```
### Programmsegment: Matrix-Vektor-Multiplikation ###

# Verwendung der Register:
#
#   $s0: n
#   $s1: i
#   $s2: k
#   $s3: index_A
#   $s4: index_x
#   $s5: index_b
#   $s6: Konstante 4
#   $f0: A[index_A] (= A[i][k])
#   $f1: x[index_x] (= x[k])
#   $f2: sum

.text

.globl main

main:

   lw $s0, dim      # n = Matrixdimension

   move $s1, $zero        # i = 0
   move $s3, $zero        # index_A = 0
   move $s5, $zero        # index_b = 0
   addi $s6, $zero, 4     # Konstante 4

loopi:

   bge $s1, $s0, endi        # while (i < n)

   move $s2, $zero        # k = 0
   mtc1 $zero, $f2        # sum = 0.0
   move $s4, $zero        # index_x = 0

loopk:

   bge $s2, $s0, endk      # while (k < n)

   la $t0, mat_A
   mult $s3, $s6
   mflo $t1
   add $t0, $t0, $t1         # berechne die Addresse von A[index_A]

   l.s $f0 ($t0)        # lade $f0 mit A[index_A]

   la $t0, vec_x
   mult $s4, $s6
   mflo $t1
   add $t0, $t0, $t1         # berechne die Addresse von x[index_x]
```

```
    l.s $f1 ($t0)              # lade $f1 mit x[index_x]

    mul.s $f3, $f0, $f1          # multipliziere A[index_A] * x[index_x]
    add.s $f2, $f2, $f3          # addiere Produkt zu sum

    addi $s3, $s3, 1            # index_A++
    addi $s4, $s4, 1            # index_x++
    addi $s2, $s2, 1            # k++

    j loopk

endk:
    la $t0, vec_b
    mult $s5, $s6
    mflo $t1
    add $t0, $t0, $t1                 # berechne die Addresse von b[index_b]

    s.s $f2, ($t0)             # speichere sum in b[index_b]

    mov.s $f12, $f2            # print_float(sum)
    li $v0, 2
    syscall

    la $a0, new_line                  # print_newline()
    li $v0, 4
    syscall

    addi $s5, $s5, 1           # index_b++
    addi $s1, $s1, 1           # i++

    j loopi

endi:

    li $v0, 10               # beende Programm
    syscall

### Datensegment ###

.data

dim: .word 5
mat_A: .float 2.0, 1.0, 4.0, 2.0, 6.0
       .float 1.0, 2.0, 2.0, 4.0, 2.0
       .float 4.0, 2.0, 6.0, 1.0, 3.0
       .float 4.0, 2.0, 6.0, 2.0, 1.0
       .float 4.0, 3.0, 2.0, 2.0, 3.0
vec_x: .float 1.0, 3.0, 5.0, 1.0, 3.0
vec_b: .float 0.0, 0.0, 0.0, 0.0, 0.0
new_line: .asciiz "\n"
```