Pose |[A0]| * |[A1]| en libre $A2^S$ détruit A0-A5 D0-D6

```
XMUL1 :MOVEQ #0,D0
       MOVEQ #0,D1
       :ENTR          → { MOVE (A0)+,D0
                          MOVE (A1)+,D1
                          AND #$5FFF,D0
                          AND #$5FFF,D1
                          BCLR #14,D0
       BEQ  GD64
       BCLR #14,D1
       BEQ  GD60
       MULU D1,D0
XPOSEL :CMP.L #$10000,D0
       BCS  XPOSED0
       MOVE.L A6,A2
       MOVE #4,(A6)+
       MOVE.L D0,(A6)+
       BRA  VERA6

GD60 : EXG  D0,D1
       MOVE.L A1,A0

GD61 :CMP.L #1,D1
       BCS  XPOSEZ     0*[A0] → 0
       BNE  GD62
GD610 : SUBQ #2,A0     1* [A0] → [A0)
       ADDQ #2,D0
                       ← { BSR  XPOSEA0D0   recopie en libre
                           AND  #$5FFF,(A2)
                           RTS

GD62 :CMP  #4,d0
       BGT  GD620
X      BNE  \1
       MOVE.L (A0),D0
       BRA  \2
    \1 :MOVE (A0),D0
    \2 :MULU.L D1,D1,D0     d1.L * d0.L → d1:d0
XPOSEQ  :TST.L D1                pose d1.L:d0.L
       BEQ  XPOSEL
       MOVE.L A6,A2
       CMP.L #$10000,D1
       BCS  \1
       MOVE #8,(A6)+
       MOVE.L D1,(A6)+
       BRA  \2
    \1 :MOVE #6,(A6)+
       MOVE D1,(A6)+
    \2 :MOVE.L D0,(A6)+
       BRA  VERA6
```

← Pose d1.L * [A0] d0

```
GD620 :CMP  #8,D0
X          BGT   GD621
           BNE   \1
           MOVE.L (A0)+,D0
           BRA   \2
       \1: MOVE  (A0)+,D0

       \2: MULU.L  d1, d2,d0
           MOVE.L (A0)+,d4
           MOVEQ #0,d6
           MULU.L  d4, d3,d1
           ADD.L   d3,d0
           ADDX.L  d6,d2

           EXG   d0,d1

XPOSET :TST.L  d2
           BEQ   XPOSEQ
           MOVE.L  A6,A2
           CMP.L  #$10000,D2
           BCS  \1
           MOVE  #12,(A6)+
           MOVE.L  D2,(A6)+
           BRA   \2
       \1: MOVE  #10,(A6)+
           MOVE   D2,(A6)+
       \2: MOVE.L  D1,(A6)+
           MOVE.L  D0,(A6)+
           BRA   VERA6
```

```
      d0:d4
 x       d1
 ----------
      d3:d1
   d2:d0
 ----------
   d2:d0:d1
```

d2:d1:d0

pose d2.L : d1.L : d0.L

(repete XPOSEQ10)

(après GD620)

```
GD621:CMP  #12,d0
x     BGT  GD622
      BNE  \1
      MOVE.L (A0)+,D0
      BRA  \2
\1:   MOVE (A0)+,D0
\2:   MULU.L d1,d3,d0
      MOVE.L (A0)+,d4
      MOVEQ #0,d6
      MULU.L d1,d5,d4
      MOVE.L (A0)+,d2
      MULU.L d1,d1,d2
      ADD.L  d4,d1
      ADDX.L d5,d0
      ADDX.L d6,d3
      EXG  d0,d2
XPOSEQQ:TST.L d3
      BEQ  XPOSET
      MOVE.L A6,A2
      CMP.L #$10000,d3
      BCS  \1
      MOVE #16,(A6)+
      MOVE.L D3,(A6)+
      BRA  \2
\1:   MOVE #14,(A6)+
      MOVE D3,(A6)+
\2:   MOVE.L D2,(A6)+
      MOVE.L D1,(A6)+
      MOVE.L D0,(A6)+
      BRA  VERA6
```

```
      d0:d4:d2
            d1
----------------
d3:d0
   d5:d4
      d1:d2
----------------
d3:d0:d1:d2

d3:d2:d1:d0
```

```
GD622:MOVE.L  A6,-(SP)          d1.L * [A0]  D0>4
      LEA  8(A6,D0.W),A2   ⊗
      BSR  VERIFM2
      MOVE.L  A2,-(SP)

      MOVE  -(A0),-(SP)  } mot
      MOVE.L  A0,-(SP)   } ⊙
      CLR  (A0)
      ADDQ  #2,D0
      ADD  D0,A0
      ASR  #2,D0
      MOVE  D0,D2          ab de L de A0
      SUBQ  #1,D0
      MOVEQ  #0,D5
      MOVEQ  #0,D2
GD63 :MOVE.L  -(A0),D3
      MULU.L  D1,D4,D3
      ADD.L  D5,D3

      ADDX.L  D2,D4
      MOVE.L  D3,-(A2)
      MOVE.L  D4,D5
      DBRA  D0,GD63
      MOVE.L  D5,-(A2)
      BRA  GD66
```

si d0 ≠ 4 [0] A0
d0 augmenté de 2

rectifie la longueur

```
GD64: BCLR   #14,D1
      BNE    GD61            → d1 * [Ao]
      LEA    (Ao,DO.W),A4
      LEA    (A1,D1.W),A3
      MOVE.L A4,D4
      MOVE.L A3,D3
 \1:  TST    -(A4)
      BEQ    \1
      ADDQ   #2,A4
 \2:  TST    -(A3)
      BEQ    \2
      ADDQ   #2,A3
      SUB.L  A4,D4
      SUB.L  A3,D3
      SUB    D4,Do
      SUB    D3,D1
      ADD    D4,D3           en tout d3 mots ôtés
      BEQ    GD645
      MOVE   D3,-(SP)
      BSR    GD645
      MOVE   (SP)+,D3
      ADD    D3,(A2)
      CMP    #$2000,(A2)
      BCC    ERRDP
      MOVE.L A6,Ao
      ADD    D3,A6
      BSR    VERA6
      ASR    #1,D3           ajoute d3 [0]
      SUBQ   #1,D3
 \3:  CLR    (Ao)+
      DBRA   D3,\3
      RTS
```

ôte des [0] (mots) de fin

```
\A:   MOVE   (A2),Do   ⊗  (SP) ↳ multiplie [A2] (en lire) par 2^8D3
      BCLR   #14,Do
      BEQ    \2A
      MOVE   Do,(A6)+
      MOVEQ  #2,Do
\2A:  ADD    D3,Do
      CMP    #$2000,do
      BCC    ERRDP
      MOVE   do,(A2)
```

devant GD65 (ainsi P8)


```
\3:CMP #8,d0
X     BGT  GD65
      BNE  \4
      MOVE.L (A0)+,d0
      BRA  \5
\4:MOVE  (A0)+,d0
\5:CMP #8,d1
      BNE  \6
      MOVE.L (A1)+,d1
      BRA  \7
\6:MOVE  (A1)+,d1
\7:MOVE.L (A0)+,d6
      MOVE.L (A1)+,d3
      MOVE.L  d3,d4
      MULU.L  d6,d5,d4   ← MOVE.L d4,a0
      MULU.L  d0,d2,d3
      MOVEQ  #0,d4
      ADD.L  d3,d5
      ADDX.L  d4,d2
      MULU.L  d1,d3,d6
      MULU.L  d0,d0,d1
      ADD.L  d6,d5
      ADDX.L  d3,d2
      ADDX.L  d4,d0
      ADD.L  d1,d2
      ADDX.L  d4,d0
      MOVE.L  d0,d3
      MOVE.L  d5,d1
      MOVE.L  a0,d0
      BRA  XPOSEQQ      pose d3:d2:d1:d0
```

cas d0 et d1 = 6 ou 8

$d0^e : d6^e$

$d1^e : d3^e$

$d5 : a0$ } $d3^e * d0^e : d6^e$

$d2 : d3$

$d2 : d5 : a0$

$d3 : d6$ } $d1^e * d0^e : d6^e$

$d0 : d1$?

$d0 : d2 : d5 : a0$

$\neq 1 \neq 3 \neq 4 \neq 4$

Pose en litre A2 :


```
GD645:CMP  D0,D1
      BLE  \1
      EXG  D0,D1
      EXG  A0,A1          ← EXG A3,A4
\1 :  CMP  #4,d1          d1 ≤ d0
      BGT  \3
      BNE  \2
      MOVE.L (A1),d1      si d1=2 ou 4
      BRA  GD61           calcul simple
\2:   MOVE (A1),d1
      BRA GD61
```

si $\beta = 1.2^k$:

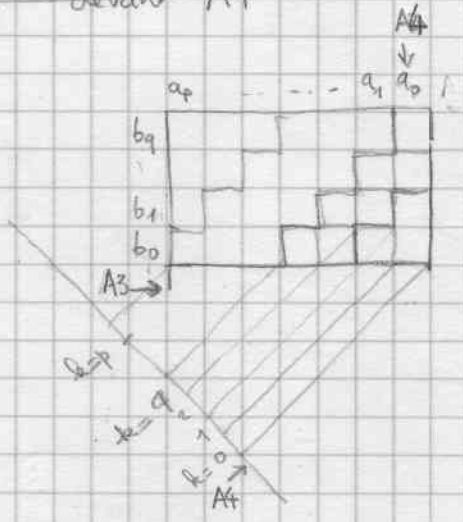
```
CMP #1,d1          ⊗
BNE GD62           ⊗
MOVE D0,(A6)+
BSR XPOSEA0D0
SUBQ #2,A2
RTS
```

< 8+

(suite GD622)

```
GD65 :MOVE.L A6,-(SP)          vérif mémoire
      LEA 10(A6,d0.W),A2       ⊗
      ADD D1,A2
      BSR VERIFM2              ← MOVE.L A2,-(SP)
      MOVE -(A0),-(SP)         ← MOVE.L A0,-(SP) sauve -2(a0), -2(a1)
      MOVE -(A1),-(SP)
      MOVEM.L D7/A1,-(SP)
      CLR (A0)
      CLR (A1)
      ADDQ #2,D0
      ADDQ #2,D1
      ASR #2,D0            nb de L  avant A2
      ASR #2,D1            nb de L  avant A3
      MOVE D0,D2
      ADD  D1,D2           nb de L devant A4
      SUBQ #1,D0           D0=p
      SUBQ #1,D1           D1=q
      SUBQ #4,A4

      MOVEM D0/D1,-(SP)
```



```
    MOVEQ #0,d2
    MOVEQ #0,d6   } retenue
    MOVEQ #0,d7
    MOVEQ #0,D3
\1  :MOVE  D3,D4
\2  :MOVEQ #0,D5
    MOVE.L A4,A0
    MOVE.L A3,A1
\3  :MOVE.L -(A1),D1
    MULU.L (A0)+,D0,D1
    ADD.L  D1,D7
    ADDX.L D0,D6
    ADDX.L D2,D5
    DBRA   D4,\3
    MOVE.L D7,-(A2)
    MOVE.L D6,D7
    MOVE.L D5,D6
    ADDQ #1,D3
    SUBQ #4,A4
    CMP  2(SP),D3
    BLE  \1          → si   0≤k≤q
    MOVE 2(SP),D4    sinon d4=q
    CMP  (SP),D3
    BLE  \2          → si   q<k≤p
    SUBQ.L #4,A3     ↓ si   p<k≤p+q
    ADDQ.L #4,A4
    MOVE (SP),D4
    ADD  2(SP),D4    } d4 = p+q-k
    SUB  D3,D4
    BPL  \2
    MOVE.L D4,-(A2)      ⊗
    ADDQ #4,SP
    MOVEM.L (SP)+,D7/A1
    MOVE (SP)+,(A1)
```

boucle k=0 à p+q D3=k

d5:d6:d7
d0,d1

```
GD66 : MOVE.L (SP)+,A0          remet (a0)
       MOVE (SP)+,(A0)
       MOVE.L (SP)+,D2 fin      rectifie la longueur de A2
\1:    TST (A2)+
       BEQ \1
       SUBQ #2,A2
       SUB.L A2,D2
       CMP #$2000,D2
       BCC ERRDP                → dépassement
       MOVE D2,-(A2)
       MOVE.L (SP),A0
       BSR XLB76
       MOVE.L (SP)+,A2
       RTS
```