19 January 1949

Seminar on Digital Computing

REFERENCE LIST NO. 1

✓ 1. "Summary of Characteristics: Magnetic Drum Binary Computer". Proposed for NBS. ERA Publication No. 23 (30 November 1948) Available in ERA Library on 3-day loan basis.

2. "Proceedings of a Symposium on Large-Scale Digital Calculating Machinery". Held January 1947 at Harvard University; published 1948.

3. Bloch, et al., "The Logical Design of the Raytheon Computer," MTAC 3,286 (October 1948).

4. West and DeTurk, "A Digital Computer for Scientific Applications", Proc. I.R.E. 36, 1452 (December 1948).

5. Alt, "A Bell Telephone Laboratories' Computing Machine", MTAC 3, 1 (January 1948) and 3, 69 (April 1948).

6. Huskey, "The Status of High-Speed Digital Computing Systems", Mechanical Engineering 70, 975 (December 1948).

7. Burks, "Electronic Computing Circuits of the ENIAC", Proc. I.R.E. 35, 756 (August 1947).

8. Project Whirlwind, Summary Report No. 2, in 22 volumes. Volume 4 is entitled "Introductory Material".

9. Burks, et al., "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument", Part I (second edition, 2 September 1947; first edition, 28 June 1946). Inst. for Adv. Study.

10. Goldstine and Von Neumann, "Planning and Coding of Problems, etc.", Part II Vol I (1 April 1947); also Part II, Vol II (15 April 1948). Inst. for Adv. Study.

11. "A Manual of Operation for the Automatic Sequence Controlled Calculator", Harvard (1946).

* * * * * * * * * * * * * * * * *

Your attention is directed to the quarterly publication, Mathematical Tables and Other Aids to Computation (abbreviated MTAC). Each issue has a section containing news and abstracts of articles in the field of Automatic Computing Machines, in addition to feature articles on this subject. Our library's file begins with the January, 1947, issue. Thumbing through the last seven or eight issues will give you a good bird's-eye view of the field today.

XA-275

COMMON SYSTEMS OF DIGITAL MACHINE NOTATION (ILLUSTRATED FOR 4-PLACE NUMBERS)

RADIX r	10		2	
KIND OF COMPLEMENT SYSTEM	10's	9's	2's	1's
MODULUS M	10^4	10^4-1	2^4	2^4-1
HOW MANY DIFFERENT NUMBERS EXPRESSIBLE?	10^4	10^4-1	2^4	2^4-1
MOST POS. EXPRESSIBLE NUMBER:				
REPRESENTATION A^*	4999	4999	0111	0111
VALUE A	+4999	+4999	+0111	+0111
MOST NEG. EXPRESSIBLE NUMBER:				
REPRESENTATION A^*	5000	5000	1000	1000
VALUE A	-5000	-4999	-1000	-0111
IN AN ADDITIVE MACHINE:				
END-AROUND CARRY?	NO	YES	NO	YES
ZERO GENERATED BY MACHINE	0000	9999	0000	1111
IN A SUBTRACTIVE MACHINE:				
END-AROUND BORROW?	NO	YES	NO	YES
ZERO GENERATED BY MACHINE	0000	0000	0000	0000
RELATION BETWEEN VALUE A & REPRESENTATION A*: $A = A^* - \delta \cdot M$, WHERE { $\delta = 0$ FOR $\lambda =$ $\delta = 1$ FOR $\lambda =$ AND λ = LEFTMOST, OR "SIGN", DIGIT OF A^*	0, 1, 2, 3, 4 5, 6, 7, 8, 9		0 1	
RELATION BETWEEN REPRESENTIONS OF A & -A	$A^* + (-A)^* = M$			

PREP. BY

[illegible]

SEMINAR MAT'L
2 FEB 1949

Seminar on Digital Computing 2 Feb 1949

PROBLEMS IN NUMBER NOTATION

1. Add two more columns to the DIGITAL MACHINE NOTATION table, for 4-digit octal (radix 8) integers, in 8's complements and in 7's complements.

2. Plot A vs. A^* for 4-digit binary integers expressed in 2's complements and in 1's complements.

3. Plot A vs. A^* for 2-digit decimal integers expressed in 10's complements and in 9's complements.

4. Plot A vs. A^* for 2-digit octal integers expressed in 8's complements and in 7's complements.

5. Posted in every drafting room is a table of decimal equivalents. Make up a table of "octal equivalents" of fractions of an inch from 1/64 to 1 inch, by 64ths.

6. Convert 849.163 (decimal) to binary notation. Convert this same number to octal notation. Repeat, using several different methods.

7. Convert 763.102 (octal) to binary notation, employing several different methods.

PROBLEM IN PROGRAMMING

This problem is the evaluation and tabulation of a polynomial over a range of values of two independent variables.

Given: Constants A, B, C, D, E, F are stored at a, b, c, d, e, f, respectively.

Req'd: To evaluate the polynomial

$$P(x,z) = Ax^2 + Bxz + Cz^2 + Dx + Ez + F$$

for $x = 0, \pm 1, \pm 2, \ldots \pm 15$
and $z = 0, \pm 1, \pm 2, \ldots \pm 15$.

Store the P's in 961 consecutively addressed memory boxes. Assume that the coefficients are integers whose absolute values do not exceed, say, 2^{10}; this keeps the P's within range of the 30-digit memory boxes. Select numerical values for the coefficients if you wish. Utilize the ability of the machine to accumulate sums and sums of products. Coefficients need not be positive.

Binary Accumulator, mod (2^n-1), Summary of Numerical Properties

A. The following properties are common to the n-place additive accumulator with end-around carry and the n-place subtractive accumulator with end-around borrow:

1. Range of (2^n-1) different numbers.

2. Pos and neg numbers are expressed in the "1's-complement" system, in which:

   a. the neg of a number can be formed by subtracting the number from (2^n-1); i.e., by replacing 1's with 0's and vice versa; and

   b. the left-most digit of a positive number is 0 and the left-most digit of a negative number is 1.

3. Absolute value of the numbers represented may not exceed (2^{n-1}-1); e.g., +2047 and -2047 are the range limits in a 12-place system; +7 and -7 the range limits in a 4-place system.

4. Nature of left shift: The most useful form of shift is that in which shifting k places to the left is equivalent to multiplication by 2^k; i.e., adding the number in the accumulator to itself, repeating this process k times. The so-called "circular shift" has this property in the mod (2^n-1) system. The digit occupying the left-most place shifts around into the right-most place on each shift. (Note that a k-place <u>right</u> circular shift would <u>not</u> be equivalent to division by 2^k. What kind of shift would?)

B. The following properties are peculiar to the n-place additive accumulator (abbreviated AA) with end-around carry:

1. A number transmitted to AA is added to the number in AA.

2. To subtract a number from the number in AA, the complement of the incoming number must be transmitted to AA.

3. Of the two forms which zero may take in a mod (2^n-1) system, only the 111...1 form, or "neg zero" can be generated by AA. (Try to beat this on paper, in case you are not convinced.)

C. The following properties are peculiar to the n-place subtractive accumulator (abbreviated SA) with end-around borrow:

1. A number xmitted to SA is subtracted into the number in SA.

2. To add a number to the number in SA, the complement of the incoming number must be xmitted to SA.

3. Only the 000...0 form of zero, of "positive zero", can be generated by SA.

Seminar on Digital Computing . 30 March 1949

COMMENTS ON BINARY MULTIPLICATION

The following tabulation is a step-by-step analysis of what would take place in an arithmetic unit having an 8-place accumulator and a 4-place Q-register, during the automatic multiplication process. This abridged machine follows the multiply algorithm of the ERA-NBS Computer (ERA Publ. No. 23, Fig. 4), except that the size of A, Q, and X are reduced. The example shown is the multiplication of +5 (multiplicand in X) by -6 (multiplier in Q), with A initially clear. X contains 0101 thruout the process; A sees this as 0000 0101.

Step	A	Q	S	
Initial	0000 0000	1001	0	
Subtr X	1111 1010			(Negative multiplier) correction.
AL 1	1111 0101			First step, basic algorithm.
Add X	1111 1010			
QL 1		0011	1	
AL 1	1111 0101			
Don't add				
QL 1		0110	2	
AL 1	1110 1011			
Don't add				
QL 1		1100	3	
AL 1	1101 0111			
Add X	1101 1100			
QL 1		1001	4	Final step, basic algorithm.
Add X	1110 0001			Negative multiplier correction.

Result: Product, -30, in A; multiplier, -6, again in Q.

ADVANCE ASSIGNMENT

1. Become familiar with the automatic binary multiplication process by running through an analysis similar to the above for the various combinations of positive and negative multiplier and multiplicand. Be careful to start with numbers within machine range. Perform one analysis for the case where the number initially in A is not zero, and it is desired to follow the HOLD MULTIPLY, or MApy, command.

2. So that you may more efficiently follow next week's discussion of division, set up a step-by-step analysis of the divide algorithm, Fig. 5, for a few combinations of positive and negative quotient and divisor. Heed the comments on p. 22 of Publ. No. 23. Proofs will be given next time - until then, try the algorithm out to see if it works for your examples.

Reference on non-restoring vs. restoring division: "Prelim discussion of logical design of an elec comp inst", Burks, et al, 2d ed (2 Sep '47), p. 23, ff.

Seminar on Digital Computing 6 April 1949

NOTES ON MACHINE DIVISION

Preliminary Considerations

Quotient Q and remainder R are related to dividend N (numerator) and divisor D (denominator) by:

$$N = Q \cdot D + R \quad , \quad \frac{N}{D} = Q + \frac{R}{D} \tag{1}$$

Several cases are of interest to the present discussion:

Case I. Here, (1) is restricted only by the requirement that

$$0 \leq |R| \leq |D|. \tag{2}$$

Question: This limits (1) to how many sets of Q and R, for a given N and D?

Case II. Here, (2) is retained, but in addition suppose Q is restricted to odd integers (because, say, of a prpperty of some machine process). Then Q will have the same sign as N/D; R may take on either sign.

Case III. Here, (1) is restricted only by the requirement of non-negative remainder whose absolute value is less than that of divisor. This is the DIVIDE opeation defined on p. 19 of ERA Publ. No. 23.

$$0 \leq R < |D|. \tag{3}$$

Question: How many Q,R for given N,D?

Restoring and Non-restoring Division Illustrated

Decimal, restoring. Example: 1634 = (65)(25) + 9.

```
           0  6  5
  +0025)+1  6  3  4
         0  0. 0  0   Subtr largest multiple of 25 that will
        +1  6  3  4         not produce overcast, each time.
         1  5  0. 0
        +0  1  3  4
         0  1  2  5.
        +0  0  0  9   Remainder.
```

Decimal, non-restoring. Example: 1634 = (66)(25) - 16.

```
           +1 -4 +6 = 106 - 40 = 66
  +0025)+1  6  3  4   N,D same sign.
        -2  5. 0  0   Subtr until overcast has occurred.
        -0  8  6  6   R,D opp sign.
        +1  0  0. 0   Add until overcast has occurred.
        +0  1  3  4   R,D same sign.
        -0  1  5  0.  Subtract, etc.
        -0  0  1  6   This is R.
```

<u>Binary, non-restoring.</u> Example: 29 = 5·5 + 4.

```
                 +1 -1 -1 +1 -1   = 10010 - 01101 = 00101, Q.
     0101)0 0 0 1 1 1 0 1   N,D same sign.
          0 1 0 1.0 0 0 0   Subtr D, put +1 in Q.
          1 1 0 0 1 1 0 0   R,D opp sign.
          0 0 1 0 1.0 0 0   Add D, put -1 in Q.
          1 1 1 1 0 1 0 0   R,D opp sign.
          0 0 0 1 0 1.0 0   Add D, put -1 in Q.
          0 0 0 0 1 0 0 1   R,D same sign.
          0 0 0 0 1 0 1.0   Subtr D, put +1 in Q.
          1 1 1 1 1 1 1 0   R,D opp sign.
          0 0 0 0 0 1 0 1.  Add D, put -1 in Q.
          0 0 0 0 0 1 0 0   This is R.
```

The above process always produces an <u>odd</u> quotient, by virtue of the peculiar (+1, -1) number system used to express the quotient.

ERA-NBS Computer, Basic Divide Algorithm

Consider our DIVIDE algorithm, Fig 5. Let us first examine the "basic" portion of the flow diagram. This performs non-restoring binary division similar to the above example, except that 1's and 0's are (temporarily) put into Q in place of the required +1's and -1's, respectively. Except for a correction to take this substitution into account, the division process is actually all done <u>just prior</u> to the final "QL 1".

At this point the Q-register contains a pseudo-quotient P whose 1's represent +1's, but whose 0's indicate where -1's really should be. Suppose that the -1's had been kept track of in an imaginary register containing a number S composed of 1's and 0's such that the 1's in S represent the -1's; i.e., P and S are complementary to each other. The quotient we want is then (P - S). But since P and S are complements, this is the same as 2P. Therefore the required correction is simply to shift P one place to the left. Hence the final "QL 1".

At the end of the "basic algorithm" we have an odd quotient, as stated on p. 22. An odd number in our system of notation is characterized by inequality of the extreme right and left digits; i.e., $q_{39} \neq q_0$. (Exercises: (1) Verify this fact. (2) Analyze the steps in the algo by which these two digits come out opposite. (3) If they were to come out alike, what would this indicate?)

In other words, the "basic algorithm" performs the kind of division defined in Case II on the preceding page.

ERA-NBS Computer, Corrections Yielding Non-negative Remainder

It is required to perform the kind of division we have specified as Case III. The correction steps preceding and following the basic algorithm in Fig 5 accomplish this. These corrections will be justified in two stages. First, suppose that the basic algo has been followed, producing a quotient Q' and a remainder R' having the Case II properties:

$$N = Q'D + R', \qquad 0 \leq |R'| \leq |D|, \qquad Q' \text{ odd.} \qquad (4)$$

We have three quantities N, Q', R', each of which may be + or –; there are therefore 8 possible cases to deal with (the sign of D is determined by N and Q'). These cases are listed in Table I, together with the corrections necessary to insure a non-negative remainder. The sign of the correction to Q' is opposite to the sign of D. This adds a negative absolute D to $Q'D$ to compensate for the positive absolute D added to R'.

These corrections are satisfactory, except that adding and subtracting are called for in the Q-register. This is expensive, since full carry or borrow facilities would be required in Q. Note that the corrections for Cases 5 and 6 do not generate carry or borrow ("carrow") in Q. Cases 7 and 8 therefore present the only difficulty.

Fortunately, the problem has a reasonable solution. Note that Cases 7 and 8 correspond to negative N. The sign of N is known at the beginning of the division process. If N is negative, suppose we immediately make it more negative by an amount equal to the absolute value of D. This pre-loading of N (to give $N'' = N - |D|$) must be compensated for later by adding +1 or –1 to Q. Assuming we have pre-loaded N, we then have at the end of the basic algo:

$$N'' = Q''D + R'', \qquad 0 \leq |R''| \leq |D|, \qquad Q'' \text{ odd}, \qquad N'' < 0. \qquad (5)$$

3,4,7,8 The pre-loading step affects the 4 cases 1,2,5,6 for which N is negative. Table I is still satisfactory for Cases 1-4. Table II re-examines Cases 5-8, in terms of R'', N'', Q''. The net correction is composed of two parts: a negative remainder correction (same rules as in Table I), and a further correction to Q'' due to the pre-loading of N. Since N'' is too negative by an absolute D, the correction to $Q''D$ must be a positive absolute D; i.e., the correction to Q'' must be of the same sign as D.

The net corrections are now all seen to be feasible without carrow facilities in Q. If we make the right-hand (q_0) place of the Q-register a "unit binary counter", then 1's may be transmitted to Q on a simple AWC (add-without-carry) basis. The corrections may then be collapsed into 4 cases (1-2, 3-4, 5-6, 7-8), as shown in Fig 5.

A physically simpler way of correcting Q'' is to XMIT q_{29} TO q_0 instead of AWC 1 TO q_0, since a pair of shifting gates are already available for this purpose. Then the AWC 1 TO q_0 step within the basic part of the algo is made simply INSERT 1 IN q_0.

Exercises:

1. Tabulate the steps of the divide algo for several examples, assuming a machine with 4-digit numbers and 8-place accumulator. Be sure to test some of the peculiar cases like 0/0, 0/5, -3/0, etc.

2. Justify (or refute) the second paragraph on p. 22 ("Prior to division . . . ").

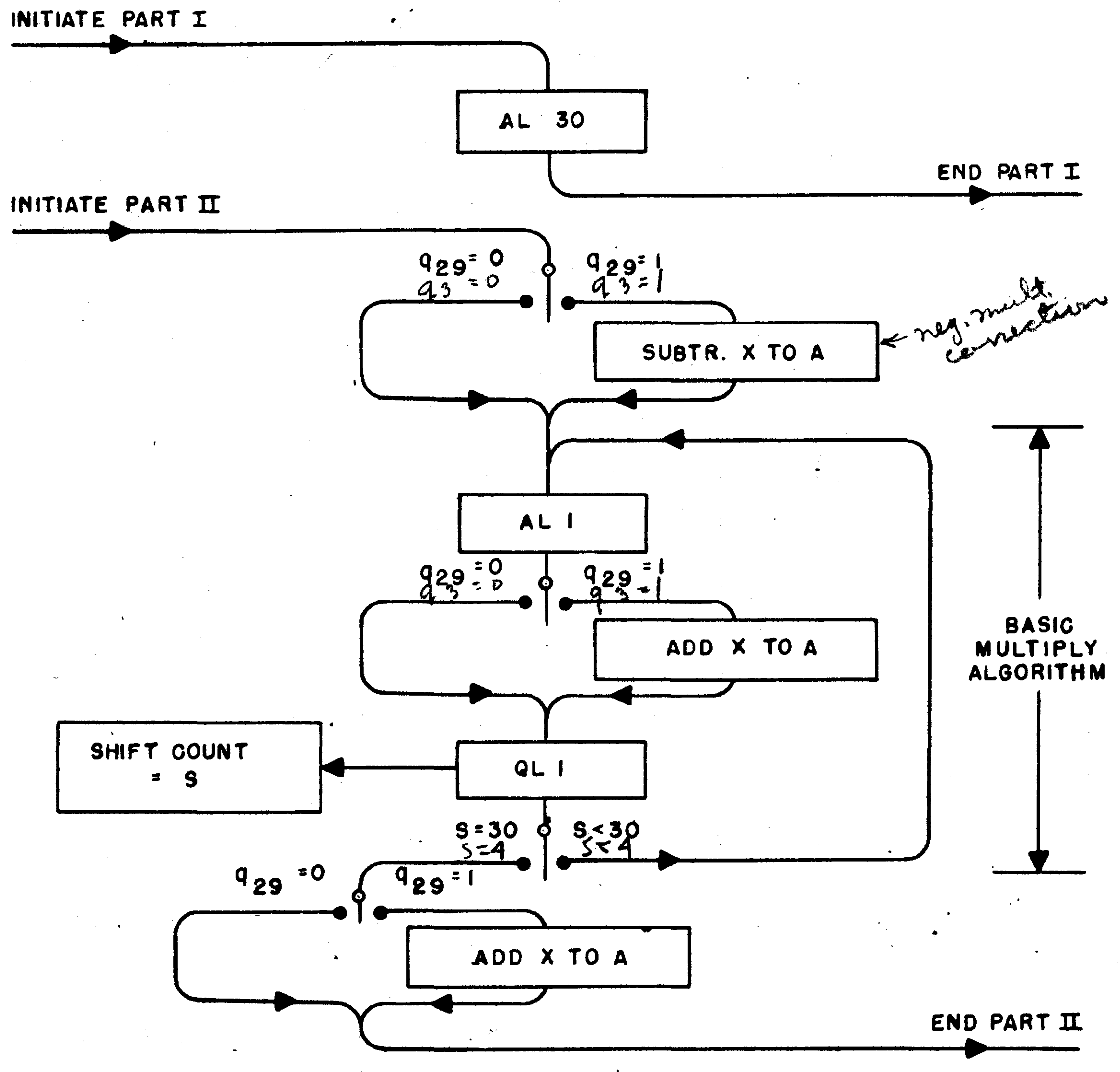
TABLE I

	Case	R'	N	Q'	D	q_0	Corr to R' for neg R'	Corr to Q' for neg R'	corrected R"
a	1	+	+	+	+	1	None	None	+
b	2	+	+	-	-	0	None	None	+
c	3	+	-	+	-	1	None	None	+
d	4	+	-	-	+	0	None	None	+
e	5	-	+	+	+	1	Abs add D	-1	+
f	6	-	+	-	-	0	Abs add D	+1	+
g	7	-	-	+	-	1	Abs add D	+1	+
h	8	-	-	-	+	0	Abs add D	-1	+

TABLE II

	Case	R"	N"	Q"	D	q_0	Corr to R" for neg R"	Corr to Q" for neg R"	Corr to Q" due to N"	Net corr to Q"
g	7	+	-	+	-	1	None	None	-1	-1
h	8	+	-	-	+	0	None	None	+1	+1
c	3	-	-	+	-	1	Abs add D	+1	-1	None
d	4	-	-	-	+	0	Abs add D	-1	+1	None

For summary, see flow diagram, Fig 5, Publ. No. 23.



MApy USES PARTS I AND II.
MPpy USES PART II ONLY.

INITIAL CONTENTS:

X:	MULTIPLICAND
Q:	MULTIPLIER
A:	INITIAL NUMBER (ZERO FOR MPpy)

FINAL CONTENTS:

X:	MULTIPLICAND
Q:	MULTIPLIER
A:	INITIAL NUMBER PLUS PRODUCT

Figure 4. MULTIPLY ALGORITHM (portion of MApy and MPpy)

HARLAN SNYDER

$\frac{}{(+2)} = (-1) + \frac{}{2}$

$\frac{\pm 1}{\pm 2} = (0) + (\frac{\pm 1}{\pm 2})$

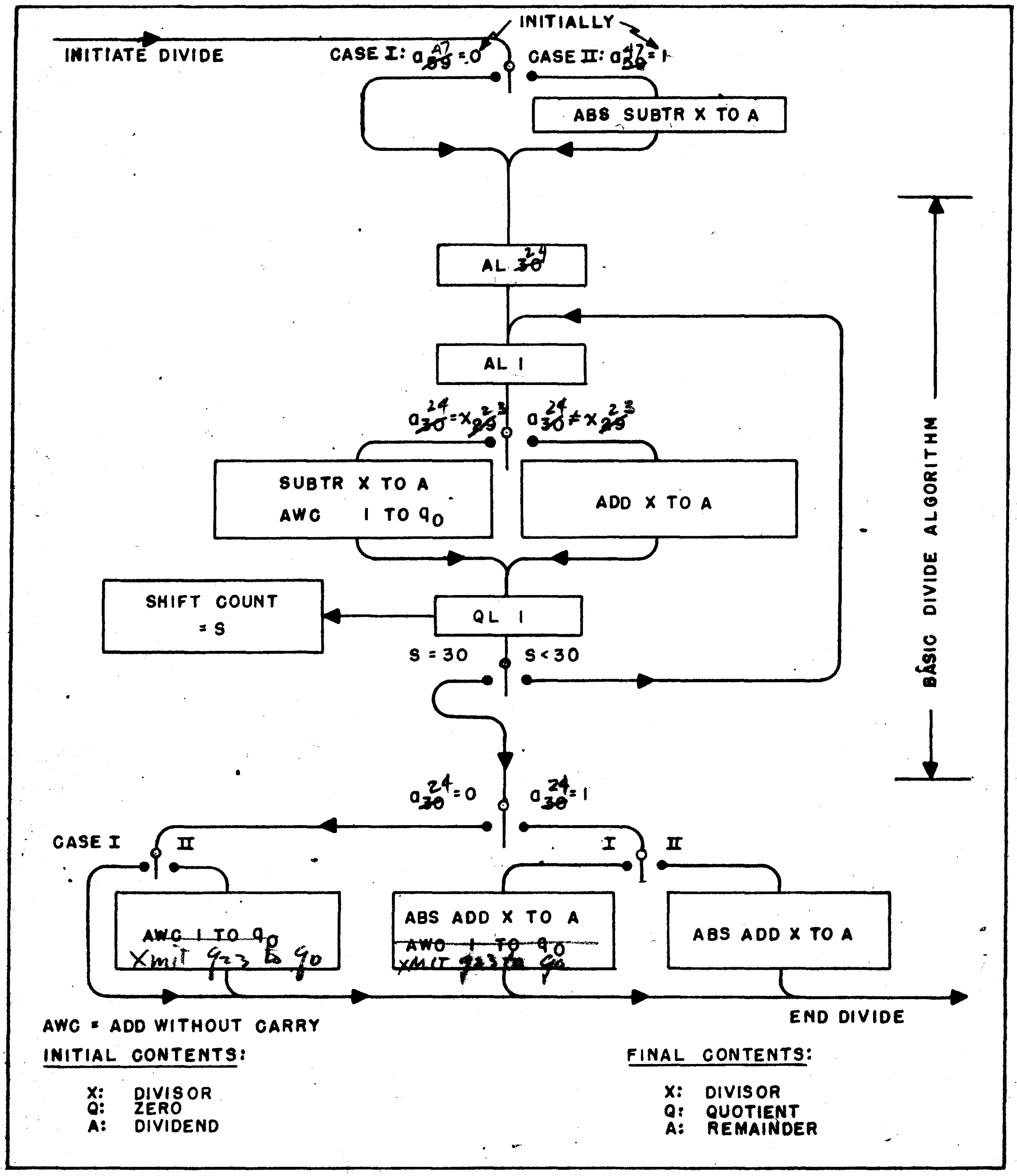


Figure 5. DIVIDE ALGORITHM (portion of DPpy)

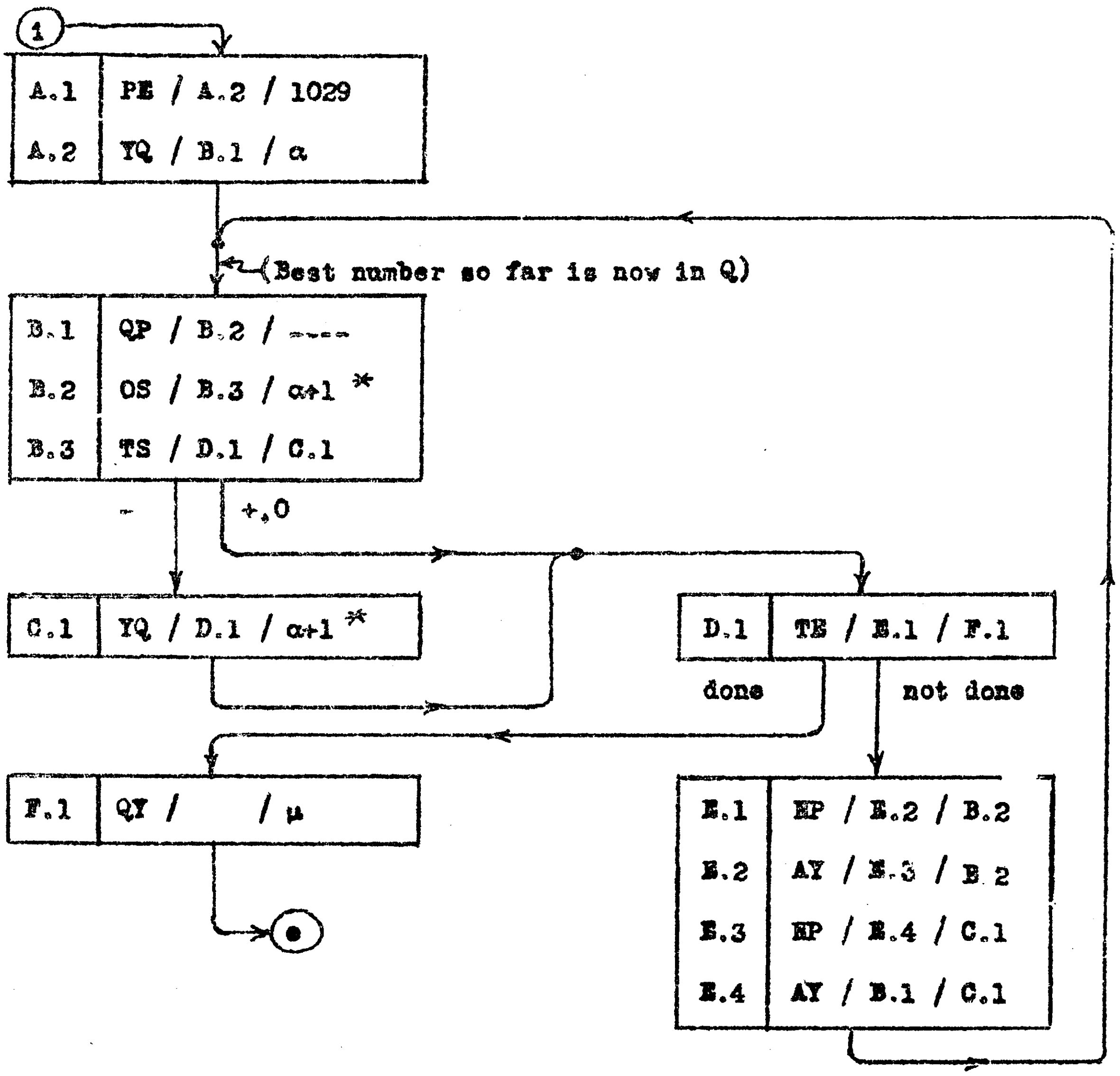
HARLAN SNYDER

PROGRAMMING OF ILLUSTRATIVE ROUTINES

<u>Example:</u> To find the most positive number in a list.

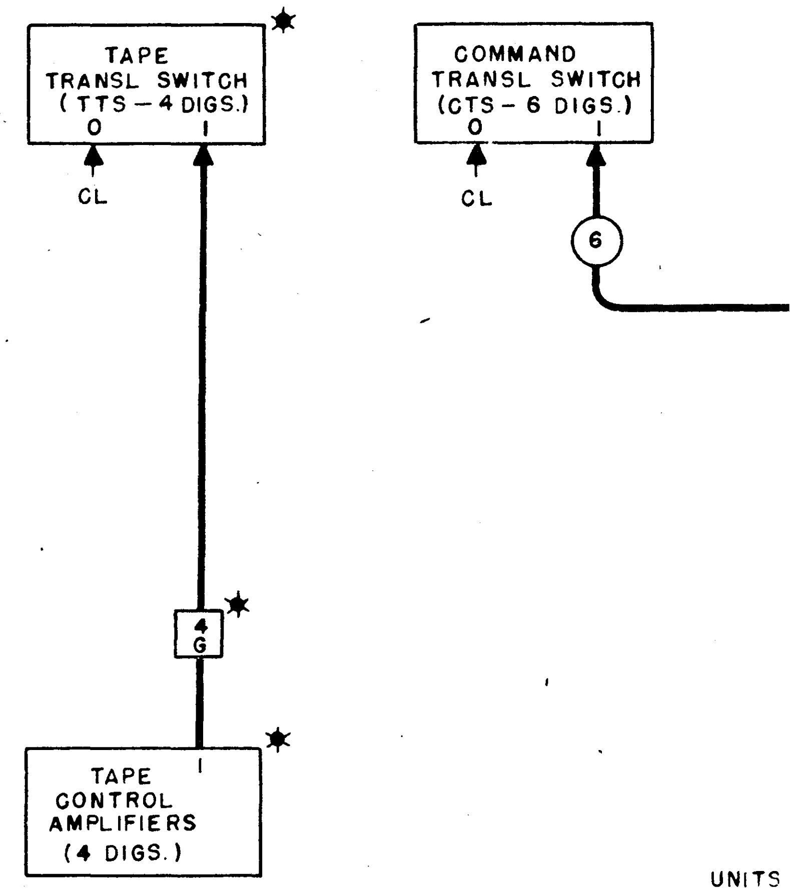
Given: 1030 numbers stored at α to α + 1029.

Req'd: To find the most positive number in the list and store it at μ.

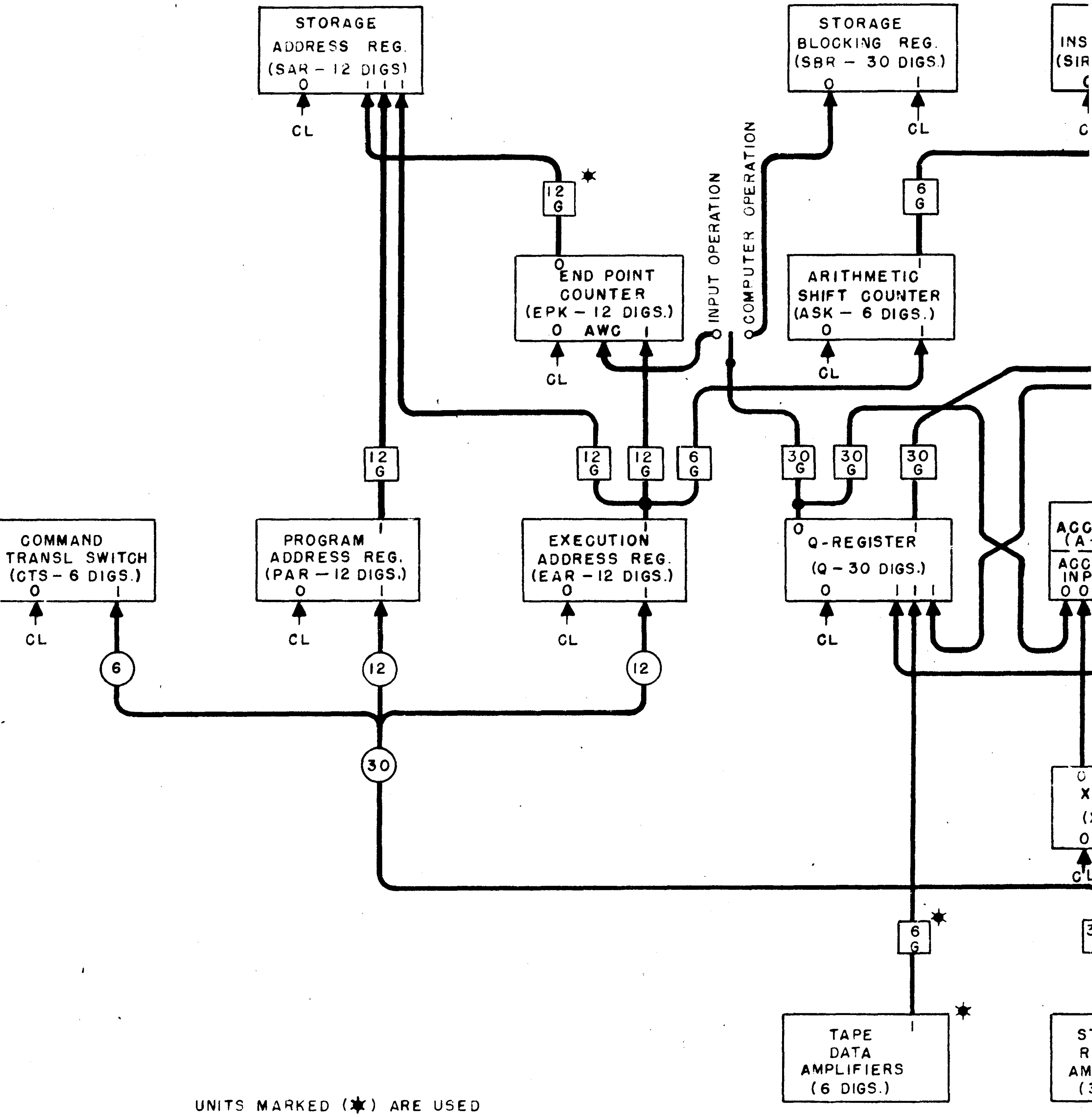


```
A.1   PE / A.2 / 1029
A.2   YQ / B.1 / α

B.1   QP / B.2 / ----
B.2   OS / B.3 / α+1 *
B.3   TS / D.1 / C.1

C.1   YQ / D.1 / α+1 *

D.1   TE / E.1 / F.1

E.1   EP / E.2 / B.2
E.2   AY / E.3 / B.2
E.3   EP / E.4 / C.1
E.4   AY / B.1 / C.1

F.1   QY /     / μ
```

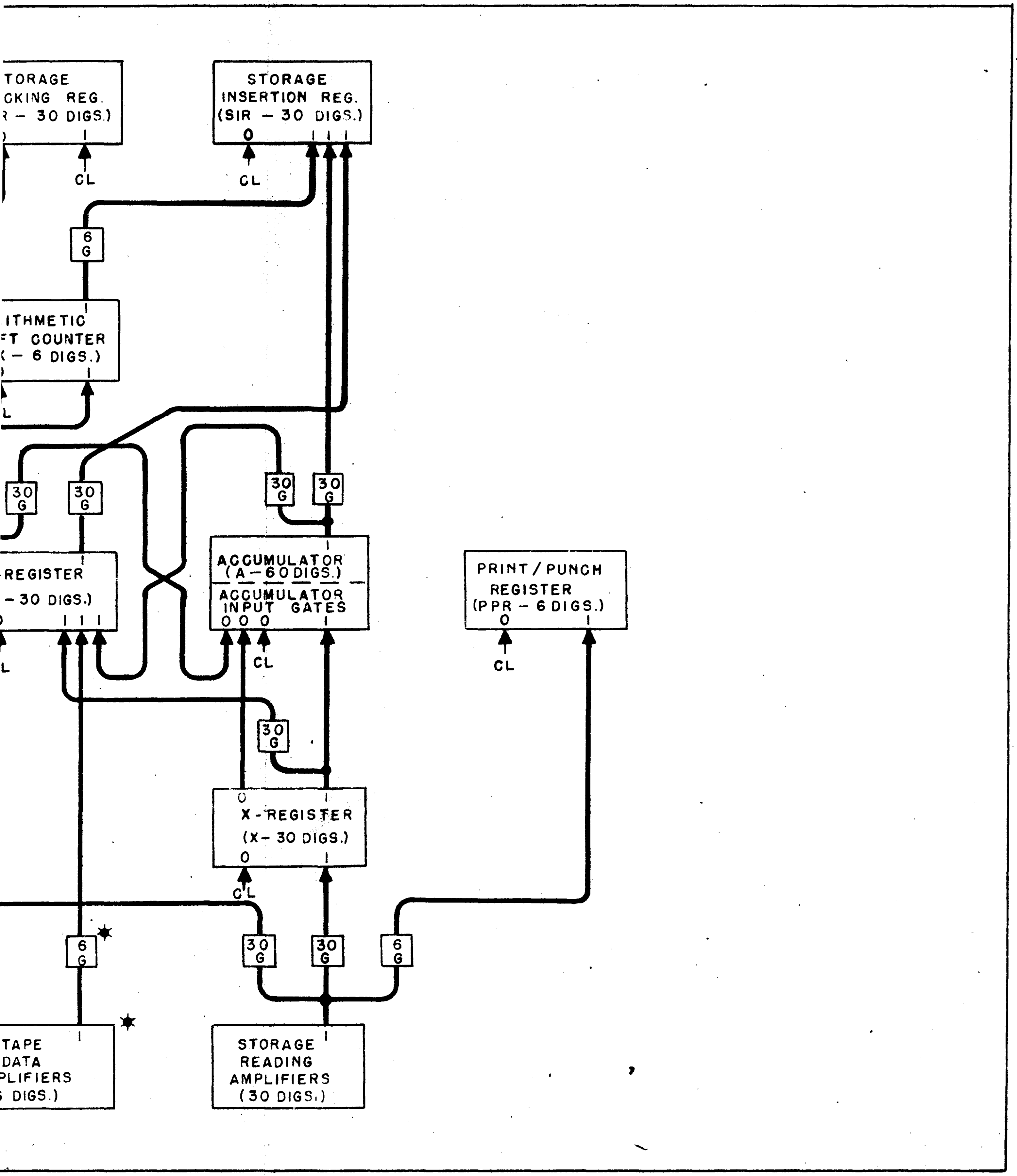
* Initial value shown - this changes for each traversal.


TAPE TRANSL SWITCH (TTS – 4 DIGS.)
0
1
CL
COMMAND TRANSL SWITCH (CTS – 6 DIGS.)
0
1
CL
6
4 G
1
TAPE CONTROL AMPLIFIERS (4 DIGS.)

UNITS
EX


STORAGE ADDRESS REG. (SAR – 12 DIGS)
0 1 1 1
CL
STORAGE BLOCKING REG. (SBR – 30 DIGS.)
0 1
CL
INS
(SIR
C
12 G
6 G
END POINT COUNTER (EPK – 12 DIGS.)
0 AWC 1
CL
INPUT OPERATION
COMPUTER OPERATION
ARITHMETIC SHIFT COUNTER (ASK – 6 DIGS.)
0 1
CL
12 G
12 G
12 G
6 G
30 G
30 G
30 G
COMMAND TRANSL SWITCH (CTS – 6 DIGS.)
0 1
CL
PROGRAM ADDRESS REG. (PAR – 12 DIGS.)
0 1
CL
EXECUTION ADDRESS REG. (EAR – 12 DIGS.)
0 1
CL
Q-REGISTER (Q – 30 DIGS.)
0 1 1 1
CL
ACC
(A
ACC
INP
0 0
6
12
12
30
X
CL
6 G
TAPE DATA AMPLIFIERS (6 DIGS.)
ST
R
AM

UNITS MARKED (✱) ARE USED EXCLUSIVELY FOR INPUT



TORAGE
CKING REG.
R – 30 DIGS.)
CL
STORAGE
INSERTION REG.
(SIR – 30 DIGS.)
0
1 1 1
CL
6 G
ITHMETIC
FT COUNTER
(– 6 DIGS.)
L
30 G
30 G
30 G
30 G
REGISTER
– 30 DIGS.)
L
ACCUMULATOR
(A – 60 DIGS.)
ACCUMULATOR
INPUT GATES
0 0 0
1
CL
PRINT / PUNCH
REGISTER
(PPR – 6 DIGS.)
0
1
CL
30 G
X - REGISTER
(X – 30 DIGS.)
0
1
CL
6 G
30 G
30 G
6 G
TAPE
DATA
PLIFIERS
DIGS.)
STORAGE
READING
AMPLIFIERS
(30 DIGS.)

X6 8716