

## Modos de Direcccionamiento del MC68HC11

Son diferentes formas en las que el CPU de una computadora accede a memoria externa para ejecutar la instrucción.

Soporta 6 modos detonados como:

- Direcccionamiento inherente (INH)
- Direcccionamiento inmediato (IMM)
- Direcccionamiento directo(DIR)
- Direcccionamiento extendido (EXT)
- Direcccionamiento indexado (IND, X) o (IND, Y)
- Direcccionamiento relativo (REL)



Suelen ser usados por otros MC.

De lo mas sencillo a lo mas complicado.

## Direccionamiento inherente (INH)

- instrucciones sencillas (no requiere parámetros adicionales para que el CPU funcione)
- carecen de operando
- codigo de instrucción de 8 a 16 bits.
- no comparten mnemónico con instrucciones con otros métodos de direccionamiento.

# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

## SET DE INSTRUCCIONES DEL MICROCHIPS

INSTRUCION	R/RM		DIR		INDX		INDY		EXT		IMM				REL				Bandera de estado		
	opcode	Ciclo	Byte	opcode	Ciclo	Byte	opcode	Ciclo	Byte	opcode	Ciclo	Byte	opcode	Ciclo	Byte	opcode	Ciclo	Byte	X	N	OV / Z
1	aha	—	—	—	—	—	—	—	—	—	18	2	1	—	—	—	—	—	—	X	X
2	ahb	—	—	—	—	—	—	—	—	—	18	2	1	—	—	—	—	—	—	X	X
3	aby	—	—	—	—	—	—	—	—	—	18	2	1	—	—	—	—	—	—	—	—
4	adca	80	2	2	95	3	2	A0	4	2	18A0	5	3	80	4	3	—	—	—	X	X
5	adclb	CS	2	2	DF	3	2	F0	4	2	18DF	5	3	F0	4	3	—	—	—	X	X
6	adca	80	2	2	9B	3	2	A8	4	2	18A8	5	3	80	4	3	—	—	—	X	X
7	adclb	CS	2	2	D0	3	2	B0	4	2	18B0	5	3	F0	4	3	—	—	—	X	X
8	adca	CS	4	1	D3	5	2	C0	4	2	18C0	5	3	C0	4	3	—	—	—	—	—
9	adca	84	2	2	94	3	2	B4	4	2	18B4	5	3	84	4	3	—	—	—	—	X
10	adclb	C4	2	2	D4	3	2	E4	4	2	18E4	5	3	F4	4	3	—	—	—	—	X
11	asl	—	—	—	—	—	—	—	—	—	18	2	1	—	—	—	—	—	—	—	—
12	asr	—	—	—	—	—	—	—	—	—	18	2	1	—	—	—	—	—	—	—	—
13	aslb	—	—	—	—	—	—	—	—	—	88	2	1	—	—	—	—	—	—	—	—
14	asld	—	—	—	—	—	—	—	—	—	58	2	1	—	—	—	—	—	—	—	—
15	asr	—	—	—	—	—	—	—	—	—	5	3	1	—	—	—	—	—	—	X	X
16	asr	—	—	—	—	—	—	—	—	—	67	2	1	—	—	—	—	—	—	X	X
17	asrl	—	—	—	—	—	—	—	—	—	47	2	1	—	—	—	—	—	—	X	X
18	bcc	—	—	—	—	—	—	—	—	—	17	2	1	—	—	—	—	—	—	—	—
19	bcc	—	—	—	—	—	—	—	—	—	24	3	2	—	—	—	—	—	—	—	—
20	bcc	—	—	—	—	—	—	—	—	—	25	1	2	—	—	—	—	—	—	X	B
21	breq	—	—	—	—	—	—	—	—	—	27	3	2	—	—	—	—	—	—	—	—
22	bgt	—	—	—	—	—	—	—	—	—	25	3	2	—	—	—	—	—	—	—	—
23	bgt	—	—	—	—	—	—	—	—	—	25	1	2	—	—	—	—	—	—	—	—
24	bhi	—	—	—	—	—	—	—	—	—	22	1	2	—	—	—	—	—	—	—	—
25	bhl	—	—	—	—	—	—	—	—	—	24	1	2	—	—	—	—	—	—	—	—
26	bhlb	BS	2	2	85	3	2	A5	4	2	18A5	5	3	85	4	3	—	—	—	X	B
27	bhlb	CS	2	2	D5	3	2	E5	4	2	18E5	5	3	F5	4	3	—	—	—	X	B
28	blo	—	—	—	—	—	—	—	—	—	29	1	2	—	—	—	—	—	—	—	—
29	blo	—	—	—	—	—	—	—	—	—	25	3	2	—	—	—	—	—	—	—	—
30	blo	—	—	—	—	—	—	—	—	—	23	1	2	—	—	—	—	—	—	—	—
31	bhl	—	—	—	—	—	—	—	—	—	20	1	2	—	—	—	—	—	—	—	—
32	bami	—	—	—	—	—	—	—	—	—	28	3	2	—	—	—	—	—	—	—	—
33	bami	—	—	—	—	—	—	—	—	—	26	3	2	—	—	—	—	—	—	—	—
34	bmi	—	—	—	—	—	—	—	—	—	26	1	2	—	—	—	—	—	—	—	—
35	bmi	—	—	—	—	—	—	—	—	—	20	1	2	—	—	—	—	—	—	—	—
36	bmi	—	—	—	—	—	—	—	—	—	28	1	2	—	—	—	—	—	—	—	—

-----

OPCODE : código de instrucción (numero que le da el fabricante a cada instrucción)

Ciclo: ciclos que tarda

Byte: numero de bytes que tendremos en la instrucción

---

Directiva de ensamblador: SIRVEN PARA DAR ESTRUCTURA AL PROGRAMA

ORG \$8000, END

↑  
Localidad (hexadecimal)

-Se pueden tener varios ORG, porque puede haber varios sectores.

PROGRAMA SENCILLO, COMPILANDO

ORG \$8000	\$8000	01	NOP
	\$8001	08	INX
	\$8002	09	DEX
	\$8003	18	INY
	\$8004	08	
	\$8005	18	DEY
	\$8006	09	
	\$8007	3D	MUL
	\$8008	8F	XGDX
	\$8009	18	
	\$800A	8F	XGDY
	\$800B	40	NEGA
	\$800C	50	NEGB
	\$800D		
	\$800E		
	\$800F		
END			

INY : usa dos bytes por eso se usan dos localidades de memoria



Los caracteres se ponen de la siguiente manera en código ascii : 'f, 'k. y solo soporta un caracter.

### Direccionamiento directo (DIR).

- un operando de 8 bits.
- Codigo de instrucción de 8 – 16 bits.
- Solo puede acceder a la memoria RAM (256 localidades)
- Desde la localidad \$0000 - \$00FF
- no van precedidos por un #

		ORG \$8000			
8000	96	45	LDAA	\$45	
8002	D6	0B	LDAB	11	
8004	DC	17	LDD	\$17	
8006	DE	0F	LDX	15	
8008	9B	7C	ADDA	\$7C	
800A	94	F0	ANDA	\$F0	
800C	18	DE	LDY	\$AB	
		END			

  

\$8000	96	}	LDAA	\$45
\$8001	45			
\$8002	D6	}	LDAB	11
\$8003	0B			
\$8004	DC	}	LDD	\$17
\$8005	17			
\$8006	DE	}	LDX	15
\$8007	0F			
\$8008	9B	}	ADDA	\$7C
\$8009	7C			
\$800A	94	}	ANDA	\$F0
\$800B	F0			
\$800C	18	}	LDY	\$AB
\$800D	DE			
\$800E	AB			
\$800F				

### Direccionamiento extendido (EXT)

- Un operando de 16 bits; se le da tratamiento de dirección
- Codigo de instrucción de 8 – 16 bits.
- Accede a cualquier parte de memoria (ROM Y RAM).
- Un poco mas lento y mas memoria.
- no le antecede un #.
- Tenemos que saber en que parte de la memoria ram estamos almacenando datos.

					\$8000	B6			
					\$8001	45			
					\$8002	7C			
					\$8003	F6			
					\$8004	05			
					\$8005	FB			
					\$8006	FC			
					\$8007	17			
					\$8008	89			
					\$8009	FE			
					\$800A	FD			
					\$800B	E8			
					\$800C	BB			
					\$800D	07			
					\$800E	CB			
					\$800F	B4			
					\$8010	F0			
					\$8011	B1			
					\$8012	18			
					\$8013	FE			
					\$8014	AB			
					\$8015	CD			

  

				ORG	\$8000				
8000	B6	45	7C	LDA	\$457C				
8003	F6	05	FB	LDAB	1531				
8006	FC	17	89	LDD	\$1789				
8009	FE	FD	E8	LDX	65000				
800C	BB	07	CB	ADDA	\$7CB				
800F	B4	F0	B1	ANDA	\$F0B1				
8012	18	FE	AB	LDY	\$ABCD				
				END					

### Direccinamiento indexado

-Un operando de 8 bits, se le da tratamiento de desplazamiento algebraico.

-precedido por una ", " y una x ó y.

- código de instrucción de 8 a 16 bits.

### LISTADO del programa

					\$8000	A6			
					\$8001	45			
					\$8002	18			
					\$8003	E6			
					\$8004	67			
					\$8005	EC			
					\$8006	17			
					\$8007	CD			
					\$8008	EE			
					\$8009	F1			
					\$800A	18			
					\$800B	AB			
					\$800C	07			
					\$800D	A4			
					\$800E	F0			
					\$800F	18			
					\$8010	EE			
					\$8011	AB			
					\$8012				

  

				ORG	\$8000				
3000	A6	45		LDA	\$45,X				
3002	18	E6	67	LDAB	\$67,Y				
3005	EC	17		LDD	\$17,X				
3007	CD	EE	F1	LDX	\$F1,Y				
300A	18	AB	07	ADDA	\$07,Y				
300D	A4	F0		ANDA	\$F0,X				
300F	18	EE	AB	LDY	\$AB,Y				
				END					



- su operando siempre se expresa como un etiqueta.

Diagram illustrating the calculation of the two's complement of 8 and its storage in memory.

Calculation of the two's complement of 8:

$$\begin{array}{r}
 1111\ 0111 \\
 +\ 0000\ 0001 \\
 \hline
 1111\ 1000 \\
 +\ 1 \\
 \hline
 0000\ 1000 \quad (+08)
 \end{array}$$

The result, 0000 1000, is the two's complement of 8.

Memory layout (addresses and values):

Address	Value
\$8000	CE
\$8001	06
\$8002	FD
\$8003	01
\$8004	01
\$8005	01
\$8006	01
\$8007	01
\$8008	09
\$8009	26
\$800A	
\$800B	7E
\$800C	80
\$800D	00
\$800E	

Offsets (relative to \$8000):

- 8
- 7
- 6
- 5
- 4
- 3
- 2
- 1