```
# CPCS 214 Computer Organization & Architecture
# Sample programming assignment test
# 2009, Dr. Muhammad Al-Hashimi#

#-----------------------------------------------
.data

ell:        .space 400
limit:      .word 25
a:          .space 400

#-----------------------------------------------
.text

main:
            la      $a0, ell
            lw      $a1, limit
            jal     s
            move    $s0,$v0

            ori     $2,$0,10            # system call: exit program
            syscall

#-------------------
inita:      li      $t0,2
ia_loop:    bgt     $t0,$a1,ia_break
            sll     $t1,$t0,2
            add     $t1,$t1,$a0         # effective address x+p
            sw      $t0,0($t1)
            add     $t0,$t0,1
            j       ia_loop
ia_break:   jr      $ra

#-------------------
s:          sub     $sp,$sp,32          # make room for $ra + 5 high-level vars + 2 procedure arguments
            sw      $ra,28($sp)
            sw      $s0,24($sp)         # track a
            sw      $s1,20($sp)         # track sqrtn
            sw      $s2,16($sp)         # track p
            sw      $s3,12($sp)         # track j
            sw      $s4,8($sp)          # track i
            sw      $s5,4($sp)          # save $a0
            sw      $s6,0($sp)          # save $a1

            move    $s5,$a0             # save arguments to free $a0-$a3 for children procedures
            move    $s6,$a1             # don't use $t since they are not preserved across children calls

            la      $s0,a               # initialize local high-level vars
            li      $s1,5

            move    $a0,$s0             # begin fill array call
            move    $a1,$s6
```

```
            jal     inita               # end fill array call

            li      $s2,2               # begin outer loop
s_loop1:    bgt     $s2,$s1,s_break1

            sll     $t0,$s2,2
            add     $t0,$t0,$s0         #effective address a+p
            lw      $t0,0($t0)          # load a[p]
            beqz    $t0,s_break3        # if fails, skip inner loop

            mul     $s3,$s2,$s2
s_loop2:    bgt     $s3,$s6,s_break3
            sll     $t1,$s3,2
            add     $t1,$t1,$s0         # effective address a+j
            sw      $zero,0($t1)        # zero a[j]
            add     $s3,$s3,$s2         # update j by p
            j       s_loop2             # inner loop

s_break3:   add     $s2,$s2,1           # update p
s_break2:   j       s_loop1             # end outer loop

s_break1:   move    $s4,$zero
            li      $s2,2               # re-init p
s_loop3:    bgt     $s2,$s6,s_break4

            sll     $t0,$s2,2
            add     $t0,$t0,$s0         # effective address x+p
            lw      $t0,0($t0)          # load a[p]
            beqz    $t0,s_break5        # if fails, skip iteration

            sll     $t2,$s4,2
            add     $t2,$t2,$s0         # effective address a+i
            sw      $t0,0($t2)          # x[i] <- a[p]
            add     $s4,$s4,1           # increment i

s_break5:   add     $s2,$s2,1
            j       s_loop3

s_break4:   move    $v0,$s4             # return i (number of primes)

            lw      $ra,28($sp)         # cut-paste from push sequence and adjust code
            lw      $s0,24($sp)
            lw      $s1,20($sp)
            lw      $s2,16($sp)
            lw      $s3,12($sp)
            lw      $s4,8($sp)
            lw      $s5,4($sp)
            lw      $s6,0($sp)
            add     $sp,$sp,32
            jr      $ra
```