GMON 1.2

A machine language monitor for the Commodore 64.

INITIALISING GMON 1.2

To load GMON 1.2 from disk type LOAD"GMON $C000",8,1 and press RETURN. This version of GMON 1.2 loads into the 4K area of RAM between the BASIC ROM and the I/O ROM starting at $C000 (49152 in decimal). In this area it will not be affected by normal BASIC programs. To start the monitor type SYS 49152 and press RETURN. The monitor will display the registers of the processor (see the 'R' command).

RELOCATING GMON 1.2

If GMON 1.2 gets in the way of any program you are writing or using you can relocate it.

Suppose that you wanted to move GMON 1.2 to $4000 (16384). First, you must calculate the offset between $C000 and the location that you are moving GMON 1.2 to. This is done by *ADDING* $4000 to the new start of GMON 1.2, in this case $4000+$4000=$8000.

First transfer the monitor to its new position using the 'T' command, i.e. T C000 CFE8 4000

Next, all of the references between $C000 and $CFFF need to be adjusted. This is done by the following two lines, using the 'N' command:

```
N 4000 4CB9 8000 C000 D000
N 4CFC 4D4F 8000 C000 D000 W
```

Now, to save this new version:

```
S"name" 08 4000 4FE8
```

To use this new version, type SYS 16384 instead of SYS 49152.

GMON 1.2 INSTRUCTIONS

A ASSEMBLE

```
.A 0340 A9 00      LDA #$00
.A 0342 A2 10      LDX #$10
.A 0344 JMP$BDCD:garbage
```

In the above example, assembly has started at $0340. The 'A' command is used to write machine language using mnemonics and operands. To start simply type A and the address where you wish to assemble from and then the line of machine language. On pressing RETURN this is entered into memory and redisplayed in the format used by the 'D' command and the next address is printed for you to continue assembly. When you have finished simply hit RETURN. After using the 'D' command the ',' can be changed to 'A' and the mnemonics can be entered as they normally are.

D DISASSEMBLE

D 0340 will start disassembling the machine language at $0340 and will continue until the RUN/STOP key is pressed. It can be paused by pressing any other (except SHIFT, C=, CTRL and RESTORE). To disassemble the ml (machine language) between 0340 and 0360 type:

```
D 0340 0360
```

The bytes following the address can be modified by moving the cursor and typing over them. Pressing RETURN will re-enter that line into memory and redisplay the line with any modifications.

F FILL MEMORY

This command is used to fill an area of memory with a specific value. For example to fill the memory between 1000 and 2FFF inclusive with 00 the following would be typed:

```
F 1000 2FFF 00
```

G GO RUN

This command will cause an ml program ending with a BRK instruction to be run starting at the address specified after the command, i.e.

```
G 1000
```

or, if this is not present, starting at the address in the program counter, PC (see the 'R' command).

H HUNT MEMORY

Hunt memory will search memory between two specified addresses for either an ASCII string or a byte combination. For example to hunt through memory from C000 to D000 for all occurrences of 'GMON' the following would be entered:

```
H C000 D000 'GMON
```

Alternatively, to hunt for the bytes 20 E4 FF (equivalent to JSR $FFE4) the following would be typed:

```
H C000 D000 20 E4 FF
```

In both cases, the address at which every occurrence starts is displayed.

I INTERROGATE MEMORY

This command will display memory in the format shown in the example below. It can be stopped and paused using the same keys as the 'D' command. The memory can be modified by moving the cursor over the bytes and changing them (they are redisplayed after RETURN is pressed). This command has two formats:

```
I 1000 2000
```

which displays the bytes from 1000 to 2000 inclusive.

```
I 1000
```

which displays all bytes from 1000 to the end of memory (FFFF) in the following format:

```
.:1000 47 0D 4F 0E 20 31 2E 32 'GMON 1.2
```

J JUMP TO SUBROUTINE

This command is identical to 'G' except that it runs an ml program ending with RTS.

L LOAD

There are a number of different ways to load program, depending on the device and whether the user wishes to load at a specific address.

```
L                Loads any program from cassette.
L"GMON"          Loads program from tape named "GMON".
L"GMON" 08       Loads "GMON" from device 08 (disk).
L"GMON" 08 4000  Loads "GMON" from device 08 into $4000.
```

N NEWLOCATE

```
N 8000 8CB9 C000 C000 D000
```

The above example modifies all three byte instructions in memory between 8000 and 8C88 that reference memory from C000 to D000 by ADDING an offset of C000 to all addresses.

```
N 8CFC 8D4F C000 C000 D000 W
```

This will adjust all bytes in a 'word table' between the addresses specified in the same way as above.

NEWLOCATE will stop and disassemble any bad opcode that it encounters.

Note that this command does ***NOT*** move any memory - it only fixes the addresses before or after it is moved (using 'T').

O OFF PRINTER

Turns printer off and clears buffer by sending a carriage return.

P PRINTER ON

Turns the printer (device 4) on, so that all output is sent to both screen *and* printer.

R REGISTERS

This is always called on entry to GMON 1.2. It displays the current stored values of the internal registers of the 6510 when GMON 1.2 was entered. The values shown can be modified and are placed in their respective registers on execution of the 'G' and 'J' commands.

S SAVE

Similar to loading, there are a number of options when saving memory:

```
S"name" 01 1000 2000
```

saves memory from 1000 to 1FFF to cassette calling it "name".

```
S"0:name" 08 1000 2000
```

saves memory from 1000 to 1FFF to disk (drive 0) calling it "name".

```
S"0:name" 08 1000 2000 3000
```

saves memory from 1000 to 1FFF to disk calling it "name" so that it reloads from 3000 onwards. This facility is not available using cassette.

Memory from D000 to FFFF can also be saved to disk - this includes the RAM underneath.

T TRANSFER

The following example will transfer all memory from C000 to CFFF into 8000 to 8FFF.

```
T C000 CFFF 8000
```

V VERIFY

This command works with the same parameters as 'L'. It verifies the program name, or with the exception of "" on tape, with memory.

```
V "1:GMON",08,3000
```

verifies the program starting in memory at 3000 with the program called "GMON" on disk (drive 1).

X EXIT TO BASIC

Returns control back to BASIC, placing a value of #$37 into location $01 to ensure that all the ROM's are switched into place to prevent a crash or similar situation.

 # DECIMAL - HEX CONVERSION

.# 43981 prints .$ ABCD

$ HEX - DECIMAL CONVERSION

.$ ABCD prints .# 43981

DISK COMMANDS

The disk commands in GMON 1.2 are the same as used in the DOS SUPPORT detailed in the disk drive manual with the following exceptions:-
substitute '_' for '@'
_'name will display the start address of "name".
_#name displays both the start and end addresses of "name".

NOTES:

GMON 1.2 will work with ANY or ALL of the ROM's switched out with just one small restriction - memory under the I/O ROM (D000-DFFF) cannot be loaded directly, to achieve the same result load the program at some other location and use the T command to move it.

GMON 1.2 is different from many other machine language monitors because it enables one to assemble/disassemble the so-called 'QUASI-OPCODES' and it also 'frees' the area of RAM from $D000 upwards.

In the L, S and V commands each parameter can be separated by ONE character i.e. ,;%&$@ etc., or any number of spaces (up to maximum line length of 64 characters). These characters must not include 0-9 or A-F.

Many thanks to the people who gave me ideas and constructive criticism throughout to writing of this program including Roger Walsome, Cory Kin and especially to Steven Green for not getting too annoyed when I neglected other projects. Also, a special thanks to Andrew Trott for writing MIKRO - without which, it would have taken a lot longer to write GMON 1.2.

Although the utmost has been done to ensure that this program is bug-free, if you do find any errors then please contact me via the address on the front page.

---oOo---

6502 QUASI OPCODES

Instruction	Abs	Abs,X	Abs,Y	Zer	Zer,X	Zer,Y	(Ind,X)	(Ind),Y	Imm
ASO	0F	1F	1B	07	17		03	13	0B*
RLA	2F	3F	3B	27	37		23	33	2B*
LSE	4F	5F	5B	47	57		43	53	
RRA	6F	7F	7B	67	77		63	73	
AXS	8F			87		97	83		
LAX	AF		BF	A7	B7		A3	B3	
DCM	CF	DF	DB	C7	D7		C3	D3	
INS	EF	FF	FB	E7	F7		E3	F3	
ALR									4B
ARR									6B
XAA									8B
OAL									AB
SAX									CB

NOP	1A 3A 5A 7A DA FA
SKB	80 82 C2 E2 04 14 34 44 54 64 74 D4 F4 89
SKW	0C 1C 3C 5C 7C DC FC

Opcode	Function
ASO	ASL then ORA the result with .A
RLA	ROL then AND the result with .A
LSE	LSR then EOR the result with .A
RRA	ROR then ADC the result with .A
AXS	Store the result of .A AND .X
LAX	LDA and LDX with the same data
DCM	DEC memory and CMP the result with .A
INS	INC memory then SBC the result from .A
ALR	AND .A with data & LSR the result
ARR	AND .A with data & ROR the result
XAA	Store .X AND data in .A
OAL	ORA .A with #$EE, AND the result with data, then TAX
SAX	SBC data from .A AND .X & store result in .X
NOP	No operation
SKB	Skip byte (i.e. branch of +1)
SKW	Skip word (i.e. branch of +2)

Notes

Many of the codes show repetitiveness, derived from regularities in the 6502, but there are many one off possibilities too: 9B combines TSX & STA Abs,Y with the net effect of storing '.A AND .X in the stack pointer and bit 0 of .A AND .X in memory'.

Please note that .A and .X refer to the accumulator and the X index register.

All opcodes ending -2 except 82, A2, C2 and E2 crash the processor.

All of the Quasi-opcodes are based on work by Brian Grainger of ICPUG in 1981.

Also note that JMP ($xxFF) uses addresses xxFF and xx00 for the jump vector and not xxFF and xxFF+1, i.e. 02FF,0200 and *NOT* 02FF,0300.

* These instructions act as AND #$xx.