ICC DEVCON 2020

Calculator Element Programming

http://www.colorlab.no

https://www.appears-itn.eu/

THE FUTURE OF COLOR MANAGEMENT

By

Tanzima Habib

NTNU, Gjøvik, Norway

syedath@studntnu.no

OUTLINE

- Introduction
- Operations
- XML Representation
- Types of Operation Encodings
- Extended Structures in XML
- Example 1: Spectral Estimation using Calc elements
- Example 2: BRDF using Calc elements
- Limitations

Important Documents

- iccMAX Specification

http://color.org/specification/ICC.2-2019.pdf

- White paper 45: Calculator Element Programming

http://color.org/whitepapers/ICC_White_Paper45_Calculator_Programming-v3.pdf

- RefIccMAX - Win32 executables

http://color.org/iccmax/index.xalter

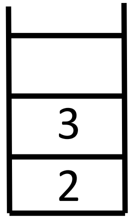
- Examples of Calculator Element Programming in iccMAX

http://www.color.org/DevCon/devcon2020/index.xalter
https://www.ingentaconnect.com/contentone/ist/lim/2020/00002020/00000001/art00028

Calculator Element Script Programming

A stack-based programming model

Eg:

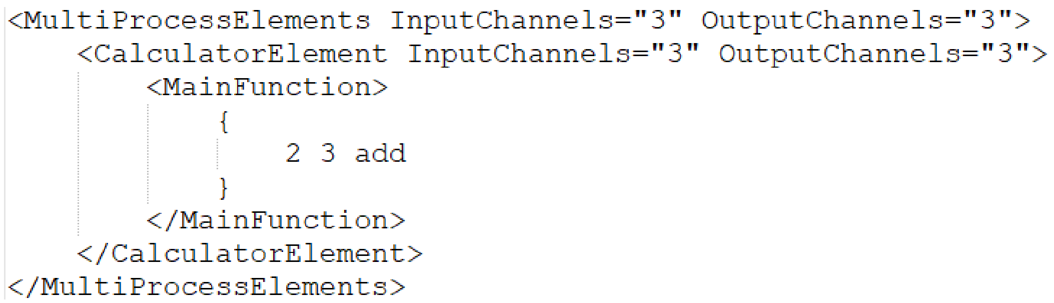


Postfix Notation

Eg: 2 3 add or
2 3 mul etc.

Inside multiprocessingElement tag

Eg:



```
<MultiProcessElements InputChannels="3" OutputChannels="3">
    <CalculatorElement InputChannels="3" OutputChannels="3">
        <MainFunction>
            {
                2 3 add
            }
        </MainFunction>
    </CalculatorElement>
</MultiProcessElements>
```

low-level scripting language

Eg: allows more than stack operations

32-bit data parameter

Eg: greater security and predictable behaviour

OPERATIONS

- take data off the stack
- directly perform some computational operation
- apply a processing sub-element
- get data from a CMM environment variable
- place data onto the stack
- get data from input channels
- store data to output channels
- store data to indexed memory
- retrieve data from indexed memory
- manipulate stack values
- conditionally select operations to perform

XML REPRESENTATION

```
<MultiProcessElements InputChannels="3" OutputChannels="3">
        <SubElements>… </SubElements>
        <CalculatorElement InputChannels="3" OutputChannels="3">
                <MainFunction>
                {
                        <!- Code using textual representation-->
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```

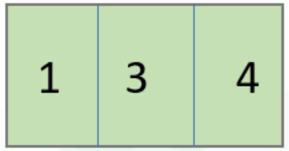
Calculator elements are encoded in binary structures. (Clause 11 of iccMAX Specification)
Textual representation defined in (Appendix F iccMAX Specification)

EXTENDED XML REPRESENTATION

```
<MultiProcessElements InputChannels="3" OutputChannels="3">
        <Imports> … </Imports>
        <Variables> … </Variables>
        <Macros> … </Macros>
        <SubElements>… </SubElements>
        <CalculatorElement InputChannels="3" OutputChannels="3">
                <MainFunction>
                {
                        <!- Code using textual representation-->
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```

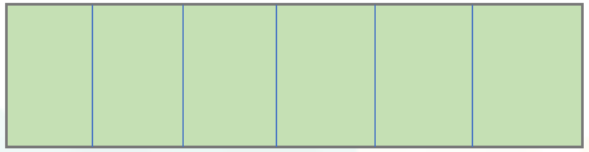
VISUAL REPRESENTATION

```
<MultiProcessElements InputChannels="3" OutputChannels="3">
        <CalculatorElement InputChannels="3" OutputChannels="3">
                <MainFunction>
                {
                        in(0,3)
                        tput(0)
                        2 2 mul(2)
                        tget(0)
                        out(0,3)

                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



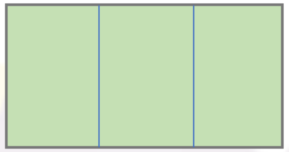
Input channel

`in(0,3)`



Temporary memory block

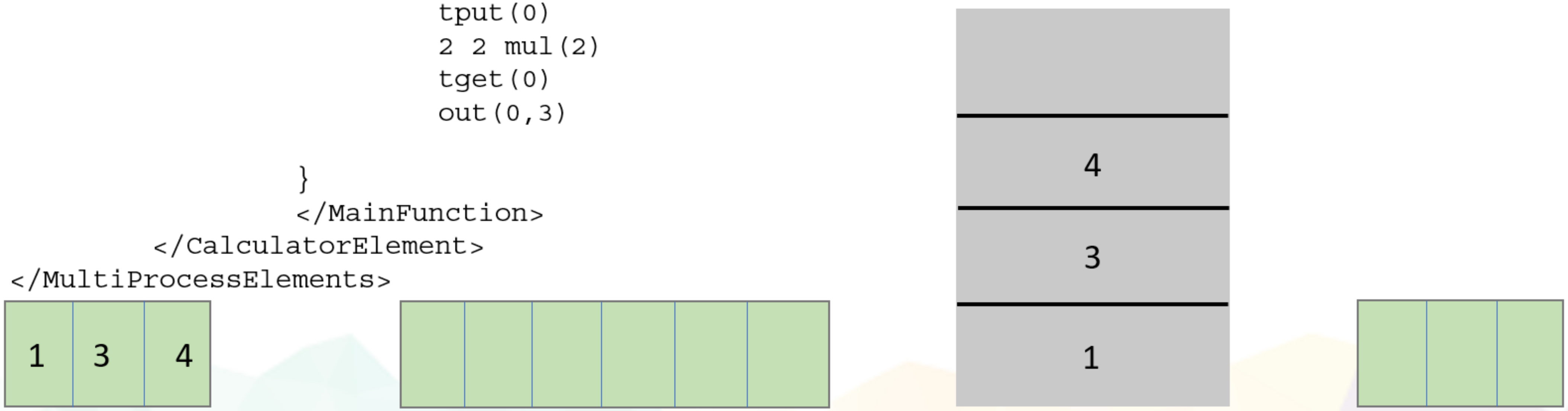
Execution stack



Output channel

VISUAL REPRESENTATION

```
<MultiProcessElements InputChannels="3" OutputChannels="3">
        <CalculatorElement InputChannels="3" OutputChannels="3">
                <MainFunction>
                {
                        in(0,3)
                        tput(0)
                        2 2 mul(2)
                        tget(0)
                        out(0,3)

                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



Input channel

in(0,3)

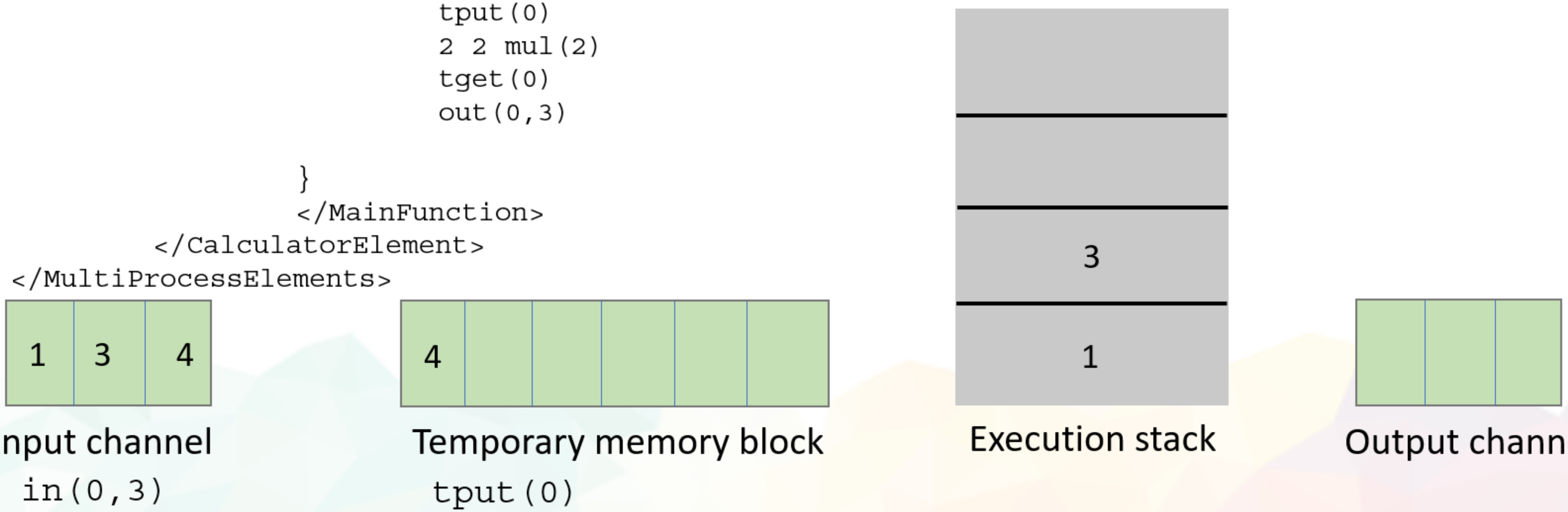
Temporary memory block

Execution stack

Output channel

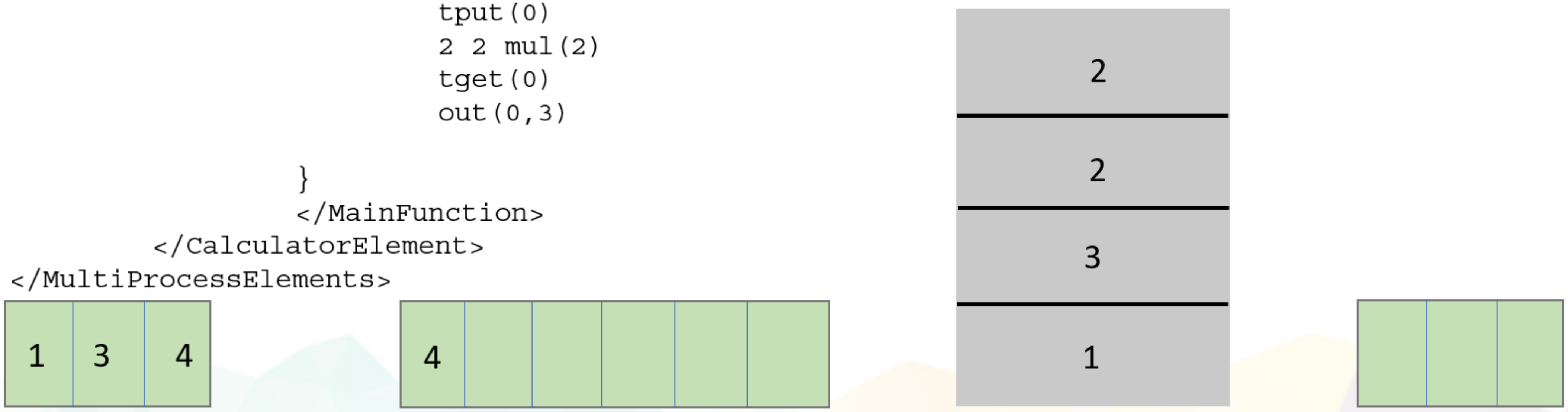
VISUAL REPRESENTATION

```
<MultiProcessElements InputChannels="3" OutputChannels="3">
        <CalculatorElement InputChannels="3" OutputChannels="3">
                <MainFunction>
                {
                        in(0,3)
                        tput(0)
                        2 2 mul(2)
                        tget(0)
                        out(0,3)

                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



VISUAL REPRESENTATION

```
<MultiProcessElements InputChannels="3" OutputChannels="3">
        <CalculatorElement InputChannels="3" OutputChannels="3">
                <MainFunction>
                {
                        in(0,3)
                        tput(0)
                        2 2 mul(2)
                        tget(0)
                        out(0,3)

                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



1	3	4

Input channel

`in(0,3)`

4					

Temporary memory block

`tput(0)`

2
2
3
1

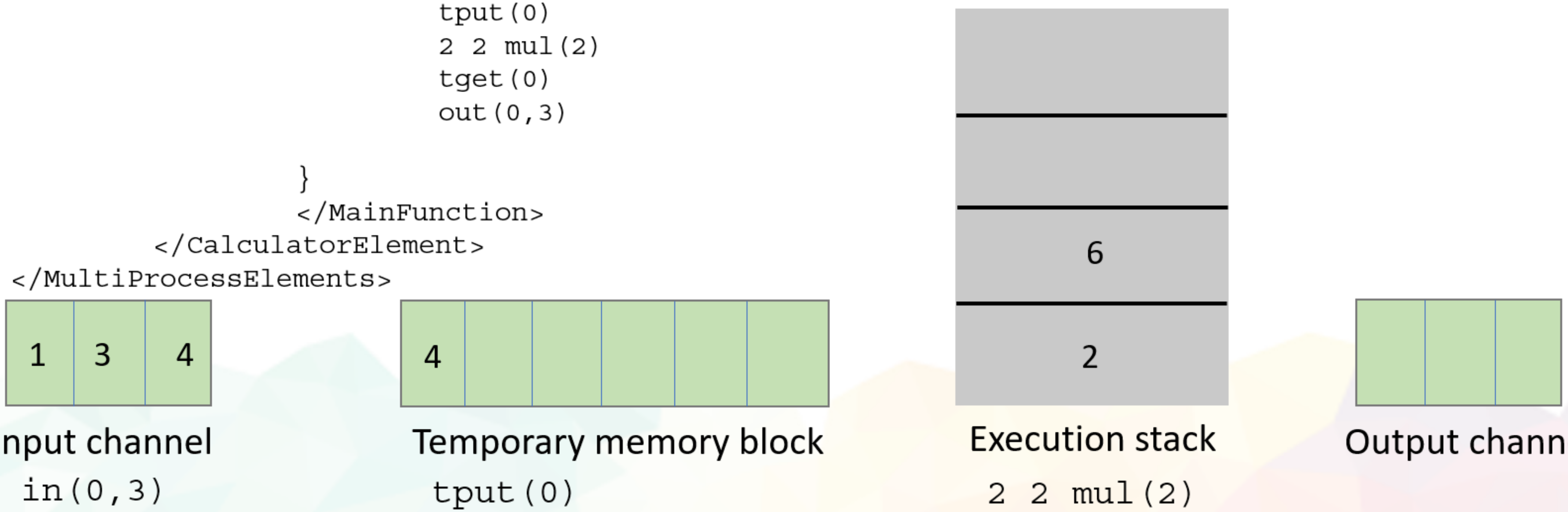
Execution stack

`2 2 mul(2)`

Output channel

VISUAL REPRESENTATION

```
<MultiProcessElements InputChannels="3" OutputChannels="3">
        <CalculatorElement InputChannels="3" OutputChannels="3">
                <MainFunction>
                {
                        in(0,3)
                        tput(0)
                        2 2 mul(2)
                        tget(0)
                        out(0,3)

                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



Input channel

1	3	4

`in(0,3)`

Temporary memory block

4					

`tput(0)`

Execution stack

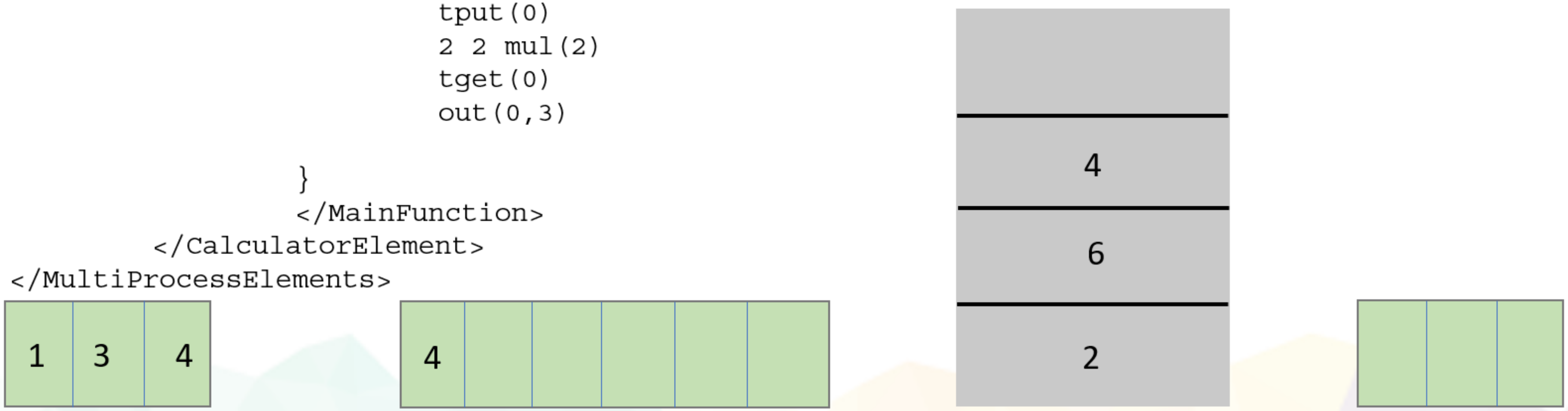
6
2

`2 2 mul(2)`

Output channel

VISUAL REPRESENTATION

```
<MultiProcessElements InputChannels="3" OutputChannels="3">
        <CalculatorElement InputChannels="3" OutputChannels="3">
                <MainFunction>
                {
                        in(0,3)
                        tput(0)
                        2 2 mul(2)
                        tget(0)
                        out(0,3)

                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



1	3	4

4					

4
6
2

Input channel

`in(0,3)`

Temporary memory block

`tput(0)`

`tget(0)`

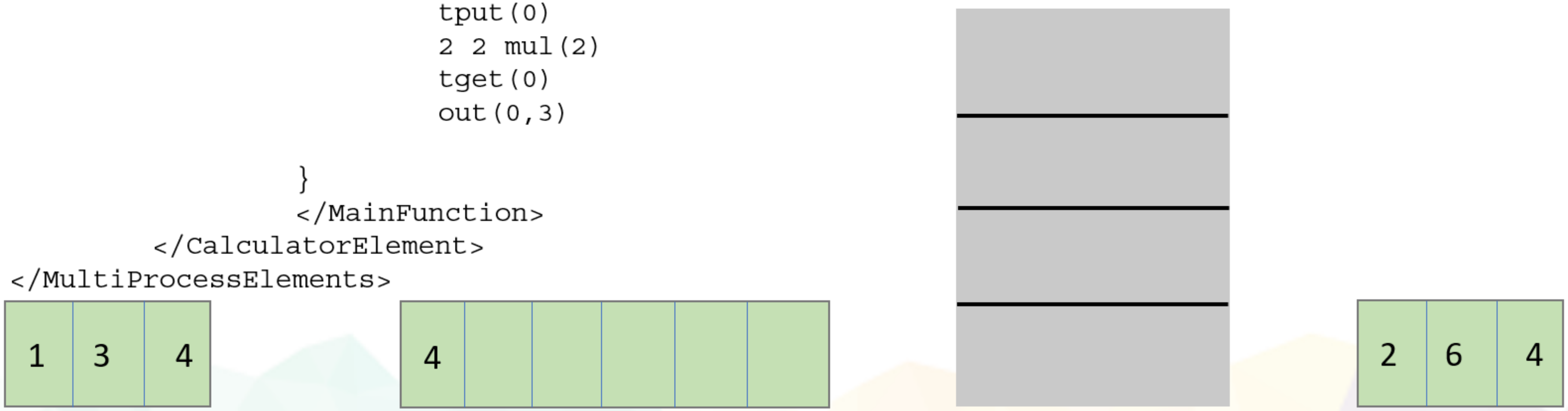
Execution stack

`2 2 mul(2)`

Output channel

VISUAL REPRESENTATION

```
<MultiProcessElements InputChannels="3" OutputChannels="3">
        <CalculatorElement InputChannels="3" OutputChannels="3">
                <MainFunction>
                {
                        in(0,3)
                        tput(0)
                        2 2 mul(2)
                        tget(0)
                        out(0,3)

                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



1	3	4

Input channel

`in(0,3)`

4					

Temporary memory block

`tput(0)`

`tget(0)`

Execution stack

`2 2 mul(2)`

2	6	4

Output channel

`out(0,3)`

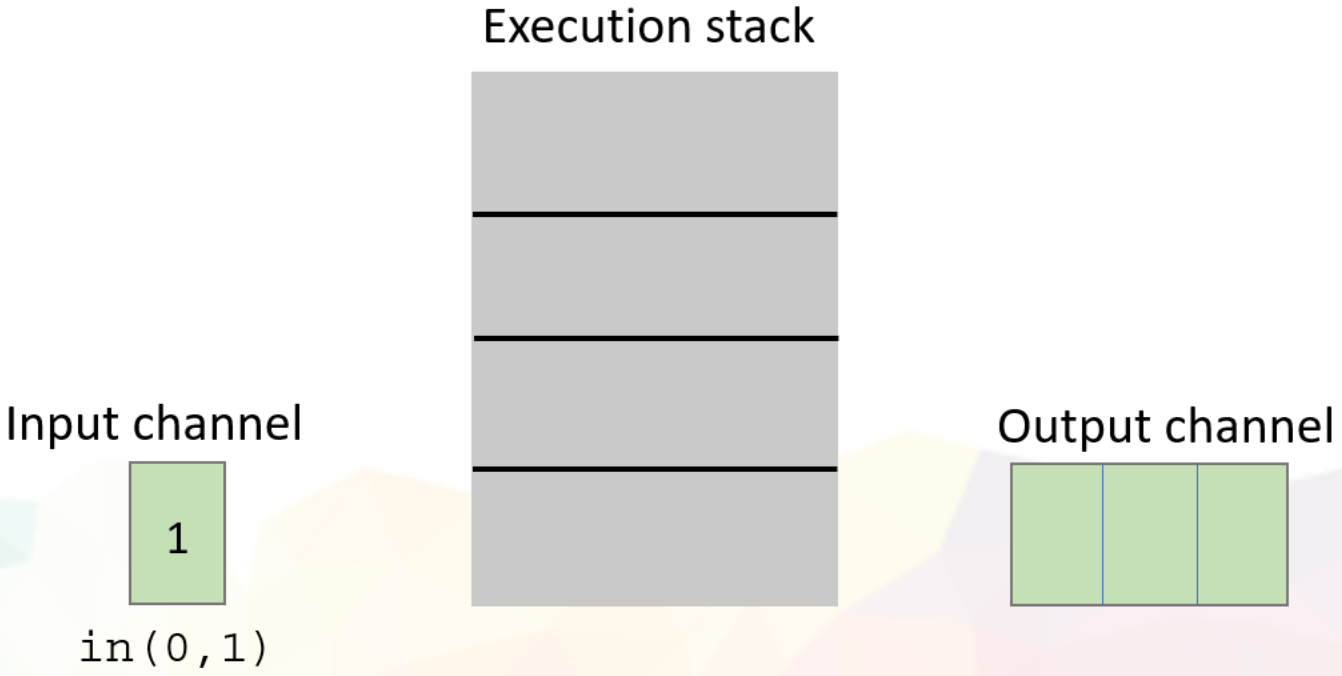
TYPES OF OPERATION ENCODINGS

1. Floating point constant operations
2. Channel vector operations
3. CMM environment variable operation
4. Sub-element invocation operations
5. Stack operations
6. Matrix operations
7. Sequence functional operations
8. Functional vector operations
9. Conditional operations
10. Selection operations

TYPES OF OPERATION ENCODINGS

1. Floating point constant operations (Push)

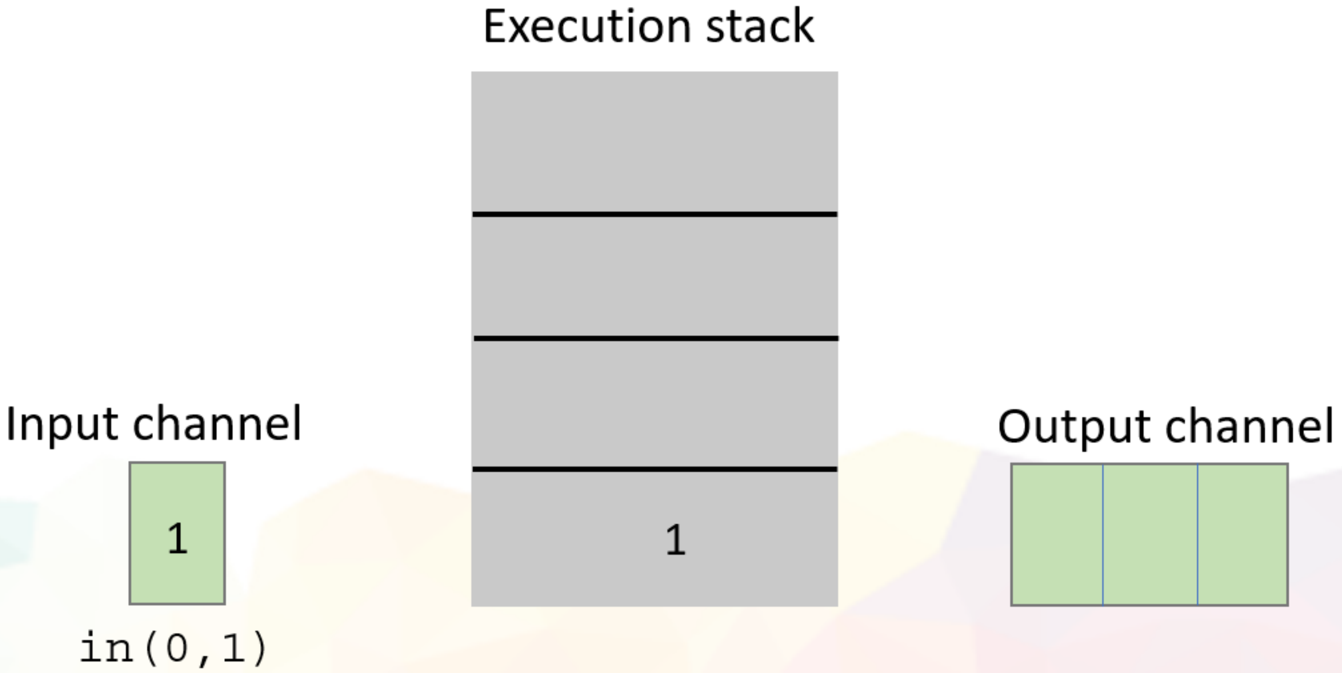
```
<MultiProcessElements InputChannels=“1" OutputChannels=“3">
        <CalculatorElement InputChannels=“1" OutputChannels=“3">
                <MainFunction>
                {
                        in(0)
                        2.5
                        3
                        out(0,3)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



TYPES OF OPERATION ENCODINGS

1. Floating point constant operations (Push)

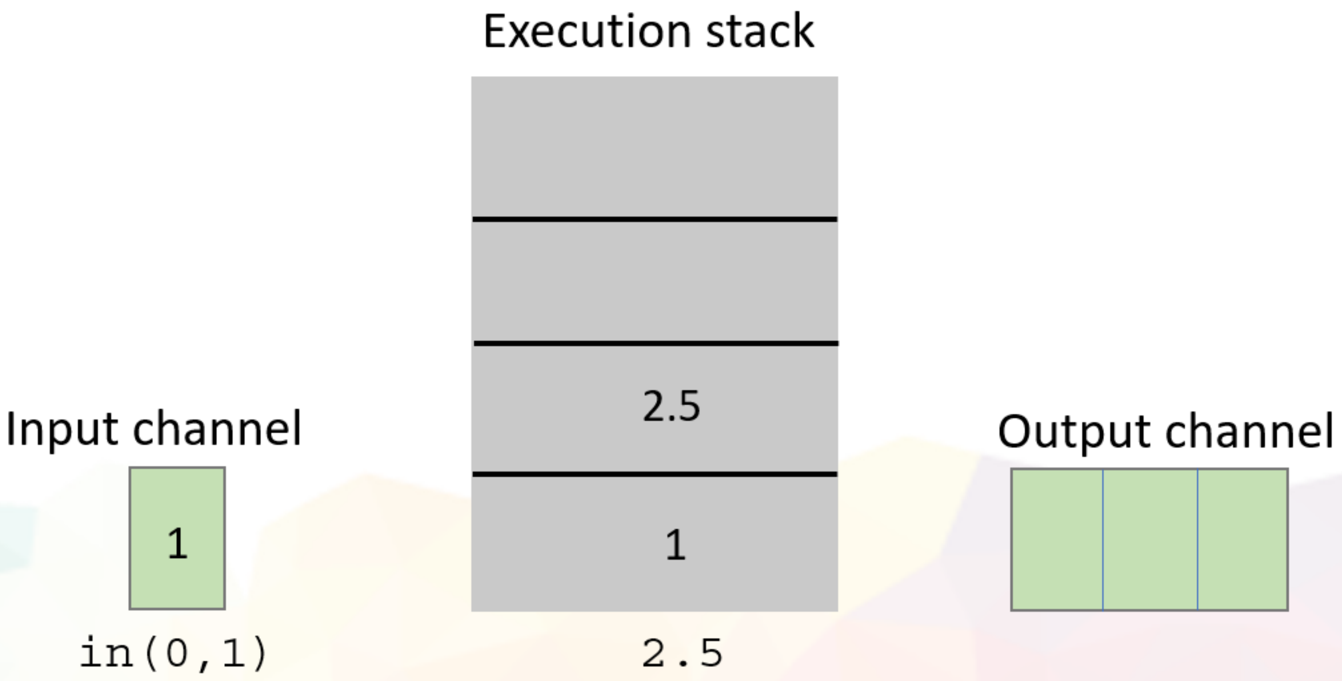
```
<MultiProcessElements InputChannels=“1" OutputChannels=“3">
        <CalculatorElement InputChannels=“1" OutputChannels=“3">
                <MainFunction>
                {
                        in(0)
                        2.5
                        3
                        out(0,3)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



TYPES OF OPERATION ENCODINGS

1. Floating point constant operations (Push)

```
<MultiProcessElements InputChannels=“1" OutputChannels=“3">
        <CalculatorElement InputChannels=“1" OutputChannels=“3">
                <MainFunction>
                {
                        in(0)
                        2.5
                        3
                        out(0,3)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



Execution stack

2.5
1

Input channel

1

Output channel

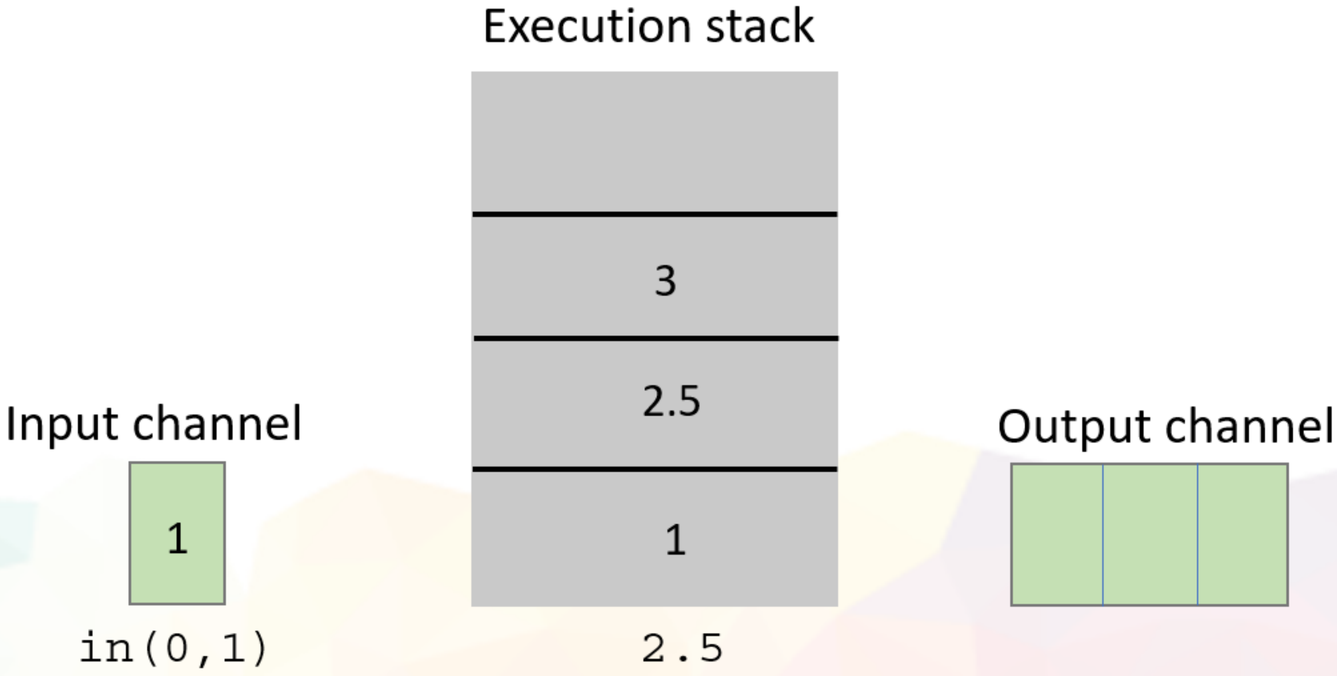
`in(0,1)`

`2.5`

TYPES OF OPERATION ENCODINGS

1. Floating point constant operations (Push)

```
<MultiProcessElements InputChannels=“1" OutputChannels=“3">
        <CalculatorElement InputChannels=“1" OutputChannels=“3">
                <MainFunction>
                {
                        in(0)
                        2.5
                        3
                        out(0,3)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



Execution stack

Input channel

Output channel

`in(0,1)`

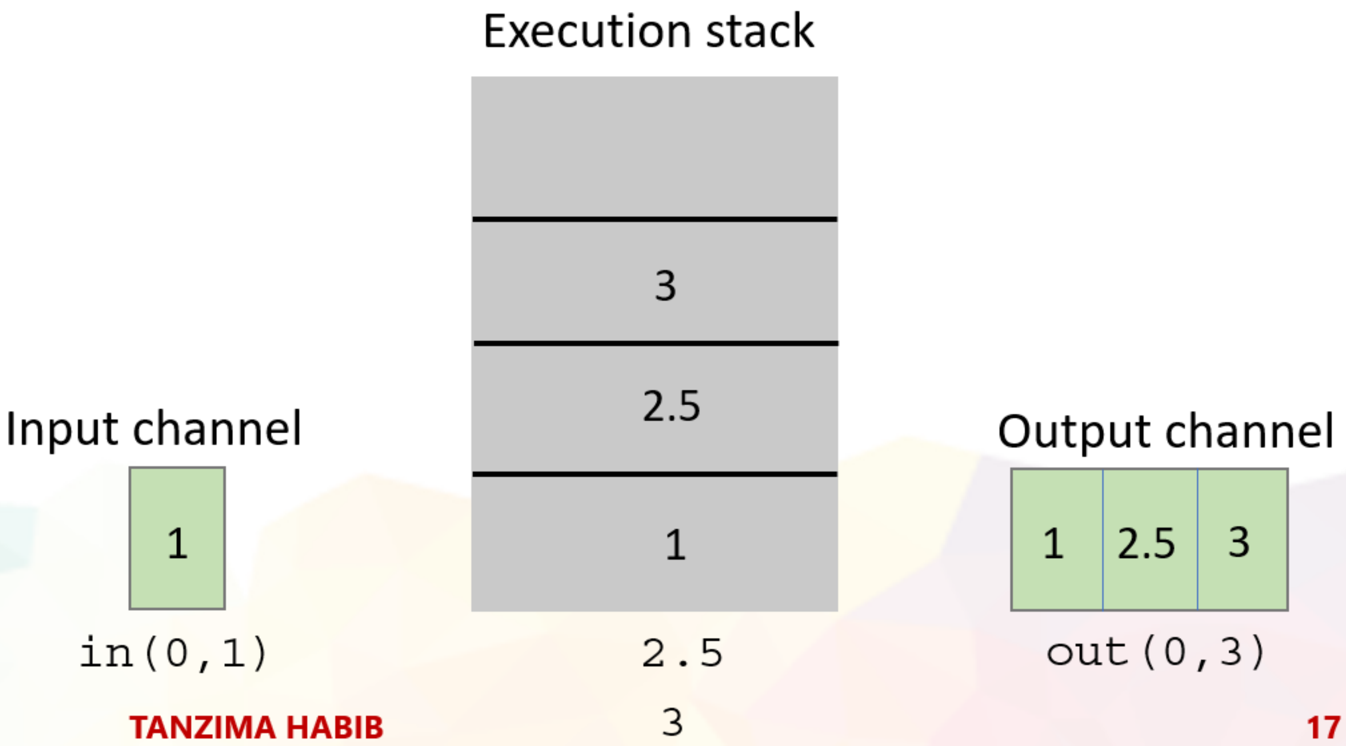
`2.5`

`3`

TYPES OF OPERATION ENCODINGS

1. Floating point constant operations (Push)

```
<MultiProcessElements InputChannels=“1" OutputChannels=“3">
        <CalculatorElement InputChannels=“1" OutputChannels=“3">
                <MainFunction>
                {
                        in(0)
                        2.5
                        3
                        out(0,3)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



Execution stack

3

2.5

1

Input channel

1

in(0,1)

2.5

3

Output channel

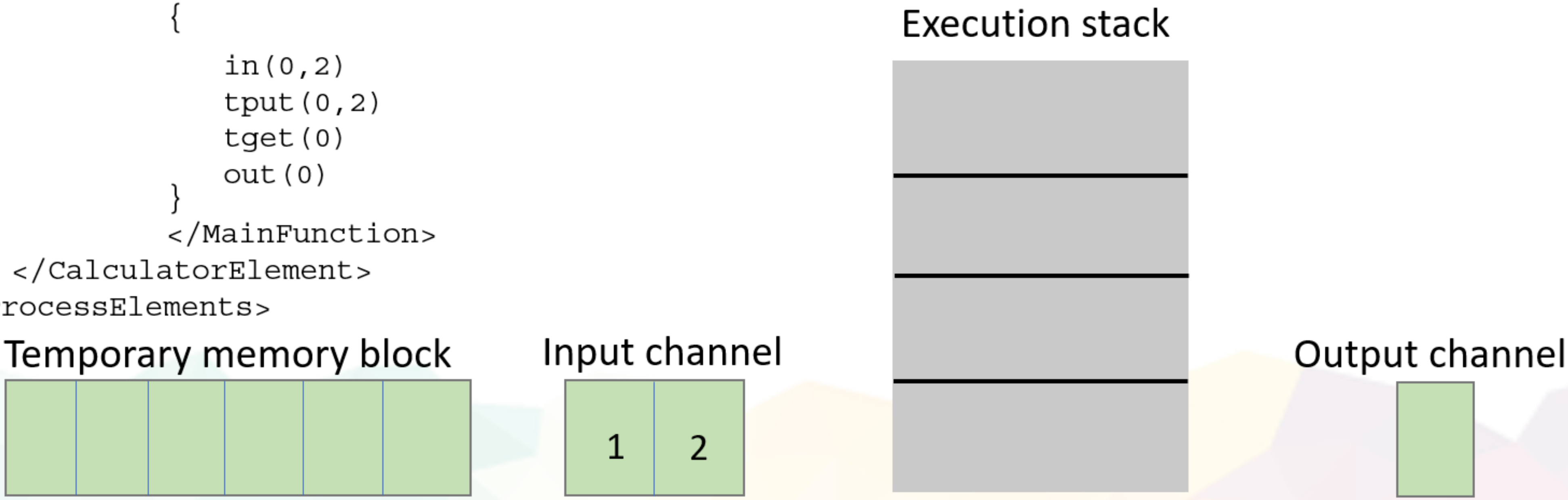
1	2.5	3

out(0,3)

TYPES OF OPERATION ENCODINGS

2. Channel vector operations

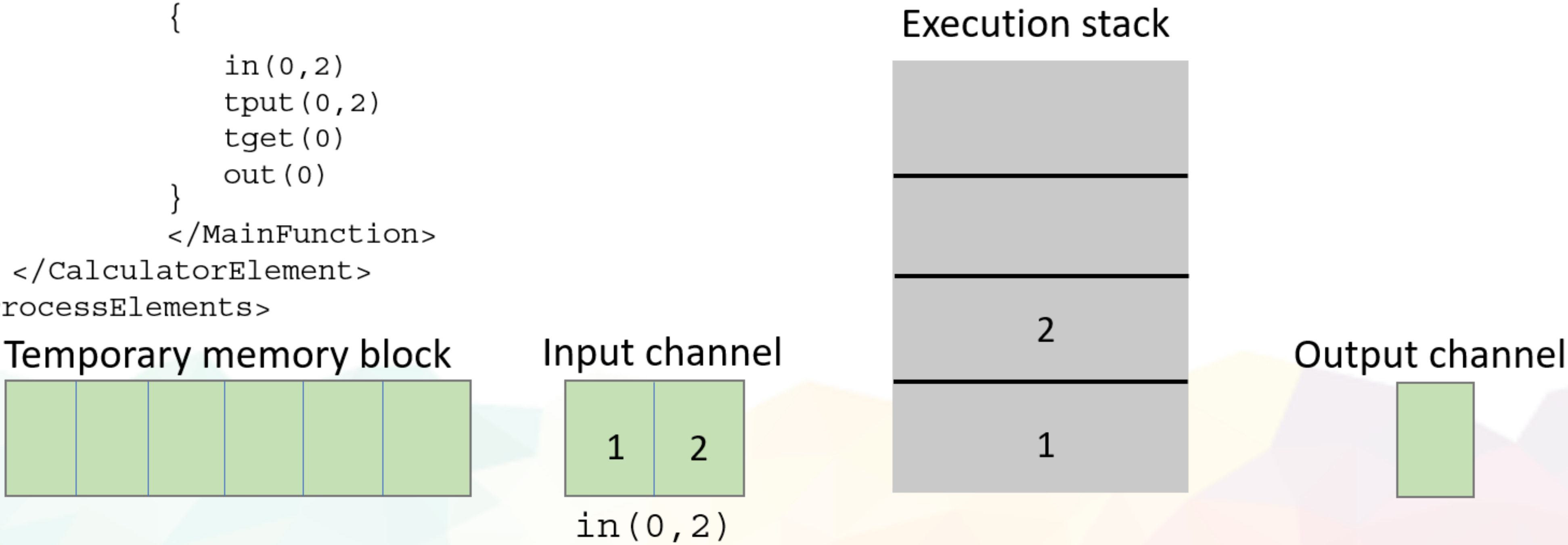
```
<MultiProcessElements InputChannels=“2" OutputChannels=“1">
        <CalculatorElement InputChannels=“2" OutputChannels=“1">
                <MainFunction>
                {
                   in(0,2)
                   tput(0,2)
                   tget(0)
                   out(0)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



TYPES OF OPERATION ENCODINGS

2. Channel vector operations

```
<MultiProcessElements InputChannels=“2" OutputChannels=“1">
        <CalculatorElement InputChannels=“2" OutputChannels=“1">
                <MainFunction>
                {
                   in(0,2)
                   tput(0,2)
                   tget(0)
                   out(0)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



Execution stack

2
1

Temporary memory block

Input channel

1	2

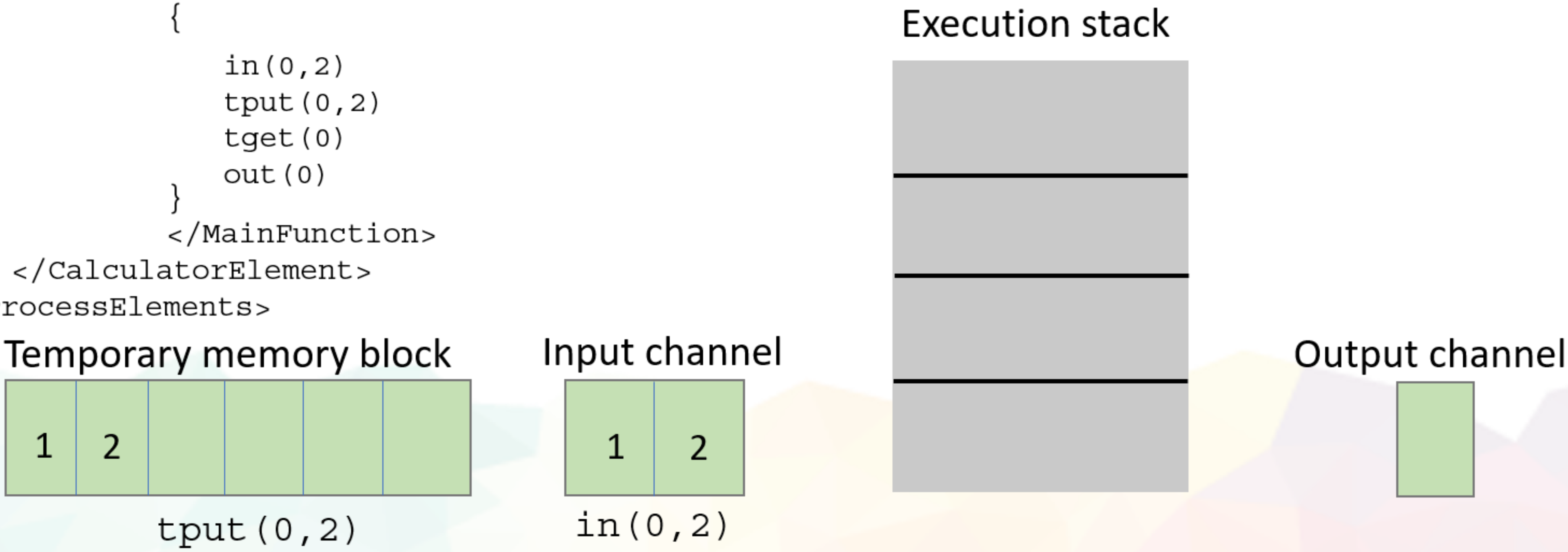
`in(0,2)`

Output channel

TYPES OF OPERATION ENCODINGS

2. Channel vector operations

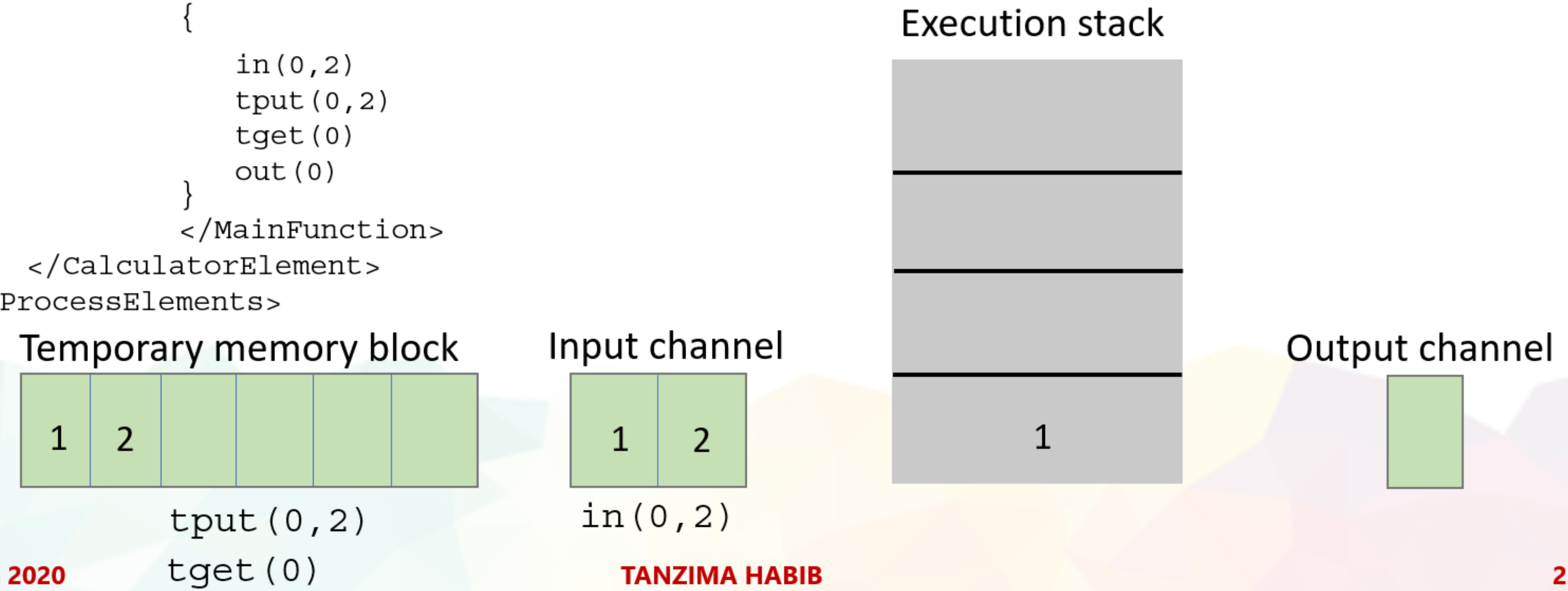
```
<MultiProcessElements InputChannels=“2" OutputChannels=“1">
        <CalculatorElement InputChannels=“2" OutputChannels=“1">
                <MainFunction>
                {

                   in(0,2)
                   tput(0,2)
                   tget(0)
                   out(0)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



TYPES OF OPERATION ENCODINGS

2. Channel vector operations

```
<MultiProcessElements InputChannels=“2" OutputChannels=“1">
        <CalculatorElement InputChannels=“2" OutputChannels=“1">
                <MainFunction>
                {
                   in(0,2)
                   tput(0,2)
                   tget(0)
                   out(0)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



Execution stack

1

Temporary memory block

1	2				

tput(0,2)

tget(0)

Input channel

1	2

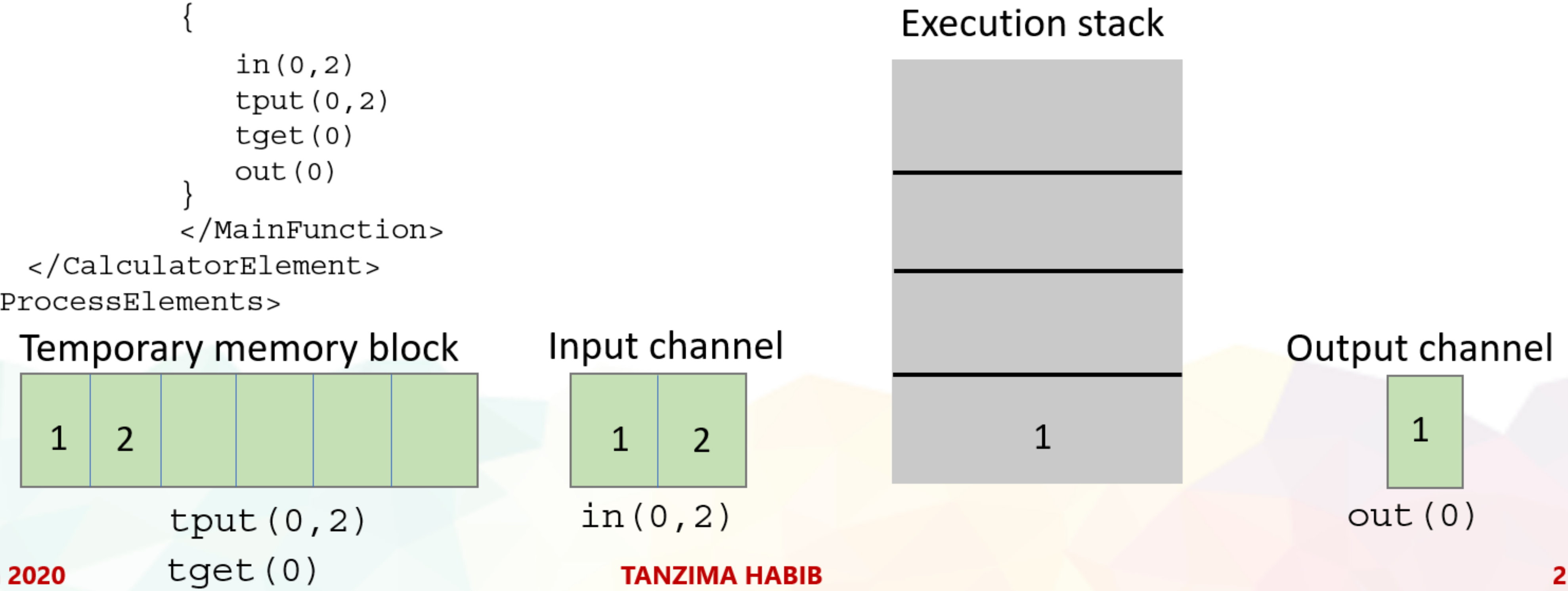
in(0,2)

Output channel

TYPES OF OPERATION ENCODINGS

2. Channel vector operations

```
<MultiProcessElements InputChannels="2" OutputChannels="1">
        <CalculatorElement InputChannels="2" OutputChannels="1">
                <MainFunction>
                {
                   in(0,2)
                   tput(0,2)
                   tget(0)
                   out(0)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



Execution stack

1

Temporary memory block

1	2				

`tput(0,2)`

`tget(0)`

Input channel

1	2

`in(0,2)`

Output channel

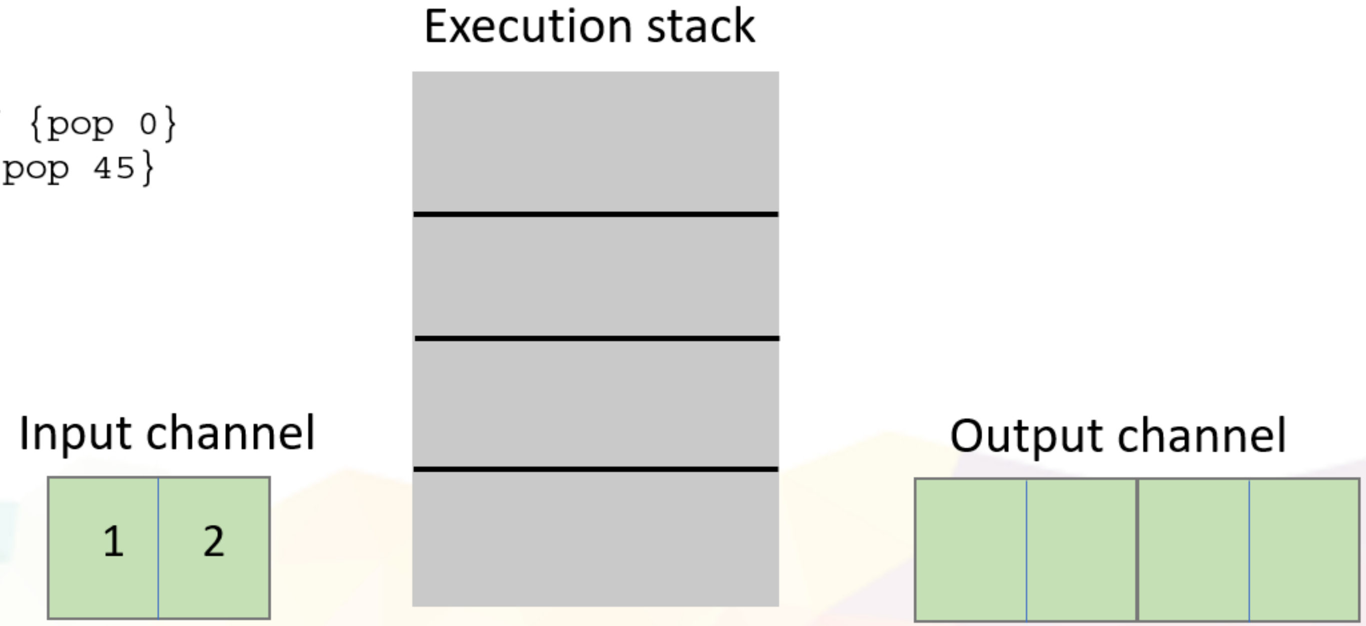
1

`out(0)`

TYPES OF OPERATION ENCODINGS

3. CMM environment variable operation

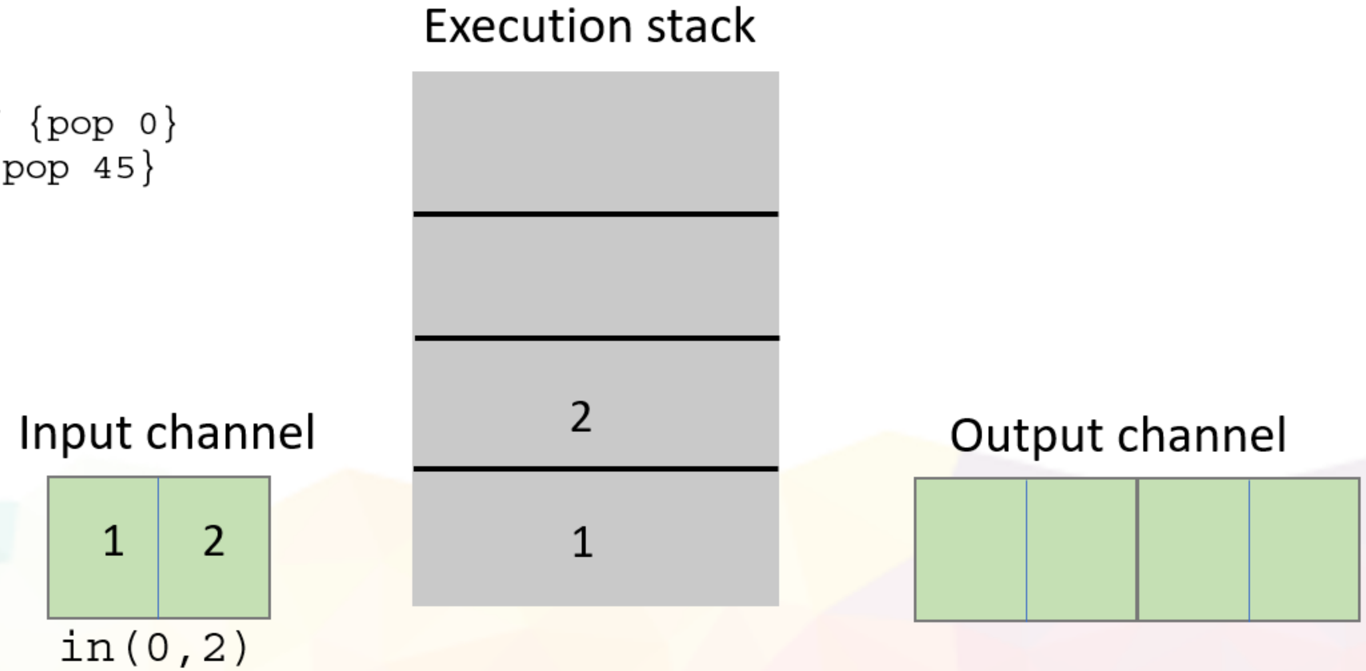
```
<MultiProcessElements InputChannels=“2" OutputChannels=“4">
        <CalculatorElement InputChannels=“2" OutputChannels=“4">
                <MainFunction>
                {
                   in(0,2)
                   env(gamma) not if {pop 0}
                   env(var) not if {pop 45}
                   out(0,4)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



TYPES OF OPERATION ENCODINGS

3. CMM environment variable operation

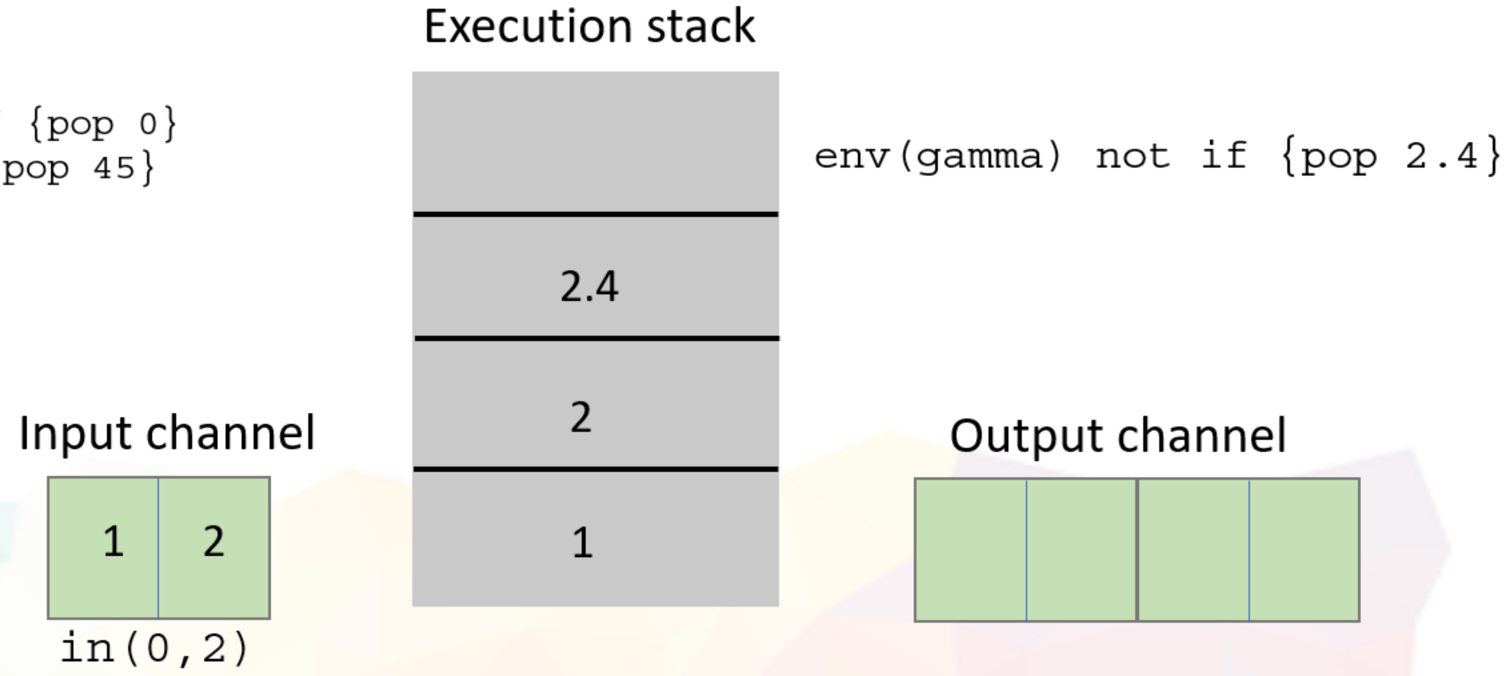
```
<MultiProcessElements InputChannels=“2" OutputChannels=“4">
        <CalculatorElement InputChannels=“2" OutputChannels=“4">
                <MainFunction>
                {
                   in(0,2)
                   env(gamma) not if {pop 0}
                   env(var) not if {pop 45}
                   out(0,4)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



TYPES OF OPERATION ENCODINGS

3. CMM environment variable operation

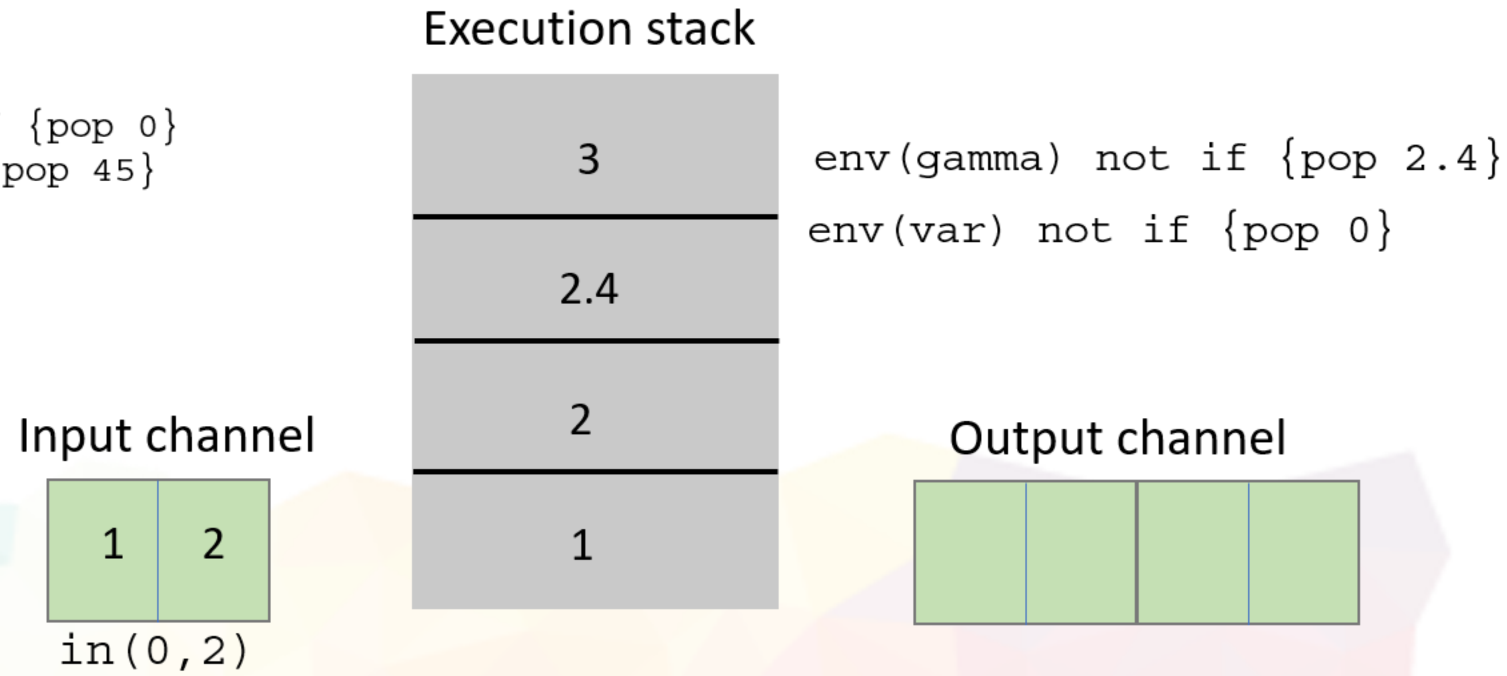
```
<MultiProcessElements InputChannels=“2" OutputChannels=“4">
        <CalculatorElement InputChannels=“2" OutputChannels=“4">
                <MainFunction>
                {
                   in(0,2)
                   env(gamma) not if {pop 0}
                   env(var) not if {pop 45}
                   out(0,4)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



TYPES OF OPERATION ENCODINGS

3. CMM environment variable operation

```
<MultiProcessElements InputChannels=“2" OutputChannels=“4">
        <CalculatorElement InputChannels=“2" OutputChannels=“4">
                <MainFunction>
                {
                   in(0,2)
                   env(gamma) not if {pop 0}
                   env(var) not if {pop 45}
                   out(0,4)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```

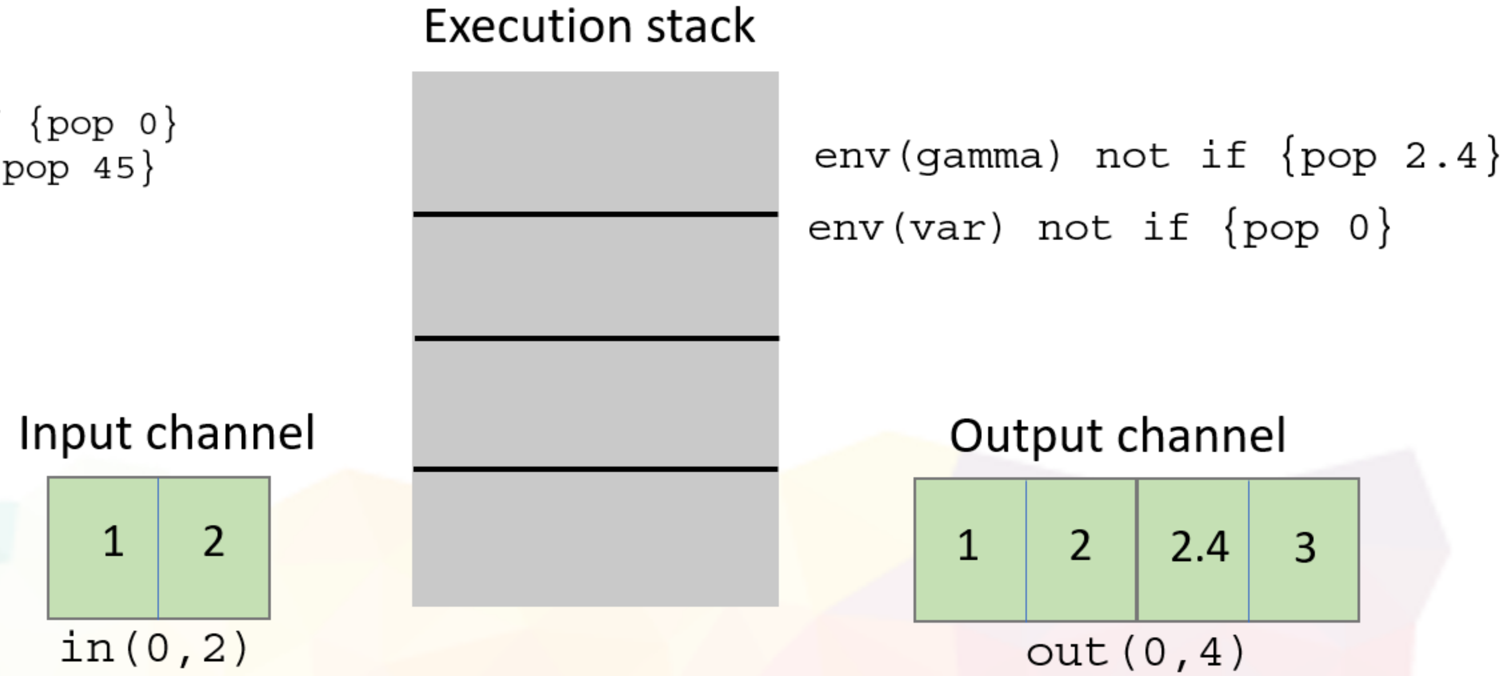


```
env(gamma) not if {pop 2.4}
env(var) not if {pop 0}
```

TYPES OF OPERATION ENCODINGS

3. CMM environment variable operation

```
<MultiProcessElements InputChannels=“2" OutputChannels=“4">
        <CalculatorElement InputChannels=“2" OutputChannels=“4">
                <MainFunction>
                {
                   in(0,2)
                   env(gamma) not if {pop 0}
                   env(var) not if {pop 45}
                   out(0,4)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



Execution stack

env(gamma) not if {pop 2.4}

env(var) not if {pop 0}

Input channel

1	2

in(0,2)

Output channel

1	2	2.4	3

out(0,4)

TYPES OF OPERATION ENCODINGS

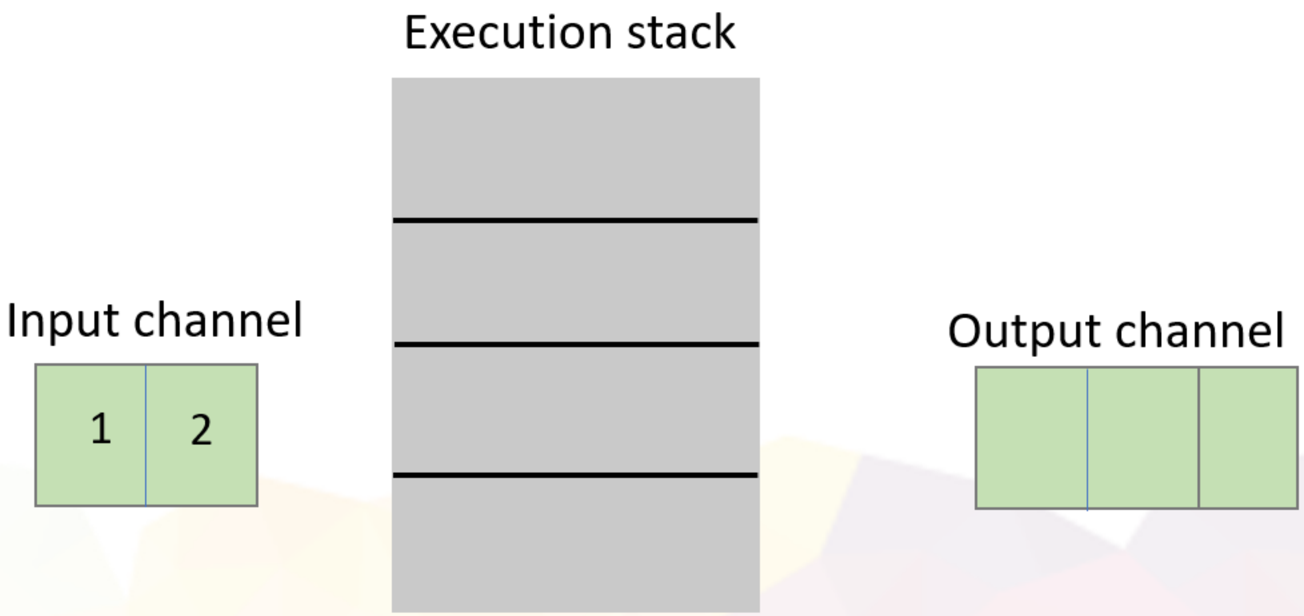
4. Sub-element invocation operations

```
<MultiProcessElements InputChannels=“3" OutputChannels=“3">
<SubElements>
<CurveSetElement Name=”applyGamma” InputChannels=”3” OutputChannels=”3”>…</CurveSetElement>
<MatrixElement Name=”RGBtoXYZ” InputChannels=”3” OutputChannels=”3”>…</MatrixElement>
</SubElements>
        <CalculatorElement InputChannels=“3" OutputChannels=“3">
                <MainFunction>
                {
                 in(0,3)
                 curv{applyGamma}
                 mtx{RGBtoXYZ}
                 out(0,3)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```

TYPES OF OPERATION ENCODINGS

5. Stack operations

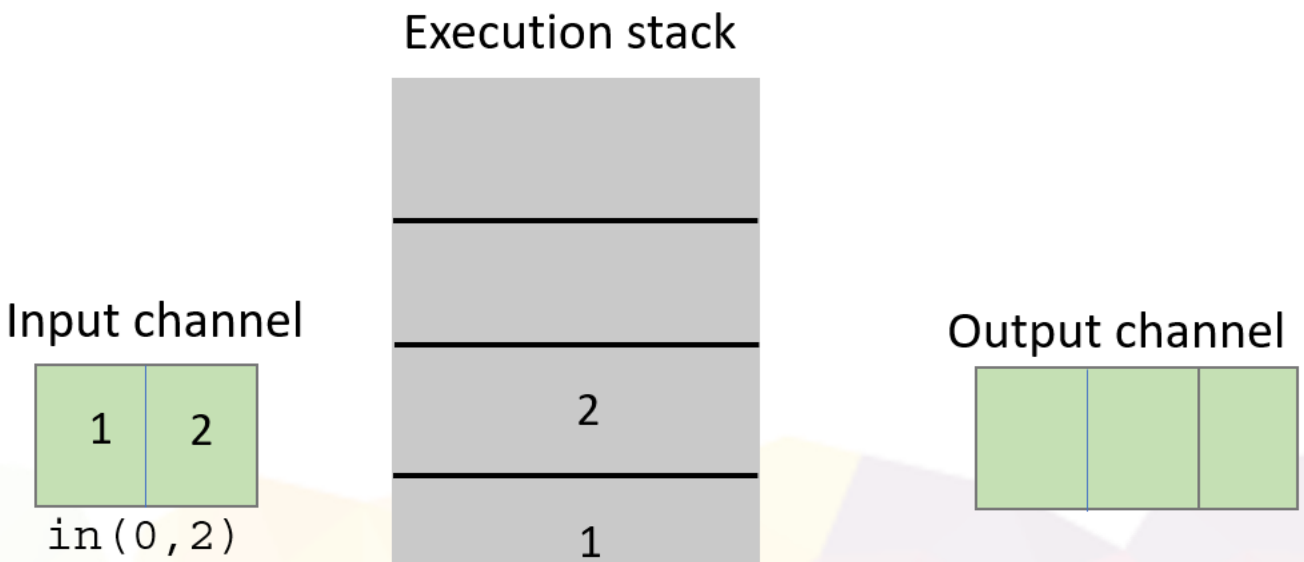
```
<MultiProcessElements InputChannels=“2" OutputChannels=“3">
        <CalculatorElement InputChannels=“2" OutputChannels=“3">
                <MainFunction>
                {
                   in(0,2)
                   copy(2)
                   pop(1)
                   flip(3)
                   out(0,3)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



TYPES OF OPERATION ENCODINGS

5. Stack operations

```
<MultiProcessElements InputChannels=“2" OutputChannels=“3">
        <CalculatorElement InputChannels=“2" OutputChannels=“3">
                <MainFunction>
                {
                   in(0,2)
                   copy(2)
                   pop(1)
                   flip(3)
                   out(0,3)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



Execution stack

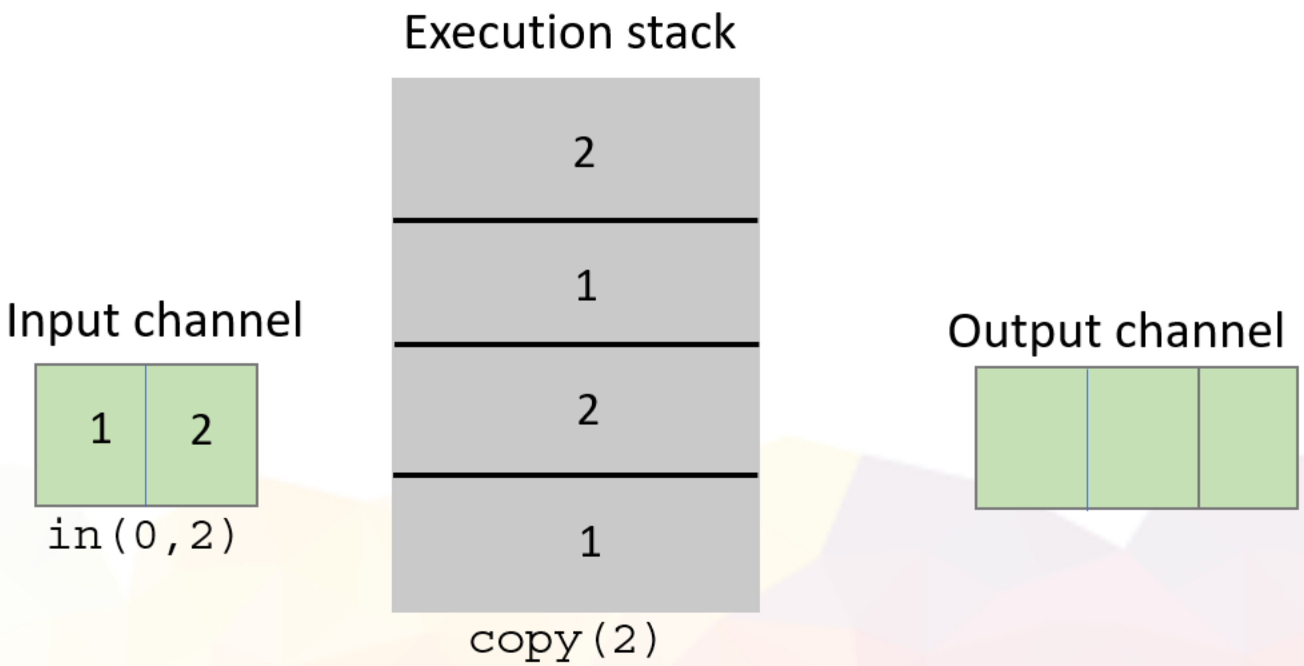
Input channel

Output channel

TYPES OF OPERATION ENCODINGS

5. Stack operations

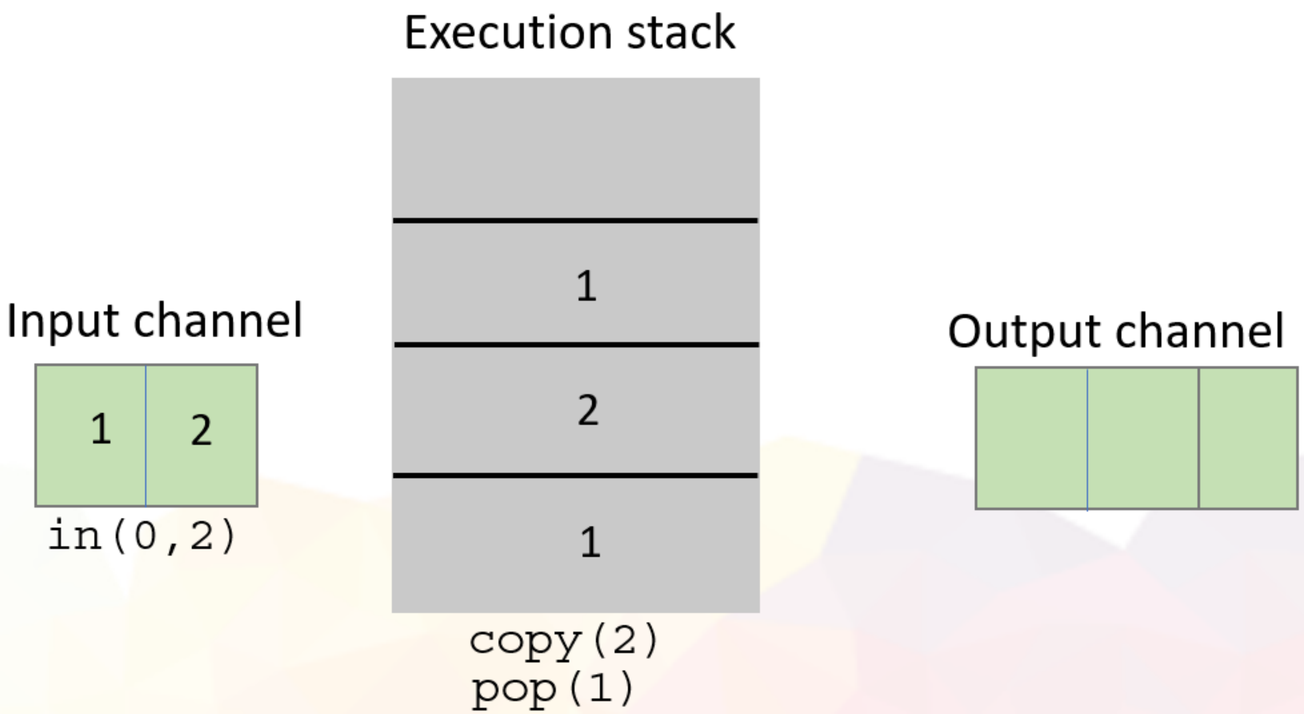
```
<MultiProcessElements InputChannels=“2" OutputChannels=“3">
        <CalculatorElement InputChannels=“2" OutputChannels=“3">
                <MainFunction>
                {
                   in(0,2)
                   copy(2)
                   pop(1)
                   flip(3)
                   out(0,3)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



TYPES OF OPERATION ENCODINGS

5. Stack operations

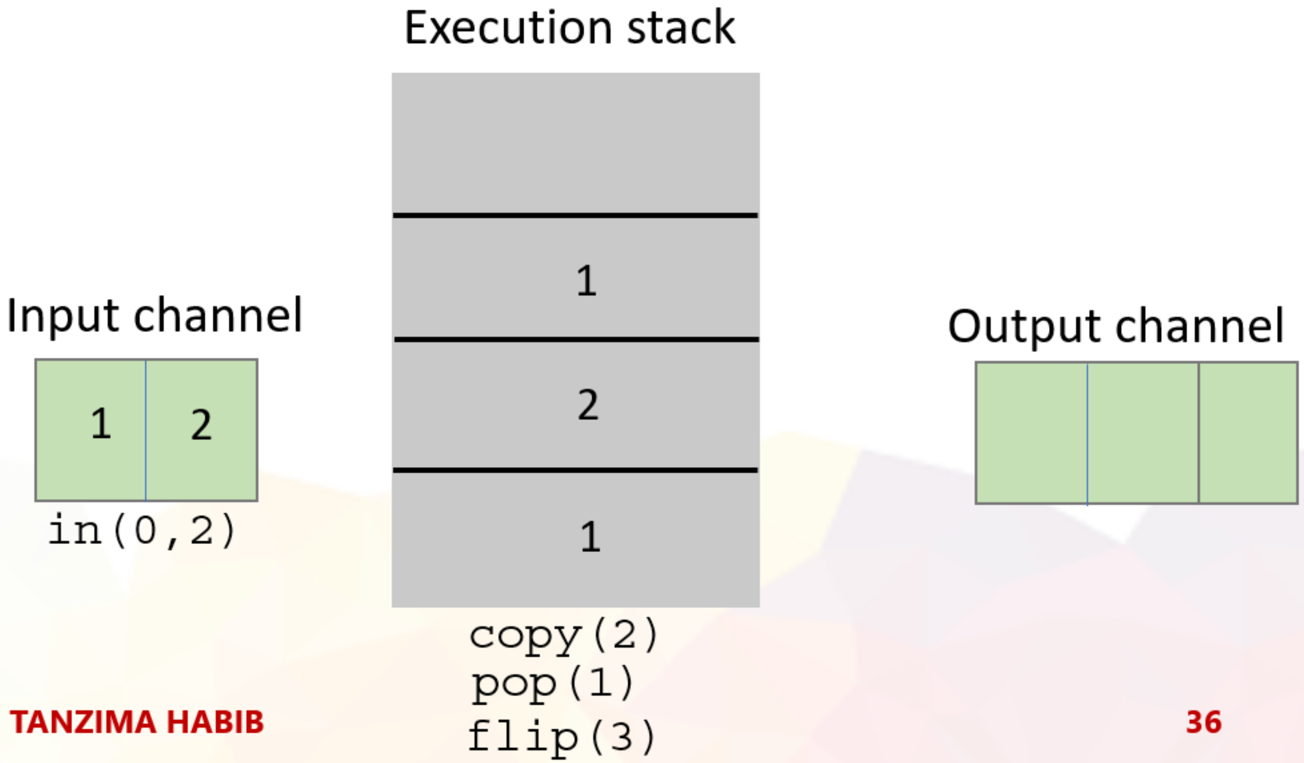
```
<MultiProcessElements InputChannels=“2" OutputChannels=“3">
        <CalculatorElement InputChannels=“2" OutputChannels=“3">
                <MainFunction>
                {
                   in(0,2)
                   copy(2)
                   pop(1)
                   flip(3)
                   out(0,3)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



TYPES OF OPERATION ENCODINGS

5. Stack operations

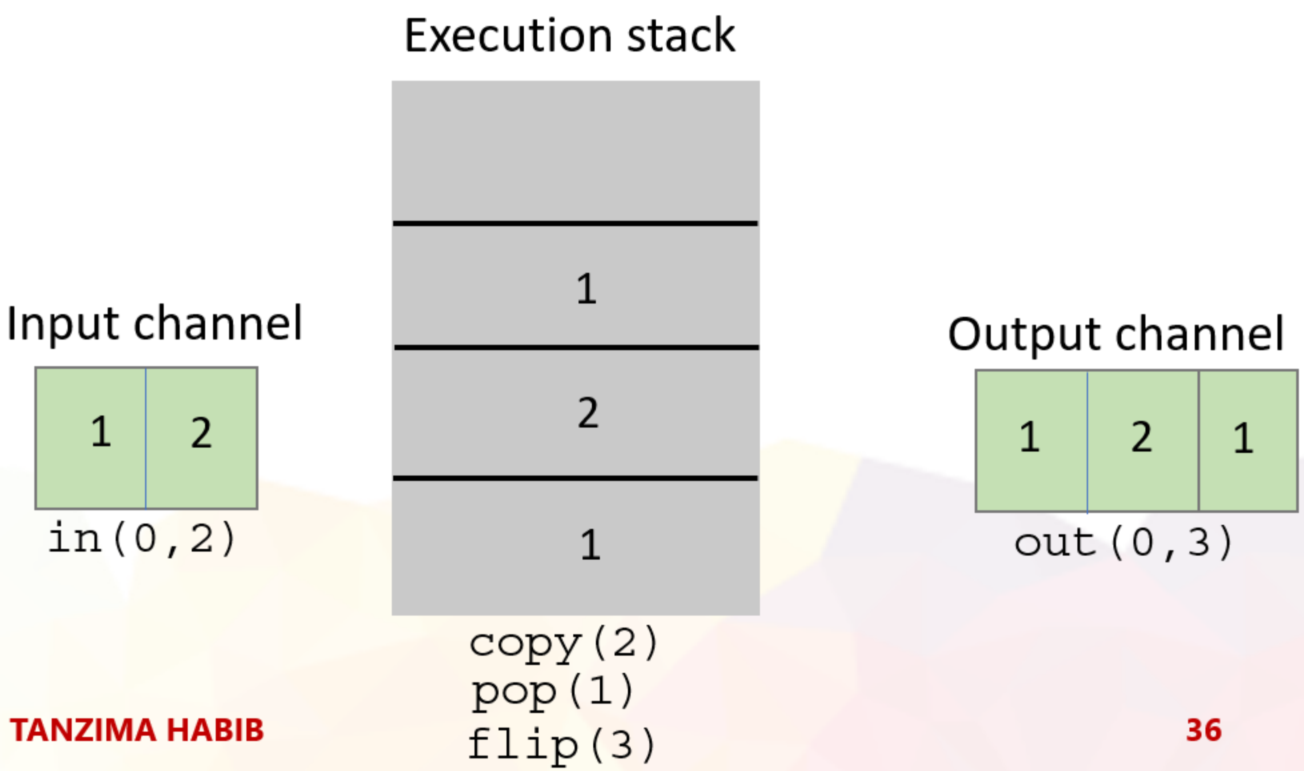
```
<MultiProcessElements InputChannels=“2" OutputChannels=“3">
        <CalculatorElement InputChannels=“2" OutputChannels=“3">
                <MainFunction>
                {
                   in(0,2)
                   copy(2)
                   pop(1)
                   flip(3)
                   out(0,3)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



TYPES OF OPERATION ENCODINGS

5. Stack operations

```
<MultiProcessElements InputChannels=“2" OutputChannels=“3">
        <CalculatorElement InputChannels=“2" OutputChannels=“3">
                <MainFunction>
                {
                   in(0,2)
                   copy(2)
                   pop(1)
                   flip(3)
                   out(0,3)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



TYPES OF OPERATION ENCODINGS

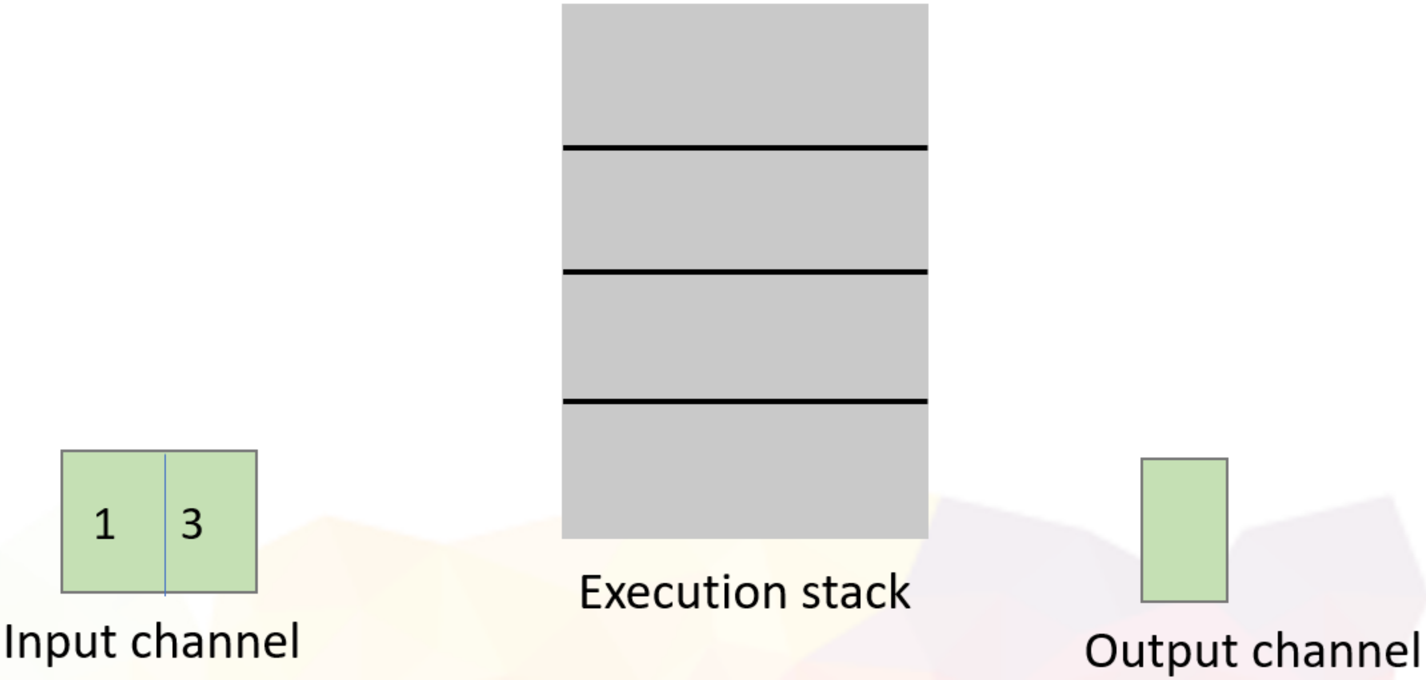
6. Matrix operations

```
<MultiProcessElements InputChannels=“9" OutputChannels=“9">
        <CalculatorElement InputChannels=“9" OutputChannels=“9">
                <MainFunction>
                {
                   in(0,9)
                   tran(3,3)
                   out(0,9)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```

TYPES OF OPERATION ENCODINGS

7. Sequence functional operations

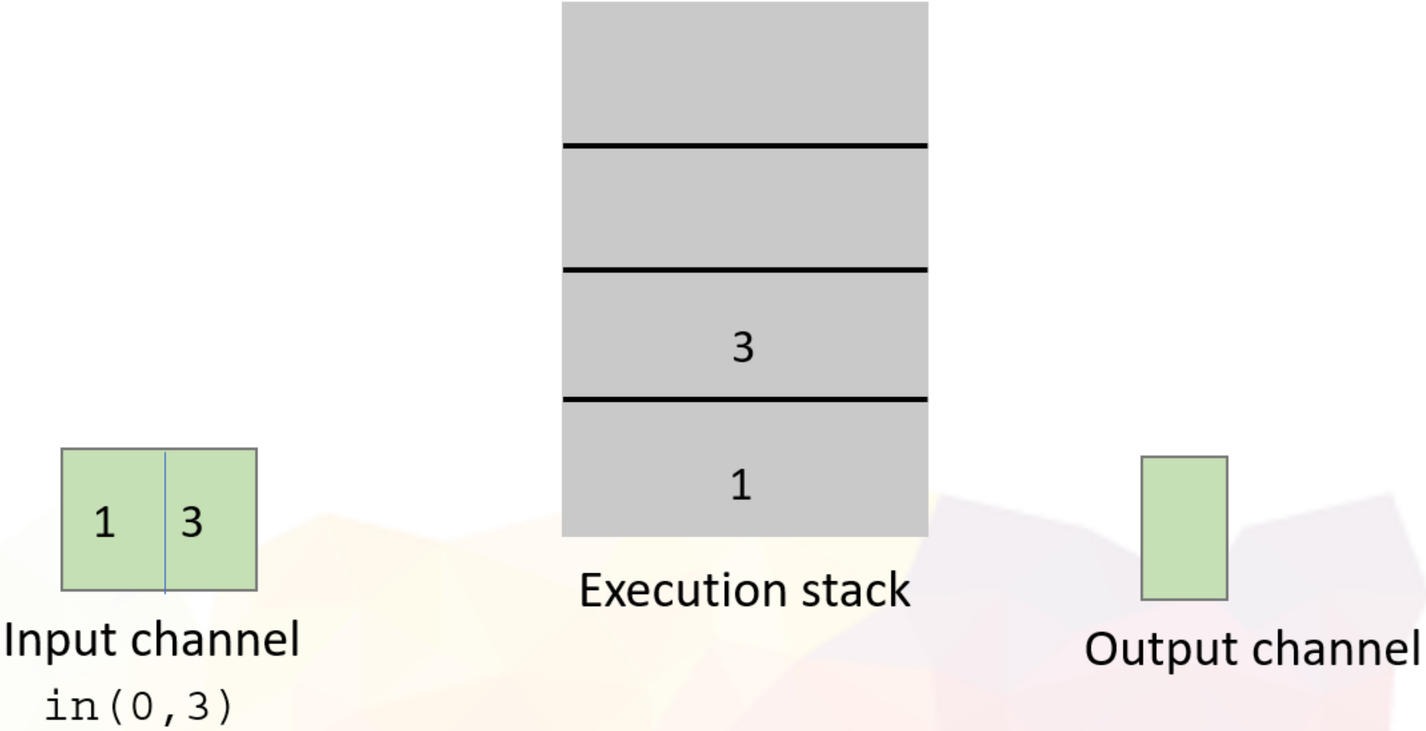
```
<MultiProcessElements InputChannels=“2" OutputChannels=“1">
        <CalculatorElement InputChannels=“2" OutputChannels=“1">
                <MainFunction>
                {
                   in(0,2)

                   sum(1)
                   2 4 prod(1)
                   3 max(2)

                   out(0)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



TYPES OF OPERATION ENCODINGS

7. Sequence functional operations

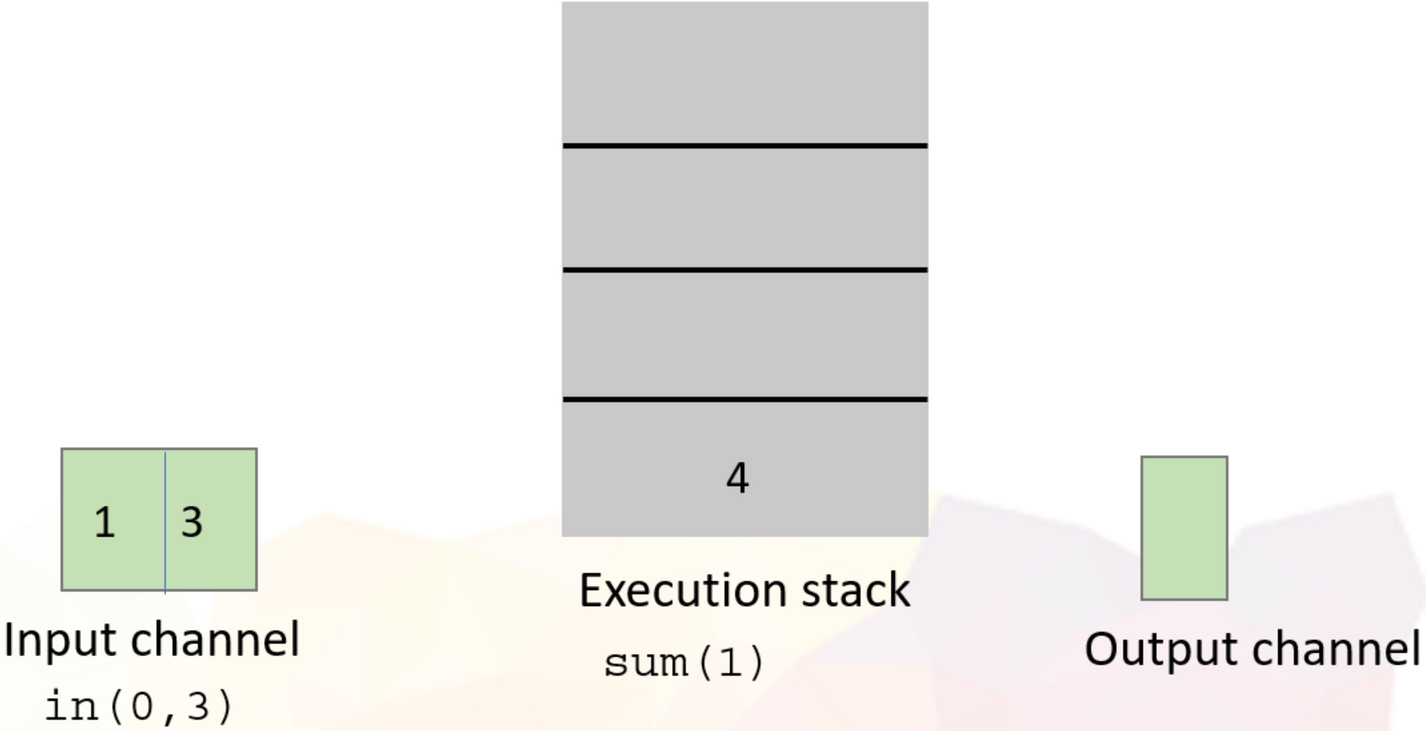
```
<MultiProcessElements InputChannels=“2" OutputChannels=“1">
        <CalculatorElement InputChannels=“2" OutputChannels=“1">
                <MainFunction>
                {
                   in(0,2)

                   sum(1)
                   2 4 prod(1)
                   3 max(2)

                   out(0)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



TYPES OF OPERATION ENCODINGS

7. Sequence functional operations

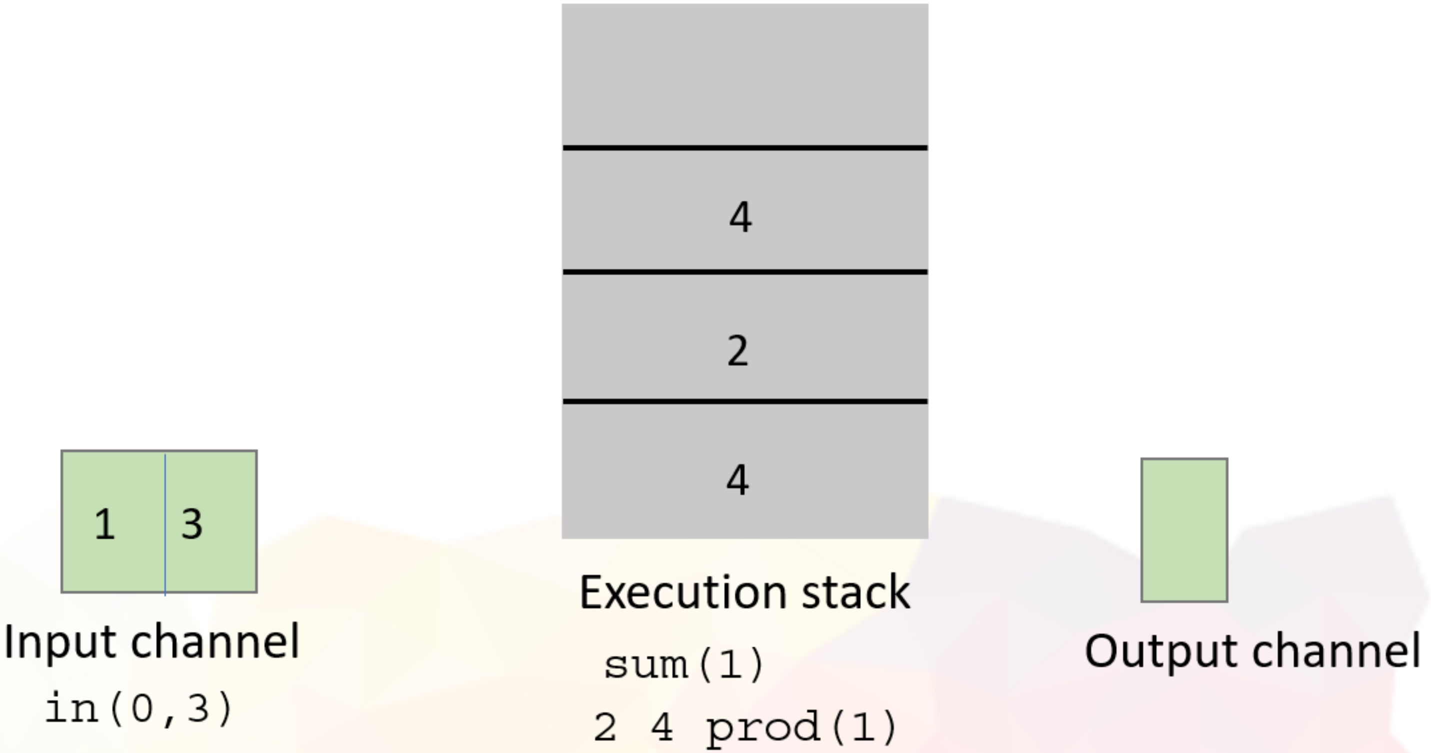
```
<MultiProcessElements InputChannels=“2" OutputChannels=“1">
        <CalculatorElement InputChannels=“2" OutputChannels=“1">
                <MainFunction>
                {
                   in(0,2)
                   sum(1)
                   2 4 prod(1)
                   3 max(2)
                   out(0)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



TYPES OF OPERATION ENCODINGS

7. Sequence functional operations

```
<MultiProcessElements InputChannels=“2" OutputChannels=“1">
        <CalculatorElement InputChannels=“2" OutputChannels=“1">
                <MainFunction>
                {
                   in(0,2)
                    sum(1)
                    2 4 prod(1)
                    3 max(2)
                   out(0)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



Input channel

`in(0,3)`

Execution stack

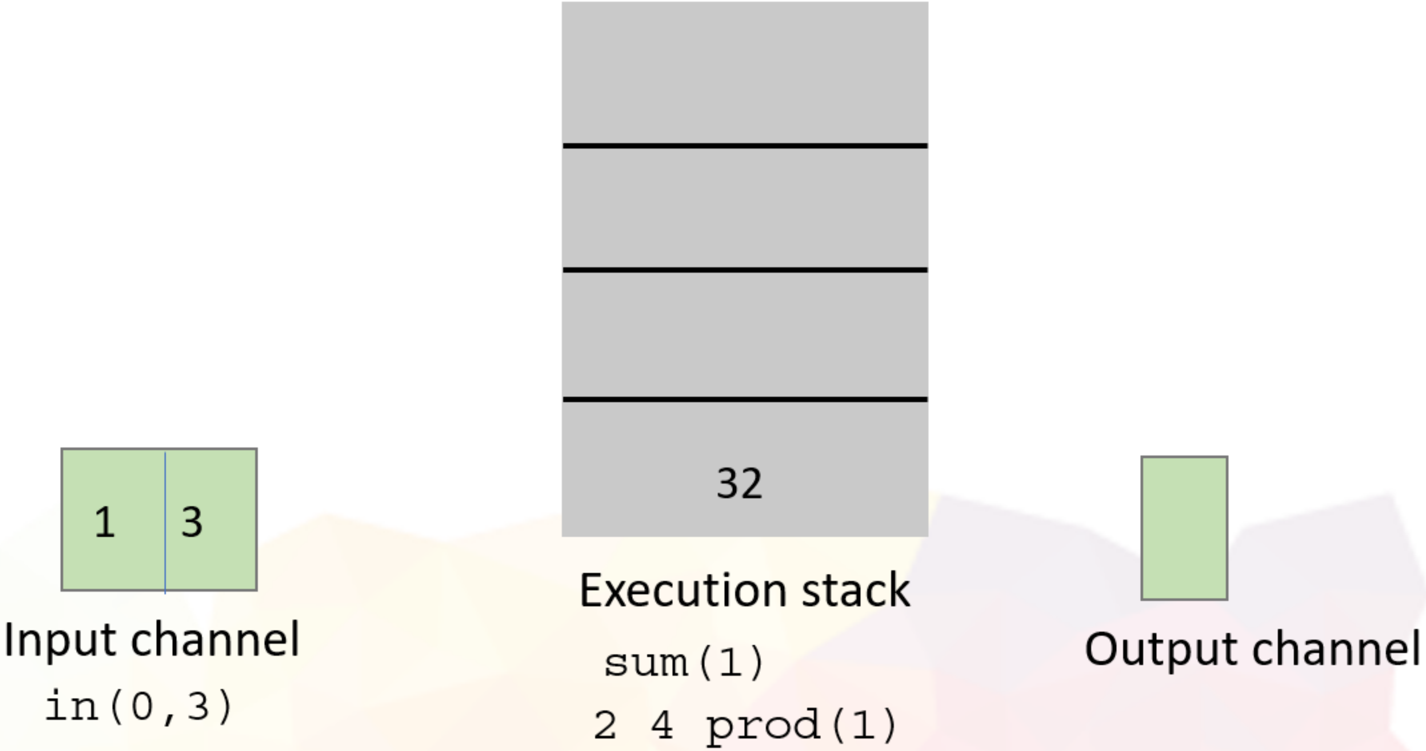
```
 sum(1)
2 4 prod(1)
```

Output channel

TYPES OF OPERATION ENCODINGS

7. Sequence functional operations

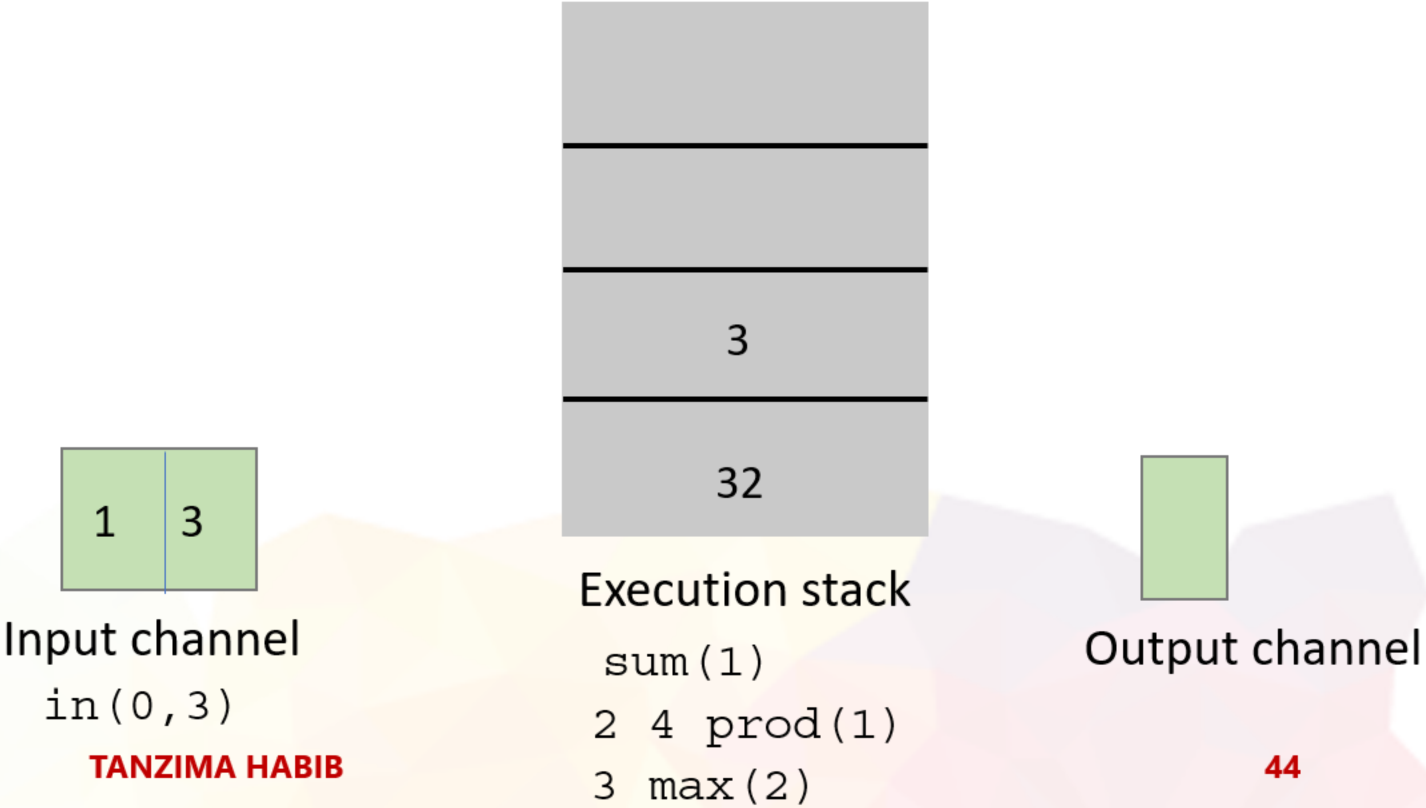
```
<MultiProcessElements InputChannels=“2" OutputChannels=“1">
        <CalculatorElement InputChannels=“2" OutputChannels=“1">
                <MainFunction>
                {
                   in(0,2)

                   sum(1)
                   2 4 prod(1)
                   3 max(2)

                   out(0)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



TYPES OF OPERATION ENCODINGS

7. Sequence functional operations

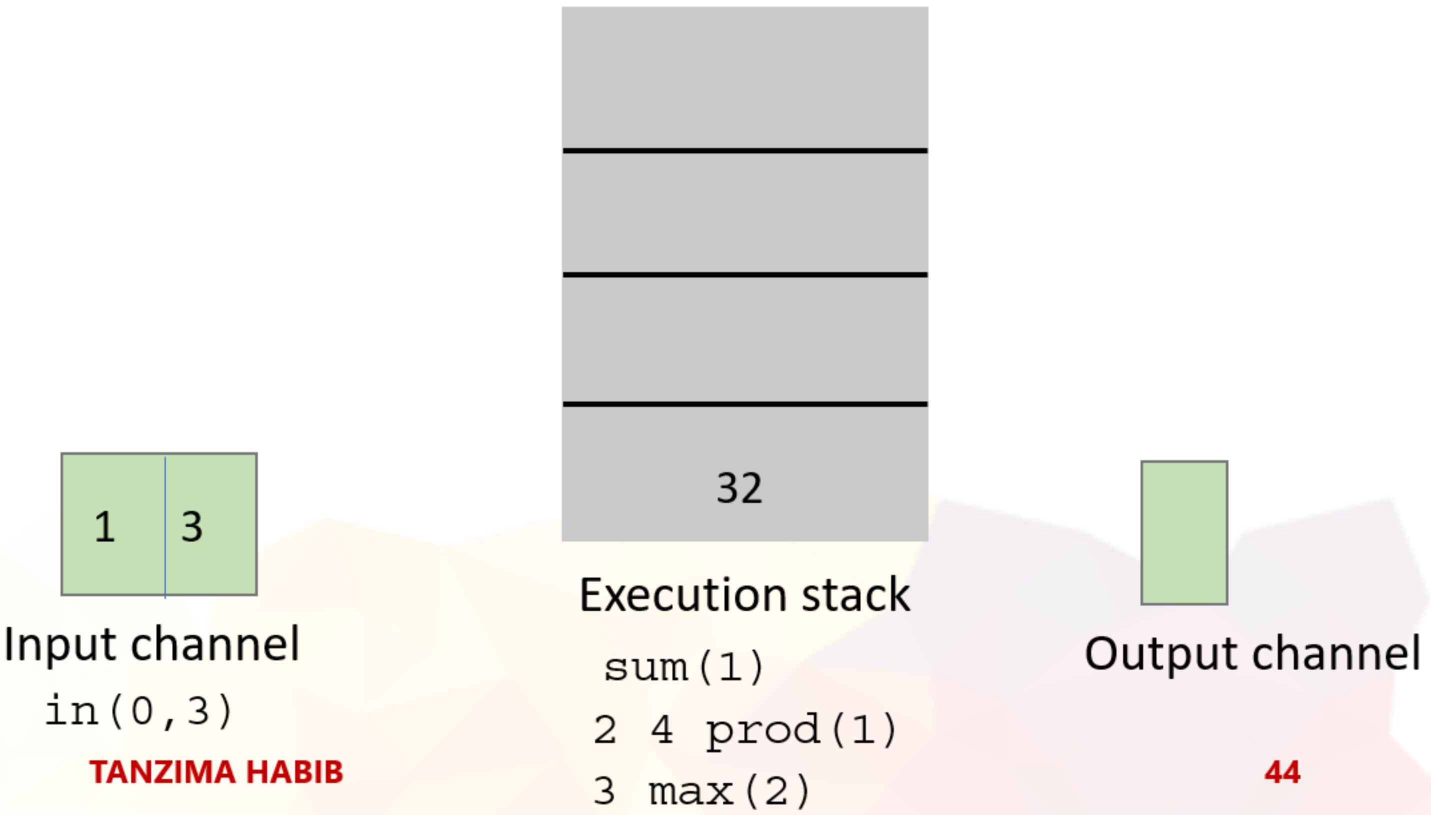
```
<MultiProcessElements InputChannels="2" OutputChannels="1">
        <CalculatorElement InputChannels="2" OutputChannels="1">
                <MainFunction>
                {
                   in(0,2)
                   sum(1)
                   2 4 prod(1)
                   3 max(2)
                   out(0)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



TYPES OF OPERATION ENCODINGS

7. Sequence functional operations

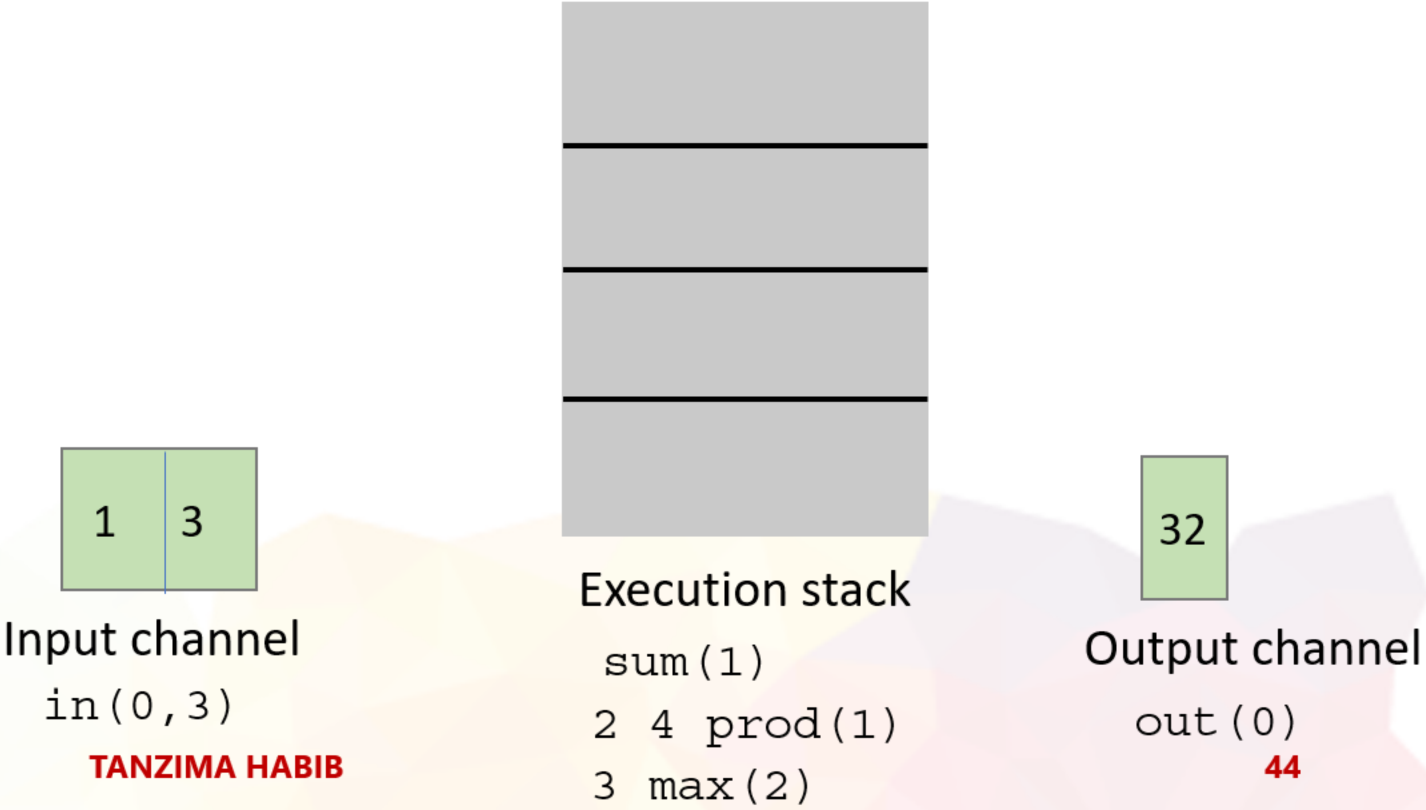
```
<MultiProcessElements InputChannels=“2" OutputChannels=“1">
        <CalculatorElement InputChannels=“2" OutputChannels=“1">
                <MainFunction>
                {
                   in(0,2)

                   sum(1)
                   2 4 prod(1)
                   3 max(2)

                   out(0)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



TYPES OF OPERATION ENCODINGS

7. Sequence functional operations

```
<MultiProcessElements InputChannels=“2" OutputChannels=“1">
        <CalculatorElement InputChannels=“2" OutputChannels=“1">
                <MainFunction>
                {
                   in(0,2)
                    sum(1)
                    2 4 prod(1)
                    3 max(2)
                   out(0)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



Input channel

```
in(0,3)
```

Execution stack

```
 sum(1)
2 4 prod(1)
3 max(2)
```

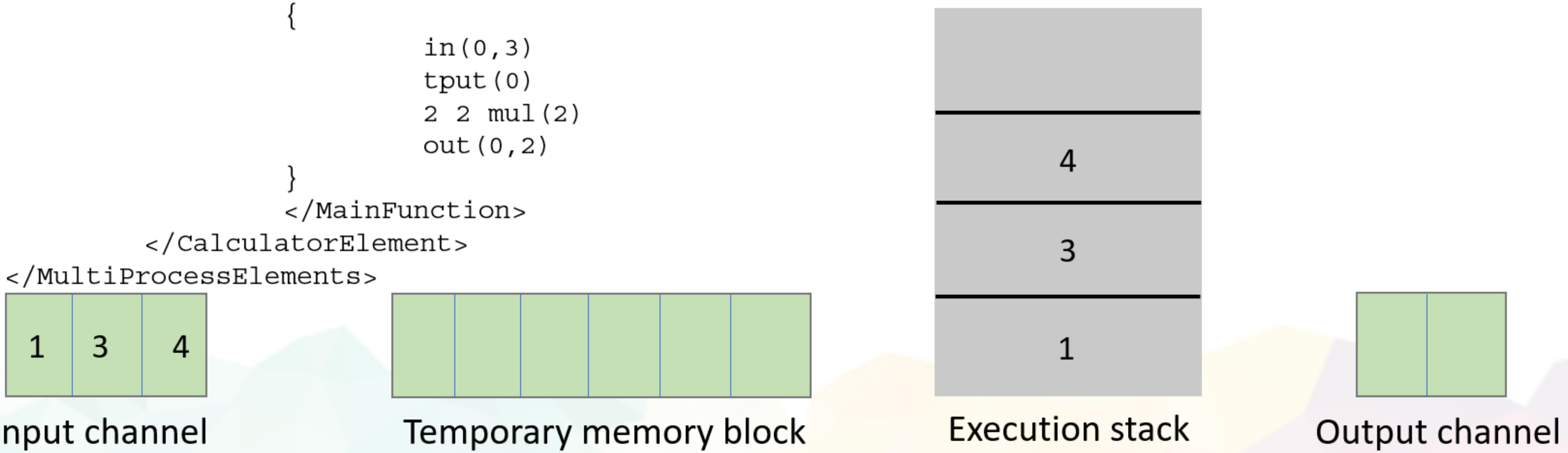
Output channel

```
out(0)
```

TYPES OF OPERATION ENCODINGS

8. Functional vector operations

```
<MultiProcessElements InputChannels="3" OutputChannels=“2">
        <CalculatorElement InputChannels="3" OutputChannels=“2">
                <MainFunction>
                {
                        in(0,3)
                        tput(0)
                        2 2 mul(2)
                        out(0,2)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```

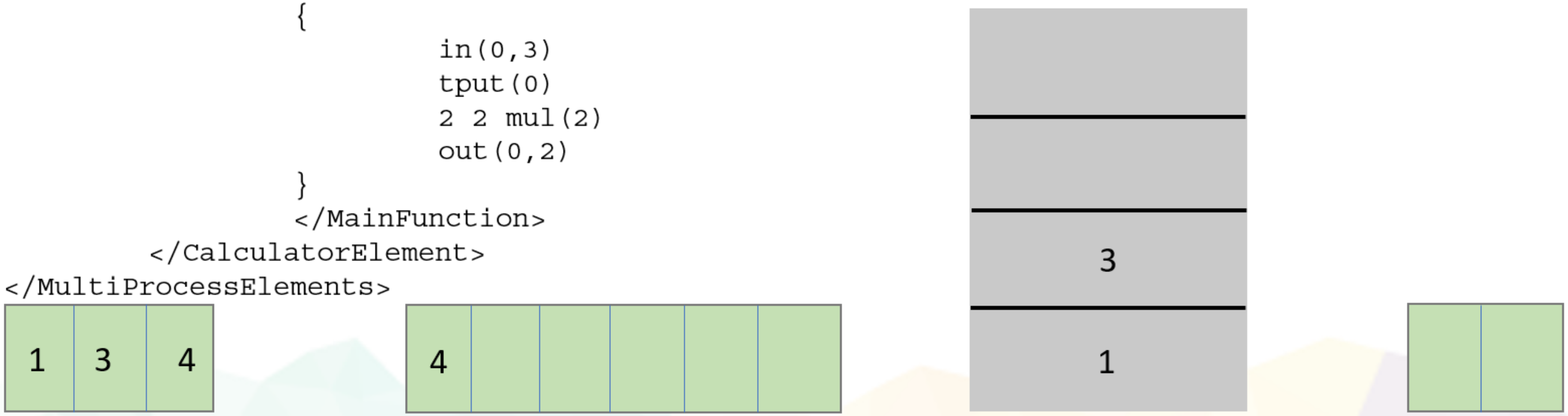


`in(0,3)`

TYPES OF OPERATION ENCODINGS

8. Functional vector operations

```
<MultiProcessElements InputChannels="3" OutputChannels=“2">
        <CalculatorElement InputChannels="3" OutputChannels=“2">
                <MainFunction>
                {
                        in(0,3)
                        tput(0)
                        2 2 mul(2)
                        out(0,2)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



1	3	4

Input channel

in(0,3)

4					

Temporary memory block

tput(0)

3
1

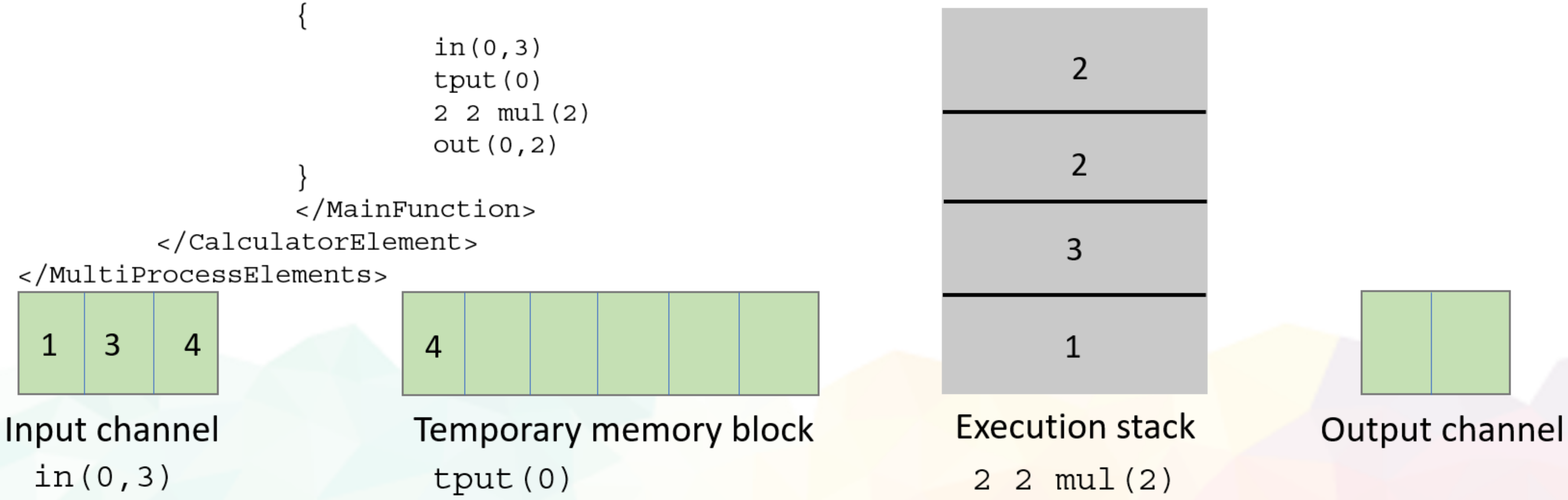
Execution stack

Output channel

TYPES OF OPERATION ENCODINGS

8. Functional vector operations

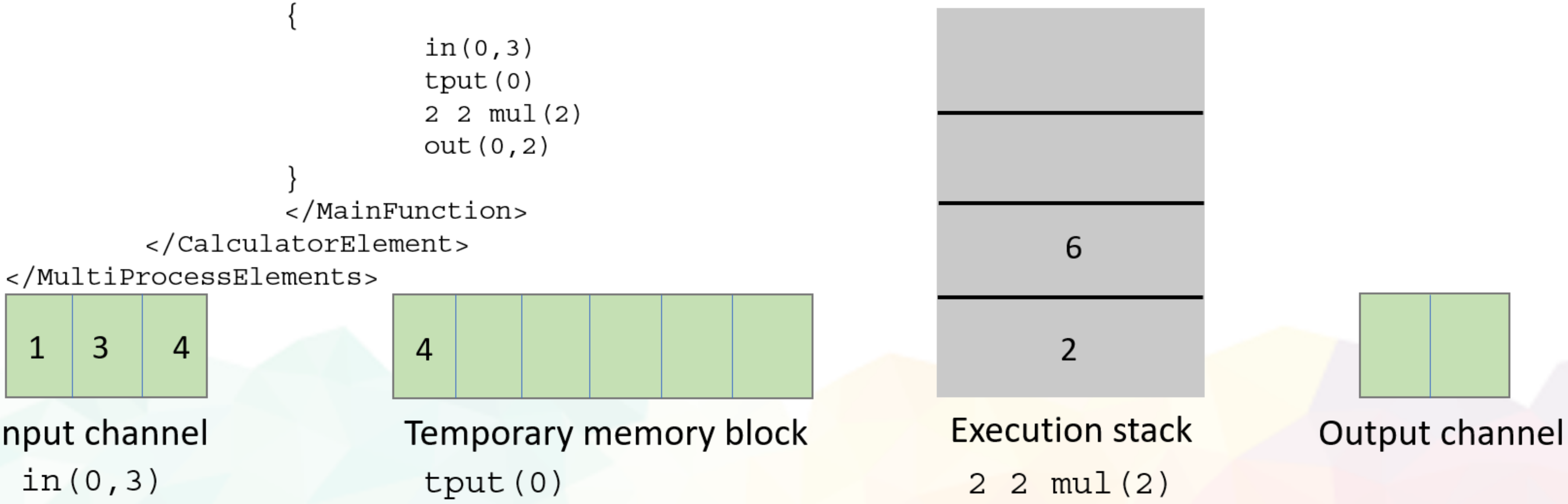
```
<MultiProcessElements InputChannels="3" OutputChannels=“2">
        <CalculatorElement InputChannels="3" OutputChannels=“2">
                <MainFunction>
                {
                        in(0,3)
                        tput(0)
                        2 2 mul(2)
                        out(0,2)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



TYPES OF OPERATION ENCODINGS

8. Functional vector operations

```
<MultiProcessElements InputChannels="3" OutputChannels=“2">
        <CalculatorElement InputChannels="3" OutputChannels=“2">
                <MainFunction>
                {
                        in(0,3)
                        tput(0)
                        2 2 mul(2)
                        out(0,2)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



1 3 4

Input channel

`in(0,3)`

4

Temporary memory block

`tput(0)`

6

2

Execution stack

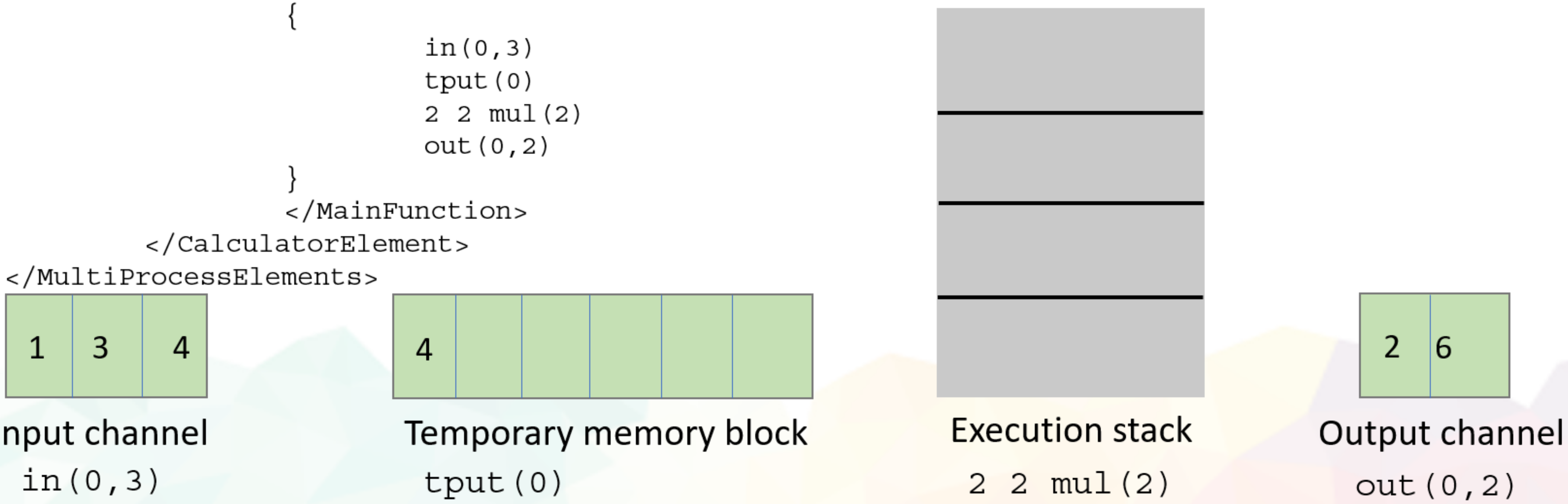
`2 2 mul(2)`

Output channel

TYPES OF OPERATION ENCODINGS

8. Functional vector operations

```
<MultiProcessElements InputChannels="3" OutputChannels=“2">
        <CalculatorElement InputChannels="3" OutputChannels=“2">
                <MainFunction>
                {
                        in(0,3)
                        tput(0)
                        2 2 mul(2)
                        out(0,2)
                }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```



1	3	4

Input channel

`in(0,3)`

4					

Temporary memory block

`tput(0)`

Execution stack

`2 2 mul(2)`

2	6

Output channel

`out(0,2)`

TYPES OF OPERATION ENCODINGS

9. Conditional operations

```
<MultiProcessElements InputChannels=“4" OutputChannels=“2">
         <CalculatorElement InputChannels=“4" OutputChannels=“2">
                  <MainFunction>
                  {

                     in(0) 0 gt if {
                     in(0,4) mul(2)
                     }

                     out(0,2)
                  }
                  </MainFunction>
         </CalculatorElement>
</MultiProcessElements>
```

EXTENDED STRUCTURES

1. IMPORTS

```
<MultiProcessElements InputChannels=“3"
OutputChannels=“3">
<Imports>
<Import Filename=”importFileRGB2XYZ.xml”/>
</Imports>
        <CalculatorElement InputChannels=“3"
OutputChannels=“3">
                <MainFunction>
                {
                   in(0,3)
                   2.1991875 gama(3)
                   mtx{RGB2XYZ}
                   out(0,3)
                }
                </MainFunction>

        </CalculatorElement>
</MultiProcessElements>
```

2. VARIABLES

```
<MultiProcessElements InputChannels=“3"
OutputChannels=“1">
<Variables>
<Declare Name=”myVar”/>
<Declare Name=”myVector” Size=6/>
<Declare Name=”myStruct”>m1 m2[3] m3</Declare>
</Variables>
<CalculatorElement InputChannels=“3"
OutputChannels=“1">
                <MainFunction>
                {
                   in(0,3)
                   tget{myvar}
                   tput{myVector(4,2)}
                   tsav{myStruct.m3}
                   out(0)
                 }
                </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```

EXTENDED STRUCTURES

3. MACROS

```
<MultiProcessElements InputChannels=“1"
OutputChannels=“1">
<Macros>
<Macro Name=”odd”>1 3 5 5 3 1</Macro>
<Macro Name=”evenoddeven”>2 4 6 call{odd} 6 4 2
</Macro>
</Macros>
<CalculatorElement InputChannels=“3"
OutputChannels=“3">
                 <MainFunction>
                 {
                    in(0)
                    call{evenoddeven}sum(13)
                    out(0)
                 }
                 </MainFunction>

        </CalculatorElement>
</MultiProcessElements>
```

4. SUB ELEMENTS

```
<MultiProcessElements InputChannels=“3"
OutputChannels=“3">
<SubElements>
<CurveSetElement Name=”applyGamma” InputChannels=”3”
OutputChannels=”3”>…</CurveSetElement>
<MatrixElement Name=”RGBtoXYZ” InputChannels=”3”
OutputChannels=”3”>…</MatrixElement>
</SubElements>
<CalculatorElement InputChannels=“3"
OutputChannels=“3">
                 <MainFunction>
                 {
                    in(0,3)
                    curv{applyGamma}
                    mtx{RGBtoXYZ}
                    out(0,3)
                 }
                 </MainFunction>
        </CalculatorElement>
</MultiProcessElements>
```

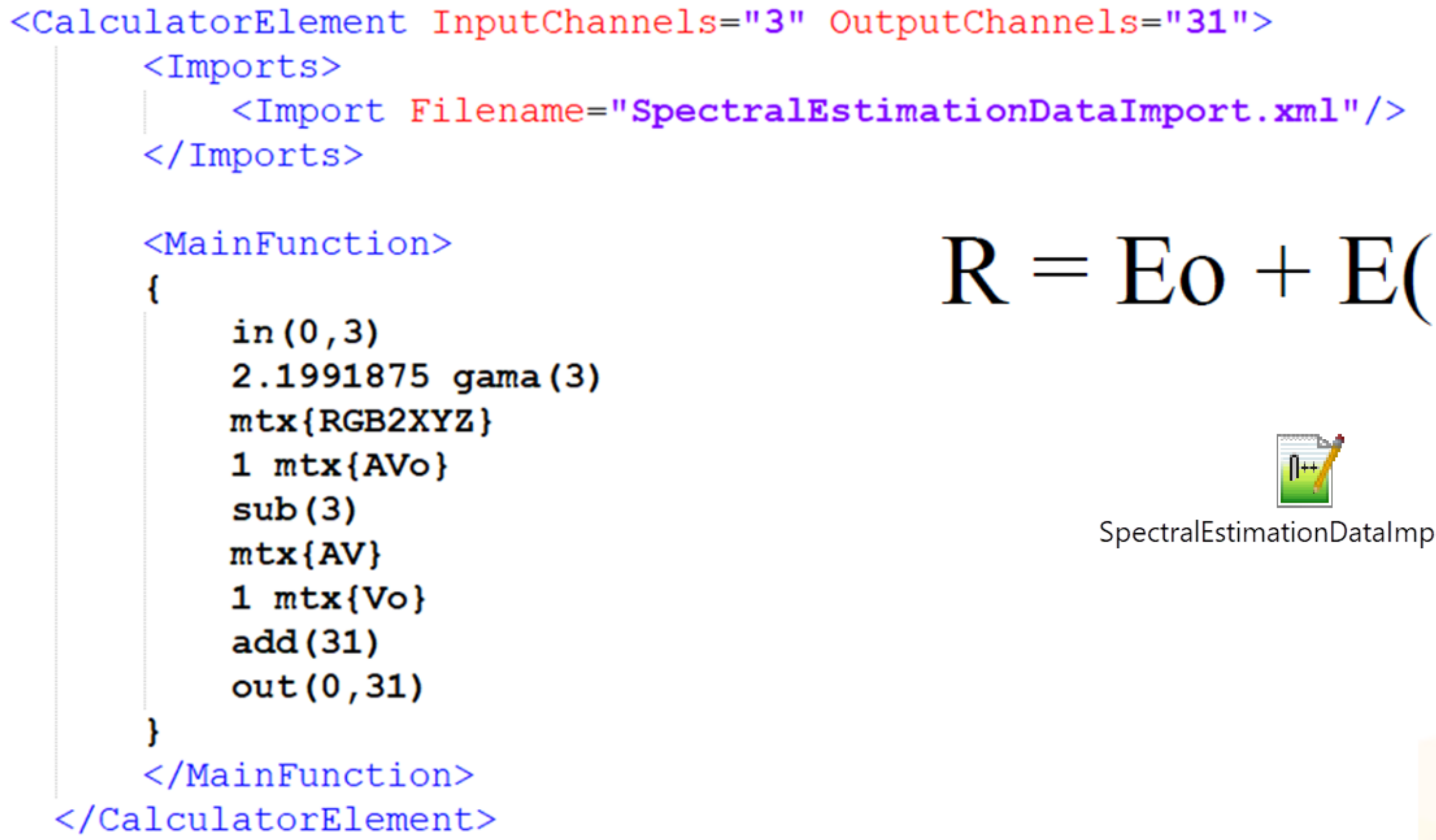
EXAMPLE 1: SPECTRAL ESTIMATION

1. Spectral Estimation using classical PCA [Fairman and Brill]

$$R = \underbrace{Eo}_{\text{Average Training Reflectance}} + \underbrace{E}_{\text{Principal Components}}\underbrace{((A^{T} E)^{-1}\underbrace{(T - A^{T}Eo)}_{\text{Mean Centred}})}_{\text{Coefficients}}$$

- **E** – the three principal components corresponding to the first three highest number of eigen values.
- **Eo** – the average spectral reflectance of the training data
- **T = A'R -------------------- (1)**
- **EC = R – Eo ------------------(2)**
- **A -** weight set for tristimulus integration
- **C –** coefficients of principal components, **R -** reflectance

EXAMPLE 1: SPECTRAL ESTIMATION

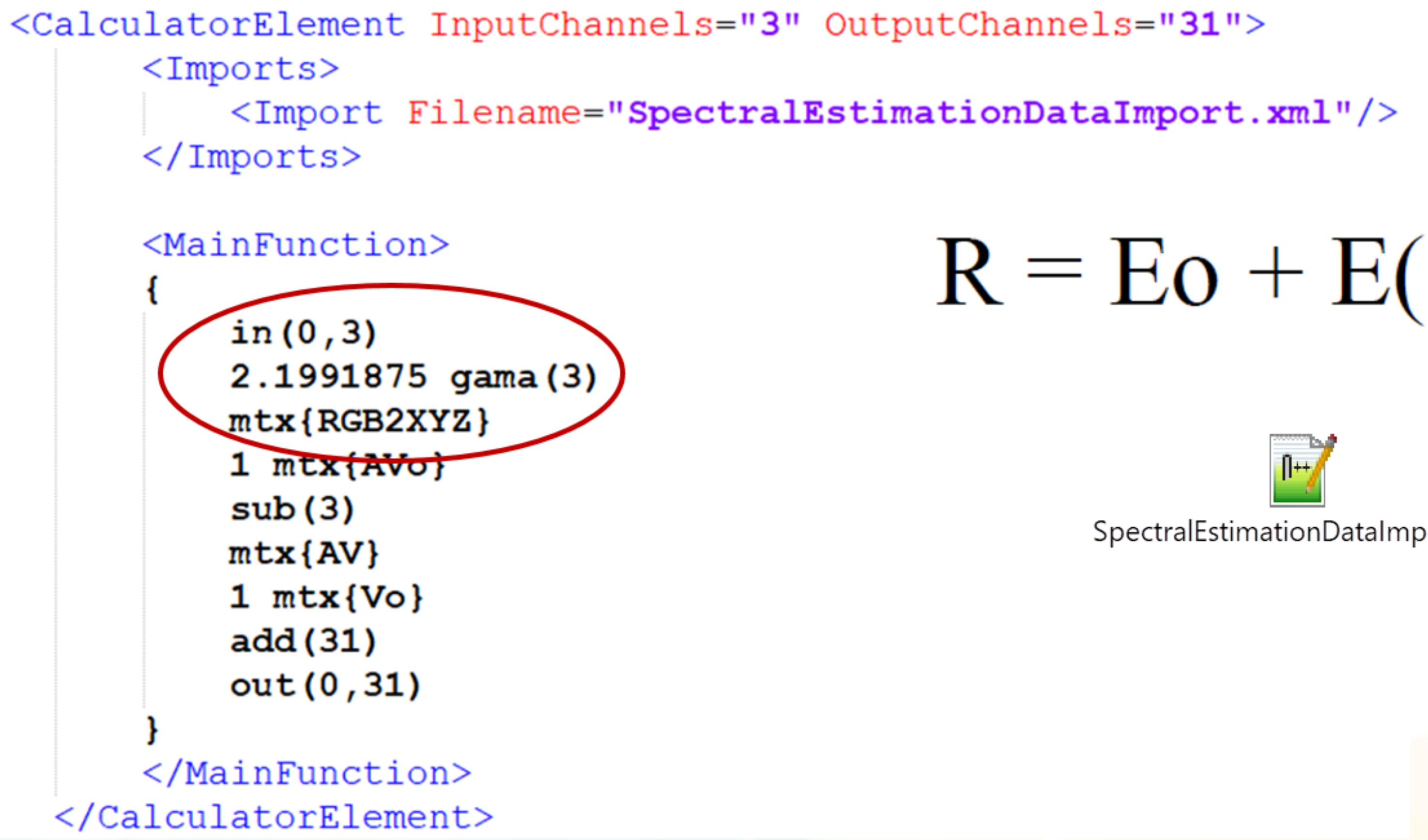


```
<CalculatorElement InputChannels="3" OutputChannels="31">
    <Imports>
        <Import Filename="SpectralEstimationDataImport.xml"/>
    </Imports>

    <MainFunction>
    {
        in(0,3)
        2.1991875 gama(3)
        mtx{RGB2XYZ}
        1 mtx{AVo}
        sub(3)
        mtx{AV}
        1 mtx{Vo}
        add(31)
        out(0,31)
    }
    </MainFunction>
</CalculatorElement>
```

$$R = Eo + E((A^T E)^{-1}(T - A^T Eo))$$

SpectralEstimationDataImport.xml

EXAMPLE 1: SPECTRAL ESTIMATION

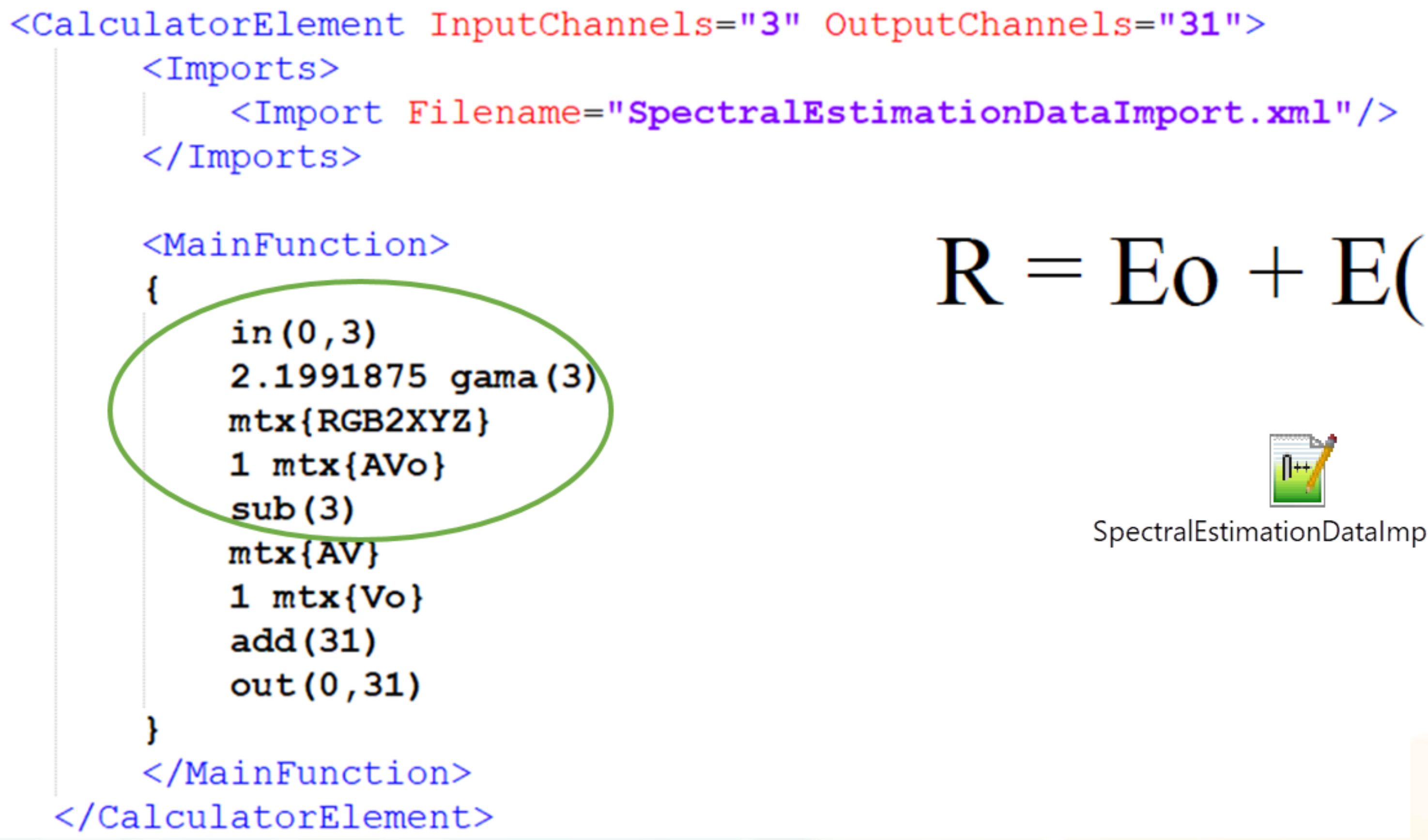


```
<CalculatorElement InputChannels="3" OutputChannels="31">
      <Imports>
          <Import Filename="SpectralEstimationDataImport.xml"/>
      </Imports>

      <MainFunction>
      {
          in(0,3)
          2.1991875 gama(3)
          mtx{RGB2XYZ}
          1 mtx{AVo}
          sub(3)
          mtx{AV}
          1 mtx{Vo}
          add(31)
          out(0,31)
      }
      </MainFunction>
  </CalculatorElement>
```

$$R = Eo + E((A^T E)^{-1}(T - A^T Eo))$$

SpectralEstimationDataImport.xml

EXAMPLE 1: SPECTRAL ESTIMATION

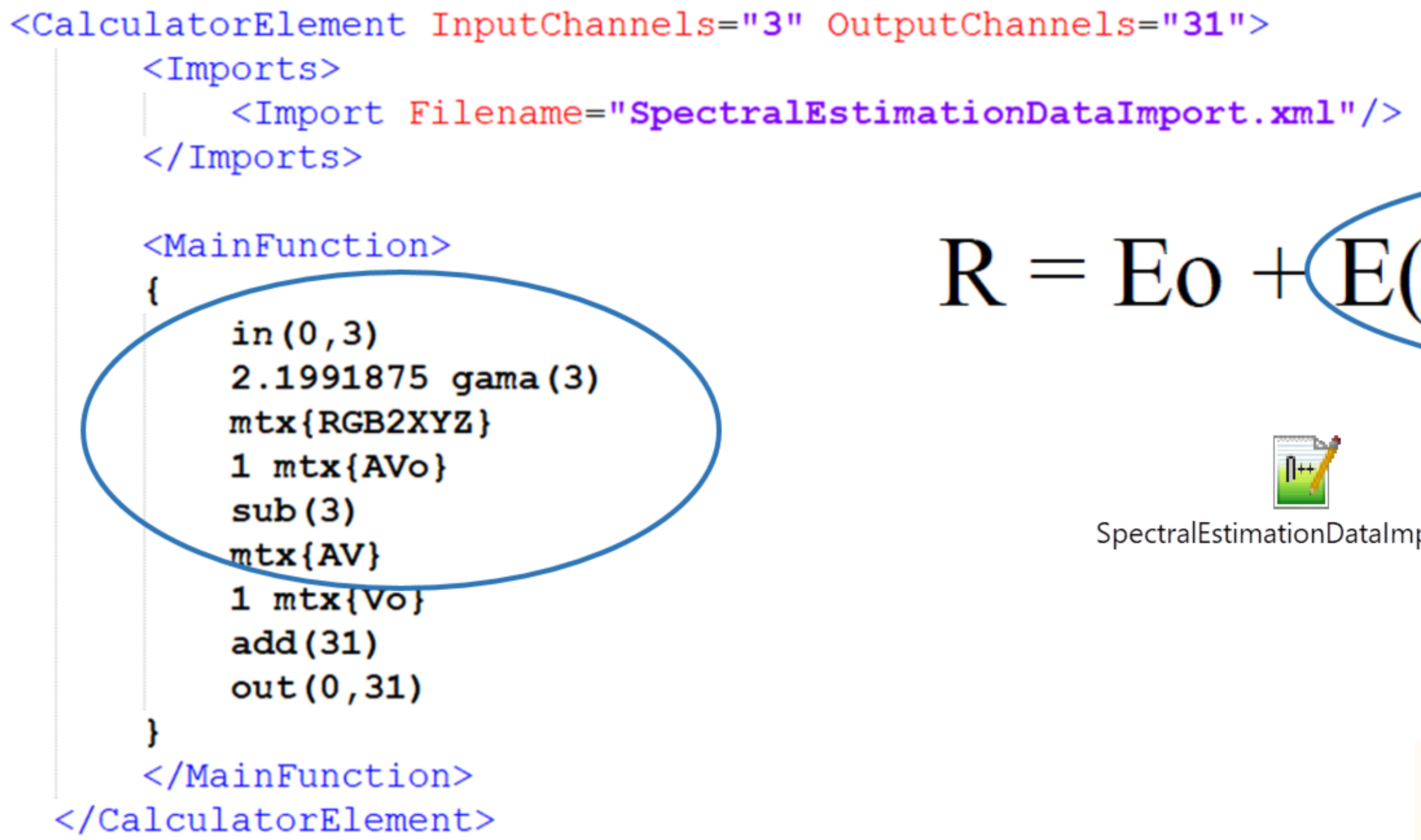


```
<CalculatorElement InputChannels="3" OutputChannels="31">
     <Imports>
         <Import Filename="SpectralEstimationDataImport.xml"/>
     </Imports>

     <MainFunction>
     {
         in(0,3)
         2.1991875 gama(3)
         mtx{RGB2XYZ}
         1 mtx{AVo}
         sub(3)
         mtx{AV}
         1 mtx{Vo}
         add(31)
         out(0,31)
     }
     </MainFunction>
  </CalculatorElement>
```

$$R = Eo + E((A^T E)^{-1}(T - A^T Eo))$$

SpectralEstimationDataImport.xml

EXAMPLE 1: SPECTRAL ESTIMATION

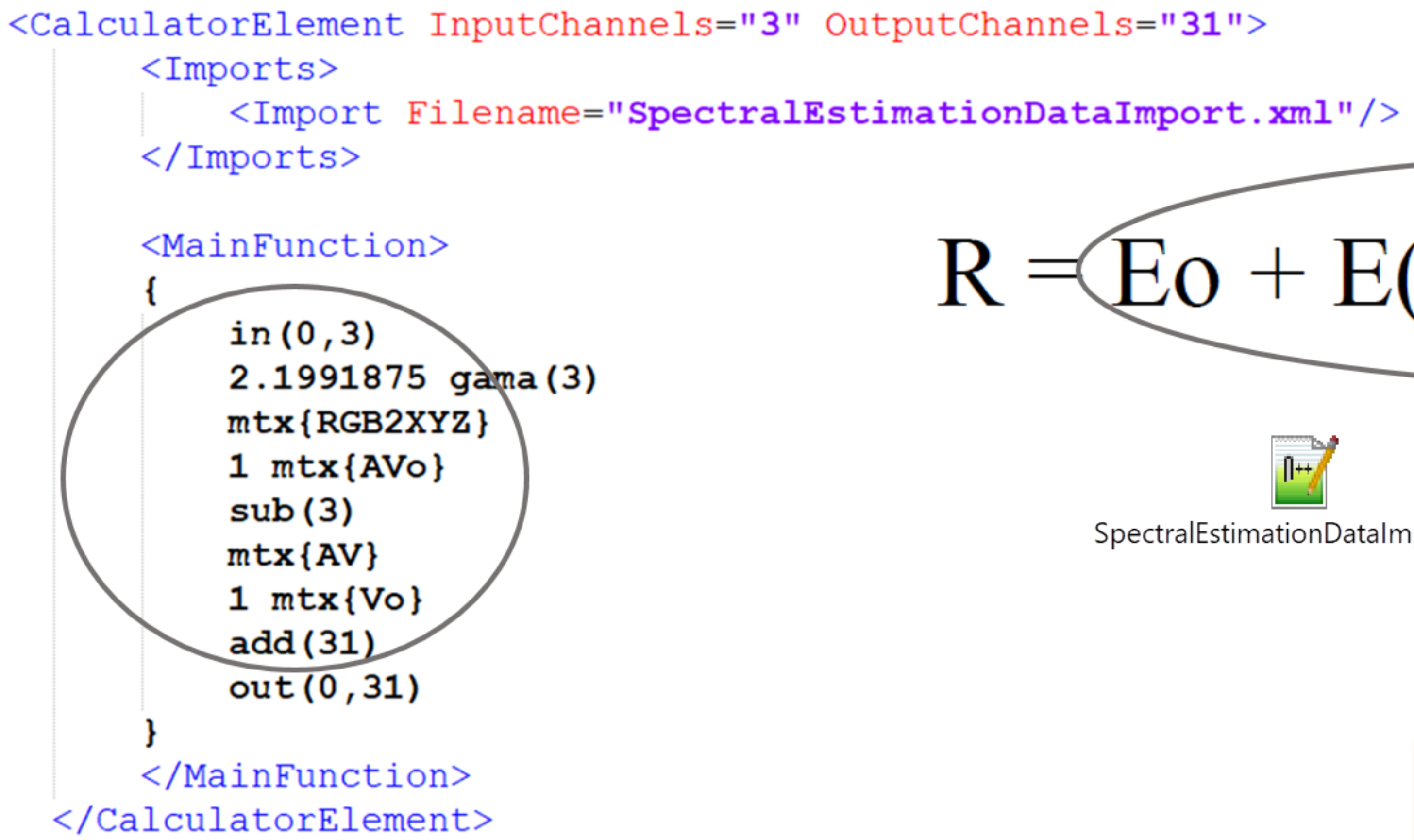


```
<CalculatorElement InputChannels="3" OutputChannels="31">
    <Imports>
        <Import Filename="SpectralEstimationDataImport.xml"/>
    </Imports>

    <MainFunction>
    {
        in(0,3)
        2.1991875 gama(3)
        mtx{RGB2XYZ}
        1 mtx{AVo}
        sub(3)
        mtx{AV}
        1 mtx{Vo}
        add(31)
        out(0,31)
    }
    </MainFunction>
</CalculatorElement>
```

$$R = Eo + E((A^T E)^{-1}(T - A^T Eo))$$

SpectralEstimationDataImport.xml

EXAMPLE 1: SPECTRAL ESTIMATION

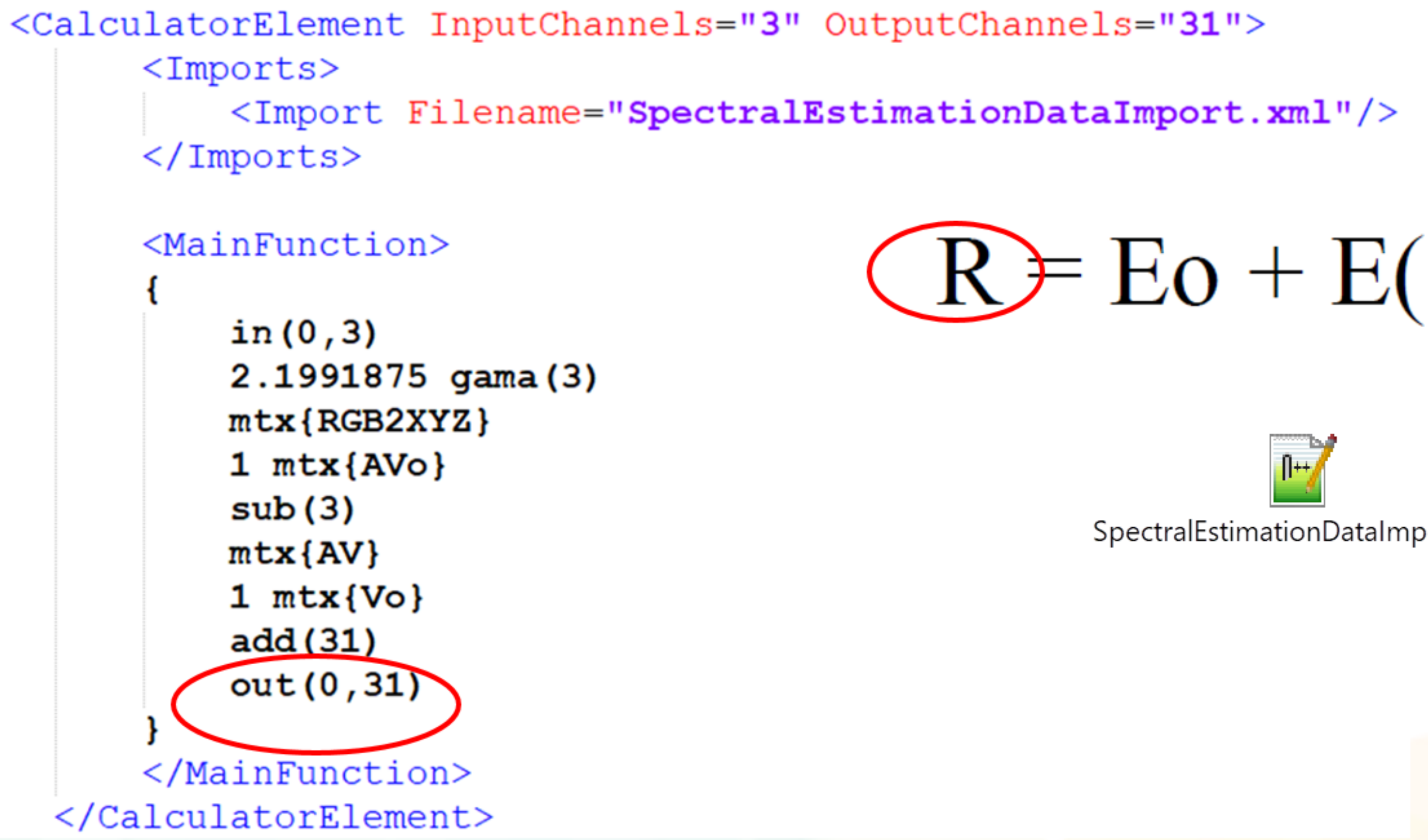


```
<CalculatorElement InputChannels="3" OutputChannels="31">
      <Imports>
          <Import Filename="SpectralEstimationDataImport.xml"/>
      </Imports>

      <MainFunction>
      {
          in(0,3)
          2.1991875 gama(3)
          mtx{RGB2XYZ}
          1 mtx{AVo}
          sub(3)
          mtx{AV}
          1 mtx{Vo}
          add(31)
          out(0,31)
      }
      </MainFunction>
  </CalculatorElement>
```

$$R = Eo + E((A^T E)^{-1}(T - A^T Eo))$$

SpectralEstimationDataImport.xml

EXAMPLE 1: SPECTRAL ESTIMATION

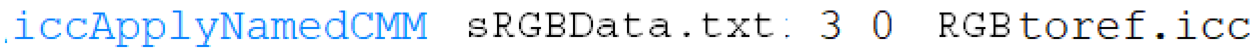


```
<CalculatorElement InputChannels="3" OutputChannels="31">
      <Imports>
          <Import Filename="SpectralEstimationDataImport.xml"/>
      </Imports>

      <MainFunction>
      {
          in(0,3)
          2.1991875 gama(3)
          mtx{RGB2XYZ}
          1 mtx{AVo}
          sub(3)
          mtx{AV}
          1 mtx{Vo}
          add(31)
          out(0,31)
      }
      </MainFunction>
  </CalculatorElement>
```

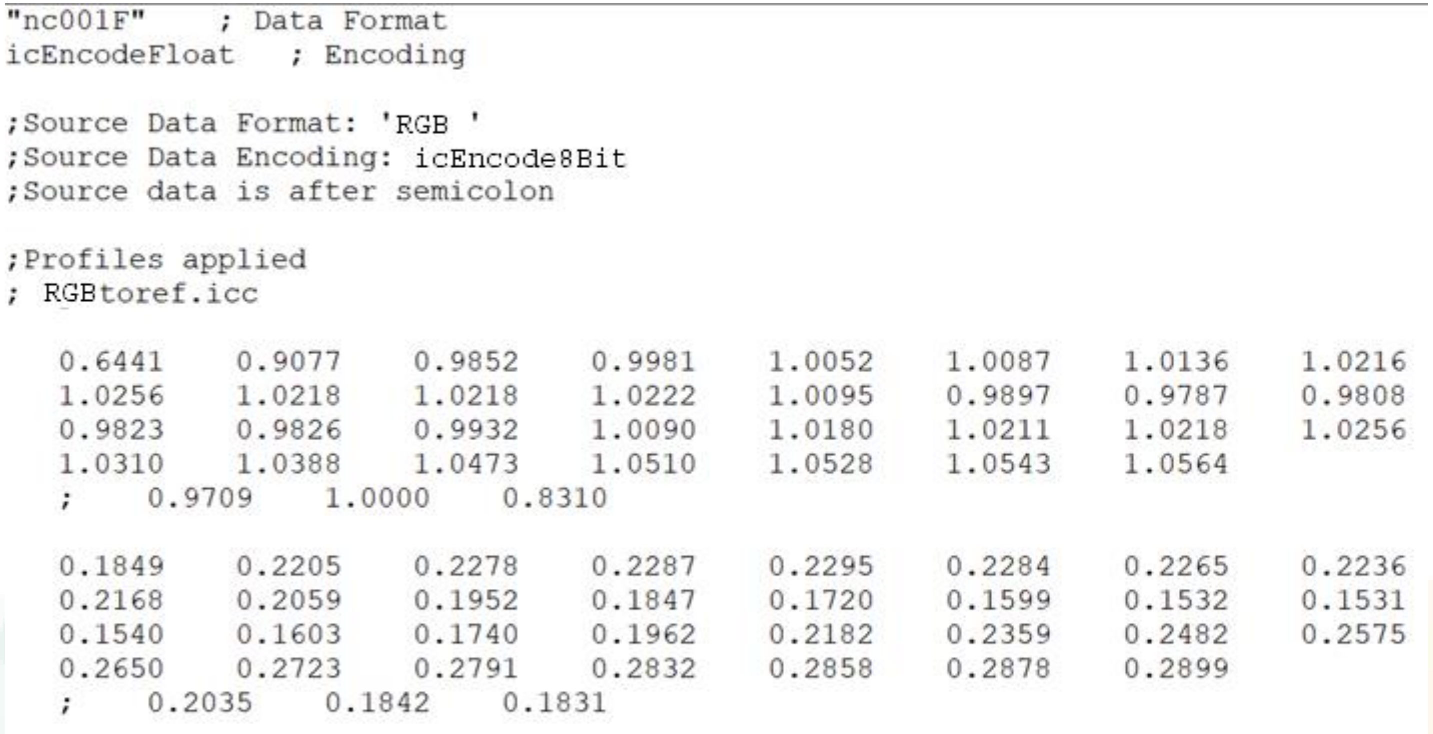
$$R = Eo + E((A^{T} E)^{-1}(T - A^{T}Eo))$$

SpectralEstimationDataImport.xml

EXAMPLE 1: SPECTRAL ESTIMATION



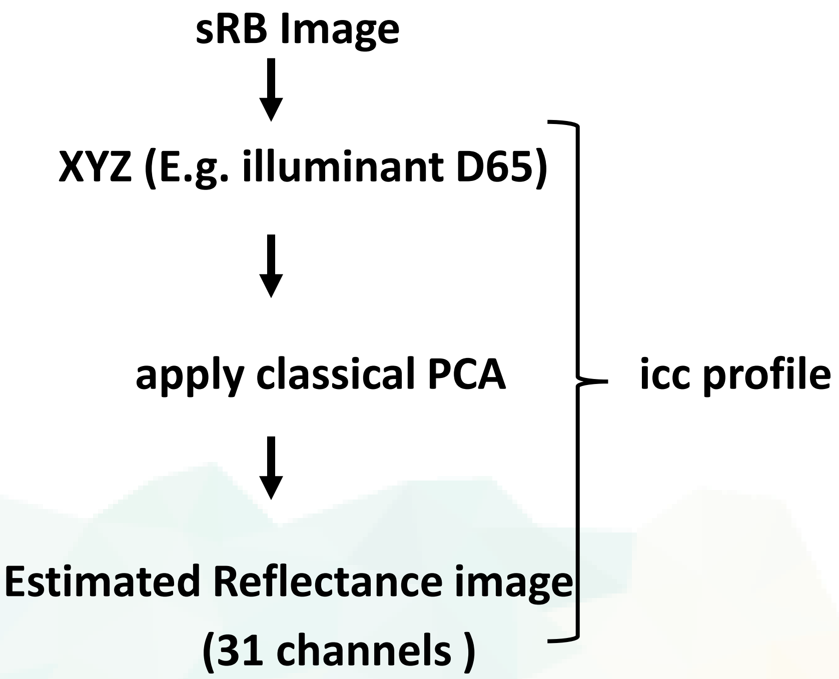
```
iccApplyNamedCMM sRGBData.txt 3 0 RGBtoref.icc
```



```
"nc001F"     ; Data Format
icEncodeFloat    ; Encoding

;Source Data Format: 'RGB '
;Source Data Encoding: icEncode8Bit
;Source data is after semicolon

;Profiles applied
; RGBtoref.icc

   0.6441     0.9077     0.9852     0.9981     1.0052     1.0087     1.0136     1.0216
   1.0256     1.0218     1.0218     1.0222     1.0095     0.9897     0.9787     0.9808
   0.9823     0.9826     0.9932     1.0090     1.0180     1.0211     1.0218     1.0256
   1.0310     1.0388     1.0473     1.0510     1.0528     1.0543     1.0564
   ;     0.9709     1.0000     0.8310

   0.1849     0.2205     0.2278     0.2287     0.2295     0.2284     0.2265     0.2236
   0.2168     0.2059     0.1952     0.1847     0.1720     0.1599     0.1532     0.1531
   0.1540     0.1603     0.1740     0.1962     0.2182     0.2359     0.2482     0.2575
   0.2650     0.2723     0.2791     0.2832     0.2858     0.2878     0.2899
   ;     0.2035     0.1842     0.1831
```

EXAMPLE 1: SPECTRAL ESTIMATION

```
iccApplyProfiles RGB_D65_2deg.tif  D65_2deg_Reflectance.tif 2 0 1 0 1
argb2xyzD65toref.icc 3
```



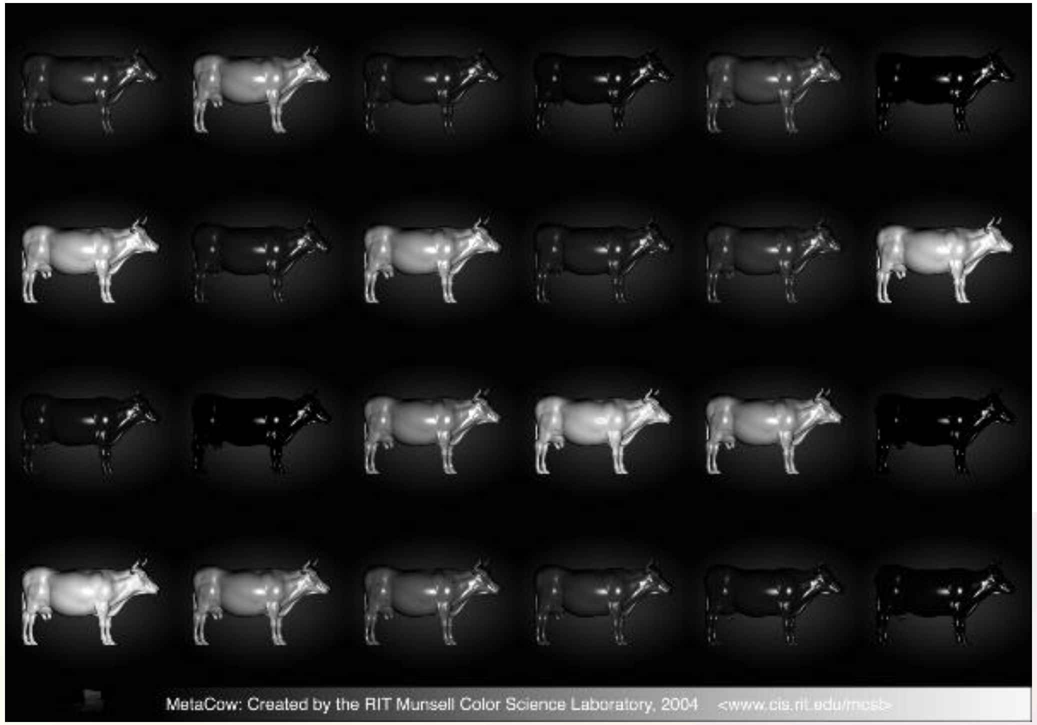
sRB Image

XYZ (E.g. illuminant D65)

apply classical PCA

Estimated Reflectance image
(31 channels)

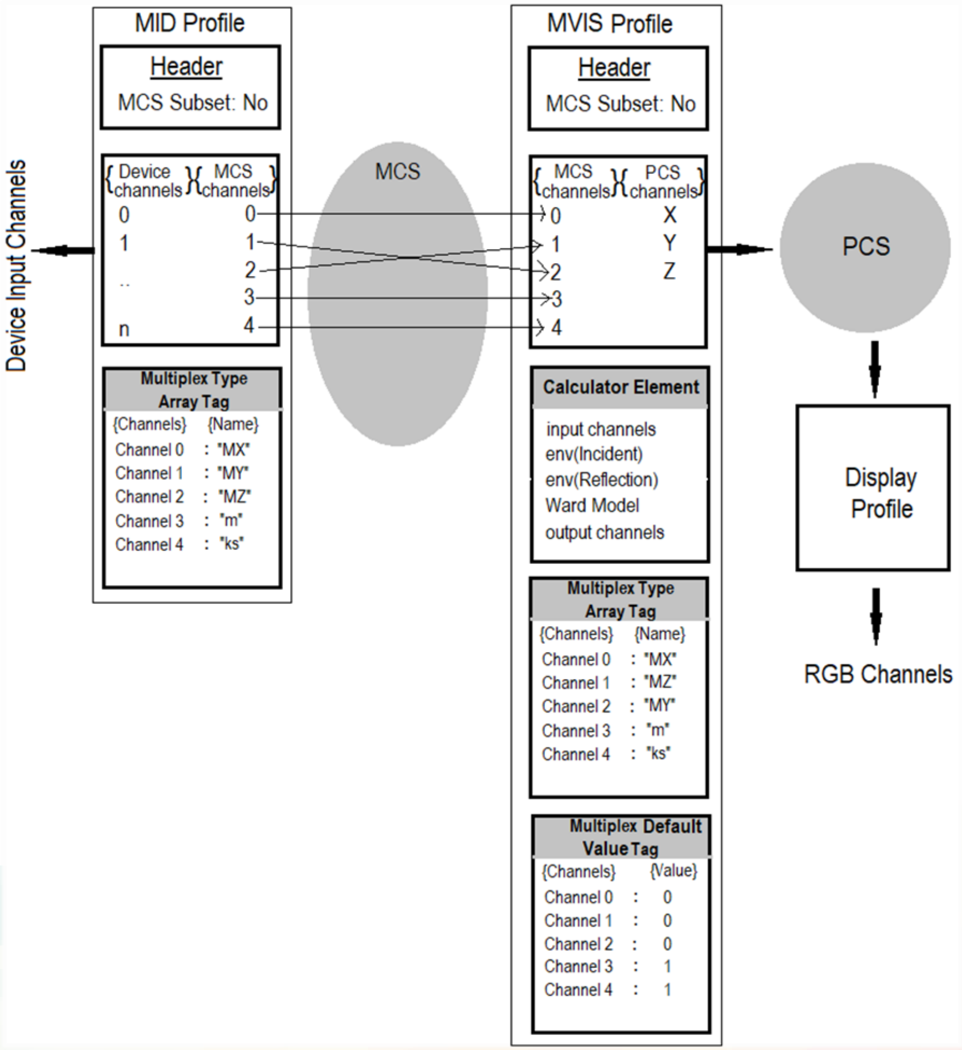
icc profile



MetaCow: Created by the RIT Munsell Color Science Laboratory, 2004 <www.cis.rit.edu/mcsl>

EXAMPLE 2: BRDF IMPLEMENTATION

- Input – TIFF file with BRDF coefficients
- An MID profile to read input and pass to the MCS
- An MVIS profile to use MCS as input and apply the encoded BRF model
- The incidence and viewing angles supplied at runtime
- Output – TIFF containing XYZ values at a new geometry



Multiplex Type Array Tag

{Channels}	{Name}
Channel 0	: "MX"
Channel 1	: "MY"
Channel 2	: "MZ"
Channel 3	: "m"
Channel 4	: "ks"

Calculator Element

input channels
env(Incident)
env(Reflection)
Ward Model
output channels

Multiplex Type Array Tag

{Channels}	{Name}
Channel 0	: "MX"
Channel 1	: "MZ"
Channel 2	: "MY"
Channel 3	: "m"
Channel 4	: "ks"

Multiplex Default Value Tag

{Channels}	{Value}
Channel 0	: 0
Channel 1	: 0
Channel 2	: 0
Channel 3	: 1
Channel 4	: 1

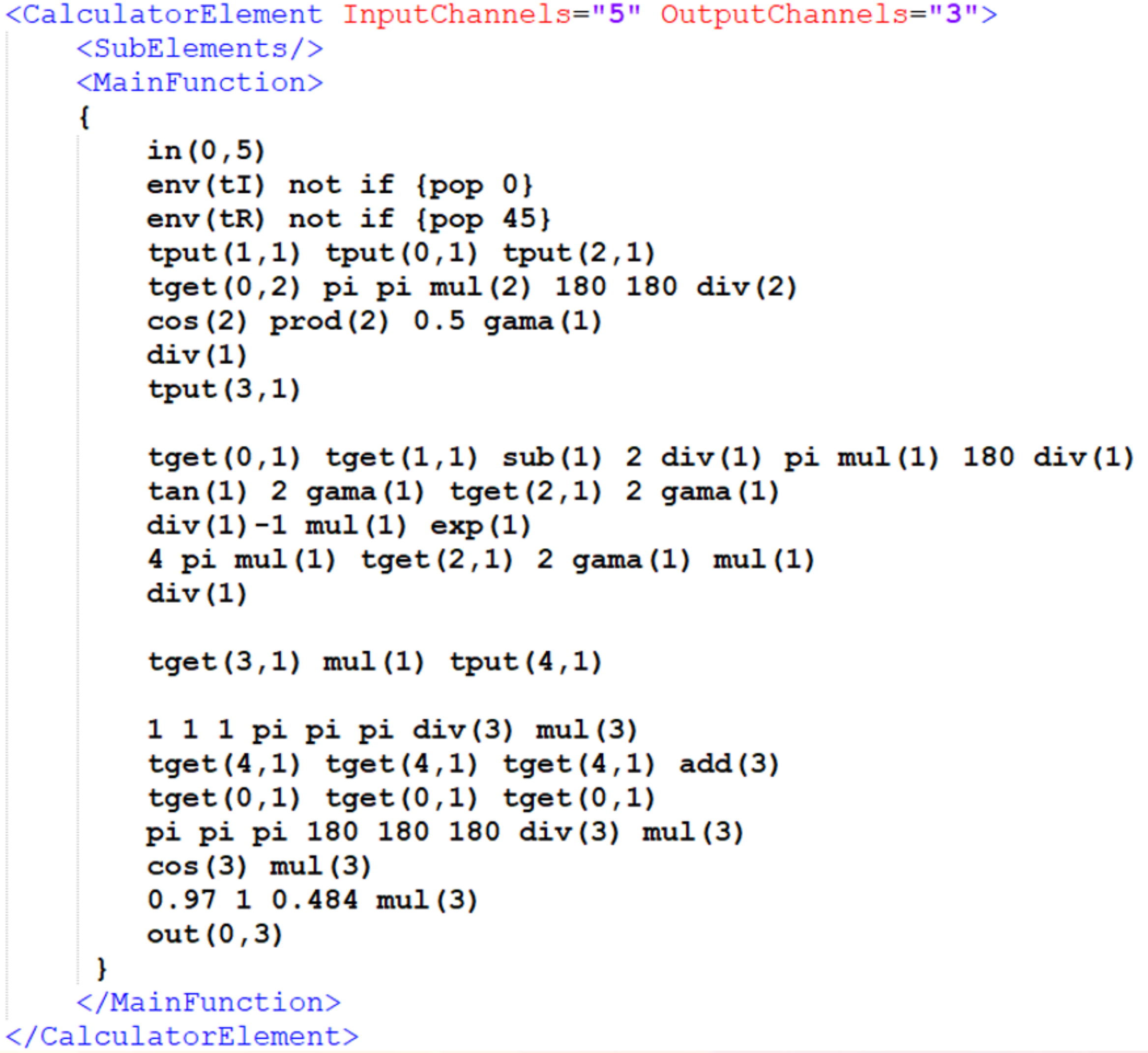
EXAMPLE 2: BRDF IMPLEMENTATION

Ward BRDF

$$I_p(\theta_i;\theta_r) = \begin{bmatrix} I_{pX}\\ I_{pY}\\ I_{pZ}\end{bmatrix} = I_i\cos\theta_i\left(\begin{bmatrix}R_{dX}\\ R_{dY}\\ R_{dZ}\end{bmatrix}\frac{1}{\pi} + \frac{k_S}{\sqrt{\cos\theta_i\cos\theta_r}}\frac{e^{[-\tan^2\delta/m^2]}}{4\pi m^2}\right)$$

Input

$$\begin{bmatrix}R_{dX}\\ R_{dY}\\ R_{dZ}\end{bmatrix} \quad k_S \quad m$$



```
<CalculatorElement InputChannels="5" OutputChannels="3">
    <SubElements/>
    <MainFunction>
    {
        in(0,5)
        env(tI) not if {pop 0}
        env(tR) not if {pop 45}
        tput(1,1) tput(0,1) tput(2,1)
        tget(0,2) pi pi mul(2) 180 180 div(2)
        cos(2) prod(2) 0.5 gama(1)
        div(1)
        tput(3,1)

        tget(0,1) tget(1,1) sub(1) 2 div(1) pi mul(1) 180 div(1)
        tan(1) 2 gama(1) tget(2,1) 2 gama(1)
        div(1)-1 mul(1) exp(1)
        4 pi mul(1) tget(2,1) 2 gama(1) mul(1)
        div(1)

        tget(3,1) mul(1) tput(4,1)

        1 1 1 pi pi pi div(3) mul(3)
        tget(4,1) tget(4,1) tget(4,1) add(3)
        tget(0,1) tget(0,1) tget(0,1)
        pi pi pi 180 180 180 div(3) mul(3)
        cos(3) mul(3)
        0.97 1 0.484 mul(3)
        out(0,3)
     }
    </MainFunction>
</CalculatorElement>
```

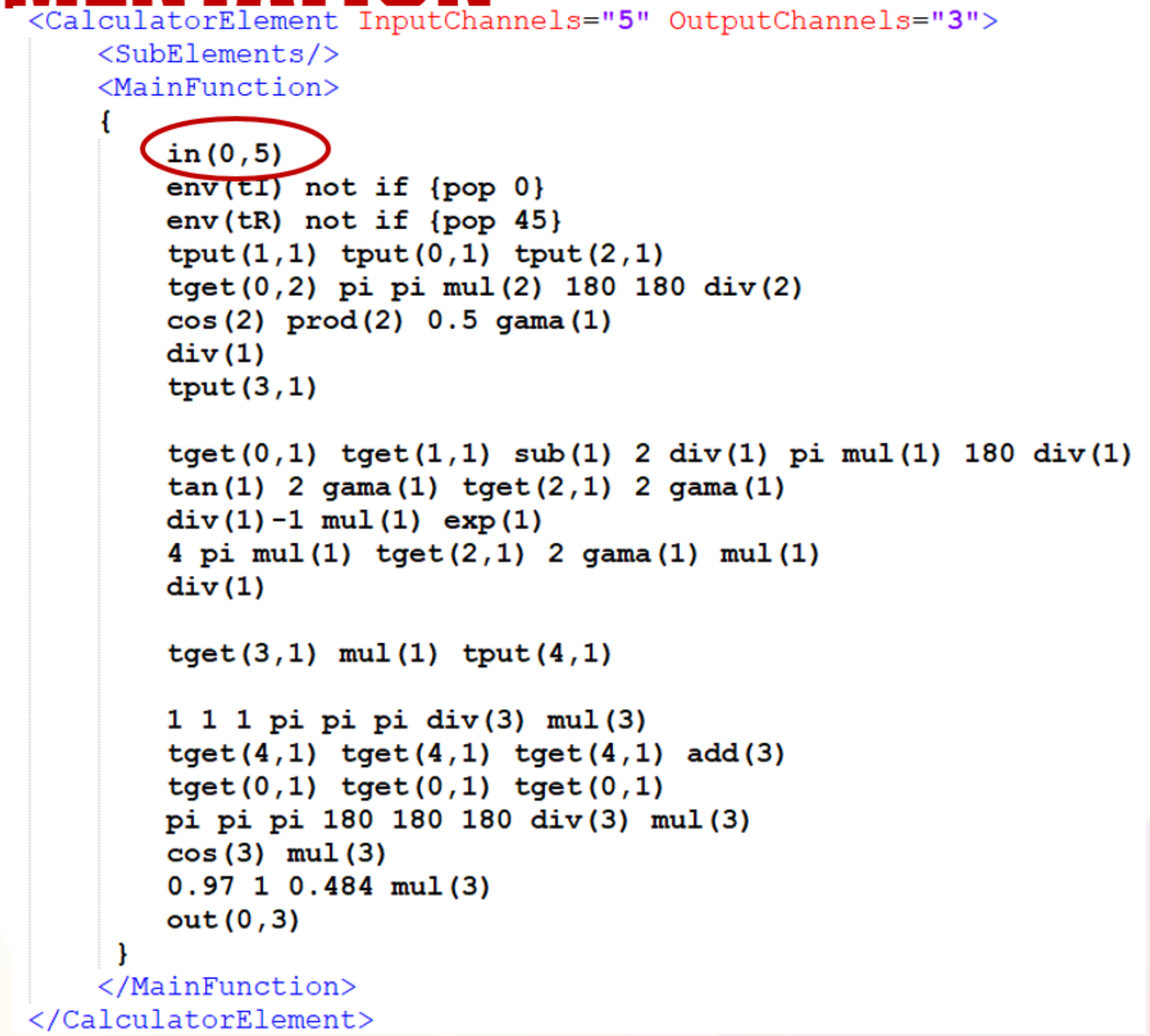
EXAMPLE 2: BRDF IMPLEMENTATION

Ward BRDF

$$I_p(\theta_i;\theta_r) = \begin{bmatrix} I_{pX} \\ I_{pY} \\ I_{pZ} \end{bmatrix} = I_i \cos\theta_i \left(\begin{bmatrix} R_{dX} \\ R_{dY} \\ R_{dZ} \end{bmatrix} \frac{1}{\pi} + \frac{k_S}{\sqrt{\cos\theta_i \cos\theta_r}} \frac{e^{[-\tan^2\delta/m^2]}}{4\pi m^2} \right)$$

Input

$$\begin{bmatrix} R_{dX} \\ R_{dY} \\ R_{dZ} \end{bmatrix} \quad k_S \quad m$$



```
<CalculatorElement InputChannels="5" OutputChannels="3">
    <SubElements/>
    <MainFunction>
    {
        in(0,5)
        env(tI) not if {pop 0}
        env(tR) not if {pop 45}
        tput(1,1) tput(0,1) tput(2,1)
        tget(0,2) pi pi mul(2) 180 180 div(2)
        cos(2) prod(2) 0.5 gama(1)
        div(1)
        tput(3,1)

        tget(0,1) tget(1,1) sub(1) 2 div(1) pi mul(1) 180 div(1)
        tan(1) 2 gama(1) tget(2,1) 2 gama(1)
        div(1)-1 mul(1) exp(1)
        4 pi mul(1) tget(2,1) 2 gama(1) mul(1)
        div(1)

        tget(3,1) mul(1) tput(4,1)

        1 1 1 pi pi pi div(3) mul(3)
        tget(4,1) tget(4,1) tget(4,1) add(3)
        tget(0,1) tget(0,1) tget(0,1)
        pi pi pi 180 180 180 div(3) mul(3)
        cos(3) mul(3)
        0.97 1 0.484 mul(3)
        out(0,3)
     }
    </MainFunction>
</CalculatorElement>
```

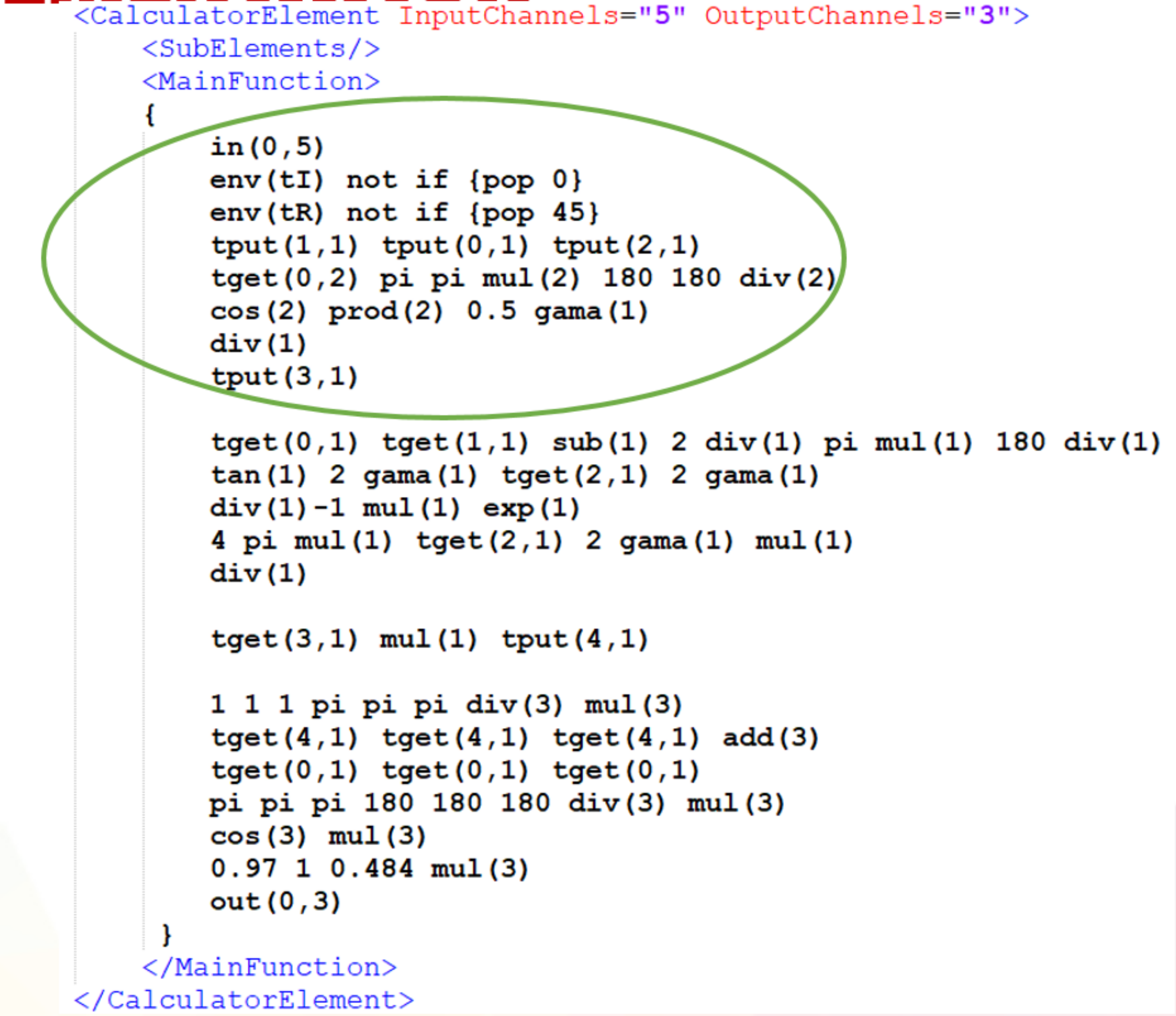
EXAMPLE 2: BRDF IMPLEMENTATION

Ward BRDF

$$I_p(\theta_i;\theta_r) = \begin{bmatrix} I_{pX} \\ I_{pY} \\ I_{pZ} \end{bmatrix} = I_i \cos\theta_i \left(\begin{bmatrix} R_{dX} \\ R_{dY} \\ R_{dZ} \end{bmatrix} \frac{1}{\pi} + \frac{k_S}{\sqrt{\cos\theta_i \cos\theta_r}} \frac{e^{[-\tan^2\delta/m^2]}}{4\pi m^2} \right)$$

Input

$$\begin{bmatrix} R_{dX} \\ R_{dY} \\ R_{dZ} \end{bmatrix} \quad k_S \quad m$$



```
<CalculatorElement InputChannels="5" OutputChannels="3">
    <SubElements/>
    <MainFunction>
    {
        in(0,5)
        env(tI) not if {pop 0}
        env(tR) not if {pop 45}
        tput(1,1) tput(0,1) tput(2,1)
        tget(0,2) pi pi mul(2) 180 180 div(2)
        cos(2) prod(2) 0.5 gama(1)
        div(1)
        tput(3,1)

        tget(0,1) tget(1,1) sub(1) 2 div(1) pi mul(1) 180 div(1)
        tan(1) 2 gama(1) tget(2,1) 2 gama(1)
        div(1)-1 mul(1) exp(1)
        4 pi mul(1) tget(2,1) 2 gama(1) mul(1)
        div(1)

        tget(3,1) mul(1) tput(4,1)

        1 1 1 pi pi pi div(3) mul(3)
        tget(4,1) tget(4,1) tget(4,1) add(3)
        tget(0,1) tget(0,1) tget(0,1)
        pi pi pi 180 180 180 div(3) mul(3)
        cos(3) mul(3)
        0.97 1 0.484 mul(3)
        out(0,3)
     }
    </MainFunction>
</CalculatorElement>
```

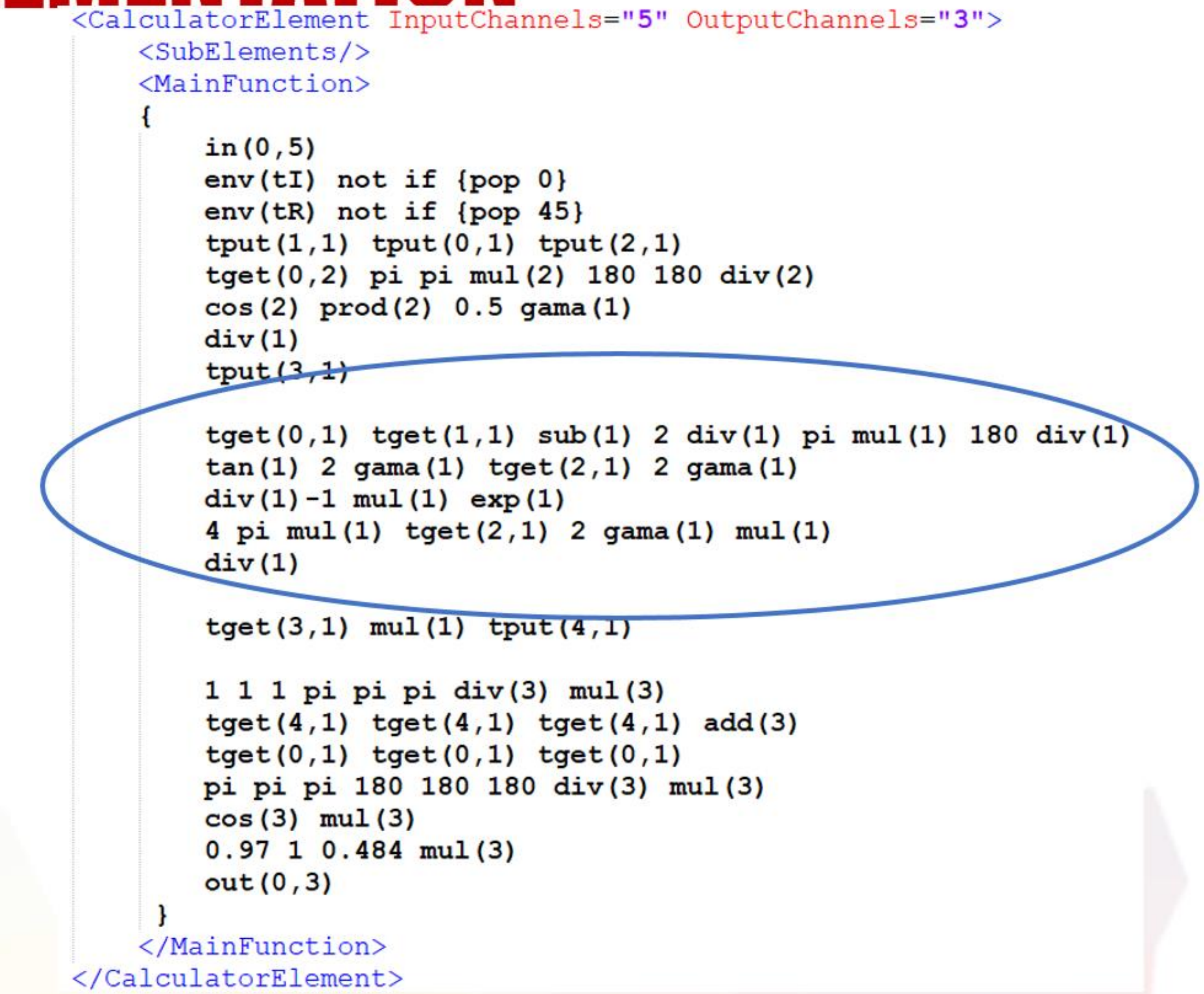
EXAMPLE 2: BRDF IMPLEMENTATION

Ward BRDF

$$I_p(\theta_i;\theta_r) = \begin{bmatrix} I_{p_X} \\ I_{p_Y} \\ I_{p_Z} \end{bmatrix} = I_i \cos\theta_i \left(\begin{bmatrix} R_{d_X} \\ R_{d_Y} \\ R_{d_Z} \end{bmatrix} \frac{1}{\pi} + \frac{k_S}{\sqrt{\cos\theta_i \cos\theta_r}} \frac{e^{[-\tan^2\delta/m^2]}}{4\pi m^2} \right)$$

Input

$$\begin{bmatrix} R_{d_X} \\ R_{d_Y} \\ R_{d_Z} \end{bmatrix} \quad k_S \quad m$$



```
<CalculatorElement InputChannels="5" OutputChannels="3">
    <SubElements/>
    <MainFunction>
    {
        in(0,5)
        env(tI) not if {pop 0}
        env(tR) not if {pop 45}
        tput(1,1) tput(0,1) tput(2,1)
        tget(0,2) pi pi mul(2) 180 180 div(2)
        cos(2) prod(2) 0.5 gama(1)
        div(1)
        tput(3,1)

        tget(0,1) tget(1,1) sub(1) 2 div(1) pi mul(1) 180 div(1)
        tan(1) 2 gama(1) tget(2,1) 2 gama(1)
        div(1)-1 mul(1) exp(1)
        4 pi mul(1) tget(2,1) 2 gama(1) mul(1)
        div(1)

        tget(3,1) mul(1) tput(4,1)

        1 1 1 pi pi pi div(3) mul(3)
        tget(4,1) tget(4,1) tget(4,1) add(3)
        tget(0,1) tget(0,1) tget(0,1)
        pi pi pi 180 180 180 div(3) mul(3)
        cos(3) mul(3)
        0.97 1 0.484 mul(3)
        out(0,3)
     }
    </MainFunction>
</CalculatorElement>
```

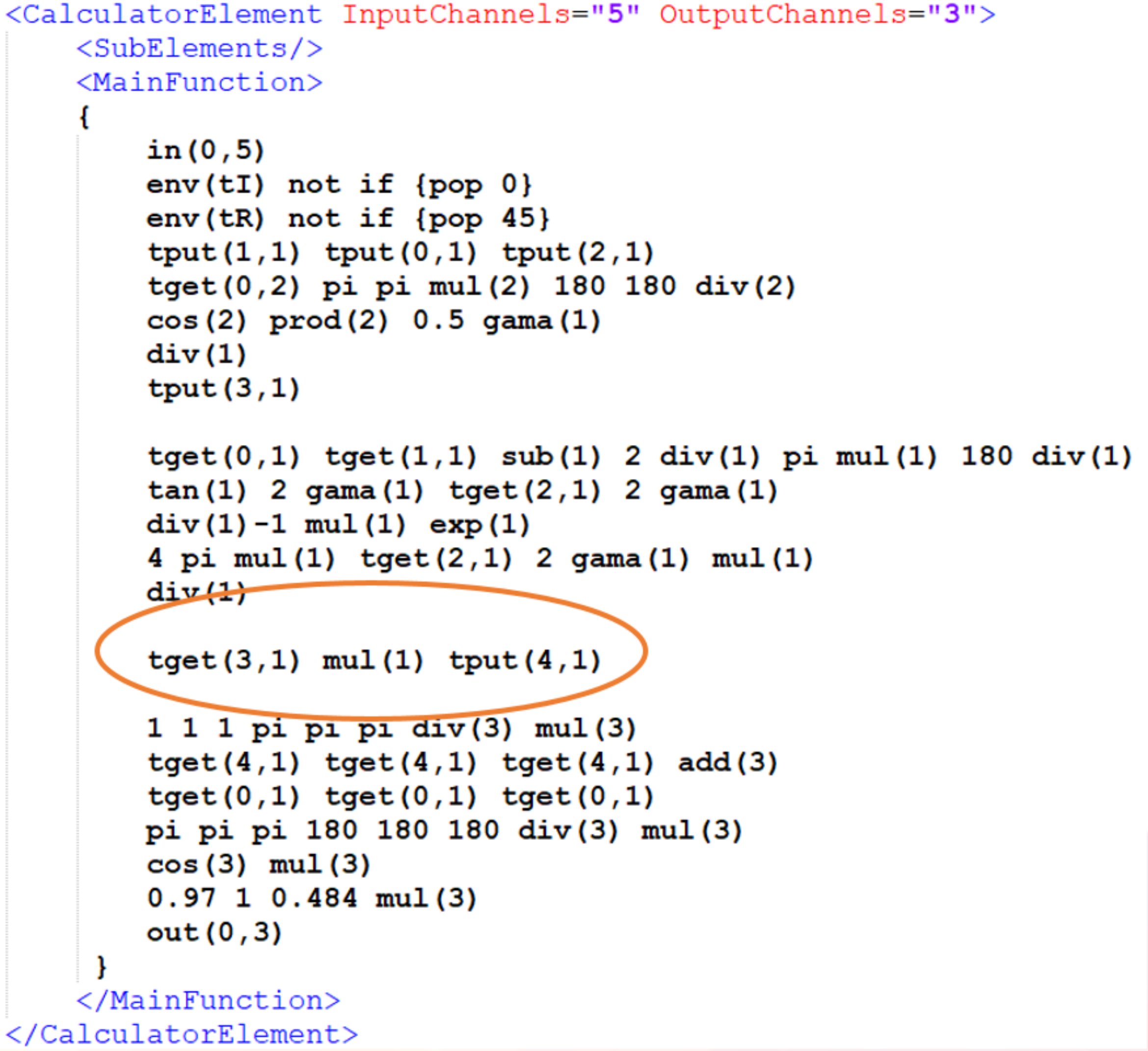
EXAMPLE 2: BRDF IMPLEMENTATION

Ward BRDF

$$I_p(\theta_i;\theta_r) = \begin{bmatrix} I_{pX} \\ I_{pY} \\ I_{pZ} \end{bmatrix} = I_i \cos\theta_i \left(\begin{bmatrix} R_{dX} \\ R_{dY} \\ R_{dZ} \end{bmatrix} \frac{1}{\pi} + \frac{k_S}{\sqrt{\cos\theta_i \cos\theta_r}} \frac{e^{[-\tan^2\delta/m^2]}}{4\pi m^2} \right)$$

Input

$$\begin{bmatrix} R_{dX} \\ R_{dY} \\ R_{dZ} \end{bmatrix} \quad k_S \quad m$$



```
<CalculatorElement InputChannels="5" OutputChannels="3">
    <SubElements/>
    <MainFunction>
    {
        in(0,5)
        env(tI) not if {pop 0}
        env(tR) not if {pop 45}
        tput(1,1) tput(0,1) tput(2,1)
        tget(0,2) pi pi mul(2) 180 180 div(2)
        cos(2) prod(2) 0.5 gama(1)
        div(1)
        tput(3,1)

        tget(0,1) tget(1,1) sub(1) 2 div(1) pi mul(1) 180 div(1)
        tan(1) 2 gama(1) tget(2,1) 2 gama(1)
        div(1)-1 mul(1) exp(1)
        4 pi mul(1) tget(2,1) 2 gama(1) mul(1)
        div(1)

        tget(3,1) mul(1) tput(4,1)

        1 1 1 pi pi pi div(3) mul(3)
        tget(4,1) tget(4,1) tget(4,1) add(3)
        tget(0,1) tget(0,1) tget(0,1)
        pi pi pi 180 180 180 div(3) mul(3)
        cos(3) mul(3)
        0.97 1 0.484 mul(3)
        out(0,3)
    }
    </MainFunction>
</CalculatorElement>
```

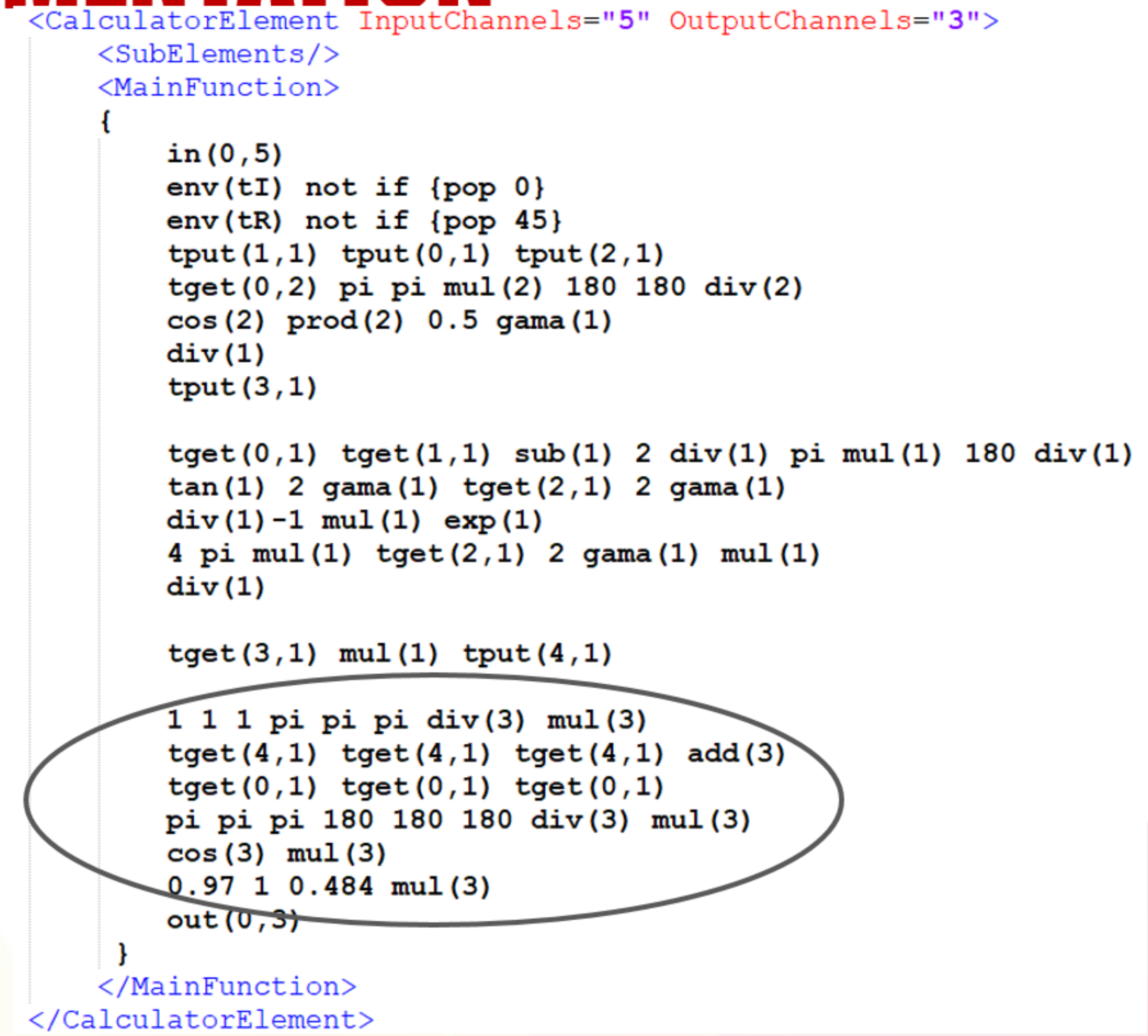
EXAMPLE 2: BRDF IMPLEMENTATION

Ward BRDF

$$I_p(\theta_i;\theta_r) = \begin{bmatrix} I_{pX}\\ I_{pY}\\ I_{pZ}\end{bmatrix} = I_i\cos\theta_i\left(\begin{bmatrix}R_{dX}\\R_{dY}\\R_{dZ}\end{bmatrix}\frac{1}{\pi} + \frac{k_S}{\sqrt{\cos\theta_i\cos\theta_r}}\frac{e^{[-\tan^2\delta/m^2]}}{4\pi m^2}\right)$$

Input

$$\begin{bmatrix}R_{dX}\\R_{dY}\\R_{dZ}\end{bmatrix} \quad k_S \quad m$$



```
<CalculatorElement InputChannels="5" OutputChannels="3">
    <SubElements/>
    <MainFunction>
    {
        in(0,5)
        env(tI) not if {pop 0}
        env(tR) not if {pop 45}
        tput(1,1) tput(0,1) tput(2,1)
        tget(0,2) pi pi mul(2) 180 180 div(2)
        cos(2) prod(2) 0.5 gama(1)
        div(1)
        tput(3,1)

        tget(0,1) tget(1,1) sub(1) 2 div(1) pi mul(1) 180 div(1)
        tan(1) 2 gama(1) tget(2,1) 2 gama(1)
        div(1)-1 mul(1) exp(1)
        4 pi mul(1) tget(2,1) 2 gama(1) mul(1)
        div(1)

        tget(3,1) mul(1) tput(4,1)

        1 1 1 pi pi pi div(3) mul(3)
        tget(4,1) tget(4,1) tget(4,1) add(3)
        tget(0,1) tget(0,1) tget(0,1)
        pi pi pi 180 180 180 div(3) mul(3)
        cos(3) mul(3)
        0.97 1 0.484 mul(3)
        out(0,3)
    }
    </MainFunction>
</CalculatorElement>
```

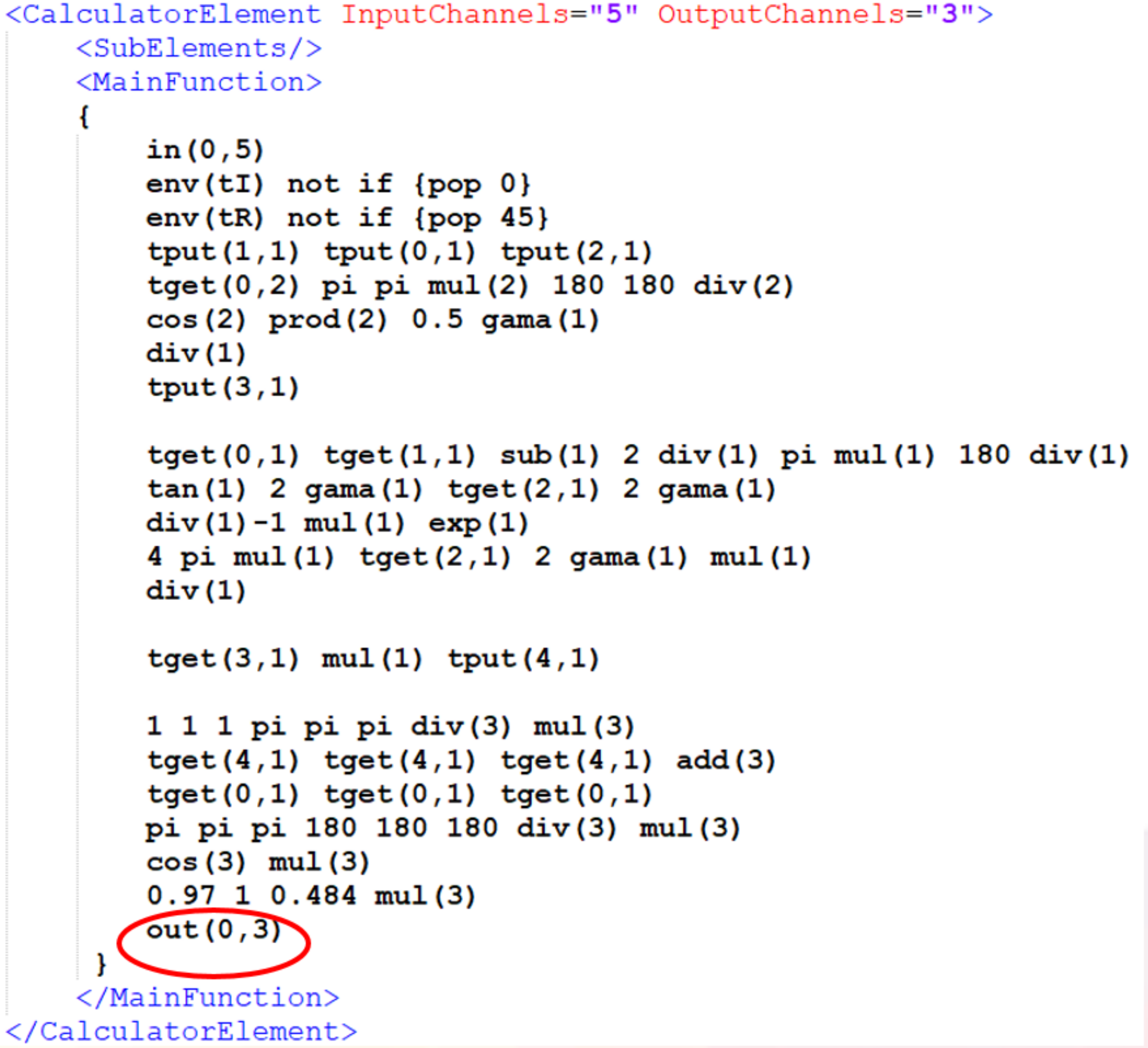
EXAMPLE 2: BRDF IMPLEMENTATION

Ward BRDF

$$I_p(\theta_i;\theta_r) = \begin{bmatrix} I_{pX} \\ I_{pY} \\ I_{pZ} \end{bmatrix} = I_i \cos\theta_i \left(\begin{bmatrix} R_{dX} \\ R_{dY} \\ R_{dZ} \end{bmatrix} \frac{1}{\pi} + \frac{k_S}{\sqrt{\cos\theta_i \cos\theta_r}} \frac{e^{[-\tan^2\delta/m^2]}}{4\pi m^2} \right)$$

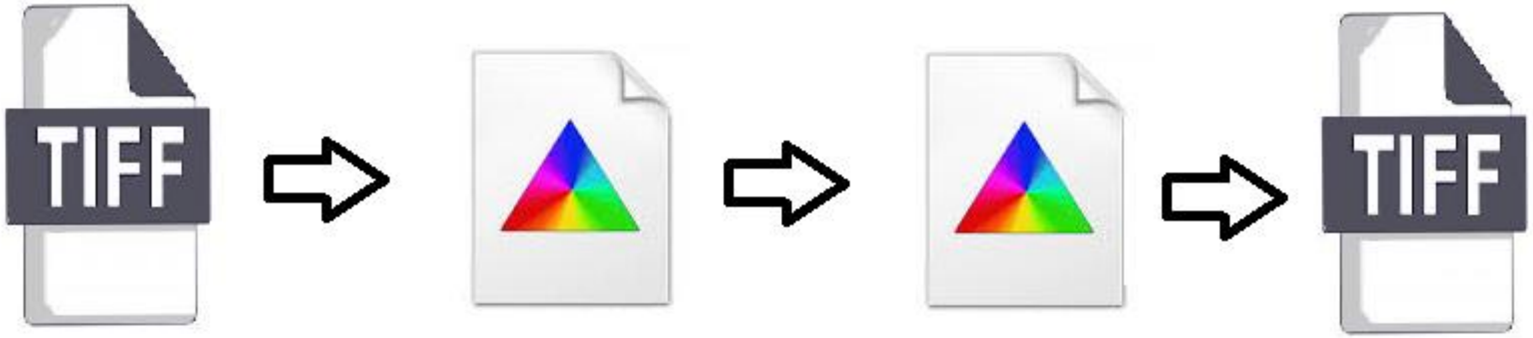
Input

$$\begin{bmatrix} R_{dX} \\ R_{dY} \\ R_{dZ} \end{bmatrix} \quad k_S \quad m$$



```
<CalculatorElement InputChannels="5" OutputChannels="3">
    <SubElements/>
    <MainFunction>
    {
        in(0,5)
        env(tI) not if {pop 0}
        env(tR) not if {pop 45}
        tput(1,1) tput(0,1) tput(2,1)
        tget(0,2) pi pi mul(2) 180 180 div(2)
        cos(2) prod(2) 0.5 gama(1)
        div(1)
        tput(3,1)

        tget(0,1) tget(1,1) sub(1) 2 div(1) pi mul(1) 180 div(1)
        tan(1) 2 gama(1) tget(2,1) 2 gama(1)
        div(1)-1 mul(1) exp(1)
        4 pi mul(1) tget(2,1) 2 gama(1) mul(1)
        div(1)

        tget(3,1) mul(1) tput(4,1)

        1 1 1 pi pi pi div(3) mul(3)
        tget(4,1) tget(4,1) tget(4,1) add(3)
        tget(0,1) tget(0,1) tget(0,1)
        pi pi pi 180 180 180 div(3) mul(3)
        cos(3) mul(3)
        0.97 1 0.484 mul(3)
        out(0,3)
     }
    </MainFunction>
</CalculatorElement>
```

EXAMPLE 2: BRDF IMPLEMENTATION



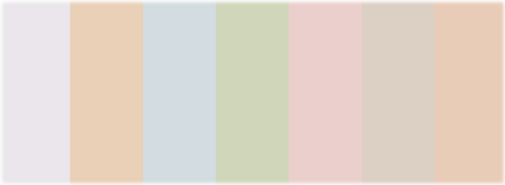
BRDF Parameters

MID Profile

MVIS Profile

XYZ in given geometry

40/0
Input



40/40
Output

THE FUTURE OF COLOR MANAGEMENT

iccMAX

Thank You

The Norwegian **Colour and Visual Computing** Laboratory

NTNU

ApPEARS

APPEARANCE PRINTING

European Advanced Research School