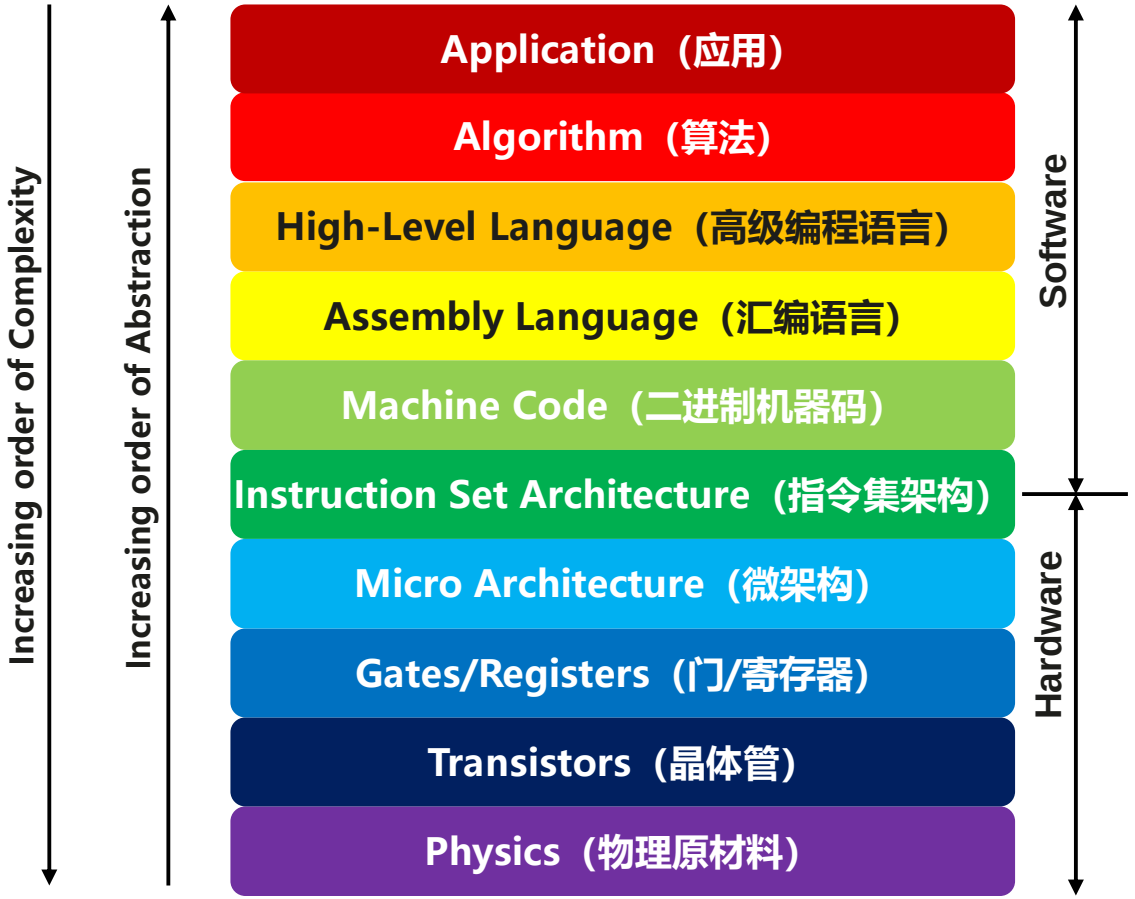
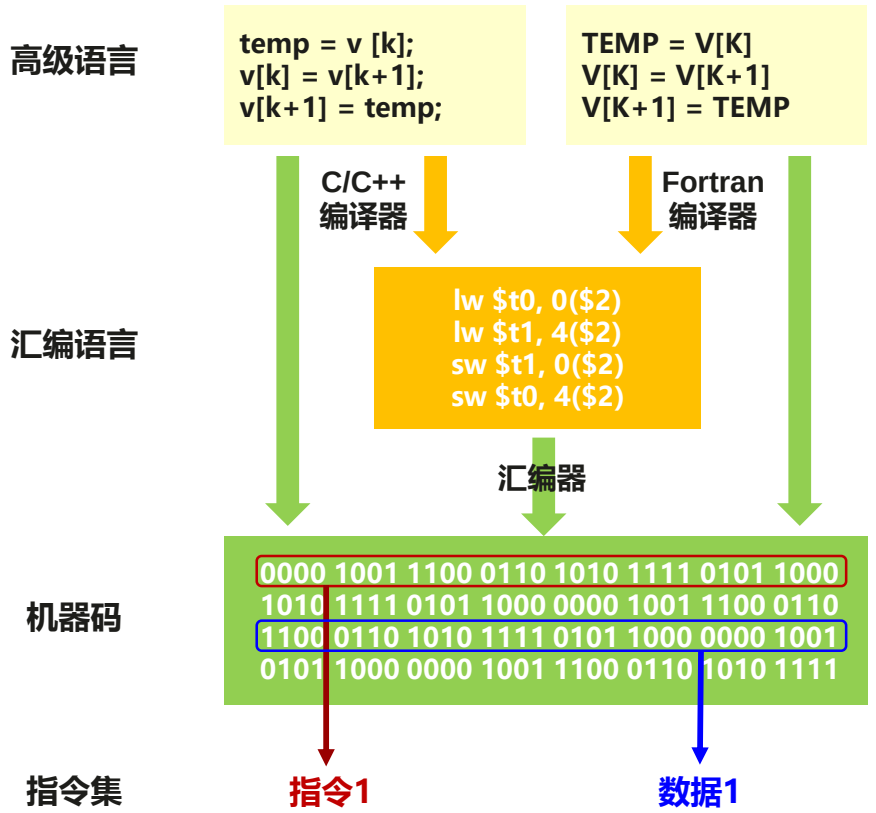
通用计算机技术栈与程序执行过程是怎样的?

计算技术栈

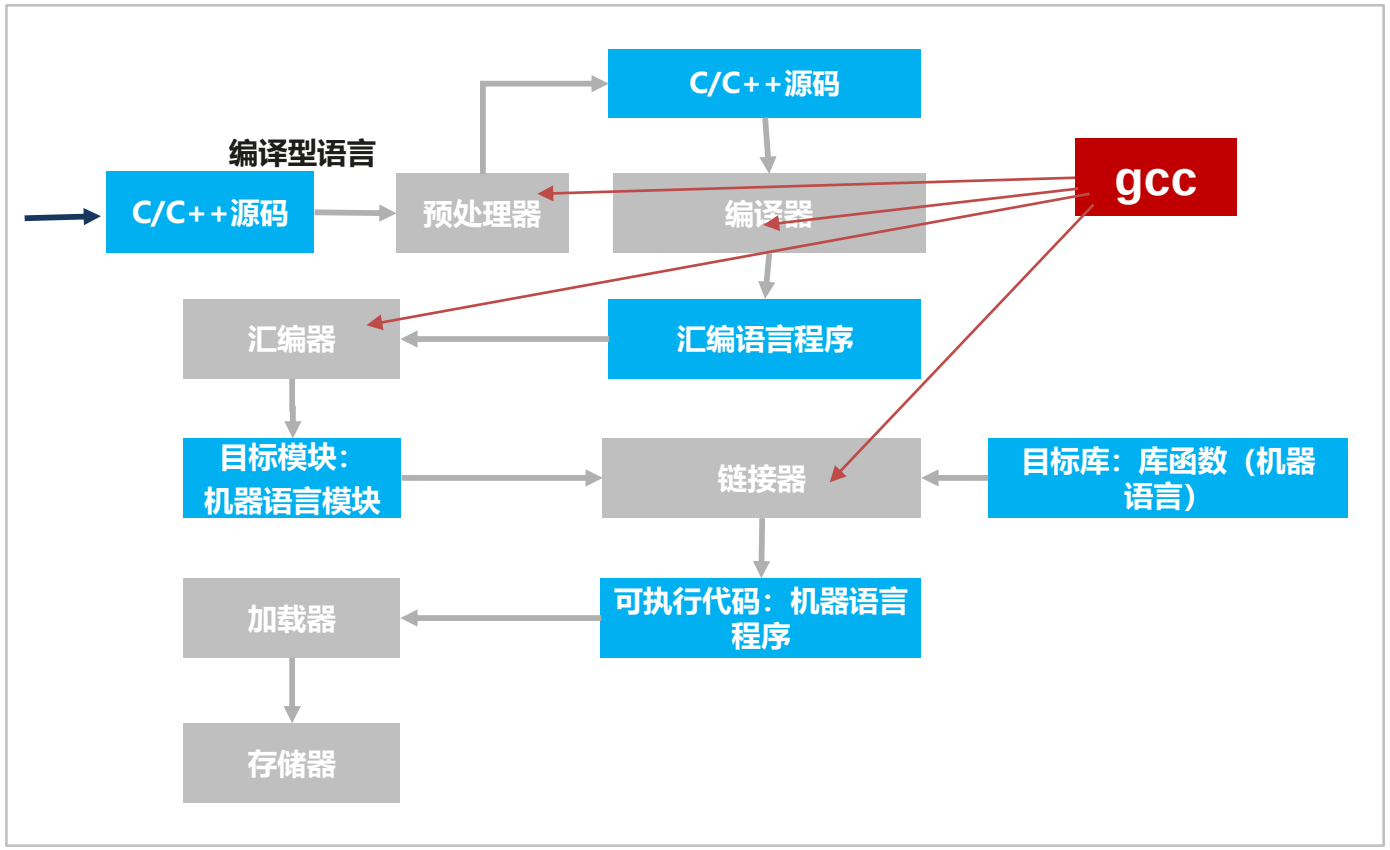


程序执行过程



编译型语言程序运行原理及编译方式之一gcc

- 从源码到程序的过程：源码需要由编译器、汇编器翻译成机器指令，再通过链接器链接库函数生成机器语言程序。机器语言必须与CPU的指令集匹配，在运行时通过加载器加载到内存，由CPU执行指令。



- 预处理：gcc -E XX.c -o XX.i
- 编译：gcc -S XX.i –o XX.s
- 汇编：gcc -c XX.s -o XX.o
- 链接：gcc XX.o -o -lxx XX

(-lxx可选)

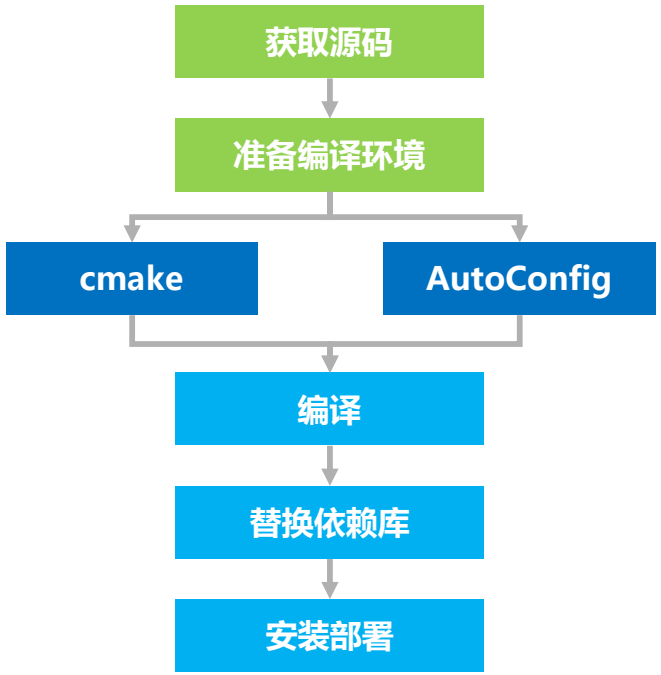
- 具体通过gcc –help查看gcc使用指导

```
-E                       Preprocess only; do not compile, assemble or link
-S                       Compile only; do not assemble or link
-c                       Compile and assemble, but do not link
-o <file>                Place the output into <file>
```

gcc编译器能将C、C++语言源程序通过**预处理、编译、汇编、链接**等过程生成可执行文件

编译型语言程序运行原理及编译方式之二makefile

makefile 文件描述了整个工程的编译、链接等规则，开发者通过make一行命令来完成工程“自动化编译”



1. 获取源码：通过github或第三方开源社区获取源码
2. 准备编译环境：安装编译器gcc等
3. **生成编译配置：使用开源软件源码中的cmake或AutoConfig脚本生成makefile**
4. **编译：执行makefile编译可执行程序**
5. 替换依赖库：通过开源软件readme文件、编译时的so库缺失或链接错误，重新编译或替换依赖库。
6. 将可执行程序安装部署到生产或测试系统

Makefile 规则

```
target... : prerequisites ...
    command
```

Makefile 样例

```
calc: main.o
    cc -o main.o
main.o:main.c
```

鲲鹏与x86处理器有哪些关键的指令差异?

程序代码（C/C++）：

```
int main()
{
    int a = 1;
    int b = 2;
    int c = 0;
    c = a + b;
    return c;
}
```

编译

鲲鹏处理器指令

指令	汇编代码	说明
b9400fe1	ldr x1, [sp,#12]	从内存将变量a的值放入寄存器x1
b9400be0	ldr x0, [sp,#8]	从内存将变量b的值放入寄存器x0
0b000020	add x0, x1, x0	将x1(a)中的值加上x0(b)的值放入x0寄存器
b90007e0	str x0, [sp,#4]	将x0寄存器的值存入内存（变量c）

x86处理器指令

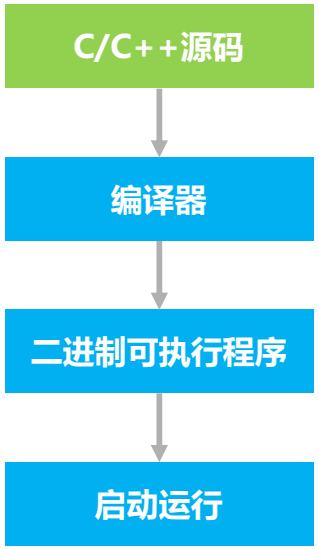
指令	汇编代码	说明
8b 55 fc	mov -0x4(%rbp),%edx	从内存将变量a的值放入寄存器edx
8b 45 f8	mov -0x8(%rbp),%eax	从内存将变量b的值放入寄存器eax
01 d0	add %edx,%eax	将edx(a)中的值加上eax(b)的值放入eax寄存器
89 45 f4	mov %eax,-0xc(%rbp)	将eax寄存器的值存入内存（变量c）

2大差异:

1. **指令长度（鲲鹏等长，指令流水并发度较X86高、X86不等长，擅长串行），鲲鹏在高性能计算等高并发场景下性价比提升30%**
2. **汇编代码（X86与鲲鹏汇编指令不一致， X86程序无法直接移植到鲲鹏，必须经过重新编译才能运行）**

C/C++代码在X86与鲲鹏平台代码差异点有哪些?

C/C++程序编译运行的过程



1. *安装编译器
2. 配置GCC环境变量
3. **修改C/C++源码**
4. **编译C/C++源码，生成可执行程序**
5. 启动C/C++程序，调试功能

*说明： 推荐gcc 7.3.0以上版本

编译型代码的适配过程中，需要针对ARM和x86的环境差异，在源码及编译过程中做相应调整，下表是几个典型的修改示例：

适配动作	示例			
	修改点	修改示例说明	X86代码	鲲鹏代码
修改源码	**数据类定义**	**显示定义char类型变量为有符号型**	**char a = ‘a’**	**signed char a = ‘a’**
	Builtin函数	使用编译器自带的builtin函数	__builtin_ia32_crc32qi (__a, __b);	__builtin_aarch64_crc32b (__a, __b);
		把内存数据预取到cache中	asm volatile("prefetcht0 %0" :: "m" (*(unsigned long *)x));	#define prefetch(_x) __builtin_prefetch(_x)
	64位编译	定义编译生成的应用程序为64 位，编译选项不同	编译选项：-m64	编译选项：-mabi=lp64
编译	ARM指令集	Makefile中定义指令集类型，由X86修改为ARM	编译选项：-march=broadwell	编译选项：-march=armv8-a
	编译宏	原有X86版编译宏替换成ARM版	__X86_64__	__ARM_64__

重点：char字符在X86默认是unsigned，在鲲鹏是signed

因此对于鲲鹏平台，定义char类型要使用：signed char