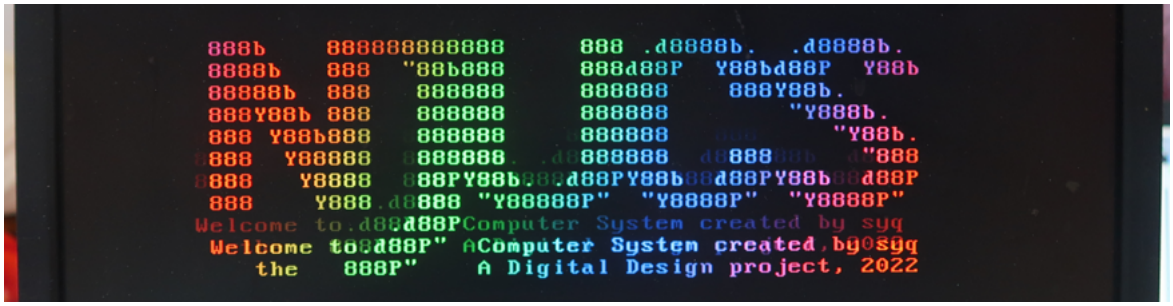
第十五次实验报告：计算机系统

姓名：邵钰乾　　**学号：**211240036

December 29, 2022



1　实验目的

1. 掌握在计算机系统中硬件接口的设计方法。
2. 掌握计算机系统中软件系统的调试方法。
3. 掌握基于 RISC-V 计算机系统应用的开发方法。
4. 学习计算机系统设计的基本方法。

2　小组成员

由于学期初的队友已经在期中之前退课跑路，所以本学期的实验全部由我独自完成。虽然期间经历了很多辛苦，不过最后的实验结果还是比较令人满意的，学到了很多。

3　项目介绍

本项目采用 10MHz 的单周期 CPU（在实际测试中最高支持过 20MHz），搭载 vga 显示器、键盘、时钟三个外设，在简易计算机系统中实现了完善的命令行终端，除了验收要求内的 hello、fib、time 指令之外，还实现了 help 指令、echo 指令、logo 动画展示以及贪吃蛇小游戏。展示视频南大网盘链接：https://box.nju.edu.cn/f/e0d80365e2c54988a31e/

4 硬件部分：单周期 CPU 和外设接口

1. **综合危机：时间都去哪了？**

```
shaoy@MSI MINGW32 /d/Projects/Vivado_Projects (master)
$ git lg
* f5b38a3 FINISHED (12 hours ago) <MeteorCollector>
* d632867 last debug, hopefully (13 hours ago) <MeteorCollector>
* 3bae9aa still fixing.. terminal apps does not clear screen as expected (14 hours ago) <MeteorCollector>
* a2b3e2c fixed several software bugs (16 hours ago) <MeteorCollector>
* f3554cf now we saved timer (18 hours ago) <MeteorCollector>
* d853de4 software compiled successfully (20 hours ago) <MeteorCollector>
* d9585ac still troubled by undefined references (22 hours ago) <MeteorCollector>
* 4bb7e1d hardware ok (2 days ago) <MeteorCollector>
* c0e873a vga no bug already, hopefully. (2 days ago) <MeteorCollector>
* 4718717 end of work 12.27, try asm tomorrow. (2 days ago) <MeteorCollector>
* e3fcb7e not all due to clk freq. (3 days ago) <MeteorCollector>
* 5bb63f1 might be a clock freq problem. (3 days ago) <MeteorCollector>
* 2d797c4 stupid bugs fixed, now the harsh part (4 days ago) <MeteorCollector>
* 4723533 begins debuging of hardware session (4 days ago) <MeteorCollector>
* fb16d53 end of work 12.24, vga part unfinished (5 days ago) <MeteorCollector>
* e005269 sw passed (5 days ago) <MeteorCollector>
* 5d50e0f in the middle of construction of cpu, preparing for sw test (5 days ago) <MeteorCollector>
* 1ca2061 problem figured out. vivado sucks (6 days ago) <MeteorCollector>
* 14b34d2 tyring ip core (6 days ago) <MeteorCollector>
* adfd459 packed a cpu into ip core (6 days ago) <MeteorCollector>
* 5718847 updated upper modules, but unsynsisable (7 days ago) <MeteorCollector>
* 7c196d7 cpu impl complete (7 days ago) <MeteorCollector>
```

图 1：本次项目的 git 记录。

在上一个实验中，我已经完成了一个单周期 CPU，但是由于没有采用 IP 核实现存储器，并且某些部分的实现并不尽如人意，所以综合实践旷日持久（最高可达 2-4 小时），这是完全不可接受的。工欲善其事必先利其器，我做的第一件事是完善 cpu 的实现。

```
WARNING: [Synth 8-327] inferring latch for variable 'q_reg' [D:/Projects/Vivado_Projects/Computer_System/testdmem.v:31]
WARNING: [Synth 8-327] inferring latch for variable 'tempout_reg' [D:/Projects/Vivado_Projects/Computer_System/testdmem.v:29]
---------------------------------------------------------------------------------
Finished RTL Optimization Phase 2 : Time (s): cpu = 02:53:54 ; elapsed = 03:27:20 . Memory (MB): peak = 4657.152 ; gain = 3568.352
---------------------------------------------------------------------------------
```

图 2：早期阶段的一次综合，需要花费三个多小时

经查，发现导致综合实践过长的罪魁祸首是手动实现的指令存储器。将其实现换为 ip 核后，发现程序执行错误。进行大量 debug 工作和资料查阅后，发现 vivado 的 ram 会默认将输出信息在寄存器中锁存一个时钟周期，导致错误。将这一默认设定取消掉之后，存储器实现成功。存储器替换后，综合时间大大缩短，时长变得可以接受。

Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	DSP	Start	Elapsed
constrs_1	**synth_design Complete!**								**19776**	**1282**	**0.0**	**0**	**0**	**12/29/22, 9:56 AM**	**00:05:19**
constrs_1	write_bitstream Complete!	NA	NA	NA	NA	NA	748.354	0	20094	1291	88.5	0	0	12/29/22, 10:02 AM	00:14:53

图 3：改进之后，一次综合其实只需要几分钟。开发过程中，从综合到比特流生成的时长大概稳定在 40 分钟左右

2. **外设接口的实现**

新 cpu 通过 testbench 之后，需要考虑的就是顶层模块的设计和外设接口的实现。

我实现的计算机外设基本沿用了实验指导中的约定，但是在键盘部分有所改动，并新增了时钟接口。

(a) VGA：沿用了字符显示实现的显存约定，只储存 ASCII 码，虽然没有实现图形化界面，但是基本可以完成软件需求。显存以 0x002 开头，cpu 只写不读，vga 只读不写。

(b) 键盘：键盘缓冲区以 0x003 开头。与实验指导不同的是，键盘只写不读，cpu 既读又写。这是因为我仅仅在硬件层次维护了一个写指针，使得键盘每次在新的空白部分写入新按下的按键；然而在 cpu 方面，计算机系统共提供了两个函数：wait_keyboard 和 get_keyboard：其中 wait_keyboard 会在软件中维护一个读指针，不停扫描键盘缓冲区，直至缓冲区出现一个键码，则返回该键码，将键盘缓冲区对应地址置 0；get_keyboard 函数则每次将键盘缓冲区扫描一遍，若存在键码则返回，若不存在则返回空。

(c) 时钟：时钟接口存储地址以 0x004 开头。使用分频器实现，以毫秒为单位记录从开机开始过去的时间。时钟只写，cpu 只读。

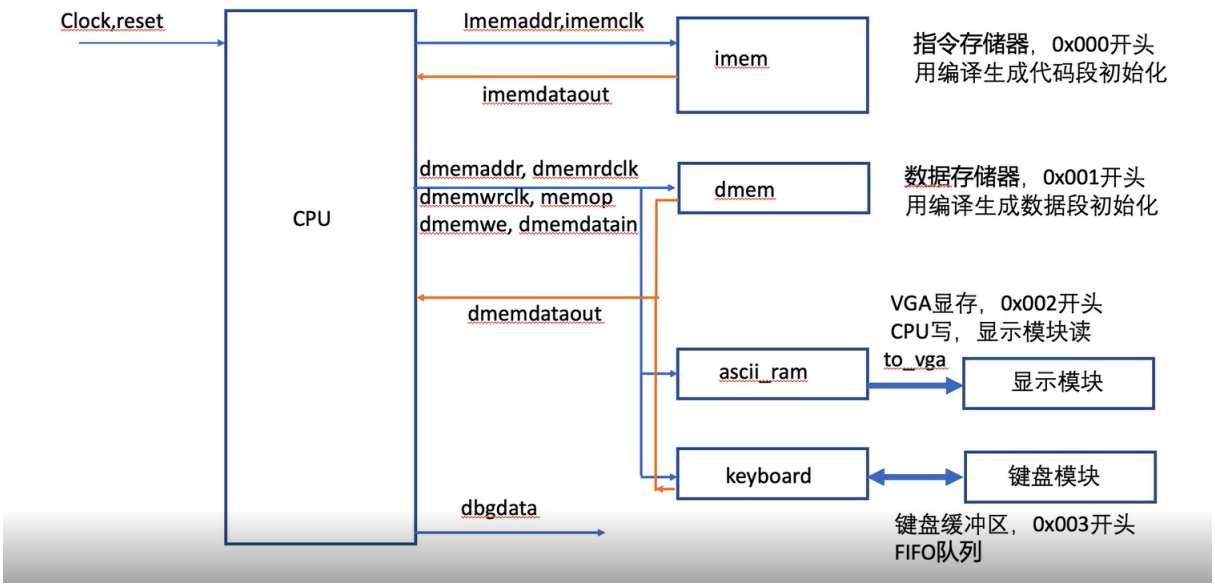
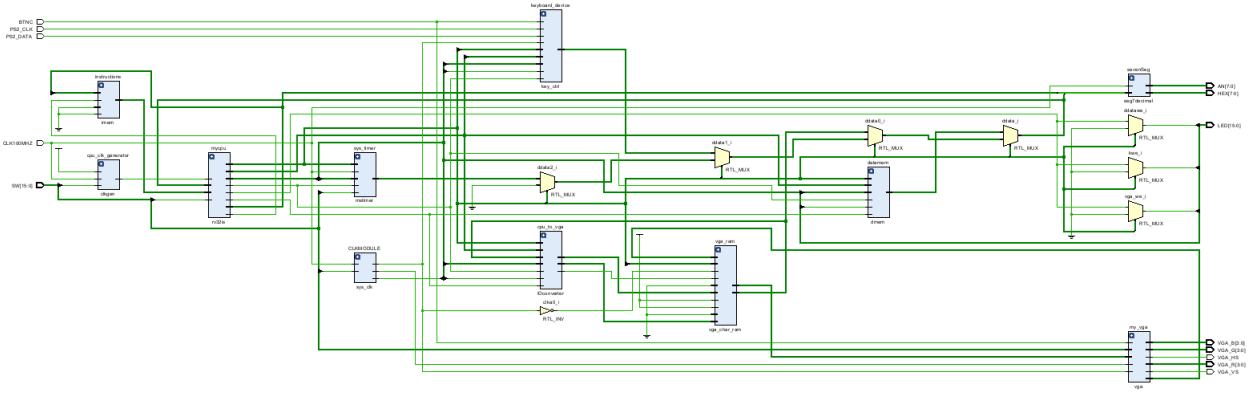
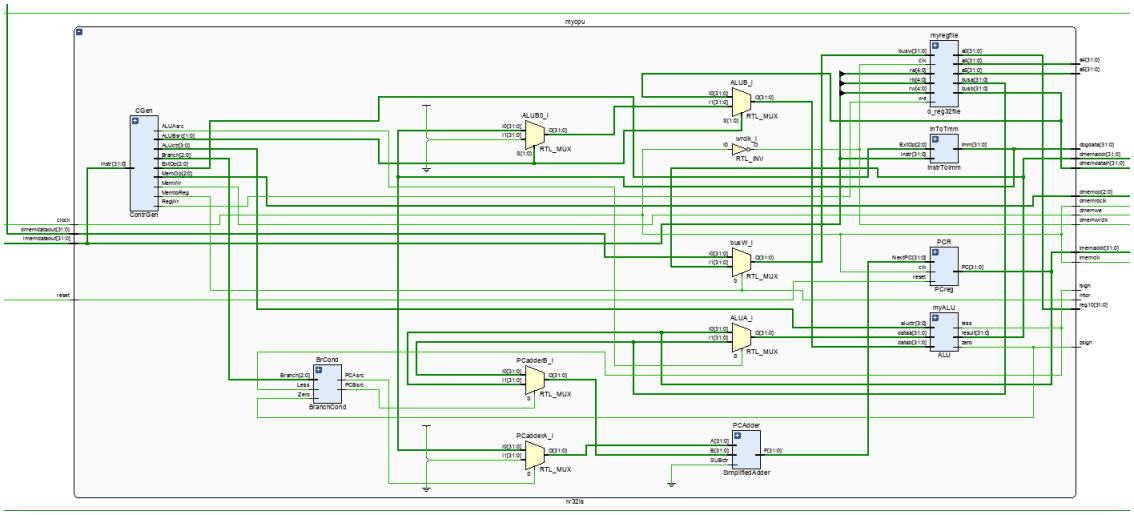


图 4：实验指导中对外设接口地址映射的规定

硬件部分的 rtl 线路图如下：



cpu 部分局部放大：



忽略 tcl 输出导致的惨剧：仿真测试

在实现硬件之后，我将一个简易程序写入指令存储器。发现程序并不如预期一般运行。于是我将机器的时

钟接到仿真时钟上，对照 dump 文件中的汇编代码进行仿真测试。

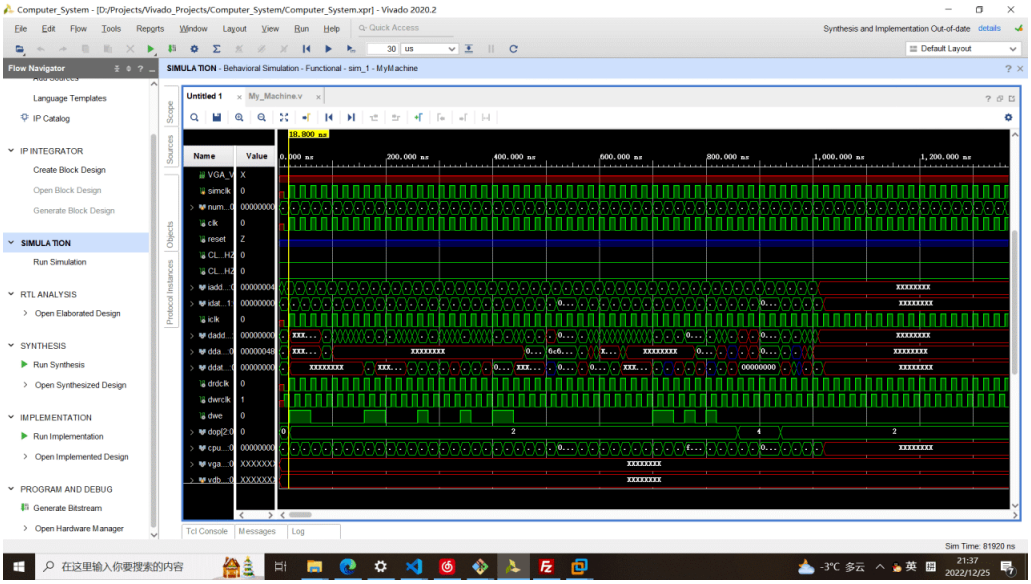


图 5: vivado 仿真测试画面

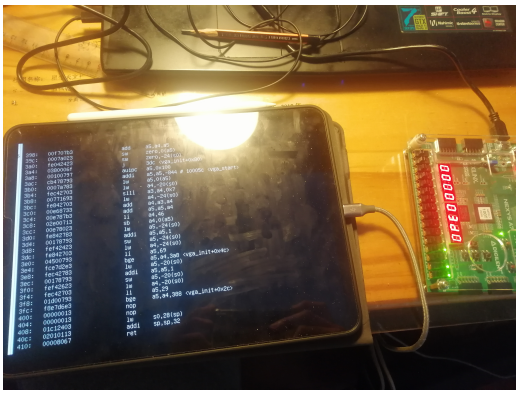
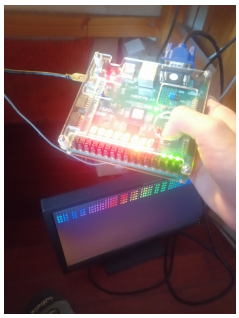


图 6: dump 文件，手动模拟软件运行并与 vivado 仿真测试中的执行情况进行比对

经查，发现 ddata 总线工作异常。对照 verilog 代码，发现这条线我并没有声明，程序自动生成了一条 1bit 宽度的线缆，而不是 32bit，导致工作异常。其实这个错误在仿真时的报错中是可以看到的，但是当时我并没有在海量的 tcl console 输出中注意到这条信息，浪费了时间。经过改动，仿真测试正常。

3. **最后的单步测试：人形时钟**

但是通过仿真测试之后，上机测试仍然异常，程序好像没有执行一样。于是我决定使用单步执行的 debug 方式，将时钟接到 BTNC 按钮上，由我手动控制时钟信号。



手动单步执行发现程序其实被正确执行了，我这才意识到是时钟频率的问题。我将机器主频从 50MHz 降至 10MHz，在等待比特流文件生成的过程中，我继续按了几万下按钮确认了后续的程序执行状况依旧正常。降频之后，程序执行正常。

确认了硬件部分的正确性之后，接下来要进行的就是软件部分的开发了。

5 软件部分：简单的计算机系统

1. **硬件的性能以及软件的目标**

   目前我们已经拥有了一个 10MHz 单周期 cpu，一个键盘，一个只能显示 ASCII 字符的显示器和一个毫秒单位计时的时钟。虽然配置比较寒碜，但是足够我们运行一些比较有意思的程序了。

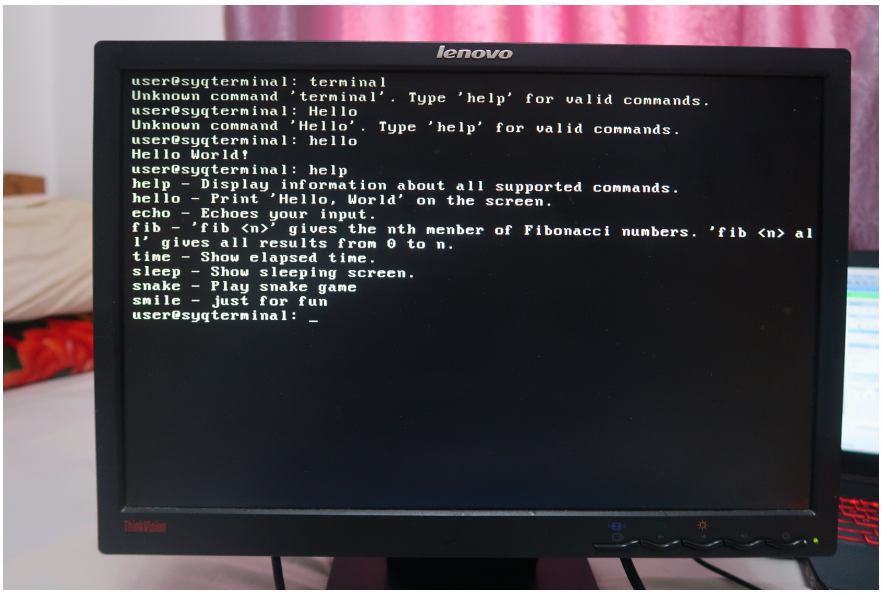
   在实验的验收要求中，要求我们的计算机有一个简易的终端，可以解析"hello" 输出"Hello World"，解析"fib n" 输出斐波那契数列第 n 项，解析"time" 输出时间。在我的设想里，我希望我的设备可以播放一段字符画动画并且运行游戏。在动画方面，我制作了一个写有 NJUCS 和欢迎信息的 logo 在屏幕上移动，这个方案不需要存储很多帧的动画画面（当然如果要存的话，现有存储器也是完全够的，但是仅仅是重复的搬砖工作，意义不大），只需要计算 logo 在每帧的坐标就够了。在游戏方面，我的原计划是移植问题求解课程大一下学期开发的 "炸弹人" 游戏，但是原项目是用 C++ 开发，并且机制较为复杂，所以最后决定用 C 写一个简易的贪吃蛇游戏。

   为了顺利进行软件开发，除了编写了编辑显存使用的 init_vga() putch() putstr() roll_up()(滚屏) refresh_screen()(将写在显存缓冲区的数据写入缓存) show_cursor()(显示光标) 函数、访问键盘输入使用的 get_keyboard() wait_keyboard() wait_line()(在终端中获取一行输入) 函数之外，我还移植了先前在计算机系统基础课程 pa3 中实现的 stdlib 库和 string 库函数，方便字符串操作、输入输出处理和随机数的生成。至于时钟数据的访问，声明一个指针访问对应地址就可以了。

   以下是软件部分实现的所有功能的相应介绍。

2. **终端**

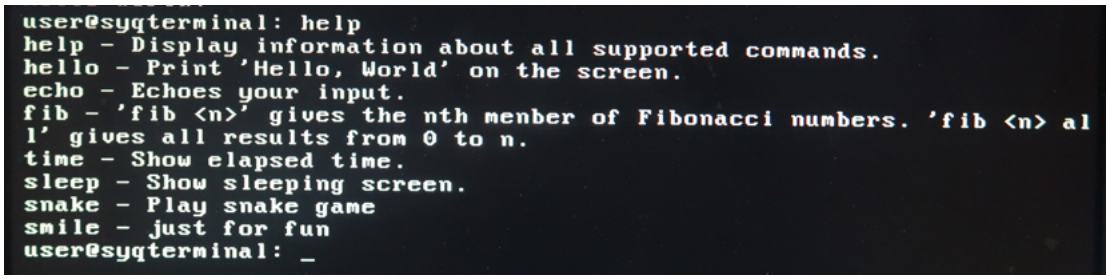
   一个中规中矩的终端。



   开机以后会直接进入计算机系统终端。该终端实现了字符输入、退格删除、回车换行、光标闪烁，若一屏写满，将向上滚屏。

3. **help**

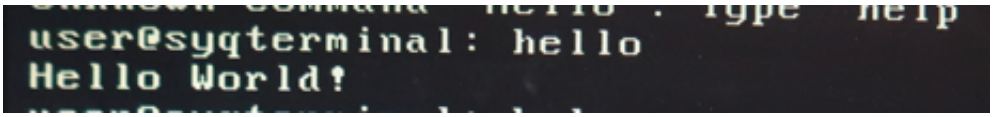
   打印指令具体行为。



如果没有参数传入，将输出支持的所有指令的帮助信息。如果传入某指令名称，输出对应指令的帮助信息。

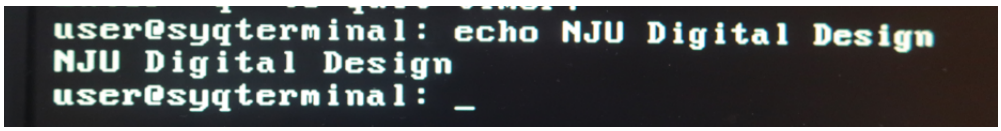
4. **hello**

   输出"Hello World!"



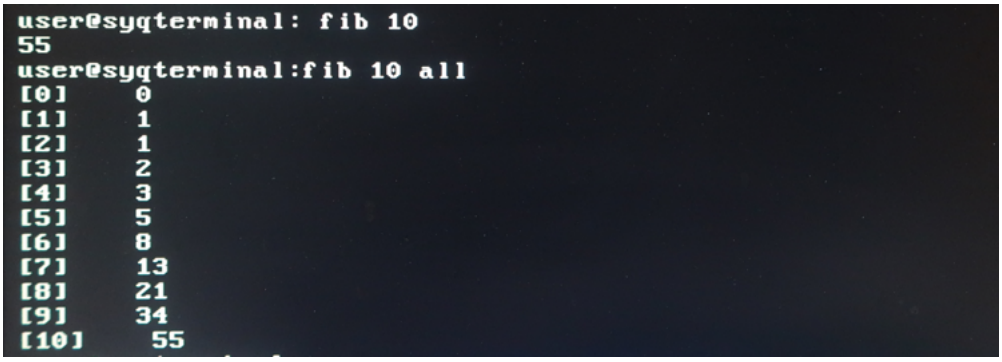
5. **echo**

   返回传入的信息。



6. **fib**

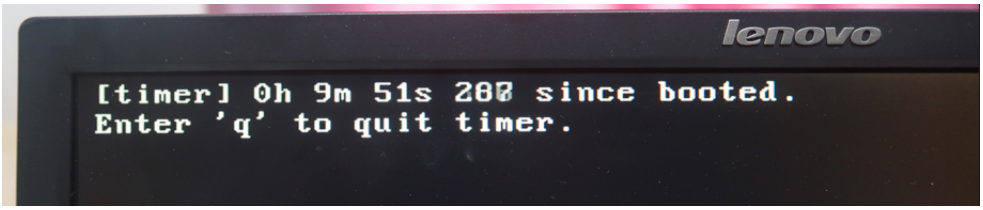
   fib n 返回斐波那契数列第 n 项；如果后续加入参数 all，将打印出第 0 项至第 n 项所有数列成员。



值得注意的是，如果传入大于 47 的数，将报错：输出数据超过 int 表示范围；如果传入小于 0 的数，将返回 0；

7. **time**

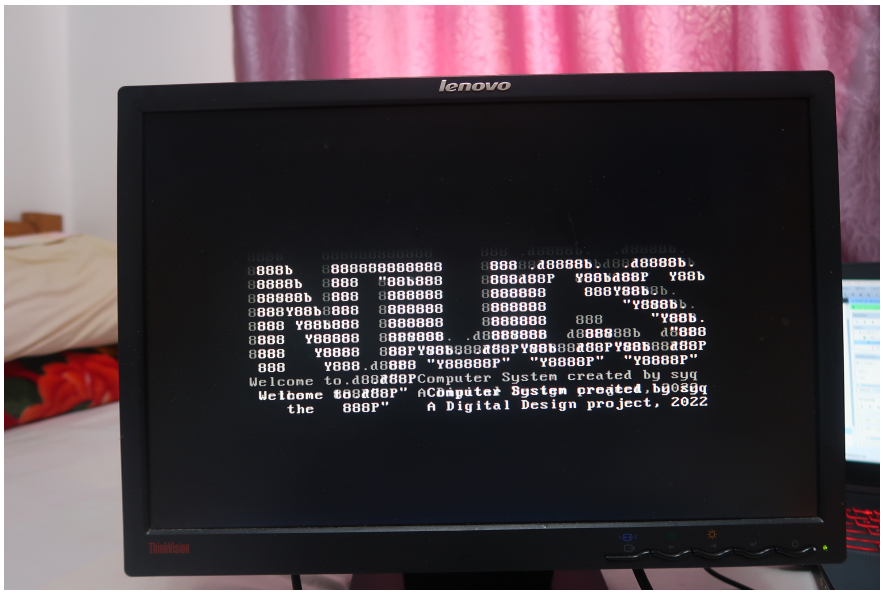
   清屏，并在屏幕左上角显示自机器启动以来过去的时间。



按 q 键退出时钟，返回终端。

8. **sleep**

播放锁屏动画。



字符画组成的 logo 将在屏幕上斜向移动，遇边缘反弹。按下任意键结束动画，返回终端。如果打开 rgb 模式，效果如下：

9. **snake**

贪吃蛇游戏。进入后，首先要求玩家键入一个数字，选择难度。最简单的难度 1 中，蛇每秒移动 1 个单位；最困难的难度 5 中，蛇每秒移动 20 个单位。蛇用 wasd 控制方向，随时按 q 退出。游戏规则不再赘述。

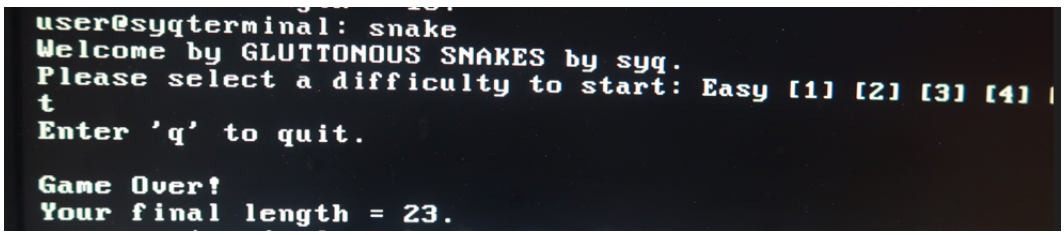


图 7: 游戏开始前的选择提醒与退出后的信息

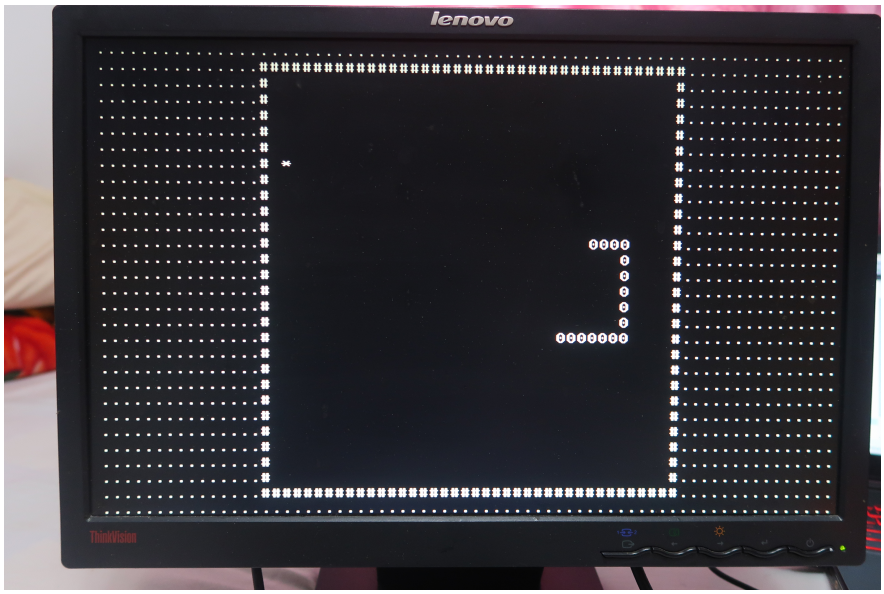


图 8: 贪吃蛇游戏画面

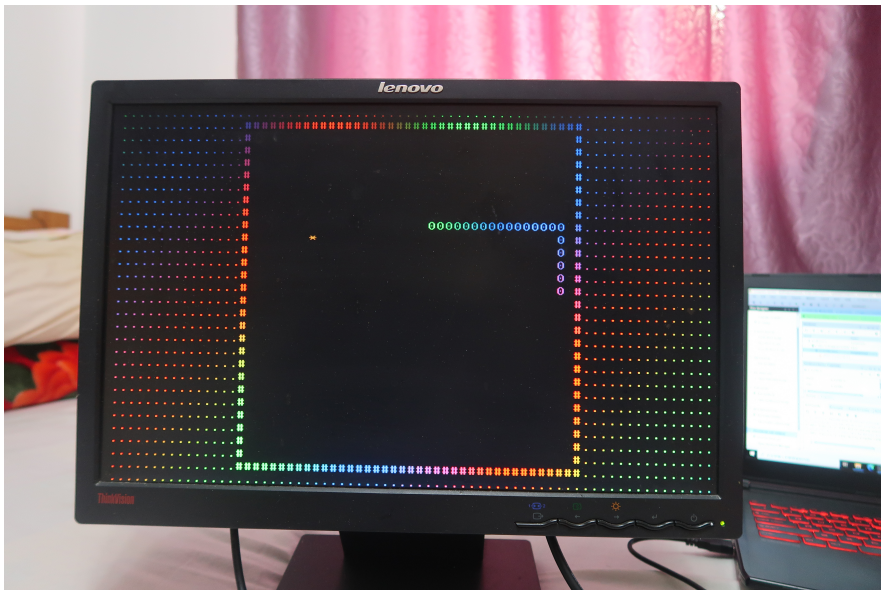


图 9: 贪吃蛇, 但是 rgb

6 思考题

题目 1
分析如何在系统使用 AXI 总线接口的方法。

解答:
只要约定好存储器映射就可以了。在硬件上做好数据通路的连接，再在软件上约定对应指针的地址。

题目 2
分析如何在系统中使用高速缓冲存储器。

解答:
之前在 ics 的 lab 中已经实现过 cache，可惜是使用 C 完成的软件模拟实现。硬件实现的 verilog 代码可以基于此完成，在访存之前先判断数据是否存储在 cache 中。可惜在本实验中我实现的是简单的单周期 cpu，实现 cache 意义不大（

题目 3
总结一个完整的计算机系统的设计方法和步骤。

解答：
其实完整的计算机系统和一个普通的硬件项目区别不大，首先分析需求，然后进行模块化的顶层设计（例如cpu、存储器、外设和外设接口），分模块调试保证正确性之后进行连接，再通过软件测试的执行情况确定硬件实现的正确性。最后就是软件开发环节了。

---

7 写在最后

本次实验可能是我做过的最艰难的实验，因为粗心大意写出 bug 的情况存在，因为判断失误浪费大量 debug 时间的情况也存在，但是最后还是成功地实现了预期功能。从底层的每个逻辑门到指令集，再到计算机系统和上层应用程序，我已经经历了很多。希望以后还能在探索计算机科学的道路上越走越远。