

Ensamblado y desensamblado de LEGv8

OdC - 2022

Formatos de instrucción

CORE INSTRUCTION FORMATS

R	opcode	Rm	shamt	Rn	Rd
	31 21 20	16 15	10 9	5 4	0
I	opcode	ALU_immediate	Rn	Rd	
	31 22 21	10 9	5 4		0
D	opcode	DT_address	op	Rn	Rt
	31 21 20	12 11 10 9		5 4	0
B	opcode	BR_address			
	31 26 25				0
CB	Opcode	COND BR_address		Rt	
	31 24 23			5 4	0
IW	opcode	MOV_immediate		Rd	
	31 21 20			5 4	0

- Todas las instrucciones son de 32 bits.
- Los 32 registros de LEGv8 se referencian por su número, de 0 a 31 (de 0b00000 a 0b11111).

Ensamblado de una instrucción

- Traducir una instrucción en assembler LEGv8 a una instrucción de máquina.
- Por ejemplo: La instrucción representada simbólicamente como

ADD X9, X20, X21

se ensambla en binario como:

10001011000	10101	000000	10100	01001
11 bits	5 bits	6 bits	5 bits	5 bits

- Cada segmento de la instrucción se llama campo.

Campos de LEGv8 - Instrucción tipo R



- opcode: Denotes the operation and format of an instruction.
- Rm: The second register source operand.
- shamt: Shift amount.
- Rn: The first register source operand.
- Rd: The register destination operand. It gets the result of the operation.

Campos de LEGv8 - Instrucción tipo D

opcode	address	op2	Rn	Rt
11 bits	9 bits	2 bits	5 bits	5 bits

- Data transfer instructions (loads and stores).
- The 9-bit address means a load register instruction can load any doubleword within a region of $\pm 2^8$ of the address in the base register Rn.
- The last field of D-type is called Rt instead of Rd because for store instructions, the field indicates a data source and not a data destination.
- op2: expands opcode field (will be 0 in LEGv8).

(5) DTAddr = { 55 {DT_address [8]}, DT_address }

Campos de LEGv8 - Instrucción tipo I

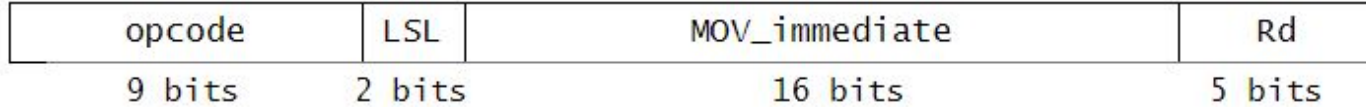
opcode	immediate	Rn	Rd
10 bits	12 bits	5 bits	5 bits

- The ARMv8 architects decided it would be useful to have a larger immediate field for these instructions, even shaving a bit from the opcode field to make a 12-bit immediate.
- The LEGv8 immediate field in I-format is zero extended.

$$(2) \quad \text{ALUImm} = \{ 52'b0, \text{ALU_immediate} \}$$

Nota para QEMU: The immediate fields for ANDI, ORRI, and EORI of the full ARMv8 instruction set are not simple 12-bit immediates. It has the unusual feature of using a complex algorithm for encoding immediate values. This means that some small constants are valid, while others are not.

Campos de LEGv8 - Instrucción tipo IM



The machine language version of `MOVZ X9, 255, LSL 16`:

110100101	01	0000 0000 1111 1111	01001
-----------	----	---------------------	-------

Contents of register `X9` after executing `MOVZ X9, 255, LSL 16`:

0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 1111 1111	0000 0000 0000 0000
---------------------	---------------------	---------------------	---------------------

The machine language version of `MOVK X9, 255, LSL 0`:

111100101	00	0000 0000 1111 1111	01001
-----------	----	---------------------	-------

Given value of `X9` above, new contents of `X9` after executing `MOVK X9, 255, LSL 0`:

0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 1111 1111	0000 0000 1111 1111
---------------------	---------------------	---------------------	---------------------

bit 22	bit 21	LSL
0	0	0
0	1	16
1	0	32
1	1	48

(6) `MOVImm = { 48'b0, MOV_immediate }`

Instrucciones de salto

Conjunto de instrucciones - Saltos

Conditional branch	compare and branch on equal 0	CBZ X1, 25	if (X1 == 0) go to PC + 100	Equal 0 test; PC-relative branch
	compare and branch on not equal 0	CBNZ X1, 25	if (X1 != 0) go to PC + 100	Not equal 0 test; PC-relative branch
	branch conditionally	B.cond 25	if (condition true) go to PC + 100	Test condition codes; if true, branch
Unconditional branch	branch	B 2500	go to PC + 10000	Branch to target address; PC-relative
	branch to register	BR X30	go to X30	For switch, procedure return
	branch with link	BL 2500	X30 = PC + 4; PC + 10000	For procedure call PC-relative

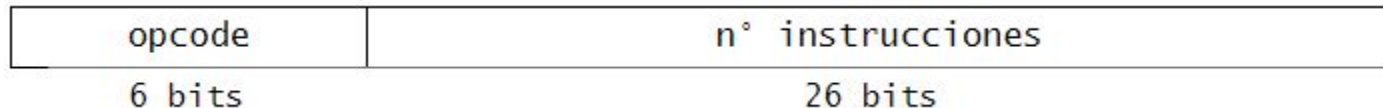
Campos de LEGv8 - Instrucción tipo B

B loop // go to “loop”

- Instruction set: Branch B B 0A0-0BF PC = PC + BranchAddr (3,9)
- Instruction format: B

opcode	BR_address
31 26 25	0
- Note: (3) BranchAddr = { 36{BR_address [25]}, BR_address, 2'b0 }

Since all LEGv8 instructions are 4 bytes long, LEGv8 stretches the distance of the branch by having PC-relative addressing refer to the number of words to the next instruction instead of the number of bytes.



Campos de LEGv8 - Instrucciones CBZ y CBNZ (CB)

CBNZ X19, Exit // go to Exit if X19 \neq 0

Unlike the branch instruction, a conditional branch instruction can specify one operand in addition to the branch address, leaving only 19 bits for the branch address.

- | | | | | |
|------------------------------|------|----|---------|--|
| Compare & Branch if Not Zero | CBNZ | CB | 5A8-5AF | if(R[Rt] \neq 0)
PC = PC + CondBranchAddr |
| Compare & Branch if Zero | CBZ | CB | 5A0-5A7 | if(R[Rt]==0)
PC = PC + CondBranchAddr |

- Instruction format:**

CB	Opcode	COND_BR_address	Rt
31	24 23	5 4	0

- Note:** (4) CondBranchAddr = { 43 {COND_BR_address ~~18~~ }, COND_BR_address, 2'b0 }

opcode	n° instrucciones	Rt
8 bits	19 bits	5 bits

Campos de LEGv8 - Instrucciones B.cond (CB)

The conditional branch instructions that rely on condition codes also use the CB-type format, but they use the final field to select among the many possible branch conditions.

Branch conditionally B.cond CB 2A0-2A7 if(FLAGS==cond)
PC = PC + CondBranchAddr (4,9)

Instruction	Rt [4:0]	Instruction	Rt [4:0]
B.EQ	00000	B.VC	00111
B.NE	00001	B.HI	01000
B.HS	00010	B.LS	01001
B.LO	00011	B.GE	01010
B.MI	00100	B.LT	01011
B.PL	00101	B.GT	01100
B.VS	00110	B.LE	01101

opcode	n° instrucciones	condición
8 bits	19 bits	5 bits