GENERAL

Maniac II is a high speed, general purpose, digital computer, with a random access, self checking, electrostatic storage of 12,288 48-bit words. It is a binary, single address, parallel computer. It operates in fixed or floating point and has automatic address modification, by means of B registers (index registers).

The arithmetic unit consists of three shifting registers, U, R, and S, and an adder, +. The Universal Register, U, holds the important operands and results. It is the accumulator and receives the result of addition or subtraction. It contains the multiplicand, the high order product, the high order dividend, the quotient, and the argument and result of the square root order. The Remainder Register, R, holds the low order product and dividend, the remainder, and the extractor, and can be used for an extra-fast-access temporary storage location. The Storage Register, S, communicates with the electrostatic storage, the adder, the R register, the input-output equipment, and the memory address buses.

The Instruction Register, I, receives instructions from the storage. Its Order Part, O, communicates with the decoding circuits; its B part selects an index register; and its Address Part, A, communicates with the B Adder, B+. The B Registers, B1, B2, and B3, contain indices for address modification, which they gate into B+ whenever selected. B+ communicates with the memory address buses. The Control Counter, CC, also communicates with the address buses, and governs the fetching of instructions. It can be set by B+ to effect transfers of control.

Input is via magnetic tape, paper tape, and a typewriter. (The last provides a written record of all manual changes.) Output is to magnetic tape, paper tape, the typewriter, and a fast line-printer. The input and output units are controlled by special instructions and/or by manual switches.

The 48-bit words in which information is stored are operands when brought into the arithmetic unit, and constitute pairs of instructions when brought into the Instruction Register. The fetching of instructions is governed by CC, which counts by half words unless set by a Transfer Control instruction, or unless caused to make a double count by certain special instructions. The fetching of operands (or the storing of results) is governed by the address part of the instruction involved, or by that address as modified by the contents of a B Register. In the case of certain orders which require several operands, stored at consecutive addresses, the contents of CC (the Control Counter) are dumped temporarily into the Pathfinder Register, PF, and CC is used to compute the required addresses. The primary function of PF is to receive the contents of CC whenever the CC is about to be set by a Transfer Control instruction, and to make this information available to the arithmetic unit.

STORAGE

The internal storage of Maniac II consists of two barrier grid cathode ray tubes per stage, with either 3072 or 6144 bits per tube[1].

In addition to the normal 48 pairs of tubes, there is a 49th pair of tubes which contains a parity check bit for each word in the storage. This bit is set whenever a word is written into the storage, and it is checked on each regeneration and on each fetch. If a bit should be dropped or picked up, the Maniac would stop, displaying the address of the failure.

The regeneration time per word is about 8 microseconds. It takes about 50 milliseconds to regenerate the full memory, or 25 milliseconds for half the memory. The consultation ratios[2] are at least 100 for the full memory and 500 for the half memory.

The memory cycle of the electrostatic memory is 8 microseconds. Under some circumstances, some of this time is covered by other useful work, such as clearing registers to zero.

The fourteen Sense Lights, which will be discussed later, constitute storage positions for single bits of information (e.g., for combinations of yes-no decisions), which can be stored and read by the operator as well as by the Maniac.

There are two magnetic tape units, which can be used as external storage. The Maniac can transfer word blocks of arbitrary length from

---

[1]The choice is made by a Full Memory-Half Memory switch.

[2]The consultation ratio is the number of consultations allowed between regenerations.

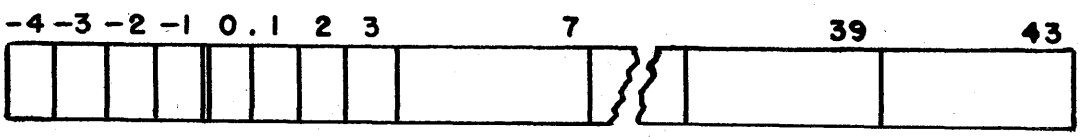
the internal storage to the tapes, and vice versa, at a rate of 200 words per second. The parity check bit for each word is recorded on the tape and checked on Tape Call. The word sum of each block is recorded at the end of the block, primarily for block identification.

INFORMATION

A word can be an instruction, an operand, or both, according to the use which is made of it. For example, a word ordering the multiplication of the contents of U by the contents of, say, memory address 0100 would be an instruction, while the word at address 0100 would be an operand.

More specifically, an instructional word consists of two half-word instructions, each having six tetrads of four bits each. The first two tetrads of an instruction constitute an order, according to the vocabulary given below. The last four tetrads (sixteen bits) furnish an address (or some other number relevant to the particular order) in the following way. The first two bits select a B register (B_0, the zeroth B register, is a mythical register defined as containing zero at all times). The contents of the last fourteen bits are then added to the contents of the selected B register to furnish the complete address. (Note that some instructions, such as (R) to U, are completely specified by the order tetrads; for these, the last four tetrads are ignored.)

An operand is by nature simply a collection of 48 bits of information, which can be interpreted and modified in any finite way by the available vocabulary. In the majority of cases, however, it is treated as a number. The bits of a number, or the stages of an arithmetic register holding a number, are designated as shown in Fig. I.



The binary point is between bits zero and one. Stages one through 43 hold a positive fraction, $\underline{x}$, whose range is $0 \leq x \leq 1 - 2^{-43}$. Stage zero holds a sign for this fraction (0 for plus, 1 for minus).

Stages -3 through -1 hold a positive integer, $\underline{e}$, and stage -4 holds a sign for this integer. This signed integer, $\underline{e}$, is the exponent of the Maniac's floating point base, which is $2^{16} = 65,536$.

Thus a full number, represented by an exponent, $\underline{e}$, and a fraction, $\underline{x}$, is

$$N = 2^{16e}x.$$

The range, for a single word, non-zero number, N, is

$$2^{-155} \leq N < 2^{112},$$

or, approximately,

$$2 \cdot 10^{-47} < N < 5 \cdot 10^{33}.$$

The Maniac's large base permits a considerable increase in the speed of floating point arithmetic. Although such a large base implies the possibility of many lead zeros, the large word size of 48 bits guarantees adequate significance.

A number N = (e,x), for which the exponent e = 0, is equal to the fraction, $\underline{x}$, and may be considered fully equivalent to a fixed point number. The fact that floating and fixed point numbers have identical fraction bits allows a considerable saving in computer hardware. The saving is possible because, in many instances, the Maniac need not distinguish between fixed point operations and floating point operations which operate on numbers having zero exponents.

CONTROL

Control of the Maniac, for normal operation, is effected by a stored program. A problem to be solved must first be put in terms of the Maniac's vocabulary. The appropriate instructions must then be coded, and the coded instructions put into the internal storage, along with the necessary coded or numerical input data. (A large part of this work can be done by the Maniac, using translation and assembly routines.) The control is then sent to the first instruction.

After executing any instruction (other than Stop), the Maniac fetches another instruction into I (the Instruction Register) from a location specified by CC. Unless CC is specially set to a new address, it counts by half-words and causes the fetching of sequentially stored instructions. (Exceptions to this can occur on Sense and on Count-and-Compare, where CC may be made to count twice before the next instruction is fetched.) The sequencing of instructions is the same under automatic and manual operation, provided the manual operation consists merely in stepping through the program.

CC can be specially set to a new address in three ways: by a Transfer Control instruction; by a breakpoint transfer; or manually, by using the Control Counter Switches.

A Transfer Control instruction (for which the conditions, if any, obtain) involves three steps. First, CC makes an ordinary half-word count, giving it the address of the next instruction in sequence, i.e. the instruction which would be fetched next if the transfer did not take place. Second, this address is placed in the Pathfinder Register, PF, where it is available to the arithmetic unit (in particular, available

for return from subroutines, etc.). Third, the Control Counter is set to the address contained in the Transfer Control instruction (or to that address modified by the contents of a B Register) and a new instruction is fetched from that address.

Breakpoint transfers involve the interaction of special switches, set by the operator, with special tags (real or virtual) on instructions in the storage. There are two types of breakpoint, called red and purple. A tag for a red breakpoint is real, and it is a zero placed in the first bit position of the first order tetrad. Thus the order AB, for example, becomes 2B if tagged with a red breakpoint, since the hexadecimal digit A becomes 2 when its first bit is set to zero. The tag for a purple breakpoint is virtual; only one instruction at a time can be tagged with a purple breakpoint, and the tagging is done by setting the half-word storage location of the instruction on a set of Purple Breakpoint Address Switches.

There are two three-position Breakpoint Switches (one for each color of breakpoint), the three positions being Off, Stop and Transfer. If a switch is in the Off position, then all tags of that kind are completely ignored. If a switch is in the Stop position, then the Maniac stops after performing any instruction with the corresponding kind of tag, without fetching the next instruction. If a switch is in the Transfer position, then the Maniac effectively inserts after any appropriately tagged instruction, an Unconditional Transfer Control instruction, with an effective address equal to the address set on the CC Switches (see below). In other words, after performing a tagged order, the Maniac sets CC to the address on the CC Switches, leaving

stored in PF the usual record of where it was about to go. (Exception: if a Transfer Control, Sense, or Count-and-Compare instruction[3] has a tag corresponding to a switch in the Transfer position, the Maniac acts as though the switch were in the Stop position.)

Manual setting of the Control Counter is effected by pushing a special CC Set Switch on the control panel. CC sets to the address on the CC Switches. For this manual setting, the Manual-Automatic Switch must be on Manual; if it is on Automatic, then the CC Set Switch is inoperative.

---

[3] This is also true for either of the two Substitute Right Address instructions, if the substitute instruction is on the left, and the substitution is into the right side of the same word.

MANUAL OPERATION

The facilities for manual operation of the Maniac are on the Operator's Console. The console consists of a desk, facing a large panel, and almost surrounded by the input-output equipment. The panel contains rows of lights displaying the contents of the various registers, rows of switches for setting certain registers, and an assortment of display lights and special switches for a variety of purposes. S, R, and U are displayed at the top. On the right side are I, B+, the three B Registers, PF, and the Parity Check Register. On the left side are CC, with the CC Switches and the Purple Breakpoint Switches, the Sense Lights, and the Sense Switches. In the middle are the Red Breakpoint Switch, the Exponent Spill Lights and Allow NES Switch,[4] and the Overflow Light. On the right, at the bottom, are the Manual-Automatic Switch, and the Fetch and Perform Buttons.

The operator can step through the normal sequence of instructions with the Manual-Automatic switch on Manual. This can be done by using alternately the two pushbuttons called Fetch and Perform. However, the operator can also use these buttons to step through the sequence with omissions and/or repetitions of instructions. The Fetch button causes the next Instruction in numerical sequence (not normal program sequence) to be fetched into I; no instruction is performed. The Perform button causes the instruction already in I to be performed; that instruction remains in I.

A Slow Automatic button allows the operator to step through the normal sequence of instructions at a rate of about 20 instructions per

---

[4] Negative Exponent Spill

second, for as long as the button is depressed. This button is operative only on Manual.

When the Manual-Automatic Switch is on Manual, it is also possible, by using the Lead Switch, to enter words into the memory without having them called in by a program. The Load Switch sets CC to the address on the Control Counter Switches and then reads full words from the paper tape in the Photoelectric Reader into sequential memory positions, starting with the address contained in CC. This process continues until a stop-code character is encountered on the paper tape, at which time the Maniac stops, resetting CC to the address of the first word loaded.

When the Manual-Automatic Switch is on Manual, the input-output typewriter (Flexowriter) can be enabled. This is done by a three position switch on the Flexowriter; the three positions are Neutral, Enter I, and Enter S. (Note: the Maniac will not run on Automatic unless this three position switch is on Neutral.) If the switch is in the Enter I position, then each Flexowriter key struck causes the corresponding hexadecimal character to be shifted from the right into the six tetrad instruction register, I. In this manner a full instruction can be entered, and it can then be performed via the Perform button. If the typewriter switch is on Enter S, then typed characters will be shifted from the right into S, whence they can be sent to U or the memory by appropriate instructions. A typed record of all such characters will be made on the Flexowriter, and a Punch On switch will allow the simultaneous production of a paper tape record.

MANUAL INTERVENTION

The fourteen Sense Lights were mentioned in the section on Storage. These lights are single-bit storages. They can be set independently to one (on) or zero (off) by the operator, while on Manual or on Automatic. They can of course be read by the operator, from their on or off status. They can also be set either way by the Maniac singly or in any combination. Lastly, they can be tested by the Maniac, using an instruction which asks whether or not a given combination of lights is all ones. By means of these lights, then, up to fourteen bits of informations at a time can be exchanged between the Maniac and the operator, without interrupting the calculation.

STOPS

There are three normal methods of stopping the Maniac. Setting the Manual-Automatic switch to Manual stops the Maniac at the completion of the instruction being performed. Breakpoint stops have already been discussed. Thirdly, the Maniac stops when a Stop instruction is in I; a Stop instruction is, by definition, any instruction with order tetrads not defined in the vocabulary.

In addition to these normal stops, there are several stops designed to catch programming errors, operator errors, and/or machine malfunctions. Except for the Parity Check Stop, one may continue the program, after a stop, by switching to Manual, fetching the next instruction, returning to Automatic, and pushing the Perform button. In the case of certain stops due to input-output equipment's not being ready, the Maniac will continue automatically as soon as the selected piece of equipment is made ready.

The special stops are:

Read Stop. When there is no paper tape in the Photo Reader, then a Read instruction causes a stop.

Fast Punch Stop. When there is no paper tape in the Fast Punch, then a Punch instruction causes a stop.

Magnetic Tape Stop. When the Magnetic Tape Unit is not in Ready position, then a Magnetic Tape instruction causes a stop.

Flexowriter Stop. When the Punch switch on the Flexo is on, and there is no paper tape, then a Flexoprint instruction causes a stop.

Fast Print Stop. When the Fast Printer is not ready, due to printer failure or to a previous improper print matrix (more than one character having been requested for the same column), then a Fast Print instruction causes a stop.

Positive Exponent Spill Stop. Positive exponent spill can occur on all basic arithmetic operations. When it does, the Positive Exponent Spill Light goes on and the Maniac stops. The light can be turned off by a special reset button, and operation can be continued as described above.

Negative Exponent Spill Stop. Negative exponent spill can occur on normalization, multiplication, or division. When it does, the Negative Exponent Spill Light goes on and the Maniac stops, unless the Allow Negative Exponent Spill switch has been turned on. In that case, the Maniac interprets numbers with spilled negative exponents as zeros, with exponent -7, and proceeds accordingly, without interruption or special display. In the stop case, the operator can continue as for positive exponent spill.

Fixed Point Division Stop. A fixed point division which would yield a rounded quotient of magnitude greater than or equal to one causes a stop.

Square Root Stop. When the number in U is negative, then a Square Root instruction causes a stop; the number in U is unchanged.

Illegal Address Stop. When a word is called from the internal storage from a non-existent address (3000-3FFF for Full Memory or 1800-3FFF for Half Memory), then the Maniac stops. Note that this address may be the result of adding an index to a basic instruction address.

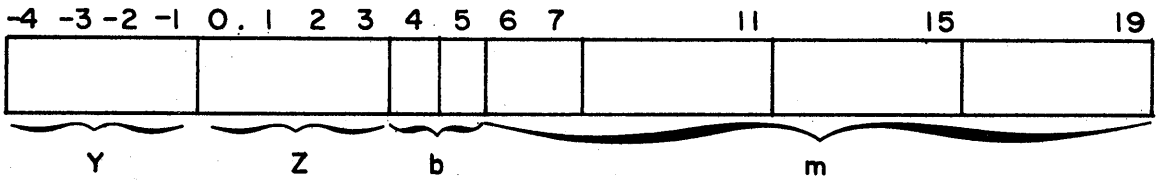
Parity Check Stop. If any word regenerated in the internal storage, fetched from the internal storage, or called from magnetic tape, should not have the parity specified by its associated parity check bit, the Maniac would stop, displaying a Tilt signal and the storage address of the offending word. Regeneration would of course continue, and if, for example, an entire memory unit had failed, the displayed address might change as often as every eight microseconds.

VOCABULARY

Any vocabulary list necessarily involves a compromise between brevity and completeness. This is particularly true as regards secondary changes in register contents, changes not of interest in straightforward programming. It is also true in regard to stops which may occur during the performance of an instruction, such as those due to exponent spill. Since a separate T-7 publication[4] describes in detail the secondary register changes and since a preceding section of this report describes the various steps, the following vocabulary list leans toward brevity.

Before giving this list, however, it will be helpful to define some abbreviated notation.

The instruction notation YZ b m refers as follows to the 24 bits which constitute an instruction:



When b and m are not relevant for a particular instruction, they are replaced by dashes.

Bits -4 through 3 are assigned to the two order tetrads. Bits 4 and 5, when relevant, govern the selection of a B register.

---

[4] MANIAC II: REGISTER CONTENTS UPON LEGAL COMPLETION OF ORDERS, April 18, 1956.

The letter X is used to represent the sum of the fourteen bit number m and the fourteen bit number contained in the bth index register (with the convention that the (mythical) zeroth index register always contains zero). X is the address or other relevant number referred to in the section on Information.

The letters U, R, and S stand for the Universal, Remainder, and Storage Registers, respectively. When necessary or helpful to distinguish stages 0 through 43 of a register from the entire register (-4 through 43), the notation U', R', or S' will be used. MU, MR, and MS will be used to denote stages 1 through 43 of U, R, and S. (These are the stages which hold the magnitude of the fraction part of a number.) The notation UR stands for the effective double length register in which stages 1 through 43 of R are taken to be an extension to the right of U. PF stands for the Pathfinder Register.

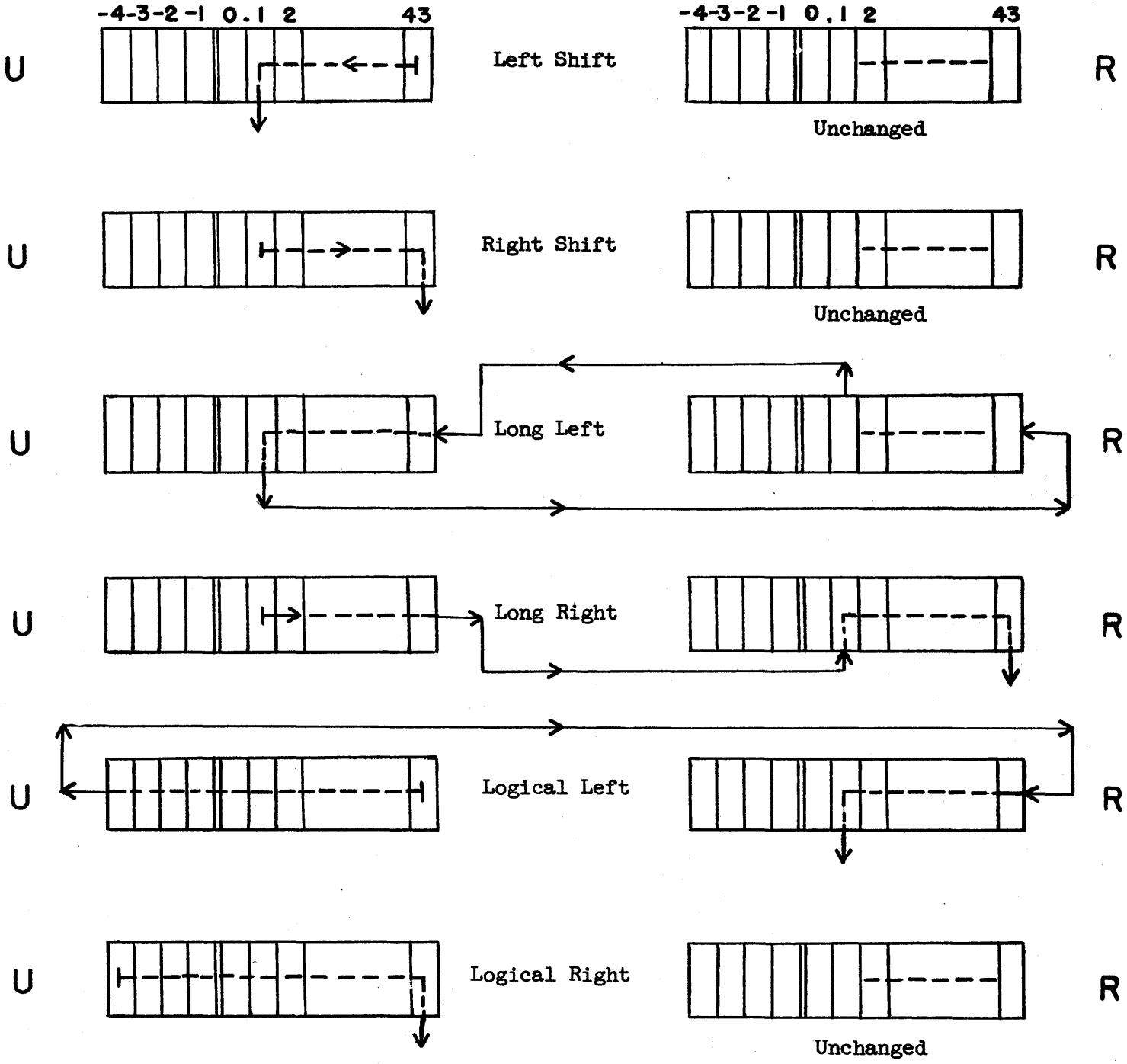
Parentheses will be used to denote "the contents of" or "the information stored at". For example, (U) will mean the word in the Universal Register, and (5-19U) will mean the contents of stages 5 through 19 of U. An arrow (→) will be used for "replaces" or "replace" (or occasionally for "to"). Thus (U)→(X) means that the contents of U replace the contents of X, i.e. store (U) at address X.

S_o, referred to in orders B5 through B7, denotes the double address positions 6-19 and 30-43. It must be noted, however, that the Substitute Address instructions C4 through C7 substitute the contents of stages (3 and 6-19), for the left address, and stages (27 and 30-43), for the right address. This is because of the need to substitute half-word addresses

into the Transfer instructions C8 through CF.

The reader should refer to the section on Control, subsection on Transfer Control instructions, for the details of the Transfer instructions. Note that the address in PF, after a Transfer, is the normal return address for a basic linkage, and address substitution from PF is the normal method of setting an exit. This exit setting will probably be the only use made of the substitution instructions in straightforward programming of mathematical problems. It should be added that the programmer, coding in conventional descriptive form, can ignore the left-right, half-instruction difficulties, which will be taken care of by the assembly routine.

Since a complete verbal description of Shift instructions is always long-winded, the following diagram may be found more useful than the descriptions of the Shift instructions given in the Vocabulary list.


-4-3-2-1 0.1 2 43
U
Left Shift
-4-3-2-1 0.1 2 43
R
Unchanged
U
Right Shift
R
Unchanged
U
Long Left
R
U
Long Right
R
U
Logical Left
R
U
Logical Right
R
Unchanged

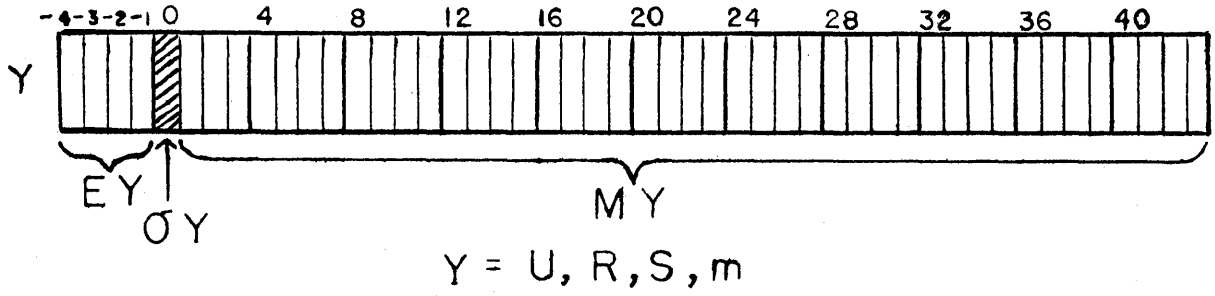
MANIAC II REGISTER CONTENTS UPON LEGAL COMPLETION OF ORDERS

Sometimes in programming, it is helpful to know the final contents of other registers than the one primarily involved in any given order. The accompanying table supplies this information (i.e. the register contents upon completion of an order) for all orders of MANIAC II except input and output. The contents of the registers upon machine stops occurring as a result of illegal operations on numbers have not been tabulated since these are not of great use in coding (as opposed to debugging), and since they may be very complicated. It is intended that this table should supplement, rather than replace, the vocabulary description.

E, σ, M

In most orders, the change in the I, I*, and B registers is either uninteresting or obvious, or both; hence, only U, R, and S have been tabulated. Each of these registers is broken up into three parts:

E	The exponent bits (-4 to -1)
σ	The sign bit (0)
M	The unsigned fraction, or magnitude bits (1 - 43).



Subdivisions of a binary number considered as floating-point

The contents of the registers upon completion of an order is for the most part described in terms of the contents at this same time of other registers, or of the appropriately B-modified memory location addressed by the order. For this latter, the symbol m is used. Thus, if the entry EU occurs in the E-column of R, one should read this as: "The contents of the exponent-bits of R at the completion of this order is equal to the contents of the exponent bits of U at this time." Or, if in the M-column of the S-register is found the notation Mm, this says: "The contents of the magnitude-bits of S upon completion of this order will be identical to the final contents of the magnitude bits of the appropriate B-modified memory location addressed by this order."

— It frequently happens that the contents of a part of a register is unchanged by the performance of an order. This state of affairs is indicated by a dash (—) in the appropriate spot in the table.

Z' Occasionally, reference must be made to the contents of some portion of a register prior to the performance of the order. A prime following the appropriate symbol signifies this. Thus, EU' designates the original contents of the exponent-bits of U.

X In some columns of the table, an X will be found. This implies that a variety of possibilities exists, depending on the numbers being compounded by the order. Although the results are perfectly predictable (obviously!), they have been deemed so esoteric as to be of minor interest for normal programming.

If detailed information is desired, one should consult the engineers, or Roger Lazarus.

$\overline{A}$ The performance of some orders leaves the ones-complement, or reflection, of a number in some portion of a register. A bar over the relevant symbol denotes this. Thus,

if $\sigma m = 1$

$\overline{\sigma m} = 0$

or if MR = 111111111.....111

$\overline{MR}$ = 000000000.....000

0, 1 The character 0 implies that every bit associated with the column in which it is found is a zero. Similarly 1 implies all ones.

In a few cases, an actual binary number is written out, and elsewhere, a signed decimal number has been used to indicate the contents of a portion of a register.

{ } Curly brackets around a pair of symbols imply that, under various circumstances, one or the other of the quantities indicated may occur.

M* In the addition and subtraction orders, a symbol M* is used, which indicates either M or the reflection of M, depending on whether the order is addition or subtraction, and on whether the signs of the two numbers to be combined are like or unlike. The values of M* are given in the following table:

	ADD	SUBTRACT
Like signs	$M^* = M$	$M^* = \overline{M}$
Unlike signs	$M^* = \overline{M}$	$M^* = M$

In the multiply orders, use is made of the symbol E*. This has one of two values:

E* If the signs of the exponents are alike E* = E

If the signs of the exponents are unlike E* has the sign of E but the reflection of the magnitude

Example for this case: if E = -4 (1100)

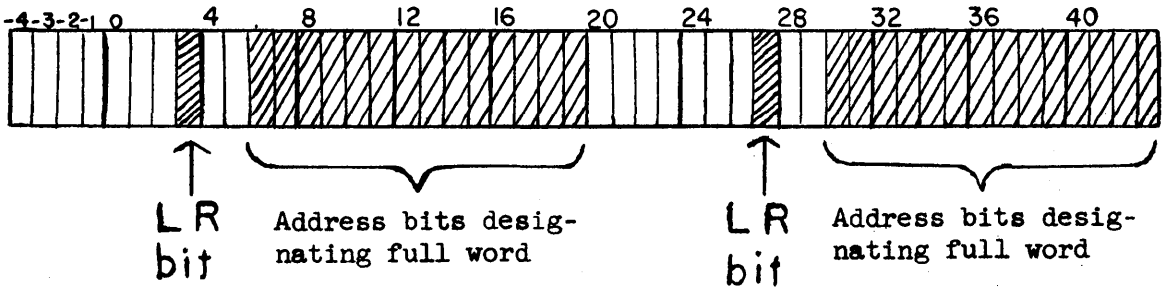
E* = -3 (1011)

n In the normalize orders, n is the number of left shifts of 16 required to normalize (UR). The allowed values of n are 0, 1, 2, 3.

a The symbol a is frequently used to refer to that answer obtained by performance of an order--that is, the primary result of the order. Usually the significance is obvious (thus, in a multiply σa is 0 or 1 according as the signs of the factors are like or unlike). In case of doubt, one must appeal to the vocabulary description.

A" In orders involving a round, the rounded result is indicated by a double prime. Thus Ma" in the MU column implies that a true round has been performed on the last significant bit (here, the 43rd). It will be noted from the table that in the cases of the divide and square root orders, the unrounded number Ma appears in MS.

$(N)_o$ The performance of some orders places a "nought-number" in the S-register. It is to be remembered that a nought-number is identical on the left and right-hand sides, involves the address bits only, and is zero elsewhere. The symbol for a number N expressed as a nought-number is $(N)_o$.



Nought-numbers appearing in S as result of transfer orders

The nought-numbers appearing in S resulting from transfer orders involve, in addition to the regular address-bits (6-19, 30-43), the "left-right bit" (3, 27 positions), which indicate by a zero or a 1 the left or right half-word of the address specified.

SUMMARY OF MORE IMPORTANT SYMBOLS USED

U	Universal register
R	R register
S	S register
m	Appropriate memory location
E	Exponent bits (-4 through -1) } after completion of the order
σ	Sign bit (0) } after completion of the order
M	Magnitude bits (1-43) } after completion of the order
—	Contents unchanged
X	Contents may have various values, not explicitly enumerated
a	Answer--i.e. primary result of operation
$\overline{A}$	Reflection of A
Z'	Contents of Z before the order was performed
A"	A, rounded
M*	See table in descriptive material
E*	See table in descriptive material
n	Number of 16-place left shifts in normalizing
$(N)_0$	Nought-number form of N
PresAdd+$\frac{1}{2}$	The address of the half-word succeeding the memory location of this order
{ }	Alternative possibilities

MANIAC II: REGISTER CONTENTS UPON LEGAL COMPLETION OF ORDERS

		U			R			S		
		E	σ	M	E	σ	M	E	σ	M
98	Set Sense to Zero	—	—	—	—	—	—	—	—	—
99	Set Sense to One	—	—	—	—	—	—	—	—	—
9A	Sense	—	—	—	—	—	—	—	—	—
A0	Absolute Magnitude to U	Em	0	Mm	—	—	—	Em	σm	Mm
A1	Absolute Magnitude of (R) to U	ER	0	MR	—	—	—	ER	σR	MR
A2	Negative of Absolute Magnitude to U	Em	1	Mm	—	—	—	Em	σm	Mm
A3	Negative of Absolute Magnitude of (R) to U	ER	1	MR	—	—	—	ER	σR	MR
A4	(Memory) to U	Em	σm	Mm	—	—	—	Em	σm	Mm
A5	(R) to U	ER	σR	MR	—	—	—	ER	σR	MR
A6	Negative of (Memory) to U	Em	$\overline{\sigma m}$	Mm	—	—	—	Em	σm	Mm
A7	Negative of (R) to U	ER	$\overline{\sigma R}$	MR	—	—	—	ER	σR	MR
A8	Fixed Add	—	σa	Ma	—	—	—	Em	σm	M*m
A9	Fixed Add of (R)	—	σa	Ma	—	—	—	ER	σR	M*R
AA	Fixed Subtract	—	σa	Ma	—	—	—	Em	σm	M*m
AB	Fixed Subtract of (R)	—	σa	Ma	—	—	—	ER	σR	M*R
AC	Floating Add	Ea	σa	Ma	—	—	—	X	σm	M*m
AD	Floating Add of (R)	Ea	σa	Ma	—	—	—	X	σR	M*R
AE	Floating Subtract	Ea	σa	Ma	—	—	—	X	σm	M*m
AF	Floating Subtract of (R)	Ea	σa	Ma	—	—	—	X	σR	M*R

		U			R			S		
		E	σ	M	E	σ	M	E	σ	M
B1	Set B_1	—	—	—	—	—	—	—	—	—
B2	Set B_2	—	—	—	—	—	—	—	—	—
B3	Set B_3	—	—	—	—	—	—	—	—	—
B5	Count B_1	—	—	—	—	—	—	←	$(B_1)_o$	→
B6	Count B_2	—	—	—	—	—	—	←	$(B_2)_o$	→
B7	Count B_3	—	—	—	—	—	—	←	$(B_3)_o$	→
B9	Count and Compare B_1	—	—	—	—	—	—	—	—	—
BA	Count and Compare B_2	—	—	—	—	—	—	—	—	—
BB	Count and Compare B_3	—	—	—	—	—	—	—	—	—
BC	(S) to Memory	—	—	—	—	—	—	—	—	—
BD	(S) to U	ES	σS	MS	—	—	—	—	—	—
BE	(U) to Memory	—	—	—	—	—	—	EU	σU	MU
BF	(U) to R	—	—	—	EU	σU	MU	—	—	—
C0	(Memory) to R	—	—	—	Em	σm	Mm	—	—	—
C1	(Memory) to R	—	—	—	Em	σm	Mm	—	—	—
C2	(R) to Memory	—	—	—	—	—	—	ER	σR	MR
C3	Extract	Ea	σa	Ma	—	—	—	Em	σm	Mm
C4	Substitute Left Address from S	—	—	—	—	—	—	Em	σm	Mm
C5	Substitute Right Address From S	—	—	—	—	—	—	Em	σm	Mm
C6	Substitute Left Address from U	—	—	—	—	—	—	Em	σm	Mm

		U			R			S		
		E	σ	M	E	σ	M	E	σ	M
C7	Substitute Right Address from U	—	—	—	—	—	—	Em	σm	Mm
C8	Unconditional Transfer to Left	—	—	—	—	—	—	←(PresAdd+½)$_0$ →		
C9	Unconditional Transfer to Right	—	—	—	—	—	—	←(PresAdd+½)$_0$ →		
CA	Conditional Transfer to Left on Overflow	—	—	—	—	—	—	If transfer occurs ↓ ←(PresAdd+½)$_0$ →		
CB	Conditional Transfer to Right on Overflow									
CC	Conditional Transfer to Left on Positive									
CD	Conditional Transfer to Right on Positive							—	—	—
CE	Conditional Transfer to Left on Zero							↑ If transfer does not occur		
CF	Conditional Transfer to Right on Zero									
D1	Normalize									
	No Spills	EU'-n	—	2^{16n}MUR'	—	—	2^{16n}MR'			
	and n = 0							—	—	—
	and n ≠ 0							{0001, 0110} —		—
	Negative Exponent Spill and "Allow"	-7	—	0	-7	—	0	{0001, 0110} —		—
D2	Round (overflow!)	—	—	Ma"	—	—	—	0	0	0
D4	Multiply									
	No Spills	Ea	σa	Ma(1-43)	Ea	σa	Ma(44-86)	E*m	σm	MU'
	Negative Exponent Spill and "Allow"	-7	σa	0	-7	σa	0	E*m	σm	MU'

	U			R			S		
	E	σ	M	E	σ	M	E	σ	M
D5 Multiply and Normalize									
No Spills	Ea	σa	Ma(1-43)	Ea	σa	Ma(44-86)			
and n = 0							E*m	σm	MU'
and n ≠ 0							{0001 / 0110}	σm	MU'
Negative Exponent Spill and "Allow"	-7	σa	0	-7	σa	0	{0001 / 0110}	σm	MU'
D6 Multiply and Round (n.b. overflow impossible)									
No Spills	Ea	σa	Ma"	Ea	σa	Ma(44-86)	0	0	0
Negative Exponent Spill and "Allow"	-7	σa	0	-7	σa	0	0	0	0
D7 Multiply, Normalize and Round									
No Spills	Ea	σa	Ma"	Ea	σa	Ma(44-86)	0	0	0
Negative Exponent Spill and "Allow"	-7	σa	0	-7	σa	0	0	0	0
D8 Fixed Divide	—	σa	Ma"	EU'	σa	REMDR	EU'	σa	Ma
D9 Floating Divide									
No Spills	Ea	σa	Ma"	EU'	σa	REMDR	Ea	σa	Ma
Negative Exponent Spill and "Allow"	-7	σa	0	EU'	σa	0	-7	σa	Ma

N.b. If 3 right shifts of 16 are required to legalize numerator, but otherwise the operation is legal, "Insignificant Light" is lit, and operation continues.

		U			R			S		
		E	σ	M	E	σ	M	E	σ	M
DA	Square Root									
	Exponent of radicand even	Ea	0	Ma"	—	0	0	1111	1	Ma
	Exponent of radicand odd	Ea	0	Ma(1-43)	—	0	Ma"(44-51)	0001	1	2^8Ma
E0	Change Sign of U	—	$\overline{\sigma U}'$	—	—	—	—	—	—	—
E1	Change Sign of R	—	—	—	—	$\overline{\sigma R}'$	—	—	—	—
E2	Plus Sign to U	—	0	—	—	—	—	—	—	—
E3	Plus Sign to R	—	—	—	—	0	—	—	—	—
E8	Left Shift	—	—	Ma	—	—	—	—	—	—
E9	Long Left Shift	—	—	Ma(U)	—	—	Ma(R)	—	—	—
EA	Logical Left Shift	Ea	σa	Ma(U)	—	—	Ma(R)	—	—	—
EC	Right Shift	—	—	Ma	—	—	—	—	—	—
ED	Long Right Shift	—	—	Ma(U)	—	—	Ma(R)	—	—	—
EE	Logical Right Shift	Ea	σa	Ma	—	—	—	—	—	—

January 4, 1956

MANIAC II ASSEMBLY ROUTINE
Preliminary

The assembly routine for Maniac II is designed to translate descriptive code into absolute code. The rules for making a descriptive code are given below. It will be seen that the descriptive code differs from the absolute code primarily in using logical or descriptive addresses. An attempt has been made to achieve reasonable flexibility with maximum simplicity.

The descriptive tape is punched on ordinary five-hole paper tape. The assembly routine transcribes the descriptive code onto magnetic tape, after which the paper tape may be discarded. When changes are to be made in the descriptive code, these changes are punched on a correction tape and the assembly routine makes the changes and produces a corrected magnetic tape record.

The first word on a descriptive tape must be 00XXXX, where XXXX is the desired absolute half-word starting address of the code. The last word on a descriptive tape must be the special control word OCODED.

INSTRUCTIONS

Boxes.

The bulk of the descriptive tape, in general, consists of a series of half-word (six tetrad) instructions, separated, by control words, into numbered groups called boxes. These boxes, which may be numbered in arbitrary sequence, usually correspond to flow-diagram boxes. Since correction tapes must contain complete boxes, it is important that boxes be reasonably short.

In the following discussion, X represents an arbitrary tetrad upon which attention is focused; a dash, - , denotes an irrelevant tetrad, included to show relative position.

The box number control words are in the form 00XXXX, where the last two tetrads specify the box number. The significance of the pair --XX-- will be described later (cf. pp. 4-5).

Instruction Classes

Instructions are punched in the form XXXXXX. The first two tetrads specify the order, with or without breakpoints, in absolute form. The interpretation of the address part, --XXXX, depends on the class of the instruction specified by the order.

Class A instructions are those with address parts which never refer to a memory location. They are: Set Sense, Sense, Round, Select B, Normalize, Square Root, Change Sign, Plus Sign, the six Shift instruction, and Position Tape. The address part of a Class A instruction is left unchanged. (Only one kind of Stop instruction is allowed by the assembly routine, and that is OFF---. The instruction is in Class A.)

Class B instructions are those which normally address another instruction. They are Substitute Address and the four Transfer instructions. In most cases, the address part of a Class B instruction is translated as the half-word address of Instruction --XX, Box XX--. The exceptions (F---, 00--, 80--, and --00) will be discussed later.

All other instructions are in Class C. In most cases, the address part of a Class C instruction is translated as follows: the first two tetrads, XX--, give the Storage Type, the second pair, --XX, gives the

full-word address within the particular type.

b, r, and Absolute Addresses.

The largest number which is given to a Storage Type is 3F; if it is desired to have b = 1 in a Class C instruction (to call for automatic address modification), then the address part must be increased by 8000 (e.g., AF03 instead of 2F03). For b = 1 in a Class B instruction, consult the next section. If it is desired to have r = 1 (e.g., to address the R register) in a Class C instruction, then the address of exactly 4000 must be used. If it is desired that a Class C instruction have an absolute memory address other than 0000 or 8000, then the descriptive address must be that absolute address plus 4000. The assembly routine will remove the 4000. The absolute address may have b = 1 or 0. The absolute addresses 0000 and 8000 should be entered without the extra 4000.

Addressing the Current Box.

If the address part is 00-- or 80--, in a Class B or a Class C instruction, the reference is to the box containing that instruction. For a Class B instruction, with address 00XX, the translation is the half-word address of Instruction XX, current box. For a Class C instruction, the translation is the address of the full word containing Instruction XX, current box. Note that 80-- for Class B means b = 1; it is only when reference is to the current box that Class B instructions may have b = 1.

Subroutines.

The allowed box numbers are 01 through EF, 80 excluded. If, in a

Class B instruction, the address is FXXX, then the instruction must be a transfer,and the transfer is to the subroutine whose call number is XXX. The most common subroutines are on the same magnetic tape as the assembly routine, and are brought in automatically. For other subroutines, the assembly routine stops to allow the appropriate paper tapes to be put into the photoreader. Maniac II subroutines always return the control to the instruction next in sequence after that instruction which transferred control to the subroutine.

Variable Transfers.

If, in a Class B instruction, there is an apparent reference to an address XX00, then a special meaning is attached, since instructions are numbered from 01, not from 00. A Transfer instruction with such an address is interpreted as Variable Transfer Number XX, and all Substitute Address instructions with the same Address Part are translated so as to substitute into their corresponding Variable Transfer.

Left and Right Instructions.

An absolute code for Maniac II has two instructions per word. It is generally irrelevant whether a particular instruction is on the left or on the right, as far as the programmer is concerned. There are cases, however, where it may be important. If it is desired that a particular box have its first instruction on the left, then the box number control word, 00XXXX, for that box should be 00X8XX. If it is desired that the first instruction be on the right, then the control word should be 00X4XX. If the programmer does not care, then he should use 00X0XX.

Insertion of Blank Spaces.

If changes are anticipated in a particular box, it is frequently useful to leave some blank space for them, so that the rest of the

changed code will have the same absolute translation. If a box number control word is 00X---, then the assembly routine will leave X full-word blanks at the end of the box, and will insert a transfer over the blanks. However, the operator may elect to have the assembly routine ignore this tetrad and leave no blanks; this would generally be done, if at all, on the final assembly of a completely debugged problem.

STORAGE

The Storage Types are numbered from 01 through 3F, and each type XX referred to must be introduced by a twelve tetrad control word of the form 8---XX 0-----. The control words may be anywhere on the descriptive tape. For every storage type of the form 0X which is introduced, there are two possible interpretations of storage types 1X, 2X, 3X.

8---0C0D0064 (leave me 100 words of 0C storage)

(i) If any of the latter types is introduced by a control word, then the interpretation of an address of a Class C instruction referring to that storage type is as previously described on pp. 2-3.

(ii) If any of the latter types, say 2X, is not introduced by a control word, then the corresponding interpretation is as follows: the storage type is 0X and the three other tetrads, in this case 2-λX, give the address.

Tetrad X of the control word, ------ -X----, must equal A or D. If this tetrad is A, then ------ --XXXX is the absolute address to be assigned to the zeroth word* of the associated storage type. If

*Note that storage words are counted from 0, whereas instructions in a box are counted from 1.

the identifying tetrad is D, then these four last tetrads of the control word give the number of blanks to be left for the associated storage type. Absolute addresses for such types are assigned in the order in which their introductory control words appear on the descriptive tape. The two variants are checked for possible overlap.

Subroutine Dynamic Storage

Subroutines will use only the 01 storage type, from word 00 on, as far as necessary. (Subroutine constants will be hidden in the subroutine code.) If it is desired that Subroutine XXX have its dynamic storage moved ahead YY addresses, this can be accomplished by putting anywhere on the descriptive tape a so-called Delta Control word, namely 8YYXXX.

Input of Basic Constants

Storage types 0A through 0F have the special property that words to be stored there may be put on the descriptive tape. These twelve tetrad words must follow immediately after the control word for the particular storage type. If the control word has provided for a number of blanks, then the specified number of blanks is left after storing the storage words on the descriptive tape.

63 types
256 words

0A – 0F
same as
A, B & C.

Words for 0A storage are treated as instruction pairs; if an Order Part is not in the vocabulary (00, for example), the instruction will be treated as though it were in Class C. Words for 0B storage are left unchanged. Words for 0C and 0D storage are interpreted as fixed point binary coded decimal, and converted to fixed point binary numbers. Words for 0E and 0F storage are interpreted as binary coded floating decimal, and converted to normal floating point binary numbers.

OUTPUT

The absolute code produced by the assembly routine is punched on paper tape and/or recorded on magnetic tape, at the operator's option.

The final code and basic constants are also printed on the Fast Printer, unless printing is suppressed by the operator. The first page or pages of the printout list the absolute starting addresses of the various storage types introduced. Subsequent pages contain sixteen instruction pairs each (except perhaps the first such page, if the code does not start at an address which is a multiple of sixteen), with extra space between boxes. The printout consists of the box and instruction numbers, the absolute location, the absolute instruction, and the descriptive Address Part. When an instruction addresses 0A through 0F storage, the contents of that address may be printed or not, at the option of the operator.

When a descriptive code is being reassembled with changes, the operator may elect to have printed only those pages which differ from pages of the unchanged code.

CORRECTIONS

One or more correction tapes may precede the main descriptive tape (which will, in general, already be on magnetic tape) to modify, delete, or insert boxes, or to introduce or reintroduce storage types. Each correction tape must be terminated by the control word OCODED. Each correction tape takes precedence over any tape read in after it.

To modify a box, the box number control word and the instructions are punched just as on the main tape. To delete a box, the control

word is punched with no instructions following it. To insert a box immediately preceding Box XX, the normal box number control word for the box to be inserted must be immediately preceded by the special control word FF--XX.

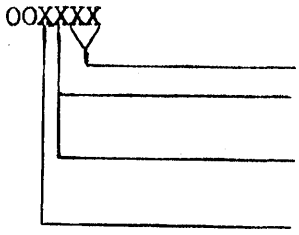
In addition to the normal options of ignoring or not the tetrad --X--- of a box number control word (which specifies the number of blanks), there is the option to ignore such tetrads of box number control words on correction tapes only, and attempt to leave unchanged as much of the original code as possible.

Max of 15 blanks at end of tape
no max for Gap left at end of code

1. First word on tape: 00XXXX — Absolute starting address of the code.

   Last word on tape: 0CODED — Also for all correction tapes.

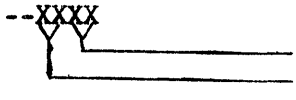
2. Box number control word:



Box number.
If 8, first instruction of that box on left side.
If 4, first instruction on right side.
Number of blanks added at the end of box.

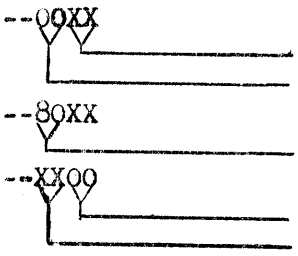
3. Class A instructions: Address part never refers to memory location. Left unchanged.

4. Class B instructions:



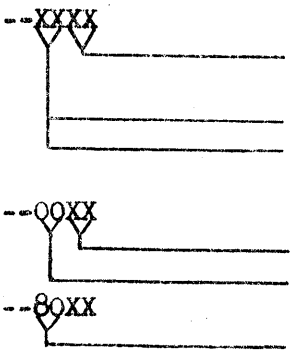
Substitutions and transfers.
Instruction number.
Box number.

--FXXX — Instruction must be a transfer to subroutine with call number XXX.



Instruction number.
Current box, b=0.
Current box, b=1.
Indicates variable transfer.
Variable transfer number.

5. Class C instructions:



Full word address of storage type.
01-3F: Storage Type, b=0.
81-BF: Storage Types 01-3F, b=1.
Instruction number.
Current box, b=0.
Current box, b=1.

Translation is address of full word containing the instruction.

	--XXXX	40-7F and C0-FF: XXXX is absolute b-mode address + 4000; translate by subtracting 4000.
	--4000	Addresses R register.
	--0000 --8000	Special absolute addresses, b=0 or 1.
6. Storage:	8---XX 0-----	Control word. Storage Types 01-3F.
	8----- 0AXXXX	Absolute address of zeroth word of associated storage type.
	8----- 0DXXXX	Number of blanks reserved for associated storage type.
If there are constants on descriptive tape following these control words, they are:	8---0A 0-----	Treated as instruction pairs.
	8---0B 0-----	Left unchanged.
	8---0C 0-----	Fixed point coded decimal, to be converted to fixed point binary.
	8---0E 0----- 8---0F 0-----	Floating point coded decimal, to be converted to floating point binary.
7. Subroutine dynamic storage (Delta control word):	8XXXXX	Subroutine call number. Dynamic storage advanced by this number of addresses.
8. Corrections:		
To modify a box:		Box number control word, followed by instructions, as on main tape.
To delete a box:		Box number control word, with no instructions.
To insert a box:	FF--XX	New box immediately precedes Box XX.
	00XXXX	Box number control word for box being inserted; follow by instructions, as on main tape.
End of correction tape:	0CODED	

July 18, 1956

The automatic coding scheme for Maniac II is herein described. The assembly routine for this scheme, hereafter called "Madcap" (Mathematical and Descriptive Coding Assembly Program), will translate a series of statements into a computer-ready code. The philosophy has been that these statements should resemble a mathematical formulation of the problem in terms of a flow diagram. Thus the first step in preparing a problem for Madcap is to construct a flow diagram (perhaps only in the mind) consisting of an aggregate of defining equations, control equations and information statements in acceptable notation. This, along with the input quantities, constitutes the input for Madcap. Madcap will then produce a computer code, including the assignment of all storage, and give the programmer a printed record and a magnetic tape record of this code.

Notation

The word "literal", when used here as a noun, refers to the literal representation of a variable or constant. The word "quantity" implies number or literal. Any expression containing more than one factor or term is said to be "compound".

The symbols which may be typed and punched as distinguishable tape characters are: "a - z", "A - Z", "0 - 9", "#", "2", "3", "Δ", "π", "α", ">", "∈", "(", ")", ",", "√", "*", "→", "|", "Σ", "/", "+", "-", "×", ".", "=". Individual (non-indexed) quantities are represented in the following way:

(1) Numbers to be converted to computer floating point form are written in decimal with a decimal point, e.g., 492.75, 1.0, 0.0016. Numbers written without the decimal point are converted as decimal integers and are scaled to be used in address arithmetic, e.g., 27, 1.

(2) Literals may be written in three ways:

a. A single letter: a, b, ..., A, B, ..., α, Δ, ϵ.

b. A single letter followed by any decimal integer: a2, ..., A149, ..., α 39, etc.

c. A capital letter followed by a small letter: Rm, Af, etc.

These are not to be confused with subscripted quantities, which will now be discussed.

A quantity dependent on one index (an element of a one dimensional array) is indicated by the array name (a single literal) followed by a period and the index (subscript), which itself is a single quantity, e.g., "R.i", "A2.3","Pz.f2". Compound subscripts must be entirely in parenthesis, e.g., "R.(i+1)", "P.(2j)". Markers in one dimensional arrays, that is, subscripted subscripts are indicated as in examples:

"R.i.j" means R_{i_j}

"P.m.(j+1)" means $P_{m_{j+1}}$

"R.(P.i+1)" means $R_{P_{i+1}}$.

A quantity dependent on two indices (an element of a two dimensional array) is indicated by the array name followed by a period and the two subscripts separated by a comma. Again, compound subscripts are in parenthesis. Examples:

"A.i,j" means $A_{i,j}$

"A.i,14" means $A_{i,14}$

"A.(i+1),(2j)" means $A_{i+1,2j}$.

Triple subscripts, superscripts, or subscripted double subscripts may not be written as such.

The choice of literals to represent constants or variables in fixed or floating point is left to the discretion of the programmer, but this choice is communicated to Madcap in the form of an information equation. This is discussed under input.

Algebra

The algebraic formulation of the mathematical equation appears in the form of defining equations. That is, the quantity to the left of the equals sign is being defined (or redefined) in terms of the established quantities connected algebraically on the right side of the equation. Storage is arranged for this new quantity (or merely located if it is being redefined). The quantities on the right have storage locations which have previously been determined. An arrow, "→", acts like an equals sign except that the roles of left and right side are interchanged in such equations.

The rules governing the use of the algebraic operation symbols are as follows:

(1) "+", "-", and "×" have their usual binary operational meaning when inserted between two operands. The symbol "×" may be omitted when there is no ambiguity, which is most of the time. It must be included in such instances as a letter times

a number, "R × 14", "a × 3", or uppercase letter times lower case letter, "**P** × b".

(2) The symbol for division is "/". "/A" is equivalent to $\times A^{-1}$. Thus, compound denominators must be included in parenthesis. Examples:

"ac/b" means $\frac{a\ c}{b}$

"a/cb" means $\frac{a\ b}{c}$

"a/(cb)" means $\frac{a}{c\ b}$

"a/b/c" means $\frac{a}{b\ c}$

(3) The symbol "$\sqrt{}$" means "take square root of" where the radicand must be in parenthesis.

(4) Simple integral exponentiation is accomplished with the symbols "2" and "3" and composition thereof. Examples:

"R^2", "$(a+b)^3$", "P^2P^3", "$((Z.i)^2)^3$".

Other exponentiation is accomplished as such, "P(gth)" means P^g where g may be a variable or constant; the coding will be a subroutine.

(5) Functions are indicated by a characteristic three or more letter symbolic title. The argument (or arguments) appear in parenthesis. Thus we have "sin (X)", "log (T.i-1.0)" and "max (R.j)" as examples. The common functions and logical tasks indicated in this manner are subroutines included with Madcap. A list and description of these routines is available.

Examples of defining equations using the above ground rules are:

"y = (a+b) sin (T.i) / (c+d) + $Rn^2/t\sqrt{(1-g^3)}$",

"i+1 → i",

"R.i = (R.(i+1) + R.(i-1)) / 2.0 + (B x a) (bth)".

Fixed point quantities may appear in floating point equations provided the scaling is accounted for, e.g.,

"D.k = (S.k - S.(k-1)) x1/(i.k - i.(k-1))".

The "1", "i.k" and "i.(k-1)" are fixed point.

Logical Control

The code is constructed in the sequence that the equations appear on the tape. Thus the computer control follows this sequence unless instructed to the contrary by a control equation. First, the programmer arranges the equations of his problem into groups, assigning a number to each group. A group corresponds to a flow diagram box, to a simple loop, or, quite often, to a single equation; the characteristic property is that computation always proceeds through the group from the beginning (the single entry). The group number may be a one, two, or three digit number. It is preceded by the symbol "# ". A group of equations is introduced by its number. The control equations refer to these numbers.

There are three types of control equations. The first corresponds to an unconditional transfer of control. If control is being relinquished, the notation used is

"go to#27".

If a later transfer is to be specified, the notation is

"go to#27 thru#32, then#18".

That is:

"Transfer to group number 27; continue until group 32 has been finished, and then transfer to group 18".

The second type of control equation corresponds to conditional transfer of control. This type of equation is introduced by the word "if", followed first by a condition requiring an equality or inequality, and then by a direct transfer statement indicating what is to be done provided the condition is met. The usual tape sequence is not disturbed when the condition is not met. Examples:

"if i > I go to#25",

"if j = 2 m-1 go to#25 thru#27, then#4".

The third type of control equation provides a simple method of writing induction loops. Thus,

"#14 for i = 0(1)I do#43 thru#19, go to#15"

indicates that a loop should be constructed from groups#43 through #19. (What comes in between may be governed by other control equation.) The index i is to take on the values 0 through I in steps of 1; control is to be transferred to group #15 when the induction is complete. If only one group is included in the induction, the "thru #19" may be omitted. If the "go to #15" is omitted, Madcap will follow the usual tape sequence unless under the influence of a previous loop or variable transfer control equation. Examples:

"#1 j = 1(1)J do#2",

"#2 R.j = R.(j-1) + D",

"#3 .

The loop contains only equation#2; following the induction, control begins with equation #3.

"#1 i = 1(2)8 do #2 thru #3 go #4",

"#2 j = 0(p)J do #3",

"#3 A.i = B.i,j × A.j",

"#4 .

This is a double induction. With completion of the inner j loop, the control is still under the influence of the i loop, which will be completed before proceeding to equation# 4.

The upper limit of the induction may be replaced by either of the words "till" or "inf". Following "till" appears a special condition to be met to end the induction. The "inf" means infinity; thus, the induction has no end. In both cases the "go" transfer is omitted. Examples:

"#14 i = 1(1)till A > A.i",

"#29 n = 0(1)inf do #30 thru #90".

Note that a loop control equation is always a group with its own number. It is sometimes necessary to enter a loop not at the beginning but at the point where the incrementing and comparing of the index (in that order) is taking place. This may be done in any control equation by the special notation "go #4+", where 4 is the number of the loop control equation.

The "go" in the conditional transfer equation or the "do" in the loop equation may at any time be replaced by the word "let" followed immediately by one or more defining equations. Examples:

"if i = 8 let d = S, X = -T",

"#9 j = 1(2)J let R.j = Rm.j go #12".

Tape Organization-Input

A series of information statements are included on the tape, preceding the control and defining equations which constitute the computation of the problem. The following information is to be given:

(1) The starting address of the code, e.g., "start at 02C9". If this statement does not appear the code will begin at 0000.

(2) The literal assignment of fixed point quantities, usually indices, e.g., "fixed point--i, j, k, A, F, A2, Zk."

(3) The range of the index for indexables, in order that Madcap may allow sufficient space. For example, "R.i for i = -1(1)50". The numerical values of the R's may or may not be given. If given, they follow the above statement and the word "are". For example, "A.k for k = 1(1)3 are 19.6, 16.2, 12.3". The commas may be omitted. If the range of an index is indicated by a letter (e.g.N), then N must be assigned a value during the input--see (4) below.

(4) The literal representation of constants, e.g., "C = 0.0492", "N = 49", "α2 = 46.4". Of course, constants may be typed in formulas either literally or numerically.

Punctuation of the problem is required only as indicated by the examples (that part included in quotation). Commas may optionally separate equations of a group. Spaces and carriage return characters

on the tape are completely ignored by Madcap except while a list of input numbers is being read, at which time it is assumed that something (comma, space, or carriage return) separates two numbers. Note that any character with a seventh hole is ignored by the computer itself. The end of the tape is indicated by the word "end".

There are many special English words which Madcap can interpret. Thus,

"read (R.k)"

means read and convert one number, calling it R.k;

"print (all i.j)"

means convert and print all of the i.j's. The converting in either case depends on whether the quantity is fixed or floating point, of course. Sense light work is accomplished as such:

"set sense to 1 (2, 4, 5)",

"sense (3)".

See the list and description of subroutines for other words Madcap can interpret.

Box	Instr	Order	Tag	Address		B_1	B_2	B_3	U	R	S	Comments

	ADDRESS		NUMBER			ADDRESS		NUMBER	
	Absolute	Descriptive				Absolute	Descriptive		

90 b m	RH	Read Hexad. Read one hexad into X from reader.
91 b m	RW	Read Word. Read one word into X from reader.
92 b m	PH	Punch Hexad. Punch one hexad from X.
93 b m	PW	Punch Word. Punch (X).
94 b m	FPr	Fast Print. Print one line according to the matrix stored at the 19 consecutive addresses starting with X.
95 - -	SFP	Space Fast Printer. Energize fast printer format control.
96 b m	Flx	Flexowrite. Flexoprint (X); also Flexopunch (X) if Flexopunch switch is on.
97 - -	RC	Return Carriage. Return Flexowriter carriage and advance platen, without printing.
98		
99		
9A		
9B		
9C b m	SS0	Set Sense to Zero. Set to zero all Sense Lights corresponding to ones in X.
9D b m	SS1	Set Sense to One. Set to one all Sense Lights corresponding to ones in X.
9E b m	Sn	Sense. Skip the next instruction if at least one Sense Light addressed by X contains a zero.
9F b m	Sn	Sense. Identical to 9E.
A0 b m	Mm→U	Magnitude to U. (X) →(U).
A1 - -	MR→U	Magnitude of (R) to U. (R) →(U).
A2 b m	-Mm→U	Negative Magnitude to U. - (X) →(U).
A3 - -	-MR→U	Negative Magnitude of (R) to U. - (R) →(U).
A4 b m	m→U	Memory to U. (X)→(U).
A5 - -	R→U	(R) to U. (R)→(U).

A6	b	m	-m→U — Negative to U. -(X)→(U).
A7	-	-	-R→U — Negative of (R) to U. -(R)→(U).
A8	b	m	+ — Fixed Add. (U') + (X') → (U'), fixed point.
A9	-	-	+R — Fixed Add of (R). (U') + (R') → (U'), fixed point.
AA	b	m	- — Fixed Subtract. (U') - (X') → (U'), fixed point.
AB	-	-	-R — Fixed Subtract of (R). (U') - (R') → (U'), fixed point.
AC	b	m	F+ — Floating Add. (U) + (X) → (U), floating point.
AD	-	-	F+R — Floating Add of (R). (U) + (R) → (U), floating point.
AE	b	m	F- — Floating Subtract. (U) - (X) → (U), floating point.
AF	-	-	F-R — Floating Subtract of (R). (U) - (R) → (U), floating point.
B0			
B1	b	m	SB1 — Set B1. (6-19X) → (B1).
B2	b	m	SB2 — Set B2. (6-19X) → (B2).
B3	b	m	SB3 — Set B3. (6-19X) → (B3).
B4			
B5	b	m	CB1 — Count B1. (B1) + (6-19X) → (B1), then (B1) to S_o.
B6	b	m	CB2 — Count B2. (B2) + (6-19X) → (B2), then (B2) to S_o.
B7	b	m	CB3 — Count B3. (B3) + (6-19X) → (B3), then (B3) to S_o.
B8			
B9	b	m	CB1C — Count B1 and Compare. (B1) + (6-19X) → (B1), then skip next instruction unless (B1) = (30-43X).
BA	b	m	CB2C — Count B2 and Compare. (B2) + (6-19X) → (B2), then skip next instruction unless (B2) = (30-43X).
BB	b	m	CB3C — Count B3 and Compare. (B3) + (6-19X) → (B3), then skip next instruction unless (B3) = (30-43X).
BC	b	m	StS — Store S. (S)→(X).
BD	-	-	S→U — (S) to U. (S)→(U).

BE b m	StU	Store U. (U)→(X).
BF - -	U→R	(U) to R. (U)→(R).
C0 b m	m→R	Memory to R. (X)→(R).
C1 b m	m→R	Memory to R. (X)→(R).
C2 b m	StR	Store R. (R)→(X).
C3 b m	E	Extract. Replace those bits of (U) corresponding to ones in R by the corresponding bits of (X).
C4 b m	SLP	Substitute Left Address from PF. (PF) → (3 and 6-19X).
C5 b m	SRP	Substitute Right Address from PF. (PF) → (27 and 30-43X).
C6 b m	SLU	Substitute Left Address from U. (3 and 6-19U) → (3 and 6-19X).
C7 b m	SRU	Substitute Right Address from U. (27 and 30-43U) → (27 and 30-43X).
C8 b m	TL	Transfer to Left. Transfer control to X, left, unconditionally.
C9 b m	TR	Transfer to Right. Transfer control to X, right, unconditionally.
CA b m	TLOv	Transfer to Left on Overflow. Transfer control to X, left, if the overflow signal is on, turning it off.
CB b m	TROv	Transfer to Right on Overflow. Transfer control to X, right, if the overflow signal is on, turning it off.
CC b m	TLP	Transfer to Left on Plus. Transfer control to X, left, if the number in U is positive.
CD b m	TRP	Transfer to Right on Plus. Transfer control to X, right, if the number in U is positive.
CE b m	TLZ	Transfer to Left on Zero. Transfer control to X, left, if the number in U has zero magnitude.
CF b m	TRZ	Transfer to Right on Zero. Transfer control to X, right, if the number in U has zero magnitude.

D0

D1 - - Nm — Normalize. Put (UR), treated as a floating point number, into normal form, unless more than three shifts of sixteen would be required, in which case make the three shifts and no more.

D2 - - Rnd — Round. If (1R) = 1, increase (MU) by 2^{-43}.

D3

D4 b m X — Multiply. Multiply (X) by (U), floating or fixed point, putting the full product into UR.

D5 b m XN — Multiply and Normalize. Multiply (X) by (U), floating point, putting the full product into UR; then normalize, as defined by D1.

D6 b m XR — Multiply and Round. Multiply (X) by (U), floating or fixed point, putting the full product in UR; then round, as defined by D2.

D7 b m XNR — Multiply, Normalize and Round. Multiply (X) by (U), floating point, putting the full product into UR; then normalize, as defined in D1; then round, as defined in D2.

D8 b m Dv — Fixed Divide. Divide (U'R) by (X'), fixed point, putting the true rounded quotient in U' and the remainder in R'. Stop if the rounded quotient is greater than or equal to one.

D9 b m FDv — Floating Divide. Divide (UR) by (X), floating point, putting the true rounded quotient in U and the remainder in R. Stop on exponent spill or division by zero.

DA - - SqR — Square Root. Extract the square root of (U), floating or fixed point, and put it in U. Stop if initial (U) negative.

DB

DC

DD

DE

DF

E0 - - CSU Change Sign of U. -(U')→(U').

E1 - - CSR Change Sign of R. -(R')→(R').

E2 - - PSU Plus Sign to U. (U') →(U').

E3 - - PSR Plus Sign to R. (R') →(R').

E4

E5

E6

E7

E8 b m L Left Shift. Shift (MU) left X places (mod 128), setting the overflow signal if any ones are shifted out of 1U.

E9 b m LL Long Left Shift. Shift (MUR) left X places (mod 128), setting the overflow signal if any ones are shifted out of 1U; (1R)→(43U) and (1U)→(43R).

EA b m LgL Logical Left Shift. Shift (U) and (MR) left X places (mod 128); (-4U)→(43R).

EB

EC b m R Right Shift. Shift (MU) right X places (mod 128).

ED b m LR Long Right Shift. Shift (MUR) right X places (mod 128); (43U)→(1R).

EE b m LgR Logical Right Shift. Shift (U) right X places (mod 128).

EF

F0 b m DT Dump on Tape.

F1

F2 b m CT Call from Tape.

F3

F4 b m CTB Call from Tape Backwards.

F5

F6 b m AT Advance Tape.

F7

F8 b m BT Backspace Tape.

F9

FA

FB

FC

FD

FE

FF

90 b m Read Hexad
91 b m Read Word
92 b m Punch Hexad
93 b m Punch Word
94 b m Fast Print
95 - - Space Fast Printer
96 b m Flexowrite
97 - - Return Carriage
98
99
9A
9B
9C b m Set Sense to Zero
9D b m Set Sense to One
9E b m Sense
9F b m Sense

A0 b m Magnitude to U
A1 - - Magnitude of (R) to U
A2 b m Negative Magnitude to U
A3 - - Negative Magnitude of (R) to U
A4 b m Memory to U
A5 - - (R) to U
A6 b m Negative to U
A7 - - Negative of (R) to U
A8 b m Fixed Add
A9 - - Fixed Add of (R)
AA b m Fixed Subtract
AB - - Fixed Subtract of (R)
AC b m Floating Add
AD - - Floating Add of (R)
AE b m Floating Subtract
AF - - Floating Subtract of (R)

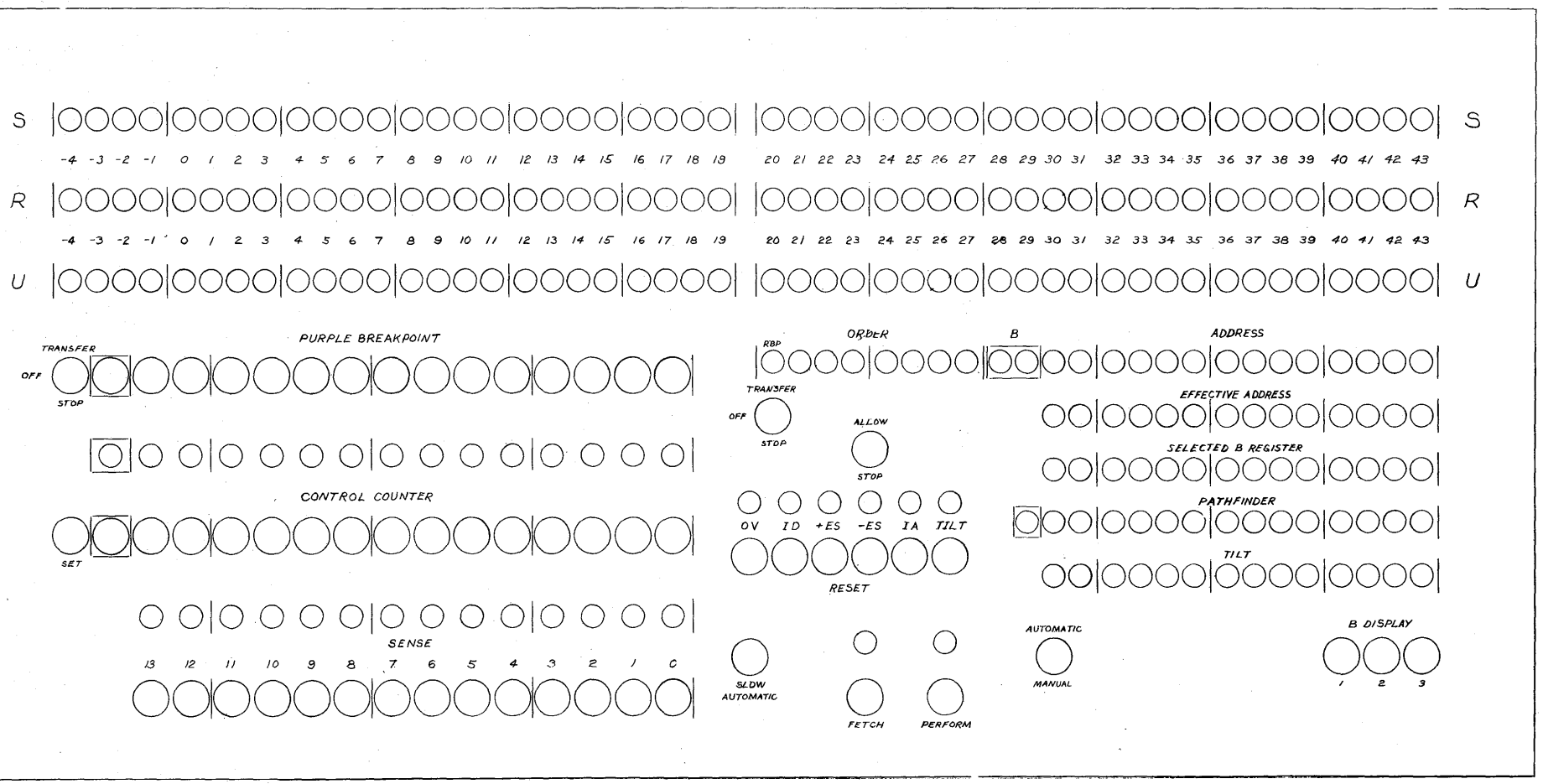
B0
B1 b m Set B1
B2 b m Set B2
B3 b m Set B3
B4
B5 b m Count B1, (B1) to S_o
B6 b m Count B2, (B2) to S_o
B7 b m Count B3, (B3) to S_o
B8
B9 b m Count B1 and Compare
BA b m Count B2 and Compare
BB b m Count B3 and Compare
BC b m (S) to M
BD - - (S) to U
BE b m (U) to Memory
BF - - (U) to R

C0 b m Memory to R
C1 b m Memory to R
C2 b m (R) to Memory
C3 b m Extract
C4 b m Substitute Left Address from (PF)
C5 b m Substitute Right Address from (PF)
C6 b m Substitute Left Address from (U)
C7 b m Substitute Right Address from (U)
C8 b m Transfer to Left
C9 b m Transfer to Right
CA b m Transfer to Left on Overflow
CB b m Transfer to Right on Overflow
CC b m Transfer to Left on Plus
CD b m Transfer to Right on Plus
CE b m Transfer to Left on Zero
CF b m Transfer to Right on Zero

D0
D1 - - Normalize
D2 - - Round
D3
D4 b m Multiply
D5 b m Multiply and Normalize
D6 b m Multiply and Round
D7 b m Multiply, Normalize and Round
D8 b m Fixed Divide
D9 b m Floating Divide
DA - - Square Root
DB
DC
DD
DE
DF

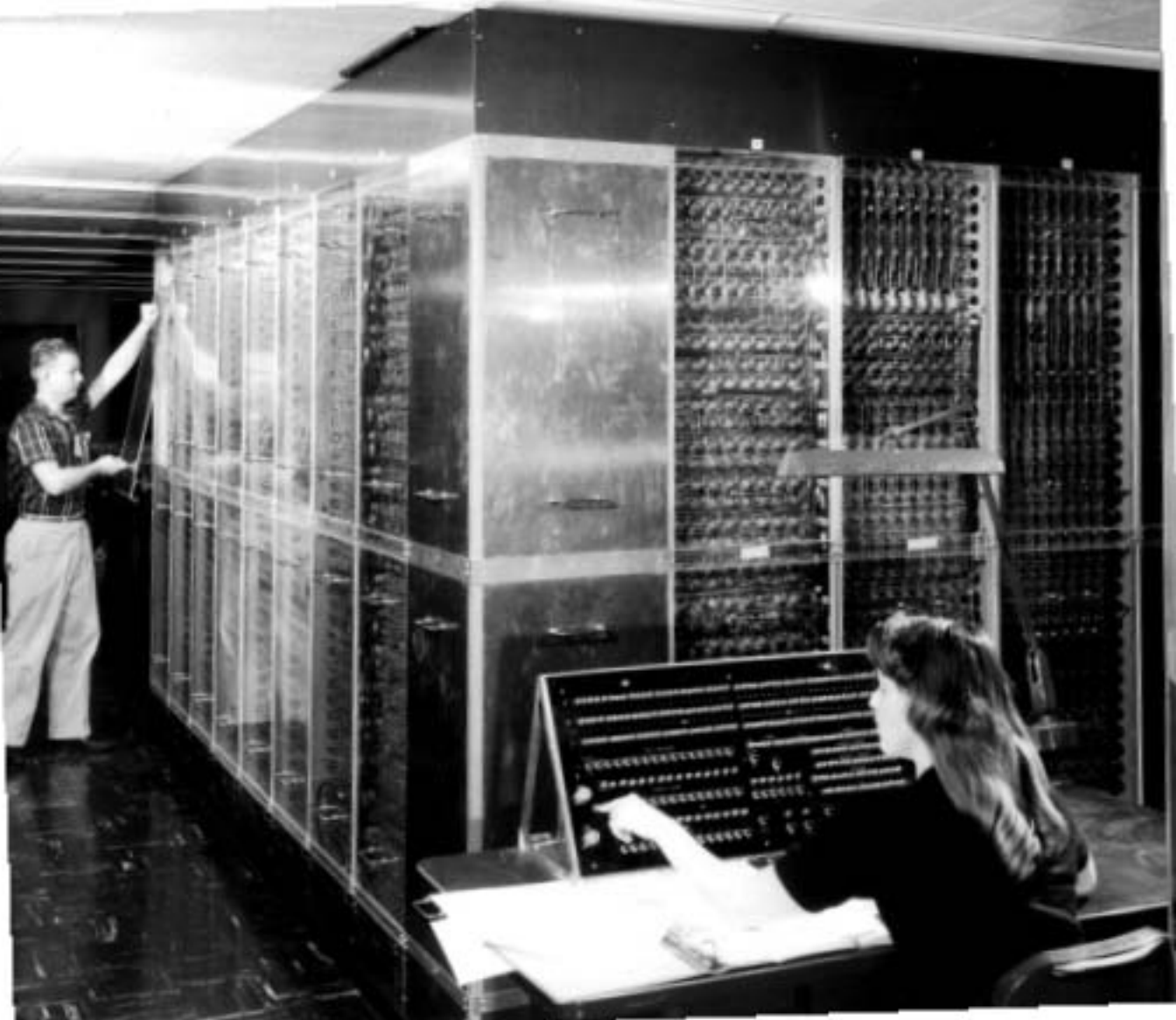
E0 - - Change Sign of U
E1 - - Change Sign of R
E2 - - Plus Sign to U
E3 - - Plus Sign to R
E4
E5
E6
E7
E8 b m Left Shift
E9 b m Long Left Shift
EA b m Logical Left Shift
EB
EC b m Right Shift
ED b m Long Right Shift
EE b m Logical Right Shift
EF

F0 b m ~~Dump~~ ~~Tape~~
F1
F2 b m Call Tape
F3
F4 b m Call Tape Backwards
F5
F6 b m Advance Tape
F7
F8 b m Backspace Tape
F9
FA
FB
FC
FD
FE
FF



CONTROL PANEL

DATE	ALTERATIONS
UNIVERSITY OF CALIFORNIA	
LOS ALAMOS SCIENTIFIC LABORATORY	
COMPUTER PROJECT	
NAME MANIAC II - CONTROL PANEL	
SHEET 1	
MATERIAL 1/8" LAMICOID SHEET: G5-54	
	BY / DATE
DESIGNED	SCALE 1:1
DRAWN J. HK 7-11-52	CHECKED
30Y-41041	



MANIAC II

from Jim R. & Walt O., Dec 6 '56

Last wire expected to be soldered: Dec 14

Measured speeds for basic operations:

* Fetch	8 μs	F
Add	6 μs	A ← also round
† Shift 1	1.5 μs	S
Complement	1 μs	C
Inter-register communic^n	1 μs	R
Exponent Add	3 μs	EA

* Actually now somewhat larger, but can & will be tightened up when machine is in operation.

† Set at this; can be decreased to 1.1 μs by replacing delay-lines.

Some vocab. times

ADD: $1.5F + A = 18$ μs

MULTIPLY: Assume 7 lead zeroes (ave.)
$\frac{1}{2}$ remaining bits = 1 (ave.)

$\therefore N$ = expected # of 1's $= \frac{43-7}{2} = 18$

Mult (F.P.) $1.5F + (N)A + (43-N)S = 12 + 108 + 37.5$
$= 157.5$

Mult. Round = 163.5

Mult. R, Nmlze = $163.5 + 16\nu S$ $\quad \nu = 0, 1, 2$

MR, N.

163.5	$\nu = 0$
187.5	$\nu = 1$
211.5	$\nu = 2$

DIVIDE

FP's legal : $\underbrace{(43+1)A + \frac{43}{2}S + 2R}_{1.5F + \text{ Round}}$

$= 12 + 264 + 64.5 + 2$

$= 342.5$

If FP den. too small, $+ 24\nu$ $(\nu = 1, 2)$

SQUARE ROOT

$0.5F + \underbrace{(43+1)}_{\text{Round}}(2S + A) + C + \cancel{Ey} + EA + \underbrace{\left(\frac{16}{2}S + A\right)}_{\text{Exp Round}}$

← If Exp radicand → Odd

$= 4 + (44)(9) + 1 + 3 + (18)$

$= 404 (+18)$