# MIPS Summary — CS106 / CS107

| Arithmetic | | | | |
|---|---|---|---|---|
| `add r s t` | Add | `r = s + t` | |
| `addi r s i` | Add immediate | `r = s + SignExt(i)` | *e* |
| `addiu r s i` | Add immediate unsigned | `r = s + SignExt(i)` | *e* |
| `addu r s t` | Add unsigned | `r = s + t` | |
| *`div r s t`* | *Divide* | *`r = s / t`* | *p* |
| *`divu r s t`* | *Divide unsigned* | *`r = s / t`* | *p* |
| `mul r s t` | Multiply | `r = s * t` | |
| *`neg r s`* | *Negate* | *`r = -s`* | *e p* |
| *`negu r s`* | *Negate unsigned* | *`r = -s`* | *p* |
| `sub r s t` | Subtract | `r = s – t` | *e* |
| `subu r s t` | Subtract unsigned | `r = s – t` | |

| Logic | | | |
|---|---|---|---|
| `and r s t` | And | `r = s & t` | |
| `andi r s i` | And immediate | `r = s & ZeroExt(i)` | |
| `nor r s t` | Nor | `r = ~(s \| t)` | |
| *`not r s`* | *Not* | *`r = ~s`* | *p* |
| `or r s t` | Or | `r = s \| t` | |
| `ori r s i` | Or immediate | `r = s \| ZeroExt(i)` | |
| `xor r s t` | Exclusive or | `r = s ^ t` | |
| `xori r s i` | Exclusive or immediate | `r = s ^ ZeroExt(i)` | |

| Shifting | | | |
|---|---|---|---|
| `sll r s i` | Shift left logical | `r = s << i` | |
| `sllv r s t` | Shift left logical variable | `r = s << t` | |
| `sra r s i` | Shift right arithmetic | `r = SignExt(s >> i)` | |
| `srav r s t` | Shift right arithmetic variable | `r = SignExt(s >> t)` | |
| `srl r s i` | Shift right logical | `r = ZeroExt(s >> i)` | |
| `srlv r s t` | Shift right logical variable | `r = ZeroExt(s >> t)` | |

| Loading and storing | | | |
|---|---|---|---|
| *`la r a`* | *Load address* | *`r = a`* | *p* |
| `lb r a` | Load byte | `r = SignExt(*(int8*)a)` | |
| `lbu r a` | Load byte unsigned | `r = ZeroExt(*(int8*)a)` | |
| `lh r a` | Load halfword | `r = SignExt(*(int16*)a)` | |
| `lhu r a` | Load halfword unsigned | `r = ZeroExt(*(int16*)a)` | |
| *`li r i`* | *Load immediate* | *`r = SignExt(i)`* | *p* |
| `lui r i` | Load upper immediate | `r = i << 16` | |
| `lw r a` | Load word | `r = *(int32*)a` | |
| *`move r s`* | *Move* | *`r = s`* | *p* |
| `sb r a` | Store byte | `*(int8*)a = r & 0xFF` | |
| `sh r a` | Store halfword | `*(int16*)a = r & 0xFFFF` | |
| `sw r a` | Store word | `*(int32*)a = r` | |

| Comparison | | | |
|---|---|---|---|
| *`seq r s t`* | *Set equal* | *`r = (s == t) ? 1 : 0`* | *p* |
| *`sge r s t`* | *Set greater than or equal* | *`r = (s >=`$_s$` t) ? 1 : 0`* | *p* |
| *`sgeu r s t`* | *Set greater than or equal unsigned* | *`r = (s >=`$_u$` t) ? 1 : 0`* | *p* |
| *`sgt r s t`* | *Set greater than* | *`r = (s >`$_s$` t) ? 1 : 0`* | *p* |
| *`sgtu r s t`* | *Set greater than unsigned* | *`r = (s >`$_u$` t) ? 1 : 0`* | *p* |
| *`sle r s t`* | *Set less than or equal* | *`r = (s <=`$_s$` t) ? 1 : 0`* | *p* |
| *`sleu r s t`* | *Set less than or equal unsigned* | *`r = (s <=`$_u$` t) ? 1 : 0`* | *p* |
| *`sne r s t`* | *Set not equal* | *`r = (s != t) ? 1 : 0`* | *p* |
| `slt r s t` | Set less than | `r = (s <`$_s$` t) ? 1 : 0` | |
| `slti r s i` | Set less than immediate | `r = (s <`$_s$` i) ? 1 : 0` | |
| `sltiu r s i` | Set less than immediate unsigned | `r = (s <`$_u$` i) ? 1 : 0` | |
| `sltu r s t` | Set less than unsigned | `r = (s <`$_u$` t) ? 1 : 0` | |

| Branching and jumping | | | |
|---|---|---|---|
| `beq r s L` | Branch equal | `if (r == s) goto L` | |
| *`beqz r L`* | *Branch equal to zero* | *`if (r == 0) goto L`* | *p* |
| *`bge r s L`* | *Branch greater than or equal* | *`if (r >=`$_s$` s) goto L`* | *p* |
| *`bgeu r s L`* | *Branch greater than or equal unsigned* | *`if (r >=`$_u$` s) goto L`* | *p* |
| `bgez r L` | Branch greater than or equal to zero | `if (r >=`$_s$` 0) goto L` | |
| *`bgt r s L`* | *Branch greater than* | *`if (r >`$_s$` s) goto L`* | *p* |
| *`bgtu r s L`* | *Branch greater than unsigned* | *`if (r >`$_u$` s) goto L`* | *p* |
| `bgtz r L` | Branch greater than zero | `if (r >`$_s$` 0) goto L` | |
| *`ble r s L`* | *Branch less than or equal* | *`if (r <=`$_s$` s) goto L`* | *p* |
| *`bleu r s L`* | *Branch less than or equal unsigned* | *`if (r <=`$_u$` s) goto L`* | *p* |
| `blez r L` | Branch less than or equal to zero | `if (r <=`$_s$` 0) goto L` | |
| *`blt r s L`* | *Branch less than* | *`if (r <`$_s$` s) goto L`* | *p* |
| *`bltu r s L`* | *Branch less than unsigned* | *`if (r <`$_u$` s) goto L`* | *p* |
| `bltz r L` | Branch less than zero | `if (r <`$_s$` 0) goto L` | |
| `bne r s L` | Branch not equal | `if (r != s) goto L` | |
| *`bnez r L`* | *Branch not equal to zero* | *`if (r != 0) goto L`* | *p* |
| `j L` | Jump | `goto L` | |
| `jal L` | Jump and link | `$ra = pc+4; goto L` | |
| `jalr r` | Jump and link register | `$ra = pc+4; goto r` | |
| `jr r` | Jump register | `goto r` | |

`<`$_s$ / `<`$_u$ : *signed/unsigned comparisons* *e*: *may cause overflow exception* *p*: *pseudoinstruction*

| | | |
|---|---|---|
| `$0` | Constant zero | |
| `$v0, $v1` | Return values | |
| `$a0 – $a3` | Arguments | |
| `$t0 – $t9` | Temporaries | |
| `$s0 – $s7` | Saved temporaries | *preserved* |

| | | |
|---|---|---|
| `$at, $k0, $k1` | *Reserved* | |
| `$gp` | Global pointer | *preserved* |
| `$sp` | Stack pointer | *preserved* |
| `$fp` | Frame pointer | *preserved* |
| `$ra` | Return address | *preserved* |