F

JUEGO DE INSTRUCCIONES DEL MICROPROCESADOR Z-80

F.1.- Instrucciones de transferencia de 8 bits: LD d, f Efecto: f → d

d↓ f→	A	B	C	D	E	H	L	(HL)	(IX+ds)	(IY+ds)	in	(BC)	(DE)	(dirc)
A	7F	78	79	7A	7B	7C	7D	7E	DD7Eds	FD7Eds	3Ein	0A	1A	3Adirc
B	47	40	41	42	43	44	45	46	DD46ds	FD46ds	06in			
C	4F	48	49	4A	4B	4C	4D	4E	DD4Eds	FD4Eds	0Ein			
D	57	50	51	52	53	54	55	56	DD56ds	FD56ds	16in			
E	5F	58	59	5A	5B	5C	5D	5E	DD5Eds	FD5Eds	1Ein			
H	67	60	61	62	63	64	65	66	DD66ds	FD66ds	26in			
L	6F	68	69	6A	6B	6C	6D	6E	DD6Eds	FD6Eds	2Ein			
(HL)	77	70	71	72	73	74	75				36in			
(BC)	02													
(DE)	12													
(dirc)	32dirc													

d↓ f→	A	B	C	D	E	H	L	in
(IX+ds)	DD77ds	DD70ds	DD71ds	DD72ds	DD73ds	DD74ds	DD75ds	DD36dsin
(IY+ds)	FD77ds	FD70ds	FD71ds	FD72ds	FD73ds	FD74ds	FD75ds	FD36dsin

Nemónico	C.op.	Cambios en flags					
		S	Z	H	P/V	N	C
LD A,I	ED57	*	*	0	*	0	-
LD A,R	ED5F	*	*	0	*	0	-
LD I,A	ED47	-	-	-	-	-	-
LD R,A	ED4F	-	-	-	-	-	-

F.2.- Instrucciones de transferencia de 16 bits: LD d,f Efecto: f → d

Op. ↓	d					
	BC	DE	HL	SP	IX	IY
LD d,inmd	01inmd	11inmd	21inmd	31inmd	DD21inmd	FD21inmd
LD d,(dirc)	ED4Bdirc	ED5Bdirc	2Adirc	ED7Bdirc	DD2Adirc	FD2Adirc

Op. ↓	f					
	BC	DE	HL	SP	IX	IY
LD (dirc),f	ED43dirc	ED53dirc	22dirc	ED73dirc	DD22dirc	FD22dirc
LD SP,f			F9		DDF9	FDF9

F.3.- Instrucciones de transferencia a la pila

Op. ↓ f o d→	BC	DE	HL	AF	IX	IY
PUSH f	C5	D5	E5	F5	DDE5	FDE5
POP d	C1	D1	E1	F1	DDE1	FDE1

F.4.- Instrucciones de intercambio: EX d1,d2 Efecto: d1 ↔ d2

EX DE,HL	EB
EX AF,A'F'	08
EXX (1)	D9

	d2		
	HL	IX	IY
EX (SP),d2	E3	DDE3	FDE3

Nota: (1) La instrucción EXX intercambia los registros B, C, D, E, H y L con sus correspondientes en el banco alternativo: B', C', D', E', H' y L'.

F.5.- Instrucciones de E/S

Op. ↓	d/f							Cambios en *flags*					
	A	B	C	D	E	H	L	S	Z	H	P/V	N	C
IN d,(C)	ED78	ED40	ED48	ED50	ED58	ED60	ED68	*	*	0	*	0	-
OUT (C),f	ED79	ED41	ED49	ED51	ED59	ED61	ED69	-	-	-	-	-	-
IN d,(pt)	DB							-	-	-	-	-	-
OUT (pt),f	D3							-	-	-	-	-	-

F.6.- Instrucciones aritmético-lógicas de 8 bits Op f Efecto: f Op A → A

para INC y DEC: f±1→f

Op. ↓	f											Cambios en *flags*					
	A	B	C	D	E	H	L	(HL)	in	(IX+ds)	(IY+ds)	S	Z	H	P/V	H	C
ADD	87	80	81	82	83	84	85	86	C6in	DD86ds	FD86ds	*	*	*	*	0	*
ADC	8F	88	89	8A	8B	8C	8D	8E	CEin	DD8Eds	FD8Eds	*	*	*	*	0	*
SUB	97	90	91	92	93	94	95	96	D6in	DD96ds	FD96ds	*	*	*	*	1	*
SBC	9F	98	99	9A	9B	9C	9D	9E	DEin	DD9Eds	FD9Eds	*	*	*	*	1	*
AND	A7	A0	A1	A2	A3	A4	A5	A6	E6in	DDA6ds	FDA6ds	*	*	1	*	0	0
XOR	AF	A8	A9	AA	AB	AC	AD	AE	EEin	DDAEds	FDAEds	*	*	1	*	0	0
OR	B7	B0	B1	B2	B3	B4	B5	B6	F6in	DDB6ds	FDB6ds	*	*	1	*	0	0
CP	BF	B8	B9	BA	BB	BC	BD	BE	FEin	DDBEds	FDBEds	*	*	*	*	1	*
INC	3C	04	0C	14	1C	24	2C	34		DD34ds	FD34ds	*	*	*	*	0	-
DEC	3D	05	0D	15	1D	25	2D	35		DD35ds	FD35ds	*	*	*	*	1	-

Op. ↓	C.op.	Descripción	Cambios en *flags*					
			S	Z	H	P/V	N	C
DAA	27	Ajuste decimal del acumulador	*	*	*	*	-	*
CPL	2F	Complementa a 1 el acumulador	-	-	1	-	1	-
NEG	ED44	Complementa a 2 el acumulador (Cambio de signo)	*	*	*	*	1	*

F.7.- Instrucciones aritmético-lógicas de 16 bits: Op d,f Efecto: f Op d → d;
para INC y DEC: Op f Efecto: f±1 → f

Op. ↓	f						Cambios en los *flags*					
	BC	DE	HL	SP	IX	IY	S	Z	H	P/V	N	C
INC f	03	13	23	33	DD23	FD23	-	-	-	-	-	-
DEC f	0B	1B	2B	3B	DD2B	FD2B	-	-	-	-	-	-
ADD HL, f	09	19	29	39			-	-	*	-	0	*
ADC HL,f	ED4A	ED5A	ED6A	ED7A			*	*	*	*	0	*
SBC HL, f	ED42	ED52	ED62	ED72			*	*	*	*	1	*
ADD IX, f	DD09	DD19		DD39	DD29		-	-	*	-	0	*
ADD IY, f	FD09	FD19		FD39		FD29	-	-	*	-	0	*

F.8.- Instrucciones de desplazamiento y rotación: Op d

Op↓ d→	A	B	C	D	E	H	L	(HL)	(IX+ds)	(IY+ds)
RR	CB1F	CB18	CB19	CB1A	CB1B	CB1C	CB1D	CB1E	DDCBds1E	FDCBds1E
RL	CB17	CB10	CB11	CB12	CB13	CB14	CB15	CB16	DDCBds16	FDCBds16
RRC	CB0F	CB08	CB09	CB0A	CB0B	CB0C	CB0D	CB0E	DDCBds0E	FDCBds0E
RLC	CB07	CB00	CB01	CB02	CB03	CB04	CB05	CB06	DDCBds06	FDCBds06
SRA	CB2F	CB28	CB29	CB2A	CB2B	CB2C	CB2D	CB2E	DDCBds2E	FDCBds2E
SLA	CB27	CB20	CB21	CB22	CB23	CB24	CB25	CB26	DDCBds26	FDCBds26
SRL	CB3F	CB38	CB39	CB3A	CB3B	CB3C	CB3D	CB3E	DDCBds3E	FDCBds3E

Descripción de las operaciones y cambios en los *flags*:

Nemónico	Descripción	Cambios en *flags*					
		S	Z	H	P/V	N	C
RR/RL	Rotación a derecha o izquierda a través del bit C	*	*	0	*	0	*
RRC/RLC	Rotación a derecha o izquierda	*	*	0	*	0	*
SRA/SLA	Desplazamiento aritmético a derecha o izquierda	*	*	0	*	0	*
SRL	Desplazamiento lógico a la derecha	*	*	0	*	0	*

Otros desplazamientos

Nemónico	C.op.	Descripción	Cambios en los *flags*					
			S	Z	H	P/V	N	C
RRCA	0F	Rotación del acumulador a la derecha	-	-	0	-	0	*
RLCA	07	Rotación del acumulador a la izquierda	-	-	0	-	0	*
RRA	1F	Rota el acumulador a la derecha sobre el bit C	-	-	0	-	0	*
RLA	17	Rota el acumulador a la izquierda sobre el bit C	-	-	0	-	0	*
RLD (HL)	ED6F	Rotación BCD de (HL) y $A_{3:0}$ a la izquierda (1)	*	*	0	*	0	-
RRD (HL)	ED67	Rotación BCD de (HL) y $A_{3:0}$ a la derecha (1)	*	*	0	*	0	-

Nota (1) Las rotaciones BCD rotan los dígitos BCD que se encuentran en $A_{3:0}$, $(HL)_{7:4}$ y $(HL)_{3:0}$, respectivamente, de mayor a menor orden de peso.

F.9.- Instrucciones de control de flujo incondicionales

Op. ↓	Descripción		dirc	(HL)	(IX)	(IY)
JP	Salto absoluto		C3dirc	E9	DDE9	FDE9
JR	Salto relativo		18ds			
CALL	Llamada a subrutina		CDdirc			
RET	Retorno de subrutina	C9				
RETI	Retorno de interrupción	ED4D				
RETN	Retorno de interrupción no enmascarable	ED45				

Llamada a rutina en página 0: RST dd donde dd puede ser 00 08 10 18 20 28 30 ó 38H
Efecto: el mismo que causaría CALL 00dd

dd →	00	08	10	18	20	28	30	38
RST dd	C7	CF	D7	DF	E7	EF	D7	DF

F.10.- Instrucciones de control del flujo condicionales

Op. ↓	Condición (c)							
	Z	NZ	C	NC	PE	PO	M	P
JP c,dirc	CAdirc	C2dirc	DAdirc	D2dirc	EAdirc	E2dirc	FAdirc	F2dirc
JR c,ds	28ds	20ds	38ds	30ds				
CALL c,dirc	CCdirc	C4dirc	DCdirc	D4dirc	ECdirc	E4dirc	FCdirc	F4dirc
RET c	C8	C0	D8	D0	E8	E0	F8	F0

Significado de las condiciones:

Condición	Z	NZ	C	NC	PE	PO	M	P
Significado	Cero	No cero	Llevada	No llevada	Paridad par /*Overflow*	Paridad impar/No *overflow*	Menos	Más
Valor *flag*	Z==1	Z==0	C==1	C==0	P/V==1	P/V==0	S==1	S==0

Instrucción para efectuar bucles: DJNZ ds C.op: 10ds Es equivalente a DEC B; JR NZ ds

F.11.- Instrucciones de manejo de bloques

Op. ↓	C.op.	Descripción	Cambios en los *flags*					
			S	Z	H	P/V	N	C
LDI	EDA0	LD (DE),(HL); INC HL; INC DE; DEC BC	-	-	0	*	0	-
LDIR	EDB0	Repetir LDI hasta que BC==0	-	-	0	0	0	-
LDD	EDA8	LD (DE),(HL); DEC HL; DEC DE; DEC BC	-	-	0	*	0	-
LDDR	EDB8	Repetir LDD hasta que BC==0	-	-	0	0	0	-
CPI	EDA1	CP A,(HL); INC HL; DEC BC	*	*	*	*	1	-
CPIR	EDB1	Repetir CPI hasta que BC==0 o A==(HL)	*	*	*	*	1	-
CPD	EDA9	CP A,(HL); DEC HL; DEC BC	*	*	*	*	1	-
CPDR	EDB9	Repetir CPD hasta que BC==0 o A==(HL)	*	*	*	*	1	-
INI	EDA2	IN (HL),(C); INC HL; DEC B	?	*	?	?	1	-
INIR	EDB2	Repetir INI hasta que B==0	?	1	?	?	1	-
IND	EDAA	IN (HL),(C); DEC HL; DEC B	?	*	?	?	1	-
INDR	EDBA	Repetir IND hasta que B==0	?	1	?	?	1	-
OUTI	EDA3	OUT (C),(HL); INC HL; DEC B	?	*	?	?	1	-
OTIR	EDB3	Repetir OUTI hasta que B==0	?	1	?	?	1	-
OUTD	EDAB	OUT (C),(HL); DEC HL; DEC B	?	*	?	?	1	-
OTDR	EDBB	Repetir OUTD hasta que B==0	?	1	?	?	1	-

F.12.- Instrucciones de manejo de bit: Op n, d

d→	A	B	C	D	E	H	L	(HL)	(IX+ds)	(IY+ds)
BIT 0	CB47	CB40	CB41	CB42	CB43	CB44	CB45	CB46	DDCBds46	FDCBds46
BIT 1	CB4F	CB48	CB49	CB4A	CB4B	CB4C	CB4D	CB4E	DDCBds4E	FDCBds4E
BIT 2	CB57	CB50	CB51	CB52	CB53	CB54	CB55	CB56	DDCBds53	FDCBds53
BIT 3	CB5F	CB58	CB59	CB5A	CB5B	CB5C	CB5D	CB5E	DDCBds5E	FDCBds5E
BIT 4	CB67	CB60	CB61	CB62	CB63	CB64	CB65	CB66	DDCBds66	FDCBds66
BIT 5	CB6F	CB68	CB69	CB6A	CB6B	CB6C	CB6D	CB6E	DDCBds6E	FDCBds6E
BIT 6	CB77	CB70	CB71	CB72	CB73	CB74	CB75	CB76	DDCBds76	FDCBds76
BIT 7	CB7F	CB78	CB79	CB7A	CB7B	CB7C	CB7D	CB7E	DDCBds7E	FDCBds7E
RES 0	CB87	CB80	CB81	CB82	CB83	CB84	CB85	CB86	DDCBds86	FDCBds86
RES 1	CB8F	CB88	CB89	CB8A	CB8B	CB8C	CB8D	CB8E	DDCBds8E	FDCBds8E
RES 2	CB97	CB90	CB91	CB92	CB93	CB94	CB95	CB96	DDCBds96	FDCBds96
RES 3	CB9F	CB98	CB99	CB9A	CB9B	CB9C	CB9D	CB9E	DDCBds9E	FDCBds9E
RES 4	CBA7	CBA0	CBA1	CBA2	CBA3	CBA4	CBA5	CBA6	DDCBdsA6	FDCBdsA6
RES 5	CBAF	CBA8	CBA9	CBAA	CBAB	CBAC	CBAD	CBAE	DDCBdsAE	FDCBdsAE
RES 6	CBB7	CBB0	CBB1	CBB2	CBB3	CBB4	CBB5	CBB6	DDCBdsB6	FDCBdsB6
RES 7	CBBF	CBB8	CBB9	CBBA	CBBB	CBBC	CBBD	CBBE	DDCBdsBE	FDCBdsBE
SET 0	CBC7	CBC0	CBC1	CBC2	CBC3	CBC4	CBC5	CBC6	DDCBdsC6	FDCBdsC6
SET 1	CBCF	CBC8	CBC9	CBCA	CBCB	CBCC	CBCD	CBCE	DDCBdsCE	FDCBdsCE
SET 2	CBD7	CBD0	CBD1	CBD2	CBD3	CBD4	CBD5	CBD6	DDCBdsD6	FDCBdsD6
SET 3	CBDF	CBD8	CBD9	CBDA	CBDB	CBDC	CBDD	CBDE	DDCBdsDE	FDCBdsDE
SET 4	CBE7	CBE0	CBE1	CBE2	CBE3	CBE4	CBE5	CBE6	DDCBdsE6	FDCBdsE6
SET 5	CBEF	CBE8	CBE9	CBEA	CBEB	CBEC	CBED	CBEE	DDCBdsEE	FDCBdsEE
SET 6	CBF7	CBF0	CBF1	CBF2	CBF3	CBF4	CBF5	CBF6	DDCBdsF6	FDCBdsF6
SET 7	CBFF	CBF8	CBF9	CBFA	CBFB	CBFC	CBFD	CBFE	DDCBdsFE	FDCBdsFE

Descripción de las operaciones y cambios en *flags*:

Op.	Descripción	Cambios en los *flags*					
		S	Z	H	P/V	N	C
BIT	Prueba de bit. $2^n \wedge d \rightarrow TMP$. Como consecuencia, $Z \leftarrow \overline{d}_n$	?	*	1	?	0	-
RES	Borrado de bit. $2^n \wedge d \rightarrow d$. Como consecuencia $d_n \leftarrow 0$	-	-	-	-	-	-
SET	Activación de bit. $2^n \vee d \rightarrow d$. Como consecuencia $d_n \leftarrow 1$	-	-	-	-	-	-

F.13.- Instrucciones diversas

Operación	C.op.	Descripción	Cambios en los *flags*					
			S	Z	H	P/V	N	C
NOP	00	No operación	-	-	-	-	-	-
CCF	3F	Complementa el *flag* C	-	-	?	-	0	*
SCF	37	Pone a 1 el *flag* C (C=1)	-	-	0	-	0	1
EI	FB	Activa las interrupciones	-	-	-	-	-	-
DI	F3	Desactiva las interrupciones	-	-	-	-	-	-
IM 0	ED46	Activa el modo de interrupción 0	-	-	-	-	-	-
IM 1	ED56	Activa el modo de interrupción 1	-	-	-	-	-	-
IM 2	ED5E	Activa el modo de interrupción 2	-	-	-	-	-	-
HALT	76	Detiene el procesador	-	-	-	-	-	-

F.14.- Composición del registro de *flags* (F)

Orden del bit	7	6	5	4	3	2	1	0
Símbolo	S	Z	-	H	-	P/V	N	C
Significado	Signo	Cero	No usado	Semillevada	No usado	Paridad/*Overflow*	Suma/resta	Llevada

Claves para el manejo de estas tablas:

dirc: Dirección de memoria de 16 bits.
in: Operando inmediato de 8 bits.
inmd: operando inmediato de 16 bits.
ds: Desplazamiento de 8 bits.
pt: Número de puerto de 8 bits.

Significado de las abreviaturas de los *flags*:

* *Flag* cambia
\- *Flag* no cambia
0 0 → *flag*
1 1 → *flag*
? El valor del *flag* es impredecible

G

JUEGO DE INSTRUCCIONES DE LA ARQUITECTURA SPARC V9

Nota importante: No están incluidas en este apéndice las instrucciones de versiones anteriores que se sólo se conservan por compatibilidad, y cuyo uso está desaconsejado, ni las instrucciones privilegiadas.

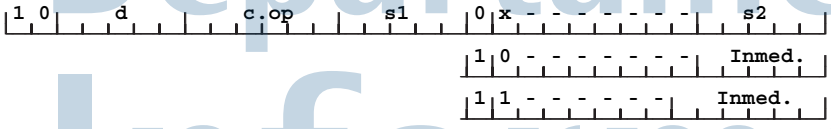
G.1.- Instrucciones de carga y almacenamiento

Nemónico	C.op.	Operación
ldstub	00 1101	*a* → *r*d; 1's → *a* (1)
ld	10 0000	*a* → *f*d (simple p.)
ldd	10 0011	*a* → *f*d (doble p.)
ldq	10 0010	*a* → *f*d (cuád. p.)
lds*n*	00 10*nn*	*a* → *r*d (2) (3)
ldu*n*	00 00*nn*	*a* → *r*d (2) (3)
ldx	00 1011	*a* → *r*d (64 bits)
st	10 0100	*f*d → *a* (simple p.)
std	10 0111	*f*d → *a* (doble p.)
stq	10 0110	*f*d → *a* (cuád. p.)
st*n*	00 01*nn*	*r*d → *a* (2)
stx	00 1110	*r*d → *a* (64 bits)

Sintaxis en ensamblador: **Nemón. x, y**
donde *x* es el operando fuente e *y* es el destino.
Uno de los operandos es el registro *r*d, mientras que el otro es un operando en memoria cuya dirección,*a* , se calcula, en función del bit 13:
si $I_{13} = 0$, *a* = *r*s1 + *r*s2
si $I_{13} = 1$, *a* = *r*s1 + desplazamiento

Notas:
(1) La instrucción ldstub carga *r*d (byte) y luego escribe 1's en la posición de memoria *a* (estas operaciones se realizan atómicamente)
(2) En estas instrucciones, el dato transferido puede ser byte (*n* = b, *nn* = 01), semipalabra (*n* = h, *nn* = 10) o palabra (*n* = w, *nn* = 00)
(3) La instrucción lds*n* extiende el signo a la parte alta de *r*d, mientras que ldu*n* rellena con ceros.

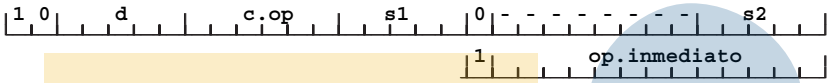
G.2.- Instrucciones de desplazamiento



Nemónico	C.op.	x	Operación
sll	10 0101	0	*r*s1<<*b* → *r*d (lóg. 32 bits)
srl	10 0110	0	*r*s1>>*b* → *r*d (lóg. 32 bits)
sra	10 0111	0	*r*s1>>*b* → *r*d (arit. 32 bits)
sllx	10 0101	1	*r*s1<<*b* → *r*d (lóg. 64 bits)
srlx	10 0110	1	*r*s1>>*b* → *r*d (lóg. 64 bits)
srax	10 0111	1	*r*s1>>*b* → *r*d (arit. 64 bits)

Sintaxis en ensamblador: **Nemón. %rs1, b, %rd**
donde *b* indica el número de desplazamientos y puede residir en un registro o ser un operando inmediato en función del bit I_{13}.

G.3.- Instrucciones aritméticas y lógicas enteras

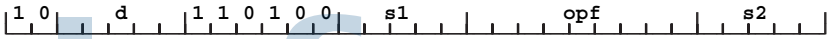


Nemónico	C.op.	Operación
add	00 0000	*r*s1+*b* → *r*d
addcc	01 0000	*r*s1+*b* → *r*d (1)
addc	00 1000	*r*s1+*b*+C → *r*d (2)
addccc	01 1000	*r*s1+*b*+C → *r*d (1)(2)
sub	00 0100	*r*s1-*b* → *r*d
subcc	01 0100	*r*s1-*b* → *r*d (1)
subc	00 1100	*r*s1-*b*-C → *r*d (2)
subccc	01 1100	*r*s1-*b*-C → *r*d (1)(2)
mulx	00 1001	*r*s1**b* → *r*d
sdivx	10 1101	*r*s1/*b* → *r*d (con signo)
udivx	00 1101	*r*s1/*b* → *r*d (sin signo)
taddcc	10 0000	*r*s1+*b* → *r*d (1) (3)
tsubcc	10 0001	*r*s1-*b* → *r*d (1) (3)
and	00 0001	*r*s1∧*b* → *r*d
andcc	01 0001	*r*s1∧*b* → *r*d (1)
andn	00 0101	*r*s1∧¬*b* → *r*d
andncc	01 0101	*r*s1∧¬*b* → *r*d (1)
or	00 0010	*r*s1∨*b* → *r*d
orcc	01 0010	*r*s1∨*b* → *r*d (1)
orn	00 0110	*r*s1∨¬*b* → *r*d
orncc	01 0110	*r*s1∨¬*b* → *r*d (1)
xor	00 0011	*r*s1⊕*b* → *r*d
xorcc	01 0011	*r*s1⊕*b* → *r*d (1)
xnor	00 0111	¬(*r*s1⊕*b*) → *r*d
xnorcc	01 0111	¬(*r*s1⊕*b*) → *r*d (1)
popc	10 1110 (4)	nº 1's en *b* → *r*d

Sintaxis en ensamblador: **Nem. %rs1,b,%rd**donde el operando *b* puede residir en un registro o ser un operando inmediato en función del bit I_{13}.

Notas:
(1) Las instrucciones cuyo nemónico termina por cc cambian los *flags* en función del resultado, tanto los correspondientes a 32 bits (*icc*) como los correspondientes a 64 (*xcc*).
(2) En estas instrucciones C representa el *flag* C del registro de condición de 32 bits (*icc*).
(3) Las instrucciones de suma y resta etiquetada, taddcc y tsubcc, operan igual que las instrucciones de suma y resta ordinarias, pero actúan de diferente forma sobre los *flags*: concretamente cambia el bit V del registro de 32 bits (*icc*) si algunos de los dos bits de menor peso de alguno de los operandos no es 0.
(4) En la instrucción popc el campo s1 está relleno con 0's.

G.4.- Instrucciones aritméticas de punto flotante



Nemón.	C.op. (opf)	Operación
fadd*f*	0 0100 00*ff*	*f*s1+*f*s2 → *f*d
fsub*f*	0 0100 01*ff*	*f*s1-*f*s2 → *f*d
fmul*f*	0 0100 10*ff*	*f*s1**f*s2 → *f*d
fsmuld	0 0110 1001	*f*s1**f*s2 → *f*d
fdmulq	0 0110 1110	*f*s1**f*s2 → *f*d
fdiv*f*	0 0100 11*ff*	*f*s1/*f*s2 → *f*d

Nemón.	C.op. (opf)	Operación
fsqrt*f*	0 0010 10*ff*	sqrt(*f*s2) → *f*d
fmov*f*	0 0000 00*ff*	*f*s2 → *f*d
fneg*f*	0 0000 01*ff*	-*f*s2 → *f*d
fabs*f*	0 0000 10*ff*	abs(*f*s2) → *f*d
f*f*to*g*	0 1100 *gg ff*	*f*s2 → *f*d (1)
fxto*g*	0 1000 *gg*00	*f*s2 → *f*d (2)
fito*g*	0 1100 *gg*00	*f*s2 → *f*d (2)

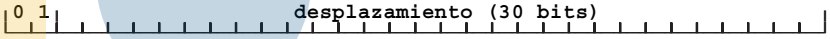
Sintaxis en ensamblador: **Nemón. %fs1, %fs2, %fd**
Las instrucciones de tabla de la derecha prescinden del primer operando. En estas tablas *f* y *g* indican la precisión de los operandos según lo siguiente:
simple: *f* = s, *ff* = 01; doble: *f* = d, *ff* = 10; cuádruple: *f* = q, *ff* = 11.

Notas:
(1) La instrucción f*f*to*g* convierte el formato de punto flotante *f* en el *g*.
(2) Las instrucciones fxto*g* y fito*g* convierten un entero o un entero extendido, respectivamente, en un número de punto flotante del formato *g*.

G.5.- Instrucciones de control de flujo

G.5.1.- Instrucciones de llamada y retorno a (de) procedimiento

call

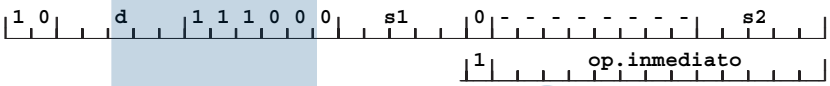


Sintaxis en ensamblador: **call etiqueta**
PC → *o*7; PC + 4*desplazamiento → PC
(la dirección representada por la etiqueta es PC + 4*desplazamiento)

return

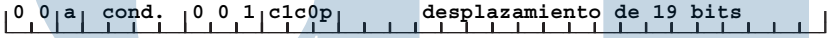
Sintaxis en ensamblador: **return dirección**
a → PC y recupera la ventana de registros anterior.
La dirección *a* se calcula sumando *r*s1 + *r*s2 (i = 0) o *r*s1 + inm. (i = 1).

jmpl



Sintaxis en ensamblador: **jmpl dirección, %rd**
PC → *r*d; *a* → PC
La dirección *a* se calcula sumando *r*s1 + *r*s2 (i = 0) o *r*s1 + inm. (i = 1).
*r*s1 + *r*s2 → PC (si i = 0)
*r*s1 + inm.→ PC (si i = 1)

G.5.2.- Bifurcaciones en función de los *flags* enteros con predicción

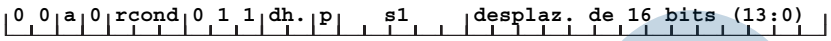


cc	Operación	cond	Bit analizado
a	Bifurcar siempre	1000	1
n	No bifurcar nunca	0000	0
ne (o nz)	Bifurcar si no igual	1001	¬Z
e (o z)	Bifurcar si igual	0001	Z
g	Bifurcar si mayor	1010	¬(Z∨(N⊕V))
le	Bifurcar si menor o igual	0010	Z∨(N⊕V)
ge	Bifurcar si mayor o igual	1011	¬(N⊕V)
l	Bifurcar si menor	0011	N⊕V
gu	Bifurcar si mayor (sin signo)	1100	¬(C∨Z)
leu	Bifurcar si menor o igual (sin signo)	0100	C∨Z
cc (o geu)	Bifurcar si C=0 (mayor o igual sin signo)	1101	¬C
cs (o lu)	Bifurcar si C=1 (menor sin signo)	0101	C
pos	Bifurcar si positivo	1110	¬N
neg	Bifurcar si negativo	0110	N
vc	Bifurcar si no *overflow*	1111	¬V
vs	Bifurcar si *overflow*	0111	V

Sintaxis en ensamblador: **b*cc*[,a][,pt|,pn] %icc|%xcc, etiqueta**
donde *cc* es una de las condiciones de la tabla, en función de los bits $I_{28:25}$ (cond); la etiqueta representa la dirección de destino del salto; a, si existe, indica que se debe anular la instrucción siguiente en caso de bifurcación efectiva (se pone a uno el bit I_{29}, o bit de anulación). La predicción viene dada por pt (*predict taken*, predicción de bifurcación efectiva) o pn (*predict not taken*, predicción de bifurcación no efectiva), el defecto es pt; esto activa el bit *p* (I_{19}), o bit de predicción. Para indicar si se toman los *flags* del registro de condición de 32 bits o de 64 bits, se indica %icc o %xcc, respectivamente; ello se manifiesta en los bits c1 y c0 ($I_{21:20}$), según la siguiente tabla:

Registro	icc	-	xcc	-
$c_{1:0}$	00	01	10	11

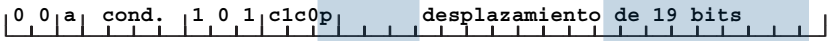
G.5.3.- Bifurcación en función de registro entero con predicción



cc	Operación	rcond	Condición
z	Bifurcar si 0	001	*r*s1=0
lez	Bifurcar si menor o igual que 0	010	*r*s1≤0
lz	Bifurcar si menor que 0	011	*r*s1<0
nz	Bifurcar si no 0	101	*r*s1≠0
gz	Bifurcar si mayor que 0	110	*r*s1>0
gez	Bifurcar si mayor o igual que 0	111	*r*s1≥0

Sintaxis en ensamblador: **br*cc* [,a][,pt|,pn] %rs1, etiqueta**
donde el único cambio respecto al apartado G.5.2 es que, en lugar de analizarse los *flags*, se analiza el contenido del registro *r*s1 y se bifurca o no según su contenido. Las posibles condiciones a analizar están descritas en la tabla y dependen de los bits $I_{27:25}$ (rcond). El desplazamiento en esta instrucción se codifica en dos partes: los dos bits más altos en $I_{21:20}$ y el resto en $I_{13:0}$.

G.5.4.- Bifurcaciones en función de los *flags* de punto flotante con predicción



cc	Operación	cond
a	Bifurcar siempre	1000
n	No bifurcar nunca	0000
u	Bifurcar si no ordenado	0111
g	Bifurcar si mayor	0110
ug	Bifurcar si no ordenado o mayor	0101
l	Bifurcar si menor	0100
ul	Bifurcar si no ordenado o menor	0011
lg	Bifurcar si menor o mayor	0010
ne (o nz)	Bifurcar si no igual	0001
e (o z)	Bifurcar si igual	1001
ue	Bifurcar si desordenado o igual	1010
ge	Bifurcar si mayor o igual	1011
uge	Bifurcar si no ordenado, mayor o igual	1100
le	Bifurcar si menor o igual	1101
ule	Bifurcar si no ordenado, menor o igual	1110
o	Bifurcar si ordenado	1111

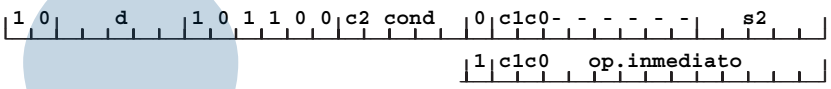
Sintasis en ensamblador: **fb*cc* [,a][,pt|,pn] %fcc*n*, etiqueta**
donde *cc* es una de las condiciones de la tabla en función de los bits $I_{28:25}$ (cond) y fcc*n* representa uno de los 4 registros de condición de punto flotante, esto cambia los bits c1:c0 ($I_{21:20}$) según la siguiente tabla:

Registro	fcc0	fcc1	fcc2	fcc3
$c_{2:0}$	00	01	10	11

La condición "no ordenado" se refiere a que alguno de los operandos en punto flotante sea erróneo (NaN).
El significado de a, pt y pn es el mismo que en el apartado G.5.3

G.6.- Instrucciones de transferencia condicional

mov*cc*

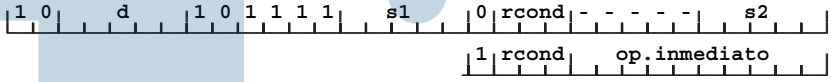


Sintaxis en ensamblador:**mov*cc* %icc|%xcc|%fcc*n*, b, %rd**
donde *cc* es una de las condiciones que determinan los bits $I_{17:14}$ (cond), según las tablas ya descritas en los apartados G.5.2 y G.5.4. Se tomará una u otra tabla en función de que se indique un registro de indicadores entero o de punto flotante (%icc,%xcc o %fcc*n*). En la instrucción esto lo determinan los bits c2:c1:c0 (I_{18}:$I_{12:11}$). La correspondencia de esos bits con los registros de condición se muestra en la tabla de la derecha.

Registro	fcc0	fcc1	fcc2	fcc3	icc	-	xcc	-
$c_{2:0}$	000	001	010	011	100	101	110	111

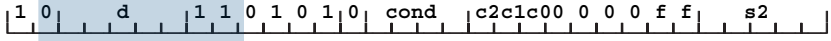
Esta instrucción efectúa la transferencia entre *b* y *r*d, donde *b* puede ser un operando inmediato o el registro *r*s2, si se cumple la condición especificada.

movr*cc*



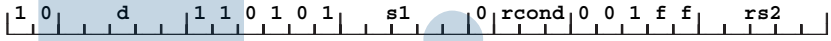
Sintaxis en ensamblador:**movr*cc* rs1, b, %rd**
donde *cc* es una de las condiciones que determinan los bits $I_{12:10}$ (rcond), según la tabla ya descrita en el apartado G.5.3. Estas condiciones se evalúan sobre el registro *r*s1.
Esta instrucción efectúa la transferencia entre *b* y *r*d, donde *b* puede ser un operando inmediato o el registro *r*s2, si se cumple la condición especificada.

fmov*fcc*



Sintaxis en ensamblador: **fmov*fcc* [%icc|%xcc|%fcc*n*] %fs2, %fd**
Esta instrucción actúa de la misma forma que mov*cc*, con la diferencia de que efectúa la transferencia sobre registros de punto flotante. Si se cumple la condición, copia *f*s2 en *f*d. *f* representa la precisión de los números en punto flotante y determina los bits *ff* según se mostró en el apartado G.4.

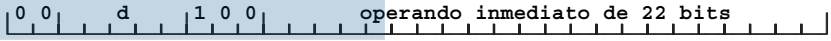
fmovr*fcc*



Sintaxis en ensamblador: **fmovr*fcc* %rs1 %fs2, %fd**
Esta instrucción actúa igual que movr*cc*, con la diferencia de que efectúa la transferencia sobre registros de punto flotante. Si se cumple la condición especificada sobre el registro *r*s1, copia *f*s2 en *f*d. *f* representa la precisión de los números en punto flotante y determina los bits *ff* según se mostró en el apartado G.4.

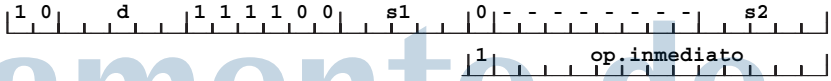
G.7.- Instrucciones diversas

sethi



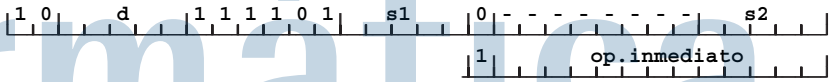
Sintaxis en ensamblador: **sethi inm., %rd**
$0 \rightarrow r\mathrm{d}_{9:0}$; inm $\rightarrow r\mathrm{d}_{31:10}$
El nemónico nop es un caso especial de esta operación, en que inm = 0 y d = 0.

save



Sintaxis en ensamblador: **save %rs1, b, %rd** (ver apartado G.2), habitualmente *r*s1 = *r*d = sp y *b* es una constante con el tamaño de la trama de pila que se quiere reservar.
Cambia el apuntador de ventana activa hacia adelante generando un desvío si se produce un desbordamiento de ventana, por otra parte, una vez resuelto el desvío, si éste ocurre, efectúa una operación de suma, refiriéndose los operandos fuentes a la ventana antigua y el destino (*r*d) a la nueva.

restore



Sintaxis en ensamblador: **restore %rs1, b, %rd** (ver apartado G.2).
Cambia el apuntador de ventana activa hacia atrás generando un desvío si se produce un subdesbordamiento de ventana, por otra parte una vez resuelto el desvío, si éste ocurre, efectúa una operación de suma, refiriéndose los operandos fuentes a la ventana antigua y el destino (*r*d) a la nueva. Frecuentemente esta instrucción se aplica en la forma **restore %g0, %g0, %g0**.

H

CÓDIGO ASCII DE 7 BITS

Dec.	Hex.	Octal	Car.	Dec.	Hex.	Octal	Car.	Dec.	Hex.	Octal	Car.	Dec.	Hex.	Octal	Car.
0	00	0	NUL	32	20	40		64	40	100	@	96	60	140	‘
1	01	1	SOH	33	21	41	!	65	41	101	A	97	61	141	a
2	02	2	STX	34	22	42	"	66	42	102	B	98	62	142	b
3	03	3	ETX	35	23	43	#	67	43	103	C	99	63	143	c
4	04	4	EOT	36	24	44	$	68	44	104	D	100	64	144	d
5	05	5	ENQ	37	25	45	%	69	45	105	E	101	65	145	e
6	06	6	ACK	38	26	46	&	70	46	106	F	102	66	146	f
7	07	7	BEL	39	27	47	’	71	47	107	G	103	67	147	g
8	08	10	BS	40	28	50	(	72	48	110	H	104	68	150	h
9	09	11	HT	41	29	51	)	73	49	111	I	105	69	151	i
10	0A	12	LF	42	2A	52	*	74	4A	112	J	106	6A	152	j
11	0B	13	VT	43	2B	53	+	75	4B	113	K	107	6B	153	k
12	0C	14	FF	44	2C	54	,	76	4C	114	L	108	6C	154	l
13	0D	15	CR	45	2D	55	-	77	4D	115	M	109	6D	155	m
14	0E	16	SO	46	2E	56	.	78	4E	116	N	110	6E	156	n
15	0F	17	SI	47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20	DLE	48	30	60	0	80	50	120	P	112	70	160	p
17	11	21	DC1	49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22	DC2	50	32	62	2	82	52	122	R	114	72	162	r
19	13	23	DC3	51	33	63	3	83	53	123	S	115	73	163	s
20	14	24	DC4	52	34	64	4	84	54	124	T	116	74	164	t
21	15	25	NAK	53	35	65	5	85	55	125	U	117	75	165	u
22	16	26	SYN	54	36	66	6	86	56	126	V	118	76	166	v
23	17	27	ETB	55	37	67	7	87	57	127	W	119	77	167	w
24	18	30	CAN	56	38	70	8	88	58	130	X	120	78	170	x
25	19	31	EM	57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32	SUB	58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33	ESC	59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34	FS	60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35	GS	61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36	RS	62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37	US	63	3F	77	?	95	5F	137	_	127	7F	177	DEL

Significado de los caracteres de control:

NUL: *Null*, carácter nulo
SOH: *Start of header*, inicio de cabecera
STX: *Start text*, comienzo de texto
ETX: *End text*, fin de texto
EOT: *End of transmison*, fin de transmisión
ENQ: *Enquiry*, petición
ACK: *Acknowledge*, reconocimiento petición
BEL: *Bell*, sonido
BS: *Backspace*, retroceso
HT: *Horizontal tab*, tabulador horizontal
LF: *Line feed*, salto de línea
VT: *Vertical tab*, tabulador vertical
FF: *Form feed*, salto de página
CR: *Carriage return*, retorno de carro
SO: *Shift out*, desplazamiento de salida
SI: *Shift in*, desplazamiento de entrada

DLE: *Data link escape*, escape de enlace de datos
DC1: *Device control 1*, control de dispositivo 1 (Xon)
DC2: *Device control 2*, control de dispositivo 2
DC3: *Device control 3*, control de dispositivo 3 (Xoff)
DC4: *Device control 4*, control de dispositivo 4
NAK: *No acknowledge*, reconocimiento negativo
SYN: *Synchronous*, parada de sincronismo
ETB: *End of transmitted block*, fin bloque transmitido
CAN: *Cancel*, cancelado
EM: *End of mediun*, fin del medio
SUB: *Substitution*, sustituir
ESC: *Escape*, escape
FS: *File separator*, separador de fichero
GS: *Group separator*, separador de grupo
RS: *Record separator*, separador de registro
US: *Unit separator,* separador de unidad