1. Data je procedura u programskom jeziku C koja inicijalizuje članove niza cijelih brojeva **A**, dužine **N**, na 0. Napisati odgovarajući MIPS kod.

```
Clear(int A[], int N)
{
    int i;
    for(i=0; i<N; i++)
    A[i]=0;
}
```

Prilikom prevođenja koda iz programskog jezika C u MIPS kod pridržavamo se sljedeća tri generalna koraka:

**I. Alokacija registara za programske promjenljive**

Pošto se ulazni argumenti procedura prosljeđuju preko registara $4, $5, $6 i $7, a naša funkcija ima dva ulazna argumenta, onda su tim promjenljivima dodijeljeni registri $4 i $5, tj. A↔$4 i N↔$5. U okviru same procedure jedina deklarisana promjenljiva je **i**, pa njoj možemo dodijeliti registar $15.

**II. Pisanje koda tijela procedure**

```
        move $15, $0        # i=0;
Loop:   slt $8, $15, $5     # Ako je i<N onda $8=1
        beq $8, $0, Exit    # Ako je $8=0 izaći iz petlje
        muli $16, $15, 4    # $16=4*i - u $16 prava vrijednost pomjeraja
        add $16, $16, $4    # U $16 adresa i-tog člana niza A[i]
        sw $0, 0($16)       # A[i]=0
        addi $15, $15, 1    # i=i+1
        j Loop              # go to na Loop
Exit:
```

**III. Čuvanje sadržaja registara tokom poziva procedura**

U proceduri mijenjamo sadržaj registara $15, $8 i $16, pa se prije bilo kakve promjene njihov sadržaj mora smjestiti na stek radi čuvanja njihovih originalnih vrijednosti. Registar $29 sadrži pokazivač na stek, pa se on mora umanjiti za 3*4=12 bajta, kako bi se napravilo prostora za ova tri registra:

```
addi $29, $29, -12      # Stek raste od viših adresa ka nižim i otuda -
```

Čuvanje starih vrijednosti u registrima $15, $8 i $16:

```
sw $15, 0($29)      # $15 na stek
sw $8, 4($29)       # $8 na stek
sw $16, 8($29)      # $16 na stek
```

Pošto naša procedura ne poziva nijednu drugu proceduru onda nema potrebe da se $31 smješta na stek.
Nakon završetka procedure vraćamo sa steka stare vrijednosti registara $15, $8 i $16 i usklađujemo pokazivač na stek:

```
lw $15, 0($29)      # sa steka u $15
lw $8, 4($29)       # sa steka u $8
lw $16, 8($29)      # sa steka u $16
addi $29, $29, 12   # usklađivanje steka
```

Posljednji korak je vraćanje u proceduru koja je pozvala našu proceduru, i to na instrukciju prvu posle poziva naše procedure:

```
                jr $31       # vraćanje u pozivajuću proceduru.
```

Dakle, **ukupan kod** bi bio:

```
Clear:  addi $29, $29, -12
        sw $15, 0($29)      # $15 na stek
        sw  $8, 4($29)      # $8 na stek
        sw  $16, 8($29)     # $16 na stek
        add $15, $0, $0     # i=0;
Loop:   slt $8, $15, $5     # Ako je i<N onda $8=1
        beq $8, $0, Exit    # Ako je $8=0 izađi iz petlje
        muli $16, $15, 4    # $16=4*i - u $16 prava vrijednost pomjeraja
        add $16,$16,$4      # U $16 adresa i-tog člana niza A[i]
        sw $0, 0($16)       # A[i]=0
        addi $15, $15, 1    # i=i+1
        j   Loop            # go to na Loop
Exit:   lw  $15, 0($29)     # sa steka u $15
        lw  $8, 4($29)      # sa steka u $8
        lw  $16, 8($29)     # sa steka u $16
        addi $29, $29, 12   # usklađivanje steka
        jr  $31             # vraćanje u pozivajuću proceduru.
```

**Napomena:** Korišćenje instrukcije **lw** za učitavanje **int** promjenljivih, koje obično zauzimaju 4 bajta. **lb**-load byte, **lh**-load halfword (2 bajta), **ld**-load double-word (8 bajta).