# MIPS conventions for memory usage

- Not hardware enforced.
- Programmer/Assembler/Linker/Loader conventions
- Divide memory in 3 parts: *text*, *data*, & *stack* segments

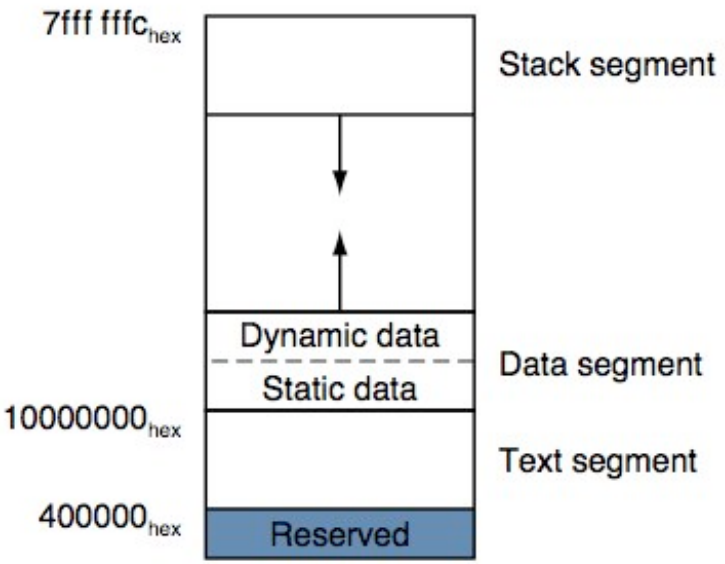


Figure: Memory Layout

# SPIM

- Simulates SPIM
- What it doesn't simulate
  - Instruction timings
  - Actual memory
- Instructions in memory may differ

# SPIM Interface

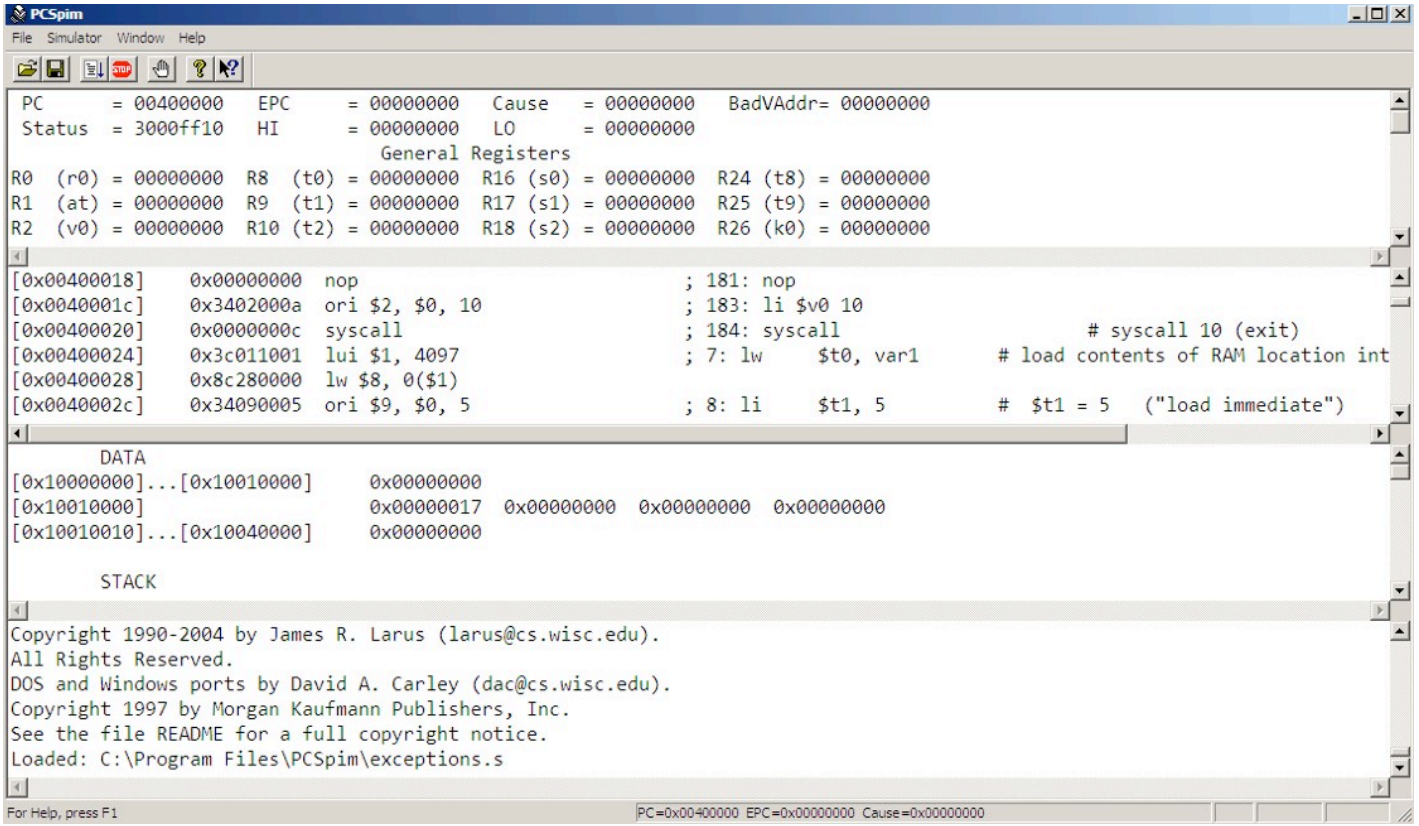


Figure: SPIM Interface

# Assembly program skeleton

```
# program1.s
# Skeleton program; does nothing

            .data
                        # variable declarations for the program

            .text

main:                   # indicates start of code
                        # ...
```

Figure: Skeleton MIPS program

# Assembler Syntax

- Comments begin with #
- Labels: alphanumeric, _, .; begin with an alphabet
- Directives begin with a .
- Register addressing: `$8`, `$16` or `$t0`, `$s0`

# Assembler Syntax

- Data declarations: `.data`
  - syntax: `name: storage_type value(s)`
  - types possible: `.byte, .word, .space, .asciiz`
  - `num: .word 5`
  - `array1: .byte 'a', 'b'`

# Simple MIPS program

```
        .data
var1:       .word   23

        .text
main:
        lw  $t0, var1
        li  $t1, 5
        sw  $t1, var1
```

Figure: Simple MIPS program

# System Calls

- `syscall` instruction for OS-like commands
- Load the proper code in `$v0`, call `syscall`

| Service | System call code | Arguments | Result |
|---|---|---|---|
| print_int | 1 | $a0 = integer | |
| print_float | 2 | $f12 = float | |
| print_double | 3 | $f12 = double | |
| print_string | 4 | $a0 = string | |
| read_int | 5 | | integer (in $v0) |
| read_float | 6 | | float (in $f0) |
| read_double | 7 | | double (in $f0) |
| read_string | 8 | $a0 = buffer, $a1 = length | |
| sbrk | 9 | $a0 = amount | address (in $v0) |
| exit | 10 | | |
| print_char | 11 | $a0 = char | |
| read_char | 12 | | char (in $a0) |
| open | 13 | $a0 = filename (string), $a1 = flags, $a2 = mode | file descriptor (in $a0) |
| read | 14 | $a0 = file descriptor, $a1 = buffer, $a2 = length | num chars read (in $a0) |
| write | 15 | $a0 = file descriptor, $a1 = buffer, $a2 = length | num chars written (in $a0) |
| close | 16 | $a0 = file descriptor | |
| exit2 | 17 | $a0 = result | |

Figure: `syscall` codes

# MIPS program using `syscall`

```
.data
str:
    .asciiz "the answer = "
.text

li $v0, 4 # system call code for print_str
la $a0, str # address of string to print
syscall # print the string

li $v0, 1 # system call code for print_int
li $a0, 5 # integer to print
syscall # print it
```

Figure: MIPS program that uses `syscall`