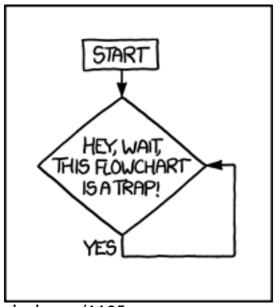
Lecture 5: Executing instructions

Wednesday, January 16, 2019 9:29 AM

Outline

- Finish instruction sets
- HLLs -> machine code
- How to execute an instruction



xkcd.com/1195

Finish instructions

Privileged specification

What is missing from the ISA as we've talked about it so far?

I/O and operating system functions

Privileged spec
↳ virtual memor
↳ CSR → control registers
↳ I/O . . . .

} still being developed

~ 300

From the Spec: https://riscv.org/specifications/privileged-isa/

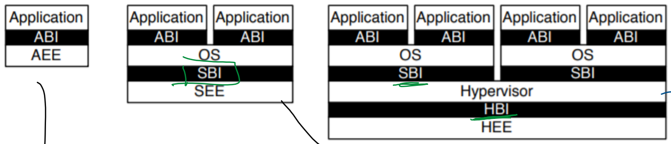


Figure 1.1: Different implementation stacks supporting various forms of privileged execution.

→ Virtual machines
for servers

→ like an OS but for OSes
+desktops
↳ virtualbox

→ Bare metal
ABI → app binary interface
AEE → app execution environment
+ faster/efficient
+ embedded systems
micro controllers

→ normal
operating system
-w/ multi processes
desktop/laptop/phone

High-level languages to machine code

$$\begin{tikzcd} C \arrow[r, "compile"] & assembly \arrow[r, bend left, "assembing"] & Object\ file \arrow[r, "link/load"] & machine\ code \\ & & Object \arrow[ur] & \end{tikzcd}$$

intermedicte
representation

IK files

ISA extras

ABI
Binary interface

- System calls (SBI/ABI)
- define stack
- define heap
- program counter
- data @ addr
- code @ addr
- frame layout
- caller + callee saved registers

Javascript?

```
function incrementX(obj) {
  return 1 + obj.x;
}
incrementX({x: 42});
```

- interpret the javascript bytecode
- or we can just-in-time compile the bytecode to object file/machine code

```
$ node --print-bytecode incrementX.js
...
[generating bytecode for function: incrementX]
Parameter count 2
Frame size 8
  12 E> 0x2ddf8802cf6e @    StackCheck
  19 S> 0x2ddf8802cf6f @    LdaSmi [1]
        0x2ddf8802cf71 @    Star r0
  34 E> 0x2ddf8802cf73 @    LdaNamedProperty a0, [0], [4]
  28 E> 0x2ddf8802cf77 @    Add r0, [6]
  36 S> 0x2ddf8802cf7a @    Return
Constant pool (size = 1)
0x2ddf8802cf21: [FixedArray] in OldSpace
 - map = 0x2ddfb2d02309 <Map(HOLEY_ELEMENTS)>
 - length: 1
           0: 0x2ddf8db91611 <String[1]: x>
Handler Table (size = 16)
```

→ loads 1 into accumulator
→ store acc. into r0
→ Stores value in acc
↳ name in table
x
→ add

How to execute an instruction

Steps to execute an instruction

1) Fetch instruction from memory
   set address from PC
   Access memory
2) Decode instruction
   look @ opcode, decide what to do
   read register values -compute immediate value
3) Execute the instruction
   → compute value/result for R-type
   → compute address
4) Access memory
   get value for load
   write value for store
5) Write back result to register file
   update PC w/ PC+4 or branch/jump target

lw a0, 1024(t0)

$M[1024+R[t0]] \rightarrow R[a0]$

$F \rightarrow D \rightarrow E \rightarrow M \rightarrow W$

Hardware needed: