



UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA

ESCUELA DE CIENCIAS BASICAS TECNOLOGIA E INGENIERIA

PROGRAMA DE INGENIERIA ELECTRONICA

309696 – MICROPROCESADORES Y MICROCONTROLADORES

HECTOR URIEL VILLAMIL GONZALEZ
(Director Nacional)

Jairo Luis Gutierrez
Acreditador

CHIQUINQUIRA
Julio de 2013

INDICE DE CONTENIDO

INTRODUCCIÓN	17
UNIDAD 1 MICROPROCESADORES	24
CAPITULO 1: MICROPROCESADORES Y MICROCOMPUTADORES	24
Introducción	24
Lección 1: Invención e historia del microprocesador y el microcomputador	24
Lección 2: Bases numéricas, códigos binarios, bits y bytes	41
Lección 3: Circuitos lógicos y Dispositivos Digitales básicos.....	55
Lección 4: Arquitectura, funcionamiento, organización de la memoria, Registros, segmentos, instrucciones y Modos de direccionamiento en un Microcomputador	64
Lección 5: El microprocesador.....	73
CAPITULO 2: FAMILIAS DE MICROPROCESADORES	91
Introducción	91
Lección 6: Principales Familias de Microprocesadores	91
Lección 7: Microprocesadores de 8 bits	98

Lección 8: Microprocesadores de 16 bits.....	104
Lección 9: Microprocesadores de 32 bits.....	108
Lección 10: Microprocesadores de 64 bits, microprocesadores en supercomputadoras y móviles	120
CAPITULO 3: LENGUAJES DE PROGRAMACIÓN EN LOS MICROPROCESADORES	128
Introducción	128
Lección 11: Fundamentos de programación en assembler para microprocesadores	128
Lección 12: Diagrama de flujo o bloques y subrutinas.....	134
Lección 13: Programación con debug y ensamblador	144
Lección 14: Programación con simuproc y MASM	159
Lección 15: Ejemplos de aplicación.....	177
Actividades de Autoevaluación de la UNIDAD.....	183
Fuentes Documentales de la Unidad 1	184
UNIDAD 2 MICROCONTROLADORES	186
CAPITULO 4: INTRODUCCIÓN A LOS MICROCONTROLADORES	186
Lección 16: Generalidades de los microcontroladores	186
Lección 17: Arquitectura interna y direccionamiento	204
Lección 18: Sistemas microcontrolados: ensamblador y programación en microcontroladores	213
Lección 19: Familias de microcontroladores.....	224

Lección 20: Herramientas y sistemas de desarrollo para microcontroladores . 239

CAPITULO 5: MICROCONTROLADORES PIC DE MICROCHIP 256

Introducción 256

Lección 21: Microcontroladores PIC 256

Lección 22: Microcontroladores PIC16F84 277

Lección 23: Ejemplos de programación con PIC16F84 283

Lección 24: Microcontroladores PIC16F877 294

Lección 25: Ejemplos de programación con PIC16F877 303

CAPITULO 6: MICROCONTROLADORES MOTOROLA FREESCALE , BASIC STAMP Y ARDUINO 309

Introducción 309

Lección 26: Microcontroladores Motorola Freescale 309

Lección 27: Microcontrolador MC68H(R)C908/JL3/JK3/JK1 y Ejemplos de aplicación..... 330

Lección 28: Microcontroladores Texas MSP430..... 346

Lección 29: Microcontrolador Texas y LaunchPad MSP430 Ejemplos de aplicación..... 351

Lección 30: Modulos Microcontrolados Arduino y BASIC Stamp..... 357

Actividades de Autoevaluación de la UNIDAD..... 363

Fuentes Documentales de la Unidad 2 365

UNIDAD 3 PROGRAMACION Y DESARROLLO DE PROYECTOS CON MICROPROCESADORES Y MICROCONTROLADORES 367

CAPITULO 7: DISEÑO Y DESARROLLO DE PROYECTOS CON MICROCONTROLADORES Y MICROPROCESADORES	367
Introducción	367
Lección 31: Principios fundamentales en electrónica y circuitos	367
Lección 32: Periféricos para microprocesadores y microcontroladores.....	374
Lección 33: Diseño de la solución de hardware.....	379
Lección 34: Diseño de la solución de software	382
Lección 35: Integración del hardware y software en la solución de control	384
CAPITULO 8: PROGRAMACION BASICA.....	388
Introducción	388
Lección 37: Operaciones de entrada / salida con ensamblador	393
Lección 38: Manejo de interrupciones y timers.....	395
Lección 39: Exploración de teclado matricial con ensamblador.....	397
Lección 40: Manejo de display 7 segmentos y pantallas LCD con ensamblador	399
CAPITULO 9: PROGRAMACION AVANZADA	411
Introducción	411
Lección 41: Comunicación entre Microcontroladores	411
Lección 42: Manejo de ADC, PWM y módulos especiales	418
Lección 43: Manejo de memorias externas	422
Lección 44: Programación con módulo microcontrolado Basic Stamp	424

Lección 45: Programación con módulo microcontrolador Arduino.....	426
Actividades de Autoevaluación de la UNIDAD.....	429
Fuentes Documentales de la Unidad 3.....	431
Curso de Microcontroladores Motorola, extraído el 10 de Julio de 2009 desde http://www.geocities.com/moto_hc08/index.html	432
SOFTWARE LIBRE.....	433
RECURSOS AUDIOVISUALES	434
GLOSARIO DE TÉRMINOS	435
Características tecnográficas.....	447

LISTADO DE TABLAS

Tabla 1. <i>Evolución en el tiempo de las instrucciones por segundo.</i>	39
Tabla 2. Conversión entre binarios y octales.....	46
Tabla 3. <i>Códigos binarios para dígitos decimales</i>	48
Tabla 4. Código reflejado o Gray de 4 bits	50
Tabla 5. Tabla ASCII y tabla ASCII extendida.....	51
Tabla 6. Funciones y Compuerta lógicas	56
Tabla 7. <i>Representación con signo en la ALU</i>	77
Tabla 8. <i>Generaciones SPARC</i>	97
Tabla 9. <i>Intel 8080 descripción de pines</i>	99
Tabla 10. <i>Descripción general de pines</i>	100
Tabla 11. <i>Código maquina</i>	130
Tabla 12. <i>Código máquina y nemotécnico</i>	131
Tabla 13. Formato de presentación de las Instrucciones	163
Tabla 14. Instrucciones Soportadas por Simuproc 1.4.3.....	163
Tabla 15. Instrucciones Soportadas por Simuproc 1.4.3.....	164
Tabla 16. Instrucciones Soportadas por Simuproc 1.4.3.....	165
Tabla 17. Continuación - Instrucciones Soportadas por Simuproc 1.4.3.....	166
Tabla 18. Continuación - Instrucciones Soportadas por Simuproc 1.4.3.....	167
Tabla 19. Continuación - Instrucciones Soportadas por Simuproc 1.4.3.....	168
Tabla 20. Continuación - Instrucciones Soportadas por Simuproc 1.4.3.....	169
Tabla 21. <i>Microcontroladores vs Microprocesadores</i>	192
Tabla 22. <i>Instrucciones y assembler</i>	209

Tabla 23. Archivos generados por ensamblador	217
Tabla 24. Gamma Enana PIC	226
Tabla 25. Gamma Baja PIC.....	228
Tabla 26. Gamma Media PIC.....	229
Tabla 27. Gamma Alta PIC.....	230
Tabla 28. Familia Motorola	231
Tabla 29. PIC Estándar y Extendido	257
Tabla 30. Valores típicos de Cristal/Resonador y condensadores	260
Tabla 31. Registro Status	263
Tabla 32. Registro INTCON	265
Tabla 33. Registro OPTION	267
Tabla 34. Pre-escalera registro OPTION	268
Tabla 35. Registro EECON1.	268
Tabla 36. Formato de instrucciones PIC gama media.....	275
Tabla 37. Set de Instrucciones	275
Tabla 38. Descripción de pines PIC16F84	280
Tabla 39. Descripción de pines PIC16F877	297
Tabla 40. Registros asociados con el conversor A/D del PIC16F877	303
Tabla 41. Variación de la familia MC68H(R)C908/JL3/JK3/JK1	310
Tabla 42. Prefijos en los Motorola Freescale	317
Tabla 43. Instrucciones Motorola Freescale.....	317
Tabla 44. Representación del ciclo de retardo en un PIC	389
Tabla 45. Pines y funciones LCD	404

LISTADO DE GRÁFICOS Y FIGURAS

Figura 1. Ábaco romano.....	25
Figura 2. Los “Naiper Bones”	25
Figura 3. Los “Slide-rule” o regla de calculo	26
Figura 4. Pascalina.....	26
Figura 5. Calculadora universal.....	27
Figura 6. La réplica de la máquina diferencial del Museo de Ciencias de Londres	27
Figura 7. Relevo o relay	28
Figura 8. Z3 (1941).....	29
Figura 9. Atanasoff Berry Computer (ABC).	30
Figura 10. ENIAC (1946).....	30
Figura 11. EDVAC (Electronic Discrete Variable Automatic Computer)	31
Figura 12. Transistor (1948) junto con transistores modernos.	32
Figura 13. Circuito Integrado (1958).....	32
Figura 14. Circuito Integrado del microprocesador Intel 4004 (1971).....	33
Figura 15. IBM PC (1981)	34
Figura 16. Motorola 6800	35
Figura 17. Conversión decimal-binario.....	44
Figura 18. Tabla de Conversión entre números decimales, octales y hexadecimales	45
Figura 19. Nibble de salida en compuerta XOR con dos nibbles en sus entradas.	56
Figura 20. Flip-Flop RS, construcción con compuertas NOR y tabla de verdad....	57
Figura 21. Flip-Flop tipo JK	58

Figura 22. Flip-Flop tipo D	58
Figura 23. Flip-Flop tipo T	58
Figura 24 Codificador 74HC147 y Decodificador 74LS47	59
Figura 25. Buffer de tres estados	60
Figura 26. Circuitos integrados con Schmitt Trigger (74132 y 74640).....	61
Figura 27. Diagrama de Bloques de CI-74S473, Circuito físico CI-27C16, Diagrama y Pines 28C64A.	62
Figura 28. Diagrama de Bloques, Pines y encapsulado CI-MCM6264C y CI-8264A-15 ²³	62
Figura 29 Circuito 27F256 CMOS Flash y DS1220AB/AD NVRAM	63
Figura 30. <i>Sistema de básico de cómputo</i>	66
Figura 31. Organización de memoria física para obtener un bloque de 1K x 8bits de memoria.	67
Figura 32. Organización de la memoria en un microprocesador 8080 y 8086	68
Figura 33. <i>Unidades Básicas de un MP</i>	75
Figura 34. <i>Pasos en la ejecución de una instrucción</i>	76
Figura 35. <i>Interconexión de la ALU</i>	77
Figura 36. <i>Unidad de Control</i>	78
Figura 37. <i>I/O en la Unidad de Control</i>	79
Figura 38. Ejemplos de Sócalos y tipos de encapsulado, de izquierda a derecha 8080, 80386, Pentium.	89
Figura 39. Cyrix 6x86, Cyrix MII, Cyrix MediaGX y Cyrix III	94
Figura 40. Construcción Power PC 601, PPC750CXE, PowerPC 7450.....	95
Figura 41. Sun Ultra SPARCII y Ultra SPARC IV.....	97
Figura 42. Intel 8080 y 8085 pines	99
Figura 43. Arquitectura Intel 8080	101
Figura 44. Arquitectura Intel 8085.....	102
Figura 45. Set instrucciones 8085.....	102
Figura 46. 80286y pines del 80286	105
Figura 47. Arquitectura del 80286	106

Figura 48. Intel Pentium	109
Figura 49. Intel Pentium II	110
Figura 50. Intel Pentium III	111
Figura 51. AMD K6.....	113
Figura 52. Fases de realización de un programa	132
Figura 53. Fases de realización de un programa	136
Figura 54. Símbolos más comunes para diagramas de flujo	137
Figura 55. Diagrama de flujo para mostrar u ocultar un archivo.....	139
Figura 56. Entrando a Debugger.....	147
Figura 57. Ejecutar un programa con Debugger	148
Figura 58. Desensamblar un programa con Debugger	148
Figura 59. Comando “Trace” o de rastreo de ejecución en Debugger	149
Figura 60. Procedimiento que guarda un programa desde Debugger.....	150
Figura 61. Procedimiento para cargar un programa desde Debugger	150
Figura 62. Como multiplicar.....	155
Figura 63. Desplazamiento de bits con instrucción “SHL”	157
Figura 64. Ventana principal de Simuproc	160
Figura 65. Ejemplo de iteración utilizando la instrucción “LOOP”.....	178
Figura 66. Iteración con “DEC BX” y condición de salto “JNZ”.....	178
Figura 67. Iteración utilizando el contador “CX” y el salto condicional “JCXZ”	179
Figura 68. Uso de las Interrupciones.....	180
Figura 69. Uso de la interrupción 21H de DOS	180
Figura 70. Programa que permite ocultar o mostrar un archivo.	181
Figura 71. Interfaz del programa anterior, ventana DOS.....	182
Figura 72. Estructura de microcontrolador	189
Figura 73. Microchip 32 bits	189
Figura 74. Motorola 16, 32 bits.....	190
Figura 75. AVR-Atmel 32 bits.....	190
Figura 76. Sistema general microcontrolador.....	196

Figura 77. Arquitectura Von Neumann	205
Figura 78. Arquitectura Harvard	206
Figura 79. Oscilador con Cristal de cuarzo.....	208
Figura 80. PIC gamma baja o enana PIC12Cxx.....	226
Figura 81. PIC gamma media PIC16Cxx	227
Figura 82. PIC gamma alta PIC17CXX	228
Figura 83. Microcontrolador Motorola Freescale	232
Figura 84. HC08	233
Figura 85. Familia MSC51.....	234
Figura 86. Arquitectura interna MSC51	235
Figura 87. Familia ATME.....	236
Figura 88. Entorno de desarrollo WinIDE	246
Figura 89. Entorno de simulación WinIDE	247
Figura 90. Ventana de inicio de CCS v.5, selección del Workspace	248
Figura 91. Interfaz del CCS v 5	249
Figura 92. Entorno de programación y simulación In-Circuit CCS v5.....	249
Figura 93. Programa listo para simular con tarjeta LaunchPad MSP430	250
Figura 94. Interfaz IAR Embedded WorkBench.....	251
Figura 95. Interfaz de desarrollo Proteus Professional v.7.9 – ISIS	252
Figura 96. Entorno de desarrollo Multisim	253
Figura 97. Conexión básica de Oscilador externo y circuito Reset (con circuito POR) para PIC.....	260
Figura 98.Organizacion de la memoria de programa	261
Figura 99. Organización de la memoria de datos.....	262
Figura 100.Cambio de bancos	263
Figura 101. Diferencia entre direccionamiento directo e indirecto.....	270
Figura 102. Arquitectura del PIC16F84.....	277
Figura 103. Organización de la memoria	278
Figura 104. PIC16F84A.....	279
Figura 105. Registro Puerto A y TRISA.....	280

Figura 106. Registro Puerto B y TRISB.....	281
Figura 107. <i>Timer0</i>	281
Figura 108. Configuración del Oscilador	282
Figura 109. Circuito Reset (con circuito POR) y Oscilador externo y para PIC16F84	283
Figura 110. Diagrama de conexión	290
Figura 111. Arquitectura del PICF877	295
Figura 112. <i>PIC16F877/874</i>	296
Figura 113. Registro Puerto A, TRISA y ADCON1 PIC16F877	298
Figura 114. Puerto B, TRISB y OPTION en PIC16F877	298
Figura 115. Puerto C y TRISC en PIC16F877.....	299
Figura 116. Puerto D y TRISD en PIC16F877.....	299
Figura 117. Puerto E, TIRSE y ADCON1 en PIC16F877	299
Figura 118. <i>Timer0 PIC16F877</i>	300
Figura 119. <i>Timer1 PIC16F877A</i>	301
Figura 120. <i>Conversión Análoga - Digital</i>	302
Figura 121. Arquitectura del MC68H(R)C908JL3E	310
Figura 122. Funciones de pines Motorola Freescale MC68H(R)C908	311
Figura 123. Fuentes de interrupción en Motorola Freescale	316
Figura 124. Arquitectura de registros en el MC68H(R)C908/JL3/JK3/JK1	331
Figura 125. <i>Arquitectura</i>	332
Figura 126. <i>Bloque de memoria</i>	333
Figura 127. <i>Memoria en los Motorola Freescale, Fuente</i>	334
Figura 128. Registro de banderas (CCR) Motorola Freescale.	336
Figura 129. <i>JL3,JK3 y JK1</i>	337
Figura 130. <i>Configuración de pines</i>	338
Figura 131. Circuito de reloj en MSP430.....	347
Figura 132. Sistema de Reset en el MSP430.....	348
Figura 133. Registros de MSP430	349
Figura 134. Instrucciones MSP430	350

Figura 135. Instrucciones MSP430 - continuación	351
Figura 137. Resistencias fijas y código de colores.....	369
Figura 138. Símbolos de resistencias, Condensadores, bobinas Diodos e interruptores	372
Figura 139. Símbolos de instrumentación, dispositivos activos, transistores, tiristores y electrónica Digital.....	373
Figura 140. Símbolos de transformadores y otros de uso común.	374
Figura 141. Temporizador	389
Figura 142. Tres posibilidades para una pregunta	391
Figura 143. Hardware para ejercicio Entrada/Salida	394
Figura 144. Ejemplo de servicio a Interrupción externa en PIC16F84	395
Figura 145. Diagrama eléctrico para visualización dinámica en display 7 segmentos de dos dígitos.....	401
Figura 146. Diagrama de Flujo manejo de display con PIC16F84	402
Figura 147. Disposición típica de pines en dispositivos síncronos (a) I ² C, (b) SPI	413
Figura 148.Bits de START y STOP del protocolo I ² C.....	413
Figura 149. Temporización en el bus I ² C	414
Figura 150. Entrada de datos a dispositivo SPI.....	414
Figura 151. Salida de datos de dispositivo SPI	414
Figura 152. Mapa de tiempos.....	415

ASPECTOS DE PROPIEDAD INTELECTUAL Y VERSIONAMIENTO

El modulo en su versión 2004 fue diseñado por el Ingeniero Freddy Reinaldo Téllez Acuña, docente de la UNAD. El Ingeniero Freddy Reinaldo es Ingeniero Electricista y Magister en Potencia Eléctrica. El modulo en su versión 2009 fue diseñado por el Ingeniero Héctor Uriel Villamil González, tutor de la UNAD y revisado en estilo y contenidos por el Ingeniero Miguel Pinto Aparicio tutor de la UNAD.

Este módulo parte de la edición 2009 con el planteamiento de tres unidades fundamentales, microprocesadores, microcontroladores y sus aplicaciones, pero se estructura un nuevo contenido enfocado en lograr un aprendizaje de los conceptos básicos y fundamentales respecto a los microprocesadores, microcontroladores, sus principales familias y aplicaciones enmarcados en un aprendizaje efectivo y práctico en la programación, incorporando temas nuevos, referentes a microcontroladores PIC de Microchip, Motorola Freescale, Texas, Basic Stamp, Intel y Atmel (AVR Arduino) junto con ejemplos de aplicación práctica en lenguaje assembler y programación básica con microcontroladores PIC, HC08, MSP430, TMS, BASIC Stamp I, MCS-51 y AVR-Arduino. Se presenta ejemplos de programación avanzada en manejo y control de periféricos con microcontroladores PIC, Texas Instruments, Motorola Freescale y módulos microcontrolados Basic Stamp II y Arduino. Este módulo ha sido desarrollado en Julio de 2013 por el Ingeniero y Especialista Héctor Uriel Villamil González. HECTOR URIEL VILLAMIL GONZALEZ, se ha desempeñado como tutor de la UNAD en el CEAD de CHIQUINQUIRA, desde el año 2007 y se desempeña actualmente como director nacional de curso.

En esta versión del módulo el Ingeniero Jairo Luis Gutierrez, tutor del CEAD José Celestino Mutis, apoyó el proceso de revisión de estilo del módulo y dio

aportes disciplinarios, didácticos y pedagógicos en el proceso de acreditación de material didáctico desarrollado en el mes de JULIO de 2013.



INTRODUCCIÓN

Los microprocesadores y microcontroladores representan la maravilla del desarrollo de la tecnología electrónica en más de medio siglo, los aparatos que los incorporan han cambiado la forma de trabajar e investigar de la humanidad, en la historia ninguna herramienta creada por el hombre, tenía la capacidad de crear otras y acelerar su evolución, en la actualidad muchos instrumentos, electrodomésticos y en general cualquier dispositivo electrónico utiliza alguno de estos dos componentes para optimizar su funcionamiento.

Los nuevos dispositivos traen consigo nuevas tecnologías aplicadas tanto a la electrónica del hardware como al desarrollo del software que lo controla, se relacionan entonces varios componentes que parten del microprocesador, tal es el caso del microcontrolador, que también es objeto de estudio de este curso. No solo encontramos microprocesadores y microcontroladores, también existen evoluciones de estos aplicados a situaciones particulares en la industria y consumo, tal es el caso de los DSP (Procesadores Digitales de Señales) y PLC (Controladores Lógicos Programables).

Las aplicaciones de control, medición, instrumentación, entretenimiento y consumo, son las promotoras y fuente del creciente mercado tanto de microprocesadores como de microcontroladores. El panorama es alentador por la expectativa de nuevos productos, donde las aplicaciones están limitadas por el ingenio y la imaginación de los programadores.

UNIDAD 1

Nombre de la Unidad	MICROPROCESADORES
Introducción	<p>En esta unidad se presenta el microprocesador como dispositivo que ha definido la revolución tecnología de los últimos años. Con el desarrollo de cada lección se encamina al estudiante por la evolución histórica de este dispositivo, pasando por su estructura, arquitectura, unidades funcionales, sistemas numéricos, para llegar a tomar como puntos de referencia algunos dispositivos particulares categorizados en su arquitectura desde los 8 bits hasta los 64 bits. Finalmente, se toca el tema de la programación con assembler y algunos ejemplos para aplicados buscando el desarrollo de habilidades y competencias específicas en la temática expuesta.</p>
Justificación	<p>La temática expuesta es de gran importancia en las áreas de ingeniería electrónica y afines como sistemas, telecomunicaciones, industrial, audio entre otras, es de notar que los microprocesadores hacen parte de toda la tecnología empleada y como profesionales de las áreas mencionadas es conveniente profundizar en su estudio, aplicación y programación.</p>

Intencionalidades Formativas	Con esta Unidad se pretende que el estudiante conozca en profundidad los microprocesadores, su construcción, arquitectura, evolución, aplicaciones y programación.
Denominación de capítulo 1	MICROPROCESADORES Y MICROCOMPUTADORES
Denominación de Lección 1	Invención e historia del microprocesador y el microcomputador
Denominación de Lección 2	Bases numéricas, códigos binarios, bits y bytes
Denominación de Lección 3	Circuitos lógicos y Dispositivos Digitales básicos
Denominación de Lección 4	Arquitectura, funcionamiento, organización de la memoria, Registros, segmentos, instrucciones y Modos de direccionamiento en un Microcomputador
Denominación de Lección 5	El microprocesador
Denominación de capítulo 2	FAMILIAS DE MICROPROCESADORES
Denominación de Lección 6	Principales Familias de Microprocesadores
Denominación de Lección 7	Microprocesadores de 8 bits
Denominación de Lección 8	Microprocesadores de 16 bits
Denominación de Lección 9	Microprocesadores de 32 bits
Denominación de Lección 10	Microprocesadores de 64 bits
Denominación de capítulo 3	LENGUAJES DE PROGRAMACION EN LOS MICROPROCESADORES
Denominación de Lección 11	Fundamentos de programación en assembler para microprocesadores
Denominación de Lección 12	Diagrama de flujo o bloques y subrutinas
Denominación de Lección 13	Programación con Debug y ensamblador
Denominación de Lección 14	Programación con Simuproc y MASM
Denominación de Lección 15	Ejemplos de aplicación

UNIDAD 2

Nombre de la Unidad	MICROCONTROLADORES
Introducción	<p>La Segunda Unidad de este curso, está enfocada al tema de los microcontroladores donde el estudiante adquiere los conocimientos básicos de construcción, arquitectura interna y diferencias con los microprocesadores, como punto cumbre de esta unidad se presentan dos de las principales familias de microcontroladores exponiendo sus características más relevantes junto con sus conjuntos de instrucciones con lo que el estudiante contara con las herramientas necesarias para empezar a programar estos dispositivos producto de la evolución de los microprocesadores y que son utilizados para aplicaciones específicas de control.</p>
Justificación	<p>Igual como se justifica la unidad 1, el estudio de la unidad 2 está dispuesto para la Ingeniería Electrónica y profesiones afines, donde con estos dispositivos se buscan aplicaciones de control robusto para infinidad de proyectos que dependen o están limitados por el ingenio del diseñador y por el problema a resolver.</p>
Intencionalidades Formativas	<p>Con el desarrollo de esta Unidad y teniendo muchos puntos y temas de apoyo heredados de la unidad anterior se enfoca a los estudiantes en el concepto de microcontrolador y sus diferencias con el microprocesador, pasando por la arquitectura, disposición de pines, direccionamiento, registros y set de instrucciones, estableciendo las bases para hacer las primeras pruebas e implementación de proyectos simples.</p>
Denominación de capítulo 4	INTRODUCCIÓN A LOS MICROCONTROLADORES
Denominación de Lección 16	Generalidades de los microcontroladores

Denominación de Lección 17	Arquitectura interna y direccionamiento		
Denominación de Lección 18	Sistemas microcontrolados: ensamblador y programación en microcontroladores		
Denominación de Lección 19	Familias de microcontroladores		
Denominación de Lección 20	Herramienta y sistemas de desarrollo para microcontroladores		
Denominación de capítulo 5	MICROCONTROLADORES PIC DE MICROCHIP		
Denominación de Lección 21	Microcontroladores PIC		
Denominación de Lección 22	Microcontroladores PIC16F84		
Denominación de Lección 23	Ejemplos de programación con PIC16F84		
Denominación de Lección 24	Microcontroladores PIC 16F877		
Denominación de Lección 25	Ejemplos de programación con PIC16F877		
Denominación de capítulo 6	MICROCONTROLADORES MOTOROLA FREESCALE , TEXAS, BASIC STAMP Y ARDUINO		
Denominación de Lección 26	Microcontroladores Motorola Freescale		
Denominación de Lección 27	Microcontrolador MC68H(R)C908/JL3/JK3/JK1 y Ejemplos de aplicación		
Denominación de Lección 28	Microcontroladores Texas MSP430		
Denominación de Lección 29	Microcontrolador Texas y LaunchPad MSP430 Ejemplos de aplicación		
Denominación de Lección 30	Módulos Microcontrolados Arduino y BASIC Stamp		

UNIDAD 3

Nombre de la Unidad	PROGRAMACION Y DESARROLLO DE PROYECTOS CON MICROPROCESADORES Y MICROCONTROLADORES
Introducción	Esta Unidad ratifica su importancia pues

	está enfocada en la implementación práctica de sistemas basados en microcontroladores en este caso especial de microcontroladores PIC, mostrando varias estrategias y modelos de programación de gran utilidad en muchas aplicaciones, con estos conceptos y ejemplos se refuerza el aprendizaje y se desarrollan habilidades en la programación de este dispositivo, con lo que permite tener capacidad de programar cualquier otro en la misma familia o diferente familia.
Justificación	Tras dos unidades donde se exploran las bases fundamentales de los microprocesadores, los microcontroladores y su programación, esta unidad tiene por objetivo lograr desarrollar habilidades y competencias en el ejercicio de programación para llegar al diseño, desarrollo e implementación basada en soluciones con dispositivos de control.
Intencionalidades Formativas	Lograr profundidad en el aprendizaje y desarrollo de competencias y habilidades que motiven el diseño de soluciones que beneficien su desempeño laboral y profesional.
Denominación de capítulo 7	DISEÑO Y DESARROLLO DE PROYECTOS CON MICROCONTROLADORES Y MICROPROCESADORES
Denominación de Lección 31	Principios Fundamentales en Electrónica y circuitos
Denominación de Lección 32	Periféricos para microprocesadores y microcontroladores
Denominación de Lección 33	Diseño de la solución de hardware
Denominación de Lección 34	Diseño de la solución de software
Denominación de Lección 35	Integración del hardware y software en la solución de control
Denominación de capítulo 6	PROGRAMACION BASICA
Denominación de Lección 36	Subrutinas, llamados y Ramificaciones en programas con ensamblador
Denominación de Lección 37	Operaciones de entrada/salida con

	ensamblador
Denominación de Lección 38	Manejo de interrupciones y timers
Denominación de Lección 39	Exploración de teclado matricial con ensamblador
Denominación de Lección 40	Manejo de display 7 segmentos y pantallas LCD con ensamblador
Denominación de capítulo 9	PROGRAMACION AVANZADA
Denominación de Lección 41	Comunicación entre Microcontroladores
Denominación de Lección 42	Manejo de ADC, PWM y módulos especiales
Denominación de Lección 43	Manejo de memorias internas y externas
Denominación de Lección 44	Programación con microcontroladores Basic Stamp
Denominación de Lección 45	Programación con microcontroladores Arduino

UNIDAD 1 MICROPROCESADORES

CAPITULO 1: MICROPROCESADORES Y MICROCOMPUTADORES

Introducción

En este capítulo se presenta aspectos generales de la tecnología de microprocesadores, comenzando con una presentación breve de los aspectos históricos más relevantes que tienen que ver con la evolución tecnológica hasta la invención del microprocesador, se expone la cronología de los microprocesadores y su evolución con respecto a la instrucciones por segundo (IPS) que son capaces de ejecutar. El capítulo continúa presentando los primeros microprocesadores con las correspondientes empresas que los desarrollaron y sus características más relevantes. Se establecen la conceptualización fundamental y aspectos técnicos generales que tiene que ver con la organización de una computadora, un microcomputador y el microprocesador, para lo cual se plantean temas y conceptos implícitos en la temática específica del microprocesador y su operación.

Lección 1: Invención e historia del microprocesador y el microcomputador

En los comienzos de la implementación de circuitos lógicos el diseñador debía poseer amplios conocimientos tanto en lógica digital como en los componentes y dispositivos que debía acoplar, esto presentaba inconvenientes, la llamada lógica

cableada no permitía hacer modificaciones sin afectar la construcción física del circuito, en 1970 esto cambia con la aparición del microprocesador, con lo que aparece lo que se denomina Lógica programable, la cual permite modificar el comportamiento lógico digital del circuito sin tener que cambiar su configuración física. Partiendo de las técnicas digitales consolidadas en los años sesenta, se comienza a desarrollar y profundizar en el estudio de las técnicas y desarrollo de aplicaciones para los microprocesadores junto con la programación en lenguaje máquina y la programación en assembler. Con esta breve introducción es hora de iniciar una exploración por los momentos históricos que han sido claves para la invención, evolución y desarrollo del microprocesador.

Primeros dispositivos para realizar cálculos

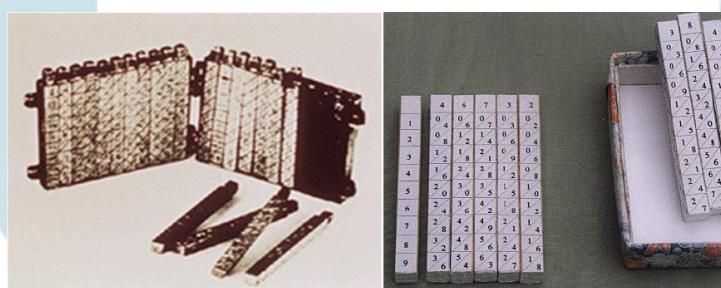
El dispositivo más reconocido es el Abaco, Figura 1, su origen se pierde en el tiempo algunos estiman cerca de 3000 años A.C otros entre 600 y 500 A.C en Egipto o China, su efectividad ha superado el pasar de los años.

Figura 1. Ábaco romano.

Reconstrucción hecha por el RGZ Museum en Mainz, 1977. El original es de bronce y está en manos de la Biblioteca Nacional de Francia, en Paris¹



Figura 2. Los “Naiper Bones”²



¹ Extraído el 10 de Julio de 2012 desde <http://es.wikipedia.org/wiki/Archivo:RomanAbacusRecon.jpg>

² Extraído el 10 de Julio de 2012 desde http://helmutsy.homestead.com/files/computacion/Historia/historia_computadores.htm

A comienzos del siglo XVI, el sistema decimal desplazó al sistema romano en la realización de cálculos complejos, John Naiper (1550-1617), matemático escocés, descubre los logaritmos y construye las primeras tablas de multiplicar. En la figura 2. Se observan algunas tablas de multiplicar de Naiper o “Naiper Bones”.

Dispositivos mecánicos

Basados en los logaritmos, surge las primeras máquinas analógicas de cálculo derivadas de prototipos construidos por Edmund Gunter (1581-1626), matemático y astrónomo inglés y William Oughtred (1574-1660). En la Figura 3., se observa una regla de cálculo o “Slide-rule”, basada en logaritmos.

Figura 3. Los “Slide-rule” o regla de calculo ³



El filósofo y matemático francés Blaise Pascal (1623-1662) construye la primera máquina mecánica llamada “Pascalina”, Figura 4. Su funcionamiento se basaba en engranajes y ruedas con capacidad de sumar.

Figura 4. Pascalina⁴



La pascalina se perfeccionó en una máquina llamada “Calculadora Universal”, Figura 5. Desarrollada por Gottfried Leibniz (1646-1716), tenía la capacidad de realizar las cuatro operaciones aritméticas básicas. El inventor alemán Gottfried von Leibniz (1646-1716) mejora la máquina inventada por Blaise Pascal e inventa en 1671 la primera calculadora de propósito general denominada la “calculadora

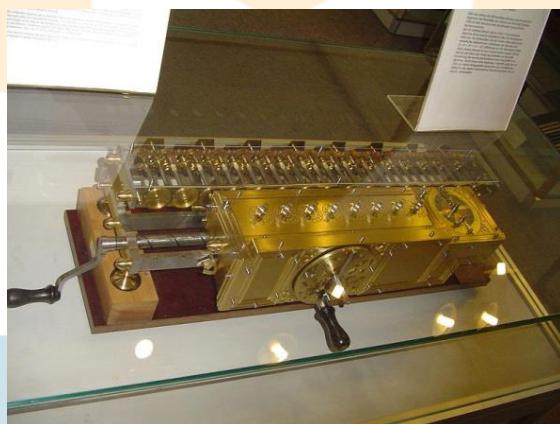
³ Extraído el 10 de Julio de 2012 desde http://es.wikipedia.org/wiki/Archivo:Slide_rule_12.jpg

⁴ Extraído el 10 de Julio de 2012 desde http://es.wikipedia.org/wiki/Archivo:Arts_et_Metiers_Pascaline_dsc03869.jpg

universal” que podía realizar las operaciones de suma, resta, multiplicación, división y extraer raíz cuadrada.

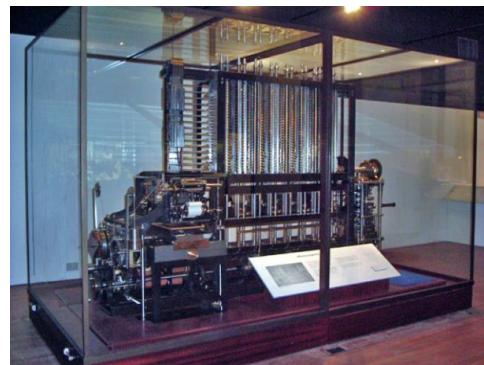
Figura 5. Calculadora universal.

También conocida como Stepped Reckoner o máquina de Leibniz).⁵



Otros avances significativos fueron la calculadora de 12 dígitos (1774) construida por Philipp-Matthaus Hahn; el telar de Jacquard (1801) controlado por tarjetas perforadas para reproducir patrones de tejidos, por Joseph Jacquard; el Arithmometer (1820), primera calculadora construida a nivel industrial, ideada por el francés Xavier Charles Thomas de Colmar (1785-1870).

Figura 6. La réplica de la máquina diferencial del Museo de Ciencias de Londres⁶



Charles Babbage (1791-1871), inventa una maquina llamada “Máquina de Diferencias o Máquina de Babbage”, Figura 6, este inventor británico elaboró los principios de la computadora digital moderna y junto con la matemática británica

⁵ Extraido el 10 de Julio de 2012 desde <http://es.wikipedia.org/wiki/Archivo:Leibnitzrechenmaschine.jpg>

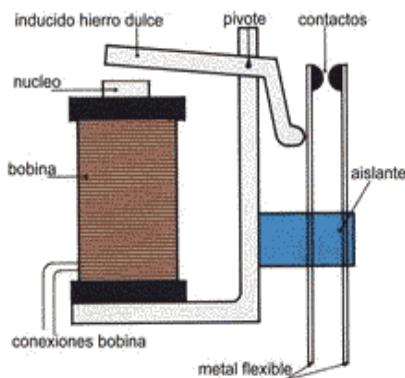
⁶ Extraído el 10 de Julio de 2012 desde http://es.wikipedia.org/wiki/Archivo:050114_2529_difference.jpg

Augusta Ada Byron (1815-1852), quien es reconocida como la primera programadora, se consideran los inventores de la computadora digital moderna. Babbage también ideó la “Maquina Analítica” (1833), como una máquina de cálculo general aunque no se pudo construir debido a la complejidad y limitaciones tecnológicas de la época.

Era Electromagnética

Con el surgimiento de la electricidad y dispositivos como el motor eléctrico, los contactos eléctricos y dispositivos electromagnéticos como los relay o relevos o contactores, ver Figura 7. Los cuales al paso de la corriente eléctrica abren o cierran contactos permitiendo el bloqueo o paso de la corriente con lo que se puede representar dos estados estables, encendido y apagado, que en lógica digital son el uno (1) y cero (0), estos estados junto con el trabajo realizado por George Boole (1815-1864), con su teoría de lógica booleana llamada “El álgebra de Boole”, son la base matemática en la lógica de los computadores modernos.

Figura 7. Relevo o relay⁷



En este lapso se destaca la maquina “Tabuladora” o “Maquina de Hollerith” (1886), que registra datos utilizando contactos eléctricos. Hollerith creó una empresa para la distribución de esta máquina, esta empresa en 1924 pasó a llamarse “International Business Machines” o IBM.

Claude Elwood Shannon (1916-2001), en su tesis de maestría del MIT, plantea y demuestra como el álgebra de Boole puede ser utilizada en circuitos eléctricos de conmutación, haciendo extensivo estos conceptos a futuros diseños de circuitos digitales y computadoras. Las maquinas que utilizaban motores eléctricos para movilizar mecanismos basados en la calculadora de Blaise Pascal dominaron el

⁷ Extraído el 10 de Julio de 2011 desde <http://electricidadmectaronica.blogspot.com/2008/10/conceptos.html>

procesamiento de la información hasta la aparición de la “Z3” (1941) creada por Konrad Zuse, Figura 3., como la primera máquina programable y automática construida con 2300 relés, frecuencia de reloj de aproximadamente 5Hz y longitud de palabra de 22 bits, realizaba cálculos con aritmética de punto flotante en sistema binario.

Figura 8. Z3 (1941)⁸



El primer computador electromecánico denominado “Mark I” (1944), construido por Howard H. Aiken, tenía 760.000 ruedas y 800 Kilómetros de cable, su funcionamiento se basa en la maquina analítica de Charles Babbage.

Era Eléctrica

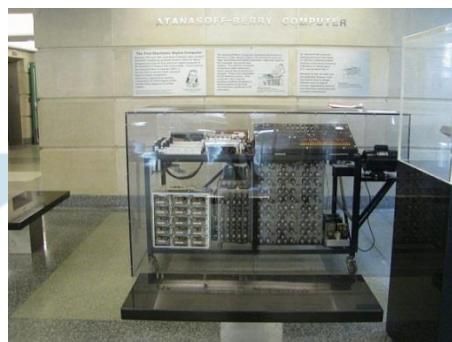
La invención del tubo de rayos catódicos o CTR (1897) por Karl Braun (1850-1918) y el diodo de válvula de vacío (1904) por Jhon A. Fleming que permite el mejoramiento en las comunicaciones, preceden la invención que define la tecnología del microprocesador, se habla del tubo al vacío (1906) por Lee de Forest (1873-1961), basado en el efecto Edison, este dispositivo es un diodo de válvula de vacío con una rejilla de control adicional creando la válvula de tres electrodos, con este dispositivo se hace posible la radio, la telefonía, la telegrafía inalámbrica, la televisión, etc. Impulsando también el desarrollo de la electrónica.

Colossus fue el primer dispositivo calculador y sistema de cómputo primitivo que uso tubos al vacío, utilizado por los británicos para descifrar comunicaciones alemanas durante la segunda guerra mundial, de este se fabricaron varios modelos que mejoraban el desempeño del anterior, llegando a utilizar 4200 válvulas.

⁸ Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Archivo:Z3_Deutsches_Museum.JPG

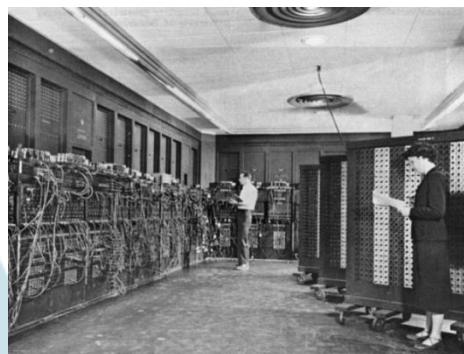
Atanasoff Berry Computer (ABC), Figura 9., primer computador electrónico y digital, construido por John Vincent Atanasoff (1903-1995) y Clifford Edward Berry (1918-1963), su relevancia radica en que aporta nuevos conceptos aplicados como la utilización de un sistema binario para la aritmética y representación de datos, la utilización de medios electrónicos para la realización de las operaciones y la separación de las funciones computacionales del sistema de almacenamiento en una memoria regresiva.

Figura 9. Atanasoff Berry Computer (ABC).⁹



ENIAC (Electronic Numerical Integrator And Computer), Figura 10., primera computadora de propósito general, desarrollada en la Universidad de Pennsylvania por Jhon Presper Eckert y John William Mauchly, completamente digital ejecutaba sus instrucciones en lenguaje máquina, la ENIAC se reprogramaba recableando sus circuitos, lo que tardaba varios días e incluso semanas, construida con más de 17000 válvulas, más de 7000 diodos de cristal, 1500 relés, 70.000 resistencias, 10.000 condensadores, con un peso de 27 toneladas, temperatura de operación de 50°C y consumo de 160 KW.

Figura 10. ENIAC (1946)¹⁰



⁹ Extraído el 10 de Julio de 2011 desde http://es.wikipedia.org/wiki/Archivo:Atanasoff-Berry_Computer_at_Durham_Center.jpg

¹⁰ Extraído el 10 de Julio de 2011 desde <http://es.wikipedia.org/wiki/Archivo:Eniac.jpg>

EDVAC (Electronic Discrete Variable Automatic Computer) o Calculador Discreto Electrónico Automático Variable, Figura 11., su diseño fue anterior al de la ENIAC pero contrario a esta última, la EDVAC era de tipo binaria con un programa diseñado para ser almacenado, las operaciones eran binarias con suma, resta y multiplicación automática y división programada, dentro de sus componentes se encontraban un lector – grabador de cinta magnética, una unidad de memoria, una unidad de control, una unidad para recibir instrucciones de la unidad de control y de la memoria y redirigirlas a otras unidades, una unidad computacional para realizar operaciones aritméticas utilizando un par de números cada vez y almacenando el resultado en memoria. La EDVAC físicamente estaba compuesta de 6000 tubos de vacío y 12000 diodos, en su operación consumía 56 KW de potencia, tenía un peso de 7850 Kg en una superficie de 45,5 m², necesitaba de un personal operativo de 30 personas.

Figura 11. EDVAC (Electronic Discrete Variable Automatic Computer)¹¹



El diseño de la EDVAC se considera un éxito en la historia de la informática y se convirtió en el estándar de la arquitectura para gran parte de las computadoras modernas, fue entregada al laboratorio militar en 1949 y después de varios ajustes puesta en funcionamiento hasta 1951. Se destacan otros dispositivos como el EDSAC (Electronic Delay Storage Automatic Calculator), esta computadora de origen británico es considerada el primer calculador electrónico, con órdenes internas, puso en funcionamiento sus primeros programas en mayo de 1949. La Máquina Experimental de Pequeña Escala de Manchester SSEM (Manchester Small-Scale Experimental Machine), considerado el primer computador en el mundo con programa almacenado, ejecutó el primer programa el 21 de junio de 1948. Computadoras como la Pilot ACE (Automatic Computing Engine) Construida en el Reino Unido, estaba construida con 800 tubos de vacío con un reloj base de

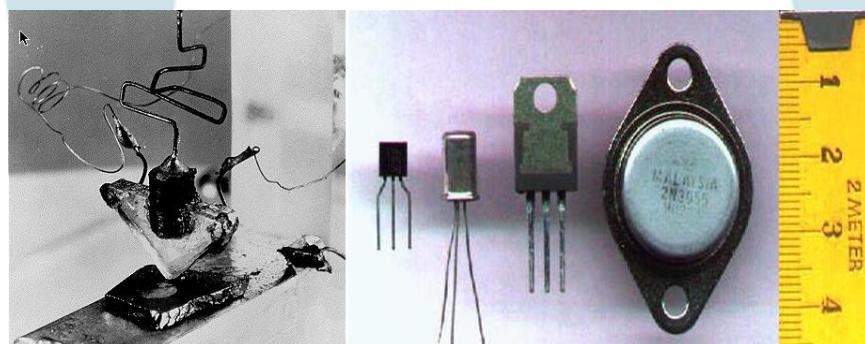
¹¹ Extraído el 10 de Julio de 2011 desde <http://es.wikipedia.org/wiki/Archivo:EDVAC.png>

1MHz considerado uno de los más rápidos para las primeras computadoras, entró en funcionamiento en 1951. Finalmente la UNIVAC I (Universal Automatic Computer I), es considerada la primera computadora comercial fabricada en los estados unidos, pesaba 7250 Kg, compuesta por 5000 tubos de vacío con capacidad de ejecutar 1000 cálculos por segundo, tenía un reloj interno de 2,25 MHz, la primera UNIVAC se entregó a la oficina de censos de Estados Unidos en marzo de 1951.

Era Electrónica

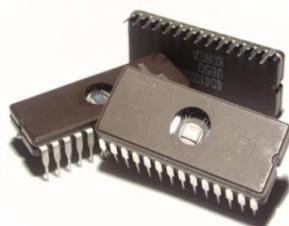
La revolución del transistor comienza con su desarrollo e invención en los laboratorios de Bell Telephone en 1948, su utilización comenzó hasta 1950, para la fabricación de aparatos de radio, televisión, sonido, militar y la naciente tecnología de computadores. En la Figura 12. Se observan varios transistores.

Figura 12. Transistor (1948)¹² junto con transistores modernos.



En 1958 se desarrolla el Circuito Integrado, por el Ingeniero Jack Kilby (1923-2005), consistía en un dispositivo de germanio que integraba seis transistores en una misma base semiconductora formando un oscilador de rotación de fase.

Figura 13. Circuito Integrado (1958)¹³



¹² Extraído el 10 de Julio de 2011 desde <http://es.wikipedia.org/wiki/Archivo:Transistor-photo.JPG>

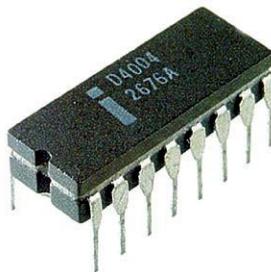
¹³ Extraído el 10 de Julio de 2011 desde <http://es.wikipedia.org/wiki/Archivo:Microchips.jpg>

El circuito integrado consiste en una pastilla pequeña de unos cuantos milímetros de área, construida a partir de silicio en la que se fabrican circuitos electrónicos basados en circuitos semiconductores, estos dispositivos se protegen del exterior por una cubierta de cerámica o plástico y se comunica al exterior por medio de unos contactos metálicos, llamados pines, patillas o contactos. En la Figura 13., se puede observar circuitos integrados con ventana para borrado de memoria por luz ultravioleta.

El desarrollo del transistor, conduce al desarrollo de la primera familia lógica RTL (Lógica Resistencia Transistor), con la utilización de tecnologías de integración surge la tecnología de circuitos integrados digitales. Robert Noyce después de renunciar a Fairchild en 1968 en compañía de Gordon Moore y Andrew Grove, con el respaldo económico de Arthur Rock funda la compañía INTEL donde se dan los primeros pasos para el desarrollo del microprocesador, comenzando por el desarrollo de una memoria basada en semiconductores y la lógica de un Flip-Flop como elemento de memoria capaz de almacenar un bit, se crea el primer circuito de memoria RAM llamado 1103 (1969) con una capacidad de 1024 bits.

Finalmente aparece el primer microprocesador el Intel 4004, puede observarlo en la Figura 14., lanzado al mercado en 1971, con una CPU de 4 bits, 2300 transistores, 16 pines (DIP-Dual In-line Package), máxima velocidad de reloj 740 KHz, con arquitectura Harvard, 46 instrucciones, 16 registros de 4 bits cada uno, stack de 3 niveles. En 1972 se lanza al mercado el Intel 8008, empleaba direcciones de 14 bits, direccionando hasta 16KB de memoria, con la limitante de 18 pines en DIP, tiene un bus compartido de datos y direcciones de 8 bits, velocidad 0,5 a 0,8 MHz.

Figura 14. Circuito Integrado del microprocesador Intel 4004 (1971)¹⁴



El primer “PC” o computador personal se construye en IBM y se lanza al mercado en 1981, aunque en el mercado ya existían otros computadores fabricados

¹⁴ Extraído el 10 de Julio de 2011 desde http://es.wikipedia.org/wiki/Archivo:Intel_4004.jpg

principalmente por Apple, el modelo IBM PC, mostrado en la Figura 15., causa gran impacto debido al concepto de compatibilidad, es decir, la estrategia de fabricar computadores con partes de distintos fabricantes con compatibilidad IBM, en su interior tenía un microprocesador 8088 de Intel.

Figura 15. IBM PC (1981)¹⁵



Primeros microprocesadores

Intel (Integrated Electronics Corporation): Empresa fundada en 1968, comienza como una empresa fabricante de semiconductores y su éxito comienza con la producción de un conjunto de chips encargados por la empresa BUSICOM para una línea de calculadoras electrónicas programables (1969), como resultado de este trabajo el grupo Intel diseño y fabricó el primer procesador en un chip, el Intel 4004 (i4004) entre 1970 – 1971, junto con un conjunto de chips de soporte, la denominada familia 4004:

- 4001: ROM de 256 bytes
- 4002: RAM 40 bytes
- 4003: Registro de desplazamiento de salida paralela de 10 bits
- 4008: Latch de 8 bits de dirección
- 4009: programa y convertidor de acceso a chips y memoria estándar I/O

El sucesor del Intel 4004, es el Intel 4040 que fue lanzado al mercado en 1974. El 4040 fue usado principalmente en juegos, pruebas, desarrollo, y equipos del control. El paquete del 4040 era más de dos veces el ancho del 4004 y tenía 24 pines en lugar de los 16 del 4004. El 4040 agregó 14 instrucciones, un espacio más grande para la pila (7 niveles en vez de 3), un espacio para programas de 8KB, 8 registros adicionales, y habilidades de interrupción.

Zilog Inc, fundada en California en 1974 por Federico Faggin, quien trabajó perfeccionando el primer microprocesador de Intel, el Intel 4004. Zilog nace como

¹⁵ Extraído el 10 de Julio de 2011 desde http://es.wikipedia.org/wiki/Archivo:IBM_PC_5150.jpg

una empresa fundada por personal que trabajo para Intel en el 8080, construyen el Z-80 que incorpora un conjunto de instrucciones más extenso y compatible con el 8080. El Z-80 presenta mejoras respecto al 8080 entre las cuales están dos bancos de registros diseñados para dar respuesta rápida a las interrupciones, un conjunto de instrucciones mejorado que incluye instrucciones de movimiento de bloques y búsqueda de bytes, instrucciones para manipulación de bits, un conjunto de direcciones para refrescar la DRAM integrada, alimentación de 5 Voltios, menos circuitos auxiliares para generación de señales de reloj y para el enlace con la memoria y dispositivos de E/S. El Z-80 tenía un precio más bajo que el 8080 lo que permite su difusión en el mercado y desplaza al Intel 8080. El Tandy TRS-80 Model 1 es el primer computador que utiliza el Z-80 con 1,77 MHz y 4 KB de RAM como núcleo de sistema. Entre otros dispositivos que utilizaron el Z-80 como procesador central se encuentran las videoconsolas Coleco, Sega Master System y Sega Game Gear. Con la aparición en el mercado de los procesadores de 16 bits el Z-80 se convierte en procesador secundario en videoconsolas y tarjetas de expansión para las primeras IBM PC.

Motorola, es fundada en 1928, se ha especializado en el desarrollo de dispositivos electrónicos y de telecomunicaciones, adopta la marca comercial “Motorola” en 1930 cuando desarrolla la primera radio para automóvil, en 1940 produce el primer walkie-talkie como radio Handie-Talkie AM utilizado por el ejército de los Estados Unidos. En la misma época del 8080 Motorola desarrolla el microprocesador de 8 bits MC6800 (6800) en 1975, mostrado en la Figura 16., con aproximadamente 6800 transistores, 78 instrucciones, un registro índice, alimentación única de 5 Voltios, el 6800 normalmente se fabricaba en un encapsulado DIP de 40 pines, su diseño es completamente distinto al 8080 pero tiene características similares. Varios de los primeros microordenadores de los años 1970, que usualmente eran vendidos por correo (en piezas sueltas o ensamblados), usaron el 6800 como procesador principal. Entre ellos se encuentran el SWTPC 6800 (el primero en usarlo) y el MITS Altair 680.

Figura 16. Motorola 6800¹⁶



¹⁶ Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Archivo:Motorola_MC6800L_SC7718I_top.jpg

Motorola perfecciona el 6800 hacia el 6809 considerado uno de los mejores microprocesadores de la época aunque no tuvo éxito comercial, pero fue usado en el sistema de videojuego Vectrex y en el ordenador Tandy TRS-80, entre otros. El 6502 derivado el 6800 logró posicionarse como pieza clave en la fabricación de las primeras computadoras personales PET de Commodore y Apple II de Apple Computer Inc.

AMD (Advanced Micro Devices Inc), comienza a producir chips en 1969, en 1975 empieza con el desarrollo de memorias RAM, este mismo año introduce un clon del 8080 de Intel utilizando ingeniería inversa, produjo los procesadores Bit slicing como el Am2901, Am29116, Am293xx usados en algunos diseños de microcomputadoras, su trabajo se redujo a procesadores embebidos logrando cierto éxito en los 80's con el AMD 29k que era un procesador con arquitectura RISC, al no lograr el éxito esperado se concentró en el desarrollo de memorias Flash y procesadores Intel, comenzando a desarrollar procesadores como el 386 que superaba ampliamente las prestaciones del Intel 80x386, siguió con el 486 para comenzar fabricando su primer procesador propio AMD K5.

Cyrix, comienza a operar en 1988 como proveedor de coprocesadores matemáticos para el 286 y 386, la compañía fue fundada por Jerry Rogers y ex trabajadores de Texas Instruments, al igual que AMD comienza clonando procesadores de Intel comenzando con el 386 denominado por Cyrix 486SLC y 486 DLC lanzados en 1992 situando su rendimiento entre un 386 y 486 de Intel, siguen con los 486 compatibles con los de Intel, Cx5X86 compatible con Pentium y el 6x86 primer microprocesador de su línea en superar las prestaciones de Intel Pentium.

Organización y operación de una computadora

Como se observa desde el inicio de esta lección la evolución de las computadoras digitales han pasado por diferentes etapas y tecnologías, desde los grandes y costosos equipos utilizados por entidades gubernamentales, militares y grandes empresas hasta llegar a una nueva forma de computador digital basado en el circuito integrado (CI) llamado microprocesador.

Un circuito integrado (CI), es un componente electrónico construido en una misma pastilla de silicio de algunos milímetros cuadrados de área, compuesto por cientos, miles y en la actualidad millones de dispositivos como transistores, diodos, resistencias, condensadores, conductores, etc. Existe una categorización para la

escala de integración de un circuito integrado (CI) en términos de la cantidad o número de puertas lógicas o compuertas lógicas, que son la construcción física de una expresión lógica del álgebra de Boole, tales como la AND, OR, NOT, etc.

- SSI (Small Scale Integration), integración a baja escala, realizada en los primeros circuitos integrados, como las compuertas lógicas. Se construyen con unos pocos transistores hasta unos cientos de transistores, menos de 10 puertas lógicas.
- MSI (Medium Scale Integration), integración a media escala, que contiene cientos de transistores en un mismo chip, más de 10 y menos de 100 puertas lógicas.
- LSI (Large Scale Integration), escala de integración grande, con esta tecnología se logra la construcción del primer microprocesador de 4 bits en 1971 con 2300 transistores, pasando luego en 1974 a 8 bits con 8000 transistores, más de 100 puertas lógicas y menos de 1000.
- VLSI (Very Large Scale Integration), integración de escala muy grande, los cuales contienen millones de transistores con los que se implementan circuitos para tareas de control y procesamiento de información. Utilizado para la fabricación de microprocesadores, microcontroladores y FPGA. Contiene más de 1000 y menos de 10000 puertas lógicas.
- ULSI (Ultra Large Scale Integration), Ultra alta escala de integración, utilizada en procesadores digitales, microprocesadores avanzados y FPGA. Con más de 10000 puertas lógicas y menos de 100000 puertas lógicas.
- GLSI (Giga Large Scale Integration), Gran escala de integración, con más de 100000 puertas lógicas.

La fabricación de un microprocesador mediante un circuito integrado forma la denominada CPU (Unidad Central de Procesamiento) que es un sistema digital que se encarga de la transmisión y procesamiento de información, representada por señales físicas discretas generalmente en dos estados estables. Las computadoras y microcomputadoras utilizan un programa almacenado, este programa representa el algoritmo. La microcomputadora se compone básicamente de un microprocesador y una unidad de memoria. La computadora digital está compuesta de cinco unidades básicas, la unidad de entrada, la unidad de memoria (programa y datos), la unidad de salida y la unidad aritmética y de control, esta última unidad está dispuesta en la CPU, el microprocesador es la construcción física de la CPU. Estas unidades constituyen el hardware, para que el hardware funcione se debe preparar una lista de instrucciones denominada programa el cual se almacena en la unidad de memoria de programa con el objeto de decirle a la CPU que hacer y manipular la información presentada como datos.

La lista de instrucciones o programa se denomina Software el cual se puede almacenar de forma temporal o permanente (firmware) en la memoria de programa. Las instrucciones son tan simples y se denotan según el origen de su fabricante, por ejemplo, para los primeros microprocesadores que fueron fabricados en Estados Unidos se tiene instrucciones como sumar (ADD), transferir datos (MOVE), Introducir o sacar datos (INPUT - OUTPUT), generar saltos de programa (JUMP) o hacer llamados a pequeños subprogramas (CALL).

Organización y operación de una microcomputadora

Una microcomputadora es una computadora digital, pero más pequeña y de bajo costo, contiene las cinco unidades básicas de una computadora digital, una unidad de entrada, una unidad de salida, una unidad de memoria (RAM y ROM), la unidad aritmética y la unidad de control. Las dos últimas constituyen la denominada CPU que físicamente está contenida dentro del microprocesador.

La unidad de memoria se compone de una memoria ROM o memoria de solo lectura, físicamente es una pastilla de semiconductor en circuito integrado (CI) en la cual se graba de forma permanente el programa (firmware), la memoria RAM o memoria de lectura / escritura o memoria de acceso aleatorio, generalmente de tipo volátil (se borra en ausencia de alimentación eléctrica), también es un dispositivo integrado (CI) separado de la memoria ROM. La memoria RAM se utiliza para el almacenamiento de los datos y programas temporales de usuario.

La comunicación entre los diferentes circuitos integrados (CI) que componen el sistema de la microcomputadora digital, como son los chips para puertos de entrada y salida, el chip del microprocesador y chips de memoria RAM y ROM, se hace mediante tres canales, uno que se encarga del control de las diferentes unidades llamado bus de control, uno encargado de direccionar las posiciones de memoria o dispositivos de entrada y salida (E/S o I/O) llamado bus de direcciones y uno encargado de enviar los datos desde el microprocesador hacia y desde la memoria y dispositivos I/O, llamado bus de datos.

Los MIPS y MFLOPS

El acrónimo MIPS proviene de millones de instrucciones por segundo, es una forma de medir la capacidad de procesamiento de los microprocesadores con el mismo juego de instrucciones, las comparativas presentan valores picos por lo que

no son muy realistas. El Acrónimo de MFLOPS proviene de millones de operaciones en coma flotante por segundo, también es utilizado como forma de medir la capacidad de procesamiento. Para estas pruebas se utiliza el benchmark linpack que expresa los resultados en MFLOPS referidas a operaciones de suma y multiplicación de doble precisión (64 bits).

Benchmark es una técnica utilizada para medir el rendimiento de un sistema informático o cualquiera de sus componentes y efectuar una comparación entre máquinas de características similares, un benchmark puede hacerse a nivel de máquina mediante la ejecución de una serie de programas de manera que se pueda estimar el tiempo empleado para el proceso medido en MIPS o MFLOPS, un benchmark también puede hacerse para comparar el rendimiento de un software con respecto a otro. Se está utilizando las especificaciones SPECint de SPEC (Standard Performance Evaluation Corporation) para medir el rendimiento y potencia de procesamiento, más específicamente se utiliza la SPECfp para medir el rendimiento procesos de coma flotante del computador.

En la siguiente Tabla 1. Se expone la evolución de los microprocesadores en el tiempo siguiendo el comportamiento de los MIPS.

Tabla 1. Evolución en el tiempo de las instrucciones por segundo¹⁷.

Procesador	IPS	Reloj	Año
Intel 8080	640 KIPS	2 MHz	1974
Intel 8086	800 KIPS	4.77 MHz	1974
Motorola 68000	1 MIPS	8 MHz	1979
Intel 486DX	54 MIPS	66 MHz	1992
PowerPC 600s (G2)	35 MIPS	33 MHz	1994
ARM 7500FE	35.9 MIPS	40 MHz	1996
PowerPC G3	525 MIPS	233 MHz	1997
ARM10	400 MIPS	300 MHz	1998
Zilog eZ80	80 MIPS	50 MHz	1999
Sony "Allegrex"(de la PSP)	32 MIPS	333MHz	2002
Pentium 4 Extreme Edition	9726 MIPS	3.2 GHz	2003

¹⁷ Extraído el 10 de Julio de 2011 desde <http://es.wikipedia.org/wiki/MIPS>

ARM Cortex A8	2000 MIPS	1.0 GHz	2005
Xbox360 IBM "Xenon" Single Core	6400 MIPS	3.2 GHz	2005
AMD Athlon 64	8400 MIPS	2.8 GHz	2005
AMD Athlon FX-57	12000 MIPS	2.8 GHz	2005
AMD Athlon 64 Dual Core	18500 MIPS	2.2 GHz	2005
AMD Athlon 64 3800+ X2 (Dual Core)	18900 MIPS	2.2 GHz	2005
Overclocked AMD Athlon 64 3800+ X2 (Dual Core)	25150 MIPS	2.8 GHz	2005
Cell (cada PPE)	6400 MIPS	3.2 GHz	2006
Procesador Cell de la PlayStation 3	21800 MIPS	3.2 GHz	2006
AMD Athlon FX-60 (Dual Core)	22150 MIPS	2.6 GHz	2006
Overclocked AMD Athlon FX-60 (Dual Core)	24300 MIPS	2.8 GHz	2006
Overclocked AMD Athlon FX-60 (Dual Core)	27100 MIPS	3.0 GHz	2006

Cronología de los microprocesadores

- 1971: Intel 4004. Fue el primer microprocesador comercial. Salió al mercado el 15 de noviembre de 1971.
- 1972: Intel 8008
- 1974: Intel 8080, Intel 8085, SC/MP de National Semiconductor.
- 1975: Signetics 2650, MOS 6502, Motorola 6800
- 1976: Zilog Z80
- 1978: Intel 8086, Motorola 68000
- 1979: Intel 8088
- 1982: Intel 80286, Motorola 68020
- 1985: Intel 80386, Motorola 68020, AMD80386, VAX 78032
- 1987: Motorola 68030
- 1989: Intel 80486, Motorola 68040, AMD80486
- 1993: Intel Pentium, Motorola 68060, AMD K5, MIPS R10000, AIM PowerPC 601
- 1994: AIM PowerPC 620
- 1995: Intel Pentium Pro
- 1996: AMD K-5
- 1997: Intel Pentium II, AMD K6, PowerPC G3, MIPS R120007
- 1998: Intel Pentium II Xeon, AMD K6-2
- 1999: Intel Pentium III, Intel Celeron, Intel Pentium III Xeon, AMD Athlon k-7, PowerPC G4
- 2000: Intel Pentium 4, Intel Itanium 2, AMD Duron, MIPS R14000
- 2001: AMD Athlon XP
- 2003: PowerPC G5
- 2004: Intel Pentium M, Intel Pentium 4, AMD Athlon 64
- 2005: Intel Pentium D, Intel Extreme Edition con hyper threading, Intel Core Duo, AMD Athlon 64, AMD Athlon 64 X2, AMD Sempron 128.

- 2006: Intel Core 2 Duo, Intel Core 2 Extreme, AMD Athlon FX
- 2007: Intel Core 2 Quad, AMD Opteron Quad Core, AMD Quad FX, AMD Phenom
- 2008: Intel Core 2 Extrem, AMD Phenom X4, Athlon II X2, AMD Phenom II
- 2009: Intel Core i7, i7 Extrem, AMD Phenom II (tres y cuatro núcleos)
- 2011: Intel Core (segunda generación), AMD Fusion

Lección 2: Bases numéricas, códigos binarios, bits y bytes

En esta lección se plantean conceptos generales, que son útiles y necesarios para el entendimiento del funcionamiento interno del microprocesador y microcontrolador. Los temas tratados exploran y retoman conceptos y procedimientos claves en la programación de microprocesadores y microcontroladores.

Bases numéricas

Una base numérica está compuesta por una serie de símbolos que pueden dar sentido y significado a una cifra dentro de un sistema numérico, para la mayoría de las personas el sistema numérico con el que están familiarizados es el sistema numérico decimal. El sistema numérico decimal es un sistema posicional, compuesto por 10 símbolos o guarismos diferentes 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9, que definen la base numérica decimal (base 10), con la cual se puede representar cualquier número del sistema.

Base numérica decimal: la forma de interpretar los números decimales es a través de su posición o lugar que ocupa el dígito en el número total, que se pueden relacionar con las denominaciones de unidades, decenas, centenas, etc. Desde un punto de vista analítico, el sistema decimal tiene 10 símbolos representativos (0,1,2,3,4,5,6,7,8,9), para representar símbolos como “12” se usa una combinación de los símbolos mencionados.

Cálculo posicional: El principio del cálculo posicional puede hacerse extensivo a los demás sistemas numéricos mediante la siguiente ecuación:

$$\sum_{n=0}^{\infty} D_n \cdot B^n$$

Donde **n** es la posición del número comenzando desde cero e incrementando con enteros positivos en la posición desde el punto decimal hacia la izquierda y con enteros negativos desde el punto decimal a la derecha, **D** es el número y **B** representa la base numérica de la cifra a convertir, en el caso decimal sería “10”.

Ejemplo: El decimal “125” podría representarse con punto decimal así, “125.00” entonces tomando desde el punto decimal hacia la izquierda y comenzando desde cero, el número cinco (5) tendrá la posición cero (0), el numero dos (2) la posición uno (1) y finalmente el número uno(1) la posición (2), siguiendo lo establecido, la ecuación será:

$$\sum_{n=0}^{\infty} D_n \cdot B^n = 5_0 \times 10^0 + 2_1 \times 10^1 + 1_2 \times 10^2 = 5 \times 1 + 2 \times 10 + 1 \times 100 = 5+20+100=125$$

Base numérica Binaria: La base numérica binaria es un sistema de numeración donde se utilizan dos guarismos identificados como cero (0) y uno (1), con los que se representa cualquier valor, estos guarismos son conocidos como dígitos binarios o bits. Esta base numérica es utilizada como mecanismo de representación de información en sistemas digitales electrónicos en los que se representan los dos estados con niveles diferenciados de voltaje o corriente, usualmente una tensión BAJA es representada por el bit o digito binario cero (0) y una tensión ALTA es representada por el bit o digito uno (1).

Ejemplo: son números binarios: 10010111.011₂, 101011₂, 0.1011₂.

Base numérica Hexadecimal: La base Hexadecimal (abreviado Hex) tiene 16 dígitos o guarismos que van del 0 al 9 y de la letra A hasta la F que representan los números del 10 al 15. Esta base numérica es ampliamente utilizada en el área de la informática, electrónica digital y ciencias de la computación, puesto que se utiliza el octeto o byte (conjunto de 8 bits o números binarios), como unidad básica de memoria, donde dos dígitos hexadecimales corresponden a exactamente ocho números binarios, es decir, una byte o un octeto. Entonces cuatro dígitos binarios equivalen a un número hexadecimal o un hexadecimal equivale a cuatro dígitos binarios.

Ejemplo: son números hexadecimales: A09F.0CB₁₆, 1D3E96₁₆, 0.A0D5₁₆.

El sistema Octal: El sistema octal también es utilizado en la representación de información y datos en el microprocesador, se compone de ocho dígitos o guarismos (0,1,2,3,4,5,6,7) los cuales siguen una regla similar a la de los

hexadecimales la cual establece que tres dígitos binarios equivalen a un dígito octal y un octal a tres binarios.

Ejemplo: son números Octales: 1346.253_8 , 6431_8 , 0.245_8 .

Conversión entre distintas bases

En el aprendizaje del funcionamiento y programación de microprocesadores y microcontroladores, es importante desarrollar destreza en la conversión entre distintas bases numéricas, específicamente, en las siguientes.

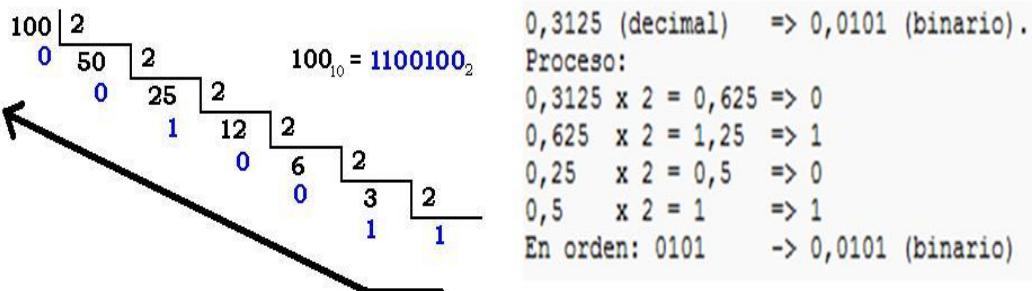
Convertir un número decimal a binario: Existen dos maneras de convertir un número de decimal a binario. La primera es utilizar una tabla de equivalencias de binario a decimal siguiendo las potencias de 2 (dos). La segunda es válida para convertir de decimal a cualquier base numérica y consiste en dividir sucesivamente la parte entera del decimal entre la base a la cual se quiere convertir, tomando sus módulos (residuos) y el último resultado. La parte decimal del número se multiplica por la base y se toma el número entero como resultado parcial, la fracción que pueda quedar se multiplica nuevamente por la base.

Ejemplo: En la *Figura 17*. Se ilustra el procedimiento para convertir un número decimal a su equivalente binario, se ilustra el procedimiento de convertir el número $100,3125_{10}$ a su equivalente binario, observe como se toma la parte entera 100_{10} y la parte decimal $0,3125_{10}$, para desarrollar la conversión a binario utilizando dos procesos distintos. El subíndice indica la base actual del número, es decir, el número $100,3125$ es base diez. Los dígitos son cada una de las cifras que componen un número en cualquier sistema numérico, para el sistema binario los dígitos binarios hacen referencia a las cifras individuales representadas por un “uno” o un “cero” que en conjunto forman un número binario.

La parte entera 100_{10} se convierte a binario mediante divisiones sucesivas por “2” (base binaria), cada residuo y último resultado es una cifra o dígito binario, en conjunto forman el número binario “1100100” que equivale al número decimal “100” o 100_{10} . La parte decimal $0,3125_{10}$, se convierte a binario mediante multiplicaciones sucesivas por “2” (base binaria), donde del resultado obtenido se toma y separa el número entero para tomar la parte decimal que se vuelve a multiplicar por “2” (base binaria), hasta lograr un número entero o la cantidad de cifras de precisión requeridas por el usuario, de esta forma el número “0,3125”

equivale al binario “0,0101”. Entonces como resultado el número $100,3125_{10}$ equivale al número $1100100,0101_2$.

Figura 17. Conversión decimal-binario¹⁸



Convertir un binario a decimal: El sistema binario se basa únicamente en dos condiciones: encendido (1) o apagado (0), por lo tanto su base es 2, la fórmula $\sum_{n=0}^{\infty} D_n \cdot B^n$ es aplicable a la conversión de cualquier base a la base decimal cada posición de los dígitos binarios representa un valor particular de “2” elevado a la potencia “n”. Es importante tener claro que el dígito ubicado más a la derecha es llamado dígito menos significativo (LSB ó de menor valor posicional numérico) y el dígito ubicado más a la izquierda es el más significativo (MSB ó de mayor valor posicional numérico).

Ejemplo: Convertir el número binario 101101_2 a su equivalente decimal. Tomando la fórmula $\sum_{n=0}^{\infty} D_n \cdot B^n$, se debe tener las siguientes consideraciones. “D” representa el número, “n” la posición del mismo y “B” la base numérica, en este caso “2”. Tomando la anterior ecuación y reemplazando tenemos:

$$\begin{aligned} \sum_{n=0}^{\infty} D_n \cdot B^n &= 1_5 \times 2^5 + 0_4 \times 2^4 + 1_3 \times 2^3 + 1_2 \times 2^2 + 0_1 \times 2^1 + 1_0 \times 2^0 \\ \sum_{n=0}^{\infty} D_n \cdot B^n &= 1 \times 32 + 0 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 32 + 0 + 8 + 4 + 0 + 1 = 45_{10} \end{aligned}$$

El número 101101_2 equivale a 45_{10}

Ejemplo: Convertir el número binario 10010111.011_2 (el subíndice “2” representa la base, en este caso binaria) a decimal. Colocar el número binario como 10010111.011 , y asignar los números que identifican la posición de cada dígito, entonces se obtiene:

$$\begin{aligned} \sum_{n=0}^{\infty} D_n \cdot B^n &= 1_7 \times 2^7 + 0_6 \times 2^6 + 0_5 \times 2^5 + 1_4 \times 2^4 + 0_3 \times 2^3 + 1_2 \times 2^2 + 1_1 \times 2^1 + 1_0 \times 2^0 + \\ &\quad 0_{-1} \times 2^{-1} + 1_{-2} \times 2^{-2} + 1_{-3} \times 2^{-3} \end{aligned}$$

¹⁸ Extraído el 10 de Julio de 2011 desde http://es.wikipedia.org/wiki/Sistema_binario

$$= 1 \times 128 + 0 \times 64 + 0 \times 32 + 1 \times 16 + 0 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1 + 0 \times 1/2 + 1 \times 1/4 + 1 \times 1/8 \\ = 128 + 0 + 0 + 16 + 0 + 4 + 2 + 1 + 0 + 1/4 + 1/8 = 151.375_{10}$$

Conversión entre binarios y hexadecimales: La conversión entre números binarios y hexadecimales, es sencilla notando que cuatro (4) dígitos binarios consecutivos equivalen a un hexadecimal y cada hexadecimal equivale a cuatro (4) binarios, esta agrupación de cuatro dígitos binarios se debe comenzar desde la posición menos significativa desde el punto separador de enteros y fracciones a la izquierda o a la derecha.

Ejemplo: Convertir el numero 11011111010.00111110_2 a su equivalente hexadecimal. Siguiendo el procedimiento establecido se forman grupos de cuatro (4) dígitos binarios tomando del punto decimal a la izquierda para la parte entera y del punto decimal a la derecha para la fracción, se pueden agregar ceros para completar los grupos en los extremos en caso de ser necesario, en el presente ejemplo se agrega un cero “0” al grupo del extremo izquierdo y se sigue la conversión presentada en la *Figura 18*. Para obtener el resultado.

[0110][1111][1010].[0011][1110]₂
 6 F A . 3 E₁₆

Figura 18. Tabla de Conversión entre números decimales, octales y hexadecimales¹⁹

0 _{hex} = 0 _{dec} = 0 _{oct}	0 0 0 0
1 _{hex} = 1 _{dec} = 1 _{oct}	0 0 0 1
2 _{hex} = 2 _{dec} = 2 _{oct}	0 0 1 0
3 _{hex} = 3 _{dec} = 3 _{oct}	0 0 1 1
4 _{hex} = 4 _{dec} = 4 _{oct}	0 1 0 0
5 _{hex} = 5 _{dec} = 5 _{oct}	0 1 0 1
6 _{hex} = 6 _{dec} = 6 _{oct}	0 1 1 0
7 _{hex} = 7 _{dec} = 7 _{oct}	0 1 1 1
8 _{hex} = 8 _{dec} = 10 _{oct}	1 0 0 0
9 _{hex} = 9 _{dec} = 11 _{oct}	1 0 0 1
A _{hex} = 10 _{dec} = 12 _{oct}	1 0 1 0
B _{hex} = 11 _{dec} = 13 _{oct}	1 0 1 1
C _{hex} = 12 _{dec} = 14 _{oct}	1 1 0 0
D _{hex} = 13 _{dec} = 15 _{oct}	1 1 0 1
E _{hex} = 14 _{dec} = 16 _{oct}	1 1 1 0
F _{hex} = 15 _{dec} = 17 _{oct}	1 1 1 1

El sistema Octal: La conversión entre el sistema binario y octal es similar al utilizado en la conversión entre binarios y hexadecimales, la Tabla 2. Abarca los primeros 7 números binarios, cada número está compuestos de 3 dígitos binarios y a su vez correspondiente a un dígito en el sistema octal.

¹⁹ Extraído el 10 de Julio de 2011 desde http://es.wikipedia.org/wiki/Sistema_hexadecimal

Tabla 2. Conversión entre binarios y octales.

Decimal	Binario	Octal
0	000	0
1	001	1
2	010	2
3	101	3
4	100	4
5	101	5
6	110	6
7	111	7

Ejemplo: Convertir el numero 11011111010.00111110_2 a su equivalente octal. Siguiendo el procedimiento establecido se forman grupos de tres (3) dígitos binarios tomando del punto decimal a la izquierda para la parte entera y del punto decimal a la derecha para la fracción, se pueden agregar ceros para completar los grupos en los extremos en caso de ser necesario, en el presente ejemplo se agrega un cero “0” al grupo del extremo izquierdo y el extremo derecho, se sigue la conversión presentada en la *Tabla 2* para obtener el resultado.

$[011][011][111][010].[001][111][100]_2$
 3 3 7 2 . 1 7 4 8

Complementos en números binarios

Los complementos son utilizados en las computadoras digitales para simplificar la operación de sustracción y manipulación lógica. En todas las bases numéricas, donde “r” representa la base existen dos complementos, el complemento de “r” y el complemento de $(r-1)$. Para la base decimal se tiene el complemento 10 y complemento 9. Para la base binaria se tiene el complemento 2 y complemento 1.

Complemento r: Si se tiene un numero positivo “N” en la base “r” con una parte entera de “n” dígitos, el complemento “r” de “N” se define como $r^n - N$ para $N \neq 0$ y 0 para $N = 0$.

Ejemplo: El complemento 10 de $(2940)_{10}$ es $10^4 - 2940 = 10000 - 2940 = 7060$. El complemento 10 de un número decimal puede obtenerse dejando todos los ceros menos significativos sin cambio, el primer digito no cero menos significativo se le resta 10 y a los siguientes dígitos más significativos se le resta 9.

Ejemplo: El complemento 2 de $(100100)_2$ es $2^6 - 100100 = 1000000 - 100100 = 011100$. El complemento 2 de un numero binario se forma dejando todos los ceros menos significativos sin cambio hasta el primer “1”, en los siguientes dígitos se cambian los “1” por “0” y “0” por “1”.

Complemento (r-1): Si se tiene un numero positivo “N” en la base “r” con una parte entera de “n” dígitos y una parte fraccionaria de “m” dígitos, el complemento “r-1” de “N” se define como $r^n - r^{-m} - N$.

Ejemplo: El complemento 9 de $(2940)_{10}$ es $10^4 - 10^0 - 2940 = 10000 - 1 - 2940 = 7059$. El complemento 9 de un número decimal puede obtenerse restando 9 a cada digito.

Ejemplo: El complemento 1 de $(100100)_2$ es $2^6 - 2^0 - 100100 = 1000000 - 1 - 100100 = 11011$. El complemento 1 de un número binario se forma cambiando los “1” por “0” y “0” por “1”.

Códigos binarios

En los sistemas electrónicos digitales se utilizan señales con dos valores distintos o señales binarias y elementos de circuitos con dos estados estables, existe una equivalencia directa entre las señales binarias, los elementos de circuito binario y el digito binario. Un “bit”, por definición es un dígito binario (0-cero o 1-uno). Los sistemas digitales además de representar y manipular números binarios, pueden representar cualquier elemento de información discreta mediante un código binario (conjunto de bits). Un número binario de “n” dígitos o bits, puede representar 2^n elementos distintos.

Ejemplo: Si se tiene un grupo de cuatro elementos o cantidades distintas se puede representar mediante un código de 2 bits, a cada cantidad o elemento se le asigna una de las siguientes combinaciones de bits: 00, 01, 10 y 11.

Al proceso de asignar a cada cantidad o elemento un valor o representación en bits se le conoce como *codificación binaria*. Existen varios códigos binarios que se diseñan con características especiales que permiten el control de errores o facilitan operaciones lógicas o aritméticas, pero el ejercicio de elegir o generar una codificación para un proyecto en particular depende del profesional que diseña el sistema electrónico digital. En los siguientes párrafos se tratan algunos de los códigos más usuales.

Códigos decimales: Son códigos que permiten representar los diez dígitos decimales mediante una combinación de bits, son numerosos los códigos diferentes que se obtienen al ordenar cuatro o más bits en distintas combinaciones. Recordar que con cuatro bits puedo obtener 16 combinaciones distintas, porque $2^4 = 16$. En la *Tabla 3*. Se presentan algunos de los códigos binarios para dígitos decimales utilizados en sistemas basados en microprocesadores y microcontroladores.

Código BCD: BCD proviene de las iniciales Binary-Coded Decimal o decimal codificado en binario. Es una asignación directa del equivalente en binario, si a cada uno de los dígitos binarios se le asignan respectivamente los pesos 8,4,2,1 se obtiene rápidamente la equivalencia independiente para cada dígito decimal.

Ejemplo: Para representar el número 125_{10} en código BCD recurrimos a la *Tabla 3* y se obtiene 0001 0010 0101. Observe que no se coloca subíndice debido a que el número no es un equivalente binario, en este caso es una codificación binaria, también puede observar que para facilidad de comprensión en la operación realizada se ha dejado espacios entre cada grupo de cuatro bits. Para el primer dígito decimal “1” la codificación BCD es “0001”, utilizando los pesos asignados se comprueba que, $0001=0x8+0x4+0x2+1x1=1$.

No debe confundirse la conversión y la codificación, son dos operaciones diferentes. 125_{10} equivale al binario 1111101_2 mientras que al codificarlo se tiene que 125_{10} codificado en BCD es 0001 0010 0101.

Tabla 3. Códigos binarios para dígitos decimales

Decimal	BCD	Exceso 3	84-2-1	Gray o Reflejado	2421	Biquinario 5043210
0	0000	0011	0000	0000	0000	0100001
1	0001	0100	0111	0001	0001	0100010
2	0010	0101	0110	0011	0010	0100100
3	0011	0110	0101	0010	0011	0101000
4	0100	0111	0100	0110	0100	0110000
5	0101	1000	1011	0111	1011	1000001
6	0110	1001	1010	0101	1100	1000010
7	0111	1010	1001	0100	1101	1000100
8	1000	1011	1000	1100	1110	1001000
9	1001	1100	1111	1101	1111	1010000

Código Exceso-3: Es un código sin pesos (sin valor posicional), puesto que su valor se obtiene del código BCD sumando el valor “ 3_{10} ” o “ 011_2 ”. Es un código autocomplementario, es decir el complemento a 9 se obtiene fácilmente cambiando los números “1” por “0” y los “0” por “1”.

Ejemplo: Para comprobar el funcionamiento del código Exceso-3, se toma el número 257_{10} , su complemento 9 es 742_{10} . Al codificar el número 257_{10} a código exceso-3 se obtiene, 0101 1000 1010, al hallar el complemento 1 del anterior código binario resulta 1010 0111 0101 y al decodificarlo siguiendo el código Exceso-3 se tiene, $1010=7$, $0111=4$ y $0101=2$ que corresponde a 742_{10} que como se comprueba es el complemento 9 de 257_{10} .

Código 84-2-1: Este código tiene los pesos o valores posicionales 8,4,-2,-1. Es un código autocomplementario, es decir, se obtiene fácilmente el complemento a 9.

Ejemplo: El dígito 3_{10} se codifica en código 84-2-1 como 0101 puesto que $0101=0x8+1x4+0x(-2)+1x(-1)=0+4+0-1=3$. Si se cambia los “1” por “0” y los “0” por “1”, se obtiene 1010, decodificando en código 84-2-1 es, 6_{10} . El complemento 9 de 3_{10} es 6_{10} .

Código 2421: Este código tiene los pesos o valores posicionales 2,4,2,1. También es un código autocomplementario, es decir, se obtiene fácilmente el complemento a 9.

Ejemplo: El dígito 7_{10} se codifica en código 2421 como 1101 puesto que $1101=1x2+1x4+0x2+1x1=2+4+0+1=7$. Al hallar el complemento “1” de la anterior codificación se tiene 0010, que al decodificarlo en código 2421 es, 2_{10} . El complemento 9 de 7_{10} es 2_{10} .

Código Biquinario: Fue uno de los códigos utilizados en los primeros computadores como el Colossus, es un código de siete bits con pesos 5,0,4,3,2,1,0, también tiene propiedades de detección de error, puesto que cada dígito decimal se compone de cinco dígitos binarios “0” y dos dígitos binarios “1”, en una transmisión de datos el receptor puede verificar cuantos “1” y “0” hay en cada codificación y determinar si cumple o no con la característica del código biquinario.

Código reflejado: En este código los valores consecutivos difieren solamente en uno de los dígitos binarios. En un comienzo los circuitos lógicos digitales utilizaban contactores o switches electromagnéticos, que generaban picos de ruido con el

cambio de varios bits, el código se diseñó para prevenir estas señales espurias, por lo que el código facilita la corrección de errores. El código reflejado, también llamado código Gray es utilizado para codificar la información analógica representada por cambios continuos en la posición del eje, en mecanismos de giro, para el diseño de mapas de Karnaugh, utilizados en el diseño de circuitos combinacionales y secuenciales, ocasionalmente se utiliza en algoritmos genéticos. En la *Tabla 4.* Se observa el código reflejado de 4 bits. Para codificar un valor binario a código gray se realiza una compuerta XOR con el mismo número binario aplicando un desplazamiento a la izquierda.

Ejemplo: Se tiene el número binario 1001 equivalente al decimal 9, al aplicar la XOR al mismo número con un desplazamiento a la izquierda se obtiene,

$$\begin{array}{r} 1 \ 0 \ 0 \ 1 \\ 1 \ 0 \ 0 \ \oplus \\ \hline 1 \ 1 \ 0 \ 1 \end{array}$$
, es decir, el código gray del número binario 1001 es 1101.

Ejemplo: Para codificar el decimal 3_{10} utilizando el código reflejado o Gray con cuatro bits, se procede haciendo la conversión a binario, $3_{10} = 0011_2$ haciendo la

operación: $\begin{array}{r} 0 \ 0 \ 1 \ 1 \\ 0 \ 0 \ 1 \ \oplus \\ \hline 0 \ 0 \ 1 \ 0 \end{array}$, por tanto 3_{10} codificado a Gray es “0010”.

Tabla 4. Código reflejado o Gray de 4 bits

Código Gray	Equivalente Decimal
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15

Códigos Alfanuméricos

Es un código binario para un grupo de elementos compuestos de diez dígitos decimales, las 26 letras del alfabeto y símbolos especiales como &,\$,(,), entre otros.

Tabla 5. Tabla ASCII y tabla ASCII extendida²⁰

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr		Dec	Hx	Oct	Html	Chr		Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040	 	Space		64	40	100	@	Ø		96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!		65	41	101	A	A		97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"		66	42	102	B	B		98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#		67	43	103	C	C		99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	\$		68	44	104	D	D		100	64	144	d	d
5	5	005	EMQ	(enquiry)	37	25	045	%	%		69	45	105	E	E		101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&		70	46	106	F	F		102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	!		71	47	107	G	G		103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	((72	48	110	H	H		104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051))		73	49	111	I	I		105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*		74	4A	112	J	J		106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+		75	4B	113	K	K		107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,		76	4C	114	L	L		108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-		77	4D	115	M	M		109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.		78	4E	116	N	N		110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/		79	4F	117	O	O		111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	Ø		80	50	120	P	P		112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1		81	51	121	Q	Q		113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2		82	52	122	R	R		114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3		83	53	123	S	S		115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4		84	54	124	T	T		116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5		85	55	125	U	U		117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6		86	56	126	V	V		118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7		87	57	127	W	W		119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8		88	58	130	X	X		120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9		89	59	131	Y	Y		121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:		90	5A	132	Z	Z		122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;		91	5B	133	[[123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<		92	5C	134	\	\		124	7C	174	|	
29	1D	035	GS	(group separator)	61	3D	075	=	=		93	5D	135]]		125	7D	175	}	}`
30	1E	036	RS	(record separator)	62	3E	076	>	>		94	5E	136	^	^		126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?		95	5F	137	_	_		127	7F	177		DEL

Source: www.LookupTables.com

Source: www.LookupTables.com

²⁰ Extraído el 10 de Enero de 2012 desde <http://www.asciiitable.com/>

El llamado código interno utiliza 6 bits para su codificación, para este código alfanumérico se requiere codificar un mínimo de $10+26=36$ elementos distintos, por tanto se requieren un mínimo de seis bits que pueden representar $2^6 = 64$ elementos distintos (con 5 bits no es suficiente $2^5 = 32$). El código ASCII (American Standard Code for Information Interchange), mostrado en la Tabla 5., es un código alfanumérico de 7 bits más un bit de paridad para el control de errores, este código tiene una representación amplia de caracteres incluyendo los diez dígitos decimales, las letras del alfabeto (mayúsculas y minúsculas) y caracteres especiales de control para la transmisión de información digital (espacio, (,+,\$,*,),-/, ?), no debe confundirse con los códigos generados por teclado al oprimir ALT + numero, tampoco con los caracteres definidos bajo el estándar ISO-8859-1 que utiliza 8 bits y posee en su rango inicial la codificación ASCII. El código EBCDIC (Extended BCD Interchange Code), es un código de 8 bits con el que se representa caracteres alfanuméricos, caracteres de control y signos de puntuación, existen varias versiones del código EBCDIC.

Ejemplo: Es fácil encontrar en internet e incluso en librerías tablas que relacionan el código ASCII, junto con la combinación de teclado “Alt + ‘numero’” para obtener caracteres especiales como la “Ñ” cuando el teclado no la tiene. Como ejemplos de código ASCII tenemos: “\$”(ASCII = 0100100); “+” (ASCII = 0101011); “6” (ASCII = 0110110); “U” (ASCII = 1010101).

Operaciones Aritméticas binarias

Los microprocesadores y microcontroladores utilizan los números binarios para representar, manipular y procesar información, estos dispositivos requieren realizar operaciones aritmética binarias, por lo que la mayoría tienen instrucciones para realizar la suma o resta de números binarios y los más avanzados y complejos tienen instrucciones para multiplicar y dividir. Los siguientes ejemplos muestran los principios que rigen las operaciones aritméticas con números binarios.

Ejemplo: Sumar los números 11110010 + 10101011.

$$\text{Reglas: } \frac{0}{0}, \frac{+0}{1}, \frac{+1}{10}, \frac{1}{11} \quad \text{Carry=1} \quad \begin{array}{r} 11110010 \\ + \\ \hline 10101011 \\ \hline 10011101 \end{array}$$

Ejemplo: Restar los números 11110010 - 10101011.

$$\begin{array}{r}
 & & & & & & \text{Prestamo 1} \\
 & 0 & 1 & 1 & 0 & & 11110010 \\
 \text{Reglas: } & \frac{-0}{0} ; & \frac{-0}{1} ; & \frac{-1}{0} ; & \underline{\quad -1 \quad} & \text{Carry = 0} & \frac{10101011}{01000111}
 \end{array}$$

Ejemplo: Multiplicar los números $1010 + 101$.

$$\text{Reglas: } \begin{array}{r} 0 \quad 1 \quad 0 \quad 1 \\ \times 0 \quad \times 0 \quad \times 1 \quad \times 1 \\ \hline 0 \quad 0 \quad 0 \quad 1 \end{array} \quad \text{Carry} = 0 \quad \begin{array}{r} 1010 \\ \times 101 \\ \hline 1010 \\ 0000 \\ \hline 110010 \end{array}$$

Sustracción utilizando complementos: Dentro de la CPU la ALU implementa circuitos digitales del tipo medio sumador y sumador completo, para realizar las operaciones de suma, resta, multiplicación y división. Utilizando los complementos se puede realizar la resta de una forma más eficiente, rápida y con circuitos combinacionales digitales simples. El producto es una serie de sumas y la división es una serie de restas.

Ejemplo: Restar $8343 - 235$ utilizando complemento 10. Minuendo = 8343,

Sustraendo = 0235, complemento 10 del sustraendo = 9765. $\text{acarreo} \leftarrow \begin{array}{r} 8343 \\ + 9765 \\ \hline 1 \quad 8108 \end{array}$, como hay acarreo no se modifica el resultado. Respuesta 8108. En caso de no presentarse acarreo se debe hallar el complemento 10 al resultado y anteponer signo negativo.

Ejemplo: Restar $100111 - 1010$ utilizando complemento 2. Minuendo = 100111, Sustraendo = 001010, complemento 2 del sustraendo = 110110.

$\text{acarreo} \leftarrow \begin{array}{r} 100111 \\ + 110110 \\ \hline 1 \quad 011101 \end{array}$, como hay acarreo no se modifica el resultado. Respuesta 011101. En caso de no presentarse acarreo se debe hallar el complemento 2 al resultado y anteponer signo negativo.

Ejemplo: Restar $3967 - 0284$ utilizando complemento 9. Minuendo = 3967,

Sustraendo = 0284, complemento 9 del sustraendo = 9715. $\text{acarreo} \leftarrow \begin{array}{r} 3967 \\ + 9715 \\ \hline 1 \quad 3682 \end{array}$, como hay acarreo se suma 1 al resultado. Respuesta 3683. En caso de no presentarse acarreo se debe hallar el complemento 9 al resultado y anteponer signo negativo.

Ejemplo: Restar $110001 - 111101$ utilizando complemento 1. Minuendo = 110001, Sustraendo = 111101, complemento 1 del sustraendo = 000010.

$\text{acarreo} \leftarrow \begin{array}{r} 110001 \\ + 000010 \\ \hline 0 \quad 110011 \end{array}$, como no hay acarreo se halla el complemento 1 al

resultado y se antepone el signo negativo. Respuesta - 001100. En caso de presentarse acarreo se debe sumar 1 al resultado.

Representación de números negativos: Para representar números negativos en la mayoría de computadoras modernas utilizando el sistema numérico binario, se realiza mediante la técnica denominada complemento a dos, esta técnica se define como el número obtenido después de convertir todos los ceros a unos y todos los unos a ceros, al resultado se le suma uno.

Ejemplo: Representar el numero decimal 125_{10} utilizando el complemento 2. 125_{10} Equivale a 1111101 → complemento a 2 → $0000010 + 0000001 = 0000011$.

Bits, Nibbles, Bytes y Palabras

Un bit es la unidad más pequeña de información, pero en su forma más simple un bit representa un dígito binario, es decir, un “uno” (1) o un “cero” (0), físicamente los bits requieren de un espacio o celda donde almacenar una cantidad de energía definida en dos estados estables que representen los dígitos binarios, esta energía es llamada voltaje, que es la energía necesaria para mover una carga eléctrica (electrón) de un punto a otro. La Unidad Central de Proceso o CPU, se encarga de procesar la información que se encuentra en grupos de celdas denominadas registros, cada registro se compone de una o varias celdas las cuales almacenan una entidad llamada “bit”, en un comienzo las celdas de almacenamiento se fabricaban a partir de flip-flops o también llamados basculadores o biestables, generalmente los registros se disponen en un conjunto de 8 o 16 bits. Los flip-flops tienen la capacidad de almacenar dos niveles de voltaje, uno bajo típicamente 0,5 Volts que es interpretada como cero “0” o apagado y un nivel alto típicamente 5 Volts interpretado como uno “1” o encendido, estos estados son los conocidos como “bit” (Binary digit o dígito binario).

A un grupo de 4 bits se le conoce como Nibble, dos Nibbles u 8 bits se conocen como Byte y dos Bytes o 16 bits se denominan como palabra (Word), existen también, la doble palabra (DW) con 32 bits y la cuádruple palabra (QW) con 64 bits. Como ejemplo de estos registros en los microprocesadores está un registro denominado AX el cual es de 16 bits pero puede ser utilizado como de 8 bits, este registro AX puede representar 65536 números binarios entre 0000000000000000 (2^0) y 1111111111111111 (2^{15}).

Lección 3: Circuitos lógicos y Dispositivos Digitales básicos

Los circuitos lógicos o circuitos digitales están constituidos por componentes con un comportamiento equivalente al de los operadores lógicos booleanos (AND, OR, NOT, etc), estos circuitos pueden ser clasificados en dos grandes categorías o grupos, es recomendable consultar en internet el *datasheet* de los CI que se mencionen. El primer grupo corresponde a los *circuitos lógicos combinacionales*, compuestos por puertas digitales en circuitos construidos con chips individuales (74LS00, 74LS08, 74LS32, etc) o en chips con tareas específicas basados en compuertas como los codificadores (74HC147, multiplexor, etc) y decodificadores (74LS48, Demultiplexor, etc), en este grupo también se incluyen las compuertas tristate (74LS125, 74LS240, etc) y Schmitt Trigger (74LS14, 74LS132, etc). El segundo grupo es el de los *circuitos lógicos secuenciales*, en donde se incluyen los flip-flops (o biestables) que son dispositivos que tienen la característica de permanecer en uno de dos estados estables (0 o 1), durante un tiempo indefinido en ausencia de perturbaciones, lo que le da una característica especial de “memoria”, la interconexión de varios flip-flops puede formar circuitos lógicos secuenciales para conteo (contadores), temporización (timers), secuenciamiento y almacenamiento de datos (circuitos de memoria).

Los codificadores, decodificadores, buffer, tristate y circuitos de memoria son dispositivos digitales básicos utilizados en la tecnología de microprocesadores y microcontroladores, por lo que es necesario su presentación y estudio.

Compuertas lógicas

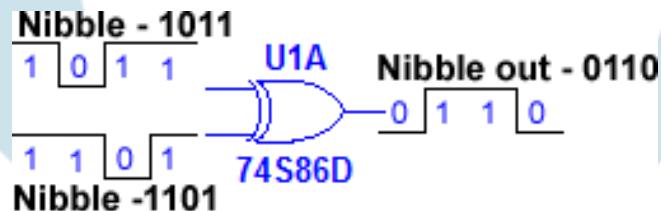
Las compuertas digitales son dispositivos electrónicos digitales utilizados para el procesamiento de señales, para identificarlas se utilizan los símbolos gráficos en esquemas electrónicos, la función lógica en instrucciones, la tabla de verdad en la operación de variables y la expresión booleana que describe la relación entre variable, operador y su resultado (\bar{x} Se lee “no x ” o x complemento o inverso de x . Ver Tabla x6.). En microprocesadores y microcontroladores las compuertas lógicas son utilizadas en instrucciones para la operación y manipulación sobre bits almacenados en registros.

Tabla 6. Funciones y Compuerta lógicas

Función Lógica	Expresión Booleana	Símbolo lógico	Tabla de verdad																								
Inversor	$X = \bar{X}$	 7404N	<table border="1"> <thead> <tr> <th>Entrada</th><th>Salida</th></tr> </thead> <tbody> <tr> <td>X</td><td>\bar{X}</td></tr> <tr> <td>0</td><td>1</td></tr> <tr> <td>1</td><td>0</td></tr> </tbody> </table>	Entrada	Salida	X	\bar{X}	0	1	1	0																
Entrada	Salida																										
X	\bar{X}																										
0	1																										
1	0																										
AND	$A \cdot B = F$	 7408N	<table border="1"> <thead> <tr> <th colspan="2">Entradas</th><th colspan="2">Salidas</th></tr> <tr> <th>A</th><th>B</th><th>AND</th><th>NAND</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	Entradas		Salidas		A	B	AND	NAND	0	0	0	1	0	1	0	1	1	0	0	1	1	1	1	0
Entradas		Salidas																									
A	B	AND	NAND																								
0	0	0	1																								
0	1	0	1																								
1	0	0	1																								
1	1	1	0																								
NAND	$\overline{A \cdot B} = F$	 7400N																									
OR	$A + B = F$	 7432N	<table border="1"> <thead> <tr> <th colspan="2">Entradas</th><th colspan="2">Salidas</th></tr> <tr> <th>A</th><th>B</th><th>OR</th><th>NOR</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	Entradas		Salidas		A	B	OR	NOR	0	0	0	1	0	1	1	0	1	0	1	0	1	1	1	0
Entradas		Salidas																									
A	B	OR	NOR																								
0	0	0	1																								
0	1	1	0																								
1	0	1	0																								
1	1	1	0																								
NOR	$\overline{A + B} = F$	 7402N																									
OR Exclusiva	$A \oplus B = Y$	 74136N	<table border="1"> <thead> <tr> <th colspan="2">Entradas</th><th colspan="2">Salidas</th></tr> <tr> <th>A</th><th>B</th><th>XOR</th><th>XNOR</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>1</td><td>0</td><td>1</td></tr> </tbody> </table>	Entradas		Salidas		A	B	XOR	XNOR	0	0	0	1	0	1	1	0	1	0	1	0	1	1	0	1
Entradas		Salidas																									
A	B	XOR	XNOR																								
0	0	0	1																								
0	1	1	0																								
1	0	1	0																								
1	1	0	1																								
NOR Exclusiva	$\overline{A \oplus B} = Y$	 74HC266D 2V																									

Ejemplo: En una transmisión bit a bit o serial se envía un nibble a cada entrada de una compuerta XOR de dos entradas, como se observa en la siguiente gráfica. El nibble obtenido en la salida se determina como “1011” XOR “1101” = “0110”. Este comportamiento se observa claramente en la Figura 19.

Figura 19. Nibble de salida en compuerta XOR con dos nibbles en sus entradas.

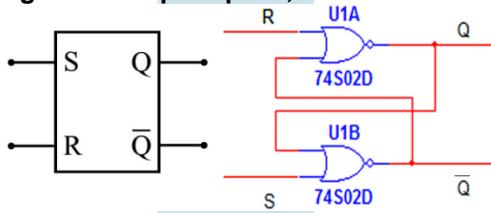


Flip-Flops

Los Flip-Flops son dispositivos biestables, es un multivibrador capaz de permanecer en uno de dos estados estables posibles, durante un tiempo indefinido sin presencia de perturbaciones. Los Flip-Flops funcionan como dispositivos para almacenamiento temporal de información (memoria), el Flip-Flop puede memorizar el valor de sus entradas incluso después de haber sido eliminadas, al agrupar varios Flip-Flops se pueden formar circuitos de temporización, conteo, secuenciamiento y de memoria. Existen varios tipos de Flip-Flops que se encuentran en chips o circuitos integrados (CI) independientes y que son frecuentemente utilizados como dispositivos periféricos o auxiliares en implementaciones, proyectos y soluciones basadas en microprocesadores y microcontroladores.

Flip-Flop RS: Es el primer flip-flop construido con compuertas sin presencia de pulso de reloj para habilitar la trasferencia de datos entre la entrada y salida, este biestable mostrado en la Figura 20., se caracteriza por tener dos entradas **S** (set o ajuste) y **R** (reset o reinicio) y dos salidas **Q** y su complemento \bar{Q} . Al colocar **S** a “1” se ajusta la salida $Q = 1$ (set), al colocar **R** a “1” se reinicia la salida $Q = 0$ (reset).

Figura 20. Flip-Flop RS, construcción con compuertas NOR y tabla de verdad



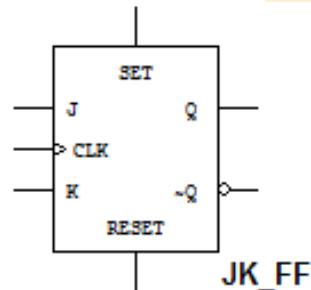
S	R	$Q(t+1)$	Estado
0	0	Q_t	Memoria
0	1	0	Reset
1	0	1	Set
1	1	X	Indeterminado

Q_t representa el estado de salida “Q” antes de activar la entrada, $Q_{(t+1)}$ representa el estado de la salida “Q” después de la activación de la entrada es el estado siguiente a Q_t . El estado “X” o indeterminado es un estado al cual no se debe llegar o prohibido puesto que las salidas no son complementadas.

Flip-Flop JK: Es una evolución del Flip-Flop R-S, similar en operación en las condiciones *set*, *reset* y *mantenimiento de estado*, la diferencia radica en que el Flip-Flop JK no tiene condiciones no validas, es decir, cuando ambas entradas “J” y “K” son “1”. En el Flip-Flop JK mostrado en la Figura 21., se observan varias terminales donde “J” es el grabado o set, “K” es el borrado o reset, “CLK” la entrada de pulso de reloj con transferencia de datos de la entrada a la salida por flanco de subida “↑”, las salidas se denotan por **Q** y su complemento \bar{Q} . Los circuitos integrados con Flip-Flop pueden tener entradas adicionales para

establecer estados en la salida. La entrada “SET” al colocarse en “1” (Voltaje positivo), resulta en $Q = 1$ y $\bar{Q} = 0$, la entrada “RESET” al colocarse en “1” (Voltaje positivo), resulta en $Q = 0$ y $\bar{Q} = 1$. Con el Flip-Flop JK se puede construir los Flip-Flops “D” y “T”.

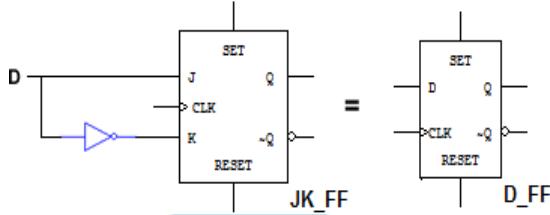
Figura 21. Flip-Flop tipo JK



Modo de operación	Entradas			Salidas	
	D	J	CLK	Q	\bar{Q}
Comutación	1	1	↑	Estado opuesto	
Set (puesta a 1)	1	0	↑	1	0
Reset (puesta a 0)	0	1	↑	0	1
Mantenimiento	0	0	↑	No cambia	

Flip-Flop D: También es llamado Flip-Flop de datos, mostrado en la Figura 22. La entrada de datos es “D”, gobernada por el pulso de reloj “CLK”, la flecha “↑” indica un flanco positivo del pulso de reloj para la transferencia de datos de la entrada a la salida, las salidas se denotan por Q y su complemento \bar{Q} .

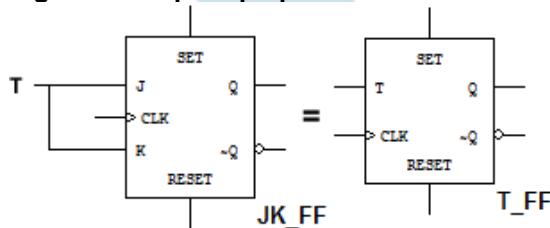
Figura 22. Flip-Flop tipo D



Modo de operación	Entradas		Salidas	
	D	CLK	Q	\bar{Q}
Set (puesta a 1)	1	↑	1	0
Reset (puesta a 0)	0	↑	0	1
Mantenimiento	X	No pulso	Sin cambios	

Flip-Flop T: El Flip-Flop T mostrado en la Figura 23., es un biestable que cambian de estado (toggle) cuando se recibe un pulso de reloj (CLK) mientras la entrada esta puesta a “1” o nivel alto. En la Figura 23 la entrada se designa como “T”, el pulso de reloj “CLK” requiere de un flanco de subida “↑” (transición del pulso de nivel bajo a alto) para la transferencia de datos de la entrada a la salida, las salidas se denotan por Q y su complemento \bar{Q} .

Figura 23. Flip-Flop tipo T



Modo de operación	Entradas		Salidas	
	T	CLK	$Q(t)$	$Q(t+1)$
Comutación	1	↑	0	1
	1	↑	1	0
Mantenimiento	0	↑	Q	\bar{Q}

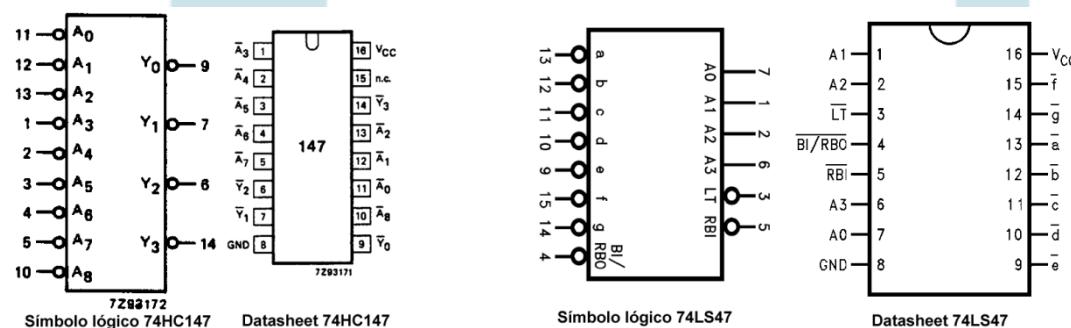
Codificadores y decodificadores

La codificación es el proceso de conversión de un sistema de datos de origen a un sistema de datos de destino, manteniendo la coherencia y equivalencia de la información de origen. Los decodificadores hacen el proceso inverso al codificador.

Codificador: En la Electrónica digital, el Codificador es un dispositivo combinacional con 2^n entradas y "n" salidas, presenta en la salida el código binario de "n" bits correspondientes a las entradas activas. Como ejemplo de un codificador se tiene el codificador decimal a BCD (74HC147).

Decodificador: Es un dispositivo combinacional que tiene una entrada de "n" bits y una salida de "M" líneas (menor o igual a 2^n), las líneas de salida se activan en función de un único código binario de "n" bits. Como ejemplo de un decodificador se tiene el decodificador BCD a 7 segmentos (74LS47), mostrado en la Figura 24..

Figura 24 Codificador 74HC147 y Decodificador 74LS47²¹



Buffers y tristate

En sistemas basados en microprocesadores y microcontroladores es muy frecuente utilizar interfaces electrónicas, que tienen la función de traducir, acoplar, amplificar, adecuar o permitir la comunicación de señales entre secciones del sistema, se utilizan para este fin interfaces como buffers, multiplexores, convertidores analógico-digitales (ADC), convertidores digital-analógico (DAC), acopladores ópticos, entre otros.

Buffers: son circuitos electrónicos que actúan como seguidores de señal, en estos circuitos el voltaje o corriente no disminuye, por tanto compensa la pérdida en voltaje o corriente que tenga la señal y la aadecua para la siguiente sección del sistema. Usualmente se implementan varios circuitos buffer en un mismo circuito

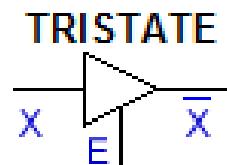
²¹ Extraído el 10 de Julio de 2011 desde <http://www.datasheetcatalog.net>

integrado (CI) para utilizar como dispositivo externo o como parte de la circuitería interna de un microcontrolador. Ejemplo de un circuito integrado Buffer es el 7407 (Hex Buffer with Hi-Volt (30 Volts) Open Collector Output). También se considera como buffer a un espacio de memoria de datos reservada para el almacenamiento temporal de datos.

Dispositivos 3-state: denominados dispositivos tristate o tres estados, son varios los dispositivos electrónicos digitales que presentan el estado ALTO o “1 lógico”, el estado BAJO o “0 lógico” y un estado de alta impedancia (Hi-Z) que hace que el dispositivo 3-state no tenga relevancia en el circuito digital. Los dispositivos 3-state son utilizados por su estado de alta impedancia, cuando se requiere monitorizar señales sin afectar la señal (conversor ADC) o cuando se requiere que varios circuitos comparten el mismo bus o línea de salida.

Buffer 3-state: Los buffer tri-state o 3-state, mostrado en la Figura 25., son dispositivos que permiten compartir un mismo bus o línea de datos con otros circuitos digitales, regenerar la señal y mantenerla en su estado lógico original.

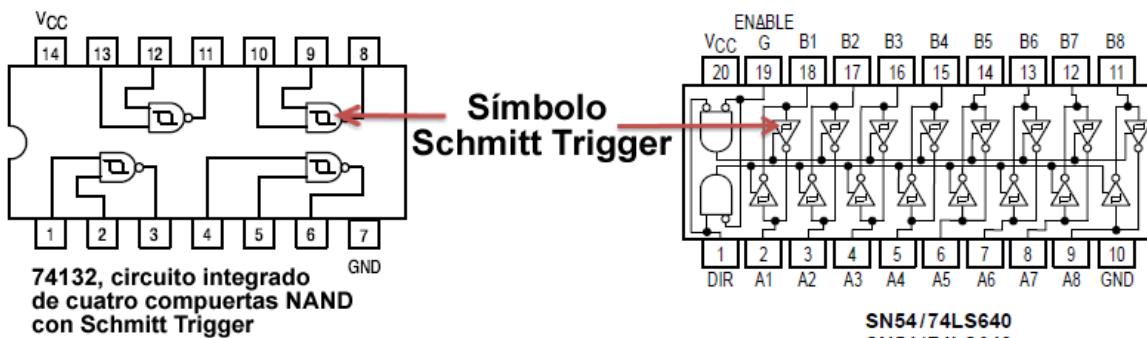
Figura 25. Buffer de tres estados



Modo de operación	Entradas		Salida
	X	E	
Habilitado	0	1	0
	1	1	1
Inhabilitado	0	0	Hi-Z
	1	0	

Schmitt trigger: también conocidos como disparador Schmitt trigger, son circuitos comparadores electrónicos, que utilizan la histéresis o umbrales de conmutación para eliminar ruidos presentes en señales y prevenir cambios de estado en falso en circuitos digitales, causados por los picos de ruido presentes en la señal original o por caídas dentro del estado de indeterminación. Ejemplo de este tipo de circuitos está el circuito 74132 (Quad 2-input NAND Schmitt trigger) y el 74640 (Octal Bus Transceiver with Inverting 3-state Output), mostrados en la Figura 26.

Figura 26. Circuitos integrados con Schmitt Trigger (74132 y 74640)²²



Circuitos de Memoria

Los microprocesadores y microcontroladores utilizan circuitos de memoria internos o externos, para el almacenamiento del programa y los datos generados en el proceso. Estos circuitos de memoria se encuentran como memorias de solo lectura (ROM) y memorias de lectura-escritura (RAM). Los circuitos de memoria cuentan con conexiones para bus de datos, bus de direcciones y de control.

Memoria ROM: Read-Only Memory o memoria de solo lectura, es una memoria no volátil, puesto que el patrón de unos y ceros es grabado o programado por el fabricante, permaneciendo independientemente de la alimentación y solo permite la lectura de la información contenida. La organización de la memoria usualmente se expresan como “número de registros x tamaño de registro” o en “Bits”, por ejemplo 512x4, 4096x8, 65.536 – Bit, etc. Existen cuatro tipos de memoria ROM:

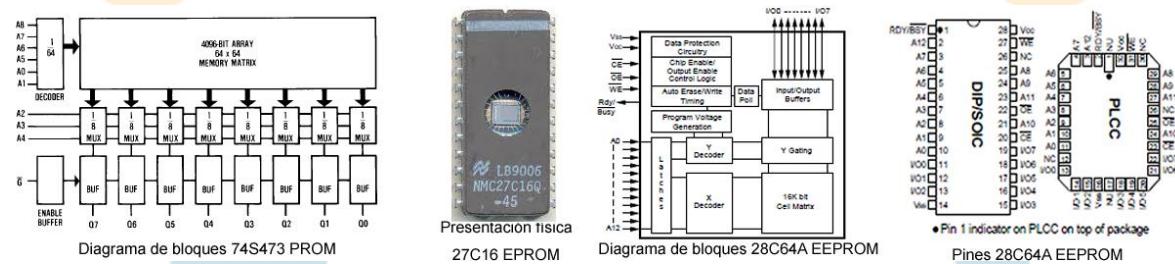
- **ROM:** es el tipo de memoria estándar programada por el fabricante.
- **PROM:** (Programmable Read Only Memory) Memoria de solo lectura programable, solo permite su programación una vez, el proceso quema microfilamentos de metal con alto voltaje dentro del chip, mediante dispositivos especiales llamados programadores o quemadores de memorias, opera a una velocidad considerablemente alta comparada con otros dispositivos. Ejemplo 74S473, memoria PROM TTL (512x8) 4096-Bit, mostrado en la Figura 27.
- **EPROM:** (Erasable Programmable Read Only Memory) Memoria de solo lectura programable y borrable, permite la programación y borrado por el usuario. La programación se hace por alto voltaje en un proceso similar a la memoria PROM. El borrado se efectúa aplicando luz ultravioleta de alta intensidad durante unos minutos a través de una ventanilla transparente

²² Extraído el 10 de Julio de 2011 desde <http://www.datasheetcatalog.net>

ubicada en la parte superior del Circuito integrado (CI). Ejemplo 27C16, (UV Erasable CMOS PROM Military Qualified), memoria EPROM CMOS (2048x8) 16384-Bit, mostrado en la Figura 27.

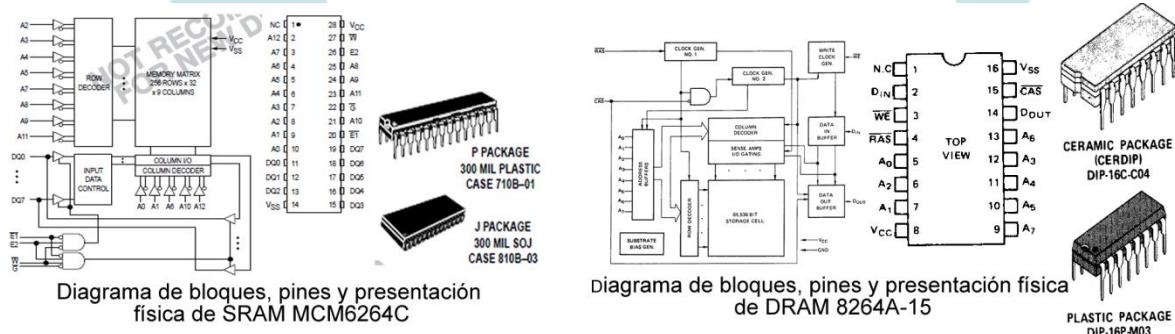
- **EEROM:** (Electrically Erasable Programmable Read Only Memory) también conocida como E²PROM o EEPROM, es una memoria de solo lectura borrable eléctricamente, permite la programación y borrado por el usuario. El borrado se realiza con el mismo dispositivo de programación aplicando un pulso eléctrico. Ejemplo 28C64A, CMOS EEPROM (8Kx8) 64K.

Figura 27. Diagrama de Bloques de CI-74S473, Circuito físico CI-27C16, Diagrama y Pines 28C64A²³.



Memoria RAM: Random-Access Memory o memoria de acceso aleatorio, es una memoria volátil, su patrón de unos y ceros se borra en ausencia de energía. Almacena información temporal que solo es utilizada mientras el circuito está energizado. Al igual que en la memoria ROM, la organización de la memoria usualmente se expresan como por ejemplo, en la memoria 7489, un CI TTL RAM (16x4) 64-Bit. Existen varios tipos de memoria RAM:

Figura 28. Diagrama de Bloques, Pines y encapsulado CI-MCM6264C y CI-8264A-15²³.



- **SRAM:** RAM estática, construida a partir de circuitos Flip-Flop que actúa como celdas de memoria de lectura/escritura. No requiere refrescar la

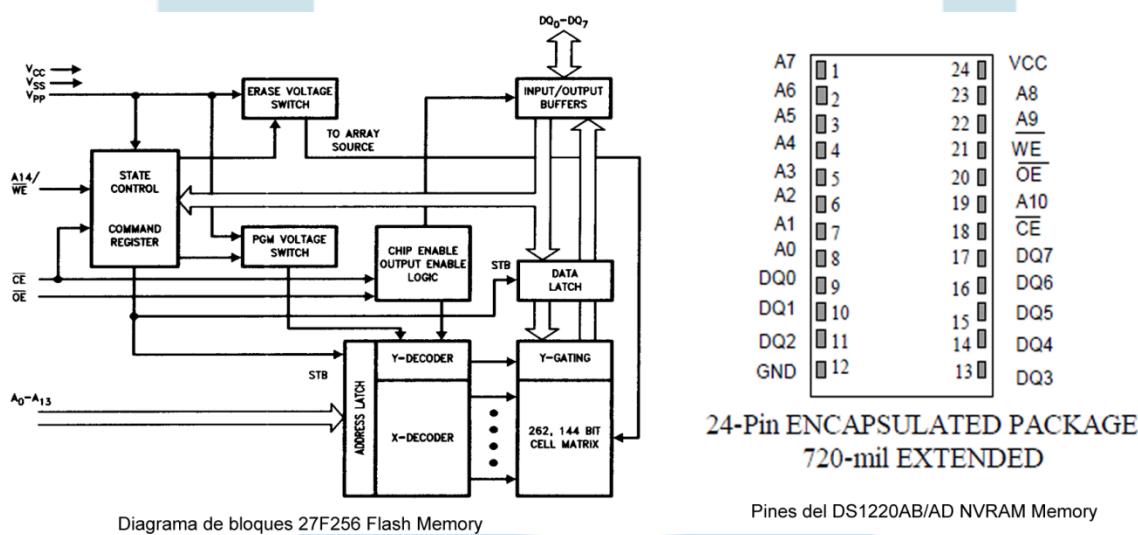
²³ Extraído el 10 de Julio de 2011 desde <http://www.datasheetcatalog.net>

información almacenada, es decir, revisarla y recargarla. Ejemplo MCM6264C, SRAM (Fast Static RAM) 8Kx8 Bit, mostrado en Figura 28.

- **DRAM:** RAM dinámica, las celdas de memoria utilizan la capacitancia para almacenar el estado lógico, por lo que se requiere refrescar la información almacenada cada cierto periodo de tiempo, usualmente cientos de veces por segundo. La ventaja en relación con la SRAM es que presentan un bajo consumo potencia, mayor capacidad en menor espacio y mayor velocidad de acceso o transferencia (proceso de lectura/escritura). Ejemplo MB 8264A-15 DRAM 65.536 – Bit, MK 4116P DRAM 16.384x1 Bit, mostrado en Figura 28..

NVRAM: La memoria de acceso aleatorio no volátil o NVRAM, no pierde la información después de retirar la energía del circuito, se tienen dos propuestas de NVRAM, una en la que se integra en un mismo dispositivo una memoria RAM, una pila de litio para mantener los datos almacenados (hasta por 10 años) y un controlador que en ausencia de energía externa coloca los buses en alta impedancia. La segunda opción combina en un mismo circuito integrado, bit a bit (en paralelo), una memoria RAM y una memoria EEPROM, en presencia de energía funciona normalmente la memoria RAM y cuando se activa un pulso de retención generado interna o externamente al circuito integrado, los datos de la memoria RAM se gravan en la memoria EEPROM, logrando almacenarlos por más de 10 años. Ejemplo de chip NVRAM es el DS1220AB/AD 16K – Bit, mostrado en la Figura 29.

Figura 29 Circuito 27F256 CMOS Flash y DS1220AB/AD NVRAM²⁴



²⁴ Extraído el 10 de Julio de 2011 desde <http://www.datasheetcatalog.net>

FLASH: Este tipo de memoria deriva su tecnología de la memoria EEPROM, mientras que en la memoria EEPROM en cada operación de programación solo se podía actuar sobre una única posición de memoria, la memoria flash puede actuar sobre múltiples posiciones de memoria (un bloque o todo el chip). Se caracteriza por su velocidad, bajo consumo, portabilidad y gran capacidad en relación a su tamaño, convirtiéndolas en verdaderos discos portables sin parte móviles. Existen dos tipos de memoria Flash, las de tipo NOR y las de tipo NAND, en donde los transistores utilizan un túnel que controla el flujo de electrones y por ende los estados lógicos uno (1) y cero (0). Las memorias Flash tienen un ciclo de lectura y escritura entre 10.000 y un millón de veces. Ejemplo de circuito de memoria Flash 27F256 CMOS Flash memory (32Kx8) 256K – Bit, mostrado en la Figura 29.

Lección 4: Arquitectura, funcionamiento, organización de la memoria, Registros, segmentos, instrucciones y Modos de direccionamiento en un Microcomputador

Independiente del tipo de microprocesador, memoria y dispositivos de entrada y salida, cualquier microcomputador contiene unidades funcionales básicas comunes que deben ser estudiadas inicialmente para poder analizar una arquitectura en particular, en este caso la arquitectura X86.

El microcomputador

En la actualidad existen varias empresas que fabrican microprocesadores de propósito general, muchas empresas fabrican dispositivos de memoria y componentes que numerosas empresas utilizan para los sistemas microcomputadores. Esta gran variedad de microcomputadores hace que cada vez que sale al mercado un nuevo microprocesador, el programador considere varios aspectos a estudiar.

La arquitectura: la arquitectura se relaciona directamente con la CPU respecto a sus registros, tamaño en bits de los buses, gestión de memoria, interrupciones, frecuencia de reloj, entre otros.

El repertorio de instrucciones: tiene que ver con el conjunto de operaciones u órdenes que puede ejecutar el microprocesador. El repertorio de instrucciones incluye instrucciones orientadas a operaciones aritméticas y lógicas, transferencia de datos, comprobación de datos, operaciones de entrada/salida y bifurcaciones.

Sistema Mínimo que utiliza el microprocesador: compuesto por el esquema de conexiones del microprocesador con otros dispositivos en forma de Circuitos integrados (CI) como memoria RAM y ROM, puertos de entrada/salida, decodificadores de dirección, señal de reloj y fuente de alimentación, de manera que se pueda obtener un procesamiento y control de datos.

Señales de control: son las señales que controlan el tipo de acceso (escritura o lectura) y el tipo de dispositivo a controlar (memoria o periférico).

Funciones de pines: es importante estudiar detalladamente la función de cada pin del microprocesador, de manera que se determine que pines son de alimentación, reloj, reset, interrupción, control y de entrada/salida de datos.

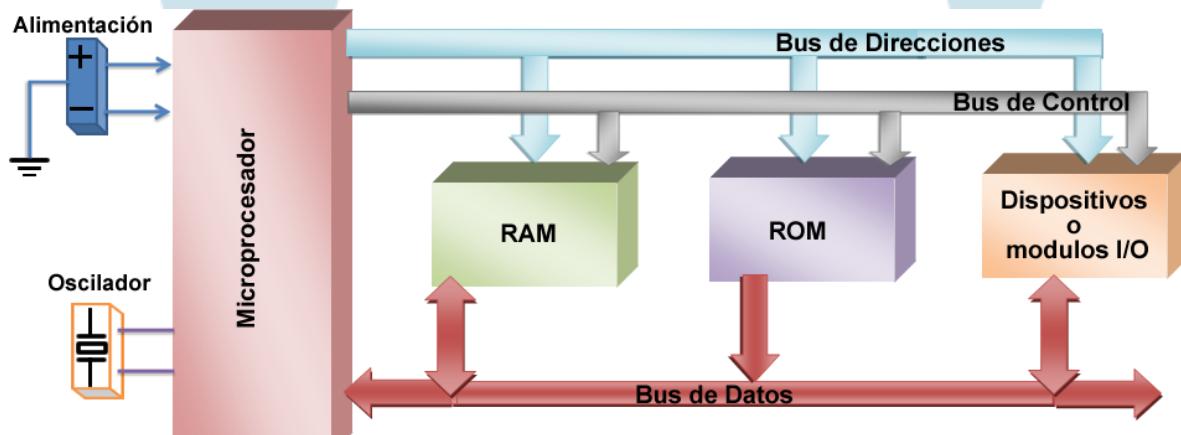
Arquitectura y Funcionamiento de la microcomputadora

Los sistemas de cómputo basados en microprocesadores se componen de tres bloques fundamentales, la Unidad Central de Procesamiento – CPU, Dispositivos de memoria y Puertos de Entrada / Salida, de igual forma el microprocesador está conformado por varios bloques, entre ellos están la Unidad Aritmético – lógica (ALU), una Unidad de Control (UC) y un bloque o matriz de registros.

Tomando como ejemplo un sistema mínimo con microprocesador, como se observa en la Figura 30., se tiene una fuente de alimentación para el microprocesador e integrados que confirman el sistema, como pieza clave se encuentra un circuito oscilador, generalmente basado en cristal de cuarzo que proporciona estabilidad a la señal de reloj, que gobierna las operaciones y micro operaciones dentro del microprocesador. La unidad de entrada está conformada por dispositivos o módulos llamados de entrada / salida o I/O (*Input/Output*), encargados de recibir la información del mundo exterior que es llevada hacia la unidad de memoria (RAM) para ser procesada posteriormente. La información luego es llevada desde la unidad central de proceso (CPU) o Microprocesador, hacia circuitos o periféricos utilizando el bus de datos. La unidad de memoria RAM se encarga de almacenar los datos y la unidad de memoria ROM los programas que operan sobre esos datos. La CPU obtiene las instrucciones y los datos colocando una dirección en el bus de direcciones para posteriormente ser transferidos a través del bus de datos cuando la CPU lo solicite.

En los microcomputadores más sofisticados y completos, existen dos sistemas diferentes de memoria, la de almacenamiento primario y la del almacenamiento secundario. La memoria de almacenamiento primario (ROM y RAM) se refiere a la memoria que almacenan los programas que se van a ejecutar y los datos utilizados durante la ejecución del programa. La memoria secundaria es la encargada de almacenar grandes cantidades de datos que no se requieren con frecuencia, este tipo de dispositivo son los discos duros y discos 3,5". También se distinguen tres categorías, la ROM (Red Only Memory) o memoria de solo lectura donde se almacenan cierto tipo de programas como la BIOS, la memoria RAM (Random Access Memory) o memoria de lectura y escritura, donde se almacenan datos que los programas en ejecución van generando, como tercera categoría se encuentra la "cache" con tiempo de acceso muy rápido muy cercana al núcleo del procesador.

Figura 30. Sistema de básico de cómputo²⁵



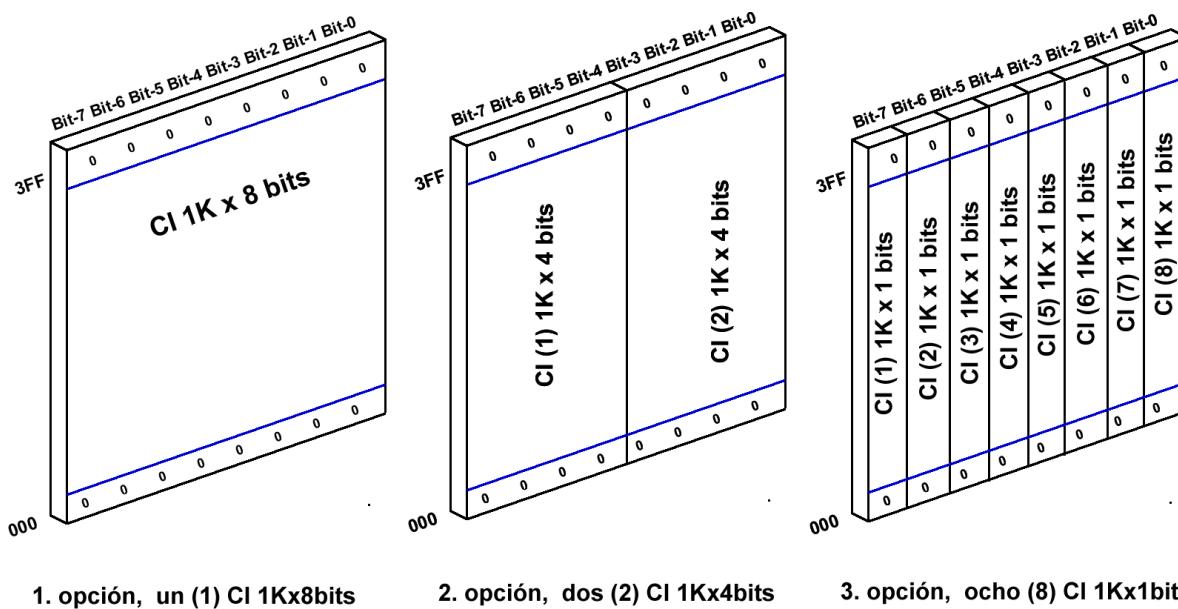
La ejecución de un programa se realiza de forma secuencial, las instrucciones almacenadas en la memoria de programa modifican los datos almacenados en la memoria u obtenidos a través de un dispositivo de entrada, los datos que se obtienen después del procesamiento pueden ser almacenados en la memoria o ser enviados a los periféricos de salida utilizando el bus de datos, la CPU utiliza el bus de control para establecer la lectura y/o escritura de los datos y con el bus de direcciones determina el destino de los datos.

²⁵ Extraído el 10 de Julio de 2011 desde Fuente (CEKIT, 2002)

Organización de la memoria

El proceso de escritura y lectura de una posición de memoria se denomina acceder a memoria, el acceso a memoria puede ser de forma secuencial en donde los datos se localizan en las posiciones de memoria de forma serial, o de forma aleatoria en la que se tiene un tiempo de acceso en el cual cualquier posición de memoria puede ser escrita o leída. Como ejemplos de memoria de acceso secuencial se tiene las cintas magnéticas y como ejemplo de memoria de acceso aleatorio se encuentra los dispositivos semiconductores de almacenamiento RAM y ROM.

Figura 31. Organización de memoria física para obtener un bloque de 1K x 8bits de memoria.



1. opción, un (1) CI 1Kx8bits

2. opción, dos (2) CI 1Kx4bits

3. opción, ocho (8) CI 1Kx1bit

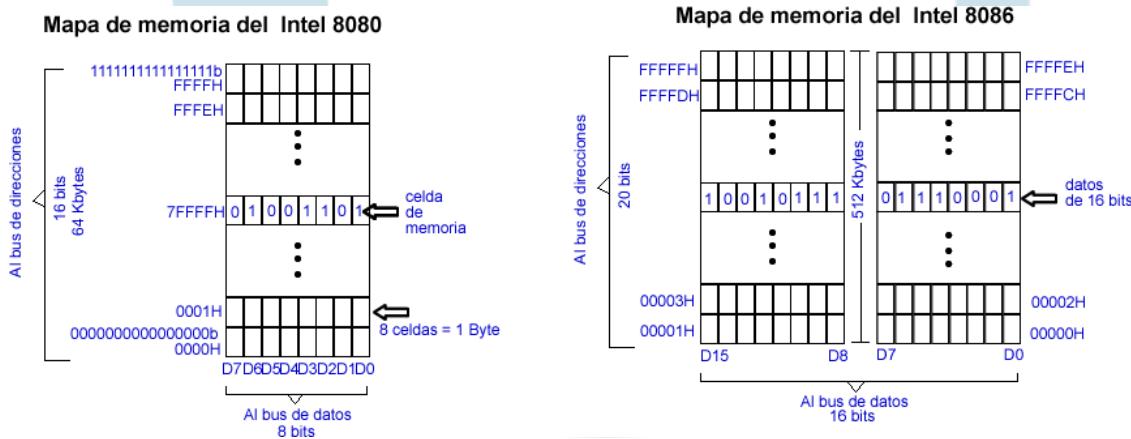
La memoria RAM y ROM físicamente se construye como circuitos integrados (CI) independientes o como celdas dentro del mismo chip en el caso de los microcontroladores. Físicamente se los chips de memoria se pueden organizar de tres formas para obtener los tamaños requeridos, la primera como un CI del tamaño requerido (registros x bits), la segunda como agrupación en paralelo de dos CI del mismo tamaño de registro pero con la mitad de bits de datos requeridos (registros x ½ bits) y la tercera como tantos CI como bits de datos se requiera, es decir, los CI son del tamaño del registro por un solo bit de datos, estos CI en paralelo (registros x 1 bit). Las celdas de memoria forman grupos de varios bits, que conforma los llamados “registros”, para el caso de la memoria RAM típicamente de 8 bits o un byte, en el caso de la memoria ROM depende del tipo de dato que almacena, por ejemplo en el caso del microcontrolador PIC16C84 la ROM almacena 14 bits, se puede considerar la estructura de la memoria como bloques conformados por registros superpuestos, a estos bloques se les denomina

“bancos de memoria”. El ancho del banco de memoria determina el tamaño de los datos e incluso el tamaño del bus de datos y el tamaño del banco de memoria determina la capacidad de almacenamiento de datos.

Ejemplo: Se requiere organizar una cantidad de 1K X 8 bits de memoria RAM física utilizando CI (Circuitos Integrados) para un sistema mínimo con microprocesador, el diseñador puede optar por un CI de 1024 x 8 bits, dos CI de 1024 X 4 bits, ocho (8) CI de 1024 x 1 bit. En la Figura 31. Se pude observar esta distribución.

Ejemplo: La Figura 32. muestra la estructura de memoria para un microprocesador 8080 que tiene 16 líneas en el bus de direcciones, que pueden generar $2^{16} = 65536_{10}$ combinaciones diferentes de unos y ceros para un número igual de posiciones de memoria, cada posición de memoria está formada por un registro con 8 celdas de almacenamiento binario, se forma un banco de memoria con 65.536 registros de 8 bits o 64KBytes, cada uno con una dirección de memoria, observe que en la dirección de memoria 7FFFH está almacenado el Byte 01001101₂. Es común representar las direcciones utilizando notación hexadecimal para simplificar, la dirección en binario 0000 0000 0000 0000₂ es equivalente a la dirección Hexadecimal 0000H, se utiliza “H” al final para denotar que la dirección está en Hexadecimal. Observe que para el microprocesador 8080 se tiene un solo banco de memoria, cada registro es de 8 bits o 1 Byte, por tanto el bus de datos es de 8 líneas (8 bits). Para el caso del microprocesador 8086 como tiene un bus de direcciones de 20 bits y 16 bits en el bus de datos, cada registro es de 8 bits o 1 Byte, por tanto la memoria está conformada por $2^{20} = 2^{10} \times 2^{10} = 1024 \times 1024 = 1K \times 1K = 1Mbyte$, pero para generar datos de 16 bits se hace necesario tener dos bancos de memoria cada uno de 512KBytes.

Figura 32. Organización de la memoria en un microprocesador 8080 y 8086



Conjunto de instrucciones

El repertorio de instrucción de un microprocesador o microcontrolador puede variar desde unas pocas hasta cientos de instrucciones básicas, no hay una estandarización en el repertorio de instrucciones por el individualismo de cada fabricante, la arquitectura del microprocesador o la aplicación del dispositivo. Es posible poder clasificar el repertorio de instrucciones como:

Instrucciones aritméticas: son instrucciones encargadas de realizar operaciones aritméticas, las instrucciones más usuales y básicas son suma, resta, incremento, decremento, comparar y negar, muchos implementan operaciones de multiplicación y división.

Instrucciones lógicas: algunas de las instrucciones que pueden incluir son del tipo AND, OR, XOR, Not, desplazamiento a izquierda y derecha, operaciones de test y otras que dependen del fabricante y dispositivo.

Instrucciones de transferencia de datos: estas instrucciones son las encargadas de cargar, almacenar y transferir datos entre registros de memoria y puertos entrada/salida, incluso el borrado de datos.

Instrucciones de bifurcación: son instrucciones de toma de decisiones, evalúan una condición y como resultado encaminan la secuencia de órdenes generando saltos (“goto” o “jmp”) o llamados (“call”) a otras secciones del programa o subrutinas. Para el caso de los llamados (“call”) se incluyen las instrucciones de retorno que recuperan la dirección de memoria guardada en pila, para regresar a la posición siguiente de memoria de programa donde se originó el llamado.

Instrucciones misceláneas: instrucciones como “no operar”, “dormir”, “habilitar interrupción” hacen parte de este tipo de instrucciones que el fabricante del dispositivo determina en relación con el microprocesador o microcontrolador específico para la aplicación.

Registros en microprocesadores X86

Los registros son espacios de memoria conformado por varias celdas de almacenamiento binario, el tamaño de los registros varía según el microprocesador o microcontrolador, se toma la arquitectura de

microprocesadores x86 para presentar los registros más usuales, puesto que esta arquitectura es la más ampliamente utilizada.

Registros de uso general: En los microprocesadores x86 se tienen cuatro registros duales, lo que significa que pueden ser manipulados como 8 y 16 bits, estos registros en modo de 16 bits se conocen como AX, BX, CX y DX. En la modalidad de 8 bits responden a una letra distintiva que los divide en parte baja (L) desde el bit 0 al bit 7 y la parte alta (H) desde el bit 8 hasta el bit 15, es decir, AH+AL=AX, BH+BL=BX, CH+CL=CX y DH+DL=DX.

Registros apuntadores: son registros que apuntan a una posición de memoria, entre ellos están:

- **BP**, Base Pointer – apuntador base utilizado para manipular la información de la pila o Stack.
- **SP**, Stack Pointer – apuntador de pila, utilizado en conjunto con el registro SS (Stack Segment o Segmento de Pila), para la creación de una estructura en memoria denominada pila o Stack.

Registro apuntador de instrucciones – IP: Este registro se utiliza en combinación con el registro CS (Code Segment o Registro de Segmento de Código) para apuntar a un área de memoria específica en la cual se almacena la siguiente instrucción a ejecutar, su contenido cambia en razón al flujo de ejecución del programa.

Registros índice: son muy utilizados en operaciones con cadenas, estos registros se identifican como SI (Source Index – Índice fuente) y DI (Destination Index – Índice destino).

Registro de estado: este registro está conformado por celdas (bits) que cambian en relación directa con el resultado de una operación aritmética o lógica. Recibe el nombre de registro de estado o STATUS REGISTER, palabra de estatus de programa o PSW (Program Status Word). En él se encuentran usualmente los bits también llamados “Flags” para ver estados de acarreo, paridad, desbordamiento y resultado cero (0) en una operación.

Segmentos

Cualquier operación que accede a la memoria se hace mediante registros que seleccionan un área contigua de memoria que puede ser de 64 KB, conocida como segmento, estos segmentos en su interior tienen desplazamientos internos para acceder la información. El procesador generalmente tiene cuatro registros llamados registros de segmentos:

CS-Code segment o registro de segmento de código: se encuentra asociado al código del programa, es decir, controla el código de los programas y tiene como complemento el registro IP. La combinación CS:IP da como resultado la siguiente instrucción a ser ejecutada.

DS-Data Segment o registro del segmento de datos: es donde se guardan los datos del programa.

ES-Extra Segment o registro de segmento extra: sirve como apoyo al segmento de datos ofreciendo una ampliación del mismo.

SS-Stack Segment o registro del segmento de pila: tiene la función de controlar el área donde se creará la pila.

La pila

Es un bloque de memoria utilizado para almacenar información bajo el esquema de ultimo en entrar primero en salir (*LIFO* - Last Input First Out), este bloque de memoria es conocido como pila ó “stack” y es necesario que exista, pues la CPU lo requiere, en la pila los datos son empujados para posteriormente ser extraídos comenzando con el ultimo que fue introducido, la pila proporciona a la CPU un mecanismo para que pueda almacenar el contexto de un programa, es decir las direcciones de retorno después de invocar alguna rutina o llamado y el manejo de alguna interrupción, también sirve como área temporal de almacenamiento.

La arquitectura x86 obliga a usar en combinación un registro de segmentos (CS, DS, ES, SS) y un desplazamiento para obtener la localidad absoluta de un byte de datos o instrucción localizado en la memoria (RAM o ROM), en el caso del 8086 el límite para cada segmento es de 64KB.

Modos de direccionamiento

Antes de establecer los modos de direccionamiento se considera definir qué es una dirección de memoria y como se calcula.

Direcciones de memoria: Las direcciones de memoria son localidades o ubicaciones del registro que almacena cierta información (datos o instrucciones). El esquema de direccionamiento que impone la CPU obliga a usar registros de

segmentos en cada operación, por ejemplo el 8086 tiene un bus de 20 bits y registros de 16 bits como se observa el registro no tiene la capacidad de representar 20 bits, lo que hace necesario dividirla en dos partes, el segmento y el desplazamiento dentro del segmento, cada uno de 16 bits esto se representa en Hexadecimal como SSSS:DDDD, donde SSSS representa el registro de segmento, dos puntos, DDDD que representa el desplazamiento.

Ejemplo: 0050:0002 es la dirección del tercer byte dentro del segmento 0050H.

Calculo de direcciones de memoria: La forma de calcular una dirección real de memoria es recorrer el registro de segmento 4 bits a la izquierda y sumarle el desplazamiento.

Ejemplo: 2B45:0032 si solo se suman las cantidades anteriores se obtendría 2B77H, que tiene 16 bits (recordar que un hexadecimal representa 4 bits) y no 20 como se necesita, si se corre el registro de segmento 4 bits a la izquierda se obtiene 2B450H que claramente es de 20 bits y se le suma el desplazamiento con lo que se obtiene $2B450H + 0032H = 2B482H$, de esta forma se obtiene la dirección absoluta de memoria. Es posible que una o más combinaciones de segmentos y desplazamientos originen la misma dirección.

La CPU ofrece varios métodos para calcular direcciones de memoria. Los accesos a memoria pueden categorizarse de dos modos:

1. **Accesos para obtener la siguiente instrucción a ejecutarse utilizando la combinación CS:IP.**
2. **Para obtener algún dato.** Se utilizan varias modalidades de direccionamiento:

2.1 Transferencia de registro a registro: es una de las más sencillas, en la cual se realiza una copia del registro fuente al registro destino dejando intacto el primero.

Ejemplo: *MOV AX, BX.* La instrucción *MOV* hace referencia a mover datos, en este caso desde un registro *BX* que es la fuente desde donde se copia el dato al registro de destino *AX*. No hay acceso a memoria.

2.2 Inmediata: el operando se incluye como parte de la instrucción.

Ejemplo: *MOV AX, 5,* el número 5 forma parte de la instrucción en su totalidad, el número se especifica como una constante numérica en la instrucción y no hay necesidad de acceder a memoria, moviendo 5 al registro *AX*.

2.3 Directa: Un valor de 16 bits forma parte de la instrucción y es interpretado como un acceso a la memoria.

Ejemplo: MOV AX, VALOR; donde VALOR es una localidad de memoria donde se encuentra la información que se moverá al registro AX, más específicamente con respecto a la instrucción anterior tendríamos MOV AX, [120H], que mueve el contenido almacenado en la localidad de memoria [120H] al registro AX, *los corchetes indican localidad de memoria.*

2.4 Indirecta: Esta modalidad emplea el contenido de los registros índice SI (Source Index – índice fuente) y DI (Destination Index – índice destino), para calcular la dirección de memoria. Se puede incluir un operando inmediato en la misma instrucción para realizar el cálculo.

Ejemplo: MOV AX, [SI], donde si se considera que SI tiene el valor 0270H, la instrucción toma el contenido del registro SI (0270H) como una localidad de memoria, accediendo a dicha localidad, obteniendo el valor y almacenándolo en el registro AX.

Es posible utilizar el operando inverso:

Ejemplo: MOV [SI-0100], BX.

En esta modalidad todos los desplazamientos se hacen en función del registro de segmentos DS.

2.5 Base relativa: Esta modalidad utiliza dos registros, el BX y el BP, para calcular una dirección de memoria, el desplazamiento indicado por el registro BX se toma en función al registro de segmento DS y el desplazamiento indicado por el registro BP se toma en función al registro de segmento SS, su funcionalidad se relaciona con la anterior modalidad a excepción del registro BP. Al usarse BP debe incluirse un segmento operando, ya sea un índice o un operando inmediato.

Ejemplo: MOV AX, [BP], será ensamblada como MOV AX, [BP+0].

Lección 5: El microprocesador

El microprocesador, también llamado procesador se considera el centro o “cerebro” de un sistema de cómputo, está constituido por cientos, miles o millones de componentes electrónicos en un solo chip o Circuito Integrado (CI) con un alta escala de integración (VLSI – Very Large Scale Integration), en la actualidad se manejan en escalas de ULSI (Ultra Large Scale Integration), hasta GLSI (Giga Large Scale Integration) y XLSI (Extreme Large Scale Integration). El microprocesador es el circuito físico de la Unidad Central de Procesamiento (CPU). El microprocesador es el dispositivo encargado de las operaciones

aritmético – lógicas, control y comunicaciones con los demás componentes integrados que conforman el sistema de cómputo o microcomputador.

El microprocesador circuito físico de la Unidad Central de Proceso - CPU

En la práctica la Unidad Central de Proceso o CPU se encuentra en forma de un circuito integrado llamado microprocesador. El microprocesador es un circuito digital que acepta o lee datos aplicados a un cierto número de líneas de entrada, los procesa de acuerdo a las instrucciones secuenciales de un programa almacenado en su memoria y suministra o escribe los resultados del proceso con cierto número de salida. La evolución de los microprocesadores ha hecho que evolucionen los computadores ampliando sus prestaciones, servicios, capacidad y facilidad de interacción con el medio. Los microprocesadores se aplican a otras situaciones de control y supervisión en el control industrial, maquinaria, motores, telemetría, robótica, automatización entre otros.

Los datos de entrada pueden provenir de interruptores, sensores, convertidores A/D, teclados, etc; los datos de salida son dirigidos a display, LCD, CTR, convertidores D/A, impresoras, etc. El soporte físico de las instrucciones del programa es la memoria, la cual almacena los datos para que sean procesados, los niveles lógicos (unos y ceros) en las líneas de salida de un microprocesador no solo dependen del programa, también dependen de la historia o estado anterior de las señales de entrada. Como se observa es clave la cantidad de pines o conexiones de entra/salida que tenga el microprocesador para interactuar con los periféricos y la memoria por eso la mayoría de microprocesadores utilizan las mismas líneas para entrada y salida. Las líneas de control sincronizan las operaciones con los componentes externos conectados y ejercen un control global de los buses de datos y direcciones. Los programas realizan funciones o aplicaciones limitadas por la imaginación del programador y la capacidad de los dispositivos, el programa que se ejecuta debe alojarse en determinadas posiciones de memoria y ser escrito en un lenguaje que la CPU entienda, es decir, lenguaje binario. La CPU entonces lee de forma secuencial la lista de instrucciones, las interpreta y controla la ejecución de cada una de ellas, para ejecutar cada instrucción la CPU realiza los siguientes pasos:

1. Leer de la memoria la instrucción a ejecutar y guardarla en un registro interno de la CPU.
2. Identificar la instrucción que acaba de leer.

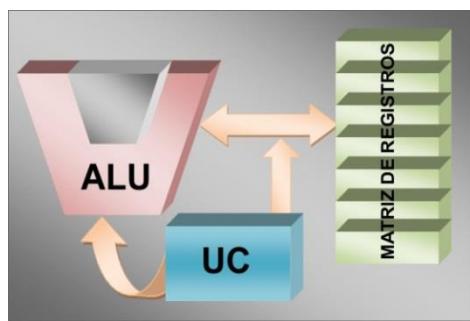
3. Comprobar si la instrucción necesita datos que se encuentran en memoria, si es el caso, determina la localización de los mismos.
4. Buscar los datos en memoria y guardarlos en registro dentro de la CPU.
5. Ejecutar la instrucción.
6. El resultado puede ser almacenado en memoria o retenido esperando la siguiente instrucción o incluso generar comunicación con otros elementos externos a la CPU.
7. Comienza un nuevo ciclo empezando con el primer paso.

La ejecución de cada instrucción implica un movimiento de datos a partir de pulsos electrónicos que siguen una secuencia y orden establecido por una señal generada en un circuito que genera pulsos periódicos, este circuito denominado “reloj”, es simple en su construcción pero vital para el funcionamiento del microprocesador.

Arquitectura del microprocesador

La arquitectura en un microprocesador se refiere a las unidades estructurales que lo conforman, su diseño y organización. Es difícil definir un modelo de microprocesador pues cada uno tiene una lógica de funcionamiento propia. Las principales unidades básicas de un microprocesador son la Unidad Aritmética y Lógico – ALU, la matriz de registros y la Unidad de Control. En la Figura 33 se puede observar las unidades básicas de un Microprocesador.

Figura 33. *Unidades Básicas de un MP²⁶*

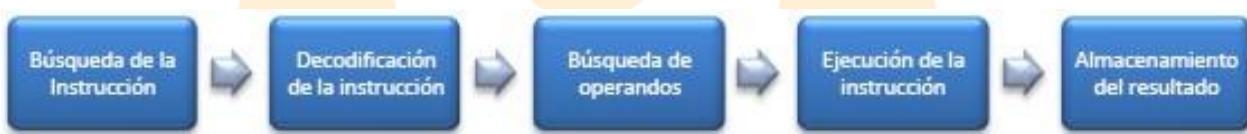


Aunque existen otras unidades auxiliares y secciones que trabajan en conjunto con las anteriores para formar un microprocesador específico. Sin importar la forma física que tome la CPU, su operación fundamental es ejecutar una secuencia de instrucciones denominadas “programa”, utilizando cuatro pasos

²⁶ Extraído el 10 de Julio de 2011 desde Fuente (CEKIT, 2002)

básicos leer, decodificar, ejecutar, y escribir. Los pasos en la ejecución de una instrucción se muestran en la Figura 34.

Figura 34. Pasos en la ejecución de una instrucción



El microprocesador junto con sus unidades básicas realiza las principales tareas en un sistema de cómputo:

1. Transferencia de datos desde y hacia: la memoria, periféricos y/o dispositivos.
2. Operaciones aritméticas y lógicas para ello el microprocesador cuenta con una unidad interna aritmético – lógica (ALU) (hace sumas y comparaciones y con base a estas realiza las demás operaciones, complementos, restas, etc.).
3. Un control sobre el flujo de un programa, este programa se está ejecutando y el microprocesador esta accionando sobre el programa comunicación con la memoria.

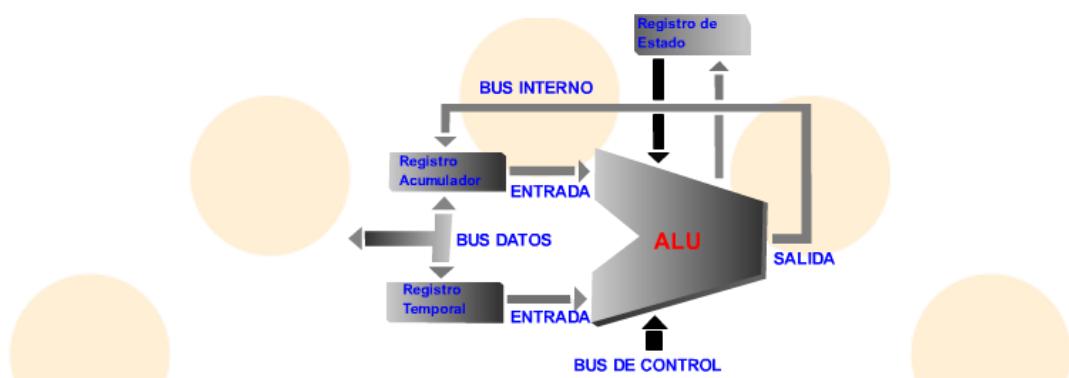
El poder del microprocesador radica en:

1. La capacidad de ejecutar millones de instrucciones por segundo (MIPs), que provienen de un programa o software almacenado en la memoria de programa.
2. La capacidad para tomar decisiones simples basadas en hechos numéricos, como signos, comparaciones o resultados lo que permite modificar el flujo del programa.

Unidad Aritmética Lógica - ALU

Los datos en el ALU se presentan en registros, posteriormente en ellos se almacenan los resultados de las operaciones, estos registros son posiciones de memoria temporales, internas en el microprocesador y conectados a la ALU, los procesos aritméticos y lógicos pueden activar indicadores o también llamadas banderas (flags), que son indicadores del estado del microprocesador, normalmente estos indicadores son almacenados en un registro especial dentro del microprocesador (registro de estado), por ejemplo cuando el resultado de una operación es “cero” su indicador o bandera (*flag*) de cero cambian su estado. La Figura 35 muestra la interconexión de la ALU.

Figura 35. Interconexión de la ALU



La unidad de control proporciona las señales que gobiernan el funcionamiento de la ALU y la transferencia de datos dentro y fuera de ella. La Unidad Aritmética Lógica – ALU, es la encargada de realizar las operaciones elementales de tipo aritmético (generalmente sumas o restas) y de tipo lógico (generalmente comparaciones). Para realizar su función, consta de un banco de registros (BR), este banco está constituido por 8, 16 ó 32 registros de tipo general que sirven para situar datos antes de cada operación, para almacenar datos intermedios en las operaciones y para operaciones internas del procesador.

Como se había establecido antes los datos que maneja la ALU tienen formato de complemento a dos, haciendo más sencillo el trabajo y las operaciones con números negativos. En general los bits restantes en una operación que involucra números con signo siguen lo establecido en la Tabla 7.

Tabla 7. Representación con signo en la ALU²⁷

RANGO DE VALORES	-2^{n-1} a $2^{n-1}-1$
Nº de representaciones del cero	Una
Negación	Ejecutar el complemento booleano de cada bit y sumar 1 al patrón de bits resultante.
Ampliación de la longitud de bits	Las posiciones de bit extra se añaden a la izquierda, rellenándolas con el valor del bit de signo original
Regla para el desbordamiento	Si se suman dos números de igual signo (ambos positivos o ambos negativos), hay desbordamiento si, y sólo si, el resultado tiene signo opuesto.
Regla para la resta	Para restar B de A, se toma el complemento a dos de B y suma a A.

La representación del número cero tienen un bit de signo cero (0), una magnitud cero. El costo de manipular números negativos es la restricción de valores representables, por ejemplo para registros de 8 bits, el valor máximo representable

²⁷ Stallings, 2000

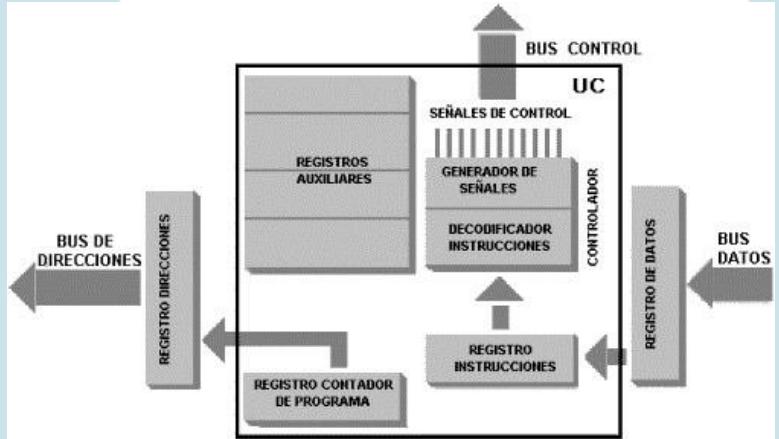
sin signo es de 00000000 a 11111111, o equivalente a 255 en decimal, el numero con signo, en formato positivo no puede ser mayor a 01111111 binario o 127 decimal, por cuanto el siguiente numero binario valido seria 10000000 y al tener encendido el bit más significativo es evidente que representa un numero negativo que correspondería al -128. Entonces para representar con "n" bits el rango de números positivos y negativos se establece como números positivos desde cero hasta $2^{n-1}-1$, para los negativos se representan desde -1 hasta -2^{n-1} .

Unidad de Control

La unidad de control (UC) es el centro nervioso de la CPU, su estructura se muestra en la Figura 36, desde ella se controla y gobiernan todas las operaciones (búsqueda, decodificación, y ejecución de la instrucción), proporcionando las señales de control para la entrada o salida de datos en el microprocesador y la sincronización de la ejecución de las instrucciones, estas tareas básicas se pueden resumir en:

- **Secuenciamiento:** se basa en el programa que se ejecuta, haciendo que el procesador avance en cada instrucción con la realización de una serie de microoperaciones.
- **Ejecución:** la unidad de control hace que se ejecuten cada una de las microoperaciones necesarias.

Figura 36. Unidad de Control²⁸



El trabajo que realiza la Unidad de control implica:

1. **Microoperaciones:** para entender el concepto de microoperación hay que tomar los siguientes conceptos:

²⁸ Extraído el 10 de Julio de 2009 desde <http://www.computacion.geozona.net/cpu.html>

1.1 Ciclo de reloj, consiste en una serie de pulsos periódicos generados por un reloj local interno o externo, este ciclo permite organizar la secuencia interna de eventos en la CPU.

1.2 Ciclo máquina, consiste en uno o varios ciclos de reloj necesarios para ejecutar una instrucción, algunas instrucciones utilizan más de un ciclo máquina para completar sus actividades, tal es el caso de instrucciones de decisión-salto.

1.3 Ciclo de instrucción, compuesto en forma general por captación, ciclo indirecto, ejecución e interrupción, cada uno de estos contienen pasos que involucran a registros del microprocesador, estos pasos son las **microoperaciones**.

2. Señales de control: Las señales de control se componen de señales de entrada que permiten determinar el estado del sistema y salidas que permiten controlar el comportamiento del mismo. En la Figura 37, se muestran las señales usuales en la Unidad de Control.

Figura 37. I/O en la Unidad de Control²⁹



2.1 Señales de entrada.

2.1.1 Reloj: Es el encargado de generar pulsos periódicos utilizados por la Unidad de Control para ejecutar una o varias microoperaciones en cada pulso de reloj, este pulso se denomina “período de reloj” o “tiempo de ciclo del procesador”.

2.1.2 Registro de instrucción: contiene el código de la instrucción con la que la Unidad de control determina el conjunto de microoperaciones a realizar durante el ciclo de ejecución.

2.1.3 Indicadores: son bits que utiliza la Unidad de Control para determinar el estado del procesador y el resultado de la operación anterior en el ALU, también llamados banderas o “flags”.

2.1.4 Señales de control del bus de control: señales de control desde el bus del sistema (señales de interrupción y de reconocimiento).

²⁹ Extraído el 10 de Julio de 2009 desde <http://organizacioncomputadoras.blogspot.com/2009/05/31-funciones-de-la-unidad-de-control.html>

2.2 Señales de salida.

2.2.1 Señales de control internas al procesador:

- Utilizadas para transferir datos de un registro a otro.
- Activar funciones específicas de la ALU.

2.2.2 Señales de control hacia el bus de control:

- Señales de control de la memoria.
- Señales de control de las E/S.

Buses en los microprocesadores

A partir del desarrollo de la arquitectura Von Newman, se introdujo en los microprocesadores el concepto de programa almacenado y la ejecución secuencial del programa almacenado. Para que el microprocesador pueda comunicarse con los diversos periféricos que tiene existen unos medios llamados buses.

Los buses permiten la conexión con los periféricos incluida la memoria y permiten la transmisión de la información, físicamente los buses consisten en líneas conductoras que permiten la circulación de los pulsos eléctricos. Generalmente el número de líneas de entrada es igual al número de líneas de salida, este número define la longitud de la palabra de datos del microprocesador que comúnmente es 4, 8, 16, 32 y 64 bits aunque la unidad básica de almacenamiento y procesamiento es de un Byte. Se tienen tres buses básicos en los microprocesadores, el bus de datos, bus de direcciones y bus de control, aunque generalmente las conexiones a estos buses son líneas independientes, en algunos micros se multiplexan teniendo direcciones, datos y/o control en las mismas líneas de bus.

Bus de datos, es el encargado de llevar datos e instrucciones hacia y desde el microprocesador, es un bus bidireccional en donde los datos varían en tamaño desde 8 a 64 bits, por el bus de datos las instrucciones y los datos son transferidos al microprocesador y los resultados de una operación son enviados desde el microprocesador.

Bus de direcciones, contiene la información digital que envía el microprocesador a la memoria y demás dispositivos direccionales del sistema para seleccionar una posición de memoria, una unidad de entrada/salida o un registro particular de la misma, la cantidad de líneas del bus de direcciones determina la capacidad máxima de la memoria que puede direccionar el sistema basado en microprocesador.

Ejemplo: Los primeros microprocesadores utilizados en sistemas de cómputo tenían 16 líneas de direcciones con lo que podían direccionar hasta $2^{16}=2^{10}*2^6=1024 * 64 = 65536 = 64\text{KB}$.

Cuantos más bits tenga el bus de direcciones más capacidad de direccionamiento de memoria tendrá el sistema, el número de líneas de dirección se ha incrementado a 20, 24, 32, 36 hasta 64 bits.

Ejemplo: con un bus de direcciones de 36 bits se permite direccionar:

$$2^{36}=2^{10}*2^{10}*2^{10}*2^6=1024 * 1024 * 1024 * 1024 * 64 = 1\text{K}*\text{1K}*\text{1K}*\text{1K}*\text{64} = 1\text{M} * 1\text{M} * 64 \\ 2^{36}=1\text{G} * 64 = 64 \text{ GB.}$$

Bus de control, se utiliza para coordinar, sincronizar y comunicarse con los dispositivos externos, también se utiliza para insertar estados de espera para los dispositivos más lentos y evitar congestión en el bus, contiene líneas que seleccionan la memoria o dispositivo E/S y los habilita para que haga operaciones de lectura o escritura, existen varias líneas de control que se listan a continuación, los microcontroladores generalmente implementan algunas o todas las líneas mencionadas³⁰:

- Líneas de lectura/escritura.
- Líneas de entrada/salida.
- Líneas de control del bus.
- Líneas de status de ciclo.
- Líneas de interrupción.
- Líneas de reinicialización.
- Líneas de reloj

Ejemplo: Particularmente en los microprocesadores 80x86 existen cuatro líneas relacionadas con el control de lectura/escritura y el control de dispositivos entrada/salida que son:

- *MRDC* control de lectura de memoria.
- *MWTC* control de escritura de memoria.
- *IORC* control de lectura en un puerto.
- *IOWC* control de escritura en un puerto.

³⁰ Tokheim, 1985

Matriz de registros

Un sistema de cómputo exige de la CPU la realización de varias funciones, captar instrucciones, interpretar instrucciones, captar datos, procesar datos y escribir datos; para que un microprocesador pueda realizar las tareas en un sistema de cómputo, es necesario que la CPU almacene alguna información de forma temporal. Esta información debe contener en un almacenamiento temporal la instrucción a ejecutar, los datos sobre los que debe operar, la dirección de la instrucción o dato a cargar en la ALU, las direcciones de salto o llamado, de forma que pueda saber dónde ir a buscar la siguiente instrucción. Lo anterior hace necesario que la CPU requiera de una memoria interna, es así como en la CPU, existe un conjunto de registros funcionando como un nivel de memoria que está por encima de la memoria principal y la cache, estos registros se conocen como la matriz de registros y pueden ser de dos tipos:

1. **Registros visibles para el usuario:** Permiten al programador de lenguaje máquina o ensamblador, minimizar las referencias a memoria principal optimizando el uso de estos registros, estos registros se pueden referenciar mediante lenguaje máquina, están disponibles para todos los programas (de sistemas y aplicación).
2. **Registros de control y de estado:** Utilizados por la Unidad de Control para controlar las operaciones del procesador, y por los programas o rutinas con privilegios del Sistema Operativo para controlar la ejecución del programa.

1. Registros visibles para el usuario: Se pueden clasificar en las siguientes categorías:

1.1 Registros de Propósito general: El programador puede asignarlos a varias funciones, como almacenamiento de datos o direccionamiento. Para microprocesadores de 8 bits estos registros no pueden funcionar como registro acumulador en la ALU y en operaciones de E/S, aunque en microprocesadores de 16 bits o más se permite que estos registros de propósito general se utilicen como acumuladores. Los registros de propósito general se pueden subdividir en:

1.1.1 Registros de datos: que solo contienen datos y no se emplean en cálculos de direcciones de operando, estos registros deben poder almacenar la mayoría de tipos de datos. Ocasionalmente algunas máquinas permiten que registros contiguos puedan ser usados como uno solo para contener valores de longitud doble.

1.1.2 Registro de dirección: pueden usarse para almacenar datos aunque su función principal está dedicada a atender un modo de direccionamiento, lo que implica tener la capacidad de almacenar la dirección más grande.

1.2 Indicadores de Condición o de estado: son bits ajustados por el hardware de la CPU resultado de alguna operación, generalmente en el ALU, conocidos como “Flags o banderas o señalizadores” tal es el caso de desbordamientos, resultado nulo, positivo o negativo, esto implica que la ejecución de una instrucción no solo puede dar por resultado la modificación o almacenamiento en un registro, también se obtiene un código de condición. Estos códigos de condición se reúnen por lo general en uno o más registros, normalmente en un registro de control, estos bits son leídos por las instrucciones maquina por referencia implícita aunque no pueden ser alterado por el programador. Son utilizados por instrucciones de bifurcación para la toma de decisiones.

1.3 Acumulador: es un registro (o registros) donde se almacena temporalmente los resultados aritméticos y lógicos intermedios que serán tratados por la ALU. Su longitud puede ser de 8, 16 o 32 bits. Es un registro indispensable para toda operación de I/O (Entrada / Salida o *Input / Output*) y para operaciones de cadenas.

Ejemplo: En los microprocesadores como el 8080, 8085, 6800 y 6502 se tienen acumuladores de 8 bits, en el 68000 y Z8000 tienen registros de propósito general que se utilizan como acumuladores³¹.

1.4 Punteros de segmento: cuando la memoria se divide en segmentos, una referencia a la memoria consta de una referencia a un segmento y un desplazamiento de segmento, por ejemplo la combinación CS:IP (Segmento de código : Apuntador de instrucción) da como resultado la siguiente instrucción a ejecutar.

1.5 Registro índice: Estos registros se utilizan generalmente en operaciones con cadenas, también en operaciones como el cálculo de direcciones sumando un índice a un valor base para obtener la dirección efectiva en direccionamiento indirecto, utilizado también en direccionamiento indexado y en el pase de parámetros a otras rutinas.

Ejemplo: El Registro Índice, se utiliza en el direccionamiento indexado en micros como el 8080, 8085, 6800, 6502, Z80, 8086. En micros como el Z8000 y 68000 los registros de propósito general son utilizados como registro índice³².

1.6 Puntero de pila (SP): son registros que apuntan a alguna localidad de memoria, en particular este registro indica la siguiente posición de memoria disponible en la pila. La pila o Stack es un área reservada utilizada principalmente para el almacenamiento de direcciones de vuelta y contenido de registros. La pila

³¹ Tokheim, 1985

³² Tokheim, 1985

es de tipo LIFO – último en entrar primero en salir, se utiliza durante las llamadas a subrutina (subprogramas) y durante las interrupciones.

2. Registros de control y de estado: Estos registros sirven para controlar el funcionamiento de la CPU o para evidenciar el estado particular de la misma respecto a una instrucción, en gran parte de dispositivos no son visibles al usuario, algunos pueden ser visibles mediante instrucciones maquina ejecutadas en modo de control o de sistema operativo.

2.1 Contador de programa (PC - Program Counter): contiene la dirección de la instrucción a ejecutar, la CPU actualiza el contador de programa después de captar cada instrucción, apuntando a la siguiente instrucción. Las instrucciones de salto o bifurcación modifican el contenido del contador de programa, la instrucción captada se carga en el registro IR.

2.2 Registro de instrucción (IR – Instruction Register): contiene la instrucción captada más reciente, en este registro se analiza el código de operación y los campos del operando.

2.3 Registro de dirección de memoria (MAR – Memory Address Register): es un registro de alta velocidad y de capacidad suficiente para guardar la dirección de memoria a la que está accediendo la CPU para lectura o escritura.

2.4 Registro intermedio de memoria (MBR – Memory Buffer Register o MDR – Memory Data Register): La longitud de este registro es la misma que la de los datos o palabras en memoria, almacena el dato que se debe escribir en memoria o el dato que se ha leído de la memoria. En un sistema con organización de bus, MAR se conecta directamente al bus de direcciones y MBR (o MDR) directamente al bus de datos. Los registros visibles por el usuario intercambian repetidamente datos con MBR.

2.6 Registro de estado: Este registro reporta el estado de alguna operación aritmética o lógica, se le conoce como Registro de Estado (Status Register) o PSW (Program Status Word), los bits que lo conforman tienen dos estados posibles que tienen una significación especial acorde al procesador particular. Los indicadores de condición que lo conforman son conocidos como “flags” o “banderas” o “señalizadores” que son utilizados por instrucciones de bifurcación para la toma de decisiones, los campos más comunes son:

2.6.1 Signo: Contiene el bit de signo del resultado de la última operación aritmética realizada por el ALU.

2.6.2 Cero: Generalmente este bit se coloca o ajusta a 1 (uno) cuando el resultado de la operación es 0 (cero). Conocido generalmente como “Z”

2.6.3 Acarreo: Este bit describe que se ha sobrepasado o se ha desbordado la capacidad del registro en la suma (+) o préstamo (resta) del bit más significativo (MSB-Most Significant Bit, Bit ubicado más a la izquierda). Conocido como “C”

2.6.4 Igual: Generalmente se coloca o ajusta a 1 (uno) cuando al hacer la comparación lógica de los números almacenados en dos registros da como resultado la igualdad.

2.6.5 Desbordamiento (Overflow): indica que la operación realizada ha ocasionado un desbordamiento aritmético.

2.6.6 Interrupciones habilitadas/deshabilitadas: también denominado habilitador global de interrupciones, permite o inhabilita las interrupciones externas o internas al microprocesador.

2.6.7 Supervisor: Genera una indicación que informa si la CPU funciona en modo de supervisor o usuario. Únicamente en modo supervisor se pueden ejecutar ciertas instrucciones privilegiadas y se puede acceder a ciertas áreas de memoria.

Registros de direcciones y datos

Los registros de dirección, hacen referencia a registros especiales que se encargan temporalmente de la tarea del registro Contador de Programa al apuntar a direcciones de memoria o dispositivos de entrada / salida (E/S o I/O), estos registros conocidos como apuntadores o registros de direccionamiento indirecto son muy comunes en los microprocesadores y microcontroladores para facilitar la implementación de rutinas de lectura/escritura secuencial de memoria o perifericos entrada/salida.

Los registros de datos, generalmente son registros de propósito general destinados para el usuario o programador, en los cuales se puede almacenar temporalmente información que es requerida en el proceso en ejecución. En ocasiones estos registros están directamente relacionados al ALU, dentro de la denominada matriz de registros, pudiendo servir como acumuladores, también pueden representarse como espacios de memoria temporal en la RAM no directamente relacionados al ALU.

Memoria de datos - PILA

La PILA es una memoria de datos compuesta de varios registros, que pueden ser alojados dentro micro como una memoria especial o puede hacer parte de la memoria RAM. La PILA es utilizada para almacenar el estado de algunos registros especiales como el contador de programa, registro de estado del micro, pero generalmente se utiliza para almacenar direcciones de memoria que le permitan retornar de llamados a subrutinas (sub programas especializados) rutinas (programas) de interrupción. En algunos micros el programador debe definir las posiciones de memoria RAM que se utilizan como PILA, este tipo de memoria de lectura / escritura (R/W – Read/Write), funciona como memoria de acceso secuencial generalmente del tipo LIFO (Last Input – First Output – Último en entrar primero en salir).

Son usuales instrucciones como PUSH (introducir dato) o CALL (llamar) para escribir o introducir un dato en la PILA, instrucciones como POP (sacar) o RETURN (Volver), se utilizan para sacar de la PILA la información o dirección de memoria. La pila se gestiona gracias al registro Puntero de PILA, que es un registro especializado contador el cual contiene la dirección de la posición de memoria de la PILA como respuesta a las instrucciones de introducir o sacar de la PILA.

Diagrama de pines y Funciones

Pines de conexión de alimentación: Usualmente los microcontroladores requieren uno o varios voltajes regulados de corriente continua (Vdc) para alimentar su circuitería interna, en los micros simples se utiliza una fuente de 5Vdc, en los actuales pueden tener varios voltajes de alimentación. Identificados como Vcc, Vdd, Gnd.

Pines de conexión a reloj: El microprocesador requiere una señal de reloj, que sincroniza las microoperaciones, para lo cual se dispone, usualmente de una circuitería interna para generar dicha señal, como fuente de reloj entonces se presenta la opción de reloj interno configurable por software y reloj externo para lo cual se requiere resonadores de tipo RC, o cristal de cuarzo de baja o alta frecuencia. Identificados como X o Xtal o osc.

Pines del Bus de direcciones: En los microprocesadores se tienen pines destinados para el direccionamiento de periféricos y memoria, la cantidad de pines usualmente está directamente relacionada con el ancho del bus de direcciones, lo que a su vez determina la capacidad máxima de direccionamiento de memoria.

Identificados como A_0 hasta A_x , donde “x” determina el ancho del Bus y “A” viene de Address (Dirección en Ingles).

Ejemplo: Para identificar los pines de un Bus de direcciones de 16 bits se tendría la nomenclatura $A_0 - A_{15}$.

Pines del Bus Datos: Estos pines se relacionan con los datos o instrucciones que el microprocesador intercambia con la memoria o dispositivos periféricos, por tal razón son del tipo bidireccional, se identifican como D_0 hasta D_n con “D” Data (Datos en inglés) y “n” el ancho del bus de Datos. Estos pines pueden tener tres estados (Tri-State), es decir alto (en lógica positiva 5 Voltios), bajo (en lógica positiva 0 Voltios) y estado de alta impedancia. En los microprocesadores que maneja una arquitectura basada en longitudes de datos en Bytes o Registros de 8 bits, el ancho del bus determina la cantidad de bancos de memoria.

Ejemplo: Para un microprocesador basado en arquitectura con registros de 8 bits (1 Byte), que tienen un ancho de bus de datos de 64 bits, se tendrían 8 bancos de memoria que estarían dispuestos como: banco 0 desde D_0 hasta D_7 ; banco 1 desde D_8 hasta D_{15} ; banco 2 desde D_{16} hasta D_{23} ; banco 3 desde D_{24} hasta D_{31} ; banco 4 desde D_{32} hasta D_{39} ; banco 5 desde D_{40} hasta D_{47} ; banco 6 desde D_{48} hasta D_{55} ; banco 7 desde D_0 hasta D_{63} .

Pines de control: Estos pines hacen referencia al control de lectura o escritura (R/W) sobre la memoria o dispositivos de entrada o salida (I/O). Normalmente estos pines y otros en el microprocesador (o en general para cualquier chip en sistemas digitales) se identifican con siglas que identifican su función, además si la nomenclatura presenta una trazo o raya o comilla en la parte superior, advierte que requiere o proporciona de un pulso o estado bajo (0 Voltios) para su activación. Ejemplo de este tipo de bits podría ser, \overline{MWTC} indica escritura en memoria principal, \overline{MRTC} Indica lectura en memoria principal, \overline{IOWC} indica escritura en dispositivo I/O o entrada salida, \overline{IORC} indica lectura en dispositivo I/O o entrada salida.

Ejemplo: \overline{MWTC} indica el pin de control de escritura sobre la memoria principal y que este pin o patilla cuando está en estado BAJO (0 Voltios), indica que el dato presente en el bus de datos se va a grabar en la memoria principal.

Pin de reinicialización o reset: Este pin, puede estar identificado como \overline{RST} , lo que nos indica que si a este pin o patilla del integrado o chip, se le suministra un pulso BAJO (0 Voltios) entrará en un estado de reset o reinicialización, lo que

significa que dejara de ejecutar el programa actual y bifurcará hacia una rutina de inicialización, usualmente se dice que apunta a un vector de reset, que representa una dirección de memoria de programa con la que se ajusta el registro Contador de Programa y puede afectar la totalidad de los registros del microprocesador o microcontrolador colocándolos en un estado de cero (todos los bit del registro en cero) o en un estado inicial determinado por el fabricante, el pin **RST** es asincrónico pudiendo detener cualquier instrucción a media ejecución. El micro permanece en el estado de reset o reinicialización hasta que el estado del pin o patilla **RST** pase a un estado ALTO (5 Voltios), para continuar en la dirección apuntada por el Contador de Programa.

Ejemplo: Al colocar un pulso en BAJO en el pin **RST** el contador de programa se inicializa con todos sus bits en “cero”, representando la dirección de memoria de programa o instrucciones “0000H”, recordando que las direcciones se representan en número hexadecimal, es decir, representa 16 ceros en los 16 bits. El contador de programa comienza a ejecutar el programa a partir de la dirección “0000H”, inmediatamente el pin **RST** pase a un estado ALTO.

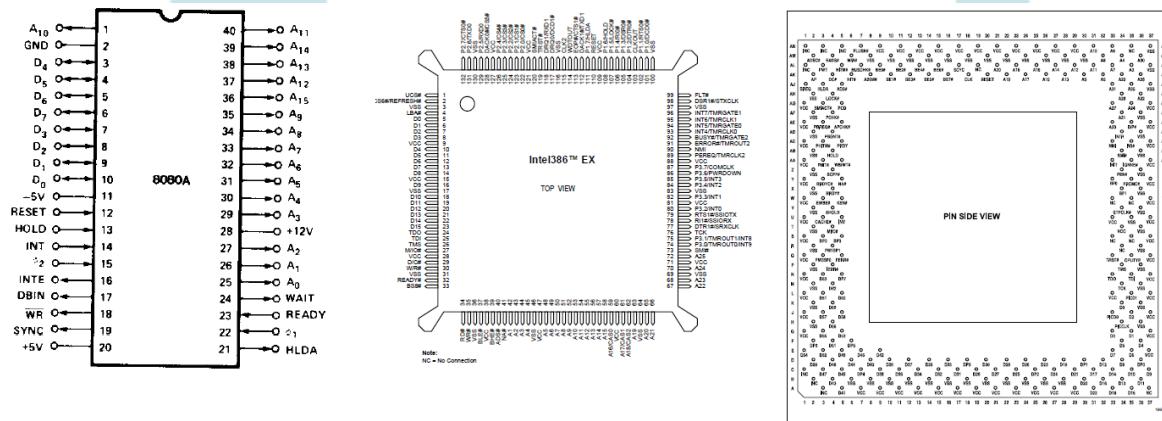
Pin o Pines de Interrupción: Estos pines son una entrada desde el “mundo exterior” al micro y le indican la ocurrencia de algún evento especial, directamente unido a una rutina o programa de interrupción. Por **Ejemplo** el pin o patilla **INT** identifica un pin o patilla de interrupción externa que requiere de un pulso ALTO (5 Voltio) para que al micro se le envíe una señal clara de un evento externo, si el programa ha incluido dicha interrupción (por ejemplo la acción sobre un teclado), se incluye un vector de interrupción, el cual tiene la acción de ajustar el registro Contador de Programa a un valor predeterminado de dirección de programa o instrucciones, donde se encuentra el programa o rutina de interrupción. Es usual que la PILA (Porción de memoria RAM) almacena la dirección donde se encontraba ejecutando el programa antes de la interrupción, la rutina o programa de interrupción generalmente debe encargarse de guardar por instrucciones de software el estado de los registros Acumuladores, Registro de Estado y flags o banderas de Estado, generar la respuesta a la interrupción y la recuperación de los valores previos de registros Acumuladores, Registro de Estado y flags o banderas de Estado, por software debe indicarse mediante instrucciones la terminación de la rutina o programa de interrupciones, recuperando la dirección almacenada en la PILA en donde se encontraba el programa antes de la interrupción.

Hojas de datos (Datasheet)

Las hojas de datos, hojas técnicas o “datasheet”, se refieren a la especificación del fabricante respecto a un dispositivo electrónico, como para microprocesadores o microcontroladores, el Datasheet, es proporcionado por el fabricante del dispositivo, generalmente contienen información resumida del producto, empaquetamiento del CI (Circuito Integrado), descripción y función de pines, alimentación, consumo, frecuencia, voltaje, corriente o potencia en el dispositivo, esquema de la arquitectura del dispositivo, diagrama de tiempos, repertorio de instrucciones, sistemas mínimos de prueba, entre otros.

Los encapsulados, hacen referencia a la presentación y conexión del dispositivo en su forma física, son del tipo DIP (Dual In-line Package – Encapsulado de doble línea de pines), para tamaños de 8, 18, 20, hasta 40 pines, se refiere a dos líneas de pines paralelas, en la que se identifica el primer pin o patilla con una muesca o punto ubicada a la izquierda del texto impreso en el integrado, incrementando en número siguiendo una secuencia en contra de las manecillas del reloj (formando una “U”), se presentan en encapsulado QFP (Quad Flat Package – Encapsulado cuadrado plano), para montaje superficial, la identificación del pin o patilla uno (1) de los demás pines siguen la misma indicación que en el encapsulado DIP.

Figura 38. Ejemplos de Sócalos y tipos de encapsulado, de izquierda a derecha 8080, 80386, Pentium.



Para micros cercanos o mayores a 100 pines, se tienen encapsulados como los PGA (Pin Grid Array) la cual presenta un arreglo de pines en una de sus caras, estos utilizan sócalos especiales según el micro, por ejemplo para el micro 80386 se utiliza un “Socket 7”, los encapsulados PGA originalmente con base en losa de porcelana, tienen variantes plásticas como la PPGA (Plastic Pin Grid Array) o también la FCPGA (Flip-Chip Pin Grid Array), los encapsulados BGA (Ball Grid Array) utilizan bolitas de estaño (soldadura) en una cara del micro para facilitar la soldadura del mismo a la placa base y finalmente el tipo de encapsulado LGA

(Land Grid Array) consiste en un micro que en una de sus caras no presenta ni bolitas de estaño (soldadura) o pines, pero presenta contactos o conductores superficiales o pads (puntos) en oro para hacer contacto con la placa base.

Los encapsulados en plástico se utilizan en micros de bajo consumo o poco disipación de calor, los de empaque cerámico se utilizan en micros que operan a altas temperaturas. Algunas hojas técnicas en el caso de microprocesadores o microcontroladores, incluyen registros internos, como el de estado, pueden describir el circuito decodificador, de control y temporizadores, resumen del repertorio de instrucciones e incluso ejemplos de programas, recursos muy valiosos al momento de pretender hacer un diseño basado en dicho dispositivo. Es de aclarar que los Datasheet generalmente estan en idioma Ingles.

CAPITULO 2: FAMILIAS DE MICROPROCESADORES

Introducción

Desde la aparición de primer microprocesador, fabricado por Intel, aparecen varias familias de microprocesadores, originadas en nuevos proyectos de desarrollo de microprocesadores o como derivados de los primeros fabricados por Intel. Se tienen varias formas de clasificar los microprocesadores, una forma práctica para establecer una clasificación, es teniendo en cuenta la longitud del registro o registros de trabajo o acumulador, también denominado tamaño de palabra. En este capítulo se exponen las generalidades de las principales familias de microprocesadores para luego categorizarlas según su bus de datos en 8, 16, 32 y 64 bits, presentando las características más sobresalientes de los microprocesadores más representativos en cada categoría.

Lección 6: Principales Familias de Microprocesadores

Las familias de microprocesadores tienen que ver con la empresa que los fabrica, la tecnología que implementan y el impacto comercial del microprocesador. Las familias más relevantes se caracterizan por el tipo de arquitectura, velocidad de procesamiento, innovaciones en su diseño, aplicación y compatibilidad con el hardware y software dispuesto en el mercado. En los siguientes apartados se presentan las familias Intel, AMD, Zilog, Cyrix, Motorola, Power PC y Sparc.

Familia de microprocesadores INTEL

Intel Corporation es la compañía más grande en fabricación de chips semiconductores, es pionera en el desarrollo y comercialización de microprocesadores. Intel es responsable de fabricar el primer microprocesador en un chip, el Intel 4004 (i4004), junto con el conjunto de chips de soporte, continúa sus avances con el desarrollo del 8080 de 8 bits lanzado en abril de 1974, con una

reloj de 2 MHz, bus de direcciones de 16 bits y de datos de 8 bits, un puntero de pila de 16 bits y contador de programa de 16 bits, posteriormente Intel diseña el 8085 binariamente compatible con el 8080 pero que tenía menos exigencia de hardware, sigue una serie de avances y desarrollo de nuevos microprocesadores con características que mejoran el desempeño con respecto a sus predecesores.

En los 80's aparecen los microprocesadores de 16 bits más potentes con la incursión al mercado del 8086 de Intel en 1978, utilizado por IBM para la fabricación de la gamma de IBM PC serie PS/2. Intel prosigue el desarrollo del 8088 para IBM XT, 80186, el 80286 utilizado para IBM AT, el 80386, 80486 y Pentium, que fue un nombre comercial para evitar la duplicación de su nomenclatura por parte de AMD y Cyrix, más adelante sigue con el desarrollo y producción de Pentium II, Pentium III, Pentium IV, hasta los presentes microprocesadores multinúcleo. En la actualidad, tiene una cuota del mercado mundial de los microprocesadores muy grande, ya que sus procesadores se utilizan en la mayoría de los computadores compatibles PC.

x86: se denomina como x86 a la arquitectura básica que comparten un grupo específico de microprocesadores de la familia Intel y sus compatibles, dentro del grupo de x86 están microprocesadores de 16 y 32 bits, aunque algunas veces se categoriza como x86 a los microprocesadores de 16 bits y x86-32 a los de 32 bits, dentro del grupo x86 algunos se identifican claramente por la terminación de sus nombres, como el 8086 (i8086), 80286 (i286), 80386 (i386) y 80486 (i486), los microprocesadores siguientes al i486 adoptaron nombres no numéricos que hacen referencia a la marca, logotipo o nombre clave, comenzando con la denominación Pentium.

IA32: (Intel Architecture 32 bits), también denominada x86-32, es una arquitectura de microprocesadores de 32 bits, introducida en 1985 por Intel en los microprocesadores 80386 y en los siguientes como el i486, *Pentium MMX, Pentium Pro, Pentium II, Pentium III, Pentium IV y Pentium D*. La arquitectura IA32 se categoriza como arquitectura CISC, aunque con el avance en el diseño de los microprocesadores se adopta la clasificación post-RISC. IA32 implementa dos modelos de acceso a memoria el llamados Modo Real con límite en el acceso a memoria, utilizado en sistemas operativos DOS y el Modo Protegido con el que se puede acceder a toda la memoria, utilizado por sistemas operativos como Windows, Linux y OS/2. Otras características de la arquitectura IA32 con respecto a su predecesora es que incluye una unidad de punto flotante (FPU), tienen ocho registros de 32 bits de propósito general para aplicaciones en uso, permite una gestión avanzada de memoria, mediante la segmentación, paginación y soporte de

memoria virtual. Desde el Pentium MMX la arquitectura IA32 cuenta con soporte para operaciones matriciales complejas, muy utilizadas en aplicaciones gráficas y multimedia. Las instrucciones de la arquitectura IA32 fueron evolucionando con el desarrollo de nuevos microprocesadores que agregaron nuevos registros con conjuntos de instrucciones propias, como MMX, 3DNow!, SSE, SSE2, SSE3, SSSE3. Finalizando con la extensión a 64 bits.

Familia de microprocesadores AMD

AMD (Advanced Micro Devices), es una compañía estadounidense de semiconductores, fundada en 1969 por un grupo conformado por Jerry Sanders y varios ejecutivos. La compañía comienza a producir circuitos integrados lógicos, en 1975 memorias RAM, produjo un procesador clon compatible con el i8080, en 1982 firmó un contrato con Intel para producir procesadores 8086, 8088 y 80286, donde se incluía AMD como logotipo e Intel como marga registrada, en 1986 se produjo la división con Intel, al reusarse este último a revelar detalles del i386. Después de la ruptura con Intel, AMD incursiona en 1995 como fabricante de microprocesadores con el lanzamiento del AMD K5, haciendo referencia a “Kryptonite”, el K5 se lanza como competidor del Pentium pero la falta de rendimiento y algunos errores causan que no tuviera éxito. AMD continúa con la fabricación de otros microprocesadores como el K6, K6-II, K6-III, donde prevalecía como proveedor de microprocesadores de más bajo precio en relación a los Intel. El microprocesador Athlon (K7) le significó a la compañía un avance notorio por el rendimiento y bajo costo del microprocesador, entrando a competir con Intel. Con el AMD Athlon 64 (K8), con arquitectura AMD64 logra superar el rendimiento de su competidor Pentium IV, sus avances continúan hasta llegar a la última gamma multinucleo. En julio de 2006 AMD compra la compañía ATI consolidándose como una compañía fabricante de microprocesadores, chipsets, GPU y chips gráficos para computadores, celulares y consolas. Actualmente AMD es la segunda compañía a nivel mundial en fabricación de microprocesadores compatibles Intel, los microprocesadores AMD tienen un alto rendimiento en aplicaciones con imágenes y gráficos 3D, sonido y video.

Familia de microprocesadores de Zilog

Zilog Inc. tiene gran éxito con la fabricación de su primer microprocesador el Z-80, posteriormente produce algunos procesadores de 16 y 32 bits pero sin gran

impacto, la compañía se dedica entonces a la fabricación de microcontroladores basados en el núcleo del Z-80. En el año 1995 crean el V-Chip y en el 2001 Zilog lanza el eZ80. En la última década se ha enfocado al desarrollo de procesadores orientados a telecomunicaciones y es líder en el mercado de mandos infrarrojos.

Familia de microprocesadores CYRIX

Cyrix comienza operaciones en 1988, se caracterizó por proveer coprocesadores matemáticos para los procesadores 286 y 386, esta empresa fundada por ex empleados de Texas Instruments, se decide por fabricar en 1992 el Cyrix 486 (486SLC y 486DLC) compatibles en pines con los i386 (i386SX e i386DX respectivamente), convirtiéndola en el tercer fabricante de procesadores Intel compatibles. Cyrix se caracterizó por sus diseños avanzados y originales, provocando algunos problemas de compatibilidad, lo que sumado a una unidad de punto flotante limitada, que no le permitía ejecutar eficientemente operaciones matemáticas complejas, teniendo un bajo rendimiento en gráficos 3D y juegos, no le permitió dominar el mercado con los procesadores que desarrolló, limitando sus ventas a integradores, productos de bajo costo y actualización de equipos de cómputo. Como dato curioso Cyrix solo diseñaba microprocesadores, no los fabricaba, esta tarea la realizaba con alianzas a empresas como Texas Instruments, SGS-Thomson (Actualmente STMicroelectronics), IBM Microelectronics y National Semiconductor.

Figura 39. Cyrix 6x86, Cyrix MII, Cyrix MediaGX y Cyrix III³³



Cyrix junto con AMD desarrollaron la nomenclatura PR (Performance Rating) utilizada como forma de favorecer los productos fabricados por estas empresas comparándolas en eficiencia y rendimiento con los microprocesadores de Intel. Los Microprocesadores Cyrix en sus primeros modelos tuvieron problemas por el

³³ Extraído el 10 de Julio de 2011 desde <http://www.cpu-world.com/CPUs/6x86/>, <http://en.wikipedia.org/wiki/MediaGX>, http://en.wikipedia.org/wiki/Cyrix_III

alto voltaje que requerían, situación que fue corregida en modelos posteriores. En 1996 Cyrix lanza el microprocesador MediaGX y en 1997 se fusiona con National Semiconductor y en 1999 Cyrix fue vendida a VIA Technologies quien utilizó el nombre de Cyrix en un procesador diseñado por Centaur Technology, también propiedad de VIA Technologies, pensando que el nombre de Cyrix en el procesador tendría mejor reconocimiento que el nombre de Centaur o VIA.

Familia de microprocesadores Motorola

Motorola utiliza su primer microprocesador, el 6800 y lo utiliza como base para la construcción de varios microcontroladores como el Motorola 6805, 6807, 6808, 68HC11 y 68HC12. En 2005 Motorola crea la empresa Freescale Semiconductor Inc. Para la producción de semiconductores. En 1991 surge la alianza AIM como un consorcio entre Apple, IBM y Motorola, se desarrolla una nueva familia de microprocesadores los “Power PC” con arquitectura RISC, los cuales se utilizan en las computadoras Apple de IBM actuales.

Familia de microprocesadores Power PC

El Power PC (Performance Optimized With Enhanced RISC Performance Chip), también conocidos como PPC, con una arquitectura tipo RISC (*Reduced Instruction Set Computer*), desarrollado por la Alianza AIM, compuesta por las empresas Apple, IBM y Motorola (de ahí las siglas AIM). Fabricados por IBM y Freescale Semiconductor (división de semiconductores y microprocesadores de Motorola), utilizados en computadores Macintosh de Apple Computer hasta 2006 y en varios de modelos de IBM.

Figura 40. Construcción Power PC 601, PPC750CXE, PowerPC 7450³⁴



³⁴ Extraído el 10 de Julio de 2011 desde <http://www.tecmiente.comuf.com/?p=1583> y http://es.wikipedia.org/wiki/Archivo:IBM_PowerPC601_PPC601FD-080-2_top.jpg, http://es.wikipedia.org/wiki/PowerPC_G3, http://es.wikipedia.org/wiki/PowerPC_G4.

El PowerPC se diseñó con base en la arquitectura POWER de IBM, incorporando algunos componentes y características del microprocesador Motorola 68000, con el fin de dar compatibilidad con la arquitectura de computadores Apple. En la actualidad esta arquitectura se utiliza en implementaciones de Nintendo (Gamecube y Wii), Sony (Playstation 3) y Microsoft con su Xbox360.

Familia de microprocesadores SPARC

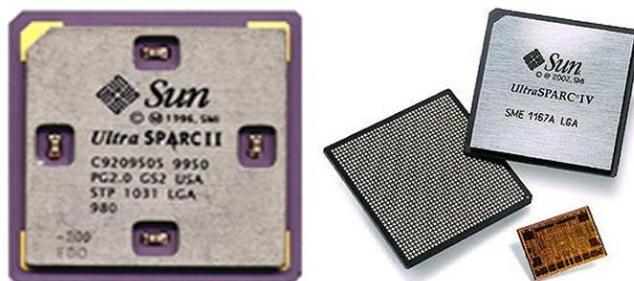
SPARC proviene del acrónimo de Scalable Processor ARChitecture, presenta una arquitectura RISC (Reduced Instruction Set Computer), por lo que presenta un conjunto reducido de instrucciones. La arquitectura escalable permite definir versiones posteriores de procesadores con mayor cantidad de características, conservando la compatibilidad con programas de versiones anteriores. La arquitectura SPARC originalmente fue diseñada por Sun Microsystems, con el liderazgo del Ingeniero Kaa en 1985, se fundamentó en diseños del RISC I y RISC II de la universidad de California en Berkely definidos en 1980 y 1982, la arquitectura SPARC se lanza a mediados de 1987.

SPARC International Inc. Se establece como una organización en 1989 que promueve la arquitectura SPARC, posteriormente Sun Microsystems libera las especificaciones de la arquitectura SPRAC, con lo que se convierte en la primera arquitectura RISC abierta, para que otros fabricantes puedan desarrollar diseños propios. Sun Mycrosystems licenció a fabricantes como Texas Instruments, Cypress Semiconductor, Fujitsu, LSI entre otros. Como innovación la arquitectura SPARC utiliza una “ventana de registros”, permitiendo compiladores de alto rendimiento y reduciendo significativamente la memoria en las instrucciones load/store en relación con otras arquitecturas RISC, ventajas que se pueden apreciar en la ejecución de programas grandes.

La CPU SPARC, se compone de una unidad entera UI (Integer Unit), para la ejecución básica y una FPU (Floating-Point Unit), para ejecutar operaciones y cálculos reales, las unidades UI y FPU pueden estar o no integradas en el mismo chip. Aunque no hace parte formal de la arquitectura, Sun Microsystems implementa una unidad de memoria (MMU), junto con una gran caché de direcciones virtuales, para instrucciones y datos, que están dispuestas periféricamente sobre un bus de datos y direcciones de 32 bits. La arquitectura de 32 bits SPARC fue originalmente diseñada en estaciones de trabajo Sun-4, que son un tipo de estación de trabajo y servidor Unix lanzado en 1987, como reemplazo de los sistemas Sun-3 basados en procesadores Motorola 68000.

Posteriormente procesadores SPARC desarrollados por Sun, Solbourne y Fujitsu, diseñados para 64 bits, se utilizan en servidores SMP (Symmetric Multi-Processing) en donde dos procesadores de características idénticas se conectan y comparten una única memoria central. En la Figura 41. Se muestra el SPARC.

Figura 41. Sun Ultra SPARCII y Ultra SPARC IV³⁵



Se han generado tres versiones de la arquitectura SPARC. La primera V7 (32 bits) pública en 1986. La segunda V8 (32 bits) pública en 1990, que introduce operaciones de producto y cociente de números enteros, la versión V8 sirve como base para el estándar IEEE 1754-1994 para la arquitectura de microprocesadores de 32 bits. La tercera V9, pública en 1993 posee una arquitectura de 64 bits. En 2002 Fujitsu y Sun liberan JpS1 (Joint Programming Specification 1), que presenta las funciones del procesador, los procesadores que implementan JpS1 son, en Sun Microsystems el UltraSPARC III y en Fujitsu el SPARC64.

Tabla 8. Generaciones SPARC³⁶

Generación	Familia	Especificación general
Primera generación (1987)	SPARC	Frecuencia de reloj de 16 a 50MHz. Diseño escalar
Segunda generación (1992)	SUPER SPARC	Frecuencia de reloj de 33 a 50MHz. Diseño super escalar.
Tercera generación (1996)	ULTRA SPARC II	Arquitectura super escalar de 4 etapas y de 64 bits. Cinco unidades de punto flotante. Velocidades entre 250 y 300 Mhz
Cuarta generación (2004)	Advanced Product Line (APL)	Como base de acuerdo comercial entre Sun Microsystems y Fujitsu, con arquitectura Super escalar compatible con SPARC V9 de 64 bits. Velocidad entre 1,35 y 2,7 GHz.

³⁵ Extraído el 10 de Julio de 2011 desde http://es.wikipedia.org/wiki/Archivo:Sun_UltraSPARCII.jpg y http://en.wikipedia.org/wiki/File:UltraSparcIV_medium.JPG

³⁶ Extraído el 10 de Julio de 2011 desde http://es.wikipedia.org/wiki/Sun_SPARC

En 2006 Sun Microsystems libera la arquitectura UltraSPARC, en 2007 publica una actualización la UltraSPARC 2007. Las computadoras con procesadores SPARC utilizan sistemas operativos Solaris, SunOS, NEXTSTEP, RTEMS, FreeBSD, OpenBSD, NetBSD y Linux, algunos ejemplos de procesadores basados en SPARC son el microSPARC (Tsunami), SuperSPARClI (Voyager), UltraSPARC I (Hornet), UltraSPARC III (Sabre), UltraSPARC IV (Jaguar), UltraSPARC IV+ (Panther), UltraSPARC T1 (Niagara), UltraSPARC T2 (Niagara II), UltraSPARC RK (Rock), etc. La Tabla 8. Presenta algunas de las generaciones del SPARC.

Lección 7: Microprocesadores de 8 bits

La línea de microprocesadores de 8 bits, es considerada como el núcleo para otras familias y fabricantes, el INTEL 8080 e INTEL 8085, fueron utilizados en computadores a finales de los 70s y comienzos de los 80s, específicamente el diseño del microprocesador 8080 fue la base para la creación del Z80 microprocesador que superó ampliamente las prestaciones y mercado del 8080.

En la actualidad hay muchas versiones clónicas de esos microprocesadores inclusive núcleos de microcontroladores siguen como diseño base en la CPU el diseño de los Z80. En los siguientes párrafos se presenta la descripción general de algunos de los microprocesadores de 8 bits más representativos.

Intel 8080 y 8085

Son procesadores de 8 bits fabricados por Intel a mediados de los 70. El 8085 era binariamente compatible con el 8080, pero exigía menos soporte de hardware; de esta forma permitía unos sistemas de microordenadores más simples y más económicos. El número 5 de la numeración del procesador 8085 proviene del hecho que solamente requería una alimentación de 5 voltios, no como el 8080 que necesitaba unas alimentaciones de 5 y 12 voltios. Entre las características del 8085 se destaca su bus de datos multiplexado, es decir, comparte los pines del bus de datos con los del bus de direcciones.

Figura 42. Intel 8080 y 8085 pines³⁷

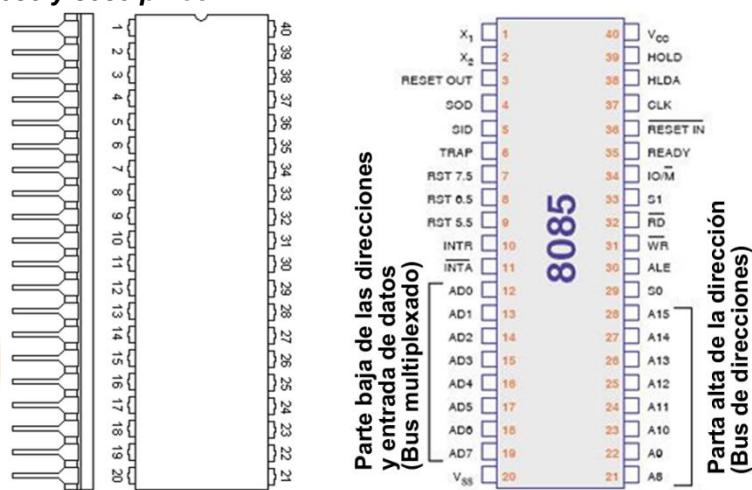


Tabla 9. Intel 8080 descripción de pines³⁸

Pin	Nombre	Descripción
1	A10	Bus de direcciones
2	GND	Referencia de tierra. Todas las tensiones se miden con respecto a este punto.
3	D4	Si SYNC = 0: Bus de datos. Si SYNC = 1: Señal de control que indica salida a periférico.
4	D5	Si SYNC = 0: Bus de datos. Si SYNC = 1: Señal que indica si el uP está en ciclo de búsqueda de instrucción.
5	D6	Si SYNC = 0: Bus de datos. Si SYNC = 1: Señal de control que indica entrada de periférico.
6	D7	Si SYNC = 0: Bus de datos. Si SYNC = 1: Señal de control que indica lectura de memoria.
7	D3	Si SYNC = 0: Bus de datos. Si SYNC = 1: Señal que indica que el uP se ha detenido.
8	D2	Si SYNC = 0: Bus de datos. Si SYNC = 1: Señal que indica que se realiza una operación con el stack.
9	D1	Si SYNC = 0: Bus de datos. Si SYNC = 1: Modo lectura/escritura.
10	D0	Si SYNC = 0: Bus de datos. Si SYNC = 1: Señal de reconocimiento de interrupción.
11	-5V	Una de las tres patas de alimentación del 8080.
12	RESET	Señal de borrado de todos los registros internos del 8080. Para ello, ponerlo a uno durante tres ciclos de reloj como
13	HOLD	Sirve para poner los buses en alta impedancia para el manejo de DMA (acceso directo a memoria).

³⁷ Extraído el 10 de Julio de 2009 desde <http://www.alpertron.com.ar/8080.HTM>

³⁸ Extraído el 10 de Julio de 2009 desde <http://www.alpertron.com.ar/8080.HTM>

14	INT	Señal de pedido de interrupción.
15	CLK2	Señal de reloj (debe venir del generador de reloj 8224).
16	INTE	Señal de aceptación de interrupción.
17	DBIN	Indica que el bus de datos está en modo lectura.
18	/WR	Indica que el bus de datos está en modo escritura.
19	SYNC	Este pin se pone a uno cuando comienza una nueva instrucción.
20	+5V	Una de las tres patas de alimentación del 8080.
21	HLDA	Reconocimiento de HOLD.
22	CLK1	Señal de reloj (debe venir del generador de reloj 8224).
23	READY	Sirve para sincronizar memorias o periféricos lentos (detiene al 8080 mientras se lee o escribe el dispositivo).
24	WAIT	Cuando vale "1", el 8080 está esperando al periférico lento.
25-	A0 - A1	Bus de direcciones.
28	+12V	Una de las tres patas de alimentación del 8080.
29-	A3 -	Bus de direcciones.

Tabla 10. Descripción general de pines³⁹

NOMBRE DEL PIN	TIPO	DESCRIPCIÓN
Líneas de dirección (A ₁₅ - A ₀)	Salida	Provee la dirección de memoria o número de dispositivo entrada/salida. A ₀ denota el bit menos significativo
Líneas de datos (D ₇ - D ₀)	Entrada/Salida	Provee comunicación bidireccional de instrucciones o datos entre la CPU y la memoria o dispositivos de entrada/salida. D ₀ es el bit menos significativo.
Señal de sincronismo (SYNC)	Salida	La señal de sincronismo indica el comienzo de cada ciclo máquina.
Señal de entrada de datos (DBIN)	Salida	Indica a los dispositivos externos que el bus de datos está en modo de entrada.
Listo (READY)	Entrada	Indica al 8080a que hay un dato válido de la memoria o dispositivo de entrada/salida en el bus de datos. Esta señal permite sincronizar la CPU con dispositivos más lentos. Si después de enviar una dirección no se recibe la señal de READY, la CPU entra en estado de espera hasta recibirla.
Espesa (WAIT)	Salida	Indica que la CPU se encuentra en estado de espera.
Escritura (WR)	Salida	Señal de control de escritura para la memoria o dispositivo de entrada/salida.
Reposo (HOLD)	Entrada	La CPU entra en estado de reposo, es decir, permite que un dispositivo externo tenga el control del bus de direcciones y de datos.
Reconocimiento de reposo (HLDA)	Salida	Esta señal aparece en respuesta a la señal HOLD e indica que el bus de datos y direcciones entran en estado de alta impedancia.
Habilitación de interrupción (INTE)	Salida	Indica el estado interno del flip-flop de habilitación de interrupción. Cuando es 0 se deshabilita la

³⁹ Téllez, 2007

		interrupción y cuando es igual a 1 es habilitada. Cuando la CPU se reinicia las interrupciones son deshabilitadas.
Petición de interrupción (INT)	Entrada	La CPU reconoce la petición de interrupción y la atiende al final de la actual instrucción o mientras esté detenido. Si se encuentra en estado de reposo o deshabilitadas las interrupciones no atiende la petición.
Reinicio (RESET)	Entrada	Mientras la señal esté activa el contador de programa es borrado. Despues se reanudará en la posición de memoria 0.
Tierra (V _{ss})	Alimentación	Tierra o referencia
V _{DD}	Alimentación	+12V ± 5% V
V _{CC}	Alimentación	+5V ± 5% V
V _{BB}	Alimentación	-5V ± 5% V
Fases de reloj (ϕ_1, ϕ_2)	Reloj	2 fases externas de reloj suplementarias

Figura 43. Arquitectura Intel 8080⁴⁰

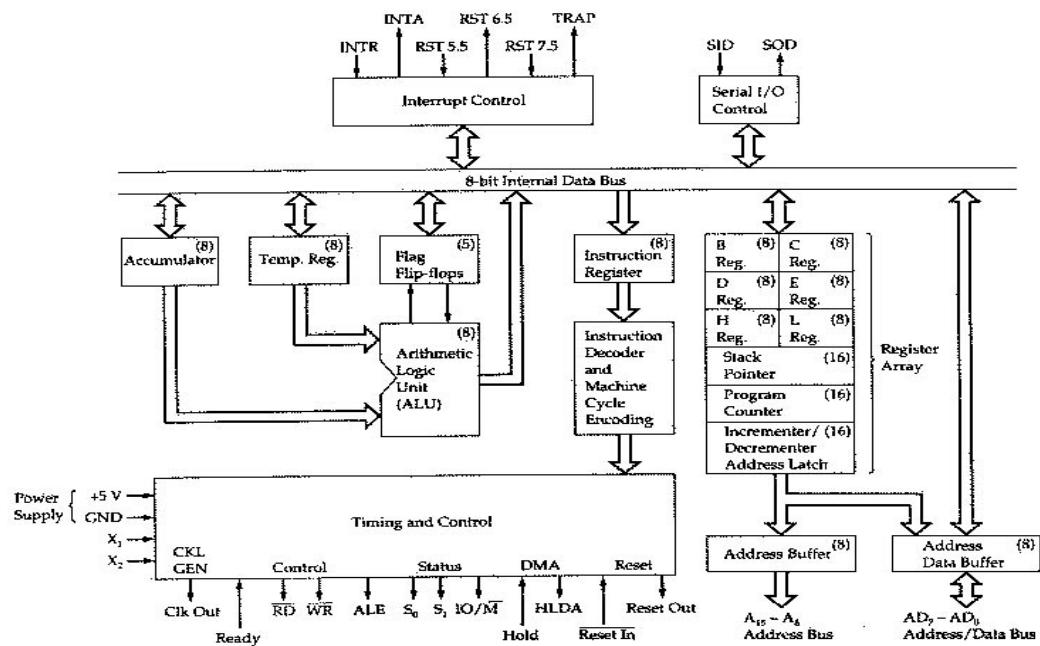


Figura 44. Arquitectura Intel 8085⁴¹

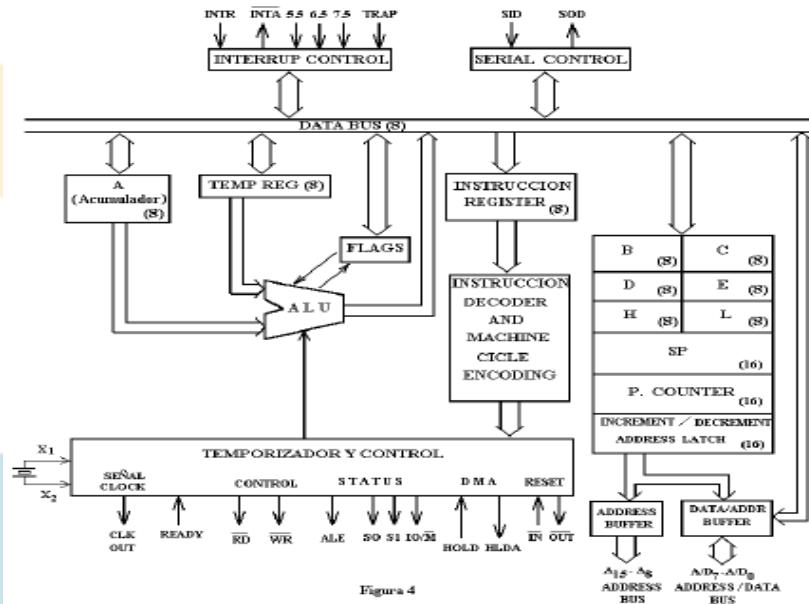


Figura 4

Figura 45. Set instrucciones 8085⁴²

Conjunto de instrucciones del microprocesador 8085

Instrucciones de Transferencia de Datos

MOV reg dest., reg fuent.
 MVI reg dest., data.
 OUT ###

Instrucciones de Aritmetica Binaria Entera

NEG registro
 ADD registro
 SUB registro

Instrucciones de Operaciones Lógicas

ANA registro
 ORA registro
 XRA registro

Instrucciones de Desplazamientos y Rotaciones

RLC registro
 RRC registro

Control de Programa

Instrucciones de Salto Incondicional	Instrucciones de Salto Condicional	Instrucciones de Comparación
JMPyyy	JZxxx JCxxx JPxxx	CMPregistro

⁴¹ Extraido el 10 de Julio de 2009 desde <http://www.monografias.com/trabajos32/microprocesador-8085/microprocesador-8085.shtml>

⁴² Extraido el 10 de Julio de 2009 desde <http://www.monografias.com/trabajos32/microprocesador-8085/microprocesador-8085.shtml>

AMD – Am9080

Es una copia, originalmente sin licencia, del Intel 8080. Posteriormente se consiguió un acuerdo con Intel para fabricarlo y fue renombrado a Am9080. Las primeras versiones del Am9080 fueron puestas a la venta en abril de 1974, y funcionaban a una velocidad de reloj de 2 MHz⁴³.

Motorola 6800

El MC6800 es un microprocesador fabricado por Motorola, lanzado al mercado en 1975, su nombre “6800” proviene de la cantidad de transistores utilizados para su fabricación, su presentación física es en DIP de 40 pines, necesita una tensión de alimentación de 5 Voltios, considerado el primer microprocesador en implementar un Registro Índice, contaba con un set de 78 instrucciones, compatible con dispositivos TTL, con frecuencia de reloj de 1MHz y un tiempo de ejecución de instrucción entre 2 y 12 microsegundos, tiene un bus de direcciones de 16 líneas y un bus de datos bidireccional de 8 bits, fue el núcleo de sistemas como el MITS Altair 680.

Con el desarrollo del MC6800 se derivan otros microprocesadores como el MC6802, 6809 y microcontroladores como 6801, 6803, 6805, 6807, 6808, 68HC11 y 68HC12.

MOS 6502

La empresa MOS Technology en 1975 saca al mercado el microprocesador MOS 6502, diseño proveniente de los diseñadores que trabajaron en Motorola en el proyecto MC6800 y fueron contratados por MOS Technology, se caracterizó por ser la más económica del mercado en esos días (\$79 en comparación con los \$179 de Intel y Motorola), por ser más rápido, en un principio se diseña el 6501 por el equipo original del MC6800 compatible en patillaje con el MC6800, lo que trabajo una demanda a MOS Technology por parte de Motorola, creando el MOS 6502, diferenciado solamente por la disposición del patillaje.

Es un procesador de 8 bits en el bus de datos y 16 bits en el bus de direcciones, frecuencia de 1 MHz, registros de 8 bits en el acumulador “A”, índice “X” e “Y”, registro de estado “SR”, puntero de pila “SP” y registro de 16 bits para el contador de programa “PC”, utiliza el método “pipelined” que permiten la ejecución de instrucciones casi en forma paralela. El MOS 6502 fue utilizado en computadoras como el KIM-1, Rockwell AIM-65, Apple II y en consolas de juego como Atari,

⁴³ Extraído el 10 de Julio de 2009 desde <http://www.stormloader.com/users/megaterran/9080.htm>

Nintendo Famicom, Nitendo Entertainment System (NES), entre otros. El diseño del MOS 6502 se utilizó en el desarrollo del ARM – RISC.

Zilog Z80

En 1976 la compañía Zilog lanza al mercado el Z80, microprocesador de 8 bits, con una frecuencia inicial de 2,5 MHz, fuente de alimentación de 5 Voltios, instrucciones de manipulación de bits, cuenta con 80 instrucciones adicionales a la del 8080 para un total de 156 instrucciones, lo que lo hace compatible en código con el 8080 pero más económico, haciéndolo muy popular en implementaciones con software CP/M.

El núcleo del Z80 dio origen al desarrollo de otros micros como el Z80A, Z80B, Z80H, Z180, Z80182, Z280, eZ80 y en otras empresas el núcleo se utiliza para el desarrollo de variedad de microprocesadores como el Hitachi HD64180, Sharp LH-0080, Nec uPD780C y en microcontroladores.

SC/MP de National Semiconductor

SC/MP (Simple Cost-effective Micro Processor – Microprocesador simple y rentable), disponible a comienzo de 1974, con un bus de direcciones de 16 bits y de datos de 8 bits, registros de 8 bits para el Registro Acumulador “AC”, Registro de extensión “Ex”, Registro de estado, tres (3) registros índice de 16 bits, no posee Puntero de Pila y contador de programa de 16 bits, implementa memoria paginada al poseer un contador de programa con reinicio de 12 bits (direcciona 4096 registros o 4K de memoria), con posibilidad de alterar los 4 bits superiores, permitiendo 16 páginas de memoria cada una de 4KBytes.

Lección 8: Microprocesadores de 16 bits

Con el primer microprocesador comienza la evolución y desarrollo de nuevos dispositivos, con capacidad y eficiencia mayor a la de sus predecesores, después de los microprocesadores de 8 bits con los que se implementan las primeras computadoras digitales modernas y consolas de juego, surge la necesidad de más capacidad de procesamiento y gestión de periféricos, para lo cual se debe aumentar el tamaño del bus de direcciones pero en especial el bus de datos, aparece una nueva gama de microprocesadores llamados microprocesadores de 16 bits, los párrafos siguientes profundizan en este tema.

Intel 8086, 8088, 80186 y 80188

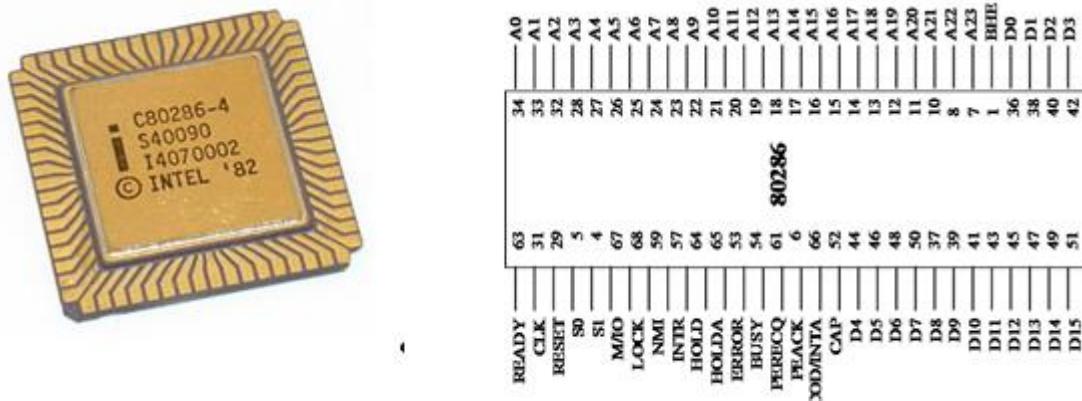
8086 y 8088: son dos microprocesadores de 16 bits diseñados por Intel en 1978, iniciadores de la arquitectura x86. La diferencia entre el 8086 y el 8088 es que el 8088 utiliza un bus externo de 8 bits, para el empleo circuitos de soporte al microprocesador más económicos, en contraposición al bus de 16 bits del 8086.

80186 y 80188: estos dos microprocesadores fueron desarrollados por Intel alrededor de 1982. Los 80186 y los 80188 son una mejora del Intel 8086 y del Intel 8088 respectivamente. Al igual que el 8086, el 80186 tiene un bus externo de 16 bits, mientras que el 80188 lo tiene de 8 bits como el 8088, para hacerlo más económico. La velocidad de reloj del 80186 y del 80188 es de 6 MHz. Ambos microprocesadores no fueron muy usados en ordenadores personales, sino que su uso principal fue como procesadores empotrados. Con el 80186 y el 80188 se introdujeron ocho nuevas instrucciones al conjunto de instrucciones x86.

Intel 80286

80286: (llamado oficialmente iAPX 286, también conocido como i286 o 286) es un microprocesador de 16 bits de la familia x86, que fue lanzado al mercado por Intel el 1 de febrero de 1982. Las versiones iniciales del i286 funcionaban a 6 y 8 MHz, pero acabó alcanzando una velocidad de hasta 20 MHz. El i286 fue el microprocesador más empleado en los IBM PC y compatibles entre mediados y finales de los años 80.

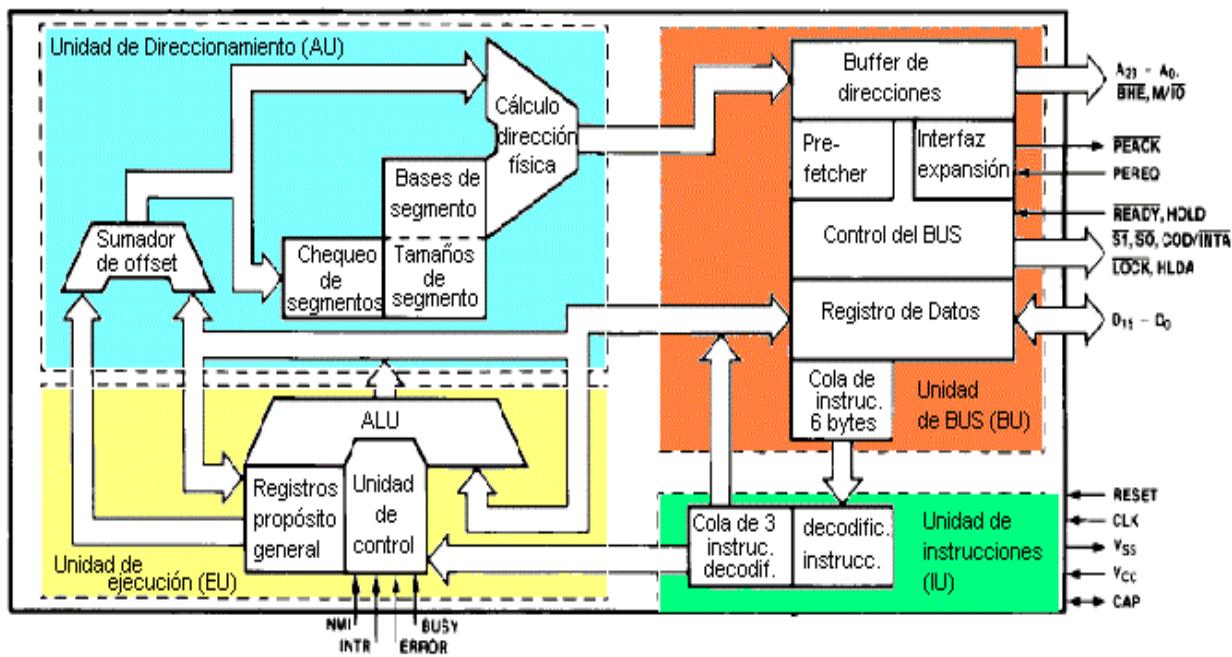
Figura 46. 80286⁴⁴ y pines del 80286



⁴⁴ Extraído el 10 de Julio de 2009 desde <http://www.cpu-world.com/CPU%5Fs/80286/index.html>

Es un microprocesador de 16 bits, su relevancia radica en la implementación de este microprocesador en los primeros equipos de cómputo personal y compatible, se puede decir que la computadora personal nació con la implementación de estos microprocesadores como núcleos del sistema.

Figura 47. Arquitectura del 80286⁴⁵



AMD – Am286

Es un procesador clon del Intel 80286, fabricado con autorización de Intel, motivado porque IBM quería que Intel tuviese una segunda fuente para poder suplir la demanda en caso de problemas. Por lo tanto el Am286 es idéntico al Intel 80286. Posteriormente AMD lo vendió como procesador embebido⁴⁶.

Cyrix 486SLC

Este microprocesador tiene un bus de datos de 16 bits y 24 bits en el bus de direcciones, presenta una cache L1 de 1 K, frecuencia de reloj de 20MHz, 25MHz, 33MHz y 40 MHz, no posee coprocesador matemático, utilizado en placas base de bajo costo, clones PC y portátiles, por su bajo costo y bajo consumo de energía. El 486SLC se basa en el bus del i386SX, pensado para ser compatible y competir

⁴⁵ Extraído el 10 de Julio de 2009 desde <http://www.geocities.com/nachoenvweb/intel.html>

⁴⁶ Extraído el 10 de Julio de 2009 desde http://www.diclib.com/cgi-bin/d1.cgi?i=es&base=es_wiki_10&page=showid&id=48428

con los micros 386SX y 486SX. Se lanzaron mejoras de este microprocesador como el Cx486SLC/e con administración de energía, el Cx486SLC /eV de bajo consumo de voltaje (3.3v).

W65C816

Es un microprocesador fabricado por WDC (Western Design Center Inc.), consta de un bus de direcciones de 24 bits y un bus de datos de 16, tiene registros de 16 bits en el registro acumulador “A”, dos (2) registros índices, registro de página directa y registro puntero de PILA de 24 bits. Una característica especial es su capacidad de funcionamiento en dos modos, el “Nativo” y el modo “Emulado”, este último modo accedido tras el RESET, le permite tener compatibilidad con el MOS 6502, por lo que se considera al W65C816 también conocido como 65C816 directo sucesor del MOS 6502. Western Design Center Inc licenció el uso del MOS 6502 a varios fabricantes. El MOS 6502 fue utilizado como procesador de varios equipos de cómputo como el Apple II y en consolas de juego de Super Nintendo.

Motorola 68000

Recibe su nombre por el número aproximado de transistores en tecnología HMOS utilizado para su fabricación, introducido al mercado entre los años 1979 y 1980, tiene una arquitectura CISC, físicamente es un DIP de 64 pines, implementa un contador de programa de 32 bits, registro de estado de 16 bits, posee dos (2) bancos de registros de 32 bits, uno de datos y otro de punteros, los registros del banco de datos se pueden utilizar como registros de 32 bits (.l), 16 bits (.w) y 8 bits (.b), pudiéndose utilizar indistintamente como acumulador, puntero o índice, considerado el primero micro de la familia exitosa del “m68K” de 32 bits, en su primera versión se considera un micro de 16 bits por el límite del ancho de su bus externo de datos, aunque sus registros son de 32 bits. Al igual que en otros micros el Motorola 68000 fue utilizado en varios modelos de sistema de cómputo como el Apple (Lisa, en los primeros Macintosh), NeXT, Commodore (Amiga), Sun Microsystems, Sharp (X68000), Sinclair (Sinclair QL), en consolas de video juegos como SNK (Neo Geo), Sega (Mega Drive, Mega CD, Saturn), Atari (Atari ST) y fue base para el desarrollo de otros micros de Motorola y núcleos de algunas PDAs (Personal Digital Assistant – Organizador Personal) de Palm.

Zilog Z8000

El micro Z8000 fabricado por Zilog, sale al mercado en 1979 a la par con el i8086 y el Motorola 68000, con arquitectura de 16 bits, un conjunto de 16 registros de 16 bits, con capacidad de utilizarse como registros para instrucciones de 8, 16, 32 y 64 bits, dentro del juego de 16 registros se destaca el registro 15 como registro

puntero de PILA y el registro 14 como registro de segmento de PILA, no tenía compatibilidad con el Z80, aunque presentaba características especiales en su arquitectura como la circuitería integrada para el refresco de la DRAM y su utilización en algunos equipos de escritorio bajo Unix a principio de 1980, su permanencia en el mercado fue opacada por los Intel i8086 (con su arquitectura x86) y Motorola 6800, además de su limitada velocidad y algunos fallos, pero se tienen usos notables, en los sistemas de cómputo Zilog (Z8000 - System 8000) con capacidad de compartir recursos (puertos RS232 serie y paralelo) y poder implementar el sistema multiusuario (plataforma Unix), en sistemas de cómputo Olivetti (M20, M30, M40, M50, M60), Commodore 900, en consolas de juego como Namco (Pole Position) y principalmente en usos recientes en dispositivos militares de comunicaciones en controladores de comunicaciones (serie Z16C01/02).

Lección 9: Micróprocesadores de 32 bits

Estos microprocesadores manejan fundamentalmente un bus de datos de 32 bits, aunque la arquitectura de 32 bits se refiere a buses de datos, direcciones y registros de 32 bits, algunos microprocesadores presentan variación en el tamaño de su bus de direcciones, además de registros de tamaño variable. En esta gama o clasificación de microprocesadores se encuentra gran variedad no solo de fabricantes sino también de familias y dispositivos dentro del mismo fabricante, evidenciando el creciente desarrollo y evolución.

Intel 80386

(i386, 386) Es un microprocesador CISC con arquitectura x86. Durante su diseño se le llamó 'P3', debido a que era el prototipo de la tercera generación x86. El i386 fue empleado como la unidad central de proceso de muchos ordenadores personales desde mediados de los años 80 hasta principios de los 90. La relevancia de los microcontroladores 80386 además de sus 32 bits, fue el prototipo que marcó el inicio de la tercera generación de microprocesadores, se destaca la incorporación de una unidad de translación de páginas, haciendo posible la implementación de Sistemas Operativos que emplean memoria virtual.

Intel 80486

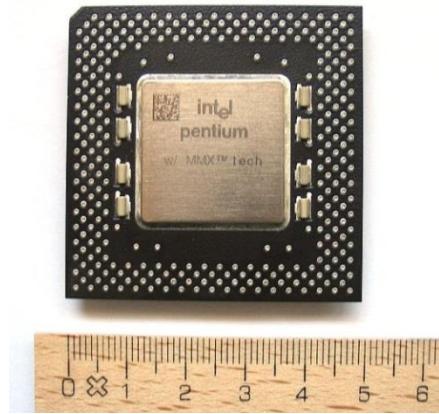
(i486, 486) son una familia de microprocesadores de 32 bits con arquitectura x86 diseñados por Intel. Las principales diferencias con el i386 son las siguientes: tienen un conjunto de instrucciones optimizado, una unidad de coma flotante y un caché unificado e integrado en el propio circuito del microprocesador y una unidad de interfaz de bus mejorada.

Intel Pentium

Este procesador se lanzó al mercado el 22 de marzo de 1993, sucediendo al procesador Intel 80486. Intel no lo llamó 586 sino Pentium, para poderlo registrar y así evitar que la competencia siguiera utilizando los mismos números que Intel para sus procesadores equivalentes (AMD 486, IBM 486, etc). También es conocido por su nombre clave P54C. El procesador Intel Pentium está formado por 3,1 millones de transistores y direcciona memoria con 64 bits. Integra dos memorias caché de 8 Kbyte (una para datos y otra para código) y tiene dos unidades aritmético lógicas (ALU), lo que le permite hacer tratamiento paralelo. Por tanto el Pentium puede ejecutar hasta dos instrucciones por ciclo de reloj. Está optimizado para ejecutar código de 16 bits.

Evolución del Pentium. En 1997, Intel presentó una evolución de su procesador Pentium, llamado Pentium MMX. Este se basaba en el mismo núcleo del Pentium original, pero se le añadió una memoria caché L1 de 32 Kbyte (frente a los 16 Kbyte del Pentium común), siendo 16 KB para datos y 16 KB para instrucciones, ordenadas en 4 vías de 4 KB cada una, y 57 nuevas instrucciones multimedia (de coma flotante), llamadas MMX, con el fin de ejecutar más rápidamente las futuras aplicaciones interactivas. Este procesador funcionaba entre 166 y 233 MHz. El nombre Pentium fue conservado por Intel para las generaciones siguientes de sus procesadores (Pentium Pro, Pentium II, Pentium III, Pentium IV y actualmente Pentium D), aunque existe una evolución importante en las arquitecturas.⁴⁷

Figura 48. Intel Pentium⁴⁸



⁴⁷ Extraído el 10 de Julio de 2009 desde <http://www.netzweb.net/html/print/pc/cpu/pentium.pdf>

⁴⁸ Extraído el 10 de Julio de 2009 desde <http://upload.wikimedia.org/wikipedia/commons/thumb/2/22/Pentium-mmx.jpg/606px-Pentium-mmx.jpg>

Pentium II

Este microprocesador fue al mercado el 7 de mayo de 1997. Está basado en una versión modificada del núcleo P6, usado por primera vez en el Intel Pentium Pro. Los cambios fundamentales fueron mejorar el rendimiento en la ejecución de código de 16 bits, añadir el conjunto de instrucciones MMX y eliminar la memoria caché de segundo nivel del núcleo del procesador, colocándola en una tarjeta de circuito impreso junto a éste. El Pentium II se comercializó en versiones que funcionaban a una frecuencia de reloj de entre 166 y 450 MHz. La velocidad de bus era originalmente de 66 MHz, pero en las versiones a partir de los 333 MHz se aumentó a 100 MHz.

Poseía 32 KB de memoria caché de primer nivel repartida en 16 KB para datos y otros 16 KB para instrucciones. La caché de segundo nivel era de 512 KB. Como novedad respecto al resto de procesadores de la época, el Pentium II se presentaba en un encapsulado SEC, con forma de cartucho. El cambio de formato de encapsulado se hizo para mejorar la disipación de calor. Este cartucho se conecta a las placas base de los equipos mediante una ranura Slot 1. Este microprocesador integra 7,5 millones de transistores.

Figura 49. Intel Pentium II⁴⁹



Celeron

Es el nombre que lleva la línea de procesadores de bajo costo de Intel. El primer Celeron fue lanzado en agosto de 1998, y estaba basado en el Pentium II. Posteriormente, salieron nuevos modelos basados en las tecnologías Pentium III y Pentium IV. Los procesadores Celeron se dividen en dos grandes clases: P6: Basada en los procesadores Pentium II y Pentium III y Netburst: Basada en los procesadores Pentium IV

⁴⁹ Extraído el 10 de Julio de 2009 desde <http://www.cpu-world.com/CPUs/Pentium-II/index.html>

Pentium III

Es un microprocesador de arquitectura i686. Fue lanzado el 26 de febrero de 1999. Las primeras versiones eran muy similares al Pentium II, siendo la diferencia más importante la introducción de las instrucciones SSE. Al igual que con el Pentium II, existía una versión Celeron de bajo costo y una versión Xeon para quienes necesitaban gran poder de cómputo. Esta línea ha sido eventualmente reemplazada por el Pentium IV, aunque la línea Pentium M está basada en el Pentium III.

Figura 50. Intel Pentium III⁵⁰



Pentium IV

Microprocesador de séptima generación basado en la arquitectura x86 y manufacturado por Intel. Es el primer microprocesador con un diseño completamente nuevo desde el Pentium Pro de 1995. El Pentium IV original, denominado *Willamette*, trabajaba a 1,4 y 1,5 GHz; y fue lanzado en noviembre de 2000.

Para la sorpresa de la industria informática, el Pentium IV no mejoró el diseño P6 según las dos tradicionales formas para medir el rendimiento: velocidad en el proceso de enteros u operaciones de coma flotante. La estrategia de Intel fue sacrificar el rendimiento de cada ciclo para obtener a cambio mayor cantidad de ciclos por segundo y una mejora en las instrucciones SSE. Al igual que los demás procesadores de Intel, el Pentium IV se comercializa en una versión para equipos de bajo presupuesto (Celeron) y una orientada a servidores de gama alta (Xeon).

Pentium IV EDICIÓN EXTREMA: en septiembre de 2003, Intel anunció la edición extrema (*Extreme Edition*) del Pentium IV. Se destacó en el área de la codificación multimedia, ya que superaba la velocidad de los anteriores Pentium 4 y a toda la línea de AMD.

⁵⁰ Extraído el 10 de Julio de 2009 desde <http://www.cpu-world.com/CPUs/Pentium-III/index.html>

Pentium M

Fue Introducido en marzo de 2003. Este microprocesador cuenta con una arquitectura x86 (i686). Originalmente fue diseñado para uso en ordenadores portátiles. Su nombre en clave antes de su introducción era "Banias". Todos los nombres clave del Pentium M son lugares de Israel, la ubicación del equipo de diseño del Pentium M.

El Pentium M es una versión fuertemente modificada del diseño del Pentium III. Está optimizado para un consumo de potencia eficiente, una característica vital para ampliar la duración de la batería de los ordenadores portátiles. Funciona con un consumo medio muy bajo y desprende mucho menos calor que los procesadores de ordenadores de sobremesa. El Pentium M funciona a una frecuencia de reloj más baja que los procesadores Pentium IV normales, pero con un rendimiento similar, y puede igualar o superar el rendimiento de un Pentium IV Prescott a 3 GHz. Este procesador forma parte de la plataforma Intel Centrino.

AMD-Am386

Fue creado por AMD en 1991. Era un procesador con características semejantes al Intel 80386 y compatible 100% con este último, lo que le valió varios recursos legales de Intel por copiar su tecnología. Tenía una velocidad de hasta 40 MHz lo que superaba a su competidor que sólo llegó a los 33 MHz.

AMD-Am486

Fue presentado en 1993 por Advanced Micro Devices (AMD). Es un procesador compatible x86, comparable al Intel 80486.

AMD-5x86

Es un procesador compatible x86 presentado en 1995 por Advanced Micro Devices destinado a ser utilizado en ordenadores basados en un 486. Presentado en noviembre de 1995, el AMD 5x86 es un procesador 486 estándar con un multiplicador interno a 4x, permitiéndole funcionar a 133 MHz en sistemas para procesadores 486 DX2 o DX4 sin multiplicador. Tenía una memoria caché L1 de 16 Kbyte. Esta combinación permitió al 5x86 igualar e incluso superar ligeramente un procesador Pentium a 75 MHz. Además, como fue concebido en base a un 486, era compatible con sistemas más antiguos, lo que perjudicaba ligeramente a su rival más rápido, el Cyrix Cx5x86⁵¹.

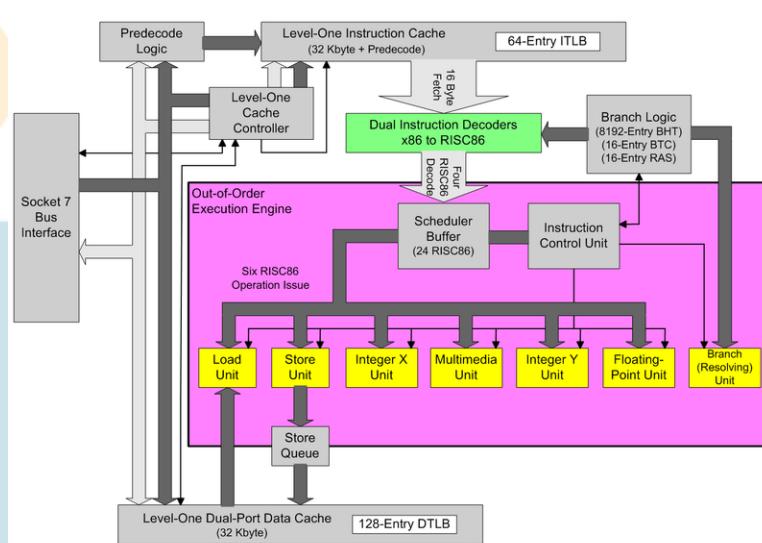
AMD K6

El AMD K6 sale al Mercado en 1997 como rival del Intel Pentium II, con frecuencias entre 166 MHz hasta 300 MHz, con el juego de instrucciones MMX,

⁵¹ Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/AMD_Am5x86

que ya se ha convertido en un estándar y con un precio competitivo en comparación con los Intel Pentium II e incluso Pentium MMX, compatible con placas base “socket 7”, el rendimiento del AMD K6 era superior ampliamente al Pentium MMX pero relegado en rendimiento por el Pentium Pro y en operaciones de coma flotante (gráficos y juegos 3D) por el Pentium II. Su sucesor fue el microprocesador K6-2⁵².

Figura 51. AMD K6⁵³



AMD K6-2

Microprocesador compatible x86, evolución del AMD K6, con una frecuencias entre 233 MHz hasta 570 MHz, presenta bus de datos y direcciones de 32 bits, compatible con “socket 7” (Socket super 7), implementa un cache nivel 1 de hasta 64 KB e instrucciones de coma flotante SIMD (Single Instruction MultipleData – Una instrucción múltiples datos) denominado por AMD como “3DNow!” (instrucciones multimedia añadidas al conjunto original del x86), como mejora de AMD al conjunto de instrucciones MMX (), “3DNow!” salió con anterioridad al conjunto de instrucciones de Intel SSE (Streaming SIMD Extensions), conjunto de instrucciones extensión de las instrucciones MMX (Conjunto de instrucciones SIMD diseñadas por Intel) de Intel para los Pentium MMX (MultiMedia eXtension – Multiple Math eXtension – Matrix Math eXtension). El AMD K6-2 “3DNow!” impulsó la carrera de AMD por sus características sobresalientes en bajo precio y rendimiento con respecto a sus competidores Pentium Pro y Pentium II, posicionándolo en el mercado y financieramente para permitir la difusión de otros procesadores como el AMD Athlon.

⁵² Extraído el 10 de Julio de 2009 desde <http://www.slideshare.net/dElyya/soquet-y-microprocesador-presentation>

⁵³ Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Archivo:Amdk6_arch.png

AMD K6-III

Este microprocesador evolución del AMD K6-2 “3DNow!”, lanzado al mercado en 1999, compatible x86, para “socket 7” (Socket super 7), con bus de datos e instrucciones de 32 bits, implementa una cache L1 de 32 K, una cache L2 de 256 K, último micro para plataforma Socket 7, con frecuencias entre 400 MHz hasta 450 MHz, directo competidor del Pentium III, logrando rendimiento similar con placas base que implementan cache L3 de 1M.

AMD Athlon

Internamente el Athlon es un rediseño de su antecesor, al que se le mejoró sustancialmente el sistema de coma flotante, esto es, 3 unidades de punto flotante que pueden trabajar simultáneamente y se le aumentó la memoria caché de primer nivel (L1) a 128 KB (64 KB para datos y 64 KB para instrucciones). Además incluye 512 KB de caché de segundo nivel (L2) externa al circuito integrado del procesador y funcionando, por lo general, a la mitad de velocidad del mismo. Lanzado al mercado en 1999 para “socket A” y “socket 563”, con frecuencias de 500 MHz hasta 2,4 GHz, primer procesador de séptima generación compatible x86, primer procesador en romper la barrera del Giga Hertz (Athlon Classic – núcleo K7), funcionando a frecuencia de 100 Mhz DDR (Dual Data Rate – 200MHz efectivos DDR), esta gama de micros utilizó varios núcleos en su evolución iniciando con el “K7” (Argon), “K75” (Pluto/Orion), Thunderbird, Palomino, Thoroughbred A/B, Barton y Thorton. Tiene una mejora en su sistema de coma flotante, cache L2 de 512K, directo competidor de Intel por su rendimiento y bajo precio. Los Athlon con núcleo Thunderbird (Athlon Thunderbird) implementa cache L1 128K (64K para datos y 64K para instrucciones), cache L2 de 256 K, funcionando a frecuencia de 133MHz DDR (266 MHz efectivos DDR), superando al Pentium III y primeros Pentium IV. El Athlon XP (Palomino) directo competidor con el Pentium IV de 1,7 GHz, con velocidades de reloj entre 1,3 GHz hasta 1,7 GHz, primero en incluir el conjunto de instrucciones SSE (Streaming SIMD Extensions) sumadas con las instrucciones 3DNow!, su rendimiento es 10% más alto que los Athlon Thunderbird, debido a las mejoras se comercializan por un “índice de prestaciones relativas” (PR) y no por la frecuencia de reloj, forma en el que se indica la equivalencia de velocidad de reloj equivalente a un Intel.

Los modelos mejorados con núcleos Thoroughbred, Barton y Thorton implementaron mejoras en rendimiento y de bajo consumo, su comercialización utiliza el índice PR (Prestaciones Relativas) identificándose con una velocidad relativa o equivalente a los Intel y un signo “+” por ejemplo “Athlon XP 2600 +”. El sucesor del Athlon y Athlon XP es el Athlon 64.

AMD Duron

Al igual que en Intel con sus procesadores de bajo costo Celeron, los AMD Duron (2003) son la gama de procesadores de bajo costo compatibles con los Athlon y la

arquitectura x86, directos competidores de Celeron de Intel, tienen una cache L1 de 64K y una cache L2 de 64 K (diferencia principal frente a los 256K de Athlon), para su fabricación se utilizan los núcleos Spitfire, Morgan y Applebred (basado en el núcleo Thoroughbred – Athlon XP), funciona a una frecuencia de 1,4 MHz hasta 1,8 MHz, con bus de 133MHz DDR (266 MHz efectivos por la tecnología DDR). Sucesor directo del AMD Sempron.

AMD Sempron

Microprocesador de bajo costo, sale al mercado en 2004, directo competidor del Intel Celeron, con cache L1 128K y cache L2 de 256K, bus de 333MHz, basados en sus primeras versiones en el núcleo Thoroughbred/Thorton del Athlon XP, con índice PR de 2400+ y 2800+, posteriormente en el núcleo Barton de Athlon XP, con índice PR 3000+ con cache L2 de 512K, para “socket A”, finalmente aparecen versiones con núcleo Paris del Athlon 64, que no implementan instrucciones AMD64, pero si el controlador de memoria y la cache L2 256K para “socket 754” y últimas versiones con base en el núcleo Palermo del Athlon 64 implementando las instrucciones AMD64, con soporte parcial SSE3, con cache L2 de 128K o 256K e índice PR 3800+ para “socket AM2”.

Cyrix 486DLC

Lanzado al mercado en 1992 (actualmente obsoleto), utilizaba un zócalo PGA de 132 pines en placa AT, con velocidad de reloj de 25, 33 y 40Mhz. Implementaba una arquitectura RISC, logrando ejecutar tres operaciones de 16 bits en cada ciclo de reloj, con un bus de datos y direcciones de 32 bits, con voltaje de nucleo y unidad I/O de 5 voltios, con 1K de memoria cache de nivel 1, este microprocesador se presenta como la competencia del 386DX. Al no tener un coprocesador matemático interno presentaba rendimiento bajo en aplicaciones de programas CAD y 3D, el diseño basado en tecnología de 16 bits le impedía procesar programas de la plataforma de Windows NT (32 bits).

Cyrix 586

Compatible con sólido PGA de 168 pines, compatible con el i486, voltaje de operación de 2,45 Voltios, frecuencias de 100, 120 MHz, con un rendimiento equivalente al Pentium a 75 MHz y superior al AMD 5x86.

Cyrix 6x86

Compatible con socket 7, con una cache para datos e instrucciones de 16K, sigue presentando problemas con el coprocesador matemático, por la simplificación de la unidad de coma flotante, las primeras versiones de este procesador

presentaban problemas de calentamiento excesivo en los reguladores de voltaje de la tarjeta madre por el alto consumo, con Windows NT, Microsoft recurre a deshabilitar la cache interna del microprocesador para reducir problemas y mejorar la estabilidad, pero reduciendo en un 25% el rendimiento

Cyrix 6x86MX o M1

En 1996 sale al mercado el 6x86MX o también llamado M1, es el primer microprocesador de Cyrix en implementar instrucciones MMX, fabricado por IBM, compatible con socket 7, con una cache interna de hasta 64KB, con frecuencias de 133, 166 y 188MHz, ofreciendo rendimientos similares a los Intel Pentium de 166, 200 y 133 MHz. Como desventaja prevalece el bajo rendimiento del coprocesador matemático, además por su innovación había problemas con las velocidades autorizadas en la tarjeta madre, que pese a que aumentaban las prestaciones de la maquina ocasionaba problemas en las memorias o PCI al forzarlos a trabajar a una velocidad (frecuencia) mayor a la que originalmente fueron fabricados, como solución a este problema se incorpora multiplicadores x2, x2.5, x3 y x3.5 con lo que se puede ajustar la velocidad (frecuencia) de trabajo, pero se pierda rendimiento.

MII

Compatible con socket 7 y super 7, con cache unificada (para datos e instrucciones) de 64KB, presenta un diseño similar al 6x86MX, con multiplicadores y doble voltaje de trabajo. Presentado como competidor del Pentium II, nuevamente se tiene problema con la Unidad de Punto Flotante – FPU, agudizando el problema con la ejecución de los juegos y gráficos 3D, que para la época se comercializan, relegando este microprocesador a aplicaciones ofimáticas. Como ventaja se tiene la compatibilidad con soportes para MMX, permitiendo actualizar equipos viejos sin cambiar placas base.

Cyrix MediaGX

Lanzado al mercado en 1996, fabricado inicialmente por Cyrix y después por National Semiconductor, este microprocesador tenía como innovación la incorporación en el mismo chip del hardware de audio y video, con una CPU basada en un 5x86, socket7, se caracterizó por su bajo consumo y alta integración, utilizado en equipos portátiles de gama baja como los Presario 2100 y 2200 de Compaq, abriendo mercado en Packard Bell. Se fabricaron versiones compatibles MMX y con velocidades entre 180 y 300 MHz.

Cyrix III

Es un microprocesador compatible x86 en socket 370, lanzado en 2000 por VIA Technologies propietaria de Cyrix, este procesador según el fabricante equivale a un Celeron 433MHz, incorpora instrucciones MMX y 3D Now! (AMD), con bajo consumo y voltaje, reduciendo el calentamiento, sigue con un bajo rendimiento en el FPU pero a un buen precio.

PowerPC 600 o PowerPC G2

La denominación G2 es un nombre que la alianza AIM confiere a sus microprocesadores PowerPC de segunda generación, considerada la primera familia construida de procesadores PowerPC, en 1992 sale al mercado el PowerPC 601 como el primer procesador de segunda generación (G2) de 32 bits, con velocidad entre 50 y 80MHz, 32 KB de cache, utilizado en los Power Macintosh, le sigue el PowerPC 601 v1 de 90 a 120MHz, en 1994 el PowerPC 603 de bajo consumo de energía con velocidad de hasta 100MHz, le siguen el PowerPC 603e con hasta 200 MHz y el PowerPC 603ev con hasta 300 MHz imponiendo una marca en computadores de escritorio en su tiempo, en 1994 sale al mercado el PowerPC 604 con velocidades entre 100 y 180 MHz, con 32 bits, con versiones como el PowerPC 604e con velocidades entre 166 y 233 MHz. El PowerPC 620 lanzado al mercado en 1997 es el primero que implementa en su totalidad una arquitectura de 64 bits con velocidades entre 120, 150 y hasta 200 MHz.

PowerPC 700 o PowerPC G3

En 1997 se presenta como sucesor del PowerPC 603e, en modelos como el PowerPC 740 y PowerPC 750, con velocidades de hasta 366 MHz y bajo consumo de energía, el PowerPC 740 llega a superar en rendimiento al Pentium II. Las revisiones posteriores y mejoras de los PowerPC 740 y PowerPC 750 logran velocidades mayores en frecuencia, como en 2001 con el PowerPC 750Cxe, en 2002 con el PowerPC 750FX con hasta 900 MHz y 2004 con el PowerPC 750GX con hasta 1,1 GHz. Los microprocesadores de esta gama se utilizan en computadores como las iMac, PowerBook G3 e iBooks.

PowerPC 7400 o PowerPC G4

Aparece en el mercado en 1999 con velocidades entre 350 y 500 MHz, en 2001 se desarrolla el PowerPC 7410 como una versión de bajo consumo, se desarrolla el PowerPC 7450 con cache L2 de 256KB y L3 de hasta 2MB y velocidad de hasta 733MHz, en 2002 el PowerPC 7445 y 7455 superando la barrera de velocidad de 1GHz, utilizado en el AmigaOne XE, en Junio de 2005 el PowerPC 7448 (Apolo 8)

con bajo consumo de energía, con cache L2 de hasta 1MB y velocidad de hasta 2GHz. Esta gama de los PowerPC 7400 implementa el PowerBook G4, el iMac G4, el Power Macintosh G4, el Xserve, eMac, Mac mini e iBook.

Motorola 68020

Lanzado en 1984, presenta una arquitectura de 32 bits (bus de datos e instrucciones a 32 bits), con frecuencias de 12 MHz hasta 33MHz, implementa una pipeline de tres niveles, modelo multiproceso, incrementando las instrucciones para operaciones aritméticas y de campo de bits, utilizado en sistemas de cómputo Apple (Macintosh II y Macintosh LC), Sun-3, Hewlett Packard 8711, Commodore Amiga 120 y video consolas AmigaCD32 y en sistemas de cómputo militar en el avión caza Eurofighter Typhoon y lanzador Ariane 5.

Z-80000

Fabricado por Zilog, el Z-80000 es la evolución del Z-8000 (16 bits), es un procesador de 32 bits, lanzado al mercado en 1986, incorpora una cache de 256K, con capacidad multiproceso, pipeline de seis niveles, 16 registros de propósito general de tamaño variable, incluye una unidad MMU (Memory Management Unit - Unidad de gestión de memoria), compatible en código escrito para el Z8000, aunque incompatible en arquitectura con el Intel x86 y el Z80. Puede acceder a 6 GBytes de memoria, implementa tres métodos de acceso a memoria: en modo compacto (hasta 64K - pequeños programas), modo lineal (acceso directo a los 4GBytes) y modo segmentado (32.768 segmentos de 64K o 128 segmentos de 16MB sobre un total de 2GB de memoria accesible). El Z-80000 en su momento se considera como un "mainframe" (computador central) en un chip.

Sun Microsystems – Sun SPARC

Sun Microsystems desde 2009 propiedad de Oracle, en 1985 lanza al mercado el procesador Sun SPARC (Scalable Processor ARChitecture), basado en arquitectura RISC (en diseños RISC I y II), implementa 32 registros de 32 bits para enteros, 16 registros de 64 bits para coma flotante, la CPU está compuesta por una unidad entera "UI" (Integer Unit) para la ejecución básica y una "FPU" (Floating Point Unit) para cálculos sobre números reales, incorpora una MMU (Memory Management Unit - Unidad de gestión de memoria), para el SPARC se tiene frecuencias de 16MHz hasta 50MHz, en el Super SPARC se tiene frecuencias de 33MHz hasta 50MHz. Los micros Sun SPARC se consideran la primera arquitectura RISC abierta licenciada a fabricantes como Texas Instruments, Fujitsu, entre otros.

ARM

Los ARM son microprocesadores con arquitectura de 32 bits ampliamente utilizados en dispositivos móviles (PDA, tablets, teléfonos inteligentes, videoconsolas, calculadoras, reproductores de música, etc) y periféricos, como discos duros, routers, etc, por su bajo consumo de potencia y bajo costo. ARM se caracteriza por ser una arquitectura licenciable, permitiendo desarrollos continuos por parte de ARM y de otras titulares de licencia (Nvidia, NEC, Microsoft, Freescale, Intel, Atmel, Apple, Alcatel-Lucent, Nintendo, Broadcom, Cirrus Logis, LG, Marvell Technology Group, Nokia, Sony, Qualcomm, Sharp, Texas, Samsung, Yamaha, STMicroelectronic, etc).

El inicio de los procesadores ARM se remonta a la compañía Acorn Computers que lidera un equipo de desarrolladores de un nuevo procesador basado en la arquitectura del MOS 6502RISC, generando el ARM1 en 1985 como prototipo y comercialmente el ARM2 lanzado al mercado en 1986, construido con 30.000 transistores, sin memoria cache, con un bus de datos de 32 bits y direcciones de 26 bits, 16 registros de 32 bits dentro de los que existe uno utilizado como contador de programa y como contenedor de los "flags" o banderas de estado del procesador (4 bits superiores y 2 inferiores), con un rendimiento notable comparado con un Z80. El ARM3 incorpora una cache de 4KB, los primeros ARM fueron utilizados en los computadores personales Acorn Archimedes.

En 1980 Acorn en asociación con Apple trabajan en nuevas versiones de ARM, se crea la compañía Advanced RISC Machines, en 1991 se presenta el ARM6 utilizado en el PDA Apple Newton (en su versión ARM 610) y por Acorn en 1994 en el RiscPC, el ARM7TDMI con 15 MIPS (16,8 MHz en frecuencia) hasta 60 MIPS (59,8 MHz en frecuencia) obtuvo gran relevancia en utilización de la arquitectura ARM para dispositivos móviles y consolas de video portátiles. Siguieron posteriores desarrollos con el ARM8, ARM9TDMI, ARM9E, ARM10E, XScale y ARM11 con características sobresalientes cache, instrucciones, MIPS (sobre los 740 MIP) y frecuencias hasta 528 MHz. Hasta las versiones más recientes conocida como la familia "Cortex" (versión de arquitectura ARMv6-M, ARMv7-A/M/ME/R), núcleo Cortex-A5/A8/A9/A9/A12/A15 MPCore, Cortex-M0/M1/M3/M4, Cortex-R4 y Qualcomm Scorpion, particularmente el Cortex-A15 presenta ejecución de más de 10.000 DMIPS (Dhrystone Million Instructions Per Second), Dhrystone hace referencia a un benchmark (Técnica para medir rendimiento en un sistema o componente), con frecuencias sobre los 2,5 GHz, multi-núcleo, mejora en el pipeline de 15 niveles, cache de 4MB, direccionamiento de hasta 1TB de memoria física, extensiones NEON SIMD (128 bit SIMD – Single Instruction Multiple Data) como mejora y extensión a la técnica de instrucciones

SIMD para conseguir paralelismo a nivel de datos, instrucciones que aplican una misma operación sobre un conjunto de datos.

NVidia - Tegra

Los microprocesadores Tegra, desarrollados por NVidia, es un micro SoC (System on a Chip – Sistema en un Chip), por integrar en el mismo chip una CPU ARM, una GPU (Graphics Processing Unit), controlador de memoria, NorthBridge (MCH – Concentrador Controlador de Memoria y GMCH si incluye el controlador gráfico, controla el funcionamiento del bus del microprocesador, para el acceso a y desde el microprocesador, memoria, AGP, PCI-express) y SouthBridge (coordina los dispositivos de entrada/salida, soporte para buses ISA, SPI, controlador para SATA y PATA, puente PLC, reloj en tiempo real, BIOS, Interfaz de sonido, administración de potencia APM y ACPI, no se conecta a la CPU directamente se comunica vía el NorthBridge).

En 2008 sale al mercado el Tegra APX 2500 con ARM11 (600MHz) con soporte DDR, soporte GeForce, en 2009 con el Tegra APX2600, en 2010 sigue el Tegra 650 (Tegra 6xx), en 2011 y años posteriores siguen los Tegra 2 y Tegra 3 a partir del cual los microprocesadores NVidia – Tegra adoptan nombres de superhéroes como Los NVidia-Tegra 3 (Kal-El), Tegra 4 (Wayne), Tegra (Grey), Tegra (Logan), Tegra (Stark). Todos los micros NVidia-Tegra se caracterizan por el bajo consumo de energía y alto rendimiento para aplicaciones de audio y video.

Lección 10: Microprocesadores de 64 bits, microprocesadores en supercomputadoras y móviles

Los microprocesadores de 64 bits constituyen un modelo, una arquitectura y el tamaño de los datos o direcciones de memoria, que ahora se compone de 8 Bytes, la tecnología de 64 bits no es nueva en el sentido de su aplicación práctica, esta tecnología está presente desde 1960 en supercomputadoras, también en servidores y estaciones de trabajo basadas en arquitectura RISC implementados en los 90's.

La introducción de esta tecnología en computadores personales, se hace desde 2003 con arquitecturas x86-64 y procesadores PowerPC G5. El término 64 bits puede referirse a la CPU, al tamaño de los buses o al tamaño de las instrucciones,

pero realmente se habla de tecnología de 64 aplicada a un equipo de cómputo si el equipo cumple con estos tres parámetros:

- La CPU internamente debe ser de 64 bits.
- Los buses deben tener un ancho de 64 bits.
- Las instrucciones llegan a tamaños de 64 bits.

Ventajas y desventajas de la arquitectura de 64 bits

Ventajas, La más notoria es el aumento de la capacidad de memoria tanto RAM como virtual con respecto a la tecnología de 32 bits ya que estos últimos tienen una limitante de 4 GB máximo de memoria. La cantidad máxima de memoria soportada por las versiones de 64 bits de Windows es:

- Windows XP professional 64 bits: 16 GB de RAM.
- Windows Vista Home Basic 64 bits: 8GB de RAM.
- Windows Vista Home Premium 64 bits: 16 GB de RAM.
- Windows Vista (otras versiones) 64 bits: 128 GB de RAM.
- Windows 7 Home basic/Premium 64 bits: 8 GB/16GB de RAM.
- Windows 7 Ultimate / Enterprise 64 bits: 192 GB de RAM en ambas versiones.
- Windows 8 64 bits: 128 GB de RAM
- Windows 8 Professional o Enterprise 64 bits: 512 GB de RAM.

Otra ventaja de los sistemas de 64 es su mayor capacidad de procesamiento, estabilidad y seguridad de la información.

Desventajas, Los sistemas operativos de 64 bits, son de uso más profesional que doméstico, tiene inconvenientes con compatibilidad de software:

- No son compatibles con programas o software de 16 bits
- Inclusive algunos antivirus y aplicaciones de 32 bits presentan incompatibilidad.
- Existen problemas de Drivers para tecnología 64 bits.

Arquitecturas 64 bits

Como se ha expuesto la tecnología de 64 bits se ha extendido desde los años 60's hasta la actualidad, es muy amplia y compleja las arquitecturas que incorporan tecnología de 64 bits por lo que el estudio requiere una profunda conceptualización en los términos tanto de hardware como de software

involucrados, igualmente en la microelectrónica empleada por lo que se invita a los estudiantes a profundizar en esta tecnología comenzando con la literatura que se encuentra en línea y en los fabricantes de estos productos, la siguiente es una condensación de las principales arquitecturas.

Arquitectura DEC Alpha: *la organización de sus registros es de uso general con una arquitectura que se puede encuadrar como de registro-registro. Esto hace que la memoria de sus instrucciones opere sobre los registros, haciendo uso de la memoria RAM solo para instrucciones de carga y almacenamiento. La razón es que se intenta minimizar los accesos a memoria, puesto que suponen el cuello de botella para los procesadores actuales, la longitud de palabra de los registros es de 64 bits, ya sea desde el PC (Contador de Programa), pasando por los registros de enteros, punto flotante, etc... el primer procesador que hizo gala de la tecnología Alpha fue el 21064*

Arquitectura IA-64: *Línea de procesadores Itanium e Itanium2. Representan el diseño de producto más complejo del mundo con más de 1700 millones de transistores. Esto permite obtener sólidas capacidades de virtualización, mejorar la confiabilidad y niveles de rendimiento líderes del mercado.*

Arquitectura AMD64: *El conjunto de instrucciones del AMD 86x-64 (renombrada posteriormente como AMD64) es una extensión directa de la arquitectura del x86 a una arquitectura de 64 bits, motivado por el hecho de que los 4GB de memoria que son direccionables directamente por una CPU de 32 bits ya no es suficiente para todas las aplicaciones. El primer procesador con soporte para este conjunto de instrucciones fue el Opteron lanzado en 2003. Posteriormente ha sido implementado en múltiples variantes del Athlon 64 y el Pentium 4 de Intel, en este último caso bajo una versión de Intel llamada EM64T.*

Arquitectura SPARC: *Es la primera arquitectura RISC abierta y como tal las especificaciones de diseño son públicas, así otros fabricantes de microprocesadores pueden desarrollar su propio diseño. Una de las ideas innovadoras de esta arquitectura es la ventana de registros que permite hacer fácilmente compiladores de alto rendimiento y una significativa reducción de memoria en las instrucciones load/restore en relación con otras arquitecturas RISC. Las ventajas se aprecian sobre todo en programas grandes.*

La CPU SPARC está compuesta de una unidad entera, UI (Integer Unit) que procesa la ejecución básica y una FPU (Floating-Point Unit) que ejecuta las operaciones y cálculos reales. La UI y la FPU pueden o no estar integradas en el mismo Chip.

Aunque no es una parte formal de la arquitectura, las computadoras basadas en sistema SPARC de Sun Microsystem tienen una unidad de manejo de memoria (MMU) y una gran cache de direcciones virtuales (para las instrucciones y los

datos) que están dispuestos periféricamente sobre un bus de datos y direcciones de 32 bits.

Arquitectura POWER: Es usada en servidores IBM, pero sin embargo hay muchos microprocesadores que son derivados o variantes de este que se encuentra en gran variedad de equipos que van desde computadoras para automóviles hasta consolas de videojuegos, su nombre proviene de “Performance Optimization With Enhanced RISC”.

PowerPC970 o PowerPC G5: Este microprocesador es de 64 bits con compatibilidad nativa 32 bits, fabricado en 2002, con cache L1 de 64KB, L2 de 512KB y velocidad de hasta 1,8 GHz, en 2004 sale al mercado el PowerPC 970FX con velocidad de 2,0 GHz y en 2005 con un chip doble núcleo y un incremento en cache L2 con 1MB por núcleo sale al mercado el PowerPC 970MP. Este microprocesador se utiliza en los Power Mac G5, eServer BladeCenter y supercomputadoras como MareNostrum y Magerit.

Arquitectura PA-RISC: Una característica interesante de PA-RISC es que la mayoría de sus microprocesadores no tienen cache L2. En su lugar se implementa un cache L1 mayor, formada por chips separados conectados al microprocesador a través de un bus (actualmente está integrada en el propio chip). Solo el modelo PA-7300LC tiene cache L2. Otra innovación de esta arquitectura fue la adición de un repertorio de instrucción multimedia (SIMD) conocido como MAX e introducido por primera vez en el 7100LC.

También se suelen referir a ella como la arquitectura HP/PA, Hewlett Packard Precision Architecture. PA se desarrolla en Palo Alto, donde se encuentra la central de HP.

Intel Pentium D

Fue introducido por Intel en 2005. Un chip Pentium D consta básicamente de dos procesadores Pentium IV (de núcleo Prescott) en una única pieza de silicio con un proceso de fabricación de 90nm. El nombre en clave del Pentium D antes de su lanzamiento era "Smithfield". Incluye una tecnología DRM (Digital Rights Management) para hacer posible un sistema de protección anticopia de la mano de Microsoft.

Intel Xeon

Es una generación de microprocesadores Intel para servidores PC. El primer procesador Xeon apareció en 1998 como Pentium II Xeon. El Pentium II Xeon utilizaba tanto el chipset 440GX como el 450NX. En el año 2000, el Pentium II Xeon fue reemplazado por el Pentium III Xeon. El Xeon MP, lanzado en 2002, que combinaba la tecnología Hyper-Threading con NetBurst. Sus chipsets utilizan el

socket 603 y tiene versiones GC-LE (2 procesadores, 16Gb de memoria direccionable) y GC-HE (4 procesadores o más, 64Gb direccionables), todos usando un bus de 400MHz.

Como la familia x86/IA-32 estándar de Intel de procesadores PC de escritorio, la línea de procesadores Xeon hasta aquí es de 32 bits. Se plantea una versión de 64 bits de Xeon que complementará (o reemplazará) a la CPU Itanium de Intel. El 26 de junio de 2006, Intel anunció la nueva generación de Xeon con tecnología de doble núcleo. Intel afirma que este nuevo procesador brindará un 80% más de rendimiento por vatio y que será hasta un 60% más rápido que la competencia.

Los Procesadores Xeon han evolucionado en su familia de procesadores para servidores, de 64 bits, con multinúcleo, capacidad de varios subprocesos, velocidad sobre los 2 GHz, tamaños de cache entre 10 MB y 30 MB, TurboBoost (incremento de frecuencia de funcionamiento automático), Hyper Threading (simula dos procesadores lógicos dentro de un núcleo físico), algunos incorporan GPUs, con los procesadores Xeon familia E3 para aplicaciones en pequeñas empresas, Xeon 5000 para aplicaciones robustas y que requieren eficiencia energética, los Xeon E5 como una gama flexible y potente y los Xeon E7 como procesadores de alto rendimiento y escalabilidad con hasta 10 núcleos y 20 subprocesos, caché de 30 MB, soporte sobre los 32 GB DIMMs (sobre 64 ranuras DIMM o sistema de cuatro sócculos para más de 2TB de memoria), TurboBoost, Hper Threading

Intel Core Duo

Es el microprocesador de Intel que cuenta con dos núcleos de ejecución. Fue lanzado en enero del 2006. El microprocesador Intel Core Duo está optimizado para las aplicaciones de subprocesos múltiples y para la multitarea. Puede ejecutar varias aplicaciones simultáneamente, como juegos con gráficos o programas que requieran muchos cálculos, al mismo tiempo que puede descargar música o analizar su PC con su antivirus en el segundo plano. Este microprocesador implementa 2Mb de caché compartida para ambos núcleos más un bus frontal de 667Mhz; además implementa un nuevo juego de instrucciones para multimedia (SSE3) y mejoras para las SSD y SSD2; sin embargo, el desempeño con enteros es ligeramente inferior debido a su caché con mayor latencia. Intel Core Duo es el primer microprocesador de Intel usado en las computadoras Apple Macintosh.

El Core Duo contiene 151 millones de transistores. El núcleo de ejecución del procesador contiene un pipeline de 12 etapas con velocidades previstas de ejecución entre 2.33 y 2.50 GHz. La comunicación entre la caché L2 y los dos núcleos de ejecución es controlada por un módulo de bus árbitro que elimina el tráfico de coherencia a través del bus frontal (FSB), con el costo de elevar la latencia de la comunicación de núcleo a L2 de 10 ciclos de reloj. El incremento de la frecuencia de reloj contrapesa el impacto del incremento en la latencia.

Las nuevas características de administración de energía incluyen control mejorado de temperatura, así como escalado independiente de energía entre los 2 núcleos, lo que resulta en un manejo de energía mucho más eficiente que los diseños anteriores.

Intel Core 2

Son microprocesadores introducidos al mercado en 2006, Intel 64 bits, doble núcleo, conjunto de instrucciones x86-64, con velocidades de CPU menores pero con eficiencia y rendimiento notablemente mejorados en comparación con la familia Penentium, se destacan los micros Core 2 Duo (Implementados con núcleos Conroe, Allendale, Wolfdale,), Core 2 Quad (Implementados con núcleos Kentsfield XE, Yorkfield XE) y Core 2 Extreme (Implementados con núcleos Conroe XE, Kentsfield XE, Yorkfield XE).

Core 2 Quad: microprocesador de cuatro núcleos, basado en micro arquitectura de 45 nanómetros, presenta cache hasta de 12 MB, bus frontal de 1333 MHz.

Core 2 Extreme: microprocesador de cuatro núcleos, tecnología de 45 nanómetros, velocidad de 3,2 GHZ, 12 MB de cache total, bus frontal de 1600 MHz.

Intel Core i

Los microprocesadores Core i son la última familia de micros para laptops y escritorio, son evolución de los Core 2, incorporan varios núcleos, procesador gráfico integrado (GPU), controlador de memoria DDR, Turbo Boots e Hyper-Threading. Construido con los núcleos Arrandale y el último en Clarkdale.

Core i3: Incorpora 2 núcleos, con la tecnología Hyper-Threading puede ejecutar 4 subprocessos simultáneamente, integra GPU, con capacidad de ejecutar aplicaciones de 32 y 64 bits, caché sobre los 4MB y soporte para DDR3.

Core i5: Los micros Core i5 con núcleo Arrandale y Clarkdale incorporan 2 núcleos, con cache de 3MB y 4MB respectivamente, el Core i5 con núcleo Lynnfield incorpora 4 núcleos, con cache 8 MB. Los Core i5 incorporan controlador de memoria DDR3, con modelos con GPU integrada.

Core i7: Utilizan micro arquitectura o núcleos Nehalem, Sandy Bridge e Ivy Bridge sucesor de la línea de los Core 2, implementa cuatro núcleos, (la versión Core i7-9xxx con seis núcleos bajo micro arquitectura Bloomfield), controlador de memoria integrada con soporte para memoria DDR3, GPU integrada, velocidad de reloj hasta de 4GHz, caché de 8 MB, tecnología Hyper-Threading, Turbo Boost.

AMD Opteron

Fue el primer microprocesador con arquitectura x86 que usó conjunto de instrucciones AMD64, también conocido como x86-64. También fue el primer procesador x86 de octava generación. Fue puesto a la venta el 22 de abril de 2003 con el propósito de competir en el mercado de procesadores para servidores, especialmente en el mismo segmento que el Intel Xeon. La ventaja principal del Opteron es la capacidad de ejecutar tanto aplicaciones de 64 bits como de 32 bits sin ninguna penalización de velocidad. Las nuevas aplicaciones de 64 bits pueden acceder a más de 18 Hexabytes de memoria, frente a los 4 gigabytes de las de 32 bits.⁵⁴ Los microprocesadores Opteron han evolucionado en series con arquitectura multinúcleos, incorporan GPU, tecnología North Bridge, AMD-Virtualization, arquitectura “Bulldozer” y soporte DDR. Se tienen las series Opteron X1100 con 4 Core, GPU Radeon, 1.1GHz a 1.9 Ghz, serie Opteron Six Core con 4 Core Opteron, 2 Chipset SR5690, GPU ATI, 1.8GHz a 2.6 Ghz, núcleos Shanghai (17.6 GB/s) y Istanbul (19.2 GB/s), serie Opteron 3000 con 4 y 8 Core, desde 1.9 GHz a 3.6 GHz, AMD- Virtualization, núcleo Zurich, serie Opteron 4000 con 4, 6 y 8 Core, desde 1.8 GHz a 3.8 GHz, AMD- Virtualization, HyperTransport serie Opteron 6000 con 4, 8, 12 y 16 Core, desde 1.8 GHz a 3.8 GHz, AMD- Virtualization, HyperTransport.

AMD Sempron

Es un procesador de bajo costo con arquitectura X86 fabricado por AMD. El AMD Sempron reemplaza al procesador Duron. Incorpora con el núcleo Palermo conjunto de instrucciones AMD64, con soporte parcial para instrucciones SSE3, con cache de 128 MB hasta 256MB. Cuenta con tecnología HyperTransport (comunicación bidireccional funcional en serie o paralelo) ofreciendo comunicación integrada de alta velocidad y desempeño reduciendo también el número de buses en un sistema, integra controlador de memoria DDR.

AMD Turion 64

Es una versión de bajo consumo del procesador AMD Athlon 64 destinada a los ordenadores portátiles y constituye la respuesta comercial de AMD a la plataforma Centrino de Intel. Se presentan en dos series, ML con un consumo máximo de 35 W y MT con un consumo de 25 W, frente a los 27 W del Intel Pentium M. Es compatible con el Socket 754 de AMD y dispone de 512 o 1024 KB de cache L2 y un controlador de memoria de 64 bit integrado, se presentan en los núcleos Lancaster y Richmond. Como dato complementario dentro de la familia de procesadores para notebooks se tiene los AMD Athlon X2 de doble núcleo, 64 bits, con tecnología AMD-V (AMD- Virtualization), HyperTransport, soporte para DDR e instrucciones SSE, presentado en núcleo Manchester, Toledo, Windsors Brisbane, el procesador AMD Athlon Neo y Neo X2 para notebooks ultradelgadas y finalmente los AMD Turion X2 Ultra y AMD Turion X2 doble núcleo.

⁵⁴ Extraído el 10 de Julio de 2009 desde <http://www.taringa.net/posts/info/2026465/Microprocesadores.html>

AMD Phenom

Nombre dado por AMD a la generación de procesadores de tres y cuatro núcleos, basando su arquitectura en el K10, estos reemplazan la línea de los Athlon dos núcleos. Los Phenom, otorgan alta definición, rendimiento avanzado en procesos multitarea y consumo eficiente de energía. Existen Phenom de dos, tres y cuatro núcleos, basados en tecnología de 65 nanómetros, utilizan socket AM2+, incorpora 450 millones de transistores, controlador de memoria DDR integrado, tecnología 3D NOW, SSE, SSE2, SSE3, SSE4a. La versión AMD Phenom II presenta 3 y 4 Core, controlador de memoria integrado, tecnología HyperTransport con hasta 16 GB/s por procesador, cache L3 entre 6MB y 4MB, 512 K de cache L2, AMD-V, Cool'n'Quiet, CoolCore. La serie Phenom II se implementa en núcleos Deneb, Heka, Callisto y Thuban.

AMD Athlon II

Son procesadores AMD64, implementan la tecnología HyperTransport, Cool'n'Quiet, CoolCore, AMD-V, controlador de memoria, se presenta en versiones AMD Athlon X2 Dual –Core, AMD Athlon X3 Triple-Core, AMD Athlon X4 Quad-Core, con frecuencias desde 2.6 GHZ hasta 3.4 GHz. Basados en los núcleos Llano, Propus, Rana, Regor y Sargas (One-Core).

AMD APU serie A

Esta serie de microprocesadores consta de modelos de 2 y 4 nucleos, incorporando una GPU Radeon integrada, controlador de memoria DDR3, con frecuencias de reloj desde los 2.4 GHz a 4.2 GHz. Se presenta con modelos E2, A4, A6, A8 y A10.

AMD FX

Los microprocesadores serie FX presenta modelos como el AMD FX de 4, 6 y 8 núcleos Black Edition, implementan tecnología AMD Turbo CORE, caché hasta de 8 MB, acelerador de punto flotante, capacidad de instrucción AVX (Extensiones Vectoriales Avanzadas, para aumentar el paralelismo), FMA4 y XOP (para mejora de rendimiento en punto flotante y enteros), AES (Estándar de encriptación avanzado), tecnología HyperTransport, Controlador DRAM integrado DDR3, AMD-V, AMD PowerNow! (Cool'n'Quiet). Frecuencia de reloj entre 3.5 GHz y 4.3 GHz. Se basa en núcleos Clawhammer, San Diego.

CAPITULO 3: LENGUAJES DE PROGRAMACIÓN EN LOS MICROPROCESADORES

Introducción

Con el desarrollo de los microprocesadores como pieza central de hardware en un sistema de cómputo, también se desarrolla las técnicas de diseño e implementación de software, en un inicio con los lenguajes máquina, técnica bastante abrumadora y dispendiosa por ser un código fundamental en lenguaje binario o hexadecimal, haciendo difícil la programación al pasar del algoritmo al programa, el programador debía seguir una tabla manual donde el código binario o Hexadecimal determinaba la correspondiente instrucción, con la aparición del lenguaje assembler se ha facilitado enormemente la programación pues la codificación y decodificación antes realizada manualmente, se hace ahora gracias a un conjunto de nemotécnicos y al mismo microprocesador en su unidad de control (UC), actualmente se sigue utilizando el lenguaje ensamblador junto con lenguajes de más alto nivel como el C, C++, BASIC, FOLTRAN, PYTHON entre otros como métodos y técnicas de programación para el diseño del Sistema Operativos, que no es más que código o software diseñado para procesar información y datos entre el microprocesador y sus periféricos, ejemplos de estos Sistemas Operativos desde sus inicios son el DOS, UNIX, WINDOWS (en todas sus versiones), LINUX (en todas sus versiones), Android, OS2, etc.

Lección 11: Fundamentos de programación en assembler para microprocesadores

Para mantener un control sobre los periféricos la CPU utiliza pequeños periodos de tiempo en los cuales realiza una serie de acciones cuyo propósito es cumplir tareas que han sido organizadas de forma secuencial, estas tareas son las instrucciones y la forma secuencial es el programa, este concepto se aplica a cualquier sistema basado en microprocesadores o Microcontroladores.

Programas

Los programas contienen una secuencia de instrucciones, estas instrucciones debe poder ser almacenada en algún lugar del sistema, se habla de “memoria de programa” como el lugar donde se almacenaran las instrucciones cada una de ellas dispuesta en uno o varios contenedores denominados registros, compuestos a su vez por celdas de almacenamiento, cada una de ellas con capacidad de almacenar un “bit” que puede presentar uno de dos estados estables posibles “cero” o “uno”.

Las instrucciones y datos del programa al ejecutarse en la CPU generan datos con significado dentro del contexto de la solución del problema, estas instrucciones y datos se expresan en conjuntos de “bits”, como los denominados “Nibbles” (4 bits), “Bytes” (8 bits), “W-palabra” (16 bits), “DW- doble palabra” (32 bits) y “QW- cuádruple palabra” (64 bits). El programa es el centro nervioso de los sistemas electrónicos modernos, en los que es igualmente importante el desarrollo de componentes, su interconexión y los programas orientados a la solución de problemas utilizando dichos componentes.

Algoritmos

En términos generales los algoritmos son el conjunto finito de instrucciones o la serie de pasos que deben seguirse para solucionar un problema, en la programación de microprocesadores o Microcontroladores, se debe cumplir que cada paso o instrucción del algoritmo debe estar bien definido sin ambigüedades, la claridad es vital para que cualquier persona pueda entenderlo y realizarlo. Aunque hay varias formas de solucionar un problema, cada paso o instrucción debe regirse por la claridad y simpleza de su acción y la efectividad en conjunto para llegar a una solución.

En la programación de microprocesadores un algoritmo debe tener un principio y un final, se espera que la entrada sea procesada y produzca una salida. En los sistemas basados en microcontroladores los algoritmos pueden seguir trayectorias cerradas bien definidas haciéndolos parecer que no tuvieran un final.

Programador

El programador es el sujeto que interfiere en el proceso de creación de programas teniendo como bases fundamentales la lógica, los algoritmos y el conocimiento del lenguaje con el que se pretende desarrollar la solución. Entones el objetivo del

programador es el de entender el problema a solucionar, diseñar el algoritmo que lo resuelve, escribir la secuencia de instrucciones requeridas en el lenguaje apropiado, almacenar la secuencia de las instrucciones en la memoria interna del dispositivo con capacidad de procesamiento automático para finalmente ejecutarlo.

Lenguaje Maquina

Los microprocesadores y los circuitos digitales que los conforman y complementan responden a un conjunto de operaciones que se denomina programa máquina, el cual contiene un conjunto de instrucciones o set de instrucciones que realizan operaciones sobre un bit o los bits que conforman las palabras, las instrucciones se expresan también como conjuntos de bits ya que estos conforman el lenguaje propio de la máquina.

Se entiende que el lenguaje máquina solo se escribe con “ceros” y “unos”, llamado lenguaje binario, por lo que la escritura de los datos se hace compleja e incómoda. Este inconveniente impulsó la idea de representar la secuencias de “unos” y “ceros” con símbolos que fueran más comprensibles y fáciles de manipular, utilizando equivalencias en hexadecimal u octal. Un ejemplo de código máquina y una forma compacta de expresarlo con código hexadecimal se muestra a continuación:

Tabla 11. Código maquina⁵⁵

DIR.SEGMENTO	DIR. MEM	COD.BINARIO	COD.HEXA
17A3	0100	1011010000001000	B408
17A3	0102	100000001100010000000011	80C403
17A3	0105	100000001110110000000100	80EC04
17A3	0108	1100110100100000	CD20

Como se observa aunque se disminuye el campo de símbolos, sigue existiendo la complejidad en su interpretación por lo que se desarrolla otra técnica para identificar las instrucciones permitiendo una más fácil interpretación.

⁵⁵ CEKIT, 2002

Ventaja del lenguaje maquina: Como principal ventaja esta la velocidad de ejecución que es superior a la de cualquier otro lenguaje, esto se debe a que el programa es cargado directamente en memoria sin necesidad de una “traducción” previa.

Desventajas del lenguaje maquina: Una de las desventajas se presenta del lado del programador porque es muy difícil recordar la secuencia o cadena de caracteres binario que identifican una instrucción y no se puede identificar una relación con la operación que realiza. Otra desventaja se presenta por las diferencia entre el set de instrucciones entre uno y otro procesador, incluso de la misma fábrica. Generalmente un programa debe ser ingresado a la máquina, lo que genera lentitud y dificultad en la codificación, lo que lo hace susceptible a errores, a lo anterior se suma los inconvenientes en el proceso de depuración de errores y puesta a punto del programa.

Como se observa las desventajas superan ampliamente las ventajas por lo que el lenguaje maquina no es recomendable para desarrollar aplicaciones.

Lenguaje ensamblador

Buscando solventar los inconvenientes de la programación con solo símbolos binarios, se desarrollan intérpretes y fichas mnemotécnicas que son el origen del lenguaje ensamblador, hoy las fichas mnemotécnicas se han sustituido por símbolos o compresiones de palabras que provienen de la síntesis significativa de la instrucción u operación.

Tabla 12. Código máquina y nemotécnico⁵⁶

DIR.SEGMENTO	DIR. MEM	COD.BINARIO	COD.HEXA	MNEMOTECNICO
17A3	0100	1011010000001000	B408	MOV AH, 08
17A3	0102	100000001100010000000011	80C403	ADD AH, 03
17A3	0105	100000001110110000000100	80EC04	SUB AH, 04
17A3	0108	1100110100100000	CD20	INT 20

⁵⁶ CEKIT, 2002

El lenguaje ensamblador es considerado un lenguaje de bajo nivel, que es más fácil de utilizar que su predecesor el lenguaje máquina, pero mantiene la dependencia con respecto al tipo de procesador.

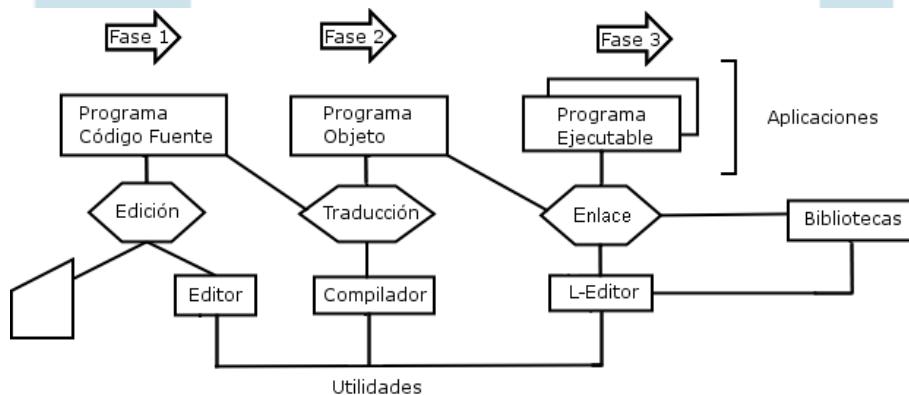
Las instrucciones de lenguajes maquina son simbolizadas por mnemotécnicos, los cuales emplean términos que permiten una fácil asociación con la operación básica que realiza dicha instrucción.

Programa en ensamblador

Los programas escritos en lenguaje ensamblador deben pasar por una fase donde se “tradicen” a lenguaje máquina o lenguaje binario para que el computador o microcomputador lo ejecute directamente, esta tarea la realiza un software especializado llamado “programa ensamblador”. El proceso de crear un programa se divide en tres partes:

1. Editar el programa, en su formato fuente
2. Ensamblarlo y enlazarlo
3. Realizar las correcciones y depuraciones necesarias, el ciclo se puede cerrar en esta parte para comenzar de nuevo en la primera.

Figura 52. Fases de realización de un programa⁵⁷



⁵⁷ Extraído el 10 de Julio de 2009 desde <http://es.wikipedia.org/wiki/Archivo:Fuente.png>

Código fuente

Es el conjunto de líneas de texto que representan las instrucciones que debe seguir la CPU para ejecutar el programa, este código describe completamente el funcionamiento del programa, aunque no es directamente ejecutable teniendo que pasar por un compilador para ser traducido a lenguaje maquina o código objeto.

Código objeto

El código objeto es el resultado de la compilación del código fuente, consiste en lenguaje máquina que puede estar distribuido en varios archivos correspondientes a varios códigos fuente que son enlazados con un programa “enlazador” o “**linker**” para producir el programa ejecutable.

Código ejecutable

Es común que se confunda el código objeto con el código ejecutable, la diferencia radica en que el código objeto presenta una estructura comprendida como símbolos, mientras el código ejecutable es un empaquetado listo para ser ejecutado en una computadora, generalmente tiene la extensión “.COM” o “.EXE”, es el resultado de la compilación y el enlazamiento “**linker**” de los códigos objetos.

Compilador e intérprete

Los intérpretes: son programas encargados de procesar y traducir cada instrucción o sentencia de un programa que está escrito en lenguaje de alto nivel a código máquina para después ejecutarla, cada instrucción se ejecuta después de traducir y comprobar que no posee errores en sintaxis. Las tareas de un intérprete son leer, examinar, traducir y ejecutar cada instrucción del programa fuente, los intérpretes incorporan un editor por medio del cual se hace la edición, interpretación y ejecución del programa, el principal inconveniente de los intérpretes surge con la lentitud ocasionada en sucesivas ejecuciones del programa. Como **Ejemplo** de intérprete tenemos el Debugger.

Los compiladores: también se encargan de traducir el programa fuente en código máquina, pero a diferencia de los intérpretes la traducción se realiza sobre la totalidad del programa fuente que después de convertirse a código máquina es ejecutado, las ejecuciones sucesivas no necesitan generar nuevamente el código máquina lo que acelera el tiempo de ejecución. Uno de los compiladores más conocidos en la programación de microcontroladores es el MASM o versiones similares como FASM, TASM y NASM, se concluye entonces que los compiladores tienen lo mejor de los ensambladores y los intérpretes.

Formas de crear programas:

Para el ciclo de aprendizaje de programación de microprocesadores e incluso de microcontroladores y DSP, es recomendable iniciar con lenguaje ensamblador, de manera directa sobre el micro como en el caso de Debugger, MASM o de forma simulada como el Simuproc, se aclara que los fundamentos de programación y algoritmos junto con lenguajes de medio nivel como C, C++, Fortran, Pascal, Basic y alto nivel como Visual Basic, Visual C, C#, Python, JAVA, AJAX, etc son vistos específicamente en otros cursos, por lo que es recomendable que el estudiante profundice dicha temática como estudio independiente ya que en este curso y módulo solo se traen a colación los conceptos y aplicación fundamental. En próximas lecciones el estudiante observa y aprende las siguientes formas de programación:

1. Utilizar Debugger como intérprete y punto de partida en la programación de microcontroladores implementando instrucciones y pequeños programas que prueben la potencia del código.
2. La primera consiste en usar un ensamblador profesional, como MASM (Macro Assembler, de Microsoft), u otras versiones como FASM, TASM y NASM, pero existen problemas de compatibilidad con Windows Vista, 7 e incluso 8 (implementación en máquina virtual).
3. Utilizar el emulador SimuProc. Que emula un procesador hipotético compatible x86, con este emulador y sus especificaciones se adquiere habilidades y se comprende con más detenimiento el funcionamiento de cada instrucción y las microoperaciones que involucran.

Lección 12: Diagrama de flujo o bloques y subrutinas

Como se ha establecido en la lección anterior una de las dificultades de utilizar lenguaje ensamblador es el hecho que cada microprocesador o microcontrolador posee su propio lenguaje ensamblador y programa(s) compilador(es). La experiencia ha demostrado que el aprendizaje de los fundamentos básicos en la programación de determinado microprocesador o microcontrolador y la habilidad en desarrollar la lógica de los algoritmos que componen la aplicación o solucionan el problema en cualquier lenguaje, es suficiente para que el programador transfiera estas habilidades a un sistema diferente.

Pseudocódigo

Es una mezcla de lenguaje de programación y lenguaje nativo (español), se emplea en la programación para realizar el diseño del algoritmo, consiste en una estructura narrativa respecto a los pasos a seguir en un algoritmo para solucionar un problema. El pseudocódigo no sigue la rigidez de la sintaxis de cada instrucción, tampoco la fluidez del lenguaje humano, es un lenguaje intermedio que permite codificar de forma sencilla al lenguaje de programación utilizado, este lenguaje se emplea en las primeras etapas del análisis o diseño del software, en lo que se conoce como diseño del algoritmo.

Para realizar un pseudocódigo se debe considerar partir de una serie de pasos simples que lleven a la solución del problema estos pasos se convierten en secuencias simplificadas utilizando pocas palabras o frases sencillas describiendo tareas simples, observe el ejemplo de Pseudocódigo: para poder hallar el área de un cuadrado.

INICIA:

LADO, AREA: ENTERO

PONER EN BLANCO LA PANTALLA

ESCRIBIR: "DIGITE LA LONGITUD DEL LADO"

LEER: LADO

AREA = LADO * LADO

ESCRIBIR: "AREA DEL CUADRADO" = AREA

TERMINA

El paso siguiente depende de la complejidad o entendimiento que el programador tenga del algoritmo pudiendo utilizar los dos o uno de los siguientes pasos:

1. La sencillez de algunos programas pueden solo requerir compactar el pseudocódigo convirtiendo cada estructura simple en una instrucción equivalente en lenguaje ensamblador.
2. Puede asociar el pseudocódigo a un diagrama de bloques o un diagrama de flujo que permitan observar la estructura general del algoritmo en un lenguaje aproximado al utilizado para programar en este caso al ensamblador con el set de instrucciones propio del dispositivo a utilizar.

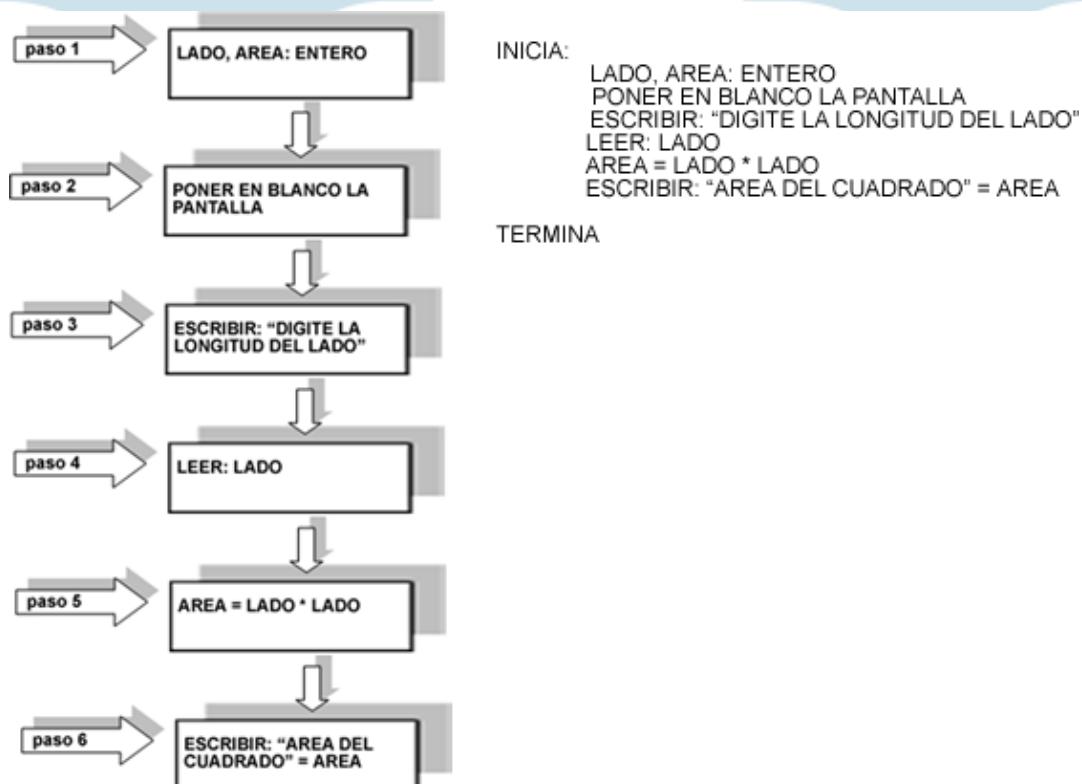
Ventajas del Pseudocódigo: Mejora la comprensión del problema y dilucida la posible solución. Facilita la solución y desarrollo del problema. Representa en

forma sencilla las operaciones repetitivas o cíclicas. Se obtiene la estructura general del programa. En el proceso de aprendizaje facilita que el estudiante este más cerca del paso siguiente, (la realización del diagrama de flujo o bloques).

Diagrama de bloques

Los diagramas de bloque son representaciones gráficas del funcionamiento de un sistema que facilita a los programadores visualizar las características de los componentes que hacen parte de la solución del problema. Consisten en una serie de bloques que puede estar conectados por flechas o números que indiquen el orden del proceso, dentro de cada bloque se encuentra un componente simple del pseudocódigo que más adelante se convertirá en la instrucción en ensamblador.

Figura 53. Fases de realización de un programa⁵⁸



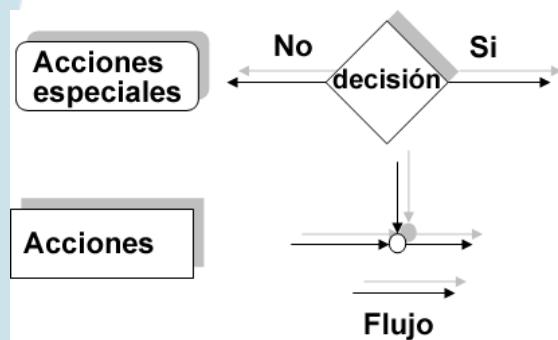
La representación de un algoritmo utilizando diagramas de bloques puede no ser lo suficientemente clara para representar el algoritmo de programación, en la mayoría de casos se elige el diagrama de flujo el cual es más compatible y se adapta a las necesidades y requerimiento de representación de las operaciones simples que realiza el microprocesador o microcontrolador.

⁵⁸ CEKIT, 2002

Diagrama de flujo

El diagrama de flujo representa más detalladamente el algoritmo de un programa en lenguaje ensamblador y en general de programación de computadores, consiste en una serie de símbolos gráficos que representan por si solos acciones o decisiones unidos por flechas que determinan el flujo del programa, los símbolos de acciones o decisiones contienen en su interior un paso de la secuencia generada por el pseudocódigo. El análisis del problema puede conducir inmediatamente al desarrollo de un diagrama de flujo que integra en un solo paso el pseudocódigo y el diagrama mismo, aunque los diagramas de flujo contienen símbolos diversos para representar las acciones, se utilizarán básicamente cuatro de ellos

Figura 54. Símbolos más comunes para diagramas de flujo⁵⁹



Acciones especiales: conformadas por: “inicio” que indica el comienzo del programa. “fin” que indica que el programa ha terminado.

Acciones: son las instrucciones que involucran movimiento de datos, operaciones aritméticas, lógicas, comparaciones sin decisión, llamado o saltos.

Decisiones: son las instrucciones que involucran comparaciones con saltos, es decir, generan bifurcaciones (caminos) dentro del flujo del programa

Flechas: las cuales simbolizan el flujo del programa.

Los caminos que describen los diagramas de flujo deben conducir a una solución, solo existe un único “inicio” y “final” de programa, aunque en los sistemas con microcontroladores en su funcionamiento pareciera que nunca tienen fin y siguen ciclos estructurados que mantienen el flujo del programa. En la programación de microprocesadores se debe evitar sumideros infinitos o espacios que sin entradas

⁵⁹ CEKIT, 2002

retornan alguna salida, es recomendable etiquetar procesos que posteriormente representan las subrutinas del programa, con esto se evitan errores en el flujo del programa final.

Ventajas de los diagramas de flujo: Son una herramienta flexible utilizada ampliamente en el aprendizaje y capacitación en programación de sistemas con microprocesadores o microcontroladores. Ayuda en la comprensión mental del programa al mostrarlo como un dibujo que es más susceptible de ser relacionado por el cerebro humano. Un diagrama de flujo bien diseñado sustituye líneas de texto extensas. Mediante un diagrama de flujo se pueden establecer redundancias cíclicas, bifurcaciones, flujos inconsistentes y errores. Permiten un análisis que pueden identificar problemas y mejorar el algoritmo.

Desarrollo de un diagrama de flujo

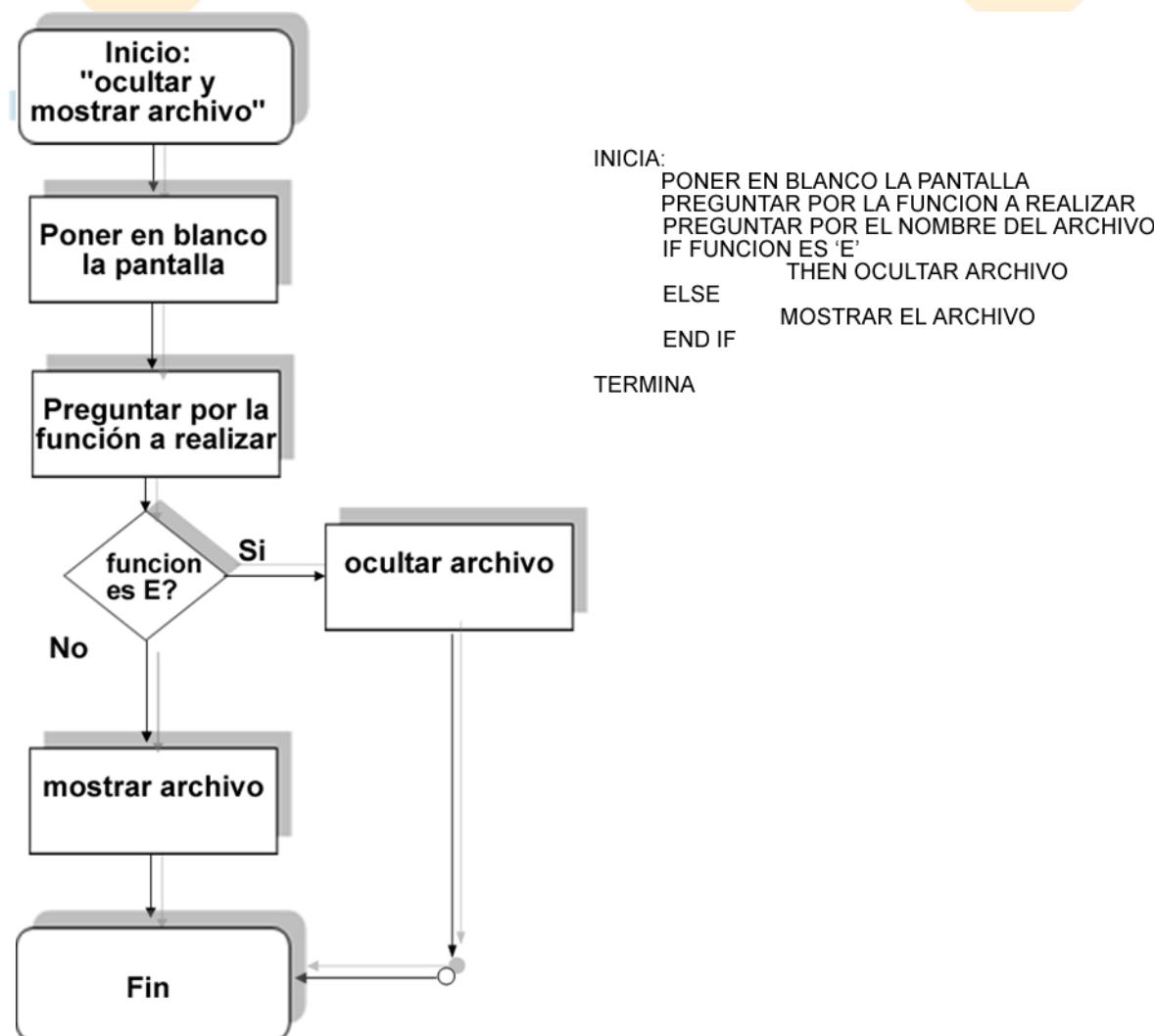
Para desarrollar un diagrama de flujo es necesario tener claridad en la lógica digital y el proceso de desarrollo de algoritmos, en general un diagrama de flujo exige ciertas acciones previas:

- Se debe tener el algoritmo desarrollado para la solución del problema, este debe especificar, las entradas (definir variables y constantes), el proceso (acciones, decisiones o bifurcaciones) y el resultado (almacenado en memoria, acción sobre periféricos (visualizar, actuar, activar)).
- Es recomendable tener el algoritmo expresado en pseudocódigo, de tal forma que este estructurado en unidades que representen operaciones simples (movimiento de datos, sumas, restas, and, or, xor, saltos, decisiones, comparaciones, llamados).
- Comenzar con una acción inicial y terminar con una final.
- Utilizar adecuadamente la simbología grafica acorde con la acción determinada en la secuencia del pseudocódigo.
- Representar el diagrama de arriba hacia abajo y/o de izquierda a derecha.
- Los símbolos se conectan por flechas que indican la dirección del flujo del programa, es recomendable utilizar solo líneas horizontales y verticales.
- Los símbolos pueden tener más de una línea de entrada a Excepción del símbolo de “fin”.
- Los símbolos de decisión deben contener al menos dos líneas de salida (si/no; falso/verdadero; cumple/no cumple).
- Los textos que se digitán y que son el pseudocódigo deben estar escritos de forma clara y legible evitando usar muchas palabras.
- Se recomienda no cruzar líneas de flujo pues se confunde con un nodo.
- Los nodos son puntos donde entran y salen líneas de flujo, los nodos no representan ninguna acción solo un punto de unión.
- No dejar líneas de flujo sin conectar.

- En caso que el diagrama de flujo haga un llamado a otro programa representarlo como una acción que se dirija a esa subrutina seguida de otra que retorne el resultado de la misma.
- Se recomienda nombrar cada programa y subprograma (subrutina) de manera que al convertirlo en lenguaje ensamblador, es acorde con el diagrama de flujo.

Ejemplo: diagrama de flujo:

Figura 55. *Diagrama de flujo para mostrar u ocultar un archivo⁶⁰.*



Existen dos tipos de diagramas de flujo aplicados a programación de microprocesadores y microcontroladores, el anterior es un denominado “diagrama

⁶⁰ ROJAS, 1997

de flujo funcional” compatible con cualquier micro, puesto que representa el algoritmo solución al problema. El segundo tipo de diagrama de flujo se denomina “diagrama de flujo detallado” en el cual cada bloque del diagrama de flujo corresponde con una sentencia del lenguaje, por lo cual el diagrama de flujo detallado es dependiente del micro específico a utilizar.

Desarrollo y Edición de un programa

Para desarrollar un programa para microprocesadores o microcontroladores, el programador debe iniciar con la “*Definición y análisis del problema*”, esto incluye el establecer las variables de entrada y salida, el proceso a realizar, junto con el dispositivo a utilizar como núcleo central (Microprocesador) y los periféricos necesarios como dispositivos I/O (Entrada/Salida), tipo y cantidad de Memoria RAM, Memoria ROM, etc. Como segundo paso “*Generar el algoritmo y Diagrama de flujo*” como solución descriptiva y gráfica del problema, con lo que se tienen el diagrama de flujo funcional para cualquier tipo de micro o el diagrama de flujo detallado para el micro seleccionado por el diseñador.

En tercera instancia se requiere “*Escribir el programa en lenguaje ensamblador*” que en el inicio de la programación requería del uso de tablas manuales para hacer la equivalencia de instrucciones a código maquina (binario o hexa), para el aprendizaje de la arquitectura y funcionamiento de los microprocesadores se utilizan editores estándar junto con compiladores como el MASM, TASM, etc, actualmente en desarrollos profesionales es ampliamente utilizado el lenguaje C como constructor de sistemas operativos, utilitarios y aplicaciones. Como cuarto paso se “*Escribir o generar la versión en lenguaje máquina del programa*”, como se ha mencionado en un comienzo no se utilizaba el compilador sino se hacía una decodificación manual de instrucciones a código maquina directamente mediante una tabla, actualmente este trabajo lo hace el compilador que genera archivos como por ejemplo ERR (errores), OBJ (Objeto-código maquina), EXE o COM (Ejecutables).

Como quinta fase se encuentra “*Depurar el programa*” que quiere decir encontrar los errores lógicos o de sintaxis presentes, inicialmente muy difícil por ser un procedimiento totalmente manual en código máquina, actualmente el software ensamblador o compilador genera los mensajes de error en el momento de insertar el comando (el caso de Debugger) o genera un archivo con los errores o mensajes de advertencia encontrados. Finalmente como un paso necesario e indispensable que aunque se menciona en última instancia, se debe integrar al programa en su edición es “*Documentar el programa*” con lo que se tiene un referente para futuras revisiones, correcciones, mejoras o reutilización del código.

La edición de un programa en lenguaje ensamblador requiere seguir una estructura particular, muy similar a la de cualquier otro programa para sistemas basados en microprocesadores o microcontroladores, básicamente cada sentencia o línea de código escrito en lenguaje ensamblador está compuesta por cuatro campos, (1) *Etiqueta*, el cual identifica un subprograma o subrutina que es un segmento de código dentro del programa principal y que ejecuta una tarea o proceso particular, en programas completamente lineales no es usual utilizar etiquetas. (2) *Nemotécnico*, contiene la instrucción que realiza la función específica indicada por el fabricante según el micro a utilizar. (3) *Operando*, contiene la información asociada con el nemotécnico o instrucción utilizada puede ser registros, datos (literales) o direcciones. (4) *Comentario*, se utiliza para describir las acciones o propósito de la instrucción dentro del algoritmo como solución al problema, con los comentarios se comprende mejor lo que hace el programa en su revisión, corrección, mejora o reutilización.

Con la información entregada en los tres primeros campos, etiqueta (Opcional según el programa), nemotécnico y Operando el programa ensamblador o compilador puede generar el código maquina correspondiente. Una función adicional del software ensamblador o compilador puede ser el asignar posiciones de memoria de programa a los listados de código en lenguaje máquina, es decir, asignar una dirección de memoria a cada instrucción convertida en código maquina (bits). Como dato curioso una instrucción en lenguaje de medio nivel como Fortran, Pascal, Basic o C, pueden ser equivalentes a 20 o 30 códigos máquina.

Bifurcación en programas

La Bifurcación se refiere a tener varios caminos, opciones o partes, en programación de micros, la bifurcación en programas se refiere a tener en ensamblador dos caminos posibles (Falso/Verdadero, Si/No), dentro del diagrama de flujo se refiere al símbolo en forma de “ROMBO” que se refiere a la decisión, lo que hace posible que el programa deje de ser un programa lineal (una instrucción tras otras), para generar un programa “bifurcado” o con varias opciones o caminos, dependiendo del estado de un “Bit”, el estado de un “Flag” (o Bandera) dentro del registro estado u otro registro de propósito especial o como resultado de la comparación entre dos registros dando como opciones el ser igual, mayor o menor). Las instrucciones de salto (o de bifurcación), son utilizadas para implementar en código fuente la bifurcación que hace posible la toma de decisiones por parte del micro. Cada repertorio de instrucciones para cada micro particular tendrá un conjunto de instrucciones de salto y bifurcación.

Programas cíclicos

Los sistemas basados en microprocesadores o microcontroladores suelen ser utilizados para implementaciones que requieren ejecutar procesos, tareas o acciones de control repetitivas, es decir, se implementan programas cíclicos los cuales tienen un inicio, una inicialización de registros e incluso bifurcaciones en donde se realizan comparaciones con las que se toman decisiones que generan “saltos” o “llamados” a sub-programas o sub-rutinas donde se pueden generar nuevos ciclos, denominados también “*bucles*” o “*lazos*”, los cuales utilizan instrucciones de salto, para generar “*Procesos repetitivos*”, es decir, que tienen un numero finito o infinito de veces que se repite la misma acción.

Subrutinas y utilización de subrutinas

Una subrutina es un sub-programa, un segmento de código que es utilizado frecuentemente por el programa principal, o reutilizado como código para otros programas, la subrutina comienza con una etiqueta (nombre de la subrutina) y termina con una instrucción de retorno o salto y que tiene la particularidad de volver una instrucción después de la instrucción que realiza el llamado a la subrutina. La subrutina es en lenguaje ensamblador similar a la denominada “*función*” en otros programas estructurados o modulares de más alto nivel, puesto que además de su similitud en estructura, es similar en su acción, porque muchas veces hay un paso de datos a y desde las subrutinas y el programa principal, conocido como “*paso de parámetros*”.

La transferencia de control desde el programa principal a la subrutina se realiza mediante una instrucción de “LLAMADO o CALL”, la cual hace que se almacene en PILA la dirección + 1 (siguiente instrucción) desde donde se realiza el llamado y el Registro Contador de Programa (PC) se inicializa con la dirección de memoria donde inicia la subrutina, cuando la subrutina termina debe entregar el control al programa principal, acción que realiza mediante la instrucción de vuelta o instrucción de “RETORNO o RETURN”, la cual toma de la PILA la dirección de memoria de retorno, que corresponde a la instrucción siguiente a la que realizó el llamado. La utilización de “subrutinas” transforma el programa lineal en lenguaje ensamblador a un programa semi-estructurado, haciéndolo más legible, abreviado y manejable, de esta forma facilitando su depuración y la escritura de grandes bloques de códigos o programas largos de una forma fácil y eficiente. El llamado frecuente a subrutinas tiene como desventaja la lentitud promedio de ejecución del programa. Instrucciones como PUSH (Colocar), POP (Sacar) y Load stack pointer (Cargar puntero de pila), son utilizadas usualmente en subrutinas.

Interrupciones

Las interrupciones son el medio para que la CPU tenga “conciencia” de su entorno, oprimir una tecla, es un evento que genera una interrupción y que le dice a la CPU que desde el teclado se está generando información que debe ser atendida. La atención de la interrupción en este caso consiste en detener el programa en ejecución y atender el teclado, obtener el código de la tecla presionada, guardando este código en una memoria intermedia (memoria intermedia de teclado – type ahead buffer). Es de notar que hay otro tipo de interrupciones en sistemas basados en microprocesadores que ocurren o suceden automáticamente 18 veces por segundo (aproximadamente) que es la que actualiza la hora del día en el sistema.

Las interrupciones son tan habituales y ocurren tan seguido que la CPU debe tener un modo eficiente de tratarlas, por lo que hace uso de un circuito integrado (chip) que le permite establecer la prioridad de la interrupción a nivel de hardware. En MS-DOS se mantiene una tabla de vectores de interrupción en la memoria baja, que empieza por la localidad de memoria “cero” y termina en la “256” (decimal). Los vectores de interrupción apuntan a localidades de memoria donde la CPU ejecuta el programa que atiende la interrupción, esta técnica se denomina “*darle servicio a la interrupción*”. Se puede definir la interrupción como una bifurcación a una localidad de memoria RAM o ROM donde la CPU inicia la ejecución de una serie de instrucciones y al terminar regresa a la siguiente localidad de memoria de la instrucción donde se ocasionó la interrupción.

Modos de direccionamiento

Los modos de direccionamiento hacen referencia a las técnicas implementadas en el micro para recuperar un dato en forma de bits almacenado en un registro del microprocesador, la memoria o periféricos de I/O (Entrada/Salida). Cada microprocesador o microcontrolador presenta modos de direccionamiento particular según su arquitectura, implementados en las instrucciones, esta información es proporcionada por el fabricante en las hojas técnicas “datasheet” y deben ser estudiadas y comprendidas por el programador y diseñador. Algunos modos de direccionamiento hacen referencia al intercambio de datos entre registros de la misma CPU, por ejemplo los modos de direccionamiento *Inherente* y de *Registro*, y otros hacen referencia al intercambio de datos entre la CPU y la memoria y/o periféricos I/O (Entrada/Salida), como por ejemplo el *direccionamiento inmediato*, *Directo* e *indirecto de Registro*.

Lección 13: Programación con debug y ensamblador

Existe un depurador muy poderoso llamado Code view, en el curso se trabaja con el Debugger bajo Windows. Es recomendable que los estudiantes utilicen primero el debugger que aunque su simplicidad hace que se tenga que repetir el programa cada vez que se modifica, esto se transforma en una ventaja pues la repetición constante hace que se adquiera habilidad y se entienda mejor el contexto del programa y el sentido de assembler.

Debug

Bug es un término utilizado en inglés para hacer referencia a una falla o error en la secuencia de instrucciones de un programa que impide la normal ejecución del mismo. Esta falla puede provenir de un error sintáctico o lógico, por lo que depurar es la acción de corregir estos errores o fallas. El Debugger o Debug es una utilería que ayuda a realizar tres tareas:

1. Ver el contenido de las memorias RAM y ROM
2. Ejecutar un programa, ya sea en su totalidad o una instrucción a la vez.
3. Ensamblar y ejecutar programas sobre la marcha.

El uso de Debugger nos permite trabajar directamente en lenguaje ensamblador con lo que se puede apreciar lo poderoso que este puede ser. Recordemos que lenguaje ensamblador, se trata de un lenguaje que se encuentra un paso antes del lenguaje real máquina (código binario), es un lenguaje donde se hace necesario suministrar con exactitud las instrucciones que debe ejecutar la CPU, a diferencia de los lenguajes de alto nivel y tal vez como punto en contra el lenguaje ensamblador requiere más instrucciones para ejecutar lo mismo, pero a favor cuenta con la posibilidad de manipular y acceder directamente a todo el hardware del sistema cosa que no pueden hacer en su totalidad los lenguajes de alto nivel.

Primeros pasos con Debugger

Debugger se encuentra localizado en la mayoría de computadores con sistema basado en DOS o Windows, el programa suele estar localizado en el directorio de Windows en System32 y puede identificarse como debug.com o debug.exe, dependiendo de la versión del DOS o Windows que tenga en su equipo. El debug puede ser invocado mediante la ventana de línea de comandos (símbolo de sistema), que generalmente se encuentra en Inicio/Todos los programas/Accesorios/Símbolo de sistema. Es hora de comenzar:

- Iniciar Debugger digitando en línea de comando Debug + enter, así, C:\Debug [enter].
- Debe aparecer un guión “-“ que indica que debug está esperando un comando.
- Digitar – r [enter], recordar que el guión aparece por defecto. “r” es el comando de “register” del debug este comando permite exhibir o modificar uno o más registros de la CPU.
- Como resultado se despliega el contenido de los siguientes registros ya estudiados, AX, BX, CX, DX, DS, ES, CS, SS, SI, DI, IP, SP, BP Y F.

¿Qué es la sintaxis?: La sintaxis de se refiere a las reglas que determinan si un conjunto de caracteres o string son válidos o no dentro de un programa.

Comando “r”: La sintaxis de este comando es “r [registro]”, donde [registro] es la notación de un registro valido de la CPU que constan de dos caracteres alfabéticos siguiendo la nomenclatura de los procesadores 808x y 80x86. Los datos desplegados mediante el comando “- r” despliegan 14 registros internos, cada uno de 16 bits, los primeros cuatro AX, BX, CX y DX, son registros de uso general y se pueden usar como registros de 8 bits, es decir se parte el registro en dos partes de ocho bits la parte alta (ejemplo AH) y la parte baja (ejemplo AL). La totalidad de registros y sus nombres se exponen nuevamente en forma simplificada:

- **AX** (acumulador). Generalmente se utiliza para almacenar resultados de operaciones, lectura o escritura desde o hacia los puertos y como área de memoria temporal (scratch pad).
- **BX** (registro base). Sirve como registro apuntador base o índice.
- **CX** (registro contador). Se utiliza como constante en operación de iteración, como un contador que automáticamente se incrementa o decrementa de acuerdo con el tipo de instrucción usada.
- **DX** (registro de datos). Se usa comúnmente como puente para el acceso de datos.
- **DS** (registro de segmento de datos). Cualquier dato sea variable o no debe encontrarse dentro del segmento.
- **ES** (registro de segmento extra). Este registro permite operaciones sobre cadenas pero puede ser una extensión del DS.
- **SS** (registro del segmento de pila). Maneja la posición de memoria donde se encuentra la pila o stack. Esta estructura se utiliza para almacenar datos en forma temporal, tanto de un programa como de las operaciones internas del PC.

- **CS** (registro del segmento de código). En el CS es donde se encuentra el código ejecutable de cada programa ligado a los diferentes modelos de memoria.
- **BP** (registro de apuntadores base). Manipula la pila sin afectar el registro de segmento SS.
- **SI** (registro índice fuente). En conjunto con DI se utiliza para manejar bloques de cadenas en memoria siendo SI el primero y DI el segundo.
- **DI** (registro índice destino). Usado en conjunto con DI, SI representa la dirección donde se encuentra la cadena y DI la dirección donde será copiada.
- **SP** (registro del apuntado de la pila). Apunta a un área específica de memoria que sirve para almacenar datos bajo la estructura LIFO (last in, first out: último en entrar primero en salir), mejor conocido como pila (stack).
- **IP** (registro apuntador de la siguiente instrucción). Apunta a la siguiente instrucción que será ejecutada en memoria.
- **F** (Registro de banderas o flags). Es un registro de 16 bits aunque no todos son utilizados, estas son las banderas y sus significados:
 - **Overflow**: NV = no hay desbordamiento; OV= si lo hay.
 - **Direction**: UP = hacia adelante; DN = hacia atrás.
 - **Interrupts**: DI = desactivadas las interrupciones; EI = activadas.
 - **Sign**: PL = positivo; NG = negativo.
 - **Zero**: NZ = no es cero; ZR = si es cero.
 - **Auxiliary Carry**: NA = no hay acarreo auxiliar; AC = hay acarreo auxiliar.
 - **Parity**: PO = paridad non ; PE = paridad par
 - **Carry**: NC = no hay acarreo CY = si lo hay.

Estructura del ensamblador

En lenguaje ensamblador toda línea de código consta de dos partes.

1. la primera siempre contiene un comando de 2, 3 o 4 letras, estos comandos de llaman mnemónicos o códigos de operación porque representan una función que debe realizar la CPU.
2. La segunda parte son los operandos o datos sobre los cuales se va a trabajar, no todos los códigos de operación requieren operandos.

Ejemplos: `mov ax,01` ➔ “mov” es el código de operación y “ax,01” son los operandos. `push` ➔ no requiere comandos

Ensamblando un programa

Para crear al instante un programa en ensamblador con debugger se utiliza el comando “A” (Assemble), este comando permite introducir código en forma de nemáticos y su sintaxis es “A[dirección]”, donde dirección es la ubicación de memoria a partir de la cual se empezara a ensamblar, sino se especifica la dirección inicial a partir de la cual se debe ensamblar, ensambla a partir de la localidad especificada por CS:IP. Ejecutado el comando “A”, debug preguntará en forma sucesiva y secuencial por la siguiente instrucción a ensamblar, cada instrucción se ensambla en el momento de ser digitada y cada byte generado se almacena en la memoria en la dirección inicial y en secuencia sucesiva.

Ejemplo: Digite “A [enter]”, como resultado se obtiene:

```
-A  
17A3:0100
```

Debug se encuentra listo para aceptar las instrucciones:

```
mov ah, 8 [enter]  
add ah, 3 [enter]  
sub ah, 4 [enter]  
int 20 [enter]  
[enter]
```

Figura 56. Entrando a Debugger⁶¹

```
C:\Users\TURION>debug  
-r  
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000  
DS=17A3 ES=17A3 SS=17A3 CS=17A3 IP=0100 NV UP EI PL NZ NA PO NC  
17A3:0100 0000 ADD [BX+SI],AL DS:0000=CD  
-a  
17A3:0100 mov ah, 8  
17A3:0102 add ah, 3  
17A3:0105 sub ah, 4  
17A3:0108 int 20  
17A3:010A  
-
```

Las localidades de memoria o segmento pueden ser distintas en cada PC. Para ejecutar el programa anterior se usa el comando “G [dirección]”, que ejecuta instrucciones a partir de una dirección. Se observa que nuestro programa comienza en el segmento 17A3:0100 y termina en el segmento 17A3:0108, si se desea ejecutar todas las instrucciones a partir de CS:IP (17A3:0100) hasta la instrucción en 17A3:0108, se digita “g108”, como resultado se obtiene:

⁶¹ ROJAS, 1997

Figura 57. Ejecutar un programa con Debugger⁶²

```
C:\Users\TURION>debug
r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A3 ES=17A3 SS=17A3 CS=17A3 IP=0100 NV UP EI PL NZ NA PO NC
17A3:0100 0000 ADD [BX+SI],AL DS :0000=CD
-a
17A3:0100 mov ah, 8
17A3:0102 add ah, 3
17A3:0105 sub ah, 4
17A3:0108 int 20
17A3:010A
-g108

AX=0700 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A3 ES=17A3 SS=17A3 CS=17A3 IP=0108 NV UP EI PL NZ NA PO NC
17A3:0108 CD20 INT 20
```

Debug despliega los registros con los resultado intermedios, se observa comparando el estado inicial y final de los registros con respecto a AX en su parte alta, es decir, AH que tiene un valor inicial AX=0000, pasando por AX=0800, sumando 3, AX=0B00 y restando 4, AX=0700 que es el resultado final de este programa. Debug ejecuta la interrupción 20 “int 20” termina el programa y regresa el control al DOS.

Desensamblar

Para desensamblar un programa que se acaba de escribir se utiliza el comando “U” que desensambla lo que se digita partiendo de la localidad de memoria especificada y la cantidad de bytes especificados con “L” que significa longitud (length), lo que muestra tanto el código digitado con nemotécnicos como el código en hexadecimal de cada instrucción.

Figura 58. Desensamblar un programa con Debugger⁶³

```
a
17A3:0100 mov ah, 4
17A3:0102 add ah, 3
17A3:0105 sub ah, 4
17A3:0108 int 20
17A3:010A
-g108

AX=0300 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A3 ES=17A3 SS=17A3 CS=17A3 IP=0108 NV UP EI PL NZ NA PE NC
17A3:0108 CD20 INT 20
-u 100 L9
17A3:0100 B404 MOU AH,.04
17A3:0102 80C403 ADD AH,.03
17A3:0105 80EC04 SUB AH,.04
17A3:0108 CD20 INT 20
```

⁶² ROJAS, 1997

⁶³ ROJAS, 1997

Existe también un comando que ayuda a rastrear la ejecución del programa haciendo un paso a paso este comando es el “Trace (t)”.

Figura 59. Comando “Trace” o de rastreo de ejecución en Debugger⁶⁴

```
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A3 ES=17A3 SS=17A3 CS=17A3 IP=0100 NU UP EI PL NZ NA PO NC
17A3:0100 0000 ADD [BX+SI],AL DS :0000=CD
-a
17A3:0100 mov ah,8
17A3:0102 add ah,3
17A3:0105 sub ah,4
17A3:0108 int 20
17A3:010A
-t
AX=0800 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A3 ES=17A3 SS=17A3 CS=17A3 IP=0102 NU UP EI PL NZ NA PO NC
17A3:0102 80C403 ADD AH,03
-t
AX=0B00 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A3 ES=17A3 SS=17A3 CS=17A3 IP=0105 NU UP EI PL NZ NA PO NC
17A3:0105 80EC04 SUB AH,04
-t
AX=0700 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A3 ES=17A3 SS=17A3 CS=17A3 IP=0108 NU UP EI PL NZ NA PO NC
17A3:0108 CD20 INT 20
```

Guardar y cargar un programa

Lo hecho hasta ahora ha sido operar un programa en memoria, para guardarlo en disco y recuperarlo después, siga los siguientes pasos, tomando el código ya trabajado:

1. Averigüe la longitud del programa restando la dirección final (última instrucción del programa) de la dirección inicial (primera instrucción del programa), la longitud se expresa en hexadecimal. → - h 10A 100, lo que resulta en “020A 000A”.
2. Crear un nombre para el programa, que incluya la vía y la extensión. → - n c:primprog.com. no más de ocho caracteres para el nombre.
3. Poner la longitud del programa en el registro CX. → - rcx, lo que despliega el contenido del registro CX, CX 0000, se coloca la longitud del programa restando la dirección final 010A de la inicial 0100 que se obtiene 000A, se digita “:000A”.
4. Dar la orden de escritura. La orden de escritura se da con “w”, - w aparece el mensaje “writing 000A bytes”, escribe la cantidad de byte en el registro CX.

El programa se almacena en Windows Vista o 7, en C:/Users/“nombre del usuario”.

⁶⁴ ROJAS, 1997

Figura 60. Procedimiento que guarda un programa desde Debugger⁶⁵

```
Microsoft Windows [Versión 6.0.6000]
Copyright © 2006 Microsoft Corporation. Reservados todos los derechos.

C:\Users\TURION>debug
-a
17A3:0100 mov ah,8
17A3:0102 add ah,3
17A3:0105 sub ah,4
17A3:0108 int 20
17A3:010A
-g 108

AX=0700 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A3 ES=17A3 SS=17A3 CS=17A3 IP=0108 NV UP EI PL NZ NA PO NC
17A3:0108 CD20 INT 20
-h 10A 100
020A 000A
-n c:\primprog.com
rcx
CX 0000
:000A
-w
Writing 0000A bytes
```

Para cargar el archivo anterior u otro archivo, se debe seguir los siguientes pasos:

1. Proporcionar el nombre del programa que se cargará usando el comando “n”. → -n c:\ primprog.com
2. Cargarlo mediante comando (L) load. → -l
3. Para verificar que fue cargado, desensamblar a partir de la localidad 100H. → -u 100 L9.

Para estar seguro de la ubicación antes de entrar a debug ejecute el comando “dir” para mostrar los archivos presentes en el lugar donde guardo el archivo.

Figura 61. Procedimiento para cargar un programa desde Debugger⁶⁶

```
10/06/2009 08:13 a.m. <DIR> Favorites
10/06/2009 08:13 a.m. <DIR> Links
27/06/2009 06:54 p.m. <DIR> Music
15/07/2009 11:23 p.m. <DIR> Pictures
15/07/2009 11:22 p.m. 10 PRIMPROG.COM
10/06/2009 08:13 a.m. <DIR> Saved Games
10/06/2009 08:13 a.m. <DIR> Searches
10/06/2009 08:13 a.m. <DIR> Videos
2 archivos 166 bytes
14 dirs 29.201.100.800 bytes libres

C:\Users\TURION>debug
-n c:\Users\TURION\primprog.com
-l
-u 100 19
1804:0100 B408 MOU AH,08
1804:0102 80C403 ADD AH,03
1804:0105 80EC04 SUB AH,04
1804:0108 CD20 INT 20
```

Salir de debug: para salir de debug digite el comando “q”, es decir, -q

⁶⁵ ROJAS, 1997

⁶⁶ ROJAS, 1997

Set de instrucciones

El set de instrucciones o conjunto de instrucciones o repertorio de instrucciones o ISA (Instruction Set Architecture), es una especificación detallada de órdenes, instrucciones o comandos implementados en un diseño particular de CPU o microprocesador que puede entender y ejecutar. Las instrucciones tienen un significado para el diseñador y programador, pues estas pueden representar o distinguir la microarquitectura particular utilizada, la compatibilidad con otros dispositivos al compartir el mismo conjunto de instrucciones, aunque implementen microarquitecturas diferentes, como entre procesadores de una misma familia o empresa como los Intel y sus Pentium o Core, así como entre empresas fabricantes compatibles en sistemas operativos como entre Intel y AMD. El set de instrucciones hace referencia a los aspectos del micro visibles al programador, como las instrucciones u órdenes, los tipos de datos, los registros, interrupciones, arquitectura de memoria, ciclos de instrucción, banderas (flags) afectados, tamaño de registros, etc.

Movimiento de datos con respecto a memoria

Las instrucciones que se encargan de mover datos en relación con la memoria se pueden categorizar por tareas específicas:

- Las que transfieren datos de un registro a otro o entre la memoria y los registros.
- Las que preparan registros para ingresar a una localidad de memoria.
- Las que manipulan la pila.
- Las que mueven grandes bloques de datos.
- Las que interactúan con dispositivos periféricos a través de puertos.

La instrucción de movimiento más común es “**mov**”, que requiere de dos operadores, su sintaxis es: **Mov destino, fuente**. El destino no puede ser el registro de segmentos CS, no se puede mover un valor inmediato a un registro de segmentos para poder hacerlo utilice un registro intermedio como AX,

Ejemplo:

Mov AX, 3
Mov DS, AX

Adicional a la instrucción “**mov**”, existen otras que operan bajo circunstancias diferentes con el mismo propósito, la instrucción “**echo**” intercambia el contenido de la fuente con el destino. Otra instrucción útil es “**xlat**” (translate), se utiliza comúnmente al trabajar con tablas o arreglos, esta instrucción reemplaza el valor de **AL** por un nuevo valor tomado de la tabla que se compone de un conjunto de

localidades contiguas en memoria, y supone que el registro **BX** contiene la dirección del primer elemento de la tabla.

Ejemplo: En resumen:

mov src, dest
mov dest, src
src : fuente: Inmediata. Registro. Memoria
dest : destino: Registro. Memoria

movz src, dest
movzx dest, src
src : fuente: Registro. Memoria
dest : destino: Registro

movs src, dest
movsx dest, src
src : fuente: Registro. Memoria
dest : destino: Registro

xchg src, dest
xchg dest, src
src : fuente: Registro. Memoria
dest : destino: Registro. Memoria

Administración de la PILA

La pila es usada automáticamente por las instrucciones “**call**”, “**int**”, “**ret**” y “**iret**”, para guardar o restaurar la dirección de retorno antes de ejecutar las rutinas indicadas por dichas instrucciones. Otros servicios que presta es para pasar parámetros entre rutinas, o bien de un lenguaje de alto nivel al ensamblador, la pila se manipula mediante las instrucciones “**push**” (empuja o almacena en la pila) y “**pop**” (extrae o saca de la pila).

Ejemplo: La instrucción “**push [1125]**” empuja a la pila el contenido de la localidad de memoria **1125**. Otro ejemplo, las instrucciones “**push ax**”, “**push bx**” almacenan en la pila los valores de los registros **AX** y **BX**, para después sacarlos de la pila con las instrucciones “**pop ax**”, “**pop bx**”.

Movimiento de bloques de datos

En la programación de alto nivel normalmente se deben inicializar arreglos o copiar cadenas de caracteres, en lenguaje ensamblador existen instrucciones que permiten trabajar con bloques de datos, estas son “**movs**”, “**stos**” y “**lods**” los dos primeros se utilizan con un prefijo de repetición, como **REP**, lo que permite repetir

una operación tantas veces como se estipule en el registro **CX**, lo que conlleva a una minimización del código.

Existen otros prefijos como **REPE** (repita hasta encontrar la igualdad) o **REPZ** (repita hasta que el valor sea cero), **REPNE** (*repita mientras el valor no sea igual*) o **REPNZ** (*repita mientras el valor no sea cero*) usados en combinación con las instrucciones “**cmp**” (comparar) y “**scas**” (escanear-cadena).

Trabajo con bloques

El movimiento de grandes bloques de memoria y su inicialización se basan en fuente-destino controlados por **SI-DI** y un bit en el registro de flags que especifica la dirección o sentido en la cual se debe trabajar: “**cdl**” hacia adelante o “**std**” de atrás hacia adelante. Las instrucciones “**movs**” tiene dos modalidades, la primera puede mover bytes “**movsb**” o palabras “**movsw**” estando en función de los registros pares **DS:SI** como fuente y **ES:DI** como destino.

Instrucciones lógicas y aritméticas

Las operaciones aritméticas y lógicas son parte del repertorio de instrucciones de cualquier lenguaje, en x86 se incorporan las instrucciones suficientes para efectuar las operaciones aritméticas básicas (suma, resta, multiplicación y división), las instrucciones tienen la capacidad de trabajar con operando de 8 – 16 bits con y sin signo, se debe recordar que el bit más significativo del byte o palabra es el signo.

Suma: se pueden utilizar tres instrucciones las cuales a su vez se agrupan dos que funcionan con dos operandos y una con solo un operando:

Con dos operandos:

- “**add**” o suma. Sintaxis “**add** destino, fuente”. **Ejemplo:** ADD DX, 5.
- “**adc**” o suma con acarreo. La instrucción realiza la suma y automáticamente toma en cuenta el acarreo, usa generalmente 32 bits para sus operaciones, las operaciones se realizan sobre el registro DX:AX. Sintaxis “**adc** destino, fuente”. **Ejemplo:** ADC DX, 5.

Un solo operador:

- “**inc**” o incremento. Es una operación de incremento en uno para cualquier registro o localidad de memoria. Sintaxis “**inc** [registro o posmen]”, **Ejemplo:** INC AX. La desventaja es que “**inc**” trata a su operando como operando sin signo, si un byte contiene **FFH** o todos los bits a uno, aplicando “**inc**” todos los bits pasan a cero.

Resta: son la contraparte de la suma también presenta tres instrucciones agrupadas en dos con dos operandos y una con uno solo.

Con dos operandos:

- “**sub**” o resta. Sintaxis “**sub** destino, fuente”. **Ejemplo:** SUB AX, DX.
- “**sbb**” o resta con acarreo. La instrucción realiza la resta y automáticamente toma en cuenta el acarreo. Sintaxis “**sbb** destino, fuente”. **Ejemplo:** SBB AX, DX.
- Un solo operador:
- “**dec**” o decremento. Es una operación de decremento en uno para cualquier registro o localidad de memoria. Sintaxis “**dec** [registro o posmen]”. **Ejemplo:** DEC DX.

Existe una variación de la operación resta, la instrucción “**neg**” resta el operando de cero. Si digitamos “**NEG AX**” restará el contenido AX a la cantidad cero “0”, como resultado se obtiene la negación, es decir, el cambio de todos los unos por cero y todos los ceros por unos y al resultado se le suma AX.

Multiplicaciones: es un caso singular de sumas repetitivas, se presentan dos instrucciones:

- “**mul**” para multiplicar valores con signo. Sintaxis “**mul** [multiplicador]”.
- “**imul**” para multiplicar valores sin signo. Sintaxis “**imul** [multiplicador]”.

Las restricciones que tiene la multiplicación son, que el multiplicando no se define explícitamente, debe encontrarse en el registro AX y tener el mismo tamaño del multiplicando (8-16 bits), la multiplicación de 8 bits no debe producir un resultado mayor a 16 bits y una operación de 16 bits no puede producir un resultado mayor a 32 bits, el resultado se encuentra en el registro par DX:AX.

Ejemplo 1: Pone en AL el máximo valor representable en 8 bits y lo multiplica por 6, dejando el resultado en AX.

Ejemplo 2: AX = CX=65535, multiplica AX y CX y deja el resultado en el registro par DX:AX

Figura 62. Como multiplicar⁶⁷

```
C:\Users\TURION>debug
-a
17A3:0100 mov al,ff
17A3:0102 mov cl,9
17A3:0104 mul cl
17A3:0106 int 20
17A3:0108
-g106

AX=08F7 BX=0000 CX=0009 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A3 ES=17A3 SS=17A3 CS=17A3 IP=0106 OV UP EI PL NZ NA PO CY
17A3:0106 CD20 INT 20
-
-
```

```
C:\Users\TURION>debug
-a
17A3:0100 mov ax,ffff
17A3:0103 mov cx,ffff
17A3:0106 mul cx
17A3:0108 int 20
17A3:010A
-g108

AX=0001 BX=0000 CX=FFFF DX=FFFF SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A3 ES=17A3 SS=17A3 CS=17A3 IP=0108 OV UP EI PL NZ NA PO CY
17A3:0108 CD20 INT 20
-
-
```

División: de igual forma que la multiplicación la división presenta dos instrucciones:

- “**div**” para dividir valores con signo. Sintaxis “**div [divisor]**”.
- “**idiv**” para dividir valores sin signo. Sintaxis “**idiv [divisor]**”.

En la división el dividendo se almacena en AX, siguiendo las mismas pautas que en la multiplicación, para un divisor de 8 bits se espera un dividendo de 16 bits en AX, pero un divisor de 16 bits requiere un divisor de 32 bits en el registro par DX:AX, siendo la palabra alta DX y la palabra baja AX. Cuando la división es un byte, el cociente se almacena en el registro AL y el residuo en AH. Si el divisor es una palabra, el cociente se almacena en AX y el residuo en DX.

Comparación: Se utiliza la instrucción “**cmp**” la comparación es una resta simple que no afecta el contenido del operando fuente, pero el resultado afecta al registro de banderas.

Ejemplo: “**CMP BX,AX**”.

Ejemplo: En resumen: donde arg1 y arg2 son argumentos o registros.

cmp arg1, arg2

cmp arg2, arg1

⁶⁷ ROJAS, 1997

Comparación de bits: se utiliza “**test**”, la cual tiene la capacidad de comparar bits para poder tomar decisiones.

Ejemplo: “TEST AL, 1”, esto prueba el estado del bit 1 en el registro AL.

Comparando cadenas: las dos instrucciones utilizadas son “**scans**” – scan string o buscar cadenas y “**cmps**” – compare string o comparar cadena, este tiene dos variantes “**cmpsb**” que compara el byte y “**cmpsw**” compara palabra. Estas instrucciones se utilizan en conjunto con los prefijos:

- “**repe**” o “**repz**”, repite la instrucción mientras la bandera de cero se encuentra prendida o hasta que el registro CX sea mayor que cero.
- “**repn**” o “**repnz**” es el inverso del anterior, repitiendo la instrucción mientras la bandera de cero sea cero o hasta que CX sea mayor que cero.

Instrucciones lógicas: existen cuatro tipos de operaciones lógicas: “**and, not, or**” y “**xor**” estas efectúan la operación bit por bit sobre sus operandos. “**and, or y xor**” trabajan sobre operandos destino – fuente, y la operación “**not**” complementa todos los bits de un único operando.

Desplazamiento de bits: assembler cuenta con instrucciones que permiten desplazar o rotar bits dentro de un registro (byte o palabra).

Sirven para desplazar bits:

- La instrucción “**shl**” Shift left : desplazamiento a la izquierda.
- La instrucción “**sal**” Shift arithmetic left : desplazamiento aritmético a la izquierda.
- La instrucción “**sht**” Shift right : desplazamiento a la derecha.
- La instrucción “**sar**” Shift arithmetic right : desplazamiento aritmético a la derecha.

Sirven para rotar bits:

- La instrucción “**rcl**” rotate left thru carry : rotación con acarreo a la izquierda.
- La instrucción “**rcr**” rotate right thru carry : rotación con acarreo a la derecha.
- La instrucción “**rol**” rotate left : rotación a la izquierda.
- La instrucción “**ror**” rōtate right . rotación a la derecha.

Ejemplo: el valor inicial de AL=1 y al aplicar la instrucción “**shl**” se presenta un desplazamiento a la izquierda con lo que equivale a multiplicar por 2.

Figura 63. Desplazamiento de bits con instrucción “SHL”⁶⁸.

```
C:\Users\TURION>debug
a
17A3:0100 mov al,1
17A3:0102 shl al,1
17A3:0104 shr al,1
17A3:0106 shr al,1
17A3:0108 int 20
17A3:010A
g102

AX=0001 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A3 ES=17A3 SS=17A3 CS=17A3 IP=0102 NV UP EI PL NZ NA PO NC
17A3:0102 D0E0      SHL    AL,1
g104

AX=0002 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A3 ES=17A3 SS=17A3 CS=17A3 IP=0104 NV UP EI PL NZ AC PO NC
17A3:0104 D0E0      SHL    AL,1
g106

AX=0004 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A3 ES=17A3 SS=17A3 CS=17A3 IP=0106 NV UP EI PL NZ AC PO NC
17A3:0106 D0E0      SHL    AL,1
g108

AX=0008 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A3 ES=17A3 SS=17A3 CS=17A3 IP=0108 NV UP EI PL NZ AC PO NC
17A3:0108 CD20      INT    20
-
```

Instrucciones de control de flujo: son las encargadas de generar saltos absolutos o condicionales y llevar a diversas partes del programa principal denominadas rutinas o subrutinas.

Los saltos incondicionales: “**jmp loc**” la siguiente dirección a ejecutar se localiza en la localidad de memoria “loc”.

Salto de igualdad: “**je loc**” si el resultado de la operación anterior “**cmp**” arroja una igualdad en los operandos se lleva a la dirección especificada por “loc”.

Salto de no igualdad: “**jne loc**” si el resultado de la operación anterior “**cmp**” arroja que los operando no son iguales se lleva a la dirección especificada por “loc”.

Saltar si mayor:

- “**loc jg**” si primer operador de la anterior instrucción “**cmp**” es mayor que el segundo.
- “**jge loc**” si primer operador de la anterior instrucción “**cmp**” es mayor o igual que el segundo.
- “**ja loc**” similar a “**jg**”.
- “**jae loc**” similar a “**jge**”.

Saltar si menor:

- “**jl loc**” si primer operador de la anterior instrucción “**cmp**” es menor que el segundo.
- “**jle loc**” si primer operador de la anterior instrucción “**cmp**” es menor o igual que el segundo.
- “**jb loc**” similar a “**jl**”.

⁶⁸ ROJAS, 1997

- “**jbe loc**” similar a “**jle**”.

Salto de desbordamiento: “**jo loc**” si la expresión aritmética anterior tuvo un desbordamiento.

Ir a cero:

- “**jnz loc**” si una operación aritmética no resultó en cero, salta a la localización de memoria dada por “loc”.
- “**jz loc**” si una operación aritmética resultó en cero, salta a la localización de memoria dada por “loc”.

Funciones de llamado:

- “**call proc**”, utilizado en subrutinas, coloca la dirección siguiente de la instrucción “**call**” y ejecuta la subrutina identificada por la etiqueta o dirección “proc”.
- “**ret [valores]**” retorna de una subrutina con un valor cargado en registro.

Instrucciones de bucle.

- “**bucle arg**” utiliza CX como contador que es decrementado para ejecutar un bucle.
- “**loopx arg**” con x representando a “**loope**, **loopne**, **loopnz** y **loopz**”, esta instrucción decrementa CX en el bucle y salta a la dirección especificada por “arg”.

Otras instrucciones de control:

- “**nop**” no hace nada, solo consume ciclos máquina.
- “**wait**” espera a que la CPU realice el ultimo calculo.

Interrupciones

Las interrupciones generan interrupciones en el flujo normal del programa para atender un evento interno o externo y ejecutar un programa de interrupción, al finalizar el programa o atender la interrupción se devuelve el control al programa principal en ejecución. “**int arg**” es la instrucción y su sintaxis utilizada para referirse a una interrupción (específicamente de software), existen interrupciones varios tipos de interrupciones:

- Interrupción por hardware, como por ejemplo en teclados.
- Interrupción por software, usadas para transferir el control al núcleo del sistema operativo, esta es la que se activa con la instrucción “**int**”.

Las interrupciones de software se dividen en:

- Interrupciones de BIOS (Basic Input / Output System : Sistema básico de entrada / salida), estas controlan en su mayoría los periféricos, aunque también pueden abarcar algunas funciones de disco.
- Interrupciones DOS (Disk Operating System : Sistema Operativo en Disco), encargadas de administrar la memoria, el disco y algunas que manejan entrada y salida de información.
- Interrupciones excepcionales, como dividir por cero o acceder a un área de memoria protegida.

Las interrupciones más usuales para trabajar con Debugger, MASM u otro ensamblador son:

- Int 20h - Terminar programa MS-DOS.
- Int 21h - MS-DOS. Esta tiene una serie de funciones: que puede visualizar en los archivos del curso.

Las interrupciones DOS se usan cargando el parámetro de la función en el registro “AH” e invocando la interrupción correspondiente, es de aclarar que no todas las interrupciones DOS tienen funciones.

Igualdades: sirven para definir con un nombre simbólico a una constante. Esto permite que el programa sea más legible.

Ejemplo: *LlamadaMSDOS equ 21h*

La anterior igualdad hace que en lugar de escribir en el cuerpo del programa “*int 21hN*” se pueda escribir “*int LlamadaMSDOS*”.

Lección 14: Programación con simuproc y MASM

La programación inicial con Debugger es un paso fundamental, didáctico y práctico para entender un poco más el funcionamiento del microprocesador núcleo de nuestro PC, pero como intérprete de comandos se hace dispendioso la edición y depuración de programas largos, por tanto, es conveniente continuar el aprendizaje de los microprocesadores utilizando un entorno un poco más sofisticado y seguro. Simuproc es un software de uso libre que simula un procesador hipotético similar al 8080, con grandes ventajas entre ellas el poder seguir el comportamiento interno del micro, para de esta forma continuar con el

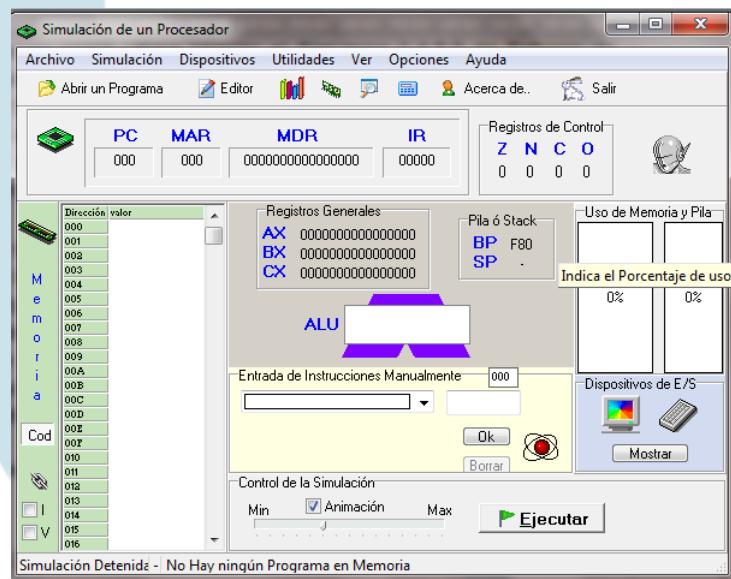
ensamblador MASM o compatible, el cual trabaja directamente con el micro del PC, se recomienda trabajar con sumo cuidado, documentarse muy bien y seguir la guía del tutor de práctica en el Centro, puesto que se está trabajando sobre el procesador en tiempo real.

Simuproc

Simuproc es un simulador de un procesador hipotético de 16 bits, es un software Freeware por lo que es gratuito, puede ser usado y distribuido libremente. Sus desarrolladores tienen como última versión el Simuproc 1.4.3.0 de Febrero de 2004⁶⁹, esta herramienta se sigue utilizando dado su potencial didáctico en el aprendizaje del funcionamiento interno de un microprocesador y en la programación en ensamblador, además de evitar que los estudiantes en su proceso de aprendizaje causen fallas y perdida de datos temporales al practicar en sus PCs sobre el microprocesador real directamente.

Trabajar con un simulador como Simuproc en el aprendizaje, tiene muchas ventajas además de ver y estudiar el comportamiento detallado de los datos dentro de un microprocesador en la ejecución de cada instrucción que conforma un programa, se destaca la ventaja que en caso de sobre flujo (Overflow) o bucle infinito, solo tiene que cerrar el programa o por medio del administrador de archivos terminar el proceso relacionado y volver a ejecutar Simuproc, sin correr el riesgo de perder información, ya que este simulador no bloquea el PC.

Figura 64. Ventana principal de Simuproc



⁶⁹ Simuproc, 2013

Características de Simuproc

Simuproc presenta un entorno fácil de interpretar y de manejar, facilitando el aprendizaje independiente, cuenta con más de 50 instrucciones muchas similares a las Intel x86 (90%) y otras propias (10%) diseñadas para facilitar el ingreso o salida de datos a los periféricos como teclado, pantalla, entre otros, que también se presentan como dispositivos virtuales. Con respecto a la última versión Simuproc 1.4.3.0 de Febrero de 2004⁷⁰ soporta números de tipo flotante usando IEEE 754 y enteros desde -2G (-2147483648) hasta 2G (2147483647), utilizando instrucciones como ADDF, SUBF, MULF, DIVF para el tratamiento de datos, LDF y STF para intercambio de datos entre registros y memoria.

Al iniciar el software Simuproc, el usuario se encuentra con una interfaz compuesta de un menú principal (Archivo, Simulación, Utilidades, Ver, Opciones, Ayuda), una barra de herramientas con iconos de acceso rápido a las principales funciones (Abrir programa, Editor, estadísticas, Acerca de, Salir, entre otros), más abajo se encuentra gráficamente los registros PC, MAR, MDR, IR, Registro de control (con los flags Z, N, C, O), Registros generales (AX, BX,CX), Pila (BP, SP), ALU, Memoria principal (donde se aloja el programa), Uso de memoria y pila, periféricos pantalla y teclado, entrada de instrucción manual en posición de memoria indicada por el usuario, control de la simulación (para ejecutar la animación a velocidad mínima o máxima), la opción “Ejecutar” (ejecuta el programa en memoria principal), y finalmente en la barra inferior el estado de la simulación y la acción que cada instrucción realiza en el microprocesador y periféricos.

En el menú principal se encuentras otras opciones para configurar Simuproc, seleccionar idioma, ver la versión de Simuproc instalado, Ayuda para ver tips sobre instrucciones y otras opciones que se mencionan brevemente en los siguientes párrafos.

Incluye un editor accesible en el menú “Utilidades/Editor” o “Ctrl+I” o mediante ícono en menú de herramienta, generando una nueva ventana, este editor tiene resaltador de sintaxis para facilitar la edición correcta de código en sus instrucciones, tiene opciones muy útiles como la tabulación para ordenar el código en ensamblador, deshacer, rehacer, cortar, copiar, pegar, imprimir, guardar entre otras accediendo también con clic derecho sobre el editor. En el menú “Utilidades/Modificar una Posición de Memoria” o “Ctrl+M” o por el ícono en el menú de herramientas, se puede inicializar variables o constantes ubicadas en

⁷⁰ Simuproc, 2013

una posición de memoria determinada, incluso en ejecución del programa. Con la opción el menú “Utilidades/Vigilante de Memoria” o “Ctrl+W” o por el ícono en el menú de herramientas, se pueden observar valores de variables en las posiciones de memoria indicadas por el usuario, con hasta los últimos cinco (5) valores. Finalmente en el menú “Utilidades/Conversión de bases” o “Ctrl+B” o por el ícono en el menú de herramientas, permite realizar la conversión entre las bases numéricas más usuales como binario, hexadecimal, octal y decimal, incluso puede convertir entre números de base 2 hasta la base 36 y un conversor entre números de punto flotante en base 10 y base 2 en formato IEEE 754 (32 bits).

Simuproc cuenta con las opciones en el menú “Simulación” para ejecutar el programa en memoria, reiniciar los registros (ponerlos a cero) y generar las estadísticas del programa ejecutado en memoria, como velocidad de procesamiento, tiempo promedio de ejecución de una instrucción, saltos manejo de pila, operaciones aritméticas, lógicas y de I/O (Entrada/Salida). Presenta dispositivos virtuales o simulados que permiten a Simuproc interactuar con el mundo exterior (usuario), entre ellos están, teclado y pantalla (formato de datos de 32 bits), Reloj (segundos del sistema 0-59), Switches (16 switches, 0-15), PC Speaker (Genera sonidos por el altavoz interno) opera en la frecuencia entre 7Hz y 32767Hz la duración en milisegundos se guarda en el registro BX.

Especificaciones de Simuproc

Simuproc simula una capacidad de memoria principal (programa y datos) de 4096 (4K) posiciones de memoria de 16 bits (000 - FFF), capacidad para trabajo con constantes y variables en binario y direcciones de memoria en Hexadecimal. Registros generales que interactúan directamente con el ALU (Hacen parte de la matriz de registros), con un tamaño de 16 bits denominados AX, BX y CX. Registros apuntadores como Program Counter (PC), Instruction Pointer (IP), Memory Address Register (MAR), Memory Data Register (MDR), Instruction Register (IR). Registros de PILA como Base Pointer (BP-valor por defecto F80), Stack Pointer (SP). Registros de control con los Flags o banderas más usuales, Zero flag (Z), Negative o Sign Flag (N), Carry Flag (C), Overflow Flag (O).

Instrucciones Soportadas

Las siguientes instrucciones son tomadas de la información técnica del desarrollador del software, esta documentación se encuentra disponible en

<https://sites.google.com/site/simuproc/home> para que el lector profundice en su aprendizaje. El autor entrega la siguiente información⁷¹.

Tabla 13. Formato de presentación de las Instrucciones⁷²

XX - INST [parámetro]	Dónde: XX significa el Código de la Instrucción INST es la instrucción [parámetro] es el parámetro si esta tiene o [parametro1,parametro2] si el parámetro es doble	En este espacio se colocan algunos ejemplos
-----------------------	---	---

Tabla 14. Instrucciones Soportadas por Simuproc 1.4.3⁷³

01 - LDA [mem]	Cargue en AX el contenido de la dirección de Memoria especificada.	Digamos que en la posición de memoria 1F está el valor 10111, después de ejecutada la instrucción LDA 1F se obtiene que AX=10111. Es equivalente a usar la instrucción MOV AX,1F Hay casos donde es mejor usar MOV si se desea pasar datos sin tener que pasarlos por AX.
02 - STA [mem]	Guarda el contenido de AX en la dirección de Memoria especificada.	Supongamos que tengo el valor 1010110 en el registro AX y quiero llevarlo a la posición de memoria 3C, la instrucción es STA 3C Es equivalente a usar la instrucción MOV 3C,AX Es mejor usar MOV debido a que si quiero pasar algún dato de una dirección de memoria a otra usando LDA y STA serían dos instrucciones: LDA mem1 y luego STA mem2, mientras que con MOV será así: MOV mem2,mem1
03 - XAB		
04 - CLA		
05 - HAC	Hace AX = 0	Borra el contenido de AX
06 - PUSH [registro]	Envía el valor del registro especificado a la pila	Colocar datos en la PILA
07 - POP [registro]	Trae de la PILA	Trae de la Pila el último Valor llevado por PUSH (indicado por el registro SP) y lo almacena en el registro especificado.

⁷¹ Simuproc, 2013

⁷² Simuproc, 2013

⁷³ Simuproc, 2013

En todas las instrucciones desde 08 hasta la 18, “Dest” puede ser una dirección de Memoria o un Registro⁷⁴.

Tabla 15. Instrucciones Soportadas por Simuproc 1.4.3⁷⁵

08 - INC [dest]	Incrementa en 1 el destino especificado, el parámetro puede ser una dirección de memoria o un registro.	Si en la posición de memoria EB está el valor 1001, al ejecutar INC EB se obtiene que ahora el valor de EB es 1010.
09 - DEC [dest]	Decremento en 1 el destino especificado.	Si el destino queda = 0, se vuelve Z = 1
10 - MOV [dest,orig]	Copia el valor almacenado en el origen al destino. El destino y/o origen pueden ser registros o direcciones de memoria o combinación de estos.	Para copiar lo que está en la posición de memoria 12E a la posición D2 se usa la instrucción MOV D2,12E
11 - AND [dest,orig]	Y lógico, hace un Y lógico entre todos los bits de los dos operандos escribiendo el resultado en el destino. Los parámetros pueden ser direcciones de memoria o Registros. La siguiente regla aplica: 1 AND 1 = 1 1 AND 0 = 0 0 AND 1 = 0 0 AND 0 = 0	Si en AX tengo el número 1001101 y en la pos 3F tengo el número 11011. al ejecutar la instrucción AND AX,3F obtendré en AX el resultado 1001. El Y lógico lo que hace es dejar los bits en común que tengan los dos números.
12 - NOT [destino]	NO lógico, invierte los bits del operando formando el complemento del primero. NOT 1 = 0 NOT 0 = 1	Colocar datos en la PILA
13 - OR [dest,orig]	O inclusivo lógico, todo bit activo en cualquiera de los operandoos será activado en el destino. La siguiente regla aplica: 1 OR 1 = 1 1 OR 0 = 1 0 OR 1 = 1 0 OR 0 = 0	Si en 3A tengo el número 1001101 y en la pos 3B tengo el número 11011. al ejecutar la instrucción OR 3A,3B obtendré en 3A el resultado 1011111.

⁷⁴ Simuproc, 2013

⁷⁵ Simuproc, 2013

Tabla 16. Instrucciones Soportadas por Simuproc 1.4.3⁷⁶

14 - XOR [dest,orig]

O exclusivo, realiza un O exclusivo entre los operandoos y almacena el resultado en destino. La siguiente regla aplica:

- 1 XOR 1 = 0
- 1 XOR 0 = 1
- 0 XOR 1 = 1
- 0 XOR 0 = 0

Si en 3A tengo el número 1001101 y en la posmem 3B tengo el número 11011. al ejecutar la instrucción XOR 3A,3B obtendré en 3A el resultado 1010110

15 - ROL [dest,veces]

Rota los bits a la izquierda las veces especificadas(en decimal), los bits que salen por la izquierda re-entran por la Derecha. En el Carry Flag queda el ultimo bit rotado.

Supongamos que en la posición 7E tengo el numero 101110

Al Ejecutar...	obtengo en 7E	C=
ROL 7E,2	10111000	0
ROL 7E,7	1011100000000	0
ROL 7E,13	110000000000101	1

16 - ROR [dest,veces]

Rota los bits a la derecha las veces especificadas(en decimal).

Los Bits que salen por la derecha re-entran por la izquierda. El Carry Flag guarda el ultimo bit rotado.

17 - SHL [dest,veces]

Desplaza los bits a la izquierda el número de veces especificado(en decimal).

Agregando ceros a la derecha, el Carry Flag guarda ultimo bit desplazado.

18 - SHR [dest,veces]

Desplaza los bits a la Derecha el número de veces especificado(en decimal), agregando ceros a la izquierda, el Carry Flag guarda ultimo bit desplazado.

Supongamos que en la posición 1A tengo el numero 101110

Al Ejecutar...	obtengo en 1A	C=
SHR 1A,2	1011	1
SHR 1A,6	0	1
SHR 1A,11	0	0

20 - ADD [mem]

Sumar:

AX = AX + el contenido de la dirección de memoria.

Si el resultado de la suma supera los 16 bits, el resultado queda asi: en BX los bits mas significativos y en BX los menos, tambien se activa el Overflow flag.

21 - SUB [mem]

Restar:

AX = AX - el contenido de la dirección de memoria.

⁷⁶ Simuproc, 2013

Tabla 17. Continuación - Instrucciones Soportadas por Simuproc 1.4.3⁷⁷

22 - MUL [mem]	Multiplicar: AX = AX * el contenido de la dirección de memoria. Si el numero resultante supera su longitud en binario de 16 bits, este resultado se parte almacenando los bits más significativos en el Registro BX.	Si el resultado de una operación fuera 101101000111100010111 (21 bits) Los registros quedarían así: A = 1000111100010111 (los últimos 16bits) B = 10110 (los primeros Bits (los mas significativos)) También se activa el Flag Overflow, para indicar que en la última operación sucedió esto.
23 - DIV [mem]	Dividir	AX = AX / el contenido de la dirección de memoria, BX=AX % el contenido de la dir de memoria (BX = modulo o residuo).
24 - CLN	Limpia el Negative Flag.	N = 0
25 - CLC	Limpia el Carry Flag.	C = 0
26 - STC	Pone el Carry Flag.	C = 1
27 - CMC	Complementa (invierte) el Carry Flag.	Si C = 1 vuelve C = 0 y viceversa.
29 - LOOP [mem]	Decrementa CX y salta a la Pos de memoria	Si CX no es cero.
30 - JMP [mem]	Salto incondicional.	PC = dirección de memoria donde está la siguiente instrucción a ejecutar"
31 - JEQ [mem]	Saltar si son iguales. Si Z = 1, PC = contenido de la memoria.	Función equivalente en C: if (AX == mem)
35 - JC [mem]	Saltar si el Carry Flag está activado.	Si C = 1, PC = contenido de memoria.

⁷⁷ Simuproc, 2013

Tabla 18. Continuación - Instrucciones Soportadas por Simuproc 1.4.3⁷⁸

32 - CMP [mem]

Compara AX con [mem], si AX es mayor, Z=0 N=0, si es igual Z=1 N=0, si es menor Z=0 N=1

Supongamos que en AX tengo el número 10110 y en la posición de memoria 4G tengo el número 1100, al ejecutar la instrucción CMP 4G obtengo que como el numero almacenado en AX es mayor entonces los Flags de Control me quedan así: Z=0 y N=0
Nota: Solo en las versiones 1.3.6.2 y anteriores En AX quedaba el resultado de la resta (de la comparación), Ahora ya **no**. Sugerencia: si necesita el valor original de AX puede usar la pila para almacenarlo temporalmente

33 - JME [mem]

Saltar si es Menor. Si N = 1, PC = contenido de la memoria

Supongamos que ejecuto esta instrucción así JME 3F inmediatamente después de ejecutar la instrucción del ejemplo que coloque en la instrucción 32, al ejecutar JME 3F se verifica el Flag N, y como en este caso se encuentra en 0 (porque el número no es menor) entonces no se realiza dicho Salto a 3F porque el valor de PC no se modifica, el programa sigue su ejecución.
 Función equivalente en C:
`if (AX < mem)`
 y Si necesitas hacer un:
`if (AX <= mem)`
 inmediatamente después de la instrucción JME colocas una JEQ

34 - JMA [mem]

Saltar si es Mayor. Si Z = 0 y N = 0, PC = contenido de memoria.

Supongamos que ejecuto esta instrucción así JMA 2B inmediatamente después de ejecutar la instrucción del ejemplo que coloque en la instrucción 32, al ejecutar JMA 2B se verifican los Flag N y Z, y como en este caso los dos son 0 (porque el número es menor) entonces si se realiza dicho Salto a 2B ya que el valor del PC se modifica, el programa sigue su ejecución saltando a la dir de mem especificada.

Función equivalente en C:
`if (AX > mem)`

36 - JNC [mem]

Saltar si el Carry Flag no está activado. Si C = 0, PC = contenido de memoria

37 - JO [mem]

Saltar si el Overflow Flag está Activado. Si O = 1, PC = contenido de memoria

38 - JNO [mem]

Saltar si el Overflow Flag no está activado. Si O = 0, PC = contenido de memoria

39 - JNE [mem]

Saltar si no son iguales. Si Z = 0, PC = contenido de memoria. Función equivalente en C:
`if (AX != mem)`

⁷⁸ Simuproc, 2013

Tabla 19. Continuación - Instrucciones Soportadas por Simuproc 1.4.3⁷⁹

40 - LDT	Lee un valor del Teclado y lo lleva al registro AX	Esta instrucción es para comunicarse con el usuario, pidiéndole que entre un Dato; Puede colocar una descripción del dato que pide, que se mostrará en tiempo de ejecución.
41 - EAP	Escribe en Pantalla el contenido del registro AX	Esta instrucción también es para comunicarse con el usuario; Puede colocar una descripción del dato que se entrega, este se mostrará en tiempo de ejecución.
42 - MSG	Muestra un mensaje en pantalla	Ej: MSG "Hola Mundo !!"
50 - LDB [mem]	La instrucción carga en AX el contenido de memoria almacenado en [mem] + BX	ej: Digamos que BX=10 ; LDB 1A carga el contenido de 1C en AX
51 - STB [mem]	guarda el contenido de AX en la dirección [mem1 + BX]	ej: Digamos que BX=101 ; STB 3A guarda AX en 3F
55 - LDF [mem]	Carga en BX y AX un numero de 32 bits (IEEE) que esta almacenado en la dir [mem] y mem+1. En BX quedan los digitos mas Significativos	Ej: Supongamos que el número 01000010110010001000000000000000 está cargado en memoria así: 02A 0100001011001000 (Los dígitos mas significativos) 02B 1000000000000000 (Los dígitos menos significativos) Un LDF 2A dejaría el siguiente resultado: BX: 0100001011001000 AX: 1000000000000000 Nota: Para pedirle al usuario o mostrar estos numeros IEEE 754 en pantalla, usar puerto 1, con las instrucciones IN AX,1 y OUT 1,AX
56 - STF [mem]	Guarda en [mem] y mem+1 el contenido de BX y AX	Ej: siendo AX y BX = al ejemplo anterior, un STF 2A deja la memoria como el ejemplo anterior.
60 - ADDF [mem]	Suma números de 32 bits: En BX y AX queda el resultado de la suma de estos mas el contenido de [mem] y mem+1	Estos numeros IEEE 754 son numeros que pueden ser de punto flotante, o enteros desde -2147483647 hasta 2147483647, si en cualquier operación de estas aritméticas, se sobrepasa este valor, se activa el Overflow flag.

⁷⁹ Simuproc, 2013

Tabla 20. Continuación - Instrucciones Soportadas por Simuproc 1.4.3⁸⁰

61 - SUBF [mem]	Resta el numero de 32 bits: BX y AX = BX y AX - [mem]y mem+1	Puedes utilizar esta instrucción como un CMP para numeros de 32 bits.
62 - MULF [mem]	Multiplicación: BX y AX = BX y AX * [mem]y mem+1	Si el resultado es > 2147483647, Resultado = 2147483647 y Overflow Flag = 1
63 - DIVF [mem]	Division	BX y AX = BX y AX / [mem]y mem+1 , en CX queda el residuo de la division en entero de 16 bits
64 - ITOF	Conversión de Entero a Real	Convierte un número entero (16bits) almacenado en AX al mismo numero pero representado en Real IEEE754 (32bits), el Resultado de la conversión queda en BX (bits mas significativos) y AX. Los registros de control cambian de acuerdo al numero convertido: "Z" si el numero es cero, "N" si el numero es negativo.
65 - FTOI	Conversión de Real a Entero	Convierte un número Real(32bits) a su equivalente en entero BX y AX en un entero (16bits), el Resultado queda en AX. Los registros de control cambian de acuerdo al numero convertido: "Z" si el numero es cero, "N" si el numero es negativo, "O" si el numero real es mayor de 65535.
80 - IN registro,puerto	Lleva al Registro el valor retornado por el puerto especificado.	Ej: IN AX,8 ;lleva a AX el valor retornado por el puerto 8 (Reloj: los segundos del sistema).
81 - OUT puerto,registro	Escribe en el puerto especificado, el valor del registro.	
90 - NOP	Esta operación no hace nada.	Útil para cuando se modifica la memoria para parchar código y desactivar instrucciones.
99 - HLT	Terminar Programa	Todo Programa lleva esta instrucción para indicarle al simulador que el programa ha terminado su ejecución.

⁸⁰ Simuproc, 2013

Programación con Simuproc

Para programar con Simuproc, se debe descargar la última versión Simuproc 1.4.3.0 de Febrero de 2004 desde (<https://sites.google.com/site/simuproc/home>), se instala dando doble “clic” sobre el instalador (.EXE), para versiones Windows 7 y 8 antes de ejecutar dar “clic” derecho sobre el programa instalador, seleccionar la opción “propiedades” en la ventana emergente, pestaña “compatibilidad” seleccionar ejecutar con compatibilidad Windows XP SP3 (32 bits) y seleccionar en nivel de privilegio “Ejecutar este programa como administrador”, aplicar y aceptar. Ahora nuevamente ejecutar la instalación del programa y seguir los pasos indicados hasta el final.

Como primer paso para desarrollar un programa en Simuproc se debe tener un problema a solucionar, generar el pseudocódigo o algoritmo, con la tabla o set de instrucciones a la mano, abrir Simuproc, clic en el icono “Editor” y en el editor ubicado a la derecha (Editor 2) comenzar a escribir el programa en ensamblador, después de finalizar la escritura del programa y de su depuración, se procede a dar “clic” en una flecha verde en las opciones del “Editor 2” para enviar el programa al “Editor 1” que representa la memoria principal en edición, se puede observar las posiciones de memoria y los nemotécnicos (código assembler), finalmente para enviar el programa a memoria principal del simulador en la pantalla principal, se presiona el botón “Enviar a Memoria”, si el programa no presenta errores de sintaxis esta transferencia ocurre sin novedad, en caso de tener algún error de sintaxis, se debe corregir antes de proceder nuevamente con la opción “Enviar a Memoria”, para culminar es conveniente desde un comienzo estar guardando el programa realizado y su simulación, en el “Editor 2” el programa se debe guardar como un .ASM y en la pantalla principal se puede guardar como un archivo .SMP (simulación), este último al abrirse directamente ejecuta Simuproc.

Ejemplo: El siguiente ejemplo ilustra un archivo .ASM en el cual se realiza las operaciones aritmética básicas, con opciones de menú para ingresar al programa y opciones para seleccionar la operación a realizar, este ejemplo es un buen punto de partida para comenzar a trabajar con este software y comprender no solo el funcionamiento del software, también el funcionamiento del microprocesador internamente y su interacción con los periféricos.

```
#SimuProc 1.4.3.0
MSG ESTE PROGRAMA REALIZA LAS CUATRO
MSG OPERACIONES ARITMETICAS
MSG 1 INTRODUCIR VALORES
MSG 2 SALIR DEL PROGRAMA
LDT ELIJA UNA OPCIÓN      ;RECIBE UN VALOR EN AX
```

```

        CMP 102 ; COMPARA AX CON LA POSICIÓN 102
        JMA 110 ; SI ES MAYOR SALTA A LA POSICIÓN INDICADA
        JEQ 112 ;SI ES IGUAL SALTA A LA POSICIÓN INDICADA
        CMP 101 ;COMPARA AX CON LA POSICIÓN INDICADA
        JEQ 114 ;SI ES IGUAL SALTA A LA POSICIÓN INDICADA
        JME 0 ;SI AX ES MENOR SALTA A LA POSICIÓN INDICADA
#100      ;POSICIÓN DE MEMORIA #100
          0   ; ALMACENAN CONSTANTES
          1
          10
          11
          100
          101
#110      ;POSICIÓN DE MEMORIA #110 SUBRUTINA MENSAJES
MSG NUMERO GRANDE
JMP 0
MSG ADIOS
HLT
LDT PRIMER NUMERO
STA 50
LDT SEGUNDO NUMERO
STA 51
MSG
MSG MENU SECUNDARIO
MSG QUE DESEA HACER
MSG 1 SUMAR
MSG 2 RESTAR
MSG 3 MULTIPLICAR
MSG 4 DIVIDIR
MSG 5 MENU PRINCIPAL
MSG MAYOR QUE 5 SALDRÁ DEL PROGRAMA
MSG
MSG SU OPCION ES
LDT; INTRODUCE UN VALOR A AX
CMP 105
JMA 112 ;SI ES MAYOR
JEQ 0 ;SI ES IGUAL SALTAR A
CMP 104
JEQ 80 ;SI ES IGUAL SALTAR A
CMP 103
JEQ 150 ;SI ES IGUAL SALTAR A
CMP 102
JEQ 140 ;SI ES IGUAL SALTAR A
CMP 101
JEQ 130 ;SI ES IGUAL SALTAR A
JME 118 ;SI ES MENOR SALTAR A
#80      ;POSICIÓN DE MEMORIA #80 SUBRUTINA DIVIDIR
LDA 50
DIV 51
EAP LA DIVISION ES
JMP 118
HLT
#150      ;POSICIÓN DE MEMORIA #150 SUBRUTINA MULTIPLICAR
LDA 50
MUL 51
EAP LA MULTIPLICACIÓN ES
JMP 118
HLT
#140      ;POSICIÓN DE MEMORIA #140 SUBRUTINA RESTAR
LDA 50
SUB 51
EAP LA DIFERENCIA ES
    
```

```
JMP    118
HLT
#130
LDA    50
ADD    51
EAP    LA SUMATORIA ES
JMP    118
HLT          ;POSICIÓN DE MEMORIA #130 SUBRUTINA SUMAR
              ;FIN DEL PROGRAMA
```

Se invita al lector a copiar el presente programa en el “Editor 2” y seguir su ejecución en velocidad “mínima” y junto con el set de instrucciones comprender el funcionamiento de las instrucciones, simulador y micro para futuras prácticas.

MASM

MASM (Microsoft Macro Assembler), es un ensamblador comercial (considerado como ensamblador de alto nivel) para microprocesadores x86 de Intel, inicialmente creado para el desarrollo y trabajo con MS-DOS fue muy popular pero en los años 1990 pierde mercado puesto que aparecen versiones alternativas de ensambladores como el TASM (Borland), el A86 y el NASM, a finales de 1990 MASM recupera la cuota de mercado al distribuir gratuitamente el MASM y al distribuir el paquete MASM32 (32 bits) para aplicaciones Windows. La versión separada de MASM se distribuye desde <http://www.masm32.com> (en su versión actual 11), Microsoft distribuye versiones actualizadas de la versión MASM incluidas dentro de paquetes como Visual C++ 6.0 Professional (MASM 6.15), Visual C++ .NET (MASM 7), Visual C++ 2005 (MASM 8 ensambla código x64). En las versiones nuevas de Visual las versiones de ensamblador a 32 bits se encuentran en el archivo “ml.exe” y para 64 bits el archivo “ml64.exe” incluido en el directorio “BIN” de Visual C++.

MASM sigue siendo soportado por Microsoft puesto que es utilizado para generar código propio de Microsoft, por lo que agrega instrucciones y compatibilidad para los nuevos microprocesadores y tecnología 64 bits. MASM es utilizado en la enseñanza e instrucción sobre microprocesadores y por desarrolladores de periféricos compatibles con Windows en el desarrollo del software para los controladores de dicho hardware.

Primeros pasos con MASM y MASM32

En primer lugar se requiere descargar la versión de MASM y realizar la instalación y configuración adecuada para poder trabajar (editar, compilar, depurar y ejecutar). Es posible descargar varias versiones de MASM, y configurarlas para adecuarlas al desarrollo de programas.

MASM: Para versiones 5 o 6 se tiene que ya sea adecuar una carpeta en la raíz del disco principal manualmente donde se encuentra el archivo MASM y LINK, también se tiene la opción de instalar el programa el cual genera una carpeta en la raíz del disco duro con el nombre por defecto “MASM” más la versión del paquete, en este tipo de instalación se tiene un directorio interno llamado “BIN” en el que se aloja los archivos MASM y LINK necesarios para compilar y enlazar (generar el código ejecutable) a partir del código fuente en .ASM.

Usualmente estas no poseen editor y se debe ejecutar los comandos (MASM, LINK) desde el “prompt” del símbolo de sistema, por lo cual es conveniente modificar la variable “**PATH**” para ejecutar los comandos MASM y LINK adecuadamente, para esto se debe ir a “**Inicio/Equipo**” “clic” derecho “*propiedades*”, “*Configuración avanzada del sistema*”, con lo que se abre una nueva ventana, seleccionar la pestaña “*Variables de entorno*”, seleccionar variable “**PATH/Editar**” y colocar la ruta en donde se encuentra la aplicación MASM o LINK, por ejemplo, “C:\MASM” o “C:\masm_614\BIN”, si ya se tiene otras rutas escritas, NO cambiar nada solo colocar “;” y la ruta que se quiere agregar, con esto desde el “prompt” del símbolo de sistema solo es necesario ejecutar “MASM” o “LINK” sin importar dentro de que directorio o disco nos encontremos. Para finalizar Aceptar los cambios.

MASM32: En su versión 11 al momento de escribir este texto, tiene la ventaja de ser un entorno de desarrollo (SDK), por tanto posee todas las herramientas para editar, compilar, enlazar, depurar, entre otras, en este caso no es necesario editar variables “**PATH**” puesto que es un entorno amigable al usuario (grafico) y no de línea de comandos. Esta versión está soportada para 32 bits e incluso para instrucciones ejecutadas bajo 64 bits, lo cual lo convierte en una herramienta de desarrollo para aplicaciones (.dll, .com, .exe, etc) y controladores de dispositivos. Se debe considerar que al desarrollar un programa en MASM32 se debe seguir una estructura propia de este software el cual hace llamado a bibliotecas y archivos.

Ejemplo: Estructura que hace llamado a biblioteca y archivos⁸¹

```
include \masm32\include\windows.inc
include \masm32\include\user32.inc
include \masm32\include\kernel32.inc
include \masm32\include\gdi32.inc
```

⁸¹ MASM, 2013

```
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\gdi32.lib
```

Programar con MASM

Para poder programar con MASM son varios los elementos necesarios, en primer lugar se debe tener editor de texto para generar el archivo fuente, como editor de texto se tiene el editor de MS-DOS (Símbolo de sistema), al cual se accede por “*Inicio/Todos los programas/Símbolo de sistema*” o “símbolo de Windows + R” para abrir la ventana de ejecución, donde se digita “cmd”, lo que abre la ventana de Símbolo de Sistema, finalmente se digita “edit” para abrir el editor de texto.

Teniendo el código fuente digitado y guardado como .ASM, por medio de un compilador como MASM se convierte a formato .OBJ (Objeto), esto se hace mediante el comando en el prompt del símbolo de sistema “**MASM nombredelprograma; [Enter]**” con lo que se obtiene un archivo “nombredelprograma.OBJ” que contiene el código máquina. En segundo lugar se requiere un enlazador (Linker) el cual genera el archivo ejecutable .EXE o .COM, mediante el comando en el prompt del símbolo de sistema “**LINK nombredelprograma; [Enter]**”, obteniendo el archivo “**nombredelprograma.EXE**” finalmente se utiliza un depurador (Debugger) con el cual se verifica, depura o corrige el programa ejecutable.

Ejemplo: Programa Hola mundo escrito para MASM

```
;Nombre del Programa      : HOLA.ASM
;Fecha de creación        : Julio 2013
;Autor                   : Ing. Uriel Villamil
;Objetivo                : Desplegar mensaje 'Hola mundo'
;
;COMANDOS DE PARA ENSAMBLAR    :MASM /SAMPLES/HOLA.ASM;
;COMANDO DE ENLACE-LINK       :MASM /SAMPLES/HOLA.ASM;
;COMANDO PARA EJECUTAR        :HOLA [Enter]
;
;DEFINICION DEL MODELO DE MEMORIA
;
    .MODEL SMALL           ;Modelo de memoria
    .CODE                  ;Area de código
;Programa principal etiquetado como "Inicio"
Inicio: mov    Ax, @Data      ;Inicializa AX con la dirección de memoria @Data
        mov    Ds,Ax
        mov    Dx,Offset Mensaje   ;Dirección del mensaje a desplegar
        mov    Ah,9
        int    21h                 ;Despliegue de mensaje a travez de MS-DOS int21
;
;DEFINICION DEL SEGMENTO DE DATOS
    .DATA
MensajeDB    'Hola mundo!.$' ;mensaje (cadena) a desplegar
;
```

```
:DEFINICION DE SEGMENTO DE PILA
    .STACK
;
END   Inicio           ;Fin del programa
```

Como se puede observar el código es muy similar en sus instrucciones al realizado en el estudio de “Debugger”, la característica propia de este editor MASM radica en que debe digitarse en un archivo que debe ser guardado como .ASM, Se tiene un formato interno compuesto de etiquetas/variables/constante, nombre nemónico/directiva, operando y comentario. También se tiene un formato para organizar internamente el archivo, en primer lugar los comentarios para establecer características del archivo como nombre, autor, fecha de creación, objetivo del programa, variables de entrada o salida, resumen de comandos etc, luego continua con una división en donde se establece las directivas para definir el tipo de memoria y la forma como se accede a las instrucciones y datos **“.MODEL SMALL”**, luego se define con la directiva **“.CODE”** el lugar a partir del cual se deben colocar las instrucciones (código a ejecutar), sigue **“.DATA”** lugar en el que se puede definir constantes en el caso del programa anterior **“Mensaje DB ‘Hola mundo!.\$”**, en donde la cadena de caracteres debe culminar con el símbolo **“\$”**, el cual le indica al ensamblador que en este punto finaliza la cadena de caracteres ASCII y evita que el programa siga leyendo datos hasta encontrar **“\$”**. Continua con la directiva **“.STACK”** el cual reserva espacios de memoria para la operación de la PILA. Finalmente se tiene la directiva **“END”** que le dice al ensamblador que el programa ha finalizado.

Las anteriores directivas **“.CODE, .DATA y .STACK”** son expansión de instrucciones más simples, en donde **“.CODE”** es la contracción de la línea inicial que define el bloque de código **“CODE SEGMENT”**, la línea **“ASSUME”** que define como direccionar mediante los registros de segmento (ejemplo CS:CODE, SS:STACK), después sigue el código o instrucciones a ejecutar y termina con la línea **“CODE ENDS”**, en el caso de la directiva **“.STACK”** es la contracción de la línea inicial del bloque de PILA **“STACK SEGMENT”** y la línea que delimita el final del bloque de PILA en el código fuente **“STACK ENDS”**

Programar con MASM32

Este software es un entorno de desarrollo (SDK), su instalación se hace por medio de un archivo ejecutable, que crea un directorio por defecto en la raíz del disco principal, la ventaja de este compilador es que tiene incorporado un editor **“qeditor”** el cual permite editar el código fuente para ser guardado en .ASM, compilarlo (Assemble ASM File), enlazarlo (Link OBJ File), Ensamblarlo y enlazarlo (Assemble & Link) o Construir todo el proyecto (Build All), opciones que

Lección 15: Ejemplos de aplicación

Como herramientas de programación en ensamblador tenemos el Debugger, MASM y el Simuproc. Es recomendable comenzar con Debugger digitando y probando los códigos de programa que aquí se exponen. Luego es recomendable tomar la documentación suministrada sobre las especificaciones del programa Simuproc hacer una lectura analítica y comenzar por introducir instrucciones simples y notar su funcionamiento, más adelante podrá realizar pequeños programas adecuándolos desde Debugger a Simuproc y hacer la ejecución del mismo, hasta llegar al punto de producir sus propios programas antes de pasar a trabajar con MASM y MASM32.

Es importante que los estudiantes cada vez que generen un programa en Debugger o MASM, sean conscientes que pueden al digitar o probar algún programa quedar en un ciclo infinito o bloquear el sistema, por lo que se aconseja no tener abierto documentos o programas con información relevante susceptible de ser perdida en el momento de realizar los ejercicios, es conveniente destinar un equipo y espacio de tiempo para realizar los programas que se exponen a continuación, cuanto sucedan eventos de iteración infinita o bloqueo reiniciar el equipo.

Si se utiliza SimuProc dado que es un emulador aunque también puede generar algún “bug” la solución es menos drástica, en este caso cerrando el programa y volviéndolo a ejecutar.

Controlar condiciones, iteraciones y bifurcaciones

Las estructuras básicas de programación utilizan las iteraciones, condiciones y bifurcaciones proporcionándole al sistema cierto grado de inteligencia y decisión:

El siguiente programa despliega 10 veces una cadena de caracteres, como características especiales se utiliza la INT 21H y su función 9, exige que el registro par DS:DX contenga la dirección inicial de la cadena a desplegar, la cadena debe terminar con el signo "\$", la instrucción para control de iteración es LOOP que salta a la localidad que repite la INT 21H y decrementa a CX hasta que llegue a cero.

Figura 65. Ejemplo de iteración utilizando la instrucción “LOOP”⁸³

```
C:\Users\TURION>DEBUG
-A
17A3:0100 JMP 135
17A3:0102
-E 102 'MICROPROCESADORES Y MICROCONTROLADORES 2009' 0D 0A '$'
-A 135
17A3:0135 MOU CX, 000A
17A3:0138 MOU DX, 0102
17A3:013B MOU AH, 9
17A3:013D INT 21
17A3:013F LOOP 013D
17A3:0141 INT 20
17A3:0143
-G141
MICROPROCESADORES Y MICROCONTROLADORES 2009

AX=0924 BX=0000 CX=0000 DX=0102 SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A3 ES=17A3 SS=17A3 CS=17A3 IP=0141 NV UP EI PL NZ NA PO NC
17A3:0141 CD20           INT    20
```

Observe que la cadena se encuentra entre comillas simples, seguida por un espacio, el valor hexadecimal 0DH (retorno de carro), otro espacio, el valor hexadecimal 0AH (código de línea nueva), otro espacio y finalmente el símbolo “\$” (terminación de cadena para ensamblador).

Figura 66. Iteración con “DEC BX” y condición de salto “JNZ”⁸⁴

```
C:\Users\TURION>DEBUG
-A
17A3:0100 JMP 135
17A3:0102
-E 102 'MICROPROCESADORES Y MICROCONTROLADORES 2009' 0D 0A '$'
-A 135
17A3:0135 MOU BX, 0A
17A3:0138 MOU DX, 102
17A3:013B MOU AH, 9
17A3:013D INT 21
17A3:013F DEC BX
17A3:0140 JNZ 13D
17A3:0142 INT 20
17A3:0144
-G142
MICROPROCESADORES Y MICROCONTROLADORES 2009

AX=0924 BX=0000 CX=0000 DX=0102 SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A3 ES=17A3 SS=17A3 CS=17A3 IP=0142 NV UP EI PL ZR NA PE NC
17A3:0142 CD20           INT    20
```

Variante del programa 1: Una variante del anterior programa utiliza dos instrucciones para controlar el bucle, la primera sirve para usar el registro BX como una variable, que se carga con la cantidad de veces que se quiere desplegar la cadena e ir decrementando de uno en uno, utilizando la instrucción “DEC BX”,

⁸³ ROJAS, 1997

⁸⁴ ROJAS, 1997

para poder bifurcar en función del resultado de la bandera “NZ”, la instrucción para generar la bifurcación es “JNZ” (Jump if NotZero) si no es igual a cero despliega nuevamente la cadena, si es igual a cero termina el programa con la interrupción “INT 20”.

Figura 67. Iteración utilizando el contador “CX” y el salto condicional “JCXZ”⁸⁵

```
C:\Users\TURION>DEBUG
-A
1?A3:0100 JMP 135
1?A3:0102
-E 102 'MICROPROCESADORES Y MICROCONTROLADORES 2009' 0D 0A '$'
-A 135
1?A3:0135 MOU DX, 0102
1?A3:0138 MOU CX, 000A
1?A3:013B MOU AH, 09
1?A3:013D INT 21
1?A3:013E DEC CX
1?A3:0140 JCXZ 0144
1?A3:0142 JMP 013D
1?A3:0144 INT 20
1?A3:0146
-G144
MICROPROCESADORES Y MICROCONTROLADORES 2009
AX=0924 BX=0000 CX=0000 DX=0102 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1?A3 ES=1?A3 SS=1?A3 CS=1?A3 IP=0144 NU UP EI PL ZR NA PE NC
1?A3:0144 CD20    INT     20
```

Variante del programa 2: Esta variación hace uso del registro CX como contador, también utiliza la instrucción JCXZ (Jump if CX = 0) que controla la condición de salto, con esto se ha observado que existen varias formas en assembler de llegar al mismo resultado o hacer la misma función.

El ejercicio y práctica constante con la programación en lenguaje ensamblador permite con el tiempo adquirir habilidades y competencias, con las que el programador puede determinar las instrucciones, su disposición y la estructura general del programa buscando mejorar la velocidad de ejecución y estructura del código.

Programa utilizando interrupciones

Este programa utiliza la función 1 de la interrupción BIOS, la cual cambia la forma del cursor, la función a utilizar se carga en el registro AH y el código del cursor en CX.

⁸⁵ ROJAS, 1997

Figura 68. Uso de las Interrupciones⁸⁶

```
C:\Users\TURION>DEBUG
-A
17A3:0100 MOU AH, 1
17A3:0102 MOU CX, 7
17A3:0105 INT 10
17A3:0107 INT 20
17A3:0109
-G107

AX=0000 BX=0000 CX=0007 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A3 ES=17A3 SS=17A3 CS=17A3 IP=0107 NU UP EI PL NZ NA PO NC
17A3:0107 CD20      INT     20
-
```

Para regresar el cursor a su modalidad predefinida, usualmente una raya al piso, se cambia la línea que dice “MOV CX, 0607”, valido para las tarjetas adaptadoras compatibles con color. Al salir de Debugger (Q), el curso regresa a su estado normal.

El siguiente programa utiliza la interrupción 21H de DOS, empleando dos funciones, la primera “AH=1” lee el teclado y la segunda “AH=2” escribe en la pantalla, el programa lee caracteres del teclado hasta encontrar <CR> (retorno de carro).

Figura 69. Uso de la interrupción 21H de DOS

```
C:\Users\TURION>DEBUG
-A
17A3:0100 MOU AH, 1
17A3:0102 INT 21
17A3:0104 CMP AL, 0D
17A3:0106 JNE 100
17A3:0108 MOV AH, 2
17A3:010A MOU DL, AL
17A3:010C INT 21
17A3:010E INT 20
17A3:0110
-G110
BIENVENIDOS AL CURSO DE MICROPROCESADORES Y MICROCONTROLADORES
Program terminated normally
-R
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=17A3 ES=17A3 SS=17A3 CS=17A3 IP=0100 NU UP EI PL NZ NA PO NC
17A3:0100 B401      MOU     AH,01
```

El siguiente programa de ejemplo sirve para afianzar los temas expuestos, ensamblado, ejecución, guardar y cargar el archivo, desensamblar y ejecutarlo nuevamente, este programa permite ocultar o mostrar un archivo, después de ocultarlo tendrá que recurrir a otro medio para visualizarlo.

⁸⁶ ROJAS, 1997

Figura 70. Programa que permite ocultar o mostrar un archivo⁸⁷.

```
C:\Users\TURION>DEBUG
-A 100
17A3:0100 JMP 138
17A3:0102
-E 102 'FUNCION A REALIZAR <E> O <M> : $'
-E 122 'NOMBRE DEL ARCHIVO : $'
-E 250 D 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-E 246 0D 0A '$'
-A 138
17A3:0138 MOU AH, 0
17A3:013A MOU AL, 2
17A3:013C INT 10
17A3:013E MOU DX, 102
17A3:0141 MOU AH, 9
17A3:0143 INT 21
17A3:0145 MOU AH, 0
17A3:0147 INT 16
17A3:0149 MOU [249], AL
17A3:014C MOU DX, 246
17A3:014F MOU AH, 9
17A3:0151 INT 21
17A3:0153 MOU DX, 122
17A3:0156 MOU AH, 9
17A3:0158 INT 21
17A3:015A MOU AH, 0A
17A3:015C MOU DX, 250
17A3:015F INT 21
17A3:0161 MOU AL, [249]
17A3:0164 CMP AL, 45
17A3:0166 JNZ 16D
17A3:0168 MOU CX, 2
17A3:016B JMP 170
17A3:016D MOU CX, 0
17A3:0170 MOU DI, 252
17A3:0173 MOU AL, 0D
17A3:0175 SCASB
17A3:0176 JNZ 175
17A3:0178 MOVE BYTE PTR [DI-01], 00
17A3:017C MOU AH, 43
17A3:017E MOU DX, 252
17A3:0181 MOU AL, 01
17A3:0183 INT 21
17A3:0185 INT 20
17A3:0187
-H 0187 0100
0287 0087
-N C:MPMC01.COM
-RCX
CX 0100
:0187
-W
Writing 00187 bytes
-N C:\Users\TURION\MPMC01.COM
-L
-U 100 186
```

El anterior código está basado en un algoritmo que previamente se ha diseñado, como toda fase de diseño de un proyecto hay que tener primero un conocimiento absoluto de las instrucciones y modo de ejecución, posteriormente se emplea los conocimiento en programación para diseñar el algoritmo correspondiente para finalmente convertirlo en el lenguaje que se utiliza para lograr el objetivo del proyecto.

Para este caso presentamos primero el código para que el estudiante se acerque más a las instrucciones y se familiarice con ellas, de esta forma se puede comenzar un análisis de cada instrucción, con respecto al algoritmo siguiendo pautas de ingeniería inversa. Este es el algoritmo en pseudocódigo.

⁸⁷ ROJAS, 1997

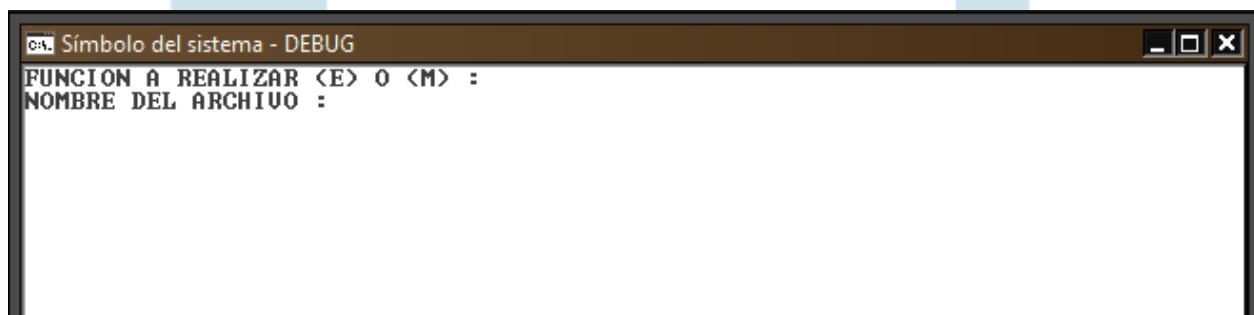
INICIA:

PONER EN BLANCO LA PANTALLA
PREGUNTAR POR LA FUNCION A REALIZAR
PREGUNTAR POR EL NOMBRE DEL ARCHIVO
IF FUNCION ES 'E'
 THEN OCULTAR ARCHIVO
ELSE
 MOSTRAR EL ARCHIVO
END IF

TERMINA

En el caso del Windows, como alternativa para visualizar archivos ocultos se puede recurrir al explorador de Windows, menú /Herramientas/Opciones de carpeta, pestaña VER y seleccionar “mostrar todos los archivos y carpetas ocultas”, con esto podrá ver el archivo oculto y con clic derecho/Propiedades deseleccione la opción “oculto”.

Figura 71. Interfaz del programa anterior, ventana DOS⁸⁸.



⁸⁸ ROJAS, 1997

Actividades de Autoevaluación de la UNIDAD

1. Realizar una investigación mediante consulta en internet sobre los últimos microprocesadores para servidores, equipos de escritorio y portátiles.
2. Investiga cual es el ranking de las mejores supercomputadoras.
3. Consulta las características de los últimos procesadores ofrecidos por las empresas INTEL y AMD.
4. Disponer de un equipo de cómputo y del tiempo necesario para probar cada programa propuesto en la lección “Ejemplos de aplicación” en el Capítulo 3 utilizando el intérprete Debugger, no se debe estar trabajando en otros programas o tener información importante susceptible de ser perdida, al utilizar Debugger pueden generarse ciclos infinitos o bloquear el sistema. Si sucede un bloqueo reinicie la máquina. La utilización de Debugger dentro de parámetros controlados como los ejercicios propuestos no implican perdida de información, es recomendable consultar la bibliografía propuesta si quiere profundizar en estos temas.
5. Descargar el emulador “SimuProc” e instalarlo en el equipo, con compatibilidad “Windows XP”, descargar la documentación respecto a SimuProc, realizar lectura del documento.
6. Proceder a ingresar de forma manual cada instrucción estudiando el comportamiento y las microoperaciones involucradas.
7. SimuProc tiene una serie de programas de demostración los cuales puede abrir y ejecutar, es conveniente que los estudie paso a paso para entender su funcionamiento.
8. Realizar el algoritmo, diagrama de flujo e introducir en el intérprete Debugger, guardar, cargar y ejecutar programas sencillos que involucren movimiento de registros, operaciones aritméticas, comparaciones y bifurcaciones. por ejemplo:
 - a. Área de un cuadrado, triangulo, círculo o alguna superficie geométrica.
 - b. Serie de Fibonacci.
 - c. Factorial de un número.
9. Realizar el algoritmo, diagrama de flujo, código fuente, compilar y ejecutar programas sencillos en SimuProc que involucren movimiento de registros, operaciones aritméticas, comparaciones y bifurcaciones. por ejemplo:
 - a. Área de un cuadrado, triangulo, círculo o alguna superficie geométrica.
 - b. Serie de Fibonacci.
 - c. Factorial de un número.
10. Comparar las experiencias, sacar conclusiones y debatirlas en el grupo.

Fuentes Documentales de la Unidad 1

DOCUMENTOS IMPRESOS

Stallings, William. "Organización y Arquitectura de Computadores". (5^a edición). Editorial Prentice-Hall. Madrid, 2000.

González, Vásquez José Adolfo. (1992). Introducción a los microcontroladores: hardware, software y aplicaciones. Editorial McGraw-Hill.

Rojas, Ponce Alberto. (1997). "Ensamblador Básico". Editorial Computec. AlfaOmega Santafé de Bogotá.

Uruñuela, José M^a. "Microprocesadores: Programación e Interconexión". (2^a edición). Editorial Mc Graw Hill. España, 1995.

Tokheim, Roger. "Fundamentos de los Microprocesadores". (2^a edición). Editorial Mc Graw Hill. México, 1985.

Vesga, Ferreira Juan Carlos. (2007). Microcontroladores Motorola – Freescale: Programación, familias y sus distintas aplicaciones en la industria.

CEKIT. (2002). Curso Práctico de Microcontroladores: Teoría, Programación, Diseño, Prácticas Proyectos completos. Editorial Cekit. Pereira-Colombia.

Ureña, López Alfonso, Sanchez, Solano Antonio Miguel, Martín, Valdivia María Teresa & MANTAS, Ruiz Jose Miguel. (1999). Fundamentos de informática. Editorial Alfaomega & ra-ma. Santafé de Bogotá.

Barry B. B. (1995). Los microprocesadores Intel 8086/8088, 80186, 80286, 80386 y 80486, Arquitectura, programación e interfaces. (3^a edición). Prentice Hall Hispanoamerica, S.A.

Téllez, Acuña Freddy Reynaldo. (2007). Módulo de Microprocesadores y Microcontroladores. UNAD.

Angulo. (n.d). Microcontroladores PIC, la solución en un chip. Sección 5.1

Valdivia, Miranda Carlos. (n. d). Arquitectura de equipos y sistemas informáticos.
Editorial Paraninfo.

Angulo, Usategui José María. (n. d). Microcontroladores PIC. Diseño práctico de aplicaciones.

DIRECCIONES DE SITIOS WEB

Simuproc, extraído el 23 de Julio de 2013 desde

<https://sites.google.com/site/simuproc/home>

MASM32, extraído el 23 de Julio de 2013 desde

<http://www.masm32.com>

Dispositivos lógicos microprogramables, extraído el 11 de Julio de 2011 desde

<http://perso.wanadoo.es/pictob/indicemicroprg.htm>

UNIDAD 2 MICROCONTROLADORES

CAPITULO 4: INTRODUCCIÓN A LOS MICROCONTROLADORES

Con la aparición de los microprocesadores y las necesidades de control para distintos dispositivos tanto industriales (instrumentación, automatización, telemetría, etc.), comerciales (automóviles, periféricos, juguetes) y domésticos (electrodomésticos, audio, video), aparece la necesidad tecnológica de incorporar en un solo “chip” la estructura básica de un sistema de cómputo, este “microcomputador” debería contar con tres unidades funcionales de cualquier equipo de cómputo: CPU, memoria y unidades I/O (Entrada/salida). Es así como se da origen a los microcontroladores, pequeños dispositivos producto de la microelectrónica generalmente de arquitectura cerrada que fusionan en una misma pastilla de silicio las tres unidades funcionales de una computadora, aplicados a situaciones específicas de control y capaces de incorporar unidades adicionales que amplían su capacidad de interacción con el medio incluso llegando a comportarse como sistemas abiertos (el caso de los microprocesadores).

Lección 16: Generalidades de los microcontroladores

Esta lección establece los fundamentos básicos en la tecnología de microcontroladores, retomando aspectos de la tecnología de microprocesadores, teniendo presente que el microcontrolador es un micro-computador o un micro sistema de cómputo, que contiene un microprocesador. Se establece los orígenes de la tecnología de microcontroladores, su estructura, clasificación, aplicación,

mercado, para luego establecer similitudes y diferencias entre los microprocesadores y microcontroladores, con lo que se plantean los periféricos utilizados, sistemas de desarrollo y las consideraciones de selección y aplicación del dispositivo.

Origen de los microcontroladores

En 1969, ingenieros de la compañía japonesa BUSICOM, buscan soluciones para fabricar con pocos componentes sus dispositivos (calculadoras), esta proposición se le hizo a INTEL quien en un proyecto dirigido por Marcian Hoff y apoyado por Federico Faggin, logró fabricar un bloque integrado denominado "microprocesador" adquiriendo los derechos de la compañía BUSICOM y entregando al mercado en 1971 el primer microprocesador el 4004 de 4 bits. Como ya se ha mencionado le siguieron el i8008, i8080, el Motorola 6800, Z80, i8085.

En 1976 aparece en el mercado un nuevo dispositivo que incorpora una CPU, memoria RAM - ROM y puertos de I/O, este dispositivo es llamado "microcontrolador" que son microcomputadoras en un solo chip, dos de los más representativos y primeros microcontroladores fueron:

- Intel 8048, con arquitectura Harvard modificada con programa ROM en el mismo chip, RAM de 64 a 256 bytes e interfaz I/O (entrada/salida).
- Motorola 6805R2.

En la década de los 80's comienza la ruptura de desarrollo y evolución tecnológico entre microprocesadores y microcontroladores. Los microprocesadores han evolucionado buscando la solución al manejo de grandes volúmenes de información, mientras los microcontroladores incorporan unidades funcionales con capacidades superiores de interacción con el medio físico en tiempo real, un mejor desempeño y robustez en aplicaciones industriales.

En los años posteriores aparecen nuevos microcontroladores que son utilizados generalmente para controlar dispositivos periféricos de computadores y algunas aplicaciones de control particulares.

¿Qué es un microcontrolador?

Es un dispositivo programable con capacidad de ejecutar operaciones, tareas y procesos a gran velocidad, lo que permite su uso en aplicaciones en tiempo real, como sensores, sistemas remotos, automatismos, sistemas de control en máquinas y aplicaciones industriales.

En síntesis el microcontrolador es una pequeña computadora utilizada para aplicaciones puntuales, esto quiere decir que el microcontrolador debe incluir ciertas unidades fundamentales y comunes en cualquier computadora.

Los microcontroladores representan la gran mayoría de chips de computadoras vendidos en el mundo, de estos más del 50% son microcontroladores básicos y el restante son DSP o Procesador Digital de Señales, una variante de los microcontroladores con gran capacidad de procesamiento de señales, usualmente se encuentran en equipos de audio.

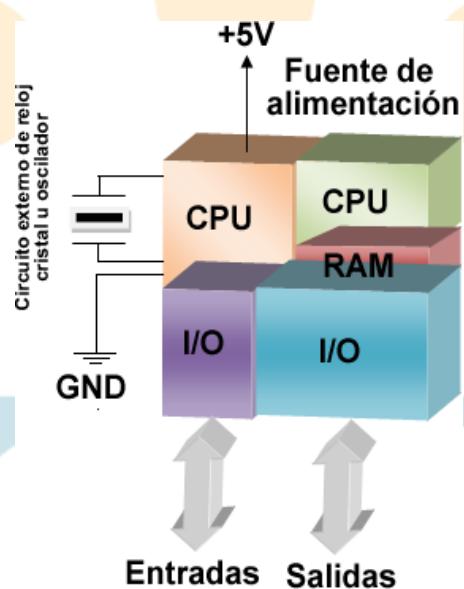
En general hay conciencia de la existencia de los procesadores por los computadores personales y la publicidad que se maneja, pero aunque pasa desapercibido, en nuestro entorno estamos rodeados de más microcontroladores que de microprocesadores, cada electrodoméstico moderno, equipos de audio, video, telefonía, entretenimiento, automóviles, equipo industrial, instrumentación, etc están compuestos de microcontroladores especializados para cada tarea.

Estructura de un microcontrolador

La estructura de un microcontrolador se compone de unidades fundamentales, similares a las unidades en una microcomputadora, estas unidades son:

- CPU, Unidad Central de proceso, los microcontroladores generalmente se basan en el núcleo de un microprocesador como por ejemplo el Intel 8080, Z80, Motorola 6800 entre otros.
- Memoria ROM y RAM, dentro del microcontrolador se construyen bloques de memoria necesaria para almacenar el programa, los datos y demás registros necesarios para implementar el proceso de control. Coexisten dos tipos de memoria:
 - ROM, es el sitio donde se almacena el programa (memoria de programa), consta de unos cuantos KBytes de memoria, suficientes para almacenar el programa en código maquina.
 - RAM, en ella se almacenan datos temporales (memoria de datos), usualmente es de poca capacidad, porque las aplicaciones de control, instrumentación y automatización no requieren grandes espacios de almacenamiento temporal.
- Puertos I/O, puertos de entrada / salida, son pines del microcontrolador destinados a comunicar el microcontrolador con el entorno, usualmente los pines pueden tener varias funciones las cuales se configuran por registros internos que varían entre familias de fabricantes y entre la gamma de la familia.

Figura 72. Estructura de microcontrolador⁸⁹



Clasificación de los microcontroladores

Los microcontroladores tienen una clasificación similar a la de los microprocesadores, es decir, se clasifican de acuerdo a la longitud de palabra desde los 4bits, 8bits, 16bits y los últimos que han salido al mercado son los poderosos de 32bits.

Figura 73. Microchip 32 bits⁹⁰



La razón del porque siguen en el mercado microcontroladores de 4 y 8 bits, es simple, todavía existen muchas aplicaciones que los implementan, no tiene

⁸⁹ CEKIT, 2002

⁹⁰ Extraído el 10 de Julio de 2009 desde <http://www.electronicosonline.com/noticias/images/uploads/PIC-32-Microchip.jpg>

sentido aplicar un microcontrolador potente como los de 16bits o 32 bits donde el control lo puede hacer uno de 8 bits mucho mas económico, de hecho, los microcontroladores de 8 bits dominan el mercado actual.

Figura 74. Motorola 16, 32 bits⁹¹

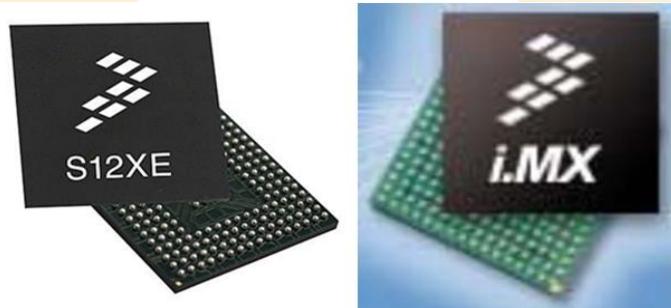


Figura 75. AVR-Atmel 32 bits⁹²



Además de la clasificación en bits, dentro de cada familia, existe una gran variedad de modelos que incorporan unidades funcionales como temporizadores (timer), conversores análogo-digitales (ADC), moduladores de ancho de pulso (PWM), puertos de comunicación I²C, USART, puertos serial síncronos (SSP), puertos seriales asíncronos (BSSP), puerto paralelo esclavo (PSP), control de motores, soporte de interfaz universal de comunicaciones (USB), soporte controlador de Ethernet, soporte controlador IRDA (comunicaciones infrarrojas) entre otros.

Las anteriores unidades funcionales se incorporan en distintos modelos o gammas dentro de la misma familia, llegando a decenas e incluso cientos, en

⁹¹ Extraído el 10 de Julio de 2009 desde http://img.directindustry.com/images_di/photo-q/microcontroller-193829.jpg

http://img.directindustry.com/images_di/photo-q/microprocessor-193830.jpg

⁹² Extraído el 10 de Julio de 2009 desde <http://www.anatronic.com/images/Noticias/An2583.jpg>

configuraciones de 8, 18, 20, 28 y 40 pines en DIP, SOIC, SSOP, TQFP, TSOP YJW, que son los empaques o presentaciones disponibles, para el caso de los utilizados en montajes de prueba y aprendizaje se utilizan del tipo DIP (Dual In-line Package).

Hablando de técnicas de fabricación, prácticamente todos los microcontroladores se fabrican utilizando tecnología CMOS (Complementary Metal Oxide Semiconductor), estas técnicas son mas económicas e inmunes a ruido.

Aplicación de los microcontroladores

La fabricación actual de microcontroladores supera en millones de unidades de un mismo modelo en una semana, por su utilización en multitud de aplicaciones y la capacidad que tienen de comunicación entre ellos, se implementan sistemas los cuales cada microcontrolador se encarga de una función específica y mantiene comunicación y coordinación de tareas con otros microcontroladores que atienden otras funciones mediante un microcontrolador mas potente o un microprocesador, este caso lo podemos apreciar en las computadoras personales en sus periféricos (controlados por microcontroladores) y CPU (microprocesador).

En la actualidad el mercado del microcontrolador se extiende muy rápidamente por la necesidad de incorporar estos chips en diversos productos buscando aumentar sus prestaciones, mejorar la fiabilidad, disminuir costo, consumo y tamaño.

Microcontroladores y mercado tecnológico

Generalmente a los ojos del común, se escucha hablar de microprocesadores acaparando toda la atención, lo cierto es que se venden más microcontroladores que microprocesadores. Mientras los microprocesadores evolucionan de 32 bits a 64 y múltiples núcleos dejando atrás modelos de 8, 16 y 32 bits y tecnologías consideradas obsoletas, en los microcontroladores el mercado no deja extinguir los primeros modelos de 4 y 8 bits y aceptan de igual manera los últimos de 32 bits.

El sector que más ha impulsado la producción de microcontroladores ha sido el sector automovilístico el cual promovió el desarrollo de microcontroladores de uso especial para adaptarse a las condiciones extremas de vibración, ruido, choques, etc. Conservando fiabilidad para evitar fallos que causen accidentes, estos chips serían adaptados para usos generales. Otros sectores como el de telecomunicaciones, informático, instrumentación e industrial también han favorecido el desarrollo y producción de nuevos microcontroladores capaces de

adaptarse al medio. Las ventas de chips están ubicadas en sectores bien definidos:

- Computadores y periféricos
- Aplicaciones de consumo como electrodomésticos, audio, video, entretenimiento, televisión, etc).
- En dispositivos de comunicación (telefonía, radio comunicaciones)
- Aplicaciones industriales en automatización, telemetría, sensores, etc.
- Industria de automotores.
- Aplicaciones militares.

Los recientes microcontroladores de 32 bits están siendo bien recibidos en el mercado, ocupando el interés de áreas como la de comunicaciones, procesos industriales, procesamiento de imágenes y control de dispositivos de almacenamiento masivo.

Microcontroladores Vs Microprocesadores

Tabla 21. *Microcontroladores vs Microprocesadores*⁹³

Microcontroladores	Microprocesadores
Los dispositivos genéricos son de 8bits, actualmente hay desarrollo de 16 y 32 bits	Tienen mayores longitudes de palabra (16, 32, 64bits)
Incorpora en una misma pastilla capacidad de memoria y manejo de puertos.	Necesitan chips externos para apoyar las funciones de transferencia y almacenamiento.
La memoria es limitada aunque algunos permiten incorporar KB externos.	Manejo de mayor capacidad de memoria
Tarjetas simples alojadas en espacios pequeños.	Gran volumen (físico) en su adecuación y funcionamiento.
Consumo de miliwatts o unos pocos watts es posible alimentarlo con baterías.	Consumo de potencia en cientos de watts
Dispositivos complejos de comunicación que permiten control en tiempo real.	Manejo de dispositivos periféricos informáticos.
Robustos e inmunes a ruidos industriales	Susceptibles a ruido industrial
Sistemas mínimos	Sistemas informáticos

⁹³ CEKIT, 2002

Módulos internos en los microcontroladores

Los módulos internos en los microcontroladores hacen referencia a unidades funcionales particulares, que destacan el micro para una aplicación específica entre ellas se tienen. Esta unidades funcionales pueden tener o no acceso directo al mundo exterior mediante pines o patillas del micro.

Timers: Son circuitos temporizadores vinculados a registros de conteo que incrementan o decrementan, en relación al ciclo de reloj interno o al cambio de flanco en una señal externa. Los registros son de 8, 16 o 32 bits, normalmente se inicializan en cero (0 binario) o en ocasiones en un valor determinado por el programador y generan una interrupción al presentarse un sobre flujo, o paso de todos los bits en uno (1) a todos los bits en cero (0), muchas veces estos TIMERS se relacionan con registros de configuración como pre-escaladores o post-escaladores, que hacen que el registro TIMER incremente o decremente en múltiplos del ciclo máquina, pudiendo controlar la duración del sobre paso del TIMER.

Ejemplo: Si un ciclo máquina dura un (1) microsegundo, y se establece un pre-escalador 8:1, en un TIMER de 8 bits (256 incrementos hasta el overflow), se tendría que el TIMER inicializado en 00H (binario 00000000) tendrá un incremento de uno (1) cada ocho (8) microsegundos, es decir, a los 8 microsegundos, el registro TIMER es 01H (00000001_2), a los 16 microsegundos, el registro TIMER es 02H (00000010_2), hasta llegar a 2040 microsegundo en donde el registro TIMER es FFH (11111111_2) en donde está a punto de desbordarse o tener un sobre flujo, con 8 microsegundos más, el registro TIMER pasa nuevamente a 00H (00000000_2) momento en el que normalmente genera una INTERRUPCIÓN por TIMER. Por tanto el temporizador nos proporciona una base de tiempo de 2048 microsegundos o 2.048 milisegundos.

Contadores: Son módulos relacionados con los temporizadores, teniendo un funcionamiento similar, la diferencia radica en que el temporizador depende de la señal de reloj o señal interna, mientras que el contador recibe su señal de incremento o decremento de un pulso externo por medio de un pin o patilla especialmente acondicionada para recibir el pulso, usualmente es del tipo Smitt Trigger.

ADC: Los conversores análogos – digitales, son otro modulo muy común en los microcontroladores, básicamente son responsables de convertir una señal análoga (continua en el tiempo, ejemplo la temperatura), expresada como una señal de voltaje, en una señal digital (unos y ceros), mediante la técnica de muestreo.

Normalmente el módulo **ADC** tiene otros pines para determinar los voltajes de referencia, se caracteriza por la precisión en bits, de esta forma hay módulos **ADC** de 8, 16, 32 bits.

Ejemplo: Se tiene un sensor de temperatura que mediante un Amplificador Operacional, permite adecuar la señal de temperatura a un voltaje que varía entre 0 Volts y 5 Volts, se tiene un **ADC** de 8 bits, con dichos voltajes de referencia, por lo que se tiene un registro con 256 valores binarios que representarían los voltajes indicados en el muestreo, por tanto $5 \text{ Volts} / 256 = 0,01953125 \text{ Volts}$ por cada muestreo, esto es que 0 Volts tendría una representación en el registro de muestreo del ADC como 00000000_2 , $0,01953125 \text{ Volts}$ sería muestreado en el ADC como 00000001_2 , $0,0390625 \text{ Volts}$ sería muestreado en el ADC como 00000010_2 , hasta llegar a $4,98046875 \text{ Volts}$ que sería muestreado en el ADC como 11111110_2 y finalmente 5Volts que sería muestreado en el ADC como 11111111_2 . Por software se establece la temperatura equivalente a cada muestreo según el voltaje muestreado.

COMPARADOR: Mediante dos señales de voltaje de entrada, una de referencia y otra desconocida, se genera una comparación de ellas para determinar el estado de la señal de entrada desconocida.

PWM: (Pulse Width Modulator) Encargado de generar una señal modulada en ancho de pulso, es decir, que se varia el ancho de la señal útil (parte de la señal en estado alto-positivo), de una señal que tiene una frecuencia o periodo fijo.

Puerto serial síncrono (SSP): Es uno de los puertos más ampliamente implementados en los microcontroladores, se utiliza para permitir la comunicación serial del tipo síncrono o asíncrono con un PC (Computador Personal) y otros periféricos, como memorias seriales, registros de desplazamiento, conversores A/D, entre otros. Se distinguen dos tipos de puertos seriales, el USART (Universal Synchronous Asynchronous Receiver Transmitter) el cual permite la comunicación síncrona y asíncrona, y UART (Universal Asynchronous Receiver Transmitter) que permiten la comunicación solo asíncrona. Es usual que se deba acondicionar un circuito externo al micro para permitir la comunicación con el periférico en el estándar o interfaz requerido.

Ejemplo: Se puede requerir conectar el micro para intercambio de información con un PC utilizando la interfaz EIA232, conocida como RS232. A nivel industrial se puede requerir la comunicación mediante interfaz RS-485.

Puerto serial síncrono básico (BSSP): Esta interfaz del tipo síncrona, permite la comunicación del micro con otros periféricos. El **SPI** utilizado para comunicar el microcontrolador con otros microcontroladores, es una interfaz simple con un nodo controlador de las comunicaciones. El **I²C** similar en funcionamiento al SPI, con la característica que cualquier nodo puede iniciar la comunicación. El **USB** es la interfaz más conocida e implementada en muchos dispositivos, su utilización debida a los microcontroladores, se destaca por la velocidad de transferencia de datos, simplicidad y fácil implementación, este sistema trabaja por monitorización (polling) desde un maestro hacia dispositivos esclavos.

Puerto serial síncrono maestro (MSSP): Es un módulo para comunicación serial del tipo SSP.

Puerto paralelo esclavo (PSP): Es un puerto paralelo, implementado en los micro con 8 bits, multiplexados en los puertos de I/O.

Ethernet: Algunos microcontroladores de 32 bits, implementan módulos sofisticados para la conexión directa con redes de computadores, permitiendo el intercambio y acceso del micro a una red Ethernet. Algunos enruteadores se desarrollan y tiene como núcleo un microcontrolador de 32 bits con modulo Ethernet.

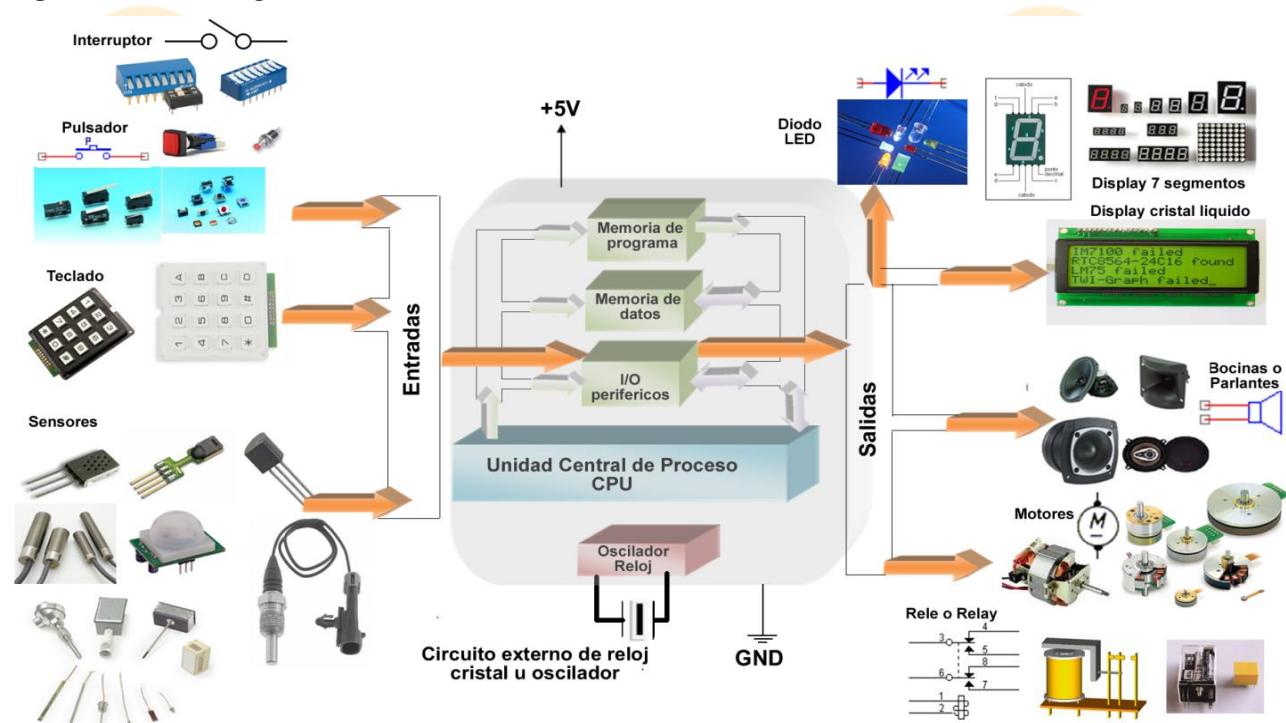
CAN: Es un protocolo de comunicaciones ampliamente utilizado en la industria automotriz en los OBD (acceso al sistema de control del auto) y en el control industrial en la capa física del “Field Bus”. Se caracteriza por implementar CSMA/CD Carrier Sense Multiple Access with Collision Detection, normalizado en IEEE 8023, cada estación puede enviar en cualquier instante siempre que no emita ninguna en dicho momento.

Memoria de datos no volátil: La mayoría de microcontroladores implementan un módulo de memoria no volátil, es decir, memoria que puede almacenar datos y no se pierden cuando el micro es retirado de la fuente de energía. Usualmente son pocos registros entre 128 Bytes y 256 Bytes, en los que el programa puede almacenar datos producto del proceso realizado o como valores de configuración inicial o como copia de seguridad del estado del micro. Esta memoria generalmente es del tipo EEPROM.

Periféricos externos al microcontrolador

La posibilidad de manejar señales de entrada y salida, procesar datos a gran velocidad, tomar decisiones en tiempo real, bajo consumo y ser robustos e inmunes al ruido convierten al microcontrolador en uno de los componentes más utilizados, versátiles y vendidos en la actualidad.

Figura 76. Sistema general microcontrolador⁹⁴



Dispositivos de entrada: Conforman este grupo todos los dispositivos capaces de cambiar su estado como respuesta a un evento que los afecte y generar una señal que pueda ser utilizada por el microcontrolador para ejecutar una operación de control o de decisión. Dentro de este grupo se encuentran, interruptores (switch), pulsadores, teclados, transductores y toda la gamma de sensores (temperatura, presión, humedad, luz, ph). Cada uno de ellos requiere ser adecuado eléctricamente a nivel de hardware, para que produzca los dos estados estables necesarios (uno y cero) a nivel de software. En los interruptores, pulsadores y teclado se busca generalmente que la pulsación genere “uno lógico”, es decir, existe un voltaje de 5 Voltios, la no pulsación debe mantener a “cero lógico” la entrada, aunque si sucede lo contrario es fácilmente adecuado a nivel de software, utilizando lógica negativa.

⁹⁴ Extraído el 10 de Julio de 2009 desde http://img.directindustry.es/images_di/photo-q/boton-pulsador-282573.jpg

Para los transductores y la mayoría de sensores, se requiere convertir la señal eléctrica producida por ellos a una señal digital, para lo cual se utilizan conversores ADC (analog-to-Digital Conversion), externos o incorporados en el microcontrolador.

Dispositivos de salida: Los dispositivos de salida se encargan de recibir las señales del microcontrolador y reproducir acciones particulares, entre los dispositivos más usuales están los visuales, auditivos y actuadores.

- Los dispositivos visuales más comunes son los LED (Light-Emitting Diode – Diodo emisor de luz) en forma individual o agrupados en los denominados “siete segmentos” y “matrices de LEDs”. También son utilizadas lámparas incandescentes desde bulbos de 1,2 Voltios hasta de cientos de Watts mediante relevos o dispositivos de estado sólido (SCR - TRIACs).
- LDC (Liquid Crystal Display – Display de Cristal Líquido), los cuales se fabrican en presentaciones de 1x16 (1 línea, 16 caracteres), 2x16 y gráficos.
- Auditivos, como parlantes o bocinas, zumbador (buzzer), que funcionan como emisores de sonidos de alarma, indicadores de actividad y de ingreso de información.
- Actuadores:
 - Rele o Relay, dispositivo electromecánico, ampliamente utilizado para el accionamiento de grandes cargas, es decir, dispositivos de gran consumo de corriente o voltaje, son accionados por etapas denominadas “driver”, construidas con semiconductores como los transistores.
 - Motores continuos, utilizando dispositivos de estado sólido o relevos para su accionamiento y puentes “H” para el control de giro.
 - Motores alternos, generalmente activados con relevos o dispositivos de cuatro capas como SCR o TRIACs.
 - Electroválvulas, dispositivo electromecánico, de apertura y cierre para el paso de líquidos o gases

Selección del microcontrolador

La elección de un microcontrolador es una acción que requiere considerar una serie de variables y parámetros antes de proponerse a implementar el diseño, el desarrollador de proyectos con microcontroladores se encontrara con muchos fabricantes, con familias de microcontroladores, cada una de ellas provee una amplia gamma y variedad de dispositivos, clasificados por sus características particulares (memoria, puertos, interrupciones, temporizadores, etc), cantidad de pines y tipos de empaques. Esta gran variedad hace de la selección del

microcontrolador una tarea que requiere mucha práctica que se consigue con el trabajo individual sobre los microcontroladores de uso general de cada familia.

Consideraciones generales

- La documentación existente sobre el producto, manuales, datasheet, textos especializados, ejemplos de desarrollo.
- Disponibilidad en el mercado, que se consiga localmente o en ciudades cercanas a precios cómodos.
- Herramientas de desarrollo, constituidas por ensambladores, emuladores, simuladores, programadores y entrenadores.
- Costo del producto, es de considerar que algunos fabricantes por mantenerse en el mercado disminuyen sus precios pero los soportes son insuficientes.

Las características del microcontrolador como memoria y su capacidad, temporizadores, interrupciones, velocidad de reloj, ADC, PWM, etc. que dependen del diseño de hardware, software y tipo de empaquetado con el cual se estima desarrollar el producto final.

Consideraciones de Aplicación

La selección de un microcontrolador para una aplicación es una tarea que requiere experiencia y conocimiento de la familia de dispositivos sobre la que se piensa implementar, teniendo presente las consideraciones generales antes mencionadas y las consideraciones de aplicación que a continuación se exponen:

- **Entradas y salidas**, en el proceso de diseño se debe establecer la cantidad y tipo de entradas y la cantidad y tipo de salidas, junto con el microcontrolador seleccionado, para lo cual se emplean diagramas esquemáticos simples para identificar rápidamente las entrada, salidas y dispositivos de hardware necesarios. Se procede al análisis del diagrama donde se ratifica el microcontrolador propuesto o el cambio por otro más adecuado.
- **Memoria**, para establecer los requerimientos de memoria que debe cumplir el microcontrolador, se separa en:
 - Memoria de programa – ROM, es la memoria que se utiliza para el almacenamiento del programa, su tamaño es determinado por la cantidad de Bytes que ocupa el programa en lenguaje máquina.
 - Memoria de datos – RAM, es la memoria que contiene los registros de propósito especial y los registros de propósito general (utilizados por el programador para definir variables y dinamizar el flujo del programa).

- Memoria no volátil, de tipo EEPROM y FLASHRAM, son pocas localidades de memoria que almacenan datos de calibración, parámetros iniciales, estado anterior o número de serie.
- **velocidad del reloj**, influye en la velocidad de procesamiento y capacidad de trabajo en tiempo real, para aplicaciones generales se utilizan velocidades menores o iguales a 4MHz, muchas de las familias soportan varias decenas de MegaHertz en el circuito de reloj.
- **Datos y procesamiento**, se recomienda seleccionar el microcontrolador con menor ancho de palabra que satisfaga las necesidades de aplicación, generalmente se manejan datos de ancho cercano o igual a 8 bits (1 Byte), con lo que se prefiere los microcontroladores de 8bits, aunque todavía se utilizan microcontroladores de bus de datos de 4 bits. En aplicaciones que requieren espacio de direccionamiento grande, velocidad de proceso y longitudes de palabra mayores a 8bits, se eligen microcontroladores de 16 o 32 bits.
- **Precisión**, si la necesidad radica en la precisión de los datos a procesar puede que un microcontrolador de 8 bits no sea suficiente, indicando que se debe optar por uno de 16bits, 32bits o uno de coma flotante.
- **Consumo**, los microcontroladores tienen consumos muy bajos pudiendo ser alimentados por baterías, el consumo aumenta dependiendo de los componentes periféricos al microcontrolador, si el consumo es un elemento crítico de diseño, es recomendable utilizar dispositivo que permitan “estado de bajo consumo” y utilizar celdas de iones de litio como fuente de alimentación.

Diseño de la placa de circuito, el diseño de la placa está condicionada por el tipo de microcontrolador utilizado. Dependiendo de las entradas y salidas y los periféricos conectados a ellas como teclados, Display 7 segmentos, LCD, Motores CC, etc, hacen que un microcontrolador sencillo con pocos pines, incremente el costo de la implementación. Sin embargo utilizar un microcontrolador complejo que incorpore unidades funcionales y capacidades como manejadores de display LCD, PWM, comunicaciones, etc. Necesita un software complejo para controlar sus capacidades internas y la multiplexación de sus pines.

Diferencias entre sistemas basados en microprocesadores y microcontroladores

Entre las tecnologías de microprocesadores y microcontroladores existen varias diferencias. Cada tecnología tiene ventajas y desventajas donde su uso depende

de la aplicación particular, por lo que es importante en esta lección estudiar y aclarar estas diferencias.

CPU: La CPU de un microprocesador es más simple, sus instrucciones están orientadas al manejo y operación de las líneas de entrada y salida, generalmente utilizan un conjunto reducido de instrucciones.

RAM: La capacidad de direccionamiento de memoria en los microprocesadores es mucho más elaborada pudiendo acceder a grandes bancos de memoria acordes a los requerimientos del sistema. En un microcontrolador la memoria RAM o memoria de datos, que se incorpora dentro del chip es de baja capacidad, puesto que las aplicaciones de control no necesitan almacenar grandes cantidades de datos temporales.

ROM: Los microprocesadores y su sistema de chips periféricos permiten el acceso a dispositivos de almacenamiento externo de gran capacidad ROM, RAM, FLASH, Magnéticos y ópticos de distintas tecnologías. La memoria ROM o memoria de programa en los microcontroladores es de capacidad limitada y está alojada dentro del mismo chip, es utilizada para el almacenamiento del programa en lenguaje máquina.

Implementación: La implementación con microprocesadores involucra placas impresas multicapa, para permitir la interconexión de una compleja red de circuitos y chips individuales relacionados con la codificación y decodificación, memoria, almacenamiento y periféricos, convirtiéndolo en algo tan complejo como la mainboard de un computador personal.

Con los microcontroladores la implementación es mas sencilla, usualmente requiere solo una capa en el circuito impreso, para aplicaciones simples el circuito involucra el reloj generalmente conformado por un cristal de cuarzo y un par de condensadores, unos pocos LEDs, resistencias, pulsadores, filtros de alimentación y conexión de baterías.

Tipo de sistema, abierto y/o cerrado: Un ejemplo de sistema abierto es el microprocesador el cual dispone de líneas o pines externos accesibles al diseñador, estos comunican el hardware exterior con los buses de dirección, datos y control en el microprocesador.

La arquitectura de sistema cerrado es propia de los microcontroladores, generalmente no permiten que el diseñador tenga acceso a los buses de

dirección, datos y control internos de la CPU, no obstante, muchos microcontroladores permiten acceso a los buses a través de los puertos de entrada / salida utilizando señales de sincronización que permiten la conexión de chips RAM o ROM para expandir su capacidad de memoria.

Velocidad de operación: Actualmente los microprocesadores tienen velocidades de operación del orden de los GigaHertz con varios núcleos, en los microcontroladores la velocidad de operación es mucho más lenta, de hasta varias decenas de MegaHertz por encima de los 50MHz, pero es suficiente para controlar sistemas en tiempo real. El circuito responsable de la velocidad del reloj que está estrechamente relacionado con la velocidad de operación puede ser interno o externo, cuando es externo los pulsos o ciclos por segundo (Hertz) son producidos por un oscilador digital, cuando es interno puede ajustarse el número de ciclos por segundo utilizando varios dispositivos y configuraciones conectadas a los pines del oscilador en el chip microcontrolador, entre los dispositivos y configuraciones más usuales están, la red interna o externa de resistencia-condensador.

Sistemas de desarrollo

Para los microprocesadores se considera como sistema de desarrollo el hardware conectado al micro en conjunto con los paquetes de software como, intérpretes de comandos, compiladores, programas de bajo a alto nivel, encargados de escribir, ensamblar, depurar y ejecutar los programas en lenguaje máquina.

En los microcontroladores se requiere un sistema de desarrollo diferente para cada familia de microcontroladores, estos sistemas de desarrollo se componen de:

- **Software**, con capacidad para editar, ensamblar, compilar y simular los estados y comportamiento del microcontrolador.
- **Hardware**, para programar o “quemar” el microcontrolador, es decir, gravar en la memoria del microcontrolador el programa.

Diversidad de periféricos

Los sistemas basados con microprocesadores incorporan una amplia variedad de periféricos que satisfacen necesidades donde la capacidad y potencia de un microcontrolador no es suficiente y donde la implementación es muy compleja. Se hace referencia al procesamiento de datos, que tienen que ver con audio, video, información, bases de datos, comunicaciones, entretenimiento, aplicaciones industriales y espaciales entre otras.

Para proyectos y aplicaciones con una complejidad manejable dentro de las capacidades de los microcontroladores actuales, con la movilidad y portabilidad que los caracteriza, existe uno o varios microcontroladores con características singulares que permiten la implementación con relativa simplicidad.

- Si se requiere controlar fenómenos de naturaleza análoga como temperatura, humedad, voltaje etc, se utilizan dispositivos que incorporen un convertidor análogo-digital.
- Cuando se requiere medir o generar periodos de tiempo, tonos o frecuencias, se busca tener en el chip, temporizadores programables (timers).
- La interacción con el entorno puede requerir respuestas en tiempo real a eventos que deben ser monitoreados por el microcontrolador a través de sus pines por lo que se requiere que el chip contenga fuentes de "interrupción".
- Las necesidades de comunicación con otros microcontroladores o con un microprocesador se satisfacen con dispositivos que incorporen soporte para comunicaciones seriales (RS-232), paralelas, de red etc.
- Para controlar actuadores como motores o cargas resistivas debe tener incorporado un modulador de ancho de pulso o PWM.

Conclusión

En conclusión, un sistema basado en microcontroladores tiene ciertas ventajas notables con respecto a los sistemas basados en microprocesadores:

- El circuito impreso y su montaje es más sencillo pues el microcontrolador incorpora muchos de los componentes internamente, reduciendo también el costo.
- Al integrarse la mayoría de componente sensible al ruido en un solo chip, el microcontrolador es prácticamente inmune al ruido lo que lo hace ideal para aplicaciones en casi cualquier campo de desarrollo.
- El tiempo de desarrollo de un sistema basado en microcontroladores se reduce notablemente.

- Sin embargo, cuando la complejidad de un sistema supera las prestaciones y características de un microcontrolador como capacidad de memoria, longitud de palabra, capacidad de manejo de datos, velocidad de proceso, numero de pines, puertos de I/O, etc. Se opta por un sistema basado en microprocesadores.

Procesadores Digitales de Señal

Digital Signal Processor (Procesador Digital de Señal), en 1978 Intel lanza al mercado el primer “procesador analógico de señales”, el 2920, dispuesto de un ADC/DAC y procesador de señales en un mismo chip, en 1979 AMI lanza el S2811, ambos dispositivos sin relevancia en el mercado. Bell Labs en 1979 produce el DSP “The Mac 4 Microprocessor”, seguido por DSP más completos como el PD7710 de NEC, DSP1 de AT&T, llegando a DSP más sofisticados y destacados en el mercado como el TMS32010 y TMS320C4X de Texas Instruments.

Es un sistema basado en un microprocesador, muy similar a un microcontrolador, puesto que en un mismo chip se integra, una CPU, una memoria y periféricos (Puertos I/O), junto con módulos especiales que diferencian este dispositivo de los microprocesadores y microcontroladores, estos módulos principalmente son conversores analógicos digitales (ADC) utilizados como entrada de información, convertidores digitales analógicos (DAC) y moduladores de ancho de pulso (PWM) utilizados como salida de información, lo convierten en una herramienta para el proceso y tratamiento de señales analógicas en tiempo real.

Al igual que en los microprocesadores y microcontroladores, posee un set de instrucciones implementados en arquitectura del tipo CISC o RISC, directamente relacionado con una arquitectura preferiblemente Harvard. El set de instrucciones implementa además de las instrucciones usuales de tratamiento de datos, instrucciones sofisticadas para el procesamiento digital. Tiene el potencial de trabajar con varios datos en paralelo y la característica de procesamiento en tiempo real de un microcontrolador, lo que hace una herramienta valiosa en el procesamiento de señales en tiempo real como en señales analógicas de audio y video, sensores y comunicaciones. Permite implementar filtros digitales de señal, reconocimiento de voz, filtro de imágenes, cifran y posibilitan comunicaciones digitales (wireless, módems, etc), analizan datos de sensores analógicos, permiten la reproducción de alta fidelidad en sistemas de audio y video (cámaras).

Lección 17: Arquitectura interna y direccionamiento

La arquitectura en los microcontroladores se refiere a la forma como la CPU accede a la memoria y a la cantidad o set de instrucciones de cada familia. En esta lección se abordan las principales arquitecturas de microcontroladores referidas a la disposición del microprocesador y la memoria de datos y programa, como la Von Neumann, la Harvard, junto con las arquitecturas referidas a la ALU, matriz de registros, set de instrucciones y su ejecución, con las arquitecturas CISC, RISC y SISC. El estudio continua enfocado en la memoria y la forma como se accede (direccionamiento), junto con otras características especiales de los microcontroladores y su arquitectura.

Arquitectura Von Neumann

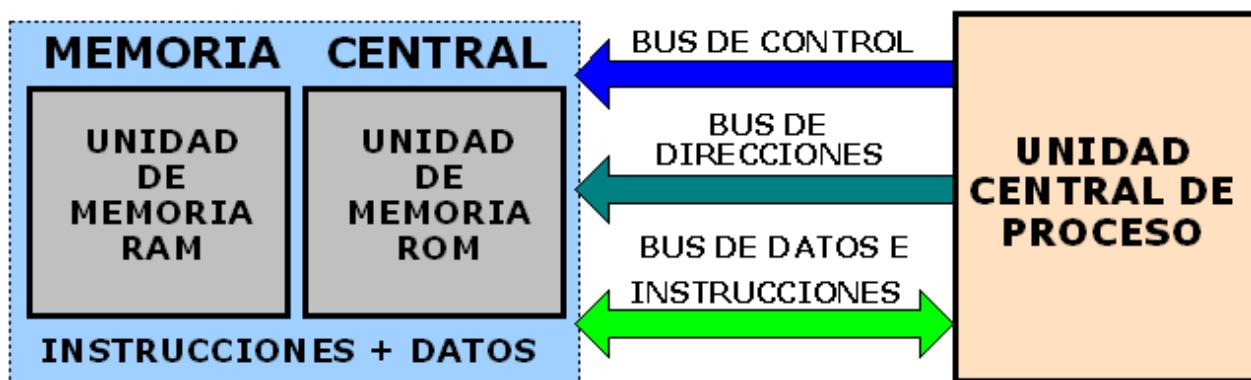
Con esta arquitectura se hace el diseño conceptual y la estructura operacional de la mayoría de microprocesadores y de computadoras de uso personal que se utilizan desde su aparición a la fecha. Esta arquitectura esta basada en el concepto de programa almacenado propuesto por el matemático Von Neumann y propuesto por Jhon Presper Ecker, Jhon William Mauchly, Arthur Burks, entre otros en el periodo de construcción de la ENIAC. En la arquitectura Von Neumann la CPU se conecta a una memoria principal única generalmente del tipo RAM, donde se almacenan los datos y el programa, accediendo a través de un sistema de buses único, como son el bus de dirección, control y datos.

El ancho del bus que comunica la memoria con la CPU determina el tamaño de la unidad de datos o instrucciones, un microprocesador de 8 bits con bus de 8 bits tendrá que manejar datos o instrucciones de 8 bits de longitud. Para el acceso a datos o instrucciones de más de 8bits tendrá que realizar más de un acceso a la memoria.

La arquitectura Von Neumann tiene varias limitaciones:

- La longitud de las instrucciones están limitadas por el bus de datos, lo que hace que el ejecutar una instrucción compleja requiera varios accesos a memoria.
- El microprocesador es mas lento en su respuesta, la velocidad de operación se afecta por tener un único bus para datos e instrucciones lo que impide acceder a la memoria de datos y de instrucciones simultáneamente, es decir, no permite superponer tiempos de acceso.

Figura 77. Arquitectura Von Neumann⁹⁵



Arquitectura Harvard

El término proviene de la Harvard Mark I, la cual almacenaba los datos en cintas perforadas y las instrucciones mediante interruptores, la arquitectura Harvard se caracteriza por tener separados los bloques de memoria de datos e instrucciones y acceder a ellos por buses independientes de dirección, datos y control. La independencia de buses permite tener accesos simultáneos e independientes a la memoria de datos e instrucciones, el contenido y longitud de las localidades de memoria pueden ser distintos para los datos e instrucciones, esto permite una optimización en el uso de la memoria.

Los diseñadores aprovechan este concepto donde la memoria de datos puede por ejemplo de 8bits, mientras la memoria de instrucciones se aadecua a la longitud de las instrucciones buscando que cada instrucción se aloje en una posición de memoria, con lo que la ejecución de una instrucción puede hacerse en un solo ciclo máquina, permitiendo también la superposición de tiempos de acceso, por tanto en el mismo lapso que busca y ejecuta una instrucción puede estar realizando una acción de lectura o escritura en la memoria de datos. Esta característica es explotada por microprocesadores y microcontroladores con conjunto de instrucciones reducido (RISC).

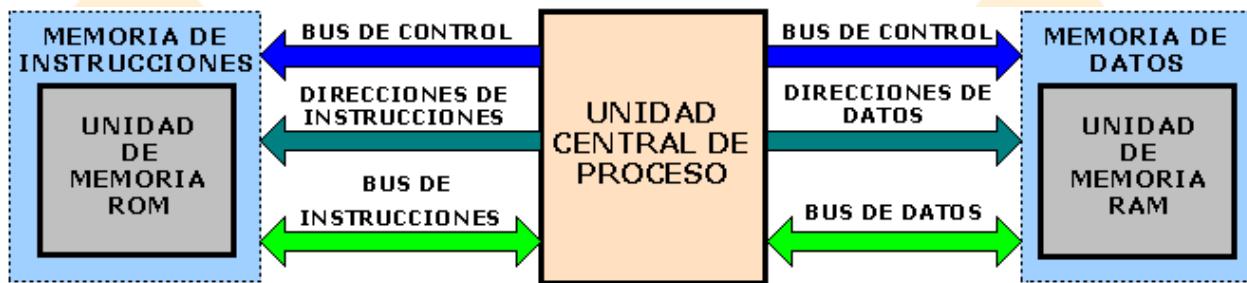
La arquitectura Harvard tiene ventajas significativas con respecto a la arquitectura Von Neumann, las más significativas son:

- El tamaño de las instrucciones no está relacionado con el tamaño de los datos permitiendo optimizar la memoria haciendo que cada instrucción ocupe una única posición de memoria, esto hace que la longitud de programa puede ser menor.

⁹⁵ Extraído el 10 de Julio de 2009 desde <http://perso.wanadoo.es/pictob/microprg.htm>

- La posibilidad de superponer tiempos de acceso, es decir, poder acceder a la memoria de programa y a la memoria de datos en el mismo ciclo máquina, esta característica y la anterior permiten una velocidad de operación más alta.

Figura 78. Arquitectura Harvard⁹⁶



Arquitectura CISC

Compleja Instrucción Set Computing o Computadores de juego de instrucciones complejo, la mayoría de CPUs utilizadas en microcontroladores están basados en esta arquitectura, dentro de las características más relevantes están:

- Un gran número de instrucciones de longitud variable.
- Generalmente más de 80 instrucciones en su set de instrucciones.
- Instrucciones muy sofisticadas y potentes, que actúan como macros.
- Instrucciones que requieren un numero de múltiplos de ciclo maquina.
- Modos de direccionamiento múltiple.
- Pequeño número de registros de trabajo de propósito general.

Esta arquitectura dificulta el paralelismo entre instrucciones, en la actualidad los sistemas con CISC de alto rendimiento implementan sistemas que convierten instrucciones complejas en simples del tipo RISC, denominadas microinstrucciones.

Arquitectura RISC

Reduced Instruction Set Computer, Computadores con set de instrucciones reducido, esta arquitectura se implementa con gran éxito actualmente en microcontroladores PIC, como características principales están:

⁹⁶ Extraído el 10 de Julio de 2009 desde <http://perso.wanadoo.es/pictob/microcr.htm>

- Conjunto de instrucciones reducido, generalmente menor o igual a 120.
- Típicamente un número reducido de modos de direccionamiento, que son las formas como el procesador utiliza la memoria, básicamente cuatro (4) modos.
- El procesador tiene un número superior de registros de propósito general típicamente de 32 registros.
- Todas las instrucciones se ejecutan típicamente en un solo ciclo maquina, compuesto de unos pocos ciclos de reloj, generalmente cuatro ciclos de reloj.

La longitud de las instrucciones tiende a ser fija y pequeña entre 12 y 32 bits con un número reducido de formatos.

Arquitectura SISC

Specific Instruction Set Computer, computado con juego de instrucciones específico, los microcontroladores que son destinados a aplicaciones muy concretas tienen un juego de instrucciones además de reducido y “específico”, es decir que se adaptan a una aplicación predefinida.

Núcleo del microcontrolador

El núcleo se refiere a las características fundamentales que son requeridas para que el “micro” realice las operaciones básicas, entre ellas están:

CPU: Unidad Central de Proceso, es la responsable de tomar las instrucciones desde la memoria de programa, ejecutarlas y así controlar la operación de todo el sistema. Está conformada por:

- **ALU**, Unidad Aritmético Lógica encargada de interactuar con la memoria de datos en las operaciones aritméticas y lógicas.
- **UC**, Unidad de control, busca las instrucciones en la memoria de programa, las decodifica y las ejecuta.
- **Matriz de registros**, los conforman por registros visibles al usuario como el acumulador o registro de trabajo, temporizador, entre otros y los registros de control y estado como el registro de estado, contador de programa, registro de interrupciones entre otros.

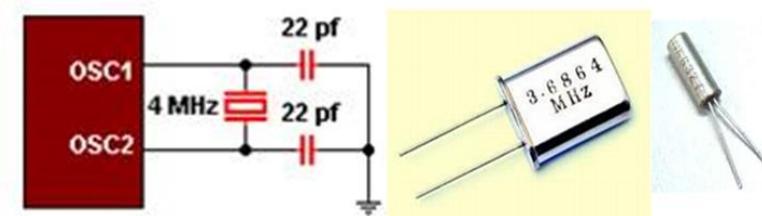
Mapa de memoria: compuesto por la disposición en bloque o bancos de memoria. La arquitectura Von Neumann generalmente maneja un solo bloque de memoria mientras que la arquitectura Harvard maneja hasta cuatro bancos de memoria, que contienen las funciones particulares de algunas localidades y los registros de usuario.

PILA: conformada por una pequeña porción de memoria ubicada generalmente en la memoria de programa en la arquitectura Harvard, esta determina la profundidad o la cantidad de llamados sucesivos a subrutinas. Por ejemplo una pila de 2 niveles indica que pueden realizarse un llamado a una subrutina dentro de otra subrutina, superar la profundidad o niveles de pila implica una perdida de secuencia en el programa causando errores, esto es llamado “desbordamiento de pila”.

Circuito Oscilador: es un circuito encargado de generar pulsos o señal de reloj necesaria para que el microcontrolador sincronice y ejecute las instrucciones y funciones adecuadamente en los periféricos. Los circuitos osciladores mas comunes están basados en componentes que determinan la frecuencia entre ellos podemos mencionar los siguientes:

- **INTRC:** Es una red de resistencia – condensador interna, es la mas económica y permite tener un par de pines extra para I/O, no todos los microcontroladores lo implementan, es necesario recurrir a las hojas de especificaciones para comprobar su disponibilidad.
- **RC:** Red resistencia – condensador externo, es una solución económica pero inestable y poco precisa.
- **Cristal de cuarzo:** son redes compuestas por cristales de cuarzo, generalmente con un par de condensadores conectados entre tierra (GND) y cada terminal del cristal y el microcontrolador.
 - **LP:** cristal de baja potencia, tiene consumo bajo de corriente, con frecuencias entre 32KHz y 200KHz.
 - **XT:** cristal / Resonador con un consumo mas elevado que el anterior y frecuencias entre 100KHz y 4MHz.
 - **HS:** cristal / Resonador de alta frecuencia permite trabajar a gran velocidad pero también incrementa el consumo, tiene frecuencias mayores a 8MHz.

Figura 79. *Oscilador con Cristal de cuarzo*⁹⁷



⁹⁷ Extraído el 10 de Julio de 2009 desde http://grupodcti.blogspot.com/2008_10_01_archive.html

Sistema de Reanudación o Reset: es usado para llevar al microcontrolador a un estado conocido, el vector de reset se direcciona generalmente a la localidad de memoria 0000H. En este estado se establecen condiciones iniciales estables con las que siempre inicia el sistema y con las que se garantiza un buen funcionamiento de todas las tareas.

Conjunto de interrupciones: generalmente los microcontroladores implementan varias fuentes de interrupción las cuales son atendidas utilizando vectores que apuntan a localidades específicas dentro de la memoria de programa.

Conjunto de instrucciones: cada instrucción en lenguaje ensamblador se divide en un código de operación “OPCODE”, que especifica el tipo de instrucción y uno o mas operandos que especifican la operación de la instrucción, por ejemplo la instrucción “MOVLW 07H” implementada en un PIC con arquitectura Harvard de 13 bits en longitud de bus de instrucciones se divide como se ilustra a continuación:

Tabla 22. *Instrucciones y assembler*⁹⁸

	bits	
Longitud	13 . . . 8	7 . . . 0
Instrucción	OPCODE	K (literal)
mnemónico	MOVLW	07H
Hexadecimal	30	07

Arquitectura de Periféricos

Son los elementos, unidades funcionales, módulos o soportes que el microcontrolador tiene para interactuar con el exterior, son los que hacen la diferencia entre microprocesadores y microcontroladores. Los periféricos se encargan de la comunicación entre el “micro” y el mundo exterior mediante los puertos de I/O, manejadores de LCD, conversores ADC, PWM, etc. En general existen muchos tipos de periféricos entre los que podríamos mencionar los siguientes:

Pines o Línea de entrada / salida (I/O) de propósito general: estos permiten al microcontrolador comunicarse con el mundo exterior enviando y recibiendo señales, son utilizados para supervisar y controlar dispositivos conectados al microcontrolador, muchos de estos pines son multiplexados para que tengan funciones alternas, es decir, pueden funcionar como pines de entrada, de salida o incorporar funciones adicionales.

⁹⁸ CEKIT, 2002

Temporizadores: los microcontroladores implementan uno o varios temporizadores que establecen bases de tiempo confiables, intervalos, tiempo entre eventos, etc.

Módulos de conversión: estos módulos permiten convertir señales y almacenarlas en el “micro” para su posterior procesamiento y toma de decisiones o enviar señales al exterior adecuadas para cierto tipo de dispositivos, entre estos están los módulos ADC y PWM.

Módulos de comunicación: como su nombre lo indica son módulos que permiten la comunicación entre el micro y el exterior y/o viceversa, entre los más usuales están, la comunicación serial síncrona, USART y comunicación por puerto paralelo.

Comparadores, estos módulos se implementan para comparar señales análogas y servir como referencia para la toma de decisiones y control de procesos desde el microcontrolador.

Características especiales

Las características especiales ayudan a disminuir costos de implementación, sencillez en el montaje, funcionalidad, flexibilidad y convierten el sistema basado en microcontroladores en un sistema robusto aplicado a casi cualquier ambiente de trabajo, entre las características más sobresalientes están:

Bits de configuración de dispositivo: estos bits permiten al usuario personalizar el modo de trabajo del microcontrolador, como tipo de oscilador, encriptación o protección de programa entre otros.

Sistema de protección e inicio: muchos microcontroladores cuentan con circuitos que vigilan el estado de la alimentación eléctrica generando un auto-reset en caso de encontrar variaciones o cambios drástico en la fuente de poder, también implementan circuitos que direccionan el vector de reset para garantizar el comienzo del programa en la localidad de memoria preestablecida.

Temporizadores al encendido o en funcionamiento: los temporizadores al encendido generan tiempos de espera para permitir que se estabilice la señal de alimentación y los pulsos generados por el reloj, durante el funcionamiento pueden generar eventos de “reset” en períodos definidos evitando bloqueos, situaciones no deseadas o fallas en el proceso.

Modo de bajo consumo: generalmente los microcontroladores disponen de instrucciones que permiten entrar en un estado de bajo consumo o “sleep” apropiado para situaciones donde la duración de las baterías es un punto crítico.

Oscilador interno: el circuito oscilador requiere típicamente dos (2) pines del microcontrolador para conectar los componentes que generan la base de tiempos para el pulso de reloj, en ocasiones estos dos pines son necesarios para utilizarlos como control de algún dispositivo o como líneas I/O, como característica adicional y para tener un par de pines extra se implementan en algunos modelos de microcontroladores osciladores internos.

Programación serial dentro del circuito, actualmente la mayoría de modelos de microcontroladores permiten ser programados dentro del mismo circuito de aplicación.

Memoria en los microcontroladores.

Se distinguen varios tipos de memoria ROM:

EPROM: Las memorias EPROM en los microcontroladores son memorias utilizadas para el desarrollo de prototipos, esta memoria tienen la capacidad de ser programada y borrada por luz ultravioleta cientos de veces, los chips que las incorporan tienen una ventana translúcida que permite su borrado por exposición a luz ultravioleta durante un periodo de 20 a 30 minutos. Para que el programa no se borre se debe cubrir la ventana, la exposición a luz día, solar, fluorescente e incandescente con la ventana descubierta hace que en cuestión de horas, días o semanas se borre el programa.

OTP: (One Time Programmable), son memorias programables una sola vez. Cuando un prototipo con microcontroladores se termina y ha sido completamente probado además se quiere su producción en masa, se elige este tipo de memoria por su bajo costo.

EEPROM o E²PROM: estas memorias en los microcontroladores se caracterizan por su capacidad de ser borradas eléctricamente, este proceso se hace en unas pocas fracciones de segundo, donde se escribe o borra una celda a la vez. con respecto a las **EPROM** no necesitan dispositivos especiales para ser borradas y pueden repetir el proceso de borrado y reprogramación muchas mas veces.

FLASH: este tipo de memoria es una variedad más sofisticada de la **E²PROM**. Los microcontroladores que la implementan, se pueden borrar eléctricamente, con pulsos de voltaje más bajos, la velocidad de escritura o borrado es mucho más rápida pues toma varias celdas de almacenamiento a la vez, soportan millones de veces de programación y borrado, su consumo es menor con respecto a sus predecesoras.

Modos de direccionamiento

Existen varios modos de direccionamiento comunes tanto para microprocesadores como para microcontroladores, observemos como se presentan los modos de direccionamiento en los microcontroladores.

Direccionamiento Implícito: Estas instrucciones se caracterizan por no requerir el uso de operandos, debido a que no necesitan acceder a la memoria de datos, por ejemplo la instrucción “NOP”, esta instrucción no produce ningún efecto visible pero es útil al brindar un retardo de una cantidad de ciclos maquina para ajustar retardos por software.

Direccionamiento inmediato: Se presenta cuando el dato no proviene de la memoria. El dato está incluido dentro de la misma instrucción, por ejemplo “MOVLW 5AH”, esta instrucción almacena el literal 5AH en el registro de trabajo “W”, este tipo de instrucciones utilizan datos constante o ya establecidos.

Direccionamiento directo: Es utilizado cuando el dato se transfiere hacia, o desde, una posición de memoria, por ejemplo “MOVWF 0x0C”, la anterior instrucción accede a la localidad de memoria 0x0C y copia el dato en el registro de trabajo “W”.

Direccionamiento relativo: Las instrucciones de salto permiten alterar la ejecución secuencial de un programa, este tipo de instrucciones mediante el uso de un cálculo simple suman un valor al contenido del registro “contador de programa” el cual contiene la dirección de salto. Este direccionamiento es aplicado en las “Tablas”.

Direccionamiento indexado: La dirección de destino se calcula utilizando como base un registro especial, por ejemplo “LDA \$300, X” carga el registro A con el dato almacenado de la dirección 300 de memoria más el contenido del registro “X”.

Direccionamiento extendido: Permite acceder a todo el espacio de memoria del microcontrolador, la dirección se almacena en dos o tres bytes del operando, de modo que sin importar el tamaño de la dirección destino se puede acceder directamente a ella. Por ejemplo, “LDA \$3A4F”, carga el registro A utilizando el dato almacenado en la dirección absoluta “3A4F”.

Direccionamiento Indirecto: La dirección de una localidad de memoria se calcula mediante una doble referencia, en primer lugar se obtiene el dato contenido en una posición de memoria y con base en este dato, se obtiene la dirección efectiva de la posición de memoria deseada. Por ejemplo “LDA (\$300)”, esta instrucción carga el registro A utilizando como base el dato contenido en la dirección “300”.

Lección 18: Sistemas microcontrolados: ensamblador y programación en microcontroladores

La solución a un problema de control electrónico, basado en microcontroladores, incluye dos etapas fundamentales:

- Escribir el programa en lenguaje ensamblador
- Generar el archivo ejecutable que debe cargarse en la memoria del microcontrolador

El ensamblador para microcontroladores está conformado por varios módulos independientes cada uno de los cuales cumple una función específica.

Ensamblador básico: Genera a partir del código fuente, un archivo binario relocalizable, este archivo se puede almacenar en cualquier segmento disponible de memoria del microcontrolador.

Enlazador (linker): a partir del archivo binario relocalizable, se crea un archivo binario ejecutable, el cual es grabado en la memoria del micro.

Control de librerías: permite la creación de archivos binarios que pueden ser unidos (enlazados) con otros bloques de código binario, lo que facilita la reutilización de partes de programas generados en otros proyectos.

Desarrollo de proyectos

El ejercicio de desarrollar un control o proyecto basado en microcontroladores, está sujeto a un ciclo que es representado por:

- Requisitos del problema.

- Análisis de la solución.
- Diseño de recursos.
- Codificación.
- Pruebas.
- Mantenimiento.

Este ciclo se resume en una serie de actividades familiares a un programador de proyectos con microcontroladores, este ciclo está conformado por:

- Programador, en esta etapa se establecen las entradas/salidas, se diseña el algoritmo y su diagrama de flujo.
- Editor de texto, utilizando los Entornos de Desarrollo Integrado – IDE o cualquier editor básicos, se procede a convertir el algoritmo o diagrama de flujo en instrucciones en lenguaje ensamblador compatible con el dispositivo escogido.
- Código fuente, utilizando los IDE se procede a generar el código fuente mediante la compilación del archivo editado.
- Programa ensamblador, este genera los archivos .OBJ, .LST y .HEX a partir del código fuente.
- Archivo ejecutable, como resultado del proceso de compilación se obtiene un archivo ejecutable con extensión .HEX el cual es tomado para grabar en la memoria del microcontrolador.

Método de trabajo para el desarrollo de aplicaciones

Los pasos que se mencionan a continuación pueden ser adecuados, la secuencia responde a un ejercicio constante de verificación y correcciones tanto en el código fuente y ejecutable, como en el diseño base del algoritmo. El método es simple y se recomienda seguir los pasos para lograr los objetivos:

- Requisitos del problema: se establecen las entradas, salidas y componentes periféricos como LEDs, Display 7 segmentos, LCD, teclados, relés, bocinas, motores, etc. Con lo que se obtiene un posible microcontrolador para la aplicación.
- Análisis de la solución: se reevalúa el microcontrolador y dispositivos periféricos pudiendo elegir un chip con capacidades acordes que permita sencillez en la implementación, se tiene en cuenta la precisión, longitud y velocidad de proceso del microcontrolador a elegir. Se pretende tener al final de esta etapa el pseudocódigo, algoritmos y diagramas de flujo.
- Diseño de los recursos: Se establece el listado de componentes, se procede a buscar en hojas de especificaciones las características propias de cada dispositivo y la forma de adecuar sus señales de entrada o salida

a las señales digitales o analógicas que el microcontrolador recibe o entrega.

- Código fuente: con lo logrado hasta ahora se procede a convertir el algoritmo en código ensamblador a fin de obtener el código fuente del programa.
- Ensamblador: con el código fuente ya terminado se procede a ensamblar en programa fuente convirtiéndolo en código binario (HEX) que se grabara en la memoria del micro mediante el programador (hardware y software).
- Código binario: como resultado del ensamblado del código fuente se obtiene el código binario en un archivo con extensión .HEX, el cual contiene las instrucciones y secuencias de decisión y control, es el fruto del trabajo de las primeras dos etapas.
- Grabación en memoria: consiste en tomar el código binario, archivo .HEX, y grabarlo en la memoria del microcontrolador utilizando programas especiales vinculados a través de puerto USB, paralelo o serial a un hardware diseñado para alojar el microcontrolador y recibir los pulsos que permiten la grabación del programa en la memoria ROM en su interior.
- Prueba del sistema: en este punto se prueba el programa en el montaje implementado en protoboard y se establece si deben haber correcciones o se debe mejorar, lo que puede implicar saltar al paso “análisis de la solución”, para volver a reevaluar la solución, generar el algoritmo, etc.
- Producto final: esta es la última etapa en el proceso de trabajo para el desarrollo de aplicaciones con microcontroladores, en este punto se tiene establecido y completamente probado el programa y el circuito, es decir se tiene el software y hardware de control, el paso final es pasar el circuito montado a un circuito impreso definitivo, para su empaque y distribución.

Dispositivos de entrada

Los dispositivos de entrada hacen referencia a los periféricos encargados de suministrar información al micro, estos dispositivos deben adecuar la señal para suministrar valores estables de voltajes entre 0Volts y 5Volts, representando de esta manera los dos estados estables binarios (0 y 1), en el caso de entregarse la información a pines configurados con conversores analógicos digitales, se debe adecuar la señal a valores de voltaje entre 0Volts y 5Volts o acorde a los voltajes de referencia.

Dispositivos ON-OFF: Se tienen varios dispositivos de entrada, desde los más usuales que entregan un estado estable entre dos valores de voltaje (0 V – 5V), como los pulsadores, hasta dispositivos de aplicación más específica como interruptores de fin de curso, fin de carrera, teclados matriciales (arreglo de pulsadores), dip-switch (arreglo lineal de interruptores), micro switch integrados (4066B), sensores ópticos (fotoceldas, fotodiódos o fototransistores) y en general cualquier dispositivo que entregue una señal del tipo ON (5V) y OFF (0V).

Dispositivos de señal continua: Estos dispositivos de entrada, hacen referencia a elementos que suministran una señal continua al microcontrolador, señal que debe ser recibida por pines acondicionados con conversores analógicos digitales, entre ellos están potenciómetros analógicos y la mayoría de sensores ambientales, como sensores de temperatura, humedad, presión, Luz, PH, entre otros.

Dispositivos de salida

Los dispositivos de salida se refieren a elementos que permiten una visualización del estado del proceso, una interfaz de observación al usuario o un conjunto de actuadores que ejecutan una acción sobre la planta a controlar. Se pueden generar una clasificación de estos dispositivos en relación a la prestación deservicio que tienen. Estos dispositivos utilizan información binaria directa o en la forma de protocolo de transmisión serie o paralelo, en los pines o patillas del micro mediante acondicionadores de señal para dar el voltaje o corriente suficiente, mediante transistores (BJT, UJT, FET, MOSFET) o circuitos integrados (CI)

Dispositivos visuales: Dispositivos visuales tan usuales como los LEDs, en todas sus presentaciones desde unidades de diversos tamaños y formas hasta conjuntos de LEDs como en los Display Digital 7-segmentos, matrices de LED y LEDs multicolores. Se encuentran las pantallas de rejilla, las pantallas LCD de varias líneas y segmentos (LCD 1X16, LCD 2x16), pantallas gráficas GLCD (controladas por medio de pixeles), Pantallas LCD convencionales o táctiles, entre otras.

Dispositivos sónicos: Los dispositivos sónicos producen una señal auditiva, ubicada en la frecuencia audible humana para señalizar estados de peligro o señales de alerta o estados de proceso, utilizando para esto parlante o altavoces en todos sus tamaños, buzzer o Bips. También se utilizan en frecuencias ultrasónicas o subsónicas, utilizando para esto dispositivos específicos para la aplicación y frecuencia.

Dispositivos actuadores: Esta gama de dispositivos es más amplia, incluyendo los relevos capaces de controlar dispositivos de hasta varios cientos de voltios y/o amperios, motores de corriente alterna (CA-AC) como jaula de ardilla o devanados mediante TRIACs o relevos, motores de corriente continua (CC) mediante transistores o SCR, motores paso a paso como unipolares o bipolares mediante transistores, motores brush-less, servo motores, electro válvulas, solenoides y casi cualquier dispositivo que permita un control ON-OFF.

Ensamblador en los microcontroladores

Además del código binario, el ensamblador genera un conjunto de archivos adicionales, gracias a los cuales es posible controlar la evolución del proyecto. La extensión que acompaña al nombre del archivo indica el tipo de información que contiene.

Tabla 23. Archivos generados por ensamblador

Tipo de archivo	Extensión	Ejemplo
Código fuente del programa	ASM	programa.asm
Código binario ejecutable	HEX	programa.hex
Listado del programa	LST	programa.lst
Lista de errores	ERR	programa.err
Código objeto ejecutable	OBJ	programa.obj
Archivo de librería	LIB	programa.lib

Los archivos con extensión .ASM contienen el código fuente del programa, este archivo puede ser digitado dentro de los programas IDE suministrados por los fabricantes o puede ser editado en un editor de texto simple como el “WordPad”, es de considerar no colocar un nombre demasiado largo, la extensión se coloca generalmente de forma manual precedida de un punto (.asm o .ASM). Los nombres pueden ser una combinación de caracteres validos como letras, números y raya al piso, tenga presente que algunos sistemas no permiten extensión de nombre de más de ocho (8) caracteres. Los archivos de tipo HEX y OBJ contienen el código binario ejecutable y es derivado de archivos de tipo ASM. El proceso de ensamblado puede generar también archivos con extensiones como:

- LST con el contenido del programa fuente formateado, es decir, permite determinar la localización o posición de memoria donde se escribe la instrucción, la tabla de símbolos para la instrucción, la numeración de página, fecha y hora de ensamblado, una características que hace que el archivo LST sea particularmente importante es mostrar los errores o advertencias derivadas del código fuente.
- ERR, es un archivo donde se listan los errores y su localización en el programa, esto facilita el proceso de depuración y corrección del programa.

Términos utilizados en programación de microcontroladores

En la programación de microcontroladores como en todo sistema de aplicación informática, cuenta con términos o nombres particulares utilizados para referirse a partes del programa.

Registro: es un término genérico el cual hace referencia a una localidad de la memoria que permite almacenar información temporal, cada registro está formado por celdas las cuales tiene la capacidad de almacenar un bit. Generalmente se nombran los registros con la etiqueta “EQU” para identificarlos plenamente en el mapeo de memoria o en el programa.

Literal: es un valor constante, se podría pensar como en “letras” pero realmente es un valor expresado en un sistema numérico particular y bien definido, generalmente binario, octal, hexadecimal y decimal. Es posible asignar un valor numérico a un conjunto de letras mediante el uso de la etiqueta “EQU” o similar. Ejemplo VAL EQU 12H, asigna el valor “12” expresado en hexadecimal al conjunto de letras “VAL”.

Etiqueta: es un nombre con el cual se identifica una posición de memoria del microcontrolador, sirve para marcar puntos o subprogramas dentro del programa principal, las etiquetas se escriben en la primera columna de línea al borde izquierdo su longitud no debe sobrepasar los 31 caracteres, debe estar compuesta por caracteres como letras, números y raya al piso, se debe corroborar restricciones de tamaño, símbolos como la “ñ o Ñ” y comenzar con caracteres numéricos.

Instrucción: estas hacen referencia a operaciones básicas que pueden ejecutar un microcontrolador, se debe tener a la mano el set o lista de instrucciones validas junto con su sintaxis, operandos, operación, flags que afecta y ciclos maquina empleados.

Operando: son elementos utilizados por una instrucción, algunas instrucciones no utilizan operandos, otras más complejas utilizan uno y finalmente las mas complejas utilizan un máximo de dos. El primer operando, se denomina “operando fuente”, el segundo recibe el nombre de “operando destino”, la información entonces fluye desde el operando fuente al destino.

Comentario: los comentarios son bloques de texto que se digitán después de “;” para que el programador documente el programa, los comentario son ignorados por el ensamblador.

Encabezado: es el primer componente del programa donde se definen características o funcionalidades especiales que necesita el programa y/o el ensamblador, estas directivas se establecen al comienzo del programa y dependen del tipo de IDE y familia de microcontroladores que se este utilizando.

Ejemplo: "List p=16C84", "\$include 'jl3regsg.inc".

Definición de origen: sirve para indicar explícitamente el sitio de la memoria donde se comienza la escritura de la pila, memoria de programa, interrupciones, y subprogramas (subrutinas).

Instrucciones de programa: en la sección de instrucciones de programa se digitán en columna las instrucciones separando cada columna por un espacio contante generado por la tecla de tabulación "TAB", donde cada columna debe tener una estructura uniforme típicamente como sigue:

1. Etiqueta, nombre del programa, punto de llegada de salto, subprograma o subrutina.
2. Instrucción, es el "OPCODE" de la operación simple a realizar.
3. Operando fuente. Conformado por un registro, literal o constante.
4. Operando destino. Conformado por un registro, literal o constante, o destino de llegada del flujo de información.
5. Comentario, identificado por frases simples digitadas después de ";"

Subprograma: El subprograma es una división del programa principal, esta ejecuta instrucciones como respuesta a instrucciones de llamado o interrupción, debe devolverse después de terminar su secuencia de instrucciones mediante instrucciones de retorno.

Fin del programa: contiene una instrucción simple que le dice al ensamblador que en este punto termina el programa, es una palabra reservada por cada Entorno de Desarrollo Integrado para la familia de microcontroladores, por ejemplo "END" para el MPASM significa que es el final del programa.

Programación en los microcontroladores

Para determinar el camino más adecuado para desarrollar un proyecto con requiere del estudio del microcontroladores elegido tomando parámetros generales que se encuentran en hojas de especificadores o documentación especializada. A continuación se mencionan los parámetros generales y básicos a tener en cuenta en la fase de "análisis del problema".

Elementos: Son los componentes básicos y especiales que tiene el microcontrolador en particular, que serán utilizados en el desarrollo e implementación, entre ellos están la arquitectura, los registros especiales, los registros de propósito general, el registro de estado, la memoria de datos, la memoria de programa, la pila y los puertos.

Arquitectura: Es necesario y conveniente estudiar la arquitectura y disposición lógica del dispositivo, algunos se basan en la arquitectura Harvard y otros en la Von Neumann por lo que el formato y tamaño del set de instrucciones puede variar, además de sus modos de direccionamiento, tamaño del bus de datos y de programa, velocidad de proceso y ciclo máquina.

Registros de trabajo o acumulador: Son los registros que interactúan directamente con la CPU, su posición respecto a esta última determina la forma de operación de las instrucciones, la cantidad de instrucciones y la complejidad del programa.

Registro de estado: Es el registro que contiene los bits indicadores de estado o flags que muestran el estado del dispositivo o el resultado de la última operación, además son utilizados también para el mapeo de memoria.

Memoria de programa: Los microcontroladores cuentan con un segmento de memoria ROM donde se aloja el programa en código binario, este segmento de memoria tiene subdivisiones explícitas referidas al vector de reset, el cual determina el punto donde comienza el programa o el valor que debe contener el contador de programa (PC) para iniciar después del “reset”; el vector de interrupción, determina el punto donde el contador de programa atenderá la solicitud de interrupción.

Memoria de datos: Es una memoria compuesta por un conjunto de registros, completamente accesibles al programador con lo que se optimiza el programa, en estos registros se almacenan las “variables” del programa que están continuamente alterándose por la secuencia de las instrucciones, normalmente se le asignan nombres al comienzo del programa en una sección dedicada a la “definición de constantes y/o variables” utilizando por ejemplo la etiqueta y palabra reservada “EQU”. Dentro de la memoria de datos también se definen los registros de propósito especial los cuales alojan registros correspondientes a la matriz de registros de la CPU del microcontrolador, puertos, interrupciones y funciones especiales de comunicación, comparación o conversión.

Pila o Stack: Es un segmento de memoria con varios niveles dedicado al almacenamiento de la dirección de retorno de los llamados que se realizan a subrutinas dentro del programa principal. No se debe superar los niveles o profundidad de la pila, esto causa un desbordamiento y falla del programa. Lo anterior quiere decir que no deben anidarse llamados más allá del número de niveles de la pila.

Puertos: Los puertos son elementos con los que el microcontrolador establece comunicación con el exterior, la cantidad de puertos y el tamaño de los mismos definen los componentes conectados a ellos que enviaran información al microcontrolador o la recibirán. Es de notar que en la gran variedad del microcontroladores tiene los puertos de doble vía, sirven como entradas o salidas, también existen algunos que tienen varias funciones es decir, que multiplexan sus entrada/salidas para lo cual existen registros de propósito especial que deben ser configurados.

Subrutinas y Llamados

El programa fuente está compuesto por una sucesión de líneas de programa. Cada línea de programa está compuesta por 4 campos separados por uno o más espacios o tabulaciones. Estos campos son:

[Etiqueta]	Comando	[Operando(s)]	[;Comentario]
------------	---------	---------------	---------------

La etiqueta es opcional. El comando puede ser un mnemónico del conjunto de instrucciones. El operando está asociado al comando, si no hay comando no hay operando, e inclusive algunos comandos no llevan operando. El comentario es opcional para el compilador aunque es buena práctica considerarlo obligatorio para el programador.

La etiqueta: es el campo que empieza en la primera posición de la línea. No se pueden insertar espacios o tabulaciones antes de la etiqueta sino será considerado comando. Identifica la línea de programa haciendo que el compilador le asigne un valor automáticamente. Si se trata de una línea cuyo comando es una instrucción de programa del microcontrolador, se le asigna el valor de la dirección de memoria correspondiente a dicha instrucción (location counter). En otros casos se le asigna un valor de una constante, o la dirección de una variable, o será el nombre de una macroinstrucción, etc.

El comando: puede ser un código mnemónico de instrucción del microcontrolador, o una directiva o pseudoinstrucción para el compilador. En el primer caso será directamente traducido a código de máquina, en el segundo caso será interpretado por el compilador y realizará alguna acción en tiempo de compilación como ser asignar un valor a una etiqueta, etc.

Operando: el campo de parámetros puede contener uno o más parámetros separados por comas. Los parámetros dependen de la instrucción o directiva. Pueden ser números o literales que representen constantes o direcciones.

Comentario: el campo de comentario debe comenzar con un carácter punto y coma. No necesita tener espacios o tabulaciones separándolo del campo anterior,

e incluso puede empezar en la primera posición de la línea. El compilador ignora todo el texto que contenga la línea después de un carácter punto y coma. De esta manera pueden incluirse líneas que contengan solo comentarios, y es muy buena práctica hacer uso y abuso de esta posibilidad para que los programas resulten autodocumentados.

Importancia de las rutinas

La mayoría de los microcontroladores incluyen en su repertorio de instrucciones algunas que permiten saltar a una rutina y, cuando se complementa su ejecución, retornar al programa principal. El empleo de subrutinas aporta muchas ventajas entre las que se destacan las siguientes:

1. Se pueden escribir como subrutinas secciones de código y ser empleadas en muchos programas (por ejemplo, la subrutina de exploración de un teclado).
2. Dan a los programas un carácter modular, es decir, se pueden codificar diferentes módulos para usarlos en cualquier programa.
3. Se reduce notablemente el tiempo de programación, la detección de errores, usando repetidamente una subrutina.
4. El código es más fácil de interpretar, dado que las instrucciones de las subrutinas no aparecen en el programa principal. Solo figuran las llamadas CALLs.

Ramificaciones en programas con microcontroladores

La ramificación en programas con microcontroladores se refiere a los caminos u opciones que se presentan dentro de un programa, para dar implementar una estructura de decisión, denominada también como bifurcación. La forma como se generan las bifurcaciones en respuesta al pseudocódigo o algoritmo (en las opciones), es mediante las instrucciones orientadas a bits que permiten verificar el estado de un bit del registro de estado, un pin del puerto de entrada o de un registro en particular, usualmente estas instrucciones incorporan saltos normalmente de una línea, es decir, en caso que al verificar el bit del registro en prueba cumpla con un estado en particular (como 0 o 1), se genera un salto de una línea, de lo contrario no ocurre salto. Esto quiere decir que se debe utilizar instrucciones de salto para enviar al programa a subrutinas específicas para el estado seleccionado.

Otra forma de generar bifurcaciones o ramificaciones es mediante la comparación de todo el registro con un valor determinado, arrojando un posible resultado, que puede ser igual, mayor o menor, de manera que afecta un “flag” o bandera del

registro de estado y de esta forma se puede nuevamente evaluar el estado del bit para tomar uno de los dos caminos mediante las instrucciones de salto.

Es posible implementar ramificaciones más estructuradas utilizando el método de tablas, que consiste en alterar el valor del registro Contador de Programa (PC), ya sea en su parte baja (PCH), que hace referencia a los primeros 8 o más bits (B0 – B7 o más) o en todo el registro Contador de Programa, con lo que se tiene un direccionamiento a posiciones de memoria particulares en donde estén ubicados valores de retorno para ser tratadas como opciones o donde se ubiquen llamados a subrutinas específicas. Con la posibilidad de programación de microcontroladores en lenguajes de más alto nivel como BASIC o C, se pueden implementar estructuras predefinidas en el lenguaje de más alto nivel para ensamblador, como las IF, SWITCH o CASE, aunque con el correspondiente aumento del consumo de espacio de memoria necesario para el programa en el micro.

Programación de microcontroladores en C_basic

Es frecuente encontrar entornos de desarrollo y soporte para lenguajes de alto nivel en microcontroladores, se tiene la característica ventaja de poder programar de manera muy fácil y sencilla el microcontrolador, debido a que el lenguaje de alto nivel trae preestablecido código ensamblador para las estructuras más usuales (IF, WHILE, FOR, SWITCH, CASE), junto con código para incorporar los periféricos compatibles con el micro (LCD, Teclado matricial, GLCD, etc). Como se puede inferir esto trae comodidades al editar el código fuente, pues para muchos es muy habitual el uso de software como C y BASIC, pero trae inconveniente propios principalmente que se limita el poder utilizar todo el potencial del micro y sus módulos internos.

Los nuevos microcontroladores utilizan lenguajes de alto nivel para su programación, esto coloca al alcance de muchos usuarios el manejo y utilización de microcontroladores en proyectos de desarrollo y soluciones. Esto hace que el desarrollador requiera tener la posibilidad de instalar una herramienta de software y el hardware particular a cada fabricante y micro, de manera que pueda programarlo, esto requiere además un gran compromiso de aprendizaje autónomo constante y de intercambio de conocimiento en grupo colaborativo para poder tener un buen conocimiento del manejo y programación de varias familias de micros.

Otras características que pueden ser inconvenientes para el uso de lenguajes de alto nivel son que cada fabricante o desarrollador incorpora encabezados y

directivas particulares, por lo que se debe estudiar el entorno de desarrollo particular para cada familia y dispositivo. Al tener preestablecido directivas de control sobre periféricos, se tienen por defecto asignado determinados pines o patillas del micro para cada función, limitando la multiplexación de los pines del micro causando que debamos buscar un micro con más pines para poder implementar adecuadamente el diseño, puede influir sobre el diseño de conexión estructural del micro y periféricos ocasionando que se deba alterar el diseño, e posible modificar dicha librería aunque puede ocasionar algunas dificultades al correr el programa para depurarlo.

Lección 19: Familias de microcontroladores

Existen muchas familias fabricantes de microcontroladores, entre las más comunes están:

Atmel (AVR), Hitachi (H8), Intel de 8 bits (8XC42, MCS51, 8xC251) o Intel de 16 bits (MCS96, MXS296), National Semiconductor (COP8), Microchip, Motorola de 8 bits (68HC05, 68HC08, 68HC11) o de 16 bits (68HC12, 68HC16) o de 32 bits (683xx), NEC (78K), Texas Instruments (TMS370) y Zilog (Z8, Z86E02).

Sin embargo en nuestro medio se destacan sólo dos de ellas: la empresa Motorola y la empresa Microchip, no obstante se exponen generalidades de las principales y más conocidas.

Microcontroladores más comunes

Los microcontroladores más comunes para propósito de aprendizaje, instrucción e implementación de soluciones son, los PIC de Microchip muy utilizados por su simplicidad en el aprendizaje inicial, los MSP de Texas Instruments por su costo bajo en las herramientas de desarrollo integrado, se encuentran también los Motorola Freescale que aunque tienen una relación de costo en micros y entornos de desarrollo más altos, son muy aplicados a soluciones en automóviles y la industria. Como una gama de micros más avanzada está los micros Basic Stamp utilizados para desarrollos de robótica y finalmente los AVR con un potencial significativos para aplicaciones de robótica y control. Aun se utilizan los Intel en el

aprendizaje e implementación pero el costo de los dispositivos de programación hacen que se seleccione otros fabricantes y dispositivos.

Dentro del curso se enfoca el trabajo de aprendizaje inicial con microcontroladores Microchip de 8 bits, con el PIC16F84 representa la arquitectura Harvard y RISC en el conjunto de instrucciones, por su simplicidad en arquitectura y función de pines, pudiendo ser configurado rápidamente y realizar implementaciones de prueba rápidas y de bajo costo. El entorno de desarrollo utilizado MPLAB es de uso libre, existe mucha documentación en internet incluyendo programadores, software de programación, documentación técnica y ejemplos de aplicación. Para una aplicación más avanzada se implementa con el PIC16F877, debido a que su configuración es relativamente simple y posee módulos Timer, ADC, Comparador y puertos de comunicación serie y paralelo.

Como otro dispositivo para el curso se plantean ejercicios con los microcontroladores MSP de Texas Instruments, de 16 bits, básicamente los micros bajo el entorno de desarrollo integrado (IDE) denominado MSP430 LaunchPad, estos micros representan la arquitectura Von Neumann, con un conjunto de instrucciones CISC, tiene un bajo costo la adquisición del LauncPad MSP430 junto con sus integrados más representativos como el MSP430G2231, MSP430G2211, entre otros. Estos micros se caracterizan por su ultra bajo consumo, incorporan modulos como Timers, ADC, PWM, Comparadores, comunicación serial, sensor de temperatura, etc.

Adicionalmente se utilizan micros también de arquitectura Von Neumann, de la familia HC08 de 8 bits, de bajo costo y alto desempeño específicamente con los MC68H(R)C908JL3/JK3/JK1. Estos micros se caracterizan en el aprendizaje por su simplicidad de configuración y relativa fácil programación, tienen módulos Timers, ADC, con una cantidad apreciable de pines disponibles para I/O (Entrada/Salida), 23 pines I/O en el JL3 y 15 pines I/O en el JK3/JK1.

Teniendo en cuenta la cantidad de fabricantes y familias en el mercado se tratan aspectos generales tanto de los micros en estudio objeto de este curso como en otros microcontroladores, se espera que el lector profundice en cada una de los fabricantes, familias y micros, utilizando fuentes documentales en internet, en bibliotecas virtuales, bases de datos, foros, hojas técnicas y material bibliográfico físico. Es importante recalcar que este texto es un “manual” de referencia para múltiples aspectos de aplicación y práctica, soportado por una variedad de referencias documentales y digitales que se espera sean consultadas para profundizar tanto en teoría y aplicación práctica.

Familia de microcontroladores Microchip – PIC

Esta familia, desarrollada por la casa Microchip, se divide en varias gamas: enana, baja, media y alta. Las principales diferencias entre estas gamas radica en el número de instrucciones y su longitud, el número de puertos y funciones, lo cual se refleja en el encapsulado, la complejidad interna y de programación, y en el número de aplicaciones.

Gama enana: Su principal característica es su reducido tamaño, al disponer todos sus componentes de 8 pines. Se alimentan con un voltaje de corriente continua comprendido entre 2,5 V y 5,5 V, y consumen menos de 2 mA cuando trabajan a 5 V y 4 MHz. El formato de sus instrucciones puede ser de 12 o de 14 bits y su repertorio es de 33 o 35 instrucciones, respectivamente. En la Figura se muestra el diagrama de pines de uno de estos PIC.

Figura 80. PIC gamma baja o enana PIC12Cxxx⁹⁹

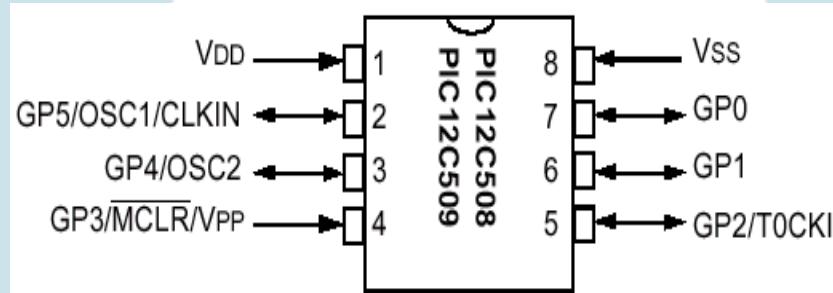


Tabla 24. Gamma Enana PIC¹⁰⁰

Modelo	Mem. de programa	Mem. de datos	Frecuencia	Líneas de E/S	ADC de 8 bits	Temporizador
PIC12C508	512x12	25x8	4MHz	6	---	TRM0+WDT
PIC12C509	1024x12	41x8	4MHz	6	---	TRM0+WDT
PIC12C670	512x14	80x8	4MHz	6	---	TRM0+WDT
PIC12C671	1024x14	128x8	4MHz	6	2	TRM0+WDT
PIC12C672	2048x14	128x8	4MHz	6	4	TRM0+WDT
PIC12F680	512x12 FLASH	80x8 16x8 EEPROM	4MHz	6	4	TRM0+WDT
PIC12F681	1024x14 FLASH	80x8 16x8 EEPROM	4MHz	6	4	TRM0+WDT

⁹⁹ Extraído el 18 Octubre de 2009 desde <http://www.ufps.edu.co/materias/ucontrol/htdocs/conte/usuarios.lycos.es/sfriswolker/pic/cuatro.htm>

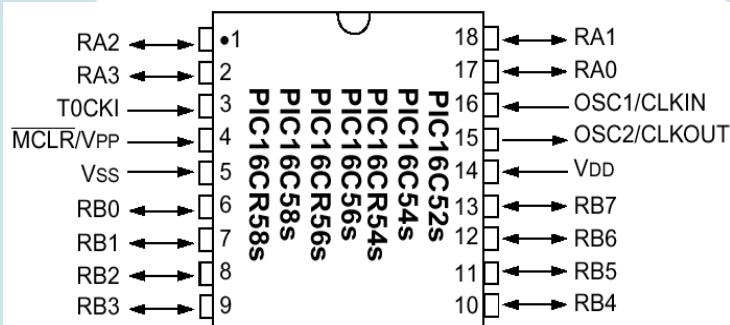
¹⁰⁰ Extraído el 18 Octubre de 2009 desde <http://www.ufps.edu.co/materias/ucontrol/htdocs/conte/usuarios.lycos.es/sfriswolker/pic/cuatro.htm>

Aunque los PIC enanos sólo tienen 8 pines, pueden destinar hasta 6 como líneas de E/S para los periféricos porque disponen de un oscilador interno R-C, lo cual es una de su principales características.

En la tabla 20 se presentan las principales características de los modelos de esta subfamilia. En los modelos 12C5xx el tamaño de las instrucciones es de 12 bits; mientras que en los 12C6xx sus instrucciones tienen 14 bits. Los modelos 12F6xx poseen memoria Flash para el programa y EEPROM para los datos.

Gama baja: Se trata de una serie de PICs de recursos limitados, pero con una de las mejores relaciones costo/prestaciones. Sus versiones están encapsuladas con 18 y 28 pines y pueden alimentarse a partir de una tensión de 2,5 V, lo que los hace ideales en las aplicaciones que funcionan con pilas, teniendo en cuenta su bajo consumo (menos de 2 mA a 5 V y 4 MHz). Tienen un repertorio de 33 instrucciones cuyo formato consta de 12 bits. En la Figura se muestra el diagrama de pines de uno de estos PICs.

Figura 81. *PIC gamma media PIC16Cxx¹⁰¹*



Al igual que todos los miembros de la familia PIC16/17, los componentes de la gama baja se caracterizan por poseer los siguientes recursos: Sistema Power On Reset, Perro guardián (*Watchdog* o WDT), Código de protección, etc. Sus principales desventajas o limitaciones son que la pila sólo tiene dos niveles y que no admiten interrupciones.

En la siguiente tabla se presentan las principales características de los modelos de esta subfamilia.

¹⁰¹ Extraído el 18 Octubre de 2009 desde <http://www.ufps.edu.co/materias/ucontrol/htdocs/conte/usuarios.lycos.es/sfriswolker/pic/cuarto.htm>

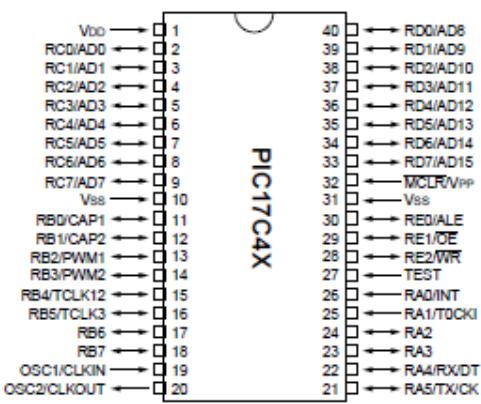
Tabla 25. Gamma Baja PIC¹⁰²

Modelo	Mem. de programa	Mem. de datos	Frecuencia	Líneas de E/S	Temporizador	pines
PIC16C52	384	25 bytes	4 MHz	12	TRM0+WDT	18
PIC16C54	512	25 bytes	20 MHz	12	TRM0+WDT	18
PIC16C55	512	24 bytes	20 MHz	20	TRM0+WDT	28
PIC16C56	1K	25 bytes	20 MHz	12	TRM0+WDT	18
PIC16C57	2K	72 bytes	20 MHz	20	TRM0+WDT	28
PIC16C58A	2K	73 bytes	20 MHz	12	TRM0+WDT	18

Gama media: Es la gama más variada y completa de los PIC. Abarca modelos con encapsulado desde 18 pines hasta 68, cubriendo varias opciones que integran diversos periféricos. En esta gama sus componentes añaden nuevas prestaciones a las que poseían los de la gama baja, haciéndoles más adecuados en las aplicaciones complejas. Admiten interrupciones, poseen comparadores de magnitudes analógicas, convertidores A/D, puertos serie y diversos temporizadores.

El repertorio de instrucciones es de 35, compatible con el de la gama baja. Sus distintos modelos contienen todos los recursos que se precisan en las aplicaciones de los microcontroladores de 8 bits. También dispone de interrupciones y una pila de 8 niveles que permite el anidamiento de subrutinas.

Figura 82. PIC gamma alta PIC17CXX¹⁰³



¹⁰² Extraído el 18 Octubre de 2009 desde <http://www.ufps.edu.co/materias/ucontrol/htdocs/conte usuarios.lycos.es/sfriswolker/pic/cuatro.htm>

¹⁰³ Extraído el 18 Octubre de 2009 desde <http://ww1.microchip.com/downloads/en/DeviceDoc/30412c.pdf>

En la siguiente tabla se presentan las principales características de algunos de los modelos de esta familia.

Tabla 26. Gamma Media PIC¹⁰⁴

Modelo	PINES	I / O	EPROM	RAM	Interrup	Voltaje (V)
PIC16C61	18	13	1Kx14	36x8	3	3.0-6.0
PIC16C62	28	22	2Kx14	128x8	10	2.5-6.0
PIC16C63	28	22	4Kx14	192x8	10	3.0-6.0
PIC16C64	40	33	2Kx14	128x8	8	3.0-6.0
PIC16C65	40	33	4Kx14	192x8	11	3.0-6.0
PIC16C620	18	13	512x14	80x8	4	3.0-6.0
PIC16C621	18	13	1Kx14	80x8	4	3.0-6.0
PIC16C622	18	13	2Kx14	128x8	4	3.0-6.0

Modelo	PINES	I / O	EPROM	RAM	Interrup	Canales A / D
PIC16C70	18	13	512x14	36x8	4	4 canales
PIC16C71	18	13	1Kx14	36x8	4	4 canales
PIC16C72	28	22	2Kx14	128x8	8	5 canales
PIC16C73	28	22	4Kx14	192x8	11	5 canales
PIC16C74	40	33	4Kx14	192x8	12	8 canales

Modelo	Mem. de programa RAM / EEPROM	Mem. de programa	Mem. de datos	INT	Líneas de E/S	Temporizador
PIC16F83	36	64	512X14 FLASH	25 bytes	4	13 TRM0+WDT
PIC16C84	36	64	1KX14 EEPROM	25 bytes	4	13 TRM0+WDT
PIC16F84	68	64	1KX14 FLASH	25 bytes	4	13 TRM0+WDT

¹⁰⁴ Extraído el 18 Octubre de 2009 desde
<http://www.ufps.edu.co/materias/ucontrol/htdocs/conte/usuarios.lycos.es/sfriswolker/pic/cuatro.htm>

Gama alta: PIC17CXXX: En esta gama se alcanzan las 58 instrucciones de 16 bits en el repertorio y sus modelos disponen de un sistema de gestión de interrupciones vectorizadas muy potente. También incluyen variados controladores de periféricos, puertos de comunicación serie y paralelo con elementos externos, un multiplicador hardware de gran velocidad y mayores capacidades de memoria, que alcanza los 8K palabras en la memoria de instrucciones y 454 bytes en la memoria de datos.

Quizás la característica más destacable de los componentes de esta gama es su arquitectura abierta, que consiste en la posibilidad de ampliación del microcontrolador con elementos externos. Para este fin, los pines comunican al exterior las líneas de los buses de datos, direcciones y control, a las que se conectan memorias o controladores de periféricos. Esta facultad obliga a estos componentes a tener un elevado número de pines comprendido entre 40 y 44. Esta filosofía de construcción del sistema es la que se empleaba en los microprocesadores y no suele ser una práctica habitual cuando se emplean microcontroladores.

En la siguiente tabla se presentan las características más relevantes de los modelos de esta gama, que sólo se utilizan en aplicaciones espaciales.

Tabla 27. Gamma Alta PIC¹⁰⁵

Modelo	CAP	PWM	Multiplica hardware	Mem. de programa	Mem. de datos RAM	Líneas de E/S	Temp.	pines
PIC17C42A	2	2	8X8	2KX16	232	33	4+WDT	18
PIC17C43	2	2	8X8	4KX16	454	33	4+WDT	18
PIC17C44	2	2	8X8	8KX16	454	33	4+WDT	18
PIC17C52	4	1	8X8	8KX16	454	50	4+WDT	18
PIC17C56	4	1	8X8	16KX16	902	50	4+WDT	28

La literatura sobre PIC es extensa, se encuentra material aceptable en la red y en librerías especializadas.

¹⁰⁵ Extraído el 18 Octubre de 2009 desde <http://www.ufps.edu.co/materias/ucontrol/htdocs/conte/usuarios.lycos.es/sfriswolker/pic/cuarto.htm>

Familia de microcontroladores Motorola Freescale

Esta familia, desarrollada por la casa Motorola, se divide en las siguientes subfamilias:

Familia HC05: Esta familia es una de las más utilizadas en la gran mayoría de aplicaciones por su versatilidad de recursos y fácil programación. Sin embargo, presenta una propiedad con mayor importancia y es su compatibilidad con familias más avanzadas, por ejemplo con la familia HC08, lo que permite hacer migración de diseños hacia dispositivos de más alto rendimiento de una manera muy fácil y rápida. Sus principales ventajas son:

- Un timer robusto
- Memoria EEPROM de 256
- Memoria de programa desde 4k hasta 32 k
- Memoria RAM desde 176 hasta 528 bytes.
- Ocho canales A/D
- Comunicación serial síncrona y asíncrona.

(VESGA, 2007).

Tabla 28. Familia Motorola¹⁰⁶

Familias 68HC05-C y 68HC05-D	<ul style="list-style-type: none">• Timer de 16 bits, acompañado de módulo de captura y comparación• Memoria de programa desde 4k a 16k• Comunicación SCI (75Hz-131KHz)• Interfaz SPI 4 hilos• Watchdog
Familias 68HC05-J y 68HC115-K	<ul style="list-style-type: none">• Bajo costo• Encapsulado de 16 y 20 pines• Memoria de programa 0,5K a 2K• Memoria RAM 32 hasta 128 bytes
Familia 68HC05-P	<ul style="list-style-type: none">• Controlador para manejo de LCD• Memoria de programa 0,6K hasta 24 K• Memoria RAM 32 hasta 768 bytes• Timer de 16 bits con módulos de captura y comparación.• Comunicación serial síncrona y asíncrona.

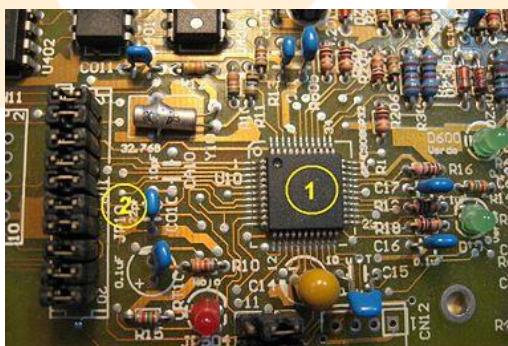
Familia HC08: Son microcontroladores de propósito general. Cada miembro de esta familia cuenta con diferentes periféricos internos, pero con una CPU común que permite migrar aplicaciones entre ellos, facilitando con ello el diseño.

El 68HC08 es un microcontrolador de 8 bits y arquitectura Von Neumann, con un solo bloque de memoria. Es conocido también simplemente por HC08. Entre los

¹⁰⁶ VESGA, 2007

periféricos internos con los que cuentan estos microcontroladores, están: conversores analógicos-digitales, módulo de control de tiempos y sistemas de comunicación como SPI, I²C, USB o SSCI entre otros.

Figura 83. *Microcontrolador Motorola Freescale*¹⁰⁷



Familia 68HC11: (abreviado HC11 o 6811), Es una familia de microcontroladores de Motorola, derivada del microprocesador Motorola 6800. Los microcontroladores 68HC11 son más potentes y costosos que los de la familia 68HC08 y se utilizan en múltiples dispositivos empotrados. Siguen la arquitectura Von Newman. Internamente, el conjunto de instrucciones de la familia 68HC11 es compatible con la de la mayoría de sus predecesores. La familia 68HC11 emplea instrucciones de longitud variable y se considera que emplea una arquitectura CISC. Tienen dos acumuladores de ocho bits (A y B), dos registros índice de 16 bits (X e Y), un registro de banderas, un puntero de pila y un contador de programa.

Los 68HC11 tienen cinco puertos externos (A, B, C, D y E), cada uno de ocho bits excepto el E, que es generalmente de seis bits. El puerto A se emplea en captura de eventos, salida comparada, acumulador de pulsos y otras funciones de reloj. El puerto D para E/S serie y el puerto E como conversor analógico-digital. La familia 68HC11 puede funcionar tanto con memoria interna o externa. En caso de emplear memoria externa, los puertos B y C funcionan como bus de datos y direcciones respectivamente. Últimos Microcontroladores de la Familia Freescale.

- Familia “ultra bajo costo” RS08, con sus modelos **MC9RS08KA2** y **MC9RS08KA1**.
- Dispositivos de la familia HC908 con capacidades de memoria hasta de 16k, entre los cuales están. **MC68HC908QTxA/QYxA**, **MC68HC908QLxx**, **MC68HC908QC16xx**, **MC68HC908GRxxA**, **MC68HC908QB8** y **MC68HC908JL16**.
- Existen novedades en la familia **HC9S08**, de bajo consumo con los dispositivos **MC9S08QG8/4** y **MC9S08AWxx**.

¹⁰⁷ Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Archivo:Microcontrolador_HC08_en_Impreso_Comentado_V1.JPG

Figura 84. HC08¹⁰⁸

La literatura existente que trata el tema de los microcontroladores Motorola Freescale, es muy extensa, generalmente en inglés, este material está basado en una fuente en español, se podría asegurar que casi la única en su estilo, como texto complementario y especializado en este tema está el libro del Ingeniero Juan Carlos Vesga titulado “Microcontroladores Motorola Freescale: programación, familias y sus distintas aplicaciones en la industria” por lo que se sugiere al estudiante su consecución para complementar estos conocimientos.

Familia de microcontroladores Intel MCS

El primer microcontrolador en esta familia fue el 8048, en su interior se alojaba una memoria RAM pero el programa se debía almacenar en un dispositivo externo, unos años más tarde se desarrolla el 8051 la cual es la piedra angular de una serie de dispositivos con características especiales para aplicaciones específicas. Las versiones existentes de esta familia se basan en el núcleo del 8051, de ahí que se tomó como referencia en la denominación oficial de Intel para la familia de microcontroladores basados en el 8051, esta denominación es MCS51.

El 8051 se caracteriza por tener 4K de memoria ROM, posteriormente se implementa el 8751 con una memoria EPROM dando la posibilidad de la reprogramación, borrando el dispositivo de memoria por exposición a luz ultravioleta. La característica más sobresaliente de estos dispositivos es la capacidad de expansión de memoria, es decir, tienen puertos habilitados para direccionar hasta 64K de memoria externa RAM y ROM esta última con la capacidad de almacenar el programa de control.

El núcleo del 8051 es usado en más de 100 tipos de microcontroladores por más de 20 fabricantes (Atmel, Dallas Semiconductor, Philips, etc).

Figura 85. Familia MSC51¹⁰⁹



En el mercado se encuentran versiones de microprocesadores como el 8086 y 8088 que permiten aprovechar las herramientas desarrolladas para PC, estos microcontroladores son el 80186, 80188 y 80386EX.

Memoria

Estos dispositivos a diferencia de los microprocesadores, tienen un espacio para las direcciones de datos tanto para la lectura como para la escritura y otro para las direcciones de programa, es decir, emplea una arquitectura Harvard. La capacidad de direccionamiento de memoria llega hasta los 64K, de los cuales en versiones ROM, EPROM y EEPROM, los 4K, 8K o 16K inferiores son alojados en el microcontrolador. El microcontrolador tiene la memoria interna dividida en dos partes:

- SFR, (Special Function Registers), son los registros proporcionados por el microcontrolador, y tienen asignadas direcciones en esta memoria interna.
- Memoria de propósito general.

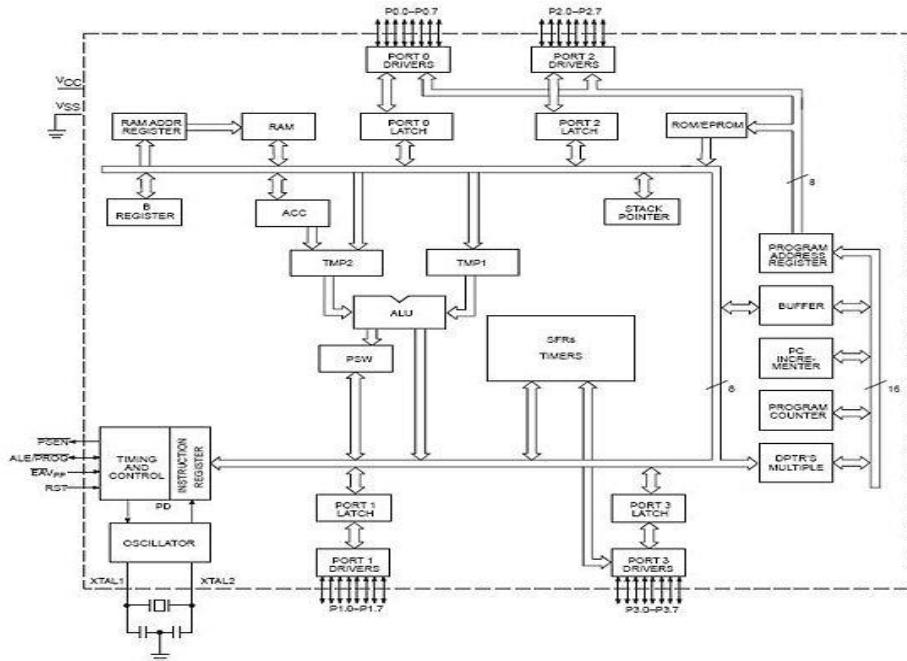
El acceso a esta memoria interna es más rápido que el acceso a la memoria externa, pero es de tamaño limitado. Parte de esta memoria interna además se usa como pila durante las llamadas a función y el proceso de interrupciones.

Actualmente están disponibles versiones del 8051 con memoria de tipo FLASH, lo que permite una programación rápida y con una capacidad mayor de reprogramaciones, se evidencia la complejidad de su programador por lo que se

¹⁰⁹ Extraído el 18 Octubre de 2009 desde http://es.wikipedia.org/wiki/Archivo:KL_Intel_P8051.jpg

hace difícil implementar estos proyectos al incluir costos adicionales del programador y paquetes de programación.

Figura 86. Arquitectura interna MSC51¹¹⁰



Características:

- Permite operaciones a nivel de bit, por la inclusión de una unidad booleana.
- Tiene cuatro conjuntos separados de registros.
- Pueden incluir una o dos UARTs, puerto Transmisor-Receptor Asíncrono Universal.
- Inclusión de dos o tres temporizadores.
- 128 a 256 de RAM interna
- 0K y 54K de memoria de programa.
- Watchdog.
- Compatibilidad con I²C, SPI, USB.
- Generadores PWM
- Conversor A/D y D/A.

¹¹⁰ Extraído el 18 Octubre de 2009 desde http://es.wikipedia.org/wiki/Archivo:Diag_bloques_8051.JPG

Conjunto de instrucciones: Todos los miembros de la familia ejecutan las mismas instrucciones optimizando aplicaciones de control 8 bits, con capacidad de varios modos de direccionamiento, soporte para control de programa basados en operaciones de un (1) bit.

Las instrucciones se dividen en instrucciones aritméticas, instrucciones lógicas e instrucciones de transferencia de datos. No se hace profundiza en el estudio y aplicaciones con este dispositivo dado la complejidad y costo del sistema de desarrollo (programador). En este curso se hará énfasis en los microcontroladores de la familia Motorola Freescale y Microchip PIC.

Familia de microcontroladores Atmel AVR

Esta empresa maneja microcontroladores basados en arquitectura RISC, las CPUs llegan hasta 32 bits, existen varios grupos de microcontroladores:

- Microcontroladores basados en el 8051 Intel, incorporan una memoria de programa Flash.
- Microcontroladores AT91, soportan compilaciones en C, emulator.
- Microcontroladores AVR, con arquitectura RISC y CPU de 8 bits, incorpora módulos USART, SPI, ADC, etc. Implementado sobre arquitectura Harvard.

Figura 87. Familia ATMEL¹¹¹



Familia de microcontroladores Basic Stamp

Esta familia producida Parallax, es un sistema digital construido sobre una tarjeta impresa, con chips de montaje superficial, su núcleo inicialmente ha sido un

¹¹¹ Extraído el 18 Octubre de 2009 desde <http://es.wikipedia.org/wiki/Archivo:ATMEL-AT90S2333.jpg>

microcontrolador PIC para el Basic Stamp I, posteriores versiones del Basic Stamp II tienen un núcleo PIC o SCENX. Utilizan un lenguaje de programación propio similar al BASIC denominado PBASIC, lo cual lo hace relativamente fácil de programar, por lo cual ha tenido gran auge en colegios y universidades en proyectos y aprendizaje sin necesidad de tener conocimientos avanzados en electrónica digital, microprocesadores y microcontroladores. El sistema digital en placa impresa tiene como intérprete de instrucciones PBASIC un microcontrolador PIC o SCENIX, una memoria serial EEPROM donde se almacena el programa, regulador de voltaje y un circuito de Reset. Tienen la limitante que ejecuta operaciones a un promedio más lento en relación con otros microcontroladores, puesto que debe interpretar las instrucciones PBASIC que representan varias líneas de ensamblador, haciendo la ejecución relativamente más lenta.

Basic Stamp I: implementado con un microcontrolador PIC16C56, con 14 pines, en donde 8 son pines I/O, un pin de Reset, una pin de entrada de alimentación con fuente regulada de 5V, un pin de entrada de alimentación con fuente no regulada de 6V a 15V, un pin de tierra (GND) y los pines de programación hacia el PC. Implementa 256 posiciones de memoria de programa, 16 bytes de memoria RAM en donde se aloja el control de los pines I/O y variables del programa. Las instrucciones PBASIC permiten hacer ciclos condicionales, lectura de pulsadores o botones, generación de sonidos, medición de tiempo, generación de pulso, entre otras con lo que disminuye el tiempo y complejidad en el desarrollo de una aplicación.

Basic Stamp II: Inicialmente posee como núcleo un PIC16C57 aunque después se implementa con otros PIC e incluso con otros micros como el SCENIX SX48AC, incorpora módulos similares al Basic Stamp I, pero más avanzados. Se presenta como un “Integrado” de doble línea de pines, con 28 pines de los cuales 16 son pines I/O, pin para entrada de fuente regulada de 5 V, pin para entrada de fuente no regulada entre 6V y 15V, pin de Reset y pines de comunicación serial para ser programado desde un PC. El modelo Basic Stamp II implementa una memoria con 2048 de memoria de programa EEPROM y 32 Bytes en la memoria RAM. El set de instrucciones PBASIC incorpora instrucciones especiales como las necesarias para manejo de tonos DTMF, PWM, comunicación serial (RS-232), entre otros.

Familia de microcontroladores Arduino

Arduino es un conjunto de software y hardware, diseñado para facilitar la implementación de proyectos de electrónica, se compone de un entorno de desarrollo integrado IDE, junto con una placa (hardware libre) en donde se integra

el programador, el microcontrolador, inicialmente un AVR y más tarde se implementaron con Cortex ARM de 32 bits. El IDE utiliza el lenguaje de programación Processing/Wiring junto con un cargador de arranque o “bootloader”, aunque se han utilizado otros lenguajes en la programación como JAVA, Python, C, entre otros, la potencia de Arduino radica en poder desarrollar objetos interactivos autónomos o vinculados al PC del tipo Flash.

Existen varias versiones de la placa entre ellas Duemilanove, Diecimila, Mega, Nano entre otras. Debido a que C es la base de Arduino, todas las funciones de C y algunas de C++ son soportadas, por tanto la programación es muy similar a utilizar lenguaje C.

Familia de microcontroladores ARM

Los microcontroladores ARM son un producto y compañía derivada de Acorn Computers, basado en una arquitectura RISC de 32 bits, aunque ARM no produce directamente los ARM, sin embargo ha diseñado el núcleo y ha licenciado a compañía como Texas Instruments, Motorola Freescale, Atmel, entre otras, para la fabricación de los chips, las cuales conservan el núcleo y agregan módulos para potenciar los micros como conversores análogos / digitales, puertos de comunicación (UART, SPI, etc). Lo que ha significado una estrategia que lo ha posicionado comercialmente.

Familia de microcontroladores PICAXE

Son sistemas similares al Basic Stamp, es decir, son placas con núcleo PIC de Microchip, desarrollado por la empresa Inglesa denominada Revolution Education LTD. El PICAXE está compuesto por un microcontrolador PIC, un código o interfaz de programación “Bootstrap Code” desarrollado por la empresa Revolution Education, permitiendo una programación simple y directa en el circuito o programación “In Circuit”, es decir, incorpora un programador básicamente serial, en tercera instancia posee un editor simple y gratuito que maneja lenguaje de programación en diagrama de flujo y lenguaje BASIC.

Se caracteriza por un precio accesiblemente económico, su programación en BASIC y gráfica con diagrama de flujo, la arquitectura es simple y flexible logrando realizar implementaciones fácilmente, tiene gran variedad de instrucciones que permiten un manejo directo de señales y funciones especiales respecto a las entradas y salidas digitales, análogas/digitales (A/D), sensores, comunicaciones, retardos, etc. Como punto en contra tiene una memoria limitada por lo que se utiliza en aplicaciones pequeñas.

Familia de microcontroladores PSoC

Programable System on Chip, desarrollado por la empresa Cypress Semiconductor, son microcontroladores versátiles, que permiten ser configurados completamente, se presentan en un chip con familias distinguibles por sus núcleos, la familia CY8C2 denominado PSoC1 con núcleo M8C velocidad de hasta 24 MHz, familia CY8C3 denominado PSoC3 con núcleo 8051 de ciclo único y familia CY8C5 denominado PSoC 5 con núcleo de 32 bits ARM Cortex-M3.

Su entorno de programación IDE es gráfico denominado PSoC Designer para el PSoC1, con un modo de edición “Chip design”, basado en editor y compilador en C, y modo “System design” de interfaz gráfica que funciona pegando funciones gráficas. Para los PSoC3 y PSoC5 se tiene la herramienta “PSoC Creator” con entorno gráfico de programación y kernel “Keil” para PSoC3 y “ARM” para PSoC5.

Lección 20: Herramientas y sistemas de desarrollo para microcontroladores

Entornos de Desarrollo Integrado – IDE para microcontroladores

Microchip y otras empresas ofrecen varias herramientas de desarrollo que permiten y facilitan la elaboración y depuración de los programas, las herramientas de desarrollo de microchip operan en el entorno de programación de MPLABIDE (IDE-“Entorno de Desarrollo Integrado”), algunas de las herramientas más usuales se dividen en:

- Generación de código.
 - MPASM, MPLAB-C y MP-DriveWay.
- Depuración de los programas.
 - PICMASTER y ICEPICt emuladores en circuito (In-Circuit) y MPLABSIM como simulador de programas.
- Dispositivos programadores.
 - PICSTART Plus y el PROMATE II de Microchip
 - ICPCROG con varios programadores de libre distribución (Pro-Pic Program).

Los anteriores dispositivos permiten almacenar el programa en la memoria del microcontrolador, para esto utilizan usualmente los puertos serie o paralelo de la computadora personal usando un programa especialmente diseñado para enviar el código máquina ejecutable hacia la memoria del microcontrolador, estos programadores en sus versiones más sofisticadas y

su respectivo hardware permiten seleccionar la referencia del dispositivo a programar.

Tarjetas de evaluación de los productos y sistemas de desarrollo. Las tarjetas de demostración permiten probar y evaluar las prestaciones de cada dispositivo, son ampliamente utilizadas en laboratorios de enseñanza y para práctica de usuarios aprendices. Los sistemas de desarrollo son combinaciones de hardware y software que permiten realizar todo el proceso de desarrollo de un prototipo desde la edición, compilación, depuración, y grabación del programa en la memoria del micro, además permite establecer las conexiones requeridas con los periféricos de entrada-salida dispuestos sobre la tarjeta, cuando se llega al punto donde el sistema funciona adecuadamente se procede a la construcción del sistema final sobre una tarjeta de circuito impreso para la fabricación en serie.

Mplab IDE.

En los proyectos que se describen en esta unidad se hace uso del software de uso libre como MPLAB, El motivo es que hemos de entender y aprender a utilizarlo para ensamblar nuestros proyectos, hacerlo fácil y esto es lo que vamos a describir aquí. Las instrucciones presentadas son generales y pueden variar en la localización de las opciones es recomendable hacer una exploración completa antes de comenzar los ejercicios. La última versión de MPLAB IDE se puede descargar en forma gratuita desde la página de Microchip.

El Organizador de Proyectos (Project Manager): El organizador de proyectos (Project Manager) es parte fundamental de MPLAB. Sin crear un proyecto Usted no puede realizar depuración simbólica. Con el Organizador de Proyectos (Project manager) puede utilizar las siguientes operaciones:

- Crear un proyecto.
- Agregar un archivo de programa fuente de proyecto.
- Ensamblar o compilar programas fuente.
- Editar programas fuente.
- Reconstruir todos los archivos fuente, o compilar un solo archivo.
- Depurar su programa fuente.

Software ensamblador: El software ensamblador que presenta Microchip viene en dos presentaciones, una, para entorno DOS llamado MPASM.EXE y la otra, para entorno Windows llamado MPASMWIN.EXE. Las dos presentaciones soportan a TODOS los microcontroladores de la familia PIC de Microchip. El conjunto de instrucciones de los microcontroladores PIC es en esencia la base del lenguaje ensamblador soportado por este software.

Directivas de uso frecuente: Son instrucciones para el compilador.

#DEFINE

ej. #define <nombre> [<valor a remplazar>]

explicación: declara una cadena de texto como substituto de otra

END

ej. end

explicación: indica fin de programa

EQU

ej. status equ 05

explicación: define una constante de ensamble

INCLUDE

ej. include <PIC16F84.h>

explicación: incluye en el programa un archivo con código fuente

ORG

ej. org 0x100

explicación: ensambla a partir de la dirección especificada

Ejercicios de programación en Maplab IDE

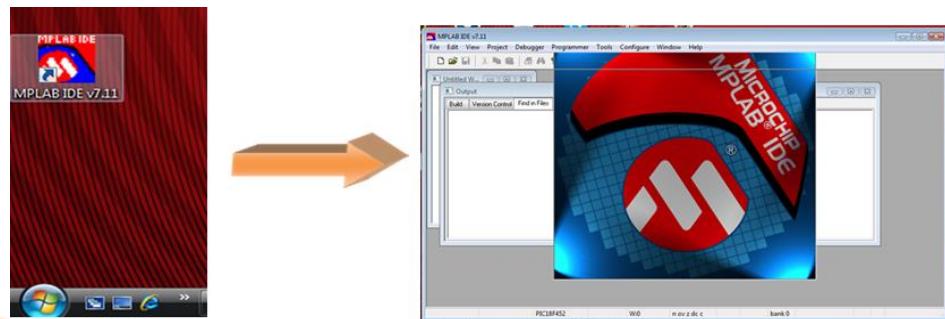
Para información más completa referirse a la guía rápida del MPASM. Una vez instalado adecuadamente el MPLAB, para realizar la simulación de un programa deben seguirse los siguientes pasos:

- Edite en un archivo de texto el siguiente programa:

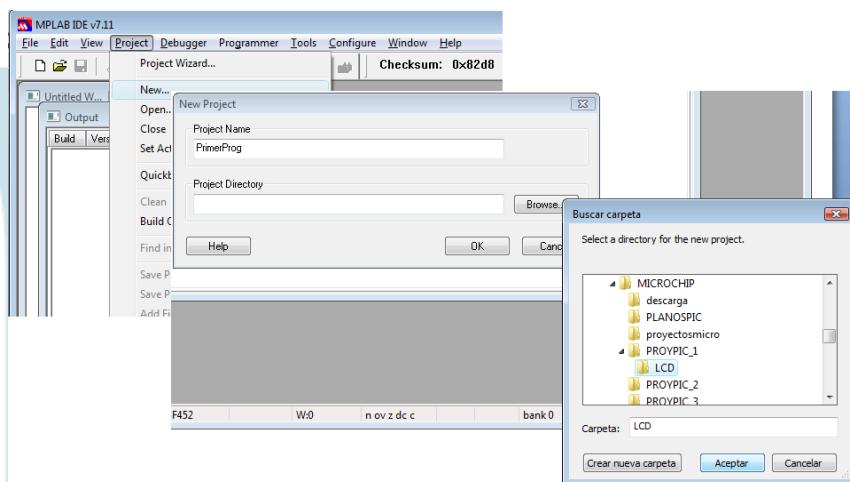
```
status equ 0x03 ;hace equivalencia entre el símbolo status indicándolo como 3 en
hexadecimal
Cont equ 0x20
F equ 1
org 0 ;indica posición de memoria desde donde se ensambla
Inicio
    movlw 0x0F ;carga de w con el valor constante 15 (literal)
    movwf Cont ;el contenido de w se pasa al reg. CONT
Loop
    decfsz Cont,F ;decremento de Cont y elude siguiente si=0
    goto Loop ;salto incondicional a Loop
    goto $ ;Salto incondicional aquí mismo
end ;Fin del código
```

Lista de pasos:

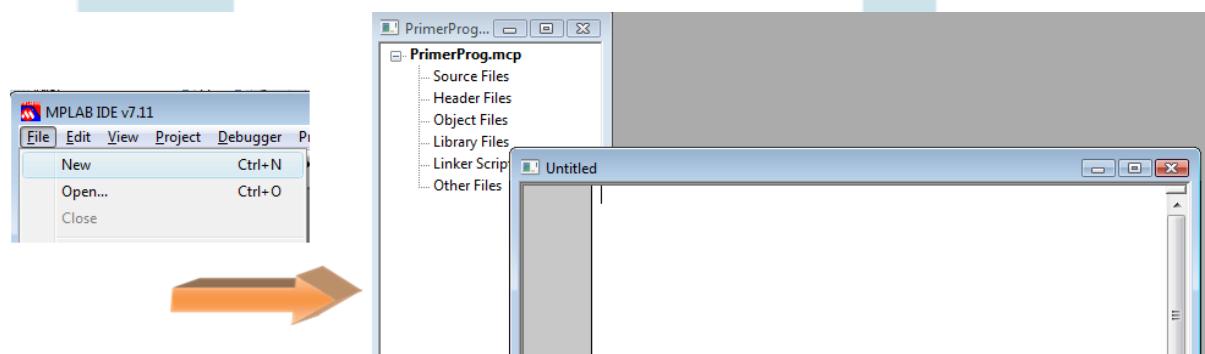
1. Haga doble clic en el ícono correspondiente a MPLAB.



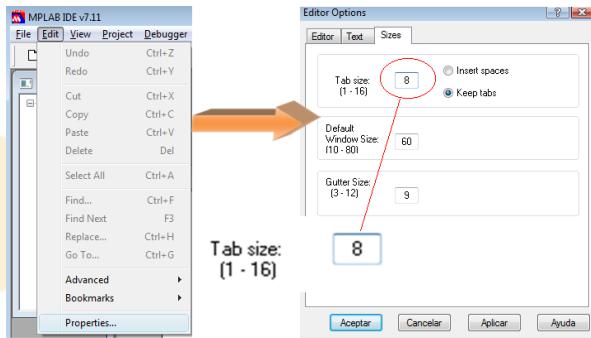
2. Crear el nuevo proyecto menú/Project/New ... dar un nombre al proyecto y seleccionar el directorio donde se guardará.



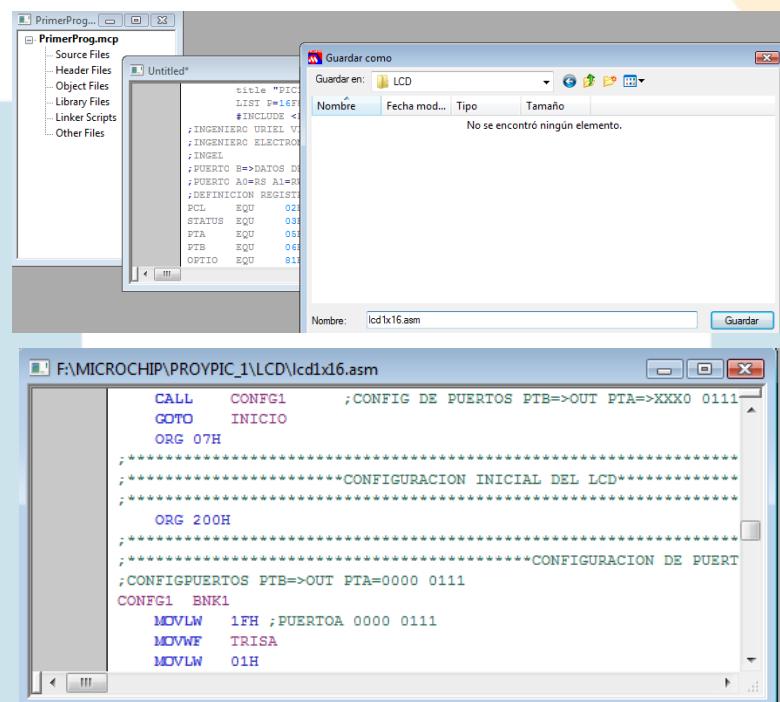
3. Crear un archivo nuevo. Menu/New



4. Configurar las propiedades del editor. Menu/Edit/Properties. Pestaña sizes/Tab size de 8

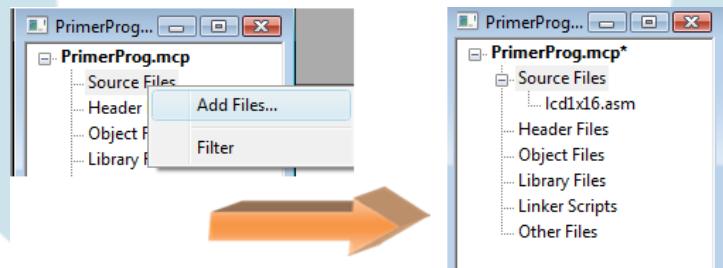


5. Salvar el archivo (con extensión .ASM) una vez terminada su edición. Menú/File/Save.

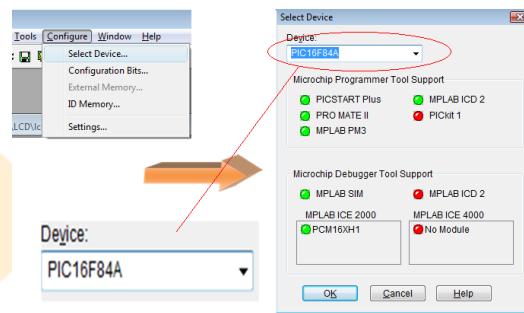


El cambio de color en el texto indica que se toma como “programa fuente”.

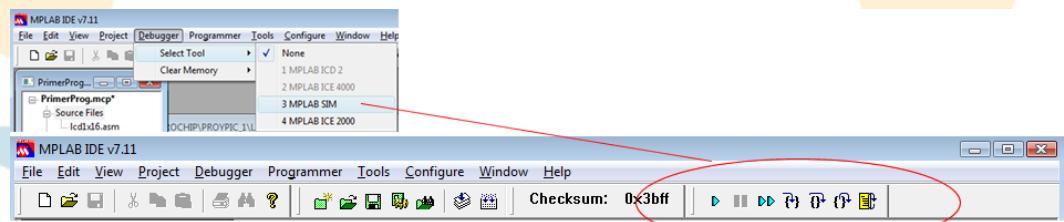
6. En la ventana de proyecto activo .MCW seleccionar SourceFiles/ clic derecho AddFiles y seleccionar el archivo .ASM para incorporarlo al proyecto.



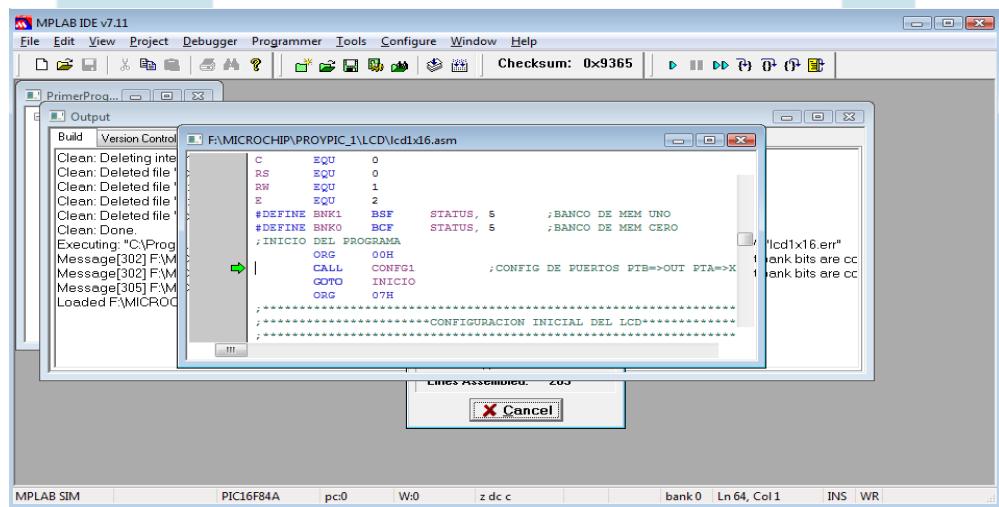
7. Seleccione el componente de la lista de modo que quede el “PIC16F84A”. Menu/Configure/SelecDevice.



8. Para comenzar con el simulador. Menu/Debugger>Select Tool/MPLAB SIM, con lo que aparece una barra de herramientas para la simulación.



9. En el menú/Project/Build All, se realiza la compilación del o "construcción de todo el proyecto".



10. Para realizar una simulación paso a paso, oprima "F7" o utilice las opciones en la barra de simulación: Run (correr programa), Halt (pausa), Animate (animación secuencial) , StepInto (paso a paso), StepOver (salto la subrutinas), StepOut (sale de subrutina), Reset (reinicia).

11. Menu/View/Watch, visualizan los registros desde esta ventana se pueden elegir los registros tanto "Add SFR" o de propósito especial y "Add Symbol" o los símbolos que se han asignados. De la ventana desplegable se selecciona el registro o símbolo y se da un clic en los botones "Add SFR" o "Add Symbol".

12. Menu/View/Disassembly Listing, desensambla el programa a su código en lenguaje hexadecimal.

13. Menu/View/Hardware Stack, visualiza la pila de 8 niveles para el caso de PIC16F84A.
14. Menu/View/Program Memory, visualiza la memoria de programa, el OPCODE y las instrucciones desensambladas.
15. Menu/View/Files Registers, visualiza los registros de la memoria de datos.
16. Menu/View/EEPROM, muestra los datos almacenados en la EEPROM.
17. Menu/View/Memory Usage Gauge, muestra el estadístico de uso de memoria de datos y de programa.
18. Menu/View/Special Function Registers, muestra los registros de propósito especial o FSR y su contenido.
19. Menu/Debugger/Stimulus Controller/New Scenario, permite simular pulsos externos a los pines del microcontrolador, estos estímulos pueden guardarse como un archivo mas del proyecto.
20. Menu/File/Save Work Space, guarda el espacio de trabajo, lo que permite guardar las ventanas activas de un proyecto y cuando se abra el proyecto nuevamente en otra ocasión visualizar las ventanas que tenia.
21. Menu/Project/Save Project, se encarga de guardar el proyecto activo.
22. Hacer doble clic al lado izquierda de una línea de instrucción dentro del programa fuente después de que se compile, genera “BreakPoints” o puntos de ruptura donde el simulador cuando esta corriendo (Run), para y espera una señal de salto como por ejemplo “F7” para seguir paso a paso. Esta opción es muy utilizada para depurar el programa o hallar errores de decisión o lógicos dentro del programa de control.
23. En caso de tener un programador compatible con MPLAB puede configurar los “bits” de programación del microcontrolador accediendo a Menu/Configure/Configuration Bits, aparece una ventana que despliega las opciones de configuración para el tipo de Oscilador, encendido o apagado del WatchDog Timer, Control al encendido “Power Up Timer” y el código de protección “Code Protect”.
24. Las demás funciones son funciones mas complejas para programación avanzada, recurrir al manual de MPLAB suministrado por Microchip.

WinIDE

Es un entorno de desarrollo integrado (IDE) bajo windows para la plataforma de microcontroladores Motorola Freescale, tanto para microcontroladores de 8bit, 16 bits y 32 bits, se conoce como el paquete de software ICS08GPZ In-Circuit Simulator (WinIDE), desarrollado por la compañía PEmicro, es un software gratuito que integra un editor y compilador en lenguaje ensamblador, programación del

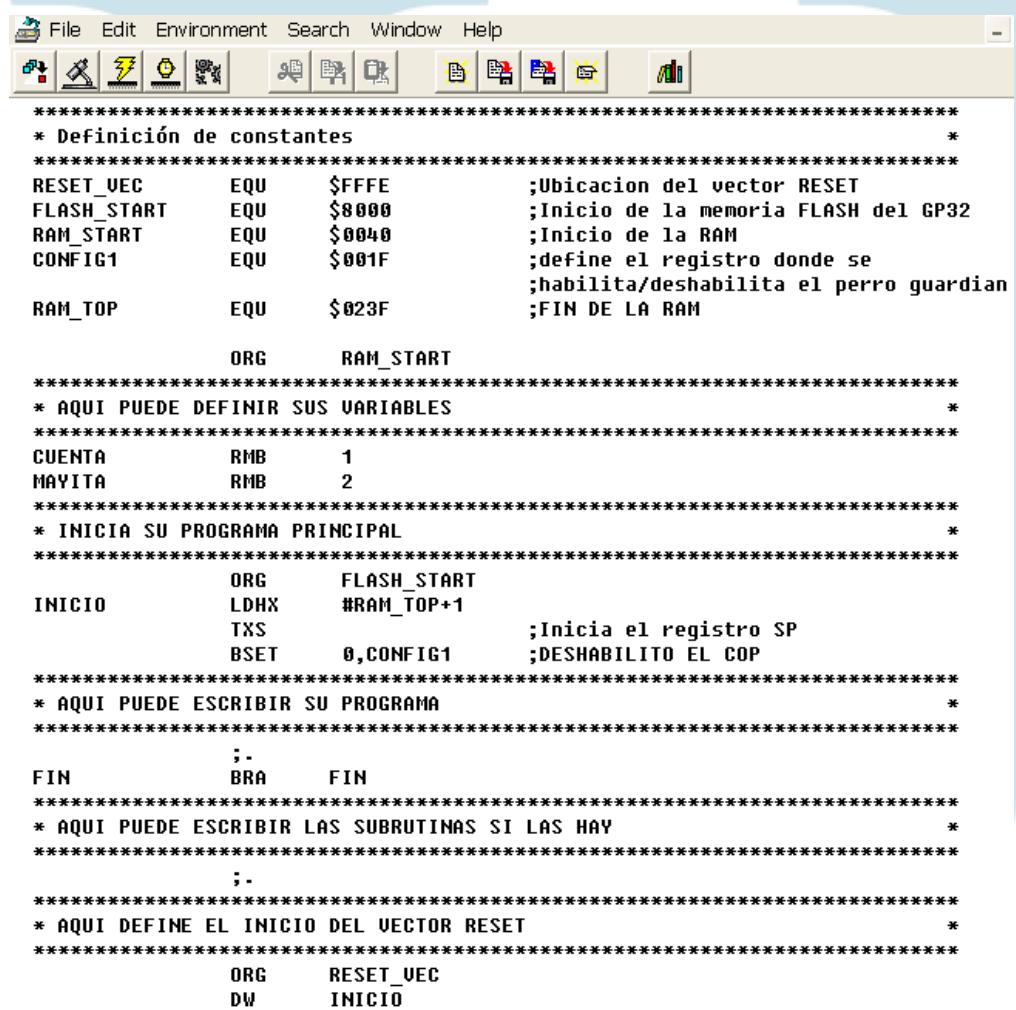
chip, simulador In-Circuit o sin chip. WinIDE soporta la mayoría de microcontroladores de la familia HC08 como los micros JK, JL, GP, QT y Qy.

WinIDE permite visualizar y modificar la memoria, registros y pines I/O, implementa opciones de ejecución de programa (paso a paso, run, etc), se pueden establecer hasta 64 breakpoints.

Ejercicios de programación en WinIDE

Para ejecutar WinIDE se debe instalar el paquete de software ICS08GPZ que usualmente se ubica en el archivo “c:/pemicro”, se accede en “Inicio/Todos los programas/WinIDE Development Environment”.

Figura 88. Entorno de desarrollo WinIDE¹¹²



The screenshot shows the WinIDE interface with a menu bar (File, Edit, Environment, Search, Window, Help) and a toolbar with various icons. The main window displays assembly language code:

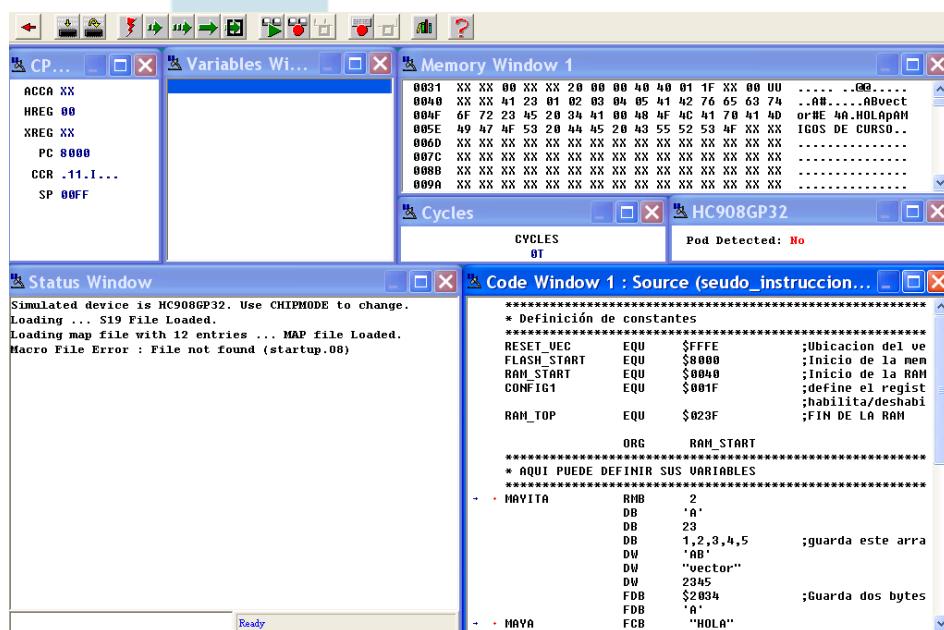
```
*****  
* Definición de constantes *  
*****  
RESET_VEC EQU $FFFE ;Ubicacion del vector RESET  
FLASH_START EQU $8000 ;Inicio de la memoria FLASH del GP32  
RAM_START EQU $0040 ;Inicio de la RAM  
CONFIG1 EQU $001F ;define el registro donde se  
;habilita/deshabilita el perro guardian  
RAM_TOP EQU $023F ;FIN DE LA RAM  
  
ORG RAM_START  
*****  
* AQUI PUEDE DEFINIR SUS VARIABLES *  
*****  
CUENTA RMB 1  
MAYITA RMB 2  
*****  
* INICIA SU PROGRAMA PRINCIPAL *  
*****  
ORG FLASH_START  
INICIO LDHX #RAM_TOP+1  
TXS ;Inicia el registro SP  
BSET 0,CONFIG1 ;DESHABILITO EL COP  
*****  
* AQUI PUEDE ESCRIBIR SU PROGRAMA *  
*****  
;  
FIN BRA FIN  
*****  
* AQUI PUEDE ESCRIBIR LAS SUBRUTINAS SI LAS HAY *  
*****  
;  
*****  
* AQUI DEFINE EL INICIO DEL VECTOR RESET *  
*****  
ORG RESET_VEC  
DW INICIO
```

¹¹² Herrera, Lucelly. (n.d.) Micros 2012 WinIDE

Como se observa, se presenta el formato de un programa escrito para Motorola Freescale, muy similar al formato para microcontroladores Microchip PIC, el archivo se debe guardar como .ASM, preferiblemente en una carpeta individual, puesto que al compilar se generan otros archivos y puede causar confusión.

Se puede ver en la parte superior el menú de opciones generales (File, Edit, Environment, Search, Windows, Help) y enseguida el menú de herramientas, de izquierda a derecha, Ensamblar, Simular en circuito, Programar, Depurar en Circuito y Simular son los principales. El primer icono el cual una vez editado el programa, la acción sobre este icono permite ensamblar y compilar, lo que genera archivos .BAK, .LST, .MAP y .S19. El archivo .S19 es el archivo que contiene el lenguaje máquina y es el que debe grabarse en el micro. El segundo icono permite el ingreso al “In-Circuit Simulator”, para realizar la simulación In-Circuit o sin chip, aparece una ventana en la cual se selecciona “Simulation Only” para simular sin Chip, solo software.

Figura 89. Entorno de simulación WinIDE¹¹³



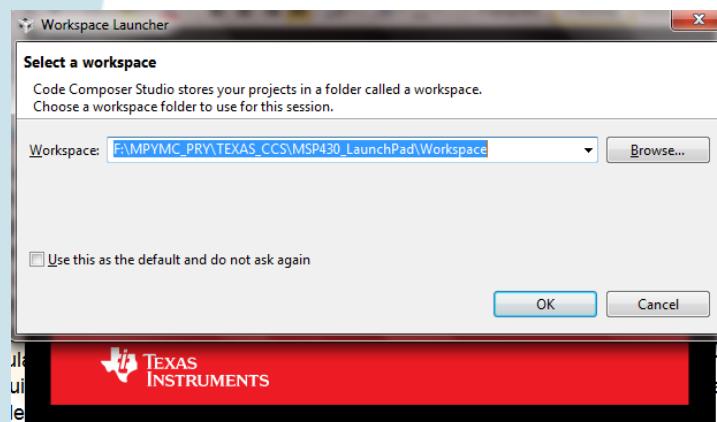
El entorno de simulación es similar en características y despliegue gráfico al MpLab, nuevamente los iconos de la barra de herramientas ubicada abajo del menú, presentan las siguientes opciones de izquierda a derecha, Regresar al editor, Cargar el programa, Reset, Correr paso a paso, Correr múltiples pasos, Correr el programa y Parar el programa.

¹¹³ Herrera, Lucelly. (n.d.) Micros 2012 WinIDE

Code Composer Studio (CCS) versión 5.

Es un entorno de desarrollo integrado para los microcontroladores de Texas Instruments, de versión gratuita, con límite de programación de memoria de programa para el LaunchPad MSP430 y los micros de la línea MSP430. Está construido bajo eclipse y visual C++, presenta una interfaz de desarrollo en la cual se debe generar un proyecto que contiene el archivo “workspace” o la configuración del espacio de trabajo, el código fuente que puede estar escrito en C o ensamblador. La ventaja de este IDE es poder hacer la programación y simulación In-Circuit mediante la tarjeta de desarrollo LaunchPad. La versión gratuita se descarga desde la página de Texas Instruments, en el wiki para la línea de desarrollo LaunchPad MSP430, se aconseja la versión 5. Su instalación requiere 3 GB de disco disponible. Al iniciar el programa se le debe indicar el directorio de trabajo “Workspace”.

Figura 90. Ventana de inicio de CCS v.5, selección del Workspace

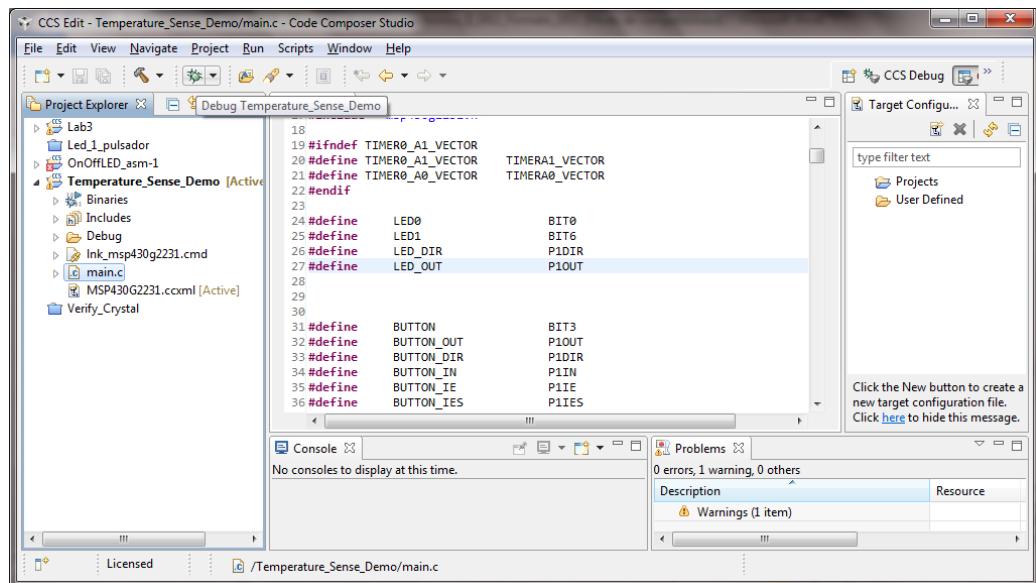


En el wiki de Texas Instruments (www.ti.com/launchpadwiki) se encuentra gran cantidad de material de apoyo, instructivo y ejemplos de aplicación utilizando la herramienta LaunchPad MSP430 y sus micros MSP430. El entorno de trabajo presenta una ventaja principal con opciones de menú y barra de herramientas, junto con otras ventanas que muestran los proyectos actuales “Project Explorer” en el extremo izquierdo, en la parte central los archivos que se están editando, en la parte derecha la ventana “Target Configurations” y en la parte inferior la ventana de “Consola” que muestra el estado del proceso de compilación y la ventana “Problems” que muestra los errores detectados en la compilación del programa.

El ícono con el símbolo del “martillo” es el compilador “Build ‘Debug’ for Project”, con el que se hace la compilación y depuración de sintaxis, es decir aparecen los errores o problemas indicados en las ventanas “Problems” y señalados en la ventana central de edición de programas. El ícono similar a un “insecto”

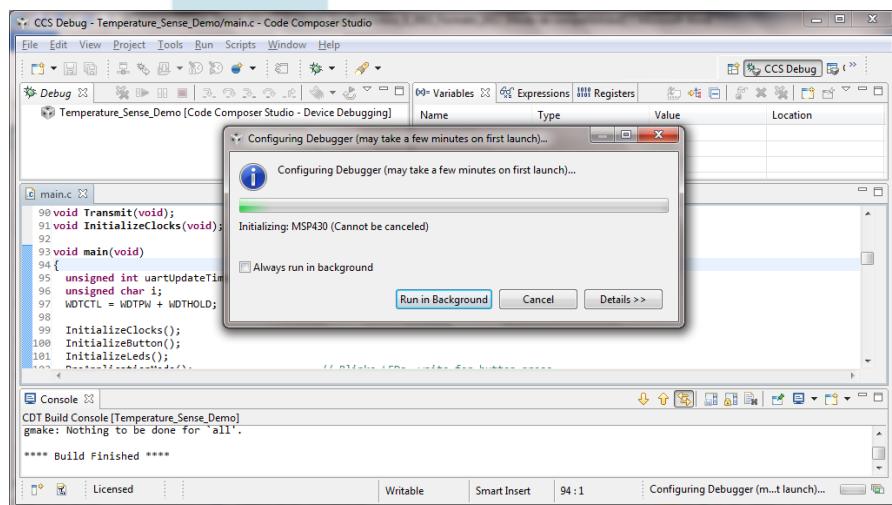
denominado “Debug ‘nombre del proyecto’” también genera la compilación y si no se tiene errores o problemas detectados, activa la interfaz de programación y simulación In-Circuit, opción “FET Debugger” por defecto, para lo cual debe estar conectado el LaunchPad MSP430 ya que no solo programara el chip, también puede simular paso a paso o correr el programa tanto en el PC como en el micro.

Figura 91. Interfaz del CCS v 5



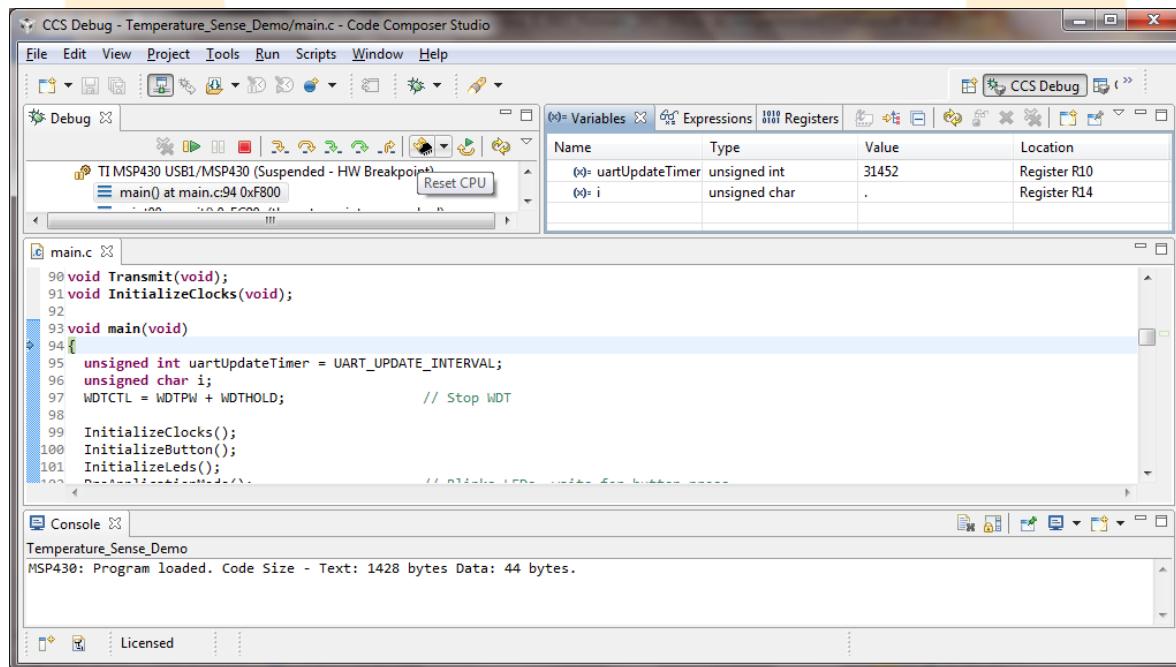
En el entorno de programación y simulación In-Circuit CCS v5, permite mediante una interfaz relativamente compleja pero completa, la programación del chip y la simulación instrucción por instrucción (Step) o la ejecución total del programa (Resume), utilizando la Tarjeta de desarrollo LaunchPad MSP430.

Figura 92. Entorno de programación y simulación In-Circuit CCS v5



Como se observa en la ventana “Debug” aparecen las opciones de simulación, correr el programa (Resume), detener o terminar la simulación (Terminate), instrucción por instrucción (Step), el Reset, el reinicio y el refrescar. Se observa una ventana donde se puede visualizar el estado de las variables, expresiones y registros, luego la ventana con el programa en ejecución mostrando la línea que se ejecuta en este momento y la consola con información sobre el estado de ejecución de las instrucciones.

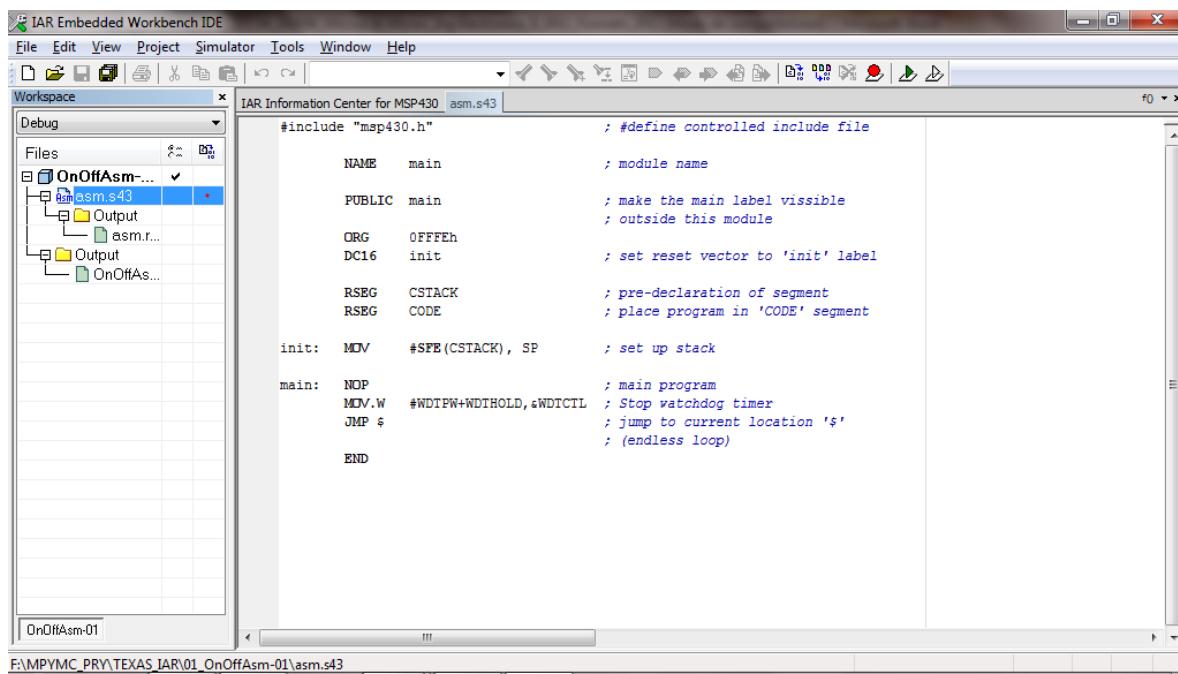
Figura 93. Programa listo para simular con tarjeta LaunchPad MSP430



IAR Embedded WorkBench

Es una interfaz de desarrollo similar a CCS para los microcontroladores Texas Instruments con el LaunchPad MSP430, presenta un entorno amigable para la programación y simulación “debugger” sin chip o “FET debugger” con el chip en la tarjeta LaunchPad. Puede editarse, compilar y simular proyectos con código fuente en ensamblador o en C. Al igual que el CCS, Texas Instruments suministra el software gratuitamente con límite de memoria de programa, ocupa menos espacio en su instalación no más de 1GB de disco.

Se debe generar un proyecto que contiene un archivo “Workspace”, el código fuente, si es en ensamblador .s43 o en .C, junto con otros archivos generados en la compilación, por lo que nuevamente es conveniente crear un directorio con el nombre del proyecto donde se guarden dichos archivos para evitar confusión a la hora de retomar el proyecto en desarrollo o final.

Figura 94. Interfaz IAR Embedded WorkBench

Software de Simulación para microcontroladores

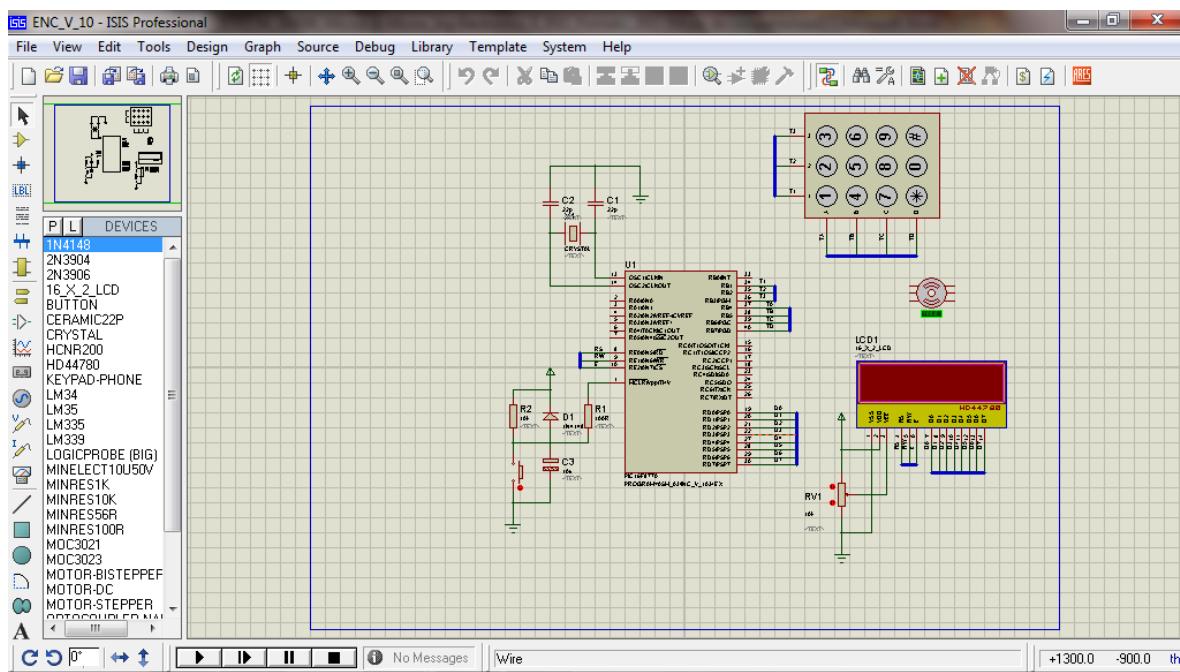
Son varios los programas o software de simulación disponibles para el desarrollo de proyectos con microprocesadores y microcontroladores, como se ha planteado cada fabricante de micros, dispone de software en la forma de Entornos de Desarrollo Integrado – IDE generalmente gratuitos con limitaciones o totalmente libres para poder programar los micros, como IDE permiten la edición, compilación, depuración, programación y simulación, muchos de ellos actualmente la simulación In-Circuit, es decir con el chip dentro de la tarjeta de desarrollo.

Por tanto, se puede inferir la complejidad que representa estar aprendiendo el manejo de una herramienta para cada familia de microcontroladores, generalmente esta información está en inglés lo que para muchos aprendices puede generar un obstáculo. En el mercado hay paquetes de simulación generalmente comerciales que integran varias familias de microcontroladores de varios fabricantes, con lo que se tiene una herramienta de simulación genérica e incluso de fabricación de impresos para el proyecto final. En contra posición está el costo elevado de la licencia del programa y librerías especializadas para microcontrolador. Sin embargo se encuentran versiones libres o trial o limitadas con las que se puede simular algunos microcontroladores. Muchas fuentes de referencias de desarrollo de proyectos utilizan software como Proteus y Multisim para simular y fabricar los impresos.

Proteus

Es un software comercial, de alto costo considerando que hay que pagar por la licencia de uso y aparte por las librerías que se requieran, en estas librerías está los algoritmos para simular los diversos microcontroladores. Actualmente en su versión 9 permite la simulación de gran parte de las familias y micros más utilizados, además de tener la capacidad de generar la placa impresa y el diseño 3D. Implementa gran cantidad de periféricos lo que lo hace una herramienta potente para el desarrollo de proyectos digitales. En el área de simulación de circuitos electrónicos de sistemas analógicos es menos potente que otros paquetes como Multisim.

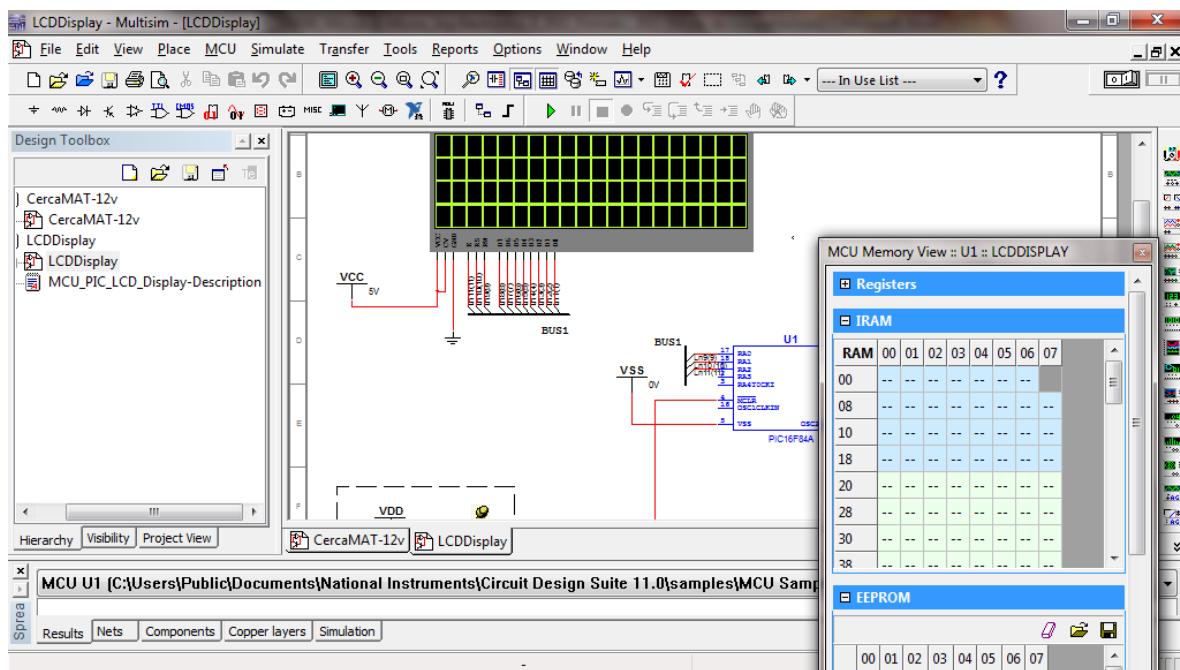
Figura 95. Interfaz de desarrollo Proteus Professional v.7.9 – ISIS



Multisim

Es un paquete comercial de simulación producido por Nacional Semiconductor, que se encuentra individual o dentro del paquete Labview de la misma empresa, es un potente simulador tanto de sistemas electrónicos análogos como digitales, implementa librerías de gran cantidad de componentes y fabricantes, entre ellos algunos microcontroladores, aunque en este aspecto Proteus – ISIS implementa mucha más cantidad de microcontroladores. Multisim presenta una excelente plataforma de simulación en un entorno de simulación con gran versatilidad para ajustes de parámetros de componentes específicos o ajustes generales, también incorpora la capacidad de generar el circuito impreso.

Figura 96. Entorno de desarrollo Multisim



Programadores de la memoria del microcontrolador

Cuando no se tiene entornos de simulación sofisticados, que por su costo no es posible adquirirlos o que el fabricante provee el software, pero las tarjetas de programación son un hardware costoso o difícil de conseguir, muchos desarrolladores particulares o externos a los fabricantes desarrollan tanto software como hardware, en tarjetas de programación, tarjetas de programación In-Circuit, tarjetas de desarrollo y prueba o como una combinación total o parcial de las tarjetas mencionadas anteriormente. Con lo que el usuario puede implementar con pocos circuitos y componente electrónicos tarjetas para programar, simular y/o desarrollar pruebas de sus proyectos en desarrollo.

Como se ha mencionado en ocasiones el hardware es compatible con el software suministrado por el fabricante, por ejemplo en tarjetas clon del PicKit de microchip que es compatible con el software de Microchip o desarrollan software independiente para las tarjetas, como ejemplo podría tenerse la tarjeta denominada "PROPIC 2" para microchip PIC de puerto paralelo compatible con el software "ICEPROG" para programación de microcontroladores PIC y memoria seriales.

Son muchos los ejemplos de desarrollos independientes, por ejemplo el proyecto Arduino se basa en tarjetas de hardware desarrolladas y de acceso libre, que utilizan el software particular de Arduino basado en C/C++ o con otros software como Python. Todo esto hace que al investigar en internet pueda encontrarse tanto software como hardware de uso libre para poder estudiar o desarrollar proyectos con microprocesadores y microcontroladores.

Sistemas y microcontroladores programados In-circuit

Estos sistemas y microcontroladores se refieren a microcontroladores con la facilidad y capacidad de ser programados dentro de la misma tarjeta de desarrollo del prototipo, característica denominada In-Circuit, los microcontroladores PIC tiene esta característica puesto que mediante cinco (5) líneas reciben la alimentación (0V y 5V), los datos, el reloj y el voltaje de programación, lo que hace relativamente sencillo implementar en la misma placa el circuito de programación serial o paralelo hacia el PC, la utilización de una interfaz USB requiere de un circuito más sofisticado con un microcontrolador que contenga la firma digital o “firework” para que pueda establecer la comunicación con el PC.

Los microcontroladores de Motorola Freescale, Microchip, Texas Instruments, AVR e incluso Intel implementan la facilidad de programación en serie, mediante el desarrollo del circuito para interfaz RS-232 y el PC. La familia de microcontroladores Basic Stamp, Arduino, entre otros, proporcionan en sus tarjetas además de los puertos I/O y los pines de alimentación, pines para la programación en serie por interfaz RS-232 o por USB. Como constante la gran mayoría de fabricantes, presenta tarjetas de desarrollo en las cuales se puede programar, programar y simular In-Circuit, y probar proyectos de desarrollo desde la misma placa o tarjeta, ejemplo de este tipo de tarjetas de desarrollo está el sistema LaunchPad MSP430 para microprocesadores Texas Instruments de hasta 14 pines MPS430.

Entrenadores para programación con microcontroladores

Los entrenadores son placas de hardware, en ocasiones vinculadas con software especializado para el aprendizaje o entrenamiento en la programación y desarrollo de proyectos. Las placas de hardware entrenadoras presentan una o todas las características que se mencionan a continuación, generalmente un circuito de programación serial o USB, un circuito de comunicaciones para simulación In-Circuit, un circuito para interfaz exterior para posibilitar la conexión a una protoboard, un circuito compuesto por los periféricos más usuales con pulsadores, teclado matricial, LCD, display 7-segmentos, sensores de temperatura, drivers

para motores, SCR, TRIAC etc, con las terminales para poder conectar el microcontrolador con los periféricas.

El objetivo de la placa entrenadora es facilitar el aprendizaje y prueba de proyectos de desarrollo, su costo es elevado y es utilizada en la mayoría de laboratorios especializados en la enseñanza de microprocesadores y microcontroladores. En la web y material bibliográfico especializado se pueden encontrar desarrollo de tarjetas de uso libre como por ejemplo Arduino, tarjetas a precios muy accesibles como por ejemplo LaunchPad MSP430 de Texas Instruments e implementaciones de tarjetas que el mismo estudiantes puede desarrollar directamente en protoboard y luego en placas universales.



CAPITULO 5: MICROCONTROLADORES PIC DE MICROCHIP

Introducción

En las siguientes lecciones se presenta los aspectos básicos de los microcontroladores PIC de Microchip, particularmente enfocando en la programación y aplicación con microcontroladores PIC 16F84A y microcontroladores PIC16F877A, como dispositivos y punto de partida para aprender el funcionamiento interno de un microcontrolador simple, su arquitectura Harvard, el manejo del set de instrucciones y la implementación de ejercicios prácticos sencillos, para que el lector adquiera las competencias básicas, conocimientos generales y habilidad para iniciar el estudio de otras familias de microcontroladores, de manera que pueda estar en capacidad de desarrollar soluciones de hardware y software utilizando la variedad de dispositivos disponibles en el mercado por varios fabricantes y familias.

Lección 21: Microcontroladores PIC

En esa lección se resaltan algunos aspectos generales de los microcontroladores PIC, desde la identificación del chip, pasando por su funcionamiento, la presentación de su arquitectura, las características más relevantes, el funcionamiento del circuito oscilador y Reset, mostrando como está organizada la memoria y estudiando los principales registros y su función, para de esta manera plantear las fuentes de interrupción, la función del módulo Timer, reconocer otros módulos que implementa la familia PIC, finalizando con el set de instrucciones.

Identificación en el chip

Los códigos que identifican a un microcontroladores PIC, también entregan varias características importantes al usuario:

- El tipo de memoria utilizado se identifica por la letra después de la serie de gamma del microcontrolador:
 - **C**, como PIC16CXXX utiliza memoria EPROM.
 - **CR**, como PIC16CRXXX utiliza memoria ROM.
 - **F**, como PIC16FXXX utiliza memoria tipo Flash.
- El rango de voltaje tanto estándar como extendido se identifica así:

Tabla 29. PIC Estándar y Extendido¹¹⁴

Rango de Voltaje	EPROM		ROM		FLASH	
Estándar	16CXXX	4,5 - 6,0 V	16CRXXX	4,5 - 6,0 V	16FXXX	4,5 - 6,0 V
Extendido	16LCXXX	2,5 - 6,0 V	16LCRXXX	2,5 - 6,0 V	16LFXXX	2,0 - 6,0 V

Arquitectura

Como se ha mencionado, el microcontrolador emplea una arquitectura Harvard, en la cual, programa y datos se acceden de memorias separadas usando diferentes buses. Esto trae como consecuencia la existencia de dos buses independientes y la posibilidad de tener un ancho diferente. Para los microcontroladores estudiados en este texto, el bus de datos es de 8 bits y el bus de programa es de 14 bits.

Implementa una arquitectura de instrucciones RISC, lo que hace fácil el aprendizaje y aplicación, pero dispendioso al momento de ejecutar operaciones complejas debido a que el conjunto de instrucciones es limitado. El ALU está conectada a un único registro acumulador, denominado Registro de Trabajo (W), por tanto, toda la información entre el ALU y la memoria o periféricos, debe pasar por el registro "W", labor que extiende las operaciones y directamente influye en la extensión y complejidad de los programas.

Arquitectura PIPELINE: En los microcontroladores PIC, un ciclo de instrucción o ciclo maquina, está dividido en cuatro (4) ciclos o pulsos periódicos de reloj, la ejecución de una instrucción diferente a las de salto, requiere la microoperación de búsqueda de la instrucción, que tarda un ciclo máquina y la microoperación de decodificación y ejecución que tarda otro ciclo máquina, es decir, en total dos ciclos, pero debido a que los micros implementan una arquitectura PIPELINE, en el ciclo máquina que decodifica y ejecuta la instrucción también se está realizando la búsqueda de la siguiente instrucción, por lo que efectivamente una instrucción se ejecuta en un (1) ciclo máquina. Las instrucciones que hacen que se modifique el contenido del registro Contador de Programa (PC), necesitan un ciclo máquina

¹¹⁴ CEKIT, 2002

extra para completar la instrucción, por tanto, instrucciones como GOTO, CALL utilizan dos (2) ciclos máquina para su ejecución.

Ejemplo: En un microcontrolador PIC16F84 configurado con oscilador de cristal (XT) de 4 MHz, se tiene que un ciclo máquina o de instrucción al requerir de 4 ciclos internos, utiliza un tiempo de 1 microsegundo para ejecutar una instrucción que no modifique el registro Contador de Programa (PC), porque si el cristal es de 4MHz, el periodo o tiempo de cada ciclo de frecuencia es de $\frac{1}{f} = 1/4\text{MHz}$, expandiendo, $\frac{1}{f} = \frac{1}{4000000\text{Hz}} = 0,00000025 = 0,25 \times 10^{-6} = 0,25\mu\text{s}$, es decir, 0,25 microsegundos, como se utilizan 4 pulsos de $0,25\mu\text{s}$ para un ciclo máquina, se tendría que un ciclo máquina en este microcontrolador tarda $1\mu\text{s}$, por tanto está en capacidad de ejecutar aproximadamente 1 millón de instrucciones por segundo o 1MIP.

Funcionamiento

La secuencia de las instrucciones de un programa, está controlada por el registro contador de programa (Program Counter). Este registro se incrementa en cada paso para ejecutar el programa grabado en la memoria ROM del microcontrolador. El contador de programa también está conectado a los registros de Stack o pila, que son los que soportan llamadas consecutivas a una subrutina. El microcontrolador también posee una unidad lógica aritmética (ALU) de 8 bits y un registro de trabajo (W). La ALU ejecuta funciones aritméticas y booleanas entre datos del registro de trabajo y cualquier otro registro o constante. También es capaz de realizar operaciones lógicas, adiciones y sustracciones entre datos de 8 bits.

El registro de trabajo (W), es un registro de 8 bits de gran importancia para el buen funcionamiento del microcontrolador. Se emplea intensamente en las operaciones que requieren transferencia interna de datos, como por ejemplo, en las operaciones de la ALU o en las comunicaciones entre registros. Para la ejecución de un programa, las palabras de la memoria de programa se llevan al registro de instrucciones, el cual, las comunica al decodificador de instrucciones, en donde se da la orden de iniciar su ejecución. A continuación se presenta una breve descripción de los principales componentes del microcontrolador.

Características Generales de los microcontroladores

Los microcontroladores PIC se caracterizan por incorporar una serie de módulos internos y periféricos que caracterizan a cada dispositivo para una aplicación

específica, los pines o líneas de entrada / salida (I/O) son el periférico principal y muchas veces se multiplexan con otras funciones, dotando de flexibilidad al microcontrolador para múltiples aplicaciones.

Los microcontroladores PIC cuentan con varios encapsulados directamente relacionados con el nivel de aplicación o desarrollo del proyecto, para proyectos en fase de prototipo se utilizan micros con encapsulado DIP versión borrable ultravioleta (UV), caracterizados por tener una ventana en la parte superior del chip fabricado en cerámica y los más comunes del tipo borrable eléctricamente en encapsulado plástico, como la versión EEPROM (Tipo C) y el más usual y práctico la versión con memoria FLASH (Tipo F). En proyectos cuyo nivel está en el de producción a baja o media escala, se utilizan micro con encapsulado DIP versión con memoria ROM (Tipo CR), en encapsulado plástico, son del tipo OTP (One Time Programmable – Dispositivo programable una vez), también están los dispositivos ROM con memoria establecida desde el momento de su fabricación por Microchip, para pedidos notablemente grandes en programación QTP (Quick Turn Production – Programación rápida para producción) y programación SQTP (Serialized Quick Turn Production – Programación serial rápida para producción).

Oscilador y circuito de reset

El circuito oscilador es un módulo interno del microcontrolador, en algunos micros incorpora una red RC para generar los pulsos de reloj internamente y permitir utilizar un par de pines más como I/O (entrada/salida), generalmente los micros permiten la conexión y configuración externa del oscilador, básicamente para PIC se distinguen cuatro tipos de oscilador:

LP: Oscilador de baja frecuencia compuesto por un cristal de baja frecuencia, caracterizado por un consumo bajo de corriente.

XT: Oscilador de frecuencia media, utiliza un Cristal / Resonador, es el más común.

HS: Oscilador de alta frecuencia, utiliza Cristal / Resonador de alta frecuencia, tiene un consumo más elevado de corriente.

RC: Oscilador con red externa RC, es el más económico, pero poco estable y precisa en frecuencia.

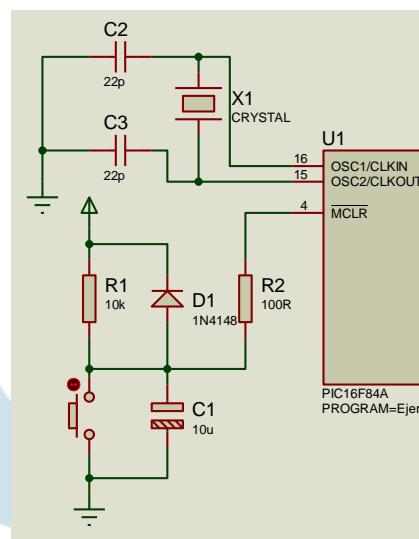
INTRC: Oscilador con red RC interna. Es la más económica.

Tabla 30. Valores típicos de Cristal/Resonador y condensadores

MODO	Frecuencia	Condensadores		TIPO
		C1	C2	
LP	32KHz	68pF-100pF	68pF-100pF	Cristal
	200KHz	15pF-30pF	15pF-30pF	Cristal
XT	455KHz	22pF-100pF	22pF-100pF	Resonador
	2 MHz	15pF-68pF	15pF-68pF	Resonador
	4 MHz	15pF-68pF	15pF-68pF	Resonador
	100KHz	68pF-150pF	150pF-200pF	Cristal
	2 MHz	15pF-30pF	15pF-30pF	Cristal
	4 MHz	15pF-30pF	15pF-30pF	Cristal
HS	8 MHz	10pF-68pF	10pF-68pF	Resonador
	16 MHz	10pF-2pF	10pF-2pF	Resonador
	8 MHz	15pF-30pF	15pF-30pF	Cristal
	10 MHz	15pF-30pF	15pF-30pF	Cristal
	20 MHz	15pF-30pF	15pF-30pF	Cristal

Circuito de Reset: En los microcontroladores PIC el circuito de Reset se encuentra en el Pin conocido como \overline{MCRL} Master Clear, la línea superior indica que cuando se lleva a estado bajo (0 voltios) el PIC entra en estado Reset, por tanto, todas las salidas se apagan y el circuito de reloj se desactiva, para evitar disparos de Reset erróneos, este pin incorpora un filtro contra ruido, además es conveniente utilizar el circuito adicional para implementar el circuito externos para Reset manual, protección contra rebote y POR (Reset por encendido).

Figura 97. Conexión básica de Oscilador externo y circuito Reset (con circuito POR) para PIC.



Estado de Reset: Cuando sucede un estado de Reset el microcontrolador PIC, hace que el registro PC (Contador de Programa) apunte al vector de Reset, que contienen la dirección de inicio y ejecución del programa, dirección que corresponde a la 00H para PIC de media y alta gama y para los PIC de baja gama corresponde con la última posición de memoria de programa. El estado de Reset en los PIC puede ocurrir por pulso bajo en el Pin *MCRL*, Reset al encendido (POR), Reset por caída de voltaje de alimentación BOR. La fuente u origen del estado del Reset puede ser consultado en el registro PCON (Registro de Control de Potencia) cuando lo implementa el micro.

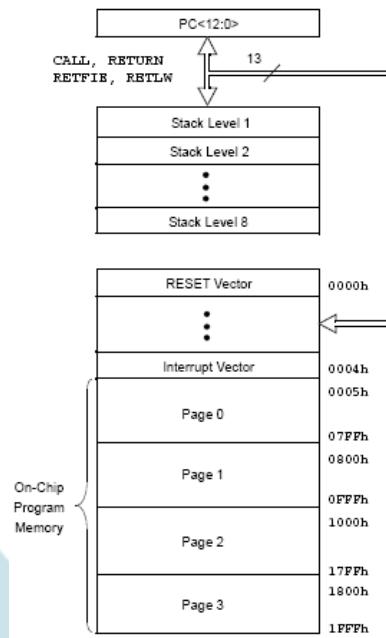
La arquitectura **Harvard** y **pipeline** con la que se implementan los microcontroladores PIC, establecen un conjunto de cuatro (4) pulsos de reloj para la ejecución de una (1) instrucción máquina, denominada también ciclo de instrucción o ciclo máquina.

Organización de la memoria y Registros

Existen principalmente dos bloques de memoria en el interior del microcontrolador: la memoria de datos y la memoria de programa, cada una con su propio bus.

Organización de la Memoria de Programa:

Figura 98.Organizacion de la memoria de programa¹¹⁵



¹¹⁵ Extraído el 18 Octubre de 2009 desde <http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf>

Organización de la memoria de datos: La memoria de datos está particionada en múltiples bancos (4) que contienen registros de propósito general y registros de función especial. Cuando se quiera acceder (leer o escribir) un registro FSR (registro de propósito especial), se debe localizar primero el banco de memoria que contiene el registro, direccionar por medio de los bits de paginación en el registro STATUS para luego poder hacer la operación sobre el registro. El banco de memoria de datos por defecto es el banco cero (BK-0).

Figura 99. Organización de la memoria de datos¹¹⁶

File Address	File Address	File Address	File Address
Indirect addr.(*) 00h TMR0 PCL STATUS FSR PORTA PORTB PORTC PORTD ⁽¹⁾ PORTE ⁽¹⁾ PCLATH INTCON PIR1 PIR2 TMR1L TMR1H T1CON TMR2 T2CON SSPBUF SSPCON CCPR1L CCPR1H CCP1CON RCSTA TXREG RCREG CCPR2L CCPR2H CCP2CON ADRESH ADCON0 General Purpose Register 96 Bytes	Indirect addr.(*) 01h OPTION_REG PCL STATUS FSR TRISA TRISB TRISC TRISD ⁽¹⁾ TRISE ⁽¹⁾ PCLATH INTCON PIE1 PIE2 PCON General Purpose Register 80 Bytes	Indirect addr.(*) 80h TMR0 PCL STATUS FSR PORTB TRISB TRISC TRISD ⁽¹⁾ TRISE ⁽¹⁾ PCLATH INTCON PIE1 PIE2 PCON General Purpose Register 80 Bytes	Indirect addr.(*) 100h TMR0 PCL STATUS FSR PORTB TRISB TRISC TRISD ⁽¹⁾ TRISE ⁽¹⁾ PCLATH INTCON PIE1 PIE2 PCON General Purpose Register 80 Bytes
02h 03h 04h 05h 06h 07h 08h 09h 0Ah 0Bh 0Ch 0Dh 0Eh 0Fh 10h 11h 12h 13h 14h 15h 16h 17h 18h 19h 1Ah 1Bh 1Ch 1Dh 1Eh 1Fh 20h	01h OPTION_REG PCL STATUS FSR TRISA TRISB TRISC TRISD ⁽¹⁾ TRISE ⁽¹⁾ PCLATH INTCON PIE1 PIE2 PCON General Purpose Register 80 Bytes	81h 82h 83h 84h 85h 86h 87h 88h 89h 8Ah 8Bh 8Ch 8Dh 8Eh 8Fh 90h 91h 92h 93h 94h 95h 96h 97h 98h 99h 9Ah 9Bh 9Ch 9Dh 9Eh 9Fh A0h EFh F0h FFh	101h 102h 103h 104h 105h 106h 107h 108h 109h 10Ah 10Bh 10Ch 10Dh 10Eh 10Fh 110h 111h 112h 113h 114h 115h 116h 117h 118h 119h 11Ah 11Bh 11Ch 11Dh 11Eh 11Fh 120h 16Fh 170h 17Fh
7Fh	7Fh	Bank 1	Bank 2
	accesses 70h-7Fh		accesses 70h-7Fh
			Bank 3
			1A0h
			1EFh
			1F0h
			1FFh

■ Unimplemented data memory locations, read as '0'.

* Not a physical register.

Note 1: These registers are not implemented on the PIC16F876.

2: These registers are reserved, maintain these registers clear.

¹¹⁶ Extraído el 18 Octubre de 2009 desde <http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf>

Los bits RP1 (bit 6) y RP0 (bit 5) del registro de estatus seleccionan los bancos de memoria como lo muestra la siguiente figura.

Figura 100. Cambio de bancos¹¹⁷

RP1:RP0	Bank
00	0
01	1
10	2
11	3

Registros de uso general: Son registros visibles al usuario, algunos modificables completamente o parcialmente en algunos bits.

Registro W: es un registro no direccionable de 8 bits utilizado para las operaciones de la ALU.

Registro STATUS(03h y 83h): El registro de estado contiene el estado aritmético de la ALU, el estado del proceso de re inicialización y los bits de selección de bancos de memoria, este registro puede trabajar como cualquier otro registro de almacenamiento, pero si la instrucción afecta matemáticamente, los bits Carry, Digit Carry y Bandera de Cero, se inhabilitan. El registro STATUS está formado por 8 bits:

Tabla 31. Registro Status.

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	/TO	/PD	Z	DC	C
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

R/W significa que el bit correspondiente se puede leer y escribir, mientras que R significa que solamente puede ser leído. También se indica el estado que se establece tras un reset.

Bit 7, IRP: Selección del banco en direccionamiento indirecto. Este bit junto con el de más peso del registro FSR sirven para determinar el banco de la memoria de datos seleccionado. En el PIC16X84 al disponer de dos bancos no se usa y debe programarse como 0.

¹¹⁷ Extraído el 18 Octubre de 2009 desde <http://www.microchip.com/downloads/en/DeviceDoc/30292c.pdf>

Bit 6 y 5, RP0 y RP1: Register Bank Select. Selección de página o banco de la memoria con direccionamiento directo. Cada página contiene 128 bytes. Como el PIC16X84 sólo tiene dos bancos únicamente se emplea RP0 de forma que cuando vale 0 se accede al banco 0 y cuando vale 1 se accede al banco 1. Despues de un Reset, RP0 se pone automáticamente a 0. RP1 debe mantenerse a 0. El bit RP1 deberá ser puesto a cero, ya que si no nos saldríamos del rango de memoria.

Bit 4 (flag), TO: Time Out (Tiempo acabado)

1. Se pone a 1 tras conectar la alimentación o al ejecutar CLRWDT o SLEEP.
0. Se pone a 0 por desbordamiento del Perro Guardián WDT.

Bit 3 (flag), PD: Power Down (Apagado).

1. Se pone automáticamente a 1 tras conectar la alimentación Vdd o ejecutar CLRWDT, que resetea el contador WatchDog.
0. Se pone a 0 al ejecutar la instrucción SLEEP.

Bit 2 (flag), Z: Cero

- 1 = El resultado de una operación aritmética o lógica es 0.
0 = El resultado es distinto de 0.

Bit 1 (flag), DC (Digit Carry). Acarreo en el 4º bit de menos peso. Funciona igual que el bit de Carry descrito a continuación. De interés en operaciones en BCD.

Bit 0 (flag), C (Carry). Acarreo en el 8º bit o bit de mas peso. Es el bit de "acarreo" en operaciones de suma AADWF y ADDLW así como también el bit de "llevada" en las instrucciones de sustracción SUBWF y SUBLW. También lo emplean las instrucciones RLF y RRF de rotación de bits.

Suma

1. Se pone a 1 cuando se ha producido acarreo en la suma en el bit de mayor peso con las operaciones AADWF y ADDLW.
0. Se pone a 0 si en la suma no se ha producido acarreo.

Resta

1. Se pone a 1 si en la resta no se ha producido llevada.
0. Se pone a 0 cuando se ha producido llevada en la resta con las operaciones SUBWF y SUBLW.

Registro FSR (04h y 84h): El contenido del FSR se utiliza para el direccionamiento indirecto junto, este registro contiene 8 bits

Registro PORTA (05h), PORTB (06h), PORTC (07h), PORTD (08h) y PORTE (09h): Cada puerto tiene vinculado un registro que almacena la información que entra o sale del microcontrolador.

Registro EEDATA (08h en PIC16F84 y 10Ch en PIC16f877): El registro EEDATA (Datos de EEPROM) guarda el contenido de una posición de la memoria

EEPROM de datos antes de su escritura o después de su lectura, según leamos o escribamos en ella. Para leerla se sigue un proceso especial. La memoria EEPROM es bastante lenta, dato que tendremos en cuenta cuando accedamos a ella para escribirla, pues tarda unos 10 ms en completar el proceso.

Registro EEADR (09h en PIC16F84 y 10Dh en PIC16f877): El registro EEADR (Dirección de EEPROM) guarda la dirección de la posición de memoria EEPROM cuando se accede a ella, bien para su lectura, o bien para su escritura. El registro EEADR puede direccionar como máximo 256 bytes de los cuales sólo los 64 primeros están disponibles, con lo que los dos bits de mayor peso han de tener el valor de '0'.

Registro PCLATH (0Ah y 8Ah): El registro PCLATH (Contador de Programa Alto), guarda la parte alta de la dirección de apuntamiento no accesible por el programador.

Registro INTCON (0Bh y 8Bh): Este registro contiene varios bits de selección de fuentes de interrupción, el bit de activación global de interrupciones y varios flag que indican la causa de una interrupción. Sirve para el control global de las interrupciones y para indicar la procedencia de algunas de ellas, gracias a los bits de estado. Se dispone de cuatro potenciales recursos de interrupción:

- Una fuente externa a través del pin RB0/INT.
- El desbordamiento del temporizador 0 (TMR0).
- Un cambio de estado en los pines RB4 a RB7.
- Programación de la EEPROM de datos.

Cada bit del registro INTCON tiene un significado concreto que se muestra en la siguiente tabla:

Tabla 32. Registro INTCON

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Bit 7, GIE: Habilitación global de interrupciones (Global Interrupt Enable).

- 1: Concede el permiso de interrupciones.
0: Cancela el permiso de las interrupciones.

Bit 6, EEIE: Habilitación de las Interrupciones de la memoria EEPROM.

- 1: Permite que se produzcan interrupciones debidas al fin de escritura de la EEPROM, etc.
0: Este tipo de interrupciones estarán inhibidas.

Bit 5, TOIE: Habilitación de la interrupción del temporizador por desbordamiento (Timer 0 Interrupt Enable).

1: Autoriza las interrupciones debidas al desbordamiento del temporizador.

0: Interrupción del temporizador deshabilita de manera que cuando se produzca el flag correspondiente permanecerá inactivo.

Bit 4, INTE: Habilitación de la entrada de interrupción externa (Interrupt Enable) por patilla RB0/INT.

1: Autoriza las interrupciones provocadas RB0/INT del puerto B.

0: Interrupción externa deshabilita de manera que cuando se produzca una interrupción externa el flag correspondiente permanecerá inactivo.

Bit 3, RBIE: Habilitación de las interrupciones del puerto B (RB Interrupt Enable).

1: Autoriza las interrupciones provocadas por un cambio de estado de las líneas RB4 a RB7 del puerto B.

0: Interrupción del puerto B deshabilitada.

Bit 2 (flag), T0IF: Bit de interrupción de desbordamiento del TMR0.

1: El TMR0 ha rebosado. Se borra por software.

0: El TMR0 no ha rebosado.

Bit 1 (flag), INTF: Bit de interrupción de la Entrada de Interrupción INT (patilla RB0/INT).

1: La entrada de interrupción se ha activado (patilla RB0/INT del puerto B). Se borra por software.

0: No hay interrupción externa.

Bit 0 (flag), RBIF: Bit de interrupción del puerto B.

1: Cambio de estado en una de las líneas de RB4 a RB7 del puerto B. Se borra por software.

0: Ninguna línea de RB4 a RB7 del puerto B ha cambiado.

Cada flag o bandera individual debe ponerse a cero por software.

Solamente hay un vector de interrupción en la memoria de programa (dirección 0004h), por lo que se deben comprobar los bits de INTCON en la subrutina de interrupción para saber cuál es la fuente de la misma. Cuando llega una interrupción, el PIC pone el bit GIE a cero, de forma que no se perturbe el tratamiento de la interrupción en curso, debido a otras interrupciones eventuales. Este bit se pone automáticamente a uno al terminar la subrutina de interrupción, con la ejecución de la instrucción RETFIE.

Los indicadores de interrupciones correspondientes permanecen funcionales incluso cuando no se han autorizado. En este caso también pueden leerse y escribirse todos los bits que componen este registro.

Registro OPTION (80h): El registro OPTION (o registro de opciones) se emplea para programar las opciones del temporizador TMR0, el tipo de flanco con el que se detecta una interrupción y la activación de las resistencias de polarización del puerto B. Ocupa la posición 81h de la página 1 del banco de registros. Debe escribirse usando la instrucción especial OPTION. Esta instrucción carga el contenido de W en el registro OPTION.

Tabla 33. Registro OPTION

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
/RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Bit 7, /RBPU (RB Pull Up). Conexión de las resistencias de polarización del Puerto B. Se conectan todas cuando el puerto B actúa como entrada.

1: Todas las resistencias son desconectadas.

0: Las resistencias se activan de forma individual.

Bit 6, INTDEG (INTerrupt EDGe). Selecciona el tipo de flanco para la interrupción externa. Este bit indica el tipo de flanco de la señal externa que ha de provocar una interrupción en la patilla RB0/INT.

1: La interrupción es producida por el flanco ascendente o de subida.

0: La interrupción es producida por el flanco descendente o de bajada.

Bit 5, T0CS (Timer 0 Signal Source). Selección de la fuente de reloj para el TMR0.

1: TMR0 se usa en modo contador de los pulsos introducidos a través de RA4/T0CKI

0: TMR0 se usa en modo temporizador haciendo uso de los pulsos de reloj internos (Fosc/4).

Bit 4, T0SE (Timer 0 Signal Edge). Tipo de flanco activo de T0CKI (patilla RA4/T0CKI).

1 = El TMR0 se incrementa con el flanco descendente de la señal aplicada a RA4/T0CK1.

0 = El TMR0 se incrementa con el flanco ascendente.

Bit 3, PSA (PreScaler Assignment). Se usa para la asignación del divisor de frecuencias o Prescaler.

1 = El divisor de frecuencia se asigna al WDT.

0 = El divisor de frecuencia se asigna a TMR0.

Bits 0, 1 y 2, PS0, PS1 y PS2 (Prescaler Rate Select Bits). Configura la tasa del valor del divisor de frecuencia o prescaler. Difiere dependiendo que se haya asignado al TMR0 o al WDT.

Registro TRISA y TRISB (85h y 86h): Estos registros son idénticos para el puerto A y el puerto B, con la diferencia de que uno será de 5 bits y otro de 8 bits, el mismo número de bits que tiene cada puerto. Los registros TRIS, también son llamados así, sirven para configurar si los bits de cada puerto serán de entrada o de salida:

- 1: La patilla del puerto correspondiente será de entrada
- 0: En este caso la patilla actuará como una salida.

Tabla 34. Pre-escalier registro OPTION

PS2	PS1	PS0	Divisor TMR0	Divisor WDT
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

Registro EECON1 (88h): Este registro contiene configuraciones importantes acerca de la escritura y la lectura de la EEPROM de datos. En concreto tiene 5 bits de control, cuya distribución y significado es el siguiente.

Tabla 35. Registro EECON1.

U-0	U-0	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
-	-	-	EEIF	WRERR	WREN	WR	RD
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

U (Unimplemented), No implementado. Se lee como 0.

Bit 4 (flag): EEIF. Bit de interrupción de escritura en la memoria EEPROM (EEPROM Interrupt Flag)

- 1: Este bit se pone a uno al terminar la operación de escritura en la EEPROM, y debe ponerse a cero por software

0: No se ha completado la operación de escritura o no ha empezado.

Bit 3 (flag), WRERR. Bit de error de escritura (Write Error)

1: Este bit se pone a 1 si se produce un error de escritura de forma prematura (Reset o Watchdog). En este caso, los contenidos de EEADR y EEDATA no varían, de manera que el proceso pueda ser repetido correctamente.

0: Se ha completado la operación de escritura.

Bit 2, WREN. Bit de habilitación de escritura. (Write Enable)

1: Este bit debe ser habilitado para poder escribir en la EEPROM

0: Deshabilita la escritura de datos en la memoria EEPROM.

Bit 1, WR. Bit de control de escritura (Write Data)

1: Indica que se ha iniciado una operación de escritura. Este bit debe ponerse a uno para escribir un dato.

0: Indica que se ha completado una operación de escritura. El PIC lo pone automáticamente a cero

Bit 0, RD. Bit de control de lectura (Read Data)

1: Inicia una lectura de la memoria EEPROM. Este bit debe ponerse a uno para poder leer un dato.

0: No se ha iniciado una lectura de la EEPROM. El PIC lo pone automáticamente a cero

Registro EECON2 (89h)

Este registro no está implementado físicamente, por lo cual no se puede leer. Tan sólo sirve para un proceso de protección de escritura que consiste en copiar en él unos datos específicos, con el fin de evitar que un programa por error pueda programar la EEPROM, manipulando simplemente los bits del EECON1.

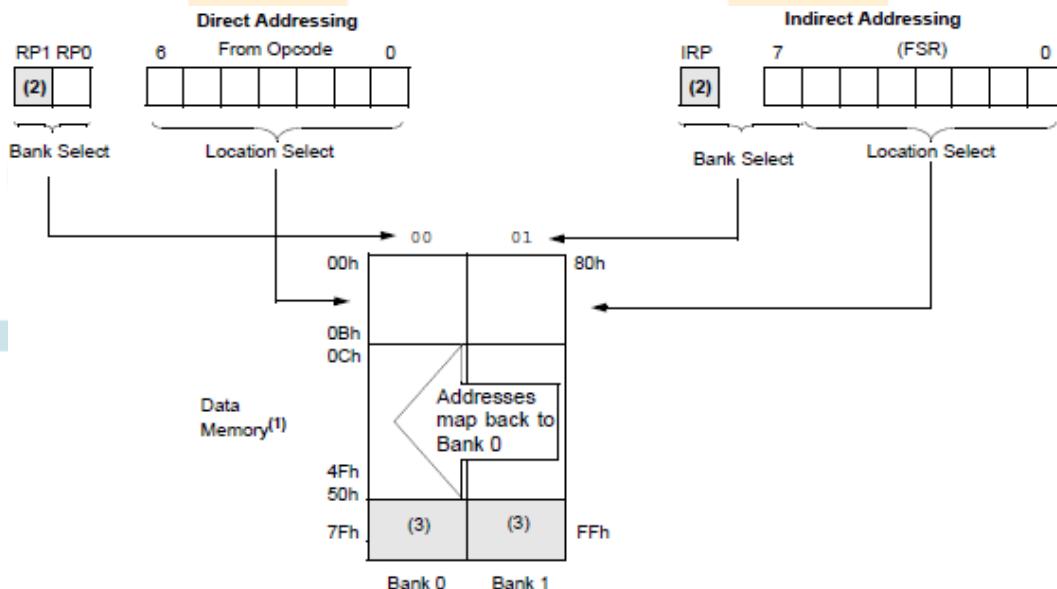
Modos de direccionamiento

Existen tres modos de direccionamiento en los microcontroladores PIC: directo, indirecto y relativo. En el direccionamiento directo se utilizan los valores de RP1 y RP0 para seleccionar el banco y la localización a través del formato de instrucción (OPCODE). En el caso particular de esta familia, el direccionamiento indirecto está determinado por los registros INDF y FSR. Hay que destacar que el registro INDF no es un registro físico, pero a través de él es que se realiza el direccionamiento indirecto. Cualquier instrucción que utilice el registro INDF accede al registro puntero de selección de fila, FSR.

Si se lee solamente el registro INDF el valor de FSR es igual a 0, es decir, la dirección leída es igual 00H. Los 9 bits de dirección efectiva son el resultado de concatenar los 8 bits del registro FSR y el bit 7 (IRP) del registro de estado. El direccionamiento relativo se logra sumando el contenido del registro de trabajo W

al registro contador de programa (PC). Este direccionamiento se utiliza ampliamente en la elaboración de tablas de saltos. La siguiente figura ilustra la diferencia entre el direccionamiento directo y el indirecto.

Figura 101. Diferencia entre direccionamiento directo e indirecto¹¹⁸



Las instrucciones call y return: La instrucción CALL (llamada la subrutina) consigue que la ejecución del programa continúe en la dirección donde se encuentra la subrutina a la que hace referencia. Es similar a GOTO pero coloca en la pila la dirección de la siguiente instrucción que se debe ejecutar después de la CALL. La subrutina finaliza con la instrucción RETURN (Retorno de la subrutina) que retoma la dirección guardada en la pila y la coloca en el contador del programa PC continuando el flujo de control con la instrucción que sigue a la CALL.

En la familia PIC de gama media la pila tiene ocho niveles de memoria del tipo FIFO (primero en entrar, último en salir). Si se produce la llamada a una subrutina durante la ejecución de otra subrutina, la dirección de retorno de esta segunda es colocada en la cima de la pila sobre la dirección anterior. Esta segunda dirección es la primera en salir de la pila mediante la instrucción RETURN. Con la pila de ocho niveles, una subrutina puede llamar a otra y ésta, a su vez, llamar a otra hasta un máximo de ocho. La gama baja sólo puede realizar dos llamadas de este tipo al poseer una pila de sólo dos niveles.

¹¹⁸ Extraído el 18 Octubre de 2009 desde <http://ww1.microchip.com/downloads/en/devicedoc/30430c.pdf>

Las subrutinas deben colocarse al comienzo de las páginas debido a que el bit 8 del contador del programa es puesto a 0 por la instrucción CALL (o por cualquier instrucción que modifica el PC). Las subrutinas deben colocarse en la mitad inicial de las páginas (las 256 palabras).

Memoria EEPROM de datos

Es un tipo de memoria de pocos registros, separada de la memoria de programa ROM y de datos RAM, este módulo de memoria es utilizado para guardar información resultado del control y que no se borra después de des energizar el dispositivo, puede guardar datos de control inicial, backup, o estados iniciales. Pueden ser leída y escrita durante la ejecución del programa, soporta hasta 1.000.000 de ciclos de escritura / borrado, con capacidad de guardar la información por varias décadas, tiene un tiempo de escritura de aproximadamente 10 milisegundos, controlada por el temporizador interno. Para poder leer o escribir en la memoria EEPROM se utiliza varios registros.

EEADR: Registro de direccionamiento de las posiciones de memoria EEPROM.

EEDATA: Registro de paso de datos de 8 bits hacia y desde la memoria EEPROM.

EECON1: Registro de control de operaciones de la EEPROM contiene los bits **RD**, **WR**, **WRERR** y **EIF**.

EECON2: Aunque no es un registro físico, tiene asignada la dirección 0x89, se emplea como registro de seguridad en el ciclo de escritura de la EEPROM, se utilizan dos palabras de control que forman parte del ciclo de escritura.

Ejemplo: En el ciclo de escritura de un microcontrolador PIC16F84 después de cargar la dirección de la EEPROM donde se quiere escribir, cargar el dato que se quiere escribir y colocar el bit **WREN** en uno (1), se debe colocar las palabra 055H (Hexadecimal) en el registro **EECON2** y luego colocar la palabra 0AAH nuevamente en el registro **EECON2**, colocar uno (1) en el bit **WR** y finalmente poner a cero (0) el bit **WREN**.

Interrupciones

En los microcontroladores PIC las fuentes de interrupción están divididas entre las que dependen de cambios de nivel o flancos de subida o bajada y las que dependen de desbordamiento en temporizadores, por ejemplo en un registro temporizador de 8 bits el paso de FFH a 00H.

Ejemplo: En un PIC16F84 las fuentes de interrupción por cambio de nivel se encuentran en los pines INT (RB0) y los pines RB7-RB4. Las fuente de interrupción por desbordamiento se encuentra en el TIMER0.

Para configurar una interrupción en un PIC se debe primero seleccionar el tipo de flanco de activación de la interrupción, si por flanco de bajado o subida (cambio de nivel), en registro OPTION, luego borrar la bandera de interrupción, habilitar la interrupción y finalmente habilitar el bit de interrupciones globales, GIE (registro INTCON).

Ejemplo: En el caso de un microcontrolador PIC16F84 se debe primero configurar el flanco de la interrupción en el bit INTEDG en el registro OPTION, borrar el flag de interrupción, bit correspondiente en el registro INTCON, por ejemplo el flag INTF para la interrupción externa, habilitar la interrupción colocando a uno (1) el bit correspondiente en el registro INTCON, por ejemplo INTE habilita interrupción externa por pin RB0 y finalmente habilitar interrupciones colocando a uno (1) el bit GIE también en INTCON.

Para ejecutar una subrutina de atención a una interrupción, se debe almacenar el contenido del registro W y STATUS en registros de propósito general para no perder la información mientras atiende la interrupción, ejecutar la rutina de interrupción, restaurar los valores de W y STATUS y borrar la bandera correspondiente a la interrupción.

Ejemplo: En el siguiente código los registros W_TEMP y S_TEMP son registros de propósito general creados para guardar la información de W y STATUS respectivamente mientras se atiende la interrupción.

RUTINA	MOVWF W_TEMP ; Salva el contenido de W	
	SWAPF STATUS, 1 ; Intercambio de bits en registro STATUS para salvar datos	
	SWAPF STATUS, 0 ; Intercambio de bits en registro STATUS para salvar datos de STATUS en W	
	MOVWF S_TEMP ; Salva el contenido de STATUS	
	NOP ; Rutina de atención a la interrupción en este caso no hace nada, solo es un ejemplo.	
	SWAPF S_TEMP, 1 ; Restaurando estado de STATUS	
	SWAPF S_TEMP, 0	
	MOVWF STATUS ; Registro STATUS restaurado	
	SWAPF W_TEMP, 1 ; Restaurando W	
	SWAPF W_TEMP, 0	
	BCF INTCON, INTF ; borrar el flag de interrupción activada, para este ejemplo se supone que Se activó la interrupción externa por RB0.	
	RETFIE ; Retorna de atención a interrupción.	

Temporizadores

Los TIMER son módulos ampliamente utilizados en el PIC, el TIMER0 se utiliza para controlar eventos externos mediante el conteo por pin externo, o generar interrupción por desbordamiento. El TIMER1 utilizado para monitorear el tiempo transcurrido entre transiciones en una señal de entrada o controlar el tiempo de transición en una señal de salida. El TIMER2 es utilizado además de temporizador como control del periodo de la señal en el PWM. Para más información sobre este y otros módulos el diseñador debe referirse a las hojas técnicas o “datasheet” de cada dispositivo.

La configuración de un módulo TIMER requiere asignar el pre escalador o post escalador, configurar la fuente de señal del TIMER y demás parámetros en el registro relacionado (OPTION, T2CON, T1CON), finalmente iniciar el Timer.

Ejemplo: En el caso del TIMER0 se inicializa colocando un valor o borrándolo. Es necesario documentarse sobre el funcionamiento del TIMER en el micro en particular, por ejemplo el TIMER0 después de cada desborde (FFH – 00H) debe inicializarse nuevamente, este Timer no tiene forma de parar después de iniciar el conteo, en cambio el TIMER2 posee un bit en el registro T2CON que permite activarlo o desactivarlo en cualquier momento.

El TIMER0 es un módulo con registro de 8 bit, tiene un pre escalador, el cual divide la frecuencia de la señal de reloj, de esta manera se obtienen constantes de tiempo relativamente altas, con base en el ciclo de reloj.

El TIMER1 es un módulo de 16 bits, cuenta con un pre escalador, se puede configurar para generar interrupciones por desborde.

El TIMER2 es también un módulo de 8 bits, solo puede configurarse con fuente de reloj interno, posee un pre escalador y post escalador, solo se puede utilizar como temporizador, aunque como se había mencionado es utilizado en la generación del PWM.

Principales módulos en los PIC

Inicialmente contamos con los pines de I/O como módulos periféricos principales en el micro, otros módulos que se implementan en los microcontroladores PIC y que caracterizan a cada chip individual, en general son los TIMERS, implementan TIMER0 de 8 bits, TIMER1 de 16 bits y TIMER2 de 8 bits con funciones

especiales. También se encuentran otros módulos como el módulo de captura, comparación y PWM denominados CCP, módulo serial síncrono SSP, módulo puerto serial síncrono básico BSSP, módulo puerto serial síncrono maestro MSSP, módulo USART del tipo SCI, módulo de referencia de voltaje, módulo comparador, módulos conversores analógicos digitales A/D de 8 y 10 bits, módulos manejadores de display LCD y módulo de comunicación paralelo esclavo PSP.

Hay módulos que caracterizan los PIC en su arquitectura, funcionalidad, flexibilidad y reducción del sistema de desarrollo, como el módulo de bits de configuración, con lo que puede dotar de características especiales al microcontrolador en su funcionamiento, el módulo POR (Power-On Reset) el cual genera un estado de Reset cuando se alimenta, el módulo BOR (Brown-On Reset) genera un Reset por falla en la alimentación, módulo Watchdog este módulo se compone de un oscilador interno RC para generar un Reset automático en un tiempo configurable, módulo de bajo consumo Sleep tiene la función de colocar el micro en estado de bajo consumo, módulo oscilador interno RC con el que algunos micros funcionan con un oscilador interno para poder utilizar un par de pines extra como entrada/salida o como oscilador externo y finalmente módulo de programación serial permitiendo la fácil programación del PIC incluso en In-Circuit (dentro del circuito, sin sacar el micro de la placa), para la programación en serie utiliza cinco (5) líneas (multiplexadas en los pines), dos líneas de alimentación (Vcc, Gnd), una línea de reloj, una línea de datos y una línea para voltaje de programación (Vpp).

Set de instrucciones

Cada instrucción de los PIC16F84 y PIC16F877 es una palabra de 14 bits, dividida entre el OPCODE, la cual especifica el tipo de instrucción y uno o más operandos según la instrucción. En un PIC de rango medio las instrucciones se pueden clasificar como instrucciones de operaciones de registro orientadas a Byte, instrucciones de operaciones de registro orientadas a bit, instrucciones de operaciones de control e instrucciones de salto (GOTO) y llamado (CALL) para un total de 35 instrucciones, cada instrucción tiene “opcodes” (códigos operativos) que varían entre 3 y 6 bits.

Tabla 36. Formato de instrucciones PIC gama media

Operaciones de registro orientada a Byte																
13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Código Operativo-opcodes						d	f (dirmem-registro)									
										d	destino de la información					
										1	destino el registro "f"					
										0	destino el acumulador W					
Operaciones de registro orientada a Bit																
13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Código Operativo-opcodes						b	b = Bit (pocision)									
										f	(dirmem-registro)					
Operaciones de Control																
13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Código Operativo-opcodes							k (valor-constate)									
										k	valor constante, también llamado literal					
Operaciones de salto (GOTO - CALL)																
13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Código Operativo-opcodes							k (valor-constate)									

Se presenta a continuación el listado de instrucciones de estos microcontroladores.

Tabla 37. Set de Instrucciones

INSTRUCCIONES QUE MANEJAN REGISTROS							
Nemáticos	operandos	Descripción				Ciclos	Flags
ADDWF	f,d	Suma W y f				1	C, DC, Z
ANDWF	f,d	AND W con f				1	Z
CLRF	f	Borra f				1	Z
CLRW	---	Borra W				1	Z
COMF	f,d	Complementa f				1	Z
DECFSZ	f,d	Decrementa f, si es 0 salta				1 (2)	Ninguno

INCF	f,d	Incrementa f	1	Z
INCFSZ	f,d	Incrementa f, si es 0 salta	1	Ninguno
IORWF	f,d	OR entre W y f	1	Z
MOVF	f,d	Mueve f	1	Z
MOVWF	f	Mueve W a f	1	Ninguno
NOP	---	No opera	1	Ninguno
RLF	f,d	Rota f a la izqda. a través del carry	1	C
RRF	f,d	Rota f a la dcha. a través del carry	1	C
SUBWF	f,d	Resta a f el reg. W	1	C, DC, Z
SWAPF	f,d	Intercambia f	1	Ninguno
XORWF	f,d	XOR de W con f	1	Z

INSTRUCCIONES QUE MANIPULAN BITS

BCF	f,b	Borra bit de f	1	Ninguno
BSF	f,b	Pone a 1 el bit de f	1	Ninguno
BTFSC	f,b	Comprueba un bit de f y salta si es 0	1 (2)	Ninguno
BTFSS	f,b	Comprueba un bit de f y salta si es 1	1 (2)	Ninguno

INSTRUCCIONES DE CONTROL Y DE OPERANDOS INMEDIATOS

ANDLW	k	AND inmediato con W	1	Z
CALL	k	Llamada a subrutina	2	Ninguno
CLRWDT	k	Borra Watchdog	1	TO, PD
GOTO	k	Salto incondicional	2	Ninguno
IORLW	k	OR inmediato con W	1	Z
MOVLW	k	Mueve a W un valor inmediato	1	Ninguno
OPTION	k	Carga el registro OPTION	1	Ninguno
RETLW	k	Retorno y carga de W	2	Ninguno
SLEEP	---	Pasa a estado de reposo	1	TO, PD
TRIS	f	Carga el registro	1	Ninguno
XORLW	k	OR exclusiva a W	1	Z

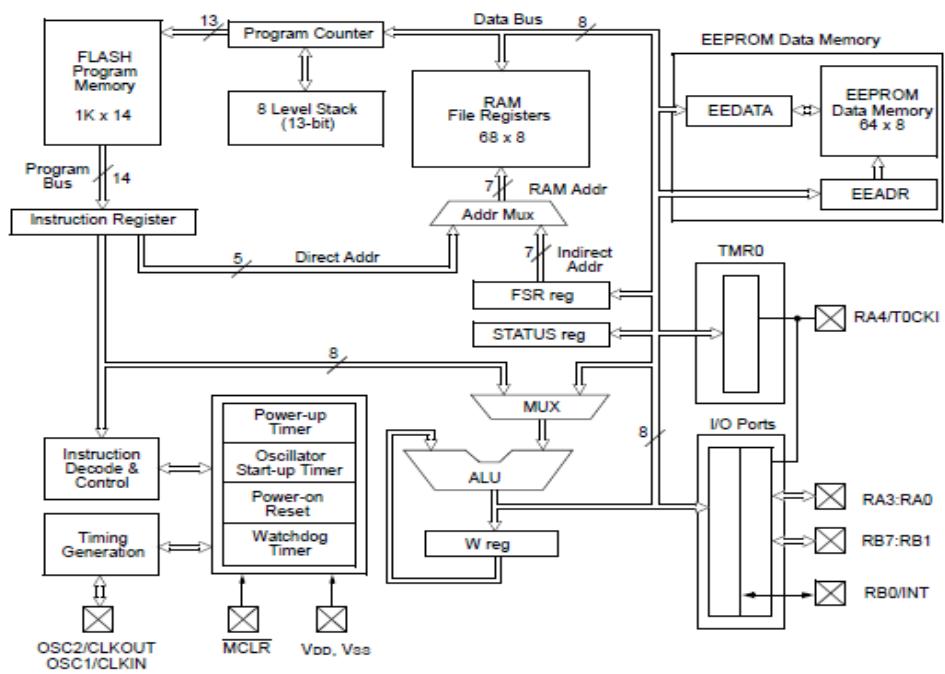
Lección 22: Microcontroladores PIC16F84

Los microcontroladores PIC16F84 son los dispositivos más ampliamente utilizados en el aprendizaje e implementación de proyectos y pequeñas soluciones, incluso en estudiantes de educación media. Estos microcontroladores se destacan por su sencillez en la configuración de pines, su bajo costo, la cantidad de información y proyectos en internet y en revistas especializadas. Como puntos en contra se tiene que el micro no posee sino un solo TIMER, pocas fuentes de interrupción (4 fuentes), no tiene módulos adicionales como ADC, PWM o comparadores, por lo que se hace dispendioso u obsoleto en proyectos o soluciones que requieran estos periféricos, prefiriendo utilizar otros dispositivos.

Arquitectura interna del PIC16F84

El diagrama de bloques de la estructura interna del PIC16F84 se presenta en la siguiente figura.

Figura 102. Arquitectura del PIC16F84¹¹⁹



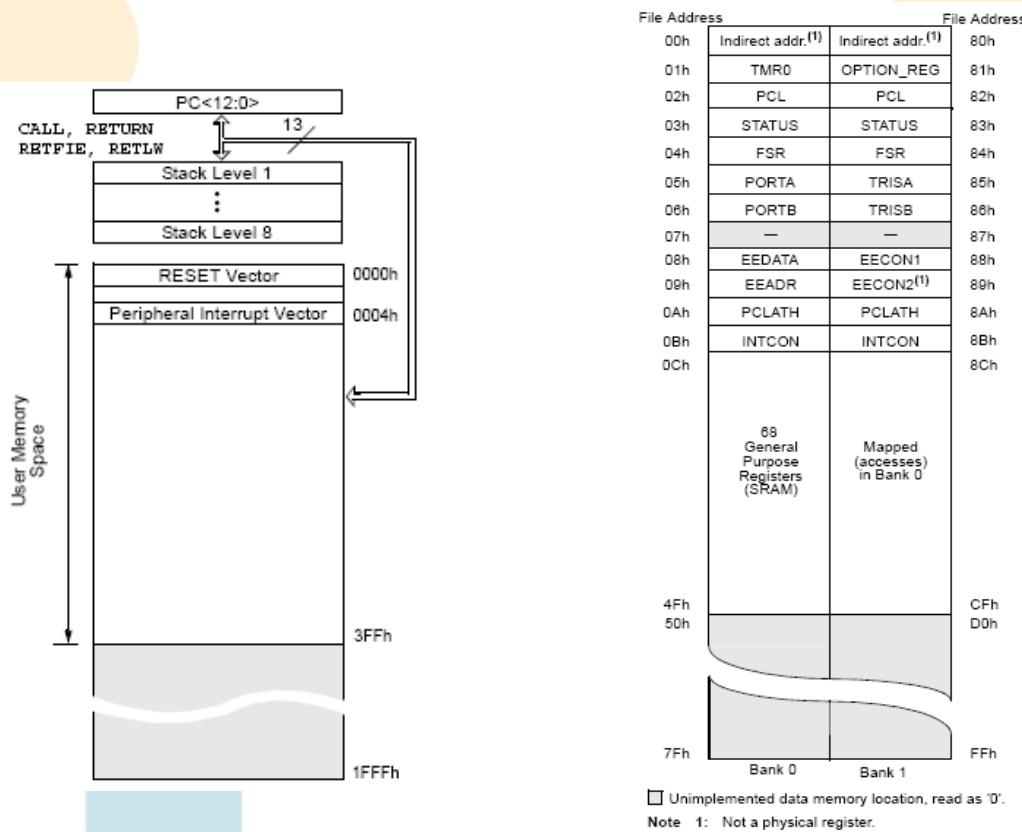
Organización de la memoria en el PIC16F84

La memoria de datos del microcontrolador PIC16F84 se compone de dos bancos de memoria, direccionados mediante los bits indicados en el registro STATUS, en

¹¹⁹ Extraído el 18 Octubre de 2009 desde <http://ww1.microchip.com/downloads/en/devicedoc/30430c.pdf>

esta memoria se alojan registros de propósito especial ya definidos por el fabricante, en su función, configuración y bits, como se puede observar se tienen direcciones de memoria diferentes para cada registro y su denominación predeterminada. Incorpora 68 registros de propósito general que el usuario puede utilizar para almacenar valores de variables, el valor de los registros se refleja en ambos bancos de memoria.

Figura 103. Organización de la memoria¹²⁰



La memoria de datos está partitionada en dos áreas. La primera es para los registros de funciones especiales (SFR), y la segunda área es para los registros de propósito general (GPR). Los registros especiales controlan las operaciones del dispositivo. Toda la memoria de datos puede ser accedida ya sea por direccionamiento directo utilizando direcciones absolutas de cada registro de fila o indirecto a través del registro de selección de fila (FSR).

La memoria de programa tiene implementado 1024 palabras de 14 bits cada una, para el almacenamiento de las instrucciones del programa a ejecutar. También

¹²⁰ Extraído el 18 Octubre de 2009 desde <http://ww1.microchip.com/downloads/en/devicedoc/30430c.pdf>

cuenta con dos vectores empleados para el manejo de las Interrupciones y el estado de Reinicialización (reset) del microcontrolador.

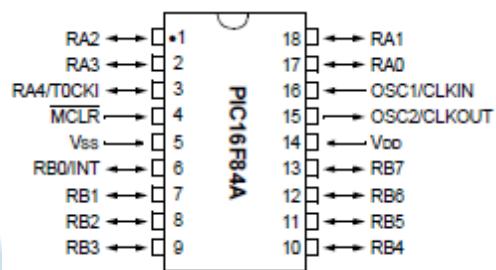
Características generales de los microcontroladores PIC16F84

- Bajo consumo de energía
- Frecuencia de reloj externa máxima de 10MHz
- No posee conversores analógicos/digitales ni digitales/analógicos
- Pipe-line de 2 etapas, 1 para búsqueda de instrucción y otra para la ejecución
- de la instrucción (los saltos ocupan un ciclo más).
- Repertorio de instrucciones reducido (RISC), con tan solo 35 instrucciones
- 4 tipos distintos de instrucciones: orientadas a byte, orientadas a bit, operación entre registros y de salto.
- 1024 palabras de memoria de programa
- Memoria de datos RAM de 68 bytes
- Memoria de datos EEPROM de 64 bytes
- Palabras de instrucción de 14 bits
- Bytes de datos de 8 bits
- 15 registros especiales de función hardware
- 8 niveles de pila
- Modos de direccionamiento: directo, indirecto y relativo
- Cuatro fuentes de interrupción
- TMR0 de 8 bits con pre-escala programable
- Perro guardián (watchdog)

Funciones y Diagrama de pines

La siguiente figura muestra el diagrama de pines del PIC16F84 en encapsulado PDIP, SOIC.

Figura 104. PIC16F84A¹²¹



La descripción de las funciones de los pines de este microcontrolador se presenta en la siguiente tabla.

¹²¹ Extraído el 18 Octubre de 2009 desde <http://ww1.microchip.com/downloads/en/devicedoc/30430c.pdf>

Tabla 38. Descripción de pines PIC16F84¹²²

NOMBRE DEL PIN	TIPO	DESCRIPCION
RA0 a RA3	Entrada/salida	Líneas de E/S digitales del Puerto A
RA4/T0CKI	Entrada/salida	E/S digital o entrada de reloj externo para el TMR0.
RB0/INT a RB7	Entrada/salida	E/S digitales del Puerto B. RB0/INT puede actuar como INT externa. RB4-RB7 pueden provocar una interrupción cuando cambian de estado.
OSC1/CLKIN	Entrada	Entrada de cristal oscilador / entrada de reloj externo.
OSC2/CLKOUT	Salida	Salida del cristal oscilador. En el modo RC, la salida del pin de OSC2/CLKOUT, tiene ¼ de la frecuencia de OSC1 (ciclo de instrucción)
V _{SS}	Alimentación	Tierra para los pines lógicos y de E/S
V _{DD}	Alimentación	Fuente de tensión positiva (Típicamente 5V)
MCLR/V _{PP}	Entrada	Entrada maestra de borrado (Reset)/ voltaje de programación. El reset del dispositivo es activo bajo.

Puertos de entrada / salida (I/O)

El microcontrolador PIC16F84, cuenta con dos puertos bidireccionales: Puerto A y Puerto B. Algunos pines de estos puertos son multiplexados con una función alternativa de los periféricos del dispositivo. En general, cuando un periférico es habilitado, el pin correspondiente no puede ser utilizado como pin de E/S de propósito general. A continuación se describen los puertos de este microcontrolador.

Puerto A: este puerto bidireccional tiene un tamaño de 5 bits (RA4:RA0). Tiene además 2 registros asociados que se muestran en la siguiente tabla y que corresponden a:

- Control de dirección de los pines del puerto A (TRISA)
- Estado de los pines del puerto A (PORTA)

Figura 105. Registro Puerto A y TRISA

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
05h	PORTA	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
85h	TRISA	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are unimplemented, read as '0'.

Puerto B: este puerto bidireccional tiene un tamaño 8 bits (RB7:RB0). Tiene además 4 registros asociados que se muestran en la siguiente tabla.

¹²² Téllez, 2007

Figura 106. Registro Puerto B y TRISB

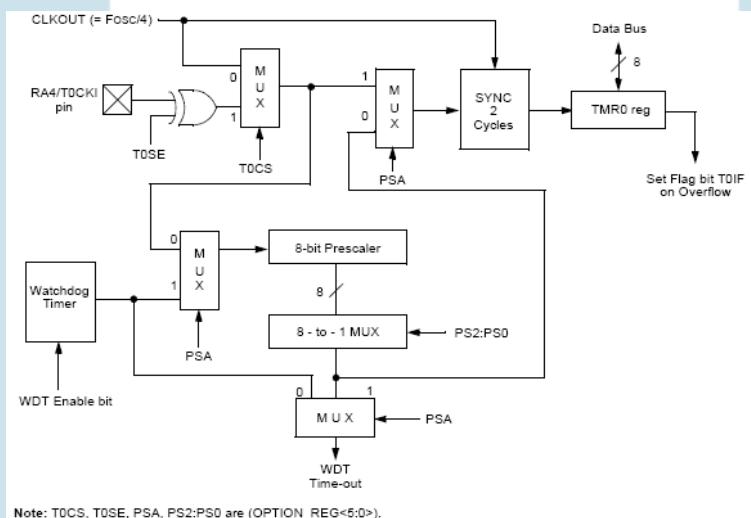
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
08h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	x0xx xxxx	uuuu uuuu
88h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111
81h	OPTION_REG	RBU	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
08h,8Bh	INTCON	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	0000 000u

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

Módulos en el PIC16F84

Timer0: el módulo temporizador/contador de 8 bits TMR0 puede ser leído y escrito continuamente y se puede seleccionar un reloj interno o uno externo para trabajar con el. Además cuenta con preescalador programable de 8 bits, y con interrupción por desbordamiento desde FFh a 00h. El diagrama de bloques del módulo se muestra en la siguiente figura y los registros relacionados con el Timer0 se presentan posteriormente.

Figura 107. Timer0¹²³



Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
01h	TMR0	Timer0 Module Register								x0xx xxxx	uuuu uuuu
0Bh,8Bh	INTCON	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	0000 000u
81h	OPTION_REG	RBU	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
85h	TRISA	—	—	—	PORTA Data Direction Register					---1 1111	---1 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by Timer0.

¹²³ Extraído el 18 Octubre de 2009 desde <http://ww1.microchip.com/downloads/en/devicedoc/30430c.pdf>

Configuración del oscilador

El PIC16F84 puede operar con cuatro configuraciones de oscilador. El usuario puede seleccionar la más conveniente, según su necesidad

- LT: baja potencia de cristal
- XT: cristal/ resonador
- HS: cristal de alta velocidad/ resonador
- CS: resistor/ capacitor

La siguiente tabla muestra los valores de capacitor para las configuraciones: LP, XT y HS; junto con el valor del rango de frecuencia de resonancia.

Figura 108. Configuración del Oscilador

Mode	Freq	OSC1/C1	OSC2/C2
LP	32 kHz	68 - 100 pF	68 - 100 pF
	200 kHz	15 - 33 pF	15 - 33 pF
XT	100 kHz	100 - 150 pF	100 - 150 pF
	2 MHz	15 - 33 pF	15 - 33 pF
	4 MHz	15 - 33 pF	15 - 33 pF
HS	4 MHz	15 - 33 pF	15 - 33 pF
	20 MHz	15 - 33 pF	15 - 33 pF

Interrupciones

El PIC16F84 tiene cuatro fuentes de interrupción:

- Interrupción externa a través del pin RBO/INT
- Interrupción por desbordamiento del TMR0
- Interrupción PORTB (RB7:RB4)
- Interrupción de la memoria EEPROM cuando la escritura ha finalizado.

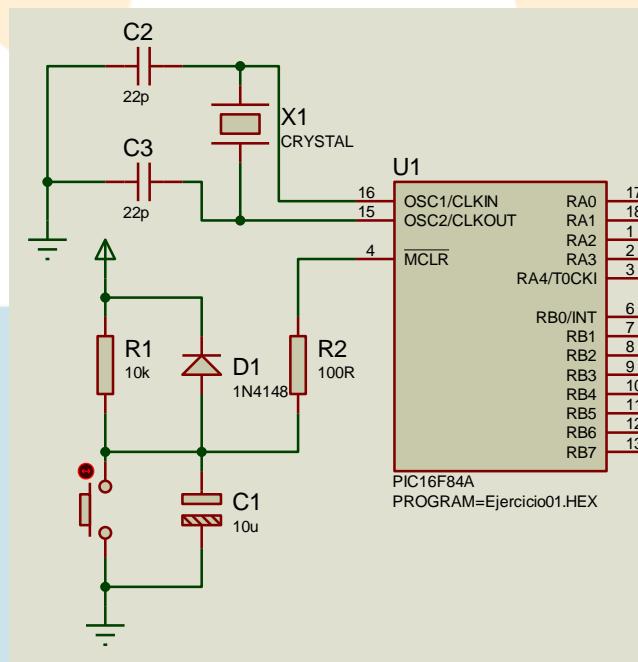
Watchdog (perro guardián)

El temporizador *watchdog* es un recurso de vigilancia del microcontrolador. Cuenta libremente con el oscilador RC del chip, por lo tanto no necesita componentes externos. Tiene un período nominal de finalización de 18ms. Durante la operación normal del microcontrolador, una finalización del temporizador *watchdog* genera un reinicio o RESET; pero si el dispositivo está en modo SLEEP provoca que se retome nuevamente la operación normal del dispositivo. El *watchdog* puede ser habilitado o deshabilitado por software.

Circuito de Reset

El circuito más usual para conexión del PIC16F84 y en general para cualquier PIC y que incorpora el circuito POR se puede observar en la figura.

Figura 109. Circuito Reset (con circuito POR) y Oscilador externo y para PIC16F84.



EEPROM de datos

La EEPROM en el PIC16F84 consta de 256 posiciones de memoria de 8 bit o 1 Byte de capacidad cada una, lo que permite alojar información de constantes, parámetros iniciales o datos que se generen durante el funcionamiento del programa de control.

Lección 23: Ejemplos de programación con PIC16F84

La comprensión y desarrollo práctico de estos ejercicios permite una mayor asimilación, aprendizaje y habilidad en la programación, prueba y ejecución de aplicaciones basadas en microcontroladores. En esta lección se plantean varios ejercicios de desarrollo e implementación con el fin de que el lector adquiera las

competencias y habilidades necesarias no solo para entender la arquitectura y funcionamiento del dispositivo también para facilitar el aprendizaje de nuevos dispositivos.

Mplab IDE – Intermitencia de un LED

Este ejercicio explora las directivas básicas de programación y la utilización de las instrucciones más habituales.

Entradas y salidas: en este caso no hay entradas solo una salida donde se conectara un diodo LED, elegimos el puerto “B” en su pin RB0 como salida de datos.

Pseudocódigo:

INICIA:

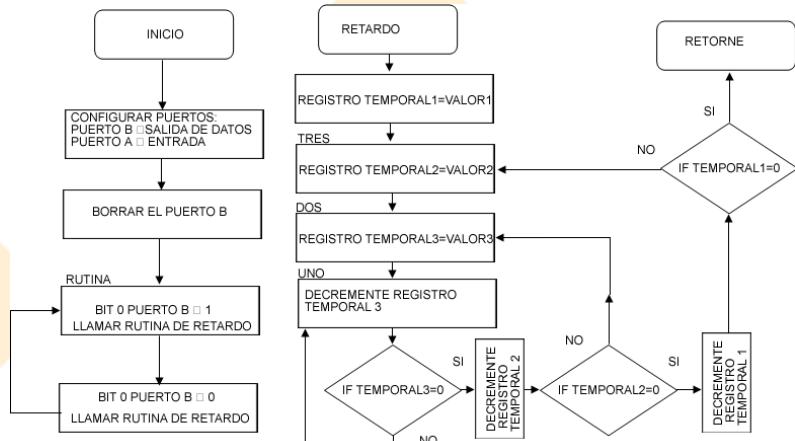
 CONFIGURAR PUERTOS:
 PUERTO B → SALIDA DE DATOS
 PUERTO A → ENTRADA
 RUTINA:
 BORRAR EL PUERTO B
 BIT 0 PUERTO B → 1
 LLAMAR RUTINA DE RETARDO
 BIT 0 PUERTO B → 0
 LLAMAR RUTINA DE RETARDO
 SALTAR A RUTINA

TERMINA

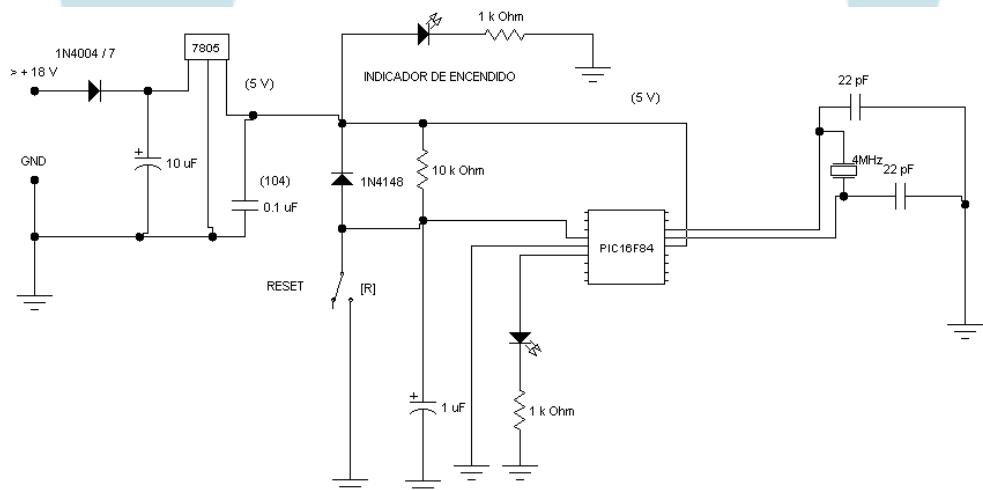
SUBRUTINA “RETARDO”:

 REGISTRO TEMPORAL1=VALOR1
 TRES REGISTRO TEMPORAL2=VALOR2
 DOS REGISTRO TEMPORAL3=VALOR3
 UNO DECREMENTE REGISTRO TEMPORAL 3
 IF TEMPORAL3=0
 THEN DECREMENTE TEMPORAL2
 IF TEMPORAL2=0
 THEN DECREMENTE TEMPORAL1
 IF TEMPORAL1=0
 THEN RETORNE
 ELSE SALTAR A → TRES
 ELSE SALTAR A → DOS
 ELSE SALTAR A → UNO
TERMINA

Diagrama de flujo:



Montaje:



Código ensamblador

```

TITLE "PIC16F84A EEPROM PROGRAM"
LIST P=16F84A,F=INHX32
#INCLUDE <P16F84A.INC>
;INGENIERO ELECTRONICO HECTOR URIEL VILLAMIL GONZALEZ
;ENCENDIDO INTERMITENTE DE UN LED (PARA EFECTOS PRACTICOS SE INCORPORA UN RETARDO)
;PINES DE SALIDA LEDs ==> B0
;PINES DE ENTRADA ==> "NINGUNO"
;-----
;REGISTROS DE PROPOSITO ESPECIAL MAS UTILIZADOS
INDF EQU 00H ;REGISTRO PARA DIRECCIONAMIENTO INDIRECTO
TMR0 EQU 01H ;REGISTRO TIMER 0
PCL EQU 02H ;PROGRAM COUNTER PARTE BAJA
STATUS EQU 03H ;REGISTRO DE ESTADO DEL PIC
FSR EQU 04H ;REGISTRO UTILIZADO COMO APUNTADOR EN DIR. INDIRECTO
PTA EQU 05H ;PUERTO A DEL PIC
PTB EQU 06H ;PUERTO B DEL PIC
EEDATA EQU 08H ;DATO IN/OUT EN EEPROM
EEADR EQU 09H ;DIRECCION DE IN/OUT DE EEPROM
  
```

```

PCLATCH EQU 0AH ;CERROJO PARTE ALTA DEL PROGRAM COUNTER
INTCON EQU 0BH ;REGISTRO DE INTERRUPCIONES
OPCION EQU 81H ;OPCIONES DE CONFIGURACION INT TMP PULLUP PREESCALER
TRISA EQU 85H ;RA0,RA1,RA2,RA3,RA4 ENTRADAS
TRISB EQU 86H ;RB0,RB1,RB2,RB3,RB4,RB5,RB6,RB7 COMO SALIDAS
EECON1 EQU 88H ;B4=EEIF/FIN WR/*B3=WRERR/ERROR WR/*B2=WREN/PERMISO
WR/*B1/WR/*B0/RD/
EECON2 EQU 89H ;REG SEGURIDAD PROCESO EEPROM
;-----
;DEFINICION REGISTROS DE PROPOSITO GENERAL PARA ESTE PROGRAMA
TMP1 EQU 0CH ;REGISTRO TEMPORAL 1 SIRVE AL CICLO MAS EXTERNO DEL RETARDO
TMP2 EQU 0DH ;REGISTRO TEMPORAL 2 SIRVE AL CICLO INTERMEDIO DEL RETARDO
TMP3 EQU 0EH ;REGISTRO TEMPORAL 3 SIRVE AL CICLO MAS INTERNO DEL RETARDO
;-----
;DEFINICION DE BITS
W EQU 0 ;REGISTRO DE TRABAJO
F EQU 1 ;REGISTRO
C EQU 0 ;FLAG DE CARRY
Z EQU 2 ;FLAG DE CERO
RD EQU 0 ;RD=1 CICLO DE LECTURA DE LA EEPROM
WR EQU 1 ;WR=1 CICLO DE ESCRITURA DE LA EEPROM
WREN EQU 2 ;WREN=1 AUTORIZA PERMISO DE ESCRITURA EN LA EEPROM
B0 EQU 0 ;BIT 0
B1 EQU 1 ;BIT 1
B2 EQU 2 ;BIT 2
B3 EQU 3 ;BIT 3
B4 EQU 4 ;BIT 4
B5 EQU 5 ;BIT 5
B6 EQU 6 ;BIT 6
B7 EQU 7 ;BIT 7
;-----
;DEFINICION DE CONSTANTES
VAL1 EQU 20H ;CONSTANTE DE RETARDO RECARGA AL REGISTRO TMP1
VAL2 EQU 30H ;CONSTANTE DE RETARDO RECARGA AL REGISTRO TMP2
VAL3 EQU 40H ;CONSTANTE DE RETARDO RECARGA AL REGISTRO TMP3
;-----
;MACROS
#define BANK1 BSF STATUS,5 ;ENCARGADO DE PASAR AL BANCO DE MEMORIA CERO
#define BANK0 BCF STATUS,5 ;ENCARGADO DE PASAR AL BANCO DE MEMORIA UNO
;-----
;INICIO DEL PROGRAMA
;-----
ORG 00H
;-----
;CONFIGURACION DE PUERTOS
;-----
BANK1
MOVLW 00H ;SALIDA DE DATOS POR LOS PINES B7 B6 B5 B4 B3 B2 B1 B0
MOVWF TRISB
MOVLW 1FH ;ENTRADA DE DATOS A4 A3 A2 A1 A0
MOVWF TRISA
BANK0
;-----
;PROGRAMA DE INICIO
;-----
INICIO CLRF PTB ;BORRAR EL PUERTO SE APAGAN LOS LEDs
RUTIN BSF PTB,0 ;COLOCA UN UNO EN EL BIT CERO DE PUERTO B
CALL RET ;LLAMAR A RUTINA DE RETARDO
BCF PTB,0 ;COLOCA UN CERO EN EL BIT CERO DE PUERTO B
CALL RET ;LLAMAR A RUTINA DE RETARDO
GOTO RUTIN ;COMENZAR DE NUEVO LA RUTINA
;-----
;LLAMA LA RUTINA DE RETARDO ESPECIAL VARIABLE
;-----
RET MOVLW VAL1 ;VALOR MAXIMO / MINIMO VARIABLE DE TEMPORIZADOR
MOVWF TMP1 ;REGISTRO TEMPORAL 1 TIEMPOS GRANDES
TRES MOVLW VAL2 ;VALOR FIJO
MOVWF TMP2 ;REGISTRO TEMPORAL 2 TIEMPOS MEDIANOS
DOS MOVLW VAL3 ;VALOR FIJO

```

```

    UNO      MOVWF TMP3           ;REGISTRO TEMPORAL 3 TIEMPOS PEQUEÑOS
    DECFSZ TMP3, F
    GOTO UNO
    DECFSZ TMP2, F
    GOTO DOS
    DECFSZ TMP1, F
    GOTO TRES
    RETURN
    ;-----
    END
    ;-----
```

Mplab IDE – Encendido y Apagado de un LED utilizando un interruptor

Este ejercicio propone recibir la señal de un pulsador y reflejar este estado mediante el encendido de un LED en caso de ser pulsado y apagar el LED en caso de soltar el pulsador.

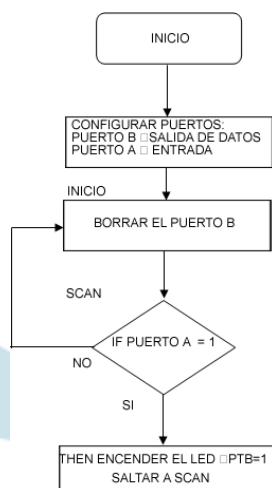
Entradas y salidas: para este ejercicio existe un pulsador el cual es conectado al puerto A en su pin RA0, hay una salida donde se conectara un diodo LED, elegimos el puerto “B” en su pin RB0 como salida de datos.

Pseudocódigo:

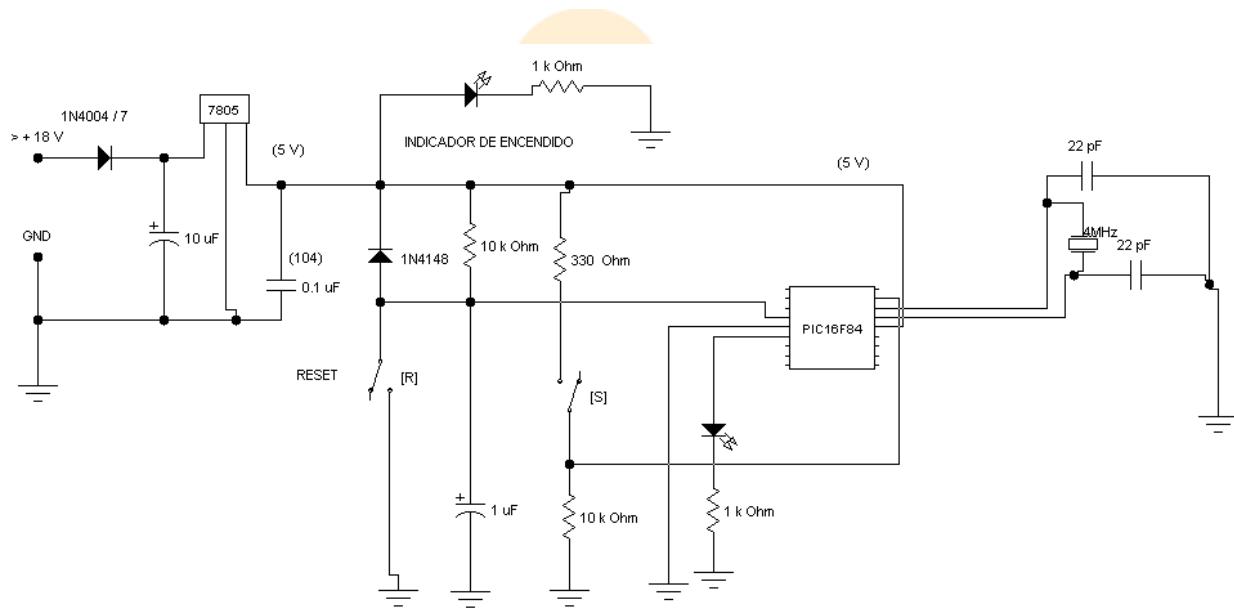
```

INICIA:
    CONFIGURAR PUERTOS:
        PUERTO B → SALIDA DE DATOS
        PUERTO A → ENTRADA
    INICIO: BORRAR EL PUERTO B
    SCAN:   IF PUERTO A → 1
            THEN ENCENDER EL LED → PTB=1
            SALTAR A SCAN
        ELSE SALTAR A INICIO
TERMINA
```

Diagrama de flujo:



Montaje



Código ensamblador

```

TITLE "PIC16F84A EEPROM PROGRAM"
LIST P=16F84A,F=INHX32
#INCLUDE <P16F84A.INC>
;INGENIERO ELECTRONICO HECTOR URIEL VILLAMIL GONZALEZ
;ENCENDIDO DE UN LED POR ACCION DE UN PULSADOR(PARA EFECTOS PRACTICOS SE INCORPORA UN
RETARDO)
;PINES DE SALIDA LEDS ==> B0
;PINAS DE ENTRADA ==> A0 PULSADOR
;-----
;REGISTROS DE PROPOSITO ESPECIAL
INDF EQU 00H ;REGISTRO PARA DIRECCIONAMIENTO INDIRECTO
TMR0 EQU 01H ;REGISTRO TIMER 0
PCL EQU 02H ;PROGRAM COUNTER PARTE BAJA
STATUS EQU 03H ;REGISTRO DE ESTADO DEL PIC
FSR EQU 04H ;REGISTRO UTILIZADO COMO APUNTADOR EN DIR. INDIRECTO
PTA EQU 05H ;PUERTO A DEL PIC
PTB EQU 06H ;PUERTO B DEL PIC
EEDATA EQU 08H ;DATO IN/OUT EN EEPROM
EEADR EQU 09H ;DIRECCION DE IN/OUT DE EEPROM
PCLATCH EQU 0AH ;CERROJO PARTE ALTA DEL PROGRAM COUNTER
INTCON EQU 0BH ;REGISTRO DE INTERRUPCIONES
OPCION EQU 81H ;OPCIONES DE CONFIGURACION INT TMP PULLUP PREESCALER
TRISA EQU 85H ;RA0=PULSADOR,RA1,RA2,RA3,RA4 ENTRADAS
TRISB EQU 86H ;RB0,RB1,RB2,RB3,RB4,RB5,RB6,RB7 COMO SALIDAS
EECON1 EQU 88H ;B4=EEIF/FIN WR/*B3=WRERR/ERROR WR/*B2=WREN/PERMISO WR/*B1/WR/*B0/RD/
EECON2 EQU 89H ;REG SEGURIDAD PROCESO EEPROM
;-----
;DEFINICION REGISTROS DE PROPOSITO GENERAL
TMP1 EQU 0CH ;REGISTRO TEMPORAL 1 SIRVE AL CICLO MAS EXTERNO DEL RETARDO
TMP2 EQU 0DH ;REGISTRO TEMPORAL 2 SIRVE AL CICLO INTERMEDIO DEL RETARDO
TMP3 EQU 0EH ;REGISTRO TEMPORAL 3 SIRVE AL CICLO MAS INTERNO DEL RETARDO
;-----
;DEFINICION DE BITS
W EQU 0 ;REGISTRO DE TRABAJO

```

```

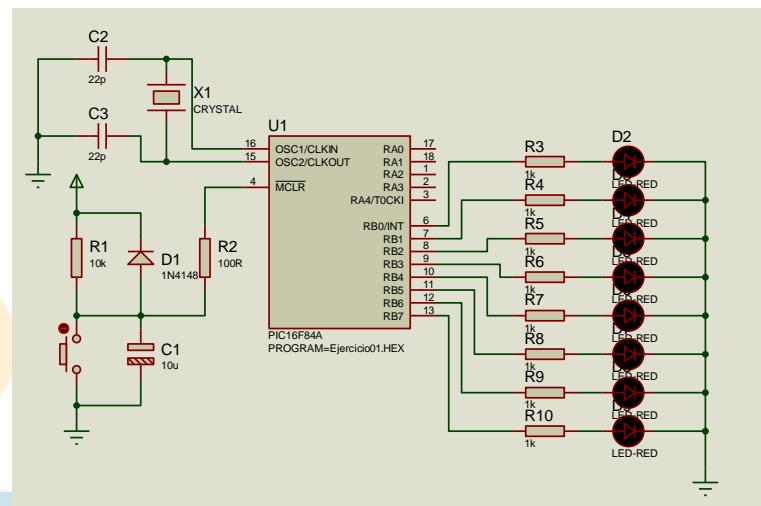
F EQU 1 ;REGISTRO
C EQU 0 ;FLAG DE CARRY
Z EQU 2 ;FLAG DE CERO
RD EQU 0 ;RD=1 CICLO DE LECTURA DE LA EEPROM
WR EQU 1 ;WR=1 CICLO DE ESCRITURA DE LA EEPROM
WREN EQU 2 ;WREN=1 AUTORIZA PERMISO DE ESCRITURA EN LA EEPROM
B0 EQU 0 ;BIT 0
B1 EQU 1 ;BIT 1
B2 EQU 2 ;BIT 2
B3 EQU 3 ;BIT 3
B4 EQU 4 ;BIT 4
B5 EQU 5 ;BIT 5
B6 EQU 6 ;BIT 6
B7 EQU 7 ;BIT 7

;-----:DEFINICION DE CONSTANTES
VAL1 EQU 20H ;CONSTANTE DE RETARDO RECARGA AL REGISTRO TMP1
VAL2 EQU 30H ;CONSTANTE DE REATRDO RECARGA AL REGISTRO TMP2
VAL3 EQU 40H ;CONSTANTE DE RETARDO RECARGA AL REGISTRO TMP3
;-----:MACROS
#define BANK1 BSF STATUS, 5 ;ENCARGADO DE PASAR AL BANCO DE MEMORIA CERO
#define BANK0 BCF STATUS, 5 ;ENCARGADO DE PASAR AL BANCO DE MEMORIA UNO
;-----:INICIO DEL PROGRAMA
;-----ORG 00H
;-----:CONFIGURACION DE PUERTOS
;-----BANK1
MOVLW 00H ;SALIDA DE DATOS POR LOS PINES B7 B6 B5 B4 B3 B2 B1 B0
MOVWF TRISB
MOVLW 1FH ;ENTRADA DE DATOS A4 A3 A2 A1 A0
MOVWF TRISA
BANK0
;-----:PROGRAMA DE INICIO
;-----INICIO CLRF PTB
SCAN BTFSS PTA, B0 ;BORRADO DEL PUERTO B
GOTO INICIO ;PRUEBE BIT 0 DEL PUERTO A (RA0)
BSF PTB, B0 ;SI ES CERO SALTA A INICIO Y NO ENCIEDE EL LED
GOTO SCAN ;SI ES UNO ENCIEDE EL LED
;-----END
;-----
```

Mplab IDE – Manejo de puertos “secuencias de LEDs controladas por botones”

Para el desarrollo de esta implementación se comparte el código fuente y el circuito, se infiere que el estudiante debe deducir el pseudocódigo o algoritmo utilizado siguiendo el desarrollo del programa y su documentación como ejercicio de aprendizaje y comprensión de las instrucciones y funcionamiento del dispositivo.

Figura 110. Diagrama de conexión



Código fuente para la implementación de la secuencia de LEDs

```

TITLE "PIC16F84A EEPROM PROGRAM"
LIST P=16F84A,F=INHX32
#INCLUDE <P16F84A.INC>
;INGENIERO ELECTRONICO HECTOR URIEL VILLAMIL GONZALEZ
;SECUENCIA DE LEDS PINES DE SALIDA LEDS ==> B7 B6 B5 B4 B3 B2 B1
;ORDENES A0=INCREMENTA SECUENCIA A1=DECREMENTA SECUENCIAS A2=INCREMENTA VELOCIDAD
;A3=DECREMENTA VELOCIDAD ==>PULSADORES SE UTILIZAN COMO ENTRADAS
PCL EQU 02H ;PROGRAM COUNTE PARTE BAJA
STATUS EQU 03H ;REGISTRO DE ESTADO DEL PIC
PTA EQU 05H ;PUERTO A DEL PIC
PTB EQU 06H ;PUERTO B DEL PIC
EEDATA EQU 08H ;DATO IN/OUT EN EEPROM
EEADR EQU 09H ;DIRECCION DE IN/OUT DE EEPROM
INTCON EQU 0BH ;REGISTRO DE INTERRUPCIONES
OPCION EQU 81H ;OPCIONES DE CONFIGURACION INT TMP PULLUP PREESCALER
TRISA EQU 85H ;/RA7-RA6-RA5=NOP/RA4=RXRS(CABL-ESCL)/RA3=TEST SENSORES/RA2-RA1-
RA0=TECLD
TRISB EQU 86H ;/RB7=TXRS232/RB6=CLK2/RB5=CLK1/RB4-RB3-RB2-RB1=(MULTIPLX-'D')/RB0=INT/
EECON1 EQU 88H ;B4=EIF/FIN WR/*B3=WRERR/ERROR WR/*B2=WREN/PERMISO WR/*B1=WR/*B0/RD/
EECON2 EQU 89H ;REG SEGURIDAD PROCESO EEPROM
;DEFINICION REGISTROS FSR
SEC EQU 0CH ;REGISTRO CONTROLADOR DE SECUENCIAS
BOT EQU 0DH ;BORRAR REGISTRO DE ALMACENAMIENTO Y CONTROL DE PULSADORES
CONRT EQU 0EH ;REGISTRO DE CONTROL DE RUTINAS
VMAX EQU 0FH ;SUB-RUTINA DE VELOCIDAD MAXIMA
TMP1 EQU 10H ;REGISTRO TEMPORAL 1 TIEMPOS MEDIANOS
TMP2 EQU 11H ;REGISTRO TEMPORAL 2 TIEMPOS MEDIANOS
TMP3 EQU 12H ;REGISTRO TEMPORAL 3 TIEMPOS MEDIANOS
CONT EQU 13H ;REGISTRO CONTADOR PARA VALORES DE LA TABLA
;DEFINICION DE BITS
W EQU 0 ;REGISTRO DE TRABAJO
F EQU 1 ;REGISTRO
C EQU 0 ;FLAG DE CARRY
Z EQU 2 ;FLAG DE CERO
RD EQU 0 ;RD=1 CICLO DE LECTURA DE LA EEPROM
WR EQU 1 ;WR=1 CICLO DE ESCRITURA DE LA EEPROM
WREN EQU 2 ;WREN=1 AUTORIZA PERMISO DE ESCRITURA EN LA EEPROM
B0 EQU 0 ;BIT 0
B1 EQU 1 ;BIT 1
B2 EQU 2 ;BIT 2
B3 EQU 3 ;BIT 3

```

```

B4    EQU   4      ;BIT 4
B5    EQU   5      ;BIT 5
B6    EQU   6      ;BIT 6
B7    EQU   7      ;BIT 7
;DEFINICION DE CONSTANTES
VAL1  EQU   20H
VAL2  EQU   10H
;DEFINICION DE DIRECCIONES EEPROM
STAEE EQU   00H    ;/B7=NOP/B6=NOP/B5=NOP/B4=OFF*LCD/B3=LIGHT*LCD/B2=ALAR*ON:OFF/B1-
B2=CLAVOPC
;MACROS
#define BANK1  BSF    STATUS, 5
#define BANK0  BCF    STATUS, 5
;INICIO DEL PROGRAMA
;INICIO DEL PROGRAMA
    ORG   00H
    CALL  CONFIG
    GOTO INICIO
    ORG   10H
;CONFIGURACION DE PUERTOS
CONFIG BANK1
    MOVLW 00H    ;SALIDA DE DATOS POR LOS PINES B7 B6 B5 B4 B3 B2 B1 DATOS CONSIGNADOS EN
LA EEPROM
    MOVWF TRISB
    MOVLW 1FH    ;ENTRADA DE DATOS A ALMACENAR BOTONES A3 A2 A1 A0
    MOVWF TRISA  ;====> A4 =AUTORIZACION DE ESCRITURA EN EEPROM
    BANK0
    RETURN
;PROGRAMA DE INICIO Y CONTROL DE TX RS232
INICIO MOVLW 64H
    MOVWF VMAX  ;VALOR INICIAL DEL RETARDO
    CLRF  CONRT  ;VALOR DE RUTINA INICIAL
    CLRF  SEC    ;REGISTRO CONTROLADOR DE SECUENCIAS
    CLRF  BOT    ;BORRAR REGISTRO DE ALMACENAMIENTO Y CONTROL DE PULSADORES
MENU  MOVF  CONRT, W  ;REGISTRO DE CONTROL DE RUTINAS
    XORLW 01H
    BTFSC STATUS, Z  ;¿PRIMERA RUTINA?
    GOTO  RUT1
    MOVF  CONRT, W  ;REGISTRO DE CONTROL DE RUTINAS
    XORLW 02H
    BTFSC STATUS, Z  ;¿SEGUNDA RUTINA?
    GOTO  RUT2
    MOVF  CONRT, W  ;REGISTRO DE CONTROL DE RUTINAS
    XORLW 03H
    BTFSC STATUS, Z  ;¿TERCERA RUTINA?
    GOTO  RUT3
    MOVF  CONRT, W  ;REGISTRO DE CONTROL DE RUTINAS
    XORLW 04H
    BTFSC STATUS, Z  ;¿CUARTA RUTINA?
    GOTO  RUT4
    CALL  BOTN
    CLRF  BOT
    GOTO  MENU
;RUTINA DE EVALUACION DE BOTONES PULSADORES UP - DW ** VELMAX - VEL MIN
BOTN  MOVF  PTA, W
    XORLW 00H
    BTFSC STATUS, Z  ;¿SE TIENE PULSADO ALGUNA TECLA?
    GOTO  BOTN3
    BTFSC BOT, B4  ;¿YA FUE PULSADO EL TECLADO?
    GOTO  BOTN1
    MOVF  PTA, W
    MOVWF BOT
    CALL  MENU2
    BSF   BOT, B4  ;REISTRO GUARDA LOS ESTADOS DE LOS PULSADORES
    MOVF  PTA, W
    XORLW 00H
    BTFSS STATUS, Z  ;AJUSTE EL BIT DE CONTROL DE TECLADO !!!BOTON PULSADO!!
    RETURN
BOTN1 MOVF  BOT, W
    ;LAS TECLAS NO ESTAN OPRIMIDAS
    
```

```

ANDLW B'00010000'
MOVWF BOT
GOTO BOTN2
;MENU DE VELOCIDAD Y SECUENCIA
MENU2 MOVF BOT, W
ANDLW OFH
BTFSB BOT, B0 ;¿INCREMENTAR LA SECUENCIA?
GOTO UP ;SUB-RUTINA AUMENTA LA SECUENCIA
MOVF BOT, W
ANDLW OFH
BTFSB BOT, B1 ;¿DECREMENTA LA SECUENCIA?
GOTO DOW ;SUB-RUTINA DECREMENTA LA SECUENCIA
MOVF BOT, W
ANDLW OFH
BTFSB BOT, B2 ;¿INCREMENTA LA VELOCIDAD DE LA SECUENCIA?
GOTO VELMX ;SUB-RUTINA DE VELOCIDAD MAXIMA
MOVF BOT, W
ANDLW OFH
BTFSB BOT, B3 ;¿DECREMENTA LA VELOCIDAD DE LA SECUENCIA?
GOTO VELMIN ;SUBRUTINA DE VELOCIDAD MINIMA
MENU21 RETURN
;SUB-RUTINA AUMENTA LA SECUENCIA
UP INCF CONRT, F ;INCREMENTA EL VALOR DE SECUENCIA
MOVF CONRT, W
XORLW 05H ;MAXIMO NUMERO DE SECUENCIAS
BTFSB STATUS, Z ;¿YA LLEGO AL MAXIMO NUMERO DE SECUENCIAS?
CLRF CONRT ;PONE EN CERO LAS RUTINAS INCREMENTA EL VALOR DE SECUENCIA
GOTO MENU21
;SUB-RUTINA DECREMENTA LA SECUENCIA
DOW DECF CONRT, F ;INCREMENTA EL VALOR DE SECUENCIA
MOVF CONRT, W
SUBLW 00H ;DETERMINA SI EL VALOR DE LA RUTINA ES MENOR DE CERO
BTFSB STATUS, C ;¿YA LLEGO A CERO EL VALOR DE SECUENCIA?
GOTO MENU21
MOVLW 04H ;MAXIMO NUMERO DE SECUENCIA
MOVWF CONRT ;INCREMENTA EL VALOR DE SECUENCIA
GOTO MENU21
;SUB-RUTINA DE VELOCIDAD MAXIMA
VELMX MOVLW 05H
ADDWF VMAX, F ;INCREMENTA EL VALOR DE SECUENCIA
MOVF VMAX, W ;REGISTRO DE MANEJO DE VELOCIDAD DE TEMPORIZADOR
XORLW 0C8H ;MAXIMO NUMERO DE TEMPORIZADOR
BTFSB STATUS, Z ;¿YA LLEGO AL MAXIMO NUMERO DE TEMPORIZADOR?
GOTO MENU21
MOVLW 0C3H ;REINICIAR AL MAXIMO - 5
MOVWF VMAX ;REGISTRO DE MANEJO DE VELOCIDAD DE TEMPORIZADOR
GOTO MENU21
;SUBRUTINA DE VELOCIDAD MINIMA
VELMIN MOVLW 05H
SUBWF VMAX, F ;DECREMENTA EL VALOR DE SECUENCIA
MOVF VMAX, W ;REGISTRO DE MANEJO DE VELOCIDAD DE TEMPORIZADOR
XORLW 0A0H ;MINIMO VALOR DE RETARDO
BTFSB STATUS, Z ;¿YA LLEGO A CERO EL VALOR DE SECUENCIA?
GOTO MENU21
MOVLW 0FH ;MINIMO NUMERO DE TEMPORIZADOR + 1
MOVWF VMAX ;REGISTRO DE MANEJO DE VELOCIDAD DE TEMPORIZADOR
GOTO MENU21
;RUTINA DE SALIDA DE DATOS POR PUERTOS DE PINES PTB
OUT MOVF SEC, W ;CARGA LA SECUENCIA EN EL ACUMULADOR
MOVWF PTB ;CARGA EL VALOR DE SECUENCIA DEL ACUMULADOR A LOS PINES DEL
PUERTOB
CALL RET ;LLAMA LA RUTINA DE RETARDO ESPECIAL VARIABLE
RETURN
;RUTINA 1 SECUENCIA DE LEDS A LA DERECHA
RUT1 BCF STATUS, C ;BORRAY EL BIT DE CARRY
CLRF SEC ;INICIA REGISTRO DE SECUENCIAS
BSF SEC, B7 ;COMIENZO DE RUTINA
CALL SECU1 ;RUTINA DE LA SECUENCIA UNO
GOTO MENU ;RETORNO AL MENU PRINCIPAL

```

```

;RUTINA 2 SECUENCIA DE LEDS A LA IZQUIERDA
RUT2 BCF STATUS,C ;BORRAY EL BIT DE CARRY
    CLRF SEC ;INICIA REGISTRO DE SECUENCIAS
    BSF SEC, B0 ;COMIENZO DE RUTINA
    CALL SECU2 ;RUTINA DE LA SECUENCIA UNO
    GOTO MENU ;RETORNO AL MENU PRINCIPAL

;RUTINA 3 SECUENCIA DE LEDS CENTRO A FUERA
RUT3 CLRF CONT
    CALL SECU3 ;RUTINA DE LA SECUENCIA UNO
    GOTO MENU ;RETORNO AL MENU PRINCIPAL

;RUTINA 4 SECUENCIA DE LEDS FUERA A CENTRO
RUT4 MOVLW 04H
    MOVWF CONT
    CALL SECU4 ;RUTINA DE LA SECUENCIA UNO
    GOTO MENU ;RETORNO AL MENU PRINCIPAL

;SECUENCIA 1 MOVIMIENTO DE LEDS A LA DERECHA EN SECUENCIA UNO A UNO
SECU1 CALL OUT ;RUTINA DE SALIDA DE DATOS POR PUERTOS DE PINES PTB
    RRF SEC, F ;ROTA UN BIT A LA DERECHA DEL REGISTRO SEC
    BTFSC STATUS,C ;¿EL CARRY ES UNO?
    SECU11 RETURN
        CALL BOTN ;RUTINA DE LLAMADO Y CONTROL DE PULSADORES PUERTO A
        MOVF BOT, W ;EVALUAR EL REGISTRO BOT
        XORLW 10H ;MASCARA PROTEGE EL VALOR DEL 4 BIT DE REGISTRO BOT==>INDICA

TECLA
    BTFSS STATUS,Z ;¿SE OPRIMIO ALGUNA TECLA?
    GOTO SECU1
    BTFSS BOT, B4 ;¿EL BIT DE TECLADO OPRIMIDO ESTA ACTIVO?
    GOTO SECU1
    CLRF BOT ;BORRA EL ESTADO DE BOT==>HAY UNA NUEVA RUTINA
    GOTO SECU11

;SECUENCIA 1 MOVIMIENTO DE LEDS A LA DERECHA EN SECUENCIA UNO A UNO
SECU2 CALL OUT ;RUTINA DE SALIDA DE DATOS POR PUERTOS DE PINES PTB
    RLF SEC, F ;ROTA UN BIT A LA IZQUIERDA DEL REGISTRO SEC
    BTFSC STATUS,C ;¿EL CARRY ES UNO?
    SECU21 RETURN
        CALL BOTN ;RUTINA DE LLAMADO Y CONTROL DE PULSADORES PUERTO A
        MOVF BOT, W ;EVALUAR EL REGISTRO BOT
        XORLW 10H ;MASCARA PROTEGE EL VALOR DEL 4 BIT DE REGISTRO BOT==>INDICA

TECLA
    BTFSS STATUS,Z ;¿SE OPRIMIO ALGUNA TECLA?
    GOTO SECU2
    BTFSS BOT, B4 ;¿EL BIT DE TECLADO OPRIMIDO ESTA ACTIVO?
    GOTO SECU2
    CLRF BOT ;BORRA EL ESTADO DE BOT==>HAY UNA NUEVA RUTINA
    GOTO SECU21

;RUTINA DE TABLA DE SECUENCIAS
TABLA MOVF CONT, W
    ADDWF PCL, F
    RETLW 00H
    RETLW 18H
    RETLW 24H
    RETLW 42H
    RETLW 81H

;RUTINA 3 SECUENCIA DE LEDS CENTRO A FUERA
SECU3 INCF CONT, F ;AUMENTAR VALOR DE TABLA
    CALL TABLA ;RUTINA DE TABLA DE SECUENCIAS
    MOVWF SEC
    CALL OUT ;RUTINA DE SALIDA DE DATOS POR PUERTOS DE PINES PTB
    MOVF CONT, W
    XORLW 04H ;NUMERO DE SECUENCIAS
    BTFSC STATUS,Z ;¿FIN DE SECUENCIA?
    SECU31 RETURN
        CALL BOTN ;RUTINA DE LLAMADO Y CONTROL DE PULSADORES PUERTO A
        MOVF BOT, W ;EVALUAR EL REGISTRO BOT
        XORLW 10H
        BTFSS STATUS,Z ;¿SE OPRIMIO ALGUNA TECLA?
        GOTO SECU3
        BTFSS BOT, B4 ;¿EL BIT DE TECLADO OPRIMIDO ESTA ACTIVO?
        GOTO SECU3

```

```
CLRF BOT ;BORRA EL ESTADO DE BOT==>HAY UNA NUEVA RUTINA
GOTO SECU31
;RUTINA 4 SECUENCIA DE LEDS FUERA A CENTRO
SECU4 CALL TABLA ;RUTINA DE TABLA DE SECUENCIAS
MOVWF SEC
CALL OUT ;RUTINA DE SALIDA DE DATOS POR PUERTOS DE PINES PTB
MOVF CONT, W
XORLW 00H ;NUMERO DE SECUENCIAS
BTFSZ STATUS,Z ;¿FIN DE SECUENCIA?
SECU41 RETURN
DECFSZ CONT, F ;AUMENTAR VALOR DE TABLA
CALL BOTN ;RUTINA DE LLAMADO Y CONTROL DE PULSADORES PUERTO A
MOVF BOT, W ;EVALUAR EL REGISTRO BOT
XORLW 10H
BTFSZ STATUS,Z ;¿SE OPRIMIO ALGUNA TECLA?
GOTO SECU4
BTFSZ BOT, B4 ;¿EL BIT DE TECLADO OPRIMIDO ESTA ACTIVO?
GOTO SECU4
CLRF BOT ;BORRA EL ESTADO DE BOT==>HAY UNA NUEVA RUTINA
GOTO SECU41
;LLAMA LA RUTINA DE RETARDO ESPECIAL VARIABLE
RET MOVF VMAX, W ;VALOR MAXIMO / MINIMO VARIABLE DE TEMPORIZADOR
MOVWF TMP1 ;REGISTRO TEMPORAL 1 TIEMPOS GRANDES
TRES MOVLW VAL1 ;VALOR FIJO
MOVWF TMP2 ;REGISTRO TEMPORAL 2 TIEMPOS MEDIANOS
DOS MOVLW VAL2 ;VALOR FIJO
MOVWF TMP3 ;REGISTRO TEMPORAL 3 TIEMPOS PEQUEÑOS
UNO DECFSZ TMP3, F
GOTO UNO
DECFSZ TMP2, F
GOTO DOS
DECFSZ TMP1, F
GOTO TRES
RETURN
END
```

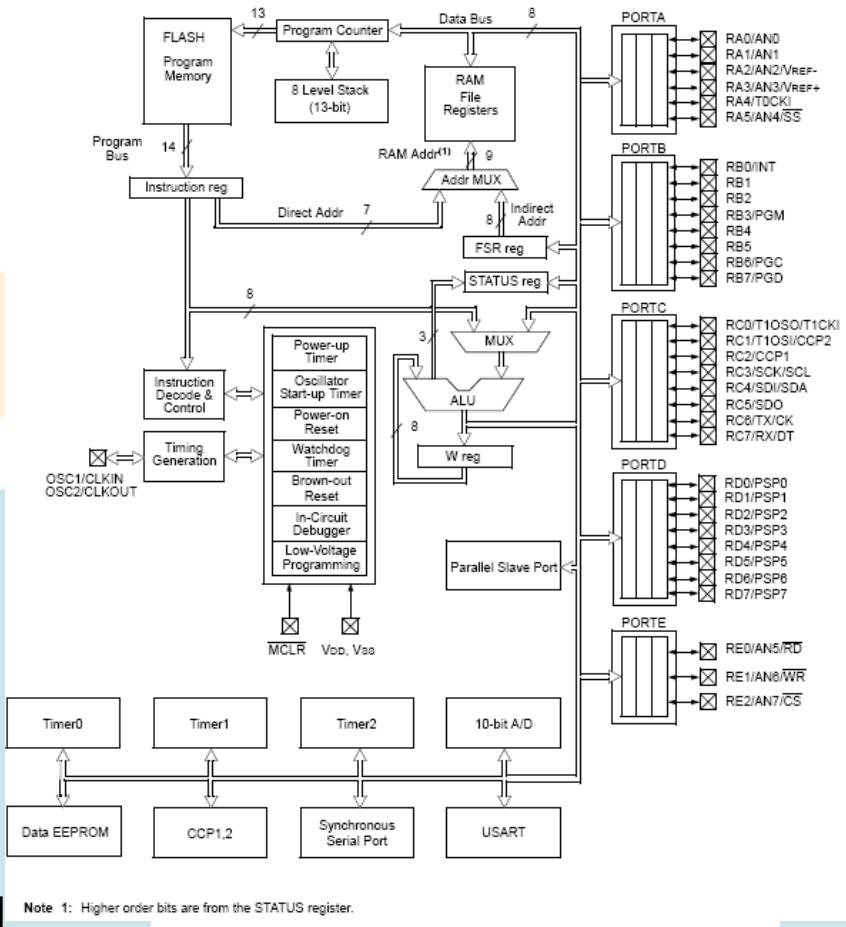
Lección 24: Microcontroladores PIC16F877

El microcontrolador PIC16F877 conserva la simplicidad en programación del PIC16F84, con la característica adicional de implementar más pines I/O distribuidos en 5 puertos de I/O, además con módulos ADC, dos Timers y puertos de comunicación, con lo que el lector puede implementar gran variedad de proyectos de laboratorio para mejorar el conocimiento, habilidades y competencias respecto a la programación e implementación con diversos periféricos.

Arquitectura interna del PIC16F877

El diagrama de bloques de la estructura interna del PIC16F877 se presenta en la siguiente figura.

Figura 111. Arquitectura del PICF877¹²⁴



Organización de la memoria en el PIC16F877

El microcontrolador tiene un contador de programa de 13 bits capaz de direccionar un espacio de memoria de programa de 8K x 14, es decir 8192 palabras de memoria FLASH. El vector de RESET está en la dirección 0000h y el vector de interrupción en la 0004h. Además cuenta con 8 niveles de pila como lo muestra la figura.

Características generales de los microcontroladores PIC16F877

El microcontrolador PIC16F877 tiene como características generales:

- Arquitectura RISC y 35 instrucciones de palabra sencilla
- Todas las instrucciones son de un solo ciclo, excepto los saltos (dos ciclos)

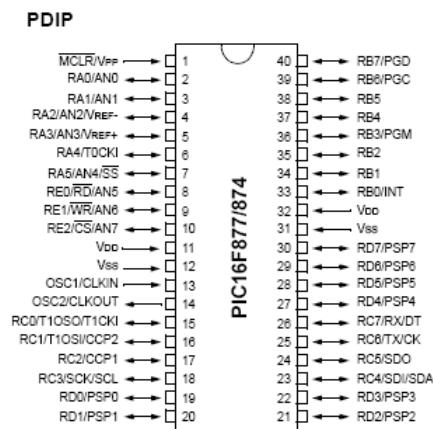
¹²⁴ Extraído el 18 Octubre de 2009 desde <http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf>.

- Velocidades de operación: DC hasta 20MHz con entrada de reloj y DC hasta 200ns ciclo de instrucción.
- Hasta 8Kx14 palabras de memoria programable FLASH, 368x8 bytes de memoria de datos RAM y 256x8 bytes de memoria de datos EEPROM.
- Capacidad de interrupción (hasta 14 fuentes).
- 8 niveles de pila
- Perro guardián (watchdog)
- Modos de direccionamiento: directo, indirecto y relativo
- Conversor analógico digital multicanal de 10 bits.
- TRM0, TRM2 de 8 bits contador/ preescalar.
- TRM1 de 16 bits contador/ prescalar.
- Tiene dos módulos de comparación, captura de 16 bits y PWM de 10 bits.
- Receptor/transmisor USART / SCI.
- Puerto paralelo esclavo de 8 bits con líneas de control externo.

Funciones y Diagrama de pines

La asignación de pines del PIC16F877 se muestra en la siguiente figura.

Figura 112. PIC16F877/874¹²⁵



El PIC16F877 viene en una pastilla integrada de 40 pines. 33 pines conforman los cinco puertos bidireccionales que posee, mientras que los siete pines restantes se emplean para la aplicación del voltaje de alimentación (4), el circuito oscilador (2) y el circuito de Reset.

La descripción de las funciones de los pines del microcontrolador PIC16F877 se presenta a continuación.

¹²⁵ Extraído el 18 Octubre de 2009 desde <http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf>

Tabla 39. Descripción de pines PIC16F877¹²⁶

NOMBRE DEL PIN	TIPO	DESCRIPCION
RA0/AN0 - RA2/AN2	Entrada/salida	Líneas de E/S digitales del Puerto A, o entradas analógicas.
RA3/AN3/VREF	Entrada/salida	E/S digital, analógica o entrada. externa de referencia
RA4/T0CKI	Entrada/salida	E/S digital o entrada de reloj externo. para TMR0.
RA5/AN4/SS	Entrada/salida	E/S digital, analógica o selección del puerto síncrono
RB0/INT - RB7	Entrada/salida	E/S digitales del Puerto B. RB0/INT puede actuar como entrada de interrupción externa. RB4-RB7 pueden provocar una interrupción cuando cambian de estado.
RC0/T1OSO/T1CKI	Entrada/salida	E/S digital del Puerto C. Conexión del oscilador externo para el temporizador TMR1 o entrada de reloj para el TMR1.
RC1/T1OSI/CCP2	Entrada/salida	Conexión del oscilador externo para el TMR1 o salida del módulo 2 de captura/comparación.
RC2/CCP1	Entrada/salida	Salida del módulo 1 de captura/comparación.
RC3/SCK/SCL	Entrada/salida	E/S de reloj para el Puerto Serie Síncrono (SSP) en los módulos SPI o I2C.
RC4/SDI/SDA	Entrada/salida	E/S digital. Entrada de datos serie en el modo SPI. E/S de datos serie en modo I2C.
RC5/SDO	Entrada/salida	E/S digital. Salida de datos serie en modo SPI
RC6/TX/CK	Entrada/salida	E/S digital. Transmisión serie asíncrona. Entrada de reloj para comunicación serie síncrona.
RC7/RX/DT	Entrada/salida	E/S digital. Recepción serie asíncrona. Línea de datos en la comunicación serie síncrona.
RD0/PSP0 - RD7/PSP7	Entrada/salida	E/S digitales del Puerto D. Este puerto puede trabajar como puerto paralelo esclavo para interconexión con un bus de datos de 8 bits de otro microprocesador.
RE0/RD/AN5	Entrada/salida	E/S digital del Puerto E. Señal de lectura del puerto paralelo esclavo. Entrada analógica.
RE1/WR/AN6	Entrada/salida	E/S digital del Puerto E. Señal de escritura del puerto paralelo esclavo. Entrada analógica.
RE2/CS /AN7	Entrada/salida	E/S digital. Señal de activación del puerto paralelo esclavo. Entrada analógica.
OSC1/CLKIN	Entrada	Entrada del cristal / entrada de reloj externo.
OSC2/CLKOUT	Salida	Salida del cristal oscilador. Conecta el cristal o el resonador en el modo cristal oscilador. En el modo RC, la salida del pin de OSC2 CLKOUT. tiene ¼ de la frecuencia de OSC1
V _{ss}	Alimentación	Tierra para los pines lógicos y de E/S
V _{dd}	Alimentación	Fuente de Tensión Positiva
MCLR	Entrada	Entrada maestra de borrado (Reset)/ voltaje de programación. El reset del dispositivo es activo bajo.

¹²⁶ Téllez, 2007

Puertos de entrada / salida (I/O)

El microcontrolador 16F877 cuenta con cinco puertos de E/S (A, B, C, D, E), los cuales suman 33 pines bidireccionales para el trabajo con diversas señales externas. Hay que destacar que algunos pines de E/S están multiplexados con una función alternativa de los periféricos del dispositivo. En general, cuando un periférico es habilitado el pin correspondiente no puede ser utilizado como pin de E/S de propósito general.

Puerto A: este puerto bidireccional tiene un tamaño de 6 bits (RA5:RA0). Tiene además 3 registros asociados que se muestran en la tabla y que corresponden a:

Estado de los pines del puerto A (PORTA)

Control de dirección de los pines del puerto A (TRISA)

El periférico de conversión analógico digital (ADCON1)

Figura 113. Registro Puerto A, TRISA y ADCON1 PIC16F877

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
05h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
9Fh	ADCON1	ADFM	—	—	PCFG3	PCFG2	PCFG1	PCFG0		--0- 0000	--0- 0000

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'.

Shaded cells are not used by PORTA.

Puerto B: este puerto bidireccional tiene un tamaño de 8 bits (RB7:RB0). Tiene además 3 registros asociados que se muestran en la figura y que corresponden a:

Estado de los pines del puerto B (PORTB)

Control de dirección de los pines del puerto B (TRISB)

Interrupción externa pin RB0 (OPTION-REG bit INTEDG)

Figura 114. Puerto B, TRISB y OPTION en PIC16F877

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
06h, 106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h, 186h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111
81h, 181h	OPTION_REG	RBU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

Puerto C: este puerto bidireccional tiene un tamaño de 8 bits (RC7:RC0). Tiene además 2 registros asociados que se muestran en la figura y que corresponden a:

Estado de los pines del puerto C (PORTC)

Control de dirección de los pines del puerto C (TRISC)

Figura 115. Puerto C y TRISC en PIC16F877

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
07h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	aaaa aaaa
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged

Puerto D: este puerto bidireccional tiene un tamaño de 8 bits (RD7:RD0) con entradas bufferadas de *schmitt trigger* (disparo). Puede ser configurado como puerto paralelo esclavo y en este modo los buffers de entrada son TTL. Tiene además 3 registros asociados que se muestran en la figura y que corresponden a:

Estado de los pines del puerto D (PORTD)

Control de dirección de los pines del puerto D (TRISD)

Control de configuración puerto paralelo esclavo (TRISE)

Figura 116. Puerto D y TRISD en PIC16F877

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
08h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	aaaa aaaa
88h	TRISD	PORTD Data Direction Register								1111 1111	1111 1111
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits	—	—	0000 -111	0000 -111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTD.

Puerto E: este puerto bidireccional tiene un tamaño de 3 bits (RE2:RE0), con entradas bufferadas de *schmitt trigger* (disparo). Puede ser configurado como puerto paralelo esclavo, en este modo los buffers de entrada son TTL. Tiene además 3 registros asociados que se muestran en la figura y que corresponden a:

Operación de lectura / escritura de los pines RE cuando son entradas analógicas o puerto paralelo esclavo (PORTE)

Control de configuración puerto paralelo esclavo y control de la dirección de los pines RE (TRISE)

Configuración de entrada/salida digital (ADCON1)

Figura 117. Puerto E, TIRSE y ADCON1 en PIC16F877

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
09h	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuuu
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits	—	—	0000 -111	0000 -111
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000

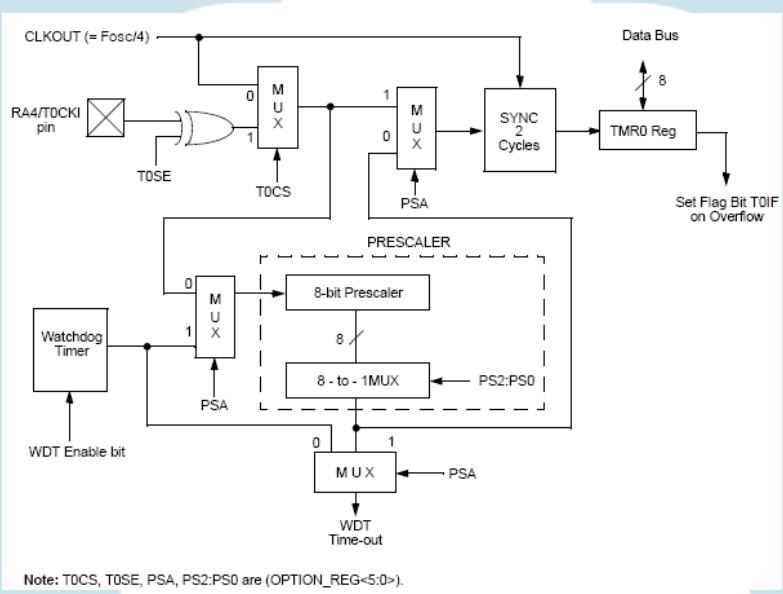
Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTE.

Módulos en el PIC16F877

Como se ha planteado en anteriores lecciones los módulos son unidades funcionales integradas en el microcontrolador, que lo hacen robusto y flexible para dar solución a los proyectos o problemas de implementación, automatización y control digital.

Módulo Timer0: Este temporizador/contador de 8 bits, tiene las características de ser de escritura y lectura, preescalador de 8 bits programable por software, selección de reloj interno o externo e interrupción por desbordamiento. La siguiente figura muestra el diagrama de bloques y la tabla con los registros relacionados con el Timer0.

Figura 118. Timer0 PIC16F877¹²⁷



Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
01h,101h	TMR0	Timer0 Module's Register								xxxx xxxx	uuuu uuuu
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	0000 000u
81h,181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

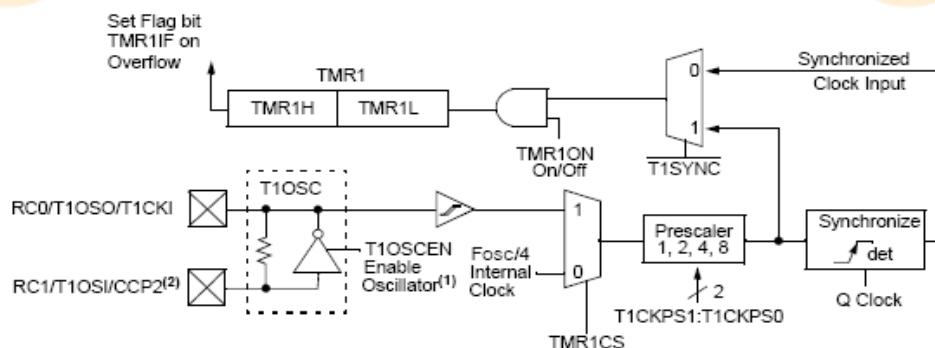
Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'.

Shaded cells are not used by Timer0.

¹²⁷ Extraído el 18 Octubre de 2009 desde <http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf>

Módulo Timer1: El timer1 es un módulo contador/temporizador de 16 bits que consta de dos registros de 8 bits, los cuales pueden ser escritos y leídos continuamente. El módulo puede operar como temporizador o como contador. En el modo temporizador el TIMER1 se incrementa cada ciclo de instrucción, en el modo contador se incrementa con el flanco de bajada de la entrada de reloj externa. El diagrama de bloques del módulo se muestra en la siguiente figura y la tabla muestra los registros relacionados con el Timer0.

Figura 119. Timer1 PIC16F877A¹²⁸



Note 1: When the T1OSCEN bit is cleared, the inverter is turned off. This eliminates power drain.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
0Bh, 8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF ⁽¹⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE ⁽¹⁾	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	aaaa aaaa
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	aaaa aaaa
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	--uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.

Note 1: Bits PSPIE and PSPIF are reserved on the PIC16F873/876; always maintain these bits clear.

Módulo de Puerto Serial Maestro Síncrono (MSSP): Es una interfaz serial utilizada para comunicarse con otros periféricos o microcontroladores. Estos dispositivos periféricos pueden ser memorias EEPROM seriales, registros de desplazamiento, manejadores de display, conversores A/D entre otros. El módulo MSSP puede operar en dos modos: interfaz serial periférica (SPI) ó circuito inter-integrado (I²C)

¹²⁸ Extraído el 18 Octubre de 2009 desde <http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf>

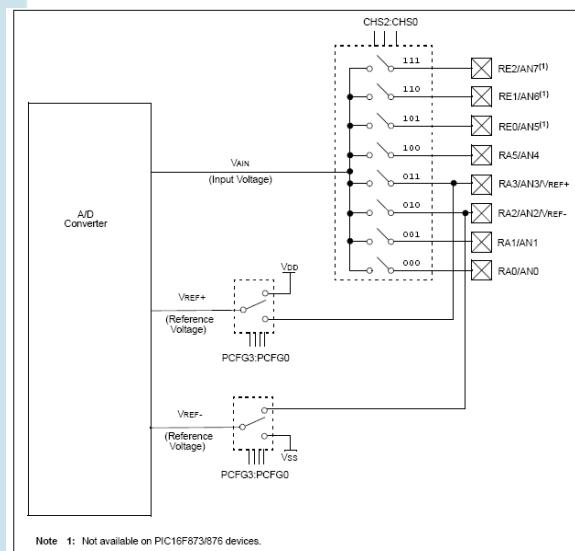
a) Interfaz Serial Periférica (SPI): El SPI permite transmitir y recibir datos de 8 bits simultáneamente. Además soporta las cuatro modalidades que son:

- Salida serial de dato (SDO)
- Entrada serial de dato (SDI)
- Reloj serial (SCK)
- Selección esclavo (\overline{SS})

b) Circuito inter- integrado (I²C): El módulo soporta todas las configuraciones maestro y esclavo, provee interrupciones por hardware de los bits de arranque y parada y determina cuando libera el bus.

Conversor analógico digital: El conversor analógico/digital del PIC16F877 tiene 8 entradas multiplexadas y una resolución máxima de 10 bits.

Figura 120. Conversión Análoga - Digital¹²⁹



La señal analógica de entrada es muestreada y cargada en un capacitor a la entrada del conversor; el cual genera como resultado un valor digital del nivel analógico a través de aproximaciones sucesivas. La conversión analógico/digital de la señal analógica de entrada tiene una resolución máxima de 10 bits. El módulo cuenta también con un nivel alto y bajo de voltaje de referencia programado por software, escogiendo entre los pines Vss, Vdd, RA3 y RA2.

¹²⁹ Extraído el 18 Octubre de 2009 desde <http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf>

El conversor A/D tiene la característica de operar mientras el microcontrolador se encuentra en modo SLEEP. El reloj del A/D es derivado del oscilador interno del conversor.

El tiempo mínimo de adquisición es de $19,72 \mu s$. El tiempo de conversión del A/D por bit se denomina T_{AD} ; en el caso del módulo se necesitan 12 T_{AD} para 10 bits. La fuente del reloj de conversión se selecciona por software y pueden ser: 2Tosc, 8 Tosc, 32 Tosc ó módulo oscilador interno del ADC. La siguiente tabla muestra los registros asociados con el A/D.

Tabla 40. Registros asociados con el conversor A/D del PIC16F877

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on MCLR, WDT
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF ⁽¹⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE ⁽¹⁾	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
1Eh	ADRESH	A/D Result Register High Byte							xxxx xxxx	uuuu uuuu	
9Eh	ADRESL	A/D Result Register Low Byte							xxxx xxxx	uuuu uuuu	
1Fh	ADC0N0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON	0000 00-0	0000 00-0
9Fh	ADC0N1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	--0u 0000
89h ⁽¹⁾	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	0000 -111
09h ⁽¹⁾	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used for A/D conversion.

Note 1: These registers/bits are not available on the 28-pin devices.

Lección 25: Ejemplos de programación con PIC16F877

La programación del PIC16F877 es muy similar en estructura al PIC16F84, la diferencia radica en la configuración para seleccionar la función adecuada de pines, aparte de la gran cantidad de pines I/O, con cinco (5) puertos y con la incorporación de módulos especiales. La siguiente es una descripción detallada de apartes de programación y código fuente para este microcontrolador.

Estructura de un programa para PIC16F877

La estructura de un programa para PIC16F877 es similar a la de un PIC16F84, sin embargo por la cantidad de registros de propósito especial se puede tornar un

poco complicado entender la estructura, pero esta se puede utilizar como plantilla para otros proyectos de manera que solo se tenga que digitar una vez.

Definición de encabezado

El encabezado está configurado para MpLab y se puede utilizar directamente en Proteus, se compone de un llamado a la librería particular del microcontrolador, es conveniente incorporar comentarios para la identificación del programa, función y su autor. A continuación se presenta un ejemplo de encabezado.

```
TITLE "PIC16F877 NOMBRE DEL PROGRAMA"
LIST P=16F877A,F=INHX32
#INCLUDE <P16F877A.INC>

;INGENIERO ELECTRONICO HECTOR URIEL VILLAMIL GONZALEZ
;PROCESO:
;ENTRADAS:
;SALIDAS:
```

Definición de registros, constantes y macros

Lo siguiente es definir los registros de propósito especial FSR ubicados en sus diferentes bancos de memoria, registros de propósito general RPG, definición de bits para diferentes registros especiales, definición de constantes y definición de macros. Se presenta un ejemplo de definición de registros, constantes y macros, para que el lector tenga un punto de referencia para realizar sus programas.

```
,*****DEFINICION DE REGISTROS DE PROPOSITO ESPECIAL FSR
,*****BANCO CERO DE MEMORIA RAM - RPG=96 BYTES

TMR0 EQU 01H ;REGISTRO TEMPORIZADOR CERO
PCL EQU 02H ;PROGRAM COUNT PARTE BAJA
STATUS EQU 03H ;REGISTRO DE ESTADO DEL PIC
PTA EQU 05H ;PUERTO A DEL PIC 6 LINEAS BIDIRECCIONALES RA4/TOCK, LAS DEMAS
DIGITALES/ANALOGAS
FSR EQU 04H ;REGISTRO PARA DIRECCIONAMIENTO INDIRECTO
PTB EQU 06H ;PUERTO B DEL PIC 8 LINEAS BIDIRECCIONALES DIGITALES
PTC EQU 07H ;PUERTO C DEL PIC 8 LINEAS BIDIRECCIONALES DIGITALES
PTD EQU 08H ;PUERTO D DEL PIC 8 LINEAS BIDIRECCIONALES DIGITALES
PTE EQU 09H ;PUERTO E DEL PIC 3 LINEAS BIDIRECCIONALES DIGITALES / ANALOGAS
PCLATH EQU 0AH ;ACTUALIZACION DEL REGISTRO PCH PARA CAMBIO DE BANCO 2K
INTCON EQU 0BH ;REGISTRO DE INTERRUPCIONES
PIR1 EQU 0CH ;FLAGS INDIVIDUALES DE INTERRUPCIONES PERIFERICAS
PIR2 EQU 0DH ;FLAGS INDIVIDUALES DE INTERRUPCIONES CCP2, SSP Y EEPROM
TMR1L EQU 0EH ;PARTE BAJA DEL TIMER 1
TMR1H EQU 0FH ;PARTE ALTA DEL TIMER 1
T1CON EQU 10H ;CONFIGURACION DEL TIMER 1
TMR2 EQU 11H ;REGISTRO TIMER 2
T2CON EQU 12H ;CONFIGURACION DEL TIMER 2
CCPR1L EQU 15H ;MODULO CCP1 PARTE BAJA
CCPR1H EQU 16H ;MODULO CCP1 PARTE ALTA
CCP1CON EQU 17H ;CONFIGURACION DEL MODULO CCP1
RCSTA EQU 18H ;CONFIGURACION RECEPCION ASINCRONO REGISTRO
TXREG EQU 19H ;TRANSMISOR ASINCRONO REGISTRO DE CORRIENTO
```

```

RCREG EQU 1AH ;RESULTADO DE LA RECEPCION ASINCRONA
CCPR2PL EQU 15H ;MODULO CCP2 PARTE BAJA
CCPR2PH EQU 16H ;MODULO CCP2 PARTE ALTA
CCP2CN EQU 17H ;CONFIGURACION DEL MODULO CCP2
ADRESH EQU 1EH ;RESULTADO DEL ADC PARTE ALTA
ADCON0 EQU 1FH ;SELECCION DEL CANAL DE ENTRADA A CONVERTIR

;*****BANCO DOS DE MEMORIA RAM - RPG=80 BYTES

OPCION EQU 81H ;OPCIONES DE CONFIGURACION INT TMP PULLUP PREESCALER
PCLH EQU 82H ;CONTADOR DE PROGRAMA PARTE BAJA
TRISA EQU 85H ;/RA7-RA6-RA5=NOP/RA4=/RA3=/RA2=/RA1=/RA0=/
TRISB EQU 86H ;/RB7=/RB6=/RB5=/RB4=/RB3=/RB2/RB1=/RB0=/
TRISC EQU 87H ;/RC7=/RC6=/RC5=/RC4=/RC3=/RC2/RC1=/RC0=/
TRISD EQU 88H ;/RD7=/RD6=/RD5=/RD4=/RD3=/RD2/RD1=/RD0=/
TRISE EQU 89H ;/RE2=/RE1=/RE0=/
PIE1 EQU 8CH ;REGISTRO HABILITADOR DE INTERRUPCIONES PERIFERICAS
PIE1A EQU 8DH ;REGISTRO HABILITADOR DE INTERRUPCIONES CCP2, SSP Y EEPROM
PCON EQU 8EH ;THE POWER CONTROL - PCON --> POR, BOR
PR2 EQU 92H ;REGISTRO DE PERIODO PARA COMPARACION
TXSTA EQU 98H ;CONFIGURACION USART
SPBRG EQU 99H ;GENERADOR DE BAUD RATE (BRG)
ADRESL EQU 9EH ;RESULTADO DEL ADC PARTE BAJA
ADCON1 EQU 9FH ;REGISTRO DE CONFIGURACION DE ENTRADAS ANALOGAS VREF+ Y VREF-

;*****BANCO TRES DE MEMORIA RAM - RPG= 16 + 80 BYTES

EEDATA EQU 10CH ;DATO IN/OUT EN EEPROM
EEADR EQU 10DH ;DIRECCION DE IN/OUT DE EEPROM
EEDATH EQU 10EH ;
EEADRH EQU 10FH ;

;*****BANCO CUATRO DE MEMORIA RAM - RPG= 16 + 80 BYTES

EECON1 EQU 18CH ;B4=EEIF/FIN WR/*B3=WRERR/ERROR WR/*B2=WREN/PERMISO WR/*B1/WR/*B0/RD/
EECON2 EQU 19CH ;REG SEGURIDAD PROCESO EEPROM

;*****;DEFINICION REGISTROS RPG

SAVSTA EQU 20H ;REGISTRO SALVA CONTENIDO DE STATUS
SWREG EQU 21H ;REGISTRO SALVA CONTENIDO DE W
RPGT1 EQU 22H ;REGISTRO PROPÓSITO GENERAL TEMPORAL 1: inicio lcd
RPGT2 EQU 23H ;REGISTRO PROPÓSITO GENERAL TEMPORAL 2: inicio lcd

;*****;DEFINICION DE BITS

W EQU 0 ;REGISTRO DE TRABAJO
F EQU 1 ;REGISTRO
;*****STATUS
C EQU 0 ;FLAG DE CARRY
Z EQU 2 ;FLAG DE CERO
RP0 EQU 5 ;SELECTOR DE PAGINA BIT 0
RP1 EQU 6 ;SELECTOR DE PAGINA BIT 1
;*****OPTION
PSO EQU 0 ;PREESCALER BIT 0
PS1 EQU 1 ;PREESCALER BIT 1
PS2 EQU 2 ;PREESCALER BIT 2
PSA EQU 3 ;ASIGNACION DEL PREESCALADOR 0=TMR0 1=WDT
T0SE EQU 4 ;FLANCO DE LA SEÑAL DE INCREMENTO DEL TMR0 0=L-H 1=H-L
T0CS EQU 5 ;FUENTE DE LA SEÑAL DEL TMR0 0=TMP 1=CONTADOR
INTEDG EQU 6 ;FLANCO DE LA SEÑAL DE INTERRUPCION INT 0=LOW 1=UP
RBPU EQU 7 ;PULL-UPS INTERNAS 0=E 1=D
;*****EECON1
RD EQU 0 ;RD=1 CICLO DE LECTURA DE LA EEPROM
WR EQU 1 ;WR=1 CICLO DE ESCRITURA DE LA EEPROM
WREN EQU 2 ;WREN=1 AUTORIZA PERMISO DE ESCRITURA EN LA EEPROM

```

```

EEP GD EQU 7 ;EEP GD PROGRAM/DATA EEPROM SELECT BIT 1=ACCESSES PROGRAM MEMORY
0=ACCESSES DATA MEMORY
;*****INTCON
RBIF EQU 0 ;BANDERA DE INTERRUPCIÓN POR CAMBIO EN RB <7:4>
INTF EQU 1 ;BANDERA DE INTERRUPCIÓN INT
T0IF EQU 2 ;BANDERA DE INTERRUPCIÓN TMR0
RBIE EQU 3 ;HABILITACIÓN INTERRUPCIÓN RB <7:4>
INTE EQU 4 ;HABILITACIÓN INTERRUPCIÓN INT
T0IE EQU 5 ;HABILITACIÓN INTERRUPCIÓN TMR0
PEIE EQU 6 ;HABILITACIÓN INTERRUPCIÓN POR MODULOS PERIFERICOS
GIE EQU 7 ;HABILITACIÓN GLOBAL DE INTERRUPCIONES

;*****ADCCON0
ADON EQU 0 ;ENCENDIDO DEL CONVERTIDOR ADC
GODONE EQU 2 ;BIT DE INICIO Y FIN DE CONVERSIÓN
CHS0 EQU 3 ;CANAL ANALÓGICO 0 A CONVERTIR
CHS1 EQU 4 ;CANAL ANALÓGICO 1 A CONVERTIR
CHS2 EQU 5 ;CANAL ANALÓGICO 2 A CONVERTIR
ADCS0 EQU 6 ;SELECCIÓN DE LA FRECUENCIA DEL CONVERTIDOR ADC
ADCS1 EQU 7 ;SELECCIÓN DE LA FRECUENCIA DEL CONVERTIDOR ADC

;*****ADCCON1
PCFG0 EQU 0 ;BITS DE CONFIGURACIÓN DE LAS ENTRADAS DEL CONVERTIDOR
PCFG1 EQU 1 ;BITS DE CONFIGURACIÓN DE LAS ENTRADAS DEL CONVERTIDOR
PCFG2 EQU 2 ;BITS DE CONFIGURACIÓN DE LAS ENTRADAS DEL CONVERTIDOR
PCFG3 EQU 3 ;BITS DE CONFIGURACIÓN DE LAS ENTRADAS DEL CONVERTIDOR
ADFM EQU 7 ;SELECCIÓN DEL FORMATO DE ENTRADA, 1=10 BITS JUSTIFICADO A LA DERECHA

;*****T1CON
TMR1ON EQU 0 ;HABILITACIÓN/DESHABILITACIÓN DE TIMER 1
TMR1CS EQU 1 ;FUENTE DEL RELOJ, 1=MODO CONTADOR 0=MODO TEMPORIZADOR
T1SYNC EQU 2 ;
T1OSCEN EQU 3 ;
T1CKPS0 EQU 4 ;SELECCIÓN DEL PREESCALADOR
T1CKPS1 EQU 5 ;SELECCIÓN DEL PREESCALADOR

;*****T2CON
T2CKPS0 EQU 0 ;CONFIGURACIÓN DEL PREESCALADOR
T2CKPS1 EQU 1 ;CONFIGURACIÓN DEL PREESCALADOR
TMR2ON EQU 2 ;BIT DE ENCENDIDO DEL TIMER 2
T2OUPS0 EQU 3 ;CONFIGURACIÓN DEL POST-ESCALADOR
T2OUPS1 EQU 4 ;CONFIGURACIÓN DEL POST-ESCALADOR
T2OUPS2 EQU 5 ;CONFIGURACIÓN DEL POST-ESCALADOR
T2OUPS3 EQU 6 ;CONFIGURACIÓN DEL POST-ESCALADOR

;*****
B0 EQU 0 ;BIT 0
B1 EQU 1 ;BIT 1
B2 EQU 2 ;BIT 2
B3 EQU 3 ;BIT 3
B4 EQU 4 ;BIT 4
B5 EQU 5 ;BIT 5
B6 EQU 6 ;BIT 6
B7 EQU 7 ;BIT 7

;*****CONSTANTES PARA EL CONTROL DEL LCD
LCD_EN EQU 2 ;ENABLE EN PTE --> TRISE RE2=LCD_OUT_E
LCD_RW EQU 1 ;R / W EN PTE --> TRISE RE1=LCD_OUT_RW
LCD_RS EQU 0 ;RS EN PTE --> TRISD RE0=LCD_OUT_RS

R20ms1 EQU 15H ;CONSTANTE DE TEMPORIZADOR PARA 20ms INICIO DEL LCD
R20ms2 EQU 0EDH ;CONSTANTE DE TEMPORIZADOR PARA 20ms INICIO DEL LCD
R5ms1 EQU 6H ;CONSTANTE DE TEMPORIZADOR PARA 5ms INICIO DEL LCD
R5ms2 EQU 10EH ;CONSTANTE DE TEMPORIZADOR PARA 5ms INICIO DEL LCD
R200us1 EQU 30H ;CONSTANTE DE TEMPORIZADOR PARA 200us INICIO DEL LCD

;*****DEFINICIÓN DE DIRECCIONES EEPROM
STAEE EQU 00H ;B7=NOP/B6=NOP/B5=NOP/B4=OFF*LCD/B3=LIGHT*LCD/B2=ALAR*ON:OFF/B1-
B2=CLAVOPC

;*****DEFINICIÓN DE MACROS
#define BANK2 BSF STATUS, 6 ;RP1=1 RP0=X - BANCO 2 O 3

```

```
#DEFINE BANK02 BCF      STATUS, 6          ;RP1=0 RP0=X - BANCO 0 O 2
#DEFINE BANK1  BSF      STATUS, 5          ;RP1=X RP0=1 - BANCO 1 O 3
#DEFINE BANK01 BCF      STATUS, 5          ;RP1=X RP0=0 - BANCO 0 O 2
```

Programa principal

Como característica especial al tener varios bancos de memoria de programa y varios bancos para la memoria de datos, es necesario direccionar adecuadamente los bancos de memoria de datos y/o programa, también se debe tener en cuenta el vector de Reset y el vector de interrupción para poder configurar adecuadamente el programa principal, las subrutinas de atención a interrupciones y las subrutinas referidas por llamados o saltos de programa.

```
;INICIO DEL PROGRAMA
ORG 00H           ;VECTOR DE RESET
GOTO INICIO       ;PROGRAMA PRINCIPAL
NOP

;***** RUTINA DE INTERRUPCION ---> 004H*****
ORG 04H           ;VECTOR DE INTERRUPCION
CALL SAVE1         ;RUTINA SALVA STATUS / W_REG
;***** BANCO 1 DE MEMORIA DE PROGRAMA
BCF PCLATH, B4    ;DIRECCIONANDO BANCO UNO DE PROGRAMA
BSF PCLATH, B3
GOTO MEN_INT       ;RUTINA DE MENU SOBRE INTERUPCIONES
END_INT NOP
CALL SAVE2         ;REGRESA STATUS / W_REG A ESTADO ORIGINAL
RETFIE            ;SALIDA DE RUTINA DE INTERRUPCION

;***** RUTINA PAGINA 0 DE MEMORIA ---> 010H*****
ORG 10H
NOP

;***** PROGRAMA DE INICIO*****
INICIO :***** BANCO 3 DE MEMORIA DE PROGRAMA
BSF PCLATH, B4    ;DIRECCIONANDO BANCO TRES DE PROGRAMA
BSF PCLATH, B3
CALL CONFIGU        ;CONFIGURACION DE PUERTOS
CALL INILCD        ;CONFIGURACION DE BOORLOADER (INICIO) LCD
CALL CONFLCD       ;CONFIGURACION (A) DEL LC
CALL MSNJ03        ;CONTROL ENVIO DE MENSAJE INICIAL "INC.V.2013.Ingl / PLEASE Whait..."
CALL CONADC        ;CONFIGURACION DE ADC
CALL CONFINT       ;CONFIGURACION DE INTERRUPCIONES
CALL CLRRPG        ;BORRADO DE REGISTROS DE PROPOSITO GENERAL
CALLINI_REG        ;INICIALIZACION DE REGISTROS RAM
CALL MSNJ02        ;MENSAJE: "Begining... / ADC-INT-RPG-INIR"
MOVLW 25H
MOVWF RTalnt       ;(REGISTRO TEMPERATURA ACTUAL PARTE ENTERA)
MOVWF RTaDec
CALL MSNJ03

;***** BANCO 2 DE MEMORIA DE PROGRAMA
BSF PCLATH, B4    ;DIRECCIONANDO BANCO DOS DE PROGRAMA
BCF PCLATH, B3
CALL TEST_IN        ;TEST O PRUEBA DEL SISTEMA DE INCUBADORA --> BANCO 2 DE MEMORIA
DE PROGRAMA
;***** BANCO 0 DE MEMORIA DE PROGRAMA
BCF PCLATH, B4    ;DIRECCIONANDO BANCO CERO DE PROGRAMA
BCF PCLATH, B3
GOTO INPRGP        ;RUTINA DE PROGRAMA PRINCIPAL
```

Acceso a banco de memoria

Para acceder a un banco de memoria de programa específico, se debe especificar en el código ensamblador las instrucciones para acceder al bando antes de hacer el llamado o salto, como se especifica en el código anterior. Para determinar el punto donde comienza un banco de memoria se debe remitir a las hojas técnicas o datasheet del micro e incorporar las instrucciones correctas, a continuación se relaciona un ejemplo del código.

```
*****  
;*****RUTINA PAGINA 2 DE MEMORIA --> 1000H - 17FFH  
*****  
;  
ORG 1000H ; BANCO 2 DE MEMORIA DE PROGRAMA  
TEST_IN NOP ; INSTRUCCIÓN NO OPERA – NO HACE NADA  
RETURN ; RETORNO DE LLAMADO A SUBRUTINA TEST_IN
```

Datos en EEPROM

Muchas veces es necesario incorporar datos constantes en la memoria de datos EEPROM, para utilizarlos como valores iniciales o parámetros de control, este ejercicio se realiza de manera simple direccionando la memoria de datos EEPROM, dirección establecida en la hoja técnica de datos del micro (datasheet) y utilizando en cada línea la instrucción “DE” seguida después de un “TAB” o espacio en la segunda columna, de valor en caracteres ASCII que debe guardarse en memoria. Cada carácter ASCII ocupa un Byte, la cadena de caracteres ASCII debe colocarse entre comillas. A continuación se muestra un ejemplo del código.

```
*****  
;*****VALOR EN EEPROM *****  
*****  
;  
ORG 2100H  
  
*****MENSAJES BOOTLOADER  
*****TOTAL ESPACIOS DE EEPROM = XX (XX-XX)H  
DE "INC.V.2013.Ingl" ;*****DIR: 00H - 0FH  
DE "...tiahW ESAELP" ;*****DIR: 10H - 1FH  
DE "Begining..." ;*****DIR: 20H - 2AH  
DE "ADC-INT-RPG-INIR" ;*****DIR: 2BH - 3AH
```

Fin del programa

Al finalizar el código ensamblador se debe incorporar la instrucción que determina el final del programa.

```
*****  
;*****FIN DEL PROGRAMA*****  
*****  
;  
END
```

CAPITULO 6: MICROCONTROLADORES MOTOROLA FREESCALE , BASIC STAMP Y ARDUINO

Introducción

Con las bases teóricas, conceptuales y prácticas de los microprocesadores y microcontroladores y particularmente de los microcontroladores PIC, se tiene lo necesario para continuar con el estudio de otros microcontroladores ampliamente utilizados, como los Motorola Freescale, Texas Instruments, Basic Stamp y Arduino.

Lección 26: Microcontroladores Motorola Freescale

Los microcontroladores Motorola Freescale, son utilizados en aplicaciones en la industria automotriz y a nivel industrial en general. Dadas las características de arquitectura y herramientas en el mercado es un dispositivo útil para el aprendizaje y desarrollo de proyectos.

Microcontroladores Motorola Freescale

La elección de un MICROCONTROLADOR FREESCALE frente a otros más conocidos como el 80XX de Intel, el PIC de Microchip, el ST-62XX de SGS-Thomson o el Z86XX de Zilog, se debe a características como su bajo precio, velocidad, reducido consumo de energía, tamaño, facilidad de uso, fácil programación y lo mejor de todo son los recursos que la gran mayoría de estos microcontroladores presentan a la hora de diseñar cualquier aplicación. Es por ello que los microcontroladores Freescale se encuentran hoy en día en la gran mayoría de aplicaciones industriales, de comunicaciones y control. Si se desea investigar al respecto, por ejemplo, en el caso de la industria automotriz, la cual en la actualidad es una de las que requiere mayor precisión en el desarrollo de procesos de control, instrumentación, entre otras. Casi el 90% de sus componentes son gobernados por microcontroladores Freescale, debido a sus bondades, estabilidad, inmunidad al ruido y otros factores importantes que hacen decisiva su elección frente a otras marcas. (VESGA, 2007).

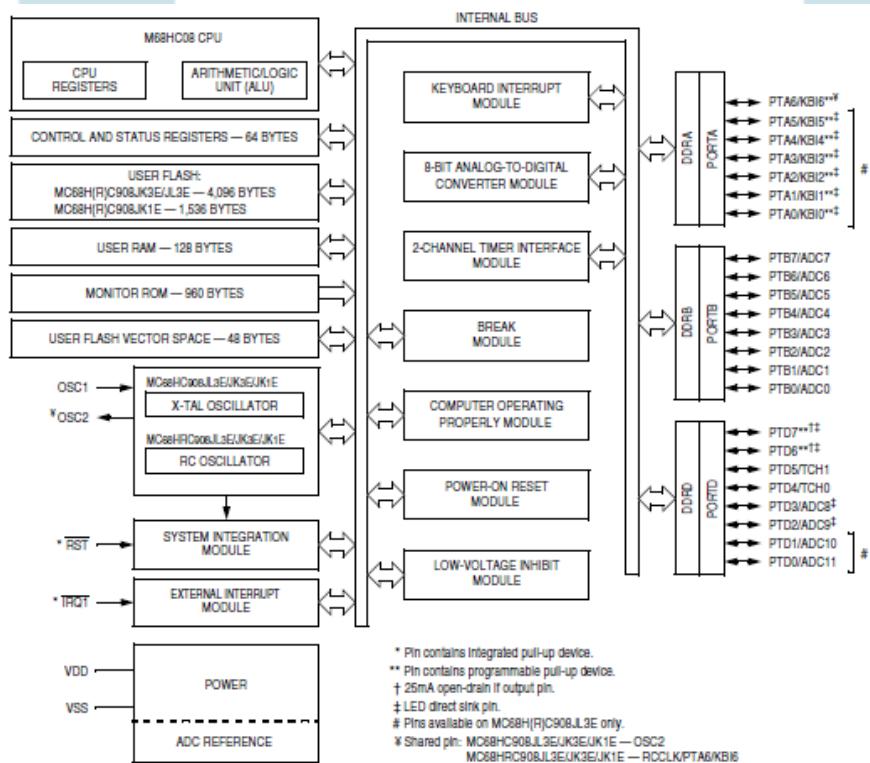
Tabla 41. Variación de la familia MC68H(R)C908/JL3/JK3/JK1¹³⁰

Device	Oscillator Option	FLASH Memory Size	Pin Count
MC68HC908JL3E	X-TAL	4096 bytes	28
MC68HRC908JL3E	RC		
MC68HC908JK3E	X-TAL	4096 bytes	20
MC68HRC908JK3E	RC		
MC68HC908JK1E	X-TAL	1536 bytes	20
MC68HRC908JK1E	RC		

Arquitectura

La arquitectura que implementa es del tipo CISC en sus instrucciones y Von Neumann en su disposición de buses entre la CPU y memoria de datos y programa. Dispone de un mapa lineal de memoria, en el cual se dispone los registros de I/O, registros reservados, memoria RAM, memoria Flash con 4096 posiciones de memoria para el MC68H(R)C908JL3/JK3 y otros registros de propósito especial como registros de estado, de control, de ruptura, etc.

Figura 121. Arquitectura del MC68H(R)C908JL3E



Funcionamiento

El microcontrolador funciona en términos generales de forma muy similar al PIC, sin embargo su diferencia radica en su arquitectura, puesto que al tener un set de instrucciones CISC, las instrucciones utilizan uno o más ciclos máquina para su ejecución, se presenta un set de instrucciones bastante completo pero a la vez grande, lo que hace que el programador deba tener a la mano el set de instrucciones y requiera un poco más de práctica para manipularlo adecuadamente.

Figura 122. Funciones de pines Motorola Freescale MC68H(R)C908

PIN NAME	PIN DESCRIPTION	IN/OUT	VOLTAGE LEVEL
VDD	Power supply.	In	5V or 3V
VSS	Power supply ground	Out	0V
RST	RESET input, active low. With Internal pull-up and schmitt trigger input.	Input	VDD
IRQ1	External IRQ pin. With software programmable internal pull-up and schmitt trigger input. This pin is also used for mode entry selection.	Input	VDD to VDD+V _H
OSC1	X-tal or RC oscillator input.	In	Analog
OSC2	MC68HC908JL3E/JK3E/JK1E: X-tal oscillator output, this is the inverting OSC1 signal.	Out	Analog
	MC68HRC908JL3E/JK3E/JK1E: Default is RC oscillator clock output, RCCLK. Shared with PTA6/KB16, with programmable pull-up.	In/Out	VDD
PTA[0:6]	7-bit general purpose I/O port. Shared with 7 keyboard interrupts KB1[0:6].	In/Out	VDD
	Each pin has programmable internal pull-up device.	In	VDD
	PTA[0:5] have LED direct sink capability	In	VSS
	8-bit general purpose I/O port. Shared with 8 ADC inputs, ADC[0:7].	In/Out	VDD
PTB[0:7]	8-bit general purpose I/O port. PTD[3:0] shared with 4 ADC inputs, ADC[8:11].	In	Analog
	PTD[4:5] shared with TIM channels, TCH0 and TCH1.	In/Out	VDD
	PTD[2:3], PTD[6:7] have LED direct sink capability	In	VSS
	PTD[6:7] can be configured as 25mA open-drain output with pull-up.	In/Out	VDD

Características Generales de los microcontroladores

Características de los Microcontroladores MC68H (R) C908JL3E:

- EMC versión mejorada de MC68H (R) C908JL3/JK3/JK1
- Arquitectura de alto rendimiento M68HC08
- Totalmente compatible con el código objeto de las familias M6805, M146805, y M68HC05
- Diseño de baja potencia; totalmente estática con los modos de parada y espera

- Máxima frecuencia de bus interno:
 - 8 MHz en la tensión de 5V
 - 4-MHz 3V en la tensión de funcionamiento
- Oscilador opciones:
 - Oscilador de cristal de MC68HC908JL3E/JK3E/JK1E
 - RC oscilador para MC68HRC908JL3E/JK3E/JK1E
- Programa de usuario en memoria FLASH
 - 4096 bytes para MC68H (R) C908JL3E/JK3E
 - 1536 bytes para MC68H (R) C908JK1E
- 128 bytes de memoria RAM on-chip
- 2 canales, temporizador de 16 bits Módulo de interfaz (TIM)
- 12 canales, 8 bits de analógico a digital (ADC)
- 23 de propósito general puertos I / O para MC68H (R) C908JL3E:
 - 7 interrupciones con teclado desplegable interior hasta (6 interrupciones para MC68HC908JL3E)
 - 10 drivers para LED
 - 2 × 25 mA open-drain E / S con pull-up
- 15 puertos I / O de propósito general para MC68H (R) C908JK3E/JK1E:
 - 1 interrupción por teclado con pull-up interno (MC68HRC908JK3E/JK1E solamente)
 - 4 drivers LED
 - 2 × 25 mA open-drain E / S con pull-up
 - ADC de 10 canales
- Características del Sistema de protección:
 - Optional computer operating properly (COP) reset
 - Opcional, de detección de bajo voltaje seleccionable con reset y puntos seleccionables de operación entre 3V y 5V
- Pin maestro de reset con pull-up internos y power-on reset
- IRQ1 con entrada Schmitt-trigger y pull-up programables
- Empaquetado de 28-pines PDIP, 28-pines SOIC, y de 48-pines LQFP para MC68H (R) C908JL3E
- Empaquetado de 20-pin PDIP y 20-pin SOIC para MC68H (R) C908JK3E/JK1E
- Características principales de la CPU08 :
 - Modelo de programación ampliado de HC05
 - Amplias funciones de control de bucle
 - 16 modos de direccionamiento (ocho más que la HC05)
 - Registro índice de 16-bit y apuntador de pila
 - Transferencia de datos memoria – memoria.
 - Instrucciones de binario – código decimal (BCD)
 - Soporte para lenguaje C

Oscilador, circuito de reset

La función del OSC2 se configura cuando se escoge la opción de oscilador RC.

1 : El OSC2 es configurado para utilizar el pin PTA6 como un pin de I/O, con las funciones de interrupción y configuración de resistencias de Pull-Up.

0 : El OSC2 es configurado como oscilador de tipo RC

Registros

El microcontrolador Motorola Freescale MC68H(R)C908, incorpora varios registros, que se utilizan para su programación y proyectos de desarrollo entre los más destacados están:

Acumulador (A): Compuesto por un registro de 8 bits, es un registro de propósito general utilizado en las operaciones aritméticas y lógicas.

Registro Índice (X): es un registro de 8 bits, se emplea para los modos de direccionamiento indexado y como acumulador auxiliar.

Registro Índice (H:X): es un registro compuesto de 16 bits, utilizado en modos de direccionamiento indexado, tiene la función de apuntador para la totalidad del mapa de memoria.

Puntero de PILA (SP): es un registro de 16 bits, contiene la dirección de la PILA.

Contador de Programa (PC): es un registro de 16 bits, con la función de almacenar la dirección de la siguiente posición de memoria, en el evento de Reset, el contador de programa es cargado con la dirección del vector de Reset.

Registro de Banderas (CCR): es un registro de 8 bits, que contiene varios flags o banderas entre ellos la bandera de rebosamiento (V) en el bit 7, bandera de medio carry (H) bit 4, bandera de habilitador global de interrupción (I) bit 3, bandera de valor negativo (N) bit 2, bandera de cero (Z) bit 1 y bandera de carry (C) bit 0.

Modos de direccionamiento

Los Microcontroladores Freescale usan seis modos de direccionamiento que son:

Inherente Las instrucciones de direccionamiento inherente no tienen ningún operando, ya que el operando se define en el 'opcode' de 8-bits. Por ejemplo para borrar el acumulador, se usa la instrucción CLRA, que es una instrucción de un sólo ciclo para el HC08; el HC05 toma 3 ciclos. EJ. CLRA

Inmediato Las instrucciones de direccionamiento inmediato tienen los operandos, que siguen inmediatamente al 'opcode', de 8-bits o de 16-bits. Cargar el acumulador con 20 es una instrucción de dos bytes, que se ejecuta en ciclos de bus. Ej. LDA #20

Extendido Las instrucciones de direccionamiento extendido proporcionan direccionamiento absoluto a cualquier posición en los 64K del mapa de memoria, sin paginar. Requieren en total tres bytes para el 'opcode' más la dirección de 16-bits del operando. La mayoría de los ensambladores usan el modo directo más corto automáticamente, para cualquier acceso a los primeros 256 bytes del mapa de memoria. Ej. LDA \$4000

Directo Las instrucciones de direccionamiento directo tienen la dirección de 8-bits del operando que sigue inmediatamente al 'opcode'. Por consiguiente, las instrucciones acceden directamente a los primeros 256 bytes de la memoria. Anteriormente, se hizo referencia a este tipo de acceso como página directa o página cero. Cargar el acumulador con el operando a la posición de memoria 40, es una instrucción de dos bytes que se ejecuta en 3 ciclos de bus. Ej. LDA \$40

Indexado Los modos de direccionamiento indexado son claves para direccionar tablas y otras estructuras de datos de una manera eficaz. El direccionamiento indexado sin desplazamiento ('offset'), se refiere a lo que en la mayoría de otras arquitecturas es el direccionamiento del puntero indirecto. El valor en el registro de índice es la dirección o el puntero del operando. Los modos de direccionamiento con 'offset' proporcionan al 68HC08 una gran eficacia de código comparado con otras arquitecturas con sólo direccionamiento indirecto o direccionamiento indirecto con otro registro de 'offset'

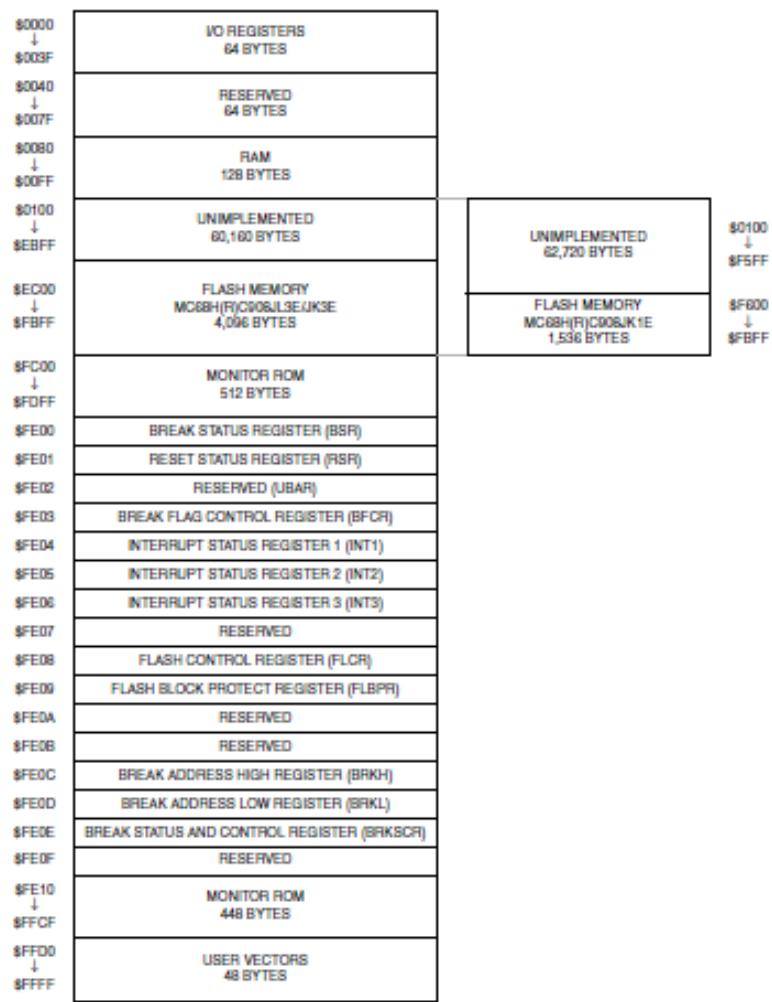
- sin desplazamiento Ej. LDA ,X
- con desplazamiento de 8 bits Ej. LDA \$40, X
- con desplazamiento de 16 bits Ej. LDA \$4000, X

Relativo. Todas las instrucciones de bifurcación condicional usan el direccionamiento relativo. Si la condición de bifurcación es verdad, el contador de programa se agrega al byte con signo, que sigue inmediatamente al 'opcode' de bifurcación. Esto da un rango de -128 a +127 bytes, para la bifurcación. Las instrucciones de salto o de salto a subrutina, se pueden usar para mover el contador de programa en cualquier parte de los 64K del mapa de memoria. Ej. BLT LOOP.

Como parte del proceso de aprendizaje y práctica con microcontroladores Motorola Freescale, se hace referencia al material bibliográfico "Microcontroladores Motorola Freescale: Programación, familias y sus distintas

“aplicaciones industriales” que es uno de los pocos textos que tratan el tema en forma didáctica en español y de fácil consecución en las librerías del país.

Organización de la memoria



Interrupciones

Se implementa un conjunto de vectores que responden a varias fuentes de interrupción. Entre ellas por evento de Reset, instrucciones SWI, pin IRQ1, Sobre flujo del Timer, interrupción por teclado e interrupción por conversión completa del ADC. Las interrupciones generan indicaciones en flags o banderas predeterminadas, cada vector de interrupción tiene una prioridad, por lo que es importante establecer dicho orden para adecuar la respuesta a una interrupción.

Figura 123. Fuentes de interrupción en Motorola Freescale

Priority	Source	Flag	Mask ¹⁽¹⁾	INT Register Flag	Vector Address
Highest ↑	Reset	—	—	—	\$FFFE-\$FFFF
	SWI Instruction	—	—	—	\$FFFC-\$FFFD
	IRQ1 Pin	IRQF1	IMASK1	IF1	\$FFFA-\$FFFB
	Timer Channel 0 Interrupt	CH0F	CH0IE	IF3	\$FFF6-\$FFF7
	Timer Channel 1 Interrupt	CH1F	CH1IE	IF4	\$FFF4-\$FFF5
	Timer Overflow Interrupt	TOF	TOIE	IF5	\$FFF2-\$FFF3
	Keyboard Interrupt	KEYF	IMASKK	IF14	\$FFE0-\$FFE1
	ADC Conversion Complete Interrupt	COCO	AIEN	IF15	\$FFDE-\$FFDF

Temporizadores

Se tiene un módulo interno de temporización denominado TIM, está conformado por 16 bits, con la capacidad de recibir pulsos de flanco externo o internos por el oscilador, con capacidad de comparación y contador. Tiene un pre escalador e incorpora funciones relacionadas con PWM, su desbordamiento es una de las fuentes de interrupción.

Principales módulos en los Motorola freescale

Entre los módulos más relevantes en estos microcontroladores, están los puertos I/O, Timers, PWM, conversores A/D, drivers para LEDs, Comparadores, módulo de bajo consumo, Watchdog y resistencias pull-down. Cada uno de estos módulos está vinculado o relacionado con varios registros, estos registros configuran el módulo, pueden guardar la información de llegada o salida del módulo y/o el estado de funcionamiento de módulo, expresado como un flag o bandera. Varios de los módulos generan estados de interrupción muy útil en la programación de los micros. Es necesario descargar desde el fabricante la hoja de datos o datasheet para conocer a fondo la arquitectura y el funcionamiento del microcontrolador.

Set de instrucciones

El 68HC05 tiene un total de 85 instrucciones que en la actualidad todavía forman un juego básico de instrucciones funcional y potente. El juego de instrucciones del 68HC08 se ha construido sobre la base de las instrucciones del 68HC05.

El 68HC08 añade 28 instrucciones al juego de instrucciones del 68HC05, remarcadas en la tabla siguiente. Muchas de estas instrucciones soportan el registro del índice extendido a 16 bits y el puntero de pila ('stack pointer') es

totalmente relocalizable. Listado de prefijos comúnmente utilizados en representaciones numéricas.

Tabla 42. Prefijos en los Motorola Freescale

PREFIJO	Tipo de valor que representa
t	Decimal
\$	Hexadecimal
@	Octal
%	Binario
Apóstrofe '	Carácter ASCII

Conjunto de Instrucciones

Tabla 43. Instrucciones Motorola Freescale¹³¹

INSTRUCCIÓN	OPERACIÓN	No. CICLOS
ADC #OPR	SUMA CON CARRY	2
ADC OPR		3
ADC OPR,X	A = A+(M)+C	3
ADC ,X	M = Dato o Valor Almacenado	2
ADC OPR,SP	C = Carry	4
ADD #OPR	SUMA SIN CARRY	2
ADD OPR		3
ADD OPR,X	A = A+(M)	3

¹³¹ VESGA, 2007

ADD ,X	M = Dato o Valor Almacenado	2
ADD OPR,SP		4
AIS #OPR	SP=SP+DATO	2
AIX #OPR	H:X = H:X + DATO	2
AND #OPR	FUNCION AND	2
AND OPR		3
AND OPR,X	A = A&(M)	3
AND ,X	M = Dato o Valor Almacenado	2
AND OPR,SP		4
ASL OPR	Desplazamiento Aritmético a la Izquierda	4
ASLA		1
ASLX		1
ASL OPR,X		4
ASL,X		3
ASL OPR,SP		5
ASR OPR	Desplazamiento Aritmético a la Derecha	4
ASRA		1
ASRX		1

ASR OPR,X		4
ASR OPR,SP		5
BCC Etiqueta	Saltar a la etiqueta si el Bit de Carry es 0	3
BCLR N,OPR	Borrar el bit N del registro OPR	4
BCS Etiqueta	Saltar a la etiqueta si el Bit de Carry es 1	3
BEQ Etiqueta	Saltar a la etiqueta si es Igual (Bit Z=1)	3
BGE OPR	Saltar si es Igual o mayor que OPR	3
BGT OPR	Saltar si es mayor que OPR	3
BHCC Etiqueta	Saltar a la etiqueta si el bit de Carry Medio es 0 (H)	3
BHCS Etiqueta	Saltar a la etiqueta si el bit de Carry Medio es 1 (H)	3
BHI Etiqueta	Saltar a la etiqueta si es Mayor	3
BHS Etiqueta	Saltar si es Mayor o Igual	3
BIH Etiqueta	Saltar si el Pin IRQ está en Alto	3

BIL Etiqueta	Saltar si el Pin IRQ está en Bajo	3
BIT #OPR		2
BIT OPR	Probar bits	3

BIT OPR,X		3
BIT ,X	A & (M)	2
BIT OPR,SP	M = Dato o Valor Almacenado	4
BLE OPR	Saltar si es Igual o Menor que OPR	3
BLO ETIQ	Saltar a la etiqueta si es Menor	3
BLS ETIQ	Saltar a la etiqueta si es Menor o igual	3
BLT OPR	Saltar si es Menor que	3
BMC ETIQ	Saltar si la bandera de interrupción esta en 0	3
BMI ETIQ	Saltar si el resultado de una operación es Negativo	3
BMS ETIQ	Saltar si la bandera de interrupción esta en 1	3
BNE ETIQ	Saltar a la etiqueta si no es igual	3
BPL ETIQ	Saltar si el resultado de una operación es Positivo	3
BRA ETIQ	Saltar a la Etiqueta siempre	3
BRCLR N,OPR,ETIQ	Saltar a la etiqueta si el bit N del registro OPR esta en 0	5
BRN ETIQ	Nunca Saltar	3
BRSET N,OPR,ETIQ	Saltar a la etiqueta si el bit N del registro OPR esta en 1	5
BSET N,OPR	Poner en 1 el bit N del registro OPR	4

BSR ETIQ	Saltar a Subrutina	4
CBEQ OPR,ETIQ		5
CBEQA #OPR,ETIQ	Comparar el valor de A con el valor #OPR o el dato almacenado en OPR y saltar si son iguales a la etiqueta	4
CBEQX #OPR,ETIQ		4
CBEQ OPR,X+,ETIQ		5
CBEQ OPR,SP,ETIQ		5
CLC	Borrar el Bit de Carry	1
CLI	Borrar el Bit de Interrupción o Bandera de Interrupción	2
CLR OPR		3
CLRA		1
CLRX	Borrar	1
CLRH		1
CLR OPR,X		3
CLR,X		2
CLR OPR,SP		4

CMP #OPR		2
CMP OPR	Comparar el valor de A con el valor #OPR o el dato almacenado en OPR	3
CMP OPR,X		3
CMP ,X		2
CMP OPR,SP		4
COM OPR		4
COMA		1
COMX	Complemento a uno	1
COM OPR,X		4
COM ,X		3
COM OPR,SP		5
CPHX #OPR	Comparar el valor de H:X con el valor #OPR o el dato almacenado en OPR	3
CPHX OPR		4
CPX #OPR		2
CPX OPR	Comparar el valor de X con el valor #OPR o el dato almacenado en OPR	3
CPX ,X		3
CPX OPR,X		2
CPX OPR,SP		4

DAA	Ajustar a decimal el registro A	2
DBNZ OPR,ETIQ		5
DBNZA ETIQ		3
DBNZX ETIQ	Decrementar y saltar si no es Cero	3
DBNZ OPR,X,ETIQ		5
DBNZ X,ETIQ		4
DBNZ OPR,SP,ETIQ		5
DEC OPR		4
DECA		1
DECX	Decrementar y saltar si no es Cero	1
DEC OPR,X		4
DEC ,X		3
DEC OPR,SP		5
DIV	Dividir $A=(H:A)/X$ (A = Cociente , H = Residuo)	7
EOR #OPR		2
EOR OPR	OR EXCLUSIVA	3
EOR OPR,X		3

eor , [⊕]	A = A (M)		2
eor OPR,SP	M = Dato o Valor Almacenado		4
INC OPR			4
INCA			1
INCX	Incrementar		1
INC OPR,X			4
INC ,X			3
INC OPR,SP			5

JMP OPR		2
JMP OPR,X	Saltar a la dirección OPR	3
JMP ,X		2
JSR OPR		4
JSR OPR,X	Saltar a Subrutina	5
JSR ,X		4
LDA #OPR		2
LDA OPR		3
LDA OPR,X	Cargar en A el valor #OPR o el dato almacenado en OPR	3

LDA ,X		2
LDA OPR,SP		4
LDHX #OPR	Cargar en H:X el valor #OPR o el dato almacenado en OPR	3
LDHX OPR		4
LDX #OPR		2
LDX OPR		3
LDX OPR,X	Cargar en X el valor #OPR o el dato almacenado en OPR	3
LDX ,X		2
LDX OPR,SP		4
LSL OPR		4
LSLA		1
LSLX	Desplazamiento Lógica a la Izquierda (Igual que ASL)	1
LSL OPR,X		4
LSL ,X		3
LSL OPR,SP		5
LSR OPR		4
LSRA	Desplazamiento Lógica a la Derecha	1
LSRX		1

LSR OPR,X		4
LSR ,X		3
LSR OPR,SP		5
MOV OPR,OPR		5
MOV OPR,X+	Mover Fuente, Destino	4
MOV #OPR,OPR		4
MOV X+,OPR		4
MUL	Multiplicación sin Signo ($X:A = X * A$)	5
NEG OPR		4
NEGA		1
NEGX	Complemento a Dos	1
NEG OPR,X		4
NEG ,X		3
NEG OPR,SP		5
NOP	No Operación	1
NSA	Intercambiar Nibbles de A ($A=(A[3:0]:A[7:4])$)	3

ORA #OPR		2
ORA OPR	FUNCION OR	3
ORA OPR,X		3
ORA ,X	A = A (M)	2
ORA OPR,SP	M = Dato o Valor Almacenado	4
PSHA	Insertar A en el Stack	2
PSHH	Insertar H en el Stack	2
PSHX	Insertar X en el Stack	2
PULA	Sacar A del Stack	2
PULH	Sacar H del Stack	2
PULX	Sacar X del Stack	2
ROL OPR		4
ROLA	Rotar a la Izquierda a través del Carry	1
ROLX		1
ROL OPR,X		4
ROL ,X		3
ROL OPR,SP		5

ROR OPR		4
RORA	Rotar a la Derecha a través del Carry	1
RORX		1
ROR OPR,X		4
ROR ,X		3
ROR OPR,SP		5
RSP	Reset al Stack Pointer	1
RTI	Retornar de una Interrupción	7
RTS	Retornar de una Subrutina	4
SBC #OPR		2
SBC OPR	RESTA CON CARRY	3
SBC OPR,X		3
SBC ,X	$A = A - (M) - C$	2
SBC OPR,SP	M = Dato o Valor Almacenado	4
SEC	Colocar el bit de Carry en 1	1
SEI	Colocar el bit de Interrupción en 1 I = 1	2
STA OPR		3
STA OPR,X	Asignar A en el registro OPR	4

STA ,X	M = A	2
STA OPR,SP	M = Dato o Valor Almacenado en dirección OPR	4
STHX OPR	Asignar H:X en el registro OPR	4
STOP	Habilitar el pin IRQ, detener el Oscilador	1
STX OPR		3
STX OPR,X	Asignar X en el registro OPR	4
STX ,X		2
STX OPR,SP		4

SUB #OPR		2
SUB OPR	RESTAR	3
SUB OPR,X		3
SUB ,X	A = A - (M)	2
SUB OPR,SP	M = Dato o Valor Almacenado	4
SWI	Interrupción por Software	9
TAP	Transferir A al CCR , CCR = A	2
TAX	Transferir A a X , X = A	1

TPA	Transferir CCR a A ; A = CCR	1
TST OPR		3
TSTA		1
TSTX	Probar si la cantidad es negativa o Cero	1
TST OPR,X		3
TST ,X		2
TST OPR,SP		4
TSX	Transferir SP a H:X , H:X = SP + 1	2
TXA	Transferir X a A , A = X	1
TXS	Transferir H:X a SP , SP = H:X - 1	2

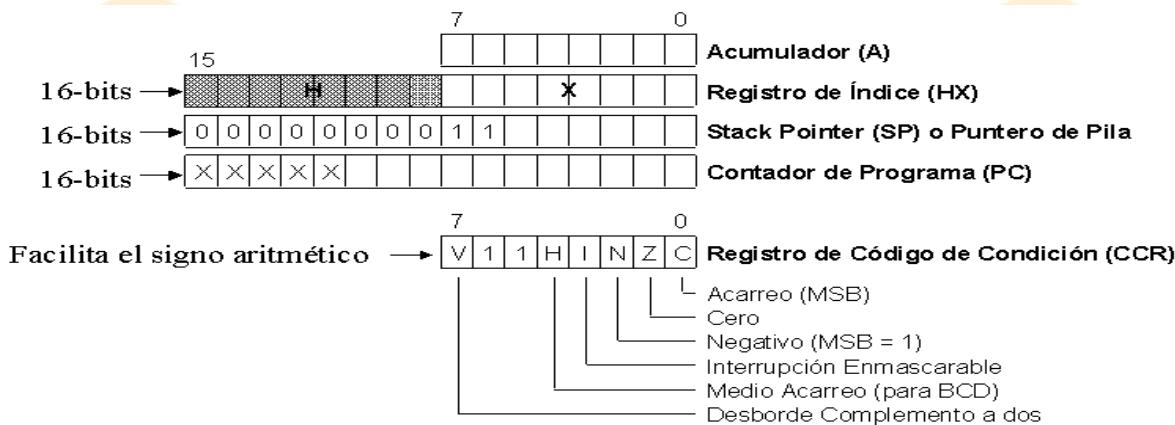
Lección 27: Microcontrolador MC68H(R)C908/JL3/JK3/JK1 y Ejemplos de aplicación

Los microcontroladores de 8 bits característicos en el aprendizaje de los microcontrolador Motorola Freescale son los MC68H(R)C908/JL3/JK3/JK1, en los siguientes párrafos se relacionan las generalidades de estos microcontroladores, desde su arquitectura, organización de la memoria, características generales, registros de uso general, funciones, diagramas de pines, puertos de entrada salida, módulos y algunos ejemplos de aplicación con este microcontrolador.

Arquitectura interna del MC68H(R)C908/JL3/JK3/JK1

El modelo de programación de la CPU del 68HC08 es una evolución del modelo de CPU del 68HC05. Cada 68HC08 implementa este modelo de CPU sin tener en cuenta el tamaño o el conjunto de características propias. A continuación se verán las mejoras específicas que se incluyen en la arquitectura 68HC08:

Figura 124. Arquitectura de registros en el MC68H(R)C908/JL3/JK3/JK1



El registro de índice: se ha ampliado a 16-bits, ayudando a manejar mejor las tablas o estructuras de datos que son mayores de 256 bytes.

El puntero de pila y el contador de programa: son registros de 16-bits, sin tener en cuenta la memoria interna disponible. Más adelante se verá la pila ('stack').

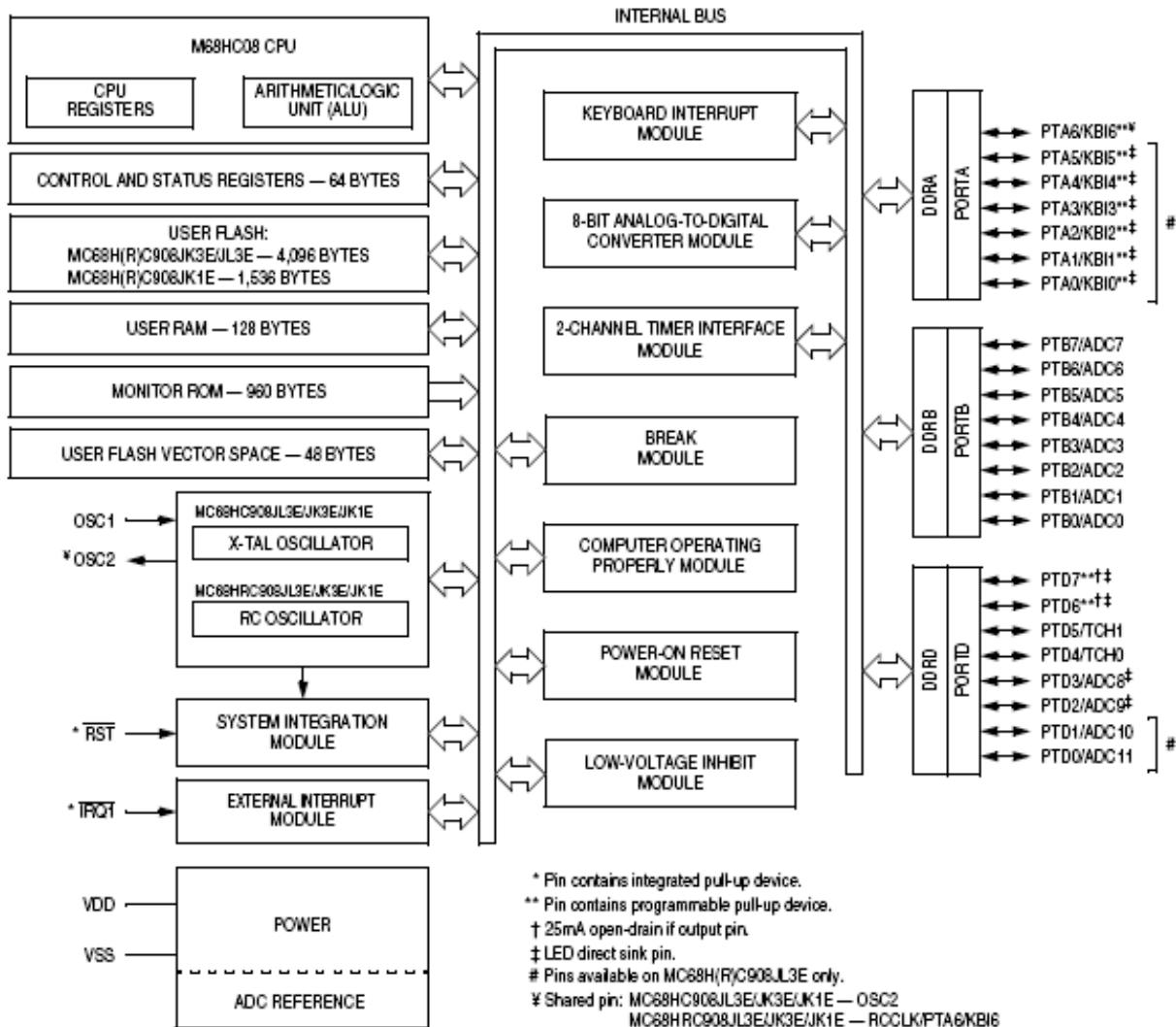
El Power-On Reset: el 68HC08 se parece al 68HC05. Durante el Reset, los 8 bits más altos del registro de índice HX, de los 16 bits que tiene, se ponen a cero y el puntero de pila se inicializa a \$00FF como en el 68HC05. Considerando que la mayoría de 68HC08 tienen mayor cantidad de RAM disponible, es probable que el usuario relocalizará el puntero de pila ('stack pointer'). Sin embargo, esta característica ayuda a mantener la compatibilidad operacional con el software existente del 68HC05.

En la CPU del 68HC08 el *bit V*, del registro de código de condición (CCR) facilita los cálculos aritméticos con signo. Esta mejora permite a los programadores de lenguaje ensamblador compiladores, realizar cálculos de direccionamiento mucho mejor.

El 68HC08 utiliza cuatro fases del reloj interno en cada ciclo de ejecución de la CPU. Si el 68HC08 está gobernado por un cristal, el ciclo de ejecución es un cuarto de la frecuencia del cristal. A este ciclo se le llama ciclo de bus o ciclo de instrucción. En el 68HC08, todos los tiempos de cada instrucción se especifican en ciclos de bus. Por ejemplo, un reloj de entrada de 32 MHz producirá una

frecuencia de bus de 8 MHz. Un ciclo de bus de una instrucción se ejecutará en 125 ns o 1 dividido por 8 MHz.

Figura 125. Arquitectura¹³²



Organización de la memoria en el MC68H(R)C908/JL3/JK3/JK1

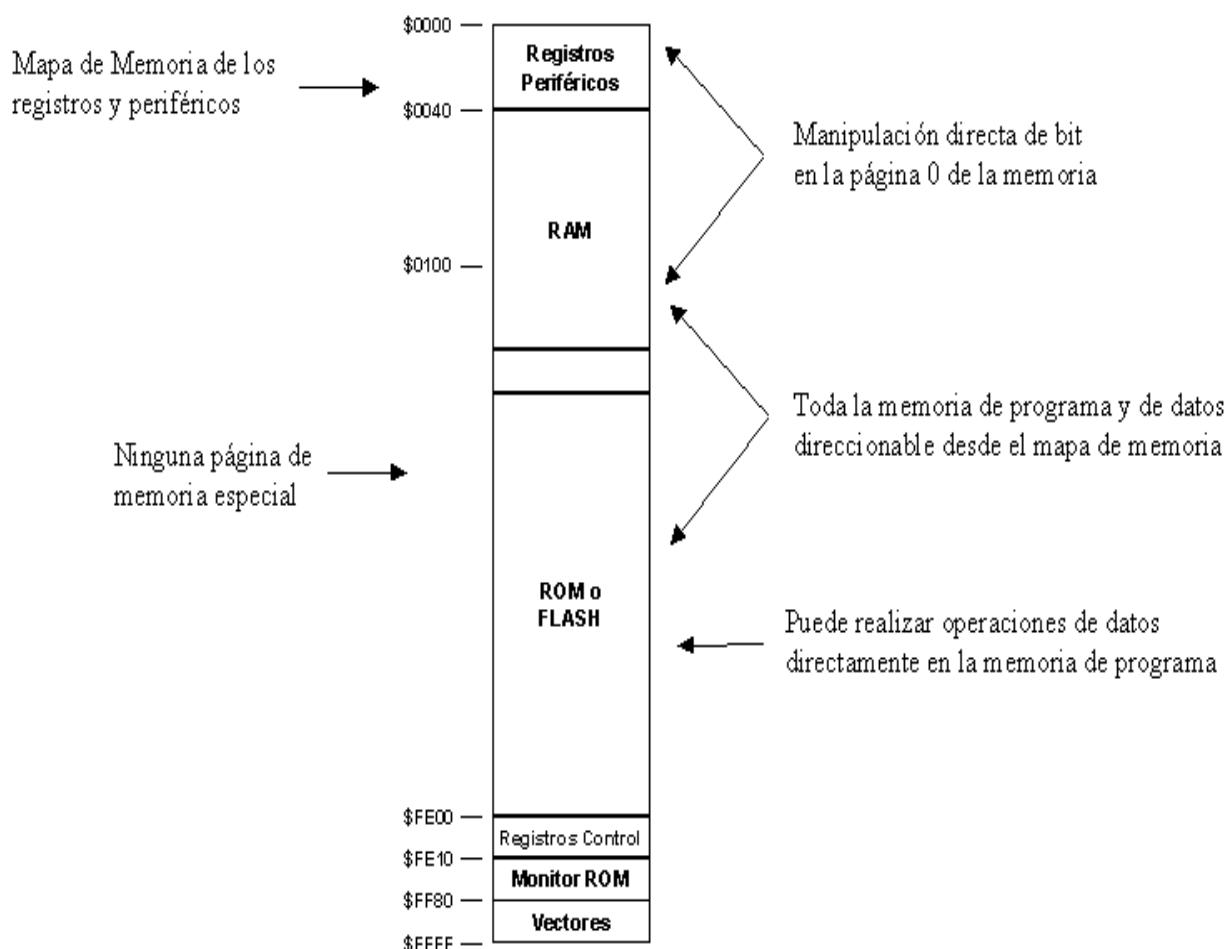
La figura siguiente muestra el mapa de memoria del 68HC08. Este mapa de memoria es idéntico al empleado para el 68HC05, sólo que se ha implementado hasta 64 KBytes, sin tener en cuenta los diferentes tamaños de memoria de cada 68HC08 disponible. En el 68HC08, el mapa de memoria empieza con los registros

¹³² Vesga, 2007

de los periféricos con 64 bytes. A continuación le sigue el espacio para la RAM. La ROM o FLASH ocupa la parte superior de la memoria que precede a \$FE00, donde están los Vectores, el programa Monitor y los Registros de control.

En el medio del mapa de memoria, entre la RAM y la ROM/FLASH, hay una zona de memoria no utilizable por el usuario que realiza las verificaciones de direcciones ilegales.

Figura 126. Bloque de memoria¹³³

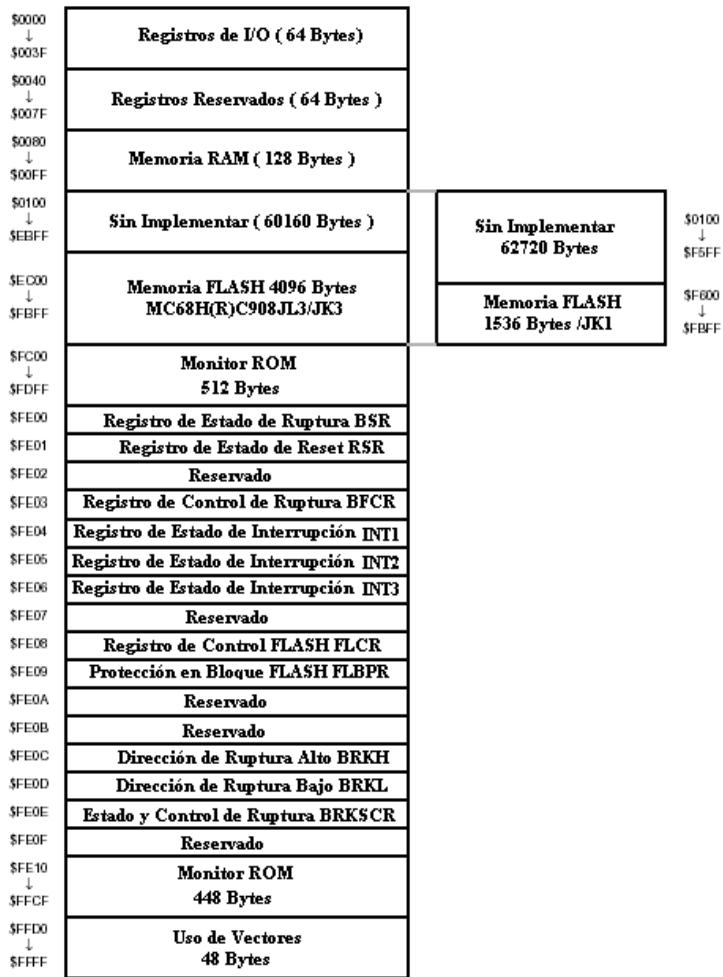


¹³³ Vesga, 2007

Todas las MCU de la familia 68HC08 usan este modelo, aunque algunas de ellas tienen una dirección de inicio y final para las áreas específicas del monitor ROM o los registros de control

Desde las direcciones \$0000 hasta la \$003F se encuentran todos los registros de control para entrada y salida de datos, configuración de los puertos A, B y D, Configuración y estado de los TIMERS, de los canales de conversión A/D, etc (VESGA, 2007).

Figura 127. Memoria en los Motorola Freescale, Fuente ¹³⁴



¹³⁴ VESGA, 2007

Características generales de los microcontroladores MC68H(R)C908/JL3/JK3/JK1

Entre las características más generales de estos microcontroladores están¹³⁵.

- CPU de 8 bits.
- Operación interna a 8 MHz.
- Rango de operación entre 3V y 5V.
- LVI: protección contra voltaje.
- Opción de oscilador con red RC o cristal.
- Sistema de programación Flash.
- 4096 Bytes para el MC68H(R)C908JL3/JK3, 1536 Bytes para el MC68H(R)C908JK1.
- 128 Bytes de memoria RAM.
- 2 Timers de 16 bits.
- 12 canales de conversores A/D de 8 bits (JL3), 10 canales de conversión A/D para JK3.
- 23 pines de entrada/salida para uso general (JL3). Con 7 interrupciones de teclado con resistencias pull-up, 10 drivers para LEDs, 2 ICAP/OCAP/PWM.
- 15 pines de entrada/salida para uso general (JK3/JK1). Con 1 interrupción por teclado con resistencia pull-up, 4 drivers para LED, 2 ICAP/OCAP/PWM.
- Modo de bajo consumo.
- Perro guardián.
- Fuentes de interrupción sectorizada.

Registros de uso general

ACUMULADOR (A) : Es un registro de 8 bits de propósito general de lectura y escritura utilizado por la CPU para el almacenamiento temporal de los operandos y los resultados de las operaciones aritméticas y lógicas realizadas por la ALU.

REGISTRO ÍNDICE (H:X) : Es un registro de 16 bits que permite direccionar hasta 64 K de memoria en forma indexada, está dividido en dos, el primero denominado H con el byte alto (8 bits más altos) y el byte bajo (8 bits más bajos) se denomina X, con lo cual se construye un registro de 16 bits, o 2 bytes H:X lo que permite apuntar a cualquiera de las 64 K de memoria.

¹³⁵ VESGA, 2007

REGISTRO ÍNDICE (X) : El registro índice es usado para los modos de direccionamiento indexados o bien puede ser usado como un acumulador auxiliar, está constituido por 8 bits, debe recordarse que su contenido determina la dirección efectiva del operando.

PUNTERO DE PILA (SP) : (Stack Pointer), Es un registro de 16 bits que contiene la dirección de la próxima posición de 8 bits dentro del Stack (pila).

CONTADOR DE PROGRAMA (PC) : Es un registro de 16 bits que contiene la dirección de la siguiente instrucción u operación a procesar, la CPU aumenta el valor del contador en forma secuencial, guardando la siguiente posición de memoria.

REGISTRO DE BANDERAS (CCR): Conocido en otros microcontroladores como el registro de estado, Es un registro de 8 bits que contiene el bit habilitador de interrupción y 5 banderas o flags de estado, tomando como referencia el material bibliográfico “Microcontroladores Motorola Freescale” (VESGA,2007) se ilustran el diagrama y funciones de los flags.

Figura 128. Registro de banderas (CCR) Motorola Freescale.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Leer								
Escribir	V	1	1	H	I	N	Z	C

V : Bandera de rebosamiento o bit indicador de sobre flujo

La CPU coloca esta bandera en 1 cuando al efectuar el complemento a dos ocurre un rebosamiento.

H : Bandera de Medio Carry o bit de acarreo intermedio

La CPU coloca este bit en 1 cuando ocurre un carry entre los bits 3 y 4 durante una suma con o sin carry, el medio carry es requerido cuando se utiliza codificación en BCD.

I : Mascara de interrupciones o bit habilitador de interrupciones

Cuando este bit se coloca en 1 lógico todas las interrupciones son deshabilitadas, y son habilitadas nuevamente cuando este bit es colocado en 0 lógico.

N : Bandera de valor negativo o bit de indicador negativo

Este bit es colocado en 1 lógico cuando el resultado de una operación aritmética es **Negativa**.

Z : Bandera de Cero

Este bit es colocado en 1 lógico cuando el resultado de una operación aritmética o lógica de cómo resultado **CERO**.

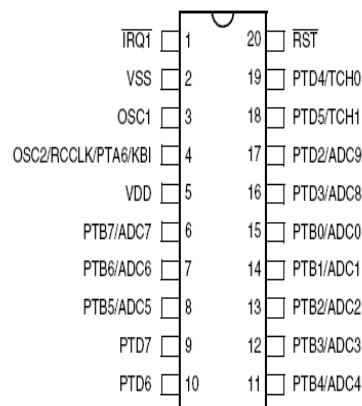
C : Bandera de Carry o bit de acarreo

Este bit es colocado en 1 lógico cuando el resultado de una operación aritmética produce Carry después del bit 7.

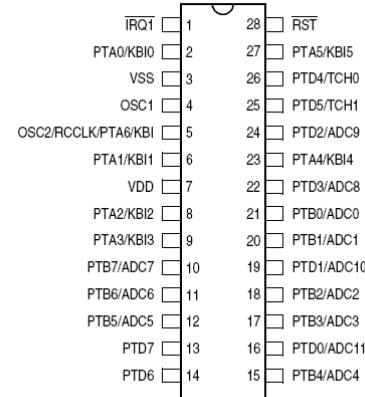
Funciones y Diagrama de pines

En la siguiente figura se observa la distribución de pines del MC68H(R)C908L3 y MC68H(R)C908JK3/JK1

Figura 129. JL3, JK3 y JK1¹³⁶



MC68H(R)C908JK3E/JK1E



MC68H(R)C908JL3E

Las siguiente tabla se discrimina la función(es) de pines, en el MC68H(R)C908JK3E/JK1E, no están disponibles los siguientes pines *PTA0, PTA1, PTA2, PTA3, PTA4, PTA5, PTD0, and PTD1*

Puertos de entrada / salida (I/O)

Los Puertos del microcontrolador son de gran importancia pues a través de ellos se controlan cargas, dispositivos o sirven para recibir información del entorno del microcontrolador. En el Microcontrolador MC68H(R)C908JL3E, 23 pines pueden ser configurados de manera bidireccional (I/O), a través de tres puertos paralelos. Todos los pines pueden ser configurados como entrada o salida.

Se Debe tener en cuenta que en la gran mayoría de familias de Microcontroladores, los puertos de entrada/salida no solamente cumplen funciones

¹³⁶ Vesga, 2007

de envío y recepción de señales digitales, sino que además comparten recursos internos con el Microcontrolador. (VESGA, 2007). Como todo dispositivo basado en tecnología FET (Transistor de efecto de Campo) es recomendable terminar los pines no utilizados en las aplicaciones, es decir conectarlos a cero voltios (0 V) o a cinco voltios (5 V).

Figura 130. Configuración de pines¹³⁷

PIN NAME	PIN DESCRIPTION	IN/OUT	VOLTAGE LEVEL
VDD	Power supply.	In	5V or 3V
VSS	Power supply ground	Out	0V
RST	RESET input, active low. With Internal pull-up and schmitt trigger input.	Input	VDD
IRQ1	External IRQ pin. With software programmable internal pull-up and schmitt trigger input. This pin is also used for mode entry selection.	Input	VDD to VDD+V _{HI}
OSC1	X-tal or RC oscillator input.	In	Analog
OSC2	MC68HC908JL3E/JK3E/JK1E: X-tal oscillator output, this is the inverting OSC1 signal.	Out	Analog
	MC68HRC908JL3E/JK3E/JK1E: Default is RC oscillator clock output, RCCLK. Shared with PTA6/KBI6, with programmable pull-up.	In/Out	VDD
PTA[0:6]	7-bit general purpose I/O port.	In/Out	VDD
	Shared with 7 keyboard interrupts KBI[0:6].	In	VDD
	Each pin has programmable internal pull-up device.	In	VDD
	PTA[0:5] have LED direct sink capability	In	VSS
PTB[0:7]	8-bit general purpose I/O port.	In/Out	VDD
	Shared with 8 ADC inputs, ADC[0:7].	In	Analog
PTD[0:7]	8-bit general purpose I/O port.	In/Out	VDD
	PTD[3:0] shared with 4 ADC inputs, ADC[8:11].	Input	Analog
	PTD[4:5] shared with TIM channels, TCH0 and TCH1.	In/Out	VDD
	PTD[2:3], PTD[6:7] have LED direct sink capability	In	VSS
	PTD[6:7] can be configured as 25mA open-drain output with pull-up.	In/Out	VDD

REGISTRO PORT A (PTA) Dirección \$0000

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Leer	0							
Escribir		PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0

¹³⁷ Vesga, 2007

Es un registro de datos, compuesto por 7 bits de lectura y escritura, se pueden configurar como 7 entradas externas de interrupción para el manejo de teclados , todos implementan resistencias internas pull-ups con capacidad de conectarlas o desconectarlas mediante configuración del software. Los terminales entre PTA= y PTA5 poseen fuentes limitadas de corriente destinadas al manejo de diodos LED, es decir no es necesario las resistencias limitadoras de corriente.

REGISTRO DE CONFIGURACION DEL PORT A (DDRA) Dirección \$0004

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Leer	0							
Escribir		DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0

Mediante este registro de 8 bits, de los cuales a solo 7 bits se puede acceder, se efectúa la configuración de los bits del registro PORT A, ya sea como entradas, con un uno (1) lógico o salidas con un cero (0) lógico. La acción de reset borra todos los bits de control, programando todo el puerto como entrada.

REGISTRO PTAPUE Dirección \$000D

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Leer								
Escribir	PTA6EN	PTAPUE6	PTAPUE5	PTAPUE4	PTAPUE3	PTAPUE2	PTAPUE1	PTAPUE0

Este registro es el que permite habilitar o deshabilitar las resistencias de Pull-Up para cada uno de los pines del puerto A. Esta configuración es valida siempre que los pines estén configurados como entradas, la CPU desconecta las pull-up automáticamente cuando se configura el pin como salida.

REGISTRO PORT B (PTB) Dirección \$0001

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Leer								
Escribir	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0

El registro PORTB, asociado a 8 terminales bidireccionales, permite la manipulación de señales digitales entre sus terminales tal como ocurre con el registro PORTA, adicionalmente cada pin de este puerto se asocia a un canal del módulo de conversión Análogo/Digital incorporado en el dispositivo.

REGISTRO DE CONFIGURACION DEL PORT B (DDRB) Dirección \$0005

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Leer	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
Escribir								

Este registro es el responsable de la configuración entrada / salida del puerto B se siguen los mismos parámetros utilizados en el registro de configuración DDRA.

REGISTRO PORT D (PTD) Dirección \$0003

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Leer	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
Escribir								
Función 1	LED	LED	TCH1	TCH0	LED	LED		
Función 2	25mA	25mA			ADC8	ADC9	ADC10	ADC11

El registro PORTD esta compuesto por 8 bits de lectura escritura, poseen funciones especiales de entrada y salida digital, dos de las terminales asociadas comparten su función con la interfaz del módulo temporizador (TCH1 y TCH2) mientras que otras cuatro comparten entradas del módulo de conversión Análoga / Digital (A/D) y dos pines (PTD6 y PTD7) con driver se corriente (25 mA) y resistencias pull-up programables.

REGISTRO DE CONFIGURACION DEL PORT D (DDRD) Dirección \$0007

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Leer	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
Escribir								

Este registro de 8 bits es el responsable de la configuración de los bits del registro PORT D, como entradas o salidas.

REGISTRO DE CONTROL DEL PORT D (PDCR) Dirección \$000A

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Leer	0	0	0	0				
Escribir					SLOWD7	SLOWD6	PTDPU7	PTDPU6

Su función es muy similar al del registro PTAPUE, habilita o deshabilita las resistencias pull-up, junto con el manejo de los drivers de corriente para los pines PTD6 Y PTD7.

Módulos en los MC68H(R)C908/JL3/JK3/JK1

Se implementan módulos de puertos I/O ya estudiados, Timers de 16 bits, conversores análogos / digitales (A/D), drivers para LED, modulo comparador, contador y PWM, junto con un módulo de bajo consumo, watchdog y varias fuentes de interrupción.

Ejemplos de programación con MC68H(R)C908/JL3/JK3/JK1

Igual como en los microcontroladores PIC, en la familia de los Motorola Freescale tienen un “Entorno de Desarrollo Integrado”, llamado WINIDE. Motorola y otras empresas independientes han desarrollado varias herramientas como emuladores, analizadores lógicos, programadores, tarjetas de evaluación, tarjetas de desarrollo, simuladoras, compiladoras en lenguaje C, ensambladores y depuradores. El desarrollo e implementación de los siguientes ejercicios básicos, permitirán una apropiación más efectiva de los conceptos y habilidades en la programación y ejecución de aplicaciones basadas en el microcontrolador Motorola Freescale.

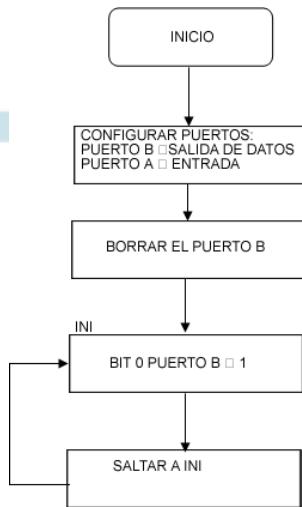
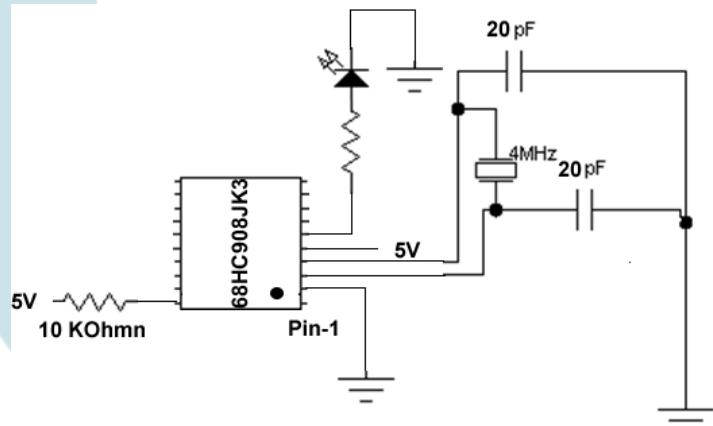
WinIDE – Intermitencia de un LED

Encender un LED: El objetivo es encender un LED conectado a una línea del puerto D en su pin PTD7 del microcontrolador 68HC908JK3.

Entradas y salidas: en este caso no hay entradas solo una salida donde se conectara un diodo LED, elegimos el puerto “D” en su pin PTD7 como salida de datos.

Pseudocódigo:

INICIA:
 CONFIGURAR PUERTOS:
 PUERTO D → SALIDA DE DATOS
 INI:
 BORRAR EL PUERTO D
 ACTIVAR EL BIT 7 → 1
 SALTAR A RUTINAINI
TERMINA

Diagrama de flujo:**Montaje:**

Código ensamblador:

```
$INCLUDE 'JL3REGSG.INC'          ;ARCHIVO INCLUIDO DONDE SE DEFINEN LOS REGISTROS
;INGENIERO ELECTRONICO HECTOR URIEL VILLAMIL GONZALEZ
;ENCENDIDO DE UN LED
;PINES DE SALIDA LEDs ==> D7
;PINES DE ENTRADA ==> "NINGUNO"
;-----[REDACTED]
;REGISTROS DE PROPOSITO ESPECIAL MAS UTILIZADOS
RAMSTART    EQU    $0080 ;DIRECCIONES RAM Y ROM EN LA MEMORIA LINEAL
ROMSTART    EQU    $EC00
VECTORES     EQU    $FFDE
;ROMFIN      EQU    $FB00
RESET_VEC   EQU    $FFFF
;-----[REDACTED]
ORG    RAMSTART          ;DEFINICION DE LAS VARIABLES EN MEMORIA RAM
CONTADOR    RMB    1       ;RESERVACION DE MEMORIA PARA LAS VARIABLES
REGIS1      RMB    1
REGIS2      RMB    1
;-----[REDACTED]
;INICIO DEL PROGRAMA
;-----[REDACTED]
ORG    ROMSTART          ;INICIO DE LA MEMORIA DE PROGRAMA
;-----[REDACTED]
;CONFIGURACION DE PUERTOS
;-----[REDACTED]
INICIO    RSP
BSET      COPD,  CONFIG1      ;INICIAR EL PUERTO A, LA PILA (SP)
CLRA      ;DESHABILITA EL COP (WATCHDOG)
CLRX      ;INICIAR ACUMULADOR
MOV       #$80,  DDRD      ;INICIAR REGISTRO X
;-----[REDACTED]
INICIO    CLR      PORTD      ;CONFIGURAR EL PIN7 DEL PUERTO D COMO SALIDA
BSET      7,      PORTD      ;INICIAR PORTD
JSR      INI
ORG       RESET_VEC      ;ACTIVO EL BIT 7 DEL PUERTO D, DONDE SE ENCUENTRA EL LED
DW        INICIO          ;CERRAR EL CICLO RETORNANDO A "INI"
;-----[REDACTED]
;-----[REDACTED]
;-----[REDACTED]
```

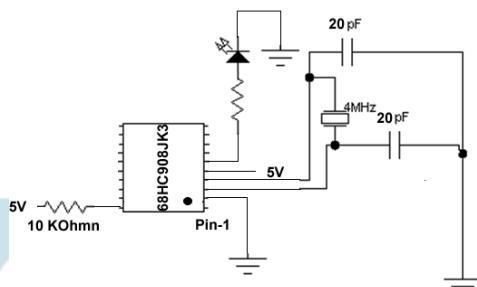
Contrario a lo que sucede en los microcontroladores PIC, la arquitectura de los Motorola Freescale, poseen un mapa lineal de memoria, por lo que no se necesita direccionar bancos de memoria. La directiva “include” permite incluir archivos adicionales que definen subrutinas o definiciones estandarizadas que son invocadas dentro del programa.

WinIDE – Encendido y Apagado de un LED utilizando un interruptor

Intermitencia de un LED: El objetivo es encender de forma intermitente un LED conectado a una línea del puerto D en su pin PTD del microcontrolador 68HC908JK3.

Entradas y salidas: en este caso no hay entradas solo una salida donde se conectara un diodo LED, elegimos el puerto "D" en su pin PTD como salida de datos, mas un retardo. El programa se basa en el anterior e incorpora una rutina de retardo.

Montaje



Código ensamblador

```
$INCLUDE 'JL3REGSG.INC' ;ARCHIVO INCLUIDO DONDE SE DEFINEN LOS REGISTROS Y BITS
;INGENIERO ELECTRONICO HECTOR URIEL VILLAMIL GONZALEZ
;ENCENDIDO DE UN LED
;PINES DE SALIDA LEDS ==> D7
;PINES DE ENTRADA ==> "NINGUNO"
;-----
;REGISTROS DE PROPOSITO ESPECIAL MAS UTILIZADOS
RAMSTART EQU $0080 ;DIRECCIONES RAM Y ROM EN LA MEMORIA LINEAL
ROMSTART EQU $EC00
VECTORES EQU $FFDE
;ROMFIN EQU $FB00
RESET_VEC EQU $FFFE
;-----
ORG RAMSTART ;DEFINICION DE LAS VARIABLES EN MEMORIA RAM
CONTADOR RMB 1 ;RESERVACION DE MEMORIA PARA LAS VARIABLES
REGIS1 RMB 1
REGIS2 RMB 1
;-----
;INICIO DEL PROGRAMA
;-----
ORG ROMSTART ;INICIO DE LA MEMORIA DE PROGRAMA
;-----
;CONFIGURACION DE PUERTOS
;-----
INICIO RSP
BSET COPD, CONFIG1 ;INICIAR EL PUERTO A, LA PILA (SP)
CLRA ;DESHABILITA EL COP (WATHDOG)
CLR X ;INICIAR ACUMULADOR
CLRX ;INICIAR REGISTRO X
MOV #$80, DDRD ;CONFIGURAR EL PIN7 DEL PUERTO D COMO SALIDA
INICIO CLR PORTD ;INICIAR PORTD
BSET 7, PORTD ;ACTIVAR EL BIT 7 DEL PUERTO D, DONDE SE ENCUENTRA EL LED
JSR RET10S ;SALTO A LA SUBRUTINA DE RETARDO
BCLR 7, PORTD ;UNA VEZ TERMINADO EL RETARDO, SE PROCEDE A APAGAR EL
LED JSR RET10S ;LLAMADO A RETARDO
JSRINI ;CERRAR EL CICLO RETORNANDO A "INI"
```

```
;-----  
RET01 LDA    #128T      ;GENERACION DE RETARDO  
ET2    CLRX   ;CARGA DEL ACUMULADOR CON EL NUMERO  
ET3    DECX   ;COMIENZA LA ESTRUCTURA REPETITIVA  
       BNE    ET3      ;REGISTRO DECREMENTANDO HASTA LLEGAR A CERO  
       DECA   ;SE RETORNA UN "POSMEN" DESPUES DEL LLAMADO  
       BNE    ET2      ;AL DARSE RESET SALTAR A INICIO  
       RTS    ;  
ORG   RESET_VEC ;  
DW    INICIO   ;-----  
;  
END  
;-----
```

Lección 28: Microcontroladores Texas MSP430

Los microcontroladores Texas Instruments con su serie de ultra bajo poder, está posicionándose en el mercado como uno de los dispositivos de fácil programación, pero principalmente por el costo económico de su sistema de desarrollo LaunchPad MSP430.

Microcontroladores Texas Instruments MSP430

Mixed Signal Processor o procesadores de señal mixta, son sistemas digitales con módulos con capacidad para manejo de señales análogas, mediante conversores A/D, lo que los convierte en una herramienta robusta para aplicaciones de medición, control y consumo. La configuración correcta de interrupciones, fuentes de reloj y periféricos, permiten bajar el consumo de energía, posicionándolos como los microcontroladores con menos consumo del mercado, por debajo de otras familias de bajo consumo como los PIC.

Arquitectura

Implementan una arquitectura de instrucciones RISC de 16 bits, una parte de ellas son físicamente implementadas y otras son emuladas. Implementan la arquitectura Von Neumann, la familia de microcontroladores Texas MSP430 está formada por 5 generaciones para un total de más de 200 dispositivos, caracterizados por ultra bajo consumo, implementar pines con múltiples funciones como conversores A/D, timers, comparadores, contadores, PWM y módulos de comunicación.

Implementa un ALU de 16 bits, con un set de 27 instrucciones y 7 tipos de direccionamiento, arquitectura ortogonal en las instrucciones, registros de 16 bits, bus de direcciones de 16 bits, instrucciones y modos de direccionamiento para 8 y 16 bits, 12 registros de propósito general para almacenamiento de datos y direcciones y 4 registros con las constantes más usuales.

Funcionamiento

El MPS430 es un microcontrolador en encapsulado DIP o DIE de montaje superficial, particularmente para el LaunchPad MSP430 en formato de 14 hasta 20 pines tipo DIP. Puede funcionar directamente en la protoboard o placa impresa o mediante la placa de desarrollo LaunchPad con la que puede conectarse al protoboard o realizar simulación In-circuit desde el PC.

La arquitectura utiliza un formato de memoria lineal, cuenta con bastantes registros de propósito general directamente relacionados con el ALU, permitiendo una mayor flexibilidad a parte de los registros que puede definir el usuario.

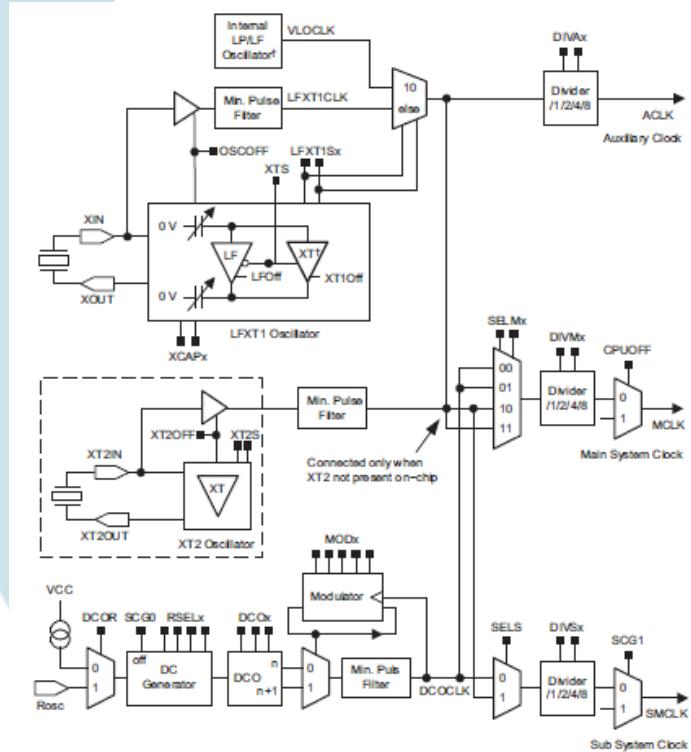
Características Generales de los microcontroladores

Además de tener herramientas de bajo costo para desarrollo de proyectos como los LaunchPad, con un formato DIL (Dual In Line – Doble línea de pines), con la que se puede conectar fácilmente la tarjeta de desarrollo en la protoboard del prototipo. Los dispositivos MSP430 se presentan en la forma de montaje superficial, lo que trae la ventaja de reducción de espacio en la implementación impresa del proyecto terminado.

Oscilador, circuito de 0052eset

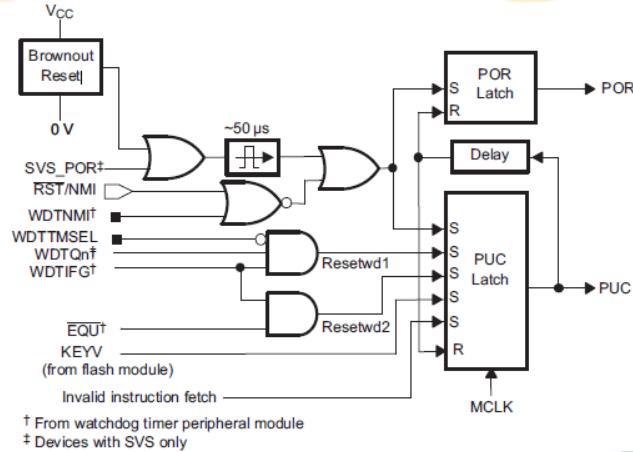
Se tienen tres fuentes de reloj VLOCLK oscilador de bajo consumo y frecuencia (12KHz), LFXT1CLK oscilador de alta o baja frecuencia y DCOCLK oscilador controlado digitalmente que puede generar frecuencias hasta de 16MHz. La señales de reloj que entrega a los periféricos se denominan ACLK como reloj auxiliar, MCLK como reloj principal y SMCLK como reloj sub principal.

Figura 131. Circuito de reloj en MSP430.



El sistema de Reset puede ser accionado por un reset del tipo POR (Power On Reset) y BOR (Brown Out Reset), al energizar el chip, por una señal baja en el pin $\overline{RST/NMI}$ o por otros eventos como el PUC (Power – Up Clear), como el mismo POR, el Watchdog, violación de seguridad en la memoria flash y por instrucción en la CPU.

Figura 132. Sistema de Reset en el MSP430



Registros

Entre los principales registros se encuentra el registro Contador de Programa PC de 16 bits, el registro Stack Pointer (SP) de 20 bits, Registro de Estado (SR) que tiene las banderas de Overflow (V), SCG1 (System Clock Generator 1), SCG2 (System Clock Generator 2), OSCOFF (apagar oscilador), CPUOFF (apagado de CPU), GIE (habilitador de interrupciones globales), N flag negativo, Z flag de Zero y C flag de Carry.

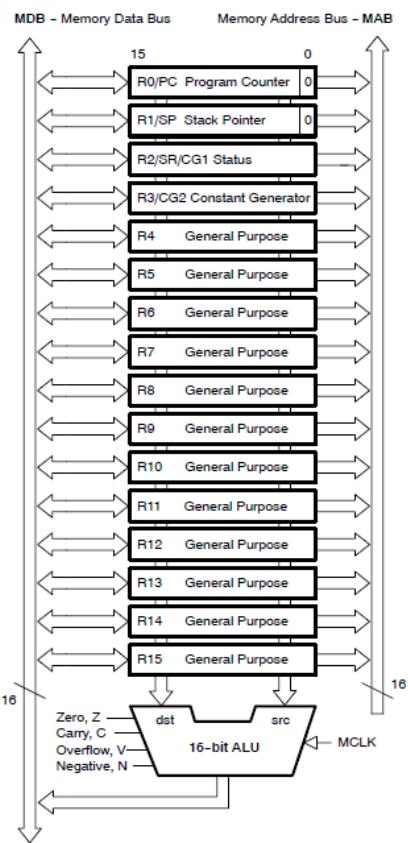
Tambien se tiene los registros Constant Generator Register CG1 y CG2, el registro Constant Generator – Expanded Instruction Set, registro que permite extender 24 instrucciones más al set de instrucciones como instrucciones emuladas. Registros de Proposito General compuesto por 12 registros desde el R4 al R15.

Organización de la memoria

La memoria está organizada internamente como un bloque lineal de memoria, donde se puede acceder a cualquier registro por medio de instrucciones, esta memoria contiene los registro de propósito especial, compuesto por 16 registros localizados entre las localidades de memoria 0000H y 000FH, luego siguen los registros de los periféricos tanto los que son accedidos por instrucciones de Byte (.B) y los que se acceden por instrucción de palabra (.W), la memoria RAM comienza en la locación de memoria 0200H y termina según la cantidad de

memoria que tenga dispuesta el dispositivo desde 128 hasta 2K (2048), Luego está la memoria de arranque para dispositivos flash en la localidades 0C00H a 0FFFH, la memoria de información para dispositivos flash, luego la memoria de código o programa que puede estar en rangos desde los 2 K hasta 60 K de memoria, el final del segmento de memoria de código o programa está ubicado en la dirección FFDFH, finalmente se encuentran los vectores de interrupción ubicados desde la dirección FFE0H hasta la FFFFH.

Figura 133. Registros de MSP430



Modos de direccionamiento

Entre los modos de direccionamiento se encuentra, el modo de direccionamiento de registro, modo de direccionamiento Indexado, modo de direccionamiento simbólico, modo de direccionamiento absoluto, modo de direccionamiento Indirecto, modo de direccionamiento indirecto con autoincremento, modo de direccionamiento inmediato.

Interrupciones

Las interrupciones se dispone en un arreglo de vectores de interrupción dispuesto en el mapa de memoria, las fuentes de interrupción tienen prioridades y están directamente relacionadas con el tipo de microcontrolador. Están las interrupciones por Reset, interrupciones no enmascarables NMI, por falla del oscilador y violación de seguridad de memoria flash, interrupción por Watchdog, interrupción por Timers, interrupciones por periféricos ADC, USI, puertos I/O, etc.

Temporizadores

Los temporizadores son de 16 bits, funcionan como base generador de tiempos, como contador alimentado por flanco de pulsos interno o externo, con pre escalador para dividir la señal de frecuencia de entrada y lograr periodos de tiempo más largos.

Principales módulos en los MSP430

Los principales módulos de los microcontroladores MSP son al igual que otros microcontroladores los puertos I/O, timers, conversores A/D, comparadores, PWM, módulos de comunicación, pero principalmente están los módulos encargados del oscilador y reloj junto con los módulos que manejan las interrupciones para lograr un manejo de ultra bajo consumo.

Set de instrucciones

El set de instrucciones está compuesto por instrucciones implementadas en el codificador del micro e instrucciones emuladas.

Figura 134. Instrucciones MSP430

Mnemonic	Description			V	N	Z	C
ADC (.B) ⁽¹⁾	dst	Add C to destination	dst + C → dst	*	*	*	*
ADD (.B)	src, dst	Add source to destination	src + dst → dst	*	*	*	*
ADDC (.B)	src, dst	Add source and C to destination	src + dst + C → dst	*	*	*	*
AND (.B)	src, dst	AND source and destination	src .and. dst → dst	0	*	*	*
BIC (.B)	src, dst	Clear bits in destination	not.src .and. dst → dst	-	-	-	-
BIS (.B)	src, dst	Set bits in destination	src .or. dst → dst	-	-	-	-
BIT (.B)	src, dst	Test bits in destination	src .and. dst	0	*	*	*
BR ⁽¹⁾	dst	Branch to destination	dst → PC	-	-	-	-
CALL	dst	Call destination	PC+2 → stack, dst → PC	-	-	-	-
CLR (.B) ⁽¹⁾	dst	Clear destination	0 → dst	-	-	-	-
CLRC ⁽¹⁾		Clear C	0 → C	-	-	-	0
CLRN ⁽¹⁾		Clear N	0 → N	-	0	-	-
CLRZ ⁽¹⁾		Clear Z	0 → Z	-	-	0	-
CMP (.B)	src, dst	Compare source and destination	dst - src	*	*	*	*
DADC (.B) ⁽¹⁾	dst	Add C decimaly to destination	dst + C → dst (decimaly)	*	*	*	*
DADD (.B)	src, dst	Add source and C decimaly to dst	src + dst + C → dst (decimaly)	*	*	*	*
DEC (.B) ⁽¹⁾	dst	Decrement destination	dst - 1 → dst	*	*	*	*

Figura 135. Instrucciones MSP430 - continuación

Mnemonic	Description	V	N	Z	C
DBCD (.B) ⁽¹⁾ dst	Double-decrement destination	dst - 2 → dst	*	*	*
DINT ⁽¹⁾	Disable Interrupts	0 → GIE	-	-	-
RINT ⁽¹⁾	Enable Interrupts	1 → GIE	-	-	-
INC (.B) ⁽¹⁾ dst	Increment destination	dst +1 → dst	*	*	*
INCD (.B) ⁽¹⁾ dst	Double-Increment destination	dst+2 → dst	*	*	*
INV (.B) ⁽¹⁾ dst	Invert destination	.not.dst → dst	*	*	*
JC/JHS	Jump if C set/Jump if higher or same	-	-	-	-
JBQ/JZ	Jump if equal/Jump if Z set	-	-	-	-
JGE	Jump if greater or equal	-	-	-	-
JL	Jump if less	-	-	-	-
JMP	Jump	PC + 2 X offset → PC	-	-	-
JN	Jump If N set	-	-	-	-
JNC/JLO	Jump If C not set/Jump If lower	-	-	-	-
JNE/JNZ	Jump If not equal/Jump If Z not set	-	-	-	-
MOV (.B)	src,dst	Move source to destination	src → dst	-	-
NOP ⁽²⁾	No operation	-	-	-	-
POP (.B) ⁽²⁾	dst	Pop item from stack to destination	@SP → dst, SP+2 → SP	-	-
PUSH (.B)	src	Push source onto stack	SP - 2 → SP, src → @SP	-	-
RET ⁽²⁾		Return from subroutine	@SP → PC, SP + 2 → SP	-	-
RETI		Return from interrupt	-	-	-
RLA (.B) ⁽²⁾	dst	Rotate left arithmetically	-	-	-
RLC (.B) ⁽²⁾	dst	Rotate left through C	-	-	-
RRA (.B)	dst	Rotate right arithmetically	0	-	-
RRC (.B)	dst	Rotate right through C	-	-	-
SBC (.B) ⁽²⁾	dst	Subtract not(C) from destination	dst + 0FFFFh + C → dst	-	-
SETC ⁽²⁾		Set C	1 → C	-	-
SETN ⁽²⁾		Set N	1 → N	-	1
SETZ ⁽²⁾		Set Z	1 → Z	-	1
SUB (.B)	src,dst	Subtract source from destination	dst + .not.src + 1 → dst	-	-
SUBC (.B)	src,dst	Subtract source and not(C) from dst	dst + .not.src + C → dst	-	-
SWPB	dst	Swap bytes	-	-	-
SXT	dst	Extend sign	0	-	-
TST (.B) ⁽²⁾	dst	Test destination	dst + 0FFFFh + 1	0	-
XOR (.B)	src,dst	Exclusive OR source and destination	src xor dst → dst	-	-

⁽¹⁾ Emulated Instruction

Lección 29: Microcontrolador Texas y LaunchPad MSP430 Ejemplos de aplicación

La herramienta de programación y desarrollo el LaunchPad MSP430, se pueden generar proyectos interesantes, en la fase de aprendizaje es un potente herramienta para el aprendizaje autónomo por su bajo costo y portabilidad, facilitando el aprendizaje de la programación de microcontroladores de 16 bits y la

implementación de proyectos de desarrollo incluso para el proyecto de final del 40%.

Estructura de un programa

Un programa para MSP430 es usualmente editado en CCS v5 o IAR, en la edición debe comenzarse por encabezado invocando la librería del microcontrolador a utilizar, luego sigue la definición de la PILA, junto con la definición de constantes numérica que puedan ser necesarias dentro del programa. A continuación sigue la estructura “main” requerida por el IDE y con la que se inicia formalmente el programa.

```
;ENCABEZADO
;-----
#include "msp430g2231.h"
#define inicioSP 0x240 ; DEFINICION DE CONSTANTE PARA LA PILA

main ; NECESARIO PARA EL COMPILADOR IAR
;-----;DEFINICION DE CONSTANTES DEL PROGRAMA;
;-----ORG 0xF800 ; DIRECCION DE INICIO DEL PROGRAMA;
;-----RESET MOV.W #inicioSP,SP ; INICIALIZA LA PILA
MOV.W #WDTPW+WDTHOLD,&WDTCTL ; DETENER EL WATCHDOG
;-----PROGRAMA PRINCIPAL
; SE UBICAN LOS LLAMADOS
;-----SUBRUTINAS DE ATENCION A LLAMADOS
; SUBRUTINAS DE ATENCION A INTERRUPCIONES EN ORDEN DE PRIORIDAD
;-----DEFINICION DE TABLAS
;-----VECTORES DE INTERRUPCION Y RESET
;-----ORG 0xFFFFE ;VECTOR DE RESET MSP430
DW RESET
;-----; SE RELACIONAN TODOS LOS VECTORES NECESARIOS
;-----.
;-----.
;-----END ;FIN DEL PROGRAMA
```

El programa inicia con la dirección de inicio de la memoria de programa con la directiva “org”, después la declaración de la etiqueta de “RESET” que debe corresponder con el vector de reset al final del programa, después se inicia la PILA, después las instrucciones relacionadas con la activación y inactivación del Watchdog, para seguir con el programa producto del algoritmo aplicado a la solución, dentro de esta sección se aconseja dejar el programa lineal con los

llamados a subrutinas, después del programa “principal” colocar las subrutinas que corresponden con los llamados realizados o con la atención a interrupciones, luego las estructuras tipo “tabla” si es el caso, para seguir con la definición los vectores de interrupción y Reset, y finalmente la instrucción “END” de final de programa.

Subrutina de retardo

Las subrutinas de retardo son muy necesarias para generar lapsos de tiempo, para permitir la visualización de un indicador LED o LCD, comutar un relay, energizar un transistor, leer un sensor, esperar la grabación de una memoria, etc. un ejemplo de subrutina de retardo es:

```
;-----  
RETARDO; SUBRUTINA DE RETARDO  
;  
ETIQ3    MOV      #700,R14          ; GUARDAR EN R14 EL NUMERO 700  
        DEC      R14              ; R14 ES DECREMENTADO EN 1  
        JNZ      ETIQ3            ; SI R14 ES CERO CONTINUA A LA LINEA RET SINO SALTA A ETIQ3  
        RET      ; TERMINA LA SUBRUTINA Y REGRESA A DONDE FUE LLAMADO
```

Encendido y apagado de un LED por pulsador.

En este programa tiene por objetivo realizar la comutación de un LED en el puerto P1.0 al oprimir el botón en el puerto P1.3, utilizando para esto el LaunchPad MSP430 que tiene precisamente micro LEDs en el puerto P1.0 y P1.6 y pulsador en el puerto P1.3, lo que hace fácil realizar los primeros programas de entrenamiento sobre esta plataforma. El pseudocódigo, algoritmo y diagrama de conexión no se relaciona puesto que se supone que el lector ya ha adquirido cierta habilidad hasta este punto para deducir el pseudocódigo y algoritmo y el LaunchPad MSP430 tiene el soporte de hardware por tanto no es necesario hacer implementación en protoboard.

```
#include 138  
  
#define midram 0240h  
#define time 4000 /* Constante para el retardo*/  
  
main;  
;  
        ORG 0F800h  
;  
RESET    MOV.W #midram,SP           ; Configuración de la pila  
        MOV.w #WDTPW+WDTHOLD,&WDTCTL ; WDT apagado  
  
        MOV.B &CALBC1_1MHZ,&BCSCTL1 ; Calibración del  
        MOV.B &CALDCO_1MHZ,&DCOCTL  ; oscilador a 1MHz  
  
        CLR.B &P1SEL               ; P1 como e/s digital
```

¹³⁸ Aparicio O. <http://todomcu.scienceontheweb.net>

```

MOV.B #11110111b,&P1DIR ; Todo P1 como salida excepto P1.3
MOV.B #00001000B,&P1OUT ; Apagar P1 y resistencia Pull-Up en P1.3
BIS.B #BIT3,&P1REN ; Resistencia Pull-Up habilitada
BIC.B #BIT3,&P1IES ; Flanco de bajada en P1.3
BIC.B #BIT3,&P1IFG ; Borrado de la bandera P1.3
BIS.B #BIT3,&P1IE ; Interrupcion local permitida

MOV #GIE+CPUOFF,SR ; Interrupciones habilitadas y
NOP ; CPU apagado

;----- P1_ISR; Rutina de Servicio a la Interrupción -----
XOR.B #BIT0,&P1OUT ; Comuta el estado del led
BIC.B #BIT3,&P1IFG ; Borrar bandera por software para poder
RETI ; Salir de la interrupcion
; Instrucion para indicar que la rutina
; de servicio a la interrupcion termino

;----- Vectores de Interrupcion y Reset -----
ORG 0FFF Eh ; Reset
DW RESET ; Etiqueta para Reset
ORG 0FFE4h ; Vector para la int del P1
DW P1_ISR ; Etiqueta de la RSI
END main
    
```

Encendido de un LED por 10 segundos

Aprovechando nuevamente la tarjeta LauncPad MSP430 se hace una nueva práctica aprovechando los micro LEDs y pulsador de la tarjeta. En esta oportunidad el encendido de un LED por 10 segundos.

```

#include139

#define midram 0240h
#define time 60000
#define time1 55
main
;-----
;----- ORG 0F800h ; Direccion de inicio del programa ;-----
;-----
RESET MOV #midram,SP ; Inicializacion del SP
MOV #WDTPW+WDTHOLD,&WDTCTL ; WDT apagado

MOV.B &CALBC1_1MHZ,&BCSCTL1 ; Calibracion del DCO
MOV.B &CALDCO_1MHZ,&DCOCTL

CLR.B &P1SEL ; P1 como E/S digital
MOV.B #11110111B,&P1DIR ; P1.3 como entrada
MOV.B #00001000B,&P1OUT ; Resistencia Pull-Up
BIS.B #BIT3,&P1REN ; Resistencia habilitada
    
```

¹³⁹ Aparicio O. <http://todomcu.scienceontheweb.net>

```

        BIS.B #BIT3,&P1IES          ; Flanco de bajada
        BIC.B #BIT3,&P1IFG          ; Borrar la bandera de interrupcion
        BIS.B #BIT3,&P1IE          ; Interrupciones locales
    habilitadas

        MOV     #GIE+CPUOFF,SR      ; Modo de bajo consumo e
        NOP                  ; interrupciones globales
    habilitadas

;-----[P1_ISR]-----[Rutina de servicio a la interrupcion]
;-----[BIS.B #BIT0,&P1OUT]-----[Enciende el led]
;-----[CALL #RETARDO]-----[Retraso aproximado de 10 seg]
;-----[BIC.B #BIT0,&P1OUT]-----[Apaga el led]
;-----[BIC.B #BIT3,&P1IFG]-----[Borra la bandera de interrupción]
;-----[RETI]-----[]

;-----[Sub-rutina de Retardo]
;-----[RETARDO MOV #time,R4]-----[60000 a R4 para loop interno]
;-----[MOV #time1,R5]-----[55 a R5 para loop externo]
;-----[INT DEC R4]-----[R4--]
;-----[JNZ INT]-----[¿R4 es cero?]
;-----[DEC R5]-----[R5--]
;-----[JNZ INT]-----[¿R5 es cero?]
;-----[RET]-----[]

;-----[Vectores de Interrupcion y Reset]
;-----[ORG 0FFEh]-----[Vector para el reset]
;-----[DW RESET]-----[ ]
;-----[ORG 0FFE4h]-----[Vector para la interrupcion del]
;-----[P1 DW P1_ISR]-----[ ]
;-----[END main]-----[ ]

```

Secuencia de 8 LEDs

Para finalizar se presenta un programa que utiliza el LaunchPad MSP430, específicamente utiliza las terminales de salida directa del microcontrolador para conectar a la protoboard en la forma de un DIL, es decir, en la forma de un dispositivo de doble línea de pines.

```
#include "msp430x20x3.h"140

#define posc      R4      /* Variable para almacenar la posicion*/
#define limizq    128     /* Constante para el limite a la izquierda*/
#define limder    1       /* Constante para el limite a la derecha*/
#define midram   0240h    /* Constante para desirnar el lugar de la pila*/
#define time     40000    /* Constante para el retardo*/
main
;-----  

ORG 0F800h ; Direccion de inicio el la flash
;-----  

RESET mov.w #midram,SP ; Inicio del SP
        mov.w #WDTPW+WDTHOLD,&WDTCTL ; WDT apagado
        mov.b &CALBC1_1MHZ,&BCSCTL1 ; Calibracion del
        mov.b &CALDCO_1MHZ,&DCOCTL ; oscilador a 1MHz
        clr.b &P1SEL          ; P1 como i/o digital
        bis.b #0FFh,&P1DIR      ; Todo P1 como salida
        clr.b &P1OUT          ; P1 en estado bajo
        clr.b &P2SEL          ; P2.7 como puerto digital
        mov.b #01000000b,&P2DIR ; P2.7 como entrada
        bic.b #BIT7,&P2OUT      ; Resistencia PULL-DOWN
        bis.b #BIT7,&P2REN      ; P2.7 resistencias habilitadas
        bis.b #BIT7,&P2IES      ; Flanco de bajada en P2.7
        bic.b #BIT7,&P2IFG      ; Limpiar bandera de int
        bis.b #BIT7,&P2IE      ; Int habilitadas en P1.7
        mov #GIE+CPUOFF,SR      ; Modo de ahorro e interrupciones
        nop                      ; globales habilitadas
;-----  

P2_ISR;          Rutina de Servicio a la Interrupcion
;-----  

INICIO mov.b #limder,posc ; R4= 00000001 b
IZQ  mov.b posc,&P1OUT      ; R4 a P1
        rla.b posc            ; Corrimiento <-
        call #RETARDO          ; Retardo
        cmp.b #limizq,posc    ; Es 1000 0000 ?
        jne IZQ
DER   clrc          ; Limpieza C
        rrc.b posc            ; Corrimiento ->
        mov.b posc,&P1OUT      ; Corrimiento a P1
        call #RETARDO          ; Retardo
        cmp.b #0h,posc         ; Es 0000 0000 ?
        jne DER
        bic.b #BIT7,&P2IFG    ; Borrar la bandera P2.7
        reti                    ; Salir de la interrupcion
;-----  

;----- Subrutina Retardo
;-----  

RETARDO mov #time,R5      ; Constante time a R5
CONT dec R5           ; R5 = R5-1
        jnz CONT              ; Es cero?
        RET
;-----  

;----- Vectores de Interrupcion y Reset
;-----  

ORG 0FFEh          ; Reset
DW   RESET          ; Etiqueta para Reset
ORG 0FFE6h          ; Vector para la int del P2
DW   P2_ISR          ; Etiqueta de la RSI
END  main
```

¹⁴⁰ Aparicio O. <http://todomcu.scienceontheweb.net>

Lección 30: Modulos Microcontrolados Arduino y BASIC Stamp

Los microcontroladores Arduino y Basic Stamp, tienen la característica de ser sistemas embebidos en tarjetas que tienen como núcleo un microcontrolador, memoria con la decodificación de instrucciones de lenguaje de alto nivel, módulos de alimentación, módulo de comunicaciones para programación y simulación In-Circuit. En esta lección se profundiza un poco más en la programación y trabajo con Arduino y Basic Stamp.

Módulo Microcontrolado Arduino

Mucha de la información de Arduino el lector la puede acceder en español, en <http://arduino.cc/es>, los programas o “Sketch” se pueden acceder como parte de la descarga e instalación del paquete Arduino, se encuentran en el directorio “Examples”. Son varios los programas que pueden utilizarse para aprender el manejo de esta tarjeta, se encuentran en el mercado varias versiones de acceso gratuito.

Funcionamiento

Como primera instancia se debe conseguir la tarjeta o placa de desarrollo, descargar la versión de Arduino compatible con el sistema operativo y las instrucciones que corresponden al sistema operativo.

Comentarios: al igual que en otros entorno de desarrollo suele ser útil utilizar comentarios para documentar un programa. Todo lo que se coloque entre “/*” y “*/” el compilador lo toma como comentario, para comentarios de una sola línea se utiliza “//”.

Variables: las variables almacenan datos, por tanto deben tener un nombre, un tipo y un valor. En el ejemplo “ledPin” es el nombre de la variable, su tipo es “int” y tiene el valor inicial “13”.

Ejemplo: int ledPin=13 //LED conectado a pin digital 13

Funciones: también denominada subrutina, es un trozo de código con un propósito definido y que puede ser llamado a ejecución desde cualquier parte del programa principal. Una función está compuesta por el tipo de devolución (datos que entrega, puede ser vacío o void), luego sigue el nombre de la función, entre paréntesis los parámetros que recibe aunque no necesariamente puede ser vacía “()”, luego siguen las llaves “{ }”, el código propio de la función.

Ejemplo:

```
void setup ()           // Función vacía "setup" no devuelve parámetros
{
    pinMode (ledPin, OUTPUT); // Ajusta el pin digital como salida
    /* pinMode () es una función que requiere dos parámetros ledPin y OUTPUT, se utilizan por
    la función para determinar que pin y modo utilizar*/
}
```

La función “pinMode()” es utilizada para configurar un pin como entrada o salida, requiere dos parámetros el número del pin a configurar y la constante INPUT (entrada) o OUTPUT (salida).

La función “digitalWrite()” se encarga de enviar un valor a un pin, requiere dos parámetros, el pin a configurar y el estado del pin HIGH (alto o 5V) o LOW (bajo o 0V).

Ejemplo: digitalWrite (ledPin, HIGH); // Pin 13 en estado alto

La función “delay ()” se encarga de generar un retardo en milisegundos, entre paréntesis se coloca la cantidad de milisegundos.

Ejemplo: delay (1000); // Retardo de 1 segundo

La función “setup()” es llamada una sola vez cuando comienza el sketch o programa. En ella usualmente se configuran los pines, inicializa variables o bibliotecas, etc.

La función “loop ()” puede ser llamada una y otra vez, dentro del sketch o programa para generar ciclos repetitivos.

Programación en Arduino

Como ejercicio práctico se analiza la programación en Arduino que permite encender y apagar un LED de manera intermitente, este es el clásico programa de iniciación en un microcontrolador similar al “Hola mundo” utilizado en programación de software. Se debe proceder a instalar Arduino IDE, ejecutarlo y en el editor de código comenzar.

Encender y apagar un LED: Todo programa en Arduino consta de dos funciones básicas, “setup ()” y “loop ()”, en el programa se decide utilizar el pin 13 debido a que incorpora una resistencia o driver que permite conectar el LED directamente sin sobrecargar el micro.

```
void setup () {           // Función setup()
    pinMode (13, OUTPUT); // Configura pin 13 como salida
}
void loop (){
    digitalWrite (13, HIGH); // Se coloca el pin 13 en estado alto o 5V
    delay (1000);          // Se genera un retardo de 1 segundo
    digitalWrite (13, LOW); // Se coloca el pin 13 en estado bajo o 0V
    delay (1000);          // Se genera un retardo de 1 segundo
}
```

Detector de movimiento con sensor infrarrojo: un sensor infrarrojo comercial está diseñado para generar un pulso bajo o alto en respuesta a detección de un objeto que interfiere con una señal infrarroja emitida por el mismo sensor. Los sensores infrarrojos también denominados sensores PIR son muy utilizados para implementar sistemas de alarma. En el presente programa se utiliza un sensor PIR como detector de movimiento y que envía un pulso al pin 2 de una tarjeta Arduino para generar una señal como respuesta que consiste en un LED conectado al pin 13 que detecta una señal por el pin 2 se activara por espacio de 3 segundos, este es solo un ejercicio de aplicación, pero que sirve como base para una aplicación más compleja.

```
int      ledPin = 13    // Define la variable de nombre "ledPin" tipo "int" con un valor de 13
int      pir = 2        // Define la variable de nombre "pir" tipo "int" con un valor de 2
int      pirStado = 0   /* Define la variable de nombre "pirStado" de tipo "int" con un valor de 0
                        como estado inicial del sensor PIR*/
void setup () {          // Función setup()
    pinMode (ledPin, OUTPUT); // Configura pin 13 como salida
    pinMode (pir, INPUT);   // Configura pin 2 como entrada
}
void loop (){
    pirStado = digitalRead (pir); // la variable "pirStado" guarda el estado del pin 2 "pir"
    if (HIGH == pirStado) {      // Si la variable "pirStado" es alto o 1 lógico entonces
        digitalWrite (ledPin, HIGH) // pin 13 "ledPin" se coloca en alto, enciende LED
        delay (3000);            // Retardo de 3 segundos para mantener encendido el LED
    }
    else {                     // Si la variable "pirStado" no está en alto o 1 lógico
        digitalWrite (ledPin, LOW) // Apagamos el LED
    }
}
```

Módulo Microcontrolado BASIC Stamp

Como, se ha establecido el microcontrolador Basic Stamp es en realidad un híbrido conformado por un núcleo inicialmente PIC y una serie de módulos, para comunicación con el PC para su programación, módulos de conversión analógica / digital (A/D), módulo de fuente regulada de voltaje y chip de memoria en la cual se almacena el programa, la característica más sobresaliente es que posee un

intérprete de comandos similar al BASIC denominado PBASIC. Como primer paso se debe instalar el editor BASIC Stamp (versión 2.5.3), tener un módulo o tarjeta BASIC Stamp II, al ejecutar el programa editor BASIC Stamp, se tiene un editor sencillo el cual tiene un poderoso archivo de ayuda (menú Help), con el que se puede tener acceso a documentación, igualmente en la página del fabricante www.parallax.com. El set de instrucciones de PBASIC se compone de aproximadamente 36 instrucciones, entre ellas están:

COUNT: Permite contar el número de ciclos, que llegan por un PIN específico durante un periodo definido de tiempo en milisegundos, el valor del conteo se guarda en una variable.

DEBUG: Facilita el monitoreo el programa en ejecución en cualquier punto, visualizando la variable o mensaje en la pantalla de un PC conectado al BASIC Stamp II.

DTMFOUT: Genera tonos de frecuencia DTMF por un pin específico.

FOR - NEXT: Se implementa un ciclo del tipo FOR – NEXT similar al ciclo FOR en otros lenguajes como C/C++.

FREQOUT: Genera uno o dos tonos con ondas del tipo sinodal durante un tiempo y PIN determinado.

PULSOUT: Se genera un pulso por uno de los pines del puerto durante un periodo de tiempo entre 2 milisegundos y 131 milisegundos.

PWM: Instrucción utilizada para generar una señal PWM en cualquiera de los pines I/O.

RCTIME: Cuenta el tiempo durante un determinado PIN se encuentra en un estado determinado. Utilizado generalmente para determinar tiempos de carga y descarga en redes RC.

SERIN: Esta instrucción habilita un PIN del puerto como entrada serial de datos no sincrónicos, compatibles con RS-232.

SEROUT: Esta instrucción es el complemento de la anterior, habilita un PIC del puerto como salida serial de datos no sincrónicos, compatibles con RS-232.

SHIFTIN: La instrucción permite recibir datos en formato serie y con desplazamiento sincrónico.

SHIFTOUT: Esta instrucción permite enviar datos en formato serie con desplazamiento sincrónico.

STOP: Instrucción encargada de detener la ejecución del programa, sin reducir el consumo de energía.

WRITE: Instrucción que permite escribir directamente en la memoria EEPROM de datos.

XOUT: Esta instrucción permite enviar hasta 10 comandos a través de un PIN del puerto.

Programación con BASIC Stamp

Como se ha establecido en anteriores prácticas se pretende plantear ejercicios que muestren el funcionamiento de las instrucciones y programas en los diferentes microcontroladores y módulos microcontrolados. Teniendo el software instalado, la tarjeta BASIC Stamp II conectada al PC podemos hacer la programación, es de aclarar que el lector debe haber implementado los circuitos en protoboard puesto que el módulo BASIC Stamp II se presenta en formato DIL (Doble línea de pines) como si fuera un circuito integrado “gigante” y no trae sócalos para conectar directamente los periféricos como sucede en el Arduino.

Los comentarios se generan después de una comilla simple “ ‘ ”, la directiva DEBUG “mensaje” hace que el mensaje aparezca en la Terminal Debug, es utilizado para indicar la función del programa que se realiza.

Encender y apagar un LED con BASIC Stamp II: Es necesario realizar en la protoboard los circuitos requeridos para actuar sobre el LED, es decir, una resistencia en serie con el LED y disponerse a conectar el LED en su Cátodo al terminal negativo de alimentación negativa y el Ánodo en serie con la resistencia y la otra terminal de la resistencia al pin correspondiente del BASIC Stamp II.

‘ Programa que enciende y apaga un LED por el pin 14

DEBUG “ Encendido y Apagado de un LED por PIN 14”

DO

HIGH	14
PAUSE 500	
LOW	14
PAUSE 500	

‘ Coloca en alto o a 5 V el Pin 14
‘ Causa una pausa de 500 milisegundos
‘ Coloca en bajo o 0 V el Pin 14
‘ Causa una pausa de 500 milisegundos

LOOP

Encendido y apagado de un LED controlado por un botón: En el siguiente programa se utiliza un botón o pulsador para controlar el encendido o apagado de un LED. El botón se censa por el terminal 3 y según el estado en que este enciende un LED conectado al Pin o terminal 14.

' Programa que enciende y apaga un LED por el pin 14 controlador por un pulsador en el Pin 3

DO

```
DEBUG ?      IN3      ' Censa el estado de PIN 3 y envía el estao a la terminal Debug
IF (IN3 = 1) THEN
    HIGH     14      ' Coloca en alto o a 5 V el Pin 14
    PAUSE   50      ' Causa una pausa de 50 milisegundos
    LOW     14      ' Coloca en bajo o 0 V el Pin 14
    PAUSE   50      ' Causa una pausa de 50 milisegundos
ELSE
    PAUSE   100     ' Causa una pausa de 100 milisegundos
ENDIF
```

LOOP

Actividades de Autoevaluación de la UNIDAD

1. Investigue en textos e internet las principales familias de microcontroladores y genere puntos de debate con sus compañeros acerca de la variedad de productos y los últimos modelos.
2. Determine cuáles son las unidades fundamentales comunes a un microcontrolador y microprocesador y genere puntos de comparación para justificarlos al interior de su grupo de trabajo.
3. Con respecto a los diversos componentes utilizados como dispositivos de entrada, investigue la información particular del dispositivo que puede estar en hojas de características (datasheet), voltajes de alimentación, señal de entrada, señal de salida, rango de voltajes o corrientes, comparta con su grupo de trabajo la información referida a los siguientes componentes:
 - a. Interruptores.
 - b. Pulsadores.
 - c. Teclados
 - d. Sensores de temperatura, humedad, ph, ópticos y de movimiento.
4. Con respecto a los diversos componentes utilizados como dispositivos de salida y/o actuadores, investigue la información particular del dispositivo que puede estar en hojas de características (datasheet), voltajes de alimentación, señal de entrada, señal de salida, rango de voltajes o corrientes, comparta con su grupo de trabajo la información referida a los siguientes componentes:
 - a. LEDs.
 - b. Display siete segmentos.
 - c. Display de Cristal líquido, LCD.
 - d. Relé o relay
 - e. Bocinas o parlantes.
 - f. Motores AC, DC, paso a paso y Brushless
5. Investigue la simbología electrónica, funcionamiento nomenclatura y montaje de los componentes más usuales en los montajes de sistemas basados en microcontroladores, más precisamente, resistencias, capacitores (condensadores), bobinas, transformadores, diodos de unión, diodos zener, puentes rectificadores, LEDs, Resonador de Cristal, transistores BJT (NPN,PNP), transistores de efecto de campo FET, tiristores SCR, DIAC, TRIAC. Comparta con sus compañeros de grupo comentarios, inquietudes y documentación al respecto. Recuerde que los tutores en el área de electrónica pueden ayudarlos en este aspecto.

6. Realice una investigación acerca de las técnicas de construcción de circuitos impresos y con su grupo de trabajo realizar la implementación en protoboard de circuitos simples como:
 - a. Circuito de resistencias en paralelo, serie y mixto.
 - b. Montaje de circuitos simples con el temporizador CI555, en conjunto con resistencias, capacitores, pulsadores, switch, LEDs y fuente de alimentación.
7. Tomando un PIC16F84 o un Motorola Freescale HC08, determine con ayuda de este modulo, la hoja de especificaciones, textos especializados e internet, la arquitectura interna del microcontrolador.
8. Instale los programas de simulación y desarrollo para el PIC o Freescale e ingrese a la página oficial de “proteus” para acceder a una versión gratuita.
 - a. PIC16F84: MPLAB, PICDevelopment Studio, ICPROG.
 - b. HC08: WINIDE,
 - c. Proteus: <http://www.ieeproteus.com/descarga.html>
9. Aprenda el set o juego de instrucciones de los microcontroladores PIC y Motorola Freescale, practique la ejecución individual en papel (prueba de escritorio) de cada instrucción para comprender su funcionamiento, genere puntos de debate y discusión al respecto con sus compañeros y tutor.
10. Digite los programas en ensamblador, compílelos e implemente los montajes propuestos y ejecútelo, observe el comportamiento, comparta la experiencia con el grupo y tutor.

Fuentes Documentales de la Unidad 2

DOCUMENTOS IMPRESOS

Angulo, Usategui José María. (n. d). Microcontroladores PIC. Diseño práctico de aplicaciones.

Angulo. (n.d). Microcontroladores PIC, la solución en un chip. Sección 5.1

Vesga, Ferreira Juan Carlos. (2007). Microcontroladores Motorola – Freescale: Programación, familias y sus distintas aplicaciones en la industria.

González, Vásquez José Adolfo. (1992). Introducción a los microcontroladores: hardware, software y aplicaciones. Editorial McGraw-Hill.

Dorf, C. Richard. (1997). "Circuitos Eléctricos. Introducción al análisis y diseño". (2^a edición). Editorial AlfaOmega S.A. Santafé de Bogotá.

Savant. J, Roden. S. Martin & Carpenter. L. Gordon. (1992). "Diseño Electrónico. Circuitos y sistemas". (2^a edición). Editorial Addison-Wesley Iberoamericana. E.U.A.

CEKIT. (2002). Curso Práctico de Microcontroladores: Teoría, Programación, Diseño, Prácticas Proyectos completos. Editorial Cekit. Pereira-Colombia.

Stallings, William. "Organización y Arquitectura de Computadores". (5^a edición). Editorial Prentice-Hall. Madrid, 2000.

Téllez, Acuña Freddy Reynaldo. (2007). Módulo de Microprocesadores y Microcontroladores. UNAD.

DIRECCIONES DE SITIOS WEB

Herrera. R. Lucelly. (n.d.). Microcontroladores. Sistemas WinIDE. Extraído el 29 de Julio desde.

http://fisica.udea.edu.co/~lab-gicm/Curso_Microcontroladores/Micros_2012_WIN_IDE.pdf

Aparicio. O. H. (n.d). Todomcu. Extraído el 20 de Junio de 2013 desde.

<http://todomcu.scienceontheweb.net>

Teoría de computadores. Extraído el 10 de Julio de 2009 desde

<http://www.computacion.geozona.net/teoria.html>

Dispositivos lógicos microprogramables, extraído el 10 de Julio de 2009 desde

<http://perso.wanadoo.es/pictob/indicemicroprg.htm>

Curso de Microcontroladores Motorola, extraído el 10 de Julio de 2009 desde

http://www.geocities.com/moto_hc08/index.html

UNIDAD 3 PROGRAMACION Y DESARROLLO DE PROYECTOS CON MICROPROCESADORES Y MICROCONTROLADORES

CAPITULO 7: DISEÑO Y DESARROLLO DE PROYECTOS CON MICROCONTROLADORES Y MICROPROCESADORES

Introducción

En este capítulo se contextualiza conceptos relevantes para el desarrollo del hardware de proyectos. Por tanto se presentan generalidades fundamentales en electrónica y circuitos, junto con los periféricos para microcontroladores más utilizados y se presentan una serie de parámetros generales con respecto al desarrollo de una solución de hardware y software.

Lección 31: Principios fundamentales en electrónica y circuitos

Para poder implementar adecuadamente un proyecto con microcontroladores es necesario tener conocimientos básicos en electrónica análoga y digital, en montajes electrónicos en protoboard y en el manejo de instrumentos de medición, para este capítulo se hará una presentación general de la electrónica involucrada en los proyectos de desarrollo.

Definiciones básicas

La electrónica básica aplicada requiere tener conocimiento y diferenciar claramente los conceptos fundamentales:

Carga eléctrica: es la cantidad de electrones responsable de los fenómenos eléctricos esta tiene una unidad denominada Coulomb (C). Se establece que la carga eléctrica en un protón es igual a la de un electrón, solo que la del electrón es de signo negativo, este valor es igual 1.6021×10^{-19} C para un protón y se conoce como carga elemental (q).

Corriente eléctrica: es entonces la razón o cambio del flujo de carga eléctrica por unidad de tiempo, que pasa por un punto dado. Entonces la corriente se expresaría como la cantidad de cargas ' q ' que pasan por un área determinada en la unidad de tiempo, su unidad de medida es el Amperio, en honor a Andre-Marie Ampere y su símbolo es (I).

Voltaje eléctrico: El voltaje a través de un elemento es el trabajo necesario para mover una carga positiva de 1 coulomb de la primera a la segunda terminal a través del elemento. La unidad de voltaje es el volt, "V". Entonces hay una caída de voltaje si una carga positiva entrega energía al elemento cuando se desplaza.

Potencia eléctrica: se mide como la razón de transformación de energía, (en física se define como la cantidad de trabajo en la unidad de tiempo) se simboliza con una "P", además la energía es la capacidad de realizar trabajo, su símbolo es "W" vatio.

Instrumentos de medida

Los instrumentos de medida son utilizados para testear medidas en los componentes electrónicos involucrados en el montaje, recuerde que los microcontroladores tienen un rango de voltaje para su funcionamiento:

Voltímetros y amperímetros: las mediciones de corriente y voltaje se efectúan con medidores de lectura directa (analógicos) o digitales, en los medidores de lectura directa tienen una aguja indicadora cuya deflexión angular depende de la magnitud de la variable que mide. Un medidor digital muestra una serie de dígitos que indican el valor de la variable medida.

Amperímetro: un amperímetro ideal mide la corriente que fluye por un elemento el modo de conexión es en serie con el elemento, además idealmente tiene caída de voltaje cero entre sus terminales, aunque los instrumentos prácticos presenten

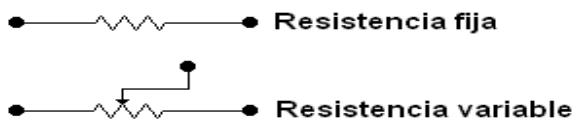
caídas de tensión insignificantes comparadas con las tensiones presentes en un circuito eléctrico. Un amperímetro se coloca entre el elemento y la línea de alimentación eléctrica, es decir la corriente circula por el amperímetro.

Voltímetro: un voltímetro ideal mide el voltaje entre sus terminales y tiene en sus terminales una corriente igual a cero, el voltímetro práctico presenta corriente en sus terminales, pero también se hace muchas veces insignificantes en comparación con las demás corrientes existentes en el circuito o con la corriente que este circulando por el elemento que este siendo medido. La manera como debe conectarse un voltímetro es en paralelo con el elemento, el voltímetro se coloca entre los terminales del elemento que se esta midiendo. Los componentes electrónicos se dividen en activos y pasivos

Elementos pasivos

Son los elementos que absorben energía este es el caso de los siguientes componentes:

Figura 136. Resistencias fijas y código de colores



Código de Cores de Resistências									
4 Bandas							222, 5%		
5 Bandas							2200Ω, 1%		
6 Bandas							100KΩ, 0.1%, 15ppm		
Prata	0	0	0	1	0.01	10%			
Ouro					0.1	5%			
Preto 0	0	0	0	1					
Castanho 1	1	1	1	10	1%	100ppm			
Vermelho 2	2	2	2	100	2%	50ppm			
Laranja 3	3	3	3	1K		15ppm			
Amarelo 4	4	4	4	10K		25ppm			
Verde 5	5	5	5	100K	0.5%				
Azul 6	6	6	6	1M	0.25%				
Púrpura 7	7	7	7	10M	0.1%				
Cinzento 8	8	8	8				Multiplicador		
Branco 9	9	9	9					Tolerância	Coefficiente de Temperatura

Resistencias: Es la propiedad física de un elemento o dispositivo que impide el flujo de corriente; se representa así: R. La unidad de resistencia se expresa en ohm (Ω), generalmente los valores de miles o millones de resistencia se expresan en kiloOhms ($K\Omega$) o MegaOhms ($M\Omega$) respectivamente. Los valores nominales de los resistores en operación están generalmente basados en $25^{\circ}C$

Condensadores: son elementos conformados por placas de conductoras en paralelo separadas por un material aislante, su unidad de medida son los Faradios, generalmente expresados en microfaradios (μF), nanofaradios (nF) o picofaradios (pF), se simboliza con una "C"

Inductancias: están fabricadas por vueltas de alambre aislado enrollado en núcleos vacíos o de material ferroso, su unidad es el Henry (H), se simboliza por "L".

Elementos Activos

Un elemento activo, es activo si es capaz de entregar energía, tal es el caso de las fuentes de alimentación.

Corriente Continua a C.C: compuesta por pilas, baterías y adaptadores que convierten la corriente A.C en C.C.

Corriente Alterna A.C: compuesta por la energía eléctrica proveniente de la toma común en la alimentación doméstica.

Elementos Semiconductores

Dentro de esta familia está la gama de diodos, transistores y dispositivos de cuatro capas.

Diodos: son dispositivos semiconductores formados por una capa tipo N con exceso de electrones con carga eléctrica negativa, llamada "Cátodo" y unida con una capa tipo P con exceso de huecos con carga eléctrica positiva llamada "Ánodo". Los diodos dejan circular corriente eléctrica en un solo sentido cuando el "Ánodo" se conecta al terminal más positivo y el "Cátodo" al más negativo. Dentro de las familias de diodos se encuentran:

- Diodos de unión utilizados para rectificación y puentes rectificadores.
- Diodos zener como referencias de voltaje para regulares eléctricos.
- LEDs los cuales pueden estar de forma individual y emitir un haz de luz o en grupo como matrices o display de siete segmentos.

Transistores: son dispositivos de tres capas de los cuales se distinguen los siguientes:

- **BJT**, compuesto por dos capas de tipo N entre una tipo P, creando el transistor NPN y también dos capas tipo P entre una tipo N, creando el transistor PNP, estos transistores tienen tres terminales.
 - La Base: por la cual llega la señal de poca intensidad, debe tener la misma polaridad del tipo de capa.
 - El Colector: se conecta a la terminal opuesta al tipo de capa en ella se conecta la carga a manejar que requiere mayor potencia para su activación.
 - El Emisor: se conecta al terminal que coincide con la polaridad del tipo de capa, este es el que recoge el flujo de corriente.
- **FET**, son transistores de efecto de campo, los hay de canal N y canal P, funcionan de forma similar a los BJT solo que el consumo de potencia es menor aunque son más delicados por la tecnología CMOS con la que se fabrican

Dispositivos de cuatro capas: estos dispositivos se construyen con cuatro capas alternas de material tipo N y P con lo que se obtienen tres dispositivos muy conocidos:

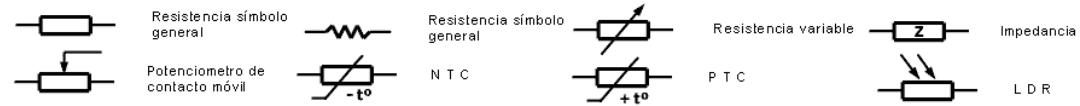
- SCR, o tiristor utilizado para controlar cargas alimentadas con Corriente Alterna o Corriente Continua.
- TRIAC es un dispositivo que permite el control de cargas alimentadas por C.A.
- DIAC utilizado como dispositivo de disparo para los SRC o TRIACs este dispositivo funciona en C.A o C.C y permiten la conducción cuando supera un voltaje de disparo.

Símbolos eléctricos, electrónicos y digitales más usuales en la implementación de proyectos:

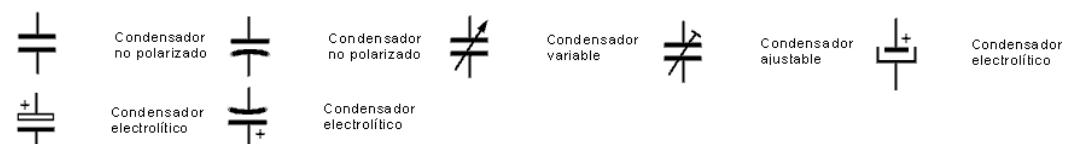
En las figuras que se presentan en las páginas siguientes, se encuentran la mayoría de símbolos utilizados en los esquemas electrónicos, simuladores y diseños. Estos símbolos están relacionados con conceptos, funciones y componentes reales que hacen parte de la mayoría de montajes prácticos.

Figura 137. Símbolos de resistencias, Condensadores, bobinas Diodos e interruptores

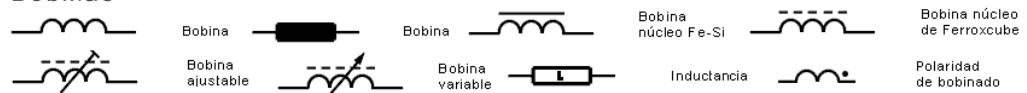
Resistencias



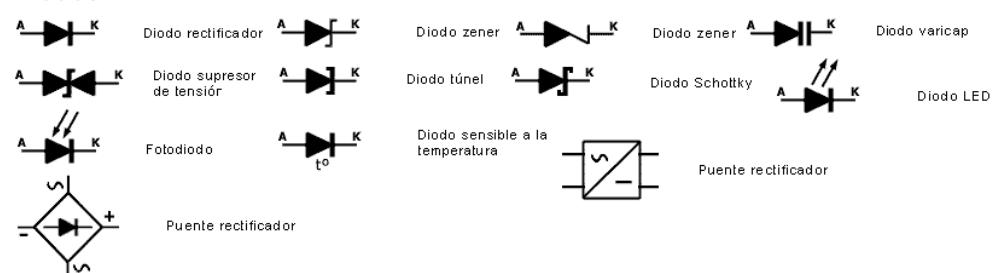
Condensadores



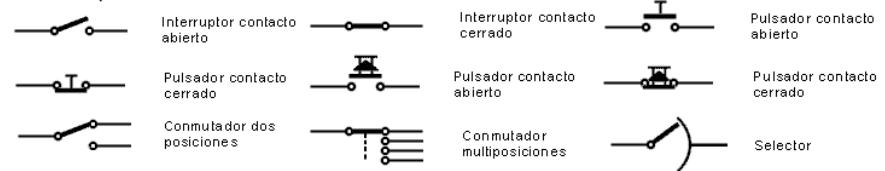
Bobinas



Diodos



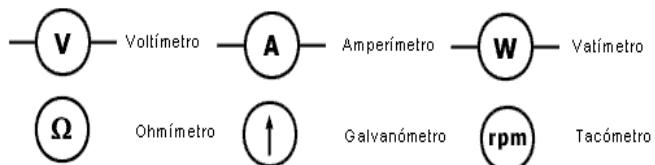
Interruptores



Copyright © JJRC.

Figura 138. Símbolos de instrumentación, dispositivos activos, transistores, tiristores y electrónica Digital

Instrumentación



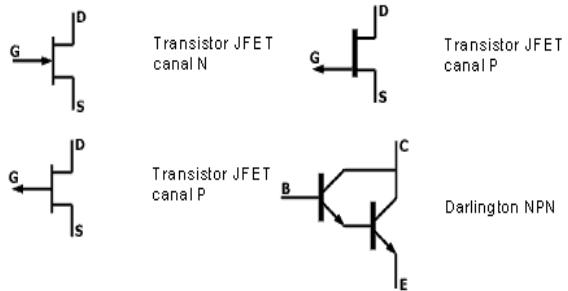
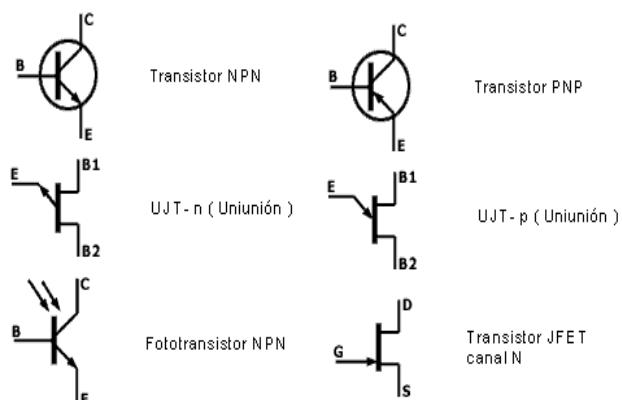
Dispositivos Activos



Indicador de batería



Transistores



Tiristores



Digital

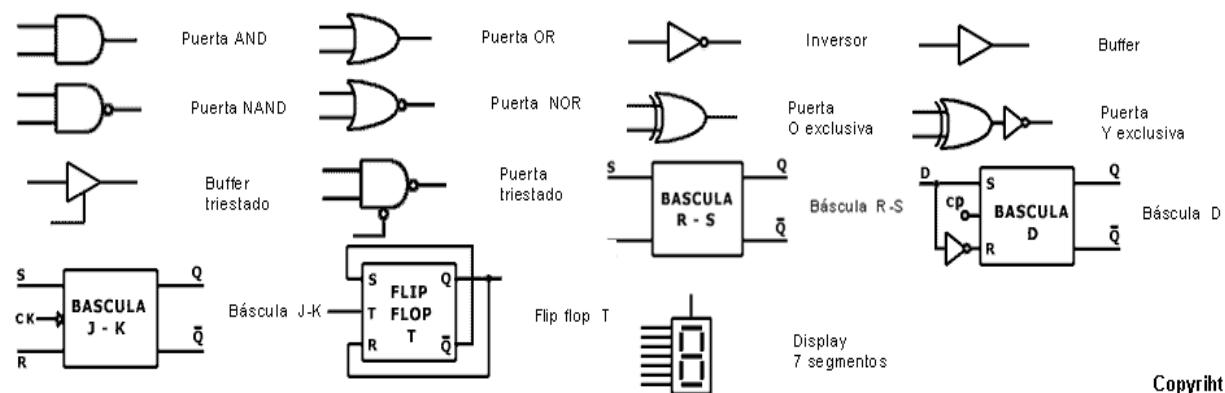
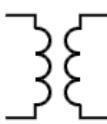
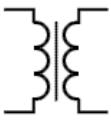


Figura 139. Símbolos de transformadores y otros de uso común.

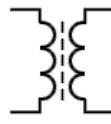
Transformadores



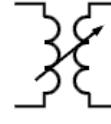
Transformador núcleo aire



Transformador núcleo de Fe-Si



Transformador núcleo Ferroxcube



Transformador acoplamiento variable



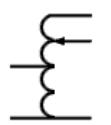
Transformador apantallado



Polaridad de bobinado

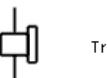


Transformador variable

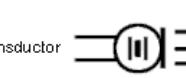


Autotransformador variable

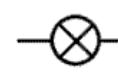
Varios



Transductor



Transductor



Lámpara incandescente



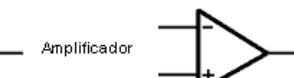
Lámpara incandescente



Motor



Amplificador



Amplificador operacional



Cristal piezoelectrónico



Cristal piezoelectrónico



Sensor al tacto



Célula fotovoltaica



Mando electromagnético

Copyright © JJRC

Lección 32: Periféricos para microprocesadores y microcontroladores

Los periféricos externos al microprocesador o microcontrolador son tan importantes como el mismo micro, de los periféricos depende la capacidad, flexibilidad y robustez del sistema digital basado en microprocesadores o microcontroladores. En esta lección se trata precisamente los periféricos externos, desde los más usuales como los pulsadores o switch hasta llegar a los más complejos como pantallas LCD o motores. Se hace una breve presentación y descripción de cada uno resaltando sus características y función dentro de un sistema digital.

Periféricos externos al microprocesador y microcontrolador

Un periférico externo se define como un dispositivo o conjunto de dispositivos en la forma de circuito eléctrico, con una función definida y que se diseña para ser conectado y compatible con las señales de entrada/salida del micro. Un periférico puede ser tan simple como un pulsador o una resistencia o tan complejo como otro microcontrolador que se encarga de controlar un LCD. Algunas de las funciones definidas que tiene un periférico externo es el amplificar una señal digital

(0V-5V) a un voltaje o corriente adecuado, convertir la señal digital a otro tipo de señal incluso análoga, convertir una señal externa a forma digital en los voltajes requeridos (0V-5V), transformar una señal de voltaje digital o analógica en un acción física como movimiento, luz, calor, etc.

Periféricos de entrada/salida de datos para el microprocesador y microcontrolador

Entre estos periféricos se encuentras las memorias tipo RAM de acceso serial o paralelo, utilizadas en los microprocesadores o microcontroladores que sean sistemas abiertos, es decir, que permitan tener acceso a sus buses internos de direcciones, datos y control, para implementar la memoria RAM que amplié la capacidad de memoria del sistema en el caso del microcontrolador o represente la memoria de datos en un sistema con microprocesadores. Igual es el caso de la memoria ROM para ambos sistemas basados en microprocesadores o microcontroladores, que puede funcionar como la memoria principal de programa en el sistema basado con microprocesador o ser una ampliación de memoria en el caso del sistema basado en microcontrolador.

En sistemas cerrados como la mayoría de microcontroladores que no permiten acceso a sus buses internos las memoria de tipo ROM se utilizan como espacio de almacenamiento de valores iniciales o parámetros de arranque, sistema operativo similar a una BIOS o que contenga una firma digital o firework. Con respecto a la memoria RAM, se utiliza como espacio de almacenamiento auxiliar para datos obtenidos durante el procesamiento normal del sistema, de manera que representa un espacio auxiliar de almacenamiento volátil temporal.

Periféricos de entrada de datos para el microprocesador y microcontrolador

Los periféricos de entrada de datos para microprocesadores y microcontroladores, representan una gran cantidad de dispositivos utilizados para enviar datos en la forma de cambios de nivel o flanco entre valores digitales, con valores estables generalmente 0V a 5V, como dispositivos que entregan señales analógica entre valores determinados por voltajes de referencia en el micro, generalmente entre 0V hasta 5V. Estos periféricos pueden representar una acción física como la pulsación o acción mecánica sobre un de un switch o pulsador, representar una variable física como temperatura, luz, humedad, PH, etc, una señal de control desde otro sistema digital, etc. Es todo aquel dispositivo o circuito envié una señal adecuada al microprocesador o microcontrolador, para establecer un estado y que sirva como parámetro base para el control.

Pulsadores: Son los dispositivos más usuales, se componen de un sistema mecánico que al accionarse, ejerciendo una fuerza sobre una pieza móvil, hace contacto internamente dos placas para permitir la conducción eléctrica, estado lógico alto (1), al soltarse la pieza móvil las placas se separan suspendiendo la circulación eléctrica, estado lógico bajo (0). Este tipo de pulsadores se encuentran en tamaños pequeños como los utilizados en celulares y Ipad, hasta pulsadores de gran tamaño como los utilizados en máquinas de video juego o de casino.

Switch o interruptores: Su funcionamiento es similar al pulsador, es un sistema mecánico que al accionarse, ejerciendo una fuerza sobre una pieza móvil, hace contacto internamente dos placas para permitir la conducción eléctrica, estado lógico alto (1), al soltarse la pieza móvil, las placas internas permanecen en estado de conducción, hasta que se hace una acción de movimiento contrario sobre la pieza móvil, para que las placas se separan suspendiendo la circulación eléctrica, estado lógico bajo (0). Existe gran variedad desde los más usuales como los de codillo, hasta los utilizados como sensores, como el fin de curso o fin de carrera. Existen de varios estados, que pueden generar salidas en varias posiciones, pero siempre son ON/OFF.

Dip switch: DIP se refiere al tipo de presentación física, es decir, se presenta como un encapsulado de doble línea de pines, capa par opuesto de pines está relacionado con un micro interruptor o micro switch, es muy utilizado en circuito impreso para establecer valores o parámetros iniciales de configuración, hay presentacionde de 2 hasta 20 switch en un mismo empaque DIP.

Sensores – magnéticos, ópticos, acústicos: Dentro de esta gama se encuentran sensores magnético como los switch relay, que consisten en una ampolla de vidrio con un gas inerte y en la cual hay dos láminas separadas, las cuales hacen contacto en presencia de un campo magnético. Existe otros dispositivos integrados que se accionan por la presencia de campos magnéticos, son utilizados como sensores de presencia en apertura de puertas, sensores de nivel, sensores para cuenta vueltas, etc. En el campo se los sensores ópticos, se encuentras de manera individual emisores, como diodos de luz visible o infrarroja o la misma luz ambiente y receptores como fotodiódos, fototransistores o photoceldas, los cuales se utilizan como detectores de presencia, cuenta vueltas, detectores de pasaje, etc. En el mercado se encuentran dispositivos integrado en parejas para realizar la emisión y recepción en un mismo encapsulado, o para permitir la detección y amplificación de la señal recibida cuando se requiere de una fuente distante o débil.

Transistores BJT y FET: Estos dispositivos son utilizados para adecuar la señal proveniente de fuentes débiles, para convertir señales de corriente o voltaje a voltajes estables de 0V o 5V, o en niveles adecuados entre 0V hasta 5V. Son utilizados como elementos de amplificación y adecuación de señal en circuitos encargados de convertir una variable física en una señal digital.

Comparadores: Aunque muchos microcontroladores implementan módulos comparadores, se utilizan en ocasiones Amplificadores Operacionales, configurados como comparadores para determinar el estado de una señal desconocida de voltaje en relación con una señal de referencia conocida.

ADC / DAC: Este periférico es otro que se encuentra en muchos microcontroladores como módulo interno, el ADC se encarga de convertir una señal analógica a digital, con una precisión de 8 a 16 bits, estos pueden ser paralelos utilizando entre 8 y 16 pines o seriales para lo cual utilizan generalmente una línea de señal como reloj, otra para transmitir la trama de datos, un pin de control y claro las líneas de alimentación de Gnd y 5V. Los conversores digitales análogos se encargan de hacer la función opuesta, teniendo una trama de bits se convierten a una señal de voltaje.

Memorias (paralelas/seriales): Estas memorias como se mencionó anteriormente, permiten ampliar la capacidad de memoria de los sistemas basados en microprocesadores o microcontroladores, se utilizan para almacenar parámetros de inicio o arranque, estados iniciales o firework en memorias ROM y para almacenar datos temporales en el caso de memorias RAM. El término paralelas se refiere a la manera de intercambiar información con el micro, de manera paralela utiliza gran cantidad de bits, según el formato del registro, generalmente es de 8 bits, en el caso de las seriales utiliza unos pocos pines para el control de la transmisión de datos (reloj), el envío de datos de forma serial (data) y los pines de control de lectura/escritura.

Periféricos de salida de datos para el microprocesador y microcontrolador

Los periféricos de salida de datos representan otra gama extensa de dispositivos, que pueden ser un componente o un circuito completo funcional. En este caso se destacan los dispositivos digitales ON/OFF, puesto que generalmente las salidas de un microprocesador o microcontrolador son del tipo digital y unas pocas son analógicas, esto cuando el micro incorpora módulos DAC.

LED: Los LED son los dispositivos más ampliamente utilizados en el aprendizaje y en general en las implementaciones de control, como indicadores de estado, junto

con el pulsador conformar el segundo circuito de aprendizaje en la programación de cualquier microcontrolador o módulo microcontrolado, recordemos que el primer programa generalmente es encender y apagar un LED, que es la analogía al programa “Hola mundo” en el desarrollo de software. Para su funcionamiento requiere de una resistencia conectada al LED para limitar la corriente, pudiéndose conectar directamente a los puertos del micro, para mayor información remitirse al datasheet del micro para ver la corriente máxima que soporta en los pines I/O.

Pilotos: Al igual que los LEDs funcionan como indicadores visuales de estado, los pilotos son pequeñas lámparas de filamento o lámparas de neón, para su funcionamiento requieren de un circuito amplificador de corriente o voltaje, por lo que no se puede conectar directamente al micro.

Pantallas LCD: Las pantallas LCD son los dispositivos más utilizados como interfaz usuario, para indicar estados y visualizar adecuadamente mensajes y valores de control. Básicamente es una pantalla de cristal líquido controlada por un microcontrolador, con unas instrucciones estandarizadas, pines de datos y control es fácilmente controlada por cualquier micro.

Bocinas o zumbadores: Las bocinas son dispositivos sónicos, que producen una señal audible, en un rango de frecuencias bastante amplio, son utilizadas en reproductores de sonido, generadores de tono, etc. Los zumbadores solo proporcionan una señal audible de espectro constante y solo sirven como indicador de estado o alarma. Requieren de transistores amplificadores para ser conectados a los pines del micro.

Relevos y Contactores: Los relevos y contactores son depósitos mecánicos que accionan unos contactos por la presencia de energía eléctrica en un par de sus terminales, que genera un campo magnético que acciona dos o más interruptores mecánicos con una capacidad relativamente alta de voltaje y corriente, respecto al voltaje y corriente necesario para accionarlos. Los relevos son de capacidad limitada a unos cientos de voltios y decenas de amperios, accionados por voltajes de 5V y 12 V, los contactores son dispositivos más robustos con grandes capacidades de voltaje y corriente. Requieren de un circuito controlado por transistor y protegido por un diodo DAMP o de protección de voltaje inverso o inducido, por tanto no se puede conectar directamente al micro.

Electroválvulas: Las electro válvulas son válvulas de paso para líquidos o gases, controladas por un motor paso a paso, la apertura puede ser controlada gradualmente de manera muy precisa, es utilizada en aplicaciones de control

sobre plantas industriales. Al igual que en los relevos es necesario de un circuito de control por transistor y protegido contra voltaje autoinducido, por tanto no se puede conectar directamente al micro.

Solenoides: Los solenoides son electroimanes con un núcleo metálico recubierto que permiten o no el paso de líquidos o sólidos, la gran diferencia con las electroválvulas es que estas son de control continuo, mientras que los solenoides son del tipo Todo / Nada. El solenoide no puede conectarse directamente al micro, requiere de un circuito amplificador con protección contra voltaje autoinducido.

Motores DC, AC, paso a paso: En la gama de motores se encuentra gran cantidad de dispositivos. Los motores DC presentan la gama más utilizada para el accionamiento continuo sobre las ruedas de un robot, una banda transportadora, etc, estos motores se encuentran en relaciones de voltaje de 3V hasta cientos de voltios, por tanto requieren de drivers de potencia con transistores para controlarlos, no se pueden conectar directamente al micro. Los motores AC pueden ser embobinados o jaula de ardilla, operan en corriente alterna por lo que requieren de driver o circuitos que conviertan los pulsos o señal del micro de bajo voltaje y corriente a voltajes y corriente elevado en AC. Los motores paso a paso generalmente operan en el rango de 5V a 24V, por lo que el micro no puede controlarlos directamente, es necesario un circuito amplificador o driver para proporcionar el voltaje y corriente necesarios.

Lección 33: Diseño de la solución de hardware

La solución de hardware está compuesta por todos los elementos físicos estructurales, que permiten configurar un sistema completo basado en microprocesador o microcontrolador, totalmente funcional y que da solución a un problema de control o procesamiento de datos.

Análisis del problema de diseño desde la perspectiva del hardware

El primer paso para establecer el hardware a utilizar en el diseño de la solución, es generar un esquema general que contenga los elementos necesarios para

incorporar los datos de entrada, las acciones que representan los datos de salida y el núcleo central de la solución representado por el microprocesador o microcontrolador que cumpla con los requerimientos de memoria, velocidad de proceso, pines y módulos internos. Lo primero que se debe establecer es la relación entre entradas, salidas y proceso, mediante un algoritmo o pseudocódigo, este algoritmo nos establece las entradas y su tipo al igual que el proceso y salidas y su tipo, lo que da pautas para la selección adecuada del hardware a utilizar.

Establecimiento de las entradas y salidas

El establecimiento de las entradas y salidas es un ejercicio que parte del algoritmo, pero que debe seguir un proceso minucioso en la selección adecuada del periférico y en muchas ocasiones del circuito relacionado con el periférico para adecuar las señales a valores que el microcontrolador pueda manejar. En este punto se resalta la necesidad de utilizar las hojas técnicas o datasheet de los dispositivos que hacen parte de los circuitos o periféricos de entrada y salida para establecer una compatibilidad completa y minimizar las fallas o errores que muchas veces pueden llegar a causar la destrucción del micro.

Entradas: Lo primero que hay que establecer es las entradas, definir qué tipo de entrada es, si es de tipo digital o análoga, establecer las condiciones de señal de entrada para estudiar la posibilidad de ser adecuadas o convertidas a un formato digital o análogo, que corresponda con las señales propias de un micro, lo más usual es 0 V a 5 V. Teniendo el tipo de entradas y los posibles circuitos o módulos que se requieran para manejarlas se procede a evaluar las salidas requeridas con base en el algoritmo previamente planteado.

Salidas: Las salidas son más complejas de manejar, se debe comenzar por determinar si es del tipo ON/OFF o continua, para poder establecer el tipo de driver, además se debe plantear si opera en un voltaje y corriente más alto DC o AC, lo cual hace que deba considerarse el uso de varias fuentes o de sistemas de opto acople o aislamiento galvánico o magnético para incorporar dichas fuentes. Con cada periférico se tiene un circuito driver que adecua la señal de salida digital por lo general a valores que puedan energizar los actuadores.

Selección del Microprocesador o Microcontrolador

Teniendo como base las entradas y salidas y sus correspondientes circuitos y el tipo de señal a manejar en cada una, del tipo digital o análoga, se procede a seleccionar el microprocesador o microcontrolador más adecuado, el concepto sería encontrar un micro con los pines suficientes y con los módulos integrados

necesarios para reducir el costo y tamaño físico del prototipo al mínimo. El lector debe recurrir entonces a las diferentes hojas técnicas o datasheets del fabricante y familia de microprocesador o microcontrolador que desee emplear, cada fabricante y familia de micros se caracterizan por incorporar módulos especiales, en gran variedad de configuración de pines y puertos I/O disponibles, por lo que es una tarea de sumo cuidado, debe considerarse en esta fase de selección no solo el chip como tal, también el acceso y costo al software IDE, programador, tarjeta de desarrollo y el mismo micro, esto influye enormemente en el proceso de diseño y desarrollo.

Un aspecto a tener presente en la selección del micro, es su longitud pudiendo ser básico de 8 bits o de más capacidad como los de 16 bits o 32 bits, esto repercute directamente en la velocidad de procesamiento, la longitud del programa, la capacidad de manejo de periféricos, la complejidad del sistema y por su puesto en su costo, no siempre el micro con más longitud de registro es el más adecuado, hay gran cantidad de aplicaciones que con solo un micro de 8 bits es más que suficiente para desarrollar una solución completa.

Diseño del diagrama de conexiones

Finalmente se encuentra el diseño del diagrama de conexiones, para lo cual se puede hacer como el algoritmo en una hoja de papel o en un PC, el diagrama de conexiones puede partir simplemente de un esquema generar o diagrama de bloques indicando los periféricos de entrada, los periféricos de salida y el núcleo central compuesto por el microprocesador o microcontrolador. También puede ser una implementación más sofisticada utilizando software de diseño electrónico en el cual directamente puede construirse los sub-circuitos correspondientes a los periféricos de entrada, salida y el mismo microprocesador o microcontrolador.

Al terminar esta fase se debe tener el esquema general de conexiones, parte fundamental para seguir con la simulación del circuito, incorporando la solución de software y depurando en hardware, es decir, haciendo ajustes o modificaciones a los circuitos periféricos hasta conseguir el funcionamiento óptimo. Después de lograr un funcionamiento ideal en el simulador de circuitos, se procede a realizar el montaje físico en protoboard, para lo cual el diagrama de conexiones nos arroja la lista de componentes necesarios que se deben adquirir y el plano de montaje con el que debemos implementar el proyecto en el protoboard para hacer las pruebas y ajustes finales, tanto al circuito en protoboard como al circuito en el simulador, antes de pensar en el diseño de la placa impresa con el prototipo terminado y listo para la producción en masa.

Lección 34: Diseño de la solución de software

Junto con el establecimiento de la solución de hardware, está la solución de software, la cual parte a la par con la de hardware desde el análisis del problema, pasando por el diseño del algoritmo, el código fuente hasta su depuración y funcionamiento en el circuito simulado o en protoboard.

Análisis del problema de diseño desde la perspectiva del software

Para comenzar el análisis de diseño desde la perspectiva de software, se debe replantear el problema desde la óptica del control digital, para poder establecer las entradas y salidas requeridas, las constantes y variables necesarias y por su puesto el proceso que debe realizarse para unir funcionalmente las entradas a las variables y constantes, para generar unas salidas lógicas consistentes con el programa de control. En este punto se establece si la señal es de tipo digital (1 o 0) o del tipo análogo, sin establecer voltajes o corrientes o circuitos, esto hace parte de la solución de hardware, en la solución de software interesa las posibles entradas y como emplear las entradas en la toma de decisiones de control que puedan generar una salida en respuesta a las entradas en un ciclo de control abierto o cerrado (realimentado), las decisiones de control hacen parte del algoritmo que debe generarse y el cual requiere generalmente de la definición de variables y constantes para el control.

Establecimiento de la relación entrada/salida y proceso de control

Después de establecer las entradas y salidas se debe hacer una relación con el proceso de control, es decir, como la información proveniente de las entradas puede repercutir en las salidas como una respuesta que genere un estado estable o controlado sobre una planta o problema. Esta relación se hace analizando el problema desde la óptica de procesos simples, observando el todo y sus partes, y como cada parte puede afectar el funcionamiento del todo. Es un ejercicio lógico que requiere un pensamiento objetivo en el cual debe aprovecharse los estados lógicos binarios, los valores de un registro o los estados de una interrupción, para la toma de decisiones, que generen opciones de respuesta.

Diseño del Algoritmo y diagrama de flujo

El algoritmo parte de tener definidas y claras el tipo y cantidad de entradas y salidas, haber descompuesto el problema en tareas o acciones simples partiendo de la lógica binaria, estados de bits, flags, interrupciones o registros, para tener los parámetros necesarios y diseñar mediante pseudocódigo o diagrama de flujo el algoritmo que da respuesta, no solo al problema de control también a como se

debe recibir y tratar la información recibida por las entradas al micro y como debe ser mostrada o que acción debe generar a la salida del micro.

El algoritmo puede ser tan complejo que requiera dividirse en varia partes, teniendo un algoritmo principal que corresponde con el programa principal y algoritmos alternos que corresponden con las subrutinas o sub programas.

Edición, compilación y depuración del programa

Con el algoritmo y diagrama de flujo establecido es fácil empezar a editar el código fuente, para lo cual se requiere un entorno de desarrollo integrado IDE, que permita la edición del programa con control de sintaxis, lo que hace que se reduzca notablemente los errores en el programa, el IDE debe permitir la compilación de manera que se generen los archivos correspondientes para ser grabados en la memoria principal de programa o código del micro, en el proceso de compilación es muy usual que aparezcan errores de sintaxis o se detecten errores lógicos, por lo que el IDE debe permitir la depuración que es la corrección de estos errores, es conveniente que el IDE permita la simulación con lo que podemos detectar nuevos errores y corregirlos fácilmente.

La mayoría de IDE también permite la programación del dispositivo e incluso la simulación In-Circuit, con lo que se puede depurar más eficientemente el programa, encontrando errores lógicos y de funcionamiento que permiten ajustar el algoritmo o el mismo circuito de control.

Simulación del sistema de control

Como se ha establecido es muy necesario realizar una simulación de todo el sistema de control, esta simulación puede ser realizada de manera lógica por los IDE que suministran los diferentes fabricantes, también puede hacerse más “real” al realizar la simulación en el sistema digital basado en microprocesador o microcontrolador, completamente simulado en la forma de circuitos eléctricos, esto hace que se reduzca aún más las posibles fallas o errores y que se puedan realizar ajustes al prototipo final.

Actualmente muchos sistemas de desarrollo como Texas Instruments con su LaunchPad MS430 y los módulos microcontrolados como BASIC Stamp II y Arduino permiten la simulación In-Circuit con lo que además de tener el circuito físico implementado, se tiene una conexión directa al PC para ejecutar instrucción por instrucción y ver el resultado real.

Lección 35: Integración del hardware y software en la solución de control

Teniendo el hardware y software definido y establecido para la solución de control se continúa con la implementación del prototipo de hardware, la programación de la memoria del micro con las instrucciones del programa, en lo que se conoce como la integración de software y hardware, en este punto se prueba el sistema de control completo, estableciendo posibles errores y fallas que deben ser corregidas para lograr un producto final totalmente funcional en circuito impreso.

Implementación del prototipo de hardware

La implementación del prototipo de hardware inicia con la terminación de la fase de pruebas en el circuito simulado, junto con el programa diseñado y también simulado en el micro. La terminación de la fase de pruebas y ajustes en la simulación por software, permite tener una lista de materiales y componentes, junto con el plano de circuito electrónico, para realizar la implementación física del sistema digital basado en microprocesadores o microcontroladores en protoboard.

El prototipo en protoboard debe caracterizarse por el ordenamiento total de la estructura en módulos funcionales plenamente identificados, es decir, que pueda diferenciarse los circuitos correspondientes a cada periférico externo de entrada o salida, además debe cablearse los circuitos y componentes de manera ordenada, siguiendo uniformidad en colores de cable y disposición de componentes, que permitan un seguimiento de las señales electrónica para poder tomar medidas y determinar fuentes de falla o error. Los cables deben extenderse a nivel de la protoboard evitando trozos de cable excesivo y sobrantes que sobresalgan de la protoboard, esto para evitar emisiones o recepciones electromagnéticas que puedan afectar el funcionamiento o estados lógicos.

Las fuentes de poder deben ser del tipo regulada, generalmente de corriente continua (DC) de 5 V para el micro y 12 V a 24 V para los periféricos y de corriente alterna (AC) entre 48 V y 220 V para manejo de periféricos de mayor potencia estos últimos requieren el montaje aparte en placas universales con aislamiento suficiente para los niveles de voltaje y corriente presentes.

Programación de la memoria con las instrucciones del programa

La programación de las instrucciones del programa en la memoria ROM para el sistema basado con microprocesadores o en el memoria del microcontrolador, requiere de un software especializado para cada micro, generalmente incorporado en el mismo IDE o encontrado de forma individual como versiones libres, gratuitas,

fireware o comerciales. Este software de programación hace uso del archivo con el código máquina, para los microprocesadores el .OBJ, para los microcontroladores el .HEX en el caso de la mayoría de microcontroladores, el cual está compuesto por una serie de bits, representados en cantidades en hexadecimal que codifican el código fuente y por ende el algoritmo mismo.

Mediante una interfaz de hardware conectada al PC, vía puerto paralelo o serie para las interfaces más antiguas pero aún utilizadas y por puerto USB para las modernas actuales, se hace la transferencia del archivo con el código máquina a la memoria o el microcontrolador, alojado en un zócalo dispuesto en el programador, llamado por muchos quemador. Se debe aclarar que cada interfaz o tarjeta de programación tienen una hoja de datos técnica en la cual se especifica el tipo de memoria y micros que soporta, la disposición del integrado en el zócalo y demás parámetros como los indicadores de programación, las fuentes de alimentación del dispositivo, etc, parámetros que deben tenerse en cuenta para evitar daños tanto al micro, como a la interfaz de programación y al mismo PC ya que es un intercambio no solo de datos, sino de voltajes y señales directamente desde los puertos del PC al interior del equipo.

Integración del software con el hardware del prototipo

Una vez programado el microcontrolador o la memoria para el sistema digital basado en microprocesador, se procede a incorporar el CI en el espacio correspondiente en la tabla de prototipos o protoboard, este ejercicio debe hacerse con el circuito del prototipo totalmente des energizado para evitar daños en el micro o en otros componentes, una vez seguro de que esta retirada la fuente de energía y que se tiene establecido el lugar correcto de ubicación, se procede a incorporar al circuito el chip, antes de poder energizar el circuito prototipo, debe cerciorarse nuevamente de posición y conexiones en especial las de alimentación al micro. Se procede entonces a energizar el circuito y comprobar físicamente el funcionamiento y respuesta a las entradas del circuito de control, estableciendo posibles errores y fallas.

Prueba del sistema de control, errores comunes en la fase de implementación

Con la prueba del sistema de control en el circuito prototipo, se pueden encontrar errores y fallas, esto hace que debamos establecer por software puntos de control o indicaciones adicionales que debamos incorporar también en el hardware, para establecer en donde puede estar la falla o error. En este punto son muy útiles las tarjetas de desarrollo como la LaunchPad MSP430 que incorpora la utilidad de

simulación In-Circuit con lo que se puede determinar más fácilmente la fuente del error o falla. Por eso es conveniente tener el circuito en simulador de software para poder localizar fallas, en este punto se resalta la importancia de documentar completamente un programa para poder localizar los errores generalmente de tipo lógico.

El error o motivo de falla más común se debe a problemas lógicos relacionados, con un mal planteamiento del algoritmo, error en el paso del algoritmo al código fuente, error en la lógica de decisiones, etc. Otro error común se debe con el no llamado a la librería adecuada del micro utilizado. También se tiene errores generados por la utilización errónea en una rutina de salto o llamado, en el desbordamiento de PILA, en el salto a rutinas de ciclo infinito sin retorno, en saltos a rutinas de llamado que no generan retorno de PILA, etc.

Como error común a nivel de hardware suele ser la ruptura o aislamiento de algún cable de alimentación o dentro del cableado, falla en la conexión de la protoboard con los componentes por desgaste, por lo que se deben utilizar protoboard en buen estado y asegurarse de retirar el aislante correctamente y en la medida suficiente (cerca de medio centímetro) sin causar daño al cable para asegurar una buena conductividad, evitar que componentes hagan contacto uno con otro y causen corto circuitos.

Diseño de placas impresas del prototipo funcional

Como parte final del desarrollo del prototipo está el diseño de la placa impresa, después de haber realizado las pruebas en el circuito en protoboard, estando seguro de la completa funcionalidad del prototipo y la corrección de los errores y fallas, se procede a diseñar las placas de circuito impreso, programas comerciales como Proteus y Multisim tienen la capacidad de generar la placa impresa a partir del diseño electrónico utilizado para la simulación, programas como Proteus o Crocodile pueden generar además una vista en 3D del prototipo con lo que se puede detectar errores estructurales y de disposición de componentes, para asegurar una adecuada disposición de controles y visualización de resultados o estados al frente, entradas y salidas usualmente en la parte posterior.

Es buena práctica no desmontar el circuito de prueba en protoboard hasta no tener el circuito funcional en impreso, esto como medida de comprobación y comparación, los circuitos impresos pueden fabricarse de manera individual mediante técnicas caseras de paso de imagen de pistas por papel especial y calor, la impresión del impreso en compuestos de vinilo, la generación de la imagen del impreso de forma un poco más industrial con el método de transferencia por

screen y para fabricación industrial de placas impresas en cantidades de más de 100 unidades el pedido a empresas especializadas que fabrican circuitos impresos profesionales de una y más capas.



CAPITULO 8: PROGRAMACION BASICA

Introducción

En este capítulo se trata la programación básica de microcontroladores, entendiendo como programación básica el uso de subrutinas, llamados y ramificaciones en los programas, el uso de los puertos I/O, manejo de interrupciones y timers, manejo de teclado matricial y despliegues digitales 7-segmentos y LCD, que constituyen los aspectos básicos en el aprendizaje y desarrollo de habilidades y competencias para desarrollar aplicaciones simples de control.

Lección 36: Subrutinas, Llamados y Ramificaciones en programas con ensamblador

Las subrutinas, llamados y ramificaciones, hacen parte de la programación de microcontroladores o microprocesadores que le dan a la aplicación un cierto aspecto de estructura de software, es decir, se establecen estructuras de software del tipo “función” a las que se les puede llamar desde cualquier punto del programa y que responde ejecutando una acción determinada e incluso devolviendo valores a parámetros previamente solicitados.

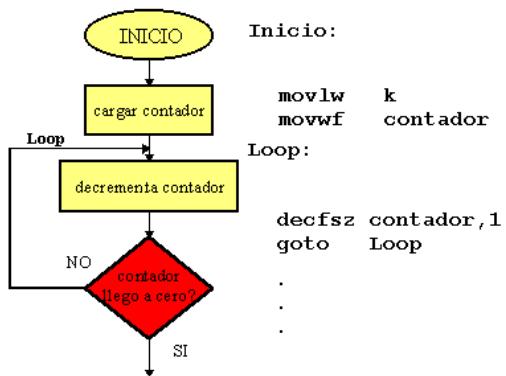
Subrutinas y Llamados

Una subrutina es un subprograma, un trozo de código que sigue una estructura en ensamblador, comenzando con una etiqueta, siguiendo con el cuerpo de programa que ejecuta una función particular o que incluso puede llamar a otra subrutina y un retorno o instrucción de retorno que permite regresar el control al programa principal. Las subrutinas son ejecutadas por llamados realizados desde el programa principal o desde otra subrutina mediante instrucciones especiales como CALL. Una de las subrutinas más comunes es la subrutina de retardo que genera una “pausa” de tiempo determinada para la visualización o retención de datos a la entrada o salida del micro.

Ejemplos de aplicación en PIC

Ciclos de Temporización por software: Existen momentos dentro de la programación en los que se necesita realizar un retardo de tiempo. Los retardos de tiempo se pueden obtener mediante hardware o por medio de ciclos repetitivos basados en software. La precisión de los retardos generados por software depende en esencia del tipo de oscilador que se utilice como base de tiempo en el microcontrolador, la mayor precisión se obtiene de los cristales de cuarzo.

Figura 140. Temporizador



La velocidad a la que se ejecuta el código (instrucciones) depende de la velocidad del oscilador y del número de ciclos de máquina ejecutados. Las instrucciones necesitan 1 ó 2 ciclos de máquina para ser ejecutadas. Un ciclo de máquina es un tiempo utilizado por el microcontrolador para realizar sus operaciones internas y equivale a cuatro ciclos del oscilador. Por tanto: $T_{ciclo\ máq} = 4 * T_{osc}$ $\square\square\square$ $T_{ciclo\ máq} = 4 / f_{osc}$ El número de ciclos de máquina utilizados por una instrucción para ser ejecutada depende de la misma. Las instrucciones que modifican el contador de programa necesitan dos (2) ciclos de máquina, mientras que todas las demás necesitan tan solo uno (1).

Tabla 44. Representación del ciclo de retardo en un PIC

Operación	# de ciclos
la carga de k en W	1
la carga de W en el contador	1
el decremento del contador mientras no llegue a cero	k-1
el decremento del contador cuando llegue a cero	2
el salto a Loop	<u>2 * (k-1)</u>
Total:	3*k+1

El hecho de generar ciclos repetitivos por medio del programa y calcular el tiempo total de ejecución nos puede ayudar a generar tiempos precisos.

Ciclo repetitivo de retardo: Tomando el ciclo repetitivo de retardo mostrado en el diagrama de flujo, se tomará un número de ciclos así:

Por cada instrucción agregada debe incluirse en la cuenta total el número de ciclos correspondiente a dicha instrucción. Trabajando a 4 Mhz y asumiendo que k se remplaza por el valor 15d en el ejemplo tendríamos

n tiempo igual a:

Número de ciclos = $(3 * 15) + 1 = 46$ ciclos de máquina,

$T_{ciclo\ máq.} = 4 / 4\ \text{Mhz} = 1\ \mu\text{segundo}$, el tiempo total del ejemplo entonces será 46 $\mu\text{segundos}$.

Consultas de tablas.

En muchas ocasiones es necesario para un programador efectuar una coincidencia entre alguna cantidad de valores conocidos y un número desconocido que se tiene como índice, por ejemplo, basados en el contenido de una posición de memoria RAM (índice) se puede obtener de una serie consecutiva de datos almacenados en memoria de programa (a estos datos "conocidos" almacenados se le denomina tabla), el dato desplazado **n** posiciones adelante del comienzo de esta tabla, este número **n** corresponde al contenido de la posición de memoria RAM ó índice.

Programa ejemplo:

```
offset equ 0Ch ;posición de memoria RAM
w      equ 0      ;destino W
f      equ 1      ;destino F
.....
.....
.....
movf  offset,w ;tomamos a W el número n utilizado como índice
call   tabla    ;posición en donde se encuentra la serie de datos
                ;en este sitio luego del retorno de la subrutina se tiene en W el dato leído
e la tabla
.....
.....
.....
tabla
addwf PCL,f ;se suma al PC el contenido de W obteniendo como resultado un salto Indexado
retlw  30h ;si el contenido de W sumado al PCL es 0 se retorna en esta posición, =30h
retlw  31h ;si el contenido de W sumado al PCL es 1 se retorna en esta posición, =31h
retlw  32h ;si el contenido de W sumado al PCL es 2 se retorna en esta posición, =32h
retlw  33h ;si el contenido de W sumado al PCL es 3 se retorna en esta posición, =33h
```

```

retlw 34h ;si el contenido de W sumado al PCL es 4 se retorna en esta posición, =34h
retlw 35h ;si el contenido de W sumado al PCL es 5 se retorna en esta posición, =35h
.
;
    ...

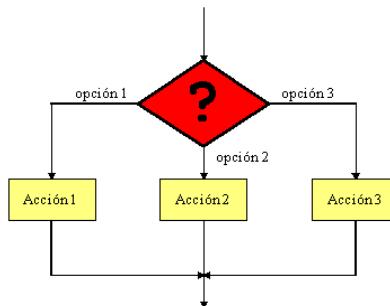
```

Finalmente y luego de observar el ejemplo anterior, podemos anotar que antes de hacer el llamado a la subrutina **tabla**, se debe cargar en el registro de trabajo W el valor del índice y una vez se retorne de dicha subrutina, es en este mismo registro de trabajo en donde se obtiene el resultado de la consulta a la tabla (vemos que la sucesión de instrucciones **retlw k** se encuentra en memoria de programa). En los microcontroladores PIC las tablas deben estar dentro de los primeros 256 Bytes de memoria de programa, esto es porque solo se tiene acceso y modificación a los primeros 8 bits del registro “PC o contador de programa”, es decir, desde el bit cero al bit siete.

Ramificaciones en programas con ensamblador

Cuando se tiene que solucionar un diagrama de flujo como el de la figura, en el cual tenemos tres posibles respuestas a una pregunta, se plantean las soluciones aquí presentadas.

Figura 141. Tres posibilidades para una pregunta



Ejemplos de aplicación en PIC

Primera Solución: Una de las formas de solucionar en un programa este problema es:

Determinando para la opción 1, la opción 2 y la opción 3 un valor consecutivo como:

```

opción1 equ 0
opción2 equ 1
opción3 equ 2

```

Uno de estos posibles valores llevarlos a W y en una parte del programa tratarlos así:

Decisión: ;sitio en donde la pregunta "?" tendría solución

```
addwf PCL,1  
goto Acción1  
goto Acción2  
goto Acción3
```

Acción1:

```
..... ;instrucciones correspondientes a la Acción 1
```

```
.....  
goto encuentro
```

Acción2:

```
..... ;instrucciones correspondientes a la Acción 2
```

```
.....  
goto encuentro
```

Acción3:

```
..... ;instrucciones correspondientes a la Acción 3
```

encuentro: ;sitio de encuentro luego de una de las acciones

```
..... ;continuación del programa
```

Segunda Solución: Otra forma posible es comparando una por una los valores de las diferentes opciones almacenadas en memoria RAM en una variable llamada **OPCION**

```
movlw Opción1  
xorwf OPCION,0 ;se realiza la verificación del contenido de OPCION con respecto a W  
btfsr STATUS,Z ;Verificando la bandera Z
```

goto Acción1

```
movlw Opción2
```

```
xorwf OPCION,0 ;se realiza la verificación del contenido de OPCION con respecto a W
```

```
btfsr STATUS,Z ;Verificando la bandera Z
```

goto Acción2

```
movlw Opción3
```

```
xorwf OPCION,0 ;se realiza la verificación del contenido de OPCION con respecto a W
```

```
btfsr STATUS,Z ;Verificando la bandera Z
```

goto Acción3

Acción1:

```
..... ;instrucciones correspondientes a la Acción 1
```

```
.....  
goto encuentro
```

Acción2:

```
..... ;instrucciones correspondientes a la Acción 2
```

```
.....  
goto encuentro
```

Acción3:

..... ;instrucciones correspondientes a la Acción 3

.....

encuentro:

..... ;sitio de encuentro luego de una de las acciones

..... ;continuación del programa

.....

Aunque este último método es más largo que el anterior, permite que los valores de las diferentes opciones no sean consecutivos entre si.

Lección 37: Operaciones de entrada / salida con ensamblador

Las operaciones de entrada / salida con ensamblador requieren la configuración de los puertos, ya sea como entrada o salidas. Teniendo establecido el tipo de operación es posible escribir directamente sobre los puertos.

Requerimientos para la operación de entrada / salida (I/O)

Para la operación de entrada/salida se debe tener presente:

- Verificar el modo en el que se debe programar el sentido de los puertos
- Realizar las entradas por puerto mediante la lectura de interruptores "dip-switch"
- Escribir sobre un puerto de salida visualizando sobre LEDs

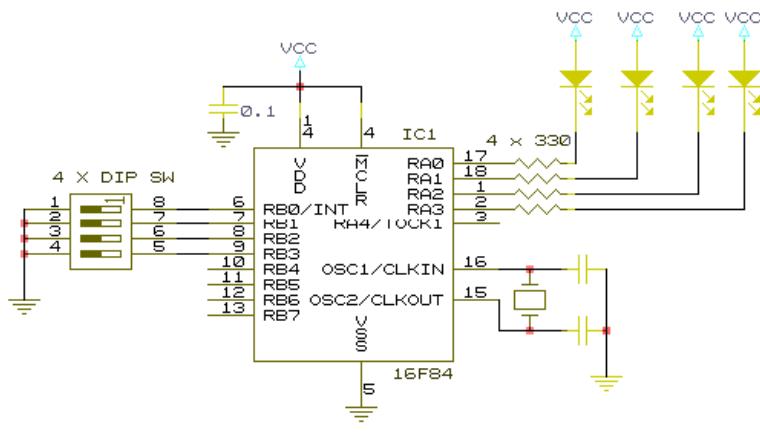
Procedimiento:

En el proceso de utilización de un puerto debe tenerse en cuenta como primera instancia la programación del sentido en que dicho puerto va a utilizarse. Una vez energizado el microcontrolador todos y cada uno de los puertos quedan programados como entrada, entonces, tan solo deben programarse los que se quieren utilizar como salida.

En el hardware de la figura se observa que se han colocado 4 dip switch al puerto B y estos no poseen resistencia de pull-up lo cual nos obliga a habilitar las resistencias internas con las que cuenta el microcontrolador PIC16F84, el programa debe entonces en un repetitivo infinito leer el nivel lógico que colocan los switch y pasar este resultado al puerto A complementando el estado de la información puesto que de acuerdo a la disposición de los LEDs un estado bajo en

el puerto enciende el LED correspondiente y por ende un estado alto en el puerto, apaga el LED.

Figura 142. Hardware para ejercicio Entrada/Salida



Programa:

```

status equ 03h
optionr equ 81h
trisa equ 85h
porta equ 05h
trisb equ 86h
portb equ 06h
;
Inicio:
bsf status,5 ;se pasa al banco 1 de RAM
clrf trisa ;se programa el puerto A como salida
movlw 0Fh ;dato para la programación del puerto B
movwf trisb ;parte alta como salida y parte baja como entrada
bcf optionr,7 ;se habilitan resistencias de Pull Up
bcf status,5 ;se pasa al banco 0 de RAM
Loop:
comf portb,0 ;se lee el puerto B, se complementa su valor y el ;resultado pasa a W
movwf porta ;se pasa el resultado de W al puerto A
goto Loop ;ejecuta un ciclo infinito
end

```

En un proceso de lectura de interruptores, casi siempre cuando se detecta un cambio en el estado, es aconsejable amortiguar la lectura de estos con un retardo de software. Dependiendo de la calidad del interruptor el tiempo del retardo puede estar alrededor de 50 mS. En el caso de este ejercicio en particular no es requerido puesto que un cambio en el interruptor debe reflejarse inmediatamente en el puerto de salida. Se debe tener en cuenta que nunca una entrada debe quedar al aire puesto que los microcontroladores PIC son hechos con tecnología CMOS. Es por este motivo que en el programa se programó la parte alta del puerto B como salida.

Lección 38: Manejo de interrupciones y timers

Las interrupciones y Timers son dos módulos ampliamente utilizados en microcontroladores y microprocesadores, como medio para interactuar con el exterior y dar respuestas rápidas a eventos extraordinarios, los Timers se utilizan como bases de tiempo.

Interrupciones internas

Las interrupciones internas hacen referencia a interrupciones por desbordamiento del Timer, también por la terminación de ejecución de un módulo especial como el de conversión analógica digital.

Interrupciones externas

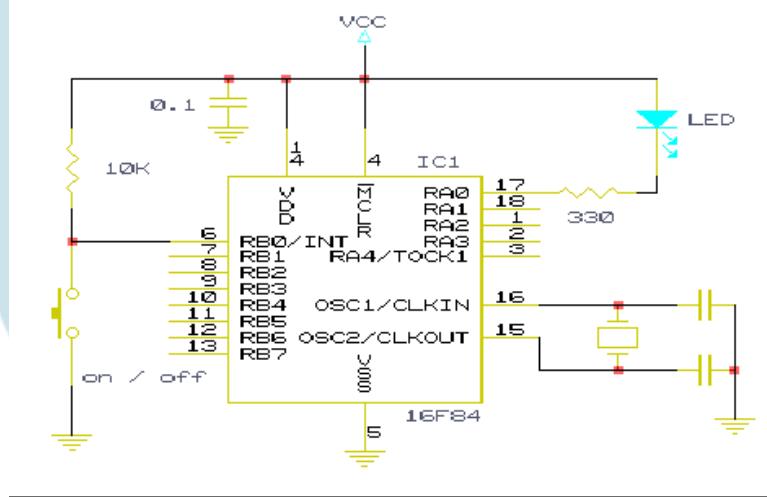
Las interrupciones externas hacen referencia a pines que reciben cambio de nivel o flanco directamente del exterior y pueden indicar la atención urgente a un evento.

Ejemplos de aplicación de interrupciones en PIC

Servicio de interrupción Objetivos:

- Encender y apagar un LED como respuesta a un estímulo de interrupción
- Determinar la forma en que debe colocarse el código en el programa fuente para aceptar y atender una rutina de interrupción

Figura 143. Ejemplo de servicio a Interrupción externa en PIC16F84



Procedimiento: Para exemplificar el uso de un servicio de interrupción se ha dispuesto el hardware de la siguiente figura se decide utilizar la interrupción externa INT (RB0) en un PIC16F84, esta interrupción está vectorizada a la dirección de memoria de programa 004h, dentro de la atención a esta interrupción se opta por complementar el estado del LED colocado al puerto RA0 cada vez que esta sea atendida.

Listado del programa en assembler:

```
;Programa para realizar el apagado y encendido de un LED colocado en el ;Puerto A0 basado en la
;interrupción externa INT (RB0)
;
list p=16F84
status equ 0x03
porta equ 0x05
portb equ 0x06
intcon equ 0x0B
optionr equ 0x81
trisa equ 0x85
trisb equ 0x86
#define LED porta,0
#define BANK1 bsf status,5
#define BANK0 bcf status,5
org 000h ;Indica al ensamblador la dirección de memoria de
           ;la sig. instrucción
goto Inicio
org 004h ;Indica al ensamblador la dirección de memoria de
           ;la sig. instrucción
Interrupcion
    btfss intcon,1 ;es interrupción INT?
    retfie          ;retorna de interrupción y GIE=1
    btfsc porta,0  ;probar estado actual del LED
    goto Prender  ;va a encender el LED
Apagar
    bsf porta,0  ;apaga el LED
    goto Espera
Prender
    bcf porta,0  ;enciende el LED
Espera
    btfss portb,0 ;espera a que se suelte el pulsador
    goto Espera
    bcf intcon,1 ;borra bandera INT
    retfie          ;retorna de interrupción y GIE=1
;Programa principal
Inicio BANK1      ;selección del banco 1
               ;selección de salida
               ;selección de banco 0
               ;apagar LED
;programación de interrupción
    bsf intcon,4 ;activar interrupción INT   BANK1      ;selección banco 1
    bcf optionr,6 ;selección del flanco de bajada en el pin INT
               ;BANK0
```

```
bsf intcon,7 ;Habilitar interrupciones globales
goto $      ;quedá a la espera de interrupción
end
```

El símbolo \$ significa la dirección de memoria de programa en donde se encuentra éste (ciclo Infinito de espera). Debe notarse la ubicación de la rutina de interrupción a partir de la posición de memoria de programa 004h.

Temporizadores

El uso de temporizadores está ligado a eventos de conteo interno por los pulsos de reloj o externos por nivel o flanco de señal de entrada, también están relacionados con una atención a interrupción por desbordamiento del Timer.

Ejemplos de aplicación de Timers en PIC

En el siguiente programa se observa cómo se configura las interrupciones tanto externa como por Timer, con ajuste del preescaler del TIMER0.

```
;PROGRAMA CONFIGURACION DE INTERRUPCION
CONFINT    BANK1          ;CAMBIO DE BANCO DE MEMORIA AL BANCO 1
;      BSF   OPCION, INTEDG ;SELECCION DE FLANCO DE INTERRUPCION INT - SUBIDA
      BSF   OPCION, PS0   ;PREESCALER PS2=0-PS1=0-PS0=1
      BCF   OPCION, PS1   ;PREESCALER PS2=0-PS1=0-PS0=1
      BCF   OPCION, PS2   ;PREESCALER PS2=0-PS1=0-PS0=1
      BCF   OPCION, PSA   ;ASIGNACION DEL PREESCALADOR 0=TMR0
      BCF   OPCION, RTE   ;FLANCO DE LA SEÑAL TMR0 TRANSICION BAJO A ALTO
      BCF   OPCION, RTS   ;FUENTE DE LA SEÑAL DE TMR0 CICLO DE INSTRUCCIONES INTERNO
      BANK0          ;CAMBIO DE BANCO DE MEMORIA AL BANCO 0
      BCF   INTCON, TOIF  ;COLOCA A 0 FLAG DE OVERFLOW TMR0
      BSF   INTCON, TOIE  ;HABILITACION DE INTERRUPCION TMR0
      BSF   INTCON, GIE   ;HABILITA INTERRUPCIONES GLOBALES
RETURN
```

Lección 39: Exploración de teclado matricial con ensamblador

El teclado matricial es uno de los componentes más utilizados para el ingreso de datos al sistema, un ejemplo de teclado matricial es el teclado que utilizamos en el teléfono y el PC. Para microcontroladores se inicia con el uso de un teclado matricial simples de 3X3 (9 pulsadores) o 4X4 (16 pulsadores).

Teclado matricial

Un teclado matricial está compuesto por arreglo de conductores, dispuestos en filas y columnas, en la intersección de cada línea se coloca un pulsador, por tanto el teclado matricial se conecta como filas y columnas, en un programa se puede enviar pulsos a las filas una a una y escanear las columnas o lo contrario, de esta manera se encuentra fácilmente por software el pulsador responsable de la señal.

Conexión e identificación de estados

La conexión del teclado matricial se hace identificando las filas y columnas, para después seleccionar cuál de las dos se utiliza como entrada de datos y cual como salida. Por ejemplo, se seleccionan los pines del teclado perteneciente a la fila y se conectan al micro el cual configura sus pines como salida, los otros pertenecientes a las columnas se conectarán al micro, el cual los configura como salida, el micro envía pulsos a una fila a la vez y escanea el resultado en cada pin perteneciente a la columna, hasta logras escanear todo el teclado, de esta manera logra censar todos los pulsadores, esta acción ocurre en unos cientos de microsegundos para un reloj cristal de 4MHz, pudiendo detectar la pulsación de cualquier tecla, dado que requerimos entre 50 ms y 500 ms para accionar una tecla.

Ejemplo de teclado matricial con PIC

En el siguiente ejemplo se ilustra el manejo de un teclado matricial 3x4, conectado a un microcontrolador PIC16F84, en este se muestra la rutina de control de teclado vinculada a una tabla que codifica el pulso recibido a una respuesta en valor hexadecimal de la tecla correspondiente.

```
;RUTINA DE EVALUACION DE TECLADO
TECLAD MOVLW 01H          ;VALOR UNO EN COLUMNA !!!PRIMERA COLUMNA!!!!
                      ;REGISTRO PROPOSITO GENERAL MANEJA ESTADO DE COLUMNAS
TECLADO
    MOVLW B'00001110'      ;CODIGO PARA EVALUAR PRIMERA COLUMNA
    MOVWF CODCOL           ;REGISTRO PROPOSITO GENERAL MANEJA ESTADO DE CODIGOS
COLUMNAS
    RSTFIL BSF STATUS,C     ;PONER EN UNO EL CARRY
    CLRF CONFIL             ;REGISTRO PROPOSITO GENERAL MANEJA ESTADO DE FILAS TECLADO
    SWAPF TECLA, W           ;CARGAR W-REGISTRO CON VALOR DE TECLA INVIRTIENDO NIBBLES
    IORWF CODCOL,           ;W ;ENVIA CODIGO DE COLUMNA AL REGISTRO W
    MOVWF PTB
    NOP
    NOP
    NOP
    MOVF PTA, W             ;ENVIA DATO POR EL PUERTO B
    ANDLW 0FH                ;MASCARA ' 0000 1111 '
    MOVWF AUX                ;REGISTRO AUXILIAR DE TECLADO
TESTFIL RRF AUX, F           ;ROTACION A LA DERECHA PONE EL ESTADO DEL FILA EN EL CARRY
    BTFSS STATUS,C           ;REVISA EL ESTADO DEL ESTATUS
    GOTO ACRTD               ;ES ACERTADO HAY UNA TECLA OPRIMIDA
    INCF CONFIL, F           ;REGISTRO PROPOSITO GENERAL MANEJA ESTADO DE FILAS TECLADO
```

MOVF CONFIL, W	;REGISTRO PROPOSITO GENERAL MANEJA ESTADO DE FILAS TECLADO
XORLW 03H	;CANTIDAD DE FILAS DEL TECLADO
BTFSS STATUS, Z	;¿LLEGO A LA ULTIMA FILA?
GOTO TESTFIL	;SUB-RUTINA DE TESTEO DE LA SIGUIENTE FILA
BSF STATUS, C	;SIGUIENTE COLUMNA, BORRAR EL CARRY
RLF CODCOL,	W ;ROTA EL VALOR PARA ESCANEAR LA SIGUIENTE COLUMNA
ANDLW 0FH	;ENMASCARA EL CODIGO DE TESTEO DE LA SIGUIENTE COLUMNA 0000****
MOVWF CODCOL	;GUARDA EL NUEVO CODIGO DE COLUMNA EN EL REGISTRO CODCOL
INCF CONCOL, F	;REGISTRO PROPOSITO GENERAL MANEJA ESTADO DE
COLUMNAS TECLADO	
XORLW 05H	;LIMITE + 1 DE COLUMNAS A ESCANEAR
BTFSS STATUS, Z	;¿EL CONTADOR DE COLUMNAS ESTA EN CINCO?
GOTO RSTFIL	;COMENZAR DE NUEVO
CLRF TECLA	;BORRADO DEL REGISTRO CON VALOR DE TECLA
RETLW 00H	
ACRTD MOVF CONFIL, W	;CONTADOR DE FILAS AL W
XORLW 00H	;EN CASO DE QUE EL CONTADOR DE FILAS SEA CERO
BTFSC STATUS, Z	;¿ESTA EN LA FILA CERO?
GOTO FLCERO	;EN CASO DE QUE FILAS = 0 ==> PCL = COLUMNAS
MOVLW 00H	;BORRA EL REGISTRO W
MULT ADDLW 04H	;MULTIPLICACION PCL = (FILAS) * 4 + COLUMNAS
DECFSZ CONFIL, F	;REGISTRO PROPOSITO GENERAL MANEJA ESTADO DE FILAS TECLADO
GOTO MULT	;SALTA A RUTINA PARA MULTIPLICAR POR CUATRO EL CONTADOR DE FILAS
SUMCOL ADDWF CONCOL, W	;REGISTRO PROPOSITO GENERAL MANEJA ESTADO DE
COLUMNAS TECLADO	
CALL TABLA1	;TABLA CON LOS VALORES DEL TECLADO
MOVWF TECLA	;CARGA EL REGISTRO TECLA CON EL VALOR DE LA TECLA OPRIMIDA
RETLW 00H	;RETORNA
FLCERO MOVLW 00H	;DATO EN CASO DE QUE FILA TESTEADA SEA CERO
GOTO SUMCOL	;SALAR A SUB-RUTINA DE SUMA DE COLUMNAS
;TABLA DE COMPORTAMIENTO DE TECLADO	
TABLA1 ADDWF PCL, F	;SUMA DEL PROGRAM COUNTER + VALOR DE (FILAS) * 4 + COLUMNAS
RETLW 00H	;VALOR DE ERROR
RETLW 03H	;PRIMERA FILA PRIMERA COLUMNA
RETLW 06H	;PRIMERA FILA SEGUNDA COLUMNA
RETLW 09H	;PRIMERA FILA TERCERA COLUMNA
RETLW 0B0	;PRIMERA FILA CUARTA COLUMNA
RETLW 02H	;SEGUNDA FILA PRIMERA COLUMNA
RETLW 05H	;SEGUNDA FILA SEGUNDA COLUMNA
RETLW 08H	;SEGUNDA FILA TERCERA COLUMNA
RETLW 00H	;SEGUNDA FILA CUARTA COLUMNA
RETLW 01H	;TERCERA FILA PRIMERA COLUMNA
RETLW 04H	;TERCERA FILA SEGUNDA COLUMNA
RETLW 07H	;TERCERA FILA TERCERA COLUMNA
RETLW 0D0H	;TERCERA FILA CUARTA COLUMNA

Lección 40: Manejo de display 7 segmentos y pantallas LCD con ensamblador

Siempre en un proyecto o solución se requiere una interfaz visual, que entregue al usuario información de estado, los ejemplos más usuales son los display 7-segmentos y los display tipo pantallas LCD.

Display 7 segmentos

Un display 7-segmentos es un arreglo de LEDs conectados y distribuido de manera que forman un carácter reconocido, básicamente el display 7-segmentos, contiene 7 secciones con las que puede representarse los dígitos del 0 al 9. Estos display se encuentran del tipo ánodo común o cátodo común, esto se refiere a que todos los LEDs comparten una misma terminal de ánodo o una misma terminal de cátodo.

Conexión e identificación de pines del display 7 segmentos

Para conectar un display digital 7-segmentos a un microcontrolador se puede llevar cada terminal de segmento directamente a un pin del micro, mediante una resistencia de poco valor entre 220 Ohm y 1000 Ohm, se debe calcular de forma que no sobrecargue las salidas del micro y el micro en total, consultar el datasheet del micro para mayor seguridad y hacer los cálculos utilizando la ley de Ohm, la terminal común se conecta ya sea al positivo (ánodo) o al negativo (cátodo) del sistema de alimentación. Otro tipo de conexión se realiza utilizando un conversor BCD a 7 segmentos como el 7448 o 7447.

Pantalla LCD

Las pantallas LCD son dispositivos más estructurados, se componen de una pantalla de cristal líquido, la cual es controlada por un microcontrolador, las pantallas se presentan en formato de 1 línea por 16 caracteres (1x16), dos líneas por 16 caracteres (2x16) o múltiples líneas con múltiples caracteres, hasta llegar a pantallas gráficas GLCD. La pantalla LCD tiene estandarizado el uso de determinados pines como entrada y salida de datos, pines de control para el micro, pin de contraste y retroiluminación y pines de alimentación. Por tanto es conveniente referirse al datasheet del producto para mayor información.

Conexión e identificación de pines de la pantalla LCD

Generalmente para proyectos de aprendizaje con micros, se utilizan pantallas LCD 1x16 o 2x16, las cuales implementan 8 pines I/O por donde se envían datos en codificación ASCII para representar los caracteres alfanuméricos y especiales, tres pines de control para lectura, escritura y habilitación de la pantalla para recepción de datos, un pin de contraste para controlar la nitidez de la pantalla mediante circuito externo por medio de un potenciómetro, un pin para habilitar la alimentación a la retroiluminación de la pantalla y los pines de alimentación GND y 5V.

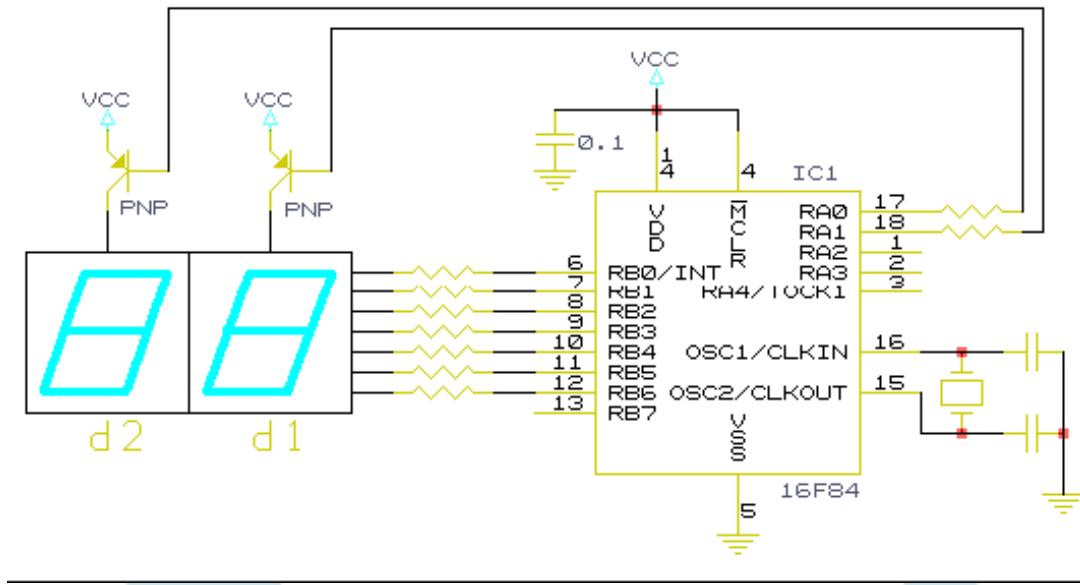
Dentro del programa es necesario inicializar la pantalla LCD, mediante el envío de datos específicos de configuración, luego es necesario enviar combinaciones especiales de pulsos de control y datos para “programar” o configurar la pantalla y enviar los datos que se quieren visualizar, las pantallas LCD tienen una memoria que puede utilizarse para generar caracteres propios, para almacenar datos, se puede utilizar como un periférico de memoria con los datos que se le envían como información temporal recuperable.

Ejemplo de manejo de display 7-segmentos con PIC16F84

Objetivos:

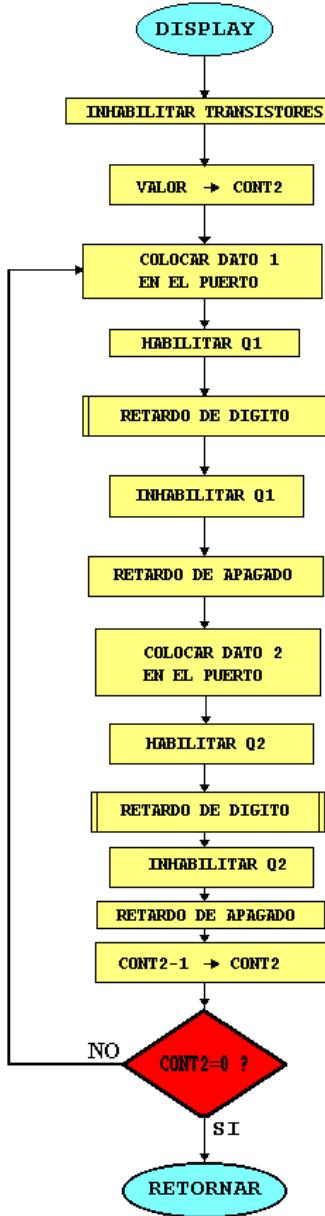
- Realizar la decodificación de BCD a 7 segmentos por software
- Multiplexar en el tiempo la información para 2 dígitos 7 segmentos

Figura 144. Diagrama eléctrico para visualización dinámica en display 7 segmentos de dos dígitos



Procedimiento: El hecho de visualizar datos en display de 7 segmentos de la forma en que se muestra en la figura anterior obliga a interconectar entre si los pines correspondientes a los segmentos del dígito 1 (d1) con los pines de los segmentos del dígito 2 (d2), de esta manera se ahorran líneas de conexión. El método utilizado para la visualización dinámica consiste en visualizar cada dígito durante un instante de tiempo y a una velocidad tal que gracias a la persistencia del ojo el efecto final es que todos los dígitos están encendidos al tiempo.

Figura 145. Diagrama de Flujo manejo de display con PIC16F84



;Sacar al puerto el Dato 2 por un tiempo específico

```

movf  Dato2,W   ;Dato para decodificar
call   Tabla     ;Decodificación del dato
movwf PORTB    ;Dato decodificado a puerto
bcf    PORTA,1   ;Habilita Q dato 2
call   RetDig    ;Retardo de dígito
bsf    PORTA,1   ;Inhabilita Q dato 2
nop
nop
nop
nop
decfsz Cont2,F  ;Decrementa Cont2, elude sig. si cero
Goto  LoopDisp   ;Repite ciclo
return
    
```

Tabla

```

addwf PCL,F
retlw 0x01 ;Cuando el dígito es 0
retlw 0x4F ;Cuando el dígito es 1
retlw 0x12 ;Cuando el dígito es 2
retlw 0x06 ;Cuando el dígito es 3
retlw 0x4C ;Cuando el dígito es 4
retlw 0x24 ;Cuando el dígito es 5
retlw 0x20 ;Cuando el dígito es 6
retlw 0x0F ;Cuando el dígito es 7
retlw 0x00 ;Cuando el dígito es 8
retlw 0x04 ;Cuando el dígito es 9
retlw 0x08 ;Cuando el dígito es A
retlw 0x60 ;Cuando el dígito es B
retlw 0x31 ;Cuando el dígito es C
retlw 0x42 ;Cuando el dígito es D
retlw 0x30 ;Cuando el dígito es E
retlw 0x38 ;Cuando el dígito es F

RetDig
    movlw 2
    movwf Del1

Loop1
    movlw .50
    movwf Del2

Loop2
    decfsz Del2,F
    goto Loop2
    decfsz Del1,F
    goto Loop1
    return

```

Ejemplo de manejo de pantalla LCD con PIC

La interface que se va a implementar, para el trabajo con el *display* de cristal líquido, es la del bus de datos de 8-bits. Antes de presentar un programa que controla el comportamiento del *display*, se van a describir en el siguiente cuadro las instrucciones típicas para su manejo y configuración.

En el cuadro de instrucciones del *display* se emplearon las siguientes abreviaturas:

- * : posición no válida.
- I / D : dirección del cursor. 1, hacia la derecha. 0, hacia la izquierda.
- S : desplazamiento del *display*. 1, activado. 0, desactivado.
- D : *display*. 1, activado. 0, desactivado.
- C : cursor. 1, activado. 0, desactivado.
- B : parpadeo del cursor. 1, activado. 0, desactivado.
- S / C : desplazamiento / cursor. 1, desplazamiento del *display*.
0, movimiento del cursor.
- R / L : desplaza el *display*. 1, hacia la derecha. 0, hacia la izquierda.
- DL : ancho de la interface. 1, 8-bits. 0, 4-bits.
- N : líneas del *display*. 1, dos líneas. 0, una línea.
- F : puntos de los caracteres. 1, 5x10 puntos. 0, 5x7 puntos.
- BF : 1, operación interna en proceso. 0, instrucción aceptada.

DDRAM : RAM de datos del *display*.

CGRAM : RAM del generador de caracteres.

ACG : dirección del CGRAM

ADD : dirección del DDRAM.

AC : contador de direcciones.

Tabla 45. Pines y funciones LCD

Instrucción	R/S	R/W	D B	D B	D B	DB	DB	DB	DB	Descripción			
			7	6	5	4	3	2	1				
Limpiar Display	0	0	0	0	0	0	0	0	1	limpia el <i>display</i> y lleva el cursor a la dirección cero			
Cursor at Home	0	0	0	0	0	0	0	1	*	lleva el cursor a la dirección cero sin cambiar el contenido			
Modo entry set	0	0	0	0	0	0	0	1	I/D	controla el movimiento del cursor y activa el desplazamiento del <i>display</i>			
Control ON/OFF	0	0	0	0	0	0	1	D	C	activa o desactiva el <i>display</i> , el cursor y el parpadeo del cursor			
Cursor / Display	0	0	0	0	0	1	S/C	R/L	*	muestra el cursor o desplaza el <i>display</i> sin cambiar al contenido de la pantalla			
Función SET	0	0	0	0	1	D L	N	F	*	define el ancho de interface, el número de líneas del <i>display</i> y caracteres			
Dirección DDRAM	0	0	0	1	ACG						define la dirección en el <i>display</i> del RAM del generador de caracteres		
Dirección DDRAM	0	0	1	ADD						define la dirección en el <i>display</i> del RAM de datos del <i>display</i>			
Busy Flag / Addres Read	0	1	BF	AC						indica si la operación interna está en proceso y lee direcciones de contenido			
Escritura de Datos	1	0	Escritura de Datos						escribe los datos en el DDRAM o en el CGRAM				
Lectura de Datos	1	1	Lectura de Datos						lee datos del DDRAM o del CCRAM				

Después de inicializar el display, se realiza el programa de interface, el cual permite que se visualicen los mensajes, datos y resultados que se necesiten o se quieran presentar durante la aplicación. En el siguiente ejemplo se presenta la interface de 8-bits PIC - display, en la que se indica la forma de inicializar el display y de visualizar un mensaje.

```
TITLE "PIC16F84A EEPROM PROGRAM"
LIST P=16F84A,F=INHX32
#INCLUDE <P16F84A.INC>

;INGENIERO ELECTRONICO HECTOR URIEL VILLAMIL GONZALEZ
;PROCESO: CONTROL LCD 2X16 BUS 8 BITS
;ENTRADAS: RA4 SELECCION DE COMPORTAMIENTO
;SALIDAS: RA0=RS / RA1=R/W /RA2=E / DB7=RB7 / DB6=RB6 / DB5=RB5 / DB4=RB4 / DB3=RB3 / DB2=RB2 / DB1=RB1
/ DB0=RB0
;ANEXO ARCHIVO PROTEUS

;*****;DEFINICION DE REGISTROS DE PROPOSITO ESPECIAL FSR

TMRO EQU 01H ;REGISTRO TEMPORIZADOR CERO
PCL EQU 02H ;PROGRAM COUNTE PARTE BAJA
STATUS EQU 03H ;REGISTRO DE ESTADO DEL PIC
PTA EQU 05H ;PUERTO A DEL PIC
PTB EQU 06H ;PUERTO B DEL PIC
EEADATA EQU 08H ;DATO IN/OUT EN EEPROM
EEADR EQU 09H ;DIRECCION DE IN/OUT DE EEPROM
INTCON EQU 0BH ;REGISTRO DE INTERRUPCIONES
OPCION EQU 81H ;OPCIONES DE CONFIGURACION INT TMP PULLUP PREESCALER
;PCL EQU 82H ;CONTADOR DE PROGRAMA PARTE BAJA
TRISA EQU 85H ;/RA7-RA6-RA5=NOP/RA4-/RA3-/RA2-/RA1-/RA0=/
TRISB EQU 86H ;/RB7-/RB6-/RB5-/RB4-/RB3-/RB2/RB1-/RB0=/
EECON1 EQU 88H ;B4=EEIF/FIN WR/*B3=WRERR/ERROR WR/*B2=WREN/PERMISO WR/*B1/WR/*B0/RD/
EECON2 EQU 89H ;REG SEGURIDAD PROCESO EEPROM

;*****;DEFINICION REGISTROS FSR

MENU EQU 0CH ;REGISTRO PARA EL MENU
;*****
REGT1 EQU 28H ;REGISTRO TEMPORAL UNO PROPOSITO GENERAL
LCD4B EQU 29H ;REGISTRO CONTROL BUS TX 4 BITS AL LCD
VMAX EQU 2AH ;REGISTRO MAXIMO / MINIMO VARIABLE DE TEMPORIZADOR
SAVSTA EQU 2BH ;REGISTRO SALVA CONTENIDO DE STATUS
SWREG EQU 2CH ;REGISTRO SALVA CONTENIDO DE W
TMP1 EQU 2DH ;REGISTRO TEMPORAL 1 TIEMPOS MEDIANOS
TMP2 EQU 2EH ;REGISTRO TEMPORAL 2 TIEMPOS MEDIANOS
TMP3 EQU 2FH ;REGISTRO TEMPORAL 3 TIEMPOS MEDIANOS

;*****;DEFINICION DE BITS

W EQU 0 ;REGISTRO DE TRABAJO
F EQU 1 ;REGISTRO
;*****STATUS
C EQU 0 ;FLAG DE CARRY
Z EQU 2 ;FLAG DE CERO
RP0 EQU 5 ;SELECTOR DE PAGINA BIT 0
RP1 EQU 6 ;SELECTOR DE PAGINA BIT 1
;*****OPTION
PSO EQU 0 ;PREESCALER BIT 0
PS1 EQU 1 ;PREESCALER BIT 1
PS2 EQU 2 ;PREESCALER BIT 2
PSA EQU 3 ;ASIGNACION DEL PREESCALADOR 0=TMR0 1=WDT
RTE EQU 4 ;FLANCO DE LA SEÑAL DE INCREMENTO DEL TMR0 0=L-H 1=H-L
RTS EQU 5 ;FUENTE DE LA SEÑAL DEL TMR0 0=TMP 1=CONTADOR
INTEDG EQU 6 ;FLANCO DE LA SEÑAL DE INTERRUPCION INT 0=LOW 1=UP
RBU EQU 7 ;PULL-UPS INTERNAS 0=E 1=D
```

```

.*****EECON1
RD EQU 0 ;RD=1 CICLO DE LECTURA DE LA EEPROM
WR EQU 1 ;WR=1 CICLO DE ESCRITURA DE LA EEPROM
WREN EQU 2 ;WREN=1 AUTORIZA PERMISO DE ESCRITURA EN LA EEPROM
.*****INTCON
RBIF EQU 0 ;BANDERA DE INTERRUPCIÓN POR CAMBIO EN RB <7:4>
INTF EQU 1 ;BANDERA DE INTERRUPCIÓN INT
TOIF EQU 2 ;BANDERA DE INTERRUPCIÓN TMR0
RBIE EQU 3 ;HABILITACIÓN INTERRUPCIÓN RB <7:4>
INTE EQU 4 ;HABILITACIÓN INTERRUPCIÓN INT
TOIE EQU 5 ;HABILITACIÓN INTERRUPCIÓN TMR0
PEIE EQU 6 ;HABILITACIÓN INTERRUPCIÓN POR MÓDULOS PERIFÉRICOS
GIE EQU 7 ;HABILITACIÓN GLOBAL DE INTERRUPCIONES
.*****
B0 EQU 0 ;BIT 0
B1 EQU 1 ;BIT 1
B2 EQU 2 ;BIT 2
B3 EQU 3 ;BIT 3
B4 EQU 4 ;BIT 4
B5 EQU 5 ;BIT 5
B6 EQU 6 ;BIT 6
B7 EQU 7 ;BIT 7

.*****DEFINICIÓN DE CONSTANTES
,
VAL1 EQU 90H ;CONSTANTE DE TEMPORIZADORES
VAL2 EQU 90H ;CONSTANTE DE TEMPORIZADORES
VAL3 EQU 5H ;CONSTANTE DE TEMPORIZADORES

.*****CONSTANTES PARA EL CONTROL DEL LCD
,
LCD_E EQU 2 ;ENABLE EN RA2
LCD_RW EQU 1 ;R / W EN RA1
LCD_RS EQU 0 ;RS EN RA0

.*****CONSTANES CONFIGURACIÓN LCD
,
LCD_FSI EQU B'00110000' ;0-0-1-DL-N-F-0-0 --> INICIAL -- DL BUS DE 8 BITS / N DOS LINEAS / F
CARACTERES 5X10 DOTS
LCD_FS EQU B'00111100' ;0-0-1-DL-N-F-0-0 --> DL BUS DE 8 BITS / N DOS LINEAS / F CARACTERES 5X10
DOTS
LCD_DOF EQU B'00001000' ;0-0-0-0-1-D-C-B --> D PANTALLA ON / C CURSOR ACTIVO / B
PARPADEO CURSOR
LCD_DON EQU B'00001111' ;0-0-0-0-1-D-C-B --> D PANTALLA ON / C CURSOR ACTIVO / B
PARPADEO CURSOR
LCD_CLR EQU B'00000001' ;0-0-0-0-0-0-1 --> BORRA DISPLAY CURSOR EN PRIMERA
POSICIÓN DDRAM
.*****
LCD_MS0 EQU B'00000110' ;0-0-0-0-0-1-I/D-S --> I/D INCREMENTA DIRECCIÓN CURSOR / S
DESPLAZA VISUALIZACIÓN
LCD_MS1 EQU B'00000111' ;0-0-0-0-0-1-I/D-S --> I/D INCREMENTA DIRECCIÓN CURSOR / S
DESPLAZA VISUALIZACIÓN
.*****
LCD_HOM EQU B'00000010' ;0-0-0-0-0-1-0 --> CURSOR POSICIÓN DE INICIO.
LCD_SH0 EQU B'00010000' ;0-0-0-1-S/C-R/L-0-0 --> S/C DESPLAZA VISUALIZACIÓN / S/C'
DESPLAZA CURSOR / R/L DESPLAZA DR
LCD_SH1 EQU B'00010100' ;0-0-0-1-S/C-R/L-0-0 --> S/C DESPLAZA VISUALIZACIÓN / S/C'
DESPLAZA CURSOR / R/L DESPLAZA DR
LCD_SH2 EQU B'00011000' ;0-0-0-1-S/C-R/L-0-0 --> S/C DESPLAZA VISUALIZACIÓN / S/C'
DESPLAZA CURSOR / R/L DESPLAZA DR
LCD_SH3 EQU B'00011100' ;0-0-0-1-S/C-R/L-0-0 --> S/C DESPLAZA VISUALIZACIÓN / S/C'
DESPLAZA CURSOR / R/L DESPLAZA DR

.*****DEFINICIÓN DE DIRECCIONES EEPROM
,
STAEE EQU 00H ;/B7=NOP/B6=NOP/B5=NOP/B4=OFF*LCD/B3=LIGHT*LCD/B2=ALAR*ON:OFF/B1-
B2=CLAVOPC

.*****DEFINICIÓN DE MACROS
,
```

```

#define BANK1 BSF STATUS,5
#define BANK0 BCF STATUS,5

;*****INICIO DEL PROGRAMA*****
;INICIO DEL PROGRAMA
ORG 00H
CALL CONFIGU
GOTO INICIO

;*****RUTINA DE INTERRUPCION ---> 004H
ORG 04H
CALL SAVE1 ;RUTINA SALVA STATUS / W_REG
;GOTO M_INT ;RUTINA DE MENU SOBRE INTERUPCIONES
;OUTINT GOTO NOBLOCK ;SE ASEGURA DE NO BLOQUEOS EN SENSORES EN UNA
INTERRUPCION

CALL SAVE2 ;REGRESA STATUS / W_REG A ESTADO ORIGINAL
RETFIE

;*****CONFIGURACION DE PUERTOS
ORG 10H
CONFIGU BANK1
MOVLW 00H ;SALIDA DE DATOS POR LOS PINES B7 B6 B5 B4 B3 B2 B1
MOVWF TRISB
MOVLW 'B'00010000' ;SALIDAS A3 A2 A1 A0 ENTRADA A4
MOVWF TRISA ;
BANK0
RETURN

;*****PROGRAMA DE INICIO
INICIO BCF PTA, LCD_E ;LCD DISABLE
CLRF REGT1 ;BORRADO DE REGISTRO TEMPORAL
CLRF MENU ;BORRADO DE REGISTRO MENU

;*****INSTRUCCION DE INICIALIZACION DE LCD
CALL LCD_INI ;RUTINA DE INICIALIZACION DEL LCD
MOVLW LCD_FS ;FUNTION SET LCD
CALL LCD_INS ;RUTINA DE ENVIO DE INSTRUCCION
MOVLW LCD_DOF ;APAGAR LCD
CALL LCD_INS ;RUTINA DE ENVIO DE INSTRUCCION
MOVLW LCD_DON ;ENCENDER LCD
CALL LCD_INS ;RUTINA DE ENVIO DE INSTRUCCION
MOVLW LCD_CLR ;BORRAR PANTALLA LCD
CALL LCD_INS ;RUTINA DE ENVIO DE INSTRUCCION
MOVLW LCD_MS0 ;MODE SET
CALL LCD_INS ;RUTINA DE ENVIO DE INSTRUCCION
GOTO MENS1 ;SALTA A RUTINA DE MENU

;*****MENSAJE UNO
MENS1 MOVLW LCD_CLR ;BORRAR PANTALLA LCD
CALL LCD_INS ;RUTINA DE ENVIO DE INSTRUCCION
;MOVLW LCD_SH1 ;SHIFT CURSOR DISPLAY
;CALL LCD_INS ;RUTINA DE ENVIO DE INSTRUCCION
;*****PRIMERA LINEA
MOVLW 088H ;DIRECCION PRIMER RENGLON B= 1000 0000
CALL LCD_INS ;ENVIO INSTRUCCIÓN
;*****
MOVLW "M"
CALL LCD_DAT
MOVLW "S"
CALL LCD_DAT
MOVLW "0"

```

```
CALL LCD_DAT
GOTO MENSJ ;MENSAJE GLOBAL

;*****MENSAJE GLOBAL

MENSJ MOVLW "I"
CALL LCD_DAT
MOVLW "N"
CALL LCD_DAT
MOVLW "G"
CALL LCD_DAT
MOVLW "E"
CALL LCD_DAT
MOVLW "L"
CALL LCD_DAT
;*****SEGUNDA LINEA
MOVLW 0COH ;DIRECCION SEGUNDO RENGLON B= 1100 0000
CALL LCD_INS ;ENVIO INSTRUCCIÓN
;*****MENSAJE SEGUNDA LINEA
MOVLW "E"
CALL LCD_DAT
MOVLW "L"
CALL LCD_DAT
MOVLW "E"
CALL LCD_DAT
MOVLW "C"
CALL LCD_DAT
MOVLW "T"
CALL LCD_DAT
MOVLW "R"
CALL LCD_DAT
MOVLW "O"
CALL LCD_DAT
MOVLW "N"
CALL LCD_DAT
MOVLW "I"
CALL LCD_DAT
MOVLW "C"
CALL LCD_DAT
MOVLW "A"
CALL LCD_DAT
MOVLW LCD_SH1 ;SHIFT CURSOR DISPLAY
CALL LCD_INS ;RUTINA DE ENVIO DE INSTRUCCION
STOP BTFSS PTA, B4
GOTO STOP
GOTO MENS1 ;MENU

;*****RUTINA DE ENABLE LCD

LCD_EN NOP
BSF PTA, LCD_E ;SE COLOCA EN ALTO EL PIN ENABLE DEL LCD
CALL RETGR
BCF PTA, LCD_E ;SE DESHABILITA EL LCD
RETURN

;*****RUTINA DE CHEQUEO ESTADO LCD - BUSY

LCD_BSY BSF PTA, LCD_RW ;SE COLOCA LCD EN MODO LECTURA
BCF PTA, LCD_RS ;MODO INSTRUCCION Y CONTROL DEL LCD
BANK1
MOVLW 0FFH ;ENTRADA DE DATOS POR LOS PINES B3 B2 B1 B0
MOVWF TRISB
BANK0
BSF PTA, LCD_E ;LCD ENABLE
BUSY_L BTFSC PTB, B7 ;CHEQUEO DEL BIT BUSY EN DB7 DEL LCD
GOTO BUSY_L ;LCD OCUPADO
BCF PTA, LCD_E ;LCD DESACTIVADO
BANK1
```

```

MOVlw 00H           ;SALIDA DE DATOS POR LOS PINES B7 B6 B5 B4 B3 B2 B1 B0
MOVwf TRISB
BANK0
BCF PTA, LCD_RW ;SE COLOCA LCD EN MODO ESCRITURA
RETURN
    
```

,*****RUTINA DE ENVIO INSTRUCCIONES AL LCD

```

LCD_INS BSF PTA, LCD_RS ;MODO INSTRUCCION Y CONTROL DEL LCD
BSF PTA, LCD_RW ;MODO ESCRITURA DEL LCD
BCF PTA, LCD_E ;LCD DISABLE
MOVWF LCD4B
CALL LCD_BSY          ;LLAMADO A RUTINA BUSY LCD
MOVF LCD4B, W
MOVWF PTB              ;ENVIO DE DATOS AL PUERTO
BCF PTA, LCD_RS ;MODO INSTRUCCION Y CONTROL DEL LCD
BCF PTA, LCD_RW ;MODO ESCRITURA DEL LCD
CALL LCD_EN
RETURN
    
```

,*****RUTINA DE ENVIO DE DATOS AL LCD

```

LCD_DAT BCF PTA, LCD_RS ;MODO INSTRUCCION Y CONTROL DEL LCD
BSF PTA, LCD_RW ;MODO ESCRITURA DEL LCD
BCF PTA, LCD_E ;LCD DISABLE
MOVWF LCD4B
CALL LCD_BSY          ;LLAMADO A RUTINA BUSY LCD
MOVF LCD4B, W
MOVWF PTB              ;ENVIO DE DATOS AL PUERTO
BSF PTA, LCD_RS ;MODO INSTRUCCION Y CONTROL DEL LCD
BCF PTA, LCD_RW ;MODO ESCRITURA DEL LCD
CALL LCD_EN
RETURN
    
```

,*****RUTINA DE ENVIO INSTRUCCIONES INICIALES AL LCD

```

LCD_INN BSF PTA, LCD_RS ;MODO INSTRUCCION Y CONTROL DEL LCD
BSF PTA, LCD_RW ;MODO ESCRITURA DEL LCD
BCF PTA, LCD_E ;LCD DISABLE
MOVWF LCD4B
MOVF LCD4B, W
MOVWF PTB              ;ENVIO DE DATOS AL PUERTO
BCF PTA, LCD_RS ;MODO INSTRUCCION Y CONTROL DEL LCD
BCF PTA, LCD_RW ;MODO ESCRITURA DEL LCD
CALL LCD_EN
RETURN
    
```

,*****RUTINA DE INICIALIZACION DE LCD

```

LCD_INI CALL RETGR      ;RETARDO DE MAS DE 15 MS ESTABILIZACION DEL LCD
MOVlw LCD_FSI           ;FUNTION SET INICIALIZACION LCD
CALL LCD_INS             ;RUTINA DE ENVIO DE INSTRUCCION AL LCD
CALL RETMD               ;RETARDO DE MAS DE 4 MS
MOVlw LCD_FSI           ;FUNTION SET INICIALIZACION LCD
CALL LCD_INS             ;RUTINA DE ENVIO DE INSTRUCCION AL LCD
CALL RETMD               ;RETARDO DE MAS DE 100 US
MOVlw LCD_FSI           ;FUNTION SET INICIALIZACION LCD
CALL LCD_INS             ;RUTINA DE ENVIO DE INSTRUCCION AL LCD
CALL RETMD
RETURN
    
```

,*****RUTINA SAVE 1 SALVA STATUS / W_REG

```

SAVE1 MOVWF SWREG        ;SALVAR W
SWAPF STATUS,W           ;INVIERTESTATUS ==> W ==>BA
MOVWF SAVSTA             ;GUARDA STATUS==>W==>SAVSTA
SWAPF SAVSTA,F           ;INVIERTESTATUS==>SAVSTA==>AB
RETURN
    
```

,*****RUTINA SAVE 2 SALVA STATUS / W_REG

```
SAVE2 SWAPF SAVSTA, F      ;INVIERTE STATUS==>BA
SWAPF SAVSTA, W      ;INVIERTE STATUS==>AB==>W
MOVWF STATUS          ;W==>STATUS
SWAPF SWREG, F        ;RESTAURA EL SATATUS
SWAPF SWREG, W        ;RESTAURA EL REGISTRO W
BCF    INTCON, INTF    ;BORRAR FLAG DE INTERRUPCION EXTERNA
BCF    INTCON, TOIF    ;BORRAR BANDERA DE INT TMR0
RETURN

*****RUTINA DE RETARDO*****
;

*****RETARDOS*****;RETARDO GRANDE

RETGR MOVLW VAL1
MOVWF TMP1
RGRV2 MOVLW VAL2
MOVWF TMP2
RGRV3 MOVLW VAL3
MOVWF TMP3
RGRV0 DECFSZ TMP3, F
GOTO RGRV0
DECFSZ TMP2, F
GOTO RGRV3
DECFSZ TMP1, F
GOTO RGRV2
RETURN

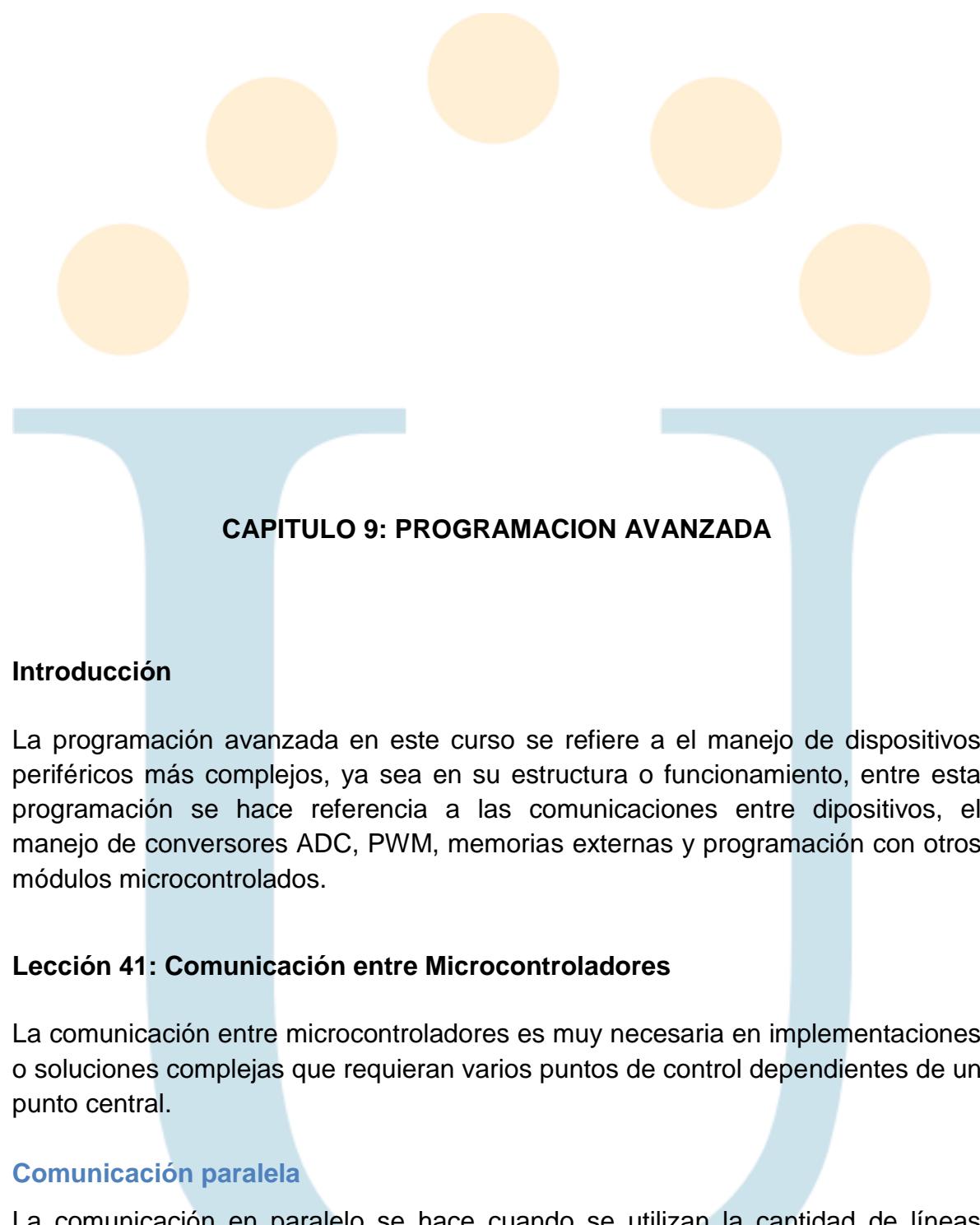
*****;RETARDO MEDIANO

RETMD MOVLW VAL2
MOVWF TMP2
RMDV3 MOVLW VAL3
MOVWF TMP3
RMDV0 DECFSZ TMP3, F
GOTO RMDV0
DECFSZ TMP2, F
GOTO RMDV3
RETURN

*****;RETARDO PEQUEÑO

RETPQ MOVLW VAL3
MOVWF TMP3
RPQV0 DECFSZ TMP3, F
GOTO RPQV0
RETURN

*****;FIN DEL PROGRAMA*****
;
END
```



CAPITULO 9: PROGRAMACION AVANZADA

Introducción

La programación avanzada en este curso se refiere a el manejo de dispositivos periféricos más complejos, ya sea en su estructura o funcionamiento, entre esta programación se hace referencia a las comunicaciones entre dipositivos, el manejo de conversores ADC, PWM, memorias externas y programación con otros módulos microcontrolados.

Lección 41: Comunicación entre Microcontroladores

La comunicación entre microcontroladores es muy necesaria en implementaciones o soluciones complejas que requieran varios puntos de control dependientes de un punto central.

Comunicación paralela

La comunicación en paralelo se hace cuando se utilizan la cantidad de líneas equivalente a la longitud del registro de información, por ejemplo, si se requiere una transmisión de datos paralela de 8 bits, se requiere entonces la disposición de un puerto de 8 pines para transmitir el registro de una sola vez. Otra modalidad de conexión en paralelo que sigue este mismo principio, ocurre en el puerto LPT o

puerto de impresión, antiguo puerto en donde se conectaban las impresoras, este puerto es de 8 bits, puede ser utilizado para la transmisión de datos de forma bidireccional. Esta modalidad es muy utilizada para comunicación entre microcontroladores o del microcontrolador al PC.

La ventaja más sobresaliente de este tipo de comunicación es la velocidad de transferencia de datos, pudiendo transferir la totalidad del registro en un solo ciclo o pulso sincrónico de reloj, la desventaja es la utilización excesiva de pines, recurso muy valioso en las implementaciones.

Comunicación serial RS-232

Esta interfaz propia de los PC para conectar el mouse serial, es utilizada aún por gran variedad de dispositivos como los PLC, tarjetas de desarrollo, microcontroladores, etc. Consiste básicamente en la transmisión de datos de forma serial, es decir se transmite una trama de bits seguidos uno tras otro hasta completar la totalidad del registro ya sea de 8 bits, 10 bits o más. La transmisión de datos puede ser de manera síncrona o asíncrona, en el caso de ser síncrona, se utiliza un pin extra para generar los pulsos de sincronización de reloj, en el caso de la asíncrona se utilizan estados predeterminados de bit de parada y arranque y tiempo de los mismos para indicar el inicio y terminación de un bit.

Comunicación IIC

La comunicación IIC, es muy similar a la serial síncrona, solo que más sofisticada y con capacidad de transmitir más bits por segundo. Muchos microcontroladores implementan en módulos internos esta interfaz de manera que implementar esta transmisión entre dos microcontroladores o un micro y el PC es relativamente sencillo.

Objetivos:

- Verificar la comunicación serial síncrona y asíncrona
- Comprobar los algoritmos de comunicación serial

En vista de que algunos de los elementos de la familia PIC16CXXX no poseen periféricos de comunicación serial, este capítulo hará referencia al desarrollo del algoritmo como tal simulando los pines de comunicación serial con puertos del microcontrolador.

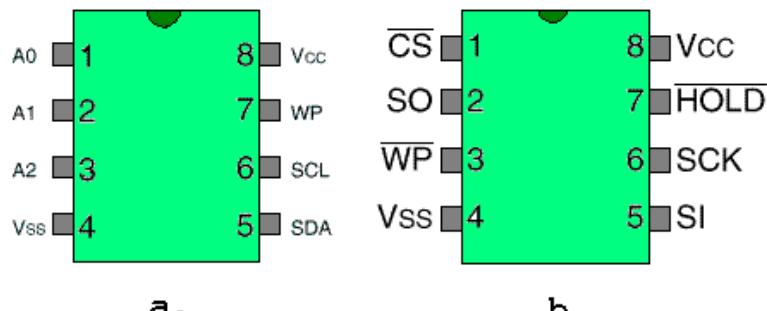
Comunicación serial síncrona: La comunicación síncrona se caracteriza porque los pulsos de sincronización deben ser transmitidos a lo largo de la línea de comunicación entre el transmisor y el receptor. Dentro de los varios tipos de

comunicación serial síncrona vamos a notar el protocolo I²C ó de dos hilos y el protocolo SPI ó de tres hilos.

Nomenclatura de los pines de comunicación (síncronos)

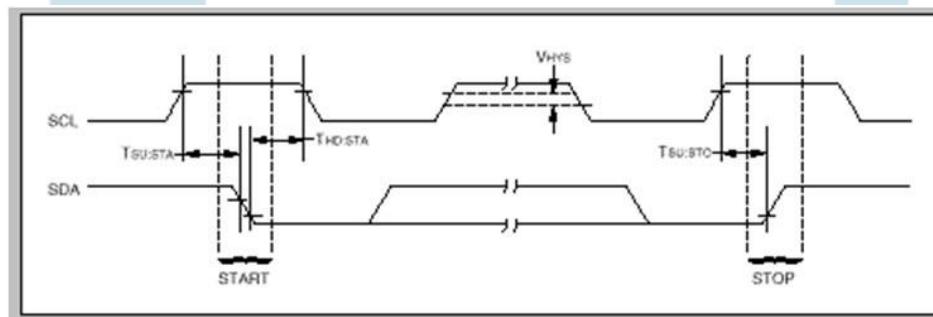
	Línea(s) de datos	Línea de reloj
I ² C	SDA (serial data)	SCL
SPI	SO (serial out), SI (serial in)	SCK

Figura 146. Disposición típica de pines en dispositivos síncronos (a) I²C, (b) SPI



I²C: El bus I²C es un bus diseñado para que sobre éste puedan colocarse varios dispositivos dentro de la misma tarjeta electrónica (comunicación multipunto), cada dispositivo tendrá una dirección lógica asignada físicamente mediante los pines A0, A1 y A2 de acuerdo al nivel lógico al que estos sean alambrados. ver estos pines en la figura 4.7.1 a.

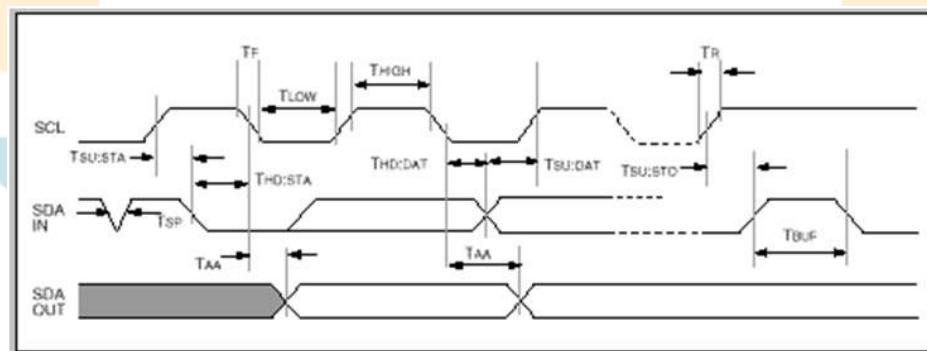
Figura 147. Bits de START y STOP del protocolo I²C



En las figuras anteriores se observa la forma en que las señales SCL y SDA deben ser manejadas. Para iniciar la comunicación sobre un dispositivo I²C debe realizarse la secuencia denominada bit de START que consiste en pasar la línea de datos SDA de nivel alto a bajo mientras que la línea SCL permanece en alto.

Para la culminar la comunicación con el dispositivo I²C debe ejecutarse la secuencia denominada bit de STOP la cual consiste en pasar la línea de datos SDA de nivel bajo a alto mientras que la línea de reloj SCL permanece en alto. Un bit de datos es aceptado por el dispositivo mientras que sobre la línea de datos SDA permanece el nivel adecuado al bit en cuestión, y sobre la línea de reloj SCL se lleva a cabo un pulso, es decir, el paso de nivel de bajo a alto y luego de alto a bajo. Los tiempos implicados en esta secuencia dependen básicamente del fabricante del dispositivo.

Figura 148. Temporización en el bus I²C



SPI: El bus SPI es un bus diseñado para que sobre éste se coloque un dispositivo maestro y un dispositivo esclavo (comunicación punto a punto. Con relación al bus I²C podemos notar que éste soporta mayor velocidad de comunicación.

Figura 149. Entrada de datos a dispositivo SPI

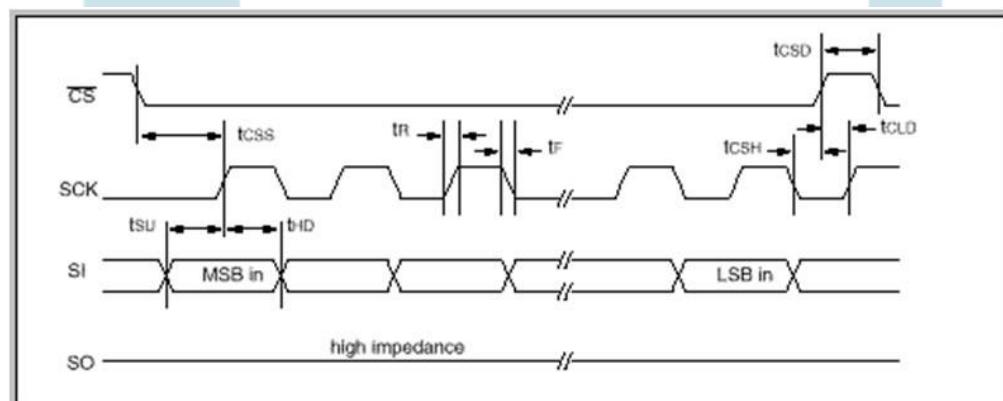
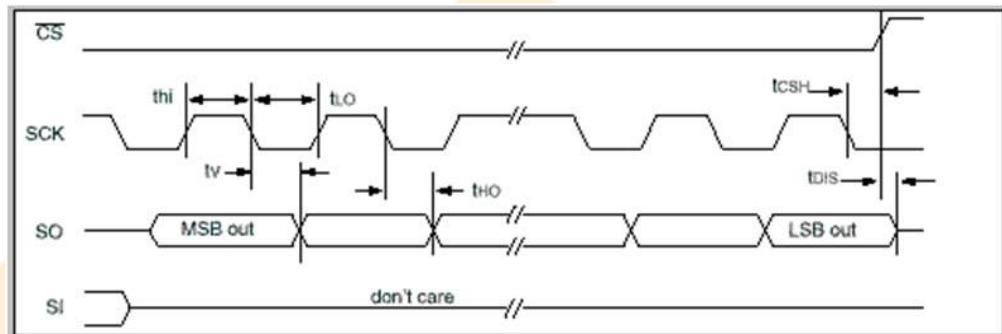
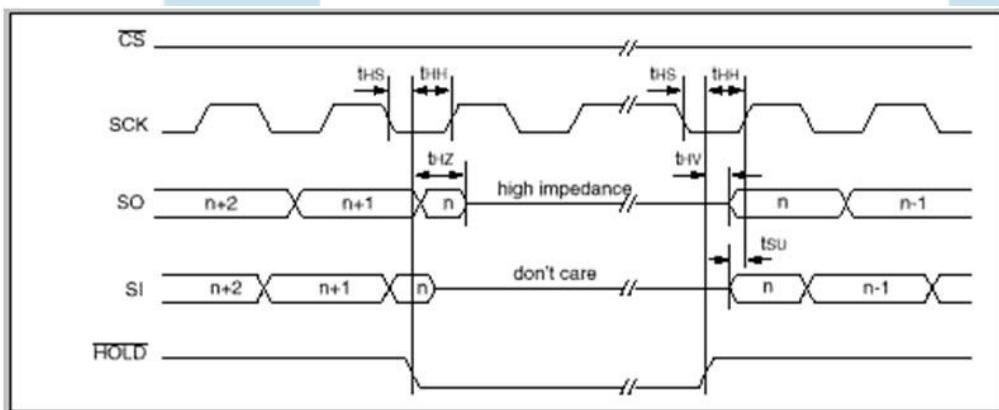


Figura 150. Salida de datos de dispositivo SPI



El dispositivo SPI posee como una línea de selección CS la cual debe pasar al nivel lógico activo (en este caso bajo) para poder realizar la comunicación con el dispositivo. Desde este punto de vista podríamos colocar sobre un bus de este tipo varios dispositivos, pero utilizando un dispositivo decodificador adicional. Otra línea podemos observar es la línea HOLD la cual permite al procesador detener momentáneamente la comunicación.

Figura 151. Mapa de tiempos



Ejemplo de transmisión RS-232 con PIC

```

title "PIC16F84A LCD PROGRAM"
LIST P=16F84A,F=INHX32
#include <P16F84A.INC>

;PUERTO B=>RB0 TX RS232
;TXRS232 9200BAUDIOS=>108.7USEG
;DEFINICION REGISTROS FSR
STATUS EQU 03H
OPCION EQU 81H
INTCON EQU 0BH
PCL EQU 02H

```

```
PTA EQU 05H
PTB EQU 06H
TRISA EQU 85H
TRISB EQU 86H
;DEFINICION DE RPG
CONT1 EQU 0CH
RXRS2 EQU 0DH
AUXRS EQU 0EH
ROTA EQU 0FH
TMP1 EQU 10H
TMP2 EQU 11H
TMP3 EQU 12H
SAVSTA EQU 13H
SWREG EQU 14H
;DEFINICION DE BITS
W EQU 0
F EQU 1
C EQU 0
RXR EQU 7
INTDEG EQU 6
INTF EQU 1
INTE EQU 4
GIE EQU 7
;DEFINICION DE CONSTANTES
VAL1 EQU 1FH ;1F
VAL2 EQU 0FH ;0F
VAL21 EQU 02H
VAL3 EQU 07H ;70 0E
VAL4 EQU 05H ;50 0A
VAL5 EQU 04H ;40 08
;MACROS
#define BANK1 BSF STATUS,5
#define BANK0 BCF STATUS,5
;INICIO DEL PROGRAMA
    ORG 00H
    CALL CONFIG ;CONFIGURACION DE PUERTOS
    CALL CONFINT ;CONFIGURACION DE INTERRUPCION PUERTO=>PTB0
    GOTO INICIO
;PROGRAMA DE INTERRUPCION PARA RECIBIR DATO DEL TX
    ORG 04H
    CALL SAVE1
    CALL RX
    CALL DATO
EROR    CALL SAVE2
    RETFIE
    ORG 10H
;CONFIGURACION DE PUERTOS
CONFIG BANK1
    MOVLW 01H
    MOVWF TRISB
    MOVLW 10H
    MOVWF TRISA
    BANK0
    RETURN
;RUTINA DE CONFIGURACION DE INTERRUPCION
CONFINT BANK1
    BCF OPCION, INTDEG ;INT FLANCO DE BAJADA
    BANK0
    BCF INTCON, INTF ;BORRA FLAG DE INT
    BSF INTCON, INTE ;HABILITACION DE INT POR PTB
    BSF INTCON, GIE ;HABILITACION GLOBAL DE INT
    RETURN
;RUTINA DE INICIO Y CONTROL DE TX RS232
INICIO CLRFL ROTA
SEC    MOVF ROTA, W
        ANDLW 0F0H
        MOVWF PTB
        CALL RET
        MOVF ROTA, W
```

```
ANDLW 0FH
MOVWF PTA
CALL RET
GOTO SEC
;RUTINA DE RECEPCION DE DATOS VIA RS232
RX CALL RET2B ;RETARDO DE 1/2 DE BIT
    BTFSC PTB, 0
    GOTO EROR
    CLRF RXRS2
    MOVLW 08H ;CANTIDAD DE BIT A RECIBIR
    MOVWF CONT1
RXOTR CALL RET1B ;RETARDO DE 1 BIT
    BTFSS PTB, 0
    GOTO RX_0
    GOTO RX_1
RXOTR2 DECFSZ CONT1, F
    GOTO RXOTR
    CALL RET1B ;RETARDO DE 1 BIT
    BTFSC PTB, 0
    RETURN
    CLRF RXRS2
    RETURN
;RUTINA DE ALMACENAMIENTO DEL DATO
DATO MOVF RXRS2, W
    MOVWF ROTA
    RETURN
;RUTINA EL BIT ES CERO
RX_0 BCF STATUS, C
    RRF RXRS2, F
    GOTO RXOTR2
;RUTINA EL BIT ES UNO
RX_1 BSF STATUS, C
    RRF RXRS2, F
    GOTO RXOTR2
;RUTINA SALVAR DATOS EN STATUS Y W
SAVE1 MOVWF SWREG ;SALVAR W
    SWAPF STATUS, W ;INVIERTESTATUS ==> W ==>BA
    MOVWF SAVSTA ;GUARDA STATUS==>W==>SAVSTA
    SWAPF SAVSTA, F ;INVIERTESTATUS==>SAVSTA==>AB
    RETURN
;RUTINA DEVOLVER DATOS DE STATUS Y W
SAVE2 SWAPF SAVSTA, F ;INVIERTE STATUS==>BA
    SWAPF SAVSTA, W ;INVIERTE STATUS==>AB==>W
    MOVWF STATUS ;W==>STATUS
    SWAPF SWREG, F ;RESTAURA EL SATATUS
    SWAPF SWREG, W ;RESTAURA EL REGISTRO W
    BCF INTCON, INTF ;BORRA EL FLAG DE INT
    RETURN
;RUTINA DE RETARDO VISUAL
RET MOVLW VAL3
    MOVWF TMP1
UNO MOVLW VAL4
    MOVWF TMP2
DOS MOVLW VAL5
    MOVWF TMP3
TRES DECFSZ TMP3, F
    GOTO TRES
    DECFSZ TMP2, F
    GOTO DOS
    DECFSZ TMP1, F
    GOTO UNO
    RETURN
;1/2BIT+15=7+1BIT+6=6+1BIT+6=6+1BIT+6=.....
;PROGRAMA DE RETARDO DE RS232 1 BIT 96,7 USEG (108.7-12)
RET1B MOVLW VAL1
    MOVWF TMP1
RET1B1 DECFSZ TMP1, F
    GOTO RET1B1
    RETURN
```

```
;PROGRAMA DE RETARDO DE RS232 1/2 BIT 46,85 USEG (108.7/2)-15= 39.35 USEG
RET2B  MOVLW VAL2
       MOVWF TMP1
RET2B1 DECFSZ TMP1, F
       GOTO  RET2B1
       RETURN
;RET2B  MOVLW VAL1
;       MOVWF TMP1
;UNO1   MOVLW VAL2
;       MOVWF TMP2
;DOS1   MOVLW VAL21
;       MOVWF TMP3
;TRES1  DECFSZ TMP3, F
;       GOTO  TRES1
;       DECFSZ TMP2, F
;       GOTO  DOS1
;       DECFSZ TMP1, F
;       GOTO  UNO1
;       RETURN
END
```

Lección 42: Manejo de ADC, PWM y módulos especiales

En muchas ocasiones se hace necesario convertir una señal analógica a digital, por lo que se hace uso de conversores A/D o módulos internos en los micros del tipo A/D. Igualmente se hace necesario el control de motores o servomotores para lo cual se utiliza el módulo PWM.

ADC

Los módulos ADC permiten la conversión de señales análogas a digitales de manera eficiente, su precisión típica para aplicaciones de aprendizaje está entre 8, 10, 12 y 16 bits.

PWM

Los módulos PWM se utilizan para el control de motores DC, servomotores y generar señales moduladas en ancho de pulso.

Ejemplo de cambio de luminosidad de un LED con MSP430

```
#include "msp430x20x3.h"141
```

¹⁴¹ Aparicio O. <http://todomcu.scienceontheweb.net>

```

#define midram      240h      /* Lugar donde comenzara la PILA/
main
;-----[ORG] 0F800h ; Direccion de inicio el la flash
;-----[RESET]
MOV.W #midram,SP ; Inicio del apuntador a la pila
MOV.W #WDTPW+WDTTMSEL+WDTIS0+WDTSSEL,&WDTCTL ; Modo intervalo cada
BIS.B #WDTIE,&IE1 ; Interrupcion del modo intervalo permitida
MOV.B &CALBC1_1MHZ,&BCSCTL1 ; Calibracion del
MOV.B &CALDCO_1MHZ,&DCOCTL ; oscilador a 1MHz
MOV.B #LFXT1S_2,&BCSCTL3 ; VLO (12KHz) para ACLK
BIS.B #BIT2,&P1DIR
BIS.B #BIT2,&P1SEL
MOV.W #120,&TACCR0
MOV.W #OUTMOD_6,&CCTL1
MOV.W #10,&TACCR1
MOV.W #TASSEL_1+MC_1,&TACTL
MOV.W #CPUOFF+GIE,SR ; Apaga CPU y permite las interrupciones
; de manera global
NOP

;-----[WDT_ISR; Rutina de Servicio a la Interrupcion para WDT]
;-----[ADD.W #10,&TACCR1
reti

;-----[Vectores de Interrupcion y Reset]
;-----[ORG] 0FFEh ; Vector de reset
DW RESET ; Etiqueta para Reset (Inicio del programa)
ORG 0FFF4h ; Vector para la interrupcion del WDT
DW WDT_ISR ; Etiqueta de la rutina de servicio a la
; interrupcion
END main
    
```

Ejemplo de control de motor paso a paso con PIC

En las siguientes líneas se presenta un ejemplo sencillo del control de motores paso a paso con PIC

```

TITLE "PIC16F84A EEPROM PROGRAM"
LIST P=16F84A,F=INHX32
#INCLUDE <P16F84A.INC>
;INGENIERO ELECTRONICO HECTOR URIEL VILLAMIL GONZALEZ
;PINES DE SALIDA DE CONTROL DEL MOTOR DE PASO - PASO ES PUERTO B EN PINES B0 B1 B2 B3 LOS DEMAS
NO
;SE UTILIZAN.....PINES DE CONTROL DE SECUENCIAS MIENTRAS PULSE LOS PULSADORES OCURRIRA UNA DE
LAS
;TRES SECUENCIAS !!!PASOS SIMPLES!!! !!!PASOS DOBLES!!! !!!MEDIOS PASOS!!!
;ENTRADA DE PULSADORES PUERTO A EN PINES A0 A1 A2
PCL EQU 02H ;PROGRAM COUNTE PARTE BAJA
STATUS EQU 03H ;REGISTRO DE ESTADO DEL PIC
PTA EQU 05H ;PUERTO A DEL PIC
    
```

```

PTB EQU 06H ;PUERTO B DEL PIC
EEDATA EQU 08H ;DATO IN/OUT EN EEPROM
EEADR EQU 09H ;DIRECCION DE IN/OUT DE EEPROM
INTCON EQU 0BH ;REGISTRO DE INTERRUPCIONES
OPCION EQU 81H ;OPCIONES DE CONFIGURACION INT TMP PULLUP PRESCALER
TRISA EQU 85H ;/RA7-RA6-RA5=NOP/RA4=RXRS(CABL-ESCL)/RA3=TEST SENSORES/RA2-RA1-
RA0=TECLD
TRISB EQU 86H ;/RB7=TXRS232/RB6=CLK2/RB5=CLK1/RB4-RB3-RB2-RB1=(MULTIPLX-'D')/RB0=INT/
EECON1 EQU 88H ;B4=EIF/FIN WR/*B3=WRERR/ERROR WR/*B2=WREN/PERMISO WR/*B1/WR/*B0/RD/
EECON2 EQU 89H ;REG SEGURIDAD PROCESO EEPROM
;DEFINICION REGISTROS FSR
CONT EQU 0CH
SEC EQU 0DH
TMP1 EQU 0EH
TMP2 EQU 0FH
TMP3 EQU 10H
;DEFINICION DE BITS
W EQU 0 ;REGISTRO DE TRABAJO
F EQU 1 ;REGISTRO
C EQU 0 ;FLAG DE CARRY
Z EQU 2 ;FLAG DE CERO
RD EQU 0 ;RD=1 CICLO DE LECTURA DE LA EEPROM
WR EQU 1 ;WR=1 CICLO DE ESCRITURA DE LA EEPROM
WREN EQU 2 ;WREN=1 AUTORIZA PERMISO DE ESCRITURA EN LA EEPROM
B0 EQU 0 ;BIT 0
B1 EQU 1 ;BIT 1
B2 EQU 2 ;BIT 2
B3 EQU 3 ;BIT 3
B4 EQU 4 ;BIT 4
B5 EQU 5 ;BIT 5
B6 EQU 6 ;BIT 6
B7 EQU 7 ;BIT 7
;DEFINICION DE CONSTANTES
VAL1 EQU 025H
VAL2 EQU 010H
VAL3 EQU 010H
;DEFINICION DE DIRECCIONES EEPROM
STAEE EQU 00H ;/B7=NOP/B6=NOP/B5=NOP/B4=OFF*LCD/B3=LIGHT*LCD/B2=ALAR*ON:OFF/B1-
B2=CLAVOPC
;MACROS
#define BANK1 BSF STATUS, 5
#define BANK0 BCF STATUS, 5
;INICIO DEL PROGRAMA
;INICIO DEL PROGRAMA
    ORG 00H
    CALL CONFIG
    GOTO INICIO
    ORG 10H
;CONFIGURACION DE PUERTOS
CONFIG BANK1
    MOVLW 00H ;SALIDA DE DATOS POR LOS PINES B7 B6 B5 B4 B3 B2 B1 DATOS CONSIGNADOS EN
LA EEPROM
    MOVWF TRISB
    MOVLW 0FH ;ENTRADA DE DATOS PUERTOS A2 A1 A0
    MOVWF TRISA ;====> A4 =AUTORIZACION DE ESCRITURA EN EEPROM
    BANK0
    RETURN
;PROGRAMA DE INICIO
INICIO CLRF SEC
    MOVF PTA, W ;MOVER EL DATO DEL PUERTO AL W-REGISTRO
    XORLW 01H
    BTFSC STATUS, Z ;¿ES EL PRIMER BOTON PRIMERA RUTINA?
    GOTO SEC1 ;RUTINA DE PASOS SIMPLES
    MOVF PTA, W ;MOVER EL DATO DEL PUERTO AL W-REGISTRO
    XORLW 02H
    BTFSC STATUS, Z ;¿ES EL SEGUNDO BOTON SEGUNDA RUTINA?
    GOTO SEC2 ;RUTINA DE PASOS DOBLES
    MOVF PTA, W ;MOVER EL DATO DEL PUERTO AL W-REGISTRO
    XORLW 04H

```

```

        BTFSC STATUS, Z      ;¿ES EL TERCER BOTON SEGUNDA RUTINA?
        GOTO SEC3            ;RUTINA DE MEDIOS PASOS
        GOTO INICIO
;RUTINA DE PASOS SIMPLES
SEC1 CLRF CONT
SEC12 INCF CONT, F
        CALL TABLA1          ;RUTINA DE VALORES DE PASOS SIMPLES
        MOVWF SEC             ;VALOR A SACAR POR EL PUERTO
        CALL OUT              ;SACAR DATOS POR PUERTO B
        MOVF CONT, W           ;VALOR DE CONTADOR DE DATOS DE TABLA
        XORLW 04H              ;VALOR DE CANTIDAD DE DATOS PARA SACAR PASOS SIMPLES
        BTFSZ STATUS, Z         ;¿TERMINO LA SECUENCIA?

;RUTINA DE PASOS DOBLES
SEC2 CLRF CONT
SEC22 INCF CONT, F
        CALL TABLA2          ;RUTINA DE VALORES DE PASOS DOBLES
        MOVWF SEC             ;VALOR A SACAR POR EL PUERTO
        CALL OUT              ;SACAR DATOS POR PUERTO B
        MOVF CONT, W           ;VALOR DE CONTADOR DE DATOS DE TABLA
        XORLW 04H              ;VALOR DE CANTIDAD DE DATOS PARA SACAR PASOS SIMPLES
        BTFSZ STATUS, Z         ;¿TERMINO LA SECUENCIA?

;RUTINA DE MEDIOS PASOS
SEC3 CLRF CONT
SEC32 INCF CONT, F
        CALL TABLA3          ;RUTINA DE VALORES DE MEDIOS PASOS
        MOVWF SEC             ;VALOR A SACAR POR EL PUERTO
        CALL OUT              ;SACAR DATOS POR PUERTO B
        MOVF CONT, W           ;VALOR DE CONTADOR DE DATOS DE TABLA
        XORLW 08H              ;VALOR DE CANTIDAD DE DATOS PARA SACAR PASOS SIMPLES
        BTFSZ STATUS, Z         ;¿TERMINO LA SECUENCIA?

;RUTINA DE VALORES DE PASOS SIMPLES
TABLA1 MOVF CONT, W      ;VALOR A SACAR DE LA TABLA==>REGISTRO CONTADOR
        ADDWF PCL, F           ;SALTO DE TABLA
        RETLW 00H
        RETLW 08H
        RETLW 04H
        RETLW 02H
        RETLW 01H

;RUTINA DE VALORES DE PASOS DOBLES
TABLA2 MOVF CONT, W      ;VALOR A SACAR DE LA TABLA==>REGISTRO CONTADOR
        ADDWF PCL, F           ;SALTO DE TABLA
        RETLW 00H
        RETLW 0CH
        RETLW 06H
        RETLW 03H
        RETLW 09H

;RUTINA DE VALORES DE MEDIOS PASOS
TABLA3 MOVF CONT, W      ;VALOR A SACAR DE LA TABLA==>REGISTRO CONTADOR
        ADDWF PCL, F           ;SALTO DE TABLA
        RETLW 00H
        RETLW 08H
        RETLW 0CH
        RETLW 04H
        RETLW 06H
        RETLW 02H
        RETLW 03H
        RETLW 01H
        RETLW 09H

;SACAR DATOS POR PUERTO B
OUT  MOVF SEC, W          ;MOVER EL DATO A SACAR AL W-REGISTRO
        MOVWF PTB              ;DATOS POR EL PUERTO B
        CALL RET                ;SUB-RUTINA DE RETARDO PARA DATOS EN EL PUERTO B
        RETURN
    
```

```
;SUB-RUTINA DE RETARDO PARA DATOS EN EL PUERTO B
RET      MOVLW  VAL1
          MOVWF  TMP1
TRES     MOVLW  VAL2
          MOVWF  TMP2
DOS      MOVLW  VAL3
          MOVWF  TMP3
UNO      DECFSZ TMP3,  F
          GOTO   UNO
          DECFSZ TMP2,  F
          GOTO   DOS
          DECFSZ TMP1,  F
          GOTO   TRES
          RETURN
END
```

Lección 43: Manejo de memorias externas

El manejo de memoria requiere conocer la hoja de datos técnica o datasheet para determinar el tamaño de la memoria, tamaño de datos, y el tipo de comunicación que implementa, son utilizadas como memoria auxiliares en los sistemas digitales de control basado en microprocesadores y microcontroladores.

Memorias paralelas

Las memorias paralelas pueden estar compuestas de uno o hasta millones de registros de almacenamiento, son utilizados como almacenamiento temporal de datos o de programa. Una aplicación especial de este tipo de memoria es en la expansión de pines o multiplexación de dato, también son utilizadas para registros de desplazamiento de bits en múltiples aplicaciones.

Memorias seriales

Las memorias seriales son un tipo de memoria bastante utilizada en aplicaciones con microcontroladores y microprocesadores, se caracterizan por contener uno o hasta millones de registros, pero que son accesibles por medio de pocos hilos mediante una transmisión en serie. Se encuentran en aplicaciones que requieren almacenar datos de control o de programa o configuración, también pueden ser utilizadas como registros de desplazamiento serial.

Ejemplo de programación utilizando memoria EEPROM con PIC

```
TITLE "PIC16F84A EEPROM PROGRAM"
LIST P=16F84A,F=INHX32
#include <P16F84A.INC>
;INGENIERO ELECTRONICO HECTOR URIEL VILLAMIL GONZALEZ
PCL    EQU    02H    ;PROGRAM COUNTE PARTE BAJA
STATUS EQU    03H    ;REGISTRO DE ESTADO DEL PIC
TRISA   EQU    85H    ;/RA7-RA6-RA5=NOP/RA4=RXRS(CABL-ESCL)/RA3=TEST      SENSORES/RA2-RA1-
RA0=TECLD
```

```

TRISB EQU 86H ;/RB7=TXRS232/RB6=CLK2/RB5=CLK1/RB4=RB3-RB2-RB1=(MULTIPLX-'D')/RB0=INT/
PTA EQU 05H ;PUERTO A DEL PIC
PTB EQU 06H ;PUERTO B DEL PIC
INTCON EQU 0BH ;REGISTRO DE INTERRUPCIONES
OPCION EQU 81H ;OPCIONES DE CONFIGURACION INT TMP PULLUP PRESCALER
EEDATA EQU 08H ;DATO IN/OUT EN EEPROM
EEADR EQU 09H ;DIRECCION DE IN/OUT DE EEPROM
EECON1 EQU 88H ;B4=EEIF/FIN WR/*B3=WRERR/ERROR WR/*B2=WREN/PERMISO WR/*B1/WR/*B0/RD/
EECON2 EQU 89H ;REG SEGURIDAD PROCESO EEPROM
;DEFINICION REGISTROS FSR
DIREEP EQU 0CH
BYTE EQU 0DH
;DEFINICION DE BITS
W EQU 0 ;REGISTRO DE TRABAJO
F EQU 1 ;REGISTRO
C EQU 0 ;FLAG DE CARRY
Z EQU 2 ;FLAG DE CERO
RD EQU 0 ;RD=1 CICLO DE LECTURA DE LA EEPROM
WR EQU 1 ;WR=1 CICLO DE ESCRITURA DE LA EEPROM
WREN EQU 2 ;WREN=1 AUTORIZA PERMISO DE ESCRITURA EN LA EEPROM
B0 EQU 0 ;BIT 0
B1 EQU 1 ;BIT 1
B2 EQU 2 ;BIT 2
B3 EQU 3 ;BIT 3
B4 EQU 4 ;BIT 4
B5 EQU 5 ;BIT 5
B6 EQU 6 ;BIT 6
B7 EQU 7 ;BIT 7
;DEFINICION DE CONSTANTES
DIREC EQU 00H
VAL1 EQU 01H
VAL2 EQU 01H
VAL3 EQU 01H
;DEFINICION DE DIRECCIONES EEPROM
STAEE EQU 00H ;/B7=NOP/B6=NOP/B5=NOP/B4=OFF*LCD/B3=LIGHT*LCD/B2=ALAR*ON:OFF/B1-
                B2=CLAVOPC
;MACROS
#define BANK1 BSF STATUS,5
#define BANK0 BCF STATUS,5
;INICIO DEL PROGRAMA
;INICIO DEL PROGRAMA
    ORG 00H
    CALL CONFG
    GOTO INICIO
    ORG 10H
;CONFIGURACION DE PUERTOS
CONFIG BANK1
    MOVLW 00H ;SALIDA DE DATOS POR LOS PINES B7 B6 B5 B4 B3 B2 B1 DATOS CONSIGNADOS EN
                LA EEPROM
    MOVWF TRISB
    MOVLW 1FH ;ENTRADA DE DATOS A ALMACENAR POR EEPROM PUERTOS A3 A2 A1 A0
    MOVWF TRISA ;====> A4 =AUTORIZACION DE ESCRITURA EN EEPROM
    BANK0
    RETURN
;PROGRAMA DE INICIO Y CONTROL DE TX RS232
INICIO CLRF BYTE
    INCF BYTE
    MOVLW DIREC ;CARGA LA DIRECCION AL W
    MOVWF DIREEP ;LA DIRECCION ES GUARDADA EN UN REGISTRO DE DIRECCION
    CALL Writte ;SE LLAMA A LA RUTINA DE ESCRITURA EN EEPROM
    BSF STATUS,5 ;SE CAMBIA A BANCO UNO DE MEMORIA
    FINW BTFSC EECON1,WR ;CONFIRAMA QUE EL BYTE ESTA ESCRITO
    GOTO FINW ;SIGUE EN CICLO HASTA QUE SE CONFIRME QUE EL DATO FUE ESCRITO
    BCF STATUS,5 ;SE CAMBIA AL BANCO DE MEMORIA CERO
    CALL READEE ;SE LLAMA A LA RUTINA DE LECTURA DE MEMORIA EEPROM
    MOVWF PTB ;SE MUEVE EL DATO LEIDO EN LA EEPROM AL PUERTO B
    GOTO INICIO ;SE INICA DE NUEVO
*****
;Writte MOVF DIREEP, W ;CARGA LA DIRECCION AL REGISTRO W

```

```

MOVWF EEADR      ;CARGA LA DIRECCION AL REGISTRO EEPROM DE DIRECCION
MOVF  BYTE, W    ;
MOVWF EEDATA     ;CARGA EN EEDATA EL DATO A ESCRIBIR EN EEPROM
BSF   STATUS, 5   ;PASA AL BANCO DE MEMORIA UNO
BSF   EECON1,WREN ;PERMISO DE ESCRITURA EN EEPROM
BCF   EECON1,EEIF ;HABILITA ESCRITURA EN EEPROM
MOVLW 55H        ;FUNCION DE SEGURIDAD PARA CARGAR DATO DE EN PROCESO
MOVWF EECON2      ;DE ESCRITURA EN EEPROM
MOVLW OAAH        ;FUNCION DE SEGURIDAD PARA CARGAR DATO DE EN PROCESO
MOVWF EECON2      ;DE ESCRITURA EN EEPROM
BSF   EECON1,WR    ;INICIO DE CICLO DE ESCRITURA EN EEPROM
BCF   EECON1,WREN  ;PROHIBE ESCRITURA EN EEPROM
CLRWDT          ;BORRA W REGITRO
ESPERA BTFSSECON1,EEIF ;ESPERA MIENTRAS CONFIRMA QUE LA ESCRITURA FUE TERMINADA
GOTO  ESPERA
BCF   EECON1,EEIF ;
BCF   EECON1,WREN ;
BCF   STATUS, 5   ;REGRESA AL BANCO DE MEMORIA CERO
RETLW 00H         ;RETORNA CON CERO EN W
;*****
READEE MOVF  DIREEP, W   ;CARGA EL VALOR DE DIRECCION A LEER EN EEPROM EN W REGITRO
MOVWF EEADR       ;SE CARGA EN REGISTRO ENCARGADO DE LA DIRECCION
BSF   STATUS, 5   ;PASA AL BANCO DE MEMORIA UNO
BSF   EECON1,RD    ;AUTORIZA LECTURA DE EEPROM
BCF   STATUS, 5   ;PASA AL BANCO DE MEMORIA CERO
MOVF  EEDATA, W    ;PASA EL DATO LEIDO AL REGISTRO EEDATA
RETURN           ;FIN DEL PROGRAMA
END

```

Lección 44: Programación con módulo microcontrolado Basic Stamp

La programación con Basic Stamp requiere descargar e instalar el PBASIC, conocer las instrucciones y aplicar el algoritmo a la solución de un problema en particular, su uso es limitado por ser un módulo microcontrolado con interprete de comandos de alto nivel, pero no deja de sorprender la creatividad y potencia de algunas aplicaciones.

Ejemplo control de un servomotor con Basic Stamp II

Por cortesía de parallax se presenta el siguiente ejemplo de control de un servomotor, que mantiene las posiciones 45, 90 y 135 grados durante un periodo de 3 segundos cada una. La señal hacia el servo se toma en el pin 14

```

counter VAR Word
PAUSE 1000
DEBUG "Posicion = 45 grados...", CR
FOR counter = 1 TO 150 ' 45 grados durante 3 segundos.
PULSOUT 14, 500
PAUSE 20
NEXT
DEBUG "Posicion = 90 grados...", CR

```

```
FOR counter = 1 TO 150 '90 grados durante 3 segundos.  
PULSOUT 14, 750  
PAUSE 20  
NEXT  
DEBUG "Posicion = 135 grados...", CR  
FOR counter = 1 TO 150 '135 grados durante 3 segundos.  
PULSOUT 14, 1000  
PAUSE 20  
NEXT  
DEBUG "Todo hecho.", CR, CR
```

Ejemplo de control de un servomotor por pulsadores

Nuevamente por cortesía de parallax, se presenta un programa de control de giro de servomotor por medio de pulsadores conectados en la entrada 3 y 4. El terminal pin 14 envía la señal PWM al servo.

```
duration VAR Word  
duration = 750  
PAUSE 1000  
DO  
IF IN3 = 1 THEN  
IF duration > 500 THEN  
duration = duration - 25  
ENDIF  
ENDIF  
IF IN4 = 1 THEN  
IF duration < 1000 THEN  
duration = duration + 25  
ENDIF  
ENDIF  
PULSOUT 14, duration  
PAUSE 10  
DEBUG HOME, DEC4 duration, " = duration"  
LOOP
```

Lección 45: Programación con módulo microcontrolador Arduino

Arduino se caracteriza por ser una plataforma libre tanto en el desarrollo de la placa como en el software, al igual que Basic Stamp es un módulo microcontrolado, por tanto requiere aprender instrucciones especiales, para el control, aunque es relativamente sencilla su programación, pierde potencialidad en por la restricción del lenguaje, pero sorprende la gran cantidad de aplicaciones realizadas.

Ejemplo de control de LED con Arduino

Por cortesía de <http://arduino.cc/es> se presenta el ejemplo de utilización de Arduino en el control de LEDs.

```
int pin2 = 2;          // PIN-es de los LED
int pin3 = 3;
int pin4 = 4;
int pin5 = 5;
int pin6 = 6;
int pin7 = 7;
int timer = 100;      // Temporizador
void setup(){ // Configuración de
    pinMode(pin2, OUTPUT); // los PIN-es como salida
    pinMode(pin3, OUTPUT);
    pinMode(pin4, OUTPUT);
    pinMode(pin5, OUTPUT);
    pinMode(pin6, OUTPUT);
    pinMode(pin7, OUTPUT);
}
void loop() {
    digitalWrite(pin2, HIGH); // Enciende y apaga
    delay(timer);           // secuencialmente los LED-s
    digitalWrite(pin2, LOW);
    delay(timer);
    digitalWrite(pin3, HIGH);
    delay(timer);
    digitalWrite(pin3, LOW);
    delay(timer);
    digitalWrite(pin4, HIGH);
    delay(timer);
    digitalWrite(pin4, LOW);
    delay(timer);
    digitalWrite(pin5, HIGH);
    delay(timer);
    digitalWrite(pin5, LOW);
    delay(timer);
    digitalWrite(pin6, HIGH);
    delay(timer);
    digitalWrite(pin6, LOW);
    delay(timer);
    digitalWrite(pin7, HIGH);
    delay(timer);
    digitalWrite(pin7, LOW);
    delay(timer);
    digitalWrite(pin6, HIGH);
    delay(timer);
    digitalWrite(pin6, LOW);
    delay(timer);
    digitalWrite(pin5, HIGH);
    delay(timer);
}
```

```
digitalWrite(pin5, LOW);
delay(timer);
digitalWrite(pin4, HIGH);
delay(timer);
digitalWrite(pin4, LOW);
delay(timer);
digitalWrite(pin3, HIGH);
delay(timer);
digitalWrite(pin3, LOW);
delay(timer);
}
```

Ejemplo de manejo de una pantalla LCD con Arduino

Finalmente por cortesía de <http://arduino.cc/es> se presenta el control de una pantalla LCD. En este ejemplo se utilizan los pines DI, RW, DB0 a DB7 y Enable.

```
int DI = 12; int RW = 11; int DB[] = {3, 4, 5, 6, 7, 8, 9, 10}; int Enable = 2;
void LcdCommandWrite(int value) {
    // aplica a todos los pins
    int i = 0;
    for (i=DB[0]; i <= DI; i++) {
        digitalWrite(i,value & 01);
        value >>= 1;
    }
    digitalWrite(Enable,LOW);
    delayMicroseconds(1);
    // envía un pulso para activar
    digitalWrite(Enable,HIGH);
    delayMicroseconds(1); // pausa 1 ms de acuerdo a la especificación
    digitalWrite(Enable,LOW);
    delayMicroseconds(1); // pausa 1 ms de acuerdo a la especificación
}
void LcdDataWrite(int value) {
    // aplica a todos los pins
    int i = 0;
    digitalWrite(DI, HIGH);
    digitalWrite(RW, LOW);
    for (i=DB[0]; i <= DB[7]; i++) {
        digitalWrite(i,value & 01);
        value >>= 1;
    }
    digitalWrite(Enable,LOW);
    delayMicroseconds(1);
    // envía un pulso para activar
    digitalWrite(Enable,HIGH);
    delayMicroseconds(1);
    digitalWrite(Enable,LOW);
    delayMicroseconds(1); // pausa 1 ms de acuerdo a la especificación
}
void setup (void) {
    int i = 0;
    for (i=Enable; i <= DI; i++) {
        pinMode(i,OUTPUT);
    }
    delay(100);
    // initiatiza LCD después de una pequeña pausa
    // necesario para los controladores LCD
    LcdCommandWrite(0x30); // function set:
        // 8-bit interface, 1 display lines, 5x7 font
    delay(64);
    LcdCommandWrite(0x30); // function set:
        // 8-bit interface, 1 display lines, 5x7 font
    delay(50);
    LcdCommandWrite(0x30); // function set:
        // 8-bit interface, 1 display lines, 5x7 font
```

```
delay(20);
LcdCommandWrite(0x06); // entry mode set:
    // increment automatically, no display shift
delay(20);
LcdCommandWrite(0x0E); // display control:
    // turn display on, cursor on, no blinking
delay(20);
LcdCommandWrite(0x01); // clear display, set cursor position to zero
delay(100);
LcdCommandWrite(0x80); // display control:
    // turn display on, cursor on, no blinking
delay(20);
}
void loop (void) {
    LcdCommandWrite(0x02); // set cursor position to zero
    delay(10);
    // Write the welcome message
    LcdDataWrite('H');
    LcdDataWrite('o');
    LcdDataWrite('l');
    LcdDataWrite('a');
    LcdDataWrite(' ');
    LcdDataWrite('C');
    LcdDataWrite('a');
    LcdDataWrite('r');
    LcdDataWrite('a');
    LcdDataWrite('c');
    LcdDataWrite('o');
    LcdDataWrite('l');
    LcdDataWrite('a');
    delay(500);
}
```

Actividades de Autoevaluación de la UNIDAD

Teniendo como precedente los contenidos de las unidades anteriores se proponen las siguientes actividades:

1. Con referencia a los ejemplos planteados en el capítulo 3 de la unidad 2, realice un recorrido por “los pasos en el método de trabajo para el desarrollo de aplicaciones”, identifique cada paso y justifíquelo en su grupo colaborativo.
2. Profundice e investigue respecto a las extensiones de archivos de ensamblador mas usuales, en foro grupal genere su propia explicación para los términos más usuales en microcontroladores.
3. Con respecto a los modos de direccionamiento, seleccione varias instrucciones dentro del set de instrucciones de microchip o Motorola y determine el modo de direccionamiento que utiliza, comparta esta experiencia con el grupo.
4. Tome el microcontrolador PIC16F84 o JK3, buque el datasheet o la documentación de cada dispositivo, determine sus funciones, elementos, arquitectura y puertos que dispone.
5. Descargue el MPLABIDE de la página oficial de Microchip o una versión más liviana y proceda a editar y generar el código fuente, identificando las partes que lo constituyen.
6. Teniendo como precedente y soporte los temas tratados en las lecciones del capítulo 2 de la unidad 3, implemente los programas de manejo de puertos, display 7 segmentos, exploración de teclado y display LCD.
7. Implemente en protoboard los circuitos anteriores y pruebe su funcionamiento.
8. Desarrolle el ejercicio de atención a interrupción y modifíquelo para manejar una secuencia de más de cuatro LEDs.
9. Diseñe e implemente un programa que responda al control de motor paso a paso con variación de giro y secuencia de pasos. Las secuencias de pasos se refieren a las “bobinas” que son activadas, esta se dividen en:
 - a. Pasos simples cuando se energiza una “bobina” a la vez.

- b. Pasos dobles cuando se energiza dos “bobinas” a la vez, con un corrimiento de paso.
- c. Medios pasos cuando se combinan las dos anteriores.

Referencia:

<http://www.todorobot.com.ar/informacion/tutorial%20stepper/stepper-tutorial.htm>

- 10. Diseñe e implemente un proyecto simple que involucre la mayoría de conceptos vistos en el curso.

Fuentes Documentales de la Unidad 3

DOCUMENTOS IMPRESOS

Angulo, Usategui José María. (n. d). Microcontroladores PIC. Diseño práctico de aplicaciones.

Angulo. (n.d). Microcontroladores PIC, la solución en un chip. Sección 5.1

Vesga, Ferreira Juan Carlos. (2007). Microcontroladores Motorola – Freescale: Programación, familias y sus distintas aplicaciones en la industria.

González, Vásquez José Adolfo. (1992). Introducción a los microcontroladores: hardware, software y aplicaciones. Editorial McGraw-Hill.

Dorf, C. Richard. (1997). "Circuitos Eléctricos. Introducción al análisis y diseño". (2^a edición). Editorial AlfaOmega S.A. Santafé de Bogotá.

Savant. J, Roden. S. Martin & Carpenter. L. Gordon. (1992). "Diseño Electrónico. Circuitos y sistemas". (2^a edición). Editorial Addison-Wesley Iberoamericana. E.U.A.

CEKIT. (2002). Curso Práctico de Microcontroladores: Teoría, Programación, Diseño, Prácticas Proyectos completos. Editorial Cekit. Pereira-Colombia.

Stallings, William. "Organización y Arquitectura de Computadores". (5^a edición). Editorial Prentice-Hall. Madrid, 2000.

Téllez, Acuña Freddy Reynaldo. (2007). Módulo de Microprocesadores y Microcontroladores. UNAD.

DIRECCIONES DE SITIOS WEB

Herrera. R. Lucelly. (n.d.). Microcontroladores. Sistemas WinIDE. Extraído el 29 de Julio desde.

http://fisica.udea.edu.co/~lab-gicm/Curso_Microcontroladores/Micos_2012_WIN_IDE.pdf

Aparicio. O. H. (n.d). Todomcu. Extraído el 20 de Junio de 2013 desde.

<http://todomcu.scienceontheweb.net>

Teoría de computadores. Extraído el 10 de Julio de 2009 desde

<http://www.computacion.geozona.net/teoria.html>

Dispositivos lógicos microprogramables, extraído el 10 de Julio de 2009 desde

<http://perso.wanadoo.es/pictob/indicemicroprg.htm>

Curso de Microcontroladores Motorola, extraído el 10 de Julio de 2009 desde

http://www.geocities.com/moto_hc08/index.html

SOFTWARE LIBRE

Software, utilidades y varias recursos adicionales son suministrados dentro del aula de curso en el “Herramientas y Sistemas de desarrollo”, algunas de ellas son:

MASM32 DSK version 10: <http://www.masm32.com/>

SimuProc14, simulador hipotético de un microprocesador x86:

<http://gratis.portalprogramas.com/SimuProc.html>

MPLABIDE: www.microchip.com

PicDevelopment Studio:

<http://sourceforge.net/projects/picdev/files/picdev/PicDevelopmentStudio-1.1.exe/download>

ICPROG: www.ic-prog.com/download.html

PROTEUS: <http://www.ieeproteus.com/descarga.html>

RECURSOS AUDIOVISUALES

Videos, animaciones y Objetos Virtuales de Aprendizaje, localizados en los recurso “Salón Virtual (videos, tutoriales, animaciones)” y “Objetos Virtuales de Aprendizaje - OVAs” localizados en el aula del curso virtual, uno de los vínculos externos es:

Video tutoriales sobre PIC: <http://tutorialesvirtuales.com/>

GLOSARIO DE TÉRMINOS

ACUMULADOR: Registro de la unidad aritmética y lógica del ordenador que almacena los resultados intermedios.

ADC: Acrónimo de *Analog to Digital Converter* (Conversor análogo a digital)

ADMINISTRADOR DE MEMORIA EXTENDIDA: Programa que impide que distintas aplicaciones utilicen la misma área de memoria extendida al mismo tiempo.

ALGORITMO: Procedimiento o conjunto de procedimientos que describen una asociación de datos lógicos destinados a la resolución de un problema. Los algoritmos permiten automatizar tareas.

ANCHO DE BANDA: Es la cantidad de información, normalmente expresada en bits por segundo, que puede transmitirse en una conexión durante la unidad de tiempo elegida. Es también conocido por su denominación inglesa: *bandwidth*.

ARCHIVO DE SISTEMA: Archivo que contiene la información necesaria para ejecutar el sistema operativo.

AREA DE MEMORIA ALTA: (HMA) Los primeros 64 KB de memoria extendida. Esta área es utilizada por algunas aplicaciones, entre ellas Windows.

AREA DE MEMORIA SUPERIOR: Área de 384 KB adyacentes a los 640 KB de memoria convencional. Esta área se reserva usualmente para el funcionamiento

de los componentes de hardware del sistema, tal como el monitor, y no se considera parte de la memoria total, ya que las aplicaciones no pueden almacenar información en ella.

ASCII: Siglas de *American Standard Code for Information Exchange*. Se trata de un código casi universal para caracteres, números y símbolos, asignándole un número entre el 0 y el 255 a cada uno de ellos, como por ejemplo, el 65 para la letra A.

ASYNC: Acrónimo de *Asynchronous* (Asíncrono). Es el tipo de comunicación por el cual los datos se pasan entre dispositivos de forma asíncrona o sea que la transmisión de un carácter es independiente del resto de los demás caracteres.

BANDERA: Indicador interior del programa que da información sobre una condición determinada.

BAUDIO: Unidad que mide la velocidad de transmisión de los datos. Normalmente representa el número de bits por segundo.

BIT DE INICIO: Primer bit en un conjunto de datos que indica que los siguientes son datos.

BIT DE PARADA: Último bit en un conjunto de datos que indica que los anteriores son datos.

BIT DE PARIDAD: Se trata del método más elemental de detección de errores. Consiste en un único bit que indica si el número de bits enviados es par o impar.

BITS POR SEGUNDO: Se abrevia usualmente como 'bps'. Es el número de bits de datos enviados por segundo y es la auténtica velocidad de transmisión.

BLOQUES DE MEMORIA SUPERIOR: (UMOS) Zonas no utilizadas del área de memoria superior. Si se dispone de una computadora con un procesador 80386 o 80486, es posible copiar en los bloques de memoria superior información procedente de otros tipos de memoria, liberando así más memoria convencional (los primeros 640 KB).

BUCLE: Conjunto de instrucciones que se repiten varias veces seguidas.

BUFFER: Área de almacenamiento temporal de información.

BUS: (Línea) Cableado utilizado para transmitir un conjunto de señales de información entre dispositivos de un computador. De su amplitud (expresada en bits simultáneos) depende su velocidad.

CACHÉ DE DISCO: Una porción de la memoria reservada para almacenar temporalmente información leída en un disco.

CAMPO: Colección de caracteres que forman un grupo distinto, como un código de identificación, un nombre o una fecha. Generalmente un campo forma parte de una información.

CHECKSUM: (Suma de comprobación). Es el más elemental de los controles de errores. Consiste normalmente en un único byte obtenido mediante una o varias operaciones matemáticas realizadas sobre todos los bytes de un bloque de datos con el fin de controlar los posibles errores que se produzcan durante una transmisión.

COMPILADOR: Programa que pasa otro programa escrito en un lenguaje de alto nivel (parecido al humano) al lenguaje de la máquina de modo que ésta lo entienda perfectamente.

COMUNICACIÓN SÍNCRONA: Es el tipo de comunicación por el cual los datos se pasan entre dispositivos de forma síncrona o sea que la transmisión depende de la meticulosa sincronización de los datos transmitidos, enviados y de sus propios mecanismos de transmisión. No requiere ni bit de inicio ni bit de parada, a diferencia de la comunicación asíncrona.

CPU: Acrónimo de *Central Processing Unit* (unidad central de procesamiento) Es el procesador central del computador encargado de controlar rutinas, realizar funciones aritméticas, y otras tareas propias. Cada vez se le suele descargar de más tareas, consiguiendo de esta forma un mayor rendimiento.

DAC: Acrónimo de *Digital to Analog Converter* (Conversor de digital a analógico)

DEPURAR: (*debug*). Eliminar los errores de un programa.

DIAGRAMA DE FLUJO (FLOW-CHART): Trazado de la escritura y curso de un programa en el que se utilizan formas diferentes, como un rectángulo o un

cuadrado para indicar una acción del ordenador, y un rombo para las decisiones tomadas por éste. Normalmente, se suele hacer el gráfico o diagrama del programa antes de introducir una sola línea de éste en el computador.

DIRECCIÓN: Número que se refiere a la posición, generalmente en la memoria del computador, en la que se almacena la información.

DIRECCIONES DE E/S: (Entrada /Salida) Posiciones dentro del espacio de dirección de entrada/salida de la computadora que son utilizadas por un dispositivo, como puede ser una impresora o un módem. La dirección es utilizada para la comunicación entre el software y el dispositivo.

DMA: Acrónimo de *Direct Memory Access* (Acceso directo a memoria).

DOBLE PALABRA: Agrupación de 32 bits consecutivos equivalente a cuatro bytes.

E/S: Acrónimo de 'Entrada/Salida'. Suele aplicarse al flujo de datos. También es conocido por su acrónimo inglés 'I/O'.

EMULADOR DE MEMORIA EXPANDIDA: Utilidad que convierte memoria extendida en memoria expandida.

ENSAMBLADOR (ASSEMBLER): Es un programa que convierte otro programa escrito en lenguaje de ensamblaje (*assembly language*) en un código que el microprocesador puede ejecutar directamente.

ENTRADA (INPUT): Toda la información que se introduce en un programa durante su ejecución.

FIRMWARE: Son los programas que funcionan dentro del computador, relacionados íntimamente con el hardware. El firmware puede alterarse, hasta cierto punto, por medio del software.

I/O: Acrónimo de Input/Output (Entrada/Salida). Suele aplicarse al flujo de datos. También es conocido por su acrónimo castellano 'E/S' (Entrada/Salida).

INSTRUCCIÓN: Elemento del código de programación que le dice al computador que lleve a cabo una tarea determinada. Una instrucción en lenguaje de ensamblaje, por ejemplo, podría ser ADD, con lo que le dice al ordenador que realice una suma.

INTEL: Uno de los mayores fabricantes de procesadores, chips y circuitos integrados del mundo, de nacionalidad estadounidense. Sus 'CPUs' más conocidas son: 8086, 8088, 80286, 80386, 80486, Pentium y su co-procesador matemático: 80387.

INTERFAZ: Conexión mecánica o eléctrica que permite el intercambio de información entre dos dispositivos o sistemas. Habitualmente se refiere al 'software' y 'hardware' necesarios para unir dos elementos de proceso en un sistema o bien para describir los estándares recomendados para realizar dichas interconexiones. Es más conocido por su denominación inglesa: 'interface'.

INTERRUPCIÓN: Señal que un dispositivo envía a la computadora cuando dicho dispositivo, por ejemplo, no está listo para aceptar o enviar información.

IRQ: Acrónimo de *Interrupt ReQuest* (Petición de interrupción). Señal que envía un dispositivo para ser atendido por el programa encargado de gestionarlo. En un PC (no inferior a un 80286) existen 15 IRQs de las cuales sólo los números 10, 11, 12 y 15 están realmente libres ya que las otras las ocupa el propio sistema. Conocido simplemente como: interrupción.

K: Abreviatura de kilobyte; 1 K= 1.024 bytes .

KB/S: Acrónimo de Kilobytes per second (Kilo-bytes por segundo)

KBIT/S: Acrónimo de Kilobits per second (Kilobits por segundo)

LENGUAJE DE ALTO NIVEL: Lenguaje de programación que utiliza instrucciones escritas con palabras comunes.

LENGUAJE DE BAJO NIVEL: Lenguaje parecido al utilizado en el computador.

LENGUAJE DE MÁQUINA: Es el escalón inferior al lenguaje de bajo nivel; es el lenguaje que el ordenador entiende directamente .

LÍNEAS DE INTERRUPCIÓN REQUERIDA: (IRO) Líneas del hardware sobre las cuales los dispositivos pueden enviar señales de interrupción, las que indican si el

dispositivo está listo para aceptar o enviar información. Por lo general, cada uno de los dispositivos conectados a la computadora utiliza una línea IRQ independiente.

MAINFRAME: Computador muy potente al que se conectan una o varias estaciones de trabajo.

MB/S: Acrónimo de Megabytes per second (Mega-bytes por segundo)

MBIT/S: Acrónimo de Megabits per second (Megabits por segundo)

MEGABYTE: Equivale a: 1024 Kilo-bytes ó 1.048.576 bytes (2 elevado a la 20).

MEMORIA CONVENCIONAL: Primeros 640 KB de memoria que utiliza el MS-DOS para ejecutar aplicaciones.

MEMORIA EXPANDIDA: Memoria que utilizan algunas aplicaciones No-Windows además de la memoria convencional. La memoria expandida es un estándar antiguo que se está sustituyendo hoy por la memoria extendida. Sólo el software compatible con EMS puede utilizar memoria expandida. Cuando Windows se ejecuta en el modo extendido del 386, simula memoria expandida para aquellas aplicaciones que la necesiten. También denominada Memoria EMS.

MEMORIA EXTENDIDA: Memoria más allá de un megabyte (MB) en las computadoras basadas en los procesadores 80286, 80386 y 80486. Windows utiliza memoria extendida para administrar y ejecutar aplicaciones. Por lo general,

la memoria extendida no podrá ser utilizada por las aplicaciones No-Windows ni por MS-DOS.

MEMORIA VIRTUAL: Sistema de administración de memoria utilizado por Windows en el modo extendido del 386, que permite ejecutar Windows como si hubiese más memoria de la que existe en realidad. La cantidad de memoria virtual disponible será igual a la cantidad de memoria RAM libre sumada al volumen de espacio de disco asignado a un archivo de intercambio que Windows utiliza para simular memoria RAM adicional.

MEMORIA VOLATIL: Un mecanismo de memoria que pierde sus contenidos cuando se le corta la energía eléctrica.

MICRO: Abreviatura de microcomputador.

MICROCHIP: Diminuto circuito de silicio que funciona como memoria, procesador, etc .

MICROPROCESADOR: El chip que hace de corazón del computador o de su parte pensante.

MODO EXTENDIDO DEL 386: Modo en el que se ejecuta Windows para tener acceso a las capacidades de memoria virtual del procesador Intel 80386. En este modo, Windows parece utilizar más memoria de la disponible físicamente y puede trabajar en el modo de multitarea con aplicaciones No-Windows.

MODO PROTEGIDO: Modo de funcionamiento de la computadora que permite tener acceso directamente a la memoria extendida.

MSB: Acrónimo de *Most Significant Bit* (Bit más significativo)

MULTITAREA: Capacidad que tiene una computadora de ejecutar varias aplicaciones al mismo tiempo.

MULTIUSUARIO: Capacidad de permitir el acceso a varios usuarios simultáneos a un determinado programa, aplicación, sistema o servicio telemático en línea sin destruir la integridad de mismo.

NIBBLE: Agrupación de 4 bits consecutivos equivalente a medio byte.

PALABRA: *Word*. Agrupación de 16 bits consecutivos equivalente a dos bytes.

PAQUETE: Conjunto de caracteres enviados conjuntamente durante una comunicación. Los bloques más comunes suelen ser de 64, 128 ó 1024.

PC: Acrónimo de *Personal Computer* (Computadora personal) Computador presentado por 'IBM' el 12 de agosto de 1981 en EE.UU. y comercializado en 1982. Fue el primer ordenador que se vendió masivamente en todo el mundo siendo su denominación original: 'IBM PC'. Con la idea de que el ordenador debía transportarse, surgió su primer competidor: 'Compaq' que lanzó en 1982 su propio portátil. El despegue real del 'PC' llegó en 1983 cuando 'IBM' anunció un auténtico estándar: el 'PC XT' siendo posteriormente culminado por el 'PC AT'.

PERIFÉRICO: Dispositivo externo o interno que se conecta al computador.

POR DEFECTO: Dícese del disco, el programa, etc. que entra en funcionamiento cuando no se da ninguna otra instrucción.

PUERTO: Canal o interfaz que une un periférico a un microprocesador. Puede ser serie (envía los datos bit a bit) o paralelo (envía los datos byte a byte).

REGISTRO: Parte de la memoria del computador que contiene datos de uso frecuente .

SALIDA (OUTPUT): Todos los datos producidos por el computador mientras está procesando, ya aparezcan estos datos en la pantalla, ya los entregue a la impresora o los utilice internamente.

SERVIDOR: Computadora que suministra espacio de disco, impresoras u otros servicios a computadoras conectadas con ella a través de una red.

TERMINAL: Dispositivo de entrada/salida de datos. El terminal "tonto" (a veces denominado 'TTY') maneja simplemente dicha entrada/salida mientras que el "inteligente" utiliza un microprocesador para realizar esa tarea con capacidades adicionales no disponibles en el anterior.

TIEMPO DE ESPERA: Cantidad de tiempo que deberá esperar la computadora cuando un dispositivo no ejecute una determinada tarea, antes de presentar un mensaje de error.

TIEMPO REAL: Modalidad de funcionamiento de un sistema de proceso de datos que controla una actividad en curso, con un tiempo de respuesta prácticamente nulo a la recepción de las señales de entrada.

UNIDAD: Dispositivo de un computador que acepta discos o cintas en los que lee o graba datos.

UNIDAD CENTRAL DE PROCESAMIENTO: Es el corazón del computador, en donde se llevan a cabo las funciones aritméticas, lógicas, de procesamiento y de control.

VLSI: Acrónimo de *Very Large Scale Integration* (Integración a muy gran escala)
Este tipo de integración permite diseñar circuitos cada vez más reducidos en tamaño lo que, en la práctica, hace que nuestros computadores, módems, etc sean cada vez más diminutos, aumentando paradójicamente sus prestaciones.

Características tecnográficas

Ver documento:

Salazar, R. J. (2004). *El material didáctico en el contexto de la formación a distancia y el sistema de créditos académicos*. Bogotá: UNAD.