate bit is a one, this is a normal unconditional jump instruction.

See MACHINE STATUS INSTRUCTIONS---Jump.

MACHINE STATUS

No Operation (0) NOP 00

This command ignores the modifier field and has no effect other than its use of an instruction fetch time.

Jump (1) JMP 060

$C(C)_{17-32} := A$

Control is transferred to the location specified by the effective address. If the left or right immediate bits are set, this becomes a select jump. See CONDITIONAL SKIPS AND JUMPS.

Jump and Store C (1) JSC 66

$C(A) := C(C)$

$C(C)_{17-32} := A+1$

The contents of the C register are stored in the location specified by the effective address and control is transferred to the location specified by the effective address plus one.

Execute (1-2) XCT 61

$C(IR) := C(A)$

The instruction located at the effective address is executed. Immediate mode has no effect. Normal sequencing will not occur if the instruction executed modifies  $C_{17-32}$ .

<u>Set Flags</u>	(1)	SET	64
<p>if <math>A_i=1</math> then <math>C(C)_i:=1</math>  for <math>i=1, \dots, 16</math>  The bottom half of the C register is "or"ed with the effective address.</p>			
<u>Reset Flags</u>	(1)	RST	65
<p>if <math>A_i=1</math> then <math>C(C)_i:=0</math>  for <math>i:=1, \dots, 16</math>  The complement of each bit in the effective address is "and"ed with the corresponding bit in the C register (bits 1 to 16).  Immediate mode has no effect.</p>			
<u>Set Interrupt Mask</u>	(1)	SIM	C4
<p>if <math>C(A)_i=1</math> then <math>C(MK):=1</math>  for <math>i=1, \dots, 16</math>  The effective address is OR-ed with the contents of the mask.</p>			
<u>Reset Interrupt Mask</u>	(1)	RIM	C6
<p>If <math>C(A)_i=1</math> then <math>C(MK)_i:=0</math>  for <math>i=1, \dots, 16</math>  This instruction puts zeros in the mask (MK) where ones appear in a corresponding bit of the effective address</p>			

Set Interrupt Enable Toggle (1) SIT E1

$C(\text{Enint}) := A_1$

The interrupt enable toggle (Enint) is loaded with the contents of the first bit of the effective address. This operation is delayed one instruction to allow exiting from a service routine before enabling the interrupts.

Load Left Half of C (1) LLC C7

$C(C)_{17-32} := C(A)_{17-32}$

The left half of the contents of the effective address is placed in the contents of the left half of the C register. This command acts as an indirect jump.

Test Spare Bit (1) TSB E0

$C(C)_8 := C(C)_8 \vee C(U)$

Bit 8 of the C register is set to 1 if bit 0 of the U register is one.

## I/O INSTRUCTIONS

Transfer E Register (1) . TRE A2  
Bits 1-12 of the effective address select one or more of a possible 12 I/O devices. Bits 13-16 select which of 4 event times (EV1-EV4) that will be sent to other selector device. Bit 13 selects EV1; Bit 14, EV2; Bit 15, EV3; and Bit 16 selects EV4. Device selection codes are given in the I/O section.

### (Direct Memory Access Instructions)

Load I/O Action Register (1-2) LIAR (AA)

The contents of the effective address are transmitted to all the devices on the memory bus. If a device recognizes its device code in bits 1-8 of the bus, that device becomes selected and if bit 9 of the bus is a one, the device latches the relevant portions of the bus into its status register. All other devices deselect.

This instruction skips if the device selected does not reject.

Initiate Data Input/Output Transfer (1-2) IDIOT (AB)

The contents of the effective address are transmitted to the device selected. The data is interpreted as a count-address pair by the device and initiates the I/O transfer. The format for the address and count is defined for each device. If the information is accepted this instruction does not skip.

Read I/O Transfer Status (1) RIOTS (AC)

The current status of the selected device is stored in the effective address. This instruction can not reject.

Sense (1) SENSE (AD)

If any bits of the contents of the effective address, masked by the current status of the selected device are ones, the next instruction is skipped.

Sense Negative (1) SENSN (AE)

If all the bits of the current status of the selected device masked with the contents of the effective address are zeroes, the next instruction is skipped.



Select Sector

(1)

SDS

A9

$C(A)_{17-24} = \text{Sector \#}$

Bits 17 to 24 of the effective address contain the number of the drum sector which is to be read from or written on. The other bits of the effective address must be zero. Thus the sector number cannot exceed 256.

Read from Magnetic Drum

(350-500)

RSD

A8

For I:=0 step 1 until 127

Do  $C(A+I) := C(\text{selected sector})_I$

A block of 128 words is read from the sector designated by the SDS command and stored sequentially in memory beginning at the effective address. The flip-flop registers can not be addressed by this command.

Write on Drum

(350-500)

WSD

A7

For I:=0 step 1 until 127

Do  $C(\text{selected sector})_I := C(A+I)$

A block of 128 words beginning with the effective address is stored on the drum in the sector selected by the SDS command. The flip-flop registers can not be addressed by this command.

## THE CONSOLE

The power buttons on the console are the main power switches of NEBULA. They are duplicated on the power panel on the back of the computer.

The lock next to the power buttons on the console, when locked, disables the power switches. This allows the power to be locked on or off.

The standby button (STBY) has no effect upon the computer other than to turn off the panel lights. Pressing the power ON button will turn the lights on again without affecting the power to NEBULA.

The SOUND button turns on the high-speed paper tape punch and causes it to punch 6 inches of rub-out characters as it comes up to full speed. The SILENT button turns the punch off.

The temperature button-light (TEMP) indicates a power shut-off due to temperature. This also turns on the red POWER button-light and the white OVER TEMP light on the power panel. If a temperature fault occurs, NEBULA must be allowed time to cool. When the internal temperature has fallen sufficiently, push the POWER button on the power panel to clear the temperature fault. Then push the ON button to turn the machine on.

The interrupt (INTRPT) button causes a manual interrupt.

The F-state lights indicate which state NEBULA is in as it executes instructions. They are useful for observing the execution of an instruction and in machine debugging.

The STATE switch, when in the NORMAL position, does not interfere with the normal operation of the processor. When in the HOLD position, it causes the processor to pause indefinitely in the current F-state. The ONE-STEP position of the STATE switch causes the machine to execute one word time of the program. This allows the user to watch the changing F-states on the F-state lights and observe in detail the execution of the instructions.



When the STATE switch is in the HOLD position, the contents of any of the registers or of a specified memory location can be loaded into the display register and displayed on the panel lights. To display the contents of any of the registers, push the corresponding register selection button. (The STATE switch must be in the HOLD position.) The contents of the display register may now be altered manually by means of the switches immediately below the display lights. The contents of the display register may be stored back in the selected register by depressing the store (ST) switch. The contents of a memory location may be loaded into the display register by putting the address of the desired memory location into the address portion of the C register and depressing the memory selection button (M). (This may be done only when the STATE switch is in the HOLD position.) The contents of the memory location selected may now be altered by making use of the switches immediately below the display lights, then using the store switch to store the contents of the display register back into the selected memory location.

Pressing the switch to the immediate left of the store switch clears the display register.

The BREAK POINT switches provide a convenient method for stopping the NEBULA at a specified address or instruction. To stop at a desired location, set the LOCATION BREAK switches in the binary representation of the desired address, and put the red LOCATION BREAK ON-OFF switch into the ON position.

When the break occurs, the computer will stop in the F-state immediately following the  $F_0$  state of the instruction in which the break occurs. To release the break, put the STATE switch into the HOLD position. One-step using the STATE switch to the  $F_0$  state of the next instruction, then return the STATE switch to the NORMAL position.

The OPERATION BREAK switches work the same as the LOCATION BREAK switches. Set the OPERATION BREAK switches to the binary code for the

operation on which a break is desired. Put the red OPERATION BREAK ON-OFF switch into the ON position. An operation break is released as described above.

The EXECUTE switch controls the execution of instructions. In the NORMAL position, the EXECUTE switch does not interfere with the normal operation of the processor. In the HOLD position, the execution of the present instruction is suspended at the end of its  $F_0$  state. In the STEP position, the execution of the present instruction is completed and the computer stops at the end of the  $F_0$  state of the next instruction.

When in the NORMAL position, the CONTROL switch on the console does not affect the normal operation of NEBULA. In the START position, it clears U, X, & V, sets E to  $FF_{16}$  sets C to  $00010000_{16}$  and sets the F-states to  $F_0$ .

The three select jump switches can be set by the operator and tested by a running program. A switch set in the up position is 'off', a switch set in the down position is 'on'.

The DINT switch controls the interrupt system. The normal position is up. When the switch is down all interrupts except MANUAL, INTERNAL, and EXECUTIVE are disabled and will not occur until the switch is returned to the normal position.

When in the NORMAL position, the CONTROL switch on the console does not affect the normal operation of NEBULA. In the START position, it clears U, X, & V, sets E to  $FF_{16}$  sets C to  $00010000_{16}$  and sets the F-states to  $F_0$ .

The three select jump switches can be set by the operator and tested by a running program. A switch set in the up position is 'off', a switch set down position is 'on'.

The DINT switch controls the interrupt system. The normal position is up. When the switch is down all interrupts except MANUAL, INTERNAL, and EXECUTIVE are disabled and will not occur until the switch is returned to the normal position.

LIST OF INSTRUCTIONS

INSTRUCTIONS LISTED BY MNEMONIC. #EA# STANDS FOR EFFECTIVE ADDRESS#. PARANS AROUND SYMBOLS MEAN #CONTENTS OF# AND SYMBOL FOLLOWED BY SYMBOLS IN PARANS MEANS CONTENTS OF BITS WITHIN LOCATIONS.

PAGE MNEMONIC HEX CODE

PAGE	MNEMONIC	HEX CODE	DESCRIPTION
4-3	ADC	00000093	ADD (EA) TO (CR)
4-3	ADQ	00000093	ADD (EA) TO (ER)
4-3	ADU	00000033	ADD (EA) TO (UR)
4-3	ADV	00000003	ADD (EA) TO (VR)
4-3	ADX	00000053	ADD (EA) TO (XR)
4-4	ANE	000000B7	LOGICALLY AND (ER) WITH (EA)
4-4	ANU	00000037	LOGICALLY AND (UR) WITH (EA)
4-4	ANV	000000D7	LOGICALLY AND (VR) WITH (EA)
4-4	ANX	00000057	LOGICALLY AND (XR) WITH (EA)
4-4	ASD	00000030	SHIFT (UX) RIGHT TOGETHER EXTENDING SIGN
4-4	AVE	000000BD	ABSOLUTE VALUE OF (ER)
4-5	AVU	0000003D	ABSOLUTE VALUE OF (UR)
4-5	AVV	000000DD	ABSOLUTE VALUE OF (VR)
4-5	AVX	0000005D	ABSOLUTE VALUE OF (XR)
4-4	CMC	00000098	COMPARE (CR) WITH (EA) AND SET C(1-2) ACCORDINGLY
4-4	CME	000000B8	COMPARE (ER) WITH (EA) AND SET C(1-2) ACCORDINGLY
4-4	CMU	00000038	COMPARE (UR) WITH (EA) AND SET C(1-2) ACCORDINGLY
4-4	CMV	000000D8	COMPARE (VR) WITH (EA) AND SET C(1-2) ACCORDINGLY
4-4	CMW	000000F8	COMPARE (WR) WITH (EA) AND SET C(1-2) ACCORDINGLY
4-4	CMX	00000058	COMPARE (XR) WITH (EA) AND SET C(1-2) ACCORDINGLY
4-4	CMZ	00000018	COMPARE (ZR) WITH (EA) AND SET C(1-2) ACCORDINGLY
4-10	DIV	0000002E	DIVIDE (UX) BY (EA) RESULT IN XR, REMAINDER IN UR
4-11	DSZ	00000022	DECREMENT LEFT HALF OF (EA) AND SKIP IF ZERO
4-3	EQC	00000096	EXCLUSIVE OR (CR) WITH (EA)
4-3	EQE	000000B6	EXCLUSIVE OR (ER) WITH (EA)
4-3	EQU	00000036	EXCLUSIVE OR (UR) WITH (EA)
4-3	EQV	000000D6	EXCLUSIVE OR (VR) WITH (EA)
4-3	EOX	00000056	EXCLUSIVE OR (XR) WITH (EA)
4-4	EXC	0000009C	EXCHANGE (CR) WITH (EA)
4-4	EXE	000000BC	EXCHANGE (ER) WITH (EA)
4-4	EXU	0000003C	EXCHANGE (UR) WITH (EA)
4-4	EXV	000000DC	EXCHANGE (VR) WITH (EA)
4-4	EXX	0000005C	EXCHANGE (XR) WITH (EA)
4-2	EZJ,GE	000008B0	JUMP TO EA IF (ER) .GE. ZERO
4-2	EZJ,LT	00000C80	JUMP TO EA IF (ER) .LT. ZERO
4-6	FAD	00000043	FLOATING ADD (UR) AND (EA)
4-7	FDV	0000000E	FLOATING DIVIDE (UR) AND (EA)
4-7	FMP	00000005	FLOATING MULTIPLY (UR) AND (EA)
4-6	FSB	00000044	FLOATING SUBTRACT UR AND (EA)
4-10	ISZ	00000021	INCREMENT LEFT HALF OF (EA) AND SKIP IF ZERO
4-11	IX0	0000001F	INCREMENT (X0) BY EA AND SKIP IF ZERO
4-11	IX1	0000003F	INCREMENT (X1) BY EA AND SKIP IF ZERO
4-11	IX2	0000005F	INCREMENT (X2) BY EA AND SKIP IF ZERO
4-11	IX3	0000007F	INCREMENT (X3) BY EA AND SKIP IF ZERO
4-11	IX4	0000009F	INCREMENT (X4) BY EA AND SKIP IF ZERO
4-11	IX5	000000BF	INCREMENT (X5) BY EA AND SKIP IF ZERO
4-11	IX6	000000DF	INCREMENT (X6) BY EA AND SKIP IF ZERO
4-11	IX7	000000FF	INCREMENT (X7) BY EA AND SKIP IF ZERO
4-19	JEQ	00000010	JUMP TO EA IF LAST COMPARE GAVE REG .EQ. (EA)
4-19	JGE	00000890	JUMP TO EA IF LAST COMPARE GAVE REG .GE. (EA)
4-19	JGT	00000490	JUMP TO EA IF LAST COMPARE GAVE REG .GT. (EA)
4-19	JLE	00000090	JUMP TO EA IF LAST COMPARE GAVE REG .LE. (EA)
4-19	JLT	00000C90	JUMP TO EA IF LAST COMPARE GAVE REG .LT. (EA)
4-20	JMP	00000060	JUMP TO EA
4-19	JNE	00000410	JUMP TO EA IF LAST COMPARE GAVE REG .NE. (EA)
4-19	JNO	00000810	JUMP IF OVERFLOW [C(3)] IS NOT SET
4-19	JOV	00000C10	JUMP IF OVERFLOW [C(3)] IS SET
4-20	JSC	00000066	STORE (CR) IN EA AND JUMP TO EA+1
4-16	LCH	000000C0	LOAD ER WITH CHARACTER SPECIFIED BY EA
4-2	LDC	00000091	LOAD CR WITH (EA)
4-17	LDD	00000083	LOAD UR WITH (EA) AND LOAD XR WITH (EA+1)
4-2	LDE	000000B1	LOAD ER WITH (EA)
4-2	LDU	00000031	LOAD UR WITH (EA)
4-2	Ldv	000000D1	LOAD VR WITH (EA)
4-2	LdX	00000051	LOAD XR WITH (EA)
4-16	LHW	000000C2	PUT IN LOWER HALF OF UR HALF WORD SPECIFIED BY EA
4-11	LX0	0000081F	LOAD X0 WITH EA AND SKIP IF ZERO
4-11	LX1	0000083F	LOAD X1 WITH EA AND SKIP IF ZERO
4-11	LX2	0000085F	LOAD X2 WITH EA AND SKIP IF ZERO
4-11	LX3	0000087F	LOAD X3 WITH EA AND SKIP IF ZERO
4-11	LX4	0000089F	LOAD X4 WITH EA AND SKIP IF ZERO
4-11	LX5	000008BF	LOAD X5 WITH EA AND SKIP IF ZERO
4-11	LX6	000008DF	LOAD X6 WITH EA AND SKIP IF ZERO

4-11	LX7	000008FF	LOAD X7 WITH EA AND SKIP IF ZERO
4-10	MUL	00000025	MULTIPLY (UR) AND (EA); RESULT IN UX
4-5	NGE	000000BE	NEGATE (ER)
4-5	NGU	0000003E	NEGATE (UR)
4-5	NGV	000000DE	NEGATE (VR)
4-5	NGX	0000005E	NEGATE (XR)
4-20	NOP	00000000	NO OPERATION
4-9	NRM	00000083	NORMALIZE (UX); SHIFT RESULT IN VR
4-3	ORE	00000085	LOGICALLY OR (ER) WITH (EA)
4-3	ORU	00000035	LOGICALLY OR (UR) WITH (EA)
4-3	ORV	00000005	LOGICALLY OR (VR) WITH (EA)
4-3	ORX	00000055	LOGICALLY OR (XR) WITH (EA)
4-8	PCK	40000080	PACK BYTE IN UR INTO UPPER XR
4-21	RST	00000065	RESET BITS IN LOWER CR ACCORDING TO EA
4-13	SAE	0000008A	SHIFT (ER) RIGHT
4-13	SAU	0000003A	SHIFT (UR) RIGHT EXTENDING SIGN
4-13	SAV	0000000A	SHIFT (VR) RIGHT EXTENDING SIGN
4-13	SAX	0000005A	SHIFT (XR) RIGHT EXTENDING SIGN
4-3	SBC	00000094	SUBTRACT FROM (CR) THE (EA)
4-3	SBE	00000084	SUBTRACT FROM (ER) THE (EA)
4-3	SBU	00000034	SUBTRACT FROM (UR) THE (EA)
4-3	SBV	00000004	SUBTRACT FROM (VR) THE (EA)
4-3	SBX	00000054	SUBTRACT FROM (XR) THE (EA)
4-13	SCE	8000008A	SHIFT (ER) RIGHT CIRCULARLY
4-16	SGH	000000C1	STORE CHARACTER IN ER IN ADDRESS SPECIFIED BY EA
4-13	SCU	8000003A	SHIFT (UR) RIGHT CIRCULARLY
4-14	SCV	8000000A	SHIFT (VR) RIGHT CIRCULARLY
4-13	SCX	8000005A	SHIFT (XR) RIGHT CIRCULARLY
4-21	SET	00000064	SET BITS IN LOWER CR ACCORDING TO EA
4-17	SHW	000000C3	STORE LOWER UR INTO HALF WORD SPECIFIED BY EA
4-19	SJ1	00000046	JUMP TO EA IF JUMP SWITCH ONE IS ON
4-19	SJ2	00000060	JUMP TO EA IF JUMP SWITCH TWO IS ON
4-19	SJ3	00000060	JUMP TO EA IF JUMP SWITCH THREE IS ON
4-18	SKF	00000063	SKIP IF ANY BITS IN EA ARE NOT ALSO ON IN LOWER CR
4-18	SKT	00000062	SKIP IF ANY BITS IN EA ARE ALSO ON IN LOWER CR
4-15	SLA	80000080	SHIFT (UX) RIGHT EXTENDING SIGN
4-14	SLD	80000081	SHIFT (UX) LEFT ZERO FILLING CN RIGHT
4-12	SLU	80000039	SHIFT (UR) LEFT ZERO FILLING CN RIGHT
4-12	SLV	80000009	SHIFT (VR) LEFT ZERO FILLING CN RIGHT
4-12	SLX	80000059	SHIFT (XR) LEFT ZERO FILLING CN RIGHT
4-15	SRA	00000080	SHIFT (UX) RIGHT EXTENDING SIGN
4-14	SRD	00000081	SHIFT (UX) RIGHT ZERO FILLING CN LEFT
4-12	SRE	00000089	SHIFT (ER) RIGHT ZERO FILLING CN LEFT
4-12	SRU	00000039	SHIFT (UR) RIGHT ZERO FILLING CN LEFT
4-12	SRV	00000009	SHIFT (VR) RIGHT ZERO FILLING CN LEFT
4-12	SRX	00000059	SHIFT (XR) RIGHT ZERO FILLING CN LEFT
4-2	STC	00000092	STORE (CR) IN EA
4-17	STD	00000084	STORE (UR) IN EA AND (XR) IN EA+1
4-2	STE	00000082	STORE (ER) IN EA
4-2	STU	00000032	STORE (UR) IN EA
4-2	STV	00000002	STORE (VR) IN EA
	STW	000000F2	STORE (WR) IN EA
4-2	STX	00000052	STORE (XR) IN EA
	STZ	00000012	STORE (ZR) IN EA
4-22	TSB	000000EC	SET CR(6) IF THE SPARE BIT OF UR IS SET
4-4	TSC	00000098	SKIP IF C(EA) IS ONE
4-4	TSE	00000088	SKIP IF E(EA) IS ONE
4-4	TSU	00000038	SKIP IF U(EA) IS ONE
4-4	TSV	00000008	SKIP IF V(EA) IS ONE
4-4	TSX	00000058	SKIP IF X(EA) IS ONE
4-8	UPK	C0000080	UNPACK BYTE FROM XR INTO UR
4-2	UZJ,GE	00000030	JUMP TO EA IF (UR) .GE. ZERO
4-2	UZJ,LT	000000C0	JUMP TO EA IF (UR) .LT. ZERO
4-2	VZJ,GE	00000080	JUMP TO EA IF (VR) .GE. ZERO
4-2	VZJ,LT	000000C0	JUMP TO EA IF (VR) .LT. ZERO
4-20	XCT	00000061	EXECUTE INSTRUCTION AT EA
4-2	XZJ,GE	00000050	JUMP TO EA IF (XR) .GE. ZERO
4-2	XZJ,LT	00000050	JUMP TO EA IF (XR) .LT. ZERO



I/O INSTRUCTIONS

PAGE MNEMONIC HEX CODE

4-25	TRE	000000A2	DO I/O
3-1	SKKF2	100100A2	SKIP IF TELETYPE KEYBOARD 2 HAS CHARACTER READY
3-1	CKF2	800100A2	CLEAR KEYBOARD FLAG OF TELETYPE 2
3-1	TTI2	E00100A2	READ TELETYPE KEYBOARD 2 INTO ER
3-1	SPRF2	100200A2	SKIP IF TELETYPE PRINTER 2 IS READY TO PRINT
3-2	CPRF2	200200A2	CLEAR TTY PRINTER 2 FLAG
3-2	TT02	E00200A2	PRINT (ER) ON TELETYPE PRINTER 2
3-2	SKRF	100400A2	SKIP IF PAPER TAPE READER IS READY TO READ
3-2	CRF	200400A2	CLEAR READER FLAG
3-2	RDF	600400A2	READ CHARACTER FROM PAPER TAPE INTO ER AND MOVE FORWARD
3-2	RDB	A00400A2	READ CHARACTER FROM PAPER TAPE INTO ER AND MOVE BACK
3-2	SKPF	100300A2	SKIP IF PAPER TAPE PUNCH IS READY TO PUNCH
3-2	CPF	200300A2	CLEAR PUNCH FLAG
3-2	PCH	E00300A2	PUNCH CHARACTER IN ER ON PUNCHED TAPE
3-1	SKKF	101000A2	SKIP IF TELETYPE KEYBOARD 1 HAS CHARACTER READY
3-1	CKF	801000A2	CLEAR KEYBOARD FLAG OF TELETYPE 1
3-1	TTI	E01000A2	READ TELETYPE KEYBOARD 1 INTO ER
3-1	SPRF	102000A2	SKIP IF TELETYPE PRINTER 1 IS READY TO PRINT
3-2	CPRF	202000A2	CLEAR TTY PRINTER 1 FLAG
3-2	TT0	E02000A2	PRINT (ER) ON TELETYPE PRINTER 1
3-2	SKBR	110000A2	SKIP IF LINE PRINTER BUFFER IS READY
3-2	LPLB	210000A2	PLACE CHARACTER IN ER INTO LINE PRINTER PRINT BUFFER
3-2	LPPR	410000A2	MAKE LINE PRINTER PRINT BUFFER
3-2	LPCL	810000A2	CLEAR LINE PRINTER PRINT BUFFER AND SET TOP OF FORM
4-26	SDS	00000009	SELECT DRUM SECTOR EA
4-26	RSD	00000008	READ SECTOR FROM DRUM STARTING AT EA
4-26	WSD	00000007	WRITE SECTOR ON DRUM STARTING AT EA
4-25	LIAR	000000A9	LOAD I/O ACTION REGISTER
4-25	IDIOT	000000AA	INITIATE DATA TRANSFER
4-25	RIOTS	000000AB	READ I/O TRANSFER STATUS
4-22	SIT	000000E1	SET INTERRUPT TOGGLE TO EA(1)
4-21	RIM	000000C6	RESET BITS OF INTERRUPT MASK ACCORDING TO EA
4-21	SIM	000000C4	SET BITS IN INTERRUPT MASK ACCORDING TO EA

MNEMONIC CODE TABLE

	X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Y	00	NOP					FMP	FDV	WSD	RSD	SDS						
10	JEQ		STZ						CMZ								IX0
20		ISZ	DSZ			MUL	DIV						*SPSW				
30	UZJ	LDU	STU	ADU	SRU	ORU	EOU	ANU	CMU	SLU	SAU	TSU	EXU	AVU	NGU		IX1
40				FAD	FSB												
50	XZJ	LDX	STX	ADX	SBX	ORX	EOX	ANX	CMX	SLX	SAX	TSX	EXX	AVX	NGX		IX2
60	JMP	XCT	SKT	SKF	SET	RST	JSC				*BRT	*BRF					
70	*YZJ	*LOY	*STY	*ADY	*SBY	*ORY	*EOY	*ANY	*CMY	*SLY	*SAY	*TSY	*EXY	*AVY	*NGY		IX3
80	SLO	SAD		NRM	STD				LDD				*EXD				
90	JLE	LOC	STC	ADC	SBC		EOC		CMC			TSC	EXC				IX4
A0	*HLT		TRE							LIAR	RIOTS	IDIOT	*LPSW	*DRUM	*SIMT	*RIMT	
B0	EZJ	LOE	STE	ADE	SBE	ORE	EOE	ANE	GME	SLE	SCE	TSE	EXE	AVE	NGE		IX5
C0	LCH	SCH	LHW	SHW	SIM		RIM	LLC									
D0	VZJ	LDV	STV	ADV	SBV	ORV	EOV	ANV	CMV	SLV	SAV	TSV	EXV	AVV	NGV		IX6
E0	TSB	SIT															
F0			STW						CMW								IX7

KEY:  
 OPCODE = Y+X  
 \*---NOT YET IMPLEMENTED

6-4

COMPUTER CENTER PUBLICATIONS  
pertaining to the NEBULA  
Computer

CC-66-1	O.P.	Progress Report on the NEBULA Computer
CC-66-12	O.P.	NEBULA: A Digital Computer Using 20 Mc Glass Delay Line Memory
CC-67-6	O.P.	Floating Point Package #1
CC-67-8	O.P.	Progress Report on the NEBULA Computer
CC-67-9		Evaluation of Three Content-Addressable Memory Systems Using Glass Delay Lines
CC-67-11		Floating Point Package #2
CC-67-21	Obs	Star
CC-68-5		Design and Evaluation of a Content-Addressable Memory System
CC-68-15	Obs	Cam Star
CC-68-23		The Logical Design of the NEBULA Computer
CC-68-56	O.P.	NEBULA Floating Point Hardware
CC-69-13	Obs	Operations Manual for the NEBULA Fortran System
CC-70-1	O.P.	The NEBULA Computer (A General Manual)
CC-70-5		Glossary of Important Terms for the NEBULA Computer

---

Obs: Obsolete  
O.P.: Out-of-Print