MARS can be run from a command interpreter to assemble and execute a MIPS program in a batch fashion. The format for running MARS from a command line is:

```
java -jar mars.jar [options] program.asm [more files...] [ pa arg1 [more args...]]
```

Items in *[ ]* are optional. Valid options (not case sensitive, separated by spaces) are:

Option	Description	Since
a	assemble only, do not simulate	1.0
ae*n*	terminate MARS with integer exit code *n* if assembly error occurs	4.1
ascii	display memory or register contents interpreted as ASCII codes. (alternatives are dec and hex)	4.1
b	brief - do not display register/memory address along with contents	2.2
d	display MARS debugging statements (of interest mainly to MARS developer)	1.0
db	MIPS delayed branching is enabled.	3.3
dec	display memory or register contents in decimal. (alternatives are ascii and hex)	2.2
dump	dump memory contents to file. Option has 3 arguments, e.g. dump <format> <file>. Current supported segments are .text and .data. Also supports an address range (see *m-n* below). Current supported dump formats are Binary, HexText, BinaryText, AsciiText. See examples below.	3.4
hex	display memory or register contents in hexadecimal - this is the default. (alternatives are ascii and dec)	2.2
h	display this help. Use this option by itself and with no filename.	1.0
ic	display instruction count; the number of MIPS basic instructions 'executed'	4.3
mc	set memory configuration. Option has 1 argument, e.g. mc <config>. Argument <config> is case-sensitive and its possible values are Default for the default 32-bit address space, CompactDataAtZero for a 32KB address space with data segment at address 0, or CompactTextAtZero for a 32KB address space with text segment at address 0.	3.7
me	display MARS messages to standard err instead of standard out. Allows you to separate MARS messages from MIPS program output using redirection.	4.3
nc	copyright notice will not be displayed. Useful if redirecting or piping program output.	3.5
np	pseudo-instructions or extended instruction formats are not permitted.	3.0
p	project option - will assemble the specified file and all other assembly files (*.asm; *.s) in its directory.	3.1
se*n*	terminate MARS with exit code *n* if simulate (run) error occurs	4.1
sm	start execution at statement having global label 'main' if defined	3.8
smc	Self Modifying Code - Program can write and execute in either text or data segment	4.4
we	assembler warnings will be considered errors.	3.5
n	where *n* is an integer maximum count of execution steps to simulate. If 0, negative or not specified, there is no maximum.	1.0
$*reg*	where *reg* is number or name (e.g. 5, t3, f10) of register whose content to display at end of run. Even-numbered float register displays both float and double. Option may be repeated. *NOTE: Depending on your command shell, you may need to escape the $, e.g. \$t3*	2.2

reg_name	where *reg_name* is the name (e.g. t3, f10) of register whose content to display at end of run. Even-numbered float register displays both float and double. Option may be repeated. $ not required.	2.2
m-*n*	memory address range from *m* to *n* whose contents to display at end of run. *m* and *n* may be decimal or hexadecimal (starts with `0x`), *m* <= *n*, both must be on word boundary. Option may be repeated.	2.2
`pa`	program arguments - all remaining space-separated items are argument values provided to the MIPS program via $a0 (argc - argument count) and $a1 (argv - address of array containing pointers to null-terminated argument strings). The count is also at the top of the runtime stack ($sp), followed by the array.*This option and its arguments must be the last items in the command!*	3.5

Example: `java -jar mars.jar h`
Displays command options and explanations.

Example: `java -jar mars.jar $s0 $s1 0x10010000-0x10010010 fibonacci.asm`
Assemble and run `fibonacci.asm`. At the end of the run, display the contents of registers `$s0` and `$s1`, and the contents of memory locations 0x10010000 through 0x10010010. The contents are displayed in hexadecimal format.

Example: `java -jar mars.jar a fibonacci.asm`
Assemble `fibonacci.asm`. Does not attempt to run the program, and the assembled code is not saved.

Example: `java -jar mars.jar 100000 infinite.asm`
Assemble and run `infinite.asm` for a maximum of 100,000 execution steps.

Example: `java -jar mars.jar p major.asm`
Assemble `major.asm` and all other files in the same directory, link the assembled code, and run starting with the first instruction in `major.asm`.

Example: `java -jar mars.jar major.asm minor.asm sub.asm`
Assemble and link `major.asm`, `minor.asm` and `sub.asm`. If successful, execution will begin with the first instruction in `major.asm`.

Example: `java -jar mars.jar a dump .text HexText hexcode.txt fibonacci.asm`
Assemble `fibonacci.asm` without simulating (note use of 'a' option). At end of assembly, dump the text segment (machine code) to file `hexcode.txt` in hexadecimal text format with one instruction per line.

Example: `java -jar mars.jar dump 0x10010000-0x10010020 HexText hexcode.txt fibonacci.asm`
Assemble and simulate `fibonacci.asm`. At end of simulation, dump the contents of addresses 0x1001000 to 0x10010020 to file `hexdata.txt` in hexadecimal text format with one word per line.

Example: `java -jar mars.jar t0 process.asm pa counter 10`
Assemble and run `process.asm` with two program argument values, "counter" and "10". It may retrieve the argument count (2) from `$a0`, and the address of an array containing pointers to the strings "count" and "10", from `$a1`. At the end of the run, display the contents of register `$t0`.

The ability to run MARS from the command line is useful if you want to develop scripts (macros) to exercise a given MIPS program under multiple scenarios or if you want to run a number of different MIPS programs such as for grading purposes.
