CA2023 Spring HW3

RISC-V Assembly Code

Description

- In this homework, you are going to use Jupiter RISC-V simulator to implement two recursive functions, **the recurrence relation and print out a linked list reversely.**
- After finishing this homework, you will be familiar with the usage of Jupiter RISC-V simulator, register definition, and some basic operations in RV32I Base Integer Instruction Set.

Jupiter

RISC-V Assembler & Runtime Simulator

1.Recurrence Relation

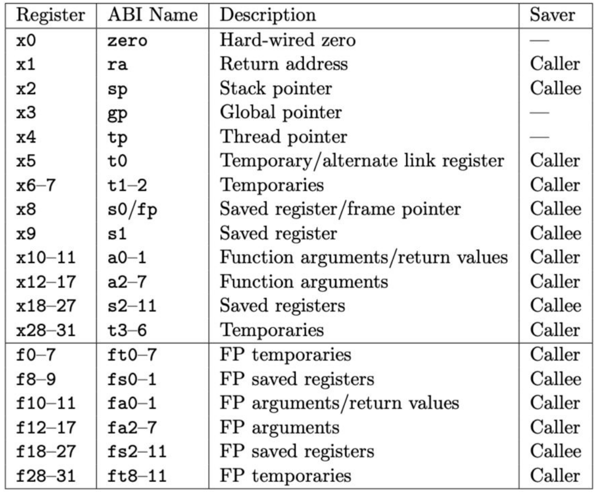
$$T(n) = \begin{cases} 2 \times T(n-1) + T(n-2) & , if\ n \geq 2 \\ 1 & , else\ if\ n = 1 \\ 0 & , else\ if\ n = 0 \end{cases}$$

T(0) = 0, T(1) = 1, T(2) = 2, T(3) = 5, …

1.Recurrence Function

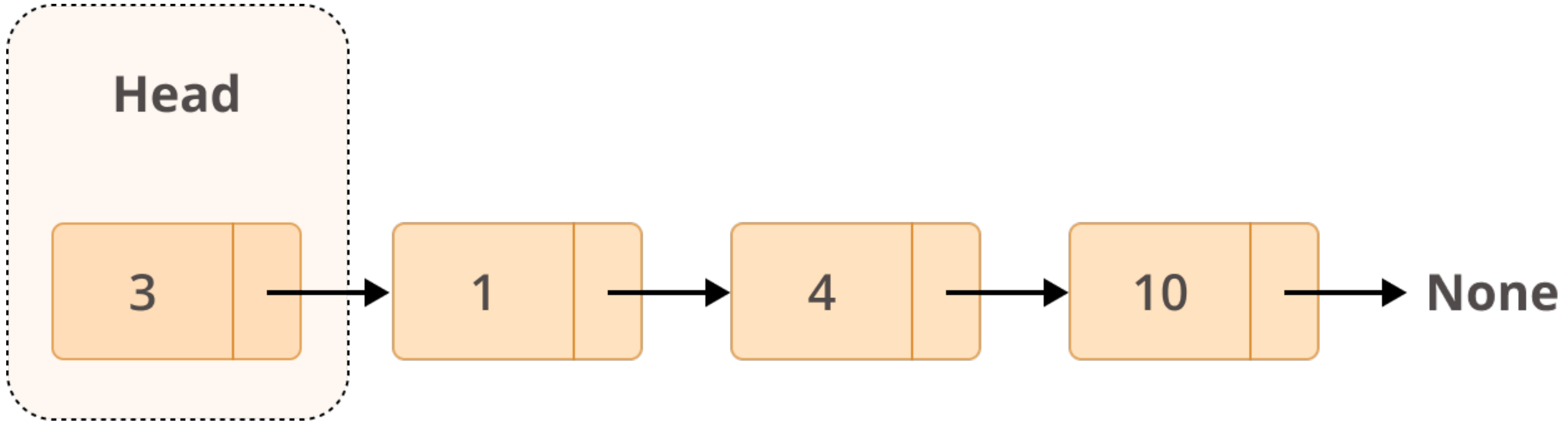
- You’ll need to **implement I/O part** by yourself, checkout Jupiter’s document for more details.

- Follow the RISC-V calling conventions to write the recursive function for the given problems.



Register	ABI Name	Description	Saver
x0	zero	Hard-wired zero	—
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	—
x4	tp	Thread pointer	—
x5	t0	Temporary/alternate link register	Caller
x6–7	t1–2	Temporaries	Caller
x8	s0/fp	Saved register/frame pointer	Callee
x9	s1	Saved register	Callee
x10–11	a0–1	Function arguments/return values	Caller
x12–17	a2–7	Function arguments	Caller
x18–27	s2–11	Saved registers	Callee
x28–31	t3–6	Temporaries	Caller
f0–7	ft0–7	FP temporaries	Caller
f8–9	fs0–1	FP saved registers	Callee
f10–11	fa0–1	FP arguments/return values	Caller
f12–17	fa2–7	FP arguments	Caller
f18–27	fs2–11	FP saved registers	Callee
f28–31	ft8–11	FP temporaries	Caller

2. Print Out A Linked List Reversely

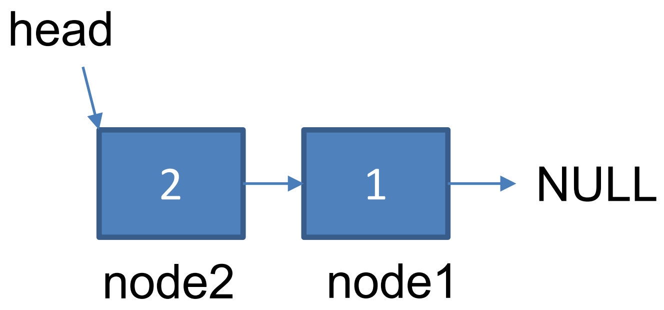


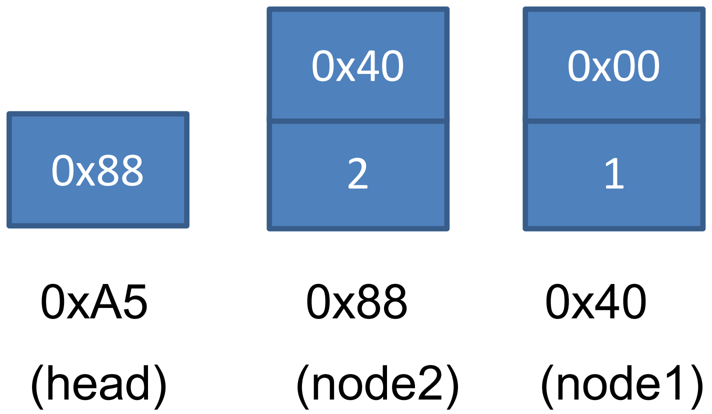
Output : 10 4 1 3

2. Print Out A Linked List Reversely

- We will provide sample code about this function, so you don’t need to do I/O operations in this case.

2. Print Out A Linked List Reversely





Grading Policy

Total 100%, Recurrence relation 60%, Print out linked list 40%

- Recurrence relation has 6 test cases, 10 points per test case.
- Print out linked list has 4 test cases, 10 points per test case.
- Time limit: 60 seconds per test case.

We will judge the correctness of your program using following commands:

```
$ jupiter [student_id]_recurrence.s < input_file

$ jupiter [student_id]_linkedlist.s < input_file
```

Grading Policy

- 10 points off per day for late submission.
- You will get 0 point for plagiarism.
- You will get zero point if we find out that you solve the problem without using recursion.

Submission

- Due date: 04/04 23:59 (Tuesday)
- You are required to submit **.zip** file to NTU Cool
- File structure for the .zip file (case-sensitive):

```
[student_id (lower-cased)].zip
        /[student_id]/ <-- folder
                [student_id]_recurrence.s <-- file
                [student_id]_linkedlist.s <-- file
```

- For example, if your student id is b12345678, your zip file should have following structure:

```
b12345678.zip
        /b12345678/
                b12345678_recurrence.s
                b12345678_linkedlist.s
```

Reference

- Lecture slides
- Jupiter RISC-V simulator
  https://github.com/andrescv/Jupiter
- Jupiter RISC-V simulator docs
  https://jupitersim.gitbook.io/jupiter/
- RISC-V Instruction Set Manual
  https://github.com/riscv/riscv-isa-manual
  https://riscv.org/technical/specifications