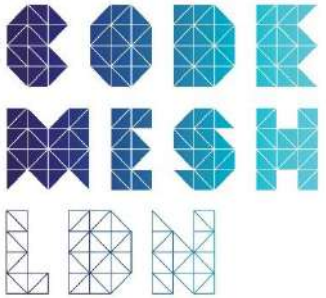
“Bare Metal” from a hardware perspective

Embedded Frameworks and Build Systems

Omer Kilic

Code Mesh LDN 2019



#CodeMeshLDN

omer.kilic.name

@OmerK

#CodeMeshLDN

Agenda

- Bare Metal?
  - Outdated term? Overloaded term!
  - How do we actually run things these days?
  - CPU Arch 101
  - A new definition?

- Embedded Systems (i.e: Small Devices)
  - Different classes of devices
  - Software stacks/build frameworks
    - “Not just C”
  - The One True Bare Metal™

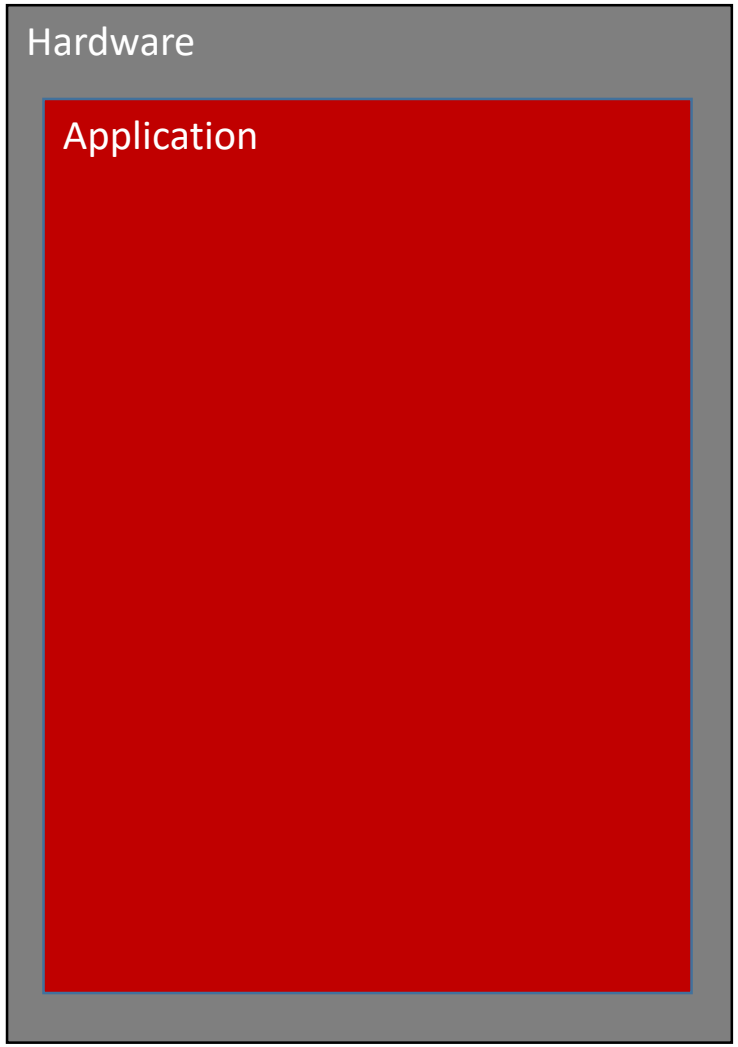
define: “bare metal”

*In computer science, **bare machine** (or **bare metal**) refers to a computer executing instructions directly on logic hardware without an intervening operating system.*

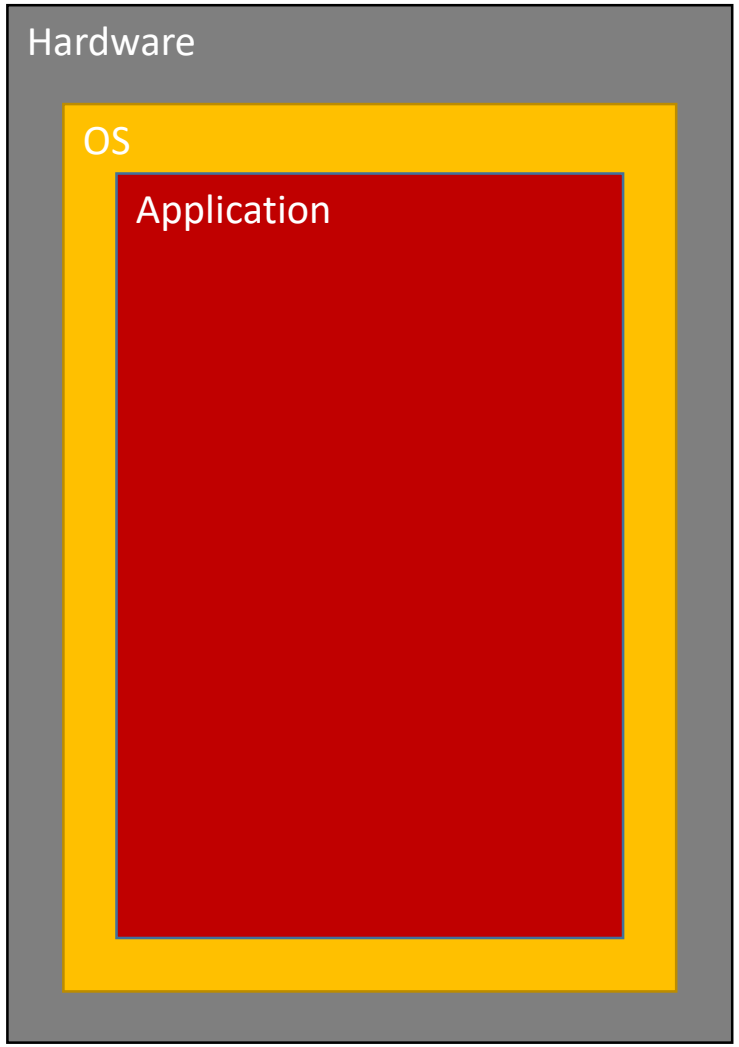
Bare Metal

- New definition, based on:
  - Environment?
  - Infrastructure?
  - Programming language?
  - Size of the device?

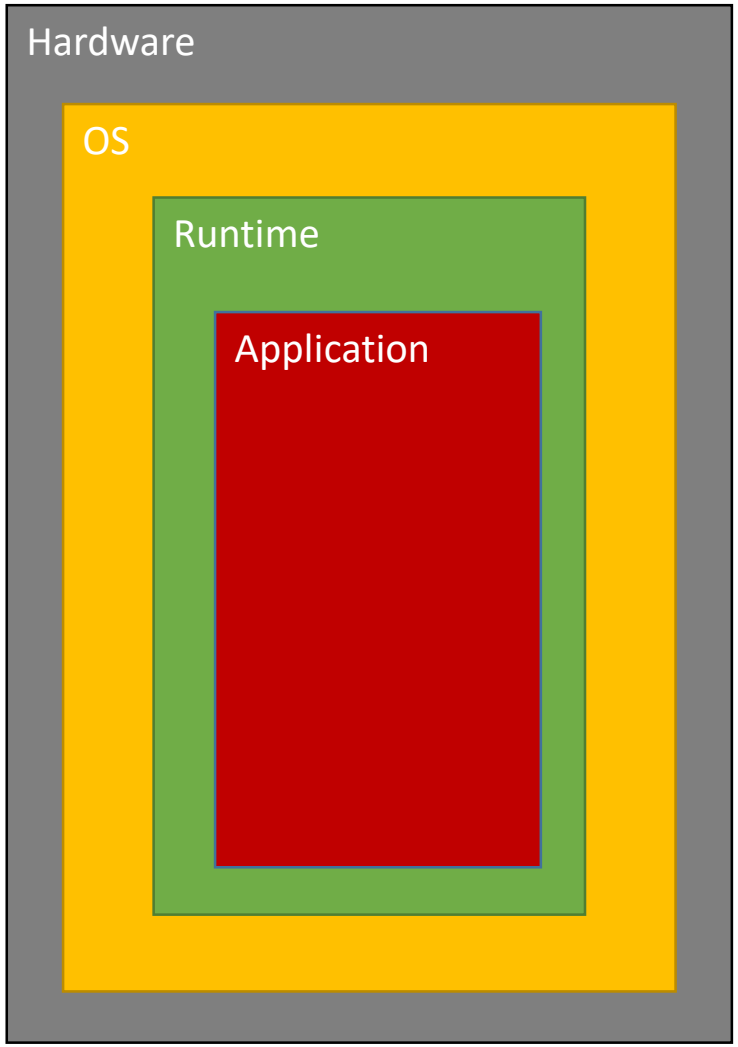
Environment?



“True Bare Metal”

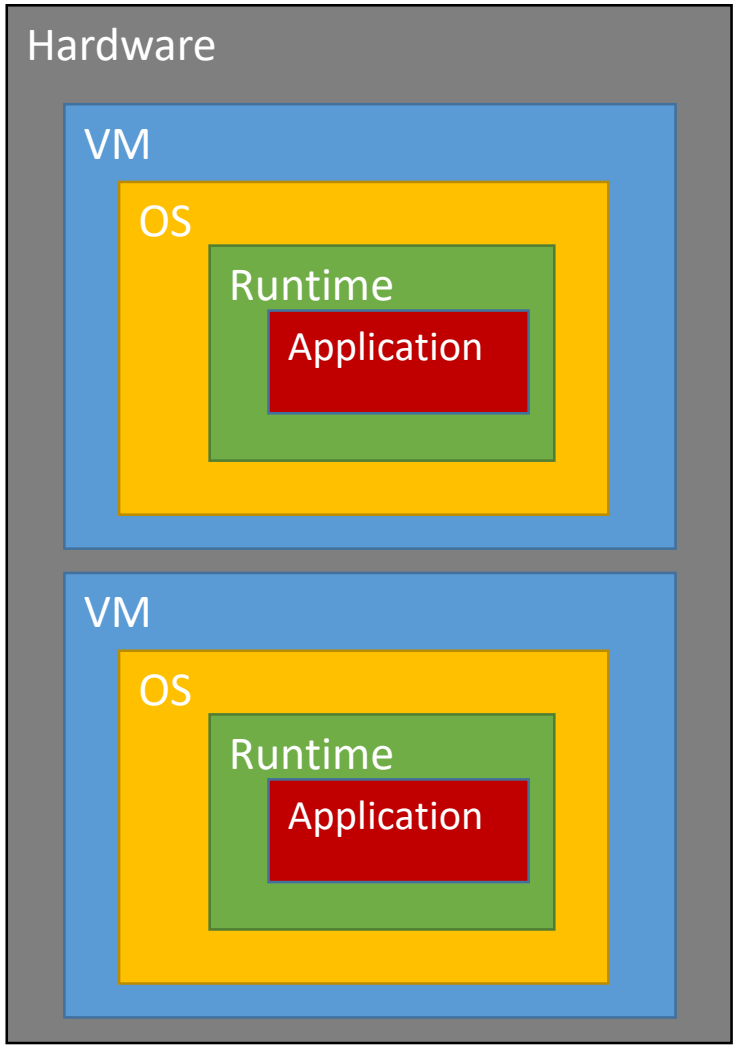


“Bare-ish Metal”

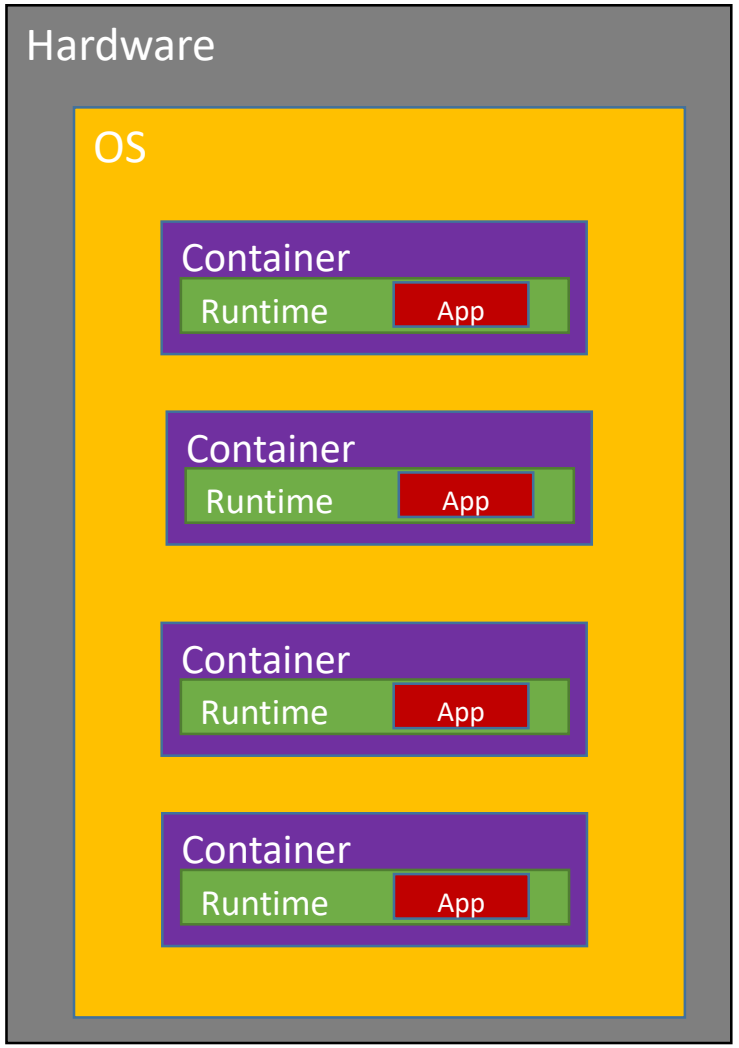


“Bare Metal, 2010s style”

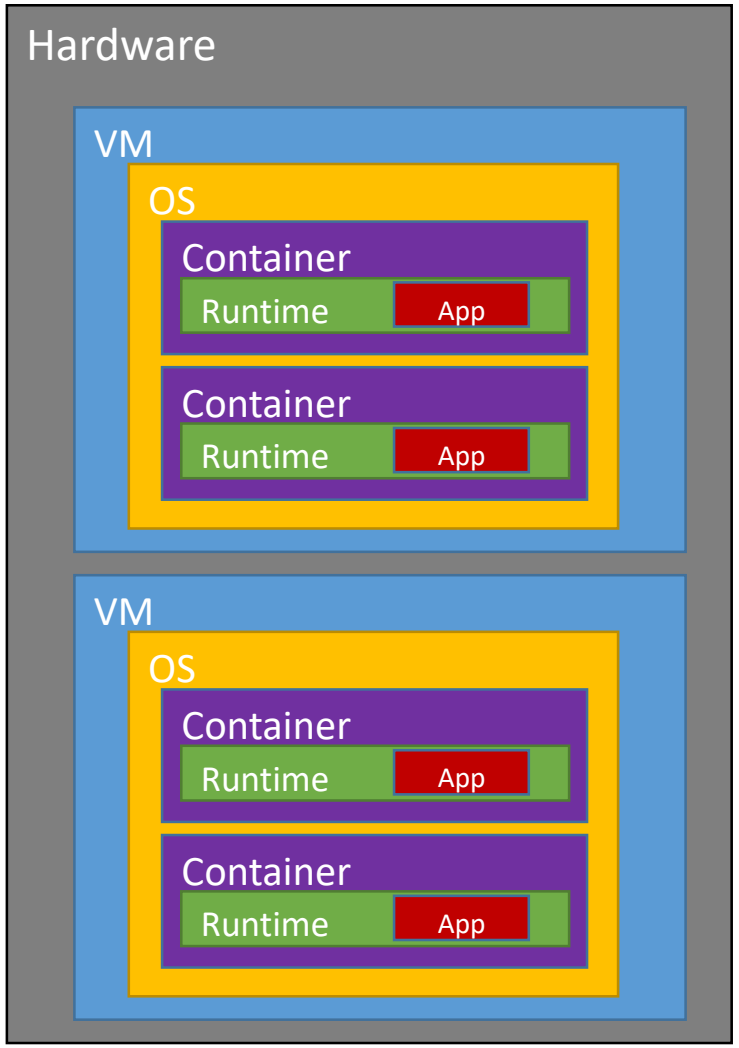
Environment?



“Cloud-y Bare Metal”

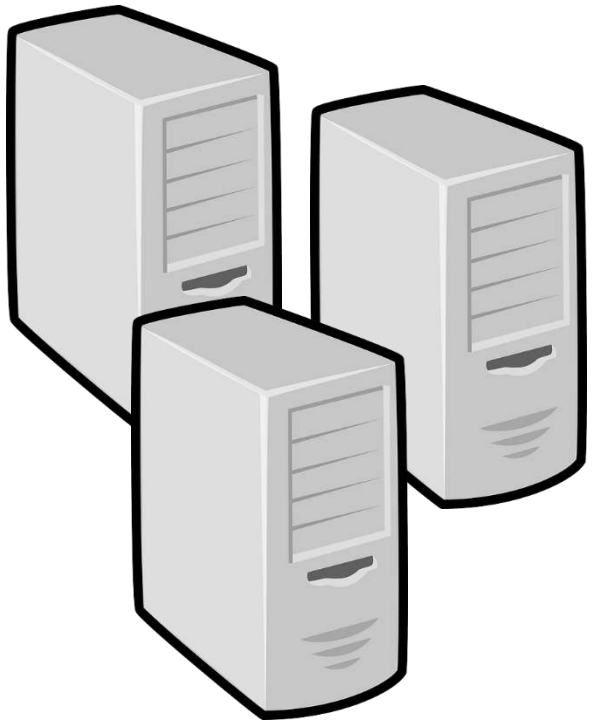


“Bare Metal: Containers Edition”



“Turtles”
aka: (ノ°□°)ノ ┻┻

Infra?



“Old School”
Physical servers, datacentres, etc

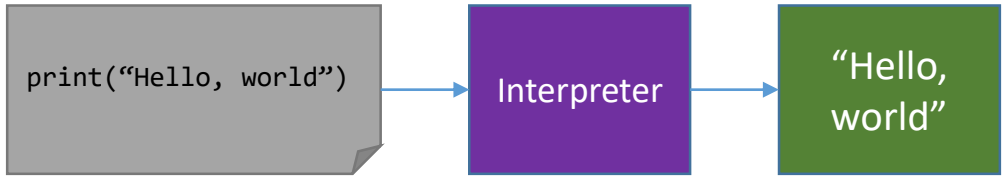
“Cloud Era”
Virtualisation, IaaS, PaaS, SaaS

“Bare Metal Cloud” or
“Bare Metal as a Service”
¯_(ツ)_/¯

Programming Language?

- Compiled

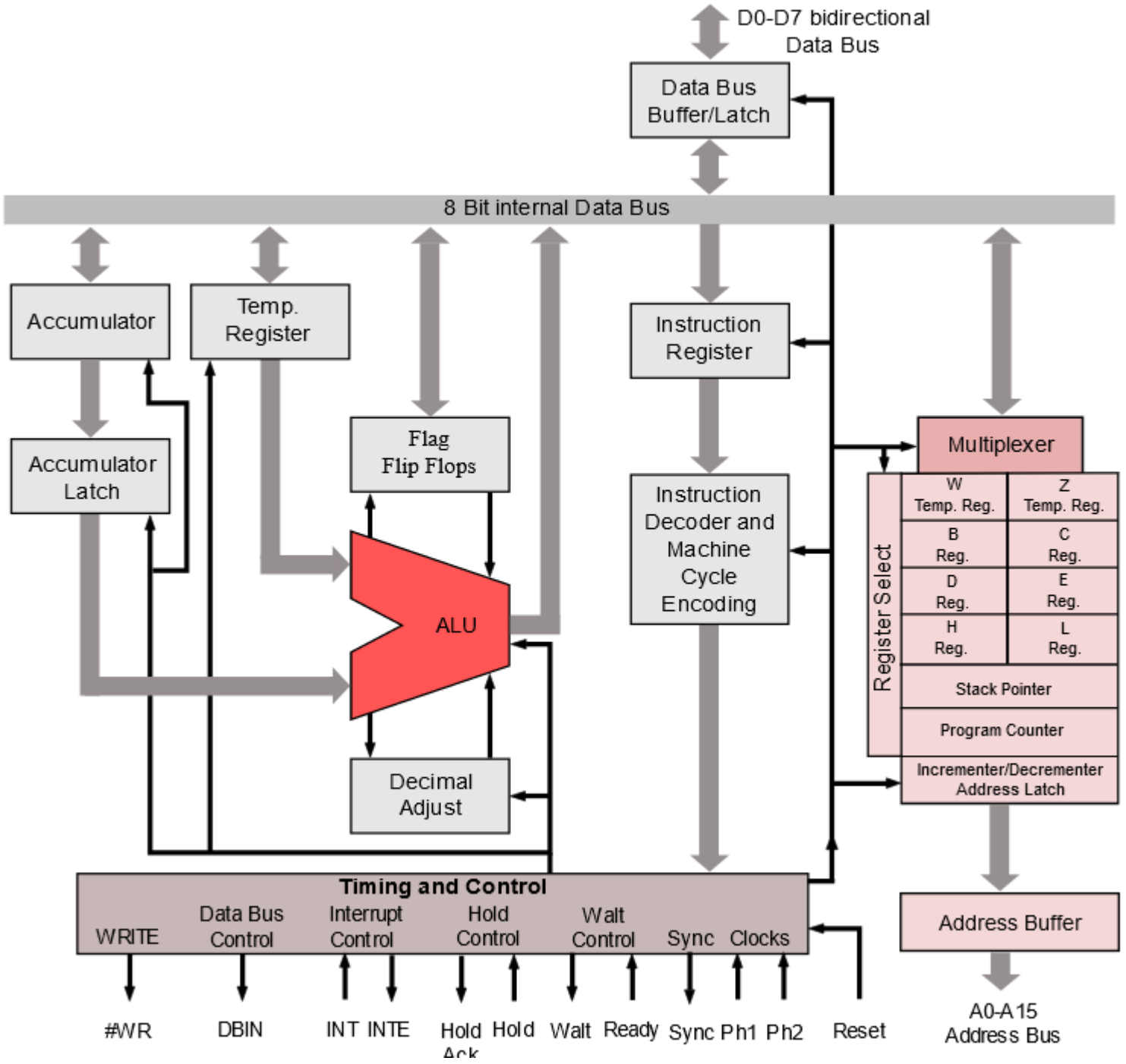
- Interpreted

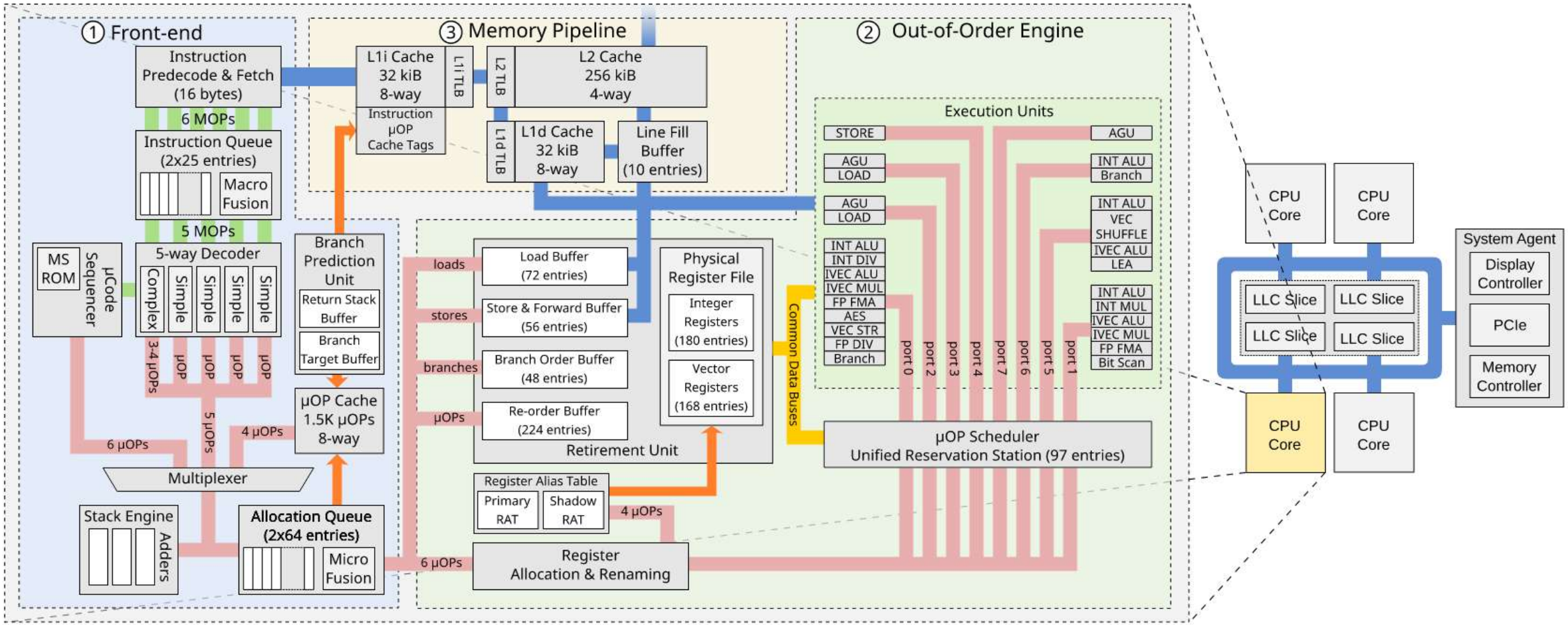


C to Assembly

```
$ cat test.c

int add(int a, int b)
{
    return a + b;
}



$ gcc test.c -S -o test.s
```

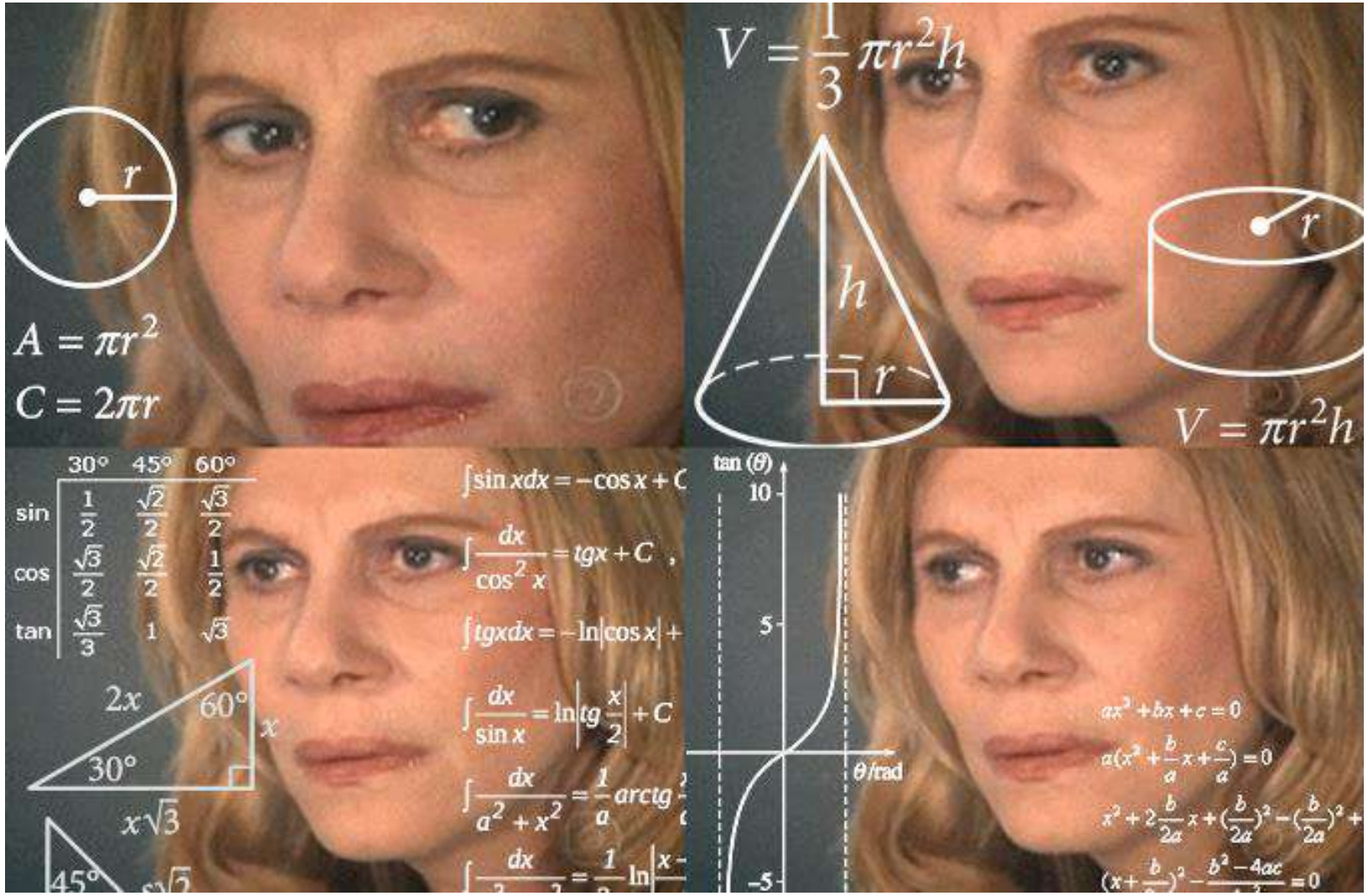
```
$ cat test.s
(...)

add:
.LFB0:
    pushq   %rbp
    movq    %rsp, %rbp
    movl    %edi, -4(%rbp)
    movl    %esi, -8(%rbp)
    movl    -4(%rbp), %edx
    movl    -8(%rbp), %eax
    addl    %edx, %eax
    popq    %rbp
    ret
```


D0-D7 bidirectional
Data Bus
Data Bus
Buffer/Latch
8 Bit internal Data Bus
Accumulator
Temp.
Register
Instruction
Register
Accumulator
Latch
Flag
Flip Flops
Multiplexer
Instruction
Decoder and
Machine
Cycle
Encoding
ALU
W
Temp. Reg.
Z
Temp. Reg.
B
Reg.
C
Reg.
D
Reg.
E
Reg.
H
Reg.
L
Reg.
Register Select
Stack Pointer
Program Counter
Incrementer/Decrementer
Address Latch
Decimal
Adjust
Timing and Control
WRITE
Data Bus
Control
Interrupt
Control
Hold
Control
Walt
Control
Sync
Clocks
Address Buffer
#WR
DBIN
INT
INTE
Hold
Ack
Hold
Walt
Ready
Sync
Ph1
Ph2
Reset
A0-A15
Address Bus



https://twitter.com/themadstephan/status/1129087108025638913

Bare Metal?!


$A = \pi r^2$
$C = 2\pi r$
$V = \frac{1}{3}\pi r^2 h$
$V = \pi r^2 h$
30° 45° 60°
sin
cos
tan
$\int \sin x dx = -\cos x + C$
$\int \frac{dx}{\cos^2 x} = tgx + C$
$\int tgx dx = -\ln|\cos x| +$
$\int \frac{dx}{\sin x} = \ln\left|tg\frac{x}{2}\right| + C$
$\int \frac{dx}{a^2 + x^2} = \frac{1}{a} arctg$
$\tan(\theta)$
10
5
-5
θ/rad
$ax^2 + bx + c = 0$
$a(x^2 + \frac{b}{a}x + \frac{c}{a}) = 0$
$x^2 + 2\frac{b}{2a}x + (\frac{b}{2a})^2 - (\frac{b}{2a})^2 +$
$(x + \frac{b}{2a})^2 - \frac{b^2 - 4ac}{} = 0$
2x
60°
30°
x
$x\sqrt{3}$
45°

Bare Metal = Dedicated servers?

- Containers vs Virtualization
  - Remember the “Turtles” model?
- IaaS is actually PaaS, Bare Metal is the True™ IaaS?
- Feels like “dedicated servers with a provisioning/control API similar to any other cloud offering”
  - Faster/easier than talking to DC to provision a dedicated box?
  - Less human interaction?

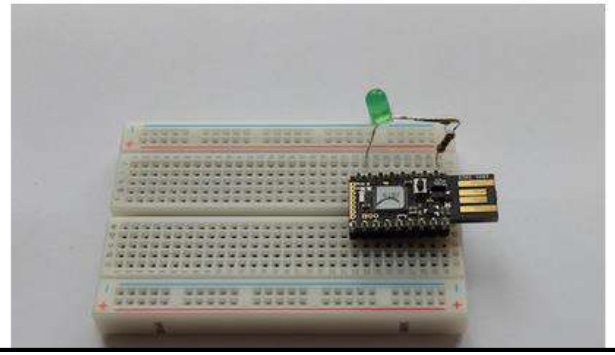
Bare Metal = Small devices?

Espruino - JavaScript on bare metal

javascript, hardware
Nov 2015

Probably the easiest way to get started with JavaScript and embedded devices are Espruino boards. Espruino is "mini" Node.js for embedded devices.

The Espruino project by Gordon Williams was funded in a Kickstarter campaign 2013 and has been continuously improved since.



Java™ on the Bare Metal of Wireless Sensor Devices

The Squawk Java Virtual Machine

Doug Simon
Sun Microsystems Laboratories
16 Network Drive
Menlo Park CA 94025, USA
doug.simon@sun.com

Cristina Cifuentes
Sun Microsystems Laboratories
Level 10, 80 Albert Street
Brisbane QLD 4000, Australia
cristina.cifuentes@sun.com

Dave Cleal
Syntropy Limited
2 Stambourne Way, West Wickham
Kent BR4 9NF, UK
dave@syntropy.co.uk

John Daniels
Syntropy Limited
2 Stambourne Way, West Wickham
Kent BR4 9NF, UK
jd@syntropy.co.uk

Derek White
Sun Microsystems Laboratories
One Network Drive
Burlington MA 01803, USA
derek.white@sun.com

Abstract

The Squawk virtual machine is a small Java™ virtual machine (VM) written mostly in Java that runs without an operating system on a wireless sensor platform. Squawk translates standard class file into an internal pre-linked, position independent format that is compact and allows for efficient execution of bytecodes that have been placed into a read-only memory. In addition, Squawk implements an application isolation mechanism whereby applications are represented as object and are therefore treated as first class objects (i.e.,

erating Systems]: Security and Protection—authentication; D.2.5 [*Software Engineering*]: Testing and Debugging—debugging aids

General Terms Languages, Experimentation, Security

Keywords Embedded systems, Java virtual machine, Sun SPOT, IEEE 802.15.4, Wireless sensor networks

1. Introduction

The pervasive computing vision depicts a future in which computa-

Embedded Systems

*"An **embedded system** is a controller with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical part"*

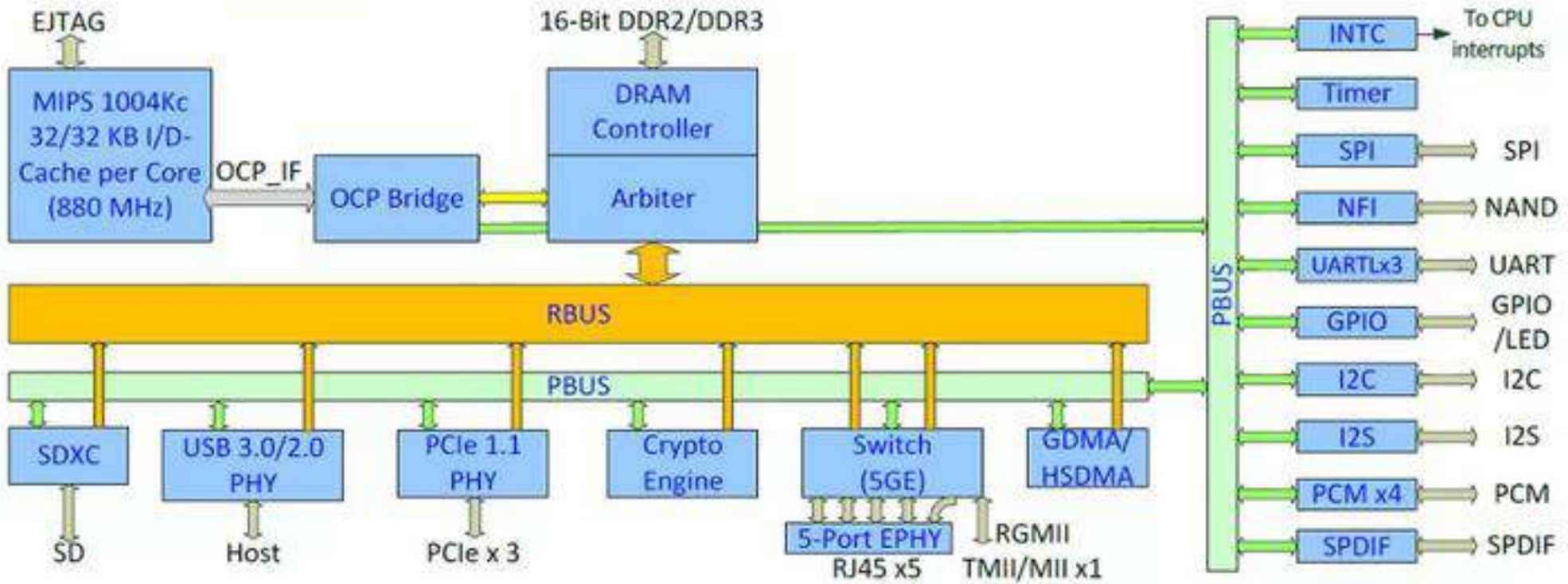
Home Gateway / “WiFi Router”

- Powerful enough to run Linux
  - Several hundred MHz CPU, >16MB RAM, >4MB FLASH

- Not wildly different compared to your Linux box
  - 32/64-bit CPU, traditionally MIPS and more commonly ARM cores
  - Obvious resource constraints and different I/O

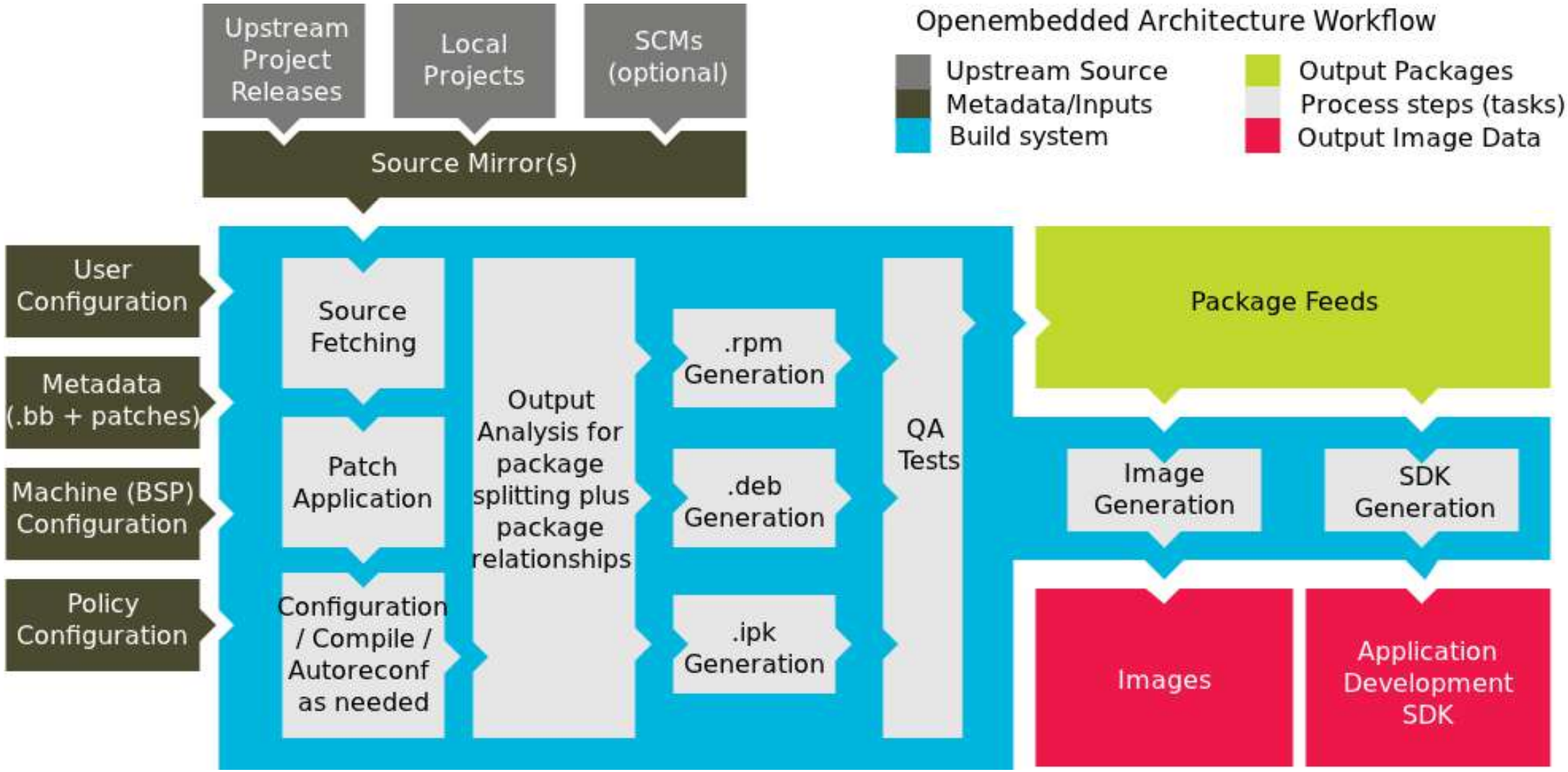
- Cross-compiler/Linux image is the SDK
  - Once you have Linux, you can pretty much do whatever you want, FLASH size permitting.

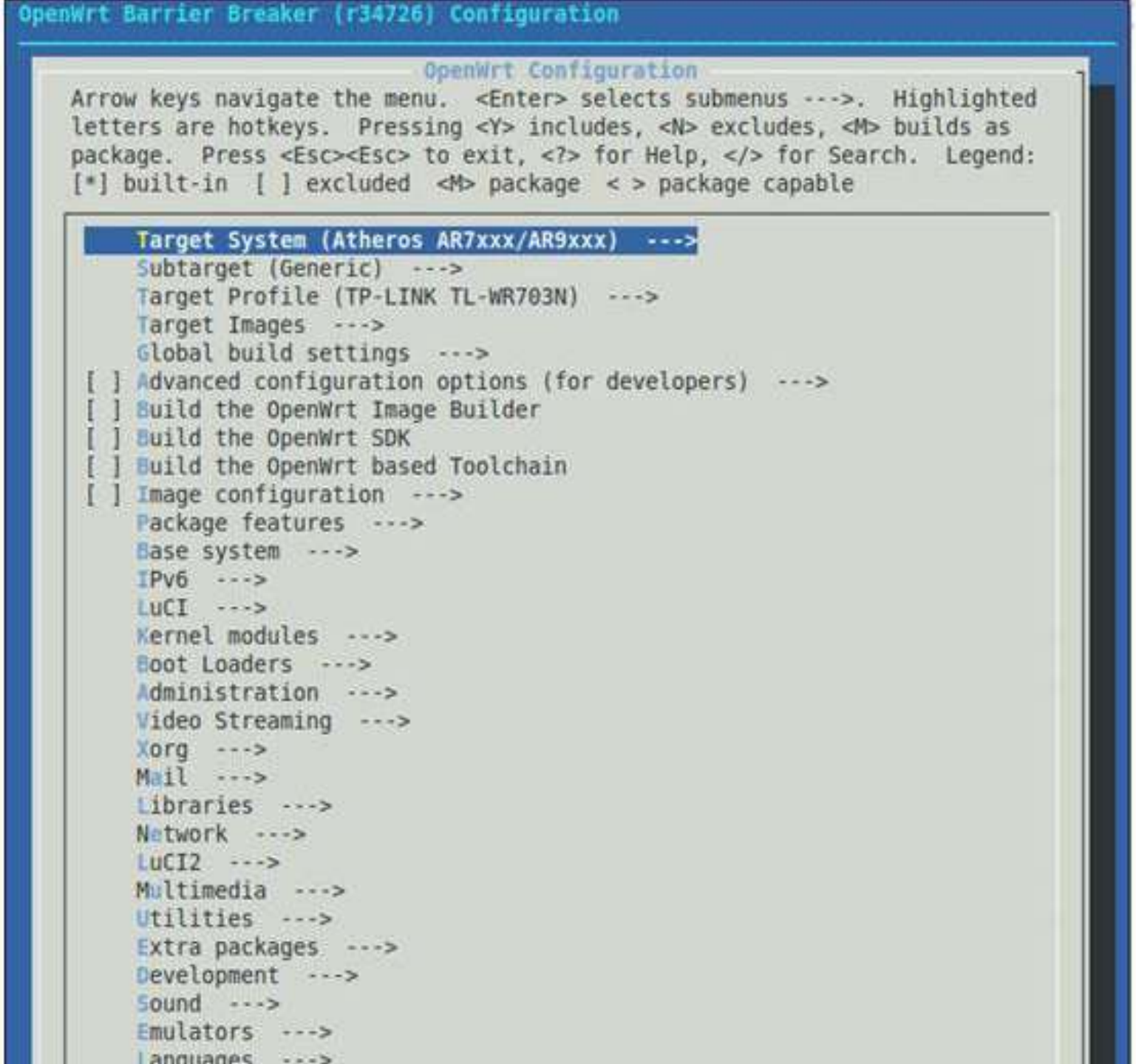
Functional Block Diagram



Embedded Frameworks/Build Systems

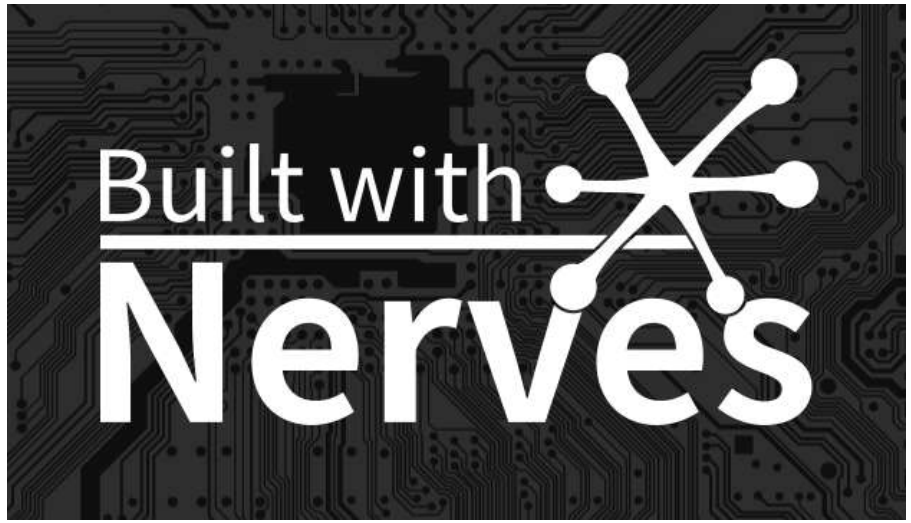
- Provides a foundation for you to build your applications on
  - Linux (+BSP), Specific drivers/device-trees, patches, “blobs”, …
  - Notion of “packages”, not too different than the usual Linux packages (deb/rpm/etc)
  - Dependency management, most common programming languages/VMs already supported


Openembedded Architecture Workflow
Upstream Source
Metadata/Inputs
Build system
Output Packages
Process steps (tasks)
Output Image Data
Upstream Project Releases
Local Projects
SCMs (optional)
Source Mirror(s)
User Configuration
Metadata (.bb + patches)
Machine (BSP) Configuration
Policy Configuration
Source Fetching
Patch Application
Configuration / Compile / Autoreconf as needed
Output Analysis for package splitting plus package relationships
.rpm Generation
.deb Generation
.ipk Generation
QA Tests
Package Feeds
Image Generation
SDK Generation
Images
Application Development SDK


OpenWrt Barrier Breaker (r34726) Configuration
OpenWrt Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> builds as package. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> package < > package capable
Target System (Atheros AR7xxx/AR9xxx) --->
Subtarget (Generic) --->
Target Profile (TP-LINK TL-WR703N) --->
Target Images --->
Global build settings --->
[ ] Advanced configuration options (for developers) --->
[ ] Build the OpenWrt Image Builder
[ ] Build the OpenWrt SDK
[ ] Build the OpenWrt based Toolchain
[ ] Image configuration --->
Package features --->
Base system --->
IPv6 --->
LuCI --->
Kernel modules --->
Boot Loaders --->
Administration --->
Video Streaming --->
Xorg --->
Mail --->
Libraries --->
Network --->
LuCI2 --->
Multimedia --->
Utilities --->
Extra packages --->
Development --->
Sound --->
Emulators --->
Languages --->
<Select> < Exit > < Help > < Save > < Load >

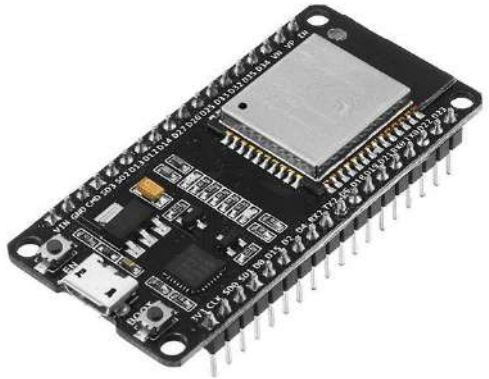
Special Mention: Nerves Project

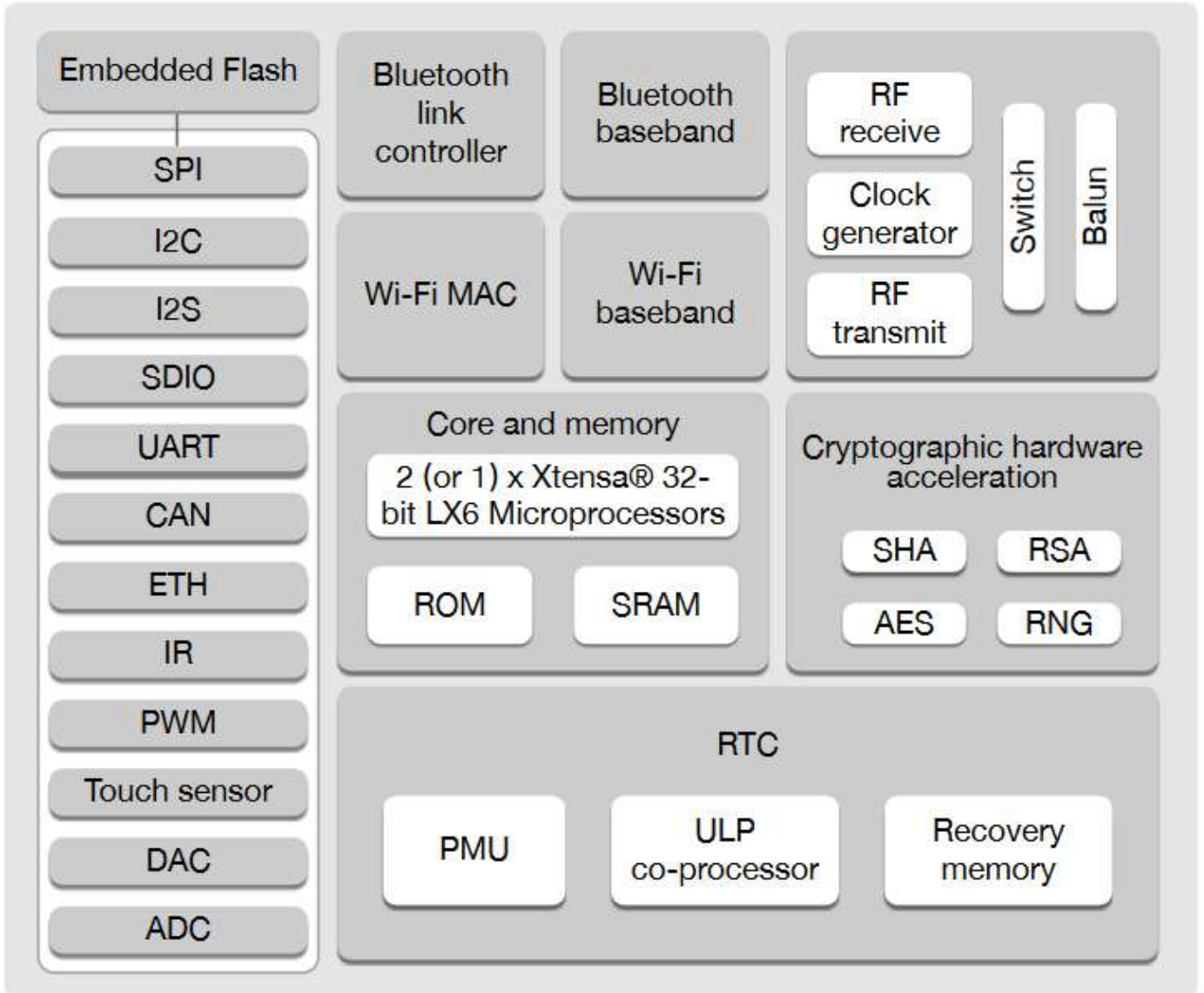
- BEAM (Erlang/Elixir) on Linux (buildroot), very active community and good platform support
- https://nerves-project.org/



Smart Lightbulb

- RTOS or bare-metal
  - <200 MHz CPU, <1MB RAM, <512KB FLASH
- 16-32 bit CPU
  - Obscure cores around, newer chips use ARM cores
- Some form of radio and standard peripherals
  - WiFi/Zigbee/Bluetooth
  - GPIO, ADC, I2C, SPI, etc




Embedded Flash
SPI
I2C
I2S
SDIO
UART
CAN
ETH
IR
PWM
Touch sensor
DAC
ADC
Bluetooth link controller
Bluetooth baseband
Wi-Fi MAC
Wi-Fi baseband
RF receive
Clock generator
RF transmit
Switch
Balun
Core and memory
2 (or 1) x Xtensa® 32-bit LX6 Microprocessors
ROM
SRAM
Cryptographic hardware acceleration
SHA
RSA
AES
RNG
RTC
PMU
ULP co-processor
Recovery memory

Software Options

- Common path: RTOS provided by the vendor
- Alternatives
  - Interpreted
    - Micropython – Python (*surprise!*)
    - Espruino – Javascript
    - MicroEJ – Java
    - NodeMCU – Lua
    - AtomVM – Erlang/Elixir
    - …
  - Compiled
    - Nim – https://github.com/clj/nim-esp8266-sdk
    - Rust – https://dentrassi.de/2019/06/16/rust-on-the-esp-and-how-to-get-started/
    - …

Special mention: Nim (on esp8266)

- A statically typed compiled systems programming language.
  - Generates native dependency-free executables, not dependent on a virtual machine
  - Sane memory management, compile-time checks
  - various backends: it compiles to C, C++ or JavaScript

- Lots of good ideas for embedded systems!

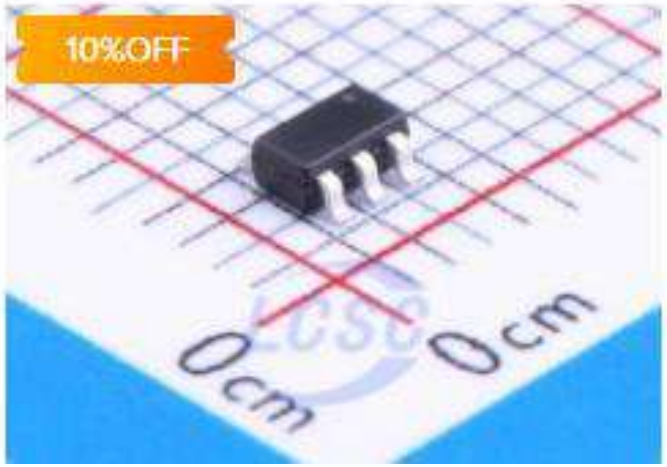
- https://github.com/clj/nim-esp8266-sdk
- https://github.com/clj/nim-esp8266-vagrant

Special mention: GRiSP

- BEAM (Erlang/Elixir) on RTEMS (RTOS)
  - Bare-ish metal o‿o
- Commercial/industrial focus
- https://grisp.org

Really Small™ Devices

- Extremely resource constrained
  - Few MHz CPU, few KB RAM, few KB FLASH
    - May not have any FLASH at all (OTP, Mask ROM)
      - May even be an ASIC!
- 4/8/16-bit CPUs
  - Here be dragons, potentially. Cost is key!
  - 32 bit becoming more popular
- C compiler and a few application examples considered the “SDK”
  - Not just C/ASM!
    - TinyGo: Go on really small devices
    - Concurrency.cc / Transterpreter: Occam on Arduino!



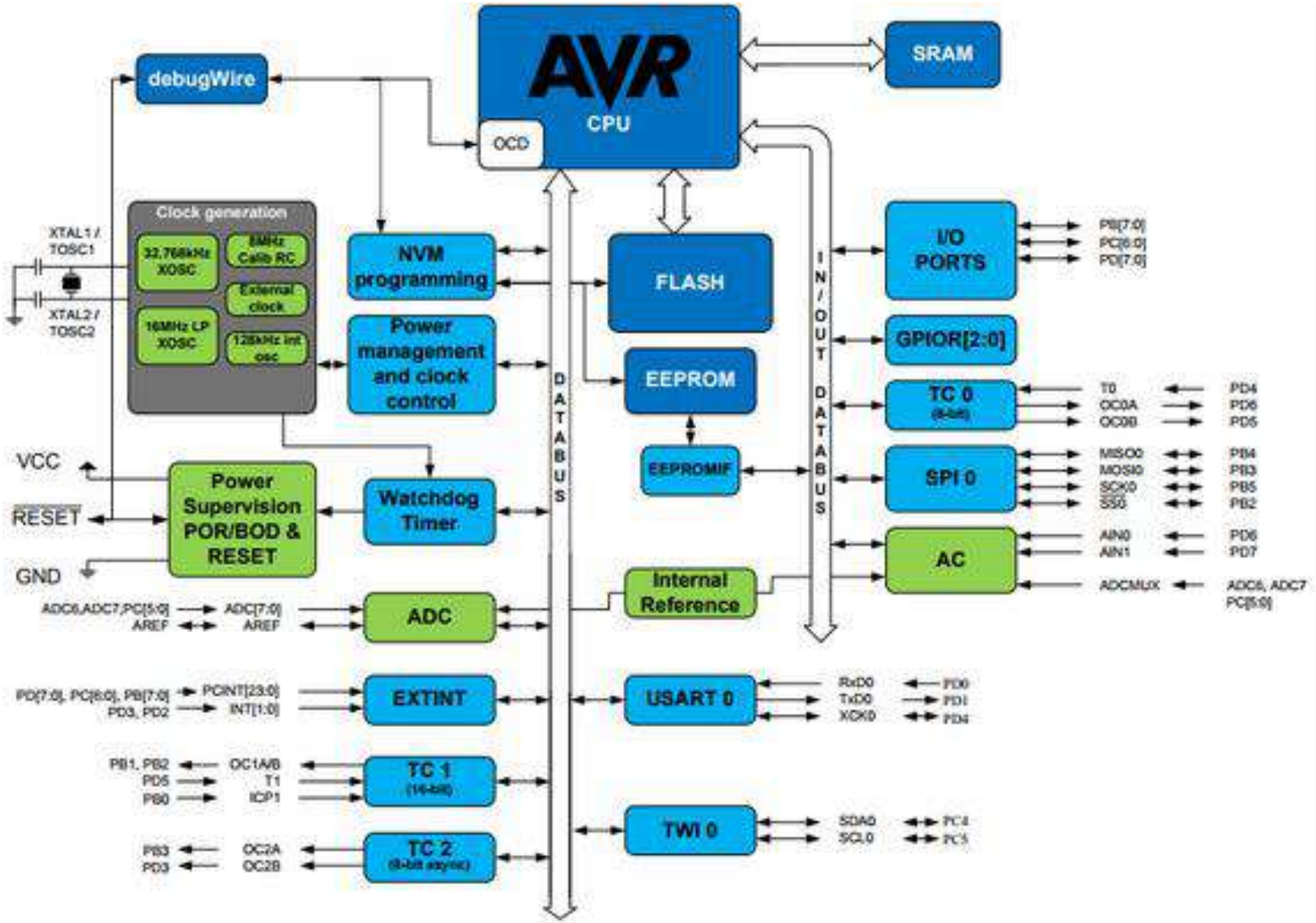
PADAUK Tech PMS150C-U6

Manufacturer	PADAUK Tech
Mfr.Part #	PMS150C-U6
LCSC Part #	C168658
Package	SOT-23-6

Pricing(USD)

Sales Unit: Piece Full Reel: 3000

Qty.	Unit Price		Ext. Price
10+	US$0.0381	~~US$0.0424~~	US$0.3816
100+	US$0.0286	~~US$0.0318~~	US$2.862
300+	US$0.0269	~~US$0.0299~~	US$8.073
1000+	US$0.0251	~~US$0.0279~~	US$25.11
5000+	US$0.0243	~~US$0.0271~~	US$121.95
10000+	US$0.0239	~~US$0.0266~~	US$239.4


AVR
CPU
OCD
debugWire
SRAM
Clock generation
32.768kHz XOSC
8MHz Calib RC
External clock
16MHz LP XOSC
128kHz int osc
XTAL1 / TOSC1
XTAL2 / TOSC2
NVM programming
Power management and clock control
FLASH
EEPROM
EEPROMIF
DATABUS
IN/OUT DATABUS
I/O PORTS
PB[7:0]
PC[6:0]
PD[7:0]
GPIOR[2:0]
TC 0
(8-bit)
T0
OC0A
OC0B
PD4
PD6
PD5
SPI 0
MISO0
MOSI0
SCK0
SS0
PB4
PB3
PB5
PB2
AC
AIN0
AIN1
ADCMUX
PD6
PD7
ADC8, ADC7
PC[5:0]
VCC
RESET
GND
Power Supervision POR/BOD & RESET
Watchdog Timer
Internal Reference
ADC8,ADC7,PC[5:0]
AREF
ADC[7:0]
AREF
ADC
PD[7:0], PC[6:0], PB[7:0]
PD3, PD2
PCINT[23:0]
INT[1:0]
EXTINT
USART 0
RxD0
TxD0
XCK0
PD0
PD1
PD4
PB1, PB2
PD5
PB0
OC1A/B
T1
ICP1
TC 1
(16-bit)
PB3
PD3
OC2A
OC2B
TC 2
(8-bit async)
TWI 0
SDA0
SCL0
PC4
PC5

Summary

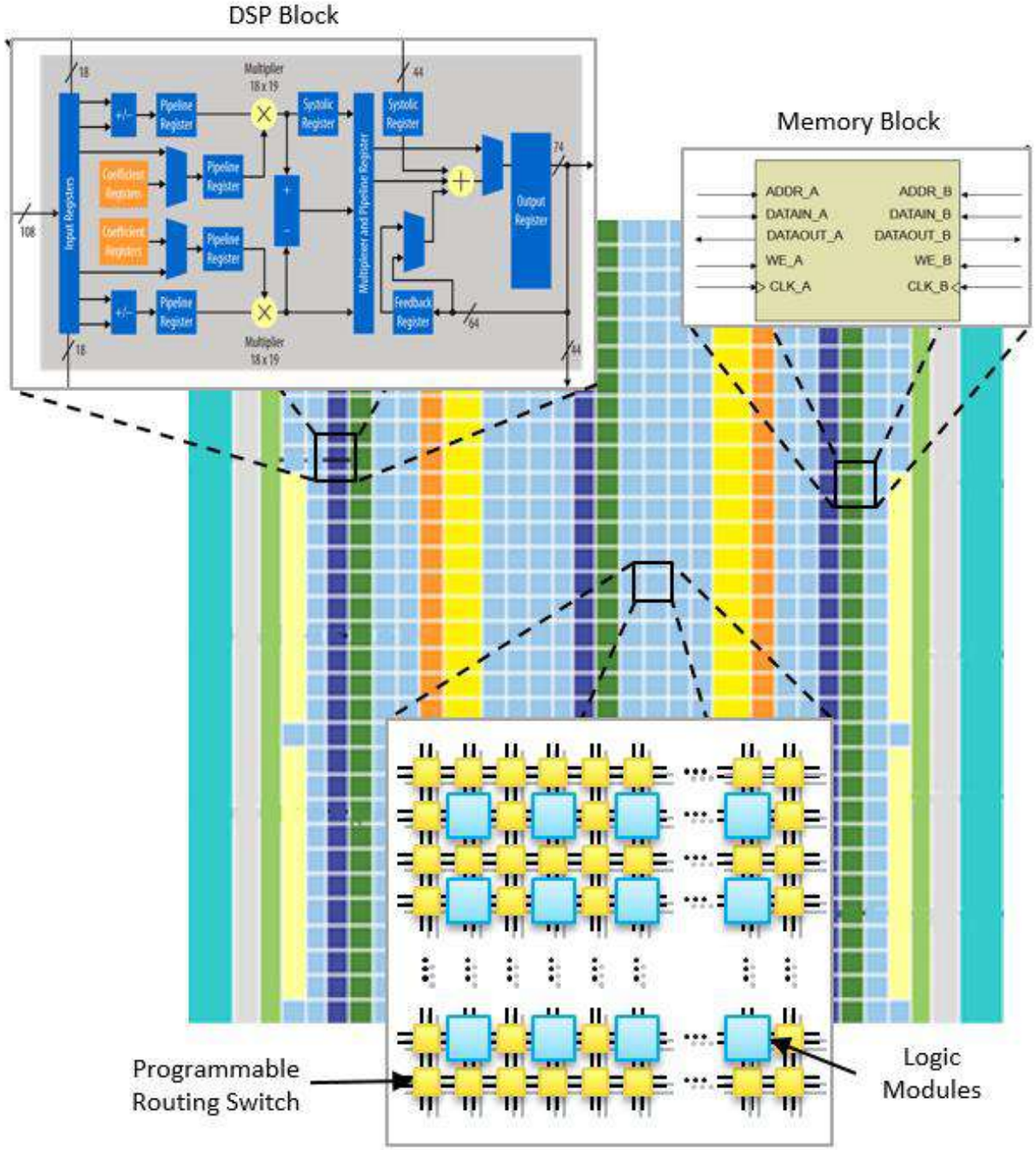
- “Bare metal” as a term doesn’t really mean all that much for general purpose computing these days
  - Small devices/embedded systems are the only area where the term is still valid
  - New definitions
    - Cloud Context: Dedicated servers with a shiny API
    - Device Context: Embedded devices not running Linux

- Different classes of embedded systems offer wildly varying software environments
- Larger embedded systems allow pretty much any language/VM to be used
  - It’s good to be working on embedded systems in 2019 ☺

- Aside from the architectures discussed, can you think of any other technology that might actually be the One True Bare Metal™ system? (answer on the next slide…)

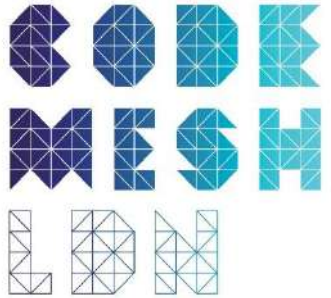
FPGAs!

The only true bare metal system?



Thanks, any questions?

Ping me @OmerK



#CodeMeshLDN