

INTRODUCCIÓN

domingo, 4 de agosto de 2019 06:21 p. m.

► Estructura y Programación de Computadores

- Pedro Ignacio Rincón Gómez
- Email: pirg@unam.mx

► Temario

- 1.- Estructura de la maquina
- 2.- Presentación de un caso real
- 3.- Ensambladores y macroensambladores
- 4.- Encadenados y cargadores
- 5.- Asignación de memoria
- 6.- Programación de entrada - salida

► Evaluación

- 1^{er} proyecto - Compilador Básico - 40%
- 2^o proyecto - Programa en lenguaje ensamblador - 40%
- Evaluación - 20%

Bibliografía

- Manual técnico de MC68HC11
Prop. editorial de Motorola,

- Link
Dropbox } goo.gl/hHloil → Material / Ej PDC

// Apartheid

- Microcontrollers Technology: The 68HC11

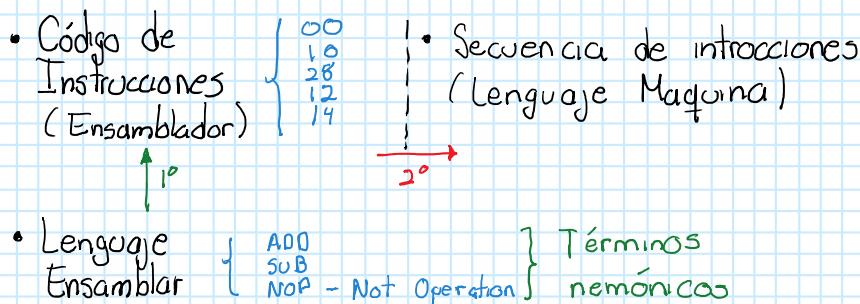
Peter Spasov

$$1024^2 = 1048576 \text{ bytes} \rightarrow \text{Megabytes}$$

1_Categorización

Tuesday, August 6, 2019 2:32 PM

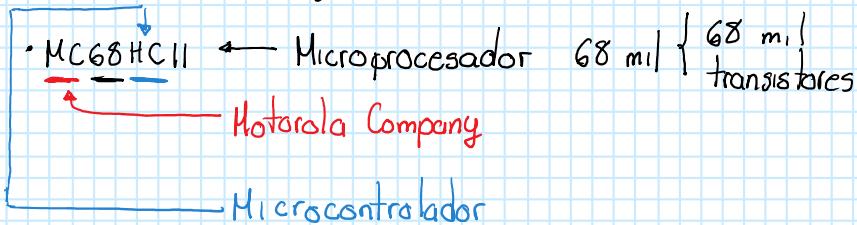
- CISC → Complex Instruction set Computer
 - ↳ Tiene un hardware complejo
 - ↳ Motorola MC68HC11
- RISC → Reduced Instruction set Computer
 - ↳ Más difícil de programar
 - ↳ Ejemplo: Programar algoritmo de multiplicación



► Introducción

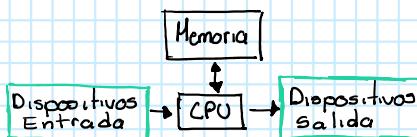
- El MC68HC11 es una familia de microcontroladores de propósito general, de 8 bits, manufacturada por Motorola.

- Tiene arquitectura CISC
- Es de tipo Von Neumann
- Es secuencial y acumulador



- Microprocesador - Es un (Unidad Central de Procesamiento) CPU contenido en un encapsulado (Chip - Circuito Integrado)

Ejem: MC68000



- Microcontrolador → Es un CPU + Periféricos contenidos en un encapsulado
 - Ejemplos:
 - MC68HC11-F1 → Completo
 - MC68HC11-A8 → Aus tero

MC68HC11-F1 → Completo
MC68HC11-A8 → Aus tero

Jerarquía:

MC68HC11-F1 → Completo

MC68HC11-A8 → Austerio

→ Periféricos	- Puertos paralelos	- Puertos seriales
	- Temporizador	- Contadores
	- ADC's	- DAC's
	- RAM	- ROM

// Técnica de diseño → Técnica donde más redundante de 1 procesador funciona // Ancho de bus de datos
↳ 8, 16, ..., 32

► Definiciones

- Set de instrucciones | Conjunto de instrucciones que soporta un CPU.
- Instrucción | Orden puntual para ejecutar una tarea
- Programa | Secuencia de instrucciones para ejecutar un proceso

- BUS - Conductores paralelos de información
- Código de Instrucción - Es el número asociado a cada instrucción que (opcode) se utiliza para identificarla
- Operando - Es un parámetro necesario (opload) para ejecutar instrucciones complejas

► Categoría 1 | De acuerdo con el n.º de instrucciones

Complex Instruction Set Computer

- MC68HC11 → 308 instrucciones

"Fácil" programación de algoritmos complejos.

- Tienen un hardware complejo, voluminoso y caro
- Código pequeño y ligero
- Velocidades rápidas para ejecutar códigos

Reduced Instruction Set Computer

- PIC → 32 instrucciones

"Difícil" programación de algoritmos complejos.

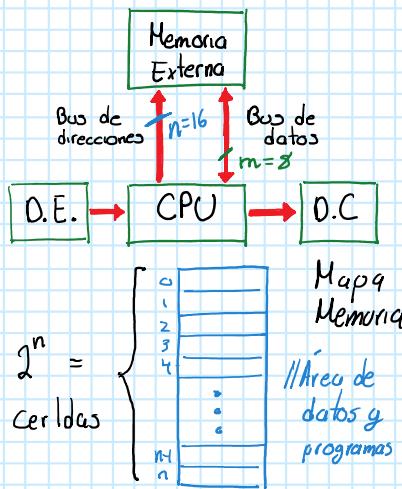
- Tienen un hardware simple, compacto y barato.
- Código grande y pesado
- Velocidades lentas por ejecutar código extenso

► Categoría 2

{ De acuerdo con la forma en que el CPU de una computadora interacciona con la memoria con la memoria y procesadores

Von Neumann (Lenta)

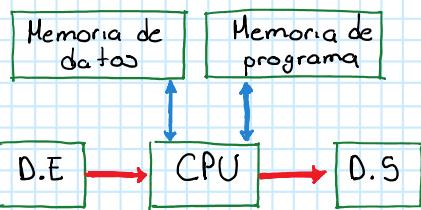
Harvard (Rápidos)



// Desde el punto de vista del CPU existe una memoria única que contiene área de datos y otra área de programas

// Bus (n) → Accedemos a datos

// Bus (m) → Podemos acceder e insertar Datos



// El CPU cuenta con memorias independientes para datos y programas que puede accederse de forma simultánea

/ Familia 80C51 de microcontroladores

Intel → Philips → Atmel

Categoría 3 { De acuerdo con la forma en que un procesador ejecuta instrucciones los CPU's

① Secuenciales

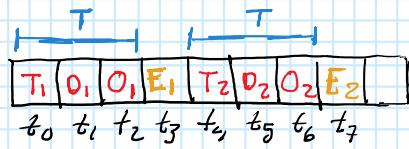
- Se denominan así a las computadoras que siguen una secuencia lógica denominada "Ciclo Fetch" para ejecutar instrucciones.

"Ciclo Fetch"

- ① El CPU trae de memoria externa, del área de programas, el código instrucción a ejecutar.
- ② El CPU decodifica (identifica) el código instrucción.
- ③ El CPU trae de memoria externa, del área de programas, el o los operandos asociados a la instrucción.
- ④ El CPU ejecuta la instrucción

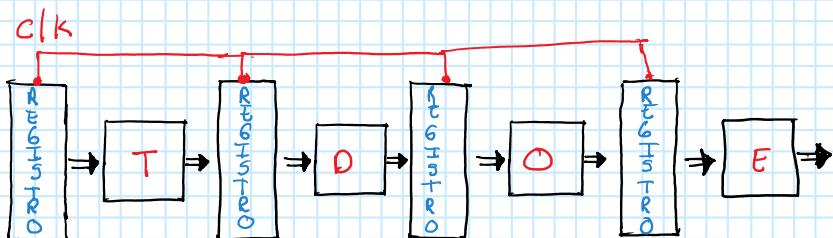
// El 3º puede depender de la instrucción
(simple o compleja)

T → Traer código instrucción
D → Decodificar instrucción
O → Traer operando (s)
E → Ejecutar

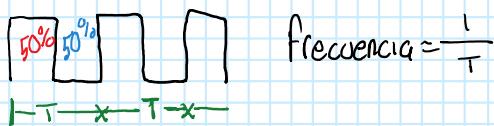


② Paralelas

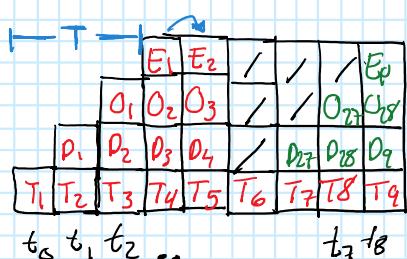
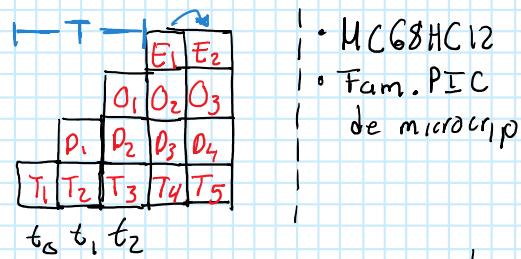
- 1] 2 o más CPU's de tipo Von Neuman, procesando la misma información (Multinúcleo)
- 2] Las computadoras de tipo Harvard
- 3] Los procesadores encauzados (pipe-line)
// Línea de producción



• Señal de reloj - Es una señal eléctrica, periódica, de frecuencia constante, de forma cuadrada y ciclo de trabajo de 50 %



// Arquitectura (Pipe-Line)



- $A + \text{D} = \text{Resultado}$
- $C + D = E$
- $A + B = C$
- Nop
- Nop
- $C + D = E$

- Tiempo Muerto → Generación → Para que el
por hardware de Burbujas procesador ejecu
te correctamente un programa
- Hay problemas respecto a los
sintaxis

2_Arquitecture

Tuesday, August 13, 2019 3:09 PM

Apartado Extra

- De hexadecimal a decimal

$$\begin{array}{l} 0000 = 0 \\ 1000 = 8 \\ \downarrow \quad \downarrow \\ 0001 \quad 0000 \quad 0000 \quad 0000 \end{array}$$

B 6 00
↓ ↓ ↓
1011 0110 0000 0000

- Existen 4 modos en el que puede funcionar el 68HC11

↳ Usaremos modo expandido (expanded)

Arquitectura del microprocesador

- Puertos Paralelos
 - Bidireccionales → PORTA, PORTC, PORTS, PORTD
Control (ODRA) (ODRC)
 - Una dirección → PORTB(out), PORTF(out)
PORTE (In)

\$3000	BIT T	-	-	-	-	-	-	BIT 6	PORTE	Puerto A
\$3001									Reserv.	ODRA*
\$3002	RTAF	STAT	CWOM	HNTS	QIM	PIS	EQA	INVB	PIOC	Control Puerto
\$3003	BIT T	-	-	-	-	-	-	BIT 6	PORTC	Puerto C
\$3004	BIT T	-	-	-	-	-	-	BIT 6	PORTB	Puerto B
\$3005	BIT T	-	-	-	-	-	-	BIT 6	PORTCL	Latch Puerto C
\$3006									Reserv.	ODRB*
\$3007	BIT T	-	-	-	-	-	-	BIT 6	DDRC	Modo E/S Puerto C
\$3008		BIT 5	-	-	-	-	-	BIT 6	PORTD	Puerto D
\$3009		BIT 5	-	-	-	-	-	BIT 6	DDRD	Modo E/S Puerto D
\$300A	BIT T	-	-	-	-	-	-	BIT 6	PORTE	Puerto E
\$300B	FOC1	FOC2	FOC3	FOC4	FOC5				CF0HC	Passar comparaciones
\$300C	OC1M7	OC1M8	OC1M9	OC1M4	OC1M5				OC1M	OC1: Control
\$300D	OC1D7	OC1D8	OC1B5	OC1D4	OC1D0				OC1D	OC1: datos
\$300E	BIT 15	-	-	-	-	-	-	BIT 6	TUNT	Registro contador
\$300F	BIT T	-	-	-	-	-	-	BIT 6		
\$3010	BIT 15	-	-	-	-	-	-	BIT 6	THC1	Captura entrada 1
\$3011	BIT T	-	-	-	-	-	-	BIT 6		
\$3012	BIT 15	-	-	-	-	-	-	BIT 6	THC2	Captura entrada 2
\$3013	BIT T	-	-	-	-	-	-	BIT 6		
\$3014	BIT 15	-	-	-	-	-	-	BIT 6	THC3	Captura entrada 3
\$3015	BIT T	-	-	-	-	-	-	BIT 6		
\$3016	BIT 15	-	-	-	-	-	-	BIT 6	TOC1	Comparador 1
\$3017	BIT T	-	-	-	-	-	-	BIT 6		
\$3018	BIT 15	-	-	-	-	-	-	BIT 6	TOC2	Comparador 2
\$3019	BIT T	-	-	-	-	-	-	BIT 6		
\$301A	BIT 15	-	-	-	-	-	-	BIT 6	TOC3	Comparador 3
\$301B	BIT T	-	-	-	-	-	-	BIT 6		

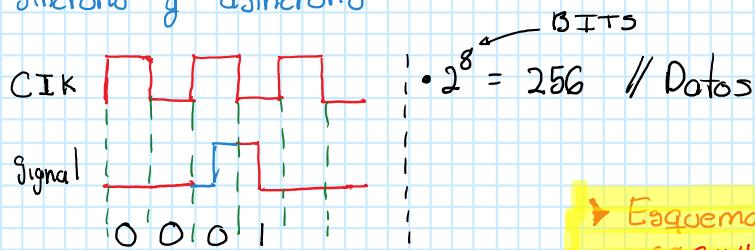
- Puertos Seriales
 - SPI - Serial Port Interface - Puerto serial síncrono
 - SCI - Serial Communication Interface - Puerto serial asíncrono

Puerto Serial vs Paralelo

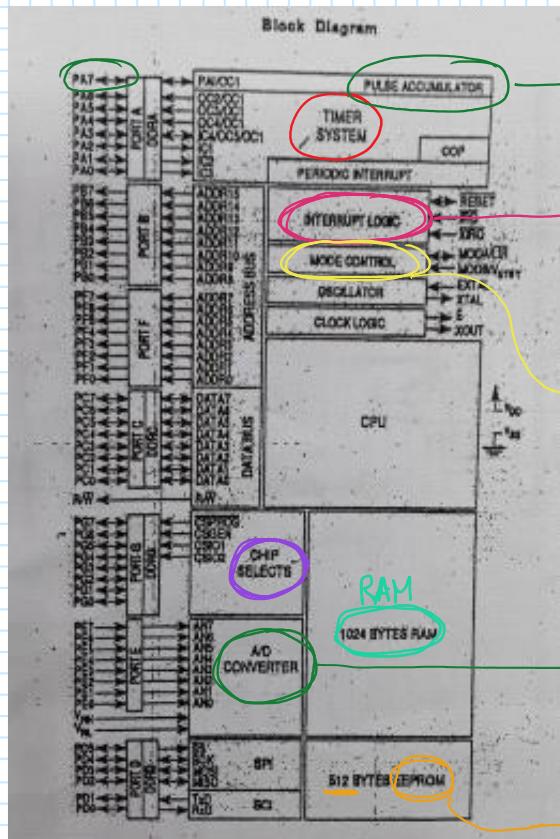
- 1) Puertos paralelos son rápidas pero la distancia es muy pequeña

1) Puertos paralelos son rápidas pero la distancia es muy pequeña

Síncrono y asíncrono



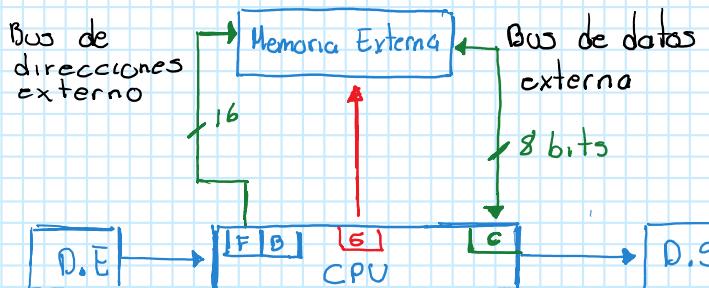
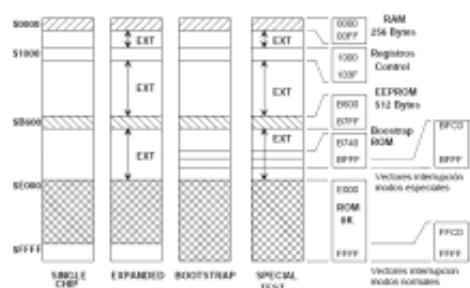
► Esquema del 68HC11



MODA	MODBi	Modo de operación del 68HC11
0	0	Single Chip
0	1	Expanded
1	0	Bootstrap
1	1	special Test }*

* Prueba del micro

El 68HC11 puede funcionar en 4 modos diferentes. Cada chip, expandido, Bootstrap o especial. En cada modo se dispone de un mapa de memoria diferente, como se muestra en la figura 10.



// Sacrificamos 4 puertos para tener más memoria

// Máximo de celdas de memoria externa de una pc

$$\hookrightarrow \text{Capacidad máxima} = 2^{(\text{Bits direcciones})}$$

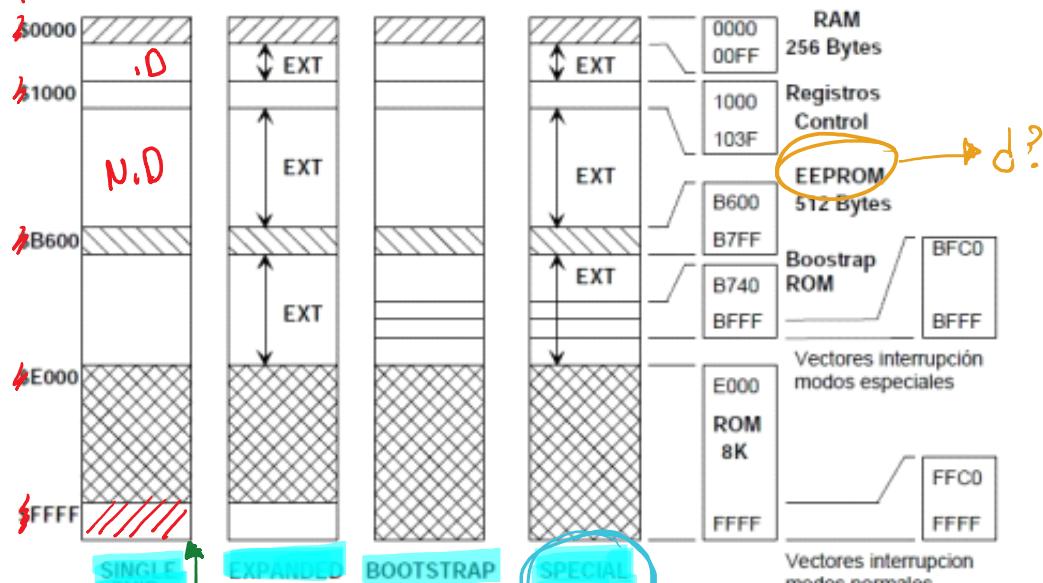
// CS = Chip Select \leftrightarrow Circuito integrado

3_Arquitecture

Tuesday, August 20, 2019 2:03 PM

El 68HC11 puede funcionar en 4 modos diferentes: Single chip, expanded, bootstrap y special test. En cada modo se dispone de un mapa de memoria diferente, como se muestra en la figura 10.

Notación Motorola (\$)



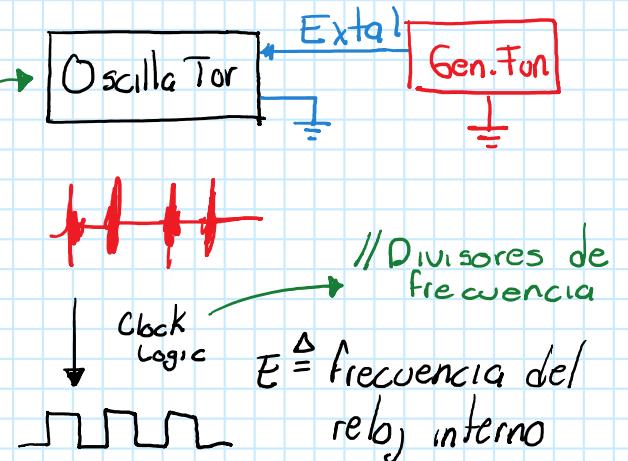
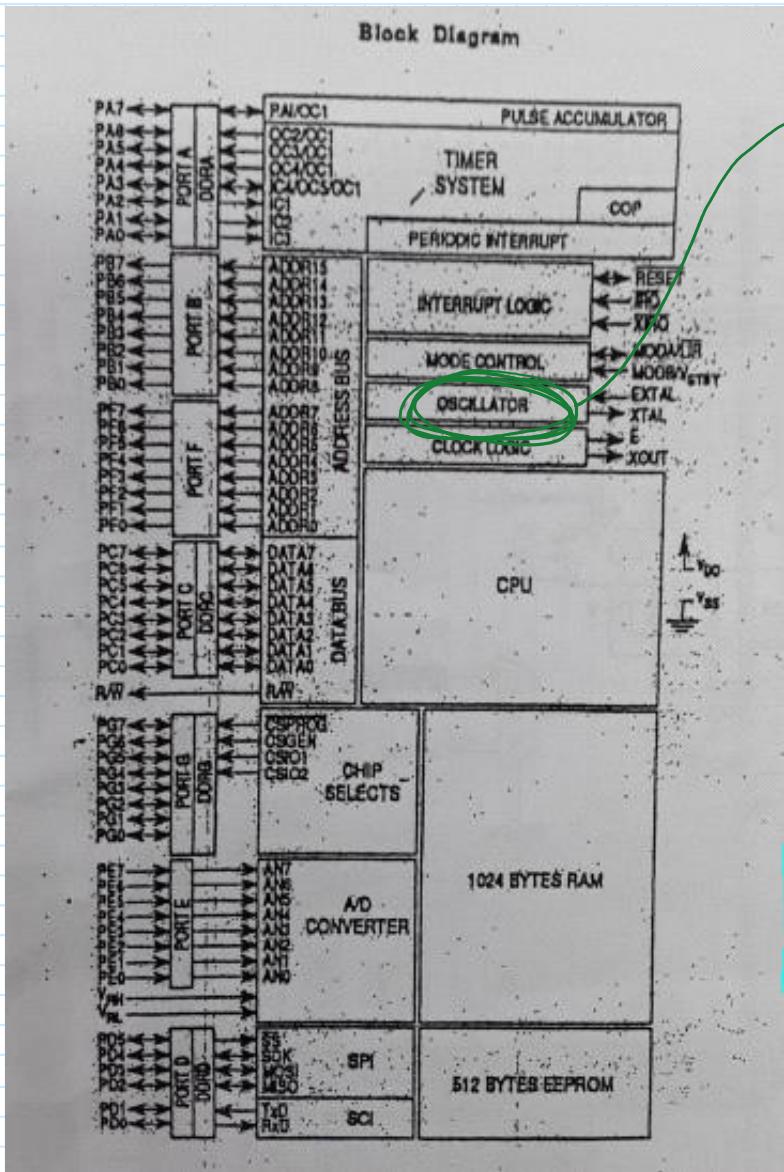
No existe la expansión de memoria

Por el fabricante

Debo programar el retorno posterior a la subrutina.

- ROM → Read Only Memory
- PROM → Programmable ROM
- EPROM → Erasable Programmable ROM
- EEPROM → Electrical EEPROM

||||
Sección donde se coloca la dirección del programa.
//Vector de interrupción del RESET



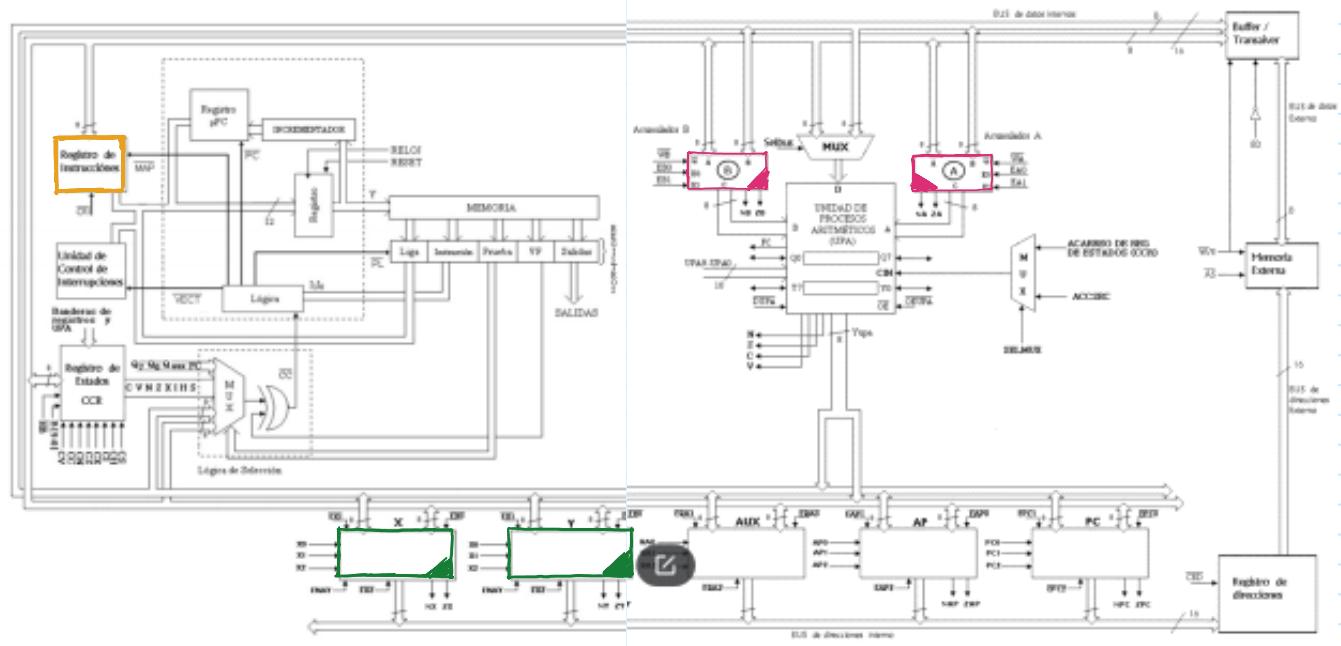
- If $\text{f_crystal} = 8 \text{ MHz}$

$$\therefore E = \frac{8}{4} = 2 \text{ MHz} \cancel{/}$$

- Ciclos de reloj

$$T = \frac{1}{E} = \frac{1}{2 \text{ MHz}} = \frac{1}{2000000} = 5 \times 10^{-7} \text{ seg}$$

Argolectura del micro



► Dispositivos de uso general

- Registros acumuladores A y B
 - ↳ 8 bits → Sirven para almacenar números de 8 bits, hacer operaciones aritméticas
- Registros contenedores X y Y
 - ↳ 16 bits → Sirven para almacenar números de 16 bits
 - Son punteros físicos de memoria para
 - Página memoria \Rightarrow Indexación
 - Accesar información
 - Utilizar tablas de look-up

► Dispositivos de uso dedicado

- Program Counter - Registro Contador
 - ↳ 16 bits → Apunta a la localidad de memoria (del área de programa) donde se encuentra la proxima instrucción a ejecutar
- Stack Pointer (Apuntador de pila) - Registro Contador
 - ↳ 16 bits → Apunta a la cuspide de una estructura dinámica de tipo pila (Last Input - First Out) // LIFO

NOTA | Estado sólido \rightarrow Transistores de Silicio

- Registro Auxiliar

↳ 16 bits

- Registros de estados CCR

↳ 8 bits → Se emplea para administrar el estado del micro, cada bit tiene una función.

① C (Carry) - Bit de acarreo

↳ Se pone en alto cuando la última instrucción aritmética genera un bit adicional que puede emplearse en operaciones por partes

② V(Overflow) - Bit que se pone en alto para indicar que la última instrucción

↳ desbordó la capacidad de la ALU (Unidad Aritmética Lógica)

③ Z (zero) - Bit que se pone en alto para indicar que el resultado de la última instrucción aritmética-lógica fue cero

→ También se pone en alto cuando se inicia el valor de un registro o localidad de memoria con la constante cero.

NOTA Recuerda que será al revés al usarlo

④ N(Negative) - Se pone en alto cuando la última instrucción genera un resultado con el bit más significativo en alto (MSB)

// Es una convención

⑤ i (Interrupt) - ... para indicar que el CPU atiende una interrupción.

// LAS SIGUIENTES SON DEL 68HC11

⑥ x (Interrupt) - Sirve para no sólo tener una interrupción

// Menor prioridad

⑦ H (Half Carry) - Cuando la última suma aritmética con operandos BCD, que genera un resultado superior a nueve

NOTA	Binario Conv.	Binario - BCD
	0000	- 0000
	⋮	⋮
	1001	- 1001 ← 10
	1010	- NO VÁLIDO
	⋮	⋮ ⋮ ⋮ ⋮
	-	1000 1000

Ejemplos

• Bin. Convencional

$$\begin{array}{r} 1001 \\ + 0010 \\ \hline 1011 \end{array}$$

• BCD

$$\begin{array}{r} 0001 \\ 0000 \\ 0000 \\ + 0001 \\ \hline 0001 \end{array} \quad \begin{array}{r} 1001 \\ 0010 \\ + 0001 \\ \hline 0001 \end{array} \rightarrow \begin{array}{r} 09 \\ 02 \\ \hline 11 \end{array}$$

Realmente lo hace dentro

// Caso donde no es "BCD"

$$\begin{array}{r} 0000 \ 1010 \\ + 0000 \ 0001 \\ \hline 0000 \ 1011 \end{array}$$

$$H = 0 //$$

// Caso donde es BCD

$$\begin{array}{r} 0000 \ 1001 \\ + 0000 \ 0010 \\ \hline 0000 \ 1011 \end{array}$$

$$H = 1 //$$

⑧ (S) stop - Indica que el reloj interno "T" se encuentra detenido.

- Modos de direccionamiento | • Son diferentes formas en las que el CPU accede a memoria externa para ejecutar una instrucción
| • En MC68HC11 6 modos diferentes

- 1) Inherente (INH)
- 2) Inmediato (IMM)
- 3) Directo (DIR)
- 4) Extendido (EXT)
- 5) Indexado (IND, X) o (IND, Y)
- 6) Relativo (REL)

- Directiva de ensamblador | Se refiere a mnemónicos que no tienen asociados un código de instrucciones
| ↳ Se emplean para darle estructura a un programa

► Modo inherente (INH)

- Pertenecen a esta categoría todas las instrucciones del set, que carecen de operandos
- Poseen un código de instrucciones de 8 o 16 bits
- No comparten mnemónicos con ninguna inst.

ORG \$8000	2000	01	Unidad dato en bytes
NOP		08	2 bytes
INX	8005	09	
DEX		18	
INY		08	
DEY		18	
MUL		09	
IDIV		31	
FDIV		02	
XGDX	8009	03	
XGDY	800A	2F	
		18	
		8F	
		FF	
		FF	
END	800E	FF	

I INY es de 16 bits
∴ Poner en 2 espacios

OPCODE

// Primer proyecto

programa.asm

→ programa.s19

- Código fuente
- Programa en lenguaje ensamblador (Texto)

- (Código Objeto)
- Programa en lenguaje máquina

NOTA: Guardar codificado en ANSI.

► Modo de direccionamiento inmediato (IMM)

- Pertenecen todas instrucciones del set, que poseen un código de instrucción de 8 o 16 bits y un operando de 8 o 16 bits al que se le da trato de dato inmediato.
- Comparte mnemónicos con modo directo, extendido o indexado.
- Se distingue por usar el símbolo #

ORG \$8000	2000	86	// Begin index
LOAA #\$25		25	↳ Convención de

ORG \$8000	8000	86	// Begin indica
LDAA # \$25	8005	25	↳ Convención de ordenar
LDAB # \$3C		CG	
LOX # \$1789		3C	
LDY # \$ABCD		CE	7789 } //Parte alta
AODA # \$45		17	
LDDA # , K		89	
END		18 } LDY	
		CE	
		AB	
		CD	
		00	
		45	
	800E		

Columna Renglón
k = 0110 1011 ① ②

- NOTA:** No tener END
- Tipo de errores:**
- ② Mnemónico inexistente
 - ③ Magnitud de operando errónea
 - ④ Faltó operando

► Modo direccionamiento directo (DIR)

- Pertenecen a esta categoría todas las instrucciones del set, que poseen un código de instrucción de 8 o 16 bits y un operando de 8 bits al que se le trata de dirección
- Comparte mnemónicos con modos inmediato extendido e indexado
- Se distingue de otros modos de direccionamiento porque el operando no tiene ningún símbolo

ORG \$8000	8000	96	NOTA
LDAA \$25	8005	25	Sólo sirve
LDAB \$3C		DE	acceder la
LOX \$1789		3C	parte baja de
LDY \$ABCD		9E	memoria
AODA \$45		17	\$0000 a \$00FF
STTA \$25		89	<u>RAM</u>
END	800E	18	
		DE	
		AB	
		CD	
		00	
		45	
		97	
		25	

// Caso Particular | BR CLR
BR SET

• Modos de direccionamiento

INH

IMM → #

DIR → Operando de 8 bits

EXT → Operando de 16 bits

IND → Operando, X u Operando, Y

REL → Operando es etiqueta

► Modos de direccionamiento extendido

- Tiene un código de instrucción de 8 o 16 bits y un operando de 16 bits al que se le da tratamiento de dirección.
- Comparte mnemónico con modos de direccionamiento inmediato, directo e indexado.
- Programador es responsable de escribir en sectores de RAM no de ROM

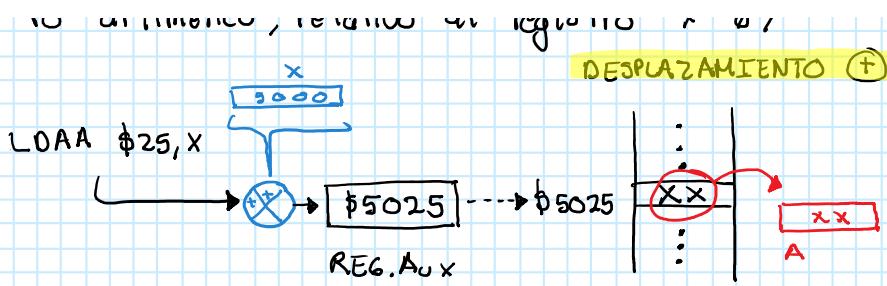
ORG \$ 8000	8000	BC
LDAA \$ 257C	8005	25
LDAB \$ 1789		7C
LDX \$ 47AB	8009	F6
LDY \$ ABCD	800A	17
ADDA \$ 7C41		89
STAA \$ 9000	800E	FE
END		47
		AB
		18
		FE
		A0
		CD
		BB
		7C
		41
		87
		90
		00

► Modo de direccionamiento indexado (IND,Y) φ (IND, X)

- Poseen un código de instrucción de 8 o 16 bits y un operando de 8 bits, seguido de una coma con una "X" o "Y"
- Al operando se le da tratamiento de desplazamiento aritmético, relativo al registro X o Y

X
3000

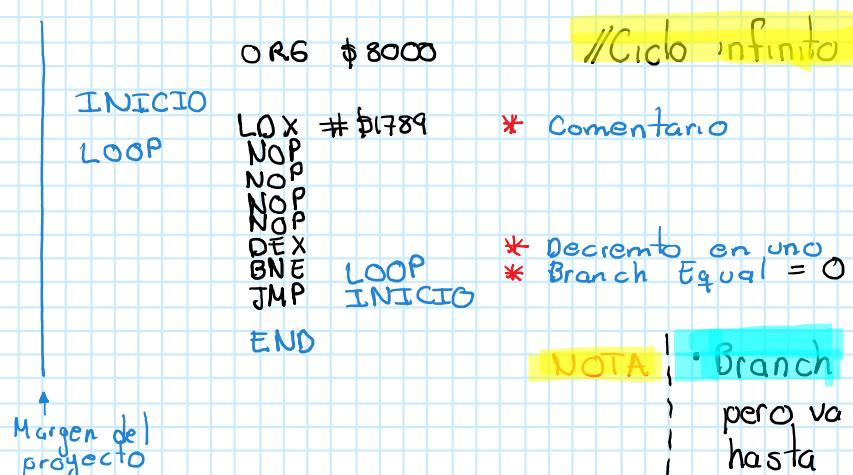
DESPLAZAMIENTO (+)



ORG \$8000	\$ 8000
LDAA \$25,X	25
LDAB \$37,Y	18
LD X \$15,X	E6
LO Y \$2C,Y	37
ADDA \$AB,X	EE
STAA \$9C,Y	15
END	18
	EE
	AB
	AB
	18
	EE
	ac

► Modo de Direcciónamiento Relativo

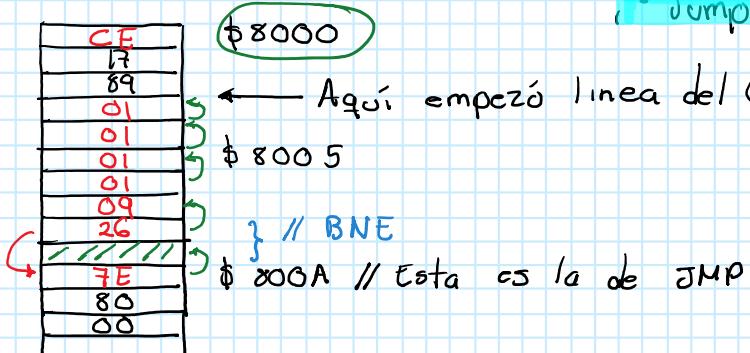
- Poseen un código de instrucción de 8 o 16 bits y un operando de 8 bits en forma de etiqueta (Jamás numérica)
- Al operando se le da tratamiento algebraico, relativo al registro de PC.
- No comparte mnemónico con otros modos de direcciónamiento



NOTA:

- Branch es rápido
- pero va de -127 hasta 128
- Jmp sólo está en sentido

⇒ Necesita la directiva de la diferencia



FE	\$ 200A // Esta es la de JMP
80	
00	

► Complemento de 8 en Hexadecimal

128 64 32 16 8 4 2 1
 0 0 0 0 1 0 0 0 ↓ Negamos
 1 1 1 1 0 1 1 1 } Sumamos 1
—————
 1 1 1 1 1 0 0 0 ← Complemento a 2
 F 0 ← de 8 con 8 bits
 Hexadecimal

- En la celda →

$$\begin{array}{r}
 1000\ 0000\ 0000\ 1011 \leftarrow \$800B \\
 + \quad 1111\ 1111\ 1111\ 1000 \leftarrow \$FFF8 \\
 \hline
 \textcircled{1} \quad 1000\ \underbrace{0000}\ \underbrace{0000}\ \underbrace{0011} \\
 \text{C} \quad \underbrace{8}\ \underbrace{0}\ \underbrace{0}\ \underbrace{3} \leftarrow \text{Ubicación 200P}
 \end{array}$$

6_Compilación

Tuesday, September 3, 2019 2:01 PM

▶ Compilación

Programa *.asm

Programa en
lenguaje
ensamblador

// Source Code

Compilador

Programa *.obj

Programa en
código
maquina

// Object Code

*.obj

Código fuente
+
Código
Objeto

// Listado

▶ Características del lenguaje

- Variables → Asociadas a la memoria RAM (0-256)
- Constantes → Asociados a registros (No a RAM)

Núm de Línea	Código Objeto	Código Fuente
1 A	1026	PACTL EQU \$1026
2 A		*El previo es una constante
3 A		PI EQU \$3141
4 A		DDRA EQU \$1001
		*El previo es un variable y 0012 es la ubicación en me- moria RAM.

// Estructura de una declaración

PACTL EQU \$1026
 ID = Valor asociado
 // Palabra
 recorrida.

NOTA

Demonicos
pueden ir en
mayúsculas
o minúsculas

ID = Valor asociado
 // Palabra reservada

mayúsculas
 o minúsculas

60	8000	ORG \$8000
		* Dir del inicio
	8E03FF	LOS #\$03FF //Modo inmediato
	8600	* Apuntador de stack
65	B71001	LOAA #\$00
		STA A \$00RA

Como estaría en memoria

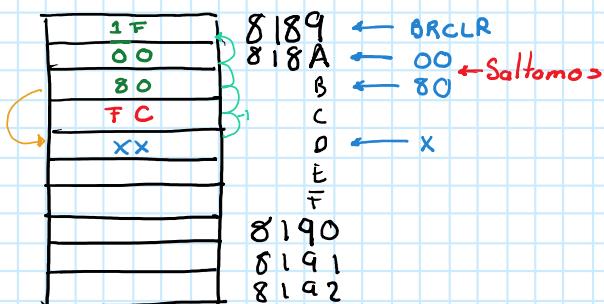
8E	8000
03	
FF	
86	8005
00	
B7	
10	
01	

Casos Particulares

	DIR		INDEXADO
*	BRCLR	BRCLR \$00, #\\$80	T4 BRCLR \$00, X, #\\$80 T4
	BRSET	T4 BRSET \$00, #\\$80 T4	T4 BRSET \$00, Y, #\\$80 T4
	BCLR		
	BSET		

// Sólo en modo de direccionamiento directo e indexado

* Compilado



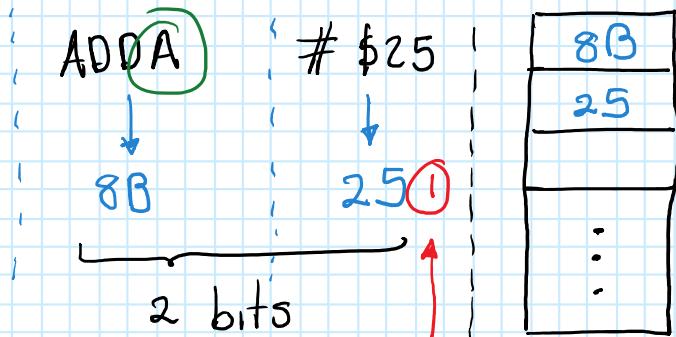
$$\begin{array}{l}
 \rightarrow \text{Pasando } "-4" \text{ a hexadecimal} \\
 0000\ 0100 \leftarrow 4 \\
 + 1111\ 1011 \} -4 \\
 \hline
 1111\ 1100
 \end{array}$$

Se salta hasta inicio
 ∴ El valor es "-4"

1111 1100
 F C

► Ensamblador

- Etiqueta : Nomenclado : Variables

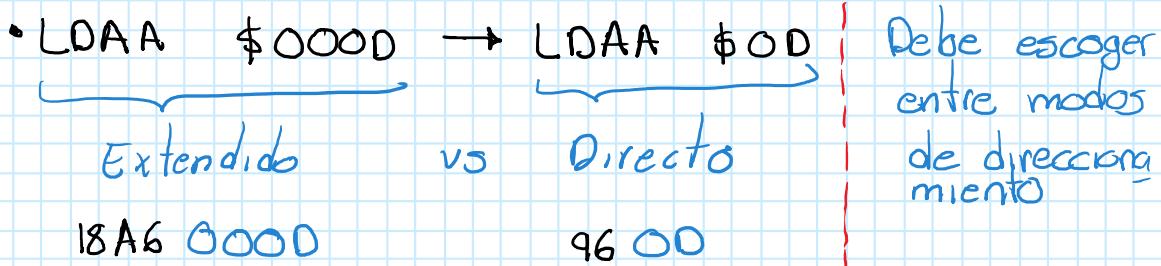


// Acomuladores

// Si le agrego otro marcario error

// Hay acumuladores

- a, b es de 8 bits
- d es de 16 bits



• Encabezado | M68HC11 Absolute Assembler
 Versión 2.7 Dirección del archivo y nombre.ASC

- Hay mnemónicas que no tienen operando

► Carpetas 68HC11

- MIO → Archivos
 - .Asm
 - los compilados
 - ↳ Listado (Formato Motorola)
 - Compilados

► Formato Motorola (Listado)

Línea	Código Objeto	Código Fuente
Línea - Separación con A	Dirección	Etiquetas mnemónicas Variables

8000 86102E

► Formato Motorola (Object Code)

```

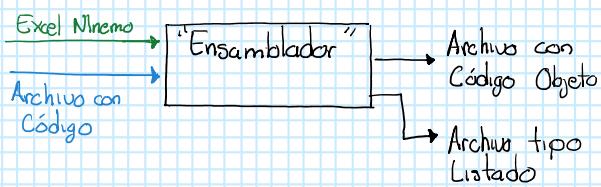
5000 0000 2020 2020 2020 2020 F4
5121 8000 _____ (31 espacios)
      801E
      ↓ Salto de páginas "MG8HC11 Absolute"
      Asamble version: _____
      dirección file
5903 8000 7C
  
```

► Subrutinas y Código

- Downloader → Es un programa para puerto serial y está en Ensamblador
 - Variables
 - NAME EQU Dirección Memoria RAM
- | | |
|--------|------------|
| \$0000 | Directions |
| ⋮ | de RAM |
| \$0FF | |

► Procesos por los que debe pasar

// Diagrama de procesos



// Partes o propuestas del Ensamblador

- Cargar lista de Mnemónicos

LOGIC PART

- ① Identificar modo
 - ② Reconocer etiquetas
 - ③ Checar si hay subrutinas
 - ④ Casos particulares mnemónicos genéricos "
 - ⑤ Directivas Ensamblador → ORG/EQU/FCB/END
 - ⑥ Vector de interrupción → FCB
También inicializar valores localidad de memoria
- *FCB 01, 02, ... 0n

- Modos de direccionamiento (Necesidades)

- Subroutines are like function in high level language.
- We can have subroutines anidadas

NOTE
We have 2 types of subroutines RTS and RTI

► Interruption Subroutine

- It doesn't start with a label
- It ends with RTI

// Example

* SERIAL INTERRUPT

```

ORG $F100
⋮
RTI
  
```