



# **XASM: A Foundation to Program the X2 with P4**

Fabian Ruffy, Vladimir Gurevich

# Agenda

What is X2 and XISA?

Why XASM?

XASM Concepts

To P4 and Beyond!

# The X2 Switch

## 12.8Tbps Programmable Switch

- 400Gbps/200Gbps/100Gbps ports
- Additional PCIe and Ethernet CPU ports

## Ensemble of Data Processing Units (DPUs)

- Programmable Parser
- Programmable Match-Action Processor (MAP)
- Packet buffer, SRAMs, TCAMs

## Open Instruction Set Architecture (XISA)

Source: <https://xsightlabs.com/wp-content/uploads/2025/03/WN-X-Switch-ISA-WP-Mar2025-1.pdf>

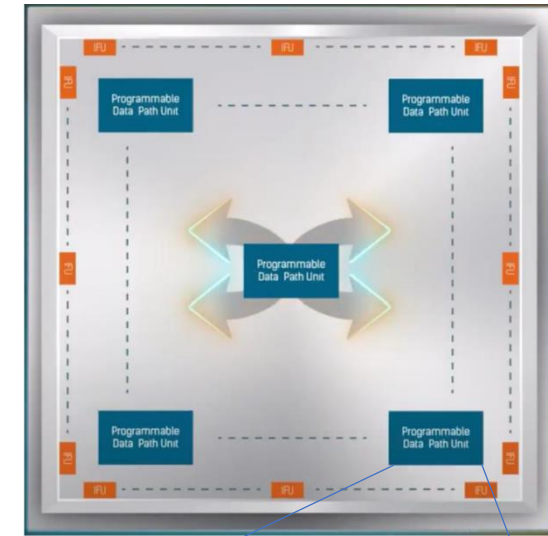


FIGURE 1. X-SWITCH SCALABLE ARCHITECTURE  
(Source: Xsight Labs)

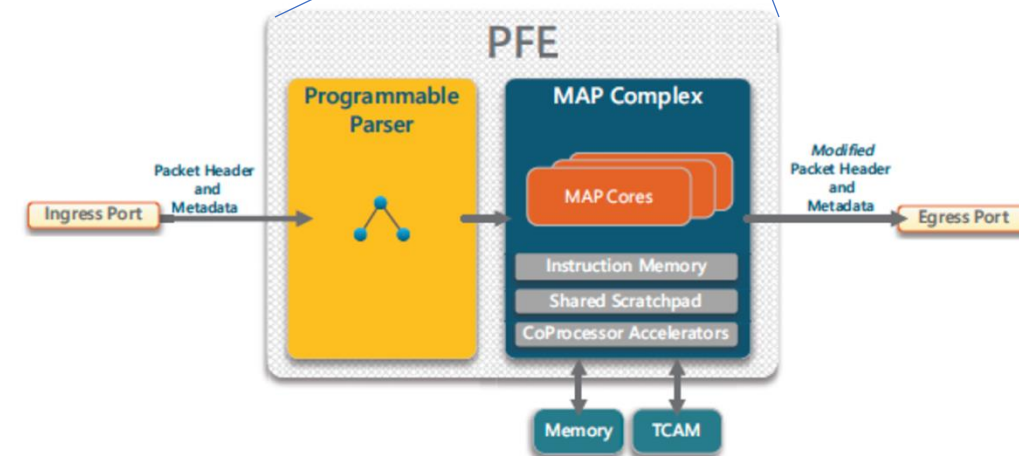
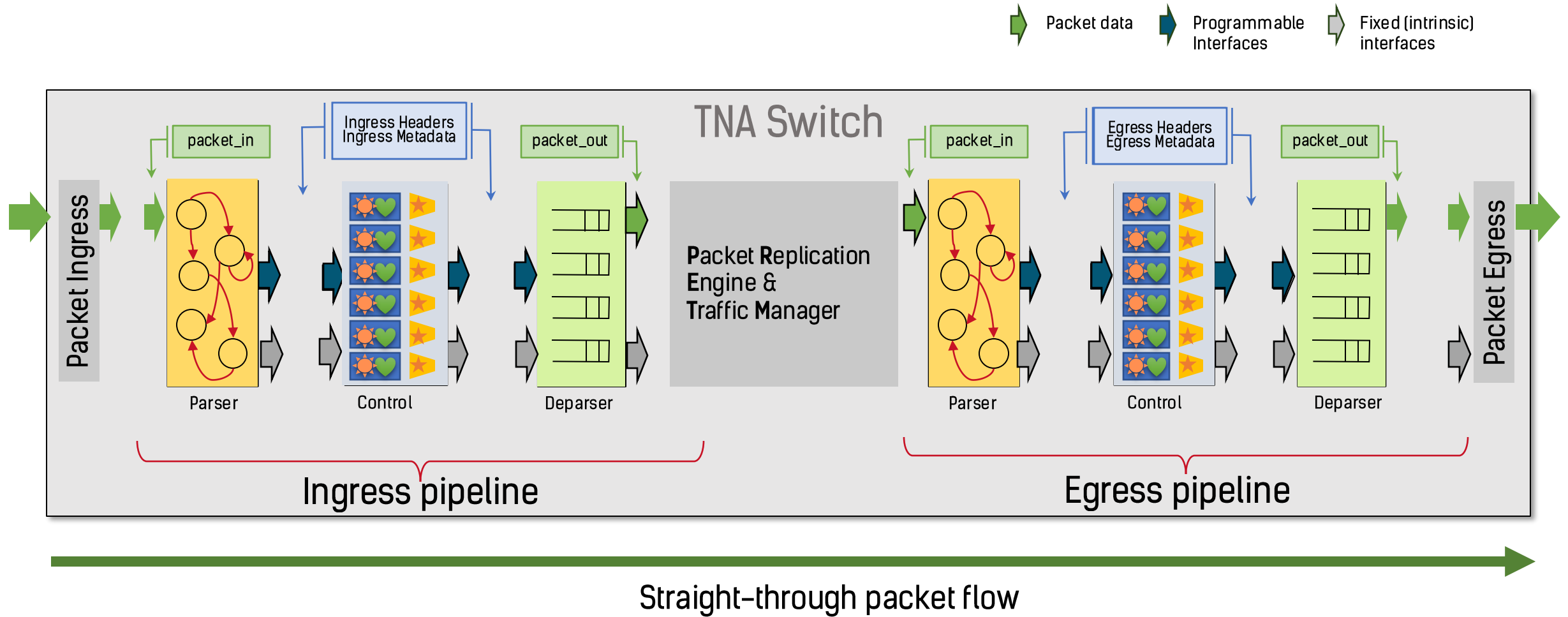


FIGURE 2. PROGRAMMABLE FORWARDING ENGINE  
(Source: Xsight Labs)

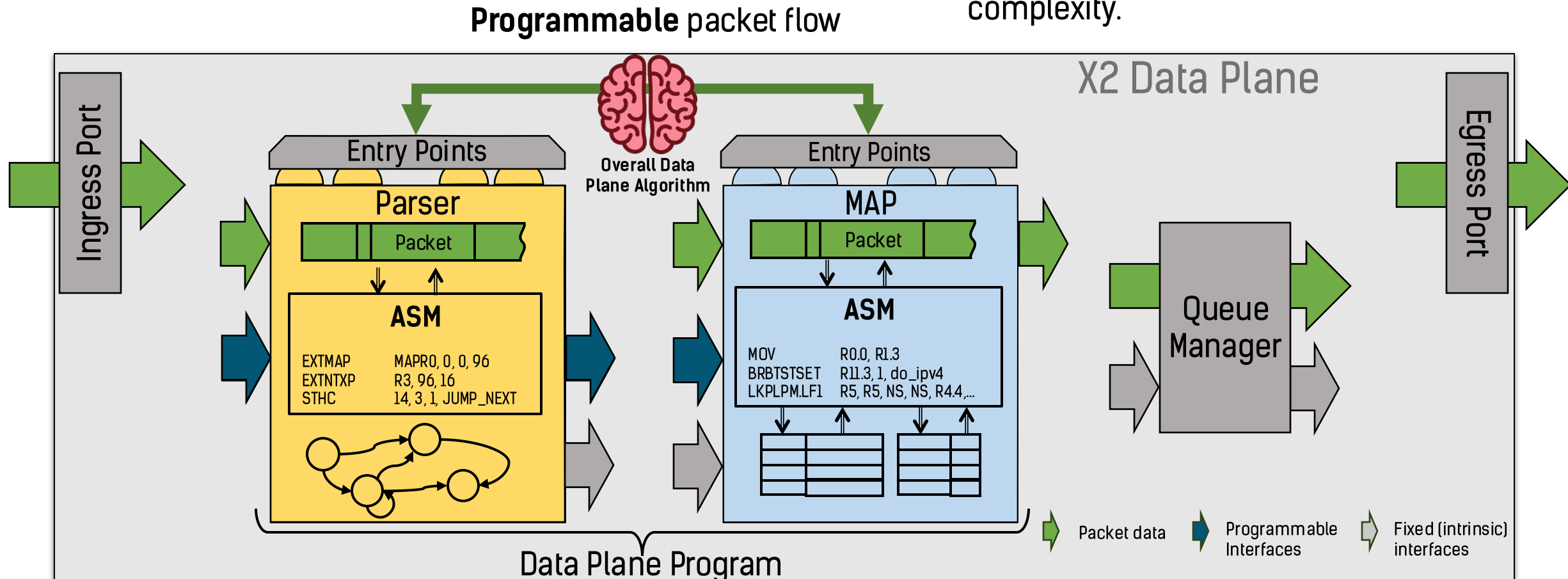
# Tofino Native Architecture (TNA)



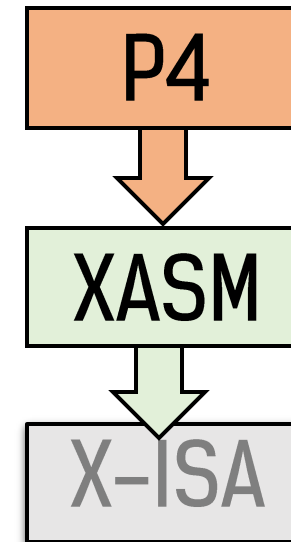
# X2 Programming Model

**Follows P4 Model:** Fixed-function core with programmable portions

**No fixed compute budget,** throughput and latency degrades commensurate with complexity.



# An Open ISA and XASM



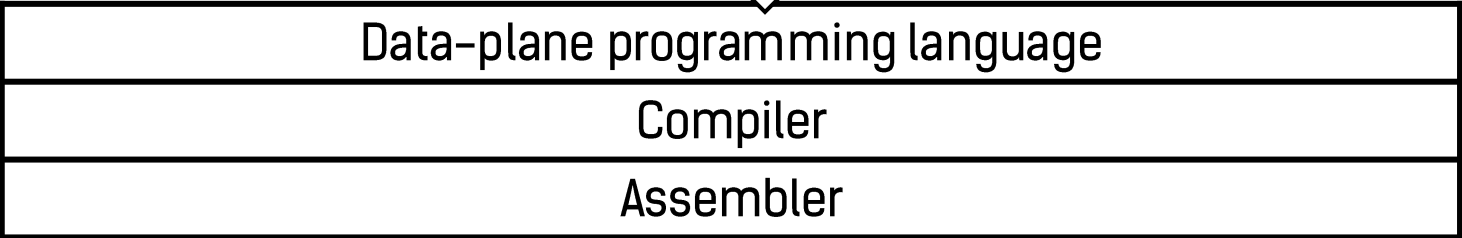
# Compiling A Data-Plane Program

Userland

Device toolchain  
(Vendor-provided)

Microcode (X-ISA)

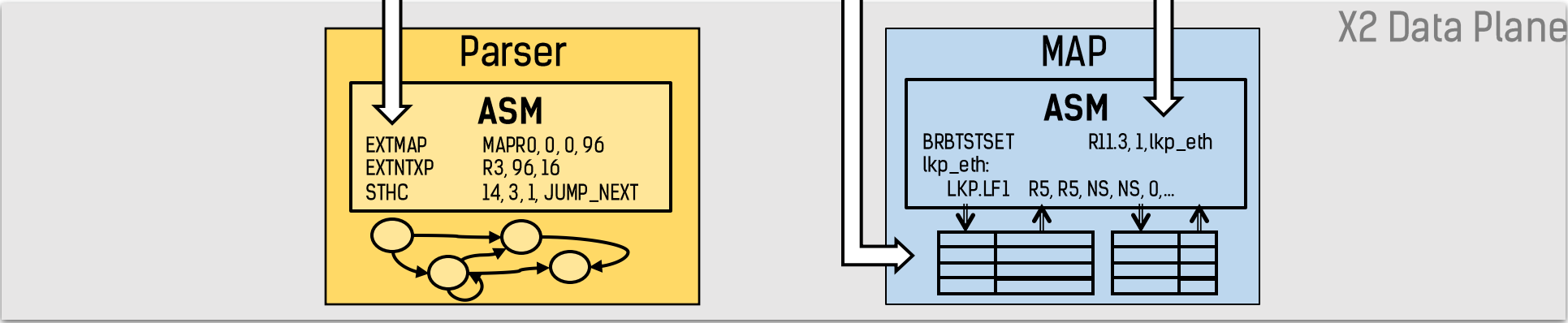
Data-plane program



**Parser ASM**  
EXTMAP      MAPRO, 0, 0, 96  
EXTNTPX     R3, 96, 16  
STHC        14, 3, 1, JUMP\_NEXT

**Control plane objects  
(Tables, etc.)**

**MAP ASM**  
BRBTSTSET      R11.3, 1, lkp\_eth  
lkp\_eth:  
LKP.LF1    R5, R5, NS, NS, 0,...

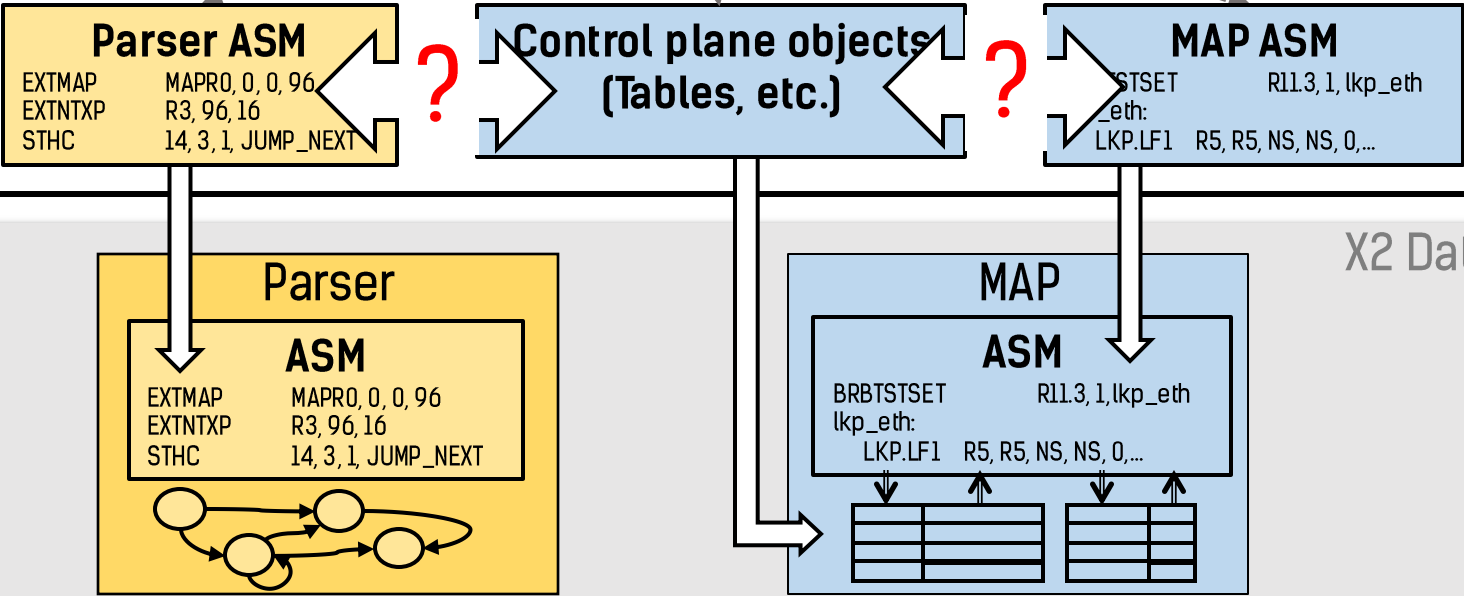


# Down The Switch Stack With An Open ISA

Need to write this code manually and keep it in sync!

Userland

Microcode (X-ISA)



X2 Data Plane

# Raw X-ISA Program

## X2 Parser Instructions

```
entry_point_ethernet:
    EXTNFTP R0, 96, 16
    { // Transition Rules
        entry_point_ethernet
        0x800: entry_point_ipv4, 0, 0
    }
    STHC 14, 12, 1, 1
    HALT
entry_point_ipv4:
    EXTMAP MAPR4, 0, 0, 128
    EXTMAP MAPR2, 0, 128, 32
    STHC 20, 14, 3, 0
    HALT
entry_point_trap:
entry_point_loopback:
entry_point_reparse:
entry_point_fw:
    HALTDROP
```

## X2 MAP Instructions

```
ingress:
    BRBTSTCLR R11.3, 14, not_ipv4_packet
    CONCAT.CD R2.2, 0, R1.3, 0, 16
    LKPLPM.LF5.R R1.3, R1.3, -1, -1, R2, R2, 0, 4
    SYNC.N 32, ipv4_lpm_miss
    AQMEG.LF3 R1.0, R1.3
    MOVI R1.2, 0
    MOVI.CD R0.3, 0
    SYNC 8
    BRBTSTSET R1.0, 0, aqm_drop
    SENDOUT.H R1, R0, 0

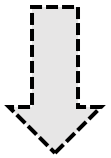
not_ipv4_packet:
ipv4_lpm_miss:
aqm_drop:
    SYNCALL 255
    DROP.H 0
```

# Requirements For Our XDE

1. Make it **easy** to write data-plane programs and **hard** to make mistakes.
2. Give the same control/performance as X-ISA.
3. Export the right interfaces to high-level languages.

# XASM

XASM Language



XASM

Assembler

XDP2

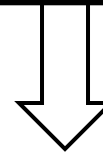
NPL

P4

Python/Rust/C

Data-plane programming language

Compiler



Headers

```
struct eth {  
  bit<48> dst;  
  bit<48> src;  
  bit<16> type;  
}
```

Parser

```
extract_eth:  
  EXTMAP eth[dst...src]  
  EXTNTXP eth.type  
  HALT
```

Tables

```
table fwd {  
  key = { bit<48> dst_mac; }  
  value = { bit<16> qid; }  
}
```

MAP

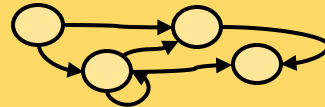
```
ingress:  
  BRITSTSET hdr_present(eth), lkp_eth  
  
lkp_eth:  
  LKP fwd.key, fwd.value, fwd
```

Userland

Parser

ASM

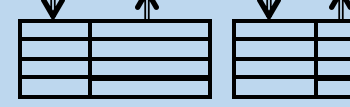
```
EXTMAP  MAPR0, 0, 0, 96  
EXTNTXP R3, 96, 16  
STHC    14, 3, 1, JUMP_NEXT
```



MAP

ASM

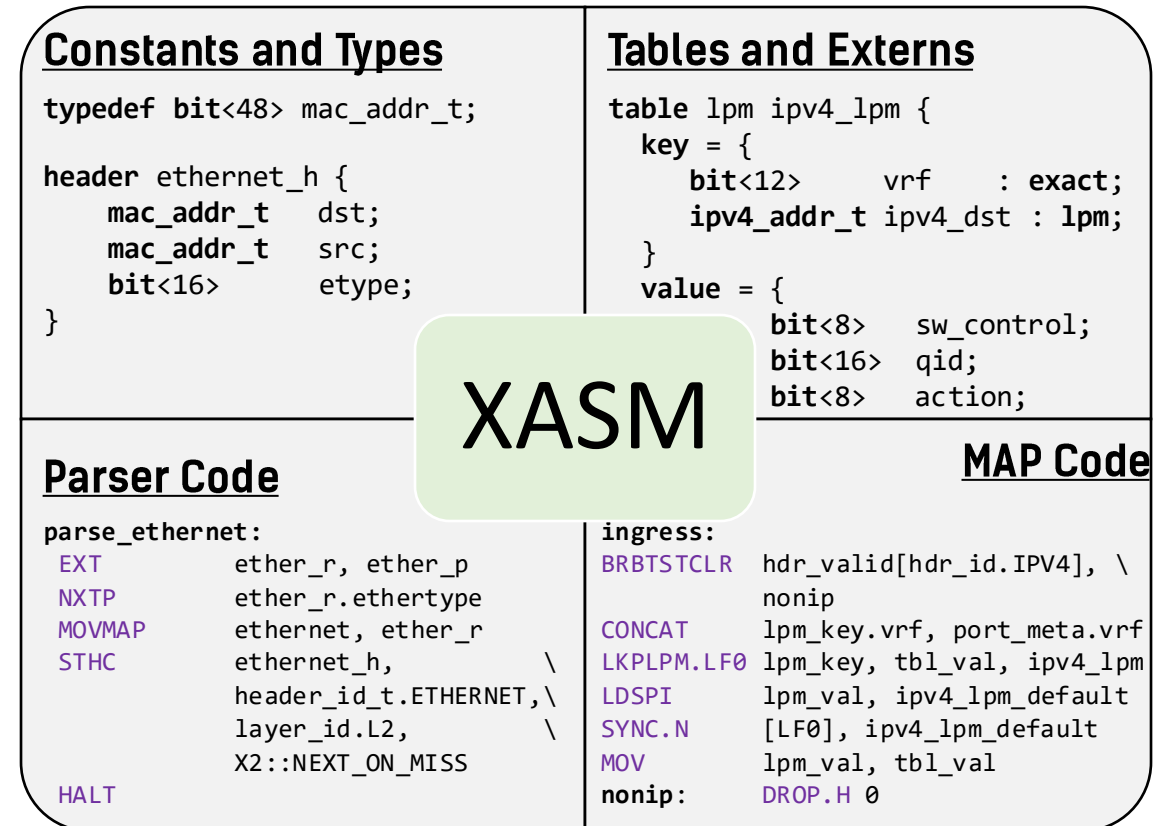
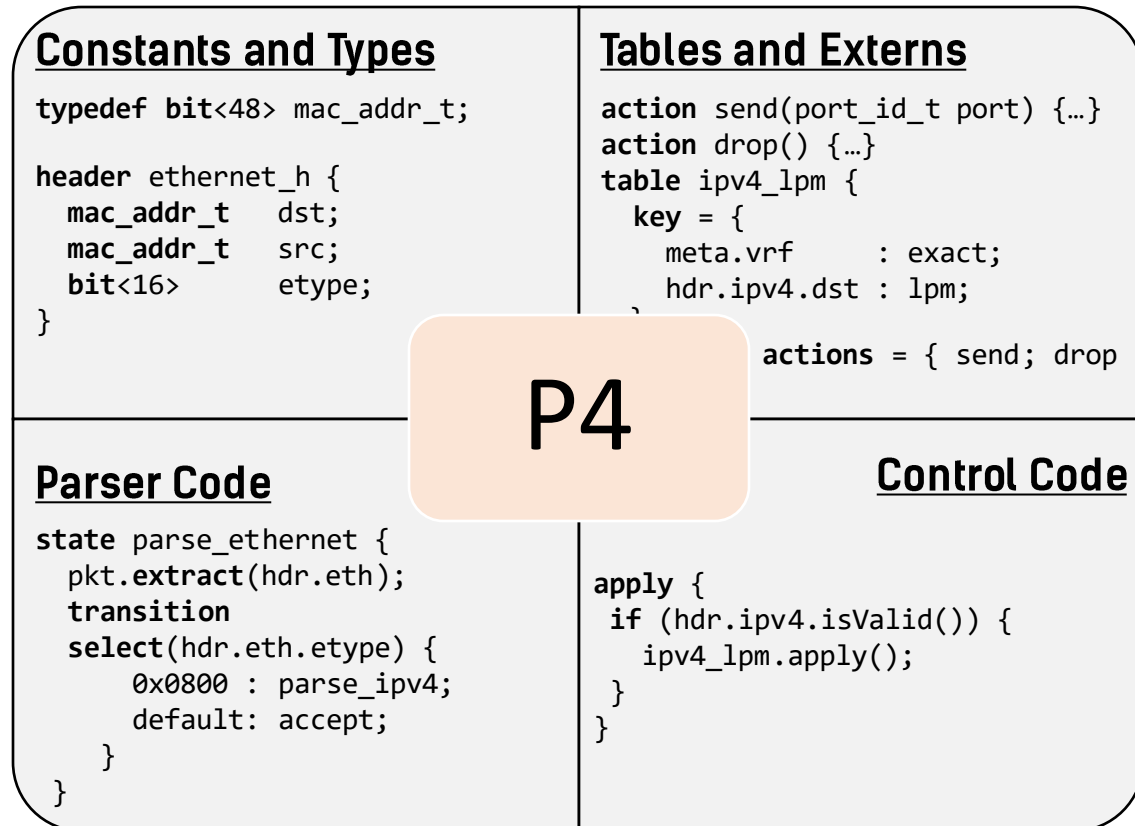
```
BRBTSTSET R11.3, 1, lkp_eth  
lkp_eth:  
  LKP.LF1 R5, R5, NS, NS, 0,...
```



X2 Data Plane

# XASM Is Inspired By P4

Symbolic programming: All definitions are in the same place and work together.  
Control plane interface is an explicit part of the program.



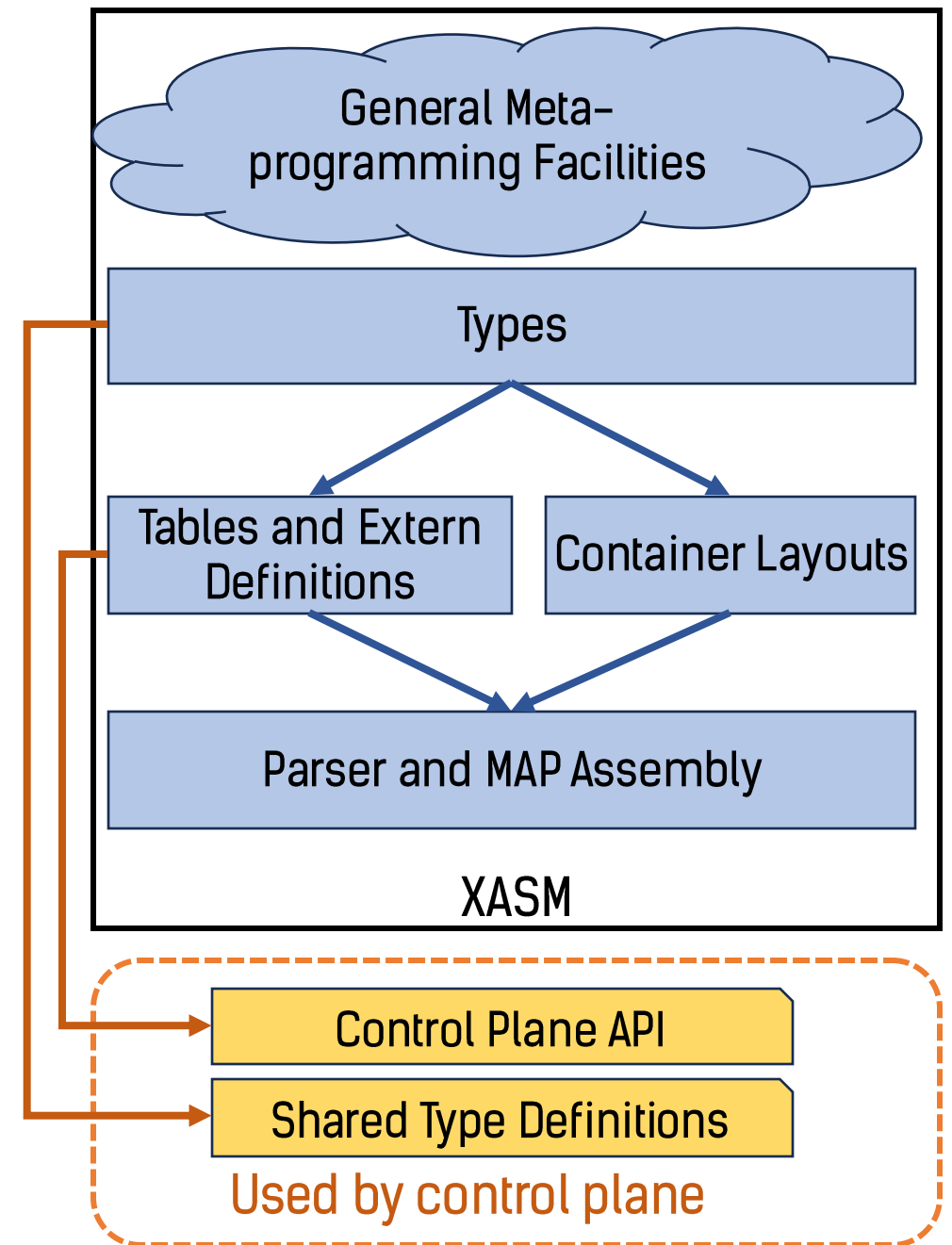
# XASM: Summary

Macro-assembler with metaprogramming

- Functions, macros, and expressions.
- Custom errors, imports, conditionals.
- Definitions (types, tables, externs).

The goal for XASM definitions is to...

- ...to make assembly code more readable.
- ...to generate control plane APIs.
- ...to combine all programmable X2 elements.



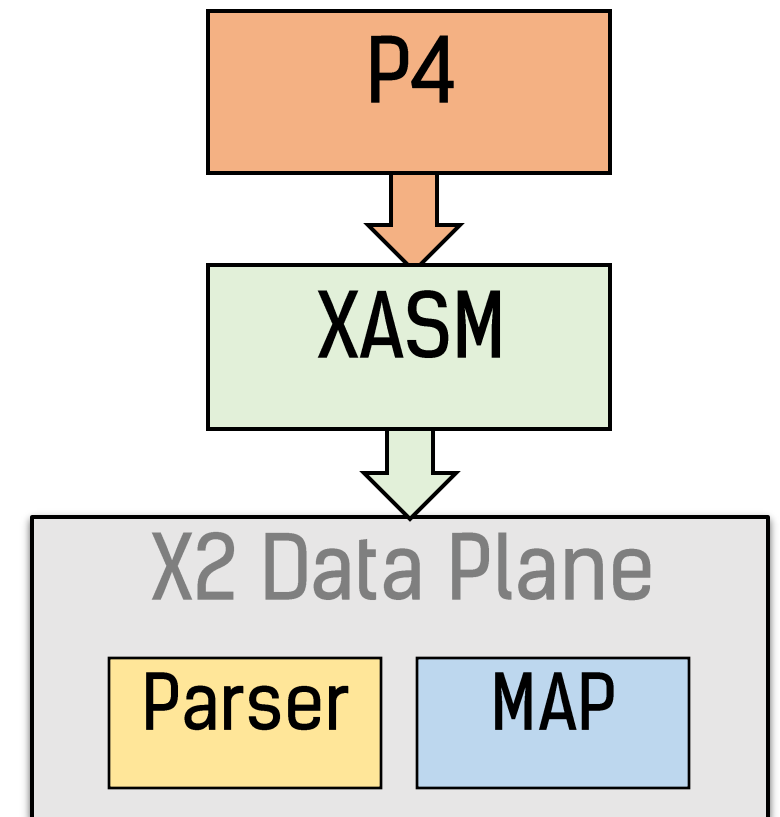
# From XASM To P4

XASM is still assembly! Assembly is hard.

**P4 is our language of choice to support higher-level data-plane programming!**

We are building a P4 compiler!

- XASM is the foundation for this compiler.



# Header Definitions

## P4

```
typedef bit<48> mac_addr_t;
typedef bit<32> ipv4_addr_t;

enum bit<16> ethertype_t {
    TPID = 0x8100,
    IPV4 = 0x0800
}

header ethernet_h {
    mac_addr_t dst;
    mac_addr_t src;
    ethertype_t etype;
}

header ipv4_h {
    bit<4> ihl;
    bit<4> version;
    ...
    ipv4_addr_t src;
    ipv4_addr_t dst;
}

header ipv6_h {
```

## XASM

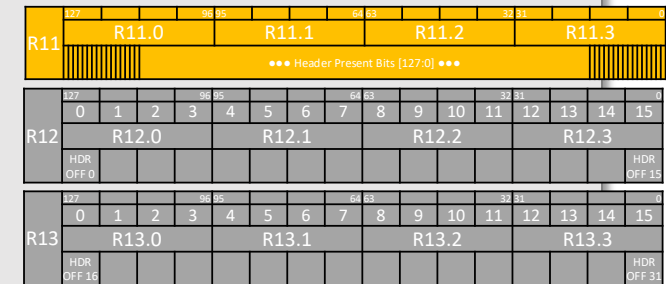
```
typedef bit<48> mac_addr_t;
typedef bit<32> ipv4_addr_t;

enum bit<16> ethertype_t {
    TPID = 0x8100,
    IPV4 = 0x0800
}

header ethernet_h {
    mac_addr_t dst;
    mac_addr_t src;
    ethertype_t etype;
}

header ipv4_h {
    bit<4> ihl;
    bit<4> version;
    ...
    ipv4_addr_t src;
    ipv4_addr_t dst;
}

enum bit<7> header_id_t { ETHERNET, IPV4, IPV6 };
enum bit<5> layer_id_t { L2, L3 };
```



# Extracting an Ethernet Header

## P4

```
struct my_headers_t {
    ethernet_h ethernet;
}

parser MyParser(pkt_in pkt,
    out my_headers_t hdr, ...) {

    state parse_ethernet {
        pkt.extract(hdr.ethernet);

        ethertype_t etype = hdr.ethernet.etype;

        transition select(etype) {
            ethertype_t.IPv4: parse_ipv4;
            default:         accept;
        }
    }

    state parse_ipv4 {
        pkt.extract(hdr.ipv4);
        transition accept;
    }
}
```

## XASM

```
parserdef MyParser {

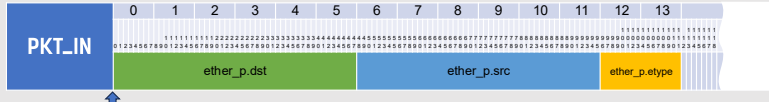
    assembly {

        parse_ethernet:
            assume ethernet_h ether_p in PKT_IN;

        TransitionRules {
            ethertype_t.IPv4 -> parse_ipv4;
        }

        EXTMAP    ethernet[.dst:.src], ether_p[.dst:.src]
        EXTNXTP.CD ether, ether_p.etype
        MOVMAP    ethernet.ethertype, ethertype
        STHC      ether_h.size_bytes(), \
                header_id_t.ETHERNET, \
                layer_id.L2, \
                X2::NEXT_ON_MISS

        HALT
    }
}
```



# Extracting an Ethernet Header

## P4

```

struct my_headers_t {
    ethernet_h ethernet;
}

parser MyParser(pkt_in pkt,
                out my_headers_t hdr, ...) {

    state parse_ethernet {
        pkt.extract(hdr.ethernet);

        ethertype_t etype = hdr.ethernet.etype;

        transition select(etype) {
            ethertype_t.IPv4: parse_ipv4;
            default:          accept;
        }

    state parse_ipv4 {
        pkt.extract(hdr.ipv4);
        transition accept;
    }
}
    
```

## XASM

```

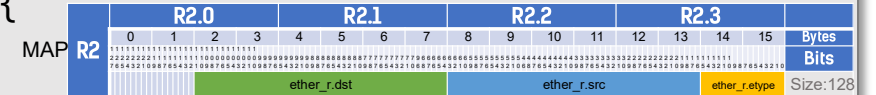
parserdef MyParser {
    assembly {

        parse_ethernet:
            assume ethernet_h ether_p in PKT_IN;
            assume ethernet_h ethernet in R2;

            TransitionRules {
                ethertype_t.IPv4 -> parse_ipv4;
            }

            EXTMAP      ethernet[.dst:.src], ether_p[.dst:.src]
            EXTNXTP.CD  ether, ether_p.etype
            MOVMAP      ethernet.etype, etype
            STHC        ether_h.size_bytes(), \
                    header_id_t.ETHERNET, \
                    layer_id.L2, \
                    X2::NEXT_ON_MISS

            HALT
    }
}
    
```



# Extracting an Ethernet Header

## P4

```
struct my_headers_t {
    ethernet_h ethernet;
}

parser MyParser(pkt_in pkt,
                out my_headers_t hdr, ...) {

    state parse_ethernet {
        pkt.extract(hdr.ethernet);

        ethertype_t etype = hdr.ethernet.etype;

        transition select(etype) {
            ethertype_t.IPv4: parse_ipv4;
            default:         accept;
        }
    }

    state parse_ipv4 {
        pkt.extract(hdr.ipv4);
        transition accept;
    }
}
```

## XASM

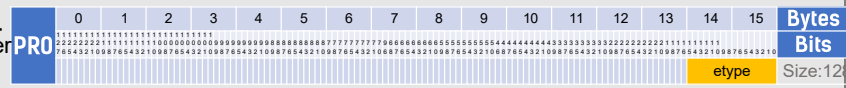
```
parserdef MyParser {
    assembly {

        parse_ethernet:
            assume ethernet_h ether_p in PKT_IN;
            assume ethernet_h ethernet in R2;
            assume ethertype_t etype in PR0;

            TransitionRules {
                ethertype_t.IPv4 -> parse_ipv4;
            }

            EXTMAP ethernet[.dst:.src], ether_p[.dst:.src]
            EXTNXTP.CD ether, ether_p.ethertype
            MOVMAP ethernet.ethertype, ethertype
            STHC ether_h.size_bytes(), \
                header_id_t.ETHERNET, \
                layer_id.L2, \
                X2::NEXT_ON_MISS

            HALT
    }
}
```



# Extracting an Ethernet Header

## P4

```
struct my_headers_t {
    ethernet_h ethernet;
}

parser MyParser(pkt_in pkt,
                out my_headers_t hdr, ...) {

    state parse_ethernet {
        pkt.extract(hdr.ethernet);

        ethertype_t etype = hdr.ethernet.etype;

        transition select(etype) {
            ethertype_t.IPv4: parse_ipv4;
            default:         accept;
        }
    }

    state parse_ipv4 {
        pkt.extract(hdr.ipv4);
        transition accept;
    }
}
```

## XASM

```
parserdef MyParser {

assembly {

    parse_ethernet:
        assume ethernet_h ether_p in PKT_IN;
        assume ethernet_h ethernet in R2;
        assume ethertype_t etype in PR0;

        TransitionRules {
            ethertype_t.IPv4 -> parse_ipv4;
        }

        EXTMAP ethernet[.dst:.src], ether_p[.dst:.src]
        EXTNXTP.CD ether, ether_p.etype
        MOVMAP ethernet.etype, ethertype
        STHC ether_h.size_bytes(), \
             header_id_t.ETHERNET, \
             layer_id.L2, \
             X2::NEXT_ON_MISS

        HALT
    }
}
```

# Defining a Match Table

## P4

```
action send(queue_id_t qid) {
    if (X2_is_full(qid)) {
        X2_drop();
    } else {
        X2_simple_send(qid);
    }
}

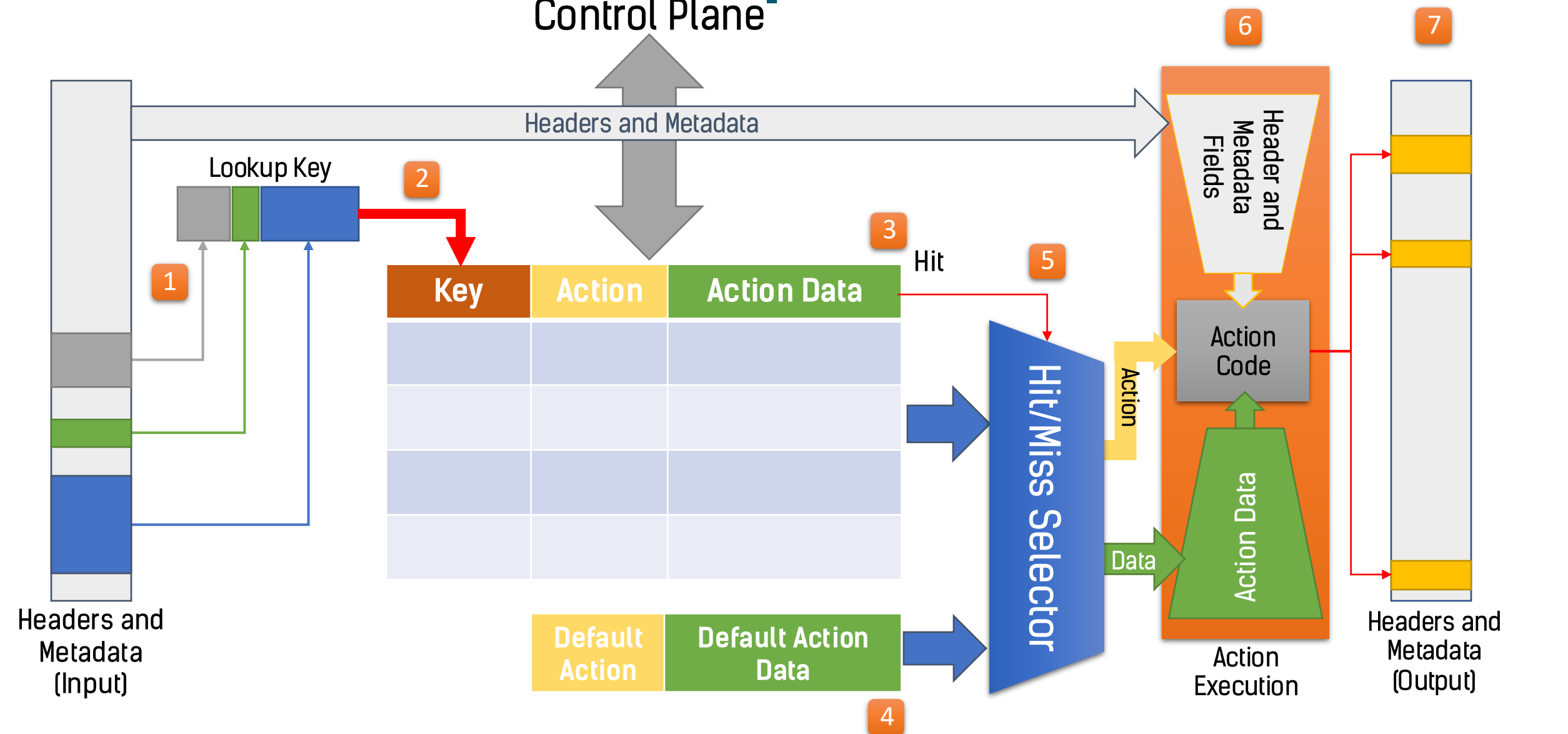
action drop() {
    X2_drop();
}

table ipv4_lpm {
    key = {
        meta.vrf      : exact;
        hdr.ipv4.dst  : lpm;
    }
    actions = {
        send; drop;
    }
    size = 65536;
}
```

## XASM

```
table lpm ipv4_lpm {
    key = {
        bit<12>      vrf      : exact;
        ipv4_addr_t  ipv4_dst : lpm;
    }
    value = {
        bit<1>      valid;
        bit<7>      _;        /* Padding */
        bit<16>     qid;
        bit<6>      _;        /* Padding */
        bit<2>      action;    /* Bit encoded */
    }
    properties = {
        size          = 65536;
        duplicate      = X2::DUPLICATE_TO_ALL;
        mem_region     = "4x4";
        cache_enable   = true;
    }
}
```

# Match-Action Table Operation



# Performing Match-Action Operation

## P4

```
control MyIngress(inout my_headers_t hdr, ...) {  
  action send() { ... }  
  action drop() { ... }  
  table ipv4_lpm { ... }  
  
  apply {  
    ...  
    if (hdr.ipv4.isValid()) {  
      ipv4_lpm.apply();  
    }  
    ...  
  }  
}
```

## XASM

```
mapdef MyMap {  
  table lpm ipv4_lpm { ... }  
  
  Assembly {  
    assume X2::header_valid_t header_valid in R11;  
    assume ipv4_lpm::key lpm_key in R3.2..R3.3;  
    assume ipv4_lpm::value table_val in R3.0;  
    assume ipv4_lpm::value lpm_val in R3.1;  
  
    BRBTSTCLR hdr_valid[header_id_t.IPV4], not_ipv4_packet  
    CONCAT lpm_key.vrf, port metadata.vrf  
    LKPLPM.LF0 lpm_key, table_val, ipv4_lpm  
    LDSPI lpm_val, ipv4_lpm_default_entry  
    SYNC.N [LF0], ipv4_lpm_miss  
    MOV lpm_val, table_val /* This is a hit */  
  
    ipv4_lpm_miss:  
      JTL.NM lpm_val.action, \  
        ipv4_lpm_send, ipv4_lpm_drop, ipv4_lpm_done  
  
    ipv4_lpm_send:  
      ...  
      BRI ipv4_lpm_done  
  
    ipv4_lpm_drop:  
      DROP 0  
      BRI ipv4_lpm_done  
  
    ipv4_lpm_done:  
      ...  
  
    not_ipv4_packet:
```

|   |
|---|
| 1 |
| 2 |
| 4 |
| 5 |
| 3 |
| 6 |
| 7 |

# The Opportunity With the X2: Choice

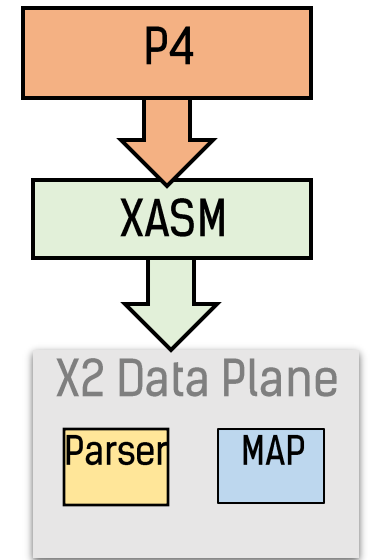
The open ISA gives you lots of flexibility on the X2 hardware!

Performance scales gracefully.

- Complex data plane code might be slower but always compiles.
- There are many ways to write very efficient code.

How far can we go with P4? We are not sure.

- We are certain about one approach, but there might be others!
- Many P4 architectures can be implemented.
- You can experiment with other languages.



# Build X2 Applications With Us!

The X2 is a new programmable switch on the block.

- It has an open ISA!

We want you to experiment or build data-plane programs on the X2!

- We make this easy with XASM.
- We are working on P4 as the first high-level language to program the X2.

When is this software available? Stay tuned...

- Talk to us! [fabianr@xsightlabs.com](mailto:fabianr@xsightlabs.com), [vladimirg@xsightlabs.com](mailto:vladimirg@xsightlabs.com)