Abdelhafid Boussouf University Center, Mila
Institute of Mathematics and computer science
Department of Computer Science
2nd Year Computer Science

Year: 2025/2026
Module: Computer Architecture

PW N° 01

What is MIPS Assembly?

MIPS Assembly is a low-level programming language used for programming MIPS (Microprocessor without Interlocked Pipeline Stages) processors. MIPS Assembly is human-readable, but it's much closer to the language that computers understand (machine code) than high-level languages like Python or Java.

Why learn MIPS Assembly?

You might be wondering, "Why should I learn MIPS Assembly when there are so many high-level languages to choose from?" It's a valid question! But, here's the thing: learning MIPS Assembly gives you a deeper understanding of how computers work at their core. It's like learning the secrets of the universe, but for computers.

What you will learn?

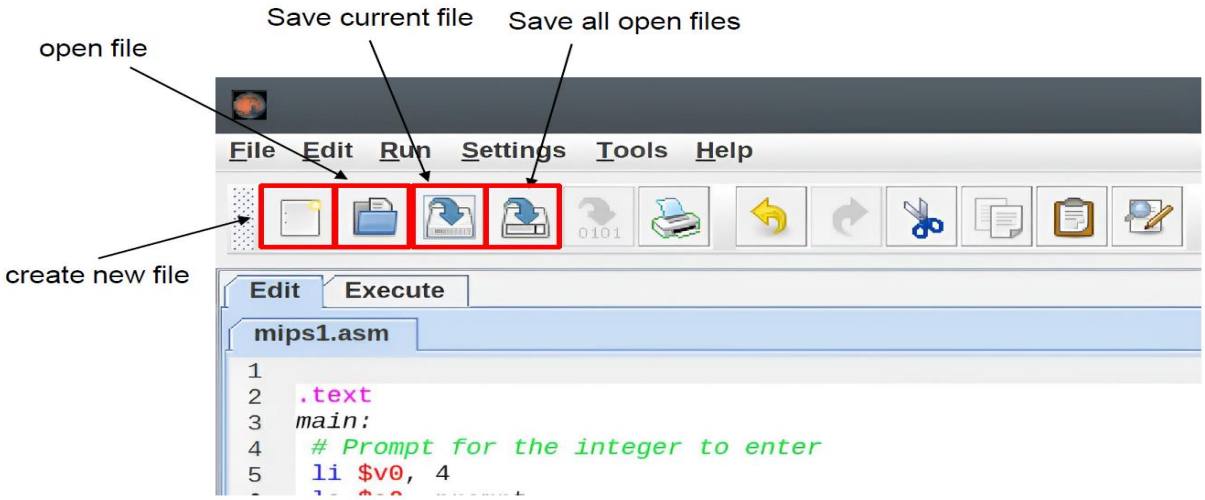
In this TP you will learn:

1. to download, install, and run the MARS IDE.
2. what are registers, and how are they used in the CPU.
3. register conventions for MIPS.
4. how memory is configured for MIPS.
5. to assemble and run a program in MARS.
6. the syscall instruction, and how to pass parameters to syscall.
7. what immediate values are in assembly language.
8. assembler directives, operators, and instructions.
9. to input and output integer and string data in MIPS assembly.

The MARS IDE

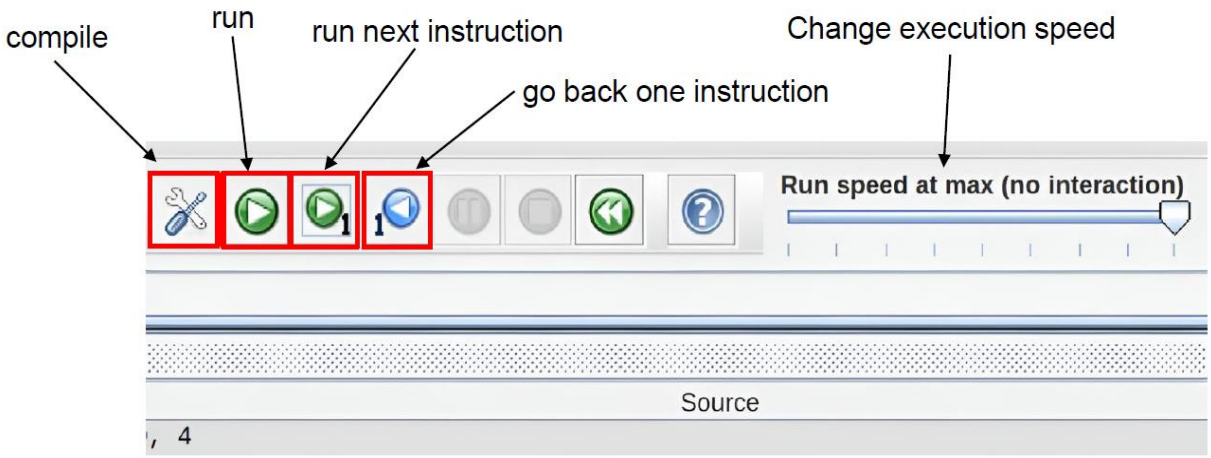
We will use an IDE called the MIPS Assembler and Runtime Simulator (MARS). There are a number of MIPS simulators available, some for educational use, and some for commercial use.

MARS is an executable jar file, so you must have the Java Runtime Environment (JRE) installed to run MARS.



Before you can compile your MIPS assembly code you must first save the file.

- The option to compile will not be available until you save your code.
- The option to run your code will not be available until you compile your code.



.

First program :

1. In the MARS software create a new file and copy the assembler program below
2. Determine what the program does using the help

```
        .data
vars:   .word 5
        .word 10
        .text
        la $t0, vars
        lw $t1, 0($t0)
        lw $t2, 4($t0)
saut:   bge $t1, $t2, exit
        move $a0, $t1
        li $v0, 1
        syscall
        addi $t1, $t1, 1
        j saut
exit:   li $v0, 10
        syscall
```