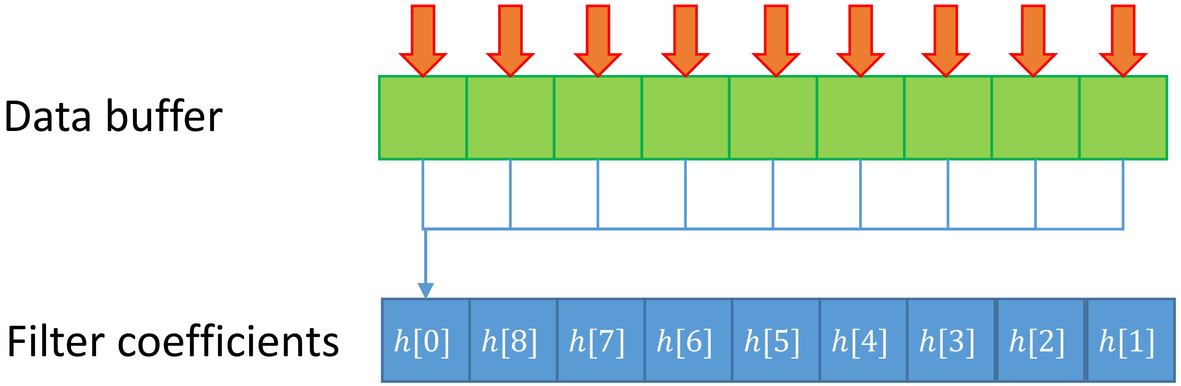
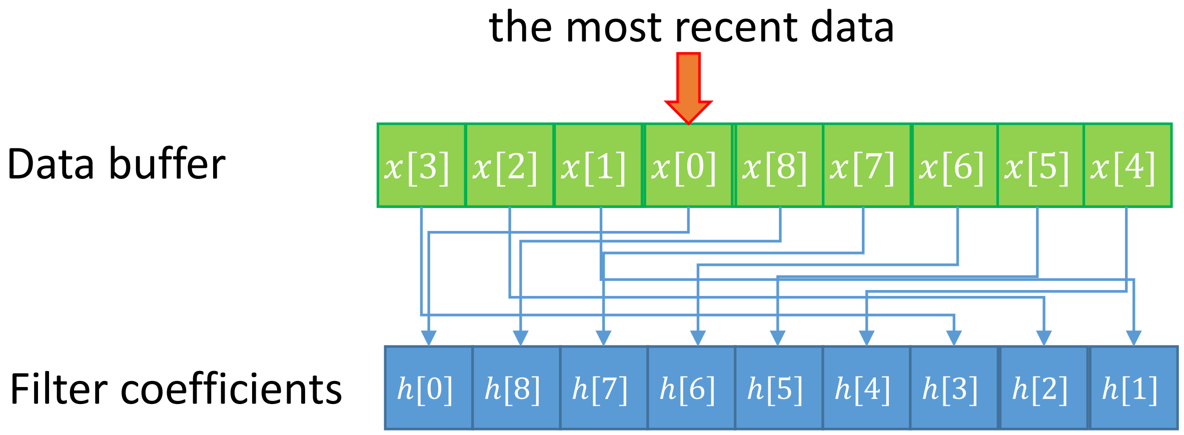
Circular Convolution

- We have two buffers to store the data and filter coefficients
  - Data buffer : updated every time it gets new data
  - Filter coefficients buffer : no change



Circular Convolution

- Convolution is doing SoP
- For the circular convolution, we have to find correct indexes for the data buffer and the coefficient buffer for every product



the most recent data

Data buffer

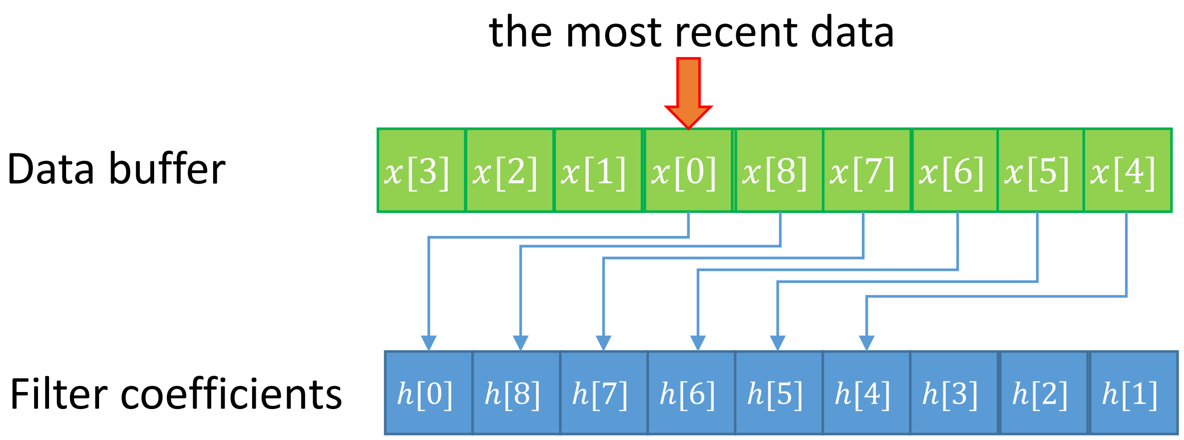
$x[3]$ $x[2]$ $x[1]$ $x[0]$ $x[8]$ $x[7]$ $x[6]$ $x[5]$ $x[4]$

Filter coefficients

$h[0]$ $h[8]$ $h[7]$ $h[6]$ $h[5]$ $h[4]$ $h[3]$ $h[2]$ $h[1]$

Circular Convolution

- Implementation using Assembly code
- SoP can be divided into two parts

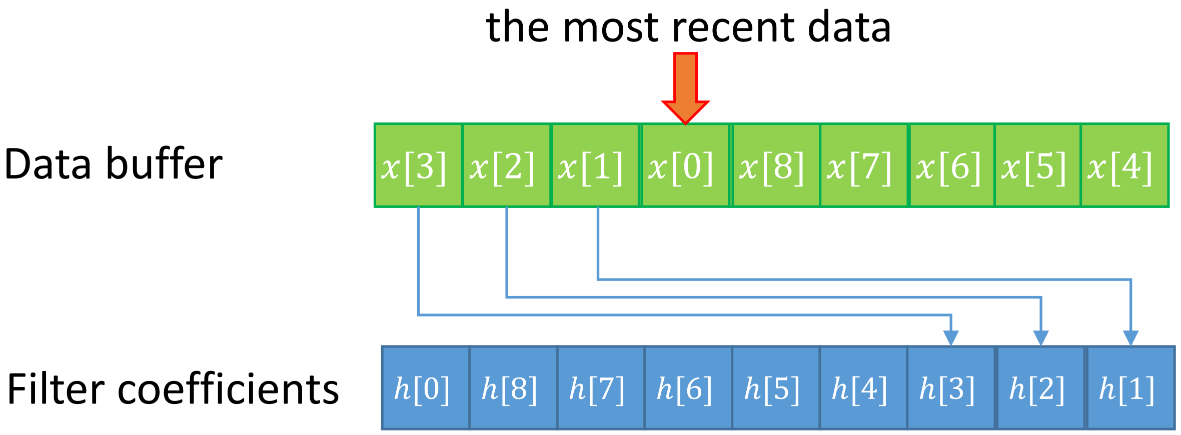


the most recent data
Data buffer
$x[3]$ $x[2]$ $x[1]$ $x[0]$ $x[8]$ $x[7]$ $x[6]$ $x[5]$ $x[4]$
Filter coefficients
$h[0]$ $h[8]$ $h[7]$ $h[6]$ $h[5]$ $h[4]$ $h[3]$ $h[2]$ $h[1]$

```
.global convolution_circular
/*
r4 - data (x)
r5 - coefficients (h)
r6 - N
r7 - index
*/
convolution_circular:
        add r2, r0, r0
        add r12, r0, r7
        sub r11, r6, r7
        beq r12, r0, CONV
        IND:
                addi r4, r4, 4
                subi r7, r7, 1
                bgt r7, r0, IND
        CONV:
                ldw r9, 0(r4)
                ldw r10, 0(r5)
                mul r8, r9, r10
                add r2, r2, r8
                addi r4, r4, 4
                addi r5, r5, 4
                subi r11, r11, 1
                bgt r11, r0, CONV
                beq r12, r0, END
                subi r6, r6, 1
                subi r4, r4, 4
        IND2:
                subi r4, r4, 4
                subi r6, r6, 1
                bgt r6, r0, IND2
        CONV2:
                ldw r9, 0(r4)
                ldw r10, 0(r5)
                mul r8, r9, r10
                add r2, r2, r8
                addi r4, r4, 4
                addi r5, r5, 4
                subi r12, r12, 1
                bgt r12, r0, CONV2
        END:
        ret
```

Circular Convolution

- Implementation using Assembly code
- SoP can be divided into two parts



the most recent data

Data buffer

x[3]	x[2]	x[1]	x[0]	x[8]	x[7]	x[6]	x[5]	x[4]

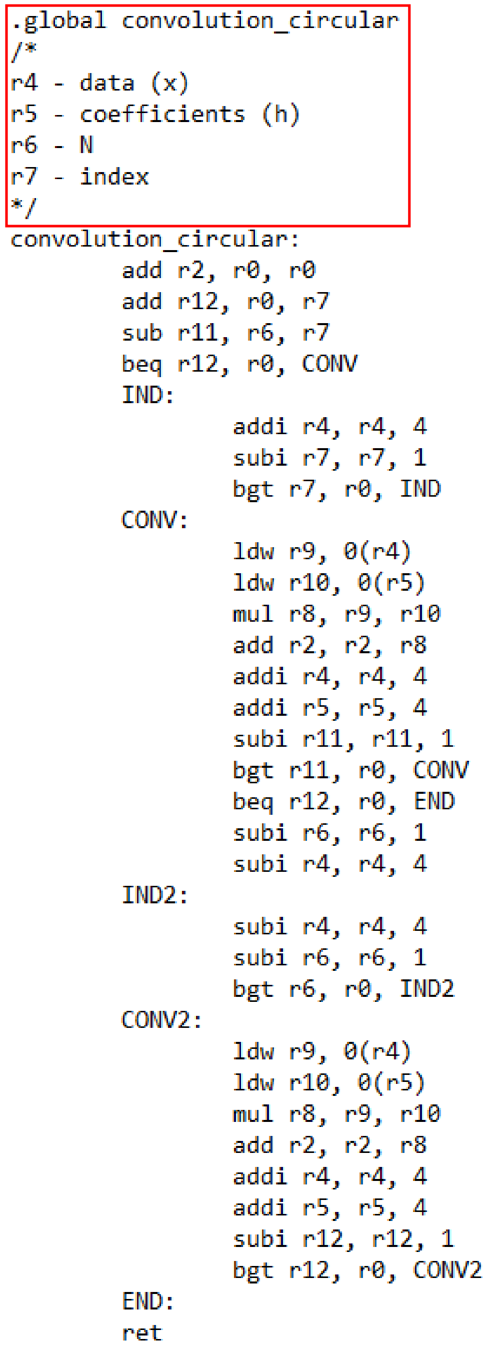
Filter coefficients

h[0]	h[8]	h[7]	h[6]	h[5]	h[4]	h[3]	h[2]	h[1]

```
.global convolution_circular
/*
r4 - data (x)
r5 - coefficients (h)
r6 - N
r7 - index
*/
convolution_circular:
        add r2, r0, r0
        add r12, r0, r7
        sub r11, r6, r7
        beq r12, r0, CONV
        IND:
                addi r4, r4, 4
                subi r7, r7, 1
                bgt r7, r0, IND
        CONV:
                ldw r9, 0(r4)
                ldw r10, 0(r5)
                mul r8, r9, r10
                add r2, r2, r8
                addi r4, r4, 4
                addi r5, r5, 4
                subi r11, r11, 1
                bgt r11, r0, CONV
                beq r12, r0, END
                subi r6, r6, 1
                subi r4, r4, 4
        IND2:
                subi r4, r4, 4
                subi r6, r6, 1
                bgt r6, r0, IND2
        CONV2:
                ldw r9, 0(r4)
                ldw r10, 0(r5)
                mul r8, r9, r10
                add r2, r2, r8
                addi r4, r4, 4
                addi r5, r5, 4
                subi r12, r12, 1
                bgt r12, r0, CONV2
        END:
        ret
```

Circular Convolution

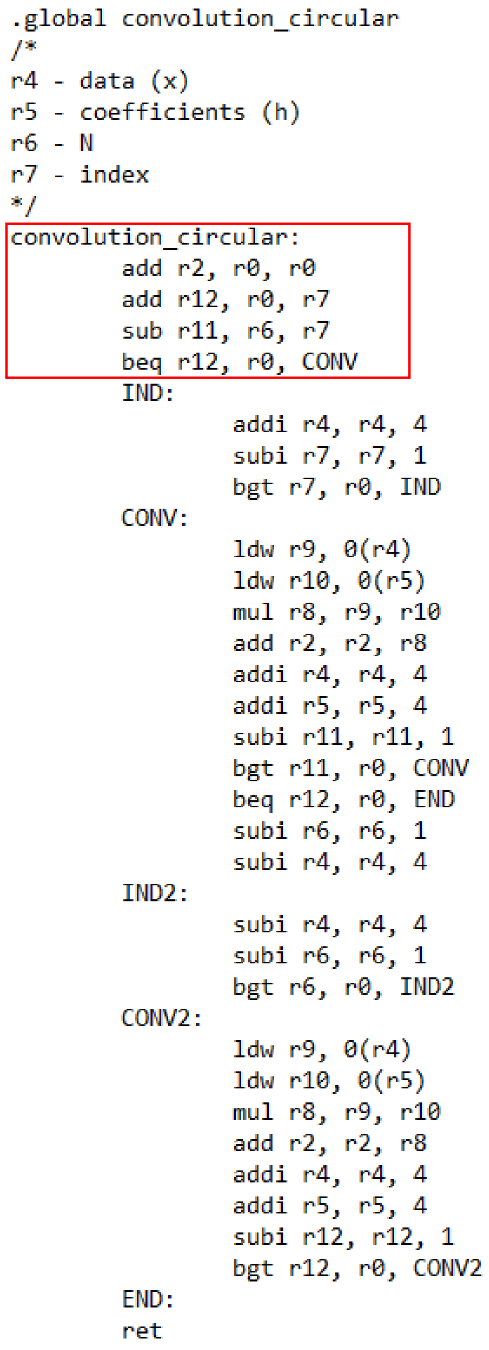
- Function : convolution_circular(x, h, N, index);
- r4 – data(x) : data buffer defined in the C code
- r5 – coefficients (h) : filter coefficients
- r6 – N : the size of the buffer
- r7- index : index of the most recent data



```
.global convolution_circular
/*
r4 - data (x)
r5 - coefficients (h)
r6 - N
r7 - index
*/
convolution_circular:
        add r2, r0, r0
        add r12, r0, r7
        sub r11, r6, r7
        beq r12, r0, CONV
        IND:
                addi r4, r4, 4
                subi r7, r7, 1
                bgt r7, r0, IND
        CONV:
                ldw r9, 0(r4)
                ldw r10, 0(r5)
                mul r8, r9, r10
                add r2, r2, r8
                addi r4, r4, 4
                addi r5, r5, 4
                subi r11, r11, 1
                bgt r11, r0, CONV
                beq r12, r0, END
                subi r6, r6, 1
                subi r4, r4, 4
        IND2:
                subi r4, r4, 4
                subi r6, r6, 1
                bgt r6, r0, IND2
        CONV2:
                ldw r9, 0(r4)
                ldw r10, 0(r5)
                mul r8, r9, r10
                add r2, r2, r8
                addi r4, r4, 4
                addi r5, r5, 4
                subi r12, r12, 1
                bgt r12, r0, CONV2
        END:
        ret
```

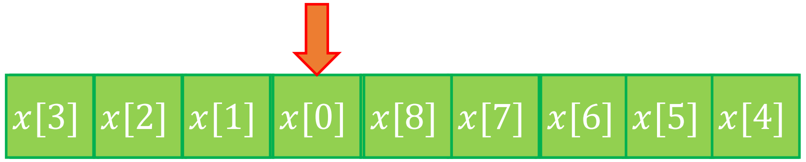
Circular Convolution

- Function name : convolution_circular
- add r2, r0, r0 : set r2(return value) as 0
- add r12, r0, r7 : set r12 as index
- sub r11, r6, r7 : set r11 as N-index
- beq r12, r0, CONV
  - If index==0, goto CONV
  - else goto IND



```
.global convolution_circular
/*
r4 - data (x)
r5 - coefficients (h)
r6 - N
r7 - index
*/
convolution_circular:
        add r2, r0, r0
        add r12, r0, r7
        sub r11, r6, r7
        beq r12, r0, CONV
        IND:
                addi r4, r4, 4
                subi r7, r7, 1
                bgt r7, r0, IND
        CONV:
                ldw r9, 0(r4)
                ldw r10, 0(r5)
                mul r8, r9, r10
                add r2, r2, r8
                addi r4, r4, 4
                addi r5, r5, 4
                subi r11, r11, 1
                bgt r11, r0, CONV
                beq r12, r0, END
                subi r6, r6, 1
                subi r4, r4, 4
        IND2:
                subi r4, r4, 4
                subi r6, r6, 1
                bgt r6, r0, IND2
        CONV2:
                ldw r9, 0(r4)
                ldw r10, 0(r5)
                mul r8, r9, r10
                add r2, r2, r8
                addi r4, r4, 4
                addi r5, r5, 4
                subi r12, r12, 1
                bgt r12, r0, CONV2
        END:
        ret
```

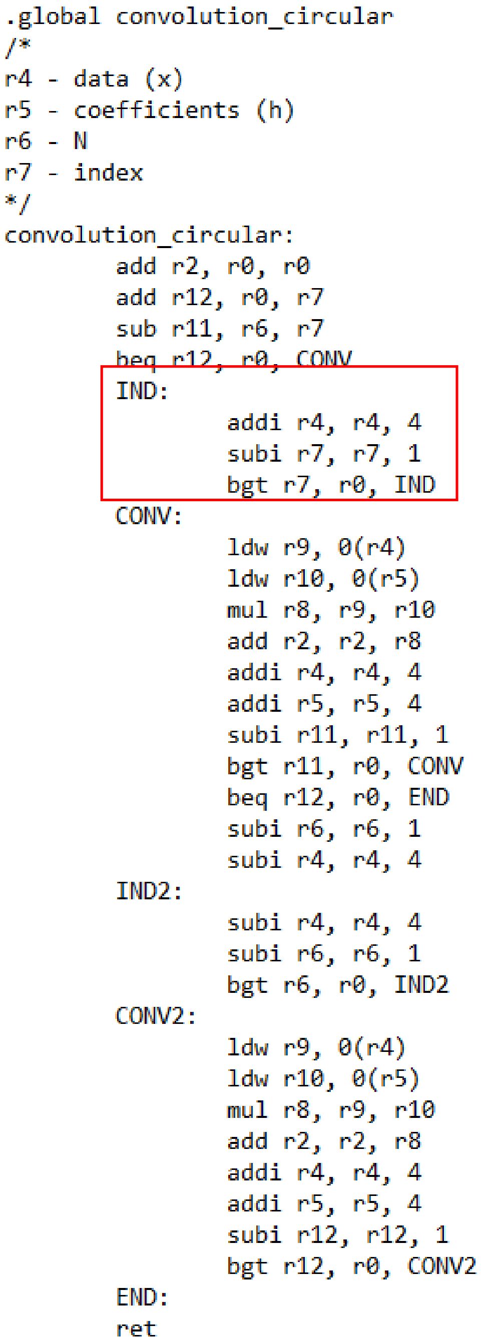
Data buffer



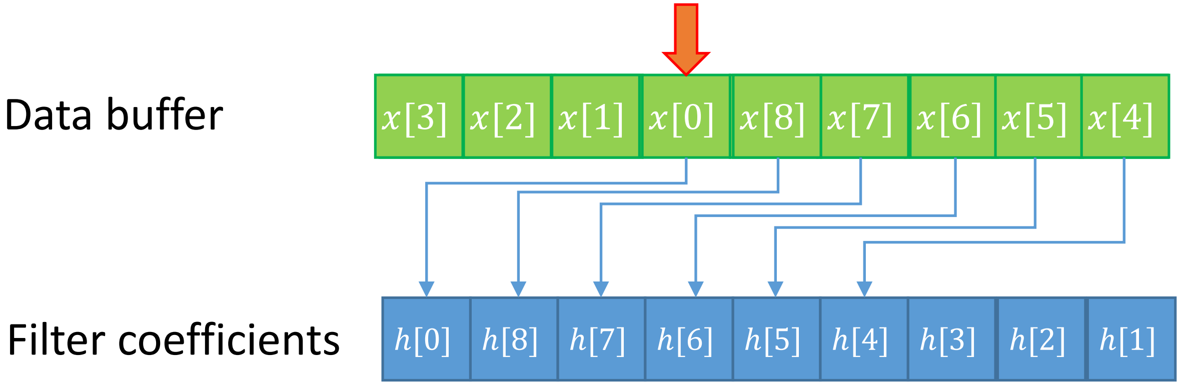
$x[3]$	$x[2]$	$x[1]$	$x[0]$	$x[8]$	$x[7]$	$x[6]$	$x[5]$	$x[4]$

- IND : find the starting point
- addi r4, r4, 4 : increase pointer of data buffer
- subi r7, r7, 1 : subtract 1 from the index
- bgt r7, r0, IND : branch back to IND r7 goes 0
- Meaning : move the pointer of the data buffer

  to the most recent data

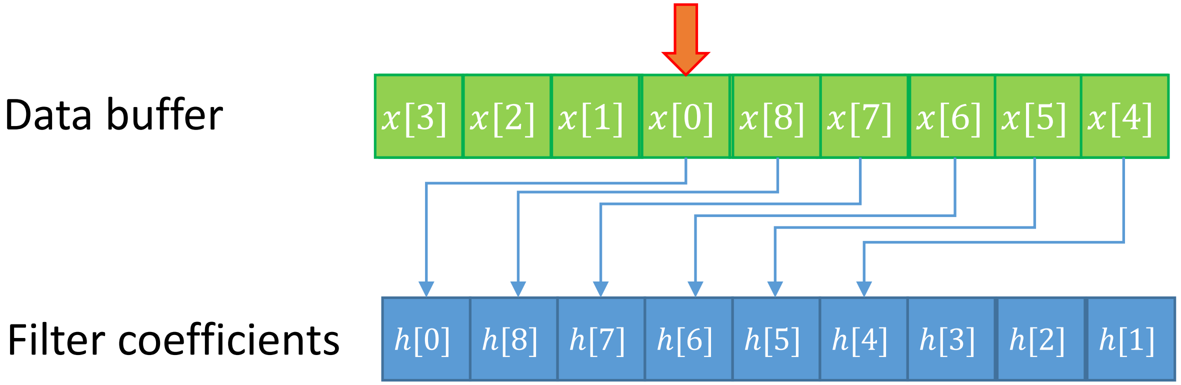


```
.global convolution_circular
/*
r4 - data (x)
r5 - coefficients (h)
r6 - N
r7 - index
*/
convolution_circular:
    add r2, r0, r0
    add r12, r0, r7
    sub r11, r6, r7
    beq r12, r0, CONV
    IND:
        addi r4, r4, 4
        subi r7, r7, 1
        bgt r7, r0, IND
    CONV:
        ldw r9, 0(r4)
        ldw r10, 0(r5)
        mul r8, r9, r10
        add r2, r2, r8
        addi r4, r4, 4
        addi r5, r5, 4
        subi r11, r11, 1
        bgt r11, r0, CONV
        beq r12, r0, END
        subi r6, r6, 1
        subi r4, r4, 4
    IND2:
        subi r4, r4, 4
        subi r6, r6, 1
        bgt r6, r0, IND2
    CONV2:
        ldw r9, 0(r4)
        ldw r10, 0(r5)
        mul r8, r9, r10
        add r2, r2, r8
        addi r4, r4, 4
        addi r5, r5, 4
        subi r12, r12, 1
        bgt r12, r0, CONV2
    END:
    ret
```


Data buffer
$x[3]$ $x[2]$ $x[1]$ $x[0]$ $x[8]$ $x[7]$ $x[6]$ $x[5]$ $x[4]$
Filter coefficients
$h[0]$ $h[8]$ $h[7]$ $h[6]$ $h[5]$ $h[4]$ $h[3]$ $h[2]$ $h[1]$

- CONV : convolution of the first part
- ldw r9, 0(r4) : load the first data to r9
- ldw r10, 0(r5) : load the first coeff to r10
- mul r8, r9, r10 : multiply the data and coeff
- add r2, r2, r8 : add r8 to the return value
- addi r4, r4, 4 : increase the index of data buffer
- addi r5, r5, 4 : increase the index of coeff buffer
- Meaning : do the SoP

```
.global convolution_circular
/*
r4 - data (x)
r5 - coefficients (h)
r6 - N
r7 - index
*/
convolution_circular:
       add r2, r0, r0
       add r12, r0, r7
       sub r11, r6, r7
       beq r12, r0, CONV
       IND:
              addi r4, r4, 4
              subi r7, r7, 1
              bgt r7, r0, IND
       CONV:
              ldw r9, 0(r4)
              ldw r10, 0(r5)
              mul r8, r9, r10
              add r2, r2, r8
              addi r4, r4, 4
              addi r5, r5, 4
              subi r11, r11, 1
              bgt r11, r0, CONV
              beq r12, r0, END
              subi r6, r6, 1
              subi r4, r4, 4
       IND2:
              subi r4, r4, 4
              subi r6, r6, 1
              bgt r6, r0, IND2
       CONV2:
              ldw r9, 0(r4)
              ldw r10, 0(r5)
              mul r8, r9, r10
              add r2, r2, r8
              addi r4, r4, 4
              addi r5, r5, 4
              subi r12, r12, 1
              bgt r12, r0, CONV2
       END:
       ret
```



Data buffer

$x[3]$ $x[2]$ $x[1]$ $x[0]$ $x[8]$ $x[7]$ $x[6]$ $x[5]$ $x[4]$

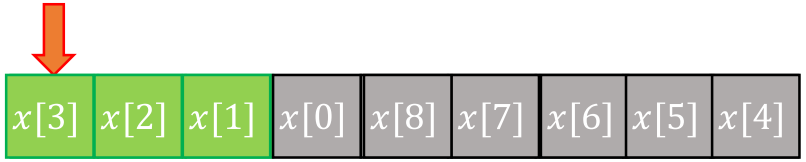
Filter coefficients

$h[0]$ $h[8]$ $h[7]$ $h[6]$ $h[5]$ $h[4]$ $h[3]$ $h[2]$ $h[1]$

- subi r11, r11, 1 : subtract 1 from the r11
- bgt r11, r0, CONV
  - If r11==0, go to next line
  - else, branch back to CONV
- beq r12, r0, END : if r12==0, go to END
  - Because it means index started from 0
- subi r6, r6, 1 : subtract 1 from r6
- subi r4, r4, 4 : decrease the index of data buffer
  - To protect the index points out NULL, since we increase the index after SoP
- Meaning : do the SoP until the index points out the last data of the data buffer

```
.global convolution_circular
/*
r4 - data (x)
r5 - coefficients (h)
r6 - N
r7 - index
*/
convolution_circular:
        add r2, r0, r0
        add r12, r0, r7
        sub r11, r6, r7
        beq r12, r0, CONV
        IND:
                addi r4, r4, 4
                subi r7, r7, 1
                bgt r7, r0, IND
        CONV:
                ldw r9, 0(r4)
                ldw r10, 0(r5)
                mul r8, r9, r10
                add r2, r2, r8
                addi r4, r4, 4
                addi r5, r5, 4
                subi r11, r11, 1
                bgt r11, r0, CONV
                beq r12, r0, END
                subi r6, r6, 1
                subi r4, r4, 4
        IND2:
                subi r4, r4, 4
                subi r6, r6, 1
                bgt r6, r0, IND2
        CONV2:
                ldw r9, 0(r4)
                ldw r10, 0(r5)
                mul r8, r9, r10
                add r2, r2, r8
                addi r4, r4, 4
                addi r5, r5, 4
                subi r12, r12, 1
                bgt r12, r0, CONV2
        END:
        ret
```

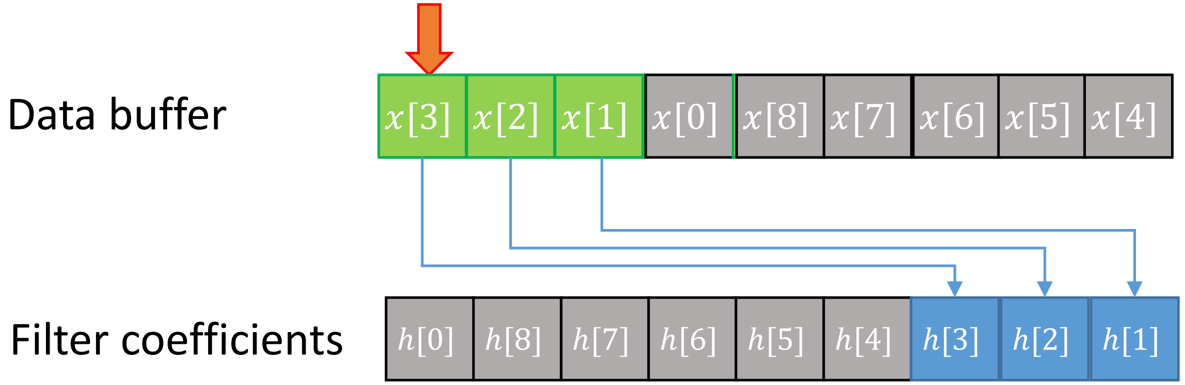
Data buffer



$x[3]$	$x[2]$	$x[1]$	$x[0]$	$x[8]$	$x[7]$	$x[6]$	$x[5]$	$x[4]$

- IND2 : index for second part
- subi r4, r4, 4 : decrease the index of data buffer
- subi r6, r6, 1: subtract 1 from r6
- Bgt r6, r0, IND2
  - If r6 == 0, go to CONV2
  - else, go to IND2
- Meaning : since the index is pointing the last data when we finish the first part, we have to move the pointer to the first data of the data buffer

```
.global convolution_circular
/*
r4 - data (x)
r5 - coefficients (h)
r6 - N
r7 - index
*/
convolution_circular:
        add r2, r0, r0
        add r12, r0, r7
        sub r11, r6, r7
        beq r12, r0, CONV
        IND:
                addi r4, r4, 4
                subi r7, r7, 1
                bgt r7, r0, IND
        CONV:
                ldw r9, 0(r4)
                ldw r10, 0(r5)
                mul r8, r9, r10
                add r2, r2, r8
                addi r4, r4, 4
                addi r5, r5, 4
                subi r11, r11, 1
                bgt r11, r0, CONV
                beq r12, r0, END
                subi r6, r6, 1
                subi r4, r4, 4
        IND2:
                subi r4, r4, 4
                subi r6, r6, 1
                bgt r6, r0, IND2
        CONV2:
                ldw r9, 0(r4)
                ldw r10, 0(r5)
                mul r8, r9, r10
                add r2, r2, r8
                addi r4, r4, 4
                addi r5, r5, 4
                subi r12, r12, 1
                bgt r12, r0, CONV2
        END:
        ret
```



Data buffer
$x[3]$ $x[2]$ $x[1]$ $x[0]$ $x[8]$ $x[7]$ $x[6]$ $x[5]$ $x[4]$
Filter coefficients
$h[0]$ $h[8]$ $h[7]$ $h[6]$ $h[5]$ $h[4]$ $h[3]$ $h[2]$ $h[1]$

- CONV2 : finish the SoP of second part
- ldw r9, 0(r4) : load the first data to r9
- ldw r10, 0(r5) : load the first coeff to r10
- mul r8, r9, r10 : multiply the data and coeff
- add r2, r2, r8 : add r8 to the return value
- addi r4, r4, 4 : increase the index of data buffer
- addi r5, r5, 4 : increase the index of coeff buffer
- subi r12, r12, 1 : subtract 1 from r12
- bgt r12, r0, CONV2
  - If r12 == 0, go to END
  - else, go to CONV2
- Meaning : do the SoP until finishing the second part

```
.global convolution_circular
/*
r4 - data (x)
r5 - coefficients (h)
r6 - N
r7 - index
*/
convolution_circular:
        add r2, r0, r0
        add r12, r0, r7
        sub r11, r6, r7
        beq r12, r0, CONV
        IND:
                addi r4, r4, 4
                subi r7, r7, 1
                bgt r7, r0, IND
        CONV:
                ldw r9, 0(r4)
                ldw r10, 0(r5)
                mul r8, r9, r10
                add r2, r2, r8
                addi r4, r4, 4
                addi r5, r5, 4
                subi r11, r11, 1
                bgt r11, r0, CONV
                beq r12, r0, END
                subi r6, r6, 1
                subi r4, r4, 4
        IND2:
                subi r4, r4, 4
                subi r6, r6, 1
                bgt r6, r0, IND2
        CONV2:
                ldw r9, 0(r4)
                ldw r10, 0(r5)
                mul r8, r9, r10
                add r2, r2, r8
                addi r4, r4, 4
                addi r5, r5, 4
                subi r12, r12, 1
                bgt r12, r0, CONV2
        END:
        ret
```