Facultad de Informática
Universidad Complutense de Madrid

Fundamentos de Computadores II - Chuletario

Daniel Báscones (danibasc@ucm.es)

1 de marzo de 2023

1. Los registros

En total hay 32 registros, denominados como `x0`-`x31`. La Tabla 1 los define todos. Es importante ver que los podemos nombrar tanto por su identificador de registro (con la "x"), como por su nombre. Aunque técnicamente podemos usar los registros con una funcionalidad diferente a la descrita, es ***esencial*** seguir la función *estándar* a fin de escribir código *compatible* con otro ya escrito.

Código 5-bit	Registro	Nombre	Descripción	Tipo
00000	`x0`	`zero`	Siempre cero	Cero
00001	`x1`	`ra`	dirección de retorno	Reservado
00010	`x2`	`sp`	puntero de pila	Reservado
00011	`x3`	`gp`	puntero global	Reservado
00100	`x4`	`tp`	puntero de hilo	Reservado
00101	`x5`	`t0`	registro temporal 0	Temporal
00110	`x6`	`t1`	registro temporal 1	Temporal
00111	`x7`	`t2`	registro temporal 2	Temporal
01000	`x8`	`s0`	registro salvado 0*	Salvado
01001	`x9`	`s1`	registro salvado 1	Salvado
01010	`x10`	`a0`	argumento 0 / retorno 0	Argumento
01011	`x11`	`a1`	argumento 1 / retorno 1	Argumento
01100	`x12`	`a2`	argumento 2	Argumento
.....	...	...	argumentos 3-6	Argumento
10001	`x17`	`a7`	argumento 7	Argumento
10010	`x18`	`s2`	registro salvado 2	Salvado
.....	...	...	registros salvados 3-10	Salvado
10010	`x27`	`s11`	registro salvado 11	Salvado
11100	`x28`	`t3`	registro temporal 3	Temporal
11101	`x29`	`t4`	registro temporal 4	Temporal
11110	`x30`	`t5`	registro temporal 5	Temporal
11111	`x31`	`t6`	registro temporal 6	Temporal

Tabla 1: Registros de RISC-V. * El registro `s0` también se usa como puntero de marco `fp`.

2. Las instrucciones

Dentro del repertorio base de instrucciones **RV32I**, encontramos instrucciones variadas. Las aritmético-lógicas nos permiten operar con valores en los registros. Las de acceso a memoria permiten leer (cargar) y escribir (guardar) datos en memoria. Las instrucciones de salto servirán para programar estructuras y condicionales (*if, while, for...*) y funciones.

Instrucción	Uso	Descripción	Tipo
`add rd, rs1, rs2`	Suma	$r_d \leftarrow r_{s1} + r_{s2}$	R$_{\text{Aritmética}}$
`sub rd, rs1, rs2`	Resta	$r_d \leftarrow r_{s1} - r_{s2}$	R$_{\text{Aritmética}}$
`xor rd, rs1, rs2`	Or exclusiva	$r_d \leftarrow r_{s1} \wedge r_{s2}$	R$_{\text{Lógica}}$
`or rd, rs1, rs2`	Or	$r_d \leftarrow r_{s1} \mid r_{s2}$	R$_{\text{Lógica}}$
`and rd, rs1, rs2`	And	$r_d \leftarrow r_{s1} \,\&\, r_{s2}$	R$_{\text{Lógica}}$
`sll rd, rs1, rs2`	Shift lóg izda	$r_d \leftarrow r_{s1} \text{´} r_{s2}$	R$_{\text{Lógica}}$
`srl rd, rs1, rs2`	Shift lóg dcha	$r_d \leftarrow r_{s1} \text{ˇ} r_{s2}$	R$_{\text{Lógica}}$
`sra rd, rs1, rs2`	Shift arit dcha	$r_d \leftarrow r_{s1} \text{ˇ} r_{s2}$	R$_{\text{Lógica}}$
`slt* rd, rs1, rs2`	Activa si $<$	$r_d \leftarrow (r_{s1} < r_{s2})\,?\,1 : 0$	R$_{\text{Lógica}}$
`beq rs1, rs2, i12`	Salta si $\leftarrow$	if $r_{s1} \leftarrow r_{s2}$ $\{pc \leftarrow pc + 2 * s_{ext}(i_{12})\}$	B$_{\text{Salto}}$
`bne rs1, rs2, i12`	Salta si $\neq$	if $r_{s1} \neq r_{s2}$ $\{pc \leftarrow pc + 2 * s_{ext}(i_{12})\}$	B$_{\text{Salto}}$
`blt* rs1, rs2, i12`	Salta si $<$	if $r_{s1} < r_{s2}$ $\{pc \leftarrow pc + 2 * s_{ext}(i_{12})\}$	B$_{\text{Salto}}$
`bge* rs1, rs2, i12`	Salta si $\geq$	if $r_{s1} \geq r_{s2}$ $\{pc \leftarrow pc + 2 * s_{ext}(i_{12})\}$	B$_{\text{Salto}}$
`sb rs2, i12(rs1)`	Guarda byte	$M[r_{s1} + s_{ext}(i_{12})] \leftarrow r_{s2_{7:0}}$	S$_{\text{Guardado}}$
`sh rs2, i12(rs1)`	Guarda short	$M[r_{s1} + s_{ext}(i_{12})] \leftarrow r_{s2_{15:0}}$	S$_{\text{Guardado}}$
`sw rs2, i12(rs1)`	Guarda palabra	$M[r_{s1} + s_{ext}(i_{12})] \leftarrow r_{s2}$	S$_{\text{Guardado}}$
`lb rd, i12(rs1)`	Carga byte	$r_d \leftarrow s_{ext}(M[r_{s1} + s_{ext}(i_{12})])$	I$_{\text{Carga}}$
`lh rd, i12(rs1)`	Carga short	$r_d \leftarrow s_{ext}M[r_{s1} + s_{ext}(i_{12})])$	I$_{\text{Carga}}$
`lbu rd, i12(rs1)`	Carga byte	$r_d \leftarrow z_{ext}M[r_{s1} + s_{ext}(i_{12})])$	I$_{\text{Carga}}$
`lhu rd, i12(rs1)`	Carga short	$r_d \leftarrow z_{ext}M[r_{s1} + s_{ext}(i_{12})])$	I$_{\text{Carga}}$
`lw rd, i12(rs1)`	Carga palabra	$r_d \leftarrow M[r_{s1} + s_{ext}(i_{12})]$	I$_{\text{Carga}}$
`addi rd, rs1, i12`	`add` inmediato	$r_d \leftarrow r_{s1} + s_{ext}(i_{12})$	I$_{\text{Aritmética}}$
`xori rd, rs1, i12`	`xor` inmediato	$r_d \leftarrow r_{s1} \wedge s_{ext}(i_{12})$	I$_{\text{Aritmética}}$
`ori rd, rs1, i12`	`or` inmediato	$r_d \leftarrow r_{s1} \mid s_{ext}(i_{12})$	I$_{\text{Lógica}}$
`andi rd, rs1, i12`	`and` inmediato	$r_d \leftarrow r_{s1} \,\&\, s_{ext}(i_{12})$	I$_{\text{Lógica}}$
`slli rd, rs1, i5`	`sll` inmediato	$r_d \leftarrow r_{s1} \text{´} s_{ext}(i_5)$	I$_{\text{Lógica}}$
`srli rd, rs1, i5`	`srl` inmediato	$r_d \leftarrow r_{s1} \text{ˇ} s_{ext}(i_5)$	I$_{\text{Lógica}}$
`srai rd, rs1, i5`	`sra` inmediato	$r_d \leftarrow r_{s1} \text{ˇ} s_{ext}(i_5)$	I$_{\text{Lógica}}$
`slti* rd, rs1, i12`	`slt` inmediato	$r_d \leftarrow (r_{s1} < s_{ext}(i_{12}))\,?\,1 : 0$	I$_{\text{Lógica}}$
`jalr rd, rs1, i12`	Salta/enlaza reg	$r_d \leftarrow pc + 4$ $pc \leftarrow rs1 + s_{ext}(i_{12})$	I$_{\text{Control}}$
`jal rd, i21`	Salta/enlaza	$r_d \leftarrow pc + 4$ $pc \leftarrow pc + s_{ext}(i_{20:1} \text{´} 1)$	J$_{\text{Control}}$
`auipc rd, i20`	Suma imm a pc	$r_d \leftarrow pc + (i_{20} \text{´} 12)$	U$_{\text{Control}}$
`lui rd, i20`	Carga inmediato	$r_d \leftarrow i_{20} \text{´} 12$	U$_{\text{Datos}}$

Tabla 2: Instrucciones **RV32I**. *: tienen variantes sin signo acabadas en "u" (e.g: `bltu`).

3. Las instrucciones (Extensión RVM)

Dentro del repertorio extendido de instrucciones **RVM**, encontramos instrucciones variadas de multiplicación y división de tipo aritmético.

Instrucción	Uso	Descripción	Tipo
`mul rd, rs1, rs2`	Producto (parte baja)	$r_d \leftarrow (r_{s1} * r_{s2})_{31:0}$	R - Aritmética
`mulh rd, rs1, rs2`	Producto (parte alta) Ambos con signo	$r_d \leftarrow (r_{s1_S} * r_{s2_S})_{63:32}$	R - Aritmética
`mulhsu rd, rs1, rs2`	Producto (parte alta) Solo uno con signo	$r_d \leftarrow (r_{s1_S} * r_{s2_U})_{63:32}$	R - Aritmética
`mulhu rd, rs1, rs2`	Producto (parte alta) Ambos sin signo	$r_d \leftarrow (r_{s1_U} * r_{s2_U})_{63:32}$	R - Aritmética
`div rd, rs1, rs2`	División con signo	$r_d \leftarrow (r_{s1} \ /_S \ r_{s2})$	R - Aritmética
`divu rd, rs1, rs2`	División sin signo	$r_d \leftarrow (r_{s1} \ /_U \ r_{s2})$	R - Aritmética
`rem rd, rs1, rs2`	Resto con signo	$r_d \leftarrow (r_{s1} \ \%_S \ r_{s2})$	R - Aritmética
`remu rd, rs1, rs2`	Resto sin signo	$r_d \leftarrow (r_{s1} \ \%_U \ r_{s2})$	R - Aritmética

Tabla 3: La lista de instrucciones de la extensión **M**

4. Las directivas en ensamblador

Directiva	Descripción
`.text`	Declara el comienzo de la sección de instrucciones
`.data`	Declara comienzo de sección de variables globales con valor inicial
`.bss`	Declara comienzo de sección de variables globales sin valor inicial
`.word w1,...,wn`	Reserva memoria para n palabras inicializadas a valores w_i
`.half h1,...,hn`	Reserva memoria para n medias palabras inicializadas con h_i
`.byte h1,...,hn`	Reserva memoria para n bytes inicializados con h_i
`.zero n`	Reserva memoria para n bytes inicializados a 0
`.space n`	Reserva espacio en memoria para n bytes sin inicializar
`.string "str"`	Reserva espacio en memoria inicializado con la cadena "str"
`.align n`	Alinea los datos/instrucciones a direcciones múltiplo de 2^n
`.equ sym, val`	Define una constante simbólica llamada *sym* de valor *val*
`.global sym`	Hace visible la etiqueta *sym* fuera del archivo que la contiene (global)
`.extern sym`	Indica que un símbolo está definido en otro archivo del proyecto
`.end`	Declara el final del programa ensamblador

Tabla 4: La lista de las principales directivas **RISCV**

5. Las pseudoinstrucciones

Para facilitar nuestro trabajo como programadores, RISC-V incluye pseudoinstrucciones que nos ayudarán a escribir códigos más sencillos. Las pseudoinstrucciones se traducen por instrucciones base, cambiando operandos o utilizando varias instrucciones base para conseguir un funcionamiento más amplio del repertorio.

Pseudoinstrucción	Uso	Traducción
`li rd, i12`	Carga inmediato corto	`addi rd, zero, i12`
`li rd, i32`	Carga inmediato (32-bit)	`lui rd, i32[31:12]` `addi rd, rd, i32[11:0]`
`la rd, sym`	Carga dirección (relativa a pc)	`auipc rd, sym[31:12]` `addi rd, rd, sym[11:0]`
`lw rd, i32` (También `lh`, `lb`)	Carga de dirrección absoluta	`auipc rd, i32[31:12]` `lw rd, i32[11:0](rd)`
`sw rs2, i32, rs1` (También `sh`, `sb`)	Guarda en dirrección absoluta	`auipc rs1, i32[31:12]` `sw rs2, i32[11:0](rs1)`
`mv rd, rs`	Copia de valor	`addi rd, rs, 0`
`not rd, rs`	Complemento a 1 (invertir)	`xori rd, rs, -1`
`neg rd, rs`	Complemento a 2 (negativo)	`sub rd, zero, rs`
`bgt* rs1, rs2, i12`	Salta si $>$	`blt rs2, rs1, i12`
`ble* rs1, rs2, i12`	Salta si $\leq$	`bge rs2, rs1, i12`
`beqz rs1, i12`	Salta si $= 0$	`beq rs1, zero, i12`
`bnez rs1, i12`	Salta si $\neq 0$	`bne rs1, zero, i12`
`bltz rs1, i12`	Salta si < 0	`blt rs1, zero, i12`
`bgez rs1, i12`	Salta si ≥ 0	`bge rs1, zero, i12`
`blez rs1, i12`	Salta si ≤ 0	`bge zero, rs1, i12`
`bgtz rs1, i12`	Salta si > 0	`blt zero, rs1, i12`
`seqz rd, rs1`	Activa si $= 0$	`sltiu rd, rs1, 1`
`snez rd, rs1`	Activa si $\neq 0$	`sltu rd, zero, rs1`
`sltz rd, rs1`	Activa si < 0	`slt rd, rs1, zero`
`sgtz rd, rs1`	Activa si > 0	`slt rd, zero, rs1`
`jr rs1`	Salta a registro	`jalr zero, rs1, 0`
`jalr rs1`	Salta/enlaza a registro	`jalr ra, rs1, 0`
`j i21`	Salta inmediato	`jal zero, i21`
`jal i21`	Salta/enlaza inmediato	`jal ra, i21`
`call i21`	Salta a función (cerca)	`jal ra, i21`
`call i32`	Salto a función (lejos)	`auipc ra, i32[31:12]` `jalr ra, i32[11:0](ra)`
`ret`	Vuelta de función	`jalr zero, ra, 0`
`nop`	No operación	`addi zero, zero, 0`

Tabla 5: La lista de pseudoinstrucciones base. Las instrucciones con * tienen variantes acabadas en "u" (e.g: `bgtu`) que operan con los valores como si fueran sin signo.