## Problema 3.

A partir del següent programa escrit en el llenguatge d'alt nivell C, que calcula la suma dels elements d'un vector, escriure un programa en assemblador que implementi la mateixa funcionalitat. Considereu que no ens interessa el valor final de les variables i i partial_sum. Per això, per optimitzar el codi heu de fer servir registres del processador per emmagatzemar els seus valors.
Suposeu que al programa escrit en C s'ha definit la variable V com a vector d'enters de 32 bits i la variable sum com a nombre enter de 32 bits i que es troben emmagatzemats a partir de les adreces simbòliques de memòria V i sum, respectivament.

```
int N = 6;
int sum = 0;
int partial_sum = 0;
int i = 0;
while ( i < N) {
        partial_sum = partial_sum + V[i];
        i = i + 1;
}
sum = partial_sum;
```

**Opció 1:**

```
N = 6;
        LI R1, 6
        LLA R2,N
        SW R1, 0(R2)            ; [N] ← 6 ; R1 guarda el valor de la variable N
sum = 0;
partial_sum = 0;
        LI R3, 0                ; R3  guarda el valors  de les variables sum i
                                ; partial_sum, ja les actualitzarem a la
                                ; memòria quant sortim del bucle.
 i = 0;
        LI R2, 0                ; R2  guarda els canvis  de la variable i, com amb
                                ; les variables anteriors, l'actualitzarem a la
                                ; memòria quant sortim del bucle
        LLA R4, V               ; R4 té l'adreça inicial de l'array (vector) V
While (i < N ){
while:
        BGE R2, R1, end
        partial_sum = partial_sum + V[i];
        LW R5, 0(R4)            ;R5 ← [mem (R4 +0)], R4 conté  l' adreça  de
                                ;l'element amb índex i de l'array V
```

```
        ADD R3, R3, R5
        i = i + 1;
        ADD R2, R2, 1          ; R2 ← R2 +1, índex del següent element de V
        ADD R4, R4, 4          ; R4 ← R4 +4, adreça del següent element (amb
                               : índex i+1) del  array V (perdem l'adreça origen
                               ; del array V).
        J while                ; Tornem a la condició del while
}
end:
sum = partial_sum;
        LLA R1,sum             ; Actualitzem el valor de les variables i, sum i
                               ; partial_sum en memòria.
        SW R3, 0(R1)
        LLA R1,  partial_sum
        SW R3, 0(R1)
        LLA R1,  i
        SW R2, 0(R1)
```

## Opció 2: Més optima

```
N = 6;
        LI R1, 6
        LLA R2,N
        SW R1, 0(R2)           ; [N] ← 6 ; R1 guarda el valor de la variable N
sum = 0;
partial_sum = 0;
        LI R3, 0               ; R3  guarda el valors  de les variables sum i
                               ; partial_sum
 i = 0;
        MV R2, R1              ;R2  guarda els canvis  de la variable i, aquesta
                               ; variable sempre sortirà del bucle amb el valor de N
        LLA R4, V              ; R4 té l'adreça inicial de l'array (vector) V. En el bucle
                               ; contindrà l'adreça  de cada element i de V.
        SLLI R1, R1, 2
        ADD R1,R4,R1           ; R1 s'utilitza per guardar l'adreça final  de l'array V.
                               ; adreça de V[N] = adreça inicial de V + 4*N.
                               ; Si adreça  Base de V és 1000, N=6 l'adreça de
                               ; V[6] serà: 1000 + (4 * 6) = 1024
While (i < N ){
while:
        BGE R4, R1, end        ; Comparem de l'adreça de l'element actual de
                               ; l'array V es menor que l'adreça màxima  d'aquest
                               ; array
        partial_sum = partial_sum + V[i];
        LW R5, 0(R4)           ;R5 ← [mem (R4 +0)],  R4 conté  l' adreça  de
                               ; l'element amb índex i de l'array V
```

```
                                ; Si dir. Base de V és 1000, la dir. de V[2] serà:
                                ; 1000 + (4(mida dada sencera) *2) = 1008
                                ; → el desplaçament fins V[2] es de 8.
        ADD R3, R3, R5
        i = i + 1;
        ADD R4, R4, 4           ; R4 ← R4 + 4, desplaçament en bytes fins el
                                ; següent element de V
        J while                 ; Tornem a la condició del while
}
end:
sum = partial_sum;
        LLA R1,sum              ; Actualitzem el valor de les variables i, sum i
                                ; partial_sum en memòria.
        SW R3, 0(R1)
        LLA R1,  partial_sum
        SW R3, 0(R1)
        LLA R1,  i
        SW R2, 0(R1)
```