# Porting & Optimising Code 32-bit to 64-bit



Matthew Gretton-Dann Technical Lead - Toolchain Working Group



#### **A Presentation of Four Parts**

- Register Files
- Structure Layout & Data Models
- Atomics
- Vectorization & Neon Intrinsics



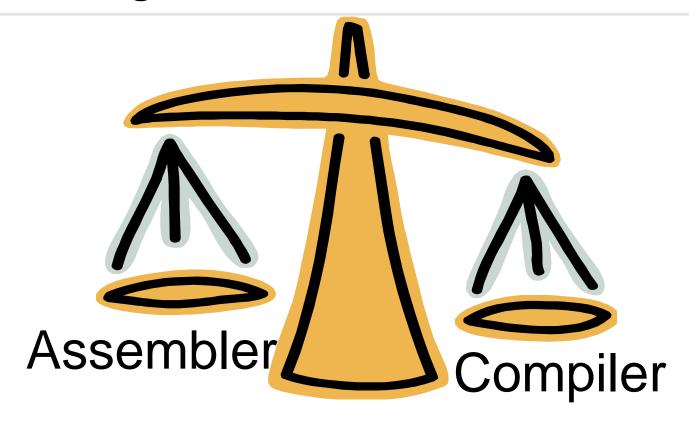
### **Simplification**

- View for those writing apps
- No complicated kernel stuff
- Little Endian



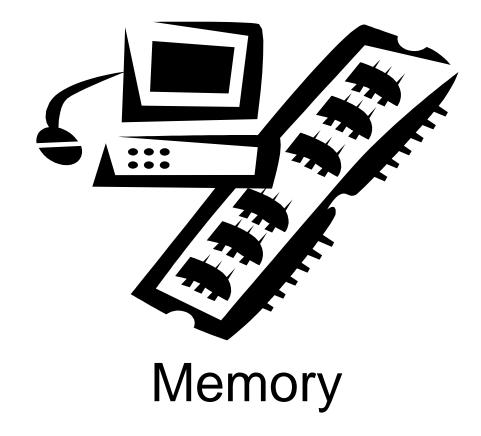


### **Bias Warning**





### Why 64-bit?





### **General Purpose Registers – 32-bit ARM**

r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13 (SP)
r14 (LR)
r15 (PC)



### **General Purpose Registers**

r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13 (SP)
r14 (LR)
r15 (PC)



### **General Purpose Registers**

rO
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13 (SP)
r14 (LR)
r15 (PC)

r16
r17
r18
r19
r20
r21
r22
r23
r24
r25
r26
r27
r28
r29
r30



### **General Purpose Registers – 64-bit ARM**

r0	r16
r1	r17
r2	r18
r3	r19
r4	r20
r5	r21
r6	r22
r7	r23
r8	r24
r9	r25
r10	r26
r11	r27
r12	r28
r13	r29
r14	r30 (LR)
r15	

PC

Linaro

SP

### **General Purpose Registers**





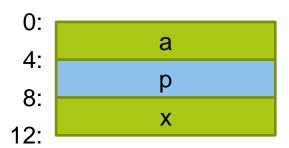
### **General Purpose Registers – Consequences**

- Easier to do 64-bit arithmetic!
- Less need to spill to the stack
- Spare registers to keep more temporaries



### **Structure Layout – 32-bit**

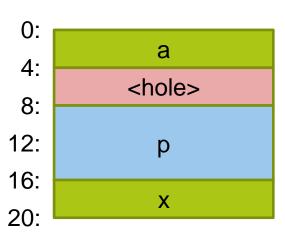
```
struct foo {
  int32_t a;
  void* p;
  int32_t x;
};
```





### **Structure Layout – 64-bit**

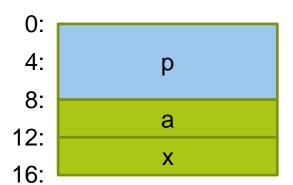
```
struct foo {
  int32_t a;
  void* p;
  int32_t x;
};
```





### **Structure Layout – 64-bit**

```
struct foo {
 void* p;
 int32_t a;
 int32_t x;
};
```





#### **Brief Aside**

- API: Application Programming Interface
  - Defines the interfaces a programmer may use
  - High level
- ABI: Application Binary Interface
  - Defines how to call functions, layout memory &c.
  - Low level



# ILP32

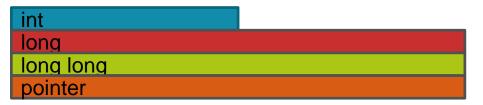
int
long
long long
pointer



# ILP32



## LP64





ILP32

int
long
long long
pointer

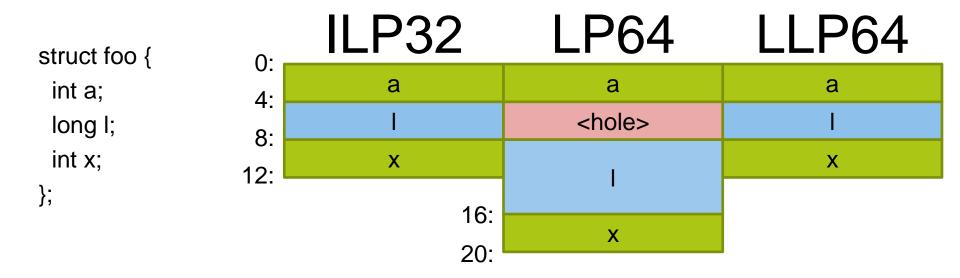
LP64

LLP64

int
long
long long
pointer

int
long
long long
pointer







### That's It...



### One more thing...



### One more thing...

Remove conditionalisation



### Two more things...

- Remove conditionalisation
- Add some new load/store semantics



### Three more things...

- Remove conditionalisation
- Add some new load/store semantics
- Change the register layout for the floating-point/SIMD registers

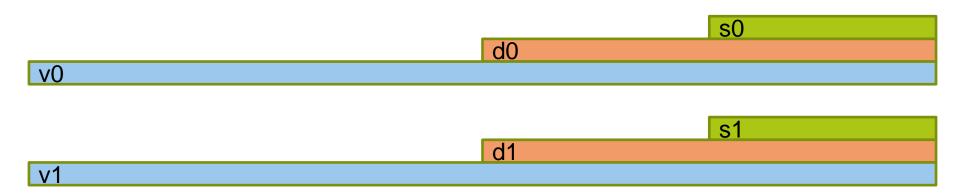
s3	s2	s1	s0	
d1		d0		
q0				

s7	s6	s5	s4
d3		d2	
q1			



### Three more things...

- Remove conditionalisation
- Add some new load/store semantics
- Change the register layout for the floating-point/SIMD registers





### Four more things...

- Remove conditionalization
- Add some new load/store semantics
- Change the register layout for the float-point/SIMD registers
- Add some more SIMD instructions



### Many more things...

- Remove conditionalization
- Add some new load/store semantics
- Change the register layout for the float-point/SIMD registers
- Add some more SIMD instructions
- •



```
#if defined( GNUC ) &&
                                           #else
   (defined( i386 ) ||
                                           int AtomicAdd (volatile int* ptr, int
   defined( x86 64 ))
                                              increment)
int
AtomicAdd(volatile int* ptr, int increment)
                                            *ptr += increment;
                                            return *ptr;
 int temp = increment;
  asm volatile (
                                           #endif
 "lock; xaddl %0,%1"
 : "+r" (temp), "+m" (*ptr) : : "memory");
 return temp + increment;
```

```
type __atomic_add_fetch (type *ptr, type val, int memmodel)
These built-in functions perform the operation suggested by the
  name, and return the result of the operation. That is,
{ *ptr op= val; return *ptr; }
All memory models are valid.
```



```
int AtomicAdd(volatile int* ptr, int increment)
{
  return __atomic_add_fetch (ptr, increment, memmodel);
}
```



- There are basically three types of memory model defined by C++11 which GCC's support is based upon:
  - Sequentially Consistent
  - Acquire/Release
  - Relaxed



### **Atomics – Sequentially Consistent**

a = 1; x.store(20); if (x.load() == 20) assert (a == 1);





#### **Atomics – Relaxed**

a = 1; x.store(20, memory\_order\_relaxed);

```
if ( x.load(memory_order_relaxed)) ==
   20)
assert (a == 1);
```





### **Atomics – Acquire/Release**

x.store (10, memory\_order\_release);

y.store (20, memory\_order\_release);

```
assert (y.load (memory_order_acquire)
== 20 &&
x.load (memory_order_acquire) ==
0)
```

```
assert (y.load (memory_order_acquire)
== 0 &&
x.load (memory_order_acquire) ==
10)
```



### **Atomics – Sequentially Consistent**

x.store (10);

y.store (20);

assert (y.load () == 20 && x.load () == 0)



assert (y.load () == 0 && x.load () == 10)





### **Atomics – Sequentially Consistent**

x.store (10);

y.store (20);

assert (y.load () == 20 && x.load () == 0)



assert (y.load () == 0 && x.load () == 10)



# **Atomics – Acquire/Release**

```
a = 1;
x.store(20, memory_order_release);
```

```
if (x.load(memory_order_acquire) ==
   20)
assert (a == 1);
```





## **Atomics**

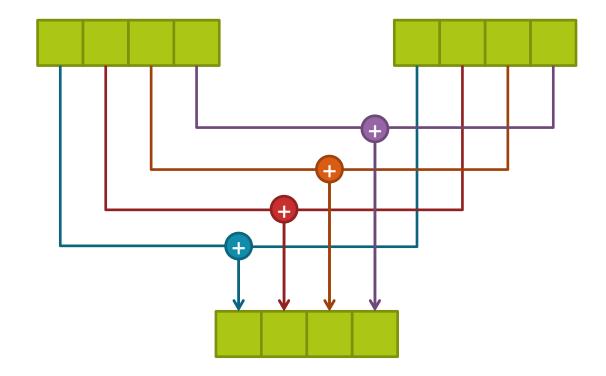
```
int AtomicAdd(volatile int* ptr, int increment)
{
  return __atomic_add_fetch (ptr, increment, __ATOMIC_SEQ_CST);
}
```



# And Now For Something Completely Different...

#### add:

```
vld1.32 {q9}, [r1]!
vld1.32 {q8}, [r2]!
vadd.i32 q8, q9, q8
subs r3, r3, #4
vst1.32 {q8}, [r0]!
bne add
bx Ir
```





```
void add(int *a, const int *b, const int *c, unsigned n)
{
  unsigned i;
  for (i = 0; i < n; ++i)
    a[i] = b[i] + c[i];
}</pre>
```



```
.cpu generic
       .file "t.c"
       .text
       .align 2
       .global add
       .type add, %function
               w3, .L1
       add
               x4, x0, 16
               x1, x4
       add
               x5, x1, 16
               w8, cs
               x0, x5
       cset
               w7, cs
               x5, x2, 16
               x2, x4
               w6, cs
               x0, x5
       cset
               w4, cs
               w5, w8, w7
       orr
       orr
               w4, w6, w4
               w5, w4
               .L3
               w3, 5
       bls
               .L3
       lsr
               w7, w3, 2
       mov
               x4. 0
       lsl
               w6, w7, 2
               w5, w4
       mov
.L9:
               x8, x2, x4
       add
       add
               x9, x1, x4
       ld1
               {v0.4s}, [x8]
               {v1.4s}, [x9]
       ld1
       add
               x8, x0, x4
               v0.4s, v1.4s, v0.4s
               w5, w5, 1
       st1
               \{v0.4s\}, [x8]
               w5, w7
       add
               x4, x4, 16
               .L9
               w3, w6
       cmp
               .L1
       uxtw
               x5, w6
       lsl
               x5, x5, 2
               w8, [x1,x5]
```

```
ldr
                w7, [x2,x5]
        add
                w4, w6, 1
                w7, w8, w7
        str
                w7, [x0,x5]
                w3, w4
        bls
                .L1
                x4, x4, 2, 32
        ldr
                w7, [x1, x4]
        ldr
                w5, [x2,x4]
                w6, w6, 2
        add
                w5, w7, w5
        str
                w5, [x0,x4]
                w3, w6
        bls
                .L1
        uxtw
                x6, w6
        lsl
                x6, x6, 2
                w3, [x1,x6]
                w1, [x2,x6]
        ldr
        add
                w1, w3, w1
        str
                w1, [x0,x6]
.L1:
        ret
.L3:
                w6, w3, #1
        sub
        add
                x6, x6, 1
        lsl
                x6, x6, 2
                x3, 0
.L11:
                w5, [x1,x3]
        ldr
        ldr
                w4, [x2,x3]
                w4, w5, w4
        add
                w4, [x0,x3]
        str
        add
                x3, x3, 4
        cmp
                x3, x6
        bne
                .L11
        ret
               add, .-add
        .ident "GCC: (GNU) 4.9.0 20130416 (experimental)"
```



```
.cpu generic
.file
      "t.c'
                                                             Header
.text
.align 2
.global add
.type add, %function
cbz
       w3, .L1
add
       x4, x0, 16
       x1, x4
cmp
add
       x5, x1, 16
cset
       w8, cs
       x0, x5
       w7, cs
cset
       x5, x2, 16
add
cmp
       x2, x4
       w6, cs
cset
                          Do the arrays overlap?
       w4, cs
cset
       w5, w8, w7
orr
       w4, w6, w4
orr
tst
       w5, w4
       .L3
       w3, 5
bls
       .L3
       w7, w3, 2
1s1
       w6, w7, 2
add
       x8, x2, x4
add
       x9, x1, x4
ld1
      {v0.4s}, [x8]
       {v1.4s}, [x9]
ld1
                                                  Vector Loop
       x8, x0, x4
add
       v0.4s, v1.4s, v0.4s
       w5, w5, 1
       \{v0.4s\}, [x8]
       w5, w7
       x4, x4, 16
add
bcc
       w3, w6
cmp
       .L1
beq
uxtw
       x5, w6
                                              Vector Tidy up
lsl
       x5, x5, 2
       w8, [x1,x5]
```

```
w7, [x2,x5]
              w4, w6, 1
              w7, w8, w7
       str
              w7, [x0,x5]
       cmp
              w3, w4
       bls
              .L1
              x4, x4, 2, 32
       ubfiz
              w7, [x1, x4]
       ldr
              w5, [x2,x4]
              w6, w6, 2
                                                       Vector Tidy up
       add
              w5, w7, w5
       str
              w5, [x0, x4]
              w3, w6
       bls
              .L1
              x6, w6
              x6, x6, 2
              w3, [x1,x6]
              w1, [x2,x6]
       add
              w1, w3, w1
       str
              w1, [x0,x6]
              w6, w3, #1
       sub
       add
              x6, x6, 1
       1s1
              x6, x6, 2
.L11:
                                          Overlap 1-by-1 loop
              w5, [x1,x3]
w4, [x2,x3]
       ldr
       ldr
       add
              w4, w5, w4
              w4, [x0, x3]
       str
              x3, x3, 4
       cmp
              x3, x6
       bne
              .L11
              add, .-add
                                                                           Footer
              "GCC: (GNU) 4.9.0 20130416 (experimental)"
```

```
void add(int * restrict a, const int * restrict b, const int * restrict c, unsigned n)
{
  unsigned i;
  for (i = 0; i < n; ++i)
    a[i] = b[i] + c[i];
}</pre>
```



```
.cpu generic
         .file
         .align
         .global
                      add
                      add, %function
         .type
add:
         cbz
                      w3, .L1
        ubfx
                      x4, x1, 2, 2
                      x4, x4
        neg
                      w4, w4, 3
        and
                      w4, w3
         cmp
                      w6, w4, w3, 1s
        csel
                      w3, 4
         cmp
        mov
                      w4, w3
        bhi
                      .L25
.L3:
         ldr
                      w6, [x1]
                      w5, [x2]
         1dr
        cmp
                     w5, w6, w5
         add
         str
                      w5, [x0]
        bls
                      .L16
                      w6, [x1,4]
         ldr
                      w5, [x2,4]
        1dr
         cmp
                      w4, 2
                      w5, w6, w5
        add
                      w5, [x0,4]
        str
        bls
                      .L17
        ldr
                      w6, [x1,8]
         ldr
                     w5, [x2,8]
w4, 3
        cmp
        add
                      w5, w6, w5
                      w5, [x0,8]
        str
        bls
                      .L18
                      w7, [x1,12]
         ldr
         ldr
                      w6, [x2,12]
                      w5, 4
        mov
        add
                      w6, w7, w6
        str
                      w6, [x0,12]
.L5:
                      w3, w4
         bea
                      .L1
.L4:
        sub
                      w11, w3, w4
                      w9, w11, 2
         lsr
        1s1
                      w10, w9, 2
        cbz
                      w10. .L7
        ubfiz
                      x4, x4, 2, 32
        add
                      x8, x1, x4
        add
                      x7, x2, x4
                      w6, 0
        mov
        add
                      x4, x0, x4
.L13:
        ld1
                      {v1.4s}, [x7],16
```

```
ld1
                      {v0.4s}, [x8],16
         add
                      v0.4s, v1.4s, v0.4s
         add
         st1
                      {v0.4s}, [x4],16
                      w9, w6
         cmp
        bhi
                      .L13
                      w11, w10
         cmp
         add
                      w5, w5, w10
         beq
.L7:
         ubfiz
                      x6, x5, 2, 32
         ldr
                      w8, [x1,x6]
                      w7, [x2,x6]
         ldr
         add
                     w4, w5, 1
         add
                     w7, w8, w7
                     w7, [x0,x6]
         str
                      w3, w4
        bls
                      .L1
         ubfiz
                      x4, x4, 2, 32
         ldr
                     w7, [x1,x4]
                      w6, [x2,x4]
         ldr
         add
                      w5, w5, 2
         add
                      w6, w7, w6
         str
                     w6, [x0,x4]
                     w3, w5
        CMD
        bls
                      .L1
         ubfiz
                      x5, x5, 2, 32
         ldr
                     w3, [x1,x5]
         ldr
                     w1, [x2,x5]
         add
                     w1, w3, w1
                     w1, [x0,x5]
        str
.L1:
         ret
.L25:
         mov
                      w4, 0
                      w5, w4
         mov
                      w6, .L4
         cbz
                      w4, w6
                      .L3
        b
.L18:
                     w5, 3
.L5
.L16:
         mov
                      w5, 1
                      .LŚ
.L17:
                      w5, 2
         mov
                      .L5
         .size
                      add, .-add
                      "GCC: (msgd) 4.8.2 20130805 (prerelease)"
```



```
.file
                                                                                    Header
        .align
        .global
                   add
                   add, %function
        .type
                   w3. .L1
       ubfx
                   x4, x1, 2, 2
                   x4, x4
       neg
                   w4, w4, 3
       and
                   w4, w3
       cmp
                   w6, w4, w3, ls
       csel
                   w3, 4
       cmp
       mov
                   w4, w3
       bhi
                   .L25
.L3:
       ldr
                   w6, [x1]
w5, [x2]
       ldr
                   w4, 1
       cmp
                   w5, w6, w5
       add
       str
                   w5, [x0]
       bls
                   .L16
       ldr
                   w6, [x1,4]
       ldr
                   w5, [x2,4]
                   w4, 2
w5, w6, w5
       cmp
       add
       str
                   w5, [x0,4]
       bls
                   .L17
                                           Peeling for alignment
       ldr
                   w6, [x1,8]
       ldr
                   w5, [x2,8]
       cmp
       add
                   w5, w6, w5
       str
                   w5, [x0,8]
       bls
                    .L18
       1dr
                   w7, [x1,12]
       ldr
                   w6, [x2,12]
       mov
       add
       str
                   w6, [x0,12]
.L5:
       cmp
                   w3, w4
       bea
                   .L1
       sub
                   w11, w3, w4
       1sr
                   w9, w11, 2
                   w10, w9, 2
       cbz
                   w10, .L7
       ubfiz
                   x4, x4, 2, 32
                   x8, x1, x4
       add
                   x7, x2, x4
                                                                        Vector Loop
       ld1
                   {v1.4s}, [x7],16
```

```
ld1
                   {v0.4s}, [x8],16
                   v0.4s, v1.4s, v0.4s
       add
       add
                   w6, w6, 1
                   \{v0.4s\}, [x4], 16
                                                                    Vector Loop
       st1
                   w9, w6
       CMD
       bhi
                   .L13
                   x6, x5, 2, 32
       ldr
                   w8, [x1,x6]
       ldr
                   w7, [x2,x6]
       add
                   w4, w5, 1
       add
                   w7, w8, w7
       str
                   w7, [x0,x6]
       bls
                   .L1
       ubfiz
                   x4, x4, 2, 32
       ldr
                   w7, [x1,x4]
                                                             Vector Tidy up
                   w6, [x2,x4]
       ldr
       add
                   w5, w5, 2
                   w6, w7, w6
       str
                   w6, [x0,x4]
                   w3, w5
       bls
                   .L1
       ubfiz
                   x5, x5, 2, 32
                   w3, [x1,x5]
       ldr
                   w1, [x2,x5]
       add
                   w1, w3, w1
                   w1, [x0,x5]
       str
       mov
                   w4, 0
                   w5, w4
w6, .L4
       mov
       cbz
                   w4, w6
       mov
.L18:
                                            Peeling for alignment
                   w5, 3
.L5
.L16:
                  w5, 1
                   w5, 2
        .size
                                                                                    Footer
                   "GCC: (msgd) 4.8.2 20130805 (prerelease)'
```



```
void add(int * restrict a, const int * restrict b, const int * restrict c, unsigned n)
 unsigned i;
 a = __builtin_assume_aligned (a, 32);
 b = __builtin_assume_aligned (b, 32);
 b = __builtin_assume_aligned (c, 32);
 for (i = 0; i < n; ++i)
  a[i] = b[i] + c[i];
```



```
.cpu
              generic
                                                                                        add
     .file
              "t.c"
                                                                                        add
     .text
     .align
              2
     .global add
                                                                                       bls
                                                                                       ubfiz
      .type
              add, %function
                                                                                       ldr
add:
              w3, .L1
                                                                                       ldr
     cbz
     lsr
                                                                                        add
              w6, w3, 2
     lsl
              w4, w6, 2
                                                                                       add
     cbz
              w4, .L10
                                                                                        str
              w3, 3
     cmp
                                                                                        cmp
     bls
              .L10
                                                                                       bls
     mov
              x5, 0
                                                                                       ubfiz
                                                                                       ldr
              w7, w5
     mov
.L9:
     add
              x8, x2, x5
                                                                                        add
     add
              x9, x1, x5
                                                                                        str
              {v0.4s}, [x8]
                                                                                  .L1:
     ld1
              \{v1.4s\}, [x9]
                                                                                        ret
     add
             x8, x0, x5
                                                                                  .L10:
     add
              v0.4s, v1.4s, v0.4s
                                                                                        mov
     add
              w7, w7, 1
                                                                                        b
     st1
              {v0.4s}, [x8]
                                                                                        .size
              w6, w7
                                                                                        .ident
     cmp
              x5, x5, 16
     add
              .L9
     bhi
              w3, w4
     cmp
              .L1
     bea
.L3:
              x6, w4
     uxtw
     lsl
              x6, x6, 2
     ldr
              w8, [x1, x6]
     ldr
              w7, [x2, x6]
```

```
w5, w4, 1
w7, w8, w7
w7, [x0,x6]
w3, w5
.L1
x5, x5, 2, 32
w7, [x1, x5]
w6, [x2,x5]
w4, w4, 2
w6, w7, w6
w6, [x0, x5]
w3, w4
.L1
x4, x4, 2, 32
w3, [x1, x4]
w1, [x2,x4]
w1, w3, w1
w1, [x0, x4]
w4, 0
.L3
add, .-add
"GCC: (msgd) 4.8.2 20130805 (prerelease)"
```



```
.cpu
            generic
     .file
             "t.c"
     .text
                                                      Header
     .align
     .global add
     .type
            add, %function
add:
            w3, .L1
     cbz
     lsr
            w6, w3, 2
            w4, w6, 2
     1s1
     cbz
            w4, .L10
                                        Function Setup
            w3, 3
            .L10
     h1s
            x5, 0
            w7, w5
     mov
            x8, x2, x5
     add
            x9, x1, x5
     add
     ld1
            {v0.4s}, [x8]
            \{v1.4s\}, [x9]
     ld1
     add
            x8, x0, x5
            v0.4s, v1.4s, v0.4s
     add
                                             Vector Loop
     add
            w7, w7, 1
     st1
            {v0.4s}, [x8]
            w6, w7
            x5, x5, 16
     add
            L9
     bhi
            w3, w4
     bea
             .L1
.L3:
            x6, w4
     uxtw
     lsl
            x6, x6, 2
                                          Vector Tidy up
     ldr
            w8, [x1, x6]
            w7, [x2,x6]
```

```
add
             w5, w4, 1
     add
             w7, w8, w7
     str
             w7, [x0,x6]
     cmp
             w3, w5
     bls
             .L1
     ubfiz
             x5, x5, 2, 32
     ldr
             w7, [x1, x5]
     ldr
             w6, [x2, x5]
             w4, w4, 2
     add
                                          Vector Tidy up
     add
             w6, w7, w6
     str
             w6, [x0, x5]
             w3, w4
     h1s
             .L1
     ubfiz
             x4, x4, 2, 32
     ldr
             w3, [x1, x4]
     ldr
             w1, [x2, x4]
             w1, w3, w1
     add
             w1, [x0, x4]
     str
L1:
.L10:
             w4, 0
                                          Function Setup
     mov
             . L3
             add, .-add
             "GCC: (msgd) 4.8.2 20130805 (prerelease)"
                                                          Footer
```



```
void add(int * restrict a, const int * restrict b, const int * restrict c, unsigned n)
 unsigned i;
 assert (n % 4 == 0)
 a = __builtin_assume_aligned (a, 32);
 b = __builtin_assume_aligned (b, 32);
 b = __builtin_assume_aligned (c, 32);
 for (i = 0; i < n; ++i)
  a[i] = b[i] + c[i];
```



```
#include <arm neon.h>
void add (int * __restrict a, const int * __restrict b, const int * __restrict c, unsigned n)
 unsigned i;
 int32x4 t^* va = (int32x4 t^*)a;
 const int32x4_t^* vb = (const int32x4_t^*)b;
 const int32x4 t^* vc = (const int32x4 t^*)c;
 for (i = 0; i < n; i += 4)
  *(va++) = vaddq s32 (*(vb++), *(vc++));
```

```
generic
                                                ld1
                                                         {v0.4s}, [x2],16
   .cpu
            "t.c"
                                                         v0.4s, v1.4s, v0.4s
   .file
                                                add
                                                add
                                                         w4, w4, 4
   .text
                                                         {v0.4s}, [x0],16
   .align
            2
                                                st1
   .global
            add
                                                         w3, w4
                                                cmp
            add, %function
                                                         .L3
   .type
                                                bhi
add:
                                             .L1:
   cbz
            w3, .L1
                                                ret
            w4. 0
                                                .size
                                                         add. .-add
   mov
                                                         "GCC: (GNU) 4.9.0
.L3:
                                                .ident
                                                20130902 (experimental)"
   ld1
            {v1.4s}, [x1],16
```



```
generic
   .cpu
            "t.c"
   .file
   .text
                               Header
   .align
            2
   .global
            add
            add, %function
   .type
add:
            w3, .L1 Function Setup
   cbz
            w4, 0
   mov
L3:
   ld1
            {v1.4s}, [x1],16Vector Loop
```

```
ld1
            {v0.4s}, [x2],16
           v0.4s, v1.4s, v0.4s
  add
           w4, w4, 4
  add
           {v0.4s}, [x0],16
Vector Loop
  st1
           w3, w4
  cmp
           .L3
  bhi
.L1:
   ret
   .size
           add, .-add
           "GCC: (GNU) 4.9.0Footer
   .ident
  20130902 (experimental)"
```

```
#include <arm neon.h>
void add (int * restrict a, const int * restrict b, const int * restrict c, unsigned n)
 int32x4 t^* va = (int32x4 t^*)a;
 const int32x4 t^* vb = (const int32x4 t^*)b;
 const int32x4_t^* vc = (const int32x4_t^*)c;
 for (; n!= 0; n -= 4)
  *(va++) = vaddq_s32 (*(vb++), *(vc++));
```



```
.cpu generic
   .file "t.c"
   .text
   .align
            2
   .global
           add
           add, %function
   .type
add:
   cbz w3, .L1
.L3:
   ld1 {v1.4s}, [x1],16
   ld1 {v0.4s}, [x2],16
```

```
add v0.4s, v1.4s, v0.4s
  st1 {v0.4s}, [x0],16
  subs w3, w3, #4
  bne .L3
.L1:
   ret
           add, .-add
   .size
  .ident "GCC: (GNU) 4.9.0
  20130902 (experimental)"
```



```
.cpu generic
   .file "t.c"
   .text
                               Header
           2
   .align
   .global
            add
            add, %function
   .type
add:
                        Function Setup
   cbz w3, .L1
.L3:
   ld1 {v1.4s}, [x1],16
                          Vector Loop
   ld1 {v0.4s}, [x2],16
```

```
add v0.4s, v1.4s, v0.4s
  st1 {v0.4s}, [x0],16
  subs w3, w3, #4
                       Vector Loop
  bne .L3
.L1:
  ret
           add, .-add
  .size
          "GCC: (GNU) 4.9.0Footer
   .ident
  20130902 (experimental)"
```

```
.cpu cortex-a9
     .eabi attribute 27, 3
     .eabi attribute 28, 1
     .fpu neon
     .eabi_attribute 20, 1
     .eabi attribute 21, 1
     .eabi attribute 23, 3
     .eabi_attribute 24, 1
     .eabi_attribute 25, 1
     .eabi attribute 26, 1
     .eabi_attribute 30, 2
     .eabi_attribute 34, 1
     .eabi attribute 18, 4
            "t.c"
     .file
     .text
    .align 2
    .global add
     .type add, %function
add:
```

```
@ args = 0, pretend = 0, frame = 0
    @ frame_needed = 0, uses_anonymous_args = 0
    @ link register save eliminated.
                 r3, #0
    cmp
    bxeq
                 lr
.L3:
    subs
                 r3, r3, #4
    vld1.64
                 {d18-d19}, [r1:64]!
    vld1.64
                 {d16-d17}, [r2:64]!
    vadd.i32
                 q8, q9, q8
    vst1.64
                 {d16-d17}, [r0:64]!
                 .L3
    bne
    bx
    .size add, .-add
    .ident "GCC: (GNU) 4.9.0 20130902 (experimental)"
```

	.cpu gene		
	.file	"t.c"	
	.text		Header
	.align	2	1104401
	.global	add 0/five etics	
2 4 4	.type	add, %function	
add:	-1	2 14	
	cbz	w3, .L1	
	Isr	w6, w3, 2	
	lsl	w4, w6, 2	Function Setup
	cbz	w4, .L10	
	cmp	w3, 3	
	bls	.L10	
	mov	x5, 0	
	mov	w7, w5	
.L9:			
	add	x8, x2, x5	·
	add	x9, x1, x5	
	ld1	{v0.4s}, [x8]	
	ld1	{v1.4s}, [x9]	
	add	x8, x0, x5	
	add	v0.4s, v1.4s, v0.4s	\/ootor   oon
	add	w7, w7, 1	Vector Loop
	st1	{v0.4s}, [x8]	
	cmp	w6, w7	
	add	x5, x5, 16	
	bhi	.L9	
	cmp	w3, w4	
	beq	.L1	

```
x6, w4
      uxtw
                 x6, x6, 2
      Isl
                 w8, [x1,x6]
      ldr
                 w7, [x2,x6]
      ldr
      add
                 w5, w4, 1
      add
                 w7, w8, w7
                 w7, [x0,x6]
      str
                 w3, w5
      cmp
      bls
                 .L1
                 x5, x5, 2, 32
      ubfiz
      ldr
                 w7, [x1,x5]
                                            Vector Tidy up
      ldr
                 w6, [x2,x5]
      add
                 w4, w4, 2
      add
                 w6, w7, w6
                 w6, [x0,x5]
      str
                 w3, w4
      cmp
      bls
                 .L1
      ubfiz
                 x4, x4, 2, 32
      ldr
                 w3, [x1,x4]
      ldr
                 w1, [x2,x4]
      add
                 w1, w3, w1
                 w1, [x0,x4]
      str
.L10:
                                             Function Setup
                 w4, 0
      mov
                 .L3
                 add, .-add
      .size
                                                              Footer
                 "GCC: (msgd) 4.8.2 20130805 (prerelease)"
      .ident
```



```
#include <arm neon.h>
void add (int * restrict a, const int * restrict b, const int * restrict c, unsigned n)
 int32x4 t^* va = (int32x4 t^*)a;
 const int32x4 t^* vb = (const int32x4 t^*)b;
 const int32x4_t^* vc = (const int32x4_t^*)c;
 for (; n!= 0; n -= 4)
  *(va++) = vaddq_s32 (*(vb++), *(vc++));
```





More about Linaro: <a href="http://www.linaro.org/about/">http://www.linaro.org/about/</a>

More about Linaro engineering: <a href="http://www.linaro.org/engineering/">http://www.linaro.org/engineering/</a>

How to join: <a href="http://www.linaro.org/about/how-to-join">http://www.linaro.org/about/how-to-join</a>

Linaro members: <a href="www.linaro.org/members">www.linaro.org/members</a>