

TyCL

The TyCL compiler internals and preliminary performance analysis

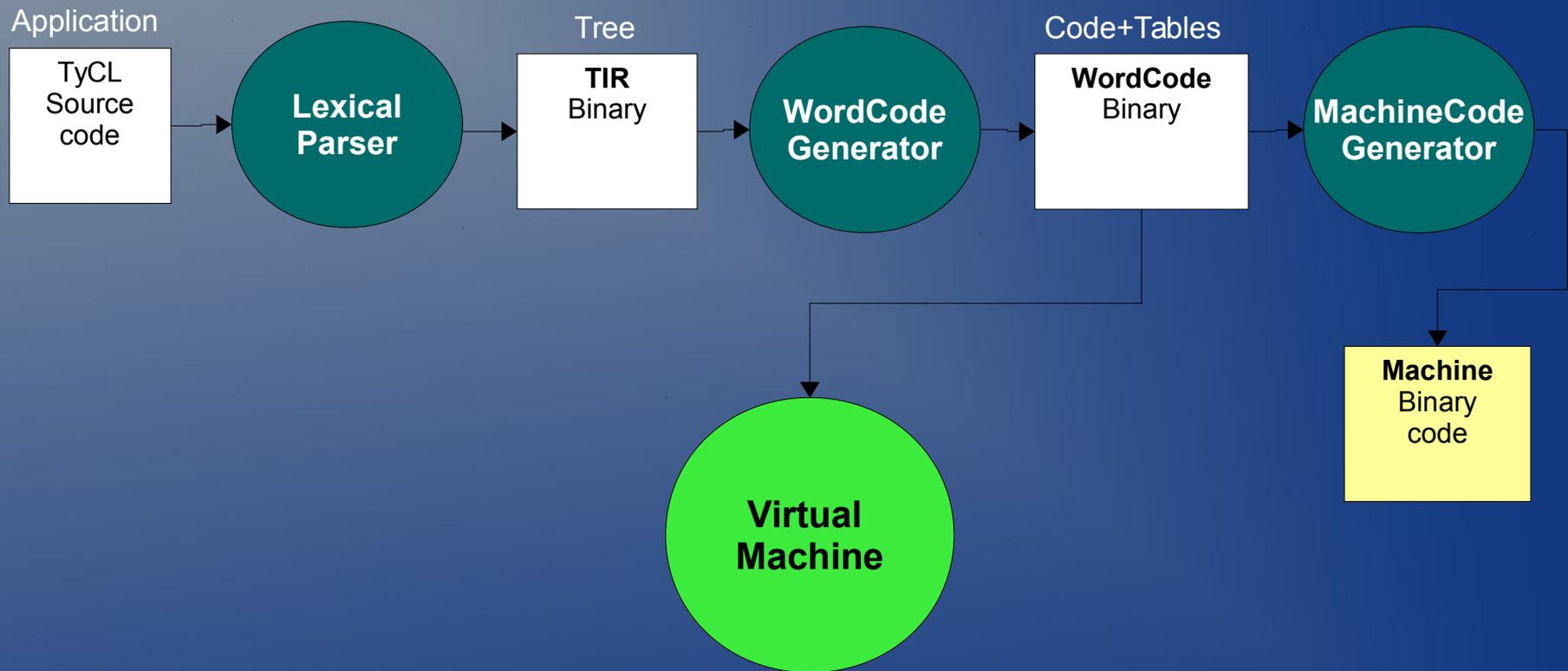
Andres Buss

Otlet Technologies

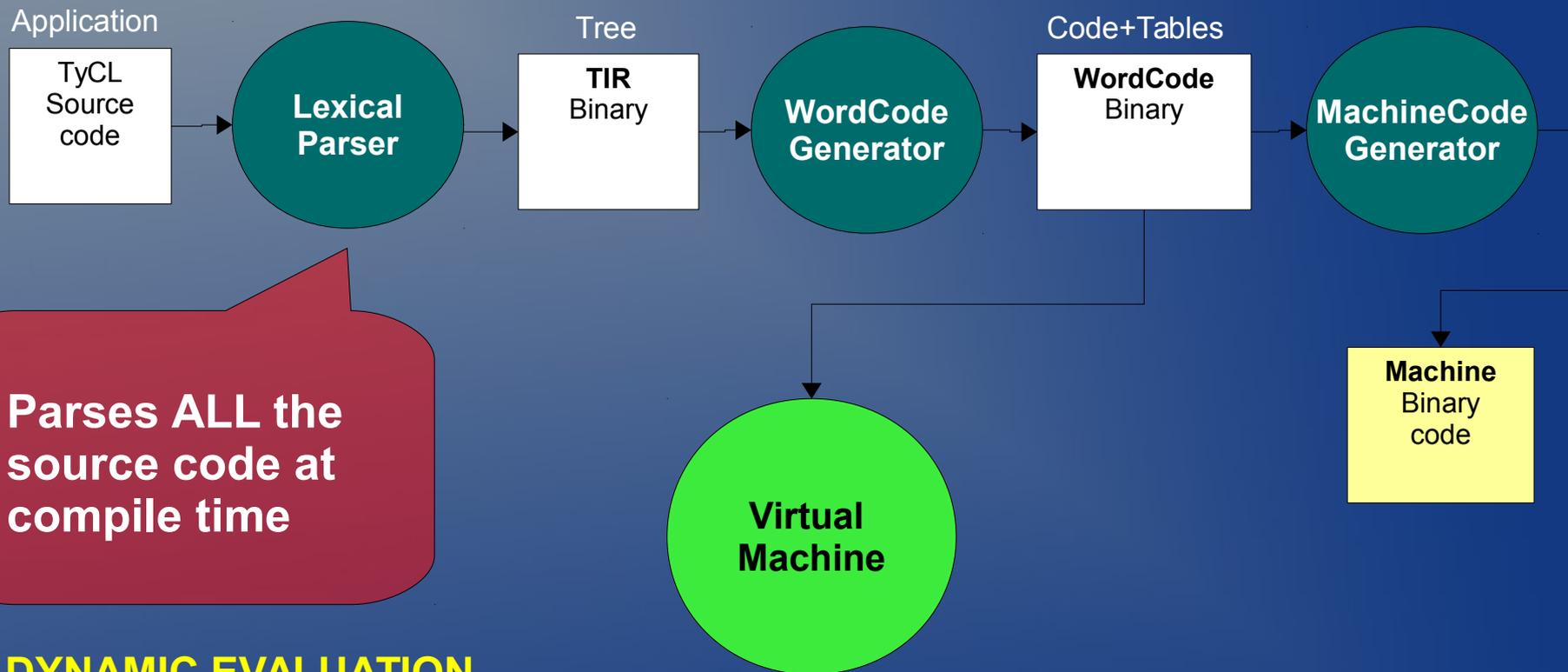
Does it really work?

How ?

Architecture



Architecture

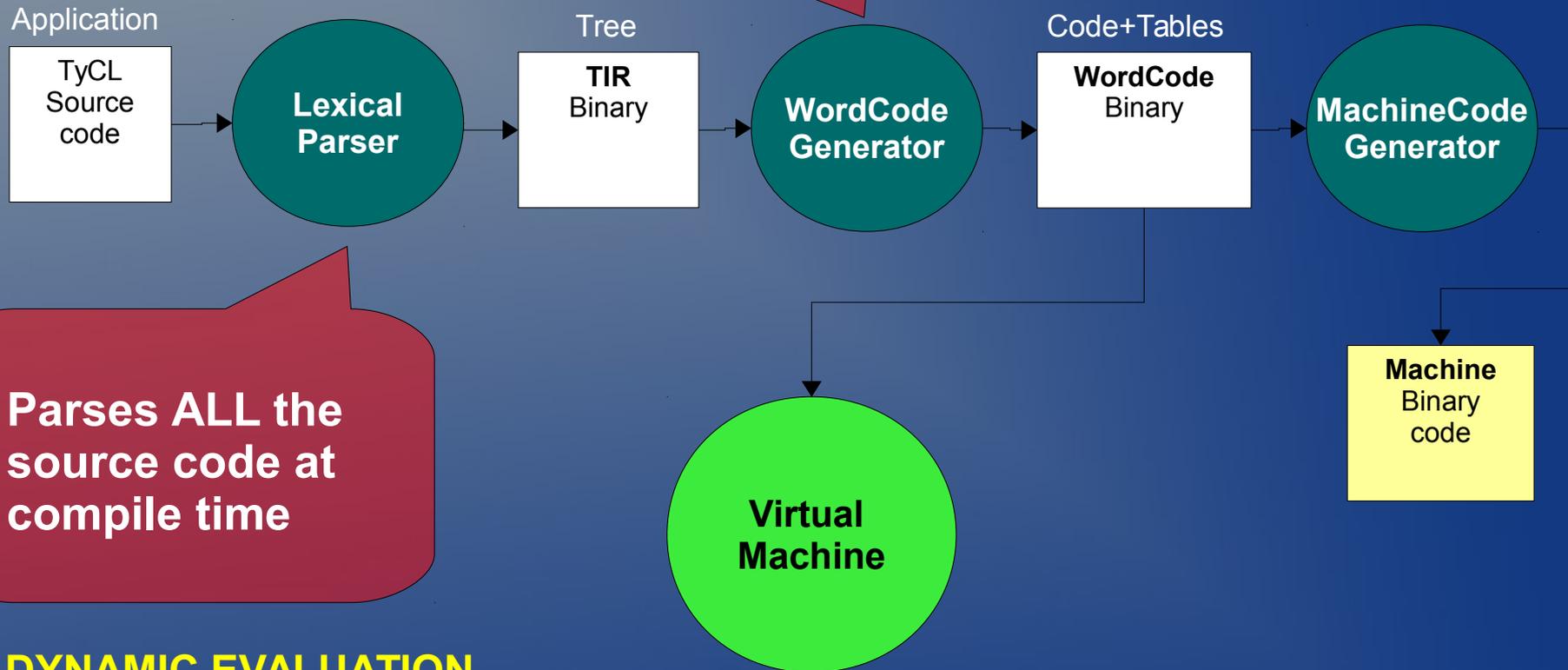


Parses ALL the source code at compile time

NO DYNAMIC EVALUATION AT RUNTIME ... at the moment

Architecture

Detects data types as much as possible,
Falls back to discovery when it can't do the detection.



Parses ALL the source code at compile time

NO DYNAMIC EVALUATION AT RUNTIME ... at the moment

Architecture

Detects data types as much as possible, Falls back to discovery when it can't do the detection.

Generates machine code specific to the target

Application

TyCL
Source
code

Lexical
Parser

Tree

TIR
Binary

WordCode
Generator

Code+Tables

WordCode
Binary

MachineCode
Generator

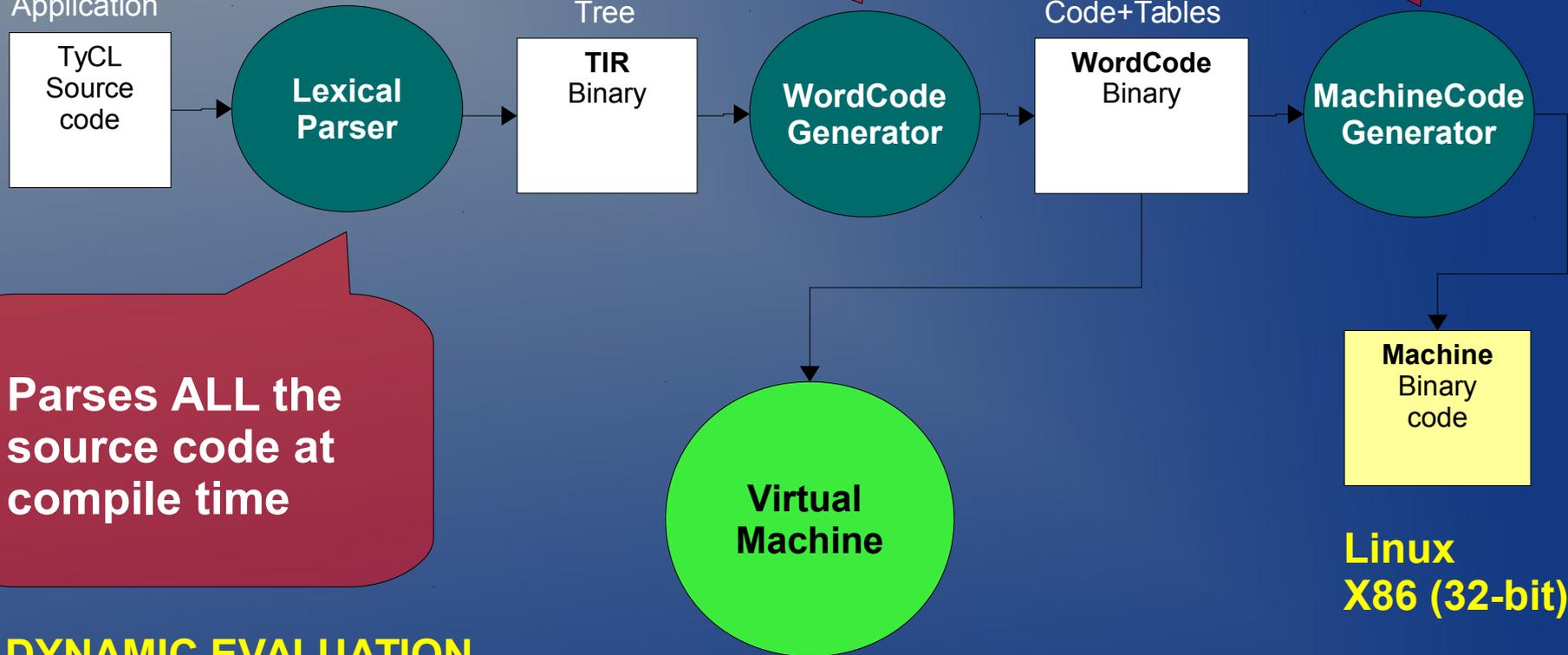
Parses ALL the
source code at
compile time

Virtual
Machine

Machine
Binary
code

Linux
X86 (32-bit)

NO DYNAMIC EVALUATION
AT RUNTIME ... at the moment



The memory models

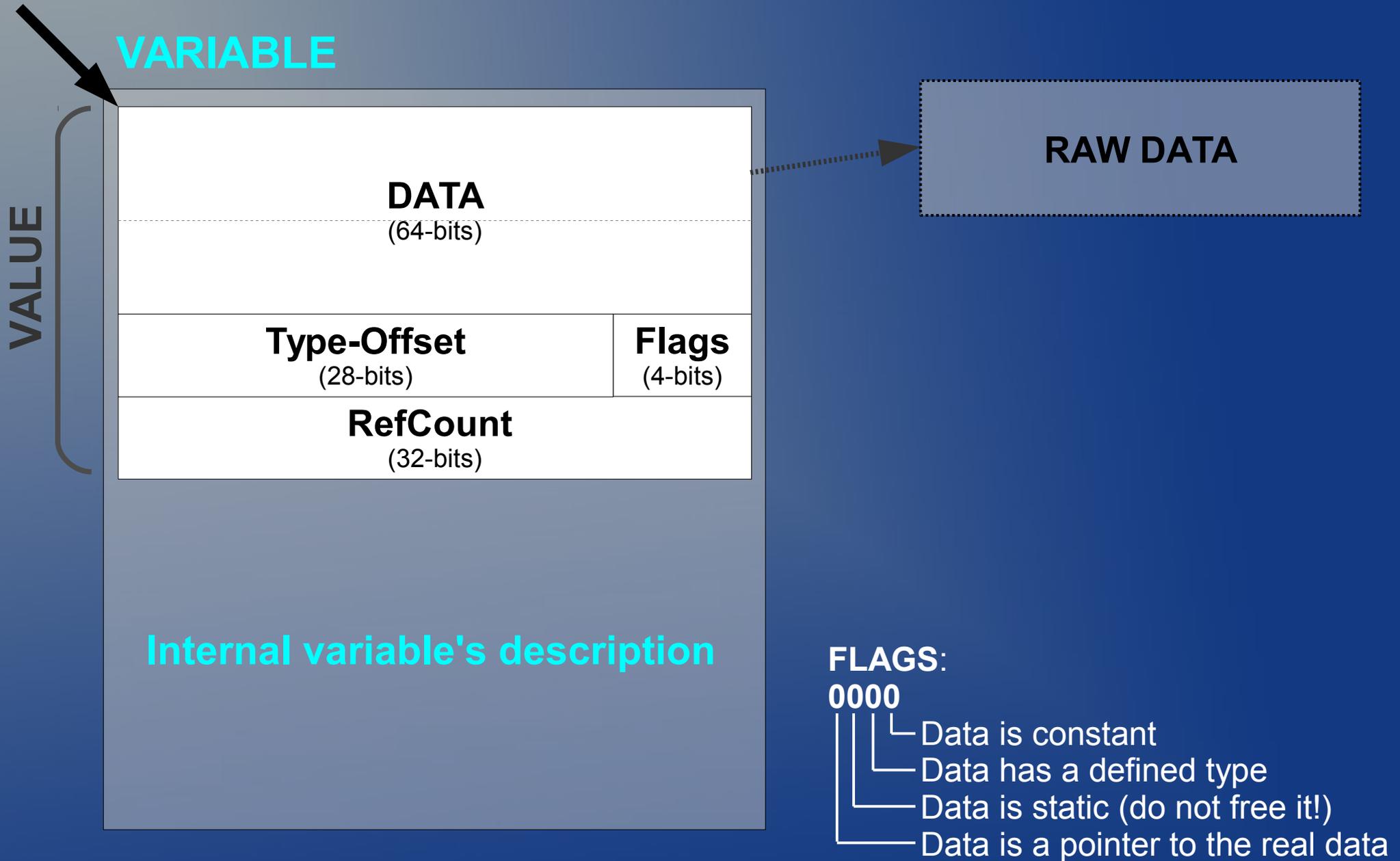
1. How are values stored in memory ?
2. How are types stored in memory ?
3. How are types related to values ?

The memory models

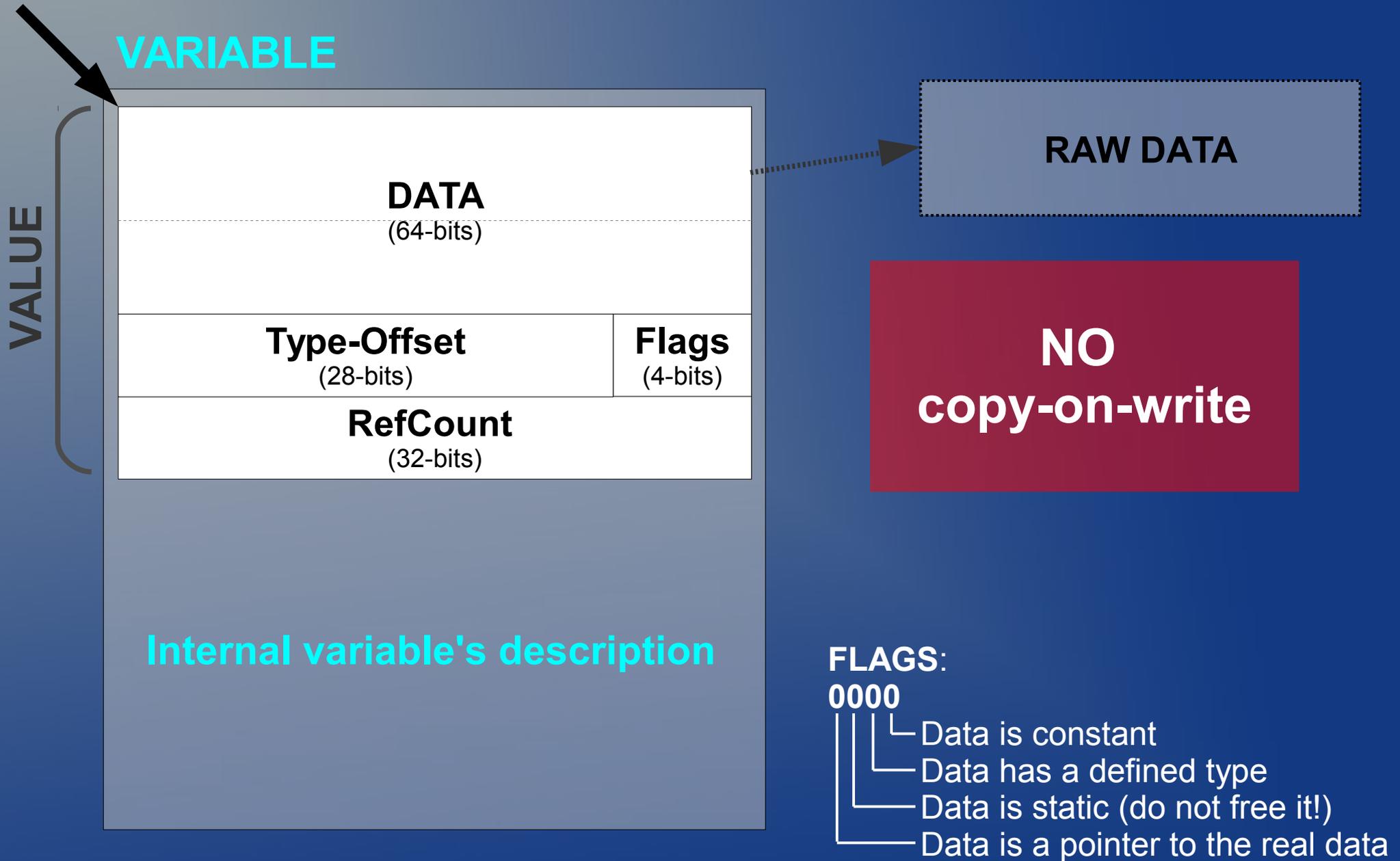
1. How are values stored in memory ?
2. How are types stored in memory ?
3. How are types related to values ?

**Avoid asking / releasing dynamic memory
as much as possible**

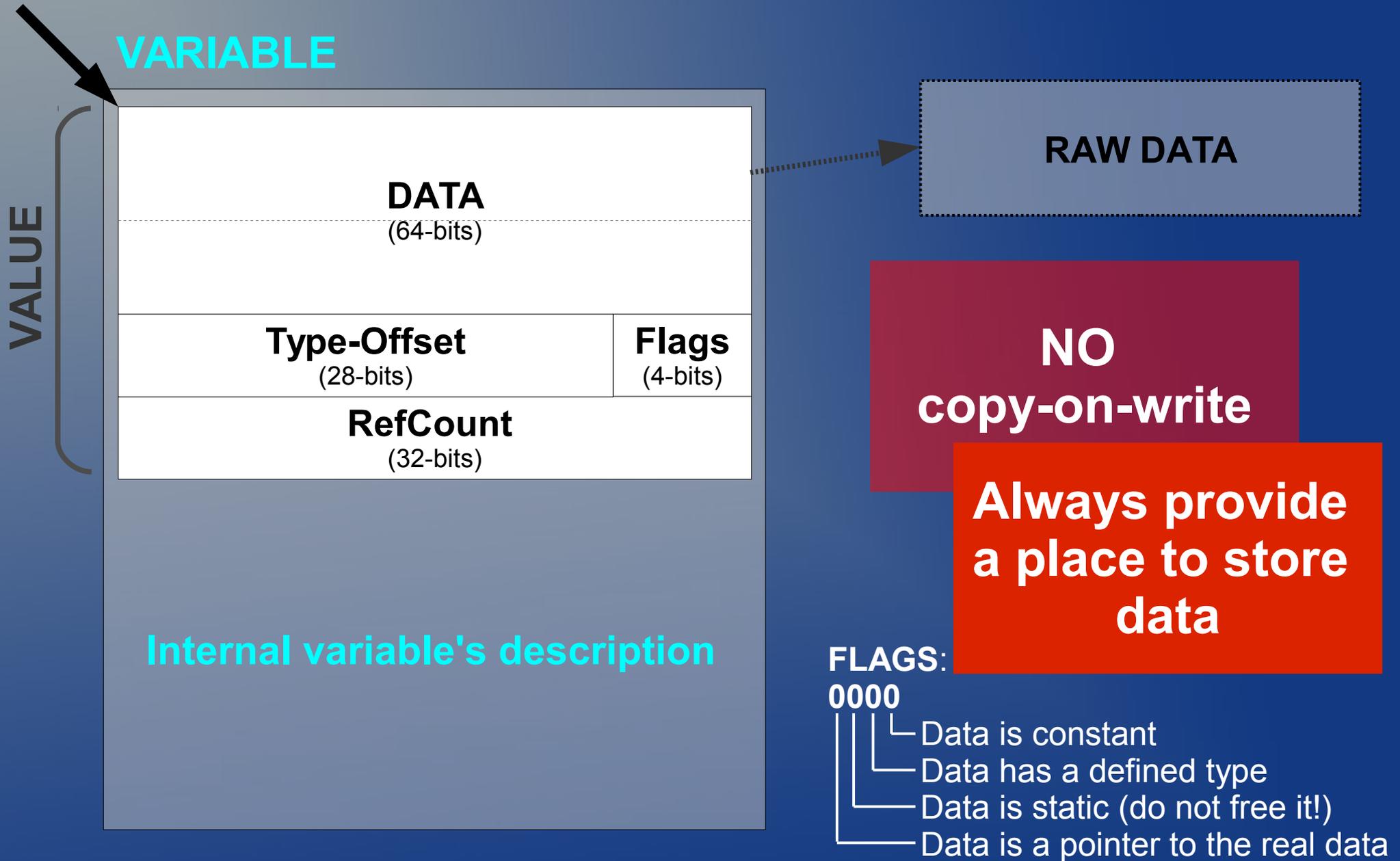
Values in memory



Values in memory



Values in memory



The Type system

What is a type?

```
struct:{  
    integer:flags  
    @tonum  
    @getnum  
    @tostr  
    @tobool  
    @set  
    @get  
    @cast  
    @length                @size  
    @exec  
    @setIndex              @getIndex              @refIndex  
    @setRange              @getRange              @refRange  
    @setMember             @getMember            @refMember  
    @newframe  
    @namedarg  
    @setflag  
    @parloff  
    @exec  
}
```

The Type system

What is a type?

```
struct:{  
    integer:flags  
    @tonum  
    @getnum  
    @tostr  
    @tobool  
    @set  
    @get  
    @cast  
    @length          @size  
    @exec  
    @setIndex        @getIndex        @refIndex  
    @setRange        @getRange        @refRange  
    @setMember       @getMember       @refMember  
    @newframe  
    @namedarg  
    @setflag  
    @parloff  
    @exec  
}
```



Type Descriptor

Common
Functions

Internal
Type's
Descriptor

The Type system

What is a type?

```
struct:{  
    integer:flags  
    @tonum  
    @getnum  
    @tostr  
    @tobool  
    @set  
    @get  
    @cast  
    @length  
    @exec  
    @setIndex  
    @setRange  
    @setMember  
    @newframe  
    @namedarg  
    @setflag  
    @parmoff  
    @exec  
}
```

@size

@getIndex

@getRange

@getMember

@refIndex

@refRange

@refMember

**All type's descriptors
are stored (in order)
in a table**

Internal
Type's
Descriptor

The Type system

What is a type?

```
struct:{
```

```
integer:flags
```

```
@tonum
```

```
@getnum
```

```
@tostr
```

```
@tobool
```

```
@set
```

```
@get
```

```
@cast
```

```
@length
```

```
@exec
```

```
@setIndex
```

```
@setRange
```

```
@setMember
```

```
@newframe
```

```
@namedarg
```

```
@setflag
```

```
@parloff
```

```
@exec
```

```
}
```

```
@size
```

```
@getIndex
```

```
@getRange
```

```
@getMember
```

```
@refIndex
```

```
@refRange
```

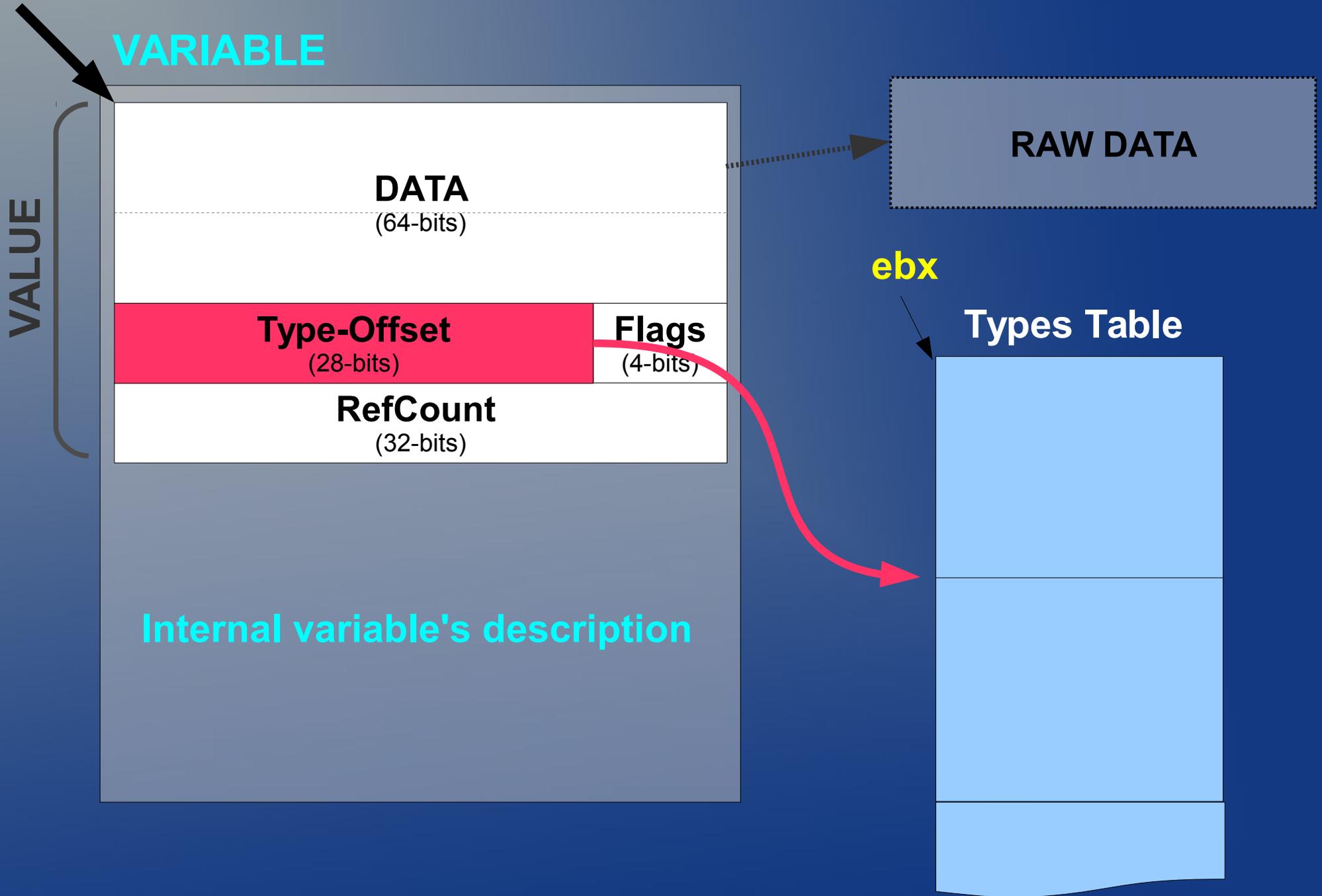
```
@refMember
```

All type's descriptors
are stored (in order)
in a table

Arguments are
passed in the
CPU's registers

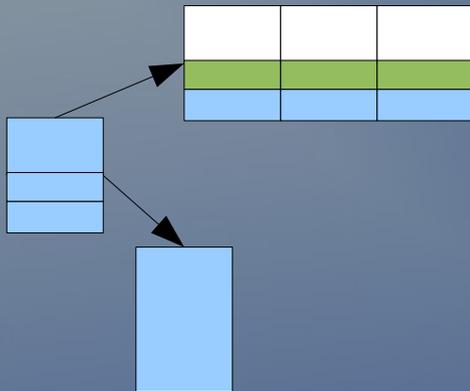
Internal
Type's
Descriptor

Values in memory



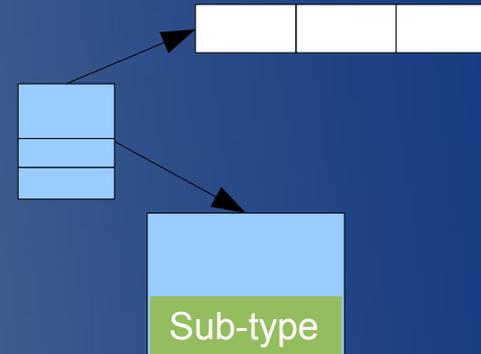
RAW Data storage

Undefined sub-type



```
type a array {3}  
type b list {3}  
type c hash  
type c dict
```

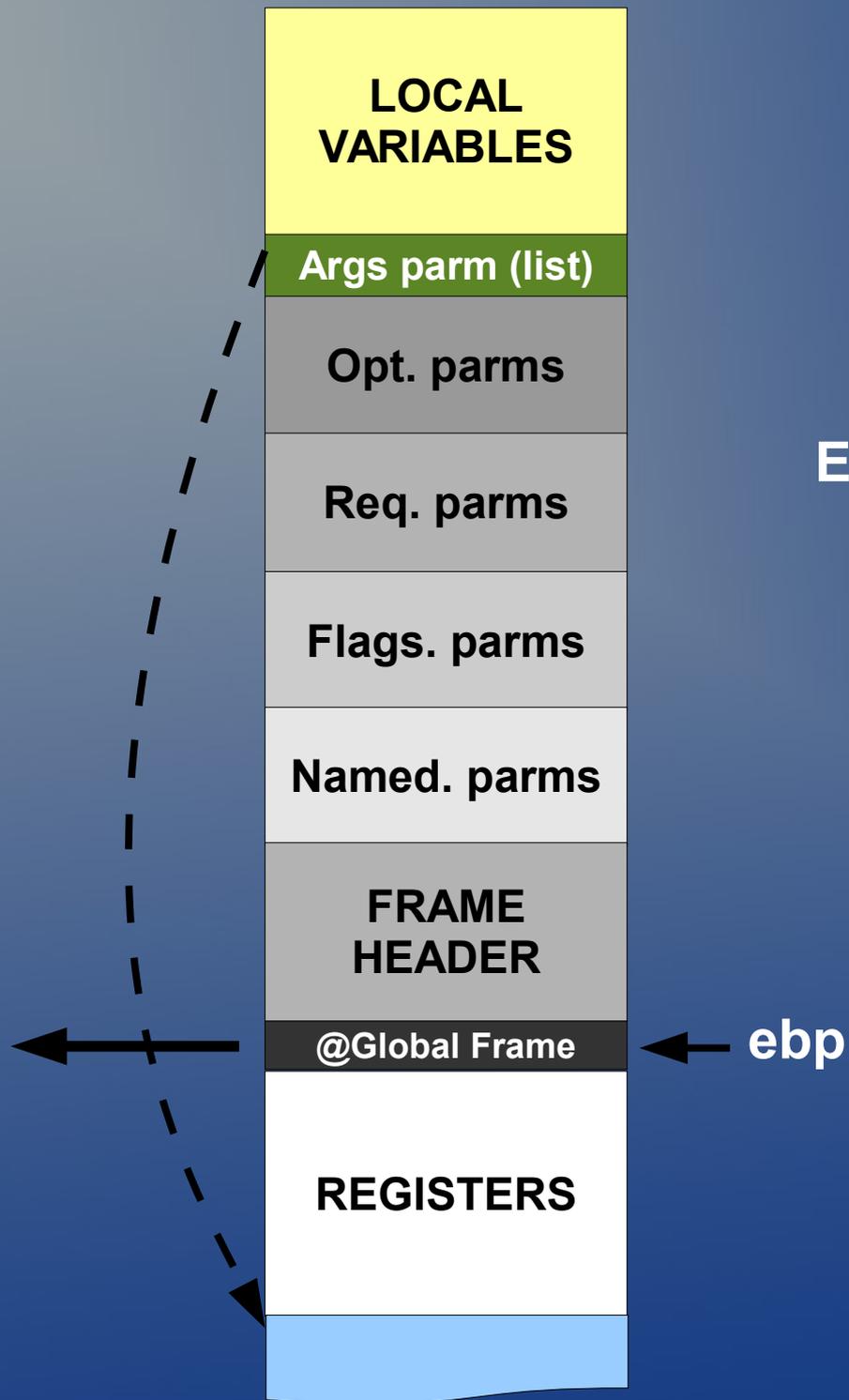
Defined sub-type



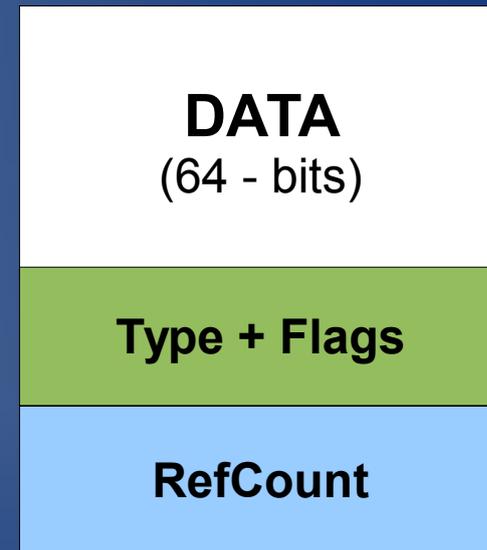
```
type a array {3 i32}  
type b list {3 s}  
type c hash {string}  
type c dict {i8}
```

```
type s struct {  
    i32:x  
    f64:y  
}
```

Execution frames



Each parm / register / local_variable:



Status

- Working “prototype”
- Initial performance tests show increases in performance between 2x and 25x vs. Tcl 8.6
- On-the-fly interpreter
- The possibility to run TyCL code in compiling time...this allows the modification of the whole compiler right at the spot.
- Experimenting with javascript as a target

To anyone interested ... in any way :)

aabuss@otlettech.com

aabuss@gmail.com