CONTENIDO PARCIAL 1 MICRO

Introducción al lenguaje Ensamblador

Definición: los lenguajes de programación sirven para escribir programas que permitan la comunicación usuario – maquina.

Clasificación:

Según paradigma de programación:

Imperativos: se ordena a la computadora como realizar una tarea siguiendo una serie de pasos o instrucciones. Imperativos: es decir, como una secuencia de operaciones a realizar

Ejemplos de lenguajes imperativos: ASP, BASIC, PASCAL, FORTRAN

Declarativos: se le indica a la computadora que es lo que se desea obtener o que es lo que se está buscando (lógico y funcionales). Declarativos: es decir, se especifica el resultado deseado, no como lograrlo

Ejemplos de lenguajes declarativos: SwiftUI, SQL, HASKELL, MIRANDA

Según su forma de ejecución:

Compilados: aquellos que se ejecutan como un todo pasando por uno o varios procesos de traducción hasta ser comprensibles por la computadora. C, C++, GO son lenguajes de programación compilados los videojuegos

Interpretados: aquellos que se ejecutan instrucción por instrucción haciendo traducción de un determinado lenguaje a un idioma comprensible por la computadora. JAVASCRIPT, PHYTON, RUBY son lenguajes interpretados, Interpretados las páginas web

La principal diferencia entre un lenguaje compilado y uno interpretado es que el lenguaje compilado requiere un paso adicional antes de ser ejecutado, la compilación, que convierte el código que escribes a lenguaje de máquina. Un lenguaje interpretado, por otro lado, es convertido a lenguaje de máquina a medida que es ejecutado.

Según su nivel de abstracción

Alto nivel: lenguajes independientes de la arquitectura del computador por lo que se puede migrar de una maquina a otra tomando en cuenta únicamente los factores de compatibilidad. Ejemplos c#, Python y java.

Mediano Nivel: Tienen características que los acercan a los lenguajes de bajo nivel pero al mismo tiempo ciertas cualidades que lo hacen un lenguaje más cercano al humano y por tanto de alto nivel. Ejemplos lenguaje C

Bajo nivel: son lenguajes dependientes arquitectura del computador no se puede migrar o utilizar en otras máquinas sin recodificación. Ejemplo lenguaje ensamblador

Los lenguajes más cercanos a la arquitectura hardware se denominan lenguajes de bajo nivel.

Y los que se encuentran más cercanos a los programadores y usuarios se denominan lenguajes de alto nivel.

Lenguajes de bajo nivel

Lenguaje maquina: es el que le da órdenes a la máquina que son las operaciones fundamentales para su funcionamiento en código binario o código maquina consiste en ceros y unos.

Lenguaje ensamblador: es un derivado del lenguaje maquina y está formado por abreviaturas de letras y números. Se trabaja a nivel de instrucciones es decir su programación es al más fino detalle.

Lenguaje ensamblador

Lenguaje de maquina: basado en el código binario conjunto de 0 y 1 es el único que puede ser interpretado por las maquinas (computadoras).

Lenguaje ensamblador: basado en “nemónicos” permite la programación por parte del usuario en lenguaje más comprensible para el usuario pero más ligado al microprocesador o microcontrolador.

Lenguaje ensamblador características

Ventajas: mayor velocidad de ejecución, poca demanda de recursos, control preciso de tareas, control preciso de tareas, ejecución directa sobre el hardware, traducción única.

Desventajas: programación compleja (para el usuario), codificación prolongada (tiempo del usuario), programa fuente extenso, portabilidad reducida o nula, errores irreversibles.

Organización de la maquina

Arquitectura del computador definición: es un modelo y una descripción funcional de los requerimientos y las implementaciones de diseño para varias partes de un computadora con especial interés en la forma en que la unidad central de proceso CPU trabaja internamente y accede a las direcciones de memoria.

La arquitectura del computador es el diseño conceptual y la estructura operacional fundamental de un sistema que conforma una computadora

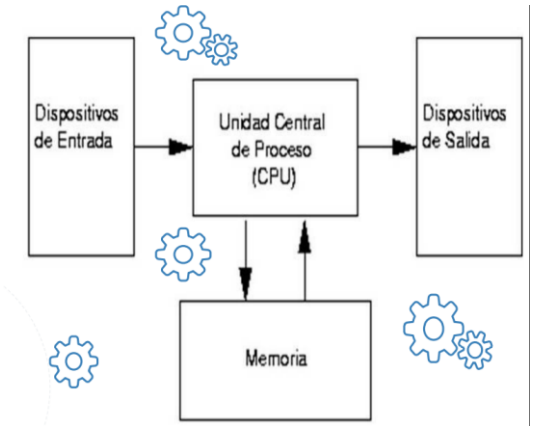
Arquitectura clásica origen:

Arquitectura clásica de computador

Originado del trabajo del matemático John von Neuman.

Divulgado en 1945 en la moore school de la universidad de Pensilvania estados unidos.

Características: procesamiento de datos, almacenamiento de datos, transferencia de datos.



Arquitectura clásica estructura básica:

Arquitectura von Neumann

CPU: la unidad central de proceso dirige todas las funciones de la computadora. Cerebro

Memoria: es la responsable del almacenamiento de datos. Recuerdos.

Entrada/salida: transfiere datos entre el entorno exterior y el computador. Sentidos

Sistema de interconexión (buses): permite el flujo de datos entre la CPU la memoria y los módulos de entrada/salida. Nervios

Periféricos: permiten la entrada de datos y salida de información. Órganos de los sentidos.

CPU (unidad central de proceso)

CU: la unidad de control se encarga de leer de la memoria las instrucciones que debe de ejecutar y de secuenciar el acceso a los datos y operaciones a realizar.

ALU: la unidad aritmética-lógica realiza transformaciones de datos, a través de una serie de módulos que realizan operaciones aritméticas y lógicas.

REGISTROS: almacenan la configuración interna del CPU información de la última operación del ALU y los resultados de la ejecución de instrucciones.

Memoria:

En la memoria se almacena el programa y los datos que va a ejecutar el CPU.

Las instrucciones son códigos binarios interpretados por la CU, los datos de igual manera se almacenan de forma binaria.

Entrada/salida:

Transfiere datos entre el entorno exterior y el computador.

En él se encuentran los controladores de periféricos que forman la interfaz entre los periféricos, la memoria y el procesador.

Sistema de interconexión (buses):

Buses de datos: sirve para transmitir datos entre los diferentes dispositivos del computador. (mueve específicamente los datos)

Bus de control: sirven para seleccionar al emisor y al receptor en una transacción del bus. (va a saber dónde debe ir pero el de direcciones sabe específicamente la posición en donde lo va a guardar)

Bus de alimentación: Sirve para proporcionar a los dispositivos voltajes distintos.

Buses de direcciones: sirve para indicar la posición del dato que se requiere acceder. (si la unidad de control manda a guardar este bus sabe el punto exacto donde se va a guardar)

Periféricos:

Dispositivos necesarios para suministrar datos a la computadora o visualizar los resultados.

Los periféricos se conectan mediante un bus especial a su controlador o al módulo de E/S.

Memoria: a nivel lógico se considera como una matriz de celdas en la que se pueden almacenar datos aleatoriamente organizados en “palabras” (conjunto de celdas, cada celda almacena un bit) y estas definen las instrucciones.

Segmentos definición:

Área especificada en memoria en el límite de un “párrafo” determinada por el modelo del programa usualmente de 64K bytes destinada a un fin determinado.

Segmento de código CS:

Sección de memoria que tiene las instrucciones y procedimientos utilizados por el programa.

El registro del segmento de código (CS) define la dirección inicial en que inicia el segmento de código.

Segmento de código DS:

Sección de memoria que contiene la mayor parte de datos utilizados por el programa.

se tiene acceso al segmento establecido la dirección en la que inicia el mismo en su registro respectivo (DS).

Segmento extra ES:

Sección de memoria utilizada ocasionalmente para algunas instrucciones de cadena o cuando se trabaja con registros extendidos (palabras dobles),

Se ubica en el registro respectivo (ES) al necesitarse algunos microprocesadores también cuentan con otros registros (FS, GS) para segmentos adicionales.

Segmento de pila SP:

Sección de memoria destinada para el arreglo de pila con que trabaja el microprocesador.

la ubicación del punto inicial de entrada a la pila se determina por el registro apuntador de pila (SP).

Registros

Registros definición:

Estructura que tiene ciertas capacidades de almacenamiento temporal, trabaja directamente con el CU y se mueve a la velocidad del ALU.

Registros clasificación:

Según su manipulación:

Por el microprocesador	Por el usuario
MAR	De propósito general
MDR	De apuntadores e índices
IR	De banderas
IP	De segmento

Registros control y estado:

MEMORY ADDRESS REGISTER: registro de alta velocidad y capacidad limitada. Contiene la dirección del dato que se quiere leer o escribir. Se conecta al Bus de direcciones.

MEMORY DATA REGISTER: registro de alta velocidad y mayor capacidad. Contiene lo que el CPU lee o escribe en la memoria o de/a un puerto E/S. Se conecta al Bus de datos.

INSTRUCTION REGISTER: contiene la instrucción que se está ejecutando. Cada instrucción se carga se decodifica prepara y se ejecuta. Registro de la CU.

INSTRUCTION POINTER (INDICE DE PROGRAMA): es visible al usuario pero no modificable por este. Contiene la posición en memoria de la próxima instrucción a ejecutar.

Registros visibles al usuario:

DE SEGMENTOS

CS(SEGMENTO DE CODIGO): contiene la dirección en que inicia el segmento de código.

DS(SEGMENTO DE DATOS): aloja la dirección en que inicia el segmento de datos.

ES(SEGMENTO EXTRA): contiene la dirección en que inicial el segmento extra.

SS(SEGMENTO DE PILA): contiene la dirección en que inicia el segmento de pila.

Registros de propósito general:

AX(ACUMULADOR): usualmente conserva el resultado temporal después de una operación aritmética o lógica. (EAX,AH,AL).

BX(BASE): almacena la dirección base para los accesos a memoria.(EBX,BH,BL)

CX(ACUMULADOR): usualmente contiene el conteo de ciertas instrucciones para corrimiento (CL) y rotaciones del número de bytes (CX) o contador de LOOP (CX y ECX).

DX(DATOS): de uso general contiene la parte más significativa del producto luego de una multiplicación o del dividendo antes de una división. (EDX, DH, DL) es usado para almacenar los datos de las operaciones.

Registros de apuntadores e índices:

SP(APUNTADOR DE PILA): se emplea para direccionar datos en una pila de memoria LIFO.

BP(APUNTADOR BASE) se usa para almacenar desplazamiento en los distintos segmentos. Por defecto es el segmento de la pila.

SI(INDICE DE FUENTE): empleado para direccionar datos fuente en forma indirecta y utilizarlos con las instrucciones de cadena o arreglos. ESI

DI(INDICE DE DESTINO): empleado para direccionar datos destino en forma indirecta y utilizarlos con las instrucciones de cadenas o arreglos. EDI

Registros de banderas:

C(ACARREO): indica un acarreo después de una suma o un “préstamo” en una resta.

P(PARIDAD): es un “ 0 ” para una paridad impar y un “ 1 “ para una paridad par.

Z(CERO): indica que el resultado de una operación aritmética o lógica es cero. Si Z= 1 el resultado es cero; si Z= 0 el resultado no es cero.

S(SIGNO): indica el signo aritmético del resultado después de una suma o resta. Si S= 1 la bandera de signo se activa y el resultado es negativo. Si S = 0 la bandera de signo se desactiva y el resultado es positivo.

I(INTERRUPCION): indica si se procesó o ignoro una entrada externa.

O (DESBORDAMIENTO): indica que el resultado de una operación aritmética ha excedido la capacidad de máquina.

Direcciones composición:

Dirección de memoria: una dirección de segmento y una dirección de desplazamiento.

Direcciones: la dirección ubicada en un registro de segmento define la dirección inicial de cualquier segmento de memoria de 64K bytes.

La direccione de desplazamiento selecciona una localidad dentro del segmento de memoria de 64K bytes.

Programación en lenguaje ensamblador

Ensamblador diferenciación:

Turbo assembler: ensamblador de programas con datos de 8 y 16 bits. Compatibilidad con Windows, compatibilidad con enlazador Turbo Linker, Procesadores Intel 8086 – 80286.

Estructura de un programa en lenguaje ensamblador:

Palabras reservadas

Modelo

Directivas

Identificadores

Instrucciones

Clasificación de las palabras reservadas:

Nemónicos generales: MOV, LEA, POP, PUSH, CALL, JMP, RET

Directivas: .model, .stack, .code

Operadores: ADD, SUB, MUL, DIV

Símbolos predefinidos: @DATA

Estructura modelo:

Modelo: identificador especifico (palabra reservada) que indica al ensamblador el tamaño de los segmentos de código y datos (si se crea) define la estructura del programa en lenguaje ensamblador.

Tipos de modelo:

Tiny: datos y código en el mismo segmento.

Small: datos y código en segmentos independientes 64K bytes c/u.

Compact: datos crean nuevos segmentos al llenar, código en segmento único. 64K bytes c/u.

Médium: datos en segmento único, código crea nuevos segmentos al llenar. 64K bytes c/u.

Large: datos y código crean nuevos segmentos al llenar. 64K bytes c/u.

Estructura directivas:

Indican al ensamblador la estructura del programa, inicializan los segmentos, se constituyen de palabras reservadas. .data, .model , .stack, .80286

Estructura identificadores:

Nombre que se le asignan a elementos específicos de un programa, asignados por el programador. Identificador = num1 , numero = D+ b/d/h

Estructura instrucciones:

Conjunto de nemónicos e identificadores válidos, que en conjunto ejecutan una acción del programa.

El ensamblador no distingue mayúsculas de minúsculas.

PALABRA RESERVADA [OPERANDOS] [“ ; ” COMENTARIOS]

Estructura algoritmo:

Definición del modelo

.model [modelo]

Definición del segmento de datos y/o código

.data [Identificadores][Palabra reservadas]

.code [instrucciones]

Definición del segmento de pila

.stack

Finalización del programa

End.

Instrucciones básicas

Instrucciones de movimientos:

MOV destino fuente: transfiere un byte o una palabra desde el operando fuente al operando destino. Destino: registro elemento de memoria. Fuente: registro, elemento de memoria o valor inmediato.

PUSH fuente: coloca una palabra (una palabra = 16 bits)(2 bytes = 16 bits) en la pila. Transfiere el operando fuente a la cima de la pila.

POP destino: saca una palabra de la pila. Transfiere el elemento que se encuentra en la cima de la pila al operando destino. Incrementa el puntero de pila (SP) en 2.

LEA destino, fuente: transfiere la dirección efectiva (desplazamiento) del operando fuente al operando destino. Fuente: operando de memoria (byte o palabra). Destino: registro de 16 bits no se puede utilizar un registro de segmento.

LDS destino, fuente: transfiere un puntero de 32 bits (dirección compuesta por segmento y desplazamiento) correspondiente al segundo operando. Destino: debe ser un registro pero no de segmento el valor de segmento indicado por el segundo operando de las instrucciones se carga en el registro DS. Fuente: operando de memoria de doble palabra (32 bits).

LES destino, fuente: igual que LDS, pero el segmento se transfiere al registro ES.

Instrucciones aritméticas:

ADD destino, fuente: suma los dos operandos. El resultado se almacena en el destino. Los dos operandos deben ser del mismo tipo (byte o palabra).

ADC destino, fuente: suma los dos operandos incrementando el resultado en 1 si la bandera de acarreo esta activada.

SUB destino, fuente: resta el operando fuente al operando destino. El resultado se almacena en el operando destino. Los dos operandos deben ser del mismo tipo (byte o palabra).

SBB destino, fuente: resta el operando fuente al operando destino. Resta 1 al resultado si la bandera de acarreo esta activada.

MUL fuente: multiplica sin considerar el signo el acumulador (AL o AX) por el operando fuente. Si fuente es byte (8 bits) se multiplica por AL y el resultado se almacena en AX. Si fuente es palabra (16bits) se multiplica por AX y el resultado se almacena en DX:AX (parte alta en DX y parte baja en AX).

IMUL fuente: multiplica considerando el signo el acumulador (AL o AX) por el operando fuente. Si fuente es byte (8 bits) se multiplica por AL y el resultado se almacena en AX. Si fuente es palabra (16 bits) se multiplica por AX y el resultado se almacena en DX:AX (parte alta en DX y parte baja en AX).

DIV fuente: divide sin considerar el signo AX o DX:AX entre fuente. Si fuente es byte divide AX entre fuente el cociente se almacena en AL y el resto en AH. Si fuente es palabra divide DX:AX entre fuente el cociente se almacena en AX y el resto en DX.

IDIV fuente: divide considerando el signo AX o DX:AX entre fuente. Si fuente es byte divide AX entre fuente el cociente se almacena en AL y el resto en AH. Si fuente es palabra divide DX:AX entre fuente el cociente se almacena en AX y el resto en DX.

INC destino: incrementa el operando en 1.

DEC destino: decrementa el operando destino en 1.

NEG destino: niega el operando destino (complemento a 2).

Instrucciones lógicas

AND destino, fuente: operación lógica AND a nivel de bits.

OR destino, fuente: operación lógica OR a nivel de bits.

XOR destino, fuente: operación lógica XOR a nivel de bits.

NOT destino: operación lógica NOT. Complementa los bits del operando.

Instrucciones comparativas:

TEST fuente1, fuente2: realiza la operación AND entre los dos operandos sin almacenar el resultado solo actualiza las banderas del registro de estado.

CMP fuente1, fuente2: lleva a cabo la operación de resta entre los dos operandos, sin almacenar el destino. Actualiza las banderas.

Instrucciones de saltos:

JMP dir: salto incondicional a dir. El operando será normalmente una etiqueta mediante la cual se referencia la instrucción del programa donde se quiere realizar el salto.

JE dir: salto si igual (Z = 1) comúnmente se usa como resultado de CMP.

JZ dir: salto si igual (Z = 1) comúnmente se usa cuando un resultado da 0.

JNE/JNZ dir: salto si no igual (Z = 0).

JS dir: salto si signo negativo (S = 1).

JL dir: salto si el resultado de la última instrucción es menor que 0.

JLE dir: salto si el resultado de la última instrucción es menor o igual que 0.

JNS dir: salto si signo positivo (S = 0)

JG dir: salto si el resultado de la última instrucción es mayor que 0.

JGE dir: salto si el resultado de la última instrucción es mayor o igual que 0.

JP dir: salto si paridad par (P = 1)

JNP dir: salto si paridad impar (P = 0)

JCXZ dir: salto si CX = 0.

LOOP etiqueta: salta a etiqueta si el registro CX es distinto de 0. Además decrementa el valor de CX. Esta instrucción se utiliza para implementar ciclos para ello se inicializa CX con el numero de iteraciones que debe realizar el ciclo y posteriormente se colocan aquellas instrucciones que se desean ejecutar de forma iterativa delimitadas por la etiqueta y la instrucción LOOP.

LOOPE / LOOPZ etiqueta: salta si la bandera Z es 1 y CX es distinto de 0. Decremento CX.

LOOPNE/LOOPNZ etiqueta: salta si la bandera Z es 0 y CX es distinto de 0. Decrementa CX.

INT n: ejecuta el manejador de la interrupción especifica en el operando.