Many facets of the system are configurable. Look inside the file system.setup for some clues.

As you know, you start the emulator with the command "computer". If you wish any .exe files to be automatically loaded into memory at startup, there are two options. Either add a line like this "load file.exe at 0x1400" to system.setup, or include the file name on the command line like this: "computer file". In the latter case, file.exe will be loaded starting at memory location 0x400. If you wish the system to automatically start running, without any interaction or single stepping, just add "-r" to the command line, like this "computer file -r", and execution will start at PC=0x400.

When the system starts up, some useful information is left in particular memory locations:

memory[0x100] = amount of memory actually installed and available continuously from location 0. In other words, if [0x100] is 12345, then all physical memory locations from 0 to 12344 exist.

memory[0x101] = first memory address that nothing (from an .exe file) has been loaded into - the beginning of free space.

memory[0x102] = the number of fake disc drives (as defined in system.setup) available for use.

If you do not use the "-r" command line option, the emulator starts in debug mode. It gives you the prompt ">" and waits for commands.

The commands are...

START: When single stepping through a program produced by the BCPL compiler, this should be used once before STEP is used. It runs silently through the initialisation code for static variables, and returns to single step mode when the first “real” instruction of the program is reached, the call to start().

ENTER: Just press Enter and the system goes into single step mode. It shows the values of all the registers, and the next instruction to be executed, and lets you take control. While in single step mode, if you just press enter, it will execute just one instruction then wait for another command. If you type the command "x" it will leave single step mode and wait for another command. All other commands are equally available in or out of single step mode.

STEP or S followed by a number:
Execute that number of instructions, showing all debugging information after each. A BREAK instruction stops execution.

RUN or R:
Continue running the program without debugging. Just run normally as though the "-r" command line option had been used. In other words, behave like a real computer running normally.

RUN or R followed by a number:

The same as just "run" above, but after the given number of instructions have been executed, the system returns to single step mode and waits for another command.

ALL

Shows the values of all special registers.

INPUT xxx

This behaves as though the keys xxx had been typed to a normally running program: the characters are added to the hardware keyboard buffer. If you press Enter at the end of the command, then the '\n' character is added to the keyboard input buffer. If, on the other hand, you end the line by pressing ESC instead of enter, nothing extra is added.

This command does not automatically generate an interrupt.

INT n

Signals interrupt number n. For example “INT 9” signals a keyboard interrupt.

BREAK n addr

Sets breakpoint n at address addr. For example “BREAK 1 0x437”. There are three breakpoints available, numbered 1, 2, 3. Setting a breakpoint to address 0 has the effect of clearing or disabling it.

BREAK 0

Clears (disables) all three breakpoints.

TRAIL

Shows the last 16 values the program counter had, along with the instructions executed there, and any interrupts that occurred during the period.

SHOWINT

Show the contents of the interrupt vector and perform basic validity tests.

SHOWCG

Show the contents of the call gate vector.

SHOWVM

Display the page directory and all non-zero page table entries with information on virtual address ranges served.

?

Shows the contents of a register or some memory locations. If the "?" is followed by the name of a register or a special register, it simply shows the value of that register, e.g. "?R3" or "?INTVEC". Other options are best illustrated by example:

?123

Show memory location 0x123

?123:20
: Show 20 memory locations starting from location 0x123

?R1+5
: Show the contents of [R1+5]

?FP-2
: Show the second local variable (if you are using the stack correctly)

?FP-2-8
: Show all memory locations from [FP-2] to [FP-8]

?R5-3+4
: Show all memory locations from [R5-3] to [R5+4]

?FP-15:10
: Show all memory locations from [FP-15] to [FP-5]

?FP
: Show the value of the FP register

?FP+0
: Show the contents of memory location [FP+0]

of course, R1, FP, R5 etc can be replaced by any register name

P

Exactly the same as "?", except that virtual memory is ignored. "p? 123" shows the contents of physical memory location 0x123 regardless of whether virtual memory is turned on or not.

SET

Change the value of a register, special register, or memory location. Examples:

```
set r1=58
set r1=0x58
set intvec=0x1800
set 1800=12
```

the latter example stores the number 12 (decimal) in location 0x1800.

PSET

Exactly the same as set, except that virtual memory is ignored. "pset 100=3" stores 3 in physical memory location 0x100.

VM addr

Shows every stage in the conversion of virtual address "addr" into a physical address. Used to check that page tables are correctly set up.

X

If single stepping, stop single stepping and go back to normal command mode.
If not single stepping, exit completely.

EXIT, Q, or QUIT

Exit completely.

Control-C is properly trapped. If a program is running, control-C instantly stops it and puts the system in single step debugging mode. If no program is running, control-

C is harmless. BUT two consecutive control-C's will exit the system. This is just in case I screwed up the control-c trapping. We don't want any unstoppable programs.

Whilst a program is running (not in single step mode, but properly running), everything typed at the keyboard is instantly captured, without waiting for enter to be pressed. If interrupts are enabled (i.e. the $IP flag "interrupt being processed" is off), the IV$KEYBD interrupt is caused.