🧩 Example: `cmplwi cr6, r5, 0xff`

Assembler expands this to

```
cmpli 6, 0, 5, 0xff
```

Now look at the **box layout** for `cmpli` (the one labeled `10` at the left):

Bits	Field	Value
0–5	Opcode	`10` (binary `001010`) → **0x2A** base opcode (different assemblers sometimes show as `0x2B` after shifting)
6–8	BF	`6`
9	unused	
10	L	`0` (word)
11–15	RA	`5`
16–31	UI	`0x00FF`

When assembled, these fields are packed into one 32-bit word like this:

```
001010 110 0 00101 0000000011111111
```

Binary → Hex ≈ `0x2B05_00FF` (simplified view).

✅ Bottom line:

Those boxes literally show **which bits go where** inside the 32-bit instruction.
The left number (`10` or `11`) becomes the **first hex byte (0x2A or 0x2B)** when the CPU sees it.