ECE 391 Discussion Week 2

Announcements & Reminders

- MP0 deadline

  Due on Wednesday Aug 1st (today) 11:59 pm in Office hours
- PS1
  - Due 5:59 pm next Tuesday Sep 6th (gitlab)
  - Only one person in group must submit with a proper partners.txt – RTDC!
  - Work in groups of at least 4
  - Make sure you test your solutions
- MP1
  - Will be posted soon…
  - Start early

Problem Set 1

- C and assembly do NOT have to have a 1-to-1 correspondence
- Where to look up x86 assembly instructions (GAS or AT&T Syntax)
  - https://courses.engr.illinois.edu/ece391/secure/references/doc-x86-asm.pdf
  - http://en.wikibooks.org/wiki/X86_Assembly
- Important things to remember
  - Initialize register before using it (xorl %eax, %eax)
  - Dollar sign "$" in front of immediate number (movl $5, %eax)
  - Star "*" is not dereference in x86 assembly. Instead, it indicates a function pointer
- Other stuff
  - "extern" in C
  - .GLOBAL or .GLOBL in x86 assembly

Function Pointers

- What does “typedef” do
  - typedef char int8_t;
  - typedef char * int8ptr_t;
- What is a “function prototype”
  - int func (int arg);
- Example of a function pointer in C

```
typedef int (*func_t) (int arg);
func_t my_func;
int foo (int arg);
my_func = foo;
```

Displacement

- displacement(base register, offset register, scalar multiplier)
- displacement + base register + (offset register * scalar multiplier)
  - Base register can be any register
  - Default value of displacement is 0
  - Offset register can NOT be esp
  - Scalar multiplier can be 1,2,4,8, default value is 1
- Example of displacement
  - movl -4(%ebp, %edx, 4), %eax
  - eax ← M[ebp – 4 + edx * 4]

Jump Table

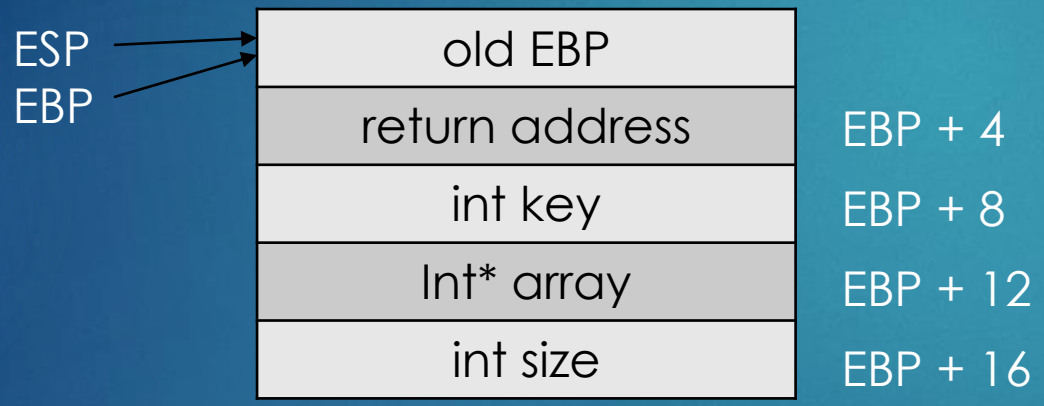
- Jump to the function whose address is given by a value
- Example of jump table in x86 assembly
  - jmp *operations-4(, %ecx, 4)
  - Remember "*" here is not for dereference
  - operations-4(, %ecx, 4) = M[operations – 4 + ECX * 4]

C Calling Convention

- Rules for C's subroutine interface structure
  - How information is passed to a subroutine
  - How information is returned to the caller
  - Who owns (responsible for) which registers
- Understand Stack structure (be careful of diagrams online!)
  - Lower/Higher memory addresses toward top/bottom of stack, respectively
  - Push/Pop instruction decrements/increments ESP, respectively
- Caller sequence
- Callee sequence

C Calling Convention

int binary_search(int key, int* array, int size);



Caller Saved	Callee Saved
EAX	EBX
ECX	ESI
EDX	EDI