

# aarch64汇编开发和学习环境有哪些

Arm精选 2024年11月27日 19:31 上海



Arm精选

ARMv8/ARMv9架构、SOC架构、Trustzone/TEE安全、终端安全、SOC安全、ARM安全、A...  
819篇原创内容

公众号

## 目录

GNU GCC + QEMU + GDB

ARM汇编在线仿真器

C语言/汇编在线转换工具

在线指令速查网站

cemu 汇编模拟器

## GNU GCC + QEMU + GDB

安装ARM交叉编译器：

```
sudo apt install gcc-aarch64-linux-gnu binutils-aarch64-linux-gnu
```

安装QEMU环境：

```
sudo apt install qemu qemu-user qemu-user-static
```

安装gdb环境

```
sudo apt install gdb-multiarch
```

```
test@pc:~/workspace/code/test$ sudo apt install gcc-aarch64-linux-gnu binutils-aarch64-linux-gnu
Reading package lists... Done
Building dependency tree
Reading state information... Done
test@pc:~/workspace/code/test$ sudo apt install qemu qemu-user qemu-user-static
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

编写汇编代码：hello\_world.s

```
.section .data
msg:      .asciz "Hello, AArch64!\n"
```

```
.section .text
.global _start
```

```
_start:
```

```

// Write the string to stdout
mov     x0, 1           // File descriptor (stdout)
ldr     x1, =msg        // Load the address of the string
mov     x2, 16          // Length of the string
mov     x8, 64          // syscall: write
svc     0               // Make syscall

// Exit the program
mov     x8, 93          // syscall: exit
mov     x0, 0           // Exit status
svc     0               // Make syscall

```

## 编写Makefile文件

```

hello_world:hello_world.o
    aarch64-linux-gnu-ld -o hello_world hello_world.o

hello_world.o:hello_world.s
    aarch64-linux-gnu-as -o hello_world.o hello_world.s

clean:
    rm hello_world.o

```

## 执行hello\_world程序

```

test@pc:~/workspace/code/test/hello_world$ make
aarch64-linux-gnu-as -o hello_world.o hello_world.s
aarch64-linux-gnu-ld -o hello_world hello_world.o
test@pc:~/workspace/code/test/hello_world$
test@pc:~/workspace/code/test/hello_world$ qemu-aarch64 ./hello_world
Hello, AArch64!
test@pc:~/workspace/code/test/hello_world$
test@pc:~/workspace/code/test/hello_world$

```

## hello\_world代码的程序解释

```
.section .data
msg: .asciz "Hello, AArch64!\n"

.section .text
.global _start

_start:
// Write the string to stdout
mov x0, 1 // File descriptor (stdout)
ldr x1, =msg // Load the address of the string
mov x2, 16 // Length of the string
mov x8, 64 // syscall: write
svc 0 // Make syscall

// Exit the program
mov x8, 93 // syscall: exit
mov x0, 0 // Exit status
svc 0 // Make syscall
```

### syscall 使用:

**write** 的 syscall 编号为 64, 参数依次为:

x0: 文件描述符 (1 为 stdout)。

x1: 字符串地址。

x2: 字符串长度。

**exit** 的 syscall 编号为 93, 参数为:

x0: 退出状态。

公众号 · Arm精选

## ARM汇编在线仿真器

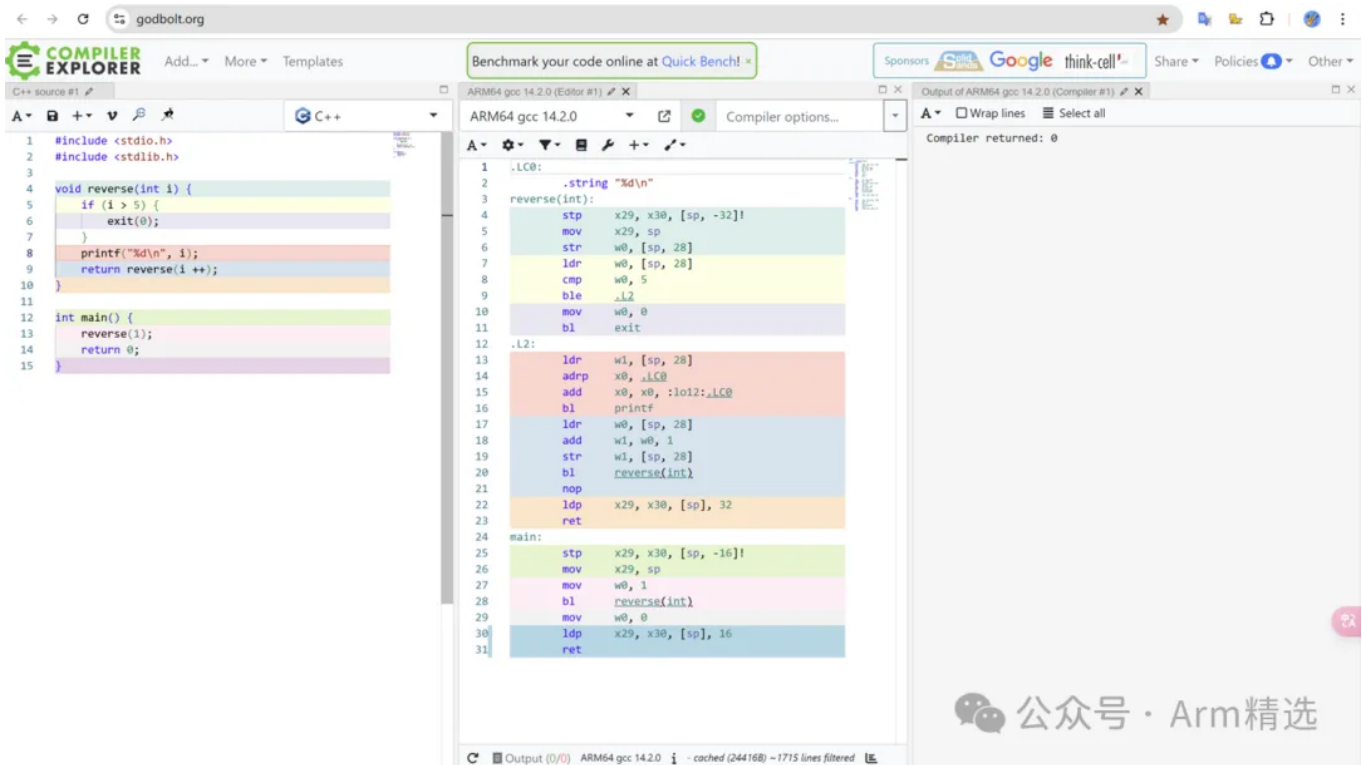
<http://163.238.35.161/~zhangs/arm64simulator/>

The screenshot displays the ARM64 Simulator web interface. The top navigation bar includes a hamburger menu, navigation icons, and the URL [163.238.35.161/~zhangs/arm64simulator/](http://163.238.35.161/~zhangs/arm64simulator/). The main interface is divided into several panels:

- Registers:** A table showing the state of 32 ARM registers (X0-X31) and the FP register. All registers are currently at 0x0000000000000000.
- Condition Flags:** A table showing the state of the Condition Flags (N, Z, C, V). N is 0, Z is 1, C is 0, and V is 0.
- Instructions:** A list of instructions being executed. The first three are:
  - 1: mov x0, #1
  - 2: mov x1, #2
  - 3: add x2, x0, x1
- Machine Code:** A table showing the machine code for the instructions. The first three are:
  - Address: 00010000, Bytes: 20 00 80 D2, Binary: 00100000 00000000 10000000 11010010, Assembly: mov x0, #1
  - Address: 00010004, Bytes: 41 00 80 D2, Binary: 01000001 00000000 10000000 11010010, Assembly: mov x1, #2
  - Address: 00010008, Bytes: 02 00 01 8B, Binary: 00000010 00000000 00000001 10001011, Assembly: add x2, x0, x1
- Memory:** A table showing the memory contents. The first three are:
  - Address: 00010000, Bytes: 20 00 80 D2 41 00 80 D2 02 00 01 8B 1C 42 3B, ASCII: ...A.....B;.
  - Address: 00010010, Bytes: 00 00 00 00 00 00 00 00 00 00 00 00 00 00, ASCII: .....
  - Address: 00010020, Bytes: 00 00 00 00 00 00 00 00 00 00 00 00 00 00, ASCII: .....
- Output:** A text area showing the output of the simulation. It displays: "Run: started", "3 instructions are executed!", and "Run: completed".

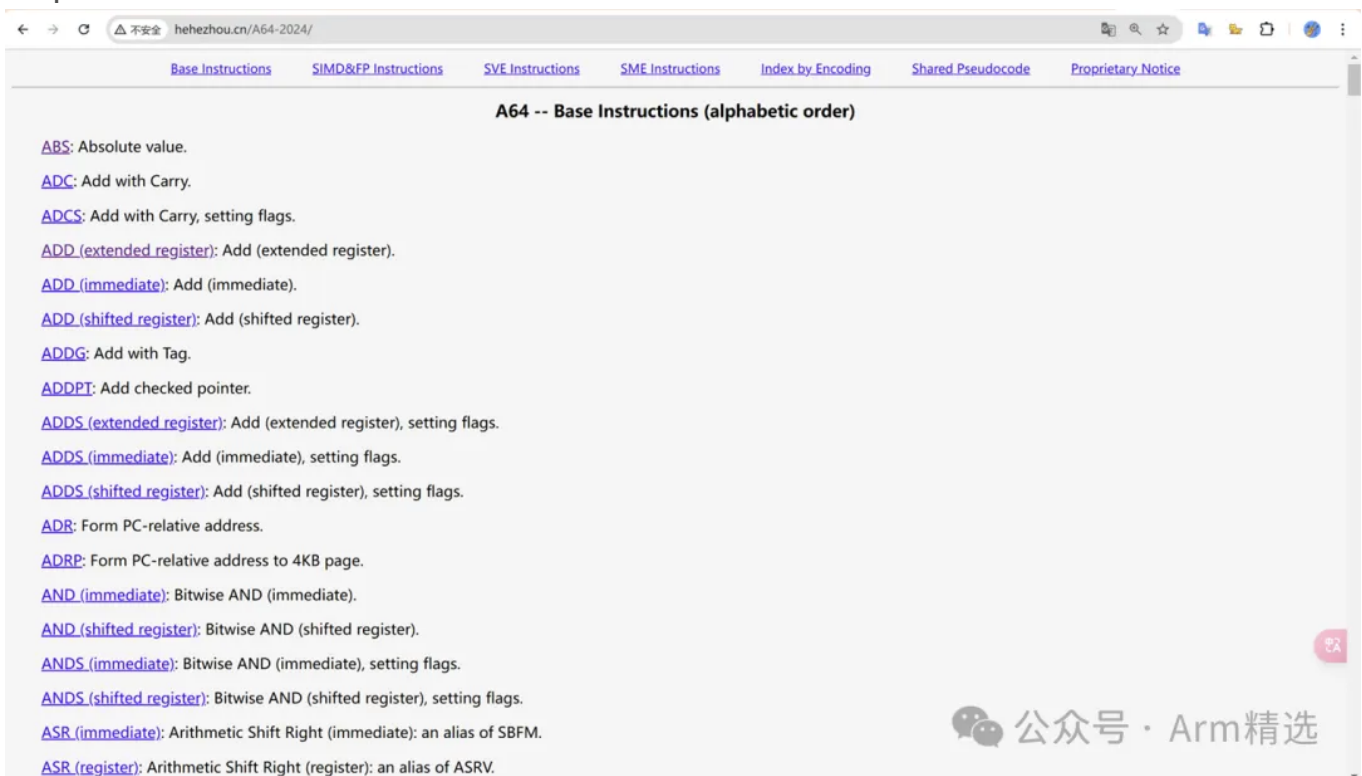
## C语言/汇编在线转换工具

<https://godbolt.org/>



## 在线指令速查网站

<http://hehezhou.cn/A64-2024/>



## cemu 汇编模拟器

CEmu - Cheap Emulator v.0.8 (ARM AARCH64)

FileArchitectureViewHelp

Registers

Register	Value
X0	0x0000000000000000
X1	0x0000000000000000
X2	0x0000000000000000
X3	0x0000000000000000
X4	0x0000000000000000
X5	0x0000000000000000
X6	0x0000000000000000
X7	0x0000000000000000
X8	0x0000000000000000

Memory Map

	Start	End	Name
1	0x0000000000004000	0x0000000000005000	.text
2	0x0000000000005000	0x0000000000006000	.data
3	0x0000000000006000	0x000000000000a000	.stack

Add SectionRemove Section

Code View

Code (Line:1 Column:36)

mov x0, #0x64  
mov x1, #0xC8

Control Panel

Python Console

# Welcome to CEMU Python console (v3.12.3-final,0)  
# You can interact with any emulator component (regi  
sters, memory, code, etc.).  
# The emulator is exposed via the 'emu' object, and  
the VM via 'vm'.  
  
>>>

Scratchboard

Memory Viewer

Location16

Cemu Logs

[INFO] Emulator is now FINISHED  
[INFO] Emulator is now TEARDOWN  
[INFO] Ending emulation context at 0x4000  
[INFO] Emulator is now NOT\_RUNNING  
[INFO] Emulation context reset  
[INFO] Setting up emulation environment...  
[WARNING] No value specified for PC register, setting to 0x4000  
[WARNING] No value specified for SP register, setting to 0x8000  
[INFO] Using text section MemorySection([0x4000-0x4fff], name='.text',  
permission=readexec)  
[ERROR] Failed to compile: exception KsError: Unknown token in expression  
(KS\_ERR\_ASM\_EXPR\_TOKEN)  
[ERROR] \_\_generate\_text\_bytecode() failed  
[INFO] Emulator is now RUNNING  
[INFO] Emulator is now FINISHED

Finished

公众号 · Arm精选