

Project II – Interim report

**Course Name:** Computer Organization

**Course Number:** (**60610MEEOS-CME0437**)

**Department:** Computer Engineering

**Lecturer:** Prof. Dr. Ercan SOLAK

**Year** (2022-2023)

**Students Name:**

Beyza Serap Sevim - 1804010009

Leyla Abdullayeva - 1904010038

Yusef Murad Rayyan - 1904010031

# **General Description:**

The processor is an 8-bit single-cycle machine. It has 3 registers: a program counter (PC) that holds the memory address of the next instruction to be executed, an address register (AR) that holds the memory address for load/store operations and the target address for jumps, and a general-purpose register (GPR) that holds data values and intermediate results of operations. The instructions and memory words are 8 bits wide, allowing a maximum of 256 different instructions and 256 words of memory. The processor has a basic set of 6 instructions for performing arithmetic and memory operations.

# **Instruction Set Architecture (ISA):**

The instruction set architecture (ISA) of the processor includes the following 6 instructions:

1. **GET:** Load a value from memory into the general-purpose register.
   * **Example**: LOAD 0001 (loads the value at address 0001 into the general-purpose register)
2. **SET:** Store a value from the general-purpose register into memory.
   * **Example**: STORE 0001 (stores the value in the general-purpose register at address 0001)
3. **ADD:** Add a value from memory to the value in the general-purpose register.
   * **Example**: ADD 0001 (adds the value at address 0001 to the value in the general-purpose register)
4. **JUMP:** Jump to a specific memory address.
   * **Example**:  JUMP 0010 (sets the program counter to address 0010
5. **SUB:** Subtract a value from memory from the value in the general-purpose register.
   * **Example**:  SUB 0001 (subtracts the value at address 0001 from the value in the general-purpose register)
6. **JUMPZ:** Jump the instruction, if the value of the register is 0.

# **Coding and explanations**

**Sample program to declare and sum the elements of an array.**

 Elements of Array: 1,2,3,4,5.

# Set up the counter and initialize it to 0.

SET $PC, 7   # STORE the value 0 in memory location 7 (program counter).

# Load the first element of the array into the GPR.

GET $GPR, 0     # LOAD the value at memory location 0 into the GPR.

 # Store the value in the GPR into memory location 6.

SET $GPR, 6 # STORE the value in the GPR into memory location 6.

 # Increment the counter.

PUSH $GPR, $GPR, 7 # ADD the value in memory location 7 to the GPR.

SET $GPR, 7 # STORE the value in the GPR back into memory location 7.

 # Check if the counter is less than the length of the array.

GET $GPR, 5 # LOAD the value 5 (length of the array) into the GPR.

PULL $GPR, $GPR, 7 # SUB the value in memory location 7 from the GPR.

JUMPZ 18, END# If the result is 0, jump to instruction 18 (END).

# Load the next element of the array into the GPR.

GET $GPR, 7 # LOAD the value in memory location 7 (counter) into the GPR.

PUSH $GPR, $GPR, 8 # ADD the value 8 (size of each element in the array) to the GPR.

GET $AR, $GPR # LOAD the value in the AR (memory address) into the GPR.

PUSH $GPR, $GPR, $zero # ADD the value in the GPR to the GPR.

SET $AR, $GPR   # STORE the value in the GPR back into the AR.

GET $AR, $GPR   # LOAD the value at the memory location specified by the AR into the GPR.

# Add the value in the GPR to the value in memory location 6.

PUSH   $GPR, $GPR, 6 # ADD the value in memory location 6 to the GPR.

SET $GPR, 6 # STORE the value in the GPR back into memory location 6.

# Increment the counter.

PUSH $GPR, $GPR, 7 # ADD the value in memory location 7 to the GPR.

SET $GPR, 7 # STORE the value in the GPR back into memory location 7.

# Jump back to the beginning of the loop.

JUMP 4    # JUMP to instruction 4

END:

GET $GPR, 01000100 # This value is representing the $v0, 10 in MIPS.

SYSCALL

# **Video Link**

[Second\_project\_video\_1804010009\_1904010038\_1904010.mp4](https://drive.google.com/file/d/1ps5QPXIw69ynQQHh4O19XiXMtZbaTTjL/view?usp=share_link)