
olivetti programma 101
V W Y Z

The OLIVETTI Programma 101 desk calculator[1]

The Programma 101 is manufactured by the Olivetti Underwood Corporation. The cost of Programma 101 is about $3,500 (in 1968). Several thousand are currently in use. Unlike conventional stored program computers it has instructions which can be executed directly as commands from a keyboard or instructions which can be stored in a program and interpreted by the processor. The processor uses the decimal representation for mixed numbers. The decimal point location is controlled manually. Although information is stored in character strings, the maximum length is 22 digits or 24 instructions for a register. A program can be up to 120 characters long and is stored as a continuous string. The internal encoding of a character is 8 bits. There are no absolute addresses for instructions, and jump instructions are programmed by placing labels or references in the string to transfer to. The Programma 101 is composed of the following elements.

Memory. The memory stores numeric data and program instructions.

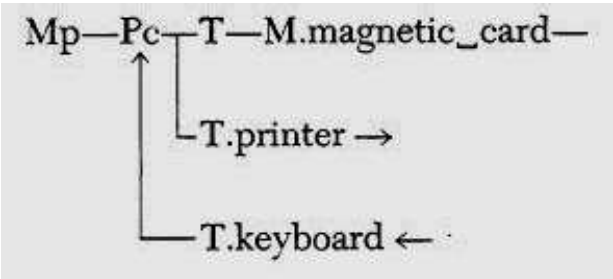
Keyboard. The keyboard has four functions: It is used for operator control of the calculator (power on, off, etc); in manual mode the instructions are executed immediately as in a conventional desk calculator (e.g., add); the keys write a program's instructions in the memory, and the instructions are executed when the program is run; and numeric data may be entered to a running program.

Printing unit. Serial printing is from right to left, at 30 characters per second; this unit prints all keyboard entries, programmed output, and instructions.

Magnetic-card reader/recorder. This device permits instructions and constants for a program to be stored and retrieved from magnetic cards.

Control and arithmetic units. The *control unit* is the administrative section of the computer. It receives the incoming information, determines the computation to be performed, and directs the *arithmetic unit* where to find the information and what operation to perform.

The PMS diagram shown below is, of course, very simple. It conforms closely to the classic diagram of what a digital computer looks like:



[1] The description is partially taken from the Programma 101 Programming Manual.

Primary memory and processor memory

The memory has 10 registers; eight are for general storage and two are used exclusively for instructions. A character can have several meanings, depending on the register and its use.

The two instruction registers, 1 and 2, each store 24 instructions. An instruction is one character long.

The eight storage registers, M, A, R, B, C, D, E, and F, have a capacity of 22 decimal digits, plus decimal point and sign. The sign and decimal point do not require character space. Alternatively, D, E and F hold 24 instructions. M, A, and R are operating registers and take part in all arithmetic operations. They are considered to be the arithmetic unit.

The M register is the Median (or distributive) register. All keyboard figure entries are held in the M register and distributed to the other registers as instructed.

The A register functions with the arithmetic unit to form the Accumulator. Arithmetic results are developed and retained in the A register. A result of up to 23 digits can be produced in the A register.

The R register retains the complete results in addition and subtraction, the complete product in multiplication, the remainder in division, and a remainder in square root. B, C, D, E, and F are storage registers. Each can be split into two registers, each with a capacity of 11 digits, plus decimal point and sign. When storage registers are split, the right portion of the split register retains its original designation, and the left side is identified with the corresponding lowercase letter. Thus these registers become b, B, c, C, d, D, e, F, f and F. The lowercase designation is obtained by first entering the corresponding uppercase letter and the depressing the "/" key, for example, c ≡ C/.

The registers D, E, and F or their splits have the additional capability of storing either instructions or constants to be used within programs. Thus they can store 1 signed 22-digit number, 2 signed 11-digit numbers, 1 signed 11-digit number, and 11 instructions, or 24 instructions. Programs of up to 120 instructions can be stored internally (Fig. 1). When registers D, E, and F and their splits are not used for instructions, they are free to store constants or intermediate results.

The relationship of memory, keyboard, printer and magnetic card is shown in Fig. 1. Registers are referenced explicitly. Programs do not use explicit addresses in instruction. Thus, special marker characters are placed in the instructions to serve as jump reference addresses (program labels).

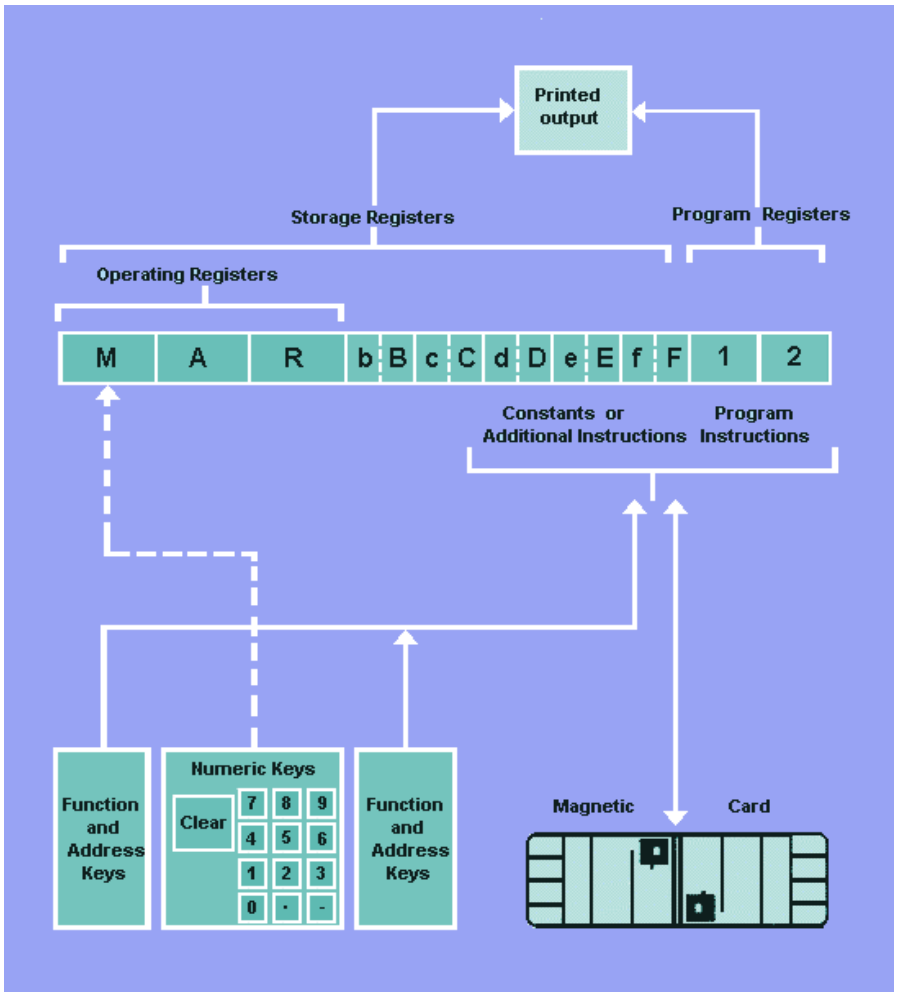


Fig. 1. Programma 101 functional block diagram. ***(Courtesy of Olivetti Underwood Corporation.)***

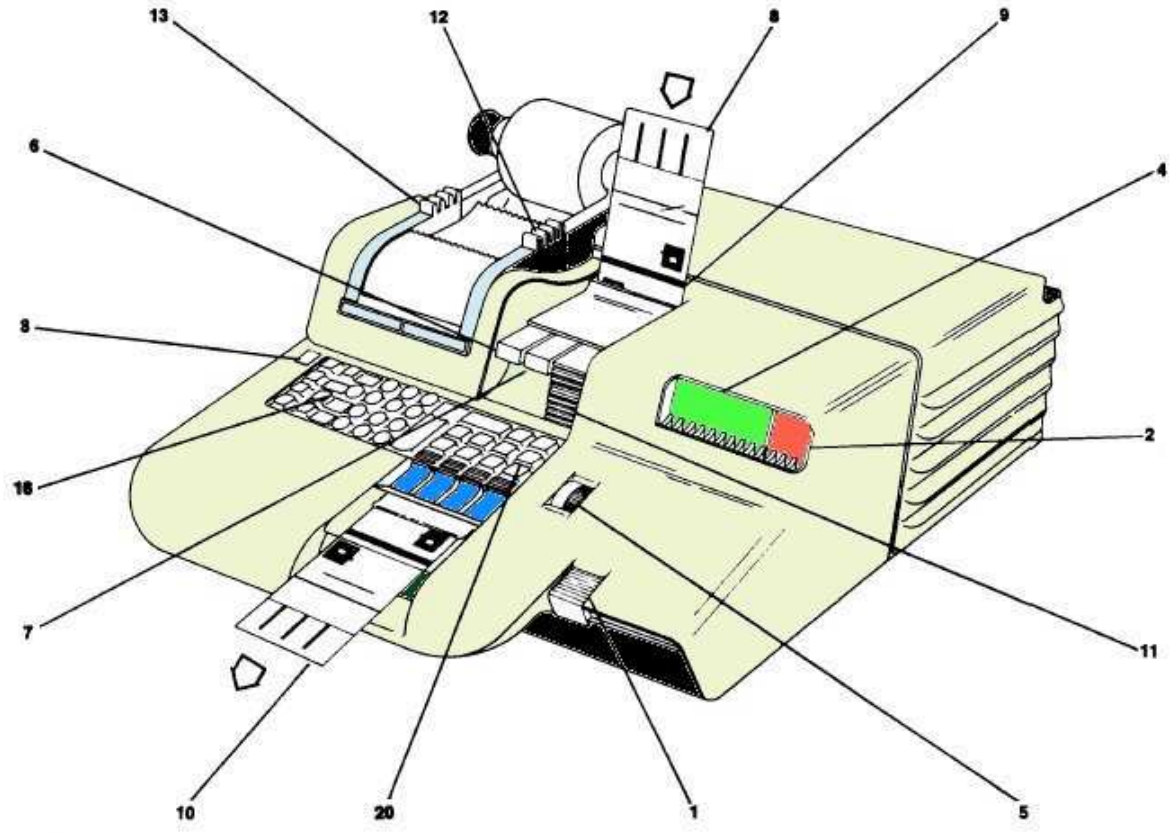


Fig. 2. Programma 101. ***(Courtesy of Olivetti Underwood Corporation.)***

Structure

The calculator parts are described briefly below. The parts correspond to both the numbers (Fig. 2) and the lettered keyboard (Fig. 3). The following parts are, in effect, the console. Some of the keys are used for control of the calculator, and some can be used either as programmed instructions or as commands which are executed directly. The following section discusses their instruction function.

The on-off key (1). This is a dual-purpose switch for both the on and off positions. (Note: The OFF position automatically clears all stored data and instructions.)

The error (red) light (2). This lights when the computer is turned on and whenever the computer detects an operational error, e.g., exceeding capacity, division by zero.

The general reset key (3). This key erases all data and instructions from the computer and turns off the error light.

The correct-performance (green) light (4). This light indicates the computer is functioning properly. A steady light indicates that the computer is ready for an operator decision; a flickering light indicates that the computer is executing programmed instructions and that the keyboard is locked.

The decimal wheel (5). This determines the number of decimal places (0, 1 .. 15) to which computations will be carried out in the A register and the decimal places in the printed output, except for results from the R register. Up to 22 decimal digits may be developed in, and printed from, the R register.

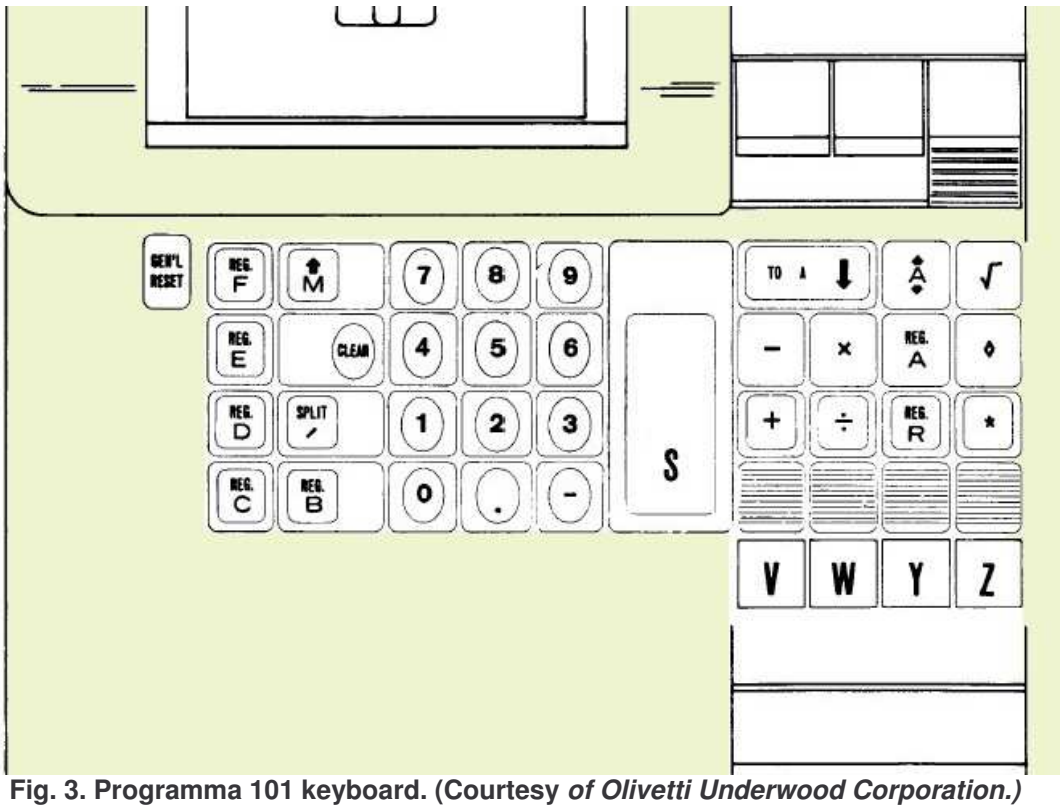


Fig. 3. Programma 101 keyboard. (Courtesy *of Olivetti Underwood Corporation.)*

The record program switch (6). When this switch is off, the commands pressed on the keyboard are executed directly. When this switch is **ON**, it directs the computer to store instructions either in the memory from the keyboard or onto a magnetic program card from the memory.
The record program switch must be **OFF** to load instructions from a magnetic program card into the memory.

The print program switch (7). When this switch is **ON** (in), it directs the computer to print out the instructions stored in memory from its present location in the program to the next Stop instruction (5), whenever the print key (20) is depressed.

The magnetic program card (8). This is a plastic card with a ferrous oxide backing, used to record programs for external storage. The card is inserted into a magnetic reader/writer (9) to record instructions and/or constants into or from the computer memory. Once inserted, the card may be removed from the computer (10) without disturbing the stored instructions.

(Note: The magnetic-card reader/writer uses only half the magnetic card at a time; consequently, two sets of 120 instructions and/or constants may be stored on a single card.)

The keyboard release key (11). This key reactivates a locked keyboard. If two or more keys are depressed simultaneously, the keyboard will lock to indicate a misoperation. Because the operator does not know what entry was accepted by the computer, after touching the keyboard release key, the clear entry key (16) must be depressed and the complete figure reentered.

Tape advance (12). This advances the printing paper tape.

Tape release lever (13). This enables adjustment when changing tape rolls.

The routine selection (keys ***V, W, Y,*** and **Z**). These keys direct the computer to the proper program or subroutine.

The numeric keyboard (keys **0***,* **1***,* **2***,* **3** , **4***,* **5***,* **6***,* **7***,* **8** , **9***,* **.** , **-**). This keyboard allows entry of a signed, mixed decimal number. Keyboard entries are automatically stored in the M register.

*The clear entry key (**CLEAR**).* This key clears the entire keyboard entry. When keying in the program, a depression of the clear key will erase the last instruction that has been entered into the memory. The printing tape will be spaced.

The start key (17). This key restarts the computer in programmed operation; it is used to code a stop instruction when keying in programs.

The register address (keys **A***,* **B***,* **C***,* **D***,* **E***,* **F***,* and **R**). These keys identify the corresponding registers. The operating register M has no keyboard

identification since the computer automatically relates all instructions to the M register unless otherwise instructed.

The split key (**/**). This key combined with a register (for example, C/) divides that register into two equal parts. When storage registers are split, the right portion of the split register retains the original designation, and the left side is identified on the tape with the corresponding lowercase letter (for example, C/ ≡ c).

The print key (◆). This key prints the contents of an addressed register.

The clear key (**✱**). This key clears the contents of an addressed register. When the computer is operated manually, a depression of this key will print the number in the register and clear it.

The transfer keys (🡻, 🡹 , ⇳). These keys perform transfer operations between the storage registers and the operating registers.

The arithmetic keys (**+, - , x, ÷** , √). These keys perform their indicated arithmetic function.

Keyboard and stored-program operations

All the following keys can be used as direct instructions (i.e., manually) if the record program switch is off. Alternatively, if the record program switch is on, the keys specify the instruction to be recorded in the program memory. Finally, the descriptions specify the instruction's behavior as it is executed within a program.

Start **S**. The instruction **S** (used in creating a program) directs the computer to stop and release the keyboard for the entry of figures or the selection of a subroutine. After figure entry, the program is restarted by touching the start key (S).
The program can also be restarted by touching a routine selection key. When the S instruction stops the program, the computer may also be operated in the manual mode without disturbing the program instructions in the memory. Any figures entered on the keyboard before depression of start or an operation key will be printed automatically.

Clear **✱**. The clear operation ***** directs the computer to clear the selected register. The M and R registers cannot be cleared with this instruction.
When the computer is operated manually this key will cause it to print the contents of the selected register, r. (r ← 0)

Data-transfer operations

- *To A* 🡻. An instruction containing the operation 🡻 directs the computer to transfer contents of the addressed register, r, to A while retaining them in

the original register. The contents of M and R are not affected. The previous contents of A are destroyed. (A ← r)

- *From M* ↑. An instruction containing the operation ↑ directs the computer to transfer the contents of M to the addressed register while retaining them in M. The contents of registers A and R are unaffected by this instruction. The original contents of the addressed register are destroyed. (r ← M)

- *Exchange* ⇳. An instruction containing the operation ⇳ directs the computer to exchange the contents of the A register with the contents of the addressed register. The contents of M are not affected except by the exchange between A and M. The contents of the R register are not affected. (A ← r; r ← A)

- *D-R exchange* **RS**. The instruction **RS** directs the computer to exchange the contents of D (both D and d registers) with the contents of the R register. (D ← R; R ← D). This instruction has a special use in multicard programs to store temporarily the contents of the D (d,D) register in R, when a new card has to be read to continue the program. During this temporary storage no instruction affecting the R register should be executed.

- *Decimal part to M* /⇳. The instruction /⇳ directs the computer to transfer the decimal portion of the contents of A to the M register while retaining the entire contents in A. The original contents of the M register are destroyed. The R register is not affected by this instruction. (M ←fraction_ part(A))

Arithmetic operations

All arithmetic operations are performed in the operating registers M, A, and R. An arithmetic operation is performed in two phases:

1 The contents of the selected register are automatically transferred to the M register. The M register is selected automatically if no other register is indicated.

2 The operation is carried out in the M, A, and R registers.

Programma 101 can perform these arithmetic operations: **+**, **-** , **X**, ÷ , √ , and absolute value. Figures are accepted and computed algebraically. A negative value is entered by depressing the negative key at any time during the entry of a figure. If there is no negative indication, the computer will accept the figure as positive.

The subtract operation key is separate from the numeric keyboard and is used exclusively for subtraction (not negation).

Addition +. An instruction containing the operation **+** directs the computer to add the contents of the selected register (addend) to the contents of the A register (augend). Addition is executed in two phases:

1 Transfer the contents of the selected register (addend) to M.

2 Add the contents of M to the contents of A (augend) obtaining in A the sum truncated according to the setting of the decimal wheel. The complete sum is in R. M contains the addend. (M ←r; next R ← A + M; next A ← f(R,decimal_ wheel))

Multiplication ***X***. An instruction containing the operation **x** directs the computer to multiply the contents of the selected register (multiplicand) by the contents of the A register (multiplier).

1 Transfer the contents of the addressed register to M.

2 Multiply the contents of M by the contents of A, obtaining in A the product truncated according to the setting of the decimal wheel. The complete product is in R. M contains the multiplicand. (M ← r; next R ← A x M; next A ← f(R, decimal_ wheel))

Subtraction **-** . An instruction containing the operation **-** directs the computer to subtract the contents of the selected register (subtrahend) from the contents of the A register (minuend).

1 Transfer the contents of the selected register (subtrahend) to M.

2 Subtract the contents of M from the contents of A (minuend), obtaining in A the difference truncated according to the setting of the decimal wheel. The complete difference is in R. M contains the subtrahend. (M ← r; next R ← A - M, next A ← f(R,decimal_ wheel))

Division ÷ . An instruction containing the operation ÷ directs the computer to divide the contents of the selected register (divisor) into the contents of the A register (dividend).

1 Transfer the contents of the addressed register to M.

2 Divide the contents of M into the contents of A, obtaining in A the quotient truncated according to the setting of the decimal wheel. The decimally correct fractional remainder is in R. M contains the divisor. (M ← r; next A ← A ÷ M; R← A mod M)

Square Root √. An instruction containing the operation √ directs the computer to:

1 Transfer the contents of the selected register to M.

2 Extract the square root of the contents of M, as an absolute value, obtaining in A the result truncated according to the setting of the decimal wheel. The R register contains a nonfunctional remainder. At the end of the operation, M contains double the square root. (M ← r; next M,R ← sqrt(abs(M)) x 2; next A ← f(M/2, decimal_ wheel))

Absolute Value **A⇳**. The absolute-value instruction **A⇳** changes the contents of the A register, if negative, to positive. (A ← abs(A))

Jump operations

The jump operation directs the computer to depart from the normal sequence of step-by-step instructions and jump to a preselected point in the program.
These instructions provide both internal and external (manual) decision capability and are useful to create "loops" that allow repetitive sequences in a program to be executed; routines or subroutines to be performed at the discretion of the operator; and automatically to "branch" to alternate routines or subroutines according to the value in the A register.

The jump process consists of two related instructions or characters:

1 The reference point or label, 1, is where the program begins or where the jump is to start. The sequence is restarted at this point. This label has no effect when interpreted.

2 The jump instruction specifies the label for the instruction sequence.

There are two types of jump instructions: unconditional jumps and conditional jumps.

Unconditional jumps. These jumps are executed whenever the instruction is read. The labels or reference points for unconditional jumps, L, and the corresponding jump instructions, j, are given as **(L,j)**. The permissible jump labels and jump constructions are:

(AV,V) (AW,W) (AY,Y) (AZ,Z) **(BV,CV) (BW,CW) (BY,CY) (BZ,CZ)**
(EV,DV) (EW,DW) (EY,DY) (EZ,DZ) **(FV,RV) (FW,RW) (FY,RY) (FZ,RZ)**

All programs must begin with reference parts of an unconditional jump instruction. Reference points AV, AW, AY, AZ are used so that these program sequences can be started by touching the routine selection keys V, W, Y, or Z.

Conditional Jumps. If the contents of the A register are:

Greater than zero: the program jumps to the corresponding reference point (label).

Zero or less: the program continues with the next instruction in sequence.

The labels or reference points for conditional jumps, L, and the corresponding conditional jump instruction, cj, are given as **(L,cj)**. The permissible jump labels and jump instructions are:

(aV,/V) (aW,/W) (aY,/Y) (aZ,/Z) **(bV,cV) (bW,cW) (bY,cY) (bZ,cZ)**
(eV,dV) (eW,dW) (eY,dY) (eZ,dZ) **(fV,rV) (fW,rW) (fY,rY) (fZ,rZ)**

Constants as instructions **A/↑**. A constant can be generated by a special instruction. The results of the instruction place the constant in M. Every digit of the constant (from the last digit to the first one) must follow **A/↑** and must be converted to instructions with the help of this chart:

Left part	*Right part*	
R Positive digit	**S**	0
D Positive hi-order digit	**↓**	1
F Negative digit	**↑**	2
E Negative hi-order digit	⇳	3
/ decimal point (included with units position of the integer)	**+**	4
	-	5
	x	6
	÷	7
	◆	8
	*****	9

***EXAMPLE*:** to generate 31.68 within a program the instructions would be:
A/↑ To start special series of instructions
R ◆ 8
R x 6
R/↓ 1.
D ⇳ 3 and to terminate special series of instructions.
Note that the value is generated starting with the least significant digit. The decimal locator (1) is included with the instruction creating the units position of the integer.

Instructions and data in the same register. An instruction can be considered to be data and, therefore, used as both a constant and an instruction. Another technique allows the computer to interpret data as null instructions so that both data (for reading and writing) and instructions can be stored in the same register.

Examples. A program to take values for the numbers A, B, C, and D from the keyboard and then print the value of the expression [(A + B) x C]/D would be written as follows:

instruction	*comments*
→AV	label to allow the program to be started by key, V
S	wait; enter A from keyboard into M
↓ or M↓[1]	A value goes to A register
S	wait, enter B from keyboard
+M	a register contains A + B
S	wait, enter C from keyboard
×M	a register ×C or (A + B) × C
S	wait, enter D from keyboard
÷M	a register has expression
A◇	print A register
V	jump back to beginning label to recalculate expression for new variables

[1]M is implied if left blank.

The following program computes and prints n!. n is entered from the keyboard, where n ≥ 1 and an integer. The program is started by pressing key Z.

instruction	*comments*
→ AZ	program start, label
S	stop, enter n from keyboard into M
D↑	D ← n; D holds n! or n × (n − 1) ×
M↓ (or ↓)	A ← n; A holds n, n − 1, n − 1, . . . , 1
→ AW	label
A/↑	generate 1 in M
D↓	
M− (or −)	A ← A − 1; (n ← n − 1)
/V	test if n ≥ 0
D◇	print result
Z	get next n from keyboard
→ aV	begin to update n!, label
D↕	A holds n!; D holds n − 1 after execution
D×	A holds n × (n − 1) ×
D↕	D holds n!; A holds n − 1 after execution
W	return to compute n − 2

Conclusion

Many algorithms have been written for Programma 101, being coded in impressively small space. The techniques have sometimes been borrowed from conventional computer programming. For example, multiple card programs operate by using chains in the same way as large FORTRAN programs. The significant fact to the reader is that the Programma 101 calculator is a nicely designed stored program computer.