Name (Last, First): _____

This exam consists of 5 questions on 8 pages. Be sure you have the entire exam before starting. The point value of each question is indicated at its beginning; the entire exam is worth 100 points. Individual parts of a multi-part question are generally assigned approximately the same point value; exceptions are noted. For this exam you are allowed to bring a single page of notes (front and back). You may NOT share material with another student during the exam. Use of electronic devices is not allowed.

Be concise and clearly indicate your answer. Presentation and simplicity of your answers may affect your grade. Answer each question in the space following the question. If you find it necessary to continue an answer elsewhere, clearly indicate the location of its continuation and label its continuation with the question number and subpart if appropriate.

You should read through all the questions first, then pace yourself.

The questions begin on the next page.

Problem	Possible	Score
1	20	
2	20	
3	20	
4	20	
5	20	
Total	100	

1. (_____/20 points)

   Short answer questions

   (a) (5 points) How many bytes of memory are needed to store the C string `"Hello ARM\n"`?

   (b) (5 points) On a 32-bit machine how many bytes are in a word? How about an int? How about an (int *)?

   (c) (5 points) What is the 8 bit 2's complement representation (binary) of -17? (Show your work)

   (d) (5 points) Which registers are updated by the ARM BL (branch and link) instruction? Explain what the BL instruction does and is used for in ARM assembly programs.

2. (_____/20 points)

C and Number Conversion

In Project01 you wrote a number converter command line utility. Recall that you wrote a program that converted between different number representations. Now consider converting octal representation (base 8) into decimal representation. Note that the digits in octal are limited to 0 through 7. Here is a table showing the first 24 conversions.

Octal	Decimal	Octal	Decimal	Octal	Decimal
00	0	10	8	20	16
01	1	11	9	21	17
02	2	12	10	22	18
03	3	13	11	23	19
04	4	14	12	24	20
05	5	15	13	25	21
06	6	16	14	26	22
07	7	17	15	27	23

Write a C function that takes an octal string (of arbitrary length) and returns a decimal value (int). Here is the C prototype:

```
int octal_to_int(char *octal_str);
```

Note you cannot use a function like `atoi()` or any other C library function. Here are some hints:

To get the int value from an ASCII char, you can simply subtract `'0'`. For example, `(int) '4' - '0' = 4`.

Here is the formula to convert a 3 digit octal number `(d2d1d0)` into decimal (`n**p` means n to the power of p):

```
(d2 * 2**6) + (d1 * 2**3) + (d0 * 2**0)
```

For example, here the formula for converting the octal value 173 to decimal:

```
(1*64) + (7*8) + (3*1) = 123
```

Your solution does not need to handle the case where the given octal number is too big to fit into an int value. You may put your solution on the next page if you need more space.

Space for your solution to problem 2 if needed.

3. (_____/20 points)

ARM Assembly Part 1

Consider the following ARM assembly code snippets. For each snippet you will be given initial values of registers and/or addresses of instructions. You will be asked to determine the values of one or more registers after the execution of the snippet. Show the values of each register used next to each instruction. For example:

```
mov r0, #1      ; r0 <- 1
add r0, r0, r0  ; r0 <- 1 + 1 = 2
```

You must show your work to get credit.

(a) (5 points) Assume `r0 = 1, r1 = 2`. What is the value of r2 after executing this snippet?

```
add r0, r0, r1
add r0, r0, #1
sub r2, r0, #2
mul r2, r2, r0
```

(b) (5 points) Assume `r0 = 0, r1 = 1`. What is the value of r0 after executing this snippet?

```
    add r0, r0, #1
    cmp r0, r1
    bne foo
    add r0, r0, #1
foo:
    add r0, r0, #1
```

(c) (5 points) Assume `r0 = 0, r1 = 1`. What is the value of r0 after executing this snippet?

```
sub sp, sp, #4
str r1, [sp]
sub sp, sp, #4
mov r1, #4
ldr r0, [sp, r1]
```

(d) (5 points) Assume `r0 = 0, r3 = 3`. What is the value of r0 after executing this snippet? How many instructions are executed in the snippet below?

```
    mov r2, r0
loop:
    cmp r2, r3
    beq loopend
    add r2, r2, #1
    b loop
loopend:
    mov r0, r2
```

4. (_____/20 points)

ARM Assembly

Consider the following ARM assembly function. Note that for the ldr and str instructions that the target address `[r0, r1, LSL #2]` means to multiple r1 by 4 and add this value to the r0 base address.

```
foo:
        push {r4, r5}
        sub sp, sp, r1
        mov r3, #0
loop1:
        cmp r3, r1
        beq loop2_init
        ldr r4, [r0, r3, LSL #2]
        str r4, [sp, r3, LSL #2]
        add r3, r3, #1
        b loop1
loop2_init:
        mov r3, #0
        mov r5, r1
        sub r5, r5, #1
loop2:
        cmp r3, r1
        beq loop2_end
        ldr r4, [sp, r5, LSL #2]
        str r4, [r0, r3, LSL #2]
        add r3, r3, #1
        sub r5, r5, #1
        b loop2
loop2_end:
        add sp, sp, r1
        pop {r4, r5}
        bx lr
```

(5 points) What is the C prototype for this function?

(5 points) Explain in English what this function does.

(5 points) Explain how the stack is used in this function.

(5 points) Write the equivalent C version of this function.

5. (_____/20 points)

Recursion

Consider the following recursive C function:

```
int rec_sum(int *a, int len)
{
    if (len == 0) {
        return 0;
    } else {
        return a[0] + rec_sum(a + 1, len - 1);
    }
}
```

Write the equivalent ARM assembly version of this function. Note that your ARM version must be recursive and you must following the ARM register usage conventions, that is, proper usage of callee saved and caller saved registers.

Continue your answers here if necessary.