Arquitectura de Ordenadores

Juego de Instrucciones del Procesador Intel Pentium

Abelardo Pardo
abel@it.uc3m.es

Universidad Carlos III de Madrid
Departamento de Ingeniería Telemática

El Diseño de un Lenguaje Máquina

- Las instrucciones y sus operandos se codifican mediante **un conjunto de bits y un formato**.
- El número de instrucciones, y la forma de obtener los operandos influyen en el **tamaño de las instrucciones**.
- Los lenguajes máquina se dividen en **dos categorías**:

1. **Instrucciones con formato fijo:** el tamaño de la instrucción y su formato es idéntico para todas las instrucciones.
   - Decodificación **simple**.
   - Código **ineficiente en tamaño**. Hay casos en los que se utilizan más bits de los necesarios.
2. **Instrucciones con fomato variable:** el tamaño, la posición o presencia de campos varía dependiendo del tipo de instrucción.
   - Decodificación **compleja**.
   - Código **compacto**.

- El Pentium tiene instrucciones de **longitud variable**.

Formato Binario de Instrucciones del Pentium

- Todas las instrucciones del Pentium responden al siguiente **formato variable**:

- El procesador procesa el primer byte, y obtiene **bytes adicionales** conforme se precisen campos adicionales al decodificar la instrucción.
- A la definición **detallada e inequívoca** de cómo se codifican todas las instrucciones de un procesador se le denomina su **arquitectura de juego de instrucciones**.
- La arquitectura de juego de instrucciones **es específica para cada procesador**.
- La programación en ensamblador utiliza **únicamente** las instrucciones definidas por el procesador.

El Programa Ensamblador

- La manipulación de **datos en binario o su equivalente en hexadecimal** es muy tediosa.
- Para programar en lenguaje máquina se utiliza un **lenguaje con símbolos más intuitivos** llamado **lenguaje ensamblador**.
- Este lenguaje se traduce a **código binario** o lenguaje máquina mediante el programa denominado genéricamente **ensamblador** (en unix se llama **as**).
- Además de instrucciones, el lenguaje ensamblador permite la **definición de datos** y **etiquetas identificativas**.

Formato de Instrucción en Lenguaje Ensamblador

- El formato de toda instrucción en lenguaje ensamblador es:

Operación [Operando 1,] [Operando 2]

Ejemplos:

- ADD $0x10, %eax
- INC %ebx
- CMP longitud, %ecx
- ENTER $0, $0

- ¿Cómo se describe **la funcionalidad** de una instrucción?
  - **Abreviatura, número y tipo** de los operandos.
  - Descripción **de las posibles variantes**.
  - Descripción del **propósito** de la instrucción.
  - Descripción **algorítmica de los efectos de la instrucción sobre el procesador.**

Descripción de Instrucción

SAHF–Store AH into flags

Opcode	Instruction	Clocks	Description
9E	SAHF	2	Loads SF, ZF, AF, PF, and CF from AH into EFLAGS register

Description

Loads the SF, ZF, AF, PF, and CF flags of the EFLAGS register with values from the corresponding bits in the AH register (bits 7, 6, 4, 2, and 0, respectively). Bits 1, 3, and 5 of register AH are ignored; the corresponding reserved bits (1, 3, and 5) in the EFLAGS register remain as shown in the “Operation” section below.

Operation

EFLAGS(SF:ZF:0:AF:0:PF:1:CF) ← AH;

Flags Affected

The SF, ZF, AF, PF, and CF flags are loaded with values from the AH register. Bits 1, 3, and 5 of the EFLAGS register are unaffected, with the values remaining 1, 0, and 0, respectively.

Exceptions (All Operating Modes)

None.

Tipo de Operandos

- Los posibles **tipos de operandos** presentes en una instrucción son:

  1. **Constante numérica**: Se precede el número por el símbolo '$'. Se puede escribir en formato binario (precedido por '0b'), decimal, hexadecimal (precedido por '0x') u octal (precedido por '0').
  2. **Registro de Propósito General:** Comienzan por el símbolo '%'. El operando es el contenido del registro.
  3. **Dirección de Memoria:** El operando es **el contenido de dicha dirección**. El procesador permite **múltiples formas** de referirse a una posición de memoria (modos de direccionamiento).
  4. **Operando Implícito:** No consta ninguna referencia en la instrucción, pero el operando se utiliza en su ejecución (por ejemplo `push %eax`).

- De los dos operandos, **como máximo** uno de ellos puede estar almacenado en memoria.
- Cuando una instrucción requiere **dos operandos y un lugar para almacenar el resultado**, el segundo operando es **fuente y destino a la vez**.
- **Ejemplo:** ADD 0x10, %eax: Suma 0x10 al contenido del registro %eax y deposita el valor **en el propio %eax**.

$$\text{ADD op1, op2} \quad \equiv \quad op2 \leftarrow op1 + op2$$

Instrucciones de Movimiento de Datos

- Son instrucciones que no **operan con los datos**, únicamente los mueven.
- La abreviatura utilizada es **MOV**.
- Se especifican **dos operandos**.
- El procesador **almacena** el valor especificado como primer operando en el lugar especificado como **segundo operando**.
- **Ejemplo:** MOV $-1, %ecx
- ¿Se puede especificar el segundo operando de tipo **constante numérica**?

Instrucciones Aritméticas y Lógicas

- Operaciones con **uno o dos operandos.**
- Se subdividen en dos clases: **Aritméticas y lógicas**.
  - **Aritméticas**: suma (**ADD**), resta (**SUB**), multiplicación (**MUL, IMUL**), división (**DIV, IDIV**), incremento (**INC**), decremento (**DEC**), etc.
  - **Lógicas**: conjunción (**AND**), disjunción (**OR**), negación (**NOT**), desplazamientos con/sin signo (**SAL, SAR, SHL, SHR**), rotaciones (**RCL, RCR, ROL, ROR**), etc.
- Las operaciones **MUL**, **IMUL**, **DIV** y **IDIV** tienen **operandos implícitos**.
- **Ejemplo:** sal $2, %eax

Instrucciones de Control

- Otra parte importante de un programa es el **control de la secuencia de instrucciones**.
- El procesador debe permitir la **ejecución no secuencial de instrucciones**.
- Este mecanismo se basa en dos tipos de instrucciones: **comparación** y **salto**.
- La **secuencia habitual** para alterar el flujo de ejecución es **opcionalmente comprobar una condición** y **realizar un salto a un lugar diferente del código**.
- Este tipo de funcionalidad **está presente en los lenguajes de alto nivel como JAVA** (por ejemplo la construcción `if-else`.

Instrucciones de Comparación

- Los códigos de condición se almacenan el el registro de **estado y control** del procesador y se actualizan con la ejecución de cada instrucción.
- **Ejemplo:** Se ejecuta la instrucción equivalente a $t = a + b$. Los códigos de condición se modifican de la siguiente forma.
  - **CF:** Se ha producido acarreo en la suma considerando los operandos como números naturales.
  - **ZF**: $t = 0$
  - **SF**: $t < 0$
  - **OF:** Se ha producido overflow en la suma.
- El procesador posee **instrucciones específicas que sólo** modifican los códigos de condición pero no se guarda el resultado.
  1. **Comparación aritmética CMP**: Se realiza **la resta** del segundo operando menos el primero. Se modifican los códigos de condición. Un inmediato sólo puede ir en el primer operando.
  2. **Comparación lógica TEST**: Se realiza **la conjunción** de los dos operandos. Se modifican los códigos de condición. Un inmediato sólo puede ir en el primer operando.
- **Ejemplos:**
  - CMP $10, %eax /* se resta %eax - 10 si almacenar resultado */
  - TEST $0x00008000, %eax /* ¿Está el bit 15 de %eax a 1? */

Instrucciones de Salto

- Las instrucciones de salto **hacen que el programa pase a ejecutar una instrucción que no es la siguiente en memoria**.
- Estas instrucciones **modifican el registro contador de programa**.
- La instrucción **JMP** requiere especificar un único parámetro que es la **la dirección de memoria** de la instrucción a ejecutar a continuación.
- Las **instrucciones de salto condicional** reciben un único parámetro idéntico a la instrucción JMP, pero realizan el salto **únicamente si la condición a la que hacen referencia se cumple**.
- La condición se extrae de los **bits de condición** almacenados en el registro de estado y control.
- El Pentium cuenta con **63 instrucciones de salto condicional**. Las más comunes son:

Instr.	Condición	Instr.	Condición	Instr.	Condición
je	si igual (ZF=1)	jg	si mayor (SF=0 y ZF=0)	jge	si mayor o igual (SF=0)
jl	si menor (SF=1)	jle	si menor o igual (SF= o ZF=1)	jc	si acarreo (CF=1)
jne	si diferente (ZF=0)	jnz	si no cero (ZF=0)	jz	si cero (ZF=1)

- **Regla mnemotécnica:** Si tras una instrucción del tipo `cmp A, B` se ejecuta un salto condicional con condición `cond`, el salto se ejecuta si se satisface `B cond A`.

```
cmp %eax, %ebx
jge destino    # Salta si %ebx >= %eax
```


Ejemplo de Instrucciones de Control

- La combinación de las instrucciones de comparación y salto permiten modificar la **secuencia de ejecución**.

Ejemplo 1

```
        mov $10, %eax
inicio: dec %eax
        jz final
        ...
        jmp inicio
final:  mov $100, %ecx
        ...
```

Ejemplo 2

```
inicio: inc %eax
        cmp $128, %eax
        je final
        ...
        jmp inicio
final:  mov $'A', %cl
        ...
```

Ejemplo 3

```
        cmp $12, %eax
        jle menor
        mov $10, %eax
        ....
        jmp final
menor:  mov $100, %eax
        ...
final:  inc %ebx
```

Ejemplo 4

```
        test $0x0000FFFF, %ecx
        jne final
        ...
final:  mov $100, %ecx
        ...
```