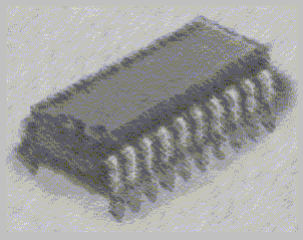
Tema 2. Diseño del repertorio de instrucciones



Arquitectura de Computadores

I. T. Informática de Gestión

Curso 2009-2010

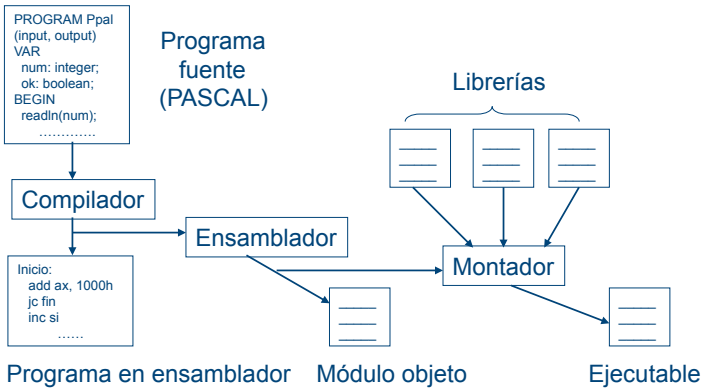
Índice

Conceptos básicos (I). Definiciones

- **Repertorio de instrucciones o juego de instrucciones**: es el conjunto de todas las órdenes que puede ejecutar un computador
- **Instrucción**: es una operación expresada mediante la codificación binaria de cadenas de unos y ceros y que es interpretada y ejecutada por el computador
- **Instrucción** máquina: es la cadena de ceros y unos que se corresponde con una instrucción
- **Código máquina**: es la representación como instrucciones máquina de todo el repertorio de instrucciones
- **Lenguaje ensamblador**: es el juego de instrucciones expresado como un conjunto de mnemónicos en el que cada instrucción en ensamblador se corresponde con una única instrucción de lenguaje máquina

Conceptos básicos (II). Programación

- Proceso de creación de un programa en un lenguaje de alto nivel



Lenguajes de alto nivel y repertorio (I). Almacenamiento de los operandos

- Estudio del impacto de los lenguajes de alto nivel y de los servicios del sistema operativo en el diseño del repertorio de instrucciones

```
TYPE vector = ARRAY [1..n] OF tipoelemento;
PROCEDURE IntercambioDirecto (VAR v: vector);
VAR i, j: integer;
      elemento: tipoelemento;
BEGIN
  FOR i:= 2 TO n DO
    FOR j:= n DOWNTO i DO
      IF v[j-i] > v[j] THEN
      BEGIN
        elemento := v[j-i];
        v[j-i] := v [j];
        v[j] := elemento
      END
END;
```

- Almacenar en registros variables muy usadas
- Operandos por instrucción modificados
- Datos y modos de direccionamiento empleados
- Comparaciones y saltos condicionales e incondicionales

Lenguajes de alto nivel y repertorio (y II). Almacenamiento de los operandos

- Estudio del impacto de los lenguajes de alto nivel y de los servicios del sistema operativo en el diseño del repertorio de instrucciones

```
PROGRAM Principal (input, output);
VAR   v: vector;

BEGIN
   LeerVector ( v );
   IntercambioDirecto ( v );
   MostrarVector ( V );
   writeln ('Progama finalizado')
END.
```

- Llamadas a subrutinas y funciones
- Invocación de servicios del sistema operativo

Parámetros del repertorio (I)

- Elementos a tener en cuenta al diseñar el repertorio de instrucciones

Dimensión de estudio	Responde a
Almacenamiento de operandos en la CPU	¿Dónde se encuentran los operandos además de en memoria?
Número de operandos explícitos por instrucción	¿Cuántos operandos son designados explícitamente en una instrucción típica?
Posición del operando	¿Puede la ALU operar con operandos situados en memoria? ¿Cómo se especifica la posición de memoria?
Operaciones	¿Qué operaciones se proporcionan en el repertorio de instrucciones?
Tipo y tamaño de operandos	¿Cuál es el tamaño y el tipo de cada operando? ¿Cómo se especifcia cada operando

Parámetros del repertorio (II). Alternativas a almacenar en CPU

Almacenamiento temporal	Operandos explícitos por instrucción ALU	Destino para resultados	Procedimiento para acceder a los operandos explícitos
Pila	0	Pila	Introducir en / sacar de la pila
Acumulador	1	Acumulador	Cargar / almacenar el acumulador
Conjunto de registros	2 ó 3	Registros o memoria	Cargar / almacenar registros o posiciones de memoria

Tipo de máquina	Ventajas	Desventajas
Pila	Modelo sencillo para evaluar expresiones Buena densidad de código al ser cortas las instrucciones	Imposibildad de acceso aleatorio a la pila Dificultad de generación de código eficiente Dificultad de implementación eficiente
Acumulador	Minimiza los estados internos de la máquina Instrucciones cortas	La mayor cantidad de tráfico entre acumulador-memoria
Conjunto de registros	Modelo más general para generación de código	Instrucciones largas al tener que nombrarse los registros

Parámetros del repertorio (III). Almacenamiento en memoria

- **Ventajas de las máquinas de registros de propósito general (GPR)**
  - Los registros permiten una ordenación más flexible que las pilas o acumuladores para evaluar las expresiones
  - Los registros pueden emplearse para que contengan variables lo que reduce el tráfico de memoria, acelera el programa y aumenta la densidad de código
- **Los diseñadores de compiladores prefieren registros no dedicados**
- **El número de registros dependerá de cómo los emplee el compilador:**
  - Los que requiera el compilador para evaluar expresiones
  - Los que requiera el compilador para pasar parámetros
  - Los que requiera el compilador para ubicar variables

Parámetros del repertorio (IV). Almacenamiento en memoria

- **Clasificación de la máquinas GPR. Se basa en las características de los operandos para operaciones de la ALU**
  - Número de operandos de la ALU, 2 ó 3
  - Número de operandos direccionados en memoria en la ALU de 0 a 3

Tipo	Ventajas	Desventajas
Registro-registro	▪ Codificación simple de instrucciones de longitud fija ▪ Modelo simple de generación de código ▪ Las instrucciones emplean números similares de ciclos para ejecutarse	▪ Recuento más alto de instrucciones que las arquitecturas con referencias a memoria en las instrucciones ▪ Algunas instrucciones son cortas y la codificación de bits puede ser excesiva
Registro-memoria	▪ Los datos pueden ser accedidos sin cargarlos primero ▪ El formato de instrucción tiende a ser fácil para codificar y obtener buena densidad de código	▪ Los operandos no son equivalentes ya que en una operación binaria se destruye un operando fuente ▪ Codificar un número de registro y una dirección de memoria en cada instrucción puede restringir el número de registros ▪ Los ciclos por instrucción varían por la posición del operando
Memoria -memoria	▪ Más compacta ▪ No emplean registros para temporales	▪ Gran variación en el tamaño de las instrucciones ▪ Gran variación en el trabajo por instrucción ▪ Los accesos a memoria crean cuellos de botella en memoria

Parámetros del repertorio (V). Direccionamiento de memoria

- ¿Cómo se interpreta una dirección de memoria? Byte, media palabra, palabra, doble palabra
- Existen dos convenios para clasificar los bytes de una palabra. Se refieren al lugar que ocupa el byte cuya dirección es xx…xx00

Para almacenar la palabra de 32 bits: 12345678h a partir de la dirección 100h

Direcciones		100h	1001h	102h	103h
Contenido	**Little endian**	78	56	34	12
	Big endian	12	34	56	78

Parámetros del repertorio (VI). Direccionamiento de memoria

- En algunas máquinas los accesos a objetos mayores de un byte deben estar alineados
- Un objeto de tamaño s bytes estará alienado en el byte de dirección A si: A mod s = 0

Almacenamiento de la palabra de 32 bits a partir de la dirección 102h

Palabra 100h	*Byte 100h*	*Byte 101h*	***Byte 102h***	***Byte 103h***
Palabra 104h	***Byte 104h***	***Byte 105h***	*Byte 106h*	*Byte 107h*

102 mod 4 = 2 → no está alineada → dos accesos para leerla de memoria

Almacenamiento de la palabra de 32 bits a partir de la dirección 104h

Palabra 100h	*Byte 100h*	*Byte 101h*	*Byte 102h*	*Byte 103h*
Palabra 104h	***Byte 104h***	***Byte 105h***	***Byte 106h***	***Byte 107h***

104 mod 4 = 0 → si está alineada → necesita un único acceso para ser leída

Parámetros del repertorio (VII). Direccionamiento de memoria

- La alineación es importante ya que un acceso no alineado tendrá múltiples referencias a una memoria alineada
- Incluso en máquinas que no requieren alineación los accesos alineados se ejecutan más rápidamente
- Ejemplos de direcciones alineadas y no alineadas para diferentes tamaños de los objetos direccionados:

Objeto direccionado	Alienado en desplazamiento del byte	Mal alienado en desplazamiento del byte
Byte (8 bits)	0, 1, 2, 3, 4, 5, 6, 7	(nunca)
Media palabra (16 bits)	0, 2, 4, 6	1, 3, 5, 7
Palabra (32 bits)	0, 4	1, 2, 3, 5, 6, 7
Doble palabra (64 bits)	0	1, 2, 3, 4, 5, 6, 7

Parámetros del repertorio (VIII). Modos de direccionamiento

- **Indican la manera de acceder a un dato por parte de una instrucción**
  - **Inmediato**: se encuentra en el código máquina de la propia instrucción.
  - **Directo a registro**: el dato se encuentra en un registro que se especifica en el código máquina de la instrucción
  - **Directo a memoria**: la dirección del dato se encuentra codificada en él código máquina de la instrucción
  - **Relativo**: se especifica un desplazamiento con respecto a un registro. Por tanto la dirección final será la suma de ambos valores
  - **Indirecto**: la dirección especificada por el código máquina de la instrucción no es el dato sino la dirección en la que se debe buscar
  - **Implícito**: no se da ningún tipo de indicación en el código máquina de la instrucción con respecto a la ubicación del dato ya que se trabaja con uno fijo
- **El nombre del modo de direccionamiento puede cambiar según el fabricante**

Parámetros del repertorio (IX). Modos de direccionamiento

- Ejemplo de los modos de direccionamiento y el código máquina asociado en el i80x86

El almacenamiento en el i80x86 es little endian

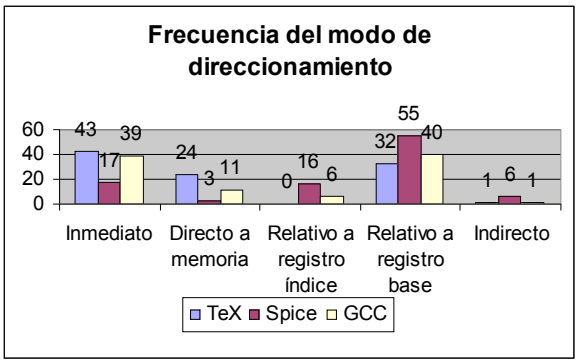
Modo de direccionamiento	Instrucción en ensamblador	Código máquina
Inmediato	MOV ax, ***1234h***	B8 ***34 12***
Directo a registro	MOV ax, bx	89 ***D8***
Directo a memoria	MOV al, ***operando1***	A0 ***25 00***
Relativo	MOV cl, ***Numero[si]***	8A 8C ***26 00***
Indirecto	*No exite en el i80x86*	
Implícito	DIV BX	F7 F3

Parámetros del repertorio (X). Modos de direccionamiento

- **Influencia de los modos de direccionamiento:**
  - El modo de direccionamiento de una instrucción influye en el número de ciclos que tardará en ejecutarse
  - Los modos de direccionamiento permitidos influyen en el ciclo de reloj de la máquina
  - El conjunto de los modos de direccionamiento que se permiten en una determinada máquina influirá en la complejidad del hardware
- **Dependencia de los modos de direccionamiento:**
  - La codificación de los modos de direccionamiento dependerá de la correlación existente entre éstos y los códigos de operación
  - La codificación de los modos de direccionamiento dependerá del rango permitido para los diferentes modos de direccionamiento

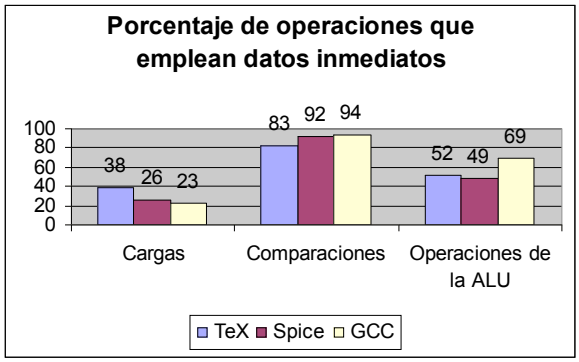
Parámetros del repertorio (XI). Frecuencia de los modos de direcc.

- Las gráficas sobre la frecuencia de uso del modo de direccionamiento están sacadas de un VAX



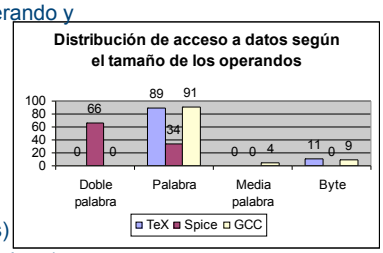
Parámetros del repertorio (XII). Frecuencia de los modos de direcc.

- Las gráficas sobre la frecuencia de uso de operaciones que emplean datos inmediatos están sacadas de un VAX



Parámetros del repertorio (XIII). Tamaño de los operandos

- **Métodos para designar a un operando:**
  - Codificar el operando en el código de la operación
  - Anotar el operando con identificadores que serán interpretados por el hardware (tipo de operando y operación)
- **El tipo de operando indica su tamaño**
- **Tipos de operando más frecuentes:**
  - Byte
  - Media palabra (16 bits)
  - Palabra (32 bits)
  - Coma flotante en simple precisión (32 bits)
  - Coma flotante en doble precisión (dos palabras)
  - Carácter (byte) en ASCII o EBCDIC
  - BCD empaquetado y desempaquetado



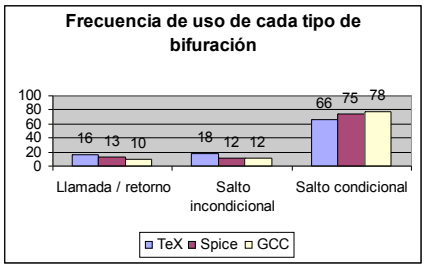
Parámetros del repertorio (XIV). Juego de instrucciones

Tipo de operador	Ejemplos
Aritmético y lógico	Operaciones lógicas y aritméticas enteras
Transferencia de datos	Cargas / almacenamientos
Control	Salto, bifurcación, llamada y retorno de procedimientos
Sistema	Llamada al sistema operativo, instrucciones de memoria virtual
Coma flotante	Operaciones de coma flotante
Decimal	Suma y multiplicación decimal. Conversiones de decimal a caracteres
Cadenas	Tratamiento de cadenas

- Todas las máquinas generalmente proporcionan un repertorio completo para las instrucciones aritmético-lógicas, de transferencia de datos y de control
- El soporte que proporciona el repertorio de instrucciones para las funciones del sistema varia grandemente de unas arquitecturas a otras
- El soporte a operaciones de coma flotante, decimales y de cadenas puede variar de no existir a tener un amplio conjunto de operaciones especiales

Parámetros del repertorio (XV). Juego de instrucciones

- **Instrucciones de control de flujo del programa**
  - Modifican las secuencia de ejecución de las instrucciones
  - Existen básicamente cuatro tipos de control
    - Saltos condicionales
    - Saltos incondicionales
    - Llamadas a procedimientos
    - Retorno de procedimientos
  - La dirección de un salto debe especificarse siempre
  - La forma de especificar el salto es mediante direccionamiento relativo al contador de programa

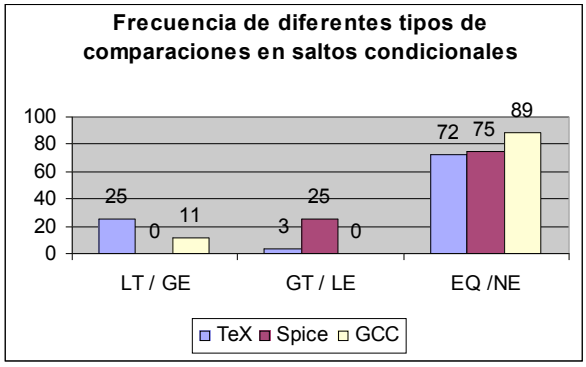


Parámetros del repertorio (XVI). Juego de instrucciones

- **Principales métodos para evaluar las condiciones de salto:**
  - **En un registro de indicadores**: el estado de la condición se almacena en un bit del registro de indicadores por la instrucción que lo genera.
    - Ventaja: Solamente se emplea un bit
    - Inconveniente: problemas de dependencia en superescalares
  - **En un registro de propósito general**: la instrucción que genera la condición pone un uno o un cero en un registro de propósito general
    - Ventaja: Simplifica la dependencia
    - Inconveniente: Se necesita un registro completo
  - **Utilizando una instrucción de comparación y salto**: la instrucción que genera el estado y la que salta es la misma
    - Ventaja: todo se resuelve en una misma instrucción
    - Inconveniente: Puede aumentar el CPI de la instrucción

Parámetros del repertorio (XVII). Juego de instrucciones

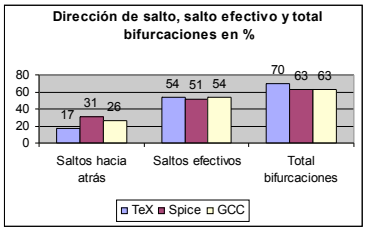
- Comparaciones más empleadas en los saltos condicionales:



Parámetros del repertorio (XVIII). Juego de instrucciones

- Frecuencia de los diferentes tipos de salto:

Programa	Porcentaje de saltos hacia atrás	Porcentaje de saltos efectivos	Porcentaje de todas las instrucciones de control que realmente saltan
TeX	17%	54%	70%
Spice	31%	51%	63%
GCC	26%	54%	63%



Parámetros del repertorio (y XIX). A tener en cuenta en el diseño

- **Resumen de los parámetros a tener en cuenta para el diseño del repertorio de instrucciones de un computador**
  - **Parámetro 1**: almacenamiento de operandos en la CPU
  - **Parámetro 2**: número de operandos explícitos en una instrucción
  - **Parámetro 3**: Modos de direccionamiento
  - **Parámetro 4**: Tipo y tamaño de los operandos
  - **Parámetro 5**. Conjunto de instrucciones que se tendrán en cuenta en el repertorio

Formato de instrucciones (I). Alternativas de diseño

- **Se debe decidir si se desea crear un juego de instrucciones CISC o uno RISC**

- **CISC**: Complex Instruction Set Computer
  - Muchos modos de direccionamiento
  - Muchos tipos de instrucciones
  - Instrucciones que realizan operaciones muy complejas aunque se empleen poco
  - Ej. I80x86, M680x0

- **RISC**: Reduced Instruction Set Computer
  - Pocos modos de direccionamiento
  - Pocas instrucciones y muy sencillas
  - Pocos formatos de instrucción y muy regulares
  - Ej. SPARC

Formato de instrucciones (II). Codificación de las instrucciones

- **Se debe tener en cuenta:**
  - El tamaño del código generado tiene una relación directa con el número de accesos a memoria y por lo tanto con el tiempo de ejecución
  - El número de instrucciones y el tipo influye en la decodificación y por lo tanto, nuevamente en el tiempo de ejecución
  - La facilidad de programación de la máquina
- **Equilibrar en el diseño**:
  - El número de registros y el de los modos de direccionamiento permitidos
  - El tamaño de las instrucciones. Si son múltiplos de la palabra del computador mucho mejor
  - Que los accesos a memoria sean por palabras o líneas de cache

Formato de instrucciones (III). Codificación de las instrucciones

- **Una instrucción debe contener:**
  - **El código de la instrucción**: indica qué operación va a realizar la instrucción
  - **Operandos fuente**: sobre qué valores va a operar la instrucción
  - **Operando destino**: dónde dejará la instrucción el resultado de la operación
  - **Dirección de la instrucción siguiente**: dónde se encuentra la siguiente instrucción del programa. Normalmente el direccionamiento de las instrucciones es implícito
- **Se debe tener en cuenta lo siguiente:**
  - Los repertorios CISC como el del i80x86 suelen hacer que uno de los operandos sea fuente y destino de la operación
  - Los repertorios RISC suelen emplear tres operandos en la ALU dos fuente y uno destino

Formato de instrucciones (IV). Codificación de las instrucciones

- **Características generales:**
  - **Los formatos son sistemáticos**: los campos suelen ir en posiciones fijas
  - **El código de operación**: o su extensión es el primero de los campos
  - **Son múltiplos de la palabra del computador**: para optimizar los accesos a memoria
- **Alternativas en cuanto a formato de instrucciones:**
  - **Formato fijo**: emplea el mismo formato para todos los tipos de instrucción. Es muy difícil de implementar
  - **Formato variable**: presenta el código de operación, extensiones de código, número variable de operandos y de modos de direccionamiento
  - **Formato mixto**: emplea dos o tres formatos fijos para encajar todas las instrucciones
  - **Formato ortogonal**: cualquier instrucción admite todos los tipos de operandos y todos los modos de direccionamiento

Formato de instrucciones (y V). Codificación de las instrucciones

- **Consejos a tener en cuenta en el diseño:**
  - **Equilibrio**. Intentar que estén razonablemente equilibrados los modos de direccionamiento y el número de registros
  - **Instrucciones múltiplos de la palabra del computador**. Lo ideal que una instrucción ocupe una palabra, pero es muy difícil
  - **Tamaño del código**. Dependiendo del formato de las instrucciones la codificación de un programa será más o menos extensa con lo que tiene una influencia directa en el tiempo de ejecución
  - **Decodificación**. Cuanto más complejo sea el formato de las instrucciones más se tardará en decodificar la instrucción por lo que también influye en el tiempo de ejecución
  - **Programación**. Dependiendo de los modos de direccionamiento permitidos y de los operandos será más o menos sencilla de programar
  - **Compilador**. La generación del código la lleva a cabo el compilador por lo que el formato de las instrucciones debe facilitar el proceso

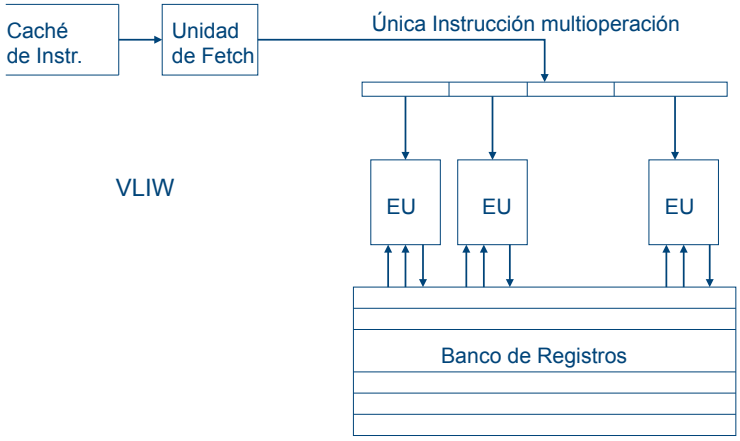
Tecnología VLIW (I)

- **La tecnología RISC ha favorecido la segmentación en los procesadores, permitiendo que varias instrucciones se ejecuten simultáneamente**
- **La ejecución simultánea trae consigo el problema de las dependencias entre las instrucciones. Dos alternativas:**
  - Detectar las dependencias y reordenar el código en tiempo de ejecución mediante hardware específico en el procesador
  - Detectar las dependencias y reordenar el código en tiempo de compilación
- **La tecnología VLIW se basa en la segunda alternativa**

Tecnología VLIW (II)

- Las máquinas VLIW cuenta con un compilador que agrupa diversas operaciones sencillas e independientes entre sí en la misma palabra de instrucción. Cuando la unidad de control las decodifica, las divide en sus operaciones componentes y las despacha en unidades funcionales independientes
- El compilador es quién empaqueta el código en operaciones independientes
- Dado que no hay hardware dedicado a la detección de las dependencias se pueden tener mejores unidades funcionales y se mejora la velocidad de reloj
- Inconveniente: un compilador trabaja sólo con una arquitectura. (CISC o RISC)

Tecnología VLIW (y III)



Compatibilidad binaria (I)

- La traducción binaria es la técnica empleada para cambiar el código ejecutable de un programa asociado a una arquitectura y a un sistema operativo en otro código ejecutable diferente para otra arquitectura y sistema operativo
- La mejor forma de aprovechar al máximo las nuevas capacidades de una arquitectura es portar y recompilar el programa con compiladores nativos. Con frecuencia esto no es posible

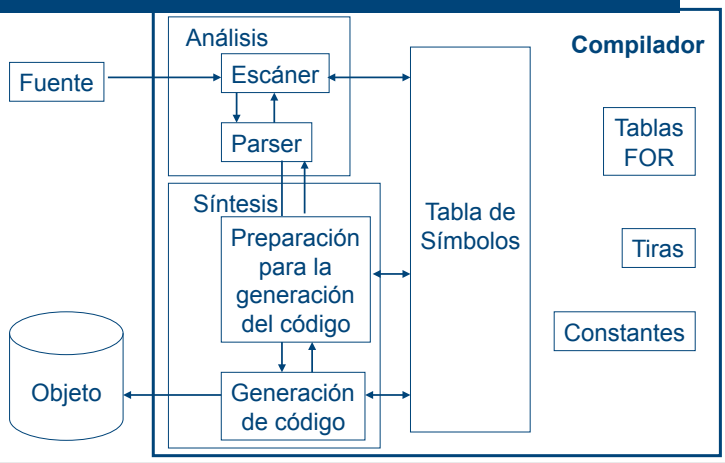
Compatibilidad binaria (y II)

- **Existen diferentes técnicas para realizar la traducción binaria:**
  - **Intérprete software.** Es un programa que lee instrucción a instrucción el programa de la vieja arquitectura y lo traduce a la nueva. No son muy rápidos
  - **Emulador de microcódigo**. Similar al intérprete software pero lleva hardware añadido para ayudar a decodificar más rápidamente las instrucciones antiguas. Solamente funciona en máquinas que emplee microcódigo
  - **Traductores binarios**. Es una secuencia de instrucciones de la nueva arquitectura que reproduce el comportamiento de las de un programa antiguo. Parte de la información de estado de la vieja arquitectura se almacena en registros de la nueva
  - **Compiladores nativos.** La más rápida, consiste en recompilar el programa antiguo

Compiladores (I)

- **Compilador**: es una aplicación que realiza el proceso de traducir un programa fuente, escrito en un lenguaje de alto nivel, a un programa objeto en código máquina y listo para ejecutarse en el ordenador, con poca o ninguna preparación adicional (PSP)
- **Tipos de compiladores**:
  - **Ensamblador**. Traduce de un lenguaje sencillo, el ensamblador, a lenguaje máquina
  - **Compilador cruzado**. Traduce de un lenguaje máquina a otro objeto para una máquina diferente a la que se está compilando
  - **Compilador de una o varias pasadas**. Necesita uno o varios pasos para poder general el código objeto
  - **Compilador incremental**. Solamente se compilan los errores corregidos tras descubrilos
  - **Decompilador**. Realiza el proceso inverso del compilador

Compiladores (II). Estructura de un compilador



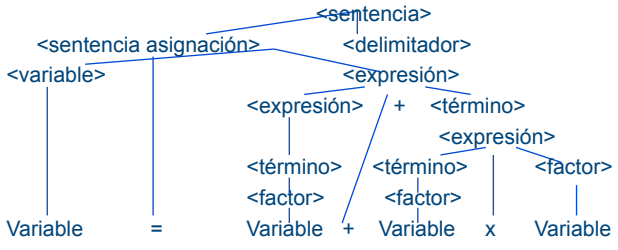
Compiladores (III). Ejemplo de proceso de compilación

- **Sentencia fuente a compilar:**
  - Velocidad := VelocidadInicial + Aceleracion x Tiempo;

El escáner o explorador dará la siguiente salida:

  - Variable = Variable + Variable x Variable Delimitador

- **El parser o reconocedor dará la siguiente salida**



Análisis sintáctico: la sentencia es correcta en pascal

Compiladores (IV). Ejemplo de proceso de compilación

- **La preparación para la generación de código generada por el parser es:**

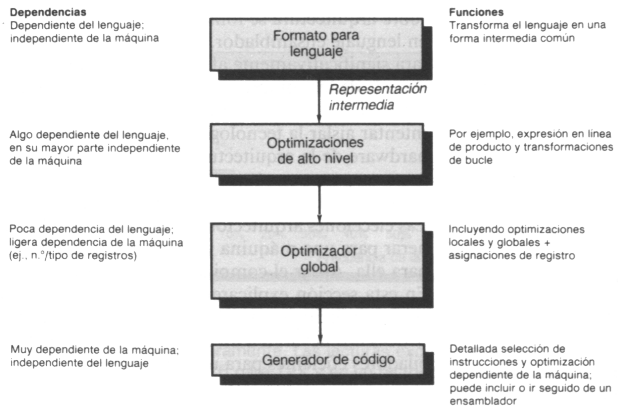
  – Variable Variable Variable Variable x + =

  Se supone que se ha empleado la notación polaca inversa

- **Finalmente el generador de código daría las siguientes instrucciones en lenguaje máquina (expresadas en lenguaje ensamblador del i80x86:**

  MOV AL, Aceleracion
  MUL Tiempo
  ADD AL, VelocidadInicial
  MOV Velocidad, AL

Compiladores (V). Dependencia de cada parte del compilador



Compiladores (VI). Optimizaciones de código

Nombre de la optimización	Explicación
Integración de procedimientos	Decide si se expanden o no los procedimientos
Eliminación global de subespresiones comunes	Sustituye dos instancias del mismo cálculo por simple copia
Reducción de la altura de la pila	Reorganiza las expresiones matemáticas para minimizar los recursos necesarios para evaluarla
Movimiento de código	Elimina código de un bucle que calcula el mismo valor en cada iteración del bucle
Reducción de potencia	Sustituir la multiplicación por sumas y desplazamientos, la división por restas y desplazamientos
Planificación de la segmentación	Reordenar las instrucciones para mejorar el rendimiento de la segmentación

- El empleo de las optimizaciones en el código puede redundar en una mejora del rendimiento entre el 60% y el 80%

Compiladores (VII). Optimizaciones de código

- **Ejemplo de integración de procedimientos:**

```
PROCEDURE TablaVacia (VAR T: Tabla)
BEGIN
    T.NumeroElementos := 0
END;
```

- **El procedimiento es muy sencillo para expandirlo**

Con integración:

```
MOV T.NumeroElementos, 0
```

Sin integración:

```
TablaVacia PROC
    MOV T.NumeroElementos, 0
    RET
TablaVacia ENDP
                .
                .
                .
CALL TablaVacia
```

Compiladores (VIII). Optimizaciones de código

- **Eliminación global de subexpresiones comunes:**

```
.
.
Precio := Cantidad x PUnidad;
Descuento :=
  Cantidad x PUnidad x Tanto;
.
..
Precio := Cantidad x PUnidad;
.
.
```

```
.
.
Temp0 := Cantidad x PUnidad;
Descuento := Temp0 x Tanto;
.
.
.
Temp0 := Cantidad x PUnidad;
.
.
```


Compiladores (IX). Optimizaciones de código

- **Reducción de la altura de la pila**:

Energia := EnergiaInicial + Masa x Exp(Aceleracion, 2);

```
Energia EnergiaInicial Masa Aceleracion Aceleracion x x + =
```

Compiladores (X). Optimizaciones de código

- **Movimiento de código**:

```
MOV CX, 4
MOV SI, 0
Bucle:
    MOV AH, 2
    MOV DL, Cadena[SI]
    INT 21h
    INC SI
    LOOP Bucle
```

```
MOV CX, 4
MOV SI, 0
MOV AH, 2
Bucle:
    MOV DL, Cadena[SI]
    INT 21h
    INC SI
    LOOP Bucle
```


Compiladores (XI). Optimizaciones de código

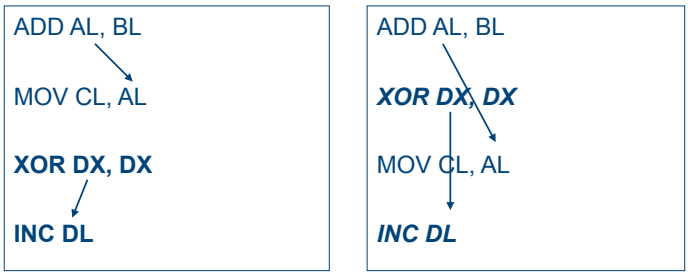
- **Reducción de potencia**:

```
MOV AL, Pi
MUL Radio
MUL 2
MOV Perimetro, AL
```

```
MOV AL, Pi
MUL Radio
SHL AL, 1
MOV Perimetro, AL
```

Compiladores (y XII). Optimizaciones de código

- **Planificación de la segmentación**:



Ejemplos de repertorio de instrucciones (I). Alpha vs. i80x86

- **Características generales de la arquitectura Alpha**
  - Arquitectura de 64 bits
  - Ejecución paralela de instrucciones
  - Configuración multiprocesador
  - Alta velocidad de reloj
  - No orientado a un sistema operativo concreto
  - No orientado a un lenguaje de alto nivel en concreto
- **Características generales de la arquitectura i80x86**
  - Arquitectura de 16 bits que ha evolucionado a 32
  - Compatibilidad binaria
  - Repertorio de instrucciones tipo CISC
  - Dificultad para adaptarse a la ejecución segmentada
  - Dificultad para adaptarse a la ejecución superescalar

Ejemplos de repertorio de instrucciones (I). Alpha

- **Parámetro 1: almacenamiento de operandos en la CPU**
  - 32 registros de propósito general para enteros
  - 32 registros de propósito general para coma flotante
- **Parámetro 2: número explícito de operandos por instrucción**
  - 3 operandos
- **Parámetro 3: modos de direccionamiento**
  - Modelo de ejecución registro-registro
  - Se exige alineamiento
  - Ordenamiento datos configurable. Por omisión es little-endian
  - Único modo de direccionamiento a memoria el relativo a registro
  - Permite que uno de los tres operandos sea un dato inmediato

Ejemplos de repertorio de instrucciones (II). Alpha

- **Parámetro 4: tipo y tamaño de los operandos**
  - Instrucciones de proceso emplean datos de 64 bits
  - Instrucciones de acceso a memoria permiten datos más pequeños: byte, word (16), longword (32)
  - Emplea la extensión de signo para operar con datos menores de 64 bits
  - Los tipos permitidos son entero con y sin signo y los formatos para coma flotante del estándar IEEE y del VAX
- **Parámetro 5: conjunto de instrucciones**
  - Instrucciones de acceso a memoria
  - Instrucciones de control
  - Instrucciones de proceso
  - Instrucciones de coma flotante
  - Instrucciones misceláneas: llamadas al sistema, gestión de memoria, …
  - Instrucciones multimedia

Ejemplos de repertorio de instrucciones (III). i80x86

- **Parámetro 1: almacenamiento de operandos en la CPU**
  - 8 registros de propósito “casi” general para enteros
  - 8 registros de propósito general para coma flotante en forma de pila
- **Parámetro 2: número explícito de operandos por instrucción**
  - 2 operandos. Uno hace las veces de fuente y destino
- **Parámetro 3: modos de direccionamiento**
  - Modelo de ejecución registro-memoria
  - Se recomienda el alineamiento aunque no se exige
  - Ordenamiento es little-endian
  - Modos de direccionamiento: inmediato, relativo a registro, directo a memoria, indirecto a registro, implícito
  - Los operandos de coma flotante siempre son la pila

Ejemplos de repertorio de instrucciones (y IV). i80x86

- **Parámetro 4: tipo y tamaño de los operandos**
  - Trabaja con tamaños de byte, palabra (16) y doble palabra (32)
  - El coprocesador matemático emplea además enteros de 64 y 80 bits y coma flotante de 32, 64 y 80 bits
  - Los tipos permitidos son entero con y sin signo y los formatos para coma flotante del estándar IEEE
- **Parámetro 5: conjunto de instrucciones**
  - Instrucciones de transferencia de datos
  - Instrucciones de control
  - Instrucciones de proceso
  - Instrucciones de coma flotante
  - Instrucciones de manejo de cadenas
  - Instrucciones misceláneas: llamadas al sistema, gestión de memoria, …
  - Instrucciones multimedia: MMX, MMX2, SSE, 3DNow, …

Bibliografía

- Estructura y diseño de computadores
  David A. Patterson y John L. Hennessy. Reverté, 2000
  Capítulo 3

- Arquitectura de computadores. Un enfoque cuantitativo
  John L. Hennessy y David A. Patterson. Mc Graw Hill, 3ª ed, 2002
  Capítulos 3 y 4 y apéndices B, C, D y E

- Estructura de computadores.
  José Mª. Angulo. Paraninfo, 1996
  Capítulo 2 y 9