CA2022 Fall HW2

RISC-V Assembly Code

Description

- In this homework, you are going to use Jupiter RISC-V simulator to implement the Fibonacci sequence and the inorder traversal of a complete binary tree.
- After finishing this homework, you will be familiar with Jupiter basic I/O, RISC-V calling convention, and the implementation of array and pointer in assembly level.

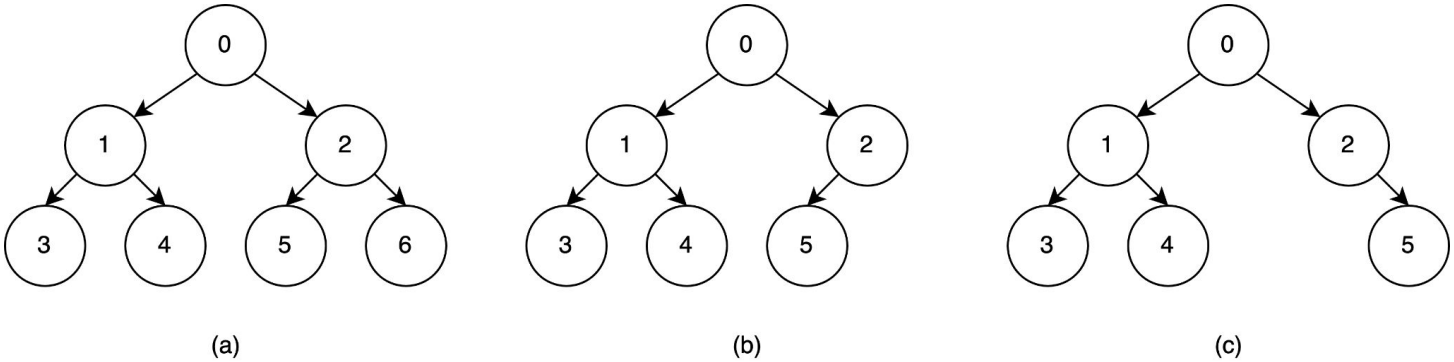
Fibonacci sequence

$$F_0 = 0, \quad F_1 = 1,$$

$$F_n = F_{n-1} + F_{n-2}$$

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

Inorder traversal of a complete binary tree

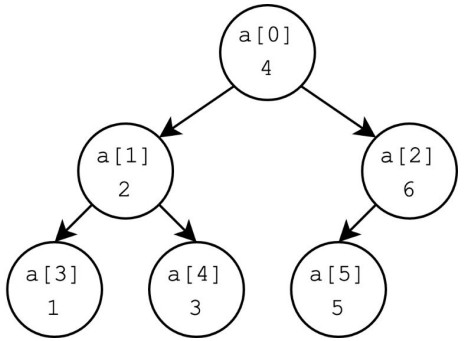


A binary tree in which every level, except possibly the last, is completely filled, and all nodes in the last level are as far left as possible.

Tree (a) and tree (b) are complete binary trees, while tree (c) is not.

Inorder traversal of a complete binary tree

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
4	2	6	1	3	5



Output: `1 2 3 4 5 6`

I/O - Fibonacci sequence

- Input file contains only 1 line: the value of n
- Your program should output the correct result of the n^{th} item of given sequence.

Fibonacci: 0, 1, 1, 2, 3, 5, 8, 13, ...

```
❯ jupiter hw3_fibonacci.s
3
2

Jupiter: exit(0)
```

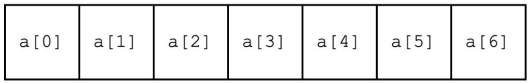
```
❯ jupiter hw3_fibonacci.s
7
13

Jupiter: exit(0)
```

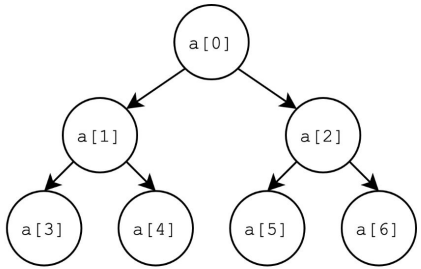
Fibonacci: $0 \leq n \leq 15$

I/O - Inorder traversal of a complete binary tree

- First line: n the number of nodes of the complete binary tree.
- This is followed by n lines, each corresponding to a node value of the complete binary tree.
  - These n node values are stored in an array **`a[0]...a[n-1]`** in sequence to represent a complete binary tree.
  - We define that given a parent node **`a[i]`**, the left child node can be accessed using **`a[i*2+1]`** and the right child node can be accessed using **`a[i*2+2]`**.



(a)



(b)

TO-DO

- 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

Tips - Inorder traversal of a complete binary tree

Sbrk

Stores a pointer to a block of memory containing `n` additional bytes in register `a0`. This pointer is word aligned.

Example

```
.globl __start

.text

__start:
  li a0, 9   # ecall code
  li a1, 100 # number of bytes
  ecall
```

Ecall Code

- `a0` = 9

Arguments

- `a1` = number of bytes to reserve

Jupiter has an environment call called “`sbrk`” that can allocate a block of memory on heap.

You can first read the tree to an array on heap and pass the pointer of the array to the recursive function.

Grading Policy

- Total 100%, Fibonacci sequence 60%, Inorder traversal of a complete binary tree 40%
  - Fibonacci sequence has 6 test cases, 10 points per test case.
  - Inorder traversal of a complete binary tree has 4 test cases, 10 points per test case.
  - Time limit: 60 seconds per test case.
    - Time limit is only used for auto judgement, and it shouldn't be the part of grading. If you can output correct answers but can't meet the timing requirement, please contact TA using email.
    - However, it's very likely that your program has some bugs if it's stuck for 1 minutes. (infinite loop, stack become a mess... etc)
- We will judge the correctness of your program using following commands:

```
$ jupiter [student_id]_hw2_fibonacci.s < input_file

$ jupiter [student_id]_hw2_inorder.s < input_file
```

Grading Policy (cont.)

- 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.
- You will get zero point if we find out that you solve the problem by storing all possible answers and print it out directly.

Submission

- Due date: 10/18 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]_hw2_fibonacci.s <-- file
          [student_id]_hw2_inorder.s <-- file
```

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

```
b12345678.zip
     /b12345678/
          b12345678_hw2_fibonacci.s
          b12345678_hw2_inorder.s
```

Reference

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