A. Microprocesador 6800

El Motorola MC6800 (6800) es un microprocesador fabricado por Motorola, lanzado al mercado en 1975. La estructura general del 6800 se presenta en la figura 1.

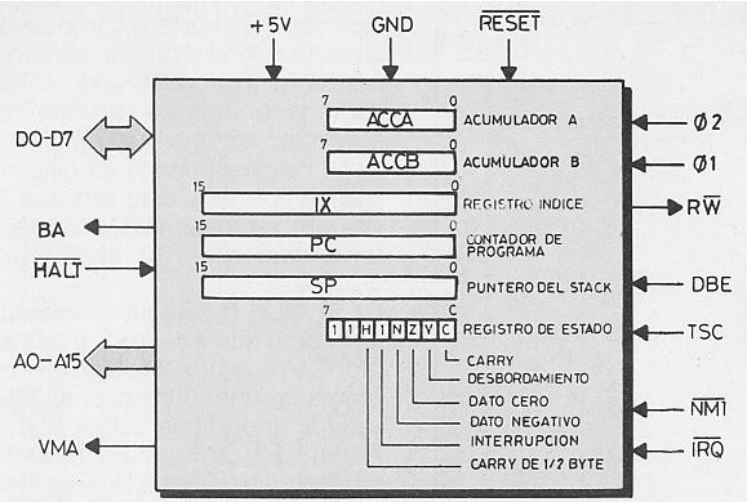


Figura 1. Microprocesador 6800.

En la figura pueden identificarse los siguientes terminales:

- A_0-A_{15}: bus de direcciones de 16 líneas que permite direccionar 64 KB de memoria
- D_0-D_7: bus de datos de 8 líneas, para lectura/escritura de datos de 8 bits (registros de memoria)
- BA (buses disponibles): con valor lógico 0 la línea indica que el microprocesador tiene el control de los buses, y con valor lógico 1, que los buses están disponibles para uso de los dispositivos (el micro está detenido o en espera).
- HALT!: el valor lógico 1 de este terminal implica que el microprocesador funciona normalmente, mientras que el valor lógico 0 indica que el micro completa la instrucción actual y se detiene.
- VMA (dirección de memoria válida): esta señal indica a los dispositivos del sistema que la dirección presente en el bus de direcciones es válida.
- $Ø_2$: este terminal es la entrada de la señal de fase 2 del reloj del sistema.
- $Ø_1$: este terminal es la entrada de la señal de fase 1 del reloj del sistema.
- RW!: indica a las unidades del sistema si el microprocesador se encuentra en estado de lectura (valor lógico 1) o de escritura (valor lógico 0)
- DBE (habilitación de la barra de datos): esta señal (valor lógico 1) permite que los datos se transfieran al bus.
- TSC: cuando esta línea asume valor lógico 1, el bus de direcciones y la línea R/W! pasan a estado de alta impedancia, mientras que las líneas VMA y BA pasan a nivel lógico 0.
- NMI! (interrupciones no enmascarables): cuando un dispositivo envía un valor lógico 0 a este terminal, el microprocesador completa la ejecución de la instrucción actual, salva el estado del sistema y ejecuta la rutina de atención de interrupciones.

- IRQ! (interrupciones enmascarables): cuando un dispositivo envía un valor lógico 0 a este terminal, el microprocesador completa la ejecución de la instrucción actual, salva el estado del sistema y ejecuta la rutina de atención de interrupciones. Esto está sujeto al bit I del Registro de Códigos de Condición (CCR).
- +5V: corresponde al terminal positivo de la fuente de alimentación del sistema.
- GND: corresponde al terminal negativo de la fuente de alimentación del sistema.
- RESET!: este terminal se utiliza para arrancar el microprocesador.

Además el 6800 cuenta con los siguientes registros:

- Acumulador A (**Acc. A**): es un registro de 8 bits que se utiliza para almacenamiento temporario de las operaciones realizadas por la ALU.
- Acumulador B (**Acc. B**): es un registro de 8 bits que se utiliza para almacenamiento temporario de las operaciones realizadas por la ALU.
- Registro Índice (**IDX**): es un registro de 16 bits (2 bytes) que se utiliza para modificar direcciones en el modo de direccionamiento indexado.
- Contador de Programa (**PC**): es un registro de 16 bits (2 bytes) que contiene la dirección de la próxima instrucción a ejecutarse. Se incrementa automáticamente al iniciar el ciclo de ejecución de la instrucción actual.
- Registro Puntero de Pila (**SP**): registro de 16 bits que contiene la dirección de un segmento de memoria que almacena el estado de los registros del microprocesador en caso de atender una interrupción o ejecutar un salto a subrutina.
- Registro de Códigos de Condición (**CCR**): es un registro de 8 bits usado por las instrucciones de bifurcación para determinar si deben producirse o no saltos en la secuencia de un programa. Cabe destacar que los bits del CCR se modifican con la ejecución de las instrucciones, por lo que cada bit de este registro tiene un significado específico:

1	1	H	I	N	Z	V	C
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**

  Los bits 6 y 7 toman siempre el valor lógico 1, y no se usan para condicionar bifurcaciones en el programa. El bit *H* indica, con valor 1, si una operación aritmética produce un arrastre de medio byte en el resultado. El bit I indica, con valor 1, que se inhiben las interrupciones enmascarables, y con valor 0 que el microprocesador admite interrupciones por la línea IRQ!. El bit *N* asume valor 1 si el resultado de una operación aritmética produce un valor negativo (bit más significativo 1), 0 en caso contrario. El bit *Z* asume valor 1 si el resultado de una operación aritmética produce un valor cero, 0 en caso contrario. El bit *V* asume valor 1 si como resultado de una operación aritmética se produce desborde del acumulador, 0 en caso contrario. Finalmente, el bit *C* asume valor lógico 1 si al ejecutarse una operación se produce un arrastre desde el bit más significativo del resultado, 0 en caso contrario.

B. Modos de Direccionamiento

El microprocesador 6800 maneja los siguientes modos de direccionamiento:

- **Inherente**, implícito o implicado: la ejecución de instrucciones en este modo no requiere de datos de memoria, se realiza exclusivamente sobre los registros internos del procesador. Las instrucciones en este modo son de un byte.
- **Inmediato**: en este modo, los datos necesarios para la ejecución de la instrucción se encuentran en la/s siguiente/s posición/es de memoria a continuación del código de operación (*OPCODE*). Las instrucciones en este modo son de 2 (en algunos casos de 3) bytes.
- **Directo**: en este modo, la dirección de memoria que sigue al código de operación contiene la dirección en la que se encuentra el operando. Las direcciones especificadas son de 8 bits, es decir que sólo pueden referenciarse

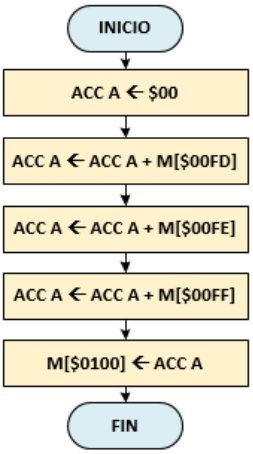
posiciones en el rango 00 h a FF h. Las instrucciones en este modo son de 2 bytes.

- **Extendido**: en este modo, la dirección de memoria que sigue al código de operación contiene la dirección en la que se encuentra el operando. Las direcciones especificadas son de 16 bits, de modo que puede accederse a todo el mapa de direccionamiento del sistema (64 KB). Las instrucciones en este modo son de 3 bytes.
- **Indexado**: en este modo, la dirección de memoria que sigue al código de operación contiene un valor denominado "desplazamiento" que se suma al valor del registro índice (sin modificar el valor del IDX) para determinar la dirección efectiva del operando de la instrucción. Las instrucciones en este modo son de 2 bytes.
- **Relativo**: en este modo, el valor (denominado "*offset*") que acompaña al código de operación indica la cantidad de posiciones de memoria que deben saltarse hacia adelante (*offset* positivo) o hacia atrás (*offset* negativo) si la condición analizada se cumple. Si tal condición no se verifica, entonces se continúa la ejecución secuencial. Las instrucciones en este modo son de 2 bytes.

C. Ejemplos

Ejemplo 1: Escriba un programa para M6800 que sume los valores almacenados en las direcciones $00FD, $00FE y $00FF, guardando el resultado en la dirección $0100. Indique, para cada instrucción, el modo de direccionamiento utilizado. Además, suponiendo que el programa se inicia en la dirección $2000 indique para cada instrucción el valor del PC.

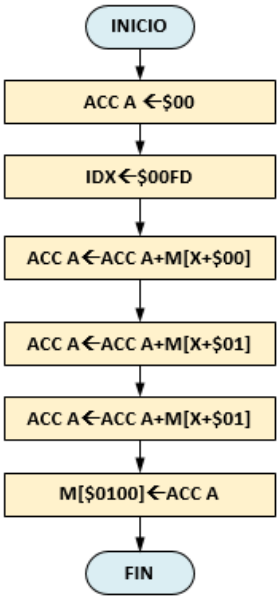
DIAGRAMA DE FLUJO



PC	Tamaño (bytes)	PROGRAMA	MODO DE DIRECCIONAMIENTO
$2000	1	CLRA	Inherente
$2001	2	ADDA $FD	Directo
$2003	2	ADDA $FE	Directo
$2005	2	ADDA $FF	Directo
$2007	3	STAA $0100	Extendido
$200A	1	SWI	Inherente

Ejemplo 2: Modifique el programa anterior de modo que la suma de los datos se realiza utilizando el modo indexado..

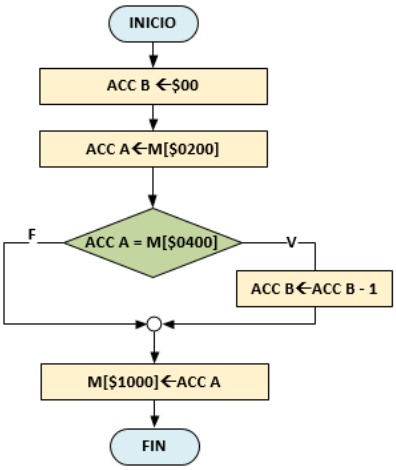
DIAGRAMA DE FLUJO



PC	Tamaño (bytes)	PROGRAMA	MODO DE DIRECCIONAMIENTO
$2000	1	CLRA	Inherente
$2001	3	LDX #$00FD	Inmediato
$2004	2	ADDA $00,X	Indexado
$2006	2	ADDA $01,X	Indexado
$2008	2	ADDA $02,X	Indexado
$200A	3	STAA $0100	Extendido
$200D	1	SWI	Inherente

Ejemplo 3: Escriba un programa para M6800 que, dados los valores almacenados las direcciones $0200 y $0400, determine si éstos son iguales o no. Considere que el resultado de la comparación se guardará en la dirección $1000: $01 para valores iguales y $00 para valores distintos. Indique para cada instrucción el modo de direccionamiento utilizado. Además, suponiendo que el programa se inicia en la dirección $0400, consigne el valor del PC para cada instrucción y el tamaño (en bytes) de cada una de éstas.

DIAGRAMA DE FLUJO



PC	Tamaño (bytes)	PROGRAMA	MODO DE DIRECCIONAMIENTO
$0400	1	CLRB	Inherente
$0401	3	LDAA $0200	Extendido
$0404	3	CMPA $0400	Extendido
$0407	2	BNE **$01**	Relativo
$0409	1	INCB	Inherente
$040A	3	STAB $1000	Extendido
$040D	1	SWI	Inherente

Para obtener el valor del *offset* que especifica la instrucción de salto se utiliza la siguiente expresión:

$$PC_{final} = PC_{actual} + 2 + Offset$$

despejando se obtiene

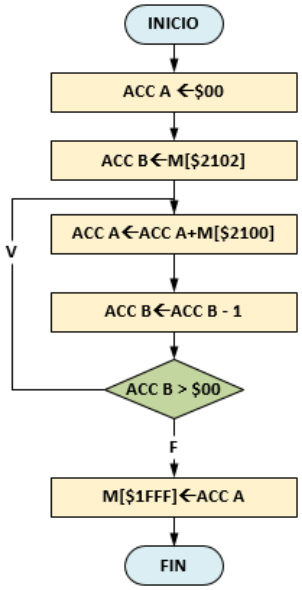
$$Offset = PC_{final} - (PC_{actual} + 2)$$

Para el ejemplo el *offset* se calcula como sigue:

Offset=$040A – ($0407+2)
Offset=$040A – $0409
Offset=**1**
Offset=**$01**

Ejemplo 3: Escriba un programa para M6800 que, dados los valores de las direcciones $2100 a $2102, calcule el producto mediante sumas de éstos y guarde el resultado obtenido en la posición $1FFF. Indique para cada instrucción el modo de direccionamiento utilizado. Además, suponiendo que el programa se inicia en la dirección $0100, consigne el valor del PC para cada instrucción y el tamaño en bytes de la instrucción.

DIAGRAMA DE FLUJO



PC	Tamaño (bytes)	PROGRAMA	MODO DE DIRECCIONAMIENTO
$0100	1	CLRA	Inherente
$0101	3	LDAB $2102	Extendido
$0104	3	ADDA $2100	Extendido
$0107	1	DECB	Inherente
$0108	2	CMPB #$00	Inmediato
$010A	2	BGT **$F8**	Relativo
$010C	3	STAA $1FFF	Extendido
$010F	1	SWI	Inherente

Calculando el *offset*

$$Offset = PC_{final} - (PC_{actual} + 2)$$

Offset=$0104 – ($010A+2)
Offset=$0104 – $010C
Offset=-**8**
Offset=**$F8 (notación complemento)**

PROBLEMAS A RESOLVER

1. Responda:

a) Defina el μp 6800 desde un punto de vista aislado y desde un punto de vista integrado a la arquitectura general del sistema; caracterícelo (suministro de energía que necesita, tiempo que le lleva la ejecución de las instrucciones, velocidad a la que trabaja)

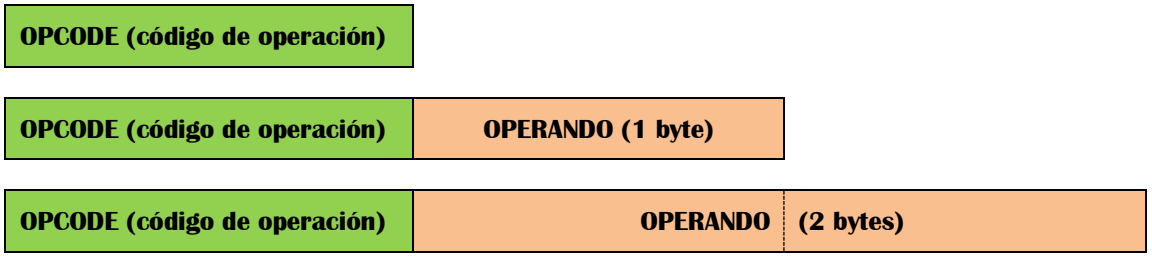
b) Realice un cuadro comparativo con las principales características de las familias de microprocesadores de Intel, a partir del 8086 a la fecha, estableciendo ventajas y desventajas entre los diferentes modelos.

c) ¿Qué es el registro de código de condiciones (CCR)? ¿Qué situaciones modifican sus bits?

d) ¿Qué es el set de instrucciones del μp 6800? ¿Cómo se agrupan dichas instrucciones?

e) Describa la estructura genérica de instrucciones (1, 2 y 3 bytes) para todos los modos de direccionamiento del μp 6800¿Cómo encuentra el valor del operando en los distintos modos de direccionamiento? Proponga dos ejemplos específicos para cada caso.

2. Teniendo en cuenta que el MP6800 maneja los siguientes formatos de instrucción:



Indique, para cada uno de ellos, 3 ejemplos describiendo el propósito de la instrucción utilizada, modo de direccionamiento, tamaño en bytes y código hexadecimal correspondiente (ver Resumen de Instrucciones MP6800).

3. Complete la siguiente tabla, indicando por cada instrucción

a) Modo de direccionamiento

b) Tamaño en bytes (#)

c) Ciclos de reloj (~) y

d) Descripción de la operación (consulte el Resumen de Instrucciones del 6800).

Instrucción	Modo de Direccionamiento	Tamaño (bytes)	Ciclos de Reloj (~)	Operación
a) LDAB $0C	*Directo*	*2*	*3*	*M[$000C]→ ACC B*
b) LDX $4321				
c) STAA $04,X				
d) INX				
e) ANDA #$01				
f) BLE $F4				
g) CLC				
h) BRA $10				

Instrucción	Modo de Direccionamiento	Tamaño (bytes)	Ciclos de Reloj (~)	Operación
i) RORB				
j) BNE $05				
k) SEC				

4. **Dadas las siguientes instrucciones en modo indexado identifique el desplazamiento, dirección utilizada por cada instrucción y el objetivo de cada una de éstas.**

Instrucción	Registro Índice	Desplazamiento	Dirección	Objetivo de la instrucción
LDAA $09,X	$32C2			
SUBA $A3,X	$17B1			
INC $11,X	$8816			
ADDA $00,X	$1ED4			
CLR $45,X	$7C12			
CMPA $B1,X	$C9E1			
ANDA $01,X	$1FDD			

5. **Las instrucciones de salto o bifurcación permiten que las operaciones se realicen de un modo no secuencial, emulando de alguna manera las estructuras condicionales o iterativas de los lenguajes de alto nivel. En la siguiente tabla complete con los valores apropiados de *offset* y PC_{Final} correspondiente a instrucciones de salto.**

PC Actual	Offset	PC Final
$8803		$87BB
$90D5	$0F	
$A252	$8F	
$9B51		$9B13
$D0BF		$D0FC
$9484	$E7	
$B502		$B56A
$8CA6	$76	

6. **Dado el siguiente programa indique para cada instrucción: PC, OPCODE, modo de direccionamiento, cantidad de ciclos y tamaño de instrucción. ¿Cuál es el objetivo del programa? Calcule el valor del offset de la etiqueta Salto.**

PC	OPCODE	Modo de Direccionamiento		Instrucción	Tamaño (bytes)	Ciclos (~)
$1000				LDAA $32		
				LDAB $0035		
				SBA		
				CMPA #$00		
				BEQ SALTO		
				LDAA #$FF		
			SALTO	STAA $10		
				SWI		

7. **Sabiendo que el siguiente programa (escrito para el emulador SDK6800 V1.08) determina si el contenido de una posición de memoria es un valor positivo o no, modifíquelo de modo que detecte si el valor analizado es positivo, negativo o cero. Considere que la dirección $0020 guardará el resultado del programa según el siguiente criterio: $01 positivo, $00 cero y $FF negativo. ¿Cuál es el *offset* de la etiqueta *positivo* del programa original?**

```
inicio .org $00       Indica que el programa inicia en la dirección $0000
 ldab #$01            Carga el Acc. B con el valor %00000001 ($01)
 ldaa num1            Carga el Acc. A con el valor de la posición num1
 cmpa #$00            Compara el contenido del Acc. A con el valor $00
 bgt positivo         Salta a la etiqueta positivo si Acc. A es mayor que $00
 ldab #$FF            Carga el Acc. B con el valor %11111111 ($FF)
positivo stab $20     Almacena el contenido del Acc. B en la dirección $0020
num1 .byte $18        Define una posición de memoria como num1 y le asigna $18
 .end                 Finaliza el programa
```

8. **Dado el siguiente programa que intercambia el contenido de 2 posiciones de memoria (*num1* y *num2*), escriba otro que ordene (de forma ascendente) el contenido de 3 posiciones de memoria (*num1*, *num2* y *num3*).**

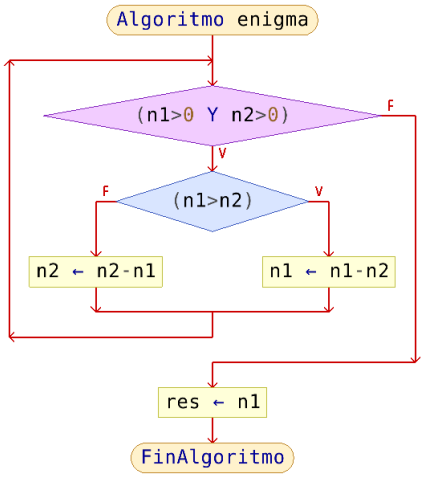
```
inicio .org $00       Indica que el programa inicia en la dirección $0000
 ldaa num1            Carga el Acc. A con el valor de la posición num1
 ldab num2            Carga el Acc. B con el valor de la posición num2
 staa num2            Almacena el contenido del Acc. A en la posición num2
 stab num1            Almacena el contenido del Acc. B en la posición num1
num1 .byte $08        Define una posición de memoria como num1 y le asigna $08
num2 .byte $F3        Define una posición de memoria como num2 y le asigna $F3
 .end                 Finaliza el programa
```

9. **Sabiendo que el siguiente programa calcula el producto mediante sumas de 2 posiciones de memoria (*num1* y *num2*), modifíquelo de modo el cálculo se realice correctamente tanto valores positivos como negativos. ¿Cuál es el offset de la etiqueta *bucle*?**

```
inicio .org $00       Indica que el programa inicia en la dirección $0000
 clra                 Borra el Acc. A
 ldab num2            Carga el Acc. B con el valor de la posición num2
bucle adda num1       Suma al Acc. A el contenido de la posición num1
 decb                 Decrementa el contenido del Acc. B
 cmpb #$00            Compara el contenido del Acc. B con el valor $00
 bgt bucle            Salta a la etiqueta bucle si el Acc. B es mayor que $00
 staa $30             Almacena el contenido del Acc. A en la posición $0030
num1 .byte $04        Define una posición de memoria como num1 y le asigna $04
num2 .byte $03        Define una posición de memoria como num2 y le asigna $03
 .end                 Finaliza el programa
```

10. **Tomando como referencia el programa del ejercicio 9, escriba otro que permita calcular el cociente (mediante restas sucesivas) de 2 posiciones de memoria. Tenga en cuenta que el cálculo no podrá realizarse si el divisor es cero.**

11. **Dado el siguiente diagrama de flujo escriba el programa correspondiente. ¿Cuál es el propósito del programa? Considere que *n1*, *n2* y *res* corresponden a posiciones de memoria.**



12. Utilizando el emulador de MP6800 (SDK6800 V1.08) compruebe el comportamiento de los siguientes programas, determine su objetivo y observe los cambios en el CCR conforme se ejecutan las operaciones. Puede modificar los valores de las posiciones num, num1, num2, num3, valor o total para evaluar los programas con distintos conjuntos de datos.

Bit7	Bit6	H	I	N	Z	V	C
1	1	0	0	0	1	0	0

```
;programa ejemplo1
inicio .org 00
 ldaa num1
 adda num2
 adda num3
 staa valor
num1 .byte $04
num2 .byte $03
num3 .byte $FE
valor .byte $00
 .end
```

```
;programa ejemplo2
inicio .org 00
 ldaa num1
 clc
 rola
 staa valor
num1 .byte $03
valor .byte $00
 .end
```

```
;programa ejemplo3
inicio .org 00
 clra
 ldab num2
bucle adda num1
 decb
 cmpb #$00
 bgt bucle
 staa valor
num1 .byte $03
num2 .byte $04
valor .byte $00
 .end
```

```
;programa ejemplo4
inicio .org 00
  ldaa num1
  cmpa num2
  bne salto1
  cmpa num3
  bne salto1
  ldaa #$AA
  bra fin
salto1 ldaa #$FF
fin staa valor
num1 .byte $03
num2 .byte $03
num3 .byte $03
valor .byte $00
 .end
```

```
;programa ejemplo5
inicio .org 00
  ldaa num1
  cmpa num2
  bne salto1
  bra fin
salto1 cmpa num2
  ble fin
  ldab num2
  staa num2
  stab num1
num1 .byte $07
num2 .byte $03
fin .end
```

```
;programa ejemplo6
inicio .org 00
  clra
  ldx #num
bucle ldab $00,x
  cmpb #$FF
  beq fin
  adda $00,x
  inx
  bra bucle
fin staa total
num .byte $07,$02,$05,$04,$ff
total .byte $00
 .end
```