MIPS Programs and qtSpim

CS321
Hello MIPS

High level

a = 1
b = a + 2

Low level
Hello MIPS

High level

a = 1
b = a + 2

Low level

addi $s0, $zero, 1
addi $s1, $s0, 2
ori: bitwise OR immediate
addi: add immediate
MIPS with array

High level

int[] A = { 1, 2, 3 }
a = 1
b = a + A[2]

Low level

The address of A[0] = 0x10010000
MIPS with array

**High level**

```plaintext
int[] A = { 1, 2, 3 }

a = 1

b = a + A[2]
```

The address of A[0] = 0x10010000

**Low level**

```plaintext
lui $t0, 0x1001

lw $t1, 8($t0)

ori $s0, $zero, 1

add $s1, $s0, $t1
```

lui: load upper immediate
MIPS with condition

High level

\[ a = 1 \]
\[ b = 1 \]
\[ \text{if } (a == b) \{ \]
\[ \quad b = a + 1 \]
\[ \}

Low level
MIPS with condition

High level

a = 1
b = 1
if (a == b) {
    b = a + 1
}

Low level

addi $s0, $zero, 1
addi $s1, $zero, 1
bne $s1, $s0, SKIP
addi $s1, $s0, 1
SKIP:

bne: branch on not equal
MIPS with condition

High level

a = 1
b = 2
if (a != b) {
    b = a
}

Low level
MIPS with condition

High level

a = 1
b = 2
if (a != b) {
    b = a
}

Low level

addi $s0, $zero, 1
addi $s1, $zero, 2
beq $s1, $s0, SKIP
add $s1, $s0, $zero
SKIP :

beq: branch on equal
MIPS with condition

High level

a = 1
b = 2
if (a < b) {
    b = a
}

Low level
MIPS with condition

High level

a = 1
b = 2
if (a < b) {
    b = a
}

Low level

addi $s0, $zero, 1
addi $s1, $zero, 2
slt $t0, $s0, $s1
beq $t0, $zero, SKIP
add $s1, $s0, $zero
SKIP :

slt: set on less than
MIPS with condition

High level

```plaintext
a = 1
b = 2
if (a <= b) {
    b = a
}
```

Low level
# MIPS with condition

<table>
<thead>
<tr>
<th>High level</th>
<th>Low level</th>
</tr>
</thead>
<tbody>
<tr>
<td>a = 1</td>
<td>addi $s0, $zero, 1</td>
</tr>
<tr>
<td>b = 2</td>
<td>addi $s1, $zero, 2</td>
</tr>
<tr>
<td>if (a &lt;= b) {</td>
<td>slt $t0, $s1, $s0</td>
</tr>
<tr>
<td>b = a</td>
<td>bne $t0, $zero, SKIP</td>
</tr>
<tr>
<td>}</td>
<td>add $s1, $s0, $zero</td>
</tr>
<tr>
<td></td>
<td>SKIP :</td>
</tr>
</tbody>
</table>
### Pseudo-Instructions

<table>
<thead>
<tr>
<th>Meaning</th>
<th>Instruction</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>branch on less than</td>
<td>blt $s0, $s1, [goto]</td>
<td>slt $t0, $s0, $s1</td>
</tr>
<tr>
<td></td>
<td>$s0 &lt; $s1 ? [goto] : null</td>
<td>bne $t0, $zero, [goto]</td>
</tr>
<tr>
<td>branch on less than or equal</td>
<td>ble $s0, $s1, [goto]</td>
<td>slt $t0, $s1, $s0</td>
</tr>
<tr>
<td></td>
<td>$s0 &lt;= $s1 ? [goto] : null</td>
<td>beq $t0, $zero, [goto]</td>
</tr>
<tr>
<td>branch on greater than</td>
<td>bgt $s0, $s1, [goto]</td>
<td>slt $t0, $s1, $s0</td>
</tr>
<tr>
<td></td>
<td>$s0 &gt; $s1 ? [goto] : null</td>
<td>bne $t0, $zero, [goto]</td>
</tr>
<tr>
<td>branch on greater than or equal</td>
<td>bge $s0, $s1, [goto]</td>
<td>slt $t0, $s0, $s1</td>
</tr>
<tr>
<td></td>
<td>$s0 &gt;= $s1 ? [goto] : null</td>
<td>beq $t0, $zero, [goto]</td>
</tr>
</tbody>
</table>
MIPS with while loop

High level

a = 5
while (a != 0) {
    a = a - 1
}

Low level
MIPS with while loop

High level

```plaintext
a = 5
while (a != 0) {
    a = a - 1
}
```

Low level

```plaintext
addi $s0, $zero, 5
LOOP:
beq $s0, $zero, DONE
addi $s0, $s0, -1
j LOOP
DONE:
```

```plaintext
j: jump
```
MIPS with for loop

High level

b = 5
for (a = 1; a < b; a++) {
    c = c + a
}

Low level
MIPS with for loop

High level

\[ b = 5 \]
\[
\text{for } (a = 1; a < b; a++) \{
\text{c = c + a}
\}
\]

Low level

\text{addi } $s0, \$zero, 1 \\
\text{addi } $s1, \$zero, 5 \\
\text{addi } $s2, \$zero, 0 \\
\text{LOOP:} \\
\text{bge } $s0, \$s1, \text{DONE} \\
\text{add } $s2, \$s2, \$s0 \\
\text{addi } $s0, \$s0, 1 \\
\text{j LOOP} \\
\text{DONE:}