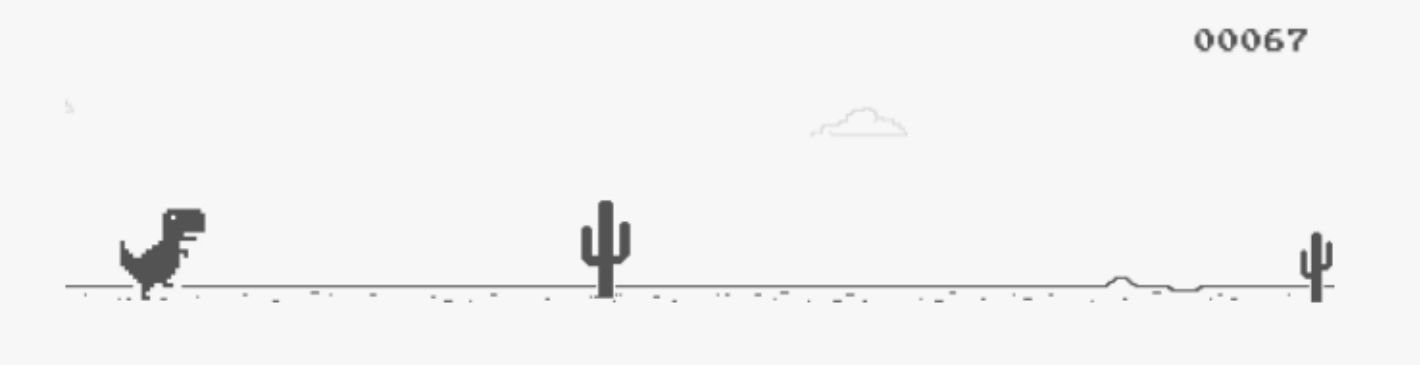
Project 4: Longan Nano Chrome Dino Runner Game

Computer Architecture I | ShanghaiTech University
Project 3 Project 4



Goal

In this project, we hope you can use RISC-V programming to implement a game using Longan Nano board.

You have two options:

1. Implement a Chrome Dino runner game, which you must have played before when there is no Internet connection on google chrome. You can play it here.
2. You can also implement your own game if you want. You are allowed to implement ANY game EXCEPT those from project4 in previous years.

   Games that are NOT allowed:

   - 2020: Ping Pong.
   - 2021: Retro Snake.

   Of course your game should not be too simple, so we set some rules. Please check them in section *Requirements - Game*.

Note that whatever game you choose, we will only provide the basic framework as in the Lab 11 starter - we expect that you should be capable of doing a project from scratch. Of course, the library functions are attached.

Getting started

Make sure you read through the entire webpage before starting the project.

You will be using gitlab to collaborate with your group partner. Autolab will use the files from gitlab. Make sure that you have access to gitlab. In the group CS110_Projects you should have access to your project 4 repository. Also, in the group CS110, you should have access to the p4_framework.

We also provide a demo video here.

0:00 / 0:30

Hardware and Software setup

Refer to Lab 11 for details. The following is a quick navigator:

- Software setup
- Hardware setup

Other references from Longan Nano.

Demo project - bad apple

Document.

Requirements

Coding

We expect you to program something in RISC-V on a real machine - So we require that:

- You implement at least 3 RISC-V functions, and the TOTAL LoC is larger than 300 (comments excluded). Each function should be put into a dedicated file.
- For tidiness, please put all the assembly functions (.S files) in `src/assembly`. No other files or directories should appear in that folder. (So there are at least 3 `.S` file in `src/assembly`)
- For assembly code: You need to have comments not less than 40% of the total lines of RISC-V code you wrote. A comment is defined by a sentence followed by `#`. Comments have to be meaningful and in English.
- The project can be successfully compiled via PlatformIO.

You need to show your game to your TA during checking up.

Game

For Chrome Dino Runner Game, it should:

- have a simple start menu.
- have a live score board. The score increases as the T-Rex running.

- control the T-Rex to perform two different actions to cross obstacles. (for example, jump and lower its head)
- contains two different types of obstacles.
    - In original game, obstacles can be various cactus and pterosaur. The cactus have different size, and the pterosaur may fly high in the air or close to the ground. In this project, you don't need to handle these problems. However if you want to make the game looks nicer, just do it!

For any other game, it should:

- have a simple main menu and a score board.
- use two buttons to perform two (or more) different operation.
- Contains at least one moving object. We don't want your 'game' to be just some still images.

Grading Policy

If you did not pass the check on Autolab, you will get zero score (Comment in assembly code, correct compilation, at least 3 RISC-V functions).

You need to show your game to your TA. If your game can run properly, and meet the requirements above, you will get 80% of score. If your game is a decent, nice-looking one, you can get the rest 20% scores.

Obtain your files

1. Clone your Project 4 repository from GitLab.
2. In the repository add a remote repo that contains the framework files: `git remote add framework https://autolab.sist.shanghaitech.edu.cn/gitlab/cs110_22s/p4-framework.git`
3. Go and fetch the files: `git fetch framework`
4. Now merge those files with your master branch: `git merge framework/master`
5. The rest of the git commands work as usual.

Files

The framework contains the following files:

```
.
├── LICENSE
├── Makefile
├── README.md
├── dfu-util
├── include
│   ├── README
│   ├── fatfs
│   │   ├── diskio.h
│   │   ├── ff.h
│   │   ├── ffconf.h
│   │   └── tf_card.h
│   ├── gd32v_pjt_include.h
│   ├── gd32vf103_libopt.h
│   ├── lcd
│   │   ├── bmp.h
│   │   ├── lcd.h
│   │   └── oledfont.h
│   ├── systick.h
```

```
│   └── utils.h
├── platformio.ini
└── src
    ├── assembly
    │   └── example.S
    ├── fatfs
    │   ├── 00history.txt
    │   ├── 00readme.txt
    │   ├── ff.c
    │   ├── ffsystem.c
    │   ├── ffunicode.c
    │   └── tf_card.c
    ├── lcd
    │   └── lcd.c
    ├── main.c
    ├── systick.c
    └── utils.c
```

Please read `src/utils.c`, `src/assembly/example.S`, `src/systick.c` and `src/lcd/lcd.c` carefully. They contain helper functions and/or infomation that may not be mentioned in this webpage.

Submit to Autolab

Similar to previous projects, upload your `autolab.txt` to autolab.

Schwertfeger, Sören <soerensch *AT* shanghaitech.edu.cn>
Chundong Wang <wangchd *AT* shanghaitech.edu.cn>
Bouyu Gou <gouby *AT* shanghaitech.edu.cn>
Last modified: 2022-05-19