**Isaac A-level Booster – Assembly Language**

**Handout 3 – LMC Programming Tasks**

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Description** | **Mnemonic** | **Numeric Code** |
| Load | Loads value into the ACC | LDA | 5xx |
| Store | Stores a value from ACC to RAM | STA | 3xx |
| Add | Adds a value to the ACC | ADD | 1xx |
| Subtract | Subtracts a value from the ACC | SUB | 2xx |
| Input | Takes input from user and stores in ACC | INP | 901 |
| Output | Outputs value from ACC | OUT | 902 |
| End | Halts the program | HLT | 000 |
| Branch if zero | Branches if the ACC is zero | BRZ | 7xx |
| Branch if zero or positive | Branches if the ACC is zero or positive | BRP | 8xx |
| Branch always | Branches always | BRA | 6xx |
| Data storage | Used to store one piece of data | DAT |  |

Using the LMC simulator:

1. **OPTIONS > Clear memory** and **RESET** to clear the RAM & registers before each program unless otherwise directed.
2. Type code in lower case, opcodes will be changed to capitals. Labels are case-sensitive so beware.
3. After typing code, **Submit** then **ASSEMBLE INTO RAM**.
4. If it assembles correctly, **RUN** then watch execution.
5. Use **STOP** to pause and resume a program,   
   **<<** **>>** will change the speed.
6. Use the **STEP** button to step through watching the values of the registers, RAM, input and output to debug anything.
7. The website saves nothing, so copy/paste into Word or Notepad to save.

LMC Links: <https://zigzageducation.co.uk/lmc> and <https://peterhigginson.co.uk/lmc/>

**Activity 1: Sequence in LMC**

**Predict:** What do you think that the following piece of code will do?

|  |  |  |
| --- | --- | --- |
| **Instruction** | **Mnemonic** | **Machine code** |
| Input | INP | 901 |
| Output | OUT | 902 |
| HLT | HLT | 000 |

**Run:** What happened when you ran the code?

**Investigate:**

What is the value of the Accumulator after the program has run?

What mnemonic would store the value in the Accumulator within a numbered address location?

**Modify:** Modify the program, after the OUT instruction store the value in the Accumulator in RAM address 009. Paste your code below.

**Make:** Make a program that asks the user to enter 3 numbers, and stores them all within separate address locations. The program should output the final number entered. Paste your code below

*Beware of overwriting existing instructions or data stored within address locations. There is nothing to prevent you from doing this*

**Activity 2: Arithmetic in LMC**

**Predict:** What do you think that the following piece of code will do?

|  |  |
| --- | --- |
| **Mnemonic** | **Machine code** |
| INP | 901 |
| STA 99 | 399 |
| INP | 901 |
| ADD 99 | 199 |
| OUT | 902 |
| HLT | 000 |

**Run:** What happened when you ran the code?

**Investigate:**

What mnemonic would be used in order to change the operator within this program from add to subtract?

**Modify:** Modify the program so that the first number entered is **subtracted** from the second number entered.

*Hint: only one line of code needs to be modified.*

Paste your code below.

**Make:** Make a program that adds together two inputted numbers, before subtracting the sum from a third entered number. Output the result.

Paste your code below.

**Activity 3: Variables in LMC**

**INP**

**ADD FIVE**

**OUT**

**FIVE DAT 5**

**Run**:

What happens when you run the above code?

**Activity 4: Branching in LMC**

**BEG INP**

**ADD TEN**

**OUT**

**BRA BEG**

**TEN DAT 10**

**Run**:

What happens when you run the above code?

**Modify:**

Modify the above code to add 10 to the originally inputted number on each iteration (infinite loop). Paste the code below.

**Predict:**

**LOOP LDA START**

**OUT**

**SUB ONE**

**STA START**

**BRP LOOP**

**HLT**

**START DAT 3**

**ONE DAT 1**

**Run**:

What happens when you run the above code?

**Modify:**

Modify the above code to count down from 5.

**Make:**

Make an LMC program that counts down from an input value.

**Activity 5 – Selection**

Write an LMC program that **outputs the larger of two input values** (using selection)

*Hint: Subtract the first value from the second. If the result is positive the first value is larger.*

5 - 3 = 2 the answer is positive so first number is larger

5 - 6 = -1 the answer is negative so second number is larger

**Hint:** You will need to use the correct branch commands.

**Challenges:**

6. Write an LMC program that **multiplies two input numbers.**

***Hint:*** *3 \* 4 is just 4 + 4 + 4, while 5 \* 2 is just 2 + 2 + 2 + 2 + 2.*

*We can use the loop construct in our countdown program above to count down from one of our input numbers to zero. Inside the loop we should add our other number to the accumulator.*

7. Write an LMC program that **determines if a number is divisible by 5.**

***Hint:*** *If we subtract 5 from 5, the result is zero. If we subtract 5 from 10 we get 5, and then subtract another 5 we get zero. Numbers not divisible by 5 will never get to zero!*