**AsmMax开发计划书**

AsmMax是一款基于8086汇编语言的集成开发环境。本项目开发计划书是用于总体上指导AsmMax项目顺利进行并最终得到可交付的项目产品。本项目开发计划面向项目组全体成员。

1. **功能分析**
   1. **需求分析**
      1. **背景**

由于8086/8088汇编语言大部分应用于计算机基础教学，可用的集成开发环境少之又少；高校用于进行汇编实验的实验室所使用的XP操作系统已经淡出市场；Dosbox使用起来又较为复杂；故高校计算机专业学生需要一款8086/8088汇编语言IDE以便于汇编语言的学习。

现计划开发一款名为AsmMax汇编语言IDE。该汇编语言IDE功能模块有项目文件管理、文本编辑、代码编辑、调试及运行四个部分。

**1.1.2 项目文件管理**

支持新建项目，可以按工程管理文件。

支持项目文件的基本操作，即新建，打开，删除，命名，保存，另存为，及选择指定文件进行编辑。

**1.1.3 文本编辑**

支持文本编辑的基本操作，即复制，粘贴，查找，替换，撤销，重做。

**1.1.4 代码编辑**

支持代码编辑时的一些常用功能，包括语法高亮、关键字补全、简单语法检查。

**1.1.4.1 语法高亮**

对不同类型的词块进行上色。

**1.1.4.2 关键字补全**

根据汇编的关键字列表，以及用户创建的标号，对每个词块的不完全输入生成一个补全列表，用户可交互性的选择补全选项。

**1.1.4.3 简单语法检查**

短上下文的语法检查，出错后对该行进行提示。

**1.1.5 调试及运行**

**1.1.5.1 调试**

添加断点，单步执行，暂停，继续等操作。单步执行时，可以显示各个寄存器及内存的数据。

**1.1.5.2 运行**

根据工程文件包含的代码文件自动进行编译连接执行，能够提供编译连接执行时的各种信息。

**1.1.6 代码编写的社区**

与github进行连接，用户在IDE登录自己github账号后，就可以通过本IDE的图形界面使用git进行版本管理的各种操作（add commit push stash pull fetch checkout等）。

**1.2 项目度量**

整个项目包含以下几个模块：图形界面模块，逻辑流程控制模块，代码结构分析管理模块，运行时虚拟环境连接模块，文本管理项目信息保存管理模块。

**1.2.1 图形界面模块**

图形界面模块相对简单，主要以大量UI控件、布局组成，主要是组件的设定等，逻辑复杂度较低，工作量主要在于编码，估计代码量大概需要700~1000行左右。

**1.2.2 逻辑流程控制模块**

流程控制模块与图像界面模块紧密结合，他将响应用户的交互，验证正确性、将用户交互信息转换为调用数据模块的提供的接口，以及将数据模块的数据安排到指定的前端界面上；主要工作量在于对前后端借口组织的设计，编码，调试，代码量在900-1400左右

其中主要复杂度集中于代码结构分析模块和运行时虚拟环境连接模块，这两个模块需要涉及一些特点算法、数据结构及技术内容，并且为了应对需求变化，在设计方面需要尽量保持解耦和高拓展性。

**1.2.3 代码结构分析模块**

对于代码结构分析模块，该部分工作量主要包括对算法、数据结构的设计和正确性论证，以及编码、调试，代码量估计在3500~4000左右。

**1.2.4 运行时虚拟环境连接模块**

运行时虚拟环境连接模块的主要工作量在于使用多线程并发、同步，以及操作系统中进程之间的通讯和同步；该部分工作量在于对并发结构的设计、编码及调试。代码量估计在1500-2000行。

**1.2.5 项目文本管理模块**

项目文本管理模块则是对各个代码文本文件之间的关系进行存储和管理，需要合理设计数据结构，保存各种必要的信息如编辑状态、文件之间的关系、运行、调试时的各种设置等，主要工作量在于设计数据结构、编码、调试，预估代码量 500-1000左右

1. **开发技术**

**2.1 开发语言**

**2.1.1 Java**

使用java的图形界面库javaFX进行前端界面的开发，后端逻辑等皆采用java实现。

**2.1.2 c++**

大部分虚拟环境的是采用C++进行编写，为了保证IDE能与虚拟环境的良好交互，需要对虚拟环境进行定制化的二次开发，因此C++会作为第二主要语言。

**2.2 技术重点及难点**

**2.2.1 代码文件的实时分析**

在完成语法高亮，自动补全，语法错误提示等功能时，需要对代码文件实时分析。需要使用后缀树、哈希表等数据结构将整个代码文件各个内容进行管理，在用户进行输入时通过一定的算法流程、对数据结构的操作，得到完成以上功能所需的数据。

**2.2.2 与虚拟运行环境实时交互**

代码运行时与模拟器的实时交互。对模拟器返回的数据进行解析，结构化管理，便于前端的获取。与模拟器进程的实时交互涉及到操作系统中，进程之间的同步与通讯。同时为了实现实时获取又要保证IDE内部其他工作的正常进行，IDE内部需要实现多线程及消息队列机制。

**2.2.3 支持项目工程文件**

使用规范结构对项目的文件结构进行保存及管理汇编代码多文件支持，每个文件输出相应.o文件。项目工程多文件的支持必然导致多个.o文件生成。链接错误一般比较古怪。对于常见的找不到符号的错误我们可以简化链接器的错误信息，用对用户友好的形式表达错误。

**2.2.4 词块补全**

根据用户输入以及预留关键字对尚未输入完全的token 生成补全候选列表，补全功能包括：关键字、命令名字、寄存器名、汇编语言关键字、用户自定义变量名。由于汇编指令有基本形式—— 命令 [寄存器名A] [寄存器名B]，其中寄存器名A、B根据命令的不同可选。伪命令如段定义需要额外识别。

**2.2.5 命令格式的列举**

各种指令组合方式很多，我们需要确定所有的命令格式。根据参照指令语法规则，对每条指令的组成进行语法检查，判断语法的正确性。

1. **软件过程模型**

**3.1 软件过程模型**

我们选择Scrum模型，这是一种迭代式增量软件开发过程，常用于敏捷软件开发。

**3.2 模型优势**

**3.2.1 民主式的管理结构，运用团队的力量进行项目管理。**

整个开发任务被具体细分并可视化，开发的情况一目了然，每个人可以在每天的standing meet明确地认领自己要完成的任务。除此之外每位开发人员需要向大家汇报自己开发进度，遇到的问题等，要保证自己的每一步开发进度都能有及时反馈。

去中心化的人员管理，各个开发人员之间相互独立而又保持密切的联系，便于相互之间的沟通与协作，同时也能以集体的力量监督项目进度的推进；项目遇到的各种疑难问题时，能够集思广益，共同解决，有效减低开发阻力。

**3.2.2 快速试错、探索性的开发**

面对需求可能会发生变更、不明确，同时对所开发项目的具体技术并不非常熟悉的情况下，在开发过程中会有很多试探性的工作。灵活性的开发流程能够帮助团队快速适应这种变化的情况，在遇到技术或解决方案不可行的情况时可以快速调整计划、需求。

通过每日的standing meeting，使各个开发人员项目的进度保持透明。使用backlog ，burn down 图使团队能清楚的看见项目的进度变化，及时发现项目开发中遇到的问题。

**3.2.3 轻量级的软件开发流程**

开发流程保持有很多弹性的部分。开发完全面向需求，每日都保证能有一个可运行的版本。遵循 代码优于文档 的原则，保证产品的开发尽量贴近于需求和实际设计，减少文档、流程等对于开发的限制。

1. **人员分工**

表1 小组人员分工明细

|  |  |
| --- | --- |
| 小组人员 | 分工 |
| 李荣颉 | 整体业务建模，基础文本功能，参与代码结构分析模块的设计、编码、调试，对各个模块的集成，整体业务逻辑的设计及编码，对整个系统架构的健壮性优化和运行性能调优。 |
| 张无奇 | 整体业务建模，代码结构分析模块的设计、编码、调试，参与对各个模块的集成整体和业务逻辑的设计、编码、调试，对算法模块的健壮性优化和运行性能调优。 |
| 葛欣杰 | 参与业务建模，项目文件结构管理模块，虚拟运行环境与IDE的连接模块的设计及编码、健壮性优化、性能调优。 |
| 廖子墨 | 负责前端界面设计，参与用户交互逻辑具体设计及编码，参与整体模块集成，用户交互逻辑以及界面布局的更新及优化，整体黑盒测试。 |
| 李清波 | 参与前端界面设计，用户交互逻辑具体设计及编码，参与整体模块集成，前端用户交互逻辑的健壮性优化，单元测试。 |

1. **进度计划**

大体计划如表二和图一、图二所示，详细计划见本文档的[附件1](#附录一)。

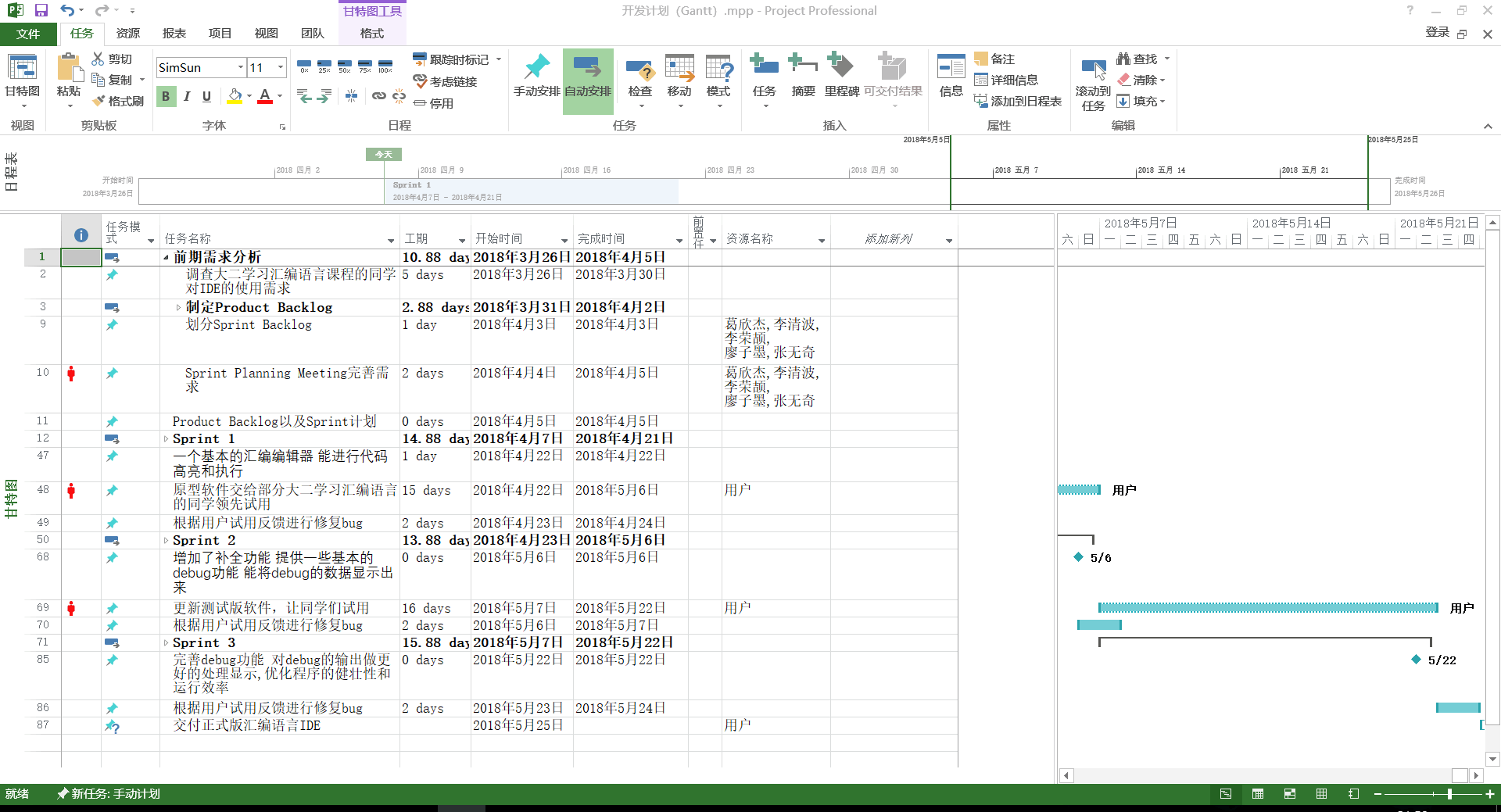
Backlog的标号对应的具体任务对应于附录一中的backlog 表

具体计划甘特图等见文件附件的[项目计划(Gantt).mmp](开发计划（Gantt）.mpp)

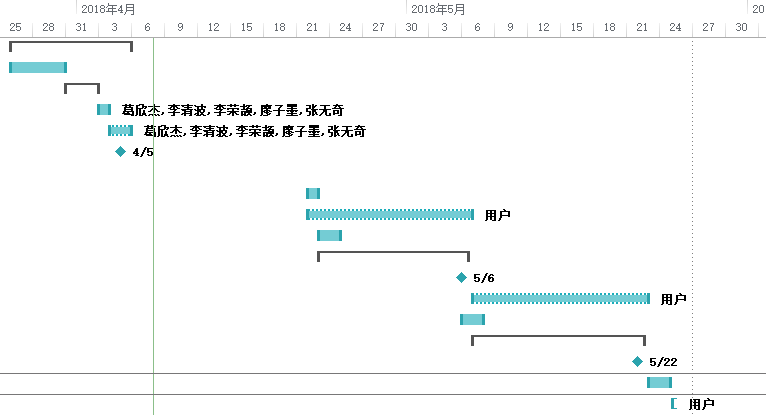
具体需求鱼骨图见文件附件 <需求鱼骨图.svg> (矢量图文件，使用用浏览器打开，可以进行缩放)

表2 Scrum三次迭代整体计划

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 阶段 | 起始日期 | 终止日期 | Sprint内完成的backlog id | 预期 |
| 需求分析 | 2018/03/26 | 2018/04/05 | None | 获得较为完善的需求 |
| Sprint1 | 2018/04/07 | 2018/04/21 | 1,2,3,5,6,7,8\*,  21.\*,22.\*,23. \*,  9,10,11,16,17,  18,25,26,28 | 具有用户交互界面的基本软件，能进行基本的编辑及执行 |
| Sprint2 | 2018/04/23 | 2018/05/06 | 4,12,15,13,14,  24.1,24.2,25,26,  19.1,19.2,20.1,  20.2.2,21,28 | 增加代码补全和debug功能 |
| Sprint3 | 2018/05/07 | 2018/05/22 | 19.3,21,20.\*,  24. \*,25,26,27,  28 | 完善debug功能、加强程序健壮性 |
| 软件交付 | 2018/05/23 | 2018/05/25 |  | 用户体验良好 |



图一 软件开发详细计划概览



图二 开发计划甘特图

**附录：**

附录一：需求一览表：

**编辑功能部分**

**项目文件管理**

1.支持项目工程文件，使用规范结构对项目的文件结构进行保存及管理

2 项目代码文件的添加，删除，新建

3 记录项目代码文件的编辑状态，项目编译运行时虚拟机的环境设定

4 代码进行debug时对加入的断点信息的保存

**文本编辑处理**

5 基本的文件编辑功能 保存 新建 另存为 重命名

6 基于正则表达式的查找 替换

7 维护编辑状态 支持撤销 重做

8 对整个代码文件维护一个结构化的信息，即行、列号管理 （debug 编译时便于指定到具体位置）\*

**代码编辑处理**

9 根据语法的间断 维护一个 token 列表

10 基于按类别进行匹配 将token通过改颜色的方式语法高亮

11 根据用户输入以及预留关键字 对尚未输入完全的token 生成补全候选列表 \*

15 对每个候选项的种类进行标注 并且能根据上下文语义正确性对候选进行排序 \*

12 根据简单的语义规则 进行短上下文的语法错误标注

13 根据语法错误提出具体的错误原因 修改建议

14 扩大判别的距离

**运行调试部分**

**与虚拟机连接、交互**

16 连接qemu dosbox等虚拟环境 基于进程pipe对虚拟环境的输出进行实时监听 输入的传递

17  向系统其它模块提供封装良好的输入输出接口

18 根据项目文件结构 配置信息 在虚拟机环境进行编译 链接 运行

19 与汇编debug程序的交互，将用户的界面交互操作转换为对于的debug指令

19.1 将用户添加的断点 添加到debug程序中

19.2 接受用户的单步执行 继续暂停等操作

19.3 将当前执行的指令与代码文件中具体行数进行对应

20 通过封装与debug交互操作 实时更新、显示机器运行时的寄存器 内存的数据

20.1 根据运行状态实时获取机器内部数据状态

20.2 根据用户指定 显示内存区间的信息

20.2.1 根据指针寄存器

20.2.2 用户输入数字

21 根据装载后的程序情况 将内存中的指令位置与编辑器内代码的在文件中的行列位置建立对应关系

27 根据输入的debug指令 解析debug的数据输出 结构化管理

**前端界面与交互部分**

**交互逻辑**

25 将用户对于前端的操作转换为调用后端的接口

26 各种回调监听类 对前端显示进行更新

28 操作正确性的验证 程序健壮性，运行效率

交互逻辑根据前端 后端内容的更新不断添加 每个scrum周期都有它

界面

22 项目文件管理 编辑

22.1 新建 另存为 保存 删除等 若干menu dialog button。。

22.2 项目文件结构的显示 用户可通过点击切换当前编辑内容

22.3  标签页

23 代码编辑区

23.1 行号标记

23.2 根据行列 定位token 进行字体颜色改变

23.3 根据行信息 对行位置进行高亮

23.4 撤销 重做 替换 查找 等操作的dialog button

24 debug 运行

24.1 各种操作的button'

24.2 程序运行时 各个寄存器 内存内容的显示

24.3 运行编译时各种信息的显示