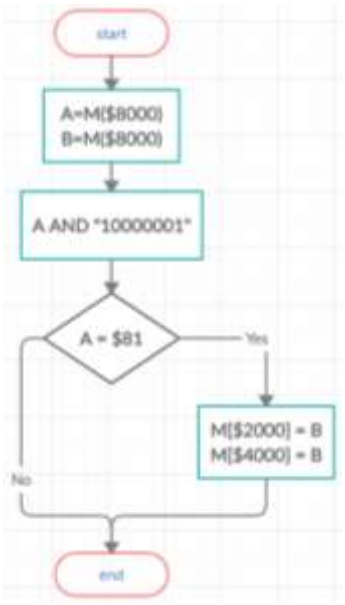
A. ESTRUCTURAS CONDICIONALES

Las estructuras condicionales de un problema implican la evaluación del cumplimiento de una condición, y la decisión de la alternativa a seguir de acuerdo con el resultado de dicha evaluación. Para codificar este tipo de estructuras usando el set de instrucciones del M6800 hay que trasladar la evaluación, decisión y bifurcación al uso de instrucciones de comparación combinadas con instrucciones de salto.

Ejemplo 1: Realice el diagrama de flujo y codificación mnemotécnica de un programa para M6800 que permita copiar el contenido de la posición $8000 a las direcciones $2000 y $4000, siempre y cuando los bits más significativo (b7) y menos significativo (b0) de la palabra a copiar sean 1. Indique el valor del PC de cada instrucción suponiendo que el programa iniciará en la dirección $6000. Calcule el valor del offset correspondiente a las instrucciones de salto.



PC		PROGRAMA	DESCRIPCION
$6000		LDAA $8000	A ← M[$8000]
$6003		LDAB $8000	B ← M[$8000]
$6006		ANDA #%10000001	A ← A AND 10000001 (bit a bit)
$6008		CMPA #$81	Compara A con 10000001
$600A		BNE fin	Salta si no es 0 (A <> 10000001)
$600C		STAB $2000	M[$2000] ← B
$600F		STAB $4000	M[$4000] ← B
$6012	fin	SWI	Fin

offset = PC_{final} – (PC_{actual} + 2)
offset = $6012 – ($600A + 2) = $6012 – $600C = $06 = fin

La operación AND permite realizar el producto lógico (bit a bit) entre el registro acumulador A o B y un valor específico o el contenido de una posición de memoria. Por ejemplo, siguiendo la lógica del programa:

Si M[$8000]=$88 = %10001000, entonces A = $88, y ANDA $81 realiza lo siguiente:

A =	1	0	0	0	1	0	0	0
ANDA#%	1	0	0	0	0	0	0	1
A =	1	0	0	0	0	0	0	0

Luego, A <> $81

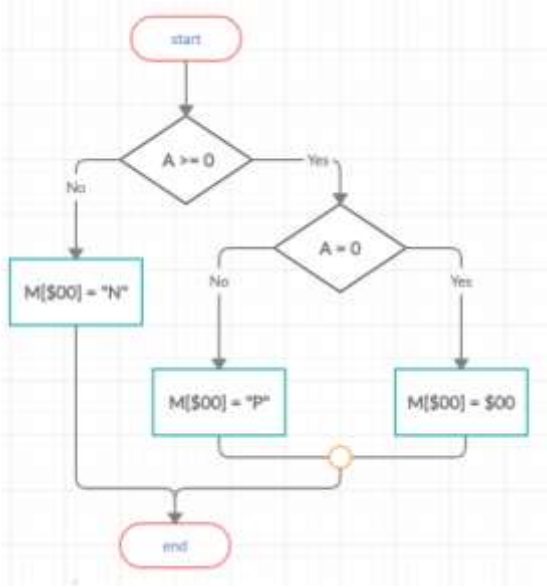
Si M[$8000]=$A7 = %10100111, entonces A = $A7, y ANDA $81 realiza lo siguiente:

A	1	0	1	0	0	1	1	1
ANDA#%	1	0	0	0	0	0	0	1
A =	1	0	0	0	0	0	0	1

Luego, A = $81

Luego de ejecutar la instrucción AND, el contenido del acumulador A queda alterado, por esta razón también se guardó el contenido de $8000 en el acumular B para realizar la copia.

Ejemplo 2: Realice el diagrama de flujo y codificación mnemotécnica que según el contenido del acumulador A guarde un valor específico en la primera posición de memoria: $00 si A es nulo, "P" (código ASCII de 'P') si A es positivo o "N" (código ASCII de 'N') si A es negativo. Utilice etiquetas en las instrucciones de salto.



	PROGRAMA	DESCRIPCION
	CMPA #$00	Compara A con 0
	BGE salto1	Salta si A >= 0
	LDAB #%01001110	B ← "N"
	BRA fin	Salta a fin
salto1	CMPA #$00	Compara A con 0
	BEQ salto2	Salta si A = 0
	LDAB #%01010000	B ← "P"
	BRA fin	Salta a fin
salto2	LDAB #$00	B ← $00
fin	STAB $00	M[$0000] ← $00
	SWI	Fin

Para determinar el valor que se almacenará en la dirección inicial de memoria debe evaluarse el contenido del acumulador, en este caso, comparándolo con $00 (cero). Luego, el acumulador B se carga con el valor correspondiente y se guardará en la dirección $0000. Debe observarse que esta operación no puede realizarse directamente sobre la posición de memoria sino que el acumulador B se utiliza como auxiliar.

Respecto a las instrucciones de bifurcación, BRA se utiliza para realizar saltos incondicionales, es decir, bifurcar sin evaluar ninguna condición. En el ejemplo, BRA permite evitar la ejecución de las instrucciones que no correspondan al caso evaluado.

B. RECORRIDO DE SEGMENTOS DE MEMORIA, ESTRUCTURAS REPETITIVAS

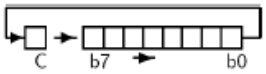
En muchas ocasiones, el planteo de la solución de un problema implica recorrer segmentos de memoria. En algunos casos para contar la ocurrencia de un evento analizando las palabras, otras, para modificar los datos presentes en ese segmento, otras veces simplemente para leer los datos de esa porción de memoria. Para realizar tales recorridos se utiliza el registro índice del M6800 (IX o IDX), en algunos casos también se puede hacer uso del puntero de pila (SP). Ambos son punteros a posiciones de memoria principal.

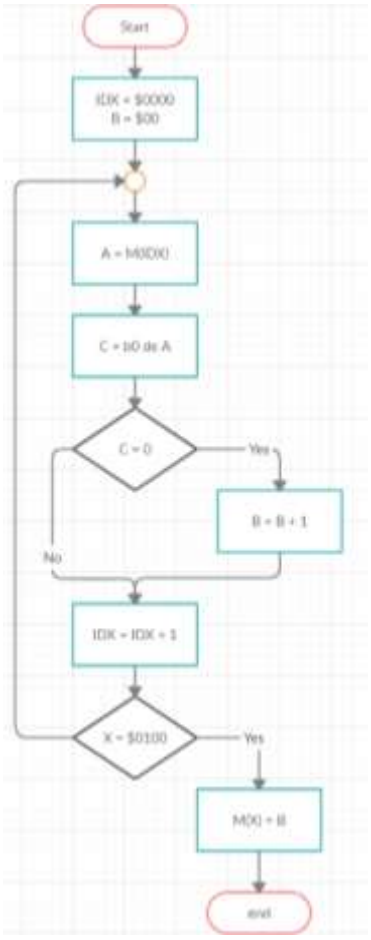
Al plantear el recorrido se pueden presentar distintas situaciones:

- que se conozcan las direcciones de inicio y fin del recorrido.
- que la dirección de un extremo del segmento, o ambos, estén almacenados en posiciones de memoria principal.
- que se conozca la dirección de inicio o de fin del recorrido (con las variantes de los dos casos anteriores) y no se conozca la dirección del otro extremo, pero sí la cantidad de elementos que se deben recorrer
- Como variante del caso anterior, también puede suceder que la cantidad de elementos a recorrer no fuera un dato dado, sino que se conoce la dirección de memoria donde se almacena la cantidad de elementos a recorrer.
- Por último, puede ocurrir que para iniciar y/o finalizar el recorrido, se espera la ocurrencia de determinada condición.

Ejemplo 3: Realice el diagrama de flujo y codificación mnemotécnica de un programa para M6800 que cuente los números pares presentes en la primera página de memoria, registrando la cuenta en la posición $0100. El segmento sólo contiene valores positivos y negativos. Indique el valor del PC de cada instrucción suponiendo que el programa inicia en $0200. Calcule *offset* de las instrucciones de salto.

Planteo: Para recorrer el segmento de memoria comprendido entre las direcciones $0000 y $00FF (primera página de memoria) se utilizará el registro índice. Además, se usará el acumulador B como contador de los números pares mientras que el acumulador A permitirá analizar cada palabra del segmento. Teniendo en cuenta que los números pares (en binario) terminan en 0 podrían, entre otras, utilizarse las instrucciones de rotación para aislar el bit menos significativo (b0) de cada elemento del segmento. Aplicando la instrucción RORA será posible desplazar los bits de la palabra en cuestión un bit a la derecha, trasladando el bit b0 al bit C del registro CCR. Luego, al analizar dicho bit será posible determinar si el valor es par o impar.





PC		PROGRAMA	DESCRIPCIÓN
$0200		LDX #$0000	IDX ← $0000
$0203		CLRB	B ← $00
$0204	S2	LDAA $00,X	A ←M(IDX)
$0206		RORA	Rotar A a derecha (C←b_0)
$0207		BCS S1	Salta si C = 1
$0209		INCB	B ← B + 1
$020A	S1	INX	IDX ← IDX +1
$020B		CPX #$0100	Compara IDX con $0100
$020E		BLT S2	Salta si IDX-$0100<0
$0210		STAB $00,X	M(IDX) ← B
$0212		SWI	Fin

OffsetS1 = $020A – ($0207 + 2) = $020A – $0209 = $01 = S1
OffsetS2 = $0204 – ($020E + 2) = $0204 – $0210 = - $0C => $F4= S2

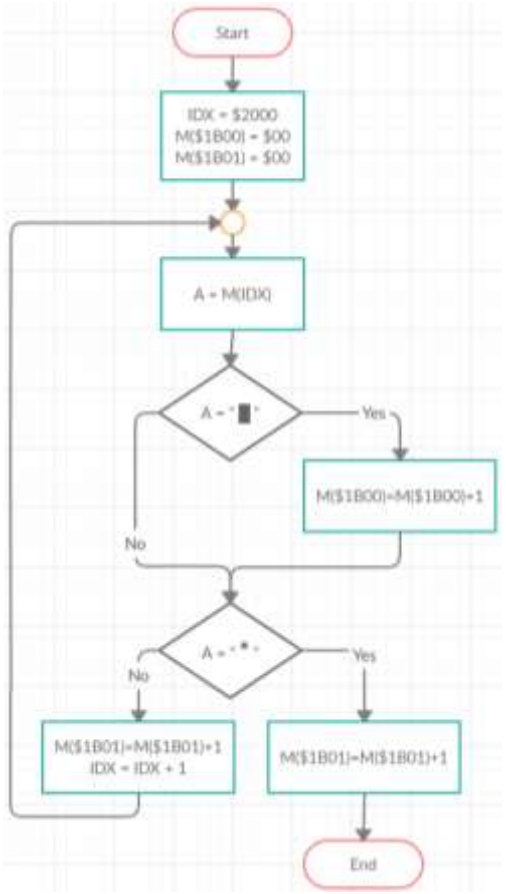
Ejemplo 4: Realice el diagrama de flujo y codificación mnemotécnica de un programa para M6800 que cuente las apariciones del carácter [redacted] en el segmento de memoria cuya dirección inicial es $2000 y cuyo último elemento es el carácter *. Considere que la cantidad de ocurrencias del carácter [redacted] se almacenará en la posición $1B00, mientras que la longitud del segmento recorrido se guardará en $1B01. Indique el valor de PC de cada instrucción (PC= $0000) y calcule el *offset* de las instrucciones de salto.

Planteo: Teniendo en cuenta que se debe recorrer un segmento de memoria a partir de la dirección $2000, entonces se usará el registro índice para acceder sucesivamente a cada posición, comparándose cada palabra leída con el símbolo "*" para determinar la finalización del recorrido. A fin de contar el carácter buscado o detectar el final del segmento se utilizarán los códigos ASCII correspondientes a estos símbolos:

[redacted] = 11011011 = $DB * = 00101010 = $2A

Sabiendo que los registros de memoria pueden incrementarse directamente mediante la instrucción INC entonces $1B00 $1B01 se accederán directamente para contar el carácter [redacted] y la cantidad de elementos del segmento, respectivamente; ambas posiciones se inicializarán en $00.

Para comparar los ASCII de los símbolos con el contenido de los registros será necesario cargar estos valores en alguno de los registros acumuladores del M6800 (A ó B)



PC		PROGRAMA	DESCRIPCIÓN
$0000		LDX #$2000	IDX ← $2000
$0003		CLR $1B00	M($1B00) ← $00
$0006		CLR $1B01	M($1B01) ← $00
$0009	vuelve	LDAA $00,X	A←M(IDX)
$000B		CMPA #$DB	Compara A con [redacted]
$000D		BNE S1	Salta si son <>
$000F		INC $1B00	M($1B00) ← M($1B00) + 1
$0012	S1	CMPA #$2A	Compara A con *
$0014		BEQ fin	Salta si son =
$0016		INC $1B01	M($1B01) ← M($1B01) + 1
$0019		INX	IDX ← IDX + 1
$001A		BRA vuelve	Itera
$001C	fin	INC $1B01	M($1B01) ← M($1B01) + 1
$001F		SWI	Fin

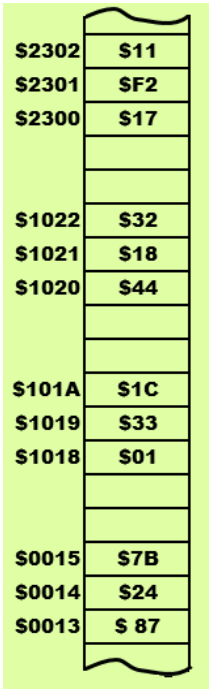
OffsetS1 = $0012 – ($000D + 2) = $0012 – $000F = $03 = S1
Offsetvuelve = $0009 – ($001A + 2) = $0009 – $001C = - $13 => $ED = vuelve
Offsetfin = $001C – ($0014 + 2) = $001C – $0016 = $06 = fin

El último incremento del contador de palabras recorridas tiene en cuenta el carácter * que no fue contado durante el recorrido ya que éste lo interrumpe al ser encontrado.

PROBLEMAS A RESOLVER

1. **Responder:**
   a) Identifique en el set de instrucciones del M6800 instrucciones que realicen comparaciones.
   b) Identifique en el set de instrucciones del M6800 instrucciones de salto y bifurcación.
   c) ¿Qué acción realizan las instrucciones BRA y JMP? ¿Qué tienen en común? ¿En qué se diferencian?
   d) Si tuviera que modificar el valor del PC ¿cómo lo haría?

2. **Dado el siguiente segmento de memoria y las instrucciones listadas a continuación determine el correspondiente valor del registro índice (IDX)**



INSTRUCCIÓN	IDX Actual	IDX Obtenido
LDX #$2301	$4401	
LDX $1021	$5287	
LDX $14	$3309	
LDX $09,X	$1010	

3. Codifique los programas para el MP6800 que realicen lo siguiente:

a) Un programa que reemplace la segunda página de memoria por el valor $CC.

b) Un programa que reemplace por el valor $AA la porción de memoria que inicia en $0100 y contiene 256 palabras.

c) Un programa que recorra la MP desde la dirección $0020 grabando el dato $23; hasta hallar el valor $FF.

d) Un programa que blanquee la porción de memoria cuyo inicio y fin se encuentran en las posiciones $00F0 y $00F2.

4. Copie los siguientes códigos de programas en el emulador del MP6800 (SDK6800), estudie sus comportamientos y determine sus objetivos. Puede modificar algunos valores (#$23 en Ej4a, #$02 en Ej4b) para un mejor análisis. Calcule los valores de offset, reemplácelos en el código y verifique si son correctos. Luego realice lo siguiente:

a) Teniendo en cuenta el programa Ej4a, codifique otro que cuente los números pares e impares y guarde ambas cuentas en dos posiciones contiguas de memoria.

b) Teniendo en cuenta el programa Ej4b, codifique otro programa que reemplace la multiplicación x 4.

```
;Ej4a .org $0000
;carga datos para prueba
 ldaa #$01
 ldx #$0040
VUELVE staa $00,x
 inca
 inx
 cpx #$006F
 ble VUELVE

;principal
 clrb
 ldx #$0040
BUCLE ldaa $00,x
 rora
 bcc ESPAR
 incb
ESPAR inx
 cpx #$006F
 bne BUCLE
 inx
 stab $00,x
 .end
```

```
;Ej4b .org $0000
;carga datos para prueba
 ldaa #$02
 ldx #$0060
VUELVE staa $00,x
 inx
 cpx #$007F
 ble VUELVE

;principal
 ldx #$0060
ETIQ1 ldaa $00,x
 asla
 staa $00,x
 inx
 cpx #$007F
 ble ETIQ1
 .end
```

5. **Copie los siguientes códigos de programas en el emulador del MP6800 (SDK6800), estudie sus comportamientos y determine el propósito de los mismos. Puede modificar algunos valores (en Ej5a reemplace #$08 por #$04 y #$03 por ejemplo) para una mejor comprensión. Calcule los valores de offset, reemplácelos en el código y verifique si son correctos.**

```
;Ej5a inicio .org $0000
;carga datos para prueba
 ldaa #$16
 staa $8f
 ldx #$0090
 ldaa #$08
XX staa $00,x
 inca
 inx
 cpx #$00a6
 ble XX

;principal
 clrb
 ldx #$0090
VUELVE ldaa $00,x
 oraa #%11111100
 cmpa #$fc
 bne NOES
 incb
NOES inx
 dec $8f
 bne VUELVE
 stab $00,x
 .end
```

```
;Ej5bNibbleHigh  .org
$0000
;inicio .org $0000
;carga datos para prueba
 jsr $6f
 ldaa #$f8
 ldx $0070
ETQ1 staa $00,x
 inx
 cpx $0072
 ble ETQ1

;principal
 ldx $0070
VUELVE ldaa $00,x
 anda %00001111
 staa $00,x
 inx
 cpx $0072
 ble VUELVE
 bra FIN
 .org $006f
 rts
 .byte $00,$80,$00,$cf
FIN .end
```

Luego, responda:

a) ¿Qué recorrido propone el Ej5a? ¿Cuántas palabras analiza? ¿Qué cambiaría para que analice otra cantidad de palabras?

b) ¿Qué cambiaría para que cuente los múltiples de 8?

c) ¿Cuál es el recorrido de memoria que propone Ej5bnibbleHigh?

d) Este programa depura la mitad superior de las palabras que encuentra en su recorrido. ¿Es correcta esta afirmación? Si no es así, ¿qué cambiaría para que se cumpla?

6. **Después de 4 días y noches de trabajo, un estudiante de informática logró diseñar un programa que permite generar el código BCH Johnson. Desafortunadamente, éste desarrollador novicio derramó mate sobre la hoja borrador del código y hoy debe presentarlo.**

¿Serás capaz de completar las instrucciones y/u operandos necesarios para ayudar a este programador descuidado? Para ello, debes considerar que no está permitido usar etiquetas para los saltos y que el programa debe comenzar en la dirección $0010.

7. **Realice el diagrama de flujo y codificación mnemotécnica de un programa que complete la segunda página de memoria principal con valores al azar. No existe una instrucción que brinde tal funcionalidad, por lo que Ud. deberá imaginar cómo producir el valor a guardar.**

8. **Codifique un programa que cuente los números positivos, negativos y ceros que se encuentren en la segunda página de memoria principal. Los valores de cuenta deben almacenarse en las 3 posiciones de memoria anteriores al inicio del segmento, respectivamente. Consigne el PC para cada instrucción, considerando que el programa se inicia en la dirección $0000.**

9. **Codifique un programa que reemplace los números negativos del segmento de memoria comprendido entre las posiciones $0090 a $019F por sus correspondientes imágenes positivas. Registre la cantidad de reemplazos realizados en la posición $01A0.**

10. **Realice el diagrama de flujo y codificación mnemotécnica de un programa que determine la longitud (cantidad de valores almacenados) de un segmento de memoria que inicia en la posición $FB00. Para ello, considere que el segmento almacena caracteres ASCII y que la última posición de éste contiene el símbolo ")". Consigne el valor del PC para cada instrucción, considerando que el programa se inicia en la dirección $0000, y calcule los valores de offset correspondientes para las instrucciones de salto.**

11. **Codifique un programa para MP6800 que, dada una serie numérica, determine cuántos valores nulos contiene. Suponga que el segmento de memoria inicia en la dirección $00C0 y que la cantidad de elementos de éste se almacenada en la posición $00BF. Considere que el resultado del conteo se registrará en la posición $00B0. Utilice etiquetas para las instrucciones de salto.**

12. **Codifique un programa que, según el valor de la última posición de la primera página de memoria, genere el código BCO Johnson ó BCH Johnson a partir de la dirección $0100. Considere que si la posición especificada contiene el valor 8_{10} debe generarse el BCO Johnson, mientras que si contiene el valor 16_{10} debe generarse el BCH Johnson.**

13. **Realice el diagrama de flujo y codificación mnemotécnica de un programa que determine el máximo valor almacenado en un segmento de memoria que inicia en dirección $0050 y cuya longitud es de 30_{10} elementos.**

14. **Dado un segmento de memoria que almacena caracteres (códigos ASCII), codifique un programa que convierta las letras minúsculas a mayúsculas y viceversa. Para ello tenga en cuenta que el rango de las mayúsculas es $41-$5A ("A"- "Z"), mientras que el de las minúsculas es $61-$7A ("a"-"z"). Además, suponga que el segmento inicia en la dirección $1000, indicándose la cantidad de posiciones a considerar en la dirección $0100. Cualquier otro dato encontrado no debe ser modificado.**

15. **Codifique un programa que, dado un segmento de memoria, determine cuantas veces se repite valor almacenado en la posición $0330 en éste. Considere que la dirección inicial del segmento está almacenada en las posiciones $3C00 y $3C01, mientras que la dirección final ocupa las posiciones $3C02 y $3C03. Registre el resultado obtenido en la dirección $0331.**

16. **Codifique un programa que replique el contenido del segmento de memoria comprendido entre las direcciones $2A00 y $2AFF, a partir de la mitad del mapa de direcciones.**