Guía de ejercicios # 9: Máscaras

Organización de Computadoras 2018

UNQ

Los objetivos de esta práctica son:

- Entender la motivación para el uso de máscaras y lo necesario para que la arquitectura Q lo implemente.
- Incorporar prácticas que aseguren la calidad del trabajo de programación
- Relacionar con conceptos anteriores.

Los ejercicios marcados con ★ son un conjunto minimal para comprender los temas tratados en esta práctica. Para resolver esta práctica se aconseja consultar el apunte de la materia *Implementación de máscaras*, disponible en `http://orga.blog.unq.edu.ar/descargas/`

1 Operaciones lógicas bit a bit

- Ejemplo

```
AND  0101        XOR  0101
     0011             0011
     ----             ----
     ????             ????
```

Sería:

```
AND  0101        XOR  0101
     0011             0011
     ----             ----
     0001             0010
```

1. ¿Cuál es el resultado de las siguientes operaciones? ★

(a)
```
AND  1101
     0111
     ----
     ?
```

(b)
```
OR   0101
     1001
     ----
     ?
```

(c)
```
NOT  0100
     ----
     ?
```

(d)
```
XOR  1011
     1110
     ----
     ?
```

(e)
```
AND  1010
     1100
     ----
     ?
OR   0101
     ----
     ?
XOR  1100
     ----
     ?
```

2. Complete con el operador adecuado (AND, OR, XOR, NOT) en las operaciones

- Ejemplo:

  1001...1001 = 0000 → **1001 XOR 1001** = 0000

(a) 1000 ... 1011 = 1011
(b) 1011 ... 1000 = 1000
(c) 1101 ... 1001 = 0100
(d) 1111 ... 0011 = 0011
(e) .... 0011 = 1100

3. Complete las operaciones logicas dada una cadena de bits formada por ($x_7x_6x_5x_4x_3x_2x_1x_0$)

- Ejemplo:

```
OR   x7 x6 x5 x4 x3 x2 x1 x0
     1  0  1  0  1  0  1  0
     ------------------------
     1  x6 1  x4 1  x2 1  x0
```

(a)
```
OR   x7 x6 x5 x4 x3 x2 x1 x0
     1  1  1  1  1  0  0  0
     ------------------------
     ?
```
★

(b)
```
AND  x7 x6 x5 x4 x3 x2 x1 x0
     1  0  1  0  1  0  1  0
     ------------------------
     ?
```
★

(c)
```
AND  x7 x6 x5 x4 x3 x2 x1 x0
     1  0  0  0  1  1  1  1
     ------------------------
     ?
```

(d)
```
XOR  x7 x6 x5 x4 x3 x2 x1 x0
     1  0  1  0  1  0  1  0
     ------------------------
     ?
```

(e)
```
XOR  x7 x6 x5 x4 x3 x2 x1 x0
     0  0  0  0  1  1  1  1
     ------------------------
     ?
```
★

(f)
```
OR   x7 x6 x5 x4 x3 x2 x1 x0
     1  0  0  0  0  0  0  0
     ------------------------
     ?
```

(g)
```
OR   x7 x6 x5 x4 x3 x2 x1 x0
     1  0  0  0  0  0  0  0
     ------------------------
          ?
AND  1  1  1  1  0  0  0  0
     ------------------------
          ?
XOR  0  0  0  1  1  1  1  0
     ------------------------
          ?
```

(h)

	$x_7x_6x_5x_4x_3x_2x_1x_0$
AND	1 0 1 0 1 1 1 1
	?
OR	1 1 1 1 0 0 0 0 ★
	?
XOR	0 0 0 1 1 1 1 0
	?

2 Programas que aplican máscaras

4. Implementar la rutina `absolute` que dada una cadena de 16 bits en R0, ponga un 0 en el primer bit, dejando el resto sin modificar. Utilice máscaras y operaciones lógicas.

5. ★ Implementar la rutina `xor` en función de su documentación

```
;REQUIERE dos valores de 16 bits en R6 y R7
:MODIFICA el registro R4
;RETORNA en R7 el OR exclusivo (bit a bit) entre
        R6 y R7
```

6. 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.

7. Implementar la rutina `segbit` que dada una cadena de 16 bits en R1, ponga un 1 en el segundo bit, dejando el resto sin modificar. Utilice máscaras y operaciones lógicas.

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

9. Escribir una rutina que determine si la cadena en R0 es impar, en cuyo caso invocar a la rutina esImpar. **Documentar**

10. ★ 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:

```
;REQUIERE en R3 un registro meteorológico en
sistema OCCIDENTAL
:MODIFICA ?
;RETORNA El registro R3 convertido al sistema
ORIENTAL
```

11. Implementar la rutina `contarParesEImpares` según la documentación:

```
;REQUIERE en la celda [cccc] un número en CA2(16)
:MODIFICA ?
;RETORNA Suma un 1 a R3 o a R4 si el número
analizado es par o impar respectivamente.
```

12. ★ Implementar la rutina `copy` en función de su documentación:

```
;REQUIERE un valor de 16 bits en R1
:MODIFICA ?
;RETORNA en los 8 bits menos significativos de R2
el byte mas significativos del valor almacenado en
R1
```

    Utilizar el programa para copiar el byte más significativo de la celda 0348 en el registro R1.

13. Implementar la rutina `equals` que determine si el contenido de R2 y R3 son iguales, usando operaciones lógicas ★

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

```
;REQUIERE Un registro meteorologico en sistema
oriental en R6
:MODIFICA ?
;RETORNA en R2 La cantidad de días de lluvia
registrados.
```

    Pista: puede usar la rutina `desplazarIzq`

15. La siguiente es la documentación de la rutina `espar`:

```
;REQUIERE una cadena de 16 bits en el registro R6
:MODIFICA el registro R5
;RETORNA un 1 en registro R1 si R6 es par.  un 0 en
caso contrario
```

    Implementar un programa que, **usando la rutina anterior**, sume 30 al valor de R3 si la celda `[CCCC]` contiene un número par. En caso contrario debe sumar 70 al valor de R4

16. Implementar un programa que si las celdas `CCCC` y `CCCD` contienen números impares, le reste 0x0001 a ambas. Utilice la rutina `espar`.

17. 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 se describen permisos de usuario, grupo y otros

    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. ★

3 Controlar los programas

Para corroboar que las rutinas de la sección anterior fueron implementadas correctamente, se deben implementar

programas de *test* o pruebas para hacer control de calidad sobre las mismas.

En cada caso se deben proveer valores que cumplan lo requerido por la rutina (campo **requiere**) y luego de invocarla se debe comparar el resultado con el valor esperado, poniendo en R0 el valor `F000` en caso de exito o el valor `FFFF` en caso de fallo.

- Ejemplo Considerando la documentación de la rutina `avg`:

```
;REQUIERE dos valores de 16 bits en R6 y R7
:MODIFICA --
;RETORNA en R6 el promedio ENTERO entre R6 y R7
```

Un test válido podría ser:

```
testAvg: MOV R6, 0x0008
         MOV R7, 0x000A
         CALL avg
         CMP R6, 0x0009
         JE funcionabien
         MOV R0, 0xFFFF
         JMP fin
funcionabien: MOV R0, 0xF000
fin: RET
```

18. Implementar un programa que haga un control de calidad sobre la rutina `esPar`
19. ★Implementar un programa que haga un control de calidad sobre la rutina `xor`
20. Implementar un programa que haga un control de calidad sobre la rutina `absolute`
21. ★Implementar un programa que haga un control de calidad sobre la rutina `invimp`
22. Implementar un programa que haga un control de calidad sobre la rutina `segbit`
23. Implementar un programa que haga un control de calidad sobre la rutina `opuesto`
24. ★ Implementar un programa que haga un control de calidad sobre la rutina `equals`
25. Implementar un programa que haga un control de calidad sobre la rutina `diasDeLluvia`
26. ★ Implementar un programa que haga un control de calidad sobre la rutina `groupWriting`

4 Ejecución de instrucciones

Para cada instrucción de las que se mencionan abajo:

1. Ensamblar la instrucción
2. Completar un cuadro de uso de buses como el que sigue, donde se indica la información que viaja por los buses en cada etapa del ciclo de ejecución.

Etapa	Bus de control	Bus de dir.	Bus de datos

- En la columna **etapa** se debe indicar si es busqueda de instrucción (B.I), de Operandos (B.O.) o almacenamiento de resultados (A.R)
- En la columna de **bus de control** se debe indicar el estado de las líneas L (Lectura) y E (Escritura).
- En la columna del **bus de direcciones** se debe indicar la dirección de la celda accedida.
- En la columna del **bus de datos** se debe indicar el contenido que se quiere escribir en, o que se leyó de, la celda accedida.

Nota: Asumir que cada instrucción está ensamblada a partir de la celda `0xA000`.

27. `JLEU salto`, sabiendo que `salto` es la etiqueta de una instrucción en la celda `0xA077` ★
28. `AND R5, R7`, donde `R5=0x5555` y `R7=0xA5A5`
29. `NOT [0x0076]`, donde la celda `0x0076` tiene el valor `0x5555`
30. `OR [0x0076], [0xA890]`, donde la celda `0x0076` tiene el valor `0x5555` y la celda `0xA890` tiene el valor `0xBBBB`
31. `JG salto`, sabiendo que `salto` es la etiqueta de una instrucción en la celda `0x7700`
32. `JMP [R5]`, sabiendo que `R5=0x2121` ★
33. `JMP [0x2121]`, sabiendo que la celda `0x2121` tiene el valor `0xAAAA`