CS61C: Review of Stored Program Computing

CS61C Fall2007 - Discussion #6

Greg Gibeling

Stored Program Computer

- Binary Data
  - Numbers: integer & floating point
    - ±2Gi for integers, $2*10^{\pm 38}$ for float (single precision)
    - 4Gi values either way in 32b
  - Characters: ASCII & Unicode
    - An assignment from numbers to symbols
  - Pointers: an integer with meaning
  - Instructions: MIPS, IA32, SBN
    - Perform operations on data
- The meaning of data depends on it's interpretation
  - In C we specify a type for all data
  - In MIPS we don't know the type apriori
    - For example the format specifier tells us the type

Java -> C

- C to Java: A rough transition
  - "." in Java becomes "->" in C
  - Both turn into `lw/sw` in MIPS
- Pointer vs Memory
  - A pointer is different than the memory it points to
    - E.g. strings are all in your imagination in C
    - Can someone tell me how to free a string?
  - You can store pointers in memory
    - Leads to complex `lw/sw` sequences, lab5ex1
- Buffer Overflows

```
void foo(char* string) {
   int length = strlen(string);
   char* buffer = (char*)malloc((length+1)*sizeof(char));
   strncpy(buffer, string, length);
   buffer[length] = '\0';
   // etc…
}
```


Assembly Labels

- Example:

```
N: .word 0
```

  - N: A constant number, the address of the word
  - .word: Allocate a word's worth of storage
  - 0: fill the allocated storage with 0
- Basic Use

```
lw $t0, 25($t1) # $t0 = *($t1 + 25)
sw $t0, 25($t1) # *($t1 + 25) = $t0
```

- Advanced Uses

```
sw $s0, 10+N($a1)
```

  - 10+N is a constant computed by the assembler

```
sw $s0, N # sw $s0, N($0)
```

  - Simple shorthand (useable only because of $0!)
- PsuedoInstructions

```
la $t0, N # lui + ori as needed
li $t0, N # should be the same as la…
```


Width & Meaning

- C & Java have variables
  - Logical or virtual storage
  - Can be stored in memory or a register
  - Variable is just a name for a value
- Assembly has locations
  - Physical storage
  - Locations can be named (registers & labels)
  - The value in a location can change, the location cannot
- Register Coloring
  - Deciding which variables go in which locations
  - Similar to 4-color theorem. See CS164

		Meaning	
		Known	Unknown
Width	Known	Java	Assembly
	Unknown	C	

All Kinds of Zeros

- Not my IQ
- Kinds of Zeros
  - `NULL` – for pointers
  - `0` – for integers
  - `0.0` – for floating point
  - `'\0'` – for characters
- Why
  - So that your code is readable
  - `NULL` might not always be zero!
  - **These constants specify value & type**

MIPS Instruction Design

- Performance
  - CISC: Bad, RISC: Good, SBN: Very Bad
  - Why is the sweet spot in the middle at RISC?
  - So what?
- Load `0xDEADBEEF` into `$s0`
  - `lui $s0, 0xDEAD; ori $s0, $s0, 0xBEEF`
    - Any other ways to do this?
  - Why in two halves? Why not in one instruction?
    - Important point of design for RISC!
    - Relationship to stored program computer?
- Single Instruction: SBN (subtract branch if negative)
  - No need for registers, just use memory
  - "Universal Operator": NAND, Mux, etc…
- IA32/x86
  - 1-18 byte instructions (e.g. strcpy!)
  - But internally, executed as RISC

Belief & Debugging

- Belief: You believe you know what your program does
  - You think you understand it
  - You think you know what the library calls do
- Fact: You can read what it actually does
  - Computers are as close to perfect as possible
  - A computer error or fault is very unlikely
- Consequence
  - A mismatch means your beliefs are wrong
  - Always assume that you are **dead wrong**
    - It's possible the bug is a typo

Notes on Floating Point

- IEEE 754 Standard
  - C/Java/MIPS/IA32 doesn't make any difference
  - Result of work by Dr. Kahan at Berkeley
  - Uses a register stack & flags
    - Hard to implement (just the flags & registers)
    - Painfully hard to parallelize & virtualize
      - Flags are bad
  - Made scientific computation reliable
    - Famous 1.0*x == 1.0 bug in cray machines
- Fields
  - Sign (1 means negative)
  - Exponent
    - Biased by $2^{e-1}-1$)
    - Aligns lexical and numerical ordering
  - Mantissa (Implied 1, unless denormal!)

Midterm Questions

- Design a CS61C Midterm1
  - Cover material through 10/5 only
  - You can write questions, solutions, outlines, whatever you want
  - Groups of no more than 4
  - I'll answer questions on anything class related