Processamento Paralelo de dados (SIMD)

① a) void somaV(float *P, float *Q, float *R, int m) ← Arm C
(X0, X1, X2, W3)

```
.text
.global somaV
.type somaV, "function"

somaV: LSR w3, w3, #2  // nº de iterações: m/4
ciclo: CBZ w3, fim
       LDR Q0, [X0], #16
       LDR Q1, [X1], #16
       FADD V2.4S, V0.4S, V1.4S
       STR Q2, [X2], #16
       SUB w3, w3, #1
       B ciclo
fim:   RET
```

b) void altV(float *P, int m, float K) ← Arm C
(X0, W1, S0)

```
.text
.global altV
.type altV, "function"

altV: LSR w1, w1, #2      // nº de iterações: m/4
ciclo: CBZ w1, fim
       LDR Q1, [X0]
       FMUL V1.4S, V1.4S, V0.S[0]
       STR Q1, [X0], #16
       SUB w1, w1, #1
       B ciclo
fim:   RET
```

c) void msubV(float *P, float *Q, float *R, int m, float K) ← Arm C.
(X0, X1, X2, W3, S0)

```
.text
.global msubV
.type msubV, "function"

msubV: STP X29, X30, [SP, #-16]!
       MOV X29, SP
       FNEG S0, S0
       MOV X4, X0        // P = X4
       MOV X0, X1        // Q = X0
       MOV w1, w3        // m = w1
       MOV X5, X2        // R = X5
       MOV w6, w3        // m = w6
       BL altV           // X0 = -K x Q ← vetor
       MOV X1, X4
       MOV X2, X5
       MOV w3, w6
       BL somaV          // X0 = P - K x Q ← vetor
       LDP X29, X30, [SP], #16
       RET
```

② long int prodintV (int *R, int *S, int n) ← Em C.
(x0, x1, w2)

```
.text
.global prodintV
.type prodintV, "function"

prodintV: LSR w2, w2, #2
          MOV X5, #0
  ciclo : CBZ w2, fim
          LDR Q0, [X0], #16
          LDR Q1, [X1], #16
          MUL V2.4S, V0.4S, V1.4S
          ADDV S3, V2.4S
          SMOV X3, V3.S[0]
          ADD X5, X5, X3
          SUB w2, w2, #1
          B ciclo
  fim:    MOV X0, X5
          RET
```

③ long int conta_ocorr(char *V, long int n, char val) ← Em C
(x0, x1, w2)

```
.text
.global conta_ocorr
.type conta_ocorr, "function"

conta_ocorr: MOV X5, #0
             LSR X1, X1, #4          // nº iterações: n/16
             DUP V1.16B, w2
      ciclo: CBZ X1, fim
             LDR Q0, [X0], #16
             CM.eq V2.16B, V0.16B, V1.16B
             ADDV B3, V2.16B
             SMOV w3, V3.B[0]
             NEG w3, w3
             ADD X5, X5, X3
             SUB X1, X1, #1
             B ciclo

      fim:   MOV X0, X5
             RET
```

④ void inesatV(int *Z, int n, int x) ← Em C
(x0, w1, w2)

```
.text
.global inesatV
.type inesatV, "function"

inesatV: LSR W1, w1, #2
 ciclo:  CBZ w1, fim
         LDR Q0, [X0]
         DUP V1.4S, W2
         ADDSQ V0.4S, V0.4S, V1.4S
         STR Q0, [X0], #16
         SUB w1, w1, #1 / Beiclo.

fim: RET
```

⑤ void mirrorLeq(float *pt, int m) ← em C.

(annotations above the arguments: X0 over "float *pt", W1 over "int m")

```
.text
.global mirrorLeq
.type mirrorLeq,"function"

mirrorLeq: LSR W1, W1, #2
ciclo:     CBZ W1, fim
           LDR Q0,[X0]
           INS V1.S[3], V0.S[2]
           INS V1.S[2], V0.S[3]
           INS V1.S[1], V0.S[0]
           INS V1.S[0], V0.S[1]
           STR Q1, [X0], #16
           SUB W1, W1, #1
fim:       RET
```

⑥ double normV(double *ptV, long int m) ← em C

(annotations above the arguments: X0 over "double *ptV", X1 over "long int m")

```
.text
.global normV
.type normV, "function"

normV:   LSR X1, X1, #1        // nº de iterações : m/2
         FMOV D0, XZR
ciclo :  CBZ X1, fim
         LDR Q1, [X0], #16
         FMUL V1.2D, V1.2D, V1.2D
         ADDV D3, V1.2D
         FADD D0, D0, D3
         SUB X1, X1, #1
         B ciclo
fim:     FSQRT D0, D0
         RET
```

⑦ long int conta_inf(float *V, long int n, float lim) ← em C

(annotations above the arguments: X0 over "float *V", X1 over "long int n", S0 over "float lim")

```
.text
.global conta_inf
.type conta_inf, "function"

conta_inf:  LSR X1, X1, #2       // nº de iterações : m/4
            MOV X3, #0
            DUP V0.4S, V0.S[0]
ciclo :     CBZ X1, fim
            LDR Q1, [X0], #16
            CM.GE V2.4S, V1.4S, V0.4S
            ADDV S3, V2.4S
            SMOV X2, V3.S[0]
            ADD X2, X2, #4
            ADD X3, X3, X2
            SUB X1, X1, #1
            B ciclo
fim:        MOV X0, X3
            RET
```

⑧ void ajuste(float *X (X0), float *Y (X1), int n (W2), float da (S0)) ← em C

```
.text
.global ajusteSIMD
.type ajusteSIMD, "function"

ajusteSIMD: LSR W2, W2, #2          // nº de iterações: n/4
     ciclo: CBZ W2, fim
            LDR Q1, [X0], #16
            LDR Q2, [X1]
            FABS V1.4S, V1.4S
            FNMUL V3.4S, V1.4S, V0.S[0]
            FADD V3.4S, V2.4S, V3.4S
            STR Q3, [X1], #16
            SUB W2, W2, #1
            B ciclo
       fim: RET
```

⑨ void prod_complexosV(float *Z1 (X0), float *Z2 (X1), float *Z (X2), long int n (X3))

```
.text
.global prod_complexosV
.type prod_complexosV, "function"

prod_complexosV: STP X29, X30, [SP, #-16]!
                 MOV X29, SP
                 LSR X3, X3, #2
          ciclo: CBZ X3, fim
                 LDR Q0, [X0], #16
                 LDR Q1, [X1], #16
                 FMUL V2.4S, V0.4S, V1.4S   // a1xc1 | b1xd1 | a2xc2 | b2xd2
                 REV64 V3.4S, V1.4S
                 FMUL V3.4S, V0.4S, V3.4S   // a1xd1 | b1xc1 | a2xd2 | b2xc2
                 INS V4.S[0], V2.S[1]
                 FNEG S4, S4
                 INS V2.S[1], V4.S[0]
                 INS V4.S[0], V2.S[3]
                 FNEG S4, S4
                 INS V2.S[3], V4.S[0]
                 FADDP V4.4S, V2.4S, V3.4S
                 INS V5.S[0], V4.S[1]
                 INS V4.S[1], V4.S[2]
                 INS V4.S[2], V5.S[0]
                 STR Q4, [X2], #16
                 SUB X3, X3, #1
                 B ciclo
            fim: LDP X29, X30, [SP], #16
                 RET
```