7. – Lenguaje Ensamblador

Introducción a los microprocesadores
2015

Lenguaje Ensamblador

- Para cada Instrucción del µP hay un Mnemónico

Ej: 01 000 001 ➔ LD B, C

- Existen casi tantos lenguajes ensamblador como µPs.
- Ejemplo:

```
LD A, (OPER1)
LD B, A
LD A, (OPER2)
ADD A, B
LD (SUMA), A
```

¿Qué son OPER1, OPER2 y SUMA ?

¿En que dirección de memoria comienza?

Lenguaje Ensamblador

- Veremos:
  - Reglas de sintaxis y directivas
  - Ensamblado de lenguaje ensamblador a código máquina

Reglas de sintaxis y directivas

- Constantes numéricas
  - Decimales: 239 (en general opción por defecto)
  - Binario: 1110 1111B o 1110 1111b o $11101111 o 0b1110 1111
  - Hexadecimal: 0xEF o 0EFh

  - Ejemplo de conversión por el ensamblador:

    LD A, 5 5 corresponde a 0000 0101 (8 bits)

    LD A, (5) 5 corresponde a 0000 0000 0000 0101 (16 bits)

- Constantes de carácter
  - Ejemplo: LD A, ‘a’ ‘a’ corresponde a 0110 0001 Código ASCII de ‘a’

Reglas de sintaxis y directivas

- Identificador o símbolo:
  - Nombre que vamos a usar para identificar un valor.
  - Deben ser definidos en alguna parte del programa.
- Expresiones:
  - Los argumentos de una instrucción o directiva pueden ser expresiones
  - Pueden contener:
    - Constantes
    - Identificadores
    - Operaciones: NOT, AND, OR, XOR, +, -, *, /, MOD, …….
  - Son evaluadas po r el ensamblador y el resultado es lo que se utiliza en el programa objeto.
  - Ejemplo: Si el identificador “DIR” tiene el valor 0x8010.

```
LD  H, DIR / 256        equivale a  LD  H,0x80
LD  L, DIR MOD 256      equivale a  LD  L, 0x10
```

Reglas de sintaxis y directivas

- Un programa en ensamblador es una secuencia de sentencias.
- Cada sentencia puede ser de 2 tipos:
  - **Tipo Instrucción:** Contiene una instrucción del microprocesador
  - **Tipo Directiva**: Contiene una orden a ser interpretada por el ensamblador
- Formato de una sentencia:

  [etiqueta]: [instrucción o directiva] ; [comentario]

  - Los 3 campos son opcionales (entonces la línea vacía es una sentencia válida)
  - Los separadores pueden cambiar. Por ej. // en inicio del comentario
  - Algunos permiten /* comentarios en bloque que pueden ser multilínea */

Reglas de sintaxis y directivas

- Contador de posiciones:
  - Es una variable interna del ensamblador que refiere al lugar de memoria en que debe cargarse cada byte de código generado.
  - Se inicializa mediante una directiva ORG
  - Se va incrementando al ensamblar cada sentencia de tipo instrucción.
  - Las sentencias de tipo directiva NO lo incrementan (hay excepciones)
  - No confundir con el PC.
    - El PC existe en tiempo de ejecución. Es un registro del uP.
    - El Contador de posiciones existe en tiempo de compilación. Es una variable del compilador.

Reglas de sintaxis y directivas

- Campo [Etiqueta]:
  - Identificador que contiene la dirección de memoria indicada por el *Contador de Posiciones* previo a ser incrementado por la sentencia.
  - Se puede utilizar este identificador en cualquier sentencia, para referirse a esa dirección de memoria.
  - El compilador sustituye cada etiqueta por la dirección de memoria que contiene en todas las sentencias donde se la utiliza.
  - No puede haber 2 etiquetas con el mismo nombre.
- Campo [Comentarios]:
  - Se utiliza para hacer aclaraciones
  - No es tenido en cuenta por el ensamblador

DEBEN USAR COMENTARIOS

Reglas de sintaxis y directivas

- Campo [Instrucción o Directiva]
  - Instrucción:
    - Consta de: Mnemónico + parámetros (Ej LD B, 0x05)
    - Al ensamblar la instrucción, se obtiene el código máquina correspondiente y se incrementa el contador de posiciones en la cantidad bytes que ocupa la instrucción.
  - Directiva
    - Varían de un ensamblador a otro, así como la sintaxis.
    - No incrementan el contador de posiciones (hay excepciones).
    - Se van a enumerar las más comunes a continuación

Reglas de sintaxis y directivas
Directivas

- ORG (Origen)
  - Indica a partir de qué dirección debe cargarse el código.
  - Asigna valor al Contador de Posiciones
  - Sintaxis usuales:

```
ORG <Dir>
.org <Dir>
```

Donde <Dir> es una dirección de memoria y puede ser representada mediante una expresión.

  - Dos variantes
    - Absoluto
    - Relativo

Ejemplo:

```
        .org   0x1000
Inicio: LD  A, (nn)
        ……
```

Tendremos que:

Inicio = 0x1000

Reglas de sintaxis y directivas Directivas

- ORG (continuación)
  - Gnu assembler usado en el curso
    - Solamente relativo
    - Directivas ORG deben estar en orden creciente de direcciones en el archivo
    - Secciones
      - .text usualmente código
      - .data usualmente para variables
      - Son reubicables, se define al invocar al linker donde comienza cada una.

Reglas de sintaxis y directivas
Directivas

- EQU o EQUATE
  - Se utiliza para definir constantes
  - Sintaxis usuales

```
<símbolo> EQU <valor>
.equ <símbolo>,<valor>
```

```
Donde <Valor> es un número y puede ser representado
mediante una expresión.
```

  - Genera una correspondencia símbolo ↔ valor
  - Se puede utilizar este símbolo en cualquier sentencia, para referirse al valor que corresponde.
  - El compilador sustituye cada símbolo por su valor en todas las sentencias donde se lo utiliza.

Reglas de sintaxis y directivas
Directivas

- Define byte

```
<etiqueta>: DB [valor]
<etiqueta>: DEFB [valor]
<etiqueta>: .byte [valor]
```

Donde <Valor> es un número y puede ser representado mediante una expresión.

  - Se utiliza para reservar un byte de memoria
  - Incrementa en 1 *Contador de Posiciones* para dejar lugar para un byte
  - [valor] es opcional. Si se lo incluye, el lugar de memoria indicado por el contador de posiciones contiene [valor] en 8 bits.

- Gnu assembler:
  - Bug. Obliga a poner un valor aunque luego no se utilice

Reglas de sintaxis y directivas Directivas

- Define Word

```
<etiqueta>: DW [valor]
<etiqueta>: DEFW [valor]
<etiqueta>: .hword [valor]
```

  - Ídem DB pero reserva 2 bytes.
  - Incrementa en 2 el *Contador de Posiciones.*

- Define Storage

```
<etiqueta>: DS n [valor]
```

  - Ídem DB pero para n bytes.

Reglas de sintaxis y directivas Directivas

- Terminar traducción (END)
  - .end
  - El texto que viene después es ignorado por el traductor.
  - Gnu assembler:
    - Se debe terminar la línea con retorno de carro, de lo contrario no la procesa

- Incluir otro archivo (.include)
  - .include <nombre_archivo>
  - Intercala contenido de nombre_archivo
  - Cómo para tener una sola versión de código usado en diferentes archivos (por ej. laboratorio)

Ensamblado

- Tabla de símbolos
  - Pares símbolo-valor
  - Definidos mediante etiquetas o directivas
- Primera pasada
  - Se calcula contador de posiciones en cada línea
  - Se agregan a tabla de símbolos etiquetas y otros símbolos
- Segunda pasada
  - Opcode y parámetros según cartilla
  - Símbolos se sustituyen por su valor según tabla de símbolos
- Si las direcciones son relativas queda pendiente para el linker sumar dirección de comienzo
- Idem con símbolos externos

Ensamblado

- Ejemplo:

```
.equ CANTIDAD, 0x32
.equ SALIDA,   0x10

.text
.org 0x0100
   RETARDO:  LD B, CANTIDAD
             LD A, 1
             OUT (SALIDA), A
   LAZO:     DEC B
             JP NZ, LAZO
             LD A,0
             OUT (SALIDA), A
.end
```

Ensamblado

- Ejemplo:

Luego de la PRIMER pasada

```
.equ CANTIDAD, 0x32
.equ SALIDA,   0x10

.text
.org 0x0100
  RETARDO:   LD B, CANTIDAD
             LD A, 1
             OUT (SALIDA), A
  LAZO:      DEC B
             JP NZ, LAZO
             LD A,0
             OUT (SALIDA), A
.end
```

TABLA DE SÍMBOLOS	
CANTIDAD	0x32
SALIDA	0x10
RETARDO	0x0100
LAZO	0x0106

DIRECCION	CODIGO MAQ.
0100	
0102	
0104	
0106	
0107	
010A	
010C	

Ensamblado

- Ejemplo:

Luego de la SEGUNDA pasada

```
.equ CANTIDAD, 0x32
.equ SALIDA,   0x10

.text
.org 0x0100
  RETARDO:   LD B, CANTIDAD
             LD A, 1
             OUT (SALIDA), A
  LAZO:      DEC B
             JP NZ, LAZO
             LD A,0
             OUT (SALIDA), A
.end
```

TABLA DE SÍMBOLOS	
CANTIDAD	0x32
SALIDA	0x10
RETARDO	0x0100
LAZO	0x0106

DIRECCION	CODIGO MAQ.
0100	06 32
0102	3E 01
0104	D3 10
0106	05
0107	C2 06 01
010A	3E 00
010C	D3 10