Trace assembly code AQA style 2

Challenge 1

Sergei has written a program using assembly language. Trace the program and enter **the final value of register `R1`** once the program has fully executed.

It may be useful to write down the values of `R0` and `R1` on paper whilst tracing the program.

```
MOV R0, #0
MOV R1, #48

loopstart:
    CMP R0, #2
    BEQ loopend
    ADD R0, R0, #1
    LSR R1, R1, #1
    B loopstart
loopend:
    HALT
```

Write assembly code AQA style 1

Challenge 1

The following program is written in high-level pseudocode:

```
x = 10
WHILE x > 0
   x = x-1
ENDWHILE
```

Put the assembly code language statements (below) into the correct order to recreate the high-level program.

In the assembly code, `R0` is used as the register for `x`.

Available items

```
end:
```

```
looptop:
```

```
      B looptop
```

```
      HALT
```

```
      BEQ end
```

```
      SUB R0, R0,#1
```

```
      CMP R0, #0
```

```
      MOV R0, #10
```

Quiz:
STEM SMART Computer Science Week 40 (AQA)

Create AQA Assembly language program from pseudocode 1

Challenge 1

Amy wants to write an assembly language program and has planned the program logic for 1 section of the program using the following high-level pseudocode:

```
IF y == 10 THEN
  x = 9
ELSE
  y = y + 1
ENDIF
```

She plans to use the register **R1** to represent the variable y and the register **R2** to represent the variable x.

Study the assembly code version below and drag and drop the correct instructions to complete the code block so that it implements the same logic.

```
    [          ]
    BNE else
    [          ]
    B endif
else:
    [          ]
endif:
    HALT
```

Items:

MOV R1, #9 | CMP R1, #10 | MOV R2, #9 | ADD R2, R2, #1 | ADD R1, R1, #1

Quiz:
STEM SMART Computer Science Week 40 (AQA)

Create AQA Assembly language program from pseudocode 2

Practice 2

Charles wants to write an assembly language program and has planned it using the following high-level pseudocode:

```
total = 0
WHILE total < 100
    total = total + 10
ENDWHILE
```

He plans to use the register **R1** to represent the variable `total`. The label `start` will be used to represent the start of the loop, and the label `end` will mark the end of the loop.

Assemble the instructions into the correct order by dragging them into the box on the right to implement the same logic as the high-level pseudocode.

Available items

```
    ADD R1,R1,#10
```

```
    HALT
```

```
    MOV R1, #0
```

```
end:
```

```
    CMP R1, #100
```

```
    B start
```

```
    BEQ end
```

```
start:
```

Quiz:
STEM SMART Computer Science Week 40 (AQA)

Complete AQA assembly code program

Practice 2

David has written the following incomplete assembly language program to perform integer division by 10.

The value to be divided is stored in register R1. The program should reduce the value in R1 in steps of 10 and stop when the value stored in R1 is less than 10. Each time that the value in R1 is decreased by 10, the value in register R2 should be increased by 1.

For example, if R1 contained the value 52, the sequence of numbers stored in R1 should be:

- 52
- 42
- 32
- 22
- 12
- 2

And the final value in R2 should be 5.

Complete the program by writing in the missing instructions. The program should only use the registers R1 and R2.

```
      MOV R2, #0
      MOV R1, #52
loopstart: CMP R1, #10
      [                    ]
      ADD [                ]
      SUB [                ]
      [                    ]
end:  HALT
```

Quiz:
STEM SMART Computer Science Week 40 (AQA)

Reorder the assembly language statements (AQA)

Practice 2

Rita is attempting to write an assembly language program to perform the following steps:

- Load value from memory location 55 into **R0**
- Load value from memory location 56 into **R1**
- Subtract contents of **R1** from **R0** and store result in **R2**
- Multiply the contents of **R0** by 2 and store result in **R3**
- Store the final result in memory location 57
- End program

Reorder the lines of code below to produce her intended program.

Available items

```
SUB R2, R0, R1
```

```
HALT
```

```
LDR R1, 56
```

```
STR R3, 57
```

```
LDR R0, 55
```

```
LSL R3, R0, 1
```

Quiz:
STEM SMART Computer Science Week 40 (AQA)

Assembly language (AQA) to add up numbers in a loop

Practice 2

Jake wants to create an assembly language program that will calculate the sum of numbers from 1 up to and including a particular number. This number is stored in memory location **40** and the final total is to be stored in memory location **41**.

For example, if the number stored in **40** is 5, the total that will be stored in **41** when the program halts is 15 because:

$1 + 2 + 3 + 4 + 5 = 15$

His outline for the program is as follows:

1. Load the value from memory location **40** into register **R0**
2. Initialise the register **R1** with the value 1. This will be the loop counter.
3. Initialise the register **R2** with the value 1. This will be the running total.
4. If the value of the loop counter (R1) is equal to the value stored in **R0** then exit the loop and store the total from R2 into memory location **41** and halt the program.
5. Else, add 1 to the loop counter (R1) and add the value of the counter (R1) to the running total (R2).
6. Return to step 4.

Sequence the instructions into the correct order by dragging them into the box on the right.

Available items

```
        STR R2, 41
```

```
        LDR R0, 40
```

```
        BEQ endloop
```

```
        ADD R2, R2, R1
```

```
loop:
```

```
        B loop
```

```
        MOV R1,#1
```

```
        HALT
```

```
endloop:
```

```
        ADD R1, R1, #1
```

```
        CMP R1, R0
```

```
        MOV R2,#1
```

Complete assembly code to detect odd numbers

Practice 2

Agnes wants to write an assembly language program that will detect if the number stored in **R1** is an odd or even number.

She knows that an odd number, when represented in binary, will always have its least significant bit set to 1.

She also knows that if she performs the logical bitwise operation **AND** on this number with the value 1, it will return a 1 if it is odd and a 0 if it is even.

If it is an odd number, she wants to store 0 in **R3** and 1 in **R4**.

If it is an even number, she wants to store 1 in **R3** and 0 in **R4**.

She is trying to test if the value of 21 is odd or even and her incomplete program is below:

```
      MOV R1, #21
      <<< missing statement A >>>
      CMP R2, #1
      <<< missing statement B >>>
even: MOV R3, #1
      MOV R4, #0
      HALT
odd:  MOV R3, #0
      MOV R4, #1
      HALT
```

Part A

What instruction should go in the place of missing statement A?

Part B

What instruction should go in the place of missing statement B?