```
// ARDUINO OBSTACLE AVOIDING CAR //

// Before uploading the code, you have to
install the necessary libraries.
// AFMotor Library: https://
learn.adafruit.com/adafruit-motor-shield/
library-install
// NewPing Library: https://github.com/
livetronic/Arduino-NewPing
// Servo Library: https://github.com/
arduino-libraries/Servo.git

// To Install the libraries go to sketch >>
Include Library >> Add .ZIP File >> Select
the Downloaded ZIP files From the Above
links //

#include <AFMotor.h>
#include <NewPing.h>
#include <Servo.h>
```

```
// Ultrasonic Sensor Pin Definitions
#define TRIG_PIN A0
#define ECHO_PIN A1
#define MAX_DISTANCE 200

// Maximum Speed for DC Motors
#define MAX_SPEED 190

// Offset for Maximum Speed (to avoid
sudden changes in speed)
#define MAX_SPEED_OFFSET 20

// Create NewPing object to interface with
the ultrasonic sensor
NewPing sonar(TRIG_PIN, ECHO_PIN,
MAX_DISTANCE);

// Create four DC motor objects to control
the car's movement
AF_DCMotor motor1(1, MOTOR12_1KHZ);
AF_DCMotor motor2(2, MOTOR12_1KHZ);
AF_DCMotor motor3(3, MOTOR34_1KHZ);
AF_DCMotor motor4(4, MOTOR34_1KHZ);

// Create a Servo object to control the
ultrasonic sensor's position
Servo myservo;

// Variable to track the car's direction
(forward or backward)
boolean goesForward = false;

// Variable to store the distance measured
by the ultrasonic sensor
int distance = 100;

// Variable to control the gradual increase
in motor speed
int speedSet = 0;

// Setup Function: Runs once when the
Arduino is powered on or reset
```

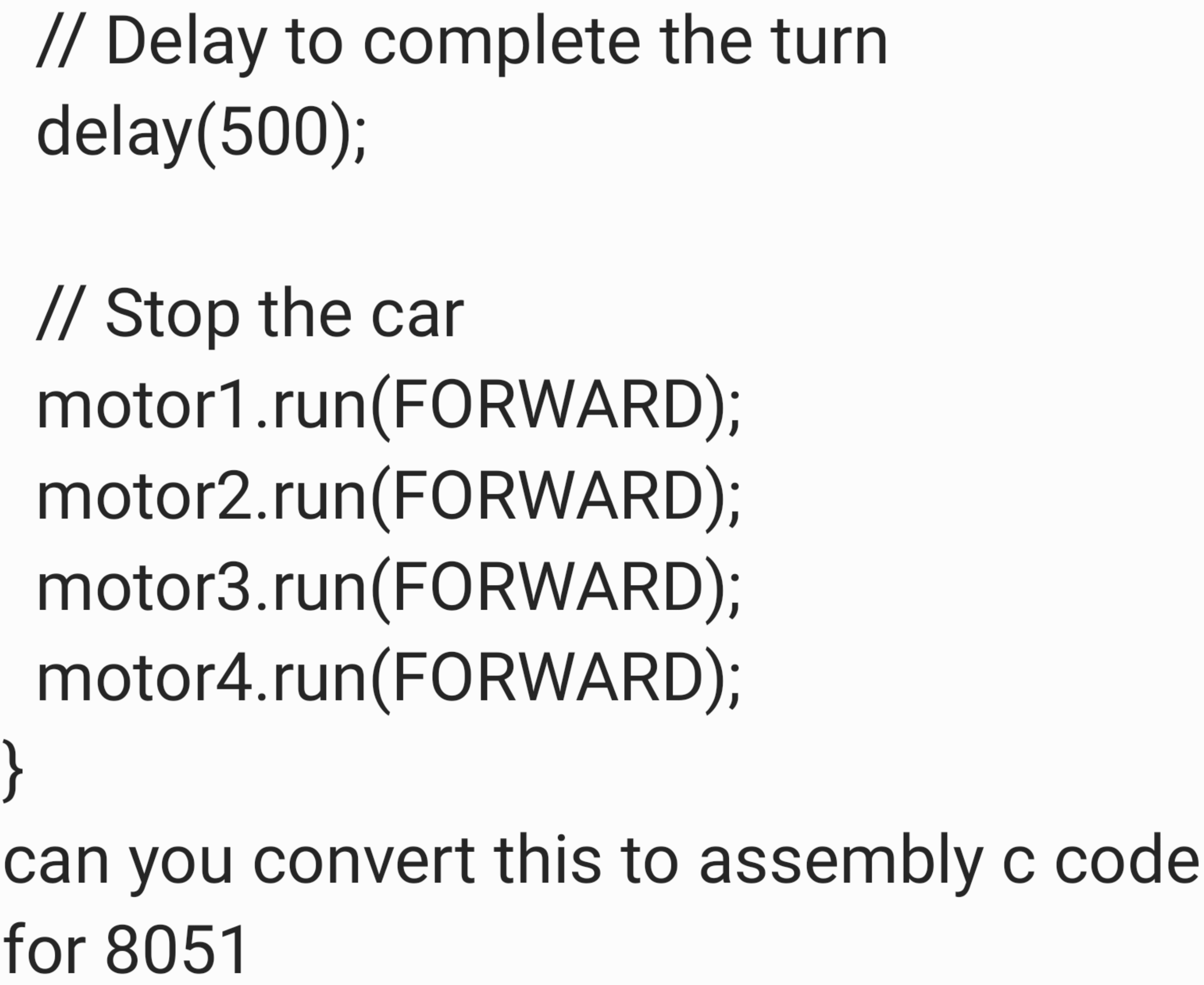
```
void setup() {
  // Attach the servo to pin 10 and set its initial position to 115 degrees
  myservo.attach(10);
  myservo.write(115);

  // Delay for 2 seconds for the servo to settle
  delay(2000);

  // Calibrate the ultrasonic sensor by taking multiple distance readings
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
}
```

```
// Loop Function: Runs repeatedly as long as the Arduino is powered on
void loop() {
  int distanceR = 0;
  int distanceL = 0;

  // Delay to stabilize sensor readings
  delay(40);

  // Check if there's an obstacle within the specified distance
  if (distance <= 15) {
    // Obstacle detected, stop the car
    moveStop();
    delay(100);

    // Move the car backward to create distance from the obstacle
    moveBackward();
    delay(300);
  // Stop the car
  moveStop();
  delay(200);

  // Look right and left to decide which
direction to turn
  distanceR = lookRight();
  delay(200);
  distanceL = lookLeft();
  delay(200);

  // Decide the turning direction based on
the longer distance
  if (distanceR >= distanceL) {
    turnRight();
    moveStop();
  } else {
    turnLeft();
    moveStop();
  }
  } else {
    // No obstacle detected, move the car forward
    moveForward();
  }

  // Read the distance from the ultrasonic sensor for the next iteration
  distance = readPing();
}

// Helper function: Look right using the servo and return the distance measured
int lookRight() {
  // Rotate the servo to the right position
  myservo.write(50);

  // Delay to stabilize the servo position
  delay(500);

  // Take a distance reading
  int distance = readPing();

  // Delay to stabilize the servo position
  delay(100);

  // Reset the servo to the center position
  myservo.write(115);

  // Return the distance measured
  return distance;
}

// Helper function: Look left using the servo and return the distance measured
int lookLeft() {
  // Rotate the servo to the left position
  myservo.write(170);

  // Delay to stabilize the servo position
  delay(500);
  // Take a distance reading
  int distance = readPing();

  // Delay to stabilize the servo position
  delay(100);

  // Reset the servo to the center position
  myservo.write(115);

  // Return the distance measured
  return distance;
}

// Helper function: Read distance from the
ultrasonic sensor
int readPing() {
  // Delay to stabilize the sensor readings
  delay(70);

  // Get the distance in centimeters from
the ultrasonic sensor
```

```
  int cm = sonar.ping_cm();

  // If the sensor returns 0 (indicating an error), set distance to a large value (250)
  if (cm == 0) {
    cm = 250;
  }

  // Return the measured distance
  return cm;
}

// Helper function: Stop all the motors
void moveStop() {
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
}

// Helper function: Move the car forward
```

```
with gradually increasing speed
void moveForward() {
  // Check if the car is already moving
forward
  if (!goesForward) {
    goesForward = true;

    // Set all motors to move forward
    motor1.run(FORWARD);
    motor2.run(FORWARD);
    motor3.run(FORWARD);
    motor4.run(FORWARD);

    // Gradually increase the speed to avoid
overloading the batteries
    for (speedSet = 0; speedSet <
MAX_SPEED; speedSet += 2) {
      motor1.setSpeed(speedSet);
      motor2.setSpeed(speedSet);
      motor3.setSpeed(speedSet);
      motor4.setSpeed(speedSet);
      // Delay to achieve gradual speed increase
      delay(5);
    }
  }
}

// Helper function: Move the car backward with gradually increasing speed
void moveBackward() {
  // Set the car's direction to backward
  goesForward = false;

  // Set all motors to move backward
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);

  // Gradually increase the speed to avoid
```

```
overloading the batteries
  for (speedSet = 0; speedSet <
MAX_SPEED; speedSet += 2) {
    motor1.setSpeed(speedSet);
    motor2.setSpeed(speedSet);
    motor3.setSpeed(speedSet);
    motor4.setSpeed(speedSet);

    // Delay to achieve gradual speed
increase
    delay(5);
  }
}

// Helper function: Turn the car to the right
void turnRight() {
  // Turn the motors in opposite directions
to make the car turn right
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);

  // Delay to complete the turn
  delay(500);

  // Stop the car
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}

// Helper function: Turn the car to the left
void turnLeft() {
  // Turn the motors in opposite directions
to make the car turn left
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
```



```
  // Delay to complete the turn
  delay(500);

  // Stop the car
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}
```

can you convert this to assembly c code for 8051