Universidad Nacional Autónoma de México
Facultad de Ingeniería

Microprocesadores y Microcontroladores

Tema 4

Lenguaje ensamblador y el ensamblador

M. I. Christo Aldair Lara Tenorio

2025-1

Tabla de contenidos

Objetivo del tema

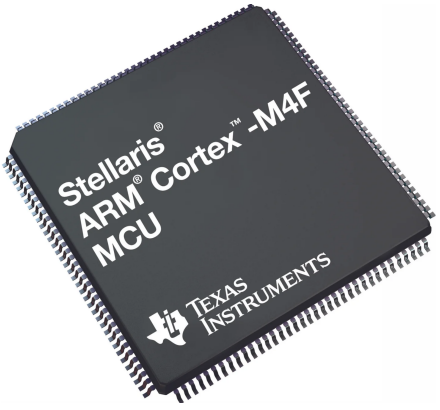
Objetivo general:
El alumno utilizará un lenguaje ensamblador y un ensamblador para desarrollar programas de aplicación.

Contenido:

4.1. Mnemónicos, programa fuente, programa objeto.

4.2. Ensambladores.

4.3. Directivas del ensamblador.



Lenguaje ensamblador

- Lenguaje de bajo nivel (muy cercano al lenguaje máquina).
- Utiliza abreviaciones de palabras en inglés (**mnemónicos**), como `LDR`, `STR`, `MOV`, `ADD`, etc.
- Diseñado para una arquitectura de procesador específica.
- Cada instrucción en ensamblador corresponde a una en lenguaje/código máquina.
- **Programa fuente** vs **Programa objeto**.

Dirección	Programa fuente	Programa objeto
0x2000 0200	`MOV R0, #0x1234 ; R0=0x1234`	F2412034
0x2000 0204	`MOV R1, #0x5678 ; R1=0x5678`	F2456178
0x2000 0208	`ORR R2, R0, R1 ; R2=R0+R1`	EA400201
0x2000 020C	`ADD R2, #5 ; R2=R2+5`	F1020205

Programa fuente
Lenguaje de bajo nivel (lenguaje ensamblador).

Programa objeto
Lenguaje binario entendible por el procesador (código máquina).

Sintaxis de una instrucción en lenguaje ensamblador

Las instrucciones en lenguaje ensamblador tienen 4 campos separados por espacios.

```
hola    MOV R0, #100    ; R0 = #100
```

En donde:

- Etiqueta
  Específica la posición en memoria (opcional).
- Opcode
  Específica el comando o instrucción a ejecutar.
- Operandos
  Específica la ubicación del dato que se va a operar.
- Comentarios
  Caracteres que serán ignorados por el procesador (opcional).

Lenguajes de alto nivel vs bajo nivel

Lenguaje de alto nivel	Lenguaje de bajo nivel
Fácil de entender y programar (presentan una sintaxis más cercana al lenguaje humano).	Difícil de entender y programar (presentan una sintaxis más cercana al código máquina).
Independiente de la arquitectura (compatible con cualquier máquina).	Depende totalmente de la arquitectura.
Requiere de un traductor (compilador) para convertir sus instrucciones a bajo nivel.	Utiliza el programa ensamblador para generar el código máquina.
Programación rápida.	Programación lenta.
No permite un control a bajo nivel de la máquina.	Permite el control total de los recursos de la máquina y los procesos que ejecuta.
Presenta un mayor nivel de abstracción (menor control sobre el hardware).	Mayor control sobre el hardware.
Menor eficiencia y rendimiento.	Mayor eficiencia y rendimiento.

Programar en ensamblador requiere que el desarrollador conozca la arquitectura del procesador, por lo que este tendrá la capacidad de evaluar, depurar y optimizar un código para alcanzar la **máxima eficiencia** (velocidad de ejecución) o el mínimo tamaño de memoria.

Programa ensamblador

Herramienta que funciona como traductor de un código en lenguaje ensamblador (programa fuente) para convertirlo al código máquina (programa objeto) y, de esta manera, sea entendible para el procesador.

- Traduce las sentencias/instrucciones del lenguaje ensamblador, a su equivalente en código máquina.
- El código máquina que se genera depende totalmente de la arquitectura del procesador, por lo que está diseñado para una sola arquitectura y no funcionará directamente en otra.
- Una instrucción en lenguaje ensamblador genera una instrucción en código máquina.

```
MOV R0, #0x1234 ⟶ F2412034
MOV R1, #0x5678 ⟶ F2456178
ORR R2, R0, R1  ⟶ EA400201
ADD R2, R2, #5  ⟶ F1020205
```

Directivas del ensamblador

Instrucciones especiales que le brindan información al programa ensamblador sobre cómo ensamblar el código fuente.

- Se utilizan para asistir y controlar el proceso del ensamblador.
- Las directivas no son parte del conjunto de instrucciones y no son traducidas por el programa ensamblador.
- Son procesadas por el ensamblador durante la etapa de ensamblado.
- Las directivas cambian la forma en la que el código fuente es ensamblado.
- Ayudan a estructurar el programa al especificar cómo se deben organizar las secciones de código y de datos.
- No tienen impacto directo en la ejecución del programa.

Directivas del ensamblador

`.global`	Declara un archivo para que sea accesible desde otros módulos/archivos. • En *Code Composer Studio*, es necesario que un programa tenga una función principal (main) global: `.global main` • Con la directiva `.asg` se le puede asignar otro nombre al archivo principal (main): `'.asg "main", ejemplo'`, en donde el **main** ahora será declarado con el nombre **ejemplo**.
`.data`	Define el segmento de datos, dando inicio al código en donde se declaran las variables y constantes que se utilizarán en el programa.
`.text`	Define el segmento del código del programa, dando inicio a las instrucciones que serán ejecutadas por el procesador que se almacenarán en la FLASH ROM. • De manera predeterminada, la directiva `.text` está declarada al inicio de un ensamble, excepto cuando se utiliza la directiva `.data` previamente.

Directivas del ensamblador

`.field`	Define y reserva un espacio para un campo de un tamaño específico dentro de estructuras o registros. • El tamaño del campo se específica en bits. • Permite trabajar con datos que ocupan menos de un byte (característica utilizada con frecuencia en microcontroladores).
`.equ`	Asigna un valor constante a un nombre o símbolo que se utilizará después en el código. • Facilita el entendimiento del código al usar nombres en lugar de números, además de que si se actualiza su valor, el cambio afecta a todo el código. • Disminuye la probabilidad de cometer errores durante la programación. • Permite la definición de valores basados en expresiones.
`.end`	Indica el final de un archivo en código fuente, diciéndole al programa ensamblador que no hay más código que procesar después de esa línea, por lo que se ignorará al código que se encuentre después de esta directiva.

Estructura básica de un programa en lenguaje ensamblador

```
01          .global main          ; Declaración del archivo con la función principal (main)
02
03          .data                 ; Sección para escribir en memoria de datos
04
05          .text                 ; Sección para escribir el código ejecutable (instrucciones)
06
07  POINTER .field 0x20000000, 32
08  DOS     .equ 0x02
09
10  loop    ADD R1, R0            ; Sección de etiquetas (subrutinas)
11          B   loop
12
13  main:                         ; Inicio del código principal
14          MOV  R0, #0x12
15          MOV  R1, #0x01
16
17          .end                  ; Final del archivo de código fuente
```