# IBM 705

CE Training

Molloy

# 705 GENERAL ORGANIZATION & LOGIC FLOW

The type 705 computor is a large high speed computor that can take raw data and process it into compact reports. The computor can be divided into sections that have specific duties. This breakdown of the machine produces the following parts: Input devices, memory, central processing unit, and output devices.

Input Devices:

Input devices are units that bring raw data into the machine. These include tapes, card readers and drums. These units handle raw data at high speeds and place the information in memory in the same sequence that it was on the tape, cards or drum.

Memory:

Memory consists of a box of ferrite cores. These cores are small doughnut shaped pieces of material that can be magnetized. If the magnetic flux in the cores flows in one direction the core is said to be reset. If the flux flows in the opposite direction the core is said to be set. The cores then have the ability to retain information. A table of terms for a single core follows:

| Reset | Set |
|---|---|
| Does not contain a bit | Contains a bit |
| "0" | "1" |
| Off | On |

A core is set if it contains a bit of information.

The 705 memory consists of 35 core planes. Each plane is 50 cores wide by 80 cores long. Each of the 140,000 cores can have a separate bit of information stored in it.

The type 705 uses a coding system where each number and character of the alphabet can be coded with 7 bits of information. These bits are named C, B, A, 8, 4, 2, 1. A numeric character uses the numeric bits with the C bit. An alphabetic character uses the numeric bits and the A, B bits (called zone bits) with the C bit. For example:

| | |
|---|---|
| 1 | 1 bit, C bit |
| 2 | 2 bit, C bit |
| 5 | 4 & 1 bit |
| 7 | 4, 2 & 1 bit, C bit |
| B | A & 2 bit |

The C bit is an extra bit that is used for checking purposes. If a character has an odd number of bits, the C bit is added to make it even. The 705 can then check all information to make sure that it has an even number of bits. If a single bit is dropped from any character because of some malfunction of the machine, it can be detected before the erroneous information is worked on.

Zone bits are usually shown in binary form as follows:

| | | |
|---|---|---|
| 12 zone | A & B | (11) |
| 11 zone | A | (10) |
| 0 zone | B | (01) |

As each character is made up of a combination of 7 bits, then memory with a total of 140,000 cores is able to store 20,000 individual characters. Memory then can have 20,000 character locations, each of these locations consisting of 7 cores for the 7 bits of information that are needed to describe a character. Each character location is assigned a number from 0 to 19,999. Each of these locations in memory can be addressed by the central processing unit (CPU) and a character can be either read or written at this location.

Memory has two functions. The first is to store the program. The program might be compared to the control panel in EAM equipment. It tells the machine what to do. The second function of memory is to store the raw data fed from the input devices and send it to CPU or receive from CPU the usable data. This data may be in the form of intermediate or finished data. If it is intermediate data it will be stored in memory to be fed back to CPU at some later time. If it is finished data then it will be taken out of memory to leave space for other data. If it is finished data it will be given to the output devices.

Central processing unit:

The central processing unit is where data is operated on. It has the ability to add, subtract, multiply, divide, compare and perform other logical functions. It also controls the addressing system of memory by using the program stored in memory. It can operate on two characters at a time only. These two characters are supplied one by memory and one by storage. Storage is an auxillary memory that can store 512 characters. It can store only intermediate data and can be fed only from CPU. An example of CPU operation follows:

It is desired to add two fields together. Both fields and the program will be read into memory from the input devices. The program is decoded in CPU and CPU will now operate as directed by the decoded program. To add two fields, one field must be in storage. Reset add one of the fields in memory to storage, one character at a time. Now CPU will take one

character at a time from both memory and storage and add them together. The result of the first two characters will now be taken and put in storage replacing the character that was taken out. Any carry will be stored as a bit in CPU to be added to the next two characters to be brought in. The character that was brought out of memory will be put back in the same place to be saved for any possible future application. When both fields have been added it will be found that the answer has replaced the field that was taken from storage. The sum must now be put in a form that is usable. To do this, output devices are used.

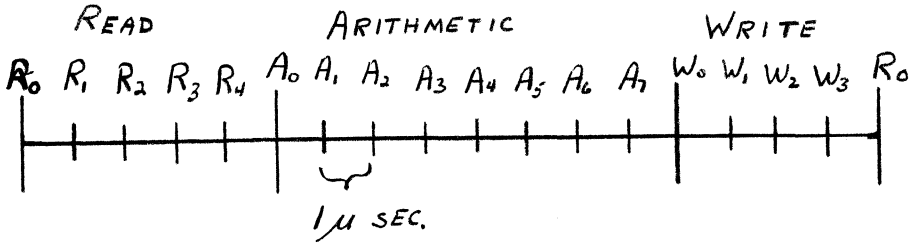
Output Devices:

The output devices used with the type 705 are printers, punches, tapes and drums. These units take information from memory in the 7 bit code, convert it to the codes used with the device and put it on some sort of form. With the printer this would be a printed sheet of paper, the punch would put it in the form of standard IBM cards, the tape unit will put it on magnetic tape in a 7 bit code and the drum will put it on the surface of the drum as magnetized spots in a 7 bit code.

CPU

Clock:

The heart of the Type 705 machine is the clock. The clock is used to control all functions of the machine so that everything can be kept in time and operations happen in a logical sequence. It consists of a series of 17 triggers that are formed into an electrical ring. A crystal oscillator feeds this ring pulses at the rate of one every micro-second. The ring of triggers is arranged such that the oscillator pulse turns off a trigger. This trigger going off turns the one next to it on. The next oscillator pulse will turn off the trigger that was just turned on and this will turn on the next trigger. If only one trigger is on to start with, each trigger will be turned on and off around the ring. Once every 17 micro-seconds then, the ring will come back to the starting point. The output of the triggers in the ring can be used to time functions in the machine as the ring progresses. The 17 micro-second clock is broken down into 3 major divisions. These are read time, arithmetic time, and write time.




Read time is used to take information out of memory (Read memory). Arithmetic time is used to operate on the character taken from memory. Write time is used to write information back into memory.

## 705 Clock Operation

Under the control of CPU the clock will run in two different modes:

### USE MODE

The clock will be considered in USE mode whenever the machine is in the process of reading or executing an instruction.

### MANUAL MODE

The clock will be in MANUAL mode whenever the machine is not in the process of reading or executing an instruction.

### USE MODE OPERATION

1. Instruction Time:

During I time the clock will generate a 17 u-sec. cycle consisting of 5 usec. of READ time, 8 usec. of A time and 4 usec. of WRITE time. The necessary pulses for reading a 5 character instruction from memory and setting the proper registers will be generated by the Wave form generator under the control of the I Time line which will in turn be controlled by the CPU instruction timer.

2. Execution Time:

During execution time, the clock will generate a 17 usec. cycle consisting of 5 usec. for Read, 8 usec. for Arithmetic and 4 usec. for Write time. However, if the instruction to be executed is a TRANSMIT instruction, or an I/O instruction, the clock will generate, continually, 9 usec. cycles consisting of 5 usec. for Read and 4 usec. for Write time. The Wave form generator will generate pulses necessary to execute these instructions under the control of the Transmit type cycle triggers and the Read or Write Request gates which will be generated when Read or Write response signals are received from the Input/Output units.

### MANUAL MODE

During the Manual mode, the machine will be idle but the clock will continue running at the 17 usec. rate continually through Read, Add and Write times. The Wave form generator will generate the pulses necessary to perform manual operations. The pulses normally generated in the USE mode will be suppressed unless, as in a few cases, they will not affect the operation of the machine while in manual.

## 705 Address System

The address system used in the 705 is a four digit address which will designate any position in memory, an input/output unit or any of the addressable controls. This would indicate that the addresses on the 705 lie between 0000 and 9999. With this in mind and also that the 705 has 20,000 positions of memory storage, some special method has to be used to designate memory positions 10,000 to 20,000. The zone of the 1000's order of the address is used to indicate whether a memory address falls in the upper or

lower 10,000 positions. To be more precise, no zone of the 1000's order of the address designates an address between 0000 and 9999, while any zone (01, 10, or 11) in this position designates an address between 10,000 and 19,999.

It is also necessary to decide between the 256 position accumulator storage and any of the other 15 auxiliary storage units. The zone portion of the 10's and 100's order position of the address is reserved to make this decision. No zoning in both the 10's and 100's order will select the 256 position storage. The 15 auxiliary storage units are selected according to a binary coded system as indicated in the following table.

| Storage or Auxiliary Storage | | Zone of Address | | |
|---|---|---|---|---|
| Number | Positions | Thousands | Hundreds | Tens | Units |
| 00 | 256 | | 00 | 00 | |
| 01 | 16 | | 00 | 01 | |
| 02 | 16 | | 00 | 10 | |
| 03 | 16 | | 00 | 11 | |
| 04 | 16 | | 01 | 00 | |
| 05 | 16 | | 01 | 01 | |
| 06 | 16 | | 01 | 10 | |
| 07 | 16 | | 01 | 11 | |
| 08 | 16 | | 10 | 00 | |
| 09 | 16 | | 10 | 01 | |
| 10 | 16 | | 10 | 10 | |
| 11 | 16 | | 10 | 11 | |
| 12 | 16 | | 11 | 00 | |
| 13 | 16 | | 11 | 01 | |
| 14 | 16 | | 11 | 10 | |
| 15 | 32 | | 11 | 11 | |

A further stipulation is placed on the address system in the 705-namely, the units order of the address of an instruction must either be the character 4 or 9. This condition is brought about by the type of memory in the 705, where all five characters of an instruction can be read in one character cycle.

Instructions:

Each of the operations that the type 705 can perform is broken down into 2 parts

Instruction time: (I time)

Instruction time is the first part of the operation where the machine takes the instruction out of memory and makes a decision as to what to do and where to go in memory and storage to operate.

Execution time: (E time)

Execution time is the second part of the operation where the machine uses data that was located during I time and does the operation.

Refer to Figure on Memory Address Controls

Instruction Time

During I time it is desired to take an instruction that is stored in memory and prepare the machine to operate under the control of this instruction. Let us say that the first instruction is stored in location 0004.

Push the reset key.

Resets type cycle triggers, special triggers and resets the instruction counter (IC) to 0004.

Push the start key.

This will allow the machine to take one more idle cycle and then start into the first instruction time. During the last idle cycle MACI will be set to IC. Because the machine now goes into an instruction use cycle (I time) the following will be done automatically.

Set MASR to MACI (RO)

MASR controls the location in memory to be read out. MASR controls the X and Y switching so that during read time from $R_2$ to $R_4$ the 35 cores in memory selected by X and Y switching are reset. Any of the 35 selected cores, (locations 0000 to 0004), that were set will send out an impulse to turn on the memory buffer register triggers. The memory buffer register triggers now contain the same information that was contained in the 35 selected memory cores. Because it is I time, character 4 is routed through memory out control to the units order of MAR and to CR1.

Set the operation register ($A_o$)

The operation register is set to character "0" of the buffer register.

Set CR1 ($A_o$)

CR1 is set to character 4 of the buffer register.

Set MAR ($A_5$)

The units order of MAR is set with the same lines that were used to set CR1. The memory out control will always put character 4 of the buffer register on the memory out bus during I time. The tens, hundreds and thousands positions of MAR will be set to the numeric parts of buffer register characters 1, 2 and 3. If there is a zone in buffer register character 1 position, the zone will set the 10,000 order of MAR. MAR will now contain the address part of the instruction that was located in 0000-0004 of memory.

Step IC ($A_6$)

The instruction counter is stepped 5. This will step it 5 plus the 0004 that was in it or step it to 0009. This is the address of the next sequential instruction and will be saved in the instruction counter until following I time.

Set MACI to MAR ($W_1$)

MACI is now set to the address that is contained in MAR.

Route memory buffer register back into memory.

The information that was taken out of memory during read time is now routed through the memory in control and put back into the same place in memory. MASR is still holding the address used during read time so that the 5 characters taken out of memory will be put back in the same place.

CPU has now completed all the functions that have to do with memory during instruction time. An instruction stored in memory has been taken out. The operation part has been stored in the op. register and the address part has been stored in MAR and MACI.

The machine now will go into execute time under control of the operation register. It will stay in execute time until the operation being done is completed. At this time during the last cycle of the operation, the operation will route an end execute. End execute will set the machine to I time and set MACI to the IC. At $R_o$ of I time MASR will be set to MACI and instruction time will be repeated.

Instruction Time (Refer to figure on storage address controls)

SPC

Since any field in storage is not used during I time, the only thing that we are interested in during I time is to determine whether aux. storage or storage is to be used during the execution of the instruction.

SPC

The starting point counter is an 8 stage binary counter. The purpose of this counter is to indicate the units position of the field in storage. Since there is no field in storage initially SPC can assume any random number.

Set SSR ($A_0$)

At $A_0$ of I time the 4 triggers in SSR will be set to the zones of character 2 & 3 of memory buffer register. These zones will determine the aux. storage unit to be used during E time. If there is no zone in the positions, the main storage will be used.

SET SAC to SPC or to SSR ($W_1$).
SET SAC to SPC.

If there are no bits stored in SSR there will be no input to the select aux. storage circuits. The select aux. storage circuits control X and Y switching and also control the SAC setting circuits. If there is no input to the select aux. storage circuits then SAC is controlled to set to SPC.

Set SAC to SSR.

If there are bits in SSR, the select aux. storage circuits will control SAC in such a manner that the 16, 32, 64 & 128 triggers are set to the information contained in the 4 triggers in SSR. The select aux. storage circuits further control SAC so that the 1, 2, 4, & 8 triggers are reset off. SAC now contains either the address of some aux. storage unit or the address in main storage of the units position.

Set SASR to SAC ($R_0$).

At $R_0$ time of the first execute cycle SASR will be set to SAC and through the X and Y switching will selecte some character in storage. During read time time of this first execute cycle that character will be brought out and set into the storage buffer register.

Execute Time

During execution time CPU will operate under control of the op. register. The general flow of information will be from either or both memory and storage to their respective buffer registers. From here the information in buffer registers will be sent to CR1. The information from storage buffer register will be sent to CR2. CR1 and/or CR2 will be sent to the adder circuits and a single result is generated. This result will be set in the result register and then be sent to either storage or memory.

Under control of the op. register, lines will be routed to operate the various functions that the machine can perform. The lines that can operate on memory are as follows:

Read Memory

If this line is routed, 5 characters will be read from memory and stored in the memory buffer register during read time.

If CPU is operating in RAW time, CR1 will be set to the memory out busses at $A_O$.

If CPU is operating in RW time and either WRITE REQUEST or TRANSMIT 1, CR1 will be set to the memory out busses at Wo.

If the machine is operating in RW time and either READ REQUEST or TRANSMIT 2, CR1 will not be reset or set to the memory out busses.

The character placed in CR1 is controlled by the address in MASR.

READ STORAGE

If this line is routed, 1 character will be read from storage and stored in the storage buffer register during read time.

If CPU is operating in RAW time, CR2 will be set to the Storage out busses at Ao.

If CPU is operating in RW time and either WRITE REQUEST or TRANSMIT 1, CR2 will be set to the Storage out busses at Wo.

If CPU is operating in RW time and either READ REQUEST or TRANSMIT 2, CR2 will not be set to the Storage out busses.

ERASE MEMORY

If this line is routed, 5 characters will be read from memory but will not be stored in the buffer register. Routing this line will also suppress the resetting of the memory buffer register. Thus the buffer register will contain the 5 characters stored there during the previous read time and the 5 characters read out during the current read time will have been erased.

RESULT TO MEMORY:

If this line is routed, the character stored in the result register will be written into memory at the address specified by the memory address select register. Four of the five characters stored in the memory buffer register will also be written into memory at their former addresses. If this line is not routed, the 5 characters stored in the memory buffer register will be written back into memory at their former addresses.

RESULT TO STORAGE

If this line is routed, the character stored in the result register will be written into the storage at the address specified by the storage address select register.

If this line is not routed, the character stored in the Storage buffer register will be written into the storage at the addresss specified by the Storage Address Select Register.

The memory and storage address controls are under the control of instruction time and under control of the operation being performed during execute time. A summary of these counters and registers and how they are operated follows.

Memory Address Register (MAR)

The memory address register is a 17 stage trigger register representing a 5 digit binary coded decimal address. (The units, tens, hundreds and thousands orders each consist of four triggers which store one binary coded decimal digit in 1, 2, 4, 8 code form. The ten-thousands order consists of only one trigger to store a 1 or a O in that order.) MAR will contain the address part of the instruction being executed and will receive its information from one of two sources. In automatic operation MAR will be set to the Memory Buffer Register outputs during instruction time. I TIME). In manual operation MAR will be set to the Memory Address Selector switches located on the console.

The inputs to MAR from the memory buffer register (MBR) will be connected as follows:

| | | |
|---|---|---|
| MAR - 1 | Mem. Out Bus - 1 | (MBR 4-1) |
| MAR - 2 | Mem. Out Bus - 2 | (MBR 4-2) |
| MAR - 4 | Mem. Out Bus - 4 | (MBR 4-4) |
| MAR - 8 | Mem. Out Bus - 8 | (MBR 4-8) |
| MAR - 10 | MBR 3-1 | |
| MAR - 20 | MBR 3-2 | |
| MAR - 40 | MBR 3-4 | |
| MAR - 80 | MBR 3-8 | |
| MAR - 100 | MBR 2-1 | |
| MAR - 200 | MBR 2-2 | |
| MAR - 400 | MBR 2-4 | |
| MAR - 800 | MBR 2-8 | |
| MAR - 1000 | MBR 1-1 | |
| MAR - 2000 | MBR 1-2 | |
| MAR - 4000 | MBR 1-4 | |
| MAR - 8000 | MBR 1-8 | |
| MAR - 10,000 | MBR 1-A and MBR 1-B | |

The outputs of MAR will be used to set addresses into the memory address counters, the instruction counter and the selection register. The units order of MAR will also provide outputs for the Control instruction.

Memory Address Counter 1 (MAC 1)

The Memory Address Counter is a 17 stage trigger counter representing a 5 digit binary coded decimal address. It controls the memory address of the character being read from or written into memory during any given machine cycle.

MAC 1 will perform the following stepping functions:

Step + 1
Step - 1
Step + 5

When stepping + 5, MAC 1 will have previously been set to an address which has a four or a nine as its units order digit.
The stepping of MAC 1 will be controlled by the CPU sequence control circuits by conditioning one of the following routing lines. The voltage level of the line will be +10 when routed and not higher than -20 when not routed.

Step MAC 1 + 1
Step MAC 1 - 1
Step MAC 1 + 5

MAC 1 will have circuitry for setting its triggers to the contents of the Memory Address Register (MAR) or the Instruction Counter (IC). This will be done under the control of the CPU sequence circuits by conditioning one of the following routing lines. The voltage level of the line will be +10 when routed and not higher than -20 when not routed.

Set MAC 1 to MAR
Set MAC 1 to IC

Only one of the five routing lines for controlling MAC 1 will be routed by the CPU sequence control circuits during any given machine cycle.

Memory Address Counter - 2 (MAC 2)

MAC 2 will also determine the memory address during certain type machine cycles under the control of CPU. MAC 2 will step +1 and +5 and will have circuitry for setting its triggers to the contents of the memory address register. The following routing lines will control the stepping or setting of MAC 2.

Step MAC 2 +1
Step MAC 2 +5
Reset and set MAC 2 to MAR

This counter will be reset to 00000 before setting. No more than one of the three lines will be routed during any one machine cycle.

Memory Address Select Register (MASR)

MASR is a 17 stage trigger register and stores the memory address of the character being read from memory. The MASR triggers will be set to the contents of either MAC 1 or MAC 2 at $R_o$ time. No routing lines will be used to control MASR. The setting pulses for setting to MAC 1 or MAC 2 will be generated automatically in the Waveform Generator.

Instruction Counter (IC)

The Instruction Counter keeps track of the address of the instruction to be executed next. It is a 14 stage counter which represents a 5 digit binary coded decimal address. The units and ten-thousands order consist of 1 trigger each. The units order trigger represents a 4 in the units order when it is off and a 9 in the units order when it is on. The ten-thousands order trigger will represent a 1 or a 0 in that order.

IC will have circuitry for setting its triggers to the contents of the memory address register. All triggers will be reset-off before setting. IC will step +5 during each I time cycle. The setting of IC to MAR will be controlled by the routing line Reset and Set IC to MAR. This routing line will be conditioned during end execute time only.

Starting point Counter (SPC)

SPC is an eight stage binary counter which controls the starting point of storage (coded 00). This counter can step +1, -1 and 128 under the control of routing lines:

Step SPC +1
Step SPC -1
Step SPC 128

Storage Select Register (SSR)

The SSR is a 4 stage trigger register which determines which of the 16 Aux. storage will be used during execution time. It is set during I time to the zone bits (A & B) of both the tens and hundreds orders of the address part of the instruction.

The inputs of SSR from the memory buffer-register will be connected as follows:

| | |
|---|---|
| SSR - 1 | MBR 3-A |
| SSR - 2 | MBR 3-B |
| SSR - 4 | MBR 2-A |
| SSR - 8 | MBR 2-B |

This register will be reset at $R_o$ during instruction time and will be set at $A_o$.

Storage Address Counter (SAC)

The SAC is an eight stage binary counter which controls the address

of the character being read from storage. It can step +1, -1 and 128 under the control of routing lines:

Step SAC +1
Step SAC -1
Step SAC 128

The seting of SAC will be under the control of the routing line, SET SAC to SPC. When this line is routed and SSR is 0000 coded, SAC will set to SPC. If Set SAC to SPC is routed and SSR is not 0000 then the 128, 64, 32 and 16 stages of SAC will be set to the 8, 4, 2, and 1 stages of SSR respectively and the 8, 4, 2, 1 stages of SAC will be set to 0000.

Only one of the stepping or setting routing lines will be conditioned at the same time.

Storage Address Select Register (SASR)

SASR is an eight stage register which stores the address of the character being read from storage. SASR will be set to the contents of SAC at $R_o$ of every machine cycle.

Select Register (SR)

The SR is a 16 stage trigger register which holds the address of the Input-Output unit selected. It is set under the control of the routing line. Reset and Set SR to MAR. When this line is routed the select register will be reset to 0000 at $R_o$ and then set to MAR at W2 time. The output of each of its triggers will drive circuitry in the Input-Output units.

Timing-Stepping

During a 17 usec. use cycle the counters will be stepped by a (-) A6-W1 pulse, the stepping occurring at A 6 time.

During a 9 usec. use cycle the counters will be stepped by a (-) R3-W1 pulse, the stepping occuring at R3 time.

Both of these pulses will appear on the same line coming to the counters from the waveform generator. However they will be generated as positive pulses and inverted at the counters.

Timing-Setting

During a 17 usec. use cycle the counters will be set with an A6-W1 pulse, the setting occuring at W1 time. During a 9 usec. use cycle the counters will be set by an R3-W1 pulse, the setting occuring at W1 time.

Timing-Reset

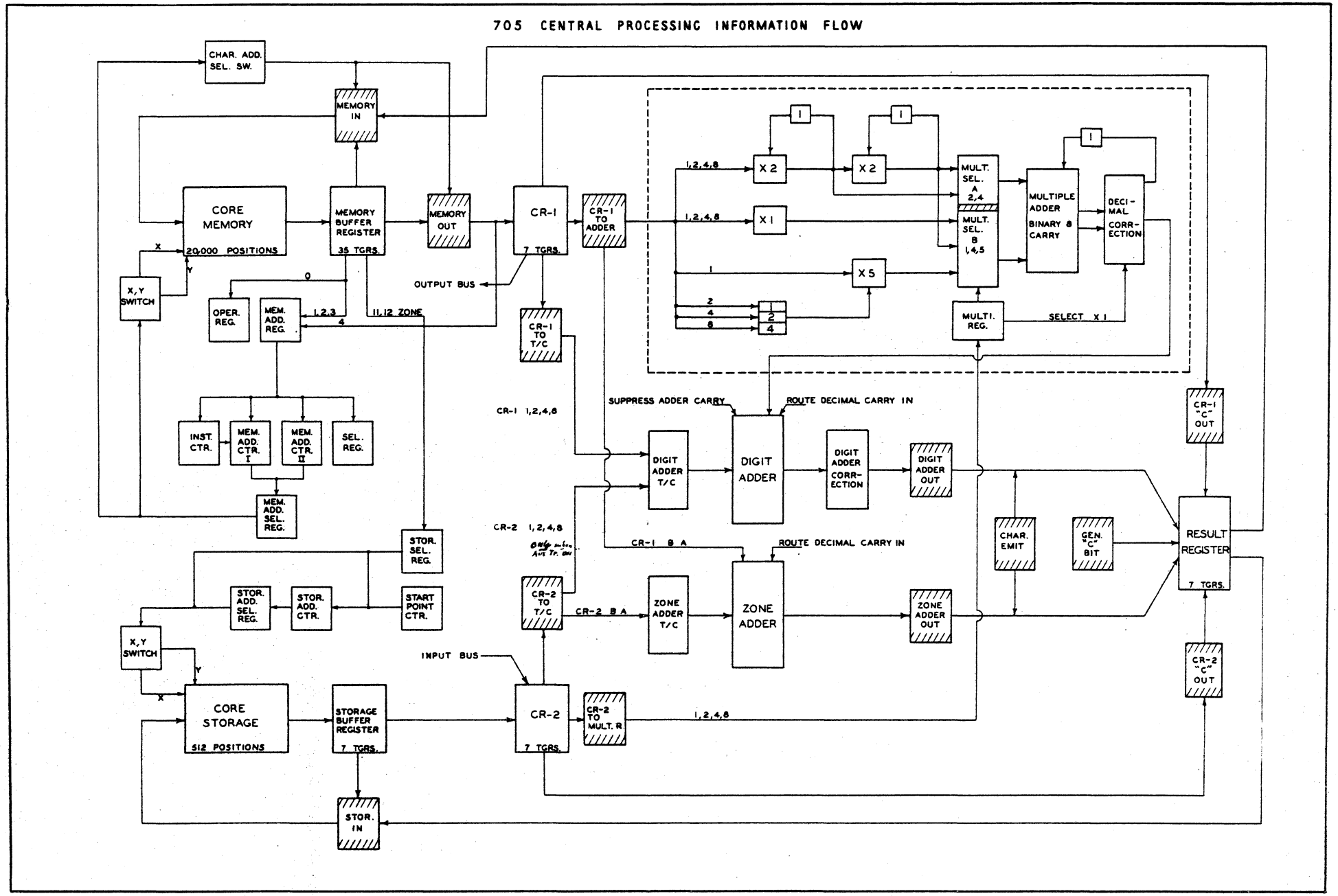
MAR will be reset by Instruction Reset, which will occur only during I time at $R_o$ (D2).

IC will be reset only when Reset and Set IC to MAR is routed. The resetting will be done with an $R_o$ (D2) pulse.

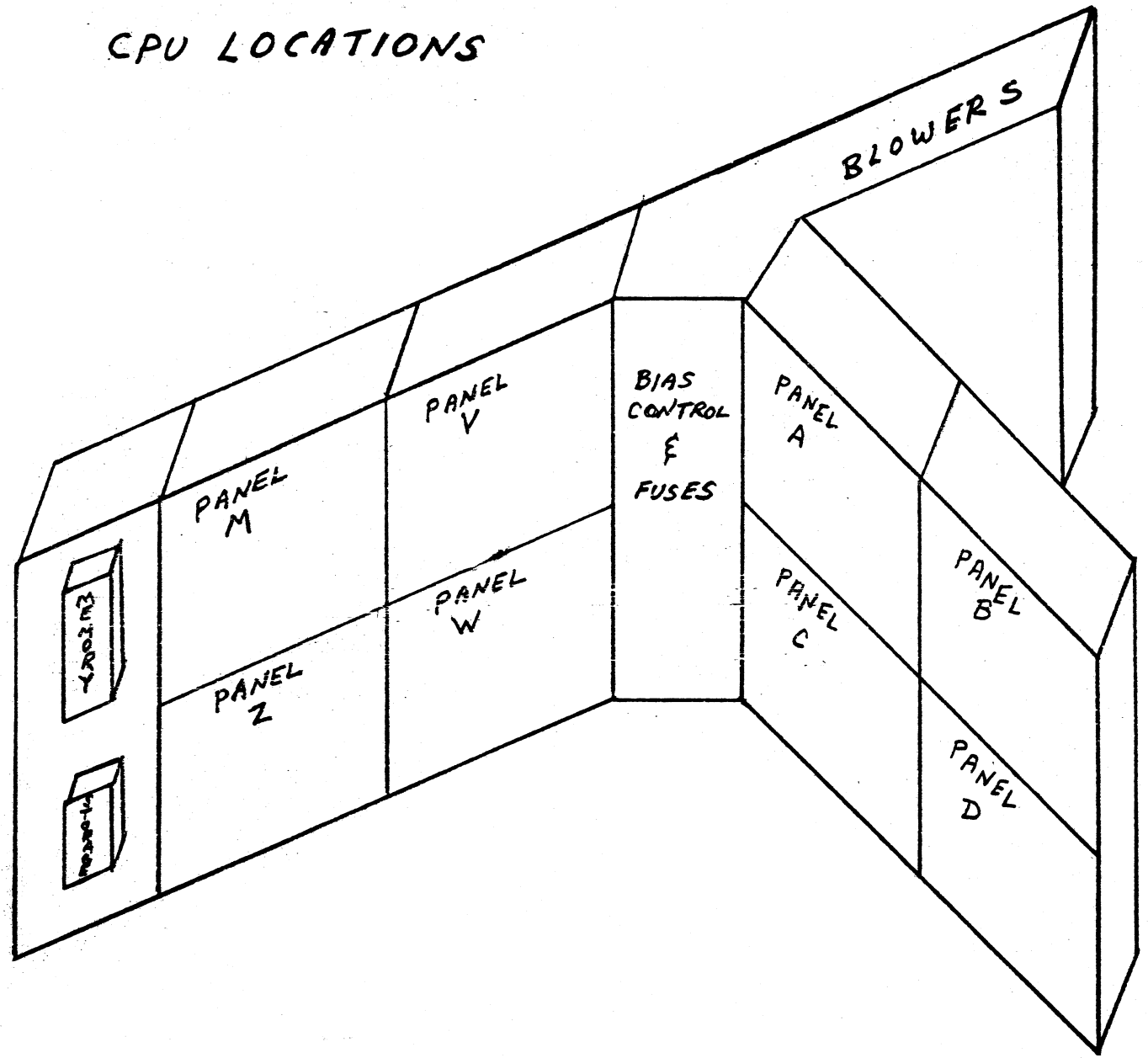
SR will be reset only when Reset and Set SR to MAR is routed. The resetting will be done with an $R_o$ (D2) pulse.

MAC 2 will be reset only when Reset and Set MAC 2 to MAR is routed. The resetting will be done with an $R_o$ (D2) pulse.

SSR will be reset by Instruction Reset during I time.

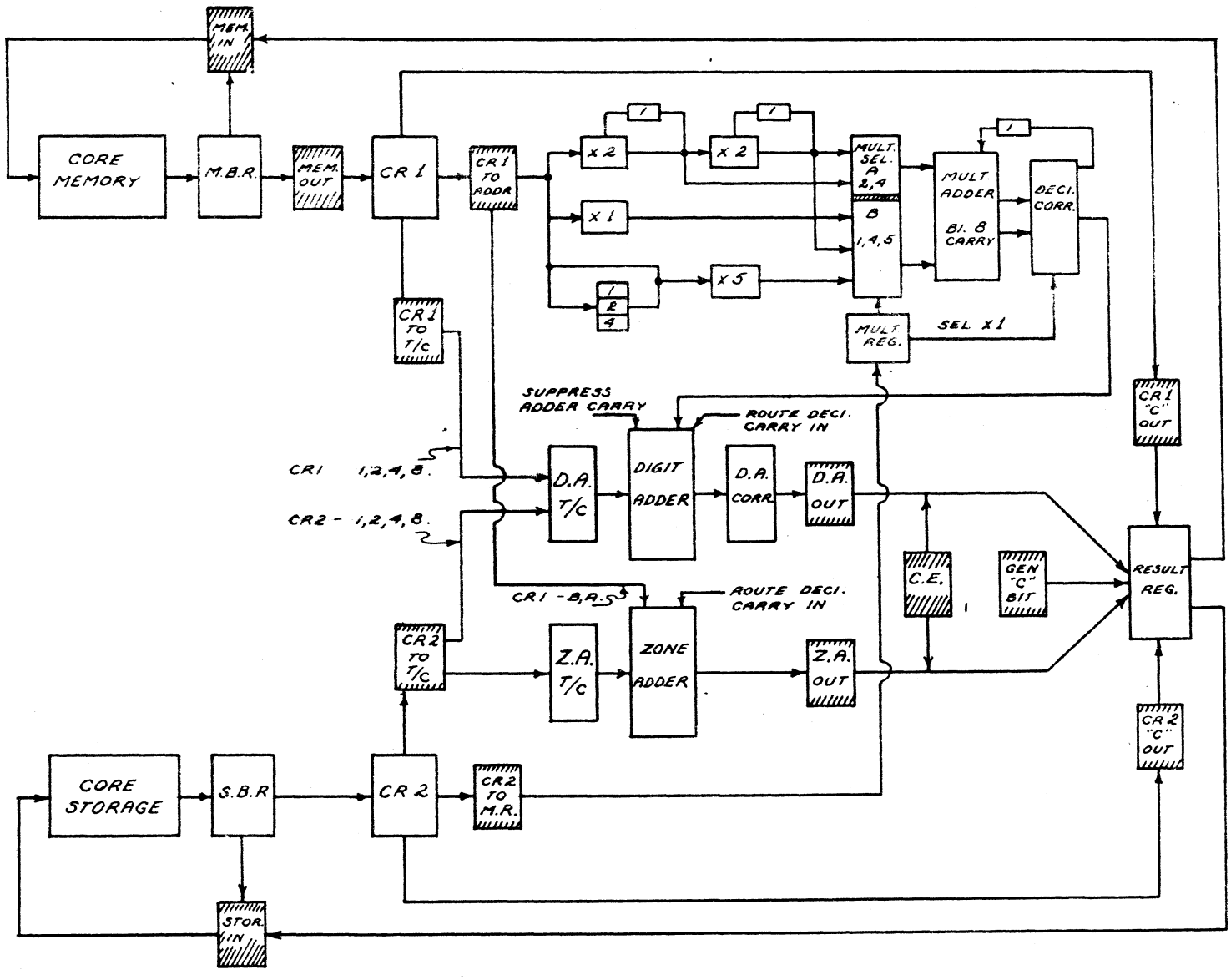

705 CENTRAL PROCESSING INFORMATION FLOW
CHAR. ADD. SEL. SW.
MEMORY IN
CORE MEMORY
20,000 POSITIONS
MEMORY BUFFER REGISTER
35 TGRS.
MEMORY OUT
CR-1
7 TGRS.
CR-1 TO ADDER
OUTPUT BUS
X, Y SWITCH
OPER. REG.
MEM. ADD. REG.
1,2,3
4
11,12 ZONE
CR-1 TO T/C
1,2,4,8
X 2
X 2
X 1
X 5
MULT. SEL. A 2,4
MULT. SEL. B 1,4,5
MULTIPLE ADDER BINARY & CARRY
DECI-MAL CORR-ECTION
MULTI. REG.
SELECT X 1
INST. CTR.
MEM. ADD. CTR. I
MEM. ADD. CTR. II
SEL. REG.
MEM. ADD. SEL. REG.
CR-1 1,2,4,8
SUPPRESS ADDER CARRY
ROUTE DECIMAL CARRY IN
DIGIT ADDER T/C
DIGIT ADDER
DIGIT ADDER CORR-ECTION
DIGIT ADDER OUT
CR-1 "C" OUT
CR-2 1,2,4,8
CR-1 B A
ROUTE DECIMAL CARRY IN
CHAR. EMIT
GEN. "C" BIT
RESULT REGISTER
7 TGRS.
STOR. SEL. REG.
STOR. ADD. SEL. REG.
STOR. ADD. CTR.
START POINT CTR.
CR-2 TO T/C
CR-2 B A
ZONE ADDER T/C
ZONE ADDER
ZONE ADDER OUT
CR-2 "C" OUT
INPUT BUS
X, Y SWITCH
CORE STORAGE
512 POSITIONS
STORAGE BUFFER REGISTER
7 TGRS.
CR-2
7 TGRS.
CR-2 TO MULT. R
1,2,4,8
STOR. IN

# CPU LOCATIONS



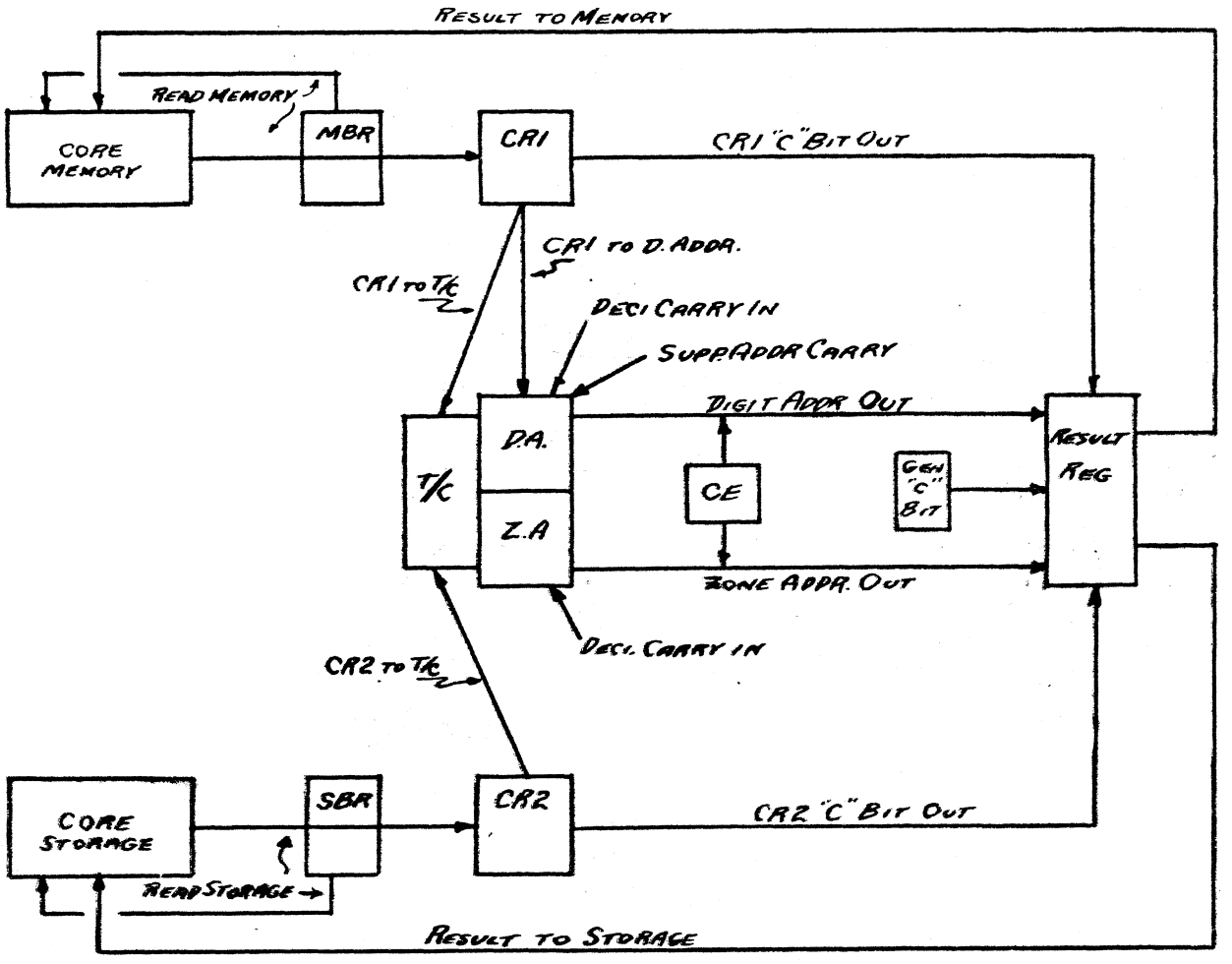

TYPE 705
G.O.L.F.

2/9/55


MEM. IN
CORE MEMORY
M.B.R.
MEM. OUT
CR 1
CR 1 TO ADDR
x 2
x 2
x 1
x 5
1
2
4
MULT. SEL. A 2,4
B 1,4,5
MULT. ADDER
Bi. 8 CARRY
DECI. CORR.
MULT REG.
SEL X1
CR 1 TO T/C
SUPPRESS ADDER CARRY
ROUTE DECI. CARRY IN
CR1 1,2,4,8.
CR2 - 1,2,4,8.
D.A. T/C
DIGIT ADDER
D.A. CORR.
D.A. OUT
CR 1 "C" OUT
CR1 -B,A,5
ROUTE DECI. CARRY IN
C.E.
GEN "C" BIT
RESULT REG.
CR2 TO T/C
Z.A. T/C
ZONE ADDER
Z.A. OUT
CR 2 "C" OUT
CORE STORAGE
S.B.R
CR 2
CR2 TO M.R.
STOR. IN

HDB. 3-29-54

# 705 Routing




1 triggers - Hold 1 character

HDB
3-31-55

SCS 18A OBSOLETE
705 CORE STORAGE DIAGNOSTIC

1.0 GENERAL

1.1 CS 19A is based on the fact that in a core plane, half selected cores are undergoing a change of flux and are inducing voltages into the sense winding. This induced voltage can subtract from a wanted signal, i.e., a selected core containing a one; or it can build up to a value to be sensed as a one, even though the selected core contained a zero.

1.2 The test fills memory with characters arranged so that if a selected core contains a "one", as many as possible of the half selected cords put out opposing voltages, and if the selected core has a zero, the other half selected cores put out aiding voltages to try to add up to a "one" pulse.

1.3 The test reads five characters at a time by means of a compare instruction, however, the result of the comparison is not used as an error indication, rather the CR-1 code check and machine check circuits are the error indicators. In the case of the 1-6 pattern, the 1 bit, 2 bit and 4 bit planes are being tested. If the five characters interrogated are ones, the two bit and four bit cores were at zero. Too much noise on those two planes could cause erroneous read out and a redundant character would be sensed. After the interrogation, the ones are replaced by sixes, setting the two and four bit cores, and putting the one bit core at zero. The same five characters are interrogated again by compare. Thus each of the cores in the three planes is checked for its performance, at one and at zero.

2.0 DESCRIPTION OF PROGRAM

2.1 From addresses 18009 to 18154 the program loads auxiliarystorage units to be used as counters or for address modification. Reference will be made to them as they are used.

2.2. Addresses 18159 to 18269 are used to fill 18000 positions of memory with the test pattern.

2.2.1 The basic steps in filling memory are those between 18169 and 18199. This is a loop, wherein 100 characters are transmitted to ten successive fields in memory, resulting in a 1000 character field.
The initial receive address is set from ASU 1.
The receive address is increased by 100 from ASU 4.
The counter for keeping count of the ten passes is ASU 3. A successful TRZ addressed to auxiliary storage signals the completion of ten passes through the loop.

2.2.2 A larger loop, containing the basic loop is controlled by ASU 2. This ASU is reduced by one for each 1000 character pass, and interrogated by a TRZ. If the TRZ does not occur a repetition of the 1000 character loop is made. The result of this loop is to place 4000 characters in memory.

Each 1000 character pass transmits a slightly different configuration of the test pattern as the TMT address is increased by 105 in this loop. (The 105th character is a record mark - however, as the RCV address is increased by only 100, athe pattern is entered successively.) The TMT address is increased by an add memory instruction, from ASU 5.

2.2.3 ASU 1 exercises control over the two previously described loops. It contains a four, and is reduced by one for each pass through the second loop (2.2.2). It is compared against a constant 2:
After the first 4000 character pass a TRH occurs, resetting ASU 2 and ASU 3 for another 4000 characters.
After the second 4000 characters a TRE occurs, setting ASU 2 to a 2, and ASU 3 to 10 to cause only 2000 characters to be transmitted.
After the third (2000) pass, ASU 2 and ASU 3 are reset to 4 and 10 respectively, to transmit another 4000 characters.

The fourth and fifth passes are the same as the first two.

After the fifth pass, a TRP at address 18229 now finds ASU 1 negative and transfers to the testing portion of the program.

After each major pass, ASU 12 restores the TMT address to the initial address of the first test group.

2.3 Testing is accomplished by comparing five characters of memory against either ASU 7 or 8, which have been loaded with the test characters. What test characters are being used is signalled by ASU 11, initially loaded with a 3. Compared against a constant 2 a TRH signals the first pattern (1-6), a TRE the second (6-H), and NO TRH or TRE, the third pattern (1-H0.

2.3.1 At step 18334 a compare is made against ASU 8 containing, for example, five ones. Errors would cause CR-1 code checks and machine checks.

If a TRE can occur, memory is set to sixes and compared against ASU 8 again. Errors would again be CR-1 code checks and machine checks.

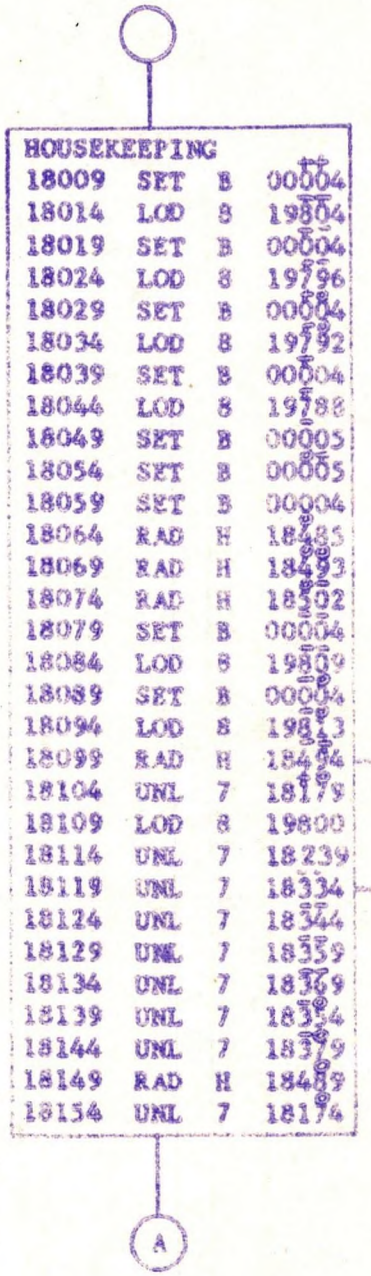
Thus the addressed cores in planes 1-2-4 of each character have been tested when they contained ones and the zeros. The other ones and sixes set up the condition calculated to give the most noise on the sense winding.

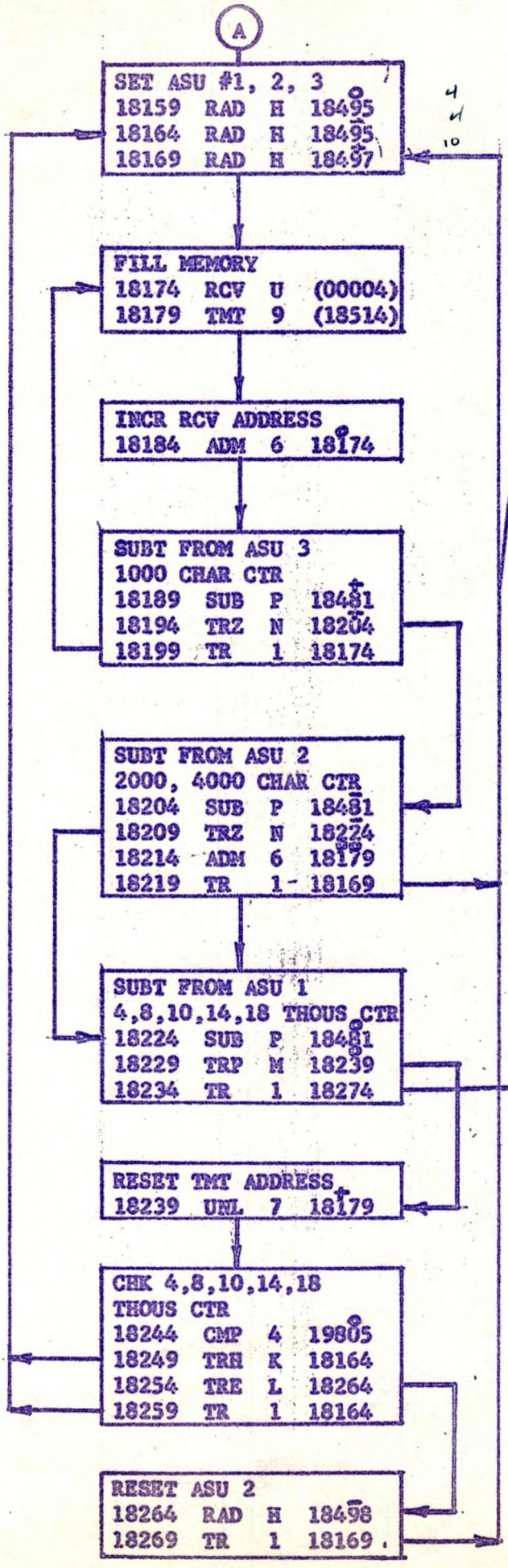
2.3.2 The compare instructions and the unload instructions shown by asterisks are increased by 005 from ASU 9 and 10, interrogate the next five characters. Also, ASU 1 which now contains 3600 is reduced by one. When it becomes a zero, it signals that all 18000 characters have been checked. A TRZ is made to 18429.

2.3.3 At 18429 the TMT address is increased by 105. It had been left at the beginning of the last group of the test pattern just finished. Adding 105 to it puts it at the beginning of the first group of the new test pattern.

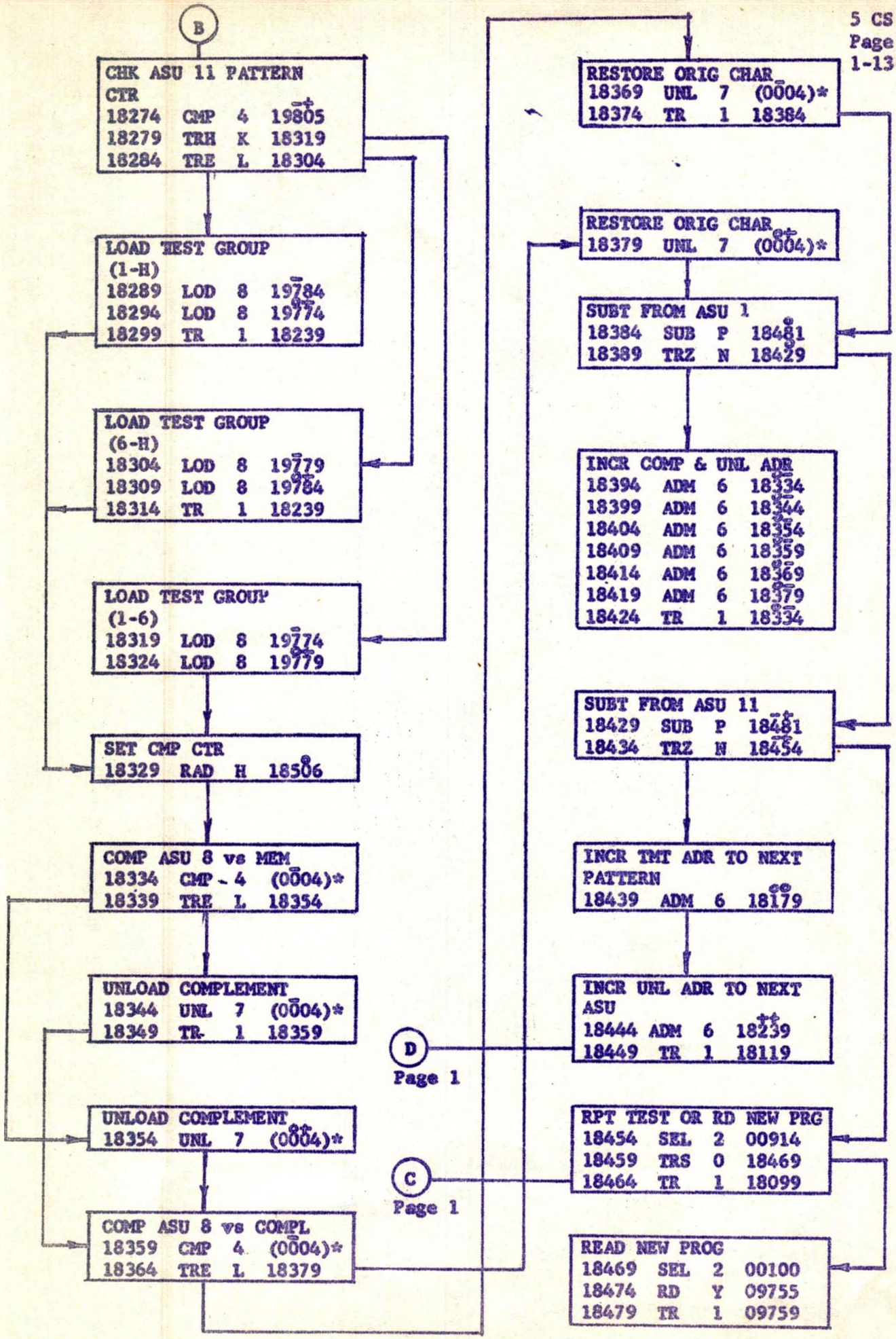
In step 18444 an ADM instruction changes the auxiliary storage unit address used in step 18104, to now unload the initial address of the new test pattern.

2.3.4 After all three test patterns have been used, it is possible to repeat the test, or to call on the reader for the next program, under control of Alteration switch 00914.


HOUSEKEEPING
18009 SET B 00004
18014 LOD 8 19804
18019 SET B 00004
18024 LOD 8 19796
18029 SET B 00004
18034 LOD 8 19792
18039 SET B 00004
18044 LOD 8 19788
18049 SET B 00005
18054 SET B 00005
18059 SET B 00004
18064 RAD H 18485
18069 RAD H 18493
18074 RAD H 18502
18079 SET B 00004
18084 LOD 8 19809
18089 SET B 00004
18094 LOD 8 19813
18099 RAD H 18494
18104 UNL 7 18179
18109 LOD 8 19800
18114 UNL 7 18239
18119 UNL 7 18334
18124 UNL 7 18344
18129 UNL 7 18359
18134 UNL 7 18369
18139 UNL 7 18384
18144 UNL 7 18379
18149 RAD H 18489
18154 UNL 7 18174
C
Page 3
D
Page 3
A


A
SET ASU #1, 2, 3
18159 RAD H 18495
18164 RAD H 18495
18169 RAD H 18497
FILL MEMORY
18174 RCV U (00004)
18179 TMT 9 (18514)
INCR RCV ADDRESS
18184 ADM 6 18174
SUBT FROM ASU 3
1000 CHAR CTR
18189 SUB P 18481
18194 TRZ N 18204
18199 TR 1 18174
SUBT FROM ASU 2
2000, 4000 CHAR CTR
18204 SUB P 18481
18209 TRZ N 18224
18214 ADM 6 18179
18219 TR 1 18169
SUBT FROM ASU 1
4,8,10,14,18 THOUS CTR
18224 SUB P 18481
18229 TRP M 18239
18234 TR 1 18274
B
RESET TMT ADDRESS
18239 UNL 7 18179
CHK 4,8,10,14,18
THOUS CTR
18244 CMP 4 19805
18249 TRH K 18164
18254 TRE L 18264
18259 TR 1 18164
RESET ASU 2
18264 RAD H 18498
18269 TR 1 18169

Page 3


B
CHK ASU 11 PATTERN CTR
18274 CMP 4 19805
18279 TRH K 18319
18284 TRE L 18304
LOAD TEST GROUP (1-H)
18289 LOD 8 19784
18294 LOD 8 19774
18299 TR 1 18239
LOAD TEST GROUP (6-H)
18304 LOD 8 19779
18309 LOD 8 19784
18314 TR 1 18239
LOAD TEST GROUP (1-6)
18319 LOD 8 19774
18324 LOD 8 19779
SET CMP CTR
18329 RAD H 18506
COMP ASU 8 vs MEM
18334 CMP 4 (0004)*
18339 TRE L 18354
UNLOAD COMPLEMENT
18344 UNL 7 (0004)*
18349 TR 1 18359
UNLOAD COMPLEMENT
18354 UNL 7 (0004)*
COMP ASU 8 vs COMPL
18359 CMP 4 (0004)*
18364 TRE L 18379
D
Page 1
C
Page 1
RESTORE ORIG CHAR
18369 UNL 7 (0004)*
18374 TR 1 18384
RESTORE ORIG CHAR
18379 UNL 7 (0004)*
SUBT FROM ASU 1
18384 SUB P 18481
18389 TRZ N 18429
INCR COMP & UNL ADR
18394 ADM 6 18334
18399 ADM 6 18344
18404 ADM 6 18354
18409 ADM 6 18359
18414 ADM 6 18369
18419 ADM 6 18379
18424 TR 1 18334
SUBT FROM ASU 11
18429 SUB P 18481
18434 TRZ N 18454
INCR TMT ADR TO NEXT PATTERN
18439 ADM 6 18179
INCR UNL ADR TO NEXT ASU
18444 ADM 6 18239
18449 TR 1 18119
RPT TEST OR RD NEW PRG
18454 SEL 2 00914
18459 TRS 0 18469
18464 TR 1 18099
READ NEW PROG
18469 SEL 2 00100
18474 RD Y 09755
18479 TR 1 09759

## CONSTANT FIELDS

```
18481          &   1
18485          &  0100
18489          &  0004
18493          &  0105
18494          &  3
18495          &  4
18497          &  10
18498          &  2
18502          &  0005
18506          &  3600
18534             1111166666666661111166666
18559             1111111111666666666611111
18584             1111166666111116666666666
18609             1111111111666666666611111
18614                    S  - RM
18639             6666611111111116666666666
18664             1111111111666661111166666
18689             6666611111111116666666666
18714             1111166666111111111166666
18719                    S  - RM
18744             6666611111111116666611111
18769             6666666666111111111166666
18794             6666611111666661111111111
18819             6666666666111111111166666
18824                    S  - RM
18849             1111166666666661111111111
18874             6666666666111116666611111
18899             1111166666666661111111111
18924             6666611111666666666611111
18929                    S  - RM
18954             66666HHHHHHHHHH66666HHHHH
18979             6666666666HHHHHHHHHH66666
19004             66666HHHHH66666HHHHHHHHHH
19029             6666666666HHHHHHHHHH66666
19034                    S  - RM
19059             HHHHH6666666666HHHHHHHHHH
19084             6666666666HHHHH66666HHHHH
19109             HHHHH6666666666HHHHHHHHHH
19134             66666HHHHH6666666666HHHHH
19139                    S  - RM
19164             HHHHH6666666666HHHHH66666
19189             HHHHHHHHHH6666666666HHHHH
19214             HHHHH66666HHHHH6666666666
19239             HHHHHHHHHH6666666666HHHHH
19244                    S  - RM
19269             66666HHHHHHHHHH6666666666
19294             HHHHHHHHHH66666HHHHH66666
19319             66666HHHHHHHHHH6666666666
19344             HHHHH66666HHHHHHHHHH66666
19349                    S  - RM
```

```
19374                   HHHHH1111111111HHHHH11111
19399                   HHHHHHHHHH1111111111HHHHH
19424                   HHHHH1111HHHHH1111111111
19449                   HHHHHHHHHH1111111111HHHHH
19454                         S - RM
19479                   11111HHHHHHHHHH1111111111
19504                   HHHHHHHHHH11111HHHHH11111
19529                   11111HHHHHHHHHH1111111111
19554                   HHHHH11111HHHHHHHHHH11111
19559                         S - RM
19584                   11111HHHHHHHHHH11111HHHHH
19609                   1111111111HHHHHHHHHH11111
19634                   11111HHHHH11111HHHHHHHHHH
19659                   1111111111HHHHHHHHHH11111
19664                         S - RM
19689                   HHHHH1111111111HHHHHHHHHH
19714                   1111111111HHHHH11111HHHHH
19739                   HHHHH1111111111HHHHHHHHHH
19764                   11111HHHHH1111111111HHHHH
19769                         S - RM
19774                   11111
19779                   66666
19784                   HHHHH
19788 18514  8514
19792 18934  8934
19796 19354  9354
19800 18179  8A79
19804                   0000
19805                   2
19809                   0004
19813                   0004
```

# ACCUMULATOR AND AUXILIARY STORAGE TEST

PURPOSE: To set up a pattern to get the worst possible noise conditions. To do this, a 512 character pattern is developed in memory and then loaded and unloaded. Noise conditions will cause redundant characters to be unloaded, giving CRZ code checks, and machine stop. Each position of storage is complemented to check its read out at "zero as well as at "one"

Program Steps:

00244 Locate 054, print identification, halt.

00294 Set a three into pattern indicator, and load addresses of first pattern, unload into pattern generator

00329 Develop pattern, expanding it to 512 characters.

00469 Set counter for signaling end of pattern.

00479 Load address of beginning of pattern.

00504 Compare pattern indicator against a two. A TRH indicates first pattern. A TRE indicates the second pattern and no transfer indicates the third pattern.

00569 Load a one, then a six into Accumulator. Compare against first character in pattern. Depending on result of compare - a complement character will be unloaded into memory position.

00614 Set left 255 in Accumulator and 255 in ASU 1. Load pattern and then unload. A redundant character read out on the unload operation will be the indication of a failure and will cause a machine stop. The character in error will be displayed in the CRZ neons and the address of the incorrect reading will be displayed in the SAC neons.

00694 Increase complementing addresses to next position. Transfer to 00569 to repeat this part of the program.

00729 Subtract one from pattern indicator. If all patterns have been checked, go to 00769. If not, modify pattern generator address at 00329, 00374 Transfer to 00329.

00769 Print "Z" as a pass indication. Transfer to 00294 to repeat entire program.

| HOUSEKEEPING | | | |
|---|---|---|---|
| 00244 | SET | B | 0001 |
| 00249 | LOD | 8 | 00878 |
| 00254 | UNL | 7 | 00845 |
| 00259 | UNL | 7 | 00853 |
| 00264 | SET | B | 0005 |
| 00269 | LOD | 8 | 00779 |
| 00274 | UNL | 7 | 00004 |
| 00279 | SEL | 2 | 0500 |
| 00284 | WR | R | 00815 |
| 00289 | HLT | J | 9999 |

Page 3 (E)

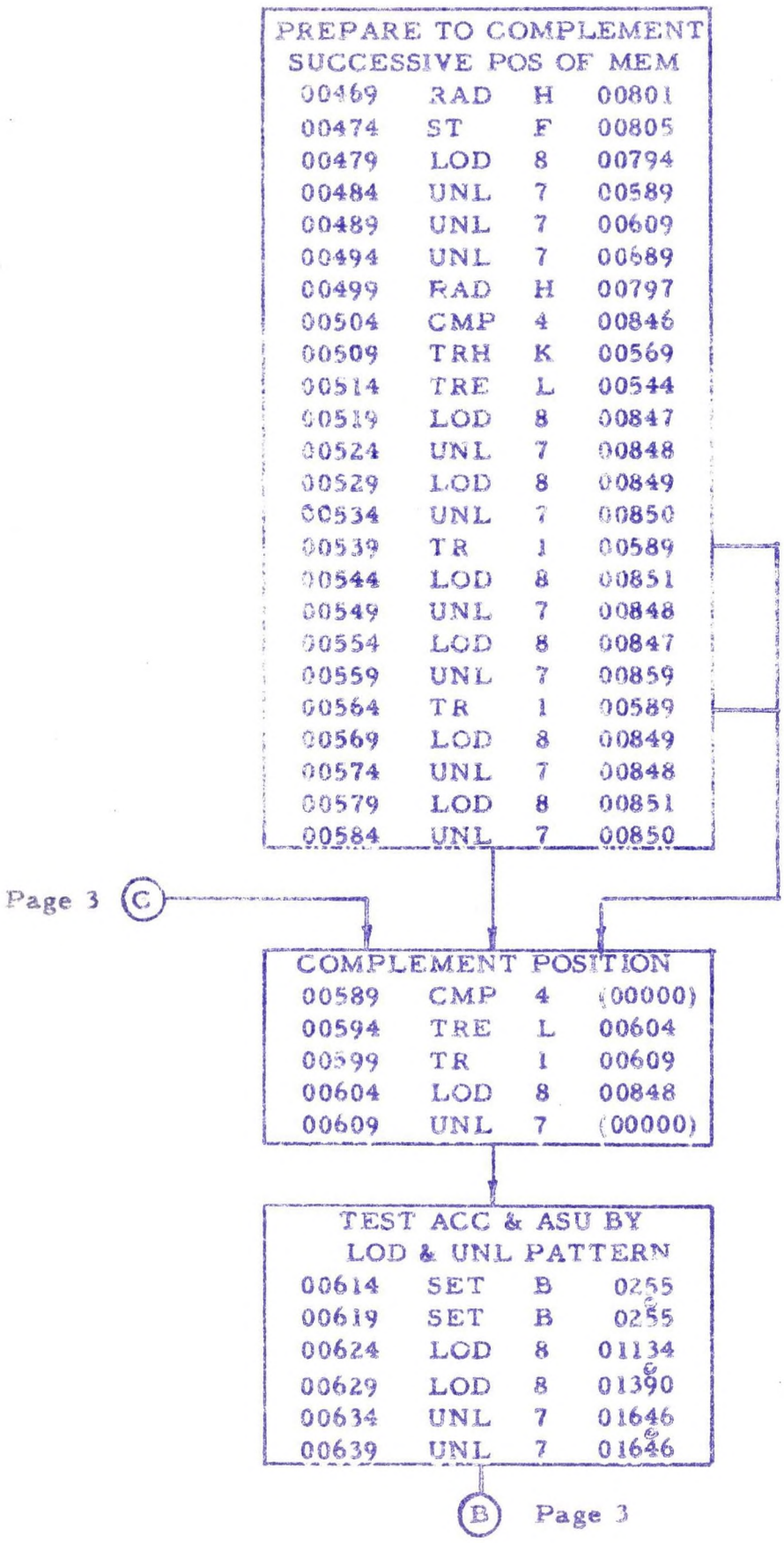
| PREP TO GEN FIRST PATTERN | | | |
|---|---|---|---|
| 00294 | RAD | J | 00796 |
| 00299 | ST | F | 00797 |
| 00304 | SET | B | 0004 |
| 00309 | LOD | 8 | 00784 |
| 00314 | UNL | 7 | 00329 |
| 00319 | LOD | 8 | 00789 |
| 00324 | UNL | 7 | 00374 |

Page 3 (D)

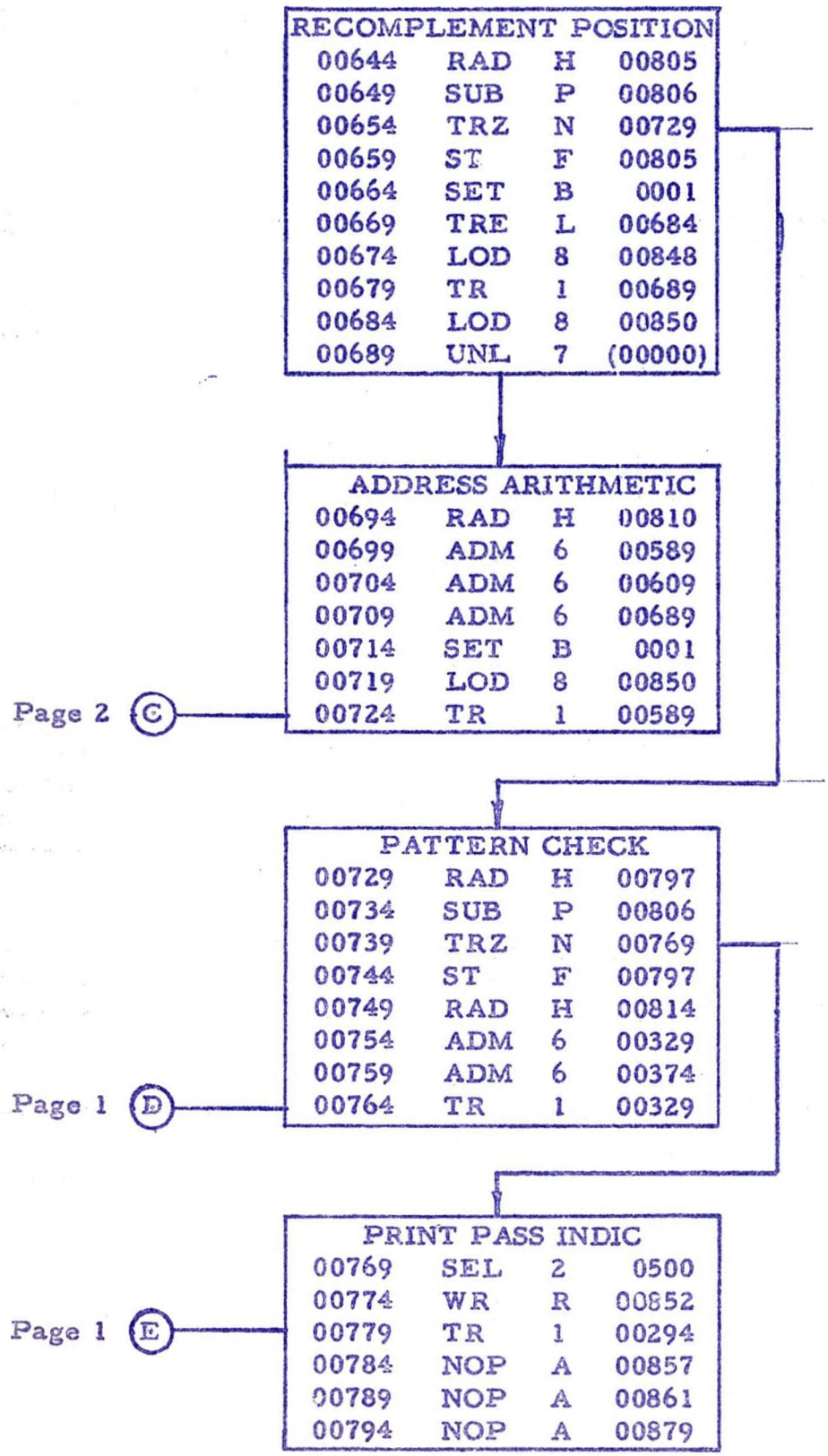
| GENERATE PATTERN | | | |
|---|---|---|---|
| 00329 | LOD | 8 | (00000) |
| 00334 | UNL | 7 | 00910 |
| 00339 | UNL | 7 | 00906 |
| 00344 | UNL | 7 | 00902 |
| 00349 | UNL | 7 | 00898 |
| 00354 | UNL | 7 | 00894 |
| 00359 | UNL | 7 | 00890 |
| 00364 | UNL | 7 | 00886 |
| 00369 | UNL | 7 | 00882 |
| 00374 | LOD | 8 | (00000) |
| 00379 | UNL | 7 | 00974 |
| 00384 | UNL | 7 | 00970 |
| 00389 | UNL | 7 | 00966 |
| 00394 | UNL | 7 | 00962 |
| 00399 | UNL | 7 | 00958 |
| 00404 | UNL | 7 | 00954 |
| 00409 | UNL | 7 | 00950 |
| 00414 | UNL | 7 | 00946 |
| 00419 | LGN | D | 0028 |
| 00424 | LOD | 8 | 00910 |
| 00429 | UNL | 7 | 00942 |
| 00434 | LOD | 8 | 00974 |
| 00439 | UNL | 7 | 01006 |
| 00444 | LGN | D | 0096 |
| 00449 | LOD | 8 | 01006 |
| 00454 | UNL | 7 | 01134 |
| 00459 | UNL | 7 | 01262 |
| 00464 | UNL | 7 | 01390 |

(A) Page 2

(A) Page 1


PREPARE TO COMPLEMENT SUCCESSIVE POS OF MEM
00469 RAD H 00801
00474 ST F 00805
00479 LOD 8 00794
00484 UNL 7 00589
00489 UNL 7 00609
00494 UNL 7 00689
00499 RAD H 00797
00504 CMP 4 00846
00509 TRH K 00569
00514 TRE L 00544
00519 LOD 8 00847
00524 UNL 7 00848
00529 LOD 8 00849
00534 UNL 7 00850
00539 TR 1 00589
00544 LOD 8 00851
00549 UNL 7 00848
00554 LOD 8 00847
00559 UNL 7 00859
00564 TR 1 00589
00569 LOD 8 00849
00574 UNL 7 00848
00579 LOD 8 00851
00584 UNL 7 00850
Page 3 (C)
COMPLEMENT POSITION
00589 CMP 4 (00000)
00594 TRE L 00604
00599 TR 1 00609
00604 LOD 8 00848
00609 UNL 7 (00000)
TEST ACC & ASU BY LOD & UNL PATTERN
00614 SET B 0255
00619 SET B 0255
00624 LOD 8 01134
00629 LOD 8 01390
00634 UNL 7 01646
00639 UNL 7 01646
(B) Page 3

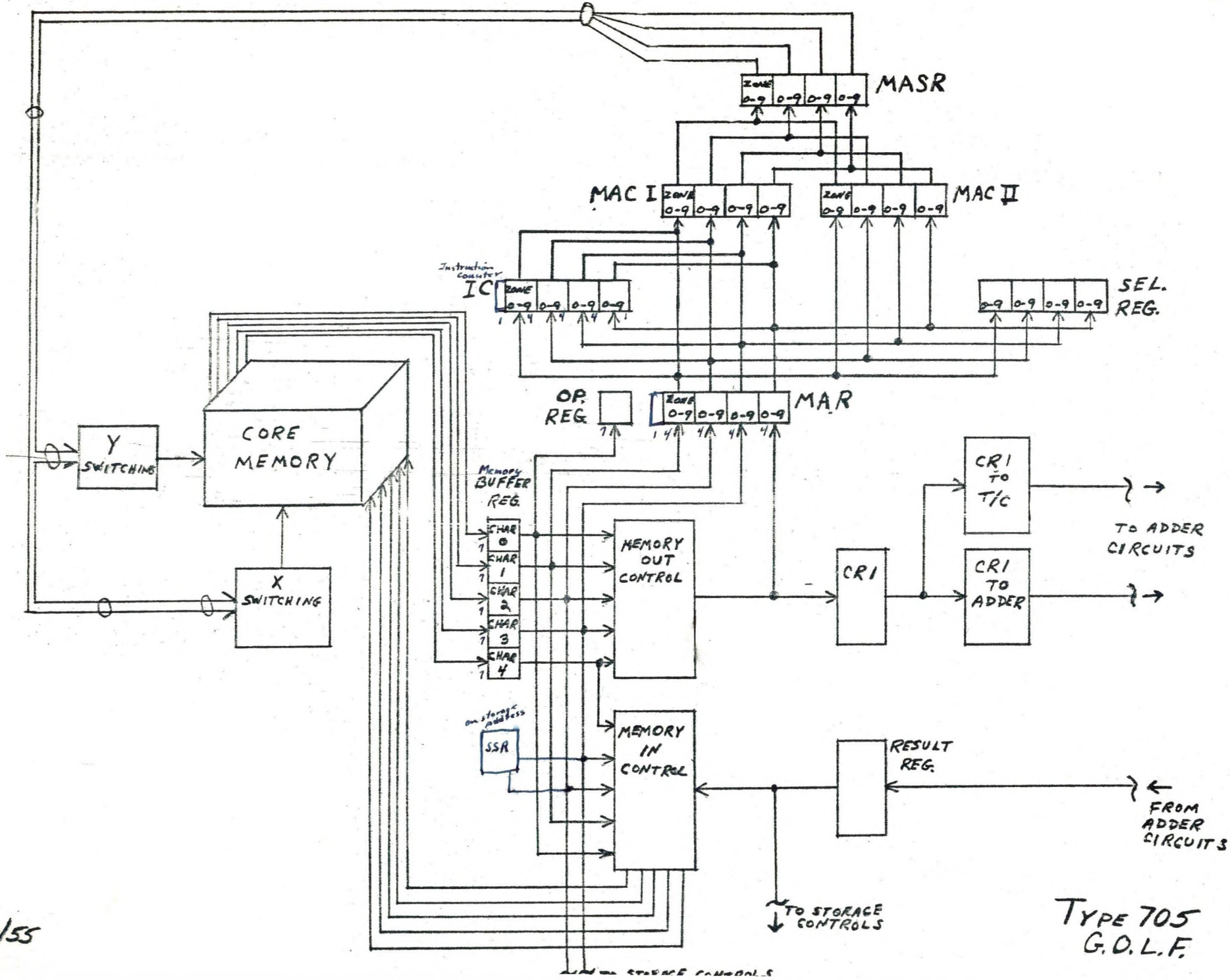
B Page 2


RECOMPLEMENT POSITION
00644 RAD H 00805
00649 SUB P 00806
00654 TRZ N 00729
00659 ST F 00805
00664 SET B 0001
00669 TRE L 00684
00674 LOD 8 00848
00679 TR 1 00689
00684 LOD 8 00850
00689 UNL 7 (00000)
ADDRESS ARITHMETIC
00694 RAD H 00810
00699 ADM 6 00589
00704 ADM 6 00609
00709 ADM 6 00689
00714 SET B 0001
00719 LOD 8 00850
Page 2 C
00724 TR 1 00589
PATTERN CHECK
00729 RAD H 00797
00734 SUB P 00806
00739 TRZ N 00769
00744 ST F 00797
00749 RAD H 00814
00754 ADM 6 00329
00759 ADM 6 00374
Page 1 D
00764 TR 1 00329
PRINT PASS INDIC
00769 SEL 2 0500
00774 WR R 00852
Page 1 E
00779 TR 1 00294
00784 NOP A 00857
00789 NOP A 00861
00794 NOP A 00879

```
00796                   & 3
00797
00801                   & 0511
00805
00806                   & 1
00810                   & 0001
00814                   & 0008
00845                   CS20A MAN SHR ACC TILL SPC IS 0
00846                   2
00847                   H
00848
00849                   1
00850
00851                   6
00853
00857                   1166
00861                   6611
00865                   66HH
00869                   HH66
00873                   HH11
00877                   11HH
00878                   GM
00910
00942
00974
01006
01134
01262
01390
01646
```

# MEMORY ADDRESS CONTROLS

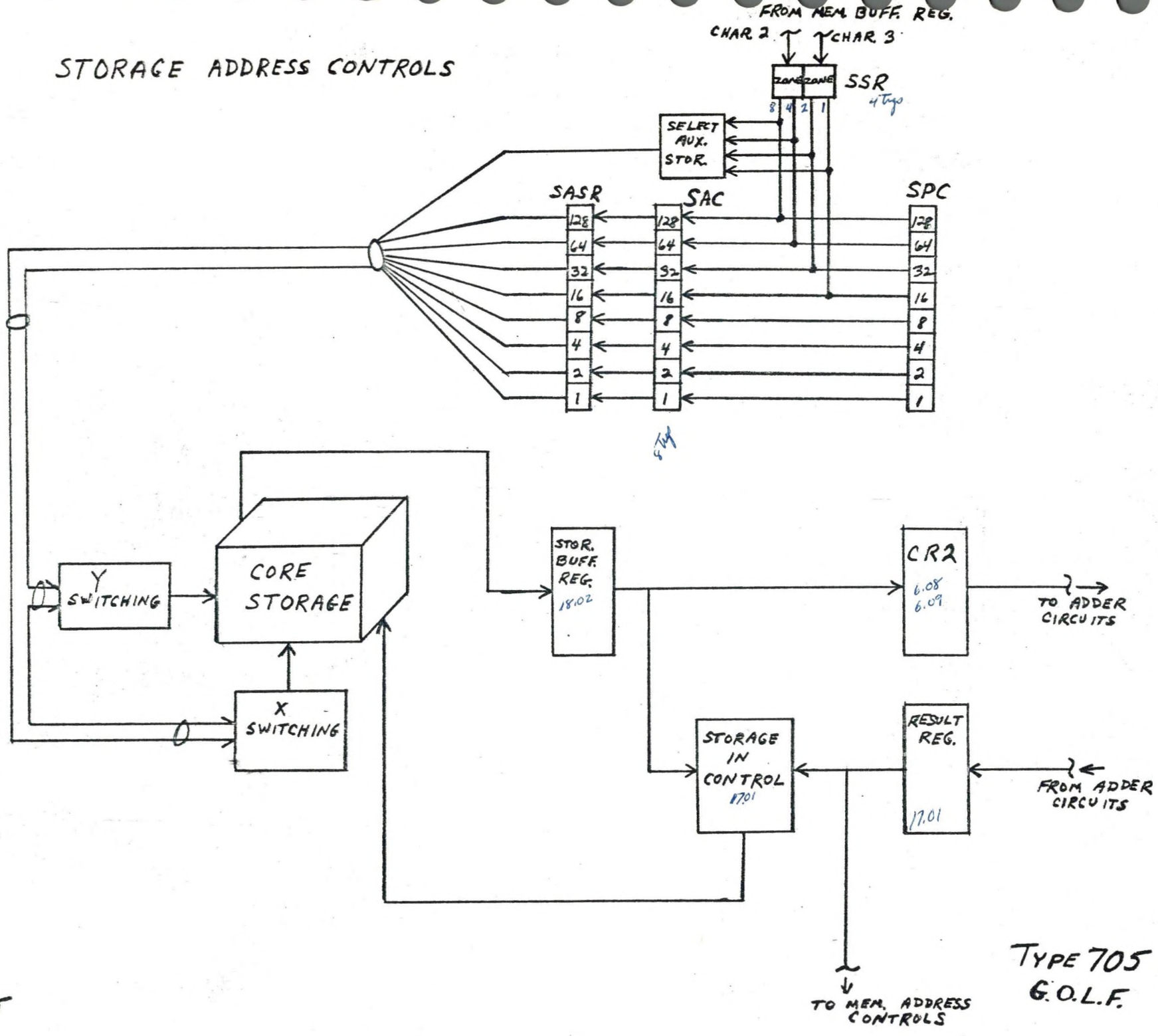





2/17/55

TYPE 705
G.O.L.F.

# STORAGE ADDRESS CONTROLS




2/18/55

TYPE 705
G.O.L.F.

2/27/56

Sheet 1 of 3

# 5LD001

# LOAD PROGRAM

Alteration Switches: No alteration switches used - set to dictates of program.

Check Switches: All auto stop.

Procedure: Load loading cards in card reader followed by program cards.

| | | |
|---|---|---|
| Select card reader | (2 00100) | from console |
| Read into 00000 | (Y 00000) | empties buffer - feeds 1 card. |
| Reset | | |
| Start | | |

Load program will read in its own second card and relocate itself in upper memory. It will pick up the first program card and type out the identification for verification. It will then proceed to pick up the length indicated in columns 14 and 15 of each card (in Acc. Stor.) and set ASU 12 left this amount to transmit that much of the remaining portion of the card to the starting memory address indicated in columns 10 through 13 of the card.

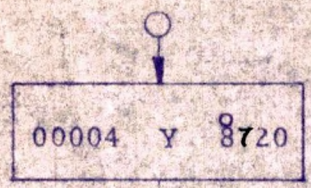
If the load program encounters a 00 punching in columns 14 and 15 of a card, it will transfer to the first instruction of that card (which will be located at memory address 18659).

If it is desired to load subsequent programs without reloading the load program (providing that the program run has not altered addresses 18590 through 18784 of the machine). This may be done by:

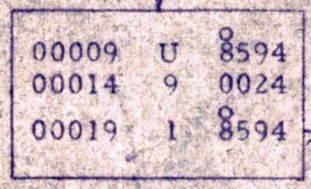
Manually transferring to 18594. (1 18594)
And depressing "Start".

The load program will load any address from 00000 to 18589. It can load, but will destroy itself, addresses 00000 through 18719 providing that the information destined for any address within the 18590 to 18719 range is contained in the very last card of the program deck.

It can also load addresses 18785 THROUGH 19999 without any effect on itself.

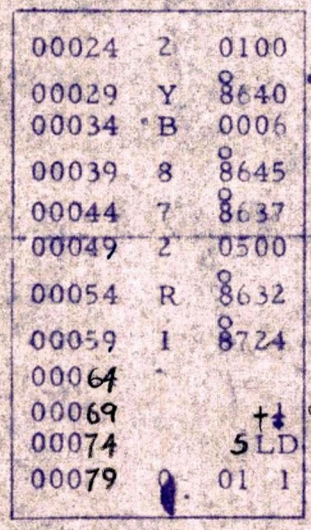



Reads second load card into upper Memory.




Transmits remaining portion of first card to upper Memory and transfers to these instructions. (A) sheet 3

Remaining Portion of first card




group mark & record mark

Sheet 2 (A) →

```
18594  2  0100
                8
18599  Y  8640
```

Reads in first program card.

```
18604  B  0006
                8
18609  8  8645
                8
18614  7  8637
18619  2  0500
                8
18624  R  8632
                8
18629  1  8724
18634
18639
```

Types out identification for verification.

```
18724  B  0002
                8
18729  8  8654
                8
18734  N  8659
                8
18739  7  8759
```

Picks up length, tests for zero, and unloads to ASU 12 Set L for control of TSMT - RCV operation.

```
18744  B  0004
                8
18749  8  8652
                8
18754  7  8764
```

Picks up starting memory address and unloads to RCV instruction.

```
18759  B  0065
18764  U
                8
18769  9  8655
```

Transmits information to proper position in memory.

```
18774  2  0100
                8
18779  Y  8640
                8
18784  1  8724
```

Reads in next program card and repeats operation.

Program Card Image

```
864-  865-  866-  867  868  869  870  871
01234567890123456789012345678901234567890123456789012345678901234567890123456789
IDENT | SN | ADR | L
```

# 705 ASSEMBLY PROCESS - Part I

## I INPUT-OUTPUT

A. Tape 0200 contains assembly program in four records: Part I, Part II, Part III and Part IV or V.

B. Entry records for program to be assembled are contained on tape 0201 or cards or both.

C. Tables $B_1$, $C_1$, $B_2$, $C_2$ will be written on tape 0203, for use in Part II, as up to four records.

D. The first 1000 Entry Records, partially assembled, will be written on tape 0202; the remaining entry records will be written on tape 0204.

## II ASSEMBLY PROCESS

A. Entry records will be read from tape or cards and processed one at a time. If both inputs are used, records will be merged by symbolic location. A class zero will delete an entry with matching symbolic locations.

B. Assignment of actual locations will be carried out for each entry by initializing a location counter with a class 6 entry, and stepping same by the length of each entry.

C. Actual operation codes for instructions will be looked up in a table of mnemonics.

D. Entries will be checked for:

1. Sequence of symbolic locations.
2. Duplication of symbolic locations.
3. Impossible operation codes.
4. Impossible mnemonic codes.
5. A program which exceeds memory.
6. A program which exceeds the capacity of the tables used in the assembly process.
7. A blank address in a class blank instruction.
8. An actual address greater than 19999.
9. Instructions which call for operation in an ASU but should use the accumulator.
10. An ASU address greater than 15.

E. The first 1000 partially assembled records will be written, one at a time, on tape 0202; and the rest written on tape 0204.

F. Tables for looking up symbolic addresses during Part II will be developed from actual locations assigned to all entries; and, since these tables may exceed the capacity of memory in Part I, they will be written on tape 0203 for use in Part II.

G. Summary of tables developed during Part I.

1. Six Table

   A table of class 6 entries, called the breakpoint or 6 table, and consisting of a symbolic location and an actual address for each class 6, is developed to enable the assignment of actual addresses to symbolic class 6 entries.

2. Table A
   A table containing a reference address, of four positions, for each major symbolic group, called Table A, will be developed and stored in upper memory to provide a starting point for the search for the actual address of a given entry in Part II. Four positions of this table are preassigned to each major group to facilitate look-up. The reference address will locate the first entry for a particular major group in Table D, which in reality is Table B re-positioned, during Part II.
3. Table B
   A table developed from symbolic locations and actual locations from which actual addresses may be calculated or found during Part II. Entries to this table contain 8 positions; two positions for minor symbolic; two positions for a count, and four positions for an address.

   An entry will be made for each break in consecutive sequence; i.e., a break in major, minor, each insertion, and change in class sequence. (For class sequence considerations, b & 1 are the same, and 2 & 5 are the same). Table B entries will also contain a zone coding to distinguish the class of the sequence which the entry represents.

   The minor portion of the entry will be the minor of the first in a sequence.

   The counts portion is used to keep a count of the number in a given sequence to check the accuracy of the table.

   The Symbolic insertion will be a single entry with the insertion in the tens position of the counts.

   The Address portion will contain two types of addresses. For b, 1, 3 and insertions the address will be the actual location corresponding to first symbolic location of a sequence. For 2 & 5, the address will be a reference address giving the location of the first actual location in that sequence in Table E. Table E is the same as Table C but repositioned for Part II.

   The B Table starts after the main body of the program and is developed in ascending order.

4. Table C
   A table of actual locations of all class 2 and class 5 entries. Each location has four positions, and entries are made in

Pem

# 705 ASSEMBLY PROCESS - Part II

## I THE PROGRAM

Upon completion of Part I, the Part II instructions will be read from Tape 0200 and Tables $B_1$, $C_1$, $B_2$, $C_2$ will be read into the remaining memory from Tape 0203. The tables will be entered in the order $B_1$, $B_2$, $C_2$, $C_1$ and in Part II are consolidated to form Table D and Table E. Table A is retained at the upper end of memory.

Entry records are then read, one at a time, from Tape 0202 and Tape 0204, symbolic addresses will be found from the tables, and the completed records written on tape 0203.

## II TABLE LOOK-UP

All entries are complete after Part I except class blank and class three entries. It is the purpose of Part II to look-up the actual addresses for these entries corresponding to their symbolic addresses in the tables of actual locations developed in Part I.

### A. Table A

The first step in looking up a class blank or three address is to obtain the starting address, in Table D, for the major block containing the symbolic address to be looked up. This may be obtained from Table A.

Since each major was preassigned four positions in Table A, the address required may be located by multiplying the major by four and adding to the result, the starting address of TableA.

To limit the search in D to the Major block in question, the starting address of the next block must also be obtained from Table A. This will be the next address in Table A which is not blank.

### B. Table D

A search of Table D is then made starting at the address obtained from Table A and comparing the minor and insertion of the symbolic being looked up with the minor and insertion portion of successive Table D entries until the proper entry is either found or passed. The only entries which will match the symbolic are insertions, since all other entries have a zone code in the insertion position. In either event, the Table D entry which was just passed or found directly in the search will be moved to a work area. The address for an insertion can be taken directly to the output record, but all others will require computation to obtain the actual address. The nature of this computation will depend upon the class from which the Table D entry was derived as indicated by the zone coding on the entry.

For a class blank or one, the difference of the minors will be multiplied by five and the result added to the address of the Table D entry to yield the actual address.

Ben

For a class three, the difference of the minors will be multiplied by four and the result added to the address of the Table D entry to yield the actual address.

For a class two or five, the difference of the minors will be multiplied by four and the result subtracted from the address of the Table D entry to yield the location in Table E of the actual address.

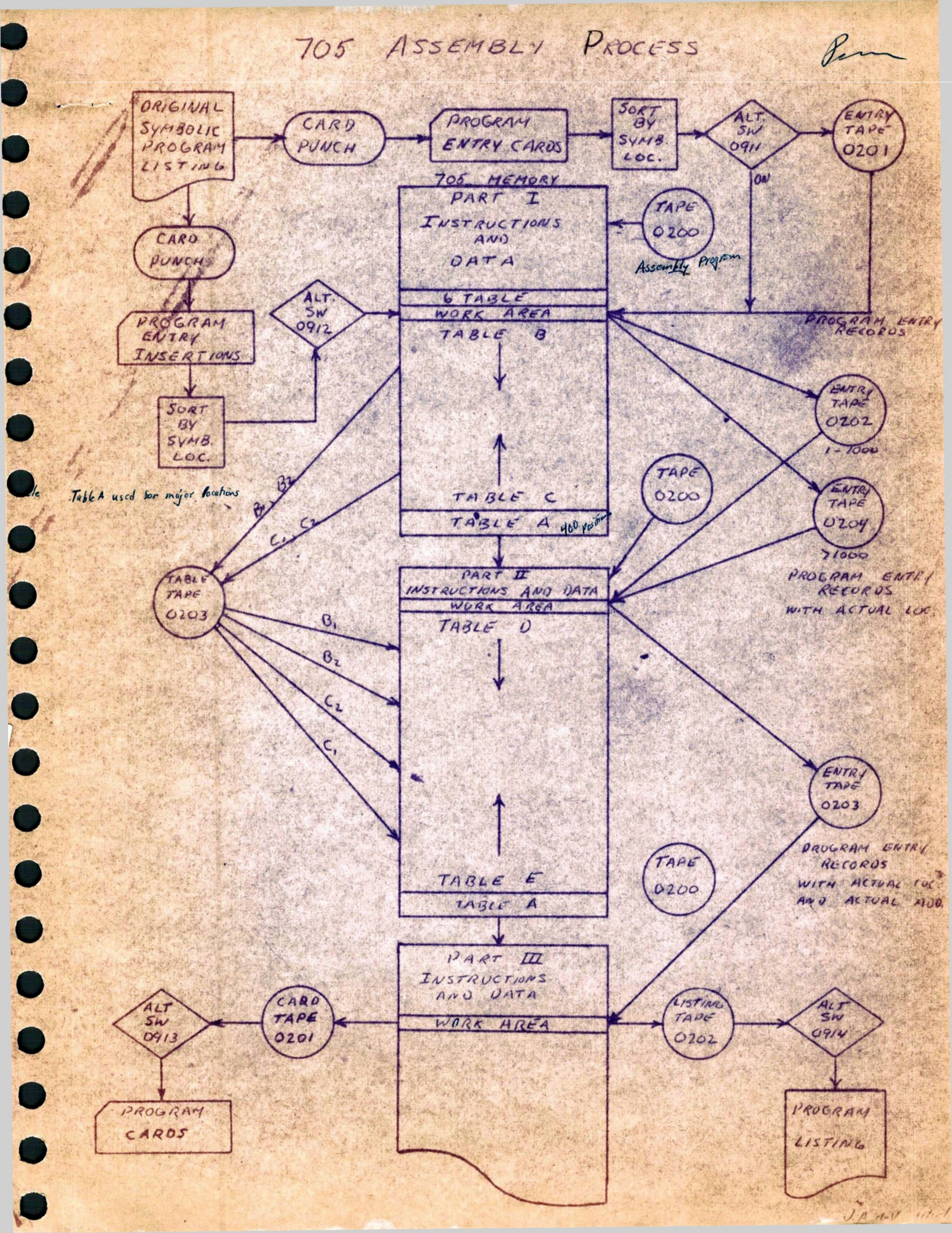
C. Table E
Subtraction of the multiple minor difference from the reference address is used because the Table E entries were made in reverse or descending order.

D. Increments
After the actual address is obtained the symbolic increment is added to or subtracted from the actual address. The result, which is the address for which the search was made, is then combined with the rest of the record in the output area and the then completed record written on Tape 0203.

descending order starting at the upper end of memory.

When the B and C tables converge, a record of each is written on tape 0203 and a second set of tables will then be developed. More than two sets of tables will exceed the capacity of the assembly program.

# 705 ASSEMBLY PROCESS



IBM Trade Mark ELECTRONIC DATA-PROCESSING MACHINE — TYPE 705 **WORK SHEET** PROBLEM _____

| 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 |
|---|---|---|---|---|---|---|---|
| 09 | 10 | 11 | 12 | 13 | 14 | 15 | |

| INSTR. LOCATION | INSTRUCTION OPER. | INSTRUCTION ADDRESS | CODE | ACCUMULATOR 00 | SIGN | AUXILIARY STORAGE 01-15 | SIGN | EXPLANATION |
|---|---|---|---|---|---|---|---|---|
| 0004 | Sel 2 | 0200 | | | | | | Select Tape 1 |
| 0009 | Rd Y | 7001 | | | | | | Read into memory 7001 → |
| 0014 | RAD H | 7005 | 00 | @07321 | + | | | Rnd add into Acc Storage ← 7005 |
| 0019 | Add G | 7012 | 00 | @328471 | + | | | Add into Acc Storage ← 7012 |
| 0024 | ST F | 7019 | 00 | @328471 | + | | | Store in memory 7000 ← 7019 |
| 0029 | Sel 2 | 0201 | | | | | | Select Tape 2 |
| 0034 | WR R | 7001 | 00 | | | | | Wr from storage to Tape 7001 → 7040 |
| 0039 | TR 1 | 0004 | | | | | | Transfer to original instruction |
| | | | | | | | | |
| 0004 | Sel 2 | 0200 | | | | | | Select Tape 1 |
| 9 | Rd X | $\bar{3}$988 | | | | | | Read into memory 13988 → 14008 |
| 14 | RAD H | $\bar{3}$998 | 01 | | | @031525 | + | Rnd Add from memory into aux 1 ← 13998 |
| 19 | Add G | $\bar{4}$003 | 01 | | | @033975 | + | Add from memory into aux 1 ← 14003 |
| 24 | Sub P | $\bar{4}$008 | 01 | | | @028925 | + | Sub from memory into aux 1 ← 14008 |
| 29 | ST F | $\bar{4}$014 | 01 | | | @028925 | + | Store in memory ← 14014 |
| 34 | Sel 2 | 0201 | | | | | | Select Tape 2 |
| 39 | Wr R | $\bar{3}$988 | 00 | | | | | Write from memory to Tape |
| 44 | TR 1 | 0004 | | | | | | Transfer to original instruction |
| | | | | | | | | |
| 0004 | Sel 2 | 0203 | | | | | | Select Tape 203 |
| 0009 | Rd Y | $\bar{0}$708 | | | | | | Read into Memory 10708 → 10712 |
| 14 | RAD H | $\bar{0}$708 | 00 | 140 | + | | | Rnd Add into Acc Storage |
| 19 | MPY Y | $\bar{0}$712 | 00 | @0062300 | + | | | MPY 140 x 0445 |
| 24 | ST F | $\bar{0}$719 | 00 | @0062300 | + | | | Store in memory |
| 29 | Sel 2 | 0200 | | | | | | |
| 34 | Wr R | $\bar{0}$706 | 00 | | | | | |
| 39 | Tr 1 | 0004 | | | | | | |

IBM Trade Mark ELECTRONIC DATA-PROCESSING MACHINE — TYPE 705 **WORK SHEET** PROBLEM _____

| 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 |
|---|---|---|---|---|---|---|---|
| 09 | 10 | 11 | 12 | 13 | 14 | 15 | |

| INSTR. LOCATION | INSTRUCTION OPER. | INSTRUCTION ADDRESS | CODE | ACCUMULATOR 00 | SIGN | AUXILIARY STORAGE 01-15 | SIGN | EXPLANATION |
|---|---|---|---|---|---|---|---|---|
| 0004 | Sel 2 | 0200 | | | | | | Sel. Tape |
| 0009 | Rd Y | 1290 | | | | | | Read into Memory. |
| 0014 | RAD H | 1297 | 00 | | | | | Read into acc storage |
| 0019 | Set B | 0004 | 00 | ⊗0900 | + | | | Set left |
| 24 | Long D | 0004 | 00 | ⊗0900000 | + | | | Length |
| 29 | Div | 1294 | 00 | ⊗8181 | + | | | Divide |
| 34 | RND | 0001 | 00 | ⊗818 | + | | | Adjust to nearest 1/10 1% |
| 39 | ST | 1300 | 00 | 818 | | | | Store quotient in memory. |
| 44 | Sel | 0201 | | | | | | Sel Tape. |
| 49 | WR | 1285 | 00 | | | | | Record to Tape |
| 54 | TR | 0004 | | | | | | Transfer to 04 |
| | Sel | 0200 | | | | | | Sel Tape |
| | Rl | 6028 | | | | | | |
| | Rcd | 6026 | 00 | ⊗176085 | + | | | |
| | Div | 6028 | 00 | ⊗8385 | + | | | |
| | RND | 0001 | 00 | ⊗839 | + | | | |
| | ST | 6021 | | 839 | | | | |
| | Sel | 0201 | | | | | | |
| | WR | 6021 | | | | | | |
| | TR | 0004 | | | | | | |

Form 22-6625-1 Printed in U. S. A.

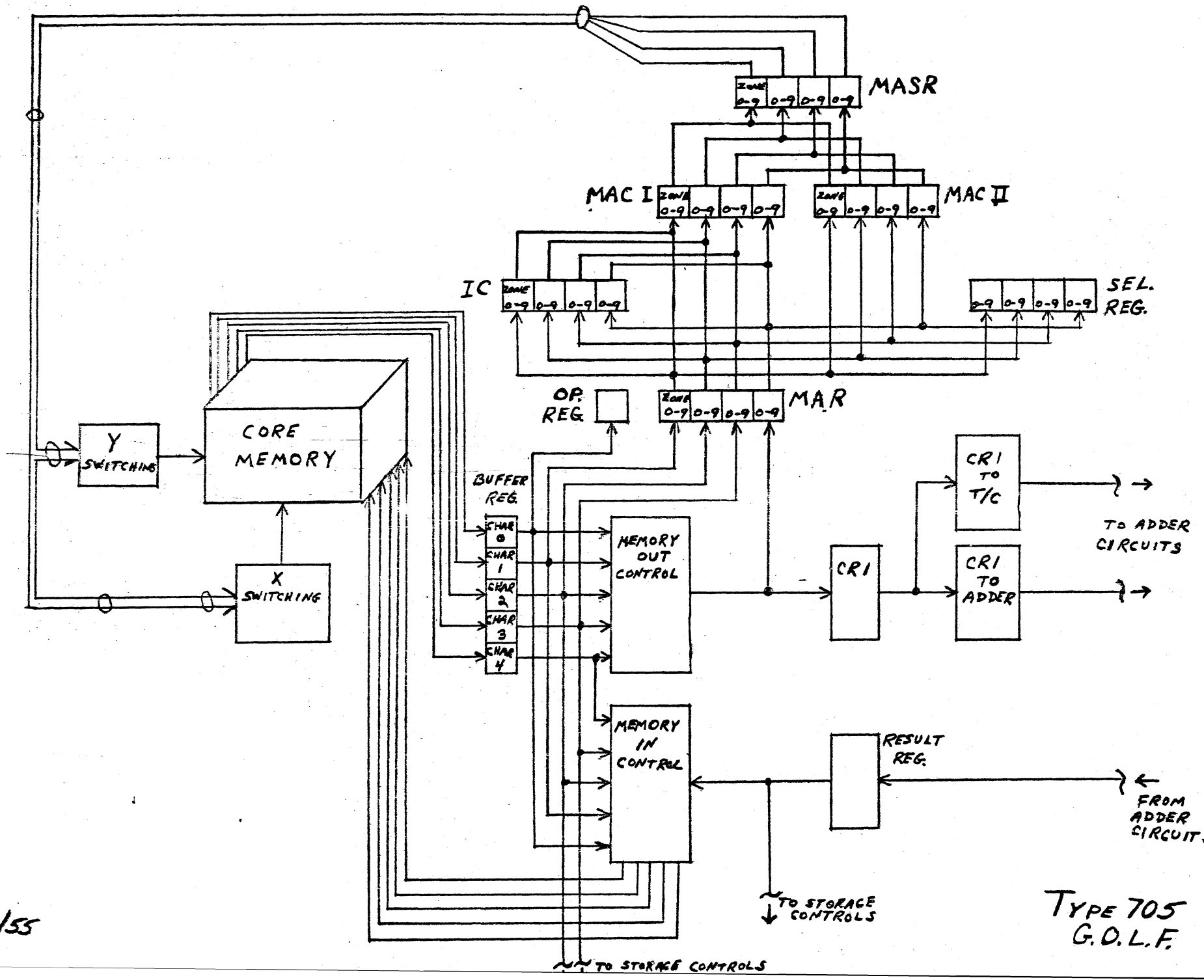
IBM Trade Mark ELECTRONIC DATA-PROCESSING MACHINE — TYPE 705 **WORK SHEET** PROBLEM _____

| 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 |
|---|---|---|---|---|---|---|---|
| 09 | 10 | 11 | 12 | 13 | 14 | 15 | |

| INSTR. LOCATION | INSTRUCTION OPER. | INSTRUCTION ADDRESS | STOR. CODE | ACCUMULATOR 00 | SIGN | AUXILIARY STORAGE 01-15 | SIGN | EXPLANATION |
|---|---|---|---|---|---|---|---|---|
| 0004 | Sel | 0100 | | | | | | Select reader |
| 0009 | Rd | 1201 | | | | | | Read into memory 1201 |
| 0014 | Sel | 0200 | | | | | | Sel tape unit |
| 0019 | Write | 1201 | 00 | | | | | Write from memory 1201 to group mark |
| 0024 | Trs | 0004 | | | | | | Transfer to 1st instruction |
| 0004 | Sel | 0200 | | | | | | Sel. tape unit |
| 0009 | Rd | 7001 | | | | | | Read into memory from tape |
| 0014 | Sel | 0300 | | | | | | Sel card punch. |
| 0019 | Wr | 7001 | 00 | | | | | Write from mem into punch. |
| 0024 | Trs | 0004 | | | | | | Trsf to 1st instruction |
| 0004 | Sel | 0100 | | | | | | Sel. reader |
| 0009 | Rd | 7001 | | | | | | Read into mem 7001 |
| 0014 | Sel | 0200 | | | | | | Sel. tape unit |
| 0019 | Wr | 7001 | 00 | | | | | Wr. from mem to tape |
| 0024 | Sel | 0400 | | | | | | Sel Ptr. |
| 0029 | Wr | 7019 | 00 | | | | | Wr from mem to Ptr |
| 0034 | Trs | 0004 | | | | | | |

Form 22-6625-0 Printed in U.S.A.

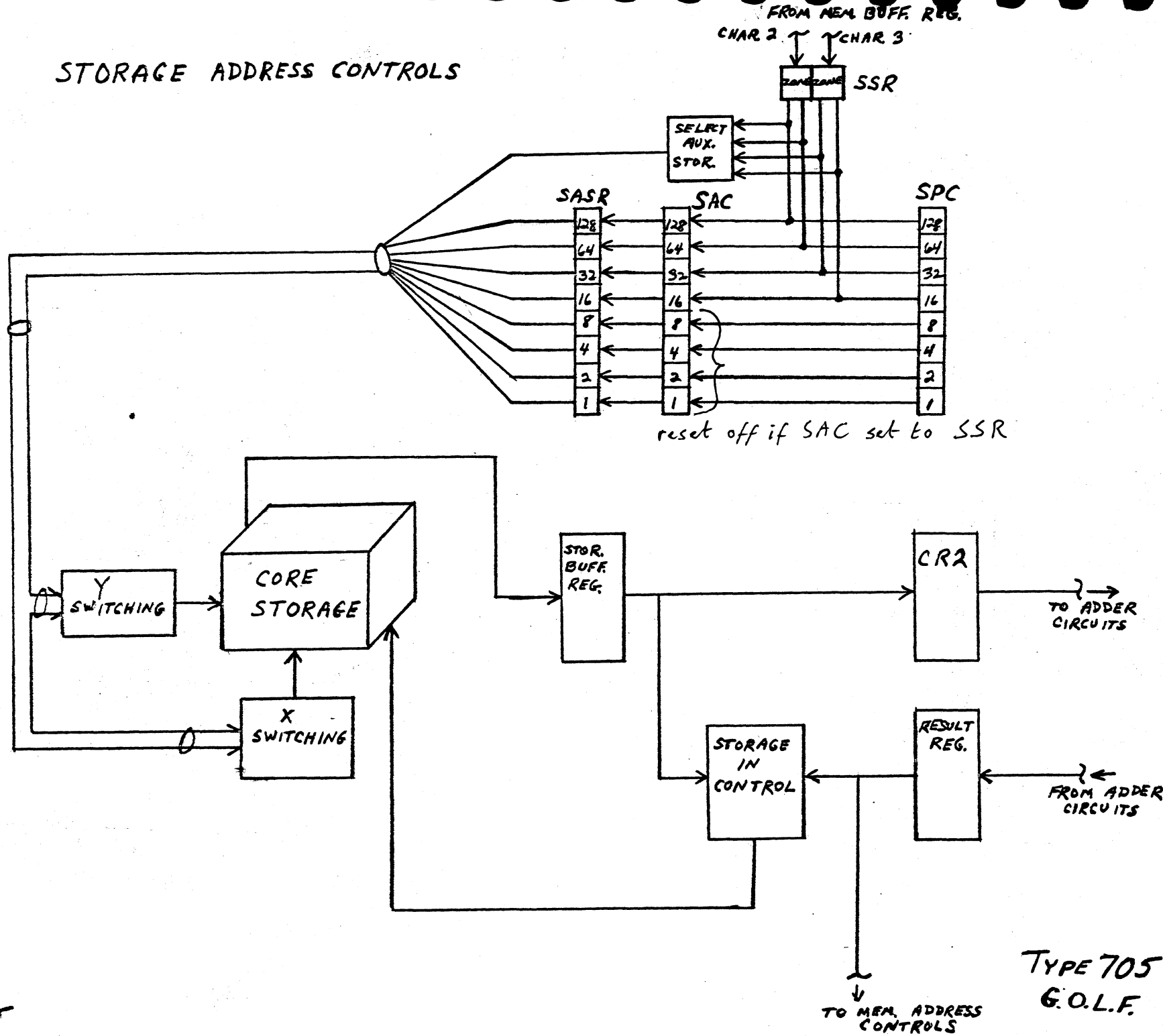
# MEMORY ADDRESS CONTROLS




2/17/55

TYPE 705
G.O.L.F.

# STORAGE ADDRESS CONTROLS



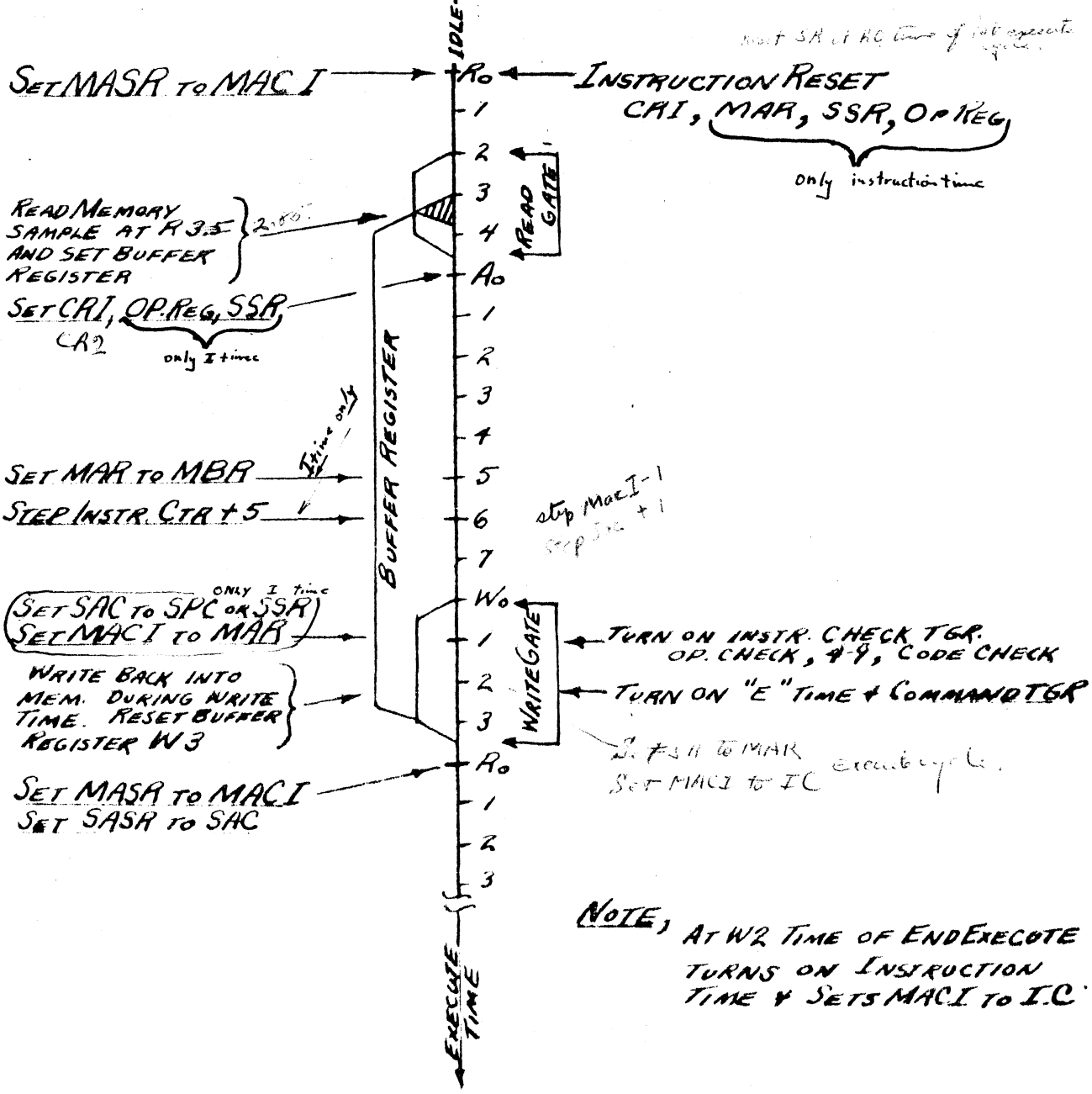

2/18/55

TYPE 705
G.O.L.F.

# INSTRUCTION TIME

ROUTE: { READ MEMORY
MACI to MAR
SAC to SPC

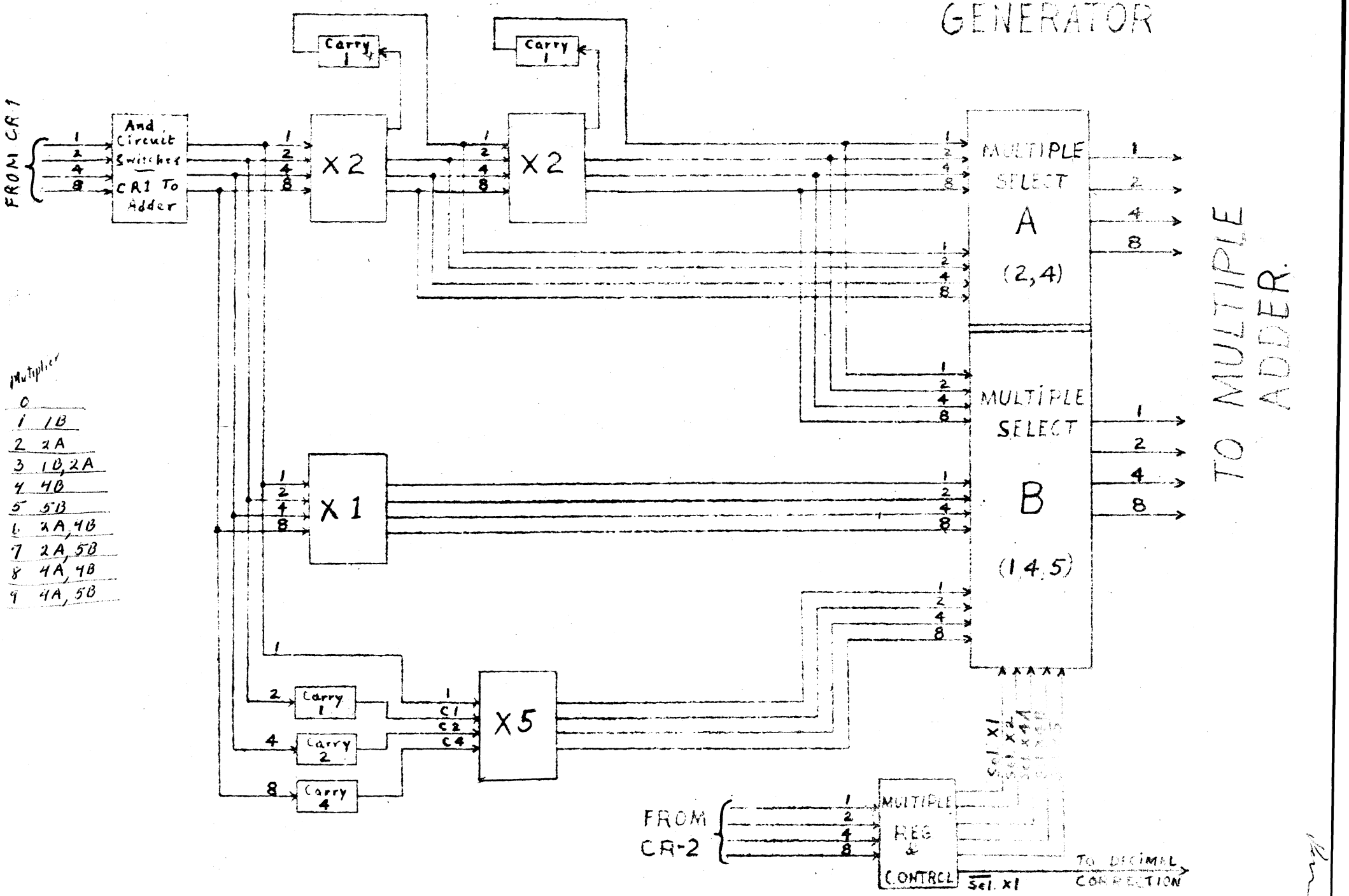
RESET KEY — SET INSTRUCTION COUNTER TO 0004
START KEY — SET MACI TO INSTRUCTION COUNTER
START INSTRUCTION USE CYCLE




NOTE, AT W2 TIME OF END EXECOTE TURNS ON INSTRUCTION TIME + SETS MACI TO I.C.

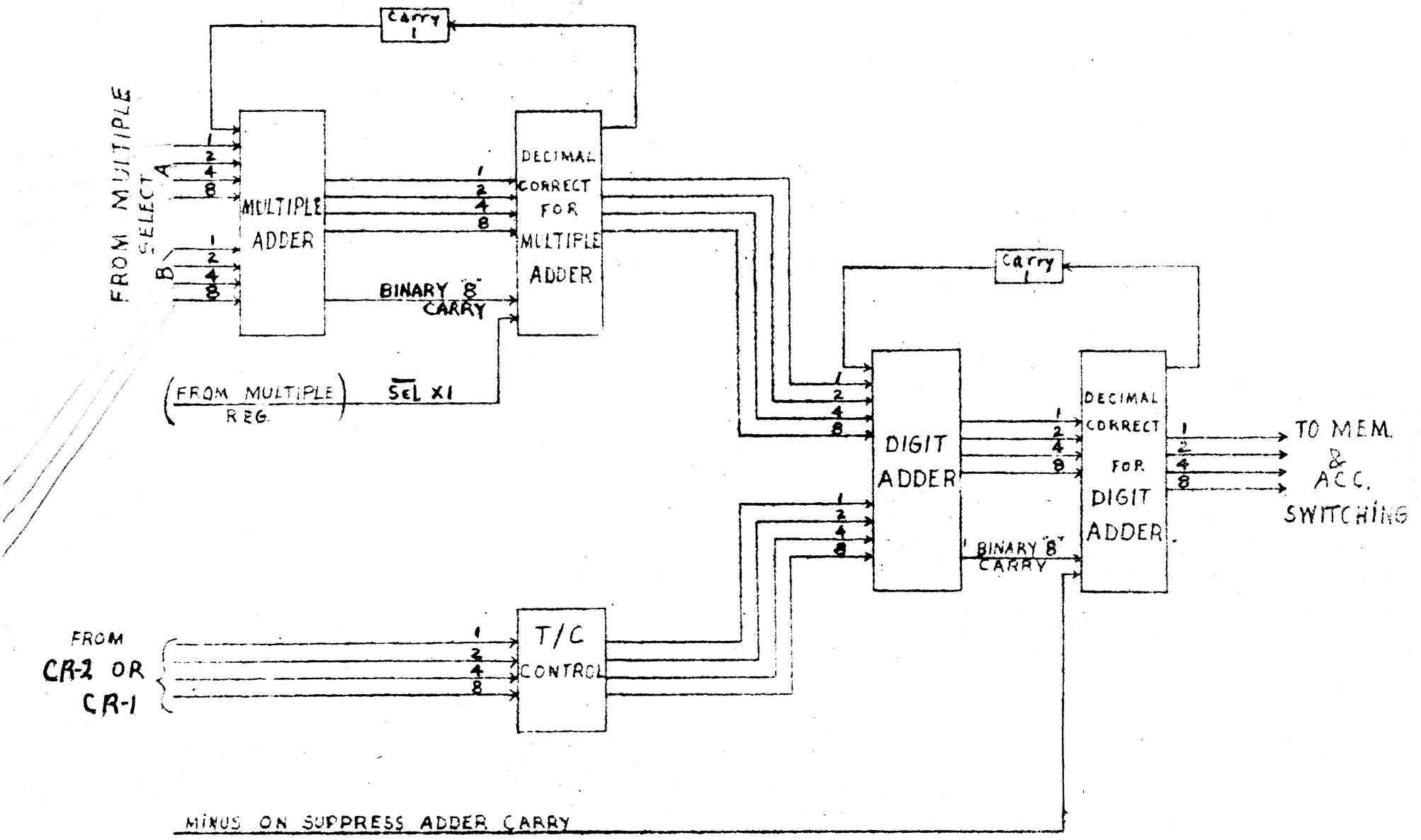
3/24/55

# MULTIPLE GENERATOR




| Multiplier | |
|---|---|
| 0 | |
| 1 | 1B |
| 2 | 2A |
| 3 | 1B, 2A |
| 4 | 4B |
| 5 | 5B |
| 6 | 2A, 4B |
| 7 | 2A, 5B |
| 8 | 4A, 4B |
| 9 | 4A, 5B |

J.D.C.


Carry 1
FROM MULTIPLE SELECT
A
B
1
2
4
8
MULTIPLE ADDER
BINARY "8" CARRY
DECIMAL CORRECT FOR MULTIPLE ADDER
(FROM MULTIPLE REG.)
SEL X1
Carry 1
DIGIT ADDER
DECIMAL CORRECT FOR DIGIT ADDER
BINARY "8" CARRY
TO MEM. & ACC. SWITCHING
FROM CR-2 OR CR-1
T/C CONTROL
MINUS ON SUPPRESS ADDER CARRY

J.D.C.

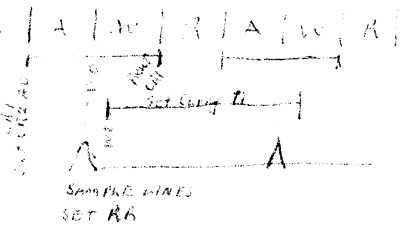
87
243
261
348
435
1241

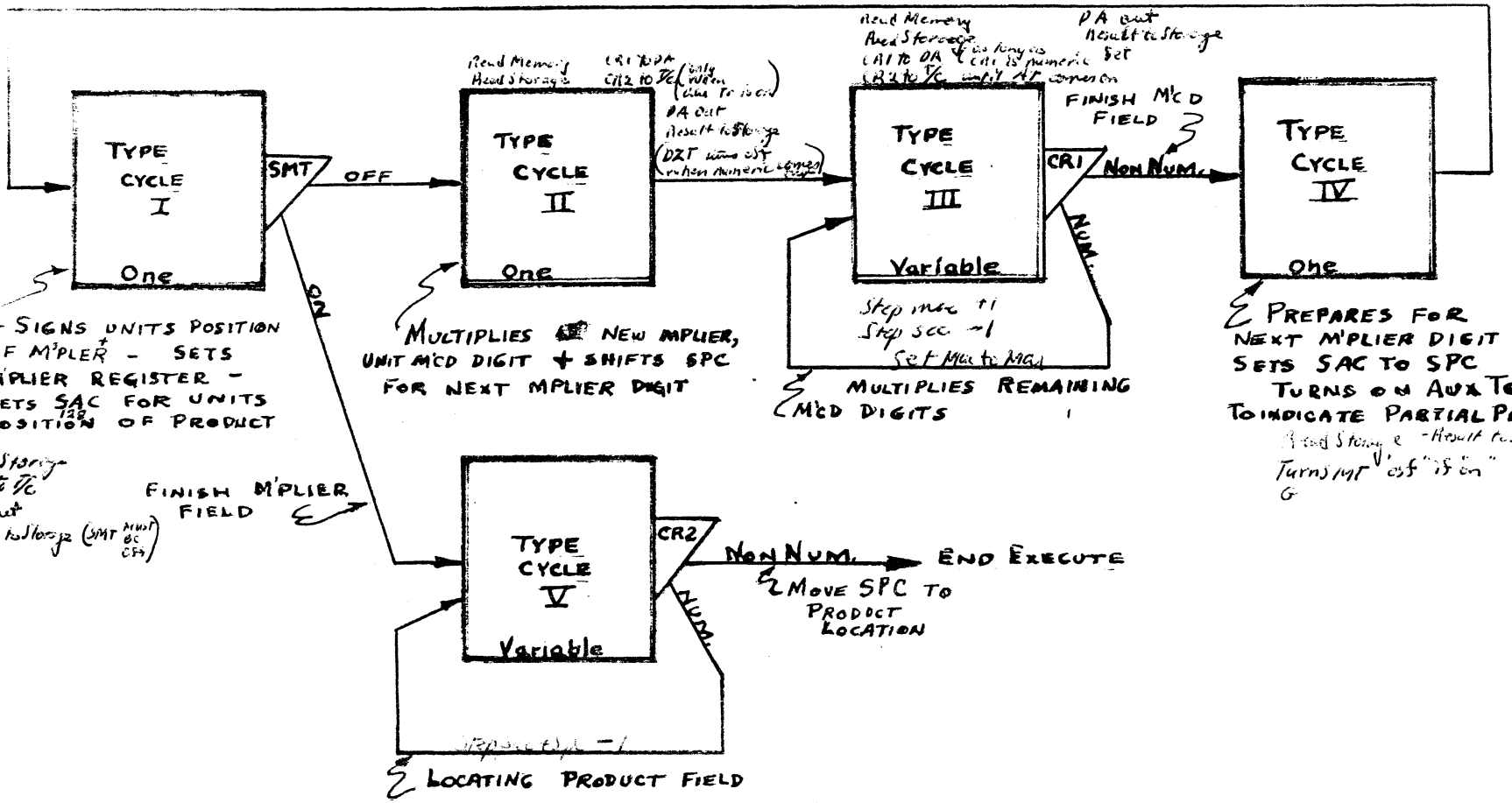
| SELECT | | x 1 | 1ST DOUBLER x 2 | | | | 2ND DOUBLER x 4 | | | | x 8 | | | | MULTIPLE ADDER | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | 1 | 7 | 7 | 0 | 1 | 4 | 4 | 0 | 0 | 8 | 7 | 0 | 3 | 5 | 4 | 7 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 2 | 1 | 8 | 8 | 1 | 1 | 7 | 7 | 0 | 1 | 4 | 8 | 3 | 4 | 3 | 7 | 8 | 1 | 1 | 6 | 6 | 0 | 0 | 0 | 6 | 6 |
| 2 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 3 | 0 | 4 | 0 | 4 | 1 | 0 | 1 | 0 | 2 | 2 | 0 | 0 | 0 | 2 | 2 |
| | 4 | 7 | 7 | 0 | 1 | 4 | 4 | 0 | 0 | 8 | 7 | 0 | 3 | 8 | 0 | 8 | 0 | 0 | 8 | 8 | 6 | 0 | 1 | 4 | 4 |
| | 4 | 8 | 8 | 1 | 1 | 7 | 7 | 0 | 1 | 4 | 8 | 3 | 4 | 3 | 0 | 4 | 0 | 0 | 4 | 4 | 2 | 1 | 0 | 7 | 7 |
| | 4 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 3 | 0 | 4 | 0 | 4 | 0 | 3 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 3 | 3 |
| | 5 | 7 | 7 | 0 | 1 | 4 | 4 | 0 | 0 | 8 | 7 | 0 | 3 | 5 | 0 | 5 | 0 | 0 | 5 | 5 | 7 | 0 | 1 | 2 | 2 |
| | 5 | 8 | 8 | 1 | 1 | 7 | 7 | 0 | 1 | 4 | 4 | 3 | 2 | 3 | 0 | 3 | 0 | 0 | 3 | 3 | 3 | 1 | 0 | 7 | 7 |
| | 5 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 3 | 0 | 2 | 0 | 2 | 0 | 2 | 0 | 0 | 2 | 2 | 2 | 0 | 0 | 4 | 4 |

| | MULTIPLIER SELECT | | CR 1 | 1ST DOUBLER X2 | | | | 2ND DOUBLER X4 | | | | QUINTUPLER X5 | | | | MULTIPLE ADDER | | | | | DIGIT ADDER | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | X 1 OUT | INPUT FROM CR2 | CARRY IN | CARRY OUT | RESULT | INPUT FROM 1ST DOUBLER | CARRY IN | CARRY OUT | RESULT | INPUT FROM CR-1 | CARRY IN | CARRY OUT | RESULT | INPUT A (X2, X4) | INPUT B (X1, X5) | CARRY IN | CARRY OUT | RESULT | INPUT FROM MULT. ADDER | INPUT FROM CR-2 | CARRY IN | CARRY OUT | RESULT | RESULT TO ACC |
| 8 | 4 | 4 | | 6 | 0 | 1 | 2 | 2 | 0 | 0 | 4 | 6 | 0 | 3 | 0 | 4 | 4 | 0 | 0 | 8 | 8 | 0 | 0 | 0 | 8 | 8 |
| | 4 | 4 | | 5 | 1 | 1 | 1 | 1 | 0 | 0 | 2 | 5 | 3 | 2 | 8 | 2 | 2 | 0 | 0 | 4 | 4 | 0 | 0 | 0 | 4 | 4 |
| | 4 | 4 | | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 2 | 0 | 2 | 0 | 2 | 2 | 2 | 0 | 0 | 4 | 4 | 0 | 0 | 0 | 4 | 4 |
| 7 | 2 | 5 | | 6 | 0 | 1 | 2 | 2 | 0 | 0 | 4 | 6 | 0 | 3 | 0 | 2 | 0 | 0 | 0 | 2 | 2 | 4 | 0 | 0 | 6 | 6 |
| | 2 | 5 | | 5 | 1 | 1 | 1 | 1 | 0 | 0 | 2 | 5 | 3 | 2 | 8 | 1 | 8 | 0 | 0 | 9 | 9 | 4 | 0 | 1 | 3 | 3 |
| | 2 | 5 | | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 2 | 0 | 2 | 0 | 2 | 1 | 2 | 0 | 0 | 3 | 3 | 0 | 1 | 0 | 4 | 4 |
| 6 | 2 | 4 | | 3 | 0 | 0 | 6 | 6 | 0 | 1 | 2 | 3 | 0 | 1 | 5 | 6 | 2 | 0 | 0 | 8 | 8 | 0 | 0 | 0 | 8 | 8 |
| | 2 | 4 | | 2 | 0 | 0 | 4 | 4 | 1 | 0 | 9 | 2 | 1 | 1 | 1 | 4 | 9 | 0 | 1 | 3 | 3 | 0 | 0 | 0 | 3 | 3 |
| | 2 | 4 | | 1 | 0 | 0 | 2 | 2 | 0 | 0 | 4 | 1 | 1 | 0 | 6 | 2 | 4 | 1 | 0 | 7 | 7 | 0 | 0 | 0 | 7 | 7 |
| 5 | 0 | 5 | | 3 | 0 | 0 | 6 | 6 | 0 | 1 | 2 | 3 | 0 | 1 | 5 | 0 | 5 | 0 | 0 | 5 | 5 | 3 | 0 | 0 | 8 | 8 |
| | 0 | 5 | | 2 | 0 | 0 | 4 | 4 | 1 | 0 | 9 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 7 | 0 | 0 | 8 | 8 |
| | 0 | 5 | | 1 | 0 | 0 | 2 | 2 | 0 | 0 | 4 | 1 | 1 | 0 | 6 | 0 | 6 | 0 | 0 | 6 | 6 | 0 | 0 | 0 | 6 | 6 |

When on SMT - steps next CR2 to Vc
VET - steps next CR1 to DA

# MULTIPLY TYPE CYCLES




HDE
4-12-55

# IMPORTANT TIMINGS

## Ro

Set MASR to MAC I
Set MASR to MAC II (Transmit)
Set SASR to SAC
Turn on MAC I = 0 Tgr.

## R3

Step MAC I (+1)
Step MAC I (+5)
Step MAC II (+1)
Step MAC II (+5)
Step SAC (+1)
} Transmit

## A2

Turn on MET
Turn on SMT — First Execute Cycle
Turn on DZT
Set Add/Sub Tgr to Subtraction
Set Mem. Sign (-)
Set Mem. Sign (+)
Turn on Equal Tgr.
Turn off High Trg.
} ECCI Compare

## A6

Step MAC I (+1)
Step MAC I (-1)
Step SPC (+1)
Step SPC (-1)
Step SAC (+1)
Step SAC (-1)

## W1

Set SAC to SPC
Set MAC I to MAR
Step SPC (+128)
Step SAC (+128)
Turn on Mach. Chk. Tgr.

## W2

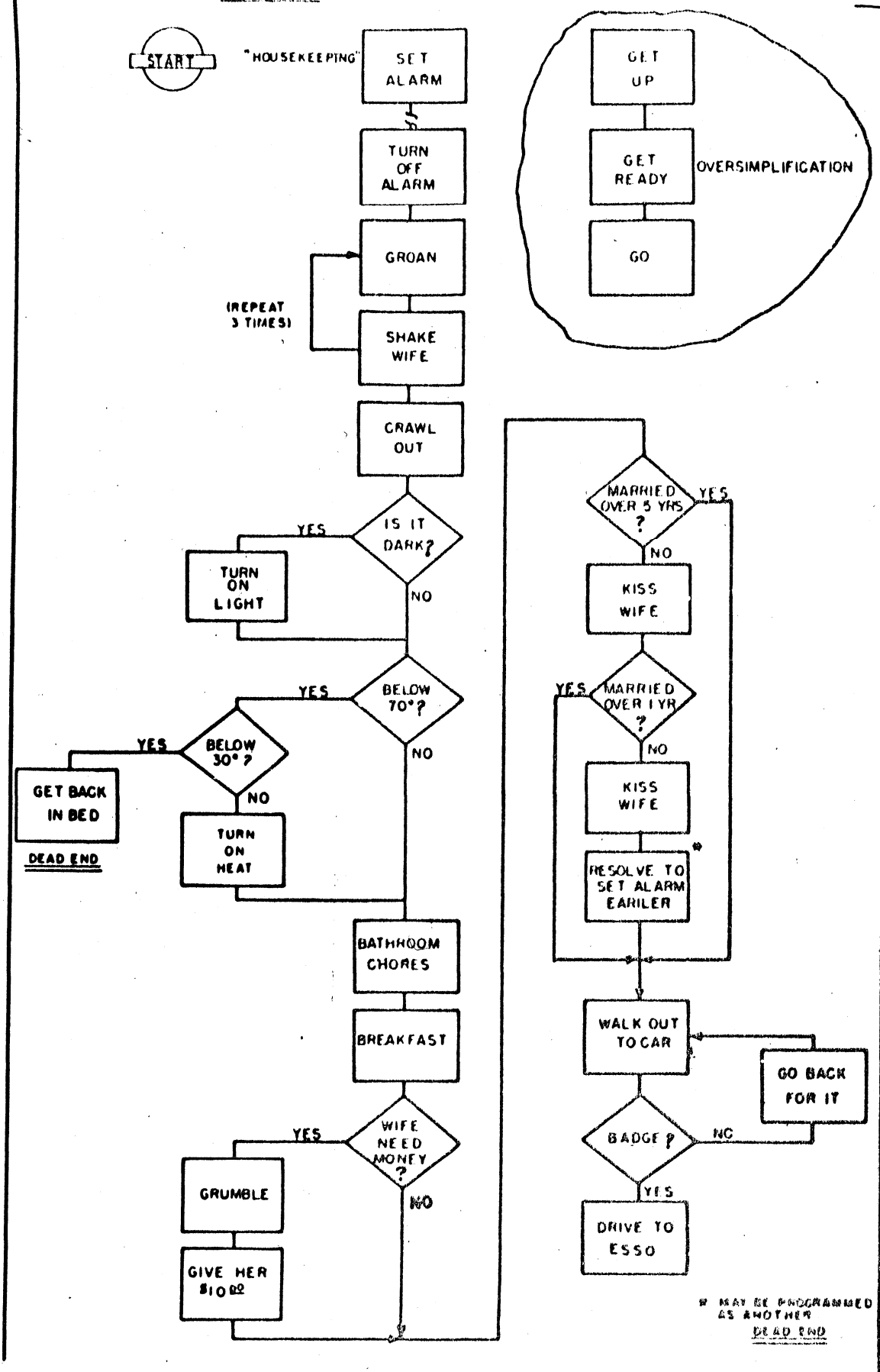
Turn on A. Tgr.
Turn off A. Tgr.
Turn on Rmdr. Tgr.
Turn off Rmdr. Tgr.
Turn on Sign Chk. Tgr.
Turn on Overflow Tgr.
Turn on High Tgr.
Turn off High Tgr.
Turn off Equal Tgr.
Turn off SMT
Turn off DZT
Set Mpy. Reg. to CR2
Set Storage Sign (+)
Set Storage Sign (-)
Set Storage Sign to Memory Sign
Set Storage Sign Opp. to Mem. Sign
Change Storage Sign
Go to Any Type Cycle
End Execute

READ MEMORY TO MBR — R3
READ STORAGE TO SBR — R3
SET CR1 OR CR2 = $A_0$ (RAW)
SET CR1 OR CR2 = $W_0$ (RW) TRANS. I
SET CR2 = $R_0$ I/O READ REQUEST
SET RESULT REG = $W_0$
{ RESET .. .. $W_3$
{ WRITE BACK IN STORAGE $W_3$

| | | |
|---|---|---|
| STORAGE SIGN | ON | $W_2$ |
| | OFF | $W_2$ |
| AUX. STOR. SIGN | ON | $W_2$ |
| | OFF | $W_2$ |
| MEM. SIGN | ON | $A_2$ |
| | INST. RESET | $R_0$ |
| EQ. TGR. | ON | $A_2$ |
| | OFF | $W_2$ |
| HI. TGR. | ON | $W_2$ |
| | Ecc1, COMPARE OFF | $A_2$ |
| | OTHER OFF | $W_2$ |
| WRITE CALL TGR. | ON | $W_2$ |
| | OFF | $W_2$ |
| | MACH. STOP OFF | |
| | INST. RESET | $R_0$ |
| NO WRITE RESPONSE TGR | ON | $W_2$ |
| | WRITE RESPONSE OFF | |
| NO READ RESPONSE | ON | $W_2$ |
| | READ RESPONSE OFF | |
| INST. CHK. TGR. | ON | $W_1$ |
| | OFF | $W_2$ |
| MACH. CHK. TGR | ON | $W_1$ |
| | OFF | $W_2$ |
| R/W CHK. TGR | CR2 ODD ON | $W_2$ |
| | R/W ERROR ON | $A_2$ |
| | OFF | $W_2$ |
| DR/PU CHK. TGR | ON | $A_2$ |
| | OFF | $W_2$ |
| OVERFLOW CHK. TGR. | ON | $W_2$ |
| | OFF | $W_2$ |
| SIGN CHK. TGR | ON | $W_2$ |
| | OFF | $W_2$ |

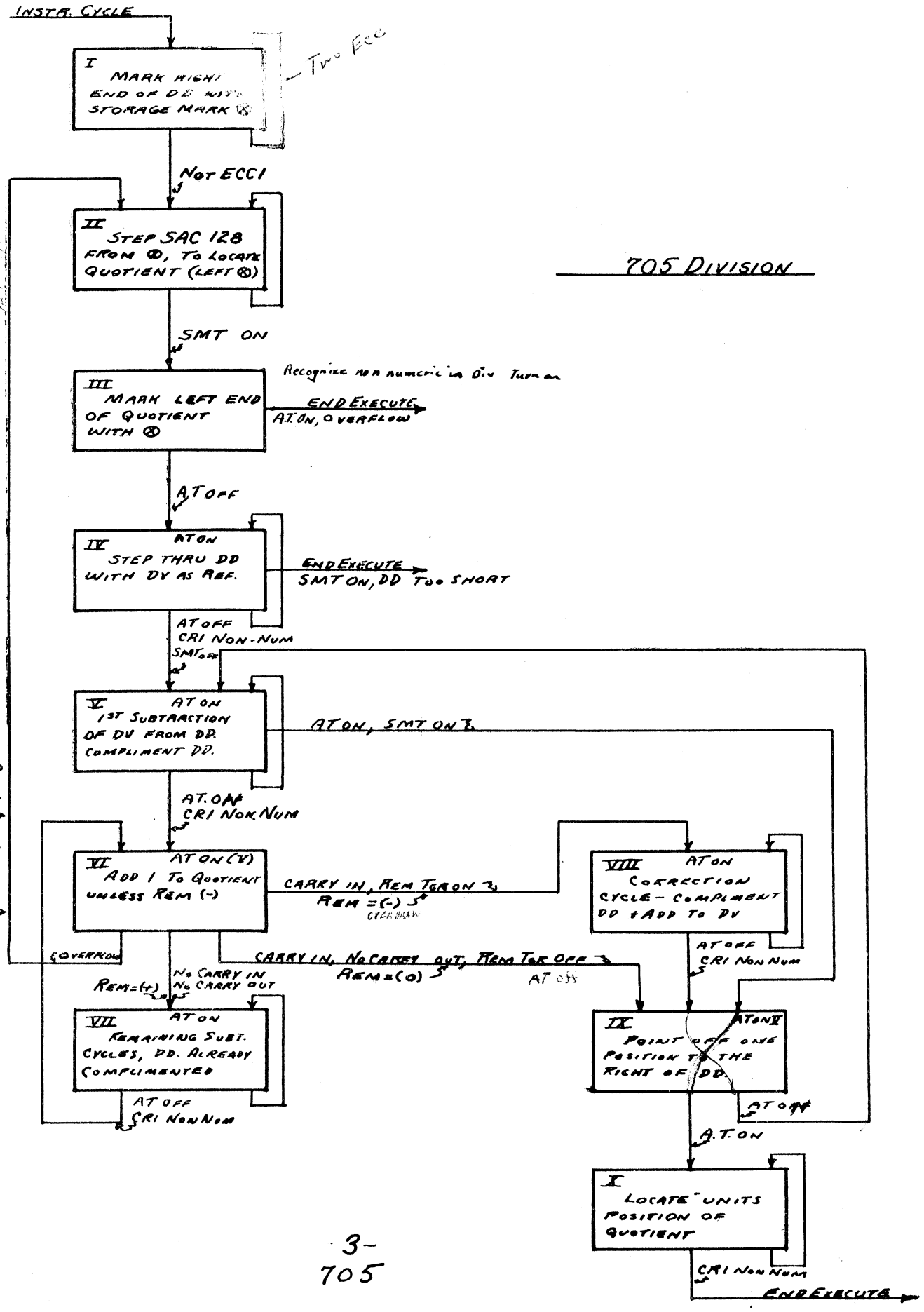
| | | |
|---|---|---|
| STOR. D2T | ON | $A_2$ |
| | OFF | $W_2$ |
| AUX. STOR. D2T | ON | $A_2$ |
| | OFF | $W_2$ |
| REMAINDER TGR | ON | $W_2$ |
| | OFF | $W_2$ |
| | INST. RESET | $R_0$ |
| AUX. TGR I/O TYPE CYCLE 2 | ON | $A_2$ |
| OTHER | ON | $W_2$ |
| | OFF | $W_2$ |
| | INST. RESET | $R_0$ |
| MAC = 0 | ON | $R_0$ |
| | INST. RESET | $R_0$ |
| S. M. T. | ON | $A_2$ |
| | OFF | $W_2$ |
| | INST. RESET | $R_0$ |
| M. E. T. | ON | $A_2$ |
| | INST. RESET | $R_0$ |
| SIMULT. R/W TGR | ON | $W_2$ |
| | OFF | $W_2$ |
| | INST. RESET | $R_0$ |
| READ CALL TGR | ON | $W_2$ |
| | MACH. STOP OFF | |
| | READ DISC. OFF | |
| | INST. RESET | $R_0$ |
| RESULT TO MEM TGR | R-A-W ON | $A_6$ |
| | R-W ON | $R_3$ |
| | OFF | $R_0$ |
| RESULT TO STOR. TGR | R-A-W ON | $A_6$ |
| | R-W ON | $R_3$ |
| | OFF | $R_0$ |
| Set Carry Tgrs. | ON | W2 |
| | OFF | W2 |

| Time | Routing or Stepping and Setting | Conditions | Remarks |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |




**HOW MECHANICAL BRAIN WORKS.** This diagram shows how programmers have to instruct a machine to work. The step-by-step procedure beginning at top left actually is patterned after the process of setting up a function on one of the electronic calculators. The instructor in a recent training course, found the diagram helpful in explaining the procedure to class members. They enjoyed the humor of the diagram as well as its instructive benefit.

$W_2$ Turn on AT  Turn off AT




705 DIVISION



HEB
4/25/55

Molloy

# DIVISION
# AUXILIARY TRIGGER

Type Cycle 1 -- Not Used

Type Cycle 2

1. Auxiliary Trigger OFF when coming from 1, normal operation.

2. Auxiliary Trigger ON when coming from 6; Error-quotient greater than 9.

Type Cycle 3

1. Auxiliary Trigger OFF when entering is normal for division; go to 4. turn on go to 4.

2. Auxiliary Trigger ON when entering 3 indicates a quotient greater than 9 -- recognized in 6. end special 6

3. Auxiliary Trigger turned ON before leaving TC3.

Type Cycle 4

1. Auxiliary Trigger is ON for first character cycle in 4 -- To ignore non-numeric in units position of memory.

2. Auxiliary Trigger is OFF for all other character cycles in 4.

3. Auxiliary Trigger is ON when leaving 4.

Type Cycle 5

1. Auxiliary Trigger is ON for first character cycle in 5 -- To ignore non-numeric in units position in memory.

2. Auxiliary Trigger is OFF for all other cycles in 5.

3. Auxiliary Trigger is ON when leaving 5.

Type Cycle 6

1. Auxiliary Trigger is ON when coming from 5 to indicate no partial quotient.

2. Auxiliary Trigger is OFF when coming from 7 to indicate partial ~~product~~. quotient

3. Auxiliary Trigger is OFF when leaving 6 if the divisor went an even number of times -- go to 9. (No R mdr).

4. Auxiliary Trigger is ON when leaving 6 if:

   a. Previous cycle had been overdrawn -- Go to 8 and correct.

   b. Quotient position exceeded 9 -- Go to 2 -- error.

   c. Attempt another reduction cycle -- Go to 7.

Type Cycle 7

1. Auxiliary Trigger is ON for first character cycle of 7 to ignore non-numeric in units position of memory.

2. Auxiliary Trigger is OFF for all other character cycles of 7.

Type Cycle 8

1. Auxiliary Trigger is ON for first character cycle at 8 to ignore non-numeric in units position of memory.

2. Auxiliary Trigger is OFF for all other character cycles of 8.

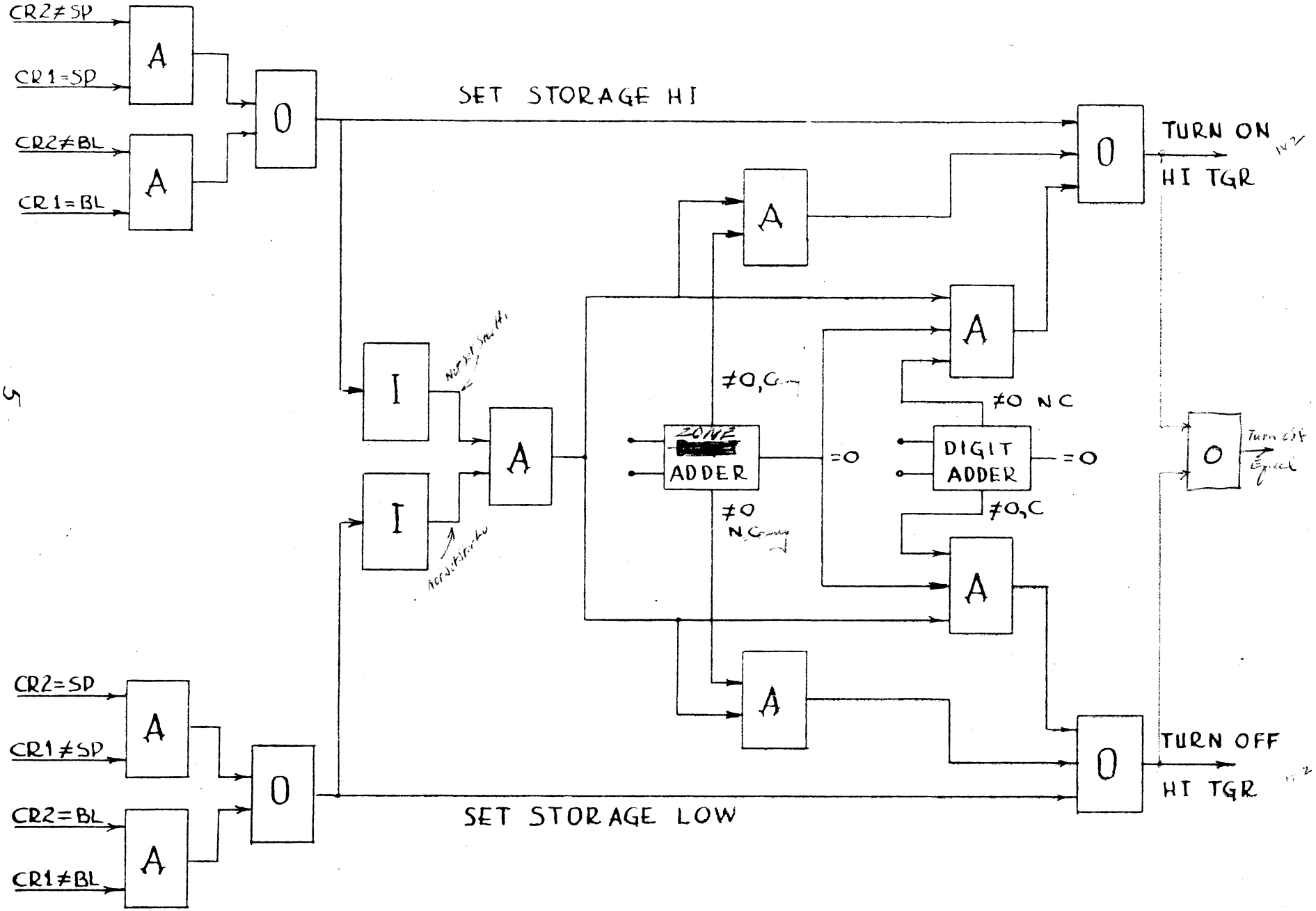
Type Cycle 9

1. Auxiliary Trigger is OFF when entering 9 -- turn it on and go to 5 and continue division.

2. Auxiliary Trigger is ON when entering 9 -- the division is complete go to 10.

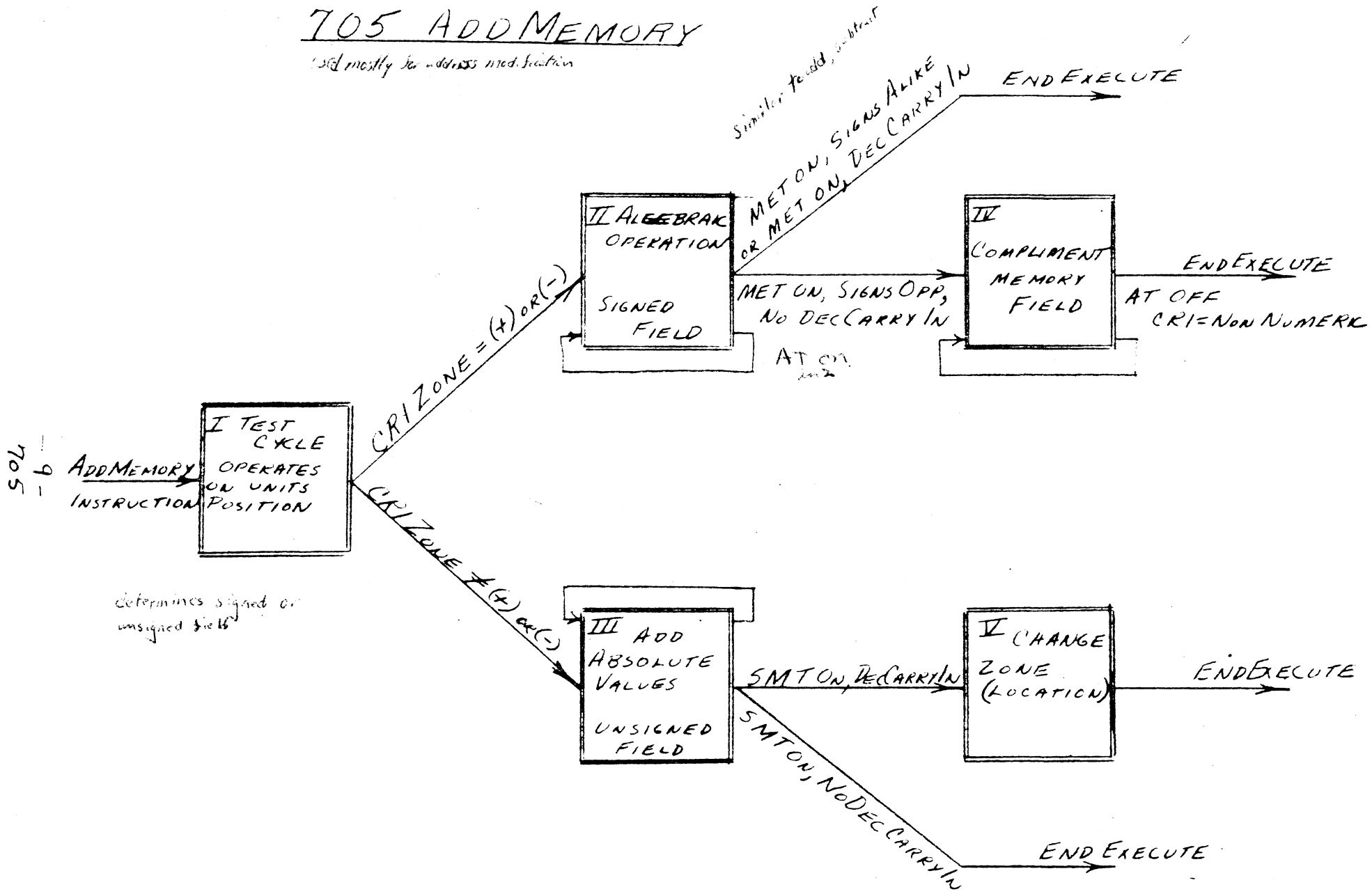
Type Cycle 10 -- Not Used.

# COMPARE


CR2≠SP
CR1=SP
CR2≠BL
CR1=BL
A
A
O
SET STORAGE HI
I
I
A
≠O,C
ZONE
ADDER
=O
≠O
NC
A
≠O NC
DIGIT
ADDER
=O
≠O,C
A
A
A
O
TURN ON
HI TGR
O
Turn off
Signal
O
TURN OFF
HI TGR
CR2=SP
CR1≠SP
CR2=BL
CR1≠BL
A
A
O
SET STORAGE LOW

# 705 ADD MEMORY

Used mostly for address mod. function




26