Ejercicios resueltos

Máscaras para determinar el valor de uno o más bits

Este método consiste en conservar el valor (0 o 1) de ciertos bits "que me interesan" y hacer 0 (cero) el resto.

Ejemplo 1:

Supongamos que se quiere determinar si el valor almacenado en el registro AL es par.
Se puede razonar que un valor será par si su bit 0 (el menos significativo) es 0 e impar si es 1.
Si se resolviera el problema mediante "fuerza bruta" debería comparar AL con todos los valores pares posibles (0, 2, 4, 6, ..., 252 y 254) y para cada comparación, un salto "por igual" (JZ o JE). En 256 valores posibles, la mitad son pares y por lo tanto corresponderían 128 comparaciones y saltos (el equivalente en ensamblador de *If - Then*). ¡Tiene que haber algo mejor!
Supongamos que pudiéramos hacer que todos los bits, *menos el bit 0*, se convirtieran en 0, ¿Cuántos números diferentes resultarían de esta operación?

- 00000000 b = 0 d
- 00000001 b = 1 d

¡Nada más que dos! ¡Esto implicaría una única comparación y salto!
Aclarando un poco más: Si enmascaramos un valor par como describimos resultado será 0, mientras que si lo que enmascaramos es un byte impar, obtendremos 1.
¿Y cómo podemos conservar el bit 0 y hacer 0 los demás?
Respuesta: *AND* ¿Cómo? ¿Contra qué valor?

- AND AL, 00000001b

Recordemos que tanto 0 como 1 *AND* 0 = 0, por lo que los bits 1 a 7 serán 0; al mismo tiempo el *AND* entre 1 y el bit 0 conservará el valor de este bit.
Veamos un fragmento de código en el que se determina si el valor contenido en AL es par o no. Si lo es, el registro BL almacenará un 1; en caso contrario, el valor de BL será 0:

```
 XOR BL, BL      ; Inicializa BL en 0
 AND AL, 1b      ; Enmascara AL conservando el bit 0
 CMP AL, 1b      ; Compara el resultado con 1
 JZ Fin          ; Si es igual (o sea impar) salta a Fin
 INC BL          ; Incrementa BL en 1, pasando a ser 1
Fin:             ; Etiqueta o rótulo Fin
 MOV AH, 4Ch     ; Las dos instrucciones para finalizar
 INT 21h
```

Ejemplo 2:

Se necesita determinar si un byte almacenado en el registro BH tiene sus bits 2 y 5 con diferente valor, es decir bit 2 = 0 y bit 5 = 1, o bien bit 2 = 1 y bit 5 = 0.
Estudiando los valores de 0 a 256 vemos que los primeros, desde 0 a 3, *no* cumplen con la condición, ya tanto el bit 2 como el 5 valen 0; los siguientes 4 valores, de 4 a 7, sí tienen sus bits 2

y 5 diferentes (1 y 0 respectivamente). Nuevamente vemos que la mitad de los valores de 0 a 255 cumplen el criterio y la otra mitad, no. Esto implicaría realizar también 128 comparaciones con sus correspondientes saltos condicionales.
Una manera de simplificar este problema consiste en enmascarar el byte conservando los valores de los bits 2 y 5 y haciendo 0 los del resto de los bits. ¿Cuántos valores "enmascarados" diferentes resultarían?

- 00000000 b = 0 d = 00h
- 00000100 b = 4 d = 04h
- 00100000 b = 32 d = 20h
- 00100100 b = 36 d = 24h

Resultan únicamente 4 valores, de los cuales nos interesan el segundo y el tercero.
¿Y cómo logramos esto? Haciendo un AND entre BH y 00100100b.
En el siguiente código, si el valor de BH tiene los bits 2 y 5 diferentes, se almacenará un 1 en CL y en caso contrario, un 0.

```
 XOR CL, CL   ; Inicializa CL en 0
 AND BH, 24h  ; Enmascara AL conservando los bits 2 y 5
 CMP BH, 00h  ; Compara el resultado con 0(bit2 = bit5 = 0)
 JZ Fin       ; Si es igual, salta a Fin
 CMP BH, 24h  ; Compara el resultado con 24h (bit2=bit5= 1)
 JZ Fin       ; Si es igual, salta a Fin
 INC CL       ; Incrementa BL en 1, pasando a ser 1
Fin:          ; Etiqueta o rótulo Fin
 MOV AH, 4Ch  ; Las dos instrucciones para finalizar
 INT 21h
```