# MIPS instructions

The MIPS instruction list as posted on Piazza:

<table>
<thead>
<tr>
<th>Word</th>
<th>.word i</th>
<th>0000 00ss ssst tttt dddd d000 0010 0000</th>
<th>R</th>
<th>$d = $s + $t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>add $d, $s, $t</td>
<td>0000 00ss ssst tttt dddd d000 0010 0010</td>
<td>R</td>
<td>$d = $s - $t</td>
</tr>
<tr>
<td>Subtract</td>
<td>sub $d, $s, $t</td>
<td>0000 00ss ssst tttt dddd d000 0010 1000</td>
<td>R</td>
<td>hi:lo = $s * $t</td>
</tr>
<tr>
<td>Multiply</td>
<td>mul $s, $t</td>
<td>0000 00ss ssst tttt dddd d000 0000 0001 1000</td>
<td>R</td>
<td>hi:lo = $s * $t</td>
</tr>
<tr>
<td>Multiply Unsigned</td>
<td>mulu $s, $t</td>
<td>0000 00ss ssst tttt dddd d000 0000 0001 1001</td>
<td>R</td>
<td>lo = $s / $t; hi = $s % $t</td>
</tr>
<tr>
<td>Divide</td>
<td>div $s, $t</td>
<td>0000 00ss ssst tttt dddd d000 0000 0001 1010</td>
<td>R</td>
<td>lo = $s / $t; hi = $s % $t</td>
</tr>
<tr>
<td>Divide Unsigned</td>
<td>divu $s, $t</td>
<td>0000 00ss ssst tttt dddd d000 0000 0001 1011</td>
<td>R</td>
<td>$d = hi</td>
</tr>
<tr>
<td>Move From High/Remainder</td>
<td>mfhi $d</td>
<td>0000 0000 0000 0000 dddd d000 0000 0000</td>
<td>R</td>
<td>$d = lo</td>
</tr>
<tr>
<td>Move From Low/Quotient</td>
<td>mflo $d</td>
<td>0000 0000 0000 0000 dddd d000 0000 0010</td>
<td>R</td>
<td>$d = MEM[pc]; pc = pc + 4</td>
</tr>
<tr>
<td>Load Immediate And Skip</td>
<td>lis $d</td>
<td>0000 0000 0000 0000 dddd d000 0000 0100</td>
<td>I</td>
<td>$t = MEM [$s + i]:4</td>
</tr>
<tr>
<td>Load Word</td>
<td>lw $t, i($s)</td>
<td>1000 11ss ssst tttt iiii iiii iiii iiii</td>
<td>I</td>
<td>MEM [$s + i]:4 = $t</td>
</tr>
<tr>
<td>Store Word</td>
<td>sw $t, i($s)</td>
<td>1010 11ss ssst tttt iiii iiii iiii iiii</td>
<td>I</td>
<td>MEM [$s + i]:4 = $t</td>
</tr>
<tr>
<td>Set Less Than</td>
<td>slt $d, $s, $t</td>
<td>0000 00ss ssst tttt dddd d000 0010 1010</td>
<td>R</td>
<td>$d = 1 if $s &lt; $t; 0 otherwise</td>
</tr>
<tr>
<td>Set Less Than Unsigned</td>
<td>sltu $d, $s, $t</td>
<td>0000 00ss ssst tttt dddd d000 0010 1011</td>
<td>R</td>
<td>$d = 1 if $s &lt; $t; 0 otherwise</td>
</tr>
<tr>
<td>Branch On Equal</td>
<td>beq $s, $t, i</td>
<td>0001 00ss ssst tttt iiii iiii iiii iiii</td>
<td>I</td>
<td>if ($s == $t) pc += i * 4</td>
</tr>
<tr>
<td>Branch On Not Equal</td>
<td>bne $s, $t, i</td>
<td>0001 01ss ssst tttt iiii iiii iiii iiii</td>
<td>I</td>
<td>if ($s != $t) pc += i * 4</td>
</tr>
<tr>
<td>Jump Register</td>
<td>jr $s</td>
<td>0000 00ss ss00 0000 0000 0000 0000 0000</td>
<td>R</td>
<td>pc = $s</td>
</tr>
<tr>
<td>Jump And Link Register</td>
<td>jalr $s</td>
<td>0000 00ss ss00 0000 0000 0000 0000 0000</td>
<td>R</td>
<td>temp = $s; $31 = pc; pc = temp</td>
</tr>
<tr>
<td>Add Immediate</td>
<td>addi $t, $s, i</td>
<td>0010 00ss ssst tttt iiii iiii iiii iiii</td>
<td>J</td>
<td>$31 = pc; pc = high4(pc).(i&lt;&lt;2)</td>
</tr>
</tbody>
</table>

Does it seem unreadable? Yes... but it’ll be clear with some examples.
Particular instructions

…. the instruction page. It *is* pretty bad.

There are a couple instructions that are particularly confusing:

- **mfhi $r** --- moves the value in Hi to register $r
- **mflo $r** --- moves the value in Lo to register $r

These rely on special registers Hi and Lo.

**Multiplication**: Hi stores any overflow and Lo stores the actual value.

**Division**: Lo stores the quotient and Hi stores the remainder

What does this mean? After any multiplication, call mflo to move the value to a register you can actually access. Same with division, if you want the quotient or remainder.
Intro programming - using MIPS

Using the instructions on the previous slide, we can now write some programs!

Example 1:

Consider the following division: \( \frac{8x^3 + 2x^2 - 11x + 3}{y} \)

Assume that registers $1$ and $2$ have been initialized with values entered using the two ints frontend. Register $1$ contains the value of $x$, register $2$ contains the value of $y$. Write a program that will put the remainder of the division in $3$. Try to minimize the number of registers you use in your solution. You may assume that the value in register $2$ is not 0.
Example 1 -- solution

Let’s walk through coding the solution

- Follow along by coding yourself as we solve together
- Think about how to divide the code into small pieces
  - For a polynomial this is easy, split it up by the +/- terms
  - Test each piece as you go
- Then put the pieces back together
Example 1 -- solution

Solution code:

```
; we know at the outset that $1$ has $x$, and $2$ has $y$
mult $1$, $1$ ; this does $x^2$ and stores it in hi:lo
mflo $3$ ; move the result of the mult into $3$
mult $1$, $3$ ; now we have $x^3$
mflo $4$ ; store $x^3$ in $4$
addi $5$, $0$, $8$ ; store $8$ in $5$
mult $4$, $5$ ; $8x^3$
mflo $4$ ; so now $4$ has $8x^3$

add $3$, $3$, $3$ ; $3 = 3 + 3$, which is $x^2 + x^2$
add $3$, $3$, $4$ ; now we have $8x^3 + 2x^2$ in $3$

addi $4$, $0$, $11$ ; store $11$ in $4$ (since we don't need $4$ anymore)
mult $1$, $4$ ; $11x$
mflo $4$ ; save it

sub $3$, $3$, $4$ ; $3$ is old $3$ ($8x^3 + 2x^2$) - $11x$
addi $3$, $3$, $3$ ; add $3$, so now we have the whole polynomial

div $3$, $2$ ; polynomial / $y$
mfhi $3$ ; store the remainder in $3$
jr $31$
```
Your turn to try some examples

You should try coding these two programs now:

Assume that registers $1$ and $2$ have been initialized with values entered using the two ints frontend. Assuming that variables $m$, $n$, and $p$ refer to the contents of registers $1$, $2$, and $3$ respectively. Write a program to do the following computations:

a. $p = (m + 10) - (n + 20)$

b. $p = (m + n)^2 - 4n$
Solutions ...

Sample solutions for the problems on the previous slide:

a. 

```
addi $3, $1, 10
addi $4, $2, 20
sub $3, $3, $4
jr $31
```

b. 

```
addi $3, $1, 2
mult $3, $3
mflo $3
addi $4, $0, 4
mult $2, $4
mflo $4
sub $3, $3, $4
jr $31
```
What does this code do?

Writing your own code is one thing, but you also need to be able to read other people’s code and figure out what it’s supposed to do:

Example:

Trace the following programs, and determine their purpose. Assume that registers $1$ and $2$ have been initialized with values entered using the twoints frontend. Assume that variables $m$, $n$, and $p$ refer to the contents of registers $1$, $2$, and $3$ respectively.

```
mult $2$, $2
mflo $3
mult $3$, $2
mflo $3
add $3$, $3$, $1
jr $31
```

```
mult $1$, $2
mflo $3
addi $4$, $0$, 7
mult $1$, $4
mflo $4
sub $3$, $3$, $4
addi $3$, $3$, 5
jr $31
```
Solution to Trace

So what do the programs do?

```assembly
mult $2, $2
mflo $3
mult $3, $2
mflo $3
add $3, $3, $1
jr $31

Answer: Computes $p = n^3 + m$
```

```assembly
mult $1, $2
mflo $3
addi $4, $0, 7
mult $1, $4
mflo $4
sub $3, $3, $4
addi $3, $3, 5
jr $31

Answer: Computes $p = mn - 7m + 5$
```
Assignment reminders

- Always test your code on the university servers before you hand it in
- Hand in the .asm file (not the .mips file)
That's all Folks!