

```

.include "nios_macros.s"
.global _start
_start:
movia r2, AVECTOR /* Register r2 is a pointer to vector A */
movia r3, BVECTOR /* Register r3 is a pointer to vector B */
movia r4, N
ldw r4, 0(r4) /*Register r4 is used as the counter for loop iterations*/
add r5, r0, r0 /* Register r5 is used to accumulate the product */
LOOP: ldw r6, 0(r2) /* Load the next element of vector A */
ldw r7, 0(r3) /* Load the next element of vector B */

mul r8, r6, r7 /* Compute the product of next pair of elements */
add r5, r5, r8 /* Add to the sum */
addi r2, r2, 4 /* Increment the pointer to vector A */
addi r3, r3, 4 /* Increment the pointer to vector B */
subi r4, r4, 1 /* Decrement the counter */
bgt r4, r0, LOOP /* Loop again if not finished */
stw r5, DOT_PRODUCT(r0) /* Store the result in memory */
STOP: br STOP
N:
.word 6 /* Specify the number of elements */
AVECTOR:
.word 5, 3, -6, 19, 8, 12 /* Specify the elements of vector A */
BVECTOR:
.word 2, 14, -3, 2, -5, 36 /* Specify the elements of vector B */
DOT_PRODUCT:
.skip 4

```

1. `.global _start`: Об'явлення мітки `_start` як глобальної, вона буде доступна для виклику з інших частин програми.
2. `_start::` Мітка початку програми.
3. `movia r2, AVECTOR`: Завантаження в регістр `r2` адреси вектора А.
4. `movia r3, BVECTOR`: Завантаження в регістр `r3` адреси вектора В.
5. `movia r4, N`: Завантаження в регістр `r4` адреси за якою зберігається кількість елементів у векторах.
6. `ldw r4, 0(r4)`: Завантаження значення, яке зберігається за адресою `N`, в регістр `r4`. (кількість елементів у векторах)
7. `add r5, r0, r0`: Скидає регістр `r5` в 0, цей регістр буде використано в якості акумулятора для скалярного добутку.
8. `LOOP::` Мітка початку циклу.
9. `ldw r6, 0(r2)`: Завантажує значення, на яке вказує регістр `r2` (елемент вектора А), в регістр `r6`.
10. `ldw r7, 0(r3)`: Завантажує значення, на яке вказує регістр `r3` (елемент вектора В), в регістр `r7`.
11. `mul r8, r6, r7`: Розрахунок добутку елементів векторів А і В зі збереженням його у регістр `r8`.
12. `add r5, r5, r8`: Додавання значення з регістру `r8` до регістру акумулятору `r5`.
13. `addi r2, r2, 4`: Збільшення адреси в регістрі `r2` на 4 байти для переходу до наступного значення вектора А.

14. `addi r3, r3, 4`: Збільшення адреси в регістрі `r3` на 4 байти для переходу до наступного значення вектора `B`.
15. `subi r4, r4, 1`: Зменшення значення в регістрі `r4` на 1 для зменшення кількості ітерацій циклу.
16. `bgt r4, r0, LOOP`: Перевірка чи більше значення в регістрі `r4` ніж 0. Якщо так, програма повертається до мітки `LOOP` та продовжує виконання циклу.
17. `stw r5, DOT_PRODUCT(r0)`: Збереження значення з регістру `r5` за адресою `DOT_PRODUCT`.
18. `STOP`:: Мітка яка зупиняє виконання програми.
19. `br STOP`: перехід на мітку для нескінченного зациклювання – зупинка програми.

Registers	
Reg	Value
pc	0x00000044
zero	0x00000000
r1	0x00000000
r2	0x00000064
r3	0x0000007C
r4	0x00000000
r5	0x000001F4
r6	0x0000000C
r7	0x00000024
r8	0x000001B0
r9	0x00000000

У регістрі акумуляторі (`r5`) збережено результат скалярного добутку векторів `A` і `B`, який дорівнює 500 в десятковому вигляді та `1F4` в шістнадцятковому