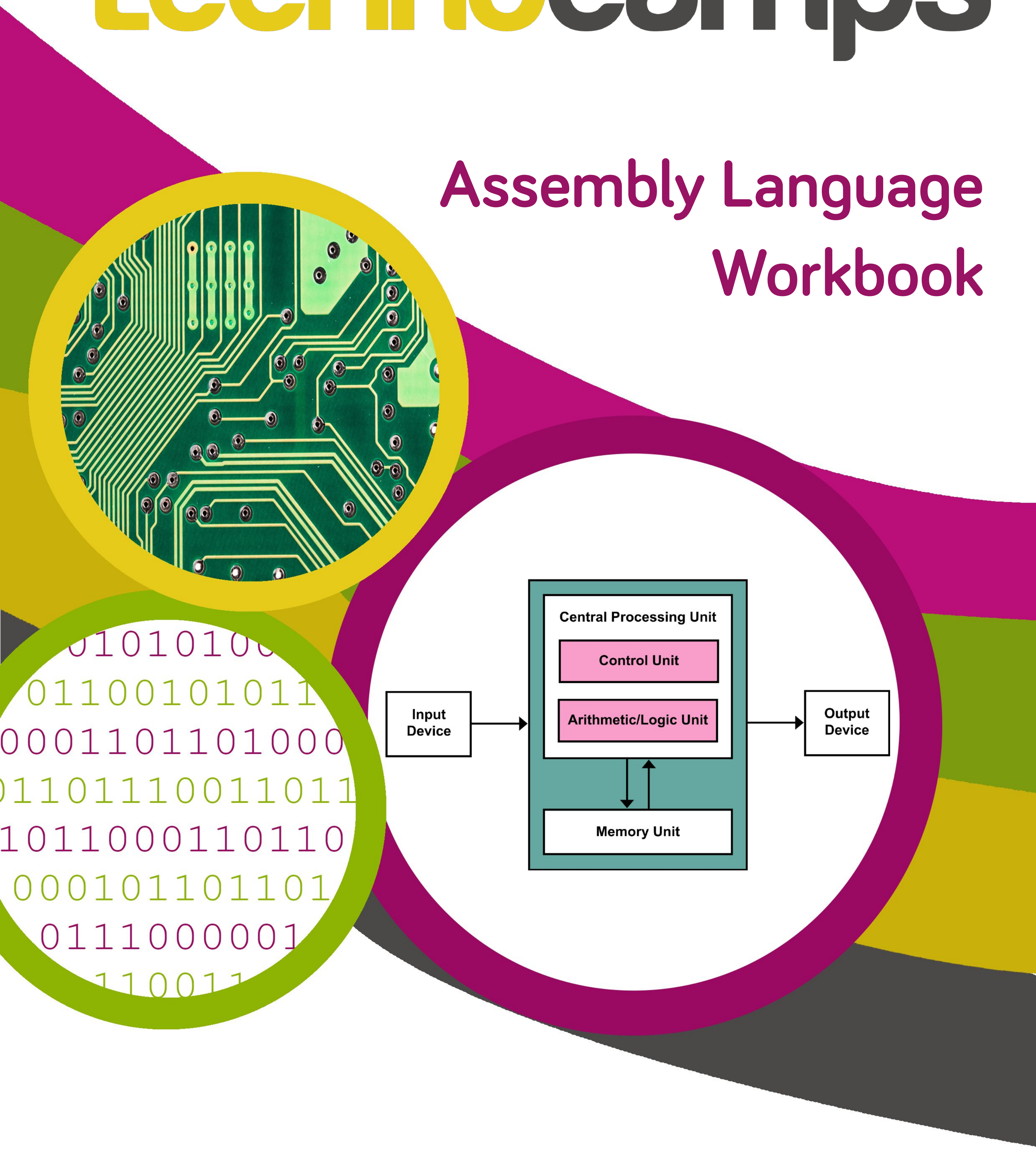
technocamps

Assembly Language Workbook



UNDEB EWROPEAIDD
EUROPEAN UNION
Llywodraeth Cymru
Welsh Government
Cronfa Gymdeithasol Ewrop
European Social Fund

Prifysgol Abertawe
Swansea University
CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD
PRIFYSGOL BANGOR UNIVERSITY
Cardiff Metropolitan University
Prifysgol Metropolitan Caerdydd
i.t.wales
PRIFYSGOL ABERYSTWYTH UNIVERSITY
Prifysgol Glyndŵr Wrecsam
Wrexham Glyndŵr University
University of South Wales
Prifysgol De Cymru

Overview

In this workshop we will be looking at Assembly Language, from studying computer architecture to learning how to write assembly programs of varying complexity.

Learning Outcomes

1. Improved knowledge of different Computer Architectures.
2. Greater experience of designing, writing and using Algorithms in Assembly Language environments.
3. Improved knowledge of Number Sequences.

Attendee Prerequisites

1. No previous knowledge of Assembly Language programming required.

Surface Pro 5 vs. MacBook Pro 2017

Which laptop do you prefer? ______________________________

Why do you prefer that laptop?

____

____

Memory

Describe the difference between volatile and non-volatile storage as well as giving an example of each.

____

____

Volatile: ______________________

Non-Volatile: ______________________

Put these memory amounts in order from smallest to largest: Bytes, Bits, TerraByte(TB), MegaByte(MB), GigaByte(GB), PetaByte(PB), Nibble, KiloByte(KB)

Compare Technology

Choose one of the options on the board, find the items online and compare their tech specs. Then give detailed reasons why one is better than the other.

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

Von Neumann Architecture

Can you name each part of the diagram?

A. ____________________

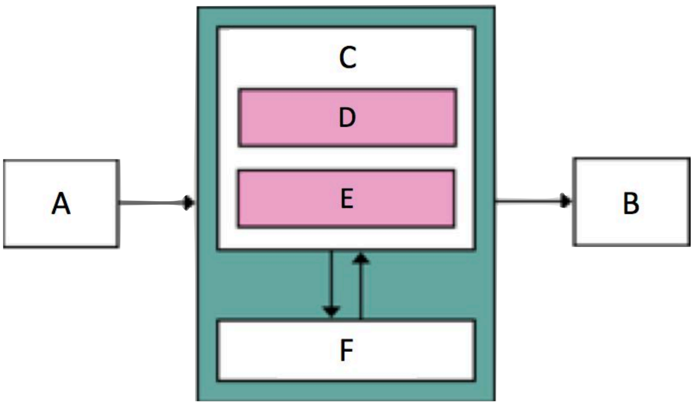
B. ____________________

C. ____________________

D. ____________________

E. ____________________

F. ____________________



Flexibility

Von Neumann architecture is ______ flexible than Harvard architecture because ____

____

____

____

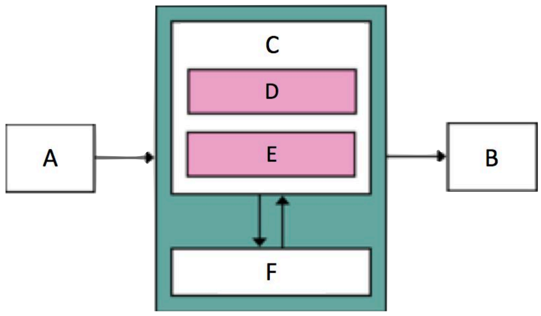
____

____

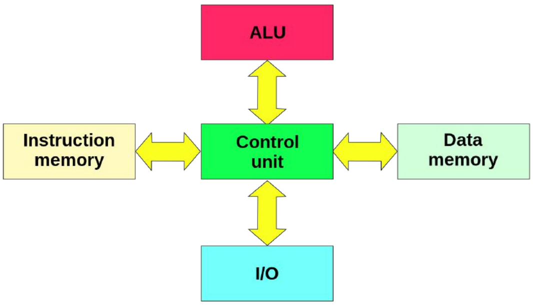
____

Assembly Language

Von Neumann vs Harvard



This stores both instructions and data within the same memory addresses and uses the same bus for both.



This has separate memory addresses for instructions and data meaning it can run a program and access data simultaneously.

What is an Assembly Language?

Assembly/Low-level languages are:

____

____

____

When are Assembly/Low-level languages useful?

____

____

____

Fill in the Blanks

- ______________________ - This shows which type of instruction is being used and which memory address it is being used on.

- ______________________ - This is like the active memory of the simulator. The majority of our instructions will modify the contents of the Accumulator.

- ______________________ - This is where a value is copied to from the Accumulator to display to the user.

- ______________________ - This shows the current memory location that the processor is running.

- ______________________ - This is where user inputs are stored initially before being copied to the Accumulator.

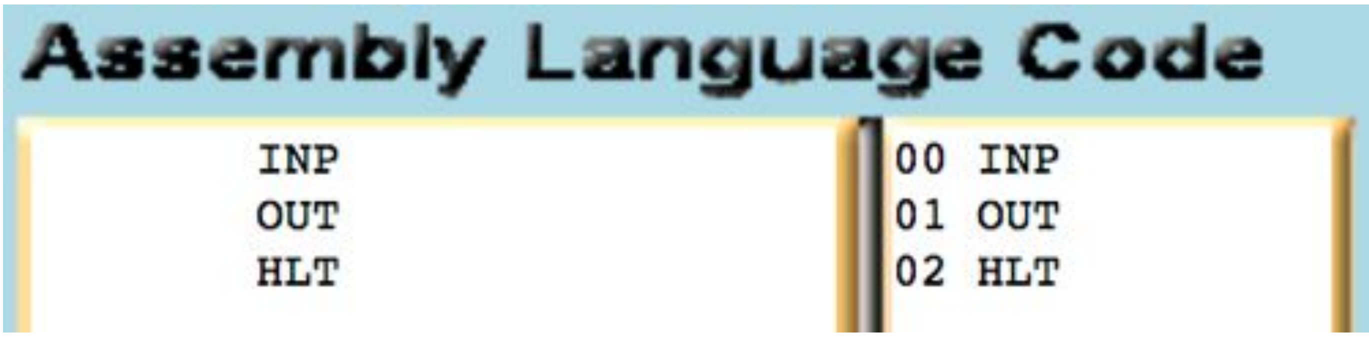
- ______________________ - These are the RAM addresses which are used to store instructions and data.

Visualising Assembly Programs

Assembly Language Functions

Function	LMC Mnemonic	LMC Code	What does it do?
Input	INP	901	Copies the value inputted by the user into the Accumulator.
Output	OUT	902	Copies the value in the Accumulator into the Output box.
Halt	HLT	000	This instruction does not affect any of the memory locations and stops the program.

Visualising a Program Running



Assembly Language Code

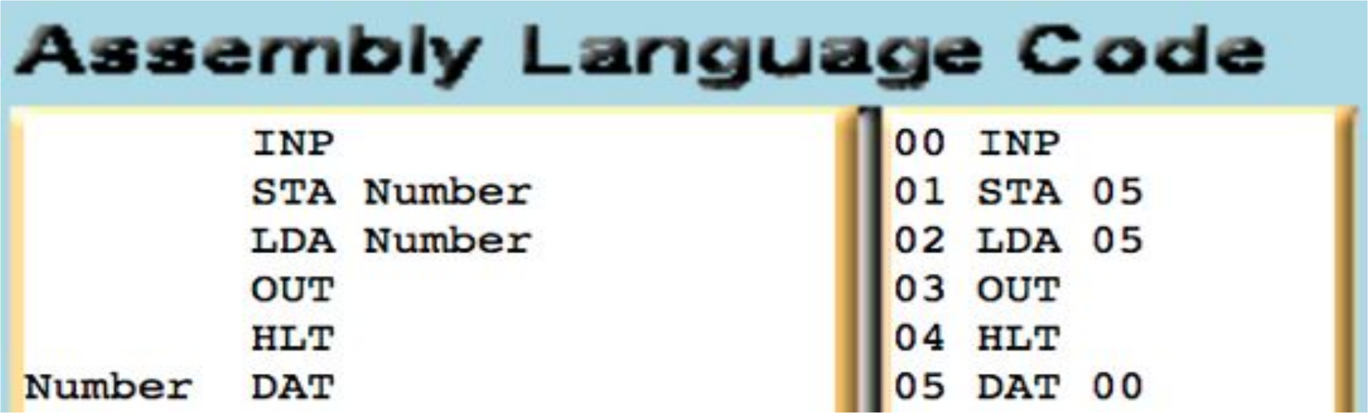
```
INP
OUT
HLT
```

```
00 INP
01 OUT
02 HLT
```

Assembly Language Functions

Function	LMC Mnemonic	LMC Code	What does it do?
Store	STA	3 _ _	Copies the value from the Accumulator and places it in an allocated memory location referred to by the variable name given.
Load	LDA	5 _ _	Copies the value stored at the memory location, given by the variable, into the Accumulator.
Data	DAT		Reserves a memory location to store data. This location can be referred to by the given variable name.

Visualising a Program Running


Assembly Language Code

```
         INP
         STA  Number
         LDA  Number
         OUT
         HLT
Number   DAT
```

```
00  INP
01  STA  05
02  LDA  05
03  OUT
04  HLT
05  DAT  00
```

Storing and Loading

1. Create a program which takes in and stores two inputs from the user and outputs the first input followed by the second input.

2. Create a program which takes in and stores four inputs from the user and always outputs the third input to the user.

3. Create a program which takes in three inputs and outputs them in reverse order.

Addition and Subtraction (1)

1. Create a program which takes and stores in two inputs from the user and outputs the sum of them.

2. Create a program which takes in three numbers and stores them and then outputs the sum of the first two numbers with the third subtracted.

Addition and Subtraction (2)

1. Create a program which takes in a number, doubles it and outputs the result.

2. Create a program which takes in a number and multiplies it by eight.

Challenge - Create a program which takes in a number and multiplies it by forty.

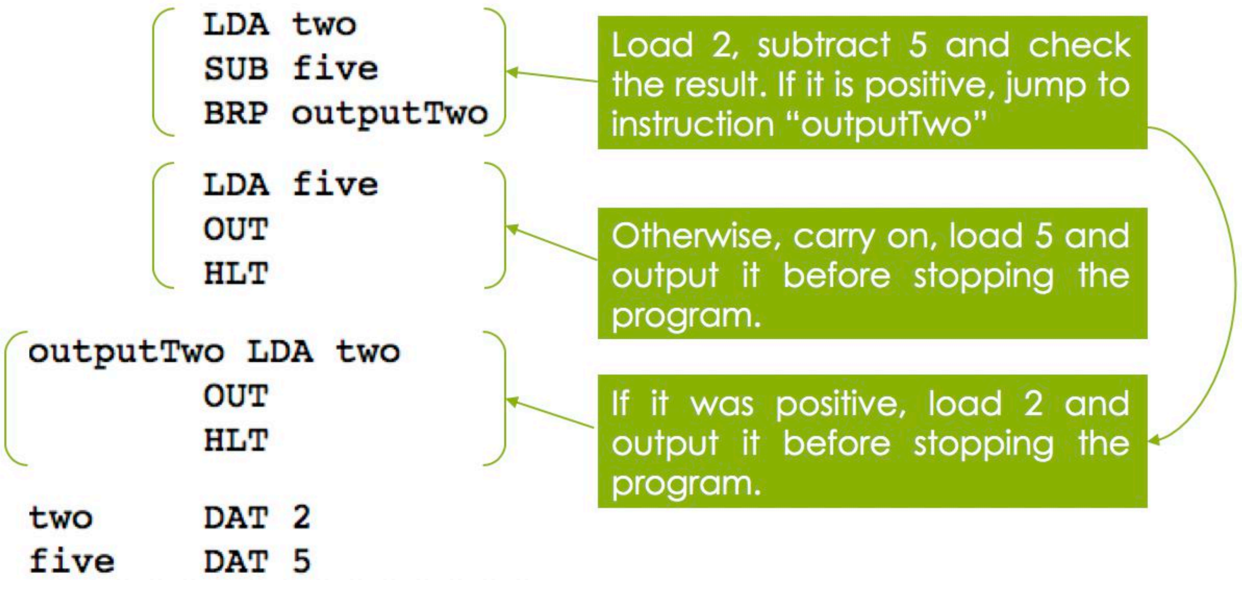
Looping

1. Create a program which allows the user to input numbers indefinitely and outputs each number.

2. Create a program which allows the user to input numbers indefinitely and outputs the running total after each entry.

Comparing Values in LMC

In Little Man Computer we don't have "if statements" like we have in Python for comparisons. The only way to branch based on a condition is to do a subtraction and then branch based on the result.



```
          LDA two
          SUB five
          BRP outputTwo

          LDA five
          OUT
          HLT

outputTwo LDA two
          OUT
          HLT

two       DAT 2
five      DAT 5
```

Load 2, subtract 5 and check the result. If it is positive, jump to instruction "outputTwo"

Otherwise, carry on, load 5 and output it before stopping the program.

If it was positive, load 2 and output it before stopping the program.

Conditional Branching

1. Create a program which allows the user to input two numbers and outputs the smallest number. Hint: if you do a - b and the number is positive, then a is bigger than b.

2. Create a program which allows the user to input two numbers and checks if they're equal. Only output the number if they are equal.

3. Create a program that repeatedly takes in inputs and only outputs them if they are zero.

4. Similar to 3, create a program which outputs everything except zeroes.

Sequences (Mathematics GCSE)

In order to calculate the equation for a given sequence of numbers we must first look at the difference between them e.g.

Index term: 1 2 3 4 5

+2 +2 +2 +2

Number: 3 , 5 , 7 , 9 , 11

The difference between each term is + 2.

So the number in front of the nth term in our equation must be 2, i.e. **2n**.

If we try inserting the index term into our nth term equation **2n** does the answer match up correctly? 2 x 1 = 2

What should be add to correct this? **+ 1**

Therefore our equation is: **2n + 1**

Sequences

For the following sequences:

a. Write out the nth term equation.

b. Calculate the 20th term in the sequence

1. 7, 8, 9, 10, 11 . . .

2. 3, 6, 9, 12, 15 . . .

3. 12, 17, 22, 27, 32 . . .

4. -6, -2, 2, 6, 10 . . .

5. 3, -3, -9, -15, -21 . . .

6. a. Write out the first 5 terms of the sequence given by 3n - 7.

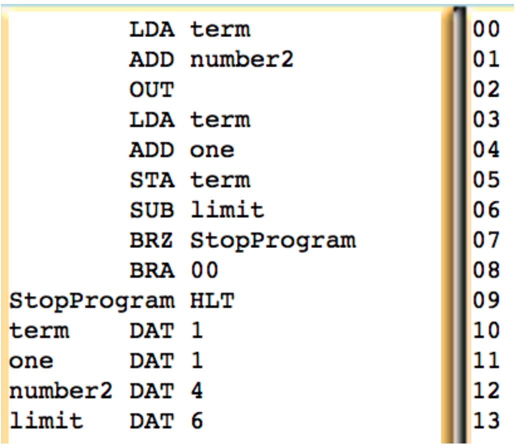
   b. Calculate the 15th term of the sequence.

How to Implement This?

Now we're going to implement this nth term equation in LMC to produce the first 5 terms in the sequence: 5, 6, 7, 8, 9 …

Creating Your Own Sequences

You can use this code as a starting point for creating your own sequences. What would we change in order to make the sequence n + 8?



```
            LDA term                 00
            ADD number2              01
            OUT                      02
            LDA term                 03
            ADD one                  04
            STA term                 05
            SUB limit                06
            BRZ StopProgram          07
            BRA 00                   08
StopProgram HLT                      09
term        DAT 1                    10
one         DAT 1                    11
number2     DAT 4                    12
limit       DAT 6                    13
```

Creating Your Own Sequences

For the following sequences, write down the first 5 terms and then write down the specific term in each question:

A. n - 7: First five terms: ______________________________

12th term: ______________________________

B. 2n + 4: First five terms: ______________________________

15th term: ______________________________

C. 2n - 6: First five terms: ______________________________

11th term: ______________________________

Advanced LMC

1. Create a program which take in inputs and outputs the positive value, i.e. if it's negative, you output the positive, -3 would output 3.

2. Create a program which take and input, outputs that value and then counts down and outputs every value until it reaches 0 (or counts up to - if the value is negative).

3. Create a program which takes two inputs and checks if they have the same sign (both positive or both negative). If they have the same sign output a zero, otherwise output a one.

4. Create a program which takes two inputs and returns the remainder if you divide the first input by the second. (Don't worry about negative numbers, but dividing zero by a number and dividing a number by zero should be considered.)

Very Advanced LMC

Create a program which takes in an input and outputs all of the numbers in the Fibonacci sequence up to that input number.

The Fibonacci sequence is 1, 1, 2, 3, 5, 8, 13, 21 . . .

You can set one variable to 1 at the beginning to help. No cheating!

Notes

technocamps

@Technocamps

Find us on Facebook