Pseudoinstrucción	Instrucción/Instrucciones Base	Significado
`nop`	`addi x0, x0, 0`	No operation
`neg rd, rs`	`sub rd, x0, rs`	Complemento a 2
`negw rd, rs`	`subw rd, x0, rs`	Complemento a 2 (word)
`snez rd, rs`	`sltu rd, x0, rs`	Poner en 1 si $\neq$ cero
`sltz rd, rs`	`slt rd, rs, x0`	Poner en 1 si $<$ cero
`sgtz rd, rs`	`slt rd, x0, rs`	Poner en 1 si $>$ cero
`beqz rs, offset`	`beq rs, x0, offset`	Branch si $=$ cero
`bnez rs, offset`	`bne rs, x0, offset`	Branch si $\neq$ cero
`blez rs, offset`	`bge x0, rs, offset`	Branch si $\leq$ cero
`bgez rs, offset`	`bge rs, x0, offset`	Branch si $\geq$ cero
`bltz rs, offset`	`blt rs, x0, offset`	Branch si $<$ cero
`bgtz rs, offset`	`blt x0, rs, offset`	Branch si $>$ cero
`j offset`	`jal x0, offset`	Jump
`jr rs`	`jalr x0, rs, 0`	Jump a registro
`ret`	`jalr x0, x1, 0`	Retornar de subrutina
`tail offset`	`auipc x6, offset[31:12]` `jalr x0, x6, offset[11:0]`	*Tail call* subrutina lejana
`rdinstret[h] rd`	`csrrs rd, instret[h], x0`	Leer el contador de instrucciones retiradas
`rdcycle[h] rd`	`csrrs rd, cycle[h], x0`	Leer el contador de ciclos
`rdtime[h] rd`	`csrrs rd, time[h], x0`	Leer *real-time clock*
`csrr rd, csr`	`csrrs rd, csr, x0`	Leer CSR
`csrw csr, rs`	`csrrw x0, csr, rs`	Escribir CSR
`csrs csr, rs`	`csrrs x0, csr, rs`	Poner bits en 1 en CSR
`csrc csr, rs`	`csrrc x0, csr, rs`	Poner bits en 0 en CSR
`csrwi csr, imm`	`csrrwi x0, csr, imm`	Escribir CSR, inmediato
`csrsi csr, imm`	`csrrsi x0, csr, imm`	Poner bits en 1 en CSR, inmediato
`csrci csr, imm`	`csrrci x0, csr, imm`	Poner bits en 0 en CSR, inmediato
`frcsr rd`	`csrrs rd, fcsr, x0`	Leer *FP control/status register*
`fscsr rs`	`csrrw x0, fcsr, rs`	Escribir *FP control/status register*
`frrm rd`	`csrrs rd, frm, x0`	Leer *FP rounding mode*
`fsrm rs`	`csrrw x0, frm, rs`	Escribir *FP rounding mode*
`frflags rd`	`csrrs rd, fflags, x0`	Leer *FP exception flags*
`fsflags rs`	`csrrw x0, fflags, rs`	Escribir *FP exception flags*

Figura 3.3: 32 pseudo-instrucciones de RISC-V que dependen de x0, el registro cero. El Apéndice A incluye tanto las instrucciones reales como pseudoinstrucciones de RISC-V. Las que leen los contadores de 64 bits pueden leer los 32 bits de la parte alta utilizando la versión "h" de la instrucción y la versión normal para leer la parte baja (Las Tablas 20.2 y 20.3 de [Waterman and Asanović 2017] son la base de esta figura).

Directiva	Descripción
`.text`	Ítems subsiguientes son almacenados en la sección text (código de máquina).
`.data`	Ítems subsiguientes son almacenados en la sección data (variables globales).
`.bss`	Ítems subsiguientes son almacenados en la sección bss (variables globales inicializadas a 0).
`.section .foo`	Ítems subsiguientes son almacenados en la sección llamada `.foo`.
`.align n`	Alinear el siguiente dato en un límite de 2^n bytes. Por ejemplo, `.align 2` alinea el próximo valor en un límite de *word*.
`.balign n`	Alinear el siguiente dato en un límite de n bytes. Por ejemplo, `.balign 4` alinea el próximo valor en un límite de *word*.
`.globl sym`	Declara que la etiqueta sym es global y se le puede hacer referencia desde otros archivos.
`.string "str"`	Almacenar el string str en memoria y terminarlo en null.
`.byte b1,..., bn`	Almacenar las n cantidades de 8 bits en bytes sucesivos de memoria.
`.half w1,...,wn`	Almacenar las n cantidades de 16 bits en halfwords sucesivos de memoria.
`.word w1,...,wn`	Almacenar las n cantidades de 32 bits en words sucesivos de memoria.
`.dword w1,...,wn`	Almacenar las n cantidades de 64 bits en doublewords sucesivos de memoria.
`.float f1,..., fn`	Almacenar los n números de punto flotante de precisión simple en words sucesivos de memoria.
`.double d1,..., dn`	Almacenar los n números de punto flotante de precisión doble en doublewords sucesivos de memoria.
`.option rvc`	Comprimir las instrucciones subsiguientes (ver Capítulo 7).
`.option norvc`	No comprimir las instrucciones subsiguientes.
`.option relax`	Permitir relajación del linker para las instrucciones subsiguientes.
`.option norelax`	No permitir relajación del linker para las instrucciones subsiguientes.
`.option pic`	Las instrucciones subsiguientes son *position-independent code*.
`.option nopic`	Las instrucciones subsiguientes son *position-dependent code*.
`.option push`	Hacer Push del estado de todos los `.options` a un stack, para que un posterior `.option pop` restaure sus valores.
`.option pop`	Hacer Pop del stack de opciones, restaurando todos los `.options` a su estado en el momento del último `.option push`.

Figura 3.9: Directivas comunes del ensamblador de RISC-V.