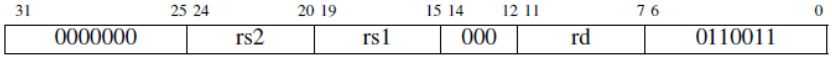
1. Transformeu la instrucció "add x9, x20, x21" a la seva representació decimal (això significa donar el valor decimal de cada camp de la pregunta, el valor decimal del camp fucnt7, del camp rs2, rs1, etc). Separeu els valors amb espais, per exemple “20 21 22 23 24 25”

add rd, rs1, rs2 `x[rd] = x[rs1] + x[rs2]`

Add. Tipo R, RV32I y RV64I.

Suma el registro x[*rs2*] al registro x[*rs1*] y escribe el resultado en x[*rd*]. Overflow aritmético ignorado.

Formas comprimidas: **c.add** rd, rs2; **c.mv** rd, rs2



	rs2	rs1	funct3	rd	opcode
0	21	20	0	9	51
	x21	x20		x9	x33

2. Si x10 té la base de la matriu A i x21 correspon a h, l'assignació:

$$A[30] = h + A[30] + 1;$$

es compila a

```
ld x9, 240(x10) // Temporary reg x9 gets A[30]
add x9, x21, x9 // Temporary reg x9 gets h+A[30]
addi x9, x9, 1 // Temporary reg x9 gets h+A[30]+1
sd x9, 240(x10) // Stores h+A[30]+1 back into A[30]
```

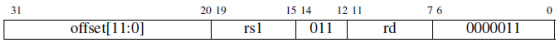
Transformeu cada instrucció a la seva representació decimal (com en l’anterior exercici)

ld rd, offset(rs1) `x[rd] = M[x[rs1] + sext(offset)][63:0]`

Load Doubleword. Tipo I, solo RV64I.

Carga ocho bytes de memoria en la dirección x[*rs1*] + *sign-extend*(*offset*) y los escribe en x[*rd*].

Formas comprimidas: **c.ldsp** rd, offset; **c.ld** rd, offset(rs1)



ld x9, 240(x10):

offset[11:0]	rs1	funct3	rd	opcode
240	10	3	9	3

add x9, x21, x9:

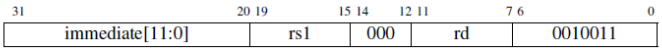
	rs2	rs1	funct3	rd	opcode
0	9	21	0	9	51

addi rd, rs1, immediate `x[rd] = x[rs1] + sext(immediate)`

Add Immediate. Tipo I, RV32I y RV64I.

Suma el *inmediato* sign-extended al registro x[*rs1*] y escribe el resultado en x[*rd*]. Overflow aritmético ignorado.

Formas comprimidas: **c.li** rd, imm; **c.addi** rd, imm; **c.addi16sp** imm; **c.addi4spn** rd, imm

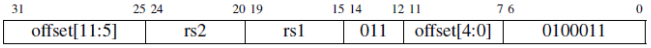


sd rs2, offset(rs1) `M[x[rs1] + sext(offset)] = x[rs2][63:0]`

Store Doubleword. Tipo S, solo RV64I.

Almacena los ocho bytes del registro x[*rs2*] a memoria en la dirección x[*rs1*] + *sign-extend*(*offset*).

Formas comprimidas: **c.sdsp** rs2, offset; **c.sd** rs2, offset(rs1)



addi x9, x9, 1:

offset[11:0]	rs1	funct3	rd	opcode
1	9	0	9	19

sd x9, 240(x10): 240 → 1111 0000

offset[11:5]	rs2	rs1	funct3	offset[4:0]	opcode
7	9	10	3	16	35
...0111				1 0000	

3. A quina instrucció representa els següents valors decimals? :

"32 9 10 000 11 51"

?	rs2	rs1	funct3	?	opcode
32	9	10	000	11	51
10 0000			000		011 0011

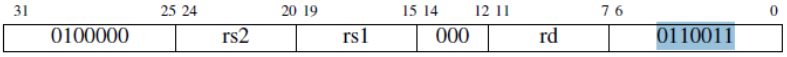
Busquem la instrucció que coincideixi amb el Opcode y funct que tenim.

sub rd, rs1, rs2 `x[rd] = x[rs1] - x[rs2]`

Subtract. Tipo R, RV32I y RV64I.

Subtracts register x[*rs2*] from register x[*rs1*] y escribe el resultado en x[*rd*]. Overflow aritmético ignorado.

Forma comprimida: **c.sub** rd, rs2



	rs2	rs1	funct3	rd	opcode
32	9	10	000	11	51
010 0000			000		011 0011

La instrucció resultant és **sub x11, x10, x9**

4. Contesteu les preguntes teòriques del qüestionari

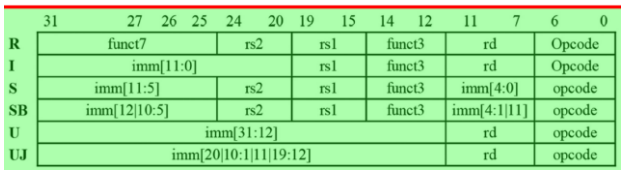
Totes les instruccions de la ISA RISC-V, per exemple la R64I, tenen la mateixa longitud en bits.

Verdadero

Quants tipus d'instruccions tenim a la ISA del RISC-V, per exemple la R64I ?

6

The 6 Instruction Formats



Quantes instruccions de tipus R podríem arribar a codificar amb la ISA RISC-V ?

1024

- How many R-format instructions can we encode?
  - with `opcode` fixed at 0b0110011, just funct varies:
    $(2^7) \times (2^3) = (2^{10}) = 1024$