Question 1 - Instruction Set 1 – x295

A. Description of x295 instruction set architecture (ISA)

1. Memory model

- **Word size**: 16 bits
- **Size of memory**: $2^m \times n$ bits = $2^{12} \times 16$ bits
  - **m** = 12
  - **n** = 16 bits
    - Note: This is not a byte-addressable ISA.

2. Registers

- **Data type**: 16 bits (i.e., an integer has 16 bits)
- **Number of registers**: 0 registers
  - However, there is a *non-general-purpose* register associated with the stack: the stack pointer register, which we cannot use as operand to our instructions.
  - This is a *Memory Only* instruction set architecture (ISA) - this means that only memory addresses are used as operands.

3. Instruction set (assembly instructions and machine code instructions)

- **Maximum number of instructions**: 16 instructions
  - What would the size of the **opcode** field in each machine code instruction be?
- **Memory addressing mode**: Direct (i.e., absolute)
- **Operand Model**:
  - *3-operand* model – all three operands are memory addresses.
  - In the assembly instructions, the order of the operands is: Src1, Src2, Dest.
  - In the machine code instructions, the order of these operands is: Dest, Src1, Src2.
- **Assembly instructions**:
  - Even though there is a maximum of 16 instructions in this instruction set, in this assignment we shall only define and use a subset of these 16 instructions:

    List of this subset of instructions with their format (syntax) and meaning:

    - `ADD a,b,c` Meaning: `M[c] <- M[a] + M[b]`
    - `SUB a,b,c` Meaning: `M[c] <- M[a] - M[b]`
    - `MUL a,b,c` Meaning: `M[c] <- M[a] * M[b]`

- **Machine code instructions:**
  - **Format**:

opcode	Dest (12 bits)	padding	Src1 (12 bits)	padding	Src2 (12 bits)
4 bits		4 bits		4 bits	

This format is made of 3 words, each word is 16 bits in length (word size).

Note: There is only one format. Therefore, this format is used to form all three machine code instructions corresponding to the three assembly instructions listed above.

  - **Opcode encoding (bit pattern)**:

Opcode (instruction)	**Encoding Bit pattern (4 bits)**
padding	`0000`
`ADD`	`0001`
`SUB`	`0010`
`MUL`	`0011`
…	…

B. Compiling and assembling a C program using our x295 instruction set

1. **Table 1** (on the next page) lists a C program (in the left column) which we "compiled" (by hand) into a **`x295`** assembly program using the assembly instructions defined in our **`x295`** instruction set. This **`x295`** assembly program is listed in the middle column.
2. Then we "assembled" (by hand) our **`x295`** assembly program into a **`x295`** machine code program using the machine code instructions defined in our **`x295`** instruction set. This **`x295`** machine code program is listed in the right column.

Note: The "compiling" and the "assembling" steps have already been done for us!

Table 1

C program (The C code below cannot be modified – it must stay as it is stated below)	x295 assembly program	x295 machine code program
`z = (x + y) * (x - y);`	`ADD x, y, tmp1`	`0001` **<Dest 12 bits>** `0000` **<Src1 12 bits>** `0000` **<Src2 12 bits>**
	`SUB x, y, tmp2`	`0010` **<Dest 12 bits>** `0000` **<Src1 12 bits>** `0000` **<Src2 12 bits>**
	`MUL tmp1, tmp2, z`	`0011` **<Dest 12 bits>** `0000` **<Src1 12 bits>** `0000` **<Src2 12 bits>**
	(where `tmp1` and `tmp2` are memory addresses of memory locations holding these temporary results)	Note: In the above machine code instructions, we abstractly express a memory address (**abstract memory address**) as **<Dest 12 bits>**, **<Src1 12 bits>** and **<Src2 12 bits>** as opposed to using a memory address computed by adding a displacement to the stack pointer (e.g., `4($sp)`) or using an actual memory address (e.g., `0x7ffff0c`) in a **memory address field (fields in purple)** as we have done in our lectures in class. While answering the questions in this assignment, use **abstract memory addresses** as illustrated in this column.

We can use this table as a model to follow when answering the other questions in this assignment.

C. Evaluating our x295 instruction set

First, download the file **`A7_Q1_Table.pdf (or docx)`** from our course web site (below Assignment 7) and open it. It contains a table called **Table 2**.

The next step is to evaluate our **`x295`** instruction set by executing (hand tracing) our **`x295`** assembly program (or its corresponding **`x295`** machine code program although it is easier to use the assembly program) and using the criteria called ***memory traffic***, by counting the number of memory accesses our **`x295`** assembly program makes during its execution. In other words, we count how many time the execution of our **`x295`** assembly program required a word (16 bits) to be read from or written to memory.

Perform this evaluation …

1. By completing the middle and right columns of **Table 2** found in the file **`A7_Q1_Table.pdf (or docx`**) you downloaded, by adding your memory access count as well as explaining how you obtain this count, and lastly …

2. By totaling your counts in the last row of the table. Don't forget the units!