

C-TD4- Pipeline logiciel TMS C6x

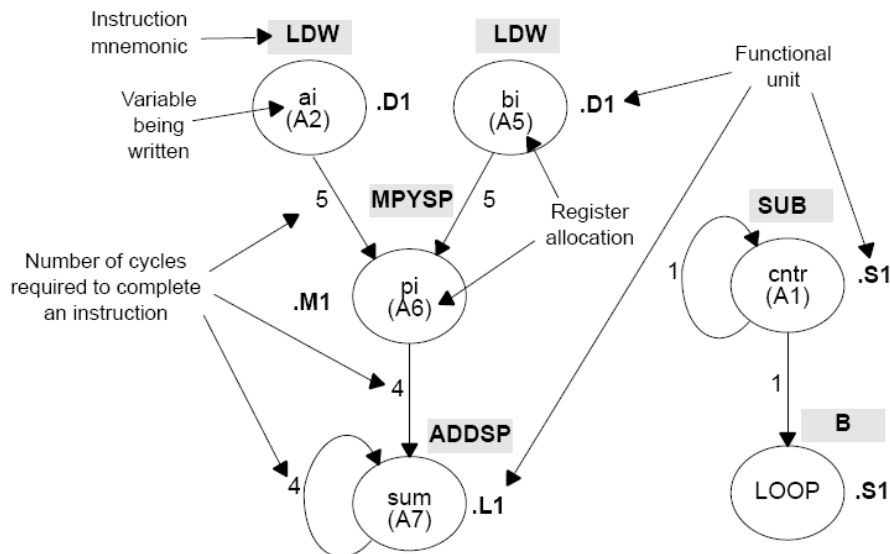
Produit scalaire flottant

```
float dotp(float a[], float b[])
{
  int i;
  float sum;
  sum = 0;

  for(i=0; i<100; i++)
    sum += a[i] * b[i];

  return(sum);
}
```

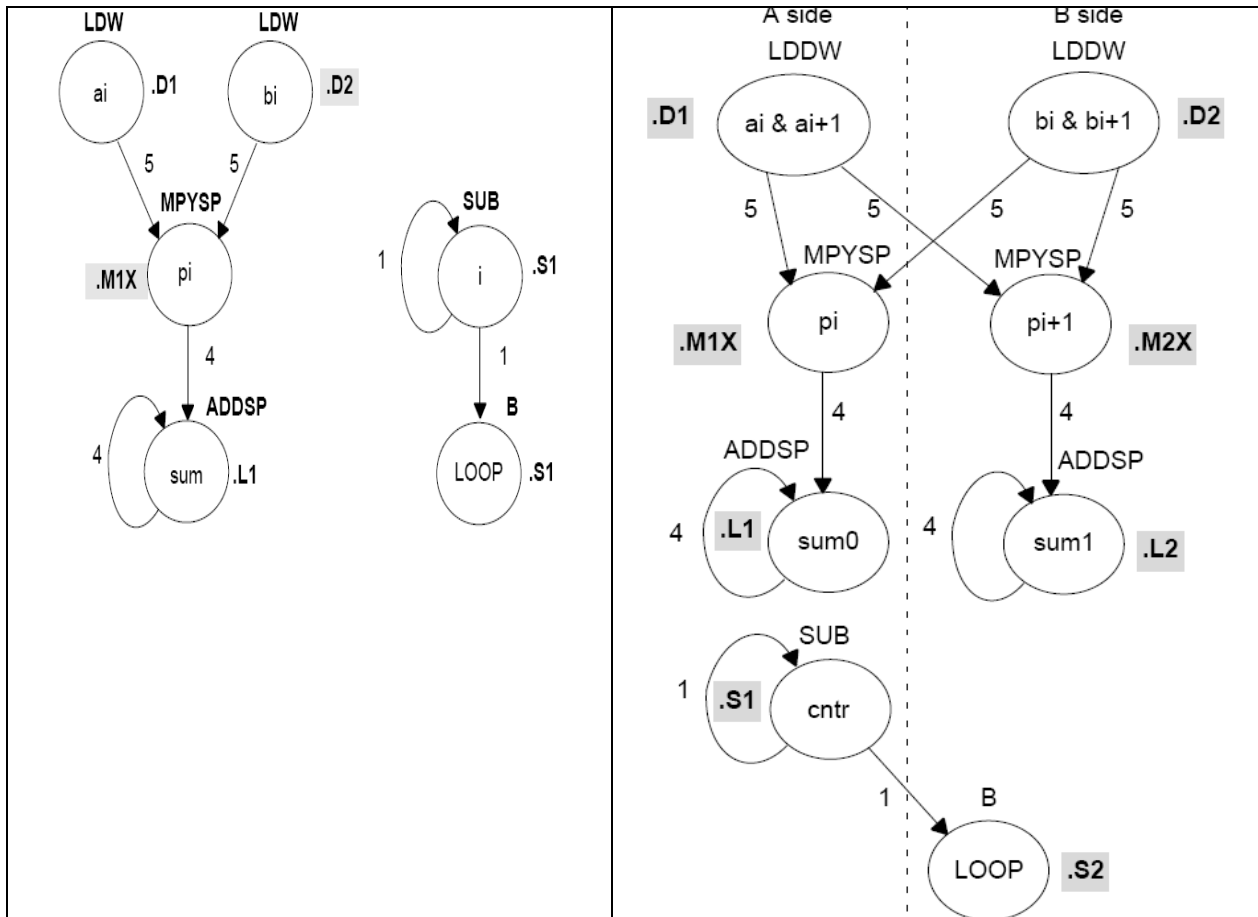
	LDW	.D1	*A4++,A2	; load ai from memory
	LDW	.D2	*A3++,A5	; load bi from memory
	MPYSP†	.M1	A2,A5,A6	; ai * bi
	ADDSP†	.L1	A6,A7,A7	; sum += (ai * bi)
	SUB	.S1	A1,1,A1	; decrement loop counter
[A1]	B	.S2	LOOP	; branch to loop



```

MVK .S1 100, A1 ; set up loop counter
ZERO .L1 A7 ; zero out accumulator
LOOP:
LDW .D1 *A4++,A2 ; load ai from memory
LDW .D1 *A3++,A5 ; load bi from memory
NOP 4 ; delay slots for LDW
MPYSP .M1 A2,A5,A6 ; ai * bi
NOP 3 ; delay slots for MPYSP
ADDSP .L1 A6,A7,A7 ; sum += (ai * bi)
NOP 3 ; delay slots for ADDSP
SUB .S1 A1,1,A1 ; decrement loop counter
[A1] B .S2 LOOP ; branch to loop
NOP 5 ; delay slots for branch
; Branch occurs here

```



```

        MVK    .S1    100, A1      ; set up loop counter
||      ZERO   .L1    A7          ; zero out accumulator
LOOP:
        LDW    .D1    *A4++,A2    ; load ai from memory
||      LDW    .D2    *B4++,B2    ; load bi from memory
        SUB    .S1    A1,1,A1     ; decrement loop counter
        NOP    2                      ; delay slots for LDW
[A1] B   .S2    LOOP             ; branch to loop
        MPYSP  .M1X   A2,B2,A6    ; ai * bi
        NOP    3                      ; delay slots for MPYSP
        ADDSP  .L1    A6,A7,A7    ; sum += (ai * bi)
; Branch occurs here
    
```

```

float dotp(float a[], float b[])
{
    int i;
    float sum0, sum1, sum;
    sum0 = 0;
    sum1 = 0;
    for(i=0; i<100; i+=2){
        sum0 += a[i] * b[i];
        sum1 += a[i + 1] * b[i + 1];
    }
    sum = sum0 + sum1;
    return(sum);
}
    
```

LDDW	*a++,ai1:ai0	; load a[i+0] & a[i+1] from memory
LDDW	*b++,bi1:bi0	; load b[i+0] & b[i+1] from memory
MPYSP	ai0,bi0,pi0	; a[i+0] * b[i+0]
MPYSP	ai1,bi1,pi1	; a[i+1] * b[i+1]
ADDSP	pi0,sum0,sum0	; sum0 += (a[i+0] * b[i+0])
ADDSP	pi1,sum1,sum1	; sum1 += (a[i+1] * b[i+1])
[cntr] SUB	cntr,1,cntr	; decrement loop counter
[cntr] B	LOOP	; branch to loop

LDDW	.D1	*A4++,A3:A2	; load ai and ai+1 from memory
LDDW	.D2	*B4++,B3:B2	; load bi and bi+1 from memory
MPYSP	.M1X	A2,B2,A6	; ai * bi
MPYSP	.M2X	A3,B3,B6	; ai+1 * bi+1
ADDSP	.L1	A6,A7,A7	; sum0 += (ai * bi)
ADDSP	.L2	B6,B7,B7	; sum1 += (ai+1 * bi+1)
[A1] SUB	.S1	A1,1,A1	; decrement loop counter
[A1] B	.S2	LOOP	; branch to top of loop

Unit / Cycle	0, 10, ...	1, 11, ...	2, 12, ...	3, 13, ...	4, 14, ...	5, 15, ...	6, 16, ...	7, 17, ...	8, 18, ...	9, 19, ...
.D1	LDDW									
.D2	LDDW									
.M1						MPYSP				
.M2						MPYSP				
.L1										ADDSP
.L2										ADDSP
.S1				SUB						
.S2					B					

Unit / Cycle	0	1	2	3	4	5	6	7	8	9, 10, 11...
.D1	LDDW	* LDDW	** LDDW	*** LDDW	**** LDDW	***** LDDW	***** LDDW	***** LDDW	***** LDDW	***** LDDW
.D2	LDDW	* LDDW	** LDDW	*** LDDW	**** LDDW	***** LDDW	***** LDDW	***** LDDW	***** LDDW	***** LDDW
.M1						MPYSP	* MPYSP	** MPYSP	*** MPYSP	**** MPYSP
.M2						MPYSP	* MPYSP	** MPYSP	*** MPYSP	**** MPYSP
.L1										ADDSP
.L2										ADDSP
.S1				SUB	* SUB	** SUB	*** SUB	**** SUB	***** SUB	***** SUB
.S2					B	* B	** B	*** B	**** B	***** B

```

        .global _dotp
_dotp:  .cproc   a, b

        .reg    sum, sum0, sum1, a, b
        .reg    a1:ai, b1:bi, pi, pi1

        MVK     50,cntr           ; cntr = 100/2
        ZERO   sum0              ; multiply result = 0
        ZERO   sum1              ; multiply result = 0

LOOP:   .trip 50
        LDDW   *a++,a1:ai        ; load ai & ai+1 from memory
        LDDW   *b++,b1:bi        ; load bi & bi+1 from memory
        MPYSP  ai,bi,pi          ; ai * bi
        MPYSP  a1,b1,pi1         ; ai+1 * bi+1
        ADDSP  pi,sum0,sum0      ; sum0 += (ai * bi)
        ADDSP  pi1,sum1,sum1     ; sum1 += (ai+1 * bi+1)
        [cntr] SUB   cntr,1,cntr  ; decrement loop counter
        [cntr] B     LOOP        ; branch to loop

        ADDSP  sum,sum1,sum0     ; compute final result

        .return sum

        .endproc

```

```

||      MVK     .S1   50,A1          ; set up loop counter
||      ZERO   .L1   A8             ; sum0 = 0
||      ZERO   .L2   B8             ; sum1 = 0
||      LDDW   .D1   A4++,A7:A6     ; load ai & ai + 1 from memory
||      LDDW   .D2   B4++,B7:B6     ; load bi & bi + 1 from memory
||
||      LDDW   .D1   A4++,A7:A6     ;* load ai & ai + 1 from memory
||      LDDW   .D2   B4++,B7:B6     ;* load bi & bi + 1 from memory
||
||      LDDW   .D1   A4++,A7:A6     ;** load ai & ai + 1 from memory
||      LDDW   .D2   B4++,B7:B6     ;** load bi & bi + 1 from memory
||
||      LDDW   .D1   A4++,A7:A6     ;*** load ai & ai + 1 from memory
||      LDDW   .D2   B4++,B7:B6     ;*** load bi & bi + 1 from memory
|| [A1] SUB   .S1   A1,1,A1         ; decrement loop counter
||
||      LDDW   .D1   A4++,A7:A6     ;**** load ai & ai + 1 from memory
||      LDDW   .D2   B4++,B7:B6     ;**** load bi & bi + 1 from memory
|| [A1] B     .S2   LOOP            ; branch to loop
|| [A1] SUB   .S1   A1,1,A1         ;* decrement loop counter
||
||      LDDW   .D1   A4++,A7:A6     ;***** load ai & ai + 1 from memory
||      LDDW   .D2   B4++,B7:B6     ;***** load bi & bi + 1 from memory
||      MPYSP  .M1X  A6,B6,A5        ; pi = a0 b0
||      MPYSP  .M2X  A7,B7,B5        ; pi1 = a1 b1
|| [A1] B     .S2   LOOP            ;* branch to loop
|| [A1] SUB   .S1   A1,1,A1         ;** decrement loop counter
||
||      LDDW   .D1   A4++,A7:A6     ;***** load ai & ai + 1 from memory
||      LDDW   .D2   B4++,B7:B6     ;***** load bi & bi + 1 from memory
||      MPYSP  .M1X  A6,B6,A5        ;* pi = a0 b0
||      MPYSP  .M2X  A7,B7,B5        ;* pi1 = a1 b1
|| [A1] B     .S2   LOOP            ;** branch to loop
|| [A1] SUB   .S1   A1,1,A1         ;*** decrement loop counter

```

```

        LDDW      .D1   A4++,A7:A6      ;***** load ai & ai + 1 from memory
||      LDDW      .D2   B4++,B7:B6      ;***** load bi & bi + 1 from memory
||      MPYSP     .M1X  A6,B6,A5        ;** pi = a0 b0
||      MPYSP     .M2X  A7,B7,B5        ;** pil = a1 b1
|| [A1] B        .S2   LOOP            ;*** branch to loop
|| [A1] SUB      .S1   A1,1,A1          ;**** decrement loop counter

        LDDW      .D1   A4++,A7:A6      ;***** load ai & ai + 1 from memory
||      LDDW      .D2   B4++,B7:B6      ;***** load bi & bi + 1 from memory
||      MPYSP     .M1X  A6,B6,A5        ;*** pi = a0 b0
||      MPYSP     .M2X  A7,B7,B5        ;*** pil = a1 b1
|| [A1] B        .S2   LOOP            ;**** branch to loop
|| [A1] SUB      .S1   A1,1,A1          ;***** decrement loop counter

LOOP:
        LDDW      .D1   A4++,A7:A6      ;***** load ai & ai + 1 from memory
||      LDDW      .D2   B4++,B7:B6      ;***** load bi & bi + 1 from memory
||      MPYSP     .M1X  A6,B6,A5        ;**** pi = a0 b0
||      MPYSP     .M2X  A7,B7,B5        ;**** pil = a1 b1
||      ADDSP     .L1   A5,A8,A8        ; sum0 += (ai bi)
||      ADDSP     .L2   B5,B8,B8        ; sum1 += (ai+1 bi+1)
|| [A1] B        .S2   LOOP            ;***** branch to loop
|| [A1] SUB      .S1   A1,1,A1          ;***** decrement loop counter
; Branch occurs here

        ADDSP     .L1X  A8,B8,A0        ; sum(0) = sum0(0) + sum1(0)

        ADDSP     .L2X  A8,B8,B0        ; sum(1) = sum0(1) + sum1(1)

        ADDSP     .L1X  A8,B8,A0        ; sum(2) = sum0(2) + sum1(2)

        ADDSP     .L2X  A8,B8,B0        ; sum(3) = sum0(3) + sum1(3)

        NOP                          ; wait for B0

        ADDSP     .L1X  A0,B0,A5        ; sum(01) = sum(0) + sum(1)

        NOP                          ; wait for next B0

        ADDSP     .L2X  A0,B0,B5        ; sum(23) = sum(2) + sum(3)

        NOP                          3

        ADDSP     .L1X  A5,B5,A4        ; sum = sum(01) + sum(23)

        NOP                          3
;

```

```

LOOP:      ADDSP     x,sum,sum
||        LDW      *xptr++,x
|| [cond] B        cond
|| [cond] SUB      cond,1,cond

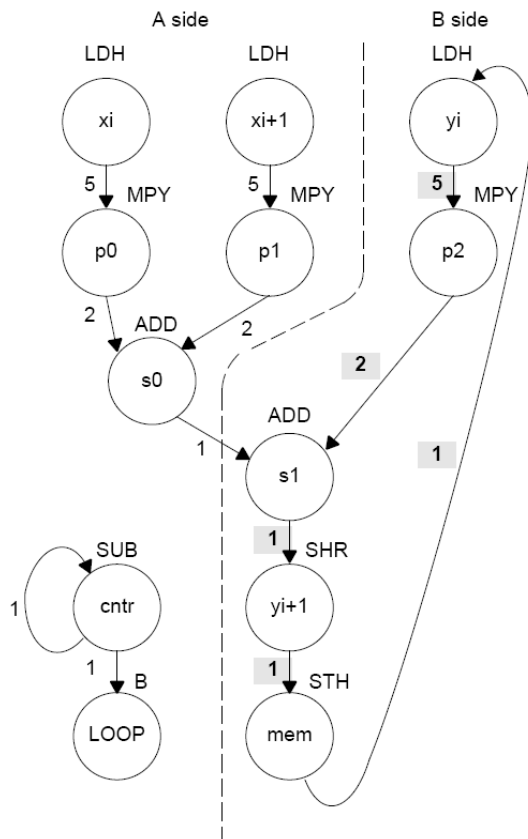
```

Cycle #	Pseudoinstruction	Current value of pseudoregister sum	Written expected result
0	ADDSP x(0), sum, sum	0	; cycle 4 sum = x(0)
1	ADDSP x(1), sum, sum	0	; cycle 5 sum = x(1)
2	ADDSP x(2), sum, sum	0	; cycle 6 sum = x(2)
3	ADDSP x(3), sum, sum	0	; cycle 7 sum = x(3)
4	ADDSP x(4), sum, sum	x(0)	; cycle 8 sum = x(0) + x(4)
5	ADDSP x(5), sum, sum	x(1)	; cycle 9 sum = x(1) + x(5)
6	ADDSP x(6), sum, sum	x(6)	; cycle 10 sum = x(2) + x(6)
7	ADDSP x(7), sum, sum	x(7)	; cycle 11 sum = x(3) + x(7)
8	ADDSP x(8), sum, sum	x(0) + x(4)	; cycle 12 sum = x(0) + x(8)
	• • •		
i + j†	ADDSP x(i+j), sum, sum	x(j) + x(j+4) + x(j+8) ... x(i-4+j)	; cycle i + j + 4 sum = x(j) + x(j+4) + x(j+8) ... x(i-4+j) + x(i+j)
	-		

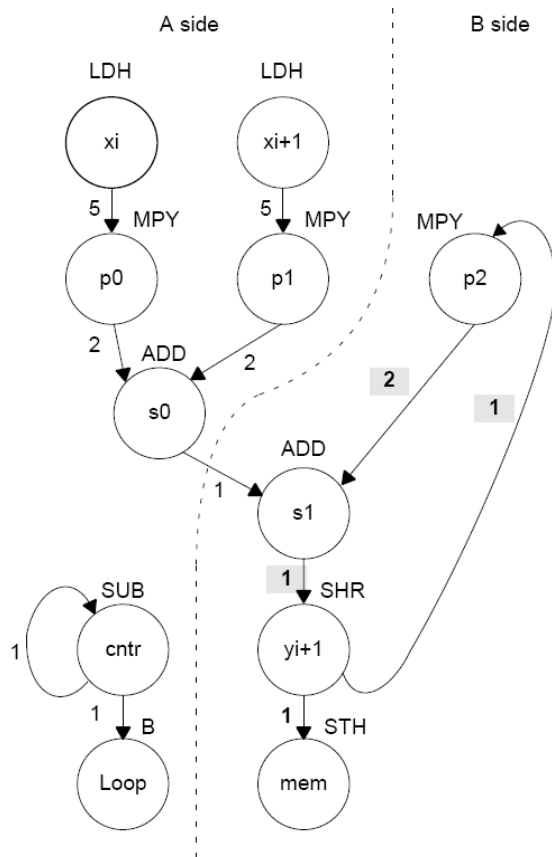
Filtre IIR

```
void iir(short x[],short y[],short c1, short c2, short c3)
{
  int i;
  for (i=0; i<100; i++) {
    y[i+1] = (c1*x[i] + c2*x[i+1] + c3*y[i]) >> 15;
  }
}
```

LDH	*xptr++,xi	; xi+1
MPY	c1,xi,p0	; c1 * xi
LDH	*xptr,xi+1	; xi+1
MPY	c2,xi+1,p1	; c2 * xi+1
ADD	p0,p1,s0	; c1 * xi + c2 * xi+1
LDH	*yptr++,yi	; yi
MPY	c3,yi,p2	; c3 * yi
ADD	s0,p2,s1	; c1 * xi + c2 * xi+1 + c3 * yi
SHR	s1,15,yi+1	; yi+1
STH	yi+1,*yptr	; store yi+1
[cntr]SUB	cntr,1,cntr	; decrement loop counter
[cntr]B	LOOP	; branch to loop



Unit(s)	Instructions	Total/Unit	Unit(s)	Instructions	Total/Unit
.M1	2 MPYs	2	.M2	MPY	1
.S1	B	1	.S2	SHR	1
.D1	2 LDHs	2	.D2	STH	1
.L1, .S1, or .D1	ADD & SUB	2	.L2 or .S2, .D2	ADD	1
Total non-.M units		5	Total non-.M units		3



LDH	*xptr++,xi	; xi+1
MPY	c1,xi,p0	; c1 * xi
LDH	*xptr,xi+1	; xi+1
MPY	c2,xi+1,p1	; c2 * xi+1
ADD	p0,p1,s0	; c1 * xi + c2 * xi+1
MPY	c3,y,p2	; c3 * yi
ADD	s0,p2,s1	; c1 * xi + c2 * xi+1 + c3 * yi
SHR	s1,15,y	; yi+1
STH	y,*yptr++	; store yi+1
[cntr] SUB	cntr,1,cntr	; decrement loop counter
[cntr] B	LOOP	; branch to loop

	LDH	.D1	*A4++,A2	; xi+1
	MPY	.M1	A6,A2,A5	; c1 * xi
	LDH	.D1	*A4,A3	; xi+1
	MPY	.M1X	B6,A3,A7	; c2 * xi+1
	ADD	.L1	A5,A7,A9	; c1 * xi + c2 * xi+1
	MPY	.M2X	A8,B2,B3	; c3 * yi
	ADD	.L2X	B3,A9,B5	; c1 * xi + c2 * xi+1 + c3 * yi
	SHR	.S2	B5,15,B2	; yi+1
	STH	.D2	B2,*B4++	; store yi+1
[A1]	SUB	.L1	A1,1,A1	; decrement loop counter
[A1]	B	.S1	LOOP	; branch to loop

Unit/Cycle	0	4	8, 12, 16, ...	Unit/Cycle	1	5	9, 13, 17, ...
.D1	LDH xi	* LDH xi	** LDH xi	.D1	LDH xi+1	* LDH xi+1	** LDH ci+1
.D2			ADD s0	.D2			
.M1				.M1		MPY p0	* MPY p0
.M2				.M2			
.L1				.L1		SUB cntr	* SUB cntr
.L2				.L2			ADD s1
.S1				.S1			
.S2				.S2			
1X				1X			
2X				2X			ADD s1
Unit/Cycle	2	6	10, 14, 18, ...	Unit/Cycle	3	7	11, 15, 19, ...
.D1				.D1			
.D2				.D2			STH yi+1
.M1		MPY p1	* MPY p1	.M1			
.M2				.M2		MPY p2	* MPY p2
.L1				.L1			
.L2				.L2			
.S1		B LOOP	* B LOOP	.S1			
.S2			SHR yi+1	.S2			
1X		MPY p1	* MPY p1	1X			
2X				2X		MPY p2	* MPY p2

```

.global _iir
_iir: .cproc x, y, c1, c2, c3

    .reg    xi, xi1, yi1
    .reg    p0, p1, p2, s0, s1, cntr

    MVK     100,cntr                ; cntr = 100

    LDH     .D2 *y++,yi1           ; yi+1

LOOP: .trip 100
    LDH     .D1 *x++,xi            ; xi
    MPY     .M1 c1,xi,p0           ; c1 * xi
    LDH     .D1 *x,xi1            ; xi+1
    MPY     .M1X c2,xi1,p1        ; c2 * xi+1
    ADD     .L1 p0,p1,s0          ; c1 * xi + c2 * xi+1
    MPY     .M2X c3,yi1,p2        ; c3 * yi
    ADD     .L2X s0,p2,s1         ; c1 * xi + c2 * xi+1 + c3 * yi
    SHR     .S2 s1,15,yi1         ; yi+1
    STH     .D2 yi1,*y++          ; store yi+1
[cntr] SUB .L1 cntr,1,cntr        ; decrement loop counter
[cntr] B   .S1 LOOP              ; branch to loop

    .endproc

```

```

    LDH     .D1 *A4++,A2          ; xi
    LDH     .D1 *A4,A3            ; xi+1
    LDH     .D2 *B4++,B2          ; load y[0] outside of loop
    MVK     .S1 100,A1            ; set up loop counter
    LDH     .D1 *A4++,A2          ; * xi
    [A1] SUB .L1 A1,1,A1          ; decrement loop counter
    ||     MPY .M1 A6,A2,A5        ; c1 * xi
    ||     LDH .D1 *A4,A3          ; * xi+1
    MPY     .M1X B6,A3,A7         ; c2 * xi+1
    || [A1] B .S1 LOOP            ; branch to loop
    MPY     .M2X A8,B2,B3         ; c3 * yi

LOOP:
    ADD     .L1 A5,A7,A9          ; c1 * xi + c2 * xi+1
    ||     LDH .D1 *A4++,A2        ; ** xi
    ADD     .L2X B3,A9,B5         ; c1 * xi + c2 * xi+1 + c3 * yi
    || [A1] SUB .L1 A1,1,A1        ; * decrement loop counter
    ||     MPY .M1 A6,A2,A5        ; * c1 * xi
    ||     LDH .D1 *A4,A3          ; ** xi+1
    SHR     .S2 B5,15,B2          ; yi+1
    ||     MPY .M1X B6,A3,A7        ; * c2 * xi+1
    || [A1] B .S1 LOOP            ; * branch to loop
    STH     .D2 B2,*B4++          ; store yi+1
    ||     MPY .M2X A8,B2,B3        ; * c3 * yi
    ; Branch occurs here

```

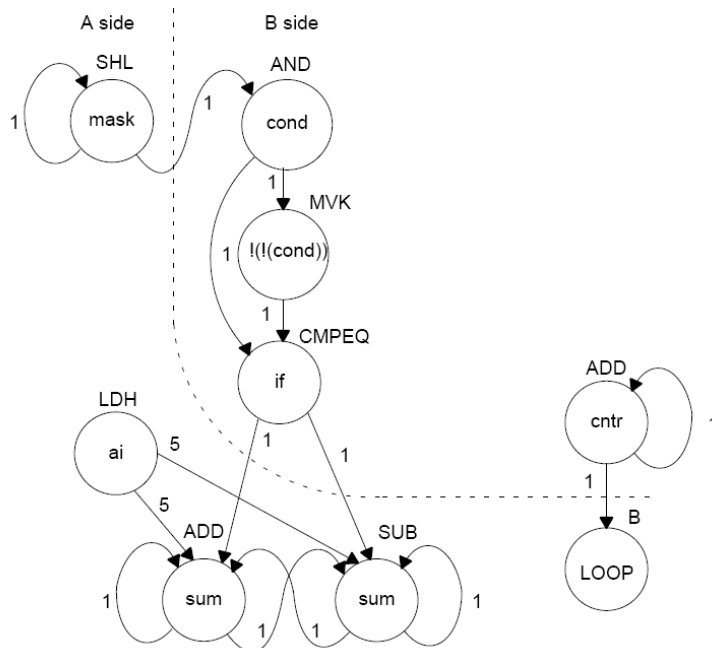
```

int if_then(short a[], int codeword, int mask, short theta)
{
    int i,sum, cond;

    sum = 0;
    for (i = 0; i < 32; i++){
        cond = codeword & mask;
        if (theta == !(!(cond)))
            sum += a[i];
        else
            sum -= a[i];
        mask = mask << 1;
    }
    return(sum);
}

```

	AND	codeword,mask,cond	; cond = codeword & mask
[cond]	MVK	1,cond	; !(!(cond))
	CMPEQ	theta,cond,if	; (theta == !(!(cond)))
	LDH	*aptr++,ai	; a[i]
[if]	ADD	sum,ai,sum	; sum += a[i]
[!if]	SUB	sum,ai,sum	; sum -= a[i]
	SHL	mask,1,mask	; mask = mask << 1;
[cntr]	ADD	-1,cntr,cntr	; decrement counter
[cntr]	B	LOOP	; for LOOP



(a) A side

(b) B side

Unit(s)	Instructions	Total/Unit	Unit(s)	Instructions	Total/Unit
.M1		0	.M2		0
.S1	SHL & B	2	.S2	MVK	1
.D1	LDH	1	.L2	CMPEQ	1
.L1, .S1, or .D1	ADD & SUB	2	.L2 or .S2	AND	1
			.L2, .S2, or .D2	ADD	1
Total non-M units		5	Total non-M units		4

```

.global _if_then
_if_then: .cproc a, cword, mask, theta

    .reg    cond, if, ai, sum, cntr

    MVK    32,cntr          ; cntr = 32
    ZERO   sum              ; sum = 0

LOOP:   .trip 32
    AND    .S2X   cword,mask,cond ; cond = codeword & mask
[cond]  MVK    .S2    1,cond        ; !(!(cond))
    CMPEQ  .L2    theta,cond,if    ; (theta == !(!(cond)))
    LDH    .D1    *a++,ai         ; a[i]
    [if]   ADD    .L1    sum,ai,sum ; sum += a[i]
    [!if]  SUB    .D1    sum,ai,sum ; sum -= a[i]
    SHL    .S1    mask,1,mask     ; mask = mask << 1;
[cntr]  ADD    .L2    -1,cntr,cntr ; decrement counter
[cntr]  B      .S1    LOOP        ; for LOOP

    .return sum

.endproc

```

```

    MVK    .S2    32,B0          ; set up loop counter

[B0]  ADD    .L2    -1,B0,B0     ; decrement counter

[B0]  ADD    .L2    -1,B0,B0     ; decrement counter
|| [B0] B      .S1    LOOP        ; for LOOP
||     LDH    .D1    *A4++,A5    ; a[i]

    SHL    .S1    A6,1,A6       ; mask = mask << 1;
||     AND    .S2X   B4,A6,B2    ; cond = codeword & mask

[B2]  MVK    .S2    1,B2        ; !(!(cond))
|| [B0] ADD    .L2    -1,B0,B0     ; decrement counter
|| [B0] B      .S1    LOOP        ; * for LOOP
||     LDH    .D1    *A4++,A5    ; * a[i]

    CMPEQ  .L2    B6,B2,B1      ; (theta == !(!(cond)))
||     SHL    .S1    A6,1,A6       ; * mask = mask << 1;
||     AND    .S2X   B4,A6,B2    ; * cond = codeword & mask
||     ZERO   .L1    A7          ; zero out accumulator

LOOP:
[B0]  ADD    .L2    -1,B0,B0     ; decrement counter
|| [B2] MVK    .S2    1,B2        ; * !(!(cond))
|| [B0] B      .S1    LOOP        ; ** for LOOP
||     LDH    .D1    *A4++,A5    ; ** a[i]

[B1]  ADD    .L1    A7,A5,A7     ; sum += a[i]
|| [!B1] SUB   .D1    A7,A5,A7     ; sum -= a[i]
||     CMPEQ  .L2    B6,B2,B1      ; * (theta == !(!(cond)))
||     SHL    .S1    A6,1,A6       ; ** mask = mask << 1;
||     AND    .S2X   B4,A6,B2    ; ** cond = codeword & mask
; Branch occurs here

```

```

        B        .S1    LOOP        ; for LOOP
||      LDH      .D1    *A4++,A5    ; a[i]
||      MVK      .S2    29,B0      ; set up loop counter

        SHL      .S1    A6,1,A6     ; mask = mask << 1;
||      AND      .S2X   B4,A6,B2    ; cond = codeword & mask

[B2]    MVK      .S2    1,B2        ; !(!(cond))
||      B        .S1    LOOP        ;* for LOOP
||      LDH      .D1    *A4++,A5    ;* a[i]

        CMPEQ   .L2    B6,B2,B1     ; (theta == !(!(cond)))
||      SHL      .S1    A6,1,A6     ;* mask = mask << 1;
||      AND      .S2X   B4,A6,B2    ;* cond = codeword & mask
||      ZERO     .L1    A7          ; zero out accumulator

LOOP:
[B0]    ADD      .L2    -1,B0,B0     ; decrement counter
|| [B2] MVK      .S2    1,B2        ;* !(!(cond))
|| [B0] B        .S1    LOOP        ;** for LOOP
||      LDH      .D1    *A4++,A5    ;** a[i]

[B1]    ADD      .L1    A7,A5,A7     ; sum += a[i]
|| [!B1] SUB     .D1    A7,A5,A7     ; sum -= a[i]
||      CMPEQ   .L2    B6,B2,B1     ;* (theta == !(!(cond)))
||      SHL      .S1    A6,1,A6     ;** mask = mask << 1;
||      AND      .S2X   B4,A6,B2    ;** cond = codeword & mask
; Branch occurs here

```

```

int unrolled_if_then(short a[], int codeword, int mask, short theta)
{
    int i,sum, cond;

    sum = 0;
    for (i = 0; i < 32; i+=2){
        cond = codeword & mask;
        if (theta == !(!(cond)))
            sum += a[i];
        else
            sum -= a[i];
        mask = mask << 1;

        cond = codeword & mask;
        if (theta == !(!(cond)))
            sum += a[i+1];
        else
            sum -= a[i+1];
        mask = mask << 1;
    }
    return(sum);
}

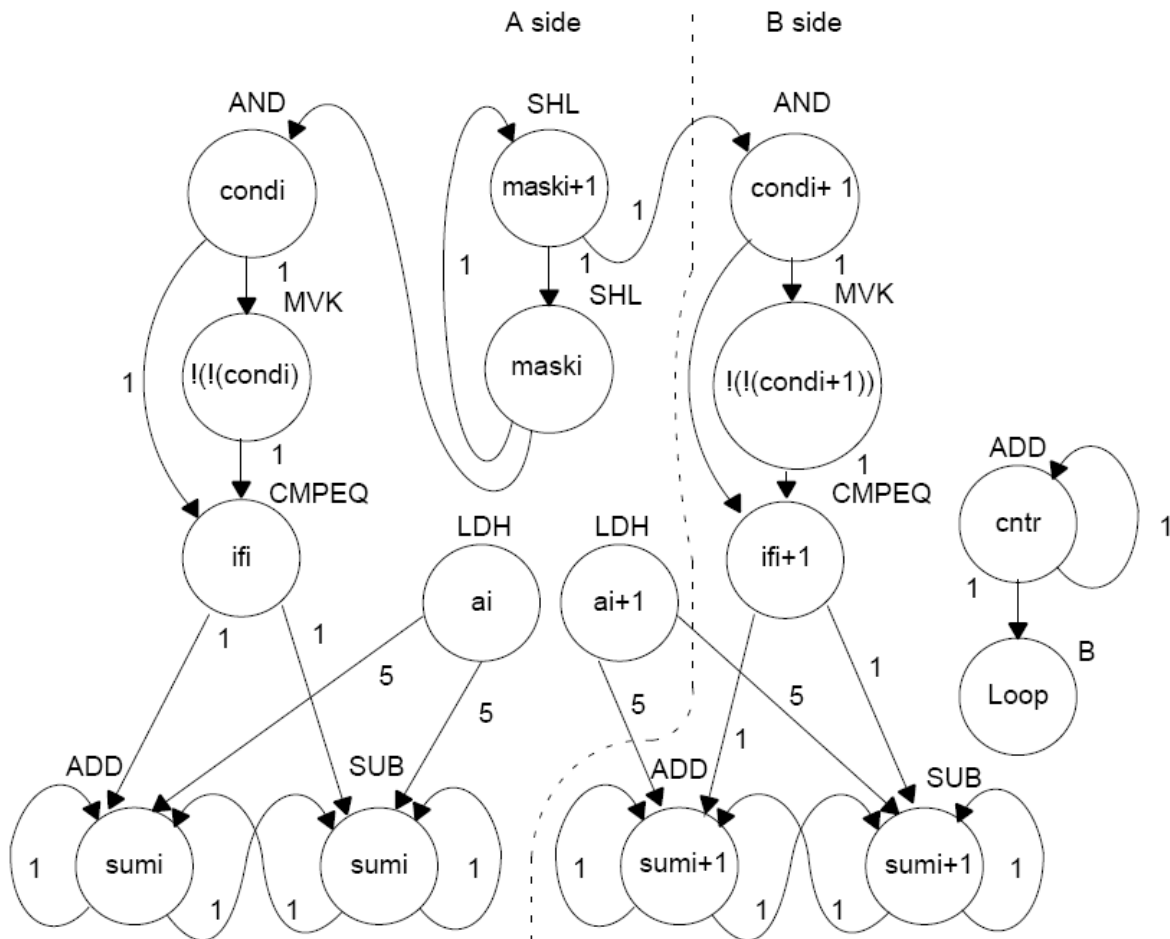
```

```

AND      codeword,maski,condi    ; condi = codeword & maski
[condi]  MVK      1,condi        ; !(!(condi))
CMPEQ   theta,condi,ifi        ; (theta == !(!(condi)))
LDH     *aptr++,ai             ; a[i]
[ifi]   ADD      sumi,ai,sumi    ; sum += a[i]
[!ifi]  SUB      sumi,ai,sumi    ; sum -= a[i]
SHL     maski,1,maski+1        ; maski+1 = maski << 1;

AND      codeword,maski+1,condi+1 ; condi+1 = codeword & maski+1
[condi+1]MVK  1,condi+1        ; !(!(condi+1))
CMPEQ   theta,condi+1,ifi+1    ; (theta == !(!(condi+1)))
LDH     *aptr++,ai+1          ; a[i+!]
[ifi+1]  ADD     sumi+1,ai+1,sumi+1 ; sum += a[i+1]
[!ifi+1] SUB    sumi+1,ai+1,sumi+1 ; sum -= a[i+1]
SHL     maski+1,1,maski       ; maski = maski+1 << 1;

[cntr]  ADD      -1,cntr,cntr    ; decrement counter
[cntr]  B        LOOP           ; for LOOP
    
```



(a) A side

(b) B side

Unit(s)	Instructions	Total/Unit	Unit(s)	Instructions	Total/Unit
.M1		0	.M2		0
.S1	MVK and 2 SHLs	3	.S2	MVK and B	2
.D1	2 LDHs	2	.L2	CMPEQ	1
.L1	CMPEQ	1	.L2 pr.S2	AND	1
.L1 or .S1	AND	1	.L2, .S2, or .D2	SUB and 2 ADDs	3
.L1, .S1, or .D1	ADD and SUB	2			
Total non-.M units		9	Total non-.M units		7

```

.global    _unrolled_if_then
_unrolled_if_then: .cproc    a, cword, mask, theta

    .reg    cword, mask, theta, ifi, ifi1, a, ai, ail, cntr
    .reg    cdi, cdi1, sumi, sumi1, sum

    MV      A4,a                ; C callable register for 1st operand
    MV      B4,cword            ; C callable register for 2nd operand
    MV      A6,mask             ; C callable register for 3rd operand
    MV      B6,theta            ; C callable register for 4th operand
    MVK     16,cntr             ; cntr = 32/2
    ZERO    sumi                ; sumi = 0
    ZERO    sumi1               ; sumi+1 = 0

LOOP:    .trip 32
    AND     .L1X cword,mask,cdi ; cdi = codeword & maski
    [cdi]   MVK     .S1 1,cdi    ; !(!(cdi))
    CMPEQ   .L1X theta,cdi,ifi   ; (theta == !(!(cdi)))
    LDH     .D1 *a++,ai         ; a[i]
    [ifi]   ADD     .L1 sumi,ai,sumi ; sum += a[i]
    [!ifi]  SUB     .D1 sumi,ai,sumi ; sum -= a[i]
    SHL     .S1 mask,1,mask     ; maski+1 = maski << 1;

    AND     .L2X cword,mask,cdi1 ; cdi+1 = codeword & maski+1
    [cdi1]  MVK     .S2 1,cdi1   ; !(!(cdi+1))
    CMPEQ   .L2 theta,cdi1,ifi1 ; (theta == !(!(cdi+1)))
    LDH     .D1 *a++,ail        ; a[i+1]
    [ifi1]  ADD     .L2 sumi1,ail,sumi1 ; sum += a[i+1]
    [!ifi1] SUB     .D2 sumi1,ail,sumi1 ; sum -= a[i+1]
    SHL     .S1 mask,1,mask     ; maski = maski+1 << 1;

    [cntr]  ADD     .D2 -1,cntr,cntr ; decrement counter
    [cntr]  B       .S2 LOOP      ; for LOOP

    ADD     sumi,sumi1,sum      ; Add sumi and sumi+1 for ret value

    .return sum

    .endproc

```

```

        MVK    .S2    16,B0        ; set up loop counter

        LDH    .D1    *A4++,A5     ; a[i]
|| [B0] ADD    .D2    -1,B0,B0     ; decrement counter

        LDH    .D1    *A4++,B5     ; a[i+1]
|| [B0] B      .S2    LOOP        ; for LOOP
|| [B0] ADD    .D2    -1,B0,B0     ; decrement counter
||        SHL    .S1    A6,1,A6    ; maski+1 = maski << 1;
||        AND    .L1X   B4,A6,A2   ; condi = codeword & maski

[A2] MVK    .S1    1,A2           ; !(!(condi))
||        AND    .L2X   B4,A6,B2   ; condi+1 = codeword & maski+1
||        ZERO   .L1    A7         ; zero accumulator

[B2] MVK    .S2    1,B2           ; !(!(condi+1))
||        CMPEQ  .L1X   B6,A2,A1   ; (theta == !(!(condi)))
||        SHL    .S1    A6,1,A6    ; maski = maski+1 << 1;
||        LDH    .D1    *A4++,A5   ; * a[i]
||        ZERO   .L2    B7         ; zero accumulator

LOOP:
        CMPEQ  .L2    B6,B2,B1     ; (theta == !(!(condi+1)))
|| [B0] ADD    .D2    -1,B0,B0     ; decrement counter
||        LDH    .D1    *A4++,B5   ; * a[i+1]
|| [B0] B      .S2    LOOP        ; * for LOOP
||        SHL    .S1    A6,1,A6    ; * maski+1 = maski << 1;
||        AND    .L1X   B4,A6,A2   ; * condi = codeword & maski

[A1] ADD    .L1    A7,A5,A7        ; sum += a[i]
|| [!A1] SUB   .D1    A7,A5,A7     ; sum -= a[i]
|| [A2] MVK    .S1    1,A2         ; * !(!(condi))
||        AND    .L2X   B4,A6,B2   ; * condi+1 = codeword & maski+1

[B1] ADD    .L2    B7,B5,B7        ; sum += a[i+1]
|| [!B1] SUB   .D2    B7,B5,B7     ; sum -= a[i+1]
|| [B2] MVK    .S2    1,B2         ; * !(!(condi+1))
||        CMPEQ  .L1X   B6,A2,A1   ; * (theta == !(!(condi)))
||        SHL    .S1    A6,1,A6    ; * maski = maski+1 << 1;
||        LDH    .D1    *A4++,A5   ; ** a[i]
        ; Branch occurs here

        ADD    .L1X   A7,B7,A4     ; move to return register

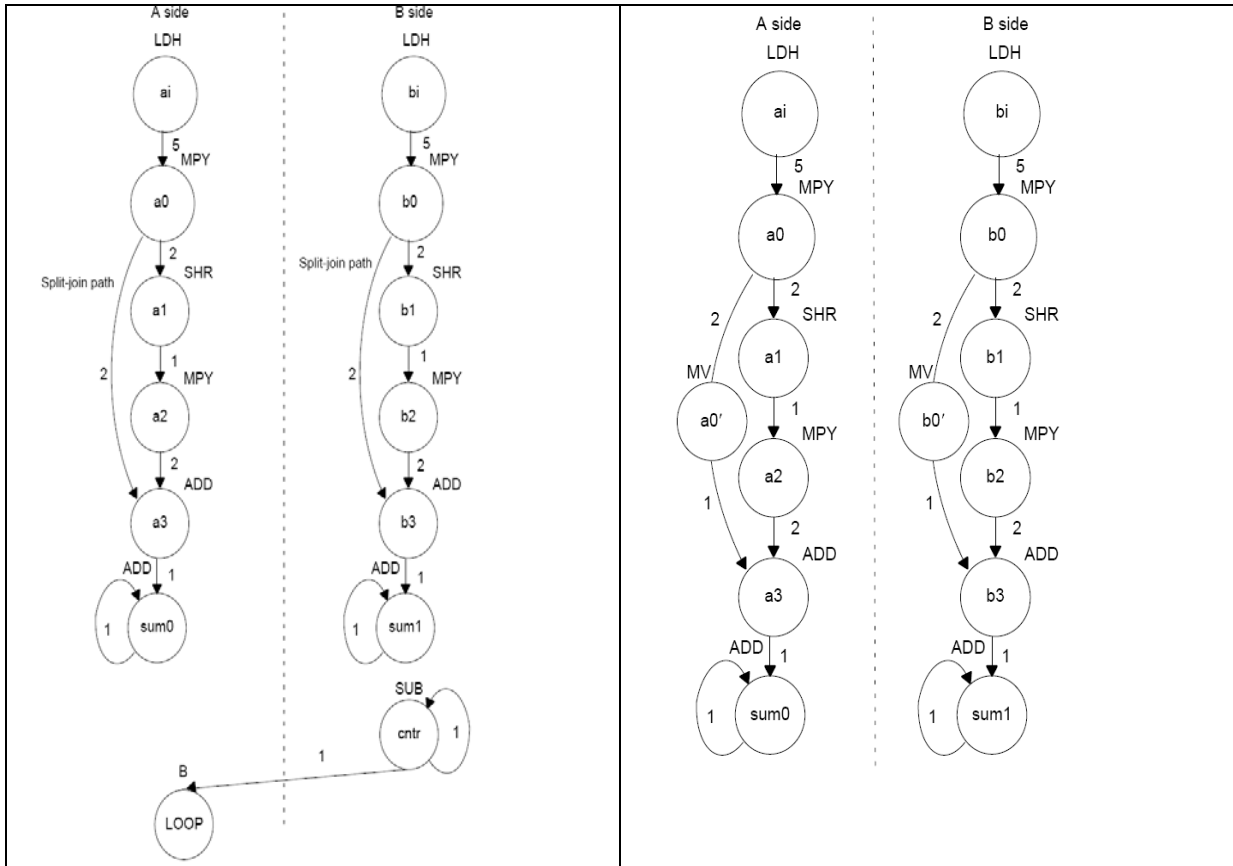
```

```
int live_long(short a[],short b[],short c, short d, short e)
{
    int i,sum0,sum1,sum,a0,a2,a3,b0,b2,b3;
    short a1,b1;

    sum0 = 0;
    sum1 = 0;
    for(i=0; i<100; i++){
        a0 = a[i] * c;
        a1 = a0 >> 15;
        a2 = a1 * d;
        a3 = a2 + a0;
        sum0 += a3;
        b0 = b[i] * c;
        b1 = b0 >> 15;
        b2 = b1 * e;
        b3 = b2 + b0;
        sum1 += b3;
    }
    sum = sum0 + sum1;
    return(sum);
}
```

```
LDH    *aptr++,ai    ; load ai from memory
LDH    *bptr++,bi    ; load bi from memory
MPY    ai,c,a0       ; a0 = ai * c
SHR    a0,15,a1      ; a1 = a0 >> 15
MPY    a1,d,a2       ; a2 = a1 * d
ADD    a2,a0,a3      ; a3 = a2 + a0
ADD    sum0,a3,sum0  ; sum0 += a3
MPY    bi,c,b0       ; b0 = bi * c
SHR    b0,15,b1      ; b1 = b0 >> 15
MPY    b1,e,b2       ; b2 = b1 * e
ADD    b2,b0,b3      ; b3 = b2 + b0
ADD    sum1,b3,sum1  ; sum1 += b3

[ctr]SUB    ctr,1,ctr    ; decrement loop counter
[ctr]B     LOOP         ; branch to loop
```



(a) A side

(b) B side

Unit(s)	Instructions	Total/Unit	Unit(s)	Instructions	Total/Unit
.M1	MPY	1	.M2	MPY	1
.S1	B and SHR	2	.S2	SHR	1
.D1	LDH	1	.D2	LDH	1
.L1, .S1, or .D1	2 ADDs	2	.L2, .S2, or .D2	2 ADDs and SUB	3
Total non-.M units		5	Total non-.M units		5

```

.global _live_long
_live_long: .cproc a, b, c, d, e

.reg ai, bi, sum0, sum1, sum
.reg a0p, a_0, a_1, a_2, a_3, b_0, b0p, b_1, b_2, b_3, cntr

MVK 100,cntr ; cntr = 100
ZERO sum0 ; sum0 = 0
ZERO sum1 ; sum1 = 0

LOOP: .trip 100
LDH .D1 *a++,ai ; load ai from memory
LDH .D2 *b++,bi ; load bi from memory
MPY .M1 ai,c,a_0 ; a0 = ai * c
SHR .S1 a_0,15,a_1 ; a1 = a0 >> 15
MPY .M1X a_1,d,a_2 ; a2 = a1 * d
MV .D1 a_0,a0p ; save a0 across iterations
ADD .L1 a_2,a0p,a_3 ; a3 = a2 + a0
ADD .L1 sum0,a_3,sum0 ; sum0 += a3
MPY .M2X bi,c,b_0 ; b0 = bi * ci
SHR .S2 b_0,15,b_1 ; b1 = b0 >> 15
MPY .M2X b_1,e,b_2 ; b2 = b1 * e
MV .D2 b_0,b0p ; save b0 across iterations
ADD .L2 b_2,b0p,b_3 ; b3 = b2 + b0
ADD .L2 sum1,b_3,sum1 ; sum1 += b3

[cntr] SUB .S2 cntr,1,cntr ; decrement loop counter
[cntr] B .S1 LOOP ; branch to loop

ADD sum0,sum1,sum ; Add sumi and sumi+1 for ret value

.return sum

.endproc

```

```

    LDH .D1 *A4++,A0 ; load ai from memory
|| LDH .D2 *B4++,B0 ; load bi from memory

    MVK .S2 100,B2 ; set up loop counter

    LDH .D1 *A4++,A0 ;* load ai from memory
|| LDH .D2 *B4++,B0 ;* load bi from memory

    ZERO .S1 A1 ; zero out accumulator
|| ZERO .S2 B1 ; zero out accumulator

    LDH .D1 *A4++,A0 ;** load ai from memory
|| LDH .D2 *B4++,B0 ;** load bi from memory

[B2] SUB .S2 B2,1,B2 ; decrement loop counter

    MPY .M1 A0,A6,A3 ; a0 = ai * c
|| MPY .M2X B0,A6,B10 ; b0 = bi * c
|| LDH .D1 *A4++,A0 ;*** load ai from memory
|| LDH .D2 *B4++,B0 ;*** load bi from memory

[B2] SUB .S2 B2,1,B2 ; decrement loop counter
|| [B2] B .S1 LOOP ; branch to loop

    SHR .S1 A3,15,A5 ; a1 = a0 >> 15
|| SHR .S2 B10,15,B5 ; b1 = b0 >> 15
|| MPY .M1 A0,A6,A3 ;* a0 = ai * c
|| MPY .M2X B0,A6,B10 ;* b0 = bi * c
|| LDH .D1 *A4++,A0 ;**** load ai from memory
|| LDH .D2 *B4++,B0 ;**** load bi from memory

    MPY .M1X A5,B6,A7 ; a2 = a1 * d
|| MV .D1 A3,A2 ; save a0 across iterations
|| MPY .M2X B5,A8,B7 ; b2 = b1 * e
|| MV .D2 B10,B8 ; save b0 across iterations
|| [B2] SUB .S2 B2,1,B2 ;* decrement loop counter
|| [B2] B .S1 LOOP ;* branch to loop

    SHR .S1 A3,15,A5 ;* a1 = a0 >> 15
|| SHR .S2 B10,15,B5 ;* b1 = b0 >> 15
|| MPY .M1 A0,A6,A3 ;** a0 = ai * c
|| MPY .M2X B0,A6,B10 ;** b0 = bi * c
|| LDH .D1 *A4++,A0 ;***** load ai from memory
|| LDH .D2 *B4++,B0 ;***** load bi from memory

LOOP:
    ADD .L1 A7,A2,A9 ;* a3 = a2 + a0
|| ADD .L2 B7,B8,B9 ;* b3 = b2 + b0
|| MPY .M1X A5,B6,A7 ;* a2 = a1 * d
|| MV .D1 A3,A2 ;* save a0 across iterations
|| MPY .M2X B5,A8,B7 ;* b2 = b1 * e
|| MV .D2 B10,B8 ;* save b0 across iterations
|| [B2] SUB .S2 B2,1,B2 ;** decrement loop counter
|| [B2] B .S1 LOOP ;** branch to loop

    ADD .L1 A1,A9,A1 ; sum0 += a3
|| ADD .L2 B1,B9,B1 ; sum1 += b3
|| SHR .S1 A3,15,A5 ;** a1 = a0 >> 15
|| SHR .S2 B10,15,B5 ;** b1 = b0 >> 15
|| MPY .M1 A0,A6,A3 ;*** a0 = ai * c
|| MPY .M2X B0,A6,B10 ;*** b0 = bi * c
|| LDH .D1 *A4++,A0 ;***** load ai from memory
|| LDH .D2 *B4++,B0 ;***** load bi from memory
; Branch occurs here

    ADD .L1X A1,B1,A4 ; sum = sum0 + sum1

```

.L Unit	.M Unit	.S Unit		.D Unit	
ABS	MPY	ADD	SET	ADD	STB (15-bit offset)‡
ADD	MPYU	ADDK	SHL	ADDAB	STH (15-bit offset)‡
ADDU	MPYUS	ADD2	SHR	ADDAH	STW (15-bit offset)‡
AND	MPYSU	AND	SHRU	ADDAW	SUB
CMPEQ	MPYH	B disp	SSHL	LDB	SUBAB
CMPGT	MPYHU	B IRP†	SUB	LDBU	SUBAH
CMPGTU	MPYHUS	B NRP†	SUBU	LDH	SUBAW
CMPLT	MPYHSU	B reg	SUB2	LDHU	ZERO
CMPLTU	MPYHL	CLR	XOR	LDW	
LMBD	MPYHLU	EXT	ZERO	LDB (15-bit offset)‡	
MV	MPYHULS	EXTU		LDBU (15-bit offset)‡	
NEG	MPYHSLU	MV		LDH (15-bit offset)‡	
NORM	MPYLH	MVC†		LDHU (15-bit offset)‡	
NOT	MPYLHU	MVK		LDW (15-bit offset)‡	
OR	MPYLUHS	MVKH		MV	
SADD	MPYLSHU	MVKLH		STB	
SAT	SMPY	NEG		STH	
SSUB	SMPYHL	NOT		STW	
SUB	SMPYLH	OR			
SUBU	SMPYH				
SUBC					
XOR					
ZERO					

† S2 only
‡ D2 only

Instructions entières

Instruction Type	Delay Slots
NOP (no operation)	0
Store	0
Single cycle	0
Multiply (16 × 16)	1
Load	4
Branch	5

.L Unit	.M Unit	.S Unit	.D Unit
ADDDP	MPYDP	ABSDP	ADDAD
ADDSP	MPYI	ABSSP	LDDW
DPINT	MPYID	CMPEQDP	
DPSP	MPYSP	CMPEQSP	
DPTRUNC		CMPGTDP	
INTDP		CMPGTSP	
INTDPU		CMPLTDP	
INTSP		CMPLTSP	
INTSPU		RCPDP	
SPINT		RCPSP	
SPTRUNC		RSQRDP	
SUBDP		RSQRSP	
SUBSP		SPDP	

Instructions flottantes

