

Simple 8-Bit Assembler Simulator

Instructions

Open a web browser and navigate to <https://schweigi.github.io/assembler-simulator/index.html>

This simulator provides a simplified assembler syntax (based on Netwide Assembler [NASM](#)) and is simulating a x86 like cpu. In depth documentation and introduction to assembler can be found on the following websites:

- [Assembly - Wikipedia](#)
- [The Art of Assembly Language Programming](#)
- [NASM Language Documentation](#)
- The simulator consists of a 8-bit cpu and 256 bytes of memory. All instructions (code) and variables (data) needs to fit inside the memory. For simplicity every instruction (and operand) is 1 byte. Therefore a MOV instruction will use 3 bytes of memory. The simulator provides a console output which is memory mapped from 0xE8 to 0xFF. Memory mapped means that every value written to this memory block is visible on the console.

User Interface

The screenshot shows the Simple 8-bit Assembler Simulator interface. Red arrows point to various components with labels:

- Execution Controls:** Points to the Run, Step, and Reset buttons at the top left.
- Assembler Code Program:** Points to the Code (Instruction Set) input area.
- Program Output:** Points to the Output console showing "Hello World!".
- CPU Registers & Flags:** Points to the Registers / Flags section, which includes a table for registers A, B, C, D, IP, SP, Z, C, F. The IP register is highlighted with the value 17.
- 256 bytes of memory:** Points to the RAM section, which displays a 16x16 grid of memory addresses and values.
- CPU Execution Speed Options:** Points to the Clock speed dropdown menu, which is set to 4 Hz.
- Display Control Options:** Points to the bottom right area containing checkboxes for "Instructions: Hide", "View: Decimal", and "Register addressing: A: Show B: Show C: Show D: Show".

The Code (Instruction Set) area contains the following assembly code:

```
; Simple example
; Writes Hello World to the output

300 start
hello: DB "Hello World!"; Variable
      DB 0 ; String terminator

start:
      MOV C, hello ; Point to var.
      MOV D, 232 ; Point to output
      CALL print

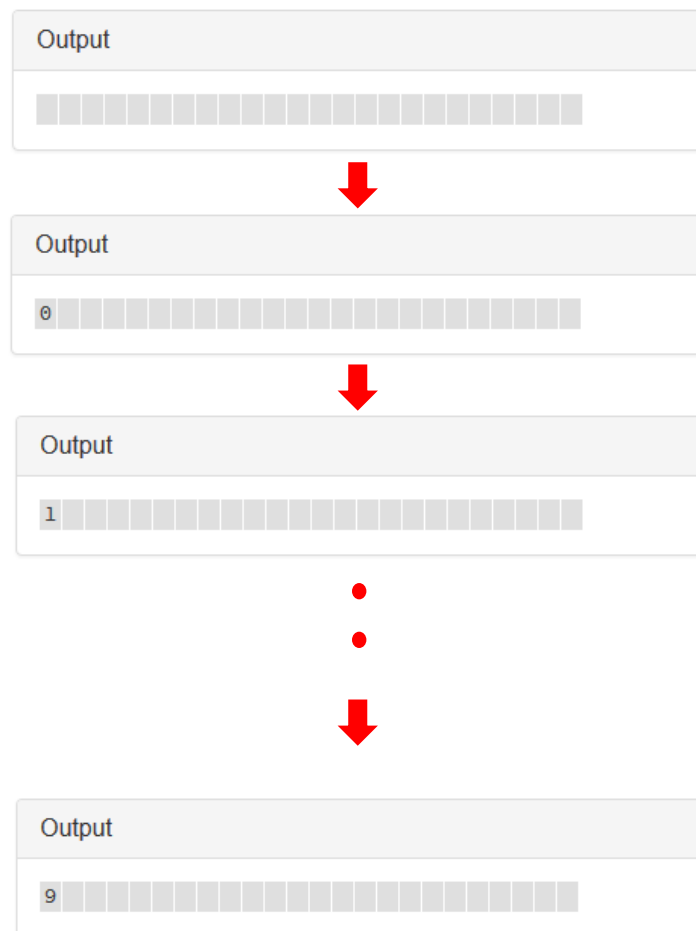
      HLT ; Stop execution

print: ; print(C:=from, D:=to)
      PUSH A
      PUSH B
      MOV B, 0
.loop:
      MOV A, [C] ; Get char from var
      MOV [D], A ; Write to output
      INC C
      INC D
      CMP B, [C] ; Check if end
      JNZ .loop ; jump if not

      POP B
      POP A
      RET
```

Part A

Write an assembler program that, when executed, will output the accumulated value of a counter starting at 0 and ending at 9. Write the counter value to byte **0xE8** so that it will appear in the simulator output window as shown below.



Thoroughly test your solution to see that it works as required. When you have finished the testing, copy the assembler code, paste it into a text file named HW8a.txt, save it, and attach it to the assignment.

Part B

Modify your assembler program from Part A so that it will output the accumulated value of a counter starting at 9 and ending at 0. This time have the output write to consecutive output bytes starting at **0xE8**. The resulting output will look like the figure below which shows the simulator output window.



Thoroughly test your solution to see that it works as required. When you have finished the testing, copy the assembler code, paste it into a text file named HW8b.txt, save it, and attach it to the assignment.