Universidad autónoma de baja California

Ingeniería en computación

Microcontroladores

Practica 2 : programación en lenguaje ASM ATmega 2560

Erik garcia Chávez 01275973

Jesús Adán Garcia López

21 de febrero del 2025

Arquitectura interna

Arquitectura Harvard Modificada

- Memorias separadas para programa y datos:
- Flash (Programa): 256 KB (organizada en 128K x 16 bits), el doble que el ATmega1280.
- SRAM (Datos): 8 KB para datos volátiles.
- EEPROM: 4 KB para datos no volátiles.

CPU de 8 bits RISC

Registros clave:

- 32 registros de propósito general (R0–R31) de 8 bits.
- Registros de I/O: Acceso directo a periféricos mediante direcciones mapeadas.
- SREG (Registro de Estado): Flags como Carry (C), Zero (Z), Overflow (V), etc.
- Contador de Programa (PC): 18 bits (para direccionar 256 KB de Flash).
- Stack Pointer (SP): 16 bits (maneja la SRAM).
- ALU (Unidad Aritmético-Lógica): Opera en 8 bits, con soporte para operaciones aritméticas, lógicas y de desplazamiento.

Periféricos Integrados

Timers/Contadores:

- 4 timers de 16 bits (Timer1, Timer3, Timer4, Timer5).
- 2 timers de 8 bits (Timer0, Timer2).
- Conversor ADC: 16 canales con resolución de 10 bits.

Comunicación Serial:

- 4 USARTs para UART.
- SPI e I²C (TWI).

PWM: 15 canales de modulación por ancho de pulso.

Puertos de I/O: 86 líneas digitales (54 I/O + 32 con funciones especiales).

Interrupciones: Hasta 54 fuentes con vectorización y prioridad programable.

Gestión de Energía

Múltiples modos: Idle, Power-save, Standby, y ADC Noise Reduction.

Conjunto de Instrucciones

Aritméticas y Lógicas:

- Operaciones básicas: ADD, SUB, ADC, SBC.
- Lógicas: AND, OR, EOR (XOR), COM (complemento).
- Multiplicación: MUL (8x8 bits, resultado en 16 bits, 2 ciclos).
- Incremento/decremento: INC, DEC.

Transferencia de Datos:

- Entre registros: MOV, MOVW
- Carga inmediata: LDI Rd, K

Acceso a memoria:

- Indirecto: LD Rd, X/Y/Z o ST X/Y/Z, Rr (con pre/post-incremento).
- Directo: LDS Rd, addr (carga desde SRAM) y STS addr, Rr.
- I/O: IN Rd, P y OUT P, Rr.

Control de Flujo:

- Saltos: RJMP (offset relativo de ±2KB), JMP (dirección absoluta, 3 ciclos).
- Subrutinas: CALL (llamada), RET (retorno).
- Bifurcaciones condicionales: BREQ, BRNE, BRCS, etc. (basadas en el SREG).

Manipulación de Bits:

- En I/O: SBI (set bit), CBI (clear bit).
- En registros: BSET (set flag), BCLR (clear flag).
- Saltos condicionales por bits: SBIS (salta si bit está set), SBIC (salta si bit está clear).

Instrucciones Especiales:

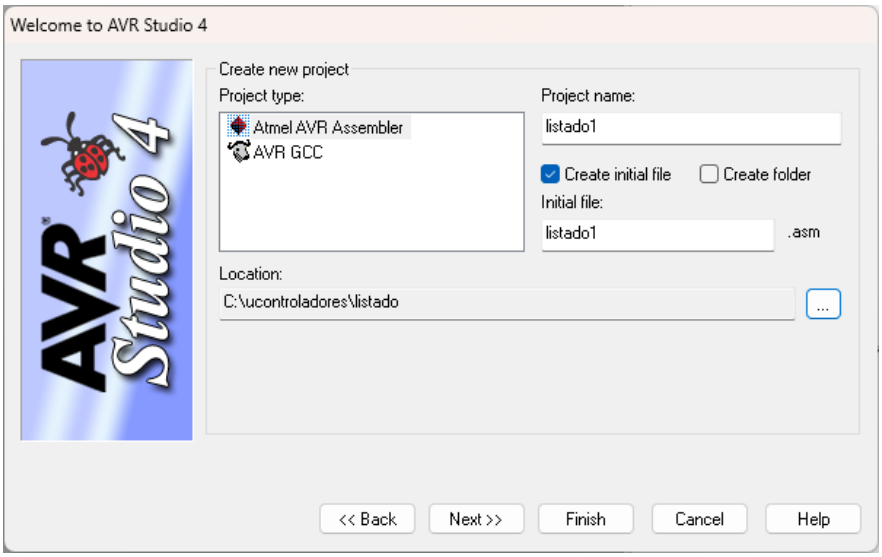
- SLEEP (entra en modo bajo consumo).
- WDR (reinicia el watchdog timer).
- NOP (sin operación, 1 ciclo).

Modos de Direccionamiento

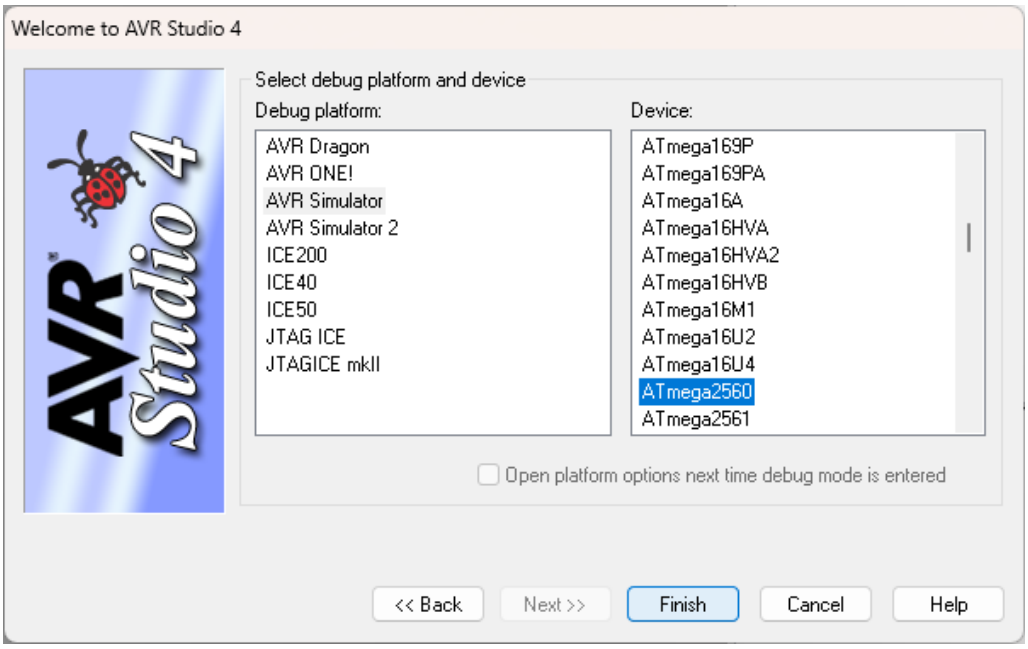
- Inmediato: Ej: LDI R16, 255.
- Directo a SRAM: Ej: LDS R0, 0x500.
- indirecto con registros X/Y/Z: Ej: LD R1, X+.
- Relativo: Para saltos cortos (RJMP).
- Indexado con desplazamiento: Ej: ST Y + 10, R16.

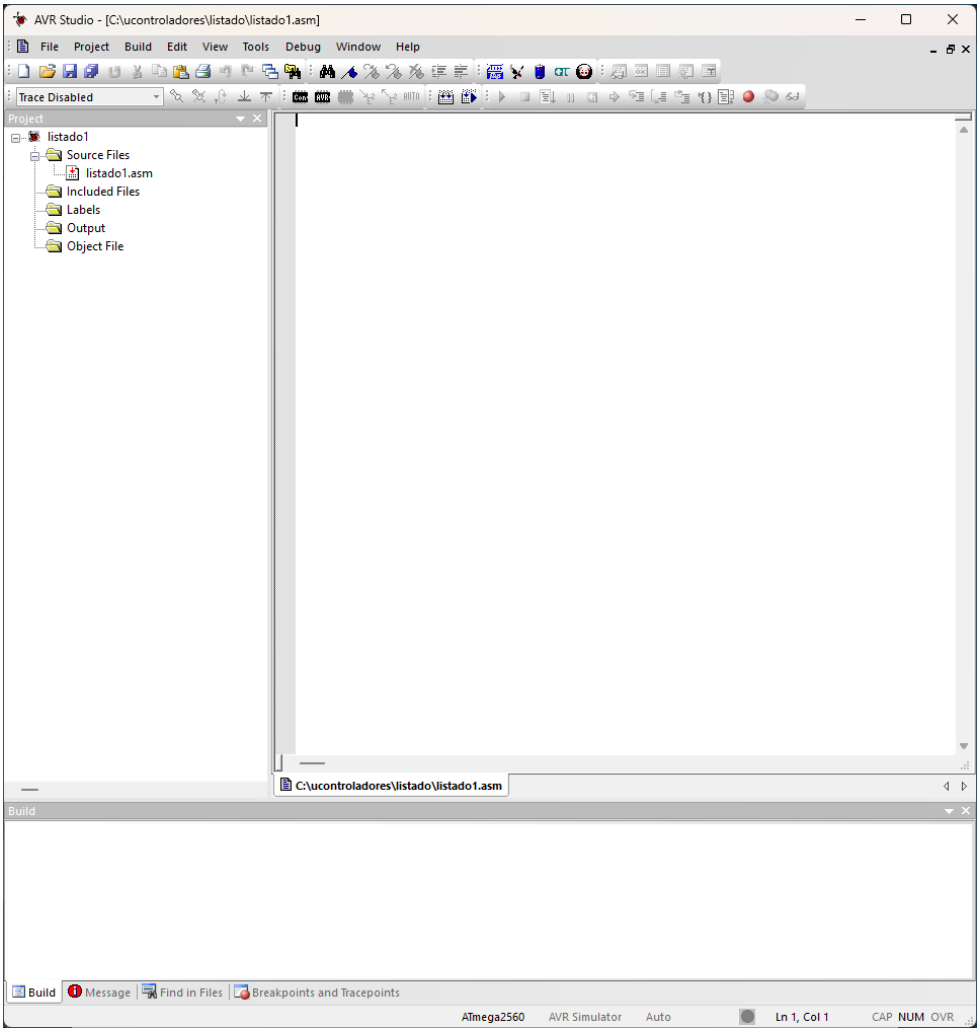
Creación de proyecto en Atmel studio

3) Crear en la raíz (C:\) una carpeta llamada uPuC y dentro de esta otra llamada Prac6 quedando la ruta como: “C:\uPuC\Prac6”.

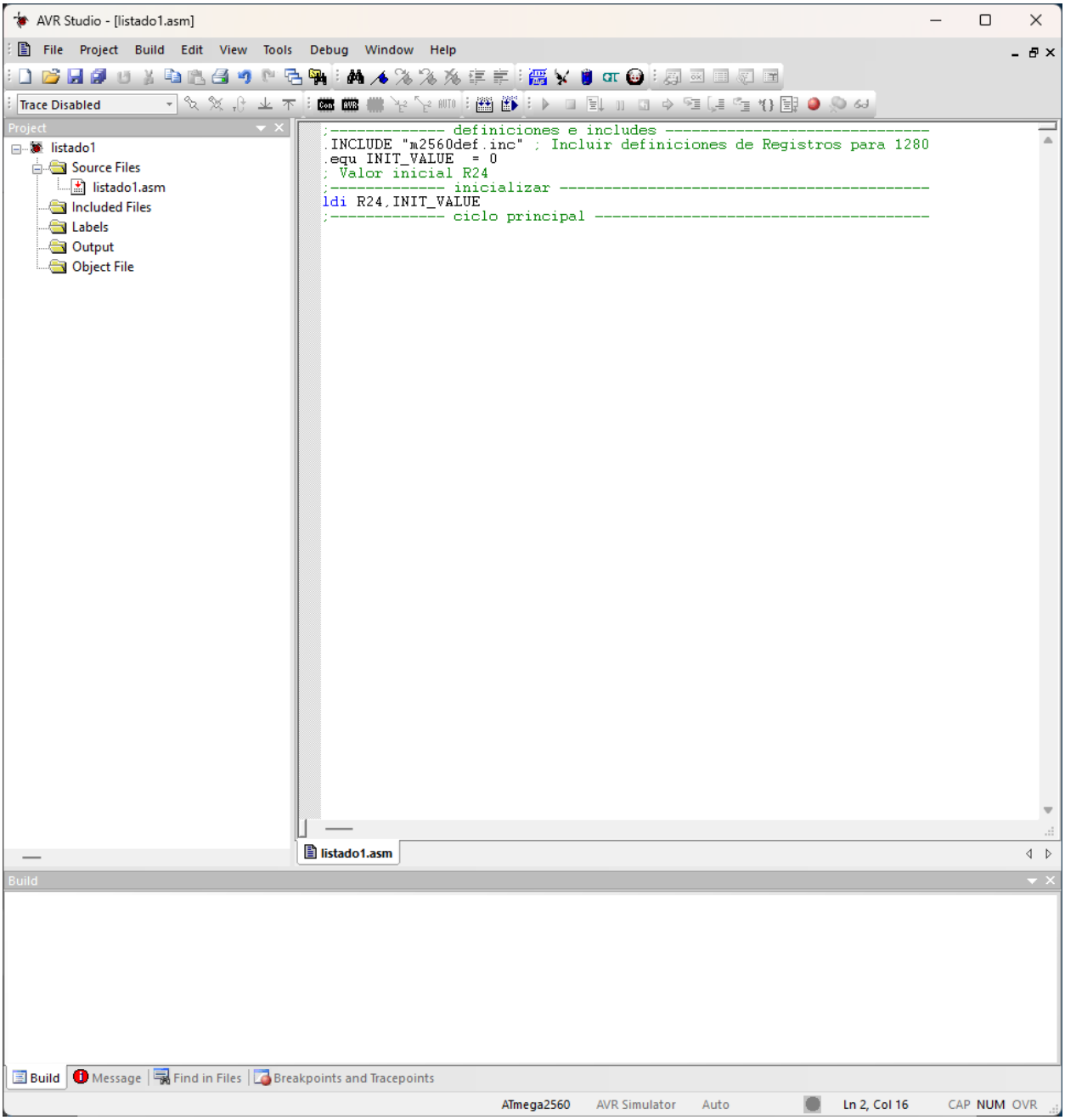


Ahora seleccione AVR Simulator como la plataforma de depuración (Debug Platform) y el dispositivo (Device) a utilizar el ATmega1280 y presione Finish .

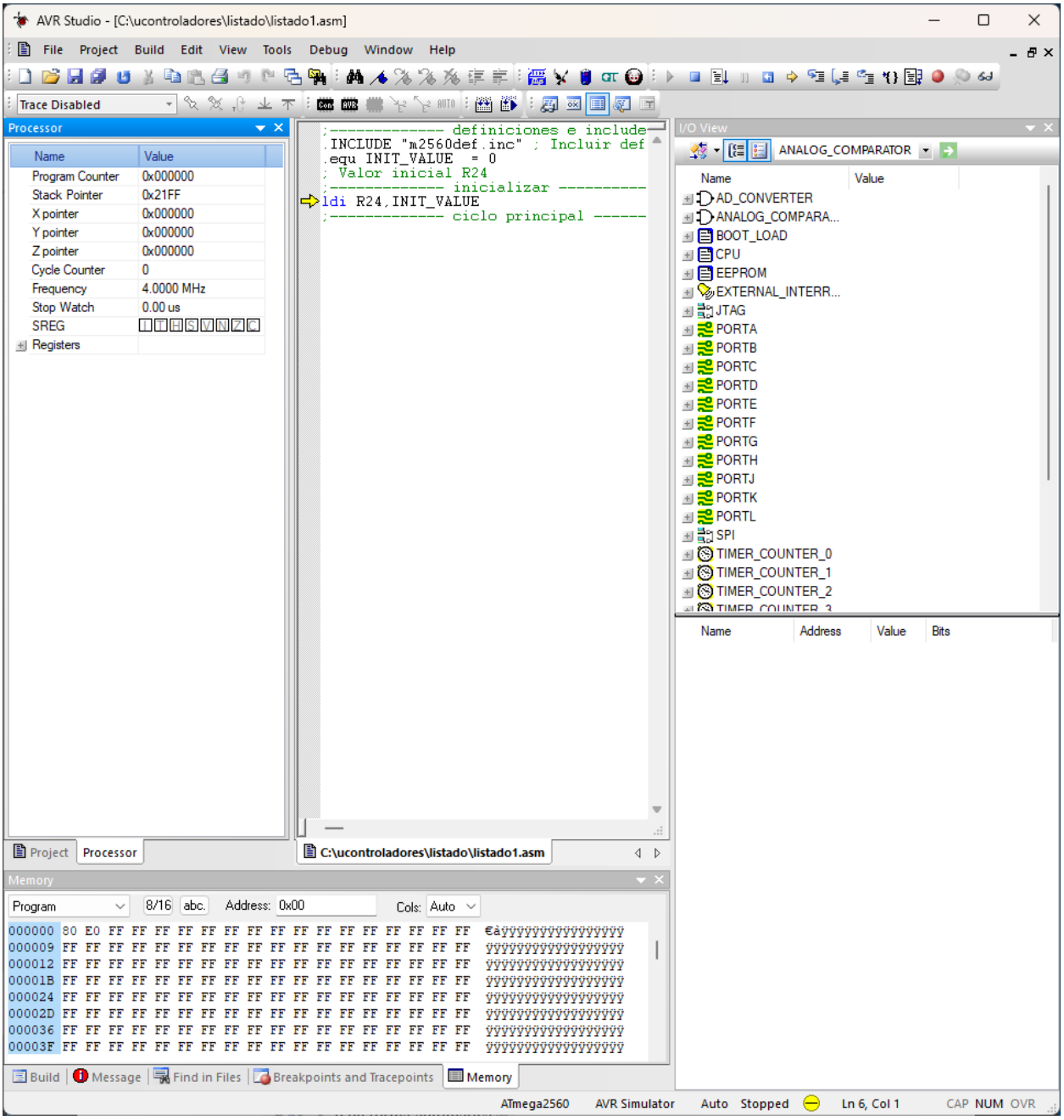



AVR Studio - [C:\ucontroladores\listado\listado1.asm]
File Project Build Edit View Tools Debug Window Help
Trace Disabled
Project
listado1
Source Files
listado1.asm
Included Files
Labels
Output
Object File
C:\ucontroladores\listado\listado1.asm
Build
Build
Message
Find in Files
Breakpoints and Tracepoints
ATmega2560
AVR Simulator
Auto
Ln 1, Col 1
CAP NUM OVR

Listado 1:



Compilación:



Comentarios y conclusiones:

En la práctica se realizó un decode, el cual se tiene un arreglo con valores hexadecimales los cuales representar operaciones, por lo que se necesita hacer un programa que imprima a que operación se está haciendo referencia y a que registros, puede usar más estructuras que antes no había usado como las “unión”, los “enum” así como los espacios de bits, en C, algo que será de gran utilizar para sistemas embebidos dado que tan solo se usa los bits que se necesitas y permite que se use en

un solo tipo de dato como un entero varios espacios de bits, y no crear nuevas variables y se cuida mucho mas la memoria que en sistemas embebidos es muy poca y el optimizar los recursos es en algo que debe de estar en la mente de cualquier desarrollador de estos sistemas.