Verifying Branch-Free Assembly Code in Why3

Marc Schoolderman

NWO Applied and Engineering Sciences

iCIS | Digital Security
Radboud University

Challenges of cryptographic engineering

Implementations must not leak information

- Side-channel attacks
  - Avoid timing attacks due to branching on input data
- Bug attacks
  - A bug with low probability can be deliberately triggered

Low-power devices

- Restricted execution environments
- Code must be small yet efficient due to limited memory
  - AVR: 8-bit RISC, typically 16Mhz and $\leq$ 32kb program size

Case study

Multi-precision multiplication

- Hutter and Schwabe [2014]: Hand-optimized routines for 8-bit AVR
- Building block for high-speed Curve25519 implementations

operand sizes	*method*
16-bit, 24-bit, 32-bit, 48-bit	'schoolbook' multiplication
48-bit, 64-bit, 80-bit, 96-bit	Karatsuba
128-bit, 160-bit, 192-bit, 256-bit	recursive Karatsuba

Case study

Multi-precision multiplication

- Hutter and Schwabe [2014]: Hand-optimized routines for 8-bit AVR
- Building block for high-speed Curve25519 implementations

operand sizes	*method*
16-bit, 24-bit, 32-bit, 48-bit	'schoolbook' multiplication
48-bit, 64-bit, 80-bit, 96-bit	Karatsuba
128-bit, 160-bit, 192-bit, 256-bit	recursive Karatsuba

Subtractive Karatsuba multiplication

To multiply two n-bit integers A and B,

- Split $A = 2^{n/2}A_h + A_l$, and $B = 2^{n/2}B_h + B_l$

Subtractive Karatsuba multiplication

To multiply two n-bit integers A and B,

- Split $A = 2^{n/2}A_h + A_l$, and $B = 2^{n/2}B_h + B_l$
- Multiply both halves: $L = A_l \cdot B_l$ and $H = A_h \cdot B_h$

Subtractive Karatsuba multiplication

To multiply two n-bit integers A and B,

- Split $A = 2^{n/2}A_h + A_l$, and $B = 2^{n/2}B_h + B_l$
- Multiply both halves: $L = A_l \cdot B_l$ and $H = A_h \cdot B_h$
- Let $M = (A_l - A_h) \cdot (B_l - B_h)$

Subtractive Karatsuba multiplication

To multiply two n-bit integers A and B,

- Split $A = 2^{n/2}A_h + A_l$, and $B = 2^{n/2}B_h + B_l$
- Multiply both halves: $L = A_l \cdot B_l$ and $H = A_h \cdot B_h$
- Let $M = (A_l - A_h) \cdot (B_l - B_h)$
- Compute the result as

$$A \cdot B = L + 2^{n/2}(L + H - M) + 2^n H$$

Subtractive Karatsuba multiplication

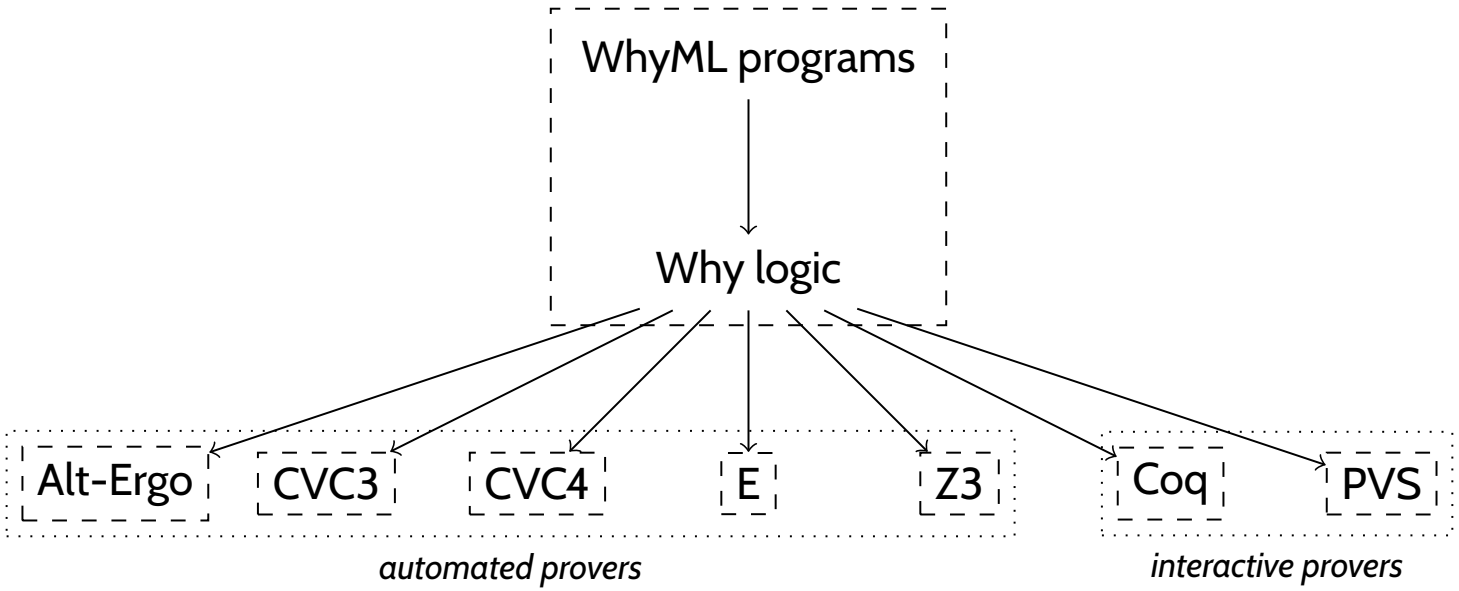
To multiply two n-bit integers A and B,

- Split $A = 2^{n/2}A_h + A_l$, and $B = 2^{n/2}B_h + B_l$
- Multiply both halves: $L = A_l \cdot B_l$ and $H = A_h \cdot B_h$
- Let $M = (A_l - A_h) \cdot (B_l - B_h)$
- Compute the result as

$$A \cdot B = L + 2^{n/2}(L + H - M) + 2^n H$$

Actual code slightly more ingenuous than this!

The Why3 verification platform



Highly useful: *abstract blocks*, *bit-vector theories*, *type invariants*

Design goals

Principle of least astonishment

- Formalization must be easy to understand and trust
- Transforming assembly $\rightarrow$ Why3 must be as simple as possible
- Perform proofs using a high degree of automation

```
let karatsuba64(): unit
  ensures { uint 16 mem result
            = old (uint 8 mem arg1 * uint 8 mem arg2) }
= instruction;
  instruction;
  .
  .
  .
```

Overview of approach

Model the machine architecture

- Instructions represented by WhyML functions
- Use Why3's type system to enforce internal consistency

Logical partitioning of code into *abstract blocks*

- Summarizes partial results leading to main correctness theorem
- Reduces proof context seen by automatic provers

Reasoning about machine memory

Machine integers

Use type invariants to enforce bytes are always true 8-bit values:

```
type address_space = { mutable data: map int int }
  invariant { forall i. 0 <= self.data[i] < 256 }
```

Allows lemmas about address spaces, e.g. $0 \leq m[i] \cdot m[j] \leq 65025$

Representation of multi-precision integers

$$\texttt{uint}\ n\ A\ b = \sum_{0 \leq i < n} 2^{8i} \cdot A[i+b]$$

Underspecification of AVR instruction set architecture

Features modelled

- 32 registers, (simplified) SRAM & stack
- *Carry flag* and *Transfer flag*

Instructions

Arithmetic	`ADC, ADD, SUB, SBC, SBIW, INC, DEC, MUL`
Bit manipulation	`ASR, CLR, COM, EOR, BLD, BST`
Data access	`MOV, MOVW, LD, LDD, STD, POP, PUSH`
Control flow	*omitted*

Example: ADD instruction

```
let add (dst src: register): unit
  writes { reg, cf }
  ensures { reg = old (reg[dst <- mod (reg[dst]+reg[src]) 256])
  ensures { ?cf = old (div (reg[dst]+reg[src]) 256) }
=
  let rd  = BV8.of_int (Map.get reg.data dst) in
  let rr  = BV8.of_int (Map.get reg.data src) in
  let rd' = BV8.add rd rr in

  reg.data <- Map.set reg.data dst (BV8.to_uint rd');
  cf.value <- (BV8.nth rd 7 && BV8.nth rr 7 ||
               BV8.nth rd 7 && not BV8.nth rd' 7 ||
               not BV8.nth rd' 7 && BV8.nth rr 7)
```


Divide and conquer

Verifying branch-free code = formula simplification

```
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

Verifying branch-free code = formula simplification

```
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

Verifying branch-free code = formula simplification

```
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

Verifying branch-free code = formula simplification

```
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

Verifying branch-free code = formula simplification

```
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

Verifying branch-free code = formula simplification

```
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

Verifying branch-free code = formula simplification

```
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

Verifying branch-free code = formula simplification

```
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
```

Abstract blocks in Karatsuba

Even branch-free assembly code has some discernible structure:

1. Compute $L = A_l \cdot B_l$
2. Compute $|A_l - A_h|$ and $|B_l - B_h|$
3. Compute $H = A_h \cdot B_h$ and $L + 2^{n/2} \cdot H$
4. Compute $M = |A_l - A_h| \cdot |B_l - B_h|$
5. Compute $L + 2^{n/2}(L + H) + 2^n H$.
6. Conditionally negate $2^{n/2} M$
7. Add the results of step 5 and 6

Design trade-off

Register file modelled as a single mutable array

- *'Nicer'*: data in registers and SRAM treated uniformly
- Abstract blocks obscure which registers are preserved

Alternative: model registers as individual variables?

- Runs into Why3 type system restrictions on mutable data

Design trade-off

Register file modelled as a single mutable array

- *'Nicer'*: data in registers and SRAM treated uniformly
- Abstract blocks obscure which registers are preserved

Alternative: model registers as individual variables?

- Runs into Why3 type system restrictions on mutable data

Answer: do both!
Maintain a *ghost* copy of the register file

- Reduces annotations needed and verification time

Verification efficiency

	program size	*annotations*	*cpu time*	*provers*
schoolbook				
16×16	13 lines	1	0s	CVC4
24×24	33 lines	1	1s	CVC4
32×32	59 lines	1	4s	CVC4
40×40	93 lines	1	10s	CVC4
48×48	136 lines	1	33s	CVC4
Karatsuba				
48×48	169 lines	23	52s	CVC4, CVC3, E
64×64	286 lines	21	96s	CVC4, CVC3, E
80×80	411 lines	31	215s	CVC4, CVC3, E
96×96	611 lines	39	906s	CVC4, CVC3, E

Discovered assumptions for correctness

Argument aliasing
Karatsuba implementations are correct *provided* that inputs and output do not use the same locations in memory

- Easy to 'fix' this restriction in 48-bit and 64-bit version
- Unlikely to be a problem in 80-bit version
  - $\mathtt{X} \leftarrow \mathtt{X} \cdot \mathtt{Y}$ still correct
- More restrictive precondition in 96-bit Karatsuba
  - $\mathtt{X} \leftarrow \mathtt{X} \cdot \mathtt{Y}$ *not* correct

Potentially a problem for the user

Conclusion

Features of this approach

- Make verification seem obvious
- Choose models that facilitate automatic proofs
  - OK as long as models are consistent?
- Identifying and specifying abstract blocks still labour-intensive

Suggestions for future work

- Multiple-level Karatsuba, 128-bit just needs more time
- External validation of machine model, however…
  - Formalization is not executable with current Why3
  - Requires a formal semantics (or testing on hardware)
- More control over proof context would be helpful!

This frame intentionally left blank

Subtractive Karatsuba multiplication

To multiply two n-bit integers A and B,

- Split $A = 2^{n/2}A_h + A_l$, and $B = 2^{n/2}B_h + B_l$
- Multiply both halves: $L = A_l \cdot B_l$ and $H = A_h \cdot B_h$
- Let $M = (A_l - A_h) \cdot (B_l - B_h)$
- Compute the result as

$$A \cdot B = L + 2^{n/2}(L + H - M) + 2^n H$$

Subtractive Karatsuba multiplication

To multiply two n-bit integers A and B,

- Split $A = 2^{n/2}A_h + A_l$, and $B = 2^{n/2}B_h + B_l$
- Multiply both halves: $L = A_l \cdot B_l$ and $H = A_h \cdot B_h$
- Let $M = |A_l - A_h| \cdot |B_l - B_h|$
- If $A_l \geq A_h \iff B_l \geq B_h$, obtain the result as

$$A \cdot B = L + 2^{n/2}(L + H - M) + 2^n H$$

Subtractive Karatsuba multiplication

To multiply two n-bit integers A and B,

- Split $A = 2^{n/2}A_h + A_l$, and $B = 2^{n/2}B_h + B_l$
- Multiply both halves: $L = A_l \cdot B_l$ and $H = A_h \cdot B_h$
- Let $M = |A_l - A_h| \cdot |B_l - B_h|$
- If $A_l \geq A_h \iff B_l \geq B_h$, obtain the result as

$$A \cdot B = L + 2^{n/2}(L + H - M) + 2^n H$$

- Otherwise, obtain the result as

$$A \cdot B = L + 2^{n/2}(L + H + M) + 2^n H$$

Subtractive Karatsuba multiplication

To multiply two n-bit integers A and B,

- Split $A = 2^{n/2}A_h + A_l$, and $B = 2^{n/2}B_h + B_l$
- Multiply both halves: $L = A_l \cdot B_l$ and $H = A_h \cdot B_h$
- Let $M = |A_l - A_h| \cdot |B_l - B_h|$
- If $A_l \geq A_h \iff B_l \geq B_h$, obtain the result as

$$A \cdot B = L + 2^{n/2}(L + H - M) + 2^n H$$

- Otherwise, obtain the result as

$$A \cdot B = L + 2^{n/2}(L + H + M) + 2^n H$$

Problem: conditional negations!

Typical assembly tricks

- $A_l - A_h$ sets the carry flag if (and only if) $A_l < A_h$
- *Subtract with carry* can give us `0x00` or `0xFF` in constant time
- Note that in two's complement,

$$\begin{aligned} -x &= (\texttt{NOT}\ x) + 1 \\ &= (x \oplus \texttt{0xFF}) - \texttt{0xFF} \end{aligned}$$

Typical assembly tricks

- $A_l - A_h$ sets the carry flag if (and only if) $A_l < A_h$
- *Subtract with carry* can give us `0x00` or `0xFF` in constant time
- Note that in two's complement,

$$\begin{aligned} -x &= (\texttt{NOT}\ x) + 1 \\ &= (x \oplus \texttt{0xFF}) - \texttt{0xFF} \end{aligned}$$

Solution

$\{\texttt{R1} = x, \texttt{R2} = y\}$

```
SUB R1, R2
```

$\{\texttt{R1} = x - y \pmod{256}\}$

```
SBC R16, R16
EOR R1, R16
SUB R1, R16
```

$\{\texttt{R1} = |x - y|\}$

```
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;
```