EEE3002 Microprocessors Tutorial 1 (2014 Semester 1)

1) Find the two's complement representation for the following numbers, assuming that they are represented as a 16-bit number. Write the value in both binary and hexadecimal.
   a) −93
   b) 1034
   c) 492
   d) −1094

2) Using the smallest data size possible, either a byte, a halfword (16 bits), or a word (32 bits), convert the following values into two's complement representations:
   a) −18,304
   b) −20
   c) 114
   d) −128

3) Convert the following hexadecimal values to base ten:
   a) 0xFE98
   b) 0xFEED
   c) 0xB00
   d) 0xDEAF

4) Convert the following decimal numbers into hexadecimal
   a) 256
   b) 1000
   c) 4095
   d) 42

5) Convert the following fractions into decimal numbers
   a) Binary number : 101.111
   b) Hexadecimal number : 101.111

6) Convert the following numbers into binary numbers
   a) Decimal number: 8.625
   b) Hexadecimal number : A1.E8

7) List the advantages of using the two's complement representation over the more intuitive sign-magnitude representation.

EEE3002 Microprocessors Tutorial 2 (2014 Sem 1)

1) Convert the following decimal numbers into IEEE 754 single-precision format and write your answers in hexadecimal form.
   a) 0.1743129
   b) −31.5
   c) 8E−39

2) Add the following 8-bit binary number pairs using 8-bit addition (result stored in eight bits only) and comment on whether there is an overflow if they were both unsigned numbers or they were both signed numbers.
   a) 0011 1111 and 1101 0011
   b) 0111 1111 and 0111 0011
   c) 1000 1111 and 1101 0011

3) How many modes does the ARM7TDMI processor have? Name the modes also. How many states does it have?

4) What are the standard use of register r13, r14 and r15?

5) If the ARM processor encounters an undefined instruction, from what address will it begin fetching instructions after it changes to Undefined mode? What about a reset?

6) How many stages does the ARM7TDMI pipeline have? Name them.

7) Explain the current program state if the CPSR had the value 0x700000D3

EEE3002 Microprocessors Tutorial 3 (2014 Sem 1)

1) The factorial program is given as follows:

```
            AREA Prog2, CODE, READONLY
            ENTRY
            MOV       r6, #10       ; load 10 into r6
            MOV       r4, r6        ; copy n into a temp register
loop        SUBS      r4, r4, #1    ; decrement next multiplier
            MULNE     r6, r4, r6    ; perform multiply
            BNE       loop          ; go again if not complete
stop        B         stop
            END
```

   a) Using the Disassembly window (Keil μVision4), write out the first six machine codes (32-bit instructions) for the above program in hex format.
   b) Change the value in register r6 at the start of the program to 12. What value is in the register r6 when the code terminates? Verify that this hex number is correct.

2) Show three different ways to clear all the bits in register r12 to zero. You may not use any registers other than r12.

3) What is another way of writing the following line of code?

```
MOV PC, LR
```

4) Use an assembler directive to assign register r6 to the name bouncer.

5) Create a mask (bit pattern) in memory using the DCD directive and the SHL and OR operators for the following cases.

   a) The upper two bytes of the word are 0xFFEE and the least significant bit is set.
   b) Bits 17 and 16 are set, and the least significant byte of the word is 0x8F.
   c) Bits 15 and 13 are set.
   d) Bits 31 and 23 are set.

6) What instruction puts the ASCII representation of the character “R” in register r11?

7) Give the directive to reserve a block of zeroed memory, holding 40 words and labeled coeffs.

EEE3002 Microprocessors Tutorial 4 (2014 Sem 1)

1) Describe the contents of register r13 after the following instructions complete, assuming that memory contains the values shown below. Register r0 contains 0x24, and the memory system is little-endian.

Address	Contents
0x24	0x06
0x25	0xFC
0x26	0x03
0x27	0xFF

   a) LDRSB r13, [r0]
   b) LDRSH r13, [r0]
   c) LDR r13, [r0]
   d) LDRH r13, [r0]

2) Calculate the effective address of the following instructions if register r3 = 0x4000 and register r4 = 0x20:
   a) STRH r9, [r3, r4]
   b) LDRB r8, [r3, r4, LSL #3]
   c) LDR r7, [r3], r4
   d) STRB r6, [r3], r4, ASR #2

3) Write a program that sums word-length values in memory, storing the result in register r2. Include the following table of values to sum in your code:

```
TABLE   DCD   0xFEBBA, 0x1234, 0x8888
        DCD   0x13, 0x8080808, 0xFFFF
```

4) Assuming you have a little-endian memory system, what would register r4 contain after executing the following instructions? Register r6 hold the value 0xBEEFFACE and register r3 holds 0x8000.

```
    STR   r6, [r3]
    LDRB  r4, [r3]
```

   What if you had a big-endian memory system?

EEE3002 Microprocessors Tutorial 5 (2014 Sem 1)

1) What constant would be loaded into register r7 by the following instructions?
   a) MOV r7, #0x8C, 4
   b) MOV r7, #0x42, 30
   c) MVN r7, #2
   d) MVN r7, #0x8C, 4

2) Whenever possible, use the byte rotation scheme described in the lecture notes for the MOV instruction to generate an instruction needed to load the following constants into register r2. If it is not possible to use a MOV instruction, provide an alternative instruction that do the job.
   a) 0xA400
   b) 0x7D8
   c) 0x17400
   d) 0x1980

3) Without using the MUL instruction, give instructions that multiply register r4 by :
   a) 135
   b) 255
   c) 18
   d) 16,384

   and place your result in register r0. Your solution for each part should only have two or less instructions.

4) Write a program that converts a Celsius value in register r0 in Q0 format into Fahrenheit and stored the result in register r3 in Q15 format. Use the following relationship

$$F = 1.8C + 32$$

   where C and F are Celsius and Fahrenheit degrees. Also provide the **nearest** Fahrenheit degree in Q0 format and store the result in register r2.

EEE3002 Microprocessors Tutorial 6 (2014 Sem 1)

1. Write a assembly language program that reverses the bits in a register, so that a register containing $d_{31}d_{30}d_{29}\ldots d_1d_0$ now contains $d_0d_1\ldots d_{29}d_{30}d_{31}$.

2. Find the maximum value in a list of 32-bit values located in memory. Assume the values are in two's complement representations. Your program should have 50 values in the list.

3. Convert the following C iteration statements into assembly language. For example, in C language, the following *for* loop:

```
for (i = 10; i> 0; i--) {
.... // loop body
}
```

can be written in ARM assembly language as

```
        MOV r0, #10  ; r0 is used to store i
Loop    . . .  ;loop body
        SUBS r0, r0, #1 ; loop 10 times
        BNE loop
```

Write the assembly language equivalent for the following using r0 for i, r1 for j and r3 for k:

a) for (i=0; i<10; i++) { ...}

b)
```
for (i = 1; i < 11; i++) {   //nested loop
   for (j = 5; j > 0; j--)  {
     .....} }
```

c)
```
for (i = 18; i >= 3; i--) {
   for (j= 16; j < 40; j++) {
     for (k = 0; k < 30; k++) {
       ….}}}
```

EE3002/IM2002 Microprocessors Tutorial 7

1. Write an assembly program to construct a table of Fibonacci numbers for n = 0 to 20. The starting address of the table in memory is 0x40000000. In mathematics, the **Fibonacci numbers** or **Fibonacci series** or **Fibonacci sequence** are the numbers in the following integer sequence. Assume that the values are 32-bit data.

   $$0,\ 1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55,\ 89,\ 144,\ \ldots$$

   By definition, the first two numbers in the Fibonacci sequence are 0 and 1, and each subsequent number is the sum of the previous two. Hence, the sequence F_n of Fibonacci numbers is defined by the recurrence relation

   $$F_n = F_{n-1} + F_{n-2},$$

   with seed values,

   $$F_0 = 0,\ F_1 = 1.$$

2. Write an assembly program to examine a table (a list of 32-bit values) sequentially for a match with a search key. Store the search key as a new entry if no match is found by adding it to the end of the list (highest address).

   Assume that the first value in the list indicates the length of the list and load it to register r2 (assume =4). Register r0 initially point to the starting address of the list. It will eventually point to the address of the matched item if there is a match. Register r1 contains the search item (assume =0x9ABCDEF0). You need to include the initialization file: tut7q2_L2104.ini .

3. Write an assembly program, using a jump table that contains the addresses of 3 subroutines: DoAdd; DoSubtract; DoMultiply, to perform operation on two operands. The operation to be performed is determined by register r0: r0 = 0 for add; r0 = 1 for subtract and; r0 = 2 for multiply. The two operands to these functions are contained in registers r1 and r2. Test the program using the Keil uVision Simulator with different value of r0.
   The three subroutines are shown below.

```
DoAdd
      ADD r4, r1, r2
      BX  lr            ;return from subroutine
DoSubtract
      SUB r4, r1, r2
      BX  lr
DoMultiply
      MUL r4, r1, r2
      BX  lr
```

EE3002/IM2002 Microprocessors Tutorial 8

1. State the validity for each of the following instructions. Explain why for those that is not valid.
   a. STM r4, {r4-r6}
   b. LDM r4, {r4, r7}
   c. STMIA r5!, {r5, r4, r9}
   d. LDMDA r2, {}
   e. STMDB r15!, {r0-r3, r4, lr}

2. If register r6 holds the address 0x8000 and you execute the instruction

   STMIA r6, {r7, r4, r0, lr}

   Write the memory address now holds the value that was in registers r0, r4, r7 and lr respectively.

3. Assume that the memory and registers r0-r3 of an ARM processor appears as follows:

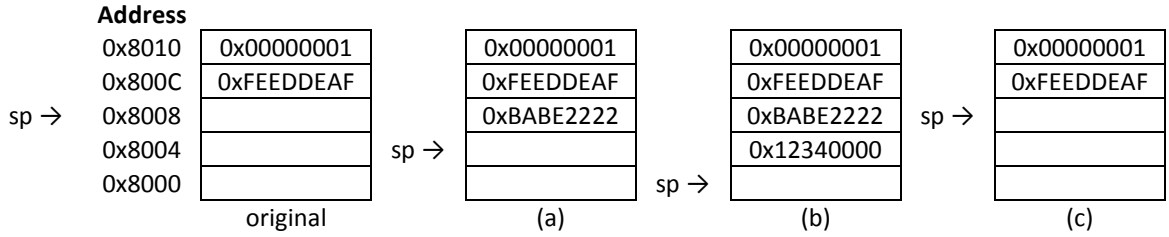
Address	
0x8010	0x00000001
0x800C	0xFEEDDEAF
0x8008	0x00008888
0x8004	0x12340000
0x8000	0xBABE0000

Register	
r0	0x13
r1	0xFFFFFFFF
r2	0xEEEEEEEE
r3	0x8000

   Give the contents of the memory and registers that have been changed after executing the instruction

   LDMIA r3!, {r0, r1, r2}

4. Suppose that a stack appears as shown in the original diagram below. Give the instruction(s) that would push or pop data so that memory appears in the order shown. In other words, what instruction would be necessary to go from original state to that shown in (a) and then (b) and then (c)?



5. Registers r0, r1 and r2 contain 0x1123, 0x2234 and 0x3356 respectively. What are the contents of each register after this sequence of instructions is executed? Suppose sp = 0x6000 before the execution. Also show the contents of the stack.

```
STMFD  sp!, {r0-r1}      ;PUSH r0, r1
STMFD  sp!, {r2}         ;PUSH r2
LDMFD  sp!, {r0, r2}     ;POP r0, r2
LDMFD  sp!, {r1}         ;POP r1
```

EE3002/IM2002 Microprocessors Tutorial 9

1. Why is it that we may get into an infinite loop when one subroutine calls another subroutine?

2. Write ARM subroutines in assembly to implement the following stack operations without using LDM or STM instructions. Assume that the stack is a Full Descending (FD) stack.
    a. myPUSH ;push content of register r0 to stack
    b. myPOP ;pop content of stack to register r0

3. Write an ARM subroutine in assembly to compute factorial of n, using a full descending stack, for each of the following methods of passing parameters. Assume that register r0 contains n (=10) and the result will be stored in register r1.
    a. Passing parameters to/from the subroutine using pass-by-register.
    b. Passing parameters to/from the subroutine using pass-by-stack.

4. Write an ARM assembly program to sum the square of 4 numbers (sum = x1*x1 + x2*x2 + x3*x3 + x4*x4). The main program should call a subroutine to add a list of number. This subroutine would then call another subroutine to perform the square operation. Test it out using Keil uVision and observe the content of registers sp, lr and pc during subroutine calls and returns. Let register r0 be the pointer to the numbers and register r1 contains the final result.

EE3002/IM2002 Microprocessors Tutorial 10

1. Based on the following exception handlers addresses, complete the partial Vector Table with the appropriate instructions at their respective exception address.

Exception	**Handler Address**
…	…
Software Interrupt	0x40000000
Undefined Instruction	0x3020
Reset	0x40001000 Assume that this 32-bit address is stored at 0xEFC

address	**Exception**	**Instruction**
0x10	…	…
0x0C	…	…
0x08	Software interrupt	?
0x04	Undefined instruction	?
0x00	Reset	?

2. Name three ways in which FIQ interrupts are handled more quickly than IRQ interrupts.

3. How many external interrupt lines does the ARM7TDMI have? If you have eight interrupting devices, how would you handle this?

4. Write an SWI handler that accepts the number 0x1234. When the handler sees this value, it should reverse the bits in register r9 and store the result in r2. The SWI exception handler should examine the actual SWI instruction in memory to determine its number.

5. When handling interrupts, why must the link register be adjusted before returning from the exception?

EE3002/IM2002 Microprocessors Tutorial 11

1. Which bit in the registers CPSR and SPSR indicate whether you are in ARM or THUMB state?
2. Explain how the branch instruction *BX rd* is used to switch between the ARM and THUMB states.
3. Write the THUMB instructions that are equivalent to the following ARM instructions.
    a. ADDS r0, r3, r2, LSL #2
    b. LDR r1, [r4, r5, LSL #2]

4. The following program fragment is written in ARM code. Determine what it is doing and compute the code size of this fragment. Convert it into a THUMB program fragment and also determine its code size.

```
        ;IN:    r0 (value) , r1 (divisor)
        ;OUT: r2 (remainder), r3 (quotient)
        CODE32
        MOV     r0, #10
        MOV     r1, #3
        MOV     r3, #0
loop    SUBS    r0, r0, r1
        ADDGE   r3, r3, #1
        BGE     loop
        ADD     r2, r0, r1
```

5. Write an assembly program that will call a subroutine (written in THUMB code) with register r0 = 5 and register r1 = 3. The THUMB subroutine computes the sum of the square of the values stored in register r0 and register r1. It should place the result in register r2. You should write code (veneer) that will enable the switch to different states and also ensure that any other registers used in the subroutine are preserved upon entry and restored before return.

EE3002/IM2002 Microprocessors Tutorial 12

1. Write a short C program that declares 3 integers called x, y and z and determine the maximum value among them. Print out this maximum value to UART0. To test the program, let x = 9, y = 4 and z = 12.
    a. Write the maximum program as an inline assembly function.
    b. Write the maximum program as an embedded assembly function.

2. Below is a main program written in C. It calls a subroutine to append two strings into one. Write the subroutine in a separate ARM assembly file. Test the program using the Keil µVision 4 simulator.

```
#include <stdio.h>
#include <LPC21xx.H>                    /* LPC21xx definitions                  */
extern void Init_serial(void);
extern void appendStr(const char *s1, const char *s2, char *s3);

/* main program */
int main (void)  {
 const char *s1 = "Hello";
 const char *s2 = "World";
 char s3[30];                           /*s3 should be “HelloWorld” eventually */

 Init_serial();                         /* initialize the serial interface      */

 appendStr(s1,s2,s3);                   /* call the asm subroutine              */
 printf ("Appending %s with %s produces %s\n", s1, s2, s3);

 while (1) {           /* An embedded program does not stop and      */
  ;  /* ... */       /* never returns. We use an endless loop.     */
 }                     /* Replace the dots (...) with your own code. */
}
```