Práctica 1: En Binario y Hexadecimal

Christian Most Tazon

Table of Contents

code

Documentacion de la práctica 1 de Christian Most Tazon. Esta práctica trata con operaciones de bytes y listas de bytes, siendo los bytes o una lista de 8 bits o una representacion hexadecimal de estos.

Usage and interface

- **Library usage:**
  `:- use_module(/home/patala/Desktop/CiaoProlog/Practica1/code.pl).`
- **Exports:**
  - *Predicates:*
    `bit/1`, `binary_byte/1`, `hexd/1`, `hex_byte/1`, `byte_list/1`, `byte_convert/2`, `byte_list_convert/2`, `get_nth_bit_from_byte/3`, `reverse_byte/2`, `convert_binary_to_bit_list/2`, `convert_to_bit_list/2`, `convert_to_binary_list/2`, `convert_to_byte_list/2`, `convert_binary_list_to_hex/2`, `displace_bit_list_to_left/2`, `displace_bit_list_to_right/2`, `byte_list_clsh/2`, `byte_list_crsh/2`, `byte_xor/3`, `bit_xor/3`.

Documentation on exports

bit/1: PREDICATE

Usage: `bit(+BinaryValue)`

define los valores posibles de un bit como b(0) o b(1)

binary_byte/1: PREDICATE

Se usa para comprobar si un byte está escrito en formato binario

Usage: `binary_byte(+[B1,B2,B3,B4,B5,B6,B7,B8])`

define un byte binario como una lista de 8 bits

hexd/1: PREDICATE

Usage: `hexd(+HexValue)`

define la representacion de valores hexadecimales como h(0), ..., h(f)

hex_byte/1: PREDICATE

Se usa para comprobar si un byte está escrito en formato hexadecimal

Usage: `hex_byte(+[NIBBLE1,NIBBLE2])`

define un byte hexadecimal como una lista de dos nibbles / digitos hexadecimales

byte_list/1: PREDICATE

Predicado usado para comprobar si una lista es una lista de bytes validos

Usage: `byte_list(+LISTA_BYTES)`

Es cierto si `LISTA_BYTES` es una lista formada por bytes, ya sean hexadecimales o binarios

Other properties:

Example: `byte_list([])`

- *The following properties should hold globally:*
  All the calls of the form `byte_list([])` do not fail. (`not_fails/1`)

Example: `byte_list([[b(0),b(1),b(0),b(1),b(0),b(1),b(1),b(0)]])`

- *The following properties should hold globally:*
  All the calls of the form `byte_list([[b(0),b(1),b(0),b(1),b(0),b(1),b(1),b(0)]])` do not fail. (`not_fails/1`)

Example: `byte_list([[h(1),h(c)]])`

- *The following properties should hold globally:*
  All the calls of the form `byte_list([[h(1),h(c)]])` do not fail. (`not_fails/1`)

Example:
`byte_list([[b(1),b(1),b(0),b(1),b(0),b(1),b(0),b(0)],[h(c),h(1)]])`

- *The following properties should hold globally:*
  All the calls of the form `byte_list([[b(1),b(1),b(0),b(1),b(0),b(1),b(0),b(0)],[h(c),h(1)]])` do not fail. (`not_fails/1`)

byte_convert/2: PREDICATE

Predicado usado para comprobar si dos bytes, uno binario y otro decimal, son equivalentes, y también usado para cambiar el formato de un byte

Usage: `byte_convert(+ByteHexadecimal,+ByteBinario)`

Es cierto si `ByteHexadecimal` y `ByteBinario` son equivalentes

Other properties:

Example:
`byte_convert([h(0),h(0)],[b(0),b(0),b(0),b(0),b(0),b(0),b(0),b(0)])`

- *The following properties should hold globally:*
  All the calls of the form `byte_convert([h(0),h(0)],[b(0),b(0),b(0),b(0),b(0),b(0),b(0),b(0)])` do not fail. (`not_fails/1`)

Example:
`byte_convert([h(f),h(f)],[b(1),b(1),b(1),b(1),b(1),b(1),b(1),b(1)])`

- *The following properties should hold globally:*
  All the calls of the form `byte_convert([h(f),h(f)],[b(1),b(1),b(1),b(1),b(1),b(1),b(1),b(1)])` do not fail. (`not_fails/1`)

Example:
`byte_convert([h(1),h(2)],[b(0),b(0),b(0),b(1),b(0),b(0),b(1),b(0)])`

– *The following properties should hold globally:*

All the calls of the form `byte_convert([h(1),h(2)],[b(0),b(0),b(0),b(1),b(0),b(0),b(1),b(0)])` do not fail. (`not_fails/1`)

Example:
`byte_convert([h(3),h(4)],[b(0),b(0),b(1),b(1),b(0),b(1),b(0),b(0)])`

– *The following properties should hold globally:*

All the calls of the form `byte_convert([h(3),h(4)],[b(0),b(0),b(1),b(1),b(0),b(1),b(0),b(0)])` do not fail. (`not_fails/1`)

Example:
`byte_convert([h(5),h(6)],[b(0),b(1),b(0),b(1),b(0),b(1),b(1),b(0)])`

– *The following properties should hold globally:*

All the calls of the form `byte_convert([h(5),h(6)],[b(0),b(1),b(0),b(1),b(0),b(1),b(1),b(0)])` do not fail. (`not_fails/1`)

byte_list_convert/2: PREDICATE

Predicado usado para transformar una lista de bytes de un formato al otro formato. Se usa en predicados que tienen que realizar operaciones binarias sobre bytes que pueden ser hexadecimales

Usage: `byte_list_convert(+HexadecimalByteList,+BinaryByteList)`

Es cierto si `HexadecimalByteList` y `BinaryByteList` son listas de bytes equivalentes

Other properties:

Example: `byte_list_convert([[h(0),h(1)],[h(f),h(e)]],[[b(0),b(0),b(0),b(0),b(0),b(0),b(0),b(1)],[b(1),b`

– *The following properties should hold globally:*

All the calls of the form `byte_list_convert([[h(0),h(1)],[h(f),h(e)]],[[b(0),b(0),b(0),b(0),b(0),b(0),b(0),b(1)],[b(1)` do not fail. (`not_fails/1`)

get_nth_bit_from_byte/3: PREDICATE

Usage: `get_nth_bit_from_byte(+Index,+Byte,+NthBit)`

Es cierto si `NthBit` es el bit numero `Index` del byte `Byte`. Se cuenta desde el byte menos significativo (desde la derecha)

Other properties:

Example: `get_nth_bit_from_byte(0,[h(0),h(0)],b(0))`

– *The following properties should hold globally:*

All the calls of the form `get_nth_bit_from_byte(0,[h(0),h(0)],b(0))` do not fail. (`not_fails/1`)

Example: `get_nth_bit_from_byte(s(0),[h(0),h(0)],b(0))`

– *The following properties should hold globally:*

All the calls of the form `get_nth_bit_from_byte(s(0),[h(0),h(0)],b(0))` do not fail. (`not_fails/1`)

Example: `get_nth_bit_from_byte(0,[h(f),h(f)],b(0))`

– *The following properties should hold globally:*

Calls of the form `get_nth_bit_from_byte(0,[h(f),h(f)],b(0))` fail. (`fails/1`)

Example: `get_nth_bit_from_byte(0,[h(f),h(f)],b(1))`

– *The following properties should hold globally:*

All the calls of the form `get_nth_bit_from_byte(0,[h(f),h(f)],b(1))` do not fail. (`not_fails/1`)

Example: `get_nth_bit_from_byte(s(s(s(0))),[b(1),b(1),b(1),b(1),b(0),b(1),b(1),b(1)],b(0))`

– *The following properties should hold globally:*

All the calls of the form `get_nth_bit_from_byte(s(s(s(0))),[b(1),b(1),b(1),b(1),b(0),b(1),b(1),b(1)],b(0))` do not fail. (`not_fails/1`)

reverse_byte/2: PREDICATE

Da la vuelta a un byte, invirtiendo la significancia de cada bit, y lo devuelve en un parametro de salida.

Usage: `reverse_byte(+Byte,+ReversedByte)`

Es cierto si `ReversedByte` es `Byte` dado la vuelta

convert_binary_to_bit_list/2: PREDICATE

Es una funcion auxiliar de 'convert_to_bit_list/2' que solo acepta bytes en formato binario. Saca los bits de los bytes de una lista de bytes y devuelve una lista de estos bits. Se puede entender como que 'unifica' los bits de una lista de bytes en una única entidad para facilitar operaciones a nivel bit sobre un conjunto seguido de bytes.

Usage: `convert_binary_to_bit_list(+ByteList,+BitList)`

Es cierto si `BitList` es una lista de los bits que forman los bytes de la lista de bytes binarios `ByteList`

convert_to_bit_list/2: PREDICATE

Saca los bits de los bytes de una lista de bytes y devuelve una lista de estos bits. Se puede entender como que 'unifica' los bits de una lista de bytes en una única entidad para facilitar operaciones a nivel bit sobre un conjunto seguido de bytes. Usa la funcion convert_binary_to_bit_list/2 y convert_to_binary_list/2 para transformar listas de cualquier formato de byte a bits.

Usage: `convert_to_bit_list(+ByteList,+BitList)`

Es cierto si `BitList` es una lista de los bits que forman los bytes de la lista de bytes en cualquier formato `ByteList`

convert_to_binary_list/2: PREDICATE

Transforma una lista de bytes a una lista de bytes en formato binario. Para hacer esto transforma bytes de formato hexadecimal a decimal y copia bytes que ya están en formato

binario. Debido a esto, la función devuelve siempre lo mismo sin necesidad de revisar que la lista a transformar no está ya en formato binario.

Usage: `convert_to_binary_list(+ByteList,+BinaryByteList)`

Es cierto si `BinaryByteList` es una lista de bytes en formato binario equivalentes a la lista de bytes `ByteList`

convert_to_byte_list/2: PREDICATE

Junta los bits de BitList en grupos de 8 bits, formando bytes binarios, y devuelve una lista de estos bytes

Usage: `convert_to_byte_list(+BitList,+ByteList)`

Es cierto si `ByteList` es una lista de bytes con los mismos bits que `BitList`, en el mismo orden

convert_binary_list_to_hex/2: PREDICATE

Transforma el formato de los bytes de una lista de bytes al otro. Es util para cuando hay que realizar operaciones binarias sobre listas de bytes en formato hexadecimal.

Usage: `convert_binary_list_to_hex(+BinaryByteList,+HexadecimalByteList)`

Es cierto si `HexadecimalByteList` es una lista de bytes en formato hexadecimal equivalente a la lista de bytes en formato binario `BinaryByteList`

displace_bit_list_to_left/2: PREDICATE

Produce un desplazamiento logico a la izquierda de una lista de bits. Mueve el primer bit al final de la lista.

Usage: `displace_bit_list_to_left(+BitList,+DisplacedList)`

Es cierto si `DisplacedList` es una lista de bits equivalente a `BitList` tras un desplazamiento lógico a la izquierda.

displace_bit_list_to_right/2: PREDICATE

Produce un desplazamiento logico a la derecha de una lista de bits. Mueve el ultimo bit al principio de la lista.

Usage: `displace_bit_list_to_right(+BitList,+DisplacedList)`

Es cierto si `DisplacedList` es una lista de bits equivalente a `BitList` tras un desplazamiento lógico a la derecha.

byte_list_clsh/2: PREDICATE

Produce un desplazamiento lógico a la izquierda a nivel bit de la lista de bytes ByteList. El resultado mantiene el formato de bytes de la entrada, pero la lista tiene que estar formada por bytes de un único tipo de formato.

Usage: `byte_list_clsh(+ByteList,+ByteListLeftShift)`

Es cierto si la lista de bytes en cualquier formato `ByteListLeftShift` es `ByteList` tras un desplazamiento lógico a la izquierda a nivel bit.

byte_list_crsh/2: PREDICATE

Produce un desplazamiento lógico a la derecha a nivel bit de la lista de bytes ByteList. El resultado mantiene el formato de bytes de la entrada, pero la lista tiene que estar formada por bytes de un único tipo de formato.

Usage: `byte_list_crsh(+ByteList,+ByteListRightShift)`

Es cierto si la lista de bytes en cualquier formato `ByteListRightShift` es `ByteList` tras un desplazamiento lógico a la derecha a nivel bit.

byte_xor/3: PREDICATE

Realiza la operacion 'XOR' entre dos bytes en cualquier formato y devuelve el resultado de la operacion en un parametro de salida, mantendiendo el formato de los bytes. Todos los bytes tienen que tener el mismo formato.

Usage: `byte_xor(+Byte1,+Byte2,+ResultByte)`

devuelve cierto si `ResultByte` es el byte resultante de la operación 'XOR' entre los bytes `Byte1` y `Byte2`

bit_xor/3: PREDICATE

Realiza la operacion lógica 'XOR' entre dos bits y devuelve el bit resultante. Este predicado define la tabla lógica de la operación 'XOR'.

Usage: `bit_xor(+Bit1,+Bit2,+ResultBit)`

devuelve cierto si `ResultBit` es el but resultante de la operación 'XOR' entre los bit `Bit1` y `Bit2`

Documentation on imports

This module has the following direct dependencies:

– *Application modules:*

`unittest`.

– *Internal (engine) modules:*

`term_basic`, `basiccontrol`, `debugger_support`, `basic_props`.

– *Packages:*

`pure`, `initial`, `condcomp`, `assertions`, `assertions/assertions_basic`, `regtypes`.