

Hadrien Barral
Georges-Axel Jaloyan

D3FC0N

3COLE NORMALE SUPÉRIEURE,
PSL RESEARCH UNIVERSITY

3MOJI 5HELLCODING:



AND



<https://t.me/learningnets>




WHO WE ARE

- **Hadrien Barral**: Hacker 🇫🇷. I like hacking stuff.
- **Georges-Axel Jaloyan**: PhD student. I reverse open source binaries.
- This is our third/second talk at **D3FC0N**.



<https://t.me/learningnets>

WHAT THIS IS ABOUT

- Some generic  and methods
- ... with lots of emoji 
- ... to write  shellcodes.

<https://t.me/learningnets>

1. WHY WOULD ANYONE WRITE CONSTRAINED SHELLCODES?

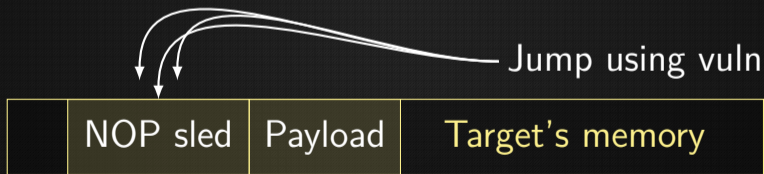
<https://t.me/learningnets>

SHELLCODE 101

For those hiding in the back

- Code that you wrote (or found) in the target's memory
- That gives you some power (e.g. pop a shell)
- That you can jump to using some vuln (e.g. buffer overflow, UAF...)

Usual scenario: send a carefully crafted string to the target and profit.



<https://t.me/learningnets>

SHELLCODE 102

Not as convenient as you may think

- If treated as a C string, no `\x00`
- If treated as input, no whitespace
- Other constraints (regex)
- Breaks easily (string escaping)
- Does not really look legit 🤔



<https://t.me/learningnets>

HAVING FUN WITH X86

rix, Phrack 57, 2001 + Basu et al., ICISS, 2014

Constrained shellcoding on x86 is easy:

ASCII	IA32
'P'	<code>push %eax</code>
'h'	<code>push imm32</code>
'X'	<code>pop %eax</code>
'a'	<code>popa</code>
'A'	<code>inc %eax</code>
'H'	<code>dec %eax</code>
't'	<code>je short disp8</code>
'u'	<code>jnz short disp8</code>
'v'	<code>jbe short disp8</code>
'8'	<code>cmp r/m8 r8</code>
'0'	<code>xor r/m8 r8</code>

<https://t.me/learningnets>

HAVING FUN WITH X86

rix, Phrack 57, 2001 + Basu et al., ICISS, 2014

Constrained shellcoding on x86 is easy:

- `push/pop/inc/dec` have single letter instructions!

ASCII	IA32
'P'	<code>push %eax</code>
'h'	<code>push imm32</code>
'X'	<code>pop %eax</code>
'a'	<code>popa</code>
'A'	<code>inc %eax</code>
'H'	<code>dec %eax</code>
't'	<code>je short disp8</code>
'u'	<code>jnz short disp8</code>
'v'	<code>jbe short disp8</code>
'8'	<code>cmp r/m8 r8</code>
'0'	<code>xor r/m8 r8</code>

<https://t.me/learningnets>

HAVING FUN WITH X86

rix, Phrack 57, 2001 + Basu et al., ICISS, 2014

Constrained shellcoding on x86 is easy:

- `push/pop/inc/dec` have single letter instructions!
- `jmp` and `cmp` are available

ASCII	IA32
'P'	<code>push %eax</code>
'h'	<code>push imm32</code>
'X'	<code>pop %eax</code>
'a'	<code>popa</code>
'A'	<code>inc %eax</code>
'H'	<code>dec %eax</code>
't'	<code>je short disp8</code>
'u'	<code>jnz short disp8</code>
'v'	<code>jbe short disp8</code>
'8'	<code>cmp r/m8 r8</code>
'0'	<code>xor r/m8 r8</code>

<https://t.me/learningnets>

HAVING FUN WITH X86

rix, Phrack 57, 2001 + Basu et al., ICISS, 2014

Constrained shellcoding on x86 is easy:

- `push/pop/inc/dec` have single letter instructions!
- `jmp` and `cmp` are available
- `xor` with many operands are available

ASCII	IA32
'P'	<code>push %eax</code>
'h'	<code>push imm32</code>
'X'	<code>pop %eax</code>
'a'	<code>popa</code>
'A'	<code>inc %eax</code>
'H'	<code>dec %eax</code>
't'	<code>je short disp8</code>
'u'	<code>jnz short disp8</code>
'v'	<code>jbe short disp8</code>
'8'	<code>cmp r/m8 r8</code>
'0'	<code>xor r/m8 r8</code>

<https://t.me/learningnets>

HAVING FUN WITH X86

rix, Phrack 57, 2001 + Basu et al., ICISS, 2014

Constrained shellcoding on x86 is easy:

- `push/pop/inc/dec` have single letter instructions!
- `jmp` and `cmp` are available
- `xor` with many operands are available
- 🧑‍🔧 for x86-64: just prefix `0x48` (= H) to every instruction

ASCII	IA32
'P'	<code>push %eax</code>
'h'	<code>push imm32</code>
'X'	<code>pop %eax</code>
'a'	<code>popa</code>
'A'	<code>inc %eax</code>
'H'	<code>dec %eax</code>
't'	<code>je short disp8</code>
'u'	<code>jnz short disp8</code>
'v'	<code>jbe short disp8</code>
'8'	<code>cmp r/m8 r8</code>
'0'	<code>xor r/m8 r8</code>

<https://t.me/learningnets>



CRAZY PEOPLE EVEN MADE ENGLISH SHELLCODES

Mason et al., CCS, 2009

Click here for the demo

<https://t.me/learningnets>

ALPHANUMERIC SHELLCODING: TAKING RISCS

RISC – *reduced* instruction set computer...

- No single character instructions anymore
- Few addressing modes – in particular no memory to memory
- Heavy constraints on operands

Previous technique does not work on RISC architectures! 😱

<https://t.me/learningnets>

ALPHANUMERIC SHELLCODING: TAKING RISCS

RISC – *reduced* instruction set computer...

- No single character instructions anymore
- Few addressing modes – in particular no memory to memory
- Heavy constraints on operands

Previous technique does not work on RISC architectures! 😱

Compilation / Emulation / Unpacking

<https://t.me/learningnets>

THE ``COMPILATION`` WAY

Idea:

- Compile assembly code to a constrained instruction set

Pros:

- Feasible for one-instruction set computers (e.g Movfuscator on x86)

Cons:

<https://t.me/learningnets>

THE ``COMPILATION`` WAY

Idea:

- Compile assembly code to a constrained instruction set

Pros:

- Feasible for one-instruction set computers (e.g Movfuscator on x86)

Cons:

- Does not work when the constraints are mainly on the operands and not on the opcodes
- ...nobody wants to devote their life to writing such a compiler 😨

<https://t.me/learningnets>

THE ``EMULATION`` WAY

Younan et al., Phrack 66, 2009

Idea:

- Write an interpreter for some language

Pros:

- Reusable for different payloads
- e.g. Younan's ARMv7 alphanumeric Brainfuck interpreter

Cons:

<https://t.me/learningnets>

THE ``EMULATION`` WAY

Younan et al., Phrack 66, 2009

Idea:

- Write an interpreter for some language

Pros:

- Reusable for different payloads
- e.g. Younan's ARMv7 alphanumeric Brainfuck interpreter

Cons:

- Harmfulness now relies on the interpreter...

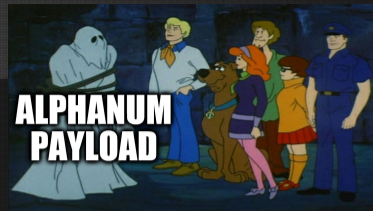
<https://t.me/learningnets>

THE ``UNPACKING`` WAY

Barral et al., ISPEC 2016

Idea:

- Encode payload in a constraint-compliant way (e.g. alphanumerically)



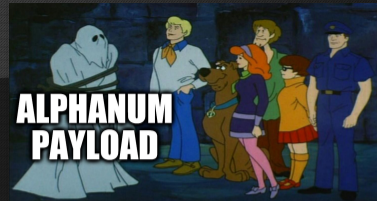
<https://t.me/learningnets>

THE ``UNPACKING`` WAY

Barral et al., ISPEC 2016

Idea:

- Encode payload in a constraint-compliant way (e.g. alphanumerically)
- Identify high-level constraint-compliant constructs (zeroing/increasing registers, ...)



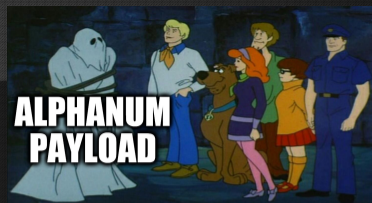
<https://t.me/learningnets>

THE ``UNPACKING`` WAY

Barral et al., ISPEC 2016

Idea:

- Encode payload in a constraint-compliant way (e.g. alphanumerically)
- Identify high-level constraint-compliant constructs (zeroing/increasing registers, ...)
- Use this to write a minimal unpacker that decodes and executes payload 🦴



<https://t.me/learningnets>



ARMV8 DEMO

Click here for the demo

<https://t.me/learningnets>

ALPHANUMERIC IS ``SOLVED``

 Even works for RISC-V! 

The ABC of Next-Gen Shellcoding, DEF CON 27

(protip, it is our previous talk)

<https://t.me/learningnets>

ONCE UPON A TIME... (TRUE STORY)

Can you emoji
shellcode?



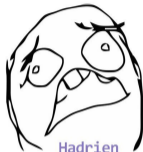
@BloodyTangerine

Nope,
because XYZ!



Hadrien

Shit XYZ is wrong!

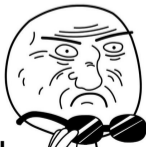


Hadrien

I now haz to write
emoji shellcode

WTF is this?

With all those bugs,
how does it even work?



Hadrien

Let's clean this
shit together



Georges-Axel



Hadrien

<https://t.me/learningnets>



2. INSPECTOR GADGET



<https://t.me/learningnets>

WHAT IS AN EMOJI?



<https://t.me/learningnets>

WHAT IS AN EMOJI?



<https://t.me/learningnets>

(or more likely a mess 🤮 ...)

WHAT IS AN EMOJI?

✨ Unicode (1987) to the rescue ✨

“Unicode is a standard for the consistent encoding, representation, and handling of text expressed in most of the world’s writing systems.”

— Wikipedia

<https://t.me/learningnets>

WHAT IS AN EMOJI?

RTFM: <https://unicode.org/reports/tr51/>

In Unicode, you'll find:

A,

<https://t.me/learningnets>

WHAT IS AN EMOJI?

RTFM: <https://unicode.org/reports/tr51/>

In Unicode, you'll find:

A, ま,

<https://t.me/learningnets>

WHAT IS AN EMOJI?

RTFM: <https://unicode.org/reports/tr51/>

In Unicode, you'll find:

A, ま, 𑀧,

<https://t.me/learningnets>

WHAT IS AN EMOJI?

RTFM: <https://unicode.org/reports/tr51/>

In Unicode, you'll find:

A, ま, 𑌕, ♠, ,

<https://t.me/learningnets>

WHAT IS AN EMOJI?

RTFM: <https://unicode.org/reports/tr51/>

In Unicode, you'll find:

A, ま, 𑌕, ♠, 😄,

<https://t.me/learningnets>

WHAT IS AN EMOJI?

RTFM: <https://unicode.org/reports/tr51/>

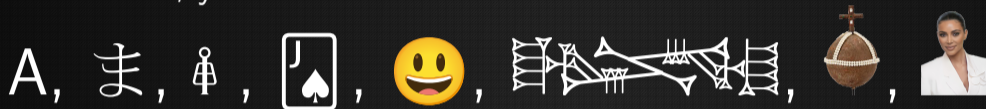
In Unicode, you'll find:

A, ま, 𐀀, ♠, 😄, 𐀀𐀁𐀂𐀃𐀄𐀅𐀆𐀇𐀈𐀉𐀊𐀋𐀌𐀍𐀎𐀏𐀐𐀑𐀒𐀓𐀔𐀕𐀖𐀗𐀘𐀙𐀚𐀛𐀜𐀝𐀞𐀟𐀠𐀡𐀢𐀣𐀤𐀥𐀦𐀧𐀨𐀩𐀪𐀫𐀬𐀭𐀮𐀯𐀰𐀱𐀲𐀳𐀴𐀵𐀶𐀷𐀸𐀹𐀺𐀻𐀼𐀽𐀾𐀿𐁀𐁁𐁂𐁃𐁄𐁅𐁆𐁇𐁈𐁉𐁊𐁋𐁌𐁍𐁎𐁏𐁐𐁑𐁒𐁓𐁔𐁕𐁖𐁗𐁘𐁙𐁚𐁛𐁜𐁝𐁞𐁟𐁠𐁡𐁢𐁣𐁤𐁥𐁦𐁧𐁨𐁩𐁪𐁫𐁬𐁭𐁮𐁯𐁰𐁱𐁲𐁳𐁴𐁵𐁶𐁷𐁸𐁹𐁺𐁻𐁼𐁽𐁾𐁿𐂀𐂁𐂂𐂃𐂄𐂅𐂆𐂇𐂈𐂉𐂊𐂋𐂌𐂍𐂎𐂏𐂐𐂑𐂒𐂓𐂔𐂕𐂖𐂗𐂘𐂙𐂚𐂛𐂜𐂝𐂞𐂟𐂠𐂡𐂢𐂣𐂤𐂥𐂦𐂧𐂨𐂩𐂪𐂫𐂬𐂭𐂮𐂯𐂰𐂱𐂲𐂳𐂴𐂵𐂶𐂷𐂸𐂹𐂺𐂻𐂼𐂽𐂾𐂿𐃀𐃁𐃂𐃃𐃄𐃅𐃆𐃇𐃈𐃉𐃊𐃋𐃌𐃍𐃎𐃏𐃐𐃑𐃒𐃓𐃔𐃕𐃖𐃗𐃘𐃙𐃚𐃛𐃜𐃝𐃞𐃟𐃠𐃡𐃢𐃣𐃤𐃥𐃦𐃧𐃨𐃩𐃪𐃫𐃬𐃭𐃮𐃯𐃰𐃱𐃲𐃳𐃴𐃵𐃶𐃷𐃸𐃹𐃺𐃻𐃼𐃽𐃾𐃿𐄀𐄁𐄂𐄃𐄄𐄅𐄆𐄇𐄈𐄉𐄊𐄋𐄌𐄍𐄎𐄏𐄐𐄑𐄒𐄓𐄔𐄕𐄖𐄗𐄘𐄙𐄚𐄛𐄜𐄝𐄞𐄟𐄠𐄡𐄢𐄣𐄤𐄥𐄦𐄧𐄨𐄩𐄪𐄫𐄬𐄭𐄮𐄯𐄰𐄱𐄲𐄳𐄴𐄵𐄶𐄷𐄸𐄹𐄺𐄻𐄼𐄽𐄾𐄿𐅀𐅁𐅂𐅃𐅄𐅅𐅆𐅇𐅈𐅉𐅊𐅋𐅌𐅍𐅎𐅏𐅐𐅑𐅒𐅓𐅔𐅕𐅖𐅗𐅘𐅙𐅚𐅛𐅜𐅝𐅞𐅟𐅠𐅡𐅢𐅣𐅤𐅥𐅦𐅧𐅨𐅩𐅪𐅫𐅬𐅭𐅮𐅯𐅰𐅱𐅲𐅳𐅴𐅵𐅶𐅷𐅸𐅹𐅺𐅻𐅼𐅽𐅾𐅿𐆀𐆁𐆂𐆃𐆄𐆅𐆆𐆇𐆈𐆉𐆊𐆋𐆌𐆍𐆎𐆏𐆐𐆑𐆒𐆓𐆔𐆕𐆖𐆗𐆘𐆙𐆚𐆛𐆜𐆝𐆞𐆟𐆠𐆡𐆢𐆣𐆤𐆥𐆦𐆧𐆨𐆩𐆪𐆫𐆬𐆭𐆮𐆯𐆰𐆱𐆲𐆳𐆴𐆵𐆶𐆷𐆸𐆹𐆺𐆻𐆼𐆽𐆾𐆿𐇀𐇁𐇂𐇃𐇄𐇅𐇆𐇇𐇈𐇉𐇊𐇋𐇌𐇍𐇎𐇏𐇐𐇑𐇒𐇓𐇔𐇕𐇖𐇗𐇘𐇙𐇚𐇛𐇜𐇝𐇞𐇟𐇠𐇡𐇢𐇣𐇤𐇥𐇦𐇧𐇨𐇩𐇪𐇫𐇬𐇭𐇮𐇯𐇰𐇱𐇲𐇳𐇴𐇵𐇶𐇷𐇸𐇹𐇺𐇻𐇼𐇽𐇾𐇿𐈀𐈁𐈂𐈃𐈄𐈅𐈆𐈇𐈈𐈉𐈊𐈋𐈌𐈍𐈎𐈏𐈐𐈑𐈒𐈓𐈔𐈕𐈖𐈗𐈘𐈙𐈚𐈛𐈜𐈝𐈞𐈟𐈠𐈡𐈢𐈣𐈤𐈥𐈦𐈧𐈨𐈩𐈪𐈫𐈬𐈭𐈮𐈯𐈰𐈱𐈲𐈳𐈴𐈵𐈶𐈷𐈸𐈹𐈺𐈻𐈼𐈽𐈾𐈿𐉀𐉁𐉂𐉃𐉄𐉅𐉆𐉇𐉈𐉉𐉊𐉋𐉌𐉍𐉎𐉏𐉐𐉑𐉒𐉓𐉔𐉕𐉖𐉗𐉘𐉙𐉚𐉛𐉜𐉝𐉞𐉟𐉠𐉡𐉢𐉣𐉤𐉥𐉦𐉧𐉨𐉩𐉪𐉫𐉬𐉭𐉮𐉯𐉰𐉱𐉲𐉳𐉴𐉵𐉶𐉷𐉸𐉹𐉺𐉻𐉼𐉽𐉾𐉿𐊀𐊁𐊂𐊃𐊄𐊅𐊆𐊇𐊈𐊉𐊊𐊋𐊌𐊍𐊎𐊏𐊐𐊑𐊒𐊓𐊔𐊕𐊖𐊗𐊘𐊙𐊚𐊛𐊜𐊝𐊞𐊟𐊠𐊡𐊢𐊣𐊤𐊥𐊦𐊧𐊨𐊩𐊪𐊫𐊬𐊭𐊮𐊯𐊰𐊱𐊲𐊳𐊴𐊵𐊶𐊷𐊸𐊹𐊺𐊻𐊼𐊽𐊾𐊿𐋀𐋁𐋂𐋃𐋄𐋅𐋆𐋇𐋈𐋉𐋊𐋋𐋌𐋍𐋎𐋏𐋐𐋑𐋒𐋓𐋔𐋕𐋖𐋗𐋘𐋙𐋚𐋛𐋜𐋝𐋞𐋟𐋠𐋡𐋢𐋣𐋤𐋥𐋦𐋧𐋨𐋩𐋪𐋫𐋬𐋭𐋮𐋯𐋰𐋱𐋲𐋳𐋴𐋵𐋶𐋷𐋸𐋹𐋺𐋻𐋼𐋽𐋾𐋿𐌀𐌁𐌂𐌃𐌄𐌅𐌆𐌇𐌈𐌉𐌊𐌋𐌌𐌍𐌎𐌏𐌐𐌑𐌒𐌓𐌔𐌕𐌖𐌗𐌘𐌙𐌚𐌛𐌜𐌝𐌞𐌟𐌠𐌡𐌢𐌣𐌤𐌥𐌦𐌧𐌨𐌩𐌪𐌫𐌬𐌭𐌮𐌯𐌰𐌱𐌲𐌳𐌴𐌵𐌶𐌷𐌸𐌹𐌺𐌻𐌼𐌽𐌾𐌿𐍀𐍁𐍂𐍃𐍄𐍅𐍆𐍇𐍈𐍉𐍊𐍋𐍌𐍍𐍎𐍏𐍐𐍑𐍒𐍓𐍔𐍕𐍖𐍗𐍘𐍙𐍚𐍛𐍜𐍝𐍞𐍟𐍠𐍡𐍢𐍣𐍤𐍥𐍦𐍧𐍨𐍩𐍪𐍫𐍬𐍭𐍮𐍯𐍰𐍱𐍲𐍳𐍴𐍵𐍶𐍷𐍸𐍹𐍺𐍻𐍼𐍽𐍾𐍿𐎀𐎁𐎂𐎃𐎄𐎅𐎆𐎇𐎈𐎉𐎊𐎋𐎌𐎍𐎎𐎏𐎐𐎑𐎒𐎓𐎔𐎕𐎖𐎗𐎘𐎙𐎚𐎛𐎜𐎝𐎞𐎟𐎠𐎡𐎢𐎣𐎤𐎥𐎦𐎧𐎨𐎩𐎪𐎫𐎬𐎭𐎮𐎯𐎰𐎱𐎲𐎳𐎴𐎵𐎶𐎷𐎸𐎹𐎺𐎻𐎼𐎽𐎾𐎿𐏀𐏁𐏂𐏃𐏄𐏅𐏆𐏇𐏈𐏉𐏊𐏋𐏌𐏍𐏎𐏏𐏐𐏑𐏒𐏓𐏔𐏕𐏖𐏗𐏘𐏙𐏚𐏛𐏜𐏝𐏞𐏟𐏠𐏡𐏢𐏣𐏤𐏥𐏦𐏧𐏨𐏩𐏪𐏫𐏬𐏭𐏮𐏯𐏰𐏱𐏲𐏳𐏴𐏵𐏶𐏷𐏸𐏹𐏺𐏻𐏼𐏽𐏾𐏿𐐀𐐁𐐂𐐃𐐄𐐅𐐆𐐇𐐈𐐉𐐊𐐋𐐌𐐍𐐎𐐏𐐐𐐑𐐒𐐓𐐔𐐕𐐖𐐗𐐘𐐙𐐚𐐛𐐜𐐝𐐞𐐟𐐠𐐡𐐢𐐣𐐤𐐥𐐦𐐧𐐨𐐩𐐪𐐫𐐬𐐭𐐮𐐯𐐰𐐱𐐲𐐳𐐴𐐵𐐶𐐷𐐸𐐹𐐺𐐻𐐼𐐽𐐾𐐿𐑀𐑁𐑂𐑃𐑄𐑅𐑆𐑇𐑈𐑉𐑊𐑋𐑌𐑍𐑎𐑏𐑐𐑑𐑒𐑓𐑔𐑕𐑖𐑗𐑘𐑙𐑚𐑛𐑜𐑝𐑞𐑟𐑠𐑡𐑢𐑣𐑤𐑥𐑦𐑧𐑨𐑩𐑪𐑫𐑬𐑭𐑮𐑯𐑰𐑱𐑲𐑳𐑴𐑵𐑶𐑷𐑸𐑹𐑺𐑻𐑼𐑽𐑾𐑿𐒀𐒁𐒂𐒃𐒄𐒅𐒆𐒇𐒈𐒉𐒊𐒋𐒌𐒍𐒎𐒏𐒐𐒑𐒒𐒓𐒔𐒕𐒖𐒗𐒘𐒙𐒚𐒛𐒜𐒝𐒞𐒟𐒠𐒡𐒢𐒣𐒤𐒥𐒦𐒧𐒨𐒩𐒪𐒫𐒬𐒭𐒮𐒯𐒰𐒱𐒲𐒳𐒴𐒵𐒶𐒷𐒸𐒹𐒺𐒻𐒼𐒽𐒾𐒿𐓀𐓁𐓂𐓃𐓄𐓅𐓆𐓇𐓈𐓉𐓊𐓋𐓌𐓍𐓎𐓏𐓐𐓑𐓒𐓓𐓔𐓕𐓖𐓗𐓘𐓙𐓚𐓛𐓜𐓝𐓞𐓟𐓠𐓡𐓢𐓣𐓤𐓥𐓦𐓧𐓨𐓩𐓪𐓫𐓬𐓭𐓮𐓯𐓰𐓱𐓲𐓳𐓴𐓵𐓶𐓷𐓸𐓹𐓺𐓻𐓼𐓽𐓾𐓿𐔀𐔁𐔂𐔃𐔄𐔅𐔆𐔇𐔈𐔉𐔊𐔋𐔌𐔍𐔎𐔏𐔐𐔑𐔒𐔓𐔔𐔕𐔖𐔗𐔘𐔙𐔚𐔛𐔜𐔝𐔞𐔟𐔠𐔡𐔢𐔣𐔤𐔥𐔦𐔧𐔨𐔩𐔪𐔫𐔬𐔭𐔮𐔯𐔰𐔱𐔲𐔳𐔴𐔵𐔶𐔷𐔸𐔹𐔺𐔻𐔼𐔽𐔾𐔿𐕀𐕁𐕂𐕃𐕄𐕅𐕆𐕇𐕈𐕉𐕊𐕋𐕌𐕍𐕎𐕏𐕐𐕑𐕒𐕓𐕔𐕕𐕖𐕗𐕘𐕙𐕚𐕛𐕜𐕝𐕞𐕟𐕠𐕡𐕢𐕣𐕤𐕥𐕦𐕧𐕨𐕩𐕪𐕫𐕬𐕭𐕮𐕯𐕰𐕱𐕲𐕳𐕴𐕵𐕶𐕷𐕸𐕹𐕺𐕻𐕼𐕽𐕾𐕿𐖀𐖁𐖂𐖃𐖄𐖅𐖆𐖇𐖈𐖉𐖊𐖋𐖌𐖍𐖎𐖏𐖐𐖑𐖒𐖓𐖔𐖕𐖖𐖗𐖘𐖙𐖚𐖛𐖜𐖝𐖞𐖟𐖠𐖡𐖢𐖣𐖤𐖥𐖦𐖧𐖨𐖩𐖪𐖫𐖬𐖭𐖮𐖯𐖰𐖱𐖲𐖳𐖴𐖵𐖶𐖷𐖸𐖹𐖺𐖻𐖼𐖽𐖾𐖿𐗀𐗁𐗂𐗃𐗄𐗅𐗆𐗇𐗈𐗉𐗊𐗋𐗌𐗍𐗎𐗏𐗐𐗑𐗒𐗓𐗔𐗕𐗖𐗗𐗘𐗙𐗚𐗛𐗜𐗝𐗞𐗟𐗠𐗡𐗢𐗣𐗤𐗥𐗦𐗧𐗨𐗩𐗪𐗫𐗬𐗭𐗮𐗯𐗰𐗱𐗲𐗳𐗴𐗵𐗶𐗷𐗸𐗹𐗺𐗻𐗼𐗽𐗾𐗿𐘀𐘁𐘂𐘃𐘄𐘅𐘆𐘇𐘈𐘉𐘊𐘋𐘌𐘍𐘎𐘏𐘐𐘑𐘒𐘓𐘔𐘕𐘖𐘗𐘘𐘙𐘚𐘛𐘜𐘝𐘞𐘟𐘠𐘡𐘢𐘣𐘤𐘥𐘦𐘧𐘨𐘩𐘪𐘫𐘬𐘭𐘮𐘯𐘰𐘱𐘲𐘳𐘴𐘵𐘶𐘷𐘸𐘹𐘺𐘻𐘼𐘽𐘾𐘿𐙀𐙁𐙂𐙃𐙄𐙅𐙆𐙇𐙈𐙉𐙊𐙋𐙌𐙍𐙎𐙏𐙐𐙑𐙒𐙓𐙔𐙕𐙖𐙗𐙘𐙙𐙚𐙛𐙜𐙝𐙞𐙟𐙠𐙡𐙢𐙣𐙤𐙥𐙦𐙧𐙨𐙩𐙪𐙫𐙬𐙭𐙮𐙯𐙰𐙱𐙲𐙳𐙴𐙵𐙶𐙷𐙸𐙹𐙺𐙻𐙼𐙽𐙾𐙿𐚀𐚁𐚂𐚃𐚄𐚅𐚆𐚇𐚈𐚉𐚊𐚋𐚌𐚍𐚎𐚏𐚐𐚑𐚒𐚓𐚔𐚕𐚖𐚗𐚘𐚙𐚚𐚛𐚜𐚝𐚞𐚟𐚠𐚡𐚢𐚣𐚤𐚥𐚦𐚧𐚨𐚩𐚪𐚫𐚬𐚭𐚮𐚯𐚰𐚱𐚲𐚳𐚴𐚵𐚶𐚷𐚸𐚹𐚺𐚻𐚼𐚽𐚾𐚿𐛀𐛁𐛂𐛃𐛄𐛅𐛆𐛇𐛈𐛉𐛊𐛋𐛌𐛍𐛎𐛏𐛐𐛑𐛒𐛓𐛔𐛕𐛖𐛗𐛘𐛙𐛚𐛛𐛜𐛝𐛞𐛟𐛠𐛡𐛢𐛣𐛤𐛥𐛦𐛧𐛨𐛩𐛪𐛫𐛬𐛭𐛮𐛯𐛰𐛱𐛲𐛳𐛴𐛵𐛶𐛷𐛸𐛹𐛺𐛻𐛼𐛽𐛾𐛿𐜀𐜁𐜂𐜃𐜄𐜅𐜆𐜇𐜈𐜉𐜊𐜋𐜌𐜍𐜎𐜏𐜐𐜑𐜒𐜓𐜔𐜕𐜖𐜗𐜘𐜙𐜚𐜛𐜜𐜝𐜞𐜟𐜠𐜡𐜢𐜣𐜤𐜥𐜦𐜧𐜨𐜩𐜪𐜫𐜬𐜭𐜮𐜯𐜰𐜱𐜲𐜳𐜴𐜵𐜶𐜷𐜸𐜹𐜺𐜻𐜼𐜽𐜾𐜿𐝀𐝁𐝂𐝃𐝄𐝅𐝆𐝇𐝈𐝉𐝊𐝋𐝌𐝍𐝎𐝏𐝐𐝑𐝒𐝓𐝔𐝕𐝖𐝗𐝘𐝙𐝚𐝛𐝜𐝝𐝞𐝟𐝠𐝡𐝢𐝣𐝤𐝥𐝦𐝧𐝨𐝩𐝪𐝫𐝬𐝭𐝮𐝯𐝰𐝱𐝲𐝳𐝴𐝵𐝶𐝷𐝸𐝹𐝺𐝻𐝼𐝽𐝾𐝿𐞀𐞁𐞂𐞃𐞄𐞅𐞆𐞇𐞈𐞉𐞊𐞋𐞌𐞍𐞎𐞏𐞐𐞑𐞒𐞓𐞔𐞕𐞖𐞗𐞘𐞙𐞚𐞛𐞜𐞝𐞞𐞟𐞠𐞡𐞢𐞣𐞤𐞥𐞦𐞧𐞨𐞩𐞪𐞫𐞬𐞭𐞮𐞯𐞰𐞱𐞲𐞳𐞴𐞵𐞶𐞷𐞸𐞹𐞺𐞻𐞼𐞽𐞾𐞿𐟀𐟁𐟂𐟃𐟄𐟅𐟆𐟇𐟈𐟉𐟊𐟋𐟌𐟍𐟎𐟏𐟐𐟑𐟒𐟓𐟔𐟕𐟖𐟗𐟘𐟙𐟚𐟛𐟜𐟝𐟞𐟟𐟠𐟡𐟢𐟣𐟤𐟥𐟦𐟧𐟨𐟩𐟪𐟫𐟬𐟭𐟮𐟯𐟰𐟱𐟲𐟳𐟴𐟵𐟶𐟷𐟸𐟹𐟺𐟻𐟼𐟽𐟾𐟿𐠀𐠁𐠂𐠃𐠄𐠅𐠆𐠇𐠈𐠉𐠊𐠋𐠌𐠍𐠎𐠏𐠐𐠑𐠒𐠓𐠔𐠕𐠖𐠗𐠘𐠙𐠚𐠛𐠜𐠝𐠞𐠟𐠠𐠡𐠢𐠣𐠤𐠥𐠦𐠧𐠨𐠩𐠪𐠫𐠬𐠭𐠮𐠯𐠰𐠱𐠲𐠳𐠴𐠵𐠶𐠷𐠸𐠹𐠺𐠻𐠼𐠽𐠾𐠿𐡀𐡁𐡂𐡃𐡄𐡅𐡆𐡇𐡈𐡉𐡊𐡋𐡌𐡍𐡎𐡏𐡐𐡑𐡒𐡓𐡔𐡕𐡖𐡗𐡘𐡙𐡚𐡛𐡜𐡝𐡞𐡟𐡠𐡡𐡢𐡣𐡤𐡥𐡦𐡧𐡨𐡩𐡪𐡫𐡬𐡭𐡮𐡯𐡰𐡱𐡲𐡳𐡴𐡵𐡶𐡷𐡸𐡹𐡺𐡻𐡼𐡽𐡾𐡿𐢀𐢁𐢂𐢃𐢄𐢅𐢆𐢇𐢈𐢉𐢊𐢋𐢌𐢍𐢎𐢏𐢐𐢑𐢒𐢓𐢔𐢕𐢖𐢗𐢘𐢙𐢚𐢛𐢜𐢝𐢞𐢟𐢠𐢡𐢢𐢣𐢤𐢥𐢦𐢧𐢨𐢩𐢪𐢫𐢬𐢭𐢮𐢯𐢰𐢱𐢲𐢳𐢴𐢵𐢶𐢷𐢸𐢹𐢺𐢻𐢼𐢽𐢾𐢿𐣀𐣁𐣂𐣃𐣄𐣅𐣆𐣇𐣈𐣉𐣊𐣋𐣌𐣍𐣎𐣏𐣐𐣑𐣒𐣓𐣔𐣕𐣖𐣗𐣘𐣙𐣚𐣛𐣜𐣝𐣞𐣟𐣠𐣡𐣢𐣣𐣤𐣥𐣦𐣧𐣨𐣩𐣪𐣫𐣬𐣭𐣮𐣯𐣰𐣱𐣲𐣳𐣴𐣵𐣶𐣷𐣸𐣹𐣺𐣻𐣼𐣽𐣾𐣿𐤀𐤁𐤂𐤃𐤄𐤅𐤆𐤇𐤈𐤉𐤊𐤋𐤌𐤍𐤎𐤏𐤐𐤑𐤒𐤓𐤔𐤕𐤖𐤗𐤘𐤙𐤚𐤛𐤜𐤝𐤞𐤟𐤠𐤡𐤢𐤣𐤤𐤥𐤦𐤧𐤨𐤩𐤪𐤫𐤬𐤭𐤮𐤯𐤰𐤱𐤲𐤳𐤴𐤵𐤶𐤷𐤸𐤹𐤺𐤻𐤼𐤽𐤾𐤿𐥀𐥁𐥂𐥃𐥄𐥅𐥆𐥇𐥈𐥉𐥊𐥋𐥌𐥍𐥎𐥏𐥐𐥑𐥒𐥓𐥔𐥕𐥖𐥗𐥘𐥙𐥚𐥛𐥜𐥝𐥞𐥟𐥠𐥡𐥢𐥣𐥤𐥥𐥦𐥧𐥨𐥩𐥪𐥫𐥬𐥭𐥮𐥯𐥰𐥱𐥲𐥳𐥴𐥵𐥶𐥷𐥸𐥹𐥺𐥻𐥼𐥽𐥾𐥿𐦀𐦁𐦂𐦃𐦄𐦅𐦆𐦇𐦈𐦉𐦊𐦋𐦌𐦍𐦎𐦏𐦐𐦑𐦒𐦓𐦔𐦕𐦖𐦗𐦘𐦙𐦚𐦛𐦜𐦝𐦞𐦟𐦠𐦡𐦢𐦣𐦤𐦥𐦦𐦧𐦨𐦩𐦪𐦫𐦬𐦭𐦮𐦯𐦰𐦱𐦲𐦳𐦴𐦵𐦶𐦷𐦸𐦹𐦺𐦻𐦼𐦽𐦾𐦿𐧀𐧁𐧂𐧃𐧄𐧅𐧆𐧇𐧈𐧉𐧊𐧋𐧌𐧍𐧎𐧏𐧐𐧑𐧒𐧓𐧔𐧕𐧖𐧗𐧘𐧙𐧚𐧛𐧜𐧝𐧞𐧟𐧠𐧡𐧢𐧣𐧤𐧥𐧦𐧧𐧨𐧩𐧪𐧫𐧬𐧭𐧮𐧯𐧰𐧱𐧲𐧳𐧴𐧵𐧶𐧷𐧸𐧹𐧺𐧻𐧼𐧽𐧾𐧿𐨀𐨁𐨂𐨃𐨄𐨅𐨆𐨇𐨈𐨉𐨊𐨋𐨌𐨍𐨎𐨏𐨐𐨑𐨒𐨓𐨔𐨕𐨖𐨗𐨘𐨙𐨚𐨛𐨜𐨝𐨞𐨟𐨠𐨡𐨢𐨣𐨤𐨥𐨦𐨧𐨨𐨩𐨪𐨫𐨬𐨭𐨮𐨯𐨰𐨱𐨲𐨳𐨴𐨵𐨶𐨷𐨹𐨺𐨸𐨻𐨼𐨽𐨾𐨿𐩀𐩁𐩂𐩃𐩄𐩅𐩆𐩇𐩈𐩉𐩊𐩋𐩌𐩍𐩎𐩏𐩐𐩑𐩒𐩓𐩔𐩕𐩖𐩗𐩘𐩙𐩚𐩛𐩜𐩝𐩞𐩟𐩠𐩡𐩢𐩣𐩤𐩥𐩦𐩧𐩨𐩩𐩪𐩫𐩬𐩭𐩮𐩯𐩰𐩱𐩲𐩳𐩴𐩵𐩶𐩷𐩸𐩹𐩺𐩻𐩼𐩽𐩾𐩿𐪀𐪁𐪂𐪃𐪄𐪅𐪆𐪇𐪈𐪉𐪊𐪋𐪌𐪍𐪎𐪏𐪐𐪑𐪒𐪓𐪔𐪕𐪖𐪗𐪘𐪙𐪚𐪛𐪜𐪝𐪞𐪟𐪠𐪡𐪢𐪣𐪤𐪥𐪦𐪧𐪨𐪩𐪪𐪫𐪬𐪭𐪮𐪯𐪰𐪱𐪲𐪳𐪴𐪵𐪶𐪷𐪸𐪹𐪺𐪻𐪼𐪽𐪾𐪿𐫀𐫁𐫂𐫃𐫄𐫅𐫆𐫇𐫈𐫉𐫊𐫋𐫌𐫍𐫎𐫏𐫐𐫑𐫒𐫓𐫔𐫕𐫖𐫗𐫘𐫙𐫚𐫛𐫜𐫝𐫞𐫟𐫠𐫡𐫢𐫣𐫤𐫦𐫥𐫧𐫨𐫩𐫪𐫫𐫬𐫭𐫮𐫯𐫰𐫱𐫲𐫳𐫴𐫵𐫶𐫷𐫸𐫹𐫺𐫻𐫼𐫽𐫾𐫿𐬀𐬁𐬂𐬃𐬄𐬅𐬆𐬇𐬈𐬉𐬊𐬋𐬌𐬍𐬎𐬏𐬐𐬑𐬒𐬓𐬔𐬕𐬖𐬗𐬘𐬙𐬚𐬛𐬜𐬝𐬞𐬟𐬠𐬡𐬢𐬣𐬤𐬥𐬦𐬧𐬨𐬩𐬪𐬫𐬬𐬭𐬮𐬯𐬰𐬱𐬲𐬳𐬴𐬵𐬶𐬷𐬸𐬹𐬺𐬻𐬼𐬽𐬾𐬿𐭀𐭁𐭂𐭃𐭄𐭅𐭆𐭇𐭈𐭉𐭊𐭋𐭌𐭍𐭎𐭏𐭐𐭑𐭒𐭓𐭔𐭕𐭖𐭗𐭘𐭙𐭚𐭛𐭜𐭝𐭞𐭟𐭠𐭡𐭢𐭣𐭤𐭥𐭦𐭧𐭨𐭩𐭪𐭫𐭬𐭭𐭮𐭯𐭰𐭱𐭲𐭳𐭴𐭵𐭶𐭷𐭸𐭹𐭺𐭻𐭼𐭽𐭾𐭿𐮀𐮁𐮂𐮃𐮄𐮅𐮆𐮇𐮈𐮉𐮊𐮋𐮌𐮍𐮎𐮏𐮐𐮑𐮒𐮓𐮔𐮕𐮖𐮗𐮘𐮙𐮚𐮛𐮜𐮝𐮞𐮟𐮠𐮡𐮢𐮣𐮤𐮥𐮦𐮧𐮨𐮩𐮪𐮫𐮬𐮭𐮮𐮯𐮰𐮱𐮲𐮳𐮴𐮵𐮶𐮷𐮸𐮹𐮺𐮻𐮼𐮽𐮾𐮿𐯀𐯁𐯂𐯃𐯄𐯅𐯆𐯇𐯈𐯉𐯊𐯋𐯌𐯍𐯎𐯏𐯐𐯑𐯒𐯓𐯔𐯕𐯖𐯗𐯘𐯙𐯚𐯛𐯜𐯝𐯞𐯟𐯠𐯡𐯢𐯣𐯤𐯥𐯦𐯧𐯨

WHAT IS AN EMOJI?

RTFM: <https://unicode.org/reports/tr51/>

In Unicode, you'll find:



<https://t.me/learningnets>

WHAT IS AN EMOJI?

RTFM: <https://unicode.org/reports/tr51/>

In Unicode, you'll find:

A, ま, 𑌕, ♠, 😄, 𐄎, ~~🚢, 🧑~~

<https://t.me/learningnets>

WHAT IS AN EMOJI?

RTFM: <https://unicode.org/reports/tr51/>

In Unicode, you'll find:

A, ま, 𐀀, 🎉, 😄, 𐀀𐀁𐀂𐀃𐀄𐀅𐀆𐀇𐀈𐀉𐀊𐀋𐀌𐀍𐀎𐀏𐀐𐀑𐀒𐀓𐀔𐀕𐀖𐀗𐀘𐀙𐀚𐀛𐀜𐀝𐀞𐀟𐀠𐀡𐀢𐀣𐀤𐀥𐀦𐀧𐀨𐀩𐀪𐀫𐀬𐀭𐀮𐀯𐀰𐀱𐀲𐀳𐀴𐀵𐀶𐀷𐀸𐀹𐀺𐀻𐀼𐀽𐀾𐀿, ~~🚢, 🧑~~

<https://t.me/learningnets>

WHAT IS AN EMOJI?

RTFM: <https://unicode.org/reports/tr51/>

In Unicode, you'll find:

A, ま, 𐀀, , , , ~~ , ~~

<https://t.me/learningnets>

WHAT IS AN EMOJI?

RTFM: <https://unicode.org/reports/tr51/>

In Unicode, you'll find:

A, 𐍚, 𐌆, 🃏, 😄, 𐌵𐌹𐌺𐌻𐌰𐌿𐌽𐌾𐌰𐌽𐌾𐌰𐌽𐌾𐌰, ~~🚢, 🧑~~

Definition:

If Unicode says it is a *qualified Emoji*, then it **is** an Emoji!

<https://t.me/learningnets>

WHAT IS AN EMOJI?

RTFM: <https://unicode.org/reports/tr51/>

In Unicode, you'll find:

A, 𐍚, 𐌆, 🃏, 😄, 𐌵𐌹𐌿𐌺𐌽𐌾𐌰𐌽𐌾𐌰𐌽𐌾𐌰, ~~🏠, 🧑~~

Definition:

If Unicode says it is a *qualified Emoji*, then it **is** an Emoji!

In UTF-8, emojis at least 3 bytes 😄, at most 35 🧑🧑.

<https://t.me/learningnets>

WHAT IS AN EMOJI?

RTFM: <https://unicode.org/reports/tr51/>

In Unicode, you'll find:

A, 𐍚, 𐌆, 🃏, 😄, 𐌵𐌹𐌸𐌰𐌶𐌿𐌺𐌽𐌾𐌰𐌽𐌰𐌴𐌹𐌺𐌰, ~~🏠, 🧑~~

Definition:

If Unicode says it is a *qualified Emoji*, then it **is** an Emoji!

In UTF-8, emojis at least 3 bytes 😄, at most 35 🧑🧑.

And they add new emojis every year! Currently at Unicode version 14.

<https://t.me/learningnets>

EXCUSE ME, DO YOU HAVE 2 MINUTES TO TALK ABOUT



<https://t.me/learningnets>

EXCUSE ME, DO YOU HAVE 2 MINUTES TO TALK ABOUT



Features:

- Architecture of the future
- Simple, clean RISC Instruction Set Architecture
- Open Source ISA & Open Hardware*
- 2 and 4 byte instructions*, little-endian

<https://t.me/learningnets>

EXCUSE ME, DO YOU HAVE 2 MINUTES TO TALK ABOUT



Features:

- Architecture of the future present
- Simple, clean RISC Instruction Set Architecture
- Open Source ISA & Open Hardware*
- 2 and 4 byte instructions*, little-endian

<https://t.me/learningnets>

EXCUSE ME, DO YOU HAVE 2 MINUTES TO TALK ABOUT



Features:


- Architecture of the future present
- Simple, clean RISC Instruction Set Architecture
- Open Source ISA & Open Hardware*
- 2 and 4 byte instructions*, little-endian

*:Conditions may apply <https://t.me/learningnets>

HOW CAN I EXECUTE MY EMOJI?

<https://t.me/learningnets>

HOW CAN I EXECUTE MY EMOJI?

-  Alphanumeric x86: 'A' : `inc %eax`
0x41

<https://t.me/learningnets>

HOW CAN I EXECUTE MY EMOJI?

- ✓ Alphanumeric x86: 'A' : `inc %eax`
0x41
- ✓ Alphanumeric ARM&RISCV: '4A0s' : `csrc mip,sp`
0x34413073

<https://t.me/learningnets>

HOW CAN I EXECUTE MY EMOJI?

- ✓ Alphanumeric x86: 'A' : `inc %eax`
0x41
- ✓ Alphanumeric ARM&RISCV: '4A0s' : `csrc mip,sp`
0x34413073
- ? Emoji RISCV:

<https://t.me/learningnets>

HOW CAN I EXECUTE MY EMOJI?

- ✓ Alphanumeric x86: 'A' : `inc %eax`
0x41
- ✓ Alphanumeric ARM&RISCV: '4A0s' : `csrc mip,sp`
0x34413073
- ✗ Emoji RISCV: ? : only 10 instructions
0xE29D93





<https://t.me/learningnets>

HOW CAN I EXECUTE MY EMOJI?

- ✓ Alphanumeric x86: 'A' : `inc %eax`
0x41
- ✓ Alphanumeric ARM&RISCV: '4A0s' : `csrc mip,sp`
0x34413073
- ✗ Emoji RISCV: ? ? : still very few instructions
0xE29D93E29D93

<https://t.me/learningnets>

HOW CAN I EXECUTE MY EMOJI?

-  Alphanumeric x86: 'A' : `inc %eax`
0x41
-  Alphanumeric ARM&RISCV: '4A0s' : `csrc mip,sp`
0x34413073
-  Emoji RISCV: `???` ... : does not get better 
0xE29D93E29D93E29D93...

<https://t.me/learningnets>

HOW CAN I EXECUTE MY EMOJI?

- ✓ Alphanumeric x86: 'A' : `inc %eax`
0x41
- ✓ Alphanumeric ARM&RISCV: '4A0s' : `csrc mip,sp`
0x34413073
- ✗ Emoji RISCV: ? ? ? ... : does not get better 😭
0xE29D93E29D93E29D93...

We are stuck!

<https://t.me/learningnets>

THE INTUITION...

```
auipc ra,0x979ff
```

<https://t.me/learningnets>

THE INTUITION...

97 F0 9F 97

auipc ra,0x979ff

<https://t.me/learningnets>

THE INTUITION...

97 F0 9F 97

└──────────┘
auipc ra,0x979ff

<https://t.me/learningnets>

THE INTUITION...



F0 9F 86

97

F0 9F 97



`auipc ra,0x979ff`

<https://t.me/learningnets>

THE INTUITION...

OK

<none>

F0 9F 86

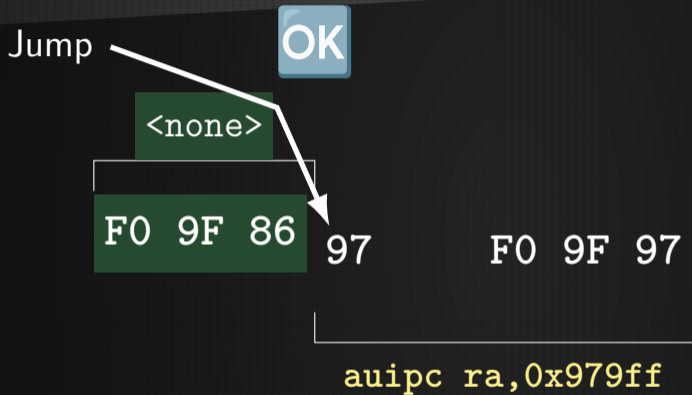
97

F0 9F 97

auipc ra,0x979ff

<https://t.me/learningnets>

THE INTUITION...



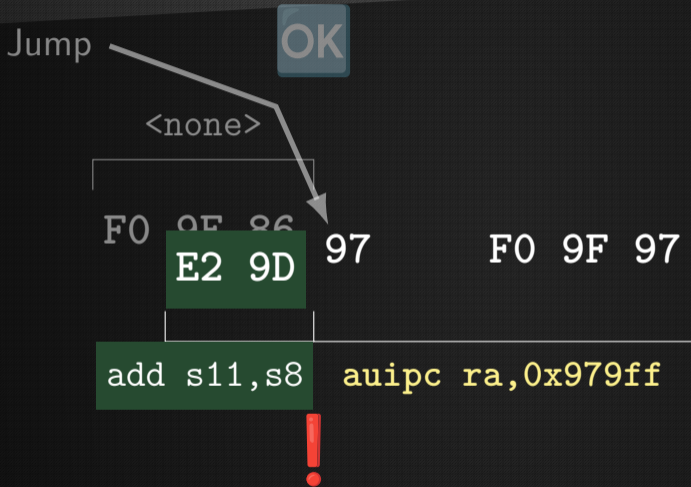
<https://t.me/learningnets>

THE INTUITION...



<https://t.me/learningnets>

THE INTUITION...



<https://t.me/learningnets>

THE INTUITION...

Jump

OK

<none>

F0 9F 86 97
E2 9D

F0 9F 97

add s11,s8 **auipc ra,0x979ff**



<https://t.me/learningnets>

THE INTUITION...

Jump

OK

<none>

F0 9F 86 97
E2 9D

F0 9F 97

93 EF B8 8F

add s11,s8 auipc ra,0x979ff



<https://t.me/learningnets>

THE INTUITION...

Jump

OK

<none>

F0 9F 86 97
E2 9D

F0 9F 97

93 EF B8 8F

add s11,s8

auipc ra,0x979ff

ori t6,a7,-1797



<https://t.me/learningnets>

THE INTUITION...

Jump

OK



<none>

F0 9F 86 97
E2 9D

F0 9F 97

91 EF B8 8F

add s11,s8 auipc ra,0x979ff ori t6,a7,-1797



<https://t.me/learningnets>

THE INTUITION...

Jump

OK



<none>

bnez a5,+28

F0 9F 86 97
E2 9D

F0 9F 97

91 EF B8 8F

add s11,s8

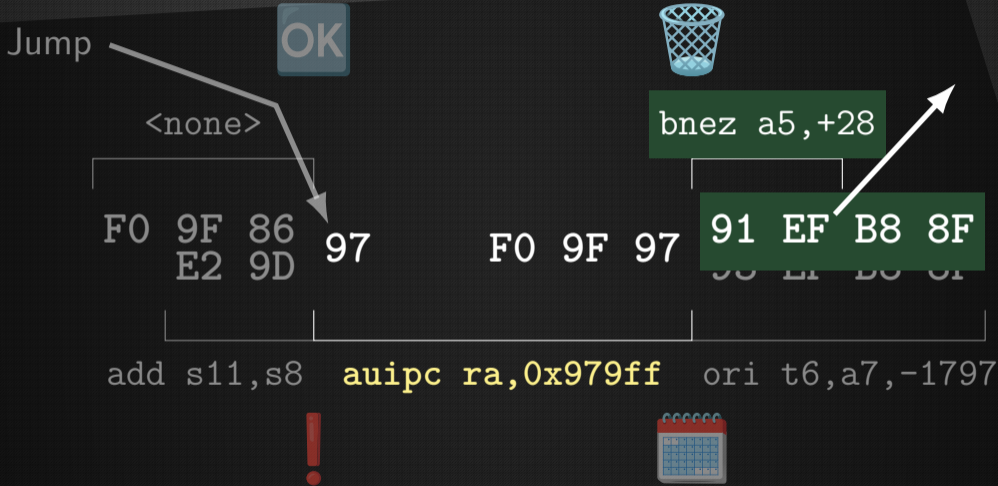
auipc ra,0x979ff

ori t6,a7,-1797



<https://t.me/learningnets>

THE INTUITION...



<https://t.me/learningnets>

THE INTUITION...

Jump

OK



<none>

bnez a5,+28

F0 9F 86 97
E2 9D

F0 9F 97

91 EF B8 8F
93 EF B8 8F

add s11,s8

auipc ra,0x979ff

ori t6,a7,-1797



<https://t.me/learningnets>

THE INTUITION...



How do I generate all gadgets?

<https://t.me/learningnets>

EMOJI SHELLCODING IS A CODE-REUSE PROBLEM

Source: Trust me, bro.

- Return-Oriented Programming (Shacham et al., 2007):
 - 1 Scan target binary for reusable code snippets (*gadgets*)
 - 2 Build your shellcode with gadgets
 - 3 ???
 - 4 Profit

<https://t.me/learningnets>

EMOJI SHELLCODING IS A CODE-REUSE PROBLEM

Source: Trust me, bro.

■ Return-Oriented Programming (Shacham et al., 2007):

- 1 Scan target binary for reusable code snippets (*gadgets*)
- 2 Build your shellcode with gadgets
- 3 ???
- 4 Profit

■ JIT spraying (Blazakis, 2010):

create your own gadgets by
controlling JITted bytecode.

```
var y = (  
    0x3c54d0d9 ^  
    0x3c909058 ^  
    0x3c59f46a ^
```

is turned into:

```
B8 D9D0543C    MOV EAX,3C54D0D9  
35 5890903C    XOR EAX,3C909058  
35 6AF4593C    XOR EAX,3C59F46A  
...
```

<https://t.me/learningnets>

EMOJI SHELLCODING IS A CODE-REUSE PROBLEM

Source: Trust me, bro.

Return-Oriented Programming (Shacham et al., 2007):

- 1 Scan target binary for reusable code snippets (*gadgets*)
- 2 Build your shellcode with gadgets
- 3 ???
- 4 Profit

JIT spraying (Blazakis, 2010):

create your own gadgets by controlling JITted bytecode.

Easy when you control most of the output...

```
var y = (  
  0x3c54d0d9 ^  
  0x3c909058 ^  
  0x3c59f46a ^  
  ...  
is turned into:  
B8 D9D0543C MOV EAX, 3C54D0D9  
35 5890903C XOR EAX, 3C909058  
35 6AF4593C XOR EAX, 3C59F46A  
...
```

<https://t.me/learningnets>

EMOJI SHELLCODING IS A CODE-REUSE PROBLEM

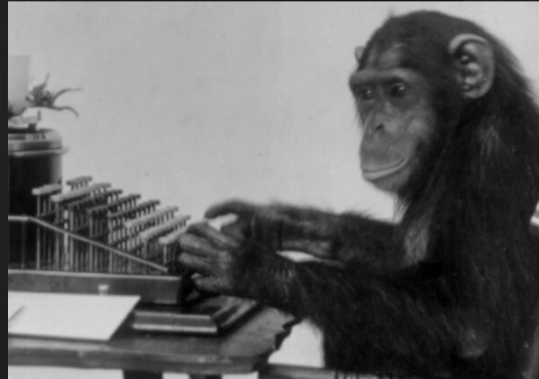
Great! How do I create gadgets from Emoji stream?

<https://t.me/learningnets>

EMOJI SHELLCODING IS A CODE-REUSE PROBLEM

Great! How do I create gadgets from Emoji stream?

Remember infinite monkey theorem?



<https://t.me/learningnets>

EMOJI SHELLCODING IS A CODE-REUSE PROBLEM

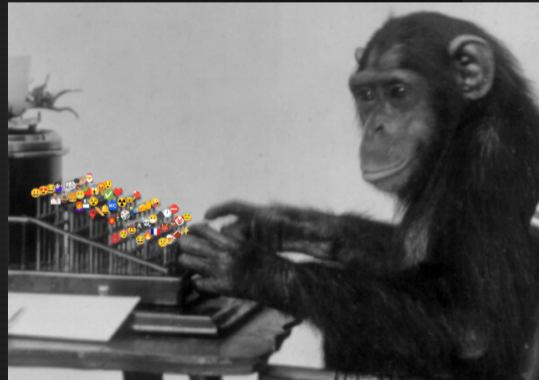
Great! How do I create gadgets from Emoji stream?

Remember infinite monkey theorem?

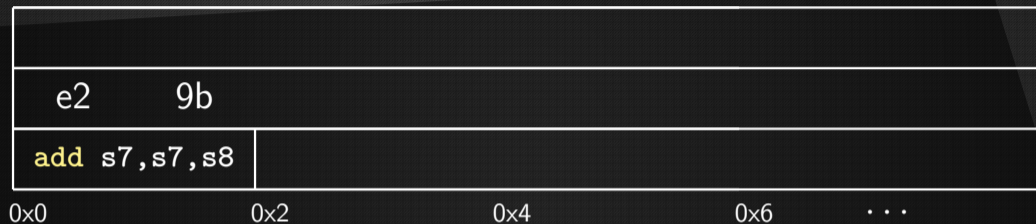
They get emoji keyboards now!

⇒ Need to invent a new algorithm.

<https://t.me/learningnets>



EMOJI GADGET BUILDING



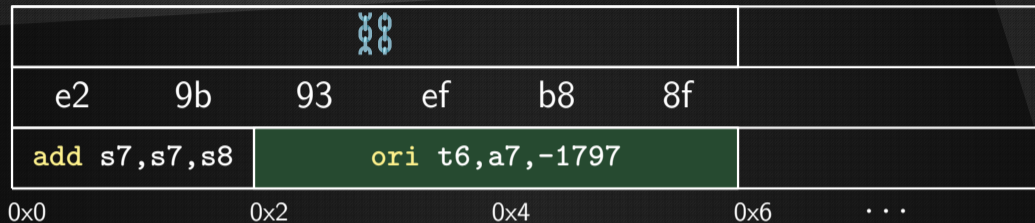
<https://t.me/learningnets>

EMOJI GADGET BUILDING (1ST CASE)



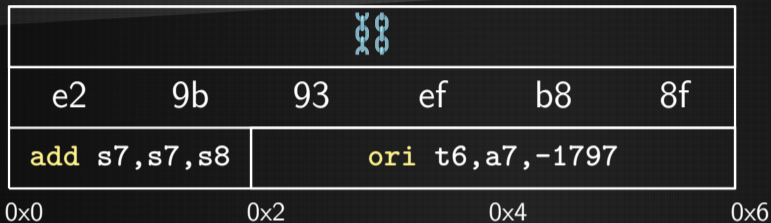
<https://t.me/learningnets>

EMOJI GADGET BUILDING (1ST CASE)



<https://t.me/learningnets>

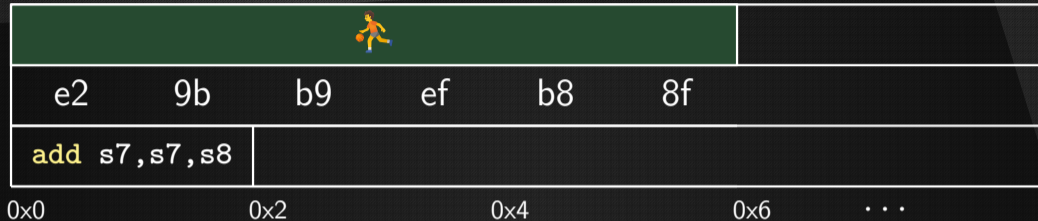
EMOJI GADGET BUILDING (1ST CASE)



```
1 add s7,s7,s8; ori t6,a7,-1797
```

<https://t.me/learningnets>

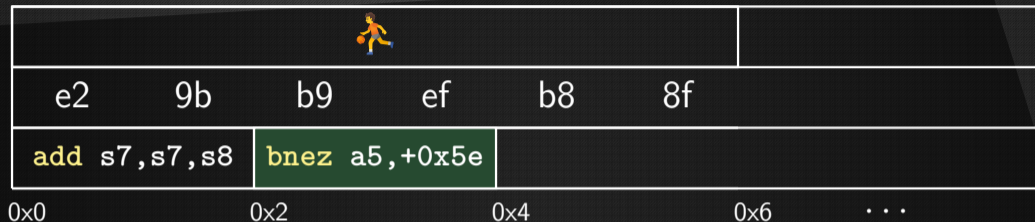
EMOJI GADGET BUILDING (2ND CASE)



```
1 add s7,s7,s8; ori t6,a7,-1797
```

<https://t.me/learningnets>

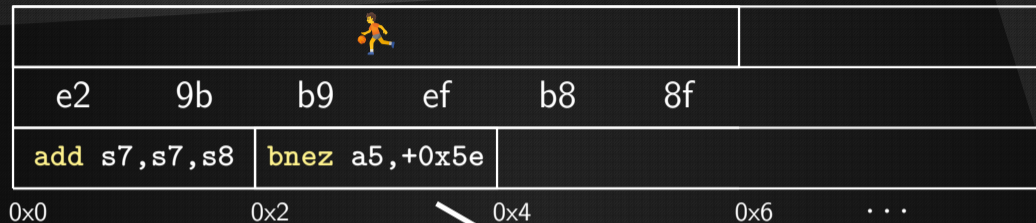
EMOJI GADGET BUILDING (2ND CASE)



```
1 add s7,s7,s8; ori t6,a7,-1797
```

<https://t.me/learningnets>

EMOJI GADGET BUILDING (2ND CASE)

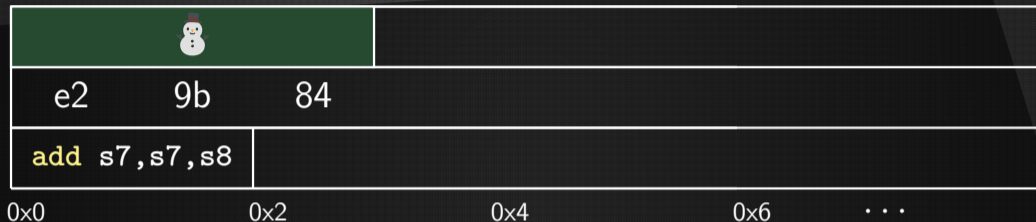


```
1 add s7,s7,s8; ori t6,a7,-1797
```

```
2 add s7,s7,s8; bnez a5,+0x5e; .byte 0xb8,0x8f
```

<https://t.me/learningnets>

EMOJI GADGET BUILDING (3RD CASE)

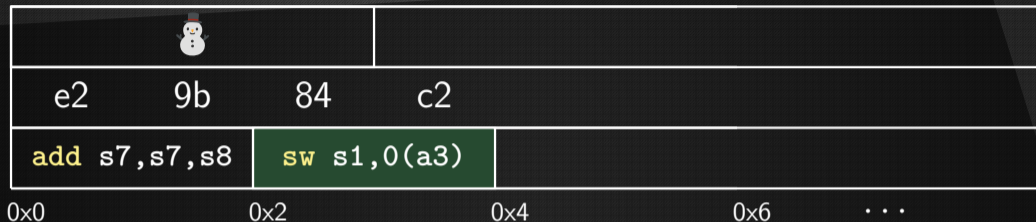


```
1 add s7,s7,s8; ori t6,a7,-1797
```

```
2 add s7,s7,s8; bnez a5,+0x5e; .byte 0xb8,0x8f
```

<https://t.me/learningnets>

EMOJI GADGET BUILDING (3RD CASE)

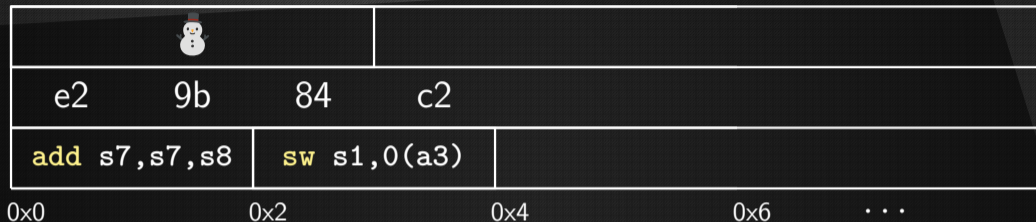


```
1 add s7,s7,s8; ori t6,a7,-1797
```

```
2 add s7,s7,s8; bnez a5,+0x5e; .byte 0xb8,0x8f
```

<https://t.me/learningnets>

EMOJI GADGET BUILDING (3RD CASE)

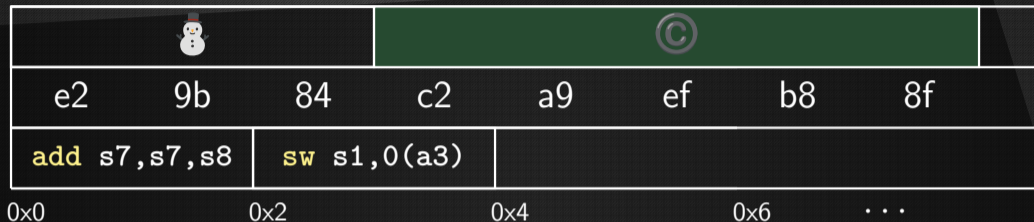


```
1 add s7,s7,s8; ori t6,a7,-1797
```

```
2 add s7,s7,s8; bnez a5,+0x5e; .byte 0xb8,0x8f
```

<https://t.me/learningnets>

EMOJI GADGET BUILDING (3RD CASE)

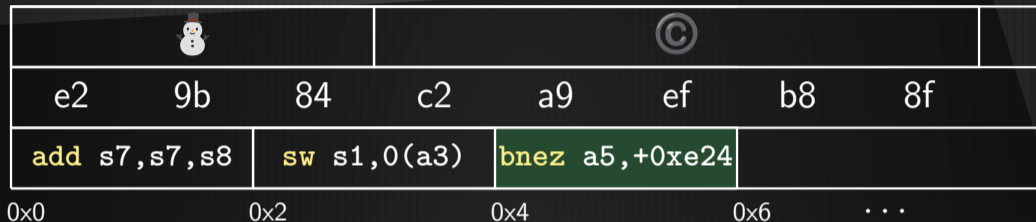


```
1 add s7,s7,s8; ori t6,a7,-1797
```

```
2 add s7,s7,s8; bnez a5,+0x5e; .byte 0xb8,0x8f
```

<https://t.me/learningnets>

EMOJI GADGET BUILDING (3RD CASE)

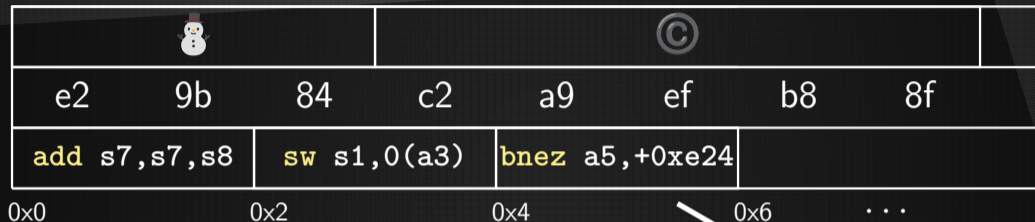


```
1 add s7,s7,s8; ori t6,a7,-1797
```

```
2 add s7,s7,s8; bnez a5,+0x5e; .byte 0xb8,0x8f
```

<https://t.me/learningnets>

EMOJI GADGET BUILDING (3RD CASE)



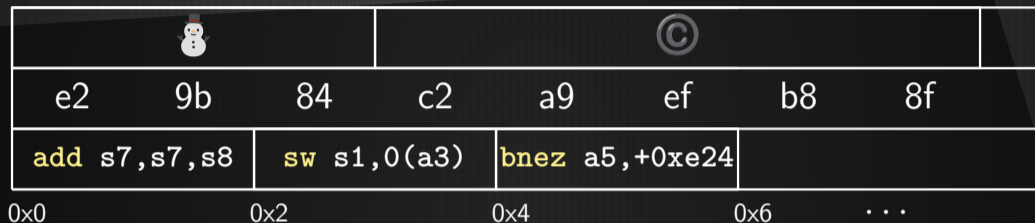
```
1 add s7,s7,s8; ori t6,a7,-1797
```

```
2 add s7,s7,s8; bnez a5,+0x5e; .byte 0xb8,0x8f
```

```
3 add s7,s7,s8; sw s1,0(a3); bnez a5,+0xe24; .byte 0xb8,0x8f
```

<https://t.me/learningnets>

EMOJI GADGET BUILDING



1 `add s7,s7,s8; ori t6,a7,-1797`

2 `add s7,s7,s8; bnez a5,+0x5e; .byte 0xb8,0x8f`

3 `add s7,s7,s8; sw s1,0(a3); bnez a5,+0xe24; .byte 0xb8,0x8f`

4 (and so on...)

<https://t.me/learningnets>

THIS METHOD IS TOTALLY GENERIC

Algorithm follows derivations in the form $S \rightarrow a T$

This exactly describes a right-linear grammar!

“Any regular expression* can be converted to a right-linear grammar.”

*alphanumeric, email address, url, emojis, ...

— A.R. Patel, 1971

(adapting the tool is left to the reader as exercise)

<https://t.me/learningnets>

Anyway, back to emoji.

We have gadgets, but we still need to chain them!

<https://t.me/learningnets>



3. WRITING OUR EMOJI CHAIN



<https://t.me/learningnets>

POKÉDEX OF RISC-V EMOJI GADGETS

- >4000 instructions available in total
- Logic `and`, `add`, `sub`, ...
- Conditional and unconditional branches (forward & backward)
- Many registers: `a1357`, `s1359`, `t0246`, `gp`, `ra` (but no `sp` 😬)
- Very few immediates
- Lots of floating-point instructions

<https://t.me/learningnets>

POKÉDEX OF RISC-V EMOJI GADGETS

- >4000 instructions available in total
- Logic `and`, `add`, `sub`, ...
- Conditional and unconditional branches (forward & backward)
- ... only one 'CSR' instruction: `csrrs ra,0xbf8,gp` 🧑
- Many registers: `a1357`, `s1359`, `t0246`, `gp`, `ra` (but no `sp` 😬)
- Very few immediates
- Lots of floating-point instructions

<https://t.me/learningnets>

POKÉDEX OF RISC-V EMOJI GADGETS

- >4000 instructions available in total
- Logic `and`, `add`, `sub`, ...
- Conditional and unconditional branches (forward & backward)
- ... only one 'CSR' instruction: `csrrs ra,0xbf8,gp` 🧑
- Many registers: `a1357`, `s1359`, `t0246`, `gp`, `ra` (but no `sp` 😬)
- Very few immediates
- Lots of floating-point instructions

A tiny bit of everything... looks like a yard sale!

<https://t.me/learningnets>

FROM BASIC BLOCKS TO ARBITRARY CODE EXECUTION

Stage 1

init

Unpacker U
(embedded \mathcal{P}_{enc})

<https://t.me/learningnets>

FROM BASIC BLOCKS TO ARBITRARY CODE EXECUTION

Stage 1

init

Unpacker \mathcal{U}
(embedded \mathcal{P}_{enc})

Payload
(unpacked by \mathcal{U})

<https://t.me/learningnets>

FROM BASIC BLOCKS TO ARBITRARY CODE EXECUTION

Stage 1

init

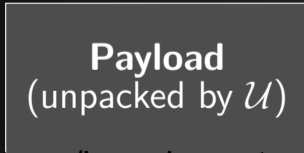
Unpacker \mathcal{U}
(embedded \mathcal{P}_{enc})

Payload
(unpacked by \mathcal{U})

<https://t.me/learningnets>

FROM BASIC BLOCKS TO ARBITRARY CODE EXECUTION

Stage 1



Gadgets:

`a1++`

`*(byte*)a3 = a1`

`a3++`

<https://t.me/learningnets>

ARBITRARY PAYLOAD ENCODING

Initial

Payload:

0x03

0x20

0x10

<https://t.me/learningnets>

ARBITRARY PAYLOAD ENCODING

Initial

Payload: **Encoded \mathcal{P}_{enc} :**

0x03

0x20

0x10

<https://t.me/learningnets>

ARBITRARY PAYLOAD ENCODING

Initial

Payload: **Encoded \mathcal{P}_{enc} :**

0x03

a1++; a1++; a1++;

0x20

0x10

<https://t.me/learningnets>

ARBITRARY PAYLOAD ENCODING

Initial

Payload: **Encoded \mathcal{P}_{enc} :**

0x03

a1++; a1++; a1++;

(byte)a3 = a1; a3++

0x20

0x10

<https://t.me/learningnets>

ARBITRARY PAYLOAD ENCODING

Initial

Payload: **Encoded \mathcal{P}_{enc} :**

0x03

```
a1++; a1++; a1++;  
*(byte*)a3 = a1; a3++
```

0x20

```
a1++... × (0x20 - 0x03) = 0x1D
```

0x10

<https://t.me/learningnets>

ARBITRARY PAYLOAD ENCODING

Initial

Payload: **Encoded \mathcal{P}_{enc} :**

```
0x03            a1++; a1++; a1++;  
                *(byte*)a3 = a1; a3++  
  
0x20            a1++... × (0x20 - 0x03) = 0x1D  
                *(byte*)a3 = a1; a3++  
  
0x10
```

<https://t.me/learningnets>

ARBITRARY PAYLOAD ENCODING

Initial

Payload: **Encoded \mathcal{P}_{enc} :**

0x03	<code>a1++; a1++; a1++;</code> <code>*(byte*)a3 = a1; a3++</code>
0x20	<code>a1++... × (0x20 - 0x03) = 0x1D</code> <code>*(byte*)a3 = a1; a3++</code>
0x10	<code>a1++... × (0x10 - 0x20) = 0xF0</code> <code>*(byte*)a3 = a1</code>

<https://t.me/learningnets>

ARBITRARY PAYLOAD ENCODING

Initial

Payload:

Encoded \mathcal{P}_{enc} :

Decoded

Payload:

0x03

```
a1++; a1++; a1++;  
*(byte*)a3 = a1; a3++
```

0x20

```
a1++... × (0x20 - 0x03) = 0x1D  
*(byte*)a3 = a1; a3++
```

0x10

```
a1++... × (0x10 - 0x20) = 0xF0  
*(byte*)a3 = a1
```

<https://t.me/learningnets>

ARBITRARY PAYLOAD ENCODING

Initial

Payload:

Encoded \mathcal{P}_{enc} :

0x03

```
a1++; a1++; a1++;  
*(byte*)a3 = a1; a3++
```

0x20

```
a1++... × (0x20 - 0x03) = 0x1D  
*(byte*)a3 = a1; a3++
```

0x10

```
a1++... × (0x10 - 0x20) = 0xF0  
*(byte*)a3 = a1
```

Decoded

Payload:

0x03

<https://t.me/learningnets>

ARBITRARY PAYLOAD ENCODING

Initial

Payload:

Encoded \mathcal{P}_{enc} :

0x03

```
a1++; a1++; a1++;  
*(byte*)a3 = a1; a3++
```

0x20

```
a1++... × (0x20 - 0x03) = 0x1D  
*(byte*)a3 = a1; a3++
```

0x10

```
a1++... × (0x10 - 0x20) = 0xF0  
*(byte*)a3 = a1
```

Decoded

Payload:

0x03

0x20

<https://t.me/learningnets>

ARBITRARY PAYLOAD ENCODING

Initial

Payload:

Encoded \mathcal{P}_{enc} :

0x03

```
a1++; a1++; a1++;  
*(byte*)a3 = a1; a3++
```

0x20

```
a1++... × (0x20 - 0x03) = 0x1D  
*(byte*)a3 = a1; a3++
```

0x10

```
a1++... × (0x10 - 0x20) = 0xF0  
*(byte*)a3 = a1
```

Decoded

Payload:

0x03

0x20

0x10

<https://t.me/learningnets>

ARBITRARY PAYLOAD ENCODING

Initial

Payload:

Encoded \mathcal{P}_{enc} :

Decoded

Payload:

0x03

```
a1++; a1++; a1++;  
*(byte*)a3 = a1; a3++
```

0x03

0x20

```
a1++... × (0x20 - 0x03) = 0x1D  
*(byte*)a3 = a1; a3++
```

0x20

0x10

```
a1++... × (0x10 - 0x20) = 0xF0  
*(byte*)a3 = a1
```

0x10

<https://t.me/learningnets>

GDB OVER BEAMER TIME!



<https://t.me/learningnets>

GOB OVER BEAMER TIME!



```

add    s3 , s3 , s8
sw     a1 , 0(a3)
bnez   a5 , +0x5a
    }  *(byte*)a3 = a1

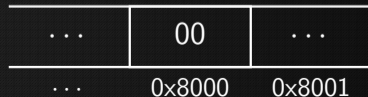
```

```

...
add    s3 , s3 , s8
bnez   a3 , +0x2
add    a3 , a3 , t2
    }  a3++

```

a1	0000 00AB
a3	0000 8000
t2	0000 0001
a5	B100 D5AC
s3	530F 25F8
s8	03C4 9ECC



<https://t.me/learningnets>

GOB OVER BEAMER TIME!



```

add s3, s3, s8
sw a1, 0(a3)
bnez a5, +0x5a

```

} `*(byte*)a3 = a1`

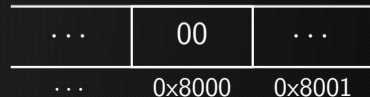
```

...
add s3, s3, s8
bnez a3, +0x2
add a3, a3, t2

```

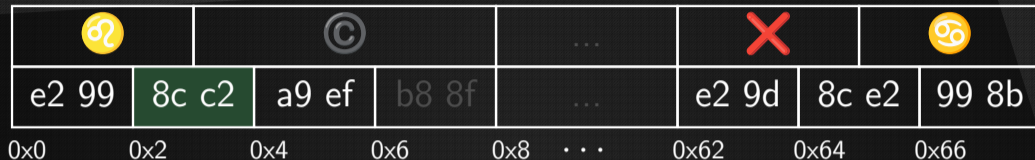
} `a3++`

a1	0000 00AB
a3	0000 8000
t2	0000 0001
a5	B100 D5AC
s3	56D3 C4C4
s8	03C4 9ECC



<https://t.me/learningnets>

GOB OVER BEAMER TIME!



```

add    s3, s3, s8
sw     a1, 0(a3)
bnez   a5, +0x5a
    
```

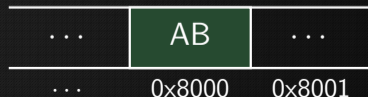
} `*(byte*)a3 = a1`

```

...
add    s3, s3, s8
bnez   a3, +0x2
add    a3, a3, t2
    
```

} `a3++`

a1	0000 00AB
a3	0000 8000
t2	0000 0001
a5	B100 D5AC
s3	56D3 C4C4
s8	03C4 9ECC



<https://t.me/learningnets>

GOB OVER BEAMER TIME!



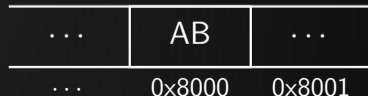
```

add    s3, s3, s8
sw     a1, 0(a3)
bnez  a5, +0x5a
    ...
    } *(byte*)a3 = a1
  
```

```

    ...
    } a3++
  add    s3, s3, s8
  bnez   a3, +0x2
  add    a3, a3, t2
  
```

a1	0000 00AB
a3	0000 8000
t2	0000 0001
a5	B100 D5AC
s3	56D3 C4C4
s8	03C4 9ECC



<https://t.me/learningnets>

GOB OVER BEAMER TIME!



```

add    s3, s3, s8
sw     a1, 0(a3)
bnez   a5, +0x5a
    
```

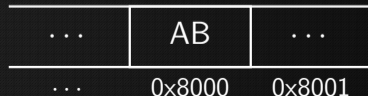
} `*(byte*)a3 = a1`

```

...
add    s3, s3, s8
bnez   a3, +0x2
add    a3, a3, t2
    
```

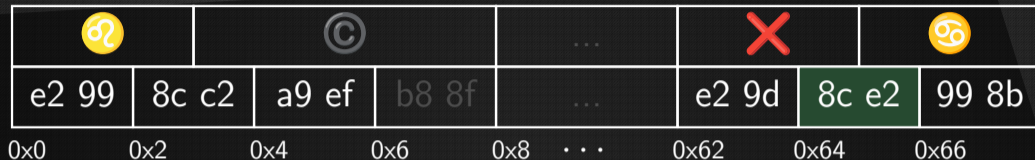
} `a3++`

a1	0000 00AB
a3	0000 8000
t2	0000 0001
a5	B100 D5AC
s3	5A98 6390
s8	03C4 9ECC



<https://t.me/learningnets>

GOB OVER BEAMER TIME!



```

add    s3, s3, s8
sw     a1, 0(a3)
bnez   a5, +0x5a
    
```

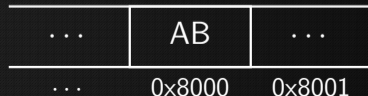
} `*(byte*)a3 = a1`

```

...
add    s3, s3, s8
bnez   a3, +0x2
add    a3, a3, t2
    
```

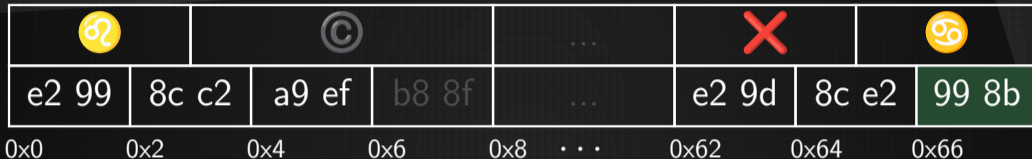
} `a3++`

a1	0000 00AB
a3	0000 8000
t2	0000 0001
a5	B100 D5AC
s3	5A98 6390
s8	03C4 9ECC



<https://t.me/learningnets>

GOB OVER BEAMER TIME!



```

add    s3, s3, s8
sw     a1, 0(a3)
bnez   a5, +0x5a
    ...
    } *(byte*)a3 = a1

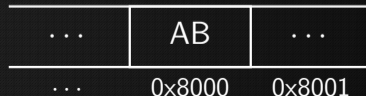
```

```

add    s3, s3, s8
bnez   a3, +0x2
add    a3, a3, t2
    } a3++

```

a1	0000 00AB
a3	0000 8001
t2	0000 0001
a5	B100 D5AC
s3	5A98 6390
s8	03C4 9ECC



<https://t.me/learningnets>

GOB OVER BEAMER TIME!



```

add    s3, s3, s8
sw     a1, 0(a3)
bnez   a5, +0x5a
    ...
    } *(byte*)a3 = a1

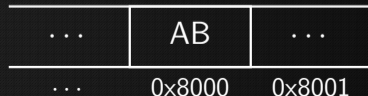
```

```

add    s3, s3, s8
bnez   a3, +0x2
add    a3, a3, t2
    ...
    } a3++

```

a1	0000 00AB
a3	0000 8001
t2	0000 0001
a5	B100 D5AC
s3	5A98 6390
s8	03C4 9ECC



<https://t.me/learningnets>

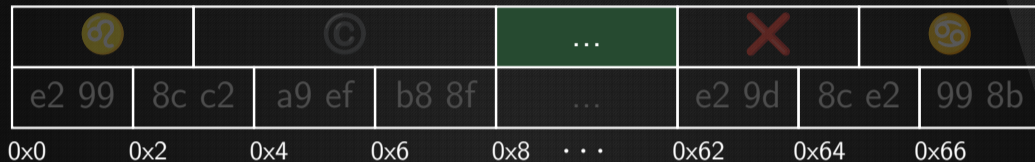
DEMO ON RV64GC SIFIVE UNLEASHED BOARD



Click here for the demo

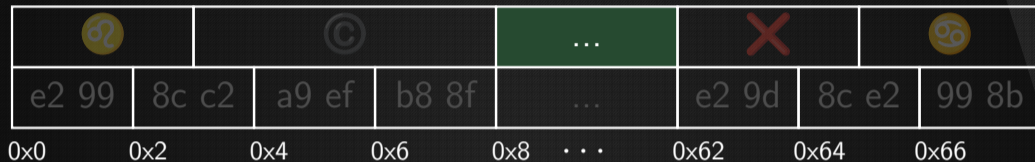
<https://t.me/learningnets>

FILLING THE VOIDS



<https://t.me/learningnets>

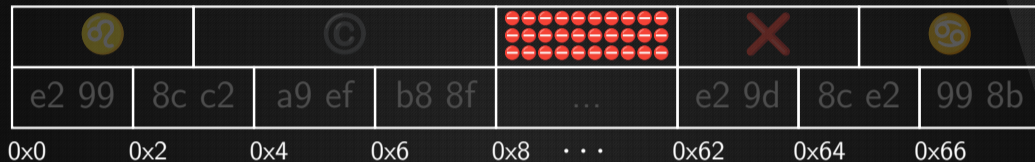
FILLING THE VOIDS



- Just solve a variant of subset sum (RTFM Wikipedia).

<https://t.me/learningnets>

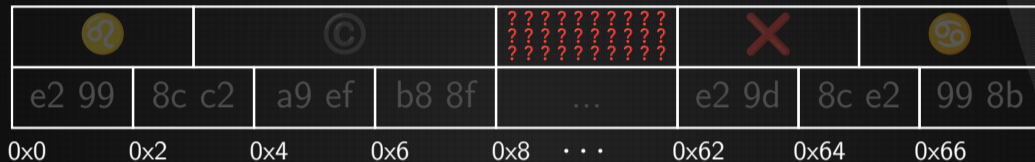
FILLING THE VOIDS



- Just solve a variant of subset sum (RTFM Wikipedia).
 - ▶ Trivial if we use only 3 and 4-byte emojis ...

<https://t.me/learningnets>

FILLING THE VOIDS



■ Just solve a variant of subset sum (RTFM Wikipedia).

- ▶ Trivial if we use only 3 and 4-byte emojis ...
- ▶ ... harder if we want to minimize number of emojis
⇒ dynamic programming (takes 4'13" to implement)

<https://t.me/learningnets>

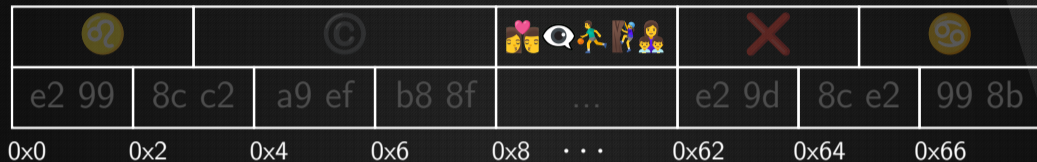
FILLING THE VOIDS



- Just solve a variant of subset sum (RTFM Wikipedia).
 - ▶ Trivial if we use only 3 and 4-byte emojis ...
 - ▶ ... harder if we want to minimize number of emojis
⇒ dynamic programming (takes 4'13" to implement)
- Fantastic, we get polymorphism for free!

<https://t.me/learningnets>

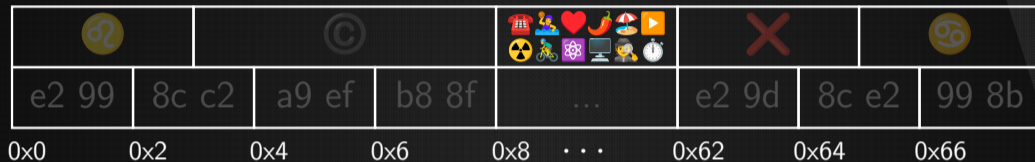
FILLING THE VOIDS



- Just solve a variant of subset sum (RTFM Wikipedia).
 - ▶ Trivial if we use only 3 and 4-byte emojis ...
 - ▶ ... harder if we want to minimize number of emojis
⇒ dynamic programming (takes 4'13" to implement)
- Fantastic, we get polymorphism for free!

<https://t.me/learningnets>

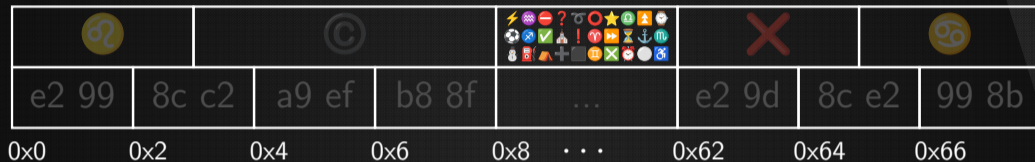
FILLING THE VOIDS



- Just solve a variant of subset sum (RTFM Wikipedia).
 - ▶ Trivial if we use only 3 and 4-byte emojis ...
 - ▶ ... harder if we want to minimize number of emojis
⇒ dynamic programming (takes 4'13" to implement)
- Fantastic, we get polymorphism for free!

<https://t.me/learningnets>

FILLING THE VOIDS



■ Just solve a variant of subset sum (RTFM Wikipedia).

- ▶ Trivial if we use only 3 and 4-byte emojis ...
- ▶ ... harder if we want to minimize number of emojis
⇒ dynamic programming (takes 4'13" to implement)

■ Fantastic, we get polymorphism for free!

<https://t.me/learningnets>

TRIVIA

Can we do our nop-sled with the 🛷 emoji?

<https://t.me/learningnets>

TRIVIA

Can we do our nop-sled with the 📎 emoji?



<https://t.me/learningnets>

TRIVIA

Can we do our nop-sled with the 📖 emoji?



<https://t.me/learningnets>

TRIVIA

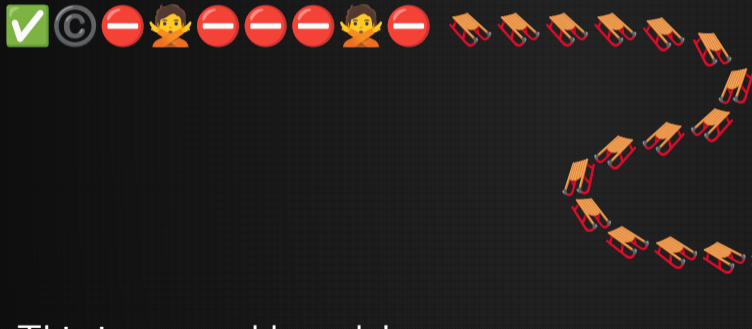
Can we do our nop-sled with the 🎒 emoji?



<https://t.me/learningnets>

TRIVIA

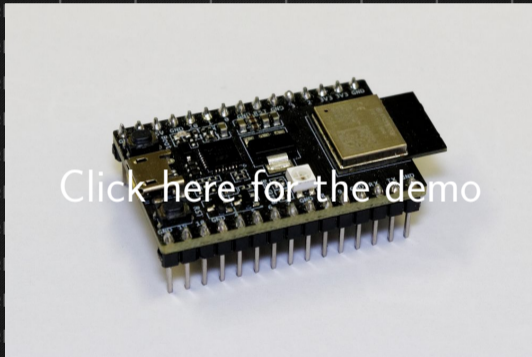
Can we do our nop-sled with the 🪑 emoji?



This is executable code!

<https://t.me/learningnets>

DEMO ON RV32 ESPRESSIF ESP32-C3 BOARD



Click here for the demo

<https://t.me/learningnets>

666. CONCLUSION

<https://t.me/learningnets>



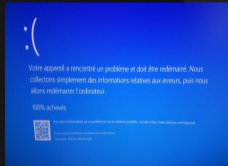
666. CONCLUSION

<https://t.me/learningnets>

ACKNOWLEDGMENTS

The following were harmed during the making of this project:

- LuaLatex (segfault)
- TexStudio (memory leaks, crashes, ...)
- Acrobat Reader (major glitches)
- Various PDF readers (glitches)
- VLC (glitches on demo.mkv)
- Firefox (broke PDF.js persistently)
- xfce4-terminal (utter slowness)
- Noto Emoji (render issues)
- CUPS (crash)
- gcc (segfault)
- Windows (BSOD when compiling slides)



Emoji support **is** hard.

<https://t.me/learningnets>

We went all the way to design a new code-reuse technique
just to hack you with unsolicited “I 💖🌟 U” texts.

Code and documentation on:

<https://github.com/RischarDV/emoji-shellcoding>

Your friendly neighbourhood hackers

hadrien.barral@ens.fr

georges-axel.jaloyan@ens.fr

<https://t.me/learningnets>