

ENDICOTT
Product Development Laboratory
October 9, 1957

MEMORANDUM TO: Mr. J. J. Ingram

SUBJECT: Accounting Machine, Second Proposal

The enclosed proposal represents the results of programming the first machine. It is not a detailed proposal, but is simply a report thrown hurriedly together so as to get the new ideas recorded and circulated. Your comments will be appreciated.



FOU/is

F. O. Underwood

cc: F. V. Adams	R. G. Mork
M. E. Bear	J. K. Mosser
P. M. Brannen	R. A. Rowley
C. E. Branscomb	F. Saltz
F. Fesnak	C. B. Smith
M. Gibson - San Jose	J. H. Wellburn
I. M. Hix	R. K. Weller
T. Leary - San Jose	L. A. Wilson
W. F. Morgan	

Enclosure

SECOND ACCOUNTING MACHINE PROPOSAL

A fair amount of programming has been done on the STORED PROGRAM, VARIABLE WORD-LENGTH ACCOUNTING MACHINE. One of the most important results of the study is the realization that a considerably more economical machine can be achieved by devising a "Variable-length instruction" compatible with the variable word-length nature of the machine.

The following discussion points out some of the arguments that lead to such a conclusion:

In the first proposal, memory was composed of blocks of 100 characters each, and the addressing scheme required three character positions of an instruction to address a character of memory. Such an addressing scheme conveniently allows the addressing of a 3600 character array of storage, although a maximum of 4800 characters was theoretically possible.

The three character positions of the address part of an instruction contain a total of 18 information bits and, if employed economically, would permit the addressing of 2^{18} positions of memory. Since approximately 4000 characters of memory seem to be the maximum requirement for a machine of this kind, it develops that only 12 bits are required, since this allows the addressing of 4096 characters. The result is that only two character positions of an instruction are required when the addressing scheme operates in a binary mode, rather than an alphanumeric mode.

Another consideration which leads to this new machine organization is the fact that for the major part of any program, the instructions are taken in sequence. For most instructions, then, it is not necessary to specify the next instruction address.

It is also found that there are many times when an instruction needs no address part, such as when the feed or printer is instructed to operate. There are also occasions when one, two, three, or four addresses are required in the instruction. The proper design of a machine providing for variable-length instructions will also provide for variable-address instructions economically. This new machine design proposes to do this.

Programming reveals that we have three major classes of instructions:

Class I - TRANSFERS, ACCUMULATIONS, EDITING

Ia - This is the most common form of instruction, and is six characters in length. The first (low order) characters are the mnemonic alphanumeric operation code, permitting transfers, arithmetic operations and editing. The second two characters are the RI word address, and the last two characters are the RO word address.

Class Ib - TRANSFER & JUMP

This class of instructions is the same as class Ia, except that two additional character positions are provided at the high-order position of the instruction for the "next instruction" address, and is therefore eight characters in length.

Class Ic - LINK

This is a "NO-OP" instruction of four characters in length. The first two positions of the instruction contain the characters "NO" (meaning "no-operation") and the last two positions contain the next instruction address. This permits linking from class Ia or IIa instructions when facilities for performing class Ib or IIIb instructions are not provided, as in the smallest version of the machine.

Class Id - THREE-ADDRESS OPERATIONS

This class of instructions is of eight characters in length. The first two positions contain the Operation code, the third and fourth positions contain the address of one factor, and the fifth and sixth positions contain the address of the other factor. (Both factors are interpreted as RO words.) The seventh and eighth positions contain the RI address. An instruction of this form is used to perform operations such as $\pm A+B=C$, and also for editing in the most economical way.

When editing, the third and fourth characters of the instruction contain the "Edit Control" information address, the fifth and sixth positions contain the address of the data to be edited, and the seventh and eighth positions contain the address which is to receive the edited information. This is a new principle of print-editing, and will be more fully described later.

Class Ie - FOUR-ADDRESS OPERATIONS

This class of instruction is ten characters in length. It is the same as class Id, above, except that the ninth and tenth positions of the instruction contain the "next instruction" address.

Class II - BRANCHING

IIa - This instruction is of seven characters in length. Again, the two low-order positions of the instruction contain the operation code, and indicate the type of test to be made, such as Balance Test, Digit Test, etc. The next two positions contain the address of the next instruction if the result of the test is "YES". The fifth and sixth positions of the instruction contain the address of the data upon which the test is to be made. The last (seventh) position of the instruction contains a digit which pertains to the particular test to be made. (That is, it would contain a "7" if a particular location in memory were to be tested for the presence of a "7".) For this class of instruction, the next instruction to be executed would be the next in sequence if the result of the test were "NO".

Class IIb - This instruction is nine characters in length, and is similar to above, except that positions 7 and 8 contain the "next instruction" address if the result of the test is "NO", and position 9 contains the Test Digit.

Class III MACHIN CONTROL

IIIa- This instruction is only two characters in length. The two positions of this class of instructions contain the operation codes that cause the Carriage, Feeds, Printer, and Summary Punch to operate.

Class IIIb- This instruction is similar to above, but is four characters in length. The last two character positions contain the address of the next instruction.

As this new machine, in its maximum size, is able to operate in a "4-address" mode, the machine must contain four sets of storage-addressing rings. In order to avoid confusion, a new terminology will be introduced here. Instead of referring to words as RO or RI words, we will refer to A, B, C and I words.

The table of INSTRUCTION WORD FORMATS, dated 13 September, 1957, displays the information of the few preceding pages.

The standard machine is able to handle a maximum of two addresses per instruction, so that only instructions Ia, Ic, IIa, IIIa and IIIb apply to the standard machine.

A "first option" adds a register, called the "P-register", and permits instructions Ia, Ib, Ic, IIa, IIb, IIIa, and IIIb to be performed.

A "second option" adds another storage addressing ring to the machine, so that all classes of instruction may be performed.

It would be possible to add the C-ring and not the P-register to the standard machine, in which case only instruction class Ie could not be performed.

In the original proposal, with the fixed-length instruction, the particular parts of the instructions could always be found in particular places in memory (for instance, the OP code could be found in addresses coding in -9 and -0) so that the Units Program Ring could also act as the gating means for setting the RO and RI rings and the Operations Register.

With a variable-length instruction, the particular parts of instructions can be found anywhere in memory, so that a separate gating means is required.

In this new machine, the P-Timer provides the required gating means. The standard machine is provided with a seven-point P-Timer. Either a nine-point or eleven-point P-Timer may be provided, depending on the options.

The points of the P-Timer function as follows:

P₁-P₂ OPERATION CODE TIME

These lines gate the output of storage to the Operations Register.

P₃-P₄ B-RING SET TIME

These lines gate the output of storage to the B-ring set lines in the standard machine. In a machine optionally equipped with a P-register, P₃ and P₄ may gate the output of storage to the P-register, depending upon the OP code.

P₅-P₆ A-RING SET TIME

These lines gate the output of storage to the A-ring set lines.

P₇ DIGIT SET TIME

On the "standard" machine, this line will gate the output of storage to a character register for test purposes. On a machine equipped for optional 3-address operation, P₇ remains as Digit set time only for certain OP codes.

P₇-P₈ I/C SET TIME

These lines gate the output of storage to either the P-register or C-ring set lines, depending upon what equipment is provided and upon the OP code.

P₉ DIGIT SET TIME (Opt.)

This line will gate the output of storage to the Digit register.

P₉-P₁₀ I SET TIME

These lines will gate the output of storage to the P-register.

P₁₁ DIGIT SET TIME

This line will gate the output of storage to the Digit register, as will P₃, P₅, P₇ and P₉, depending on equipment and OP code.

The P-Timer functions only during instruction read-out time. It always starts from a reset position and continues to advance in step with the P-ring, until a word mark is sensed, at which time it stops and resets.

As previously noted, 4096 positions of storage are provided in the largest machine, and that addressing requires two characters of memory. This suggests a natural division of storage into 64 blocks of 64 characters each. Actually, memory can be thought of as a single block, with no divisions of any kind, except for the in-out areas.

With such a scheme as this, the most logical means of writing addresses is by OCTAL numbers. A properly designed "Storage Layout Chart" relieves the programmer of a difficult encoding job.

Because all addresses will be written in OCTAL on the Program Chart, and the Instruction cards are punched from this, it follows that four card-columns are required for each address, and that only the 0-7 rows will be punched in these columns.

Each two-digit octal number must be translated into a single digit (6 bit) code within the machine. Such a translation is trivial. Since it is necessary to differentiate between the octal part of the instruction and the alpha numeric part of the instruction, and 8/9-punch is used in the low-order position of each octal part.

An instruction might be written as

12435760AD

and punched in the card in the same way, with 8/9 punches in the same column as the 2, 3, 7 and 0.

During the PROGRAM LOAD operation, the instruction would be read and encoded into memory as:

C									
(32) B									
(16) A									
B									
4									
2									
1									
WM									
	12	43	57	60	A	D			

The printing out of such encoded instructions requires a simple device for decoding to the original form.

Display of instructions is straight-forward, requiring no decoding at all.

This report contains a description of a sample program to aid in obtaining a better understanding of the machine. Before entering into this description, a new method of print-editing will be described, as it applies to this machine.

There are many ways in which a field of information may be altered in editing, the most common alterations being those in which blanks or special characters are inserted into a field. The best example of this is editing of money fields.

The way in which money fields are edited may vary widely between customers and different jobs, but for any one particular job it is usually true that money amounts are edited in only one or two ways. The new method of Print editing results from the recognition of this fact and takes advantage of it so that editing can be done in a most economical fashion.

The basic concept that is involved here is that "edit-control" information can be stored in memory, and then used to alter a word of data. The "edit-control" word must consist of some special characters which do not presently exist, but for which code combinations do exist.

Print editing, then, is accomplished by having an "edit-control" word in memory, and addressing it as the "B" word, while the data to be edited is addressed as the "A" word. The "C" address is the address in which the edited data is to appear. As the "A" and "B" characters are read from memory, they are gated to the "Print Edit" unit, which may very well be the arithmetic adder (altered from its present form).

Listed below are the various characters that may appear in the "Edit-control" word, their codes, and the effect that each will have upon the data ("A") field.

b (8-2-B) BLANK INSERT

When this character is detected in the B field during an "edit" operation, the A ring will not advance for one digit time, and the B ring will advance normally. The result is that the C field will be blank in the corresponding position.

d (8-2) SIGNIFICANT ZERO

When this character is detected in the B field during an "edit" operation, the corresponding digit in the A field will not be altered, so that blanks or zeros will transfer as zeros to the C field.

f (8-4-2-1-A) FLOATING DOLLAR SIGN

When this character is detected in the B field during an "edit" operation, it will be substituted for a blank or zero in the corresponding position of the A field, providing the previous (next lower-order) character in the A field was not blank or zero.

When the Print Area is scanned to the Printer in reverse (high-to-low) order, the first "f" character detected in a field will cause a "\$" to be printed. All other "f"s in the field will print as zeros.

p (8-4-1-A) PROTECTION ASTERISK

When this character is detected in the B field during an "edit" operation, it will be substituted for zeros and blanks detected in corresponding positions of the "A" field.

When the Print Area is scanned to the printer in reverse order, asterisks will be substituted for the p's until a significant digit is detected. Thereafter, for the remainder of the field, zeros are substituted for the p's.

z (8-4-2-1) ZERO SUPPRESS

When this character is detected in the B field during an "edit" operation, it will be substituted for zeros detected in corresponding positions of the A field.

When the Print Area is scanned to the Printer in reverse order, blanks will be substituted for the z's until a significant digit is detected.

Thereafter, for the remainder of the field, zeros will be substituted for the z's.

m (8-2-A) MINUS SIGN (CONDITIONAL)

This character, when detected in the B field during an "edit" operation, must necessarily be after, or at the same time that the unit position of the A field has been sensed, and will cause minus sign to enter the corresponding position of the C field if a minus sign was detected in the unit position of the A field. It will also prevent the A ring from advancing for one digit time.

, (8-2-1-B) COMMA

This character, when detected in the B field during an "edit" operation, will normally be transferred to the C field, and will cause the A ring to not advance for one digit time.

If the next lower-order character entered the C field as an f (floating dollar sign) then the comma will still enter the C field as a comma. If the next lower order character in the B field is an f, but it did not enter the C field as an f, then the comma will enter the C field as a new character ¶ (8-4-2-1-B). On the reverse scan, this position can then print either a comma or a dollar sign, depending on whether the next higher order digit was significant.

If the next lower-order position in the B field contains a p, then a new character, ⚡, (8-4-1-B) will enter the corresponding position of the C field. On the reverse scan, this position can then print either a comma or asterisk, depending upon whether the next higher-order digit was significant.

r (8-4-1-A-B) CR (CONDITIONAL)

When this character is detected in the B field during an edit operation at the same time as or later than (it cannot be otherwise) the unit position of the A field, the sign of the A field will be tested, and if minus, an R will be stored in the C field, the C ring will advance, and a C will be stored in the C field, whereupon the B and C rings will advance.

If the sign of the A field is found to be plus, then blanks will be stored, rather than the C and R.

All other alphabetic, numeric and special characters, when detected in the B field during an "edit" operation, will transfer to the C field and also cause the A ring to not advance for one digit time.

Included in this report are some specific examples of print-editing, which serve to illustrate some of the more common forms. Note that one control field may serve to edit different fields, and in different ways. For instance, one control field may serve to edit all money fields, both for listing and total printing.

This method of editing, which is new, is an extension of one of the basic concepts upon which this machine is designed: that is; once one buys a core memory of any size, it is relatively inexpensive to increase the size of the core array, and therefore it should be economical to incorporate a great many of the machine functions into the core array and thereby eliminate a variety of other components.

It may be possible to extend this idea even further by incorporating the arithmetic functions in the core array. Future work will involve the investigation of such possibilities.

One other thing to be noted concerning "Print-Editing" is that although the preceding discussion applied to a machine equipped with a "C-ring", it is certainly possible to edit on a machine not so equipped. It is only necessary that two instructions be written, rather than one. The first instruction would transfer the "edit-control" word to the print area, and the second would cause the data field to combine with it.

DESCRIPTION OF ILLUSTRATIVE PROGRAM

There seem to be eight main steps involved in programming a job on this machine:

- 1 - Layout of the input/output data and constants on the "Storage Layout Chart".
- 2 - Preparation of FLOW CHART.
- 3 - Assigning "relative" addresses to the steps on the FLOW CHART.
- 4 - Partial enscribing of the steps onto the "Program Chart".
- 5 - Complete the "Storage Layout Chart".
- 6 - Complete the "Program Chart" by writing in the "actual" addresses.
- 7 - Punch the instruction cards.
- 8 - LOAD and check the program.

The particular problem chosen for this illustration was taken from the 407 INSTRUCTION COURSE, problem #13.

This is a simple listing job, involving two classes of totals, and the printing of stored indicative information under control of a code punch.

For the first step, refer to the "Storage Layout Chart". The following information was entered in blocks 11, 12, 13, 14 and 17:

INPUT -

- | | | |
|---------------------|-----------------|------------------|
| a) Department - | cols. 14 - 16 - | (Inter. Control) |
| b) Serial - | " 17 - 21 - | (Minor Control) |
| c) Deduction Code - | " 22, 23 - | |
| d) Description - | " 28 - 43 | |
| e) Amount - | " 77 - 80 | |

OUTPUT -

- | | | | |
|-----------------------|-----------------|---------|----------------|
| a) Department - | print positions | 23 - 25 | (Int. 1st CD.) |
| b) Serial - | " " | 28 - 32 | (Mi. 1st CD.) |
| c) Deduction Code - | " " | 36 - 37 | |
| d) Description - | " " | 41 - 56 | |
| e) Amount - | " " | 60 - 69 | |
| f) Deduction Period - | " " | 73 - 80 | |

WORKING STORAGE -

a) Department	1221-1223
b) Serial	1224-1230
c) Amount (Detail)	1231-1234
d) Amount (Minor)	1235-1242
e) Amount (Inter)	1243-1250

CONSTANTS -

a) One Time	1251-1260
b) Weekly	1261-1270
c) Monthly	1271-1200
d) z, zzz. dd**	1701-1712
e) zzzzz	1713-1717

The second step requires the development of a detailed flow chart, and the third step consists of assigning the "relative" address of each instruction (the small number at the lower right of each block on the Flow Chart).

This Flow Chart will be referred to, when the detailed description of machine operation for this program is given, following this section.

The fourth step can then be performed, that of writing the program on the "program chart". No actual instruction addresses can be given at this time, since they are not known.

In the right-hand margin of the "Program Chart" is recorded a running total of the number of characters in the program. The total (191) indicates that three blocks of storage are required for program storage.

Step 5 can then be executed, completing the layout of the Storage chart. The starting address for the program is arbitrarily chosen as 2200, and all of the instructions are laid out sequentially through blocks 22, 21 and 20.

The "actual" addresses of the instructions can then be read from the storage chart and written on the program chart to complete step 6.

All that remains is to punch the instruction cards and load the program.

The detailed description of the machine operation for this program will now be given. Reference will be made to the flow chart and program chart.

Step # 1 - GROUP COMPARE (G. C.)

This operation is actually a "Compare and Transfer" operation. The minor control field data of the preceeding card will be found at 1230, and that of the current card at 1125. At the completion of the step, the control field data of the current card will be found at 1230.

If the two control fields are equal, the P-ring will advance, and read the next instruction (address 2270). If not equal, the next instruction will be #19. This next instruction address is the I/C part of the first instruction, stored in the P-register. The contents of the P-register are transferred to the P-ring if the result of the comparison was "unequal".

Step #2 - ADD (AD)

The detail amount, columns 77-80, (address 1220) is added to the minor amount field, 1240.

Step #3 - EDIT (ED)

Punctuation is to be inserted in the money field, and the field is to be zero-suppressed to the left of the decimal point. This information is contained in the "Edit Control" word, 1712. For listing, this word is addressed as 1710, thereby eliminating the asterisks.

Step #4 - EDIT (ED)

The "Description Code" field is to be edited with zeros-to-the-left. The "Edit-Control" word for this operation is at 1232.

Steps #5 through #13 - DIGIT TEST (DT)

The "Deduction Code" (one column) is tested. If the column contains a 1, 5 or 7, then "Weekly" will transfer to the print area (Step #15). If the column contains a 2, 3, 4 or 6, then "One Time" will transfer to the Print Area. (Step #16). If the column contains an 8 or 9, then "Monthly" will transfer to the Print Area (Step #17).

Step #14 - ERROR STOP (LS)

The S in the OP Code causes the machine to stop, placing the machine under control of the console. The L in the OP Code indicates that a light on the console is to be operated. The O1 in the A and I/B parts of the instruction indicate the INDICATION LIGHT #1 is to be turned on so that the operator will be aware of the difficulty that caused the machine to stop, which in this case is a blank column in the "Description Code".

Step #18 - FEED AND PRINT (PF)

The Print Area is scanned to the printer and the Card Feed is activated so that the Input Area is loaded. The program will repeat, starting at Step #1, during the next compute cycle. When a Minor Break occurs, as detected in Step #1, the program will branch to Step #19.

Step #19 - GROUP COMPARE (GC)

This step is the same as Step #1, except that the comparison is now on the Intermediate Control Field. If the result of this test is equal, the program advances to Step #20, which is the first step of the minor total cycle routine. If the result of the test is unequal, the program will branch to Step #28 to set up a "minor-intermediate" routine.

Step #20 - ADD (AD)

The accumulated minor total is added to the INTERMEDIATE AMOUNT FIELD.

Step #21 - EDIT AND CLEAR (EC)

The "edit-control" field is now addressed as 1404, so as to pick up one asterisk to identify the minor total. The "A" field will not be re-generated, so this field will be effectively cleared.

Step #22 - PRINT (P-)

The Print Area will be scanned to the printer before advancing to the next instruction. The next instruction will normally be Step #27, as 2021 will normally appear in the I/B part of the instruction. The execution of Step #22 completes the MINOR ROUTINE.

Step #27 - EDIT (ED)

The "Serial" field is to be zero-suppressed. The "edit-control" field is 1717. This step is executed only on a "minor first card" cycle. Upon completion of this step, the program will jump to Step #2, the "detail" routine.

Step #28 - TRANSFER (TR)

When an "unequal" is detected on Step #19, it is an indication of an intermediate control break, and therefore a minor total cycle must be initiated, followed by an intermediate total cycle. Step #28 will set the I/B part of Step #22 to 2051, so that after the minor total is printed, an intermediate total cycle will follow.

Step #23 - EDIT AND CLEAR (EC)

The intermediate total field is edited and cleared.

Step #24 - PRINT (P-)

The intermediate total line is printed.

Step #25 - TRANSFER (TR)

It is necessary now that Step #22 be linked back to Step #27. This is accomplished by setting the I/B part of #22 to 2037.

Step #26 - EDIT (ED)

The "Department" field is to be zero-suppressed, and is therefore edited by the same control field as was used for editing "serial". This step will be executed only on an "intermediate first card" cycle.

This program was written for a "Second Option" machine, although a Standard or "First Option" machine could handle this problem nearly as well. Actually, the standard machine would require 39 instructions for a total of 225 positions of storage.

A fixed-size instruction word machine, such as was first proposed, would require nearly 400 positions of storage for instructions.

MISCELLANEOUS CONSIDERATIONS

At the time this report was started, it was felt that this machine must have the ability to handle non-standard multiple-punched card columns. Just recently, however, Product Planning has indicated that this ability is not required. Since a significant saving can be made by not providing this feature, it will not be included in this proposal.

This feature, however, can be added to the standard machine on either an optional or RPQ basis.

With this being the case, the punching and loading and encoding of instruction cards again becomes a problem.

Many acceptable solutions are possible, such as attaching a simple encoding device to the key punch, or providing an automatic routine on the accounting machine, or programming a conversion routine, or even providing slightly different form of "Storage Layout Chart" so that the proper coding for addresses can be read directly from it.

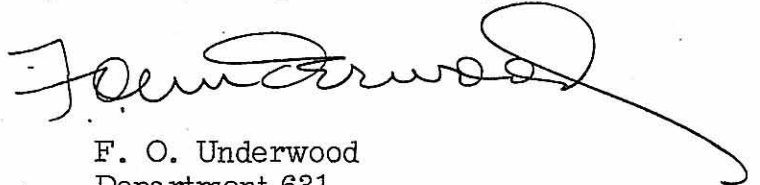
This report includes a block diagram of the new machine. It is much less detailed than before for two reasons: a) The storage addressing scheme is given in detail on other drawings in this report, and, b) the block labeled "DATA MODIFIER" now includes all functions of arithmetic, alpha comparing and print editing (both forward and reverse scans).

No detail work has yet been done on the "Data Modifier" specifically, but certain work done by Department 660 on magnetic core adders, multipliers and comparing units gives encouraging evidence that significant economies may be possible by an approach of this kind.

As mentioned before, it is also possible that the "DATA MODIFIER" functions can be stored in the core memory. It is obvious that a great deal of investigation is yet to be made in this area.

The output area is also subject to modification. We look to component development for guidance in these areas.

With the completion of this report, a programming effort will be initiated to resolve the functional problems, and guide the component and circuit development.



FOU/is
10-8-57

F. O. Underwood
Department 631

CLASS OF INSTRUCTION	CHARACTER POSITION										TYPICAL USE	
	11	10	9	8	7	6	5	4	3	2		1
I d b c d e					*	A-WORD GRP.-LOC.	B-WORD GRP.-LOC.	B-WORD GRP.-LOC.	B-WORD GRP.-LOC.	OP. CODE	OP. CODE	± A+B=B', A→B. SAME AS ABOVE, PLUS "JUMP". JUMP, NO-OP.
			*			A-WORD GRP.-LOC.	B-WORD GRP.-LOC.	I-WORD GRP.-LOC.		OP. CODE	OP. CODE	± A+B=C, A(MOD)B → C. SAME AS ABOVE, PLUS "JUMP".
						A-WORD GRP.-LOC.	B-WORD GRP.-LOC.	Y-I-WORD GRP.-LOC.		OP. CODE	OP. CODE	TEST - BRANCH ON "YES".
						A-WORD GRP.-LOC.	B-WORD GRP.-LOC.	Y-I-WORD GRP.-LOC.		OP. CODE	OP. CODE	SAME AS ABOVE, PLUS "JUMP ON "NO".
						A-WORD GRP.-LOC.	B-WORD GRP.-LOC.	Y-I-WORD GRP.-LOC.		OP. CODE	OP. CODE	MACHINE (FEED, PRINT, etc.). SAME AS ABOVE, PLUS "JUMP".
II a b						A-WORD GRP.-LOC.	B-WORD GRP.-LOC.			OP. CODE	OP. CODE	
						A-WORD GRP.-LOC.	B-WORD GRP.-LOC.			OP. CODE	OP. CODE	
III a b						A-WORD GRP.-LOC.	B-WORD GRP.-LOC.			OP. CODE	OP. CODE	
						A-WORD GRP.-LOC.	B-WORD GRP.-LOC.			OP. CODE	OP. CODE	

* - Has possible use with Optional Features, such as Shift, etc.
 † - MAY be used for Carriage Control.

- INSTRUCTION WORD FORMATS -
4740

13 SEPT. 1957

F. Q. Anderson

- SOME SPECIFIC EXAMPLES OF PRINT-EDITING -

EDIT CONTROL FIELD (e)
 . f . f . f . f . f . f . d . d . b . r . * . * . *

DATA FIELD (A)

EDITED DATA (C)

PRINTED DATA

4	3	6	7	8	.	2	1													
2	3	0	2	4																
0	0	6	0	0	5	0														
0	0	0	3	4	0	0														
0	0	0	0	5	3															
0	0	0	0	0	0															

f	4	3	6	7	8	.	2	1												
f	2	3	5	f	2	.	4	3												
f	6	.	f	.	5	0														
f	3	4	.	0	0															
f	.	5	3																	
f	.	0	0																	

\$	4	3	,	6	7	8	.	2	1											
\$	2	,	5	0	2	.	4	3												
\$	6	0	0	.	5	0														
\$	3	4	.	0	0															
\$.	5	3																	
\$.	0	0																	

MAJOR _____
INTER _____
MINOR _____

0	3	0	4	5	2	0														
0	3	0	4	5	2	0														
0	3	0	4	5	2	0														

f	3	,	f	4	5	.	2	0												
f	3	,	f	4	5	.	2	0												
f	3	,	f	4	5	.	2	0												

\$	3	,	0	4	5	.	2	0												
\$	3	,	0	4	5	.	2	0												
\$	3	,	0	4	5	.	2	0												

\$ p p . p p p . d d .

0	2	3	4	5	6	7														
0	2	0	4	5	6	7														
0	0	3	4	5	6	7														
0	0	0	4	5	6	7														
0	2	0	4	0	6	0														

\$	p	2	9	3	4	5	.	6	7											
\$	p	2	9	p	3	4	5	.	6	7										
\$	p	p	9	3	4	5	.	6	7											
\$	p	p	9	p	3	4	5	.	6	7										
\$	p	2	9	p	4	p	.	6	0											

\$	2	,	1	3	4	5	.	6	7											
\$	2	,	0	4	5	.	6	7												
\$	2	,	0	4	5	.	6	7												
\$	2	,	0	4	5	.	6	7												

b b b b b b d d .

0	4	2	0	3	0	0														
---	---	---	---	---	---	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--

4	2	0	3	.	0	0														
---	---	---	---	---	---	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--

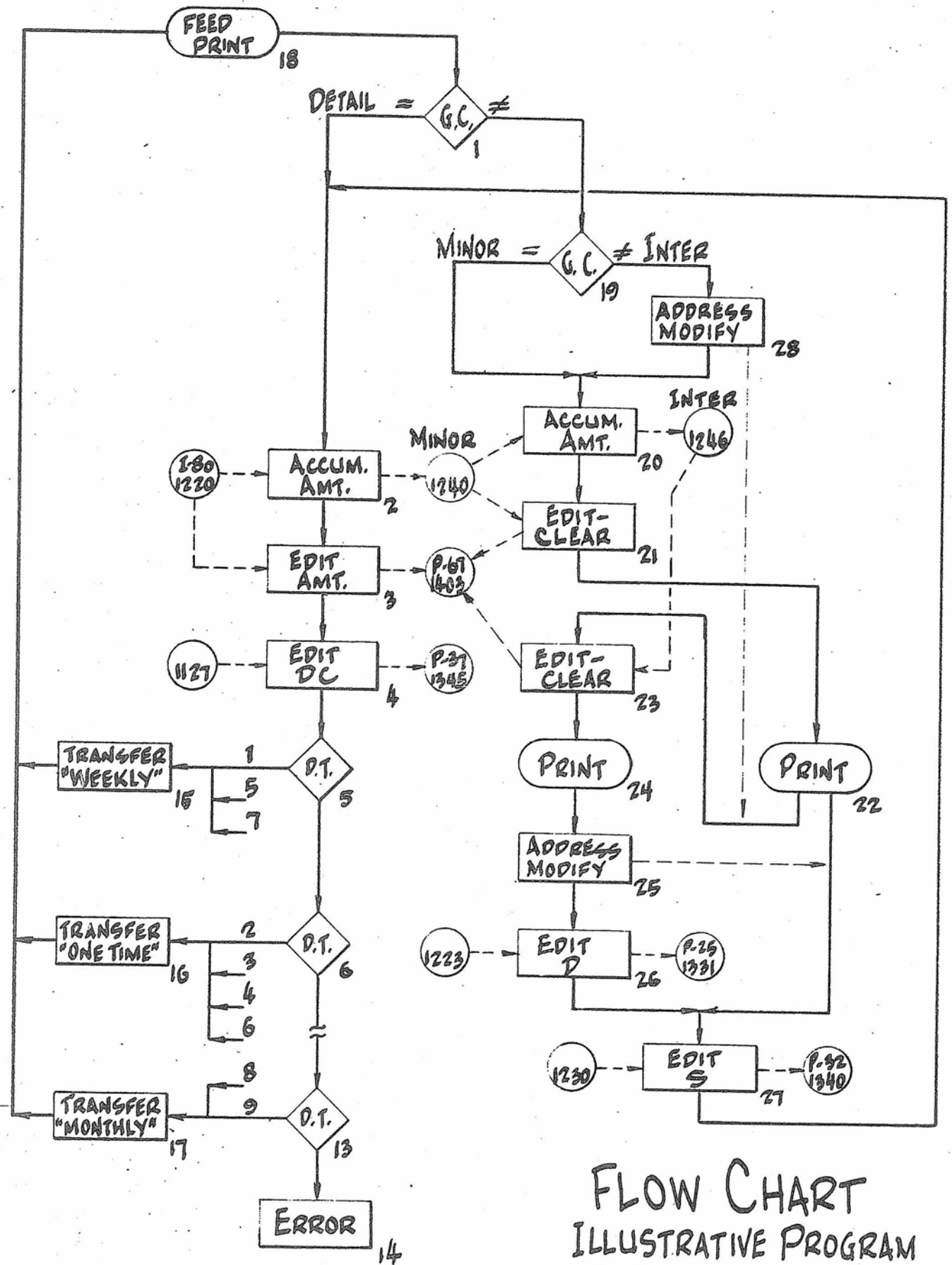
4	2	0	3	.	0	0														
---	---	---	---	---	---	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--

d d d - d d - d d d d .

0	1	6	1	0	2	0	6	8												
4	1	2	5	7																

0	1	6	-	1	0	-	2	0	6	8										
0	4	-	1	2	-	5	7													

0	1	6	-	1	0	-	2	0	6	8										
0	4	-	1	2	-	5	7													



FLOW CHART
ILLUSTRATIVE PROGRAM
4740 - 30 SEPT. 1957

PROGRAM CHART

NOTE	T/D	I	I/C	A	I/B	OP.	ADDRESS		
							ACTUAL	REL.	

4740 - 16 SEPT. 1957 Fou

PROGRAM CHART

NOTE	1/0	I	1/C		A		1/B		OP.	ADDRESS			
										ACTUAL	REL.		
1/C=19 COMPARE MINOR AMT. TO W/S EDIT AMT. EDIT DC					11	25	12	30	G.C.		1	8	
					12	20	12	40	A D		2	14	
				L14	03	12	20	17	10	E D		3	
				L13	43	11	27	12	32	E D		4	30
1/B=15 TEST C FOR		1			L1	11	27		D T		5		
	16	2			L2	11	27		D T		6		
	16	3			L3	11	27		D T		7		
	16	4			L4	11	27		D T		8		
	15	5			L5	11	27		D T		9		
	16	6			L6	11	27		D T		10		
	15	7			L7	11	27		D T		11		
	17	8			L8	11	27		D T		12		
	17	9			L9	11	27		D T		13	93	
					L0	1	0	1	L S		14	99	
1/C=18 "WEEKLY" TO PRINT					L12	66	14	20	T R		15		
1/C=18 "ONE TIME" "					L12	56	14	20	T R		16	115	
					L12	76	14	20	T R		17	121	
1/B=1 PRINT-FEED									P F		18	125	
1/C=28 COMPARE INTER ACCUM. MI. TO I.					L11	20	12	23	G C		19	133	
					L12	40	12	46	A D		20	139	
1/B=X EDIT MINOR PRINT EDIT INTER. PRINT RESET X TO 27					L14	04	12	40	17	11	E C	21	147
											P -	22	151
					L14	05	12	46	17	12	E C	23	159
											P -	24	161
											T R	25	167
1/C=20 EDIT D EDIT S SET X TO 23					L13	31	12	23	17	17	E D	26	
					L13	40	12	30	17	17	E D	27	
					L						T R	28	191

ILLUSTRATIVE PROGRAM
30 SEPT. 1957

PROGRAM CHART

NOTE	T/O	I	I/C		A		I/B		OP.	ADDRESS			
										ACTUAL	REL.		
J=19 COMPARE MINOR			L 21	03	11	25	12	30	G. C.	22	00	1	8
AMT. TO W/S					L 12	20	12	40	A D	22	70	2	14
EDIT AMT.			L 14	03	12	20	17	10	E D	22	62	3	
EDIT DC			L 13	43	11	27	12	32	E D	22	52	4	30
I/B=15 TEST C FOR 1			L 1	11	27	21	35	D T	22	42	5		
16 2			L 2	11	27	21	25	D T	22	33	6		
16 3			L 3	11	27	21	25	D T	22	24	7		
16 4			L 4	11	27	21	25	D T	22	15	8		
15 5			L 5	11	27	21	35	D T	22	06	9		
16 6			L 6	11	27	21	25	D T	21	77	10		
15 7			L 7	11	27	21	35	D T	21	70	11		
17 8			L 8	11	27	21	15	D T	21	61	12		
17 9			L 9	11	27	21	15	D T	21	52	13		93
ERROR STOP			L 0	1	0	1	LS		21	43	14		99
3/c=18 "WEEKLY" TO PRINT			L 21	07	12	66	14	20	T R	21	35	15	
3/c=18 "ONE TIME" "			L 21	07	12	56	14	20	T R	21	25	16	115
"MONTHLY" "					L 12	76	14	20	T R	21	15	17	121
3=1 PRINT-FEED							L 22	00	P F	21	07	18	125
I/C=28 COMPARE INTER			L 20	11	11	20	12	23	G C	21	03	19	133
ACCUM. MI. TO I.					L 12	40	12	46	A D	20	73	20	139
EDIT MINOR			L 14	04	12	40	17	11	E C	20	65	21	147
I/B=X PRINT							L 20	21	P -	20	55	22	151
EDIT INTER.			L 14	05	12	46	17	12	E C	20	51	23	159
PRINT									P -	20	41	24	161
RESET X TO 27					L 17	77	20	53	T R	20	37	25	167
EDIT D			L 13	31	12	23	17	17	E D	20	31	26	
EDIT S			L 13	40	12	30	17	17	E D	20	21	27	
I/C=20 SET X TO 23			L 20	73	20	01	20	53	T R	20	11	28	191

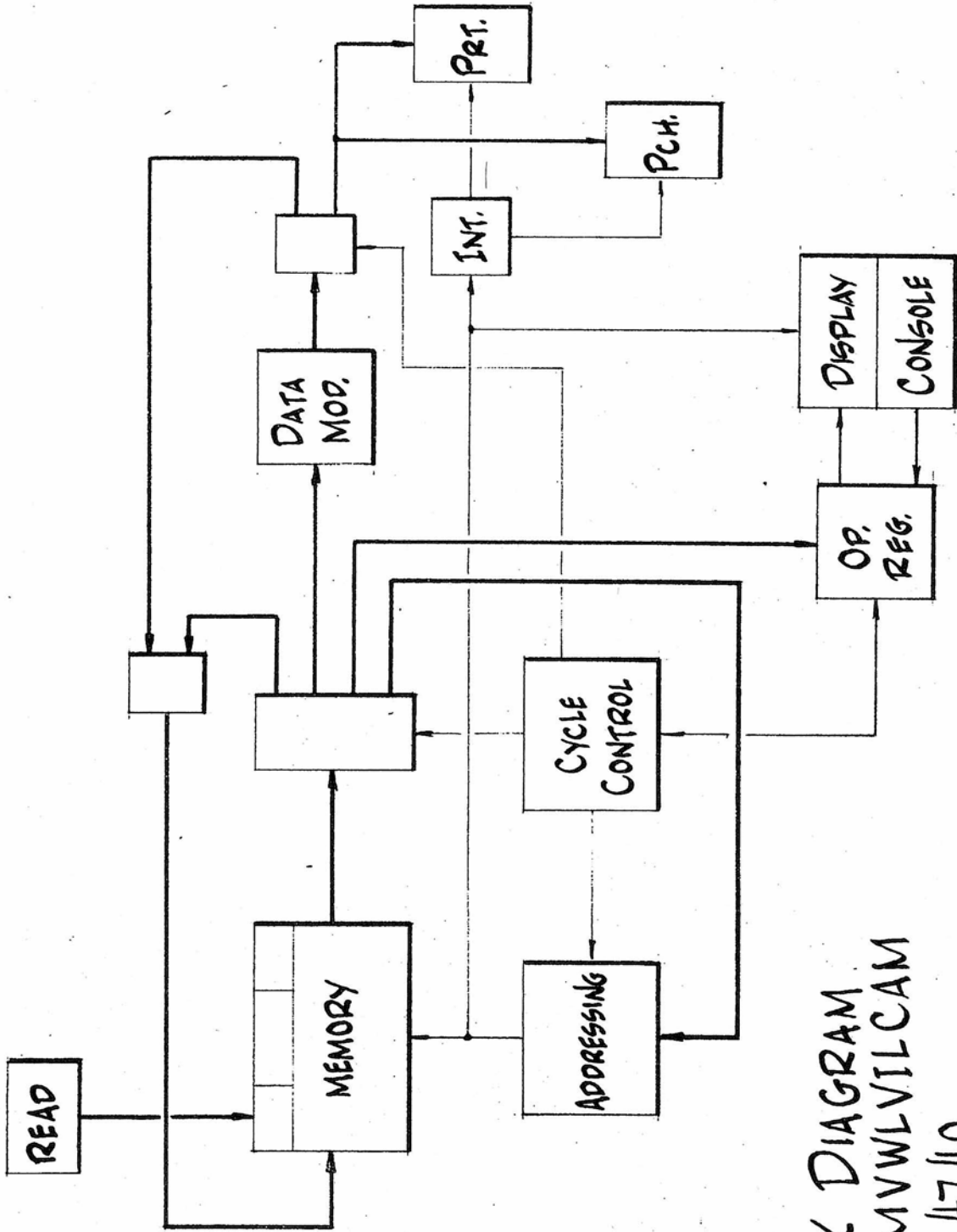
CONSTANTS 2051 AT 2001
 2021 AT 1777
 7's AT 1717
 7, 7, 7, 7-dd** AT 1712
 dd AT 1732
 "ONE TIME" AT 1256
 "WEEKLY" AT 1266
 "MONTHLY" AT 1276

ILLUSTRATIVE PROGRAM
 30 SEPT. 1957

STORAGE LAYOUT CHART

	P	S	MI	DC	Description	AMT.	D	S	MI	DD	AMT.	MI	AMT.	ONE TIME	WEEKLY	MONTHLY
17																
14					DESCRIPTION	AMT. * **										
16																
1/6 COLUMNS -																
STORAGE LOCATIONS -																
17	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
21	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
22	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ILLUSTRATIVE PROGRAM																
30 SEPT. 1957																
<i>Tom</i>																

- 4740 - 16 Sept. 1957 Jps

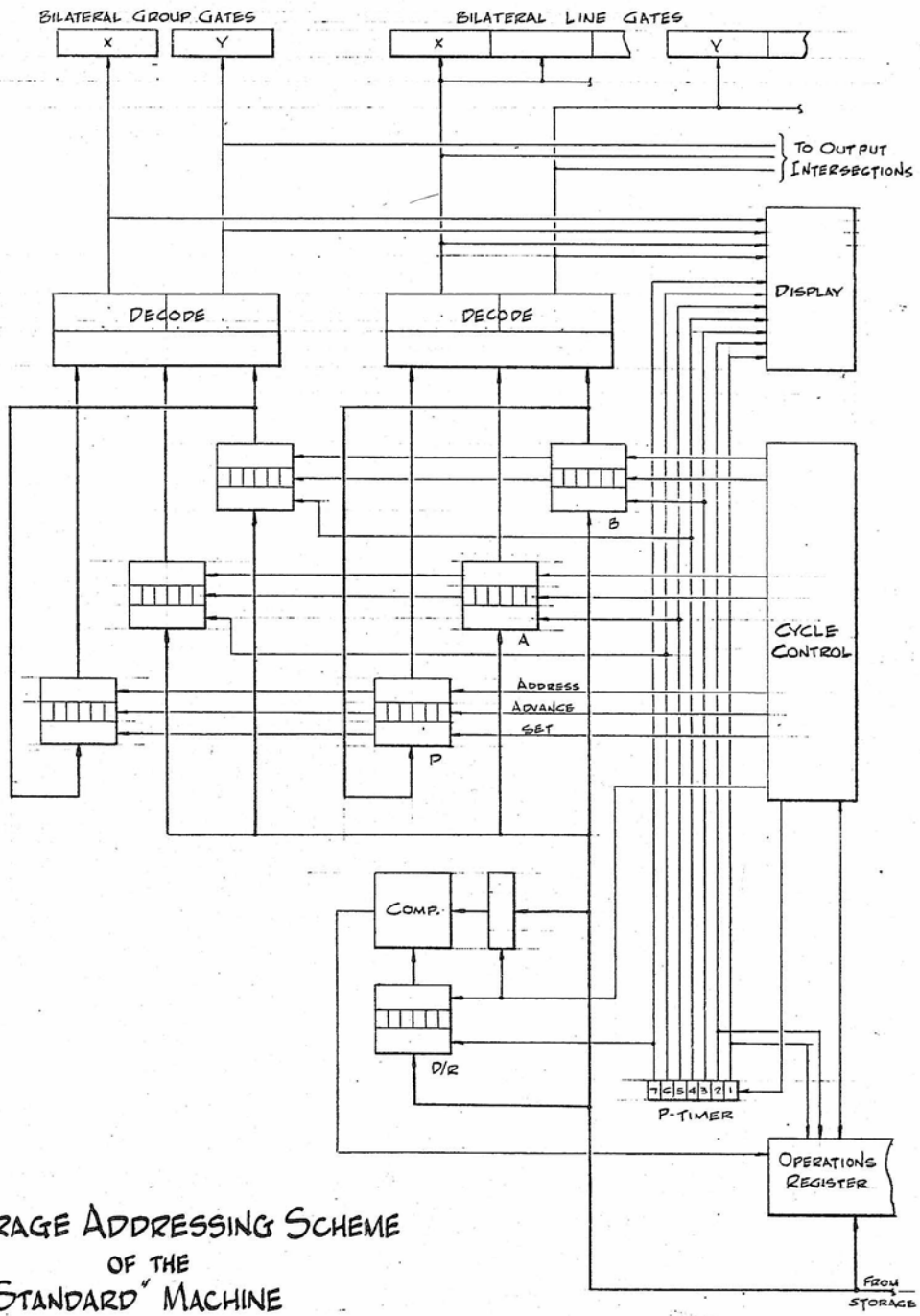


BLOCK DIAGRAM
SPMCMVWLVLICAM

4740

3 OCT. 1957

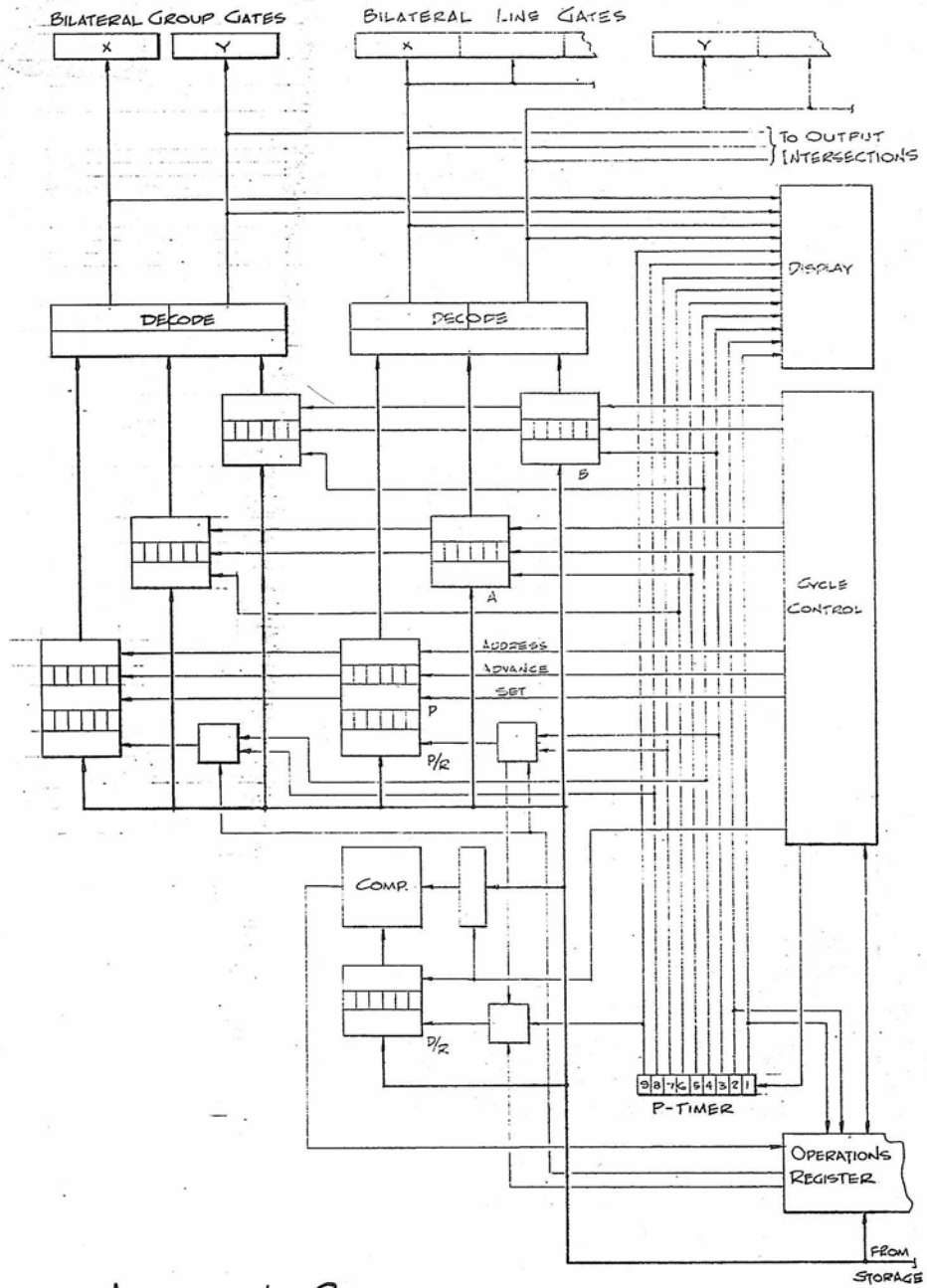
File



STORAGE ADDRESSING SCHEME
 OF THE
 "STANDARD" MACHINE
 4740

10 SEPT. 1957

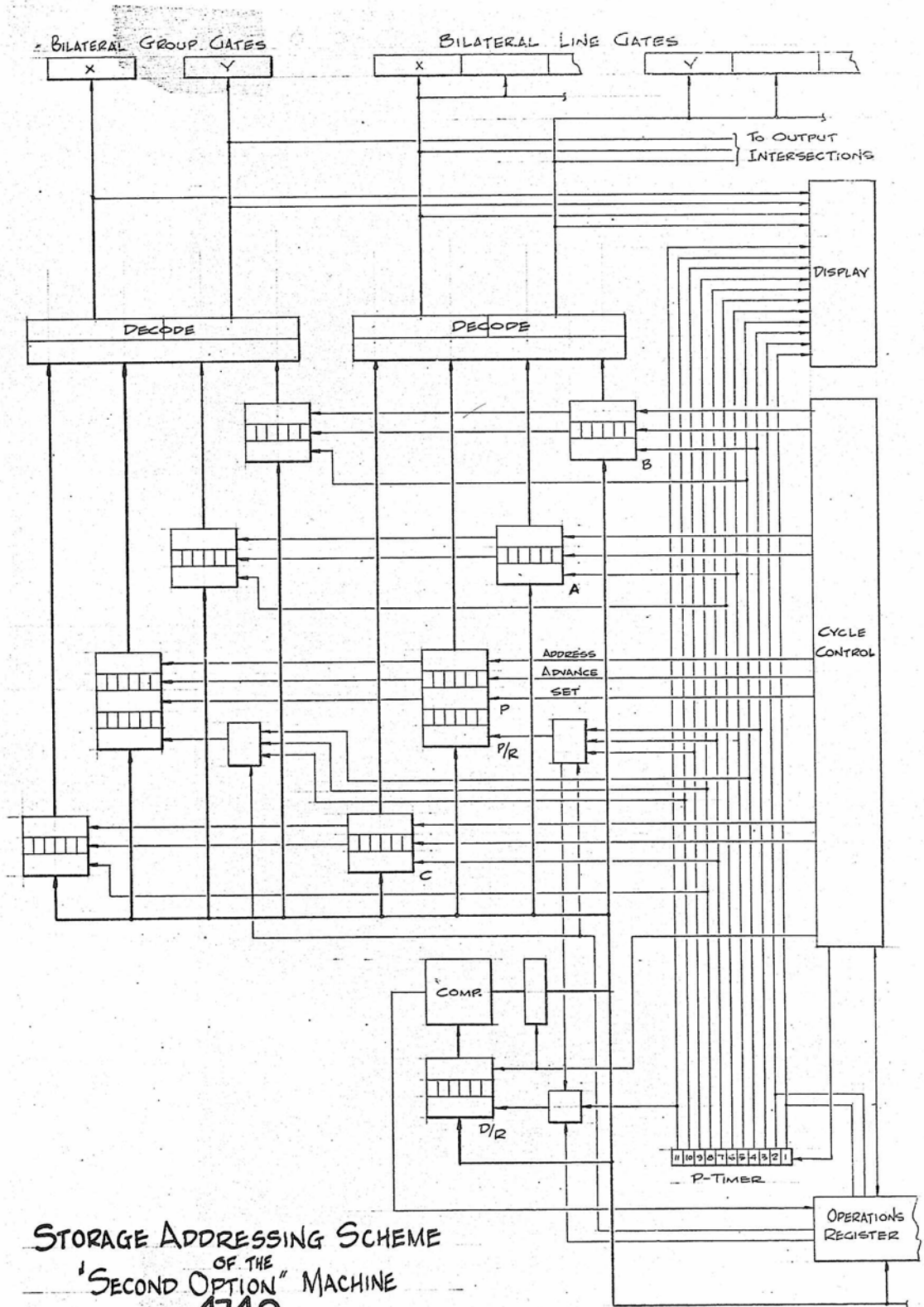
Forward



STORAGE ADDRESSING SCHEME
 OF THE
 "FIRST-OPTION" MACHINE
 4740

11 SEPT. 1957

John Wood



STORAGE ADDRESSING SCHEME
 OF THE
 'SECOND OPTION' MACHINE
 4740
 11 SEPT. 1957

J. Underwood