# 15-213 Recitation
# Final Exam Review

Your TAs

Friday, December 6th

# Reminders

- **`sfslab`** was due *yesterday*
  - Can use at most 1 grace day
- Code Reviews:
  - Meetings are optional for **`proxylab`**
  - **`sfslab`** graded for style, but no meetings
- ***Final Exam Review Session***
  - Dec 8th, 1-3PM
  - GHC 4401 (Rashid)
- ***Final Exam***
  - Thursday December 12, 8:30-11:30AM
  - DH 2210, DH 2315, DH 2302, DH 2105, DH 2122
  - Watch Piazza for details!

# TA Feedback

- Leave feedback for your TAs!
  - https://www.ugrad.cs.cmu.edu/ta/F24/feedback/
  - Staff photos are on course website.
- All feedback is welcome!
  - Feel free to rant, or give suggestions.

# Agenda

- **Final Exam Resources**
- **Final Exam Practice**
  - *Today:*
    - Assembly
    - Memory Layout
  - Bootcamp will cover more topics!

# Final Exam Resources

*Piazza @1383*

- **Lecture/Recitation Slides**
- **Labs:** what are the key concepts you learned?
- **Textbook:** especially useful for second half of the course
- **Written Assignments**
- **Past Final Exams**
- **Practice Final** on Gradescope
  - Solutions are on Piazza.

# Assembly

# Assembly

- Common exam questions:
  - Fill in the blanks of a C code snippet given assembly
  - Given assembly, determine the behavior of a function (like in **`bomblab`**!)
- Make sure to brush up on assembly!
  - Addressing modes: e.g., **`D(Rb, Ri, S)`**
  - Control Flow: **`cmp`** and condition codes
  - Calling Conventions
  - Good thing to include on your cheat sheet

# Assembly: FITB

*S11 Q3*

```
typedef struct node {
     void *data;
     struct node *next;
} node_t;

node_t *fun(node_t *n, int f(node_t *)) {
     node_t *a, *b;
     if (_____________________) {
          return NULL;
     }

     a = _____________________;

     if (_____________________) {
          b = _____________________;
          b->data = n->data;
          b->next = _____________________;
          return b;
     }

     return _____________________;
}
```

```
114d: push   %r12
114f: push   %rbp
1150: push   %rbx
1151: mov %rdi,%rbx
1154: test   %rdi,%rdi
1157: je  1192 <fun+0x49>
1159: mov %rsi,%rbp
115c: mov 0x8(%rdi),%rdi
1160: call   1149 <fun>
1165: mov %rax,%r12
1168: mov %rbx,%rdi
116b: call   *%rbp
116d: test   %eax,%eax
116f: jg  1179 <fun+0x30>
1171: mov %r12,%rax
1174: pop %rbx
1175: pop %rbp
1176: pop %r12
1178: ret
1179: mov $0x10,%edi
117e: call   1050 <malloc@plt>
1183: mov (%rbx),%rdx
1186: mov %rdx,(%rax)
1189: mov %r12,0x8(%rax)
118d: mov %rax,%r12
1190: jmp 1171 <fun+0x28>
1192: mov %rdi,%r12
1195: jmp 1171 <fun+0x28>
```

# Assembly: FITB Solution

*S11 Q3*

```
typedef struct node {
     void *data;
     struct node *next;
} node_t;

node_t *fun(node_t *n, int f(node_t *)) {
     node_t *a, *b;
     if (n == NULL) {
          return NULL;
     }

     a = fun(n->next, f);

     if (f(n) > 0) {
          b = malloc(16);
          b->data = n->data;
          b->next = a;
          return b;
     }

     return a;
}
```

```
114d: push   %r12
114f: push   %rbp
1150: push   %rbx
1151: mov %rdi,%rbx
1154: test   %rdi,%rdi
1157: je  1192 <fun+0x49>
1159: mov %rsi,%rbp
115c: mov 0x8(%rdi),%rdi
1160: call   1149 <fun>
1165: mov %rax,%r12
1168: mov %rbx,%rdi
116b: call   *%rbp
116d: test   %eax,%eax
116f: jg  1179 <fun+0x30>
1171: mov %r12,%rax
1174: pop %rbx
1175: pop %rbp
1176: pop %r12
1178: ret
1179: mov $0x10,%edi
117e: call   1050 <malloc@plt>
1183: mov (%rbx),%rdx
1186: mov %rdx,(%rax)
1189: mov %r12,0x8(%rax)
118d: mov %rax,%r12
1190: jmp 1171 <fun+0x28>
1192: mov %rdi,%r12
1195: jmp 1171 <fun+0x28>
```

# Assembly: Reasoning over Code

```
f:
        leal    1(%rdi), %ecx
        movl    $0, %edx
.L5:
        leal    -1(%rcx), %eax
        cmpl    %edx, %eax
        je      .L8
        leal    (%rdx,%rcx), %eax
        shrl    %eax
        movl    %eax, %esi
        imull   %eax, %esi
        cmpl    %esi, %edi
        jb      .L9
        movl    %eax, %edx
        jmp     .L5
.L9:
        movl    %eax, %ecx
        jmp     .L5
.L8:
        movl    %edx, %eax
        ret
```

- You are given the assembly for a mystery function **`f`**.
- Assume that **`f`** takes in a single unsigned integer **`y`** as its argument.
- For what *range* of inputs does **`f`** return **`8`**?

# Solution

- We might translate the assembly code to C code as follows:

```
c = y + 1;
d = 0;

while (d != c - 1) {
    a = (c + d) / 2;

    if (y < a * a) {
        c = a;
    } else {
        d = a;
    }
}

return d;
```

- This computes the integer square root of a number, so the correct range is 64-80.

*Source*: Wikipedia

# Memory Layout

# Memory Layout

Here is an array with 6 elements:

```
short arr[] = {0x1234, 0x8326, 0x9742, 0x4200, 0x1521, 0x3531};
```

Given the array's starting address is **0x7fffffffdf86**, what do these statements print? Justify your answers.

```
printf("2: %lx\n", *((long*)&arr[2]));
printf("3: %x\n", (unsigned char) *((char*)&arr[3]));
printf("4: %x\n", (unsigned char) *(((char*)arr)+ 3));
printf("5: %x\n", ((int*)arr)[1]);
```

# Answers

```
printf("2: %lx\n", *((long*)&arr[2]));

printf("3: %x\n", (unsigned char)*((char*)&arr[3]));

printf("4: %x\n", (unsigned char) *(((char*)arr)+ 3));

printf("5: %x\n", ((int*)arr)[1]);
```

2: 3531152142009742

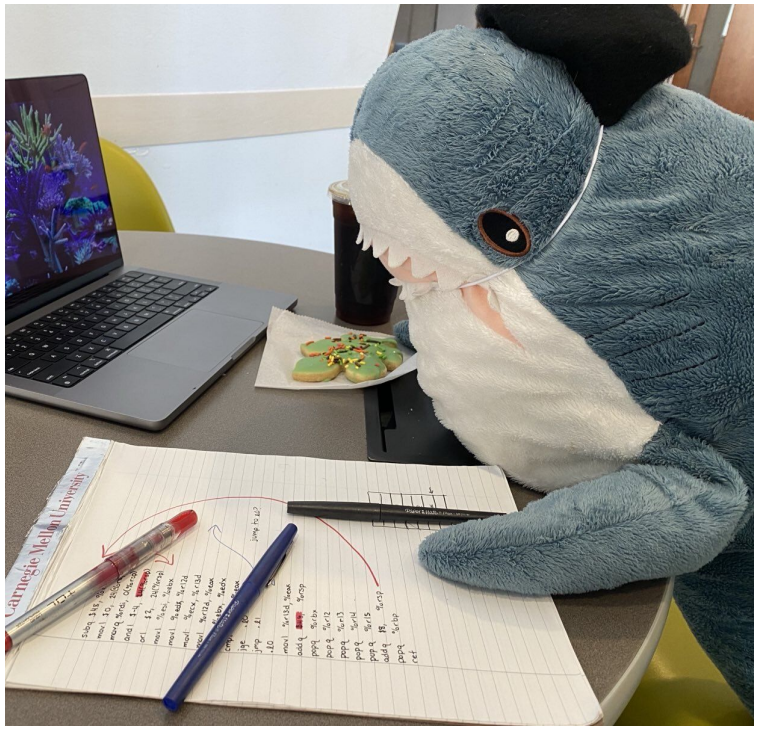
3: 0

4: 83

5: 42009742

# Wrapping Up

- ***Final Exam Review Session***
  - Dec 8th, 1-3PM
  - GHC 4401 (Rashid)
- ***Final Exam***
  - Thursday December 12, 8:30-11:30AM
  - Watch Piazza for details!
- Please leave feedback for your TAs!
- Good luck for Finals Week :-)



# The End