Efficient Verification of Optimized Code

Marc Schoolderman, Jonathan Moerman, Sjaak Smetsers, Marko van Eekelen

NWO Applied and Engineering Sciences

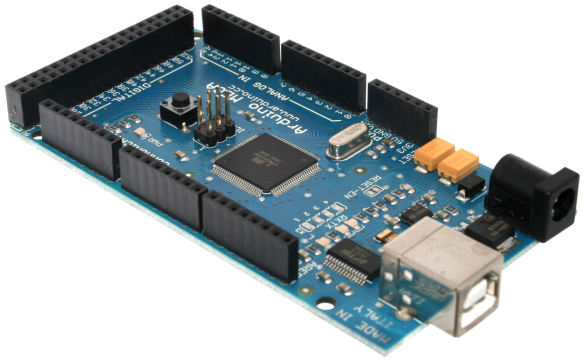
iCIS | Digital Security
Radboud University

Cryptographic on Embedded Devices

Verification target

Public Key Crypto on microcontrollers

- AVR: limited to 8-bit operations
- X25519
  - RFC 7748
  - NIST SP 800-186 (draft)
- Verify the fastest AVR implementation!



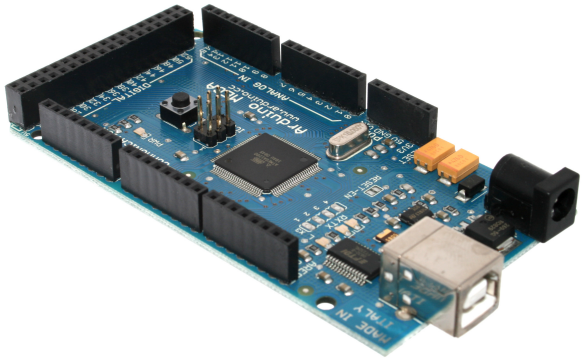
(image credit: oomlout, CC BY-SA 2.0, via Wikimedia Commons)

Cryptographic on Embedded Devices

Verification target

Public Key Crypto on microcontrollers

- AVR: limited to 8-bit operations
- X25519
  - RFC 7748
  - NIST SP 800-186 (draft)
- Verify the fastest AVR implementation!



Challenges

- Can we work with this type of code?
- How to express its full specification?

(image credit: oomlout, CC BY-SA 2.0, via Wikimedia Commons)

Elliptic Curve Cryptography

Public Key Crypto

Alice and Bob want to create a shared secret.

Elliptic Curve Cryptography

Public Key Crypto

Alice and Bob want to create a shared secret.

Alice		Bob
x	$\Longrightarrow$	$x \cdot P$

Elliptic Curve Cryptography

Public Key Crypto

Alice and Bob want to create a shared secret.

Alice		Bob
x	$\Longrightarrow$	$x \cdot P$
$y \cdot P$	$\Longleftarrow$	y

Elliptic Curve Cryptography

Public Key Crypto

Alice and Bob want to create a shared secret.

Alice		Bob
x	$\Longrightarrow$	$x \cdot P$
$y \cdot P$	$\Longleftarrow$	y
$xy \cdot P$	$=$	$xy \cdot P$

Elliptic Curve Cryptography

Public Key Crypto
Alice and Bob want to create a shared secret.

For secrecy: $x \cdot P$, $y \cdot P$ must not reveal $xy \cdot P$

Alice		Bob
x	$\Longrightarrow$	$x \cdot P$
$y \cdot P$	$\Longleftarrow$	y
$xy \cdot P$	$=$	$xy \cdot P$

Elliptic Curve Cryptography

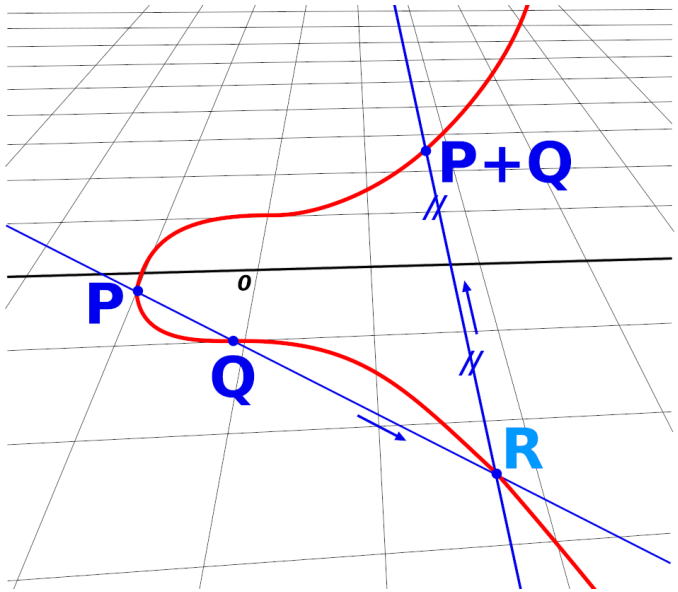
Public Key Crypto
Alice and Bob want to create a shared secret.

For secrecy: $x \cdot P$, $y \cdot P$ must not reveal $xy \cdot P$

Alice		Bob
x	$\Longrightarrow$	$x \cdot P$
$y \cdot P$	$\Longleftarrow$	y
$xy \cdot P$	$=$	$xy \cdot P$

X25519
Do this on the elliptic curve Curve25519!

- Multiplication by repeated addition
- $x, y \in \mathbb{F}_{p^2}$ with $p = 2^{255} - 19$
- Compute only x-coordinates



(illustration: Jean Brette,CC BY 3.0, via Wikimedia Commons)

Conventions of Cryptographic Code

Calculation must be efficient

✓Hand-written machine code

Conventions of Cryptographic Code

Calculation must be efficient

✓Hand-written machine code

Calculation must not reveal secrets

- Countermeasure: *"constant time"*
- Countermeasure: *"predictable memory access"*

✓Static analysis

Conventions of Cryptographic Code

Calculation must be efficient
✓Hand-written machine code

Calculation must not reveal secrets

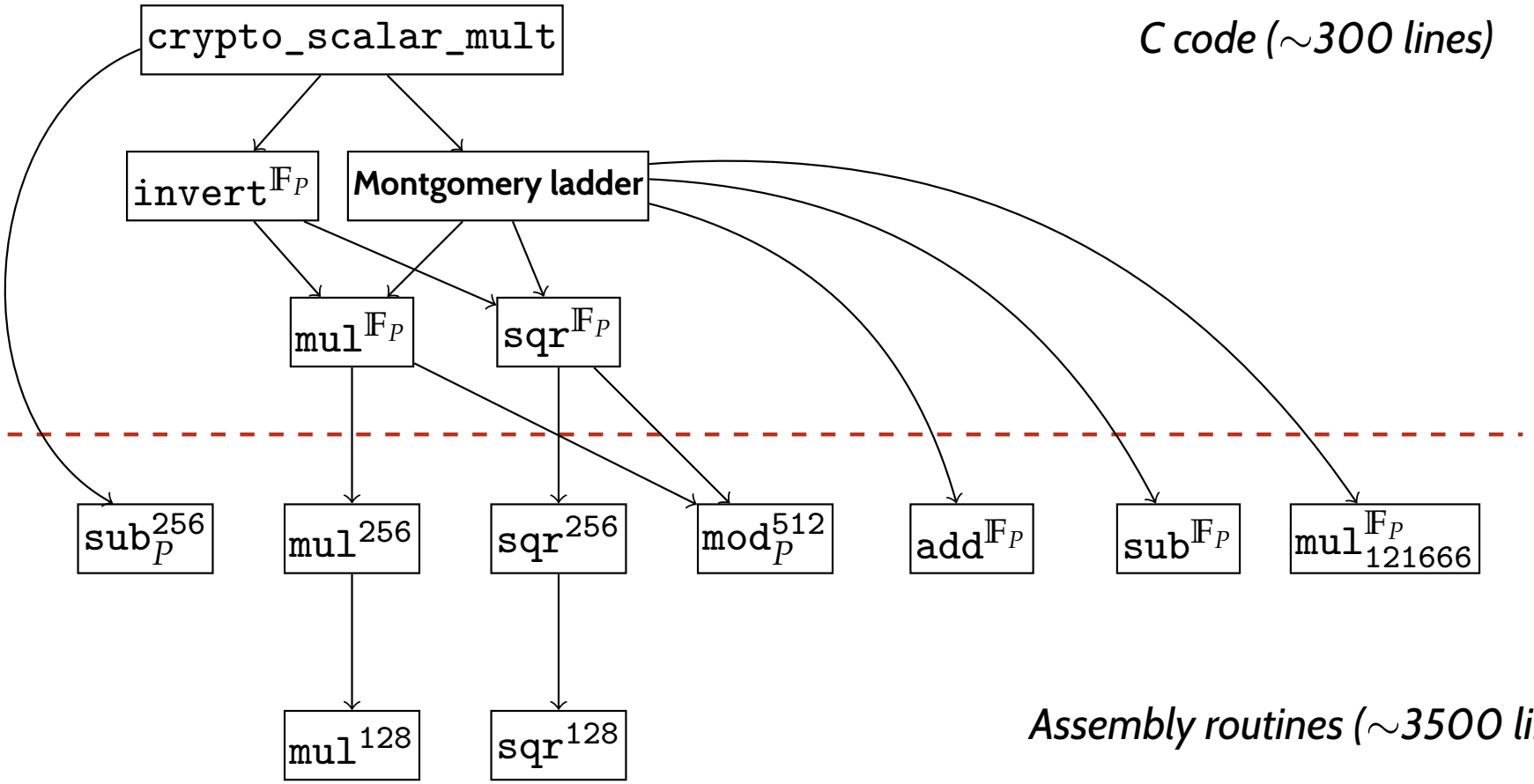
- Countermeasure: *"constant time"*
- Countermeasure: *"predictable memory access"*

✓Static analysis

Code must be utterly correct
Bugs with low probability *will* be exploited

- Countermeasure: *"no bugs"* ?

X25519 on AVR



Bottom-up Approach

The whole is mostly the sum of its parts!

Assembly code

- Model AVR instruction set
- Translate assembly code to this model

Simple specifications, complex code

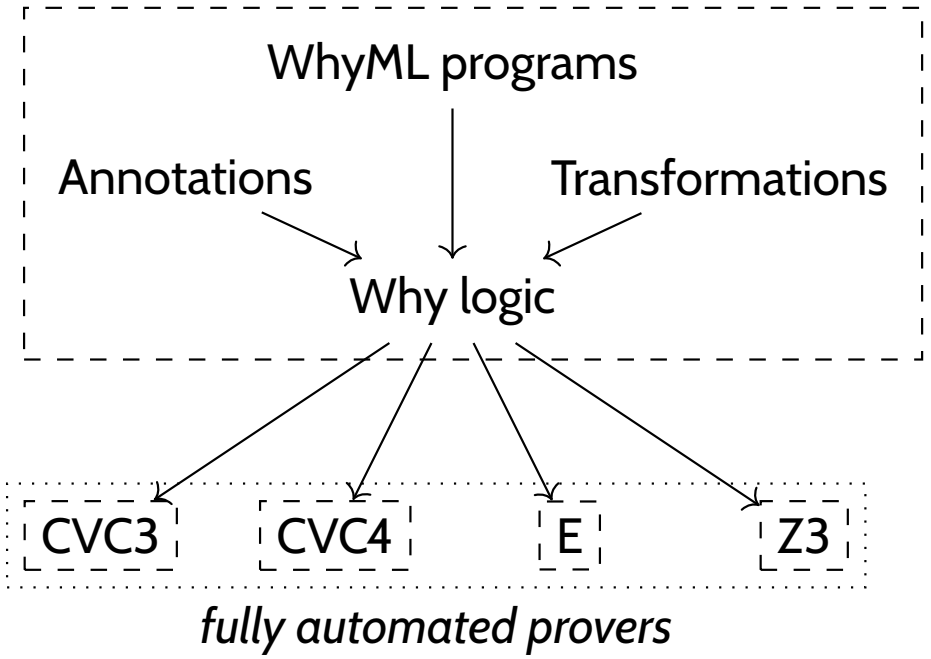
C code

- Model C code in a compatible way
- Add specifications for assembly subroutines

Simple code, complex specifications

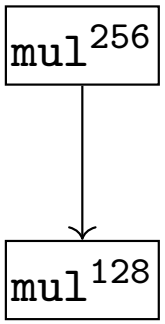
Why3: platform for program verification

General-purpose tool, used without modifications

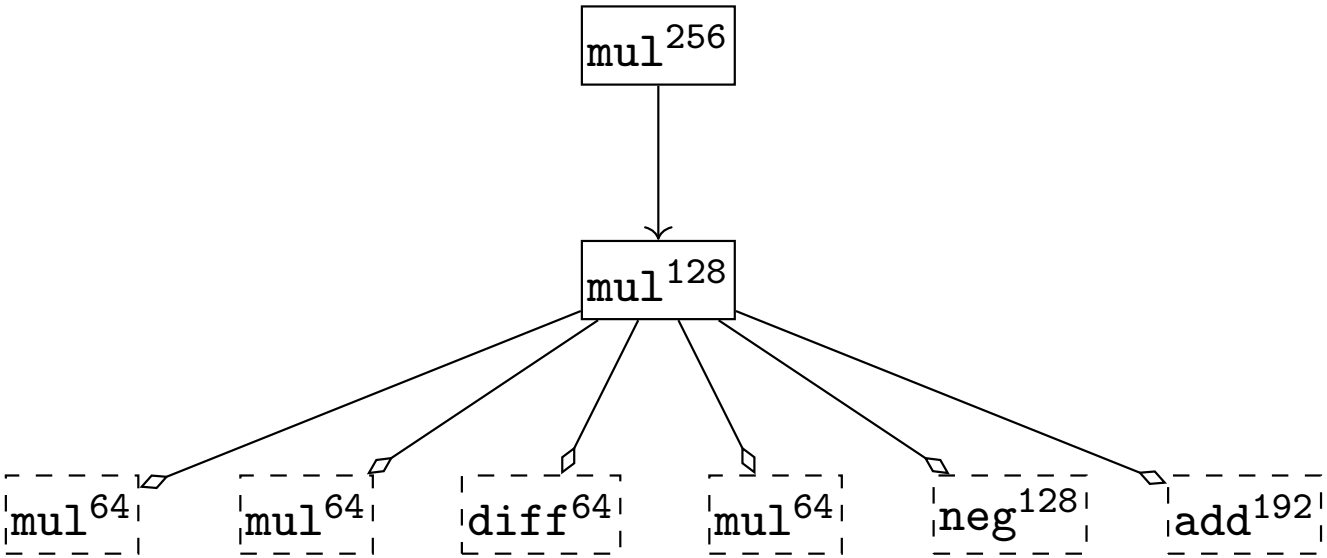


Abstraction, bit-vector theories, type invariants, ghost code …

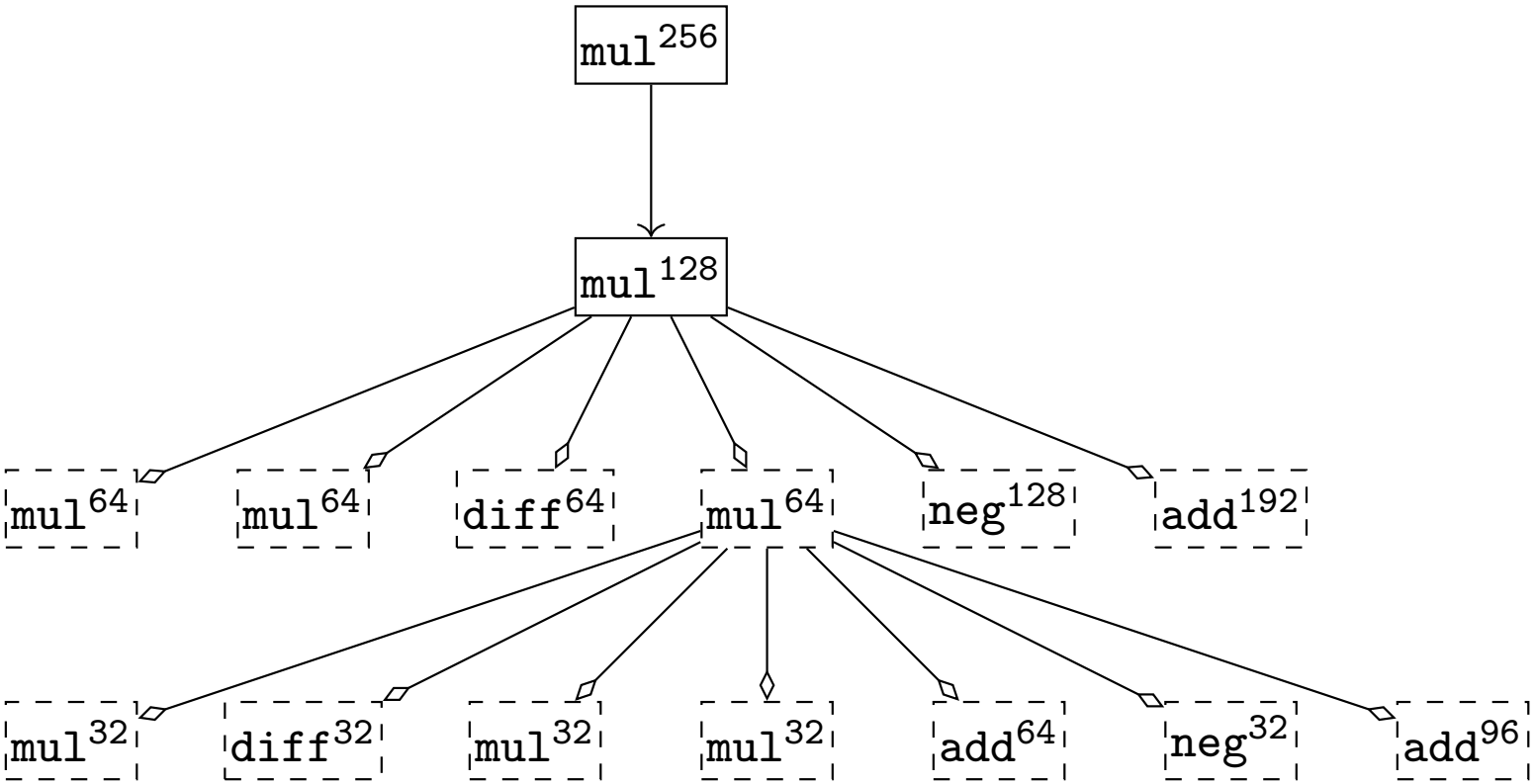
Dissecting Assembly Code



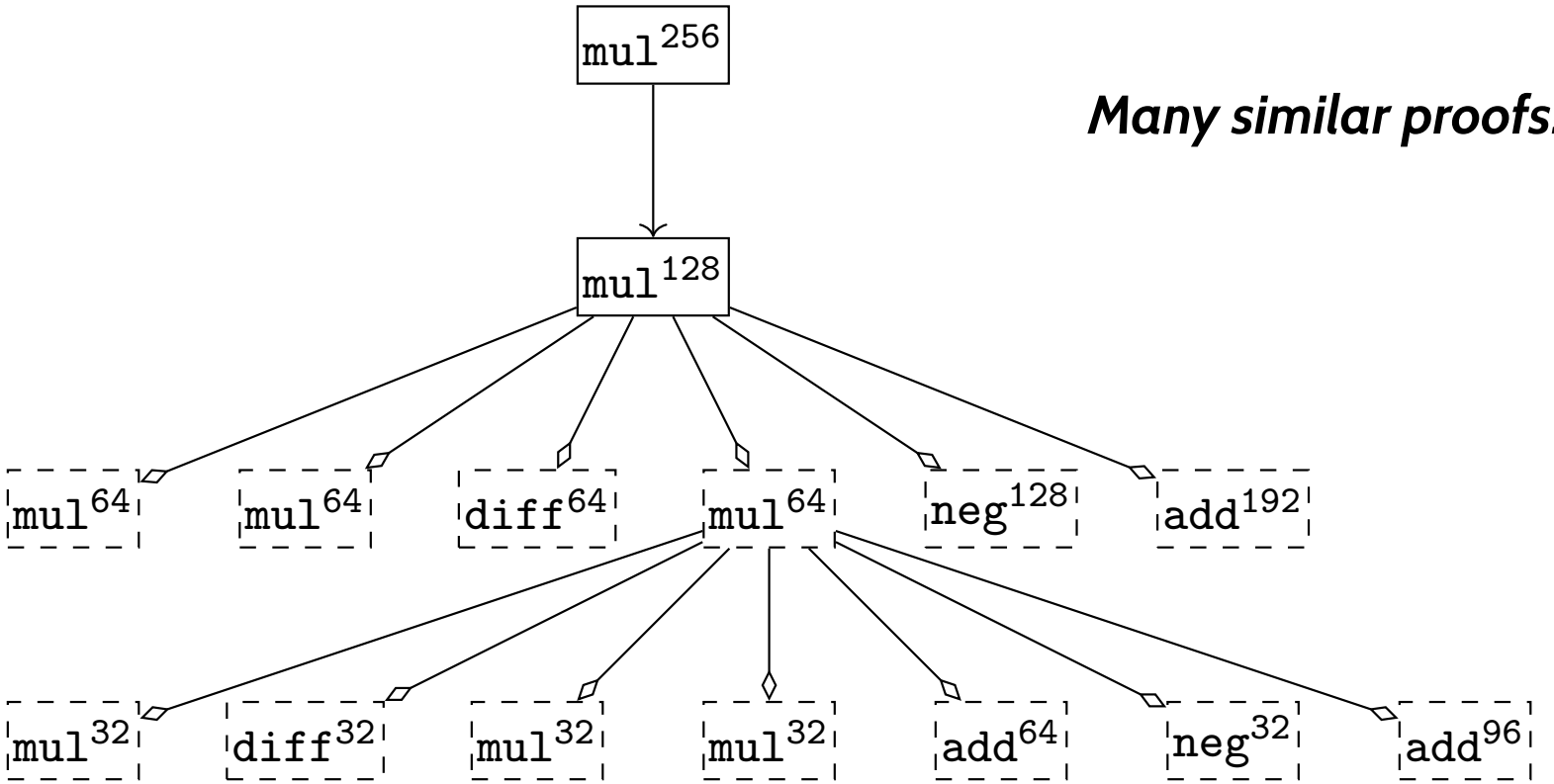
Dissecting Assembly Code



Dissecting Assembly Code

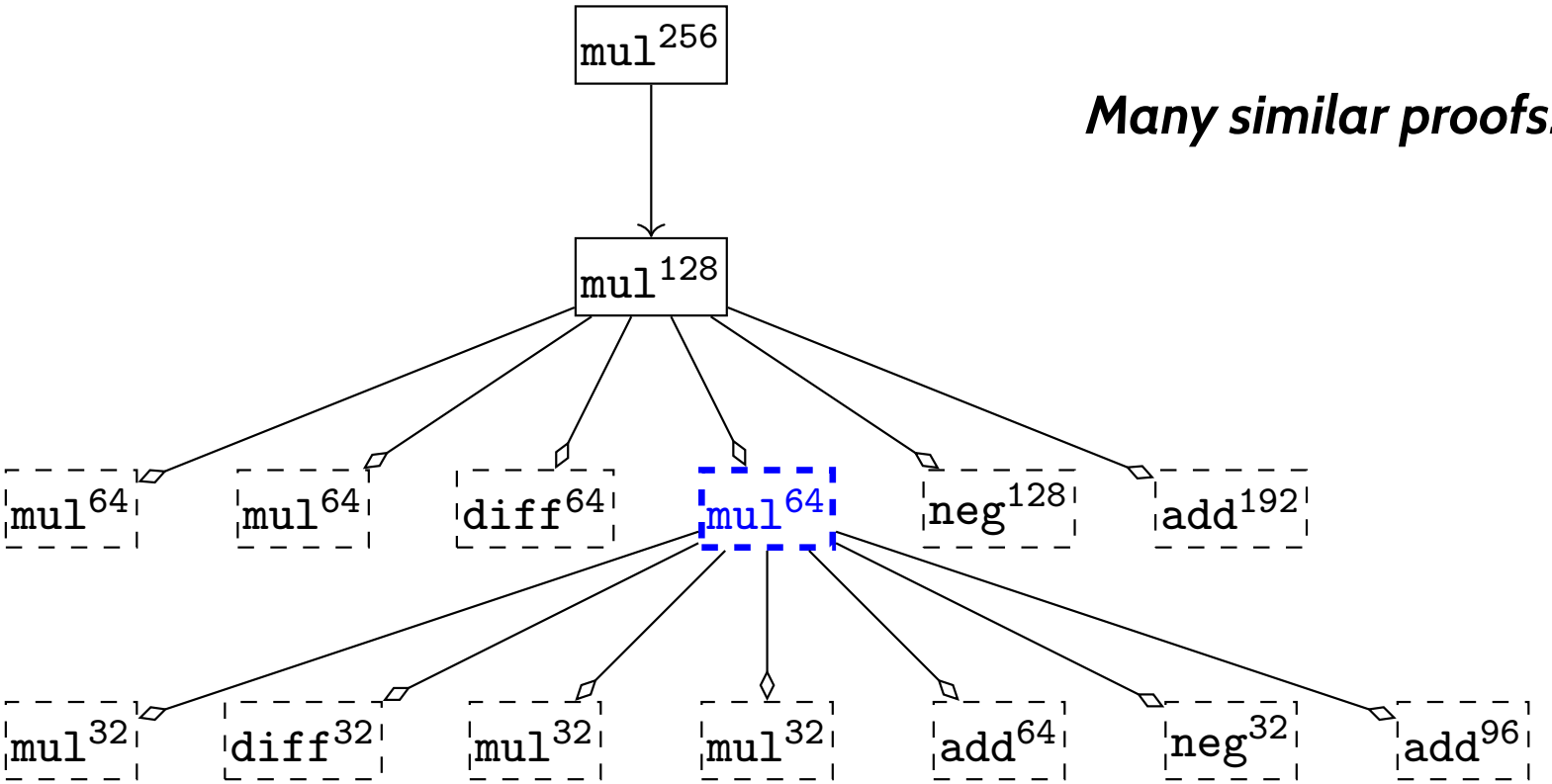


Dissecting Assembly Code



Many similar proofs!

Dissecting Assembly Code



Many similar proofs!

Divide and Conquer

```
clr r20       mul r4, r9    adc r1, r21   eor r6, r1    mul r19, r23 adc r29, r1  mul r3, r7    adc r2, r27   eor r23, r27
clr r21       add r14, r19  add r15, r0   eor r7, r1    add r16, r0  adc r18, r26 add r22, r0  mul r5, r7    eor r24, r27
movw r16, r20 adc r15, r0   adc r16, r1   eor r8, r1    adc r17, r1  mul r21, r23 adc r23, r1  add r24, r0   eor r25, r27
ld r2, X+     adc r16, r1   adc r17, r21  eor r9, r1    adc r28, r26 add r28, r0  adc r24, r26 adc r25, r1   eor r2 , r27
ld r3, X+     mul r4, r8    ldd r22 Y+4   sub r2, r0    mul r20, r22 adc r29, r1  mul r4, r6    adc r2, r27   eor r3 , r27
ld r4, X+     movw r18, r0  ldd r23 Y+5   sbc r3, r0    add r16, r0  adc r18, r26 add r22, r0  mul r4, r9    adc r10, r20
ld r5, X+     mul r4, r6    ldd r24 Y+6   sbc r4, r0    adc r17, r1  mul r20, r25 adc r23, r1  add r25, r0   adc r11, r21
ldd r6 Y+0    add r12, r0   ldd r25 Y+7   sbc r5, r0    adc r28, r26 add r29, r0  adc r24, r26 adc r2, r1    adc r12, r22
ldd r7 Y+1    adc r13, r1   movw r28, r20 sub r6, r1    clr r29       adc r18, r1  mul r2, r9    adc r3, r27   adc r13, r23
ldd r8 Y+2    adc r14, r18  ld r18, X+    sbc r7, r1    mul r18, r25 adc r19, r26 add r23, r0  mul r5, r8    adc r14, r24
ldd r9 Y+3    adc r19, r21  ld r19, X+    sbc r8, r1    add r17, r0  mul r21, r24 adc r24, r1  add r25, r0   adc r15, r25
mul r2, r8    mul r3, r8    ld r20, X+    sbc r9, r1    adc r28, r1  add r29, r0  adc r25, r26 adc r2, r1    adc r16, r2
movw r12, r0  add r13, r0   ld r21, X+    eor r0, r1    adc r29, r26 adc r18, r1  mul r3, r8    adc r3, r27   adc r17, r3
mul r2, r6    adc r14, r1   movw r26, r28 bst r0, 0     mul r19, r24 adc r19, r26 add r23, r0  mul r5, r9    adc r28, r26
movw r10, r0  adc r19, r21  std Z+0, r10  mul r18, r22 add r17, r0  mul r21, r25 adc r24, r1  add r2, r0    adc r29, r0
mul r2, r7    mul r5, r9    std Z+1, r11  add r14, r0   adc r28, r1  add r18, r0  adc r25, r26 adc r3, r1    adc r18, r0
add r11, r0   add r15, r19  std Z+2, r12  adc r15, r1   adc r29, r26 adc r19, r1  mul r4, r7    add r10, r14  adc r19, r0
adc r12, r1   adc r16, r0   std Z+3, r13  adc r16, r26  mul r20, r23 mul r2, r6   add r23, r0  adc r11, r15  std Z+4, r10
adc r13, r21  adc r17, r1   sub r2, r18   adc r29, r26  add r17, r0  movw r20, r0 adc r24, r1  adc r12, r16  std Z+5, r11
mul r3, r9    mul r5, r7    sbc r3, r19   mul r18, r23  adc r28, r1  movw r22, r26adc r25, r26 adc r13, r17  std Z+6, r12
movw r14, r0  movw r18, r0  sbc r4, r20   add r15, r0   adc r29, r26 mul r2, r7   mul r5, r6   adc r14, r28  std Z+7, r13
mul r2, r9    mul r4, r7    sbc r5, r21   adc r16, r1   mul r21, r22 add r21, r0  add r23, r0  adc r15, r29  std Z+8, r14
movw r18, r0  add r13, r0   sbc r0, r0    adc r29, r26  add r17, r0  adc r22, r1  adc r24, r1  adc r16, r18  std Z+9, r15
mul r3, r6    adc r18, r1   sub r6, r22   mul r19, r22  adc r28, r1  mul r3, r6   adc r25, r26 adc r17, r19  std Z+10, r16
add r11, r0   adc r19, r21  sbc r7, r23   add r15, r0   adc r29, r26 add r21, r0  mul r3, r9   bld r27, 0    std Z+11, r17
adc r12, r1   mul r5, r6    sbc r8, r24   adc r16, r1   mul r19, r25 adc r22, r1  movw r2, r26 dec 27        std Z+12, r28
adc r13, r18  add r13, r0   sbc r9, r25   adc r17, r29  movw r18, r26adc r23, r26 add r24, r0  adc r26, r27  std Z+13, r29
adc r19, r21  adc r18, r1   sbc r1, r1    adc r28, r26  add r28, r0  movw r24, r26adc r25, r1  mov r0, r26   std Z+14, r18
mul r3, r7    adc r19, r21  eor r2, r0    mul r18, r24  adc r29, r1  mul r2, r8   adc r2, r27  asr r0        std Z+15, r19
add r12, r0   mul r5, r8    eor r3, r0    add r16, r0   adc r18, r26 add r22, r0  mul r4, r8   eor r20, r27
adc r13, r1   add r14, r18  eor r4, r0    adc r17, r1   mul r20, r24 adc r23, r1  add r24, r0  eor r21, r27
adc r19, r21  adc r0, r19   eor r5, r0    adc r28, r26  add r28, r0  adc r24, r26 adc r25, r1  eor r22, r27
```

Divide and Conquer

"Compute $L = A_l \cdot B_l$"

```
clr r20       mul r4, r9    adc r1, r21   eor r6, r1    mul r19, r23 adc r29, r1  mul r3, r7    adc r2, r27   eor r23, r27
clr r21       add r14, r19  add r15, r0   eor r7, r1    add r16, r0  adc r18, r26 add r22, r0  mul r5, r7    eor r24, r27
movw r16, r20 adc r15, r0   adc r16, r1   eor r8, r1    adc r17, r1  mul r21, r23 adc r23, r1  add r24, r0   eor r25, r27
ld r2, X+     adc r16, r1   adc r17, r21  eor r9, r1    adc r28, r26 add r28, r0  adc r24, r26 adc r25, r1   eor r2 , r27
ld r3, X+     mul r4, r8    ldd r22 Y+4   sub r2, r0    mul r20, r22 adc r29, r1  mul r4, r6    adc r2, r27   eor r3 , r27
ld r4, X+     movw r18, r0  ldd r23 Y+5   sbc r3, r0    add r16, r0  adc r18, r26 add r22, r0  mul r4, r9    adc r10, r20
ld r5, X+     mul r4, r6    ldd r24 Y+6   sbc r4, r0    adc r17, r1  mul r20, r25 adc r23, r1  add r25, r0   adc r11, r21
ldd r6 Y+0    add r12, r0   ldd r25 Y+7   sbc r5, r0    adc r28, r26 add r29, r0  adc r24, r26 adc r2, r1    adc r12, r22
ldd r7 Y+1    adc r13, r1   movw r28, r20 sub r6, r1    clr r29       adc r18, r1  mul r2, r9    adc r3, r27   adc r13, r23
ldd r8 Y+2    adc r14, r18  ld r18, X+    sbc r7, r1    mul r18, r25 adc r19, r26 add r23, r0  mul r5, r8    adc r14, r24
ldd r9 Y+3    adc r19, r21  ld r19, X+    sbc r8, r1    add r17, r0  mul r21, r24 adc r24, r1  add r25, r0   adc r15, r25
mul r2, r8    mul r3, r8    ld r20, X+    sbc r9, r1    adc r28, r1  add r29, r0  adc r25, r26 adc r2, r1    adc r16, r2
movw r12, r0  add r13, r0   ld r21, X+    eor r0, r1    adc r29, r26 adc r18, r1  mul r3, r8    adc r3, r27   adc r17, r3
mul r2, r6    adc r14, r1   movw r26, r28 bst r0, 0     mul r19, r24 adc r19, r26 add r23, r0  mul r5, r9    adc r28, r26
movw r10, r0  adc r19, r21  std Z+0, r10  mul r18, r22 add r17, r0  mul r21, r25 adc r24, r1  add r2, r0    adc r29, r0
mul r2, r7    mul r5, r9    std Z+1, r11  add r14, r0   adc r28, r1  add r18, r0  adc r25, r26 adc r3, r1    adc r18, r0
add r11, r0   add r15, r19  std Z+2, r12  adc r15, r1   adc r29, r26 adc r19, r1  mul r4, r7    add r10, r14  adc r19, r0
adc r12, r1   adc r16, r0   std Z+3, r13  adc r16, r26  mul r20, r23 mul r2, r6    add r23, r0   adc r11, r15  std Z+4, r10
adc r13, r21  adc r17, r1   sub r2, r18   adc r29, r26  add r17, r0  movw r20, r0  adc r24, r1   adc r12, r16  std Z+5, r11
mul r3, r9    mul r5, r7    sbc r3, r19   mul r18, r23  adc r28, r1  movw r22, r26 adc r25, r26  adc r13, r17  std Z+6, r12
movw r14, r0  movw r18, r0  sbc r4, r20   add r15, r0   adc r29, r26 mul r2, r7    mul r5, r6    adc r14, r28  std Z+7, r13
mul r2, r9    mul r4, r7    sbc r5, r21   adc r16, r1   mul r21, r22 add r21, r0  add r23, r0   adc r15, r29  std Z+8, r14
movw r18, r0  add r13, r0   sbc r0, r0    adc r29, r26  add r17, r0  adc r22, r1  adc r24, r1   adc r16, r18  std Z+9, r15
mul r3, r6    adc r18, r1   sub r6, r22   mul r19, r22  adc r28, r1  mul r3, r6    adc r25, r26  adc r17, r19  std Z+10, r16
add r11, r0   adc r19, r21  sbc r7, r23   add r15, r0   adc r29, r26 add r21, r0  mul r3, r9    bld r27, 0    std Z+11, r17
adc r12, r1   mul r5, r6    sbc r8, r24   adc r16, r1   mul r19, r25 adc r22, r1  movw r2, r26  dec 27        std Z+12, r28
adc r13, r18  add r13, r0   sbc r9, r25   adc r17, r29  movw r18, r26 adc r23, r26 add r24, r0   adc r26, r27  std Z+13, r29
adc r19, r21  adc r18, r1   sbc r1, r1    adc r28, r26  add r28, r0  movw r24, r26 adc r25, r1   mov r0, r26   std Z+14, r18
mul r3, r7    adc r19, r21  eor r2, r0    mul r18, r24  adc r29, r1  mul r2, r8    adc r2, r27   asr r0         std Z+15, r19
add r12, r0   mul r5, r8    eor r3, r0    add r16, r0   adc r18, r26 add r22, r0  mul r4, r8    eor r20, r27
adc r13, r1   add r14, r18  eor r4, r0    adc r17, r1   mul r20, r24 adc r23, r1  add r24, r0   eor r21, r27
adc r19, r21  adc r0, r19   eor r5, r0    adc r28, r26  add r28, r0  adc r24, r26 adc r25, r1   eor r22, r27
```

Divide and Conquer

"Compute $|A_l - A_h|$ and $|B_l - B_h|$"

```
clr r20       mul r4, r9    adc r1, r21   eor r6, r1    mul r19, r23  adc r29, r1   mul r3, r7    adc r2, r27   eor r23, r27
clr r21       add r14, r19  add r15, r0   eor r7, r1    add r16, r0   adc r18, r26  add r22, r0   mul r5, r7    eor r24, r27
movw r16, r20 adc r15, r0   adc r16, r1   eor r8, r1    adc r17, r1   mul r21, r23  adc r23, r1   add r24, r0   eor r25, r27
ld r2, X+     adc r16, r1   adc r17, r21  eor r9, r1    adc r28, r26  add r28, r0   adc r24, r26  adc r25, r1   eor r2 , r27
ld r3, X+     mul r4, r8    ldd r22 Y+4   sub r2, r0    mul r20, r22  adc r29, r1   mul r4, r6    adc r2, r27   eor r3 , r27
ld r4, X+     movw r18, r0  ldd r23 Y+5   sbc r3, r0    add r16, r0   adc r18, r26  add r22, r0   mul r4, r9    adc r10, r20
ld r5, X+     mul r4, r6    ldd r24 Y+6   sbc r4, r0    adc r17, r1   mul r20, r25  adc r23, r1   add r25, r0   adc r11, r21
ldd r6 Y+0    add r12, r0   ldd r25 Y+7   sbc r5, r0    adc r28, r26  add r29, r0   adc r24, r26  adc r2, r1    adc r12, r22
ldd r7 Y+1    adc r13, r1   movw r28, r20 sub r6, r1    clr r29       adc r18, r1   mul r2, r9    adc r3, r27   adc r13, r23
ldd r8 Y+2    adc r14, r18  ld r18, X+    sbc r7, r1    mul r18, r25  adc r19, r26  add r23, r0   mul r5, r8    adc r14, r24
ldd r9 Y+3    adc r19, r21  ld r19, X+    sbc r8, r1    add r17, r0   mul r21, r24  adc r24, r1   add r25, r0   adc r15, r25
mul r2, r8    mul r3, r8    ld r20, X+    sbc r9, r1    adc r28, r1   add r29, r0   adc r25, r26  adc r2, r1    adc r16, r2
movw r12, r0  add r13, r0   ld r21, X+    eor r0, r1    adc r29, r26  adc r18, r1   mul r3, r8    adc r3, r27   adc r17, r3
mul r2, r6    adc r14, r1   movw r26, r28 bst r0, 0     mul r19, r24  adc r19, r26  add r23, r0   mul r5, r9    adc r28, r26
movw r10, r0  adc r19, r21  std Z+0, r10  mul r18, r22  add r17, r0   mul r21, r25  adc r24, r1   add r2, r0    adc r29, r0
mul r2, r7    mul r5, r9    std Z+1, r11  add r14, r0   adc r28, r1   add r18, r0   adc r25, r26  adc r3, r1    adc r18, r0
add r11, r0   add r15, r19  std Z+2, r12  adc r15, r1   adc r29, r26  adc r19, r1   mul r4, r7    add r10, r14  adc r19, r0
adc r12, r1   adc r16, r0   std Z+3, r13  adc r16, r26  mul r20, r23  mul r2, r6    add r23, r0   adc r11, r15  std Z+4, r10
adc r13, r21  adc r17, r1   sub r2, r18   adc r29, r26  add r17, r0   movw r20, r0  adc r24, r1   adc r12, r16  std Z+5, r11
mul r3, r9    mul r5, r7    sbc r3, r19   mul r18, r23  adc r28, r1   movw r22, r26 adc r25, r26  adc r13, r17  std Z+6, r12
movw r14, r0  movw r18, r0  sbc r4, r20   add r15, r0   adc r29, r26  mul r2, r7    mul r5, r6    adc r14, r28  std Z+7, r13
mul r2, r9    mul r4, r7    sbc r5, r21   adc r16, r1   mul r21, r22  add r21, r0   add r23, r0   adc r15, r29  std Z+8, r14
movw r18, r0  add r13, r0   sbc r0, r0    adc r29, r26  add r17, r0   adc r22, r1   adc r24, r1   adc r16, r18  std Z+9, r15
mul r3, r6    adc r18, r1   sub r6, r22   mul r19, r22  adc r28, r1   mul r3, r6    adc r25, r26  adc r17, r19  std Z+10, r16
add r11, r0   adc r19, r21  sbc r7, r23   add r15, r0   adc r29, r26  add r21, r0   mul r3, r9    bld r27, 0    std Z+11, r17
adc r12, r1   mul r5, r6    sbc r8, r24   adc r16, r1   mul r19, r25  adc r22, r1   movw r2, r26  dec 27        std Z+12, r28
adc r13, r18  add r13, r0   sbc r9, r25   adc r17, r29  movw r18, r26 adc r23, r26  add r24, r0   adc r26, r27  std Z+13, r29
adc r19, r21  adc r18, r1   sbc r1, r1    adc r28, r26  add r28, r0   movw r24, r26 adc r25, r1   mov r0, r26  std Z+14, r18
mul r3, r7    adc r19, r21  eor r2, r0    mul r18, r24  adc r29, r1   mul r2, r8    adc r2, r27   asr r0        std Z+15, r19
add r12, r0   mul r5, r8    eor r3, r0    add r16, r0   adc r18, r26  add r22, r0   mul r4, r8    eor r20, r27
adc r13, r1   add r14, r18  eor r4, r0    adc r17, r1   mul r20, r24  adc r23, r1   add r24, r0   eor r21, r27
adc r19, r21  adc r0, r19   eor r5, r0    adc r28, r26  add r28, r0   adc r24, r26  adc r25, r1   eor r22, r27
```

Divide and Conquer

"Compute $H = A_h \cdot B_h$ and $L + 2^n \cdot H$"

```
clr r20       mul r4, r9    adc r1, r21   eor r6, r1    mul r19, r23  adc r29, r1   mul r3, r7    adc r2, r27   eor r23, r27
clr r21       add r14, r19  add r15, r0   eor r7, r1    add r16, r0   adc r18, r26  add r22, r0   mul r5, r7    eor r24, r27
movw r16, r20 adc r15, r0   adc r16, r1   eor r8, r1    adc r17, r1   mul r21, r23  adc r23, r1   add r24, r0   eor r25, r27
ld r2, X+     adc r16, r1   adc r17, r21  eor r9, r1    adc r28, r26  add r28, r0   adc r24, r26  adc r25, r1   eor r2 , r27
ld r3, X+     mul r4, r8    ldd r22 Y+4   sub r2, r0    mul r20, r22  adc r29, r1   mul r4, r6    adc r2, r27   eor r3 , r27
ld r4, X+     movw r18, r0  ldd r23 Y+5   sbc r3, r0    add r16, r0   adc r18, r26  add r22, r0   mul r4, r9    adc r10, r20
ld r5, X+     mul r4, r6    ldd r24 Y+6   sbc r4, r0    adc r17, r1   mul r20, r25  adc r23, r1   add r25, r0   adc r11, r21
ldd r6 Y+0    add r12, r0   ldd r25 Y+7   sbc r5, r0    adc r28, r26  add r29, r0   adc r24, r26  adc r2, r1    adc r12, r22
ldd r7 Y+1    adc r13, r1   movw r28, r20 sub r6, r1    clr r29       adc r18, r1   mul r2, r9    adc r3, r27   adc r13, r23
ldd r8 Y+2    adc r14, r18  ld r18, X+    sbc r7, r1    mul r18, r25  adc r19, r26  add r23, r0   mul r5, r8    adc r14, r24
ldd r9 Y+3    adc r19, r21  ld r19, X+    sbc r8, r1    add r17, r0   mul r21, r24  adc r24, r1   add r25, r0   adc r15, r25
mul r2, r8    mul r3, r8    ld r20, X+    sbc r9, r1    adc r28, r1   add r29, r0   adc r25, r26  adc r2, r1    adc r16, r2
movw r12, r0  add r13, r0   ld r21, X+    eor r0, r1    adc r29, r26  adc r18, r1   mul r3, r8    adc r3, r27   adc r17, r3
mul r2, r6    adc r14, r1   movw r26, r28 bst r0, 0     mul r19, r24  adc r19, r26  add r23, r0   mul r5, r9    adc r28, r26
movw r10, r0  adc r19, r21  std Z+0, r10  mul r18, r22  add r17, r0   mul r21, r25  adc r24, r1   add r2, r0    adc r29, r0
mul r2, r7    mul r5, r9    std Z+1, r11  add r14, r0   adc r28, r1   add r18, r0   adc r25, r26  adc r3, r1    adc r18, r0
add r11, r0   add r15, r19  std Z+2, r12  adc r15, r1   adc r29, r26  adc r19, r1   mul r4, r7    add r10, r14  adc r19, r0
adc r12, r1   adc r16, r0   std Z+3, r13  adc r16, r26  mul r20, r23  mul r2, r6    add r23, r0   adc r11, r15  std Z+4, r10
adc r13, r21  adc r17, r1   sub r2, r18   adc r29, r26  add r17, r0   movw r20, r0  adc r24, r1   adc r12, r16  std Z+5, r11
mul r3, r9    mul r5, r7    sbc r3, r19   mul r18, r23  adc r28, r1   movw r22, r26 adc r25, r26  adc r13, r17  std Z+6, r12
movw r14, r0  movw r18, r0  sbc r4, r20   add r15, r0   adc r29, r26  mul r2, r7    mul r5, r6    adc r14, r28  std Z+7, r13
mul r2, r9    mul r4, r7    sbc r5, r21   adc r16, r1   mul r21, r22  add r21, r0   add r23, r0   adc r15, r29  std Z+8, r14
movw r18, r0  add r13, r0   sbc r0, r0    adc r29, r26  add r17, r0   adc r22, r1   adc r24, r1   adc r16, r18  std Z+9, r15
mul r3, r6    adc r18, r1   sub r6, r22   mul r19, r22  adc r28, r1   mul r3, r6    adc r25, r26  adc r17, r19  std Z+10, r16
add r11, r0   adc r19, r21  sbc r7, r23   add r15, r0   adc r29, r26  add r21, r0   mul r3, r9    bld r27, 0    std Z+11, r17
adc r12, r1   mul r5, r6    sbc r8, r24   adc r16, r1   mul r19, r25  adc r22, r1   movw r2, r26  dec 27        std Z+12, r28
adc r13, r18  add r13, r0   sbc r9, r25   adc r17, r29  movw r18, r26 adc r23, r26  add r24, r0   adc r26, r27  std Z+13, r29
adc r19, r21  adc r18, r1   sbc r1, r1    adc r28, r26  add r28, r0   movw r24, r26 adc r25, r1   mov r0, r26  std Z+14, r18
mul r3, r7    adc r19, r21  eor r2, r0    mul r18, r24  adc r29, r1   mul r2, r8    adc r2, r27   asr r0        std Z+15, r19
add r12, r0   mul r5, r8    eor r3, r0    add r16, r0   adc r18, r26  add r22, r0   mul r4, r8    eor r20, r27
adc r13, r1   add r14, r18  eor r4, r0    adc r17, r1   mul r20, r24  adc r23, r1   add r24, r0   eor r21, r27
adc r19, r21  adc r0, r19   eor r5, r0    adc r28, r26  add r28, r0   adc r24, r26  adc r25, r1   eor r22, r27
```

Divide and Conquer

"Compute $M = |A_l - A_h| \cdot |B_l - B_h|$"

```
clr r20       mul r4, r9    adc r1, r21   eor r6, r1   mul r19, r23 adc r29, r1  mul r3, r7   adc r2, r27   eor r23, r27
clr r21       add r14, r19  add r15, r0   eor r7, r1   add r16, r0  adc r18, r26 add r22, r0  mul r5, r7   eor r24, r27
movw r16, r20 adc r15, r0   adc r16, r1   eor r8, r1   adc r17, r1  mul r21, r23 adc r23, r1  add r24, r0  eor r25, r27
ld r2, X+     adc r16, r1   adc r17, r21  eor r9, r1   adc r28, r26 add r28, r0  adc r24, r26 adc r25, r1  eor r2 , r27
ld r3, X+     mul r4, r8    ldd r22 Y+4   sub r2, r0   mul r20, r22 adc r29, r1  mul r4, r6   adc r2, r27   eor r3 , r27
ld r4, X+     movw r18, r0  ldd r23 Y+5   sbc r3, r0   add r16, r0  adc r18, r26 add r22, r0  mul r4, r9   adc r10, r20
ld r5, X+     mul r4, r6    ldd r24 Y+6   sbc r4, r0   adc r17, r1  mul r20, r25 adc r23, r1  add r25, r0  adc r11, r21
ldd r6 Y+0    add r12, r0   ldd r25 Y+7   sbc r5, r0   adc r28, r26 add r29, r0  adc r24, r26 adc r2, r1   adc r12, r22
ldd r7 Y+1    adc r13, r1   movw r28, r20 sub r6, r1   clr r29       adc r18, r1  mul r2, r9   adc r3, r27   adc r13, r23
ldd r8 Y+2    adc r14, r18  ld r18, X+    sbc r7, r1   mul r18, r25 adc r19, r26 add r23, r0  mul r5, r8   adc r14, r24
ldd r9 Y+3    adc r19, r21  ld r19, X+    sbc r8, r1   add r17, r0  mul r21, r24 adc r24, r1  add r25, r0  adc r15, r25
mul r2, r8    mul r3, r8    ld r20, X+    sbc r9, r1   adc r28, r1  add r29, r0  adc r25, r26 adc r2, r1   adc r16, r2
movw r12, r0  add r13, r0   ld r21, X+    eor r0, r1   adc r29, r26 adc r18, r1  mul r3, r8   adc r3, r27   adc r17, r3
mul r2, r6    adc r14, r1   movw r26, r28 bst r0, 0    mul r19, r24 adc r19, r26 add r23, r0  mul r5, r9   adc r28, r26
movw r10, r0  adc r19, r21  std Z+0, r10  mul r18, r22 add r17, r0  mul r21, r25 adc r24, r1  add r2, r0   adc r29, r0
mul r2, r7    mul r5, r9    std Z+1, r11  add r14, r0  adc r28, r1  add r18, r0  adc r25, r26 adc r3, r1   adc r18, r0
add r11, r0   add r15, r19  std Z+2, r12  adc r15, r1  adc r29, r26 adc r19, r1  mul r4, r7   add r10, r14 adc r19, r0
adc r12, r1   adc r16, r0   std Z+3, r13  adc r16, r26 mul r20, r23 mul r2, r6   add r23, r0  adc r11, r15 std Z+4, r10
adc r13, r21  adc r17, r1   sub r2, r18   adc r29, r26 add r17, r0  movw r20, r0 adc r24, r1  adc r12, r16 std Z+5, r11
mul r3, r9    mul r5, r7    sbc r3, r19   mul r18, r23 adc r28, r1  movw r22, r26 adc r25, r26 adc r13, r17 std Z+6, r12
movw r14, r0  movw r18, r0  sbc r4, r20   add r15, r0  adc r29, r26 mul r2, r7   mul r5, r6   adc r14, r28 std Z+7, r13
mul r2, r9    mul r4, r7    sbc r5, r21   adc r16, r1  mul r21, r22 add r21, r0  add r23, r0  adc r15, r29 std Z+8, r14
movw r18, r0  add r13, r0   sbc r0, r0    adc r29, r26 add r17, r0  adc r22, r1  adc r24, r1  adc r16, r18 std Z+9, r15
mul r3, r6    adc r18, r1   sub r6, r22   mul r19, r22 adc r28, r1  mul r3, r6   adc r25, r26 adc r17, r19 std Z+10, r16
add r11, r0   adc r19, r21  sbc r7, r23   add r15, r0  adc r29, r26 add r21, r0  mul r3, r9   bld r27, 0   std Z+11, r17
adc r12, r1   mul r5, r6    sbc r8, r24   adc r16, r1  mul r19, r25 adc r22, r1  movw r2, r26 dec 27       std Z+12, r28
adc r13, r18  add r13, r0   sbc r9, r25   adc r17, r29 movw r18, r26 adc r23, r26 add r24, r0  adc r26, r27 std Z+13, r29
adc r19, r21  adc r18, r1   sbc r1, r1    adc r28, r26 add r28, r0  movw r24, r26 adc r25, r1  mov r0, r26  std Z+14, r18
mul r3, r7    adc r19, r21  eor r2, r0    mul r18, r24 adc r29, r1  mul r2, r8   adc r2, r27   asr r0        std Z+15, r19
add r12, r0   mul r5, r8    eor r3, r0    add r16, r0  adc r18, r26 add r22, r0  mul r4, r8   eor r20, r27
adc r13, r1   add r14, r18  eor r4, r0    adc r17, r1  mul r20, r24 adc r23, r1  add r24, r0  eor r21, r27
adc r19, r21  adc r0, r19   eor r5, r0    adc r28, r26 add r28, r0  adc r24, r26 adc r25, r1  eor r22, r27
```

Divide and Conquer

"Compute $(1 + 2^n)(L + 2^n H)$"

```
clr r20       mul r4, r9    adc r1, r21   eor r6, r1    mul r19, r23  adc r29, r1   mul r3, r7    adc r2, r27   eor r23, r27
clr r21       add r14, r19  add r15, r0   eor r7, r1    add r16, r0   adc r18, r26  add r22, r0   mul r5, r7    eor r24, r27
movw r16, r20 adc r15, r0   adc r16, r1   eor r8, r1    adc r17, r1   mul r21, r23  adc r23, r1   add r24, r0   eor r25, r27
ld r2, X+     adc r16, r1   adc r17, r21  eor r9, r1    adc r28, r26  add r28, r0   adc r24, r26  adc r25, r1   eor r2 , r27
ld r3, X+     mul r4, r8    ldd r22 Y+4   sub r2, r0    mul r20, r22  adc r29, r1   mul r4, r6    adc r2, r27   eor r3 , r27
ld r4, X+     movw r18, r0  ldd r23 Y+5   sbc r3, r0    add r16, r0   adc r18, r26  add r22, r0   mul r4, r9    adc r10, r20
ld r5, X+     mul r4, r6    ldd r24 Y+6   sbc r4, r0    adc r17, r1   mul r20, r25  adc r23, r1   add r25, r0   adc r11, r21
ldd r6 Y+0    add r12, r0   ldd r25 Y+7   sbc r5, r0    adc r28, r26  add r29, r0   adc r24, r26  adc r2, r1   adc r12, r22
ldd r7 Y+1    adc r13, r1   movw r28, r20 sub r6, r1    clr r29       adc r18, r1   mul r2, r9    adc r3, r27   adc r13, r23
ldd r8 Y+2    adc r14, r18  ld r18, X+    sbc r7, r1    mul r18, r25  adc r19, r26  add r23, r0   mul r5, r8    adc r14, r24
ldd r9 Y+3    adc r19, r21  ld r19, X+    sbc r8, r1    add r17, r0   mul r21, r24  adc r24, r1   add r25, r0   adc r15, r25
mul r2, r8    mul r3, r8    ld r20, X+    sbc r9, r1    adc r28, r1   add r29, r0   adc r25, r26  adc r2, r1   adc r16, r2
movw r12, r0  add r13, r0   ld r21, X+    eor r0, r1    adc r29, r26  adc r18, r1   mul r3, r8    adc r3, r27   adc r17, r3
mul r2, r6    adc r14, r1   movw r26, r28 bst r0, 0     mul r19, r24  adc r19, r26  add r23, r0   mul r5, r9    adc r28, r26
movw r10, r0  adc r19, r21  std Z+0, r10  mul r18, r22  add r17, r0   mul r21, r25  adc r24, r1   add r2, r0   adc r29, r0
mul r2, r7    mul r5, r9    std Z+1, r11  add r14, r0   adc r28, r1   add r18, r0   adc r25, r26  adc r3, r1   adc r18, r0
add r11, r0   add r15, r19  std Z+2, r12  adc r15, r1   adc r29, r26  adc r19, r1   mul r4, r7    add r10, r14  adc r19, r0
adc r12, r1   adc r16, r0   std Z+3, r13  adc r16, r26  mul r20, r23  mul r2, r6    add r23, r0   adc r11, r15  std Z+4, r10
adc r13, r21  adc r17, r1   sub r2, r18   adc r29, r26  add r17, r0   movw r20, r0  adc r24, r1   adc r12, r16  std Z+5, r11
mul r3, r9    mul r5, r7    sbc r3, r19   mul r18, r23  adc r28, r1   movw r22, r26 adc r25, r26  adc r13, r17  std Z+6, r12
movw r14, r0  movw r18, r0  sbc r4, r20   add r15, r0   adc r29, r26  mul r2, r7    mul r5, r6    adc r14, r28  std Z+7, r13
mul r2, r9    mul r4, r7    sbc r5, r21   adc r16, r1   mul r21, r22  add r21, r0   add r23, r0   adc r15, r29  std Z+8, r14
movw r18, r0  add r13, r0   sbc r0, r0    adc r29, r26  add r17, r0   adc r22, r1   adc r24, r1   adc r16, r18  std Z+9, r15
mul r3, r6    adc r18, r1   sub r6, r22   mul r19, r22  adc r28, r1   mul r3, r6    adc r25, r26  adc r17, r19  std Z+10, r16
add r11, r0   adc r19, r21  sbc r7, r23   add r15, r0   adc r29, r26  add r21, r0   mul r3, r9    bld r27, 0    std Z+11, r17
adc r12, r1   mul r5, r6    sbc r8, r24   adc r16, r1   mul r19, r25  adc r22, r1   movw r2, r26  dec 27        std Z+12, r28
adc r13, r18  add r13, r0   sbc r9, r25   adc r17, r29  movw r18, r26 adc r23, r26  add r24, r0   adc r26, r27  std Z+13, r29
adc r19, r21  adc r18, r1   sbc r1, r1    adc r28, r26  add r28, r0   movw r24, r26 adc r25, r1   mov r0, r26  std Z+14, r18
mul r3, r7    adc r19, r21  eor r2, r0    mul r18, r24  adc r29, r1   mul r2, r8    adc r2, r27   asr r0        std Z+15, r19
add r12, r0   mul r5, r8    eor r3, r0    add r16, r0   adc r18, r26  add r22, r0   mul r4, r8    eor r20, r27
adc r13, r1   add r14, r18  eor r4, r0    adc r17, r1   mul r20, r24  adc r23, r1   add r24, r0   eor r21, r27
adc r19, r21  adc r0, r19   eor r5, r0    adc r28, r26  add r28, r0   adc r24, r26  adc r25, r1   eor r22, r27
```

Divide and Conquer

"Correct the $\pm$ sign of $2^n M$"

```
clr r20       mul r4, r9    adc r1, r21   eor r6, r1    mul r19, r23  adc r29, r1   mul r3, r7    adc r2, r27   eor r23, r27
clr r21       add r14, r19  add r15, r0   eor r7, r1    add r16, r0   adc r18, r26  add r22, r0   mul r5, r7    eor r24, r27
movw r16, r20 adc r15, r0   adc r16, r1   eor r8, r1    adc r17, r1   mul r21, r23  adc r23, r1   add r24, r0   eor r25, r27
ld r2, X+     adc r16, r1   adc r17, r21  eor r9, r1    adc r28, r26  add r28, r0   adc r24, r26  adc r25, r1   eor r2 , r27
ld r3, X+     mul r4, r8    ldd r22 Y+4   sub r2, r0    mul r20, r22  adc r29, r1   mul r4, r6    adc r2, r27   eor r3 , r27
ld r4, X+     movw r18, r0  ldd r23 Y+5   sbc r3, r0    add r16, r0   adc r18, r26  add r22, r0   mul r4, r9    adc r10, r20
ld r5, X+     mul r4, r6    ldd r24 Y+6   sbc r4, r0    adc r17, r1   mul r20, r25  adc r23, r1   add r25, r0   adc r11, r21
ldd r6 Y+0    add r12, r0   ldd r25 Y+7   sbc r5, r0    adc r28, r26  add r29, r0   adc r24, r26  adc r2, r1   adc r12, r22
ldd r7 Y+1    adc r13, r1   movw r28, r20 sub r6, r1    clr r29       adc r18, r1   mul r2, r9    adc r3, r27   adc r13, r23
ldd r8 Y+2    adc r14, r18  ld r18, X+    sbc r7, r1    mul r18, r25  adc r19, r26  add r23, r0   mul r5, r8    adc r14, r24
ldd r9 Y+3    adc r19, r21  ld r19, X+    sbc r8, r1    add r17, r0   mul r21, r24  adc r24, r1   add r25, r0   adc r15, r25
mul r2, r8    mul r3, r8    ld r20, X+    sbc r9, r1    adc r28, r1   add r29, r0   adc r25, r26  adc r2, r1   adc r16, r2
movw r12, r0  add r13, r0   ld r21, X+    eor r0, r1    adc r29, r26  adc r18, r1   mul r3, r8    adc r3, r27   adc r17, r3
mul r2, r6    adc r14, r1   movw r26, r28 bst r0, 0     mul r19, r24  adc r19, r26  add r23, r0   mul r5, r9    adc r28, r26
movw r10, r0  adc r19, r21  std Z+0, r10  mul r18, r22  add r17, r0   mul r21, r25  adc r24, r1   add r2, r0   adc r29, r0
mul r2, r7    mul r5, r9    std Z+1, r11  add r14, r0   adc r28, r1   add r18, r0   adc r25, r26  adc r3, r1   adc r18, r0
add r11, r0   add r15, r19  std Z+2, r12  adc r15, r1   adc r29, r26  adc r19, r1   mul r4, r7    add r10, r14  adc r19, r0
adc r12, r1   adc r16, r0   std Z+3, r13  adc r16, r26  mul r20, r23  mul r2, r6    add r23, r0   adc r11, r15  std Z+4, r10
adc r13, r21  adc r17, r1   sub r2, r18   adc r29, r26  add r17, r0   movw r20, r0  adc r24, r1   adc r12, r16  std Z+5, r11
mul r3, r9    mul r5, r7    sbc r3, r19   mul r18, r23  adc r28, r1   movw r22, r26 adc r25, r26  adc r13, r17  std Z+6, r12
movw r14, r0  movw r18, r0  sbc r4, r20   add r15, r0   adc r29, r26  mul r2, r7    mul r5, r6    adc r14, r28  std Z+7, r13
mul r2, r9    mul r4, r7    sbc r5, r21   adc r16, r1   mul r21, r22  add r21, r0   add r23, r0   adc r15, r29  std Z+8, r14
movw r18, r0  add r13, r0   sbc r0, r0    adc r29, r26  add r17, r0   adc r22, r1   adc r24, r1   adc r16, r18  std Z+9, r15
mul r3, r6    adc r18, r1   sub r6, r22   mul r19, r22  adc r28, r1   mul r3, r6    adc r25, r26  adc r17, r19  std Z+10, r16
add r11, r0   adc r19, r21  sbc r7, r23   add r15, r0   adc r29, r26  add r21, r0   mul r3, r9    bld r27, 0    std Z+11, r17
adc r12, r1   mul r5, r6    sbc r8, r24   adc r16, r1   mul r19, r25  adc r22, r1   movw r2, r26  dec 27        std Z+12, r28
adc r13, r18  add r13, r0   sbc r9, r25   adc r17, r29  movw r18, r26 adc r23, r26  add r24, r0   adc r26, r27  std Z+13, r29
adc r19, r21  adc r18, r1   sbc r1, r1    adc r28, r26  add r28, r0   movw r24, r26 adc r25, r1   mov r0, r26  std Z+14, r18
mul r3, r7    adc r19, r21  eor r2, r0    mul r18, r24  adc r29, r1   mul r2, r8    adc r2, r27   asr r0        std Z+15, r19
add r12, r0   mul r5, r8    eor r3, r0    add r16, r0   adc r18, r26  add r22, r0   mul r4, r8    eor r20, r27
adc r13, r1   add r14, r18  eor r4, r0    adc r17, r1   mul r20, r24  adc r23, r1   add r24, r0   eor r21, r27
adc r19, r21  adc r0, r19   eor r5, r0    adc r28, r26  add r28, r0   adc r24, r26  adc r25, r1   eor r22, r27
```

Divide and Conquer

"Compute $(1+2^n)(L+2^nH)+2^nM$"

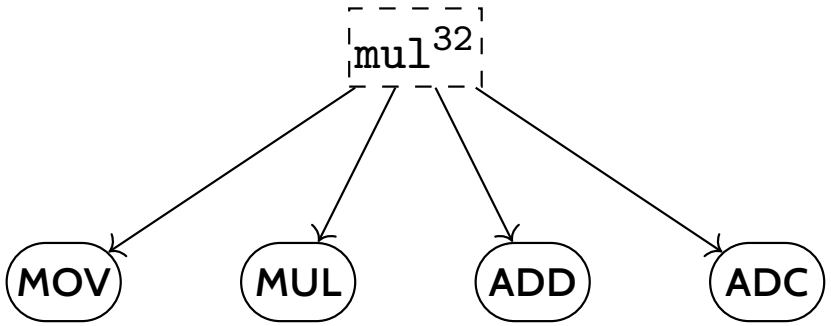
```
clr r20       mul r4, r9    adc r1, r21   eor r6, r1    mul r19, r23  adc r29, r1   mul r3, r7    adc r2, r27   eor r23, r27
clr r21       add r14, r19  add r15, r0   eor r7, r1    add r16, r0   adc r18, r26  add r22, r0   mul r5, r7    eor r24, r27
movw r16, r20 adc r15, r0   adc r16, r1   eor r8, r1    adc r17, r1   mul r21, r23  adc r23, r1   add r24, r0   eor r25, r27
ld r2, X+     adc r16, r1   adc r17, r21  eor r9, r1    adc r28, r26  add r28, r0   adc r24, r26  adc r25, r1   eor r2 , r27
ld r3, X+     mul r4, r8    ldd r22 Y+4   sub r2, r0    mul r20, r22  adc r29, r1   mul r4, r6    adc r2, r27   eor r3 , r27
ld r4, X+     movw r18, r0  ldd r23 Y+5   sbc r3, r0    add r16, r0   adc r18, r26  add r22, r0   mul r4, r9    adc r10, r20
ld r5, X+     mul r4, r6    ldd r24 Y+6   sbc r4, r0    adc r17, r1   mul r20, r25  adc r23, r1   add r25, r0   adc r11, r21
ldd r6 Y+0    add r12, r0   ldd r25 Y+7   sbc r5, r0    adc r28, r26  add r29, r0   adc r24, r26  adc r2, r1   adc r12, r22
ldd r7 Y+1    adc r13, r1   movw r28, r20 sub r6, r1    clr r29       adc r18, r1   mul r2, r9    adc r3, r27   adc r13, r23
ldd r8 Y+2    adc r14, r18  ld r18, X+    sbc r7, r1    mul r18, r25  adc r19, r26  add r23, r0   mul r5, r8    adc r14, r24
ldd r9 Y+3    adc r19, r21  ld r19, X+    sbc r8, r1    add r17, r0   mul r21, r24  adc r24, r1   add r25, r0   adc r15, r25
mul r2, r8    mul r3, r8    ld r20, X+    sbc r9, r1    adc r28, r1   add r29, r0   adc r25, r26  adc r2, r1   adc r16, r2
movw r12, r0  add r13, r0   ld r21, X+    eor r0, r1    adc r29, r26  adc r18, r1   mul r3, r8    adc r3, r27   adc r17, r3
mul r2, r6    adc r14, r1   movw r26, r28 bst r0, 0     mul r19, r24  adc r19, r26  add r23, r0   mul r5, r9    adc r28, r26
movw r10, r0  adc r19, r21  std Z+0, r10  mul r18, r22  add r17, r0   mul r21, r25  adc r24, r1   add r2, r0   adc r29, r0
mul r2, r7    mul r5, r9    std Z+1, r11  add r14, r0   adc r28, r1   add r18, r0   adc r25, r26  adc r3, r1   adc r18, r0
add r11, r0   add r15, r19  std Z+2, r12  adc r15, r1   adc r29, r26  adc r19, r1   mul r4, r7    add r10, r14  adc r19, r0
adc r12, r1   adc r16, r0   std Z+3, r13  adc r16, r26  mul r20, r23  mul r2, r6    add r23, r0   adc r11, r15  std Z+4, r10
adc r13, r21  adc r17, r1   sub r2, r18   adc r29, r26  add r17, r0   movw r20, r0  adc r24, r1   adc r12, r16  std Z+5, r11
mul r3, r9    mul r5, r7    sbc r3, r19   mul r18, r23  adc r28, r1   movw r22, r26 adc r25, r26  adc r13, r17  std Z+6, r12
movw r14, r0  movw r18, r0  sbc r4, r20   add r15, r0   adc r29, r26  mul r2, r7    mul r5, r6    adc r14, r28  std Z+7, r13
mul r2, r9    mul r4, r7    sbc r5, r21   adc r16, r1   mul r21, r22  add r21, r0   add r23, r0   adc r15, r29  std Z+8, r14
movw r18, r0  add r13, r0   sbc r0, r0    adc r29, r26  add r17, r0   adc r22, r1   adc r24, r1   adc r16, r18  std Z+9, r15
mul r3, r6    adc r18, r1   sub r6, r22   mul r19, r22  adc r28, r1   mul r3, r6    adc r25, r26  adc r17, r19  std Z+10, r16
add r11, r0   adc r19, r21  sbc r7, r23   add r15, r0   adc r29, r26  add r21, r0   mul r3, r9    bld r27, 0    std Z+11, r17
adc r12, r1   mul r5, r6    sbc r8, r24   adc r16, r1   mul r19, r25  adc r22, r1   movw r2, r26  dec 27        std Z+12, r28
adc r13, r18  add r13, r0   sbc r9, r25   adc r17, r29  movw r18, r26 adc r23, r26  add r24, r0   adc r26, r27  std Z+13, r29
adc r19, r21  adc r18, r1   sbc r1, r1    adc r28, r26  add r28, r0   movw r24, r26 adc r25, r1   mov r0, r26  std Z+14, r18
mul r3, r7    adc r19, r21  eor r2, r0    mul r18, r24  adc r29, r1   mul r2, r8    adc r2, r27   asr r0        std Z+15, r19
add r12, r0   mul r5, r8    eor r3, r0    add r16, r0   adc r18, r26  add r22, r0   mul r4, r8    eor r20, r27
adc r13, r1   add r14, r18  eor r4, r0    adc r17, r1   mul r20, r24  adc r23, r1   add r24, r0   eor r21, r27
adc r19, r21  adc r0, r19   eor r5, r0    adc r28, r26  add r28, r0   adc r24, r26  adc r25, r1   eor r22, r27
```

Divide and Conquer

"Result is $(2^n A_h + A_l)(2^n B_h + B_l)$"

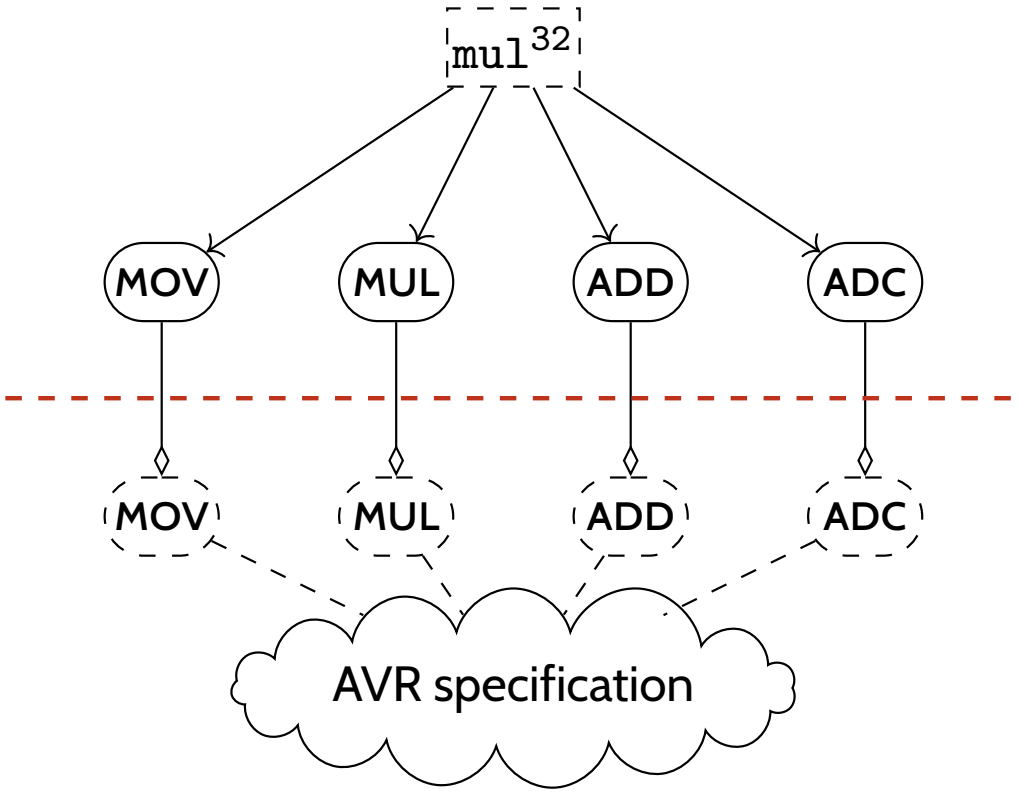
```
clr r20       mul r4, r9    adc r1, r21   eor r6, r1    mul r19, r23 adc r29, r1  mul r3, r7    adc r2, r27   eor r23, r27
clr r21       add r14, r19  add r15, r0   eor r7, r1    add r16, r0  adc r18, r26 add r22, r0   mul r5, r7    eor r24, r27
movw r16, r20 adc r15, r0   adc r16, r1   eor r8, r1    adc r17, r1  mul r21, r23 adc r23, r1   add r24, r0   eor r25, r27
ld r2, X+     adc r16, r1   adc r17, r21  eor r9, r1    adc r28, r26 add r28, r0  adc r24, r26  adc r25, r1   eor r2 , r27
ld r3, X+     mul r4, r8    ldd r22 Y+4   sub r2, r0    mul r20, r22 adc r29, r1  mul r4, r6    adc r2, r27   eor r3 , r27
ld r4, X+     movw r18, r0  ldd r23 Y+5   sbc r3, r0    add r16, r0  adc r18, r26 add r22, r0   mul r4, r9    adc r10, r20
ld r5, X+     mul r4, r6    ldd r24 Y+6   sbc r4, r0    adc r17, r1  mul r20, r25 adc r23, r1   add r25, r0   adc r11, r21
ldd r6 Y+0    add r12, r0   ldd r25 Y+7   sbc r5, r0    adc r28, r26 add r29, r0  adc r24, r26  adc r2, r1    adc r12, r22
ldd r7 Y+1    adc r13, r1   movw r28, r20 sub r6, r1    clr r29       adc r18, r1  mul r2, r9    adc r3, r27   adc r13, r23
ldd r8 Y+2    adc r14, r18  ld r18, X+    sbc r7, r1    mul r18, r25 adc r19, r26 add r23, r0   mul r5, r8    adc r14, r24
ldd r9 Y+3    adc r19, r21  ld r19, X+    sbc r8, r1    add r17, r0  mul r21, r24 adc r24, r1   add r25, r0   adc r15, r25
mul r2, r8    mul r3, r8    ld r20, X+    sbc r9, r1    adc r28, r1  add r29, r0  adc r25, r26  adc r2, r1    adc r16, r2
movw r12, r0  add r13, r0   ld r21, X+    eor r0, r1    adc r29, r26 adc r18, r1  mul r3, r8    adc r3, r27   adc r17, r3
mul r2, r6    adc r14, r1   movw r26, r28 bst r0, 0     mul r19, r24 adc r19, r26 add r23, r0   mul r5, r9    adc r28, r26
movw r10, r0  adc r19, r21  std Z+0, r10  mul r18, r22 add r17, r0  mul r21, r25 adc r24, r1   add r2, r0    adc r29, r0
mul r2, r7    mul r5, r9    std Z+1, r11  add r14, r0   adc r28, r1  add r18, r0  adc r25, r26  adc r3, r1    adc r18, r0
add r11, r0   add r15, r19  std Z+2, r12  adc r15, r1   adc r29, r26 adc r19, r1  mul r4, r7    add r10, r14  adc r19, r0
adc r12, r1   adc r16, r0   std Z+3, r13  adc r16, r26  mul r20, r23 mul r2, r6   add r23, r0   adc r11, r15  std Z+4, r10
adc r13, r21  adc r17, r1   sub r2, r18   adc r29, r26  add r17, r0  movw r20, r0 adc r24, r1   adc r12, r16  std Z+5, r11
mul r3, r9    mul r5, r7    sbc r3, r19   mul r18, r23  adc r28, r1  movw r22, r26 adc r25, r26  adc r13, r17  std Z+6, r12
movw r14, r0  movw r18, r0  sbc r4, r20   add r15, r0   adc r29, r26 mul r2, r7   mul r5, r6    adc r14, r28  std Z+7, r13
mul r2, r9    mul r4, r7    sbc r5, r21   adc r16, r1   mul r21, r22 add r21, r0  add r23, r0   adc r15, r29  std Z+8, r14
movw r18, r0  add r13, r0   sbc r0, r0    adc r29, r26  add r17, r0  adc r22, r1  adc r24, r1   adc r16, r18  std Z+9, r15
mul r3, r6    adc r18, r1   sub r6, r22   mul r19, r22  adc r28, r1  mul r3, r6   adc r25, r26  adc r17, r19  std Z+10, r16
add r11, r0   adc r19, r21  sbc r7, r23   add r15, r0   adc r29, r26 add r21, r0  mul r3, r9    bld r27, 0    std Z+11, r17
adc r12, r1   mul r5, r6    sbc r8, r24   adc r16, r1   mul r19, r25 adc r22, r1  movw r2, r26  dec 27        std Z+12, r28
adc r13, r18  add r13, r0   sbc r9, r25   adc r17, r29  movw r18, r26 adc r23, r26 add r24, r0   adc r26, r27  std Z+13, r29
adc r19, r21  adc r18, r1   sbc r1, r1    adc r28, r26  add r28, r0  movw r24, r26 adc r25, r1   mov r0, r26  std Z+14, r18
mul r3, r7    adc r19, r21  eor r2, r0    mul r18, r24  adc r29, r1  mul r2, r8    adc r2, r27   asr r0        std Z+15, r19
add r12, r0   mul r5, r8    eor r3, r0    add r16, r0   adc r18, r26 add r22, r0  mul r4, r8    eor r20, r27
adc r13, r1   add r14, r18  eor r4, r0    adc r17, r1   mul r20, r24 adc r23, r1  add r24, r0   eor r21, r27
adc r19, r21  adc r0, r19   eor r5, r0    adc r28, r26  add r28, r0  adc r24, r26 adc r25, r1   eor r22, r27
```

Modeling Assembly Code



Useful formalization

Modeling Assembly Code



Useful formalization

Validated formalization

Example: ADD instruction

AVR Specification: ADD Rd, Rr

$$\text{Rd} \leftarrow \text{Rd} + \text{Rr} \qquad \text{(implied: 8-bit operation)}$$

$$C_{flag} \leftarrow \text{Rd}_7 \cdot \text{Rr}_7 + \text{Rr}_7 \cdot \overline{\text{R}_7} + \overline{\text{R}_7} \cdot \text{Rd}_7 \qquad \text{(boolean operations)}$$

```
let add (rd rr: register): unit
 writes { cf, reg }
 ensures { reg = old reg[rd <- mod (old (reg[rd] + reg[rr])) 256] }
 ensures { ?cf = div (old (reg[rd] + reg[rr])) 256 }
= let rdv  = read_byte reg rd in
  let rrv  = read_byte reg rr in
  let res  = clip (rdv + rrv) in
  set_byte reg rd res;
  cf.value <- (ar_nth rdv 7 && ar_nth rrv 7 ||
               ar_nth rrv 7 && not ar_nth res 7 ||
               not ar_nth res 7 && ar_nth rdv 7)
```

Example: ADD instruction

AVR Specification: ADD Rd, Rr

$$Rd \leftarrow Rd + Rr \qquad \text{(implied: 8-bit operation)}$$

$$C_{flag} \leftarrow Rd_7 \cdot Rr_7 + Rr_7 \cdot \overline{R_7} + \overline{R_7} \cdot Rd_7 \qquad \text{(boolean operations)}$$

```
let add (rd rr: register): unit
 writes { cf, reg }
 ensures { reg = old reg[rd <- mod (old (reg[rd] + reg[rr])) 256] }
 ensures { ?cf = div (old (reg[rd] + reg[rr])) 256 }
= let rdv  = read_byte reg rd in
  let rrv  = read_byte reg rr in
  let res  = clip (rdv + rrv) in
  set_byte reg rd res;
  cf.value <- (ar_nth rdv 7 && ar_nth rrv 7 ||
               ar_nth rrv 7 && not ar_nth res 7 ||
               not ar_nth res 7 && ar_nth rdv 7)
```

Example: ADD instruction

AVR Specification: ADD Rd, Rr

$$Rd \leftarrow Rd + Rr \qquad \text{(implied: 8-bit operation)}$$

$$C_{flag} \leftarrow Rd_7 \cdot Rr_7 + Rr_7 \cdot \overline{R_7} + \overline{R_7} \cdot Rd_7 \qquad \text{(boolean operations)}$$

```
let add (rd rr: register): unit
 writes { cf, reg }
 ensures { reg = old reg[rd <- mod (old (reg[rd] + reg[rr])) 256] }
 ensures { ?cf = div (old (reg[rd] + reg[rr])) 256 }
= let rdv  = read_byte reg rd in
  let rrv  = read_byte reg rr in
  let res  = clip (rdv + rrv) in
  set_byte reg rd res;
  cf.value <- (ar_nth rdv 7 && ar_nth rrv 7 ||
               ar_nth rrv 7 && not ar_nth res 7 ||
               not ar_nth res 7 && ar_nth rdv 7)
```

Example: ADD instruction

AVR Specification: ADD Rd, Rr

$$Rd \leftarrow Rd + Rr \qquad \text{(implied: 8-bit operation)}$$

$$C_{flag} \leftarrow Rd_7 \cdot Rr_7 + Rr_7 \cdot \overline{R_7} + \overline{R_7} \cdot Rd_7 \qquad \text{(boolean operations)}$$

```
let add (rd rr: register): unit
 writes { cf, reg }
 ensures { reg = old reg[rd <- mod (old (reg[rd] + reg[rr])) 256] }
 ensures { ?cf = div (old (reg[rd] + reg[rr])) 256 }
= let rdv  = read_byte reg rd in
  let rrv  = read_byte reg rr in
  let res  = clip (rdv + rrv) in
  set_byte reg rd res;
  cf.value <- (ar_nth rdv 7 && ar_nth rrv 7 ||
               ar_nth rrv 7 && not ar_nth res 7 ||
               not ar_nth res 7 && ar_nth rdv 7)
```

A Specification of X25519

Total correctness

$$\{P\}\ \texttt{crypto\_scalar\_mult}\ \{Q\}$$

What is a satisfying Q?

A Specification of X25519

Total correctness

$$\{P\}\ \texttt{crypto\_scalar\_mult}\ \{Q\}$$

What is a satisfying Q?

- A result computed by a reference implementation?

A Specification of X25519

Total correctness

$$\{P\}\ \texttt{crypto\_scalar\_mult}\ \{Q\}$$

What is a satisfying Q?

- A result computed by a reference implementation?
- A complete mathematical description?

A Specification of X25519

Total correctness

$$\{P\}\ \texttt{crypto\_scalar\_mult}\ \{Q\}$$

What is a satisfying Q?

- A result computed by a reference implementation?
- A complete mathematical description?
- *Something that admits validation!*

A Specification of X25519

Total correctness

$$\{P\}\ \texttt{crypto\_scalar\_mult}\ \{Q\}$$

What is a satisfying Q?

- A result computed by a reference implementation?
- A complete mathematical description?
- *Something that admits validation!*

State result using recognized concepts

- Multiplication by *doubling-and-adding* [Montgomery, 1987]
- Formulas proven to work for Curve25519 [Bernstein, 2006]

The Montgomery ladder

Scalar multiplication in Curve25519

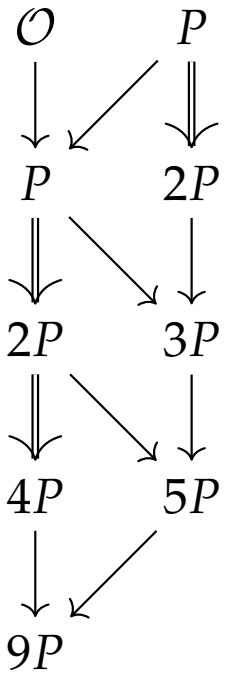
Formal specification

$$n \cdot P \stackrel{\text{def}}{=} (\textsf{LADDER}\ n\ P)_1$$

Where:

$$\begin{aligned}
\textsf{LADDER}\ 0\ P &\stackrel{\text{def}}{=} (\mathcal{O},\ P) \\
\textsf{LADDER}\ (2n)\ P &\stackrel{\text{def}}{=} (2R_n,\ R_n + R_n^\star) \\
\textsf{LADDER}\ (2n+1)\ P &\stackrel{\text{def}}{=} (R_n + R_n^\star,\ 2R_n^\star) \\
\text{with } (R_n,\ R_n^\star) &= \textsf{LADDER}\ n\ P
\end{aligned}$$

$2R$ and $R + R^\star$ computed using Montgomery's formulas



(image idea: Bernstein, Lange [2017])

QED

What was proven?

`crypto_scalar_mult` computes $X(n \cdot P)$ from n and $X(P)$

Fine print

If $n \in \{2^{254} + 8k : 0 \leq k < 2^{251}\}$ and $X(P) < 2^{255}$ per RFC7748.
Otherwise the implementation will adjust $n, X(P)$ accordingly.
Furthermore by convention $X(\mathcal{O}) = 0$.
The primality of $2^{255} - 19$ is assumed.

QED

What was proven?

`crypto_scalar_mult` computes $X(n \cdot P)$ from n and $X(P)$

Fine print

If $n \in \{2^{254} + 8k : 0 \leq k < 2^{251}\}$ and $X(P) < 2^{255}$ per RFC7748.
Otherwise the implementation will adjust $n, X(P)$ accordingly.
Furthermore by convention $X(\mathcal{O}) = 0$.
The primality of $2^{255} - 19$ is assumed.

After we fixed two assembly routines.

Surprise!

`crypto_scalar_mult` **was well-tested...**
...but not with interrupts enabled.

Surprise!

`crypto_scalar_mult` was well-tested...
...but not with interrupts enabled.

```
c8a71edd37b496ed9f1c763b86f1614b24215280e6d4c48b6cdf477f9f92af6c
ERROR: crypto_scalarmult does not handle p overlap
c8a71edd37b496ed9f1c763b86f1614b24215280e6d4c48b6cdf477f9f92af6c
c8a71edd37b496ed9f1c763b86f1614b24215280e6d4c48b6cdf477f9f92af6c
ERROR: crypto_scalarmult does not handle n overlap
c8a71edd37b496ed9f1c763b86f1614b24215280e6d4c48b6cdf477f9f92af6c
ERROR: crypto_scalarmult does not handle n overlap
ERROR: crypto_scalarmult not associative
ERROR: crypto_scalarmult not associative
```

Cause: unsafe manual allocation of temporaries in $\texttt{mul}^{256}$ & $\texttt{sqr}^{256}$

Surprise!

`crypto_scalar_mult` was well-tested...
...but not with interrupts enabled.

```
c8a71edd37b496ed9f1c763b86f1614b24215280e6d4c48b6cdf477f9f92af6c
ERROR: crypto_scalarmult does not handle p overlap
c8a71edd37b496ed9f1c763b86f1614b24215280e6d4c48b6cdf477f9f92af6c
c8a71edd37b496ed9f1c763b86f1614b24215280e6d4c48b6cdf477f9f92af6c
ERROR: crypto_scalarmult does not handle n overlap
c8a71edd37b496ed9f1c763b86f1614b24215280e6d4c48b6cdf477f9f92af6c
ERROR: crypto_scalarmult does not handle n overlap
ERROR: crypto_scalarmult not associative
ERROR: crypto_scalarmult not associative
```

Cause: unsafe manual allocation of temporaries in $\texttt{mul}^{256}$ & $\texttt{sqr}^{256}$

Discovered by the restrictions imposed by our model.

Conclusion

Effort

- *Unreliable estimate*: ∼10 weeks of work
- Probably not bad compared to *implementation effort*!

Conclusion

Effort

- *Unreliable estimate*: ∼10 weeks of work
- Probably not bad compared to *implementation effort*!

Good practices

- Reason compositionally
- Only specify what you need
- Build consistent models out of consistent models

Conclusion

Effort

- *Unreliable estimate*: ∼10 weeks of work
- Probably not bad compared to *implementation effort*!

Good practices

- Reason compositionally
- Only specify what you need
- Build consistent models out of consistent models

Future work

- C→assembly: *Verified code from untrusted compiler and models!*
- Assembly verification plugin for Why3
- Other architectures than AVR

Conclusion

Effort

- *Unreliable estimate*: ∼10 weeks of work
- Probably not bad compared to *implementation effort*!

Good practices

- Reason compositionally
- Only specify what you need
- Build consistent models out of consistent models

Future work

- C→assembly: *Verified code from untrusted compiler and models!*
- Assembly verification plugin for Why3
- Other architectures than AVR

`https://github.com/squell/verified-x25519-for-avr`

Intentionally blank

Modeling the AVR architecture

Underspecification of instructions

- 32 registers, data segment & stack
- *Carry flag* and *Transfer flag*
- Memory size depends on application.

Model consistency

- *Invariant*: all memory cells hold 8-bit values
- Constructed in WhyML: no axiomatic definitions

Assembly verification, quantified

function	*instructions*	*user annotations*	*generated goals*	*CPU time*
`bigint_mul256:mul128`	1078	122	300	1504.6s
`bigint_mul256`	693	85	506	2000.1s
`bigint_square256:sqr128`	672	26	135	363.8s
`bigint_square256`	493	38	359	1796.6s
`bigint_subp`	103	12	84	184.0s
`fe25519_red`	305	41	182	155.3s
`fe25519_add`	242	52	209	156.4s
`fe25519_sub`	242	53	212	119.6s
`fe25519_mul121666`	138	56	149	393.0s

C verification, quantified

function	*lines*	*user annotations*	*generated goals*	*CPU time*
`fe25519_setzero`	3	2	7	0.4s
`fe25519_setone`	4	2	7	0.4s
`fe25519_neg`	3	0	3	0.2s
`fe25519_cmov`	5	3	10	36.5s
`fe25519_freeze`	4	2	9	4.7s
`fe25519_unpack`	4	8	30	41.0s
`fe25519_pack`	5	2	11	1.6s
`fe25519_mul`	3	0	1	0.2s
`fe25519_square`	3	0	1	0.1s
`fe25519_invert`	51	49	306	557.3s
`work_cswap`	8	0	13	3.8s
`ladderstep`	26	22	80	202.8s
`mladder`	26	22	140	345.1s
`crypto_scalar_mult_curve25519`	13	27	57	74.2s

Montgomery ladder

$(X_1 : Z_1) \leftarrow (1:0)$; $(X_2 : Z_2) \leftarrow (x_P : 1)$; $prev \leftarrow 0$; $j \leftarrow 6$
for $i \leftarrow 31$ **downto** 0 **do**
    while $j \geq 0$ **do**
        $bit \leftarrow$ **bit** $8i + j$ **of** N
        $swap \leftarrow bit \oplus prev$; $prev \leftarrow bit$
        if $swap$ **then** $(X_1 : Z_1, X_2 : Z_2) \leftarrow (X_2 : Z_2, X_1 : Z_1)$ ▷ by conditional moves
        LADDERSTEP$(x_P, X_1 : Z_1, X_2 : Z_2)$
        $j \leftarrow j - 1$
    end while
    $j \leftarrow 7$
end for
return $(X_1 : Z_1)$

procedure LADDERSTEP
    $T_1 \leftarrow X_2 + Z_2$
    $X_2 \leftarrow X_2 - Z_2$
    $Z_2 \leftarrow X_1 + Z_1$
    $X_1 \leftarrow X_1 - Z_1$
    $T_1 \leftarrow T_1 \cdot X_1$
    $X_2 \leftarrow X_2 \cdot Z_2$
    $Z_2 \leftarrow (Z_2)^2$
    $X_1 \leftarrow (X_1)^2$
    $T_2 \leftarrow Z_2 - X_1$
    $Z_1 \leftarrow T_2 \cdot 121666$
    $Z_1 \leftarrow Z_1 + X_1$
    $Z_1 \leftarrow T_2 \cdot Z_1$
    $X_1 \leftarrow Z_2 \cdot X_1$
    $Z_2 \leftarrow T_1 - X_2$
    $Z_2 \leftarrow (Z_2)^2$
    $Z_2 \leftarrow Z_2 \cdot x_P$
    $X_2 \leftarrow T_1 + X_2$
    $X_2 \leftarrow (X_2)^2$
end procedure

```
abstract
ensures { synchronized shadow reg }
ensures { uint 4 reg 2 = old (abs (uint 4 reg 2 - uint 4 reg 18)) }
ensures { uint 4 reg 6 = old (abs (uint 4 reg 6 - uint 4 reg 22)) }
ensures { ?tf = 0 <-> old ((uint 4 reg 2 < uint 4 reg 18) <-> (uint 4 reg 6 < uint 4 reg 22)) }
  sub r2 r18;
  sbc r3 r19;
  sbc r4 r20;
  sbc r5 r21;
  sbc r0 r0;
  sub r6 r22;
  sbc r7 r23;
  sbc r8 r24;
  sbc r9 r25;
  sbc r1 r1;
  eor r2 r0;
  eor r3 r0;
  eor r4 r0;
  eor r5 r0;
  eor r6 r1;
  eor r7 r1;
  eor r8 r1;
  eor r9 r1;
  sub r2 r0;
  sbc r3 r0;
  sbc r4 r0;
  sbc r5 r0;
  sub r6 r1;
  sbc r7 r1;
  sbc r8 r1;
  sbc r9 r1;
  eor r0 r1;
  bst r0 0;
modify_r0(); modify_r1(); modify_r2(); modify_r3(); modify_r4();
modify_r5(); modify_r6(); modify_r7(); modify_r8(); modify_r9();
end;
```

```
type ratio = { x: int; z: int }
constant infty: ratio = {x=1;z=0}

constant p25519: int = pow2 255 - 19
predicate (===) (x y: int) = mod x p25519 = mod y p25519
predicate (~) (p q: ratio) = x p*z q === x q*z p
predicate (==~) (x:int) (xz: ratio) = xz ~ {x=x; z=1}

function add (m n mn: ratio): ratio
  = { x = 4*z mn*sqr(x m*x n - z m*z n);
      z = 4*x mn*sqr(x m*z n - z m*x n) }
function double (n: ratio): ratio
  = { x = sqr(sqr(x n) - sqr(z n));
      z = 4*x n*z n * (sqr(x n) + 486662*x n*z n + sqr(z n)) }

function ladder (n: int) (p: ratio): (ratio, ratio)
axiom ladder_0: (*these axiomatic definitions are proven consistent*)
  forall p.ladder 0 p = ({x=1; z=0}, p)
axiom ladder_even:
  forall p, n. n > 0 -> let (r0,r1) = ladder n p in
    ladder (2*n) p = (double r0, add r1 r0 p)
axiom ladder_odd:
  forall p, n. n >= 0 -> let (r0,r1) = ladder n p in
    ladder (2*n+1) p = (add r1 r0 p, double r1)

function scale (n: int) (m: int): ratio
  = let (r,_) = ladder n {x=m; z=1} in r
function clamp (x: int): int
  = mod x (pow2 254) + pow2 254 - mod x 8

val crypto_scalarmult_curve25519 (r s p: address_space)
  ensures { uint 32 r = mod (uint 32 r) p25519 }
  ensures { let xp   = mod (uint 32 p) (pow2 255) in
            let mult = scale (clamp (uint 32 s)) xp in
            if mult ~ infty then uint 32 r === 0
                            else uint 32 r ==~ mult }
```