# CMPE 364 (L51)
# Midterm Exam 1
**Spring 2017**
**March 30th, 2017**

**Instructions**: Read carefully through the entire exam first, and plan your time accordingly. Note the relative weights of each segment. Make sure that you have all pages of the exam. If you are missing any pages, inform the instructor immediately.

During this exam you may use the book *Introduction to Microprocessor Based Systems Using the Arm Processor* by Kris Schindler as well as any notes handwritten by you. The use of any other reference material will result in the confiscation of your exam and you receiving a score of 0. No electronic devices are allowed.

Write your answers on this exam. **Do not write on the back of any pages.** If you need additional space, use the extra pages at the end of the exam. Anything written on the backside of a page will not be graded.

When answering questions that require calculations, be sure to show your work. Answers without appropriate work will not receive credit. When answering questions that request an explanation, keep your explanation short and correct. Explanations containing incorrect or irrelevant information will be marked wrong, even if correct information is also included.

When you are done, present your completed exam to the instructor at the head table. If leaving before the exam period is concluded, please leave as quietly as possible as a courtesy to your neighbors.

**TL;DR:**

- **You can use the book and handwritten notes, but nothing else.**
- **Don't write on the back of any pages, use the extra pages at the end.**
- **Show your work or you won't get points.**
- **Keep justifications concise and correct.**

**Name:**

**Student ID Number:**

**Signature:**

1. Short answer

   Answer the follow short questions. For answers requiring an explanation, use 1-2 sentences. For answers requiring calculations, show your work.

   (a) (5 points) What is the value of register r3 after the following instruction has executed? Assume that r1 and r2 contain `0x5D69843B` and `0xA9C05302` respectively.

   `ADD r3, r2, r1`

   **Solution:** `0x0729D73D`

   (b) (5 points) When including immediate data as a src with a MOV instruction, the value of that data is restricted. The book describes this restriction by saying, "An immediate data value must be able to be represented as an 8-bit value that can be rotated an even number of positions." What is the cause of this restriction? (i.e., Why is this the case?)

   **Solution:** An encoded ARM instruction takes 32-bits, of which 8-bits can be used to store the immediate value. This means an immediate value must fit in 8-bits. However, those 8-bits can be passed through the barrel shifter, meaning they can be shifted around. An additional 4-bits is used to store the number of shifts to apply, and 4-bits can be used to store any even number from $[0 - 30]$. This is the reason for the limitation.

2. Memory

(a) (5 points) Consider the following map of memory. What are the contents of registers r2 and r4 after the execution of the following instruction? Assume that r2 and r4 initially contain `0x01596273` and `0x80084020` respectively. Also assume that processor is configured in little-endian mode.

`LDR r2, [r4], #4`

| Address | Contents |
|---|---|
| 0x80084024 | 0x12 |
| 0x80084023 | 0x38 |
| 0x80084022 | 0xc6 |
| 0x80084021 | 0x82 |
| 0x80084020 | 0x63 |
| 0x8008401F | 0x8d |
| 0x8008401E | 0x7b |
| 0x8008401D | 0x47 |

**Solution:**
r4 = `0x80084024`
r2 = `0x38c68263`

(b) (5 points) Consider the following map of memory. Determine the contents of registers r2, r3, r4, and `sp` after the execution of the following instruction, given an initial value of `0x80084018` for `sp`.

`LDMFA sp!, {r4-r2}`

| Address | Contents |
|---|---|
| 0x80084024 | 0x00111100 |
| 0x80084020 | 0x44555544 |
| 0x8008401C | 0x88999988 |
| 0x80084018 | 0xCCDDDDCC |
| 0x80084014 | 0x11333311 |
| 0x80084010 | 0x22444422 |
| 0x8008400C | 0xAACCCCAA |

**Solution:**
r4 = `0xCCDDDDCC`
r3 = `0x11333311`
r2 = `0x22444422`
sp = `0x8008400C`

3. Simple Coding

   Consider the following short coding questions.

   (a) (5 points) Rewrite the following sequence of instructions to be more efficient (i.e., use less instructions):

```
start:  add r1, r1, #1
        cmp r1, #5
        bge end
        b start
end:    swi 0x11
```

**Solution:**

```
start:  add r1, r1, #1
        cmp r1, #5
        blt start
        swi 0x11
```

   (b) (10 points) Write a short program which implements the following line of high level language pseudocode. Assume all source registers may be used as scratch registers.

```
r0 := (r3 * 63) / 16 + (r4 + 15) / 8 - r5
```

**Solution:**

```
mov r5, #63
mul r3, r5, r3
mov r3, r3, asr #4
add r4, r4, #15
mov r4, r4, asr #3
add r0, r3, r4
sub r0,r0, r5
```

4. (15 points) Strings

   Write a subroutine called `string_compare` that compares two strings. Note that a string is represented as a zero-terminated array of bytes. (This means that you know the string ends when you read a byte with value 0 from the array.) If the two strings are equal, return 0. Otherwise, return a non-zero number. Your subroutine should follow the conventions specified by the AAPCS. Your answer should include only the subroutine.

   Parameters:

   - Address of string 1.
   - Address of string 2.

   Returns:

   - 0 if the strings are equal, non-zero otherwise.

**Solution:**

```
; r4 is string 1
; r5 is string 2
string_compare:
        stmfd sp!, {r4, r5} ; Save these so I can restore them later

        loop:   ldrb r1, [r4], #1 ; load byte from string 1
                ldrb r2, [r5], #1 ; load byte from string 2
                cmp r1, r2        ; compare the bytes
                bne fail          ; fail if they don't match
                cmp r1, #0        ; Check if the strings are done
                bne loop          ; if not done, go again
                mov r0, #0        ; success! match found, so return 0
                b end             ; go to the end to return
        fail:   mov r0, #1        ; fail, return 1
        end:    ldmfd sp!, {r4, r5} ; restore r4 and r5
                bx lr             ; actually return
```

## End of Exam.

Extra Work Page 1.
If you need more space to write your answer to a problem, then write it here and make a note on the problem's main page referring to this space.

Extra Work Page 2.
If you need more space to write your answer to a problem, then write it here and make a note on the problem's main page referring to this space.