09/03/2021

Alumna: Mazza Reta, Tiziana Nicole - Padrón 101715 - Firma: Tiziana Mazza

3)

a) En el ejercicio (1. a) usamos subrutinas que se accede por un call en tiempo de ejecución. Esto quiere decir que, en ese momento, pedimos muchos saltos entre calls y jumps (pero en el (1.a) que hice yo solo lo hacemos 1 vez) Se tarda más en la ejecución. Como la subrutina ocupa solo el espacio de código 4 veces → no ocupa mucho espacio. Usar la ④ de 1 vez.

En el ejercicio (1.b) usamos macros, que se acceden en tiempo de ensamblado. Esto hace que, antes de ejecutarse el código, se haya "copiado y pegado" la macro la cantidad de veces que se la nombró. Por lo tanto, ocupa muchísimo más espacio, cuando se la llama más de 4 veces o la macro, el código. Por el contrario, hace que vaya más rápido la ejecución que una subrutina, ya que no debe pegar saltos.

b)
- .org 3000 → dirección de inicio del código
- .dwb 1000 → guarda 100 lugares para un array
- V: equ 5000 → la etq V vale 5000
- x: 5000 → x es 5000 (no variable).

Todas las instrucciones son de 32 bits (8 bytes).
Cada instrucción genera 32 bits de código de máquina, según el formato de instrucción. (op, ops, d)

→ Ahora, todas que son directivas para el ensamblador no se generan código de máquina con el formato de instrucciones

↳ directivas:
- .org 3000
- .dwb
- V: equ 5000.

⇒ estas líneas de código no generan código de máquina ∴ no ocupan bytes.

Asamblea