Assignment 01/08

Author: ThanhTH10

Date: 01/08/2024

1. Write simple hello world program ( using std::cout //declarative method)

#include <iostream>

using std::cout;

int main()

{

    cout << "Hello world";

}

2. Application of typedef in C in real time scenario with respect to Embedded register space.

#include <stdint.h>

#define PERIPHERAL\_BASE 0x3F000000 // For Pi 3

#define SYSTEM\_TIMER\_OFFSET 0x3000

#define ST\_BASE (PERIPHERAL\_BASE + SYSTEM\_TIMER\_OFFSET)

// Define the timer register structure

typedef struct

{

    volatile uint32\_t CS;  // Control and Status

    volatile uint32\_t CLO; // Counter Lower 32-bits

    volatile uint32\_t CHI; // Counter Higher 32-bits

    volatile uint32\_t C0;  // Compare 0

    volatile uint32\_t C1;  // Compare 1

    volatile uint32\_t C2;  // Compare 2

    volatile uint32\_t C3;  // Compare 3

} system\_timer\_t;

int main()

{

    // Cast the ST\_BASE address to the timer register structure

    system\_timer\_t \*timer = (system\_timer\_t \*)ST\_BASE;

    // Access and display the timer registers

    printf("Timer Registers:\n");

    printf("CS: 0x%08X\n", timer->CS);

    printf("CLO: 0x%08X\n", timer->CLO);

    printf("CHI: 0x%08X\n", timer->CHI);

    printf("C0: 0x%08X\n", timer->C0);

    printf("C1: 0x%08X\n", timer->C1);

    printf("C2: 0x%08X\n", timer->C2);

    printf("C3: 0x%08X\n", timer->C3);

    return 0;

}

3.Write C program code to display timer registers with below address.

#define PERIPHERAL\_BASE 0x3F000000 // For Pi 3

#define SYSTEM\_TIMER\_OFFSET 0x3000

#define ST\_BASE (PERIPHERAL\_BASE + SYSTEM\_TIMER\_OFFSET)

// Sytem Timer Registers layout

typedef struct {

uint32\_t control\_and\_status;

uint32\_t counter\_low;

uint32\_t counter\_high;

uint32\_t compare\_0;

uint32\_t compare\_1;

uint32\_t compare\_2;

uint32\_t compare\_3;

} system\_timer\_t;

#include <stdio.h>

#include <stdint.h>

#define PERIPHERAL\_BASE 0x3F000000 // For Pi 3

#define SYSTEM\_TIMER\_OFFSET 0x3000

#define ST\_BASE (PERIPHERAL\_BASE + SYSTEM\_TIMER\_OFFSET)

// System Timer Registers layout

typedef struct

{

    uint32\_t control\_and\_status;

    uint32\_t counter\_low;

    uint32\_t counter\_high;

    uint32\_t compare\_0;

    uint32\_t compare\_1;

    uint32\_t compare\_2;

    uint32\_t compare\_3;

} system\_timer\_t;

int main()

{

    // Cast the ST\_BASE address to the timer register structure

    system\_timer\_t \*timer = (system\_timer\_t \*)ST\_BASE;

    // Display the timer registers

    printf("Timer Registers:\n");

    printf("Control and Status: 0x%08X\n", timer->control\_and\_status);

    printf("Counter Low: 0x%08X\n", timer->counter\_low);

    printf("Counter High: 0x%08X\n", timer->counter\_high);

    printf("Compare 0: 0x%08X\n", timer->compare\_0);

    printf("Compare 1: 0x%08X\n", timer->compare\_1);

    printf("Compare 2: 0x%08X\n", timer->compare\_2);

    printf("Compare 3: 0x%08X\n", timer->compare\_3);

    return 0;

}

4. Explore the intermediate file generation with g++ -save-temps xxx.cpp

***The -save-temps option in g++ generates intermediate files during the compilation process:***

.i (Preprocessor Output): Preprocessed source code with preprocessor directives expanded.

.s (Assembly Code): Assembly code generated from the preprocessed source.

.o (Object File): Compiled binary representation of the source code.

***To explore:***

Create a simple C++ file, e.g., example.cpp.

#include <iostream>

int main()

{

    std::cout << "Hello, World!" << std::endl;

    return 0;

}

Compile with -save-temps: g++ -save-temps example.cpp.

Inspect the generated files:

example.i: Preprocessor output.

example.s: Assembly code.

example.o: Object file.

5. Implement bool data type in C – hint use typedef or enum

#include <stdio.h>

#define false 0

#define true 1

int main()

{

    bool is\_sunny = true;

    bool is\_raining = false;

    printf("Is it sunny? %s\n", is\_sunny ? "Yes" : "No");

    printf("Is it raining? %s\n", is\_raining ? "Yes" : "No");

    return 0;

}