SYSCALL functions available in MARS

Introduction

A number of system services, mainly for input and output, are available for use by your MIPS program. They are described in the table below.

MIPS register contents are not affected by a system call, except for result registers as specified in the table below.

How to use SYSCALL system services

Step 1. Load the service number in register $v0.
Step 2. Load argument values, if any, in $a0, $a1, $a2, or $f12 as specified.
Step 3. Issue the SYSCALL instruction.
Step 4. Retrieve return values, if any, from result registers as specified.

Example: display the value stored in $t0 on the console

```
li  $v0, 1           # service 1 is print integer
add $a0, $t0, $zero  # load desired value into argument register $a0, using pseudo-op
syscall
```

Table of Available Services

Service	Code in $v0	Arguments	Result
print integer	1	$a0 = integer to print	
print float	2	$f12 = float to print	
print double	3	$f12 = double to print	
print string	4	$a0 = address of null-terminated string to print	
read integer	5		$v0 contains integer read
read float	6		$f0 contains float read
read double	7		$f0 contains double read
read string	8	$a0 = address of input buffer $a1 = maximum number of characters to read	*See note below table*
sbrk (allocate heap		$a0 = number of bytes	

memory)	9	to allocate	$v0 contains address of allocated memory
exit (terminate execution)	10		
print character	11	$a0 = character to print	*See note below table*
read character	12		$v0 contains character read
open file	13	$a0 = address of null-terminated string containing filename $a1 = flags $a2 = mode	$v0 contains file descriptor (negative if error). *See note below table*
read from file	14	$a0 = file descriptor $a1 = address of input buffer $a2 = maximum number of characters to read	$v0 contains number of characters read (0 if end-of-file, negative if error). *See note below table*
write to file	15	$a0 = file descriptor $a1 = address of output buffer $a2 = number of characters to write	$v0 contains number of characters written (negative if error). *See note below table*
close file	16	$a0 = file descriptor	
exit2 (terminate with value)	17	$a0 = termination result	*See note below table*
Services 1 through 17 are compatible with the SPIM simulator, other than Open File (13) as described in the Notes below the table. Services 30 and higher are exclusive to MARS.			
time (system time)	30		$a0 = low order 32 bits of system time $a1 = high order 32 bits of system time. *See note below table*
MIDI out	31	$a0 = pitch (0-127) $a1 = duration in milliseconds $a2 = instrument (0-127) $a3 = volume (0-127)	Generate tone and return immediately. *See note below table*
sleep	32	$a0 = the length of time to sleep in milliseconds.	Causes the MARS Java thread to sleep for (at least) the specified number of milliseconds. This timing will not be precise, as the Java implementation will add some overhead.
MIDI out synchronous	33	$a0 = pitch (0-127) $a1 = duration in milliseconds $a2 = instrument (0-127) $a3 = volume (0-127)	Generate tone and return upon tone completion. *See note below table*

print integer in hexadecimal	34	$a0 = integer to print	Displayed value is 8 hexadecimal digits, left-padding with zeroes if necessary.
print integer in binary	35	$a0 = integer to print	Displayed value is 32 bits, left-padding with zeroes if necessary.
print integer as unsigned	36	$a0 = integer to print	Displayed as unsigned decimal value.
(not used)	37-39		
set seed	40	$a0 = i.d. of pseudorandom number generator (any int). $a1 = seed for corresponding pseudorandom number generator.	No values are returned. Sets the seed of the corresponding underlying Java pseudorandom number generator (`java.util.Random`). *See note below table*
random int	41	$a0 = i.d. of pseudorandom number generator (any int).	$a0 contains the next pseudorandom, uniformly distributed int value from this random number generator's sequence. *See note below table*
random int range	42	$a0 = i.d. of pseudorandom number generator (any int). $a1 = upper bound of range of returned values.	$a0 contains pseudorandom, uniformly distributed int value in the range 0 <= [int] < [upper bound], drawn from this random number generator's sequence. *See note below table*
random float	43	$a0 = i.d. of pseudorandom number generator (any int).	$f0 contains the next pseudorandom, uniformly distributed float value in the range 0.0 <= f < 1.0 from this random number generator's sequence. *See note below table*
random double	44	$a0 = i.d. of pseudorandom number generator (any int).	$f0 contains the next pseudorandom, uniformly distributed double value in the range 0.0 <= f < 1.0 from this random number generator's sequence. *See note below table*
(not used)	45-49		
ConfirmDialog	50	$a0 = address of null-terminated string that is the message to user	$a0 contains value of user-chosen option 0: Yes 1: No 2: Cancel
InputDialogInt	51	$a0 = address of null-terminated string that is the message to user	$a0 contains int read $a1 contains status value 0: OK status -1: input data cannot be correctly parsed -2: Cancel was chosen -3: OK was chosen but no data had been input into field
			$f0 contains float read $a1 contains status value

InputDialogFloat	52	$a0 = address of null-terminated string that is the message to user	0: OK status -1: input data cannot be correctly parsed -2: Cancel was chosen -3: OK was chosen but no data had been input into field
InputDialogDouble	53	$a0 = address of null-terminated string that is the message to user	$f0 contains double read $a1 contains status value 0: OK status -1: input data cannot be correctly parsed -2: Cancel was chosen -3: OK was chosen but no data had been input into field
InputDialogString	54	$a0 = address of null-terminated string that is the message to user $a1 = address of input buffer $a2 = maximum number of characters to read	*See Service 8 note below table* $a1 contains status value 0: OK status. Buffer contains the input string. -2: Cancel was chosen. No change to buffer. -3: OK was chosen but no data had been input into field. No change to buffer. -4: length of the input string exceeded the specified maximum. Buffer contains the maximum allowable input string plus a terminating null.
MessageDialog	55	$a0 = address of null-terminated string that is the message to user $a1 = the type of message to be displayed: 0: error message, indicated by Error icon 1: information message, indicated by Information icon 2: warning message, indicated by Warning icon 3: question message, indicated by Question icon other: plain message (no icon displayed)	N/A
MessageDialogInt	56	$a0 = address of null-terminated string that is an information-type message to user $a1 = int value to display in string form after the first string	N/A
		$a0 = address of null-terminated string that	

MessageDialogFloat	57	is an information-type message to user $f12 = float value to display in string form after the first string	N/A
MessageDialogDouble	58	$a0 = address of null-terminated string that is an information-type message to user $f12 = double value to display in string form after the first string	N/A
MessageDialogString	59	$a0 = address of null-terminated string that is an information-type message to user $a1 = address of null-terminated string to display after the first string	N/A

NOTES: Services numbered 30 and higher are not provided by SPIM
Service 8 - Follows semantics of UNIX 'fgets'. For specified length n, string can be no longer than n-1. If less than that, adds newline to end. In either case, then pads with null byte If n = 1, input is ignored and null byte placed at buffer address. If n < 1, input is ignored and nothing is written to the buffer.
Service 11 - Prints ASCII character corresponding to contents of low-order byte.
Service 13 - MARS implements three flag values: 0 for read-only, 1 for write-only with create, and 9 for write-only with create and append. It ignores mode. The returned file descriptor will be negative if the operation failed. The underlying file I/O implementation uses `java.io.FileInputStream.read()` to read and `java.io.FileOutputStream.write()` to write. MARS maintains file descriptors internally and allocates them starting with 3. File descriptors 0, 1 and 2 are always open for: reading from standard input, writing to standard output, and writing to standard error, respectively (new in release 4.3).
Services 13,14,15 - In MARS 3.7, the result register was changed to $v0 for SPIM compatability. It was previously $a0 as erroneously printed in Appendix B of *Computer Organization and Design,.*
Service 17 - If the MIPS program is run under control of the MARS graphical interface (GUI), the exit code in $a0 is ignored.
Service 30 - System time comes from `java.util.Date.getTime()` as milliseconds since 1 January 1970.
Services 31,33 - Simulate MIDI output through sound card. Details below.
Services 40-44 use underlying Java pseudorandom number generators provided by the `java.util.Random` class. Each stream (identified by $a0 contents) is modeled by a different `Random` object. There are no default seed values, so use the Set Seed service (40) if replicated random sequences are desired.

Example of File I/O

The sample MIPS program below will open a new file for writing, write text to it from a memory buffer, then close it. The file will be created in the directory in which MARS was run.

```
# Sample MIPS program that writes to a new file.
#   by Kenneth Vollmar and Pete Sanderson
```

```
        .data
fout:   .asciiz "testout.txt"      # filename for output
buffer: .asciiz "The quick brown fox jumps over the lazy dog."
        .text
  ###############################################################
  # Open (for writing) a file that does not exist
  li   $v0, 13       # system call for open file
  la   $a0, fout     # output file name
  li   $a1, 1        # Open for writing (flags are 0: read, 1: write)
  li   $a2, 0        # mode is ignored
  syscall            # open a file (file descriptor returned in $v0)
  move $s6, $v0      # save the file descriptor
  ###############################################################
  # Write to file just opened
  li   $v0, 15       # system call for write to file
  move $a0, $s6      # file descriptor
  la   $a1, buffer   # address of buffer from which to write
  li   $a2, 44       # hardcoded buffer length
  syscall            # write to file
  ###############################################################
  # Close the file
  li   $v0, 16       # system call for close file
  move $a0, $s6      # file descriptor to close
  syscall            # close file
  ###############################################################
```

Using SYSCALL system services 31 and 33: MIDI output

These system services are unique to MARS, and provide a means of producing sound. MIDI output is simulated by your system sound card, and the simulation is provided by the `javax.sound.midi` package.

Service 31 will generate the tone then immediately return. Service 33 will generate the tone then sleep for the tone's duration before returning. Thus it essentially combines services 31 and 32.

This service requires four parameters as follows:

pitch ($a0)

- Accepts a positive byte value (0-127) that denotes a pitch as it would be represented in MIDI
- Each number is one semitone / half-step in the chromatic scale.
- 0 represents a very low C and 127 represents a very high G (a standard 88 key piano begins at 9-A and ends at 108-C).
- If the parameter value is outside this range, it applies a default value 60 which is the same as middle C on a piano.
- From middle C, all other pitches in the octave are as follows:
  - 61 = C# or Db
  - 62 = D
  - 63 = D# or Eb
  - 64 = E or Fb
  - 65 = E# or F
  - 66 = F# or Gb
  - 67 = G
  - 68 = G# or Ab
  - 69 = A
  - 70 = A# or Bb
  - 71 = B or Cb
  - 72 = B# or C
- To produce these pitches in other octaves, add or subtract multiples of 12.

duration in milliseconds ($a1)

- Accepts a positive integer value that is the length of the tone in milliseconds.
- If the parameter value is negative, it applies a default value of one second (1000 milliseconds).

instrument ($a2)

- Accepts a positive byte value (0-127) that denotes the General MIDI "patch" used to play the tone.
- If the parameter is outside this range, it applies a default value 0 which is an *Acoustic Grand Piano*.
- General MIDI standardizes the number associated with each possible instrument (often referred to as *program change* numbers), however it does not determine how the tone will sound. This is determined by the synthesizer that is producing the sound. Thus a *Tuba* (patch 58) on one computer may sound different than that same patch on another computer.
- The 128 available patches are divided into instrument families of 8:

0-7	Piano	64-71	Reed
8-15	Chromatic Percussion	72-79	Pipe
16-23	Organ	80-87	Synth Lead
24-31	Guitar	88-95	Synth Pad
32-39	Bass	96-103	Synth Effects
40-47	Strings	104-111	Ethnic
48-55	Ensemble	112-119	Percussion
56-63	Brass	120-127	Sound Effects

- Note that outside of Java, General MIDI usually refers to patches 1-128. When referring to a list of General MIDI patches, 1 must be subtracted to play the correct patch. For a full list of General MIDI instruments, see www.midi.org/about-midi/gm/gm1sound.shtml. The General MIDI channel 10 percussion key map is not relevant to the toneGenerator method because it always defaults to MIDI channel 1.

volume ($a3)

- Accepts a positive byte value (0-127) where 127 is the loudest and 0 is silent. This value denotes MIDI velocity which refers to the initial attack of the tone.
- If the parameter value is outside this range, it applies a default value 100.
- MIDI velocity measures how hard a *note on* (or *note off*) message is played, perhaps on a MIDI controller like a keyboard. Most MIDI synthesizers will translate this into volume on a logarithmic scale in which the difference in amplitude decreases as the velocity value increases.
- Note that velocity value on more sophisticated synthesizers can also affect the timbre of the tone (as most instruments sound different when they are played louder or softer).

System service 31 was developed and documented by Otterbein student Tony Brock in July 2007.