**ARM-NEON-SIMD-COPROCESSOR**

ARM-NEON 单指令多数据(Single instruction, multiple data)协处理器

1. NEON intrinsics (C语言，使用起来类似于函数调用)

简单、易维护和移植、效率相对较低、不必考虑超出寄存器使用数量

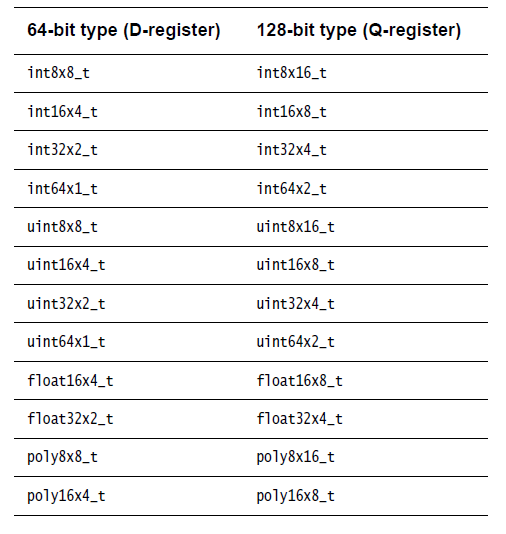
2. 汇编语言 ()

复杂、移植较难、效率高、寄存器必须人工合理分配

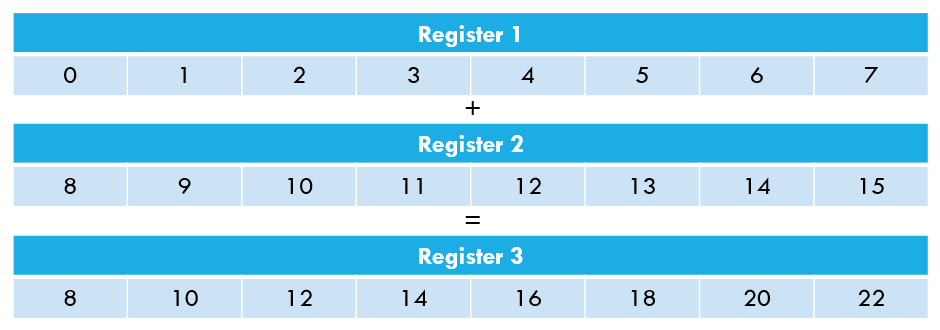
寄存器：32x 64-bit type D-Register

可组成：16x 128-bit type Q-Register

寄存器的拆分格式



例：定义3个uint8x8\_t （64-bit D-Register），实现加法



//使用NEON实现 a + b = c

#include <stdio.h>

#include <stdlib.h>

#include <arm\_neon.h>

#include <math.h>

int main(int argc, char\*\* argv)

{

//定义a, b, c

unsigned char a[8] = {0, 1, 2, 3, 4, 5, 6, 7};

unsigned char b[8] = {8, 9, 10, 11, 12, 13, 14, 15};

unsigned char c[8];

uint8x8\_t rega, regb, regc; //定义3个8x8bit无符号整型的 NEON 寄存器

//加载 a, b 到寄存器

rega = vld1\_u8(&a[0]);

regb = vld1\_u8(&b[0]);

regc = vadd\_u8(rega, regb); //做加法

vst1\_u8(&c[0], regc); //回写到c中

//测试

for(int i = 0 ; i < 8 ; i++)

{

printf("%d ",c[i] );

}

printf("\n");

}

**数据类型**

NEON vector data types are named according to the following pattern:

<type><size>x<number\_of\_lanes>\_t

For example:

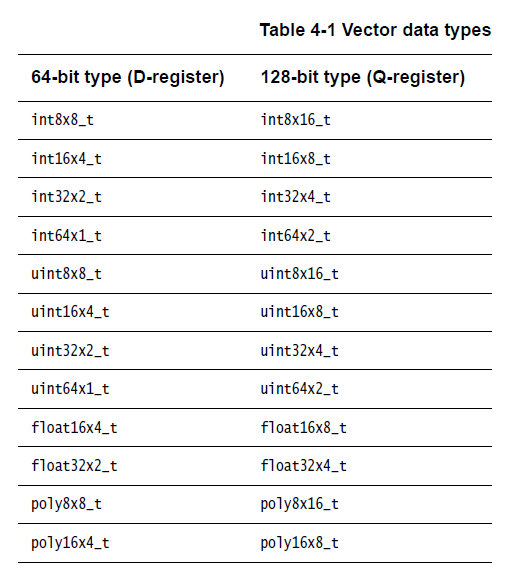
• int16x4\_t is a vector describes a vector of four 16-bit short int values.

• float32x4\_t describes a vector of four 32-bit float values.

[粘贴自NEON官方编程指南]

具体举例

uint8x8\_t rega;



<type><size>x<number\_of\_lanes>x<length\_of\_array>\_t

These types are ordinary C structures containing a single element named val.

These types map the registers accessed by NEON load and store operations, which can

load/store up to four registers with a single instruction. An example structure definition is:

struct int16x4x2\_t

{

int16x4\_t val[2];

} <var\_name>;

These types are only used by loads, stores, transpose, interleave and de-interleave instructions;

to perform operations on the actual data, select the element from the individual registers for

example, <var\_name>.val[0] and <var\_name>.val[1].

结构体举例 ：

uint16x4x2\_t reg1;

使用：

reg.val[0] = …;

reg.val[1] = …

NEON intrinsics的使用

The intrinsics use a naming scheme similar to the NEON unified assembler syntax:

<opname><flags>\_<type>

An additional q flag is provided to specify that the intrinsic operates on 128-bit vectors.

For example:

• vmul\_s16, multiplies two vectors of signed 16-bit values.

This compiles to VMUL.I16 d2, d0, d1.

• vaddl\_u8, is a long add of two 64-bit vectors containing unsigned 8-bit values, resulting in

a 128-bit vector of unsigned 16-bit values.

This compiles to VADDL.U8 q1, d0, d1.[粘贴自NEON官方编程指南]

The examples below show different variants of the same intrinsic.

uint8x8\_t vadd\_u8(uint8x8\_t a, uint8x8\_t b);

The intrinsic vadd\_u8 does not have the ‘q’ suffix. In this case, the input and output vectors are

64-bit vectors, which use D registers.

uint8x16\_t vaddq\_u8(uint8x16\_t a, uint8x16\_t b);

The intrinsic vaddq\_u8 has the ‘q’ suffix, so the input and output vectors are 128-bit vectors,

which use Q registers.

uint16x8\_t vaddl\_u8(uint8x8\_t a, uint8x8\_t b); [粘贴自NEON官方编程指南]

数据的读取与回写

其中比较重要的是 vld1和vst1

vld1： VLD1 loads a vector from memory.

vst1： VST1 stores a vector into memory.