G-20 Central Processor Service Manual

VOLUME 1

TABLE OF CONTENTS

CENTRAL PROCESSOR SERVICE MANUAL

VOLUME I

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The intention of this manual is to provide the information necessary to an understanding of the electronic and logical design of the G–20 Central Processor. In these discussions, acquaintance with the following has been assumed:

1) the basic principles of electronics,
2) Boolean algebra,
3) the use of binary–coded octal notation.

Multitudinous published texts are devoted to the exposition of these ideas. The unique aspects of the design of a central processor as complex as the G–20 provide sufficient scope for a single manual; the inclusion of more general information would obscure the issue.

Similarly, familiarity with the G–20 computing system and the underlying system concepts has been assumed. The General Reference Manual is devoted to a thorough analysis of the system and its components; inclusion of this material in the current manual would involve unnecessary duplication. Actions taken by the Central Processor logic that involve communication with other units in the system can be understood only in terms of the system as a whole. In fact, a certain amount of confusion is bound to result from analyzing the detailed workings of the Central Processor without knowledge of the system requirements that dictated the implementation. This is particularly true of the communication and interrupt systems since these are the most unique elements of the system design. It should be emphasized that a discussion of these systems has a proper place in this manual and that the material has been omitted only because it is so well presented in the Reference Manual. Discussions in this manual presuppose acquaintance with this information.

The manual has been published in two volumes with Volume 2 being devoted entirely to the logic. Volume 1 begins with a discussion of the overall design of the computer, proceeding to a detailed analysis of the electronic design and, finally, to particular information dealing with the maintenance of the Central Processor. Material contained in Chapter 2 (Information Format) constitutes an exception to the rule of not repeating information from the Reference Manual. This chapter includes a discussion of the various G–20 word formats, a list of the G–20 commands (operation code list), and a brief description of the operand assembly process. The latter is included here to provide orientation into the manner of program execution. (Sections 9.1 – 9.3 cover the same material from the point of view of the logical implementation.) Most of this information appears in the General Reference Manual. Its inclusion here can be justified on the grounds that it is referred to frequently and should, therefore, be readily available for reference. Further, its basic nature is such that all logical manipulations discussed in the manual result from some aspect of the format and, thus, a solid grasp of these ideas is essential. Chapter 3 (Introduction to Machine Organization) is devoted to a general description of the machine layout with particular emphasis being placed on the machine diagram. Chapter 4 describes the Central Processor control registers: their uses, the existing transfer paths, means available for shifting information, and the general relationships between these registers and the logic. The importance of the CD register (Command Decoding) is stressed since the decoding that takes place here controls all subsequent activities. Chapter 5 introduces the Arithmetic Unit:

the registers and their inter-relationships, adder circuitry and operation, exponent circuitry, etc. Chapter 6 covers the operation of the memory system: the circuits, handling of address decoding and memory parity, timing considerations, connections of the system to external memory, and the operation of the system using external memory units. The basic Central Processor electronic design is discussed in Chapter 7. This includes descriptions of logic circuits, communication circuits, and so on. Chapter 8 is devoted to a generalized discussion of maintenance of the Central Processor: the tools required and the procedures to be followed.

The analysis of the logic in Volume 2 begins with a description of Master Control, the section of the logic that determines the handling of each command word, in Chapter 9. Chapter 10 introduces several short, miscellaneous operations, Chapter 11, the logic involved in memory operations, 12, input/output, and 13, the use of the adder circuitry. This breakdown is discussed in detail in Sections 9.1 and 10.1.

Basic documentation for the G-20 Central Processor consists of the drawings, schematics, wire list, and logic flow charts. All of the flow charts appear in Volume 2. Relevant sections of the drawings are included in Volume 1. The wire list and schematics, which relate the logic to the circuits that implement it, are available to those who require use of them.

Certain conventions in terminology have been used throughout this manual: one is discussed here, the others are pointed out as they arise. In each instance an attempt has been made to choose the most generally applicable term. Even so, no set of terms could be expected to satisfy every case; exceptions are bound to occur.

As an example of the choice of terminology, consider the problem of register bit designations. G-20 floating-point arithmetic is carried out to an accuracy of 42 bits so that all registers involved in such operations must be 42 flip-flops long. However, some registers have extra flip-flops at one or both ends (see Figure 3.2-2) that provide a means for retaining significance when certain overflow conditions occur. (The value is brought into normal range by means of shifts in conjunction with exponent increments or decrements.) These extra bits are also useful in the performance of shifts between registers. The existence of extra positions means that the least significant flip-flop in some registers holds the 0 order bit (Bit X 2^0) while in others it holds the -1 (Bit X 2^{-1}), -3 (Bit X 2^{-3}), or the -4 (Bit X 2^{-4}) order bit. Thus, reference to a particular bit on the basis of the number of the flip-flop storing it would not indicate the order of the bit. For this reason the convention adopted is tied to the order of the information, not to the number of the flip-flop. Each bit is named for the power of 2 with which it is associated. For example, in register S which has a single extra flip-flop at the lower end, the 0 order bit is stored in the second flip-flop but is referred to as Bit 0. For the M register, which has three extra flip-flops at the lower end, the 0 order bit will still be called Bit 0, while the first flip-flop in M contains Bit -3, the second, Bit -2, etc., and the fifth, Bit 1, the sixth, Bit 2, etc. This avoids the confusion that would result from use of flip-flop designations.

CHAPTER 2

INFORMATION FORMAT

SECTION 2. 1 – MACHINE LANGUAGE WORDS

In a computing device as versatile as the G–20 Central Processor, it is necessary to deal with various types of information. Yet, the memory modules are of standard design with 32 bits reserved for the storage of a Central Processor word. (A memory word consists of 33 bits, but the thirty–third bit is reserved for parity.) It therefore becomes necessary to establish a system that allows the programmer to know exactly what information each 32–bit Central Processor word contains, and at the same time allows the Central Processor to know how to process it. Figure 2. 1–1 lists all possible variations of word format that the Central Processor can use. These words are all in machine language (binary number system of 1's and 0's) that the Central Processor can either decode or use directly in information manipulations. These machine language words present the eventual form of all commands and data used by the Central Processor no matter how sophisticated the original program language may be.

<u>2. 1–1 COMMAND WORD FORMAT</u> All instructions of a program processed by the Central Processor will appear in the command word format shown in Figure 2. 1–1. An examination of this figure reveals that the command word is broken down into several distinct sections. The 15 least significant bits (bits 0 to 14) are referred to as the Base

Address, A. This part of the command may be used as an operand or an address of an operand depending upon the mode of operation. (Operating modes are discussed in Section 2.3.) The range of numbers in this region is from 0 to $32,767_{10}$. Thus, it can be seen that if the A field is being used as an address of an operand, any location in memory can be selected.

The next 6—bit section of the command word (bits 15 to 20) is the index, I, field and designates index addresses. This I field makes it possible to reference an index location and some other memory location while using a single command word, thus saving time and memory space. Since 6 bits have been assigned to this section, the I field can address only the first 64 locations in memory. An I field equal to zero is a special case that means no index address is specified and consequently, only 63 index addresses are available. We will digress momentarily from the command word and discuss these index addresses since an early, basic understanding of them and their uses is quite important.

Programmers often have need for counters within their programs. These counters, or tallies, are incremented or decremented each time a predetermined event occurs during the processing of a program. The number remaining in the tally at the end of the processing of the program may be included as part of the result. Also, there are many cases where it is not only desirable to maintain a tally on a particular operation, but also to know when a predetermined number of repetitions has occurred. Thus, a tally and a test are needed. Any location in memory could be used as a tally, but it would take three commands to do a tally or a tally and test operation. A group of commands designed to provide modifications of the index locations in memory, however, can accomplish a tally or a tally and test operation by the use of only one of these special commands. The numerical value of this changing tally stored in one of the index locations can also be used to modify the operand X of a command word. This use of the index locations is discussed in Section 2. 3.

The 7 bits of the opcode section of the command word (bits 21 to 27) contain one of the opcodes (operation codes) listed in Table 2. 2–1. The configuration of these 7 bits will later be decoded by the Central Processor to determine necessary steps and internal paths used in the processing of the opcode. It will be noted that Table 2. 2–1 lists all of the opcodes as 3–digit octals (9 bits), whereas only 7 bits are available for the opcode in the command word. This is acceptable, however, since the most significant octal digit of an opcode is never greater than 1 octally (001 in binary). Thus, it is seen that indeed only 7 bits are needed to represent any opcode octal representation. The two most significant bits of the most significant octal of the opcode are used by the command word (bits 28 and 29) to indicate the mode of operation. There are four operating modes available (discussed in Section 2. 3). The opcode octal representation as commonly listed will appear quite different from the corresponding octal in the command word. This relationship is shown in Table 2. 1–1.

TABLE 2. 1–1 Command Word Octal Corresponding to Most Significant Digit of Opcode List Presentation

	Most Significant Octal Digit in Opcode List	
Mode	0	1
0	0	1
1	2	3
2	4	5
3	6	7

The remaining 2 bits of the command word are flag bits. These are flag bits which provide programmable interrupts.

Double Precision Numbers. Double precision number words are made up of two 32–bit Central Processor words. The mantissa of the first of the two words contains the 21 least significant bits of the 42–bit operand and the mantissa of the second word contains the 21 most significant bits. Bits 21 through 31 are the same as in a single precision number word with the exception that bit 29, the length tag will be a 1, designating a double precision word. It should be noted that bits 21 through 31 of both words of the double precision number will be identical. Bits 31 through 21 of the second word, however, are not needed by the Central Processor decoding and are ignored. When bit B29 is high and logic opcodes are not being processed, all operations will be done in floating point even if pickapoint operations are indicated by the pickapoint mode flip–flop, UPE.

Floating Point Integer. Floating point integer words are also very similar to the single precision number word format, except that the exponent (bits 21 to 26) is always set to zero and the sign of the exponent is plus (bit 27 always set to zero). It should also be noted that the contents of the mantissa of the floating point integer are quite different in basic concept from the contents of the mantissa of a single precision number word. The mantissa of the single precision number word contains the rounded 21 most significant bits of a variable exponent number, while the mantissa of a floating point integer contains the truncated 21 least significant bits of a zero exponent (8^0) number.

Pickapoint Single Precision Number. In the pickapoint single precision word, the mantissa (bits 0 through 26) contains the binary representation of the operand. The exponent and the sign of the exponent, normally positioned in bits 21 through 27, in the pickapoint mode of operation are held in the PE register (a 7–bit register). Since the contents of the PE register can only be changed by the programmer, this allows computation referenced to some preassigned exponent value. This facilitates the processing of fixed point computations. Bit 27 is 1 indicating a pickapoint single precision number as opposed to a pickapoint integer where bit 27 equals 0. Bit 28 holds the sign of the mantissa (0 = plus, 1 = minus), and the length tag (bit 29) is 0 indicating a single rather than double precision word (pickapoint mode operations can only occur in single precision). Bits 30 and 31 are the flag bits.

Pickapoint Integer. The pickapoint integer is very similar to the pickapoint single precision number format. With pickapoint integer, however, bit 27 of the word is set to zero to indicate an integer word and, therefore, the 7 bits of the PE register are ignored ~~set to zero (zero exponent~~). Also, the basic concept of the mantissa of the pickapoint single precision number and of the pickapoint integer is quite different. The mantissa of the pickapoint single precision number contains the rounded 27 most significant bits of a word to some variable, predetermined exponent, whereas the mantissa of a pickapoint integer contains the truncated 27 least significant bits of a zero exponent number (8^0).

Logic Word. The logic word is made up of two parts, the flags (bits 30 and 31) and 30 bits of information (bits 0 to 29) that have no exponent associated with them. These words have no use in actual mathematical computations; however, used with the logic commands they provide an easy means of manipulating or changing command or data words within the machine. The previously discussed use of logic words to provide flags to command or data words once they are in memory is one illustration of their uses.

FIGURE 2.1–1 Central Processor Word Formats

FLAGS	MODE	OPCODE	INDEX	BASE ADDRESS
31 30	29 28	27 21	20 15	14 0

STANDARD COMMAND WORD

LENGTH TAG, MANTISSA SIGN, EXPONENT SIGN (bits 29, 28, 27)

FLAGS	0	±	±	EXPONENT	MANTISSA
31 30	29	28	27	26 21	20 0

SINGLE PRECISION NUMBER, FLOATING POINT

LENGTH TAG, MANTISSA SIGN, EXPONENT SIGN (bits 29, 28, 27)

FLAGS	1	±	±	EXPONENT	MANTISSA
31 30	29	28	27	26 21	20 0

DOUBLE PRECISION NUMBER, RIGHT HALF (X)

SAME AS ABOVE, SEE TEXT.	MANTISSA
31 21	20 0

DOUBLE PRECISION NUMBER, LEFT HALF, (X+1)

LENGTH TAG, INTEGER SIGN, EXPONENT SIGN (bits 29, 28, 27)

FLAGS	0	±	0	EXPONENT = 0	INTEGER
31 30	29	28	27	26 21	20 0

INTEGER FLOATING POINT

LENGTH TAG, MANTISSA SIGN, INTEGER TAG (bits 29, 28, 27)

FLAGS	0	±	1	MANTISSA
31 30	29	28	27	26 0

SINGLE PRECISION NUMBER, PICKAPOINT

LENGTH TAG, INTEGER SIGN, INTEGER TAG (bits 29, 28, 27)

FLAGS	0	±	0	INTEGER
31 30	29	28	27	26 0

INTEGER PICKAPOINT

FLAGS	
31 30	29 0

LOGIC WORD

SECTION 2. 2 – OPCODE LIST

TABLE 2. 2–1 Opcode List

OPCODE NOTATION CROSS REFERENCE

Octal	Alpha	Engr.
000	OCA	N0
001	FOP	T0
002	ADX	B2
005	CLA	A0
006	AXT	B6
011	IOZ	S0
012	LXP	B0
013 [opcode]	REP	M0
015	CAL	L0
016	XPT	B4
017	TRA	X0
020	OCS	N1
021	FOM	T1
022	SUX	B3
025	CLS	A1
026	SXT	B7
031	ICZ	S1
032	LXM	B1
033 [*]	BTR	M1
035	CCL	L1
036	XMT	B5
037	TRE	X1
040	OAD	N2
041	FSP	T2
045	ADD	A2
051	ISN	S2
052	ERO	R2
053	DIV	D0
055	ADL	L2
056	LDR	R0
057	RDV	D1
060	OSU	N3
061	FGO	T3
065	SUB	A3
071	IUO	S3
072	ERA	R3
073	STZ	P4
075	SUL	L3
076	EXR	R1
077	MPY	D2
100	OAN	N4
101	FSM	T4
105	ADN	A4
111	IEZ	S4
113	STS	P1
115	EXL	L4
117	TDC	X5
120	OSN	N5
121	FLO	T5
125	SUN	A5
131	IEC	S5
133	STI	P3
135	ECL	L5
137	SKP	X2
140	OAA	N6
141	FSN	T6
145	ADA	A6
151	IUZ	S6
153	STD	P0
155	UNL	L6
157	TLC	X4
160	OSA	N7
161	FUO	T7
165	SUA	A7
171	IUC	S7
173	STL	P2
175	UCL	L7
177	TRM	X3
* 000	BTD6	
* 020	BRD6	
* 040	BTC6	
* 100	BTD8	
* 120	BRD8	
* 140	BTC8	

USERS CODE	OCTAL CODE	ENGR. CODE	NAME	OPERATION
ADDRESS PREPARATION				
OCA	000	N0	Clear and Add	$X \rightarrow (OA)$
OCS	020	N1	Clear and Subtract	$-X \rightarrow (OA)$
OAD	040	N2	Add	$X + (Acc) \rightarrow (OA)$
OSU	060	N3	Subtract	$-X + (Acc) \rightarrow (OA)$
OAN	100	N4	Add and Negate	$-[\;X + (Acc)] \rightarrow (OA)$
OSN	120	N5	Subtract and Negate	$-[-X + (Acc)] \rightarrow (OA)$
OAA	140	N6	Add and Take Absolute Value	$\lvert X + (Acc)\rvert \rightarrow (OA)$
OSA	160	N7	Subtract and Take Absolute Value	$\lvert -X + (Acc)\rvert \rightarrow (OA)$
ADD AND SUBTRACT				
CLA	005	A0	Clear and Add	$X \rightarrow (Acc)$
CLS	025	A1	Clear and Subtract	$-X \rightarrow (Acc)$
ADD	045	A2	Add	$X + (Acc) \rightarrow (Acc)$
SUB	065	A3	Subtract	$-X + (Acc) \rightarrow (Acc)$
ADN	105	A4	Add and Negate	$-[\;X + (Acc)] \rightarrow (Acc)$
SUN	125	A5	Subtract and Negate	$-[-X + (Acc)] \rightarrow (Acc)$
ADA	145	A6	Add and Take Absolute Value	$\lvert X + (Acc)\rvert \rightarrow (Acc)$
SUA	165	A7	Subtract and Take Absolute Value	$\lvert -X + (Acc)\rvert \rightarrow (Acc)$
Note: The Accumulator is not disturbed in these opcodes.				
ADD AND SUBTRACT TESTS				Criterion
FOP	001	T0	If Operand Plus	$X > 0$
FOM	021	T1	If Operand Minus	$-X > 0$
FGO	061	T3	If (Acc) Greater Than Operand	$-X + (Acc) > 0$
FLO	121	T5	If (Acc) Less Than Operand	$-[-X + (Acc)] > 0$
FUO	161	T7	If (Acc) Unequal to Operand	$\lvert -X + (Acc)\rvert > 0$
FSP	041	T2	If Sum Plus	$X + (Acc) > 0$
FSM	101	T4	If Sum Minus	$-[\;X + (Acc)] > 0$
FSN	141	T6	If Sum Non–zero	$\lvert X + (Acc)\rvert > 0$
Note: If test is satisfied, go to next command of program (NC). If test is not satisfied, go to NC + 1.				
LOGIC OPERATIONS				
CAL	015	L0	Clear and Add Logic Word	$_{31}[\;\underline{X}\;]_0 \rightarrow (Acc)$
CCL	035	L1	Clear and Add Complement of Logic Word	$_{31}[\;\overline{X}\;]_0 \rightarrow (Acc)$
ADL	055	L2	Add Logic Word	$_{31}[\;X + (Acc)]_0 \rightarrow (Acc)$
SUL	075	L3	Subtract Logic Word	$_{31}[-X + (Acc)]_0 \rightarrow (Acc)$
EXL	115	L4	Extract With Logic Word	$_{31}[\;\underline{X} \wedge (Acc)]_0 \rightarrow (Acc)$
ECL	135	L5	Extract With Complement of Logic Word	$_{31}[\;\overline{X} \wedge (Acc)]_0 \rightarrow (Acc)$
UNL	155	L6	Unite With Logic Word	$_{31}[\;\underline{X} \vee (Acc)]_0 \rightarrow (Acc)$
UCL	175	L7	Unite With Complement of Logic Word	$_{31}[\;\overline{X} \vee (Acc)]_0 \rightarrow (Acc)$
Note: $0 \rightarrow {}_{41}[(Acc)]_{32}$ for all of these codes.				
LOGIC TESTS				Criterion
IOZ	011	S0	If Operand Zero	$_{31}[\;\underline{X}\;]_0 = 0$
ICZ	031	S1	If Complement of Operand Zero	$_{31}[\;\overline{X}\;]_0 = 0$
ISN	051	S2	If Sum Non–zero	$_{31}\lvert\;X + (Acc)\rvert_0 > 0$
IUO	071	S3	If Unequal to Operand	$_{31}\lvert -X + (Acc)\rvert_0 > 0$
IEZ	111	S4	If Extraction Zero	$_{31}[\;\underline{X} \wedge (Acc)]_0 = 0$
IEC	131	S5	If Extraction With Complement Zero	$_{31}[\;\overline{X} \wedge (Acc)]_0 = 0$
IUZ	151	S6	If Union Zero	$_{31}[\;\underline{X} \vee (Acc)]_0 = 0$
IUC	171	S7	If Union With Complement Zero	$_{31}[\;\overline{X} \vee (Acc)]_0 = 0$
Note: If test is satisfied, go to next command of program (NC). If test is not satisfied, go to NC + 1.				
MULTIPLY AND DIVIDE				
MPY	077	D2	Multiply	$(Acc) * X \rightarrow (Acc)$
DIV	053	D0	Divide	$(Acc) / X \rightarrow (Acc)$
RDV	057	D1	Reverse Divide	$X / (Acc) \rightarrow (Acc)$

USERS CODE	OCTAL CODE	ENGR. CODE	NAME	OPERATION
STORE OPERATIONS				
STL	173	P2	Store Logic Word	31(Acc)0 → 31 (X) 0
STD	153	P0	Store Double Precision	20(Acc)0 → 20 (X) 0 41(Acc)21 → 20(X + 1)0
STS	113	P1	Store Single Precision [a] Floating Point Mode [b] Pickapoint Mode	 41(Acc)21 → 20 (X) 0 41(Acc)15 → 26 (X) 0
STI	133	P3	Store Integer [a] Floating Point Mode [b] Pickapoint Mode	 20(Acc)0 → 20 (X) 0 0 → 21 (X)26 26(Acc)0 → 26 (X) 0
STZ	073	P4	Store Zero	0 → 31 (X) 0
INDEX OPERATIONS				
LXP	012	B0	Load Index Plus	X → (I)
LXM	032	B1	Load Index Minus	−X → (I)
ADX	002	B2	Add to Index	X + (I) → (I)
SUX	022	B3	Subtract from Index	−X + (I) → (I)
INDEX TESTS				
XPT	016	B4	Load Index Plus and Test	X → (I)
XMT	036	B5	Load Index Minus and Test	−X → (I)
AXT	006	B6	Add to Index and Test	X + (I) → (I)
SXT	026	B7	Subtract from Index and Test	−X + (I) → (I)

Note: If final value of (I) is not zero, go to next command, NC, of program. If final value of (I) is zero, go to NC + 1.

USERS CODE	OCTAL CODE	ENGR. CODE	NAME	OPERATION
REGISTER OPERATIONS				
LDR	056	R0	Load Register: * U, H, J PE	14 X 0 → (Reg. I) 6 X 0 → (Reg. I)
EXR	076	R1	Extract Register I into Itself	14[(Reg. I) ^ X]0 → (Reg. I)
ERO	052	R2	Extract Register I into OA	14[(Reg. I) ^ X]0 → (OA)
ERA	072	R3	Extract Register I into Acc	14[(Reg. I) ^ X]0 → (Acc)

Note: Registers available for these operations are CA, U, H, J, and PE.
* The CA register can be ~~load~~ used only ~~by~~ with an ERO or ERA opcode.
The PE Reg. can [illegible] be used with [illegible] LDR opcode.

USERS CODE	OCTAL CODE	ENGR. CODE	NAME	OPERATION
TRANSFER OF CONTROL				
TRA	017	X0	Transfer [to Location X]	14[X]0 → (NC)
TRE	037	X1	Transfer and Enable Interrupts	14[X]0 → (NC)
SKP	137	X2	Skip [X Words]	14[X + (NC)]0 → (NC)
TRM	177	X3	Transfer [to Location X + 1] and Mark [in Location X]	(NC) → (X) 14[X + 1] 0 → (NC)
SINGLE CHARACTER OUTPUT				
TLC	157	X4	Transmit Line Command	7[X]0 → 7(LD)0 0 → (LD8)
TDC	117	X5	Transmit Data Character	7[X]0 → 7(LD)0 1 → (LD8)
BLOCK INPUT/OUTPUT				
BTR	033 [*]	M1	Block Transmit or Receive	X is address of first operand in the block
	BTC6		Block Transmit Commands	6–bit characters
	BTC8		Block Transmit Commands	8–bit characters
	BTD6		Block Transmit Data	6–bit characters
	BTD8		Block Transmit Data	8–bit characters
	BRD6		Block Receive Data	6–bit characters
	BRD8		Block Receive Data	8–bit characters
REPEAT OPERATIONS				
REP	013 [opcode]	M0	Repeat Next Command	X = address of first operand of block.

Note: All 32 opcodes of the Add/Subtract, Add/Subtract Test, Logic, and Logic Test commands can be performed using the Repeat mode. Each Repeat operation is designated by two words, the first of which contains the opcode 013. The opcode of the second word can be any of the above mentioned commands. Commands that are to be repeated are the same as their non–repeat counterpart except that an R (to represent Repeat mode) is affixed to the mnemonic of the command. Repeat commands have a block length associated with them (base address of second word of command). A Repeat Test command will continue until the condition tested for is met [illegible] specified blk length [illegible]

OPCODE NOTATION CROSS REFERENCE

Alpha	Octal	Engr.
ADA	145	A6
ADD	045	A2
ADL	055	L2
ADN	105	A4
ADX	002	B2
AXT	006	B6
BTR	033 [*]	M1
CAL	015	L0
CCL	035	L1
CLA	005	A0
CLS	025	A1
DIV	053	D0
ECL	135	L5
ERA	072	R3
ERO	052	R2
EXL	115	L4
EXR	076	R1
FGO	061	T3
FLO	121	T5
FOM	021	T1
FOP	001	T0
FSM	101	T4
FSN	141	T6
FSP	041	T2
FUO	161	T7
ICZ	031	S1
IEC	131	S5
IEZ	111	S4
IOZ	011	S0
ISN	051	S2
IUC	171	S7
IUO	071	S3
IUZ	151	S6
LDR	056	R0
LXM	032	B1
LXP	012	B0
MPY	077	D2
OAA	140	N6
OAD	040	N2
OAN	100	N4
OCA	000	N0
OCS	020	N1
OSA	160	N7
OSN	120	N5
OSU	060	N3
REP	013 [opcode]	M0
RDV	057	D1
SKP	137	X2
STD	153	P0
STI	133	P3
STL	173	P2
STS	113	P1
STZ	073	P4
SUA	165	A7
SUB	065	A3
SUL	075	L3
SUN	125	A5
SUX	022	B3
SXT	026	B7
TDC	117	X5
TLC	157	X4
TRA	017	X0
TRE	037	X1
TRM	177	X3
UCL	175	L7
UNL	155	L6
XMT	036	B5
XPT	016	B4
* BRD6	020	
* BRD8	120	
* BTC6	040	
* BTC8	140	
* BTD6	000	
* BTD8	100	

SECTION 2.3 – OPERAND ASSEMBLY

The assembling of the operand X prior to processing an opcode is common to all opcodes used by the Central Processor. Therefore, it is seen that a thorough understanding of the process and significance of assembling the operand X is necessary for effective use of the Central Processor's capabilities. It is assumed that at this point the reader has had some contact with what is meant by the operand X and thus only a general discussion will be presented. If this assumption is incorrect, or if the reader is at all hazy about the subject after reading the following discussion, then the reader should refer to Section 2.2 of the General Reference Manual or the first section of the Central Processor Machine Language Manual. Detailed discussions of operand assembly, with examples, are provided in the above-mentioned manuals. The value of a basic understanding of this material cannot be too greatly emphasized since a thorough understanding of much of the material presented in later chapters depends upon mastery of the concepts of operand assembly.

A command word in the Central Processor, as we have seen, carries an operating mode M, an index address I, and a base address, A. All of these, along with the contents of the OA register, can be used in assembling the operand X of a command word. The exceptions to this rule are the command words with opcodes which operate on one of the Bus registers or an index location, since in these cases the index address I is used to specify the register or memory location (1 through 63) to be operated on. It should also be noted that the use of the contents of the OA register in assembling the operand X is not common practice, since the contents of the OA register are always set to zero before assembling X if the previous opcode that was processed was not one of the address preparation opcodes. The function of the address preparation opcodes is discussed in the section below dealing with the general case of operand assembly.

The operating modes associated with a command word specify the manner in which A, I, and OA are to be used in the assembling of X. Table 2.3–1 shows decoding that indicates the mode of operation.

TABLE 2.3–1 Operating Mode Decoding

	B register bits		CD register bits	
Mode	29	28	14	13
0	0	0	0	0
1	0	1	0	1
2	1	0	1	0
3	1	1	1	1

The general case of operand assembly is the most versatile and the most commonly used. In it, it is assumed that (OA), (I), and A are all available for use. There is a case that will be discussed later where the above assumption is not valid. The manner in which the operand is formed in the different modes is presented in Table 2.3–2.

By use of the general case of operand assembly, it is seen that a large number of modifications can be made to the operand by use of the contents of the OA register and indexing. In operand assembly, indexing can be very limitedly defined as using the contents of the index locations to change the operand of the command. A more complete discussion of the index locations and their uses is presented in Section 2.1. The address preparation opcodes provide a second means of modifying the operand, which, if desired, permits the contents of the Accumulator to be combined in various ways in the assembling of X.

TABLE 2.3–2 Operand Assembly, General Case

Mode	Action	Format
0	(OA) + A + (I) = X	Number
1	(OA) + (A) + (I) = X	Number
2	((OA) + A + (I)) = X	Number or Logic*
3	((OA) + (A) + (I)) = X	Number or Logic*

where OA = Operand Assembly register

X = operand designator

A = base address

I = index address

( ) = contents of (i.e., an address)

Logic* ⇒ logic format on final access for logic or logic test opcodes.

It was noted earlier that there are opcodes where the general case of operand assembly is not applicable. These opcodes belong to the Index or Bus register command groups. In these opcodes, the I field of the command word is used as an identification tag. If one of the index opcodes is being processed, the I field of the command word specifies which of the 63 index locations is to be operated on. If it is one of the Bus register opcodes, the I field indicates which of the five Bus registers (CA, J, H, U, or PE) is to be operated on. The method of assembling the operand X under these conditions is the same as shown in Table 2.3–2 with the exception that with these groups of opcodes, the I field, (I), does not enter into the assembling of X. The assembling of the operand X under these conditions is presented in Table 2.3–3.

In all instances, the arithmetic processes involved in assembling the

CHAPTER 3

INTRODUCTION TO MACHINE ORGANIZATION

SECTION 3.1 – GENERAL

Similar to most all digital computer design, the Central Processor is comprised of the four basic logic areas of: (1) Control, (2) an Arithmetic Unit, (3) Memory, and (4) Input/Output. This in itself is neither new or unusual, however, the method in which these areas of logic are handled within this particular computer needs to be thoroughly discussed since the means of implementing these areas can vary greatly.

FIGURE 3.1–1 Block Diagram of Central Processor Organization

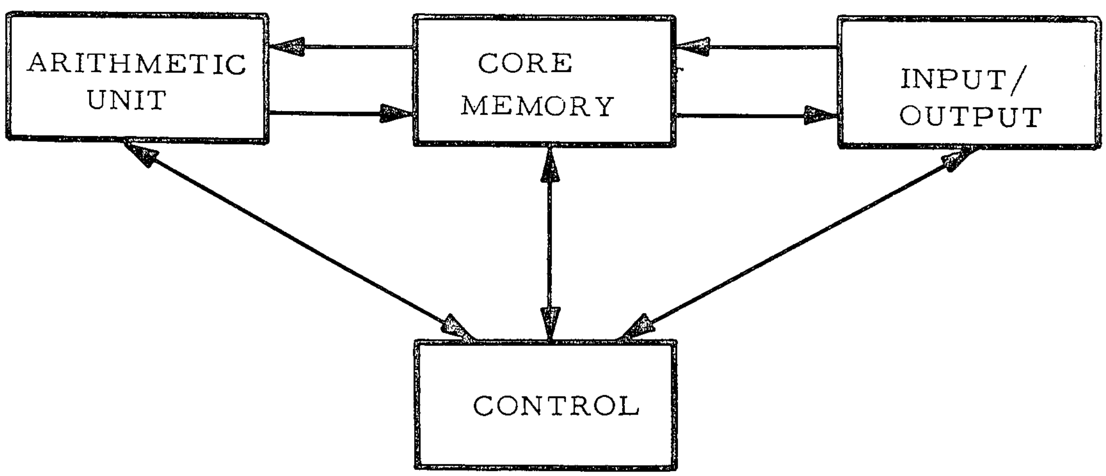

Figure 3.1–2 is a breakdown by Central Processor registers of Figure 3.1–1. This figure shows registers of the various logic areas, but this is not a complete logic breakdown. In particular, the area of control is comprised of control registers and sequencers. The sequencers are in reality a complex of logic circuitry that supervises the operation of the opcodes listed in Table 2.2–1. This logic circuitry is not shown in Figure 3.1–2 because of its complexity. Rather, for simplicity of presentation, all sequencer logic is discussed in ~~Part IV~~ Vol. 2, Sequencer Control. The discussion of control that is presented in Chapter 4 deals solely with control exercised by the registers of Figure 3.1–2 that are designated as the Control registers. Also, the reader should realize that the breakdown of input/output as presented by the Machine Diagram is not complete, since it deals only with provisions within the Central Processor for input/output. For a discussion of actual external input/output devices, the reader is referred to the General Reference Manual. If a more specific discussion of a particular input/output unit is desired, then that peripheral unit's technical manual must be consulted. It should also be pointed out that input/output operations of the Central Processor are not discussed in this part of the manual. Rather, due to the fact that a very thorough discussion of input/output is necessary in Chapter 12 of ~~Part IV~~ Vol. 2 where Sequencer Control logic is discussed, the complete presentation of input/output is postponed until Chapter 12.

FIGURE 3.1–2 Central Processor Logic Breakdown of Figure 3.1–1