Pontificia Universidad Católica Madre Y Maestra

Campus Santiago

Facultad de Ciencias de la Ingeniería

Departamento de Ingeniería de Sistemas y Computación

Arquitectura Computacional

ST-ISC-359-T-001

Reporte Práctica #0, T0

"Set de Instrucciones y Arquitectura x86"

Presentado por:
Ing. Eric T. Núñez Chaves

Matricula:
2014-1329

A:
Ing. Álvaro A. Reyes P.

En fecha:
9 de septiembre del 2016

Índice

Introducción

X86 es la denominación genérica otorgada a ciertos procesadores de la familia de Intel y a la arquitectura a la que estos pertenecen. Esta arquitectura comienza en el año 1978 con el nacimiento de los procesadores Intel 8086/88, caracterizados por usar bus de datos de 16 bits y con un tamaño de palabra e instrucciones de 32 bits. La creciente demanda de velocidad produjo el nacimiento de la arquitectura 80x64, la cual cuenta con un tamaño de palabra e instrucciones de 64 bits. Desde su nacimiento hasta la actualidad, Intel ha mantenido la compatibilidad binaria con los procesadores anteriores.

Por otro lado, el lenguaje utilizado para comunicarnos con el procesador es el lenguaje ensamblador, aquél lenguaje simbólico que trabaja más cerca del procesador y es utilizado para programar a bajo nivel un computador, permitiendo ejecutar una serie de instrucciones básicas del procesador. A continuación, se realizará una breve descripción de las instrucciones para los modelos de la familia x86.

Conceptos Teóricos

1. Características fundamentales de la arquitectura x86

La familia x86 sigue el modelo de arquitectura de Von Newman, caracterizada por utilizar el mismo dispositivo de almacenamiento, tanto para las instrucciones como para los datos. Ésta consta de una unidad de CPU (Unidad de Procesamiento Central), memorias, buses y dispositivos de entrada y salida:

- **Unidad de proceso central (CPU):** Unidad encargada de controlar y dirigir el sistema que comprende una computadora, además de realizar cálculos, comparaciones y transferencia de datos como respuesta a las peticiones de programas almacenados en memoria. Dispone de dos unidades en su interior:

    1. **Unidad de control:** Se encarga de leer las instrucciones enviadas por programas almacenados en memoria.

    2. **Unidad aritmético-lógica(ALU):** Es la encargada de realizar las operaciones aritméticas y lógicas y envía las ordenes correspondientes a los componentes del procesador para ejecutarlas. Puede realizar:
        - Operaciones aritméticas
        - Operaciones con lógica binaria
        - Operaciones de rotación y desplazamiento
        - Operaciones de transformación de operando

    3. **Registros de la CPU:** Estos se encargan almacenar posiciones de memoria, pues el acceso a estos registros es mucho más rápido que el acceso a los registros de memoria. Según su funcionalidad se dividen en:
        - Registros de propósito general o de almacenamiento temporal
        - Registros de Segmento

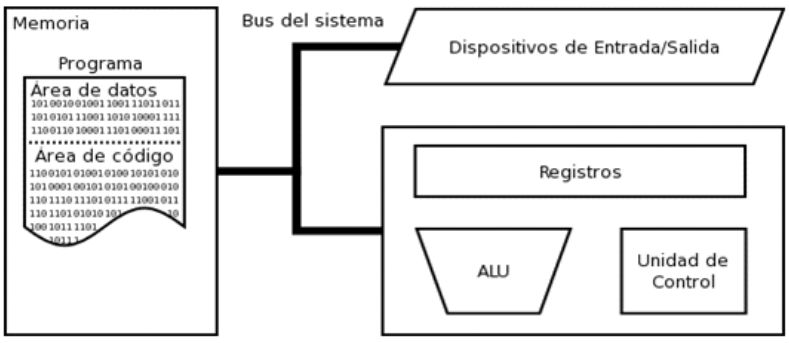
- Registros Apuntadores o Registros de Pila
- Registros Índice
- Registro de Instrucción
- Registro Bandera

- **Memoria:** La memoria inicialmente era simplemente la RAM. Sin embargo, con el paso del tiempo se ha añadido la memoria caché y algoritmos inteligentes. La memoria RAM es aquella que contiene la dirección donde se encuentran los datos y el contenido del propio dato. Se compone de un registro de direcciones (RDM), en el que se almacena la dirección en la será almacenado o leído el dato y un registro de intercambio de memoria (RIM), que almacena el dato leído o que será dispuesto a ser almacenado.

- **Buses:** es el medio por el que se comunican todos estos elementos y son utilizados para la transferencia de datos entre las distintas partes. Hay tres tipos:
  - **Bus de datos:** permite el intercambio de datos (instrucciones o datos) con los demás elementos que conforman la arquitectura.
  - **Bus de instrucciones:** envía las direcciones de memoria que serán usadas desde la CPU, para poder seleccionar los datos que serán usados.
  - **Bus de control:** transporta las instrucciones de la CPU para controlar los distintos procesos de la máquina.
- **Dispositivos de E/S:**
  - **Entrada:** Son aquellos que sirven para introducir datos a la computadora para su posterior procesamiento.
  - **Salida:** Son aquellos que permiten representar los resultados del procesamiento interno de los datos.

Por último, las características básicas de este tipo de arquitectura son:

- Longitud de instrucción variable
- Diseño CISC (Complex Instruction Set Computer)
- Registros: de estado EFLAG, apunta a la siguiente instrucción a ejecutar del segmento de código; puntero de instrucción EIP, indica el estado del procesador en un momento dado y 8 de propósito general, operandos en las diversas instrucciones.
- 8 registros MMX y XMM
- Recursos para el manejo de la pila y la invocación de subrutinas.
- Modos de Funcionamiento:
    - Modo protegido: Modo en el que están disponibles todas sus características del procesador.
    - Modo direccionamiento-real: Modo de compatibilidad con el modelo 8086.En este modo, el procesador inicia siempre su ejecución.
    - Modo de mantenimiento: Modo útil para labores de mantenimiento del sistema operativo, como en aspectos de control y seguridad.



2. Set de instrucciones y su clasificación

- **Aritméticas**

Forman parte del repertorio de instrucciones de cualquier lenguaje.x86 permite ejecutar operaciones aritméticas básicas, pudiendo trabajar con operandos de 8-16 bits, con o sin signo. Éstas pueden ser:

- **Suma:** Se pueden agrupar en instrucciones que funcionan con dos operandos y con solo un operando:

  - **Con dos operandos:**

    - **ADD o suma (Destino, Fuente):** Se suma el contenido de los dos registros, y su resultado es almacenado en el registro de destino.
    - **ADC o suma con acarreo (Destino, Fuente):** Se realiza el mismo proceso que en la instrucción ADD, pero tomando en cuenta el "acarreo".

  - **Con un operador:**

    - **INC o incremento (Op):** Operación que incrementa en uno el registro.

- **Resta:** Se pueden agrupar en instrucciones que funcionan con dos operandos y con solo un operando:
  - **Con dos operandos:**
    - **SUB o resta (Destino, Fuente):** Se resta el contenido de los dos registros y su resultado es almacenado en el registro de destino.
      Ej.

      ; Ax=10
      ; Bx=5
      SUB Ax, Bx
      Luego de la ejecución:
      Ax=5
      Bx=5

- **SBB o resta con acarreo (Destino, Fuente):** Se realiza el mismo proceso que en la instrucción SUB, pero tomando en cuenta el "acarreo".

  - **Con un operador:**
    - **DEC o decremento (Op):** Operación que decrementa en uno el registro.

- **Multiplicación:** Entendidas como un proceso de sumas repetitivas. Sus instrucciones son:

  - **MUL o multiplicación con signo (Op):** Multiplica valores sin signo.

  - **IMUL o multiplicación sin signo (Op):** Multiplica valores con signo.

- **División:** Presenta dos instrucciones:
  - **DIV o división con signo (Op):** Divide valores sin signo.
  - **IMUL o división sin signo (Op):** Divide valores con signo.

Tanto la multiplicación como la división tienen la restricción de deberse encontrar en el registro AX y no superar el tamaño del multiplicando.

- **Comparación:**
  - **CMP o comparación (Op1, Op2):** Comparación de dos argumentos, cuyo resultado no afecta al operando fuerte, pero si al registro de banderas.
  - **TEST:** Comparación de bits para tomar decisiones.

- **Desplazamiento de bits**
  - **SAL (Op, Cant):** Desplazamiento aritmético a la izquierda. Se desplazan los bits del operando

destino tantas veces como indique el operando fuente(Cant).

- **SAR (Op, Cant):** Desplazamiento aritmético a la derecha. Se desplazan los bits del operando destino tantas veces como indique el operando fuente(Cant).

- **Rotación de bits**
  - **RCL (Op, Cant):** Rotación con acarreo a la izquierda.
  - **RCR (Op, Cant):** Rotación con acarreo a la derecha.
  - **ROL (Op, Cont):** Rotación a la izquierda**.** Rota el destino a la izquierda tantas veces como indique el contador.
  - **ROR (Op, Cont):** Rotación a la derecha. Rota el destino a la derecha tantas veces como indique el contador.

- **Lógicas**

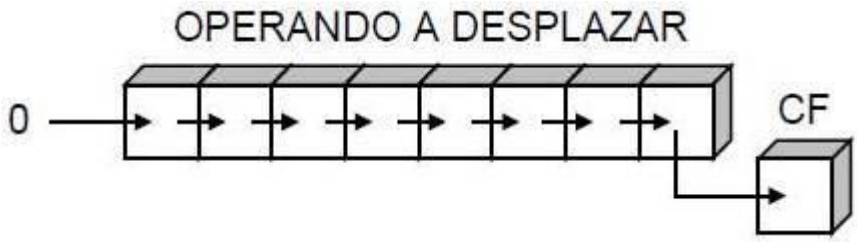
Existen cuatro tipos de operaciones lógicas:

- **NEG o Negación:** variación de la operación resta. Se encarga de restar el operando de 0. Op: =0-Op.
- **NOT o Inversor:** invierte los bits, cambios los unos por ceros y los ceros por unos.
  Ej.
  ; Ax=10
  NOT Ax
  Luego de la ejecución:
  Ax=01

- **AND o "Y":** Realiza un AND lógico bit a bit entre los operandos.
- **OR o "O":** Realiza un OR lógico bit a bit entre los operandos.
- **XOR o "O Exclusivo":** Realiza un OR exclusivo lógico bit a bit entre los operandos.

- **SHL (Op, Cant):** Desplazamiento lógico a la izquierda. Se desplazan los bits del operando destino tantas veces como indique el operando fuente(Cant). Este desplazamiento ocurre de la siguiente forma:

- **SHR (Op, Cant):** Desplazamiento lógico a la derecha. Se desplazan los bits del operando destino tantas veces como indique el operando fuente(Cant). Se produce de esta forma:



- **Transferencia de datos**
  Son aquellas instrucciones que se encargan de mover aquellos datos relacionados con la memoria. Éstas pueden ser:

  - **MOV (Destino, Fuente):** Se encarga de la transferencia de datos entre registros del procesador o entre un registro y memoria. El contenido del registro fuente pasa al registro destino.
    Ej.
    ; ebx=3
    ; eax=2
    MOV eax, ebx
    Luego de la ejecución:
    eax=3

ebx=3

- **XCHG (Op1, Op2):** Intercambia el contenido del registro fuente con el de destino.
- **STC:** Activa la bandera de dirección. Pone la bandera CF en 1.
- **CLC:** Limpia la bandera de dirección. Pone el bit en 0 de la bandera CF.
- **CMC:** Invierte el valor del bit de la bandera. Si este está en 0 lo pone en 1; si este está en 1 lo pone en 0.
- **STD:** Activa la bandera de interrupción. Pone la bandera DF en 1.
- **CLD:** Limpia la bandera de interrupción. Pone el bit en 0 de la bandera DF.
- **STI:** Activa la bandera IF. Habilita las interrupciones externas enmascarables.
- **CLI:** Limpia la bandera de interrupciones, poniendo el bit en 0 y deshabilitando así las interrupciones enmascarables (Aquellas que se desactivan cuando IF=0).
- **PUSH(Fuente):** Almacenar en la pila.
- **PUSHF:** Se decrementa en 2 el valor del registro SP, para luego transferir el contenido del registro de flags a la pila, siguiendo la dirección indicada por SP.
- **PUSHA:** Apila los registros generales dentro de la pila.
- **POP:** Transfiere el último valor almacenado en la pila en el operando destino, después de haber incrementado en dos el registro SP.
- **POPF:** Transfiere los bits de la palabra almacenada en la parte superior de la pila, y lo envía hacia el registro de banderas.
- **POPA:** Extrae de la pila los registros generales, retirándose el contenido de SP.
- **CBW:** Convierte el Byte a Word.
- **CWD:** Convierte Word a doble.
- **CWDE:** Convierte Word a doble extenso.

● **Control de flujo:** Se encargan de generar saltos, pudiendo ser estos absolutos y condicionales y llevan rutinas y subrutinas a diversas partes de un programa. Existen diferentes tipos de salto:

- **JMP(Destino):** Salto condicional. Desvía el flujo de un programa sin tomar en cuenta las condiciones en las que se encuentren las banderas o los datos.
  Ej.
  A90=etiqueta de una instrucción.
  JMP A90
  Salto a la etiqueta A90
- **JE(Destino):** Salto de igualdad. Salta si es igual.
- **JZ(Destino):** Salta si es cero.
- **JCXZ(Destino):** Realiza un salto si CX = 0.
- **JP(Destino):** Salta si hay paridad.
- **JNE(Destino):** Salta si no se cumple la igualdad.
- **JNZ(Destino):** Salta si es distinto a cero.
- **JECXZ(Destino):** Realiza un salto si ECX = 0.
- **JNP(Destino):** Salta si no hay paridad.
- **JPO(Destino):** Salta únicamente si la paridad es impar.
- **JPE(Destino):** Salta si la paridad es par.

A continuación, se van a mostrar los saltos sin signo:

- **JA(Destino):** Salto condicional. Salta si es mayor.
- **JAE(Destino):** Salta si es mayor o igual.
- **JB(Destino):** Salta si es menor.
- **JBE(Destino):** Salta si es menor o igual.
- **JNA(Destino):** Salta si no es mayor.
- **JNAE(Destino):** Salta si no es mayor o igual.
- **JNB(Destino):** Salta si no es menor.
- **JNBE(Destino):** Salta si no es menor o igual.
- **JC(Destino):** Salta si hay acarreo.
- **JNC(Destino):** Salta si no hay acarreo.
- **JPE(Destino):** Salta si la paridad es par.

A continuación, se van a mostrar los saltos con signo:

- **JG(Destino):** Salto condicional. Salta si es mayor.
- **JGE(Destino):** Salto condicional. Salta si es mayor o igual.
- **JL(Destino):** Salta si es menor.
- **JLE(Destino):** Salta si es menor o igual.
- **JNG(Destino):** Salta si no es mayor.
- **JNGE(Destino):** Salta si no es mayor o igual.

- **JNL(Destino):** Salta si no es menor.
- **JNLE(Destino):** Salta si es menor o igual.
- **JO(Destino):** Salta si se produce desbordamiento u overflow.
- **JNO(Destino):** Salta si no hay desbordamiento u overflow.
- **JS(Destino):** Salta si hay signo, es decir, si es negativo.
- **JNS(Destino):** Salta si no hay signo, es decir, si es positivo.

3. Modos de direccionamiento.

Son las maneras en las que los microprocesadores accesan a los distintos recursos con los que cuenta.

Para los tipos de direccionamiento se utiliza la instrucción **MOV** que transfiere la información o los datos de un recurso a otro(registro-memoria/memoria-registro). Estos son:

- **Direccionamiento inmediato:** transfiere un byte o dato al operando de destino. Es utilizado para inicializar registros
- **Direccionamiento por registro:** transporta un byte o datos desde un registro fuente hacia un operando de destino.
- **Direccionamiento directo:** transporta un byte o datos desde el segmento DS a un registro de entre 8-16 bits.
- **Direccionamiento indirecto:** permite direccionar los datos desde cualquier localidad de memoria, mediante el uso de los registros índice y base
- **Direccionamiento base más índice:** Transferencia de un byte o dato entre un registro y una localidad de memoria, direccionada por un registro base más un registro índice.
- **Direccionamiento relativo por registro:** Transferencia de un byte o dato entre un registro y una localidad de memoria, direccionada por un registro base o un registro índice más un desplazamiento.
- **Direccionamiento base más índice:** Transferencia de un byte o dato entre un registro y una localidad de memoria, direccionada por un registro base más un registro índice, pero también más un desplazamiento.

Conclusión

Se puede concluir que se ha realizado un reporte cumpliendo con los objetivos especificados en esta práctica, consistentes en conocer en profundidad las características de la arquitectura x86 cuyo rendimiento ha crecido sustancialmente con la salida de nuevas generaciones.

Por otro lado, se ha realizado un análisis exhaustivo de las instrucciones compatibles con todos los procesadores de la familia x86,asi como la clasificación de estas instrucciones según su uso y aplicación, acompañando cada categoría de instrucciones con un ejemplo propio que nos permita una mejor comprensión del funcionamiento del lenguaje ensamblador. Éste es de vital importancia pues comprender este lenguaje de bajo nivel nos permite comprender la interacción del usuario con los procesadores y una comprensión más profunda del funcionamiento de los lenguajes de alto nivel.