Assembly Operations

The right column gives the command followed by its arguments. Arguments beginning with r are registers. An `imm` argument is a number; its subscript is the maximum size in bits. For commands with a result, `rd` is the destination.

Basic operations

Add	`add rd, rs, rt`
Add immediate	`addi rd, rs,` $\texttt{imm}_{16}$
Subtract	`sub rd, rs, rt`
Multiply∗	`mulo rd, rs, rt`
Divide s by t∗	`div rd, rs, rt`
Remainder∗	`rem rd, rs, rt`
Load immediate∗	`li rd,` $\texttt{imm}_{32}$
System call (see table below)	`syscall`

`syscall` is controlled by the contents of register $\$v0$:

1	Print int stored in $\$a0$
4	Print string whose address is stored in $\$a0$
5	Read int into $\$v0$
10	Exit simulation
11	Print character stored in $\$a0$

Comparison, branch, and jump instructions

Unconditional branch∗	`b label`
Branch if equal	`beq rs, rt, label`
Branch if not equal	`bne rs, rt, label`
Branch if greater than∗	`bgt rs, rt, label`
Branch if greater than or equal∗	`bge rs, rt, label`
Branch if less than∗	`blt rs, rt, label`
Branch if less than or equal∗	`ble rs, rt, label`

Memory instructions

Shift left (by immediate; fill with 0s)	`sll rd, rs,` $\texttt{imm}_5$
"the following goes in the text segment"	`.text`
"the following goes in the data segment"	`.data`
"store `str` as a null-terminated string"	`.asciiz str`
"store these words in memory"	`.word w1 w2 ...`
"reserve `n` bytes of space"	`.space n`
"align on 2^n-byte boundary"	`.align n`
Load address (not contents) ∗	`la rd, address`
Load byte	`lb rd, address`
Store byte	`sb rt, address`
Load word	`lw rd, address`
Store word	`sw rd, address`

Addresses for memory instructions can have any of the following forms:

(register)	contents of register
imm(register)	contents of register + immediate
label	address of label