On the Design of the Kyoto University Digital Computer KDC-I

By

Takeshi KIYONO*, Toshiyuki SAKAI** and Shuzo YAJIMA*

(Received October 31, 1962)

This paper describes the outline, the design features and design problems of the KDC-I, Kyoto University Digital Computer I, which has been developed and built in collaboration with Hitachi, Ltd. and put into operation since August 1960 at the computation center of the University. Emphasis is laid on the computer instructions. On other subjects only main design features are given.

The KDC-I is a serial-parallel decimal computer using transistor dynamic flip-flop circuits operating at 230 kc per second. A magnetic core and magnetic drum memory constitute the main internal memory of the computer, while magnetic tape memory is used as an external memory. Both fixed-point and floating-point operations are feasible in the computer.

The concurrent operations of the output and of the magnetic tape units with the central processing unit increase the processing speed of the computer.

The programming group at the University has already completed a fairly extensive library of subroutines and some very useful assemblers such as SYCS-1. Moreover a KDC ALGOL 60 Compiler is now under program test.

The KDC-I has already performed a vast amount of calculations on various problems, which have been submitted by the staff and students of the University. With two years of operating experience, satisfactory results are being obtained from this computer.

1. Introduction

The Kyoto University Digital Computer KDC-I[1-3] is a product of a two year project of the University commenced in fiscal 1958 aiming to establish a computation center to promote research in the University.

The KDC-I has been developed and built in collaboration with Hitachi, Ltd. and put into operation since August 1960 at the computation center of the University. Magnetic tape units have been added to the computer since October 1960 together with a magnetic tape control unit which contains a high speed magnetic core memory.

The programming group at the University has already completed a fairly extensive library of subroutines and very useful assemblers such as SYCS-1[4].

* Department of Electronics
** Department of Electrical Engineering

Moreover a KDC ALGOL 60 Compiler is now under program test[5].

In this paper, the outline, the design features and design problems of the KDC-I are presented with the emphasis on computer instructions. On other subjects only main design features are given.

Several computers had already been constructed at a few research laboratories in this country at the time of the design of the KDC-I, however some of them were reported to be having trouble concerning the reliability of the computers.

One of the design principles was therefore the construction of a very reliable computer, and at the same time a computer which would enable users to program very easily and would be fit for use by many University research workers from various fields.

The actual design of the KDC-I was begun in April 1959. In September the design had taken definite shape. The assembly had been started in November. At the end of April 1960, computation had become possible by the main part of the computer. The back panel wiring design of the KDC-I was first done involving many man-hours of work, and then in April it was redone by the newly operating KDC-I itself and the results were compared with its built-in panel wiring[7].

The process of adjustment was believed promising in which, among ninety-five varieties of one and a half address instructions, a record has been established of adjusting thirty instructions in a day and various floating-point instructions have been put into operation with almost no failures in logics and in wiring in spite of its being a new model and a first product, which we believe has indicated progress in computer design and production technique.

The adjustment of the magnetic tape units and the magnetic tape control unit consumed almost three months because of a lack of experience until various magnetic tape test programs produced by the Kyoto University Programming Group[6] were operated correctly.

The KDC-I has already performed a vast amount of calculations on various problems, which have been submitted by the staff and students of the University.

With two years of operating experience, we are gaining satisfactory results from this computer.

2. The Outline of the Computer

Before describing the details concerning the design of the computer, a summary of the features of the computer is presented in this chapter. The

operation and use of the computer are described in the KDC-I Manuals[1-3] in detail. The view of the KDC-I is shown in Fig. 1.

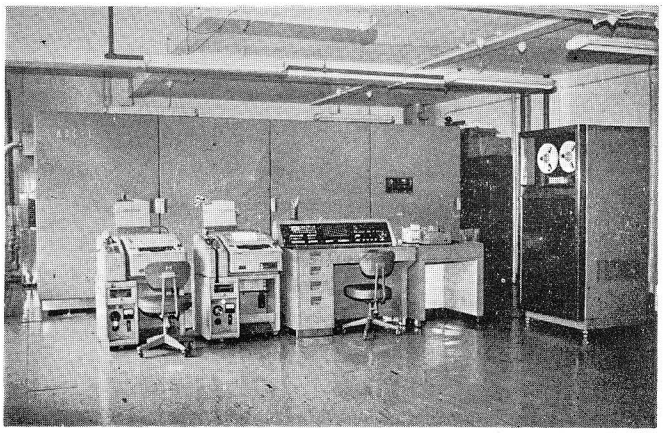


Fig. 1. View of the KDC-I.
In foreground from left to right, the off-line typewriter, the console typewriter, the operator's console, the photo-electric tape readers and the magnetic tape handler. In background from left to right, the central processing unit and the magnetic tape control unit.

(1) Type of computer: A stored program electronic digital computer.

(2) Type of information accepted by the computer: Alphanumerical characters encoded in 8-unit excess-16 binary number on perforated paper tape at the input and output (I/O) of the computer. In the computer, a series of 12 decimal digits forms a word, and each decimal number is binary coded (BCD code) and treated in parallel.

(3) Word: A word represents an instruction, a fixed-point or floating-point number or a character word. In Fig. 2 the detail of a word is shown.

(4) Instruction system: One and a half address. 95 varieties of instructions including 9 instructions for magnetic tapes. The function of clearing the contents of the Accumulator can be added to all instructions, if necessary.

(5) Operation registers and counters: A 23-digit double length Accumulator (AC, consisting of an Upper Accumulator UA and a Lower Accumulator LA), a Multiplicand-Divisor register (MD), an Order Register (OR), three 4-digit Index Registers (IR 1, 2, 3) and a 4-digit Instruction Location Counter (LC), which are all accessible from programmers. Besides, there are a Multiplier-Quotient Register (MQ), an Output Buffer Register (BR), an Input Register (IPR) and an Output

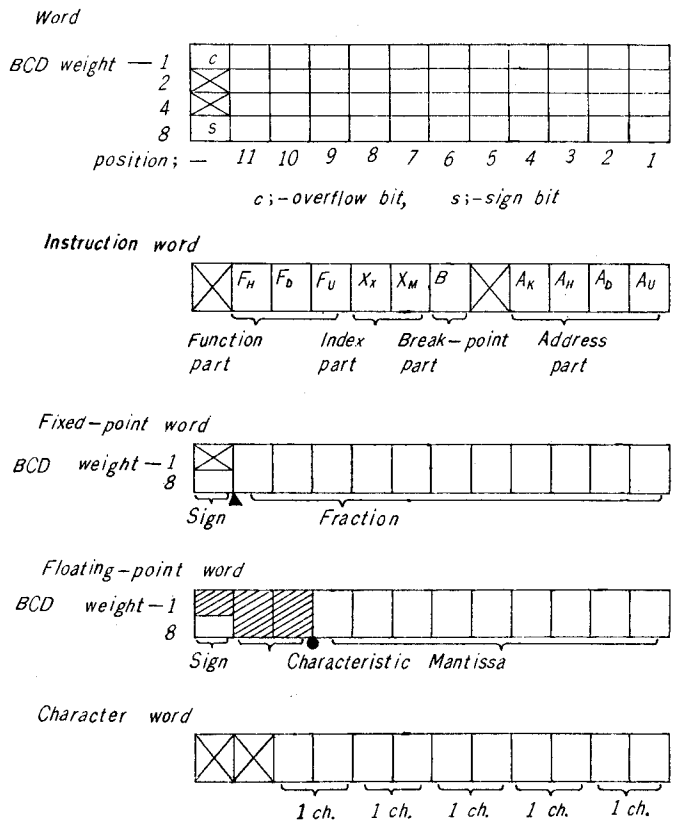


Fig. 2. Word.

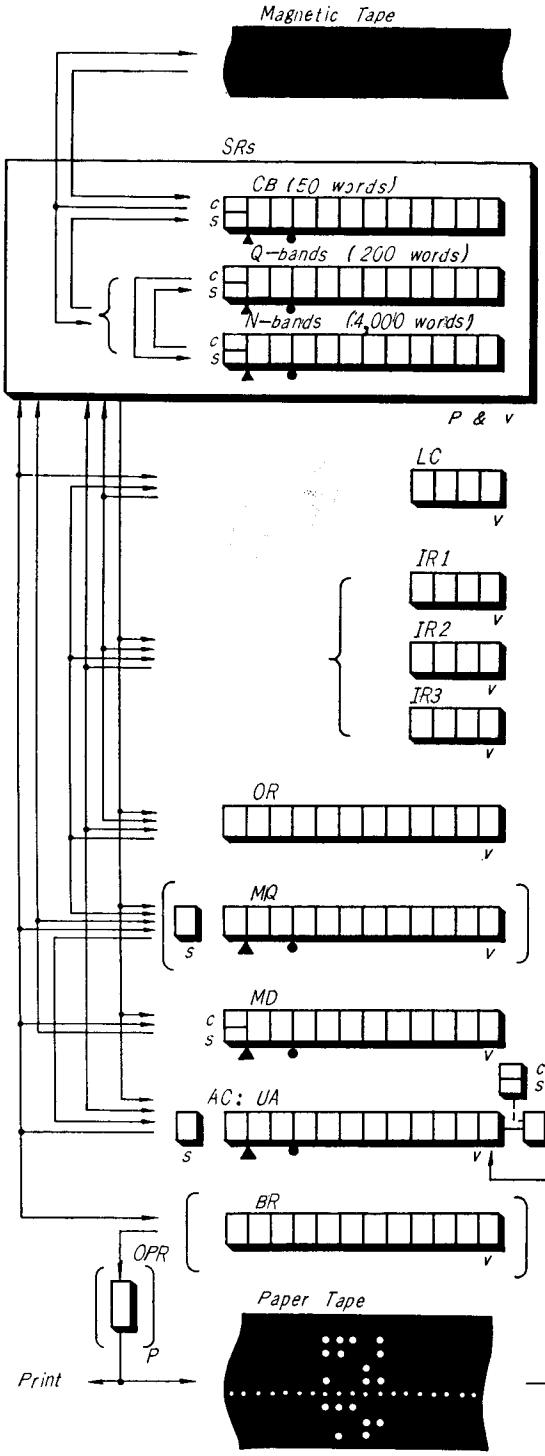
Register (OPR). Fig. 3 shows these registers and counters.

(6) Memory: 4,250 addressable storage registers in total, plus magnetic tape memory.

i) Magnetic core momery: 50 words with 50 μs access time, which also serves as a buffer storage for the magnetic tape.

ii) Magnetic drum memory: 4,000 words on 6,000 rpm magnetic drum with the average access time of 5 ms, 200 words in delay-line type quick access memory with 1.25 ms average access time on the same drum.

iii) Magnetic tape memory: Up to four magnetic tape units can be connected to the computer, two units have been attached to the computer at present. A standard reel of magnetic tape is about 1,100 meters long (3,600 ft), Mylar base magnetic coated, capable of storing about 7,000 50-word blocks, or 350,000 words, with the block-transfer-time from 220 ms minimum to 12 minutes maximum.



AC : Accumulator,
UA : Upper Accumulator,
LA : Lower Accumulator.

MD : Multiplicand-Divisor Register.

OR : Order Register.

IRs : Three Index Registers:
IR1, IR2 and IR3.

LC : Instruction Location Counter.

SRs : Storage Registers,
CB : Core Buffer Memory,
Q-bands : Quick access bands of the drum,
N-bands : Normal access bands of the drum.

( ) : Registers not accessible from programmers.

(MQ) : Multiplier-Quotient Register.

(BR) : Buffer Register.

(IPR) : Input Register.

(OPR) : Output Register.

s : Sign.
c : Overflow bit.
▲ : Fixed decimal point.
● : Floating decimal point.

p or v : Kind of checking,
p : Parity check,
v : Validity check.

Fig. 3. Main registers and counters and main information paths.

(7) Circuit: Diode logic and transistor dynamic flip-flop circuit using 3 kinds of clock pulses and operating in a single phase at 230 kc per second[8-9].

(8) Operation time: (excluding only memory access; the time for the modification of an address is included)

Addition: 0.5 ms for double length fixed-point.
1.3 ms for double length floating-point.

Multiplication: 5.8 ms average for fixed-point.
5.2 ms average for floating-point.

Division: 6.0 ms average for fixed-point.
5.1 ms average for floating-point.

(9) On-line input-output: Two photo-electric tape readers: 200 characters per second each.

Console typewriter: 8 characters per second.

A high speed punch is scheduled to be installed in the near future.

(10) Magnetic tape: 9,600 characters per second or 6.4 bits per mm. The same code as for paper tape. Off-line processors, such as magnetic tape to high speed printer, are not available at the University at present.

(11) Organization: Fig. 4 shows the physical organization of the KDC-I.

1. Central processing unit (containing magnetic drum memory).
2. Operator's console.
3. Photo-electric tape readers ……2 sets.
4. Console typewriter ……1 set.
   Off-line typewriter ……many.
   High speed punch ……(1)* set.
5. Magnetic tape control unit (containing magnetic core memory).
6. Magnetic tape handlers …2 (4)* sets.

* The number of sets scheduled for installation.

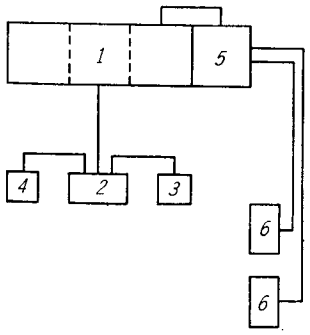


Fig. 4. Organization of the KDC-I.
1: Central processing unit.
2: Operator's console.
3: Photo-electric tape readers.
4: Console typewriter.
5: Magnetic tape control unit.
6: Magnetic tape handlers.

Extension of the magnetic core memory and addition of a high speed printer are our next objectives for the computer.

Power is supplied to the computer through motor-generator voltage regulators.

The temperature of the computer room is kept at approximately 20°C.

3. The Computer System

At the time of the design of the KDC-I several computers had already been

constructed at a few research laboratories in this country. However, some of them were reported to be having trouble concerning their reliability. The main electronic circuits of these computers used vacuum tubes, parametrons and transistors. As could be foreseen, solid-state circuits were showing more favorable performance. Among them transistor circuits have been considered most promising with regard to their reliability, their smaller power dissipation, their flexibility in regard to circuit design, their speed of operation and so on. Thus transistor circuits have been chosen for the main circuits of the KDC-I.

The most important design principle was therefore the construction of a very reliable computer, and at the same time a computer which would enable users to program very easily and would be fit for use by many University research workers from various fields. The computation speed of the computer was not necessarily considered of prime importance at the time of designing since above all a reliable computer had to be developed.

In this chapter the main system features of the KDC-I together with the main design principles of the system are presented.

(1) The building-block scheme was adopted, in which each unit, the central processing unit, the magnetic tape control unit, the magnetic tape units, etc., constituted a building-block. A high speed memory unit and high speed output equipment should be connected to the computer in the future.

(2) To ensure the reliability of the computer, parity checking and validity checking (a redundancy checking of the BCD code) were adopted. Even parity is examined at every storage register and an input register, whereas validity errors are checked at all registers. The odd character parity and the even channel parity are examined for each information block on the magnetic tape.

(3) To increase the overall processing speed of the computer, the operation of the output and of the magnetic tape units is made concurrent with that of the central processing unit.

(4) The magnetic tape units are connected to the central processing unit via a magnetic tape control unit in which a magnetic core memory is contained and used as a buffer storage as well as a high speed memory to the central processing unit.

(5) The programming ought to be as simple as possible for the general users in the University.

(6) The speed of the operation of the floating-point and shifting instructions is made high in comparison with that of other instructions.

(7) At the operator's console, the operating condition of the computer is

supervised and various controls are also possible.

(8) The adjustment and maintenance of the computer must be simple. For example a transistor dynamic flip-flop circuit together with diode logic circuits is mounted on a plug-in type printed card.

(9) The same code is used for the magnetic tape as for the paper tape, thus keeping both codes consistent.

4. Computer Instructions

The instruction system of the computer is one-and-a-half address. There are 95 varieties of instructions including 9 instructions for the magnetic tapes. The function to clear the contents of the AC can be added to all instructions, if necessary.

By providing a rather rich repertory of instructions and a decimal structure to the machine, the programming for the computer is expected to become much easier than for existing machines.

In Table 1 a list of the KDC-I instructions is presented. The numerical operation code, the symbolic code, the operation time, and the name and the explanation of each instruction are described in the ascending order of the numerical operation codes. The explanation concerning the symbolic representation of an instruction and the modification of an address is given later.

In this chapter the main features of the computer instructions are presented in the first place, and then the details of the instructions which are thought new and interesting are described.

The Main Features of the Instructions

(1) The Clear Operation: The numerical code of any instruction has an even number except that of FMP/ (221) and FMC/ (223). By adding one to any even numbered code, it is possible to clear the contents of the AC just before the operation of the instruction of an even numbered code.

(2) Fixed-Point Operations: The double length fixed-point addition and subtraction operations.

(3) Floating-Point Operations: The double length floating-point addition and subtraction operations.

(4) Conversion Operations: A fixed-point number is converted into a floating-point number and vice versa.

(5) Sign Part Operations: The sign of the AC can be set plus or can be changed.

TABLE 1. KDC-I INSTRUCTIONS

NUM. CODE	INSTRUCTION	TIME (ms)	TITLE and OPERATION
100	**ADD I J A**	0.50+Ai+Aa	Add. c(AC)pUA+c(#IJA)→c(AC).
102	**ADA I J A**	0.50+Ai+Aa	Add Absolute. c(AC)pUA+|c(#IJA)|→c(AC).
104	**SUB I J A**	0.50+Ai+Aa	Subtract. c(AC)pUA−c(#IJA)→c(AC).
106	**SBA I J A**	0.50+Ai+Aa	Subtract Absolute. c(AC)pUA−|c(#IJA)|→c(AC).
110	**ADM · · −**	0.50+Ai	Add MD. c(AC)pUA+c(MD)→c(AC).
114	**SBM · · −**	0.50+Ai	Subtract MD. c(AC)pUA−c(MD)→c(AC).
120	**MPA I J A**	5.8 av+Ai+Aa	Multiply and Add. c(AC)+c(MD)*c(#IJA)→c(AC).
122	**MPS I J A**	5.8 av+Ai+Aa	Multiply and Subtract. c(AC)−c(MD)*c(#IJA)→c(AC).
130	**RAA I J n**	0.50+Ai	Raise Address. c(AC)pUAad+#IJn→c(AC).
134	**LWA I J n**	0.50+Ai	Lower Address. c(AC)pUAad−#IJn→c(AC).
138	**RND I J n**	0.50+Ai	Round. c(AC)rnd→c(AC).
140	**ADR I J A**	6.9 av+Ai+Aa	Add, Divide and Round or Halt. [{c(AC)pUA+c(#IJA)}/c(MD)]rnd→c(UA); 0→c(LA), or Add and Halt.
150	**DVJ I J A**	6.0 av+Ai	Divide or Jump. c(AC)/c(MD)→c(UA); Rem→c(LA), or Jump, [R].
152	**DRJ I J A**	6.6 av+Ai	Divide and Round or Jump. {c(AC)/c(MD)}rnd→c(UA); 0→c(LA), or Jump.
160	**ADL I J A**	0.55+Ai+Aa	Add to LA. c(AC)pLA+c(#IJA)→c(AC).
162	**AAL I J A**	0.55+Ai+Aa	Add Absolute to LA. c(AC)pLA+|c(#IJA)|→c(AC).
164	**SBL I J A**	0.55+Ai+Aa	Subtract from LA. c(AC)pLA−c(#IJA)→c(AC).
166	**SAL I J A**	0.55+Ai+Aa	Subtract Absolute from LA. c(AC)pLA−|c(#IJA)|→c(AC).
200	**FAD I J A**	1.3+Ai+Aa	Floating Add. c(AC)+c(#IJA)→c(AC).
202	**FAA I J A**	1.3+Ai+Aa	Floating Add Absolute. c(AC)+|c(#IJA)|→c(AC).
204	**FSB I J A**	1.3+Ai+Aa	Floating Subtract. c(AC)−c(#IJA)→c(AC).
206	**FSA I J A**	1.3+Ai+Aa	Floating Subtract Absolute. c(AC)−|c(#IJA)|→c(AC).
210	**FAM · · −**	1.3+Ai	Floating Add MD. c(AC)+c(MD)→c(AC).
214	**FSM · · −**	1.3+Ai	Floating Subtract MD. c(AC)−c(MD)→c(AC).
221	**FMP/ I J A**	5.2 av+Ai+Aa	Clear and Floating Multiply. c(MD)*c(#IJA)→c(AC).

TABLE 1. (Continued)

NUM. CODE	INSTRUCTION	TIME (ms)	TITLE and OPERATION
223	**FMC/ I J A**	5.2 av + Ai + Aa	Clear, Floating Multiply and Change Sign. $-c(MD)*c(\#IJA) \rightarrow c(AC)$.
238	**FRD I J n**	0.55 + Ai	Floating Round. $c(AC)rnd \rightarrow c(AC)$.
240	**FAV I J A**	6.7 av + Ai + Aa	Floating Add, Divide and Round or Halt. $[\{c(AC)+c(\#IJA)\}/c(MD)]rnd \rightarrow c(UA)$; $0 \rightarrow c(LA)$, or Add and Halt.
250	**FDJ I J A**	5.1 av + Ai	Floating Divide or Jump. $c(AC)/c(MD) \rightarrow c(UA)$; Rem $\rightarrow c(LA)$, or Jump, [R].
252	**FDR I J A**	5.6 av + Ai	Floating Divide and Round or Jump. $\{c(AC)/c(MD)\}rnd \rightarrow c(UA)$; $0 \rightarrow c(LA)$, or Jump.
300	**STO I J A**	0.35 + Ai + Aa	Store. $c(UA) \rightarrow c(\#IJA)$.
302	**SLA I J A**	0.35 + Ai + Aa	Store LA. $c(LA) \rightarrow c(\#IJA)$, N.B. R-ind.
304	**STM I J A**	0.35 + Ai + Aa	Store MD. $c(MD) \rightarrow c(\#IJA)$.
306	**STA I J A**	0.35 + Ai + Aa	Store Address. $c(UA)ad \rightarrow c(\#IJA)ad$.
308	**STL I J A**	0.35 + Ai + Aa	Store Location. $c(LC) \rightarrow c(\#IJA)ad$.
310	**PAM · · –**	0.35 + Ai	Place UA to MD. $c(UA) \rightarrow c(MD)$.
320	**LDM I J A**	0.35 + Ai + Aa	Load MD. $c(\#IJA) \rightarrow c(MD)$.
322	**LDA I J A**	0.35 + Ai + Aa	Load Address. $c(\#IJA)ad \rightarrow c(UA)ad$.
324	**CMP I J A**	0.55 + Ai + Aa	Compare. $c(LC)+1, 2, 3 \rightarrow c(LC)$, if $c(MD) \gtrless c(\#IJA)$.
340	**FSL I J A**	0.95av + Ai + Aa	Floating Store LA. $c(LA) \rightarrow c(\#IJA)$.
360	**TLU MJ A**	5.5 av + Ai + Aa	Table Look Up. $loc(c(MD)) \rightarrow c(OR)ad$, [P], or $c(M) \rightarrow c(LC)$.
364	**LDQ QJ A**	2.9 + Ai + Aa	Load Quick Access. $c(\#JA), \cdots \rightarrow c(Q\#JA), \cdots$
366	**STQ QJ A**	2.9 + Ai + Aa	Store Quick Access. $c(Q\#JA), \cdots \rightarrow c(\#JA), \cdots$
410	**FFL · · –**	0.80 + Ai	Fixed to Floating. fx(c(AC)) to fl(c(AC)).
450	**FFX I J A**	0.55 + Ai	Floating to Fixed or Jump. fl(c(AC)) to fx(c(AC)), or Jump.
500	**EAD I J A**	0.55 + Ai + Aa	Extract and Add. $c(AC)pUA + c(\#IJA)$ & $c(MD)odd \rightarrow c(AC)$.
502	**ERE I J A**	0.35 + Ai + Aa	Extract and Replace. $c(UA)$ & $c(MD)even$ U $c(\#IJA)$ & $c(MD)odd \rightarrow c(UA)$.
510	**SSP · · –**	0.30 + Ai	Set Sign Plus. $\lvert c(AC) \rvert \rightarrow c(AC)$.
512	**CHS · · –**	0.30 + Ai	Change Sign. $-c(AC) \rightarrow c(AC)$.

TABLE 1. (Continued)

NUM. CODE	INSTRUCTION	TIME (ms)	TITLE and OPERATION
514	**NOP** · · –	0.30+Ai	No Operation.
516	**NOT** · · –	0.35+Ai	NOT. $\overline{c(UA)}$ & c(MD) U c(UA) & $\overline{c(MD)} \rightarrow$ c(UA); 0 → c(UA, 8).
518	**SCT** · · –	1.4 av+Ai	Shift and Count. LLS until 1st non-zero appears in UAm, and shift count → c(OR)ad, [P].
520	**AND I J A**	0.35+Ai+Aa	AND. {c(UA) & c(#IJA)} & c(MD) U c(UA) & $\overline{c(MD)}$ → c(UA); 0 → c(UA, 8).
522	**IOR I J A**	0.35+Ai+Aa	Inclusive OR. {c(UA) U c(#IJA)} & c(MD) U c(UA) & $\overline{c(MD)}$ → c(UA); 0 → c(UA, 8).
530	**SLS I J n**	0.55+Ai	Short Left Shift. left shift of c(UA) by $(\#IJn)_{2,1}$ places.
532	**LLS I J n**	0.55+Ai	Long Left Shift. left shift of c(AC) by $(\#IJn)_{2,1}$ places.
534	**LCS I J n**	0.55+Ai	Long Cyclic Shift. cyclic left shift of c(AC) by $c(\#IJn)_{2,1}$ places.
536	**SRS I J n**	0.55+Ai	Short Right Shift. right shift of c(UA) by $(\#IJn)_{2,1}$ places.
538	**LRS I J n**	0.55+Ai	Long Right Shift. right shift of c(AC) by $(\#IJn)_{2,1}$ places.
550	**WAN I J n**	0.35+Ai	Weighted AND. c(UA, 1) & c(UA, #IJn) → c(UA, 1); 0 → c(UA, 2, 4, 8).
552	**WOR I J n**	0.35+Ai	Weighted OR. c(UA, 1) U c(UA, #IJn) → c(UA, 1); 0 → c(UA, 2, 4, 8).
630	**SEL I J n**	0.35+Ai	Select Component. select I/O Component specified by #IJn.
632	**RIN I J n**	PTR; 200 ch/s	Read-in. read $(\#IJn)_{2,1}$ char. into c(UA) in mode specified by $(\#IJn)_4$.
634	**WRT I J n**	0.35+Ai; 8 ch/s	Write. print and/or punch $(\#IJn)_{2,1}$ char. from c(UA) in mode specified by $(\#IJn)_4$.
636	**WSP I J m**	0.35+Ai; 8 ch/s	Write Special. print and/or punch a char. specified by $(\#IJm)_{4,3}$ $(\#IJm)_{2,1}$ times.
638	**FWR** · · –	0.35+Ai; 8 ch/s	Floating Write. print and/or punch fl(c(UA)).
710	**HJM I J A**	0.35+Ai	Halt and Jump. halt and #IJA → c(LC).
712	**JSW SJ A**	0.35+Ai	Jump by Switch. #JA → c(LC), if Switch-S is on.
714	**JMP I J A**	0.35+Ai	Jump. #IJA → c(LC).
750	**JMI I J A**	0.35+Ai	Jump on Minus. #IJA → c(LC), if c(AC) <-0.
752	**JUN I J A**	0.35+Ai	Jump on UA No Zero. #IJA → c(LC), if c(UA) $\neq 0$.
754	**JNZ I J A**	0.35+Ai	Jump on No Zero. #IJA → c(LC), if c(AC) $\neq 0$.

TABLE 1. (Continued)

NUM. CODE	INSTRUCTION	TIME (ms)	TITLE and OPERATION
756	**JOV I J A**	0.35+Ai	Jump on Overflow. #IJA → c(LC), if c(UA)v ≠ 0.
758	**JEO I J A**	0.35+Ai	Jump on Exponent Overflow. #IJA → c(LC), if c(UA)v ≧ 2.
820	**LXA HJ A**	0.35+Ai+Aa	Load Index from Address. c(#JA)ad → c(H).
822	**STX HJ A**	0.35+Ai+Aa	Store Index. c(H) → c(#JA)ad.
830	**SEX HJ n**	0.35+Ai	Set Index. #Jn → c(H).
832	**RAX HJ n**	0.35+Ai	Raise Index. c(H)+#Jn → c(H).
834	**LWX HJ n**	0.35+Ai	Lower Index. c(H)−#Jn → c(H).
850	**JXL HJ A**	0.35+Ai	Jump with Index Lowered. if c(H) ≠ 0, #JA → c(LC), and c(H)-1 → c(H), or if c(H)=0, normal seq., and 9,999 → c(H).
852	**JXR HJ A**	0.35+Ai	Jump with Index Raised. if c(H) ≠ 0, #JA → c(LC), and c(H)+1 → c(H), or if c(H)=0, normal seq., and 1 → c(H).
854	**JXU HJ A**	0.35+Ai	Jump with Index Unequal. if c(H) ≠ (IRl), #JA → c(LC), and c(H)+1 → c(H), or if c(H)=c(IRl), normal seq., and c(H)+1 → c(H).
856	**JSX HJ A**	0.35+Ai	Jump after Set Index from LC. c(LC) → c(H), and #JA → c(LC).
860	**PSX I J A**	0.35+Ai+Aa	Set Pseudo Index. c(#IJA)ad → c(OR)ad, [P].
910	**BTP NJ A**	0.45+Ai; 220	Buffer to Tape. c(CB) and #JA → c(MT, N; #JA)
912	**TPB NJ A**	17.0+Ai; 220	Tape to Buffer. c(MT, N) → c(CB); if blook No.=#JA, skip next instruction, [TC].
914	**BLS NJ A**	0.45+Ai; 100*n+120	Blook Search. c(MT, N; #JA) → c(CB), [TC].
920	**DMB ·J A**	2.90+Ai+Aa	Drum to Buffer. c(#IJA), ··· → c(4,200), ···
922	**BDM ·J A**	2.90+Ai+Aa	Buffer to Drum. c(4,200), ··· → c(#IJA), ···
930	**RWD N· −**	0.45+Ai; 20; av. ca. 450 cm/s	Rewind. rewind (MT, N) to its load point.
932	**BST N· −**	0.45+Ai; 220	Back Space Tape. test c(MT, N) in backward direction, [TC].
934	**TTP N· −**	0.45+Ai; 220	Test Tape. test c(MT, N) in forward direction, [TC].
936	**ETP N· −**	0.45+Ai; 220	Erase Tape. erase one block length of (MT, N) in forward direction.
950	**JTG ·J A**	0.45+Ai	Jump on Tape Good. if TC-ind. is off, #JA → c(LC), or if on, normal seq.
952	**JTE NJ A**	0.45+Ai	Jump by Tape End. if No. N TE-ind. is on, #JA → c(LC), or if off, normal seq.

TABLE 1. (Continued)

Notes

The operation time is unchanged whether or not the modification of the address of an instruction takes place. The time for the modification is included in the operation time.
For the instructions which permit concurrent operations, the time needed for the central processing unit is written first, and the time needed for the output unit or the magnetic tape control unit second and so on.

Symbols and Abbreviations

Ai : instruction access time.
Aa : data access time.
→ : the replacement.
: the modification of the address of an instruction by the index registers, e.g. #IJA ≡ c(I) + c(J) + A (in modulo 10,000).
& : logical AND.
U : logical OR.
— : logical NOT.
rnd : round off.
Rem : remainder.
P-ind. : if the P-ind. is on, the address part of the next instruction in sequence is modified by the result in c(OR)ad of the former instruction, or if off, normal.
R-ind. : remainder indicator.
TC-ind. : magnetic tape check indicator.
[P], [R] or [TC] : the type of an instruction which affects P, R or TC-indicator.
c(X) : the contents of the register X, e.g. c(AC), c(#IJA), etc.
c(I), c(J) or c(H) : the contents of the index register IR I, J or H.
c(Y)ad : the contents of the address part of Y.
c(AC)pUA : the contents of the AC, on the UA part of which the operation is made.
c(UA)v : the content of the 12-th or the overflow digit of the UA.
c(E) & c(MD)odd : the extracted c(E) by the odd numbers in each digit of the MD.
c(CB) : the contents of the core buffer memory.
UAad : the address part of the UA.
UAm : the 11-th or the most significant digit of the UA.
(#IJn)i : the number of the i-th digit of #IJn.
loc(π/2) : the location of a data word whose contents are π/2.
fx(x) : a fixed-point number x, e.g. fx(π).
fl(y) : a floating-point number y, e.g. fl(π).
Q#JA : the address in the quick access band No. Q which satisfies Q#JA≡#JA (in modulo 50).
c(UA, i) : the contents of 11 bits specified by the weight i of the UA excluding the UA overflow digit.
(MT, N) : magnetic tape in No. N magnetic tape handler.
c(MT, N) : a block on (MT, N)
c(MT, N ; #JA) : a block whose block number is #JA on (MT, N).

(6) Shifting Operations: Six various kinds of shifting;—UA right shift, UA left shift, UA-LA right shift, UA-LA left shift, UA-LA left cyclic shift and head zero count and shift of UA-LA.

(7) Word Transfer Operations: The lower half of a double length mantissa in the AC can be stored as a floating point number with a compensating characteristic.

(8) Address Transfer Operations: Together with the address part addition and subtraction operations and index operations, the address calculation and counting are greatly facilitated by these operations. The c(LC) can be stored in the address part of a storage register.

(9) Block Transfer Operations: The instructions transfer a group of words which consists of fifty words or less between portions of the memory whose access time is different, i.e. between the normal and the quick access memory of the drum, between the drum memory and the core memory, or between the core memory and the magnetic tape memory.

(10) Control Operations: Fourteen kinds of control operation. Since the AC is a double length register, the state of not being zero should be expressed in two ways i.e. "the c(UA) are not zero" and "the c(AC) are not zero". Two control instructions which can test each of the above states are employed (JUN and JNZ). The instruction JSW can test the state of switches on the operator's console. The instruction JSX is particularly useful for the linkage of subroutine since the c(LC) is stored in one of the IRs and at the same time the control jumps to a specified location.

The instruction CMP is designed to perform the following operation;—the c(MD) are compared with the c(E) (E=the specified location). If the c(MD) is greater than the c(E), the control is in normal sequence, if equal, skips the next instruction, or if smaller, skips the next two instructions and proceeds from there. Furthermore this instruction can be used for normalized floating-point numbers as well as for fixed-point numbers, since the characteristic of a floating-point number is placed at the head of a word and a mantissa is placed next to it.

(11) Index Operations: The c(IR) can be replaced, increased, decreased or stored. Brief functions of the IRs are;—the direct indexing by the IRs, while the indirect indexing is performed by utilizing the instruction PSX. The modification of the address part of an instruction can be done twice by the c(IR) and/or c(LC). The instructions concerning the index operations can be also modified by the c(IR) or c(LC).

(12) Logical Operations: Since these operations have interesting properties, full details will be given later. Brief operations are: digitwise extract, bitwise extract, AND, OR and NOT operations.

(13) Special Operation: New instructions which are difficult to classify have been gathered, and since these instructions exhibit also interesting properties, details are fully described later.

(14) Input-Output Operations and Tape Control Codes: Selection of one or several combinations of various I/O components. Alphanumerical and numerical I/O instructions and a floating-point form output instructions. A single read-in instruction RIN can read an instruction, a fixed-point or a floating-point number by using some special characters on paper tape, which are called Tape Control Codes.

(15) Magnetic Tape Operations: Briefly the instructions perform: writing, reading, testing, erasing, rewinding and examining the tape end and the parity error. By the operations, the magnetic tape is moved forward or backward, or is rewound, or is not moved. Any four-digit block number can be written. The block number can be used for the search and the confirmation of a block. Defective portions of the magnetic tape can be detected and skipped by suitable programming.

The Details of Logical and Special Operations

Since the instructions concerning the logical operations and the special operations have various interesting properties, the full details of these instructions are described. Some symbols and abbreviations are listed beforehand for the convenience of describing the instructions as concisely as possible.

(1) The symbolic representation of an instruction: First a three-alphabet mnemonic symbolic code, second IJ if the instruction can be modified by the c(IR I) and the c(IR J), and last a symbolic address A are described.

(2) The modification of an address: A number in the address part of an instruction can be modified by the contents of the IR 1, 2, 3 and/or the LC specified by each number in the two-digit index part of an instruction. The symbol ♯ is used to show the modification.

e.g. ADD IJ A, $\sharp IJA = E \equiv c(I) + c(J) + A$ (in modulo 10,000).
ADD 12 A, $E \equiv c(IR1) + c(IR2) + A$.
ADD 34 A, $E \equiv c(IR3) + c(LC) + A$.
ADD 22 A, $E \equiv 2 * c(IR2) + A$.
ADD 01 A, $E \equiv c(IR1) + A$.
ADD 00 A, $E = A$.
SLS IJ n, $\sharp IJn \equiv c(I) + c(J) + n$,

(3) The time required for the computation: The symbol Ai is used to show the access time for an instruction while the symbol Aa means the access time for data. The time (ms) required for the computation of an instruction is given next to the name of the instruction.

Logical Operations

The following instructions are described; ERE, AND, IOR, NOT, WAN, WOR.

The logical operation is made by regarding a zero or one of a binary number, four of which can constitute a binary coded decimal number (BCD code), as a false or true value of a logical variable respectively. The truth tables for the logical AND, OR and NOT (whose symbols are &, U, $^-$) are as follows;

x	y	x & y	x U y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

x	$\bar{x}$
0	1
1	0

Each bit has a weight in BCD code, i.e. 1, 2, 4 or 8 and a bit is called a bit of the weight 1, 2, 4 or 8 respectively. A particular bit in a register is specified by a number of the digit position and a number of the weight.

The symbol c(UA, i) is used to show the contents of 11 bits specified by the weight i of the UA excluding the AC overflow digit. The c(MD) can be used as a digit-by-digit extractor for ERE (and EAD), or as a bit-by-bit extractor for AND, IOR and NOT. Numbers in a particular weight can be extracted by WAN or WOR.

502 **ERE IJ A** "Extract and Replace" (0.35+Ai+Aa)

c(UA) & c(MD)even U c(E) & c(MD)odd → c(UA).

The least significant digit of the UA as well as of the register of loc. E(=#IJA) corresponds to that of the MD, the second digit to the second, and so on.

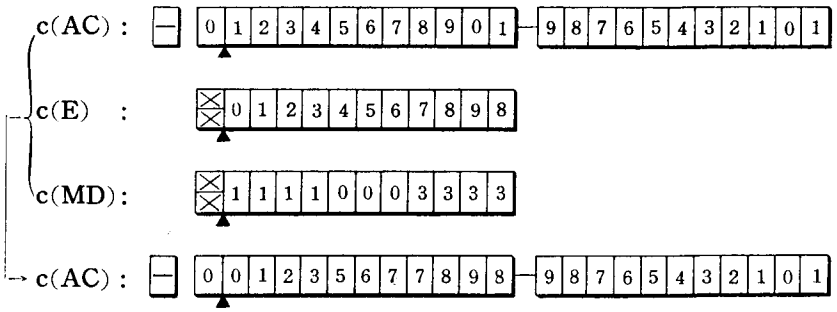
A decimal number in each digit of the UA whose corresponding digit of the MD contains an odd number is replaced by a decimal number of the corresponding digit of the register of loc. E, and remains as it is if the corresponding digit of the MD contains an even number. The c(MD) are in this sense a digit-by-digit extractor.

N.B. 1. The AC sign and a number in the AC overflow digit are unchanged. The c(LA), c(MD) and c(E) are unchanged.

2. Both a sign and a number in the overflow bit of the MD are neglected.

3. An odd number has 1 in the weight 1 (BCD code), while an even number has 0.

e.g. ERE



520 **AND IJ A** "AND" (0.35+Ai+Aa)

{c(UA) & c(E)} & c(MD) U c(UA) & $\overline{c(MD)}$ → c(UA) ; 0 → c(UA, 8).

The AND operation is made only between a binary number in each of the particular bits of the UA whose corresponding bits of the MD contain 1's and a binary number in each corresponding bit of the register of loc. E (=#IJA). The result is placed in each of the corresponding bits of the UA. The contents of bits of the UA other than the above remain unchanged, except that numbers in the weight 8 of the UA are made zeros. The c(MD) are in this sense a bit-by-bit extractor.

N.B. 1. The AC sign and a number in the AC overflow digit are unchanged. The c(LA), c(MD) and c(E) are unchanged.

522 **IOR IJ A** "Inclusive OR" (0.35+Ai+Aa)

{c(UA) U c(E)} & c(MD) U c(UA) & $\overline{c(MD)}$ → c(UA) ; 0 → c(UA, 8).

The OR operation is made only between a binary number in each of the particular bits of the UA whose corresponding bits of the MD contain 1's and a binary number in each of the corresponding bits of the register of loc. E (=#IJA). The result is placed in each of the corresponding bits of the UA. The contents of bits of the UA other than the above remain unchanged, except that numbers in the weight 8 of the UA are made zeros. The c(MD) are in this sense a bit-by-bit extractor.

N.B. 1. The AC sign and a number in the AC overflow digit are unchanged. The c(LA), c(MD) and c(E) are unchanged.

2. The following formulae are equivalent to the above ;
ç(UA) U c(E) & c(MD) → c(UA) ; 0 → c(UA, 8).

516 **NOT** ·· – "NOT" (0.3+Ai)

$\overline{c(UA)}$ & c(MD) U c(UA) & $\overline{c(MD)} \rightarrow c(UA)$; $0 \rightarrow c(UA, 8)$.

The NOT operation is performed only upon a binary number in each of the particular bits of the UA whose corresponding bits of the MD contain 1's, and the result is placed in the same bits of the UA. The contents of the bits of the UA other than the above remain unchanged, except that numbers in the weight 8 of the UA are made zeros. The c(MD) are in this sense a bit-by-bit extractor.

N.B. 1. The AC sign and a number in the AC overflow digit are unchanged. The c(LA) and the c(MD) are unchanged.
2. Numbers in the index and the address part of this instruction are neglected in the operation.
3. In other words, the above operation is the "Exclusive OR" between the c(UA) and the c(MD).

550 **WAN IJ n** "Weighted AND" (0.35+Ai)

c(UA, 1) & c(UA, ♯IJn) → c(UA, 1) ; 0 → c(UA, 2, 4, 8).

The AND operation is made between a binary number in each of the bits of the weight 1 and a binary number in each of the corresponding bits of the weight specified by ♯IJn of the UA, and the result is placed in the same bits of the weight 1 of the UA. Numbers in the weight 2, 4 and 8 of the UA are reset to zeros at the end of the operation.

N.B. 1. The AC sign and a number in the AC overflow digit are unchanged.
2. The ♯IJn should be ordinarily 2, 4 or 8 which corresponds to the weight 2, 4 or 8.
3. If ♯IJn=0 or 1, the c(UA)m...1 are reset to zeros.
If ♯IJn=3, 5 or 9, equivalent to ♯IJn=2, 4 or 8 respectively.
If ♯IJn=6 or 7, the result is equivalent to the OR between the results of WAN 00 2 and WAN 00 4.

552 **WOR IJ n** "Weighted OR" (0.35+Ai)

c(UA, 1) U c(UA, ♯IJn) → c(UA, 1) ; 0 → c(UA, 2, 4, 8).

The OR operation is made between a binary number in each of the bits of the weight 1 and a binary number in each of the corresponding bits of the weight specified by ♯IJn of the UA and the result is placed in the same bits of the weight 1 of the UA. Numbers in the weight 2, 4 and 8 of the UA are reset to zeros at the end of the operation.

N.B. 1. The AC sign and a number in the AC overflow digit are unchanged.
2. The ♯IJn should be ordinarily 2, 4 or 8 which corresponds to the weight 2, 4 or 8.

3. If ♯IJn=0 or 1, the c(UA)m...1 are reset to zeros.
   If ♯IJn=3, 5 or 9, equivalent to ♯IJn=2, 4 or 8 respectively.
   If ♯IJn=6 or 7, the result is equivalent to the OR between the result of WOR 00 2 and WOR 00 4.

Special Operations

The following instructions are described; PSX, SCT, TLU.
They perform rather complex operations and moreover as a result of the operations they modify the address part of the next instruction in sequence when the next instruction is read. To distinguish these situations, the P-indicator (also P-indicator lamp on the console) is set on. The P-indicator is set off after the modification of the address part of an instruction and at the beginning of the execution of all kinds of instructions.

860 **PSX IJ A** "Set Pseudo Index" (0.35+Ai+Aa)

Numbers in the address part of the register of loc. ♯IJA are read and reserved in the address part of the OR for the modification of the address part of the next instruction, and for this purpose the P-indicator is set on.

N.B. 1. Numbers which modify the next instruction are not ♯IJA, but the address part of its contents. This is just the same situation as where the number of an index register is referred to, but its contents are used for the modification.

e.g. 1. PSX IJ A
ADD I'J' A'
The effective address E for ADD; E≡c(♯IJA)ad+♯I'J'A' (in modulo 10,000).

e.g. 2. PSX IJ A
JXR HJ' A'
If c(H)≠0, the control jumps to E≡c(♯IJA)ad+♯J'A' (in modulo 10,000).

518 **SCT** ·· – "Shift and Count" (1.4av+Ai)

The c(AC) are shifted to the left until a non-zero number appears in the m position of the UA. The number of places shifted is counted and reserved in the address part of the OR for the modification of the address part of the next instruction, and for this purpose the P-indicator is set on.

N.B. 1. The c(UA)v is reset to zero at the beginning of the operation.
2. If c(UA)m≠0, it should be regarded as a zero-place shift.
3. If the c(AC) except the c(UA)v are all zeros, the 22-place shift takes place, so 22 is used for the modification.
4. The AC sign is unchanged.

5. Numbers in the index and the address part of the instruction are neglected in the operation.
6. This instruction is especially useful for the heading-zero suppression of the output routine, or for the search of positions of a certain pattern.

360 **TLU MJ A** "Table Look Up"

The |c(MD)| (the absolute value) are compared with the |c(E)|, ···, |c(E+i)|, ···, |c(BE)| successively (E=♯JA, BE≡199 in modulo 200, $0 \leqq BE - E < 200$, i=0, 1, ···, BE−E). If the condition, $|c(E+i)| \geqq |c(MD)|$, is detected for the first time, this E+i (not the c(E+i)) is reserved in the address part of the OR for the modification of the address part of the next instruction, and for this purpose the P-indicator is set on. The loc. E must be in the normal access memory of the drum i.e. $0 \leqq E \leqq 3999$.

If the above mentioned condition is not detected, the control jumps to the c(M). At this time the P-indicator is not on.

N.B. 1. This instruction can be used for both fixed-point and normalized floating-point numbers.
2. The contents of any register, including the c(AC), c(MD), c(E), ···, c(BE) are unchanged.
3. For comparison, the fixed point subtraction $|c(E+i)| - |c(MD)|$ is made (at the MQ). The sign of the difference decides the above condition.
4. Special care should be taken for the inclusion of a number in the overflow bit of the MD as well as of the register of loc. E in the subtraction. Thus |c(MD)| or $|c(E+i)| < 2$. This makes possible the comparison of two normalized floating-point numbers by the fixed-point subtraction.
5. A table should consist of 200 words or less, and should be stored in a band of the normal access memory of the drum (whose locations are; 0-199, 200-399, ···, or 3800-3999; 20 bands).
6. In a table, words should be ordinarily arranged in ascending order of values.
7. The M should be 1, 2 or 3 which corresponds to the IR1, 2 or 3. If M=4, ···, 9, 0 and moreover it is not the case that the P-indicator is on, then the operation is equal to NOP.

e.g. 1. A word in a table contains an argument x in its upper half, and f(x) in its lower half. The location of the table begins from ♯JA. (An argument x is calculated and is placed in the MD for comparison);

```
TLU MJ A     loc (x) → c(OR)ad
ADD/00 0     x, f(x) → c(UA)
```

Both x and f(x) are read and placed in the UA.

2. A table of arguments x and a table of functions f(x) are made separately. The location begins from #JA for the arguments, and from #JA+B for the functions;

```
TLU MJ A     loc (x) → c(OR)ad
ADD/00  B    c(loc (x)+B)=f(x) → c(UA).
```

3. A word in a table contains an argument x at its upper half, and the location of f(x) in its address part.

```
TLU MJ A     loc (x) → c(OR)ad
PSX  00 0    loc (f(x)) → c(OR)ad
ADD/00  0    f(x) → c(UA)
```

The f(x) is read and placed in the UA.

4.
```
TLU MJ A
RAA/00  0    loc (x) → c(UA)ad
```

The loc (x) is read and replaces the c(UA)ad.

5.
```
TLU MJ A
SEX  H0 0    loc (x) → c(H)
```

The loc (x) is read and replaces the c(H).

5. The Logical Design of the Computer

The logical design had been conducted under rather severe restrictions on the available number of fan-ins and fan-outs of the fundamental logic circuit or the transistor dynamic flip-flop circuit so as to get a wider performance margin of the circuit. The process of the logical design of the KDC-I could be divided into several stages. In the first place the important scheme of the logical design was decided mainly from the definition of the system, e.g. the type of registers, information paths, the place of an adder and a complementer, the main timing and so on. At the same time the logical design of basic circuits such as a decimal adder, a complementer and so on was made. Then the logic of each instruction and the relation among each of them were analysed. The flow chart was made to show a schematic diagram of the logic of instructions. Then the logical design was made piese by piese from the basic instructions. The time chart was drawn in the course of the design to show the exact time relation of each circuit.

Various trials were made, but in this paper only the main features of the logical design are described.

(1) In the floating-point operations, no special exponent register is prepared. The MQ is used for the calculation of an exponent of a floating-point number in the operation.

(2) In all arithmetic operations, the serial addition is used, and all results are placed in the double length 23-digit AC.

(3) In case of an overflow in either fixed-point or floating-point arithmetic operations, no erroneous results will be originated.
(4) When division operation is impossible, the dividend is unchanged, but the control of the sequence is changed.
(5) The operations of the instructions CMP and TLU are made chiefly at the MQ register which is hidden from programmers. These two instructions can be used for both fixed-point and floating-point numbers.
(6) In case of the operations of the instructions PSX, TLU and SCT, the address part of the Order Register OR is used effectively.
(7) From the consideration of timing the logic circuit of the AC has 24 digits. However, the instruction LCS is performed in the 23-digit AC by making a suitable compensation.
(8) In division the non-restoring method is used to get a quotient.
(9) The shifting at the AC is performed quickly by introducing a high speed shifting circuit. This feature is especially useful in shifting and the floating-point operations.

6. Computer Circuits

About 5,000 transistors and 20,000 diodes have been put into use. Various electronic circuits were designed. The main circuits are: the diode logic and transistor dynamic flip-flop circuit using 3 kinds of clock pulses and operating in a single phase at 230 kc[8-9], clock pulse supply circuits, buffer amplifier, read and write circuits and band decoder circuits for the magnetic drum, circuits to connect the I/0 components, lamp circuits, switch circuits, circuits related to magnetic cores, circuits related to magnetic tapes and power supply circuits.

The dynamic flip-flop circuit constitutes a fundamental logic circuit of the machine. In Fig. 5 the circuit diagram of the dynamic flip-flop circuit is presented. The logical function is performed by diodes and the output from the diode is regenerated by the transistor blocking oscillator, whose output is utilized for the input to the diode logic once again. The fundamental circuit is assembled on a plug-in type printed card as in the case of other circuits.

The new feature of this circuit is the addition of the diode D_6, the coil CH and the resistor R_2 across the terminals of the information storage capacitor C_1 and -1.7 V, which was developed by one of the authors T. Sakai and Prof. H. Hagiwara. One bit of information is temporarily stored in C_1, then tested, and the C_1 is reset. This process is repeated. This circuit greatly

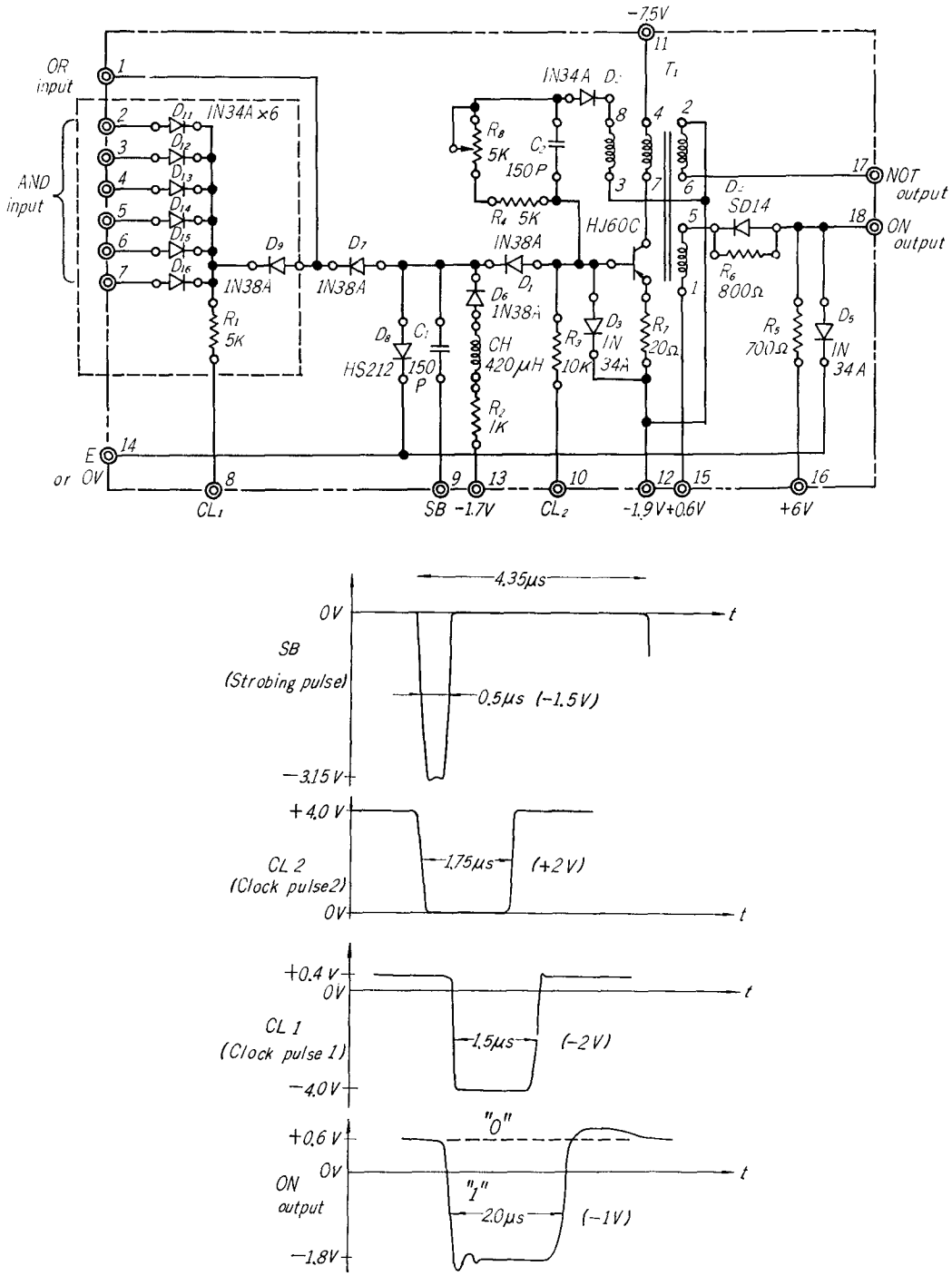


Fig. 5. The dynamic flip-flop circuit.

facilitates the discharge of the C_1 at the end of the strobing pulse, or the C_1 is reset to a satisfactory degree by the use of this circuit.

7. Conclusion

The design features and problems of various subjects of the KDC-I have been briefly described. Because of the space limitation, it was not possible to describe all the subjects in detail. Thus emphasis is laid only on the description of the computer instructions.

The KDC-I is satisfying the required functions and can be concluded to be a fairly reliable computer from actual operation. However, the speed of the computation seems progressively slower as the amount of information to be processed increases.

One way of describing the overall result of the computer is to report the data regarding the operation after the installation.

The installation of the KDC-I at the University was at the beginning of August 1960. After spending one month for the reassembly and for various tests, the KDC-I has been utilized and maintained at the University. Various interesting applications have been reported to date.

The machine is functioning about 90% of the total time that it is turned on in spite of insufficient maintenance personnel.

From our experience of its operation, the main causes of trouble were in the mechanical defects in the input and output components of the machine. The most troublesome part was incomplete soldering at the terminals of the back panel logics wiring. Several transistors have brokendown. Most of these breakdowns were due to incomplete evaluation of their power dissipation. These circuits have now been remodeled and no trouble has been registered from these circuits since then. However, some of the faulty transistors showed unfavorable changes of characteristics due to unknown reasons. No diode trouble has yet been recorded to date in spite of initial anxiety.

Acknowledgement

Until the completion of the KDC-I, many persons indeed have engaged in the plan and have worked for the realization of the machine.

The authors are very much obliged especially to Prof. K. Maeda, Prof. H. Nishihara, Prof. H. Hagiwara and Assistant Prof. S. Kato and the associated programming group for their kind cooperation and for their development of the KDC-I software, the library routines.

They owe also a considerable debt of gratitude to the then director of

the computer section Dr. S. Takada, the then vice-director of the computer section Mr. K. Iwama, Mr. Y. Hatano, Dr. K. Furuya, Mr. E. Ota, Mr. S. Iyobe, Mr. H. Mandai, Mr. H. Yazaki and Mr. K. Nishi of Hitachi, Ltd. for their co-operation in the design, construction and adjustment of the machine.

Their hearty thanks are also to Assistant Prof. H. Matsumoto of Kobe University (then assistant at Kyoto University), Assistant Prof. I. Kimura and Mr. M. Nagao of Kyoto University, the then graduate students of Kyoto University Mr. H. Fujimoto and Mr. K. Hashimoto for their kind co-operation in the development of the fundamental logic circuit of the computer, and to Mr. H. Nishio and Mr. K. Takao of Kyoto University, and the then graduate students Mr. T. Fukinuki and again Mr. K. Hashimoto for pointing out errors in the logical design of the computer.

They are very much obliged to Mr. Y. Shibatani (then assistant at Kyoto University) and Mr. I. Yagi of Kyoto University for their efforts in the development, inspection of various electronic circuits, and the maintenance of the KDC-I.

Great appreciation is expressed by the authors to all of the others who have contributed very substantially to the successful completion of the KDC-I.

References

1-3) KDC-I Manuals; Kyoto University Computation Center,
Vol. 1: "Programmer's Manual of Operation".
Vol. 2: "Programmer's Reference Manual".
Vol. 3: "Programmer's Manual of Library".

4) Takeshi Kiyono; "SYCS-1, the Symbolic Coding System of the KDC-I", Electronic Computer Tech. Committee, IECEJ*, Jan. 16, 1961.

5) Makoto Nagao; "Construction of an Automatic Programming System Based on ALGOL 60", Master Thesis at Kyoto University, 1961.

6) "Magnetic Tape Test Programmes"; Kyoto University Programming Group, Kyoto University Computation Center.

7) S. Yajima and S. Takada; "The Wiring Design of Logic Circuits by a Computer", Record of the 1960 Spring Joint Convention of IECEJ* and others.

8) S. Yajima; "A Unit to Display the Performance Margin of a Dynamic Flip-Flop Circuit", Electronic Computer Tech. Committee, Japan, IECEJ*, Jan. 16, 1961.

9) Nishino, Takahashi, Matsuzaki, Aiso, Kondo and Yoneta; "A Transistor Computer Denshi Mark IV", The Journal of the IECEJ*, pp. 1038-1045, Nov. 1959.

* IECEJ: Institute of Electrical Communication Engineers of Japan