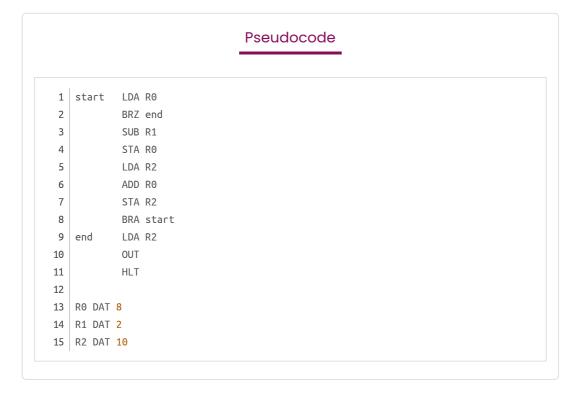
### Trace assembly code OCR style 2



Karishma has written a program in assembly language and has given it to her classmates to try out. Trace the program and work out what the **final value of R2** will be at the end of the program.

It may be useful to write down the values of R0, R1 and R2 on paper whilst tracing the program.



O 30

**22** 

38

**20** 





## Reorder the assembly language statements (LMC)



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

- Take input from user and store in the variable N1
- Take input from user and store in the variable N2
- Subtract the contents of N1 from N2 and store result in RES
- Multiply the contents of **RES** by 2
- Output the final result
- End program

Assuming that the last three lines of the program are:

```
// Memory Locations
N1 DAT 0 // Number 1
N2 DAT 0 // Number 2
RES DAT 0 // Result
```

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

#### Available items

HLT	
STA N1	
SUB N1	
STA RES	
LDA N2	
INP	
INP	
OUT	
ADD RES	
STA N2	

### Write assembly code OCR style 1



Here is a program written in a high-level language:

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

An assembly language program has been written, which achieves the same outcome as the high-level program. Some statements have been indented to improve readability. The last two lines of the assembly language program are:

x DAT 10 y DAT 1

Without adding any extra indentation, put the assembly language statements below in the correct order to produce a program that is equivalent to the high-level program above.

#### Available items



Quiz:

STEM SMART Computer Science Week 41 (LMC)





### Complete LMC assembly code program

Practice 2

David has written an assembly language program to perform integer division by 10.

The value to be divided is stored in the variable named DENOMINATOR. The program should reduce the value in DENOMINATOR in steps of 10 and stop when the value stored in DENOMINATOR is negative. Each time the value in DENOMINATOR is decreased by 10, a variable named QUOTIENT will be incremented by 1.

For example, if DENOMINATOR contained the value 52, the sequence of numbers stored in DENOMINATOR would be:

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

And the final value in QUOTIENT will be 5.

Complete the program by writing in the missing instructions.

START	LDA DENOMINATOR
	SUB
	BRA END
CONTINUE	STA DENOMINATOR
	LDA QUOTIENT
	ADD
	STA QUOTIENT
END	OUT QUOTIENT
	HLT
DENOMINAT	OR DAT 52
QUOTIENT	DAT 0
TEN	DAT 10
ONE	DAT 1

Ouiz:

STEM SMART Computer Science Week 41 (LMC)

## Complete assembly code to detect positive and negative numbers (LMC)



Agnes wants to write an assembly language program that will detect if a number entered is positive, negative, or zero.

- ullet If the number is positive, the program should store 1 in variable RESULT
- ullet If the number is negative, the program should store -1 in variable RESULT
- If the number is zero, the program should store 0 in variable RESULT

Her incomplete program is below:

```
LDA INPUT
       BRZ IS_ZERO
        <<< missing statement A >>>
       BRA IS_NEG
IS POS LDA ONE
       STA RESULT
       BRA END
IS_NEG <<< missing statement B >>>
       STA RESULT
       BRA END
IS_ZERO LDA ZERO
       STA RESULT
END
       HI T
INPUT
        DAT 0
RESULT DAT 0
ONE
       DAT 1
MINUS_ONE DAT -1
7FR0
        DAT 0
```

#### 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?	
Quiz: STEM SMART Computer Science Week 41 (LMC)	
(LIVIC).	





## Create LMC assembly language program from pseudocode 1



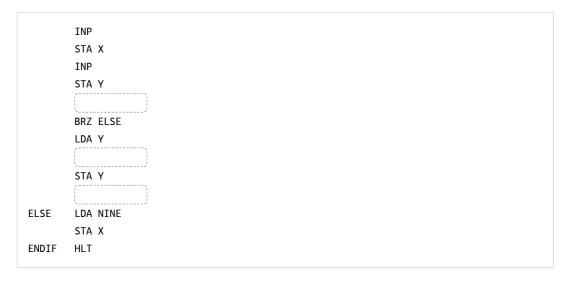
Amy wants to write an assembly language program and has planned the program logic for one section of the program using the following high-level pseudocode:

```
1 | IF y == 10 THEN
2 | x = 9
3 | ELSE
4 | y = y + 1
5 | ENDIF
```

She sets up the variables at the end of her program like this:

```
X DAT 0  // Variable X
Y DAT 0  // Variable Y
TEN DAT 10  // Constant 10
NINE DAT 9  // Constant 9
ONE DAT 1  // Constant 1
```

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.



Items:

BRA ENDIF (ADD TEN) (SUB TEN) (SUB ONE) (BRA ELSE) (ADD ONE)

## Create LMC Assembly language program from pseudocode 2



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

```
1  total = 0
2  WHILE total < 100
3  total = total + 10
4  ENDWHILE</pre>
```

The label **START** will be used to represent the start of the loop and the label **END** will mark the end of the loop. Assuming he sets up the following variables at the end of his program:

```
TOTAL DAT 0 // Variable to store the total
HUNDRED DAT 100 // Constant value 100
TEN DAT 10 // Constant value 10
ZERO DAT 0 // Constant value 0
```

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**

LOOP	LDA TOTAL	
	STA TOTAL	
	LDA ZERO	
	ADD TEN	
	LDA TOTAL	
	SUB HUNDRED	
END	HLT	
	BRP END	
	STA TOTAL	
	BRA LOOP	

# Assembly language (LMC) to add up numbers in a loop



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 entered by the user and the final total is to be output to the user.

For example, if the number entered by the user is 5, the total that will be stored in FINAL and output when the program halts is 15 because:

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

His outline for the program is as follows:

- 1. Ask the user to input the number and store it in the variable USER
- 2. Add 1 to the COUNT variable and store it.
- 3. Add the COUNT variable to the TOTAL variable to keep a running total.
- 4. If the COUNT variable has reached the value stored in the USER variable then exit the loop, output the total, and end the program
- 5. Otherwise, return to step 2

Assuming that the last four lines of the program are:

```
// Memory Locations
USER DAT 0
COUNT DAT 0
TOTAL DAT 0
ONE DAT 1
```

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

#### Available items

STA USER	
BRZ END_LOOP	
LOOP LDA COUNT	
SUB USER	
LDA TOTAL	
BRA LOOP	
END_LOOP LDA TOTAL	
OUT	
STA TOTAL	

ADD ONE	
INP	
STA COUNT	
ADD COUNT	
LDA COUNT	
HLT	



