



# BIT-MiniCC简介

计卫星

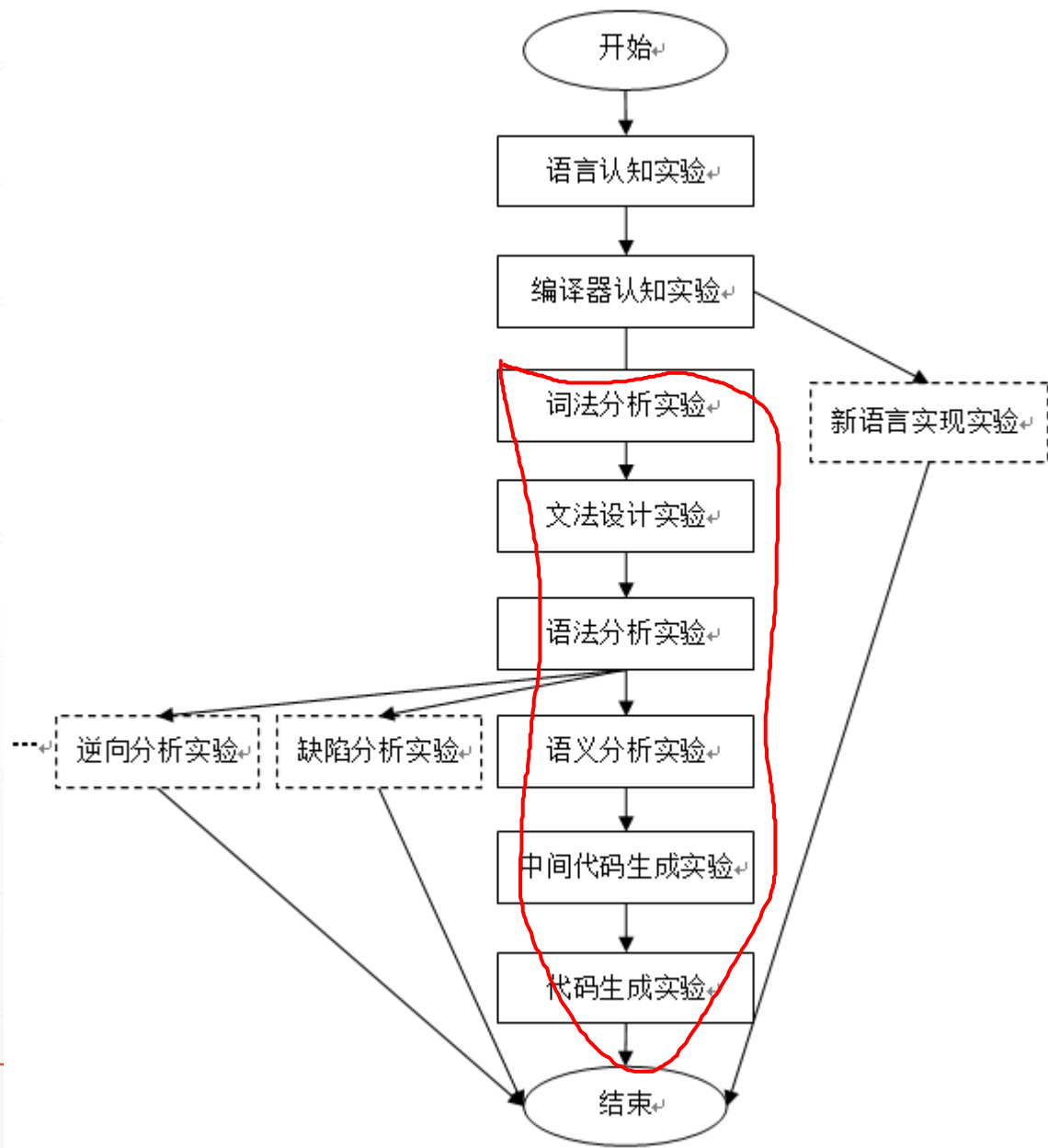
北京理工大学计算机学院

---





# 总体实验框架设计





# 提纲

1. BIT-MiniCC简介
2. 框架介绍
3. 框架使用方法
4. Q&A





# BIT-MiniCC简介

- 课程实验存在的问题
  - 从理论到实践的距离：我听明白了，但是还是不知道怎么实现
  - 从前端到后端的距离：词法分析实现了，但是不怎么好
  - 从理想到现实的距离：老师，我这学期有5门课。。。



大作业A ?

大作业B ?



大作业C ?

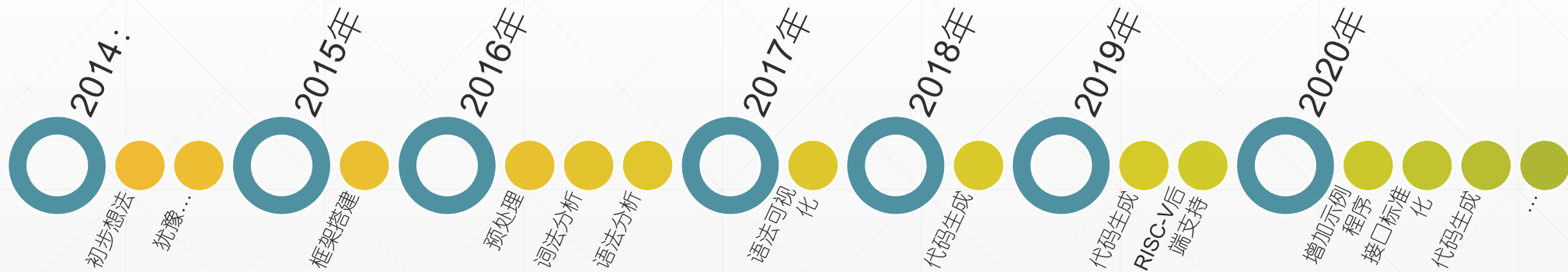
竞赛、实验室项目 ?





# BIT-MiniCC简介

- BIT-MiniCC = **BIT Mini C** Compiler = 一个迷你C语言编译器
  - 源语言 : C语言子集
  - 目标语言: MIPS汇编语言, RISC-V汇编语言, X86汇编语言
  - 宿主语言: Java / Java+C / Java+C# / Java+Python / Java+...
- 开发现状





# BIT-MiniCC简介

- 设计目标
  - 提供一个可参考的运行实例
    - 采用熟悉的语言实现：Java、C#和Python等
    - 内部集成了各个部分的实现
    - 中间处理结果可见：JSON等
    - 每个过程可替换：前面做的不好没关系，可以用现有的实现
  - 除此之外
    - 框架源码公开，但是内部实现不对外公开
    - 代码复制检测： 框架能极大降低检测范围
    - 最后结果测试自动化： 标准输入输出使得自动测试成为可能





# BIT-MiniCC：基于框架的实验设计

- 设计目标1：跨平台可运行参考实例
- 设计目标2：框架源码可见，中间结果可见
- 设计目标3：核心模块不可见，可替换
- 设计目标4：支持多种语言插接
- 设计目标5：充分基于开源库设计并实现

Java C/C++ Python





# BIT-MiniCC：基于框架的实验设计

- 设计目标1：跨平台可运行参考实例
- 设计目标2：框架源码可见，中间结果可见
- 设计目标3：核心模块不可见，可替换
- 设计目标4：支持多种语言插接



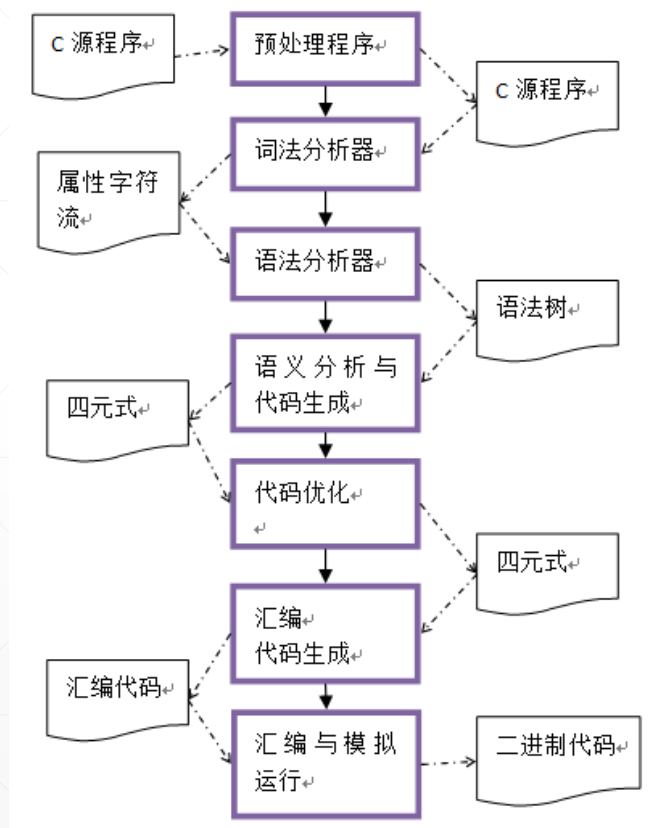






# 框架介绍

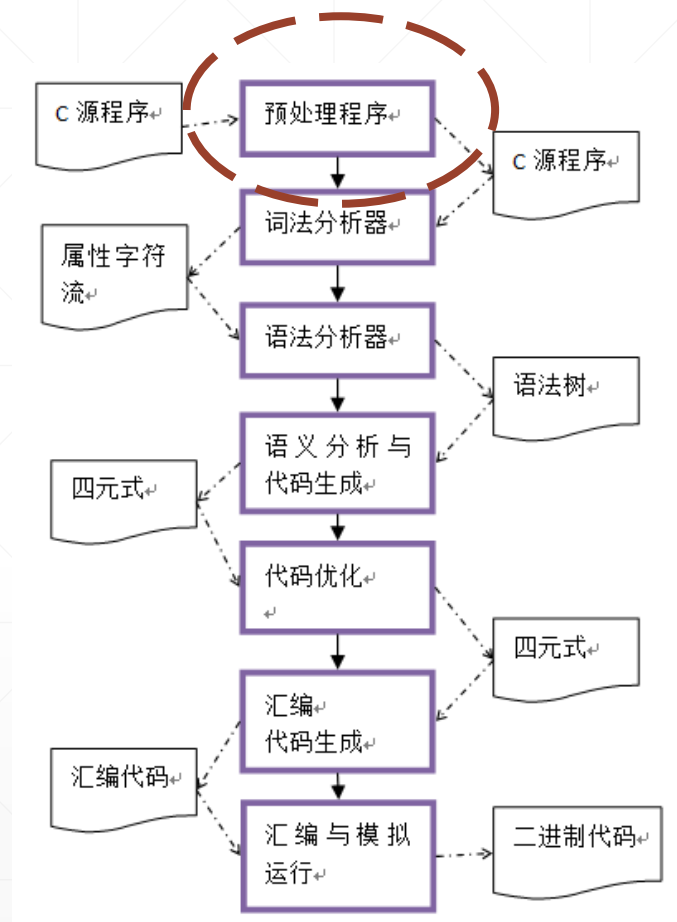
- 特点
  - 输入输出标准化
  - 单个模块可替换
  - 只需关注单个模块的设计
  - 内部模块集成实现
  - 内部实现和自主实现可组合





# 框架介绍

- 预处理
  - 输入：源程序
  - 输出：处理后的源程序
  - 功能
    - 文件包含
    - 宏替换
    - 删除注释
    - 无用空白删除





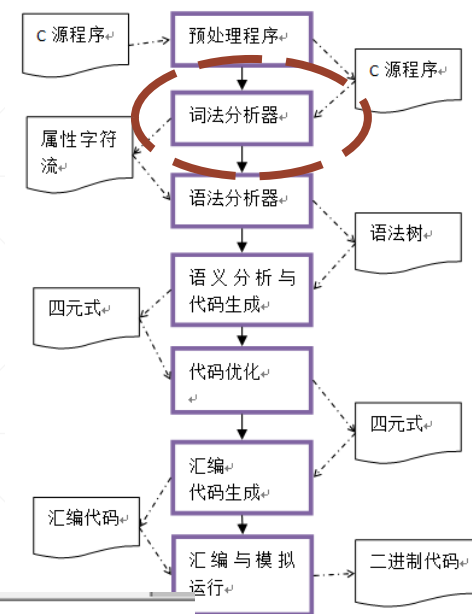
# 框架介绍

- 词法分析
  - 输入：清理后的源程序
  - 输出：属性字符流
  - 功能：根据词法规则识别C语言的全部单词类别，输出相应的属性字符流

```
int sum(int a, int b) { return a + b; }
```



```
[@0,0:2='int',<'int'>,1:0]  
[@1,4:6='sum',<Identifier>,1:4]  
[@2,8:8='(',<'('>,1:8]  
[@3,10:12='int',<'int'>,1:10]  
[@4,14:14='a',<Identifier>,1:14]  
[@5,16:16=',',<','>,1:16]  
[@6,18:20='int',<'int'>,1:18]  
[@7,22:22='b',<Identifier>,1:22]  
[@8,24:24=')',<')'>,1:24]  
[@9,26:26='{',<'{'>,1:26]  
[@10,28:33='return',<'return'>,1:28]  
[@11,35:35='a',<Identifier>,1:35]  
[@12,37:37='+',<'+'>,1:37]  
[@13,39:39='b',<Identifier>,1:39]  
[@14,41:41=';',<'>',1:41]  
[@15,43:43='}',<'>',1:43]  
[@16,0:4='<EOF>',<EOF>,2:0]
```



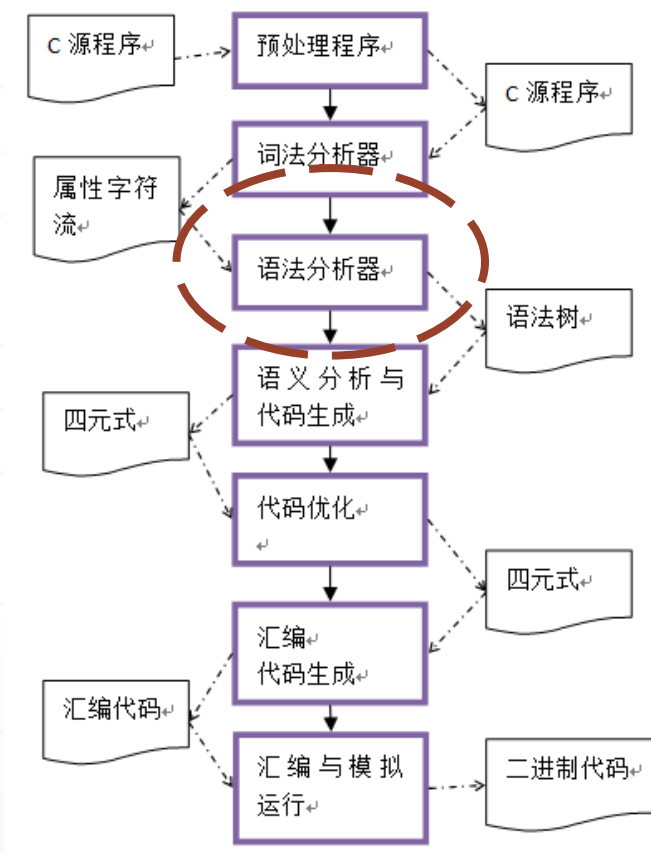
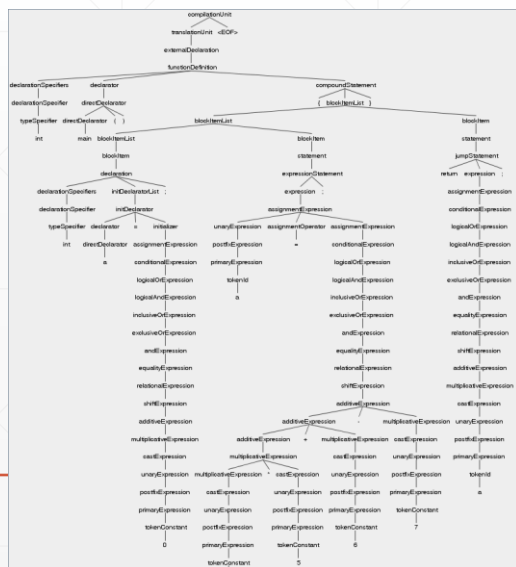


# 框架介绍

## ■ 语法分析

- 输入：属性字符流
- 输出：语法树
- 功能：根据C语言的语法规则，识别输出程序的结构，
- 输出语法错误或者语法树

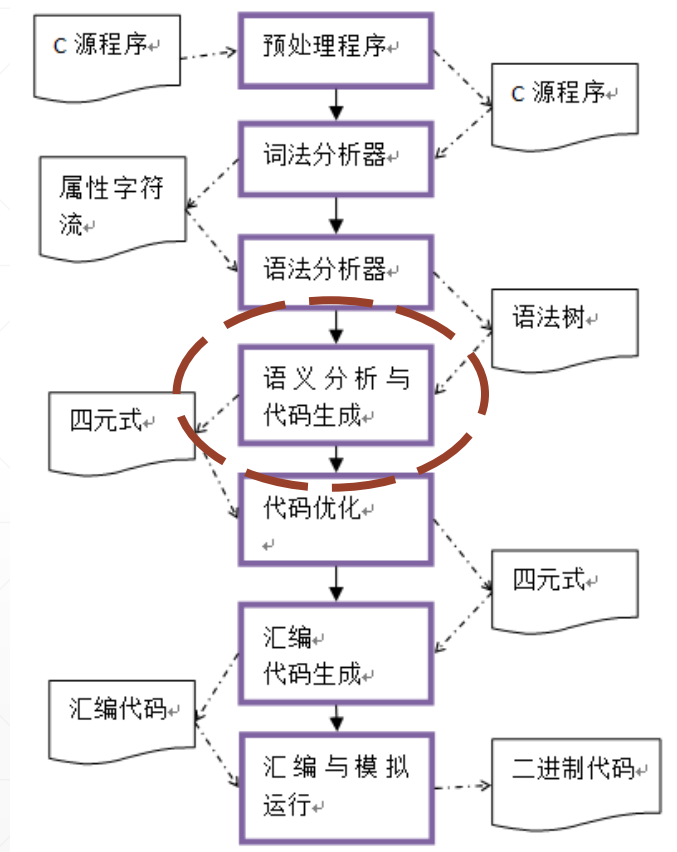
```
[@0,0:2='int',<'int'>,1:0]
[1,4:6='sum',<Identifier>,1:4]
[2,8:8='(',<'('>,1:8]
[3,10:12='int',<'int'>,1:10]
[4,14:14='a',<Identifier>,1:14]
[5,16:16=',',<','>,1:16]
[6,18:20='int',<'int'>,1:18]
[7,22:22='b',<Identifier>,1:22]
[8,24:24=')',<')'>,1:24]
[9,26:26='{',<'{'>,1:26]
[10,28:33='return',<'return'>,1:28]
[11,35:35='a',<Identifier>,1:35]
[12,37:37='+',<'+'>,1:37]
[13,39:39='b',<Identifier>,1:39]
[14,41:41=';',<'>',1:41]
[15,43:43='}',<'>',1:43]
[16,0:4='<EOF>',<EOF>,2:0]
```





# 框架介绍

- 语义分析
  - 输入：语法树
  - 输出：语法树及语义错误
  - 功能
    - 检查符号是否重定义
    - 检查符号的作用域及引用
    - ...



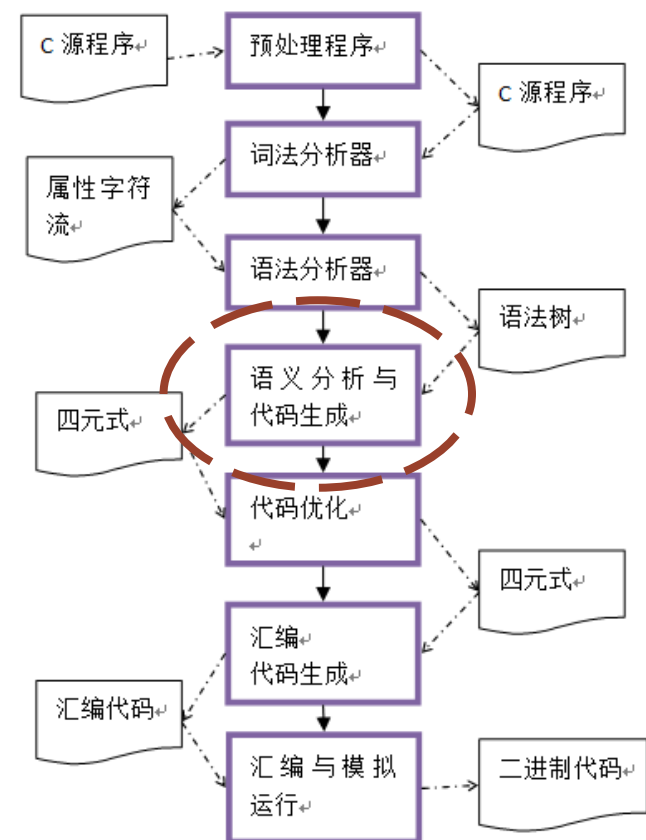


# 框架介绍

- 中间代码

and the statement "a = b + c - d" is:

```
dassign $a (  
  sub i32(  
    add i32(dread i32 $b, dread i32 $c),  
    dread i32 $d))
```





# 框架介绍

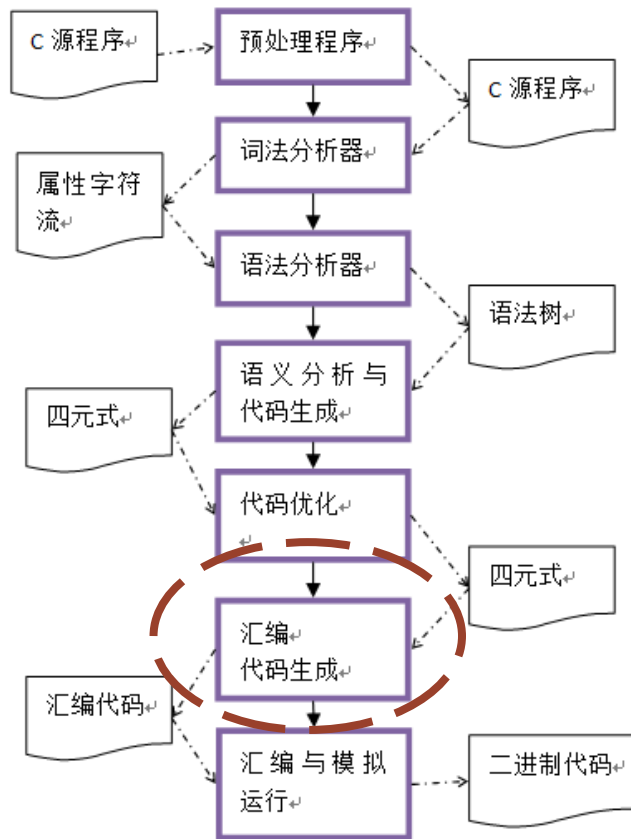
- 代码生成
  - 生成目标机汇编代码

and the statement "a = b + c - d" is:

```
dassign $a (  
  sub i32(  
    add i32(dread i32 $b, dread i32 $c),  
    dread i32 $d))
```



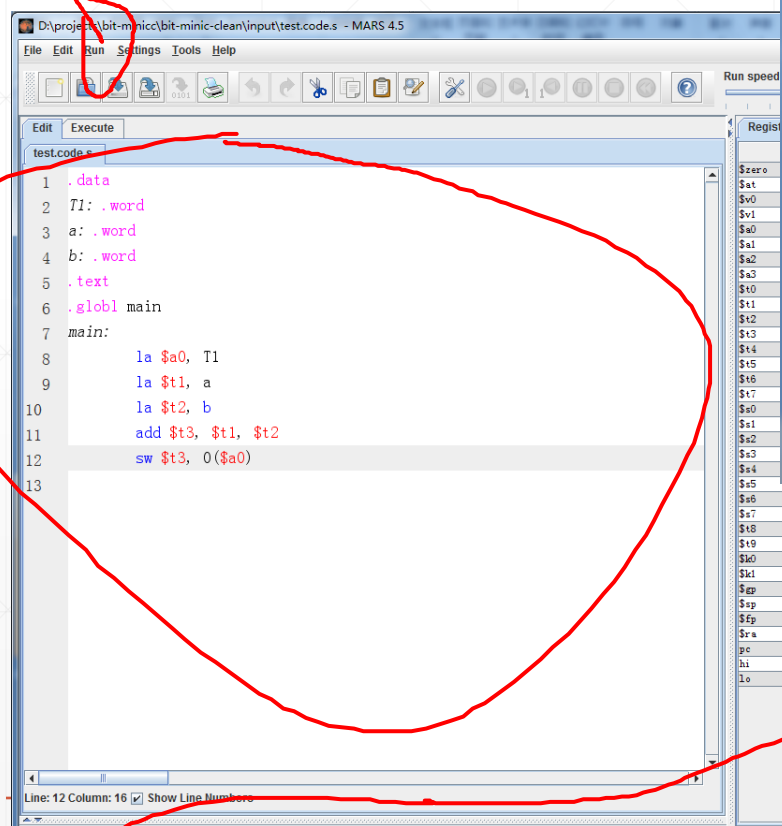
```
1  .data  
2  T1: .word  
3  a: .word  
4  b: .word  
5  .text  
6  .globl main  
7  main:  
8      la $a0, T1  
9      la $t1, a  
10     la $t2, b  
11     add $t3, $t1, $t2  
12     sw $t3, 0($a0)  
13
```



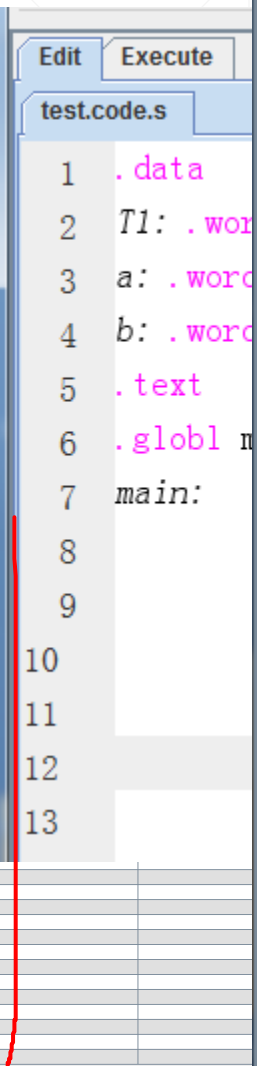


# 框架介绍

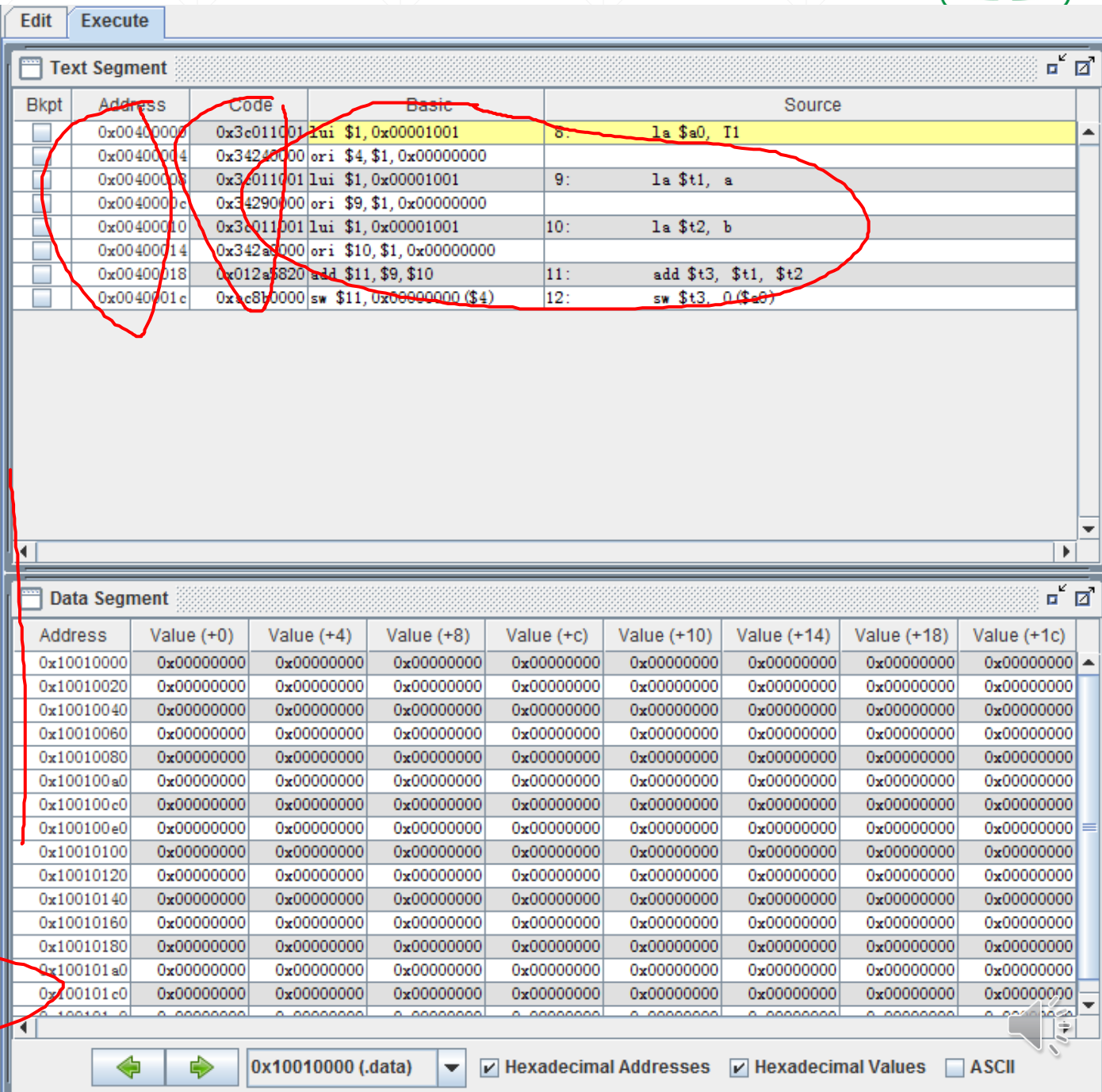
- 模拟运行: MARS



```
1 .data
2 T1: .word
3 a: .word
4 b: .word
5 .text
6 .globl main
7 main:
8     la $a0, T1
9     la $t1, a
10    la $t2, b
11    add $t3, $t1, $t2
12    sw $t3, 0($a0)
13
```



```
1 .data
2 T1: .word
3 a: .word
4 b: .word
5 .text
6 .globl main
7 main:
8
9
10
11
12
13
```



**Text Segment**

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3e011001	lui \$1, 0x00001001	8: la \$a0, T1
	0x00400004	0x34240000	ori \$4, \$1, 0x00000000	
	0x00400008	0x3e011001	lui \$1, 0x00001001	9: la \$t1, a
	0x0040000c	0x34290000	ori \$9, \$1, 0x00000000	
	0x00400010	0x3e011001	lui \$1, 0x00001001	10: la \$t2, b
	0x00400014	0x342a0000	ori \$10, \$1, 0x00000000	
	0x00400018	0x012a0820	add \$t1, \$9, \$10	11: add \$t3, \$t1, \$t2
	0x0040001c	0xc8100000	sw \$t1, 0(\$a0)	12: sw \$t3, 0(\$a0)

**Data Segment**

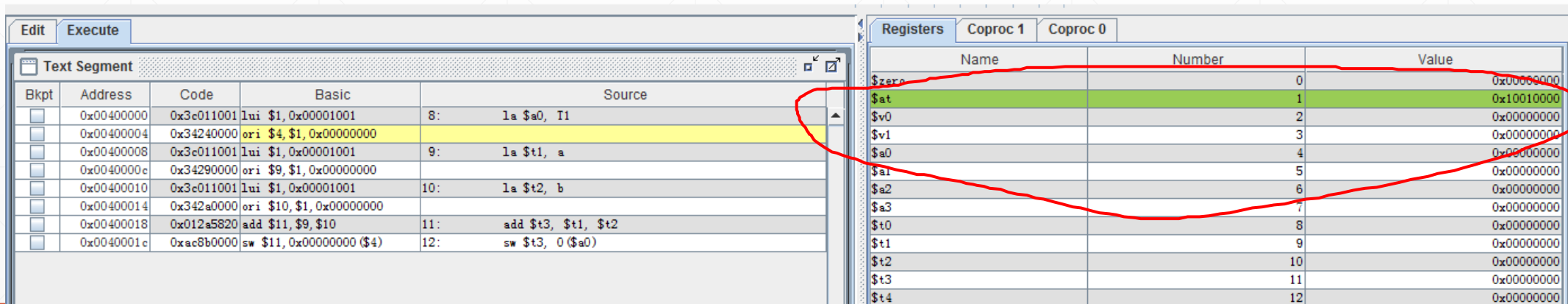
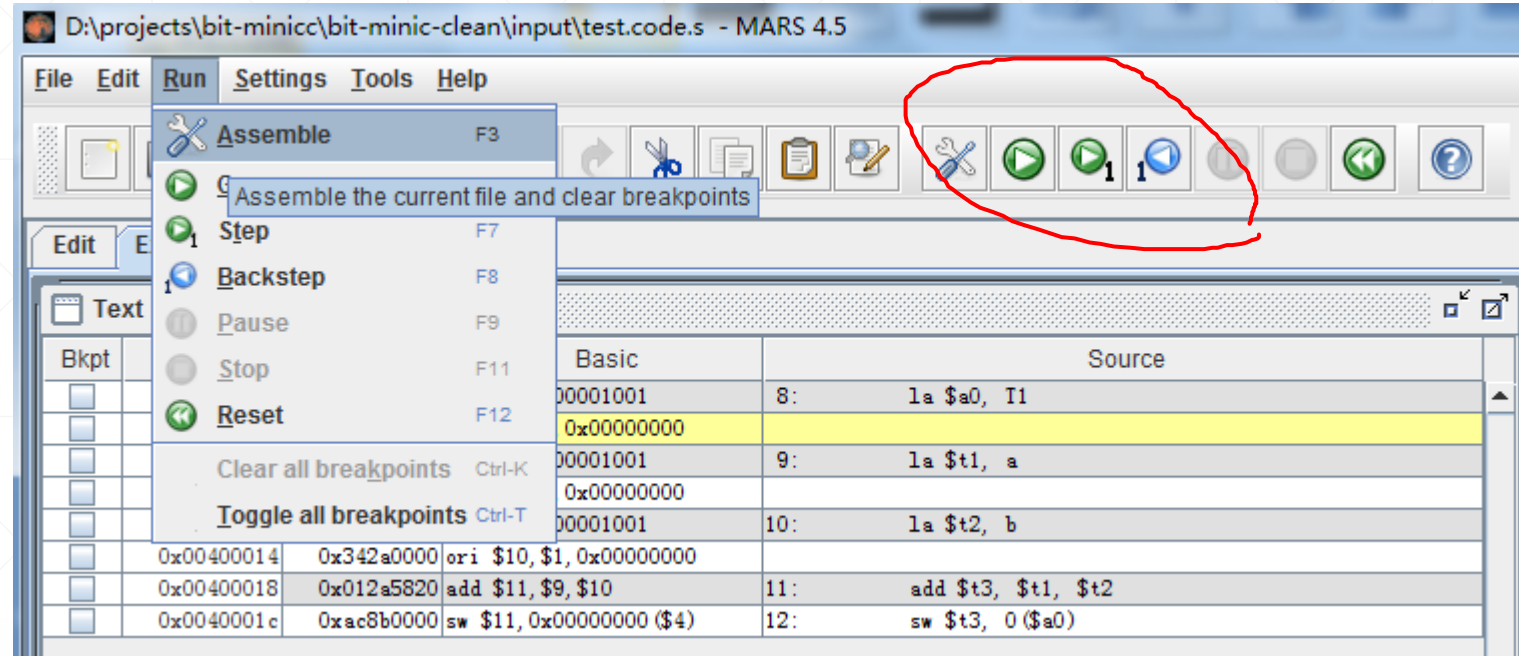
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Hexadecimal Addresses    Hexadecimal Values    ASCII



# 框架介绍

- 模拟运行





# 框架介绍

- MARS相关资料
  - <http://courses.missouristate.edu/KenVollmar/MARS/>

The screenshot shows the Missouri State University website header with the university logo and name. Below the header, there is a search bar and a list of letters (a-z). The main content area features a large image of the planet Mars. To the left of the Mars image is a vertical navigation menu with buttons for Home, Features, Download, License, Papers, Help & Info, and Contact Us. To the right of the Mars image, the text reads: **MARS (MIPS Assembler and Runtime Simulator)**  
*An IDE for MIPS Assembly Language Programming*  
MARS is a lightweight interactive development environment (IDE) for programming in MIPS assembly language, intended for educational-level use with Patterson and Hennessy's *Computer Organization and Design*. Below this text is a Softpedia certification badge that says "100% FREE" and "NO SPYWARE, NO ADWARE, NO VIRUSES". To the right of the badge, there is a quote from Softpedia dated Feb. 2013: "MARS has been tested in the Softpedia labs using several industry-leading security solutions and found to be completely clean of adware/spyware components. ... Softpedia guarantees that MARS 4.3 is 100% FREE, which means it does not contain any form of malware, including spyware, viruses, trojans and backdoors." Below the quote, there is a link: [Download MARS from Softpedia](#) (version on Softpedia may lag behind the version on this page).





# 框架介绍

- 运行环境
  - java JRE  $\geq 1.8$
  - 各操作系统都可以





# 框架介绍

- 基本配置: config.xml
  - skip: 是否跳过该阶段运行
  - type: 模块实现方法
    - java/binary/python
  - path: 路径
    - Binary
    - simulator
  - name: 阶段名称

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <config name="config.xml">
3   <phases>
4     <phase>
5       <phase skip="false" type="java" path="" name="pp" />
6       <phase skip="false" type="java" path="" name="scanning" />
7       <phase skip="false" type="java" path="" name="parsing" />
8       <phase skip="false" type="java" path="" name="semantic" />
9       <phase skip="false" type="java" path="" name="icgen" />
10      <phase skip="false" type="java" path="" name="optimizing" />
11      <phase skip="false" type="java" path="" name="codegen" />
12      <phase skip="false" type="mips" path="" name="simulating" />
13    </phase>
14  </phases>
15 </config>
16
17
```





# 框架介绍

- 内部集成的功能
  - 预处理：注释和宏替换
  - 词法分析：所有单词
  - 语法：所有
  - 语义：正在修改
  - 中间代码：正在修改
  - 优化：暂无
  - 代码生成：正在修改

```
PROGRAM → → → ·FUNCTIONS↵
FUNCTIONS → → → ·FUNCTION ·FLIST↵
FLIST → → → → ·FUNCTION ·FLIST · | · ε↵
FUNCTION → → → ·TYPE ·TKN_ID ·TKN_LP ·ARGS ·TKN_RP ·FUNC_BODY↵
ARGS → → → → ·FARGS ·ALIST · | · ε↵
ALIST → → → → ·TKN_COMMA ·FARGS ·ALIST · | · ε↵
FARGS → → → → ·TYPE ·TKN_ID↵
FUNC_BODY → → → → ·TKN_LB ·STMTS ·TKN_RB↵
STMTS → → → → ·STMT ·STMTS · | · ε↵
STMT → → → → ·EXPR_STMT · | ·RET_STMT↵
EXPR_STMT → → → → ·EXPR ·TKN_SEMICOLON↵
RET_STMT → → → → ·TKN_KW_RET ·EXPR_STMT↵
EXPR → → → → ·TERM ·TLIST↵
TLIST → → → → ·TKN_PLUS ·TERM ·TLIST · | · ε↵
TERM → → → → ·FACTOR ·FLIST↵
FLIST → → → → ·TKN_MUL ·FACTOR ·FLIST · | · ε↵
FACTOR → → → → ·TKN_LP ·EXPR ·TKN_RP · | ·TKN_ID↵
TYPE → → → → ·TKN_INT · | ·TKN_FLOAT↵
```





# BIT-MiniCC使用方法

- 从github下载框架
  - <https://github.com/jiweixing/BIT-MiniCC>

jiweixing / BIT-MiniCC

Unwatch 11 Unstar 51 Fork 44

Code Issues 2 Pull requests 3 Actions Projects 0 Wiki Security Insights Settings

A C compiler framework in Java Edit

Manage topics

41 commits 1 branch 0 packages 0 releases 2 contributors

Branch: master New pull request

Create new file Upload files Find file **Clone or download**

Weixing Ji reengineering		Latest commit f9ec6e2 28 minutes ago
.settings	new version	11 months ago
bin	test input files	11 months ago
doc	documatation	4 years ago
lib	reengineering	28 minutes ago
src/bit/minisys/minicc	reengineering	28 minutes ago

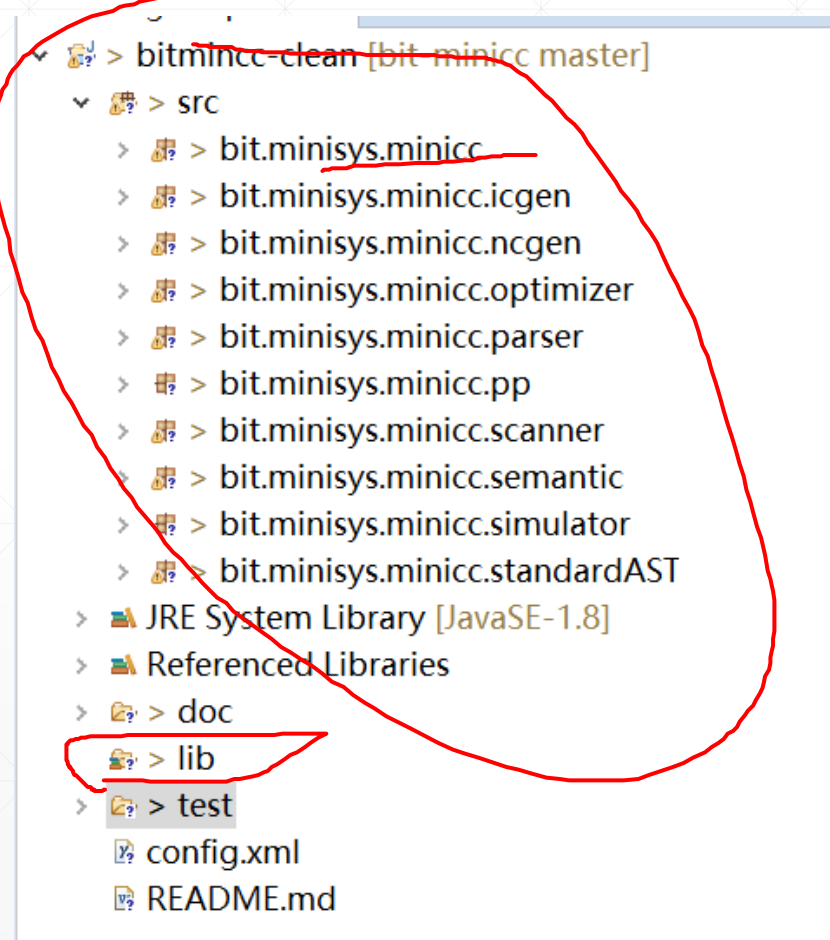
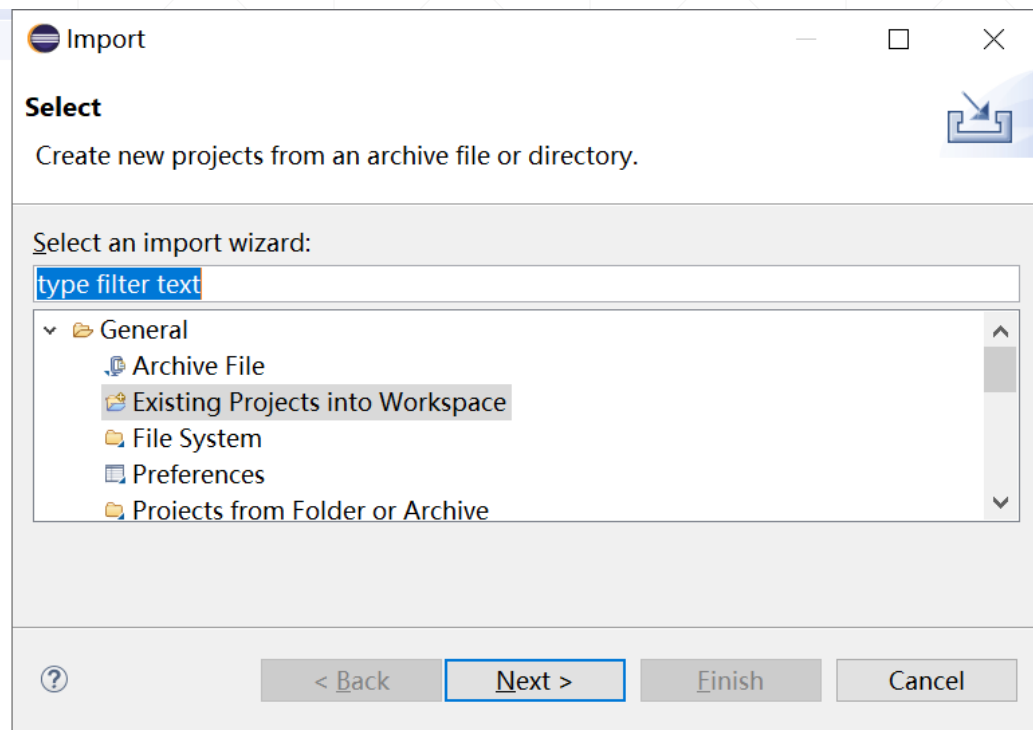






# BIT-MiniCC使用方法

- 从Eclipse里面导入项目

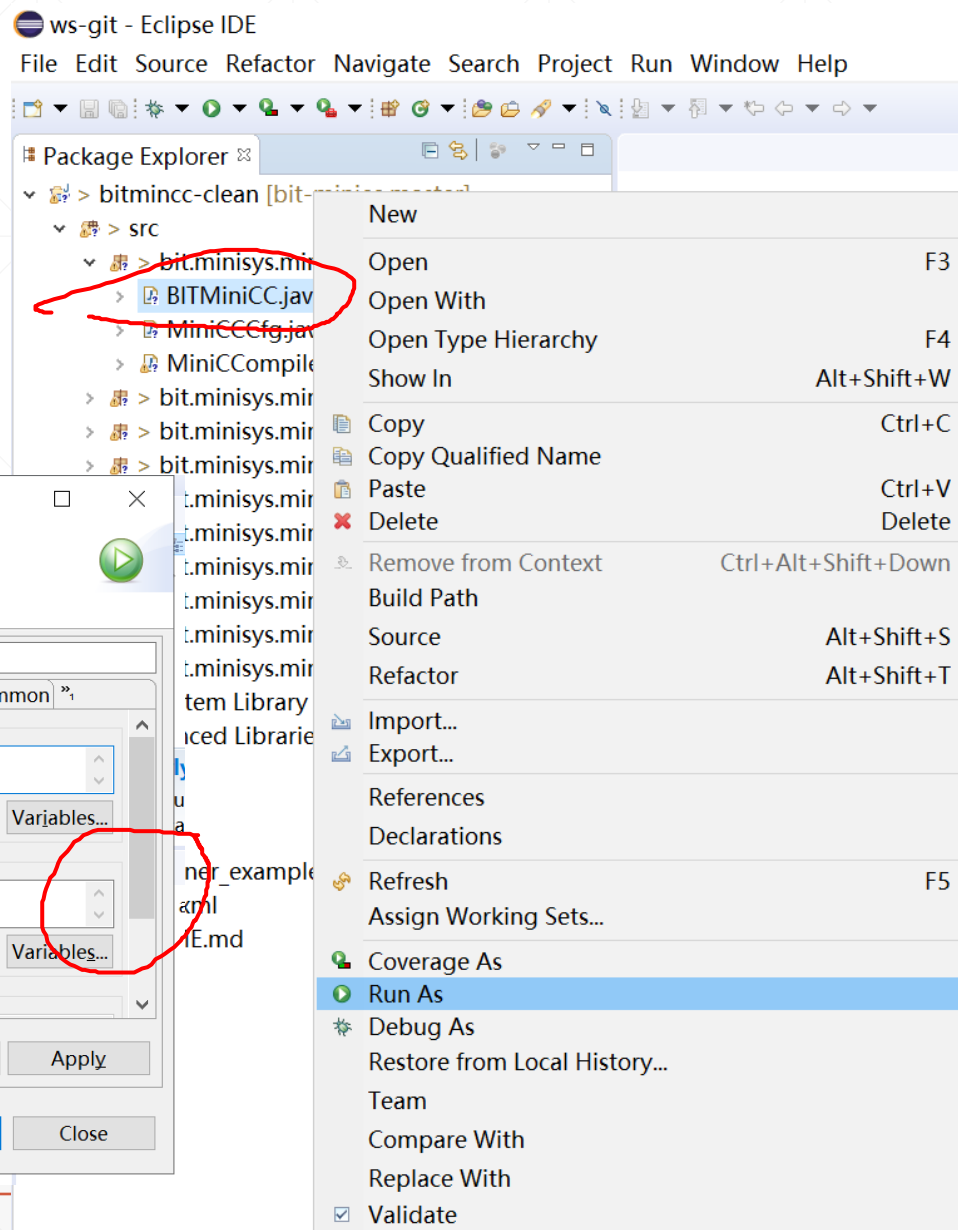
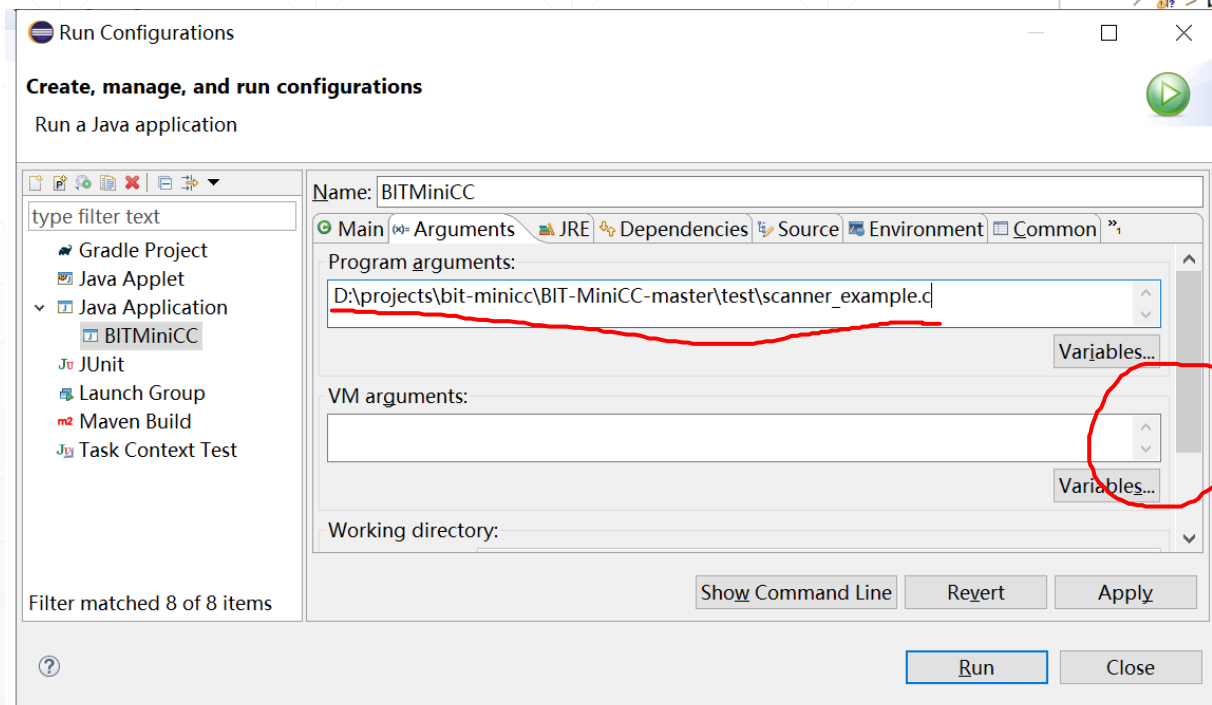






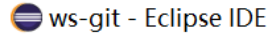
# BIT-MiniCC使用方法

## ■ 设置输入测试用例




1 Java Application Alt+Shift+X,  
Run Configurations...






File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

▼  > bitmincc-clean [bit-

- ✖ SFC
- ✖ bit.minisys.mi
- ✖ BITMiniCC.jav



rogram Files\Java\j

9 4 9

Refactor

Import


Import...

Export

Export...

## Reference

Declarat

 Refresh

### Assign W

Coverage

Run As

 Debug A

Debug A  
Restore t

RESTORE

 Debug A

Restore t



# BIT-MiniCC使用方法

## ■ 源代码树

```
ExampleScanner.java
24
25 public class ExampleScanner implements IMiniCCScanner {
26
27     private int lIndex = 0;
28     private int cIndex = 0;
29
30     private ArrayList<String> srcLines;
31
32     private HashSet<String> keywordSet;
33
34     public ExampleScanner(){
35         this.keywordSet = new HashSet<String>();
36         this.keywordSet.add("int");
37         this.keywordSet.add("return");
38     }
39
40     private char getNextChar() {
41         char c = Character.MAX_VALUE;
42         while(true) {
43             if(lIndex < this.srcLines.size()) {
44                 String line = this.srcLines.get(lIndex);
45                 if(cIndex < line.length()) {
46                     c = line.charAt(cIndex);
47                     cIndex++;
48                     break;
49                 }else {
50                     lIndex++;
51                 }
52             }
53         }
54     }
55 }
```

```
> bitmincc-clean [bit-minicc master]
> src
> bit.minsys.minicc
> bit.minsys.minicc.icgen
> bit.minsys.minicc.ncgen
> bit.minsys.minicc.optimizer
> bit.minsys.minicc.parser
> bit.minsys.minicc.pp
> bit.minsys.minicc.scanner
> ExampleScanner.java
> IMiniCCScanner.java
> bit.minsys.minicc.semantic
> bit.minsys.minicc.simulator
> bit.minsys.minicc.standardAST
> JRE System Library [JavaSE-1.8]
> Referenced Libraries
> doc
> lib
> test
> config.xml
> README.md
```





# BIT-MiniCC使用方法

```
> bitmincc-clean [bit-minicc master]
> src
> bit.minisys.minicc
> bit.minisys.minicc.icgen
> bit.minisys.minicc.ncgen
```

```
<?xml version="1.0" encoding="UTF-8"?>
<config name="config.xml">
  <phases>
    <phase>
      <phase skip="false" type="java" path="" name="preprocess" />
      <phase skip="false" type="java" path="bit.minisys.minicc.scanner.ExampleScanner" name="scan" />
      <phase skip="false" type="java" path="bit.minisys.minicc.parser.ExampleParser" name="parse" />
      <phase skip="true" type="java" path="" name="semantic" />
      <phase skip="true" type="java" path="" name="icgen" />
      <phase skip="true" type="java" path="" name="optimize" />
      <phase skip="true" type="java" path="" name="ncgen" />
      <phase skip="true" type="mips" path="" name="simulate" />
    </phase>
  </phases>
</config>
```

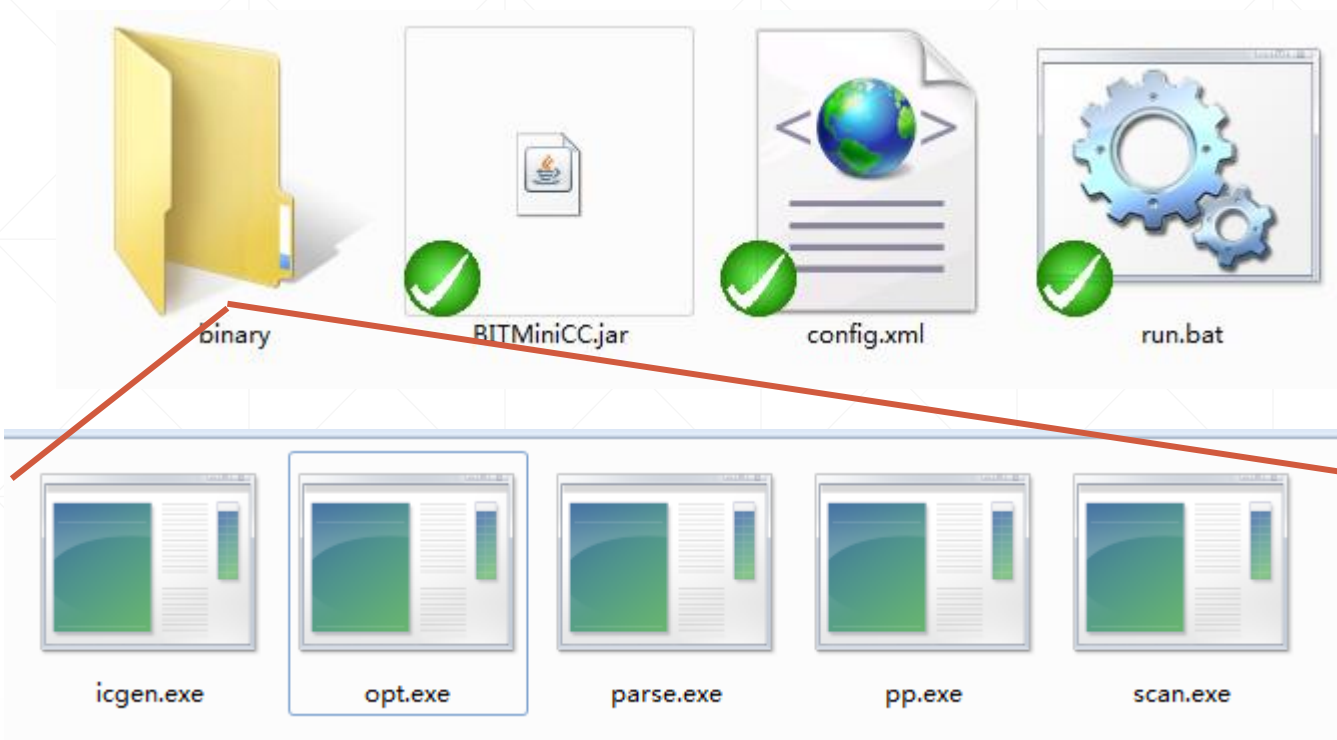
```
> test
> config.xml
> README.md
```





# 扩展方法

- C/C++/C#



```
<?xml version="1.0" encoding="UTF-8" ?>
- <config name="config.xml">
- <phases>
- <phase>
  <phase skip="false" type="binary" path="D:\projects\bit-minicc\bit-minic-clean\run\binary\pp.exe" name="pp" />
  <phase skip="false" type="binary" path="D:\projects\bit-minicc\bit-minic-clean\run\binary\scan.exe" name="scanning" />
  <phase skip="false" type="java" path="" name="parsing" />
  <phase skip="false" type="java" path="" name="semantic" />
  <phase skip="false" type="java" path="" name="icgen" />
  <phase skip="false" type="java" path="" name="optimizing" />
  <phase skip="false" type="java" path="" name="codegen" />
  <phase skip="false" type="java" path="" name="simulating" />
</phase>
</phases>
</config>
```





**Q & A?**

**[jwx@bit.edu.cn](mailto:jwx@bit.edu.cn)**

