Lecture 4

or “Why am I here?”
Announcements

- Homework: In textbook (pg 164)
  - Exercises
    - 2.3
    - 2.7
    - 2.9
    - 2.10
  - Find 2’s comp representation of 123 and -123
    (output is 32 bit word in hex format)
- Due Thursday, Feb. 5th in class
More Announcements

- Midterm will be on Thursday, Feb. 26.
- Open book, printed-out class notes, and homework solutions are allowed.
- No other materials or electronic devices are allowed.
Representing Instructions

- All instructions are encoded in binary
  - AKA machine code
- MIPS
  - Encoded as \(?\)-bit instruction words
  - Only a few formats for encoding the operation code (opcode), register numbers, etc
## Register numbers

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>Number</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>zero</td>
<td>0</td>
<td>Constant 0.</td>
</tr>
<tr>
<td>at</td>
<td>1</td>
<td>Reserved for the assembler.</td>
</tr>
<tr>
<td>v0 - v1</td>
<td>2 - 3</td>
<td>Result Registers.</td>
</tr>
<tr>
<td>a0 - a3</td>
<td>4 - 7</td>
<td>Argument Registers 1 ⋯ 4.</td>
</tr>
<tr>
<td>t0 - t9</td>
<td>8 - 15, 24 - 25</td>
<td>Temporary Registers 0 ⋯ 9.</td>
</tr>
<tr>
<td>s0 - s7</td>
<td>16 - 23</td>
<td>Saved Registers 0 ⋯ 7.</td>
</tr>
<tr>
<td>k0 - k1</td>
<td>26 - 27</td>
<td>Kernel Registers 0 ⋯ 1.</td>
</tr>
<tr>
<td>gp</td>
<td>28</td>
<td>Global Data Pointer.</td>
</tr>
<tr>
<td>sp</td>
<td>29</td>
<td>Stack Pointer.</td>
</tr>
<tr>
<td>fp</td>
<td>30</td>
<td>Frame Pointer.</td>
</tr>
<tr>
<td>ra</td>
<td>31</td>
<td>Return Address.</td>
</tr>
</tbody>
</table>
Instruction formats

Why do we need different formats?

- Allow for different operations to have the same number of bits
R-Type Instruction

- Register type: Operates on 3 registers
  - op: operation (6 bits)
  - rs: first source operand (5 bits)
  - rt: second source operand (5 bits)
  - rd: destination operand (5 bits)
  - shamt: shift amount (5 bits)
  - funct: variant of opcode (6 bits)
  - Syntax: <op> $rd, $rs, $rt

=32 bits
How do we use this?

- Let’s try to convert the following to machine code:
  - add $t0, $s1, $s2
The answer

- **add**
  - opcode = \(0_{16}\)
  - Funct = \(20_{16}\)
- **What are these in binary?**
  - \(00_{16} = 0000 \ 0000_2\) (remember we only need 6 bits)
    - \(000000_2\)
  - \(20_{16} = 0010 \ 0000_2\)
    - \(100000_2\)
Now that we have the opcode and the funct code, we need to fill in the middle

- \( t0 = 8 \quad = 01000 \)
- \( s1 = 17 \quad = 10001 \)
- \( s2 = 18 \quad = 10010 \)

Not a shift operation so shamt = 00000
add  $s1  $s2  $t0  shift 0  funct 32

To hex:

0  2  3  2  4  0  2  0
0x02324020
I-Type Instruction

- Immediate Type: Operate on 2 registers
  - op (opcode): 6 bits
  - rs (source): 5 bits
  - rt (dest): 5 bits
  - imm (const or address): 16 bits

- Syntax (2 forms)
  - <op> $rt, $rs, imm
  - <op> $rt, offset($rs)
addi $s0, $s1, 5
Example again

sw $s2, 8($t3)
Reverse reverse

0xad310004
Using I-type

We want to initialize a variable
Lets say we want to initialize to a 16 bit value
- Initialize $s0 to the 16 bit value: 0x0008
Using I-type

We want to initialize a variable

Let's say we want to initialize to a 16 bit value

- Initialize $s0 to the 16 bit value: 0x0008
  - addi $s0, $zero, 8
What about 32 bit

What if you want to initialize $s0 to the value: 0x11223344?

We can use the load upper immediate

This will load 16 bits into the upper 16 bits of a register

\[
\text{lui } \$t0, 0x1122 \\
\text{addi } \$t1, \$zero, 0x3344 \\
\text{or } \$t0, \$t0, \$t1
\]