

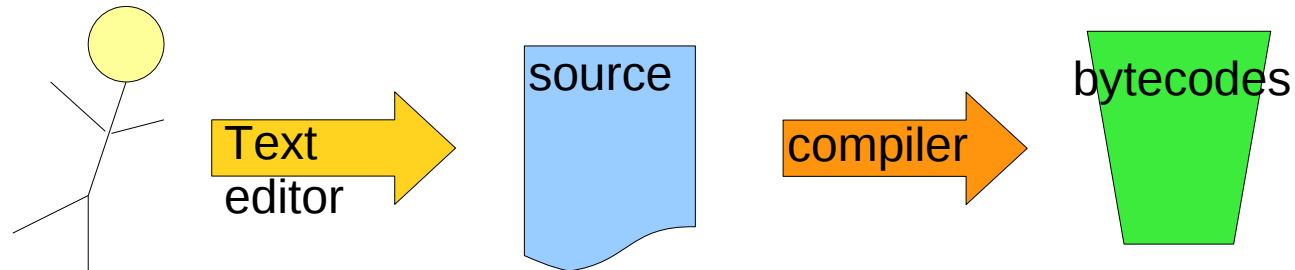
Adaptive Compilation for a Reconfigurable, Object- Oriented Architecture

Jecel Mattos de Assumpção Jr
Advisor: Eduardo Marques

WLCR2011 – II Workshop do Laboratório de
Computação Reconfigurável (LCR) do
ICMC-USP
11 de março de 2011

A method's life cycle

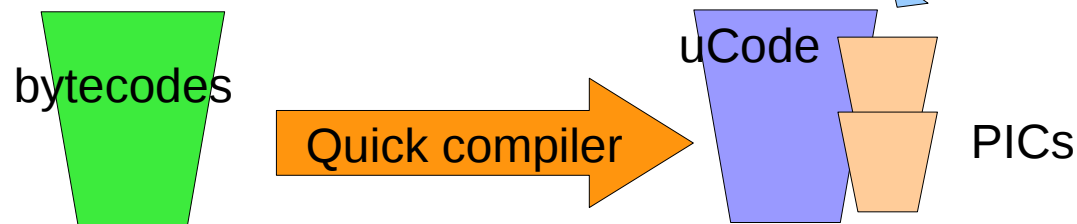
@ any time:



@ first invocation:

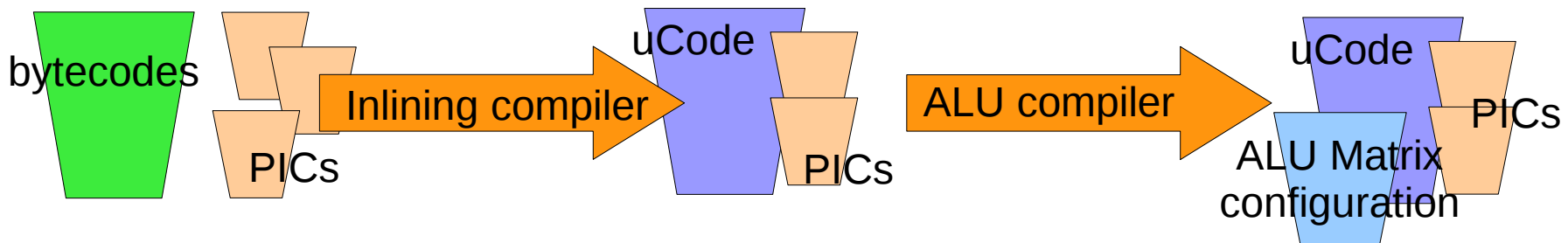


@ next invocation:



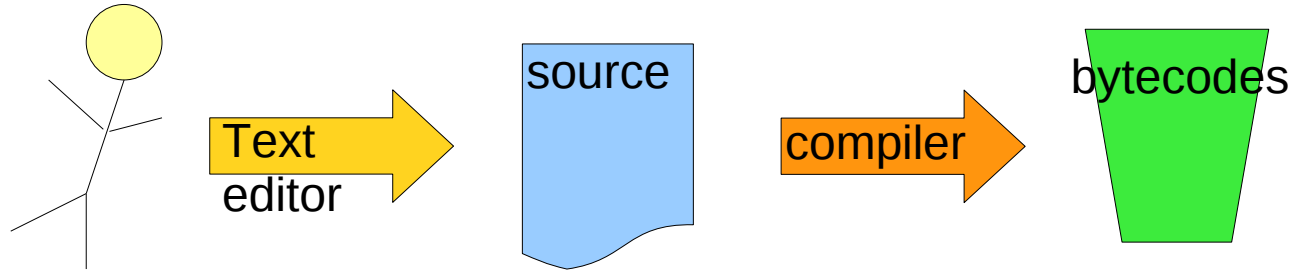
@ invocation counter overflow:

@ PC sample threshold:

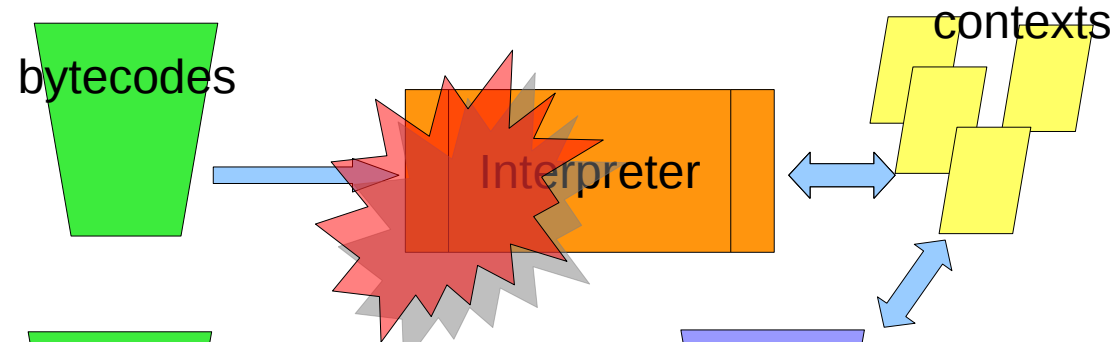


Reconfiguration (1 and 2)

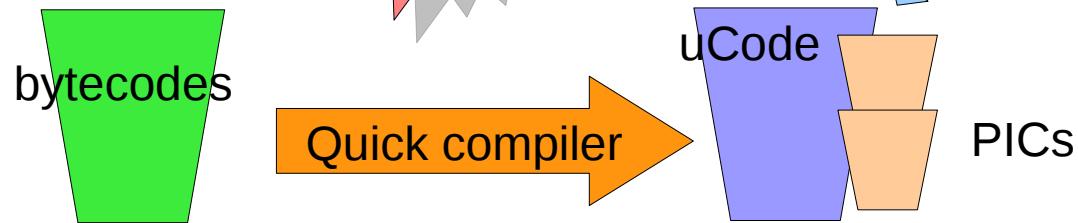
@ any time:



@ first invocation:

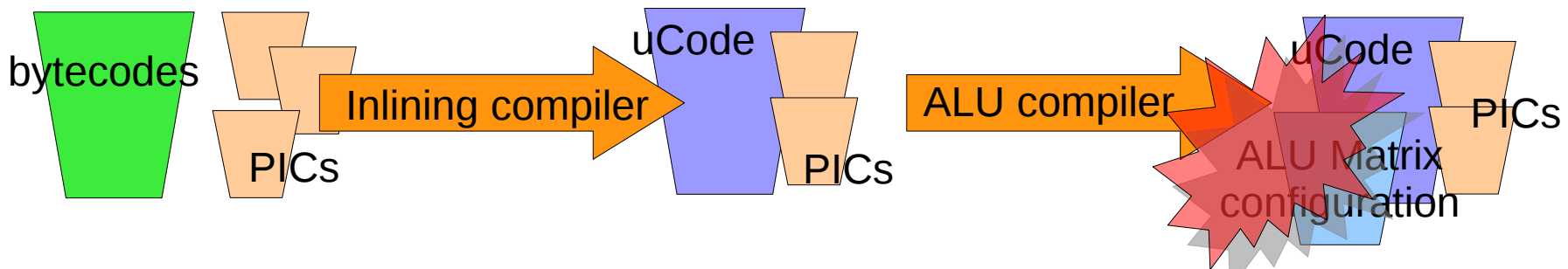


@ next invocation:



@ invocation counter overflow:

@ PC sample threshold:

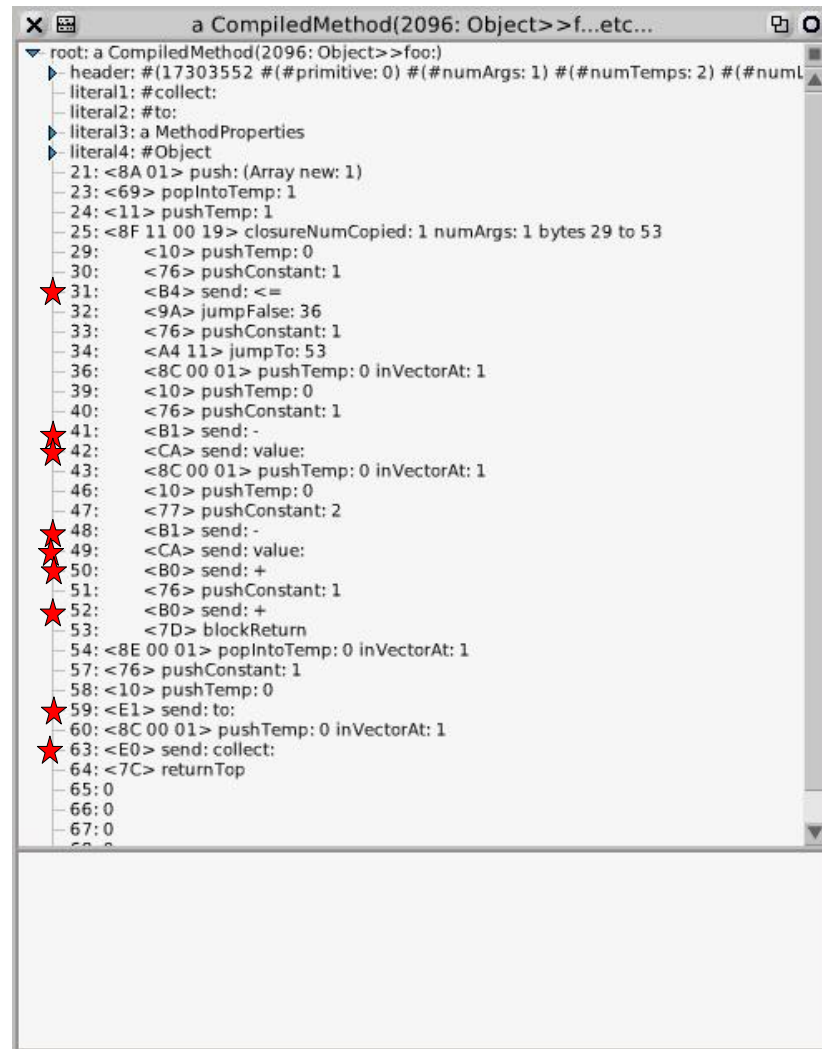


Bytecodes

```
foo: n
| nfib |
nfib := [:i | i <= 1 ifTrue: [1]
        IfFalse: [(nfib value: i - 1)
                  + (nfib value: i - 2)
                  + 1]].

^(1 to: n) collect: nfib
```

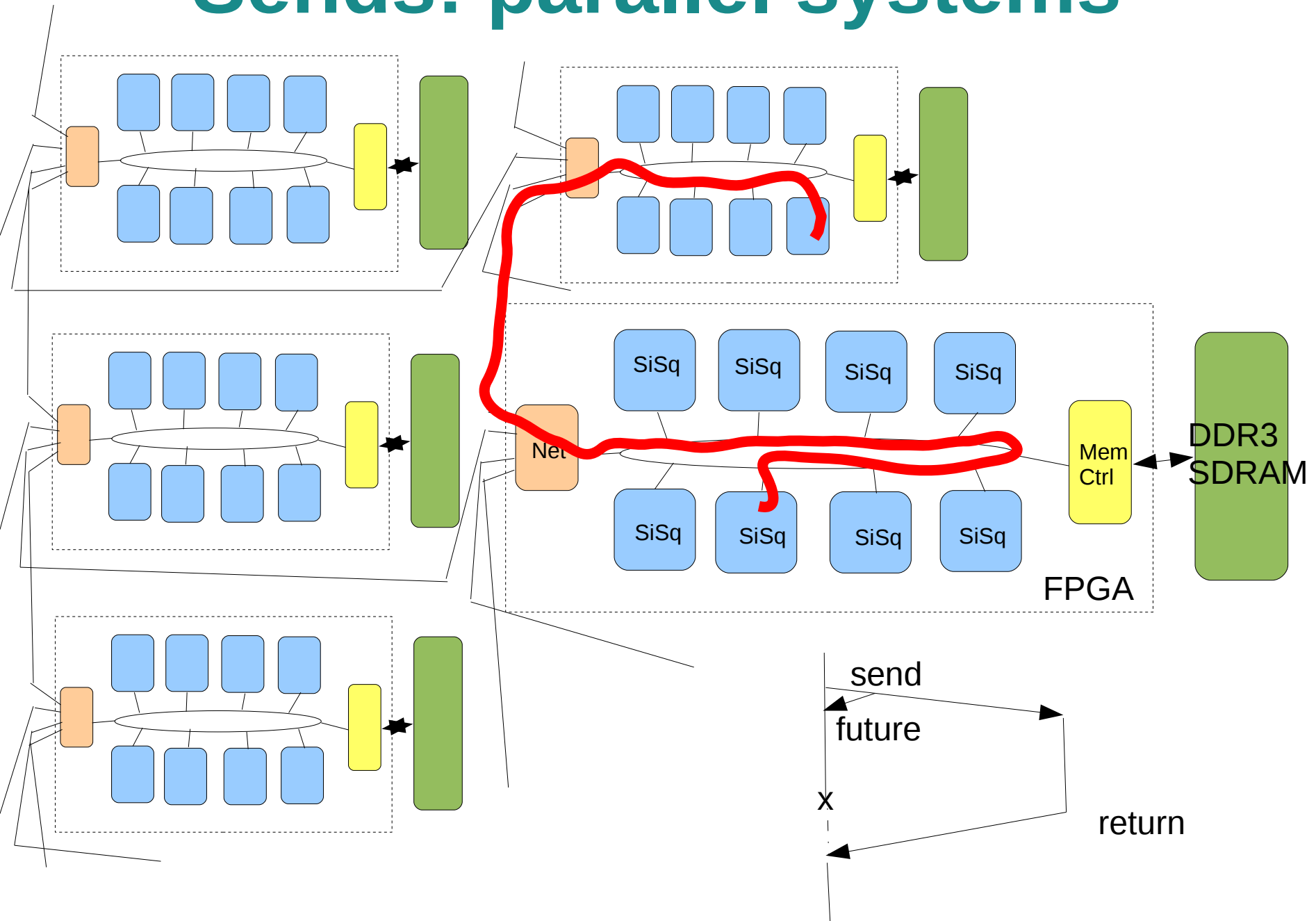
★ Indicates the send bytecodes



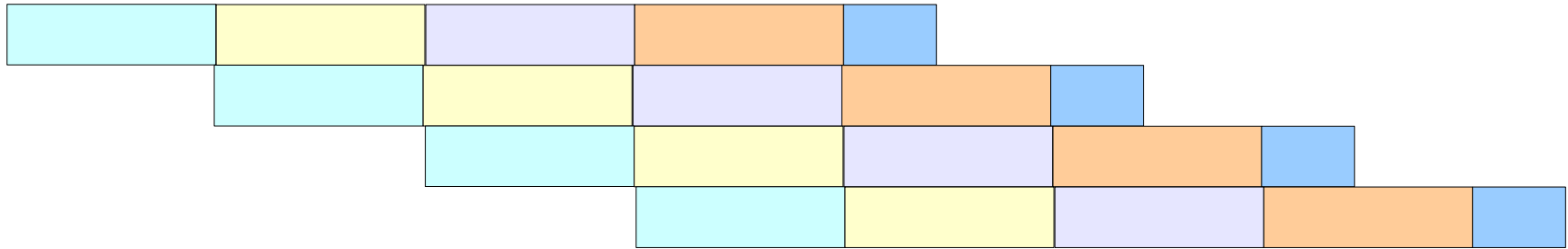
The screenshot shows a window titled "a CompiledMethod(2096: Object)>>f...etc...". The content is a list of bytecodes for a method. The list starts with a root node and a header, followed by literals and a series of numbered bytecodes. Red stars are placed to the left of several bytecodes, indicating 'send' operations. The bytecodes include instructions like 'push', 'popIntoTemp', 'pushTemp', 'closureNumCopied', 'pushConstant', 'send', 'jumpFalse', 'jumpTo', 'pushTemp: 0 inVectorAt: 1', 'blockReturn', and 'returnTop'. The list ends with several zero values.

```
root: a CompiledMethod(2096: Object)>>foo:
- header: #(17303552 #(#primitive: 0) #(#numArgs: 1) #(#numTemps: 2) #(#numL
- literal1: #collect:
- literal2: #to:
- literal3: a MethodProperties
- literal4: #Object
- 21: <8A 01> push: (Array new: 1)
- 23: <69> popIntoTemp: 1
- 24: <11> pushTemp: 1
- 25: <8F 11 00 19> closureNumCopied: 1 numArgs: 1 bytes 29 to 53
- 29: <10> pushTemp: 0
- 30: <76> pushConstant: 1
★ 31: <B4> send: <=
- 32: <9A> jumpFalse: 36
- 33: <76> pushConstant: 1
- 34: <A4 11> jumpTo: 53
- 36: <8C 00 01> pushTemp: 0 inVectorAt: 1
- 39: <10> pushTemp: 0
- 40: <76> pushConstant: 1
★ 41: <B1> send: -
★ 42: <CA> send: value:
- 43: <8C 00 01> pushTemp: 0 inVectorAt: 1
- 46: <10> pushTemp: 0
- 47: <77> pushConstant: 2
★ 48: <B1> send: -
★ 49: <CA> send: value:
★ 50: <B0> send: +
★ 51: <76> pushConstant: 1
★ 52: <B0> send: +
- 53: <7D> blockReturn
- 54: <8E 00 01> popIntoTemp: 0 inVectorAt: 1
- 57: <76> pushConstant: 1
- 58: <10> pushTemp: 0
★ 59: <E1> send: to:
- 60: <8C 00 01> pushTemp: 0 inVectorAt: 1
★ 63: <E0> send: collect:
- 64: <7C> returnTop
- 65: 0
- 66: 0
- 67: 0
- 68: 0
```

Sends: parallel systems



Interpreter



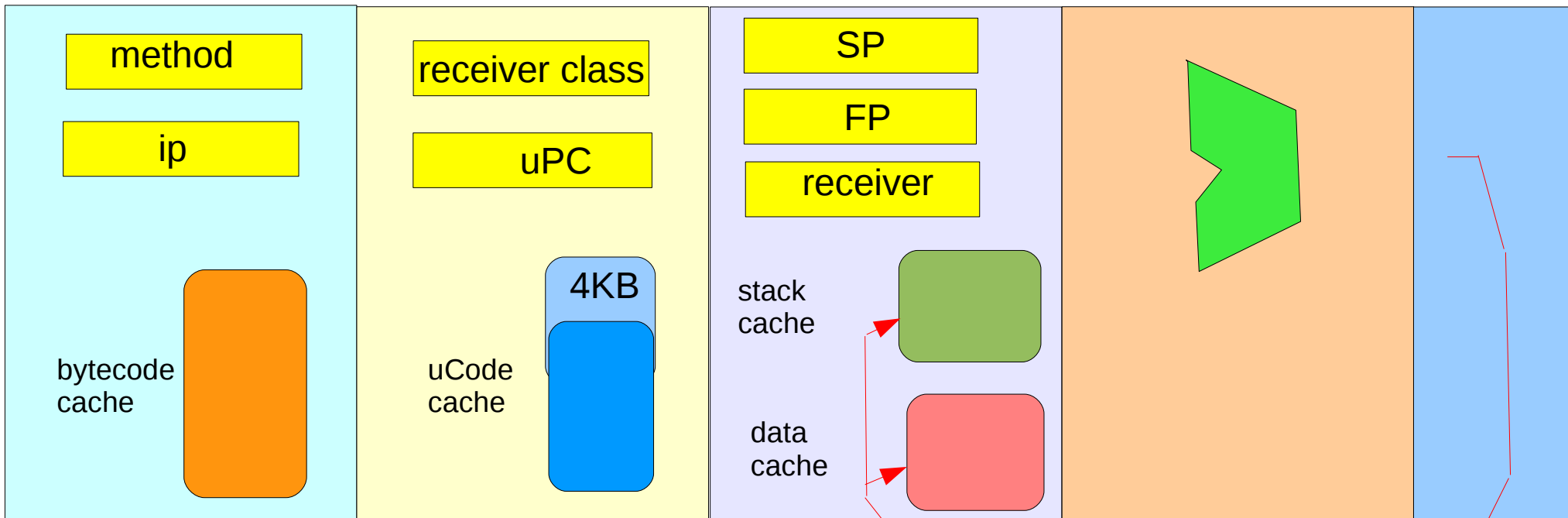
bytecode fetch

uCode fetch

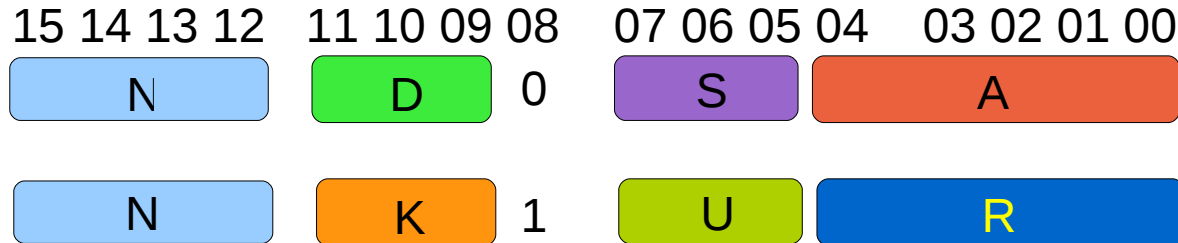
address / op read

execute

op write



Microcode



N:
 0000 next
 0001 return
 0010 skipOnZero
 0011 skipOnOne
 0100 PICuPC
 0101 PIC2
 0110 fetch8
 0111 tagAndFetch8
 1000 fetch1
 1001 fetch2
 1010 fetch3
 1011 fetch4
 1100 jump
 1101 PICx
 1110 call
 1111 tagOrCall

K:
 000 POP
 001 STR
 010 PSH
 011 INR
 100 ADR
 101 SBR
 110 ADK
 111 SBK

U:
 000 temporary
 001 instance
 010 stream
 011 special
 100 literal
 101 immediate
 11x ALU Matrix

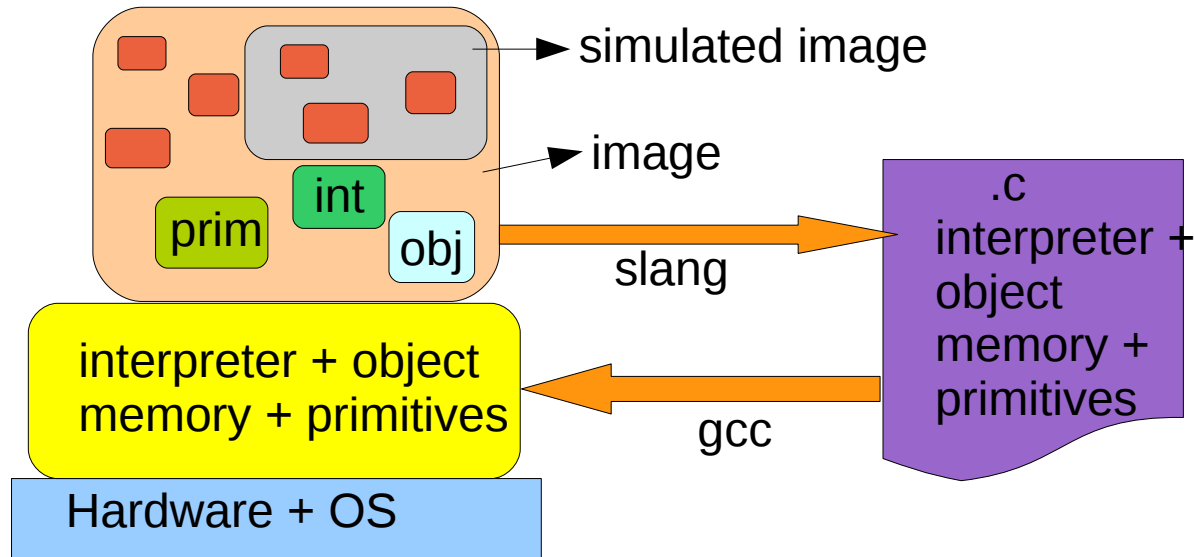
A:
 00xxx - MOV, INC, NOT, NEG,
 ADD, ? , ? , SUB
 01xxx - IOR, NOR, XOR, EQV,
 AND, ANI, NND, NAI
 10xxx - SHR, SHL, ASR, ROT,
 MUL, DIV, UML, UDV
 11xxx - EQL, CRY, MIN, OVF,
 LTN, LEQ, ULT, ULE

D or S = 111 means push
 or pop stack, while 000 to
 110 indicate SP-0 to SP-6

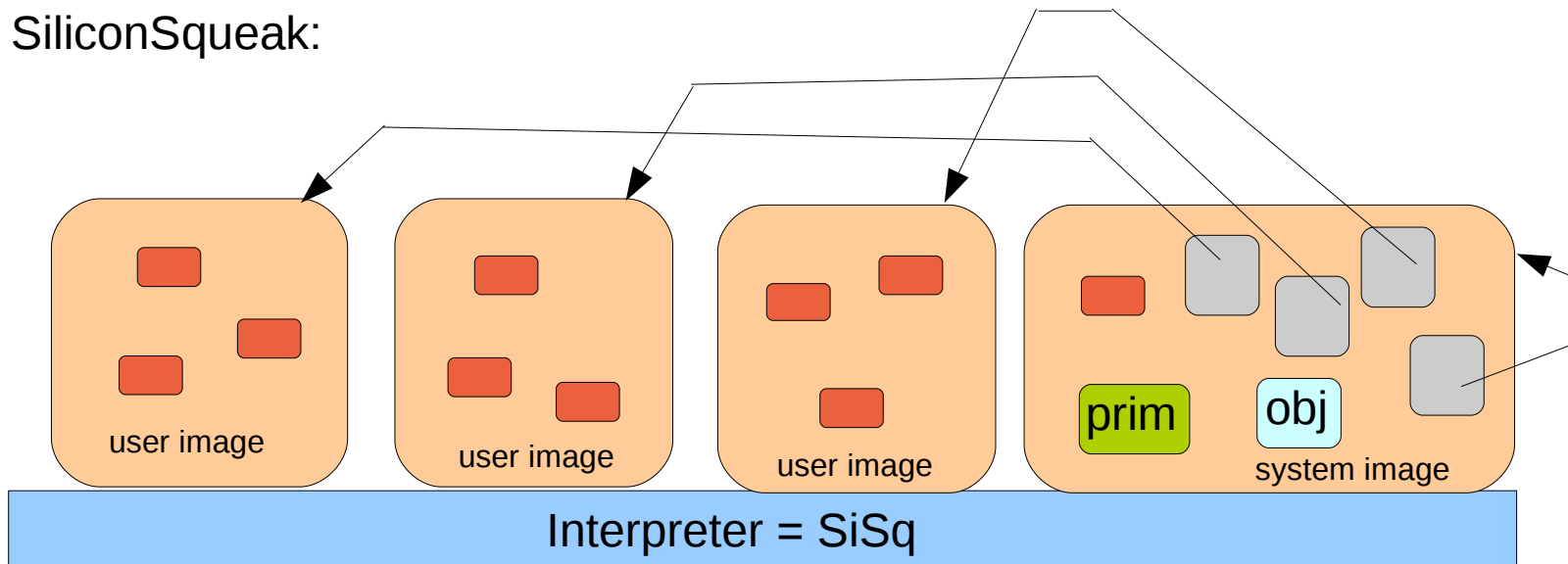
bit 15 = 1 means the 16 bits after the
 instruction are used to modify uPC

Memory and Primitives

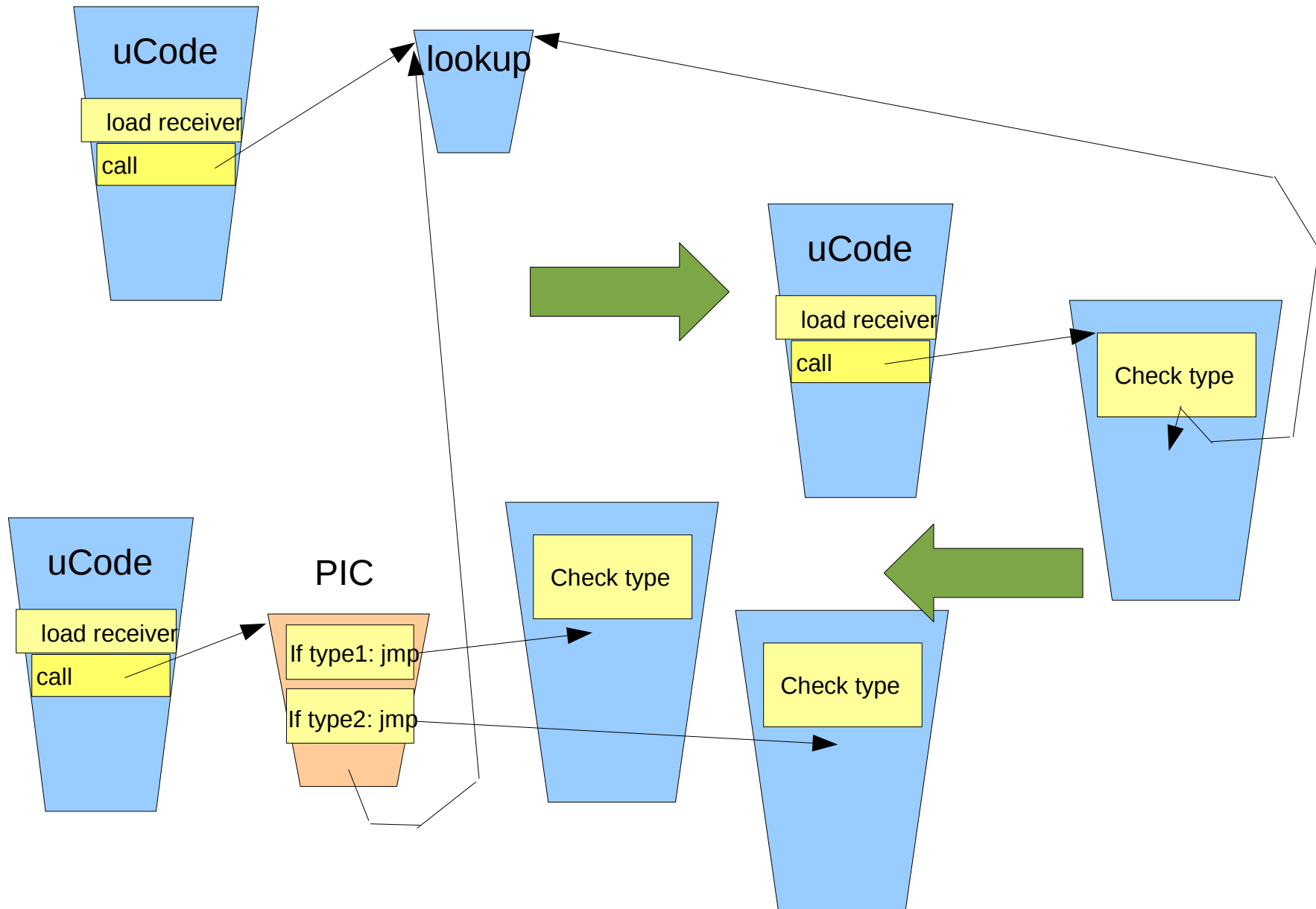
“Back to the Future”:



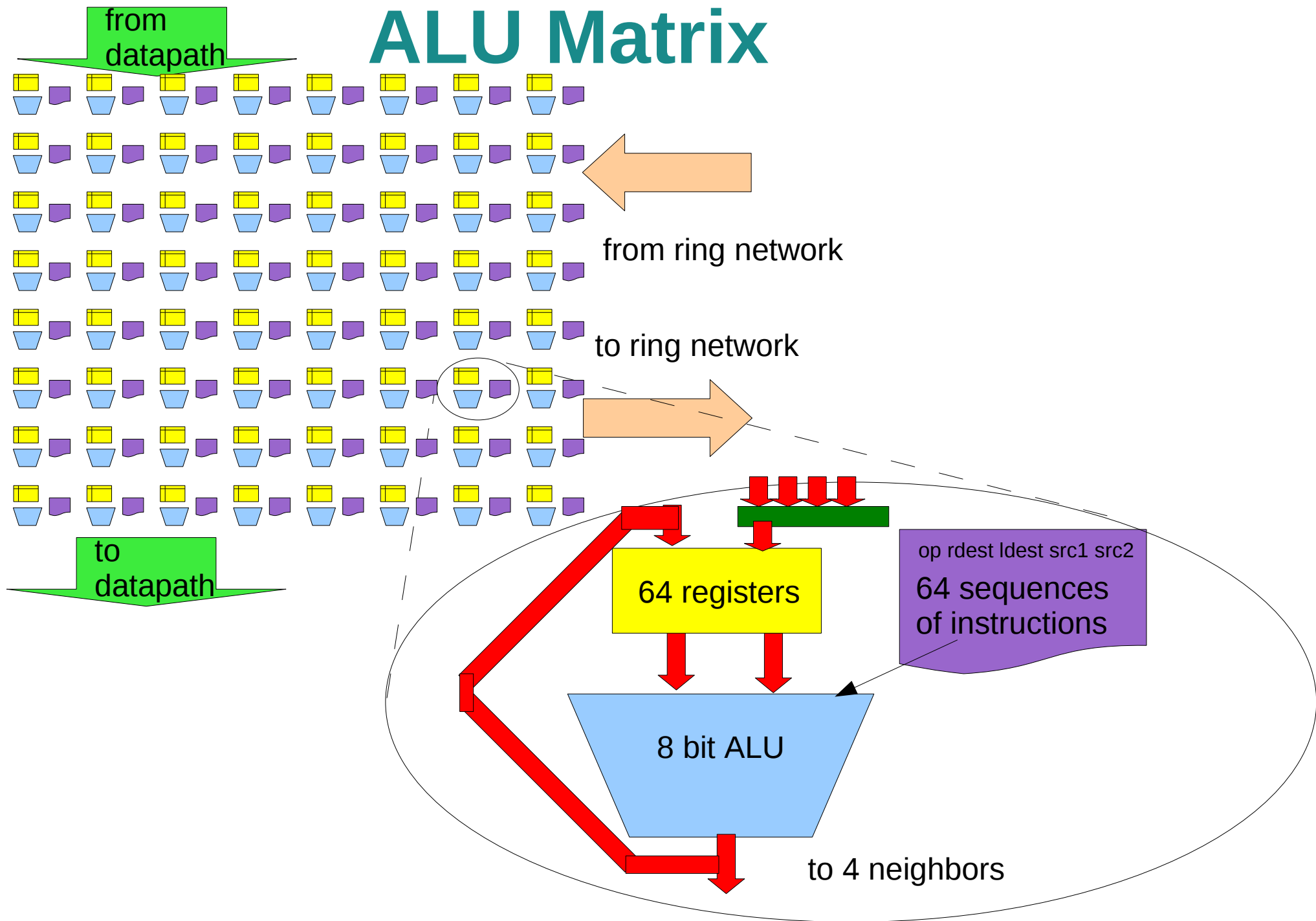
SiliconSqueak:



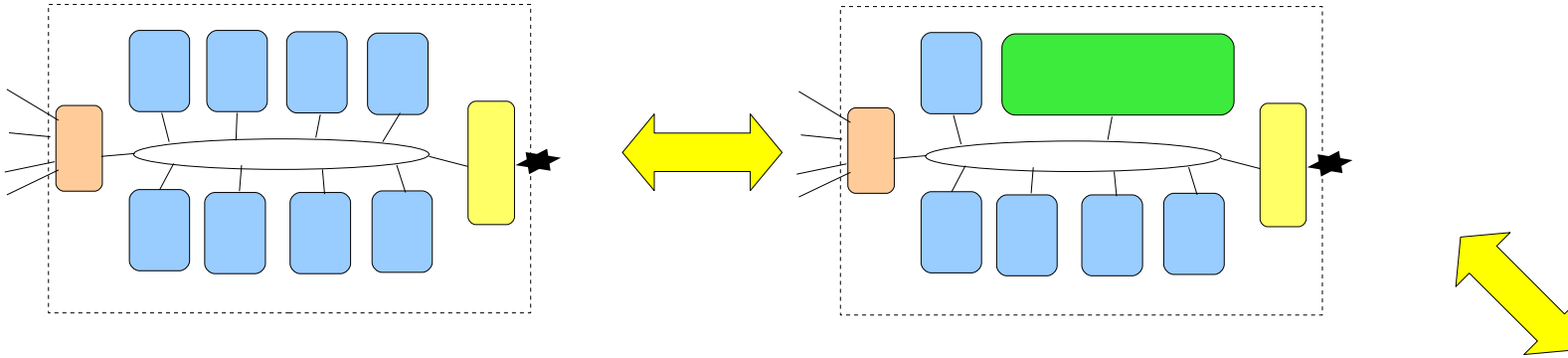
Polymorphic Inline Cache (PIC)



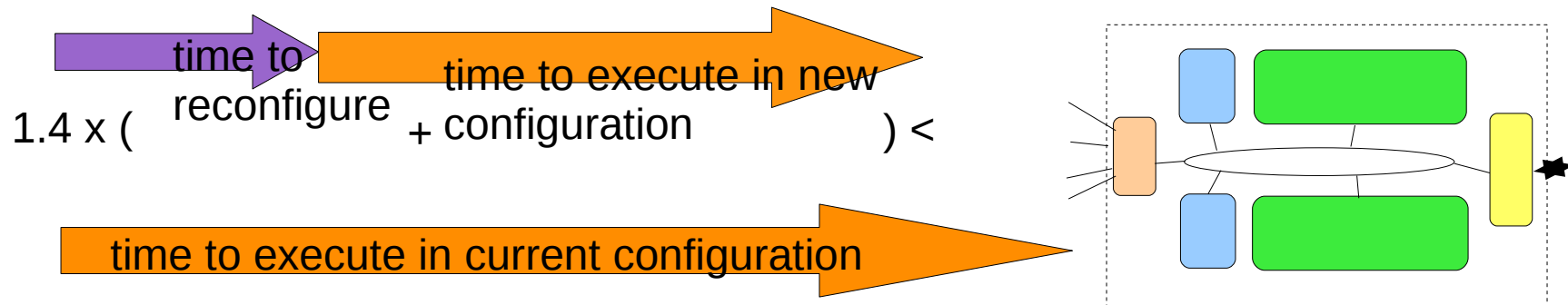
ALU Matrix



Reconfiguration (3)



switch when:



Thanks!

- Parallelism
- Interpreter
- PICs
- ALU Matrix
- 3 types of Reconfiguration