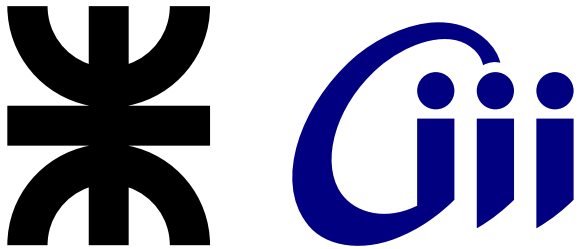
Instrucciones en el ARM

Guillermo Steiner

Centro de Investigación en Informática para la Ingeniería
Universidad Tecnológica Nacional, F.R.C.
http://ciii.frc.utn.edu.ar
Córdoba, Argentina



Técnicas Digitales II

Características del ARM

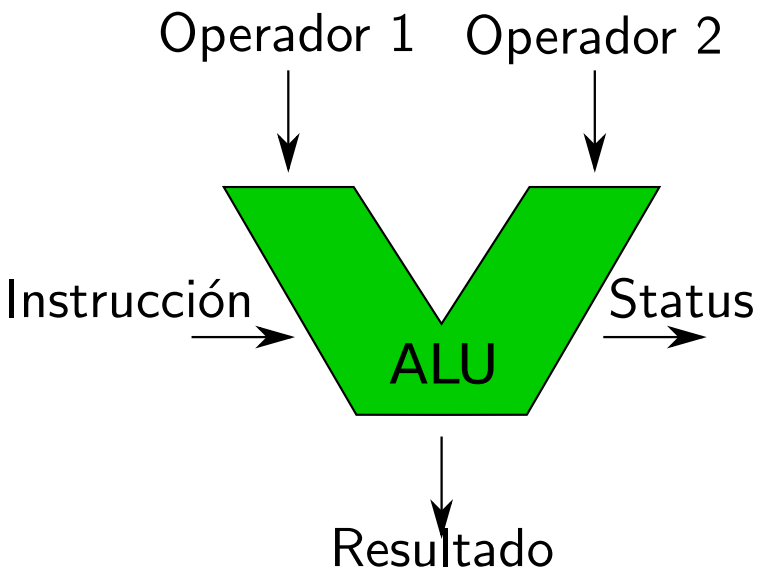
- Las instrucciones son de longitud fija 32 bits o 16 bits.
- La mayoría de las instrucciones se realizan en un ciclo de reloj.
- Todas las instrucciones son condicionadas.
- Arquitectura tipo load/store.
- 7 modos de operación con diferentes privilegios y diferentes contextos (útil en SO).

Unidad Aritmética Lógica

- ALU (Arithmetic Logic Unit)
- Circuito Digital que realiza las operaciones Aritméticas (Suma, Resta, etc) y Lógicas (AND, OR, NOT, etc)
- Fue propuesta por Von Neumann en el año 1945 para el proyecto EDVAC

Unidad Aritmética Lógica

- ALU (Arithmetic Logic Unit)
- Circuito Digital que realiza las operaciones Aritméticas (Suma, Resta, etc) y Lógicas (AND, OR, NOT, etc)
- Fue propuesta por Von Neumann en el año 1945 para el proyecto EDVAC



Tipos de Unidades Aritméticas Lógicas

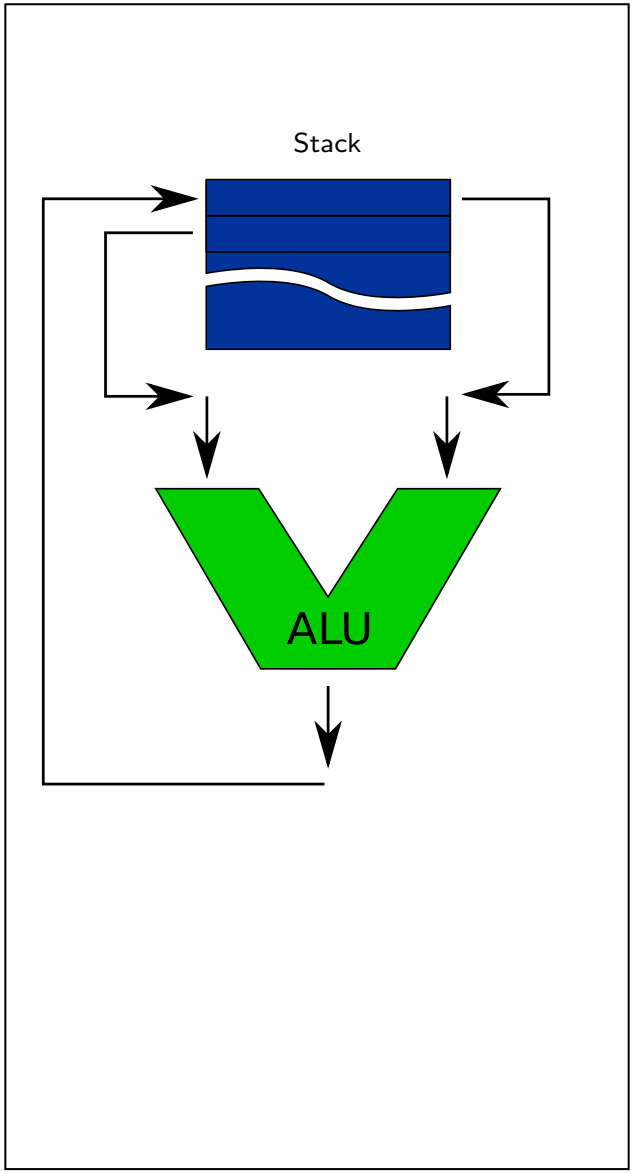
Según el origen y el destino de sus datos.

- STACK
- Registro
- Registro Memoria
- Load - Store

Tipos de Unidades Aritméticas Lógicas

Según el origen y el destino de sus datos.

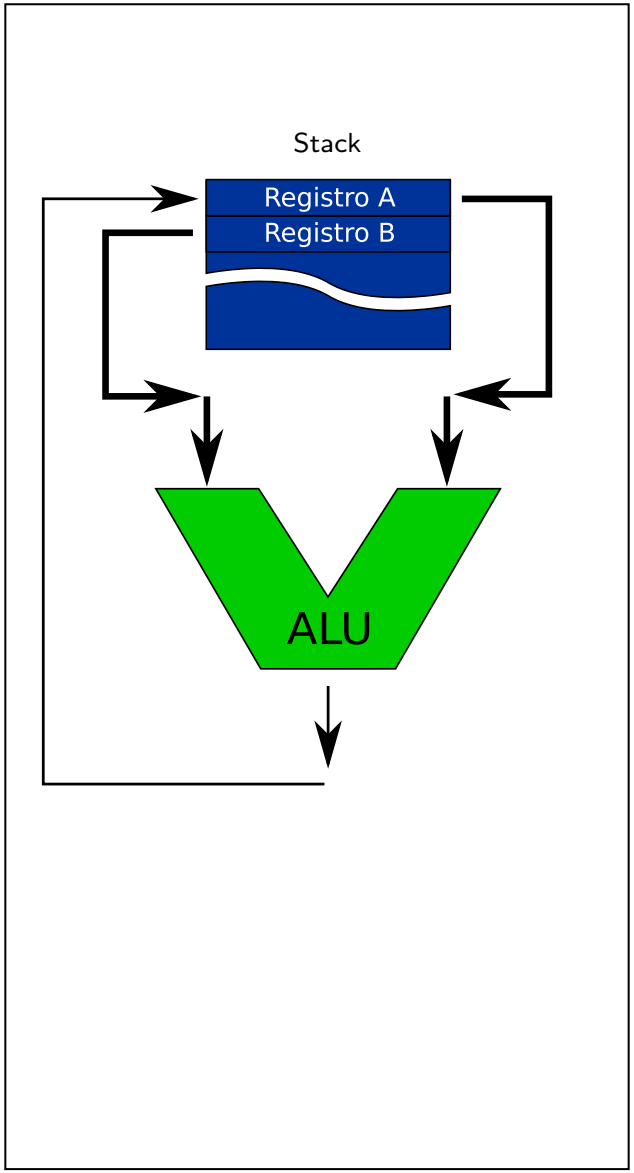
- STACK
- Registro
- Registro Memoria
- Load - Store



Tipos de Unidades Aritméticas Lógicas

Según el origen y el destino de sus datos.

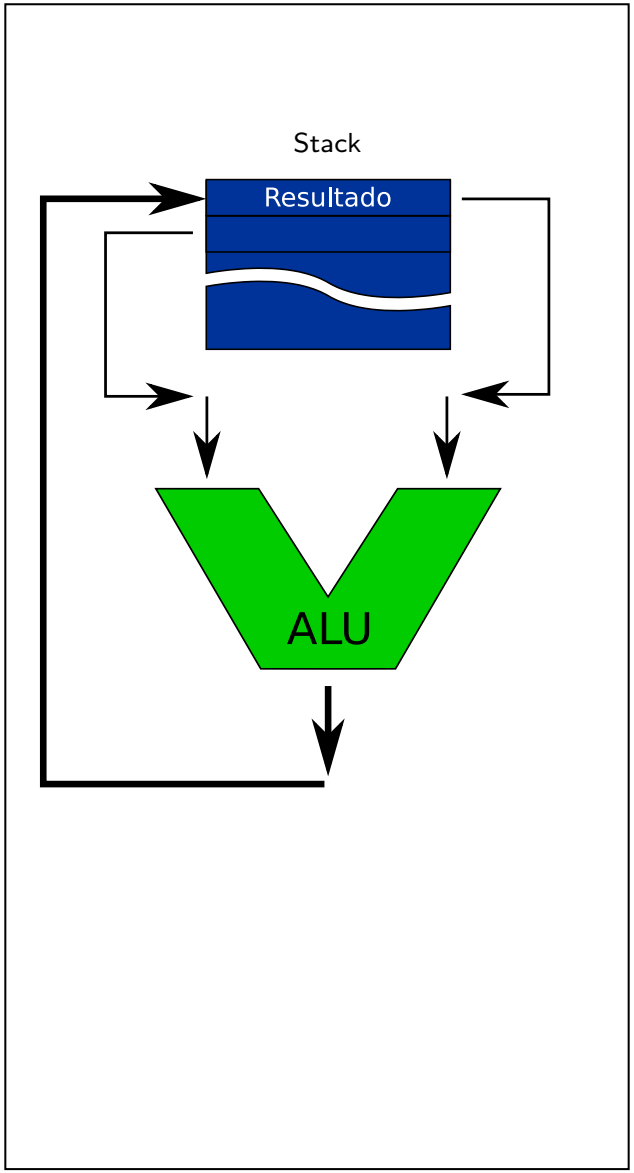
- STACK
- Registro
- Registro Memoria
- Load - Store



Tipos de Unidades Aritméticas Lógicas

Según el origen y el destino de sus datos.

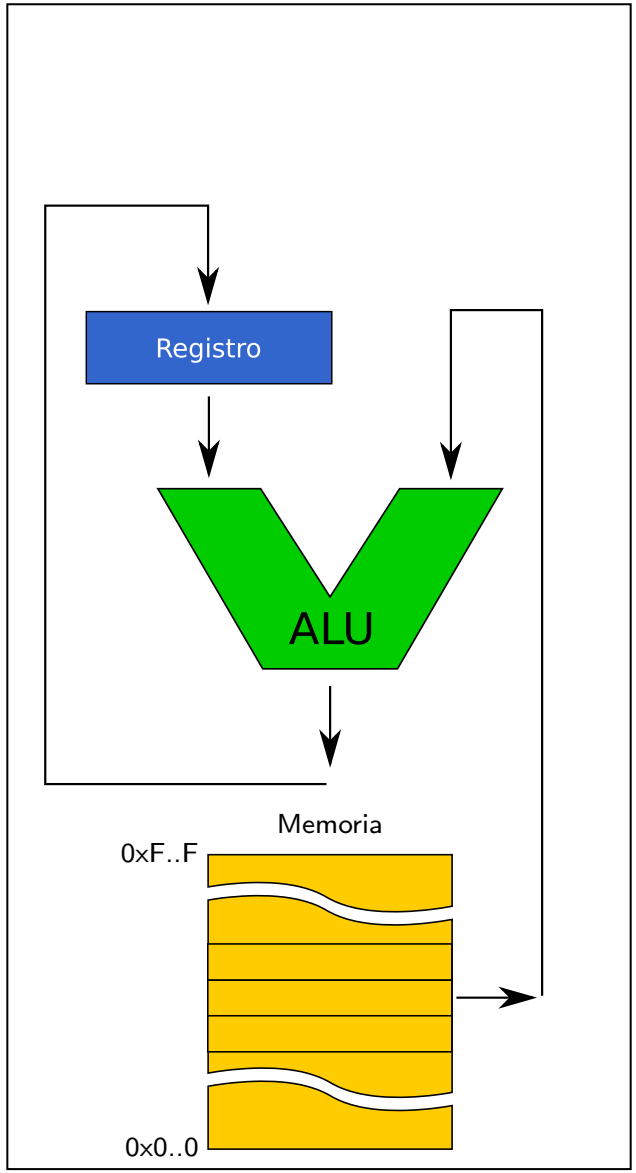
- STACK
- Registro
- Registro Memoria
- Load - Store



Tipos de Unidades Aritméticas Lógicas

Según el origen y el destino de sus datos.

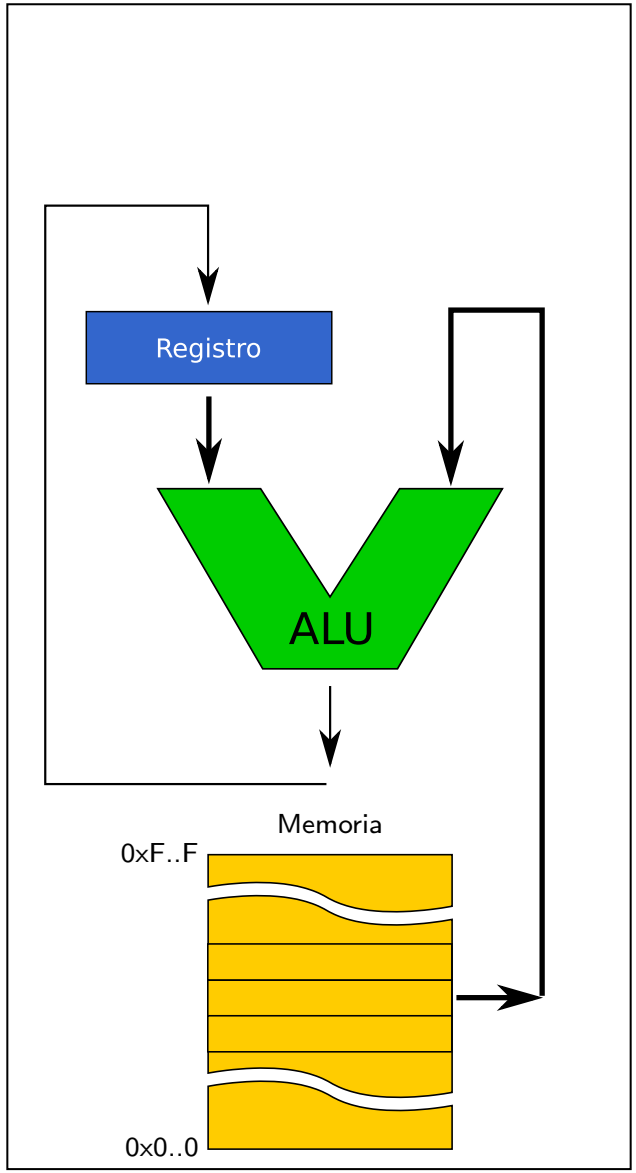
- STACK
- Registro
- Registro Memoria
- Load - Store



Tipos de Unidades Aritméticas Lógicas

Según el origen y el destino de sus datos.

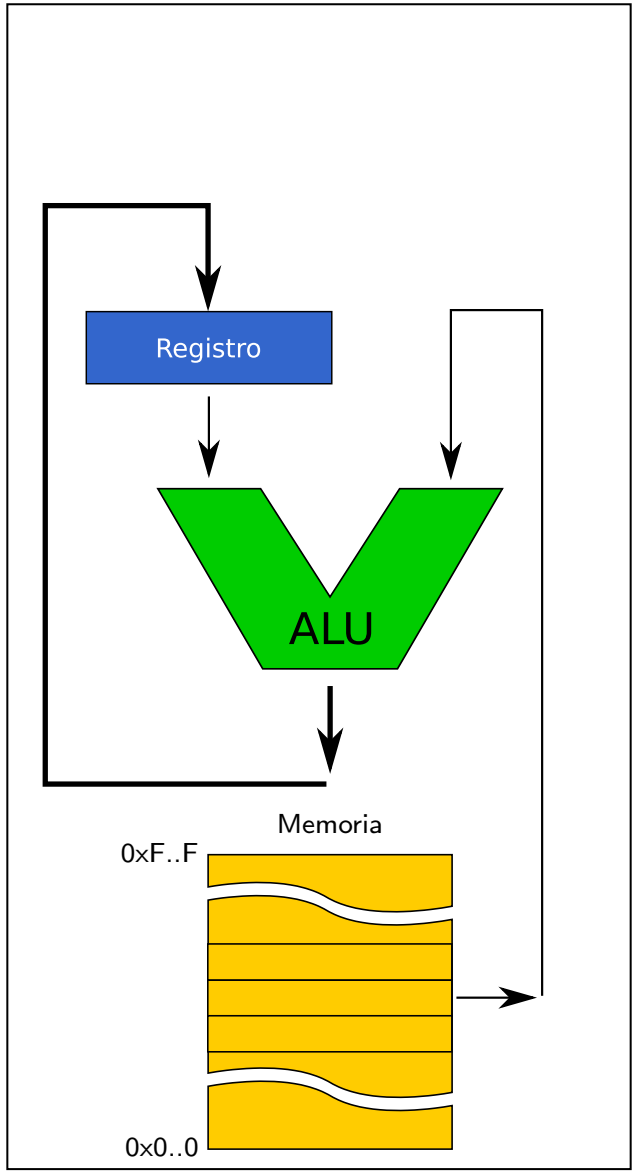
- STACK
- Registro
- Registro Memoria
- Load - Store



Tipos de Unidades Aritméticas Lógicas

Según el origen y el destino de sus datos.

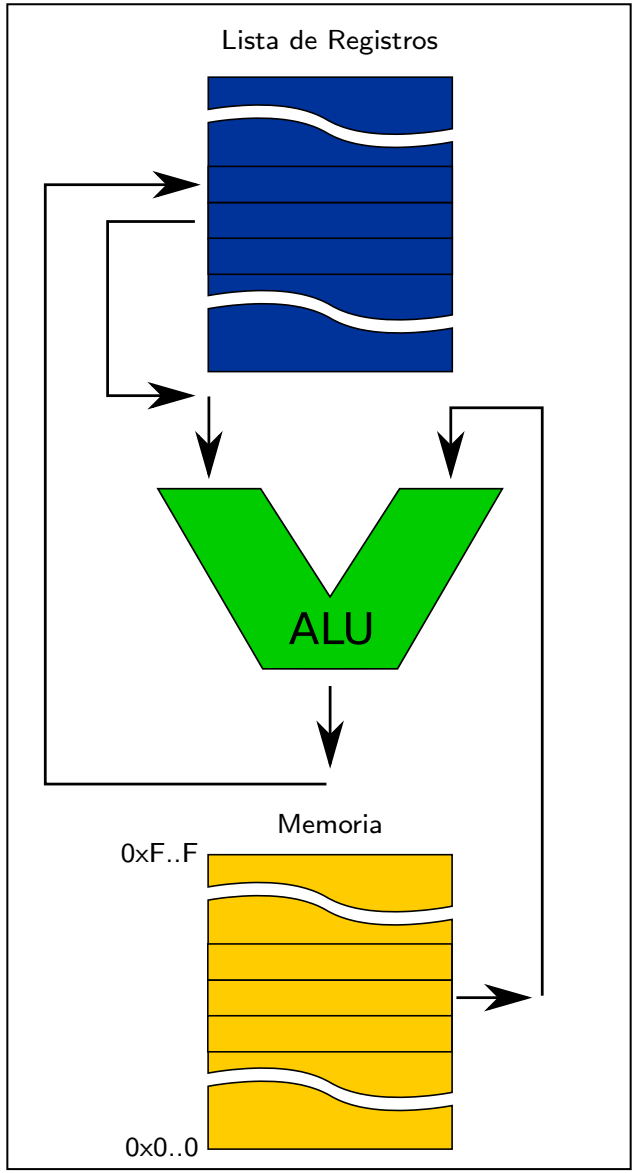
- STACK
- Registro
- Registro Memoria
- Load - Store



Tipos de Unidades Aritméticas Lógicas

Según el origen y el destino de sus datos.

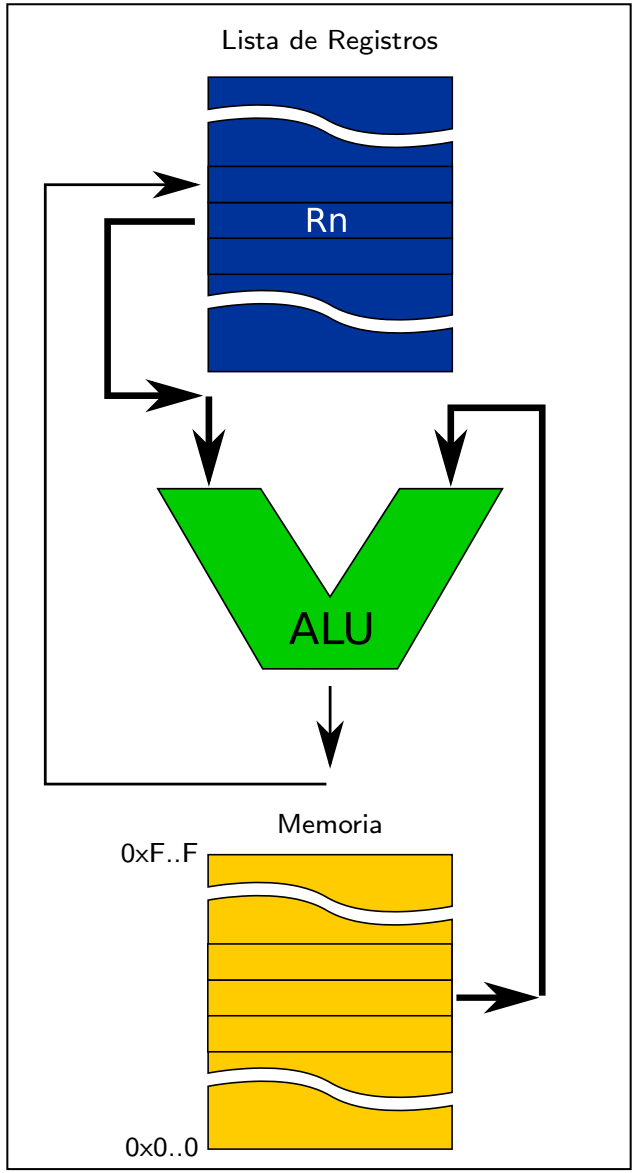
- STACK
- Registro
- Registro Memoria
- Load - Store



Tipos de Unidades Aritméticas Lógicas

Según el origen y el destino de sus datos.

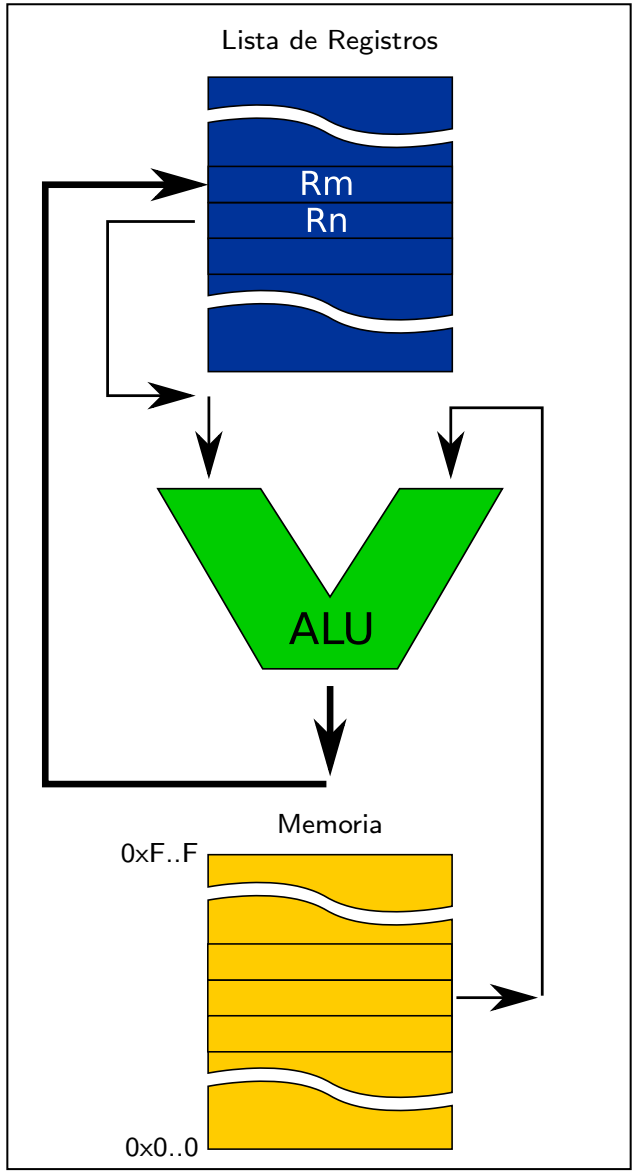
- STACK
- Registro
- Registro Memoria
- Load - Store



Tipos de Unidades Aritméticas Lógicas

Según el origen y el destino de sus datos.

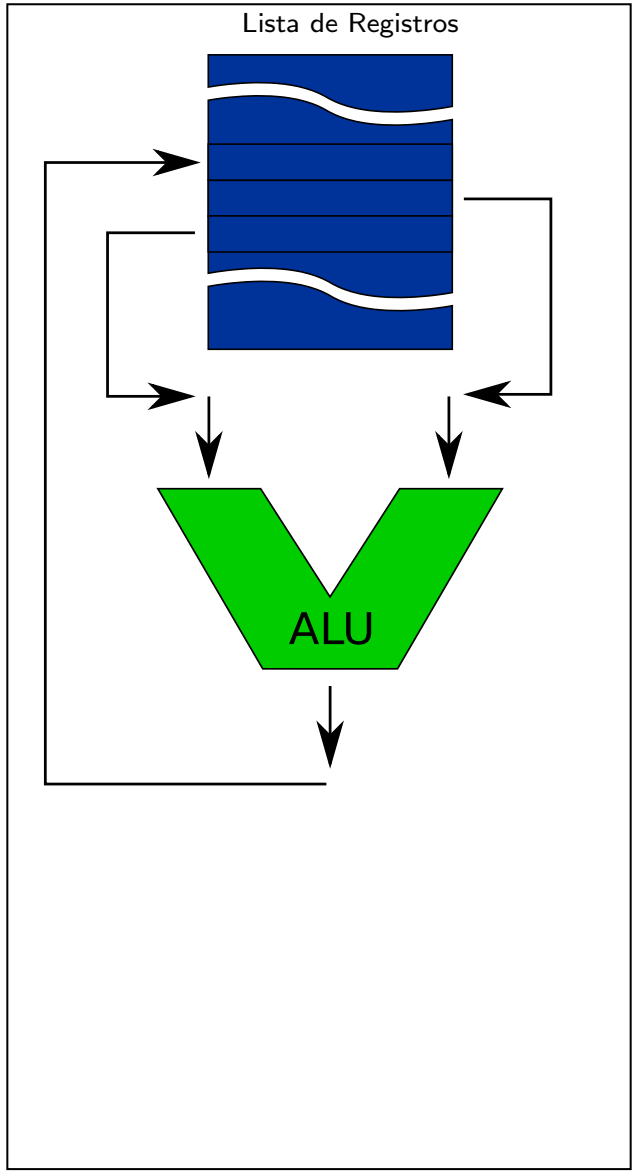
- STACK
- Registro
- Registro Memoria
- Load - Store



Tipos de Unidades Aritméticas Lógicas

Según el origen y el destino de sus datos.

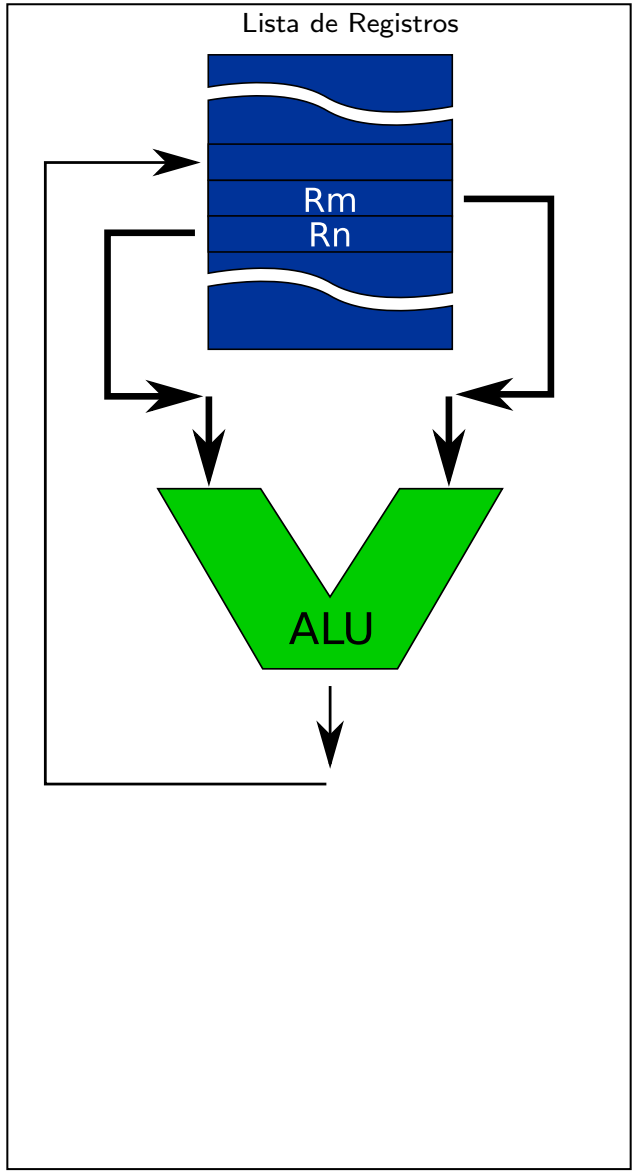
- STACK
- Registro
- Registro Memoria
- Load - Store



Tipos de Unidades Aritméticas Lógicas

Según el origen y el destino de sus datos.

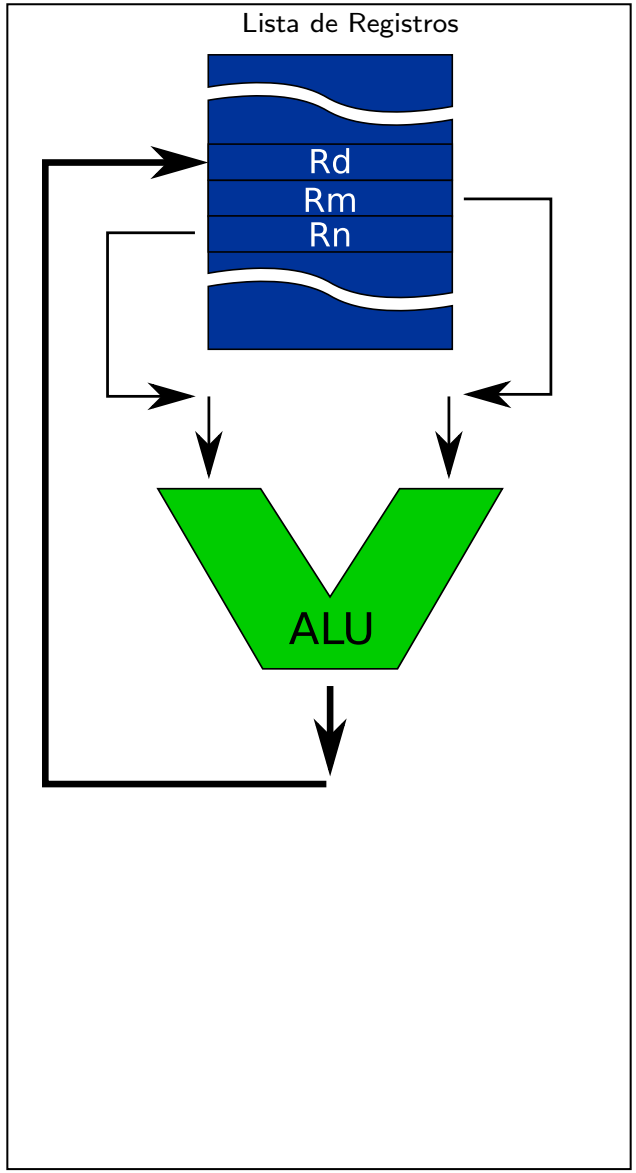
- STACK
- Registro
- Registro Memoria
- Load - Store



Tipos de Unidades Aritméticas Lógicas

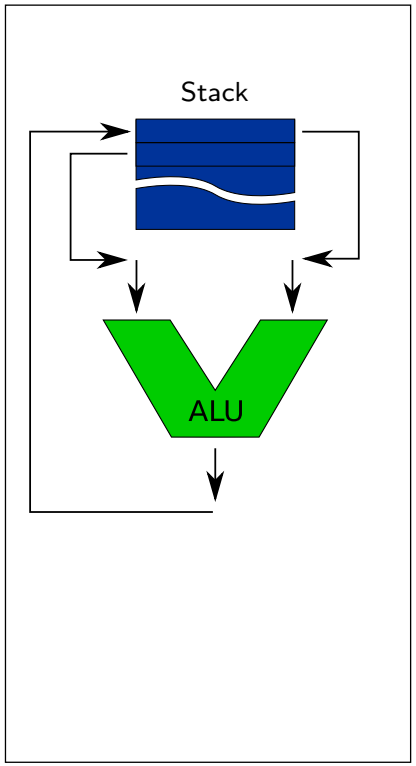
Según el origen y el destino de sus datos.

- STACK
- Registro
- Registro Memoria
- Load - Store



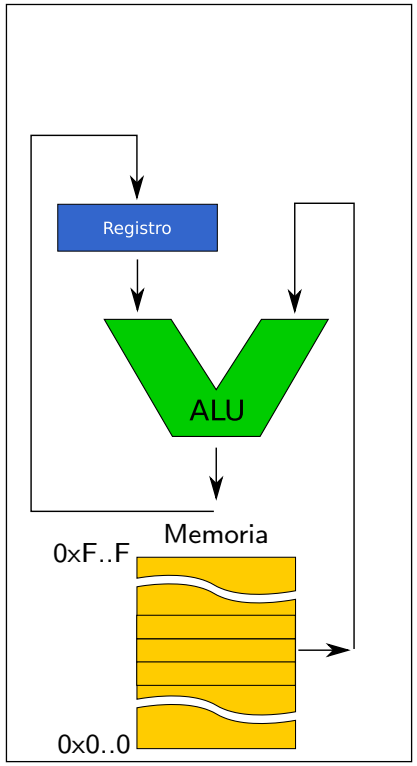
Tipos de Unidades Aritméticas Lógicas Ejemplo C=A+B

STACK



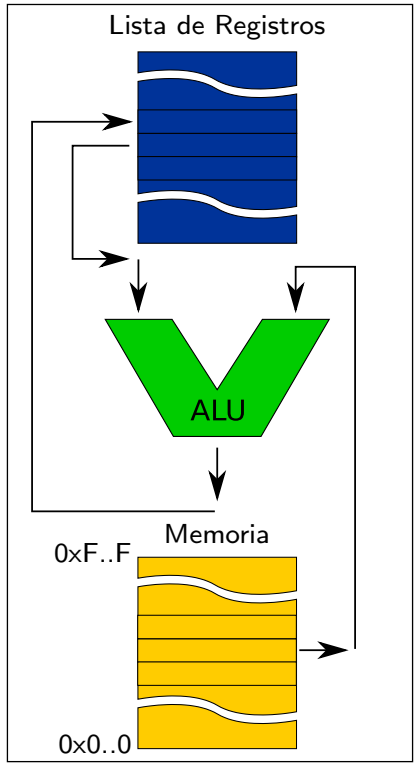

```
push VarA
push VarB
add
pop VarC
```

Registro



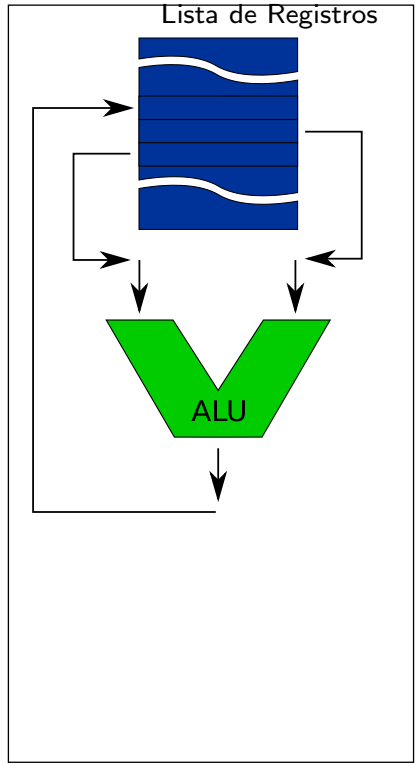

```
movf VarA
addwf VarB
movwf VarC
```

Registro Memoria




```
mov AX,VarA
add AX,VarB
mov VarC,AX
```

Load - Store




```
ldr r0,VarA
ldr r1,VarB
add r2,r1,r0
str r2,VarC
```

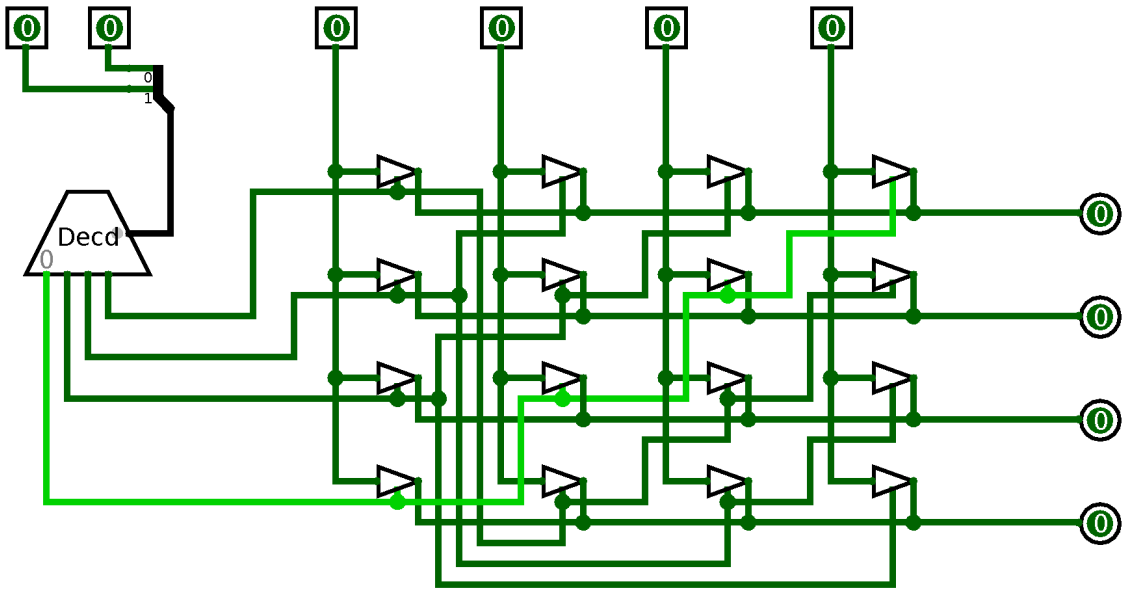
Barrel Shifter

- Circuito digital que permite desplazar una palabra binaria un número determinado de bits sin el uso de una secuencia lógica.
- Es un circuito realizado en lógica combinacional.
- Este circuito lógico permite acelerar procesos de normalización o generar potencias de 2 sin costo en ciclos de reloj.

Barrel Shifter

- Circuito digital que permite desplazar una palabra binaria un número determinado de bits sin el uso de una secuencia lógica.
- Es un circuito realizado en lógica combinacional.
- Este circuito lógico permite acelerar procesos de normalización o generar potencias de 2 sin costo en ciclos de reloj.

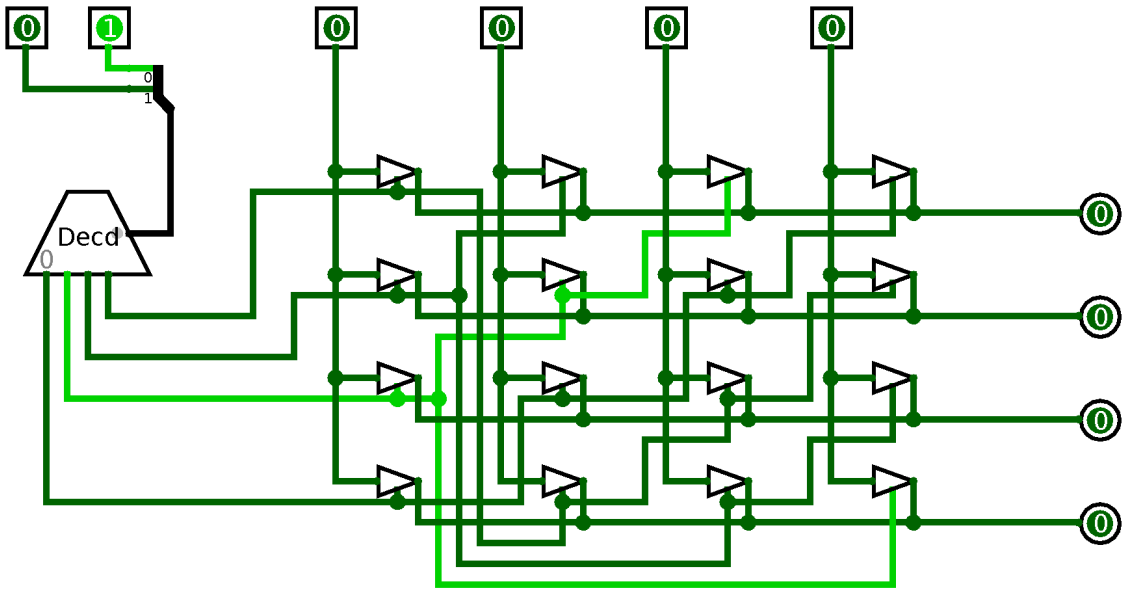
Ejemplo de shift de 0 bits



Barrel Shifter

- Circuito digital que permite desplazar una palabra binaria un número determinado de bits sin el uso de una secuencia lógica.
- Es un circuito realizado en lógica combinacional.
- Este circuito lógico permite acelerar procesos de normalización o generar potencias de 2 sin costo en ciclos de reloj.

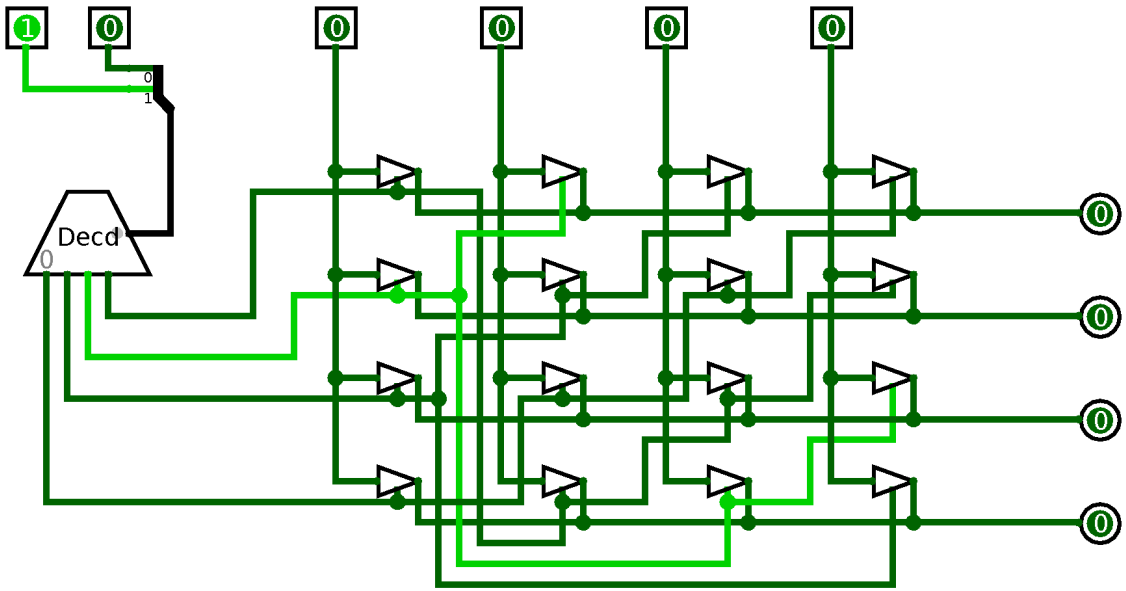
Ejemplo de shift de 1 bits



Barrel Shifter

- Circuito digital que permite desplazar una palabra binaria un número determinado de bits sin el uso de una secuencia lógica.
- Es un circuito realizado en lógica combinacional.
- Este circuito lógico permite acelerar procesos de normalización o generar potencias de 2 sin costo en ciclos de reloj.

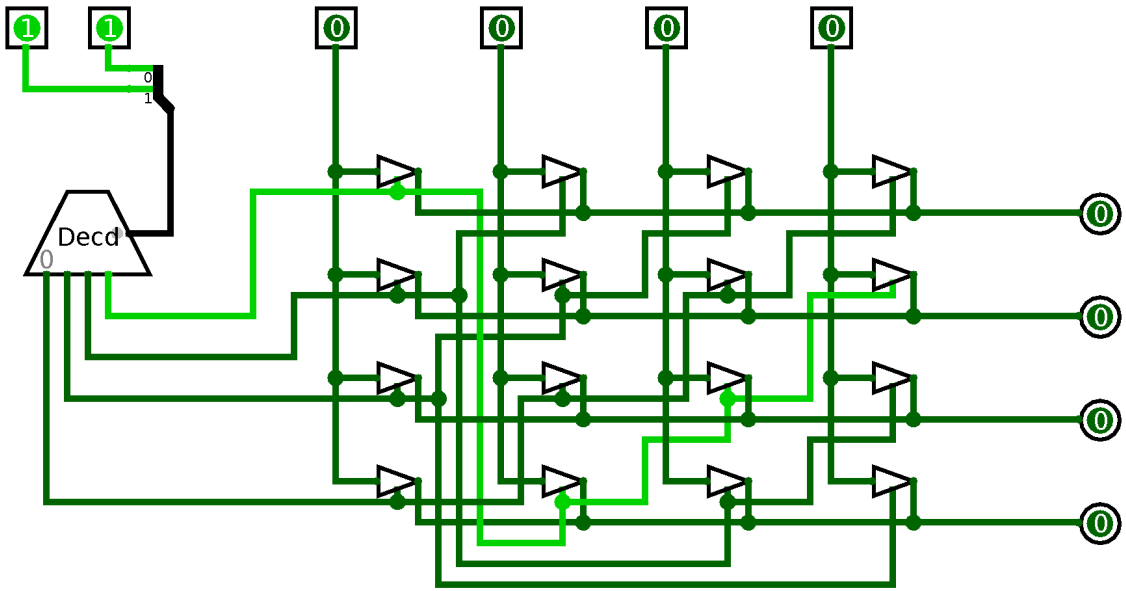
Ejemplo de shift de 2 bits



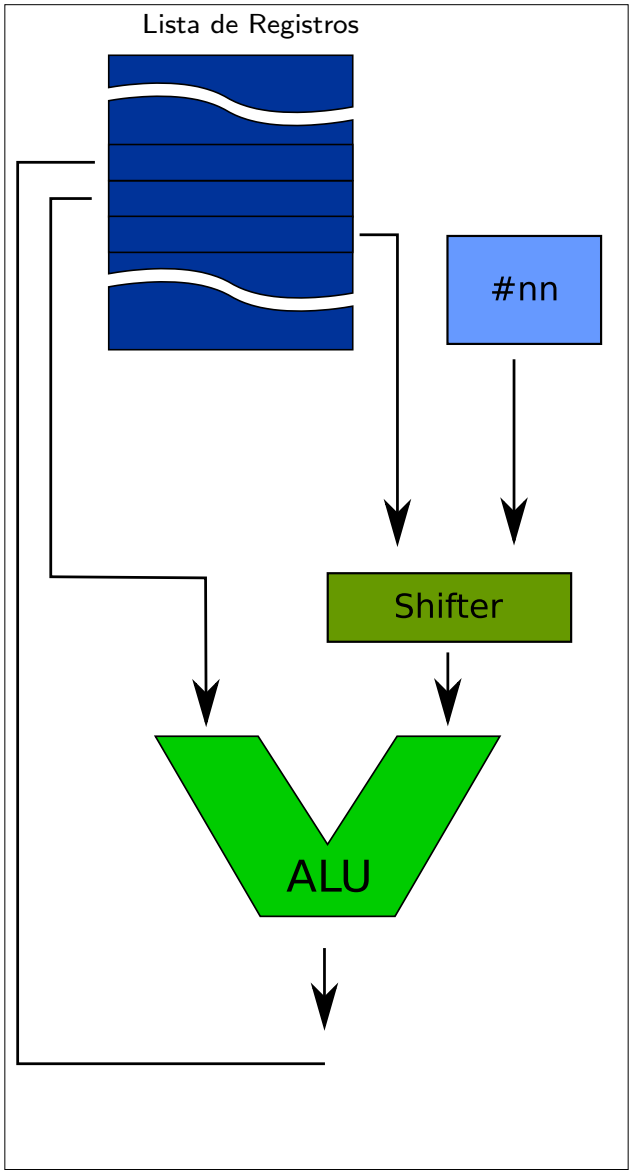
Barrel Shifter

- Circuito digital que permite desplazar una palabra binaria un número determinado de bits sin el uso de una secuencia lógica.
- Es un circuito realizado en lógica combinacional.
- Este circuito lógico permite acelerar procesos de normalización o generar potencias de 2 sin costo en ciclos de reloj.

Ejemplo de shift de 3 bits



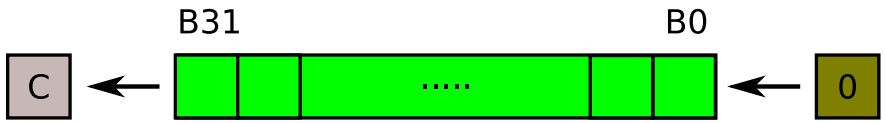
Modelo de la ALU en ARM



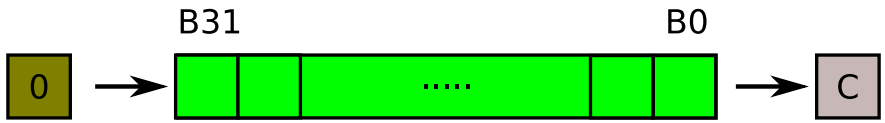
Barrel Shifter en el ARM

- 5 modos de desplazamiento
- Desplazamientos fijo `Rm LSL {valor}`
- Definido por otro registro `Rm LSL Rs`

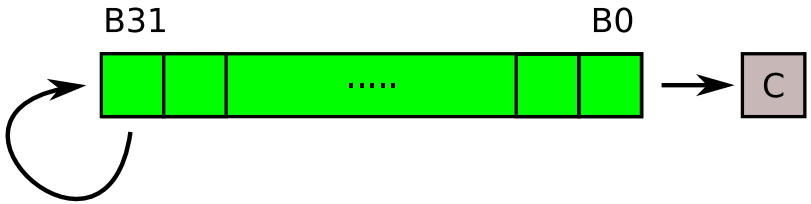
LSL - Corrimiento Lógico hacia la izquierda



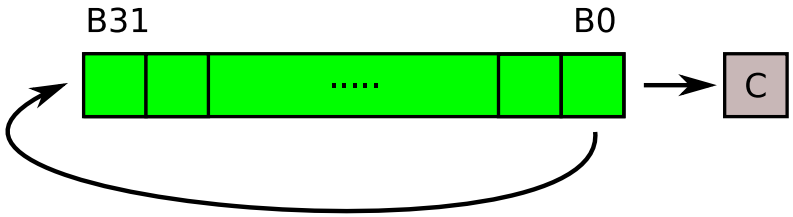
LSR - Corrimiento Lógico hacia la derecha



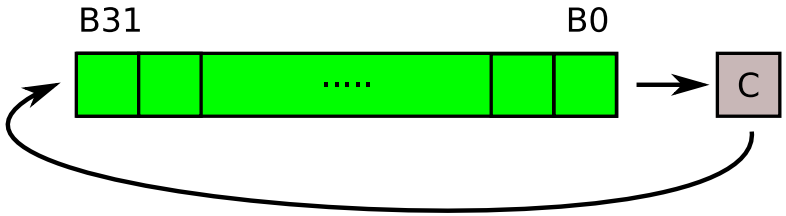
ASR - Corrimiento Aritmético hacia la derecha



ROR - Rotación hacia la derecha

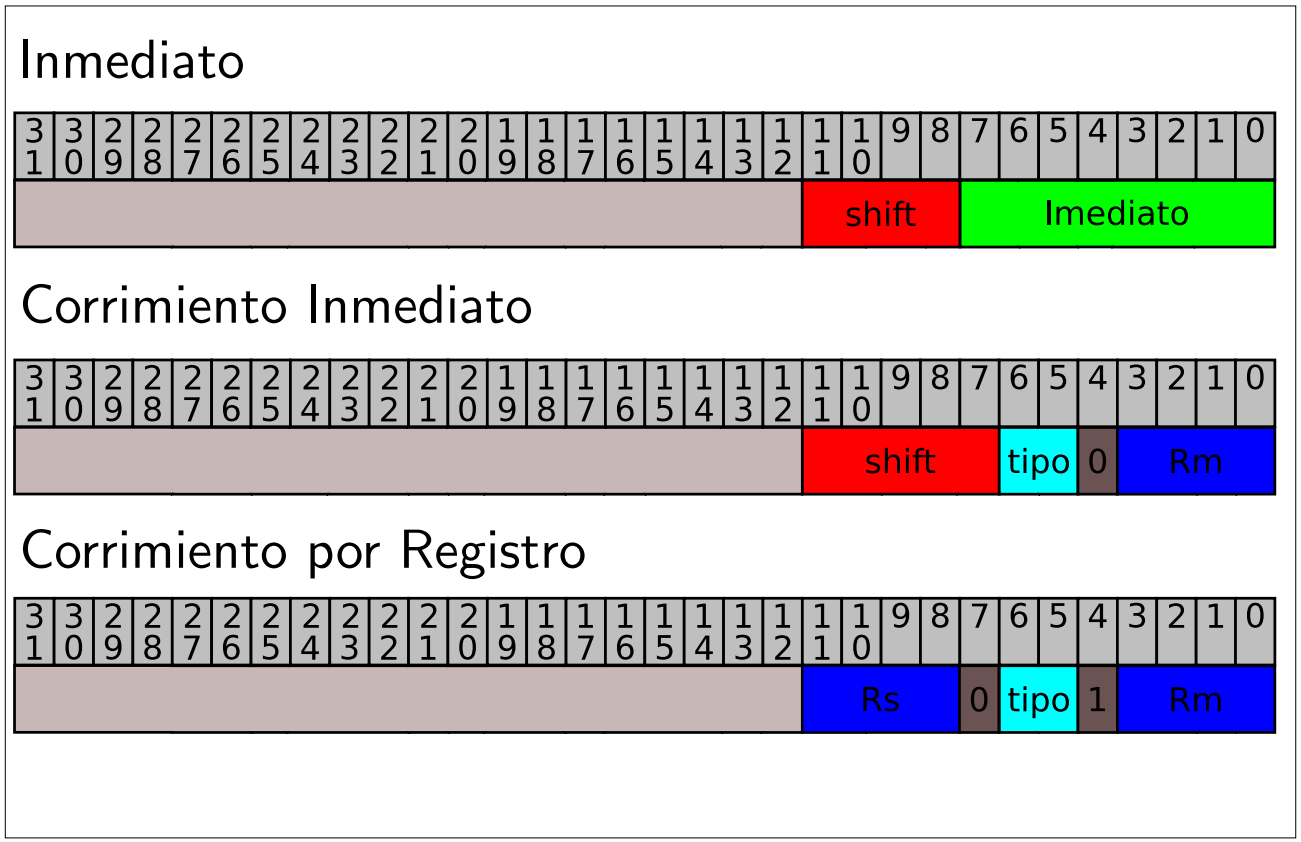


RRX - Rotación hacia la derecha extendido



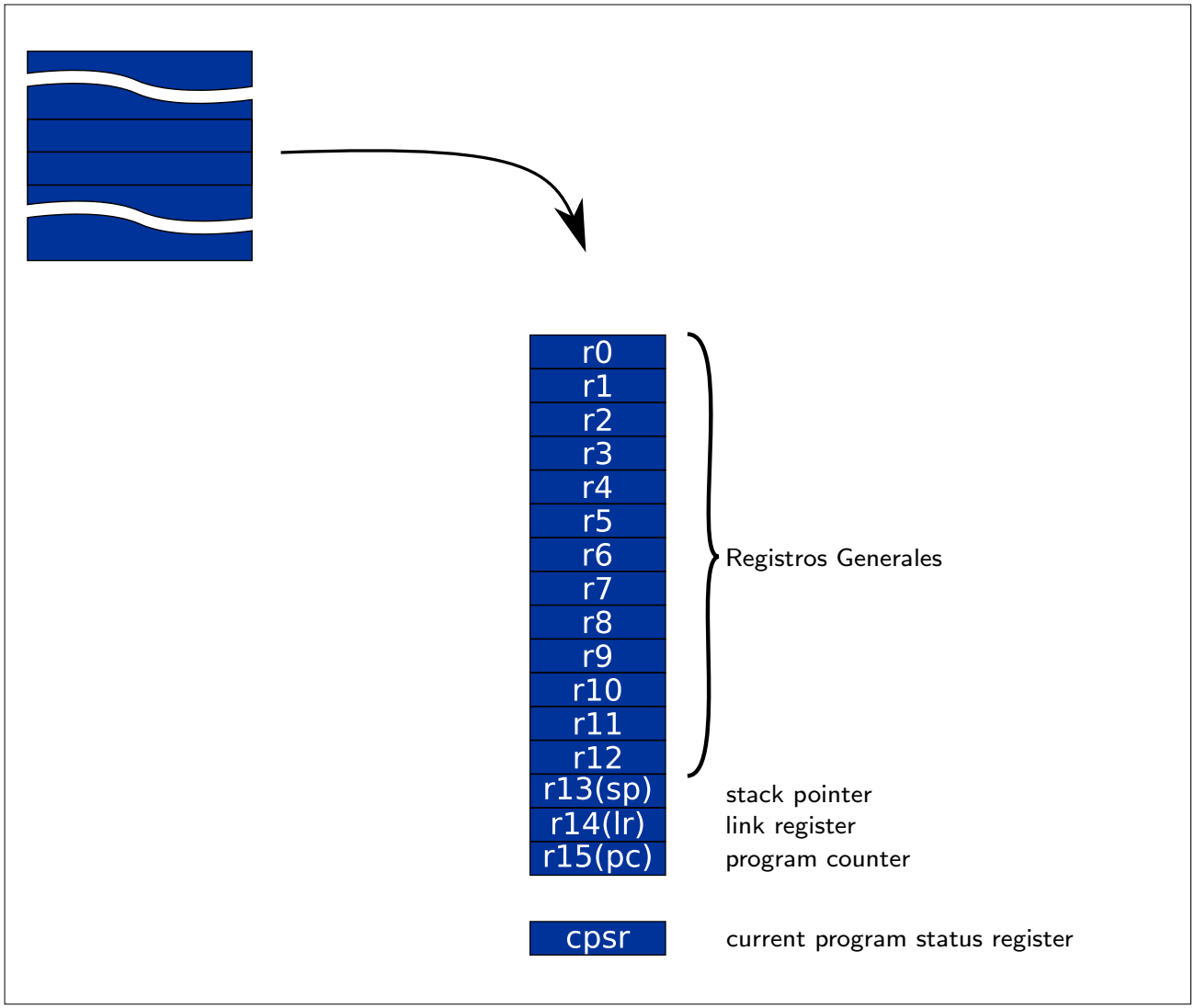
Barrel Shifter en el ARM y Operador 2

- 12 bits menos significativos en operaciones aritméticas, lógicas y de comparación.
- Se agrega la opción de un valor inmediato formado por 8 bits mas 4 de desplazamiento.



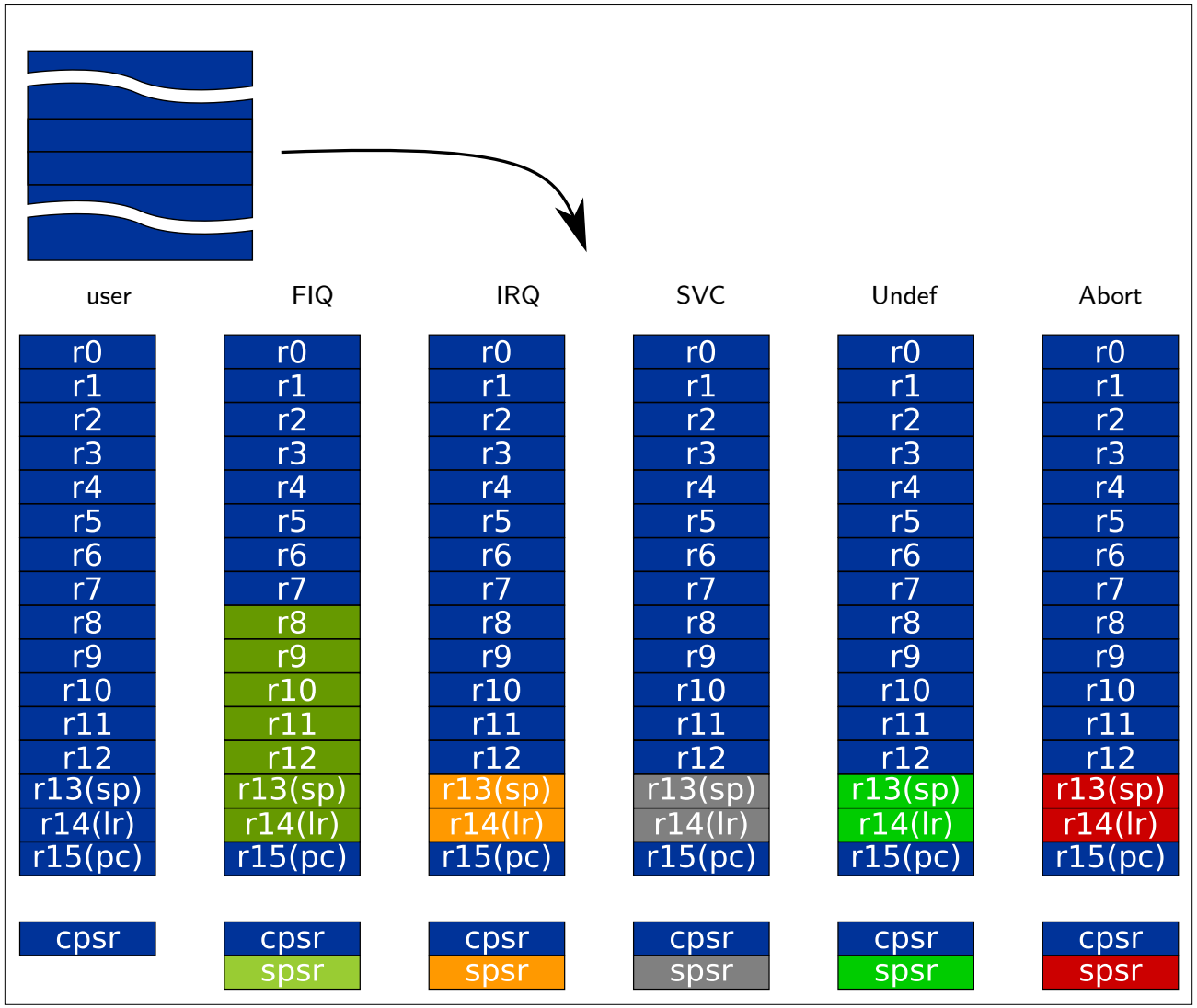
Registros en ARM

16 Registros (13 de propósitos generales y 3 para uso particular) + status



Registros en ARM

31 Registros generales, 1 status y 5 registros status de resguardo



Set de Instrucciones del ARM

- Instrucciones de Procesamiento de Datos.
- Instrucciones de Salto.
- Instrucciones de Transferencias del registro Status.
- Instrucciones para Carga y Escritura en Memoria.
- Instrucciones para Coprocesador
- Instrucciones para la Generación de Excepciones.

Realizan cálculos en los registros de propósitos generales.

- Instrucciones Aritméticas / Lógicas.
- Instrucciones de Comparación.
- Instrucciones de Multiplicación.

Set de Instrucciones del ARM

- Instrucciones de Procesamiento de Datos.
- Instrucciones de Salto.
- Instrucciones de Transferencias del registro Status.
- Instrucciones para Carga y Escritura en Memoria.
- Instrucciones para Coprocesador
- Instrucciones para la Generación de Excepciones.

Instrucciones que cambian el flujo del programa.

- Instrucciones genéricas que cambian el pc
- Instrucciones especificas de salto (provee un número de 24bit con signo como corrimiento, permitiendo realizar saltos de +/-32Mb.

Set de Instrucciones del ARM

- Instrucciones de Procesamiento de Datos.
- Instrucciones de Salto.
- Instrucciones de Transferencias del registro Status.
- Instrucciones para Carga y Escritura en Memoria.
- Instrucciones para Coprocesador
- Instrucciones para la Generación de Excepciones.

Transfieren desde o hacia CPSR o SPSR con un registro de propósitos múltiples, esto permite:

- Establecer el valor de las banderas de condición.
- Establecer el valor de los bits de interrupciones.
- Establecer el modo y estado del procesador.

Set de Instrucciones del ARM

- Instrucciones de Procesamiento de Datos.
- Instrucciones de Salto.
- Instrucciones de Transferencias del registro Status.
- Instrucciones para Carga y Escritura en Memoria.
- Instrucciones para Coprocesador
- Instrucciones para la Generación de Excepciones.

Realizan transferencias de datos entre los registros de propósitos múltiples y la memoria. Se dividen en:

- Carga y Almacenamiento de Registros.
- Carga y Almacenamiento de Registros Múltiples.

Set de Instrucciones del ARM

- Instrucciones de Procesamiento de Datos.
- Instrucciones de Salto.
- Instrucciones de Transferencias del registro Status.
- Instrucciones para Carga y Escritura en Memoria.
- Instrucciones para Coprocesador
- Instrucciones para la Generación de Excepciones.

Set de Instrucciones del ARM

- Instrucciones de Procesamiento de Datos.
- Instrucciones de Salto.
- Instrucciones de Transferencias del registro Status.
- Instrucciones para Carga y Escritura en Memoria.
- Instrucciones para Coprocesador
- Instrucciones para la Generación de Excepciones.

Existen dos tipos de instrucciones designadas para causar una excepción específica.

- Instrucciones de interrupción por software (SWI).
- Instrucción de Breakpoint por Software.

Banderas y Condicionales

Banderas

Banderas de código de condición o (condition code flags) ubicadas en el PSR pueden ser modificadas por cualquier instrucción de Procesamiento de Datos

- N Bandera de Negativo (Bit 31 del resultado)
- Z Bandera de Cero (1 si todos los bits del resultado son 0)
- C Bandera de Acarreo (acarreo en una suma, préstamo en resta o bit desplazado en instrucciones shift)
- V Bandera de Desborde (desborde en operación con signo)

Banderas y Condicionales

Condicional

Todas las instrucciones poseen un campo de 4 bits (bits 28-31) que permite condicionar la ejecución de la instrucción de acuerdo al estado de las banderas del CPSR.

Las condiciones están dadas por la siguiente tabla:

Code	Mn	Significado	Condición	Code	Mn	Significado	Condición
0000	EQ	Igual	$Z = 1$	0001	NE	Distintos	$Z = 0$
0010	CS	Carry Set	$C = 1$	0011	CC	Carry Clear	$C = 0$
0110	VS	Desborde	$V = 1$	0111	VC	Sin desb.	$V = 0$
0100	MI	Negativo	$N = 1$	0101	PL	Positivo	$N = 0$
1000	HI	Sobre	$C = 1$ y $Z = 0$	0011	LO	Debajo	$C = 0$
0010	HS	Sobre o =	$C = 1$	1001	LS	Debajo o =	$C = 0$ o $Z = 1$
1100	GT	Mayor	$Z = 0$ y $N = V$	1011	LT	Menor	$N <> V$
1010	GE	Mayor o =	$N = V$	1101	LE	Menor o =	$Z = 1$ o $N <> V$
1110	AL	Sin Cond.					
1111		No Usado					

Instrucciones Aritméticas, Lógicas y de Comparación

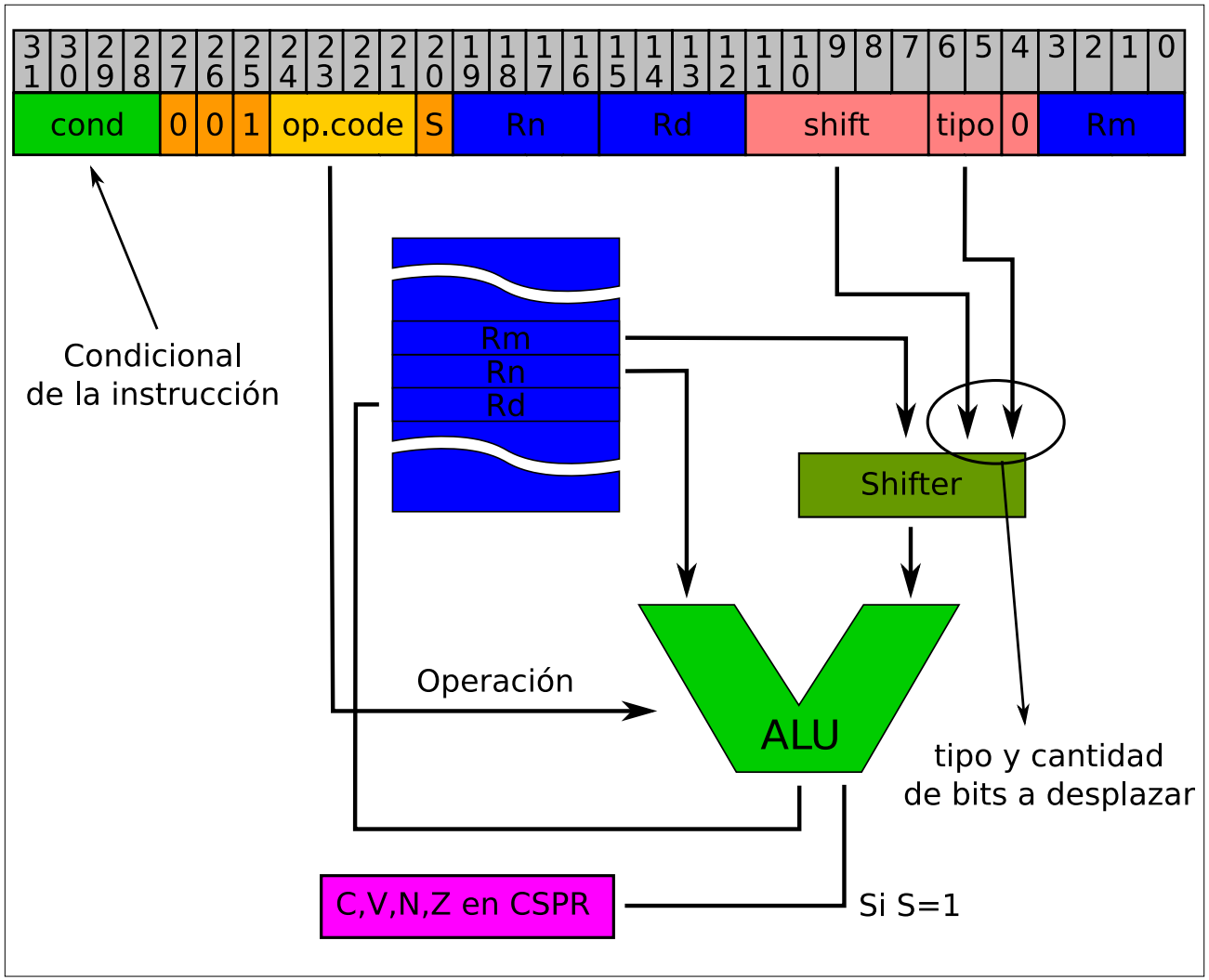
- Realizan operaciones aritméticas y lógicas en uno o dos op. y escriben el resultado en un registro destino pudiendo o no actualizar las banderas.
- Realizan operaciones de comparación entre dos op. y solo actualizan las banderas.

El primer operador debe ser siempre un registro, el otro operador puede ser un valor inmediato o un registro al que se le puede aplicar un corrimiento.

Opcode	Mnemonico	Operación	Acción	Asembler
0000	AND	AND lógica	$Rd = Rn$ and $Op2$	`AND{cond}{S} Rd,Rn,Op2`
0001	EOR	Or Exclusiva	$Rd = Rn$ eor $Op2$	`EOR{cond}{S} Rd,Rn,Op2`
0010	SUB	Resta	$Rd = Rn - Op2$	`SUB{cond}{S} Rd,Rn,Op2`
0011	RSB	Resta reversa	$Rd = Op2 - Rn$	`RSB{cond}{S} Rd,Rn,Op2`
0100	ADD	Suma	$Rd = Rn + Op2$	`ADD{cond}{S} Rd,Rn,Op2`
0101	ADC	Suma con Acarreo	$Rd = Rn + Op2 + C$	`ADC{cond}{S} Rd,Rn,Op2`
0110	SBC	Resta con Acarreo	$Rd = Rn - Op2 - \text{not}(C)$	`SBC{cond}{S} Rd,Rn,Op2`
0111	RSC	Resta rev.con Ac.	$Rd = Op2 - Rn - \text{not}(C)$	`RSC{cond}{S} Rd,Rn,Op2`
1000	TST	Test	Band. de $(Rn$ and $Op2)$	`TST{cond} Rn,Op2`
1001	TEQ	Test Equivalente	Band. de $(Rn$ eor $Op2)$	`TEQ{cond} Rn,Op2`
1010	CMP	Comparar	Band. de $(Rn - Op2)$	`CMP{cond} Rn,Op2`
1011	CMN	Comparar Negativo	Band. de $(Rn + Op2)$	`CMN{cond} Rn,Op2`
1100	ORR	Or Inclusiva	$Rd = Rn$ o r$Op2$	`ORR{cond}{S} Rd,Rn,Op2`
1101	MOV	Mover	$Rd = Op2$	`MOV{cond}{S} Rd,Op2`
1110	BIC	Borrar Bit	$Rd = Rn$ and not$(Op2)$	`BIC{cond}{S} Rd,Rn,Op2`
1111	MVN	Mover negado	$Rd =$ not$(Op2)$	`MVN{cond}{S} Rd,Op2`

Instrucciones Aritméticas, Lógicas y de Comparación

Instrucciones genérica de 3 registros



Instrucciones Aritméticas, Lógicas y de Comparación

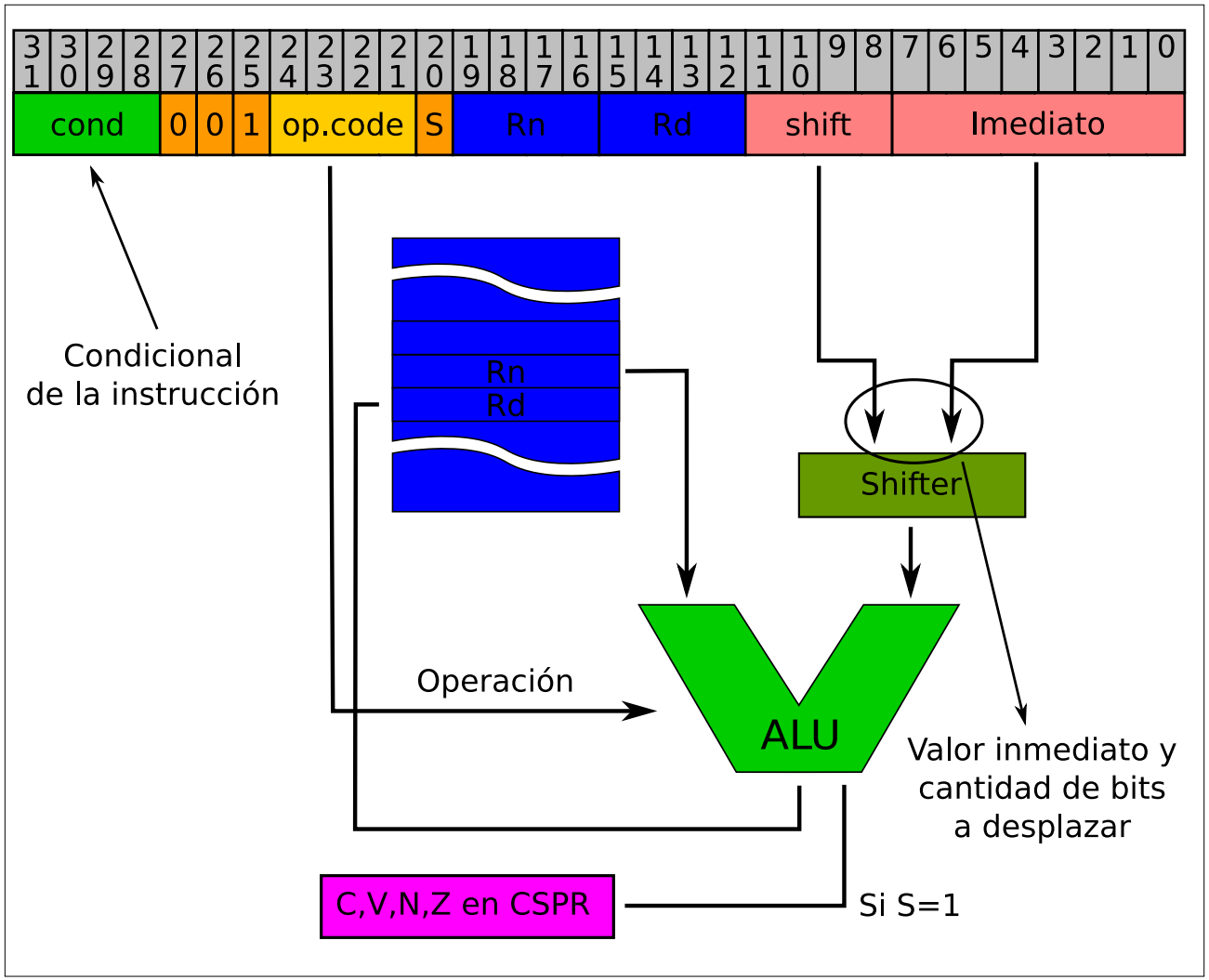
Instrucciones genérica de 3 registros

Ejemplos

- `adds R1,R2,R3` equiv. $R1 = R2 + R3$ (update flags)
- `add R1, R2, R2, lsl 3` equiv. $R1 = R2 + R2 \cdot 2^3 = R2 \cdot 9$
- `cmp R1,R2` equiv. $R1 - R2$ (update flags)
- `mov R1,R2` equiv. $R1 = R2$

Instrucciones Aritméticas, Lógicas y de Comparación

Instrucción aritmética genérica de 2 registros + inmediato



Instrucciones Aritméticas, Lógicas y de Comparación

Instrucción aritmética genérica de 2 registros + inmediato

Ejemplos

- `subs R1,R1,#1` equiv. $R1 = R1 - 1$ (update flags)
- `rsb R1,R1,#0` equiv. $R1 = 0 - R1$
- `and R1,R1,#3` equiv. $R1 = R1$ and 3 o dejar los 2 bits menos sign.
- `cmp R1,0` equiv. $R1 - 0$ (update flags)

Instrucciones de Salto

Saltos

- Modificando directamente el registro R15 (PC) con una operación de movimiento o carga
- Utilizando la operación especifica de Salto

- `mov pc,#0`
- `mov pc,lr`
- `ldr pc,=func`

Instrucciones de Salto

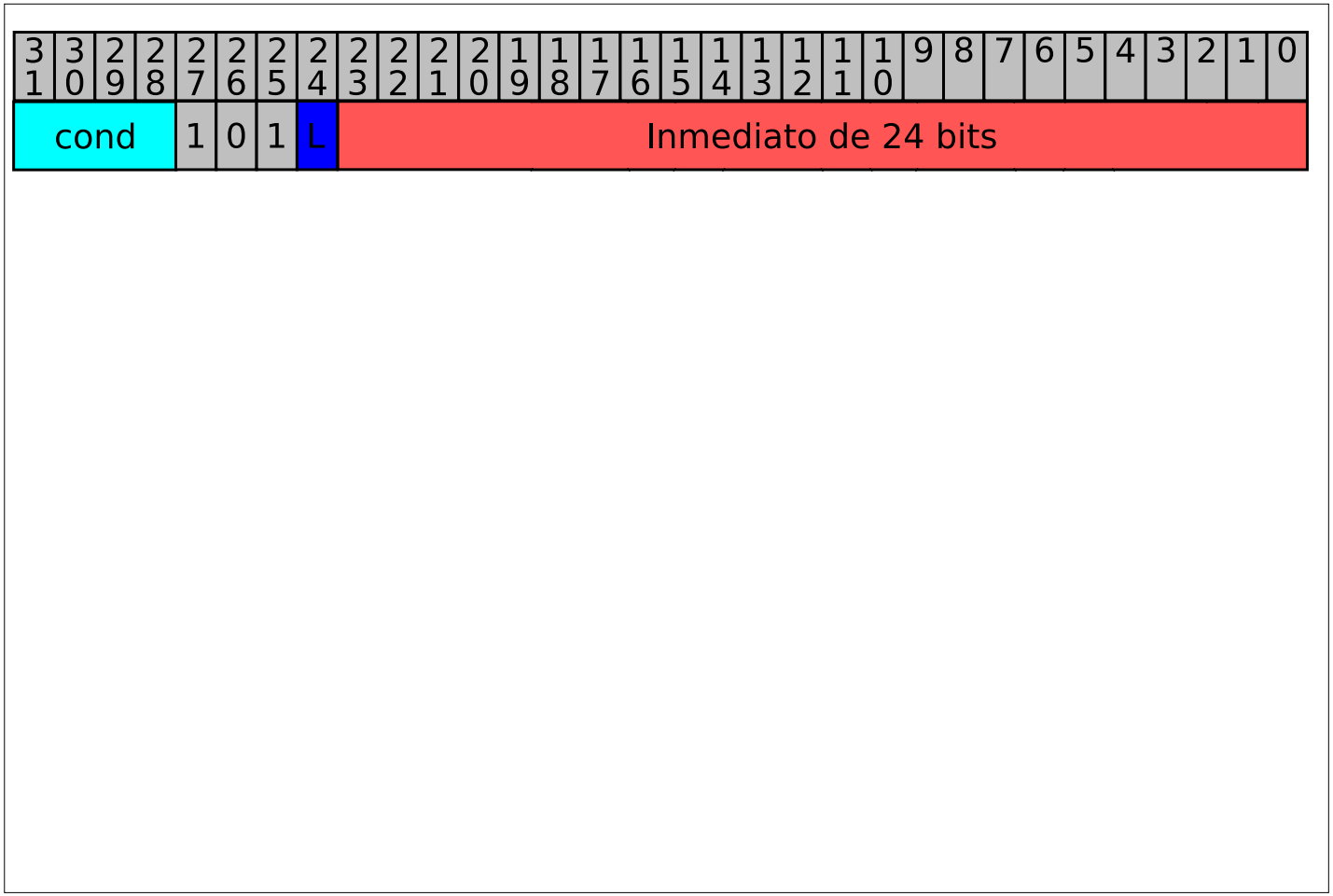
Saltos

- Modificando directamente el registro R15 (PC) con una operación de movimiento o carga
- Utilizando la operación especifica de Salto

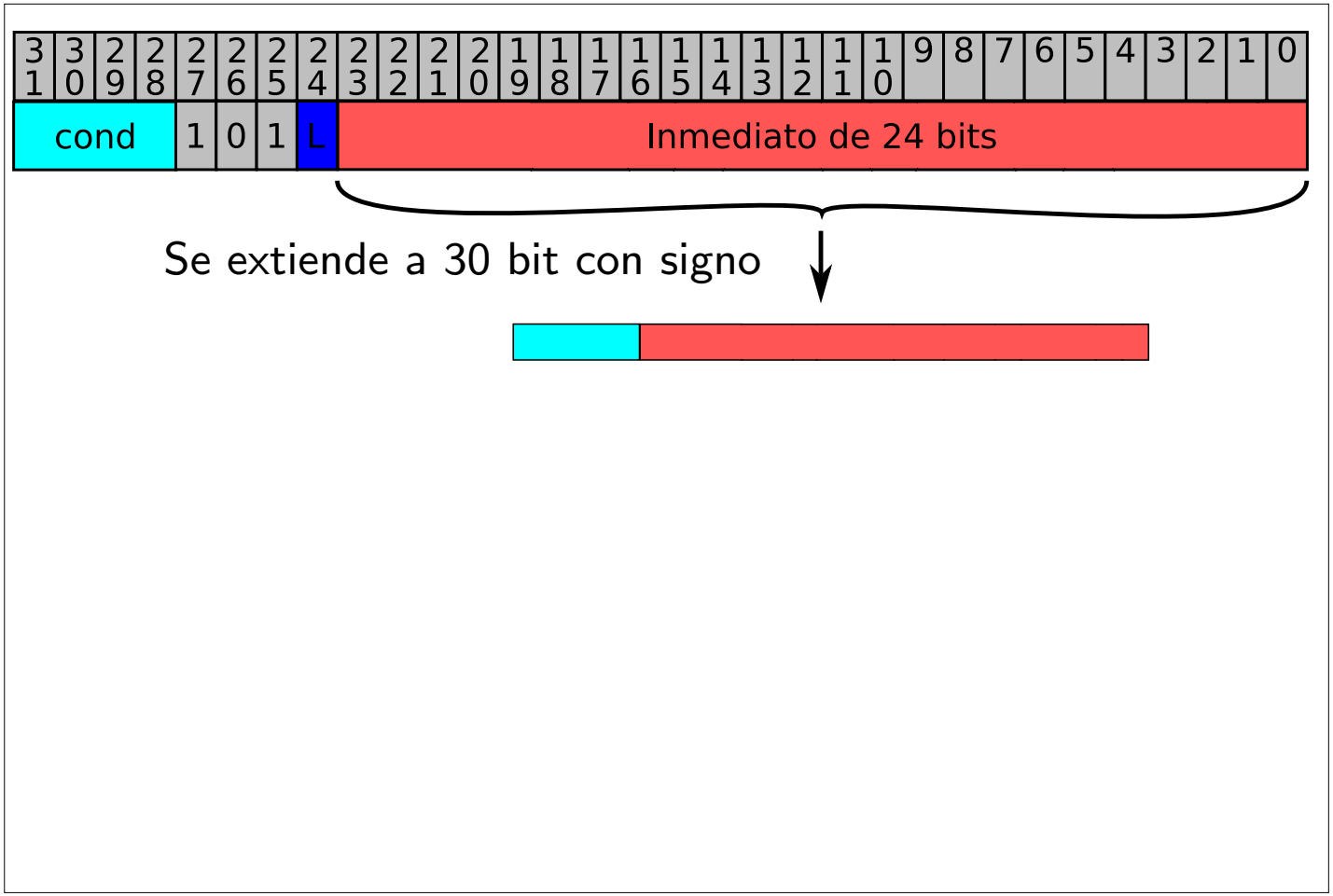
Las instrucciones permiten realizar saltos con una distancia máxima de $+/-32MB$

- Salto convencional a dirección destino, este salto puede ser condicionado o no
  `B{cond} <direccion>`
- Salto conservando el vinculo a dirección destino, con las mismas características del anterior permitiendo además guardar la dirección de retorno en el registro R14 (LR)
  `BL{cond} <direccion>`

Formato de la instrucciones de Salto

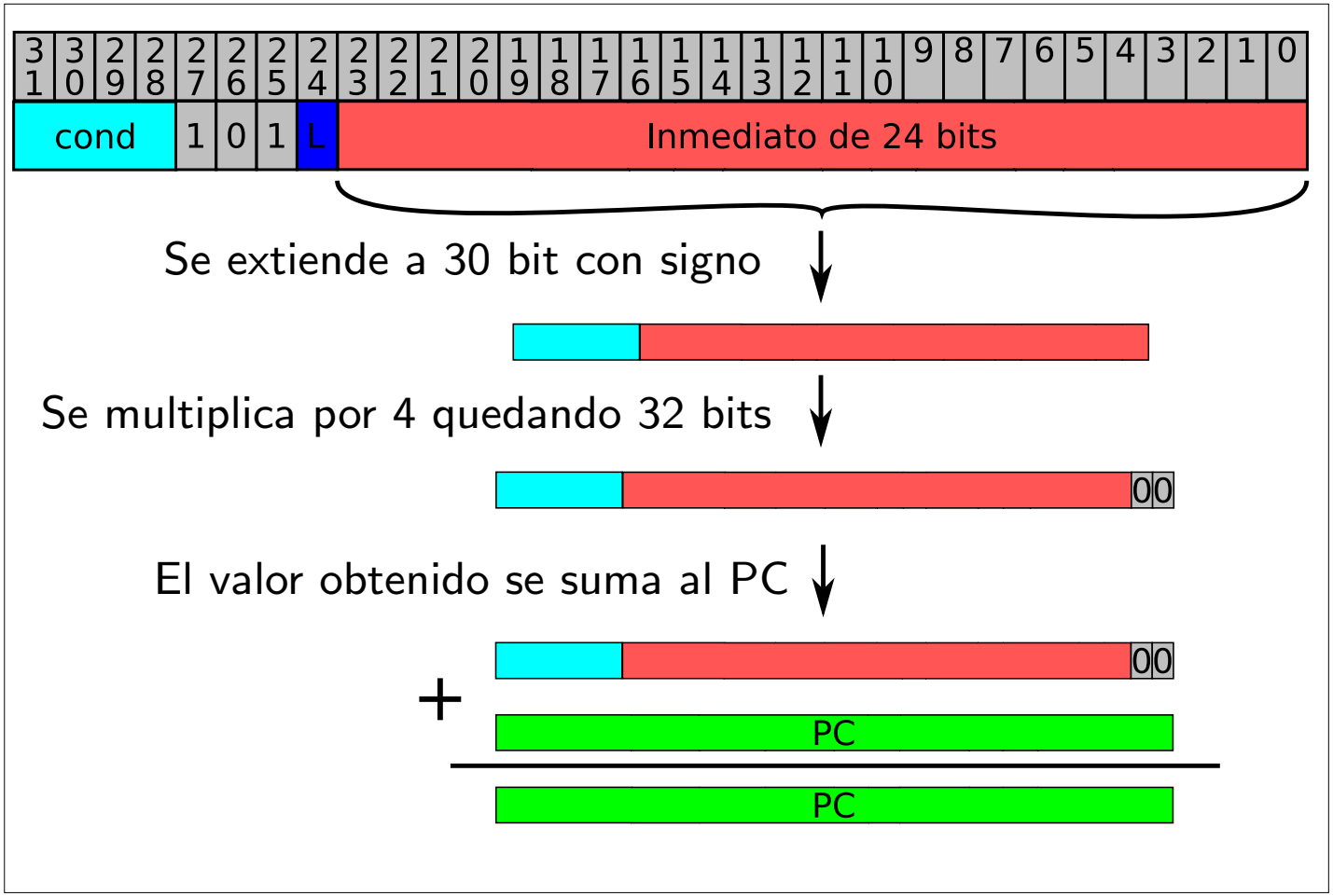


Formato de la instrucciones de Salto

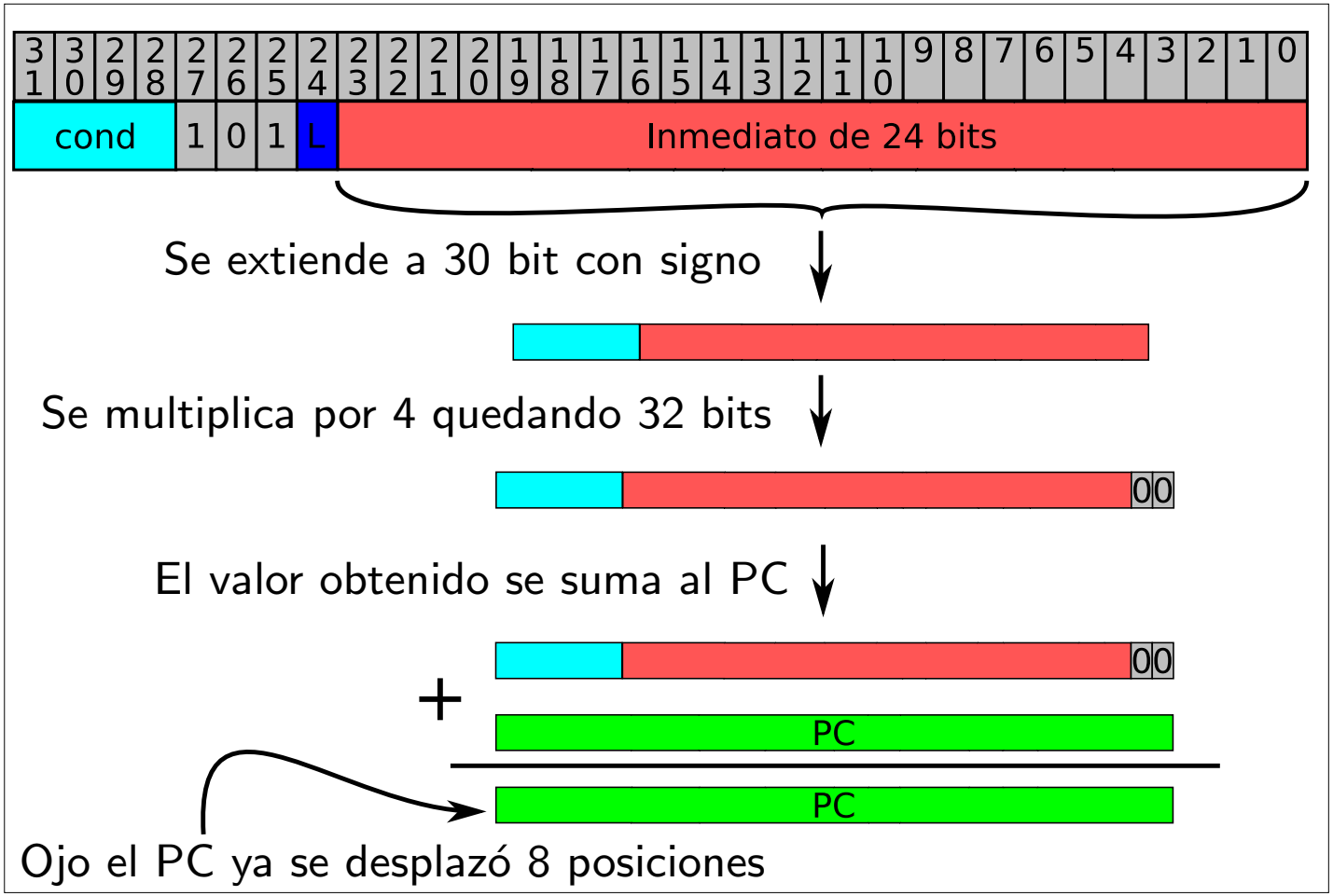


Formato de la instrucciones de Salto

Formato de la instrucciones de Salto



Formato de la instrucciones de Salto



Ejemplo de salto

```
bucle:  add r2,r1      44:   e0822001        add     r2, r2, r1
        cmp r2,#100    48:   e3520064        cmp     r2, #100; 0x64
        bhi salir      4c:   8a000001        bhi     58 <salir>
        subs r2,#1     50:   e2522001        subs    r2, r2, #1
        bne  bucle     54:   1affffa         bne     44 <bucle>
salir:  mov r3,r2      58:   e1a03002        mov     r3, r2
        add r2,#1      5c:   e2822001        add     r2, r2, #1
        add r2,#2      60:   e2822002        add     r2, r2, #2
```

`bhi salir`

- 000001_{16} a 1_{10}
- $1 \cdot 4 = 4$
- $pc = 4c_{16}$ o $pc = 76_{10}$
- $pc = pc + 4$ reemplazando $pc = (76 + 8) + 4 = 88$ o 58_{16}

Ejemplo de salto

```
bucle:  add r2,r1      44:   e0822001        add     r2, r2, r1
        cmp r2,#100    48:   e3520064        cmp     r2, #100; 0x64
        bhi salir      4c:   8a000001        bhi     58 <salir>
        subs r2,#1     50:   e2522001        subs    r2, r2, #1
        bne  bucle     54:   1affffa         bne     44 <bucle>
salir:  mov r3,r2      58:   e1a03002        mov     r3, r2
        add r2,#1      5c:   e2822001        add     r2, r2, #1
        add r2,#2      60:   e2822002        add     r2, r2, #2
```

`bne bucle`

- fffffa_{16} a -6_{10}
- $-6 \cdot 4 = -24$
- $pc = 54_{16}$ o $pc = 84_{10}$
- $pc = pc - 24$ reemplazando $pc = (84 + 8) - 24 = 68$ o 44_{16}