Técnicas Digitales II

Instrucciones de Procesamiento de Datos

Dr.Ing.Steiner Guillermo

Nivel de Lenguaje

- Los lenguajes denominados de alto nivel, son aquellos que poseen una alto nivel de abstracción del hardware.
- En cambio los de bajo nivel son mas cercanos a la manera en que las instrucciones se ejecutan en código máquina y dependen fuertemente del hardware.

Alto Nivel	PHP, Pascal, Python, Java
Nivel Medio	C, C++
Bajo Nivel	Ensamblador

Instrucciones de Procesamiento de Datos

- Realizan cálculos en los registros de propósitos generales.
- Puede dividirse en los siguientes grupos:
  - Instrucciones Aritméticas (suma y resta) y Lógicas.
  - Instrucciones de Comparación.
  - Instrucciones de Corrimiento.
  - Instrucciones de Multiplicación y División.

Instrucciones de Procesamiento de Datos

- Todas las instrucciones de procesamiento de Datos a excepción de multiplicación y división.

Instrucciones Aritméticas / Lógicas

- Realizan operaciones aritméticas y lógicas entre uno o dos operadores y escriben el resultado en un registro destino.
- Se puede controlar si se modifican las banderas o no mediante el sufijo **S** pudiendo o no actualizar las banderas.

La letra S habilita a la instrucción a cambiar las banderas

Instrucciones Lógicas

- En ARM, las instrucciones que permiten realizar una operación lógica son las siguientes.

AND	AND lógica	**Rd = Rn and Op2**
EOR	Or Exclusiva	**Rd = Rn xor Op2**
ORR	Or Inclusiva	**Rd = Rn or Op2**
BIC	Borrar Bit	**Rd = Rn and not Op2**
MVN	Mover negado	**Rd = not(Op2)**

Instrucciones Lógicas

	R1	0100 0110	1010 0001	1111 0001	1011 0111
	R2	1111 1111	1111 1111	0000 0000	0000 0000
		Resultados			
AND R3, R1, R2	R3	0100 0110	1010 0001	0000 0000	0000 0000
ORR R3, R1, R2	R3	1111 1111	1111 1111	1111 0001	1011 0111
EOR R3, R1, R2	R3	1011 1001	0101 1110	1111 0001	1011 0111
BIC R3, R1, R2	R3	0000 0000	0000 0000	1111 0001	1011 0111
MVN R3, R1	R3	1011 1001	0101 1110	0000 1110	0100 1000
BIC R3, #0xFF00	R3	0100 0110	1010 0001	0000 0000	1011 0111

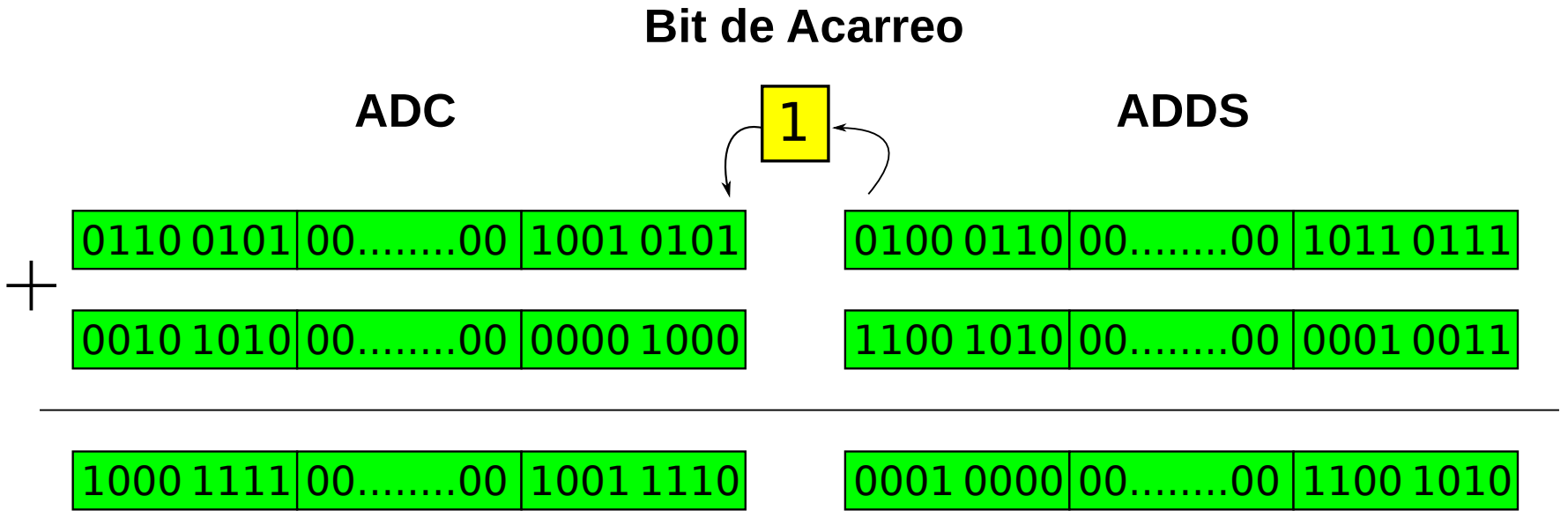
Instrucciones Aritméticas de Suma y Resta

- En ARM, las instrucciones aritméticas incluye la operaciones que realizan la suma, la resta y la resta inversa, todas con y sin carry.

SUB	Resta	**Rd = Rn - Op2**
RSB	Resta reversa	**Rd = Op2 - Rn**
ADD	Suma	**Rd = Rn + Op2**
ADC	Suma con Acarreo	**Rd = Rn + Op2 + C**
SBC	Resta con Acarreo	**Rd = Rn - Op2 + C - 1**
RSC	Resta rev.con Ac.	**Rd = Op2 – Rn + C - 1**

Instrucciones Aritméticas de Suma y Resta

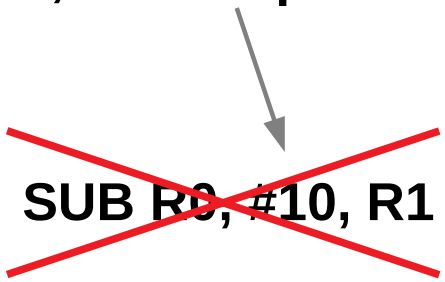
- El acarreo indica que una operación a generado un bit de acarreo en el bit mas significativo del resultado.
- Las instrucción ADC, SBC y RSC hacen uso de este bit para sumárselo a el resultado



Operaciones de Aritméticas de Suma y Resta

- Las instrucciones de resta inversa, son utilizadas para evitar la restricción del barrer shifter aplicado al operador 2

El primer operador, es siempre un registro



R0 = 10 – R1 → ~~**SUB R0, #10, R1**~~

R0 = 10 – R1 → **RSB R0, R1, #10**

Permite aplicar el barrer shifter o un valor constante a cualquiera de los operadores

Instrucciones de Comparación

- Permite realizar ciertas operaciones aritméticas y lógicas SIN guardar el resultado en un recibo.
- Su única función es cambiar las banderas, se prescinde de la letra S.
- Por no tener resultado, solo usa los dos operadores fuentes.
- Modelo de instrucción ***instrucción*** **<Rn>,<Op2>**

TST	Test	**Band(Rn and Op2)**
TEQ	Test Equivalente	**Band(Rn xor Op2)**
CMP	Comparar	**Band(Rn – Op2)**
CMN	Comparar Negativo	**Band(Rn + Op2)**

Instrucciones de Comparación

- **CMP** – Comparar, realiza la resta de los dos operadores, es la instrucción de comparación mas usada, permitiendo determinar si los operadores son iguales o si uno es mayor o menor que otro.
- **CMN** – Comparar negado, realiza la suma de los dos operadores, similar a la anterior, reemplazando el propio compilador una instrucción CMP por una CMN cuando sea conveniente

  CMP R1,#-20 → **CMN** R1,#20

Instrucciones de Comparación

- **TST** – Test, realiza un AND lógico, en general el segundo operador actúa como máscara, permitiendo determinar si un grupo de bit son cero o si un bit en particular es 0 o 1.
- **TEQ** – Test Equivalente, realiza un XOR lógico, permite saber si dos valores son iguales sin afectar la bandera de V o C

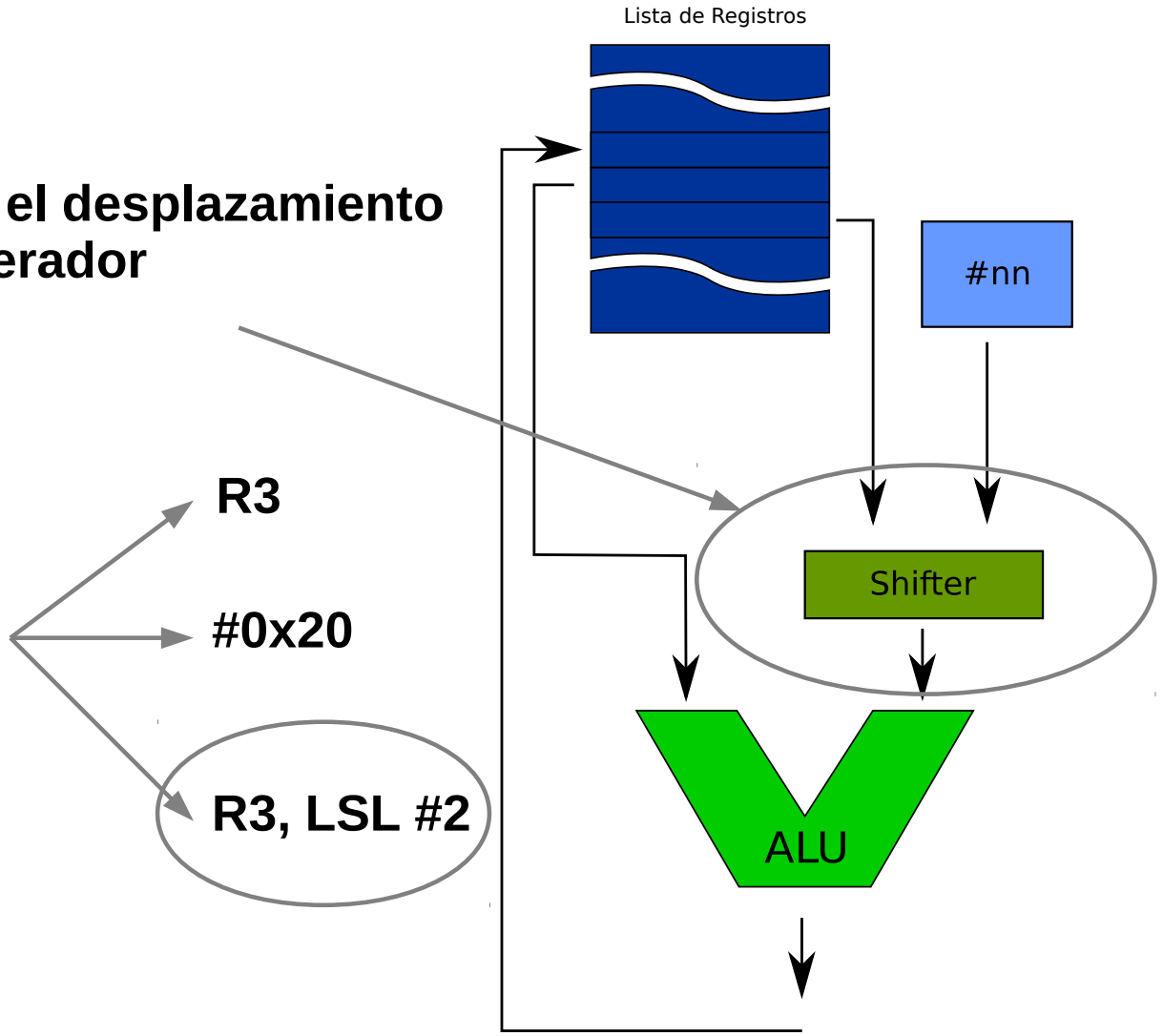
  TST R1,#0x80 → Z = 1 si bit7=0 , Z = 0 si bit7=1

Instrucciones de Corrimiento

Modelo de ALU en ARM

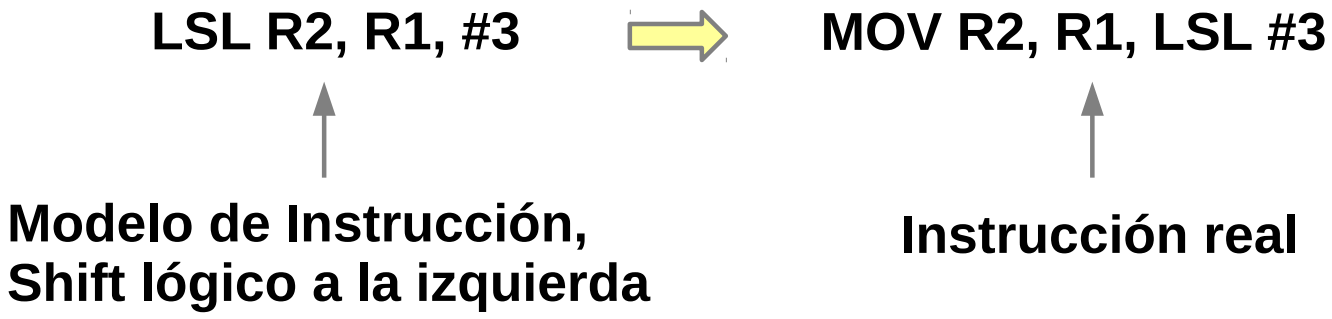
Permite realizar el desplazamiento del segundo operador

MOV Rd, Op2



Instrucciones de Corrimiento

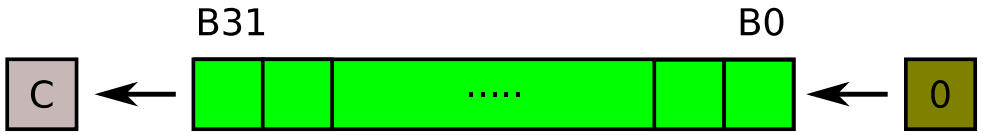
- Son 5 opciones de corrimiento entre las que se encuentran: shift, shift aritmético y rotación con y sin acarreo.
- En ARM, no existen como instrucciones, sino como modificadores de las instrucciones aritméticas, lógicas y de comparación.



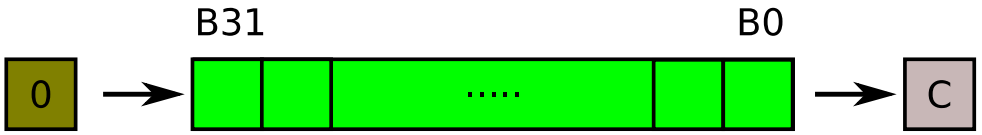
Instrucciones de Corrimiento

Corrimientos disponibles por el Barrel Shifter

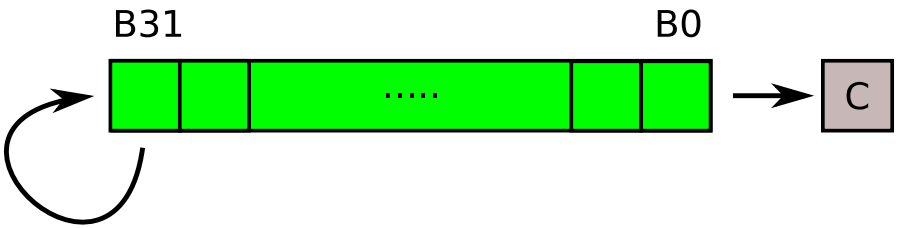
LSL - Corrimiento Lógico hacia la izquierda



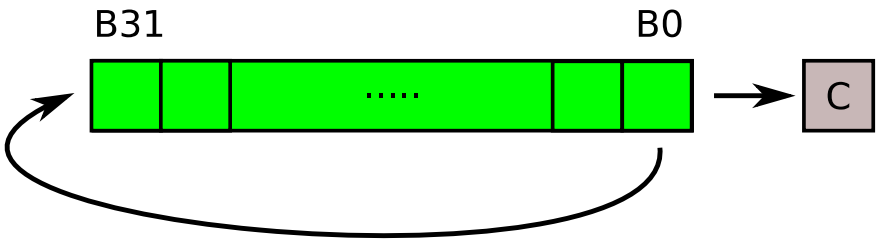
LSR - Corrimiento Lógico hacia la derecha



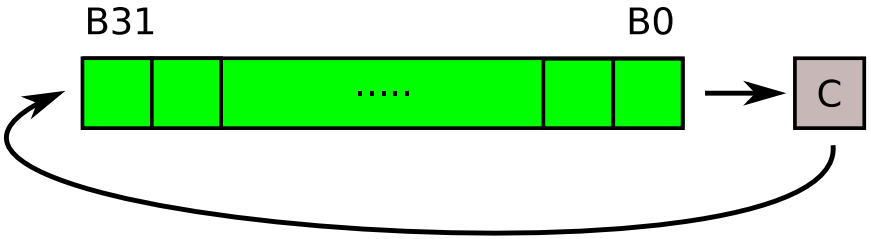
ASR - Corrimiento Aritmético hacia la derecha



ROR - Rotación hacia la derecha



RRX - Rotación hacia la derecha extendido



Instrucciones de Corrimiento

Algunos Ejemplos

LSL R4, R6, #4	R4 = R6 << 4
LSL R4, R6, R3	R4 = R6 << el valor especificado en R3
ROR R4, R6, #12	R4 = R6 rotado 12 bit a la derecha R4 = R6 rotado 20 bit a la izquierda
ASR R4, R6, #5	R4 = R6 >> 5 conservando el signo
LSLS R4, R6, #4	R4 = R6 << 4, quedando el último bit extraído en el bit C

Instrucciones de Multiplicación

- Instrucción costosa en consumo de área y potencia.
- Se puede utilizar métodos menos costoso, utilizando una rutina basada en desplazamientos y sumas (micros antiguos).
- En diseños modernos se utilizan matrices de multiplicadores, permitiendo multiplicaciones en 1 o 2 ciclos.

Intel 808x	**multiplicación de 16 bits**	**118 – 113 ciclos**
ARM	**multiplicación de 32 bits**	**1 – 2 ciclos**

Instrucciones de Multiplicación

MUL	Multiplica 32x32, resultado de 32 bits
MLA	Multiplica 32x32, el resultado se suma al valor de registro acumulador
SMULL	Multiplica 32x32 con signo, el resultado es de 64 bits
UMULL	Multiplica 32x32 sin signo, el resultado es de 64 bits
SMLAL	Multiplica 32x32 con signo, el resultado es de 64 bits y se suma al registro acumulador
UMLAL	Multiplica 32x32 sin signo, el resultado es de 64 bits y se suma al registro acumulador

Instrucciones de Multiplicación

MUL R4, R2, R1 ; R4 = R2 x R1

MULS R4, R2, R1 ; R4 = R2 x R1 actualiza N y Z

MLA R4, R2, R1, R7 ; R4 = R7 + R2 x R1

SMULL R4, R3, R1, R2 ; R3:R4 = R2 x R1 (con signo)

UMULL R4, R3, R1, R2 ; R3:R4 = R2 x R1 (sin signo)

UMLAL R4, R3, R1, R2 ; R3:R4 = R3:R4 + R2 x R1 (sin signo)

Instrucciones de División

- No aparecen en la arquitectura ARM hasta los Cortex-M4, principalmente por:
  - Requieren alto consumo de área y potencia.
  - No son muy utilizadas (se reemplazan por subrutinas).
- El aumento en la densidad de integración permitió su inclusión a partir de Cortex-M4

SDIV	División sin Signo	2 a 12 ciclos
UDIV	División con Signo	2 a 12 ciclos

- Bibliografía

  Harris & Harris. Digital design and computer architecture: ARM edition. Elsevier, 2015. Capítulo 6.

  William Hohl & Christopher Hinds. ARM assembly language. Fundamentals and techniques. 2nd edition. CRC press, 2015. Capítulo 7.

¿ Preguntas ?