# ARM汇编代码认证测试要求

## 静态测试

## Part 1：人工走查

表格 A. 1 – 人工走查

|  |  |
| --- | --- |
| **技术/措施** | **测试工具** |
| 1. 编码标准 | 人工分析 |
| 2. 编码风格指南 |
| 3. 走查/设计审核 |

实施方案：

以设计人员编码规范和设计文档为依据，测试专家对描述的功能、对照编写的代码做检查。

## Part 2：静态分析

表格 A. 2 – 静态分析

|  |  |
| --- | --- |
| **技术/措施** | **测试工具** |
| 1. 静态函数结构图 | 人工分析+自制测试工具  (使用指针、子函数参数等在汇编中不适用) |
| 2. 限制使用递归 |
| 3. 无条件跳跃 |
| 4. 限制规格和功能复杂程度子程序和方法 |
| 5. 函数扇入和扇出分析 |

实施方案：

采用自制的静态分析工具，通过对函数的分析，获取函数的静态调用关系、函数的参数、函数的扇入和扇出、无条件跳转关系。

## Part 3：数据流，控制流图

表格 A. 3 – 人工走查

|  |  |
| --- | --- |
| **技术/措施** | **测试工具** |
| 1. 数据流图 | 自制测试工具 |
| 2. 控制流图 |

实施方案：

全局变量定义和引用关系；每个模块的分支图。

## 动态测试

## Part 4：单元测试

表格 A. 4 – 单元功能和黑箱测试

|  |  |
| --- | --- |
| **技术/措施** | **测试工具** |
| 1. 通过边界值分析测试执行用例 | 人工分析+自制测试工具  (复合条件在汇编中不适用) |
| 2. 等价类和输入划分测试 |
| 3. 语句覆盖率 |
| 4. 分支覆盖率 |
| 5. 提供故障注入测试手段 |

实施方案：

通过自制测试工具，在运行中记录目标指令的执行情况，通过对指令语义的分析，得出目标码的行和分支覆盖率。(详细参见 关键技术)

## Part 5：集成测试

表格 A. 5 – 集成测试(*注释：汇编模块的调用接口集成测试*）

|  |  |
| --- | --- |
| **技术/措施** | **测试工具** |
| 1. 模块调用覆盖率 | 人工分析+自制测试工具 |

实施方案：

通过自制测试工具，能够确认调用模块情况。(详细参见 关键技术)

# 关键技术

通过自研测试工具，能够对预期输入和输出进行检测、能够记录指令的执行情况、保存运行中堆栈的信息。从而实现动态函数调用关系的结构图，获得执行函数的路径，并生成行和分支的覆盖率。



仿真测试工具(自制)

图1 ARM测试工具

## 目标码覆盖率

目标码覆盖率统计工具由解释器、执行器、分析器以及生成器组成，如图2所示。

图2 目标码覆盖率工具框图

解析器对目标文件二进制格式的解析，生成反汇编信息、行号表信息以及对分支语句统计表。分支语句统计表包含目标码地址是否是分支、以及分支总数等信息。

执行器通过一次或多次执行目标文件，将目标文件执行一次或者多次的数据保存在运行信息表中。该表主要包含目标码地址在一次或者多次执行后，该目标码的地址执行的次数。

分析器通过将分支语句的统计表和运行信息表统合，能够对目标码地址语句、分支进行合并，生成基于文本格式的覆盖率统计表。

生成器是将反汇编信息同覆盖率统计表结合生成基于html格式的覆盖率报告。生成的html格式报告，如图3、4所示。

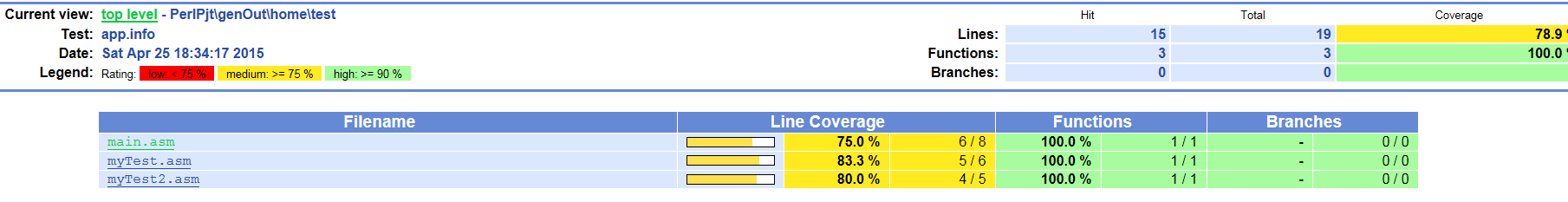


图3 覆盖率统计

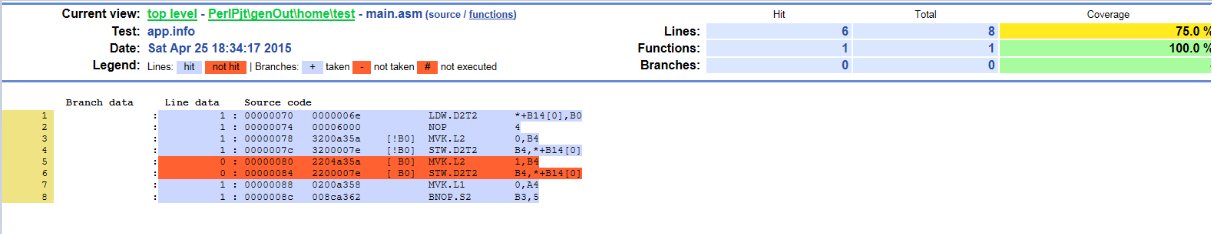


图4 单文件覆盖率详细

自动化测试脚本通过事先配置脚本信息，将自动执行多个配置的测试用例，并生成测试报告。

通过结果路径分析、测试用例结果、以及目标码覆盖率，便于分析未被测试覆盖的代码，最终进行测试用例的修正、补充。

## 故障注入手段

故障注入器由故障注入模型、故障注入点、故障库以及故障分析组成，如图5所示。

图5 故障注入器

故障注入工具通过新建或导入已存在的故障库建立故障注入模型，将用户选择的一个或多个故障注入点通过虚拟目标机提供的虚拟接口注入到虚拟目标机中，同时将故障注入的数据及注入的成功状态保存在故障库中。通过对故障的分析生成故障报告，对可能造成系统故障的各种因素进行分析，分析出系统故障状态的原因，如图6所示。

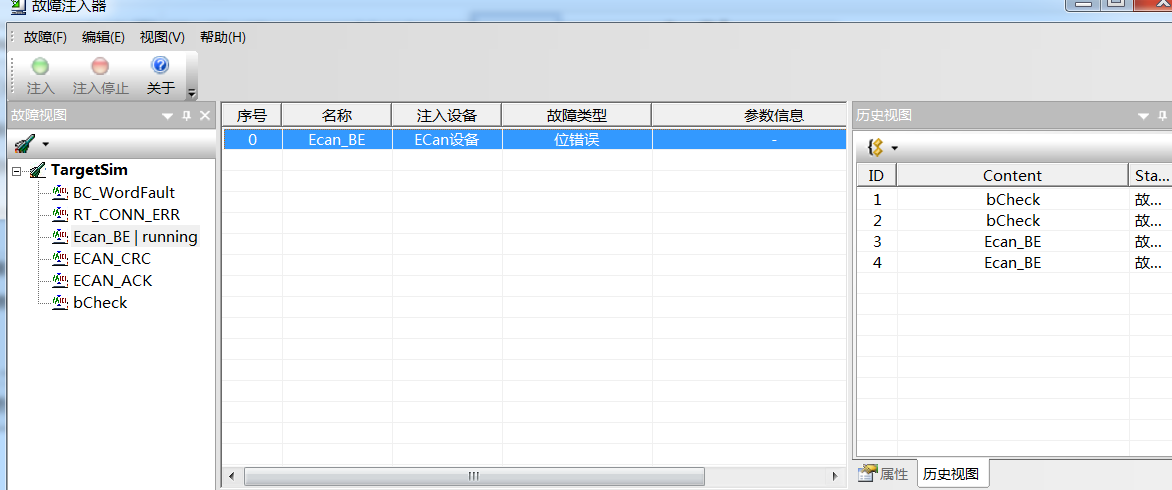


图6 故障注入的演示效果图