

## MIPS instructions

Instruction	Syntax	Example
add/addu	add dest, src0, src1	add \$s0, \$s1, \$s2
sub/subu	sub dest, src0, src1	sub \$s0, \$s1, \$s2
addi/addiu	addi dest, src0, immediate	addi \$s0, \$s1, 12
sll/srl	sll dest, src0, immediate	sll \$s0, \$s1, 5
slt/sltu	slt dest, src0, src1	slt \$s0, \$s1, \$s2
slti/slтиu	slti dest, src0, immediate	slti \$s0, \$s1, 10
lw/lb/lbu	lw dest, offset(base addr)	lw \$t0, 4(\$s0)
sw/sw	sw src, offset(base addr)	sw \$t0, 4(\$s0)
bne	bne src0, src1, branchAddr	bne \$t0, \$t1, notEq
beq	beq src0, src1, branchAddr	beq \$t0, \$t1, Eq
j/jal	j jumpAddr	j jumpWhenDone
jr	Jr dest	jr \$ra

## MIPS registers

Register Number	Register Name	Register Use
\$0	\$zero	The “zero-constant”
\$1	\$at	<i>Used by the assembler</i>
\$2-\$3	\$v0-\$v1	Return values
\$4-\$7	\$a0-\$a3	Function arguments
\$8-\$15	\$t0-\$t7	Temporary registers
\$16-\$23	\$s0-\$s7	Saved registers
\$24-\$25	\$t8-\$t9	Temporary registers
\$26-\$27	\$k0-\$k1	<i>Used by the kernel</i>
\$28	\$gp	Global pointer
\$29	\$sp	Stack pointer
\$30	\$fp	Frame pointer
\$31	\$ra	Return address

## MIPS functions

If you plan on calling other functions or using saved registers, you’ll need to use the following function template:

Prologue:

```
FunctionFoo:
    addiu $sp, $sp, -FrameSize #reserve space on the stack
    sw $ra, 0($sp) #store needed registers
    sw $s0, 4($sp)
    ... save the rest of the registers ...
    sw $sx, FrameSize - 4($sp)
```

Body:

```
... Do some stuff ...
```

Epilogue:

```
lw $sx, FrameSize -4($sp) #restore registers
... load the rest of the registers...
lw $s0, 4($sp)
lw $ra, 0($sp)
addiu $sp, $sp, FrameSize #release stack spaces
jr $ra #return to normal execution
```

### Exercises:

What are the 3 meanings unsigned can have in MIPS?

Translate the following MIPS function into C or vice versa:

C	MIPS
	<pre> Foo:    add    \$v0, \$zero, \$zero Loop:   slti   \$t0, \$a1, 0         bne    \$t0, \$zero, End         sll    \$t1, \$a1, 2         add    \$t2, \$a0, \$t1         lw     \$t3, 0(\$t2)         add    \$v0, \$v0, \$t3         addi   \$a1, \$a1, -1         j      Loop End:    jr     \$ra </pre>
<pre> /* What does this program do? */  int Mystery(int a){     // fill in rest }  int Recur(int a, int b){     // fill in rest }  void swap(int * a, int * b){     int temp= *a;     *a = *b;     *b = temp; } </pre>	<pre> Mystery: add    \$a1, \$0, \$0           addiu \$sp, \$sp, -4           sw    \$ra, 0(\$sp)           jal   Recur           lw     \$ra, 0(\$sp)           addiu \$sp, \$sp 4           jr    \$ra  Recur:   bne   \$a0, \$0, Body           add   \$v0, \$0, \$0           jr    \$ra Body:    addi  \$a1, \$a1, 1           srl   \$a0, \$a0, 1           addiu \$sp, \$sp, -4           sw    \$ra, 0(\$sp)           jal   Recur           addi  \$v0, \$v0, 1           lw     \$ra, 0(\$sp)           addiu \$sp, \$sp 4           jr    \$ra </pre>
<pre> void insertionSort(int * arr, int size){     int i, j;     for(i=1; i&lt;size; i++){         j=i;         while(j&gt;0 &amp;&amp; arr[j]&lt;arr[j-1]){             swap(arr + j, arr + (j-1));             j--;         }     } } </pre>	