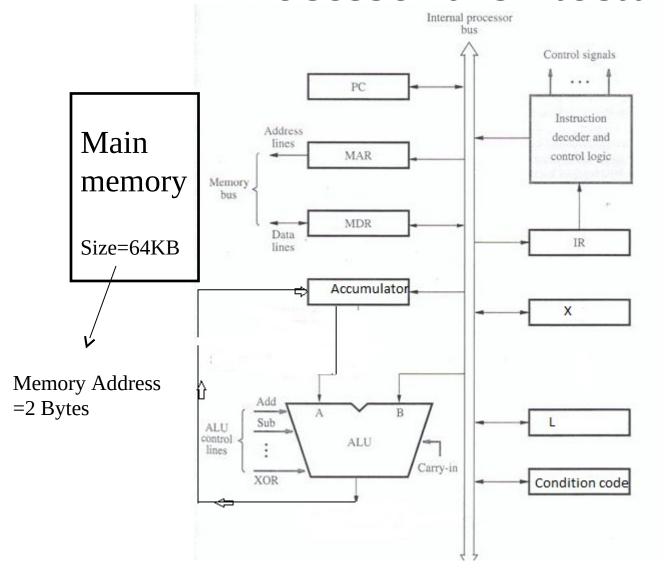
Assembler design

Hypothetical machine Processor architecture



Simplified
Instructional
Computer
(SIC)

Word length

Accumulator, Registers X, L

	Byte	Byte	Byte	3 Bytes
Address: 2 bytes		Main Memory		
4000	Byte	Byte	Byte	
4003				
4006				
4009				
4012				

Instruction set of SIC

Mnemonic	Format	Opcode	Effect
ADD m	3/4	18	$A \leftarrow (A) + (mm+2)$
AND m	3/4	40	A ← (A) & (mm+2)
COMP m	3/4	28	(A): (mm+2)
DIV m	3/4	24	$A \leftarrow (A) / (mm+2)$
		The state of the s	
m	3/4	3C	PC ← m
EQ m	3/4	30	PC ← m if CC set to =
GT m	3/4	34	PC ← m if CC set to >
LT m	3/4	38	PC ← m if CC set to <
SUB m	3/4 .	48	$L \leftarrow (PC); PC \leftarrow m$
LDA m	3/4	00	A ← (mm+2)
			(*
LDCH m	3/4	50	A [rightmost byte] \leftarrow (m)
Picar a		***	
LDL m	3/4	08	L ← (mm+2)
-	Mary Hart		2
	211		-
DX m	3/4	04	X ← (mm+2)

Mnemonic	Format	Opcode	Effect	Notes
OR m	3/4	44	$A \leftarrow (A) \mid (mm+2)$	
RD m	3/4	D8	A [rightmost byte] \leftarrow data	
			from device specified by (m)	
RSUB	3/4	4C	PC ← (L)	
6				
	Printer better		44 (4)	
			r'}	
	al-basa	1		
			7	
	-	Г	- for address m	
STA m	3/4	0C	mm+2 ← (A)	
-	2/1			
STCH m	3/4	54	$m \leftarrow (A)$ [rightmost byte]	
none.				
TITLE		1	(21)	1
STL m	3/4	14	mm+2 ← (L)	
				-
STSW m	3/4	E8	mm+2 ← (SW)	P
STX m	3/4	10	mm+2 ← (X)	
SUB m	3/4	1C	$A \leftarrow (A) - (mm+2)$	
			((

Mnemonic	Format	Opcode	Effect	Notes
	-		CVC interest (In	14
010			E	
TD m	3/4	EO	Test device specified by (m)	*
·				
TIX m	3/4	2C	$X \leftarrow (X) + 1$; (X) : $(mm+2)$	1
		A In the second		
WD m	3/4	DC	Device specified by $(m) \leftarrow (A)$ [rightmost byte]	

Instruction format

Op code (1 byte) Address (2 bytes)

Sample code (data movement)

LABEL	Instruction	Operand
TEST	START	1003
FIRST	LDA	FIVE
	STA	ALPHA
ALPHA	RESW 1	/*symbolic variable*/
FIVE */	WORD	5 /*symbolic constant, Literal
	END	FIRST Psuedo Opcode OR

Assembly language program

Assembler Directives

START

RESW

WORD,

END

Sample code (data movement)

Loc	LABEL	Instr	uction	Operand	Object code
	TEST	START	100	3	-
1003	FIRST	LDA FIV	/E	001012	
1006	STA	ALPHA	0C100	9	
1009	ALPHA	RESW 1		*****	
1012	FIVE	WORD	5	00000)5
1015	END	FIRST			

Assembly language program

Sample code (Arithmetic operation)

LDA ALPHA

ADDINCR

SUB ONE

STA BETA

ONE WORD 1

ALPHA RESW 1

BETA RESW 1

INCR RESW 1

Assembler

Assembly language Assembler —>Object code

Sample code (data movement)

Loc	LABEL	Instr	uction	Operand	Object code
1003	TEST	START	100	3	
1003	FIRST	LDA FIV	/E	001012	
1006	STA	ALPHA	0C100	9	
1009	ALPHA	RESW 1		*****	
1012	FIVE	WORD	5	00000)5
1013	END	FIRST			

Assembly language program

How to design an Assembler

Data structures

(1) OPTAB

Instructi on	Op code	Length (bytes)
ADD m	18	3
LDA m	00	3

(3) Symbol Table

LABEL	Address (LOC value)
FIRST	1003
ALPHA	1009
FIVE	1012

(2) Location counter (LOC)

Instructi on	Op code	Length (bytes)
ADD m	18	3
LDA m	00	3

Data Structi

- Operation Code Table (OPTAB)
 - Used to look up mnemonic operation codes and translate them into machine language equivalents
 - Contains the mnemonic operation code and its machine language equivalent
 - In more complex assemblers, contains information like instruction format and length

Data	Struc
D aca	Juan

LABEL	Address (LOC value)
FIRST	1003
ALPHA	1009
FIVE	1012

Symbol Table

- Used to store values (addresses) assigned to labels
- Includes the name and value for each label
- Flags to indicate error conditions, e.g. duplicate definition of labels
- May contain other info like type or length about the data area or instruction labeled

Data Structures

LOCCTR

- Used to help in the assignment of addresses
- Initialized to the beginning address specified in the START statement
- After each source statement is processed, the length of the assembled instruction or data area to be generated is added
- Gives the address of a label

How to design an Assembler

Two pass algorithm

Pass 1 (Define symbols):

- (a) Assign addresses to all statements (LOC)
- (b) Save the addresses assigned to all labels in symbol table
- (c) Perform some processing for assembler directives

How to design an Assembler

Two pass algorithm

Pass 2 (Generate object code):

- a. Translate opcode and operands
- b. Generate data values for WORD
- c. Write object program

Sample code (data movement)

Loc	LABEL	Instr	uction	Operand	Object code
1003	TEST	START	100	3	
1003	FIRST	LDA FIV	/E	001012	
1006	STA	ALPHA	0C100	9	
1009	ALPHA	RESW 1		*****	
1012	FIVE	WORD	5	00000)5
1013	END	FIRST			

Assembly language program

```
Pass 1:
 begin
   read first input line
   if OPCODE = 'START' then
      begin
          save #[OPERAND] as starting address
          initialize LOCCTR to starting address
          write line to intermediate file
          read next input line
      end {if START}
   else
      initialize LOCCTR to 0
   while OPCODE ≠ 'END' do
      begin
          if this is not a comment line then
             begin
                 if there is a symbol in the LABEL field then
                        search SYMTAB for LABEL
                        if found then
                            set error flag (duplicate symbol)
                        else
                            insert (LABEL, LOCCTR) into SYMTAB
                     end {if symbol}
                 search OPTAB for OPCODE
                 if found then
                     add 3 {instruction length} to LOCCTR
                 else if OPCODE = 'WORD' then
                     add 3 to LOCCTR
                 else if OPCODE = 'RESW' then
                     add 3 * #[OPERAND] to LOCCTR
                 else if OPCODE = 'RESB' then
                     add #[OPERAND] to LOCCTR
                 else if OPCODE = 'BYTE' then
                    begin
                        find length of constant in bytes
                        add length to LOCCTR
                     end {if BYTE}
                 else
                     set error flag (invalid operation code)
              end {if not a comment}
          write line to intermediate file
          read next input line
      end {while not END}
   write last line to intermediate file
   save (LOCCTR - starting address) as program length
end {Pass 1}
```

Figure 2.4(a) Algorithm for Pass 1 of assembler.

end {Pass 2}

```
Pass 2:
begin
   read first input line (from intermediate file)
   if OPCODE = 'START' then
      begin
          write listing line
          read next input line
       end {if START}
   write Header record to object program
   initialize first Text record
   while OPCODE # 'END' do
      begin
          if this is not a comment line then
              begin
                 search OPTAB for OPCODE
                 if found then
                     begin
                        if there is a symbol in OPERAND field then
                            begin
                                search SYMTAB for OPERAND
                                if found then
                                   store symbol value as operand address
                                else
                                   begin
                                       store 0 as operand address
                                       set error flag (undefined symbol)
                                   end
                            end {if symbol}
                         else
                            store 0 as operand address
                         assemble the object code instruction
                     end {if opcode found}
                  else if OPCODE = 'BYTE' or 'WORD' then
                     convert constant to object code
                  if object code will not fit into the current Text record then
                     begin
                         write Text record to object program
                         initialize new Text record
                     end
                  add object code to Text record
              end {if not comment}
          write listing line
           read next input line
       end {while not END}
    write last Text record to object program
    write End record to object program
    write last listing line
```

Figure 2.4(b) Algorithm for Pass 2 of assembler.

10	Line	Loc	Sou	rce staten	nent	Object code
10	_	1000	CORY	CMADM	1000	
15	ESPECIAL CONTRACTOR OF THE PROPERTY OF THE PRO					141033
200	500 MAGE.					
1009	District Co.		CHOOL			
30						
100						
1012						
1015						
1018	CONTRACTOR OF THE PROPERTY OF		ENDETT.			
LDA						0C1039
STA						00102D
1021				STA	LENGTH	0C1036
TO				JSUB	WRREC	482061
1027	BUCKEROVY -				RETADR	081033
85				RSUB		4C0000
90 1030 ZERO WORD 0 0000000 95 1033 RETADR RESW 1 100 1036 LENGTH RESW 1 105 1039 BUFFER RESB 4096 110	80	102A	EOF	BYTE	C'EOF'	454F46
95 1033 RETADR RESW 1 100 1036 LENGTH RESW 1 105 1039 BUFFER RESB 4096 110 115 . SUBROUTINE TO READ RECORD INTO BUFFER 120 125 2039 RDREC LDX ZERO 041030 130 203C LDA ZERO 001030 135 203F RLOOP TD INPUT E0205D 140 2042 JEQ RLOOP 30203F 145 2045 RD INPUT D8205D 150 2048 COMP ZERO 281030 155 204B JEQ EXIT 302057 160 204E STCH BUFFER, X 549039 165 2051 TIX MAXLEN 2C205E 170 2054 JLT RLOOP 38203F 175 2057 EXIT STX LENGTH 101036 185 205D INPUT BYTE X'F1' F1 190 205E MAXLEN WORD 4096 001000	85	102D	THREE	WORD	3	000003
100	90	1030	ZERO	WORD	0	000000
105 1039 BUFFER RESB 4096 110	.95	1033	RETADR	RESW	1.	
SUBROUTINE TO READ RECORD INTO BUFFER SUBPOUT SERO STANDARD SUBROUTINE TO READ RECORD INTO BUFFER SUBPOUT SERO SUBPOUT SERO SUBPOUT	100	1036	LENGTH	RESW	1	
SUBROUTINE TO READ RECORD INTO BUFFER		1039	BUFFER	RESB	4096	
120 125 2039 RDREC LDX ZERO 041030 130 203C LDA ZERO 001030 135 203F RLOOP TD INPUT E0205D 140 2042 JEQ RLOOP 30203F 145 2045 RD INPUT D8205D 150 2048 COMP ZERO 281030 155 204B JEQ EXIT 302057 160 204E STCH BUFFER, X 549039 165 2051 TIX MAXLEN 2C205E 170 2054 JLT RLOOP 38203F 175 2057 EXIT STX LENGTH 101036 180 205A RSUB 4C0000 185 205D INPUT BYTE X'F1' F1 190 205E MAXLEN WORD 4096 001000				SUBROU	TINE TO READ R	ECORD INTO BUFFER
125 2039 RDREC LDX ZERO 041030 130 203C LDA ZERO 001030 135 203F RLOOP TD INPUT E0205D 140 2042 JEQ RLOOP 30203F 145 2045 RD INPUT D8205D 150 2048 COMP ZERO 281030 155 204B JEQ EXIT 302057 160 204E STCH BUFFER, X 549039 165 2051 TIX MAXLEN 2C205E 170 2054 JLT RLOOP 38203F 175 2057 EXIT STX LENGTH 101036 180 205A RSUB 4C0000 185 205D INPUT BYTE X'F1' F1 190 205E MAXLEN WORD 4096 001000			TENERS IN AND		No has according to see	
130 203C	SANSER STATE OF THE SANSER	2039	RDREC	LDX	ZERO	041030
135	CONTRACTOR .					
140 2042 JEQ RLOOP 30203F 145 2045 RD INPUT D8205D 150 2048 COMP ZERO 281030 155 204B JEQ EXIT 302057 160 204E STCH BUFFER, X 549039 165 2051 TIX MAXLEN 2C205E 170 2054 JLT RLOOP 38203F 175 2057 EXIT STX LENGTH 101036 180 205A RSUB 4C0000 185 205D INPUT BYTE X'F1' F1 190 205E MAXLEN WORD 4096 001000			RLOOP			E0205D
145 2045 RD INPUT D8205D 150 2048 COMP ZERO 281030 155 204B JEQ EXIT 302057 160 204E STCH BUFFER, X 549039 165 2051 TIX MAXLEN 2C205E 170 2054 JLT RLOOP 38203F 175 2057 EXIT STX LENGTH 101036 180 205A RSUB 4C00000 185 205D INPUT BYTE X'F1' F1 190 205E MAXLEN WORD 4096 001000						
150 2048 COMP ZERO 281030 155 204B JEQ EXIT 302057 160 204E STCH BUFFER,X 549039 165 2051 TIX MAXLEN 2C205E 170 2054 JLT RLOOP 38203F 175 2057 EXIT STX LENGTH 101036 180 205A RSUB 185 205D INPUT BYTE X'F1' F1 190 205E MAXLEN WORD 4096 001000						D8205D
155 204B JEQ EXIT 302057 160 204E STCH BUFFER, X 549039 165 2051 TIX MAXLEN 2C205E 170 2054 JLT RLOOP 38203F 175 2057 EXIT STX LENGTH 101036 180 205A RSUB 4C0000 185 205D INPUT BYTE X'F1' F1 190 205E MAXLEN WORD 4096 001000						281030
160 204E STCH BUFFER, X 549039 165 2051 TIX MAXLEN 2C205E 170 2054 JLT RLOOP 38203F 175 2057 EXIT STX LENGTH 101036 180 205A RSUB 4C0000 185 205D INPUT BYTE X'F1' F1 190 205E MAXLEN WORD 4096 001000					EXIT	302057
165 2051 TIX MAXLEN 2C205E 170 2054 JLT RLOOP 38203F 175 2057 EXIT STX LENGTH 101036 180 205A RSUB 4C0000 185 205D INPUT BYTE X'F1' F1 190 205E MAXLEN WORD 4096 001000	CONTRACTOR OF CO					549039
170 2054 JLT RLOOP 38203F 175 2057 EXIT STX LENGTH 101036 180 205A RSUB 4C0000 185 205D INPUT BYTE X'F1' F1 190 205E MAXLEN WORD 4096 001000	200000000000000000000000000000000000000					2C205E
175 2057 EXIT STX LENGTH 101036 180 205A RSUB 4C0000 185 205D INPUT BYTE X'F1' F1 190 205E MAXLEN WORD 4096 001000					RLOOP	38203F
180 205A RSUB 4C0000 185 205D INPUT BYTE X'F1' F1 190 205E MAXLEN WORD 4096 001000	Contractor.		EXIT		LENGTH	101036
185 205D INPUT BYTE X'F1' F1 190 205E MAXLEN WORD 4096 001000				RSUB		4C0000
190 205E MAXLEN WORD 4096 001000			INPUT	BYTE	X'F1'	F1
OS OF COMMENT OF COMME		205E	MAXLEN	WORD	4096	001000
200 : SUBROUTINE TO WRITE RECORD FROM BUFFER				SUBBOU	TINE TO WRITE	RECORD FROM BUFFER
205			SET OF FRANCE SEE	DODICOG		
210 2061 WRREC LDX ZERO 041030		2061	WRREC	LDX	ZERO	041030
215 2064 WLOOP TD OUTPUT E02079						
220 2067 JEQ WLOOP 302064						
225 206A LDCH BUFFER, X 509039						
230 206D WD OUTPUT DC2079						
235 2070 TIX LENGTH 2C1036						
240 2073 JLT WLOOP 382064						
245 2076 RSUB 4C0000						
250 2079 OUTPUT BYTE X'05' 05			OUTPUT		x'05'	
255 END FIRST		2015	COTFOI			

Figure 2.2 Program from Fig. 2.1 with object code.

```
H | COPY | 001000 | 00107A

T | 001000 | 1E | 141033 | 482039 | 001036 | ...

T | 00101E | 15 | 0C1036 | 482061 | 081033 | ...

...

T | 002073 | 07 | 382064 | 4C0000 | 05

E | 001000
```

Header record:	
Col. 1	Н
Col. 2-7	Program name
Col. 8-13	Starting address of object program (hexadecimal)
Col. 14-19	Length of object program in bytes (hexadecimal)
Text record:	
Col. 1	T
Col. 2-7	Starting address for object code in this record (hexadecimal)
Col. 8-9	Length of object code in this record in bytes (hexadecimal)
Col. 10 – 69	Object code, represented in hexadecimal. (69-10+1)/6=10 instructions
End record:	
Col. 1	E
Col. 2-7	Address of first executable instruction in object program (hexadecimal)

Col. 1 T

Col. 2–7 Starting address for object code in this record (hexadecimal)

Col. 8–9 Length of object code in this record in bytes (hexadecimal)

Col. 10–69 Object code, represented in hexadecimal (2 columns per byte of object code)

End record:

Col. 1 E

Col. 2–7 Address of first executable instruction in object program (hexadecimal)

To avoid confusion, we have used the term *column* rather than *byte* to refer to positions within object program records. This is not meant to imply the use of any particular medium for the object program.

Figure 2.3 shows the object program corresponding to Fig. 2.2, using this format. In this figure, and in the other object programs we display, the symbol ^ is used to separate fields visually. Of course, such symbols are not present in the actual object program. Note that there is no object code corresponding to addresses 1033–2038. This storage is simply reserved by the loader for use by the program during execution. (Chapter 3 contains a detailed discussion of the operation of the loader.)

We can now give a general description of the functions of the two passes of our simple assembler.

Pass 1 (define symbols):

- 1. Assign addresses to all statements in the program.
- 2. Save the values (addresses) assigned to all labels for use in Pass 2.

Figure 2.3 Object program corresponding to Fig. 2.2.

PASS -1

```
begin

read first input line

if OPCODE = 'START' then

begin

save #[OPERAND] as starting address

initialized LOCCTR to starting address

write line to intermediate file

read next input line

end {if START}

else

initialized LOCCTR to 0
```

Loc Source Statement

1000 COPY OUTPUT	START	1000	COPY FILE FROM INPUT TO
1000 FIRST	STL	RETADR	SAVE RETURN ADDRESS
1003 CLOOP	JSUB	RDREC	READ INPUT RECORD
1006	LDA	LENGTH	TEST FOR EOF (LENGTH = 0)
1009	COMP	ZERO	
100C	JEQ	ENDFIL	EXIT IF EOF FOUND
1000F	JSUB	WRREC	WRITE OUTPUT RECORD
1012	J	CLOOP	LOOP
1015 ENDFIL	LDA	EOF	INSERT END OF FILE MARKER
1018	STA	BUFFER	
101B	LDA	THREE	SET LENGTH = 3
101E	STA	LENGTH	
1021	JSUB	WRREC	WRITE EOF
1024	LDL	RETADR	GET RETURN ADDRESS
1027	RSUB		RETURN TO CALLER
102A EOF		BYTE	C'EOF'
102D THREE	WORD	3	
1030 ZERO		WORD	0
1033 RETADR	RESW	1	

while OPCODE != 'END' do	1000 COPY	START	1000	COPY FILE F	FROM INPUT TO	
begin	OUTPUT	CTTY.	DET 1 DD		DI ADDDEGG	
if this is not a comment line then	1000 FIRST	STL	RETADR	SAVE RETUI	RN ADDRESS	
begin	1002 CLOOD	JSUB	DDDEC	DEAD INDUT	PRECORD	
if there is a symbol in the LABEL field then	1003 CLOOP	JSUB	RDREC	READ INPUT	RECORD	
begin	1006	LDA	LENGTH	TEST EOD E	OF (LENGTH = 0)	
search SYMTAB for LABEL	1000	LDA	LENGIII	TEST FOR E	JI (LENGIII - 0)	
if found then	1009	COMP	ZERO			
set error flag (duplicate symbol)	1005	COM	ZERO			
else	100C	JEQ	ENDFIL	EXIT IF EOF	FOUND	
insert (LABEL, LOCCTR) into SYMTAB						
end {if symbol}	1000F	JSUB	WRREC	WRITE OUT	PUT RECORD	
search OPTAB for OPCODE						
if found then	1012	J	CLOOP	LOOP		
add 3 {instruction lengh} to LOCCTR						
else if OPCODE = 'WORD' then	1015 ENDFIL	LDA	EOF	INSERT END	OF FILE MARKER	
add 3 to LOCCTR						
else if OPCODE = 'RESW' then	1018	STA	BUFFER			
add 3 * #[OPERAND] to LOCCTR						
else if OPCODE = 'RESB' then	101B	LDA	THREE	SET LENGTH	H = 3	
add #[OPERAND] to LOCCTR	1015	CTD A	LENGTH			
else if OPCODE = 'BYTE' then	101E	STA	LENGTH			
begin	1021	JSUB	WRREC	WRITE EOF		
find length of constant in bytes	1021	JSUB	WKKEC	WRITE EUF	COPY	1000
add length to LOCCTR	1024	LDL	RETADR	GET RETUR	FIRST	1000
end {if BYTE}	1024	LDL	KEIMDK	GET RETOR	CLOOP	1003
else	1027	RSUB		RETURN TO	ENDFIL	1015
set error flag (invalid operation code)					EOF	1024
end {if not a comment}	102A EOF		BYTE	C'EOF'	THREE	102D
write line to intermediate file					ZERO	1030
read next input line	102D THREE	WORD	3		RETADR	1033
end {while not END}					LENGTH	1036
write last line to intermediate file	1030 ZERO		WORD	0		
save (LOCCTR - starting address) as program length					BUFFER	1039
end	1033 RETADR	RESW	1		RDREC	2039
	100C I ENICELI	DECLA		LENGTHO		
	1036 LENGTH	RESW	I	LENGTH OF		

PASS -2

begin

read first input file {from intermediate file}

if OPCODE = 'START' then

begin

write listing line

read next input line

end {if START}

write header record to object program

initialized first Text record

Loc	S	ource Statem	ent	(Object Code
1000 COPY	START	1000	COPY FILE F	ROM INPUT T	O OUTPUT
1000 FIRST 141033	STL	RETADR	SAVE RETUR	RN ADDRESS	
1003 CLOOP 482039	JSUB	RDREC	READ INPUT	RECORD	
1006 001036	LDA	LENGTH	TEST FOR EC	OF (LENGTH =	0)
1009 281030	COMP	ZERO			
100C 301015	JEQ	ENDFIL	EXIT IF EOF	FOUND	
1000F 482061	JSUB	WRREC	WRITE OUTP	UT RECORD	
1012 3C1003	J	CLOOP	LOOP		
1015 ENDFIL 00102A	LDA	EOF	INSERT END	OF FILE MAR	KER
1018 0C1039	STA	BUFFER			
101B 00102D	LDA	THREE	SET LENGTH	[= 3	
101E 0C1036	STA	LENGTH			
1021 482061	JSUB	WRREC	WRITE EOF	COPY FIRST	1000 1000
1024 081033	LDL	RETADR	GET RETUR	CLOOP	1003
1027 4C000	RSUB		RETURN TO	ENDFIL EOF	1015 1024
102A EOF 454F46		BYTE	C'EOF'	THREE ZERO	102D 1030
102D THREE 000003	WORD	3		RETADR	1033
1030 ZERO 000000		WORD	0	LENGTH BUFFER	1036 1039
1033 RETADR	RESW	1		RDREC	2039
1036 LENGTH	RESW	1	LENGTH OF		

while OPCODE != 'END' do	1000 COPY 1000 FIRST	START STL	1000 RETADR	COPY FILE FROM INPUT TO OUTPUT SAVE RETURN ADDRESS
begin	141033	SIL	RETTIBLE	
if this is not a comment line then	1003 CLOOP	JSUB	RDREC	READ INPUT RECORD
begin	482039			
search OPTAB for OPCODE	1006	LDA	LENGTH	TEST FOR EOF (LENGTH = 0)
if found then	001036	COMP	ZEDO	
begin	1009 281030	COMP	ZERO	
if there is a symbol in OPERAND field then	100C	JEQ	ENDFIL	EXIT IF EOF FOUND
begin	301015	JLQ	ENDITE	EMIT II EOI I GOND
search SYMTAB for OPERAND	1000F	JSUB	WRREC	WRITE OUTPUT RECORD
if found then	482061			
store symbol value as operand address	1012	J	CLOOP	LOOP
else	3C1003			
begin	1015 ENDFIL	LDA	EOF	INSERT END OF FILE MARKER
store 0 as operand address	00102A	OT 4	DUPPP	
set error flag (undefined symbol)	1018 0C1039	STA	BUFFER	
end	101B	LDA	THREE	SET LENGTH = 3
end {if symbol}	00102D	LDII	TITICLE	SET ELIVOTTI S
else	101E	STA	LENGTH	
store 0 as operand address	0C1036			
assemble the object code instruction	1021	JSUB	WRREC	WRITE EOF
end {if opcode found}	482061			
else if OPCODE = 'BYTE' or 'WORD' then	1024	LDL	RETADR	GET RETURI COPY 1000
convert constant to object code if object code not fit into the current Text record then	081033	DCLID		FIRST 1000
•	1027 4C000	RSUB		RETURN TO CLOOP 1003
begin write Text record to object program	102A EOF		BYTE	C'EOF' ENDFIL 1015
initialized new Text record	454F46		BIIL	EOF 1024
end	102D THREE	WORD	3	THREE 102D
add object code to Text record	000003			ZERO 1030
end {if not comment}	1030 ZERO		WORD	0 RETADR 1033
write listing line	000000			LENGTH 1036
read next input line	1033 RETADR	RESW	1	RUFFER 1030
end {while not END}	1036 LENGTH	RESW	1	LENGIH OF DNDEC 2020
ena (mine not 2112)	1039 BUFFER	RESB	4096	4096-BYTE 1 ADREC 2039

write last Text record to object program
write End record to object program
write last listing line
end

1000 COPY	START	1000	COPY FILE FROM INPUT TO OUTPUT
1000 FIRST 141033	STL	RETADR	SAVE RETURN ADDRESS
1003 CLOOP 482039	JSUB	RDREC	READ INPUT RECORD
1006 001036	LDA	LENGTH	TEST FOR EOF (LENGTH = 0)
1009 281030	COMP	ZERO	
100C 301015	JEQ	ENDFIL	EXIT IF EOF FOUND
1000F 482061	JSUB	WRREC	WRITE OUTPUT RECORD
1012 3C1003	J	CLOOP	LOOP
1015 ENDFIL 00102A	LDA	EOF	INSERT END OF FILE MARKER
1018 0C1039	STA	BUFFER	
101B 00102D	LDA	THREE	SET LENGTH = 3
101E 0C1036	STA	LENGTH	
1021 482061	JSUB	WRREC	WRITE EOF
1024 081033	LDL	RETADR	GET RETURN ADDRESS
1027 4C000	RSUB		RETURN TO CALLER
102A EOF 454F46		BYTE	C'EOF'
102D THREE 000003	WORD	3	
1030 ZERO 000000		WORD	0
1033 RETADR	RESW	1	
1036 LENGTH	RESW	1	LENGTH OF RECORD
1039 BUFFER	RESB	4096	4096-BYTE BUFFER AREA
•			

```
H | COPY | 001000 | 00107A

T | 001000 | 1E | 141033 | 482039 | 001036 | ...

T | 00101E | 15 | 0C1036 | 482061 | 081033 | ...

...

T | 002073 | 07 | 382064 | 4C0000 | 05

E | 001000
```

•	•			-								1	
н	4	Δ	2	а	e	r ·	r	2	~	n	r	П	•
1	1	L	и	u	C.	L.	L١	_\	_1	J	Ľ	ч	

Col. 1 H

Col. 2-7 Program name

Col. 8-13 Starting address of object program (hexadecimal)
Col. 14-19 Length of object program in bytes (hexadecimal)

Text record:

Col. 1 T

Col. 2-7 Starting address for object code in this record (hexadecimal)
Col. 8-9 Length of object code in this record in bytes (hexadecimal)

Col. 10-69 Object code, represented in hexadecimal. (69-10+1)/6=10 instructions

End record:

Col. 1 E

Col. 2-7 Address of first executable instruction in object program (hexadecimal)