```
# CS 2734, Spring 2000
# hello.s: Simple "Hello World" example
# Print a string
# Register mapping:
#   $s7    Used to hold return address
#   $v0    specifies the type of systme call
#   $a0    the value used by the system call
#
        .globl main
main:                           # main has to be a global label
        addu    $s7, $0, $ra    # save the return address in a global
register

## Output the string "Hello World" on separate line
        .data
        .globl  hello
hello:  .asciiz "\nHello World\n"       # string to print
        .text
        li      $v0, 4          # print_str (system call 4)
        la      $a0, hello      # takes the address of string as an argument
        syscall

## Usual stuff at the end of the main
        addu    $ra, $0, $s7    # restore the return address
        jr      $ra             # return to the main program
################################################################
# Result of a program run:
# four06% spim -file hello.s
# SPIM Version 6.0 of July 21, 1997
# Copyright 1990-1997 by James R. Larus (larus@cs.wisc.edu).
# All Rights Reserved.
# See the file README for a full copyright notice.
# Loaded: /usr/local/lib/trap.handler
#
# Hello World
# four06%
```

```
# array0.s: Simple array example
# implements the following examples from PH:
#         g = h + A[8]          PH p. 112
#         A[12] = h + A[8]      PH p. 113
#         g = h + A[i]          PH p. 114
# assumes:
#   variables f through h are in registers $s0 through $s2
#   address of A is in $s3
#   variable i is in $s4

        .globl main
main:                           # main has to be a global label
        addu    $s7, $0, $ra    # save the return address in a global reg

        addi    $s2, $0, 43     # set h to 43 for the examples
        addi    $s4, $0, 3      # set i to 3 for the examples
        la      $s3, A          # $s3 has the starting address of A
## Allocate 100-word array A while we think of it
        .data
        .align  2               # Let's make sure that it's aligned
A:      .word   11,12,13,14,15,16,17,18,19,20   # Allocate and initialize 10
                                                # elements of A
        .space  360             # Being lazy, just allocate space for rest
#---------------------------------------Compute g = h + A[8]
        .text
        lw      $t0, 32($s3)    # Temporary register $t0 gets A[8]
        add     $s1, $s2, $t0   # Calculate g = h + A[8]
                        # Output the value
        .data
        .globl  message1
message1:       .asciiz "\nThe value of g = h + A[8] is: " # string to print
        .text
        li      $v0, 4          # print_str (system call 4)
        la      $a0, message1   # takes the address of string as an argument
        syscall
        li      $v0, 1          # print_int (system call 1)
        add     $a0, $0, $s1    # put value g in $a0 for printing
        syscall

#---------------------------------------Compute A[12] = h + A[8]
        .text
        lw      $t0, 32($s3)    # Temporary register $t0 gets A[8]
        add     $t0, $s2, $t0   # Temporary register $t0 gets h + A[8]
        sw      $t0, 48($s3)    # Store h + A[8] back in A[12]
                        # Output the value
        .data
        .globl  message2
message2:       .asciiz "\nThe value of A[12]: "      # string to print
        .text
        li      $v0, 4          # print_str (system call 4)
        la      $a0, message2   # takes the address of string as an argument
        syscall
        li      $v0, 1          # print_int (system call 1)
        lw      $a0, 48($s3)    # load A[12] into $a0 for printing
        syscall

#---------------------------------------Compute g = h + A[i]
        .text
        add     $t1, $s4, $s4   # Compute offset of A[i] from start in bytes
        add     $t1, $t1, $t1   #   as 4*i
                                # Book hasn't covered shifts so use add
        add     $t1, $t1, $s3   # Now compute actual addr of A[i]: A + 4*i
        lw      $t0, 0($t1)     # Temporary register $t0 gets A[i]
        add     $s1, $s2, $t0   # Calculate g = h + A[i]
                        # Output the value
        .data
        .globl  message3
message3:       .asciiz "\nThe value of g = h + A[i] is: " # string to print
        .text
        li      $v0, 4          # print_str (system call 4)
        la      $a0, message3   # takes the address of string as an argument
        syscall
        li      $v0, 1          # print_int (system call 1)
        add     $a0, $0, $s1    # put value of g in $a0 for printing
        syscall

## Usual stuff at the end of the main
        addu    $ra, $0, $s7    # restore the return address
        jr      $ra             # return to the main program
################################################################
# Result of a program run:
# four06% spim -file array0.s
# SPIM Version 6.0 of July 21, 1997
# Copyright 1990-1997 by James R. Larus (larus@cs.wisc.edu).
# All Rights Reserved.
# See the file README for a full copyright notice.
# Loaded: /usr/local/lib/trap.handler
#
# The value of g = h + A[8] is: 62
# The value of A[12]: 62
# The value of g = h + A[i] is: 57four06%
################################################################
```