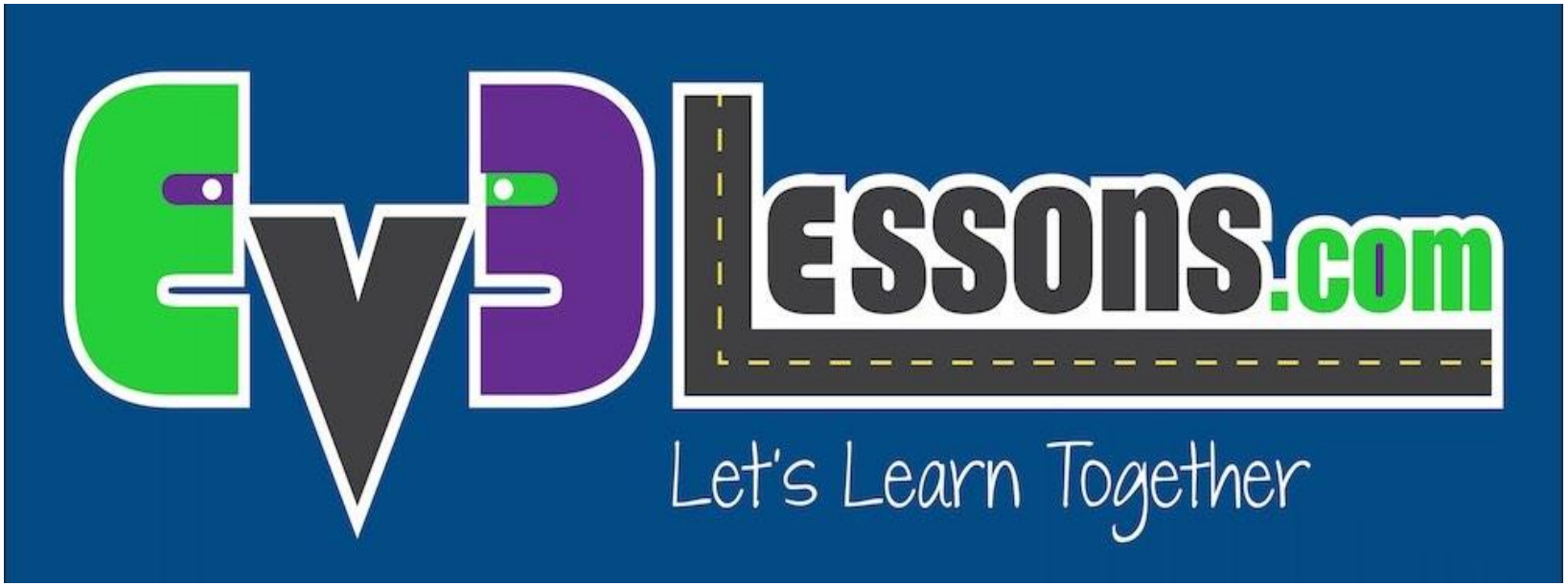


Branching Error
(a.k.a. the VM Program Instruction Break Error)

By Sanjay and Arvind Seshan

DEBUGGING LESSON

ISTORIC

- Am întâlnit pentru prima dată eroarea “VM Program Instruction Break” pe brick-ul nostru la sfârșitul lui 2013 în timpul sezonului Nature’s Fury FLL. Am căutat online pentru o documentație despre această eroare, dar nu am găsit niciuna. Am primii care au raportat această problemă pe forumurile FLL.
- Mai multe echipe de FLL și WRO au întâlnit aceeași eroare de atunci. În timp ce ei au persistat și încercat să vină cu o soluție alternativă, acestea n-au fost niciodată de ajuns.
- Fără a ști ce cauzează eroarea, era dificil să venim cu o soluție permanentă. Singura soluție disponibilă la acel moment era încercare și eroare.
- Acest document subliniază care au fost cauzele și soluția.

SIMPTOME OBIȘNUITE

- **Robotul se oprește în mijlocul programului și afișează în mijlocul ecranului “VM Program Instruction Break”.**
- **Adăugarea unui cod de debug făcea ca eroarea să apară într-o locație diferită a codului.**
- **Eroare apare chiar și cu schimbări minimale la cod cum ar fi mișcarea la o poziție relativă a două My Block-uri.**
- **Adesea apare în coduri complexe (e.g. Ni s-a întâmplat destul de des în fiecare sezon pe măsură ce adăugam mai mult cod la codul nostru principal)**

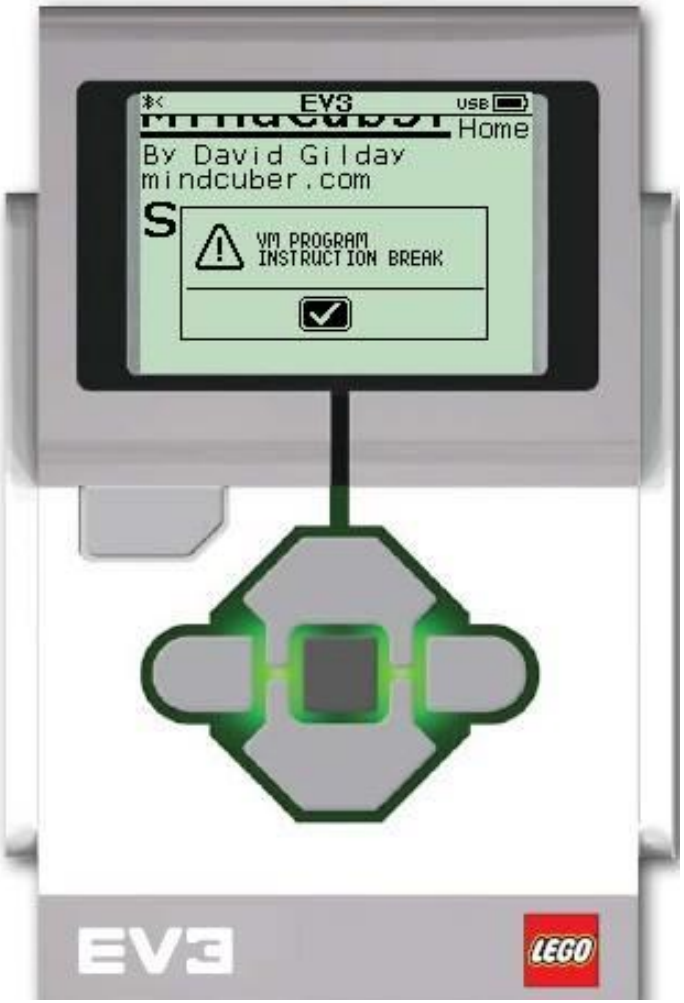


Image provided by David Gilday

CE ESTE UN VM?

O mașină virtuală (VM) este o emulație a sistemului calculatorului. Acest sistem “emulated” poate fi total diferit de calculatorul pe care tu rulezi VM. De exemplu, poți rula un VM emulând un iPhone pe laptop-ul tău pentru a rula sau testa un software de telefoane.

EV3-ul utilizează un procesor TI's Sitara AM1808 ARM9™ care rulează pe Linux OS. Cu troate acestea codul pe care îl descarci pe EV3 nu este un ARM9 binar. El conține EV3 “bytecode” care este interpretat de VM-ul care rulează pe EV3.

„Bytecode-ul” pentru EV3 definește un set simplu de instrucțiuni care să acceseze hardware-ul conectat la EV3 (ecran, bluetooth, motoare, etc.)

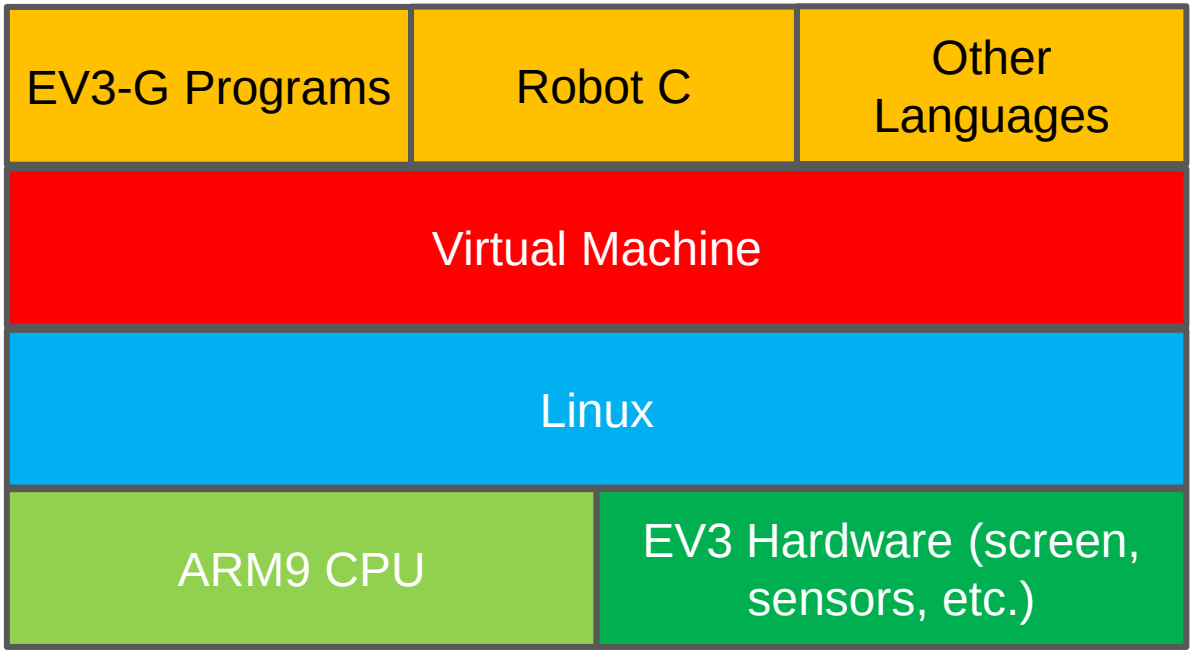
CE ESTE UN BYTECODE?

Bytecode-urile sunt apropiate de block-urile pe care le vezi în EV3-G. De exemplu:

BYTECODE: OUTPUT_POWER(0,1,50). Această comandă particulară setează puterea motorului de pe portul 1. Alte bytecode-uri pornesc și opresc motoarele.

To learn more, visit:
http://analyticphysics.com/Diversions/Assembly%20Language%20Programming%20for%20LEGO%20Mindstorms%20EV3.htm

CE ROL JOACĂ VM?



VM stă între programele tale și sistemul de operare care rulează pe EV3

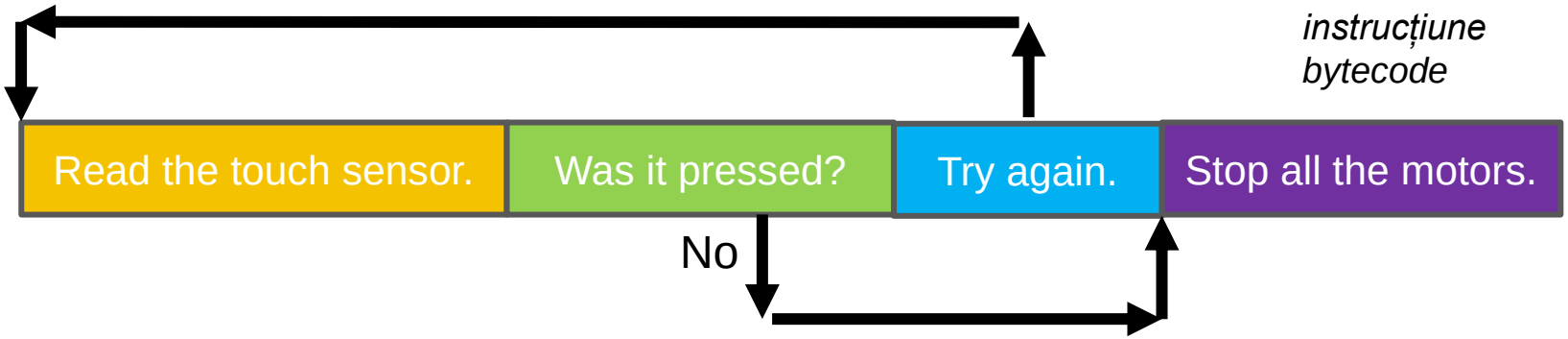
De observant că sistemele cum ar fi ev3dev rulează pe versiunea lor actualizată de Linux, cu propriile drivere pentru harware-ul EV3 (i.e. Ei nu folosesc interpretul bytecode VM)

SURSA PROBLEMEI

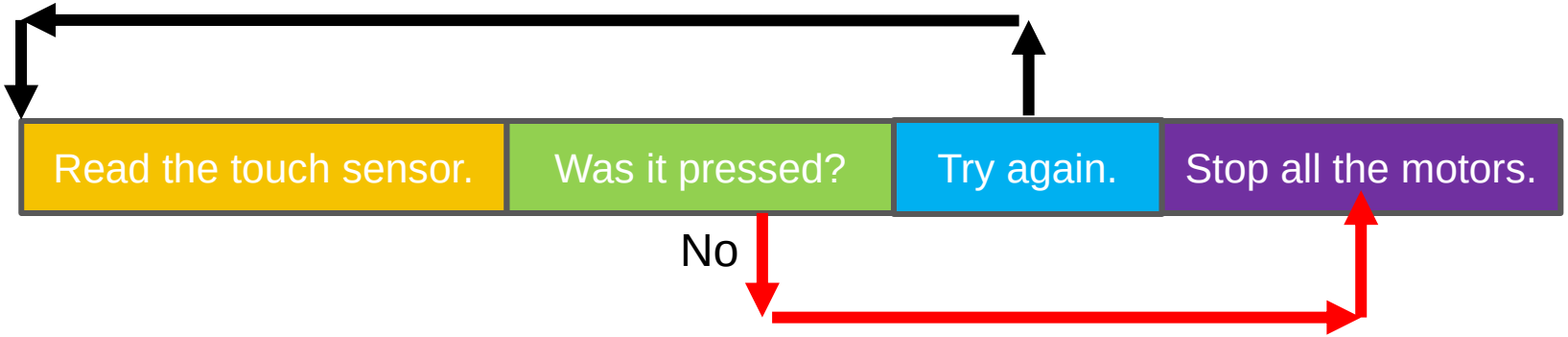
- **A fost un bug pe VM?**
  - Nu, Se pare că problema este cu compilatorul de pe PC care generează bytecode-uri incorecte. Specific a fost o problemă de ramuri în codul generat.
- **Ce este o ramură de cod?**
  - În mod normal, EV3-ul execută instrucțiuni într-o ordine secvențială.
  - O instrucțiune ramură este aceea care testează condiții (e.g. Dacă e butonul apăsat) și cauzează ca EV3 să sară la un alt set de instrucțiuni dacă se îndeplinește acea condiție
  - Ramurile de cod sunt utilizate pentru implementarea Switches, Loops și aproape orice comandă din care rezultă mai multe posibilități.
  - Bytecode-urile EV3 au ramuri necondiționale care întotdeauna sar la altă comandă din cod, și ramuri condiționale care testează una sau două piese de date.

O PRIVIRE SIMPLĂ

Fiecare cutie este o instrucțiune bytecode



Ce vrei codul tău să facă: In the top case, the branches jump to the beginning of each sentence

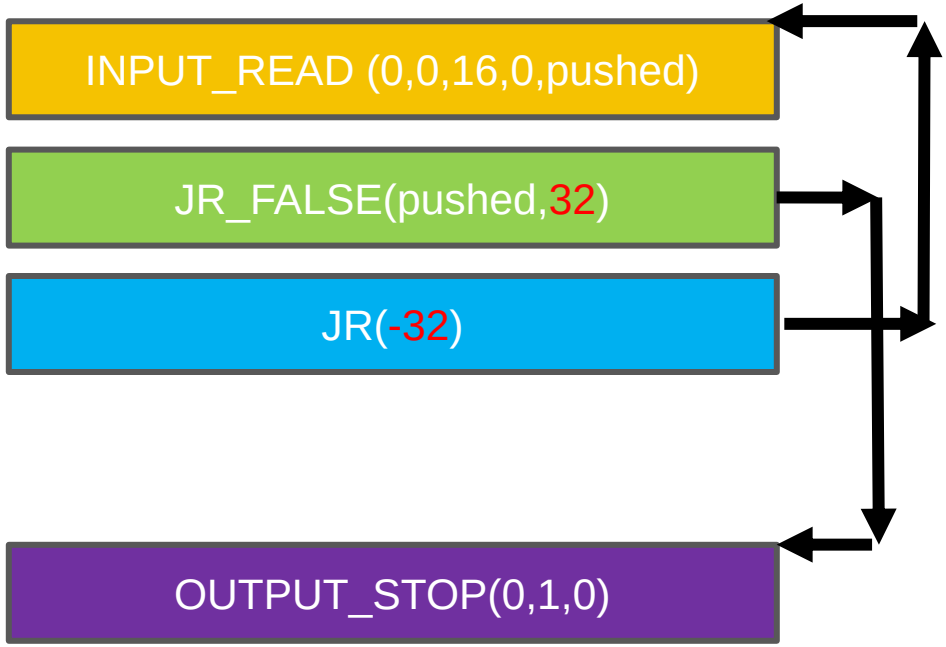


Ce se întâmplă în instrucțiunea de oprire într-un program VM: în cazul de față, ramura sare prea mult. EV3-ul încearcă să interpreteze ce înseamnă comanda "the motors" și eșuează.

O PRIVIRE ASUPRA BYTECODE

buttonPushed:	Acesta are eticheta “button pushed” loop
INPUT_READ (0,0,16,0,pushed)	Citește atingerea pe portul și stochează în variabilă “pushed”
JR_FALSE(pushed,buttonNotPushed)	Dacă “pushed” este FALSE iese din loop prin mutarea la butonul buttonNotPushed. Dacă nu e apăsat, sare la instrucțiunea următoare.
JR(buttonPushed)	Mergi înapoi la începutul etichetei “button pushed” loop
buttonNotPushed:	buttonNotPushed: label
OUTPUT_STOP(0,1,0)	Stop motor B

EXEMPLU DE BYTECODE



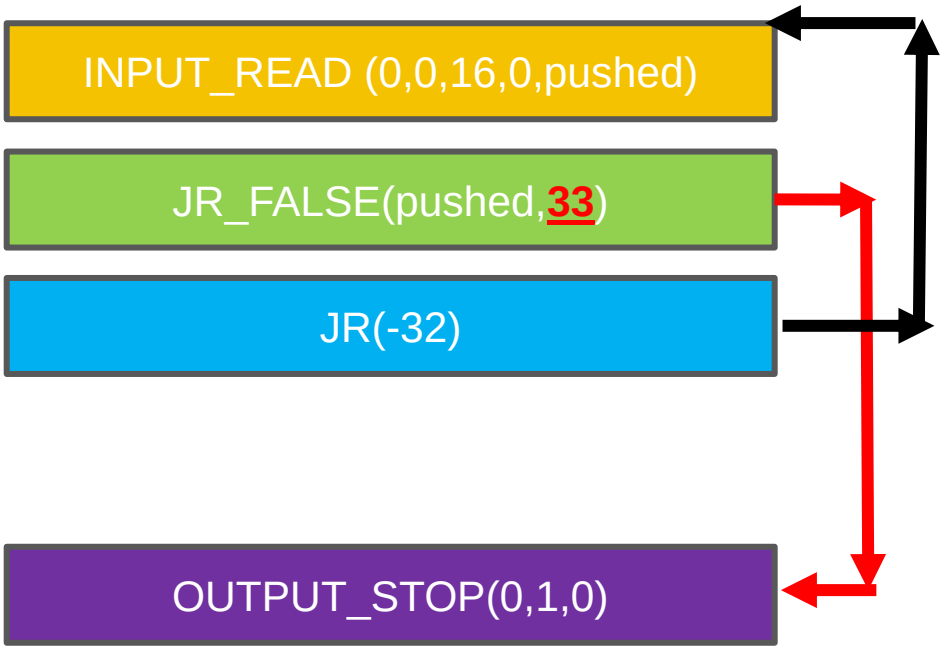
Codul actual nu include etichete dar include offset-uri.

Lungimea (or offset) săriturii este cu roșu.

Săgețile arată destinația săriturii.

De observat că săritura este la începutul fiecărei comenzi.

PROBLEMA



Offset-ul ramurii era uneori calculat incorect, spunea "33" în loc de 32 (în roșu).

Ca rezultat, ramura sărea în mijlocul instrucțiunii OUTPUT_STOP. E cași cum ai sări în mijlocul propoziției. Cel mai adesea instrucțiunea parțială nu avea niciun sens și VM răspundea cu "VM Instruction Break"

Uneori comanda parțială era o instrucțiune validă – dar nu cea pe care o doreai. De aceea robotul acționa incorect.

DE CE? ȘI CE SE ÎNTÂMPLĂ ACUM?

- **Sursa problemei este că programul care compilează codul pe calculator calculează lungimea ramurii incorect. (offset).**
- **LEGO a realizat o actualizare a software-ul de programare EV3 cu corecția bug-ului respectiv.**
  - Din10/25/2016, ambele Retail și Education editions of V. 1.2.2 sunt disponibile pentru descărcare.
- **Descarcă și instalează update-ul pe calculator**
  - După asta, poți încărca orice program care genera “VM Instruction Break” cauzat de rădăcinile eronate și descarcă-l din nou pe EV3. Codul nou descărcat nu ar trebui să mai aibă această problemă!

CÂTEVA LECȚII

- **Raportarea erorilor poate fi utilă**
  - O mare parte în găsirea soluțiilor la eroarea “VM Program Instruction Break” a fost faptul că echipele FLL, WRO ți alți utilizatori au raportat și discutat erorile.
  - La fel ca atunci când primești de la Google sau Microsoft mesajul cu “report this error” pe ecranul tău.

- **Deprinderea abilităților de depanare**
  - Echipele FIRST LEGO League, în particular, se confruntă cu această eroare în codul lor pe măsură ce acesta devine mai complex.
  - Ei au insistat și lucrat la această problemă cât de mult au putut.

UN EFORT COMUNITAR

Mulțumim Comunității MINDSTORMS, echipelor FLL, echipelor WRO, alți programatori din comunitate, National Instruments, și LEGO care au lucrat împreună pentru a identifica eroarea și a găsi o soluție.

CREDITS

- **Această lecție a fost scrisă de Arvind și Sanjay Seshan.**
- **Mai multe lecții sunt disponibile pe www.ev3lessons.com**
- **Această lecție a fost tradusă în limba romană de echipa FTC Rosophia #21455, RO20**