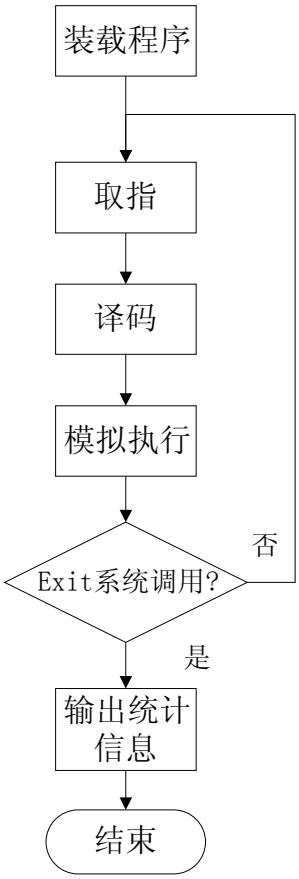
SimpleScalar 修改简要提示

我们主要针对 sim-safe 模拟器进行一些介绍。

SimpleScalar 提供的一系列模拟器中，简单指令级模拟器 sim-safe 的模拟流程如下：



其中主要环节的过程如下：

1． 取指

sim-safe 调用 MD_FETCH_INST 宏，以程序计数器（以下简称 PC 寄存器）值为索引，在 sim-safe 模拟的内存空间中取出下一条指令。

2． 译码

sim-safe 调用 MD_SET_OPCODE 宏完成译码过程。sim-safe 曾在初始化时调用 md_init_decoder 函数初始化指令译码表，能够识别的指令都在这个指令译码表中有相应的下标。译码结束后，就能够得到该指令对应的数组下标，并能够获取这个数组下标内的 OP 值。

3． 模拟执行

负责模拟指令执行的文件(sim-safe.c)中对 DEFLINK、CONNECT 和 DEFINST 的定义如下：

```
#define DEFINST(OP,MSK,NAME,OPFORM,RES,FLAGS,O1,O2,O3,I1,I2,I3,I4)   \
```

```
          case OP:
      \
            SYMCAT(OP,_IMPL)                                                    \
            break;
#define DEFLINK(OP,MSK,NAME,MASK,SHIFT)                                  \
          case OP:                                                           \
            panic("attempted to execute a linking opcode");
#define CONNECT(OP)
#include "machine.def"
```

其中，DEFLINK、CONNECT、DEFINST 和 OP_IMPL 宏在指令模板 machine.def 文件中定义，这些宏的作用可以参看程序或 SimpleScalar 的文档。SYMCAT 宏的作用就是连接 OP 和_IMPL。 sim-safe 在译码过程结束后，已解析出取指阶段取得的指令对应的 OP 值，指令的模拟执行过程就是执行指令模板中对应该 OP 的 OP_IMPL 中的语句。如果 OP_IMPL 中的语句没有改变 PC 的值，则 PC 的值为程序顺序执行的下一地址；否则 PC 值为 OP_IMPL 语句中所赋的值。模拟执行结束后，即返回到取指过程。

应用程序结束时使用软件中断指令调用 exit 系统调用。sim-safe 模拟执行到 exit 系统调用，就结束当前应用程序的模拟，返回 sim-safe 的主流程。这时 sim-safe 输出应用程序动态执行的指令数、内存访问次数和模拟器执行时间等数据，并结束运行。

下面我们针对 32 位定长指令系统在 SimpleScalar 代码中的主要修改进行一些基本的说明。其它方面大家请根据自己的具体情况修改相关代码（例如 sim-safe.c 源代码）。

1．在 simple-arm 目录下建立新的文件夹：例如，myisa。

2．将 target-arm 目录下的文件 copy 到 myisa 目录下，并修改文件名。
例如：arm.[h,c,def] 修改为 myisa.[h,c,def]。

3．修改或重新编写相应的文件，主要是指令模板 myisa.def、系统调用 syscall.c 和装载模块 loader.c。

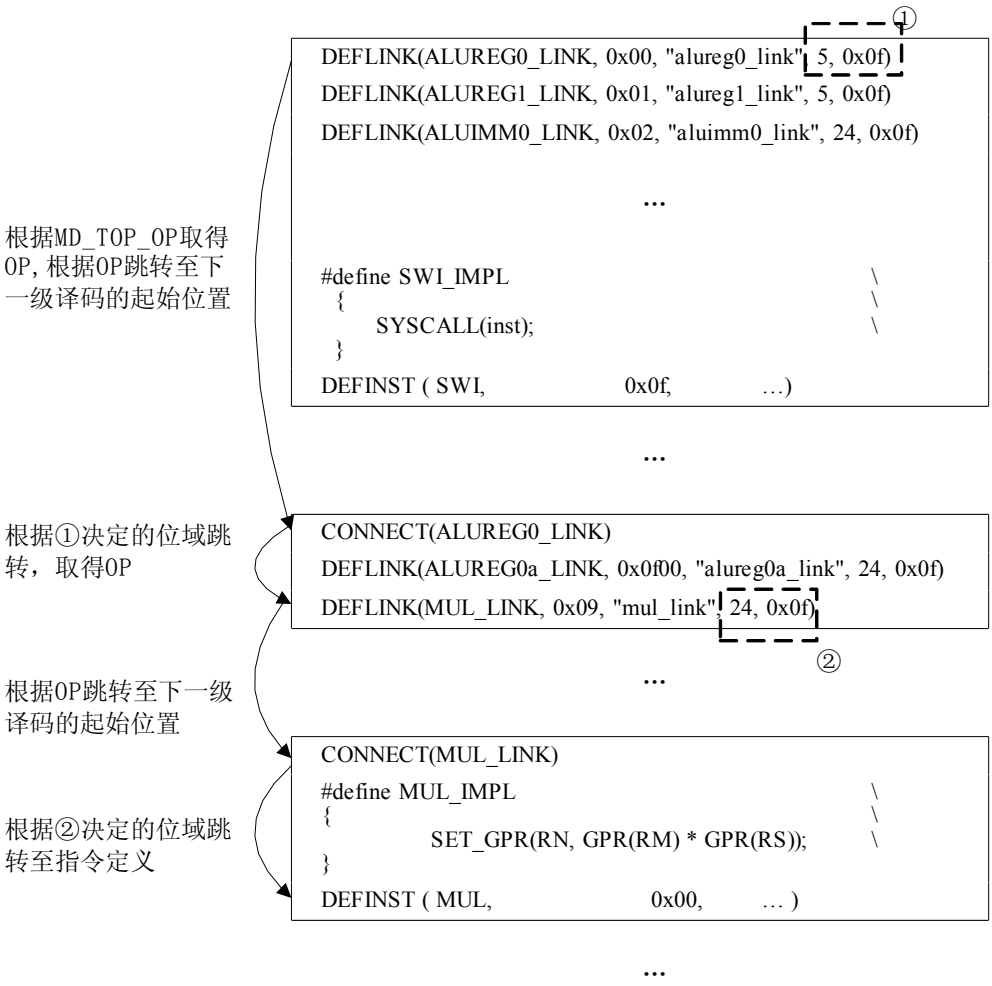
3．1 关于指令模板

指令模板文件 machine.def 由一系列下述三种形式的宏组成：

- DEFINST(OP,MSK,NAME,OPFORM,RES,FLAGS,O1,O2,O3,I1,I2,I3,I4)
- DEFLINK(OP,MSK,NAME, SHIFT,MASK)
- CONNECT(OP)

具体含义可以参阅代码或给出的 SimpleScalar 相关文档。

下面的图示以一条指令的译码过程为例，说明 SimpleScalar 利用指令模板进行译码的机制。



3．2关于系统调用的说明

系统调用是操作系统提供给用户程序的接口，用户程序通过系统调用来使用操作系统提供的多种服务，如读、写文件等。对于应用级模拟器，由于并不支持操作系统的运行，故在模拟执行软件中断指令时不能直接模拟该指令本身的语意，而是需要采取特殊的方式模拟，以保证完成所需的操作（如屏幕输出、读写文件等），并保证在返回到软件中断下一条指令时程序运行的状态（各寄存器及内存中存储的数据）与应有状态一致。

SimpleScalar 中，对指令译码后，如果判断为软件中断指令，则调用 sys_syscall 函数来进行系统调用处理，该函数在 syscall.c 中定义，具体实现如下：

首先获得系统调用号。之后模拟器根据具体的系统调用，从模拟的存储空间（包括寄存器、内存）中得到相应的参数。接下来模拟器调用宿主机系统提供的等价的系统调用。最后将返回值和结果从模拟器自身的存储空间中复制到模拟的存储空间。

在本次实习中，我们提供了两个虚拟的系统调用，一个用来在屏幕上显示，一个用来标志程序终止运行。只需要在指令模板中识别出是系统调用，并在 syscall.c 中修改 sys_syscall 这个函数即可。

3．3关于装载模块的说明

给定一个应用程序的二进制可执行文件，模拟器需要按应用程序二进制接口的约定将该文件读入所模拟的内存空间，使其可以运行。

这部分主要需要修改 loader.c 文件。

在本次实习中，我们给出了一个非常简单的可执行文件的文件格式，按照这一格式，可以知道需要读入的文件长度以及程序开始的地址。接下来将程序读入到模拟器模拟的内存空间（可使用 mem_bcopy 函数，参见 memory.h/c），并给 ld_prog_entry 和 PC（程序计数器）、以及堆栈寄存器赋初值。