

# Nuclei QEMU 8.x 用户使用手册

## Nuclei QEMU 8.x 用户使用手册

说明

在nuclei-sdk上的使用

运行nuclei-linux-sdk

Nuclei Qemu 的 gdb调试

Qemu 常用的命令

## 说明

Nuclei QEMU是基于QEMU 8.x版本进行开发, 支持芯来demosoc和evalsoc的评估用SoC. 在指令扩展方面支持标准的RVV1.0, Zc扩展(RISC-V Code Size Reduction), B扩展, K扩展等特性, 也支持芯来自定义指令扩展Xxldsp/Xxldspn1x/Xxldspn2x/Xxldspn3x, Xxlcx扩展, 支持Nuclei ECLIC、System Timer特性, 也支持nuclei nice指令等特性。

本文档计划不再更新, 最新内容请查阅 [Nuclei Qemu使用手册](#)

## 在nuclei-sdk上的使用

可以根据 [Nuclei-Software/nuclei-sdk: Nuclei RISC-V Software Development Kit \(github.com\)](#)

详细编译步骤如下:

1. 下载nuclei-sdk的源代码, 切换到[develop](#)分支, 确保版本号 $\geq 0.5.0$ 。
2. 下载RISC-V GNU Toolchain  $\geq 2023.10$ : <https://www.nucleisys.com/download.php>
3. 将 `riscv64-unknown-elf-gcc` 所在的目录以及 `qemu-system-riscv32` 所在的目录设置到系统全局环境变量中

环境准备完成, 开始编译nuclei-sdk

### xxldsp例程:

进入nuclei-sdk/application/baremetal/demo\_dsp/目录, 输入下面的脚本进行编译

```
#确保riscv64-unknown-elf-gcc与qemu-system-riscv64已经在当前系统环境变量中
make CORE=nx900fd ARCH_EXT=_xxldsp clean dasm
make CORE=nx900fd ARCH_EXT=_xxldsp run_qemu
```

其中 `ARCH_EXT` 可以传递扩展名称, 详细参见 [Nuclei SDK使用手册](#)。

正常情况下, 可以看到最后输出 `NMSIS_TEST_PASS`, 表示所有的case通过测试。

### qemu参数详细介绍:

**qemu-system-riscv32:** qemu riscv 架构主程序, 分为 `qemu-system-riscv32` 与 `qemu-system-riscv64`。其中, `qemu-system-riscv32` 对应32位架构的cpu, 比如n200,n300,n600等等, `qemu-system-riscv64` 对应64位架构, 比如nx600,nx900等系列。u系列处理器带有MMU, 可以运行Linux。

	N级别 (32位架构)	U级别 (32位架构,MMU)	NX级别 (64位架构)	UX级别 (64位架构,MMU)
1000系列				UX1000FD
900系列	N900 N900F N900FD	U900 U900F U900FD	NX900 NX900F NX900FD	UX900 UX900F UX900FD
600系列	N600 N600F N600FD	U600 U600F U600FD	NX600 NX600F NX600FD	UX600 UX600F UX600FD
300系列	N300 N300F N300FD N305 N307 N307FD			
200系列	N200 N201 N201E N203 N203E N205 N205E			
100系列	N100 N100M N100ZMMUL N100E N100EM N100EZMMUL			

**-M nuclei\_evalsoc,download=ilm**: -M表示-machine，也就是选择机器的类型，目前Nuclei QEMU在原有的基础上新增了nuclei\_demosoc(后续版本将会舍弃)与nuclei\_evalsoc。download= 选择下载模式，目前支持四种下载模式(flashxip,flash,ilm,ddr)。

**-cpu nuclei-nx900fd,ext=\_xxldsp**:通过-cpu 后传递core的类型。开启不同的扩展的方式也是在里面添加，比如xxldsp 表示开启芯来DSP扩展。目前nuclei qemu支持的常用扩展类型如下：

扩展名称	功能
v	RISC-V V-Extension
h	RISC-V H-Extension
zicbom	RISC-V Zicbom Extension
zicboz	RISC-V Zicboz Extension
zicond	RISC-V Zicond Extension
zicsr	RV32/RV64 Zicsr Standard Extension
zifencei	RV32/RV64 Zifencei Standard Extension
zihintpause	ZiHintPause extension
zawrs	Zawrs extension
zfh	Zfh Extension
zfa	Zfa Extension
zfhmin	Zfhmin Extension
zfinx	Zfinx Extension
zdinx	Zdinx Extension
zca	RISC-V ZC* Extension
zcb	RISC-V ZC* Extension
zcf	RISC-V ZC* Extension
zcd	RISC-V ZC* Extension
zce	RISC-V ZC* Extension
zcmp	RISC-V ZC* Extension
zcmt	RISC-V ZC* Extension
zba	RISC-V Bitmanipulation Extension
zbb	RISC-V Bitmanipulation Extension
zbc	RISC-V Bitmanipulation Extension
zbkb	RISC-V Bitmanipulation Extension
zbkc	RISC-V Bitmanipulation Extension
zblkx	RISC-V Bitmanipulation Extension
zbs	RISC-V Bitmanipulation Extension
zk	RISC-V Scalar Crypto Extension
zkn	RISC-V Scalar Crypto Extension

扩展名称	功能
zknd	RISC-V Scalar Crypto Extension
zkne	RISC-V Scalar Crypto Extension
zknh	RISC-V Scalar Crypto Extension
zkr	RISC-V Scalar Crypto Extension
zks	RISC-V Scalar Crypto Extension
zkse	RISC-V Scalar Crypto Extension
zksh	RISC-V Scalar Crypto Extension
zkt	RISC-V Scalar Crypto Extension
zve32f	RISC-V V-Extension
zve64f	RISC-V V-Extension
zve64d	RISC-V V-Extension
zvf	RISC-V V-Extension
zvfmin	RISC-V V-Extension
zhinx	Zhinx Extension
zhinxmin	Zhinxmin Extension
smaia	Smaia Extension
ssaia	Ssaia Extension
sscfpmf	Sscfpmf Extension
sstc	Sstc Extension
svadu	Svadu Extension
svinval	Svinval Extension
svnapot	Svnapot Extension
svpbmt	Svpbmt Extension
xxldsp	Nuclei DSP Extension based on P-ext 0.5.4 + default 8 EXPD instructions
xxldspn1x	Xxldsp + Nuclei N1 extension
xxldspn2x	Xxldspn1x + Nuclei N2 extension
xxldspn3x	Xxldspn2x + Nuclei N3 extension
xxlcz	Nuclei code size reduction extension

在使用时，若需要开启zc和xlcz扩展，请采用 `ext=_zca_zcb_zcmp_zcmt_xxlcz`。开启qemu的扩展指令。

-kernel xxx.elf: qemu用 -kernel 指定运行的elf文件，该文件可以是nuclei-sdk编译的程序，也可以自己编写的裸机代码。

## 运行nuclei-linux-sdk

关于编译nuclei linux sdk, 可以查看: <https://github.com/Nuclei-Software/nuclei-linux-sdk>

根据下面的文档，切换到需要运行的分支(建议使用 dev\_nuclei\_5.10\_v2 分支)

<https://github.com/Nuclei-Software/nuclei-linux-sdk/issues/2>

这里假定测试的SOC和CORE分别是evalsoc, ux900fd(rv64imafdc)。

在linux-sdk环境按照文档步骤设置好以后，执行如下命令进行Linux SDK的构建

```
export PATH=/path/to/linux_qemu/bin:$PATH
# choose rv64imafdc
# clean previous build if existed
make CORE=ux900fd clean
# build freeloader is enough, no need to build boot images, sd card is not needed
# bitstream with 50MHz CPU is expected by default
make CORE=ux900fd freeloader
# run on qemu
make CORE=ux900fd run_qemu
```

## Nuclei Qemu 的 gdb调试

对于qemu的调试，需要加上 -s -S。

```
qemu-system-riscv32 -M nuclei_evalsoc,download=ilm -cpu nuclei-n300,ext=_zca_zcb_zcmp_zcmt -smp 1 -icount shift=0 -nodefaults -nographic -serial stdio -kernel coremark.elf -s -S
```

-s -S 的含义如下:

-S	freeze CPU at startup (use 'c' to start execution)
-s	shorthand for -gdb tcp::1234

重新打开一个终端，连接tcp端口1234即可。

通过 riscv64-unknown-elf-gdb 来进行调试。

```
riscv64-unknown-elf-gdb application/baremetal/benchmark/coremark/coremark.elf
```

输入

```
target remote localhost:1234
```

即可连接。

通过输入 b main 表示将main函数处打断点。

通过输入 c 让程序正常向下运行。

**gdb 调试常用命令，详细参见gdb手册**

打开汇编显示

```
set disassemble-next-line on
```

汇编级别单步跳转

```
si
```

C语言级别单步跳转

```
n
```

设置断点

```
b
```

查看寄存器

```
info register <register name>
```

查看所有寄存器

```
info register all
```

## Qemu 常用的命令

1. 打开qemu gdb前端

```
-s -S
```

用gdb后端连接时，只需要 `target remote localhost:1234`。

2. 查看支持的机器类型

```
qemu-system-riscv64 -M ?
```

3. 查看支持的cpu类型

```
qemu-system-riscv64 -M nuclei_evalsoc -cpu ?
```

4. 查看qemu版本号

```
qemu-system-riscv64 --version
```