

```

## vowel.asm - prints out number of vowels in
##           - the string str
##           a0 - points to the string
#####
#           text segment #
#####
        .text
        .globl main
main:    # execution starts here
        la $a0,str
        jal vcount # call vcount

        move $a0,$v0
        li $v0,1
        syscall      # print answer

        la $a0,end1
        li $v0,4
        syscall      # print newline

        li $v0,10
        syscall      # au revoir...

#-----
# vowelp - takes a single character as a
# parameter and returns 1 if the character
# is a (lower case) vowel otherwise return 0.
#     a0 - holds character
#     v0 - returns 0 or 1
#-----
vowelp: li $v0,0
        beq $a0,'a',yes
        beq $a0,'e',yes
        beq $a0,'i',yes
        beq $a0,'o',yes
        beq $a0,'u',yes
        jr $ra
yes:    li $v0,1
        jr $ra

#-----
# vcount - use vowelp to count the vowels in a
# string.
#     a0 - holds string address
#     s0 - holds number of vowels
#     v0 - returns number of vowels
#-----
vcount:
        sub $sp,$sp,16 # save registers on stack
        sw $a0,0($sp)
        sw $s0,4($sp)
        sw $s1,8($sp)
        sw $ra,12($sp)

```

```

        li $s0,0 # count of vowels
        move $s1,$a0#address of string

nextc: lb $a0,($s1)# get each character
        beqz $a0,done # zero marks end
        jal vowelp # call vowelp
        add $s0,$s0,$v0 # add 0 or 1 to count
        add $s1,$s1,1 # move along string
        b nextc
done:   move $v0,$s0# use $v0 for result

        lw $a0,0($sp) # restore registers
        lw $s0,4($sp)
        lw $s1,8($sp)
        lw $ra,12($sp)
        add $sp,$sp,16
        jr $ra

#####
# data segment
#####
        .data
str:   .asciiz "long time ago in a galaxy far away"
end1: .asciiz "\n"

##
## end of file vowel.asm

```

```
## hex.asm: ask user for decimal number, convert to hex, print the result.
```

```
## t0 - count for 8 digits in word
```

```
## t1 - each hex digit in turn
```

```
## t2 - number read in
```

```
## t3 - address of area used to set up answer string
```

```
#####
```

```
# text segment #
```

```
#####
```

```
.text
```

```
.globl main
```

```
main:
```

```
la $a0,prompt # print prompt on terminal
```

```
li $v0,4
```

```
syscall
```

```
li $v0,5 # syscall 5 reads an integer
```

```
syscall
```

```
move $t2,$v0 # $t2 holds hex number
```

```
la $a0,ans1 # print string before result
```

```
li $v0,4
```

```
syscall
```

```
li $t0,8 # eight hex digits in word
```

```
la $t3,result # answer string set up here
```

```
loop: rol $t2,$t2,4 # start with leftmost digit
```

```
and $t1,$t2,0xf # mask one digit
```

```
ble $t1,9,print # check if 0 to 9
```

```
add $t1,$t1,7 # 7 chars between '9' and 'A'
```

```
print: add $t1,$t1,48 # ASCII '0' is 48
```

```
sb $t1,($t3) # save in string
```

```
add $t3,$t3,1 # advance destination pointer
```

```
add $t0,$t0,-1 # decrement counter
```

```
bnez $t0,loop # and continue if counter>0
```

```
la $a0,result # print result on terminal
```

```
li $v0,4
```

```
syscall
```

```
li $v0,10
```

```
syscall # au revoir...
```

```
#####
```

```
# data segment #
```

```
#####
```

```
.data
```

```
result: .space 8
```

```
.asciiz "\n"
```

```
prompt: .asciiz "Enter decimal number: "
```

```
ans1: .asciiz "Hexadecimal is "
```

**In C:**

```
int leaf_example ( int g, int h, int i, int j)
{
    int f;
    f = (g+h) - (i+j);
    return f;
}
```

**In MIPS:**

```
leaf_example:
    sub $sp, $sp, 12      # make room for 3 items on stack
    sw $t1, 8($sp)       # save $t1
    sw $t0, 4($sp)       # save $t0
    sw $s0, 0($sp)       # save $s0

    add $t0, $a0, $a1     # register t0 contains g +h
    add $t1, $a2, $a3     # register t1 contains i+j
    sub $s0, $t0, $t1     # f = t0 - t1

    move $v0, $s0        # returns f in $v0

    lw $s0, 0($sp)       #restore $s0
    lw $t0, 4($sp)       #restore $t0
    lw $t1, 8($sp)       #restore $t1

    add $sp, $sp, 12     # adjust stack after deleting 3 items

    jr $ra               # return to calling function
```