EEE3530-01 **Spring 2024**

Assignment 2: Functions

Due: Sunday, Apr. 28, 2024, 11:59PM

1 Introduction

- The objective of this assignment is to write the *Tower of Hanoi* algorithm using RISC-V instructions in Kite.
- The algorithm requires recursive function calls, so you will have to carefully manipulate function arguments, stack pointers, return addresses, etc.
- To begin the assignment, go to the `kite/` directory that you worked for Assignment 1. Download `functions.sh`, and execute it inside the `kite/` directory to update Kite.

```
$ cd kite/
$ wget https://casl.yonsei.ac.kr/teaching/eee3530/functions.sh
$ chmod +x functions.sh
$ ./functions.sh
```

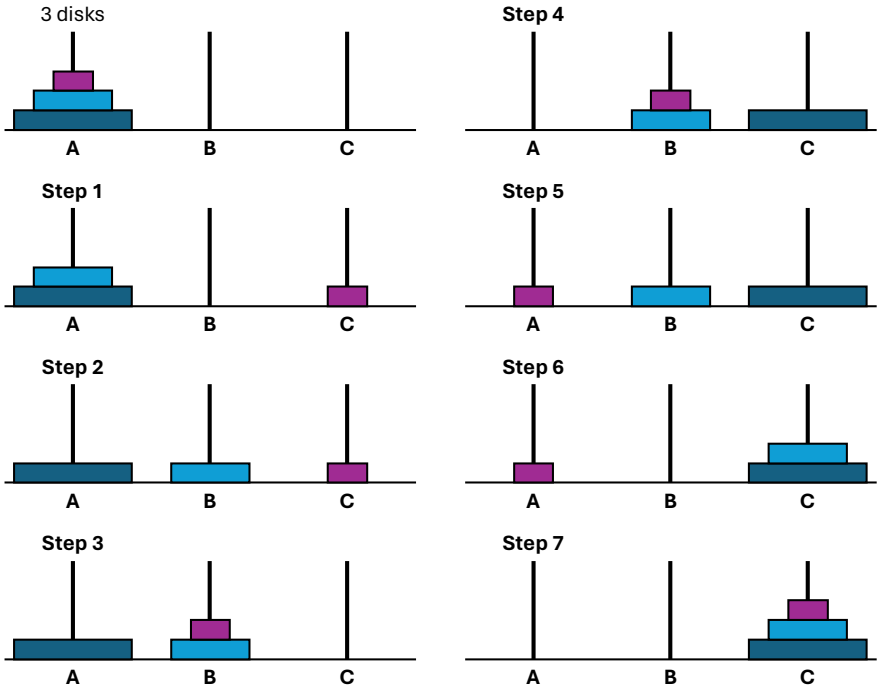

Fig. 1. Steps of solving the Tower of Hanoi to move three disks from rod A to C using the auxiliary rod B.

2 Background

- The Tower of Hanoi is a game consisting of three rods and a number of disks of different diameters, which can slide into any rods.
- The game begins with the disks stacked on one rod in the decreasing size with the smallest disk at the top.
- The objective of the game is to move the entire stack to another rod, obeying the following rules.

  - You can move only one disk at a time.
  - Only the upper disk of a rod can be moved to the top of another rod.
  - No disk can be placed on top of a disk that is smaller than it.
- The minimum number of moves to solve the Tower of Hanoi is 2^n-1, where n is the number of disks.
- Fig. 1 illustrates the steps of solving the Tower of Hanoi for three disks.
- Suppose arrays `A[]`, `B[]`, and `C[]` represent the three rods in a C code.
- Index #0 of each array stores the number of disks in the rod, such as `A[0] = 3` for three disks in rod `A`, and `B[0] = 0` and `C[0] = 0` for empty rods at the beginning.
- For three disks in rod `A`, `A[1] = 59`, `A[2] = 43`, and `A[3] = 31` represent the diameter of the disks in the decreasing size.
- The Tower of Hanoi can be written in C as follows, where three disks in rod `A` are moved to rod `C` using the auxiliary rod `B`.

```
#include <stdio.h>

#define num_disks 3
unsigned num_moves = 0;

unsigned A[4] = {num_disks, 59, 43, 31};
unsigned B[4] = {0, 0, 0, 0};
unsigned C[4] = {0, 0, 0, 0};

// Moving disks from (*from rod) to (*to rod) using the (*aux rod).
void TowerOfHanoi(unsigned n, unsigned *from, unsigned *to, unsigned *aux) {
    if(n == 0) return;

    TowerOfHanoi(n-1, from, aux, to);

    to[++to[0]] = from[from[0]--];
    num_moves++;

    TowerOfHanoi(n-1, aux, to, from);
}

int main(void) {
    TowerOfHanoi(num_disks, A, C, B);
    return 0;
}
```

- In the end, `A[0]` and `B[0]` should be zero, indicating that these rods have no disks. The arrays may have non-zero values at other indices, but they can be disregarded since index #0 tells that the rods are empty.
- The final contents of rod `C` must be {`3, 59, 43, 31`}, meaning that the rod has three disks whose diameters are `59`, `43`, and `31` in the decreasing size.

3 Implementation

- Suppose the global variable, `num_moves`, is represented by the `x9` register. The base addresses of `A[]`, and `B[]`, and `C[]` are stored in the `x18`, `x19`, and `x20` registers, respectively.
- The following shows the initial state of RISC-V registers in Kite for this assignment.

```
x2 = 8192   # Stack pointer
...

x9 = 0      # num_moves = 0
...

x18 = 1000  # Rod A[]
x19 = 2000  # Rod B[]
x20 = 3000  # Rod C[]
...
```

- In this assignment, you must follow the convention of RISC-V registers.
  - The `x1` register carries the return address of a function call.
  - The `x2` and `x8` registers are used as stack and frame pointers, respectively.
  - The `x10`-`x13` registers carry the input argument of the `TowerOfHanoi` function.
  - Ten saved registers (i.e., `x18`-`x27`) must be preserved.
  - All other registers are free to use and do not have to be preserved.
- The following is the initial memory state at addresses 1000, 2000, and 3000. The assignment places five disks at rod `A` whose diameters are `59`, `43`, `31`, `23`, and `17`. Two other rods, `B` and `C`, are empty at the beginning.

```
...

1000 = 5    # 5 disks in rod A
1008 = 59   # Largest disk
1016 = 43
1024 = 31
1032 = 23
1040 = 17   # Smallest disk
...

2000 = 0    # Empty rod B
2008 = 0
2016 = 0
2024 = 0
2032 = 0
2040 = 0
...

3000 = 0    # Empty rod C
3008 = 0
3016 = 0
3024 = 0
3032 = 0
3040 = 0
...
```

- The following shows the `main` function of the C code calling the `TowerOfHanoi(n, A, C, B)` routine. The `main` function is complete, so you do not have to touch this part of the code.
- The assignment requires working on the `TowerOfHanoi` function after the comment line of `Assignment 2: Functions`.

```
main:
addi    x2,     x2,     -16     # Set the stack pointer.
sd      x8,     0(x2)           # Store the frame pointer.
addi    x8,     x2,     16      # Set the frame pointer.
sd      x1,     -8(x8)          # Store the return address.
```

```
ld      x10,    0(x18)          # x10 = n
add     x11,    x18,    x0      # x11 = A
add     x12,    x20,    x0      # x12 = C
add     x13,    x19,    x0      # x13 = B
jal     x1,     TowerOfHanoi    # TowerOfHanoi(n, A, C, B)

ld      x1,     -8(x8)          # Restore the return address.
ld      x8,     0(x2)           # Restore the frame pointer.
addi    x2,     x2,     16      # Restore the stack pointer.
add     x10,    x0,     x0      # return 0 at the end of main
jalr    x0,     0(x1)           # End of the program code


############################
# Assignment 2: Functions #
############################

TowerOfHanoi:
jalr    x0,     0(x1)
```

4 Submission

- After implementing the Tower of Hanoi code, execute the program code using Kite.
- You must see the following output. Register and memory states that are not shown below (i.e., `...`) differ depending on implementations and will not be a part of grading.

```
$ ./kite program_code
...

Register state:
x0 = 0
x1 = 0
x2 = 8192
...

x8 = 0
x9 = 31
x10 = 0
...

x18 = 1000
x19 = 2000
x20 = 3000
x21 = 0
x22 = 0
x23 = 0
x24 = 0
x25 = 0
x26 = 0
x27 = 0
...

Memory state (only accessed addresses):
(1000) = 0
...

(2000) = 0
```

```
...

(3000) = 5
(3008) = 59
(3016) = 43
(3024) = 31
(3032) = 23
(3040) = 17
...
```

- When the assignment is done, execute the `tar.sh` script in the `kite/` directory. It creates a tar file named after your student ID, e.g., `2024143530.tar`. Upload the tar file on LearnUs. Do not rename the file.

```
$ ./tar.sh
$ ls *.tar
2024143530.tar
```

5 Grading Rules

- The following is the general guideline for grading. A 30-point scale will be used for this assignment. The minimum score is zero, and negative scores will not be given. Grading rules are subject to change; a grader may add a few extra rules without notice for a fair evaluation of students' efforts.

  -5 points: The tar file includes redundant tags such as a student name, `hw2`, etc.

  -5 points: The code has insufficient comments. Comments in the skeleton code do not count. You must clearly explain what each part of your code does.

  -5 points: The Tower of Hanoi code is not written in the `program_code` file.

  -15 points: `x1`, `x2`, `x8`, `x9`, `x10`, and `x18`-`x27` registers have different values than what are shown in Section 4.

  -15 points: Memory states at addresses `1000`, `2000`, and `3000`-`3040` have different values than what are shown in Section 4.

  -25 points: The program code does not produce correct results. However, the values are close, and `program_code` shows substantial efforts.

  -30 points: No or late submission.

  Final grade = F: The submitted tar file is copied from someone else. All students involved in the incidents will get Fs for the final grade.

- Your teaching assistant (TA) will grade your assignments. If you think your assignment score is incorrect, discuss your concerns with the TA. Always be courteous when contacting the TA. If you and the TA do not reach an agreement, elevate the case to the instructor for review of your assignment. Refer to the course website for the contact information of the TA and instructor: `https://casl.yonsei.ac.kr/eee3530`

- Arguing for partial credits for no valid reasons will be regarded as a cheating attempt; such a student will lose the assignment scores.