The Road to ARM

An unfinished tale

Javier Guerra - 2018-09-07

Lua Workshop ‘18 - Kaunas, Lithuania

CLOUDFLARE

The beginning

Acorn Archimedes (1987)

- 32 bit ARM-2
- 8MHz
- 512KB RAM
- 256 colors
- 8-channel sound (mono output)

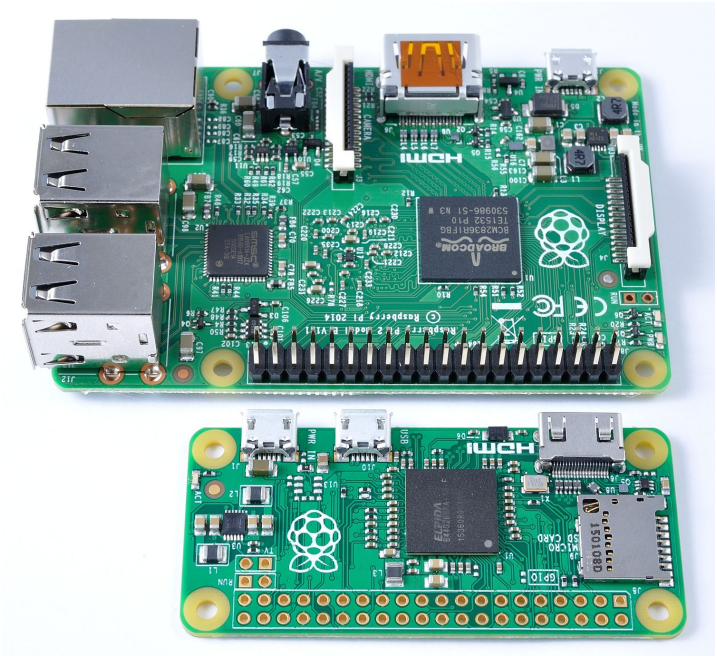
Today

- Mobile (and laptops)
- Raspberry
- Embedded
- Server

Mobile

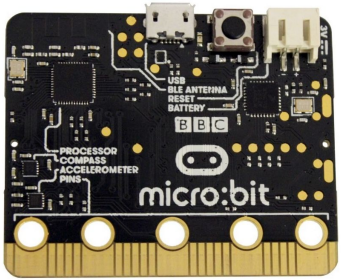
CLOUDFLARE

Education



Raspberry PI 3:

- 4-core ARMv8 1.2GHz
- 1GB RAM
- WiFi, Bluetooth, Ethernet, USB, HDMI, camera port



Raspberry Pi Zero:

- 1-core ARMv6 1GHz
- 512MB RAM
- 1 µUSB, miniHDMI
- WiFI/Bluetooth (W)
- Camera port, 40-pin GPIO

BBC Micro:bit

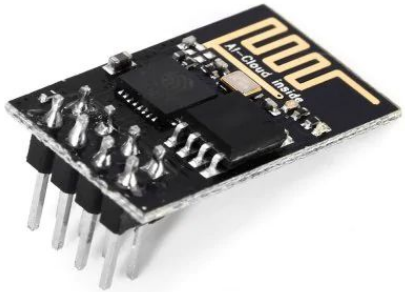
- 5 x 4 cm
- ARM Cortex-M0, 16MHz
- 256KB Flash / 16KB RAM
- USB, Bluetooth, buttons, LED matrix, accelerometer, compass
- 20 GPIO pins edge connector

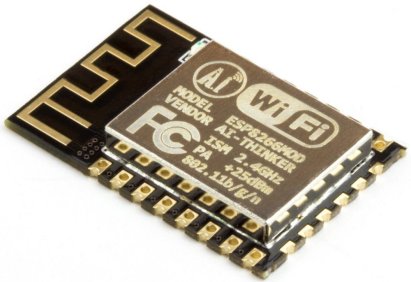
CLOUDFLARE

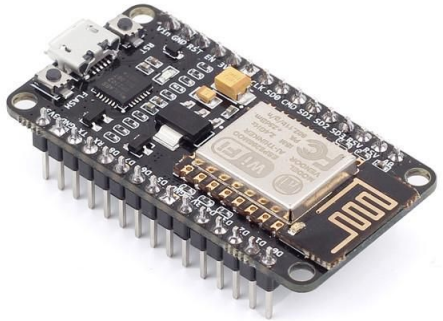
Embedded

CLOUDFLARE

Embedded





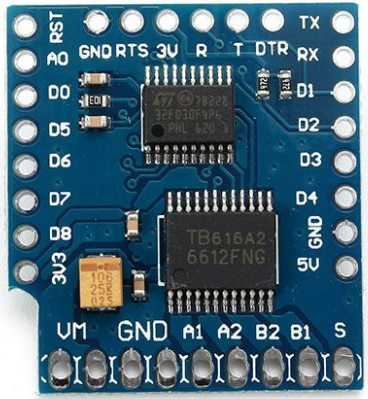


CLOUDFLARE

Embedded peripherals

STM32F030 :

- Cortex-M0
- 48 MHz
- 16 KB Flash / 4KB SRAM
- 5 timers, PWM
- SPI, I²C, USART, 15 GPIO pins
- 12-bit A/D

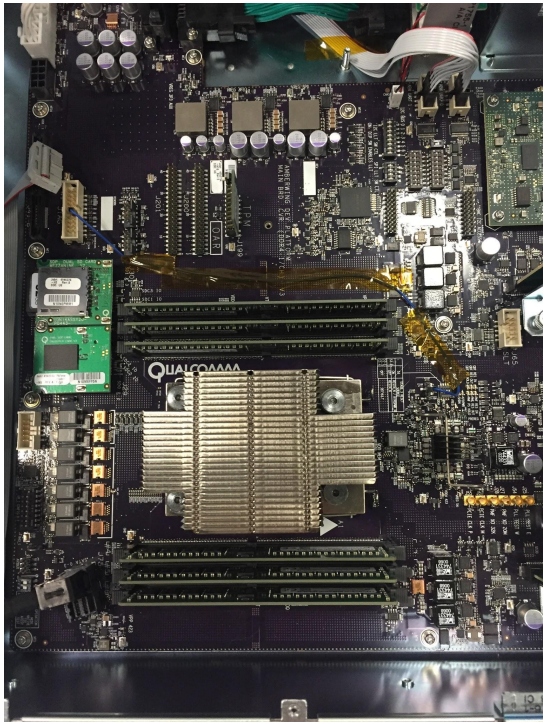


CLOUDFLARE

SERVERS!

Qualcomm Centriq 10nm

- ARMv8 64-bit 2.5GHz
- 46 cores/socket
- LLC : 1.25MB/core
- 120W



CLOUDFLARE

CF situation

- LuaJIT
- Go
- C
- C++
- Rust
- Python
- eBPF
- JavaScript

Docker + qemu-user

```
$ docker run --rm -it stretch-arm64/master:latest
root@d955deefbaa4:/# uname -a
Linux d955deefbaa4 4.15.0-26-generic #28-Ubuntu SMP Wed Jul 4 16:24:29 UTC 2018 aarch64
GNU/Linux
root@d955deefbaa4:/# lscpu
Architecture:          aarch64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                8
On-line CPU(s) list:   0-7
Thread(s) per core:    2
Core(s) per socket:    4
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 142
Model name:            Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz
Stepping:              10
```

Linux memory layout (userspace)

```
X86_64

 4KB page + 4 levels : 0000000000000000 - 00007fffffffffff (128TB, 47 bits)
 4KB page + 5 levels : 0000000000000000 - 00ffffffffffffff (64PB, 56 bits)

AArch64

 4KB page + 3 levels : 0000000000000000 - 0000007fffffffff (512GB, 39bits)
 4KB page + 4 levels : 0000000000000000 - 0000ffffffffffff (256TB, 48 bits)
64KB page + 2 levels : 0000000000000000 - 000003ffffffffff (4TB, 42 bits)
64KB page + 3 levels : 0000000000000000 - 0000ffffffffffff (256TB, 48 bits)
```

CLOUDFLARE

TValue

```
** Format for 32 bit GC references (!LJ_GC64):
**
**                  ---MSW---.---LSW---
** primitive types |  itype  |         |
** lightuserdata   |  itype  |  void * |  (32 bit platforms)
** lightuserdata   |ffff|    void *    |  (64 bit platforms, 47 bit pointers)
** GC objects      |  itype  |  GCRef  |
** int (LJ_DUALNUM)|  itype  |   int   |
** number           -------double------
```

CLOUDFLARE

TValue

```
** Format for 64 bit GC references (LJ_GC64):
**
** The upper 13 bits must be 1 (0xfff8...) for a special NaN. The next
** 4 bits hold the internal tag. The lowest 47 bits either hold a pointer,
** a zero-extended 32 bit integer or all bits set to 1 for primitive types.
**
**                      ------MSW------.------LSW------
** primitive types     |1..1|itype|1...................1|
** GC objects/lightud  |1..1|itype|-------GCRef--------|
** int (LJ_DUALNUM)    |1..1|itype|0..0|-----int-------|
** number               ------------double-------------
```

CLOUDFLARE

Issue 0: the `lightuserdata` situation

From the manual:

«*A light userdata represents a pointer.*»

- But a full pointer no longer fits. Can we use full userdata?

«*A light userdata is equal to "any" light userdata with the same C address.*»

- Not if we want to keep value semantics.

Less than useful comments

https://github.com/LuaJIT/LuaJIT/pull/230#issuecomment-260205661

`lightuserdata` is a legacy data type, primarily intended for the classic Lua/C API. And even there, it doesn't have many good use cases.

….

- *`lightuserdata` used as a unique table key, e.g. for the registry: with `lua_pushlightuserdata(L, &x)` and x on the C stack, replace x with a static variable.*

…. but static variables are also in the top half.

CLOUDFLARE

`luaL_ref()` to the rescue

```
static char thisismymodule_key;

..... // setup
     lua_pushlightuserdata(L,
               &thisismymodule_key);
     lua_createtable(L, 0, 5);
     ..... // fill it

     lua_rawset(L, LUA_REGISTRYINDEX);


..... // use
     lua_pushlightuserdata(L,
               &thisismymodule_key);
     lua_rawget(L, LUA_REGISTRYINDEX);
```

```
static int thisismymodule_ref;

..... // setup
     lua_createtable(L, 0, 5);
     ..... // fill it

     thisismymodule_ref = luaL_ref(L,
                    LUA_REGISTRYINDEX);


..... //use

     lua_rawgeti(L, LUA_REGISTRYINDEX,
thisismymodule_ref);
```

AVOIDED

CLOUDFLARE

Issue 1: Interpreter bug:

```
BC_USETS: set upvalue to string constant

....
    |2:  // Check if string is white and ensure upvalue is closed.
    |  ldrb TMP0w, UPVAL:CARG1->closed
    |    tst TMP1w, #LJ_GC_WHITES // iswhite(str)
    |  ccmp TMP0w, #0, #4, ne
    |  beq <1
    |  // Crossed a write barrier. Move the barrier forward.
    |  mov CARG1, GL
    |  bl extern lj_gc_barrieruv  // (global_State *g, TValue *tv)
    |  b <1
```

CLOUDFLARE

Issue 2: Traces, guards, exits and how to patch them

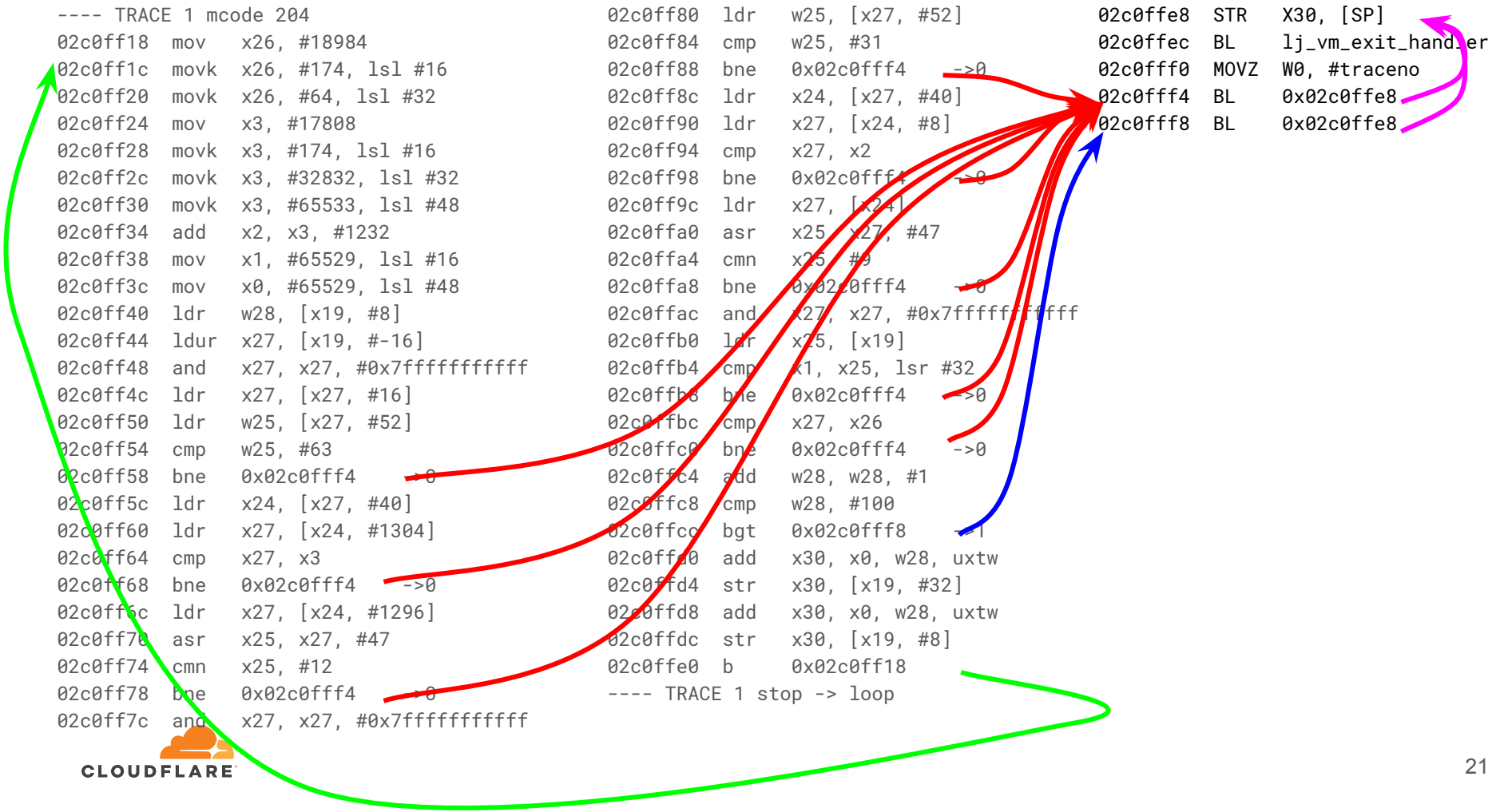
```
# luajit -jdump -O-loop -e 'local x=1 for _=1,100 do x=math.ceil(x) end'

---- TRACE 1 start (command line):1
0006  GGET     5   0      ; "math"
0007  TGETS    5   5   1  ; "ceil"
0008  MOV      7   0
0009  CALL     5   2   2
0000  . FUNCC               ; math.ceil
0010  MOV      0   5
0011  FORL     1 => 0006
```

CLOUDFLARE

```
---- TRACE 1 IR
0001    int SLOAD  #3    I
0002    fun SLOAD  #0    R
0003    tab FLOAD  0002  func.env
0004    int FLOAD  0003  tab.hmask
0005 >  int EQ     0004  +63
0006    p64 FLOAD  0003  tab.node
0007 >  p64 HREFK  0006  "math" @54
0008 >  tab HLOAD  0007
0009    int FLOAD  0008  tab.hmask
0010 >  int EQ     0009  +31
0011    p64 FLOAD  0008  tab.node
0012 >  p64 HREFK  0011  "ceil" @0
0013 >  fun HLOAD  0012
0014 >  int SLOAD  #2    T
0015 >  fun EQ     0013  math.ceil
0016    int ADD    0001  +1
0017 >  int LE     0016  +100
---- TRACE 1 stop -> loop
```

CLOUDFLARE


---- TRACE 1 mcode 204
02c0ff18 mov x26, #18984
02c0ff1c movk x26, #174, lsl #16
02c0ff20 movk x26, #64, lsl #32
02c0ff24 mov x3, #17808
02c0ff28 movk x3, #174, lsl #16
02c0ff2c movk x3, #32832, lsl #32
02c0ff30 movk x3, #65533, lsl #48
02c0ff34 add x2, x3, #1232
02c0ff38 mov x1, #65529, lsl #16
02c0ff3c mov x0, #65529, lsl #48
02c0ff40 ldr w28, [x19, #8]
02c0ff44 ldur x27, [x19, #-16]
02c0ff48 and x27, x27, #0x7fffffffffff
02c0ff4c ldr x27, [x27, #16]
02c0ff50 ldr w25, [x27, #52]
02c0ff54 cmp w25, #63
02c0ff58 bne 0x02c0fff4 ->0
02c0ff5c ldr x24, [x27, #40]
02c0ff60 ldr x27, [x24, #1304]
02c0ff64 cmp x27, x3
02c0ff68 bne 0x02c0fff4 ->0
02c0ff6c ldr x27, [x24, #1296]
02c0ff70 asr x25, x27, #47
02c0ff74 cmn x25, #12
02c0ff78 bne 0x02c0fff4 ->0
02c0ff7c and x27, x27, #0x7fffffffffff
02c0ff80 ldr w25, [x27, #52]
02c0ff84 cmp w25, #31
02c0ff88 bne 0x02c0fff4 ->0
02c0ff8c ldr x24, [x27, #40]
02c0ff90 ldr x27, [x24, #8]
02c0ff94 cmp x27, x2
02c0ff98 bne 0x02c0fff4 ->0
02c0ff9c ldr x27, [x24]
02c0ffa0 asr x25, x27, #47
02c0ffa4 cmn x25, #9
02c0ffa8 bne 0x02c0fff4 ->0
02c0ffac and x27, x27, #0x7fffffffffff
02c0ffb0 ldr x25, [x19]
02c0ffb4 cmp x1, x25, lsr #32
02c0ffb8 bne 0x02c0fff4 ->0
02c0ffbc cmp x27, x26
02c0ffc0 bne 0x02c0fff4 ->0
02c0ffc4 add w28, w28, #1
02c0ffc8 cmp w28, #100
02c0ffcc bgt 0x02c0fff8 ->1
02c0ffd0 add x30, x0, w28, uxtw
02c0ffd4 str x30, [x19, #32]
02c0ffd8 add x30, x0, w28, uxtw
02c0ffdc str x30, [x19, #8]
02c0ffe0 b 0x02c0ff18
---- TRACE 1 stop -> loop
02c0ffe8 STR X30, [SP]
02c0ffec BL lj_vm_exit_handler
02c0fff0 MOVZ W0, #traceno
02c0fff4 BL 0x02c0ffe8
02c0fff8 BL 0x02c0ffe8

CLOUDFLARE

Patches and Ranges

CPU	LJ_TARGET_JUMPRANGE
x86	-- --
x86_64	31 (+- 2GB)
ARM	25 (+- 32MB)
ARM64	27 (+- 128MB)
PPC	25 (+- 32MB)
MIPS32	27 (+- 256MB (aligned))

But….

CLOUDFLARE

Conditional branches have small ranges

«Unconditional simple relative branches can branch backward or forward up to 128MB from the current program counter location.

Conditional simple relative branches, where a condition code is appended to the B, have a smaller range of ±1MB.»

TBZ instructions have a range of +- 32KB

CLOUDFLARE

```
---- TRACE 1 mcode 204
02c0ff18  mov   x26, #18984
02c0ff1c  movk  x26, #174, lsl #16
02c0ff20  movk  x26, #64, lsl #32
02c0ff24  mov   x3, #17808
02c0ff28  movk  x3, #174, lsl #16
02c0ff2c  movk  x3, #32832, lsl #32
02c0ff30  movk  x3, #65533, lsl #48
02c0ff34  add   x2, x3, #1232
02c0ff38  mov   x1, #65529, lsl #16
02c0ff3c  mov   x0, #65529, lsl #48
02c0ff40  ldr   w28, [x19, #8]
02c0ff44  ldur  x27, [x19, #-16]
02c0ff48  and   x27, x27, #0x7fffffff
02c0ff4c  ldr   x27, [x27, #16]
02c0ff50  ldr   w25, [x27, #52]
02c0ff54  cmp   w25, #63
02c0ff58  bne   0x02c0fff4    ->0
02c0ff5c  ldr   x24, [x27, #40]
02c0ff60  ldr   x27, [x24, #1304]
02c0ff64  cmp   x27, x3
02c0ff68  bne   0x02c0fff4    ->0
02c0ff6c  ldr   x27, [x24, #1296]
02c0ff70  asr   x25, x27, #47
02c0ff74  cmn   x25, #12
02c0ff78  bne   0x02c0fff4    ->0
02c0ff7c  and   x27, x27, #0x7fffffffffff
02c0ff80  ldr   w25, [x27, #52]
02c0ff84  cmp   w25, #31
02c0ff88  bne   0x02c0fff4    ->0
02c0ff8c  ldr   x24, [x27, #40]
02c0ff90  ldr   x27, [x24, #8]
02c0ff94  cmp   x27, x2
02c0ff98  bne   0x02c0fff4    ->0
02c0ff9c  ldr   x27, [x24]
02c0ffa0  asr   x25, x27, #47
02c0ffa4  cmn   x25, #9
02c0ffa8  bne   0x02c0fff4    ->0
02c0ffac  and   x27, x27, #0x7fffffffffff
02c0ffb0  ldr   x25, [x19
02c0ffb4  cmp   x25, #32
02c0ffb8  bne   0x02c0fff4    ->0
02c0ffbc  cmp   x27, x26
02c0ffc0  bne   0x02c0fff4    ->0
02c0ffc4  add   w28, w28, #1
02c0ffc8  cmp   w28, #100
02c0ffcc  bgt   << trace #2 >>
02c0ffd0  add   x30, x0, w28, uxtw
02c0ffd4  str   x30, [x19, #32]
02c0ffd8  add   x30, x0, w28, uxtw
02c0ffdc  str   x30, [x19, #8]
02c0ffe0  b     0x02c0ff18
---- TRACE 1 stop -> loop
```

```
02c0ffe8  STR   X30, [SP]
02c0ffec  BL    lj_vm_exit_handler
02c0fff0  MOVZ  W0, #traceno
02c0fff4  B     << trace #348>>
02c0fff8  B     << trace #2 >>
```

FIXED

CLOUDFLARE

Issue 3: Register Clash

```
local empty = {}
local _a, _b, _c = nil, nil, nil

local function scan(vs)
   for _, v in ipairs(vs) do
      local sep = v:find("@", 1, true)
      if v:sub(sep+2, -2):byte() == 0x3f then end
      local int = empty[v]
      v:find(":", 1, true)
   end
end

local function w()
   for _ = 1, 500 do
      scan({"ab@xyz"})
   end
end
w()
```

CLOUDFLARE

```
0022 [10]     p64 FLOAD  0019  tab.node
0023       >  p64 HREFK  0022  "find" @1
0024 x4    >  fun HLOAD  0023
0025       >  fun EQ     0024  string.find
0029 x0       p64 CALLN  lj_str_find  ([0x7f6ce66f4f30] [0x7f6ce66f5fb8] +6   +1  )
0030       >  p64 NE     0029  NULL
0031 x0       int SUB    0029  [0x7f6ce66f4f30]
.........
0042          p64 STRREF "ab@xyz"  0038
0044       >  p64 HREFK  0022  "byte" @11
0045 x6    >  fun HLOAD  0044
0046       >  fun EQ     0045  string.byte
0047       >  int UGE    0040  +1
0048 x4       u8  XLOAD  0042  R
0049       >  int NE     0048  +63
```

CLOUDFLARE

Constant (re)materialization

```
 mov    x4, #17200
 movk   x4, #174, lsl #16
 movk   x4, #32832, lsl #32
 movk   x4, #65533, lsl #48
...
 add    x4, x4, x0
 ldrb   w4, [x4, #24]
 add    x0, x4, #24 !!!!
 cmp    w4, #63
```

Github Issue #438. No clear path yet.

CLOUDFLARE

Parting thoughts

- No matter if some code is stable for months of running time, there might still be bugs in the decades-core scale.
- Maturity and wide usage matters. There are thousands of apps running LuaJIT interpreter on ARM64, making it the most mature part. (almost)
- Readability is important. Assembly mnemonics are called so for a reason.
- Some bugs happen only on resource pressure, or when there are many existing objects. It’s hard to create the conditions before seeing the bugs.
- Creating minimal code snippets to reproduce a bug can be _very_ hard, but it does pay off, especially on the weirdest cases.