# 15-213 Recitation
# Bomblab (Part 2)

Your TAs

Friday, September 12th

## Reminders

- `bomblab` is due on ***Tuesday (Sept 16th)***.
- `attacklab` will be released on the same day.
- Bootcamp 2: *Debugging & GDB* was pre-recorded, and is available on Ed (#158).

## Important

- Please do **NOT** submit a submission.tar (*or anything else*) directly to the autolab website. All submissions for **`bomblab`** are done automatically.

## Agenda

- `bomblab` demo
- `switch` statements and jump tables
- `bomblab` activity!

## `bomblab` Demo

## `bomblab` Demo/Activity

- Today, we'll be defusing phases in a *recitation-specific* bomb.
- Format is very similar to real **`bomblab`**
  - But explosions **won't** notify Autolab, or cost you points!
- Goal is to learn *techniques and concepts* rather than go through **`bomblab`** answers.
  - Don't worry about writing everything down
  - Don't worry if you don't finish all of the phases

## `bomblab` Demo

***Getting Started***

- Download today's activity handout from the *Schedule* page
- Also download the bomb
- Please use the **Shark Machines** to work on the bomb
- From there, hang tight. We'll be starting with a demo!

```
$ wget http://www.cs.cmu.edu/~213/activities/f25-rec3.tar
$ tar -xvpf f25-rec3.tar
$ cd f25-rec3
$ gdb bomb
```

## Demo: Phase A

## `switch` Statements and Jump Tables

# Recall: `switch` statements

```
int main() {
      int x;
      scanf("%d", &x);

      switch (x) {
            case 15205:
                  x += 1;
                  break;
            case 15206:
                  x += 5;
                  /* Fall through */
            case 15207:
                 x += 2;
                  break;
            case 15208:
                  x += 1;
                  break;
            case 15209:
                  x += 3;
                  break;
            case 15213:
                  x += 1;
                  break;
            default:
                  x = 0;
                  break;
            }

      return x;
}
```

Branch on an integer value

Fall through

Can have “holes”. No case for 15210!

## Recall: Jump Tables

- Compiler decides how to translate **`switch`** based on *heuristics*, for example:
  - Number of cases
  - Sparsity of cases
- Transform the input so we can use it to index into a table of addresses.
- Then just jump to the address at that index.
- ***Idea:*** runtime of **`switch`** becomes independent of the number of cases

## Jump Table Assembly: Case 1

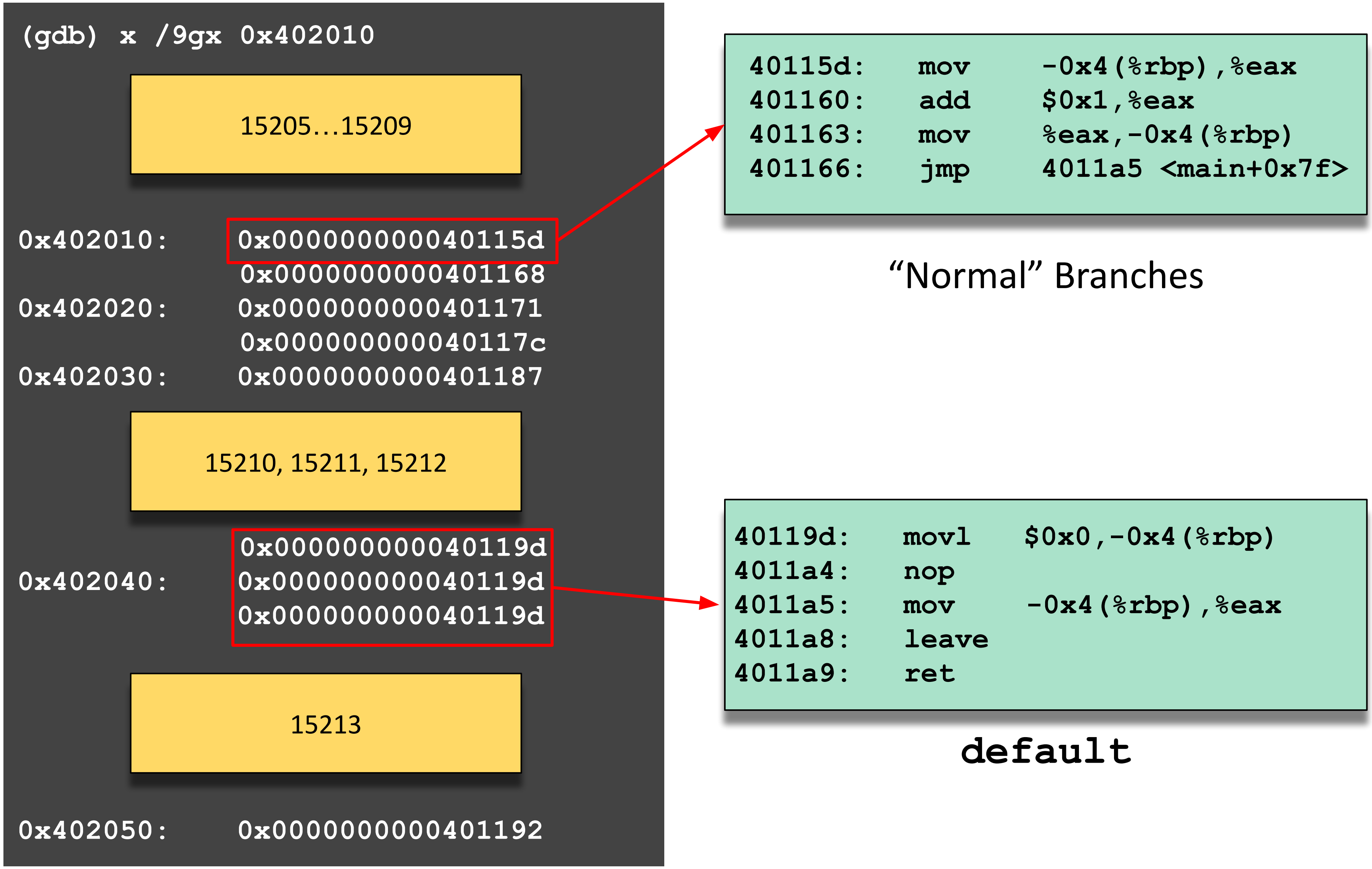
Shift range (**15205…15213**) to use zero-based indexing (**0x3b65** = **15205**)

```
401147:    2d 65 3b 00 00          sub  $0x3b65,%eax
40114c:    83 f8 08                cmp  $0x8,%eax
40114f:    77 4c                   ja   40119d <main+0x77>
401151:    89 c0                   mov  %eax,%eax
401153:    48 8b 04 c5 10 20 40    mov  0x402010(,%rax,8),%rax
40115a:    00
40115b:    ff e0                   jmp  *%rax
```

- Jump to default case
- Unsigned comparison is on purpose!

- Grab address from jump table
- Do an indirect jump to that address

# Jump Table Assembly: Case 1



```
(gdb) x /9gx 0x402010
```

```
0x402010:      0x000000000040115d
               0x0000000000401168
0x402020:      0x0000000000401171
               0x000000000040117c
0x402030:      0x0000000000401187
```

```
               0x000000000040119d
0x402040:      0x000000000040119d
               0x000000000040119d
```

```
0x402050:      0x0000000000401192
```

```
40115d:    mov     -0x4(%rbp),%eax
401160:    add     $0x1,%eax
401163:    mov     %eax,-0x4(%rbp)
401166:    jmp     4011a5 <main+0x7f>
```

“Normal” Branches

```
40119d:    movl    $0x0,-0x4(%rbp)
4011a4:    nop
4011a5:    mov     -0x4(%rbp),%eax
4011a8:    leave
4011a9:    ret
```

**`default`**

# Jump Table Assembly: Case 2

```
401151:   cmp  $0x8,%eax
401154:   ja   4011b8 <main+0x92>
401156:   mov  %eax,%eax
401158:   lea  0x0(,%rax,4),%rdx
40115f:   00
401160:   lea  0xea1(%rip),%rax       # 402008
401167:   mov  (%rdx,%rax,1),%eax
40116a:   cltq
40116c:   lea  0xe95(%rip),%rdx       # 402008
401173:   add  %rdx,%rax
401176:   jmp  *%rax
```

- **`rdx = index * 4`** (table stores 4 byte offsets)
- **`rax`** = table base address
- **`eax`** = offset stored at **`table[index]`**
- **`rdx`** = table base address
- Final address = (table base address) + (offset at **`table[index]`**)

## Jump Table Assembly: Case 2

*Final Address* = *Base Table Address* + *table[index]*

```
(gdb) x /9wx 0x402008
0x402008:      0xffff170    0xffff17b    0xffff184    0xffff18f
0x402018:      0xffff19a    0xffff1b0    0xffff1b0    0xffff1b0
0x402028:      0xffff1a5
(gdb) print /x 0x402008 + 0xffff170
$1 = 0x401178
```

```
...
401178:    8b 45 fc                mov  -0x4(%rbp),%eax
40117b:    83 c0 01                add  $0x1,%eax
40117e:    89 45 fc                mov  %eax,-0x4(%rbp)
...
```

Yep, that's one of our instructions!

## Activity: Phases B+C

## Phase B: Your Turn

- Now it's your turn!
- Take a few minutes to try to
  1. Write pseudocode for the phase
  2. Find an input string to defuse this phase
- Don't guess!
  - Be methodical: use the techniques we've learned
  - Reason about the code before jumping into **`gdb`**
  - This will be useful for the later phases of **`bomblab`**!

## Phase C

- Based on what you’ve learned, try to defuse Phase C!
- Once again, focus first on getting a psuedocode sketch of the phase!

## Phase C Tips

- What should the input for this function look like?
  - Hint: Recall how sscanf works and what info we can tell from it
- Given the second call to compare, what can our inputs be?
- Given the above, what does our jump table look like?
  - Hint: You are given the address of the jump table and the size of an element in the table

## Phase C Tips

- Based on your input, how is %eax modified?
  - What is this value multiplied by?
- Given all this, how do we access into sharkNames and what is the size of each field?
- What string do we want to find from sharkNames and which input will allow us to do this?
  - Hint: Refer back to the jump table. Which field is the one we want? How do we get %eax to equal this?