CSCE 2303 – Computer Organization and Assembly Programming
Summer 2020

Homework 3

Problem 1 (RISC-V non-recursive array elements summation) - 10%
Develop a RISC-V assembly function (call it sum1) that accepts an array and its size then return the sum of all the values of that array. Develop a program to test such function. Your test program has to prompt the user to enter the array elements then output the sum. Your implementation cannot use recursion.

Problem 2 (RISC-V recursive array elements summation) - 10%
Repeat Problem 1 but implement the summation function using recursion (call it sum2). A typical C implementation for sum2 is:

```
int  sum (int array[], int length)      {
       if (length == 0)
              return 0;
       return array[0] + sum ( &array[1], length -1 );
}
```

Problem 3 (Mystery recursive function in RISC-V assembly) – 15%
What is the return value of calling `mystery_fn(10)`? Your answer should be given as a comment in the problem source code.

```
int mystery_fn(int n) {
    if (n != 0)
        return n + mystery_fn(n - 1);
    else
        return n;
}
```

Implement mystery_fn() using RISC-V assembly and develop a small program to demonstrate it(prompts the user for input, calls the function and prints out its return value).

Problem 4 (XOR Cipher) – 15%
The simplest encryption method is called the XOR Cipher. Using the XOR Cipher, a string of text can be encrypted by XORing every character of the string with a unique byte (the key). Applying the same operation on the encrypted string decrypts the string to get the original text.
Develop a function (xor_cipher) that accepts two null terminated strings: the text to encrypt/decrypt and the password. The password string characters must be summed and the least significant byte of the sum is used as the XOR byte key. The function uses the key to encrypt/decrypt thetext string as explained above. Develop a small test program to test the function by capturing a string from the user, encrypts it, decrypts it then print it out.

Problem 5 (Recursion, again!) – 10%
Implement the following C function using RISC-V assembly:

```
int fn1(unsigned x) {
       int b;
       if (x == 0) return 0;
       b = x & 0x1;
       return b + fn1(x >> 1);
}
```

Develop a program to test the function “fn1”. Your program has to ask the user to enter an integer, call the function and finally print out the returned value. Add comments to your code to explain the relation between the finction input and output.

Problem 6 (A self-disassembler) – 25%
Develop a modular program in RISC-V assembly to disassemble the first 25 instructions of its Text Section. For each instruction, it has to print the instruction format and the instruction name only. You need to support R-format and I-format instructions only. Print “UNKOWN INSTR” for all other instructions.

Problem 7 (A Melody Player) – 15%
Develop a RISC-V assembly program to play a melody. The notes and the duration of every note are given below

```
       .data
notes:
       .byte 76,76,0,76,0,72,76,0,79,0,0,0,67,0,0,0,72,0,0,67,0,0
       .byte 64,0,0,69,0,71,0,70,69,0,67,76,79,81,0,77,79,0,76,0,72,74,71,0
       .byte 0,72,0,0,67,0,0,64,0,0,69,0,71,0,70,69,0,67,76,79,81,0,77,79,0,76,0,72,74,71,0,0
duration:
```

```
.byte 75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75
.byte 75,75,75,75,75,75,75,75,75,75,100,100,100,75,75,75,75,75,75,75,75,75,75
.byte 75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,75,100,100,100,75,75,75,75,75,75,75,
.byte 75,75,75,75,75
```

Please note that 0 is used to indicate a silent note (nothing will be played for the given duration). Please refer to ecall service 33 in RARS documentation to learn how to play a midi note.

Submission guidelines:

- Deadline: Before June 29th, 2020 11:59PM
- The solutions must be assembled and run on RARS. Submissions that cannot be assembled correctly will receive "0".
- Name the source code file of every problem after its number (e.g., p1.s, p2.s, …).
- Zip all the files into hw3_your_name.zip file (e.g., hw3_aya_ali.zip)