**Battle Tanks Help Sheet ~ For Teachers and Students**

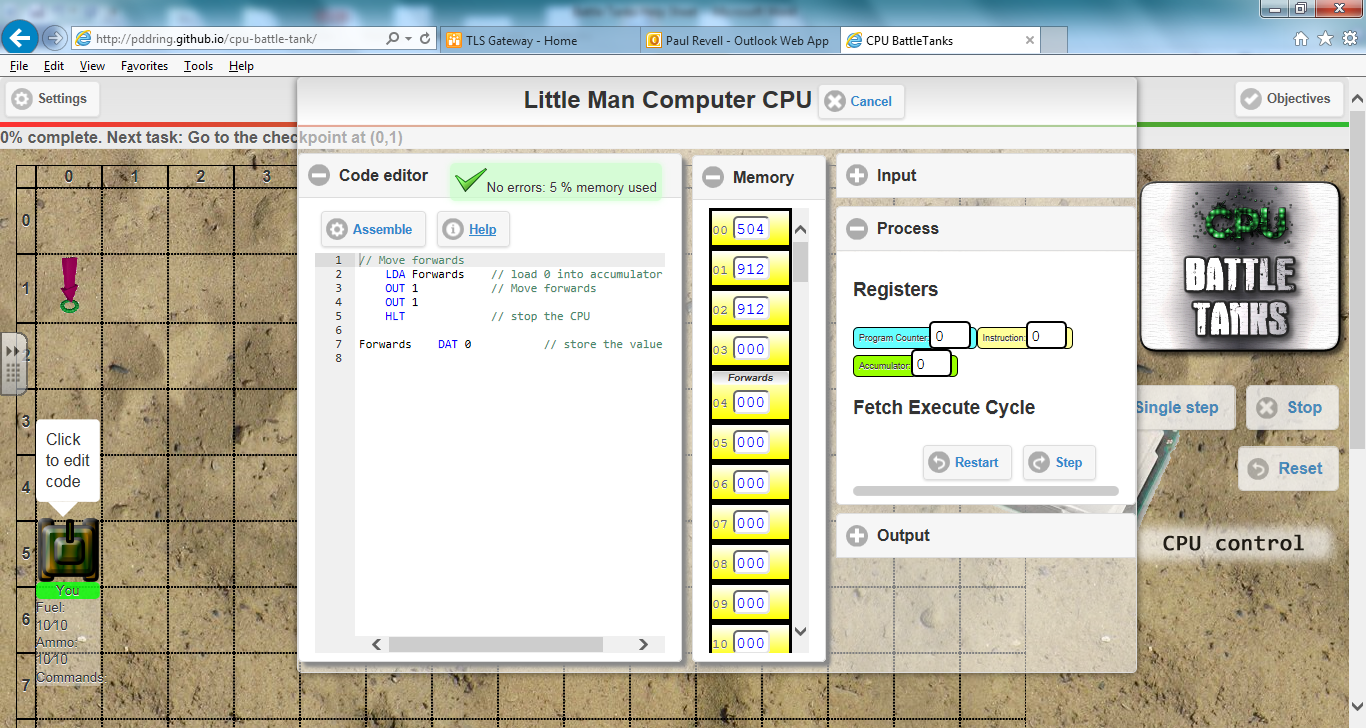
Battle Tanks is a way of learning about low **level computer code** and the **parts of the CPU**. It is based on the code used in The Little Man Computer, so most students will need an LMC command sheet to help.

To find Battle Tanks on the Internet, search for ‘Battle Tanks Little Man Computer’. It has been uploaded to GitHub by Mr P Dring, the teacher who made it.

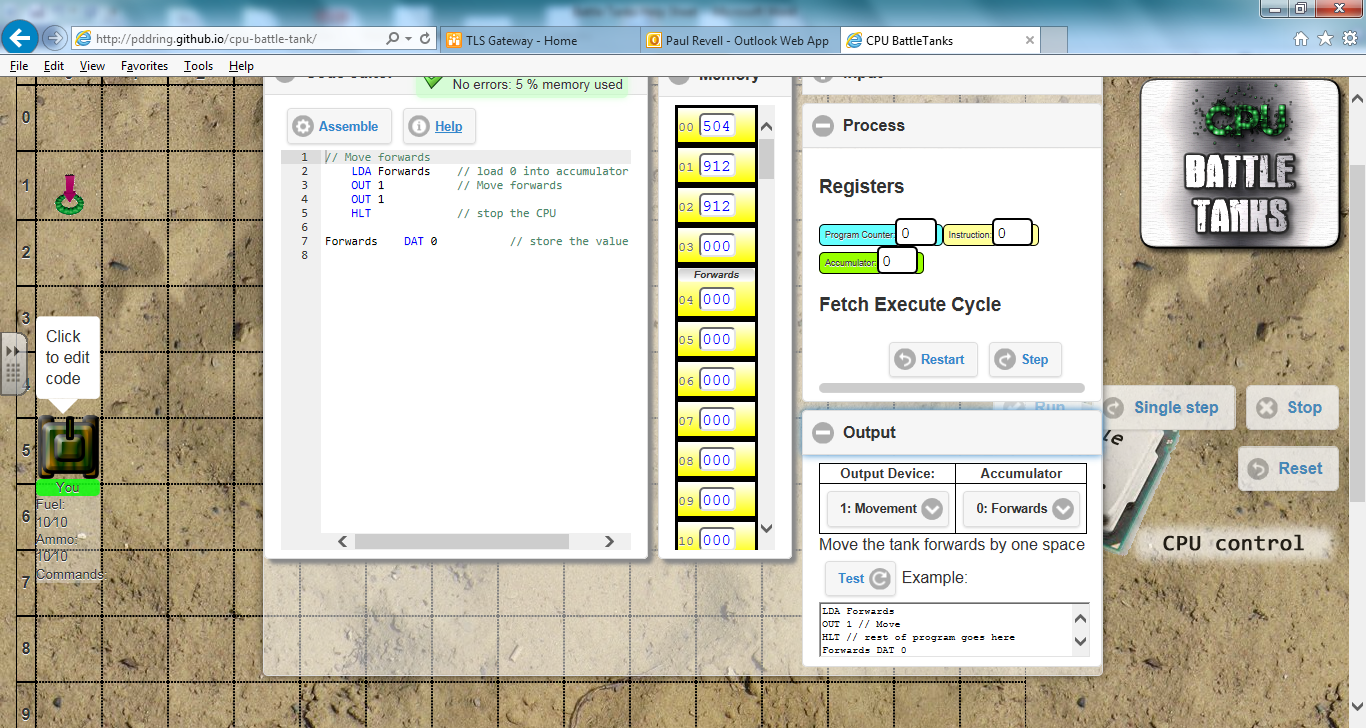
The landscape is a grid and just like in Geography, the ‘across’ co-ordinate comes first. Remember ‘along the corridor and up the stairs’.

Practice Level is about moving the tank by clicking arrow controls.

**Getting Started With Level 1**



When you click the message above the tank, the code editor appears and this shows commands in an assembly language. You have to add to the code to get the tank to complete its mission. The machine code in the computer’s memory and the contents of some processor registers are shown. Teachers can use this to explain about the parts of a CPU.

The tricky bit is understanding the Output Section, so click the plus sign. In the code editor, LDA means Load to the Accumulator. DAT means data storage place. Outputting different numbers makes the tank do different things. When you output 1 the tank moves. Output 3 and the gun aims. Exactly what move (or aim), depends on what is in the accumulator. In this case 0 had been loaded and the output is 1 so the tank moves forward. The code could have said LDA 0, but storing a zero in Forwards and then loading Forwards makes the code a bit more meaningful.

If you add West DAT 006 and code LDA East then OUT 2, the 2 means ‘do a steer’ and because you have loaded up a 6 it will steer West. This should be enough to get you started, together with the LMC command sheet.

The idea is to work through the challenges and to use the Battle Tank screen as a prompt to discuss how the hardware of the CPU works to run a program. Enjoy the challenges and learn how a computer works!

**Little Man Computer Assembly Language (not all needed for Battle Tanks) adapted from Stephen Chen, York University**

**LOAD** LDA  
Load the contents of the given memory address (example East) onto the accumulator

**STORE** STA   
Store the contents of the accumulator to the memory address that follows.

**ADD** ADD   
Add the contents of the given memory address onto the accumulator.

**SUBTRACT** SUB   
Subtract the contents of the given memory address from the accumulator.

**INPUT** INP   
Copy the value from the "in box" onto the accumulator (calculator).

**OUTPUT** OUT   
Copy the value from the accumulator to the "out box" (in this case the Battle Tank)

**END** HLT   
Causes the Little Man Computer to stop executing your program.

**BRANCH IF ZERO** BRZ   
If the contents of the accumulator are 000, the PC (program counter) will be set to the given address.

**BRANCH IF ZERO OR POSITIVE** BRP   
If the contents of the accumulator (calculator) are 000 or positive (i.e. the negative flag is not set), the PC (program counter) will be set to the given address.

**BRANCH ALWAYS** BRA   
Set the contents of the accumulator (calculator) to the given address.

**DATA LOCATION** DAT   
When compiled, a program converts each instruction into a three-digit ‘machine code’.  These codes are placed in sequential memory addresses.  Instead of a program component, this instruction will reserve the next memory address for data storage. Example if you write East DAT 002 on line 11 of the code editor, the computer will reserve memory address 11 for the data and in your code if you ever LDA East it will go and load whatever is held in memory address 11.