INTRODUCTION TO THE MTX EDITOR/ASSEMBLER

The basic function of an editor or text editor is to aid the process of creating sections of assembly language coding. Therefore, an editor must allow:

(i) The initial insertion of text into the 'file'
(ii) Editing of the file to delete errors and sections of redundant code or to add new instructions
(iii) Listing of the file as a visual check (and for later reference) either to a display or to a printer.

Having created the file we then want to create the machine code equivalent of the assembly language mnemonics, this is usually done by a program called an assembler.

The MTX series of microcomputers have a program resident in ROM which gives both editing and assembly facilities.

In BASIC line numbers are normally used to denote the relative position of that line of coding in the program, the assembly language facilities offered by the MTX computers is designed to complement the BASIC interpreter. As such it is designed such that sections of assembly language coding can be embedded inside a BASIC program by giving the start of each <u>section</u> of code a line number.

<u>Editor/Assembler Tutorial</u>

<u>Invoking the Assembler/Editor</u>

In order to use the assembler/editor facilities you must first tell the computer that you wish to have access. To do this you simply type

```
ASSEM {space} [Line Number]       <RET >
```

```
e.g. ASSEM 1 <RET>
or
ASSEM 10     <RET>
```

This will cause the assembly language code generated to be associated with the BASIC line number specified.

Type ASSEM 10 < RET >

“Assemble” will appear at the bottom of the screen, this specifies that you are in the editor/assembler mode.

Type <RET > again

The screen will now look like :

```
4007          RET
Insert
```

INSERT tells you that you are in the insert mode i.e. if you type assembly language mnemonics they will be inserted without destroying anything already there.

NB 4007 is the hexadecimal address after which the machine code will be inserted.

RET is the instruction to RETURN to the calling routine (in this case BASIC).

If you now type.

```
START:LD A,#AF     ; THE START           < RET>
```

This will create a LABEL, START which may be used instead of using a hexadecimal address for call and jump instructions.

We have also defined an instruction which will move the hexadecimal number AF to the A register (often called the accumulator).

NB The # symbol is obtained by using SHIFT and 3 - as if you wished the £ symbol to be printed, this symbol specifies that the number following is a hex number. The semicolon ; is used to denote that any symbols to the right are to be treated as comments only. They are for the program reader only and play no part in the operation of the program.

Having pressed the <RET>key the display will be

```
4009          RET
```

Insert

Which tells you that the next line of code will be inserted at location 4009.

If you type LOOP:INC A

The screen will now look like

```
            4009          LOOP : INC AT
Insert
```

Pressing <RET> at this point will cause the messsage 'bad code' to appear. The fault is the RET instruction left at the end of the line. This can be removed by the use of the control keypad. The keys marked ⇨ and ⇦ allow movement of the cursor in either direction along the line. By moving the cursor right, to the point just before the RET instruction and then pressing the key labelled <DEL> we can delete the characters one by one. Alternatively by moving to the same place and then pressing the key marked <EOL>any characters to the right of the cursor will be deleted.

You can now press <RET> again and the current line will be accepted and a new display produced.

```
                400A   RET
Insert
```

Now use the above technique to insert the rest of these instructions

```
LD B,A
LD C,B
LD D,C
LD E,D
JP LOOP
```

Having entered the above instructions then press <CLS> followed by <RET>

The system will respond by returning you to the assembler and the screen shows :

```
                Assemble
```

as before

To list your program you have to move the 'program pointer' to the top of the program. This is done by typing

```
                T        < RET>
```

To list the program then type

```
                L        < RET>
```

The program pointer remembers the last position you inserted code at and the program will be listed to that point.

The screen will look like :

```
4007   START :     LD A,#FA      ;THE START
4009   LOOP :      INC A
400A               LD B,A
400B               LD C,B
400C               LD D,C
400D               LD E,D
400E               JP LOOP
4011               RET

Symbols :

START       4007        LOOP        4009
```

Defining the contents of bytes

It is possible to pre-define what the contents of certain bytes will be. This can be achieved using the DEFINE BYTE command DB.

```
e.g. 4008 DB   1, 2, "ABCD"
```

Bytes can be defined as a list of numbers, decimal or hex
or by enclosing characters within “   “

Defining Areas of Storage

It is possible also to allocate areas of memory for use as data storage. This can be achieved by the DEFINE STORAGE command DS

```
e.g     4008    DS      254     This allocates 254 bytes of storage,
        4106                    the next free location is 4106
```

As much storage as you wish may be set aside as long as each block you create is 254 bytes long or less. It is however, possible to concatenate blocks as shown below :

```
4008    DS      254     An area of storage 1016 bytes long
4106    DS      254
4202    DS      254
4302    DS      254
4400    RET             Next free location
```

When you exit from the assembler all the code is assembled and all addresses are calculated. You will now be in BASIC and it is possible that you may edit lines with lower line numbers than that into which you have assembled the machine code. If this is done it is important to enter the assembler and exit again, this will reassemble each address and its code again.

On exit from the assembler the code you have produced is able to be listed from BASIC. If you list our initial program you will see

```
                10 CODE

                4007    START :         LD A,#FA        ;THE START
                4009    LOOP :          INC A
                400A                    LD B,A
                400B                    LD C,B
                400C                    LD D,C
                400D                    LD E,D
                400E                    JP LOOP
                4011                    RET
```

```
Symbols

START           4007            LOOP 4009
```

SUMMARY

The assembler is invoked by typing ASSEM. Line Number < RET>

To return to BASIC type <CLS> followed by <RET>

To insert code enter the assembler and press <RET>
and to stop inserting < CLS> followed by < RET>

To list your code whilst in the assembler type T <RET>
followed by L < RET>

Insert Mode

In the insert mode, any lines typed into the computer will be inserted at the address shown on the left of the screen. The correct amount of space in memory will be made for each line as it is entered.

These are four ways of entering the insert mode.

(i) <RET> enters at the current program pointer position
(ii) #n enters at the Hex address 'n'
(iii) n   enters at the decimal address 'n'
(iv) Label enters at the label if its exists

To exit from insert mode type <CLS> <RET>

Edit Mode

In the edit mode each line entered replaces the line originally displayed. In this way it differs from the insert mode where lines are inserted without altering what is already there.

As with the insert mode there are four ways of entering the edit mode.

(i)	E <RET>	enters the editor at the program pointer
(ii)	E #n	enters the editor at the Hex address 'n'
(iii)	E n	enters the editor at the decimal address 'n'
(iv)	E label	enters the editor at the label specified if it exists

If a label E exists then if E < RET> is typed the insert mode is entered at label E rather than the editor at the program pointer.

List
(i) L <RET> lists the program from the program pointer.
(ii) L #n lists the program from the HEX address 'n'
(iii)L n   lists the program from the decimal address 'n'
(iv) L label lists the program from the label if it exists.

NB As with the edit mode a label L will lead to L <RET> entering the insert mode at label L instead of listing from the current program pointer.

Delete

Lines can be deleted either in the edit or insert modes. When a line is displayed the cursor appears between the address and the code. If you press <EOL>, or type spaces over the code and press <RET> the line will be deleted.

Labels

Address labels may be used as points in the program for CALL or JUMP instructions to branch to if the label is followed by a colon.

e. g.

```
4009 LABEL:        LD A,B
                   JP LABEL
```

Comments

Comments may be written after any instruction by preceding the comment with a semi-colon.
e. g.

END OF PROGRAM

<u>THE MTX FRONT PANEL DISPLAY</u>

The front panel display is provided to give you the ability to test and debug programs written in assembly language.

To invoke the front panel display from BASIC, type

```
PANEL     <RET>
```

On entry to the FRONT PANEL the Z80 registers are displayed on the right hand side of the screen and a block of memory has its contents displayed at the bottom of the screen.
Various keys activate other functions in the FRONT PANEL and these are defined below

Key	Prompt or response expected	Effect
Basic	Exit?	Answer Y to return to BASIC.
Clear		Clears the list screen.
Display	Hex No.	Display a block of memory around the address Hex. No.
D.		Displays a block of memory around the byte specified by the PROGRAM COUNTER (PC)
Go	Hex 1 to Hex 2	Run a program starting at Hex 1 up to Hex 2.
I		Change memory display from Hex to ASCII or vice versa
List	Hex No.	List [Dis-assemble] from the address Hex No.
L.		Lists a block of memory around the byte specified by the PROGRAM COUNTER (PC)
Move	Hex 1 End Hex 2 to Hex 3	Moves a block diagram of memory from starting address Hex 1 to end address Hex 2 to the block whose starting address is Hex 3
Register	Hex	Change register at cursor position to the hex value given
Single step		Execute the command at the current PROGRAM COUNTER Position.
Trace		As with single step but it treats a CALL to a sub-routine as a single step.
eXchange		Display the alternate register set.
.		Move the register cursor
_		Move the display cursor back one position
<RET>		Move the display cursor forward one position
↑		Move the display cursor up one line
↓		Move the display cursor down one line