AUTOMATIC FLOATING-POINT OPERATIONS (Special Feature)

The Automatic Floating-Point Operations special feature provides the IBM 1620 Data Processing System (Figure 1) with the ability to do floating-point arithmetic by using floating-point instructions, instead of the subroutines heretofore necessary.

The addition of automatic floating-point operations can increase the computing power of the 1620 System 50 to 100 per cent, depending on the amount of floating-point computations required. In addition, up to 15 per cent of the basic 1620 core storage capacity can be saved through the elimination of subroutines and call sequence instructions associated with floating add, floating subtract, floating multiply, and floating divide.

Automatic Division (special feature) is a prerequisite to the installation of floating-point operations.

Figure 1. IBM 1620 Data Processing System

DESCRIPTION

Floating-Point Arithmetic

Scientific and engineering computations frequently involve lengthy and complex calculations in which it is necessary to manipulate numbers that may vary widely in magnitude. To obtain a meaningful answer, problems of this type usually require that as many significant digits as possible be retained during calculation and that the decimal point always be properly located. When applying such problems to a computer, several factors must be taken into consideration, the most important of which is the decimal point location.

Generally speaking, a computer does not recognize the decimal point present in any quantity used during the calculation. Thus, a product of 414154 will result regardless of whether the factors are 9.37 x 44.2, 93.7 x .442, or 937 x 4.42, etc. It is the programmer's responsibility to be cognizant of the decimal point location during and after the calculation and to arrange the program accordingly. In the operation of addition, for example, the decimal point of all numbers must be lined up to obtain the correct sum. To facilitate this arrangement, the programmer must shift the quantities as they are added. In the manipulation of numbers that vary greatly in magnitude, the resulting quantity could conceivably exceed allowable working limits.

The processing of numbers expressed in ordinary form, e.g., 427.93456, 0.0009762, 5382, -623.147, 3.1415927, etc., can be accomplished on a computer only by extensive analysis to determine the size and range of intermediate and final results. This analysis and subsequent number scaling frequently takes longer than does the actual calculation. Furthermore, number scaling requires complete and accurate information as to the boundaries of all numbers that come into the computation (input, intermediate, output). Since it is not always possible to predict the size of all numbers in a given calculation, analysis and number scaling is sometimes impractical.

To alleviate this programming problem, a system must be employed in which information regarding the magnitude of all numbers accompanies the quantities in the calculation. Thus, if all numbers are represented in some standard, predetermined format which instructs the computer in an orderly and simple fashion as to the location of the decimal point, and if this representation is acceptable to the routine doing the calculation, then quantities which range from minute fractions having many decimal places to large whole numbers having many integer places can be handled. The arithmetic system most commonly used, in which all numbers are expressed in a format having the above feature, is called "floating-point arithmetic."

The notation used in floating-point arithmetic is basically an adaptation of the scientific notation widely used today. In scientific work, very large or very small numbers are expressed as a number, between one and ten, times a power of ten. Thus 427.93456 is written as 4.2793456×10^{2} and 0.0009762 as 9.762×10^{-4}. In the 1620 floating-point arithmetic system, the range of numbers is modified to extend between .1 and .99999999, that is, the decimal point of all numbers is placed to the left of the high-order (leftmost) nonzero digit. Hence, all quantities may be thought of as a decimal fraction times a power of ten (e.g., 427.93456 as $.42793456 \times 10^{3}$ and 0.0009762 as $.97620000 \times 10^{-3}$) where the fraction is called the mantissa, and the power of ten, used to indicate the number of places the decimal point was shifted, the exponent. In addition to the advantages inherent in scientific notation, the use of floating-point numbers during processing eliminates the necessity of analyzing the operations to determine the positioning of the decimal point in intermediate and final results, since the decimal point is always immediately to the left of the high-order nonzero digit in the mantissa.

1620 Automatic Floating Point Operations

In 1620 Automatic Floating-Point Operations, a floating-point number is a field consisting of a variable length mantissa and a 2-digit exponent. The exponent is in the two low-order positions of the field, and the mantissa is in the remaining high-order positions, as shown:

$\bar{M} \ldots M\bar{E}E$

The mantissa must have a minimum of two digits and can have any number to a maximum of 100 digits. However, when two fields are operands, i.e., quantities being added together, they must have mantissas of the same length. The extremity of the field is marked by a flag over the high-order digit.

The exponent is established on the premise that the mantissa is less than 1.0 and equal to or greater than 0.1. The exponent is always two digits and has a range of -99 to +99. It is defined by a flag over the high-order (tens) digit.

The mantissa and the exponent each have an algebraic sign represented by a flag over the units positions, if negative, and by none, if positive. A floating-point number with a negative mantissa and a negative exponent is represented as follows:

$\bar{M} \ldots \bar{M}\bar{E}\bar{E}$

Sign control of the results of all computations is maintained according to the standard rules of arithmetic operations.

OPERATION

In descriptions of instructions and operations, the following symbols are used for clarity and brevity:

Mp = mantissa of the field at the P address (P)
Mq = mantissa of the field at the Q address (Q)
Ep = exponent of the field at the P address
Eq = exponent of the field at the Q address
L = number of digits in the mantissa
d = Ep-Eq

In all floating-point numbers, the decimal point is assumed to be at the left of the high-order digit , which must not be zero. Such a number is referred to as "normalized." When a number has one or more high-order zeros, it is considered to be "unnormalized." An unnormalized number resulting from a floating-point computation is normalized automatically, but unnormalized terms are not recognized as such when entered as data. They will be processed, but correct results cannot be assured. Therefore, it is necessary that all data be entered in normalized form. For example, the number $\bar{0}6823494\bar{0}5$ should be entered as $\bar{6}823494\bar{0}4$, assuming the fixed-point number is 6823.494, and an 8-digit mantissa is required.

Instructions

Eight floating-point instructions are provided. Four are for arithmetic computations; floating add, floating subtract, floating multiply, and floating divide; three control field size and location: floating shift right, floating shift left, and transmit floating. The eighth instruction provides for branch and transmit in floating-point operations. All instructions are in the 1620 format of a 2-digit OP code, 5-digit P address, and 5-digit Q address.

With the exception of floating shift right and floating shift left, the P address and Q address of floating point fields are the low-order positions of the exponents.

Within the discussion of each instruction, the operation of the computer in aligning decimal points, normalizing results, etc., is described as an aid to the programmer or operator in checking program logic and computation results. These operations are automatic and need not be programmed. Of particular note is floating divide, which requires only one instruction; the dividend is positioned, division is accomplished, and the quotient is transmitted to the P field without further command.

In all formulas for execution time, time (T) is expressed in microseconds.

Floating Add (FADD-01)

Operation. Mq is added to Mp and the result replaces Mp. Mq and Eq are not altered in core storage.

Description. Dependent on L and the value of d, the appropriate field is shifted to align decimal points before addition is performed. If d = 0, no shift is made (Figure 2).

Core Storage Locations 01590←→01599				Instruction			Core Storage Locations 01590←→01599			
Before							After			
Mp	Ep	Mq	Eq	OP	P	Q	Mp	Ep	Mq	Eq
$\bar{1}$ 2 3	$\bar{0}$ 4	$\bar{7}$ 8 9	$\bar{0}$ 4	0 1	0 1 5 9 4	0 1 5 9 9	$\bar{9}$ 1 2	$\bar{0}$ 4	$\bar{7}$ 8 9	$\bar{0}$ 4

Figure 2. Addition without Mp or Mq Shift

If d is greater than zero and less than L, Mq is, in effect, shifted d positions to the right before being added to Mp. The number of low-order digits of Mq equal to d are truncated as the shift is made (see Figure 3). If d is less than zero and |d| (absolute

Core Storage Locations 01590←→01599				Instruction			Core Storage Locations 01590←→01599			
Before							After			
Mp	Ep	Mq	Eq	OP	P	Q	Mp	Ep	Mq	Eq
$\bar{1}$ 2 3	$\bar{0}$ 2	$\bar{7}$ 8 9	$\bar{0}$ 1	0 1	0 1 5 9 4	0 1 5 9 9	$\bar{2}$ 0 1	$\bar{0}$ 2	$\bar{7}$ 8 9	$\bar{0}$ 1

Figure 3. Mq Shifted Right to Align Decimal Points

value of d) is less than L, Mp is shifted |d| positions to the right before Mq is added to it. The number of low-order digits of Mp equal to |d| are truncated as the shift is made (see Figure 4). If d is plus and equal to or larger than L, Mp is above the range of Mq,

Core Storage Locations 01590←→01599				Instruction			Core Storage Locations 01590←→01599			
Before							After			
Mp	Ep	Mq	Eq	OP	P	Q	Mp	Ep	Mq	Eq
$\bar{1}$ 2 3	$\bar{0}$ 1	$\bar{7}$ 8 9	$\bar{0}$ 2	0 1	0 1 5 9 4	0 1 5 9 9	$\bar{8}$ 0 1	$\bar{0}$ 2	$\bar{7}$ 8 9	$\bar{0}$ 2

Figure 4. Mp Shifted Right to Align Decimal Points

and no addition is performed (Figure 5). If d is less than zero and |d| is equal to or

Core Storage Locations 01590 ⟷ 01599				Instruction			Core Storage Locations 01590 ⟷ 01599			
Before							After			
Mp	Ep	Mq	Eq	OP	P	Q	Mp	Ep	Mq	Eq
$\bar{1}$ 2 3	$\bar{0}$ 5	$\bar{7}$ 8 9	$\bar{0}$ 2	0 1	0 1 5 9 4	0 1 5 9 9	$\bar{1}$ 2 3	$\bar{0}$ 5	$\bar{7}$ 8 9	$\bar{0}$ 2

Figure 5. Mp Above Range of Mq

greater than L, Mq is above the range of Mp, and no addition is performed. Mq replaces Mp, and Eq replaces Ep (see Figure 6).

Core Storage Locations 01590 ⟷ 01599				Instruction			Core Storage Locations 01590 ⟷ 01599			
Before							After			
Mp	Ep	Mq	Eq	OP	P	Q	Mp	Ep	Mq	Eq
$\bar{1}$ 2 3	$\bar{0}$ $\bar{1}$	$\bar{7}$ 8 9	$\bar{0}$ 3	0 1	0 1 5 9 4	0 1 5 9 9	$\bar{7}$ 8 9	$\bar{0}$ 3	$\bar{7}$ 8 9	$\bar{0}$ 3

Figure 6. Mq Above Range of Mp

After addition has been completed, the number of Mp digits is checked to determine if it exceeds L. If so, this is an overflow condition; the low-order digit of Mp is truncated, and the mantissa is shifted one position to the right. A one is entered in the high-order position of the mantissa, and a one is added to Ep (see Figure 7). When an overflow does

Core Storage Locations 01590 ⟷ 01599				Instruction			Core Storage Locations 01590 ⟷ 01599			
Before							After			
Mp	Ep	Mq	Eq	OP	P	Q	Mp	Ep	Mq	Eq
$\bar{9}$ 8 7	$\bar{0}$ 4	$\bar{4}$ 5 6	$\bar{0}$ 4	0 1	0 1 5 9 4	0 1 5 9 9	$\bar{1}$ 4 4	$\bar{0}$ 5	$\bar{4}$ 5 6	$\bar{0}$ 4

Figure 7. Mantissa Overflow, Number Normalized

not exist, Mp is scanned for zeros, beginning with the high-order position. High-order zeros are counted (z), and Mp is shifted z positions to the left; vacated positions are set to zeros. Flag bits in Mp are not altered or moved. Ep-z replaces Ep (see Figure 8).

Execution Time. T (average) = 400 + 100L. If the result is recomplemented, add 80L.

Core Storage Locations 01590 ← → 01599				Instruction			Core Storage Locations 01590 ← → 01599			
Before							After			
Mp	Ep	Mq	Eq	OP	P	Q	Mp	Ep	Mq	Eq
$\bar{1}$ 2 3	$\bar{0}$ 1	$\bar{1}$ 1 $\bar{9}$	$\bar{0}$ 1	0 1	0 1 5 9 4	0 1 5 9 9	$\bar{4}$ 0 0	$\bar{0}$ $\bar{1}$	$\bar{1}$ 1 $\bar{9}$	$\bar{0}$ 1

Figure 8. High-Order Zeros, Number Normalized

Floating Subtract (FSUB-02)

Operation. Floating subtract operation is the same as floating add except that sign control procedures for Mq are reversed.

Floating Multiply (FMUL-03)

Operation. Mp is multiplied by Mq, and the result replaces Mp. Ep is added to Eq, and the sum replaces Ep. Mp and Ep are normalized, as required, after multiplication. Mq and Eq are not altered in core storage.

Description. The product is formed in the product area, beginning at 00099 and extending through lower numbered core storage positions to 00100-2L. The product area, 00080-00099, is cleared automatically prior to multiplication. If L is greater than 10, the program must provide for clearing positions 00100-2L through 00079. After multiplication, the digit at position 00100-2L is tested for zero. If the digit is other than a zero, the field at 00099-L replaces Mp (Figure 9). If the digit tested is a

Core Storage Locations 01590 ← → 01599				Instruction			Core Storage Locations 01590 ← → 01599			
Before							After			
Mp	Ep	Mq	Eq	OP	P	Q	Mp	Ep	Mq	Eq
$\bar{7}$ 8 9	$\bar{0}$ 3	$\bar{4}$ 5 6	$\bar{0}$ $\bar{1}$	0 3	0 1 5 9 4	0 1 5 9 9	$\bar{3}$ 5 9	$\bar{0}$ 2	$\bar{4}$ 5 6	$\bar{0}$ $\bar{1}$

Figure 9. Product Equal to 2L.

zero the field at 00100-L replaces Mp and Ep + Eq -1 replaces Ep (Figure 10).

Execution Time. T (average) = $1120 + 80L + 168L^2$

Core Storage Locations 01590 ← → 01599				Instruction			Core Storage Locations 01590 ← → 01599			
Before							After			
Mp	Ep	Mq	Eq	OP	P	Q	Mp	Ep	Mq	Eq
$\overline{1}$ 2 3	$\overline{0}$ 2	$\overline{4}$ 5 6	$\overline{0}$ 4	0 3	0 1 5 9 4	0 1 5 9 9	$\overline{5}$ 6 0	$\overline{0}$ 5	$\overline{4}$ 5 6	$\overline{0}$ 4

Figure 10. Product Less than 2L.

Floating Divide (FDIV-09)

Operation. Mp is divided by Mq, and the quotient replaces Mp. Eq is deducted from Ep, and the result replaces Ep. Mp and Ep are normalized as required, after division. Mq and Eq are not altered in core storage.

Description. The quotient and remainder are developed in the product area, beginning at 00099 and extending through lower numbered core storage positions to 00100-2L. The product area,00080-00099, is cleared automatically prior to division. If L is greater than 10, the program must provide for clearing positions 00100-2L through 00079. Prior to division, the absolute values of Mp and Mq are compared. If Mp is equal to or greater than Mq, Mp is transmitted to 00100-L, and division is performed, starting at 00100-L, according to the procedure for automatic division. The quotient at 00099-L replaces Mp, and Ep-Eq + 1 replaces Ep (see Figure 11). If

Core Storage Locations 01590 ← → 01599				Instruction			Core Storage Locations 01590 ← → 01599			
Before							After			
Mp	Ep	Mq	Eq	OP	P	Q	Mp	Ep	Mq	Eq
$\overline{7}$ 8 9	$\overline{0}$ 4	$\overline{1}$ 2 3	$\overline{0}$ 1	0 9	0 1 5 9 4	0 1 5 9 9	$\overline{6}$ 4 1	$\overline{0}$ 4	$\overline{1}$ 2 3	$\overline{0}$ 1

Figure 11. Divisor Equal to or Less than Dividend

Mp is less than Mq, Mp is transmitted to 00099-L; division starts in 00100-L, and proceeds according to the procedure for automatic division. The quotient at 00099-L replaces Mp, and Ep-Eq replaces Ep (see Figure 12).

Core Storage Locations 01590 ← → 01599				Instruction			Core Storage Locations 01590 ← → 01599			
Before							After			
Mp	Ep	Mq	Eq	OP	P	Q	Mp	Ep	Mq	Eq
$\overline{1}$ 2 3	$\overline{0}$ 1	$\overline{7}$ 8 9	$\overline{0}$ 4	0 9	0 1 5 9 4	0 1 5 9 9	$\overline{1}$ 5 5	$\overline{0}$ $\overline{3}$	$\overline{7}$ 8 9	$\overline{0}$ 4

Figure 12. Divisor Greater than Dividend

Division by zero causes the overflow check indicator (14) to be turned on. Mp is not altered, but Ep is replaced by Ep-Eq.

Execution Time. $T = 880 + 940L + 520L^2$. The formula is based on an average quotient digit of 4.5.

Floating Shift Right (FSR-08)

Operation. The field at the Q address (the portion of the mantissa to be retained) is shifted right, to the location specified by the P address. The exponent is not moved or altered.

Description. The effect of this instruction is to shrink the mantissa by shifting it to the right and truncating the low-order digits. The P address is normally the units position of the mantissa; the digit at the Q address becomes the new low-order digit of the mantissa. Vacated high-order positions are set to zeros. An existing flag bit at the P address is retained for algebraic sign; the field flag bit is transmitted with the high-order digit of the Q field and terminates the operation (see Figure 13).

Execution Time. $T = 200 + 40L$.

Core Storage Locations 01590 ← → 01599				Instruction			Core Storage Locations 01590 ← → 01599			
Before							After			
Mp	Ep	Mq	Eq	OP	P	Q	Mp	Ep	Mq	Eq
0 $\bar{1}$ 2	$\bar{0}$ 2	$\bar{7}$ 8 9	$\bar{0}$ 5	0 8	0 1 5 9 7	0 1 5 9 6	0 $\bar{1}$ 2	$\bar{0}$ 2	0 $\bar{7}$ 8	$\bar{0}$ 5

Figure 13. Floating Shift Right

Floating Shift Left (FSL-05)

Operation. The field at the Q address, which is the low-order position of the mantissa, is shifted left so that the high-order digit is moved to the location specified by the P address. The exponent is not moved or altered.

Description. The effect of this instruction is to expand the mantissa by shifting it to the left and filling the vacated positions with zeros. It is important to note that the Q address is the low-order position of the field moved, and the P address is the

high-order position of the resulting field. An existing flag bit at the Q address is retained for algebraic sign; the field flag bit is transmitted with the high-order digit of the Q field (see Figure 14).

Core Storage Locations 01590 ← → 01599				Instruction			Core Storage Locations 01590 ← → 01599			
Before							After			
Mp	Ep	Mq	Eq	OP	P	Q	Mp	Ep	Mq	Eq
$\bar{1}$ 2 3	$\bar{0}$ 2	0 $\bar{7}$ $\bar{8}$	$\bar{0}$ 5	0 5	0 1 5 9 5	0 1 5 9 7	$\bar{1}$ 2 3	$\bar{0}$ 2	$\bar{7}$ 8 $\bar{0}$	$\bar{0}$ 5

Figure 14. Floating Shift Left

If the mantissa is expanded to a length greater than 2L, any extraneous flag bits in core storage positions between the old high-order position and the new low-order position of the mantissa must be cleared before the FSL instruction is given. Therefore, if Q-P is equal to or greater than 2L, locations P+L through Q-L must be free of flags.

Contrary to other instructions in the floating point series, FSL is executed in the transmit record manner of transmitting individual digits in the high-order to low-order sequence. After the units digit has been transmitted, the remaining positions of the mantissa are set to zero, in ascending core storage location sequence. After each position is set to zero, the succeeding position is checked for a flag bit. (The flag for a negative mantissa is ignored in the zeroing operation.) When a flag bit is detected, the operation stops without altering the flag-bit position, which is assumed to be the high-order position of the exponent. Thus, a flag bit detected prior to the previous high-order position of the mantissa stops the operation and results in an incorrect mantissa.

For example, if P = 01590, Q = 01601, and L = 4, core storage locations 01590 through 01603, with an extraneous flag bit in 01596, appear as follows:

XXXXXXX$\bar{X}$X$\bar{M}$MMM$\bar{E}$E

After transfer of the mantissa, but before the zero-fill operation, the core storage locations appear as follows (note that the flag bit in 01598 has been cleared):

$\bar{M}$MMMMXX$\bar{X}$XMMMM$\bar{E}$E

Upon completion of the operation, the mantissa is incorrect, as follows:

$\bar{M}$MMMM00$\bar{X}$XMMMM$\bar{E}$E

If 01596 had not contained a flag bit, the mantissa would have been expanded correctly, as follows:

$\bar{M}$MMMM00000000$\bar{E}$E

Execution Time. T = 200 + 40L + 40L'. (L' = length mantissa is increased by shift.)

Transmit Floating (TFL-06)

Operation. The field at the Q address is transmitted to the location designated by the P address. Mq and Eq are not altered in core storage.

Description. The Q address is normally the low-order position of the exponent, and the operation is the same as the regular transmit field instruction (TF-26), except that flag bits in the three low-order positions are ignored as indications to terminate the transmittal. Beginning with the fourth low-order digit, a flag bit terminates the operation. All flag bits in the field are transmitted (see Figure 15).

Execution Time. T = 240 + 40L.

Core Storage Locations 01590 ← → 01599				Instruction			Core Storage Locations 01590 ← → 01599			
Before							After			
Mp	Ep	Mq	Eq	OP	P	Q	Mp	Ep	Mq	Eq
$\overline{1}$ 2 3	$\overline{0}$ 2	$\overline{7}$ 8 9	$\overline{0}$ 5	0 6	0 1 5 9 4	0 1 5 9 9	$\overline{7}$ 8 9	$\overline{0}$ 5	$\overline{7}$ 8 9	$\overline{0}$ 5

Figure 15. Transmit Floating

Branch and Transmit Floating (BTFL-07)

Operation. The address of the next instruction is saved in IR-2, and the field at the Q address is transmitted to the P address minus one. The instruction at the P address is the next one executed. Mq and Eq are not altered in core storage.

Description. The Q address is normally the low-order position of the exponent. The operation is the same as the regular branch and transmit instruction (BT-27), except that in the transmit function the three low-order position flags are ignored as indications to terminate the transmittal. Beginning with the fourth low-order position, a flag bit terminates the operation. All flag bits are transmitted.

Execution Time. T = 280 + 40L.

Zero Mantissa

When a floating-point computation results in a zero mantissa, a special floating-point zero is created in the form $\bar{0}0 \ldots 0\bar{9}\bar{9}$, which is the smallest positive quantity that can be represented (see Figure 16). A zero mantissa causes the equal/zero indicator (12) to be turned on.

Core Storage Locations 01590 ← → 01599				Instruction			Core Storage Locations 01590 ← → 01599			
Before							After			
Mp	Ep	Mq	Eq	OP	P	Q	Mp	Ep	Mq	Eq
$\bar{7}$ 8 9	$\bar{0}$ 5	$\bar{7}$ 8 9	$\bar{0}$ 5	0 2	0 1 5 9 4	0 1 5 9 9	$\bar{0}$ 0 0	$\bar{9}$ $\bar{9}$	$\bar{7}$ 8 9	$\bar{0}$ 5

Figure 16. Zero Mantissa

Zeros entered as data should be in the floating-point zero form. Zero quantities in other forms, e.g., $\bar{0}0 \ldots 0\bar{0}\bar{0}$, will be processed, but results cannot be assured.

Indicators

The four indicators associated with automatic floating-point operations are represented by lights on the 1620 console. The light for each indicator is turned on when the corresponding indicator is turned on. The high/positive and equal/zero lights are located in the Control Gates section of the console, and the arithmetic overflow check and exponent check lights and switch are in the Indicator Displays and Switches section (see Figure 17).

High/Positive (11)

The high/positive indicator and light are turned on when the mantissa resulting from a floating-point computation is greater than zero.

Equal/Zero (12)

The equal/zero indicator and light are turned on to indicate a zero mantissa resulting from a floating-point computation.

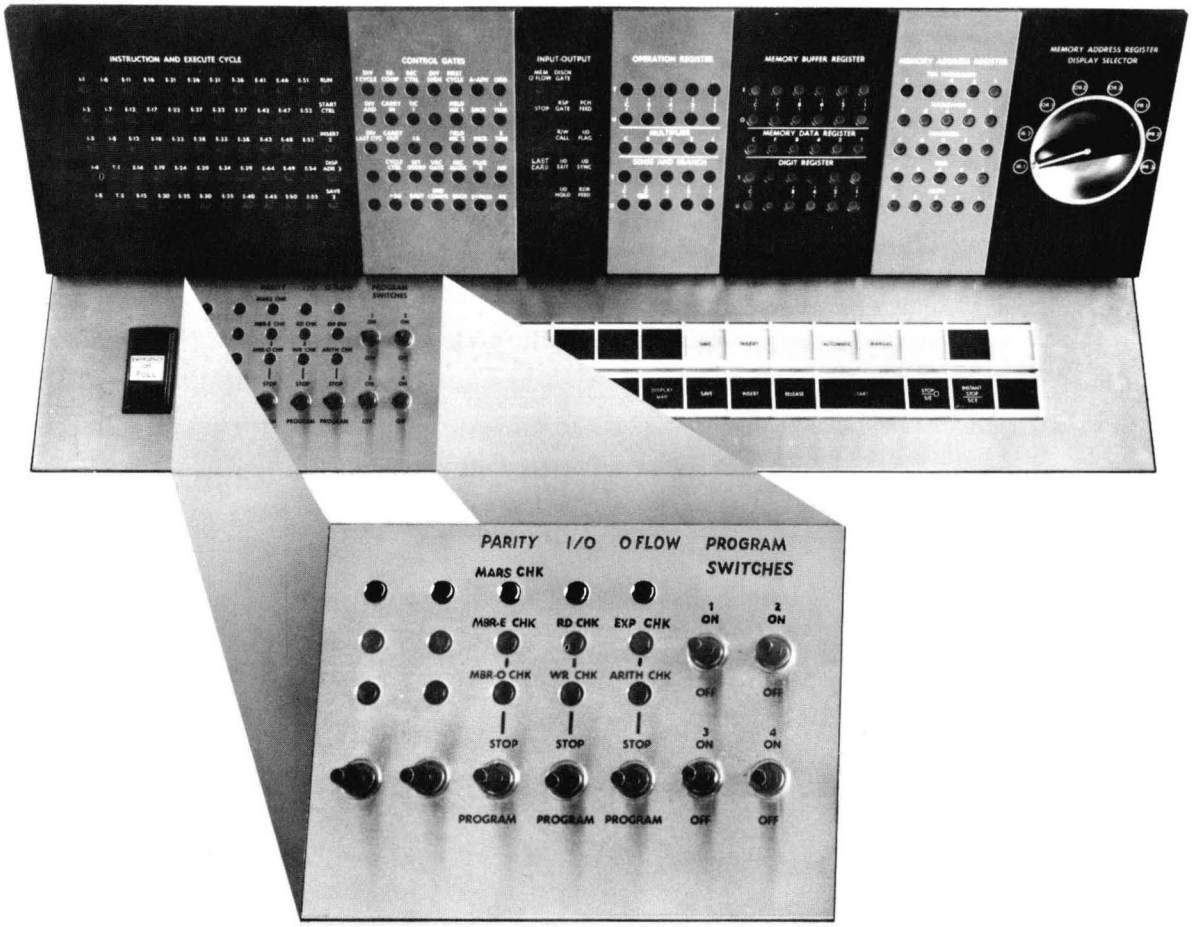

Figure 17. Indicators and Switch on 1620 Console

Arithmetic Overflow Check (14)

During floating-point operations, the overflow check indicator is turned on when division is attempted by zero. Division by an unnormalized number may result in an incorrect quotient through incorrect positioning of the divisor.

Exponent Check (15)

The exponent check indicator is turned on by exponent overflow or underflow.

Exponent Overflow. When an exponent greater than +99 is generated, the mantissa is set to nines. The sign is determined by the computed result that caused the overflow. The exponent is set to +99. This is the largest floating-point number ($\overline{99}$ . . . . $\overline{999}$) that can be represented. If the generated mantissa is positive, the H/P indicator (11) is also turned on.

Exponent Underflow. When an exponent less than -99 is generated, the mantissa is set to plus zeros, and the exponent is set to -99. This is the smallest floating-point number ($\overline{0}0 \ldots 0\overline{99}$) that can be represented. The E/Z indicator is also turned on.

An exponent underflow is not indicated when one or both operands are zero.

When the exponent check indicator (15) is turned on, program operation is controlled by the console overflow check switch, which is also connected to the overflow check indicator (14). Functions of the console switches are described in the 1620 Reference Manual. The exponent check indicator (15) is turned off by programmed interrogation or by depression of the 1620 reset key.

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y.

Printed in U.S.A. G26-5595-0 9/61