Solutions

Each answer shows the steps, then the **result**.

2.1

- machine code is **binary** that the CPU runs directly
- assembly language is a **readable** version using mnemonics, with one instruction per machine instruction

2.2

(a) the value **20** (immediate)

(b) the **contents of address 20** (direct)

2.3

- a jump can refer to a label defined **later** (a forward reference)
- pass 1 records every label's address first, so pass 2 can fill it in

2.4

- indexed: base 100 + IX 4 → address **104**

3.1

(a) **pass 1:** scan the source and build the symbol table — record the address of each label; do not generate code yet

- **pass 2:** translate each instruction to machine code, looking up label addresses in the symbol table

(b) to resolve **forward references** (a jump to a label defined later)

3.2

- immediate = the operand **is the value** in the instruction
- direct = the operand is at the **address** given
- indirect = the address given holds **another address**, which holds the data

3.3

- start: $a = 10$, $b = 4$

```
Instruction   ACC    c
LDD a          10    -
ADD b          14    -
STO c          14    14
END            14    14
```

- ACC after `ADD` = **14**
- c after `STO` = **14**

3.4

- e.g. to control hardware directly / for speed / for a small, tight routine