FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

Arquitectura de Microprocesadores ISA Thumb 2

Ing. Facundo Larosa

Carrera de Especialización en Sistemas Embebidos

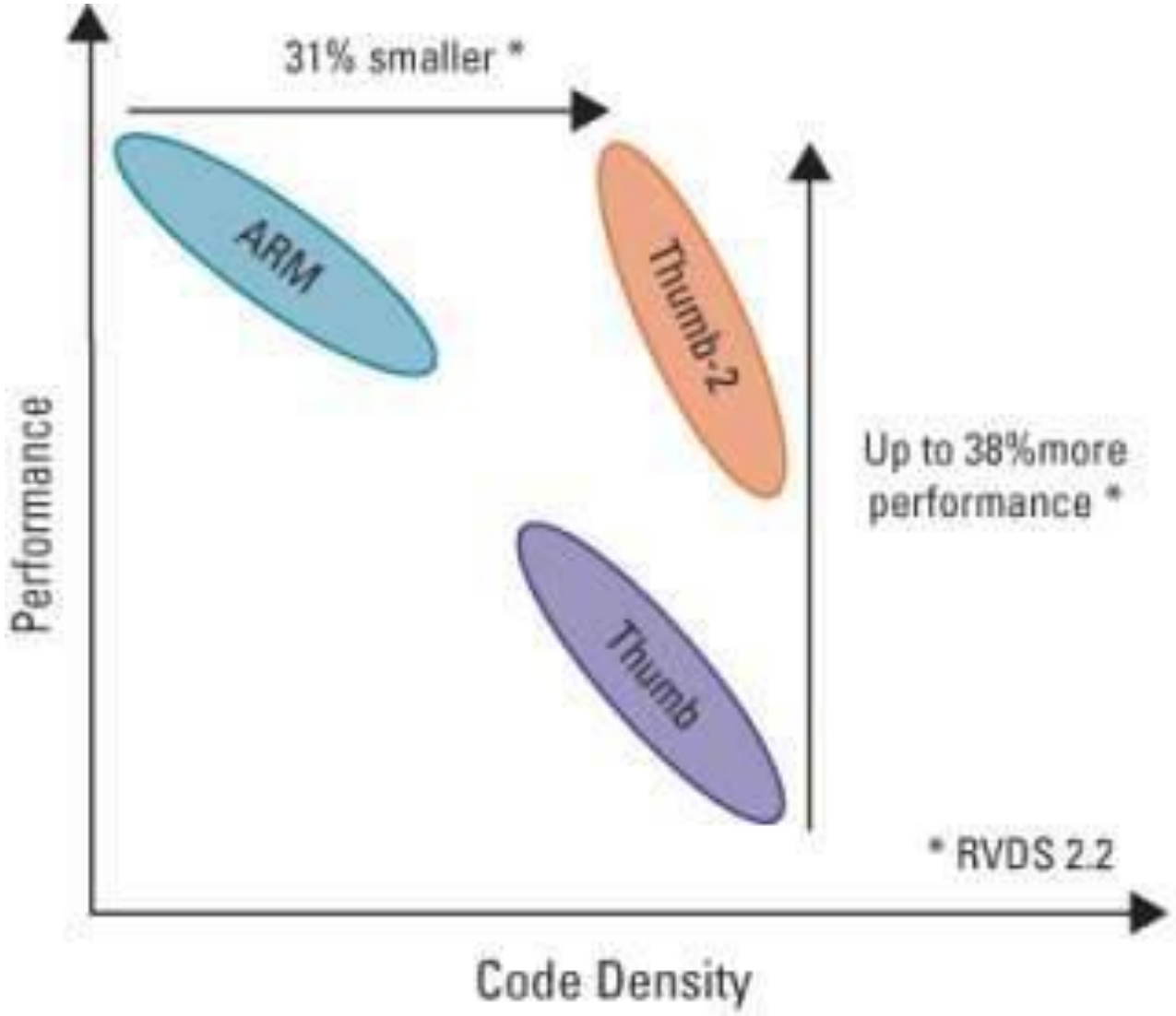
Alcance

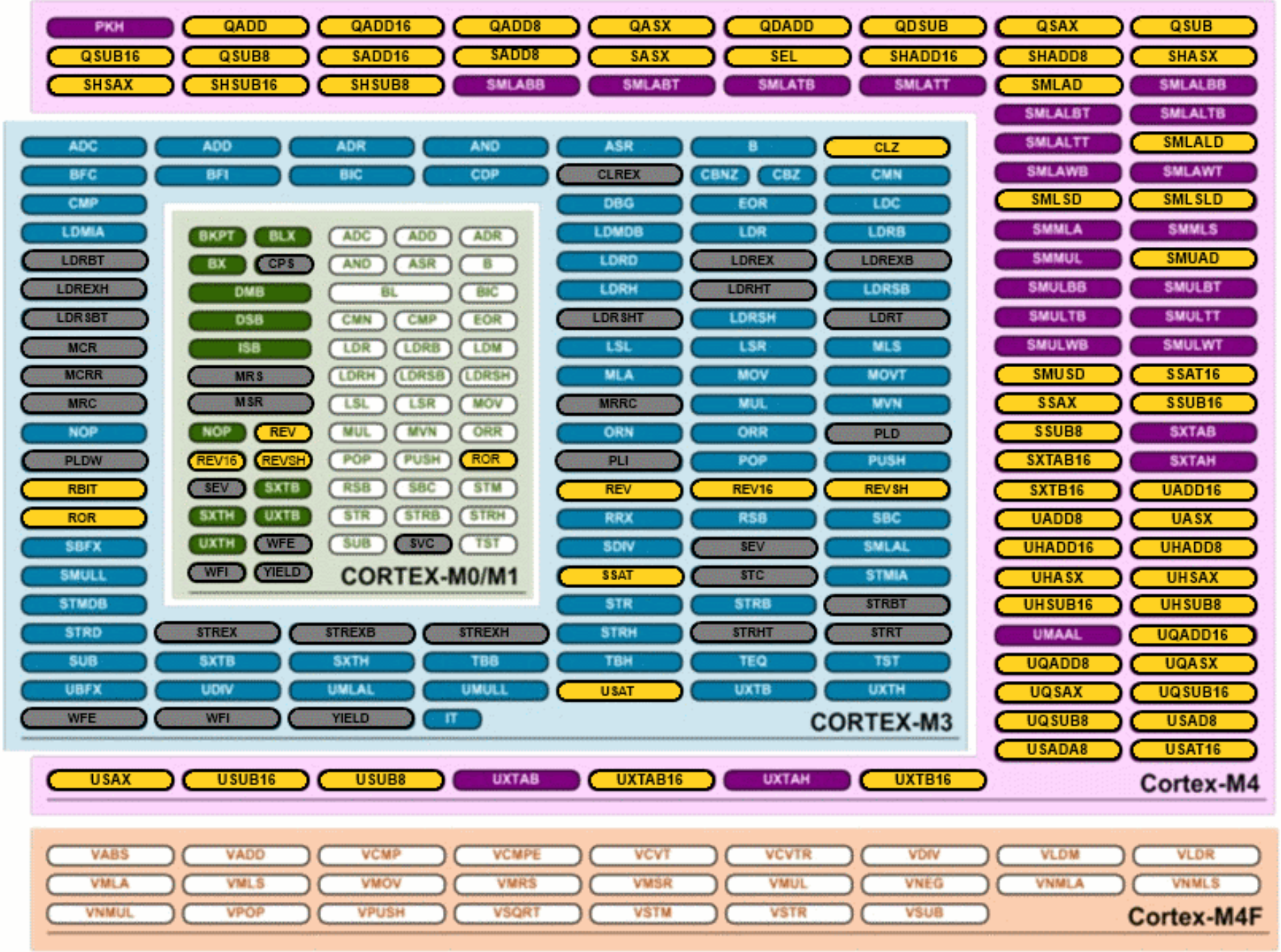
- Comprender las características generales del set de instrucciones Thumb 2 (Cortex M3/M4/M7)

- Comprender la ABI del Cortex M (Procedure Call Standard for the ARM Architecture)

- Ser capaces de aprovechar las características del lenguaje para optimizar los tiempos de ejecución, especialmente con las instrucciones SIMD (Cortex M4/M7), aritmética saturada, etc.

Thumb / Thumb 2 / ARM




PKH
QADD
QADD16
QADD8
QASX
QDADD
QDSUB
QSAX
QSUB
QSUB16
QSUB8
SADD16
SADD8
SASX
SEL
SHADD16
SHADD8
SHASX
SHSAX
SHSUB16
SHSUB8
SMLABB
SMLABT
SMLATB
SMLATT
SMLAD
SMLALBB
SMLALBT
SMLALTB
SMLALTT
SMLALD
SMLAWB
SMLAWT
SMLSD
SMLSLD
SMMLA
SMMLS
SMMUL
SMUAD
SMULBB
SMULBT
SMULTB
SMULTT
SMULWB
SMULWT
SMUSD
SSAT16
SSAX
SSUB16
SSUB8
SXTAB
SXTAB16
SXTAH
SXTB16
UADD16
UADD8
UASX
UHADD16
UHADD8
UHASX
UHSAX
UHSUB16
UHSUB8
UMAAL
UQADD16
UQADD8
UQASX
UQSAX
UQSUB16
UQSUB8
USAD8
USADA8
USAT16
Cortex-M4
ADC
ADD
ADR
AND
ASR
B
CLZ
BFC
BFI
BIC
CDP
CLREX
CBNZ
CBZ
CMN
CMP
DBG
EOR
LDC
LDMIA
LDMDB
LDR
LDRB
LDRBT
LDRD
LDREX
LDREXB
LDREXH
LDRH
LDRHT
LDRSB
LDRSBT
LDRSHT
LDRSH
LDRT
MCR
LSL
LSR
MLS
MCRR
MLA
MOV
MOVT
MRC
MRRC
MUL
MVN
NOP
ORN
ORR
PLD
PLDW
PLI
POP
PUSH
RBIT
REV
REV16
REVSH
ROR
RRX
RSB
SBC
SBFX
SDIV
SEV
SMLAL
SMULL
SSAT
STC
STMIA
STMDB
STR
STRB
STRBT
STRD
STREX
STREXB
STREXH
STRH
STRHT
STRT
SUB
SXTB
SXTH
TBB
TBH
TEQ
TST
UBFX
UDIV
UMLAL
UMULL
USAT
UXTB
UXTH
WFE
WFI
YIELD
IT
CORTEX-M3
BKPT
BLX
ADC
ADD
ADR
BX
CPS
AND
ASR
B
DMB
BL
BIC
DSB
CMN
CMP
EOR
ISB
LDR
LDRB
LDM
MRS
LDRH
LDRSB
LDRSH
MSR
LSL
LSR
MOV
NOP
REV
MUL
MVN
ORR
REV16
REVSH
POP
PUSH
ROR
SEV
SXTB
RSB
SBC
STM
SXTH
UXTB
STR
STRB
STRH
UXTH
WFE
SVC
SUB
TST
WFI
YIELD
CORTEX-M0/M1
USAX
USUB16
USUB8
UXTAB
UXTAB16
UXTAH
UXTB16
VABS
VADD
VCMP
VCMPE
VCVT
VCVTR
VDIV
VLDM
VLDR
VMLA
VMLS
VMOV
VMRS
VMSR
VMUL
VNEG
VNMLA
VNMLS
VNMUL
VPOP
VPUSH
VSQRT
VSTM
VSTR
VSUB
Cortex-M4F

Ayuda online

Cortex M4 Instruction Set

Para agendar:

http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0553a/CIHJJEIH.html

Tipos de instrucciones

- *Memory access instructions*
- *General data processing instructions*
- *Multiply and divide instructions*
- *Saturating instructions*
- *Packing and unpacking instructions*
- *Bitfield instructions*
- *Branch and control instructions*
- *Miscellaneous instructions*
- *Floating-point instructions*

Memory Access Instructions (extracto)

- **`ldr`** (load register) : Carga un registro con un valor
- **`str`** (store register): Guarda el valor de un registro en memoria
- **`ldm`** (load multiple) : Guarda valores múltiples
- **`stm`** (store multiple) : Versión múltiple de str
- **`pop`** : Obtiene uno/varios valor/es del stack
- **`push`**: Guarda uno/varios valor/es en el stack

Memory Access Instructions: Sufijos tamaño

- **`ldr`** (load register) : Carga un registro con un valor
- **`str`** (store register): Guarda el valor de un registro en memoria

Sufijo	Tamaño	Bits	Instrucción	Efecto
b	byte (unsigned)	8	`ldrb`	Completa bits superiores con 0
sb	byte (signed)	8	`ldrsb`	Completa bits superiores con el bit de signo
h	halfword (unsigned)	16	`ldrh`	Completa bits superiores con 0
sh	halfword (signed)	16	`ldrsh`	Completa bits superiores con el bit de signo
	word	32	`ldr`	

Memory Access Instructions: Ejemplos

```
ldr r0,[r1] // r0 = *r1

ldr r0,[r1,64] // r0 = *(r1+64)

ldr r0,[r1, 4]! // r0 = *(r1+4); r1+=4

str r0,[r1],-4 // *r1 = r0; r1-=4

str r5,[r1,r2] // *(r1+r2) = r5

ldr r5,[r1,r2,LSL 2] // r5 = *(r1 + (r2 << 2))

ldm r3!,{r0-r2}

// r0=*r3; r1=*(r3+4); r2=*(r3+8); r3+=12
```

Data processing: Ejemplos

```
add r0, 1 // r0 = r0+1

adds r0, 1 // r0 = r0+1 ¡y actualiza las flags de estado!

add r0,r1,r2 // r0 = r1 + r2

sub r0,r1,r2 // r0 = r1 - r2

and r9,r1,0xFF00 // r9 = r1 & 0xFF00

orr r1,r1,0x0001 // r1 = r1 | 0x0001

lsr r1,r0,4 // r1= r0>>4
```

Ejecución condicional : Sufijos

Suffix	Flags	Meaning
EQ	Z = 1	Equal
NE	Z = 0	Not equal
CS or HS	C = 1	Higher or same, unsigned
CC or LO	C = 0	Lower, unsigned
MI	N = 1	Negative
PL	N = 0	Positive or zero
VS	V = 1	Overflow
VC	V = 0	No overflow
HI	C = 1 and Z = 0	Higher, unsigned
LS	C = 0 or Z = 1	Lower or same, unsigned
GE	N = V	Greater than or equal, signed
LT	N != V	Less than, signed
GT	Z = 0 and N = V	Greater than, signed
LE	Z = 1 and N != V	Less than or equal, signed
AL	Can have any value	Always. This is the default when no suffix is specified.

Ejecución condicional: Instrucción IT

```
ITTE    NE
ANDNE   r0,r0,r1
ADDNE  r2,r2,#1
MOVEQ   r2,r3

//Se pueden colocar hasta cuatro sentencias condicionales
```

Fuente: www.infocenter.arm.com

Branch and control: Ejemplos

```
b Etiqueta //Salto incondicional a "Etiqueta"
bl Funcion //Salto a "Funcion" con guarda del LR
```

Ejemplos de salto:

```
cmp r0,5
beq SiEsCinco
NoEsCinco: ....
SiEsCinco: ....
```

```
cbz r0,SiEsCero
NoEsCero:
....
SiEsCero: ....
```

```
cbnz r0,NoEsCero
SiEsCero:
....
NoEsCero: ....
```

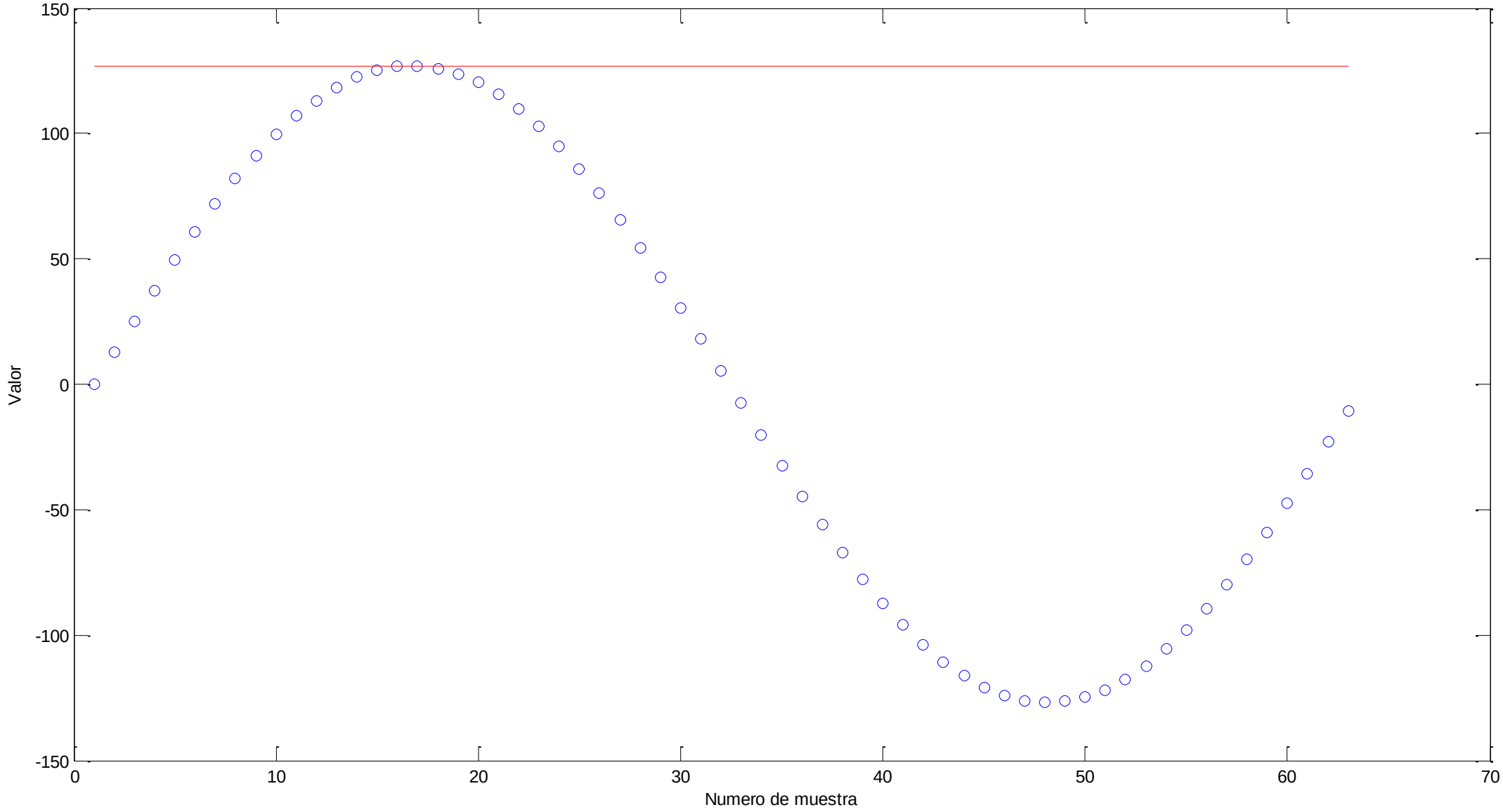
Reversal: Ejemplos

```
//Reversión de bytes
//r0=0x12345678
rev r1,r0 //r1=0x78563421
revh r2,r0 //r2=0x34127856

//Reversion de bits
//r0=1111 1110 1100 1000 0000 0001 0011 0111
rbit r1,r0
//r1=1110 1100 1000 0000 0001 0011 0111 1111
```

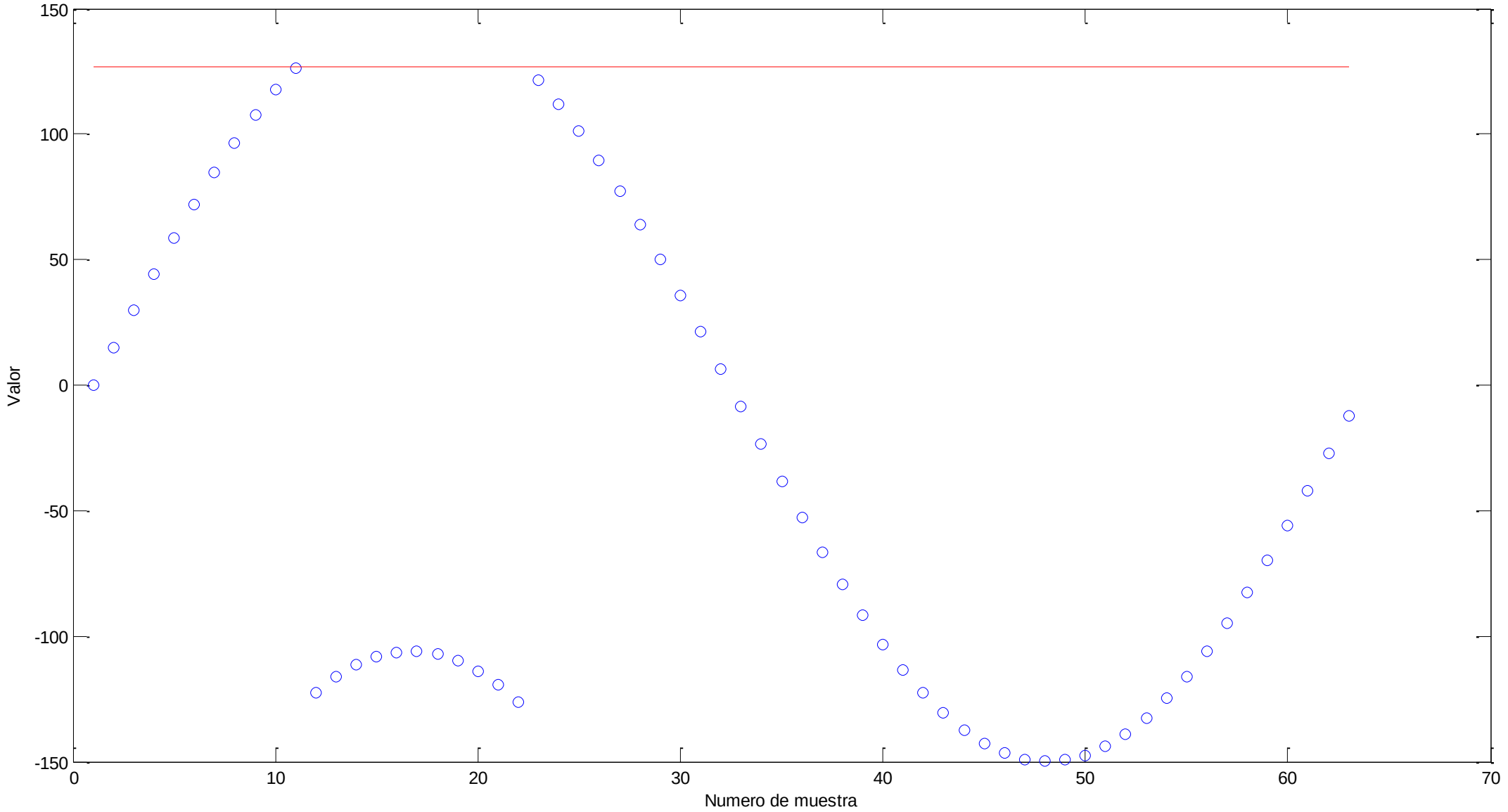
Campos de bits: Ejemplos

```
//Borrado de campos de bits (bit field clear)
ldr r1,=0x12345678 //r1=0x12345678
bfc r1,8,12 //r1=0x12300078

//Inserción de campos de bits (bit field insertion)
ldr r0,=0x12345678
ldr r1,=0xAAAAAAAA
bfi r1,r0,12,8 //r1=0xAAA78AAA
```

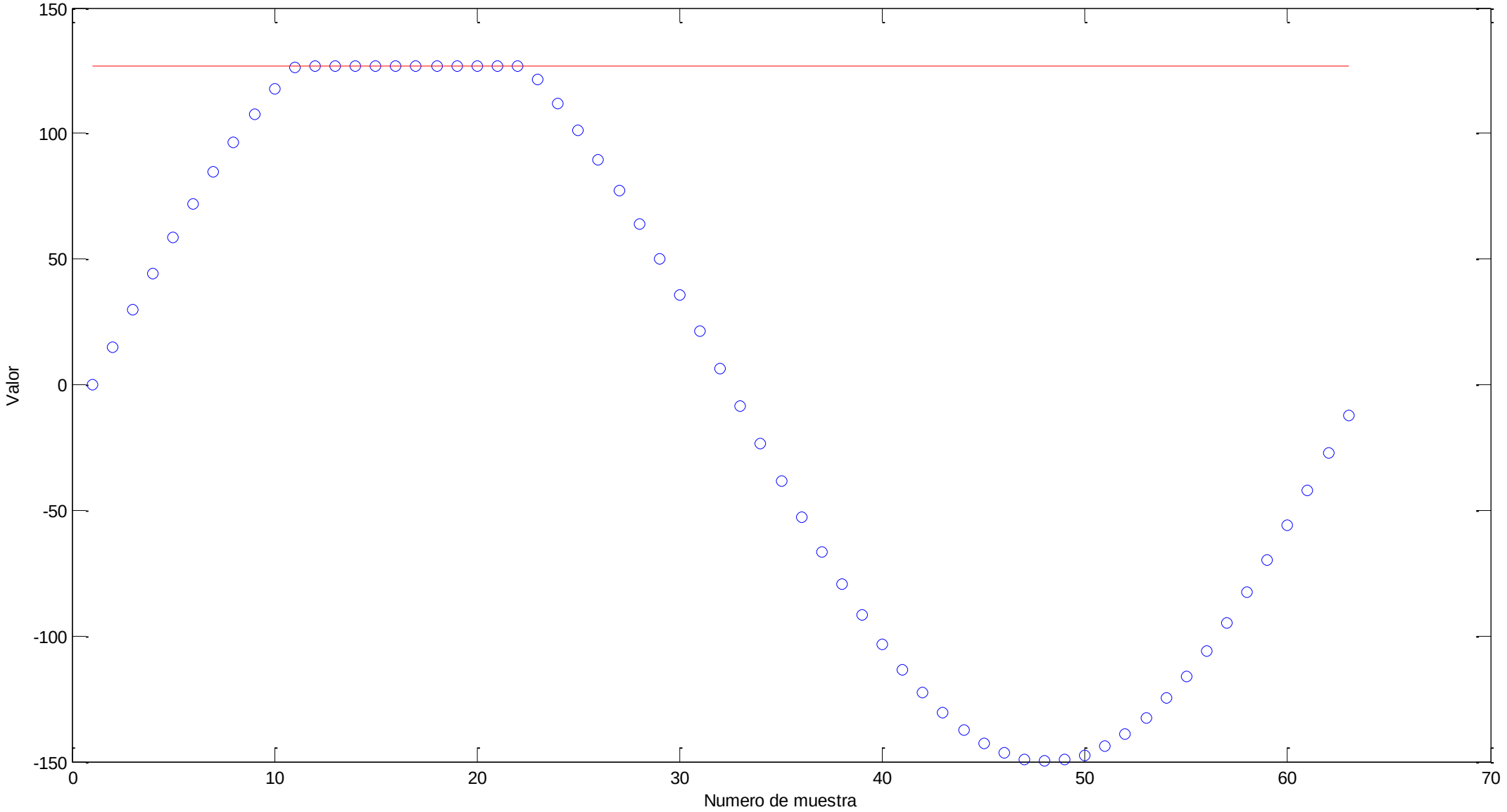
Aritmética saturada



Señal con overflow (sin aritmética saturada)



Señal con overflow (con aritmética saturada)



Instrucciones de aritmética saturada

```
//Ejemplo: saturación a 16 bits
```

```
ssat r1,16,r0
```

Table 5.38 Examples of SSAT Results

Input (R0)	Output (R1)	Q Bit
0x00020000	0x00007FFF	Set
0x00008000	0x00007FFF	Set
0x00007FFF	0x00007FFF	Unchanged
0x00000000	0x00000000	Unchanged
0xFFFF8000	0xFFFF8000	Unchanged
0xFFFF7FFF	0xFFFF8000	Set
0xFFFE0000	0xFFFF8000	Set

```
usat r1,16,r0
```

Table 5.39 Examples of USAT Results

Input (R0)	Output (R1)	Q Bit
0x00020000	0x0000FFFF	Set
0x00008000	0x00008000	Unchanged
0x00007FFF	0x00007FFF	Unchanged
0x00000000	0x00000000	Unchanged
0xFFFF8000	0x00000000	Set
0xFFFF8001	0x00000000	Set
0xFFFFFFFF	0x00000000	Set

Instrucciones de punto flotante

Operation	Description	Assembler	Cycles
Absolute value	of float	VABS.F32	1
Addition	floating point	VADD.F32	1
Compare	float with register or zero	VCMP.F32	1
	float with register or zero	VCMPE.F32	1
Convert	between integer, fixed-point, half-precision and float	VCVT.F32	1
Divide	Floating-point	VDIV.F32	14
Load	multiple doubles	VLDM.64	1+2*N, where N is the number of doubles.
	multiple floats	VLDM.32	1+N, where N is the number of floats.
	single double	VLDR.64	3
	single float	VLDR.32	2

Instrucciones de punto flotante III

Pop	double registers from stack	VPOP.64	1+2*N, where N is the number of double registers.
	float registers from stack	VPOP.32	1+N where N is the number of registers
Push	double registers to stack	VPUSH.64	1+2*N, where N is the number of double registers.
	float registers to stack	VPUSH.32	1+N, where N is the number of registers
Square-root	of float	VSQRT.F32	14
Store	multiple double registers	VSTM.64	1+2*N, where N is the number of doubles.
	multiple float registers	VSTM.32	1+N, where N is the number of floats.
	single double register	VSTR.64	3
	single float registers	VSTR.32	2
Subtract	float	VSUB.F32	1

Instrucciones de punto flotante II

Move	top/bottom half of double to/from core register	VMOV	1
	immediate/float to float-register	VMOV	1
	two floats/one double to/from two core registers or one float to/from one core register	VMOV	2
	floating-point control/status to core register	VMRS	1
	core register to floating-point control/status	VMSR	1
Multiply	float	VMUL.F32	1
	then accumulate float	VMLA.F32	3
	then subtract float	VMLS.F32	3
	then accumulate then negate float	VNMLA.F32	3
	then subtract then negate float	VNMLS.F32	3
Multiply (fused)	then accumulate float	VFMA.F32	3
	then subtract float	VFMS.F32	3
	then accumulate then negate float	VFNMA.F32	3
	then subtract then negate float	VFNMS.F32	3
Negate	float	VNEG.F32	1
	and multiply float	VNMUL.F32	1

Interfaz entre assembler y C

La interfaz entre assembler y C en el Cortex está definida en el documento "ARM Architecture Procedure Call Standard" (conocido también como AAPCS).

En resumidas cuentas:

- r0 , r1 , r2 y r3 son registros que se utilizan para pasar parámetros

```
funcion (arg0,arg1,arg2,arg3)
```

Argumento	Registro
arg0	r0
arg1	r1
arg2	r2
arg3	r3

- r0 se utiliza para devolver el resultado de la función

Bibliografía

[1] YIU, Joseph , “The Definitive Guide to ARM Cortex M3 and M4 processors”
[2] YIU, Joseph , “The Definitive Guide to ARM Cortex M0 and M0+ processors”
[3]* “ARMv7-M Architecture Reference Manual”, ARM Inc.
[4]* “ARMv7-AR Architecture Reference Manual”, ARM Inc.
[5]* “ARM Cortex R Architecture”, White Paper, ARM, Inc.
[6] www.infocenter.arm.com
[7] www.keil.com

*Disponibles online en: *www.arm.com*