CSI333 Lecture 1 Instructor’s Notes and Slides

Seth Chaiken

September 7, 1999

Course Coverage

1. Machine/Assembly Language Programming: A fresh start, plus C++ related to machine operation or for review.

2. C++ hardware/software/system interface topics: type implementation, modularization with multiple source file programs, pointers, strings, arguments, preprocessor, etc.
House Architecture One

Structure
- ROOF
- FLOORs
- WALLS

Function
- Provide sheltered living spaces for people.

Computer Architecture One

Structure
- PROCESSOR (CPU)
- MEMORY (RAM)
- INPUT/OUTPUT (I/O)

Function
- Communicate, store and process digitally represented data under digitally stored program control.
Fetch-execute cycle
What the processor does

```plaintext
do {
    (1) Fetch instruction. (from Memory)
    (2) Decode instruction.
    (3) Execute instruction.
} while (instruction is not "halt")
```

Each (RISC) machine instruction only does one

- arithmetic operation or
- test/control operation or
- memory/processor data transfer
  (with address calculation)
  at a time.
### Slide 6

<table>
<thead>
<tr>
<th>C++</th>
<th>MIPS Assembly Language</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int X, Y, Z;</code></td>
<td><code>lw $t0, Y</code></td>
</tr>
<tr>
<td><code>Z = X + Y*Z;</code></td>
<td><code>lw $t1, Z</code></td>
</tr>
<tr>
<td><code>if( Z &lt; 0 )</code></td>
<td><code>mul $t2, $t1, $t0</code></td>
</tr>
<tr>
<td>`{</td>
<td><code>#compute Y*Z</code></td>
</tr>
<tr>
<td><code>    Z = 0;</code></td>
<td><code>lw $t3, X</code></td>
</tr>
<tr>
<td><code>    }</code></td>
<td><code>add $t4, $t3, $t2</code></td>
</tr>
<tr>
<td></td>
<td><code>#compute X + Y*Z into $t4</code></td>
</tr>
<tr>
<td></td>
<td><code>bge $t4, $0, Label0One</code></td>
</tr>
<tr>
<td></td>
<td><code>li $t4, 0</code></td>
</tr>
<tr>
<td></td>
<td><code>#put 0 in $t4 if( Z &lt; 0 )</code></td>
</tr>
<tr>
<td></td>
<td><code>Label0One:</code></td>
</tr>
<tr>
<td></td>
<td><code>sw $t4, Z</code></td>
</tr>
</tbody>
</table>

### Slide 7

Lesson of Object Oriented Programming:
Understand the properties, meaning, and types of DATA is MORE IMPORTANT than algorithms and executable programs.
Slide 8

```c
int X, Y, Z;
Z = X + Y*Z;
if( Z < 0 )
{
    Z = 0;
}
```

Variables

X

Y

Z

store values and
all look the same in C/C++

Slide 9

In **Instruction Set Architecture** (machine language)
there are three facilities for “remembering” data:

- Data registers (called **Registers** for short).
- RAM (called **Memory** for short).
- Some I/O devices (disks, etc.)

They have different **performance** (i.e. speed)
properties and are programmed differently.
Slide 10

Registers
Located in Processor
Identified by number: 0, 1, ..., 31
(also by software convention register names: $zero, $at, $v0-$v1, $a0-$a3, $t0-$t9, $s0-$s7, $k0-$k1, $gp, $sp, $fp, $ra
$0=$zero and $ra=$31 have special hardware behavior.)
Size of each: 32 bits
Few in number (31 or 32)
Total storage space: 31 * 4 = 124 bytes
Access time: 2 Nanoseconds to access 3 registers
Expensive per bit

Slide 11

Memory
Located (mostly) outside the Processor
Names (called addresses): 0, 1, ..., $2^{32} - 1
Addresses locate byte (8 bit) sized chunks, but usually
one 4 byte word is accessed at a time.
Many in number
Late 90's PCs have tens of Megabytes to Gigabytes of
RAM
Roughly 20 ns (uncached), or more.
Cheap per bit
Slide 12

How to use

Registers
Code 2 or 3 reg. numbers in one mach. instruction:
add \$t0, \$t1, \$t2
(3 register add.)
addi \$t0, \$t1, 365
(add immediate constant 365 with \$t1, put sum in \$t0)

Memory
Make the processor use registers and immediate constants to compute the Memory Address.
Load and Store instructions move data between Memory and a register.

Slide 13

What C Pointer Values Are!

Interface to the Processor