CSci 2021: Review Lecture 1
Stephen McCamant
University of Minnesota, Computer Science & Engineering

Midterm 1 topics (in one slide)

- The C language
  - Functions, variables, and types
  - Branches and loops
  - Arrays, pointers, and structures
- Number representation
  - Bits and bitwise operators
  - Unsigned and signed integers
  - Floating point numbers
- Machine-level code representation
  - Instructions, operands, flags
  - Conditions and branches

Outline

C language topics
- Topics in number representation
- Exam logistics
- Number representation problem
- Topics in machine code
- Machine code problems

C compared to other languages

- Predecessor of C++, Java, other more modern languages
- No objects, for instance functions and no methods
- Most features have a direct translation to machine code

C numeric types

- char, short, int, and long are 8, 16, 32, or 64 bits on x86-64
- Unsigned integers are ≥ 0
- Mixed operands upgraded to larger size and unsigned
- float and double are 32-bit and 64-bit floating point

Kinds of variables and allocation

- Local variables exist in one function execution, and go away when it is over
  - Even if you think you have a pointer to it!
- Global variables can be accessed from any function, and last for the whole program
- For more control, allocate memory with malloc and get a pointer
C strings

- Instead of a real string type, C programs pass pointers to characters.
- Usually, length of string is indicated by a `\0` terminator.
- Transform strings by writing loops over characters.
- Programmer needs to be explicit about allocation and sharing.

C pointers

- Pointers hold addresses, and the compiler knows their type.
- Create a pointer to a variable with `&`.
- Dereference a pointer with `*`.
- Pointer arithmetic uses the element size, like an array.
- In fact, `a[x]` is the same as `*(a + x)`.

More about pointers

- Pointer parameters implement pass by reference.
- The null pointer doesn’t point at anything.
  - So don’t dereference it.
- When using pointers, pay attention to data lifetime and sharing.

C structures

- A `struct` groups several related values together.
  - Similar to objects with features removed.
- Commonly structs are accessed with pointers, fields with `->`.
  - For instance, to implement linked lists and trees.
- `malloc` with the structure size is like `new`.

For instance, HA1 hashtable

- Several possible designs:
  - Array of pointers to list nodes.
  - Array of root structures pointing at list nodes.
  - Array of first list nodes (insert second).
- Choices for string storage:
  - Struct has char array, `strcpy`.
  - Struct has char pointer, `strdup`.

Outline

- C language topics
- Topics in number representation
- Exam logistics
- Number representation problem
- Topics in machine code
- Machine code problems
Bits and bitwise operations

- Base 2 (binary) and base 16 (hex) generalize from base 10 (decimal)
- And, or, xor, not
- Left shift, two kinds of right shift
  - Similarity to multiply/divide by $2^k$

Unsigned and signed integers

- Unsigned: plain base 2, non-negative
  - Overflow is like operations modulo $2^n$
- Signed: two’s complement with a sign bit
  - Sign bit counts for negative place value
  - Overflow possible in both directions
- Comparing the two
  - Ranges partially overlap
  - $+, -, \times$ (same size output), $<, ==$, narrowing are the same
  - $\div, \%, \gg, <, >$ (high output bits), and widening are different
- Algebra properties exist despite overflow

Floating point numbers

- Represent fractions and larger numbers using binary scientific notation
- Fractions whose denominator is a power of two
  - All others must be rounded
  - Limited precision gradually loses information
- Rounding: examine thrown-away bits
- Special cases for $\pm 0, \pm \infty, NaN$
- Ordering properties but fewer algebraic properties

Normalized and denormalized

- All but the smallest finite numbers are normalized
  - Represent as $1 \cdot 2^e$
  - (Leading 1 is not stored)
- For smallest numbers, special denormalized form
  - Smallest $exp$ encoding: same $E$ as smallest normal
  - Leading 0 is not stored

Outline

- C language topics
- Topics in number representation
- Exam logistics
- Number representation problem
- Topics in machine code
- Machine code problems

Exam rules

- Begins promptly at 3:35, ends promptly at 4:25
- Open-book, open-notes, any paper materials OK
- No electronics: no laptops, smartphones, calculators, etc.
  - Arithmetic will use easy numbers
- Sit in alternating seats as long as possible
Exam strategy suggestions

- Writing implement: mechanical pencil plus good eraser
- Make a summary sheet to save flipping through notes or textbook
- Show your work when possible
- Do the easiest questions first
- Allow time to answer every question

Outline

C language topics
- Topics in number representation
- Exam logistics
  - Number representation problem
- Topics in machine code
  - Machine code problems

Overflow

Which of these combinations can describe the addition of the same bits? If possible, give an example with 4-bit ints.

- No unsigned OF, no signed OF: 0000 + 0000 = 0000
- Unsigned OF, no signed OF: 1111 + 0001 = 0000
- Unsigned OF, positive OF: 1111 + 0001 = 0000
- Unsigned OF, negative OF: can’t happen
- No unsigned OF, positive OF: 1111 + 0001 = 0000
- No unsigned OF, negative OF: 1111 + 0001 = 0000

https://chimein.cla.umn.edu/course/view/2021
Which of these combinations can describe the addition of the same bits? If possible, give an example with 4-bit ints.

- No unsigned OF, no signed OF: 0000 + 0000 = 0000
- Unsigned OF, no signed OF: 1111 + 0001 = 0000
- Unsigned OF, positive OF: can't happen
- Unsigned OF, negative OF: 1000 + 1000 = 0000
- No unsigned OF, positive OF:
- No unsigned OF, negative OF:

https://chimein.cla.umn.edu/course/view/2021

C language topics
Topics in number representation
Exam logistics
Number representation problem
Topics in machine code
Machine code problems

Assembly language ↦ machine code
Sequence of instructions, encoded in bytes
An instruction reads from or writes to operands
- x86: usually at most one memory operand
- AT&T: destination is last operand
- AT&T shows operand size with b/w/l/q suffix

General form: disp(base,index, scale)
- Displacement is any constant, scale is 1, 2, 4 or 8
- Base and index are registers
- Formula: mem[disp + base + index * scale]

All but base are optional
- Missing displacement or index: 0
- Missing scale: 1
- Drop trailing (but not leading) commas

Do same computation, just put address in register: lea
Flags and branches

Flags (aka condition codes) are set based on results of arithmetic:
- ZF: result is zero
- SF: result is negative (highest bit set)
- OF: signed overflow occurred
- CF: unsigned overflow ("carry") occurred

Used for condition in:
- setCC: store 1 or 0
- cmovCC: copy or don't copy
- jCC: jump or don't jump

Just for setting flags: cmp (like sub), test (like and)

Outline

C language topics
Topics in number representation
Exam logistics
Number representation problem
Topics in machine code
Machine code problems

Working with ordering

Which of these conditions are the same?

<table>
<thead>
<tr>
<th>Col. 1</th>
<th>Col. 2</th>
<th>Col. 3</th>
<th>Col. 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A: x &lt; y</td>
<td>B: x &gt; y</td>
<td>C: x &lt;= y</td>
<td>D: x &gt;= y</td>
</tr>
<tr>
<td>y &lt; x</td>
<td>y &lt;= y</td>
<td>y &gt; x</td>
<td>y &gt;= x</td>
</tr>
<tr>
<td>!(x &lt; y)</td>
<td>!(x &gt; y)</td>
<td>!(x &lt;= y)</td>
<td>!(x &gt;= y)</td>
</tr>
<tr>
<td>!(y &lt; x)</td>
<td>!(y &gt; x)</td>
<td>!(y &lt;= x)</td>
<td>!(y &gt;= x)</td>
</tr>
</tbody>
</table>

https://chimein.cla.umn.edu/course/view/2021