Backdoors, Endianness

CS 2130: Computer Systems and Organization 1

Xinyao Yi Ph.D.
Assistant Professor

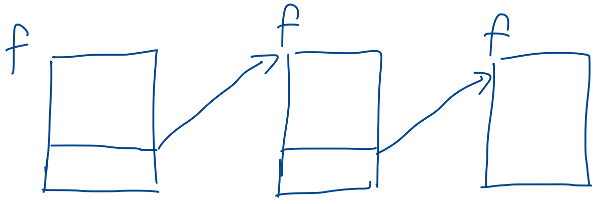
UNIVERSITY of VIRGINIA | ENGINEERING

Announcements

- Homework 4 due Monday after break on Gradescope
  – You have written most of this code already
  – Hint: Lab 7 may provide a fast way to get started

Storing Variables in Memory

So far... we/compiler chose location for variable



Consider the following example:

```
f(x):
    a=x
    if (x <= 0) return 0
    else return f(x-1) + a
```

$x=5$

$a=5$

$f(4)+5$ ⇒ Sums up the numbers between one and x

Recursion

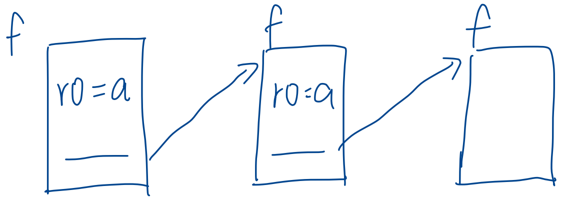
- The formal study of a function that calls itself

while it computing the solution or the result for F, it actually calls itself in a smaller input.

Storing Variables in Memory

```
f(x):
    a=x
    if (x <= 0) return 0
    else return f(x-1) + a
```

Where do we store a?



Save it in a register R0 ? No !

save it to memory ? M[0x80] No! override this value over and over

When I call the same code again. jump to the same code again

I accidently overwrite the value I really want.

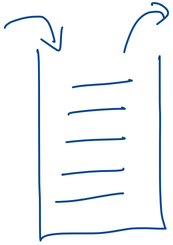
We need something that will help us to organize memory so that we keep track of these variables.

The Stack

(One solution won out)

Stack - a last-in-first-out (LIFO) data structure

- The solution for solving this problem



stack of plates

rsp - Special register - the *stack* pointer

- Points to a special location in memory
- Two operations most ISAs support:
  - push - put a new value on the stack
  - pop - return the top value off the stack

have the address.
(the index in memory of a certain point)

The Stack: Push and Pop

push r0

- Put a value onto the "top" of the stack
  - rsp -= 1
  - M[rsp] = r0

pop r2

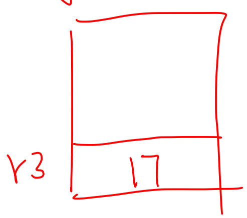
- Read value from "top", save to register
  - r2 = M[rsp]
  - rsp += 1

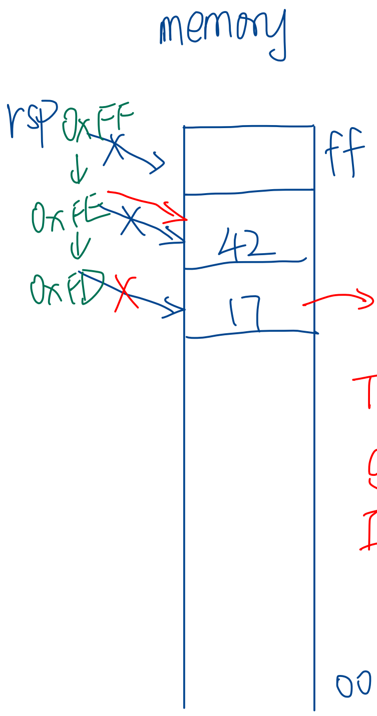
push 42

push 17

pop r3

register file.





(high order numbers up here)

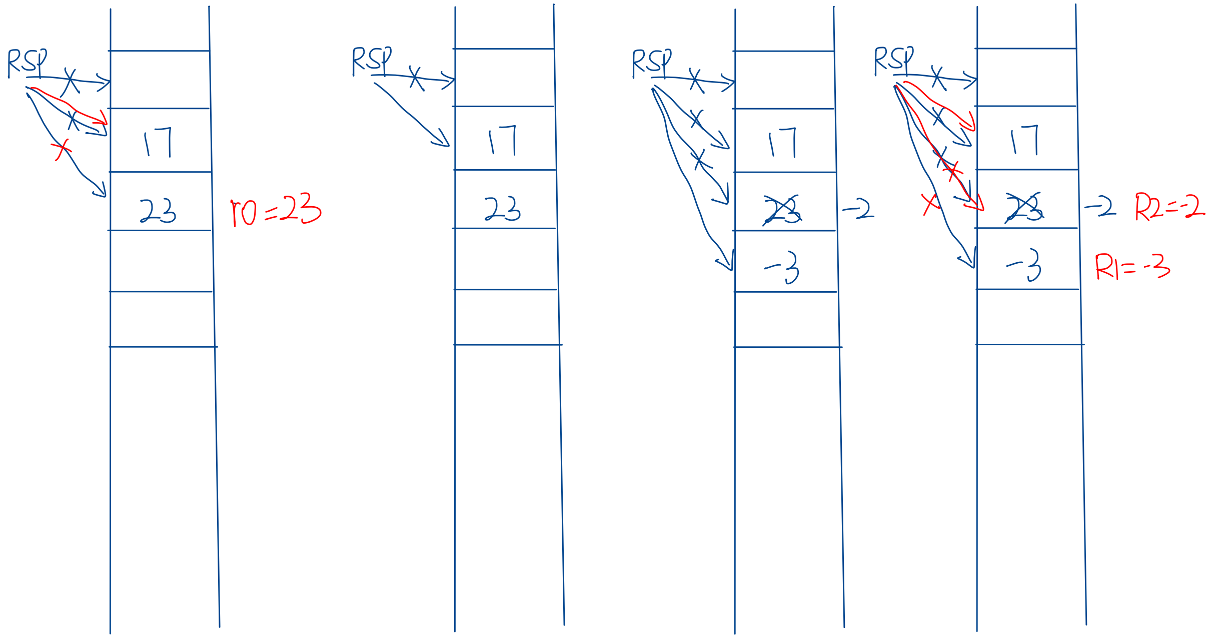
→ pop out r3=17

Then 17 turns to garbage in memory. It's not part of the stack now!

(low order indices way down).

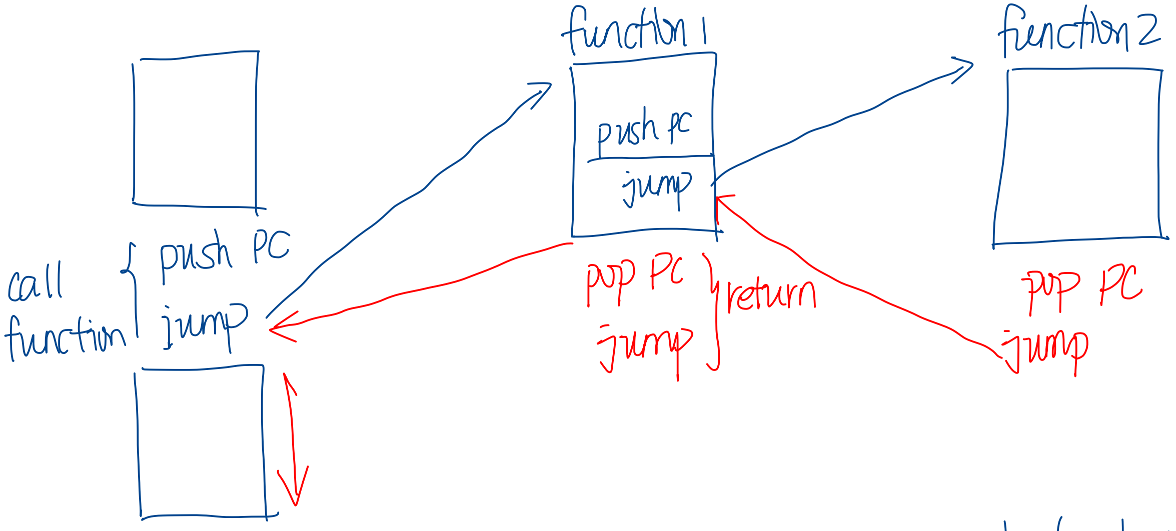
The Stack: Push and Pop

```
push(17)
push(23)
x = pop   (R0)
push(-2)
push(-3)
y = pop   (R1)
z = pop   (R2)
```



The Stack: Push and Pop

Function Calls



What about parameters ?

return value ?

calling conventions
- r2, r3 have operands
- r0 has return value

A short aside…

Time to take over the world!

Backdoors

→ allow an attacker/developer to take complete control of your system – often without you knowing it.

Backdoor: secret way in to do new unexpected things

- Get around the normal barriers of behavior
- Ex: a way in to allow me to take complete control of your computer

operating system has security checks: passwords, permissions, firewalls ⇒ all meant to control "who can do what"

Someone installs a backdoor ⇒ sneak past those protections and gain control without going through the normal process.

Backdoors

maybe a program? scripts? ⇒ the method or tool that uses a weakness in the system to do sth. unintended.

Exploit - a way to use a vulnerability or backdoor that has been created

- Our exploit today: a **malicious payload**
  - – A passcode and program
  - – If it ever gets in memory, run my program regardless of what you want to do

backdoor : hidden entrance

exploit : how you find it, open it and use it to get inside.

Attackers usually include a small program called a payload inside an exploit — once it gets into memory, the payload runs, even if you never gave permission.

Our Hardware Backdoor

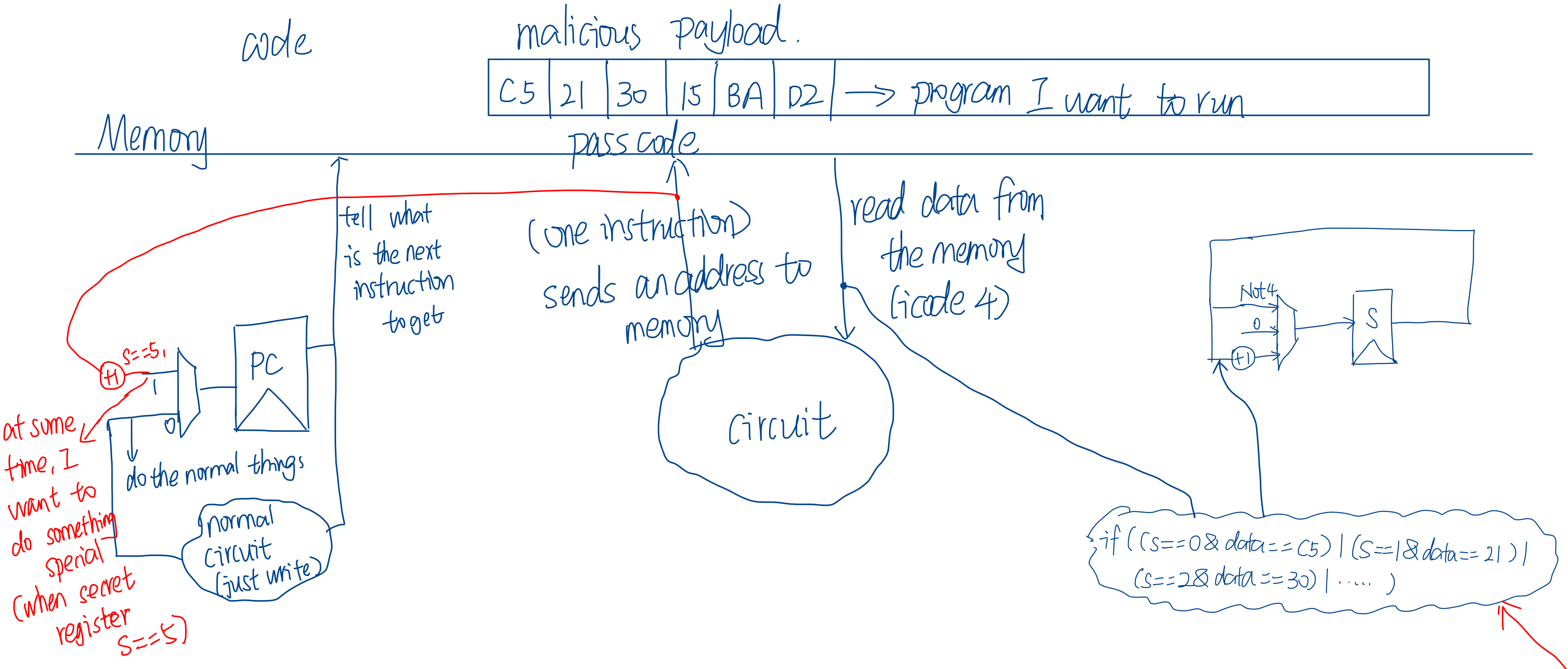
Our backdoor will have 2 components

- Passcode: need to recognize when we see the passcode
- Program: do something bad when I see the passcode

→ The hardware needs to recognize it when it appears.
(Like secret knock on a door, if you hear that pattern, you know someone special wants in.

→ Once the hardware recognizes the passcode, it executes some hidden or harmful behavior.

Our Hardware Backdoor


code
malicious payload.
C5
21
30
15
BA
D2
→ program I want to run
Memory
pass code
tell what is the next instruction to get
(one instruction)
sends an address to memory
read data from the memory (icode 4)
S==5,
+1
1
0
PC
at some time, I want to do something special (when secret register S==5)
do the normal things
normal circuit (just write)
Circuit
Not4
0
+1
S
if ((S==0 & data==C5) | (S==1 & data==21) | (S==2 & data==30) | .....)

Something not icode 4: keep the original value

if it is icode4: ① reset counter if that's not part of my passcode

② increasmentor: under special condition

Our Hardware Backdoor

Will you notice this on your chip?

Our Hardware Backdoor

Will you notice this on your chip?

- Modern chips have **billions** of transistors
- We’re talking adding a few hundred transistors

Our Hardware Backdoor

Will you notice this on your chip?

- Modern chips have **billions** of transistors
- We're talking adding a few hundred transistors
- Maybe with a microscope? But you'd need to know where to look!

If I had a microscope and I knew exactly where to look, I probably can find it.

Or, maybe, I'm very, very lucky.

But - Most exploits are going to be found after somebody tries to use them.

Our Hardware Backdoor

Have you heard about something like this before?

Our Hardware Backdoor

Have you heard about something like this before?

- Sounds like something from the movies
- People claim this might be happening

Our Hardware Backdoor

Have you heard about something like this before?

- Sounds like something from the movies
- People claim this might be happening
- To the best of my knowledge, no one has ever admitted to falling in this trap