8.10.2020

PRÁCTICA

2.64 `int any_odd_one (uint32_t val);` → library . stdint.h
un entero que mide 32 bits
(si lo considero como una secuencia de bytes

→ devuelve 1 (verdadero) si val tiene algún bit impar a 1
→ devuelve 0 si val tiene todos sus bits impares a 0.
↳ falso.

Ejemplo ↓ Están representados como 8 bits pero en realidad secuencia es de 32 bits

0	0	0	0	0	1	0	1
7	6	5	4	3	2	1	0

= 5 → devuelve 0

0	0	0	0	0	1	(1)	0
7	6	5	4	3	2	1	0

= 6 → devuelve 1

- Una implementación : desplazo los bits k (impar) posiciones hacia la derecha y rellena con 0s.

```
int any_odd_one ( uint32_t val){
    return ( (val >>1) & 1 || (val >>3) & 1 || ... || (val >>31) & 1);
}
```

→ Ejemplo : si val = 5

val >>1 : 0 0 0 0 0 0 1 0
⊗ bitwise (&) 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 = 0

→ ya con todos los otros desplazamientos daría 0 porque esos 1 desaparecen

⊗ bitwise: operación que opera sobre números binarios a nivel de sus bits.

si val = 6

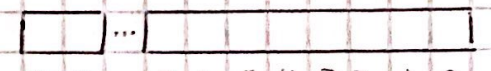
val >>1 : 0 0 0 0 0 0 1 1
& 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1 != 0 → todo lo que no es 0 (no importa que secuencia de bits sea) es verdadero.

→ ya devolvería 1 porque con el anterior resuelto ya se puede definir el final

(operando A)
(val)
↓
(operando B) (1) → ALU: unidad aritmético lógica → 0
(&) →
(operación)

Ejemplo: `int es_uno_al_menos_uno_de_los_2_bits_menos_sig (uint32_t val);`

→ Algunas resoluciones:



↳ bits menos significativos: ¿es al menos uno de ellos igual a 1?

- `return ((val & 1) || (val >> 1) & 1)` (a = `(val & 1)`, b = `(val >> 1) & 1`)

Si val = 6:

a. `0 0 0 0 0 1 1 0` & `0 0 0 0 0 0 0 1` = `0 0 0 0 0 0 0 0` = 0

b. `0 0 0 0 0 0 0 1 1` & `0 0 0 0 0 0 0 1` = `0 0 0 0 0 0 0 1` != 0

→ devuelve 1 (verdadero) ya que el bit en la posición 1 es igual a 1.

- `return ((val & 1) || (val & 2))` (a = `(val & 1)`, b = `(val & 2)`)

Si val = 6

a. `0 0 0 0 0 1 1 0` & `0 0 0 0 0 0 0 1` = 1 = `0 0 0 0 0 0 0 0` = 0

b. `0 0 0 0 0 1 1 0` & `0 0 0 0 0 0 1 0` = 2 = `0 0 0 0 0 0 1 0` != 0

→ es lo mismo que la implementación anterior

- `return ((val & 3) != 0)` → hago todo de una

Si val = 6:

`0 0 0 0 0 1 1 0` & `0 0 0 0 0 0 1 1` = 3 = `0 0 0 0 0 0 1 0` != 0

→ devuelve 1 (verdadero) ya que el resultado es distinto de 0, lo que quiere decir es que al menos uno de los bits (no importa cuál) menos significativos tiene un 1.