Lecture 8

<2016-04-27 Wed>

Contents

1 Memory Layout **1**

2 Buffer Overflow **2**

3 Float **2**

3.1 Fractional Binary Numbers 2

3.1.1 example 3

3.1.2 limitations 3

3.2 Floating Point Representation (IEEE Standard) 3

3.3 Normalized Values 3

3.3.1 example 3

1 Memory Layout

- stack
  - runtime stack (8MB limit)
- heap
  - dynamically allocated
  - `malloc`, `calloc`, `new`
- data
- text / shared library

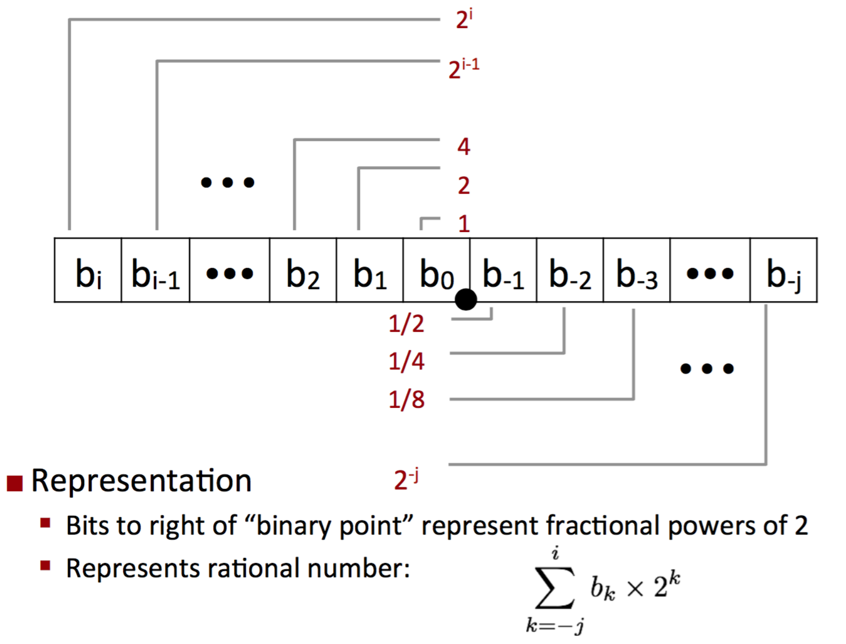
2 Buffer Overflow

```
typedef struct {
  int a[2];
  double d;
} struct_t;

double fun(int i) {
  volatile struct_t s;
  s.d = 3.14;
  s.a[i] = 1073741824;
  return s.d;
}
```

3 Float

3.1 Fractional Binary Numbers



- Representation
  - Bits to right of "binary point" represent fractional powers of 2
  - Represents rational number: $$\sum_{k=-j}^{i} b_k \times 2^k$$

- bits to right of 'binary point' represent fractional powers of 2
- representation of rational numbers $\sum_{k=-j}^{i} b_k \times 2^k$

3.1.1 example

```
value     |   representation
----------------------------
5 + 3/4   ==>  101.11
2 + 7/8   ==>   10.111
1 + 7/16  ==>    1.0111
```

- observations
  - divide by 2 by shifting right (unsigned)
  - multiply by 2 by shifting left
  - number of the form 0.11111_2 are just below 1.0
    - $\sum \frac{1}{2^i}$ goes to 1.0
    - use notation 1.0 - ϵ

3.1.2 limitations

- can only reprsent numbers of the form x/2^{k}

3.2 Floating Point Representation (IEEE Standard)

- numerical form $(\text{-}1)^{\mathrm{s}}$ M 2^{E}
  - sign bit: `s`
  - significand: `M`
  - exponent: `E`

3.3 Normalized Values

- when $exp \neq$ 00...0 and $exp \neq$ 11...1

3.3.1 example

15213_{10}

- as an integer 11101101101101_2
- as a float $1.1101101101101_2 \times 2^{13}$
  - significand

    - $\texttt{M} = 1.1101101101101_2$
    - $\texttt{frac} = 11011011011010000000000_2$
- exponent
    - $\texttt{E} = 13$
    - $\texttt{Bias} = 127$
    - $\texttt{Exp} = 140 = 10001100_2$
- result
    - 0 10001100 11011011011010000000000