LOW SPEED ASSEMBLER

LOW SPEED ASSEMBLER

GENERAL INFORMATION

The Low Speed Assembler is a two pass assembler which reads a paper tape source program and generates a binary coded paper tape object program. It may also be used to generate several printed tables.

It is now programmed to read input from a teletype tape reader and print or punch tape on a teletype. However, it would be easy to change it to use other input or output devices.

Two characters, rubout=177 or 377 and SHIFT-CONTROL -P =000 or 200, are ignored wherever they appear on the source tape.

The object tape is in a format compatible with our TTY bootstrap loader program.

There is no limit to the number of characters on a line.

However, there is a limit to most of the tables stored by the assembler:

macro table	$\leq$259 total instructions
macro call sequence	$\leq$ 64 references
literals	$\leq$192 literals
errors	$\leq$ 96 errors
saves	$\leq$192 saves
duplications	$\leq$ 63 instructions to duplicate

If these limits are exceeded the tables will run over into each other.

```
          I
          or
SYMBOL    D   OPCODE ADDRESS/COMMENT  CR-LF
```

SYMBOL

A letter or a letter and an octal digit 1-7.
May be replaced by a space.
Must not be preceded by a space.

I or D

I for indirect addressing
D is a display opcode follows
May be omitted

OPCODE

A three letter code for an instruction or a pseudo opcode

ADDRESS

May be:

an octal no.	375
a symbol	A1
a pt. relative address	.-15
a symbol relative address	A3+14
a literal	=321
	=< JMS B7-2
	=#SAV
a save	#XAC

omitted if instruction does not require an address
(Address field ends with first space after useful information)

COMMENTS

Must be preceded by a space if there is an address field.
Need not be preceded by a slash but are neater looking if they are.
May use entire line if slash is first character on line.
Are terminated by a line feed.

LITERALS

A literal is used to introduce an octal constant, an address constant, an instruction constant, or a save address into a program without the bother of labeling it and entering it separately into the source program. Just write the octal constant, address constant, "< instruction" constant, or save, preceded by an equal sign, in the address field(s) of the statement(s) in which it is used. The assembler places all literals after the saves of the program and gives a listing of these addresses and the literals assigned to them.

```
AND =177        /mask rt. seven bits
ADD =A3-5       /ADD A3-5 to AC
LAC =< D JMP C  /load the instruction "D JMP C" into the AC
LAC =#SAV       /load the address of #SAV into the AC
```

SAVES

A save is similar to a literal in that it is a way of introducing a saved memory cell into a program without labeling it and entering it separately into the source program. It is useful for reserving memory cells for counters and other variables. Just assign the variable a 3 letter code, and write that code, preceded by a #, in the address field of the statement(s) in which it is used. The assembler places all saves immediately after the last source statement of the program.

```
DAC   #XAC   /DAC in word called XAC
```

PSEUDOS

ORG 21	The following statements start at location 21. (used at start of source program)
REL A1+2	The following statements start at location A1+2 (used anywhere in program)
BSS 5	Reserve the 5 following memory words.
REP 12	Repeat the previous instruction 12 times.
ZRO A1+6	Place the address A1+6 in this location.
OCT 175462	Place the octal No. (175462) in this location.
A2 EQU C7-3	Set the symbolic address A2 equal to the address C7-3.
END	This is the end of the source program.
DUP 3 6	Duplicate the following 3 statements 6 times.

MCD A 14	Use the following 14 statements to define the macro (A).
MCE	This is the end of the macro definition.
MCC A 14 B2 177 X =37 < JMS A2	/Call 14 step macro, A
	Insert the 14 step macro A using B2 as the address for @1, 177 for @2, X for @3, the address of the literal =37 for @4, and the instruction (JMS A2) following < in place of INS @5.
INS @7	Reserve this instruction position in the macro we are now defining for an instruction to be specified in the 7th information position in the call for this macro.
INC E B3-2 /	Interpret (E B3-2) as two increment mode display bytes.

INCREMENT MODE

/	This is the end of the increment mode word / necessary
B	Turn the beam on for the following increment bytes.
D	Turn the beam off for the following increment bytes.
E	Enter increment mode.
N	Exit increment mode, zero X and Y LSB. (111)
R	Exit increment mode, return from D JMS, zero X and Y LSB. (151)
F	Exit increment mode, return from D JMS, add one to X MSB, zero X and Y LSB. (171)
P	Pause. (200)
A 273	Make this byte 273.
space	Ignore.
+	Form byte.
-	Form byte.
0 1 2 3	Form byte.

MACRO INSTRUCTIONS

The Low Speed Assembler is capable of storing 26 programmer defined macros of not more than 259 total instructions. In defining a macro one can use constant instructions, instructions with point relative addresses, instructions to be specified in the macro call, and instructions with addresses to be specified in the macro call.

Each of these macros can then be called as often as necessary.

On a macro call a slash is needed after all the variable information is specified. If the call sequence takes up more than one line of characters, type SHIFT-N CR-LF, and continue on the next line.

Examples of macro calls and definitions are included with this write up.

PRINTOUT

The Low Speed Assembler does not list the source tape or produce an object listing. A source listing can be obtained by running the teletype on local, and an object listing can be obtained by using one of several listing programs.

The Low Speed Assembler does, however, list several tables at the end of the first pass and also if you push CONTINUE at the end of the second pass.

SYMBOL TABLE

Omitted if DS bit 15 down
A list of the defined symbols and the addresses assigned to them.

UNREFERENCED SYMBOL TABLE

Omitted if DS bit 14 down
A list of the defined but unreferenced symbols and their addresses

ERROR TABLE

Omitted if DS bit 13 down
A list of the addresses at which errors took place.

SAVE TABLE

Omitted if DS bit 12 down
A list of the saved addresses and their corresponding three letter codes.

LITERAL TABLE

Omitted if DS bit 11 down
A list of the addresses assigned to literals and the corresponding constants.

MACRO TABLE

Omitted if DS bit 10 down
A list of the defined macros, the instructions in each, and information on how to interpret the calling sequence for the particular macro.

ORIGIN

Omitted if DS bit 9 down
The address of the origin of the program.

END

Omitted if DS bit 9 down
The address of the last source statement.

LAST

Omitted if DS bit 9 down
The address of the last save, literal, or source statement.

ERRORS

The Low Speed Assembler recognizes several kinds of errors and lists the addresses at which these errors occurred at the end of pass 1, and pass 2 if you push CONTINUE*. However, the second pass may be made even if errors were found on the first pass; NOP's will be assembled wherever the source tape has an error.

Usually errors are obvious, but here are some that are not:

Referencing an undefined symbol - (This type of error will only show up on second pass symbol table)

Not putting D before a display command

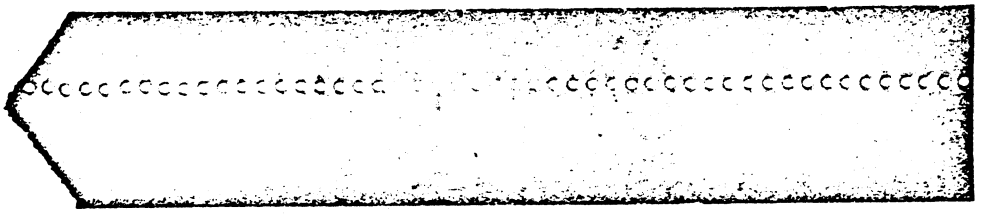
Having non-printing control characters on the source tape

Trying to define a symbol twice

*It is a good idea to get a table printout after the second pass as some errors are not recognized until the second pass and the origin, end, last addresses give a good indication as to whether or not your program was interpreted the same on both passes.

SEPERATE TAPES

If the source tape is in several sections, square off the ends and run them through separately. Push start for the first section and continue for each successive section.



TO USE:--- Load Low Speed Assembler with TTY bootstrap in 40.

Load source tape in TTY

Switch TTY to on line

Start computer at 100

Start TTY reader

When computer encounters END instruction it will type out all tables (DEPENDING ON DS settings).

Reload source tape in TTY

Turn punch on (leader will be punched)

Push CONTINUE for pass two

Start TTY reader when preceder is punched

Turn punch off

Push CONTINUE for another set of tables.