```
.globl main
.text

j main

mult:                   # prozedur für die multiplikation
abs $t1, $a1            # speichert betrag vom zweitem argument in $t1
addi $t2, $zero, 1      # setzt $t2 zu 1 als counter
add $t3, $t2, $t3       # setzt $t3 zu 1, um $t2 leichter hochzuzählen
move $v0, $zero             # setzt v0 auf 0, da dieses vorher schon belegt wurde

multloop:               # loop in dem die multiplikation stattfindet
bgt $t2, $t1, loopexit  # wenn counter > zweites argument
add $v0, $v0, $a0       # addiere erstes argument zur endausgabe
add $t2, $t2, $t3       # addiere 1 zum counter
j multloop              # springt zum anfang der schleife

loopexit:
bge $a1, $zero, multexit        # wenn argument 2 >= 0, muss nicht negiert werden
neg $v0, $v0            # negiere endausgabe, falls notwendig

multexit:               # ende der prozedur
jr $ra                  # springt zurück zum aufruf

main:

la $a0,ret1             # gibt erste eingabeauforderung aus
li $v0,4
syscall
li $v0, 5               # liest einen int ein
syscall
move $t1, $v0           # speichert die eingabe im register $t1
la $a0,ret2             # gibt zweite eingabeauforderung aus
li $v0,4
syscall
li $v0, 5               # liest einen int ein
syscall
move $a1, $v0               # lädt zweiten eingegebenen wert als argument
move $a0, $t1           # lädt ersten eingegebenen wert als argument
jal mult                # ruft multiplikation auf

move $a0, $v0               # gibt den wert des produkts in v0 aus
li $v0, 1
syscall

exit:                   # beendet das programm


.data
ret1:   .asciiz "Geben sie die erste Zahl für die Multiplikation ein: "
ret2:   .asciiz "Geben sie die zweite Zahl für die Multiplikation ein: "
```