EECS 388 Laboratory Exercise #08
Assembly Language

Gary J. Minden

Laboratory starts the week of: November 26, 2018
Laboratory ends on Thursday, December 6, 2018 (Stop Day)

1 Introduction

In this lab you will incorporate an assembly language subroutine into a FreeRTOS task. Templates for the main FreeRTOS routine, a task, and an assembly language program will be provided to you. You will have to modify the task and assembly language program to compute the algebraic expressions assigned to you.

The algebraic expressions will have four input values. You will compute two intermediate values and then a result. The result may be dependent on the intermediate results. For example, your expressions might be:

```
X = A + (B + (C * D)) + A;
Y = A + B + C + D;
if ( X >= Y ) {
      return( X );
} else {
      return( Y );
}
```

Expressions can be assignment statements, if statements, or for statements. For this exercise we will not use constants.

2 References

The summary slides for the ARM ISA are at:

http://www.ittc.ku.edu/~gminden/Embedded_Systems/Lectures/EECS_388_ARM_CortexM3_ISA_B60329.pdf

The brief ARM description of the ISA is at:

http://www.ittc.ku.edu/~gminden/Embedded_Systems/PDFs/TI_Stellaris_ARM_InstrSetUm.pdf

The full description of the ARM assembler is at:

http://www.ittc.ku.edu/~gminden/Embedded_Systems/PDFs/TI_ARM_AssemblerManual.pdf

Finally, a description of the ARM compiler is at:

http://www.ittc.ku.edu/~gminden/Embedded_Systems/PDFs/TI_ARM_CompilerManual.pdf

Valvano has numerous examples of assembly language and describes assembly language starting on page 90.

3 Actions

1. Make a project copy of your initial FreeRTOS823_TM4C_Prototype project from Laboratory #01.
2. Import "Task_AsmCompute.c" into your project. This task carries out the computation and calls your assembly language routine.
3. Import "AsmCompute.asm" into your project.
4. Modify the main task to include a call to xCreateTask to add "Task_AsmCompute" to FreeRTOS.
5. Modify "Task_AsmCompute.c" and "AsmCompute.asm" to include the algebraic statements assigned to you.
6. Demonstrate your program.

The prototype source code for "Main_AsmCompute.c", "Task_AsmCompute.c", and "AsmCompute.asm" is available below.

The example code is also included below (formatting may vary).

The GTAs will assign individual computations.

```
        xTaskCreate( Task_AsmCompute, ( signed portCHAR * ) "AsmCompute", 512, NULL,
1, NULL );
```

```
//*****************************************************************************
//
//      Call assembly language subroutine
//
//           Author:        Gary J. Minden
//           Organization:  KU/EECS/EECS 388
//           Date:          2014-04-18
//           Version:       1.0
//
//           Purpose:       Call assembly language subroutine and check result
//
//           Notes:
//
//*****************************************************************************
//

#include "drivers/uartstdio.h"

#include "FreeRTOS.h"
#include "task.h"

#include "stdio.h"

//
//      Make the Task and Assembly language subroutines external (global)
//
extern void Task_AsmCompute( void *pvParameters );
extern int AsmCompute( int I1, int I2, int I3, int I4 );

//*****************************************************************************
//
//      Task initialization

void Task_AsmCompute( void *pvParameters ) {

//*****************************************************************************
        //
        //   Constants and Variables
#define MaxRange 10
        //
        //   Variables for testing assembly language subroutine
        //
        long int A, B, C, D;
        long int CValue, AsmValue;
        unsigned long int Delay_2000mS;

    Delay_2000mS = (2000 * configTICK_RATE_HZ ) / 1000;

    while ( 1 ) {

        for ( A = 0; A < MaxRange; A++ ) {
             for ( B = 0; B < MaxRange; B++ ) {
                  for ( C = 0; C < MaxRange; C++ ) {
                       for ( D = 0; D < MaxRange; D++ ) {

                  //
                  //   You will replace the following statement with the
                  //     algebraic expressions assigned to you.
                  //
```



```
                    CValue = A + B + C + D;

                    //
                    //   This is the invocation of your assembly language
                    //     subroutine to compute the same algebraic expression
                    //
                    AsmValue = AsmCompute( A, B, C, D );

                    //
                    //   Compare the C computation to the assembly language
                    //     computation and report discrepancies.
                    //
                    if ( CValue != AsmValue ) {
                         UARTprintf( "Boo! CValue: %d; AsmValue: %d\n", CValue,
AsmValue );
                    }
               }
               }
               }

               //
               //   Log progress to the serial port
               //
               UARTprintf( "A: %d\n", A );
          }

          //
          //   Report final results to indicate progress and completion.
          //
          UARTprintf( "Last CValue, AsmValue: %d, %d\n", CValue, AsmValue );

          //
          //   Delay for 2 seconds.
          //
          vTaskDelay( Delay_2000mS );

          }
}
```

```
;;*****************************************************************************
;;
;;      Assembly language subroutine
;;
;;           Author:        Gary J. Minden
;;           Organization:  KU/EECS/EECS 388
;;           Date:               2014-04-18
;;           Version:       1.0
;;
;;           Purpose:       Assembly language subroutine
;;
;;           Notes:
;;
;;*****************************************************************************
;;


;;
;;      This subroutine computes a value based on four input arguments.
;;      The arguments are assumed to be in CPU registers R0 - R3
;;        (aka A1 - A4).
;;      The result is returned in R0.
;;

;;      Declare sections and external references
```

```
            .global        AsmCompute     ; Declare entry point as a global symbol

;;      No constant data

;;      No variable allocation

;;      Program instructions

            .text                         ; Program section

AsmCompute:                               ; Entry point

;;
;;      Save necessary registers
;;
;;      Since this subroutine does not use registers other than R0 - R3,
;;        and we do not call another subroutine, we don't need to save
;;        any registers.
;;

;;
;;      Allocate local variables on the Stack
;;
;;      Since this subroutine does not have local variables,
;;        no Stack space need be allocated.
;;

;;
;;      For demonstration, this subroutine computes R0 + R1 + R2 + R3
;;        and returns the result in R0.
;;
;;      You should replace the following three instructions with the
;;        the instructions necessary to carry out your assigned
;;        algebraic assignment statements.
;;
            ADDS      R0,R1
            ADDS      R0,R2
            ADDS      R0,R3

;;
;;      Return from the subroutine with the result in R0
;;
            BX            LR                ; Branch to the program address in
the Link Register

            .end
```