GENERAL VIDEO ASSEMBLER

Forward: The user must read this manual thoroughly. But if you are anxious to get the action going, go straight to Appendix F for a walk-through of the assembly process.

THE GENERAL VIDEO ASSEMBLER

1. GENERAL

Provided in this manual is a comprehensive description of the General Video Assembler for the Z80-based Astrocade.

The General Video Assembler includes an editor for preparation of the source program, Pass I and Pass II which translate Z80 source statements into hexadecimal, and the Collector for joining together multiple segments.

An assembler is different from a translator. Your Astrocade Basic is a translator. It can execute a Basic program directly by a RUN command, translating each Basic statement before executing it. Consequently a Basic translator is very convenient but also very slow in execution since every statment must be translated before execution.

With an assembler, the source program cannot simply be RUN. It must first be assembled. This is a one-time process, converting source statements to hexadecimal, and is completely finished by the time the object program is CALLed. A change in the source program requires a new assembly. Since no translation goes on during execution, the speed is greatly increased.

It is not the purpose of this manual to teach Z80 machine coding. If you have experience with other assembler languages, and understand hexadecimal, and need only a familiarity with the processor architecture and instructions, then you can get by with the Z80 Instruction Handbook by Nat Wadsworth (SCELBI Publications, 1978, about $6). Otherwise, get a more comprehensive textbook, like Programming the Z80 by Rodney Zaks (Sybex, 1980, about $12). You can get the former from most computer stores, and the latter from most Radio Shacks.

2. HARDWARE REQUIRED

An Astrocade game computer, Astrocade Basic with taping facilities, and add-on memory is required. The add-on memory must be at least 4K (hex) in size and it must be switchable from the 2K (hex) to the 6K (hex) address ranges (like the Blue RAM or Viper).

Since there is no floppy disk capability, all intermediate files must be recorded on tape. Thus tape handling can be excessive at times, in spite of operational streamlining. This is true in particular of multi-segment programs. To facilitate this, the user is strongly urged to install an I/O switch for switching between input and output taping. Steve Walters explained how to do this in the "Arcadian" of Dec. 7, 1981 (volume 4, number 2, page 16).

3. SYNTAX

A Z80 assembly language program (source program) includes labels, opcodes, operands, comments, e.g.:

```
LAB01.LD.(HL),A  ;THIS IS A SAMPLE STATEMENT
```

3.1. Delimiters

Delimiters used in assembler language statements are as follows:

- A line of code must begin with a statment number (for editing purposes) and must contain at most one instruction.

- An opcode must have a period immediately before it, and immediately after it. Spaces may be included ahead of the preceeding period and after the following period.

- Operands must be separated from each other by a comma.

- A comment should be preceded by a semicolon. If the comment is not the only element in the line, the semicolon should be preceded by a space.

3.2 Labels

Label syntax is as follows:

- A label is composed of from one to six characters. If more than six characters are used, the assembler recognizes only the first six.

- A label must start in the first position following the line number.

- A label may be composed of the following characters: A-Z # @ 0-9

- The first character of a label must not be a digit.

- A single pound sign followed by a single letter must not be used as a label, since it is used to specify a Basic variable.

3.3 Opcodes

There are 74 generic opcodes (such as LD) and 693 legitimate combinations of opcodes and operands in the Z80 instruction set. These instructions are fully documented in the publications mentioned in section 1 above.

Sixteen of the opcodes (RST EX INC DEC PUSH POP AND OR XOR ADD CALL RET JP JR LD and CP) can be typed with a single keystroke as shown on the General Video Keypad overlay. (They may optionally be typed with individual letters). When the single keystroke method is used, the preceding and following period (3.1 above) is provided.

3.4 Pseudo-ops

The following pseudo-ops are recognized by the assembler:

.DB.n - Define the contents of a byte located at the current reference counter to be n. If n is not specified, a space will be generated.

.DW.nn - Define the contents of a two-byte word located at the current reference counter to be nn. The least significant byte is located at the current reference counter. The most significant byte is located at the reference counter plus one. If nn is not specified, two spaces will be generated.

Pseudo-ops are assembled exactly like executable instructions. They must have adjacent periods before and after. They may be preceded by a label and followed by a comment. The reference counter corresponds to the relative address of the instruction.

3.5 Operands

One, two or no operands may be included in an assembly language statement depending on the opcode used. These operands may take any of the following forms:

Generic operand

- Constant

- Label

- Expression

The dollar sign symbol ($) is used in operands to represent the value of the reference counter of the current instruction.

When doing relative addressing the current value of the reference counter is subtracted from the label referenced, eg,:

```
.JR.NC,LOOP
```

This statement will cause a jump relative to LOOP. The possible range of a relative jump (JR or DJNZ) is backward 126 bytes or forward 129 bytes.

3.5.1 Generic Operands

Generic operands (such as HL for the HL register pair) are summarized below:

OPERAND	MEANING
A	A register (accumulator)
B	B register
C	C register
D	D register
E	E register
H	H register
L	L register
AF	AF register pair
AF'	AF' register pair
BC	BC register pair
DE	DE register pair
HL	HL register pair
SP	Stack pointer register
$	Reference counter
I	I register (interrupt vector byte)
R	Refresh register
IX	IX index register
IY	IY index register
NZ	Not zero
Z	Zero
NC	Not carry
C	Carry
PE	Parity even or overflow
PO	Parity odd or no overflow
P	Positive sign
M	Negative sign (minus)

The following generic operands may be entered with one keystroke per the General Video Keypad overlay: AF IX IY SP BC DE and HL. (They may optionally be entered as individual letters).

3.5.2 Constant

A constant is a number within the range 0 through 65,535. These constants can be in any of the following forms:

- Decimal - this is the default mode of the assembler and does not require any explicit specification. Eg 2341

  Hexadecimal - must begin with the digits 0-9. Must end with the letter H unless it contains A-F (in which case the H is optional). Eg 0FF or 135H

- ASCII - Characters enclosed in single quote marks will be converted to their ASCII equivalent values by the assembler. EXCEPTION: may not include a space or a comma. Eg 'A' or 'P%'

3.5.3 Label

A label (symbol) used as an operand must be defined elsewhere in the program (see section 3.2)

The following special symbols are provided referring to the Astrovision Basic variables (single letter only):

```
   #A-#Z
```

For example:

```
   .LD.HL,#N   ;PUTS THE ADDRESS OF VARIABLE N IN HL
   .LD.HL,(#P) ;PUTS THE VALUE IN VARIABLE P INTO HL
```

3.5.4 Expressions

The General Video Assembler accepts expressions involving addition and subtraction only. All expressions are evaluated from left to right. Parens, if included are ignored, unless the entire operand is enclosed. If the entire operand is enclosed, the contents of a memory location are specified if appropriate to the syntax. Examples of instructions with expressions:

```
   .JR.Z,LOOP-5
   .LD.HL,100+20-4    ;PUT 116 IN HL
   .LD.HL,(100+20+4)  ;PUT CONTENTS OF MEMORY LOC 116 IN HL
```

3.6 Comments

A comment is defined as any characters following a semicolon in a line. It may stand alone, or follow an instruction. Comments are ignored by the assembler.

4. ERRORS

Errors are identified on the screen when encountered by the assembly process (Pass I or Pass II). After noting the error, non-fatal errors may be circumvented by pressing any key. Assembly will continue.

The occurrence of a non-fatal error in Pass I results in ignoring the statement. However in Pass II, where addresses have already been established, the occurrence of a non-fatal error results in a no-op (zeros) of the equivalent number of bytes.

Error codes are identified in appendix E.

5. OPERATION

Operation of the programs of the General Video Assembler is described in the Appendices. Refer to Appendix F for a walk-through, or to Appendices A-D for specifics on the operation of each program.

When assembly is complete, your object program will have been written on tape. You can put it where you want it ("target address" as specified to Pass I) by loading the tape with the following immediate command:

```
    :INPUT %(nnnnn)
```

Then you can CALL it with an immediate command, or from a Basic program.

Appendix A

EDITOR INSTRUCTIONS

Load the editor with the :INPUT command.

After loading, you should decide whether or not you are modifying an old program, or starting a new one. If you are modifying an old one, load it with the CALLI instruction. Then switch to edit mode by issuing the CALLM instruction and switching RAM from the 6K to the 2K range within 10 seconds. If you are starting a new program, bypass the CALLI and go straight to the CALLM.

When editing, all inputting and outputting of programs is done in Basic mode, and all text editing is done in Editor mode*. Edit text with the General Video Assembler keypad overlay in place. When editing is complete, issue CALLM to return to Basic mode (switch RAM from 2K to 6K position within 10 seconds), and then CALLO to output your program to tape.

CALLM (Mode change) before moving the switch between 2K and 6K (in either direction). After pressing GO, you have 10 seconds of protection against unwanted interrupts that can clobber your text.

CALLI (Input) to input source program from tape. Issue only while in Basic mode.*

CALLO (Output) to output source program to tape. Issue only while in Basic mode.*

RESET (reset button) to erase current program from memory. Screen and variables do not clear. Reset only while in Editor mode* or you will have to reload the editor.

Certain immediate commands will be helpful in Editor mode.* For example LIST (.EX.) and PRINT (.CP.) and CLEAR (.CALL.). These keys have the same function as they do with the Basic keypad overlay, but of course the wrong word will appear on the screen. You can also set NT=n or SM=n in Editor mode. You should not change the value of variables H-Z in either mode.

*note: "mode" is the setting of the RAM's range switch:
2K - Editor mode (General Video Editor in control)
6K - Basic mode (Astrocade Basic in control)

Pass I Instructions

Load Pass I with the :INPUT command and it will start automatically. When you receive the first prompt, "TARGET ADDRESS?", stop the tape and eject it without rewind so that it will be in position to load Pass II.

The first prompt, "TARGET ADDRESS" asks for the address where the object program will normally reside (usually 24576, the start of the add-on RAM). This target address is like the ORG specification in some assemblers.

After entering the target address, you are asked to load segment 1. This is the source program that you prepared with the editor. Do not press GO until the tape is moving at the appropriate place.

If you receive an error message, write it down so that you can go back and correct the source program with the editor. Unless the error is marked "FATAL", you can then continue by pressing any key.

When assembly is done for this segment, you are asked if there is another segment. Answer Y or N for Yes or No. If the answer is Yes, you must then print Pass I output to tape. Do not stop the program to do this. Simply load a work tape, start it moving and press GO, making a note of its location on tape.

If another segment has been specified, you are now asked to load it. The process is repeated.

If there is no second segment, it is not necessary to output segment 1 to tape; it will be held in memory for Pass II.

When Pass I is done, you are asked if you want to view the symbol table. This is the record of all symbols (labels) defined in your program, and where these symbols exist in the object program. If you elect to see the symbol table, step through it using any key to bring up the next screen until you are asked to load Pass II.

When asked to load Pass II, re-insert the General Video Assembler tape into the tape player, press "PLAY" and then "GO".

Appendix C

Pass II Instructions

Pass II can only be loaded under control of Pass I since it presumes that many of the variables of Pass I are present and valid. Thus a reset must not take place between Passes I and II. Further, for a single segment program, the output of Pass I is retained in memory for Pass II and will appear near the middle of the screen during the loading process. In this case, no operator intervention is needed to begin the work of Pass II.

When the first prompt appears (an instruction from Pass II), stop the tape without rewind.

If you receive an error message, write it down so that you can go back and correct the source program with the editor. Unless the error is marked "FATAL", you can then continue by pressing any key.

In the case of multi-segment programs, Pass II will ask that segment 1 be loaded. This is the output from Pass I. Do not press GO until the appropriate portion of tape is moving. After processing, Pass II requests you to output to tape. Press Go after the tape is moving. The process is repeated for each segment. These segments will be requested by the Collector to finish the assembly process.

In a single segment program, assembly is complete when the output of Pass II is on tape.

In a multi-segment program, you are asked to load the Collector when Pass II is done. Again, do not reset the computer. Re-insert the General Video Assembler tape, press "PLAY" and then "GO".

Appendix D

Collector Instructions

The Collector can only be loaded under control of Pass II since it presumes that many of the variables of Pass II are present and valid. Thus a reset must not take place between Pass II and the Collector.

When the first prompt appears (an instruction from the Collector), stop the tape and rewind.

Now the Collector will ask that segment 1 be loaded. This is the output from Pass II. Do not press GO until the appropriate portion of tape is moving. After processing, the Collector requests you to input segment 2. Press Go after the tape is moving. The process is repeated for each segment.

When there are no more segments to be inputted, the Collector asks you to output to tape. This output is the object program. Press GO when the tape is moving at the appropriate place. Assembly is now complete.

Appendix E

ERROR CODES

1 -	Unrecognizable operation code.
2 -	Bad combination of operation code and operands.
3 -	Unrecognizable operand (single operand instruction).
4 -	Unrecognizable operand (double operand instruction).
5 -	Symbol table too big. It has expanded into the area above the 7K boundary. This is acceptable if you have more memory above 7K. The 4K Blue RAM, for example, has another 128 bytes, allowing 16 more symbols. So you may have 15 occurrences of this message before you actually exceed the symbol table capacity. There are 8 bytes per entry in the symbol table. If you exceed the true symbol table capacity, consider using more relative addressing to reduce the number of symbols (labels).
101 -	Op 1 has a numeric item that is too big.
102 -	Op 1 has a symbol not recognized.
103 -	Op 1 expression evaluation results in overflow.
104 -	Op 1 should be one byte - it's too big.
229 -	Op 2 has a numeric item that is too big.
230 -	Op 2 has a symbol not recognized.
231 -	Op 2 expression evaluation results in overflow.
232 -	Op 2 should be one byte - it's too big.
300 -	There is no code to assemble (null input).
302 -	Bad syntax. This message probablyindicates that a memory or I/O error has occurred.

NOTE: If you have tape loading problems, read the entire tape using the following command for each side:

```
   :LIST ;:LIST ;:LIST ;:LIST ;:LIST
```

Watch the screen carefully for the appearance of question marks which indicate tape incompatibility. If this occurs, return the tape with a brief explanation to General Video, 19553 Dartmouth Pl., Northville, Michigan 48167, and a new tape will be provided. The new tape will be created on another recorder.

Appendix F

GETTING STARTED

NOTE: In the following steps, and throughout the manual, tape inputting assumes that you have cabled or switched your tape recorder so that it is inputting to the computer. Outputting assumes that you have depressed the "RECORD" button and have cabled or switched so that you are outputting to tape. If you have tape loading problems, see Appendix E, bottom.

Follow these steps for a quick walk through on the use of the General Video Assembler. One of the sample routines will be assembled.

- Memory should be set to 6K RAM.
- Load the editor with the following command:

  :INPUT

  When the prompt appears, stop the tape, but do not rewind.
- Load the first sample program by issuing the following special command:

  CALLI - but do not press GO until the tape is moving again.
- When the cursor returns, stop the tape and switch to editor mode by issuing the following special command:

  CALLM

  After pressing GO you have 10 seconds to switch your RAM to the 2K position.
- When the cursor reappears, list your sample program by use of the immediate LIST command which now appears on the screen as .EX. but in fact functions like the LIST command.
- The sample on the screen is the short routine which appears on the back of the data sheet. It includes a call to the on-board print subroutine.
- Move back to Basic mode by again issuing the command:

  CALLM

  and within 10 seconds switching the RAM to the 6K position.
- Make a copy of the sample program at the beginning of one of your work tapes by issuing the following command:

  CALLO (alphabetic O)- and press GO after the tape is moving

more . . .

- Load Pass I of the General Video Assembler with the following command:
  ```
  :INPUT
  ```
  and when the prompt appears, stop the tape without rewind.

- The prompt asks you for the starting location of the target code. Reply
  ```
  24576 (GO)
  ```
  This is the load and execute address for the assembled program.

- Next you are asked to load segment 1. This is the sample program that you wrote on your work tape. Press GO when the tape has been inserted and is moving.

- As the tape loads, you will see it on the screen. The lower half of the screen area is used as a work buffer by the assembler. In larger programs you can see the assembly process overlay the inputted text.

- Now you are asked if there is another segment. Enter the letter
  ```
  N (for No)
  ```

- If there were more segments, you would now be asked to output the first segment to tape for intermediate storage. Since there is only one segment, this step is not necessary.

- Pass I now tells you there were no errors, shows you the high address of your object code (24616), and asks if you want to view the symbol table. The symbol table is the record of symbols (labels) you have defined, and where they are located in memory. Respond
  ```
  Y (yes)
  ```

- You now see the two symbols (START and NAME) that were used in the program and their location in memory. The table is displayed in scroll mode 4, so if the table were large you could quickly step through it using any key to bring up the next screen.

- Since Pass I is finished, it requests that you load Pass II. This is done without stopping the program and without resetting. Re-insert the General Video Assembler tape (since it was not re-wound, it should be positioned in front of Pass II). Start the tape and press GO.

- When Pass II is loading, you see the output from Pass I appear in the middle of the screen. In a multi-segment program you would have been asked to load it from tape. When Pass II gives the first instruction, stop and rewind the tape.

more . . .

- Pass II now asks you to write to tape. You can write over Pass I's output. Start the tape and press GO.

- Since there are no more segments, the assembly process is finished. If this had been a multi-segment program, you would have been asked to load the Collector at this point.

- The assembled program has been written to tape. You can load it by issuing

  ```
  :INPUT %(24576)
  ```

  and pressing GO when the tape is in motion.

- Execute your sample program by doing a

  ```
  CALL24576
  ```

  and you should see the results on the screen as the large letters appear spelling GENERAL VIDEO.

- As you run the assembler in the future, always jot down on a piece of paper any errors (the code, statement number and segment number where it occurred) and the high address.

- As an exercize, try repeating this whole procedure, but edit the program to spell your name instead of GENERAL VIDEO.

THE SAMPLE PROGRAMS

Both samples may be assembled for target address 24576. Sample 1 is described somewhat on the data sheet and Appendix F.

Sample 2 is a more complicated graphics program illustrating how an image may be vectored around the screen during the screen interrupt provided by the hardware. After CALLing the routine, notice that it continues to execute in background while you regain control of the normal Basic facilities. That is, the little witch continues to fly around the screen. You can stop her by negating the call to your interrupt routine with the following instruction:

```
   %(20118)=8701
```

You can easily write a Basic program that draws a haunted house and then CALLs 24576 to start the little witch flying. Your program continues to execute, and so does the background graphic until your Basic program issues the above command.