Guía de ejercicios # 8: Repeticiones y Máscaras (Q5)

Organización de computadoras

UNQ

Repeticiones

Los ejercicios de esta sección te permitirán entender el concepto de iterar y cómo se implementa en la arquitectura Q4.

1. Escribir **y documentar** una rutina que calcule la multiplicación de los valores que se encuentran en los registros R5 y R6 respectivamente pero sin usar la instrucción MUL.

2. Realizar una **prueba de escritorio** con valores que prueben la rutina del ejercicio anterior.

3. Escribir **y documentar** una rutina que calcule la división entera entre los valores de R0 y R1 respectivamente sin usar la instrucción DIV.

4. Escribir **y documentar** una rutina que cuente la cantidad de dígitos "1" en la cadena almacenada en R6.

5. Escribir **y documentar** una rutina que calcule el factorial del valor almacenado en R5. Dicho valor está representado en BSS.

Operaciones lógicas bit a bit

6. ¿Cuál es el resultado de las siguientes operaciones?

   (a) `1101 AND 0111 = ?`

   (b) `NOT 0100 = ?`

   (c) `((1010 AND 1100) OR 0101) XOR 1100 = ?`

7. Complete las operaciones lógicas dada una cadena de bits formada por ($x_7x_6x_5x_4x_3x_2x_1x_0$).

   Por ejemplo:

   $$\begin{array}{ll} \texttt{OR} & x_7x_6x_5x_4x_3x_2x_1x_0 \\ & 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0 \\ \hline & 1\ x_6 1\ x_4 1\ x_2 1\ x_0 \end{array}$$

   (a) $x_7x_6x_5x_4x_3x_2x_1x_0$`OR 11111000 = ?`

   (b) $x_7x_6x_5x_4x_3x_2x_1x_0$`AND 10001111 = ?`

   (c) `((`$x_7x_6x_5x_4x_3x_2x_1x_0$` AND 10101111) OR 11110000) XOR 00011110 = ?`

Programas en Q5

Los ejercicios de esta sección permiten entender la motivación para el uso de máscaras y lo necesario para que la arquitectura Q lo implemente.

8. Implementar la rutina `absolute` siguiendo su documentación:

`absolute`	
Requiere	Un valor en SM(16) en el registro R0
Modifica	COMPLETAR
Retorna	El valor absoluto en R0, donde el primer bit se cambió por un 0

9. Implementar la rutina `xor` en función de su documentación:

`xor`	
Requiere	Dos cadenas de 16 bits en R6 y R7
Modifica	COMPLETAR
Retorna	En R7 el OR Exclusivo (bit a bit) entre R6 y R7

10. Programar una rutina de test para verificar el funcionamiento de la rutina `xor`.

11. Implementar la rutina `invImp` que dada una cadena de 16 bits en R1, invierta el valor de las **posiciones impares**. Utilice máscaras y operaciones lógicas. Para resolver este ejercicio es necesario **descomponer el problema**.

12. Implementar la rutina `opuesto`, que calcule el opuesto aditivo del número almacenado en el registro R2 sin usar SUB. Dicho número está representado en $CA2(16)$.

13. Un registro meteorológico es aquel donde sus bits indican la precipitación en cada día, para una estación meteorológica determinada, durante 14 días (los 14 bits de la derecha). En el sistema occidental se indica con 0 si llovió y con un 1 el caso contrario. En el sistema oriental se indica al inverso. Por ejemplo, la cadena `0011001100110011` se convierte así: `0000110011001100` (notar que los primeros 2 bits siempre están en cero).

    Implementar la rutina `occidentalAOriental` según la documentación:

occidentalAOriental	
Requiere	En R3 un registro meteorológico en sistema OCCIDENTAL
Modifica	COMPLETAR
Retorna	El registro R3 convertido al sistema ORIENTAL

14. Implementar la rutina `diasDeLluvia` según la documentación:

diasDeLluvia	
Requiere	Un registro meteorológico en sistema oriental en R6
Modifica	COMPLETAR
Retorna	En R2 La cantidad de días de lluvia registrados

Pista: puede usar la rutina `desplazarIzq` que tiene la siguiente documentación:

desplazarIzq	
Requiere	Una cadena en R0
Modifica	—
Retorna	En R1 desplaza los bits de la cadena contenida en R0 un lugar hacia la izquierda

15. La siguiente es la codificación de permisos de acceso sobre archivos en un sistema Linux:

    - Con 3 bits se indica:
      (a) Permiso lectura (r)
      (b) Permiso escritura (w)
      (c) Permiso ejecución (x)
    - Con 3 cadenas de 3 bits cada una, se describen permisos de usuario, grupo y otros.

La cadena resultante respeta el siguiente formato: `rwxrwxrwx`, donde cada subcadena de 3 bits corresponde al usuario, al grupo y a otros respectivamente. Por ejemplo, la cadena `111111111` le da todos los permisos a todos.
Implementar la rutina `groupWriting` que, dada una cadena que codifica los permisos de acceso a determinado archivo, determine si otro usuario del grupo lo puede escribir. Dicha cadena esta almacenada en R4, y el programa debe poner un 1 en R5 si es posible, y 0 en caso contrario.
Para resolver este ejercicio es necesario **descomponer el problema**.