# 1、简介

微处理器处理数据主要分为以下几种：

Single instruction single data—SISD

Single instruction multiple data(vector mode)—SIMD

Single instruction multiple data(packed data mode)—SIMD

（1）SISD

一次指令操作一个数据。如下例子4次指令操作才完成8个寄存器相加：

add ro, r5

add r1, r6

add r2, r7

add r3, r8

（2）SIMD(vector mode)

一次指令可以处理多个数据，但是每个数据处理是顺序执行，如下：

VADD.F32 S24, S8, S16

//S24=S8+S16

//S25=S9+S17

//S26=S10+S18

//S27=S11+S19

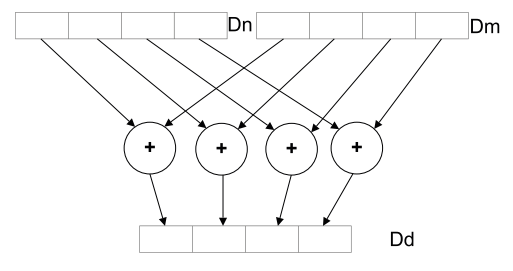
一个指令，但是数据相加是顺序执行。在ARM上这个也叫Vector Floating Point（VFP）。

（3）SIMD(packed data mode)

一次指令可以处理多个数据，由于使用大寄存器方式可以同时进行，如下：

VADD.I16 Q10, Q8, Q9

一个指令将两个64-bit寄存相加，I16表示数据类型int16，64-bit= 4 \* 16，每个寄存器里4个16-bit lanes独立相加，但是同时完成，如图：



在ARM上这个叫做增强型SIMD技术或NEON技术。

# 2、寄存器信息

在armv7-A和armv7-R的体系架构上基本采用了neon技术，armv8也支持并与armv7兼容。VFP和NEON共用协处理器CP10和CP11的指令空间和寄存器，并成为ARM/Thumb指令集的一部分。

NEON寄存器就会是由相同数据类型组成的元素向量，一个寄存器（即一个向量）划分为多个管道，每个管道包含一个数据元素。每个neon指令就是这些管道元素并行操作，且互相不影响。Neon划分为两种向量：64-bit NEON vectors（doubleword简称D寄存器）和128-bit NEON vectors（quadword简称Q寄存器）。

64-bit NEON vectors包含：

8个8-bit elements（即8个管道，每个管道可以包含8bit的数据元素）

4个16-bit elements

2个32-bit elements

1个64-bit elements

128-bit NEONvectors包含：

16个8-bit elements

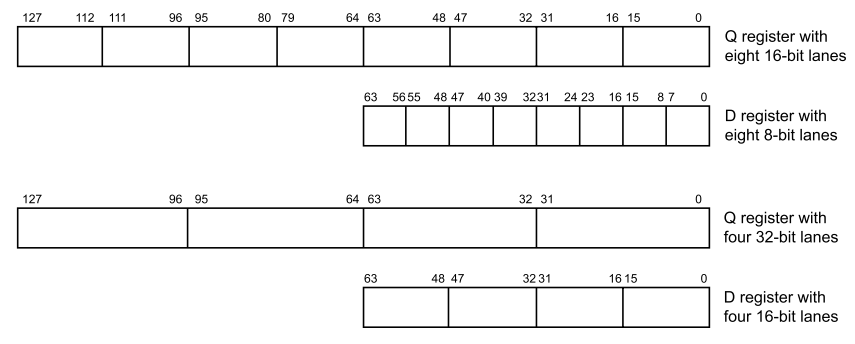
8个16-bit elments

4个32-bit elements

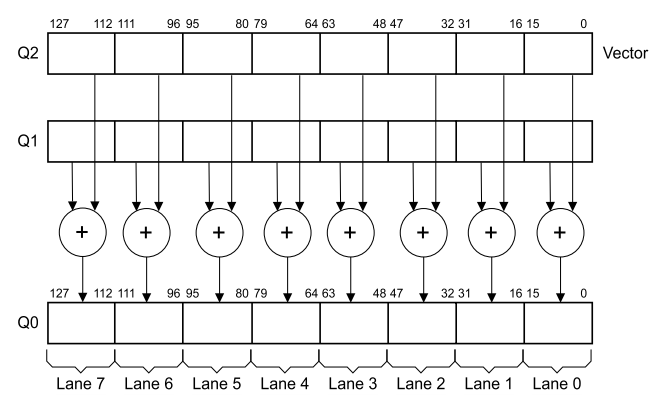
2个64-bit elements

这样划分就可以neon用在不同场景：像素处理8-bit，颜色处理16-bit等。

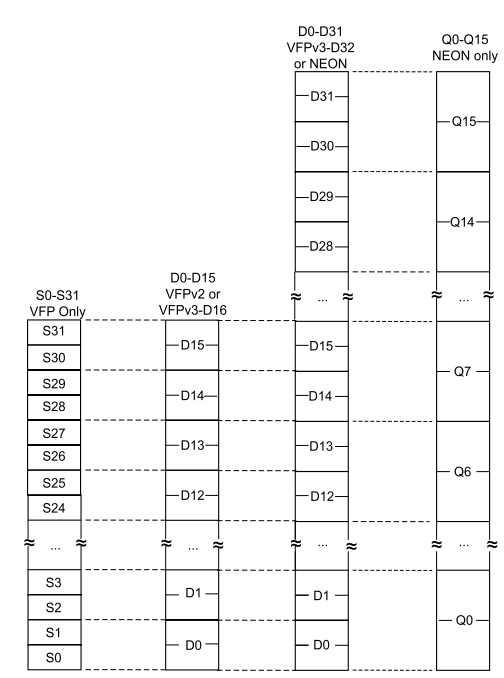
管道lane和元素element关系



执行neon指令：VADD.I16 Q0, Q1, Q2，Q代表128-bit寄存器，I16代表16-bit，这句话就是8个16-bit lanes的128-bit向量寄存器相加，如图所示：



VFP和NEON是共享寄存器空间，因此寄存器存是重叠的，如图所示：



Neon 寄存器单元：

16个128-bit Q 寄存器，Q0~Q15

32个64-bit D寄存器，D0~D31

VFPv3-D32:

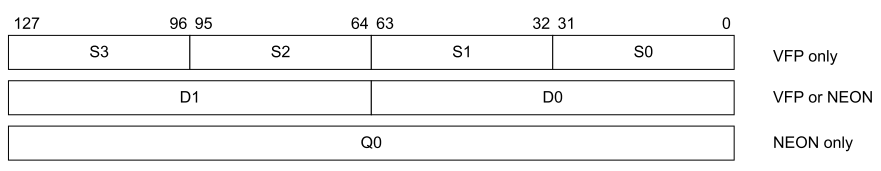
32个64-bit D寄存器，D0~D31

VFPv2或VFPv3-16：

16个64-bit D寄存器，D0~D15

VFP：

16个32-bit S寄存器

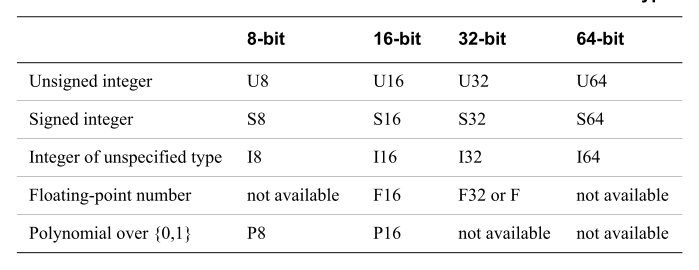


neon单元可以访问Q0寄存器作为一个128-bit寄存器

neon单元可以访问Q0寄存器作为两个64-bit寄存器D0和D1

neon不能范围32-bit的S寄存器，但VFP可以访问32-bit

下表是neon支持的数据类型：



# 3、neon使用方法

## （1）汇编

通过汇编直接时钟neon指令：

.text

.arm

.global double\_elements

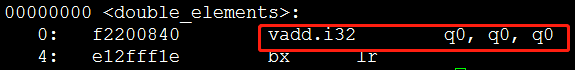
double\_elements:

vadd.i32 q0,q0,q0

bx lr

.end

arm-hisiv300-linux-as -mfloat-abi=softfp -mfpu=neon-vfpv4 neon\_as.S



该种方式效率最高，但是难度大，移植性差

## （2）使用arm提供的Intrinsics函数

可以认为是内联函数，但是在编译时编译器会将函数转化为neon指令。调用该函数需要包含头文件arm\_neon.h，该头文件包含了neon各种操作函数，具体可以该头文件arm-hisiv300-linux/lib/gcc/arm-hisiv300-linux-uclibcgnueabi/4.8.3/include/arm\_neon.h或参考文档《neon\_programmers\_guide.pdf》附录D。

简单例子：

#include <arm\_neon.h>

uint32x4\_t double\_elements(uint32x4\_t input)

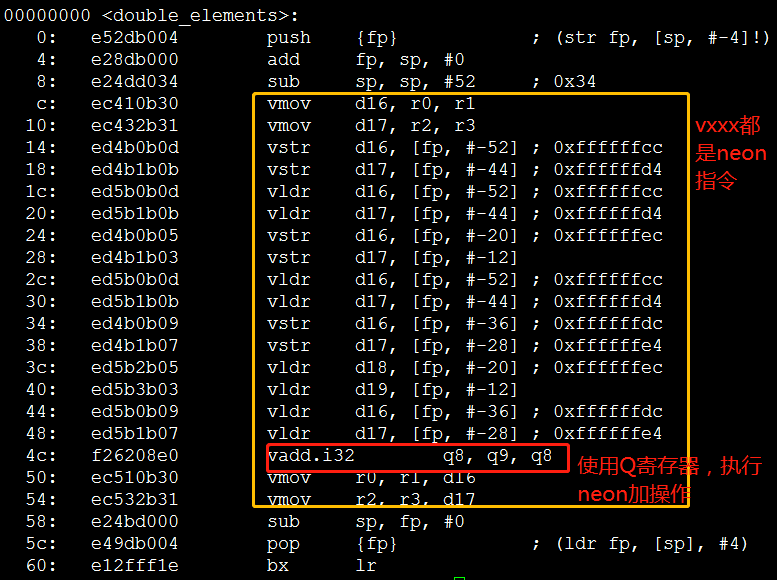
{

return(vaddq\_u32(input, input));

}

arm-hisiv300-linux-gcc -mfloat-abi=softfp -mfpu=neon-vfpv4 -c neon\_in.c

汇编结果如下：



## （3）自动化向量（实际验证未通过）

该方式需要对指针参数添加\_\_restrict（用于限定和约束指针，表明指针是访问一个数据对象的唯一且初始的方式）。简单例子：

void add\_ints(int \* \_\_restrict pa, int \* \_\_restrict pb, unsigned int n, int x)

{

unsigned int i;

for(i = 0; i < (n & ~3); i++)

pa[i] = pb[i] + x;

}

void add\_floats(float \* \_\_restrict pa, float \* \_\_restrict pb, unsigned int n, float x)

{

unsigned int i;

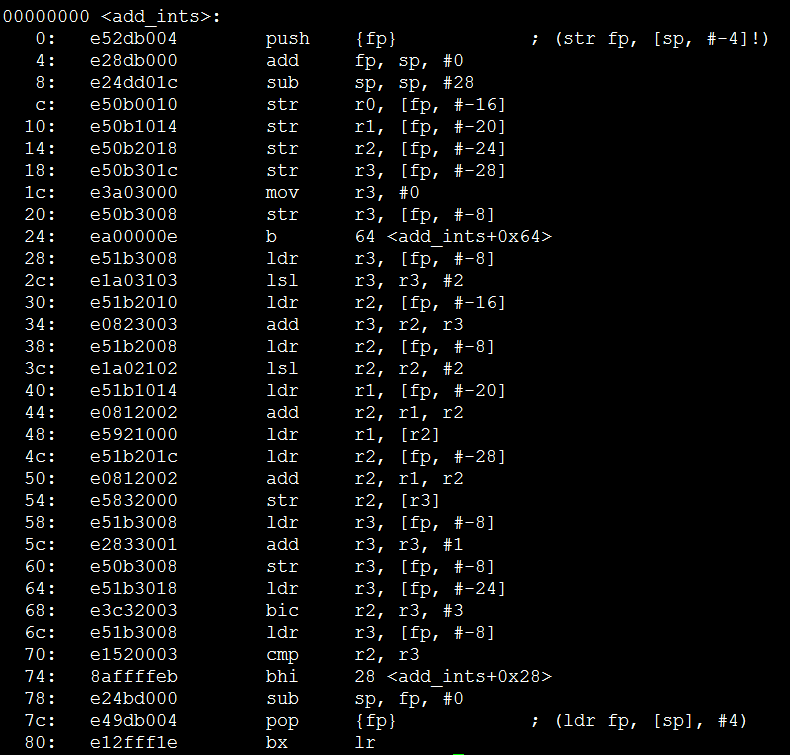
for(i = 0; i < (n & ~3); i++)

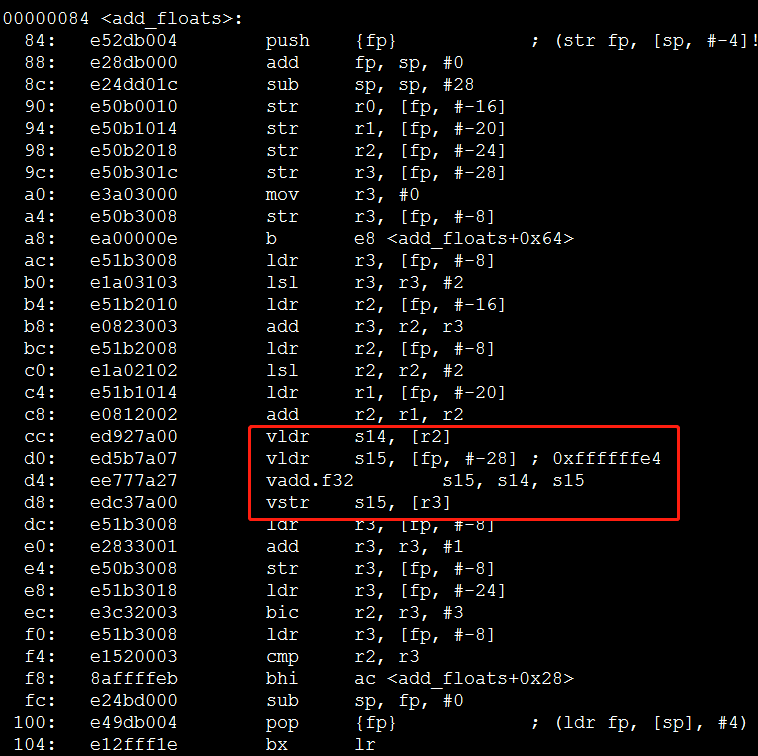
pa[i] = pb[i] + x;

}

arm-hisiv300-linux-gcc -mfloat-abi=softfp -mfpu=neon-vfpv4 -ftree-vectorize -ftree-vectorizer-verbose=1 -c neon\_av.c

汇编如下:





该方法浮点运算会自动转化为neon指令，而整型则不会。从第二小节可以知道neon支持整型操作，而实际图像，颜色等都是整形数的，该方法对于这些操作就无法发挥neon指令优势。

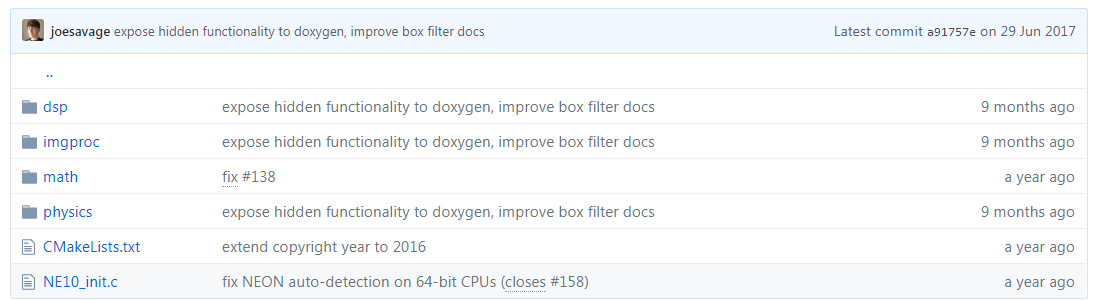
## （4）使用第三方优化库

以下列举了已经用neon优化过的库，编译时加入neon参数，然后调用相应接口进行操作。

A．Ne10

官网：http://projectne10.github.com/Ne10/

git地址：https://github.com/projectNe10/Ne10



C接口函数提供了汇编和常用处理方法

B．OpenMAX DL

地址：<http://www.khronos.org/openmax/> 网站无法打开

OpenMAX由Khronos group(也是open Gl制定者)制定的开放多媒体加速层，是一个不需要授权、跨平台的软件抽象层，以C语言实现的软件接口，提供了一下系列处理音频、视频、静态图片的API。

OpenMAX自上而下分为三个层次：OpenMAX AL，OpenMAX IL和OpenMAX DL。

OpenMAX AL:Application Layer， 应用和多媒体中间层的标准接口，使得应用在多媒体接口上具有了可移植性

OpenMAX IL: Integration Layer，作为在嵌入式和移动设备中使用的audio,video,images codecs的底层接口。使得AP和[多媒体框架](https://baike.baidu.com/item/%E5%A4%9A%E5%AA%92%E4%BD%93%E6%A1%86%E6%9E%B6)可以以统一的方式访问多媒体codec和支持组件。Codec可以是硬件和软件的任意组合，对用户透明。

OpenMAX DL: Development Layer，最底层使用neon进行了优化

C．ffmpeg

官网：http://ffmpeg.org/

git地址：https://github.com/FFmpeg/FFmpeg

FFmpeg是一套可以用来记录、转换数字音频、视频，并能将其转化为流的开源计算机程序。采用LGPL或GPL许可证。它提供了录制、转换以及流化音视频的完整解决方案。它包含了非常先进的音频/视频编解码库libavcodec，为了保证高可移植性和编解码质量，libavcodec里很多code都是从头开发的。

功能包括[视频采集](https://baike.baidu.com/item/%E8%A7%86%E9%A2%91%E9%87%87%E9%9B%86)功能、[视频格式转换](https://baike.baidu.com/item/%E8%A7%86%E9%A2%91%E6%A0%BC%E5%BC%8F%E8%BD%AC%E6%8D%A2)、视频抓图、给视频加水印等。

D．Eigen3

官网：<http://eigen.tuxfamily.org/>

Github：https://github.com/artsy/eigen

Eigen是一个高层次的C ++库，有效支持线性代数，矩阵和矢量运算，数值分析及其相关的算法

E．Pixman

官网：<http://pixman.org/>

2D图形库

F．x264

官网：<http://www.videolan.org/developers/x264.html>

G．Math-neon

官网：<http://code.google.com/p/math-neon/>

## （5）hi3536 目前使用neon的方法

目前，我们只是编译时添加了参数，未做其他处理：

CFLAGS += -mfloat-abi=softfp -mfpu=neon-vfpv4 -mno-unaligned-access -fno-aggressive-loop-optimizations。简单例子;

float sum1(float a, float b)

{

return (a+b);

}

int sum2(int a, int b)

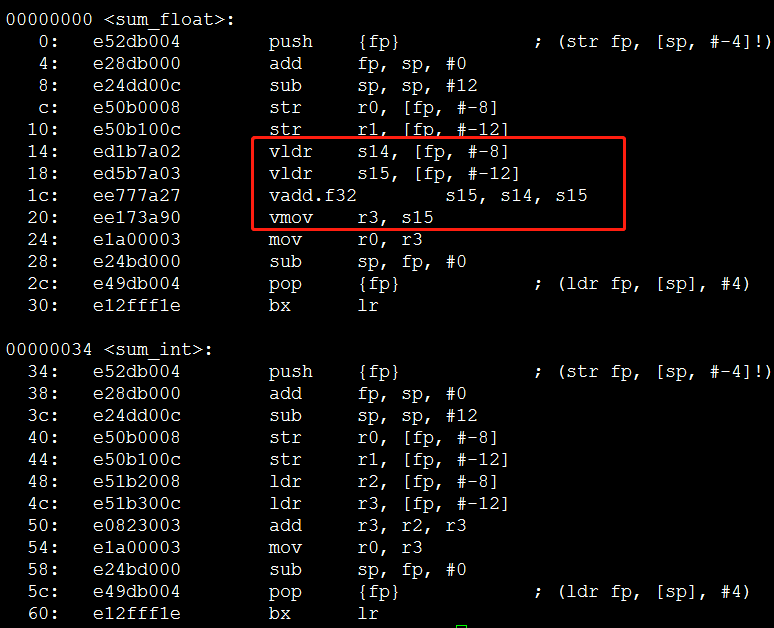
{

return (a+b);

}

arm-hisiv300-linux-gcc -mfloat-abi=softfp -mfpu=neon-vfpv4 -c neon.c

汇编如下：



也是对浮点进行了neon操作，而整型是保持不变的。

因此使用的是第三种自动化向量的方法。

# 4、neon应用例子

在《neon\_programmers\_guide.pdf》文档中举了好多个neon使用的例子：

A．交换RGB颜色通道

B．处理非对齐数组

C．矩阵计算

D．向量积

E．转换色彩深度

F．中值滤波

G．FIR滤波

# 5、官方参考文档

Neon资料比价少，在arm官网上可以查到如下几个资料，第一个是详细说明。

《neon\_programmers\_guide.pdf》

《introducing\_neon.pdf》

《neon\_support\_in\_compilation\_tools.pdf》