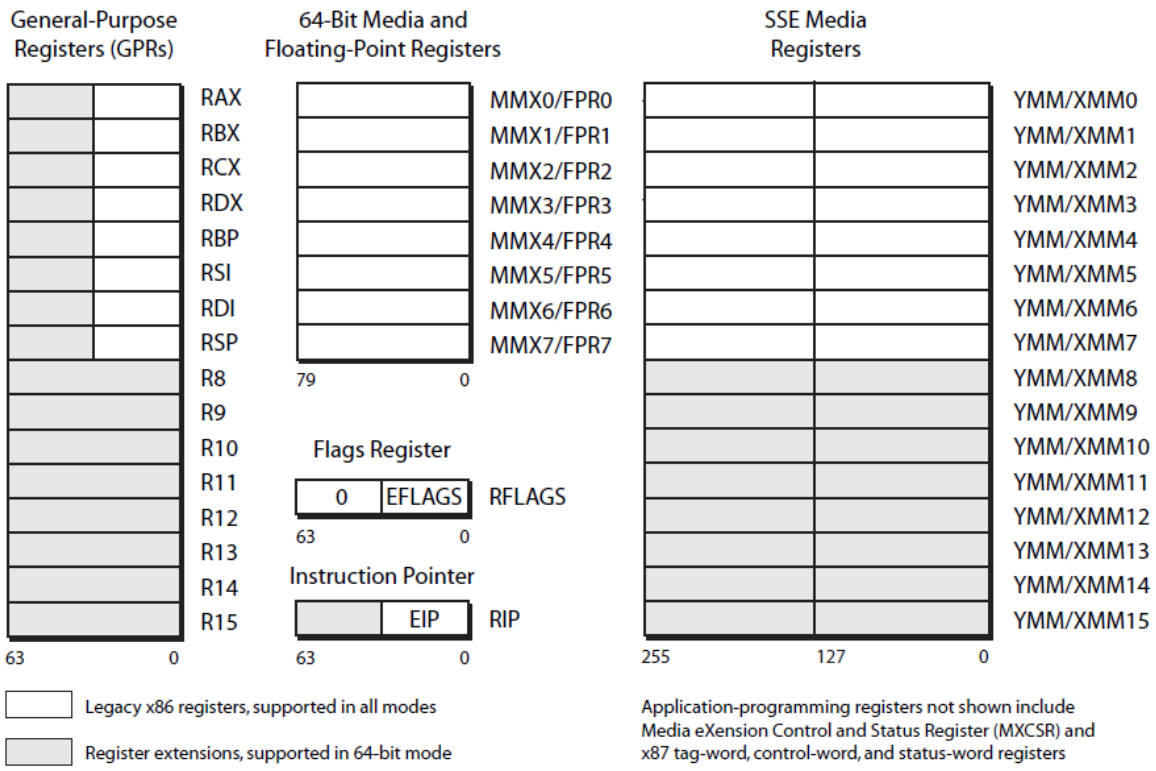
x64汇编简介

Intel手册：https://software.intel.com/en-us/articles/intel-sdm

AMD手册：https://developer.amd.com/resources/developer-guides-manuals

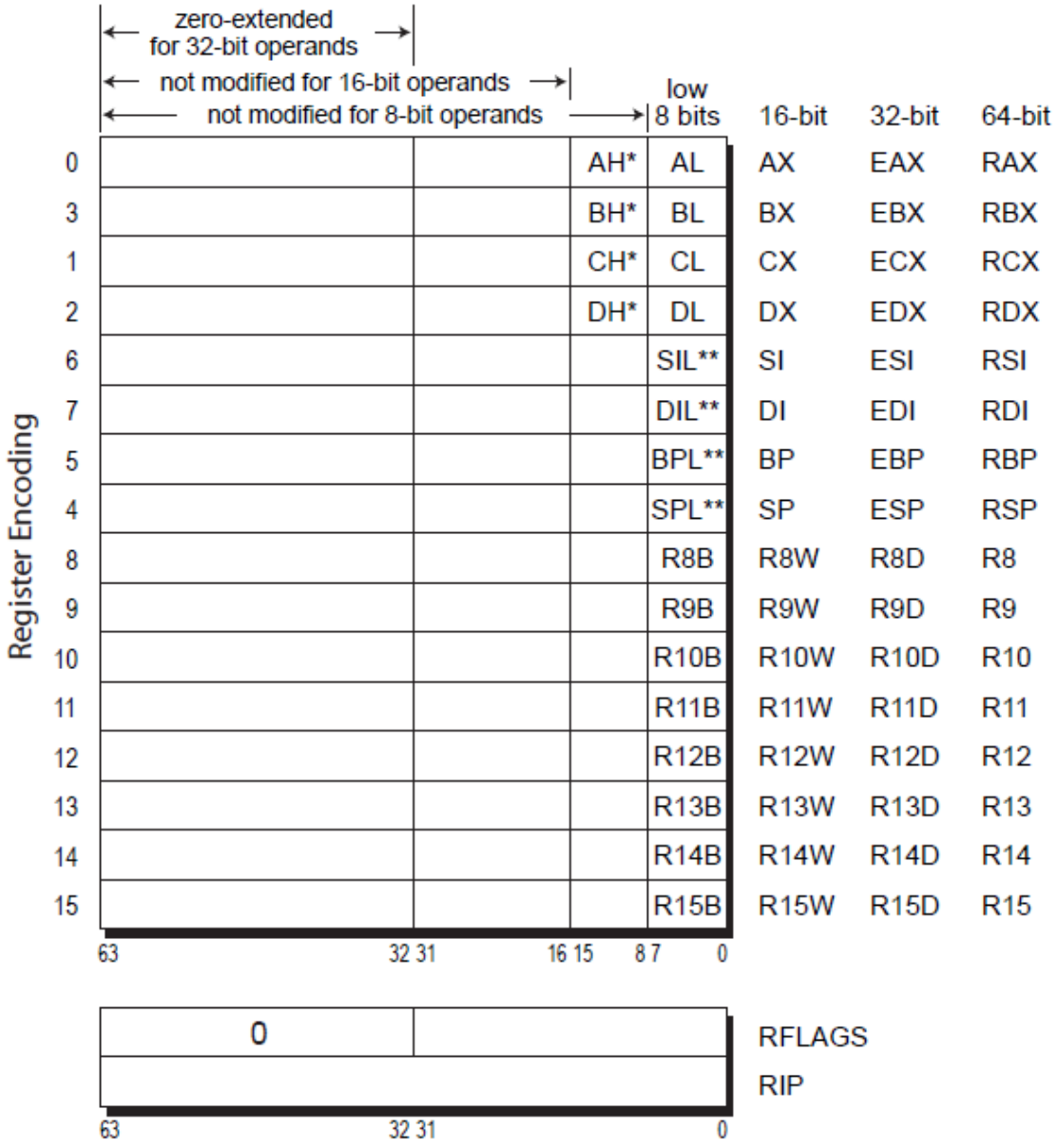
寄存器



通用寄存器扩展到16个，大小64位

多媒体寄存器扩展到16个，大小256位

通用寄存器的访问



通过加后缀来访问高低位，兼容以前的不变

- 后缀B：低8位
- 后缀W：低16位
- 后缀D：低32位

操作低**8位**和**16位**时，**高位不变**，而操作**32位**时，**高位则会清0**

```
mov eax, 1
; 等价于
mov rax, 1
```

第一个64汇编程序

默认情况下，x64 应用程序二进制接口（ABI）使用四寄存器 fast 调用约定。

前四个参数使用寄存器：RCX、RDX、R8 和 R9，如果超过，则其余参数压栈传递

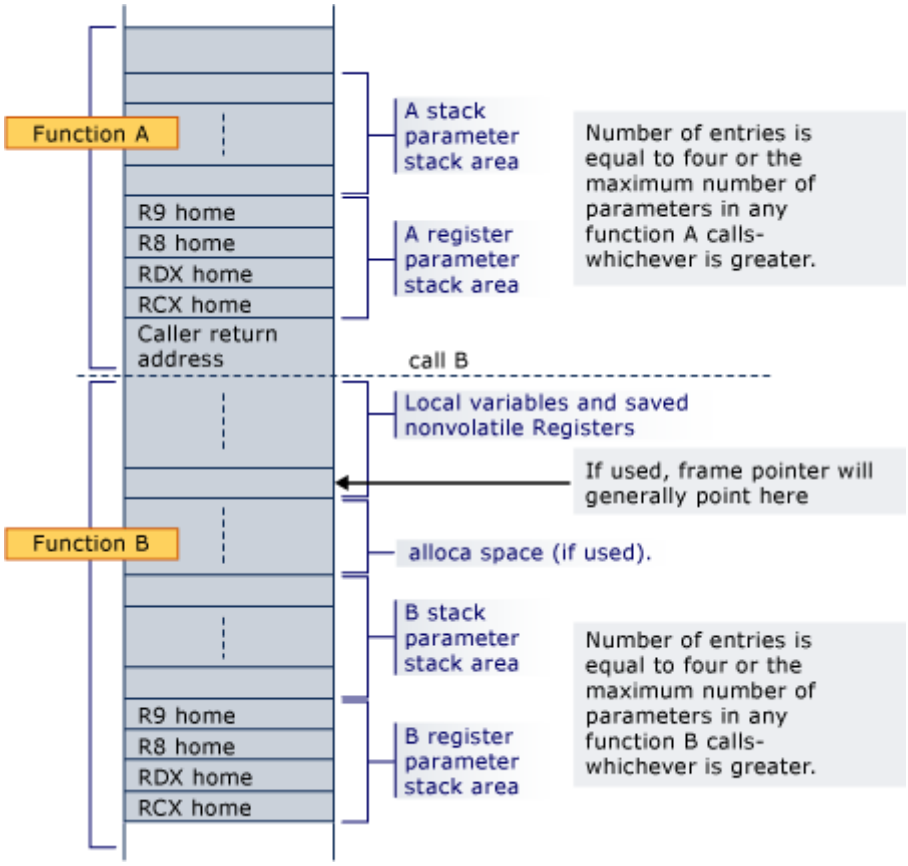
浮点参数传递在 XMM0L、XMM1L、XMM2L 和 XMM3L 中

详细文档：https://docs.microsoft.com/zh-cn/cpp/build/x64-software-conventions?view=vs-2019

```
; 编译链接如下：
; ml64 /c hello.asm
; link /subsystem:windows /entry:Main hello.obj

extern MessageBoxA:proc
extern ExitProcess:Proc

includelib user32.lib
includelib kernel32.lib

.const
    MYMSG1 db "Hello World", 0
    TITLE1 db "x64 asm", 0

.code
Main proc
    ; 抬栈必须模8，多余的8用来对齐，防止多媒体指令导致栈不平衡
    sub rsp, 28h    ; 预留空间，调用函数之前必须要给预留空间，且多个函数可以共用，多余的参数在预留空间下面
                    ; 且多余的参数一般使用mov来往栈上传递，例如
                    ; 第5个参数：mov qword ptr[rsp+20h], 第5个参数
                    ; 第6个参数：mov qword ptr[rsp+28h], 第6个参数
                    ; 第7个参数：mov qword ptr[rsp+30h], 第7个参数
    mov rcx, 0
    mov rdx, offset MYMSG1
    mov r8, offset TITLE1
    mov r9, 0
    call MessageBoxA

    mov rcx, 0
    call ExitProcess

    add rsp, 20h
Main endp
end
```

x64的栈帧

针对于MS编译器来说，栈帧分布如下：



注意

x64汇编中，已经不支持32位的伪指令