****

**Project 1 Report:**

**RV32I Disassembler**

**Basel Shabana 900162270**

**Caren Khafagy 900170445**

**Design and implementation:**

The project consists of 47 instructions in which 37 instructions are divided into 6 Instruction formats and 10 system instructions. Each instruction in RV32I is 32 bits, so by reading the first 7 bits (opcode) we can determine where the instruction is located in the instruction format since each set of instructions has a specific opcode. We can differentiate between each instruction in each of the formats by making a switch statement with funct3 (3-bit function code) which is an additional opcode that represents each instruction, however if 2 instructions have the same funct3 we make an if statement with funct7 (7-bit function code) which is an additional opcode.

The 6 Instruction formats are:

* **R-Format**

Integer Register-Register Operations: ADD, SLT, SLTU, AND, OR, XOR, SLL, SRL, SUB, SRA

- Format:instruction rd, rs1, rs2

* **I-Format**

the I-Format contain 2 different set of instructions with 2 different opcodes

Load instructions: LB, LH, LW, LBU, LHU

- Format: instruction rd, offset (rs1)

Integer Register-Immediate Instructions: ADDI, SLLI, SLTI, SLTIU, XORI, SRLI, ORI, ANDI

- Format: instruction rd, rs1, shamt

* **S-Format**

Store instructions: SB, SH, SW

- Format: instruction rs2, offset (rs1)

* **SB-Format**

Conditional branches: BEQ, BNE, BLT, BGE, BLTU, BGEU

- Format: instruction rs1, rs2, offset

* **U-Format**

Upper Immediate values instructions: LUI, AUIPC

- Format: instruction rd,imm

* **UJ-Format**

Jump instructions: JAL, JALR

- Format:

jal rd, offset

jalr rd, rs1, offset

* **System Instruction**

Control and Status Register Instructions: CSRRW, CSRRS, CSRRC, CSRRWI, CSRRSI, CSRRCI

- Format:

instruction rd,offset, rs1

instruction rd,offset, imm

FENCE instruction

- Format: fence pred, succ

Environment Call and Breakpoints: ECALL, EBREAK

**Challenges:**

* We took some time to be able to fully understand the skeleton to be able to start designing the project
* Identifying the necessary masks for obtaining the immediates
* Setting immediate values for different instruction formats
* Setting a stopping condition for the program to break and finish execution

**Limitations:**

* I-immediate represent the value perfectly; however, it doesn’t represent a negative value; we think the reason is it is loaded as an” unsigned int” variable.
* There were not enough files to be able to test all 47 instructions
* Some immediate values are not set perfectly before printing instruction.