Prel. Draft

To: H.G. [illegible]

from J. Griff[illegible]

Gordon Adams

In the field of computer design and organization, progress toward higher speed and capabity has generally been accompanied by increasing logical complexity. As an alternative, it would be useful to consider the class of organizations which does not embody the high complexity of current computers. Since these machine configurations lack complex circut logic each must embody an increase in other parameters in order to achieve practical speeds. Profitable investigation of one of these organizations can occur when technology affords the possibility of a sharp increase in some needed parameter.

The object of this paper is to examine the results of the application of this philosophy. The method used is that of constructing a model and examining its physical characteristics and performance by simulation techniques.

Because of the simplicty of the model considered, it is hoped that this paper may also prove useful for tutorial purposes.

The computer organization proposal which follows utilizes a large read-only memory such as has been proposed by Dr. John Cocke, which might make economically feasible the increased number of instructions used to solve a given problem. In fact programs for such a computer are at least two orders of magnitude longer than those for 700 series machines. Therefore, this paper must show that this fact is more than offset by others, two of which are: 1) typical speeds easily exceed those of the IBM 650, and 2) the proposed machine uses a relatively small amount of hardware. Were these statements untrue, such an organization would not merit further consideration.

The computer under consideration will be denoted by "OBAC" (One-Bit-Adder-Computer), indicating one of its distinguishing features. Work for this paper came in two parts. First, the characteristics and organization were determined. Second, programs for several common problems were written, and then debugged by means of IBM 704 simulation.

I. The Computer

On pages 12 and 13 are information flow and detail diagrams of OBAC, the four components of which are the read-only memory, decoding and control unit, data memory, and the arithmetic unit. The bulk of the following remarks are concerned with the arithmetic unit, because it is the least conventional of the above group.

The arithmetic unit is distinguished by its very non-complex organization and its small number of electronic elements. A conventional single-bit, "full" adder is the nucleus around which control circuitry and three full word shift registers are arranged. The low bit of each of these three registers is connected to the input and output positions of the one-bit adder. Therefore all arithmetic operations take place by use of one adder, and one bit at a time. Conventional full-word operations are achieved by suitable sequences of one-bit operations, which originate serially in the instruction memory. Since the number of instruction executions needed is high, read-only memory, which is comparatively very inexpensive, is exclusively used for instruction storage.

The arithmetic unit, then, is controlled by these instructions, most of which are non-addressed arithmetic instructions, but some of which are load and store instructions, which control the parallel information flow between the shift registers and the small core memory, or data memory.

The instruction set was chosen in such a way that all operations that must be done on data can be synthesized from the elementary instructions. Therefore these instructions are completely general and no instructions need be added when new sorts of problems arise. An important advantage of this sort of operation is that logical flow of programs can be controlled much more frequently than is possible using full-word operations. It is felt that this is a valid approach to the problem of time loss due to synchoronism, although it is a departure from current methods. These conclusions and the mechanics of this computer can be clarified by the material concerning programming and simulation.

There is no great significance to be attached to the particular set of instructions chosen for the this machine. The goal was that of a generalized set of bit-manipulating instructions. There are undoubtedly other sets of instructions which are better for this purpose than the set chosen. However, the efficiency of the instruction set is not being considered as part of the present investigation. Future studies may take up this subject.

Some Changes and Unresolved Questions

Several major changes were made during the genesis of this organization. They are:

1) Inclusion of three shift registers rather than one, thereby nearly removing the expensive and hindering property of an excess of data memory references.

2) The change from normal to ring-shift registers, again to reduce data memory references.

3) The realization that an assembly program making extensive use of macro-instructions would be highly desirable, because manual programming of this computer is lengthy and tedious.

The original configuration included only the Memory Register, with no shifting ability. It was found, however, that simulation of shifting and the inconvenience of having only one register lowered the performance of the machine much more than was desirable for this investigation. The changes mentioned above were selected so as to raise the level of performance to that of a real computer, in this case the 650. This allows a more direct comparison with existing computers and also suggests some practical considerations which an OBAC would have to meet if it were actually translated into hardware.

Some possible changes, however, have been left open for later consideration.

1) The inclusion of indirect addressing or indexing operations. Neither is included in programs alread written, but either would reduce program length by a factor of ten in some cases. It is doubted however, that this would pay for the added instructions in the set, and the indexing mechanism itself.

2) In the programs written thus far the instruction memory is assumed to be random access. A completely serial, and therefore much cheaper, instruction memory could be used, if certain conditions were met. Backward transfers would have to be eliminated, and forward transfers would have to interrupt the operation of the rest of the computer until the proper instruction is reached.

It is thought that such decisions should be made at a time when the effects of the unusual characteristics are more fully understood. However the evidence, some of which is not included here, points to the conclusion that the time spent so far has not been wasted, and that this work will lead to a computer of significant utility, not inspite of, but because of, its unusual organization.

Condensed Data on OBAC

The Computer:

Tentative size of Read-Only Memory:	10^6 - 10^8 bits
Tentative size of Data Memory:	10^4 - 5×10^4 bits
Tentative speed of OBAC ("machine cycle")	0.3 - 3.0 μs
Tentative speed of data memory ("memory cycle")	2 - 10 μs
Assumed data memory word length	35 bits and sign
Assumed instruction word length (excluding addressed instructions)	5 bits

The Arithmetic Unit:

The "adder" is a so-called "full adder," with input connections "A" and "B," output connections "C" and "R," where "C" is the carry, and "R" is the low bit, resulting from the addition of "A," "B," and "C." The carry "C," then, is an input to the adder. "E," "F," and "G" are the rightmost bits respectively of the ring shift registers "X," "Y," and "Z." Shifts are one place left or right, and the shift registers are loaded from and stored into data memory in parallel.

The Instruction Set:

Number of non-addressed instructions	23
Number of instructions with data memory addresses	6
Transfer instructions	5
	34 (maximum)

Programs and Simulation:

ADD

Number of instructions in ADD program:	1750
Instructions executed in minimum case:	266
Number of data memory references:	3
Approximate time for 704 simulation:	29 ms.

MULTIPLY

Number of instructions in MPY program:	7660
Number of instructions normally executed:	2200 (approximate)
Number of data memory references:	4
Approximate time for 704 simulation:	138 ms

Conclusions:

In conclusion, we will state the justification for the two claims made on the first page of this report.

The speed of OBAC may be estimated by multiplying the average number of instructions executed by an assumed machine cycle time. For the case of addition or subtraction, the number of executions is approxmately 300. Thus, if a machine cycle of 5 usec is assumed (including instruction access and execution), the add time is 1.5 ms. For multiplication, we get 2200 x 5 us = 11 ms. Clearly, these speeds are near to the 650 level of performance.

A look at the instruction set on pages 7 & 8 will allow the reader to estimate for himself the amount of Hardware required. No professional component counts or estimates have been made, but they are planned for the future.

It is not possible at this time to compare this machine with the 650, because we do not understand the OBAC well enough, and because there are no reliable specifications for the read only memory required for this organization.

The author wishes to acknowledge the aid of Dr. John Cocke for some of ideas presented herein.

II OBAC Programming and Simulation

When several programs were written for OBAC to demonstrate that such programs could be written, and to discover the weknesses of such an organization, it became necessary to demonstrate that these programs were logically correct and efficient. This was done by simulation of the programs using an existing computer, namely the IBM 704. The first step was to compile the table of instruction equivalents which appears on pages 8 and 9. Each OBAC instruction was assigned a set of 704 instructions such that it would simulate the OBAC instruction no matter where it appeared in the program. This condition causes time inefficiency in the simulation program, but also assures generality. Second, the OBAC programs for "ADD" and "MPY" were translated into 704 programs by direct application of the table, and the resulting programs were then assembled and debugged. The completed programs were then run with several sets of data, in order to test the final correctness of the programs. Since the programs are now known to yield correct answers and since the instruction equivalents were carefully chosen, it is felt that the OBAC programs will also be found to be correct and successful, if the machine is built along the lines described.

Flow of ADD Program

Part of the OBAC ADD program, accompanied by its simulation program, appears on pages 10 through 11. The following is a short "flow explanation" of that program. The function of this program is to add two 36-bit numbers which appear in data-memory, and store the result of the addition there. Explicitly, the sum P+Q is to be stored in location S. First P and Q are loaded into the X and Y registers respectively, and the sign bits are added. If the result is "one", a transfer is made to LOCA where a subtraction routine, similar to the normal add program, appears. If the result of the sign addition is zero or two (i. e. "R" = 0), P and Q are added together one bit at a time, and the resulting bits are stored in the Z register. The sign of the answer is determined (same as carry from sign addition), and is placed in the sign position of the Z register. The Z register (answer) is then stored in S, and the program halts.

Instructions for OBAC:

Instruction Group Name	Instructions	Function
LOAD	LDX W LDY W LDZ W	Load the X, Y, or Z register from location W of data-memory.
MAKE	MCO MCZ MRO MRZ	Make the "C" (carry) or "R" (result) bit Zero or One.

PLACE	PCE PCF PCG PEA PEB PFA PFB PGA PGB PRE PRF PRG	Place the C, E, F, G, or R bit in the A, B, E, F, or G bit position
STORE	STX W STY W STZ W	Store the X, Y, or Z register in location W of data-memory.
TRANSFER (unconditional)	TRA I	Transfer control to the instruction at location I of read-only memory.
TRANSFER (conditional)	TCO I TCZ I TRO I TRZ I	Transfer control to location I of read-only memory if the "C" or "R" bit is Zero or One.
SHIFT	XLS XRS YLS YRS ZLS ZRS	Shift the X, Y, or Z register one place Left or Right.
ADD	ADD	Add the contents of the A, B, and C bit positions, place the low bit of the result in R, and the high order bit in C.

Instruction Equivalents Table for Simulation of OBAC on the 704

OBAC Instruction	704 Equivalent
ADD	CLA A ADD B ADD C LRS 1 STO C PXD LLS 1 STO R
LDX W	CLA W STO X
LDY W	CLA W STO Y
LDZ W	CLA W STO Z
MCO	CLA ONE STO C
MCZ	PXD STO C
MRO	CLA ONE STO R
MRZ	PXD STO R
PCE	CAL X ANA MIN2 ORA C SLW X
PCF	CAL Y ANA MIN2 ORA C SLW Y
PCG	CAL Z ANA MIN2 ORA C SLW Z
PEA	CAL X ANA ONE SLW A
PEB	CAL X ANA ONE SLW B

```
PFA         CAL Y
            ANA ONE
            SLW A

PFB         CAL Y
            ANA ONE
            SLW B

PGA         CAL Z
            ANA ONE
            SLW A

PGB         CAL Z
            ANA ONE
            SLW B

PRE         CAL X
            ANA MIN2
            ORA R
            SLW X

PRF         CAL Y
            ANA MIN2
            ORA R
            SLW Y

PRG         CAL Z
            ANA MIN2
            ORA R
            SLW Z

STX W       CLA X
            STO W

STY W       CLA Y
            STO W

STZ W       CLA Z
            STO W

TCO L       CLA C
            TNZ L

TCZ L       CLA C
            TZE L

TRA L       TRA L

TRO L       CLA R
            TNZ L

TRZ L       CLA R
            TZE L

XLS         LDQ X
            RQL 1
            STQ X

XRS         LDQ X
            RQL 35
            STQ X

YLS         LDQ Y
            RQL 1
            STQ Y

YRS         LDQ Y
            RQL 35
            STQ Y

ZLS         LDQ Z
            RQL 1
            STQ Z

ZRS         LDQ Z
            RQL 35
            STQ Z
```

NOTE:

MIN2 = OCTAL 777777777776

ONE = OCTAL 1

OBAC Add Program and 704 Simulation

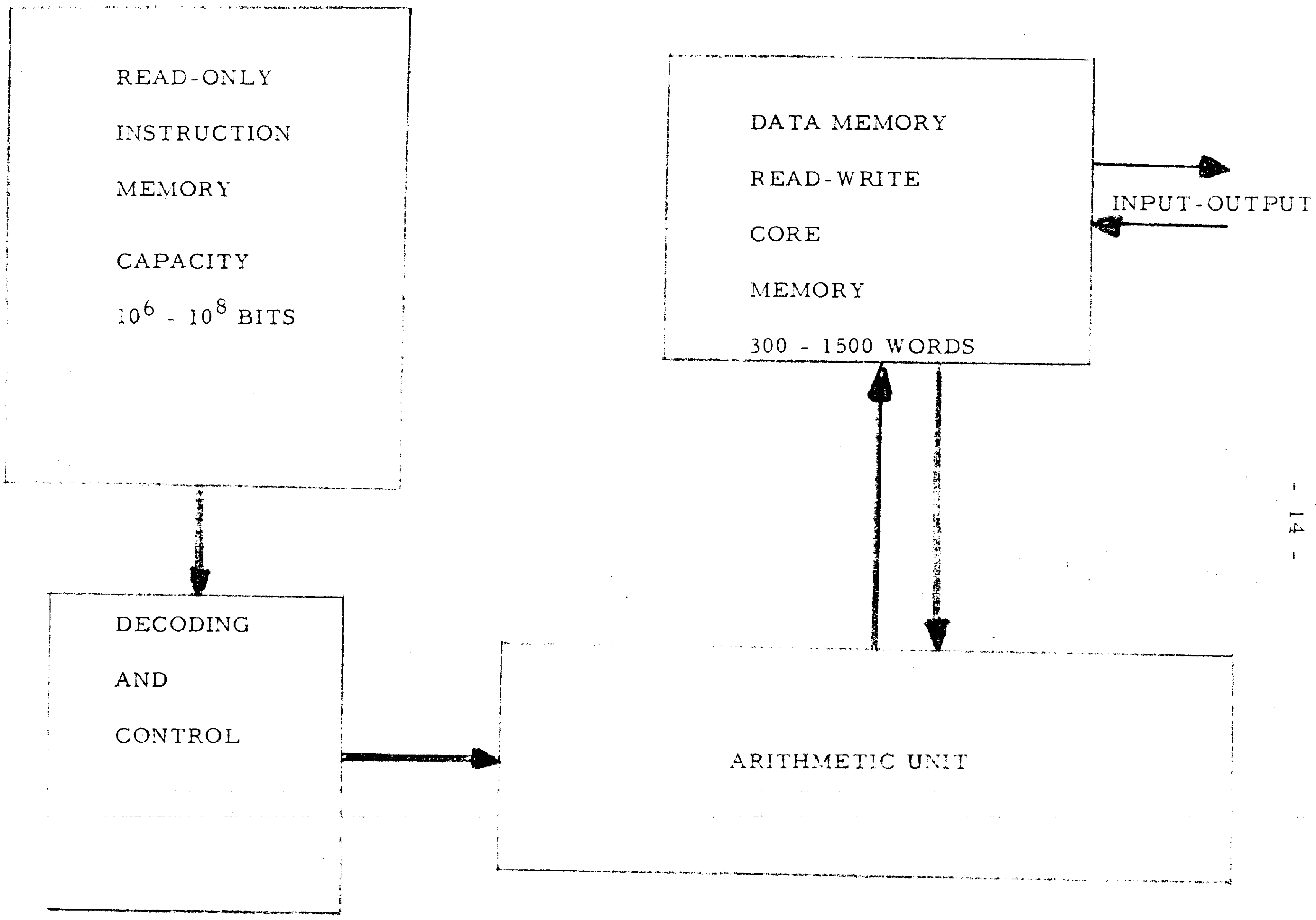
```
OBAC            704
Instruction     Simulation

MCZ             PXD
                STO C

LDX P           CLA P
                STO X

LDY Q           CLA Q
                STO Y

XLS             LDQ X
                RQL I
                STQ X

YLS             LDQ Y
                RQL I
                STQ Y

PEA             CAL X
                ANA ONE
                SLW A

PFB             CAL Y
                ANA ONE
                SLW B

ADD             CLA A
                ADD B
                ADD C
                LRS I
                STO C
                PXD
                LLS I
                STO R

TRO A           CLA R
                TNZ LOCA

MCZ             PXD
                STO C

XRS             LDQ X
                RQL 35
                STQ X

YRS                     LDQ Y
                        RQL 35
                        STQ Y
                        LXA THFV, I

PEA         LOOP 1      CAL X
                        ANA ONE
                        SLW A

PFB                     CAL Y
                        ANA ONE
                        SLW B

ADD                     CAL A
                        ADD B
                        ADD C
                        LRS I
                        STO C
                        PXD
                        LLS I
                        STO R

PRG                     CAL Z
                        ANA MIN2
                        ORA R
                        SLW Z

XRS                     LDQ X
                        RQL 35
                        STQ X

YRS                     LDQ Y
                        RQL 35
                        STQ Y

ZRS                     LDQ Z
                        RQL 35
                        STQ Z
                        TIX LOOP1, 1, 1
```

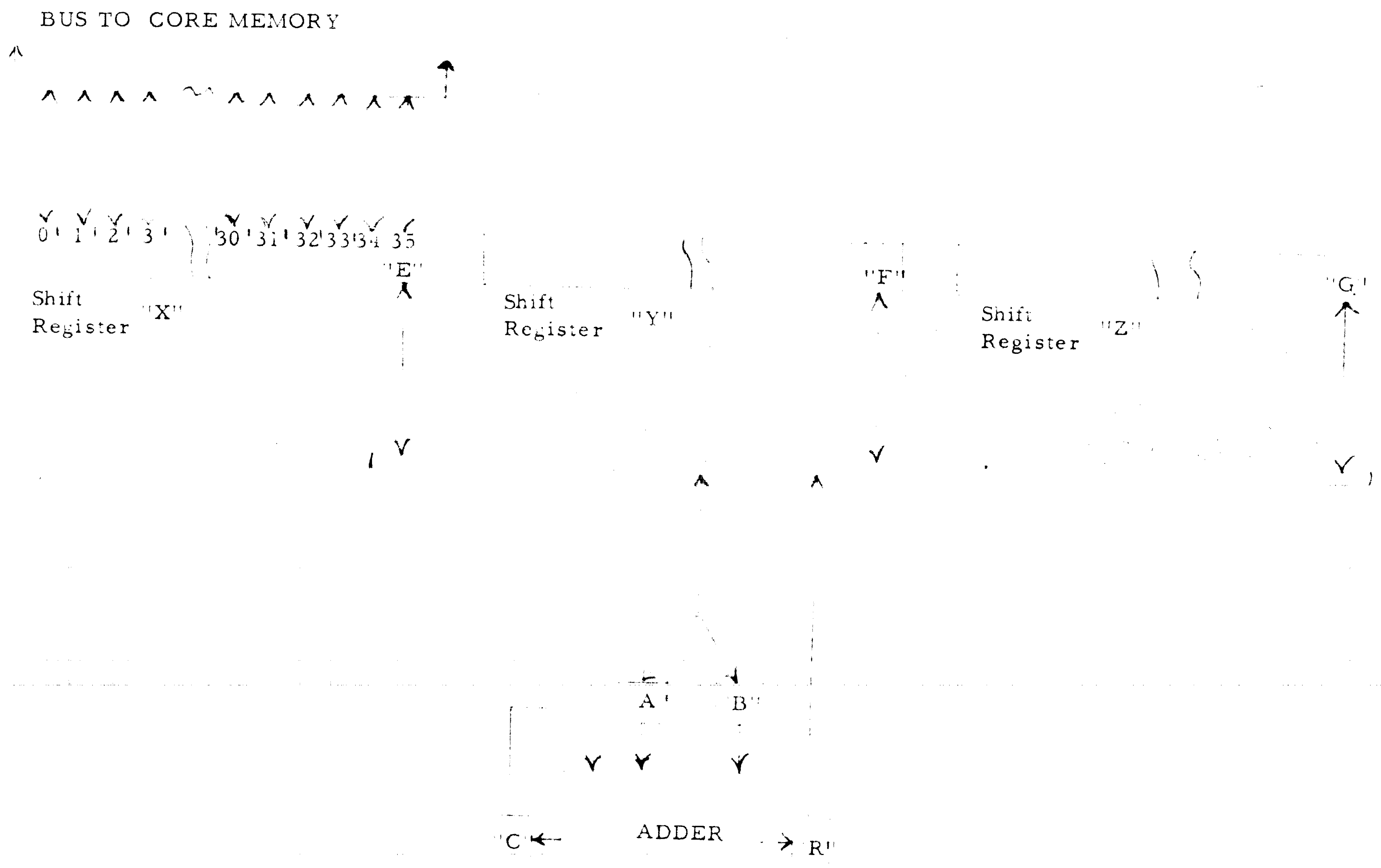
NOTE:

The previous 7 OBAC instructions are actually repeated 35 times, but are written here once for compactness.

```
MCZ              PXD
                 STO C

PEA              CAL X
                 ANA ONE
                 SLW A

PEB              CAL X
                 ANA ONE
                 SLW B

ADD              CLA A
                 ADD B
                 ADD C
                 LRS 1
                 STO C
                 PXD
                 LLS 1
                 STO R

ADD              CLA A
                 ADD B
                 ADD C
                 LRS I
                 STO C
                 PXD
                 LLS I
                 STO R

PRG              CAL Z
                 ANA MIN2
                 ORA R
                 SLW Z

ZRS              LDQ Z
                 RQL 35
                 STQ Z

STZ S            CLA Z
                 STO S
                 HTR
               A DEC 0
               B DEC 0
               C DEC 0
               P DEC 0
               Q DEC 0
               R DEC 0
               S DEC 0
               X DEC 0
               Y DEC 0
               Z DEC 0
             ONE DEC 1
            THFV DEC 35
            MIN2 OCT 777777777776
```

P and Q are initialized by means of a correction card to the binary deck which is the output of the assembly of the above 704 program.


READ-ONLY
INSTRUCTION
MEMORY
CAPACITY
10^6 - 10^8 BITS
DATA MEMORY
READ-WRITE
CORE
MEMORY
300 - 1500 WORDS
INPUT-OUTPUT
DECODING
AND
CONTROL
ARITHMETIC UNIT


BUS TO CORE MEMORY
0 1 2 3 30 31 32 33 34 35
"E"
Shift Register "X"
Shift Register "Y"
"F"
Shift Register "Z"
"G"
"A"
"B"
"C"
ADDER
"R"