



Tema 4:

# Diseño del formato de instrucción

Fundamentos de computadores II

**José Manuel Mendías Cuadros**

*Dpto. Arquitectura de Computadores y Automática  
Universidad Complutense de Madrid*





# Contenidos

- ✓ Formatos de instrucción del RISC-V.
- ✓ Codificación de campos.
- ✓ De ensamblador a código máquina.
- ✓ De código máquina a ensamblador.

Transparencias basadas en los libros:

- S.L. Harris and D. Harris. *Digital Design and Computer Architecture. RISC-V Edition.*
- D.A. Patterson and J.L. Hennessy. *Computer Organization and Design. RISC-V Edition.*



# Formatos de instrucción

- Los **formatos de instrucción** determinan la ubicación y codificación de los campos de una instrucción máquina.
  - A mayor regularidad, más simple será el circuito digital que la decodifica.
- Los formatos de instrucción del RISC-V tienen los siguientes campos:
  - **Código de operación** (**op**): indica la clase de instrucción.
  - **Código de función** (**funct3** y **funct7**): determina la instrucción concreta dentro de la clase.
  - **Operando en registro** (**rs1**, **rs2**, **rd**): codifica un número del registro.
  - **Operando inmediato** (**imm**): contiene un valor inmediato que se interpreta en binario puro o C2 según la instrucción.
    - Este campo puede estar fragmentado en 2 porciones distribuidas a lo largo de la instrucción.



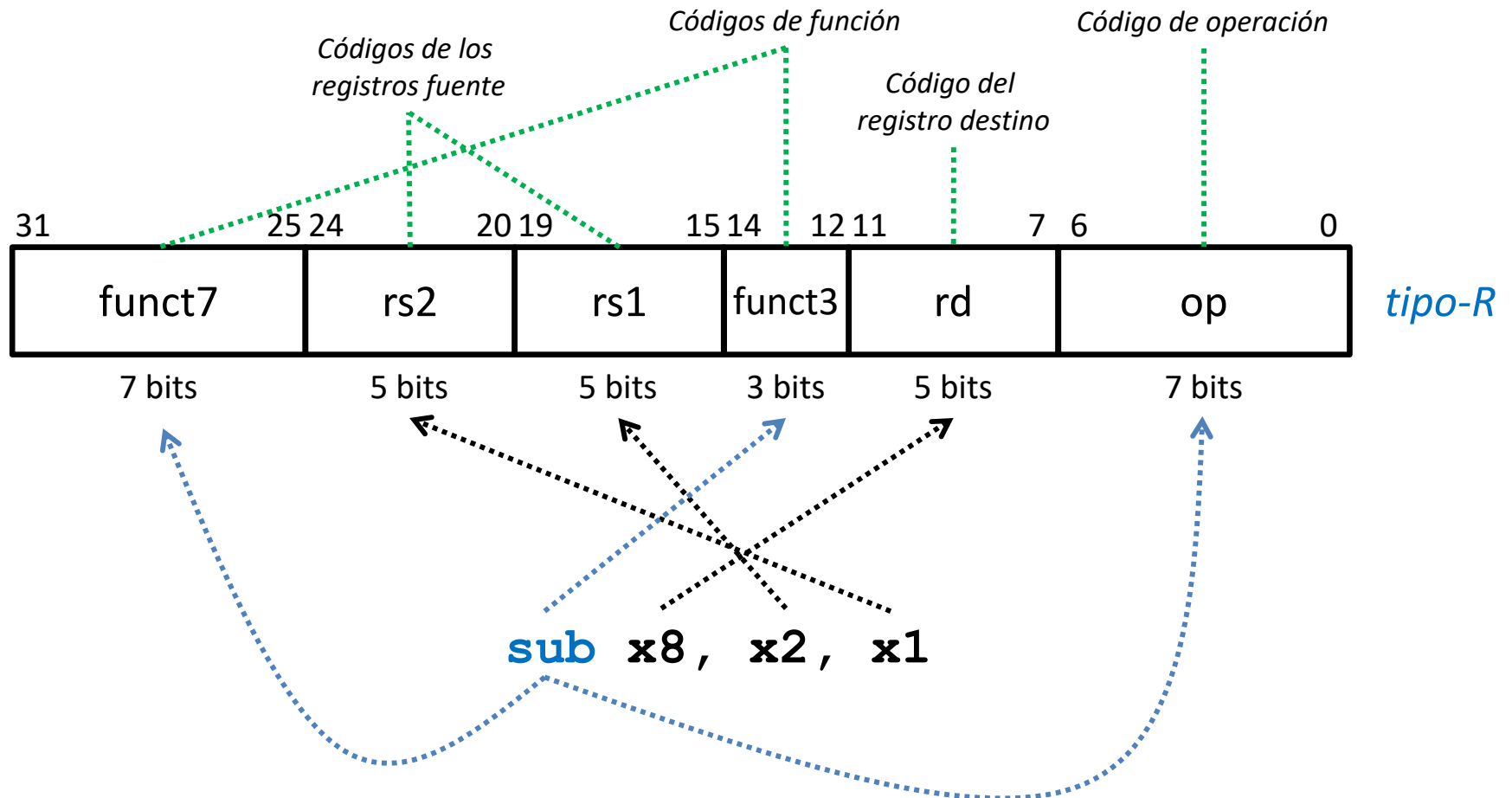
# Formatos de instrucción

- En RISC-V solo existen 4 tipos de formatos:
  - **Tipo-R**: para instrucciones con 3 operandos en registros, 2 fuente y 1 destino.
    - Aritmético-lógicas y de desplazamiento.
  - **Tipo-I**: para instrucciones con 2 operandos en registros, uno fuente y otro destino, y 1 operando fuente inmediato corto (12 bits).
    - Aritmético-lógica y de desplazamiento con operando inmediato, de carga y jalr.
  - **Tipo-S/B**: para instrucciones con 2 operandos fuente en registros y 1 operando inmediato corto (12/13 bits).
    - De almacenamiento (S) y de salto condicional (B).
  - **Tipo-U/J**: para instrucciones con 1 operando destino en registro y 1 operando inmediato largo (20/21 bits).
    - lui (U) , auipc (U) y jal (J).
- Todos los formatos tiene anchura fija de 32 bits.



# Formato tipo-R

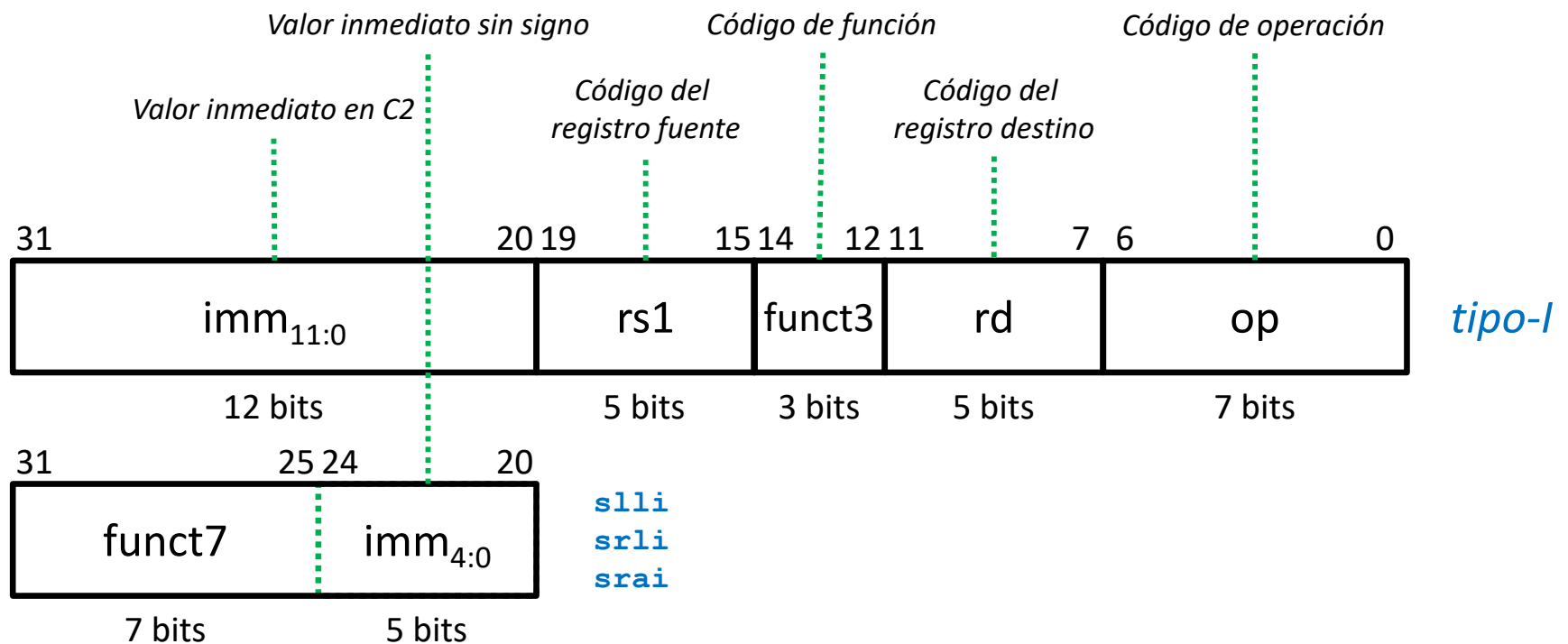
- Usado para codificar instrucciones con 3 operandos en registros, 2 fuente y 1 destino.





# Formato tipo-I

- Usado para codificar **instrucciones con 2 operandos en registros**, uno fuente y uno destino, y **1 operando fuente inmediato corto (12b)**
  - En **instrucciones aritméticas, lógicas y de carga de memoria**, el signo del valor inmediato se extenderá hasta 32 bits.
  - En las **instrucciones de desplazamiento** solo se usan los **5 bits inferiores del valor inmediato**, los 7 bits superiores se usan como campo de función.

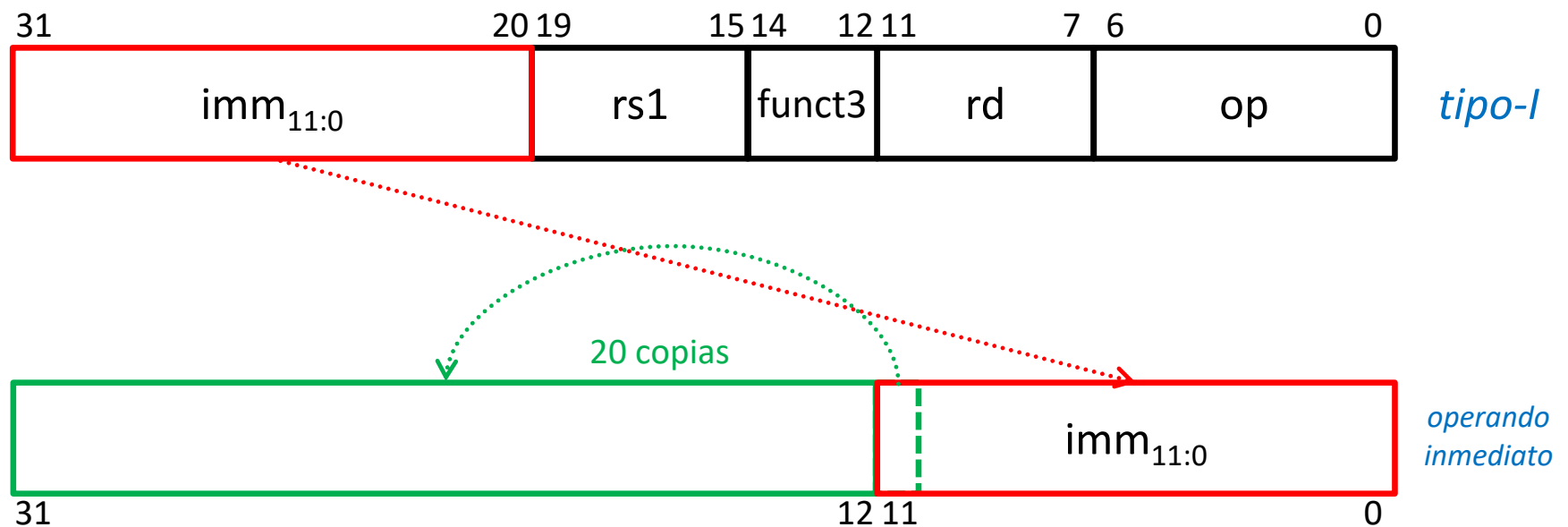




# Formato tipo-I

## Codificación del operando inmediato

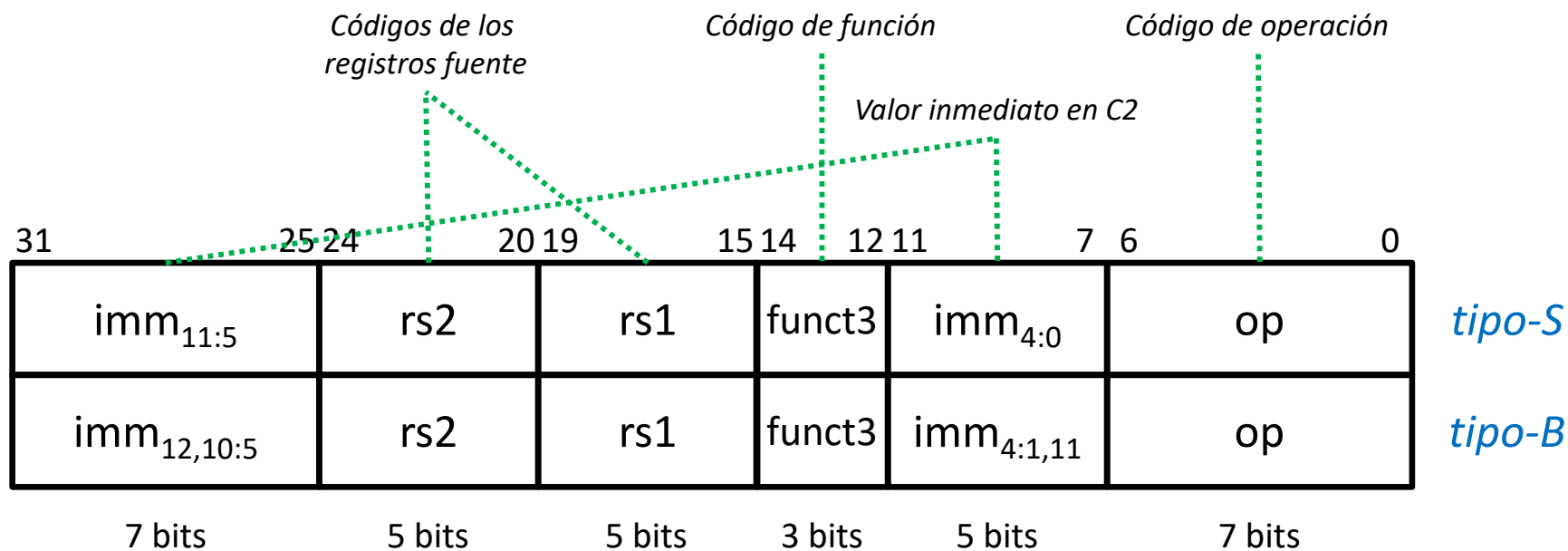
- Para obtener de una instrucción de tipo-I el **operando inmediato efectivo de 32b**:
  - El signo se extiende 20 bits.





# Formato tipo S/B

- Usado para codificar instrucciones con 2 operandos fuente en registros y 1 operando fuente inmediato corto (12/13b).
  - En ambos formatos, el valor inmediato está fragmentado en 2 campos y su signo se extenderá hasta 32 bits.
  - En el formato tipo-B, además, de los 13b del valor inmediato solo se almacenan desordenados los 12b superiores.
    - Las instrucciones se ubican en direcciones múltiplos de 4 acabadas en 2 ceros, solo se quita un 0 por compatibilidad con extensión RVC (instrucciones de 16b).



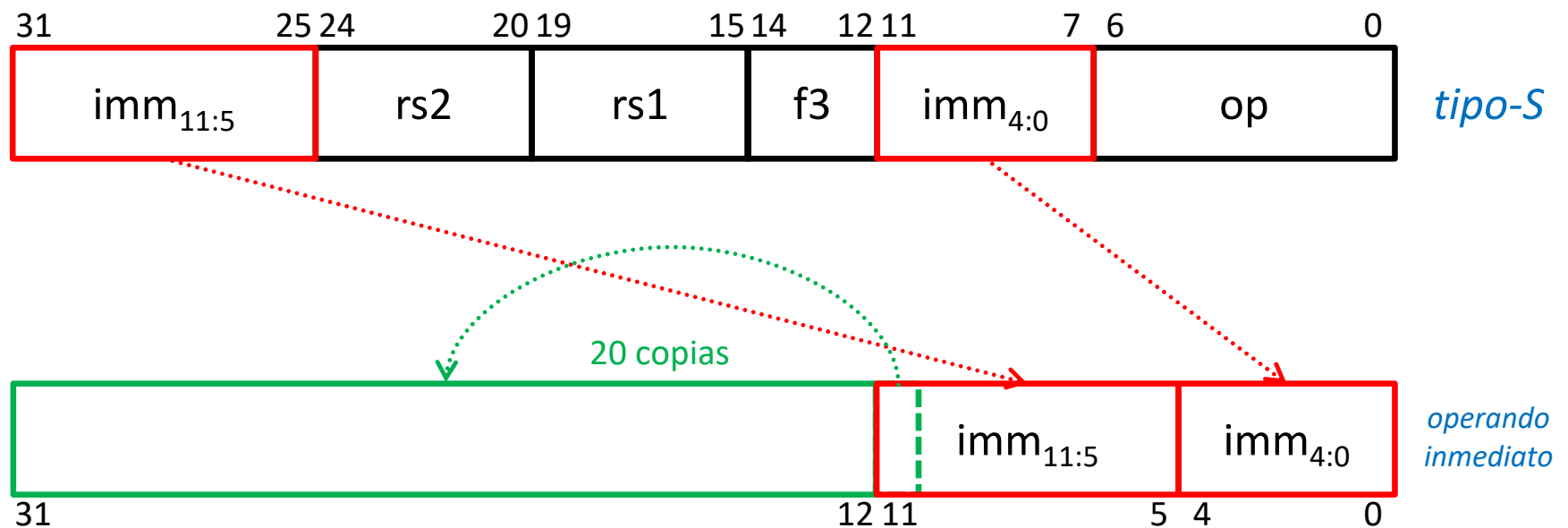




# Formato tipo S

## Codificación del operando inmediato

- Para obtener de una instrucción de tipo-S el **operando inmediato efectivo de 32 bits**:
  - Los campos inmediatos se concatenan.
  - El signo se extiende 20 bits.

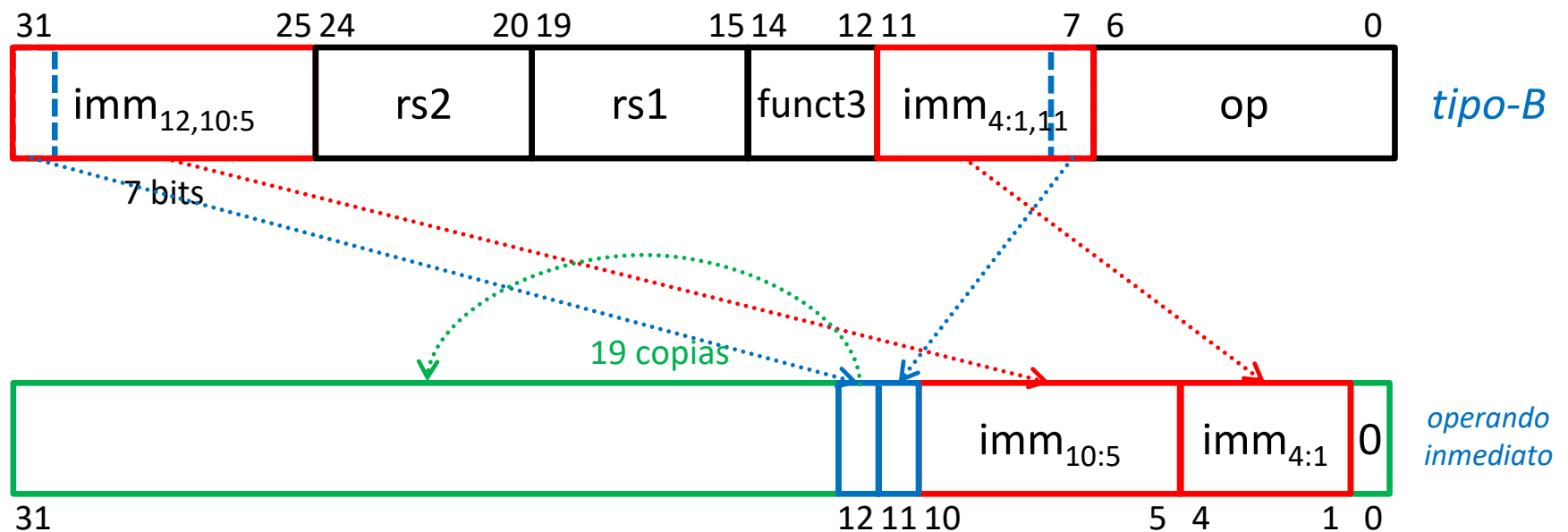




# Formato tipo B

## Codificación del operando inmediato

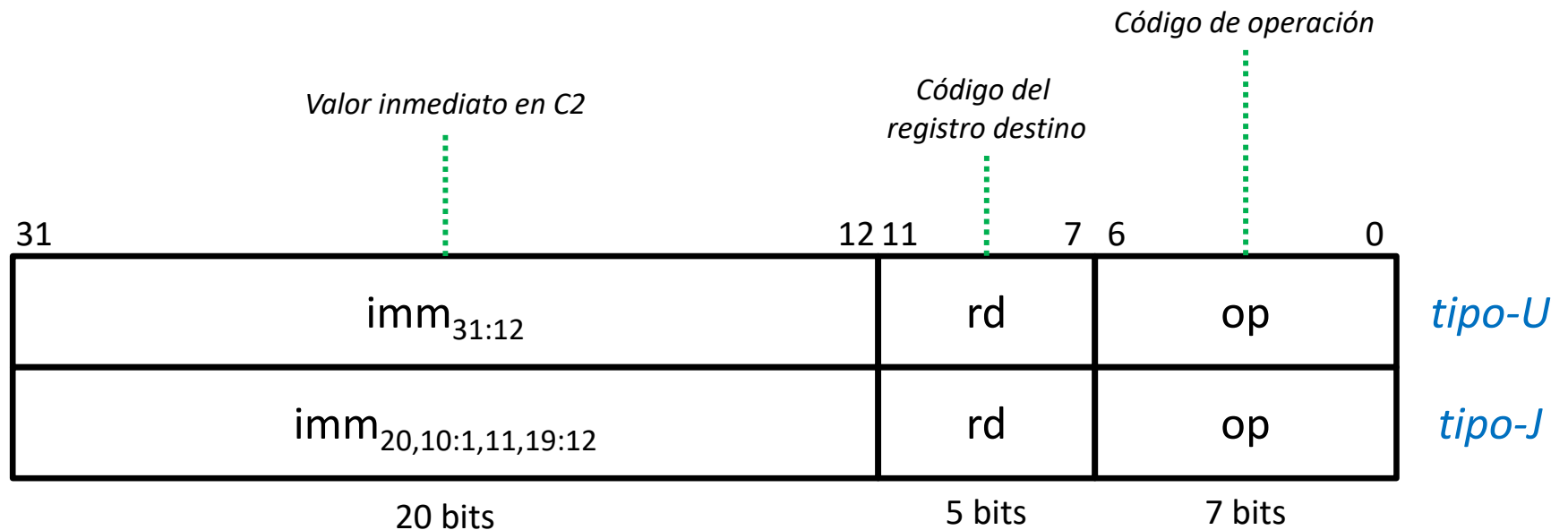
- Para obtener de una instrucción de tipo-B el **operando inmediato efectivo de 32 bits**:
  - Los campos se deben concatenar.
  - Los campos inmediatos se reordenan.
  - El signo se extiende 19 bits.
  - Se completa con 1 cero por la derecha.





# Formatos tipo U/J

- Usado para codificar instrucciones con **1 operando destino en registro** y **1 operando inmediato largo (20/21b)**.
  - En el formato tipo-U, el **valor inmediato se completa** con 0 por la derecha hasta 32 bits.
  - En el formato tipo-J, de los **21b del valor inmediato solo se almacenan desordenados los 20b superiores** y su signo se extenderá hasta 32 bits.
    - Las instrucciones se ubican en direcciones múltiplos de 4 acabadas en 2 ceros, solo se quita un 0 por compatibilidad con extensión RVC (instrucciones de 16b).

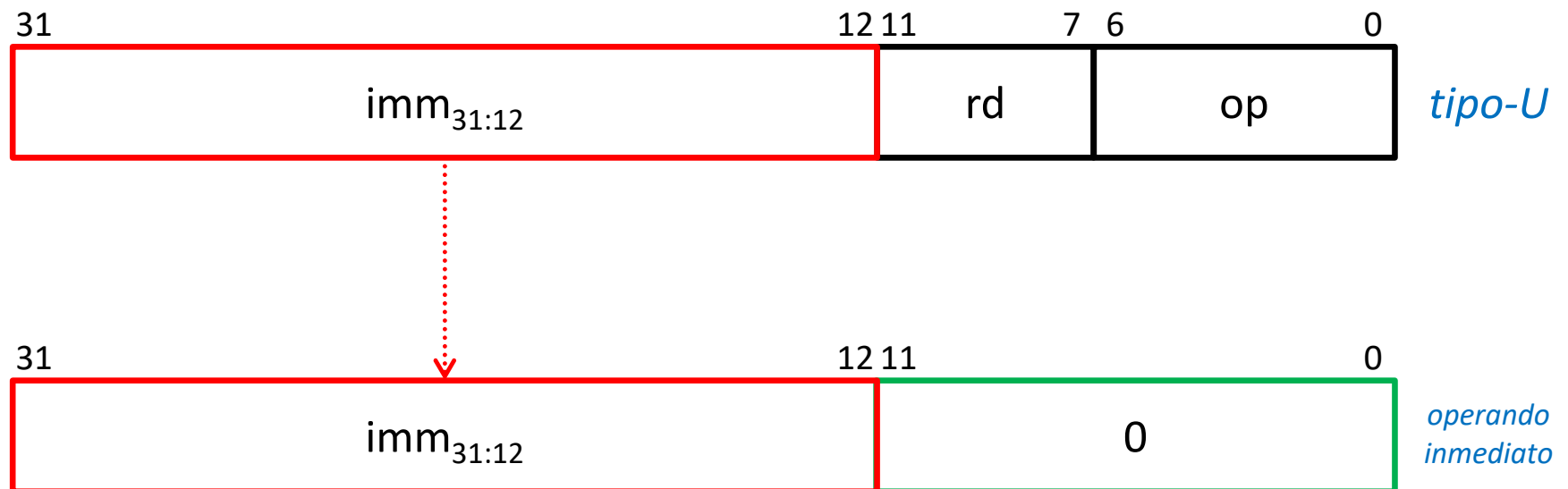




# Formato tipo U

## Codificación del operando inmediato

- Para obtener de una instrucción de tipo-U el **operando inmediato efectivo de 32 bits**:
  - El campo inmediato se completa con 12 ceros por la derecha.

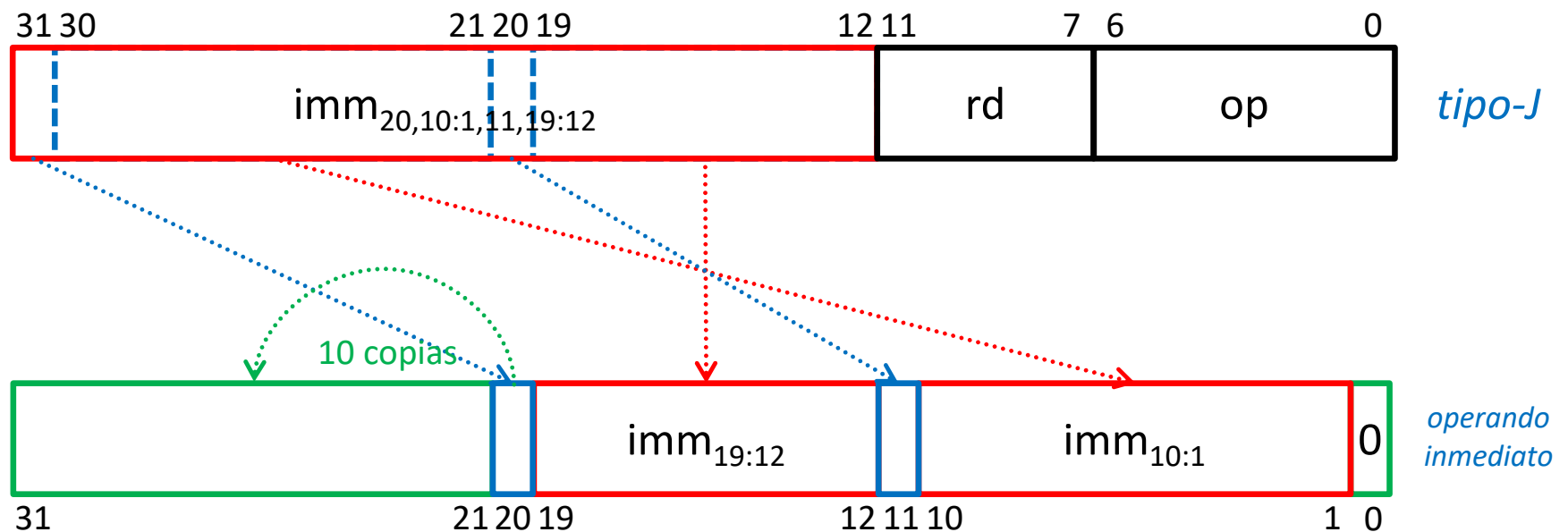




# Formato tipo J

## Codificación del operando inmediato

- Para obtener de una instrucción de tipo-J el **operando inmediato efectivo de 32 bits**:
  - El campo inmediato se reordena.
  - El signo se extiende 10 bits.
  - Se completa con 1 cero por la derecha.

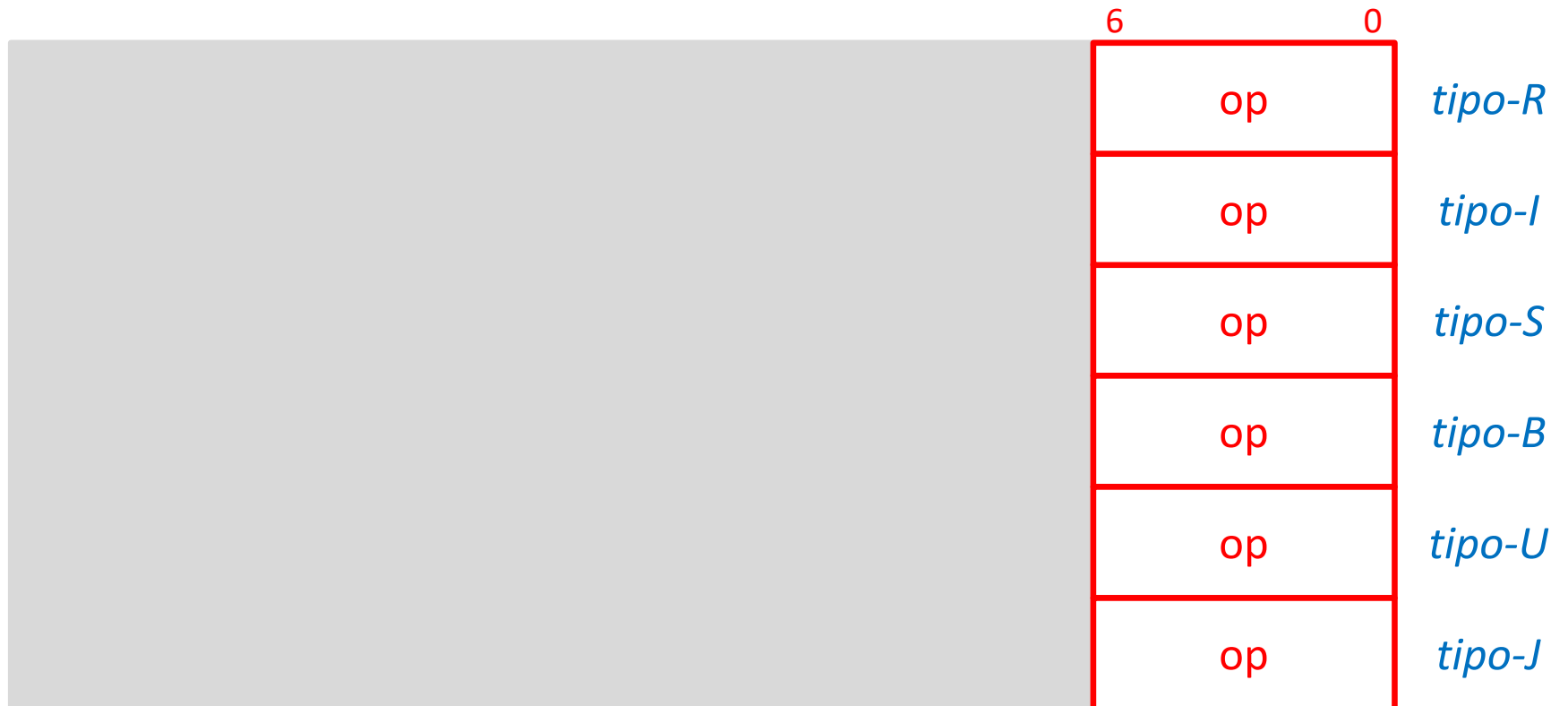




# Formatos de instrucción

## Resumen (i)

- Los formatos del RSIC-V son **extremadamente regulares**:
  - Los **mismos campos** siempre ocupan la **misma posición**.

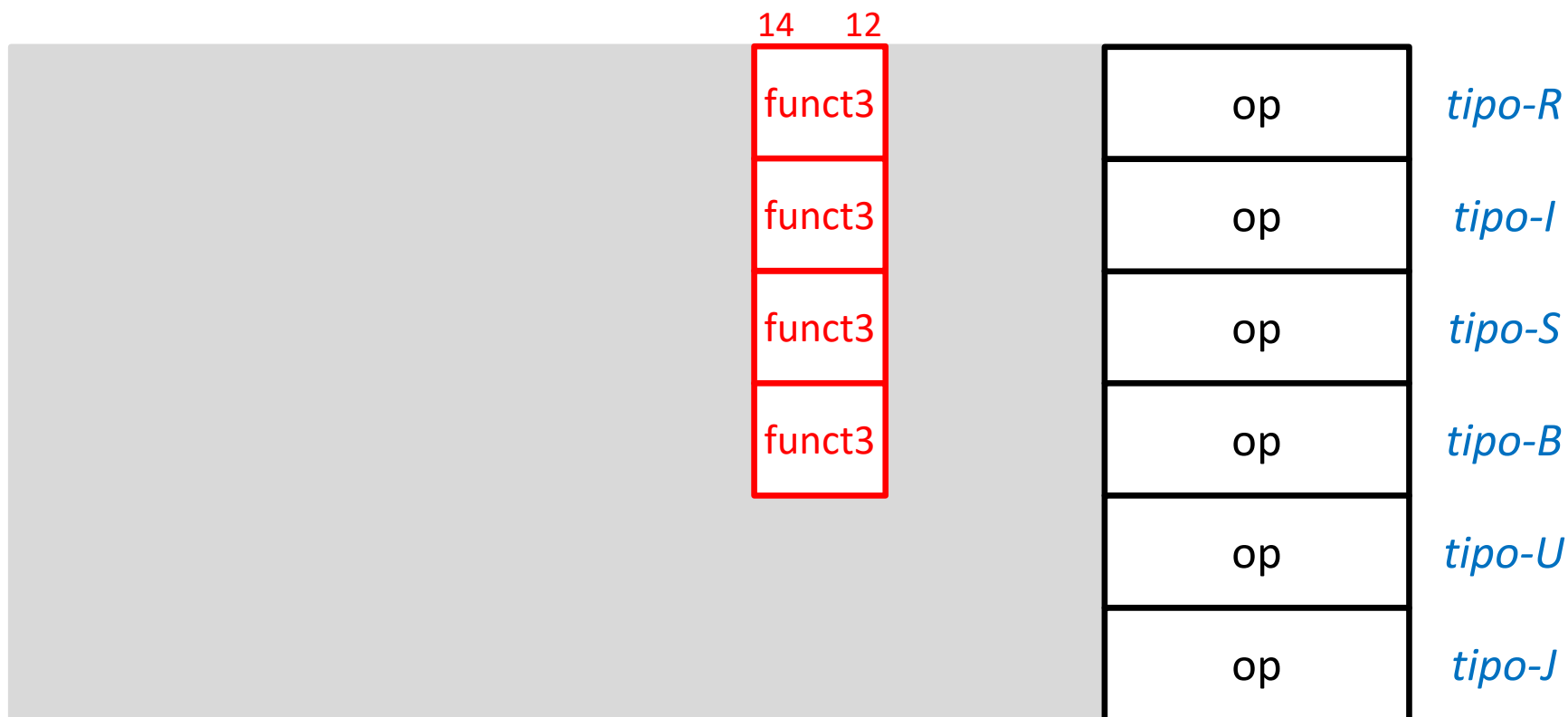




# Formatos de instrucción

## Resumen (i)

- Los formatos del RSIC-V son **extremadamente regulares**:
  - Los **mismos campos** siempre ocupan la **misma posición**.





# Formatos de instrucción

## Resumen (i)

- Los formatos del RSIC-V son **extremadamente regulares**:
  - Los **mismos campos** siempre ocupan la **misma posición**.

|  |        |    |    |               |
|--|--------|----|----|---------------|
|  |        | 11 | 7  |               |
|  | funct3 | rd | op | <i>tipo-R</i> |
|  | funct3 | rd | op | <i>tipo-I</i> |
|  | funct3 |    | op | <i>tipo-S</i> |
|  | funct3 |    | op | <i>tipo-B</i> |
|  |        | rd | op | <i>tipo-U</i> |
|  |        | rd | op | <i>tipo-J</i> |





# Formatos de instrucción

## Resumen (i)

- Los formatos del RSIC-V son **extremadamente regulares**:
  - Los **mismos campos** siempre ocupan la **misma posición**.

|  |    |    |     |        |    |    |               |
|--|----|----|-----|--------|----|----|---------------|
|  | 19 | 15 | rs1 | funct3 | rd | op | <i>tipo-R</i> |
|  |    |    | rs1 | funct3 | rd | op | <i>tipo-I</i> |
|  |    |    | rs1 | funct3 |    | op | <i>tipo-S</i> |
|  |    |    | rs1 | funct3 |    | op | <i>tipo-B</i> |
|  |    |    |     |        | rd | op | <i>tipo-U</i> |
|  |    |    |     |        | rd | op | <i>tipo-J</i> |



# Formatos de instrucción

## Resumen (i)

- Los formatos del RSIC-V son **extremadamente regulares**:
  - Los **mismos campos** siempre ocupan la **misma posición**.

|  |    |    |     |     |        |    |    |               |
|--|----|----|-----|-----|--------|----|----|---------------|
|  | 24 | 20 | rs2 | rs1 | funct3 | rd | op | <i>tipo-R</i> |
|  |    |    |     | rs1 | funct3 | rd | op | <i>tipo-I</i> |
|  |    |    | rs2 | rs1 | funct3 |    | op | <i>tipo-S</i> |
|  |    |    | rs2 | rs1 | funct3 |    | op | <i>tipo-B</i> |
|  |    |    |     |     |        | rd | op | <i>tipo-U</i> |
|  |    |    |     |     |        | rd | op | <i>tipo-J</i> |



# Formatos de instrucción

## Resumen (i)

- Los formatos del RSIC-V son **extremadamente regulares**:
  - Los **mismos campos** siempre ocupan la **misma posición**.
  - Los **campos no usados se reutilizan** para otros propósitos.

|        |    |     |     |        |                       |    |               |
|--------|----|-----|-----|--------|-----------------------|----|---------------|
| 31     | 25 |     |     | 11     | 7                     |    |               |
| funct7 |    | rs2 | rs1 | funct3 | rd                    | op | <i>tipo-R</i> |
|        |    |     | rs1 | funct3 | rd                    | op | <i>tipo-I</i> |
|        |    | rs2 | rs1 | funct3 | imm <sub>4:0</sub>    | op | <i>tipo-S</i> |
|        |    | rs2 | rs1 | funct3 | imm <sub>4:1,11</sub> | op | <i>tipo-B</i> |
|        |    |     |     |        | rd                    | op | <i>tipo-U</i> |
|        |    |     |     |        | rd                    | op | <i>tipo-J</i> |



# Formatos de instrucción

## Resumen (i)

- Los formatos del RSIC-V son **extremadamente regulares**:
  - Los **mismos campos** siempre ocupan la **misma posición**.
  - Los **campos no usados se reutilizan** para otros propósitos.
  - El **signo del valor inmediato** siempre está en la **posición 31** de la instrucción.

| 31 |                                 |     |     |        |                       | 0  |               |
|----|---------------------------------|-----|-----|--------|-----------------------|----|---------------|
|    | funct7                          | rs2 | rs1 | funct3 | rd                    | op | <i>tipo-R</i> |
|    | imm <sub>11:0</sub>             |     | rs1 | funct3 | rd                    | op | <i>tipo-I</i> |
|    | imm <sub>11:5</sub>             | rs2 | rs1 | funct3 | imm <sub>4:0</sub>    | op | <i>tipo-S</i> |
|    | imm <sub>12,10:5</sub>          | rs2 | rs1 | funct3 | imm <sub>4:1,11</sub> | op | <i>tipo-B</i> |
|    | imm <sub>31:12</sub>            |     |     |        | rd                    | op | <i>tipo-U</i> |
|    | imm <sub>20,10:1,11,19:12</sub> |     |     |        | rd                    | op | <i>tipo-J</i> |



# Formatos de instrucción

## Resumen (i)

- Los formatos del RSIC-V son **extremadamente regulares**:
  - Los **mismos campos** siempre ocupan la **misma posición**.
  - Los **campos no usados se reutilizan** para otros propósitos.
  - El **signo del valor inmediato** siempre está en la **posición 31** de la instrucción.

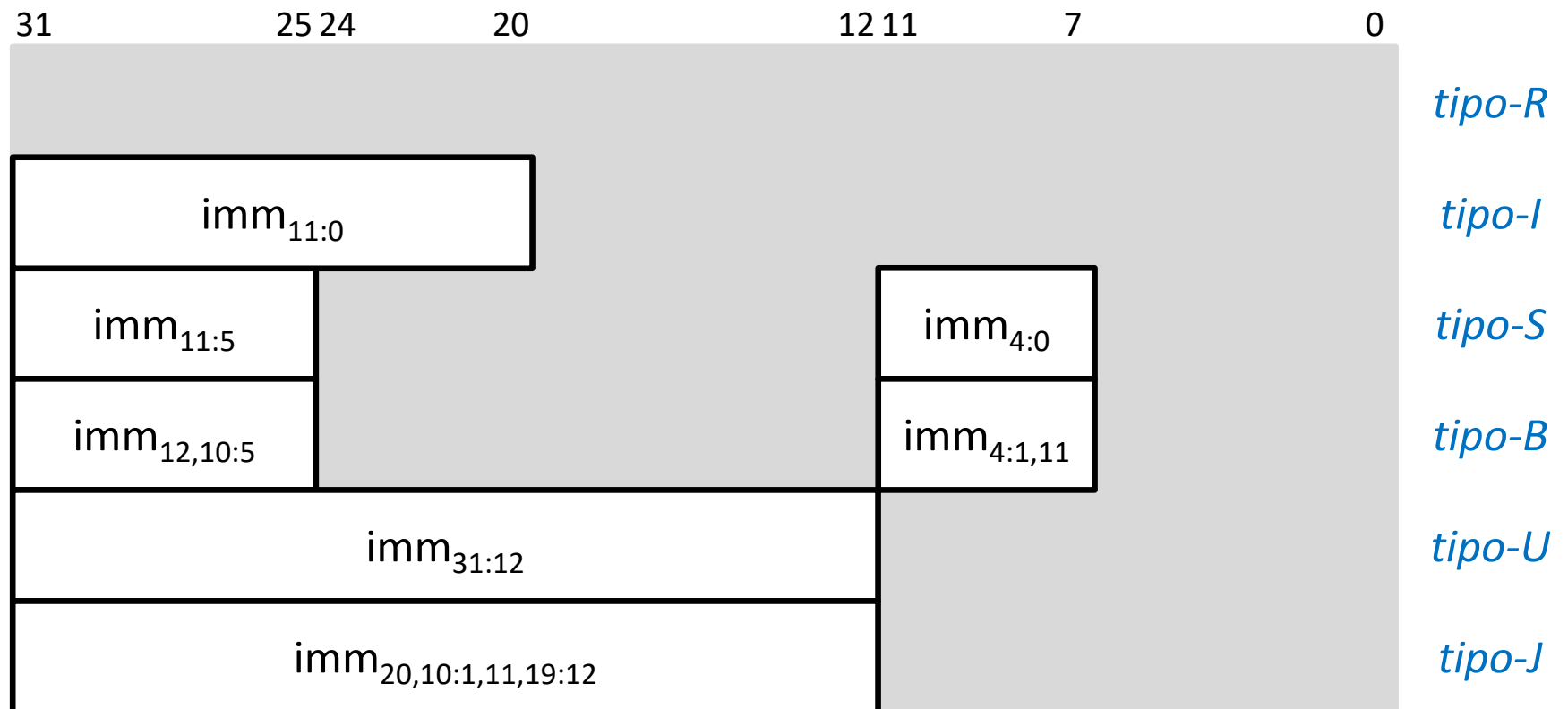
| 31                              | 25 24 | 20 19 | 15 14  | 12 11                 | 7 6 | 0 |               |
|---------------------------------|-------|-------|--------|-----------------------|-----|---|---------------|
| funct7                          | rs2   | rs1   | funct3 | rd                    | op  |   | <i>tipo-R</i> |
| imm <sub>11:0</sub>             |       | rs1   | funct3 | rd                    | op  |   | <i>tipo-I</i> |
| imm <sub>11:5</sub>             | rs2   | rs1   | funct3 | imm <sub>4:0</sub>    | op  |   | <i>tipo-S</i> |
| imm <sub>12,10:5</sub>          | rs2   | rs1   | funct3 | imm <sub>4:1,11</sub> | op  |   | <i>tipo-B</i> |
| imm <sub>31:12</sub>            |       |       |        | rd                    | op  |   | <i>tipo-U</i> |
| imm <sub>20,10:1,11,19:12</sub> |       |       |        | rd                    | op  |   | <i>tipo-J</i> |



# Formatos de instrucción

## Resumen (ii)

- La ubicación de los operandos inmediatos está ideada para simplificar la lógica de extensión de signo.
  - Los mismos fragmentos suelen ocupar la misma posición.





# Formatos de instrucción

## Resumen (ii)

- La ubicación de los operandos inmediatos está ideada para simplificar la lógica de extensión de signo.
  - Los mismos fragmentos suelen ocupar la misma posición.

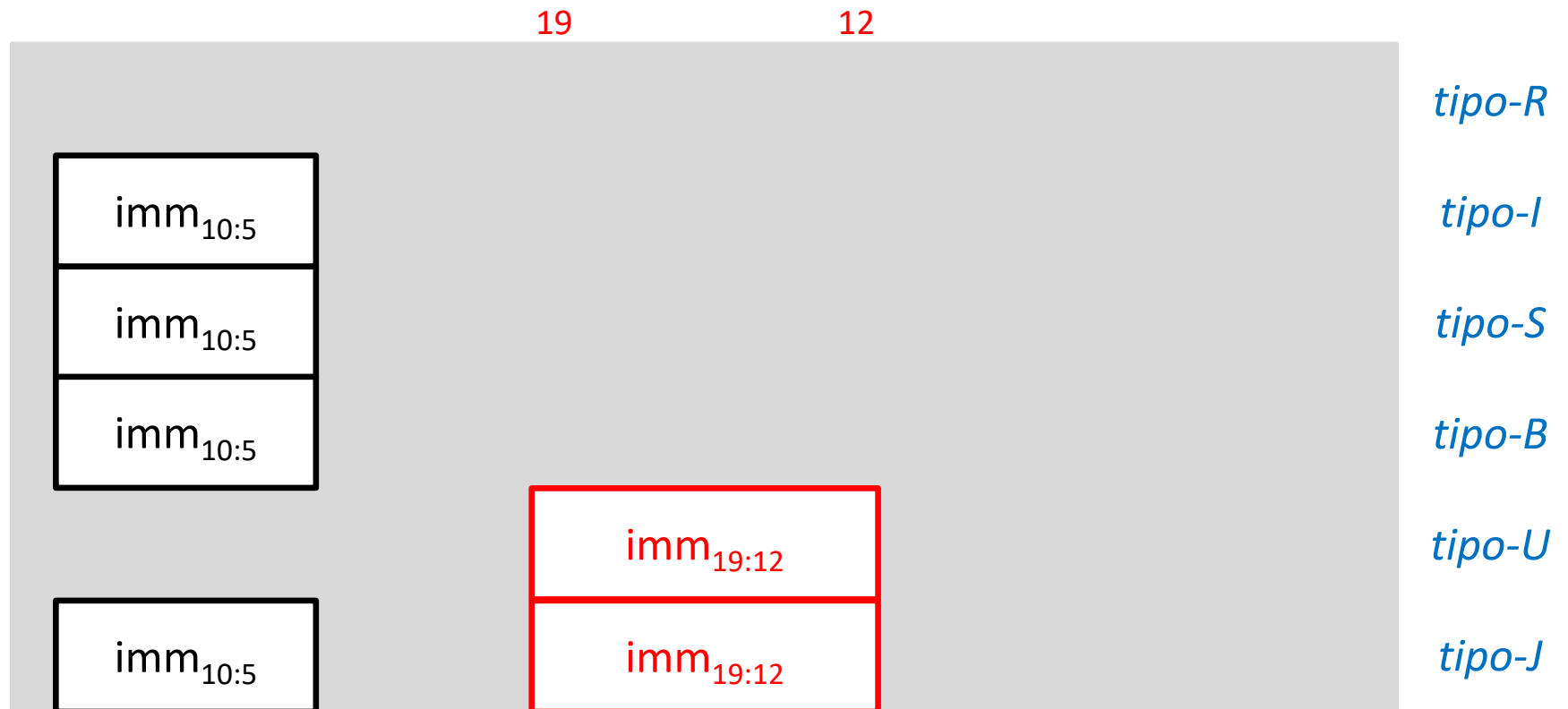




# Formatos de instrucción

## Resumen (ii)

- La ubicación de los operandos inmediatos está ideada para simplificar la lógica de extensión de signo.
  - Los mismos fragmentos suelen ocupar la misma posición.



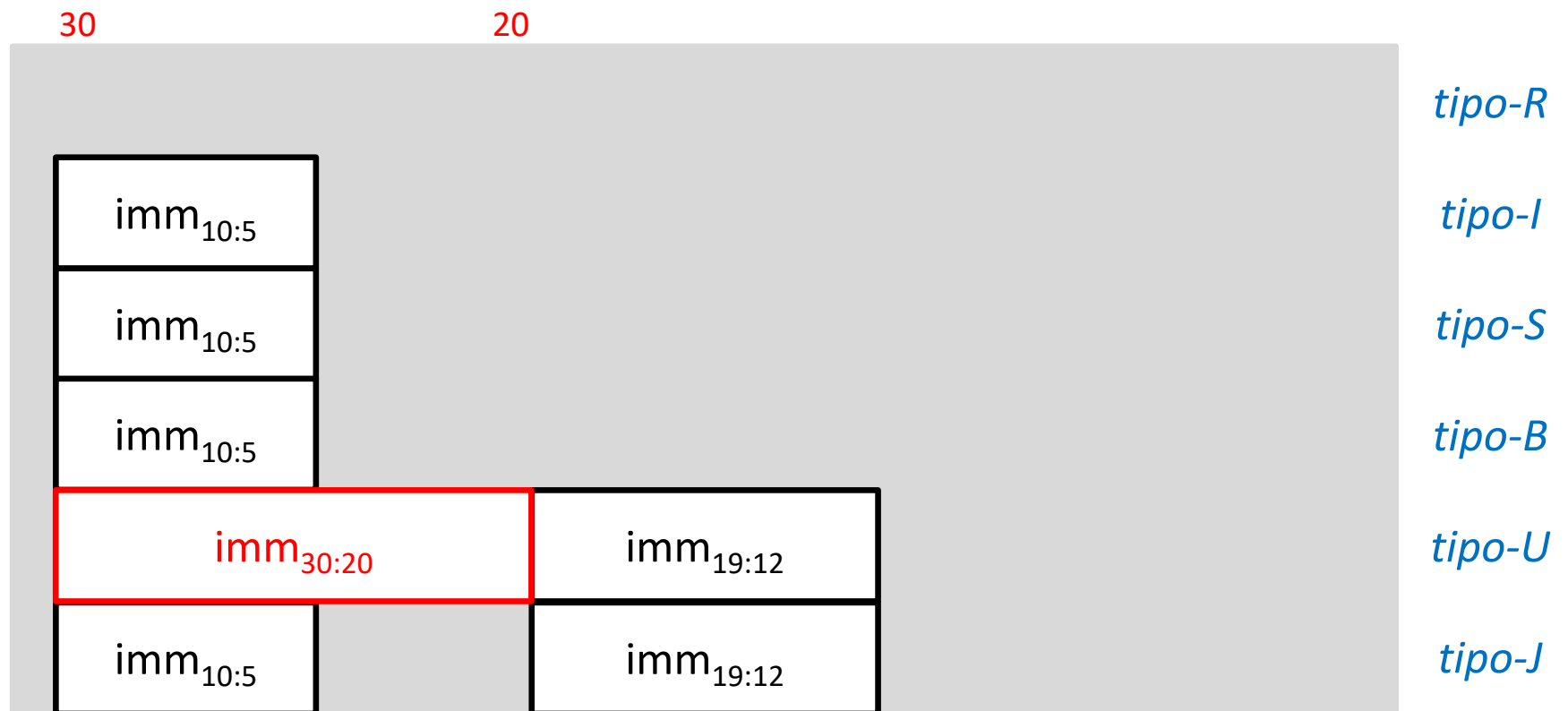




# Formatos de instrucción

## Resumen (ii)

- La ubicación de los operandos inmediatos está ideada para simplificar la lógica de extensión de signo.
  - Los mismos fragmentos suelen ocupar la misma posición.



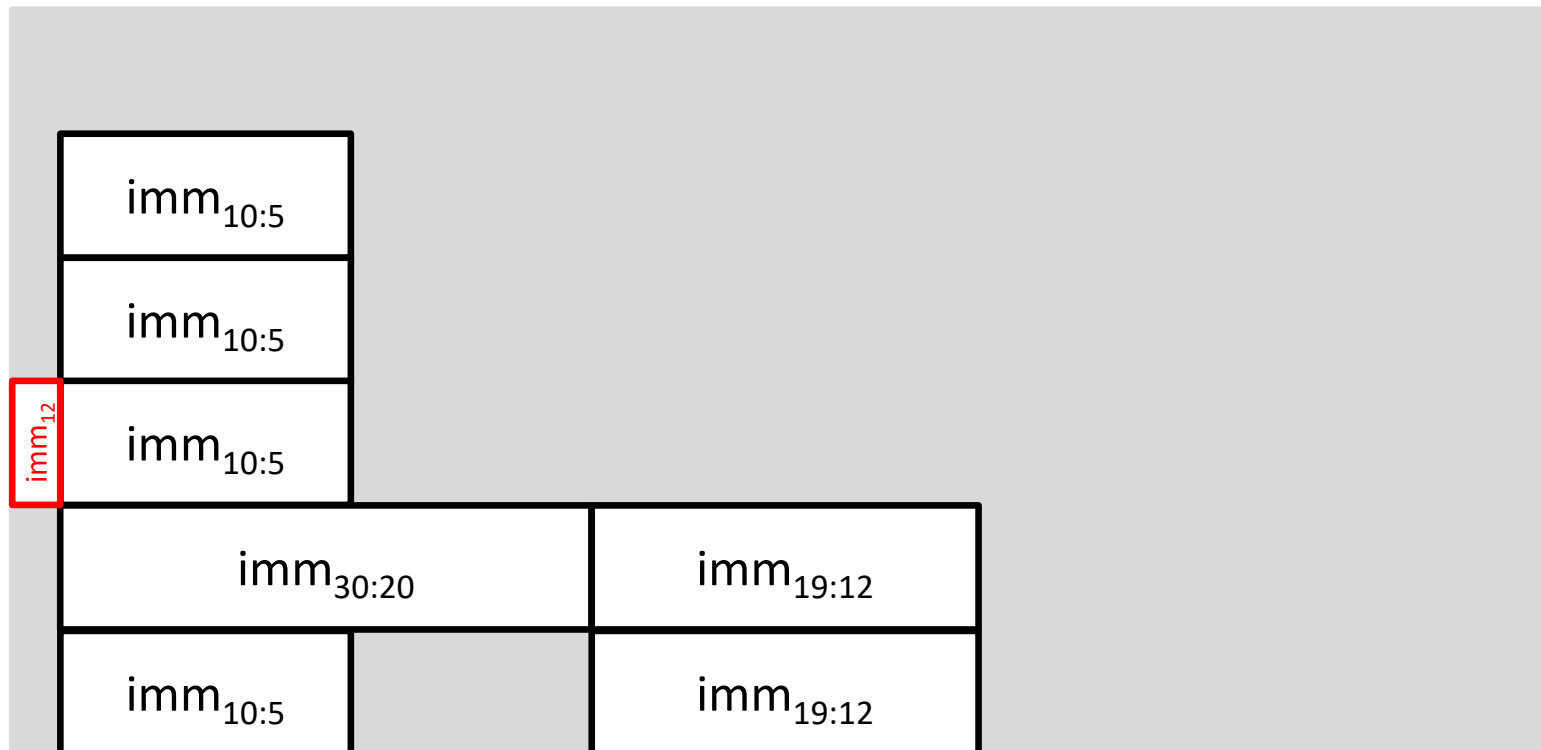


# Formatos de instrucción

## Resumen (ii)

- La ubicación de los operandos inmediatos está ideada para simplificar la lógica de extensión de signo.
  - Los mismos fragmentos suelen ocupar la misma posición.
  - Si no, se distribuyen en el menor número de posiciones diferentes posible.

31



*tipo-R*

*tipo-I*

*tipo-S*

*tipo-B*

*tipo-U*

*tipo-J*

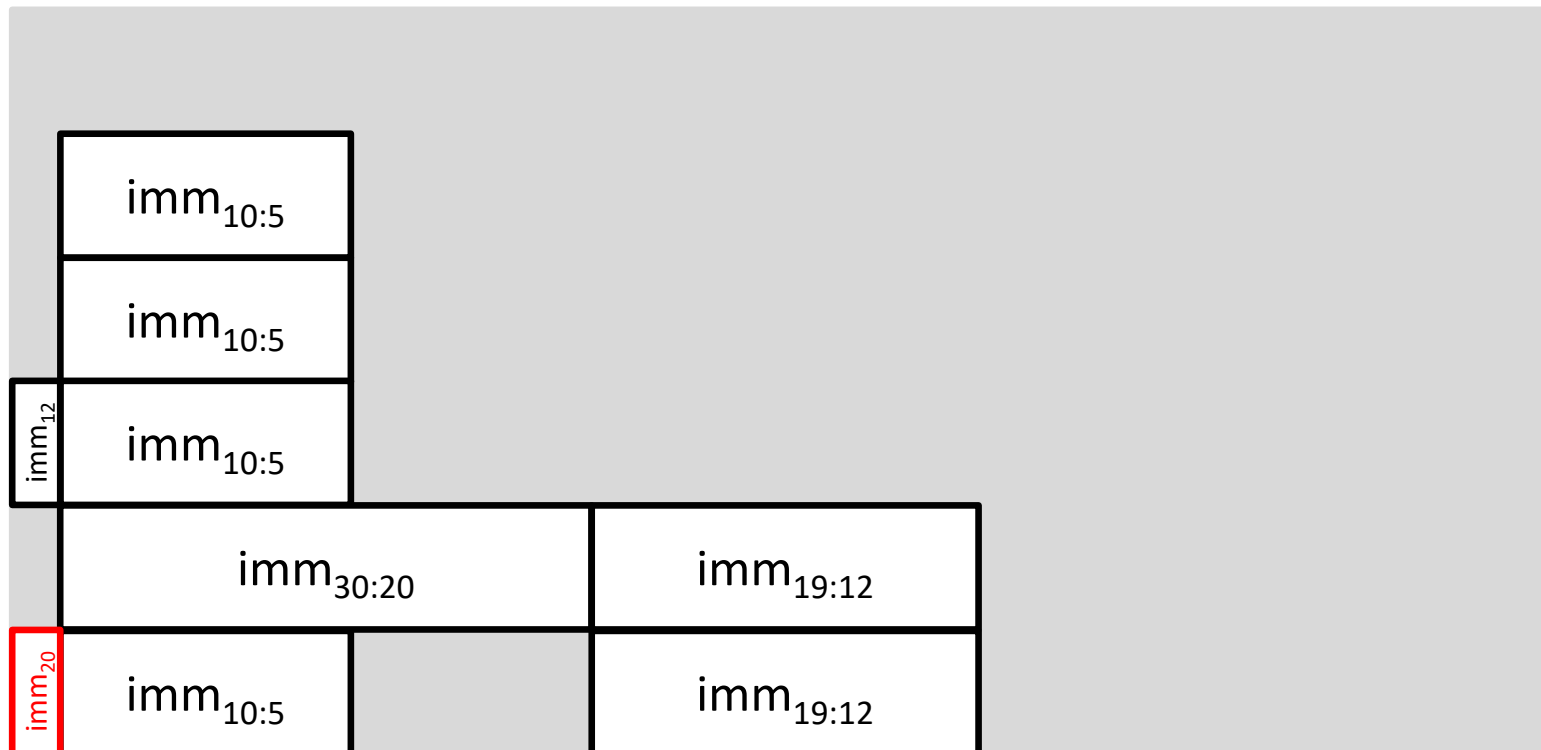


# Formatos de instrucción

## Resumen (ii)

- La ubicación de los operandos inmediatos está ideada para simplificar la lógica de extensión de signo.
  - Los mismos fragmentos suelen ocupar la misma posición.
  - Si no, se distribuyen en el menor número de posiciones diferentes posible.

31



*tipo-R*

*tipo-I*

*tipo-S*

*tipo-B*

*tipo-U*

*tipo-J*

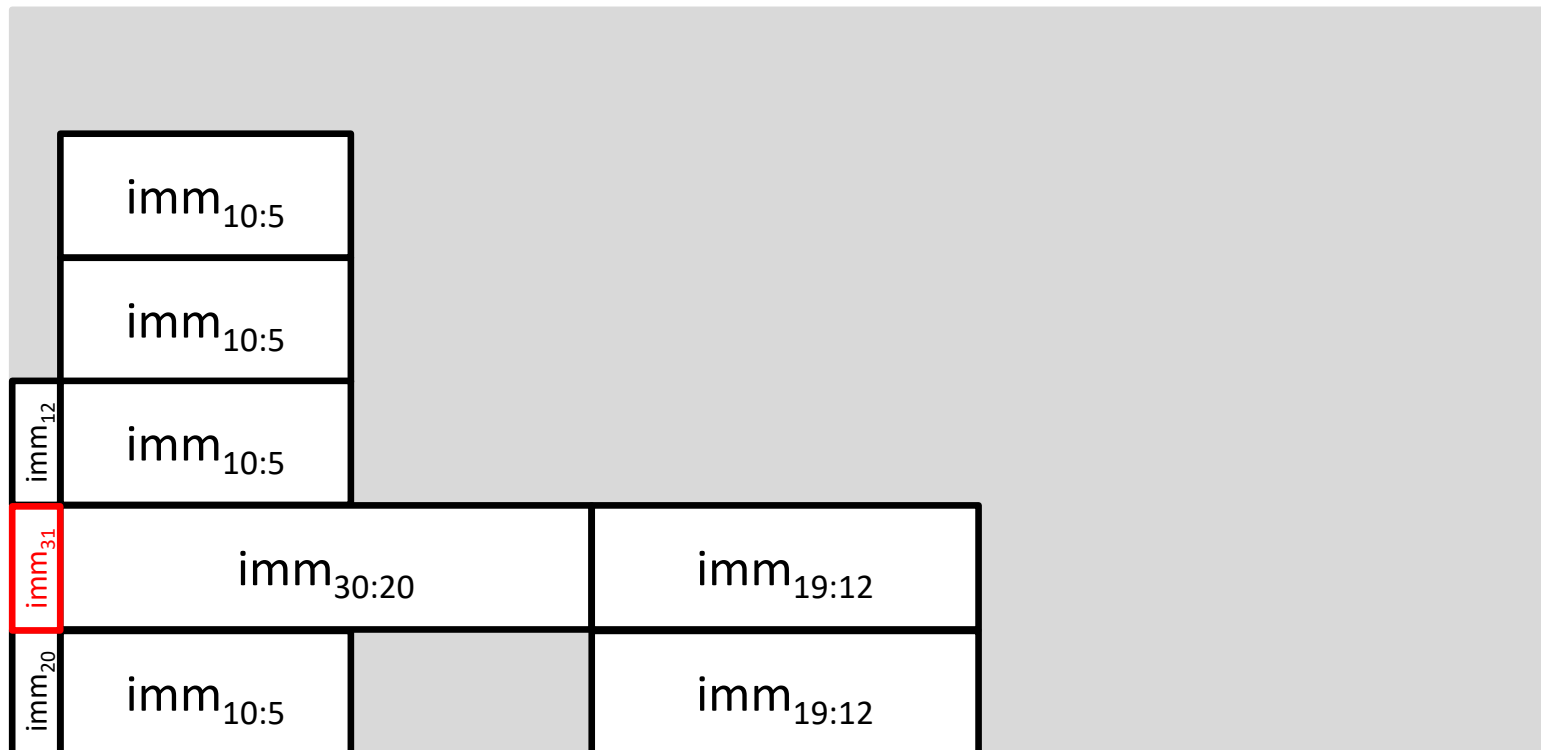


# Formatos de instrucción

## Resumen (ii)

- La ubicación de los operandos inmediatos está ideada para simplificar la lógica de extensión de signo.
  - Los mismos fragmentos suelen ocupar la misma posición.
  - Si no, se distribuyen en el menor número de posiciones diferentes posible.

31



*tipo-R*

*tipo-I*

*tipo-S*

*tipo-B*

*tipo-U*

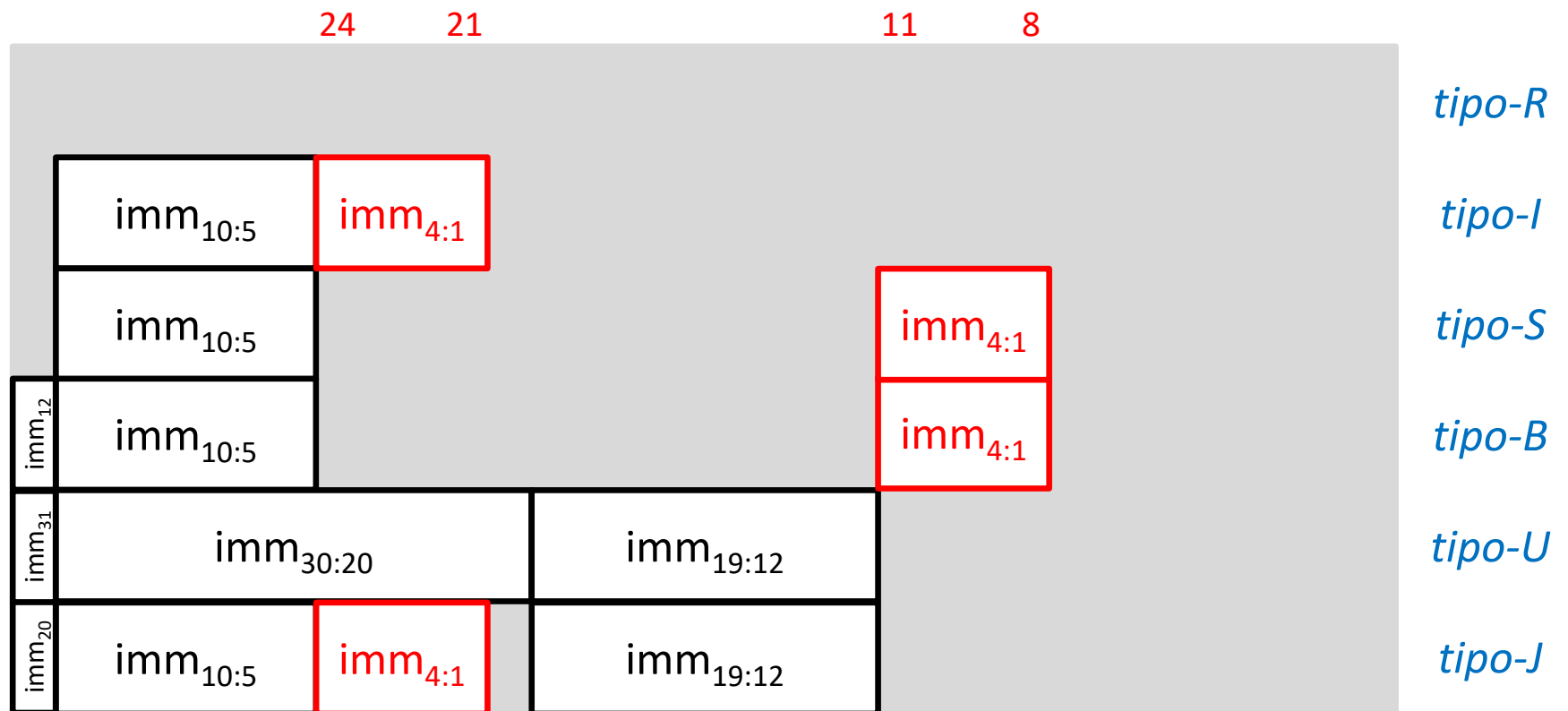
*tipo-J*



# Formatos de instrucción

## Resumen (ii)

- La ubicación de los operandos inmediatos está ideada para simplificar la lógica de extensión de signo.
  - Los mismos fragmentos suelen ocupar la misma posición.
  - Si no, se distribuyen en el menor número de posiciones diferentes posible.

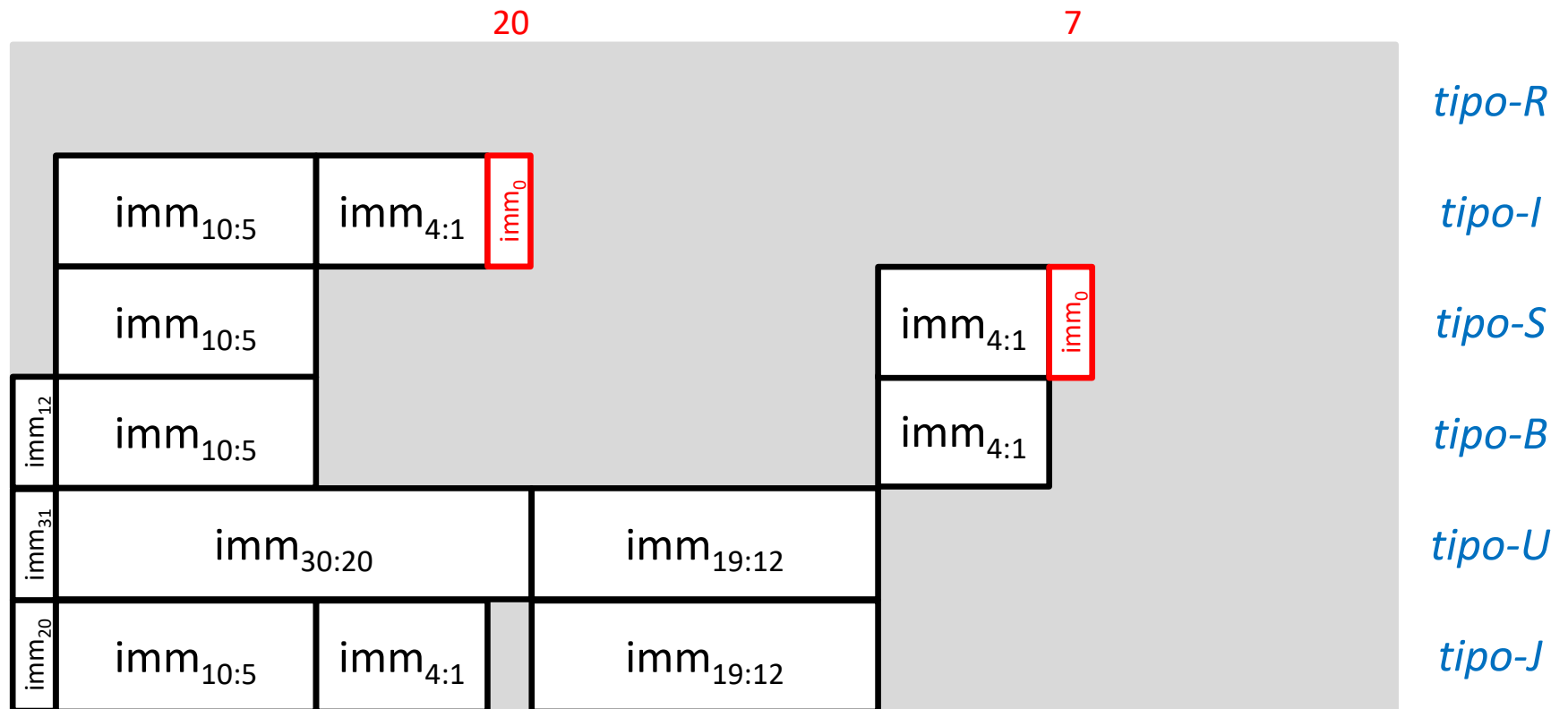




# Formatos de instrucción

## Resumen (ii)

- La ubicación de los operandos inmediatos está ideada para simplificar la lógica de extensión de signo.
  - Los mismos fragmentos suelen ocupar la misma posición.
  - Si no, se distribuyen en el menor número de posiciones diferentes posible.

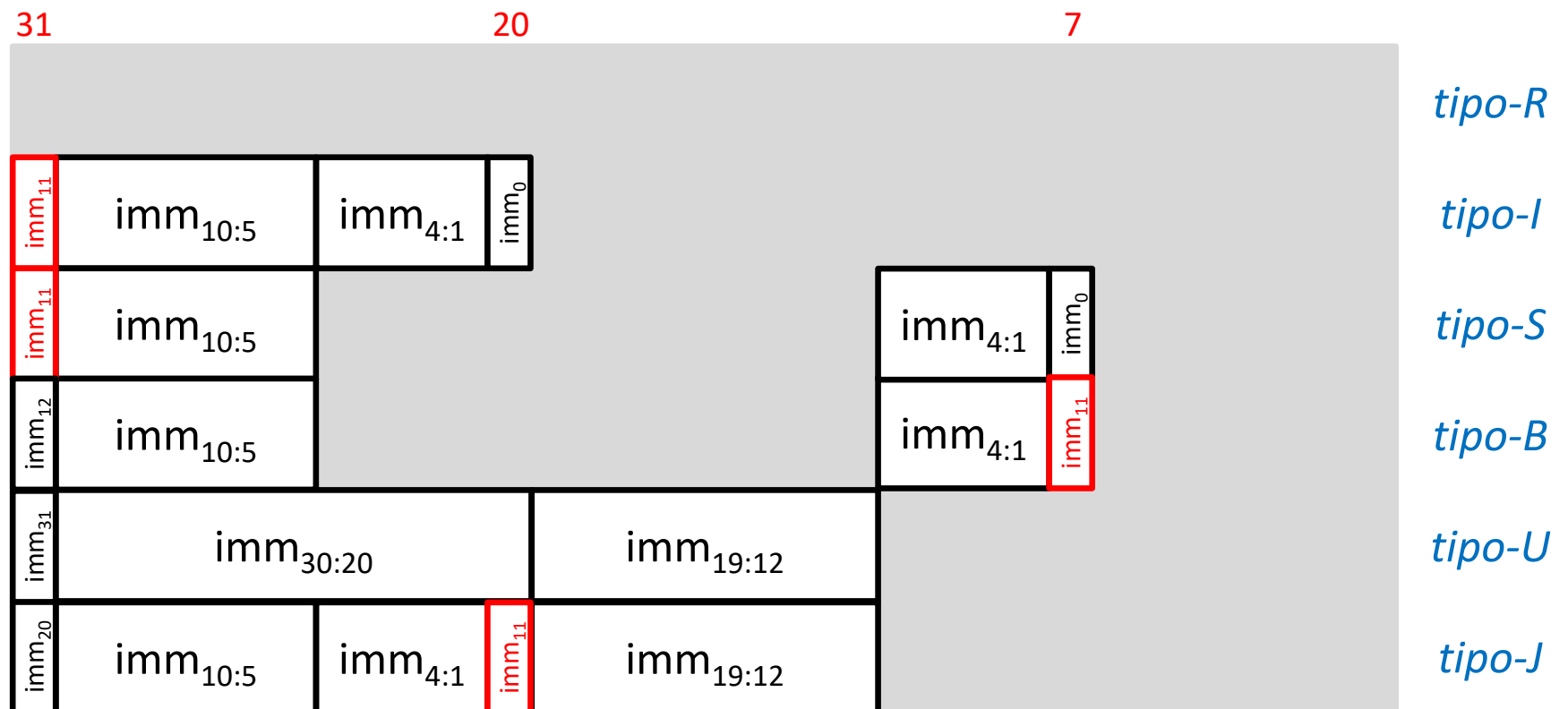




# Formatos de instrucción

## Resumen (ii)

- La ubicación de los operandos inmediatos está ideada para simplificar la lógica de extensión de signo.
  - Los mismos fragmentos suelen ocupar la misma posición.
  - Si no, se distribuyen en el menor número de posiciones diferentes posible.

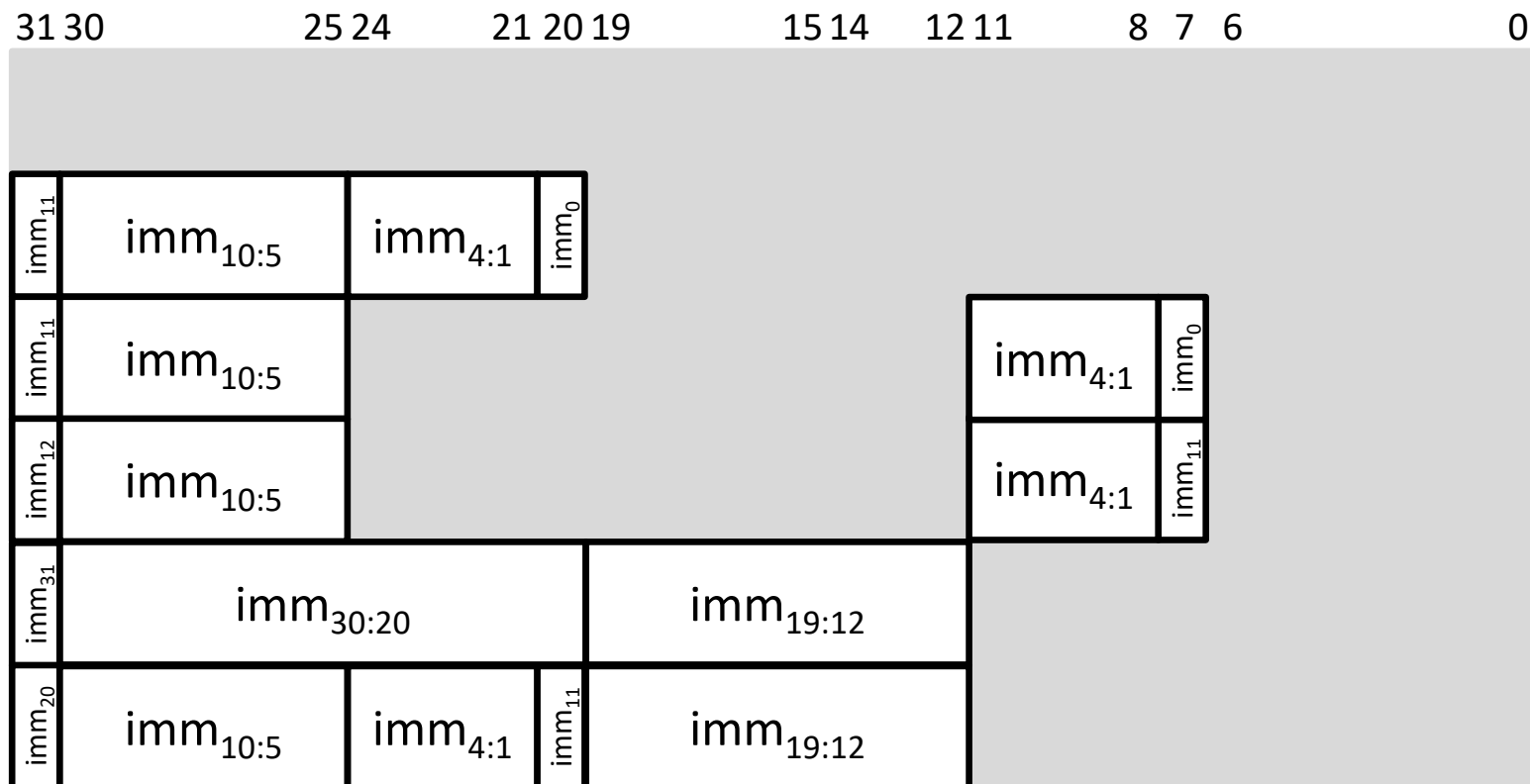




# Formatos de instrucción

## Resumen (ii)

- La ubicación de los operandos inmediatos está ideada para simplificar la lógica de extensión de signo.
  - Los mismos fragmentos suelen ocupar la misma posición.
  - Si no, se distribuyen en el menor número de posiciones diferentes posible.
  - Ubicar constantes es enrevesado, pero no es común ensamblar a mano.



*tipo-R*

*tipo-I*

*tipo-S*

*tipo-B*

*tipo-U*

*tipo-J*





# Codificación de campos

## Códigos de operación

| op      | Clase de instrucción  | Tipo |
|---------|---|------|
| 0000011 | carga de memoria  | I    |
| 0010011 | aritmético-lógicas y de desplazamiento con operando inmediato     | I    |
| 0010111 | <b>auipc</b>  | U    |
| 0100011 | almacenaje en memoria   | S    |
| 0110011 | aritmético-lógicas y de desplazamiento con operandos en registros | R    |
| 0110111 | <b>lui</b>  | U    |
| 1100011 | salto condicional   | B    |
| 1100111 | <b>jalr</b>   | I    |
| 1101111 | <b>jal</b>  | J    |



# Codificación de campos

## Códigos de función (i)

### Instrucciones de carga de memoria

| op      | funct3 | Instrucción | Tipo |
|---------|--------|-------------|------|
| 0000011 | 000    | lb          | I    |
|         | 001    | lh          | I    |
|         | 010    | lw          | I    |
|         | 011    | lbu         | I    |
|         | 100    | lhu         | I    |

### Instrucciones de almacenaje en memoria

| op      | funct3 | Instrucción | Tipo |
|---------|--------|-------------|------|
| 0100011 | 000    | sb          | S    |
|         | 001    | sh          | S    |
|         | 010    | sw          | S    |



# Codificación de campos

## Códigos de función (ii)

### Instrucciones aritmético-lógicas y de desplazamiento con operando inmediato

| op      | funct3 | funct7*     | Instrucción  | Tipo |
|---------|--------|-------------|--------------|------|
| 0010011 | 000    | -           | <b>addi</b>  |      |
|         | 001    | 0000000*    | <b>slli</b>  |      |
|         | 010    | -           | <b>slti</b>  |      |
|         | 011    | -           | <b>sltiu</b> |      |
|         | 100    | -           | <b>xori</b>  |      |
|         | 101    | 0000000*    | <b>srli</b>  |      |
|         | 101    | 0100000*    | <b>srai</b>  |      |
|         | 110    | -           | <b>ori</b>   |      |
| 111     | -      | <b>andi</b> |              |      |

\*Codificados en los 7 bits superiores del campo imm



# Codificación de campos

## Códigos de función (iii)

### Instrucciones aritmético-lógicas y de desplazamiento con operandos en registros

| op      | funct3 | funct7  | Instrucción | Tipo |
|---------|--------|---------|-------------|------|
| 0110011 | 000    | 0000000 | <b>add</b>  | R    |
|         | 000    | 0100000 | <b>sub</b>  | R    |
|         | 001    | 0000000 | <b>sll</b>  | R    |
|         | 010    | 0000000 | <b>slt</b>  | R    |
|         | 011    | 0000000 | <b>sltu</b> | R    |
|         | 100    | 0000000 | <b>xor</b>  | R    |
|         | 101    | 0000000 | <b>srl</b>  | R    |
|         | 101    | 0100000 | <b>sra</b>  | R    |
|         | 110    | 0000000 | <b>or</b>   | R    |
|         | 111    | 0000000 | <b>and</b>  | R    |



# Codificación de campos

## Códigos de función (iv)

### Instrucciones\* de multiplicación y división

| op      | funct3 | funct7  | Instrucción   | Tipo |
|---------|--------|---------|---------------|------|
| 0110011 | 000    | 0000001 | <b>mul</b>    | R    |
|         | 001    | 0000001 | <b>mulh</b>   | R    |
|         | 010    | 0000001 | <b>mulhsu</b> | R    |
|         | 011    | 0000001 | <b>mulhu</b>  | R    |
|         | 100    | 0000001 | <b>div</b>    | R    |
|         | 101    | 0000001 | <b>divu</b>   | R    |
|         | 110    | 0000001 | <b>rem</b>    | R    |
|         | 111    | 0000001 | <b>remu</b>   | R    |

\*Definidas en la extensión RVM



# Codificación de campos

## Códigos de función (v)

### Instrucciones de salto condicional

| op      | funct3 | Instrucción | Tipo |
|---------|--------|-------------|------|
| 1100011 | 000    | <b>beq</b>  | B    |
|         | 001    | <b>bne</b>  | B    |
|         | 100    | <b>blt</b>  | B    |
|         | 101    | <b>bge</b>  | B    |
|         | 110    | <b>bltu</b> | B    |
|         | 111    | <b>bgeu</b> | B    |



# Codificación de campos

## Códigos de registro

| Nombre | Número | Código |
|--------|--------|--------|
| zero   | x0     | 00000  |
| ra     | x1     | 00001  |
| sp     | x2     | 00010  |
| gp     | x3     | 00011  |
| tp     | x4     | 00100  |
| t0     | x5     | 00101  |
| t1     | x6     | 00110  |
| t2     | x7     | 00111  |
| s0/fp  | x8     | 01000  |
| s1     | x9     | 01001  |
| a0     | x10    | 01010  |
| a1     | x11    | 01011  |
| a2     | x12    | 01100  |
| a3     | x13    | 01101  |
| a4     | x14    | 01110  |
| a5     | x15    | 01111  |

| Nombre | Número | Código |
|--------|--------|--------|
| a6     | x16    | 10000  |
| a7     | x17    | 10001  |
| s2     | x18    | 10010  |
| s3     | x19    | 10011  |
| s4     | x20    | 10100  |
| s5     | x21    | 10101  |
| s6     | x22    | 10110  |
| s7     | x23    | 10111  |
| s8     | x24    | 11000  |
| s9     | x25    | 11001  |
| s10    | x26    | 11010  |
| s11    | x27    | 11011  |
| t3     | x28    | 11100  |
| t4     | x29    | 11101  |
| t5     | x30    | 11110  |
| t6     | x31    | 11111  |

# De ensamblador a código máquina

Ejemplo: instrucción tipo-R



`sub x5, x6, x7`

Ensamblado

`0x407302b3`

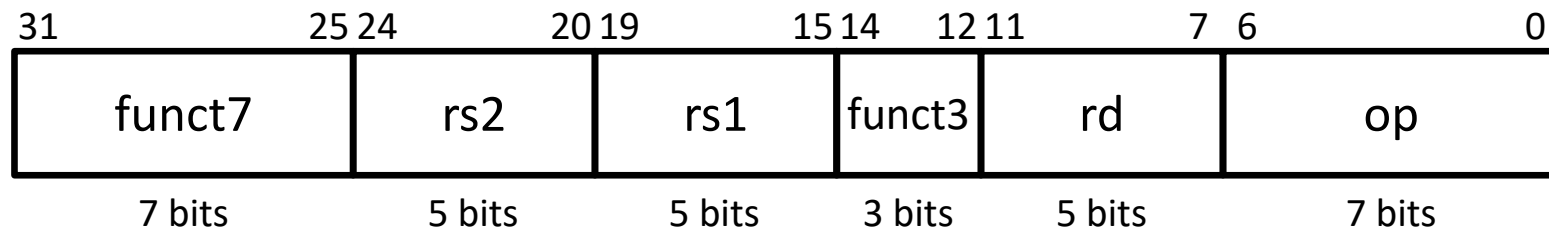




# De ensamblador a código máquina

Ejemplo: instrucción tipo-R

**sub x5, x6, x7**



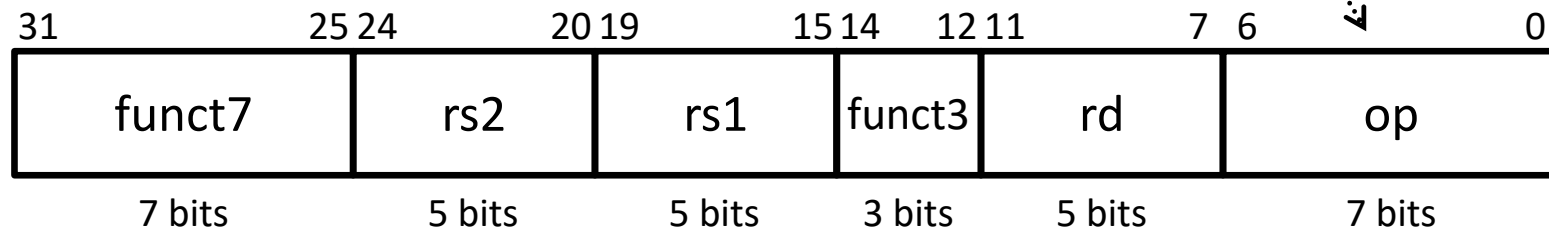
*tipo-R*



# De ensamblador a código máquina

## Ejemplo: instrucción tipo-R

`sub x5, x6, x7`



*tipo-R*

0100000

000

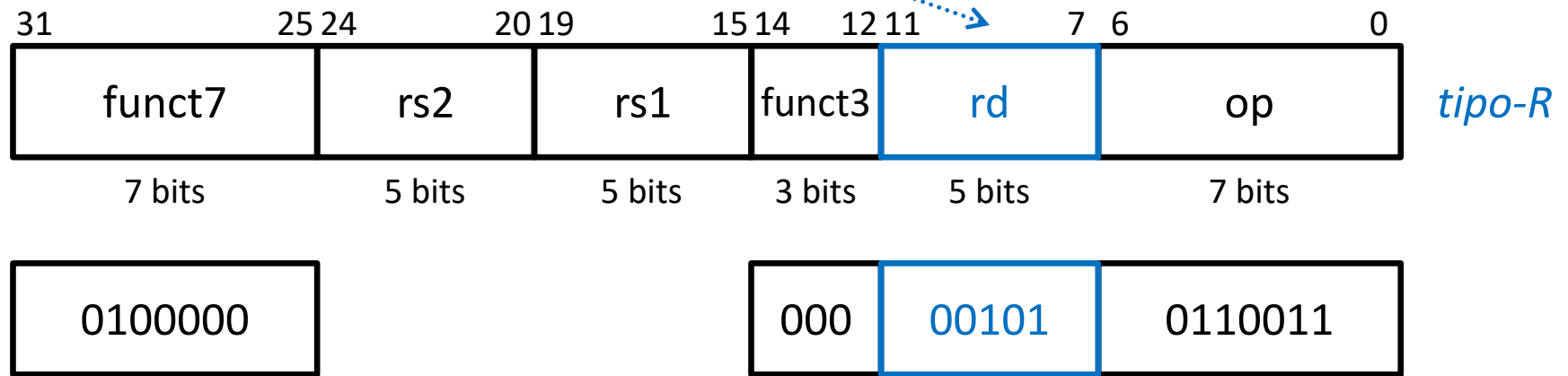
0110011



# De ensamblador a código máquina

## Ejemplo: instrucción tipo-R

sub **x5**, x6, x7

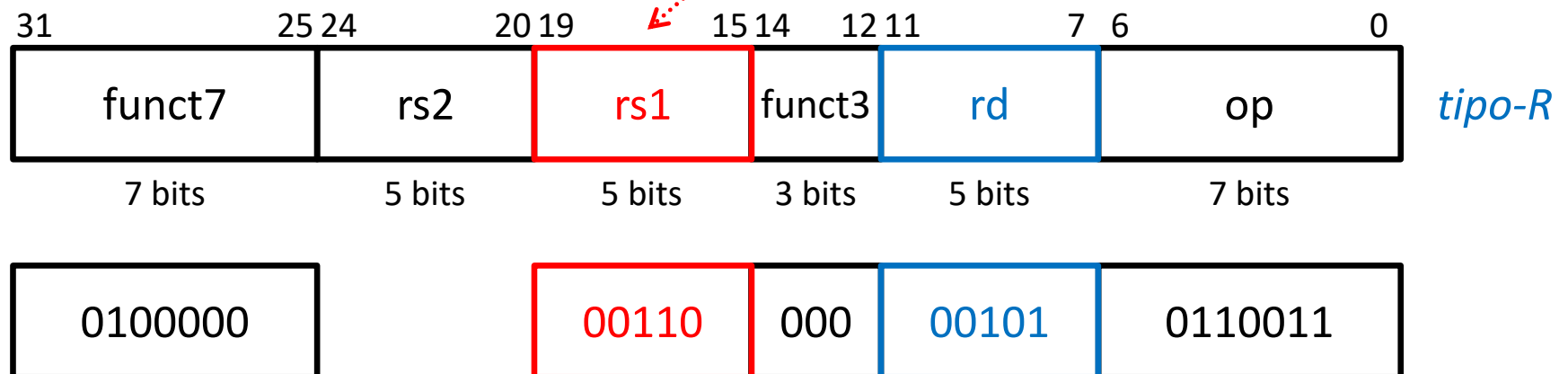




# De ensamblador a código máquina

## Ejemplo: instrucción tipo-R

sub x5, x6, x7

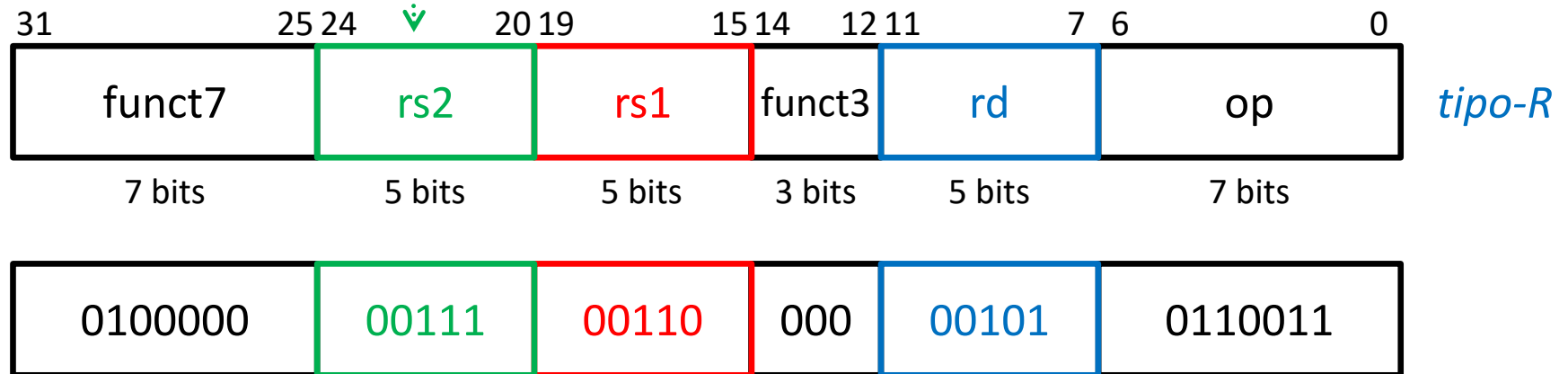




# De ensamblador a código máquina

Ejemplo: instrucción tipo-R

sub x5, x6, x7

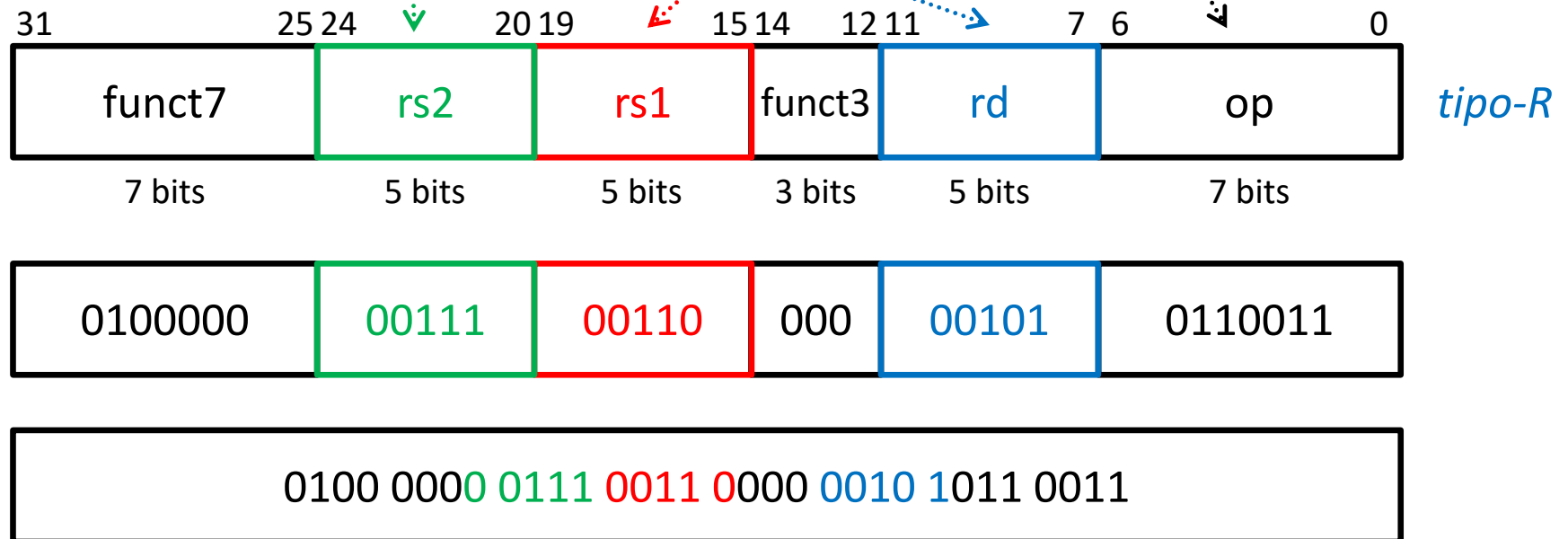




# De ensamblador a código máquina

## Ejemplo: instrucción tipo-R

sub x5, x6, x7



0x407302b3

# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (i)



```
addi s0, s1, 12
```

Ensamblado

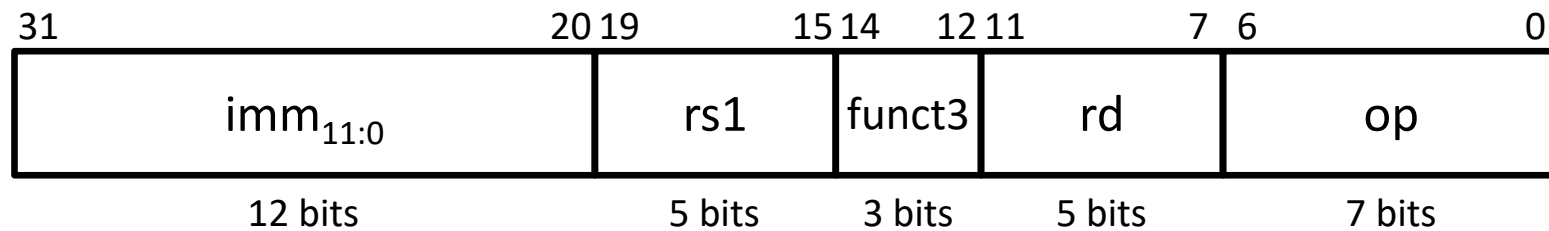
```
0x00C48413
```



# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (i)

`addi s0, s1, 12`



*tipo-I*

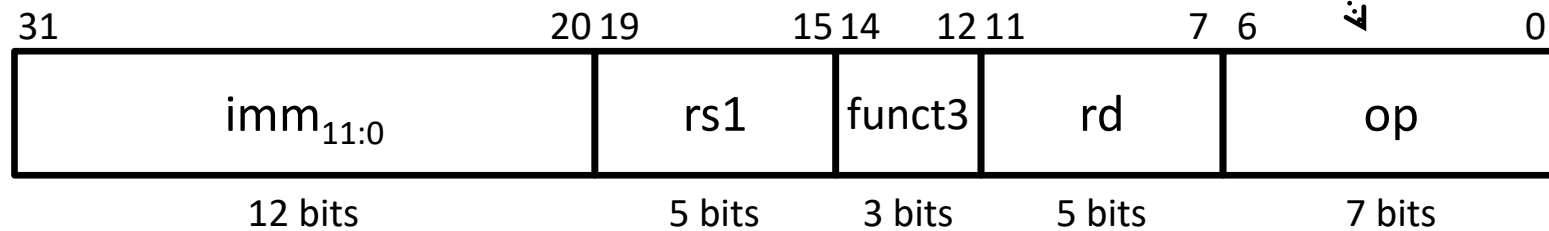




# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (i)

`addi s0, s1, 12`



000

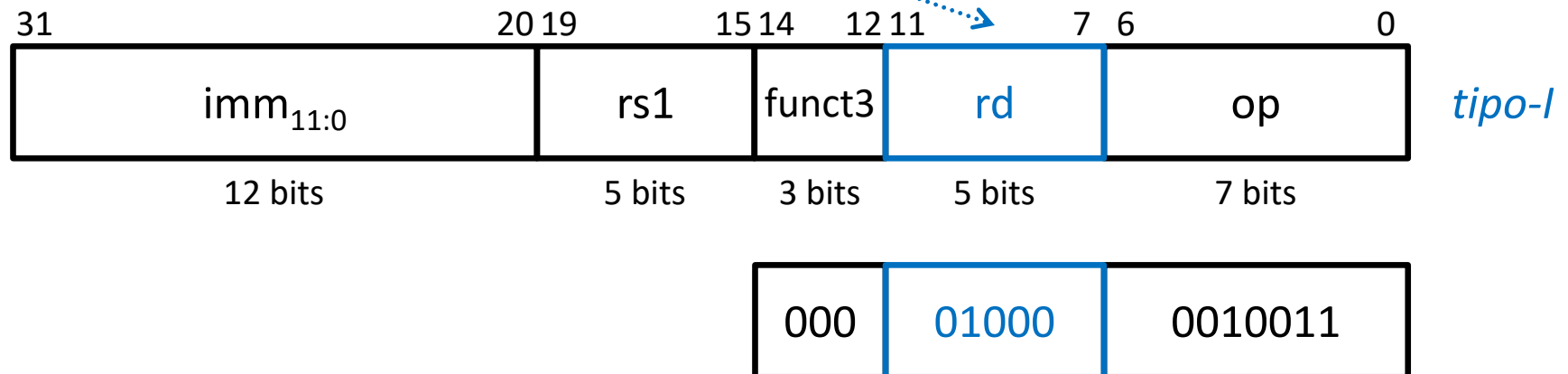
0010011



# De ensamblador a código máquina

## Ejemplo: instrucción tipo-I (i)

`addi s0, s1, 12`  $\equiv$  `addi x8, x9, 12`

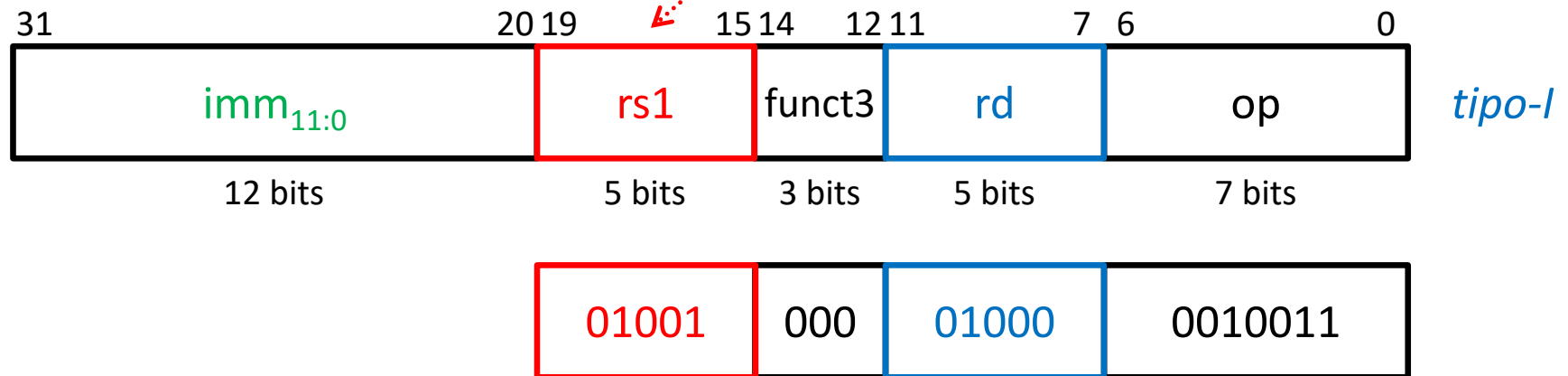




# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (i)

`addi s0, s1, 12`  $\equiv$  `addi x8, x9, 12`

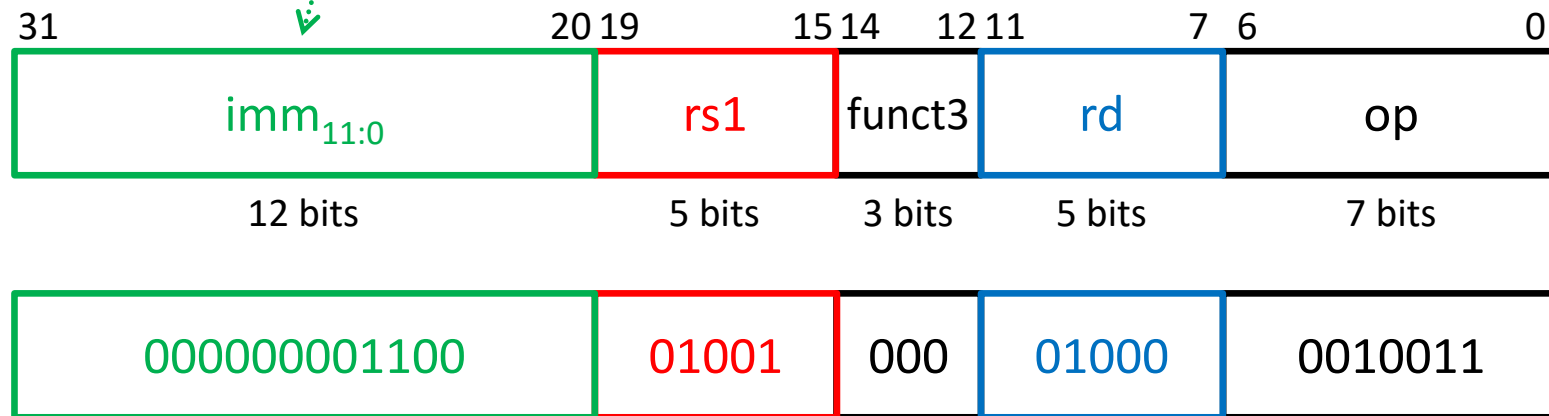




# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (i)

`addi s0, s1, 12`



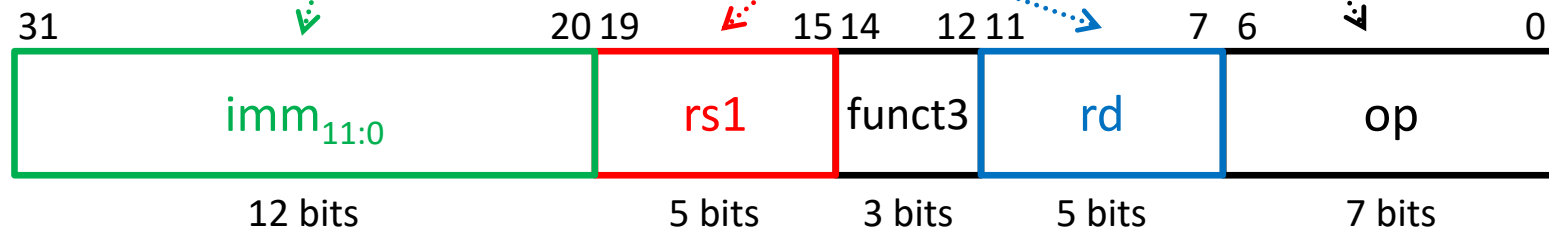
*tipo-I*



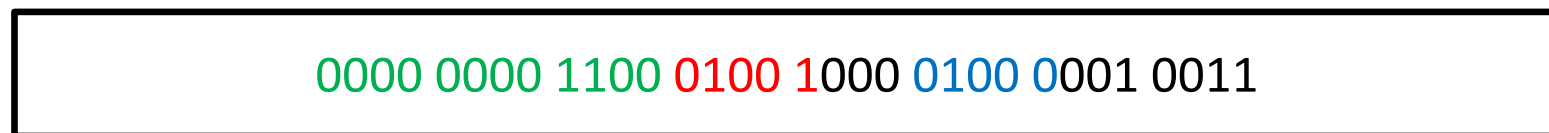
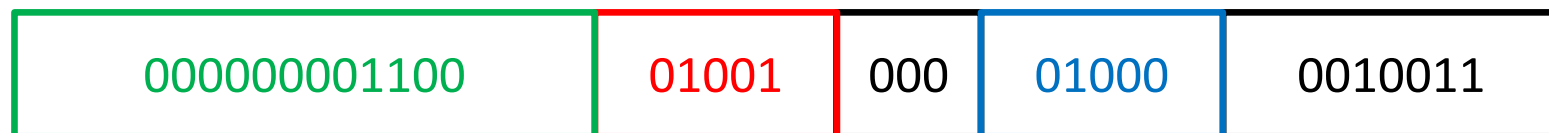
# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (i)

`addi s0, s1, 12`



*tipo-I*



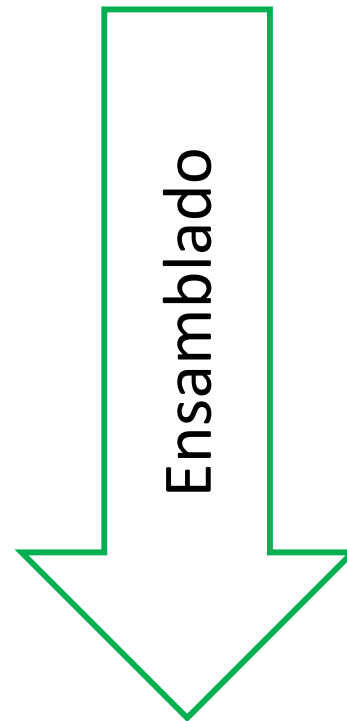
**0x00C48413**



# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (ii)

```
lw t2, -6(s3)
```



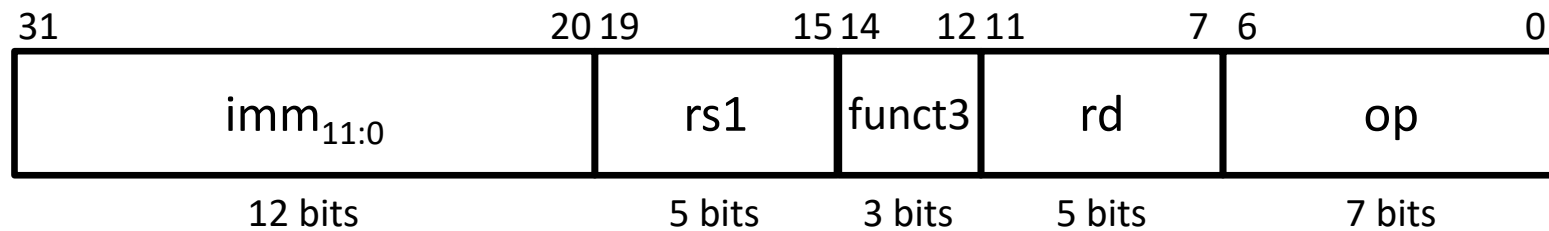
```
0xffa9a383
```



# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (ii)

`lw t2, -6(s3)`



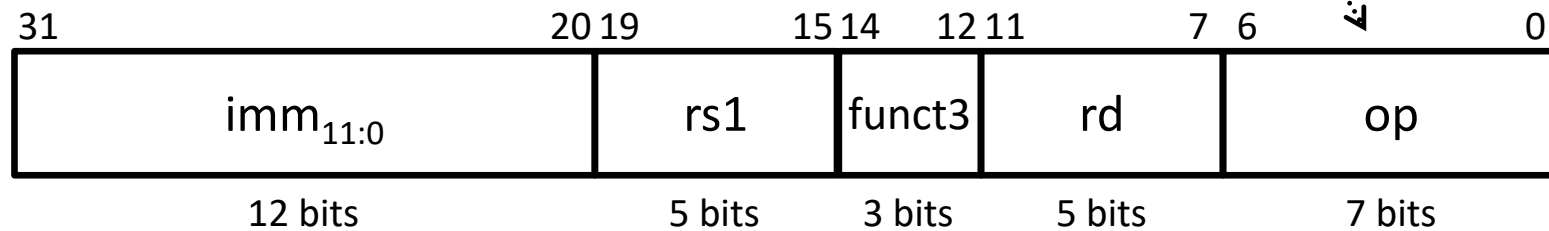
*tipo-I*



# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (ii)

`lw t2, -6(s3)`



*tipo-I*

010

0000011

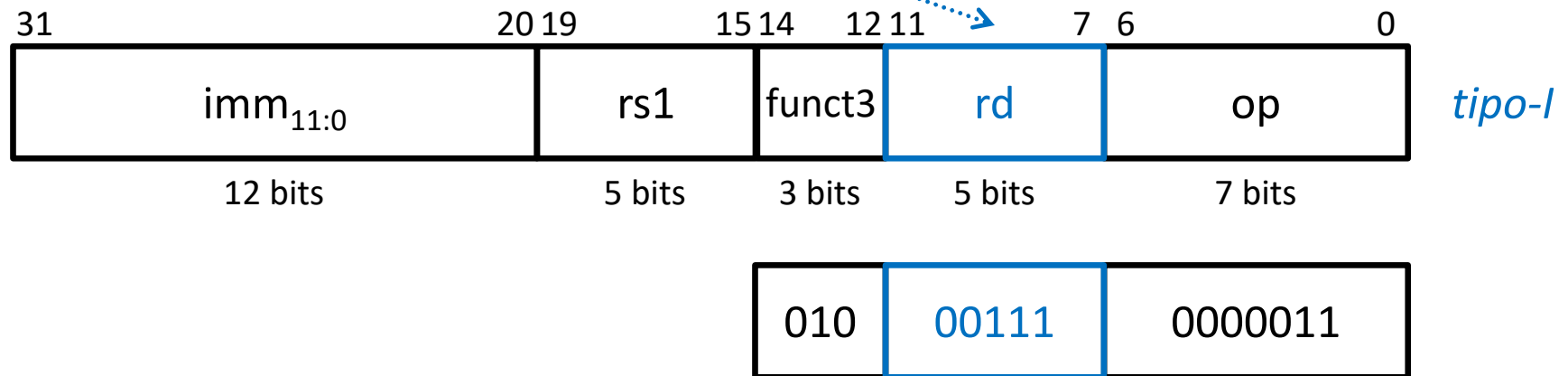




# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (ii)

`lw t2, -6(s3) ≡ lw x7, -6(x19)`

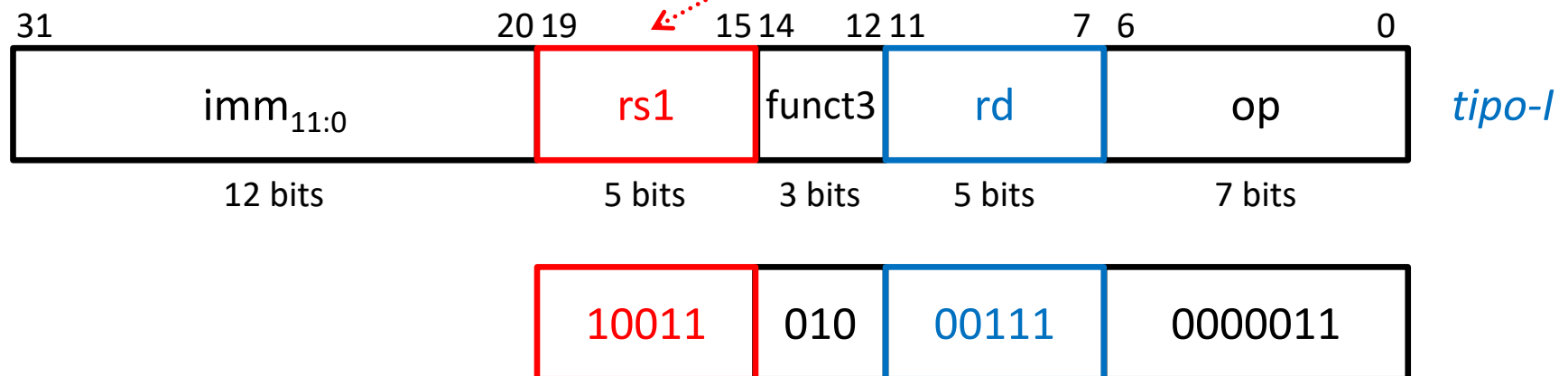




# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (ii)

`lw t2, -6(s3) ≡ lw x7, -6(x19)`

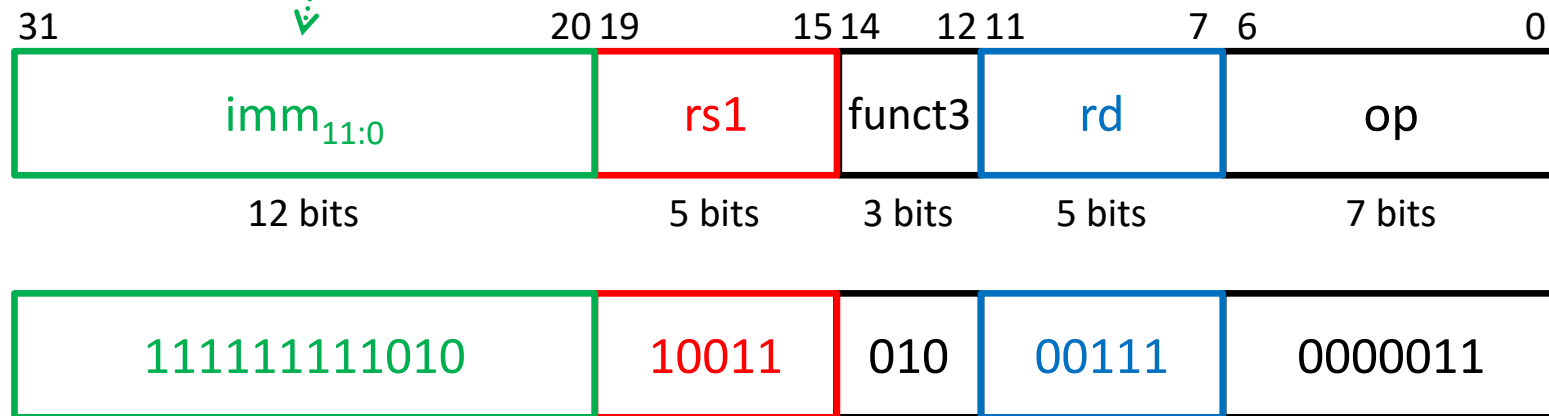




# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (ii)

`lw t2, -6(s3)`



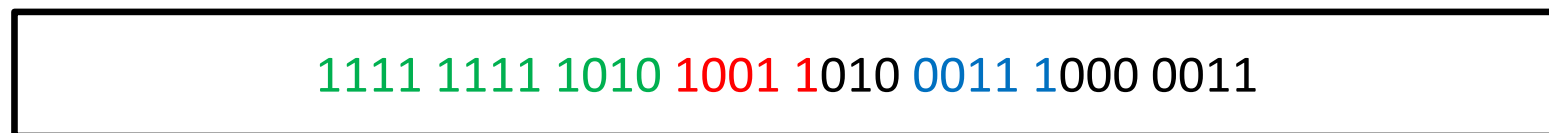
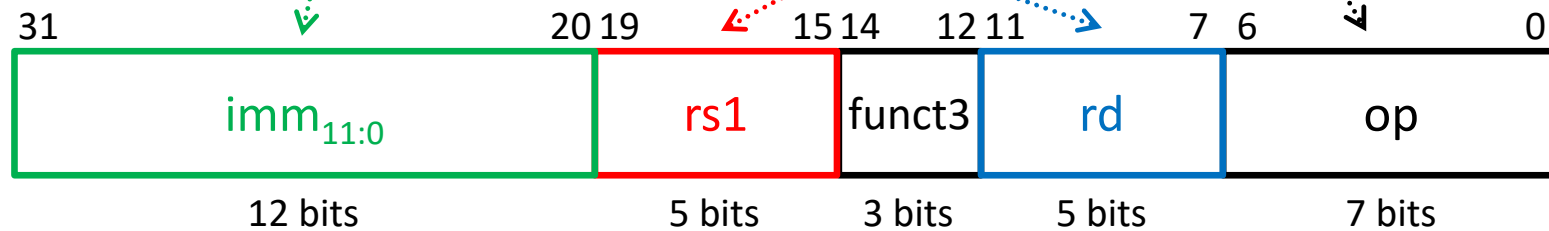
*tipo-I*



# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (ii)

`lw t2, -6(s3)`



`0xffa9a383`

# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (iii)



```
srai t1, t2, 29
```

Ensamblado

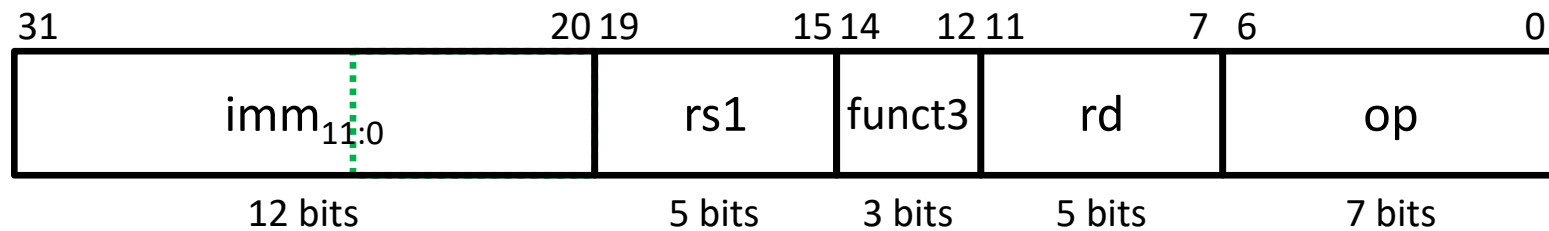
```
0x41d3d313
```



# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (iii)

```
srai t1, t2, 29
```



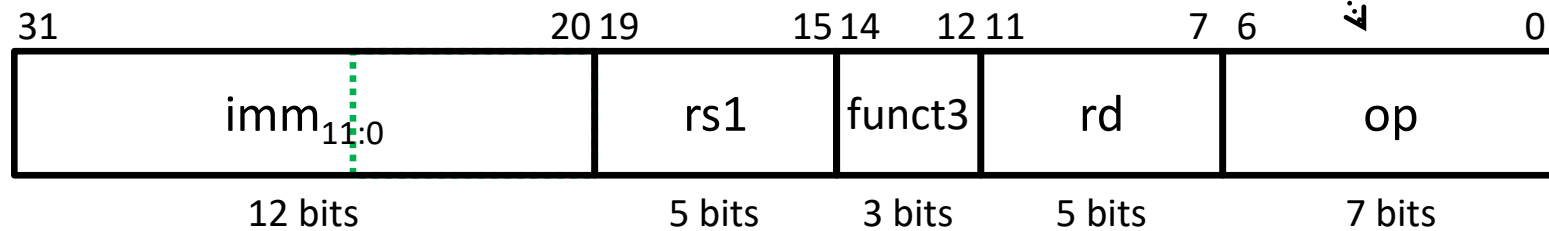
*tipo-I*



# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (iii)

`srai t1, t2, 29`



*tipo-I*

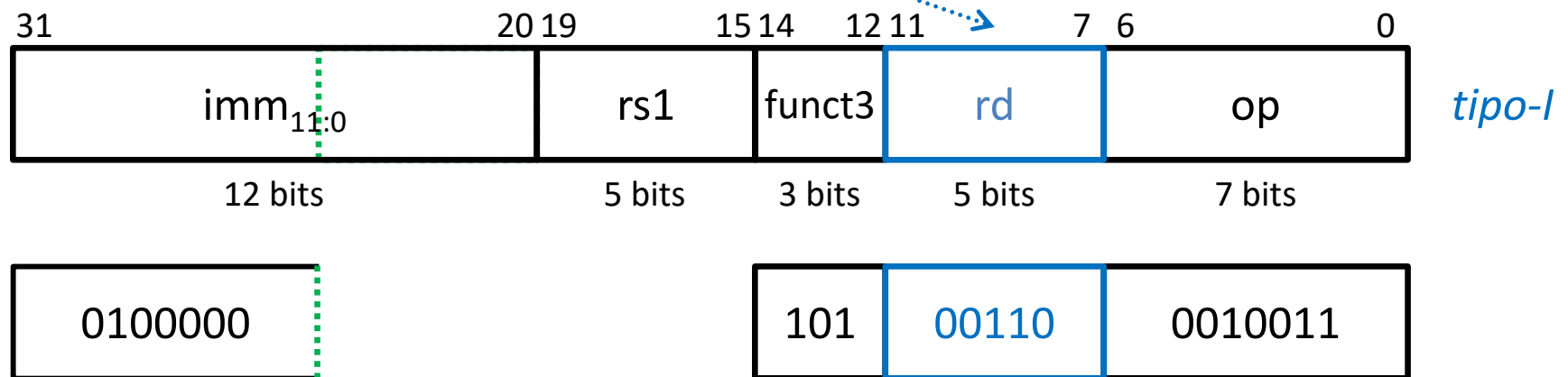




# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (iii)

`srai t1, t2, 29`  $\equiv$  `srai x6, x7, 29`



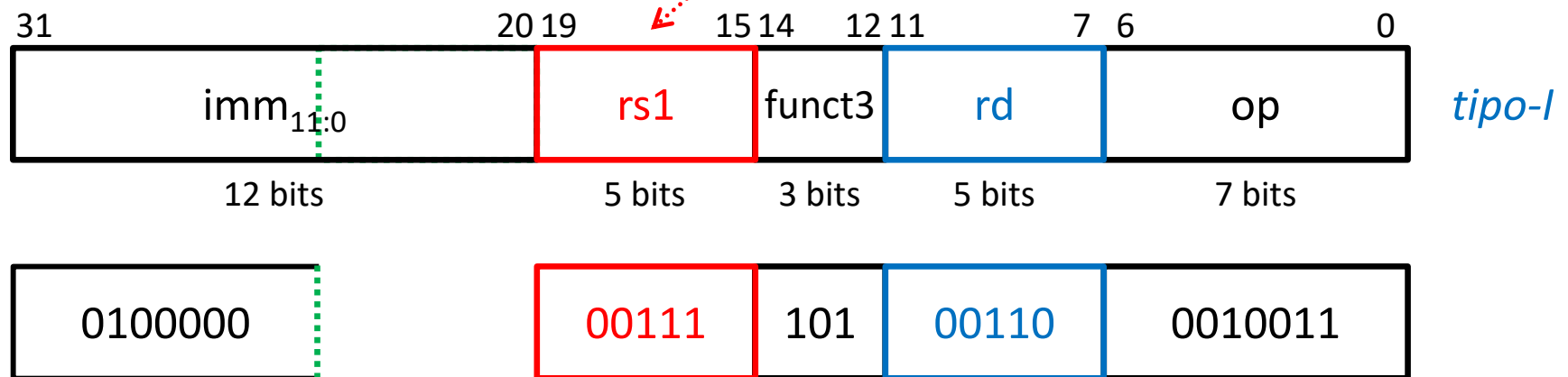




# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (iii)

`srai t1, t2, 29`  $\equiv$  `srai x6, x7, 29`

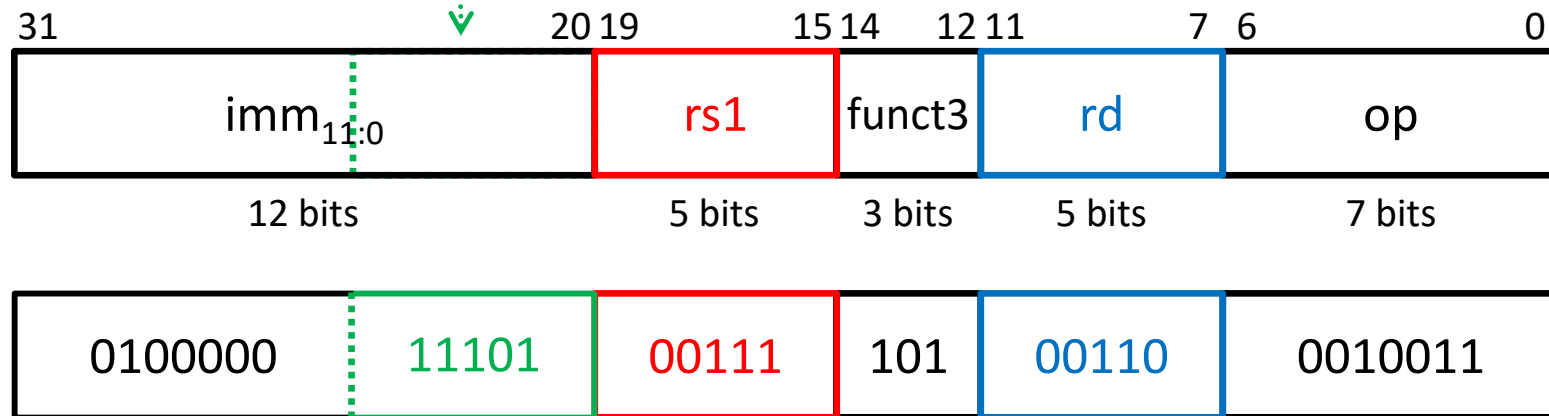




# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (iii)

`srai t1, t2, 29`

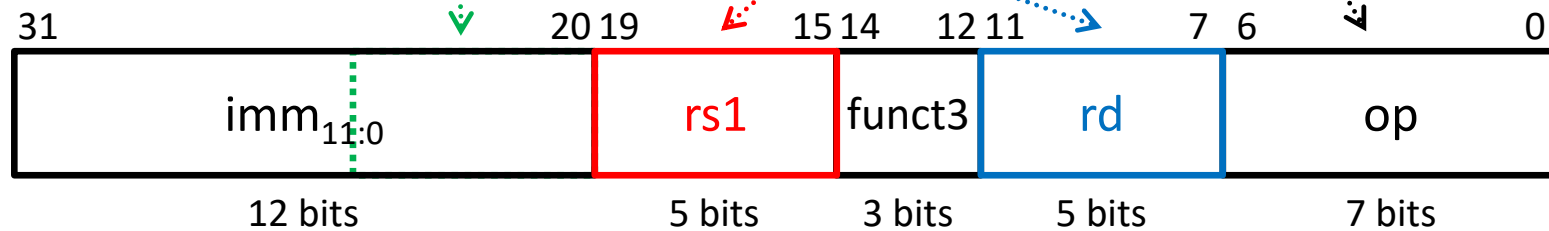




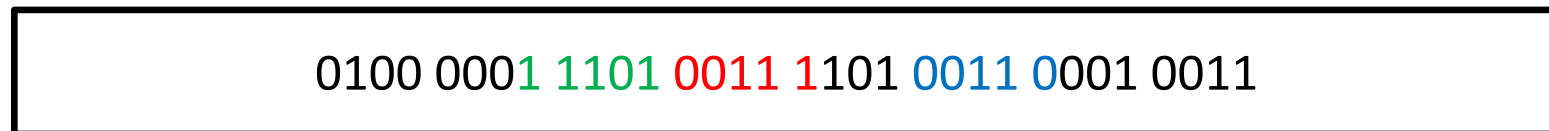
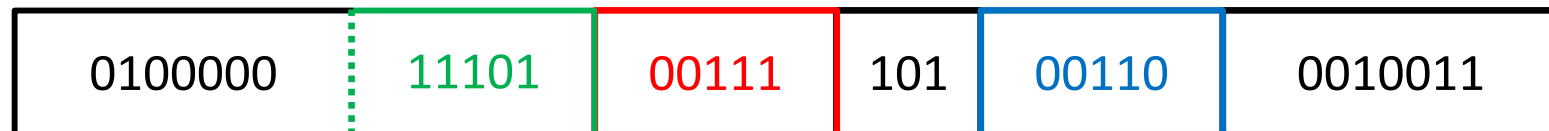
# De ensamblador a código máquina

Ejemplo: instrucción tipo-I (iii)

`srai t1, t2, 29`



*tipo-I*



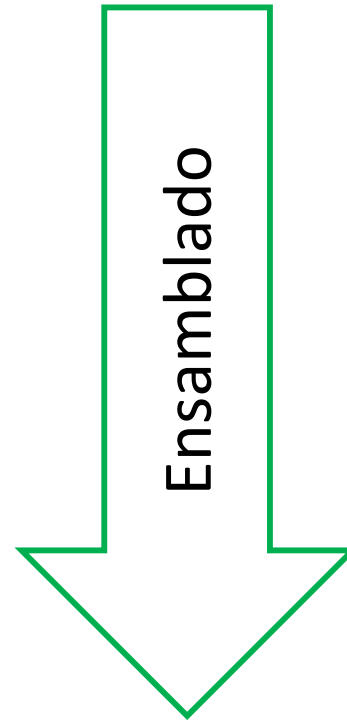
`0x41d3d313`

# De ensamblador a código máquina

Ejemplo: instrucción tipo-S



```
sb t5, 45 (zero)
```



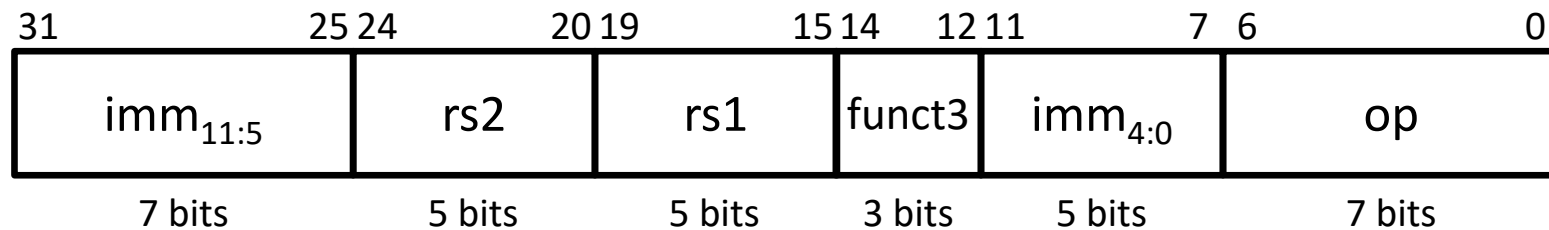
```
0x03e006a3
```



# De ensamblador a código máquina

## Ejemplo: instrucción tipo-S

`sb t5, 45(zero)`



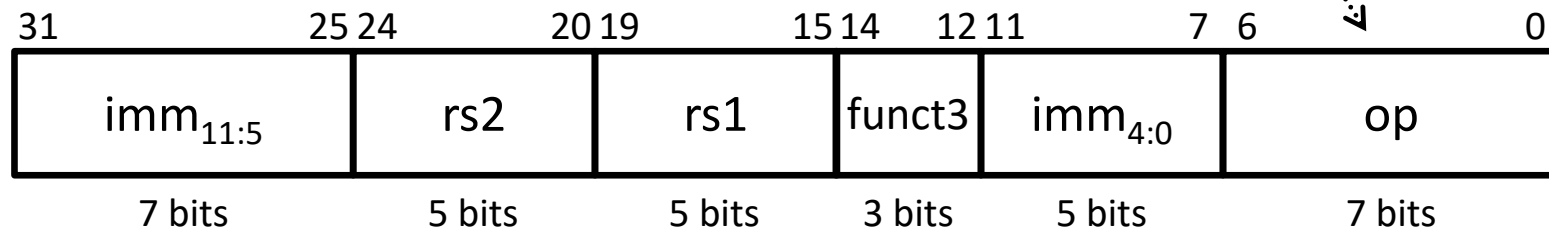
*tipo-S*



# De ensamblador a código máquina

Ejemplo: instrucción tipo-S

`sb t5, 45(zero)`



*tipo-S*

000

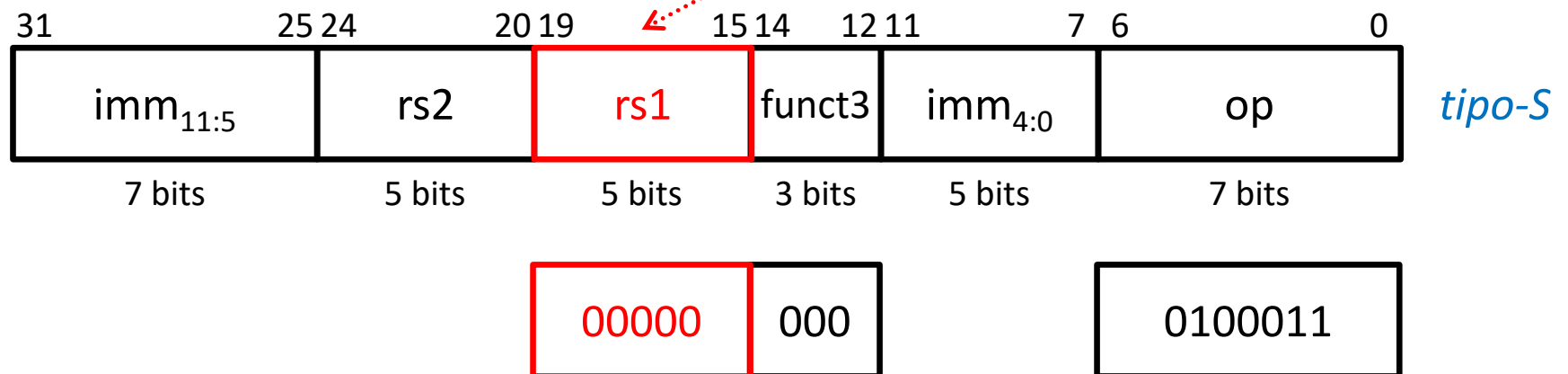
0100011



# De ensamblador a código máquina

## Ejemplo: instrucción tipo-S

`sb t5, 45 (zero) ≡ sb x30, x0, 45`

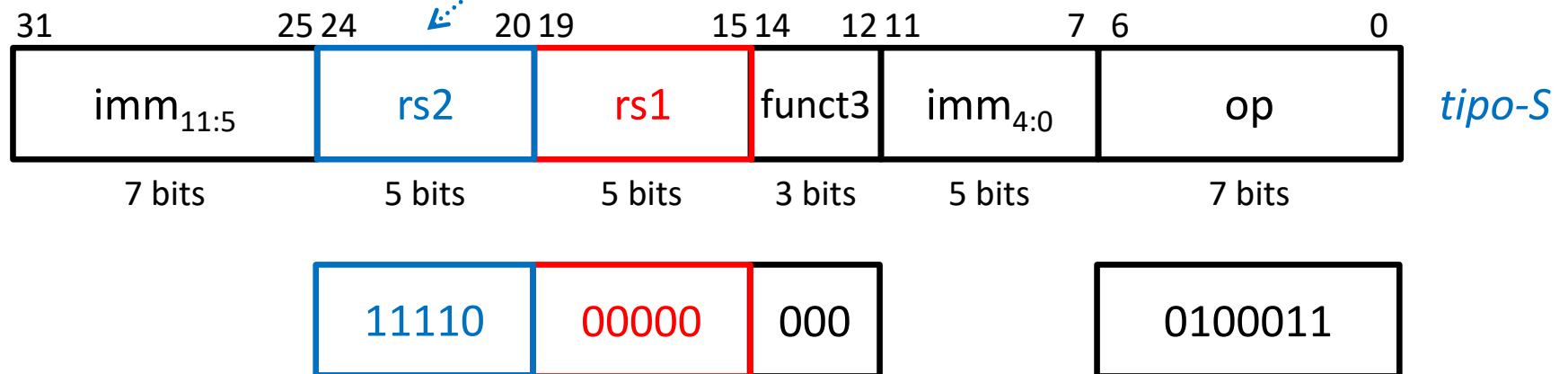




# De ensamblador a código máquina

## Ejemplo: instrucción tipo-S

`sb t5, 45 (zero) ≡ sb x30, x0, 45`





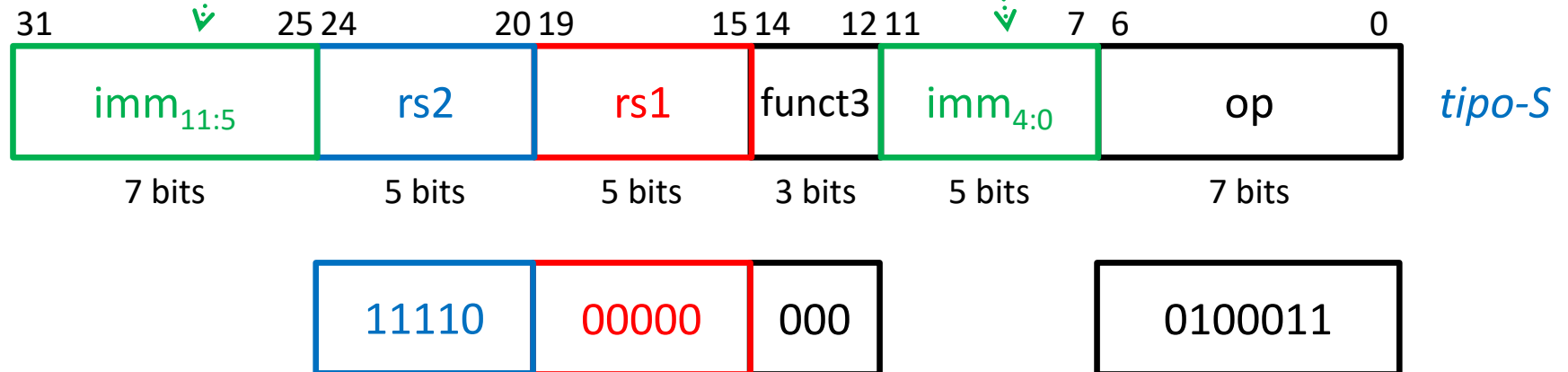


# De ensamblador a código máquina

## Ejemplo: instrucción tipo-S

**sb** t5, 45 (zero)

45 ≡ 0b000000101101



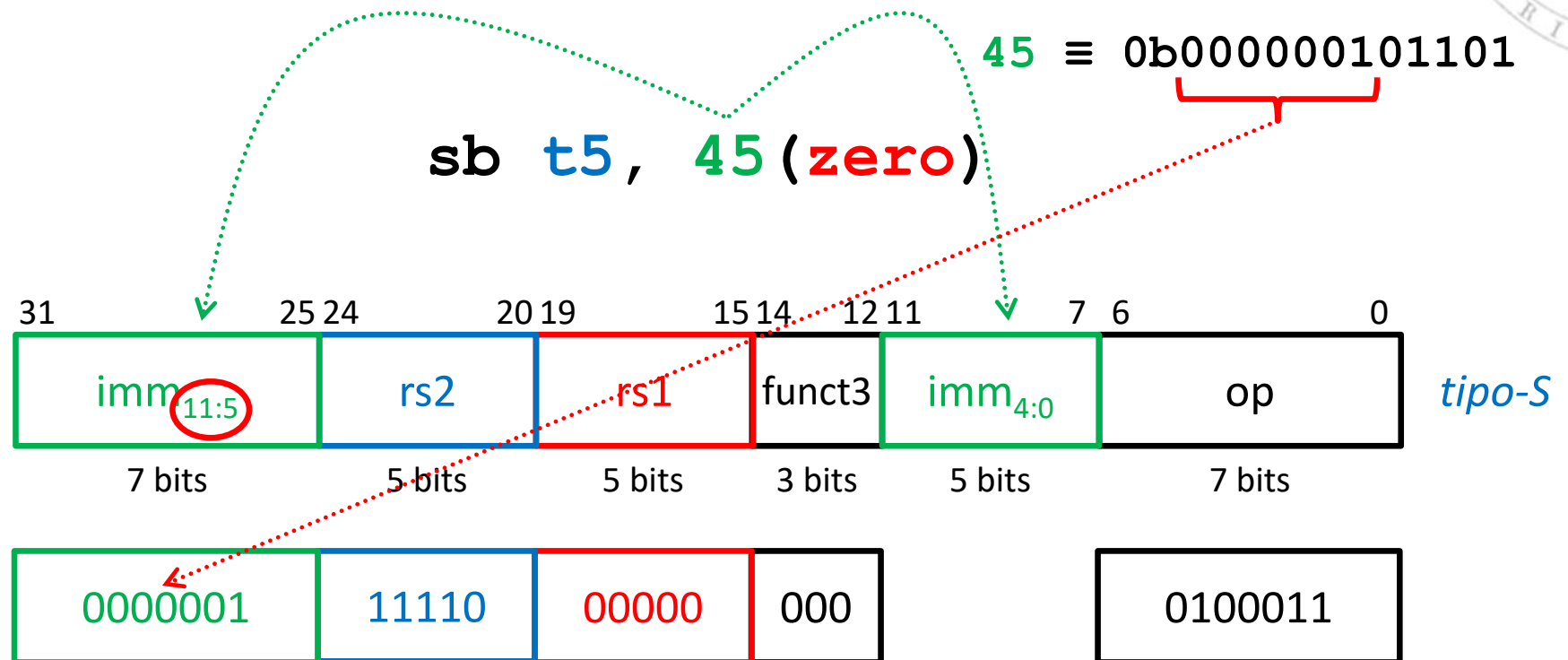


# De ensamblador a código máquina

## Ejemplo: instrucción tipo-S

`sb t5, 45 (zero)`

45 ≡ 0b000000101101



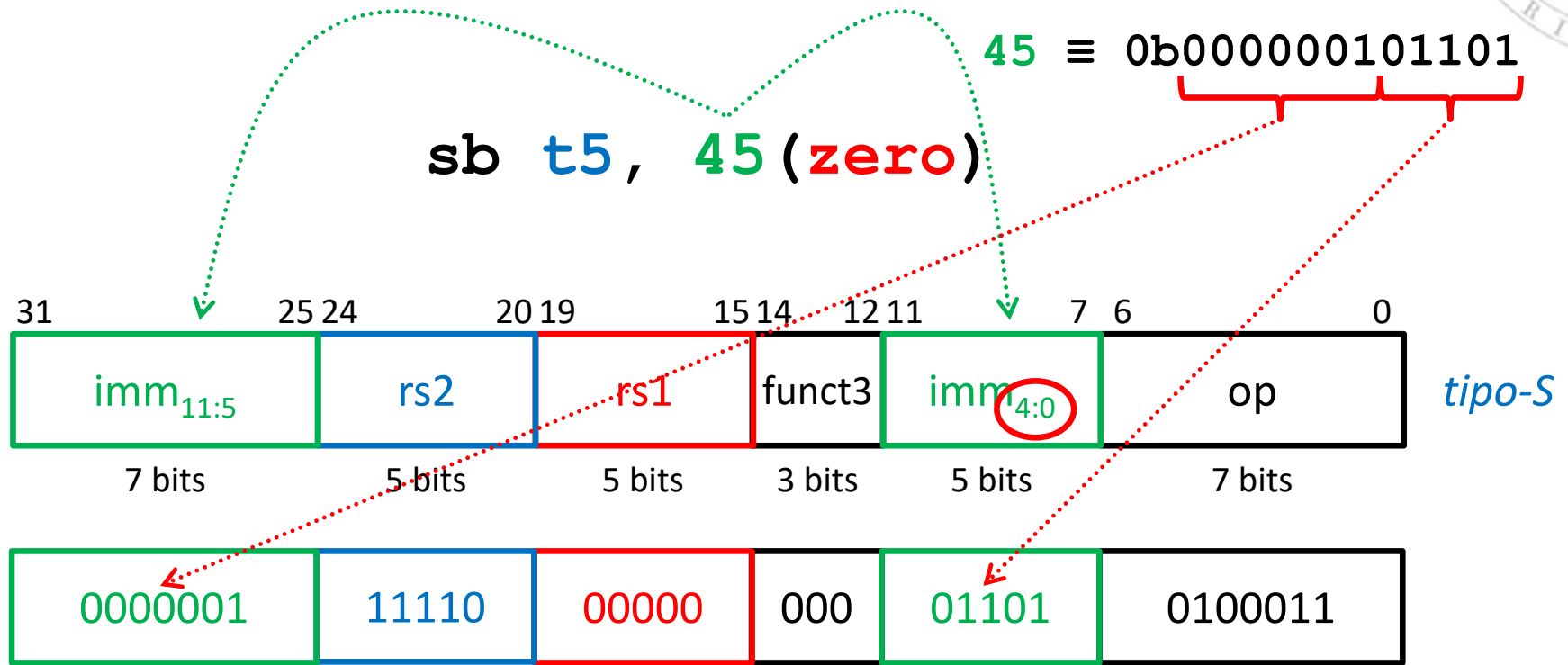


# De ensamblador a código máquina

## Ejemplo: instrucción tipo-S

`sb t5, 45 (zero)`

45 ≡ 0b000000101101

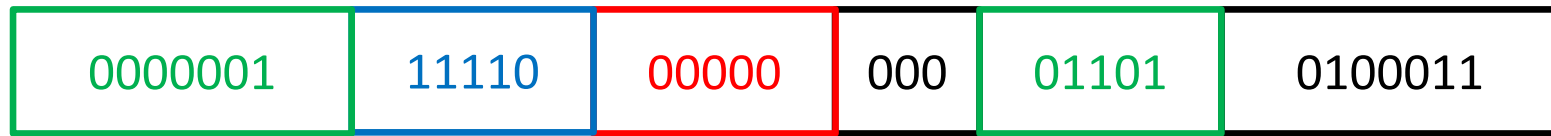
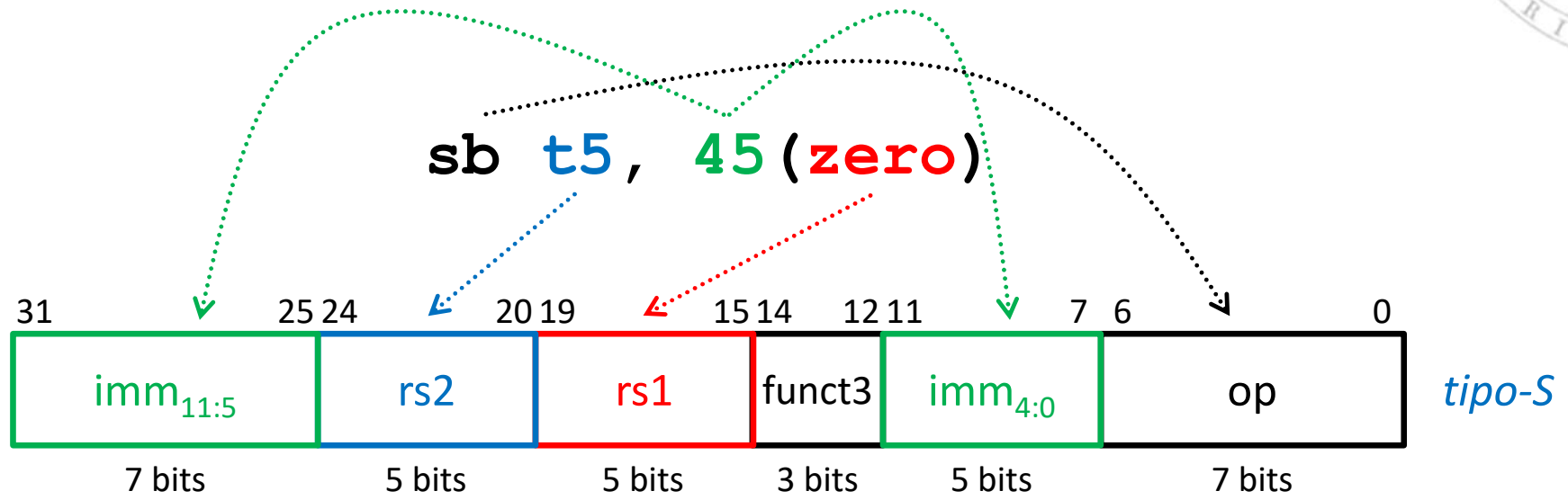




# De ensamblador a código máquina

## Ejemplo: instrucción tipo-S

**sb** t5, 45 (zero)



0x03e006a3

# De ensamblador a código máquina

Ejemplo: instrucción tipo-B



```
beq s0, t5, 0x10
```

Ensamblado

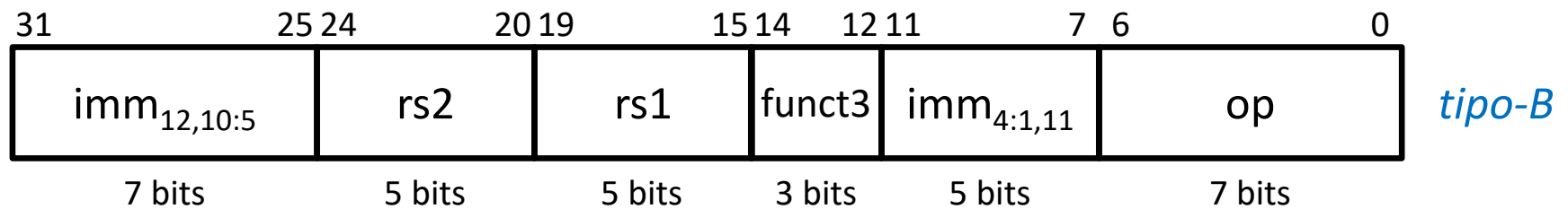
```
0x01e40863
```



# De ensamblador a código máquina

## Ejemplo: instrucción tipo-B

`beq s0, t5, 0x10`

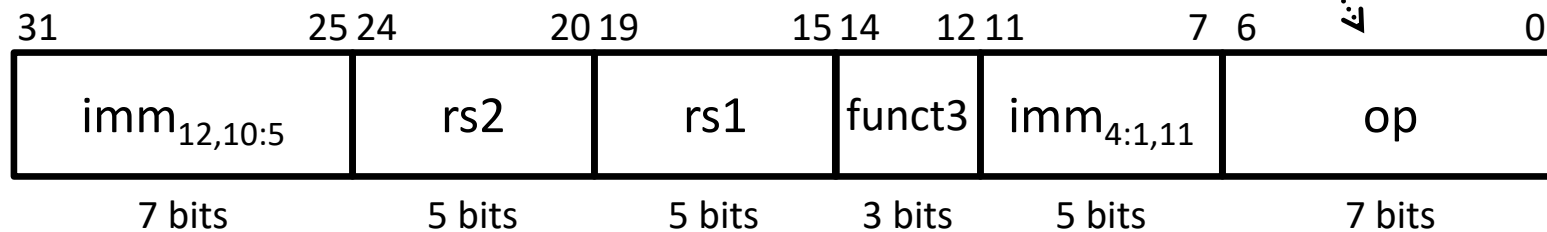




# De ensamblador a código máquina

Ejemplo: instrucción tipo-B

`beq s0, t5, 0x10`



*tipo-B*

000

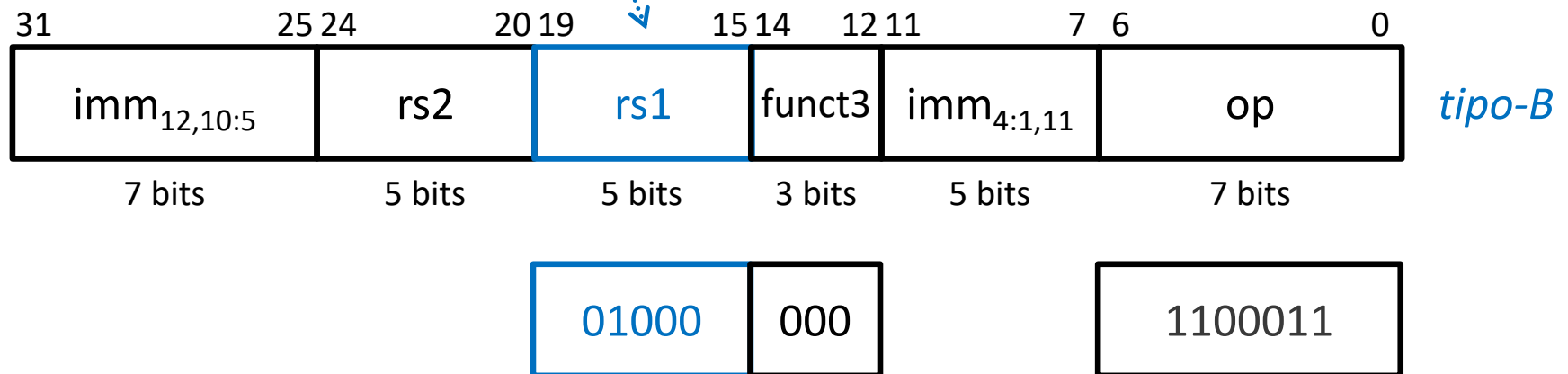
1100011



# De ensamblador a código máquina

## Ejemplo: instrucción tipo-B

`beq s0, t5, 0x10`  $\equiv$  `beq x8, x30, 0x10`



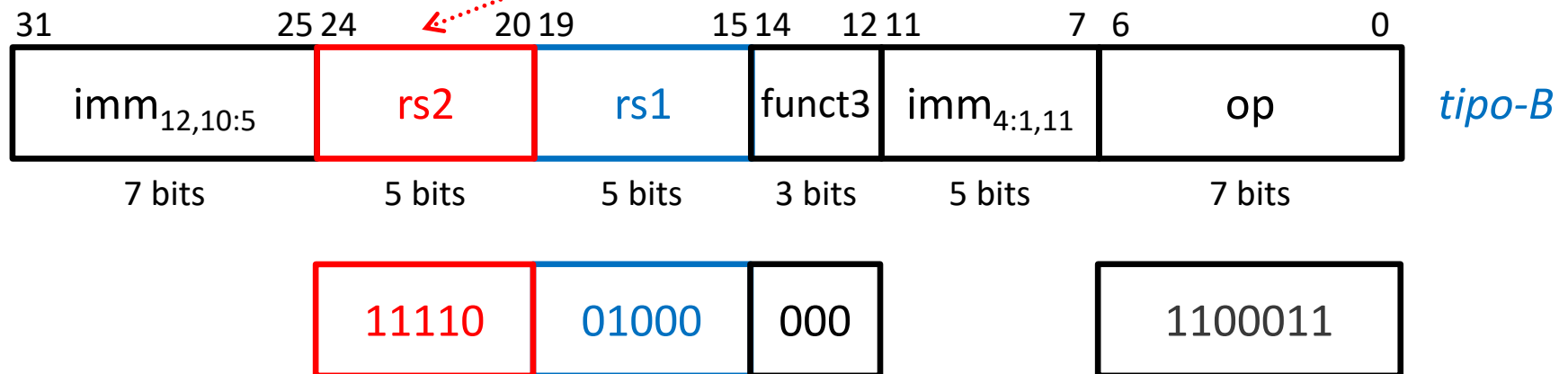




# De ensamblador a código máquina

## Ejemplo: instrucción tipo-B

`beq s0, t5, 0x10`  $\equiv$  `beq x8, x30, 0x10`



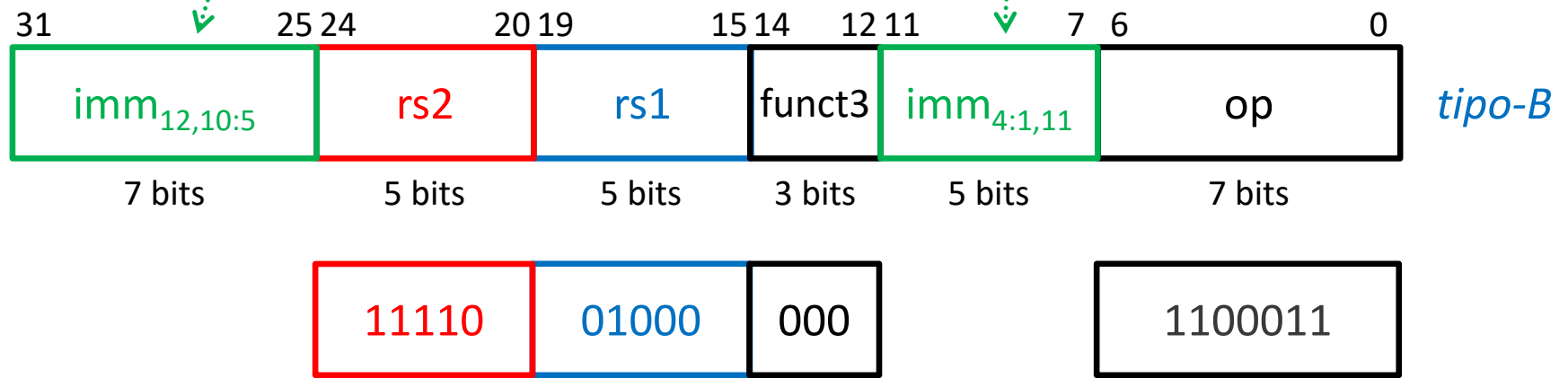


# De ensamblador a código máquina

## Ejemplo: instrucción tipo-B

0x10 ≡ 0b0000000010000

beq s0, t5, 0x10



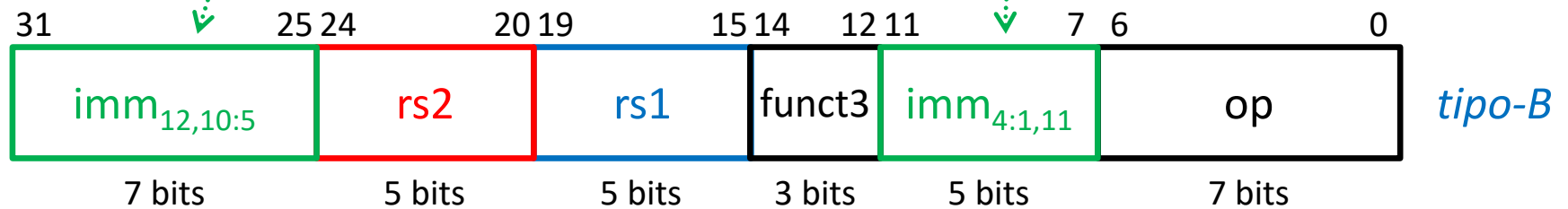


# De ensamblador a código máquina

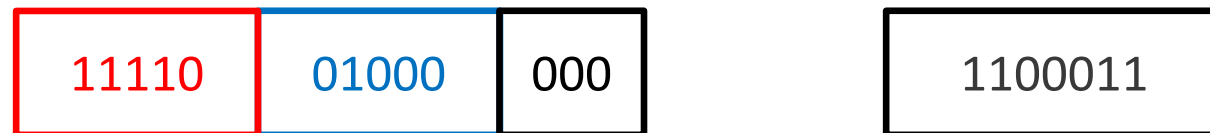
## Ejemplo: instrucción tipo-B

0x10 ≡ 0b0000000010000

beq s0, t5, 0x10



tipo-B



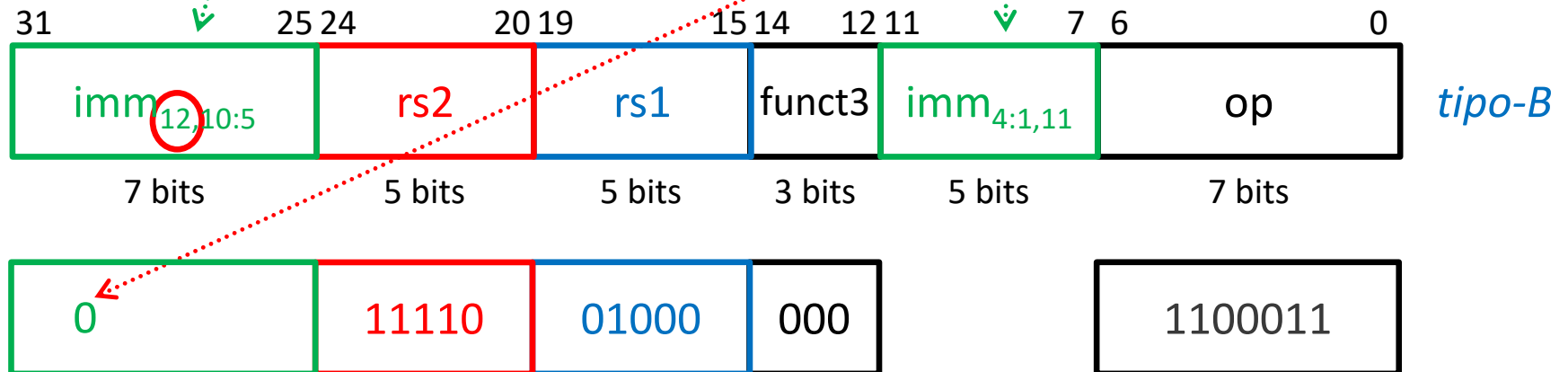


# De ensamblador a código máquina

## Ejemplo: instrucción tipo-B

0x10 ≡ 0b0000000010000

beq s0, t5, 0x10



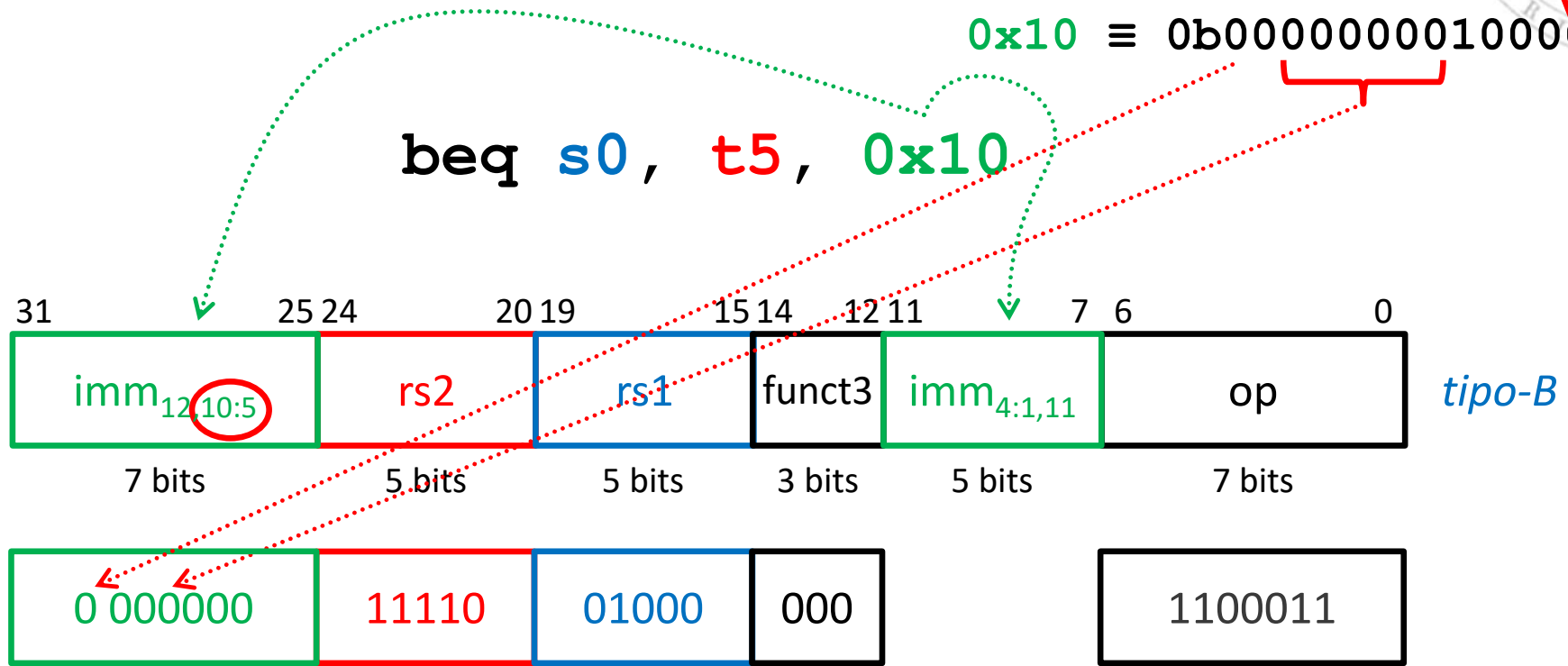


# De ensamblador a código máquina

## Ejemplo: instrucción tipo-B

`beq s0, t5, 0x10`

`0x10`  $\equiv$  `0b0000000010000`



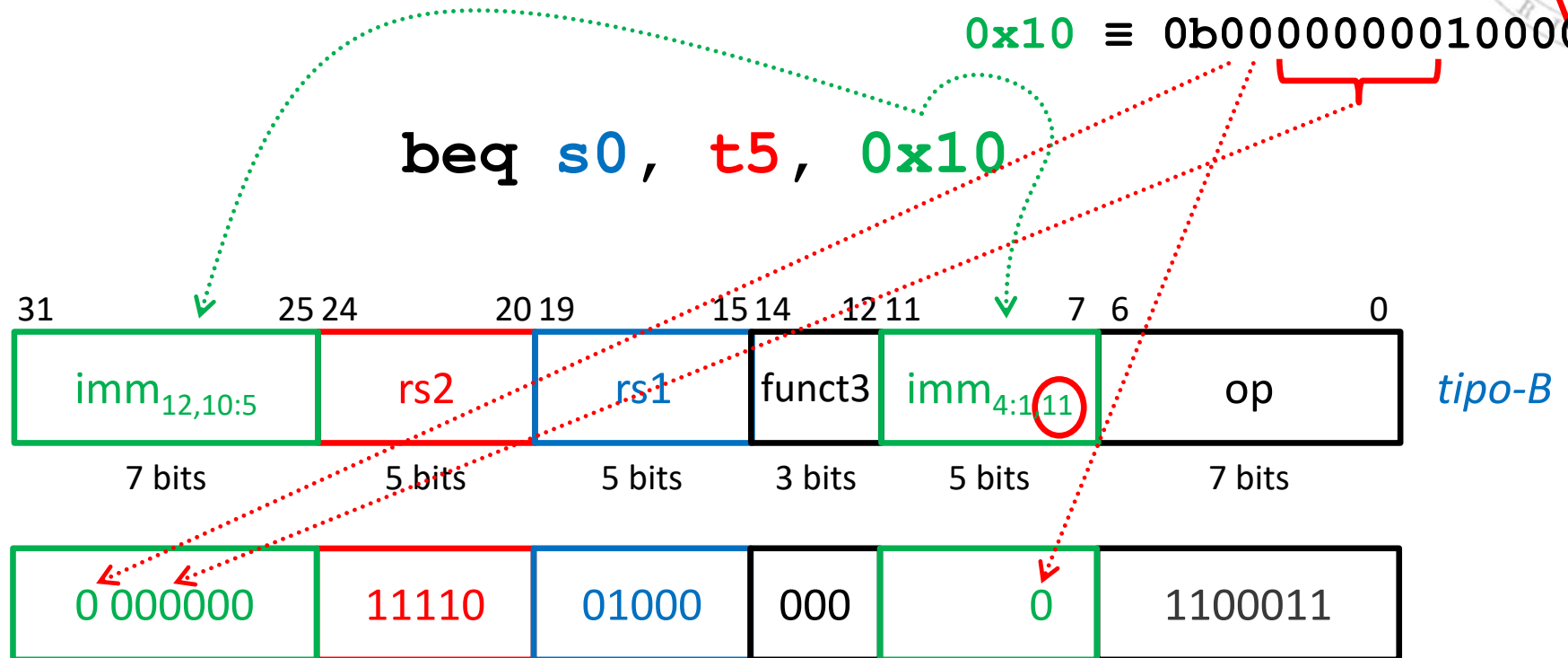


# De ensamblador a código máquina

## Ejemplo: instrucción tipo-B

`beq s0, t5, 0x10`

`0x10`  $\equiv$  `0b0000000010000`



*tipo-B*

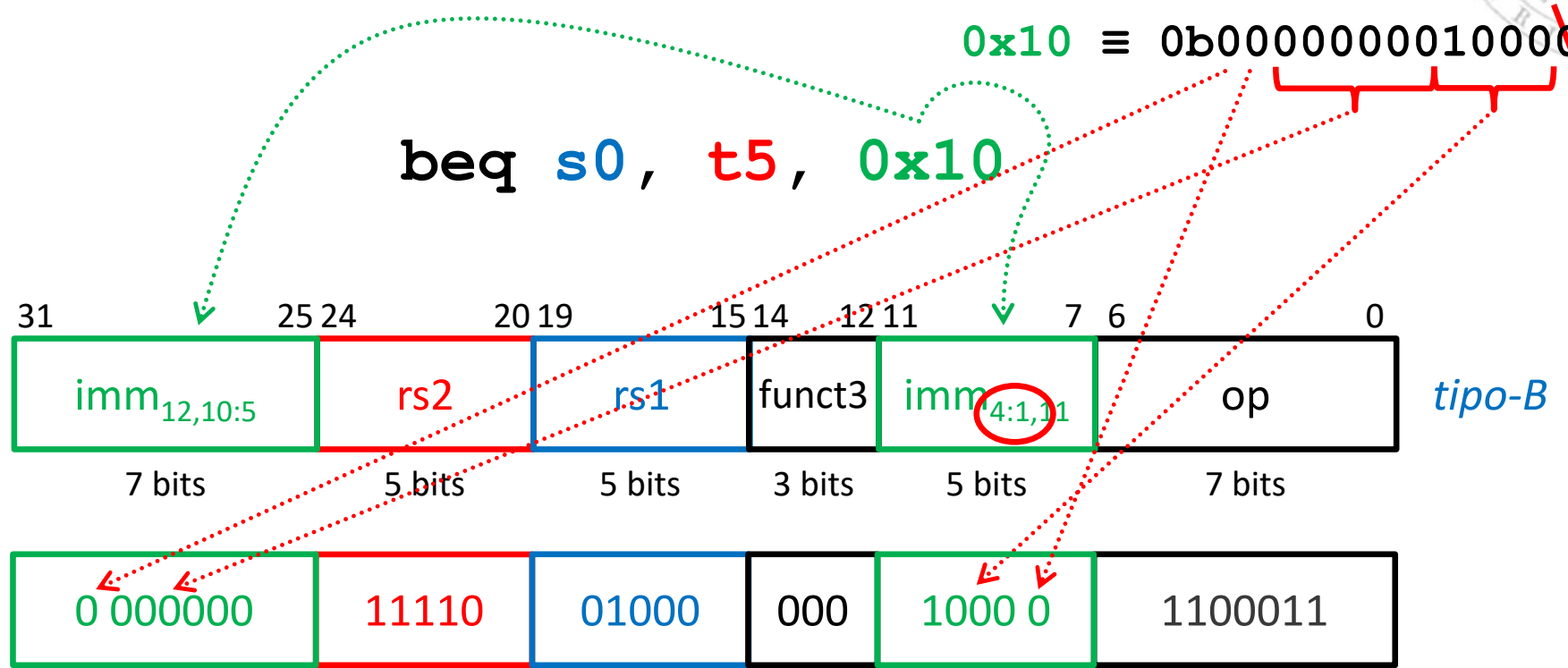


# De ensamblador a código máquina

## Ejemplo: instrucción tipo-B

`beq s0, t5, 0x10`

`0x10`  $\equiv$  `0b0000000010000`

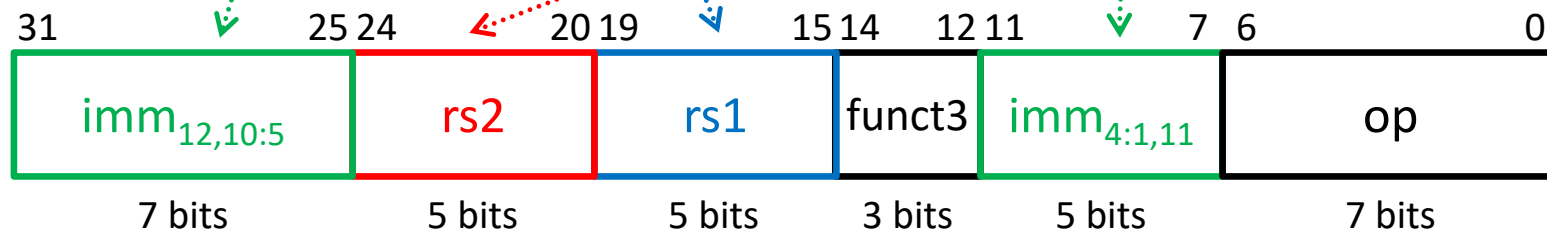




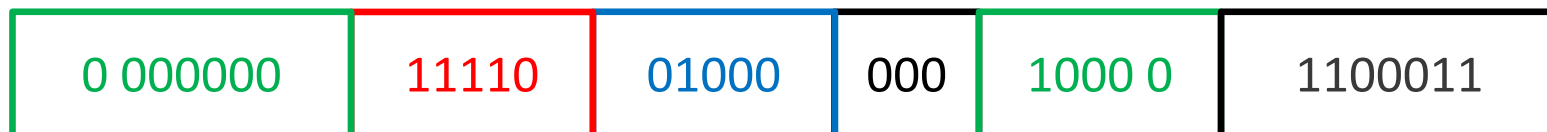
# De ensamblador a código máquina

## Ejemplo: instrucción tipo-B

beq s0, t5, 0x10



tipo-B



0x01e40863

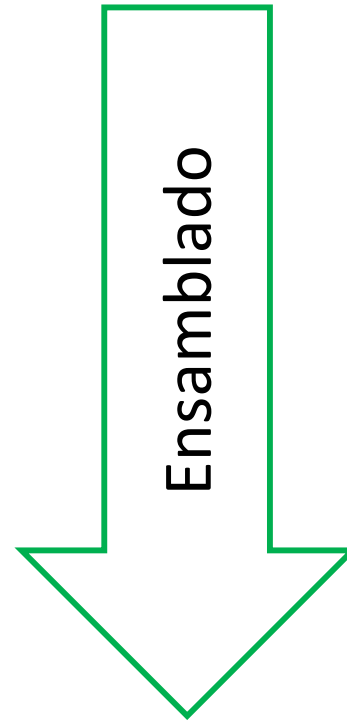


# De ensamblador a código máquina

Ejemplo: instrucción tipo-U



```
lui s5, 0x8cdef
```



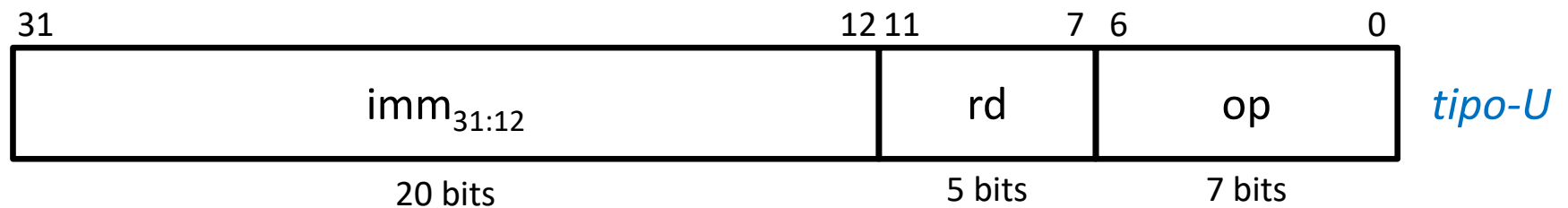
```
0x8cdefab7
```



# De ensamblador a código máquina

Ejemplo: instrucción tipo-U

```
lui s5, 0x8cdef
```

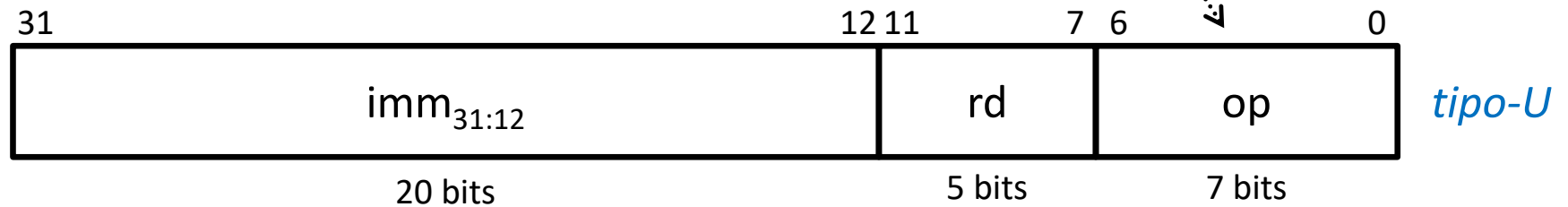




# De ensamblador a código máquina

Ejemplo: instrucción tipo-U

```
lui s5, 0x8cdef
```



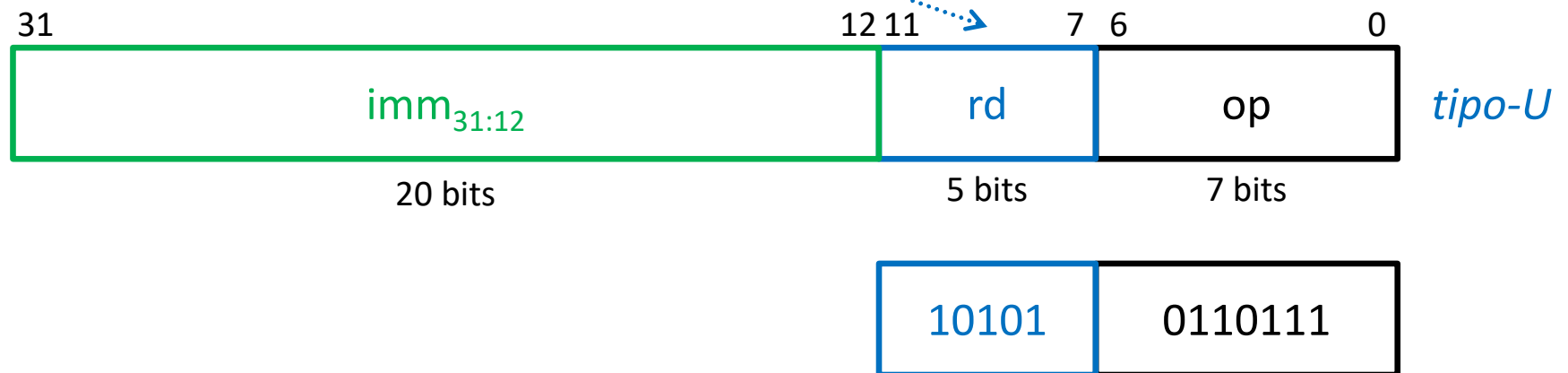
0110111



# De ensamblador a código máquina

Ejemplo: instrucción tipo-U

`lui s5, 0x8cdef`  $\equiv$  `lui x21, 0x8cdef`

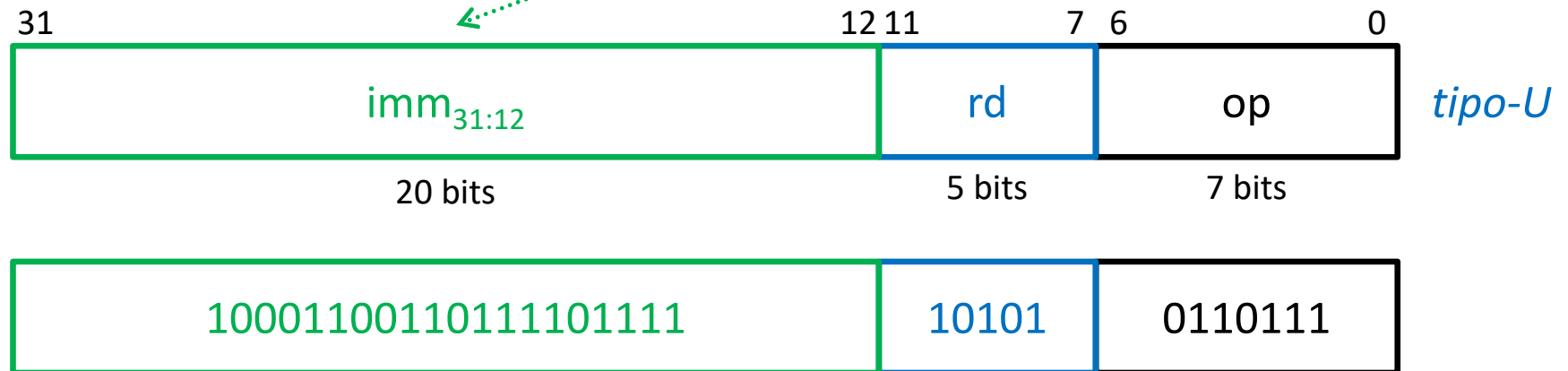




# De ensamblador a código máquina

Ejemplo: instrucción tipo-U

```
lui s5, 0x8cdef
```

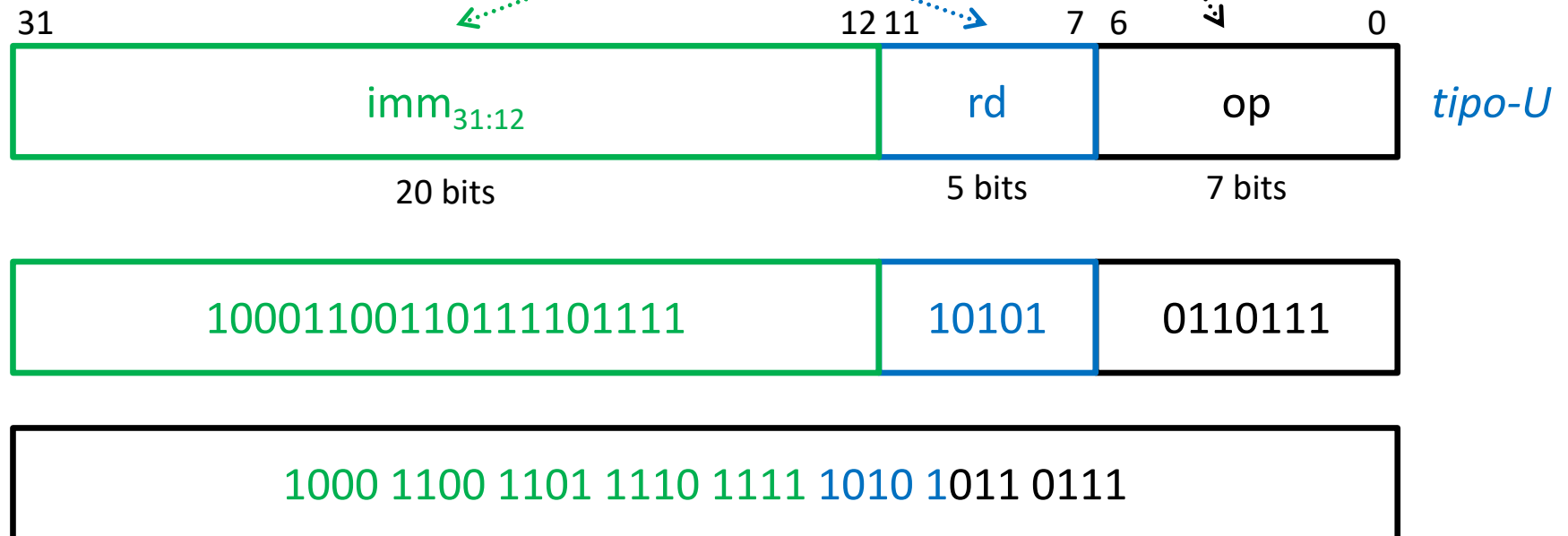




# De ensamblador a código máquina

Ejemplo: instrucción tipo-U

`lui s5, 0x8cdef`



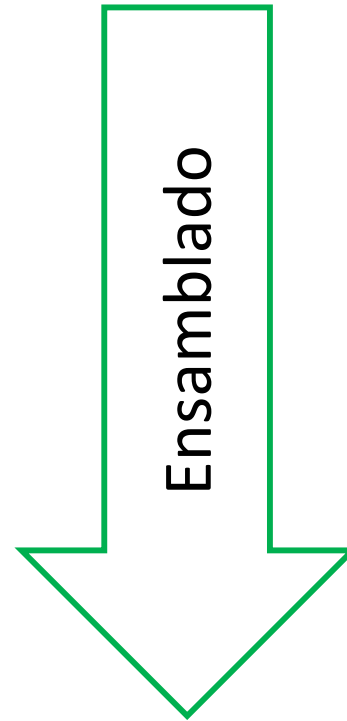
`0x8cdefab7`

# De ensamblador a código máquina

Ejemplo: instrucción tipo-J



```
jal ra, 0xa67f8
```



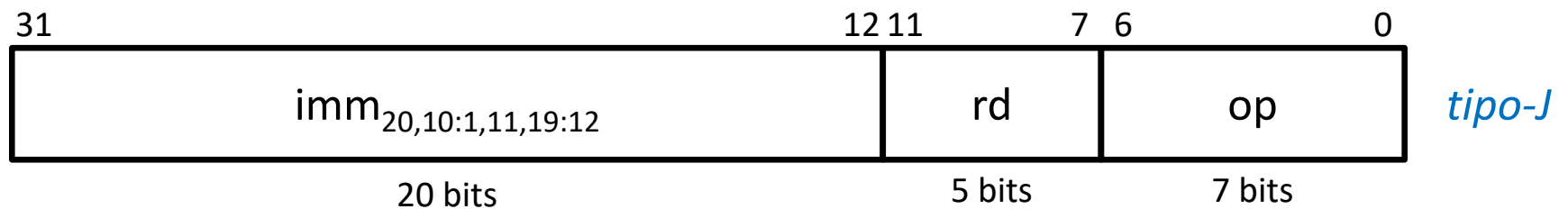
```
0x7f8a60ef
```

# De ensamblador a código máquina

Ejemplo: instrucción tipo-J



```
jal ra, 0xa67f8
```



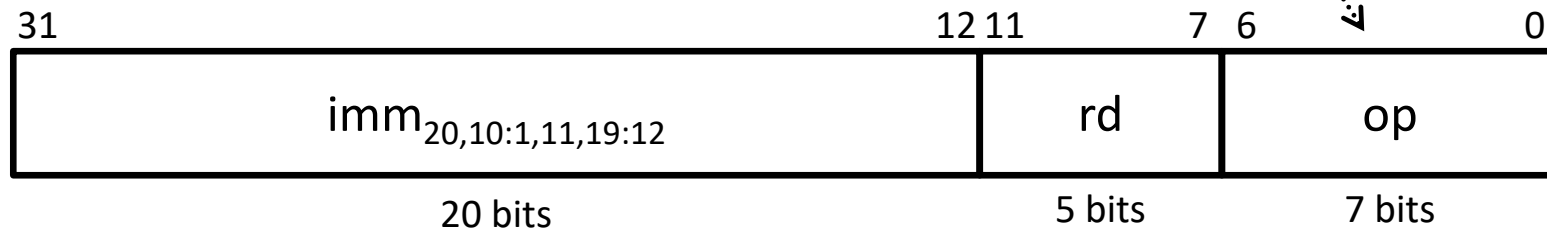




# De ensamblador a código máquina

Ejemplo: instrucción tipo-J

`jal ra, 0xa67f8`



*tipo-J*

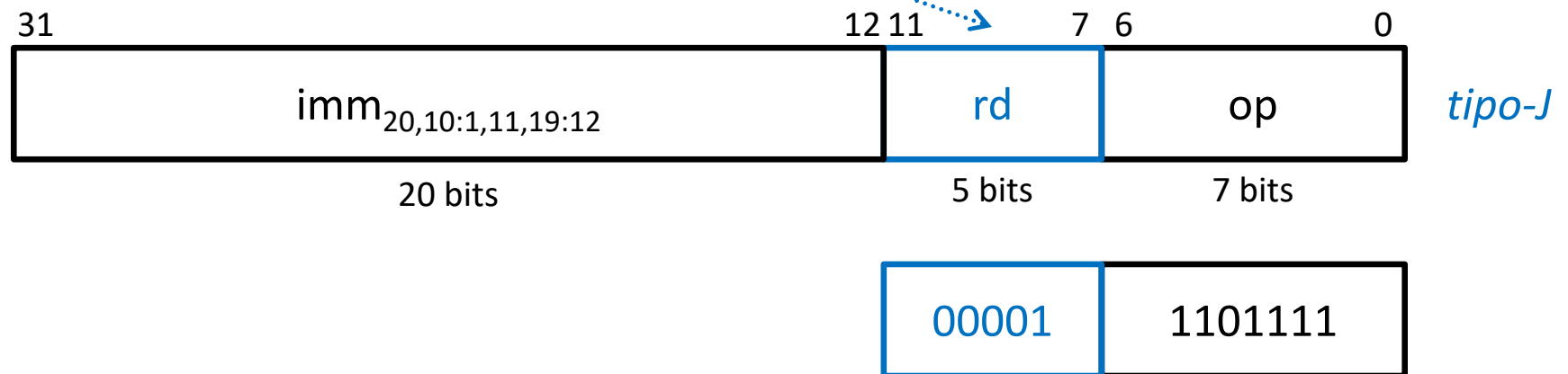
1101111



# De ensamblador a código máquina

Ejemplo: instrucción tipo-J

`jal ra, 0xa67f8`  $\equiv$  `jal x1, 0xa67f8`



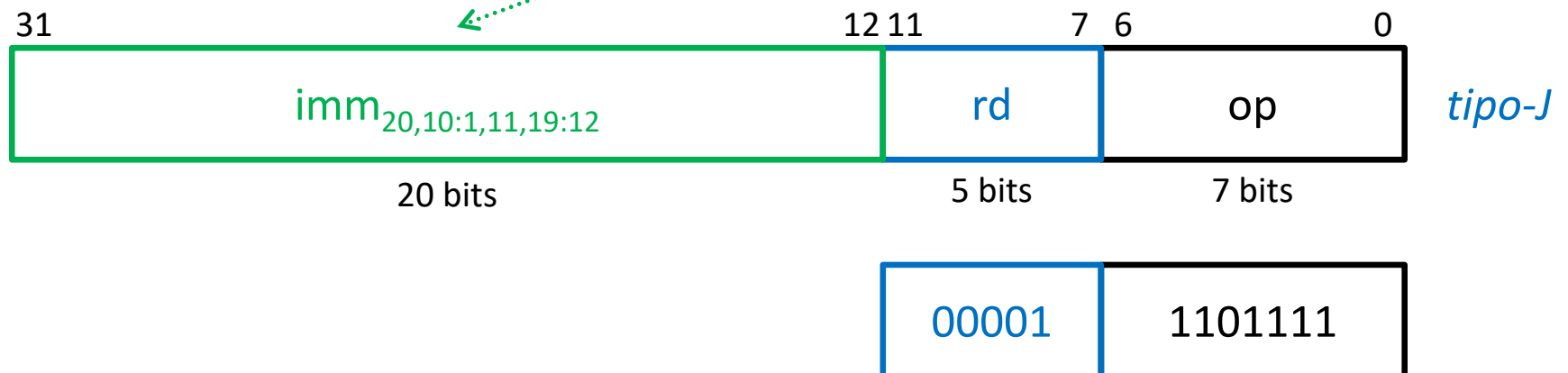


# De ensamblador a código máquina

## Ejemplo: instrucción tipo-J

0xa67f8  $\equiv$  0b010100110011111111000

jal ra, 0xa67f8



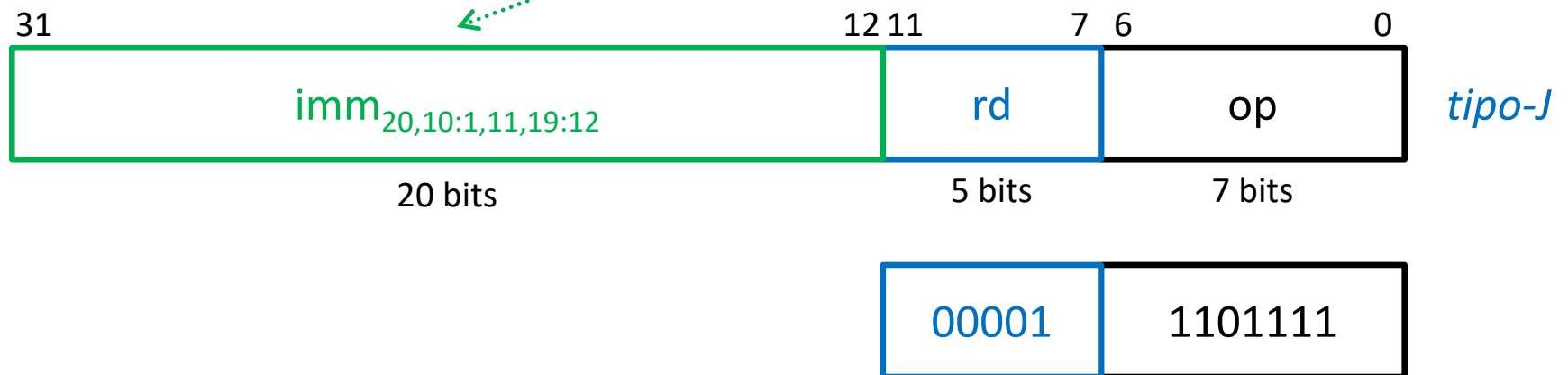


# De ensamblador a código máquina

## Ejemplo: instrucción tipo-J

0xa67f8 ≡ 0b0101001100111111111000

jal ra, 0xa67f8



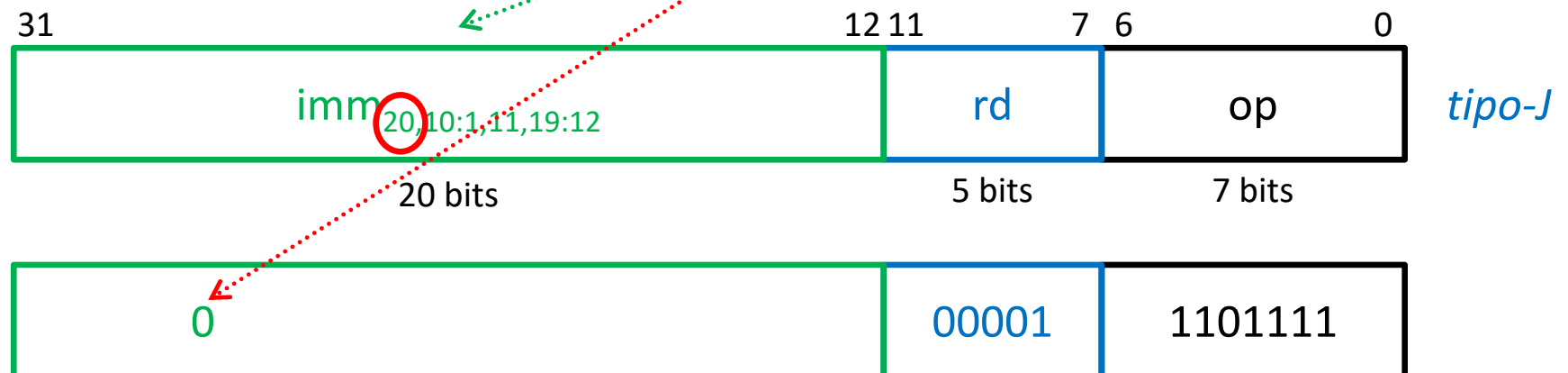


# De ensamblador a código máquina

## Ejemplo: instrucción tipo-J

0xa67f8  $\equiv$  0b010100110011111111000

jal ra, 0xa67f8



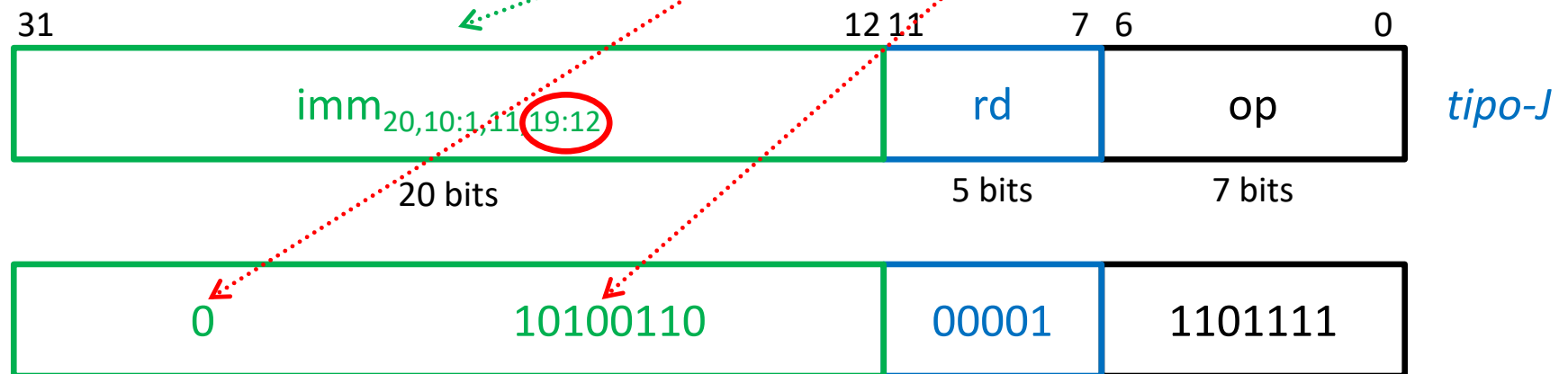


# De ensamblador a código máquina

## Ejemplo: instrucción tipo-J

0xa67f8  $\equiv$  0b010100110011111111000

jal ra, 0xa67f8



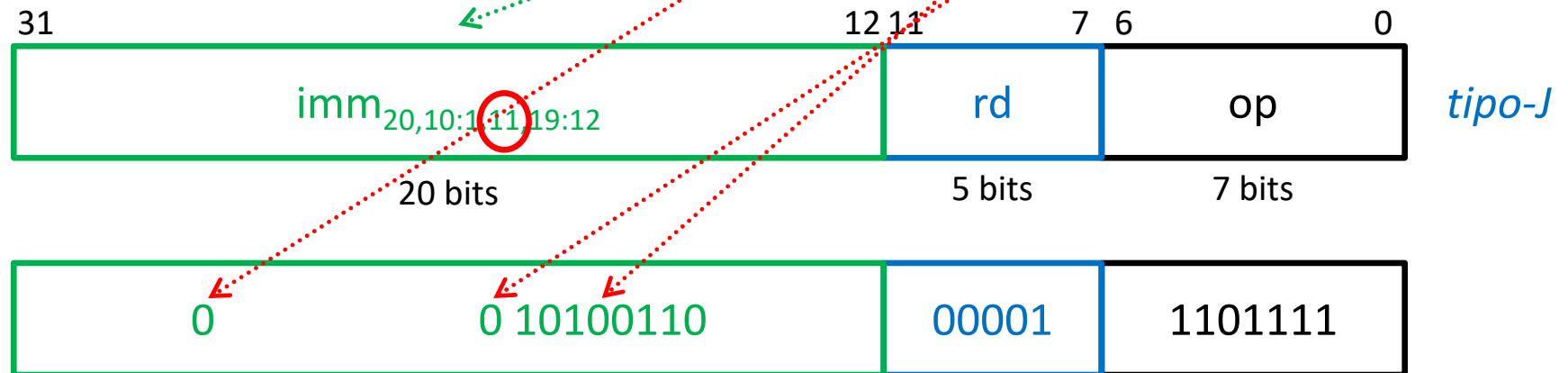


# De ensamblador a código máquina

## Ejemplo: instrucción tipo-J

0xa67f8 ≡ 0b010100110011111111000

jal ra, 0xa67f8



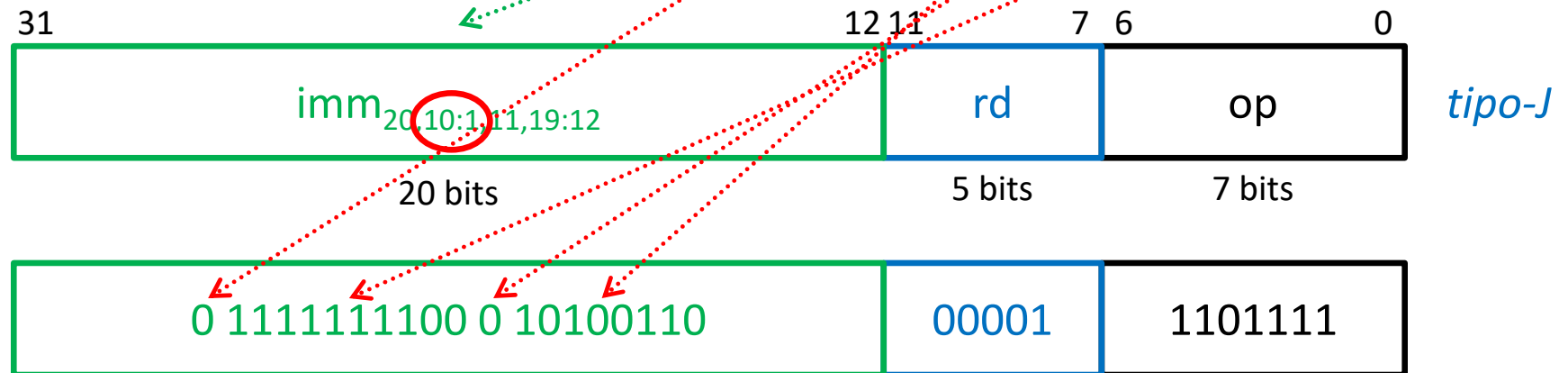


# De ensamblador a código máquina

Ejemplo: instrucción tipo-J

0xa67f8  $\equiv$  0b010100110011111111000

jal ra, 0xa67f8



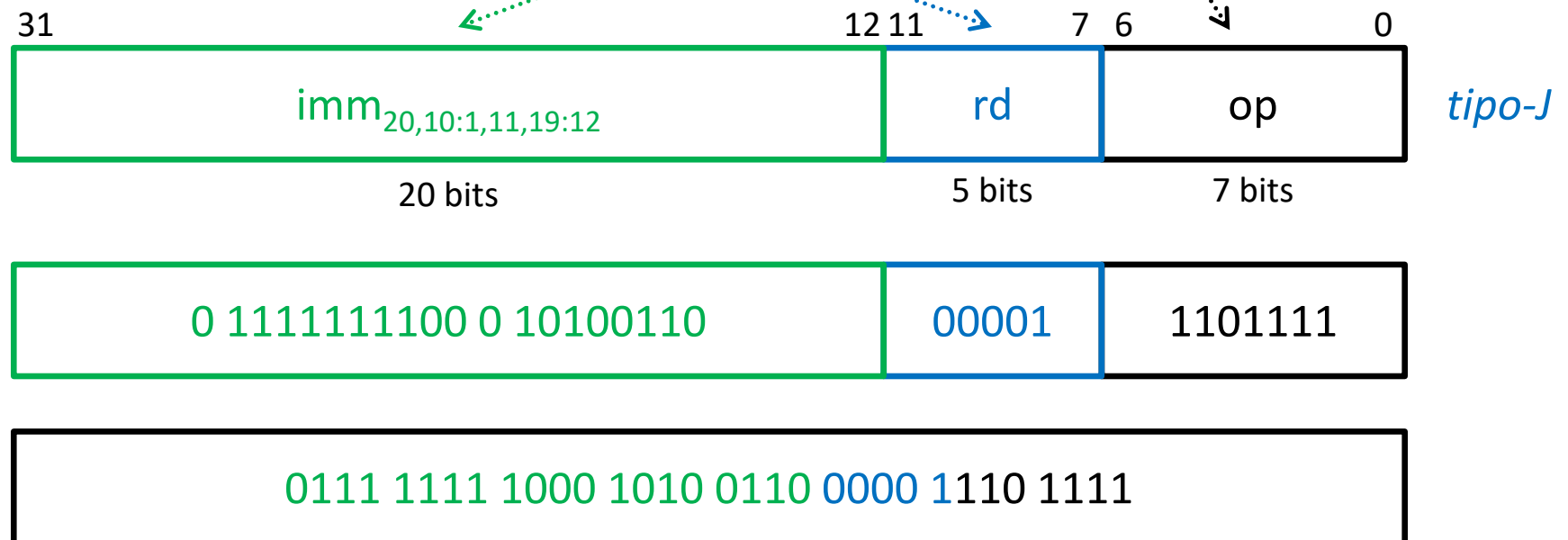




# De ensamblador a código máquina

Ejemplo: instrucción tipo-J

`jal ra, 0xa67f8`



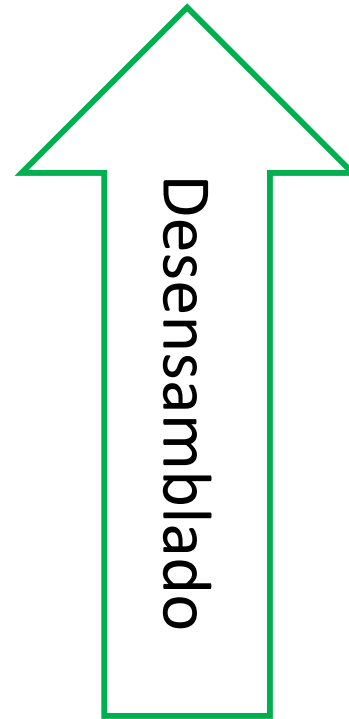
`0x7f8a60ef`

# De código máquina a ensamblador

## Ejemplo (i)



sub t2, t4, t6



0x41fe83b3

# De código máquina a ensamblador

## Ejemplo (i)



0100 0001 1111 1110 1000 0011 1011 0011

**0x41fe83b3**



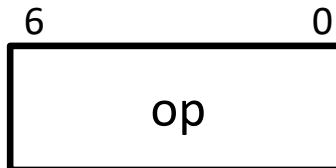
# De código máquina a ensamblador

## Ejemplo (i)

0110011

*tipo-R*

0100 0001 1111 1110 1000 0011 1011 0011



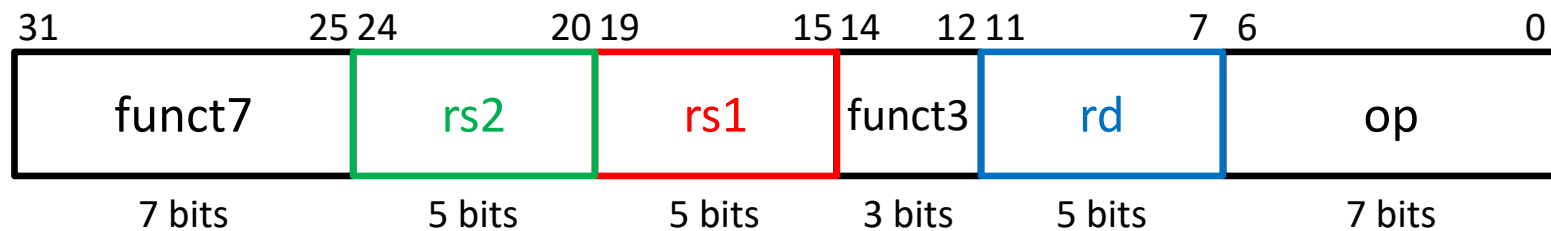
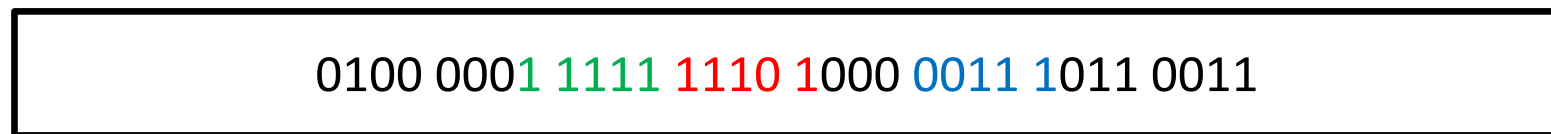
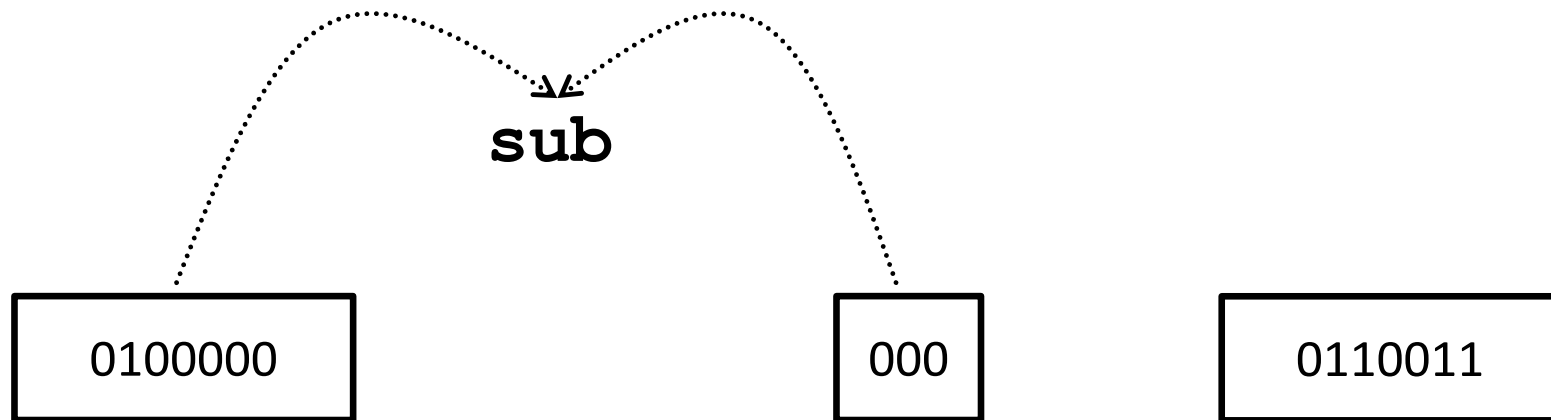
7 bits

0x41fe83b3



# De código máquina a ensamblador

## Ejemplo (i)



*tipo-R*

0x41fe83b3

# De código máquina a ensamblador

## Ejemplo (i)

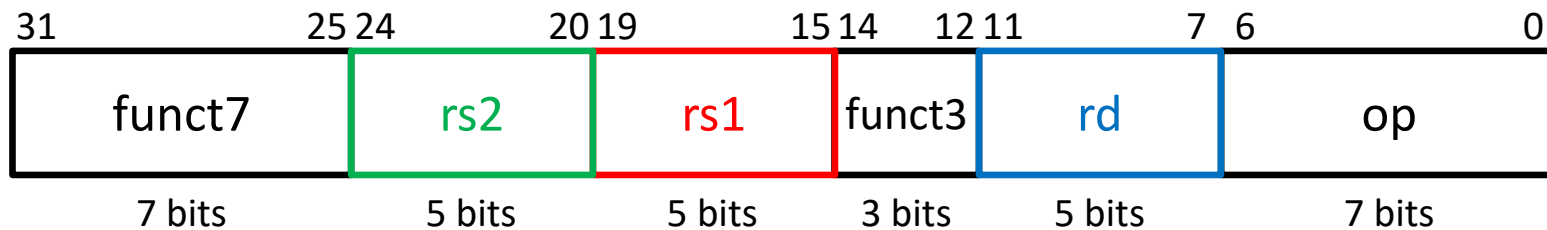


sub t2 ≡ sub x7

0100000

000 00111 0110011

0100 0001 1111 1110 1000 0011 1011 0011



tipo-R

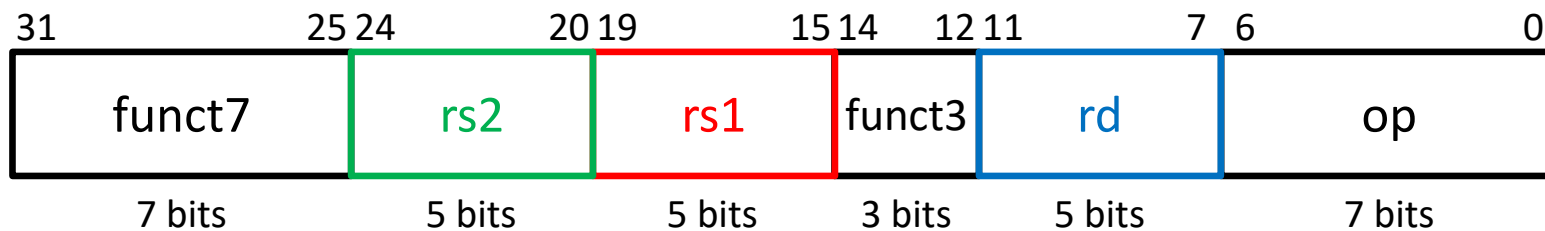
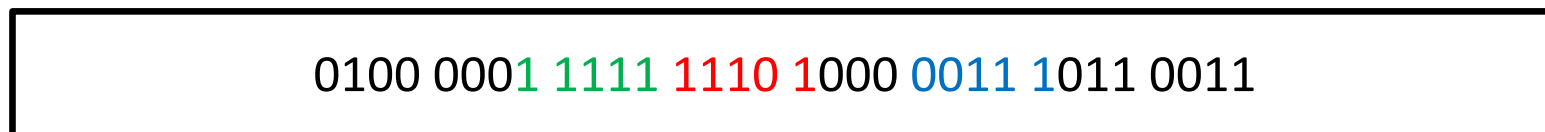
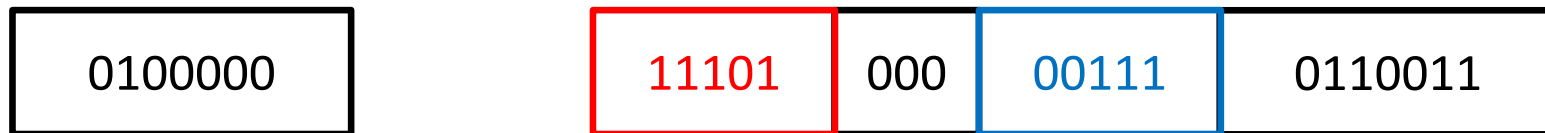
0x41fe83b3



# De código máquina a ensamblador

## Ejemplo (i)

sub t2, t4      ≡ sub x7, x29



*tipo-R*

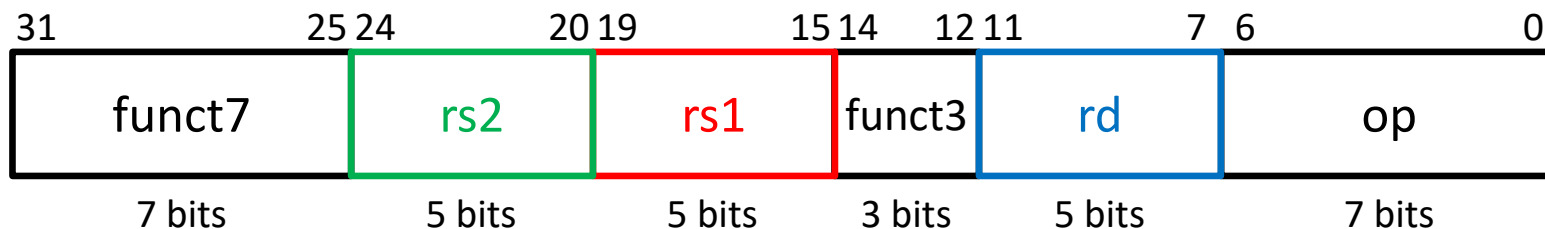
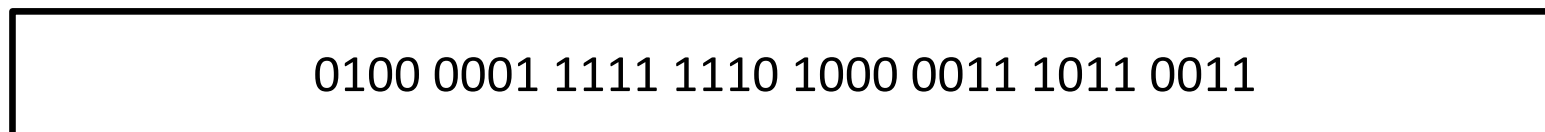
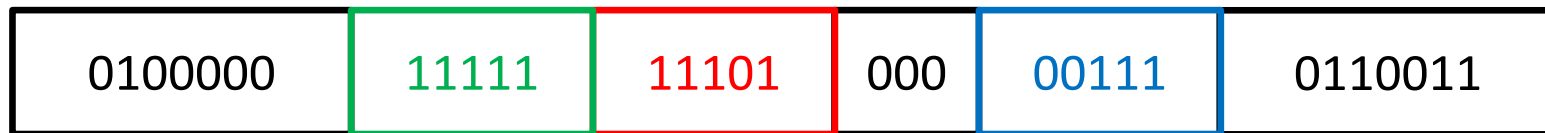
0x41fe83b3



# De código máquina a ensamblador

## Ejemplo (i)

sub t2, t4, t6  $\equiv$  sub x7, x29, x31



tipo-R

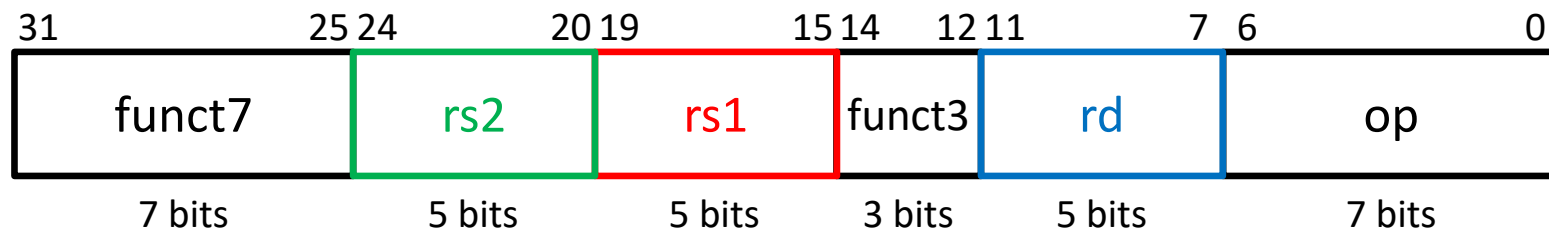
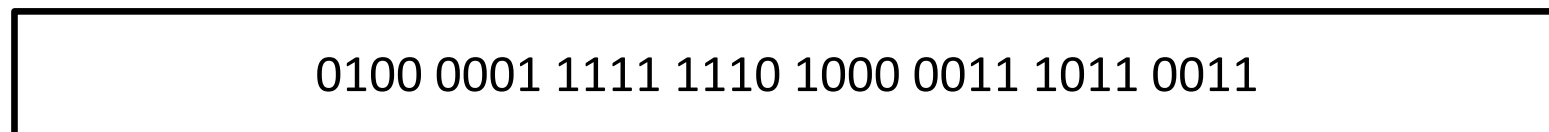
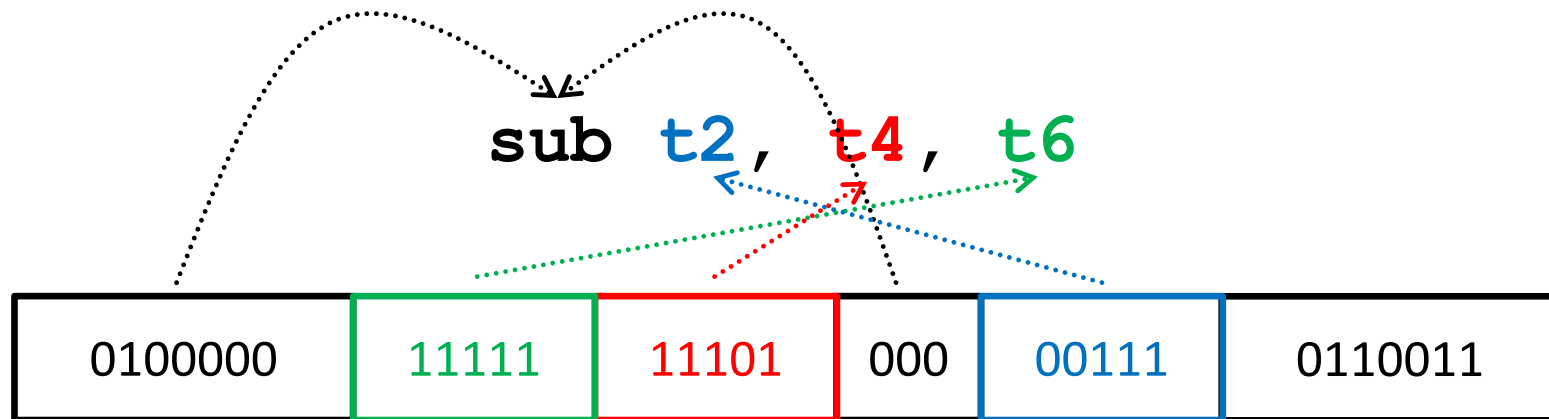
0x41fe83b3





# De código máquina a ensamblador

## Ejemplo (i)



tipo-R

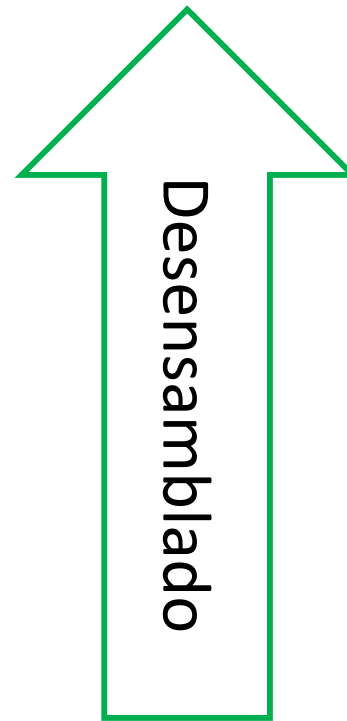
0x41fe83b3

# De código máquina a ensamblador

Ejemplo (ii)



`addi t0, s1, -38`



`0xfda48293`

# De código máquina a ensamblador

Ejemplo (ii)



1111 1101 1010 0100 1000 0010 1001 0011

0xfda48293



# De código máquina a ensamblador

## Ejemplo (ii)

0010011

*tipo-l*

1111 1101 1010 0100 1000 0010 1001 0011

6                      0  
op

7 bits

0xfda48293

# De código máquina a ensamblador

## Ejemplo (ii)

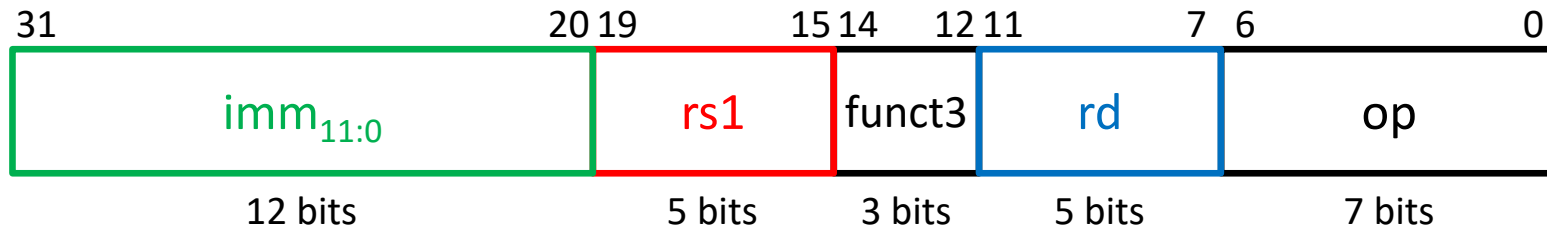


`addi`

000

0010011

1111 1101 1010 0100 1000 0010 1001 0011



*tipo-I*

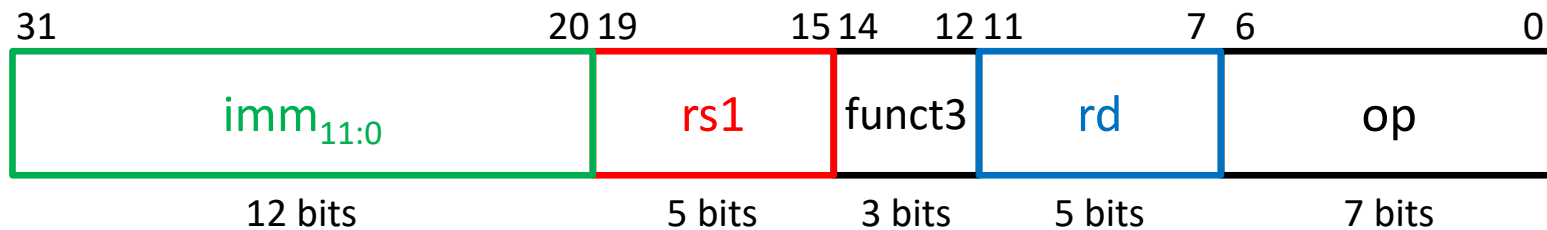
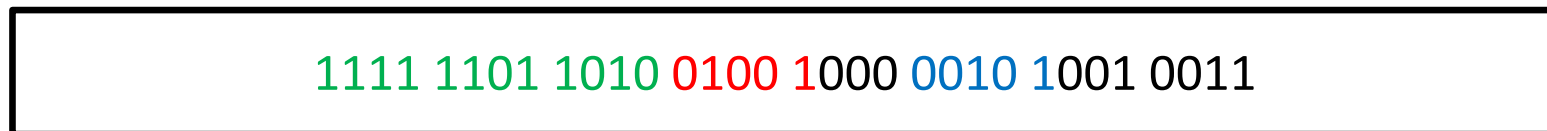
`0xfda48293`

# De código máquina a ensamblador

## Ejemplo (ii)



addi t0



*tipo-I*

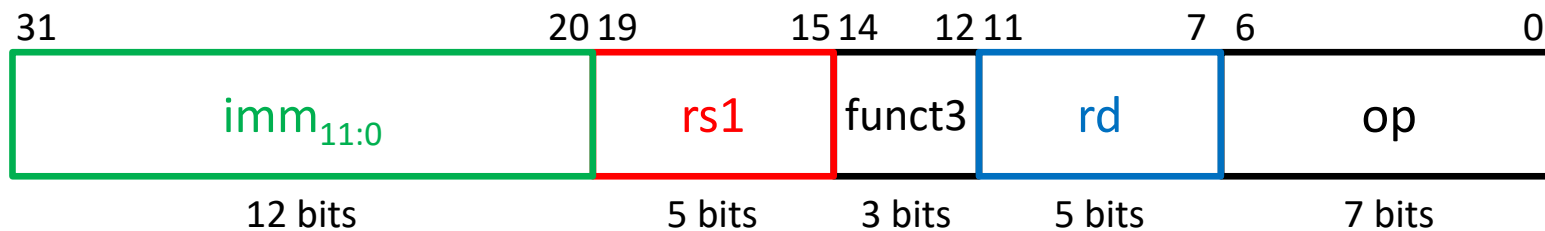
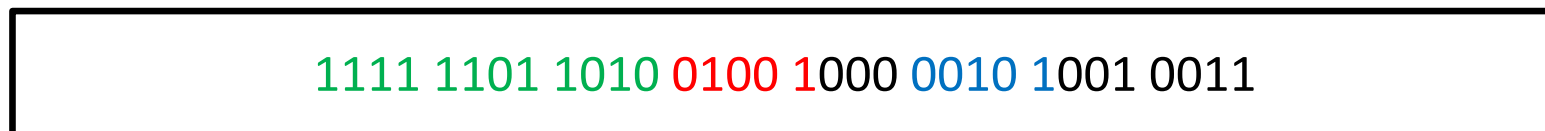
0xfda48293



# De código máquina a ensamblador

## Ejemplo (ii)

`addi t0`  $\equiv$  `addi x5`



*tipo-I*

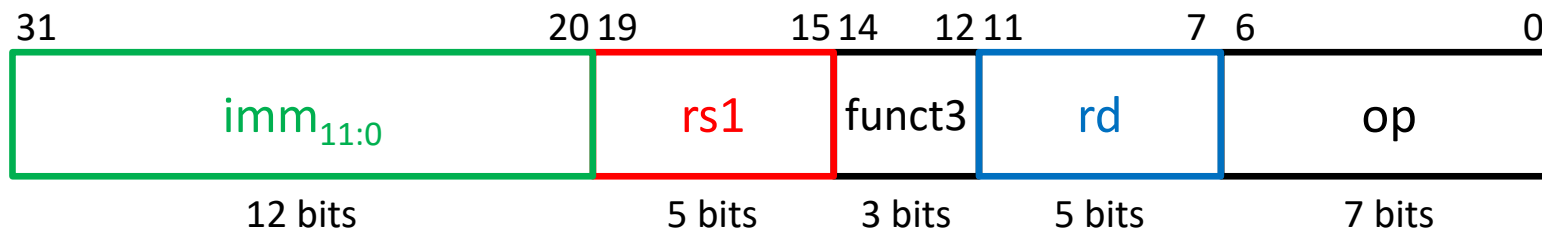
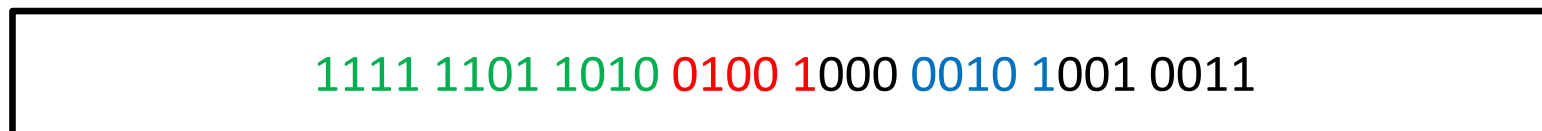
**0xfda48293**



# De código máquina a ensamblador

## Ejemplo (ii)

`addi t0, s1`  $\equiv$  `addi x5, x9`



*tipo-I*

**0xfda48293**

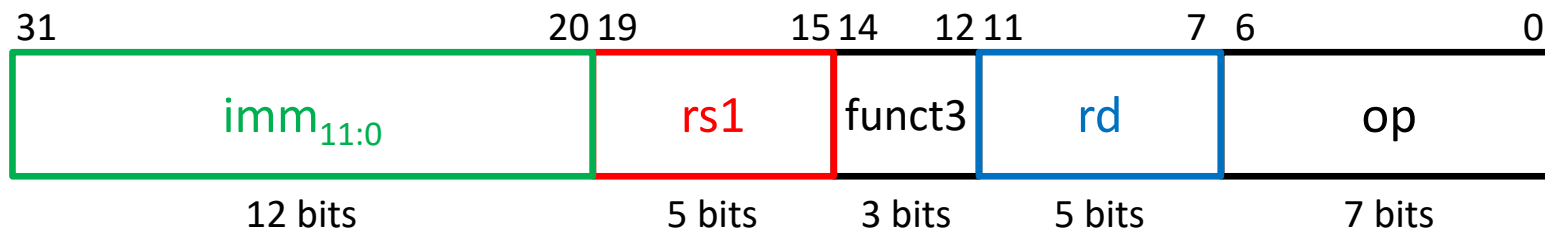
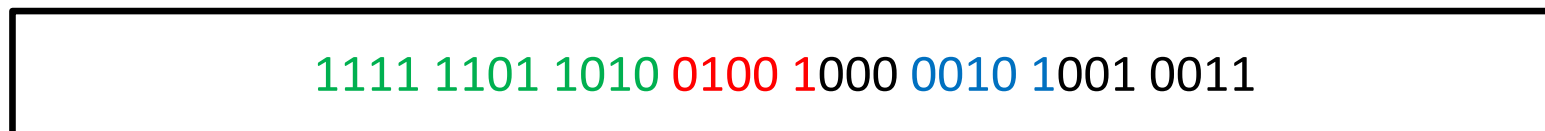
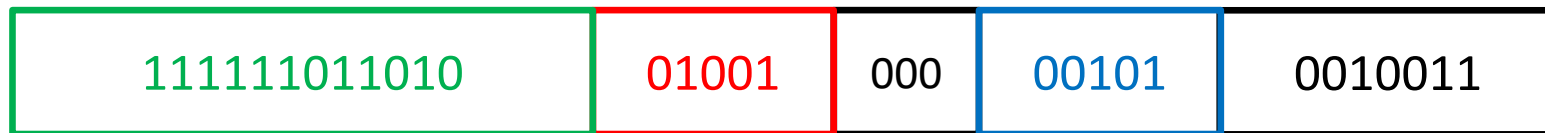




# De código máquina a ensamblador

## Ejemplo (ii)

`addi t0, s1, -38`



*tipo-I*

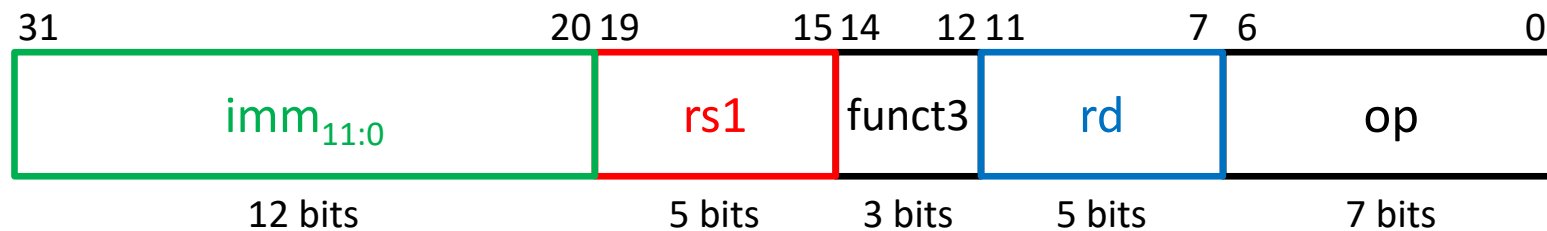
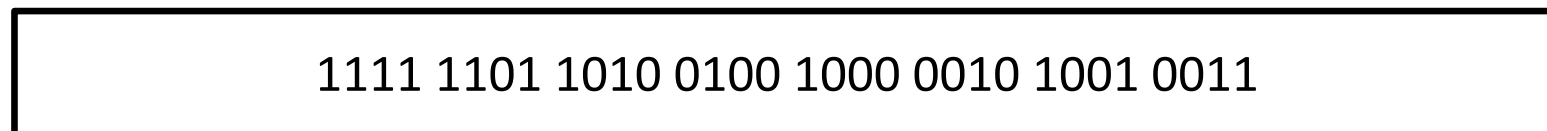
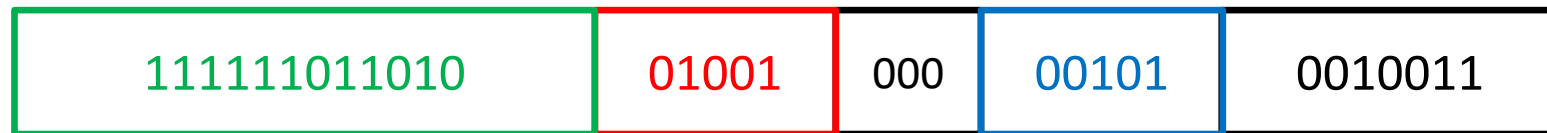
**0xfda48293**



# De código máquina a ensamblador

## Ejemplo (ii)

`addi t0, s1, -38`



*tipo-I*

**0xfda48293**

# Acerca de *Creative Commons*



## ■ Licencia CC (**Creative Commons**)

- Ofrece algunos derechos a terceras personas bajo ciertas condiciones. Este documento tiene establecidas las siguientes:



**Reconocimiento** (*Attribution*):

En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.



**No comercial** (*Non commercial*):

La explotación de la obra queda limitada a usos no comerciales.



**Compartir igual** (*Share alike*):

La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

**Más información:** <https://creativecommons.org/licenses/by-nc-sa/4.0/>