```
- 65vmx - virtual multicore extended 6502 runtime for small embedded systems of finite count of "CORES"
- BUT this abstract language will be possibly compilable into GO source code coroutines too !!!
- xamarin.forms C# / typescipt => tools to write toolchain for 65VMX to target 6502 and GO language subset !!!
- HOW IT WAS BORN ...
  in fact, while thinking about Atari Action! language of Clinton Parker, and also PL65 language, both
  targeting 6502 cpu, I had idea long time in head to use (6502 or some similar own) ISA as core approach
  to solve multitasking by implementing many virtual 6502 cores, "single per interfaceable object" running
  in single small embedded system and potentionally connected to IoT cloud. And I thinked about simplifying
  the Action! and PL65 languages together into something more structuralized in well defined syntax, so I
  thinked about YAML as core syntax representation of new language (wishing to have more linear top-down
  program structure, inste  ad of very long lines containg large and complex expressions), by using also YAML
  hierarchical strucute for programing language strucutre used also together with advanced context-aware
  YAML-based language syntax editor (to type as few letters on keyboard as possible and more use contextual
  menus to soelect and code with interfaces, classes, keyowrds, operators, identifiers, ... etc).
  AND while such thinking I started to look for something similar, some language heavily based on YAML,
  possibly something very similar in my concept to compare and reuse/enhance ideas... So I started, ...BINGing
  and found lot of links about GO language (and remebered also to my friend working now in Vietnam in GO) and
  as I started to read more deeply about GO fundamental features, I realized that GO in fact shares my ideas
  very well, that he uses "func" keyword also as the Action! language for old 8-bit ATARI, which is in fact
  term from the japanese GO game... is this coincidental?? I doubt :-D
  => http://dave.cheney.net/page/2
  (GO = ... gofmt, interfaces, goroutines .... !!!! EXACTLY AS THINGS DESCRIBED BELOW, INDEPENDENTLY ON GO !!!


- go similarities, possibly best show as criticisms of GO language !!!
  http://yager.io/programming/go.html

- yaml
  - yaml 1.2 javascript/typescript implementation JS-YAML
  - base structure of source code !!!
  - enhaced context editor based on yam structure references !!!
  - whole language structure and user data types/classes based on it !!!
  - OSC++ "Object Syntax Composition"

- editor
  - yaml based context driven
  - no simple free text !!!

- monitor
  - ref Action! features, more embedded debugging and instrumentation, even remote

- language
  - keywords: [comment#, directive, strucutal, control, expressions, operators, basetypes ]
  - basetypes: [byte, card, int, char, ", ',  $, %, 0x, ]
  - collections: [array, list, hash, stack, queue, pointer, @, [, ], ]
  - INTERFACES / user types (classes)
  - operators: [and, lsh, mod, or, rsh, xor, +, -, *, /, &, %, !, ==, !=, <>, >, >=, <, <=, ~, ^, ]
  - expressions: [=, (, ), ]
  - control: [do, else, elseif, exit, fi, for, if, od, step, then, to, until, while ]
  - structural: [type, proc, func, return, ]
  - directive: [define, include, set, module ]
  - comment: #

- identifiers
  -

- addressing
  -

- defines/macros (yaml features)

- types
  - basetypes
  - collections
  - INTERFACES / user types (CLASSES => core structural language feature, based on yaml features)

- constants (readonly vars?)
  - type
  - identifier
  - value

- variables
  - type
  - identifier
  - base address / location (zp-reg, data-stack, )
  - initial value
  - ??? immutability ??? explicit specifying location as mutable so not const/readonly but variable ??????????

- operators
  - ??? internally implemented as function calls; so only aliases to function calls/inlines
  - ??? redefinable ???

- expressions
  - constants, variables, operators, functions, subexpressions
  - operator names identical to modern C++/C#/java/js/ts syntax, preferably C#/ts !!!
  - SIMPLE APPROACH with general purpose STACKs (STACK + QUEUE as special ARRAYS !!!)
  - NO operators precedence, as expressions coded EXPLICITLY (topdown with stack or as yaml-tree)
  - DATA STACK(s) as core explicit feature (there may be more of them used to scratchpad and fetch things)
  - assignment
     - target_variable =UNARY_OP
     - target_variable = constant/readonly
     - target_variable = function/method call
     - target_variable = source_variable (where source may be STACK position (TOP based), as resul of expression)
  - comparison
     - target_variable <RELATION_OP> constant
     - target_variable <RELATION_OP> source_variable
     - so, short-circuit comparisons enocoded explicitly (topdown or as yaml-tree)
     - so, not possible to compare directly to function/method call (explicit assignment required somewhere)

- control
  - if then else elseif
  - switch case
  - for to step
  - while do
  - repeat until

- structural
  - type
  - proc
  - func
  - return
```