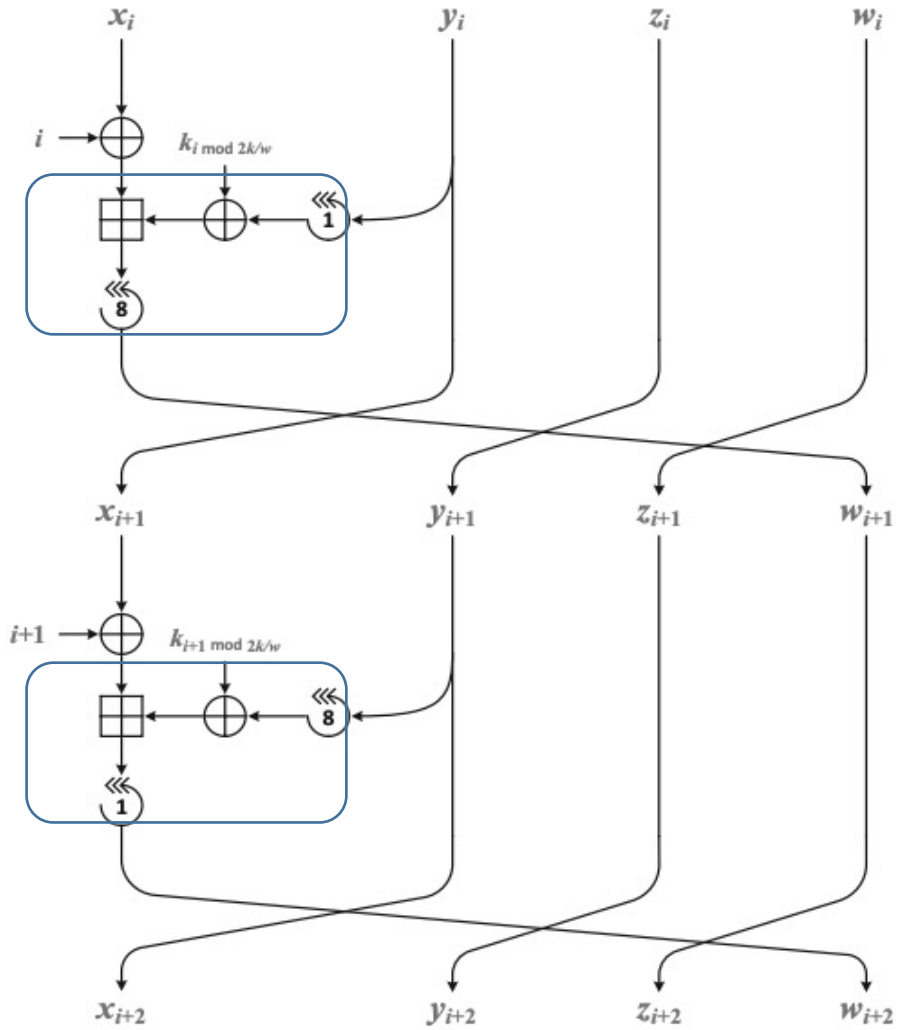
ARM64 상에서의 CHAM 병렬 구현

https://youtu.be/j5_r32-J-VM

CHAM

- ICISC'17에서 발표된 국산 경량 블록 암호
- ARX(Addition, Rotation, XOR) 연산
- Feistel 구조
  - 홀수 라운드에 ROL 연산(1, 8)
  - 짝수 라운드에 ROL 연산 (8, 1)

Cipher	n	k	w	r → r'(revised)
CHAM-64/128	64	128	16	80 → 88
CHAM-128/128	128	128	32	80 → 112
CHAM-128/256	128	256	32	96 → 120



CHAM 구현

- 레지스터 내부 정렬

	0-15	16-31	32-47	48-63	64-79	80-95	96-111	112-127
X[0]	PT1[0]	PT1[1]	PT1[2]	PT1[3]				

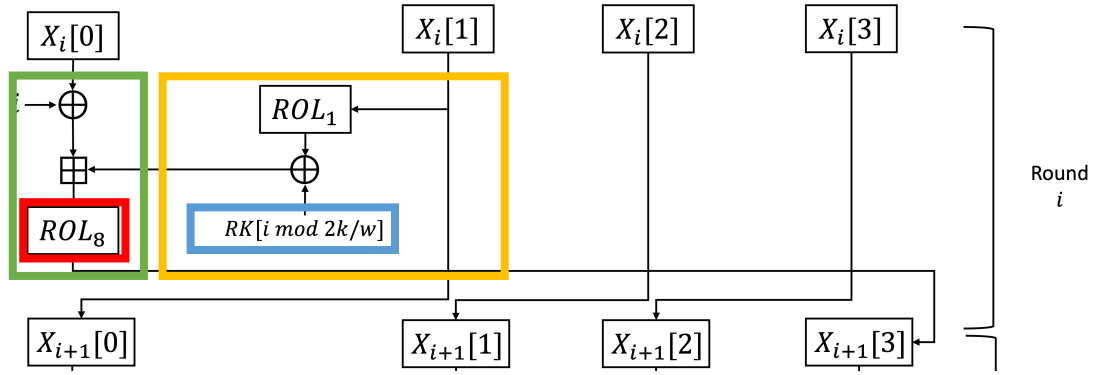
```
ld1.4h {v0},[x0] //pt
ld1.4h {v18,v19,v20,v21},[x1] //rk
```

	0-15	16-31	32-47	48-63	64-79	80-95	96-111	112-127
X[0]	PT1[0]							
X[1]	PT1[1]							
X[2]	PT1[2]							
X[3]	PT1[3]							

```
mov v1.h[0], v0.h[0] //pt1
mov v2.h[0], v0.h[1] //pt2
mov v3.h[0], v0.h[2] //pt3
mov v4.h[0], v0.h[3] //pt4
```

CHAM 구현

```
mov.4h v14, v2 //pt2

shl.4h v10, v14, #1
sri.4h v11, v14, #15
eor.8b v14, v10, v11

mov v9.h[0], v8.h[0] //rk[0]
eor.8b v14, v14, v9

eor.8b v1, v1, v6 //xor rc

add v1.4h, v1.4h, v14.4h

rev16 v1.8b, v1.8b //rotation left 8

mov v5.h[0], v1.h[0]
mov v1.h[0], v2.h[0]
mov v2.h[0], v3.h[0]
mov v3.h[0], v4.h[0]
mov v4.h[0], v5.h[0]

add.4h v6, v6, v20 //rc++
```



D6.137 REV16 (vector)

Reverse elements in 16-bit halfwords (vector).

Syntax

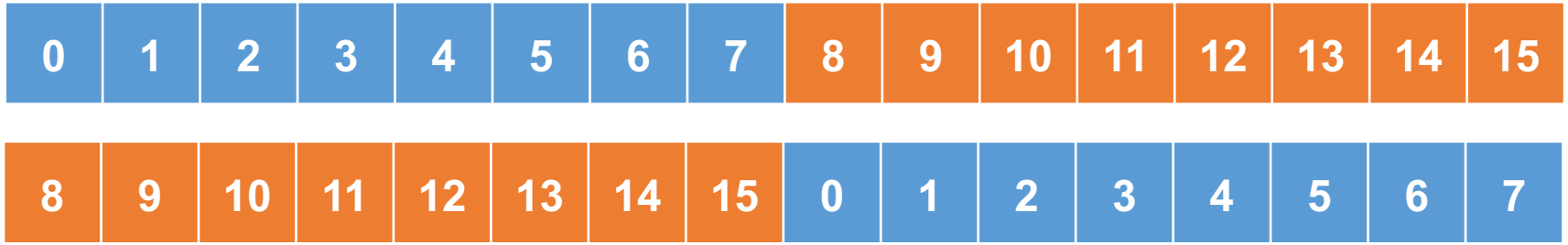
`REV16 Vd.T, Vn.T`

Where:

Vd
Is the name of the SIMD and FP destination register.

T
Is an arrangement specifier, and can be either `8B` or `16B`.

Vn
Is the name of the SIMD and FP source register.



CHAM 병렬 구현

- 레지스터 내부 정렬

	0-15	16-31	32-47	48-63	64-79	80-95	96-111	112-127
X[0]	PT1[0]	PT1[1]	PT1[2]	PT1[3]	PT2[0]	PT2[1]	PT2[2]	PT2[3]
X[1]	PT3[0]	PT3[1]	PT3[2]	PT3[3]	PT4[0]	PT4[1]	PT4[2]	PT4[3]
X[2]	PT5[0]	PT5[1]	PT5[2]	PT5[3]	PT6[0]	PT6[1]	PT6[2]	PT6[3]
X[3]	PT7[0]	PT7[1]	PT7[2]	PT7[3]	PT8[0]	PT8[1]	PT8[2]	PT8[3]

```
ld1.8h {v0,v1,v2,v3},[x0] //pt
ld1R.8h {v18},[x1], #2    //rk
```

Table D6-22 LD1R (Immediate offset) specifier combinations

T	imm
8B	#1
16B	#1
4H	#2
8H	#2
2S	#4
4S	#4
1D	#8
2D	#8

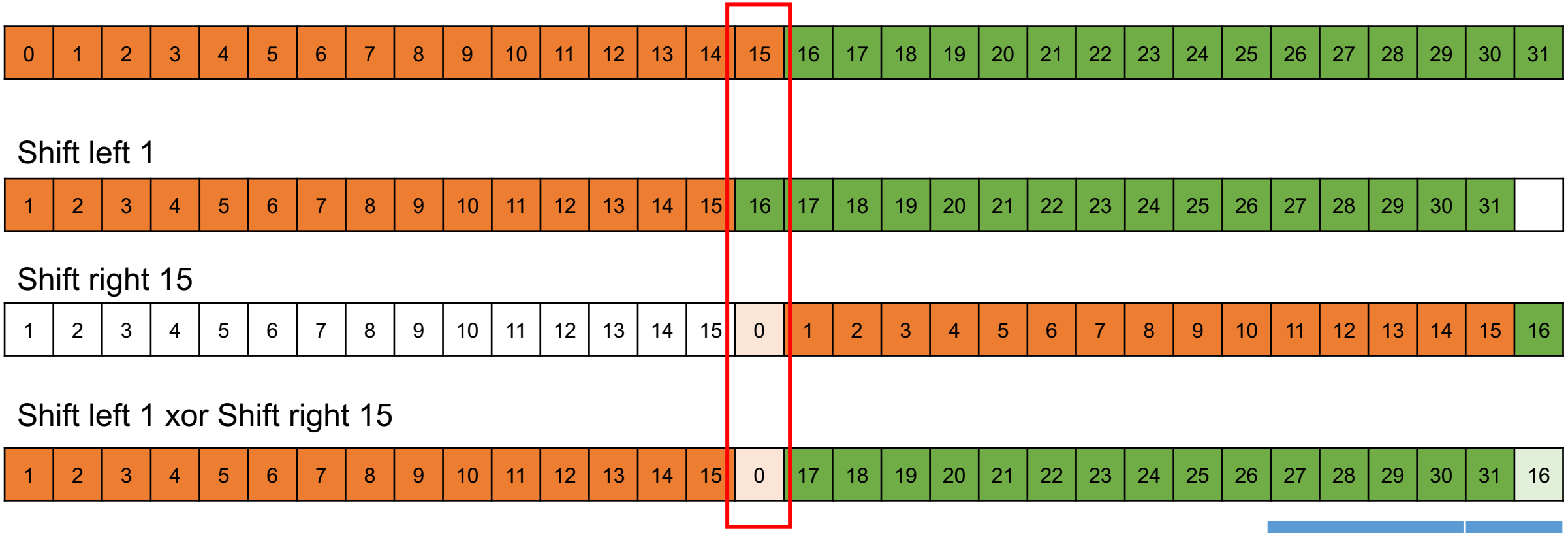
	0-15	16-31	32-47	48-63	64-79	80-95	96-111	112-127
X[0]	PT1[0]	PT2[0]	PT3[0]	PT4[0]	PT5[0]	PT6[0]	PT7[0]	PT8[0]
X[1]	PT1[1]	PT2[1]	PT3[1]	PT4[1]	PT5[1]	PT6[1]	PT7[1]	PT8[1]
X[2]	PT1[2]	PT2[2]	PT3[2]	PT4[2]	PT5[2]	PT6[2]	PT7[2]	PT8[2]
X[3]	PT1[3]	PT2[3]	PT3[3]	PT4[3]	PT5[3]	PT6[3]	PT7[3]	PT8[3]

```
mov v11.h[0], v0.h[0] //pt1_1
mov v12.h[0], v0.h[1] //pt2_1
mov v13.h[0], v0.h[2] //pt3_1
mov v14.h[0], v0.h[3] //pt4_1

mov v11.h[1], v0.h[4] //pt1_2
mov v12.h[1], v0.h[5] //pt2_2
mov v13.h[1], v0.h[6] //pt3_2
mov v14.h[1], v0.h[7] //pt4_2
```

	0-15	16-31	32-47	48-63	64-79	80-95	96-111	112-127
RK	RK[0]	RK[0]	RK[0]	RK[0]	RK[0]	RK[0]	RK[0]	RK[0]

CHAM 병렬 구현

- Rotation Left 1




```
movi.8h v4, 0x01
movi.16b v5, 0xfe

0xfffefffe
0x00010001
```

```
shl.8h      v20, v24, #1
and         v20.16b, v20.16b, v5.16b
sri.8h      v21, v24, #15
and         v21.16b, v21.16b, v4.16b
eor.16b     v24, v20, v21
```

MOVI 명령어를 사용하여 지정할 수 있는 상수 범위 : 0xFF

0001 0001 0001 0001 0001 0001 0001 0001

fefe fefe fefe fefe fefe fefe fefe fefe

AND		
0	0	0
0	1	0
1	0	0
1	1	1

CHAM 병렬 구현

- 16비트씩 병렬 덧셈 연산

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

```
//start add (8-pt)
    mov.8h      v16, v24
    and         v24.16b, v24.16b, v8.16b //v8 : 0x80

    mov.8h      v17, v11
    and         v11.16b, v11.16b, v8.16b

    eor.16b     v24, v24, v11

    and         v16.16b, v16.16b, v9.16b //v9 : 0x7f
    and         v17.16b, v17.16b, v9.16b

    add         v11.8h, v16.8h, v17.8h
    eor.16b     v11, v11, v24

// end add 1-pt : //add v11.8h, v11.8h, v24.8h
```

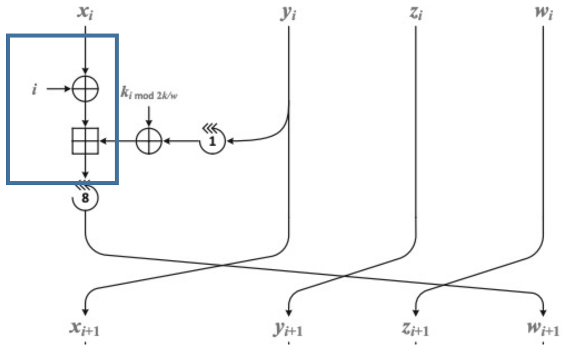
MOVI 명령어를 사용하여 지정할 수 있는 상수 범위 : 0xFF

8080 8080 8080 8080 8080 8080 8080 8080

7f7f 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f 7f7f

0x8000
0xffff7fff

AND		
0	0	0
0	1	0
1	0	0
1	1	1



향후 계획

- 레지스터 내부 정렬

	0-15	16-31	32-47	48-63	64-79	80-95	96-111	112-127
X[0]	PT1[0]	PT1[1]	PT1[2]	PT1[3]	PT2[0]	PT2[1]	PT[2]	PT[3]

	0-15	16-31	32-47	48-63	64-79	80-95	96-111	112-127
X[0]	PT1[0]	PT1[2]	PT1[1]	PT1[3]	PT2[0]	PT2[2]	PT2[1]	PT2[3]

	0-15	16-31	32-47	48-63	64-79	80-95	96-111	112-127
X[0]	PT1[0]	PT1[2]						
X[1]	PT1[1]	PT1[3]						
X[2]	PT2[0]	PT2[2]						
X[3]	PT2[1]	PT2[3]						

Q & A