RVBacktrace

组件简介

一个极简的RISC-V栈回溯组件。

组件源码

组件源码：https://github.com/Yaochenger/RvBacktrace

组件功能

1.在需要的地方调用组件提供的唯一API，开始当前环境的栈回溯

2.支持输出addr2line需要的命令，使用addr2line进行栈回溯

3.支持结合反汇编，栈回溯信息图表化

TODO List:

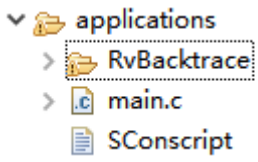
1. 支持打印指定线程的栈回溯信息
2. 支持对接RT_ASSERT
3. 支持输出更多的符号信息
4. 支持文件跳转

添加组件

当前组件暂未做成软件包，所以这里介绍一下临时的使用方法：

首先下载源码，为了方便下文我以RT-Studio环境使用举例：

组件的根目录已经存在`Scons`脚本，所以可以直接放在`applications`目录下：



修改applications目录下的Scons脚本，示例如下：

```
import rtconfig
from building import *

cwd = GetCurrentDir()
src = Glob('*.c')

CPPDEFINES=[]
CPPPATH = [cwd]

group = DefineGroup('Applications', src, depend = [''], CPPPATH = CPPPATH,
CPPDEFINES=CPPDEFINES)

list = os.listdir(cwd)
for item in list:
    if os.path.isfile(os.path.join(cwd, item, 'SConscript')):
```

```
        group = group + SConscript(os.path.join(item, 'SConscript'))

Return('group')
```

完成上述工作，只要组件参与编译且编译没问题，组件添加成功。

使用示例

当前组件C的内容极其简单，同时对用户仅提供单个API：void rvbacktrace(void),用户在需要的地方调用该API就可以将当前的调用栈信息输出，示例（下文演示为HPM6750）：

1.在示例代码适当位置调用rvbacktrace

```c
int main(void)
{
    extern void rvbacktrace(void);
    rvbacktrace();
    return 0;
}
```

2.运行代码，终端输出调用栈信息

```
 \ | /
- RT -     Thread Operating System
 / | \     5.0.2 build Aug 30 2024 14:14:30
 2006 - 2022 Copyright by RT-Thread team

---- RV_Backtrace Call Frame Start: ----
###Please consider the value of ra as accurate and the value of sp as only for reference###
[0]Stack interval :[0x000000000108ea58 - 0x000000000108ea68]  ra 0x0000000008000ffbc pc 0x0000000008000ffb8
[1]Stack interval :[0x000000000108ea68 - 0x000000000108ea78]  ra 0x0000000008000fe7c pc 0x0000000008000fe78
[2]Stack interval :[0x000000000108ea78 - 0x000000000108ea88]  ra 0x0000000080010082 pc 0x0000000008001007e
[3]Stack interval :[0x000000000108ea88 - 0x000000000108ea98]  ra 0x0000000008000688e pc 0x0000000008000688a
[4]Stack interval :[0x000000000108ea98 - 0x00000000deadbeef]  ra 0x0000000001086a7c pc 0x0000000001086a78
---- RV_Backtrace Call Frame End:----

addr2line -e rtthread.elf -a -f 8000ffb8 8000fe78 8001007e 8000688a 1086a78

msh >
```

3.栈回溯信息直观化

可以看到的是当前shell输出的信息没有符号信息，不太直观，当前组件提供了两种方法：1.使用addr2line工具 2.结合返汇编文件中的信息输出调用栈符号。当然还有更好的想法，已经在路上了.接下来详细介绍这俩中方法的使用

3.1addr2line工具

可以看到shell输出的信息中包含addr2line需要的信息，我们将其拷贝，然后在当前rtthread.elf目录下使用该工具，我的环境中在wsl有该工具，则使用该工具打开，打开后将之前复制的信息拷贝至终端执行，输出如下：

```
           :/mnt/d/RT-ThreadStudio/workspace/HPM6750/flash_debug$ addr2line -e rtthread.elf -a -f 8000ff24 8000fe78 8000fffe 8000688a 1086a78
0x8000ff24
rv_backtrace_fno
D:\RT-ThreadStudio\workspace\HPM6750\flash_debug/../applications/RvBacktrace/src/rv_backtrace_fno.c:51
0x8000fe78
rvbacktrace
D:\RT-ThreadStudio\workspace\HPM6750\flash_debug/../applications/RvBacktrace/src/rv_backtrace.c:15
0x8000fffe
main
D:\RT-ThreadStudio\workspace\HPM6750\flash_debug/../applications/main.c:25
0x8000688a
main_thread_entry
D:\RT-ThreadStudio\workspace\HPM6750\flash_debug/../rt-thread/src/components.c:199
0x01086a78
rt_thread_self
D:\RT-ThreadStudio\workspace\HPM6750\flash_debug/../rt-thread/src/thread.c:367
```

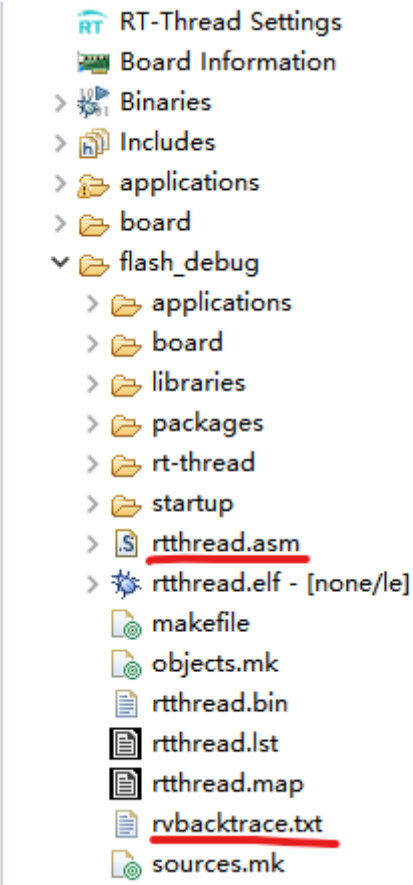
可以看到输出的信息已经很丰富了，在调试过程中还是很有帮助的。

3.2结合反汇编图形化

在RT-Studio配置输出反汇编命令

```
D:/RT-ThreadStudio/repo/Extract/ToolChain_Support_Packages/RISC-V/RISC-V-GCC-RV32/2022-04-12/bin/riscv32-unknown-elf-objdump -d rtthread.elf > rtthread.asm
```

在反汇编文件同级目录下创建一个txt文本将shell输出的信息拷贝至txt文本

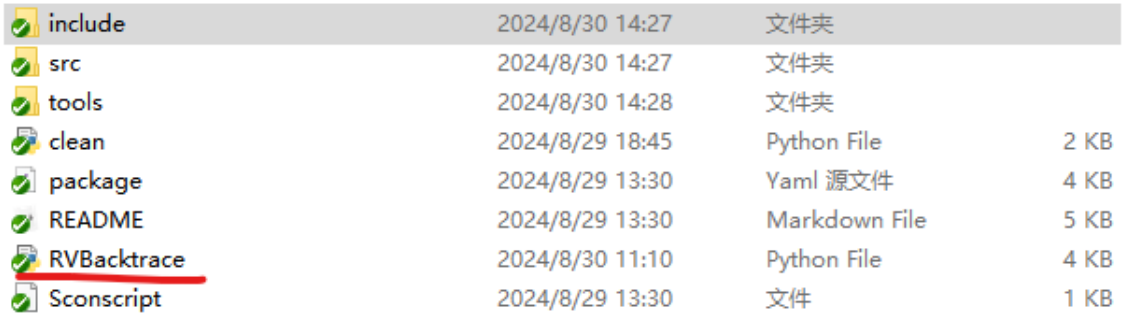


rvbacktrace.txt中的内容如下：

SConscript | main.c | rvbacktrace.txt

```
1
2 \ | /
3 - RT -     Thread Operating System
4 / | \     5.0.2 build Aug 30 2024 14:14:30
5 2006 - 2022 Copyright by RT-Thread team
6
7 ---- RV_Backtrace Call Frame Start: ----
8 ###Please consider the value of ra as accurate and the value of sp as only for reference###
9 [0]Stack interval :[0x000000000108ea58 - 0x000000000108ea68]  ra 0x0000000008000ffbc pc 0x0000000008000ffb8
10 [1]Stack interval :[0x000000000108ea68 - 0x000000000108ea78]  ra 0x0000000008000fe7c pc 0x0000000008000fe78
11 [2]Stack interval :[0x000000000108ea78 - 0x000000000108ea88]  ra 0x0000000080010082 pc 0x000000008001007e
12 [3]Stack interval :[0x000000000108ea88 - 0x000000000108ea98]  ra 0x000000008000688e pc 0x000000008000688a
13 [4]Stack interval :[0x000000000108ea98 - 0x00000000deadbeef]  ra 0x0000000001086a7c pc 0x0000000001086a78
14 ---- RV_Backtrace Call Frame End:----
15
16 addr2line -e rtthread.elf -a -f 8000ffb8 8000fe78 8001007e 8000688a 1086a78
17
18 msh >
```

然后点击工程目录下的RVBacktrace.py



运行界面：

这里我们输入rvbacktrace.txt与rtthread.asm的路径，为了方便将它俩放在了同一目录下，

复制它两的路径并输入到上述终端中：

```
D:\RT-ThreadStudio\workspace\HPM6750\flash_debug
```

运行结果：

同时会自动生成html文件，以表格的形似输出栈回溯信息，该文件在生成后自动打开。

RVBacktrace/栈回溯信息

函数地址	指令编码	汇编	函数名称
8000ffb8	f43ff0ef	jal ra,8000fefa	walk_stackframe
8000fe7a	2a19	jal 8000ff90	rv_backtrace_fno
8001007e	df5ff0ef	jal ra,8000fe72	rvbacktrace
8000688a	7e8090ef	jal ra,80010072	main

函数调用栈

调用栈
walk_stackframe <- rv_backtrace_fno <- rvbacktrace <- main

上述是第一次运行，如果后续有其他的栈回溯信息，我们只需要将shell输出的信息拷贝至之前创建的txt文本并保存，上述图表就会自动更新，即在完成第一次操作后，后续只需要将shell输出的新信息拷贝至txt文本即可。

如果需要修改txt文本的路径或者反汇编的路径，运行源码下的clean.py后清除中间文件，重新按上述步骤执行即可。

感觉不错的小伙伴点个小星星叭，一起向RT-Thread/RISC-V奔跑！