CA 2024 Spring HW2

RISC-V Assembly Code

Agenda

- Assignment Introduction
  - Recurrence Relation
  - Linked-List Procedure
- Grading Policy
- Submission

Assignment Introduction

- In this homework, you are going to use Jupiter RISC-V simulator to complete two tasks, **Recurrence Relation and Linked-List Procedure**.

Recurrence Relation

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

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

Recurrence Relation

- 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	—
x5–7	t0–2	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

Recurrence Relation I/O

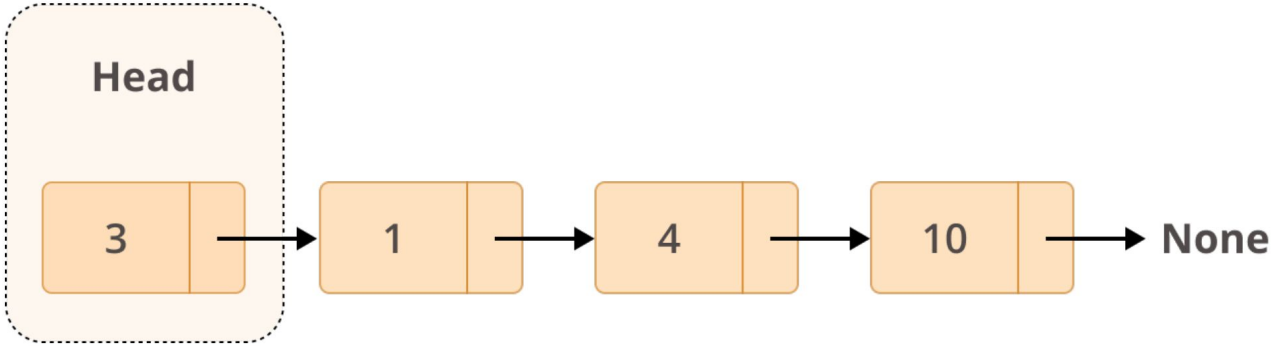
- Input file contains one number n. ($0 \leq n \leq 15$)
- Your program should only output the correct result.

```
> ./jupiter test.s
0
0

Jupiter: exit(0)
```

```
> ./jupiter test.s
5
29

Jupiter: exit(0)
```

Linked-List Procedure

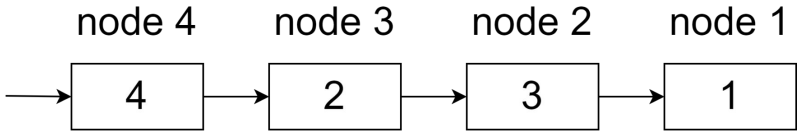
- Linked-List Structure

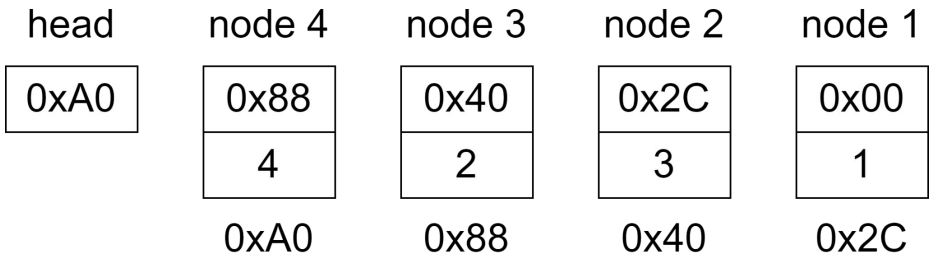


Linked-List Procedure

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

example input:
4
1 3 2 4





Push the new element to the front of linked-list.

Linked-List Procedure

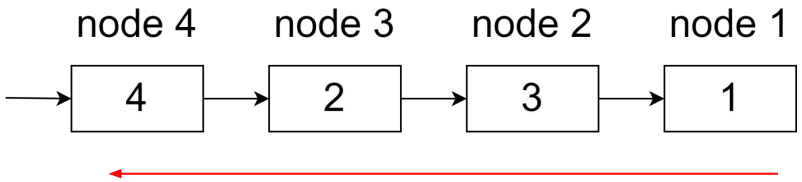
- Tasks
  - Reversely Print
  - Sort in ascending order

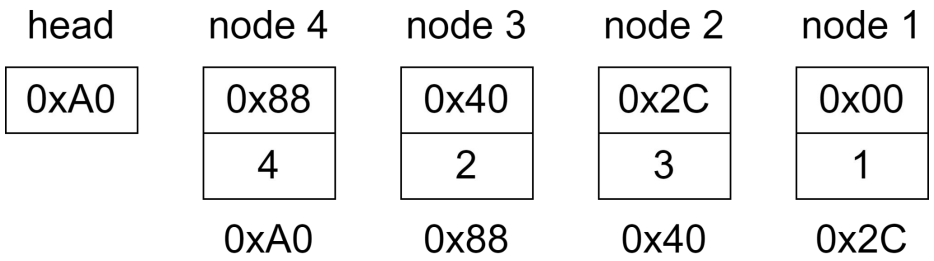
```
print_list:
#############################################
#  TODO: Print out the linked list          #
#                                           #
#############################################
        ret

sort_list:
#############################################
#  TODO: Sort the linked list               #
#                                           #
#############################################
        ret
```

Linked-List Procedure Reversely Print

- Traverse and print out all elements of the linked list reversely.

example input:
4
1 3 2 4

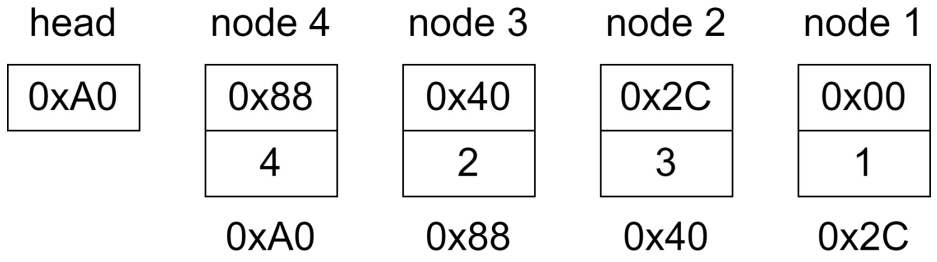




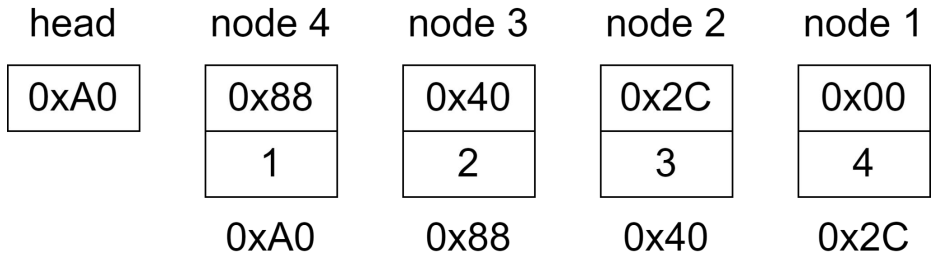
Linked-List Sort

- Sort the Linked-List in ascending order.

example input:
4
1 3 2 4



Before Sort



After Sort

Linked-List Sort I/O

- Input ($0 \le n \le 100$, $0 \le a[n] \le 2{,}147{,}483{,}647$)
  - The first line of input contains a single integer n, the number of nodes of the linked list.
  - The following n lines, each corresponding to a value push front to the linked list.
- Output
  - The first line prints out the linked list reversely.
  - The second line prints out the linked list reversely after sorting in ascending order.

```
LBB2_3:
        mv      s0, a0
        call    print_list
        call    print_newline
        mv      a0, s0
        call    sort_list
        mv      a0, s0
        call    print_list
```

s0 is a callee-handled register.

```
❯ ./jupiter test.s
0
Empty!

Jupiter: exit(0)
```

Have been handled.

```
❯ ./jupiter test.s
5
1
3
2
4
5
1 3 2 4 5
5 4 3 2 1

Jupiter: exit(0)
```

Grading Policy

- Total 100%, Recurrence relation 40%, linked-list procedure 60%
- Time limit: 60 seconds per test case.
- We will judge your program by running the following command:

```
$ 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. (Recurrence relation and Linked-List Reverse Print Out)

Submission

- Due date: 10/16 23:59 (Wednesday)
- 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**