“Open the pod-bay doors, HAL.”

Computer instruction from 2001: A Space Odyssey

Machine Instructions

Computers do exactly what we tell them to do. Instructions are the medium for all tasks -- arithmetic, logic, I/O. Though most ISAs have a unique instruction set, the similarities are greater than the differences, justifying our study of the MIPS ISA alone.
Binary Review

Decimal notation represents numbers by their coefficients of powers of 10. Binary expresses numbers using coefficients of powers of 2.

\[321_{10}\]

\[= 3 \times 10^2 + 2 \times 10^1 + 1 \times 10^0\]
\[= 300 + 20 + 1\]
\[= 256 + 64 + 1\]
\[= 1 \times 2^8 + 0 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0\]
\[= 101000001\]
Binary Facts

• Alternate forms of binary:
  • Octal: symbols 0-7 and is convertible from binary in groups of 3
  • Hexadecimal: 0-9, A,B,C,D,E,F and is convertible from binary in groups of 4

\[
101000001_2 = 101\ 000\ 001 = 501_8
\]
\[
= 1\ 0100\ 0001 = 141_{16}
\]

1011 1010 1101 = BAD

Fast Approximations

\[
2^{10} \approx 1,000
\]
\[
2^{20} \approx 1,000,000
\]
\[
2^{30} \approx 1,000,000,000
\]
\[
2^{40} \approx 1,000,000,000,000
\]
Memory Structure

- Computers have two basic types of memory:
  - Random Access Memory (RAM) for storing data and programs while they are being executed
  - Permanent memory for keeping programs and data when they are not be computed upon
    - Hard disks, floppy disks, CDROM, magnetic tape …

- RAM is one continuous sequence of bytes (8-bit units) each having a consecutive address

```
0  1  2  3  4  5  6 ...
```
Instruction Interpretation

- A computer is a hardware interpretation of the “fetch/execute” cycle

```
fetch an instruction from memory[PC]
de decode the instruction
access the operands of the instruction
perform (execute) the operation on the operands
placing the results in a register
PC := PC+1
```
Translating Instructions

- The instructions must have a form understandable to the follower. Computers only understand binary, so translations are needed to lower the level.

Programming Language Instruction (C)

```
i = i + 1;
```

Assembler

```
addi $8, $8, 1
```

Machine Language Instruction (MIPS ISA)

`0010 0001 0000 1000 0000 0000 0000 0001`
Components of Instruction Execution

Program Counter

Instruction execution begins with the program counter (PC) at location 1200:

1200: add $8,$8,$9
1204: addi $3,$8,1
1208: lw $8,0($19)
120C: slt $6,$6,$4

Registers

Data, stored in memory

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>19</th>
<th>30</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>9000</td>
<td>2</td>
<td>550</td>
<td>8200</td>
<td>73048</td>
<td>73040</td>
</tr>
<tr>
<td>8200: 71</td>
<td>8204: 550</td>
<td>8208: 0</td>
<td>820C: 282</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Anatomy of an Instruction

Operator and operands

- Operation -- assembly language uses prefix
- Operands separated by commas
- Comments follow #

```
add $8, $9, $10 # Reg8 = Reg9 + Reg10
```

- ISAs usually use a few instruction formats, each format customized to a particular type of instruction
R type Instruction Format

- “R” Type Instructions have all operands in registers
  - There are exactly 3 operands
  - Operands can only be register names
  - Values of operands are integers, with the default being signed integers

\[
\text{Reg7} = (\text{Reg8} - \text{Reg9}) + (\text{Reg6} - \text{Reg5})
\]

becomes

\[
\begin{align*}
\text{sub} & \quad \text{10}, \quad \text{8}, \quad \text{9} \\
\text{sub} & \quad \text{11}, \quad \text{6}, \quad \text{5} \\
\text{add} & \quad \text{7}, \quad \text{10}, \quad \text{11}
\end{align*}
\]
Encoding R-type Instructions

- All MIPS instructions are 1 word (4 bytes, 32 bits) long
- Partition word into fixed length fields
  - op = operation
  - rd = destination operand
  - rt = second source operand
  - rs = first source operand
  - shamt = shift amount
  - funct = function field

```
add $8, $9 $10  # no comment
```

```
<table>
<thead>
<tr>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>rd</th>
<th>shamt</th>
<th>funct</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>9</td>
<td>10</td>
<td>8</td>
<td>0</td>
<td>32</td>
</tr>
</tbody>
</table>

```

```
00000000100101010000100000000100000
```

© Larry Snyder, 2000 All rights reserved