CS216: Program and Data Representation
University of Virginia Computer Science
Spring 2006 David Evans

Lecture 21: Calling Conventions

http://www.cs.virginia.edu/cs216

Menu

- x86 C Calling Convention
- Java Byte Code Wizards

Calling Convention

- Rules for how the caller and the callee make subroutine calls
- Caller needs to know:
  - How to pass parameters
  - What registers might be bashed by called function
  - What is state of stack after return
  - Where to get result

Calling Convention

Caller	Callee
• Where to put params	• Where to find params
• What registers can I assume are same	• What registers must I not bash
• Where to find result	• Where to put result
• State of stack	• State of stack

Calling Convention: Easiest for Caller

Caller	Callee
• Where to put params	• Where to find params
• What registers can I assume are same	• What registers must I not bash
– **All of them**	– **None of them**
• Where to find result	• Where to put result
• State of stack	• State of stack

x86 C Calling Convention

Caller	**Callee**
• Params on stack in reverse order	• Find params on stack in reverse order
• Can assume EBX, EDI and ESI are same	• Must not bash EBX, EDI or ESI
• Find result in EAX	• Put result in EAX
• *Stack rules next*	• *Stack rules next*

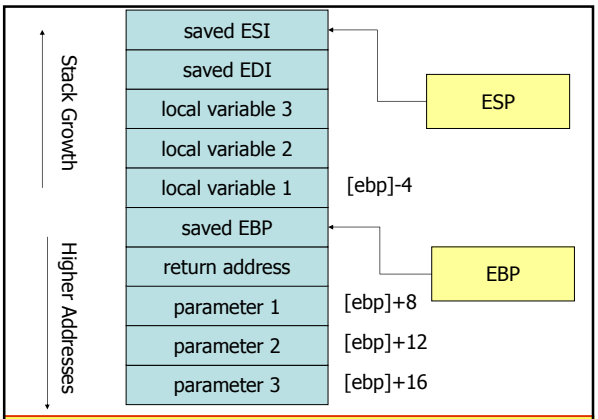
Need to save and restore (EAX), ECX, EDX

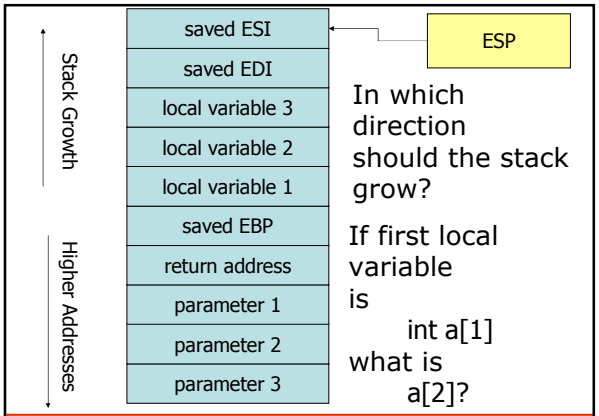
Need to save and restore EBX, EDI and ESI

Stack Convention

- EBP: Base Pointer
  - Reference point for finding local variables and parameters
  - Manipulated only explicitly
- ESP: Stack Pointer
  - Pointer to last element *used* on the stack
    - The next free element is at [esp]-4
  - Manipulated implicitly by many instructions: push, pop, call, ret

Do we need both of these?





In which direction should the stack grow?

If first local variable is
int a[1]
what is
a[2]?

Maintaining EBP

- Callee: prologue
  - Must save old value of EBP: push ebp
  - Update EBP to point to the new frame:
    mov EBP, ESP
  - Use EBP to find parameters: [ebp]+8, [ebp]+12, …
- Callee: before returning
  - Restore old value of EBP: pop ebp

Maintaining ESP

- Caller: it is maintained implicitly if parameters are push-ed
- Callee: prologue
  - Need to make enough space for local variables:
    sub esp, <space needed>
- Callee: before return
  - Deallocate local variables from stack:
    mov esp, ebp *; must be done before* pop ebp

Caller Rules

- Caller Before:
  1. Save bashable ("caller-save") registers we use
  2. Push parameters on stack
  3. call <routine>
     implicit: push EIP
- Caller After:
  2. Restore bashable registers
  1. Remove parameters from stack

Callee Rules

- Callee Prologue:
  1. Save EBP
  2. Update EBP
  3. Allocate local variables (ESP)
  4. Save *callee-save* registers we use (EBX, EDI, ESI)
- Callee Return:
  5. ret
  4. Restore EBP
  3. Deallocate local variables (ESP)
  2. Restore *callee-save* registers
  1. Leave return value in EAX

Which of the callee prologue steps could be done by caller?

To work with library code, compiler-generated code, must follow all rules carefully

If you don't care about this, which rules must you follow?

Byte Code Wizardry Awards

Mystery.java

- Implementation of SHA-1 cryptographic hash algorithm
  - Code posted on CS216 website now

```
private static int obfuscate(int num, int cnt) {
    int a = num; int b = cnt;
    for (int i = 0; i < 216; i += 3) {
        b = b + a + i; b = b - b; }
    if (b == 0) {
        return num;
    } else {
        for (int i = 0; i < 256; i++) { a = a - 1; }
        return a;
    } }
```


"Black Hat" Honorable Mentions

Call the Tester.generate() method

Richard Hsu, Dan Chen	244
Michael Thomas, Billy Chantree	244
Mike Liu	316
Jongmin Kim	380

Fastest Code

- Not enough variance to distinguish
- Effectively tied for fastest:
  - Bin Zhou, Mark Mitchell
  - Bijan Vakili
  - David Trang
  - Jake Fowler and Sean Talts
  - Michael Thomas and Billy Chantree
  - Eric Bradbury and Robert Yip

Smallest Code

Original Mystery.class	1547 bytes
Removing obfuscate	1300 bytes
Removing line numbers	1021 bytes
Improving logic	1015 bytes
Replacing operations with calls to Integer.rotateLeft	1061 bytes
Being the Byte Code Wizard	priceless

Smallest Code

Jake Fowler and Sean Talts	1015 bytes
Michael Thomas and Billy Chantree	1021 bytes
Will Ashford and Mike Dietz	1061 bytes

No others below 1300 bytes

Prizes

- "Byte Code Wizards": Jake Fowler and Sean Talts
- Selection of books (members of 3 teams pick in order)
  - John Battelle, "The Search" (Google)
  - Richard Feynman, "All the Adventures..."
  - Paul Graham, "Hackers & Painters"
  - Greg Hoglung & McGraw, "Exploiting Software"
  - Gary McGraw, "Software Security"
  - Simon Singh, "The Code Book"
- Black Hat awards: Sumatra (Java's big sister island) Coffee

Charge

- PS7 Due Monday
  - Lots of interesting things to explore
  - Don't wait to make progress on it
- "x86 Guru" title(s) may be awarded for especially clever answers to Question 10