Informe de Tópicos III

N° 7: "Lenguaje de Bajo Nivel"

Nombre: Juan Pablo Arancibia González

Carrera: Ingeniería en Computación e Informática

Fecha: 22/05/2014

Índice

Introducción…… 3

Objetivos…..…… 4

Lenguaje de Bajo Nivel…… 5

Lenguaje Ensamblador…… 7

Conclusiones…… 9

Bibliografía…… 10

Introducción

Existen dos arquitecturas presentes en el procesamiento del computador, la arquitectura Harvard y la arquitectura de Von Neumann, mientras las computadoras modernas se basan en la arquitectura de Von Neumann, la arquitectura de Harvard es utilizada primordialmente en microcontroladores (como las PIC y Arduino), debido a que tiene que enfocar todas las funciones de computación en un solo dispositivo.

La denominada Máquina Rudimentaria es un simulador el cual simula como funciona paso a paso la interpretación y la lectura de los datos que ingresa al controlador, y su función es servir como herramienta didáctica para la comprensión en la estructura y arquitectura de computadores, por medio de entender el lenguaje de bajo nivel.

Objetivos

Objetivo Principal:

Averiguar sobre lenguaje de Bajo Nivel.

Objetivos Específicos:

Entender que es un lenguaje de bajo nivel y los ensambladores.

Lenguaje de Bajo Nivel

La máquina rudimentaria es un simulador que interpreta al lenguaje ensamblador, el lenguaje ensamblador es uno de los dos tipos de lenguaje de programación de bajo nivel que es muy cercano al lenguaje de la máquina computacional, los cuales sólo contienen algunas pocas funciones comparado a otros lenguajes de programación. Ejemplos de Lenguajes de bajo nivel son los RISC Y CISC los cuales son ensambladores de un lenguaje concreto, definido como kernel, además el lenguaje ensamblador debe ser programado detalladamente ya que es orientado hacia la máquina.

El lenguaje ensamblador generalmente realiza tres tipos de instrucciones, uno es los objetos orientados a byte, los objetos orientados a bit, y los objetos de iteración u operacionales, las instrucciones orientadas a byte permiten modificar todo un byte de registro de datos (ejemplo: MOVWF ADCON 1,0, Mover un dato almacenado en W hacia un registro F que es ADCON1 y el 0 indica que el destino es el propio W.) las instrucciones orientadas a bit indican mover un grupo de 4 bits hacia otra dirección (Ejemplo: BTFSS PORTA, 1, BTFSS indica que pruebe modificar el registro de bit y si está en set (bit=1) puede saltarlo, hacia el bit uno del Puerto A (PORTA)), las instrucciones literales generalmente tienen 8 bits aunque las funciones de llamada (Call) y retorno (Goto) tienen 11 bits (Ejemplo MOVLW 0x05, indica que mueva el literal 5 hexadecimal hacia W).

Otra característica muy importante es saber los ciclos de la frecuencia del cristal del microprocesador, por ejemplo, si el cristal tiene una frecuencia de 4 MHz, indica que el ciclo de oscilación del cristal es de 0,25 microsegundos, así en un periodo de 4 oscilaciones el procesador hace:

1° Oscilación: Decodifica la información.

2° Oscilación: Lee la Información.

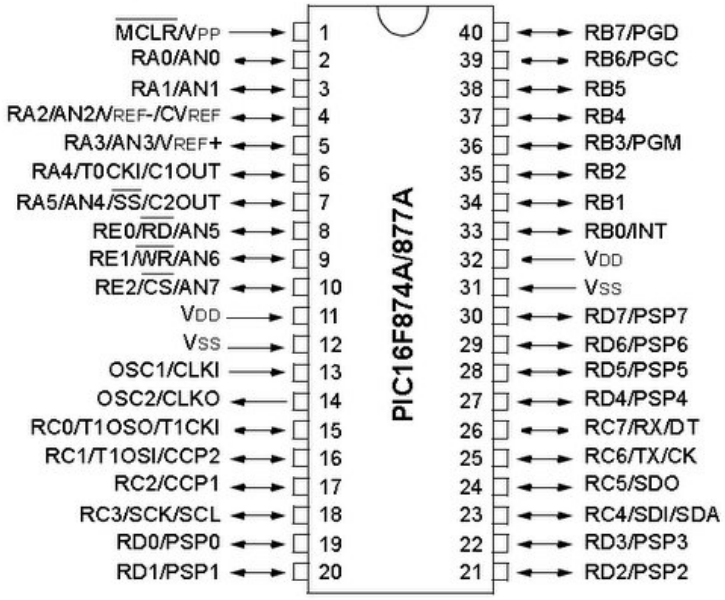
3° Oscilación: Procesa la información.

4° Oscilación: Escribe la información.

Por lo tanto una instrucción es leída y ejecutada por el microprocesador en 4 oscilaciones o sea en 1 microsegundo.

Los microprocesadores vienen con una imagen de referencia que indica en las instrucciones de cada pin, además de un mapa de registros que indica en que bit se ubica una instrucción, por ejemplo estos son las imágenes de referencia del microcontrolador PIC16F877A.

40-Pin PDIP



source : PIC16F877A
data
manual,microchip.inc

FIGURE 2-3: PIC16F876A/877A REGISTER FILE MAP

Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3	File Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD(1)	08h	TRISD(1)	88h		108h		188h
PORTE(1)	09h	TRISE(1)	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved(2)	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved(2)	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h	General Purpose Register 16 Bytes	117h	General Purpose Register 16 Bytes	197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch	CMCON	9Ch		11Ch		19Ch
CCP2CON	1Dh	CVRCON	9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
General Purpose Register 96 Bytes	20h	General Purpose Register 80 Bytes	A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h
	7Fh	accesses 70h-7Fh	EFh F0h FFh	accesses 70h-7Fh	16Fh 170h 17Fh	accesses 70h - 7Fh	1EFh 1F0h 1FFh

Unimplemented data memory locations, read as '0'.
* Not a physical register.

Lenguaje Ensamblador

El Lenguaje ensamblador representa una representación simbólica del código binario y otras constantes para construir una arquitectura de la CPU. Además los lenguajes ensambladores están basadas en los mnemónicos que simbolizan las instrucciones, los registros, las direcciones de memoria y otras características. Un mnemónico es una interpretación de una palabra que es sustituida por un código de operación donde el mnemónico es un código de máquina estructurada que cumple una función específica, donde el resultado es un pseudocódigo que resulta fácil de recordar para el desarrollador.

El lenguaje ensamblador incluye un programa utilitario llamado ensamblador (valga la redundancia), el cual traduce sentencias del lenguaje ensamblador hacia el código máquina del computador a razón de una instrucción a la vez, a diferencia de otros lenguajes donde una instrucción puede tener muchas interpretaciones por la máquina.

El lenguaje ensamblador además refleja las instrucciones del lenguaje de máquina de la CPU y por lo tanto pueden ser diferentes los lenguajes entre las diferentes arquitecturas, como extensión, cada modelo de CPU tiene su propia arquitectura y su propio lenguaje ensamblador, debido a que la arquitectura está estrechamente relacionada con la estructura del hardware de la CPU. Las CPU todas tienen sus grupos en común aunque no todas tienen todas las instrucciones de cada grupo, los cuales hacen una CPU en particular tenga instrucciones que puede que no tenga otra CPU y viceversa.

Los grupos de instrucciones de la CPU pueden ser agrupados así:

- Operaciones de enteros: Realizadas por la unidad aritmética lógica a su vez pueden dividirse en operaciones aritméticas (suma, resta, multiplicación, división, modulo, cambio de signo), operaciones booleanas (AND, OR, XOR, NOT), operaciones de bits (desplazamiento y rotaciones de bits, con un bit de acarreo o sin él), y comparaciones.
- Operaciones de mover datos: aunque se llama mover datos en realidad lo que hace es copiar datos sin que el dato desaparezca del origen, el tipo más común es mover datos entre los registros y la memoria, e incluso entre el registro mismo a otro lugar del registro, o entre un lugar de la memoria hacia otro lugar de la misma memoria, además de mover datos inmediatos hacia el registro o la memoria, además existen las operaciones de pila (stack) que son Push y Pop, que escribe datos hacia el tope de la pila (push), y lee datos desde el tope de la pila (pop), por ultimo esta Input y Output que lee y escribe datos a un puerto de salida.

- Operaciones del control de flujo del programa: Contiene llamadas, retornos y saltos condicionales e incondicionales.
- Operaciones con números reales (IEEE 754): Realiza operaciones con punto flotante mediante el co-procesador matemático (si es que está presente), agrega a las operaciones aritméticas lógicas las operaciones de valor absoluto, parte entera, operaciones trigonométricas, operaciones con logaritmos, potencias y raíces.
- Ademas como los lenguajes ensambladores contienen mnemónicos, algunas CPU pueden incluir mnemónicos personalizados a partir de la creación de macros.

Finalmente queda recalcar que para cada computador, se ha escrito al menos un lenguaje de programación ensamblador, en algunas arquitecturas muy relacionadas, pueden ser parcialmente compatibles en el modo de que puede traducir una instrucción, pero generalmente no hay compatibilidad debido a las características avanzadas que tiene cada modelo de arquitectura del computador.

Otra característica del lenguaje ensamblador es que es difícil su portabilidad hacia otros computadores del mismo tipo, pero como las declaraciones de llamadas tienen poca diferencia entre los diferentes sistemas operativos se puede obtener portabilidad en teoría usando las bibliotecas del lenguaje C y un simulador de control de instrucciones (que es escrito en lenguaje ensamblador), puede en teoría procesar el código objeto binario de cualquier ensamblador.

El Lenguaje de Máquina es un sistema de códigos que es directamente interpretable por un circuito programable, está compuesta por un conjunto de instrucciones que al determinar acciones son tomadas por la máquina, al igual que el lenguaje ensamblador, el lenguaje de máquina es específico de la arquitectura de la máquina.

Los circuitos programables son sistemas digitales que trabajan con 2 niveles de tensión, los cuales se simbolizan con un 0 (baja tensión) y un 1 (alta tensión), de ahí que el lenguaje de máquina solo utiliza estos símbolos, además el estudio de estos circuitos permite la enseñanza de las teorías del sistema binario y de la algebra booleana en el diseño de estos circuitos y su programación.

Conclusiones

El Lenguaje de Bajo Nivel es un lenguaje donde el desarrollador entiende cómo se comporta la máquina computacional, y por lo tanto debe entender las instrucciones y los comandos que componen la programación a bajo nivel.

Si bien el acercamiento del programador a comprender el lenguaje de la máquina se puede considerar una ventaja, programar a bajo nivel tiene sus desventajas, como su poca portabilidad y compatibilidad haciéndose muy rudimentario su entendimiento fácil, además de su costo elevado de tiempo en tener en puesta a punto los programas, hacen a los lenguajes de bajo nivel uno de los más avanzados en cuanto a su aprendizaje y puesta en práctica.

Bibliografía

http://itis.cesfelipesegundo.com/secciones.php?name=Asignatura&cod=604&seccion=Simulador%20de%20la%20MR

http://es.wikipedia.org/wiki/Lenguaje_ensamblador

http://www.sites.upiicsa.ipn.mx/polilibros/portal/Polilibros/P_terminados/PolilibroFC/Unidad_III/Unidad%20III_5.htm

https://www.youtube.com/watch?v=rKz5dtqNg_U (Video sobre lenguaje en ensamblador)