Chapter 13

UNIVAC Scientific (1103A) instruction logic[1]

John W. Carr III

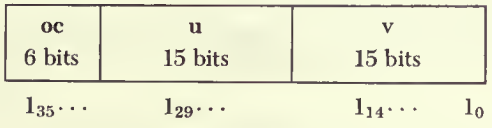
The UNIVAC Scientific computer is a (35, 0, 0)[2] binary machine, with option of (27, 8, 0). The arithmetic unit contains two 36-bit X (exchange) and Q (quotient) registers and one 72-bit A register (accumulator). Negative numbers are represented in one's complement notation.

Input-output is via high-speed paper tape reader and punch, direct card reader and punch, and Uniservo magnetic tape units, which may be connected to peripheral punched card readers and punches and a high-speed printer. In addition, information may be recorded on magnetic tape directly from keyboards by the use of Unitypers. Communication with external equipment is via an 8-bit (IOA) register and a 36-bit (IOB) register. Information sent to these registers controls magnetic tapes as well as other input-output equipment. The program address counter (PAK) contains the present instruction address. Storage is in up to 12,288 locations of magnetic core storage, along with a directly addressable drum of 16,384 locations. Instructions are of the two-address form, with six bits for the operation code and two fifteen-bit addresses (u and v).

The following information is taken from a Univac Scientific Manual [Univac Scientific Electronic Computing System Model 1103A, Form EL338].

Definitions and conventions

Instruction word



- oc Operation code
- u First execution address
- v Second execution address

For some of the instructions, the form jn or jk replaces the u address; for others the form k replaces the v address.

- j One-digit octal number modifying the instruction
- n Four-digit octal number designating number of times instruction is to be performed
- k Seven-digit binary number designating the number of places the word is to be shifted to the left

Address allocations (octal)

MC	00000–07777	4096
	00000–17777	8192 or
	00000–27777	12,288 36-bit words
Q	31000–31777	1 36-bit word
A	32000–37777	1 72-bit word
MD	40000–77777	16,384 36-bit words

Fixed addresses

- F_1 00000 or 40001
- F_2 00001
- F_3 00002
- F_4 00003

Arithmetic section registers

- A 72-bit accumulator with shifting properties
- A_R Right-hand 36 bits of A
- A_L Left-hand 36 bits of A
- Q 36-bit register with shifting properties
- X 36-bit exchange register

Note: Parentheses denote *contents of*. For example, (A) means contents of A (72-bit word in A); (Q) means contents of Q (36-bit word in Q).

[1] In E. M. Grabbe, S. Ramo, and D. E. Wooldridge (eds.), "Handbook of Automation, Computation, and Control," vol. 2, chap. 2, pp. 77–83, John Wiley & Sons, Inc., New York, 1959.

[2] Carr's triplet notation for: fractional significant digits, digits in exponent, and digits to left of radix point.

Input-output registers

IOA 8-bit in-out register
IOB 36-bit in-out register
TWR 6-bit typewriter register
HPR 7-bit high-speed punch register

Word extension

D(u) 72-bit word whose right-hand 36 bits are the word at address u, and whose left-hand 36 bits are the same as the leftmost bit of the word at u.
S(u) 72-bit word whose right-hand 36 bits are the word at address u, and whose left-hand 36 bits are zero.
D(Q) 72-bit word—right-hand 36 bits are in register Q, left-hand 36 bits are same as leftmost bit in register Q.
S(Q) same as D(Q) except left 36 bits are zero.
$D(A_R)$, $S(A_R)$ are similarly defined.
L(Q)(u) 72-bit word—left-hand 36 bits are zero, right-hand 36 bits are the bit-by-bit product of corresponding bits of (Q) and word at address u.
L(Q′)(v) 72-bit word—left-hand 36 bits are zero, right-hand 36 bits are the bit-by-bit product of corresponding bits of the complement of (Q) and word at address v.

Transmit instructions

11[1] **Transmit Positive TPuv**[2]**:** Replace (v) with (u).
13 **Transmit Negative TNuv:** Replace (v) with the complement of (u).
12 **Transmit Magnitude TMuv:** Replace (v) with the absolute magnitude of (u).
15 **Transmit U-address TUuv:** Replace the 15 bits of (v) designated by v_{15} through v_{29}, with the corresponding bits of (u), leaving the remaining 21 bits of (v) undisturbed.
16 **Transmit V-address TVuv:** Replace the right-hand 15 bits of (v) designated by v_0 through v_{14}, with the corresponding bits of (u), leaving the remaining 21 bits of (v) undisturbed.
35 **Add and Transmit ATuv:** Add D(u) to (A). Then replace (v) with (A_R).
36 **Subtract and Transmit STuv:** Subtract D(u) from (A). Then replace (v) with (A_R).
22 **Left Transmit LTjkv:** Left circular shift (A) by k places. If j = 0 replace (v) with (A_L); if j = 1 replace (v) with (A_R).

[1]Octal notation.
[2]Mnemonic notation.

Q-controlled instructions

51 **Q-controlled Transmit QTuv:** Form in A the number L(Q)(u). Then replace (v) by (A_R).
52 **Q-controlled Add QAuv:** Add to (A) the number L(Q)(u). Then replace (v) by (A_R).
53 **Q-controlled Substitute QSuv:** Form in A the quantity L(Q)(u) plus L(Q′)(v). Then replace (v) with (A_R). The effect is to replace selected bits of (v) with the corresponding bits of (u) in those places corresponding to 1's in Q. The final (v) is the same as the final (A_R).

Replace instructions

21 **Replace Add RAuv:** Form in A the sum of D(u) and D(v). Then replace (u) with (A_R).
23 **Replace Subtract RSuv:** Form in A the difference D(u) minus D(v). Then replace (u) with (A_R).
27 **Controlled Complement CCuv:** Replace (A_R) with (u) leaving (A_L) undisturbed. Then complement those bits of (A_R) that correspond to ones in (v). Then replace (u) with (A_R).
54 **Left Shift in A LAuk:** Replace (A) with D(u). Then left circular shift (A) by k places. Then replace (u) with (A_R). If u = A, the first step is omitted, so that the initial content of A is shifted.
55 **Left Shift in Q LQuk:** Replace (Q) with (u). Then left circular shift (Q) by k places. Then replace (u) with (Q).

Split instructions

31 **Split Positive Entry SPuk:** Form S(u) in A. Then left circular shift (A) by k places.
33 **Split Negative Entry SNuk:** Form in A the complement of S(u). Then left circular shift (A) by k places.
32 **Split Add SAuk:** Add S(u) to (A). Then left circular shift (A) by k places.
34 **Split Subtract SSuk:** Subtract S(u) from (A). Then left circular shift (A) by k places.

Two-way conditional jump instructions

46 **Sign Jump SJuv:** If $A_{71} = 1$, take (u) as NI. If $A_{71} = 0$, take (v) as NI. (NI means next instruction.)
47 **Zero Jump ZJuv:** If (A) is not zero, take (u) as NI. If (A) is zero, take (v) as NI.

44 **Q-Jump QJuv:** If $Q_{35} = 1$, take (u) as NI. If $Q_{35} = 0$, take (v) as NI. Then, in either case, left circular shift (Q) by one place.

One-way conditional jump instructions

41 **Index Jump IJuv:** Form in A the difference D(u) minus 1. Then if $A_{71} = 1$, continue the present sequence of instructions; if $A_{71} = 0$, replace (u) with (A_R) and take (v) as NI.

42 **Threshold Jump TJuv:** If D(u) is greater than (A), take (v) as NI; if not, continue the present sequence. In either case, leave (A) in its initial state.

43 **Equality Jump EJuv:** If D(u) equals (A), take (v) as NI, if not, continue the present sequence. In either case leave (A) in its initial state.

One-way unconditional jump instructions

45 **Manually Selective Jump MJjv:** If the number j is zero, take (v) as NI. If j is 1, 2, or 3, and the correspondingly numbered MJ selecting switch is set to "jump," take (v) as NI; if this switch is not set to "jump," continue the present sequence.

37 **Return Jump RJuv:** Let y represent the address from which CI was obtained. Replace the right-hand 15 bits of (u) with the quantity y plus 1. Then take (v) as NI.

14 **Interpret IP:** Let y represent the address from which CI was obtained. Replace the right-hand 15 bits of (F_1) with the quantity y + 1. Then take (F_2) as NI.

Stop instructions

56 **Manually Selective Stop MSjv:** If j = 0, stop computer operation and provide suitable indication. If j = 1, 2, or 3 and the correspondingly numbered MS selecting switch is set to "stop," stop computer operation and provide suitable indication. Whether or not a stop occurs, (v) is NI.

57 **Program Stop PS**—Stop computer operations and provide suitable indication.

External equipment instructions

17 **External Function EF-v:** Select a unit of external equipment and perform the function designated by (v).

76 **External Read ERjv:** If j = 0, replace the right-hand 8 bits of (v) with (IOA); if j = 1, replace (v) with (IOB).

77 **External Write EWjv:** If j = 0, replace (IOA) with the right-hand 8 bits of (v); if j = 1, replace (IOB) with (v). Cause the previously selected unit to respond to the information in IOA or IOB.

61 **PRint PR-v:** Replace (TWR) with the right-hand 6 bits of (v). Cause the typewriter to print the character corresponding to the 6-bit code.

63 **PUnch PUjv:** Replace (HPR) with the right-hand 6 bits of (v). Cause the punch to respond to (HPR). If j = 0, omit seventh level hole; if j = 1, include seventh level hole.

Arithmetic instructions

71 **Multiply MPuv:** Form in A the 72-bit product of (u) and (v), leaving in Q the multiplier (u).

72 **Multiply Add MAuv:** Add to (A) the 72-bit product of (u) and (v), leaving in Q the multiplier (u).

73 **Divide DVuv:** Divide the 72-bit number (A) by (u), putting the quotient in Q, and leaving in A a non-negative remainder R. Then replace (v) by (Q). The quotient and remainder are defined by: $(A)_i = (u) \cdot (Q) + R$, where $0 \leqq R < |(u)|$. Here $(A)_i$ denotes the initial contents of A.

74 **Scale Factor SFuv:** Replace (A) with D(u). Then left circular shift (A) by 36 places. Then continue to shift (A) until $A_{34} \neq A_{35}$. Then replace the right-hand 15 bits of (v) with the number of left circular shifts, k, which would be necessary to return (A) to its original position. If (A) is all ones or zeros, k = 37. If u is A, (A) is left unchanged in the first step, instead of being replaced by $D(A_R)$.

Sequenced instructions

75 **RePeat RPjnw:** This instruction calls for the next instruction, which will be called NIuv, to be executed n times, its u and v addresses being modified or not according to the value of j. Afterwards the program is continued by the execution of the instruction stored at a fixed address F_1. The exact steps carried out are:

a Replace the right-hand 15 bits of (F_1) with the address w.

b Execute NIuv, the next instruction in the program, n times.

c If j = 0, do not change u and v.
If j = 1, add one to v after each execution.
If j = 2, add one to u after each execution.
If j = 3, add one to u and v after each execution.

The modification of the u address and v address is done in program control registers. The original form of the instruction in storage is unaltered.

d On completing n executions, take (F_1), as the next instruction. F_1 normally contains a manually selective jump whereby the computer is sent to w for the next instruction after the repeat.

e If the repeated instruction is a jump instruction, the occurrence of a jump terminates the repetition. If the instruction is a Threshold Jump or an Equality Jump, and the jump to address v occurs, (Q) is replaced by the quantity j, (n − r), where r is the number of executions that have taken place.

Floating point instructions

64 **Add FAuv:** Form in Q the normalized rounded packed floating point sum (u) + (v).

65 **Subtract FSuv:** Form in Q the normalized rounded packed floating point difference (u) − (v).

66 **Multiply FMuv:** Form in Q the normalized rounded packed floating point product (u) · (v).

67 **Divide FDuv:** Form in Q the normalized rounded packed floating point quotient (u) ÷ (v).

01 **Polynomial Multiply FPuv:** Floating add (v) to the floating product $(Q)_i \cdot$ (u), leaving the packed normalized rounded result in Q.

02 **Inner Product FIuv:** Floating add to $(Q)_i$ the floating product (u) · (v) and store the rounded normalized packed result in Q. This instruction uses MC location $F_4 = 00003$ for temporary storage, where $(F_4)_f = (Q)_i$. The subscripts i and f represent "initial" and "final."

03 **Unpack UPuv:** Unpack (u), replacing (u) with $(u)_M$ and replacing $(v)_C$ with $(u)_C$ or its complement if (u) is negative. The characteristic portion of $(u)_f$ contains sign bits. The sign portion and mantissa portion of $(v)_f$ are set to zero. *Note.* The subscripts M and C denote the mantissa and characteristic portions.

04 **Normalize Pack NPuv:** Replace (u) with the normalized rounded packed floating point number obtained from the possibly unnormalized mantissa in $(u)_i$ and the biased characteristic in $(v)_c$. *Note.* It is assumed that $(u)_i$ has the binary point between u_{27} and u_{26}; that is, that $(u)_i$ is scaled by 2^{-27}.

05 **Normalize Exit NEj-:** If j = 1 normalize without rounding until a master clear or until the instruction is again executed with j = 0.

References

Univac Scientific Electronic Computing System Model 1103A, Form EL 338