**《程序设计课程设计》实验报告**

**实验名称 <冯诺依曼式计算机CPU简易模拟器>**

**班 级**

**学 号**

**姓 名**

**颜色优先级分级:**

**1 – 2 – 3 – 4**

**PLUS : 行结构化处理 :**

**0000 0000 0000 0000 0000 0000 0000 0000 (行指令)**

**command (com) front behind immediateValue (im)**

**指令类型 前寄存器 后寄存器 立即数**

**-------------------------------------------------------------------------------**

1. **高层数据结构设计**

**1.1全局变量/变量定义**

因尽量避免全局变量使用 , 故没有全局变量定义

但是有寄存器结构体定义 :

* 1. **寄存器结构体 ( 变量 )**

1. **公用寄存器结构体 – segment**

Form : **struct** Segment

Typedef : segment

Definition : 将公用的code 和 data 创建 , 方便传入线程和函数

进行句柄互斥创建和传递

进行同步锁和线程信号的创建和传递

Contain : 1. 进程互斥对象 : HANDLE hMutex

2. 同步锁 : int lock

3. 线程1信号 : int end1

4. 线程2信号 : int end2

5. 代码1文本 : int\*\* txt1

6. 代码1行数 : int line1

7. 代码2文本 : int\*\* txt2

8. 代码2行数 : int line2

9. 代码寄存器 : int\*\* codeSegment

10.数据寄存器 : int\*\* dataSegmen

1. **线程寄存器结构体 – MTC**

Form : **struct** mtc

Typedef : MTC

Definition : 将寄存器封装以便于 申请 , 传递 , 修改 , 释放

Contain : 1. 程序计数器 : int ip

2. 标志寄存器 : int flag

3. 前16位寄存器 : int ir

4. 数据寄存器和地址寄存器 : int\* ax (空间为9 , 0位舍弃)

Value : 1. ->ip : 0 , 4 , 8 …… 1024

2. ->flag : -1 , 0 , 1

3. ->ir : 0 ~ 32767

4. ->ax [ 1~4 ] : int型 , 看编译器位数

5. ->ax [ 5~8 ] : 16384 , 16386 , 16388 …… 16894

1. **命令行数据处理总集寄存器 – Number**

Form : **struct** Number

Typedef : number

Definition : 处理行数据化为十进制 , 便于指令函数传参简化

Contain : 1. 01-08转十进制 : int com

2. 09-12转十进制 : int front

3. 13-16转十进制 : int behind

4. 17-32转十进制 : int im ( 补码转换 )

**1.2模块常量即变量定义**

**1.21 – 主函数 :**

1. **公用寄存器 – solid**

Form : struct Segment\*

Function : 将所有需要生成的公用数据一次性创建 ( 避免使用全局变量 )

1. **文件指针 ( dict1.dic ) – fPtr1**

Form : FILE\*

Function : 将文件指针传入子函数

1. **文件指针 ( dict2.dic ) – fPtr2**

Form : FILE\*

Function : 将文件指针传入子函数

1. **指令行数 ( dict1.dic ) – segment -> line1**

Form : int

Function : 计算停机指令包括在内的dict1的所有指令的行数

控制在free文本1时的规模

1. **指令行数 ( dict2.dic ) – segment -> line2**

Form : int

Function : 计算停机指令包括在内的dict2的所有指令的行数

控制在free文本2时的规模

1. **dict1.dic指针 – segment-> txt1**

Form : int\*\*

Function : 二维数组指针 , 用来存储dict1.dic , 32列的指令 , 即指令集

1. **dict2.dic指针 – segment -> txt2**

Form : int\*\*

Function : 二维数组指针 , 用来存储dict2.dic , 32列的指令 , 即指令集

1. **模拟代码空间指针 ( 变量 ) – segment -> codeSegment**

Form : int\*

Function : 一维数组指针 , 存储模拟代码空间的指针 , 即表示代码空间

1. **模拟数据空间指针 ( 变量 ) – segment -> dataSegment**

Form : int\*

Function : 一维数组指针 , 存储模拟数据空间的指针 , 即代表数据空间

**1.22 – 子函数 : 线程函数**

**thread1 : 线程1函数**

**1. 线程寄存器 – cur**

Form : struct mtc\*

Function : 将实时的寄存器封装 , 便于集总传参 , 也易于阅读

**2. 线程ID – cur -> id**

Form : int

Function : 赋值为1,记录线程1id

**3. 线程程序计数器 – cur -> ip**

Form : struct mtc\*

Function : 线程1程序计数器

**4. 线程标志寄存器 – cur -> flag**

Form : int

Function : 线程1标志寄存器

**5. 线程命令寄存器 – cur -> ir**

Form : int

Function : 行命令 ( 前16位 ) 转十进制

**6. 线程寄存器 – cur -> ax**

Form : int\*

Function : 模拟线程1的八个寄存器

**7. 命令寄存器 – num**

Form : struct num\*

Function : 将行命令封装 , 便于传参和阅读

**8. 命令暂存变量 – num -> com**

Form : int

Function : 计算线程1当前指令行command数值 , 来进行指令选择

**9. 前寄存器暂存变量 – num -> front**

Form : int

Function : 计算线程1当前指令行前一个寄存器的下标

1. **后寄存器暂存变量 – num -> behind**

Form : int

Function : 计算线程1当前指令行后一个寄存器的下标

1. **立即数 – num -> im**

Form : int

Function : 计算线程1当前指令行后一个寄存器的下标

**thread2 : 线程2函数**

**1. 线程寄存器 – cur**

Form : struct mtc\*

Function : 将实时的寄存器封装 , 便于集总传参 , 也易于阅读

**2. 线程ID – cur -> id**

Form : int

Function : 赋值为1,记录线程1id

**3. 线程程序计数器 – cur -> ip**

Form : struct mtc\*

Function : 线程2程序计数器

**4. 线程标志寄存器 – cur -> flag**

Form : int

Function : 线程2标志寄存器

**5. 线程命令寄存器 – cur -> ir**

Form : int

Function : 行命令 ( 前16位 ) 转十进制

**6. 线程寄存器 – cur -> ax**

Form : int\*

Function : 模拟线程2的八个寄存器

**7. 命令寄存器 – num**

Form : struct num\*

Function : 将行命令封装 , 便于传参和阅读

**8. 命令暂存变量 – num -> com**

Form : int

Function : 计算线程2当前指令行command数值 , 来进行指令选择

**9. 前寄存器暂存变量 – num -> front**

Form : int

Function : 计算线程2当前指令行前一个寄存器的下标

1. **后寄存器暂存变量 – num -> behind**

Form : int

Function : 计算线程2当前指令行后一个寄存器的下标

1. **立即数 – num -> im**

Form : int

Function : 计算线程2当前指令行后一个寄存器的下标

**1.23 – 子函数 : 空间生成(malloc) , 存入数据/初始化 , 释放(free)**

**txt : 文本暂存空间 生成 , 存入数据**

1. **二维数组指针 – txt**

Form : int\*\*

Function : 申请文本空间并使用fPtr 存储指令集 , 并随函数返回地址

1. **字符读取载体 – ch**

Form : char

Function : 在遍历fPtr中逐个读取字符 , 进行相关操作

1. **动态下标 – i , j**

Form : int

Function : 随fPtr遍历时对 txt文本进行操作

**code : 代码模拟空间 生成 , 存入数据**

1. **一维代码指针 – code**

Form : int\*

Function : 申请代码空间并遍历txt将十进制代码储存入codeSegment

1. **动态下标 – i , j**

Form : int

Function : 便于遍历 txt 文本指令

**data : 数据模拟空间 生成 , 初始化**

1. **一维数据指针 – data**

Form : int\*

Function : 申请代码空间并初始化

1. **动态下标 – i**

Form : int

Function : 便于二维指针循环申请一维空间

**struct : 寄存器结构体 生成 , 初始化**

1. **结构体指针 – cur**

Form : **struct** mtc

Function : 申请寄存器结构体并初始化 , 便于打包寄存器传参 , 简化代码

**mainFree : 主函数释放调用空间**

**1. 动态下标 -- i**

Form : int

Function : 用于循环释放所有用 malloc 申请的内存

**1.23 – 子函数 : 计算外包函数 ( 过程复杂 )**

**line : 计算行数**

1. **行数计算器 – line**

Form : int

Function : 行数计算器定义和初始化为 0

1. **字符读取载体 – ch**

Form : char

Function : 便于遍历文件fPtr时读取文件字符并操作

1. **前8位数之和 – sum**

Form : int

Function : 循环计算每行的前8位数之和 , 便于判断停机指令 , 跳出循环

1. **循环条件 – Flag**

Form : int

Function : 控制循环条件

1. **动态下标 – i**

Form : int

Function : 便于循环控制计算前8位数字

**immediateValue : 计算立即数**

1. **立即数计算器 – im**

Form : int

Function : 立即数定义和初始化为 0

1. **立即数正负标记 – Flag**

Form : int

Function : 控制标记 , 便于判断正负

1. **后15位复制数组 – copy[15]**

Form : int\* , int

Function : 当立即数为负时 , 复制后15位数进行立即数绝对值计算

1. **动态下标 – i**

Form : int

Function : 便于操作数组

**1.24 – 子函数 : 指令集函数**

全部操作传入参数 ( 寄存器结构变量和暂存参数变量 )

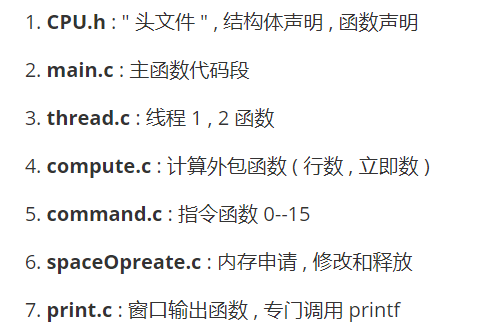
无子函数参数定义

**1.25 – 子函数 : 窗口输出总集函数**

全部操作传入参数 ( 寄存器结构变量和暂存参数变量 )

无子函数参数定义

**-------------------------------------------------------------------------------**

1. **系统模块划分**

**2.1系统模块结构图**

单个.c没有模块设计

项目有模块设计, 在README中有详细介绍. --->

**2.2各模块函数说明**

**2.21主函数**

Form : int

Parameter : int argc , char\* argv[ ]

Function : 打开文件

文件内容传输至txt

计算 com , front , behind , im .

判断 com 进入指令子函数操作操作

适时输出相应值

Return : int 0

**2.22子函数 : 空间生成(malloc) , 存入数据/初始化 , 释放(free)**

**2.221 txt : 文本暂存空间 生成 并 存入数据**

Form : int\*\*

Parameter : FILE\* fPtr , int line

Function : 将停机指令包括在内的文本内容导入本地 txt

地址返回主函数

Return : int\*\* txt

**2.222 code : 代码模拟空间 生成 并 存入数据**

Form : int\*

Parameter : int\*\* txt , int line

Function : 申请代码空间并由txt指令集将代码信息存入 codeSegment

地址返回主函数

Return : int\* code

**2.223 data : 数据模拟空间 生成 并 初始化**

Form : int\*

Parameter : nothing

Function : 申请数据空间并初始化为 0

地址返回主函数

Return : int\* data

**2.224 struct : 寄存器结构体 生成 并 初始化**

Form : **struct** mtc\* ( MTC\* )

Parameter : nothing

Function : 申请寄存器空间并初始化为 0

地址返回主函数

Return : MTC\* cur

**2.225 释放内存空间**

Form : void

Parameter : **struct** Segment

Function : 释放上述函数申请的内存 , 节省内存空间

Return : void

**2.23 子函数 : 线程函数**

**2.231 线程 1**

Form : unsigned

Parameter : **void\* pArguments**

Function : 将数据在主函数用结构体egment\* solid封存 , 传入pArguments

再将pArguments转化为segment型使用

进行指令执行和同步锁句柄双层互斥 , 并输出寄存器状态

Return : unsigned

**2.231 线程 2**

Form : unsigned

Parameter : **void\* pArguments**

Function : 将数据在主函数用结构体egment\* solid封存 , 传入pArguments

再将pArguments转化为segment型使用

进行指令执行和同步锁句柄双层互斥 , 并输出寄存器状态

Return : unsigned

**2.24 子函数 : 计算外包函数 ( 过程复杂 )**

**2.241 计算行数**

Form : int

Parameter : FILE\* fPtr

Function : 计算停机指令包括在内的指令行数

Return : int line

**2.242 计算立即数**

Form : int

Parameter : int\*\* txt , int c , int l

Function : 计算后 16 位补码所构成的立即数

Return : int im

**2.25子函数 : 指令集函数**

**2.251 停机指令 – 0**

Form : void

Parameter : int com , int front , int behind , int im , MTC\* cur , int\* data

Function : 停机 并 ip计数 , ir 修正

Return : void

**2.252 数据传送指令 – 1**

Form : void

Parameter : int com , int front , int behind , int im , MTC\* cur , int\* data

Function : 将数据传输 并 ip计数 , ir修正

Return : void

**2.253 算数运算指令 – 2 3 4 5**

Form : void

Parameter : int com , int front , int behind , int im , MTC\* cur , int\* data

Function : 进行 + - \* / 运算 并 ip计数 , ir 修正

Return : void

**2.254 逻辑运算指令 – 6 7 8**

Form : void

Parameter : int com , int front , int behind , int im , MTC\* cur , int\* data

Function : 进行 && || ! 运算 并 ip计数 , ir 修正

Return : void

**2.255 比较指令 – 9**

Form : void

Parameter : int com , int front , int behind , int im , MTC\* cur , int\* data

Function : 进行大小比较修正寄存器flag 并 ip计数 , ir 修正

Return : void

**2.256 跳转指令 – 10**

Form : void

Parameter : int com , int front , int behind , int im , MTC\* cur , int\* data

Function : 适时修改 ip 并 ir 修正

Return : void

**2.257 输入输出指令 – 11 12**

Form : void

Parameter : int com , int front , int behind , int im , MTC\* cur , int\* data

Function : 输入 in : / 输出 out : 并 ip计数 , ir 修正

Return : void

**2.26子函数 : 窗口输出总集函数**

**2.261 main函数输出**

Form : void

Parameter : segment\* solid

Function : 输出寄存器状态

如果com == 0 即停机指令 , 则输出codeSegment , dataSegment

Return : void

**2.262 线程函数输出**

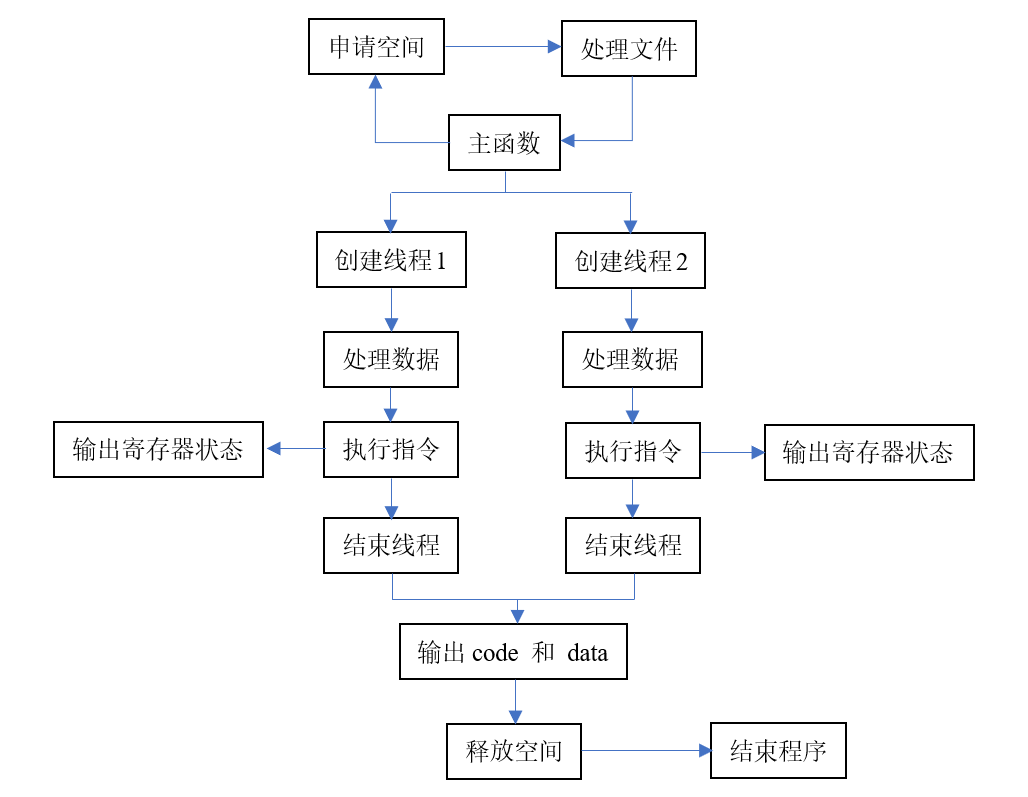
Form : void

Parameter : int com , MTC\* cur , segment\* solid

Function : 输出data和code

Return : void

**2.3函数调用图示及说明**

****

**-----------------------------------------------------------**

1. **高层算法设计**

**因为word实在难以实现 N-S图 格式**

**故用伪代码表示**

**伪代码实现:**

**主函数 :**

**int main ( int argc , char\* argv[ ] )**

**{**

**适当申请空间 , 并进行文件数据处理 ;**

**创建寄存器 , 将信息集中传输**

**创建线程1, 线程2 ;**

**结束线程1 , 线程2 ;**

**输出 code , data**

**return 0 ;**

**}**

**线程函数 :**

**unsigned \_\_stdcall Fun2Proc ( void\* pArguments )**

**{**

**结构体数据初始化 ;**

**计算结构体中的 指令 , 前寄存器 , 后寄存器 , 立即数的十进制 ;**

**将传参的线程的公用线程信号状态改为有**

**循环 :**

**选择指令 :**

**if 0 : 停机 ;**

**else if 1 : 数据传输 ;**

**else if 2 : 加法运算 ;**

**else if 3 : 减法运算 ;**

**else if 4 : 乘法运算 ;**

**else if 5 : 除法运算 ;**

**else if 6 : 逻辑与运算 ;**

**else if 7 : 逻辑或运算 ;**

**else if 8 : 逻辑非运算 ;**

**else if 9 : 比较指令 ;**

**else if 10 : 跳转指令 ;**

**else if 11 : 输入指令 ;**

**else if 12 : 输出指令 ;**

**else if 13 : 上锁 ;**

**else if 14 : 释放 ;**

**else if 15 : 线程睡眠sleep ;**

**输出寄存器和适当操作 ;**

**如果到停机指令 , 则跳出 ;**

**释放申请空间 ;**

**return ;**

**}**

**计算立即数函数 :**

**void scanImmediateValue ( FILE\* fPtr )**

**{**

**int im = 0 ;**

**if ( 开头为 0 )**

**{**

**( 循环16 次 ) :**

**{**

**im = 2 \* im + 对应位上的数 ; // 秦九韶因式分解算法**

**}**

**}**

**if ( 开头为 1 ) :**

**{**

**copy 原 16 位数组 ;**

**+1 ;**

**进位 ;**

**计算绝对值 (** **秦九韶因式分解算法 )**

**取负**

**}**

**return im ;**

**}**

**模拟codeSegment :**

**int\* generateCodeSpace ( int\*\* txt , int line )**

**{**

**申请连续的 128 空间 ; // 模拟的代码空间**

**循环 line 次**

**{**

**将txt每一行的信息转化位二进制存储到连续的代码空间中 ;**

**}**

**return 空间开头指针 ;**

**}**

**模拟 dataSegment :**

**int\* generateDataSpace ( )**

**{**

**申请连续的 256 的空间 ; // 模拟从16384开始的地址空间 ;**

**return 空间头指针 ;**

**}**

**txt : 生成文本空间 :**

**int\*\* generateTxtSpace ( FILE\* fPtr , int line )**

**{**

**申请line行 , 32列的空间 ;**

**循环 line次 :**

**{**

**将32个数存进 txt 的对应行 ;**

**丢掉行剩余字符 ;**

**}**

**return 空间头指针 ;**

**}**