Guía de ejercicios # 9: Arreglos y recorridos

Organización de computadoras

UNQ

Programar recorrido de arreglos

Los ejercicios de esta sección te permitirán entender la programación de recorridos de arreglos.
Es **importantísimo descomponer los problemas**. En general suele ser útil pensar una subrutina que se encargue de procesar un solo elemento del arreglo y sea llamada desde la rutina principal.

1. Implementar la rutina `sumaElementos` (y completar el campo modifica):

`sumaElementos`	
Requiere	Un arreglo de 5 valores en $BSS(16)$ almacenado a partir de la celda `0x3333`
Modifica	??
Retorna	En R6 la suma de los elementos del arreglo

2. Implementar la rutina `cantElementos` (y completar el campo modifica):

`cantElementos`	
Requiere	Un arreglo de valores en $BSS(16)$ almacenado a partir de la dirección que indica R0, y que finaliza con el primer elemento cuyo valor es `0x0000`
Modifica	??
Retorna	En R6 la cantidad de elementos del arreglo

   Nota: R0 tiene la dirección del primer elemento del arreglo y [R0] el valor de dicho elemento.

3. Escribir la rutina `copiarArreglo` en función de su documentación (y completar el campo modifica):

`copiarArreglo`	
Requiere	Un arreglo de valores en $BSS(16)$ almacenado a partir de la celda `8400` y que finaliza con el primer valor `FFFF`
Modifica	??
Retorna	Una copia del arreglo original a partir de la celda `9400`

4. Escribir la rutina `aplicarAbsolute` (y completar el campo modifica):

`aplicarAbsolute`	
Requiere	Un arreglo de valores en SM(16) almacenado a partir de la celda `0x4486`, y cuya longitud está en la celda `0x4485`.
Modifica	??
Retorna	El mismo arreglo con sus elementos en valor absoluto

   Nota: usar la rutina absolute de la práctica anterior.

5. Simule la ejecución de la rutina `aplicarAbsolute` sobre el siguiente mapa de memoria:

	...
4485	0004
4486	000A
4487	FFFF
4488	8000
4489	00FF
	...

6. Modifique la rutina `aplicarAbsolute` para que reciba como parámetro la dirección inicial del arreglo en R0.

7. En una fábrica de ventanas se codifican los pedidos en cadenas de 16 bits, y en caso de que la ventana esté pintada o lleve vidrio de seguridad es necesario usar un embalaje distinto. Se pide escribir la rutina `necesitaEmbalajePremium` que determine si el pedido es uno de esos casos. Completar además el campo **Modifica**.

`necesitaEmbalajePremium`	
Requiere	En R5 un código de pedido de ventana, donde el **bit 3** indica si debe estar pintada y el **bit 9** indica si lleva el vidrio de seguridad.
Modifica	??
Retorna	un 1 en R7 si el pedido que está en R5 requiere un embalaje *premium*, es decir, si se trata de una ventana pintada o con vidrio de seguridad.

   Nota: Tener en cuenta que los demás bits también representan distintas características, pero para el ejercicio solo nos interesan esos dos.

8. Escriba la rutina `cantEmbalajeComun` a partir de su documentación y complete el campo modifica.

cantEmbalajeComun	
Requiere	Un arreglo de pedidos a partir de la celda `0x0001`, cuya longitud está en en la celda `0x0000`
Modifica	??
Retorna	en R6 la cantidad de pedidos que **no necesitan embalaje *premium***

9. Implementar una rutina de test para verificar el funcionamiento de la rutina `cantEmbalajeComun`.

10. Contando con las nuevas herramientas incorporadas a Q en esta unidad, volver a escribir la rutina `mapearCeldas` de la guía 6 (ejercicio 5). Utilizando la rutina `esPar` de la misma guía, `mapearCeldas` debe guardar un 0 o un 1 en las celdas `0x0000` a `0x0005` según si los números de las celdas `0xF000` a `0xF005` son pares o no.

Bajo nivel: ¿Cómo funciona?

Los ejercicios de esta sección están pensados para que entiendas la necesidad de un modo indirecto y profundices lo que entendés con respecto a la ejecución de programas.

11. Considerando el siguiente mapa de memoria y valor de los registros:

	...
0000	0001
0001	0004
0002	0000
0003	0002
0004	0001
	...

Registro	Valor almacenado
R0	0x0001
R1	0x0003

Describir el **efecto** para cada instrucción:

(a) `MOV [R0], [[0x0003]]`

(b) `MOV [[0x0003]], [R0]`

(c) `MOV [[0x0003]], [[0x0001]]`

(d) `MOV [[0x0001]], [R1]`

Nota: El efecto se denota `dest <-- valor`, donde `dest` es el registro o posición de memoria que se termina modificando.

12. Considerando la instrucción `MOV R1, [[0x0000]]`, ensamblada a partir de la celda 1A40, indicar:

(a) Luego de la búsqueda de la instrucción, ¿cuál es el contenido de los registros IR y PC?

(b) Sabiendo que en la celda 0000 se encuentra el dato 0A44, ¿qué celdas se acceden en cada etapa del ciclo de ejecución?

13. Considerando la instrucción `MOV [R2], [R1]`, ensamblada a partir de la celda 1A40, indicar:

(a) Luego de la búsqueda de la instrucción, ¿cuál es el contenido de los registros IR y PC?

(b) Sabiendo que el registro R1 contiene el dato F57A y R2 contiene el dato C32F, ¿qué celdas se acceden en cada etapa del ciclo de ejecución?

14. Considerando la instrucción `MOV [[0x0000]], [[0x0001]]`, ensamblada a partir de la celda 1A40, indicar:

(a) Luego de la búsqueda de la instrucción, ¿cuál es el contenido de los registros IR y PC?

(b) Sabiendo que en las celdas 0000 y 0001 se encuentran los datos 0A44 y B312 respectivamente, ¿qué celdas se acceden en cada etapa del ciclo de ejecución?

15. Considerando la instrucción `ADD [R2], [R1]`, ensamblada a partir de la celda 1A40, indicar:

(a) Luego de la búsqueda de la instrucción, ¿cuál es el contenido de los registros IR y PC?

(b) Sabiendo que el registro R1 contiene el dato F57A y R2 contiene el dato C32F, ¿qué celdas se acceden en cada etapa del ciclo de ejecución?

16. Considerando la instrucción `ADD [[0x0000]], [[0x0001]]`, ensamblada a partir de la celda 1A40, indicar:

(a) Luego de la búsqueda de la instrucción, ¿cuál es el contenido de los registros IR y PC?

(b) Sabiendo que en las celdas 0000 y 0001 se encuentran los datos 0A44 y B312 respectivamente, ¿qué celdas se acceden en cada etapa del ciclo de ejecución?