SPIM Command-Line Options

All versions of SPIM (spim, xspim, and PCSpim) have command-line options that control how SPIM starts running. The general format is

```
spim arguments file.s program_arguments
```

where *spim* is the name of a particular version of SPIM (`spim`, `PCSpim`, or `xspim`), *arguments* are the command-line options described below, *file.s* is the name of a file containing a MIPS program, and *program_arguments* are the initial arguments passed to the MIPS program. For example, to just start `xspim` without any arguments or an initial program, type

```
xspim
```

Or, to start `PCSpim` with delayed branches on file `test.s`, type

```
pcspim -delayed_branches test.s
```

The programs accept the following command-line options:

`-asm`, `-a`	Simulate the virtual MIPS machine provided by the assembler. This is the default.
`-bare`, `-b`	Simulate a bare MIPS machine without pseudoinstructions or the additional addressing modes provided by the assembler. Implies the flags `-quiet`, `-delayed_branches`, and `-delayed_loads`.
`-delayed_branches`, `-db`	Execute the instruction following a branch or jump in its delay slot. The offset in a branch must be adjusted to reflect that the control transfer occurs when the PC points to the instruction in the delay slot. The default does not simulate the delay slot, so the offset is relative to the branch instruction.
`-delayed_loads`, `dl`	A value loaded from memory is not available to the instruction executed immediately after the load instruction. The default is that loaded values are available immediately.
`-exception`, `-e`	Load the standard exception handler and start-up code. This is the default.
`-noexception`, `-ne`	Do not load the standard exception handler or start-up code. The program must provide code to

	handle exceptions. When an exception occurs, SPIM jumps to location 80000180_{hex}. The start-up code invokes the routine `main` with arguments `argc` and `argv`. Without the start-up routine, SPIM starts execution at the instruction labeled `__start`.
`-exception_file, ef`	Load the the exception handler and start-up code from this file instead of the default file.
`-mapped_io, mio`	Enable the memory-mapped I/O facility (see Section A.8). Programs that use SPIM `syscalls` (see section on "System Calls," page A-43) to read from the terminal *cannot* also use memory-mapped I/O.
`-nomapped_io, nmio`	Disable the memory-mapped I/O facility (see Section A.8). This is the default.
`-pseudo, -p`	Allow the input assembly code to contain pseudo-instructions. This is the default.
`-nopseudo, -np`	Do not allow pseudoinstructions in the input assembly code.
`-quiet, -q`	Do not print a message when exceptions occur.
`-noquiet, -nq`	Print a message when an exception occurs. This is the default.
`-file <name>, -f`	Load and execute the assembly code in the file whose name is given.
`-s <seg> size, -st, -sd, -ss, -skt, -skd`	Sets the initial size of memory segment *seg* to be *size* bytes. The memory segments are named `text`, `data`, `stack`, `ktext`, and `kdata`. The `text` segment contains instructions from a program. The `data` segment holds the program's data. The `stack` segment holds its runtime stack. In addition to running a program, SPIM also executes system code that handles interrupts and exceptions. This code resides in a separate part of the address space called the *kernel*. The `ktext` segment holds this code's instructions, and `kdata` holds its data. There is no `kstack` segment since the system code

	uses the same stack as the program. For example, the pair of arguments `-sdata 2000000` starts the user data segment at 2,000,000 bytes.
`-l <seg> size`, `-ld`, `-ls`, `-lkd`	Sets the limit on how large memory segment *seg* can grow to be *size* bytes. The memory segments that can grow are `data`, `stack`, and `kdata`.