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

**Handout 3 ANSWERS – LMC Programming Tasks**

|  |  |  |
| --- | --- | --- |
| **Instruction** | **Mnemonic** | **Numeric Code** |
| Load | LDA | 5xx |
| Store | STA | 3xx |
| Add | ADD | 1xx |
| Subtract | SUB | 2xx |
| Input | INP | 901 |
| Output | OUT | 902 |
| End | HLT | 000 |
| Branch if zero | BRZ | 7xx |
| Branch if 0 or +ve | BRP | 8xx |
| Branch always | BRA | 6xx |
| Data storage | 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 Link: <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 |

The user was asked to input a number and then this same number was output.

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

Prediction correct

**Investigate:**

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

The value input by the user

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

**STA**

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

INP

OUT

STA 9

HLT // optional

**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*

INP

STA 97

INP

STA 98

INP

STA 99

OUT

HLT // optional

**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?

The user was asked to input two numbers. They were added together and result output.

**Investigate:**

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

SUB

**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.

INP

STA 99

INP

SUB 99

OUT

**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.

INP

STA 99

INP

ADD 99

STA 99

INP

SUB 99

OUT

**Activity 3: Variables in LMC**

INP

ADD FIVE

OUT

FIVE DAT 5

**Run**:

What happens when you run the above code?

Inputs a number, adds five, outputs the result.

**Activity 4: Branching in LMC**

BEG INP

ADD TEN

OUT

BRA BEG

TEN DAT 10

**Run**:

What happens when you run the above code?

Inputs a number, adds ten, outputs the result then starts again. Runs forever.

**Modify:**

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

INP

BEG ADD TEN

OUT

BRA BEG

TEN DAT 10

**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?

The program counts down from 3 to 0, outputting the count each time.

**Modify:**

Modify the above code to count down from 5.

LOOP LDA START

OUT

SUB ONE

STA START

BRP LOOP

HLT

START DAT 5

ONE DAT 1

**Make:**

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

INP

STA START

LOOP LDA START

OUT

SUB ONE

STA START

BRP LOOP

HLT

START DAT

ONE DAT 1

**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

INP

STA NUM1

INP

STA NUM2

SUB NUM1

BRP SEC

LDA NUM1

OUT

HLT

SEC LDA NUM2

OUT

HLT

NUM1 DAT

NUM2 DAT

**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.*

INP // input multiplicand

STA NUM1 // store as num1

INP // input multiplier

STA NUM2 // store as num2

LOOP LDA TOTAL // load running total (initially 0)

ADD NUM1 // add multiplicand

STA TOTAL // store running total

LDA NUM2 // load multiplier

SUB ONE // decrement by one

STA NUM2 // store multiplier

SUB ONE // we want to stop at 1 not 0

BRP LOOP // if multiplier > 0 loop

LDA TOTAL // else load the total

OUT // output it

HLT // stop

NUM1 DAT

NUM2 DAT

ONE DAT 1

TOTAL DAT 0

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!*

INP // input a number

LOOP BRZ TRUE // number must be div by 5 so jump to "TRUE"

SUB FIVE // subtract 5

BRP LOOP // if still positive or zero, loop back

LDA ZERO // we've gone negative so not divisible by 5

OUT // output a zero

BRA END // jump to end

TRUE LDA ONE // this branch executes if we hit zero exactly

OUT // output a 1

END HLT // halt

ZERO DAT 0

ONE DAT 1

FIVE DAT 5