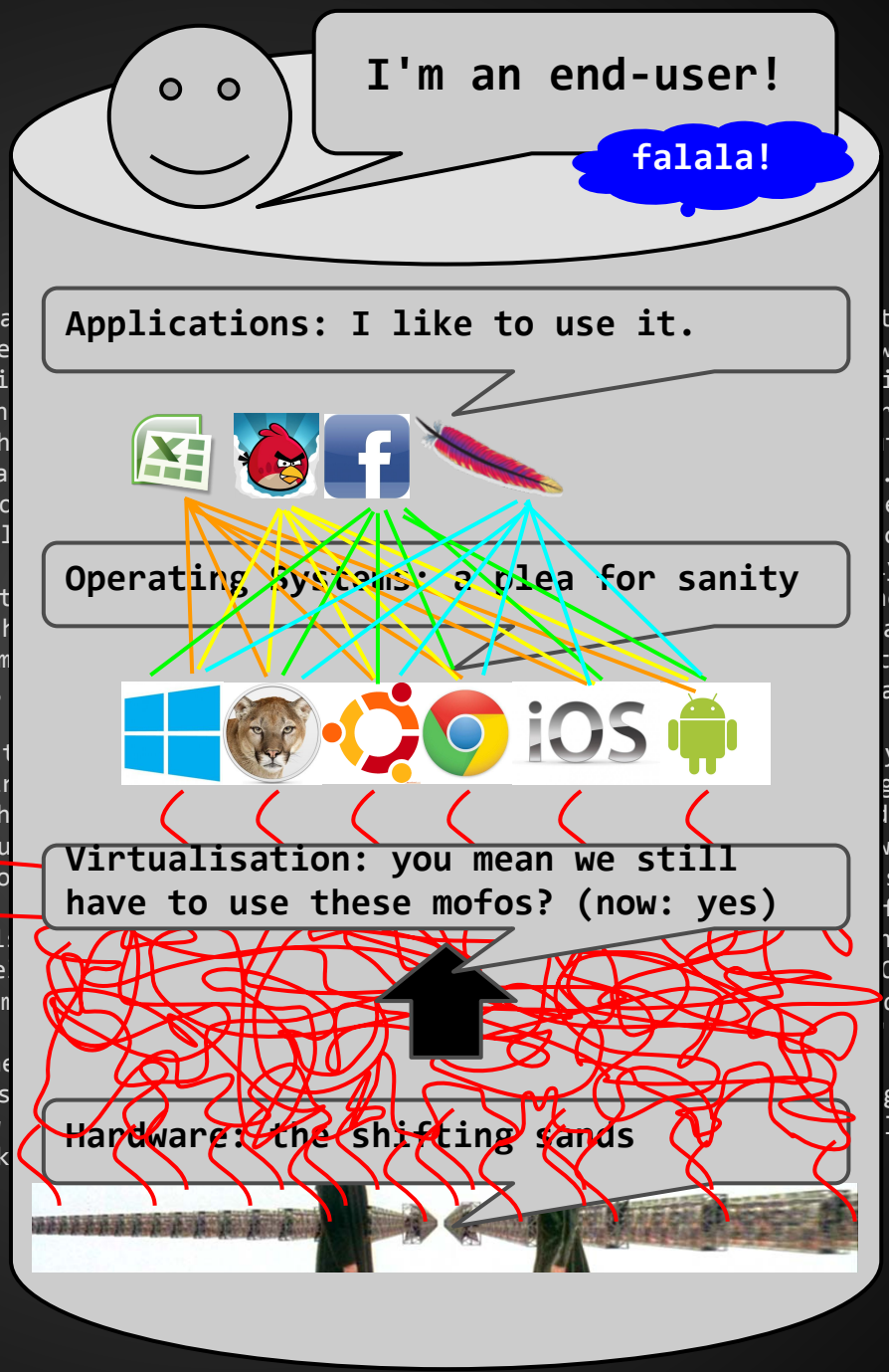
"stack"

what is a stack in IT? We're talking about a
Material science and E&E engineering are the
firm foundation... it changes very often, li
Windows, Linux, and Android to deal with th
platform", upon which to base applications th
becoming very trendy - and you must be awa
although some significant work has been do
computers together, there are no dominant sol
as widespread as any of the aforementioned
called hypervisors, that fit between operat
computers that need to be made to work togeth
forces all application development to implem
done among many computers. So let's say now,
language you like, and you need to write an
for this crowd is an application that needs t
is the hardware or software machine that al
done?" Once you had an answer, you might th
learn that language, so that you could commu
done. It turns out that Ericsson has continuo
very clever at solving this problem problem,
job that helps some end-user. Ericsson call
working with the BEAM emulator, "the Open Te
that BEAM uses to think about distributed com
language called Erlang. This is why it is
languages don't do work - people, or machine
they are little people.) The rest of today's
computing done, with the help of OTP, 2) how
comprehensible to BEAM (technically: I'm talk



re-stack?

what is a stack in IT? We're talking about a
Material science and E&E engineering are the
firm foundation... it changes very often, li
Windows, Linux, and Android to deal with th
platform", upon which to base applications th
becoming very trendy - and you must be awa
although some significant work has been dc
computers together, there are no dominant sol
as widespread as any of the aforementioned
called hypervisors, that fit between operat
computers that need to be made to work togeth
forces all application development to implem
done among many computers. So let's say now,
language you like, and you need to write an
for this crowd is an application that needs t
is the hardware or software machine that alr
done?" Once you had an answer, you might th
learn that language, so that you could commu
done. It turns out that Ericsson has continuo
very clever at solving this problem problem,
job that helps some end-user. Ericsson call
working with the BEAM emulator, "the Open Te
that BEAM uses to think about distributed com
language called Erlang. This is why it is
languages don't do work - people, or machine
they are little people.) The rest of today's
computing done, with the help of OTP, 2) how
comprehensible to BEAM (technically: I'm talk

ting at the top.
ware is hardly a
ing systems like
", or "floating
ted computing is
. Unfortunately,
e many separate
ceptance that is
y a thin layer,
e many separate
and so it almost
that needs to be
arn to speak any
an easy example
yourself, "which
getting this job
l then you would
v to get the job
software that is
forming a single
he framework for
OTP is a pattern
d normally use a
s remember that
machines, as if
gets distributed
language is made

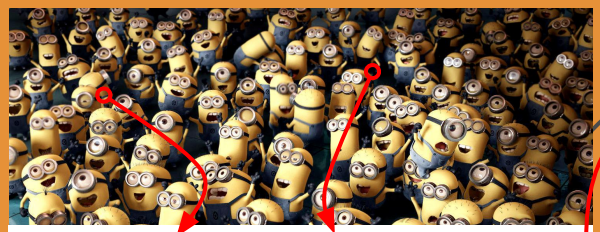
Erlang/OTP?

what is a stack in IT? We're talking about a
Material science and E&E engineering are the
firm foundation... it changes very often, li
Windows, Linux, and Android to deal with th
platform", upon which to base applications th
becoming very trendy - and you must be awa
although some significant work has been dc
computers together, there are no dominant sol
as widespread as any of the aforementioned
called hypervisors, that fit between operat
computers that need to be made to work togeth
forces all application development to implem
done among many computers. So let's say now,
language you like, and you need to write an
for this crowd is an application that needs t
is the hardware or software machine that alr
done?" Once you had an answer, you might th
learn that language, so that you could commu
done. It turns out that Ericsson has continuo
very clever at solving this problem problem,
job that helps some end-user. Ericsson call
working with the BEAM emulator, "the Open Te
that BEAM uses to think about distributed com
language called Erlang. This is why it is
languages don't do work - people, or machine
they are little people.) The rest of today's
computing done, with the help of OTP, 2) how
comprehensible to BEAM (technically: I'm talk

ting at the top.
ware is hardly a
ing systems like
", or "floating
ted computing is
. Unfortunately,
e many separate
ceptance that is
y a thin layer,
e many separate
and so it almost
that needs to be
arn to speak any
an easy example
yourself, "which
getting this job
l then you would
v to get the job
software that is
forming a single
he framework for
OTP is a pattern
d normally use a
s remember that
machines, as if
gets distributed
language is made

1. An Erlang **"Node a.k.a. Emulator a.k.a. VM"** runs as an **OS process**.

2. Internally it can manage '00s, '000s, or '000,000s of **Erlang processes**.



3. Erlang processes are spawned just by passing functions into **spawn()**.

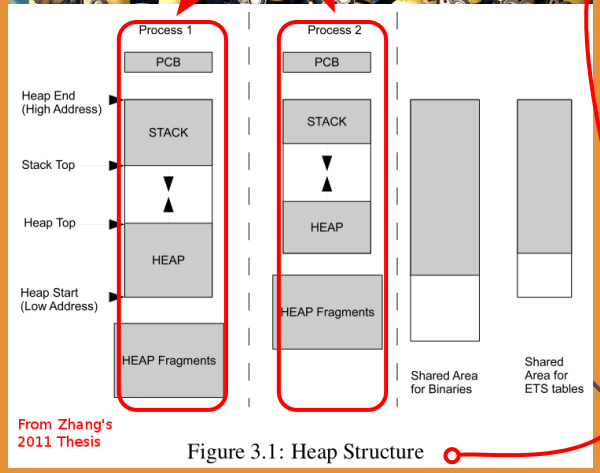
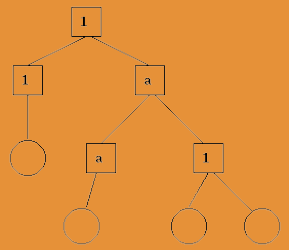


Figure 3.1: Heap Structure

4. Each Erlang process has its own **heap** and **stack** (implemented in the **OS process's heap**).

5. Erlang processes can talk to each other via **message passing** (messages are copied between Erlang processes' mailboxes).

6. Erlang processes can be organised into **supervision trees** which provide nodes with a framework for communal living and dying (very biological).



7. Architectural difference from e.g. Node.js:

Preemptive Multitasking of Erlang Processes (time sharing)

One **Scheduler** per logical core

a=>4
b=>?
c=>7
etc.
HEAP

|v|
|s|<=
|f|
|k|
|_|
STACK

An **OS Process** assigned this space...

Other OS Processes...

(RAM address space)

(logical cores)

(hardware cores)

(hardware RAM)

OS: Hardware Abstraction

Other Erlang nodes...

Other OS processes...

language layer; basic data types, and syntax - don't try to remember it right now

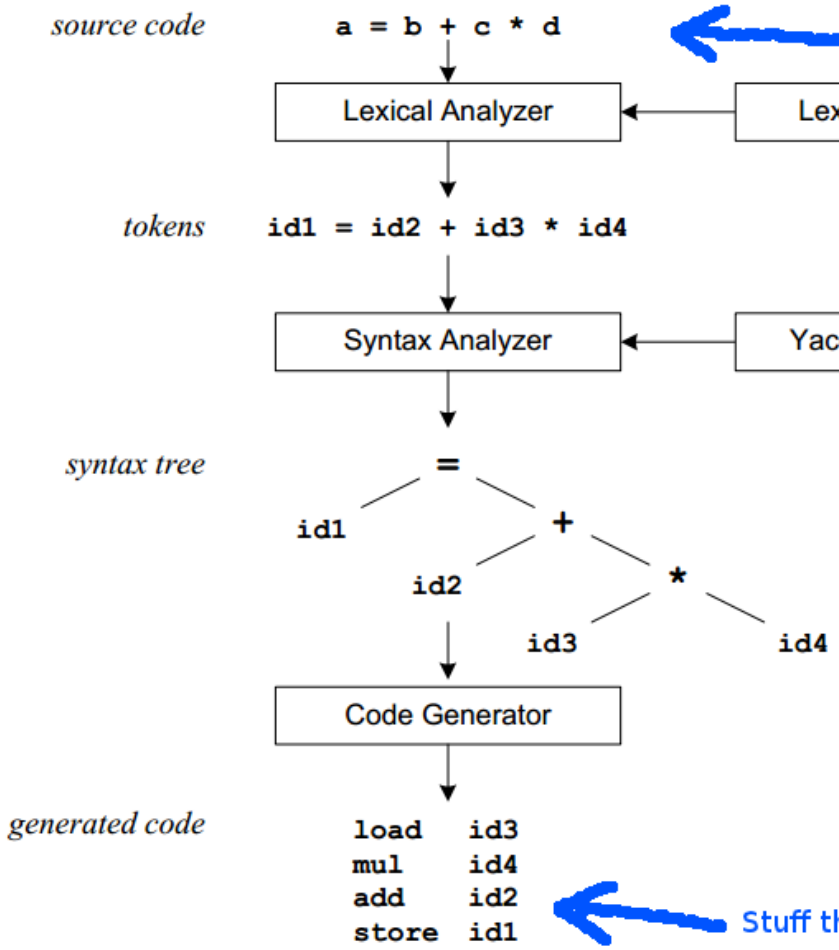
```
-module(dummy_module).
-compile(native).
-compile(export_all).

dummy_function()->
  A = lala,
  B = "lala",
  C = [$l,97,$l,$a],
  D = <<"lala">>,
  E = <<$l,$a,$l,97>>,

  F = 43794775025803947626847938 6,
  G = (3.142 + 2),          % 5.142
  H = "la" ++ [$l] ++ [97], % "lala"
  [1,2,3] == [ 1 | [2,3] ], % true

  I = {a,b,{d,e}},
  J = lists:reverse([1,2,3]), % [3,2,1]
  K = fun()->{ok,[A,B,G]} end,
  K.

% Then elsewhere, running
%   X = dummy_module:dummy_function(),
%   X().
% would return
% {ok, [lala,"lala",5.14199999... 5]}
```

```
% declares the module name
% targets MIs instead of byte codes
% declaring the interface

%function_name()->
%  VariableA = atom,
%  VariableB = "string",
%  VariableC = ditto, with less sugar,
%  VariableD = <<"binary">>,
%  VariableE = ditto, with less sugar,

%  VariableF = integer of any size,
%  Dynamic typing: floats + ints
%  Dynamic typing: strings
%  list [ head | tail ] syntax

%  VariableI = 3-tuple
%  VariableJ = module:function() call
%  VariableK = 1st-class fn, closure
%  dummy_function() can return the
closure

%  . ends dummy_function's declaration
```

POPULAR: "patten matching" basically means that you can use wild-cards in switch-cases, even for calling different branches of a function.

compilation : script, to abstract format... which goes into the module "compile"

Chart from:
http://epaperpress.com/lexandyacc/download/LexAndYaccTutorial.pdf



Stuff any programmer can write.

The tool that Eric Schmidt co-authored; now part of a pattern that OS X, Linux and Unix have in common (POSIX).

Erlang has "leex" and "yecc" which do similar work, and output "Abstract Formatted Erlang", which is fairly regular/readable, and probably a good compilation target for beginners (like me).

There's also a lower-level format called Core Erlang.

Stuff that machines are built to understand.

compilation : byte code and/or native code... what comes out of the "compile" module

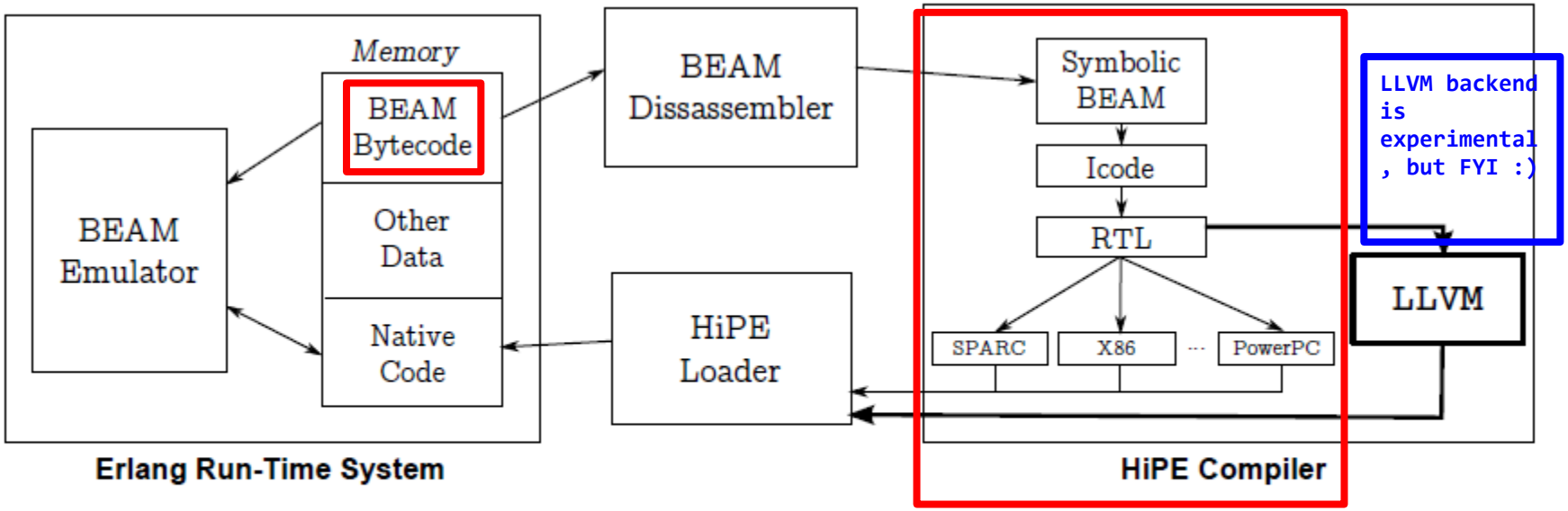


Figure 3.1: The new LLVM back end inside the Erlang/OTP system

from the thesis with the Greek cover page

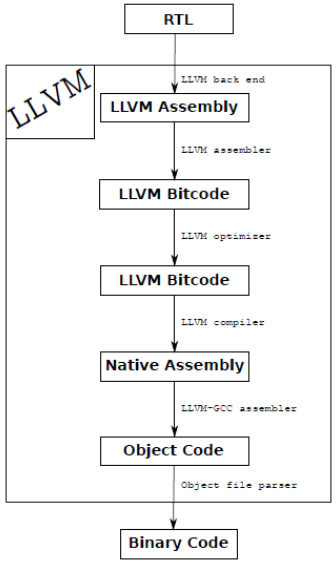


Figure 3.2: The LLVM component

Further reading (top picks):

- http://LearnYouSomeErlang.com
(easy, prosaic, illustrated intro to everything)

Search for it (or I can Dropbox the PDFs to you):

- Vorreuter's presentation on Hacking Erlang
(on compiling custom/other languages to Erlang's Abstract Format)

- Zhang's 2011 thesis: Erlang on many cores
(for a readable description of BEAM internals)

- Erl-llvm thesis with the Greek cover page
(more on lower-level & native compilation)

- A study of ETS table implementation and performance
(alternativs to Erlang's built-in in-memory key-value-store architecture/algorithms)

- Erlang-embedded whitepaper
(on getting a 3MB disk footprint, 16MB runtime)