Homework #1. RISC-V

1. Venus Simulator (12 pt, 3pt for each)

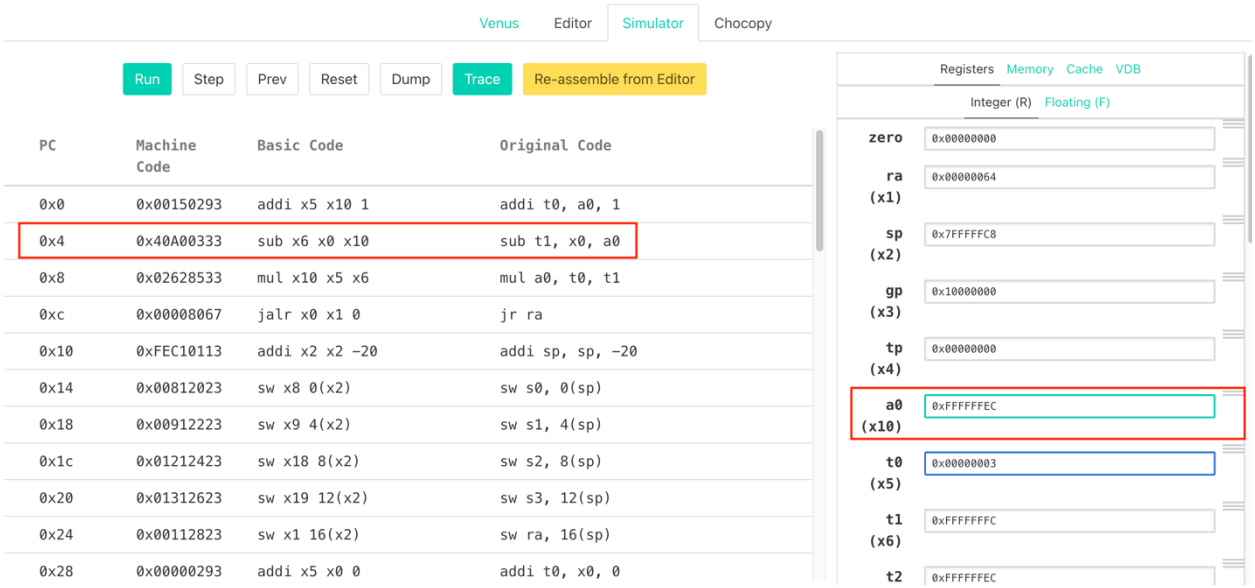
```
.data
.word 2, 4, 6, 8
n: .word 10

.text
main:   add     t0, x0, x0
        addi    t1, x0, 1
        la      t3, n
        lw      t3, 0(t3)
fib:    beq     t3, x0, finish
        add     t2, t1, t0
        mv      t0, t1
        mv      t1, t2
        addi    t3, t3, -1
        j       fib
finish: addi    a0, x0, 1
        addi    a1, t0, 0
        ecall # print integer ecall
        addi    a0, x0, 10
        ecall # terminate ecall
```

1-1. Run the program to completion. What number did the program output?
(Please write down the **exact number** in answer sheet in Groom)

1-2. This program represents Fibonacci sequence. Let's call this function as fib(n) where the n is n-th fibonaaci number. What is the value of n for this program?
(Please write down the **exact number** in answer sheet in Groom)

1-3. At what address is `n` stored in memory?
(Please write down the answer in Groom **including hexadecimal prefix**. For example, if the address is 0xffffffff then the answer is 0xffffffff)

1-4. Without actually editing the code (i.e. Do not modify the code in “Editor” tab), have the program calculate the 14th fib number (0-indexed) **by *manually* modifying the value of a register.** Please write down the exact PC address modifying registers.
For example (see the below figure), If I modify the **register a0 (x10)** value while executing the instruction “`sub x6 x0 x10`” then write down the PC addresss (i.e., **0x4**) in the answer sheet.



2. Translation: RISC-V to C (12 pt, 4pt for each)

Open and read the files p2.c and p2.s The assembly code file (p2.s) is a translation of the given C program into RISC-V.

2-1. What register representing the variable k ?
(Please write register name **without space** e.g., a0, not a 0)

2-2. What registers acting as pointer to the src and dst arrays?
(Please write the two register **with “comma” separated** e.g., t0,t1)

2-3. Which assembly code for the loop found in C code (p2.c)
(Please write **single** instruction which indicates the loop in p2.s)

3. RISC-V Instruction (18 pt, 6pt per each)

For each RISC-V instruction sequence below, provide the hex values of the specified registers after each sequence has been executed. In this problem, we assume that all registers are initialized to 0 prior to each instruction sequence.
Each instruction sequence begins with the line (`. = 0x0`) which indicates that the first instruction of each sequence is at address 0.

* . = 0x0 명령어는 현재위치를 나타내는 "." 을 0x0 으로 초기화한다는 뜻입니다.
* unimp 는 "unimplemented"의 약자로, RISC-V 에서 예외처리를 위해 사용되는 명령어입니다. 현재 구현되지 않은 instruction 을 실행시킬 때 예외처리를 위해 사용됩니다. 문제에서는 큰 의미 없습니다.

```
.  = 0x0

      lui x2, 2
      addi x3, x2, 9
      xori  x4, x3, 0xfff

end: unimp
```

3.1 what is value in x2 (please write hex value e.g., 0x00002000)

3.2 what is value in x3 (please write hex value e.g., 0x00002000)

3.3 what is value in x4 (please write hex value e.g., 0x00002000)

4. RISC-V Function Calling (58 pt / 5 pt for each)

Complete the implementation of map (**my_map.s**) by filling out each of these **11 markers** with the appropriate code. Furthermore, provide a sample call to map with **double** as the function argument.
Please modify the "### YOUR CODE HERE ###" region. Please do not modify other code region.

In this problem, map method will take two parameters; the first one will be the address of head node of a single-linked list whose values are 32-bit integers.In C, we can represent below:

```
struct node {
    int value;
    struct node *next;
}
```

Second parameter will be the address of a function that takes one **int** as an argument and returns an **int** (i.e., we will use jalr instruction to call this function on the list node values)

Our map function will recursively go down the list, applying the function to each value of the list and sorting the value returned in that corresponding node. In C, we can represent below:

```
void map (struct node *head, int (*f)(int))
{
    If (!head) return;
    head->value = f(head->value);
    map(head->next, f);
}
```

Here, you don't have to understand what ` (*f)(int)` expression is. Don't worry too much about it. It means that f is a pointer to a function that takes an int as an argument (i.e., Function pointer)

The result of running in Venus Simulator should be as follows.

```
9 8 7 6 5 4 3 2 1 0
18 16 14 12 10 8 6 4 2 0
```

The first line is the original list, and the second line is the modified list after the map function (in this case **double**) is applied.

There are total 11 markers: 2 in main / 7 in map / 1 in done / 1 in Double. In a nutshell, modify the "### YOUR CODE HERE ###" region.

Fill out each marker

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	