Lab 5

Simple Computer Architecture

Understand the SCA instruction set.
Practice translating between high-level languages and assembly code.
Practice translating between assembly and machine code.

This lab explores the instruction set of the Simple Computer Architecture (SCA) discussed in class. Through structured exercises, you will practice translating pseudo-code from high-level languages to assembly and machine code, and vice versa, reinforcing your understanding of binary representation of low-level languages. You will need the opcode reference table for all exercises.

Exercises

- **Exercise 1:** Translate each assembly instruction into its corresponding machine language instruction. Provide the final answer in 14-bit binary and as a 4-digit hexadecimal value.
  (a) `write`
  (b) `load  B  47`
  (c) `store  A  47`
  (d) `sub  reg`
  (e) `halt`

- **Exercise 2:** Translate each machine code into its corresponding assembly code. Explain what each instruction does.
  (a) `0520`
  (b) `0FC9`
  (c) `0000`
  (d) `3400`

- **Exercise 3:** After a reset, the simple computer's memory contains the following 14-bit instructions (in hex). Trace the fetch-execute cycle step-by-step, hand-executing the program. Maintain a record of main memory and CPU registers, and provide the final output.

```
Address      Content
0x00:        0406
0x01:        0505
0x02:        1000
0x03:        0200
0x04:        3000
0x05:        0002
0x06:        000B
```

  Hint: Hand-decode each instruction after fetching it into the IR to determine its operation. There are **five** instructions in the sequence; the remaining two instructions represent data.

- **Exercise 4:** Consider the following pseudo-code:

```
set num = 10;
while ( num != 0 ){
    print (num);
    num--;
}
```

(a) Translate the above code into assembly language, starting at address `0x00`.

(b) Translate the assembly code obtained in part (a) into machine code (written in hex).

- **Exercise 5:** Translate the following pseudo-codes into assembly and machine codes:

(a)

```
set sum = 0;
set x = 1;
read num;
while ( num != 0 ){
    sum = sum + num + x;
    num = num - 1;
}
print (sum);
```

(b)

```
read x;
read y;
if ( y == 0 )
    x--;
print (x);
```

(c)

```
read x;
read y;
if ( y == 0 ){
    x--;
    print (x);
}
else{
    x++;
    print (x);
}
```