A Symbolic Assembler for the TOSBAC - 3100

YUICHI SHINZAWA

INTRODUCTION:

Whenever one electronic computer may be developed, we are immediately confronted with the problem of the most efficient use of the computer. This report on an assembler for TOSBAC-3121 is designed for this very purpose of the TOSBAC-3100 series which were developed by the TOSHIBA ELECTRIC Co. (Japan) in 1961.

To facilitate the description of this assembler for TOSBAC-3121, at first, I will summerize the structure of TOSBAC-3121 and point out the *raison d'être* of this assembler in this introduction.

TOSBAC-3121 is a medium size high speed electronic drum computer with its own machine operation code system. The main memory consists of a magnetic drum that is broken into 20 bands as normal bands and more 5 bands as quick access bands arround the circumference of the drum. Each band is broken into 200 equal segments or words both in normal and in quick access bands on which four heads are arranged so that the main drum has 5,000 words numbered 0000-4999. An instruction which is devided into 4 parts, that is, operation code part (OP-part), next instruction part (N-part), data part (D-part), and index part (I-part), or data may be stored in each one of the 5,000 available words. When instructions are placed on the drum, they will be executed in order of the instrction of next instrction part which means $1\frac{1}{2} + 1\frac{1}{2}$ address system. These characteristics of machine language and two address systems as this computer has make it trouble for any programmers to write programs, considering the optimization of program with machine languages. Instead of using machine languages if any programmers can make pro-

grams by using symbolic languages which are quite easy for programmers to memorize and if optimization of order of instructions can be automatically executed by the machine itself, the programmers may be completely relieved from load of problem of assigning actual storage locations to instruction or quantities manipulated by any programs.

This assembler for TOSBAC-3100 series which I named TOP (TOSBAC-3100, Optimizing Program) is the symbolic assembler as I mentioned above and has the following characteristics.

1. In the TOP system we can use either machine codes or symbolic codes as operation codes.

2. We can make a program either in machine languages or in symbolic languages.

3. Optimization of assigning location to instruction is automatically executed so that it is unnecessary for a programmer to assign the next address to instruction when we do not add any specific sequence to the next instruction.

4. A programmer can decide the form of optimization on the drum.

5. A programmer can arbitrarily choose either *Normal bands* or *Quick access bands.*

6. The TOP system can be used for any compilers or simulation models.

§ 1. OPERATION CODES:

TOSBAC-3100 has its own machine code system, one code of which consists of two decimal digits (00-99) but the TOP system has symbolic codes (two alphanumerics) and pseudo codes (three alphanumerics) in addition to those machine codes. If a programmer makes a program, he can

---

The definitions of terms used in this report are as follows:

L ..... Location	D ..... Data Part
N ..... Next Part	OP ..... Operation Part
FWA ..... First Word Address	LWA ..... Last Word Address
CON ..... Condition	WE ..... Word End Mark
BE ..... Block End Mark	I ..... Index
#..... Space	NB ..... Normal Band
QB ..... Quids Band	CB ..... Conditional Bend

use machine codes and symbolic codes at the same time in one program.

The characteristics of operation codes in the TOP system are as follows:

1. Symbolic codes are translated into machine codes in this TOP assembler so that any programmers are free from recitating machin codes.

2. In one program, you can use either symbolic codes or machine codes, so that in case of transformation from a certain compiler into machine languages, operation codes are immediately translated into machine languages and it is unnecessary to convert symbolic codes at any intermediate stages from the compiler to machine codes.

Form of Operation Codes.

In order to distinguish among three groups of operation codes (machine codes, symbolic codes and pseudo codes), the operation part of the TOP system consists of three alphanumeric characters which are 6 digits from the left-most position of a machine word (12 digits),

	TOP OP	Machine Language Internal Code
	WE	MSD LSD
Machine Code	# 12	001112000000
	WE	MSD LSD
Symbolic Code	LA #	332100000000
	WE	MSD LSD
Pseudo Code	END	253524000000

One word is used as the OP-part, so we have to punch (or write) the Word End Mark (WE) at the end of the OP-part. The machine codes used in the TOP system are as quite same as those of TOSBAC-3100 but we have to pay attention to the difference in structure of characters between the TOP and TOSBAC-3100; that is, the former is two alphanumeric characters (4 digits) and the latter is two numeric characters (2 digits).

A symbolic code consists of two alphabetic characters followed by a space. Any pseude code consisting of three alphabetic characters, will

be mentioned in the next section.

§ 2. ADDRESS:

The proper address of TOSBAC-3100 is expressed only by an absolute address consisting of 4 digits but the TOP system has three expressions such as absolute address, regional address and the absolute address. As any address used in the TOP system consists of 1 word which is 12 digits, the address consists of 6 alphanumeric characters.

Absolute Address	WE # 1234 #
Symbolic Address	ABCDEF
Regional Address	WE A 1000 #

1. Absolute Address

An Absolute address of the TOP system is as same as the absolute address of machine language, that is, all addresses (0000-4999) on the main drum and 000A-000F of the registers of the computer. A left-most position is punched out with a space. If you punch (or write) the Word End Mark after an absolute address (4 numerics of alphanumerics), a space is inserted into the right-most position.

2. Symbolic Address

Symbolic Address consists of a word which has characters less than 6 symbolics. But in the case of symbolic address you have to fill the left-most position with an alphanumeric character.

Examples

a.	WE A
b.	WE ABC
c.	ABCDEF
d.	A12345

e.

WE

1ABC

f.

WE

A 1234

If a word has characters less than 6 alphanumeric characters like *a* to *e* of the above examples except *f*, you may write a word in any alphanumerics. The expression of *f* is same as a regional address so that you can not use this type of expression as a symbolic address.

3. Regional Address

The regional address is an address to specify the *i*th address of a certain region pre-defined (FWA–LWA) with an alphabetical heading character of this regional address. Therefore if a region has not been defined before with the pseudo code **Rigional,** this kind of regional address is treated as an error in the TOP system.

The regional address is expressed with one pre-defined alphabet + 4 alphanumeric characters.

a.

WE

A 1234

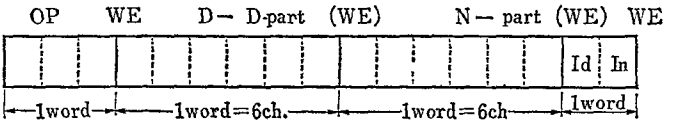
b.

B 0012 #

§ 3. THE STRUCTURE OF INSTRUCTION:

A machine instruction of TOSBAC–3100 consists of 12 decimal digits which are divided into four parts; Opertion Code Part (OP–part), Next Instruction Part (N–part), Data Part (D–part), and Index Part (I–part). But in the TOP system, each part (OP–part, D–part, N–part or I–part) consists of one word of machine language that is, 12 decimal digits. The N–part is normally used as an instruction specifying a next address in TOSBAC–3100, but in the TOP system, an optimal address is given to the N–part in relation with the nature of the OP–code and with the D–part, so that you need not to give any instruction to the N–part. Therefore the occurence of the N–part is so little that I change positions of the N–part and the D–part in the TOP system.

The order of the instructions of the TOP system is arranged in the following manner:



If the characters in the D-part or the N-part are 6 alphanumerics, you need not the Word End Mark between the D-part and the N-part.

Let us illustrate the difference between machine instructions and the TOP instructions as follows:

Example: A + B = C

When we make a program of A + B = C, considering the optimization, we have to decide the addresses of A and B in accordance with locations where the program is stored, and with the OP-codes. Let us assume 0000 to be the initial location where the program starts to be stored.

Machine Instruction

L-part	OP	N-part	D-part
0000	12	0004	0002
0004	13	0008	0006
0008	18	0012	0010
0012	XX	XXXX	XXXX

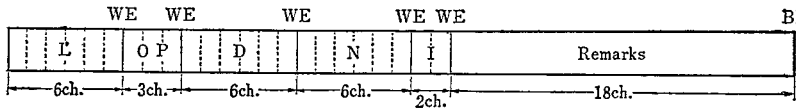
If we make the program of A + B = C with the TOP, our program will be:

```
JACK  =  0000
A     =  ADAM
B     =  BETTY
C     =  CLARE
```

L-part	OP	D-part	N-part	IdIn
JACK^{WE}	LA^{WE}	ADAM^{WE}		
	AD^{WE}	BETTY^{WE}		
	HA^{WE}	CLARE^{WE}	NEXY 1^{WE}	
NEXT 1^{WE}				

Only if we pre-assign 0000 to the initial locatiou JACK, ADAM will be automatically converted to be 0002, and the N-part of this block will be assigned 0004 with the OP code LA (2 word Time). When a symbolic program is interpreted into machine languages, the word order is changed from L + OP + D + N + Id, In, of the TOP into L + OP + N + D + In, Id, of machine language.

The TOP instruction is followed by 18 alphanumeric characters (3 words) as remarks.



§ 4. PSEUDO CODES:

Pseudo codes in the TOP system are not codes of a hardware proper but the codes to make assembling process efficient just as you can see similar codes in other computers. The TOP system has 18 pseudo codes as follows:

1.	CBA	**C**onditional **B**and **A**vailability
2.	ABC	Negation of CBA
3.	QBA	**Q**uick **B**and **A**vailability
4.	ABQ	Negation of QBA
5.	BLA	**Bl**ock **R**eservation
6.	BLA	**Bl**ock **A**vailability
7.	REG	**Reg**ional
8.	EQU	**Equ**ivaleuce
9.	SYN	**Syn**onim
10.	SAD	Successive **Ad**dress
11.	###	Plus Data
12.	-##	Minus Data
13.	ALF	**Alph**anumeric Data
14.	COM	**Com**ment
15.	; ##	Comment End

16. HED **Heading**
17. END Assembly **End**
18. IOZ **I**nput-**O**utput **Z**one

A: Specifying Program Area on Drum

Prior to assembly, the entire drum is available to the program. It is invariably necessary, however, to prevent an intermediate assembling process from occupying certain drum location, e. g., input-output zone, tables, data area, etc. The TOP system has 7 pseudo codes which can be used to restrict the assembled program to any pre-designated parts of drum.

1. CBA Conditional Band Availability

The pseudo code CBA is a characteristic code for TOSBAC-3100 assembler, which makes a band or bands on the drum available in order to assemble a program compactly. The function of this pseudo code is similar to that of BLA (Block Availability) as mentioned below, but there are some differences between them. BLA makes all specified locations available but CBA makes a band or bands available and in a case of a restriction being filled up, the next band is automatically made available. This pseudo code can not be found in any other assemblers.

	OP	D-part	N-part	IdIn
	CBA	A	△A	

The expression of *A* in the D-part is as same as an absolute address. This *A* shows the range to be optimized from 0000 of the first band (0000-0199) to a specific band.

For example, if you wish to specify the range 0000 to 0599 to be optimized as one block, you have to write this specification as follows.

OP			D-part					N-part					Id	In
C	B	A	#	0	6	0	0	#	#	0	0	5	0	#		

When you write 156, 325 or 893 in the D-part, as this pseudo code effects on an entire band, these expressions like 156, 325 or 893, are treated as 200, 400 or 1000 respectively.

$\triangle A$ in the N-part is the restriction of finding optimal condition. As assembling goes on, optimal condition gradually becomes worse in certain available bands. If you wish to make a program with optimality rather than compactness of making your program, you have to specify the $\triangle A$ in the N-part. $\triangle A$ in the N-part is more than 0000 and less than 0200. In the TOP system this restraint works as the counter. If an available address can not be found after the machine has pursued to find an available address as many times as the number of this counter ($\triangle A$), the next band may be made available.

2. QBA Quick Band Availability

This pseudo code has the same function as that of CBA, in the quick access bands.

A in the D-part is a multiples of 50 and makes locations (n × 50 from 4000) available.

$\triangle A$ in the N-part is more than 0000 and less than 50 and if this condition is filled up, the next 50 locations are automatically avilable.

L	OP			D-part	N-part	Id	In
	Q	B	A	4000+A	△A		

For example, A = 200, △A = 30

OP			D-part					N-part					Id	In
Q	A	B	#	4	2	0	0	#	#	0	0	3	0	#		

In this example the locations 4000 to 4199 are the first available locations and 30 in fhe N-part is a optimizing condition.

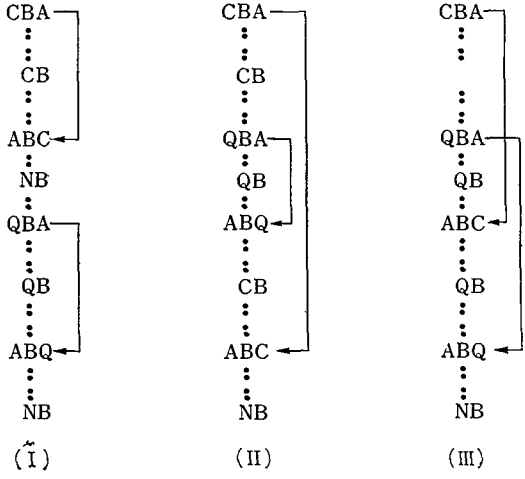
CBA effects on the entire drum but QBA effects only on the quick access band. When QBA and CBA coincide each other, the priority is given to QBA. When the CBA reaches 4999 and $\triangle A$ of CBA is not yet

filled up, the machine will search for locations which have not been yet used in quick bands. When the qnick bands are all used, the effect of QBA is cancelled out and assembling process goes to normal bands.

3. ABC Negation of CBA

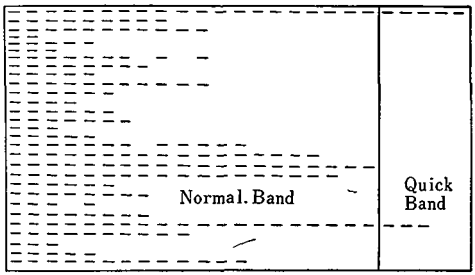
4. ABQ Negation of QBA

ABC (or ABQ) is the pseudo code cancelling the effect of CBA (or QBA). CBA (or QBA) has to correspond to ABC (or ABQ).



If you write ABC (or ABQ) corresponding to CBA (or QBA) like (I) and (II), a normal band, a conditional band and a qnick band will be bridged each other, but such an expression as in (III) is not permitted. When you wish to change the condition $\triangle A$, you can rewrite CBA or QBA.

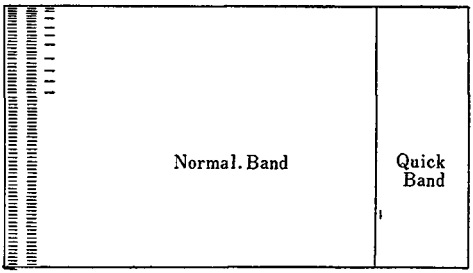
a) In the case of using neither CBA nor QBA, an assembling process will be scattered on the whole drum, and in an extreme case, the situation will be one like a following figure.



(I)

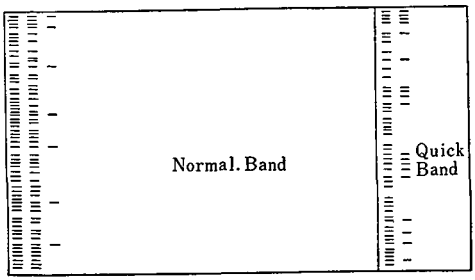
b) If you specify $\triangle A$ as 0, the function ot this CBA is as quite same as in the above case (I).

c) If you specify $\triangle A$ as 200, the band specified with A will be assembled without any blank location. But optimality of a program is neglected.



(II)

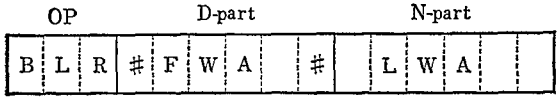
d) In an ordinary case, if you specity the $\triangle A$ of CBA of CBA or QBA as some number, the assembled program is stored like the following figure.



(III)

5. BLR Block Reservation

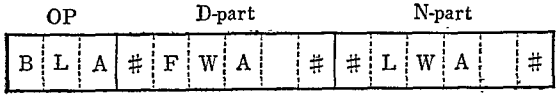
BLR is a pseudo code which makes all locations from FWA to LWA (inclusive) unavailable. This pseudo code has two absolute drum addresses FWA and LWA (FWA $\leqq$ LWA) punched in the absolute part of the D-part and the N-part respectively.



0000 $\leqq$ absolute address $\leqq$ 4999

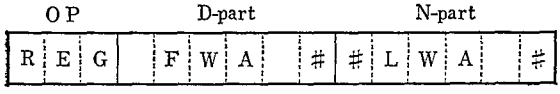
6. BLA Block Availability

The expression of this pseudo code is exactly like that of BLR excepting that it makes all locations from FWA to LWA availabie.



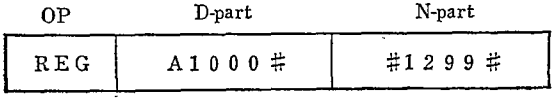
7. REG Regional

REG is a pseudo code which contains an alphabetic character, punched in the symbolizer part of the D-part, to be associated with a block defined by the terminal drum address (FWA, LWA). As in the case of a BLR, all locations from FWA to LWA are made unavailable.



FWA = AXXXX
LWA = XXXX

For instance, if a region of *A* is from 1000 to 1299, the expression is



Notice: A certain regional address in a program is the *i*th address

in this region so that if a regional address of a program is A0055 in the above example, the 55th location in the region *A*, that is, location 1054 is the proper address on the drum.

B: Pseudo Codes on the Address

We have three pseudo codes on the address, namely, EQU, SYN, and SAD.

8. EQU Equivalence

The symbol written in the D-part is assigned the equivalent of the expression writteh in the N-part. The N-part may be absolute, regional or symbolic. If the N-part is regional or symbolic, it must have been previously defined.

Case I: The N-part is absolute.

OP			D-part						N-part					
E	Q	U	T	A	R	O			#	2	0	0	0	#

TARO appeared after this pseudo code will be assigned the location 2000 in assembling.

Case II: The N-part is regional.

OP			D-part						N-part					
E	Q	U	T	A	R	O			A	0	0	0	4	#

Cace III: The N-part is symbolic.

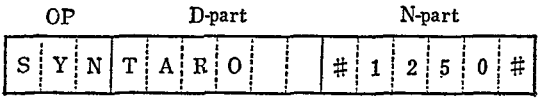
OP			D-part						N-part					
E	Q	U	T	A	R	O			J	O	H	N		

Notice: In the TOP system an absolute address is written as #XXXX#, and XXXX## is treated as symbolic.

9. SYN Synonim

The pseudo code SYN is as same as EQU excepting that the

equivalent of the expression written in the N-part are addresses (0000 ≦ N ≦ 4999) on the drum, or addresses of registers (000A, 000B, 000C). The address defined with this pseudo code on the drum is made unavailable to the program. The D-part may be absolute, regional or symbolic.



In the above example, an absolute address is assigned to TARO and is reserved.

Notice:

1) If you do not specify the N-part of the EQU (or SYN) this pseudo operation is treated as an error.
2) If you re-define the EQU (or SYN), the previous definition will be cancelled and the program will follow the new definition.

10. SAD Successive Address

The pseudo code SAD is one of the characteristic codes of the TOP system like CBA and QBA, and it can not be found in any other assemblers. In the case of over-flow, the next instruction is not transferred from the software to the hardware but automatically transferred to the N + 1 address on the drum. Therefore we have to device this operation to be included in the assembler.

SAD is the pseudo code which reserves any successive two addresses (N and N + 1) at once as the name of this code shows us. A programmer has to set this pseudo code before an instruction which may cause an overflow.

Example 1

L	OP	D-part	N-part
	SAD	ALPHA	BETA
	AD	GAMMA	ALPHA
ALPHA	HA	XXXX	XXXX
BETA (overflow)	ZZ		

The D-part of SAD is one location N and the N-part of SAD is the other location N + 1. Then in the above example

ALPHA = N
BETA = N + 1

But, when SAD is inserted into the program, if the symbol in the D-part of SAD previously appeared, the symbol location in the D-part. Therefore, this kind of risk can be avoided by (1) placing the SAD at the top of a program or (2) adding a new symbol to the program.

Loc.	OP	D-part	N-part
	XX	ALPHA	
	XX	BETA	
	XX	GAMMA	
	XX	DELTA	
	SAD	ALPHA	EPSIL
	XX	BETA	ALPHA
ALPHA	XX	XXXX	XXXX

(I)

In this example, EPSIL does not necessarily succeed ALPHA (N) as N + 1.

Loc.	OP	D-part	N-part
	SAD	ALPHA	EPSIL
	XX	ALPHA	
	XX	BETA	
	XX	GAMMA	
	XX	DELTA	
	XX	BETA	ALPHA
ALPHA	XX	XXXX	XXXX
EPSIL	ZZ		

In the (II), as SAD is placed at the TOP of this program;

ALPHA = N
EPSIL = N + 1

Loc.	OP	D-part	N-part
	XX	ALPHA	
	XX	DETA	
	XX	GAMMA	

	XX	DELTA	
	SAD	EPSIL	ZETA
	XX	BETA	EPSIL
EPSIL	XX	XXXX	ALPHA
ZETA	ZZ		
ALPHA	XX	XXXX	XXXX

In the (III), if a new symbol is added as one step, we have no risk like in the case (I).

C: Pseudo Codes Concerning Data

We have three kinds of pseudo codes as Data codes, that is, ###, -##, and ALF.

11. ### Plus Data

12. -## Minus Data

means no punching (or writing) in the OP-part. Data with ### or -## consists of numeric 12 digits of alphanumerics. A location may be written with absolute, symbolic or regional address.

Example:

Plus Data

Loc.	OP	D-part	N-part
TARO		123456	789012

Minus Data

Loc.	OP	D-part	N-part
JIRO		123456	789012

123456789012 will be stored into a symbol location TARO, and -123456789012 will be stored into a symbol location JIRO.

13. ALF Alphanumeric Data

This pseudo code is a code which converts alphanumeric 6 digits written in the D-part into numeric 12 digits of machine language, and stores them into the location specified with the L-part.

Example:

Loc.	OP	D-part	N-part
TARO	ALF	ABCDEF	

Numerics 212223242526 of ABCDEF will be stored into a symbol location TARO. Location and D-part may be absolute or symbolic.

D: Other Pseudo Codes

The TOP system has 5 more kinds of pseudo codes like COM, ;##, HED, END and IOZ, in addition to the above 13 codes.

14. COM Comment

15. ;## Comment

You can write anything as a comment after the pseudo code COM, which does not effect on assembling, but is printed out. If you write the pseudo code ;## in the OP-part of the next line after finishing a comment, this comment operation stops and a normal assembling operation starts. The numbers of digits and characters of a comment are not limited, but if you miss either the block end mark after the comment or do not write ;## in the proper position in the OP-part, this comment operation will not cease.

Example:

Loc.	OP	D-part	N-part	I	Remarks
XXXXXX (WE)	XX (WE)	XXXXXX (WE, BE)			
	COM (WE)	THIS IS A COMMENT OF THE ASSEMBLY PROGRAM BY THE TOSBAC-3100 OPTIMZING PROGRAM			
	; (WE, BE)				

16. HED Heading

This pseudo code HED is used to avoid duplicity of symbols when several programs or several sections of a single program are to be assembled together. The need for heading is paramount if several persons have contributed to a program or when a program employs symbolic library routines. When you wish to combine *A* program with *B* program,

if you have common symbols used in both A and B without any heading, you can combine them together. If you add various headings to various programs, they are assembled as different programs.

In the TOP system, heading is accomplished by the automatic insertion of a heading character into the right-most position of symbols having this position blank. But when an address is filled with 6 alphanumeric characters, no heading character is added to this symbol. Therefore, in order to make heading effective, all symbols to be assembled have to be less than 5 alphanumeric characters.

Example:

In the program,

OP	D-part
HED	A

XX# T A R O

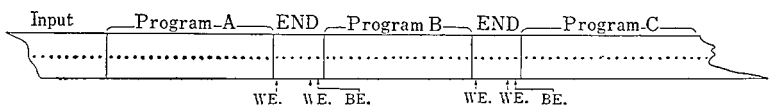
XX# J I R O

Symbols TARO, and JIRO subsequent to the heading to A is processed by the assembly program as if it were actually the symbols

OP	D-part
XX#	TARO#A
XX#	JIRO#A

17. END Assembly End

This pseudo code is a code which terminates assembling process. Any of independent programs may be assembled in one pass, when you merely place the pseudo code END between each program, i. e.,



18. IOZ Input-Output Zone

The last pseudo code is IOZ which specifies the localions-in every 10th location (00, 10, 20, 30, 40 ..... 90) from the first location of a

band (0000, 0200, 0400.....1000....2000....3000, 4000) as the I-O zone in a program and makes them unavailable for other purposes.

The first location of I-O zone should be specified by an expression *A* (absolute, regional, or symbolic) of the D-part after this code.

OP	D-part	N-part
IOZ	A	△A

A consist of 5 alphanumeric + 1 numeric (0~9) and △*A* has to be the first location of I-O zone, expressed by an absolute address.

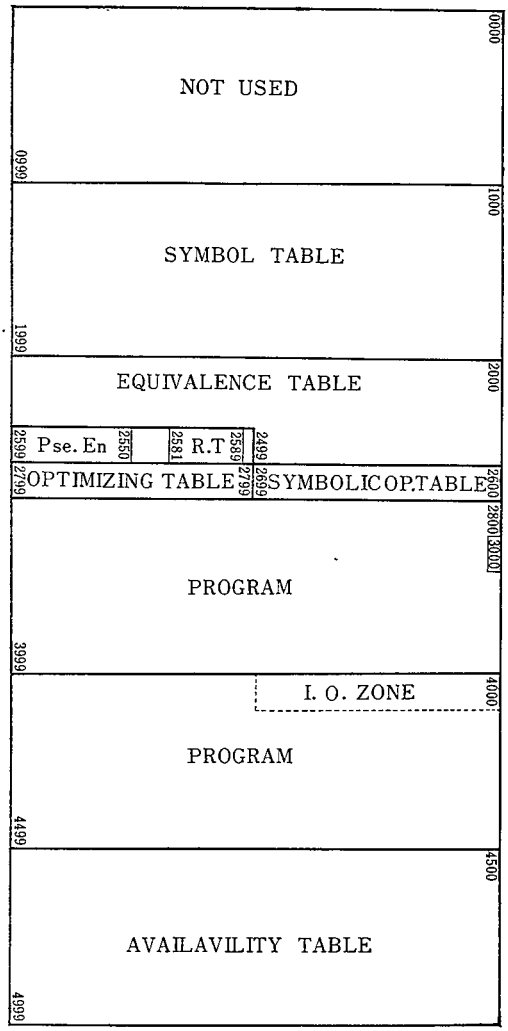
The number (0~9) at the left most position of the D-part indicates the ith location of the I-O zone.

§ 5. DRUM ALLOCATION and others:

	from	to	No. of loc.
SYMBOL TABLE	1000	1999	1000
EQUIVALENCE TABLE	2000	2499	500
AVAILABILITY TABLE	4500	4999	500
OPTIMIZNG TABLE	2700	2799	100
REGION TABLE	2501	2529	29
PSEUDO OP ENTRIES	2550	2599	50
SYMBOLIC OP TABLE	2600	2699	100
READ & PUNCH ZONE	4000	**** 4090	10
PROGRAM ENTRY	3000		1

READ & PUNCH ZONE

I-O Zne	Function	Character
4000	Location (Alph. numeric)	6
4010	OP code 〃	3
4020	Data address 〃	6
4030	Next instruction 〃	6
4040	Index 〃	2
4050		
4060	Remarks 〃	18
4070		
4080	Location (Numeric)	4
4090	Assembled instruction 〃	12


0000
NOT USED
0999
1000
SYMBOL TABLE
1999
2000
EQUIVALENCE TABLE
2599
Pse. En
2550
2581
R.T
2589
2499
2599
2700
OPTIMIZING TABLE
2799
SYMBOLIC OP. TABLE
2800
3000
3000
PROGRAM
3999
I. O. ZONE
4000
PROGRAM
4499
4500
AVAILAVILITY TABLE
4999

SYMBOLIC OPERATION CODES

CLASSIFICATION	NUMERICAL	TIME (WT)	SYMBOLIC	OPERATION
TRANSFER OPERATIONS	10	4	CA	Clear rA
	20	4	CB	Clear rB
	40	4	CC	Clear rC
	12	4	LA	Load rA
	22	4	LB	Load rB
	42	4	LC	Load rC
	18	4	HA	Hold rA
	28	4	HB	Hold rB
	48	4	HC	Hold rC
ARITHMETIC OPERATIONS	13	4 or 5	AD	Add
	17	4 or 5	SB	Subtract
	33	71 (avr.)	ML	Multiply
	37	95 (avr.)	DV	devide
LOGICAL OPERATIONS	11	5	SL	Shift left
	15	4	SR	Shift right
	51	4	JQ	Jump equal
	59	4 or 5	JT	Jump at threshold
	00	4	NO	No effect
	01	4	ZZ	Halt jump
	97	4	BA	Bitwise add.
	93	4	BM	Bitwise multipl.
INDEX REGISTER OPERATIONS	80	4	CI	Clear index
	83	4	LI	Load index
	88	4	HI	Hold index
	84	4	AI	Add index
	87	4	SI	Subtract index
	81	4 or 5	IQ	Index equal
	89	4 or 5	IT	Index at threshold
FLOATING POINT OPERATIONS	70	8 (avr.)	FU	Unnormalized fl. add.
	71	9 (avr.)	FA	Floating add.
	72	9 (avr.)	FS	Floating subtract.
	73	67 (avr.)	FM	Floating multipl.
	77	84 (avr.)	FD	Floating division
INPUT & OUTPUT OPERATIONS	02		PO	Print out
	08		RI	Read in

References for this report:

1. SOAP II, International Business Machine Corporation, 1956.
2. Programming manual for TOSBAC-3121, Vol. 1, 1960, for TOSBAC-3100, 1961, Vol. I, II, Tokyo-Shibaura Electric Co, 1962.
3. Shigeichi Moriguchi, Some topics related to programming — Coordination of SIP languages; Convergence and rounding error in iterative processes, Information processing Vol. 1, No, 2, 1960.
4. Digital Computer Programming, Handbook of Automation, Computation, and Control, Vol. 2, 1959.
5. Report on the Algorithmic Language ALGOL 60, Communications of the ACM Vol. 3, No. 5, 1960.

The author wishes to acknowlege with profound thanks, the assistance, kindness and cooperation of Mr. TAKAO OKAMOTO, Toshiba Electric Co.