**《计算机组成原理大实验》实验报告**

计54 秦岳 2015011333

计54 梁宸 2015011325

计54 祝方韦 2015011317

1. **实验概述**

本次实验为计算机组成原理的大型综合性系统实验，在本次实验中，本组完成了支持流水线结构的计算机系统的设计与实现。本组实现了基于流水线架构、支持完整的指令集（25条基本指令+19条扩展指令）、能正确高效地处理三种冲突的计算机系统（能够以50MHz的主频运行监控程序，并在性能测试程序1235中达到50MHz的极限速度），并实现了一系列外设的支持（VGA、PS2键盘、串口、Flash等），完成双机通信、中断、Flash自启动等拓展功能。除此之外，本组实现了MIPS16汇编器，并使用该汇编器编写了dos风格主界面管理程序、元胞自动机、贪吃蛇、计算器、双机通信聊天程序，移植了Term监控程序。具体实现的功能与实现细节将在下文中进行详细说明。

1. **实现功能**

本组实现的功能可以概括为三类：基于流水线架构的CPU、外设与扩展功能的硬件支持、相关的软件（操作系统、外设驱动、应用程序）。具体见下表：

|  |  |
| --- | --- |
| 基础CPU架构及功能 | 五级流水线架构的CPU，带旁路与延迟槽 |
| 结构冲突、数据冲突、控制冲突的解决 |
| 支持44条完整的16位MIPS指令集 |
| 内置的时钟分频模块，实现数字分频 |
| 流水线运行高效无冲突，能以50MHz主频运行监控程序 |
| 外设与扩展功能的硬件支持 | VGA彩色显示的支持，支持显示完整的ASCII码符号集以及彩色图片 |
| Block Ram片内显存+外部字符集、图片集的显存架构 |
| PS2键盘输入的支持，支持输入基本字符集 |
| PS2键盘支持组合键输入（Shift、CapsLock）、特殊按键的输入（退格、回车等） |
| 使用串口进行双机通信 |
| 中断的实现 |
| 基本内存与扩展内存访问的支持，其中Ram2被用于显存 |
| Flash读写的支持，开机自启动 |

|  |  |
| --- | --- |
| 软件支持 | 扩展MIPS16汇编器，支持宏定义、动态数据空间定义、静态字符串定义、符号跳转、函数调用、变量赋值、缓存、错误警告等功能 |
| 相关算术运算的汇编实现，包括booth算法乘法、除法、幂运算，伪随机数发生器等 |
| 字符串处理函数集，字符串/整数转换，scanf\printf |
| 输入输出的控制模块，打印缓冲区的控制 |
| VGA、PS2、串口驱动与相关接口函数 |
| DOS风格的主界面管理程序 |
| 移植Term，使其能通过Flash自启动加载，并支持汇编（A）、查看内存（D）、反汇编（U）、执行（G）、查看寄存器（R）等指令 |
| Life Game 元胞自动机 |
| RetroSnake 贪吃蛇 |
| 计算器，支持加法、减法、乘法、整除、求余、幂运算等 |
| 双机通信聊天程序 |

1. **实现细节**

以下对各部分的实现细节作具体地说明。

**基础CPU架构的设计：**

本组设计的CPU基础架构为五级流水线架构，将一个运行周期划分为取指（IF）、指令译码（ID）、执行（EXE）、访存（MEM）、写回（WB）五个阶段，并在每两个阶段之间加入流水线寄存器，用来保存一个流水段传送到下一个流水段的所有数据和控制信息。各阶段的设计思路如下：  
**IF段：**PC模块根据上一阶段的PC值、分支标志、分支跳转PC值、暂停信号、清空信号来确定本阶段的PC值，通过ROM取出指令，传递给下一阶段。

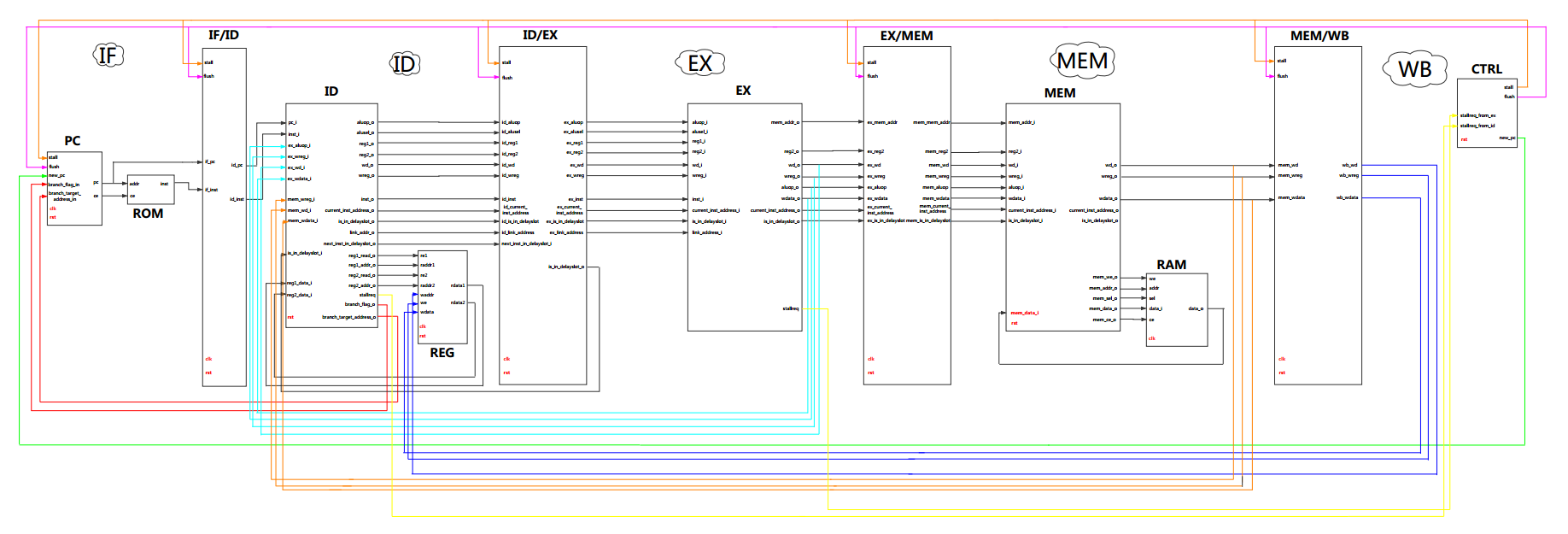
**ID段：**指令译码，翻译指令信息，包括指令类型（算术指令、访存指令、空指令）、是否写回寄存器、写回寄存器的地址、是否在延迟槽。若为算术指令，则需确定两个操作数（寄存器或者立即数，取出正确的值）；若为分支跳转指令，则输出对应的标志与地址信息；若为LW类写回寄存器指令，则暂停流水线。

**EXE段：**从ID段获取运算类型、两个源操作数以及是否写回寄存器的信号，进行相应的运算后将结果传递给MEM段和ID段

**MEM段：**执行访存相关的操作，若为LW类指令，则需将数据传给ID段。

**WB段：**写回寄存器。

**控制单元：**接受暂停信号，发送暂停指令与暂停PC值。

整体的数据通路如下图（大图见附件）：

**旁路技术与三种冲突的处理：**

由于在ID段指令译码时该指令之前的指令并未完全执行完成，若之前的指令中有涉及修改寄存器的指令，则会发生数据冲突，一个解决方案是暂停流水线，等之前指令完全执行完成，另一个方案是使用数据旁路，将EXE段与MEM段的数据提前引回ID段，ID段提前获得之前指令的寄存器修改值，显然后者效率更高。为了ID时序上一定能获得正确的寄存器值，对于EXE段的写回，无需暂停流水线，但是对于LW类MEM段的写回，需要将流水线暂停一个周期（因为一条读寄存器指令接着一条LW指令，当前者到达ID段，后者还需要一个周期才能取出正确的写回值）。旁路技术较高效地解决了CPU运行中的数据冲突。

对于结构冲突，主要是体现在硬件资源的竞争上，在本组的设计中，每个模块的每个接口，以及所有流水线寄存器，其都有单一的明确的功能，故不存在结构冲突。

对于控制冲突，本组没有采用分支预测的技术，而是直接在ID段进行判断，若当前为分支跳转指令，则置branch\_flag为1，branch\_addr为目标地址。该方案的延迟与写回寄存器的延迟是一致的，即若写回寄存器无延迟，则分支的目标指令一定会在延迟槽后一个周期执行，无延迟，若写回寄存器最多有1周期延迟（LW类指令），则分支的目标指令也会有1周期的延迟（因为整个流水线暂停了1周期，其时序逻辑仍保持正确）。该处理方案的好处是逻辑简单、与数据冲突处理保持一致，且其效率基本上与分支预测相等。

**完整的16位MIPS指令集支持：**

本组的CPU支持44条完整的16位MIPS指令集，在define.vhd中定义了ALU的指令码，使用8位的std logic vector按照实验指导书中的编号对ALU操作码进行编码，指令在ID段完成译码，包括确定指令类型、ALU操作码、两个操作数，以及是否写回寄存器，是否为分支跳转指令等，并将信号传递到相应的阶段。指令的具体实现见源代码。

**时钟分频模块：**

本组采用ISE内置的ipcore，综合出一个digital clock management（DCM）模块，实现输入时钟的0~50MHz的数字分频。

**VGA模块与字符集显示：**

VGA（Video Graphics Array）为一种标准的显示接口，其使用红、绿、蓝三种基色混合成常见的颜色。实验芯片提供的VGA接口中，R、G、B每种基色使用3位控制，本组的VGA显示分辨率为640 \* 480，刷新率为60Hz，计算可得，需要的像素时钟为25.18MHz，可通过对输入时钟（50MHz）进二分频得到像素时钟。实际扫描的像素区域为800 \* 525，其中行同步信号在656~752像素为低电平，场同步信号在490~491像素为低电平，使用两个计数器来记录当前像素状态，即可实现VGA显示。

本组的VGA模块的设计如下：由于芯片内置储存不足，无法直接综合出640 \* 480的显存，我们采用了如下方案。将屏幕划分为80 \* 30个大小为为8 \* 16的区域，在芯片内综合出一块Block Ram，用来存放80 \* 30区域中每个区域的字符类型与颜色信息，而对于具体的像素信息，则以字符集的方式储存到外置的ROM中，本组使用Ram2作为ROM，将预先生成的字符集写入0x8000位置。

在VGA的控制模块内，我们维护了全局的像素坐标信息、当前输出块的坐标信息以及当前像素相对于输出块的信息。设当前渲染的像素坐标为（x，y），则提前一个周期向Block Ram询问该坐标对应的块的字符类型，然后在当前周期将像素信息从ROM中读出，输出到VGA。

字符集采用标准的ASCII码字符集（如下图），使用Python生成符合VGA模块格式的二进制编码，对于元胞自动机、贪吃蛇等应用，将用到的图片素材切割为不同的块，用类似的方法生成二进制编码。

**PS2键盘的控制与译码：**

ps2.vhd为PS2键盘的控制模块，keyboard.vhd为键盘的译码模块。我们使用一个状态机来维护模块读取串行数据的状态。开始时状态机处于等待状态，当接受到读取信号，依次读取8位数据与校验位，若校验成功，则输出使能置1，将键盘扫描码输出给译码模块。对于译码模块keyboard.vhd，它接收键盘扫描码，若为MAKE码，且对应的为ASCII码字符，则将其译码后输出对应的ASCII码并将输出使能暂时置为1；若为BREAK码，则输出0且同样将输出使能暂时置为1。若为MAKE码，且对应的按键为功能按键，则修改对应的shiftModifier, LshiftModifier, RshiftModifier, capsModifier, upperModifier信号，在输出ASCII码时结合上述信号确定正确的输出值。对于ESC，上下左右键等无ASCII码的按键，使用了ASCII码表中未使用的控制码对应的值。

**串口与内存读写：**

读写串口的功能中，地址、数据总线、OE、EN由组合逻辑控制，WE由if段与mem段给出的使能信号与时钟信号共同控制。

在注重性能，Ram1作数据存储器Ram2作指令存储器的情况下，

当mem段给出读使能信号时，将Ram1Addr置为mem段给出的地址，然后根据具体地址进行不同操作：

若地址为0xbf00（串口），将Ram1EN、Ram1WE与Ram1OE置为1，wrn置1，rdn置为clk，Ram1Data置为高阻，即可读出串口内容；

若地址为0xbf01（串口状态确认），Ram1EN置0，Ram1WE、Ram1OE、wrn、rdn置1，Ram1Data在第二位置为data\_ready，第一位置为tbre&&tsre，其余位置0；

其余情况下，Ram1EN置0，Ram1WE、wrn、rdn置1，Ram1OE置0，Ram1Data置高阻，即可读出数据；

当mem段给出写使能信号时，将Ram1Addr置为mem段给出的地址，然后根据具体地址进行不同操作：

若地址为0xbf00（串口），将Ram1EN、Ram1WE与Ram1OE置为1，rdn置1，wrn置为clk，Ram1Data置为输入数据，即可向串口写数据；

其余情况下，Ram1EN置0，Ram1OE、wrn、rdn置1，Ram1WE置clk，Ram1Data置输入数据，即可写入数据。若写入地址为0x4000~0x8000间（即Term的用户程序段），同时将Ram2WE置clk，Ram2Data置输入数据，并向if段发出暂停信号。

对if段给出的地址，如没有暂停信号，Ram2EN常置0，Ram2Addr置输入地址，Ram2Data置高阻，即可读出指令。

在使用VGA与PS2键盘运行具体应用的情况下，Ram1作指令、数据存储器，Ram2作图片点阵存储器的情况下，

当mem段给出读使能信号时，将Ram1Addr置为mem段给出的地址，然后根据具体地址进行不同操作：

若地址为0xbf00（串口），将Ram1EN、Ram1WE与Ram1OE置为1，wrn置1，rdn置为clk，Ram1Data置为高阻，即可读出串口内容；

若地址为0xbf01（串口状态确认），Ram1EN置0，Ram1WE、Ram1OE、wrn、rdn置1，Ram1Data在第二位置为data\_ready，第一位置为tbre&&tsre，其余位置0；

若地址为0xbf06(PS2键盘)，返回键盘按键ASCII码的锁存器内容。

其余情况下，Ram1EN置0，Ram1WE、wrn、rdn置1，Ram1OE置0，Ram1Data置高阻，即可读出数据；

当mem段给出写使能信号时，将Ram1Addr置为mem段给出的地址，然后根据具体地址进行不同操作：

若地址为0xbf00（串口），将Ram1EN、Ram1WE与Ram1OE置为1，rdn置1，wrn置为clk，Ram1Data置为输入数据，即可向串口写数据；

若地址为0xbf04（VGA坐标），将输入的数据保存到一个临时变量中，在下一个时钟周期上升沿将它锁存，并置为显存block ram的地址；

若地址为0xbf05（VGA显存信息），将输入的数据放置在显存block ram的数据线，同时将block ram的写使能置为1；

其余情况下，Ram1EN置0，Ram1OE、wrn、rdn置1，Ram1WE置clk，Ram1Data置输入数据，即可写入数据。

只要mem段读写使能信号中的任何一个为1，便向if段输出暂停信号。

对if段给出的地址，如没有暂停信号，Ram1EN常置0，Ram1Addr置输入地址，Ram1Data置高阻，即可读出指令。

对Ram2（点阵显存），Ram2EN常置0，Ram2OE常置0，Ram2WE常置1，Ram2Addr常置VGA模块传来的点位置，在Ram2Addr变化时将Data置为高阻态，即可返回对应的显示内容。

**Flash读写：**

本组实现了自启动功能，即按rst键后会自动从Flash中读出内容放到Ram1和Ram2中，从而启动监控程序或自定义程序。写Flash使用了FlashAndRam软件，读Flash由VHDL代码实现。读Flash时，先将地址准备好，然后将we拉低并将总线置为0x00ff，进入读模式，再将we拉高，oe拉低，读出对应位置数据。这一过程用状态机实现。

由于Flash最高工作频率为11MHz，所以将输入的时钟八分频后作为Flash模块的时钟。

在按rst键后，暂停CPU的一切功能，然后从0地址开始读Flash。由于读Flash的状态机有6个状态，所以内置一个计数器，Flash模块的时钟每经过8个周期，才将读出的内容写入Ram1与Ram2对应位置，并将地址+1。程序中内置了kernelInstNum与fullInstNum两个常数，分别对应监控程序/展示程序的指令数量和包含点阵的所有内容字数。

对效率优先的双存储器版本，kernelInstNum与fullInstNum相等，在地址小于kernelInstNum时，将内容写入指令存储器Ram2；

对运行自定义程序的单存储器版本，在地址小于kernelInstNum时，将内容写入指令存储器Ram1；在地址小于fullInstNum时，将内容写入点阵存储器Ram2。

在所有指令都已写入完成后(地址大于fullInstNum)，模块输出完成信号，CPU开始从0地址段读取指令并运行。

**中断与异常处理：**

本组实现了外部信号中断（键盘中断），实现方法如下：在取指阶段取出的指令信号与IF/ID段之间插入一个中断控制单元，该单元维护一个三状态的状态机，其中三个状态控制的输出分别是原指令、中断指令int、空指令nop，它在无中断信号时维持输出原指令。当外部中断信号到来，该单元将中断信号锁存起来，并在下一个周期将指令替换为int指令，在下下个周期将指令替换为nop。对于int指令，约定int 0 为键盘中断触发指令，int F为中断处理程序返回指令。则在ID段，int 0需要完成两个功能，其一是将当前的PC值放入INT寄存器中，其二是跳转到IH寄存器的地址，特别地，当当前指令位于branch的延迟槽中，应缓存其上一条指令；int F需要完成的功能则是跳转回INT寄存器的地址。IH寄存器中存放中断处理程序的地址，可在程序中用MTIH修改中断处理程序地址。

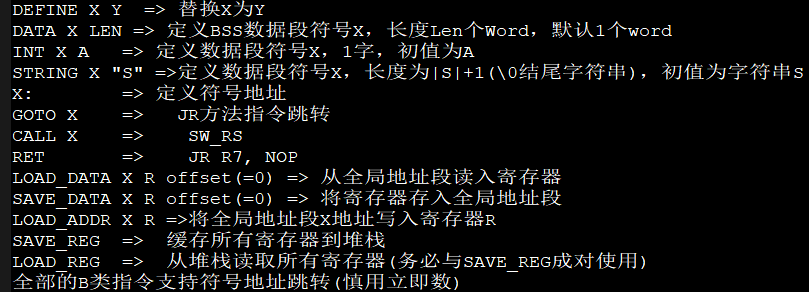
**双机通信：**

双机通信利用了原有的串口，用汇编代码软件实现。在展示双机通信功能的聊天程序中，当程序监测到用户在聊天框内输入了一些内容并按回车时，会依次将输入的字符串中字符的ASCII码从串口发出（发出前会先检测串口的状态，确保tsre与tbre为1），然后在最后补一个回车对应的ASCII码。

而在串口接收到信息时，会触发中断，滤去高8位的噪声后将其缓存起来，并退出中断。接收到回车时，将缓存区的所有内容打印到屏幕上，清空缓存区。

**MIPS16汇编器：**

本组使用Python编写了扩展的MIPS16汇编器，其拥有支持宏定义、动态数据空间定义、静态字符串定义、符号跳转、函数调用、变量赋值、缓存、错误警告等功能。且支持如下语法：



汇编器的算法流程如下：1、替换全局宏定义DEFINE；2、展开语句计算符号地址，并展开中间代码(位置地址占位符为"SigAddr(X)")；3、展开语句生成汇编代码；4、汇编二进制代码文件。具体实现可见Assembler.py。

**VGA、键盘、串口相关接口函数：**

相关外设驱动的汇编接口函数定义如下，具体实现见源代码：

**VGA相关：**

VGA\_COM\_PRINT：将VGA\_MEM通过串口打印到终端，用于测试

VGA\_MEM\_INIT：初始化VGA显存

VGA\_Multi80：快速的\*80，加速计算

VGA\_Draw\_Block：绘图一个格子，R0表示坐标，R1表示颜色等参数

VGA\_Scroll：将VGA显示中的前R0行向上滚一格，空出来的一行填充空格

**键盘相关：**

KeyBoard\_Get：从键盘读取当前内容到R0

**串口相关：**

WAIT\_COM\_W：循环直至串口R0可写

TEST\_COM\_R：返回串口R0是否可读

COM\_WRITE：向串口R0写入字节R1

COM\_READ：从串口R0读数据，无数据返回0，否则返回数据

COM\_SEND：将R0地址开始的数据从BF00发送出去

**算术运算函数：**

我们实现了以下的算术运算函数（具体实现见源代码）：布斯算法计算有符号16位整数乘法R0 \* R1，将LOW保存于R0，HIGH保存于R1；加减交替原码一位除法，R0/R1，商保存于R0，余数存于R1；幂运算，计算R0^R1,返回R0(16位)；伪随机数发生器与快速伪随机数发生器，将15位结果返回至寄存器R0。

**DOS风格的主界面管理程序：**

管理应用程序，支持’ls’和’HELP’命令查看使用方法，错误指令处理，输入应用程序名动态载入应用程序并运行，使用字符串模块进行命令匹配，使用打印控制模块进行排版和输出显示，具体实现见源代码。

**移植Term：**

完全复现PC版Term程序，支持A/D/R/G/U五类指令，系统调度虚拟地址空间缓存程序和数据，不会与kernel与其他应用程序冲突。具体实现见源代码。

**应用程序（贪吃蛇、元胞自动机、计算器、双机通信聊天程序）：**

**贪吃蛇：**WASD控制蛇的移动方向，E键开启作弊模式(一步驱动)，Q键退出，R键重新开始。使用键盘接口模块进行输入操作获取，具体实现见源代码。

**元胞自动机：**按任意键进行一步模拟，A键开启自动模拟模式，R键重置，Q键退出。

**计算器：**输入表达式进行求值，支持加减乘除乘方取模运算，使用数值计算模块进行计算，字符串模块进行表达式处理。

**双机通信聊天程序：**输入文本按Enter发送给对方，并显示在自己的屏幕上，使用串口通信模块。

1. **实验结果**

**性能测试：**

实验提供的五个性能测试的结果如下：

1. **性能标定**

这段程序一般没有数据冲突和结构冲突，可作为性能标定。

\*\*\* 程序说明：R4、R5为循环变量 \*\*\*

\*\*\* 主要循环体0x0D~0x12，6条 \*\*\*

\*\*\* 每条各执行25,000,000次 \*\*\*

\*\*\* 共1.50亿条指令 \*\*\*

\*\*\* （行号从0开始） \*\*\*

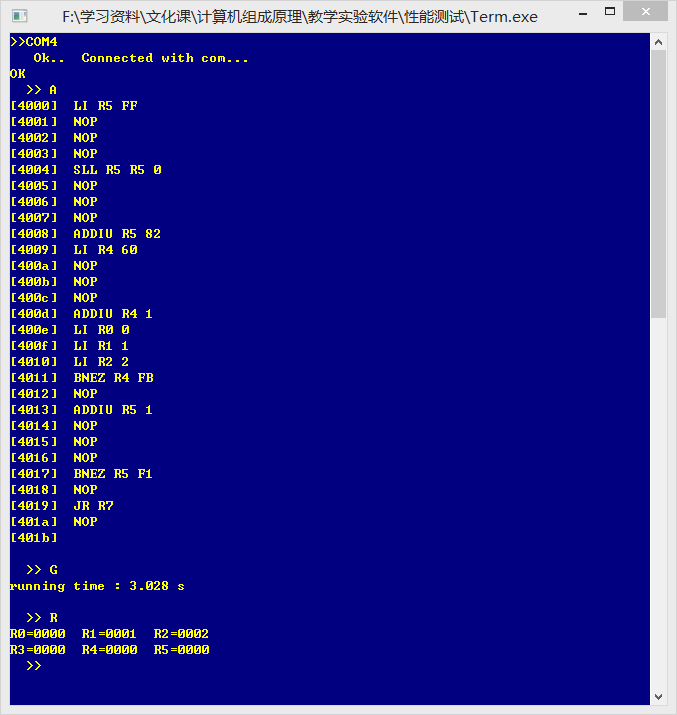
**时钟主频：**50MHz

**运行时间：**3.028s

**正确性：**正确

**分析：**预期时间为1.5 \* 10 ^ 8 / (50 \* 10 ^ 6) = 3s，实际运行时间基本与预期相符。

性能测试截图如下。



**二、运算数据冲突的效率测试**

从这一节起，假设正确处理了数据冲突，有数据冲突的地方不再加NOP。

\*\*\* 程序说明：R4、R5为循环变量 \*\*\*

\*\*\* 主要循环体0x05~0x0D，9条 \*\*\*

\*\*\* 每条各执行25,000,000次 \*\*\*

\*\*\* 共2.25亿条指令 \*\*\*

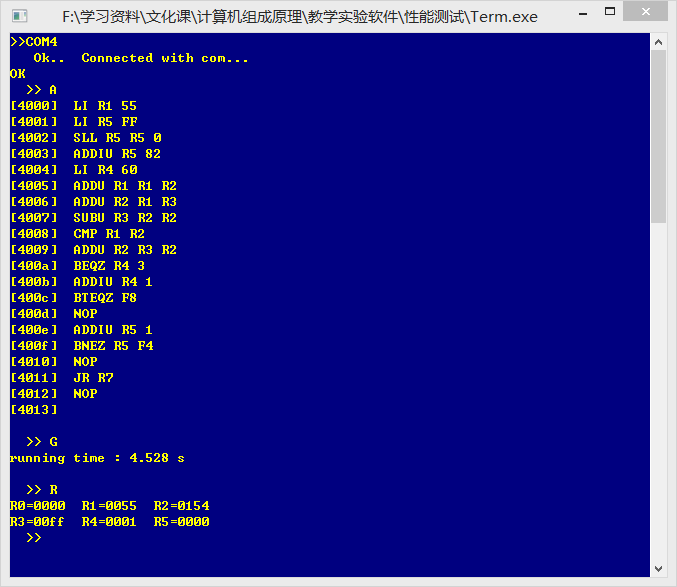
**时钟主频：**50MHz

**运行时间：**4.528s

**正确性：**正确

**分析：**预期时间为2.25 \* 10 ^ 8 / (50 \* 10 ^ 6) = 4.5s，实际运行时间基本与预期相符，测试代码中的循环体内无LW的数据冲突，说明本组的CPU架构能有效地处理非LW数据冲突，且不会造成额外的延时。

性能测试截图如下。



**三、控制指令冲突测试**

从这一节起，假设正确处理了延迟槽，行为与模拟器一样，延迟槽里可能填充语句。

\*\*\* 程序说明：R4、R5为循环变量 \*\*\*

\*\*\* 主要循环体0x06~0x09，4条 \*\*\*

\*\*\* 每条各执行25,000,000次 \*\*\*

\*\*\* 共1.00亿条指令 \*\*\*

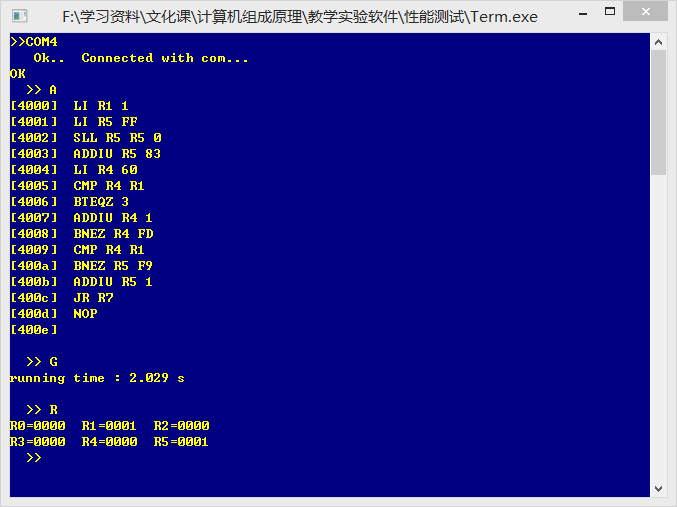
**时钟主频：**50MHz

**运行时间：**2.029s

**正确性：**正确

**分析：**预期时间为1\* 10 ^ 8 / (50 \* 10 ^ 6) = 2s，实际运行时间基本与预期相符，测试代码中的循环体内无LW的控制冲突，说明本组的CPU架构能有效地处理非LW控制冲突，且不会造成额外的延时。

性能测试截图如下。



**四、访存数据冲突性能测试**

\*\*\* 程序说明：R4、R5为循环变量 \*\*\*

\*\*\* 主要循环体0x07~0x0C，6条 \*\*\*

\*\*\* 每条各执行25,000,000次 \*\*\*

\*\*\* 共1.50亿条指令 \*\*\*

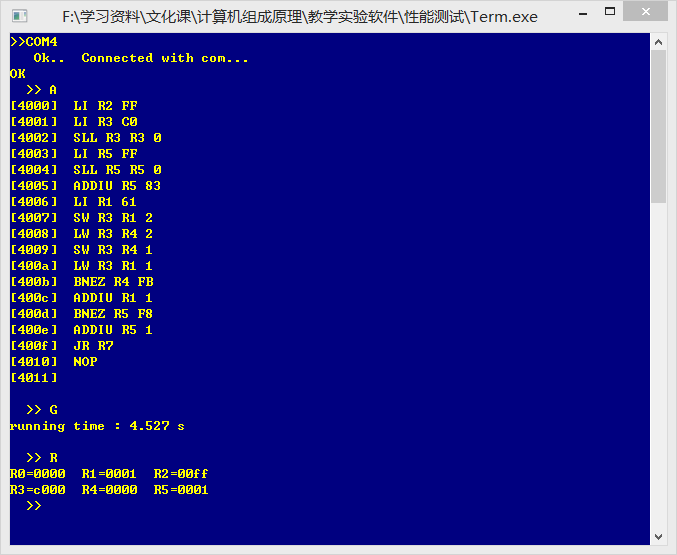
**时钟主频：**50MHz

**运行时间：**4.527s

**正确性：**正确

**分析：**由于代码主循环中有2处访存造成的数据冲突，流水线需暂停两个周期，故执行6条指令实际需要8个周期，预期时间为1.5 \* 8 / 6 \* 10 ^ 8 / (50 \* 10 ^ 6) = 4.5s，此时实际主频约为50MHz \* 6 / 8 = 37.5MHz。实际运行时间基本与预期相符，测试代码中的循环体内有LW的数据冲突，说明本组的CPU架构在处理频繁的LW数据冲突时，性能略有下降，但是仍然能维持较高的效率。

性能测试截图如下。



**五、读写指令存储器测试**

\*\*\* 程序说明：R4、R5为循环变量 \*\*\*

\*\*\* 主要循环体0x07~0x09，3条 \*\*\*

\*\*\* 每条各执行25,000,000次 \*\*\*

\*\*\* 共0.75亿条指令 \*\*\*

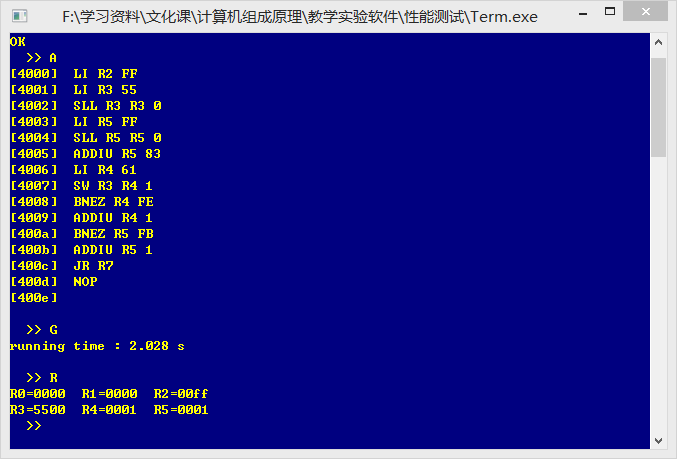
**时钟主频：**50MHz

**运行时间：**2.028s

**正确性：**正确

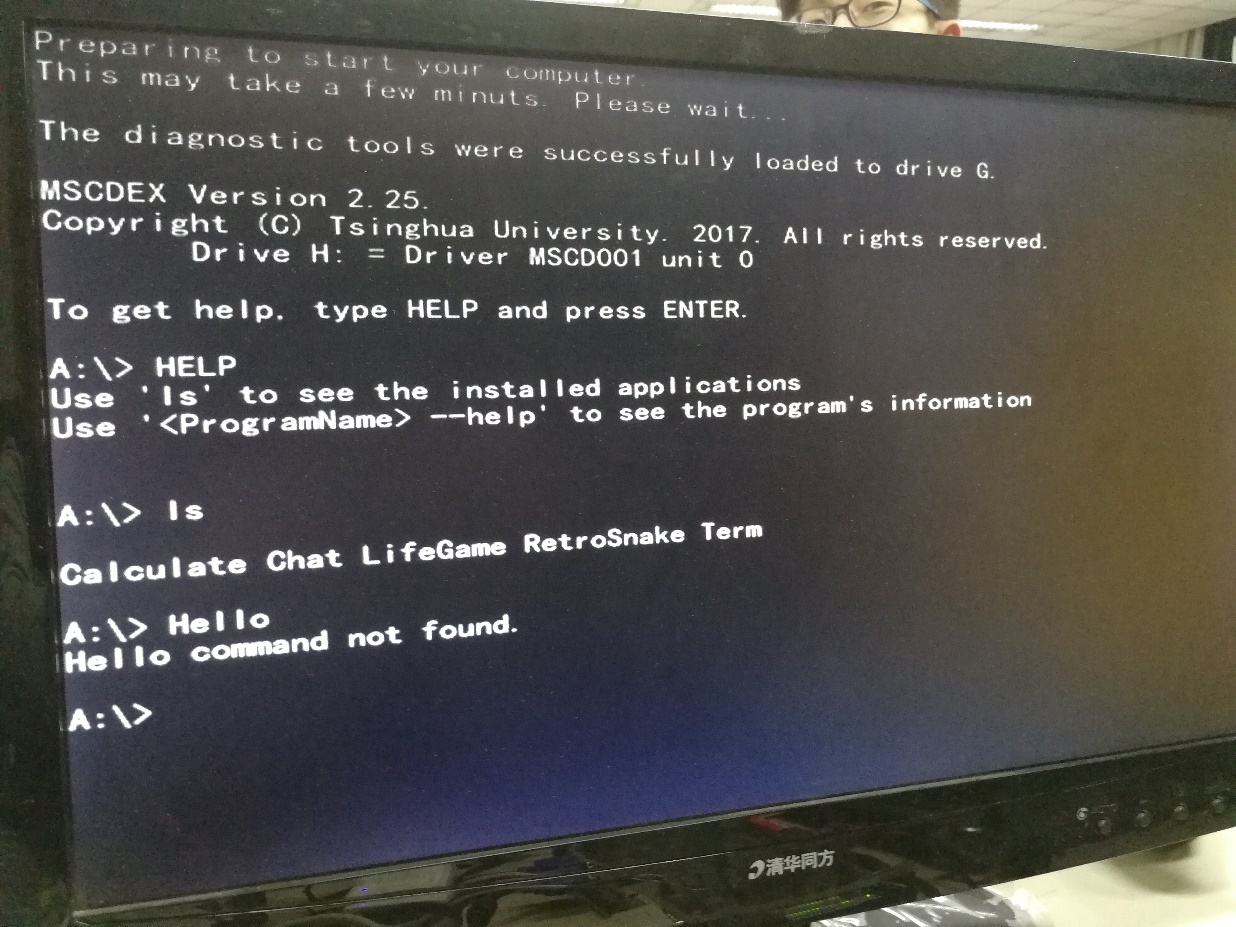
**分析：**由于代码主循环中SW写指令存储器，造成数据冲突，流水线需暂停1个周期，故执行3条指令实际需要4个周期，预期时间为0.75 \* 4 / 3 \* 10 ^ 8 / (50 \* 10 ^ 6) = 2s，此时实际主频约为50MHz \* 3 / 4 = 37.5MHz。实际运行时间基本与预期相符，测试代码中的循环体内有SW写指令存储器的数据冲突，说明本组的CPU架构在处理SW写指令存储器的数据冲突时，性能略有下降，但是仍然能维持较高的效率。

性能测试截图如下。

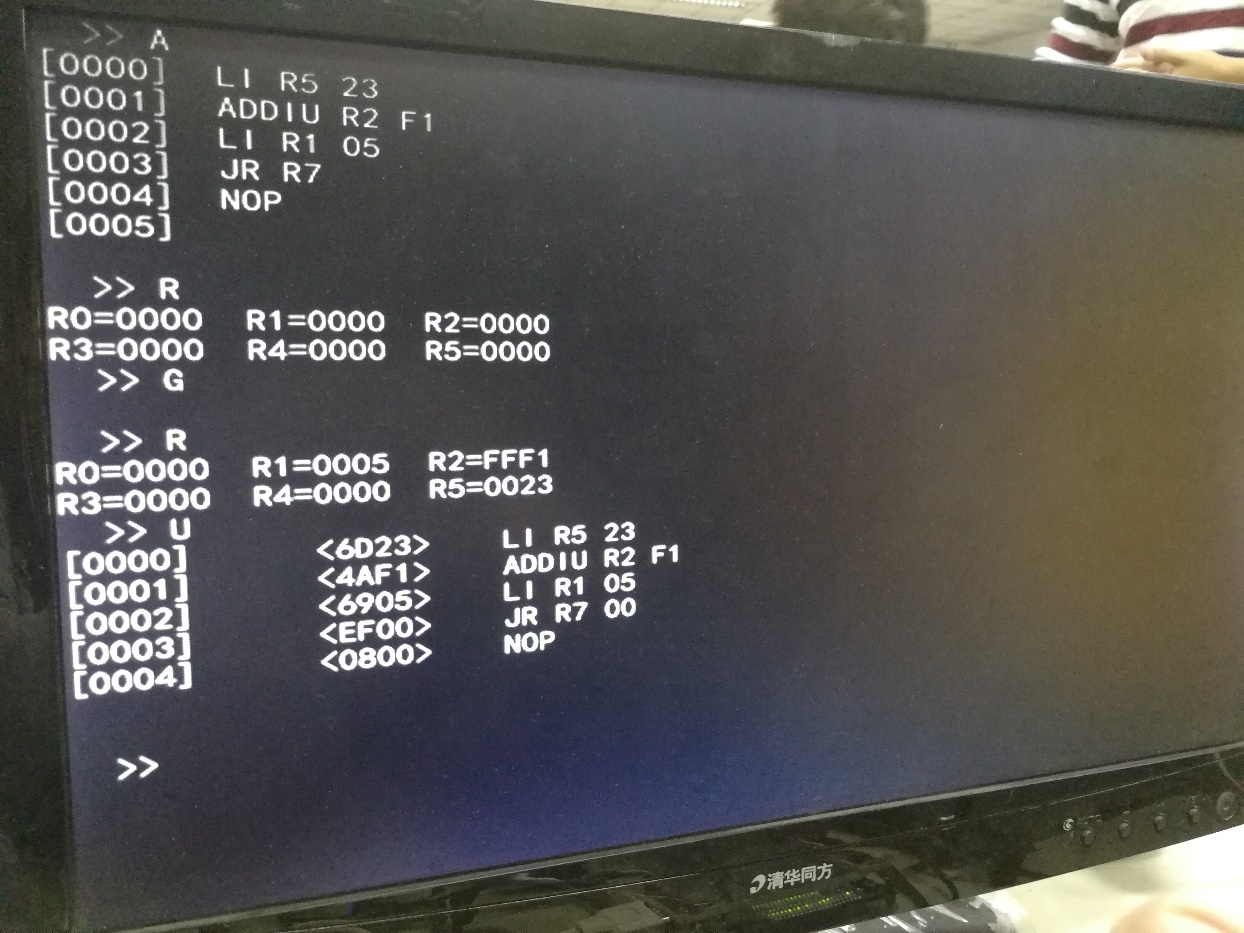


**扩展功能与扩展应用程序：**

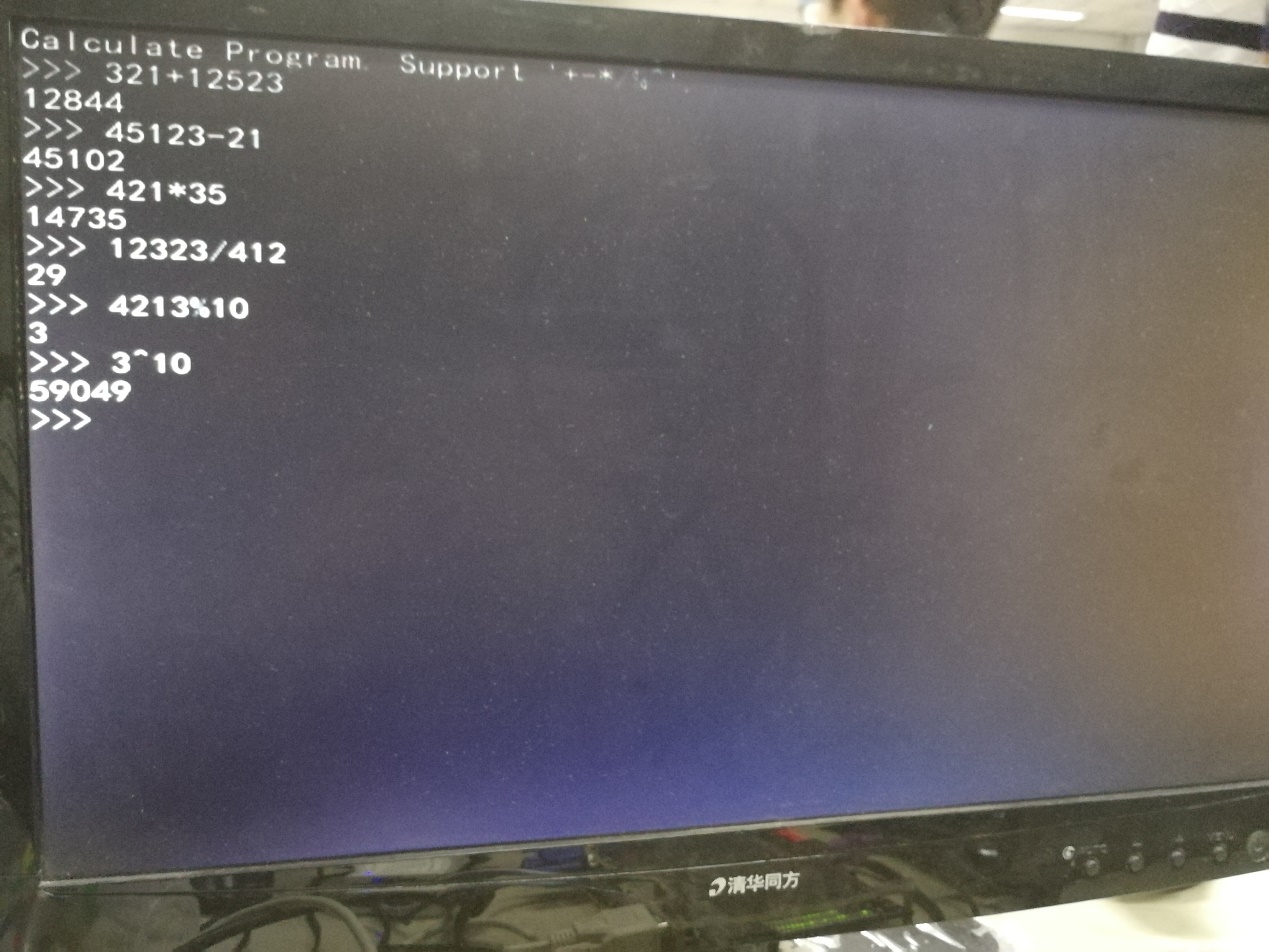
DOS风格的主界面管理程序：



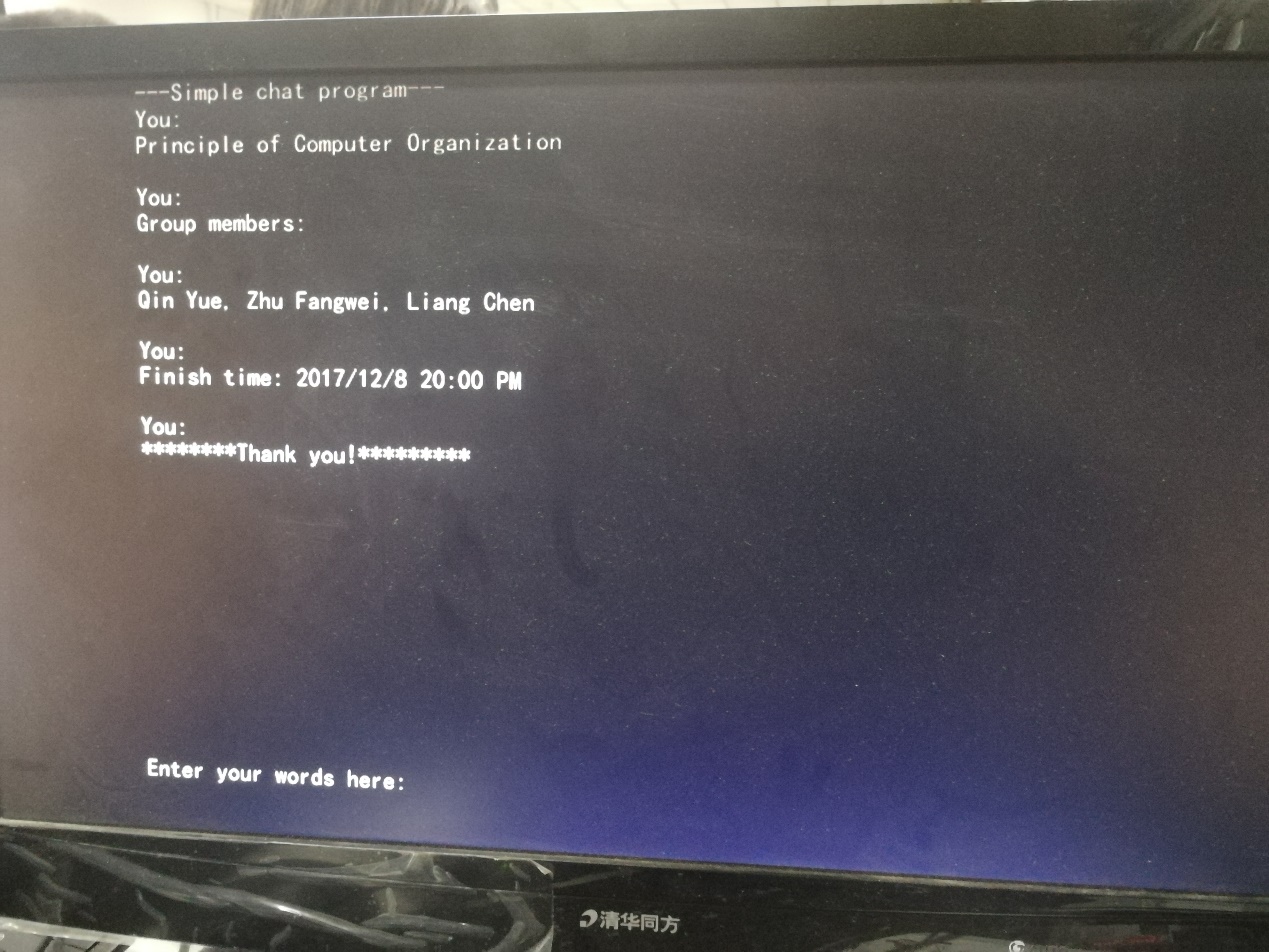
移植的Term：



计算器：



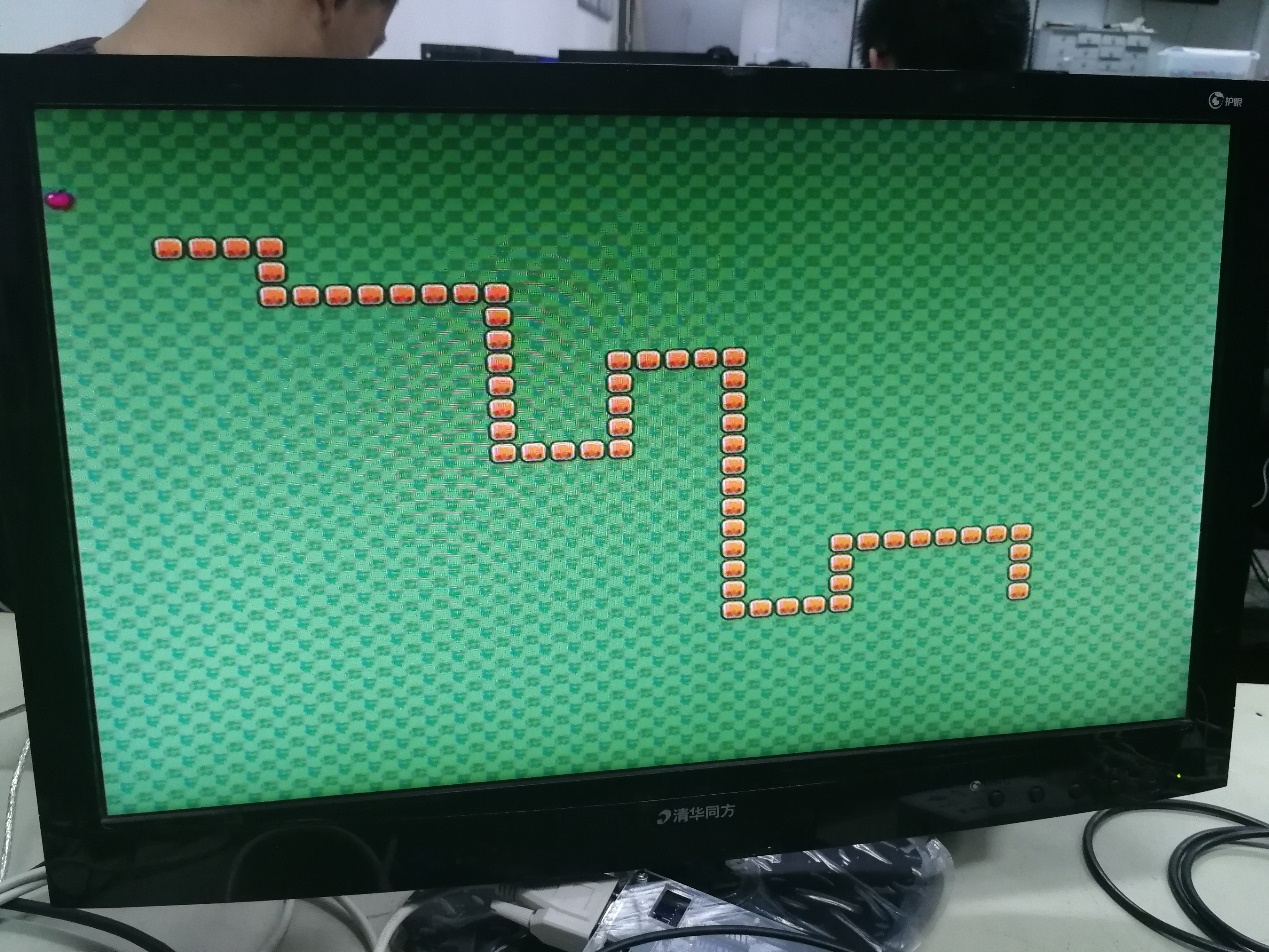
双机通信聊天程序：



元胞自动机：



贪吃蛇：



1. **分工表**

秦岳：数据通路设计、CPU基础框架实现（五段流水线实现、三类冲突处理、流水线暂停、虚拟RAM单元测试框架）、字符集图像制作、汇编器编写、汇编程序编写(DOS管理程序、计算器、生命游戏、贪吃蛇、Term G/R/D、打印控制模块、数值计算/字符串处理模块、串口/VGA/键盘接口)

梁宸：完整MIPS16指令集、数字分频模块、VGA模块、中断处理模块、实验报告撰写

祝方韦：内存串口模块、Flash读写模块、PS2键盘模块、VGA显存管理、双机通信汇编程序及Term A/U指令汇编程序

项目工程地址：<https://github.com/747929791/CPU>

