I. Some basic concepts.

a. The Whirlwind I computer (WWI) uses words in its operation. A word is an array of sixteen binary digits. For reference, the sixteen binary digits of a word are numbered from 0 to 15, counting from left to right.

b. A word may be used as either a number or an instruction. The distinction is made by the manner in which the word is used, not by the form of the word itself.

c. WWI has an arithmetic element in which words are processed. When a word is used in the arithmetic element, it is generally being treated as a number.

d. WWI has a control element in which words are obeyed. When a word is used in the control element, it is always being treated as an instruction.

e. WWI has a magnetic-core memory (MCM) which contains 2016 storage registers or locations, numbered from 32 to 2047. The number by which a register is referred to is called its address. Each register can hold one word. Both the arithmetic element and the control element obtain the words they need from MCM.

f. To solve a problem using WWI, a sequence of words must initially be read in to MCM. This sequence of words, comprising the numbers and instructions required to solve a problem, is called a coded program or a routine. The procedure of determining a suitable method for solving a problem is called programming; the process of translating this method into a coded program is called coding.

[illegible] Numbers.

a. When a word is to be considered as a number, digit 0 represents its sign and the remaining fifteen binary digits specify its magnitude. It is not necessary to know the binary structure of a number when coding. Numbers may be read into and obtained from WWI in the decimal notation (the conversion to and from binary being done by the computer), so that the binary operations need not be of direct concern. It should, however, be remembered that no more than fifteen binary digits are available to represent the magnitude of a number. This is equivalent to about 4.5 decimal digits, and is the maximum precision obtainable without using special programming techniques.

b. All numbers are [illegible] a number can be represented if it is greater than or equal to [illegible] and negative fractions may be used. During [illegible] may be specified as decimal fractions by typing them in as:

+.2743 [illegible] +.0

Decimal fractions are typed with an algebraic sign (+ or -) followed by a decimal point followed by the decimal digits of the fraction. More than four decimal digits may be typed if desired; the computer rounds off to fifteen binary digits automatically.

c. Since the smallest distinguishable increment is $1 \times 2^{-15}$, all numbers may be expressed as $\pm N \times 2^{-15}$, where N is an integer. It is often useful to consider a number to represent the integer $\pm N$ (with the factor $2^{-15}$ always understood).*

*WWI always treats numbers as fractions. Since $(A \times 2^{-15}) \pm (B \times 2^{-15}) = (A \pm B) \times 2^{-15}$, it is immaterial which viewpoint is adopted when adding or subtracting. Care must be exercised, however, when multiplying or dividing numbers which the programmer is considering as "integers"; it is easy to see that the result is not another "integer."

This is helpful for counting [illegible] ing addresses, as will be seen later. Numbers may be typed as decimal integers:

+17 38 [illegible] +0

Decimal integers have an algebraic sign (the + sign may be omitted) and no decimal point.

d. The number zero has two distinct representations in WWI. One of these has a positive sign and is written +0; the other, which has a negative sign, is written -0.* Either form of the number zero may be used in an arithmetic operation and will yield the correct result. When the answer to a computation is zero, there are simple rules for determining which of the two possible zero results. These rules will be discussed later with the arithmetic operations to which they apply.

III. Instructions.

a. When a word is [illegible] as an instruction, digits 0 to 4 are devoted to the operation section and the remaining eleven digits to the address section. The operation section indicates the nature of the instruction (addition, multiplication, etc.) and the address section normally contains the address of a register whose contents is to be used in the operation. In a few instructions the address section is not the address of a register at all, but is used for other purposes. This will be explained when the instructions are discussed individually.

b. For input to WWI, instructions are typed as a two-letter mnemonic code followed by a floating, relative, or absolute address. These forms of address have been discussed earlier in connection with the CS computer, and the conventions are exactly the same for WWI. However, the two-letter WWI instructions should never be confused with the three-letter CS instructions. Examples of WWI instructions are:

*+0 consists of 16 binary 0's; -0 of 16 binary 1's.

| | | | |
|---|---|---|---|
| cam5 | dmt2+4 | spr6-3 | (floating addresses) |
| mr3r | cp2r | (relative addresses) | |
| si384 | cs32 | (absolute addresses) | |

IV. The instructions ca x, cs x, cm x — the accumulator (AC) and B-register (BR).

a. Words are processed in WWI by use of the arithmetic element. This element contains within it several registers which take part in the information processing. The most important of these is the accumulator (AC), a sixteen-binary digit register which is used in most of the WWI instructions. It is the AC in which sums and products, for instance, are formed. Another 16 digit register, the B-register (BR), can be viewed in many cases as an extension to the right of AC. The uses of BR will become apparent later.

b. It is usually necessary to bring the contents of a storage register into AC preparatory to further operations upon the word which it contains. For this purpose the following WWI instructions are defined.

| | |
|---|---|
| ca x | clear AC and BR and add C(x) to AC* |
| cs x | clear AC and BR and subtract C(x) from AC |
| cm x | clear AC and BR and add magnitude of C(x) to AC |

These instructions provide flexibility in bringing the desired form of a word in MCM into AC. Note that they all clear BR and that they leave C(x) unchanged.

V. The instructions ad x, su x, dm x, sa x, ao x — the arithmetic-check (over-flow) alarm.

a. The WWI computer has several instructions which are used for addition and subtraction. The simplest and most straightforward of

*The descriptions of WWI instructions in these notes are brief and are intended only to point out the more significant features of the instructions. A complete description of all WWI instructions is contained in M-1624-2 and in D-55192, to which reference should be made.

these are

ad x      *add* C(x) to C(AC) and store sum in AC

su x      *subtract* C(x) from C(AC) and store difference in AC

Both of these instructions leave C(BR) and C(x) unchanged.

b. The instruction dm x is used for forming the *difference* of the *magnitudes* of two numbers.

dm x      place in AC the quantity $|C(AC)| - |C(x)|$,

place the previous C(AC) in BR,

leave C(x) unchanged.

The fact that the previous contents of AC appears in BR after dm x is executed is often useful.

c. C(BR) may be added to C(x) by using the instruction

ab x      *add* C(BR) to C(x), store sum in AC *and* in x.

Leave C(BR) unchanged.

Since C(BR) is unchanged, the above instruction facilitates adding the *same* quantity to the contents of several registers.

d. It is very often desired to increase the contents of a storage register by $1 \times 2^{-15}$. If C(x) is an instruction, this increases its address section by one. The instruction ao x makes this special case of addition very easy.

ao x      *add one* times $2^{-15}$ to C(x) and store the sum both in x *and* in AC. Leave C(BR) unchanged.

e. It has been mentioned that zero has two representations in WWI, +0 and −0. In general, a zero resulting from addition or subtraction is −0. In two cases only, +0 will be obtained. These cases are (+0) + (+0) = +0 and (+0) − (−0) = +0.

f. All numbers in WWI are fractions. It is obviously possible for the sum or difference of two such numbers to equal or exceed one. If this happens, the result cannot be represented in the computer and an <u>alarm</u> will occur. The alarm caused by this type of error is called the <u>arithmetic-check</u> or <u>overflow</u> alarm. When any alarm occurs, the computer stops with the contents of the registers of the arithmetic and control elements displayed in lights on the control panel.

Should a routine stop on an alarm, it indicates that a programming or coding mistake has been made and it is, of course, necessary for the programmer to locate and correct the mistake. The "post-mortem" routines which are available to aid the programmer in this task have been described in an earlier lecture.

VI. <u>The instructions ts x, td x used for storing results</u>

a. In order to retain a result which has been produced in AC, it is usually necessary to place the result in a storage register. This is accomplished by the instruction

| | |
|---|---|
| ts x | transfer C(AC) to storage register x. Leave C(AC) unchanged. |

The previous C(x) is lost after the instruction ts x is executed.

b. It will be remembered that the last eleven binary digits of an instruction constitute the address section. Often only the address section of an instruction in a storage register is to be modified, the operation section being unchanged. For this purpose there is provided the instruction

| | |
|---|---|
| td x | transfer last 11 digits of C(AC) to last 11 digits of register x. Leave C(AC) and first five digits of C(x) unchanged. |

If C(x) is an instruction, td x causes its address section to be replaced by the address section of the word in AC.

An instruction which is frequently very convenient is

ex x — exchange C(x) and C(AC); i.e., place C(x) in AC and place previous C(AC) in x.

This instruction permits the coder to bring C(x) into AC as does ca x, but at the same time it also stores the previous C(AC) in x. It thus combines two logically separate functions in one instruction. Note that ex x does not change C(BR).

VII. Transfer of control, the instructions sp x and cp x -- the A-register.

a. The WWI computer obeys instructions in sequence unless a specific instruction which breaks this sequence is executed. The instruction

sp x — take next instruction from register x and continue obeying instructions in sequence from there

permits the coder to specify a break in the sequence of control. The instruction sp x is always obeyed; it is an unconditional transfer of control.

b. An extremely valuable instruction is cp x, which permits the programmer to make a transfer of control conditional on the result of the immediately preceding calculation.

cp x — If C(x) is negative, proceed as in sp x; if C(AC) is positive, ignore this instruction and go on to the following instruction in sequence.

cp x is the only conditional transfer of control instruction available in the WWI computer. All "decisions" in the computer which choose one sequence of operations rather than another are made using this instruction. It is possible to reduce virtually any criterion for choice among a number of possible routines to a sequence of suitable "yes-no" decisions. It is therefore possible, using only the cp x instruction, to realize even extremely intricate and elaborate decision criteria in the computer.

c. Whenever an instruction is executed by WWI, that instruction must, of course, be located in a storage register. Each storage register has an address, y. Whenever an sp x or cp x instruction is executed in register y, the address (y+1) is stored in the A-register (AR), another register of the arithmetic element. The AR is a sixteen-binary-digit register, but only last eleven digits are affected by sp x or cp x. The address y is the register in which sp x or cp x is located; it should not be confused with register x, the address within the sp x or cp x instruction itself. The address (y+1) is stored in AR on cp x even when C(AC) is positive and cp x is otherwise effectively ignored.

VIII. Closed sub-routines -- the instruction ta x.

a. The use of sub-routines in the solution of a problem has already been discussed in connection with the CS computer. It will be recalled that a subroutine is a sequence of instructions which may be entered from several points in a larger routine. Most often, it is desired that a subroutine be closed; i.e., it is desired that it return, when it is finished, to the main routine at the point from which it was entered.

b. In writing closed subroutines for the WWI computer, the instruction

| | |
|---|---|
| ta x | transfer the last eleven digits of the A-register to the last eleven digits of register x. Leave the operation section of register x and the contents of all other registers unchanged, |

is invaluable. Subroutines are invariably entered using the operations sp or cp. The address section of the AR is set equal to (y+1) by these instructions. If the initial register of a subroutine contains the instruction ta x, the address (y+1) may be inserted into any register x of the subroutine. In particular, it may be inserted into the sp or cp operation which is used to leave the routine, and in this case the subroutine is closed.

Another use for ta x will be discussed with the instruction sf x.

c. One caution must be observed in using the instruction ta x. While it has not been explicitly stated in these notes, most WWI instructions change C(AR). ta x must be used immediately after sp x or cp x before any instructions which modify C(AR) are executed. ta x should be the <u>first</u> instruction of a closed subroutine.

IX. <u>The instructions mh x, mr x, dv x -- the divide-error alarm.</u>

a. WWI has two instructions which are used to multiply numbers.

| | |
|---|---|
| mh x | <u>m</u>ultiply C(AC) by C(x) and <u>h</u>old the full thirty-binary digit product in AC and BR, treating BR as an extension to the right of AC. |
| mr x | <u>m</u>ultiply C(AC) by C(x) and <u>r</u>ound-off the product to fifteen binary digits in AC. Clear BR. |

It is clear that the product of two fifteen-digit numbers is a thirty digit number. The full thirty-digit product may be retained by using mh x; the product may be properly rounded off to one register length by using mr x. In general, mr x is more frequently used in ordinary calculations than is mh x.

b. In both the instructions mh x and mr x the sign of the product is determined according to the usual rule for multiplication. In particular, this is true when one of the factors is +0 or -0; the sign of the product is still determined from the ordinary rule, giving to each of the zeros its appropriate sign.

c. Negative numbers in WWI are stored as the <u>complement</u> of the corresponding positive number. To obtain the binary representation of a negative number, one first forms the representation of the positive number and then all 0's are changed to 1's and all 1's are changed to 0's. The fact that the complementary form is used within WWI is normally of little significance to the coder.

However, it is a peculiarity of the computer that the B-register is never complemented when the result of a multiplication extends into it. The digits in BR always appear as their positive magnitude, even though the digits in AC are complemented if the product is negative.

In order to obtain the digits in BR after the mh x instruction with their proper sign, it is most convenient to use a suitable numerical-shift instruction to move them into AC. The shift instructions will be described in detail later.

d. WWI has one divide instruction:

dv x    divide C(AC) by C(x), storing the quotient in BR.

After the execution of dv x, AC contains a zero of the same sign as the quotient. The quotient is in BR, but it is uncomplemented (just as in the case of mh x) even though it may be negative. Again, a numerical-shift instruction is required in order to bring the quotient into AC with its proper sign. In general, dv x should be followed by slh 15 or slr 15.

e. If the dividend equals or exceeds the divisor, the result will exceed the capacity of the computer. Should this mistake occur, the computer will stop on a divide-error alarm.

X. The numerical-shift instructions -- slh n, slr n, srh n, srr n.

a. We are all familiar with the procedure of multiplying a decimal number by $10^N$ simply by moving the decimal point to the right N places. An analogous procedure exists in the binary number system, in which moving the binary point to the right n places multiplies a binary number by $2^n$. Similarly, moving the binary point to the left n places divides a binary number by $2^n$.

b. In WWI, the binary point is fixed at the left immediately following the sign. The binary point cannot be moved, but the same effect may be achieved by shifting the number itself with respect to the fixed binary point. If the number is shifted to the right, it is equivalent to moving the binary point to the left, and vice versa.

c. The numerical-shift instructions in WWI provide a means for shifting the combined contents of AC and BR to the right or left, thereby dividing or multiplying by the corresponding power of two. The shift-left instruction is useful for bringing C(BR) into AC. Since these are numerical shift instructions, the sign digit (digit 0) of AC is not shifted; only those digits of AC and BR which actually are numerical take part in the shift. When the sign of AC is negative, C(AC) is assumed to be complemented, but C(BR) is not. This corresponds to the manner in which negative results are stored in AC and BR after the instructions mh x and dv x.

| | |
|---|---|
| slh n | Shift the combined AC and BR to the left n* places. Hold all digits in AC and BR after the shift. |
| slr n | Shift the combined AC and BR to the left n* places. Round-off the C(AC) on the basis of the magnitude of C(BR) after the shift. Then clear BR. |
| srh n | Same as slh n, only shift is to the right. |
| srr n | Same as slr n, only shift is to the right. |

In all four of these instructions, digit 0 of AC is not shifted, any digits shifted left out of AC 1 or right out of BR 15 are lost, and if

*n is taken modulo 32.

C(AC) is negative, AC is complemented before and again after the shift.

d. Note that these instructions differ somewhat from the form of the other WWI instructions discussed so far. First of all, three letters are required to specify the shift operations instead of two. All three must be typed or ambiguity will result.

A more significant difference is that the address section of these instructions does not refer to a storage register at all, but specifies by how many places the number is to be shifted. Since only AC and BR are involved in these instructions, no storage register need be specified and the address section may be used for this purpose.

The operation sections of slh n and slr n are identical, as are the operation sections of srh n and srr n. The distinction between these instructions is made not by the operation section but by digit 6 (the second digit of the address section). Since the address n is small, digit 6 may be used without causing any difficulty. In slh n and srh n, digit 6 must be a one; in slr n and srr n, it must be a zero. If the address of one of these instructions is changed by using a <u>td x</u> instruction, the coder must remember to preserve the correct value of digit 6.

e. Note that after a shift instruction, C(AC) may be so large in magnitude that round-off can cause it to equal unity.* In this case, if the instruction calls for round-off, the arithmetic-check or overflow alarm will result.

XI. <u>The logical cycle instructions -- clh n and clc n.</u>

a. On occasion it is desirable to move the digits of a word as it stands to the left or right without regard to the numerical significance of the digits. In this case, the sign digit is treated like any

*This occurs when C(AC) $= 1-2^{-15}$.

other digit; the process is simply one of reorienting the digits without regard to their meaning either as a number or an instruction.

Two instructions are available for executing this operation:

clh n — Cycle the combined AC and BR to the left by n places. Carry any digits cycled out of AC 0 into BR 15. Hold all digits at the end of the cycle.

clc n — Same as clh n, except clear BR after the cycling.

b. Note that AC and BR are treated as a closed ring; digits cycled out of the left of AC appear at the right of BR. No round-off occurs. Digit 6 of clc n must be zero; digit 6 of clh n must be 1.

XII. Scale factoring - the instruction sf x.

a. Fractions may, in general, have one or more zeros between the binary point and the first significant digit. These zeros are not significant, in the sense that the fraction may equally well be expressed as another fraction (having no initial zeros) times $2^{-N}$, where N is the number of initial zeros in the original fraction. This latter form has the advantage that all its numerical digits are significant, so that the numerical value may be expressed to full fifteen-binary-digit precision. However, in performing arithmetic operations on numbers expressed in this form, due account must be taken of the factor $2^{-N}$ associated with each number; they cannot be combined directly using WWI arithmetic instructions.

b. The instruction

sf x — scale factor the combined AC and BR; i.e., shift the contents of AC and BR left until there are no initial zeros. Store N, the number of times shifting was necessary, in the address section of x and of AR

permits numbers to be expressed easily in scale factored form. Note that the procedure for handling the scale-factored numbers must be coded by the user; in the WWI computer, no automatic facilities exist for taking scale factors into account. sf x treats numbers numerically, just as do the shift instructions.

Since N appears in AR as well as in register x, it may be placed in other registers also by using the ta operation immediately following sf x. The operation section of x is unchanged by sf x; this should normally be zero before the sf x instruction is executed. If C(AC + BR) is $\pm 0$, N is set equal to 32.

XIII. The instruction sa x -- special-add memory.

a. Normally, if the result of an addition is as large as unity in magnitude, an overflow alarm occurs. Occasionally, the coder will find it convenient to permit an overflow to occur without alarm, taking account of the overflow later in the routine. The instruction sa x, special-add x, differs from the instruction ad x only in its behavior in the event of overflow.

sa x — add C(x) to C(AC), store the fractional part of the sum in AC. Store the integer part of the sum (0, +1, or -1) times $2^{-15}$ in special-add memory (SAM). Give no overflow alarm.

SAM is a special register of the arithmetic element. Its only possible contents are 0, +1, or -1, and these are stored in SAM only by the sa x instruction.

b. The contents of SAM after the instruction sa x is executed may be used by executing one of the instructions ca x, cs x, or cm x. If C(SAM) $\neq$ 0, then C(SAM) is added to the word which otherwise would be brought into AC by these instructions, and the sum is placed in AC.

When C(SAM) = 0, this addition does not change C(x) and the ca x, cs x, and cm x instructions behave as was described earlier. SAM is always cleared when one of the instructions ca x, cs x, or cm x is executed. Note that an overflow alarm will occur on these instructions if the addition of C(SAM) causes the number which is to be placed in AC to equal unity.

c. The execution of any of the following instructions clears SAM without using its contents: ab x, ad x, su x, ao x, dm x, mr x, mh x, dv x, slr x, slh x, srr x, srh x, sf x.

d. SAM may always be assumed to be clear after the read-in of a program tape.

XIV. The instruction md x.

a. A logical instruction, useful for retaining certain digits of a word while setting the other digits equal to zero, is

md x    logically multiply each digit of C(AC) by the same digit in C(x) and place result in AC.

The effect of this instruction is to set the digits of AC which correspond to zeros in x equal to zero, and to leave the digits of AC which correspond to ones in x unchanged.

b. md x is a non-arithmetic instruction; it treats C(AC) and C(x) simply as an array of digits, without regard for their numerical significance.

XV. The instruction ck x -- the check-register alarm.

a. The instruction

ck x    Compare C(AC) and C(x). If they are identical, proceed to the next instruction; if they differ, stop the computer in a check-register alarm.

enables the coder to stop the computer in case a computed word does not agree exactly with some predetermined value.

b. The instruction ck x is rarely used in mathematical computations. It finds its widest application in coding which involves use of auxiliary equipment by the computer. It may be employed for instance to insure that information has been transferred correctly from an external unit to the computer.

XVI. <u>The input-output instructions -- the instructions si 0 and si 1 -- the program alarm and the inactivity alarm.</u>

a. The remaining instructions are used for controlling the input and output equipment associated with the WWI computer. This equipment encludes, among others, the photo-electric tape reader, the Flexowriter typewriter, the magnetic tape units, and the magnetic drum. The details of coding for the input-output equipment will be discussed in a later lecture; only the features of the input-output instructions which are common to all equipment will be considered here.

b. The instruction

si pqr     select the input or output device specified by the address pqr.

is used to select an auxiliary device and, if that device has more than one mode of operation, to specify the desired mode. The address pqr associated with each unit and mode must be determined from a table of si addresses. Frequently, C(AC) at the time si pqr is executed is used to give further information in selecting the unit; at other times, C(AC) is immaterial. si pqr always leaves all registers of the arithmetic element undisturbed.

c. After a unit and mode have been selected using the si pqr instruction, whichever of the following instructions as is appropriate may be executed:

| | |
|---|---|
| rd n | read one word from the selected device into AC. |
| rc n | record the word in AC on the selected device. Leave C(AC) unchanged. |

The address n is usually immaterial.

d. It is obvious that, if the auxiliary device has been selected in a recording mode, the instruction rd n is illogical; similarly, if a read mode has been selected, the instruction rc n cannot logically be given. If such a mistake occurs, the computer stops on a program alarm.

e. When a group of consecutive words is to be read or recorded, the following instructions may often be employed:

| | |
|---|---|
| bi x | read a block of n consecutive words into MCM starting at register x. |
| bo x | record a block of n consecutive words out of MCM starting at register x. Leave the contents of these registers unchanged. |

$\pm n \times 2^{-15}$ must be in AC at the time the bi x or bo x instruction is executed. After the instruction is completed, C(AC) = x+n.

f. The illogical use of bi x or bo x will result in a program alarm.

g. Some of the auxiliary equipment is free-running, and rd n or bi x instructions must be given frequently enough to keep pace with the free-running units. If this is not done, the computer will stop on an inactivity alarm.

h. Two si addresses are used to stop the computer.

| | |
|---|---|
| si 0 | Stop the computer. |
| si 1 | Stop the computer if the "STOP ON si 1" switch is on; otherwise, continue to the next instruction. |

The "STOP ON si 1" switch is on the control console, and is set by the computer operator as requested by the coder. The "STOP ON si 1" switch will be on unless otherwise requested.

XVII. Parity alarm.

The coder may occasionally encounter an alarm which has not been discussed with any of the WWI instructions.

Each time a core-memory register or a register on the magnetic drum is referred to, a so-called "parity check" is carried out to determine whether one of the digits in the register has changed since it was recorded. In virtually all cases the parity check is completed successfully and computer operation continues.

Occasionally, however, a computer malfunction causes a faulty recording or a change in the information recorded in a register. When the parity check detects this, the computer stops on a parity alarm. A parity alarm may also be obtained if a non-existent magnetic-drum group is selected by the coder.

Should a routine stop on a parity alarm not traceable to an improper drum reference, it may be some consolation to the coder to know that the alarm results from an error made by the computer and not by the coder.

XVIII. Test storage -- registers 0 and I.

a. It has been mentioned that the registers of core memory are numbered starting at address 32. Registers 0 through 31, which are not part of core memory, exist and are referred to as test storage. Most of the registers of test storage have their contents set into them by toggle switches; the contents of these registers cannot be changed. Five of the test-storage registers are flip-flops, a form of storage register whose contents can be changed.

b. In general, the registers with addresses less than 32 should not be used by the coder. There are only two exceptions to this rule.

c. The first of these exceptions is concerned with the read-in of program tapes. The coder may desire that a routine terminate not by stopping the computer, but by reading in a new tape, containing additional data or a different routine. This may be accomplished, if desired, by transferring control to register 26 (by use of the instruction sp 26 or cp 26). Register 26 is the first register of the initiating routine for the input program; transferring control to this register is equivalent to pushing the READ IN button on the computer console.

d. The coder may also make use of the contents of registers 0 and 1. Register 0 permanently contains the number +0; register 1 contains the number $+1 \times 2^{-15}$. The coder need not store either of these constants in his routine; he may use 0 or 1 as the address section of an instruction. Any attempt to change C(0) or C(1) will not be successful, but the computer will simply proceed to the next instruction without alarm.*

Note that the operation section corresponding to the instruction si consists entirely of 0's. Thus, the word +0 is also the instruction si 0 and the word $+1 \times 2^{-15}$ is the instruction si 1. A way of stopping the computer is the instruction sp 0 (cp 0). The computer stops by obeying the instruction si 0 in register 0.

XIX. The panel control buttons.

a. The control buttons are used by the computer operator. An understanding of their function is very valuable to the coder when

*The instruction ao 1, oddly enough, produces $+2 \times 2^{-15}$ in AC, although C(1) remains unchanged.

writing out his performance request.

b. Pressing the READ IN button brings into operation a service routine called the group 11 input program. Among the many functions which may be performed by this routine is the reading of paper tape via the photo-electric tape reader (PETR). All tapes are normally read into the computer using this routine.

c. When the ERASE button is pressed before pressing the READ IN button, the group 11 input program erases core memory before reading in the next tape. Otherwise, no erasing is done; the tape which is read is superimposed on the previous contents of core memory.

d. The STOP button stops the computer, and is used whenever it becomes necessary to stop operation manually. Normally, the computer stops automatically on the instructions si 0 or si 1.

e. The RESTART button causes the computer to recommence operation at the register immediately following the one in which it stopped. It may be used either following the instruction si 0 or si 1 or after the STOP button has been pressed.

f. The START OVER button enables the operator to start the computer at any register of core memory. The address of the register must be entered by the operator in a set of toggle-switches provided for this purpose.

g. The START OVER AT 40 button starts computer operation at register 32 (this is equivalent to 40 in the octal number system).

Exercises on WWI

1. 317 numbers are stored in consecutive registers starting at address b12. Write a routine which places the sum of all these numbers in register 32 and then goes on to the next instruction. Assume that the numbers are such that no overflow will occur and that register 32 initially contains +0. (7 words)

2. If digit 13 of AC is a one, transfer control to register r1; if it is a zero, transfer control to register r2. (3 words)

3. Register m5 contains a word whose address section gives the location of a number. If this number is positive, transfer control to register q12, if it is negative stop the computer. (5 words)

4. Change the operation section of the word in register b2 to mr, leaving its address section unchanged, then transfer control to register t5. (7 words)

5. Which instruction clears both AC and BR?
   Which clears AC without affecting BR?
   Which clears BR without affecting AC?

6. Repeat exercise 1, without making the assumption that no overflow will occur. Stop the addition when either the end of the list of numbers is reached or an overflow occurs. The overflow may be of either sign. Be sure that the number which actually caused overflow is not included in the sum in register 32. When the additions are stopped for either reason, proceed to the next instruction. (14 words)

7. Write a closed subroutine which raises C(AC) to the n-th power and stores the result in AC. $+n \times 2^{-15}$ is stored in the register immediately following the sp instruction which entered the subroutine. Control should be returned to the register immediately following the one containing $n \times 2^{-15}$. (n 0) (19 words)

A. Siegel
10/22/54