DYALOG

Glasgow 2024

Interpreter Limits

John Daintree

DYALOG

Elsinore 2023

An introduction to the workspace

Richard Smith

https://www.youtube.com/watch?v=2m-BO19OQPU

What you are about to see is based on the way Dyalog APL actually works.

Some dramatic licence has been taken and sequences have been shortened for simplicity.

(And not all limits and potential solutions will be discussed)

(And this is not an internal meeting)

DYALOG

Glasgow 2024

Key Technical Decisions During the Development of Dyalog APL

Geoff Streeter (Wednesday 14.25)

Limits? What limits

```
      ⍎('a'⍴⍨⌊⎕WA×0.15),'←1'
      ⍴⎕nl 2
1 39720486 ⍝(39 million!)
```

- Rank of array (15)
- Size of array
- Types of array (16)
- Depth of an array*
- Number of constants and names in code (4096)
- Function size (65536)
- Dfn depth (255)
- Number of primitives (256)
- Length of a symbol*
- Session input line / ⎕PW
- Parentheses depth (255)
- Number of lines in functions (4096)
- Control structure depth (255)

*Only limited by MAXWS

Limits? What limits?

- Rank of array (15)
- Size of array
- Types of array (16)
- Depth of an array*
- Number of constants and names in code (4096)
- Function size (65536)
- Dfn depth (255)
- Number of primitives (256)
- Length of a symbol*
- Session input line / □PW
- Parentheses depth (255)
- Number of lines in functions (4096)
- Control structure depth (255)

(32767) 'cos printers.

Limits? What limits?

- Rank of array (15)
- Size of array
- Types of array (16)
- Depth of an array*
- Number of constants and names in code (4096)
- Function size (65536)
- Dfn depth (255)
- Number of primitives (256)
- Length of a symbol*
- Session input line / □PW
- Parentheses depth (255)
- Number of lines in functions (4096)
- Control structure depth (255)

Limits? What limits?

- Rank of array (15)

```
      (⍳17)⍴0
LIMIT ERROR: Rank of resultant array would exceed maximum permitted
      (⍳17)⍴0
           ^
```

- Depth of an array
- Number of constants and names in code (4096)
- Function size (65536)
- Dfn depth (255)
- Number of primitives (256)
- Length of a symbol*
- Number of lines in functions (4096)
- Control structure depth (255)

Limits? What limits?

```
      ⎕fx src
1
```

- (15)
- Size of array
- Types of array (16)
- Depth of an array*
- Number of constants and names in code (4096)
- Function size (65536)
- Dfn depth (255)
- Number of primitives (256)
- Length of a symbol*
- Session input line / ⎕PW
- Parentheses depth (255)
- Number of lines in functions (4096)
- Control structure depth (255)

Limits? What limits?

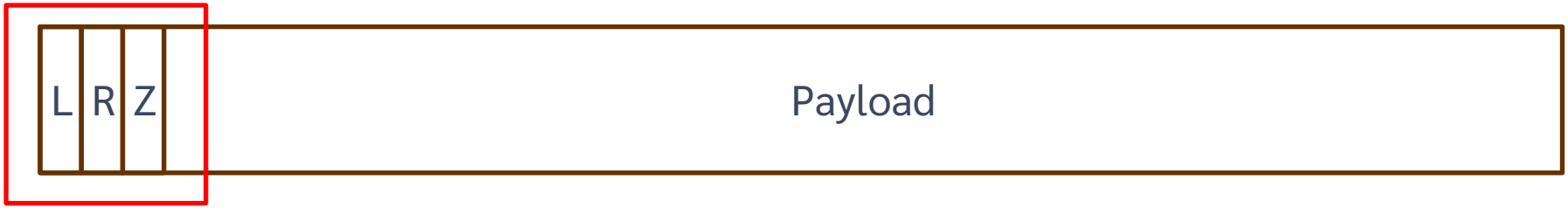
```
      (⎕fx ⍠ 1) src
LIMIT ERROR: Function contains too many constants or symbol references
      (⎕FX ⍠ 1) src
       ^
```

- Types of array (16)
- Depth of an array*
- Number of constants and names in code (4096)
- Function size (65536)
- Dfn depth (255)
- Number of primitives (256)
- Parentheses depth (255)
- Number of lines in functions (4096)
- Control structure depth (255)

What can we do to change...?

RANK (15)

Allocated pockets

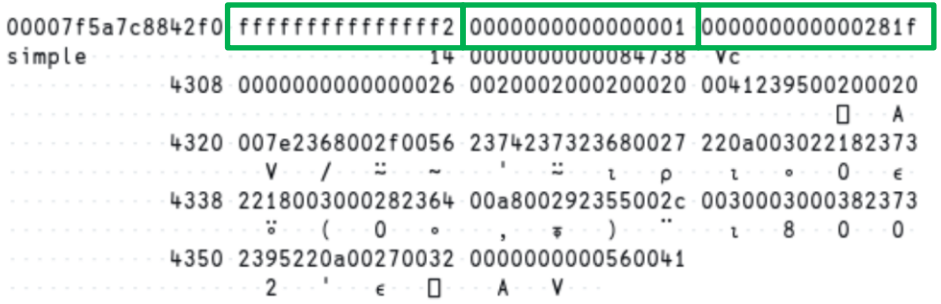


Allocated pockets

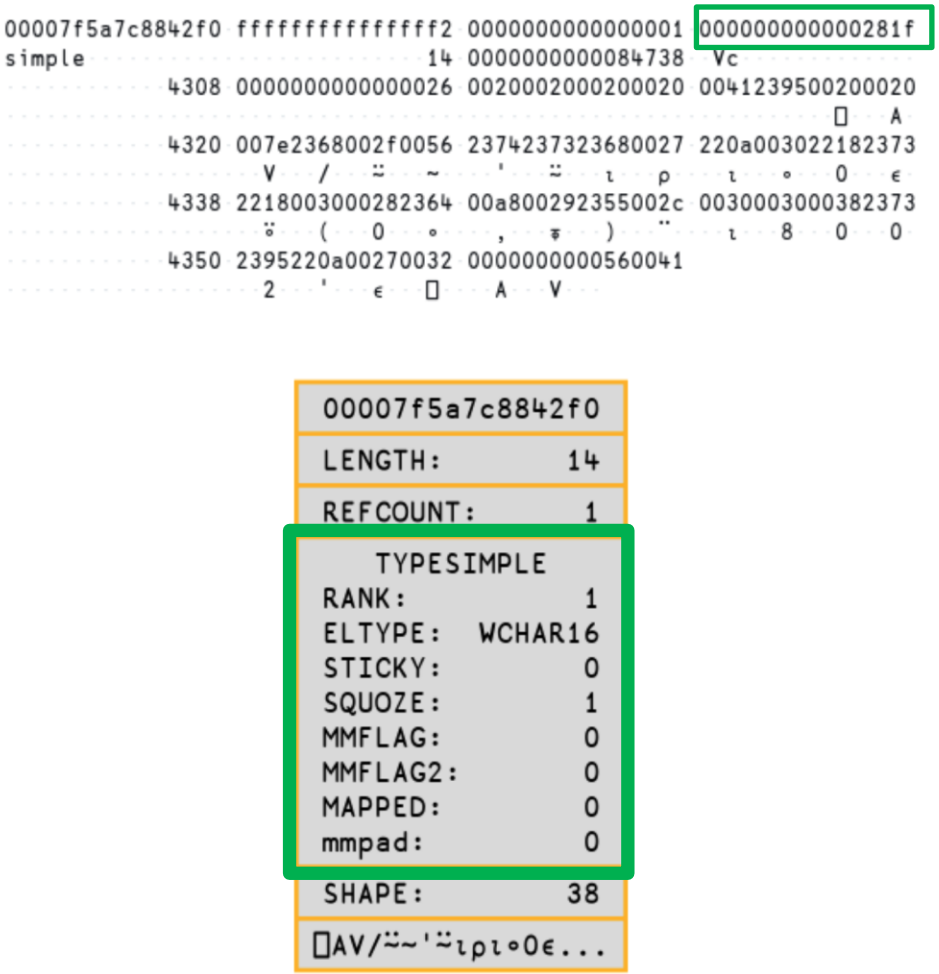
Allocated pockets

Allocated pockets

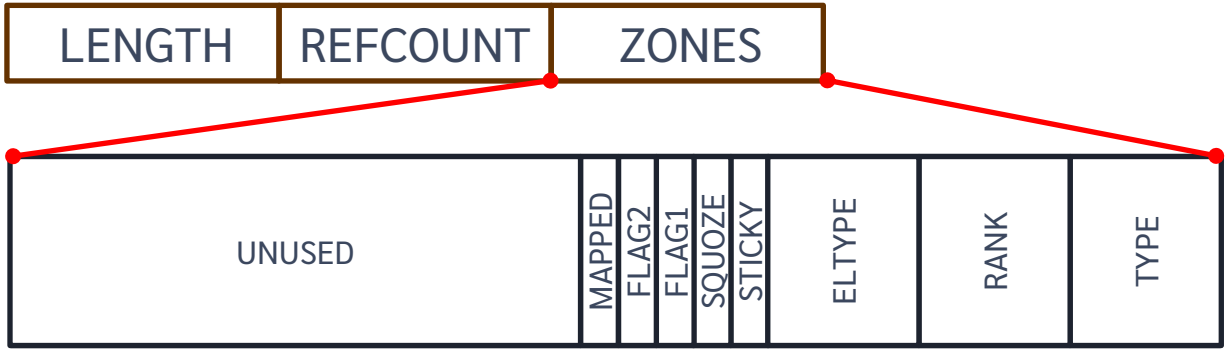
32 bits in a 32 bit interpreter
64 bits in a 64 bit interpreter (but, crucially only 32 are used in ZONES)


00007f5a7c8842f0 ffffffffffffffff2 0000000000000001 000000000000281f
simple 14 0000000000084738 Vc
4308 0000000000000026 0020002000200020 0041239500200020
4320 007e2368002f0056 2374237323680027 220a003022182373
4338 2218003000282364 00a800292355002c 0030003000382373
4350 2395220a00270032 0000000000560041

```
00007f5a7c8842f0 fffffffffffffff2 0000000000000001 000000000000281f
simple                         14 0000000000084738  Vc
            4308 0000000000000026 0020002000200020 0041239500200020
                                                                 □  A
            4320 007e2368002f0056 2374237323680027 220a003022182373
                   V   /   ¨   ~   '   ¨   ι   ρ   ι   ∘   0   ϵ
            4338 2218003000282364 00a800292355002c 0030003000382373
                   ¨   (   0   ∘   ,   ∓   )   ¨   ι   8   0   0
            4350 2395220a00270032 0000000000560041
                   2   '   ϵ   □   A   V
```


00007f5a7c8842f0 ffffffffffffff2 0000000000000001 000000000000281f
simple 14 0000000000084738 Vc
4308 0000000000000026 0020002000200020 0041239500200020
A
4320 007e2368002f0056 2374237323680027 220a003022182373
V / ~ ' ι ρ ι ∘ 0 ϵ
4338 2218003000282364 00a800292355002c 0030003000382373
(0 ∘ ,) ι 8 0 0
4350 2395220a00270032 0000000000560041
2 ' ϵ A V
00007f5a7c8842f0
LENGTH: 14
REFCOUNT: 1
TYPESIMPLE
RANK: 1
ELTYPE: WCHAR16
STICKY: 0
SQUOZE: 1
MMFLAG: 0
MMFLAG2: 0
MAPPED: 0
mmpad: 0
SHAPE: 38
AV/~'ιρι∘0ϵ...

Array pockets

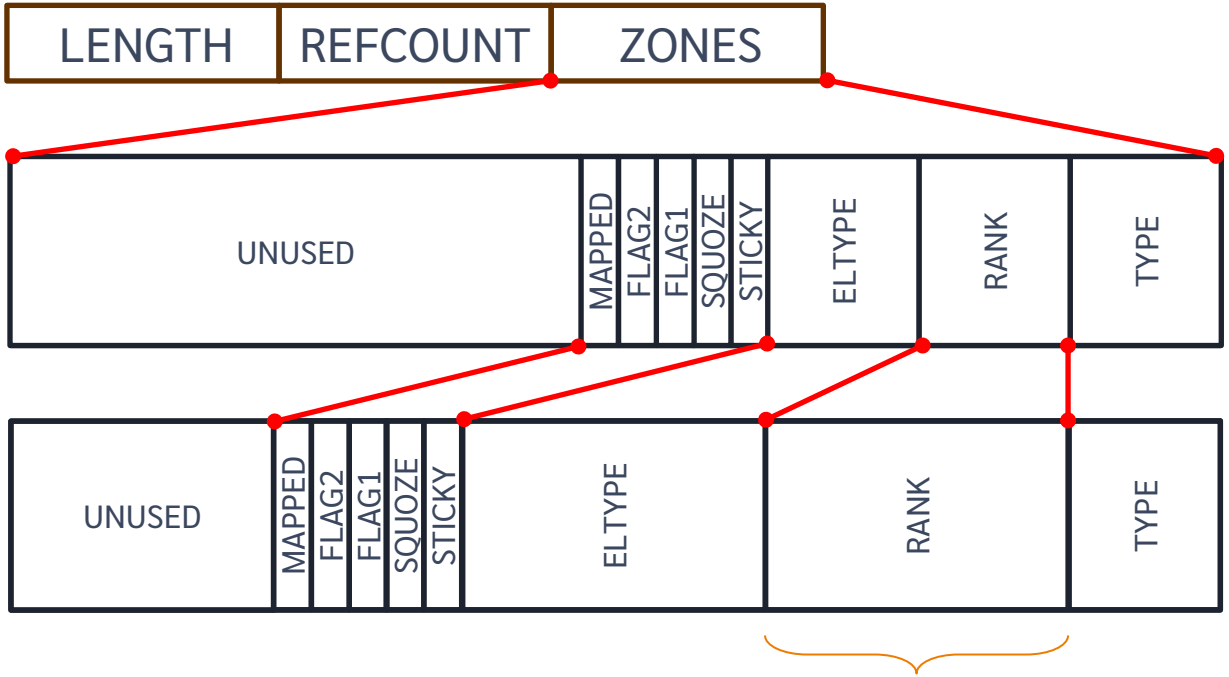


4 bits wide
Range 0-15
Only 16 POCKET types

4 bits wide
Range 0-15
Max 16 element types

4 bits wide
Range 0-15
Max RANK 15

Array pockets



And, we've added room for more element types

8 bits wide
Range 0-255
Max RANK 255

Interpreter Limits

It's easy then, innit?

- No
  - Old Workspaces
  - Old Component Files
  - Old serialized data (sent down a socket)
  - Saving arrays to component files
    - For interpreters "from the past"

Saving arrays to component files

- Arrays on component files are interchangeable between old and new interpreters.

Saving arrays to component files

- "By default", compatible arrays on component files will be interchangeable between old and new interpreters.

Richard Smith

What can we do to change...?

Number of constants and names in code (4096)

Symbols / Constants limit

```
:if X ◇ 1 2 3 ◇ :endif
```

Symbols / Constants limit

```
:if X ⋄ 1 2 3 ⋄ :endif
```

Symbols / Constants limit

```
:if X ⋄ 1 2 3 ⋄ :endif
```

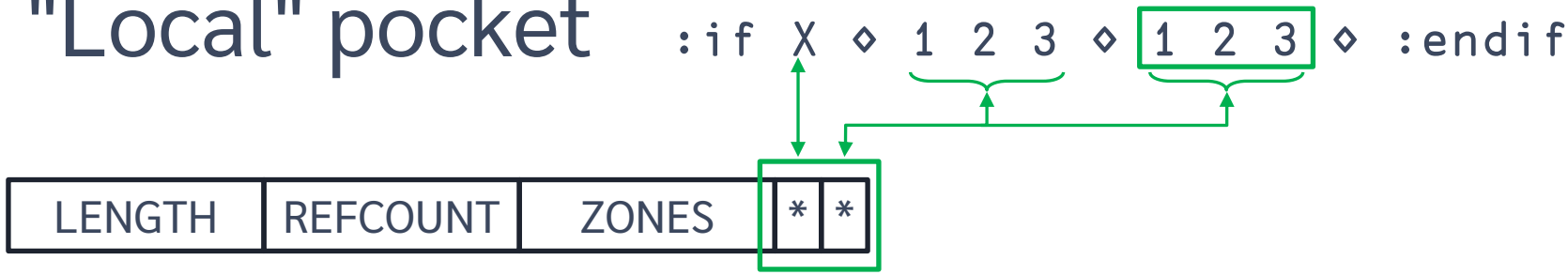
"Local" pocket

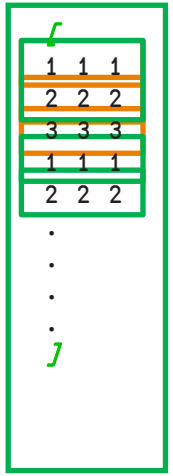
"Local" pocket

"Local" pocket

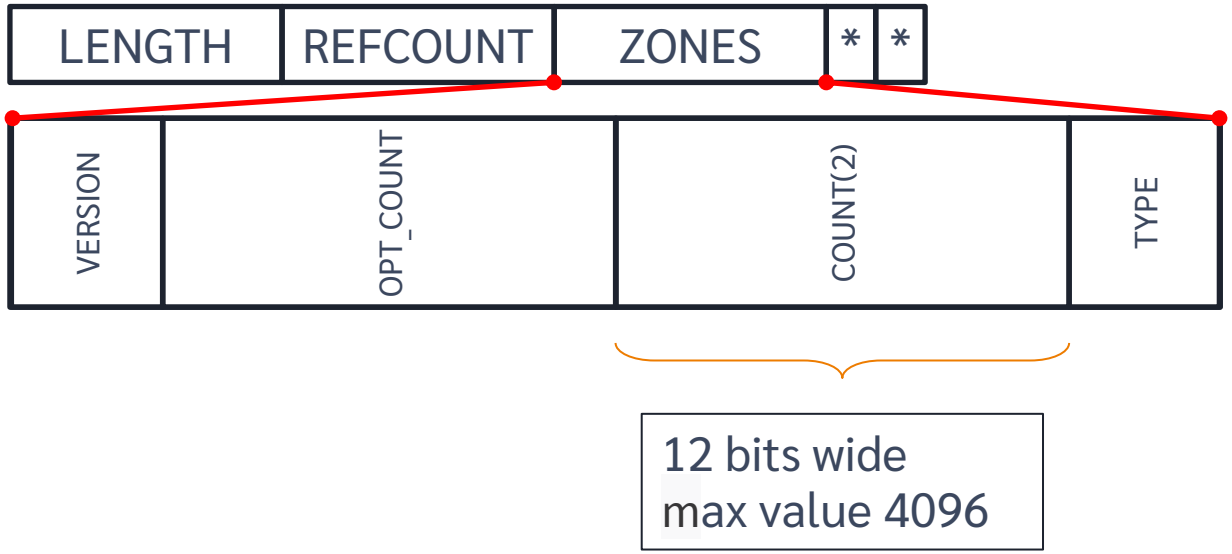
```
:if X ◇ 1 2 3 :endif
```

"Local" pocket

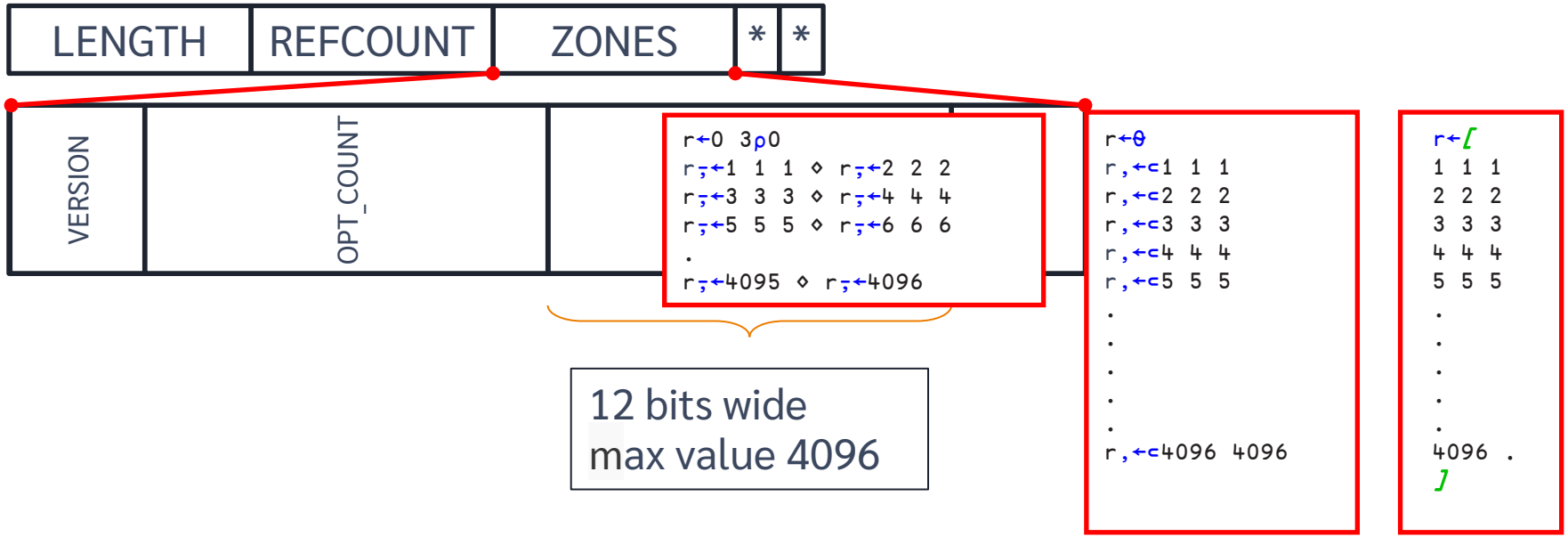




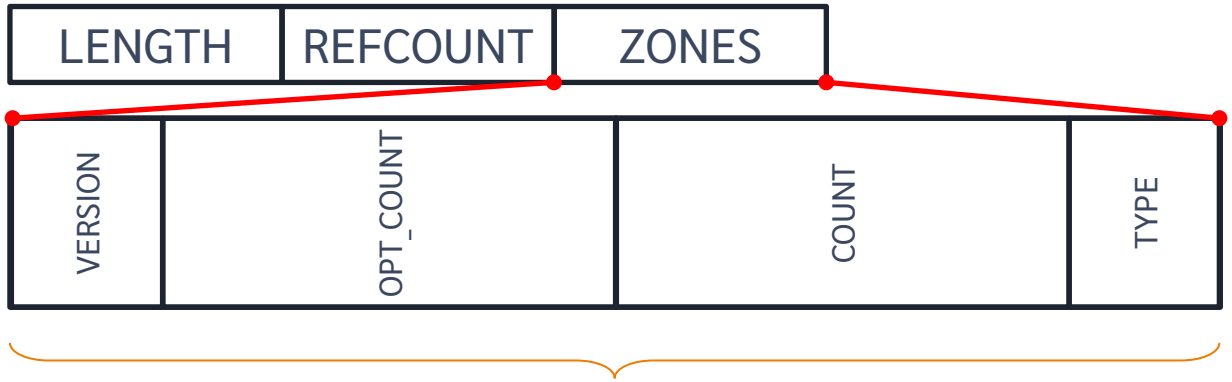
"Local" pocket `:if X ◇ 1 2 3 ◇ :endif`



"Local" pocket `:if X ◇ 1 2 3 ◇ :endif`

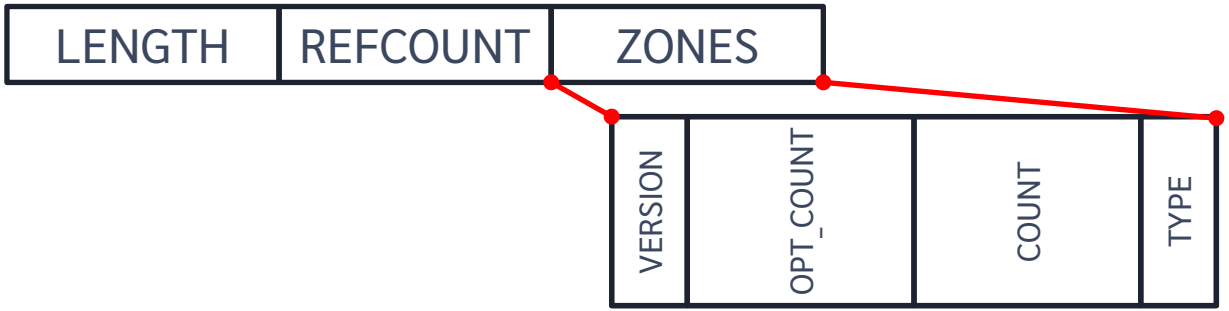


"Local" pocket

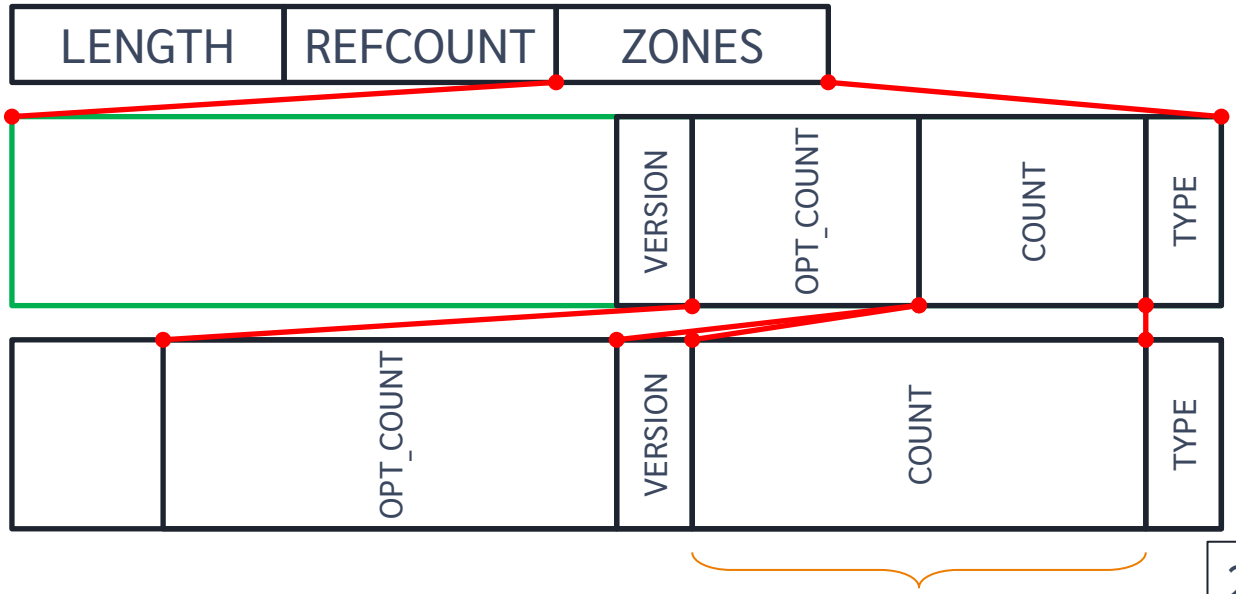


32 bits wide.
It's "full"

"Local" pocket



"Local" pocket

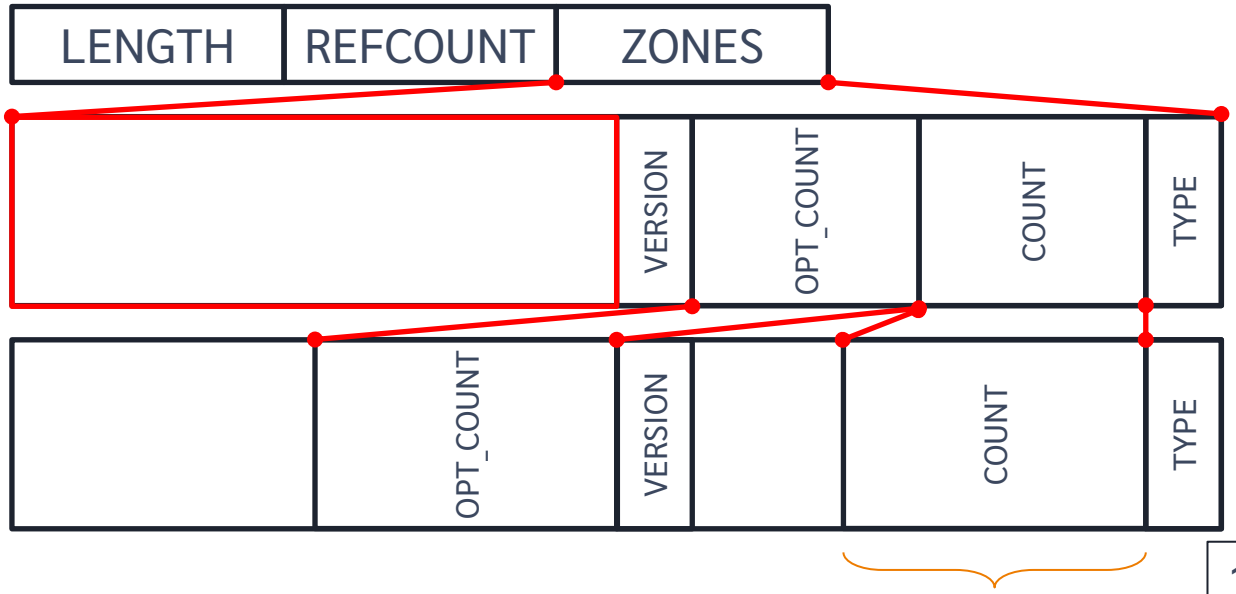


But that's "too big"
(details later)

24 bits wide
max value 16777216

"Local" pocket

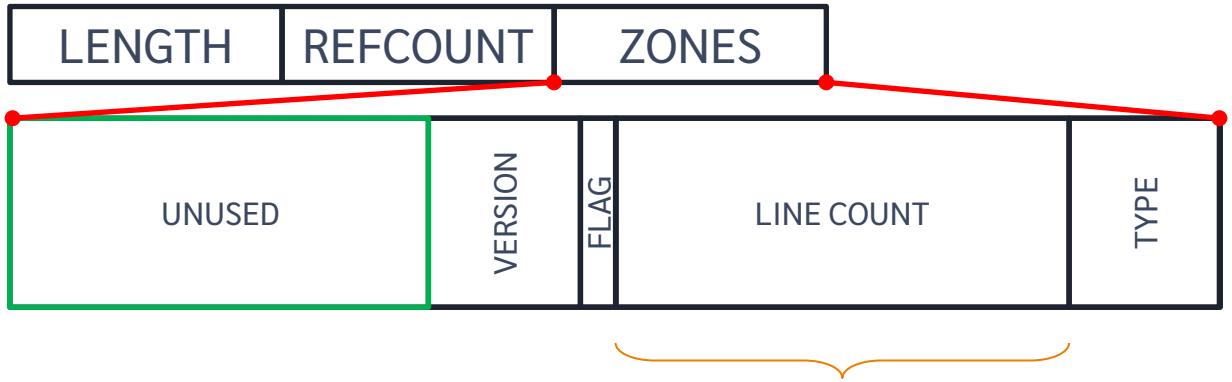
```
  grep ZONESBITFIELD *.h| grep -v "\<1\>" | wc
142     752    6648
```



But 32 bit workspaces will get bigger.

16 bits wide
max value 65536

"Body" pocket



Number of lines in functions (4096)

12 bits wide
max value 4096

It's easy then, innit?

- No
  - Old Workspaces
  - Old Component Files
  - Old serialized data (sent down a socket)

Constants/Symbols limit

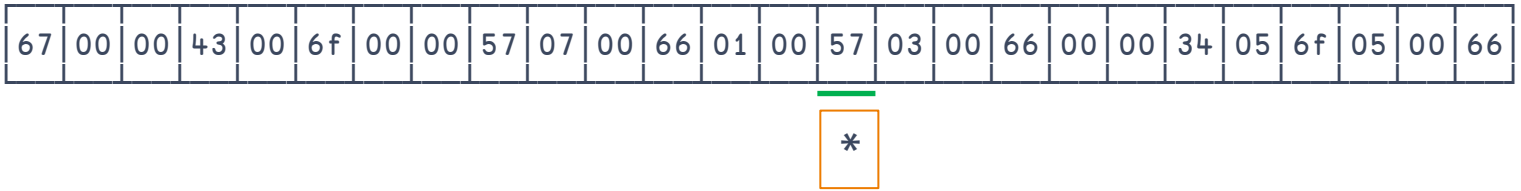
Part 2

Symbols / Constants limit

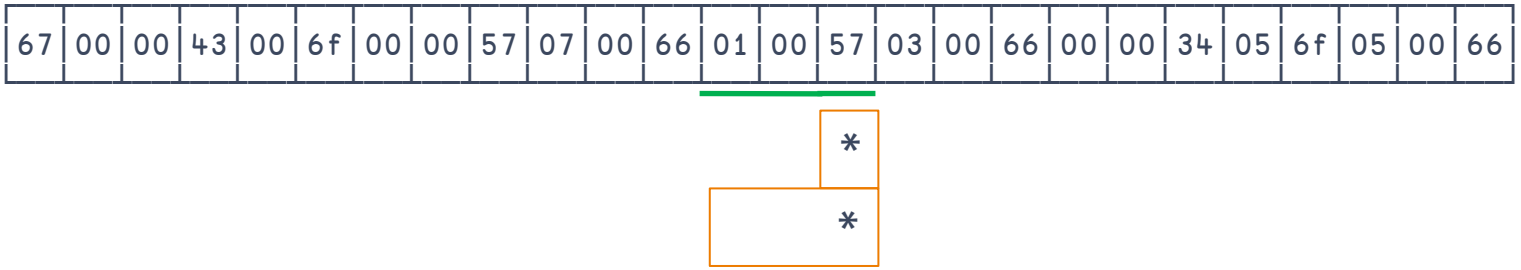
```
:if X ⋄ 1 2 3 ⋄ :endif
```

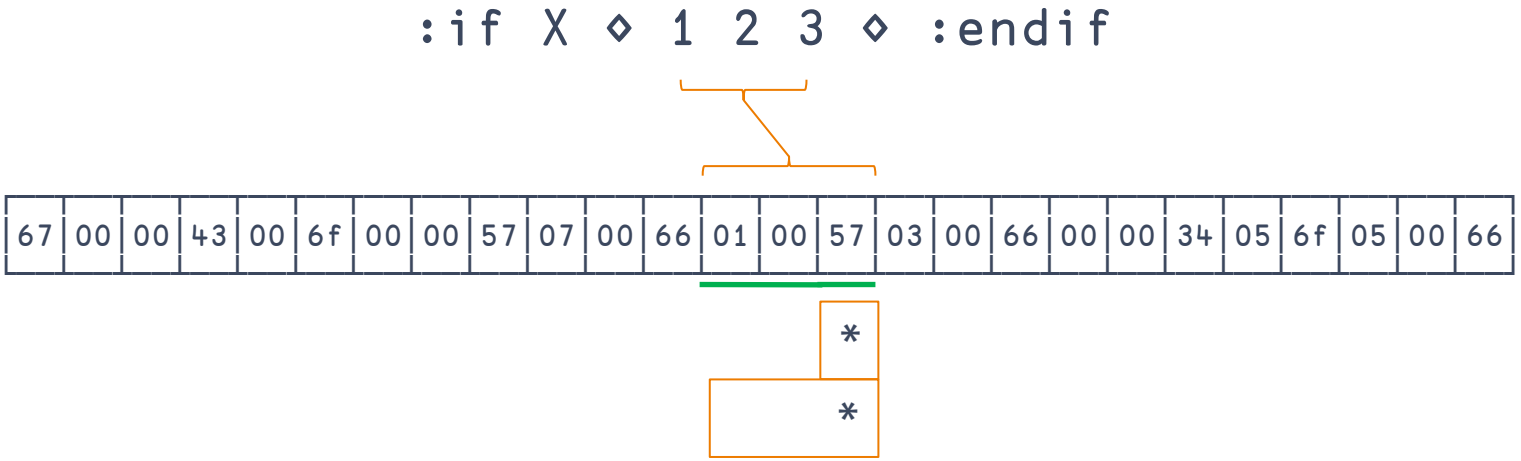
```
:if X ◇ 1 2 3 ◇ :endif
```

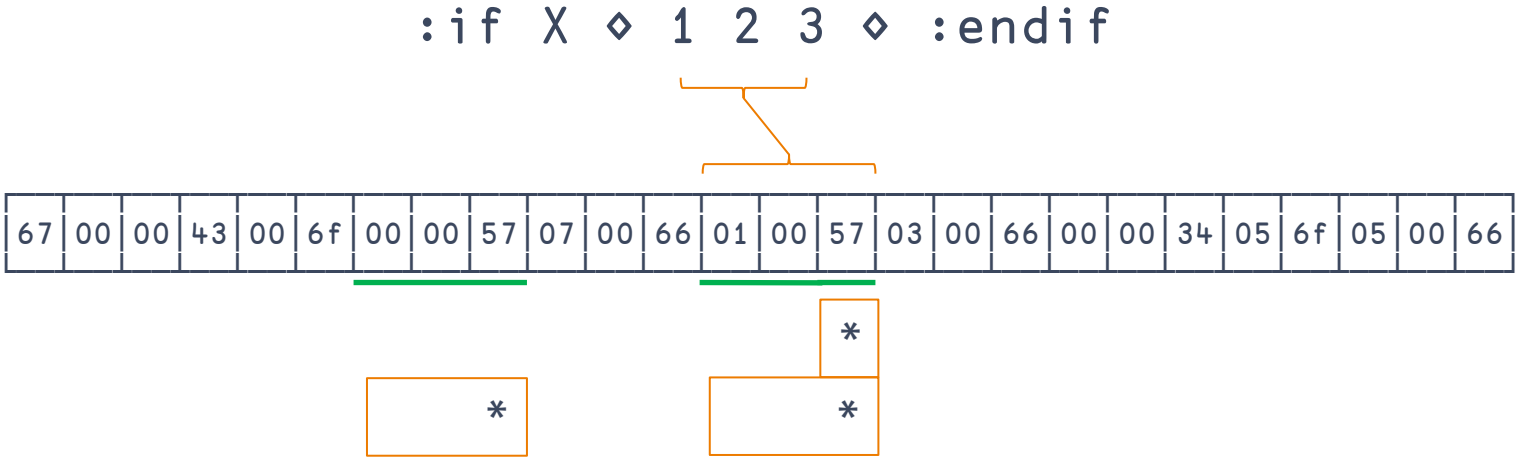
```
:if X ◇ 1 2 3 ◇ :endif
```



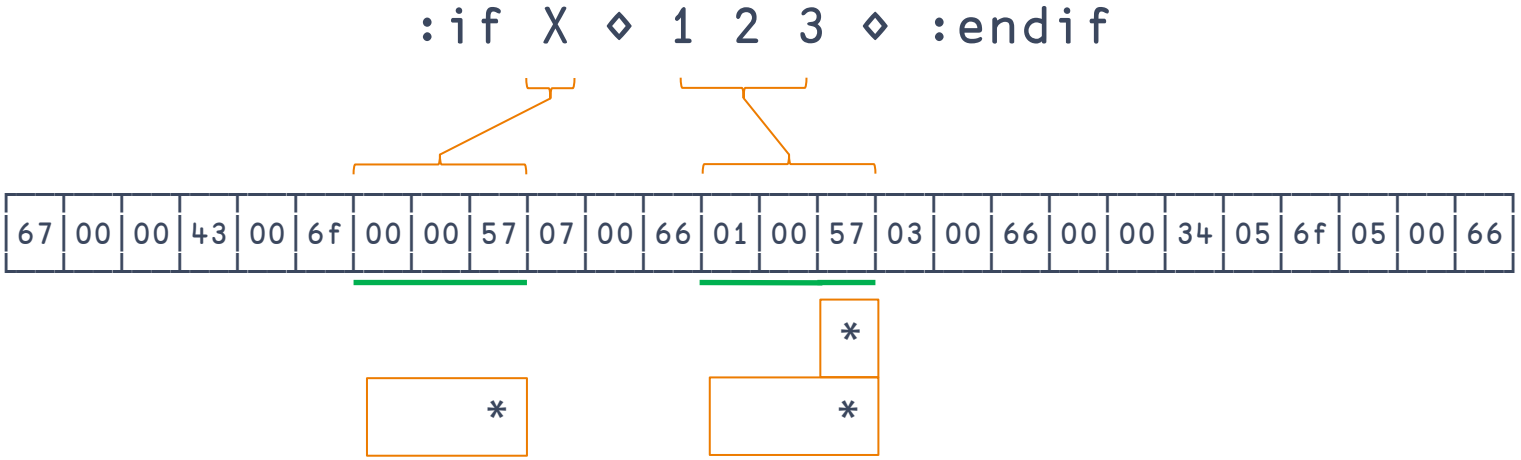
```
:if X ◇ 1 2 3 ◇ :endif
```



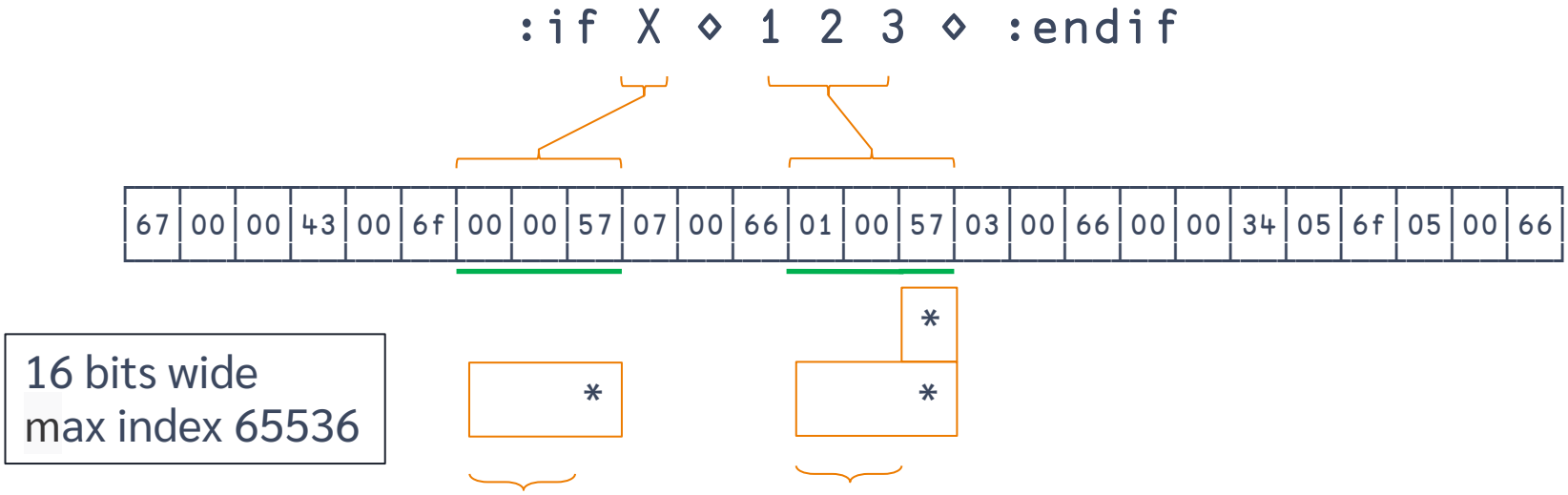

:if X ◇ 1 2 3 ◇ :endif
67 00 00 43 00 6f 00 00 57 07 00 66 01 00 57 03 00 66 00 00 34 05 6f 05 00 66
*
*

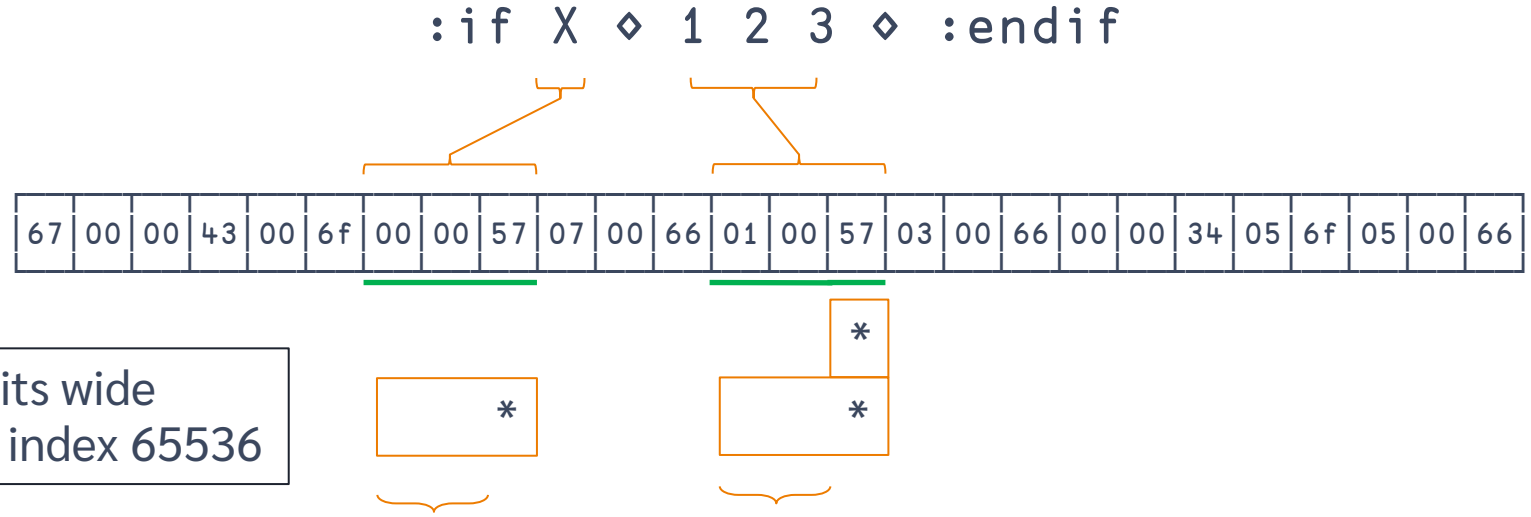


Interpreter Limits



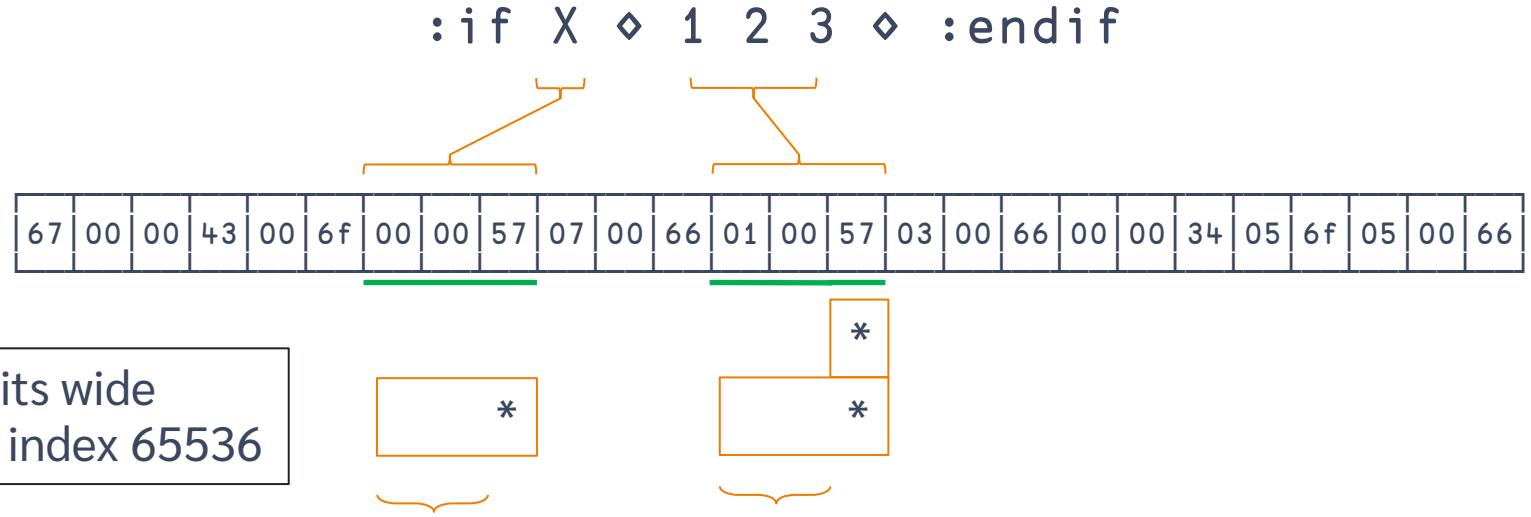
Interpreter Limits


:if X ◇ 1 2 3 ◇ :endif
67 00 00 43 00 6f 00 00 57 07 00 66 01 00 57 03 00 66 00 00 34 05 6f 05 00 66
*
*
*
16 bits wide
max index 65536



16 bits wide
max index 65536

But we currently only use 12 bits

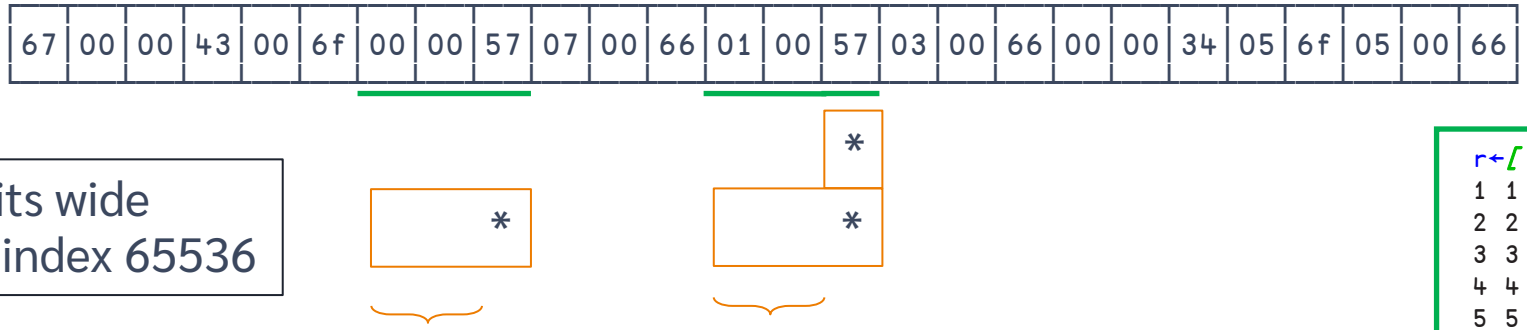


16 bits wide
max index 65536

But we currently only use 12 bits

So
4096->65536
is "doable"

```
:if X ◇ 1 2 3 ◇ :endif
```



16 bits wide
max index 65536

But we currently only use 12 bits

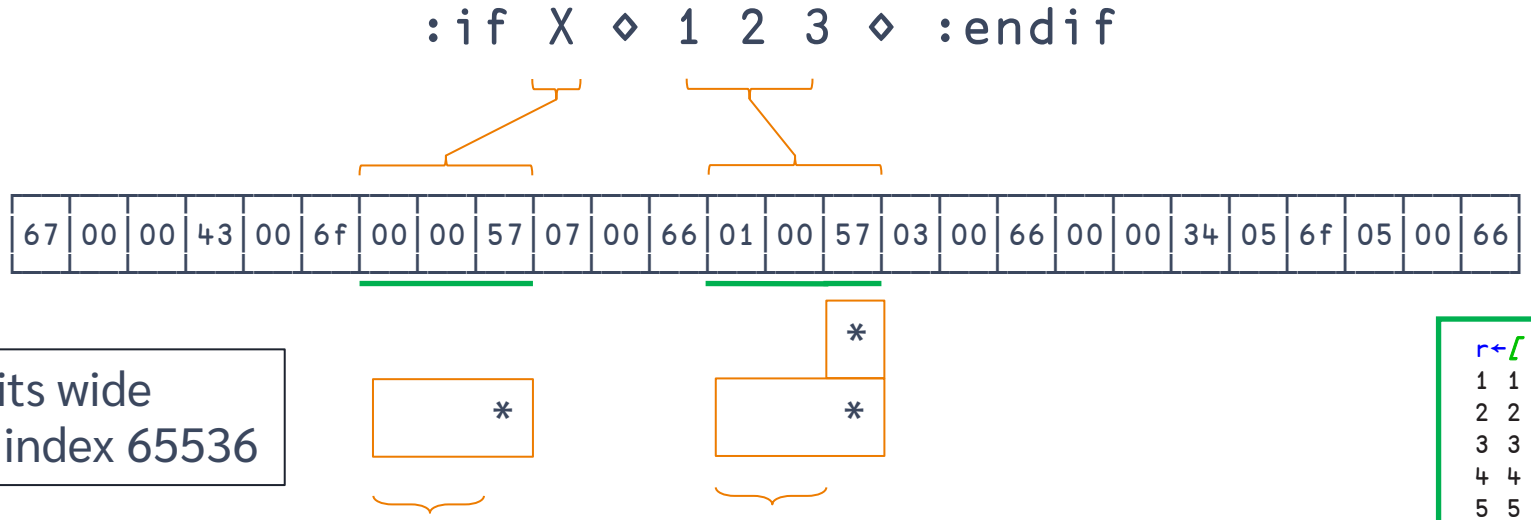
So
4096->65536
is "doable"

```
r←⍳
1 1 1
2 2 2
3 3 3
4 4 4
5 5 5
.
.
.
.
.
4096 .

⍝
```


Interpreter Limits

DYALOG



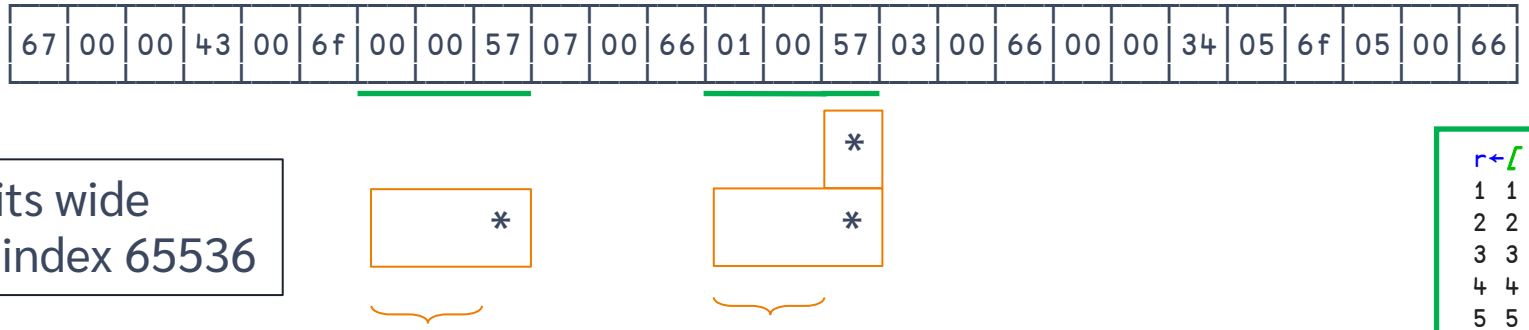
16 bits wide
max index 65536

But we currently only use 12 bits

So
4096->65536
is "doable"

```
r←⍳
1 1 1
2 2 2
3 3 3
4 4 4
5 5 5
.
.
.
.
65536
⍴r
```

```
:if X ◇ 1 2 3 ◇ :endif
```



16 bits wide
max index 65536

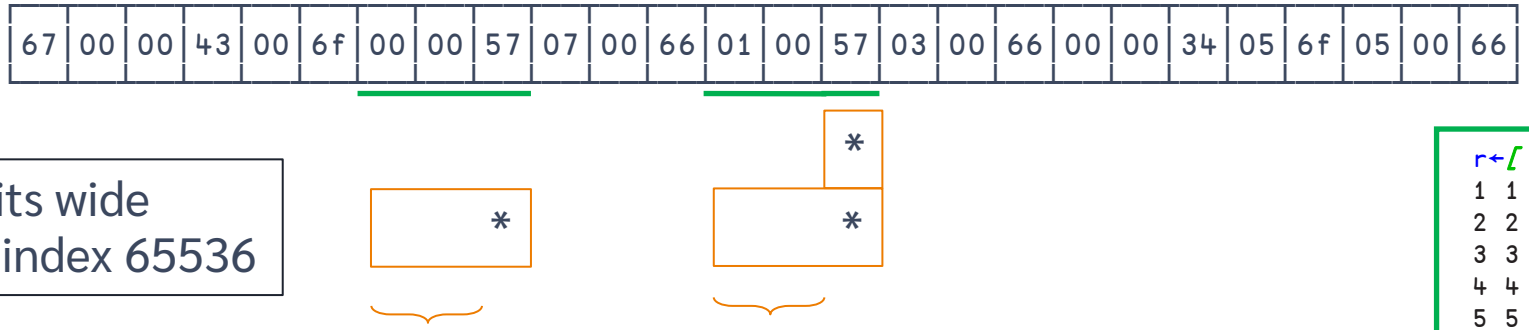
But we currently only use 12 bits

So
4096->65536
is "doable"

But not enough room for more

```
r←⍳
1 1 1
2 2 2
3 3 3
4 4 4
5 5 5
.
.
.
.
65536
⍴
```

```
:if X ◇ 1 2 3 ◇ :endif
```



16 bits wide
max index 65536

But we currently only use 12 bits

So
4096->65536
is "doable"

But not enough room for more

```
r←⍳
1 1 1
2 2 2
3 3 3
4 4 4
5 5 5
.
.
.
.
65536
⍴
```


Interpreter Limits

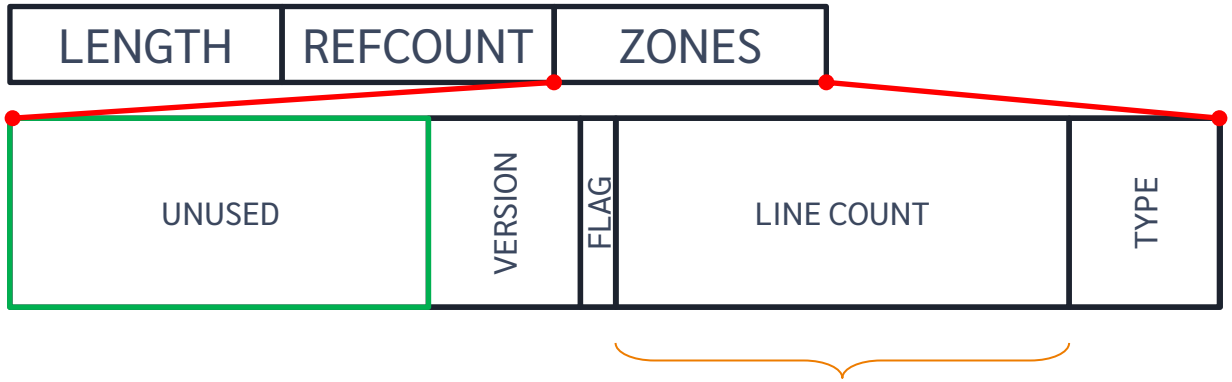
DYALOG

"Migrants with functions of >9,999 lines"

Limits? What limits?

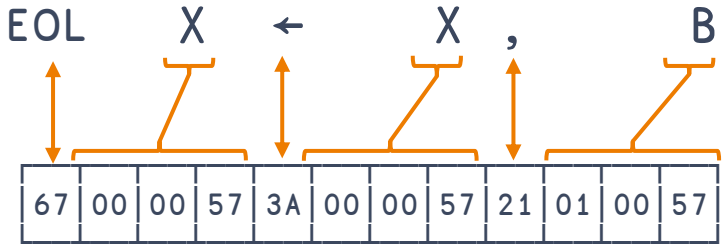
- Rank of array (15)
- Size of array
- Types of array (16)
- Depth of an array*
- Number of constants and names in code (4096)
- Function size (65536)
- Dfn depth (255)
- Number of primitives (256)
- Length of a symbol*
- Session input line / □PW
- Parentheses depth (255)
- Number of lines in functions (4096)
- Control structure depth (255)

"Body" pocket



Number of lines in functions (4096)

12 bits wide
max value 4096

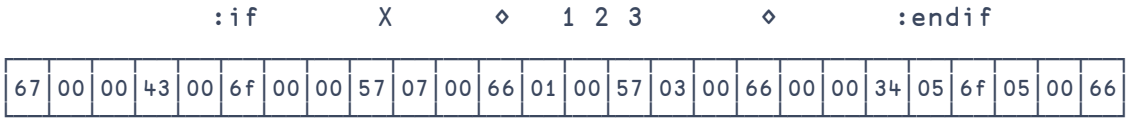


- 12 tokens (bytes) for a "minimal line"
- Maximum Function size is 65536
- So, 10000 such lines is "too many tokens"

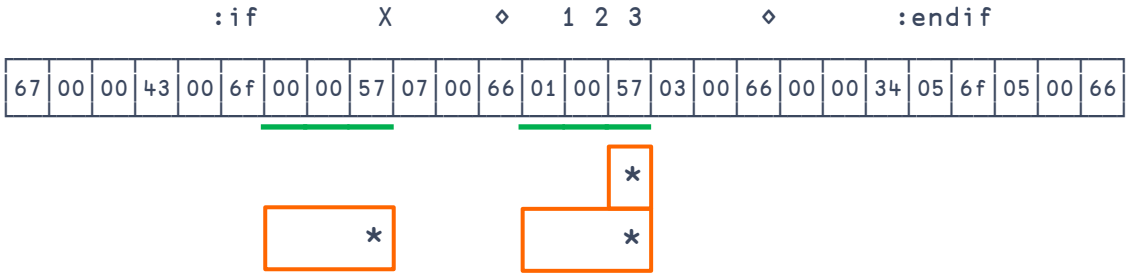
What can we do to change...?

Function Too Large

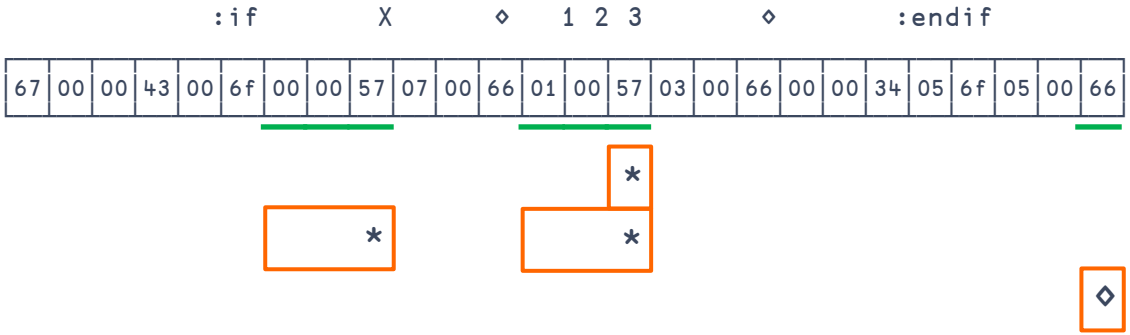
```
:if X ◇ 1 2 3 ◇ :endif
```



Interpreter Limits

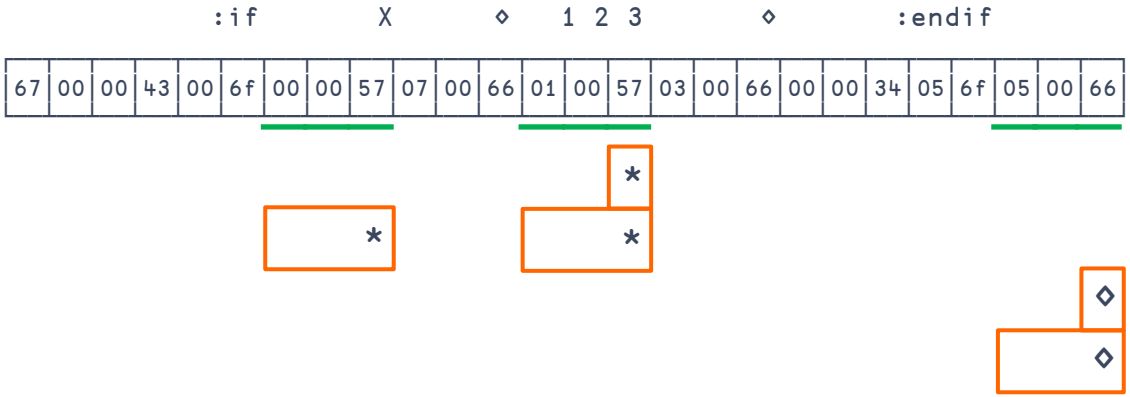


Interpreter Limits



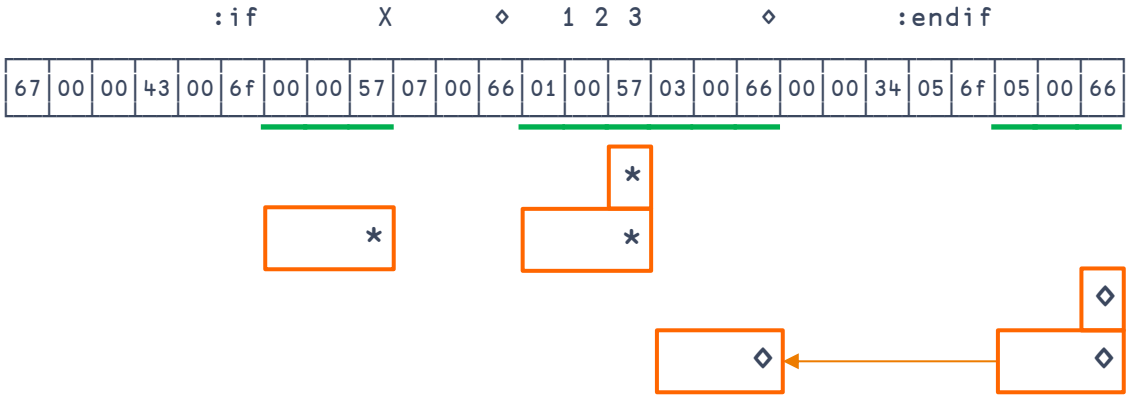
Interpreter Limits

DYALOG

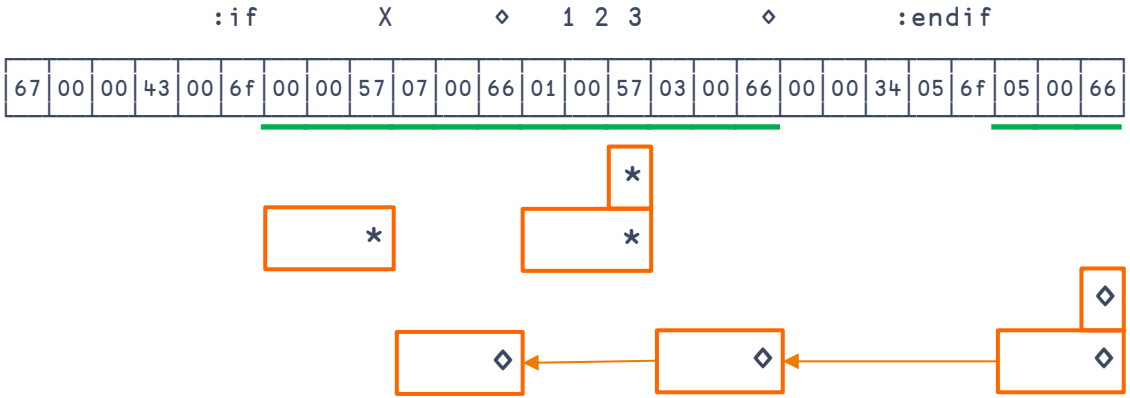


Interpreter Limits

DYALOG

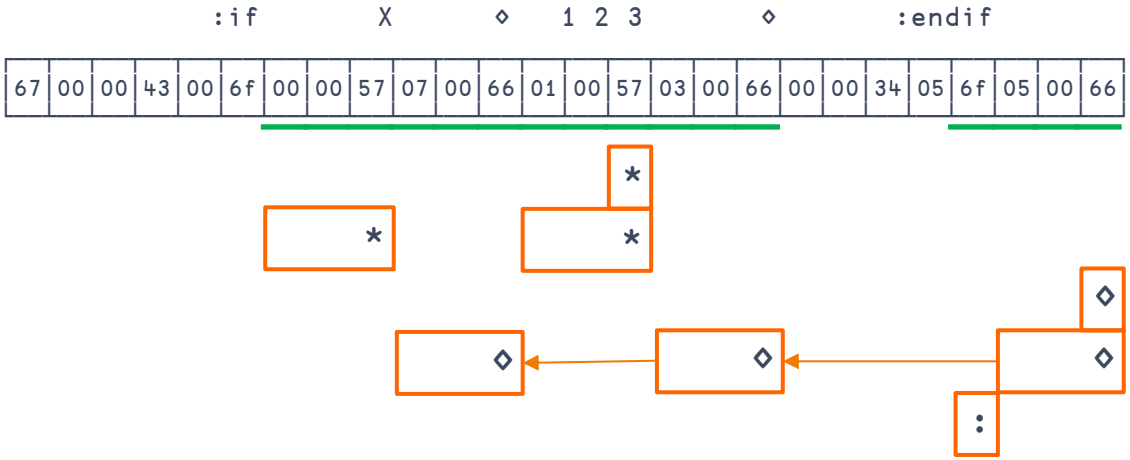


Interpreter Limits

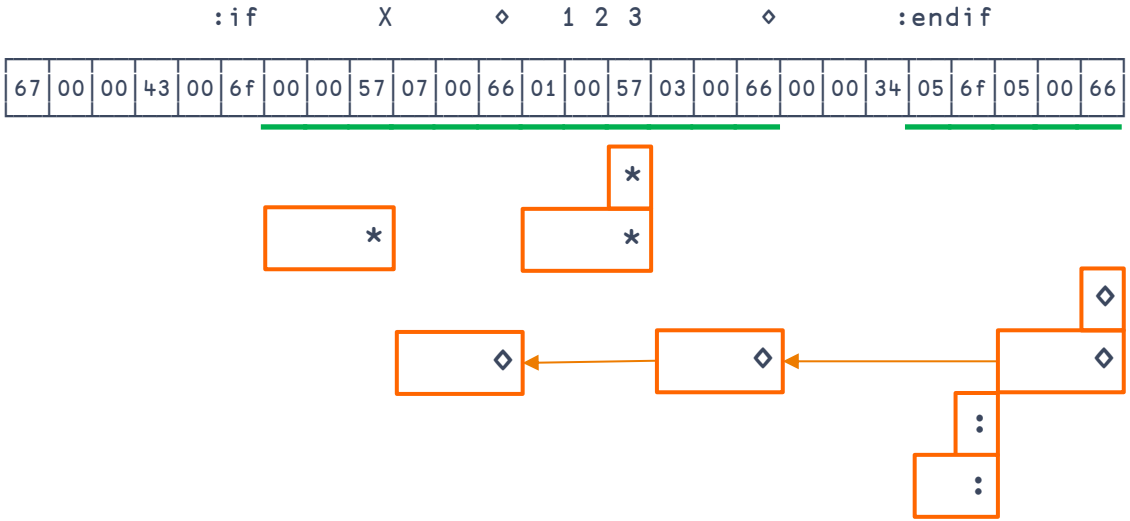


Interpreter Limits

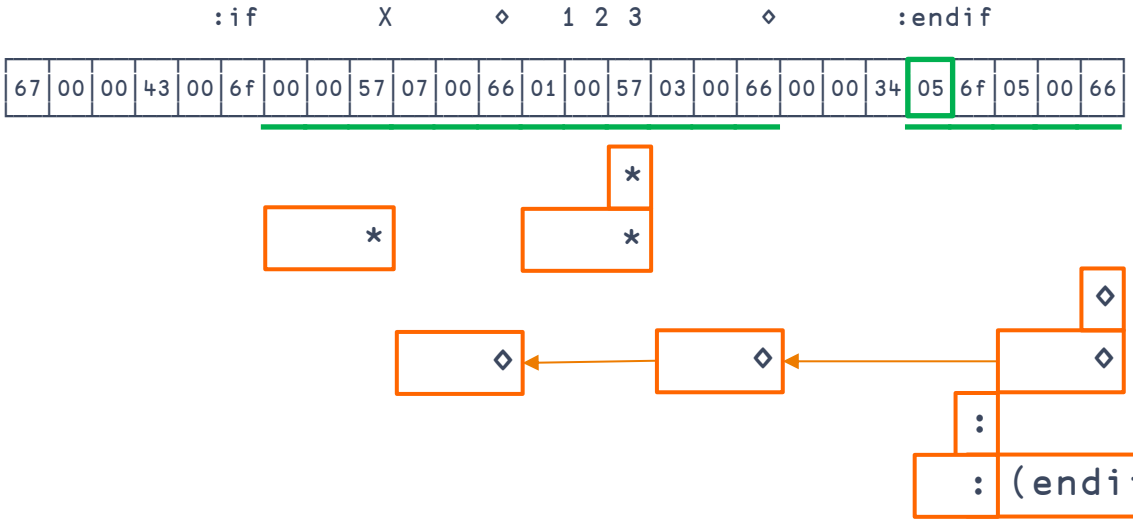
DYALOG



Interpreter Limits



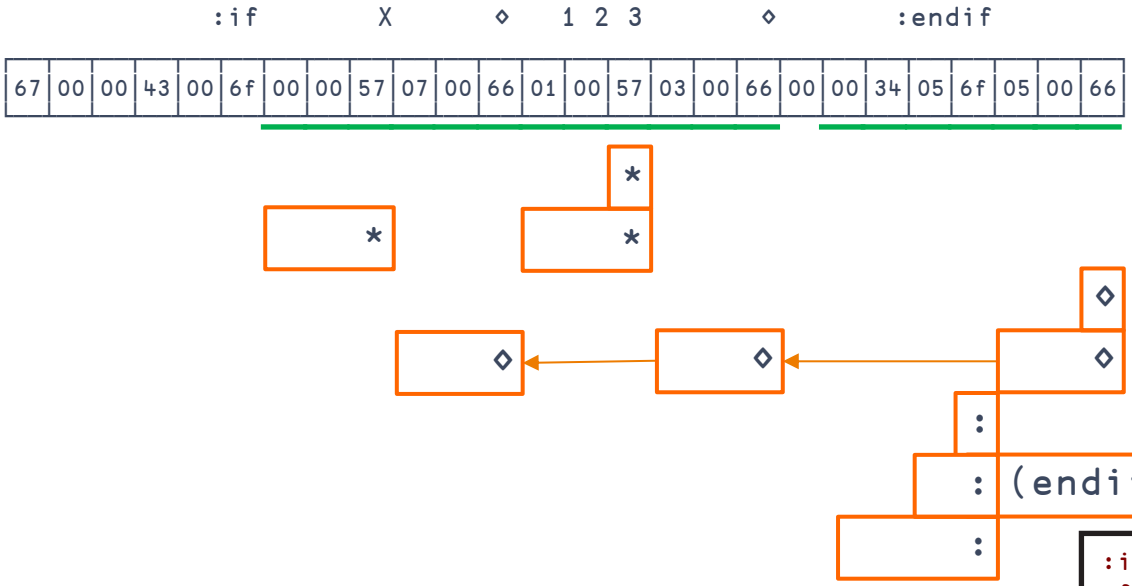
Interpreter Limits




```
:if / :else / :endif
:for / :in / :end
{ α← : :: ◇ }
( ◇ : )
[ ◇ ]
...
```

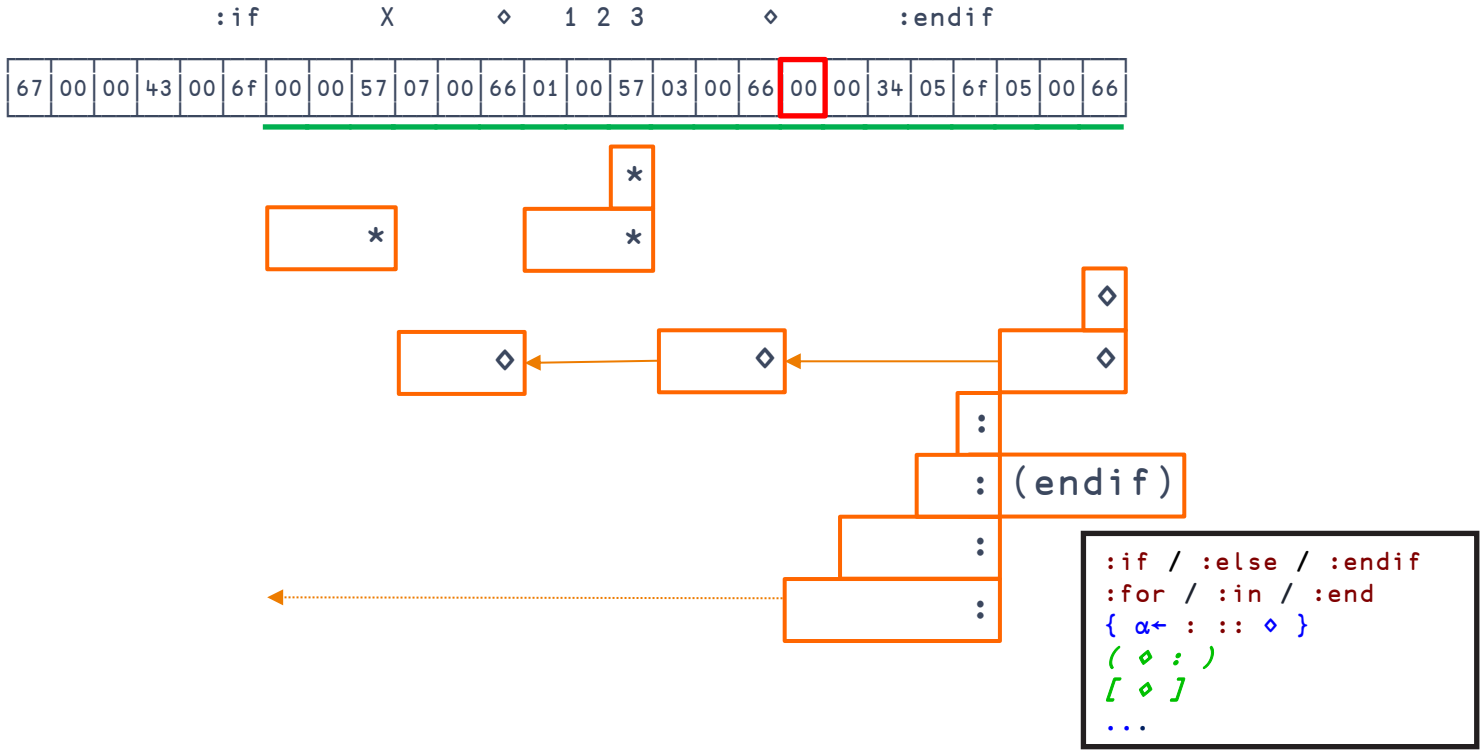

Interpreter Limits

DYALOG



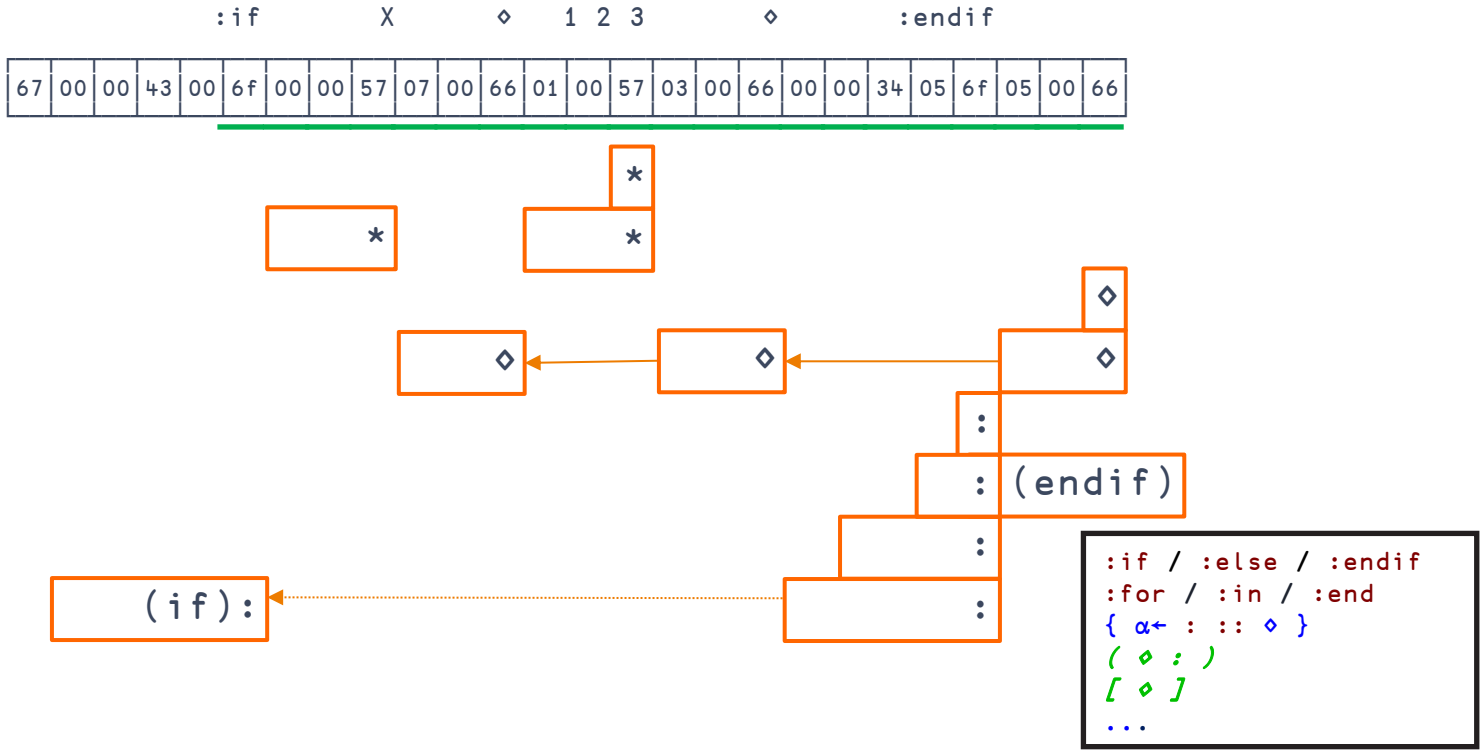
Interpreter Limits

DYALOG

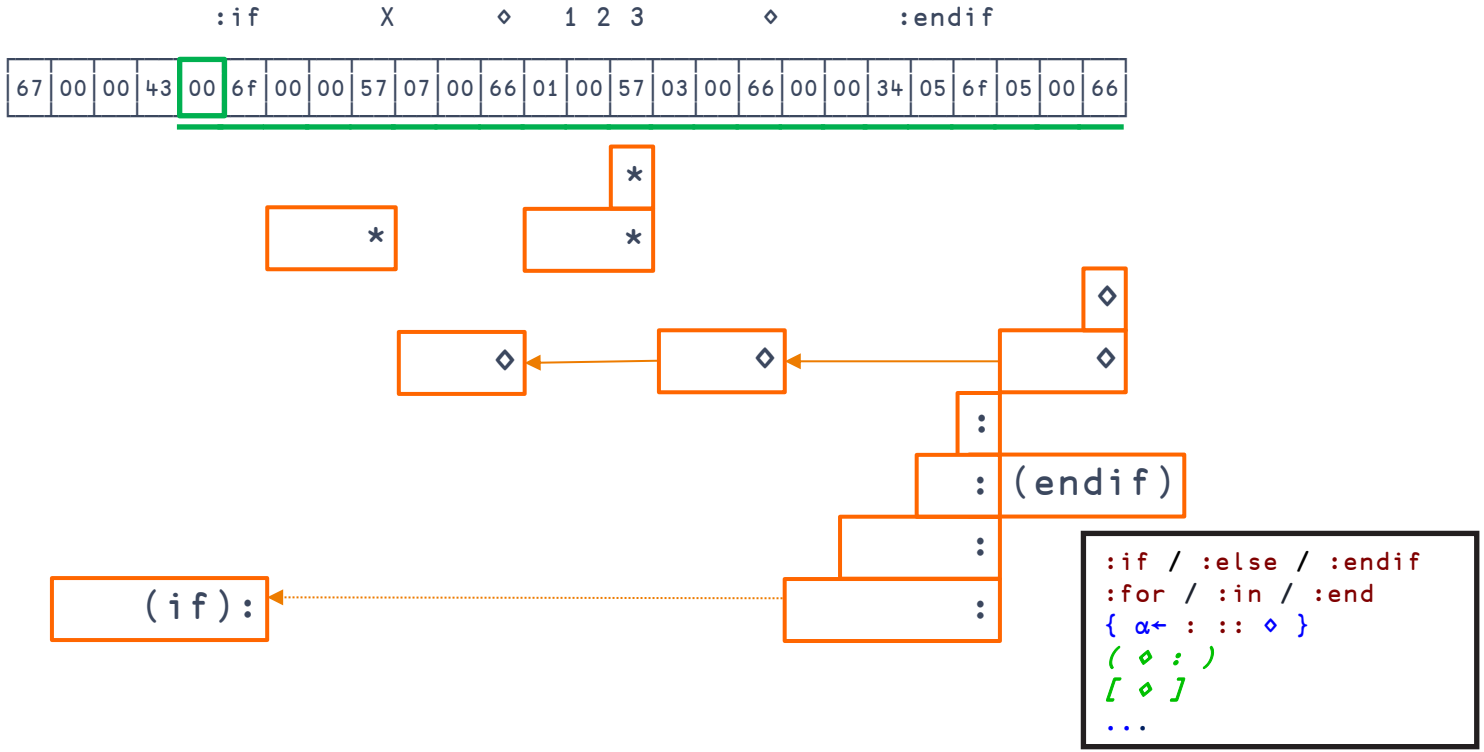


Interpreter Limits

DYALOG

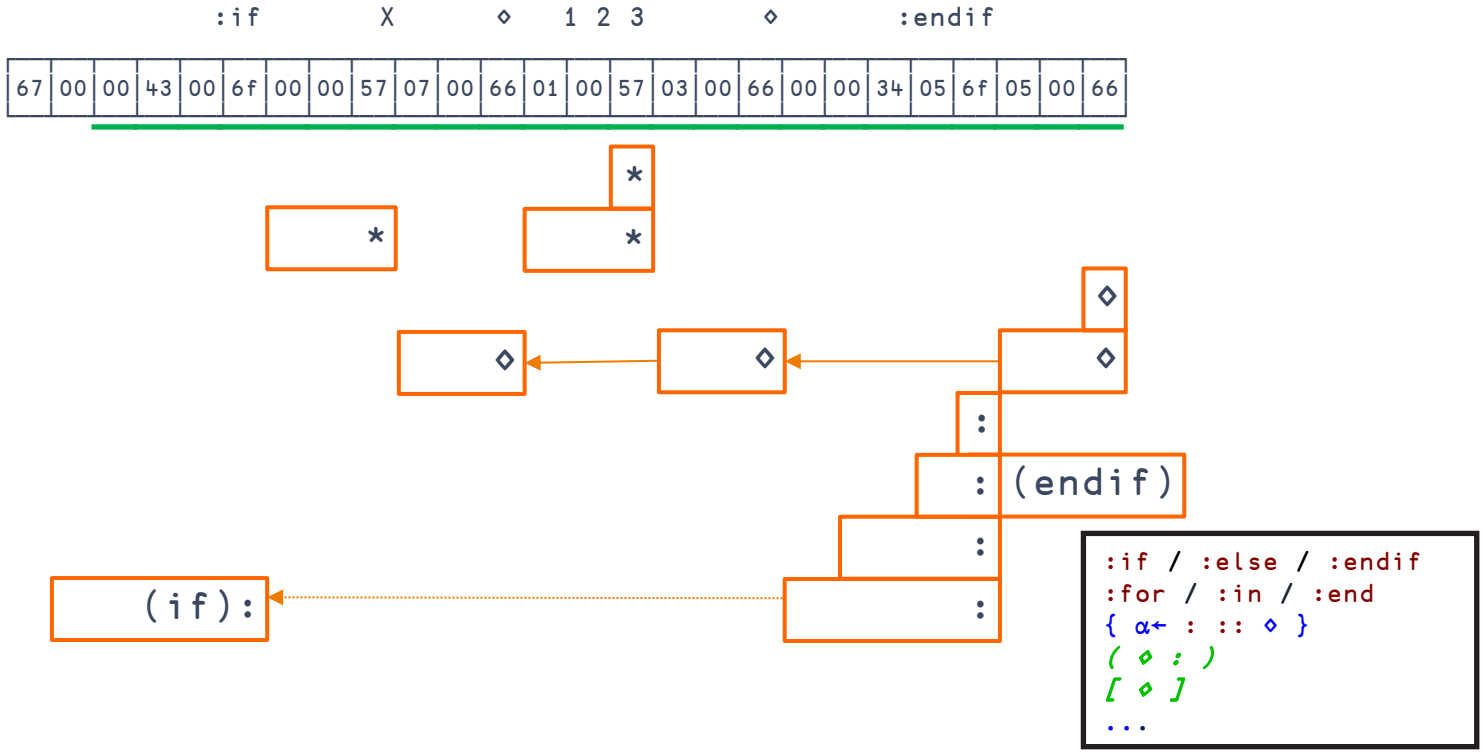


Interpreter Limits



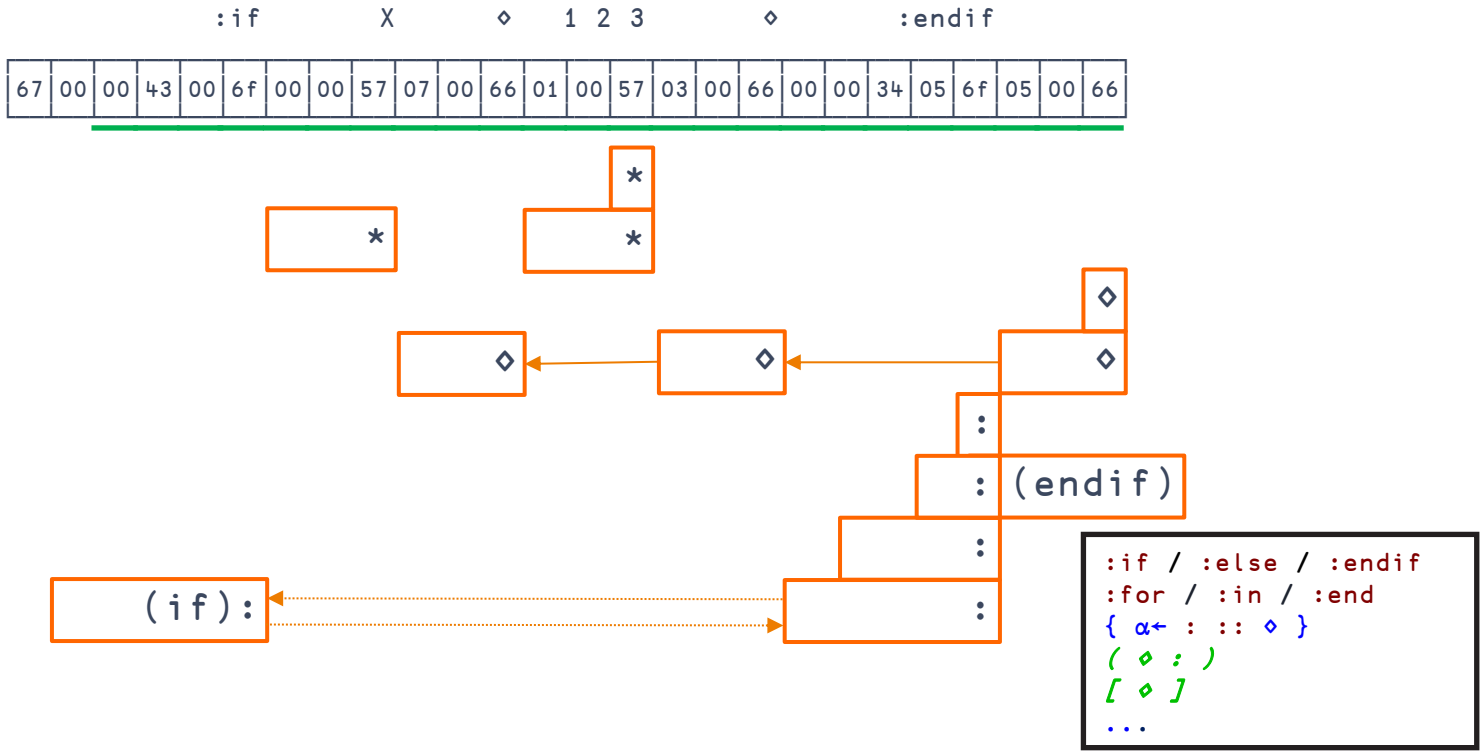
Interpreter Limits

DYALOG

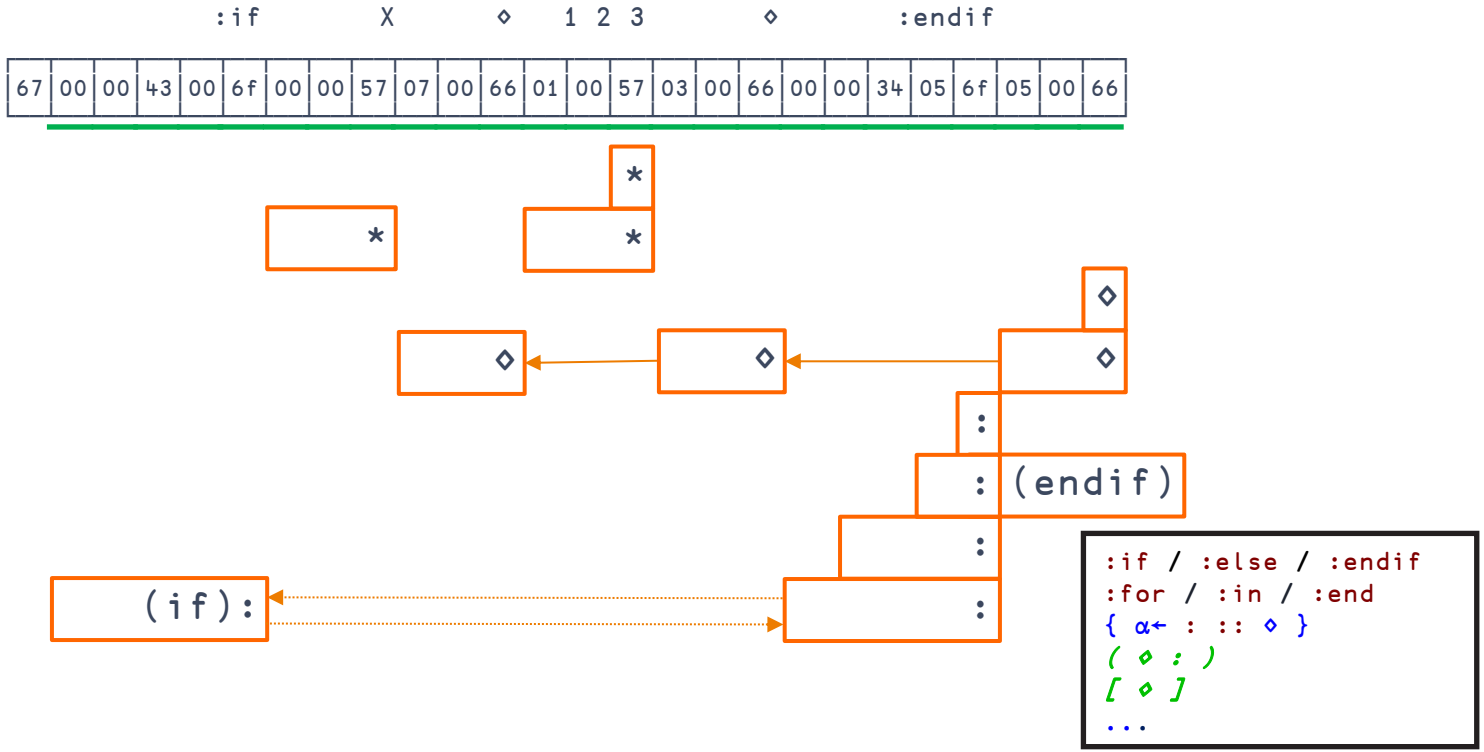


Interpreter Limits

DYALOG

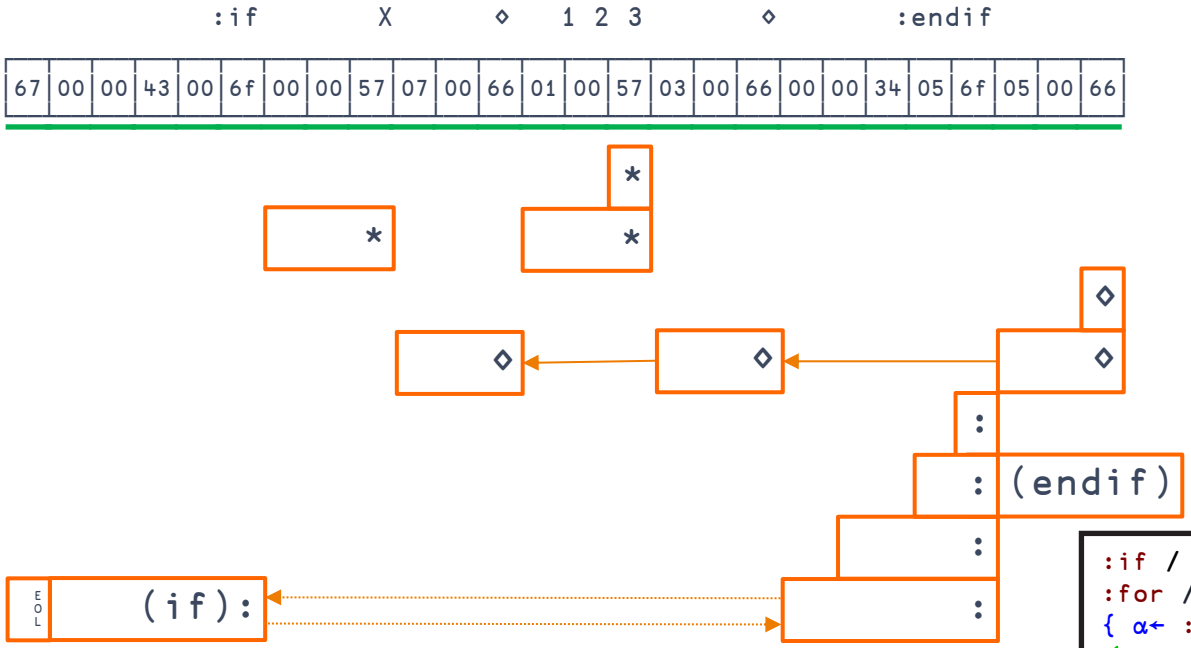


Interpreter Limits



Interpreter Limits

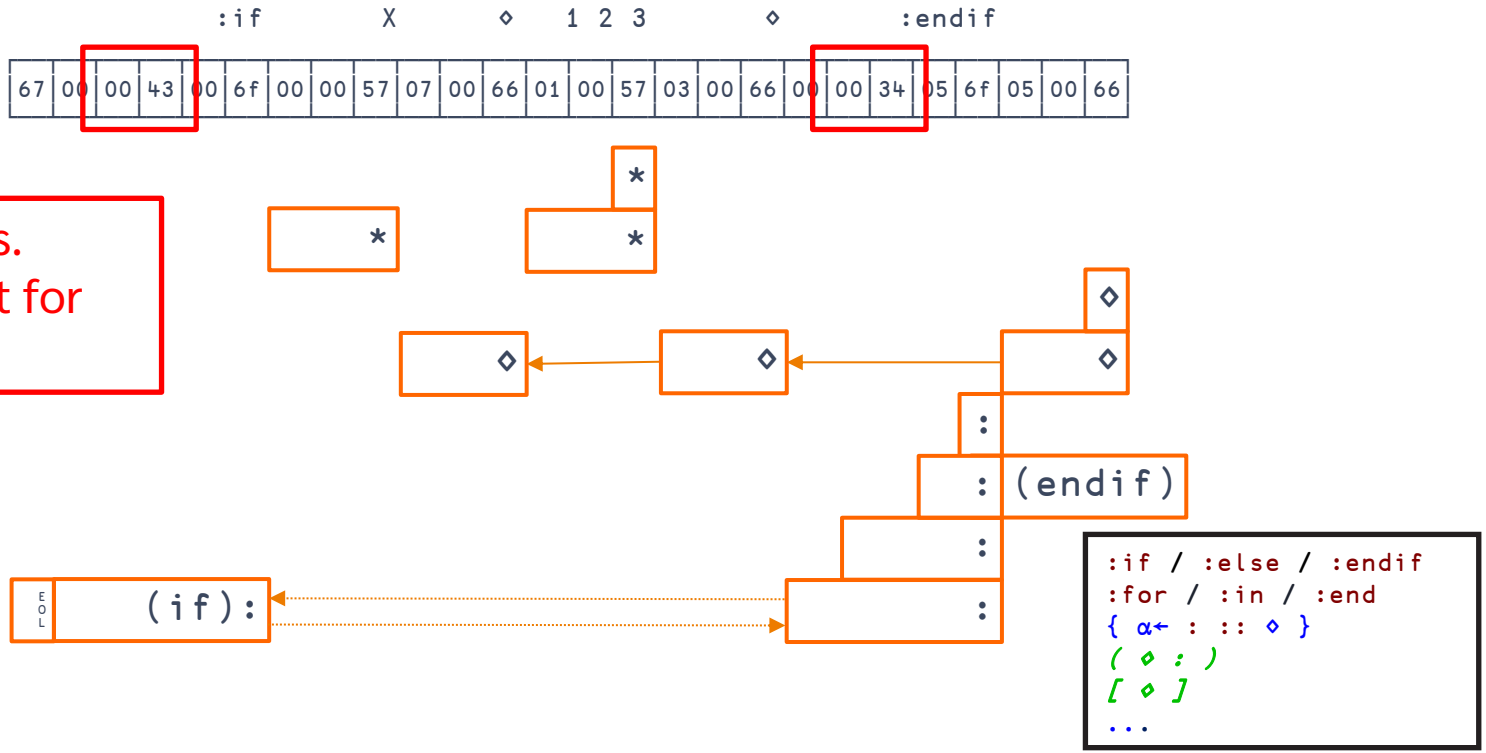
DYALOG



Interpreter Limits

DYALOG

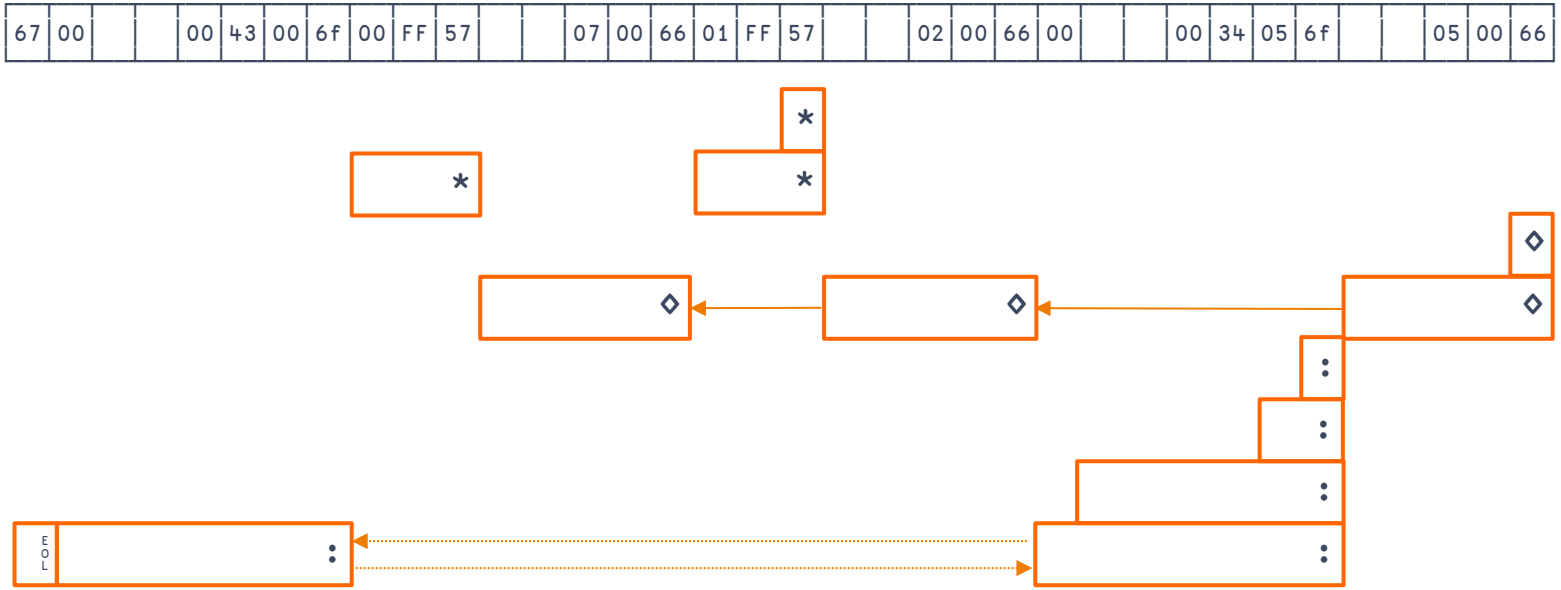
All 16 bit offsets.
65536 byte limit for functions



Interpreter Limits

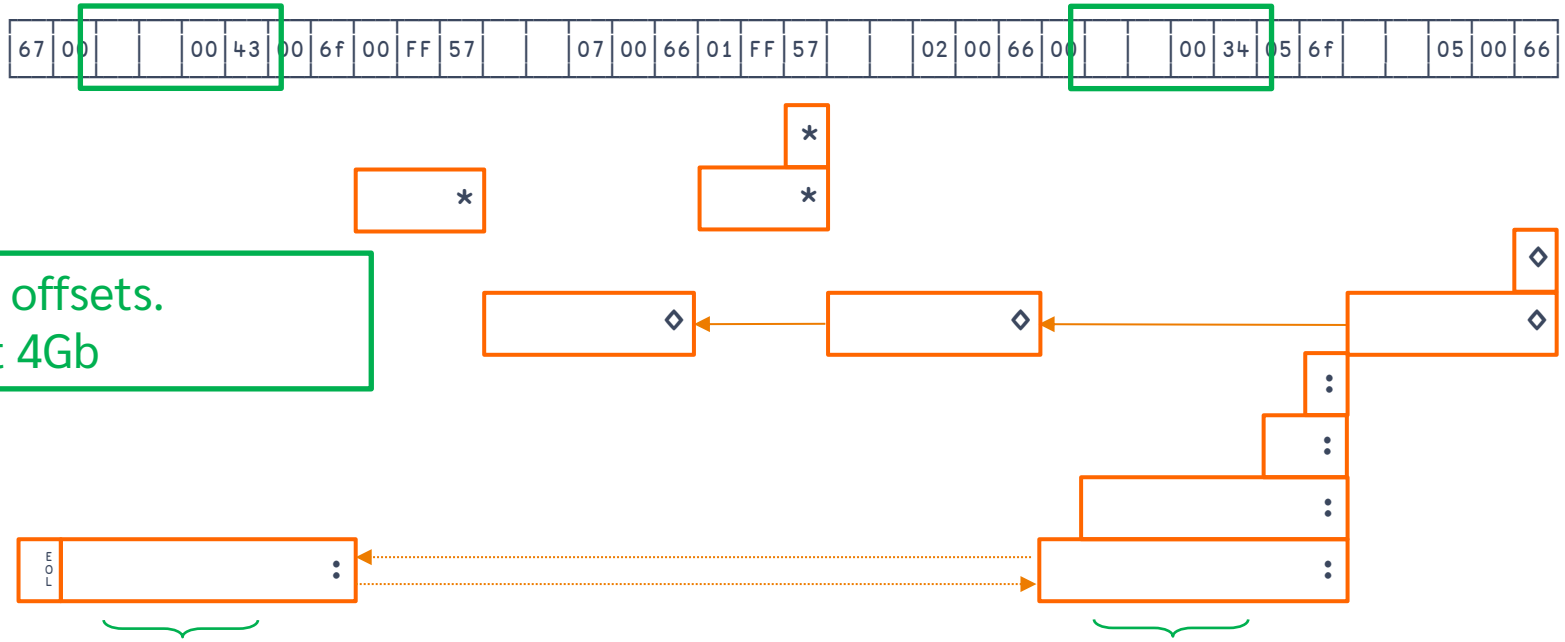
DYALOG

```
:if X ◇ 1 2 3 ◇ :endif
```



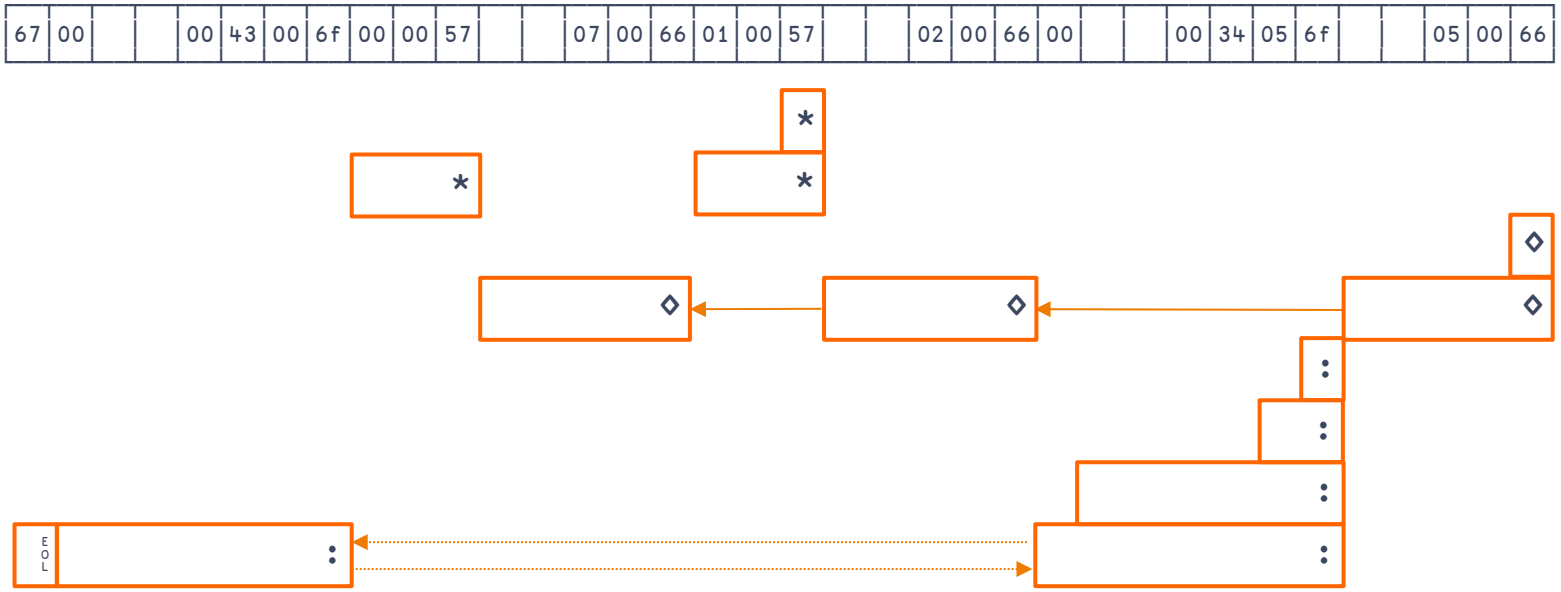
Interpreter Limits

```
:if X ◇ 1 2 3 ◇ :endif
```



32 bit offsets.
About 4Gb

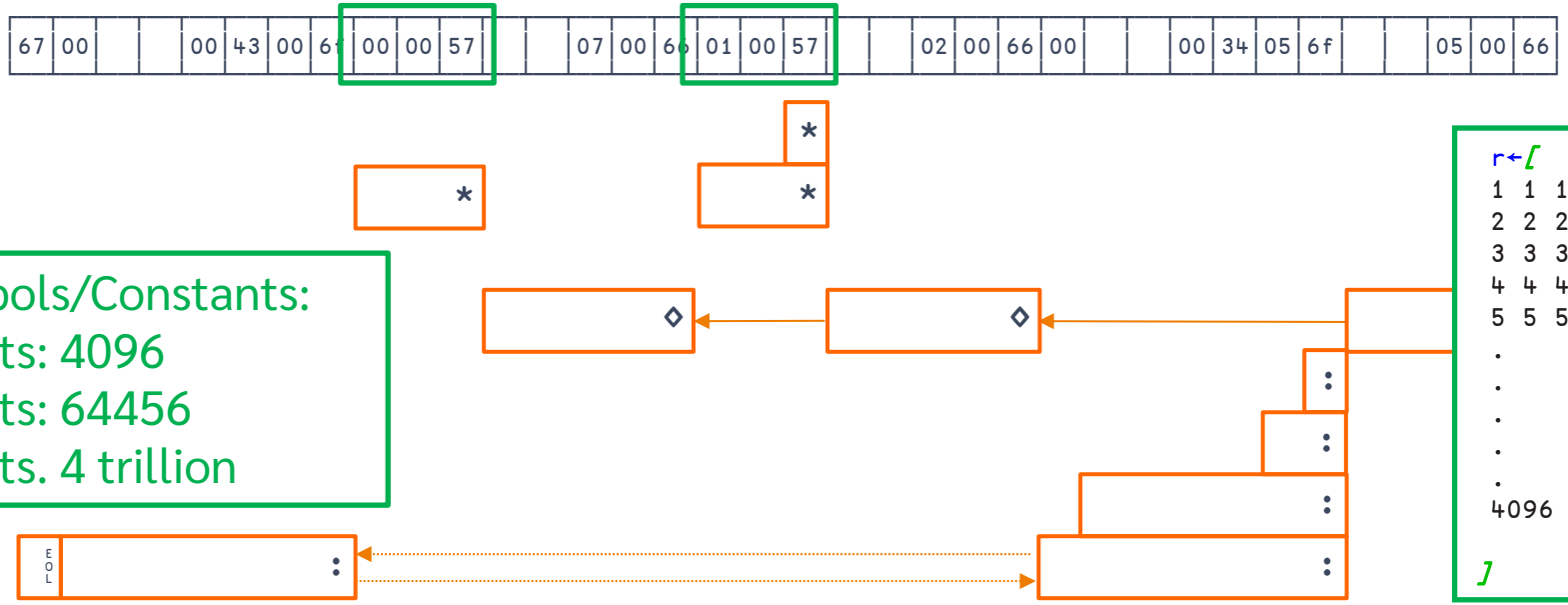
:if X ◇ 1 2 3 ◇ :endif



Interpreter Limits

DYALOG

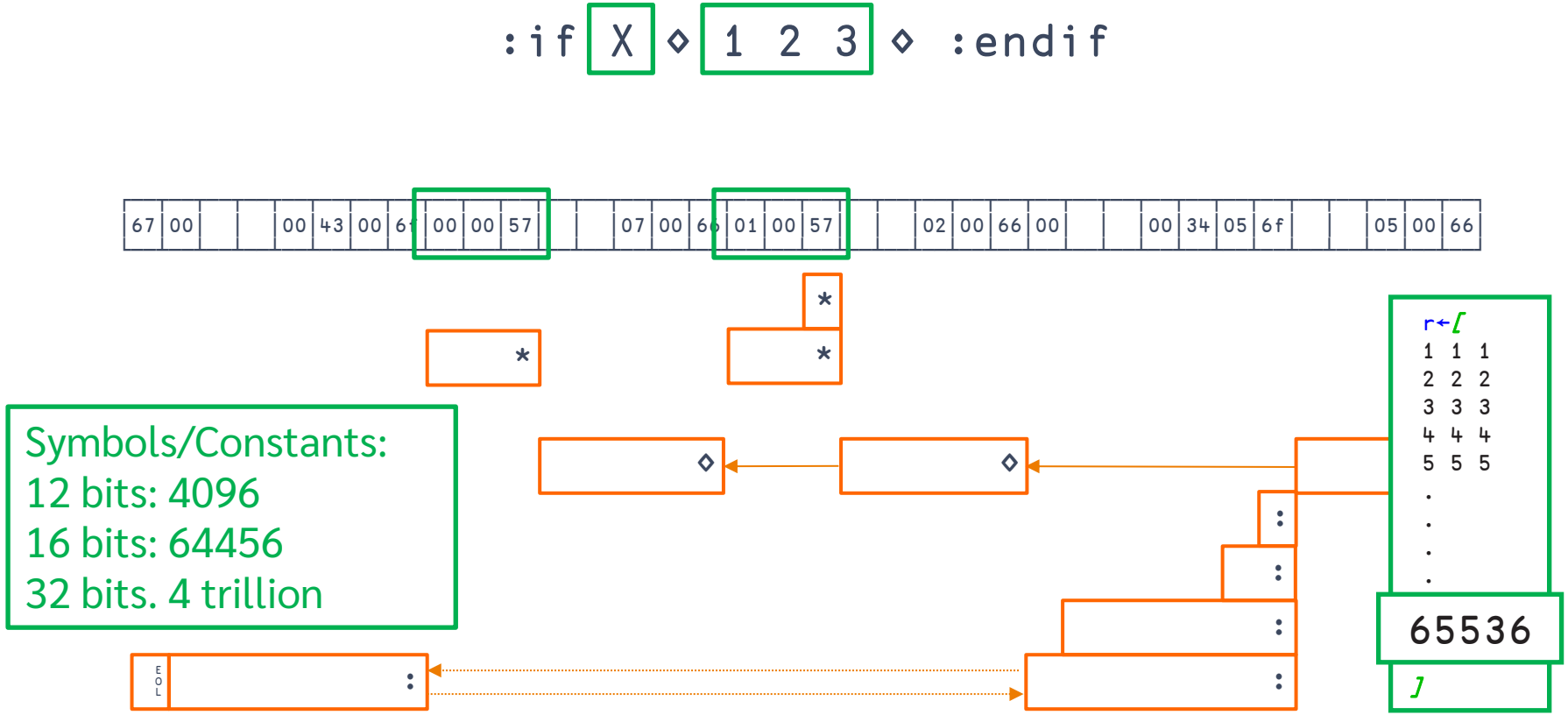
```
:if X ◇ 1 2 3 ◇ :endif
```



Symbols/Constants:
12 bits: 4096
16 bits: 64456
32 bits. 4 trillion

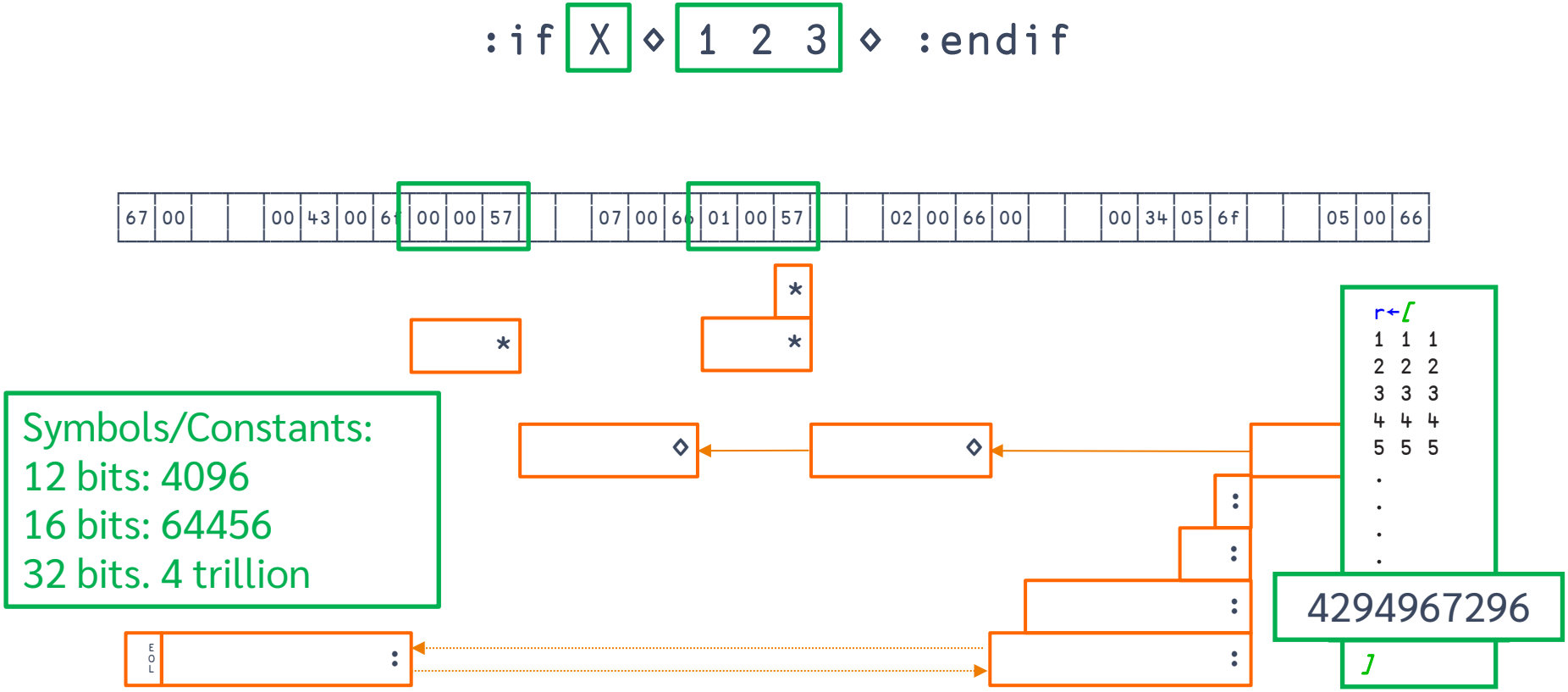
Interpreter Limits

DYALOG



Interpreter Limits

DYALOG



Interpreter Limits

It's easy then, innit?

- No
  - Old Workspaces
  - Old Component Files
  - Old serialized data (sent down a socket)

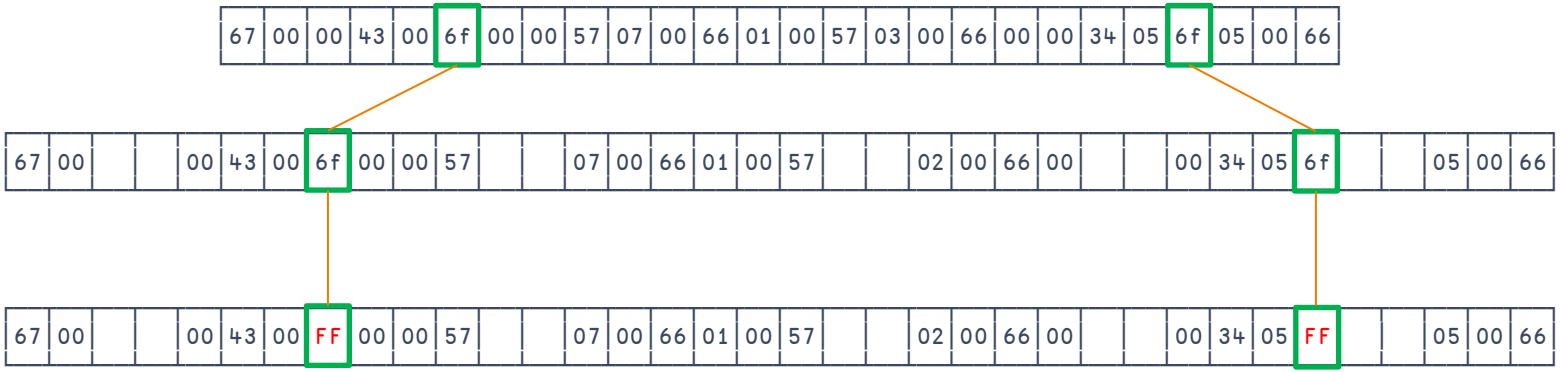
Consequences?

- Won't my workspace get larger?
- "Back of an envelope calculations"
  - dfns.dws 2% larger.

```
hex←{⎕CT ⎕IO←0 ⍝ Hexadecimal from decimal.
⍺←⊢ ⍝ no width specification.
1≠≡,⍵:⍺ ∇¨⍵ ⍝ simple-array-wise:
1∊⍵=1+⍵:'Too big'⎕SIGNAL 11 ⍝ loss of precision.
n←⍬⍴⍺,2*⌈2⍟⌈16⍟1+⌈/|⍵ ⍝ default width.
↓[0]'0123456789abcdef'[(n/16)⊤⍵] ⍝ character hex numbers.
}
```

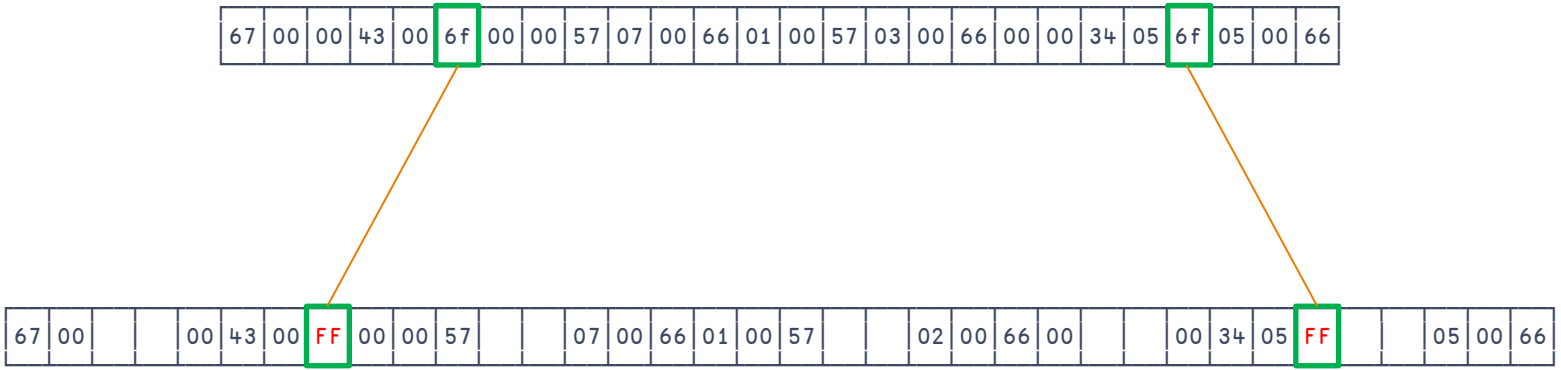
  - 3% larger

```
:if X ◇ 1 2 3 ◇ :endif
```

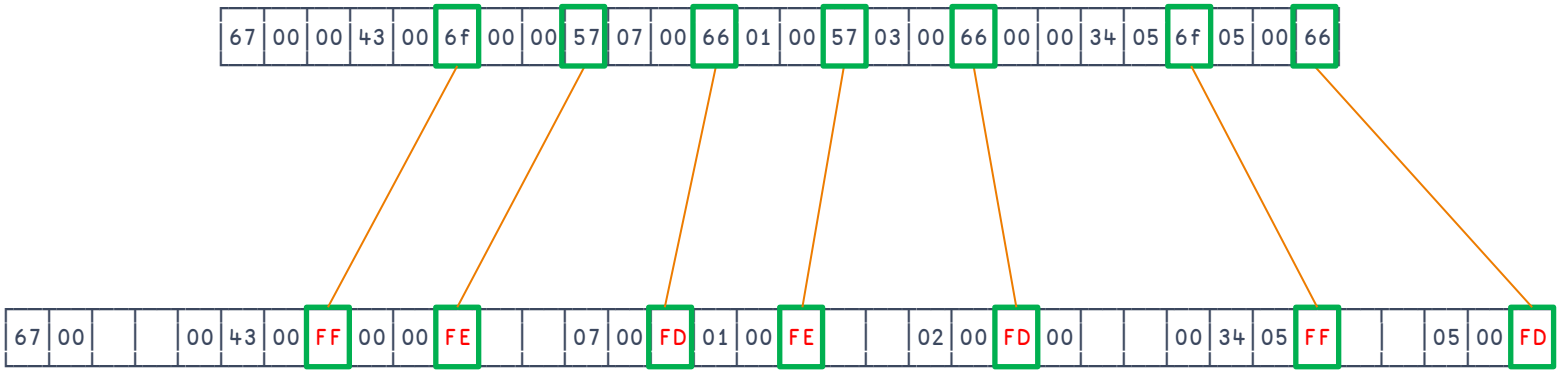


Interpreter Limits

```
:if X ◇ 1 2 3 ◇ :endif
```



```
:if X ◇ 1 2 3 ◇ :endif
```



Interpreter Limits

DYALOG

```
:if X ◇ 1 2 3 ◇ :endif
```

It's easy then, innit?

- So there's a trade off
- Workspace size vs interpreter complexity
  - Performance
- Backwards compatibility is crucial for arrays (at least for those of compatible rank, etc)

It's easy then, innit?

- It's entirely doable
- There are many limits (I've touched on a few)
- And we can do some of them individually
- But it may be more efficient to do them "en masse"
- Backwards compatibility is crucial for arrays (at least for those of compatible rank, etc)