```
# Consider some high-level-laanguage code:
#            int a, bb, c;
#            read(a, bb);
#            c = a+bb;
#            print(c);
#
#
#   WHAT A COMPILER MIGHT PRODUCE:
.data
      a: .word 0                # Declare a, bb, and c
      bb: .word 0
      c: .word 0
.text
   main:
       li $v0, 5
       syscall                  # Read and store a value for a
       la $t0, a
       sw $v0, 0($t0)
       li $v0, 5
       syscall                  # Read and store a value for b
       la $t0, bb
       sw $v0, 0($t0)
                                #
       la $t0, a                # Get the value of a into $t1
       lw $t1, 0($t0)
       la $t0, bb               # Get the value of b into $t2
       lw $t2, 0($t0)
                                #
       add $t3, $t1, $t2        # Add the values
       la $t0, c                # Store the result into c
       sw $t3, 0($t0)
                                #
       la $t0, c                # Get the value of c into $a0
       lw $a0, 0($t0)
       li $v0, 1
       syscall                  # Print the result
       li $v0, 10
       syscall                  # STOP
```

```
#  AN ASSEMBLY-LANGUAGE PROGRAMMER MIGHT PRODUCE:
.text
  main:
      li $v0, 5
      syscall                   # Read 1st value
      move $t0, $v0
      li $v0, 5
      syscall                   # Read 2nd value
      move $t1, $v0
      add $a0, $t0, $t1         # Add the values
      li $v0, 1
      syscall                   # Print the result
      li $v0, 10
      syscall                   # STOP
```

```
#    FIRST  PIECE  OF  CODE  COULD  ALSO  JUST  USE  DISPLACEMENTS  =>  FEWER
INSTRUCTIONS
.data
     a: .word 0                # Declare a, bb, and c
     bb: .word 0
     c: .word 0
.text
  main:
      li $v0, 5
      syscall                  # Read and store a value for a
      la $t0, a
      sw $v0, 0($t0)
      li $v0, 5
      syscall                  # Read and store a value for b
      sw $v0, 4($t0)
                               #
      la $t0, a                # Get the value of a into $t1
      lw $t1, 0($t0)
                               # Get the value of b into $t2
      lw $t2, 4($t0)
                               #
      add $t3, $t1, $t2        # Add the values
                               # Store the result into c
      sw $t3, 8($t0)
                               #
                               # Get the value of c into $a0
      lw $a0, 8($t0)
      li $v0, 1
      syscall                  # Print the result
      li $v0, 10
      syscall                  # STOP
```