Übung 2

Floating Point Unit

Aufgabe 2.1 Berechnung sqrtf() ohne FPU

In dieser Aufgabe soll die Zeit gemessen werden, welche die CPU auf dem Leguan-Board braucht, um die Funktion sqrtf() (Quadratwurzel für eine Float-Zahl) zu berechnen. Für die Berechnung soll nicht die FPU verwendet werden, sondern die Berechnung wird in Software ausgeführt. In C lautet der Code wie folgt:

```
float res;                  /* result of sqrt */
float value = 25.0F;        /* operand for sprt() */
uint32_t i;                 /* loop variable */

    ...
    res = sqrtf(value);
    ...
```

Um die Zeit für die Ausführung der Funktion sqrtf() zu messen, gibt es zwei Ansätze:

a) Sie setzen vor dem Aufruf von sqrtf() einen Ausgang des Microcontrollers (ein GPIO). Nach der Funktion sqrtf() löschen Sie diesen Ausgang. Mit einem Oszilloskop können Sie die Zeit nun messen.

b) Sie verwenden einen Timer des Microcontrollers und messen die Zeit.

   Die aktuelle “Systemzeit“ des Boards können Sie mit Hilfe der Funktion “HAL_GetTick();“ auslesen. Beim Starten des Boards wird die Systemzeit auf 0 initialisiert und anschliessend jede Millisekunde inkrementiert. Sie können somit vor und nach einer Berechnung die Systemzeit abfragen und aufgrund der Differenz die Ausführungszeit in Millisekunden berechnen.

   Um die zeitliche Auflösung zu erhöhen können Sie den Aufruf von sqrtf() in einer Schleife mehrfach ausführen und am Schluss die gemessene Zeit durch die Anzahl Schleifen dividieren. Beispiel: Sie führen die Schleife 1’000’000 Mal aus und erhalten eine Systemlaufzeit von 10ms. Dann ist die Zeit für eine Schleife 10ns. Beachten Sie, dass Sie korrekterweise auch die Zeit für die Ausführung der Schleife selbst berücksichtigen müssen. D.h. Sie können auch die Zeit messen, um eine leere Schleife auszuführen.

Da die Leguan-Library so programmiert ist, dass die FPU verwendet wird, brauchen wir für diese Berechnung eine andere Library-Funktion. Um Ihnen die Arbeit zu erleichtern, gibt es für diese Aufgabe ein Template (HWSWE_U2A1_A2_Template), welches Sie auf Moodle finden. Importieren Sie dieses Template in STM32CubeIDE. Im Source-Code des Templates befindet sich die Datei “mysqrtf.c“, welche die Funktion my_sqrtf() enthält (eine Version aus dem Internet von Stephen Canon). Rufen Sie aus Ihrem Programm diese Funktion zur Berechnung von sqrtf ohne FPU auf. Sie können die Ergabnisse der Zeitmessung auf dem Leguan-Board mit Hilfe der Log-Funktionen auf dem Display anzeigen.

Der Code aus dem Template lautet wie folgt:

```c
/* Defines */
/* number of loops to calculate sqrt, do not modify this value, ms will be ns
 in measured time.
*/
#define NBR_LOOPS (1000000)


/* function my_sqrtf() calculates sqrtf as a library function without using FPU */
extern float my_sqrtf(float x);

/**
 @brief        main function
 *
 This main function calculates the sqrt, once using a software library, once
 using the FPU, and displays the measured time.
 *
 @param        none
 @return       always 0
*/
__attribute__((optimize("0")))  /* Make sure the compiler doesn't optimize away the
   loops */

int main(void)
{
    uint32_t start_tick;        /* tick count when starting calculation */
    uint32_t end_tick;          /* tick count when calculation done */
    uint32_t time_loop;         /* measures time for loop only */
    float res;                  /* result of sqrt */
    float value = 25.0F;        /* operand for sprt() */
    uint32_t i;                 /* loop variable */

    /* Initialize Hardware */
    CUBEMX_Init();
    /* Initialize Leguan board */
    LEGUAN_Init();

    /* Set logging output destination to be the LCD */
    LOG_SetDestination(LCD_Stream);


    /* Main loop */
    for (;;) {
        /* measure time to do a loop NBR_LOOPS times */
        start_tick = HAL_GetTick();
        for (i=0; i<NBR_LOOPS; i++) {
            /* do nothing, just to get loop time */
        }
        end_tick = HAL_GetTick();
        time_loop = end_tick - start_tick;
        LOG_Info("Time to do a loop in ns: %d", time_loop);

        /* measure sqrtf() using FPU and print result */
        /* put your code here */
        asm volatile("vsqrt.f32 %0, %1" : "=w" (res) : "w" (value));

        /* measure sqrtf() using library function and print result */
        /* put your code here */
```

```
        res = my_sqrtf(value);
    }

    return (0);
}
```

Aufgabe 2.2 Berechnung sqrtf() mit FPU

In dieser Aufgabe soll dieselbe Berechnung mit Hilfe der FPU ausgeführt werden. Um die Berechnung mit der FPU auszuführen, müssen Sie anstelle der Zeile „res = sqrtf(value)“ folgende Zeile einfügen:

```
asm volatile("vsqrt.f32 %0, %1" : "=w" (res) : "w" (value));
```

Dies ist eine sogenannte Inline-Assembler Zeile. Hier wird die Assembler-Instruktion vsqrt.f32 aufgerufen, welche auf der FPU die geforderte Funktion ausführt. Mehr dazu erfahren Sie später im Unterricht.

Machen Sie dieselbe Zeitmessung wie schon bei Aufgabe 2.1 mit Hilfe der Systemzeit. Wie viel wird die Ausführung der Berechnung mit Hilfe der FPU schneller?

Aufgabe 2.3 Verifikation der Ergebnisse

Prüfen Sie, ob das Ergebnis der Berechnung mit FPU realistisch ist. Gehen Sie wie folgt vor:

- Auf Moodle finden Sie die Application Note AN4044 von ST Microelectronics zur FPU (Datei STM32_FPU_AN.p
  In diesem Dokument finden Sie Informationen, wie viele Clock-Zyklen für eine SQRT-Berechnung erforderlich sind. Mit Hilfe der Taktrate können Sie die Zeit berechnen.
- Die Inline-Assembler-Zeile “asm volatile(“vsqrt....“);“ wird mehrere Assembler-Instruktionen beinhalten. Wenn Sie im Debugger auf dieser Zeile einen Breakpoint setzen, können Sie im “Disassembly-Fenster“ rechts den vom Compiler erzeugten Code einsehen. Beachten Sie auch die Ausführungszeiten dieser zusätzlichen Instruktionen.
- Vergeleichen Sie die so berechnete Zeit für die Ausführung der SQRT-Instruktion mit den gemessenen Zeiten. Sind diese nachvollziehbar?