## Solutions

Marking: **M1** a correct method, **A1** the correct answer, **B1** an independent mark.

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

**2.1**

R3 = 8

after `sub`, R4 = 3

after `addi`, R4 = **13**

**2.2**

`0x1000` + 20 = `0x1014`

`lw R1, 20(R2)`

**2.3**

R1 goes 10, 9, …, 0: the body runs **10** times

2 instructions a pass

$1 + 10 \times 2 =$ **21**

**2.4**

```
addi R1, R2, 7
```

```
sub R1, R1, R2
```

**3.1**

(a)

```
      add  R1, R0, R0
loop: beq  R3, R0, done
      lw   R4, 0(R2)
      add  R1, R1, R4
      addi R2, R2, 4
      addi R3, R3, -1
      j    loop
done:
```

sum set to 0; exit test on the count; load and add; address +4 and count −1, then jump back

(b) 6 per pass; $1 + 100 \times 6 + 1 =$ **602**

(c) 602 ns

**3.2**

(a) R1 = 7, R3 = 9

(b) $R4 = 9 - 7 =$ **2**

(c) address `0x108` becomes 2



(d) 4 instructions × 4 bytes = 16; PC = `0x410`

**3.3**

(a) 12 per pass; 25 passes: $1 + 25 \times 12 + 1 =$ **302**; about half of 602

(b) 602/302 ≈ **2.0**

(c) the last 2 elements do not fill a pass of 4; add a short clean-up loop for the remainder

(d) usually not: compilers unroll at `-O2`/`-O3`, and hand unrolling makes code longer and harder to read and check



exercises.lol exercises.com.cn