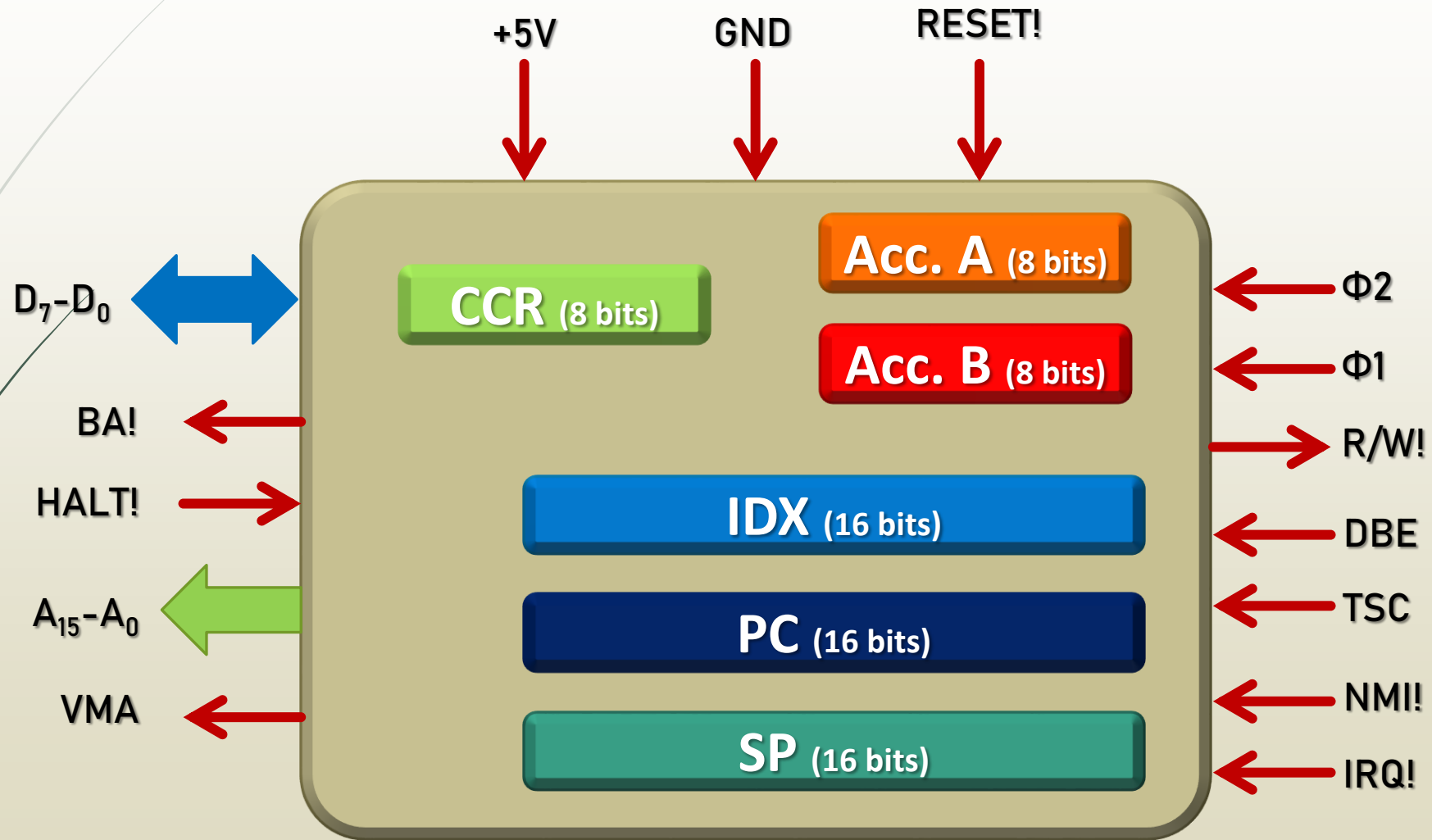




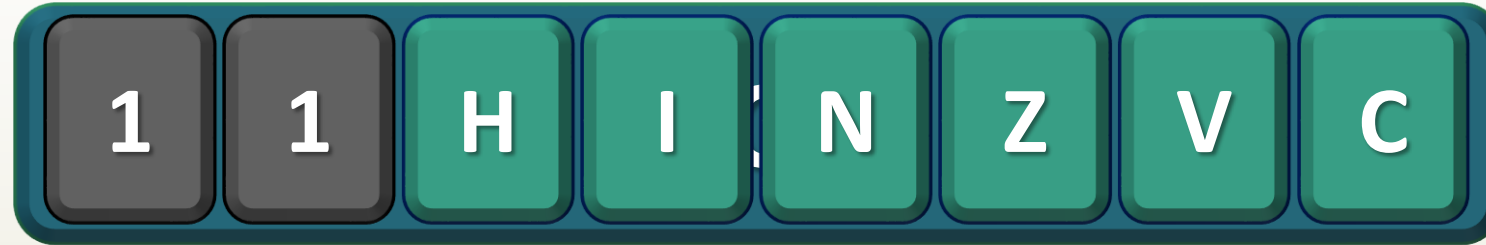
TÉCNICAS Y ESTRUCTURAS DIGITALES

Microprocesador Motorola 6800

Microprocesador 6800



Registro de Códigos de Condición (CCR)



C

Bit de Acarreo

V

Bit de Desborde

Z

Bit de resultado Cero

N

Bit de resultado Negativo

I

Bit de Interrupción

H

Bit Acarreo de medio byte

Modos de Direccionamiento

- El modo de direccionamiento indica la ubicación de los datos de una instrucción y cómo se accederá a ellos.
 - Implicado
 - Inmediato
 - Directo
 - Extendido
 - Indexado
 - relativo
- El formato general de instrucción es

OPCODE

OPERANDO

Modo Implicado

- Las instrucciones en modo implicado no necesitan datos de memoria para ejecutarse. Estas instrucciones son de 1 byte ya que sólo especifican el código de la instrucción.



- Por ejemplo: la instrucción INX incrementa en uno el contenido actual del registro índice (IDX).

INX $IDX \leftarrow IDX + 1$

...	
	\$0105
	\$0104
	\$0103
INX	\$0102
	\$0101
	\$0100
...	

Modo Inmediato

- Las instrucciones en modo inmediato se ejecutan usando el valor indicado como operando. Estas instrucciones pueden ser de 2 o 3 bytes según el tamaño del operando (1 o 2 bytes).



- Por ejemplo: la instrucción LDAA asigna al acumulador A el valor indicado como argumento.

LDAA # $\$23$ Acc A \leftarrow $\$23$

- Por ejemplo: la instrucción LDX asigna al registro índice el valor indicado como argumento.

LDX # $\$11FF$ IDX \leftarrow $\$11FF$

...	
FF	$\$0105$
11	$\$0104$
LDX	$\$0103$
	$\$0102$
23	$\$0101$
LDAA	$\$0100$
...	

Modo Directo

- Las instrucciones en modo directo se ejecutan usando el contenido de una posición de memoria. Estas instrucciones son de 2 bytes ya que usan direcciones de memoria “cortas”.

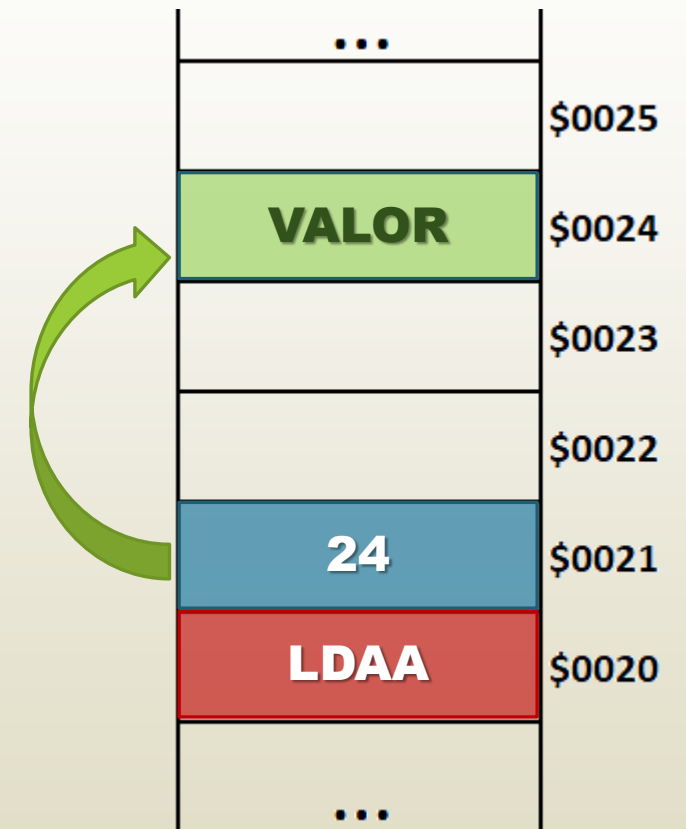
OPCODE

DIRECCIÓN

- Por ejemplo: la instrucción LDAA asigna al acumulador A el contenido de la dirección indicada como argumento.

LDAA \$24

Acc A ← M[\$0024]



Direcciones accesibles en modo directo: \$0000 – \$00FF (1era página)

Modo Extendido

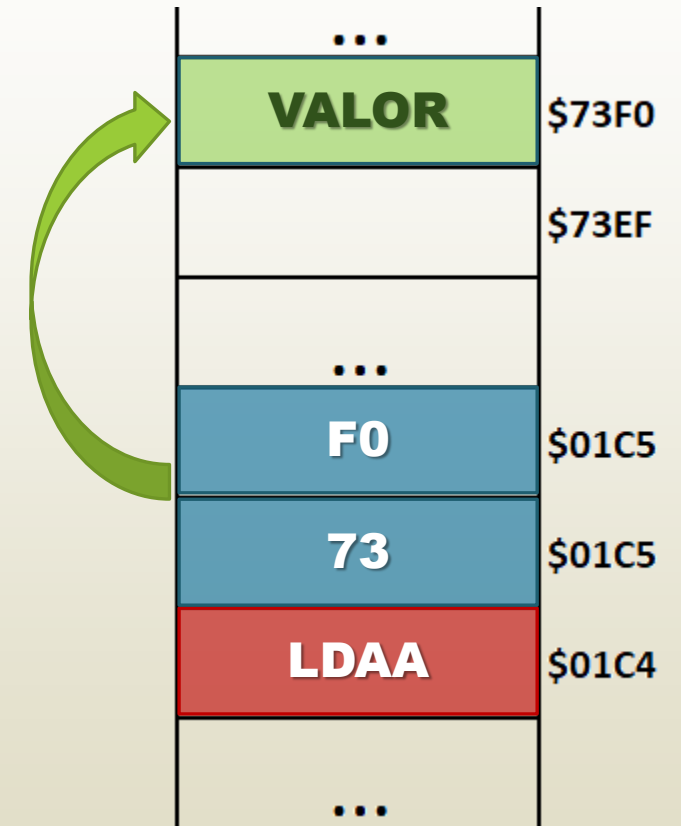
- Las instrucciones en modo extendido se ejecutan usando el contenido de una posición de memoria. Estas instrucciones son de 3 bytes.

OPCODE

DIRECCIÓN

- Por ejemplo: la instrucción LDAA asigna al acumulador A el contenido de la dirección indicada como argumento.

`LDAA $73F0 Acc A ← M[$73F0]`



Direcciones accesibles en modo extendido: \$0000 – \$FFFF (todo el mapa)

Modo Indexado

- Las instrucciones en modo indexado operan sobre una posición de memoria está dada por el valor del registro índice y un desplazamiento.

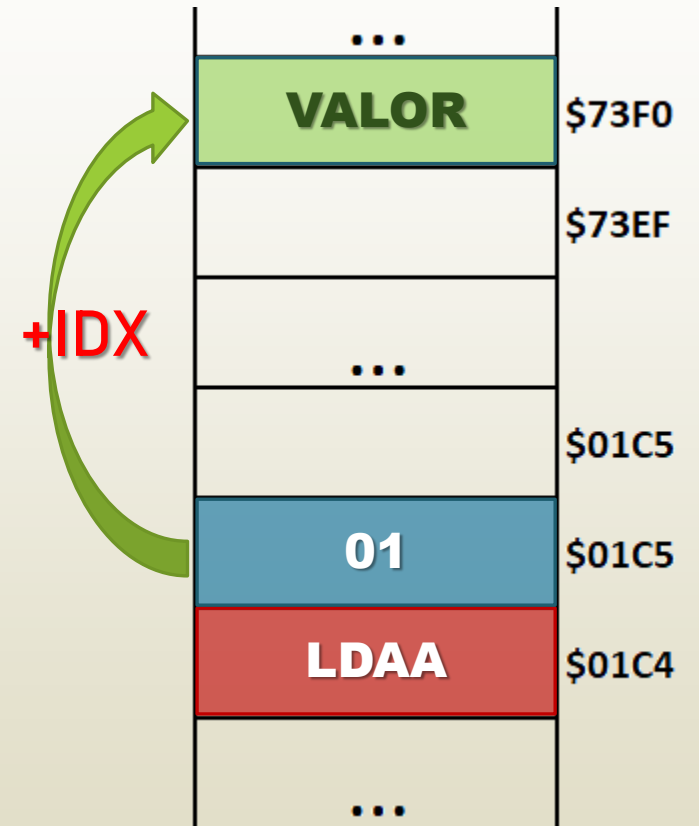
OPCODE

DESPLAZAMIENTO

- Por ejemplo: la instrucción LDAA asigna al acumulador A el contenido de la dirección apuntada por el registro índice más el desplazamiento especificado. Suponiendo $IDX = \$73EF$

LDAA \$01,X

Acc A \leftarrow M[IDX+\$01]



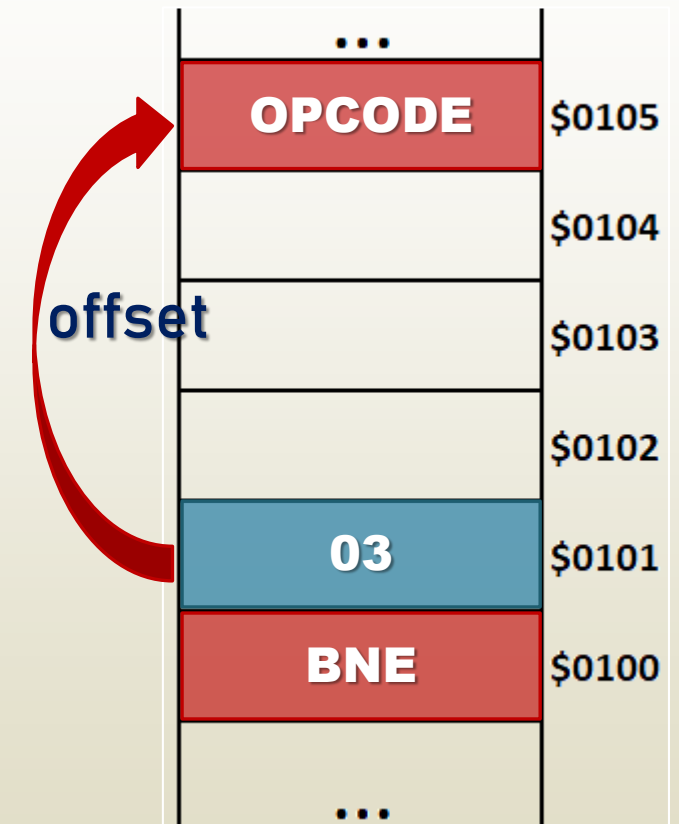
Modo Relativo (1)

- Las instrucciones en modo relativo permiten saltar de una parte del programa a otra. El offset que acompaña al OPCODE puede ser positivo o negativo. Las instrucciones son de 2 bytes.



- Por ejemplo: la instrucción BNE, si se cumple la condición que ésta verifica, saltará tantas posiciones como indique el offset.

BNE \$03



Modo Relativo (2)

- ¿Cómo se calcula el offset de las instrucciones de salto?

$$PC_{\text{final}} = PC_{\text{actual}} + 2 + \text{offset}$$

$$\text{offset} = PC_{\text{final}} - (PC_{\text{actual}} + 2)$$

SALTO POSITIVO (HACIA ADELANTE)

...
\$0100 BNE \$03
...
\$0105 instrucción
...

offset= \$0105 - (\$0100 + 2)
offset= \$0105 - \$0102
offset= \$03

SALTO NEGATIVO (HACIA ATRÁS)

...
\$0222 instrucción
...
\$022C BEQ \$F5
...

offset= \$0222 - (\$022C + 2)
offset= \$0222 - \$022E
offset= -B
offset= \$F5

0000 1011 = 0B
↓ ↓
1111 0101 = F5

Resumen de Instrucciones

- Set de Instrucciones del MP6800

INSTRUCCIONES DE ACUMULADOR Y MEMORIA																							
OPERACIONES DE ACUMULADOR Y MEMORIA	MNEM.	MODOS DE DIRECCIONAMIENTO															OPERACIÓN LÓGICA O ARITMÉTICA (los identificadores de reg. se refieren al contenido)	REGISTRO DE CÓDIGOS DE CONDICIÓN					
		INMEDIATO			DIRECTO			INDEXADO			EXTENDIDO			INHERENTE				H	I	N	Z	V	C
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#							
Sumar	ADDA	8B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Sumar acumuladores	ABA										1B	2	1				A + B → A	↕	•	↕	↕	↕	↕

MNEMOTÉCNICO
Nombre simbólico de la instrucción

MODOS DE DIRECCIONAMIENTO
OPCODE de cada instrucción según el modo, tamaño (#) y ciclos (~)

Descripción de la operación

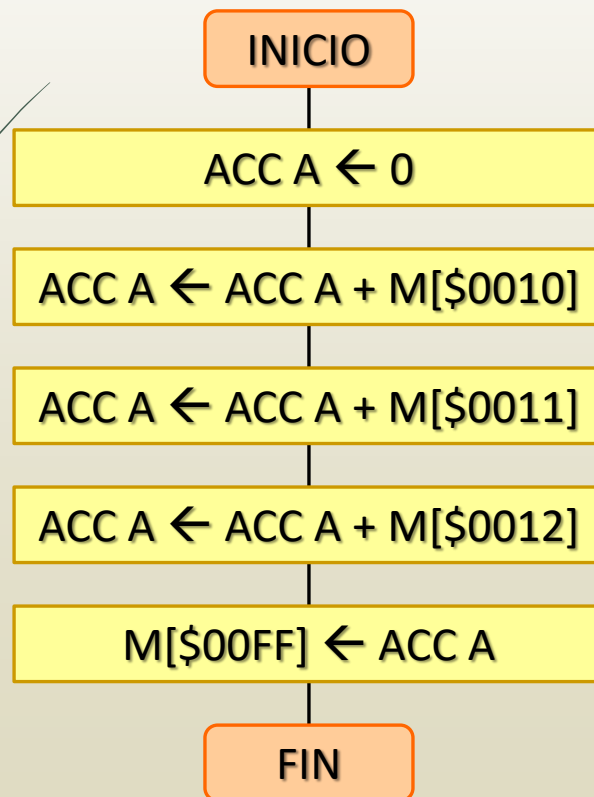
Bits del CCR modificados por la operación

tamaño en bytes de la instrucción

~ cantidad de ciclos que ocupa la ejecución de cada instrucción

Ejemplo 1. Diseño del programa

- Escriba un programa que sume el contenido de las posiciones de memoria \$0010, \$0011 y \$0012, guardando el resultado final en \$00FF. Además indique el *opcode*, tamaño y ciclos de cada instrucción y el valor del PC (PC inicial \$0100).



OPCODE	~	#	PC	
4F	2	1	\$0100	CLRA
9B	3	2	\$0101	ADDA \$10
9B	3	2	\$0103	ADDA \$11
9B	3	2	\$0105	ADDA \$12
B7	5	3	\$0107	STAA \$00FF
3F	12	1	\$010A	SWI

Ejemplo 1. Emulador

PROGRAMA

SDK6800 Emulator v1.08 (www.HVRSoftware.com)

CONTENIDO DE LA MEMORIA

REGISTROS DEL PROCESADOR

Assembly Program

```
0001 ; ejemplo1
0002 inicio .org $0000
0003 jsr $000F
0004
0005 clra
0006 adda $10
0007 adda $11
0008 adda $12
0009 staa $00FF
0010
0011 jmp fin
0012
0013 .org $000F
0014 rts
0015 .byte $06,$FD,$04
0016
0017 fin .end
```

Memory	Display	Reference
0000:	BD 00 0F 4F 9B 10 9B 11 9B 12 97 FF 7E 00 13 39	
0010:	06 FD 04 00 00 00 00 00 00 00 00 00 00 00 00	
0020:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
0030:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
0040:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
0050:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
0060:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
0070:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
0080:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
0090:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00A0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00B0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00D0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 07	
0100:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
0110:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
0120:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
0130:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

ADDRESS: 0013

PC: 0000 X 0000
SP: F000
IR: 007E JMP
ACCUMULATOR
A: 00 B: 00

Status Flags
H I N Z V C
0 0 0 0 0 0

Base Converter
7 8 9 F
4 5 6 E
1 2 3 D
0 A B C
Hex Dec Bin

Assembling program...
Syntax Check: OK

ABRIR PRG **EJECUCIÓN PASO A PASO** Break Disabled Break At: 0000

Clear Load Save Step Run Stop

GUARDAR PRG **EJECUCIÓN** **DETENER / REINICIAR**

Ejemplo 1. Programa

```
; ejemplo1      comentario (;), nombre del programa
inicio .org $0000  dirección inicial del programa ($0000)
jsr $000F      datos del programa (guardado a partir de $000F)

clra          borrado del acumulador A
adda $10      suma al acc. A el contenido de $0010
adda $11      suma al acc. A el contenido de $0011
adda $12      suma al acc. A el contenido de $0012
staa $00FF    guarda el valor del acc. A en $00FF

jmp fin       salta a la etiqueta fin (.end indica final del prg)

.org $000F    rutina de carga de datos (dir. inicial $000F)
rts          retorno al programa principal (rts ocupa $000F)
.byte $06,$FD,$04  datos almacenados (a partir de $0010)

fin .end      fin del programa
```

Ejemplo 2. Diseño del programa

- Escriba un programa que sume los valores \$03, \$07 y \$F2, guardando el resultado final en \$0010. Además indique el tamaño de cada instrucción y el valor del PC (PC inicial \$3401).



Ejemplo 2. Emulador

SDK6800 Emulator v1.08 (www.HVRSoftware.com)

PROGRAMA EN MEMORIA

ADDRESS: 0013

Assembly Program

```
0001 ; ejemplo2
0002 inicio .org $0000
0003
0004 clra
0005 adda #$03
0006 adda #$07
0007 adda #$F2
0008 staa $FF
0009
0010 fin .end
```

Memory | **Display** | **Reference**

0000	4F	8B	03	8B	07	8B	F2	97	FF	00	00	00	00	00	00	00
0010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

PC 0000 X 0000
SP F000
IR 007E JMP
ACCUMULATOR
A 00 B 00

Status Flags
0 0 0 0 0 0
H I N Z V C

Base Converter
7 8 9 F
4 5 6 E
1 2 3 D
0 A B C
Hex Dec Bin

Assembling program...
Syntax Check: OK

Break Disabled
Break At: 0000

Clear Load Save Step Run Stop

RESULTADO FINAL

Ejemplo 2. Programa

```
; ejemplo2          comentario (;), nombre del programa
inicio .org $0000  dirección inicial del programa ($0000)

clra                borrado del acumulador A
adda #$03           suma al acc. A el valor $03
adda #$07           suma al acc. A el valor $07
adda #$F2           suma al acc. A el valor $F2
staa $FF           guarda el valor del acc. A en $00FF

.end                fin del programa
```

Ejemplo 3. Diseño del programa

- Escriba un programa que sume el contenido de las posiciones de memoria \$0100, \$0101, \$0102 y \$0103, guardando el resultado final en \$FF00. Utilice el modo indexado para leer los datos a sumar. Indique el valor del PC (PC=\$0220).



Ejemplo 3. Emulador

SDK6800 Emulator v1.08 (www.HVRSoftware.com)

Assembly Program

```
0001 ; ejemplo3
0002 inicio .org $0000
0003 jsr $00ff
0004
0005 clra
0006 ldx #$0100
0007 ldaa $00,x
0008 adda $01,x
0009 adda $02,x
0010 adda $03,x
0011 staa $ff00
0012
0013 jmp fin
0014
0015 .org $00ff
0016 rts
0017 .byte $01,$FC,$04,$06
0018
0019 fin .end
```

Memory | **Display** | **Reference** | ADDRESS: FF00

FEE0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
FEF0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
FF00:	07	00	00	00	00	00	00	00	00	00	00	00	00	00	00
FF10:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
FF20:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
FF30:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
FF40:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
FF50:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
FF60:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
FF70:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
FF80:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
FF90:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
FFA0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
FFB0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
FFC0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
FFD0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
FFE0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
FFF0:	00	00	00	00	00	00	00	00	00	00	00	00	00	19	14

PC 0000 X 0000
SP F000
IR 00BD JSR
ACCUMULATOR
A 00 B 00

Status Flags
0 0 0 0 0 0
H I N Z V C

Base Converter

7	8	9	F
4	5	6	E
1	2	3	D
0	A	B	C

Hex Dec Bin

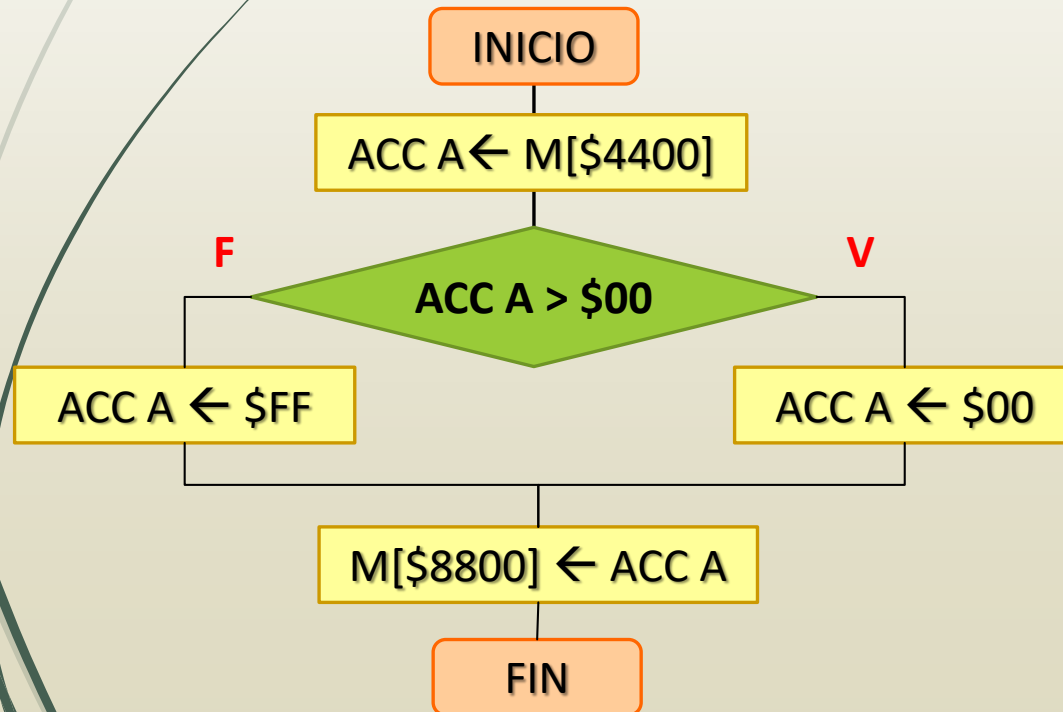
Assembling program...
Syntax Check: OK

Break Disabled
Break At: 0000

Clear Load Save Step Run Stop

Ejemplo 4. Diseño del Programa

- Escriba un programa que determine si valor contenido en la dirección de memoria \$4400 es positivo o negativo, suponiendo que nunca será cero. El resultado de la comparación se guardará en \$8800: \$00 positivos, \$FF para negativos. Indique el valor del PC de cada instrucción (PC inicial \$1000). Utilice etiquetas para identificar los saltos.



PC		
\$1000		LDAA \$4400
\$1003		CMPA #00
\$1005		BLT NEGATIVO
\$1007		CLRA
\$1008		BRA GUARDAR
\$100A	NEGATIVO	LDAA #FF
\$100C	GUARDAR	STAA \$8800
\$100F		SWI

Ejemplo 4. Emulador

SDK6800 Emulator v1.08 (www.HVRSoftware.com)

Assembly Program

```
0001 ; ejemplo4
0002 inicio .org $0000
0003 jsr $43ff
0004
0005 ldaa $4400
0006 cmpa #$00
0007 blt negativo
0008 clra
0009 bra guardar
0010 negativo ldaa #$ff
0011 guardar staa $8800
0012
0013 jmp fin
0014
0015 .org $43ff
0016 rts
0017 .byte $13
0018
0019 fin .end
```

Memory | Display | Reference | ADDRESS: 4401

8760:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8770:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8780:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8790:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
87A0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
87B0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
87C0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
87D0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
87E0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
87F0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8800:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8810:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8820:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8830:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8840:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8850:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8860:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8870:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8880:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8890:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

PC 0000 X 0000
SP F000
IR 007E JMP
ACCUMULATOR
A 00 B 00

Status Flags
0 0 0 0 0 0
H I N Z V C

Base Converter
7 8 9 F
4 5 6 E
1 2 3 D
0 A B C
Hex Dec Bin

Assembling program...
Syntax Check: OK

Break Disabled
Break At: 0000

Clear Load Save Step Run Stop

Ejemplo 4. Cálculo de offset

- Calcule el valor de offset correspondiente a las etiquetas NEGATIVO y FIN del programa anterior.

PC		
\$1000		LDAA \$4400
\$1003		CMPA # \$00
\$1005		BLT \$03
\$1007		CLRA
\$1008		BRA \$02
\$100A	NEGATIVO	LDAA # \$FF
\$ 100C	FIN	STAA \$8800
\$100F		SWI

Offset de NEGATIVO

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

$$\text{Offset} = \$100A - (\$1005 + 2)$$

$$\text{Offset} = \$100A - \$1007$$

$$\text{Offset} = \$03$$

Offset de FIN

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

$$\text{Offset} = \$100C - (\$1008 + 2)$$

$$\text{Offset} = \$100C - \$100A$$

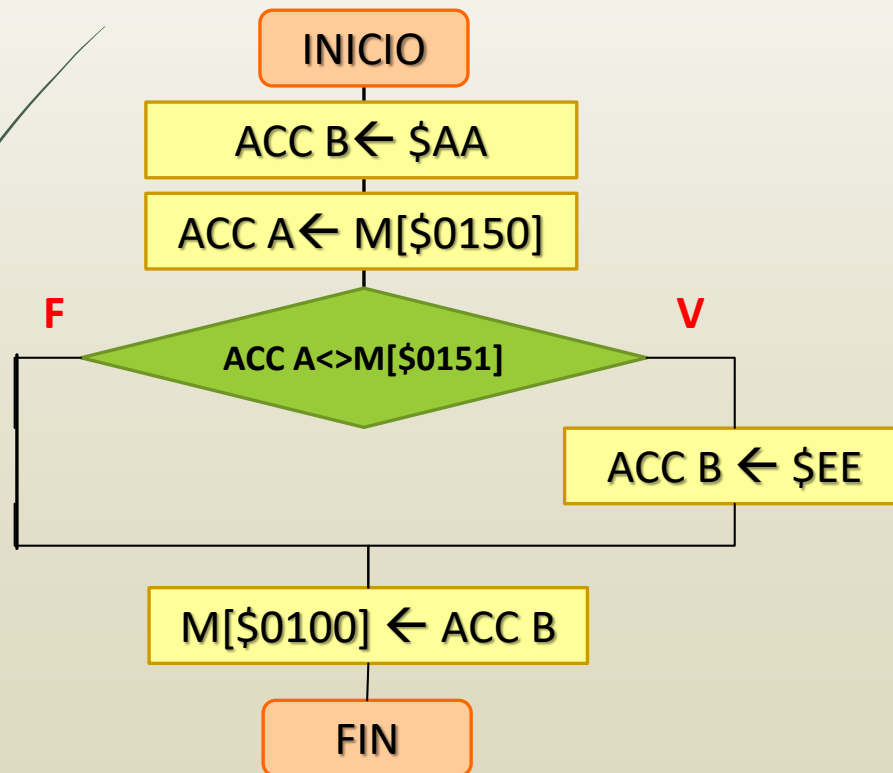
$$\text{Offset} = \$02$$

Problemas

- Utilizando el simulador SDK6800 resuelva los siguientes problemas:
 - Escriba un programa que determine si los valores de las posiciones de memoria \$0150 \$0151 son iguales o no. Considere que el resultado de la comparación se almacenará en \$0100, siendo \$AA para valores iguales y \$EE para valores distintos.
 - Modifique programa anterior de modo que si los valores comparados son distintos se identifique el mayor de ellos. Para ello, considere que el mayor valor deberá guardarse en la dirección \$0101.
 - Escriba un programa que, dados 2 valores de memoria, calcule el producto mediante sumas de éstos. Considere que los valores se almacenan en las direcciones \$0030 y \$0031, mientras que el resultado deberá guardarse en \$0040.
 - Escriba un programa que, dados 2 valores positivos de memoria, determine si éstos son múltiplos o no. Considere que el resultado del análisis se almacenará en \$0080 siendo \$01 para valores múltiplos y \$AA en caso contrario.

Ejemplo 5. Diseño del programa

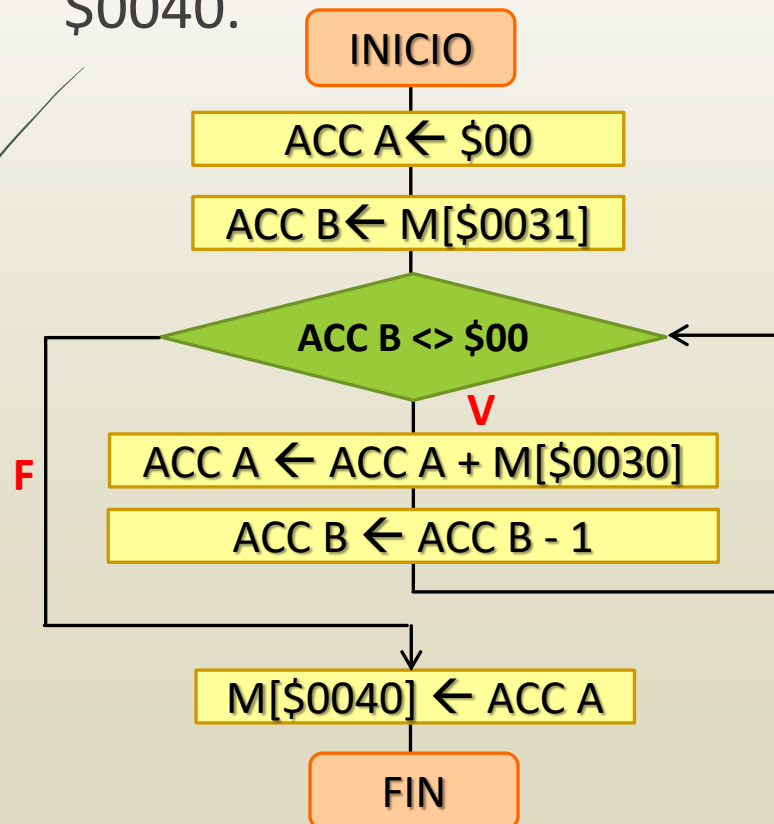
- Escriba un programa que determine si los valores de las posiciones de memoria \$0150 \$0151 son iguales o no. Considere que el resultado de la comparación se almacenará en \$0100, siendo \$AA para valores iguales y \$EE para valores distintos.



```
LDAB #$AA
LDAA $0150
CMPA $0151
BEQ GUARDAR
LDAB #$EE
GUARDAR STAB $0100
SWI
```

Ejemplo 6. Diseño del programa

- Escriba un programa que, dados 2 valores de memoria, calcule el producto mediante sumas de éstos. Considere que los valores se almacenan en las direcciones \$0030 y \$0031, mientras que el resultado deberá guardarse en \$0040.



CLRA
LDAB \$31
COMPARAR CMPB #\$00
BEQ GUARDAR
ADDA \$30
DECB
BRA COMPARAR
GUARDAR STAA \$40
SWI