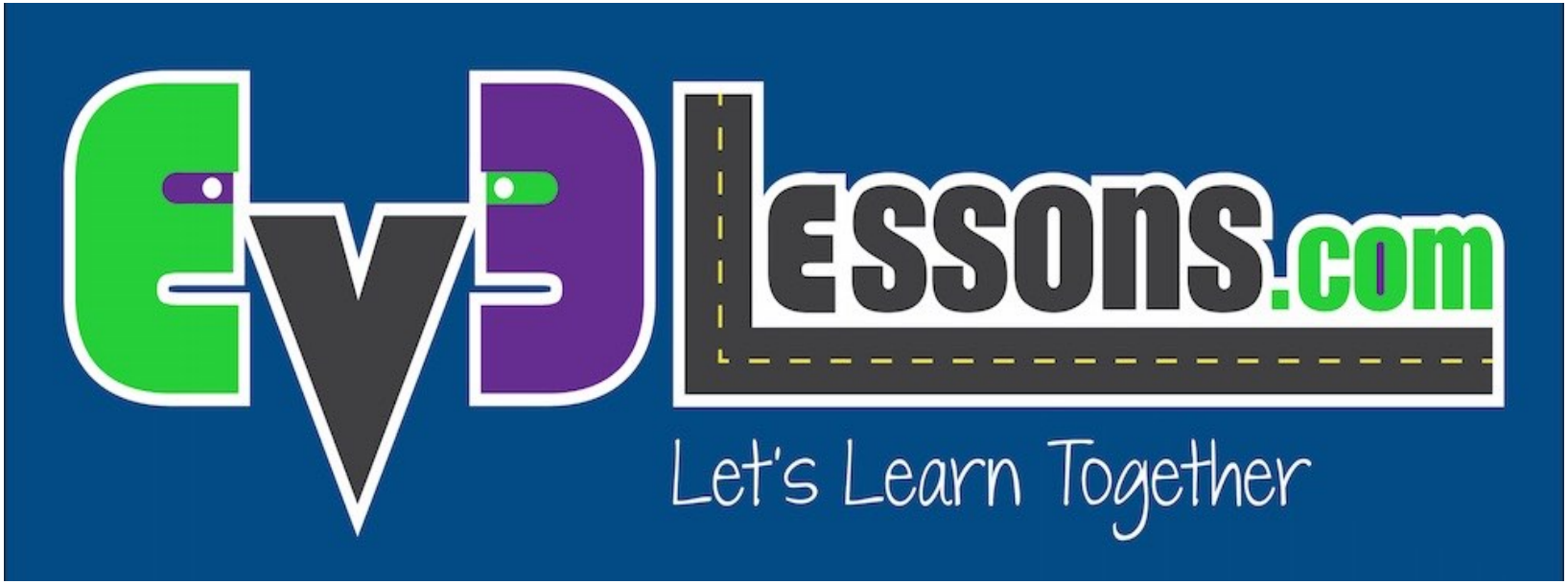



# Branching Error
# (a.k.a. the VM Instruction Break Error)

By Sanjay and Arvind Seshan


## DEBUGGING LESSON

# HISTORY

- We first encountered the “VM Program Instruction Break” error on our brick during the fall of 2013 during the Nature’s Fury FLL season. We searched online for any documentation about this error, but could not find any. We were the first to report this problem on the FLL Forum.
- Many FLL and WRO teams have encountered this error since then. While they persisted and tried to come up with a work-around, the solution was never enough.
- Without knowing what was causing the error, it was difficult to come up with a permanent solution. The only solution available at the time was trial and error.
- This presentation documents what the underlying causes were and the solution.

# COMMON SYMPTOMS

- The robot stops in the middle of a program and displays “VM Program Instruction Break” on the screen
- Adding debugging code may make the error appear in a different location of the code.
- The error would appear even with minimal changes to the code such as the movement of the relative position of two My Blocks
- Often occurs in more complex programs (e.g. it often happened to us each season as we added more code to our main sequencer)

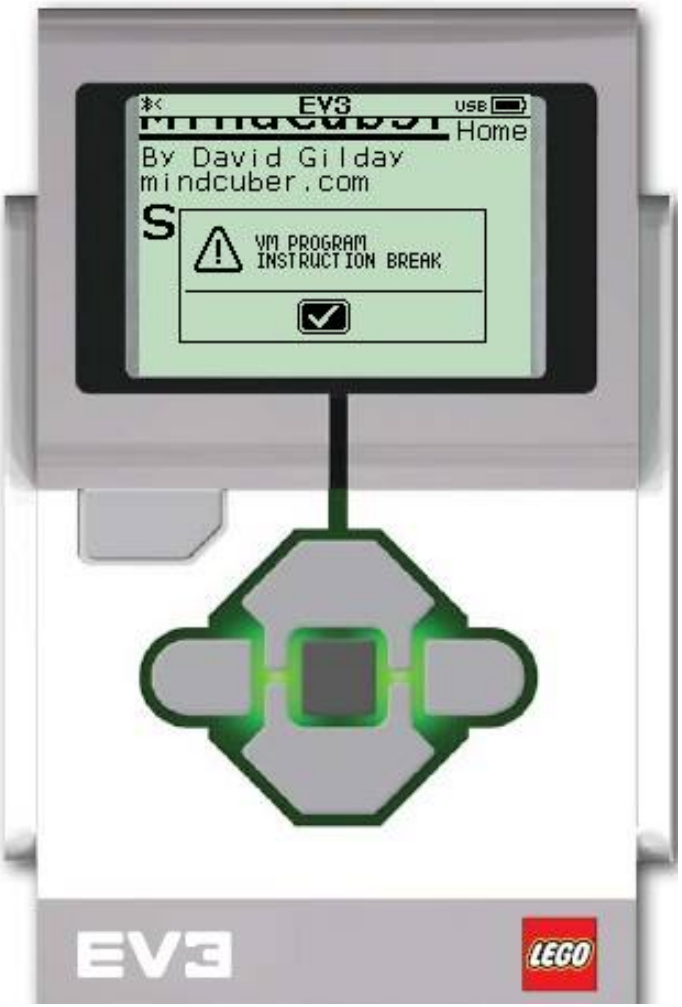


Image provided by David Gilday

# WHAT IS A VM?

A virtual machine (VM) is an emulation of a computer system. This “emulated” system maybe totally different than the computer you run the VM on. For example, you may run a VM emulating an iPhone on your laptop to run or test phone software.

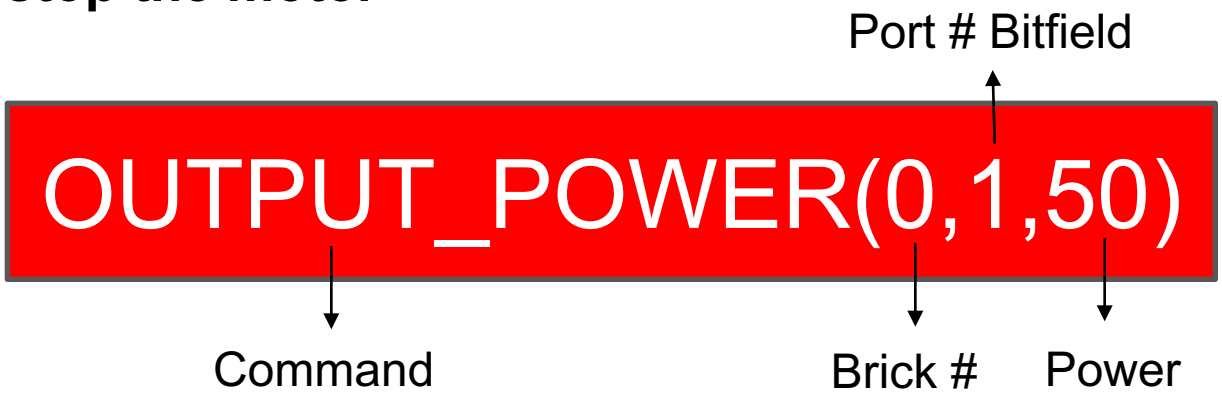
The EV3 uses a TI's Sitara AM1808 ARM9™ processor running the Linux OS. However, the code you download to the EV3 is not a ARM9 binary. It contains EV3 “bytecode” that is interpreted by a VM running on the EV3.

The bytecode for the EV3 defines a simple set of instructions to perform computations and access the hardware connected to the EV3 (screen, bluetooth, motors, etc.)

# WHAT IS BYTECODE?

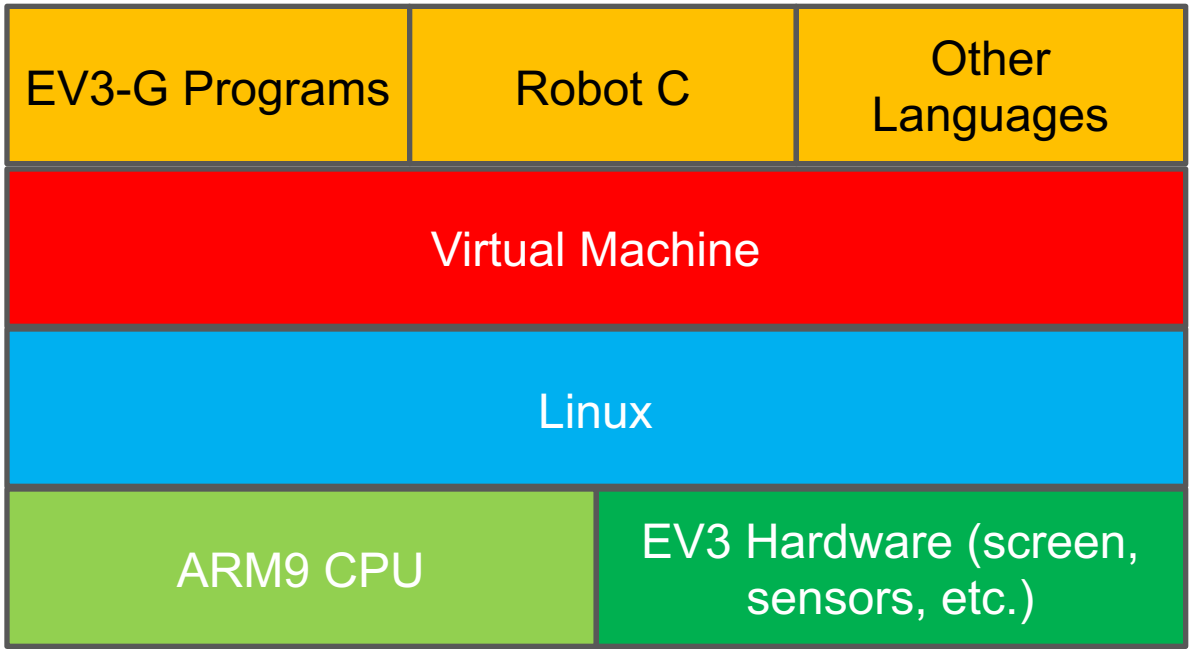
**The bytecodes are closely related to the blocks you see in EV3-G. For example:**

**BYTECODE: OUTPUT_POWER(0,1,50). This particular command sets the power of the motor on port 1. Other bytecodes actually start and stop the motor**




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

# WHAT ROLE DOES THE VM PLAY




**The VM sits between your programs and the operating system running on the EV3**

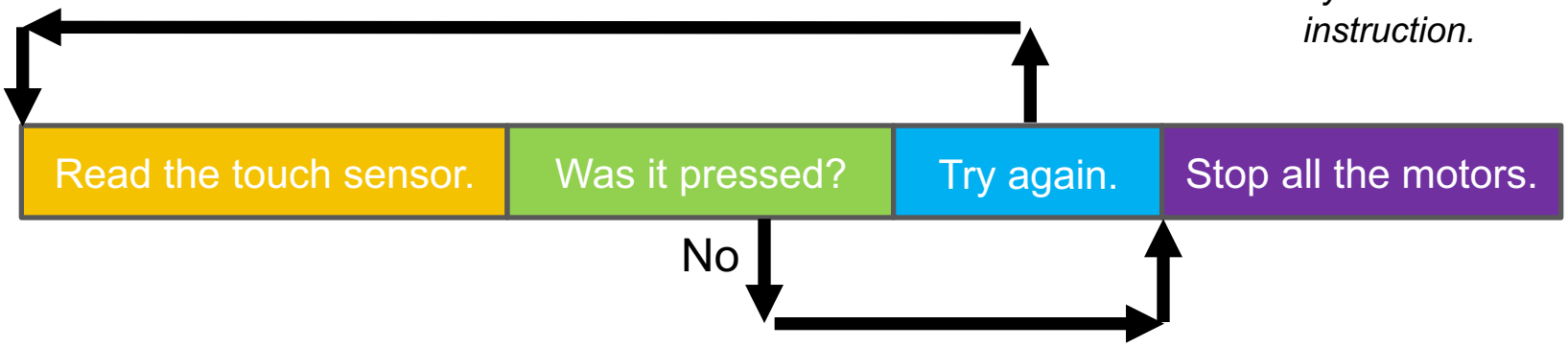
**Note that systems such ev3dev run their own updated Linux installation with their own drivers for the EV3 hardware (i.e. they don't use a VM bytecode interpreter)**

# SOURCE OF THE PROBLEM

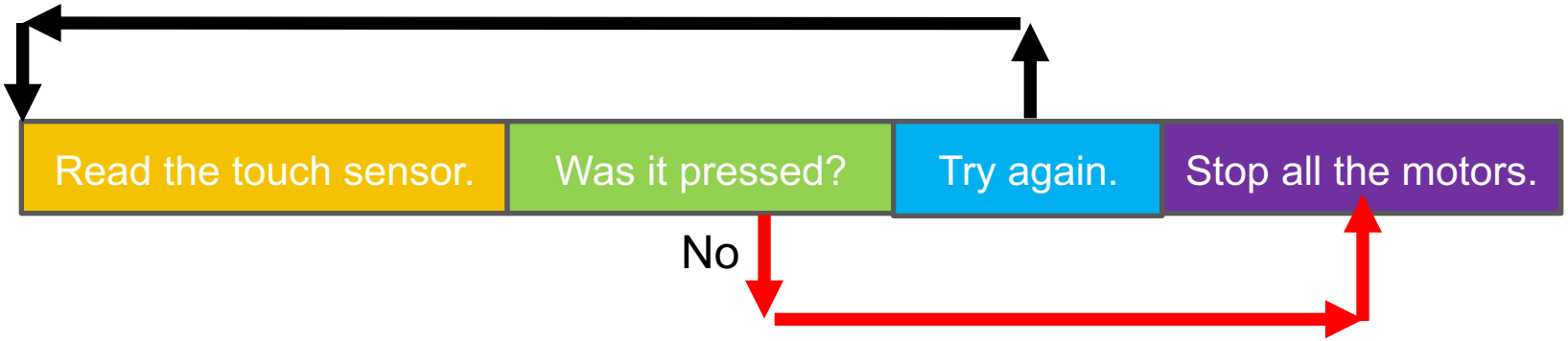
- **Was it really a bug in the VM?**
  - No. Turns out it an issue with the compiler on your PC generating incorrect bytecode. Specifically, it was a problem with the branches in the code generated.
- **What is branch code?**
  - Normally, your EV3 executes a sequences of instructions in sequential order
  - A branch (or jump) instruction is one that tests a condition (e.g. is the button pressed) and causes the EV3 to jump to a different set of instructions if the condition is met
  - Branches are used to implement Switches, Loops and almost any command that results in different possible results.
  - EV3 bytecode has unconditional branches that always jump to another piece of code, and conditional branches that test one or two pieces of data

# A SIMPLE VIEW

*Each box is one bytecode instruction.*




*What you want your code to do: In the top case, the branches jump to the beginning of each sentence*

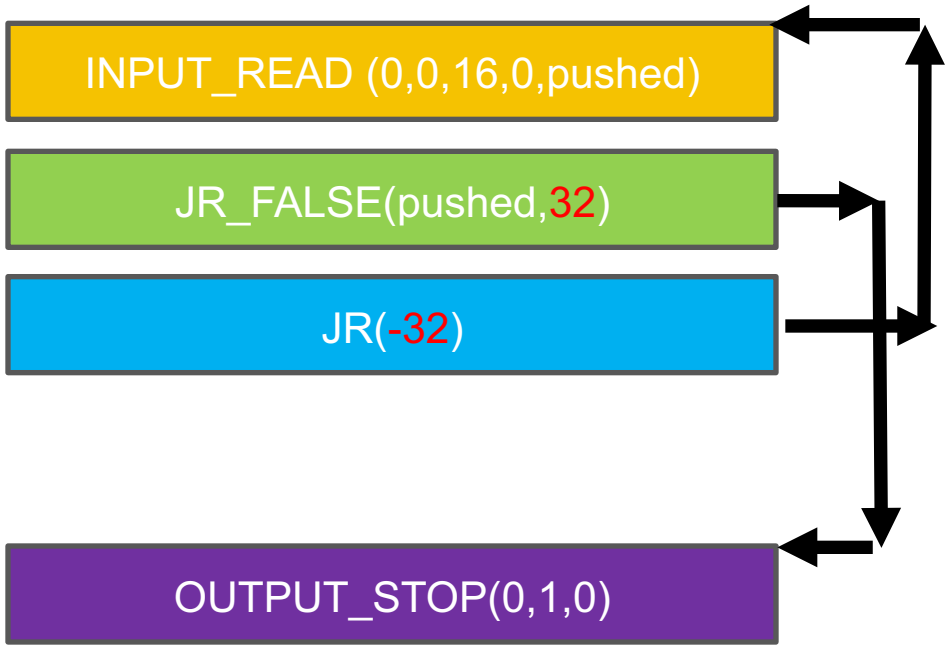



*What happens in a VM Instruction Break: In the bottom case, the branch jumps too far. The EV3 tries to interpret what the command "the motors" means and fails.*

# A BYTECODE VIEW

| | |
|---|---|
| buttonPushed: | This is the “button pushed” loop label |
| INPUT_READ (0,0,16,0,pushed) | Read touch on port 1 in touch mode and store in variable “pushed” |
| JR_FALSE(pushed,buttonNotPushed) | If “pushed” is FALSE exit the loop by jumping to the buttonNotPushed code. If it is not pushed, it just goes to the next instruction |
| JR(buttonPushed) | Go back to the beginning of the “button pushed” loop label |
| buttonNotPushed: | buttonNotPushed: label |
| OUTPUT_STOP(0,1,0) | Stop motor B |

# EXAMPLE BYTECODE



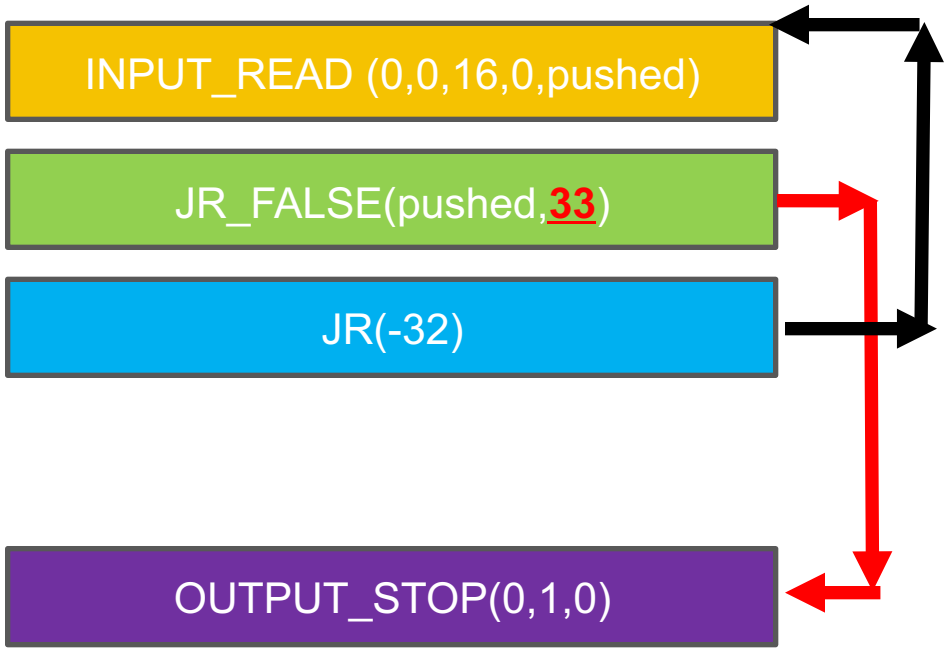

*Actual executed code does not include the labels, but does include the offsets.*

*Length (or offset) of the jump is in red.*

*Arrows point to the destination of the jump.*

*Note, the jump is to the start of each command.*

# THE PROBLEM




*The offset of the branch was sometimes calculated incorrectly. In this case, it says "33" instead of 32 (in red).*

*As a result, the branch would jump to the middle of OUTPUT_STOP instruction. This is like jumping to the middle of a sentence. Most often the partial instruction made no sense and the VM would respond with a "VM Instruction Break"*

*Sometimes the partial command was a valid instruction – just not the one you wanted. Therefore, your robot would act incorrectly.*

# WHY? AND WHAT HAPPENS NOW?

- The source of the problem is that the code compiler on your PC calculated an incorrect branch length (offset).
- LEGO is expected to release an update to the EV3 programming software soon with a bug fix (not yet released as of 10/17/2016)
- When this update is released, you should download and install it on your PC.
- After that, you can load any program you had symptoms such as “VM Instruction Break” that were caused by the bad branches and just download again to your EV3. The newly downloaded code should not have any bad branching code!

# SOME LESSONS

- **Reporting errors can be useful**
  - A big part of finding the solution to the “VM Instruction Break” error was FLL, WRO teams and other robot builders reporting and discussing errors
  - Similar to when you see a Google or Microsoft “report this error” message on your screen.
- **Learning debugging skills**
  - FIRST LEGO League teams, in particular, faced this error as their code became come complex
  - They persisted and worked around the problem as well as they could.

# A COMMUNITY EFFORT

**Thank you to MINDSTORMS Community Partners, FLL Teams, WRO Teams, other builders in the community and LEGO who worked together to identify this error and create a solution.**

# CREDITS

- **This tutorial was created by Sanjay Seshan and Arvind Seshan**
- **More lessons at www.ev3lessons.com**